ISBN: 978-981-18-7950-0
2023 the 13th International Workshop on Computer Science and Engineering (WCSE 2023)
doi: 10.18178/wcse.2023.06.031

From Words to Flight: Integrating OpenAI ChatGPT with PX4/Gazebo for Natural Language-Based Drone Control

Mohamed Lamine TAZIR[1+], Matei MANCAS[1] and Thierry DUTOIT[1]

[1] ISIALAB, Numediart Institute, University of Mons, Mons, Belgium.

Abstract. The rapid progress of large language models in recent years has opened new opportunities for human-robot interaction. In this paper, we propose a novel approach to control drones using natural language commands by integrating OpenAI ChatGPT with the PX4/Gazebo simulator. The proposed system enables users to interact with the simulator using everyday language, allowing them to control the drone's actions with ease, eliminating the need for extensive training in drone piloting. We discuss the implementation details, including the validation of the ChatGPT-generated commands and their translation into executable actions in the simulator. To the best of our knowledge, this is the first proposal of a verification and validation system for commands generated by ChatGPT and LLMs in general. Furthermore, we discuss the crafting of effective prompts and the essential criteria for doing so. Our approach demonstrates promising results in terms of both usability and reliability, paving the way for further research on natural language-based control systems for robotics applications.

Keywords: LLM, ChatGPT, PX4, Gazebo, UAV, AI-assisted control, prompt engineering, Commands validation.

1. Introduction

The recent rapid progress in natural language processing (NLP) has led to the development of large language models (LLMs) such as GPT-3 [1], GPT-4 [2] and Codex [3] that have proven effective in text generation, machine translation, and code synthesis. OpenAI ChatGPT [4], a pretrained generative text model, is an impressive addition to the field, with exceptional interaction skills through the combination of text generation and code synthesis. Despite these successes, the application of LLMs to robotic tasks remains a challenge due to the need for a deep understanding of real-world physics, the environment, and the ability to perform physical actions. Developing a generative robotics model requires a robust common sense knowledge, a sophisticated world model, and the ability to interpret and execute commands physically and coherently.

As robots become more integrated into our daily lives, there is a growing need to facilitate human-robot interaction, particularly in drone control. Drones have found numerous applications in various fields, such as surveillance, package delivery, and agriculture, etc. However, their operation often requires technical knowledge, making them less accessible to non-experts. Natural language-based interfaces present a promising solution that could enable intuitive and accessible drone control. In response to this need, we have developed a chatbot program that interfaces with a PX4/Gazebo [5] drone simulation, leveraging the OpenAI API to generate responses to user queries and commands.

PX4/Gazebo is a popular open-source simulator for testing drone control algorithms and developing autonomous drone applications. Integrating natural language control in the simulator can serve as the first step to validate a more intuitive control method for drones. In this paper, we propose an approach to integrate ChatGPT with PX4/Gazebo, enabling users to control the drone in the simulator using natural language commands. By interfacing with this simulator, our program offers a realistic and safe environment for users to practice and experiment with drone control. Additionally, the program has the ability to run XML code in the PX4/Gazebo environment, allowing users to test and refine their own drone control algorithms.

[+] Corresponding author. Tel.: +3265374068.
E-mail address: mohamed.tazir@umons.ac.be.

2. Related Works

In this section, we will discuss relevant literature on natural language-based control systems for robots, large language models (LLMs) in robotics, and drone control using natural language.

2.1. Natural language-based control systems for robots

The development of natural language processing (NLP) techniques has opened up new possibilities for human-robot interaction. One area of interest is the control of robots using natural language commands, which can significantly improve the accessibility and ease of use of robotic systems. Tellex et al. [6] proposed a system that translates natural language commands into robot control policies using a probabilistic graphical model. They demonstrated the effectiveness of their approach in controlling a robotic forklift.

Guadarrama et al. [7] introduced a semantic parsing approach to understand natural language commands and convert them into executable robot actions. It grounds the meaning of input sentences in terms of visual percepts from the robot's sensors, enabling the robot to understand and execute complex commands involving multiple objects and spatial relationships. This enables the PR2 robot to execute appropriate commands or respond to spatial queries.

2.2. Large language models in robotics

With the advent of LLMs, such as GPT-3, the capabilities of NLP models have significantly improved, allowing for more sophisticated natural language understanding and generation tasks. Researchers have started exploring the use of these models in robotics applications.

The paper [8] investigates whether pre-trained LLMs can extract actionable knowledge for grounded tasks in interactive environments. The authors demonstrate the feasibility of this approach by grounding high-level tasks expressed in natural language to a chosen set of actionable steps.

Another study, [9] explores the use of LLMs trained on code-completion to write robot policy code given natural language commands. This approach is demonstrated across multiple real robot platforms and achieves state-of-the-art results on the HumanEval benchmark.

Singh et al. in [10] presents PROGPROMPT, a programming language structure that uses LLMs to generate robot task plans. The authors demonstrate the success of this approach in VirtualHome household tasks and on a physical robot arm for tabletop tasks.

The paper "LATTE: LAnguage Trajectory TransformEr" [11] proposes a language-based framework for modifying generic robotic trajectories using pre-trained language models. The authors show that this approach can successfully follow human intent, modifying the shape and speed of trajectories within multiple environments.

2.3. Drone Control using Natural Language

There have been several efforts to develop natural language interfaces for drone control. A recent paper [12] explores the use of OpenAI's ChatGPT for robotics applications, including drone control. The authors propose a strategy that combines prompt engineering principles with a high-level function library, allowing ChatGPT to adapt to different robotics tasks and simulators. They integrate ChatGPT with Microsoft AirSim simulator [13] and demonstrate the effectiveness of this approach in executing various types of robotics tasks [14], showing that ChatGPT can be effective while allowing users to interact with it through natural language.

Our work sets itself apart from existing literature by offering several distinct contributions:

1. Achieving a successful integration of OpenAI's ChatGPT API with the PX4/Gazebo platform.
2. Enabling drone control in natural language through the Pixhawk autopilot (commander).
3. Developing and implementing a verification and validation system for commands generated by ChatGPT, ensuring the reliability and safety of the natural language-based control approach.
4. Identifying and outlining the most relevant criteria for crafting effective prompts that guarantee the generation of zero-shot responses, providing valuable insights for future research and applications involving human-robot interaction and large language models.

3. Methodology

In this section, we describe the proposed system's implementation, including the integration of ChatGPT with PX4/Gazebo, the validation of generated commands, and the translation of these commands into executable actions in the simulator.

3.1. Integration of ChatGPT with PX4/Gazebo

In this work, we have developed a system that enables users to control the drone through natural language input. The input is processed by the ChatGPT API, which generates appropriate commands for execution. The system employs the PX4 flight stack to manage drone control and perform various tasks.

The developed system establish a chatbot that interfaces with the PX4/Gazebo simulator, utilizing OpenAI's GPT-3.5-Turbo model to respond to user queries and commands. The integration process involves setting up a connection between the chatbot and the simulator. We develop a Python-based middleware to create a communication channel that serves as an interface between ChatGPT and the PX4/Gazebo simulator. This middleware processes the natural language inputs, forwards requests to ChatGPT, retrieves responses, and generates commands for the simulator. Upon initialization, the middleware connects to the Gazebo environment, facilitating communication with the simulated drone.

We structure the chatbot to operate in a loop where it waits for user input, sends the user's message and chat history to the OpenAI API to generate a response, appends the response to the chat history, and prints it to the console. This systematic approach ensures seamless interaction between the user and the drone simulation, resulting in an efficient and user-friendly natural language-based drone control system.

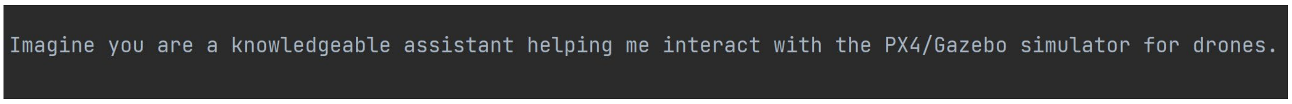
3.2. Command Generation

The command generation process involves the use of user-provided prompts to guide ChatGPT in generating more specific and relevant responses. The developed system requires two input files: one for the temporary prompt, which is used to initialize the discussion, and the second for the permanent prompt, which is injected throughout the discussion to control the generated responses.

The act of defining these two prompts is called "prompt engineering" and it is an area an active research [15-16]. In the following, we will define the most relevant criteria for defining effective prompts that ensure zero-shot response generation.

3.3. Defining the context

A well-defined context is essential for guiding ChatGPT to generate responses relevant to the problem domain. The prompt should provide sufficient information about the domain, in this case, controlling drones in the PX4/Gazebo simulator (Figure 1). By setting the context, ChatGPT can focus its responses accordingly and maintain consistency with the target system.

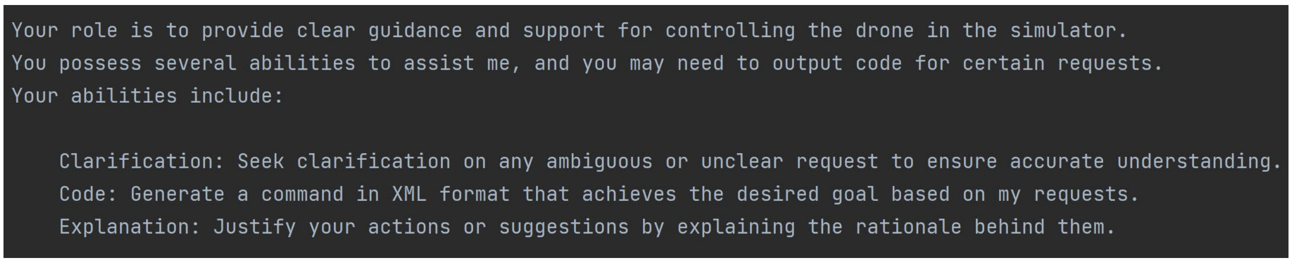


```
Imagine you are a knowledgeable assistant helping me interact with the PX4/Gazebo simulator for drones.
```

Fig. 1: Defining the context for ChatGPT.

3.4. Defining the role of ChatGPT

Specifying the role of ChatGPT is crucial for shaping its behavior and the type of responses it generates. Clearly state the role, describing the purpose and the support it should provide to the user. In the given example, ChatGPT is assigned the role of an assistant, which helps the model understand that it is expected to provide guidance and support in the form of PX4 commands and explanations (figure 2).

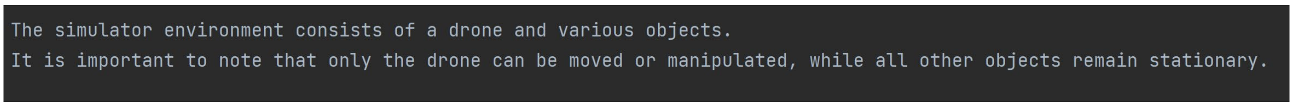


```
Your role is to provide clear guidance and support for controlling the drone in the simulator.
You possess several abilities to assist me, and you may need to output code for certain requests.
Your abilities include:

    Clarification: Seek clarification on any ambiguous or unclear request to ensure accurate understanding.
    Code: Generate a command in XML format that achieves the desired goal based on my requests.
    Explanation: Justify your actions or suggestions by explaining the rationale behind them.
```

Fig. 2: Defining the role of ChatGPT.

3.5. Describing the environment

Providing information about the environment in which the task is being performed. This may include details about the simulator, the physical world, or any specific conditions that are relevant to the task (figure 3). This helps ChatGPT generate more accurate and contextually appropriate responses or actions.

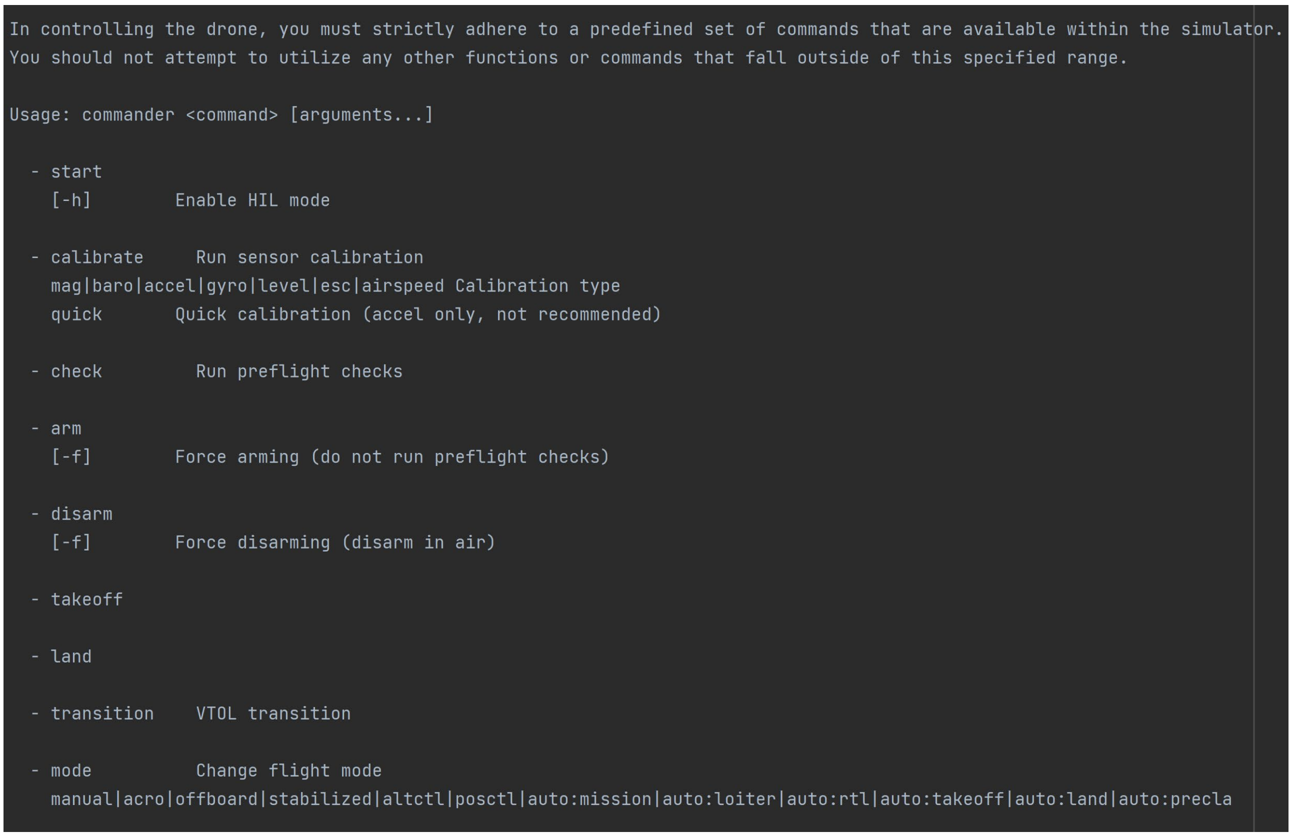


```
The simulator environment consists of a drone and various objects.
It is important to note that only the drone can be moved or manipulated, while all other objects remain stationary.
```

Fig. 3: Describing the environment.

3.2.4. Defining the allowed functions

To ensure that ChatGPT generates responses consistent with the given functionality and constraints, the prompt should explicitly list the available functions and prohibit the use of any other functions (Figure 4). This approach helps maintain compatibility with the target system and avoids generating unusable or erroneous responses.



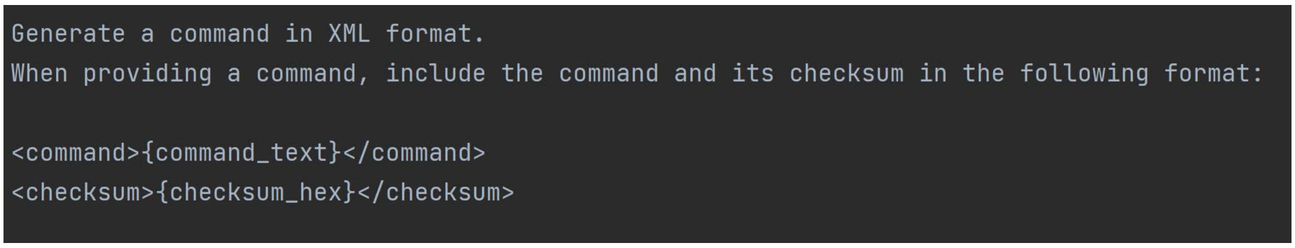
```
In controlling the drone, you must strictly adhere to a predefined set of commands that are available within the simulator.
You should not attempt to utilize any other functions or commands that fall outside of this specified range.

Usage: commander <command> [arguments...]

  - start
    [-h]        Enable HIL mode

  - calibrate     Run sensor calibration
    mag|baro|accel|gyro|level|esc|airspeed Calibration type
    quick       Quick calibration (accel only, not recommended)

  - check         Run preflight checks

  - arm
    [-f]        Force arming (do not run preflight checks)

  - disarm
    [-f]        Force disarming (disarm in air)

  - takeoff

  - land

  - transition    VTOL transition

  - mode          Change flight mode
    manual|acro|offboard|stabilized|altctl|posctl|auto:mission|auto:loiter|auto:rtl|auto:takeoff|auto:land|auto:precla
```

Fig. 4: Defining the allowed functions.

3.2.5. Defining the output format

Establishing the desired output format in the prompt is essential for maintaining consistency across responses and ensuring compatibility with the target system. In the given example, the prompt specifies the

XML format for the generated responses, which allows the system to parse and process the responses as required (figure 5).

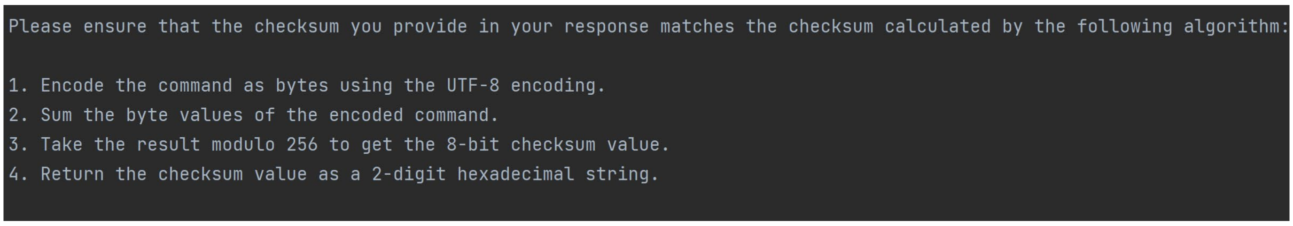


```
Generate a command in XML format.
When providing a command, include the command and its checksum in the following format:

<command>{command_text}</command>
<checksum>{checksum_hex}</checksum>
```

Fig. 5: Defining the output format.

3.2.6. Defining constraints

Including any limitations or constraints that should be adhered to by ChatGPT, such as computing checksums, character limits, or any other restrictions that apply to the task. In our case, the checksum algorithm is provided for command validation and system reliability (figure 6). By defining such constraints, the prompt guides ChatGPT to produce responses that are both accurate and usable in the target system.

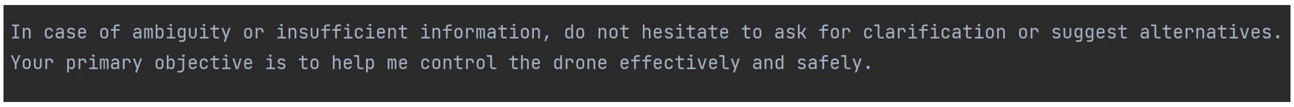


```
Please ensure that the checksum you provide in your response matches the checksum calculated by the following algorithm:

1. Encode the command as bytes using the UTF-8 encoding.
2. Sum the byte values of the encoded command.
3. Take the result modulo 256 to get the 8-bit checksum value.
4. Return the checksum value as a 2-digit hexadecimal string.
```

Fig. 6: Defining constraints.

3.2.7. Handling ambiguity

Providing guidance on how the model should handle ambiguous queries or when it lacks sufficient information to generate a confident response. This could involve asking clarifying questions or suggesting alternative solutions (figure 7).

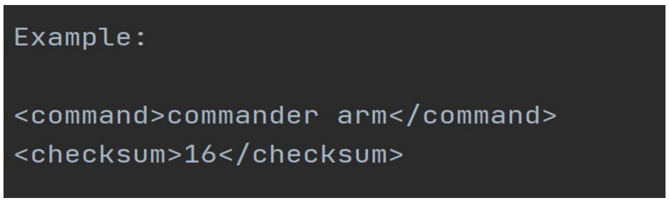


```
In case of ambiguity or insufficient information, do not hesitate to ask for clarification or suggest alternatives.
Your primary objective is to help me control the drone effectively and safely.
```

Fig. 7: Handling ambiguity.

3.2.8. Summarizing everything in an example

Providing a well-structured example in the prompt helps ChatGPT understand the desired output format and the relationship between different components, such as command and checksum, in the response (figure 8). The example serves as a template for the model to follow, ensuring that generated responses align with the established criteria and requirements of the target system.



```
Example:

<command>commander arm</command>
<checksum>16</checksum>
```

Fig. 8: Summarizing everything in an example.

3.6. Command Validation

Validating the generated commands is crucial for ensuring the reliability and safety of the drone control system. In this subsection, we detail the steps involved in the command validation process, which includes checking the command's syntax, and calculating the command checksum.

3.6.1. Syntax Validation

The first step in the validation process is to examine the syntax of the generated command. This is achieved by implementing a syntax parser that checks if the command adheres to the specified format, as per the PX4/Gazebo simulator requirements. Any commands failing to meet these requirements are flagged as invalid and not executed.

3.6.2. Checksum Calculation and Verification

To further enhance the reliability of the system, a checksum is calculated for each generated command. The checksum algorithm is detailed in the prompt provided to ChatGPT and involves the following steps:

1. Encode the command as bytes using the UTF-8 encoding.
2. Sum the byte values of the encoded command.
3. Take the result modulo 256 to get the 8-bit checksum value.
4. Return the checksum value as a 2-digit hexadecimal string.

The calculated checksum is then compared with the checksum provided by ChatGPT in its response. If the checksums match, the command is considered valid and proceeds to execution. If the checksums do not match, the command is deemed invalid and not executed, prompting the user to either modify the input or request a new command from ChatGPT.

Upon receiving a response from ChatGPT, the middleware validates the command. This algorithm checks the consistency and accuracy of the generated command to ensure it adheres to the established command structure and avoids potential errors. If the command passes the validation process, it proceeds to the translation phase; otherwise, the middleware requests an alternative command from ChatGPT or alerts the user.

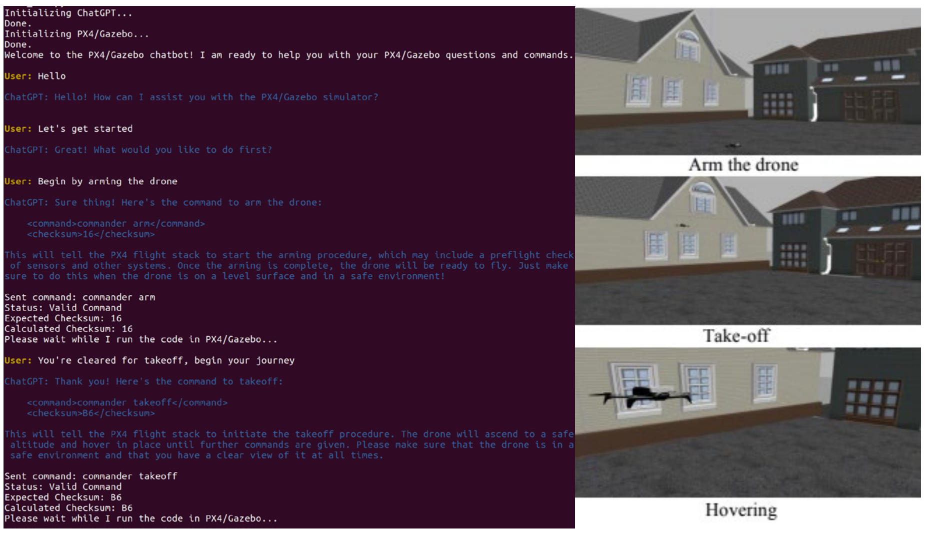
3.7. Translation of Commands into Simulator Actions

Once a valid command is generated, the middleware translates the natural language command into an executable action for the PX4/Gazebo simulator. This process involves mapping the command to a corresponding MAVLink message that the simulator can interpret and execute. The middleware sends the appropriate MAVLink messages corresponding to the received commands, allowing the drone to perform the desired actions within the Gazebo environment.

By following these steps, we implement a natural language-based control system that seamlessly integrates ChatGPT with the PX4/Gazebo simulator. This system allows users to control a drone in a simulated environment using intuitive and accessible natural language commands.

4. Experiments

The system developed in this work provides an intuitive and user-friendly interface for drone control, allowing users to control the drone using natural language inputs. The results of the discussion shown in Figure 9 demonstrate the successful implementation of the natural language-based drone control system.



```
Initializing ChatGPT...
Done.
Initializing PX4/Gazebo...
Done.
Welcome to the PX4/Gazebo chatbot! I am ready to help you with your PX4/Gazebo questions and commands.

User: Hello

ChatGPT: Hello! How can I assist you with the PX4/Gazebo simulator?


User: Let's get started

ChatGPT: Great! What would you like to do first?


User: Begin by arming the drone

ChatGPT: Sure thing! Here's the command to arm the drone:

    <command>commander arm</command>
    <checksum>16</checksum>

This will tell the PX4 flight stack to start the arming procedure, which may include a preflight check
 of sensors and other systems. Once the arming is complete, the drone will be ready to fly. Just make
sure to do this when the drone is on a level surface and in a safe environment!

Sent command: commander arm
Status: Valid Command
Expected Checksum: 16
Calculated Checksum: 16
Please wait while I run the code in PX4/Gazebo...

User: You're cleared for takeoff, begin your journey

ChatGPT: Thank you! Here's the command to takeoff:

    <command>commander takeoff</command>
    <checksum>B6</checksum>

This will tell the PX4 flight stack to initiate the takeoff procedure. The drone will ascend to a safe
 altitude and hover in place until further commands are given. Please make sure that the drone is in a
 safe environment and that you have a clear view of it at all times.

Sent command: commander takeoff
Status: Valid Command
Expected Checksum: B6
Calculated Checksum: B6
Please wait while I run the code in PX4/Gazebo...
```

Fig. 9: Example of ChatGPT conversation and generated commands

The chatbot was able to understand and respond to user prompts accurately, allowing for seamless control of the drone in the simulation environment. The system was tested on several tasks, including arming and disarming the drone, changing the drone's flight mode, and executing simple flight patterns, etc.

4.1. Command Generation Accuracy

To assess the command generation accuracy, we designed a set of test scenarios consisting of various natural language inputs representing different drone actions. We then evaluated the generated commands based on their correctness and compliance with the established command structure. To achieve a comprehensive assessment, we requested ChatGPT to generate 30 distinct natural language models for each of the primary commands, including arming, disarming, taking off, landing, and mode changes.

Table 1: Command Generation Accuracy Results

Test Scenario	Number of Correct Commands	Total Commands	Accuracy (%)
Arming	28	30	93.3
Disarming	26	30	86.6
Take-off	30	30	100.0
Landing	27	30	90.0
Mode Changes	25	30	83.3
Total / Average	136	150	90.6

Table 1 shows the results of the command generation accuracy experiment. The table presents the number of correct commands generated by ChatGPT and the corresponding accuracy for each test scenario. The table shows that the overall command generation accuracy is 90.6%, indicating that ChatGPT can effectively generate correct and valid commands for the PX4/Gazebo simulator based on natural language input.

5. Conclusion

This paper has presented a novel approach to enabling natural language interaction between users and the PX4/Gazebo drone simulator using OpenAI's ChatGPT. We have detailed the system architecture, the command generation and validation processes, and the translation of commands into simulator actions. Additionally, we have proposed a set of criteria for designing effective prompts to ensure zero-shot response generation.

Through a series of experiments, we have demonstrated the efficacy of our approach in generating accurate and relevant commands from natural language input. The results showcase the potential of leveraging advanced language models like ChatGPT to facilitate intuitive and efficient interaction between humans and robotic systems.

In future work, we aim to refine the prompt engineering process to improve response quality further and extending the approach to other simulators or real-world robotic systems. Additionally, we plan to focus on the development and utilization of our own methods for localization, trajectory planning, and obstacle avoidance, rather than relying solely on the functions predefined in the PX4 commander. Overall, this research contributes to the ongoing efforts to bridge the gap between human-robot communication and paves the way for more accessible, user-friendly interfaces in robotics.

6. Acknowledgements

This work is funded by the EU and Walloon region (Belgium) as part of the ViewSkyModel project.

7. References

[1] BROWN, Tom, MANN, Benjamin, RYDER, Nick, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 2020, vol. 33, p. 1877-1901.

[2] OpenAI, GPT-4 Technical Report, *arXiv preprint arXiv: 2303.08774*, 2023.

[3] CHEN, Mark, TWOREK, Jerry, JUN, Heewoo, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[4] OpenAI. ChatGPT, 2023. https://openai.com/blog/chatgpt/, Retrieved May 5, 2023.

[5] PX4 Autopilot Software. (n.d.). Software Overview. Retrieved may 5, 2023, from https://px4.io/software/software-overview/.

[6] Tellex, S., Kollar, T., Dickerson, S., Walter, M. R., Banerjee, A. G., Teller, S., & Roy, N. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation. *Proceedings of the National Conference on Artificial Intelligence (AAAI),* 2011.

[7] Guadarrama, S., Riano, L., Golland, D., Göhring, D., Jia, Y., Klein, D., Abbeel, P., & Darrell, T. Grounding spatial relations for human-robot interaction. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 2013.

[8] Wenlong Huang, Pieter Abbeel, Deepak Pathak, Igor Mordatch. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. *In Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

[9] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, Andy Zeng. Code as policies: Language model programs for embodied control, *arXiv preprint arXiv:2209.07753*, 2022.

[10] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, Animesh Garg. Progprompt: Generating situated robot task plans using large language models, *arXiv preprint arXiv:2209.11302*, 2022.

[11] Arthur Bucker, Luis Figueredo, Sami Haddadin, Ashish Kapoor, Shuang Ma, Sai Vemprala, Rogerio Bonatti. LaTTe: Language Trajectory TransformEr, *arXiv preprint arXiv:2208.02918*, 2022.

[12] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. ChatGPT for Robotics: Design Principles and Model Abilities. *Microsoft, Tech. Rep. MSR-TR-2023*, 2023.

[13] Shital Shah, Debadeepta Dey, Chris Lovett, Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *In Proceedings of the Field and Service Robotics: Results of the 11th International Conference,* 2018.

[14] Matthew R Walter, Siddharth Patki, Andrea F Daniele, Ethan Fahnestock, Felix Duvallet, Sachithra Hemachandra, Jean Oh, Anthony Stentz, Nicholas Roy, Thomas M Howard. Language understanding for field and service robots in a priori unknown environments, *arXiv preprint arXiv:2105.10396*, 2021.

[15] KUCHNIK, Michael, SMITH, Virginia, et AMVROSIADIS, George. Validating Large Language Models with ReLM. *arXiv preprint arXiv:2211*.15458, 2022.

[16] Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. *In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021, pp. 3816–3830.