Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/43/20230819

Automated arduino robot design with multi-level natural language processing chains

Shengqi Zhang

Department of Bioengineering, Royal School of Mines, Imperial College London, Exhibition Road, London, SW7 2AZ, UK

Garfield.zhang.gz@gmail.com

Abstract. This research explores the potential of leveraging OpenAI's GPT-3.5-Turbo API for automating Arduino robot design through a structured multi-level language processing chain termed "LangChain." The system breaks down the design process into six stages, from preliminary design sketching to failure analysis. Each stage generates robot design components using user inputs, progressively building upon the previous outputs. The results highlight the model's capabilities in generating functional preliminary designs, suggesting hardware and software components, and offering assembly instructions. While demonstrating a commendable level of technical knowledge, the system requires further refinement in numerical analysis and design reviews. This approach provides a foundational framework for bridging the knowledge gap between amateurs and professionals in robotic design.

Keywords: Arduino Robotics, ChatGPT, Automated Robot Design, Multi-level Language Processing Chain.

1. Introduction

The field of robotics is undergoing rapid transformations, opening up new opportunities in several sectors, including healthcare, manufacturing, education, and agriculture, through the incorporation of robots. However, creating robots demands a profound understanding of various disciplines, such as mechanical engineering, electronics, and programming. This multidisciplinary requirement often poses a formidable barrier for beginners and aspiring innovators.

The advent of the Arduino platform has simplified the robot creation process, making it accessible to hobbyists, students, and professionals alike. This is largely due to its user-friendly nature, open-source environment, and the supportive community surrounding it. Arduino offers a cost-effective approach to robotics, unlocking many opportunities for developing innovative and functional robot designs.

The work of G. M. Debele has demonstrated how Arduino can be utilised to create user-friendly automated systems, illustrating its adaptability in various automation tasks [1]. This technology has also facilitated assistive technology developments, aiding individuals with physical disabilities through brainwave signal-controlled home automation systems conceived by K. M. Chandra Babu [2].

Despite its merits, designing and programming a robot can still be intimidating, especially for newcomers. Integrating technologies such as natural language processing (NLP) and AI can further simplify this process, broadening the accessibility to robotic development.

DOI: 10.54254/2755-2721/43/20230819

Recent advancements in NLP, including the introduction of OpenAI's GPT-3.5-Turbo API, offer promising avenues in robotics. For instance, S. Said and the team leveraged this technology to develop "Adam", an interactive animatronic robot, showcasing the potential for enhanced human-robot interactions through affordable solutions [3]. Moreover, this API has contributed significantly to education, assisting in student engagement during programming lessons, as evidenced in a study by B. Banić [4]. It has also found applications in the healthcare sector, optimising post-colonoscopy patient management, a critical stride in handling intricate logical concepts crucial in robotic design automation [5].

We propose a structured approach leveraging a multi-level language processing chain to simplify the Arduino robot development process through several GPT-3.5-Turbo APIs, each performing a specific function in the development process. This framework builds upon the potent capabilities of ChatGPT in aiding instructional methodologies, as explored by A. J. Spasić in lesson preparation techniques [6]. This initiative encapsulates the active role of GPT-3 based solutions in server security analysis, as highlighted by N. Petrović, emphasising the capability to detect suspicious activities through automated log analysis, albeit with existing limitations that necessitate further enhancements [7].

Our strategy encompasses multiple stages, from design sketching to virtual scenario creation for failure analyses, pivoting on the prowess of ChatGPT in various facets, such as bug-fixing performance in software development analysed by D. Sobania et al., which portrays a competitive edge against other deep learning methods [8]. Furthermore, the venture utilises the substantial success exhibited by ChatGPT in discerning misinformation from real news content, a capability tested extensively by K. M. Caramancion [9]. Moreover, the system integrates the concepts of generating data visualisations through natural language inputs, an area explored profoundly by P. Maddigan, into the development process to further simplify the user interaction with the system [10].

By tapping into the multi-faceted potential of ChatGPT, this comprehensive strategy aims to foster an innovative environment, aiding users, irrespective of their technical expertise in robotics, to bring their concepts to fruition with minimal hurdles.

2. Method

The research methodology encompassed designing and developing an automated design framework for Arduino robots. This was achieved using the LangChain model, a structured multi-level language processing chain that harnesses the capabilities of GPT-3.5-Turbo API. The methodology aimed to provide an automated, step-by-step, and user-friendly approach to Arduino robot design, bridging the knowledge gap between amateurs and professionals. For detailed source code, developers should refer to the repository at

https://github.com/GarfieldGZ/Automated-Arduino-Robot-Design-with-Multi-Level-Natural-Language-Processing-Chains

2.1. Framework and Environment Setup

The initial step in establishing the research methodology involved integrating the *LangChain* framework, a system engineered to aid the creation of applications propelled by advanced language models. Essential packages, including *openai* and *dotenv*, were incorporated; the former facilitated seamless interaction with the GPT-3.5-Turbo API, leveraging OpenAI's vast language processing services, while the latter ensured secure management of environment variables, thus protecting sensitive data such as API keys from unauthorised access. The *os* package was enlisted to manage environment variables effectively at the operating system level. Secure and efficient communication was established through a dedicated API endpoint, https://api.chatanywhere.com.cn/v1, chosen to handle the service requests, setting a secure foundation for the detailed, multi-level language processing chain.

2.2. Structured Multi-Level Language Processing Chain

In AI language generation, "temperature" refers to the randomness incorporated in the generated text: a higher temperature results in more random outputs, while a lower temperature yields more deterministic

DOI: 10.54254/2755-2721/43/20230819

results. “Prompts” are the inputs given to the AI to guide the generation process, essentially instructing it on the kind of content to produce. The following delineates a multi-stage approach, utilising the capabilities of the GPT-3.5-Turbo API to streamline Arduino robot development.

2.2.1. Preliminary Design (Stage 1)

In the initial stage, the system interprets the user prompt to draft a rudimentary design sketch. Setting the model’s temperature to 0.9 encourages a balance between creativity and focus, assisting in formulating a conceptual sketch that outlines the envisioned robot’s principal type and key components. This primary sketch is grounded on the user’s brief description, establishing a foundational blueprint for the hardware design in the next stage.

2.2.2. Hardware Design (Stage 2)

Building upon the insights garnered from the preliminary design, stage 2 delves deeply into the hardware necessities of the robot. Keeping the temperature at 0.9, the model undertakes a detailed enumeration of the hardware components, including the requisite processors, sensors, and actuators, and maps out their interconnections to secure optimal functionality. Therefore, this stage crystallises the hardware framework needed for the envisaged robot, setting the stage for the following software design.

2.2.3. Software Design (Stage 3)

Utilising the hardware specifications established previously, this stage revolves around crafting the software architecture that would govern the Arduino robot’s functionalities. Maintaining a 0.9 temperature setting, the model leans on the information derived from stages one and two to draft the Arduino software framework, which establishes the control logic fundamental for the robot’s operations, enabling the user to directly implement the code for controlling the robot.

2.2.4. Design Review (Stage 4)

Stage 4 is a critical juncture in the chain, meticulously scrutinising the outputs from the preceding stages to ascertain adherence to safety norms and industry best practices. A reduced temperature setting of 0.7 enhances focused and detailed scrutiny, facilitating a rigorous review process that identifies areas necessitating improvements and potential safety hazards, thereby steering the project towards refined and reliable outcomes.

2.2.5. User Instruction (Stage 5)

This penultimate stage is dedicated to transmuting the technical outcomes of the earlier stages into user-friendly guidelines. With the model operating at a temperature of 0.9, it synthesises the data accrued to create a coherent, step-by-step guide that aids users in assembling and operating the robot effectively, thereby elevating the user experience by proffering clear and articulate assembly and operational directives.

2.2.6. Failure Analysis (Stage 6)

Before ushering the robot into real-world applications, this final stage undertakes a preemptive analysis of potential operational failures by simulating diverse virtual scenarios. Operating at a lower temperature of 0.5, it fosters intricate and precise analyses within a virtual testing arena that emulates real-world physics to evaluate potential malfunctions and proffer viable solutions, ensuring a robust and resilient robot ready for real-world deployment. This stage embodies a proactive approach, addressing issues before they manifest in real-world scenarios, thus paving the way for a refined and resilient robotic solution.

2.3. Sequential Chains

In the customised LangChain model, every stage is delineated as a separate entity referred to as an LLMChain, functioning within the larger LangChain framework. This configuration is designed to

DOI: 10.54254/2755-2721/43/20230819

facilitate the sequential chaining of prompts, a mechanism central to the system's operational efficiency. This approach, denominated as "Sequential Chain," warrants a seamless transition where the output derived from one stage is automatically utilised as the input for the subsequent stage. This strategy ensures a structured and logical progression through the various phases of the robot design process, promoting a systematic and coherent workflow.

2.4. Output Format

For better presentation, the outputs were formatted with a touch of aesthetics. Using the *Colorama* library, each stage output was distinctly color-coded. This differentiation aims to enhance clarity, making the data interpretation more intuitive.

3. Results

To verify the effectiveness and performance of the model, three user prompts were chosen for input into the system, which then generated three corresponding robot designs. These designs are a line follower robot, a fire-fighting robot, and an Arduino robot arm. On the whole, all the design stages were successfully executed and created. For an in-depth look at the generated content, readers are encouraged to visit the repository at

https://github.com/GarfieldGZ/Automated-Arduino-Robot-Design-with-Multi-Level-Natural-Language-Processing-Chains

In examining the robotic designs generated through the Multi-Level Language Processing Chain utilising the GPT-3.5-turbo model, it is evident that the model has potential in the initial stages of robotic design through a structured methodological approach. However, a thorough inspection of the generated outputs exposes the strengths and weaknesses inherent in this automated approach to design generation.

3.1. Preliminary Design (Refer to LineFollower.txt in the repository)

The foundational type outlined for the robot is a wheeled mobile robot that operates primarily as a line follower, utilising infrared sensors to detect and follow lines, particularly a black line on a light-coloured surface, as evidenced by line 10. This points to the robot's fundamental tracking line functionality through specific colour contrasts detected by the infrared sensors installed at its frontal part.

The secondary module incorporated in the design is an obstacle avoidance sensor charged with detecting impediments in the robot's path and initiating evasive actions. This module leverages ultrasonic sensor technology positioned at the robot's front to identify obstacles, a choice attributed to the sensor's reliable detection range, as noted in line 13.

The motor controller is central to the robot's functioning and is designed to regulate the speed and direction of the robot's motors through commands received from an Arduino board connected to a motor driver module, as line 16 describes. The motor driver will be connected to the Arduino board and receive commands to control the motor." This design choice underscores the necessity for seamless, responsive control over the robot's mobility features, mediated by a motor driver module that ensures optimal operation of the motors.

The design anticipates using a rechargeable battery as the power source module to supply the robot with a stable energy flow. The module is connected to the Arduino board via a voltage regulator, facilitating a steady power supply, as stated in line 19. This choice showcases the priority of achieving sustained, stable operations through a reliable power source.

At the outset, it is essential to note that the success of the preliminary stage significantly impacts the entire chain, given that it sets the trajectory for the following stages. While the preliminary design is typically briefer, the integrity of subsequent stages is highly reliant on the quality of the content at this initial phase. The preliminary design generated by the model is generally adequate, featuring accurate core components and a sufficient comprehension of their functions. Nevertheless, it should be noted that the model is yet to exhibit a high level of creativity.

DOI: 10.54254/2755-2721/43/20230819

3.2. Hardware and Software Design (Refer to LineFollower.txt in the repository)

The hardware sections outline the hardware design of a mobile robot utilising an Arduino board as the central controlling unit, delineating various components, their functionalities, and connections.

The "Physical Structure" section lays down the foundational aspects of the robot, specifying the dimensions and the materials utilised in its construction. As line 39 illustrates. This part showcases the physical attributes that are aimed at achieving stability and the ability to manoeuvre in tight spaces.

The "Components" section introduces the four primary components crucial for the robot's functioning. First, the line follower sensor has three infrared sensors, which detect lines connected to specific pins on the Arduino board, as noted in line 58. Following this, an obstacle avoidance sensor is portrayed, utilising an ultrasonic sensor for detecting obstructions in the robot's path. Next, the motor controller and power source details highlight the use of an L293D motor driver module and a rechargeable lithium-ion battery safeguarded by a voltage regulator to maintain a steady power supply.

Subsequently, in the "Connections" subsection, the detailed connections between different components and the Arduino board are elucidated, establishing how sensors, motor drivers, and power sources are linked through specified pins on the Arduino board using jumper wires, as specified in line 64.

Overall, the hardware and software design stages demonstrate a commendable level of technical knowledge, practical feasibility, and design progression, with each stage building upon the insights gleaned from the preceding one.

Nevertheless, upon closer examination, it becomes evident that the hardware stage needs more depth in numerical analysis. This limitation arises because the model can generate design concepts but cannot perform rigorous physics calculations. Consequently, the component selection is primarily driven by functionality rather than quantified performance metrics.

The transition between the hardware and software components is generally successful, with few syntax errors in the code and logically structured code. Overall, the hardware and software designs are valuable references for constructing the robot. However, it is important to note that they may not function optimally out of the box, especially for more intricate and complex designs that necessitate thorough mechanical and circuit analysis, as well as precise calculations.

3.3. Design Review and User Instruction (Refer to LineFollower.txt in the repository)

The design review sections provide critical insights into the robot's design facets. Starting with "Feasibility," the review suggests that while the design appears logical with appropriately chosen modules, a deeper feasibility study would be beneficial. This is evident from line 186. The feasibility study would verify if all components integrate and cooperate efficiently. Under "Reliability," there's a concern about the robot's performance consistency in real-world scenarios, as denoted by line 189. This implies that the robot's operational effectiveness is yet to be verified in various practical situations.

The "Safety" section emphasises the significance of ensuring that the robot operates without causing harm, which can be inferred from line 192. It underscores the importance of the safety of humans and robot safety during operation. Addressing "Design Flaws," the review pinpoints specific shortcomings, like the dual-functionality of a single sensor for obstacle avoidance and line following. This limitation is highlighted in line 195. The line also mentions the restriction in the robot's motion, indicating the need for more flexible movement options.

The "Improvements" section offers solutions to the flaws mentioned above. The recommendation to introduce a dedicated obstacle avoidance sensor and a turning mechanism stems from line 198. The suggestions are valuable to enhance the robot's navigational capabilities. Lastly, in evaluating the software aspect, the review in line 200 points towards potential enhancement in obstacle avoidance: "...the motor speed and direction should be adjusted based on the distance from the obstacle."

In general, the design review demonstrates a commendable ability to address minor flaws in design details and offer logical and sensible improvements. However, it primarily builds upon existing design elements and struggles to detect or rectify significant design flaws that require common sense solutions. Additionally, it must be more capable of providing entirely alternative design solutions.

DOI: 10.54254/2755-2721/43/20230819

The content is detailed during the user instruction stage and carries over from previous design stages. Nevertheless, the intricate assembly and installation procedures are relatively challenging to follow without more intuitive visual aids. The design or instructional diagrams are presently unattainable in a technical sense, as they demand precise specifications, symbols, and markings—areas where automated image generation falls short.

Nonetheless, the reviews and instructions are highly valuable for users with prior mechanical and electronics knowledge. However, individuals entirely new to these subjects may encounter difficulties in comprehension.

3.4. Failure Analysis (Refer to LineFollower.txt in the repository)

The virtual failure analysis represents an experimental alternative approach to enhance the resolution of issues identified during the design review stage.

For instance, from lines 247 to 249, when "The robot encounters a steep incline and cannot climb it," the analysis identifies that "The motors are not powerful enough to climb steep inclines." Accordingly, the proposed solution is to "Upgrade the motors to more powerful ones that can handle steep inclines." This format is consistent throughout the analysis, directly correlating the issue faced, its cause, and a potential rectification measure.

Overall, the virtual failure analysis stage enables the model to comprehensively assess significant design flaws, resulting in a broader perspective that fosters critical and creative viewpoints. The findings indicate that the failure analysis is notably more effective in identifying major design oversights, although it may present challenges in terms of user-friendliness. Future iterations of the model may integrate this analysis stage before the review stage to optimise readability and usability.

4. Conclusion

In conclusion, utilising the GPT-3.5-Turbo API in facilitating a multi-stage Arduino robot design process emerges as a venture with notable potential, carving a pathway for both novices and professionals to delve into robotics with a guided, structured approach. While the early stages of the framework exhibited a robust understanding and generation of fundamental design concepts, it is observed that as the complexity escalates in subsequent stages, there is a discernible decline in the depth of analysis and creativity, signalling a necessity for further advancements in the system's ability to handle intricate design details and comprehensive safety evaluations. Moreover, the lack of capabilities in generating precise numerical and graphical content to assist users hints at considerable room for refinement. Although the LangChain methodology serves as a commendable blueprint in the automated design landscape, translating this embryonic potential into a full-fledged, reliable, and user-friendly tool undeniably demands a more rigorous development trajectory focussed on enhancing depth, creativity, and analytical prowess, fostering a more holistic and resourceful tool for Arduino robotic prototyping. It is a promising endeavour, encouraging a rich ground for exploration and learning, albeit with a significant journey ahead towards refinement and perfection.

References

[1] G. M. Debele and X. Qian, "Automatic Room Temperature Control System Using Arduino UNO R3 and DHT11 Sensor," 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2020, pp. 428-432, doi: 10.1109/ICCWAMTIP51612.2020.9317307.

[2] K. M. Chandra Babu and P. A. Harsha Vardhini, "Brain Computer Interface based Arduino Home Automation System for Physically Challenged," 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India, 2020, pp. 125-130, doi: 10.1109/ICISS49785.2020.9315999.

[3] S. Said, G. AlAsfour, F. Alghannam, S. Khalaf, T. Susilo, B. Prasad, K. Youssef, S. Alkork, and T. Beyrouthy, "Experimental Investigation of an Interactive Animatronic Robotic Head

DOI: 10.54254/2755-2721/43/20230819

Connected to ChatGPT," in BioSMART 2023, 2023, pp. 1-4, doi: 10.1109/BioSMART58455.2023.10162099.

[4] B. Banić, M. Konecki and M. Konecki, "Pair Programming Education Aided by ChatGPT," 2023 46th MIPRO ICT and Electronics Convention (MIPRO), Opatija, Croatia, 2023, pp. 911-915, doi: 10.23919/MIPRO57284.2023.10159727.

[5] Yuri Gorelik, Itay Ghersin, Itay Maza, Amir Klein, "Harnessing Language Models for Streamlined Post-Colonoscopy Patient Management: A Novel Approach," Gastrointestinal Endoscopy, 2023, ISSN 0016-5107, https://doi.org/10.1016/j.gie.2023.06.025.

[6] A. J. Spasić and D. S. Janković, "Using ChatGPT Standard Prompt Engineering Techniques in Lesson Preparation: Role, Instructions and Seed-Word Prompts," 2023 58th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), Nis, Serbia, 2023, pp. 47-50, doi: 10.1109/ICEST58410.2023.10187269.

[7] N. Petrović, "Machine Learning-Based Run-Time DevSecOps: ChatGPT Against Traditional Approach," 2023 10th International Conference on Electrical, Electronic and Computing Engineering (IcETRAN), East Sarajevo, Bosnia and Herzegovina, 2023, pp. 1-5, doi: 10.1109/IcETRAN59631.2023.10192161.

[8] D. Sobania, M. Briesch, C. Hanna and J. Petke, "An Analysis of the Automatic Bug Fixing Performance of ChatGPT," 2023 IEEE/ACM International Workshop on Automated Program Repair (APR), Melbourne, Australia, 2023, pp. 23-30, doi: 10.1109/APR59189.2023.00012.

[9] K. M. Caramancion, "Harnessing the Power of ChatGPT to Decimate Mis/Disinformation: Using ChatGPT for Fake News Detection," 2023 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 2023, pp. 0042-0046, doi: 10.1109/AIIoT58121.2023.10174450.

[10] P. Maddigan and T. Susnjak, "Chat2VIS: Generating Data Visualizations via Natural Language Using ChatGPT, Codex and GPT-3 Large Language Models," in IEEE Access, **vol. 11,** pp. 45181-45193, 2023, doi: 10.1109/ACCESS.2023.3274199.