VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

PROGRAMOVÁNÍ ROBOTICKÉHO RAMENE POMOCÍ CHATGPT

PROGRAMMING A ROBOTIC ARM WITH CHATGPT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JOSEF KUBA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. ZDENĚK MATERNA, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

153794

Ústav:	Ústav počítačové grafiky a multimédií (UPGM)
Student:	**Kuba Josef**
Program:	Informační technologie
Název:	**Programování robotického ramene pomocí ChatGPT**
Kategorie:	Uživatelská rozhraní
Akademický rok:	2023/24

Zadání:

1. Seznamte se s možnostmi ChatGPT a OpenAI API.
2. Proveďte rešerši existujících řešení pro programování robotů pomocí přirozeného jazyka.
3. Navrhněte řešení umožňující programování manipulačních úloh pomocí ChatGPT.
4. Navržené řešení implementujte.
5. Proveďte kvalitativní uživatelské testování.
6. Vytvořte video prezentující motivaci, cíle a výsledky vaší práce.

Literatura:

- Koubaa, Anis. "ROSGPT: Next-Generation Human-Robot Interaction with ChatGPT and ROS." (2023, preprint), dostupné z https://www.preprints.org/manuscript/202304.0827.
- Li, Chen, et al. "Bringing a natural language-enabled virtual assistant to industrial mobile robots for learning, training and assistance of manufacturing tasks." *2022 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2022.
- Bimbatti, Giorgio, Daniela Fogli, and Luigi Gargioni. "Can ChatGPT Support End-User Development of Robot Programs?." 9th International Symposium on End-User Development (IS-EUD 2023), workshop proceedings.

Při obhajobě semestrální části projektu je požadováno:

- Hotové body 1-3, rozpracovaný bod 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Materna Zdeněk, Ing., Ph.D.**
Vedoucí ústavu:	Černocký Jan, prof. Dr. Ing.
Datum zadání:	1.11.2023
Termín pro odevzdání:	9.5.2024
Datum schválení:	9.11.2023

Abstrakt

Cílem této bakalářské práce je vývoj virtuálního asistenta, který umožňuje uživatelům bez pokročilých technických znalostí efektivně ovládat robotické rameno. Práce využívá technologii ChatGPT spolu s funkcí „function calling“ pro generování příkazů pro API robotického ramene na základě uživatelských vstupů. Důraz je kladen na vývoj a testování vhodných vstupů pro ChatGPT (prompt engineering), s cílem vytvořit intuitivní a uživatelsky přívětivé rozhraní. Testování s uživateli odhalilo možnosti pro zlepšení a poskytlo cennou zpětnou vazbu pro další vývoj. Uživatelé byli schopni bez větších obtíží vytvářet jednoduché programy pro manipulaci s objekty. Výsledky ukazují, že vytvoření takového asistenta je možné a že hlavní výzvou je zadání správně navrženého systémového vstupu pro správné generování kódu. Práce také porovnává výkon a efektivitu ChatGPT verzí 3.5 Turbo a 4, přičemž zdůrazňuje význam výběru vhodné verze pro konkrétní aplikace.

Abstract

This bachelor thesis aims to develop a virtual assistant that allows users without advanced technical knowledge to effectively control a robotic arm. The thesis uses ChatGPT technology along with „function calling“ to generate commands for the robotic arm API based on user input. The focus is on developing and testing appropriate inputs for ChatGPT (prompt engineering), to create an intuitive and user-friendly interface. Testing with users revealed opportunities for improvement and provided valuable feedback for further development. Users were able to create simple object manipulation programs without much difficulty. The results show that the creation of such an assistant is possible and that the main challenge is to specify the correctly designed system input for proper code generation. The paper also compares the performance and efficiency of ChatGPT versions 3.5 Turbo and 4, emphasizing the importance of choosing the appropriate version for a particular application.

Klíčová slova

LLM, ChatGPT, function calling, systémový vstup, prompt engineering, NLP, programování robotů

Keywords

LLM, ChatGPT, function calling, system prompt, prompt engineering, NLP, robot programing

Citace

KUBA, Josef. *Programování robotického ramene pomocí ChatGPT*. Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Zdeněk Materna, Ph.D.

Programování robotického ramene pomocí ChatGPT

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Zdeňka Materny Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Josef Kuba
7. května 2024

Poděkování

Rád bych poděkoval vedoucímu práce, panu Ing. Zdeňku Maternovi, Ph.D., za jeho cenné vedení, podporu a směřování, které mi byly velkou pomocí během psaní této práce a při vývoji aplikace. Dále bych chtěl poděkovat svým přátelům a rodině, kteří mi pomohli otestovat aplikaci a poskytli mi cennou zpětnou vazbu.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V rychle se vyvíjejícím technologickém světě, kde automatizace a robotika stále více prostupují naše pracovní a osobní životy, se objevuje stále větší potřeba znalostí v oblastech jako je informatika, programování a robotika. S nástupem pokročilých jazykových modelů, jako je ChatGPT a Gemini, se mají potenciál stát přístupnějšími a méně náročnými pro širokou veřejnost.

Tato bakalářská práce se zaměřuje na ověření možnosti inovativního přístupu k programování robotického ramene pomocí virtuálního asistenta založeného na službě ChatGPT. Hlavním cílem je umožnit uživatelům bez hlubších znalostí v programování nebo robotice efektivně a intuitivně ovládat robotické rameno. Jelikož takových řešení existuje zatím minimum, tato práce slouží nejen k ověření použitelnosti, ale také jako studie proveditelnosti. Asistent umožňuje uživatelům formulovat příkazy v běžném jazyce, které jsou následně převedeny na HTTP požadavky pro API robotického ramene. To zahrnuje širokou škálu funkcí, především pro manipulaci s objekty. Důraz je kladen na vytváření a testování efektivního systémového vstupu pro ChatGPT a na analýzu jeho schopnosti generovat funkční kód pro robotické rameno. Práce také zkoumá rozdíly a použitelnost různých verzí ChatGPT (3.5 turbo a 4) a jejich vliv na kvalitu a efektivitu výstupů.

V teoretické části práce 2 jsou detailně popsány klíčové aspekty ChatGPT, které jsou nezbytné pro efektivní fungování asistenta, porozumění prostředí, hardware použitého během vývoje a testování. Průzkum existujících řešení, které využívají ChatGPT je podrobněji popsáno v kapitole 3. Následuje kapitola o návrhu aplikace 4, včetně aspektů systémového vstupu, uživatelských nápověd a vizualizace odpovědí. Implementace, vysvětlená v kapitole 5, zahrnuje podrobný popis jednotlivých modulů a jejich interakce. Závěrečná kapitola se věnuje uživatelskému testování 6 a porovnání různých verzí ChatGPT, kde jsou zdokumentovány a diskutovány zjištěné nedostatky a možná vylepšení.

Kapitola 2

ChatGPT

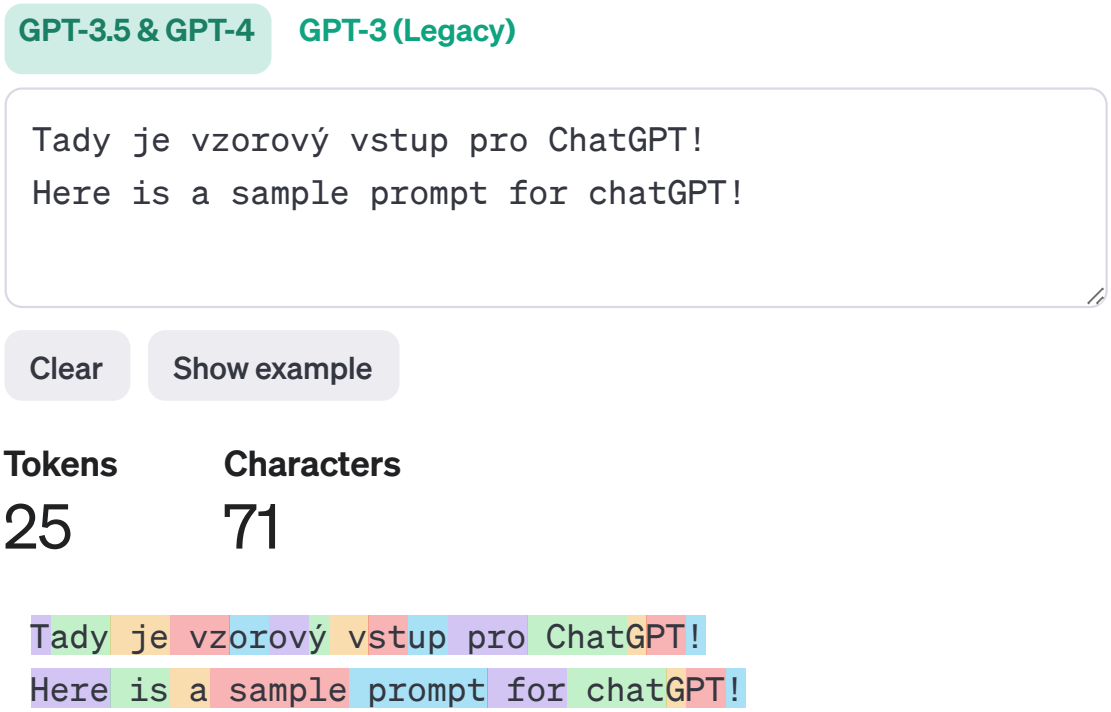
V této kapitole se soustředím na průzkum možností, které nabízí ChatGPT, a na výběr nástrojů, které byly využity v rámci této práce. Ačkoliv je práce primárně zaměřená na využití ChatGPT, je pro pochopení širšího kontextu důležité také prozkoumat jiná řešení v oblasti programování robotického ramene. Dále se věnuji výběru programovacího jazyka Python, který byl zvolen pro realizaci tohoto projektu.

2.1 Princip

ChatGPT je velký jazykový model (LLM - large language model) vyvinutý OpenAI, založený na architektuře transformátoru. Byl předtrénován na velkém množství strukturovaných textových dat (knihy, články, webové stránky atd.), což mu umožňuje generovat srozumitelný a souvislý text, odpovídat na otázky, a vykonávat další jazykové úkoly.

Jádro modelu tvoří transformátor, který zpracovává vstupní data rozdělená na tokeny viz obrázek 2.1 — jednotky textu jako jsou slova nebo části slov. Tyto tokeny jsou převedeny na mnohodimenzionální vektory a procházejí transformačními bloky, kde se na základě vzájemných vztahů a kontextu aktualizují. Model předpovídá nejpravděpodobnější následující token založený na předchozím kontextu, což ChatGPT umožňuje generovat text. K regulaci rozmanitosti generovaného textu se využívá parametr „temperature“, který ovlivňuje pravděpodobnostní rozdělení předpovědí. Nižší hodnoty vedou ke konzervativnějším a předvídatelnějším textům, zatímco vyšší hodnoty podporují větší kreativitu a diverzitu ve výstupu [1].

Největším problémem LLM je už z principu jejich tendence generovat chybné nebo zavádějící odpovědi, které mohou znít rozumně, ale nejsou přesné. Na rozdíl od vyhledávače, který by v případě nedostatku informací nevrátil žádné výsledky, ChatGPT může generovat odpovědi, které jsou pravděpodobné, ale nesprávné. Například, když je požádán výsledek sportovního utkání, může ChatGPT správně identifikovat zápas v příslušný datum, kdy se zápas odehrál, ale může začít „halucinovat“ – vymýšlet si pravděpodobně znějící, ale nepravdivé informace nebo detaily, jako jsou skórující hráči [4].



Obrázek 2.1: Rozdělení vstupu na tokeny [9].

2.2 Rozdíly konkrétních verzí

Při porovnání modelů GPT-3.5 Turbo a GPT-4 je klíčové pochopit, jak různé technické specifikace ovlivňují jejich výkon, schopnosti a použitelnost. Tyto aspekty zahrnují velikost modelu (počet parametrů), velikost kontextového okna, velikost slovníku a architektonické inovace, které přinášejí zásadní rozdíly mezi oběma modely. Dále vycházím z [1] a [2].

1. **Velikost modelu (počet parametrů)** GPT-3.5 Turbo má srovnatelný počet parametrů s ostatními verzemi GPT-3, což zahrnuje optimalizace pro zvýšení rychlosti bez výrazného navýšení počtu parametrů. Méně parametrů může znamenat rychlejší reakce při nižších výpočetních nárocích, ale potenciálně méně hluboké porozumění složitým dotazům.

   GPT-4 využívá výrazně větší počet parametrů, což mu umožňuje lépe modelovat jemnější nuance jazyka a efektivněji řešit komplexní úkoly. Větší počet parametrů obecně znamená lepší schopnost generalizace a sofistikovanější zpracování jazyka.

2. **Velikost kontextového okna** GPT-3.5 Turbo a GPT-4 se liší ve schopnosti uchovávat kontext v dlouhých textech, což má zásadní význam pro složitější úkoly, jako je psaní článků nebo tvorba souvislých dialogů. Zatímco GPT-4 obvykle disponuje větším kontextovým oknem, umožňující mu efektivněji zpracovávat a uchovávat kontext v delších pasážích textu, v rámci tohoto projektu byla z ekonomických důvodů použita jeho cenově dostupnější verze s menším kontextovým oknem než verze GPT-3.5 Turbo.

3. **Velikost slovníku** Oba modely mají rozšířené slovníky, které jim umožňují lépe zpracovávat a generovat bohatší a technicky náročnější jazyk. Větší slovník umožňuje modelům pokrýt širší spektrum jazykových konstrukcí a odborných termínů.

4. **Architektonické inovace** GPT-3.5 Turbo zahrnuje optimalizace pro efektivní zpracování, což umožňuje modelu rychleji reagovat na dotazy uživatelů.

   GPT-4 přináší nové architektonické prvky a algoritmy, které vylepšují učení a minimalizují chyby způsobené předsudky nebo přetrénováním. To vede k vylepšené schopnosti generalizace a robustnějšímu zpracování dat.

Model	**Popis**	**Velikost kontextu**	**Trénovací data**
gpt-3.5-turbo-0125	Aktuálně poslední model GPT-3.5 Turbo	16,385 tokenů	Do září 2021
gpt-4-0613	Snapshot modelu gpt-4 z 13. června 2023	8,192 tokenů	Do září 2021

Tabulka 2.1: Používané modely GPT [6].

Model	**Cena za vstup**	**Cena za výstup**
gpt-3.5-turbo-0125	$0.50 / 1M tokenů	$1.50 / 1M tokenů
gpt-4	$30.00 / 1M tokenů	$60.00 / 1M tokenů

Tabulka 2.2: Ceny za použití API konkrétních modelů GPT [8]

2.3 Možnosti OpenAI API

V této části práce prozkoumáme tři klíčové komponenty: Actions (plugins), Function Calling a nově Assistant. Tyto komponenty představují různé způsoby, jakými může ChatGPT interagovat s vnějším světem a rozšířit své využití pro specifické účely. V dalších sekcích podrobně rozebereme každou z těchto možností a vysvětlíme jejich funkce pro vývoj inteligentních aplikací. Informace o těchto možnostech jsou shromážděny z oficiálních zdrojů OpenAI [7], které poskytují podrobné technické specifikace a příklady použití.

2.3.1 Plugins

Technologie plugins (Actions) od OpenAI umožňuje modelům ChatGPT integrovat a komunikovat s různými externími API, což výrazně rozšiřuje jejich schopnosti. Tato funkcionalita je zásadní pro vytvoření interaktivních aplikací, které mohou využívat data a služby přístupné přes internet. Zde je detailnější popis, jak tato technologie funguje:

1. Definice API a koncových bodů: Prvním krokem je definice a specifikace API, které plugin bude používat. To zahrnuje určení konkrétních koncových bodů, které model bude volat. Tyto koncové body mohou zahrnovat operace jako získání dat o počasí, akciových trzích, nebo jiné informace dostupné prostřednictvím webu.

2. Registrace pluginu u OpenAI: Po definici funkcí API je nutné plugin zaregistrovat u OpenAI. Tento krok zahrnuje poskytnutí metadata pluginu, jako jsou jeho název,

popis a konfigurace koncových bodů. OpenAI tuto registraci vyžaduje, aby mohla správně integrovat plugin do své platformy.

3. Vývoj a integrace kódu: Následně je třeba vyvinout kód, který umožní ChatGPT interagovat s externím API. Tento kód se stará o zasílání požadavků na API a zpracování odpovědí, které se vrátí zpět do modelu ChatGPT. Kód musí zahrnovat správné zacházení s daty, jako je například parsování, validace a bezpečnostní opatření.

4. Ovládání kontextu: Jednou z klíčových funkcí plugins je schopnost udržet kontext během interakcí. To znamená, že ChatGPT si pamatuje předchozí výměny informací s uživatelem a může použít tuto historii pro informovanější a relevantnější odpovědi.

2.3.2 Function calling

Oproti *Actions* (dříve Plugins), možnost Function Calling umožňuje odesílat spolu s kontextem zpráv také specifikace dostupných funkcí, které si model může vyžádat k provedení. Tyto specifikace jsou zasílány pomocí parametru *functions* (v novější verzi označovaného jako *tools*). To, zda a která funkce bude provedena, je ve výchozím nastavení na rozhodnutí ChatGPT. Přidáním volitelného parametru při odesílání zprávy lze ovšem ovlivnit, jestli a případně která funkce bude spuštěna. Tato možnost není omezena pouze na přímé spuštění funkce ale lze ji také využít i pro „vynucení“ formátu, v jakém má být výsledek vrácen. Dále je zde možnost paralelního volání funkcí, což je užitečné v případech, kdy je třeba současně provádět více úkolů s delším zpracováním. V rámci této práce tyto dvě poslední možnosti nejsou implementovány, ale pro budoucí rozšíření a pokračování práce je dobré mít o nich povědomí.

```
[{
    "name": "beltSpeed",
    "description": "Moves the belt in certain speed and direction.",
    "parameters": {
        "type": "object",
        "properties": {
            "direction": {
                "type": "string",
                "enum": ["forward", "backwards"],
                "description": "Type of movement"
            },
            "velocity": {
                "type": "number",
                "description": "Velocity of movement 1-50"
            }
        },
        "requiredParams": ["direction", "velocity"]
    }
}]
```

Výpis 2.1: Příklad specifikace funkce

2.3.3 Assistants

Assistant API od OpenAI umožňuje asistentům přistupovat k široké škále nástrojů a funkcí, které lze specificky nastavit a přizpůsobit tak, aby odpovídaly konkrétním potřebám a požadavkům aplikací. Asistenti mohou volat modely OpenAI s konkrétními instrukcemi, což umožňuje ladění jejich osobnosti a schopností pro specifické úkoly. Tato technologie nebyla dostupná na začátku vývoje. Její přítomnost by značně usnadnila implementaci a rozšířila by naše možnosti. Informace v této podsekci jsou čerpány z oficiální dokumentace OpenAI [7].

Přístup k nástrojům: Asistenti mohou paralelně využívat několik nástrojů, které mohou být hostovány přímo OpenAI — jako je code_interpreter nebo file_search — nebo mohou být externě hostované prostřednictvím funkcí, jako je function calling. Toto umožňuje široké možnosti integrace a rozšíření funkcionalit aplikací.

Persistentní Threads: API poskytuje možnost využívat trvalá komunikační vlákna, tzv. Threads, které usnadňují vývoj aplikací s AI tím, že uchovávají historii zpráv a automaticky ji zkracují. Když délka konverzace překročí limit pro kontextovou délku modelu, systém se snaží zprávy inteligentně zkracovat a nakonec odstraní ty, které jsou považovány za nejméně důležité.

Přístup k souborům: Asistenti mohou přistupovat k souborům v různých formátech, ať už jsou součástí jejich vytváření nebo jsou začleněny do Threads mezi asistenty a uživatele. Při používání nástrojů mohou asistenti také vytvářet soubory (například obrázky, tabulky atd.) a odkazovat na soubory, které byly pro tuto tvorbu relevantí.

2.4 Python

V této části se zaměřím na důvody, proč byl pro tento projekt zvolen programovací jazyk Python, a jaké výhody to přináší jak pro vývojáře, tak pro uživatele.

Python je ideální pro tento projekt, jelikož je hojně rozšířený a podporován velkou komunitou vývojářů. Díky své popularitě poskytuje přístup k široké škále knihoven a nástrojů, což usnadňuje rychlé prototypování a efektivní vývoj aplikací. Python je známý svou flexibilitou a rozšiřitelností, což umožňuje snadné začlenění nových funkcí a integraci s různými API, což je důležité pro tento projekt, kde interakce s robotickými systémy je nezbytná [11].

Python usnadňuje učení a práci s programováním díky své jednoduchosti a přístupnosti. Jeho syntaxe je intuitivní a zřetelná, což novým uživatelům umožňuje rychle pochopit základní koncepty a začít efektivně využívat jeho funkce pro své projekty. Tato jednoduchost a přehlednost kódu v Pythonu značně snižuje bariéru pro začínající programátory a zvyšuje celkovou efektivitu vývoje aplikací [11].

Kapitola 3

Existující řešení

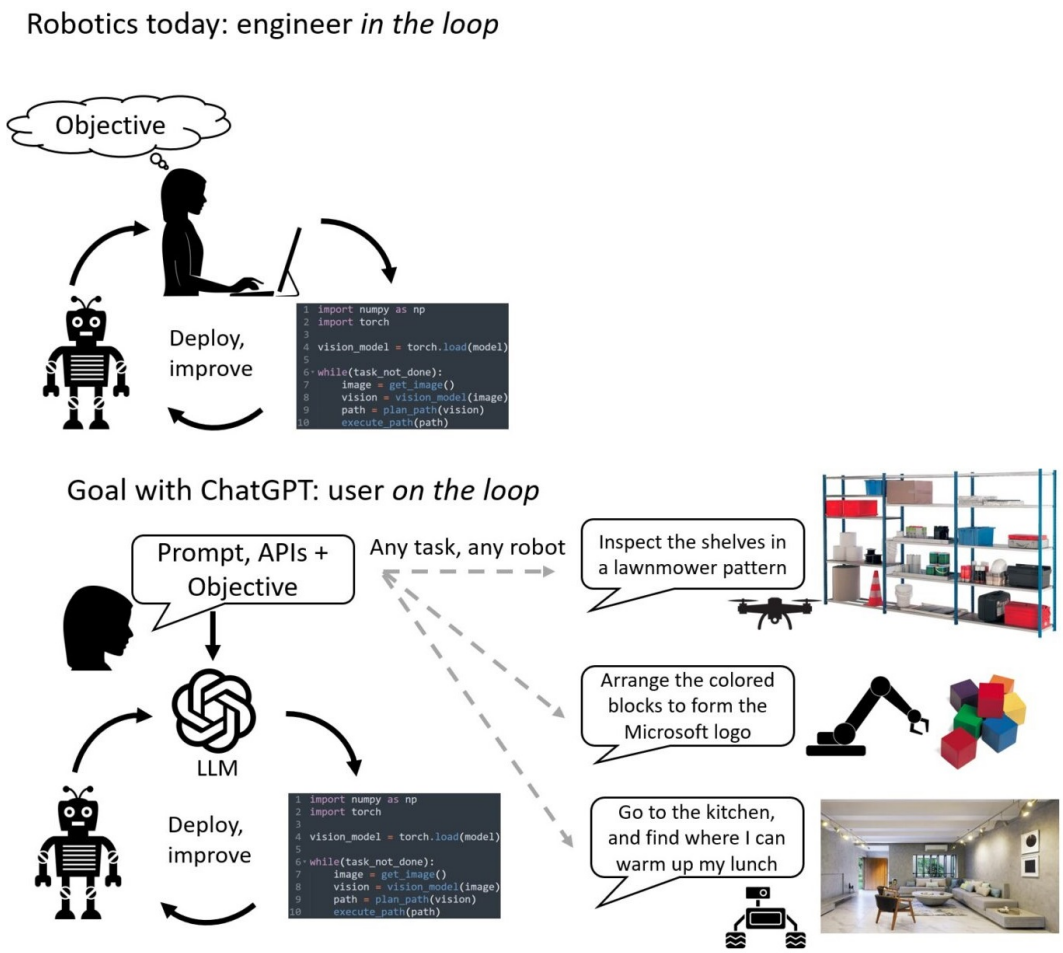
V této kapitole se zaměřím na průzkum stávajících aplikací a technologií v oblasti robotiky, které využívají umělou inteligenci a jazykové modely. Přestože aplikace technologie ChatGPT v robotice nabývají na popularitě, stále je toto pole relativně nové a existující řešení jsou omezená. Tato kapitola analyzuje různé přístupy a modely, které byly vyzkoušeny, a poskytuje přehled o tom, jak ChatGPT může být integrován do robotických systémů.

3.1 Základní řešení

Při zkoumání podobných řešení jsem narazil na tento [12] velmi raný pokus o programování a řízení průmyslových robotů s využitím jednoduchého lidského jazyka, OpenAI ChatGPT a knihovny „fanucpy“. Každý uživatelský vstup je vložen do předem připravené šablony, která upřesňuje informace a zahrnuje vzorový program. Odpovědi obsahující název knihovny „fanucpy“ jsou dále zpracovávány tak, že se kód extrahuje z bloku kódu, uloží do souboru a poté se spustí tento Python skript jako podproces. Výstupy skriptu jsou vypsány a přidány do kontextu konverzace. Toto řešení považuji za solidní základ, na kterém lze dále stavět. Ukázkový kód, který je součástí projektu, poskytuje pouze základní soubor funkcí, které může robot provádět. Současný přístup se primárně zaměřuje na vykonávání generovaného kódu bez dalších podpůrných funkcí, jako je asistence v programování nebo odpovídání na otázky uživatelů.

3.2 Principy návrhu a modelové schopnosti

Práce [13] představuje vizi, ve které netechničtí uživatelé mohou efektivně pracovat s roboty pomocí přirozeného jazyka. Současně je provoz robotů závislý na technicky znalých inženýrech, ale s využitím modelů jako je ChatGPT, se otevírá možnost, aby roboty ovládali uživatelé bez technického vzdělání viz obrázek 3.1. Interakce s roboty by se mohla stát stejně přirozenou jako používání běžných aplikací, což by umožnilo rozšíření robotických technologií do nových oblastí a průmyslových odvětví.



Obrázek 3.1: Současný vývoj v robotice obvykle vyžaduje odborníka s hlubokými znalostmi, který píše a optimalizuje kód. Cílem s ChatGPT je mít v procesu uživatele (potenciálně netechnického), který bude interagovat s jazykovým modelem prostřednictvím příkazů ve vysokoúrovňovém jazyce a bude schopen bezproblémově nasazovat různá řešení [13].

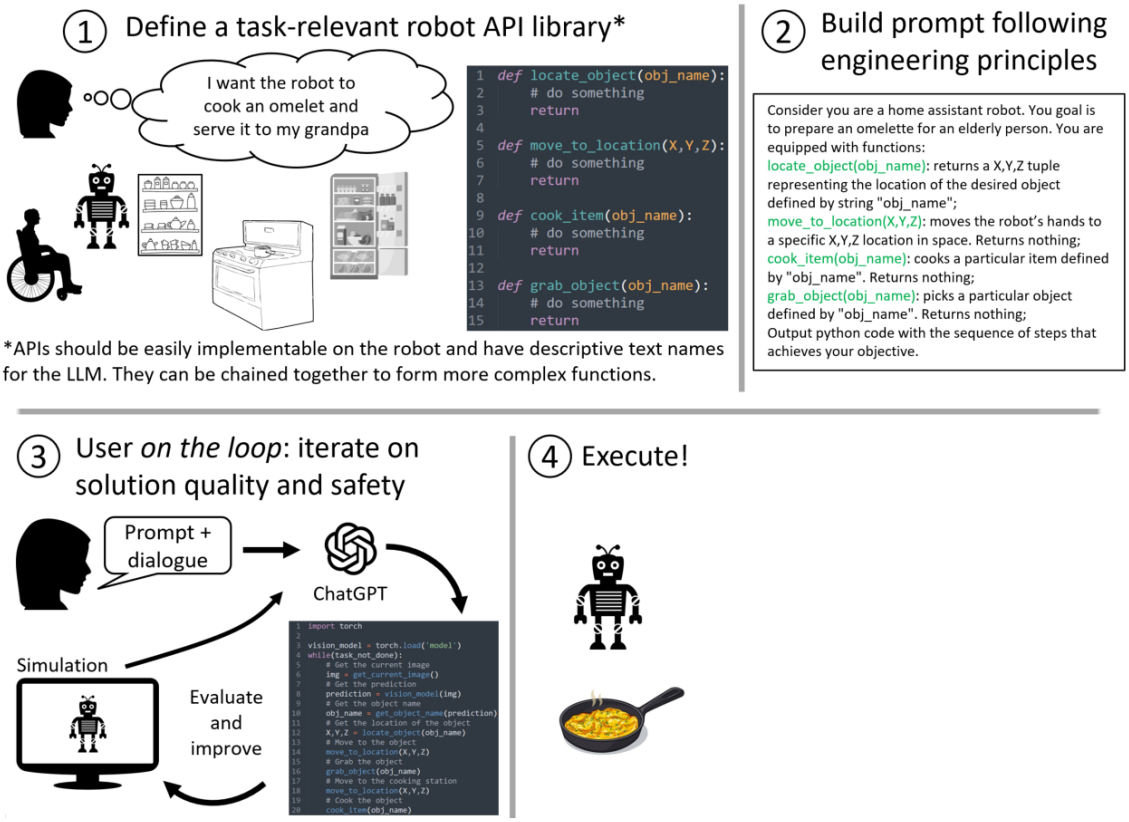
Použití jazykových modelů, jako je ChatGPT, pro práci s roboty přináší několik výzev, jako jsou poskytování kompletních a přesných popisů problémů, identifikace správné sady povolených funkcí a API, a nastavení struktury odpovědi. Tento procesní řetězec 3.2, který jsem se rozhodl adoptovat a dále rozvíjet ve své bakalářské práci, je sestaven z následujících kroků:

1. **Definice knihovny robotických funkcí na vysoké úrovni abstrakce:** Tato knihovna může být specifická pro určitý formát nebo scénář a měla by odpovídat reálným implementacím robotických systémů. Je důležité, aby byla dostatečně deskriptivně pojmenována a popsána, aby ji ChatGPT mohl snadno interpretovat.

2. **Vytvoření vstupu pro ChatGPT:** Je třeba sestavit výzvu, která jasně popisuje cíl a identifikuje sadu povolených funkcí z knihovny na vysoké úrovni abstrakce. Výzva by měla také obsahovat informace o omezeních a popis, jak by ChatGPT měl strukturovat své odpovědi.

3. **Zapojení uživatele do procesu:** Uživatel by měl být aktivně zapojen do hodnocení kódu generovaného ChatGPT, a to buď prostřednictvím přímé analýzy nebo simulace. Uživatel by měl ChatGPT poskytovat zpětnou vazbu ohledně kvality a bezpečnosti vygenerovaného kódu.

4. **Iterace a nasazení finálního kódu:** Po opakovaných iteracích a úpravách v implementaci a přizpůsobení výstupu dle uživatelských požadavků je kód nasazen.



Obrázek 3.2: Procesní řetězec návrhu aplikace [13]

3.3 Rozdělení zadání do částí

Oproti ostatním přístupům zadávání úkolů pro ChatGPT, je ten popsaný v článku [14] výjimečný tím, že rozděluje každý úkol do šesti specificky navržených vstupů. Tato pečlivě strukturovaná metodika umožňuje ChatGPT lépe pochopit podstatu úkolu a naplánovat robotické operace přizpůsobené různým prostředím. Každý z těchto šesti promptů je cíleně sestaven tak, aby maximalizoval srozumitelnost pokynů a zároveň zajistil, že výstupy budou nejen přesné, ale i přímo aplikovatelné v daném kontextu.

1. **Role ChatGPT:** Ve výzvě je ChatGPT poskytnut kontext pro plánování úkolů tím, že je vysvětlena role, kterou by měl při plánování zastávat. Aby bylo umožněno zpracování více podnětů, je přidána instrukce, aby ChatGPT počkal s odpovědí, dokud nebudou všechny výzvy zadány.

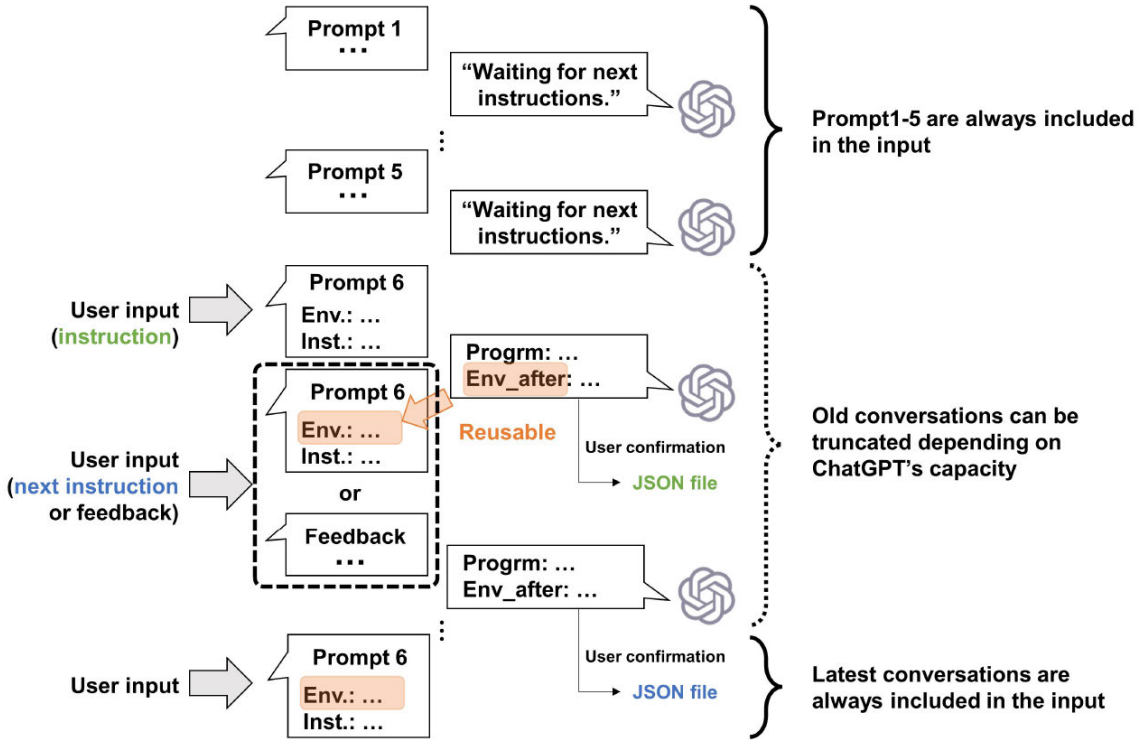
2. **Popis dostupných akcí:** Ve výzvě je specifikována sada akcí, které může robot provádět. Tato sada akcí se liší v závislosti na konkrétní aplikaci a implementaci robotického softwaru, a proto by měla být výzva přizpůsobena experimentátory podle jejich potřeb.

3. **Popis prostředí:** Tato výzva definuje pravidla pro reprezentaci pracovního prostředí. Ve specifikovaném případě jsou všechny fyzické entity kategorizovány do dvou hlavních tříd: `assets` a `objects`. `Assets` jsou nemanipulovatelné překážky, které jsou stálými prvky prostředí. Naopak `objects` zahrnují manipulovatelné objekty, s nimiž může robot interagovat. Toto rozlišení usnadňuje ChatGPT pochopení prostředí a podporuje logické usuzování o možných akcích robota. Prostorové vztahy mezi entitami jsou popsány jako stavy.

4. **Formát generovaného výstupu:** Tato výzva specifikuje formát výstupu, který generuje ChatGPT, aby usnadnila jeho integraci s dalšími systémy, jako jsou robotické řídicí systémy a programy pro vizuální rozpoznávání. Je doporučeno, aby ChatGPT vytvářel výstup ve formě Python slovníku, který lze uložit jako JSON soubor. Navíc by měl ChatGPT zahrnovat nejen sekvenci akcí robota, ale také vysvětlení jednotlivých kroků a doplňkové informace o prostředí po provedení akcí. Tyto doplňkové informace umožňují uživatelům ověřovat, zda ChatGPT správně zpracovává vstupní data.

5. **Příklad vstupů a výstupů:** V této výzvě jsou poskytnuty příklady očekávaných vstupů a výstupů. Bylo zjištěno, že poskytování většího množství příkladů pomáhá ChatGPT generovat požadovanou sekvenci, což minimalizuje úsilí, které musí uživatelé vynaložit na opravu výstupu prostřednictvím konverzací.

6. **Specifické zadání od uživatele:** V rámci procesu plánování úkolů je šestá výzva dynamicky generována úpravou šablony, kde se [INSTRUCTION] nahrazuje danou instrukcí a [ENVIRONMENT] odpovídajícími informacemi o prostředí. Uživatel musí původně poskytnout informace o prostředí odděleným procesem (např. manuální přípravou), avšak pro následné instance není tato snaha potřebná, protože lze využít aktualizované prostředí zahrnuté v posledním výstupu ChatGPT viz obrázek 3.3.



Obrázek 3.3: Celá struktura konverzace s ChatGPT při plánování úkolů [14]

3.4 Shrnutí

Průzkum existujících řešení poskytl užitečný vhled do problematiky a možností využití ChatGPT v praxi, což mi pomohlo lépe pochopit potenciální vylepšení a inovace v této oblasti. Analyzování existujících řešení výrazně usnadnilo návrh a strukturu mé aplikace. Specificky jsem využil procesního řetězce v principu návrhu a byl jsem inspirován strukturou systémového vstupu. Experimentoval jsem s rozdělením vstupu, ale nakonec jsem se rozhodl tuto metodu nezavést. Největší výzvou zůstává efektivní formulace problémů, které mají být řešeny jazykovými modely, jako je ChatGPT, tak aby byly správně interpretovány a zpracovány.

Kapitola 4

Návrh řešení

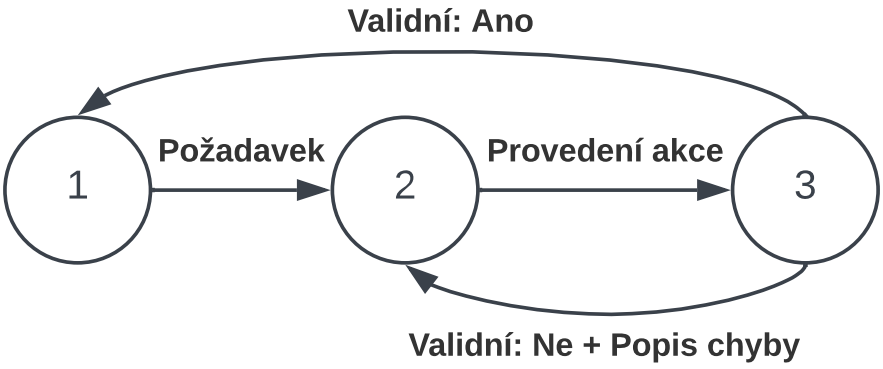
Adaptace na rychlý pokrok v technologiích je důležitá v různých odvětvích, nejen v průmyslu. I technicky méně zdatní uživatelé se nyní mohou setkávat s roboty, jejichž ovládání a nastavení pro ně může být náročné. Díky pokročilým jazykovým modelům, jako je ChatGPT, je již možné robotiku zpřístupnit širšímu okruhu lidí.

Cílem práce je zde navrhnout rozhraní, které bude sloužit jako prostředník mezi uživatelem a robotickou paží. Přestože se zaměřuji hlavně na robotické paže, tento přístup je možné aplikovat i na další druhy robotů. Robotické paže nacházejí uplatnění v různých oblastech – od výrobních linek až po gastronomii, kde automatizují opakované úkoly. Tento přístup je optimální pro realizaci krátkých a variabilních sekvenčních úkolů, které se nebudou opakovat v milionových množstvích, ale spíše v řádu vyšších desítek až stovek.

I když název práce naznačuje zaměření na „programování“, považuji za podstatné, aby ChatGPT mohl přímo interagovat s robotem. Programování zahrnuje práci s polohou robota, například rozhodování o tom, kam se má robot pohnout. Samotné načtení aktuální polohy robota je již interakcí s ním. V případech, kdy manipulace s robotem není možná přímo na místě nebo je z bezpečnostních důvodů nevhodná (například při práci s chemikáliemi), je efektivnější sdělit ChatGPT konkrétní akce, které má robot provést, místo zbytečného programování, ověřování a spouštění celého programu. To se týká zejména neopakujících se činností, kde přímý příkaz může rychle a bezpečně řešit specifické úlohy.

4.1 Uživatelsky orientovaný návrh interakce a workflow

Když se zabýváme návrhem uživatelského rozhraní pro práci s virtuálním asistentem, je zásadní si uvědomit, že interakce bude probíhat dialogově viz obrázek 4.1. Původně bylo plánováno vytvoření jednoduché konzolové aplikace, avšak s ohledem na cílovou skupinu, která zahrnuje uživatele s menšími zkušenostmi v IT, se ukázalo vhodnější rozvíjet webové grafické uživatelské rozhraní (GUI). Služba s webovým rozhraním navíc nabízí lepší možnosti pro nasazení a integraci s existujícími systémy. Webové GUI je intuitivnější a usnadňuje interakci, což je klíčové pro zvýšení přístupnosti a úspěšnosti aplikace.



Obrázek 4.1: Schéma popisující průběh práce s asistentem

Na obrázku je zachyceno, jak probíhá interakce uživatele s virtuálním asistentem. Uživatel zahájí komunikaci zadáním požadavku, který může být různorodý — od konkrétních úkolů, jako je posunutí robota o určenou vzdálenost, až po informační dotazy týkající se stavu systému nebo požadavky na sestavení programu. Následně ChatGPT zpracuje přijatý požadavek a reaguje buď provedením akce, vrácením relevantní odpovědi nebo generováním potřebného kódu. Uživatel poté vyhodnotí, zda asistent úkol zvládl uspokojivě. Pokud ano, může přejít k dalšímu úkolu; pokud ne, specifikuje, co je třeba změnit nebo opravit. Tento proces je klíčový pro efektivní a interaktivní práci s robotickým systémem a umožňuje uživatelům plynulejší a intuitivnější ovládání technologie.

4.1.1 Co by to mělo umět?

Jak jsem již dříve zmínil, mým cílem bylo, aby ChatGPT mohl přímo ovládat robotické rameno, což zahrnuje přístup ke všem funkcím robota. To znamená, že by ChatGPT měl být schopen provádět stavové funkce, jako jsou spouštění, zastavování a zjišťování aktuálního stavu. Dále by měl být schopen ovládat funkce spojené s dopravníkovým pásem, jako jsou posunutí na určitou vzdálenost nebo rychlost posunu pásu, a také základní funkce robota, jako jsou zjištění a změna pozice, kalibrace, nasávání a uvolnění přísavky pro zachycení objektů.

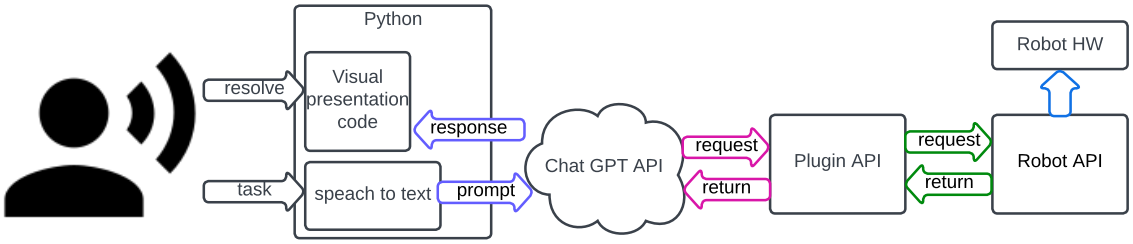
Neméně důležité jsou také funkce pro správu programů robota. Sem patří možnosti ukládání a mazání programů, načítání seznamu dostupných programů, načítání specifických programů a jejich spouštění. Tyto funkce jsou klíčové pro efektivní a flexibilní ovládání robota pomocí ChatGPT.

4.2 Technický návrh aplikace

Při prozkoumávání funkcí nabízených OpenAI API jsem hledal řešení, která by umožňovala pohodlné vzdálené provádění kódu. V návrhu asistenta jsem nekladl důraz pouze na schopnost generovat kvalitní a ideálně okamžitě spustitelný kód. Snažil jsem se také zajistit, aby asistent dokázal odpovídat na přímé příkazy a vykonávat je podle uživatelských požadavků.

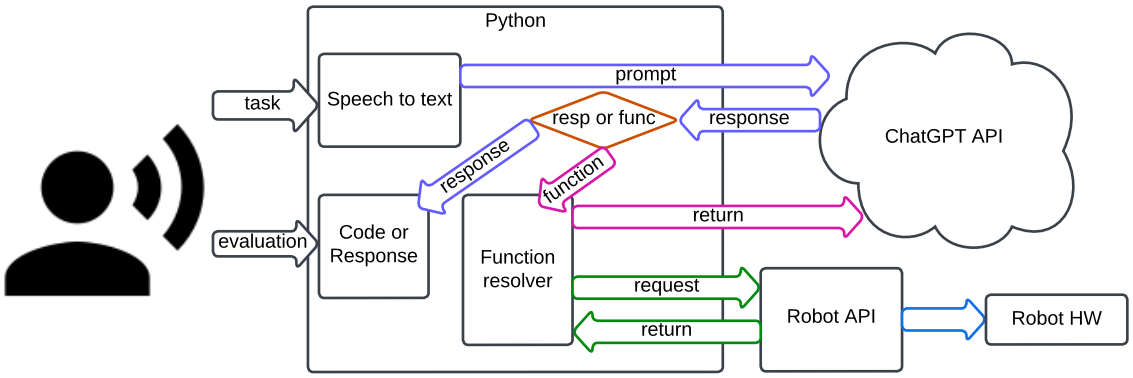
Samy o sobě však tyto funkce nestačí k zajištění plné provozuschopnosti asistenta. Pro účinné generování kvalitního kódu, který by mohl být spuštěn bez nutnosti dalších zásahů uživatele, je nezbytné vytvořit co možná nejlepší systémový vstup pro nastavení kontextu konverzace. Kromě toho jsem považoval za důležité vymyslet jednoduchou uvítací zprávu

pro uživatele, která by jim poskytla přehled o tom, jak asistent funguje a jak ho mohou efektivně využívat.



Obrázek 4.2: Blokové schéma popisující funkci aplikace s využitím pluginů

Původní návrh 4.2 aplikace zahrnoval využití technologie Plugins (actions), která byla nakonec ve finálním provedení projektu opuštěna.



Obrázek 4.3: Blokové schéma popisující funkci aplikace s využitím volání funkcí

Na blokovém schématu výše lze vidět návrh aplikace 4.3, která využívá Function calling. Z obrázku je patrné, že mluvená řeč je transformována na textový vstup. Při přijetí odpovědi systém zkontroluje, zda je součástí nějaká volaná funkce. Pokud není vyžádána žádná funkce, je odpověď jednoduše vrácena uživateli. V případě, že je požadováno provedení konkrétní akce, tato akce je provedena a následně je odeslána odpověď funkce, která obsahuje buď data nebo informaci o úspěšnosti provedení. Function resolver se tak postará o provedení akce a zajišťuje případné další předání požadavků k dalšímu zpracování. Například u funkce pro uložení programu dojde k uložení obsahu parametru funkce do souboru. Když ale potřebujeme vytvořit program využívající určitou absolutní pozici, je odeslán GET požadavek na API robota s žádostí o aktuální pozici robota.

4.2.1 Výběr technologie

Pro tuto práci jsem se rozhodl použít technologii „function calling“ místo dalších možností, protože pro mě představovala nejlepší řešení. Jedním z hlavních důvodů byla jednoduchost a flexibilita této technologie. Nevyžaduje hostování na serverech, což znamená, že může být provozována lokálně, což usnadňuje rychlejší implementaci a snižuje závislost na externích službách. Toto řešení mi umožnilo efektivněji pracovat s funkcemi přímo v rámci aplikace, což zlepšilo celkovou kontrolu a přizpůsobení projektu mým specifickým potřebám.

4.3 Systémový vstup

Klíčovou součástí návrhu aplikace je systémový vstup, který zásadně ovlivňuje kvalitu generovaných odpovědí a kódu ChatGPT. I když samozřejmě existují omezení daná schopnostmi modelu, adekvátním systémovým vstupem lze tyto hranice co nejvíce přiblížit ideálu. Systémový vstup byl proto napsán v angličtině, jelikož se očekává, že tato volba zajistí nejlepší srozumitelnost pro ChatGPT.

Jedním ze známých (a i doporučovaných) přístupů je nechat, aby na sebe ChatGPT vzal nějakou roli. V tomto případě asistenta, který bude ovládat a generovat kód pro robotické rameno. Letmý popis pracovního prostředí, v němž bude asistent působit, byl také součástí systémového vstupu [3], [5]. Toto zahrnovalo specifikaci, že robotické rameno je vybaveno přísavkou a bude obvykle manipulovat s molitanovými kostkami. Tato přesná specifikace umožňuje ChatGPT lépe chápat a reagovat na konkrétní scénáře.

Pozornost byla také věnována aspektu relativního pozicování. Příkladem může být požadavek typu „přesuň rameno o 5 cm vlevo“.

ChatGPT byl směrován, aby generoval kód v Pythonu, přičemž primárně využíval specifický modul „robot“. Tento popis modulu je integrován přímo do systémového vstupu. Bylo klíčové, aby model vyžadoval dodatečné informace v případě nejasných zadání, a aby se přísně držel používání specifikovaného modulu. Systémový vstup dále obsahoval upřesnění, jako je třeba použití metrických jednotek a zaokrouhlování čísel na tři desetinná místa.

Dále bylo zavedeno pravidlo, aby všechny změny v generovaném Python kódu byly vždy zobrazeny v příslušném kódovém bloku. Toto zajišťuje nejen zvýraznění celého kódu jako Python kódu, ale také vizuálně odlišuje nové změny od předešlého kódu. Tento přístup pomáhá uživatelům snadno identifikovat upravené části kódu. Součástí systémového vstupu je i vzorový kód s komentáři, poskytující další inspiraci pro ChatGPT.

Posledním krokem v nastavení systémového vstupu je výběr českého jazyka pro další komunikaci.

```
Assume you are a virtual assistant who helps users control and create
    ↪ programs for a robotic arm with four degrees of freedom, equipped
    ↪ with a suction cup. The arm base is placed approximately at 0,0,0
    ↪ coordinates and the reach of the robot is approximately 320mm. Next
    ↪ to the robot, there is also a controllable conveyor belt. Most of
    ↪ the time you will be working with cubes measuring 2.5cm side.
Your task is to provide code based on specific user requests. Do not use
    ↪ any functions other than those explicitly requested.
Regarding relative position: 'Up' means an increase in the z-coordinate, '
    ↪ Right' means an increase in the y-coordinate, and 'Towards me' means
    ↪  an increase in the x-coordinate.
The robot's position could be changed by the user. Always expect that the
    ↪ robot could have been moved.
Please provide concise responses. An introduction is not necessary. Inform
    ↪ me when you are ready to proceed.

When a user talks about a program, script, or Python, your job is to write
    ↪ code.
1. Feel free to ask any questions for clarification.
2. You need to create an instance of a robot and then use only that
    ↪ instance.
```

```
3. If necessary set up Poses from the user and store it in Class Pose.
4. Please use this module. This module is ready to use
import modules.robot as robot
Module description:
###MODULE###

Round coordinates by 3 decimal places. Use moveType JUMP as default. Use
    ↪ 100 speed and acceleration as default.
Values are in meters.
Tasks are simple so don't make it more complicated by adding function/class
    ↪ /imports.
The workflow involves you writing a program encapsulated in a code block (
    ↪ meaning ```python...```), which I will then validate or adjust. You
    ↪ must always write the whole code. The user can save and run the
    ↪ adjusted program (Don't do anything without being told to).

###CODE###

Do you see the difference between "move the robot up 5cm" and "write a
    ↪ program to move the robot up 5cm"?
For your task is very important to understand this.
To reply use max 10 words. Speak czech / Mluv česky prosím.
```

Výpis 4.1: Systémový vstup před doplněním vzorového kódu a informací o knihovně

4.3.1 Popis modulu robot

V rámci návrhové fáze práce byla věnována zvláštní pozornost vývoji modulu robot, který měl sloužit jako hlavní komponent pro interakci s API robota. Tento modul byl navržen tak, aby pracoval s API robota a zároveň poskytoval nezbytnou úroveň abstrakce pro programování robota. Hlavním cílem tohoto návrhu bylo umožnit ChatGPT snadněji překládat uživatelské příkazy do volání metod robota.

Při vývoji jsem používal rozhraní Swagger, které umožnilo efektivní analýzu a testování robotických funkcí, zajišťující přesnou specifikaci požadavků modulu a jeho správnou funkčnost.

Celkově byl návrh modulu robot zásadní pro úspěch celého projektu, protože zajišťoval, že interakce mezi ChatGPT, robotickým API a uživateli bude plynulá a intuitivní. Při návrhu tohoto modulu byla prioritou snadná čitelnost a srozumitelnost kódu, což bylo dosaženo prostřednictvím intuitivního pojmenování tříd a metod, aby byl kód čitelný i pro méně zkušené programátory. Tato praxe přispěla k tomu, že celková struktura a funkčnost modulu je přehledná a snadno pochopitelná, což usnadňuje další vývoj a údržbu kódu.

K tomu je důležité poznamenat, že popis modulu robot není staticky uložený v souboru, ale je načítán přímo z docstringů při každém spuštění aplikace. Z tohoto důvodu je nezbytné, aby kód v modulu byl důkladně dokumentován a co nejvíce aktuální. Tento přístup umožňuje flexibilní a dynamické doplňování nových funkcí, které může asistent využívat při psaní kódu. To výrazně zjednodušuje proces aktualizací a rozšíření funkcionalit modulu, čímž se dále zvyšuje efektivita a adaptabilita celé aplikace.

Vzorový program používající modul robot

V rámci projektu bylo nezbytné vyvinout vzorový program, který by sloužil jako návod pro ChatGPT v procesu generování robotických programů. Tento vzorový program je záměrně navržen s důrazem na přehlednost a obsáhlost komentářů, aby ilustroval typický přístup k programování robotických úloh. Specifickým cílem tohoto programu je prezentovat úkol, v němž robot zvedne krychli do výšky 5 cm a poté ji opět pustí.

V programu se nejprve provádí import použitého modulu, a pak následuje inicializace třídy Pose, která slouží k definování polohy krychle v prostoru. Zde jsou využity konkrétní hodnoty pro určení x, y, z souřadnic a orientace krychle. Po tomto kroku program vytváří instanci robota a řídí jeho pohyb k místu umístění krychle. Následuje proces přisátí krychle funkcí suck() a její zvednutí o 5 cm změnou hodnoty z v objektu Pose. Nakonec program instruuje robota, aby se vrátil do nové pozice a uvolnil krychli pomocí funkce release().

```
#Sample task for creating program: Write me a program that lifts up a cube
    ↪ (5cm) from a certain position and then drops it. Load the current
    ↪ arm position as a cube location and use it as a variable.
#Here is the sample code:
# Import is always needed
import modules.robot as robot
# If it is necessary to initialize Pose (usually using function calling),
    ↪ use literal values
# It is nice when you separate on multiple lines so it becomes more
    ↪ readable
cube_pose = robot.Pose(
    robot.Position(x=0.014, y=-0.292, z=-0.052),
    robot.Orientation(w=0, x=-0.690, y=0.724, z=0)
)
# create an instance (you don't need to know why but it is important for
    ↪ some reasons)
r = robot.Robot()
# Moves to cube position
r.move_to(pose=cube_pose, moveType="JUMP", velocity=100, acceleration=100,
    ↪ safe=False)
# attach cube
r.suck()
# create or change Pose (values are in meters so 0.05 is 5 cm on the z axis
    ↪ )
cube_pose.position.z += 0.05
# move to change / new Pose
r.move_to(pose=cube_pose, moveType="JUMP", velocity=100, acceleration=100,
    ↪ safe=False)
# release cube
r.release()
```

Výpis 4.2: Vzorový program pro ChatGPT

4.4 Nápověda pro uživatele

Při spuštění aplikace se uživatelům zobrazí úvodní zpráva, která lze kdykoliv znovu vyvolat zadáním příkazu „help“. Tato zpráva má za úkol představit možnosti a schopnosti virtuálního asistenta, poskytnout uživatelům přehled o tom, jak s ním mohou pracovat a jaké úkoly mohou pomocí asistenta provádět.

Zpráva uvádí, že asistent je navržený k pomoci s ovládáním robotického ramene a podává uživatelům informace o tom, jak mohou přímo manipulovat s ramenem nebo vytvářet programy pro automatizaci opakovaných úloh. Dále jsou v ní obsaženy příklady úkolů, které lze asistentovi zadat, jako jsou například požadavky na zobrazení aktuální pozice ramene, provádění kalibrace, pohyb ramene o určenou vzdálenost nebo výpis dostupných programů. Uživatelům jsou také nabídnuty tipy, jak mohou tvořit vlastní programy, a jak by měli formulovat příkazy pro maximální přesnost a efektivitu.

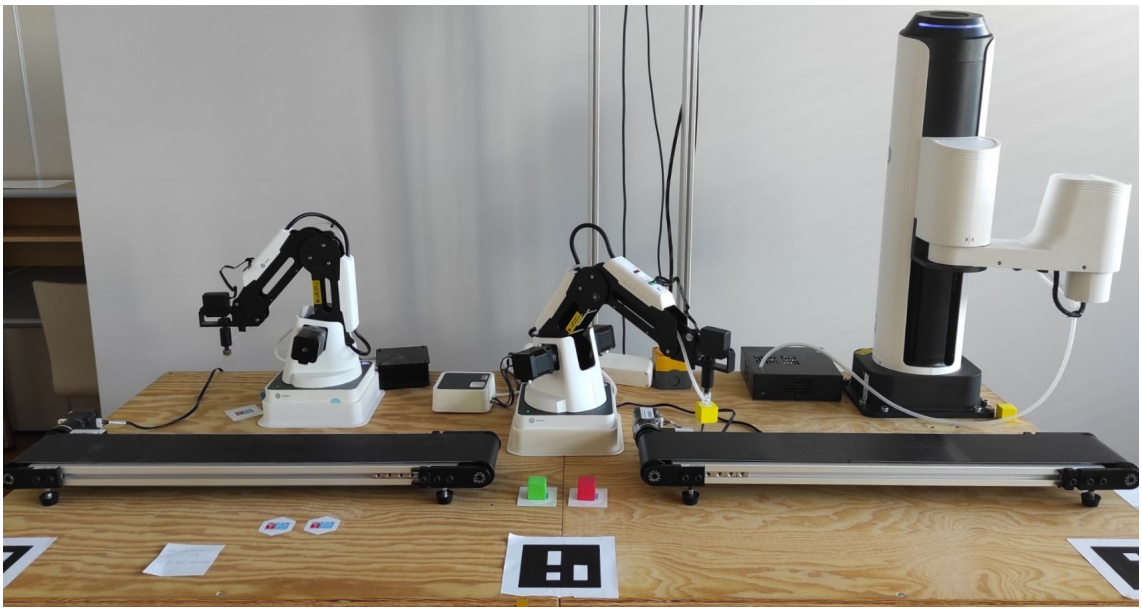
Tato úvodní zpráva tedy funguje jako jednoduchý návod a zároveň jako vstupní bod pro uživatele, kteří se s aplikací teprve seznamují, a usnadňuje jim pochopení, jak asistent může být využit pro práci s robotickým ramenem.

```
Ahoj, jsem tvůj virtuální asistent, který ti pomůže pracovat s robotickým
    ramenem.
Umožním ti buď přímo pracovat s ramenem nebo vytvořit program pro nějakou
    rutinní operaci.
Robot se může pohybovat, přesouvat objekty, a také je možné používat
    dopravníkový pás.

Pro opakované úkoly si můžeš vytvořit program, který následně můžeš uložit
    pod libovolným názvem a spustit ho. Také se k programu můžeš vrátit a
    upravit ho nebo ho smazat.
Příklady úkolů které mi můžeš zadat:
- Vypiš mi aktuální pozici (funkci pro získání pozice můžeš použít i při
    tvorbě programu)
- Proveď kalibraci (vhodné pokud dojde k chybě - označuje červená dioda)
- Posuň se o 5 cm směrem nahoru
- Vypiš mi dostupné programy
- Napiš mi program který přesune objekt z jedné pozice na druhou. (pro
    inspiraci)
 - Při takovém úkolu je nutné získat zdrojovou a cílovou pozici.
 - Zároveň je nutné co nejpřesněji a nejjasněji popsat požadavek.
 - Pro efektivní práci s pozicemi je doporučeno buď nejprve načíst všechny
    potřebné pozice, které jsou využívány v zadáních úkolů, nebo vytvořit
    kostru programu, načíst pozice a poté je nechat začlenit do kódu.

Klidně se na cokoliv zeptej :)
Pro výpis této nápovědy zadej "help"
Pro ukončení zadej "exit"
```

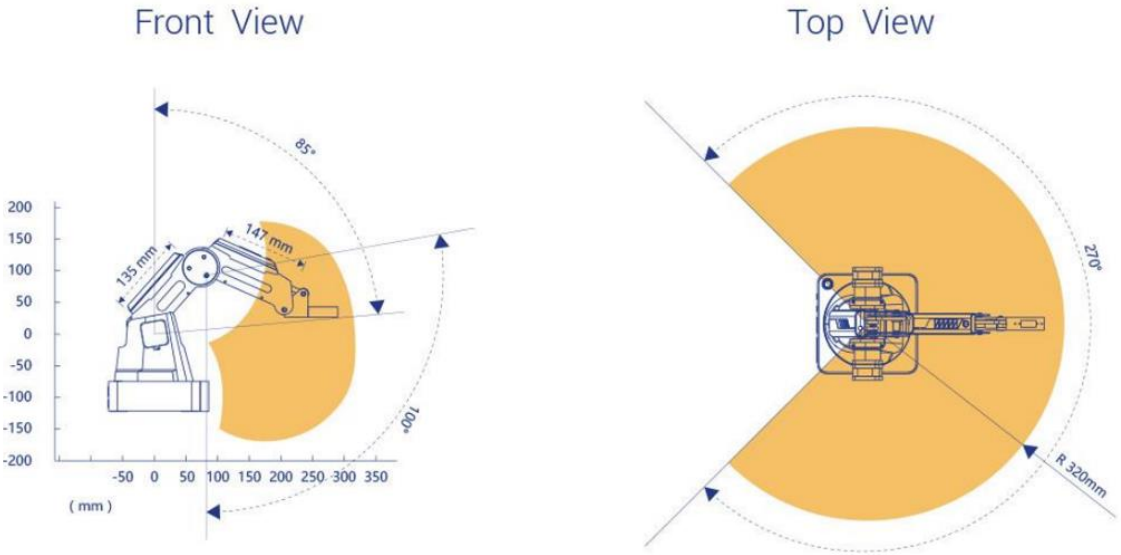
Výpis 4.3: Příklad takového logu

4.5 Popis pracoviště

Vývoj a testování aplikace probíhalo s využitím zařízení a prostředků viz obrázek 4.4 robotické laboratoře O104. V této laboratoři jsem měl k dispozici robota „Dobot Magician V2“ 4.6, který byl vybaven přísavkou a dopravníkovým pásem. Pro účely testování jsem využíval také barevné molitanové kostky o velikosti strany přibližně 2.5 cm, které byly v laboratoři dostupné.



Obrázek 4.4: Ukázka pracoviště s roboty (Dobot Magician je vlevo a uprostřed)



Obrázek 4.5: Dosah robota Dobot Magician [10]

Na obrázku 4.5 je patrné, že robot disponuje pěti klouby, díky čemuž má schopnost pohybovat se ve třech osách a otáčet úchop kolem osy z.

Obrázek 4.6: robot Dobot Magician

Dobot Magician je výukový robot vybaven mechanickým ramenem, který umí pracovat s širokým spektrem příslušenství pro různé účely a podporuje komunikaci prostřednictvím Bluetooth a WiFi rozhraní. Robot je designován tak, aby byl přístupný pro širokou škálu uživatelů, od začátečníků po studenty technických oborů, což jej činí vhodným pro výukové účely.

Robot dokáže pracovat s několika nástavci, které rozšiřují jeho funkčnost: pneumatický a vakuový gripper s pumpou, nástroje pro 3D tisk a držák pera pro psaní a kreslení. Tyto komponenty umožňují robotu provádět rozličné úkoly, od manipulace s objekty po grafické práce. K dispozici je také Laser kit pro základní laserové gravírování.

Ovládání robota je možné prostřednictvím joysticku, což přidává praktickou složku k učebnímu procesu, a podporuje tak interaktivní výuku.

Kapitola 5

Implementace aplikace

Tato kapitola je věnována podrobnému popisu implementace aplikace asistenta, včetně klíčových rozhodnutí, která jsem při tomto procesu učinil. Při zpětném pohledu je důležité zmínit, že se od doby začátku vývoje technologie posunuly. V současné době existují alternativní přístupy k implementaci, které by mohly přinést určité výhody a vyhnout se některým z výzev, kterým jsem čelil.

Modularita projektu zjednodušuje přidávání nových funkcí a aktualizace, což napomáhá jeho dlouhodobé udržitelnosti a adaptabilitě. Hlavní část projektu tvoří skripty assistant.py a chat_interface.py, které představují základní logickou strukturu na nejvyšší úrovni abstrakce. V adresáři modules se nacházejí další důležité moduly, jako jsou `functions.py` a `logger.py`, které jsou přímo integrovány do hlavní aplikace. Modul `robot.py` se využívá ve functions.py k řízení interakcí s robotickým API a je také součástí generovaného kódu.

Modul functions.py je navržen tak, aby zpracovával žádosti o akce od ChatGPT a zároveň poskytoval další podpůrné funkce, které zvyšují efektivitu a spolehlivost vykonávaných operací. Modul logger.py hrál klíčovou roli v záznamu kontextu operací a umožňuje vizuální odlišení změn v kódu, což je užitečné pro rychlé identifikace a opravy. chat_interface.py propojuje všechny tyto funkce do grafického uživatelského rozhraní v prohlížeči, což umožňuje intuitivní interakci s uživateli.

5.1 Odesílání a příjem zpráv na OpenAI API

Tato sekce se věnuje funkci send_to_chatGPT, která zajišťuje veškerou komunikaci s ChatGPT. Zahrnuje kontrolu délky kontextu, zpracování chyb, logování odpovědí a provádění akcí požadovaných ChatGPT, přičemž současně poskytuje zpětnou vazbu o výsledcích těchto akcí prostřednictvím rekurzivních volání. Funkce eventuelně vrací textovou odpověď.

5.1.1 Proměnné prostředí

Pro správný chod aplikace je nezbytné definovat proměnné prostředí (environmental variables).

- **OPENAI_API_KEY** Tato proměnná je klíčová pro funkčnost aplikace, neboť umožňuje komunikaci s OpenAI API.
- **ROBOT_URL** Tato proměnná je nezbytná pro práci s robotem v reálném čase, což umožňuje přímé ovládání a spouštění programů.

- **MODEL** Proměnná umožňuje výběr specifické verze ChatGPT pro práci. V případě, že není specifikována, je automaticky použita verze „gpt-3.5-turbo-0125“, na které byla aplikace testována.
- **DEBUG** Pro získávání detailních informací o chodu aplikace lze nastavit tuto proměnnou. Standardní nastavení je na hodnotě 0, přičemž lze zvolit hodnoty od 0 do 10.

5.1.2 Kontext konverzace

Kontext konverzace je uchováván v seznamu slovníků, přičemž každý slovník zahrnuje klíče a hodnoty ve formátu typu řetězec. Tento kontext je průběžně zapisován do logových souborů ve formě JSON struktury. Logovací soubory lze využít pro načtení kontextu konverzace při novém spuštění programu. Tato funkcionalita je obzvláště užitečná v případech, kdy dojde k neočekávanému ukončení programu, což umožňuje uživatelům bez problémů pokračovat v práci tam, kde skončili. Dále lze logy využít k obnovení kontextu, ve kterém jsou uloženy často používané pozice na pracovišti, což usnadňuje rutinní operace. Tato metoda také umožňuje uživatelům navázat na předchozí neukončené úkoly a pokračovat ve spolupráci s aplikací bez nutnosti opětovného nastavování kontextu od začátku. Při ukončení aplikace jsou na konec ještě přidány dodatečné informace jako je počet použitých tokenů a konkrétní model který byl použit. Velikost kontextu se odvíjí od verze GPT, viz tabulka 2.1. Při každém odesílání požadavku provádím kontrolu stavu kontextu, abych zajistil, že je v něm dostatečný prostor pro odpověď. Specificky se snažím udržet alespoň 1000 tokenů, což zahrnuje potřebnou rezervu pro zpracování odpovědi.

```
[{
    "role": "system",
    "content": "Assume you are a virtual assistant...",
{
    "role": "user",
    "content": "zkalibruj se"
},
{
    "role": "assistant",
    "content": null,
    "function_call": {
        "name": "putHome",
        "arguments": "{}"
    }
},
{
    "role": "function",
    "name": "putHome",
    "content": "Success!"
},
{
    "role": "assistant",
    "content": "Kalibrace byla úspěšně provedena."
},
{
    "used_tokens": "2668",
    "gpt": "gpt-3.5-turbo-0125"
}]
```

Výpis 5.1: Příklad takového logu

5.1.3 Kontrola vstupu

Pro kontrolu vstupu jsem implementoval detekci klíčových slov (*funkce is_command*), která spouští určité operace, jako je výpis nápovědy nebo ukončení aplikace. Bylo by možné využít i Function calling, ale například v případě ukončení aplikace je to zbytečné. Tímto způsobem také zabraňuji zaslání prázdného vstupu.

5.1.4 Odeslání zprávy

Před odesláním požadavku se provádí kontrola, zda je velikost kontextu dostatečná pro očekávanou odpověď. Pro odesílání požadavků na API OpenAI využívána knihovna openai ve verzi 0.28.1. Následující kód ilustruje proces odeslání zprávy a přijetí odpovědi.

Pro odeslání používám metodu `openai.ChatCompleton.create`, která je v současné době už označená jako „legacy“. Při volání této metody je nutné specifikovat model, který bude použit a kontext pro ChatGPT. Dále volitelně jsou zde specifikace dostupných funkcí (více v sekci Function calling 2.3.2) a maximální počet tokenů použitých pro odpověď.

5.1.5 Příjem zpráv

Než předám textovou odpověď pro další zpracování, je nezbytné nejdříve zkontrolovat, zda ChatGPT nepožaduje spuštění některé z funkcí uvedených ve specifikacích. V případě, že byla nějaká funkce vyžádána, pokusím se ji provést s danými parametry viz obrázek 4.3. Výsledek provedení funkce, ať už úspěšný nebo informace o případných chybách, je přidán do kontextu a znovu odeslán na API. Jeden z významných aspektů této fáze je nutnost zpracování chyb, neboť ChatGPT nemusí vždy vrátit validní JSON strukturu. Toto dosáhnu rekurzivním voláním funkce pro odesílání zpráv, přičemž vracím její návratovou hodnotu. Tento přístup umožňuje i řetězené volání funkcí, kdy se postupně provedou všechny funkce vyžádané ChatGPT, následované vrácením textové odpovědi.

5.2 Modul robot

Modul „robot“ je navržen tak, aby poskytoval komplexní řešení pro interakci s robotickým API, zatímco zajišťuje, že kód, který generuje, je snadno pochopitelný pro uživatele. Jeho struktura a funkcionalita jsou klíčové pro efektivní práci s robotickým zařízením.

Pomocné třídy

Pro uchování polohy a orientace v prostoru jsou vytvořeny třídy `Position` a `Orientation`, které jsou následně zkombinovány ve třídě `Pose`. Tato třída je důležitá při pohybu paže robota, protože při pohybu API robota vyžaduje nejen cílovou pozici ale zároveň orientaci.

Třída Robot

Hlavní třída „Robot“ pak slouží jako rozhraní pro všechny operace, které je možné s robotem provádět. Inicializace této třídy zahrnuje možnost specifikovat URL adresu robota a režim, ve kterém bude modul pracovat. Toto nastavení ovlivňuje, jak modul reaguje na chyby – buď vyvoláním výjimky, nebo vrácením chybové zprávy. Při inicializaci se provádí ověření spojení s robotem. Pokud nedojde k úspěšnému navázání spojení s robotem, aplikace se automaticky ukončí.

Třída Robot v modulu robot obsahuje řadu metod, které umožňují komplexní interakci s robotickým zařízením. Kromě základních metod, jako je spuštění, zastavení a dotaz zda je robot spuštěn, zahrnuje i specifičtější funkce pro práci s robotem:

move_to: Tato metoda umožňuje robotu přesunout se na specifikovanou pózu. Uživatel může specifikovat typ pohybu (například JUMP, LINEAR, JOINS), který ovlivňuje trajektorii pohybu robota. Parametry jako rychlost a zrychlení pohybu lze také nastavit, což dává uživatelům flexibilitu při práci s robotem.

suck a release: Tyto metody kontrolují vakuum robota, kde suck aktivuje sací mechanismus pro chycení objektů a release ho deaktivuje, což umožňuje pustit objekt. Tyto funkce jsou klíčové pro manipulaci s předměty.

get_pose: Vrací aktuální pózu robota, což je důležité především pro generování kódu. Tato metoda volá v rámci Function calling, aby mohly být hodnoty následně doplněny do kódu.

belt_distance a belt_speed: Metody pro kontrolu dopravníkového pásu, které umožňují nastavit směr a rychlost otáčení pásu, a v případě belt_distance i vzdálenost, o kterou se má pás posunout.

home: Pokud vznikne chyba v provozu robota, lze robota zkalibrovat pomocí této metody.

V závěru jsem se pokusil přidat metody, jako jsou rotace základny, rotace přísavky, nebo přesun objektu. Tento proces zahrnuje řetězení příkazů, jako je přesun na první pozici, aktivace přísavky, následný přesun na druhou pozici, deaktivace přísavky a potenciální návrat na výchozí pozici.

Všechny důležité třídy a metody obsahují docstringy, které jsou potom načítány v modulu functions do systémového vstupu. Tyto dokumentační řetězce slouží především jako vestavěná dokumentace pro jednotlivé třídy a metody, což umožňuje uživatelům a vývojářům lepší orientaci ve funkcích a správném využívání modulu. Docstringy tak přispívají k lepší srozumitelnosti kódu a usnadňují jeho údržbu a aktualizace.

5.3 Modul functions

Modul „functions“ zprostředkovává interakci mezi ChatGPT a robotem, kde hlavní role třídy `FunctionHandler` spočívá v řízení a provádění funkcí, které ChatGPT vyžaduje. Tento modul tedy nejen umožňuje ChatGPT spouštět konkrétní akce, ale obsahuje i další pomocné funkce.

5.3.1 Třída FunctionHandler

Při vytváření instance třídy `FunctionHandler` se provádí mapování textových názvů funkcí na metody třídy. To zahrnuje nastavení základních parametrů, jako je URL pro komunikaci s robotem. Přímé ovládání robota funguje prostřednictvím instance třídy `robot`. Díky tomu je možné při změně v modulu robot automaticky odrážet tyto změny i ve funkcionalitách třídy FunctionHandler, což zjednodušuje aktualizace a údržbu celého systému.

Metoda handle_function:

Tato klíčová metoda provádí specifické funkce na základě mapování, které je definováno předem. Na základě příchozího požadavku od ChatGPT metoda aktivuje odpovídající funkci s příslušnými parametry. Následně vrací textové výstupy, ať už se jedná o potvrzení úspěšného provedení nebo o chybové hlášení.

Pomocné funkce

Funkce `get_all_specs` slouží k načítání specifikací funkcí uložených ve formátu JSON, které poskytují podrobnosti o dostupných akcích asistenta pro práci s programy a pro ovládání robota. `load_module_info` extrahuje podrobné informace o metodách modulu z jejich dokumentačních řetězců, což je důležité pro správné pochopení a efektivní využití těchto metod a jsou esenciální pro systémový vstup. `get_prompt_message` skládá a vrací systémový vstup 4.3, zatímco `get_welcome_message` poskytuje uvítací zprávy, které nastavují počáteční kontext pro uživatele a vítají je při spuštění systému.

Funkce pro přímé ovládání robota

Tato skupina funkcí interaguje s API robota skrze instanci robot a zahrnuje operace jako jsou spuštění, zastavení, přesun, kalibrace, zapnutí/vypnutí přísavky, atd. Pro některé z akcí

modul vyžaduje specifické parametry, které musí být přítomny pro úspěšné vykonání požadované operace. Funkce dále ověřuje, zda jsou všechny potřebné argumenty k dispozici a správně zadané, a v případě nesrovnalostí vrací užitečné chybové hlášení.

Funkce pro práci s programy

Aplikace poskytuje správu lokálně uložených skriptů a programů, zásadní pro dynamickou interakci s robotem. Nabízí funkci pro ukládání textů do souborů, což uživatelům umožňuje vytvářet a uchovávat nové skripty. Vzhledem k tomu, že každý program by měl obsahovat importy nezbytné pro správnou funkci s robotickým modulem, byla implementována **statická kontrola**. Tato kontrola zajišťuje, že soubory s příponou „.py“ obsahují potřebné importy a instance robotů, což zvyšuje spolehlivost a integritu skriptů. Dále aplikace umožňuje procházení uložených souborů a načítání jejich obsahu, což je klíčové pro úpravy a další použití skriptů.

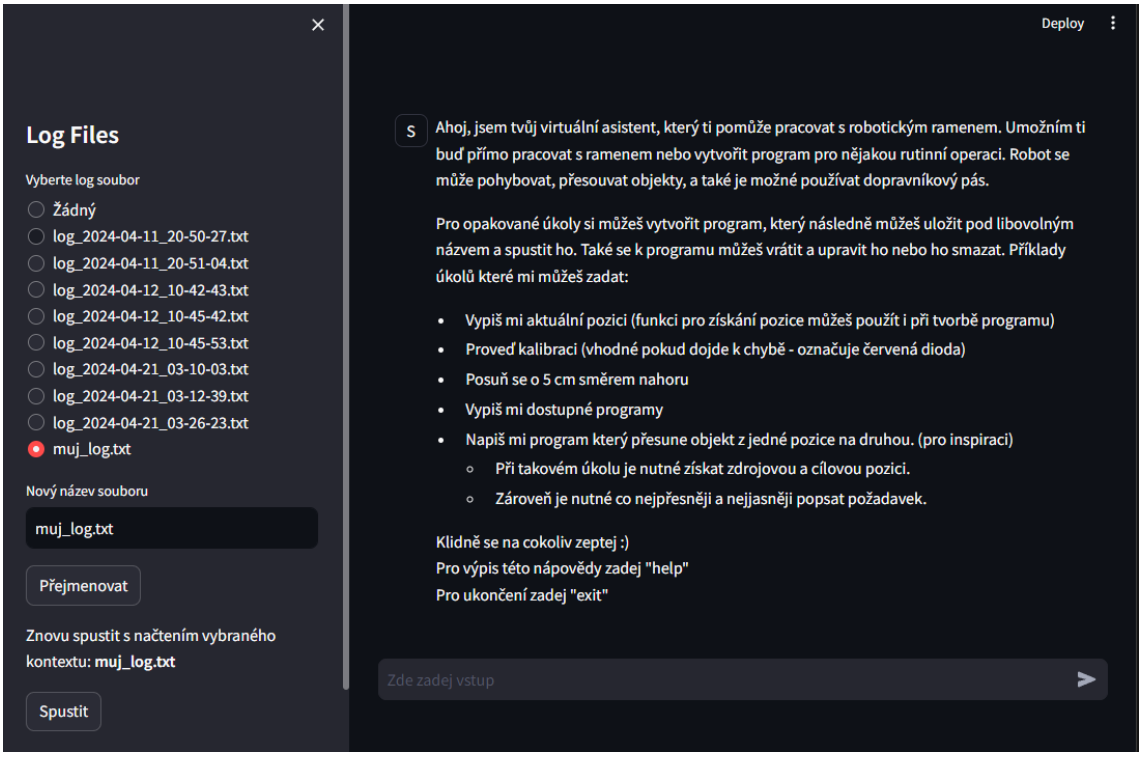
Funkce pro odstranění programů soubory neodstraňuje definitivně, ale přesouvá je do složky „trash_src“, což uživatelům poskytuje možnost obnovit soubory v případě potřeby. Tento bezpečnější způsob řízení mazání dat doplňuje funkci spouštění skriptů jako podprocesů, která vrací výstup (stdout) nebo chybové hlášení (stderr). Toto umožňuje uživatelům rychle identifikovat a řešit případné problémy s pomocí ChatGPT.

5.4 Integrace jiného robota

Při integraci nového robota je třeba aktualizovat specifikace a vybavení v systémovém vstupu a upravit logiku v modulu robot. Dále je potřeba přizpůsobit mapování funkcí v modulu functions a aktualizovat soubor „robot_func.json“ pro přímou práci s novým robotem. Rozsah nezbytných změn závisí na rozdílech mezi původním a novým robotem. V budoucnu by mohlo být možné tyto specifikace částečně automatizovat, což by zjednodušilo proces adaptace na různé robotické systémy.

5.5 Implementace grafického uživatelského rozhraní

Pro zvýšení uživatelské přívětivosti jsem ve svém projektu vytvořil webové GUI 5.1 s využitím knihovny Streamlit. Tato platforma je oblíbená díky své schopnosti poskytovat jednoduché a interaktivní webové aplikace napsané v Pythonu. Nástroje jsou importovány z dalších souborů projektu, což usnadňuje organizaci kódu. Streamlit umožňuje vývojářům rychle implementovat uživatelské rozhraní, které zefektivňuje interakci s aplikací a zvyšuje její přístupnost.



Obrázek 5.1: Ukázka grafického uživatelského rozhraní

Kapitola 6

Testování

Tato část práce poskytuje přehled testovacích metod použitých k hodnocení použitelnosti a uživatelské přívětivosti aplikace. Testování bylo rozděleno do dvou hlavních fází. První se zaměřila na interakci skutečných uživatelů s aplikací, aby byla ověřena její praktická použitelnost a identifikovány případné problémy v běžném provozu. Druhá fáze porovnávala efektivitu různých verzí ChatGPT v řešení programovacích úkolů podle předem definovaných scénářů. Výsledky těchto testů poskytují cenné vhledy do funkcionality aplikace a ukazují, jak silně ovlivňuje výběr modelu efektivitu celé aplikace.

6.1 Uživatelské testování

Během této fáze bylo provedeno detailní testování, jehož primárním cílem bylo nejen ověření praktické použitelnosti textového uživatelského prostředí navržené aplikace, ale také identifikace a odhalení možných chyb, které by mohly uniknout během běžného vývojového procesu.

Testování se zaměřilo na pozorování, jak uživatelé interagují s aplikací, a na jejich schopnost efektivně řešit dané úkoly s pomocí ChatGPT verze „gpt-3.5-turbo-0125“ 2.1. Jako doplňkovou možnost jsem vybraným uživatelům nabídl vyzkoušení pokročilejší verze „gpt-4“, abych získal zpětnou vazbu o tom, zda zaznamenají rozdíly v kvalitě a přesnosti odpovědí.

Klíčovým kritériem byla celková použitelnost pro programování robotického ramene.

K testování byly využity barevné molitanové kostky dostupné v laboratoři.

V této fázi testování byla aplikace provozována v prostředí příkazové řádky. Je důležité poznamenat, že podmínky testování se pro různé účastníky mírně lišily. Důvodem bylo to, že během testovacího procesu byly některé zjištěné chyby identifikovány a průběžně opravovány. Tento postup průběžných oprav vedl ke zvýšení stability a spolehlivosti aplikace v průběhu testovacího procesu.

1. **Seznámení se s prostředím** V této části jsem uživatele seznámil s pracovním prostředím a asistentem. Uživatelům byla představena aktivní aplikace a bylo jim doporučeno, aby se nejen seznámili s **uvítací zprávou** 4.4, ale také aby byli aktivní v komunikaci s asistentem. Cílem bylo povzbudit uživatele, aby kladli dotazy a testovali různé funkce asistenta, čímž by si lépe osvojili možnosti aplikace.

2. **Přímé ovládání ramene** Dalším úkolem bylo ovládat robotické rameno a posunout jej o určenou vzdálenost v libovolném směru. Cílem bylo posoudit, jak dobře ChatGPT

rozumí uživatelským pokynům pro fyzické pohyby a jak přesně dokáže interpretovat relativní pozicování.

3. **Program pro zvednutí a upuštění kostky** Tento úkol testoval schopnosti asistenta v práci s konkrétními programovacími úkoly. Uživatelé byli požádáni o vytvoření a testování programu, který by zvedl a následně upustil kostku. Úkol zkoumal schopnost uživatelů pracovat s různými pozicemi a testoval flexibilitu aplikace v přizpůsobení se různým programovacím přístupům.

4. **Přesun kostky z jedné pozice na druhou** Podobný předchozímu úkolu, ale s větší mírou složitosti. Účastníci byli vyzváni k vytvoření programu, který by přesunul kostku z jedné pozice na druhou. Zde bylo klíčové sledovat, jak uživatelé pracují se dvěma rozdílnými pozicemi, a jak aplikace reaguje na složitější sady pokynů.

5. **Výměna kostek mezi dvěma pozicemi** Finální úkol spočíval ve vytvoření programu, který by umožnil vyměnit kostky mezi dvěma pozicemi. Tento úkol byl zaměřen na to, jak uživatelé formulují a popisují složitější problémy a zda ChatGPT dokáže efektivně využít svých znalostí pro řešení běžného problému výměny prvků.

6.1.1 Výsledky testování

Tabulka níže 6.1 shrnuje odpovědi uživatelů na různé aspekty použití aplikace, kde hodnocení 0 znamená „vůbec nesouhlasím/nevyužívám“ a 5 „plně souhlasím/často využívám“.

Označení:	A	B	C	D	E	F	G
Věk:	22	23	22	23	22	22	34
Jaké máte zkušenosti s ChatGPT?	4	2	5	4	4	4	3
Jaké máte zkušenosti s prací s roboty?	1	1	1	1	1	1	5
Jaké máte zkušenosti s programováním?	3	4	5	5	5	4	5
Přišlo Vám uživatelské prostředí přehledné?	3	4	5	4	5	4	4
Přišel Vám kód čitelný?	3	5	4	4	4	5	5
Jak byste ohodnotil(a) kvalitu generovaného kódu?	4	4	3	4	4	4	4
Byla pro Vás příjemná práce s asistentem?	4	4	5	4	4	3	5
Byla nápověda pro použití dostatečná?	5	4	5	4	2	5	5

Tabulka 6.1: Tabulka výsledků dotazníku

1. **Věkové složení uživatelů:** Většina uživatelů testujících aplikaci byla ve věku 22 až 23 let (celkem 7), s jedním respondentem ve věku 34 let. Tento věkový rozsah není ideální pro primární cílovou skupinu aplikace, avšak pro testování byli dostupní právě tito účastníci.

2. **Zkušenosti s ChatGPT:** Čtyři ze sedmi uživatelů pravidelně využívají ChatGPT, což naznačuje dobrou obeznámenost s funkcemi tohoto nástroje. Zároveň to poukazuje na zvýšenou schopnost zadávat přesné vstupy pro ChatGPT.

3. **Zkušenosti s robotikou:** Většina respondentů (6 z 7) uvedla, že má alespoň základní znalosti v oblasti robotiky, což může ovlivnit jejich schopnost interagovat s robotem.

4. **Programovací dovednosti:** Téměř všichni uživatelé (6 z 7) se identifikovali jako pokročilí nebo expertní programátoři, což přispívá k jejich schopnosti efektivně vyhodnotit generovaný kód.

5. **Přehlednost uživatelského rozhraní:** Uživatelské prostředí aplikace bylo hodnoceno jako velmi přehledné většinou respondentů, což naznačuje dobrou intuitivnost rozhraní.

6. **Čitelnost kódu:** Většina uživatelů (6 z 7) považovala kód za čitelný a srozumitelný, což ukazuje na úspěšnou implementaci intuitivních názvů metod a celkově modulu robot.

7. **Spokojenost s asistentem:** Práce s asistentem byla obecně hodnocena jako příjemná, což svědčí o uživatelské přívětivosti a funkčnosti nástroje.

Komentáře uživatelů:

- „Velký rozdíl při testování asistenta byl v použité verzi GPT, a to konkrétně mezi verzemi 3.5 a 4.0. Některé úkoly byly s verzí 3.5 obtížné, až nereálné, kdežto s novější verzí bylo možné s robotem pracovat bez jakýchkoliv komplikací. V budoucnu by mohlo být zajímavé ovládání pomocí hlasu."

- „Můj největší problém byl robotovi sdělit, aby při generování kódu používal mnou definovanou a pojmenovanou lokaci místo volání funkce get_pose(). Časem se nám povedlo upravit dotazy, aby toho byl schopný, ale rozhodně jsem se na tom zasekl."

- „Při prvním kontaktu jsem si nebyl úplně jistý, co robot umí a co tedy po ChatGPT můžu chtít. Pro ovládání robota je používané REST API, tak bych uvítal podle něj shrnout, co teda robot vůbec dokáže."

Průběh testování:

- **Úkoly 1 a 2:** Tyto úkoly byly spojeny do jedné sekce kvůli jejich překrývající se povaze. Většina uživatelů se seznámila s aplikací bez větších problémů, avšak vyskytly se určité nedostatky a nesrovnalosti. Specificky, byla zjištěna nejasnost v uvítací zprávě, která vyžadovala zkrácení a upřesnění. Také se jeden uživatel pokusil odkázat na instrukce v uvítací zprávě, která ale není součástí kontextu. Dále chyby v relativním pozicování, kde ChatGPT nesprávně interpretoval směrové pokyny. Někteří uživatelé narazili na problémy s voláním funkcí, kde chyběly některé povinné argumenty, což ChatGPT nemohl vždy správně doplnit.

- **Úkol 3:** V tomto úkolu se ChatGPT pokusil integrovat načítání pozice do programu, ale neuspěl kvůli statické kontrole 5.3.1. Byly zaznamenány také problémy s nepřesným označováním pozic, což mohlo být způsobeno změnami polohy robota, o kterých ChatGPT neměl přehled. Tento problém byl později řešen upravením systémového vstupu. Někdy ChatGPT vygeneroval pouze část kódu nebo uživatelé vyjádřili přání upravit kód, což bylo splněno s proměnlivými výsledky.

- **Úkol 4:** Přestože úkol 4 byl obecně úspěšný, vyskytly se některé problémy při práci s pozicemi a další chyby, které byly již dříve zmíněny. Nicméně, pět ze sedmi uživatelů byli schopni úkol splnit bez větších obtíží, což ukazuje na jejich rostoucí seznámení s asistentem a jeho funkcemi.

- **Úkol 5:** Tento úkol se ukázal být náročnější, a většina uživatelů ho na původní testované verzi „gpt-3.5-turbo-0125“ nezvládla splnit pod danými podmínkami. Pouze jeden z sedmi uživatelů byl schopen vytvořit funkční program. Úspěšnou se ukázala strategie serializace úlohy, čímž výrazně usnadnili práci modelu. Uživatelům, kteří projevili zájem, byla nabídnuta možnost vyzkoušet úkol na modelu „gpt-4“, a tři ze tří poznamenali, že rozdíl v kvalitě byl patrný, což vedlo k bezproblémovému vytvoření programu. Je tedy ještě nutné podotknout, že už byli do jisté míry zkušení v používání asistenta.

- **Obecné chyby:** Kromě specifických problémů u jednotlivých úkolů jsme zaznamenali i obecné chyby, jako překročení kontextu, chybné požadavky na openai API, nevyžádané volání funkcí, chybějící nebo neúplné argumenty a nesprávné jméno metody robota. Bylo také zjištěno, že ChatGPT někdy nerozlišuje mezi vykonáváním akcí a generováním kódu.

- **Uživatelská zkušenost:** Někteří uživatelé poznamenali, že byli příjemně překvapeni relevantními odpověďmi a zároveň kvalitou generovaného kódu, který byl v mnohých případech rovnou spustitelný. Oceňují také nápomocnost asistenta a jeho schopnost poskytovat užitečné rady.

- **Návrhy na zlepšení:** Mezi navrhovaná vylepšení patří úprava uvítací zprávy pro lepší srozumitelnost, barevné označení výstupů pro lepší vizuální orientaci, změna úvodu, aby bylo jasné, že na uvítací zprávu nelze odkazovat v kontextu, a možnost uložení často používaných pozic. Dále bylo navrženo obalit bloky kódu pro lepší identifikaci a označení ve výstupech, zvláště v konzolové aplikaci.

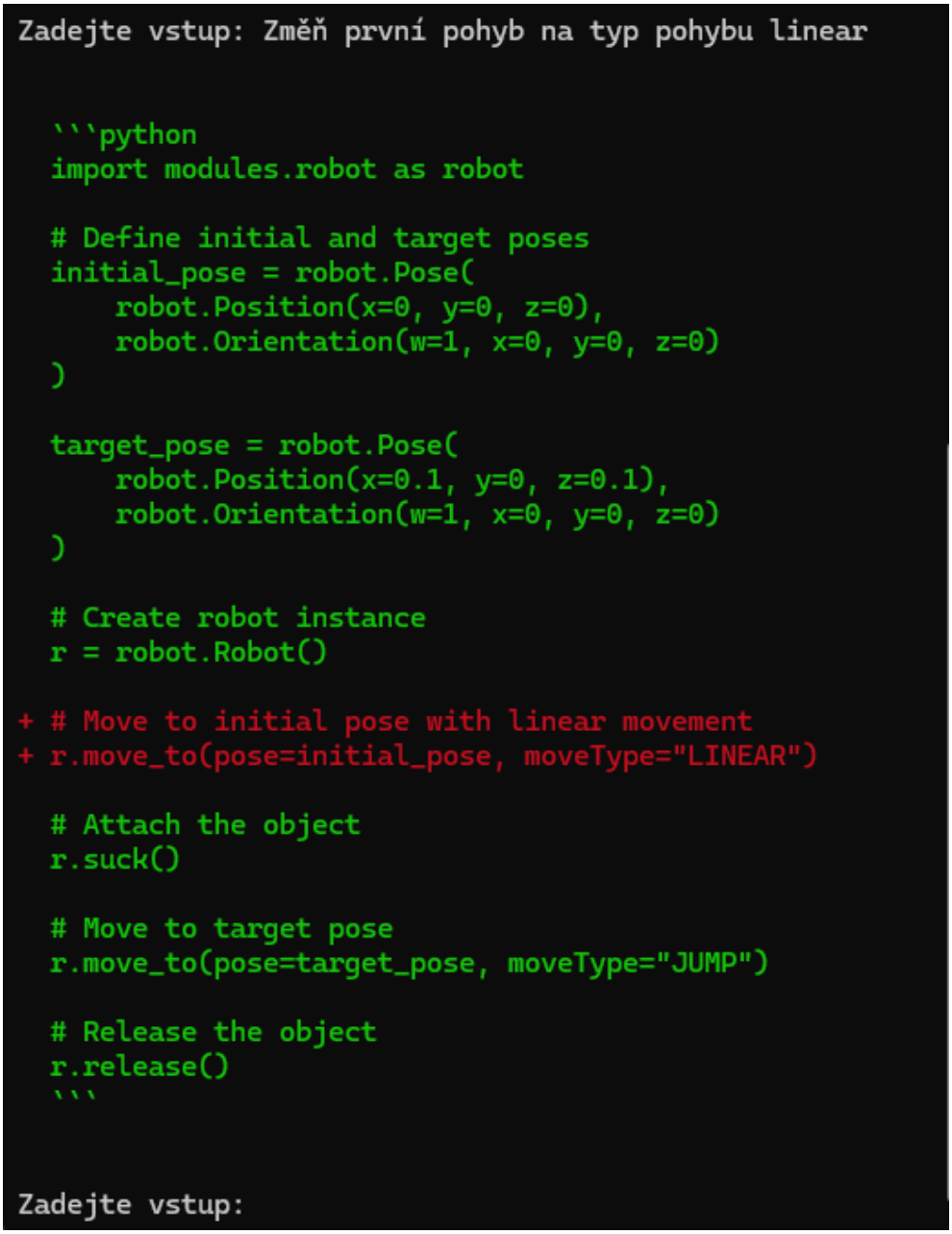
- **Poznatky z testování:** Z testů vyplývá, že chování asistenta lze efektivně modifikovat podle preferencí uživatele. Uživatelé s předchozími zkušenostmi s ChatGPT tvořili jasnější a specifičtější vstupy, což vedlo k lepším výsledkům. Pro práci s pozicemi se jako nejefektivnější ukázal přístup, kdy jsou nejprve načteny všechny potřebné pozice, na něž se poté odkazuje v zadáních úkolů, nebo kdy je nejprve vytvořena kostra programu, načteny pozice a ty jsou poté doplněny do kódu.

- **Opravy a změny:** V reakci na problémy byla implementována dodatečná sekce systémového vstupu 4.3. Tato část se zaměřuje na zlepšení schopnosti ChatGPT rozlišovat mezi přímými příkazy a žádostmi o kód. Tím se snažíme zvýšit efektivitu a přesnost odpovědí modelu.

  Také bylo důležité přidat upozornění, že robot může být fyzicky přesunut mezi jednotlivými akcemi, aby se předešlo opakovanému přiřazování stejných souřadnic různým pozicím.

  Byl rovněž realizován návrh na barevné označení výstupů (viz obrázek 6.1), což zlepšuje vizuální orientaci a usnadňuje práci s aplikací.

  Uvítací text 4.4 byl nakonec doplněn o informace týkající se efektivních postupů při vytváření programů.



Obrázek 6.1: Ukázka konzolové aplikace s barevně zvýrazněnými zprávami

6.2 Porovnání verzí ChatGPT

V této části se věnuji porovnání výsledků získaných z různých verzí ChatGPT, jak je podrobněji specifikováno v tabulce 2.1. Při testování jsem se zaměřil na poskytnutí co nejjasnějších a nejpřesnějších instrukcí pro zajištění relevantních výsledků. Je třeba mít na paměti, že výstupy ChatGPT mohou být do určité míry ovlivněny náhodou, což může mít vliv na interpretaci dat, zejména vzhledem k omezenému rozsahu vzorku.

Pro účely tohoto testu jsem požádal o generování pouze základní kostry programu, aby se předešlo potřebě neustálého načítání specifických pozic. V reálných aplikacích by bylo

efektivní načíst a uložit si potřebné pozice pod označeními proměnných a pouze doplnit literály do generovaného kódu.

Abych zajistil stručnost a přehlednost této sekce, rozhodl jsem se vynechat z vygenerovaného kódu standardní prvky, jako jsou importy, inicializace pozic a inicializace tříd robota. Kompletní příklad programu, obsahující všechny tyto prvky, lze nalézt v předchozí části dokumentu, která popisuje vzorový program 4.3.1.

1. **Zadání: Napiš mi kostru kódu pro přesun objektu z pozice A do pozice B**

   Popis úkolu: Tento úkol byl vybrán jako základní test pro srovnání funkčnosti obou verzí při generování jednoduchého kódu. **Výsledek:** Oba generované kódy byly funkční, avšak s rozdíly v jazyce komentářů a způsobu inicializace pozic. **Poměr potřebných vstupů:** Verze 3.5 (1), verze 4 (1).

2. **Zadání: Napiš mi kostru kódu pro výměnu objektů na pozici A a pozici B za použití pozice C**

   Popis úkolu: Testování složitějšího scénáře pro vyhodnocení schopnosti modelů zvládnout komplexní logiku. **Výsledek:** Verze GPT-4 byla schopna ihned vygenerovat funkční kód, zatímco verze 3.5 měla problémy s porozuměním obsazení pozic. **Poměr potřebných vstupů:** Verze 3.5 (3), verze 4 (1).

3. **Zadání: Napiš mi kostru kódu pro přesun 3 kostek o straně 2.5cm postavených na sebe**

   Popis úkolu: Test schopností modelů zvládnout algoritmické zpracování úkolu s více kroky. **Výsledek:** GPT-4 se ukázalo jako efektivnější v algoritmizaci úkolu, což umožnilo lepší rozšiřitelnost kódu. **Poměr potřebných vstupů:** Verze 3.5 (7), verze 4 (3).

4. **Zadání: Napiš mi kostru kódu, který vezme objekt z pásu (pozice A), namočí ho do barvy (pozice B) a nakonec ho položí na odkládací plochu (pozice C). Objekty jsou připraveny na dopravníkovém pásu s rozestupy 8 cm a je jich celkem 5**

   Popis úkolu: Cílem bylo otestovat schopnost modelů zvládnout přesné sekvencování akcí v kontextu průmyslové výroby. **Výsledek:** Verze GPT-4 úspěšně generovala kód, který byl přímo použitelný, zatímco verze GPT-3.5 Turbo generovala kód, který obsahoval chyby a nesprávně aplikované metody, které ve skutečnosti neexistují (například „beltSpeed“ namísto „belt_speed“). **Poměr potřebných vstupů:** Verze 3.5 (4), verze 4 (1).

5. **Zadání: Napiš mi kostru kódu, který bude brát objekty z pásu (pozice A) a bude je skládat vedle pásu do řady(pozice B) směrem vlevo. Objekty jsou připraveny na dopravníkovém pásu s rozestupy 6 cm a je jich celkem 5. Objekty na pozici B od sebe budou vzdáleny 4 cm**

   Popis úkolu: Testování schopnosti modelů zvládnout logistické uspořádání a prostorovou orientaci objektů. **Výsledek:** Opět se ukázalo, že verze GPT-4 je schopná efektivněji generovat kód, který odpovídá zadání, bez nepotřebných akcí nebo chyb. Verze GPT-3.5 Turbo způsobila několik nevhodných úprav kódu, které byly nejen neefektivní, ale i kontraproduktivní. **Poměr potřebných vstupů:** Verze 3.5 (5), verze 4 (2).

6.2.1 Výsledky testování

Model GPT-4 nejenže vykazoval lepší porozumění zadání a schopnost generovat efektivnější a přesnější kódy ve srovnání s verzí GPT-3.5, ale také ukázal značnou schopnost redukovat množství nutných uživatelských vstupů pro dosažení žádaných výsledků. V praxi to znamená, že GPT-4 byl schopen splnit stejné úkoly s menším počtem interakcí od uživatele, což značně zvyšuje jeho praktickou aplikovatelnost a efektivitu v reálných projektech.

Kapitola 7

Závěr

Tato bakalářská práce představila inovativní přístup k programování robotického ramene s využitím ChatGPT, což otevírá nové možnosti pro uživatele bez pokročilých znalostí programování. Práce naznačila, že použití virtuálního asistenta má potenciál zjednodušit procesy a umožnit efektivní ovládání robotických systémů. Zjistil jsem, že výkon a efektivita tohoto přístupu závisí značně na schopnostech použitého modelu a kvalitě uživatelského vstupu, který by měl být co nejpřesněji a nejjasněji formulovaný. Silnější modely mohou několikanásobně zvýšit efektivitu asistenta, což výrazně snižuje množství potřebných uživatelských interakcí pro dosažení požadovaných výsledků.

Během testování byly identifikovány klíčové oblasti pro zlepšení. Ačkoliv implementace aplikace ukázala potenciál, existuje mnoho možností, jak systém zefektivnit. Simulace proveditelnosti a průběhu programů před jejich spuštěním, spolu s vizualizací aktivních procesů, by uživatelům usnadnily porozumění o fungování programů, čímž by se zvýšila jejich použitelnost a zjednodušila práce s nimi.

Navíc, integrace dalších technologií, jako je speech-to-text pro ovládání hlasem nebo technologie OpenAI Visions pro vizuální rozpoznání a interakce, by mohla přinést ještě efektivnější a flexibilnější řešení pro budoucí projekty. Díky OpenAI Visions by se například ChatGPT mohl sám orientovat v prostoru a identifikovat dostupné zdroje, což by zjednodušilo interakci a zvýšilo efektivitu práce. Rozšíření aplikace na další typy robotů a práce s více roboty najednou by mohlo přinést zásadní změnu v interakci člověka s robotickými systémy.

V závěru lze říci, že přístup založený na ChatGPT, jak byl prezentován v této práci, nabízí slibnou platformu pro budoucí výzkum a rozvoj v oblasti automatizace a robotiky. Tato práce nejenže potvrdila použitelnost ChatGPT pro ovládání robotických systémů, ale také nastínila cesty pro budoucí inovace a zlepšení.

Literatura

[1] 3Blue1Brown. *But what is a GPT? Visual intro to transformers | Chapter 5, Deep Learning* [https://www.youtube.com/watch?v=wjZofJX0v4M]. 2024. Datum přístupu: 20. dubna 2024.

[2] Diogo Goncalves. *Practical considerations in LLM sizes & deployments* [https://superwise.ai/blog/practical-considerations-in-llm-sizes-deployments/]. 2023. Datum přístupu: 20. dubna 2024.

[3] Ekin, S. Prompt engineering for ChatGPT: a quick guide to techniques, tips, and best practices. *Authorea Preprints*. Authorea. 2023.

[4] Lucy Tancredi. *UNDERSTANDING CHATGPT: THE PROMISE AND NUANCES OF LARGE LANGUAGE MODELS* [https://insight.factset.com/understanding-chatgpt-the-promise-and-nuances-of-large-language-models]. 2023. Datum přístupu: 20. dubna 2024.

[5] Mittal, A. *Základní průvodce rychlým inženýrstvím v ChatGPT* [https://www.unite.ai/cs/prompt-engineering-in-chatgpt/]. 2023. Datum přístupu: 20. dubna 2024.

[6] OpenAI. *Models* [https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4]. 2024. Datum přístupu: 20. dubna 2024.

[7] OpenAI. *OpenAI docs* [https://platform.openai.com/docs]. 2024. Datum přístupu: 20. dubna 2024.

[8] OpenAI. *Pricing* [https://www.openai.com/pricing]. 2024. Datum přístupu: 14. dubna 2024.

[9] OpenAI. *Tokenizer* [https://platform.openai.com/tokenizer]. 2024. Datum přístupu: 20. dubna 2024.

[10] RobotLAB. *Dobot Magician Specification* [https://www.robotlab.com/hubfs/Dobot/Dobot%20Magician%20Specifications.pdf?t=1515807244104]. 2024. Datum přístupu: 30. dubna 2024.

[11] Tereza Blažková. *Umí Python opravdu všechno, aby se ti jej vyplatilo umět?* [https://www.itnetwork.cz/blog/umi-python-opravdu-vsechno-aby-se-ti-jej-vyplatilo-umet]. 2024. Datum přístupu: 20. dubna 2024.

[12] Torayev, A., Martínez Arellano, G., Chaplin, J. C. a Sanderson, D. *fanucpy: Python Interface for FANUC Robots* [https://github.com/torayeff/fanucpy]. GitHub, 2022.

[13] Vemprala, S., Bonatti, R., Bucker, A. a Kapoor, A. ChatGPT for Robotics: Design Principles and Model Abilities. *ArXiv.org*. Ithaca: Cornell University Library, arXiv.org. 2023. ISSN 2331-8422.

[14] Wake, N., Kanehira, A., Sasabuchi, K., Takamatsu, J. a Ikeuchi, K. ChatGPT Empowered Long-Step Robot Control in Various Environments: A Case Application. *IEEE Access*. 2023, sv. 11, s. 95060–95078. DOI: 10.1109/ACCESS.2023.3310935.

Příloha A

Obsah přiloženého paměťového média

```
/ Bachelor_Thesis
├── assistant.py - Konzolová aplikace
├── chat_interface.py - Webové GUI
├── modules/
│   ├── __init__.py
│   ├── functions.py - Modul pro provedení vyžádaných akcí
│   ├── robot.py - Modul pro komunikaci s robotem
│   └── logger.py - Modul pro práci s logy
├── txt_sources/
│   ├── functions.json - Specifikace pro práci s programy
│   ├── robot_func.json - Specifikace pro přímou práci s robotem
│   ├── intro.txt - Uvítací zpráva pro uživatele
│   ├── prompt.txt - Systémový vstup
│   └── sample.py - Vzorový program
├── logs/ - Složka pro ukládání logů
├── src/ - Složka pro ukládání vytvořených programů
├── trash_src/ - Složka pro odstraněné programy
├── thesis.pdf - Bakalářská práce ve formátu pdf
├── bp/ - Složka se soubory pro překlad textu bakalářské práce
├── AssistantVideo.mp4 - Ukázkové video
├── requirements.txt - Seznam knihoven a verzí
├── Dockerfile
└── README.md - Návod pro spuštění
```