IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

not really a review style article but I wonder if there is a way to highlight or improve the visibility of your article to a wider audience by a slightly different title - I'm just thinking out loud on this one

Vega: LLM-driven Intelligent Chatbot Platform for Internet of Things Development

Control and

Are you? surely not ?

is there a reason for different capitalisation here?

HARITH AL-SAFI, (Fellow, IEEE), HARITH IBRAHIM, and Paul Steenson, (SeniroMember, IEEE)

School of Electronics and Electrical Engineering, University of Leeds, Leeds LS2 9JT, U.K

yes I'm a Senior Member

Corresponding author: Harith Al-Safi (e-mail: harith.alsafi@gmail.com).

This paragraph of the first footnote will contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456."

needs correcting - obvs - but easy to miss when you are looking at a document for so long you can miss the obvious u are exhausted and suffering with word-blindness after looking at a familiar do

ABSTRACT Large language models (LLMs) have revolutionized natural language processing, yet their potential in Internet of Things (IoT) and embedded systems (ESys) applications remains largely untapped. Traditional IoT interfaces often require specialized knowledge, creating barriers for non-technical users. We present a modular system that leverages LLMs to enable intuitive, natural language control of IoT devices, specifically a Raspberry Pi (RPi) connected to various sensors and devices. Our solution comprises three key components: a physical circuit with input and output devices, an RPi integrating a control server, and a web application integrating LLM logic. Users interact with the system through natural language commands, which the LLM interprets to call appropriate commands for the RPi. The RPi executes these instructions on the connected circuit, with outcomes communicated back to the user via LLM-generated responses. We empirically evaluate our system's performance across a range of task complexities and user scenarios, demonstrating its ability to handle complex, conditional logic without additional RPi-level coding. Our findings reveal that LLM-driven IoT control can effectively bridge the gap between complex device functionality and user-friendly interaction. We discuss the system's scalability, exploring its potential applications in diverse settings such as smart homes, industrial monitoring, and educational environments. By enabling natural language interaction with IoT devices, our approach not only enhances accessibility for non-technical users but also opens new avenues for creative and intelligent IoT applications. This research contributes to the growing body of work on interactive intelligent systems for IoT, offering insights into the design and implementation of LLM-integrated IoT interfaces.

and interrogation

actuators - natural antonym of sensor

remotely

The systems performance is empirically evaluated using a range ..

the system's

some

Key to this significance is that by ..

and reduces system development time for

and intuitive intelligent..

(just how significant is this statement - without additional RPi-level coding - seems highly significant useful to me, so do you need to draw more attention to this for the general reader - " ... without additional RPi-level coding, which is highly significant because.. "- and explain very briefly why. Or ... because our findings show that LLM...

INDEX TERMS Enter key words or phrases in alphabetical order, separated by commas. Autocorrelation, beamforming, communications technology, dictionary learning, feedback, fMRI, mmWave, multipath, system design, multipath, slight fault, underlubrication fault.

we can work on this

I. INTRODUCTION

Large language models (LLMs) have revolutionized natural language processing, demonstrating unprecedented capabilities in understanding and generating human-like text [1]. However, their potential in Internet of Things (IoT) and embedded systems (ESys) applications remains largely untapped. IoT systems have become increasingly prevalent across various domains, from smart homes to industrial automation [2]. Despite their widespread adoption, developing and interacting with adaptive IoT systems often requires specialized knowledge and programming skills, creating significant barriers for non-technical users [3].

good

new or

Traditional IoT interfaces typically rely on graphical user interfaces (GUIs) or specific programming languages, which can be challenging for users without technical expertise [3].

to develop

This limitation hinders the widespread adoption and utilization of IoT technologies, particularly in scenarios where rapid deployment and intuitive interaction are crucial. While research has been conducted on natural language interfaces for IoT, the application of advanced language models to IoT control and interaction remains an underexplored area [4].

user

in inverted commas Vega"

To address these challenges, we propose Vega, an intelligent chatbot platform that leverages LLMs to enable intuitive, natural language control of IoT devices. Our system focuses on a Raspberry Pi (RPi) connected to various sensors and devices as a representative IoT setup. By integrating LLMs with IoT infrastructure, we aim to bridge the gap between complex device functionality and user-friendly interaction, allowing users to control and query IoT systems using everyday language.

and security issues notwithstanding,

Our research builds upon recent advancements in LLMs, specifically OpenAI's GPT-based models [5], which utilize transformer neural network architectures to capture context and relationships within text data. By applying these powerful language understanding capabilities to IoT interaction, we aim to create a more accessible and flexible approach to device control and monitoring. Our approach not only enhances accessibility for non-technical users but also opens new avenues for creative and intelligent IoT applications, addressing the standardization challenges highlighted by Al-Qaseem [6].

Vega's architecture comprises three key components: a physical circuit with input and output devices, an RPi integrating a control server, and a web application incorporating LLM logic. This modular design allows for flexibility and scalability, enabling the system to adapt to various IoT scenarios and user requirements [7]. By utilizing the RPi as a central hub, we can leverage its versatility and widespread adoption in the IoT community [8].

The main contributions of this paper are as follows:

1) We present an exemplar modular architecture that integrates LLMs with IoT systems, specifically designed for natural language interaction with RPi-based setups.
2) We develop a novel approach for translating natural language commands into executable instructions for IoT devices, capable of handling complex, conditional logic without additional RPi-level coding.
3) We implement and evaluate a generalised prototype system demonstrating the feasibility and effectiveness of LLM-driven IoT control across a range of task complexities and user scenarios.
4) We provide insights into the scalability and potential applications of our approach in diverse settings such as smart homes, industrial monitoring, and educational environments.

The remainder of this paper is organized as follows: Section II provides background information and discusses related work in IoT interfaces and natural language processing. Section III details our methodology, including the overall system architecture, physical circuit design, RPi configuration, and web application implementation. Section IV presents our experimental setup, results, and analysis, showcasing the system's performance in handling complex commands and its potential real-world applications. Finally, Section V concludes the paper and outlines directions for future research.

II. BACKGROUND AND RELATED WORK

A. INDUSTRIAL APPLICATIONS OF LLM'S

LLM's have revolutionized natural language processing, with the Transformer architecture [9] serving as a foundational breakthrough. These models, trained on vast corpora*, have demonstrated remarkable versatility across diverse domains, including robotics and IoT applications.

Maddiga et al. [10] showcased this versatility with Chat2VIS, leveraging ChatGPT and GPT-3 to generate data visualizations from natural language queries. Their innovative approach demonstrated how LLM's could be effectively used to convert free-form natural language directly into visualization code, even when queries were highly misspecified or underspecified. Meanwhile, Gupta et al. [11] explored the dual-edged implications of ChatGPT in cybersecurity, coining the term "ThreatGPT" to highlight potential risks. Their research delved into both the offensive and defensive applications of generative AI in cybersecurity. As they demonstrated how malicious actors could exploit ChatGPT's capabilities for creating social engineering attacks. They also examined how these same tools could be used to enhance cybersecurity measures, such as improving threat intelligence.

Recent research has explored the integration of LLMs with robotic systems, paving the way for intuitive human-robot interaction. Singh and Blukis [12] introduced ProgPrompt, a novel approach leveraging LLMs to generate action sequences based on natural language instructions. By prompting LLMs with program-like specifications of available actions and objects, along with example programs, their method enables plan generation across diverse environments, robot capabilities, and tasks. This work demonstrated state-of-the-art success rates in VirtualHome household tasks and was successfully deployed on a physical robot arm for tabletop tasks.

Expanding on this concept, Driess et al. [13] proposed PaLM-E, an embodied multimodal language model that incorporates real-world sensor data into language models. PaLM-E is trained on tasks such as robotic manipulation planning and visual question answering, exhibiting positive transfer across language, vision, and visual-language domains. This research highlights the potential of LLMs in grounding language understanding in physical environments, a crucial aspect for IoT applications.

In the context of multi-agent systems, Kannan et al. [14] developed SMART-LLM, a framework for embodied multi-robot task planning. This approach uses LLMs to convert high-level task instructions into multi-robot task plans through a series of stages, including task decomposition, coalition formation, and task allocation. The authors created a benchmark dataset for validating multi-robot task planning problems, demonstrating the framework's effectiveness in both simulated and real-world scenarios.

Wu et al. [15] presented TidyBot, a system that combines language-based planning and perception with LLMs to infer generalized user preferences for household cleanup tasks. This research demonstrates the potential of LLMs in personalizing robot assistance, achieving 91.2% accuracy on unseen objects in their benchmark dataset and successfully putting away 85.0% of objects in real-world test scenarios.

While these advancements primarily focus on robotics, they lay a solid foundation for extending similar techniques to IoT scenarios. The ability to interpret natural language instructions, generate action sequences, and integrate multimodal sensor data holds significant potential for enabling intuitive and intelligent control of IoT devices and systems.

do you need to add .. vast corpora of data, ? ..

As research progresses, we anticipate further innovations in LLM-driven IoT interfaces, potentially revolutionizing how users interact with smart environments.

B. NATURAL LANGUAGE PROCESSING FOR IOT

Natural Language Processing (NLP) has emerged as a transformative technology in IoT applications, enabling intuitive human-machine interactions. The integration of NLP in IoT systems allows users to control and query devices using everyday language, bridging the gap between complex technological interfaces and user-friendly experiences [16]. This integration is particularly crucial as IoT devices become ubiquitous in various domains, from smart homes to industrial settings, where ease of use and accessibility are paramount.

Recent research has demonstrated the potential of NLP in IoT contexts. For instance, Petrović et al. explored the use of ChatGPT in IoT systems, focusing on Arduino-based applications [17]. Their work highlighted the possibilities of leveraging LLMs for both question-answering and automated code generation in IoT environments. Similarly, Zhong et al. proposed CASIT, a collective intelligent agent system for IoT that utilizes LLMs to process and interpret data from multiple sources efficiently [18]. These studies underscore the growing interest in applying advanced NLP techniques to enhance IoT functionality and user experience.

The integration of LLMs represents a significant advancement in NLP capabilities for IoT. Traditional NLP methods often struggle with context understanding and complex query interpretation, limitations that LLMs can overcome. LLMs offer improved natural language understanding, enabling more nuanced and context-aware interactions with IoT devices. For example, King et al. demonstrated how LLMs can interpret under-specified commands in smart home environments, translating vague user intentions into specific device actions [16].

The potential of LLMs in IoT extends beyond simple command interpretation. They can enable more sophisticated applications such as predictive maintenance, anomaly detection, and personalized user experiences. Sarzaeim et al. explored the use of LLMs in smart policing systems, showcasing their potential in complex data analysis and pattern recognition [19]. This application hints at the broader possibilities of LLMs in IoT, where they could be used to analyze and interpret vast amounts of sensor data, making IoT systems more intelligent and proactive.

However, integrating LLMs into IoT systems also presents challenges, including privacy concerns, computational requirements, and the need for domain-specific training. Despite these challenges, the potential benefits of LLM-enhanced NLP in IoT are significant. As demonstrated by Xu et al., natural language interfaces can greatly improve the usability of IoT platforms, allowing for more complex and nuanced interactions [20]. By leveraging the advanced capabilities of LLMs, future IoT systems could offer unprecedented levels of intuitive control and intelligent automation, paving the way for more accessible and powerful IoT applications across various domains.

C. CHAT ORIENTED ARCHITECTURES

Chatbots have gained significant traction across various industries, serving as direct communication channels between companies and end-users [21]. However, existing frameworks often require advanced technical knowledge for complex interactions and lack flexibility in adapting to evolving company needs. The deployment of chatbot applications typically demands a deep understanding of targeted platforms, particularly back-end connections, which increases development and maintenance costs [21].

To address these challenges, researchers have proposed novel approaches to chatbot development. Xatkit, for instance, offers a set of Domain Specific Languages to define chatbots in a platform-independent manner, along with a runtime engine for automatic deployment and conversation management [21]. Similarly, SPADE 3 presents a redesigned middleware to support the new generation of multi-agent systems, aiming to popularize agent technology as a dynamic and flexible solution to current problems [22].

Recent studies have explored multi-modal chatbots in intelligent manufacturing settings, demonstrating the potential for AI-powered dialogue systems to assist users in complex assembly tasks [23]. These systems leverage both textual and visual features to improve intent classification and provide relevant information to users. The development of conversation-driven approaches for chatbot management has also shown promise in evolving chatbot content through the analysis of user interactions, allowing for a cyclic and human-supervised process [24].

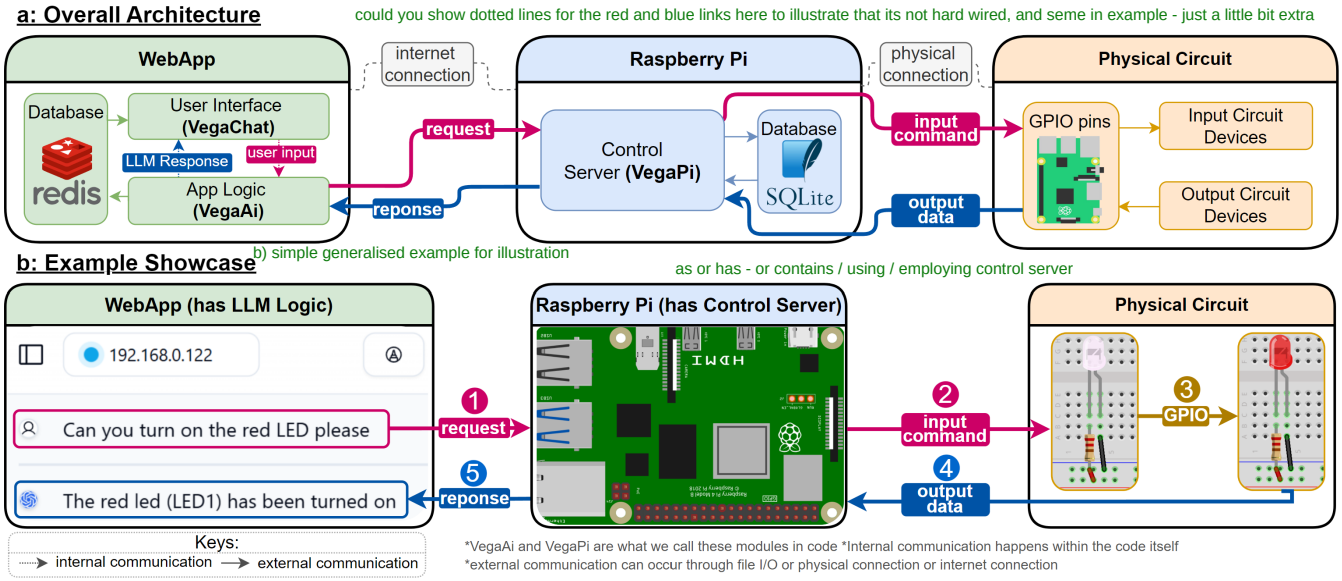
In the realm of human-robot interaction, researchers have developed task-oriented dialogue systems for industrial robots, addressing the lack of domain-specific discourse corpora and emphasizing user experience alongside task completion rates [25]. These efforts have resulted in datasets like IR-WoZ and frameworks such as ToD4IR, which integrate small talk concepts and human-to-human conversation strategies to support more natural and adaptable dialogue environments.

The potential of LLMs in easing chatbot development has been demonstrated through large-scale models that can learn blended conversational skills when provided with appropriate training data and generation strategies [26]. These models have shown improvements in multi-turn dialogue engagingness and humanness measurements, paving the way for more sophisticated chatbot applications in various domains, including IoT systems.

III. METHODOLOGY

A. OVERALL ARCHITECTURE

The architecture of the Vega system follows key principles of software design to ensure scalability, maintainability, and robustness [7]. The system adopts a modular approach, dividing functionality into distinct components with specific purposes. This design promotes code reuse, facilitates testing,



(a) and (b) alongside..

FIGURE 1. Overall architecture of Vega alongside a simple example.

See** > this control is great for simple cause and effect but again to keep the reader engaged - can or did you develop a more complex interaction - involving conditional statements and returning more complex graphical data or averages - you implied as much previously - - is it possible to briefly summarise this here to avoid a more expert / interested reader discontinuing at this point? just a question.

and enhances overall maintenance. The architecture also implements separation of concerns, where different aspects such as user interface, core functionality, and data management are segregated into distinct layers, improving code organization and enabling independent development.

As shown in Figure 1 (a), Vega's architecture comprises three main modules: a Web Application, a RPi, and a Physical Circuit. These modules interact in a client-server model [27], with the Web Application serving as the client and the RPi as the server. The Physical Circuit is connected to the RPi via hardwired connections.

and is overly simplistic as shown merely to illustrate the principle.

The Web Application consists of two primary submodules: the User Interface (VegaChat) and the App Logic (VegaAi). The App Logic incorporates LLM logic for translating user input into commands and generating responses. Redis [28] is employed as a non-relational database to store chat history, messages, and RPi connection states.

The RPi module hosts a Control Server (VegaPi) responsible for parsing requests from the App Logic and executing them on the Physical Circuit. An SQLite database [29], [30] is used to store data extracted from the physical circuit. The Physical Circuit comprises input devices (sensors) and output devices (LEDs, LCDs, etc.) connected to the RPi's General Purpose Input/Output (GPIO) pins.

pertaining to and extracted from..

a multiplicity of sensors

several typical

A typical use case shown in Figure 1 (b) involves a user interacting with the Web Application interface, sending a natural language command such as "Turn on the red LED." The LLM interprets this command and sends the appropriate instruction to the RPi's Control Server. The server then relays the command to the Physical Circuit via GPIO pins. Upon execution, the circuit sends feedback to the RPi, which is then communicated back to the user through the Web Application.

(..typical overly simplistic ..)

translates this to the run code and

The technology stack for Vega has been carefully selected to ensure robustness, scalability, and accessibility [31]. The Web Application is built using React [32] with TypeScript, employing RadixUI [33] for accessible components and TailwindCSS [34] for responsive design. The App Logic utilizes Node.js and integrates with OpenAI's GPT models [5] for natural language processing. The RPi Control Server is developed using Flask [35], a lightweight Python web framework, while the circuit code leverages the RPi library for GPIO interaction.

This architecture enables Vega to bridge the gap between complex IoT functionality and user-friendly interaction. By leveraging LLMs for natural language processing and control, the system opens up new possibilities for intuitive IoT applications in various domains, from smart homes to industrial monitoring and educational environments [8]. The modular design and carefully chosen technology stack ensure that Vega remains adaptable, maintainable, and scalable as IoT applications continue to evolve and expand. **

B. PHYSICAL CIRCUIT DESIGN

The physical implementation of the Vega platform comprises a custom-designed circuit board that interfaces with the RPi, integrating various input and output devices to facilitate IoT and embedded systems applications. This hardware configuration forms the foundation for the natural language-controlled system, enabling users to interact with physical components through LLM-interpreted commands.

As shown in Figure 2, the circuit board incorporates a diverse array of input devices, including an ultrasonic sensor for distance measurement, a limit switch for binary state detection, a temperature and humidity sensor for environmental monitoring, a GPS module for location tracking, and a push button for direct user input [36]. These components collectively provide a rich set of data sources, enabling the system

generalised

to best illustrate the potential, including..

illustrating the system's

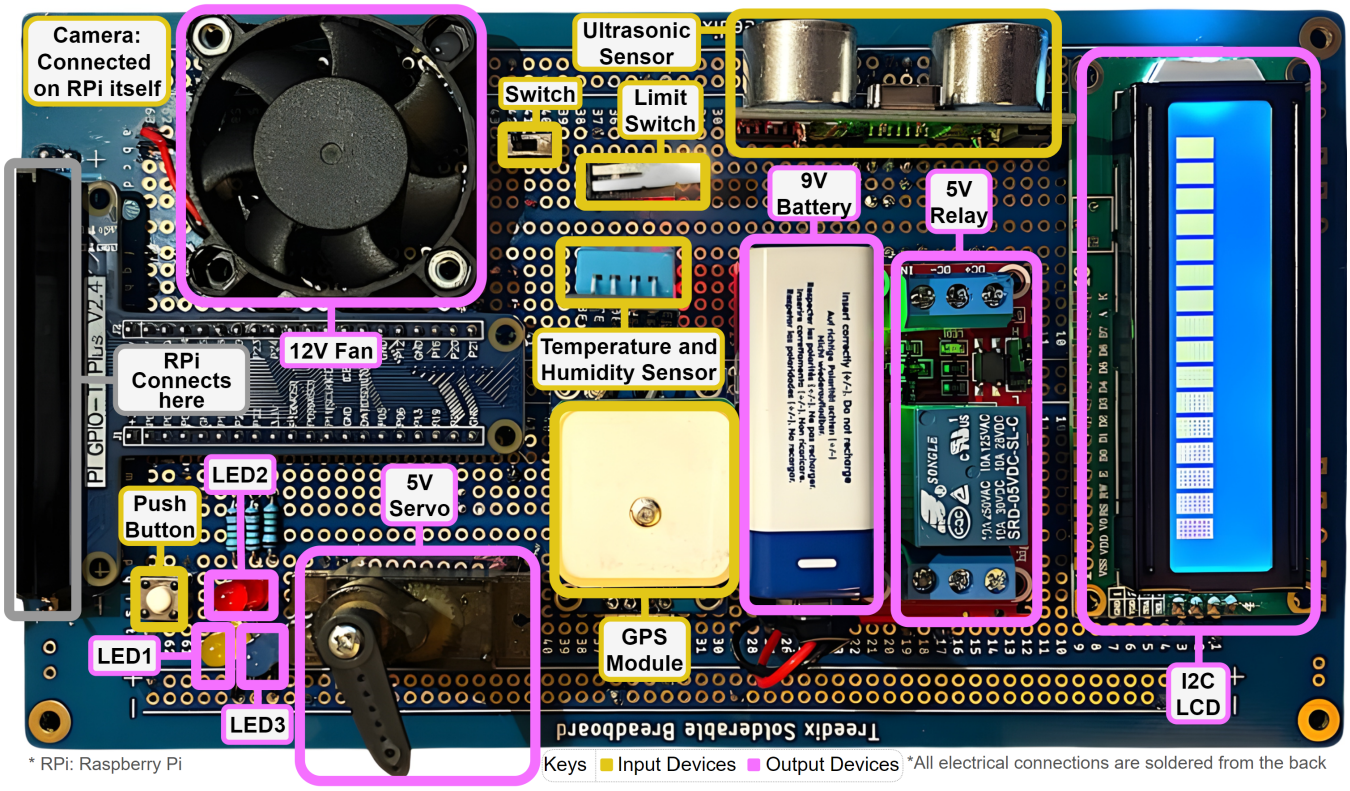


FIGURE 2. Soldered physical circuit connected to the RPi

ability to respond to..

to respond to complex, context-aware queries and commands.

Again a typical array of actuators on the board include but not limited to; ..

Output devices on the board include a 12V fan for cooling or air circulation, multiple LEDs (yellow, red, and blue) for visual indicators, a 5V servo motor for precise rotational control, and an I2C LCD display for text output. A 5V relay is incorporated to control the 12V fan, demonstrating the system's capability to manage higher-voltage components safely [37]. The inclusion of these diverse output devices allows for a wide range of physical responses to user commands, from simple visual feedback to more complex mechanical actions.

in principle to ..

Power management is a crucial aspect of the circuit design. While most components operate on the 5V supply provided by the RPi, the 12V fan requires a separate power source. To address this, a 9V battery is utilized in conjunction with the relay, ensuring proper voltage supply while maintaining RPi-based control [38]. This setup illustrates the system's ability to accommodate components with varying power requirements within a unified control structure.

and just for generalized illustration.

potential

various components and transducers

environment?

The circuit board is designed to connect directly to the RPi's GPIO pins, streamlining the interface between the physical components and the computational core of the system. A camera module, while not physically present on the circuit board, is connected directly to the RPi, expanding the system's capabilities to include image capture and analysis [39].

scene interpretation / analysis as will be reported later

showcases

This hardware configuration supports a wide range of potential applications. In smart home scenarios, the temperature sensor and fan could be used for automated climate control, while the GPS module could enable location-based automation in mobile or outdoor settings. In industrial environments, the ultrasonic sensor and limit switch could be employed for proximity detection and safety systems, with the LEDs and LCD providing status information to operators [40].

and similar the camera module .

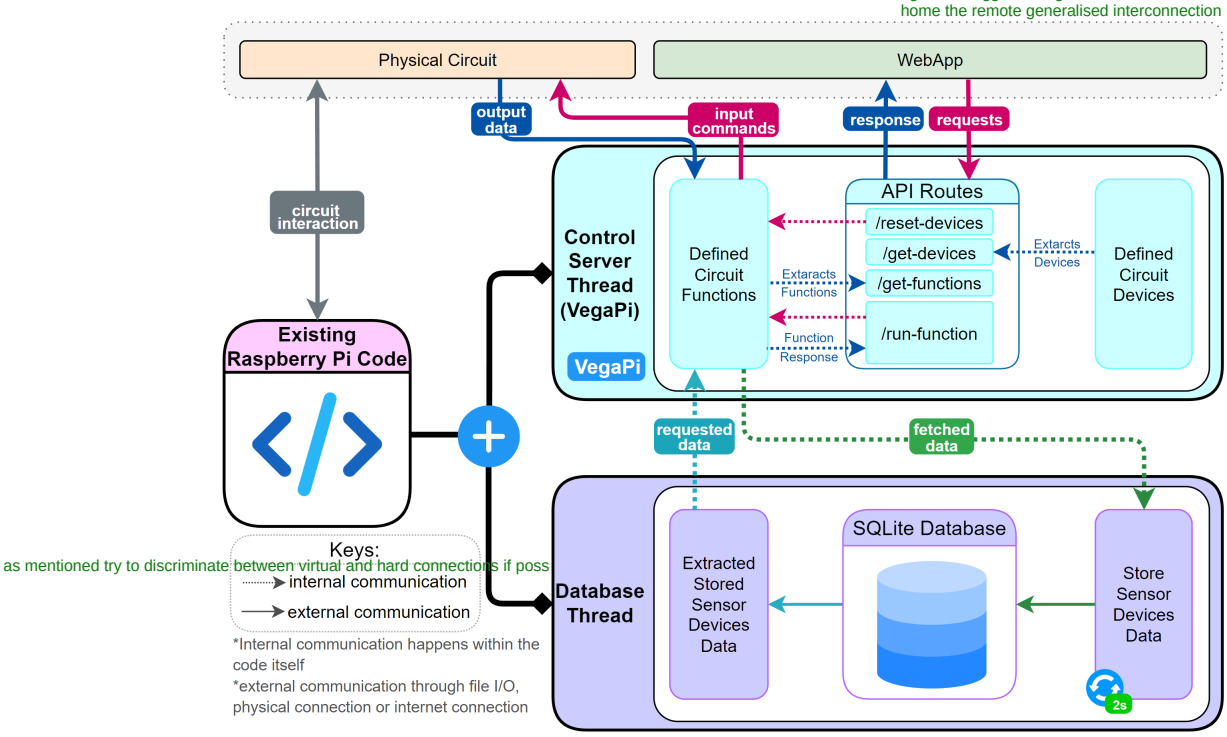
The versatility of this hardware setup, combined with the LLM-driven control system, enables the exploration of complex, conditional logic without requiring additional RPi-level coding. This integration of diverse sensors and actuators with natural language processing capabilities represents a significant step forward in creating intuitive, user-friendly interfaces for IoT and embedded systems, bridging the gap between sophisticated device functionality and accessible user interaction.

C. RASPBERRY PI DESIGN

The architecture of the RPi integration with the existing codebase is designed to enable seamless control and manipulation of the circuit without interfering with pre-existing logic. This approach leverages parallel computing concepts, utilizing processor cores and threads to execute specific logic concurrently with existing code [41].

The system architecture, illustrated in Figure 3, comprises two main threads: the Control Server Thread and the Database Thread. The Control Server Thread manages a Flask-based web framework, storing predefined functions for a set of circuit devices. These functions are exposed through a REST

again I'd suggest using dotted lines for the "remote" control interface to hammer home the remote generalised interconnection



as mentioned try to discriminate between virtual and hard connections if poss

FIGURE 3. Architecture design of the RPi control server

TABLE 1. Physical devices defined on the Control Server, which are then supplied to the LLM

transducer type

Symbol	Type	Description
ULTS	Input	Ultrasonic Distance Sensor in 'cm'
CAM	Input	Camera device for picture input
GPS	Input	GPS device for longitude and latitude coordinates
TMP	Input	Temperature sensor giving response in degree celcuis
FAN	Output	12V fan controled by a digital GPIO pin through a relay
LCD	Output	I2C LCD for displaying strings
SRV	Output	Servo motor rotates using PWM to a given angles
LED1	Output	Yellow LED light
LED2	Output	Red LED light
LED3	Output	Blue LED light

image acquisition?

TABLE 2. Defined functions on the Control Server, called by the LLM based on user input, executes on the RPi and processed on the webapp

Function	Description	Use Case
set_led	Toggles specefic LED	"Turn on yellow LED"
set_fan	Toggles fan on or off	"Turn on the fan"
get_recorded_sensor_data	Gets interval sensor data from database	"Plot me the distance data in last 30 seconds"
get_raspberry_stats	Gets CPU, RAM, Disk of RPi	"What is the currrent disk usage"
capture_image	Capture and upload image to Imgur	"Capture an image, does it contain a pen?"
get_connected_devices	Fetches the data of connected devices	"What is the current humidity and temperature"
get_location_	Gets the current location from GPS	"From the location are we currently in Leeds?"
set_servo_angles	Turn servo to certain angle	"Turn the servo to 10 then 180 degrees"

????

API, facilitating communication between different software systems over the internet [42].

The Database Thread retrieves sensor data at two-second intervals, storing it in an SQLite database. This persistent storage solution ensures data preservation in the event of system failures, enabling data recovery, analytics, and statistical analysis. The stored data can be retrieved upon request and provided to the LLMs in the web application, enhancing system monitoring and diagnostic capabilities.

Table 1 presents the devices defined in the Control Server, categorized as inputs or outputs. This information is stored and transmitted in JSON format via the "get-devices" REST API endpoint. Input devices primarily transmit data for database storage, while output devices receive commands for circuit manipulation.

The Control Server exposes a set of defined functions, listed in Table 2, which the LLM utilizes to determine logic and execute commands on circuit components. These functions are accessible to the LLM through the "get-functions" REST API endpoint. To execute a particular function, the LLM passes the function identifier and required parameters to the web application logic, which then invokes the "run-function" API endpoint.

The choice of REST API over alternative protocols such as MQTT was based on several factors. REST offers simplicity, scalability, and statelessness, making it well-suited for web-based applications [43]. It also provides a uniform interface, enabling easier integration with various client applications. While MQTT excels in low-bandwidth, high-latency environ-

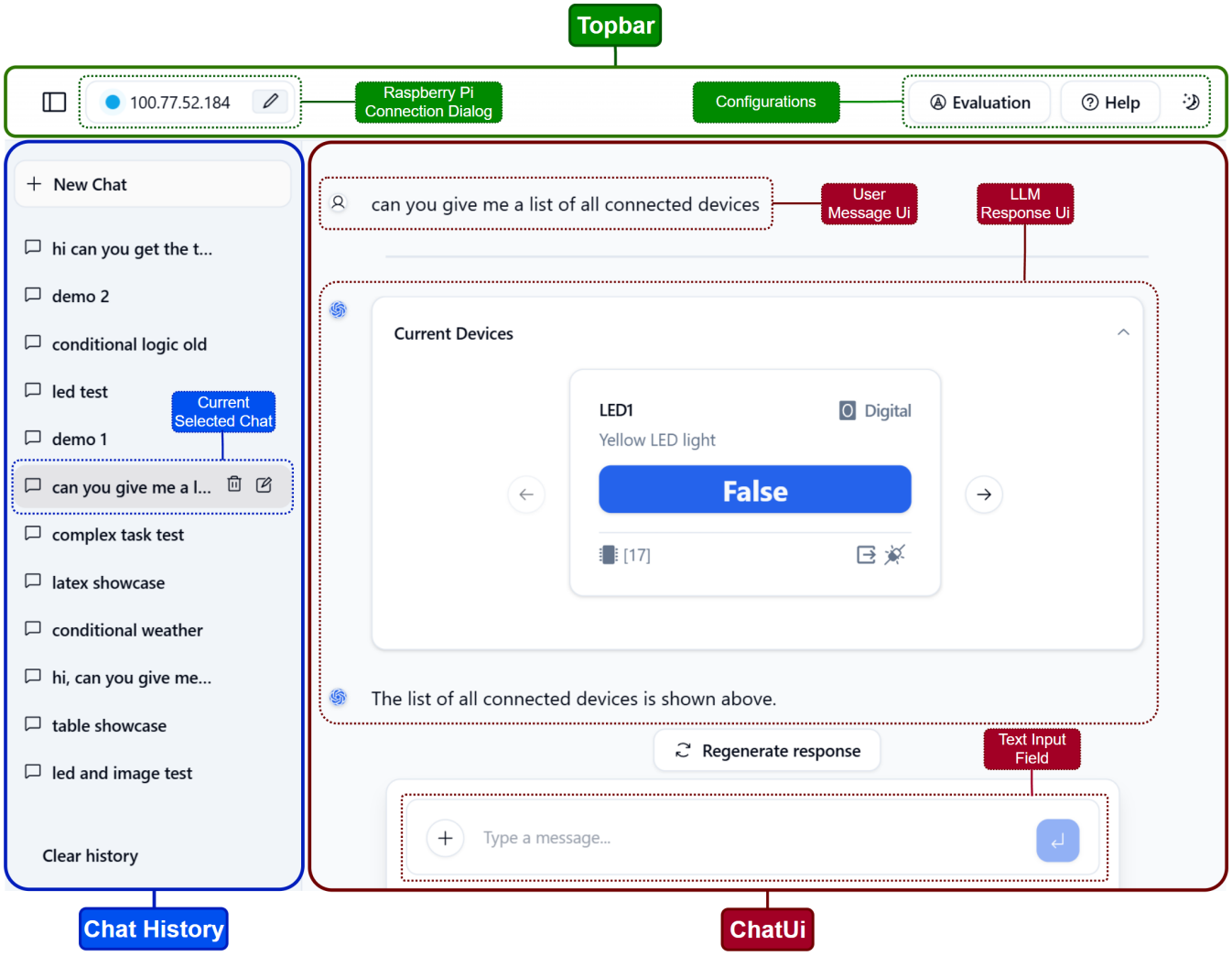


FIGURE 4. **Webapp user interface implementation**

ments, the current system architecture prioritizes the flexibility and widespread support offered by REST APIs in web development ecosystems.

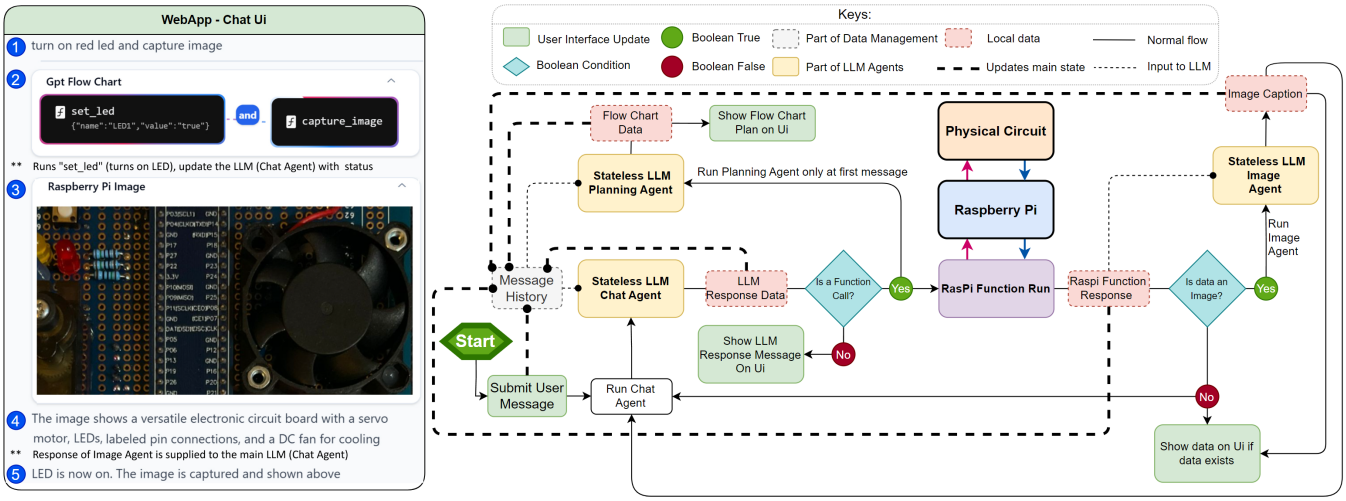
The communication flow between the web application and the RPi follows a request-response pattern. The web application sends REST API requests with JSON data specifying the function and arguments for the RPi to execute. The RPi processes these requests, executes the specified functions, and returns JSON responses with the execution status to the web application. This bidirectional communication enables real-time control and monitoring of the IoT devices.

This architecture facilitates a modular and extensible system, allowing for easy addition of new devices and functions. It also provides a layer of abstraction between the physical hardware and the LLM-driven interface, enabling natural language control of IoT devices without requiring users to understand the underlying technical details. The integration of LLMs with this IoT control system represents a significant step towards more intuitive and accessible IoT interfaces, potentially broadening the application of IoT technologies across various domains [7].

D. WEB APP USER INTERFACE

The web application forms the core of the Vega platform, initiating all LLM processing and circuit manipulation tasks. Its architecture is modular, separating the User Interface (VegaChat) from the App Logic (VegaAi). The User Interface shown in Figure 4 comprises a Top Bar with RPi connection management and configuration options, and a Chat UI displaying LLM responses and user messages. A Chat History UI manages previous interactions. The App Logic includes Data Management, RPi Bridge, and LLM Agents components, handling data entities, RPi communication, and LLM processing respectively.

The UI design prioritizes usability, drawing inspiration from established chatbot interfaces [5]. It features a sidebar for chat history, a top bar for configurations, and a main chat area. An automated mode facilitates efficient testing for



trivial thing but is the read light shown illuminated here?

FIGURE 5. Webapp logic design

advanced users while maintaining simplicity for novices. The interface incorporates a Markdown renderer to appropriately display formatted text generated by the LLM.

The platform supports various data types and formats to enhance user interaction. It can display GPS data as maps, sensor readings as plots, and camera module output as images. Additionally, it visualizes LLM-generated plans as flow charts. This versatility allows the interface to accommodate diverse IoT devices and sensors, presenting their data in intuitive, visual formats.

By leveraging OpenAI's API, the application accesses LLM capabilities without the substantial computational overhead of local hosting [44]. This design choice enhances the platform's accessibility and scalability, enabling its deployment across a wide range of devices and use cases in IoT and embedded systems development.

E. WEB APP LOGIC

The Application Logic component serves as the core operational engine of the Vega system, managing communication with the RPi Control Server and integrating LLM's for natural language processing and command interpretation. This component acts as a bridge between external elements, orchestrating the flow of information and translating user input into appropriate actions within the system architecture [**?**].

??

To establish a connection between the web application and the RPi, the user provides the IP address and port number of the RPi running the Control Server. The web application then initiates concurrent API calls to fetch circuit functions and device information from the Control Server. Upon receiving responses, the connection state is updated, synchronizing the "Raspi Devices" and "Raspi Functions" states which are fed to the LLM.

Figure 5 illustrates the system's workflow, demonstrating how user commands are processed through various stages involving LLM agents. When a user inputs a natural language command (e.g., "turn on red LED and capture image"), the system follows these steps:

1) The LLM Planning Agent generates a flowchart visualizing the planned steps.
2) The Stateless LLM Chat Agent processes the message and determines if a function call to the RPi is necessary.
3) If required, the function is executed on the RPi, which returns a response.
4) For image data, the Stateless LLM Image Agent analyzes and generates a description.
5) Results are displayed on the web application's UI, providing feedback to the user.

The system employs three main LLM agents: Chat, Planning, and Image. The LLM Chat Agent, as depicted in Figure 6, operates in two scenarios. In a normal chat scenario, it processes user input and generates a textual response. In a function call scenario, it recognizes the need for a hardware

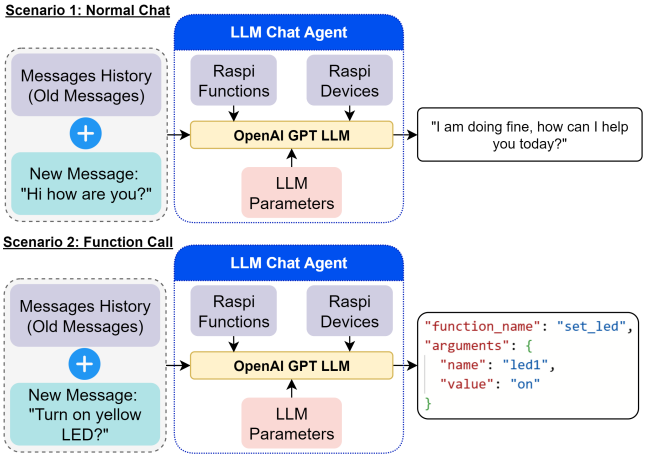
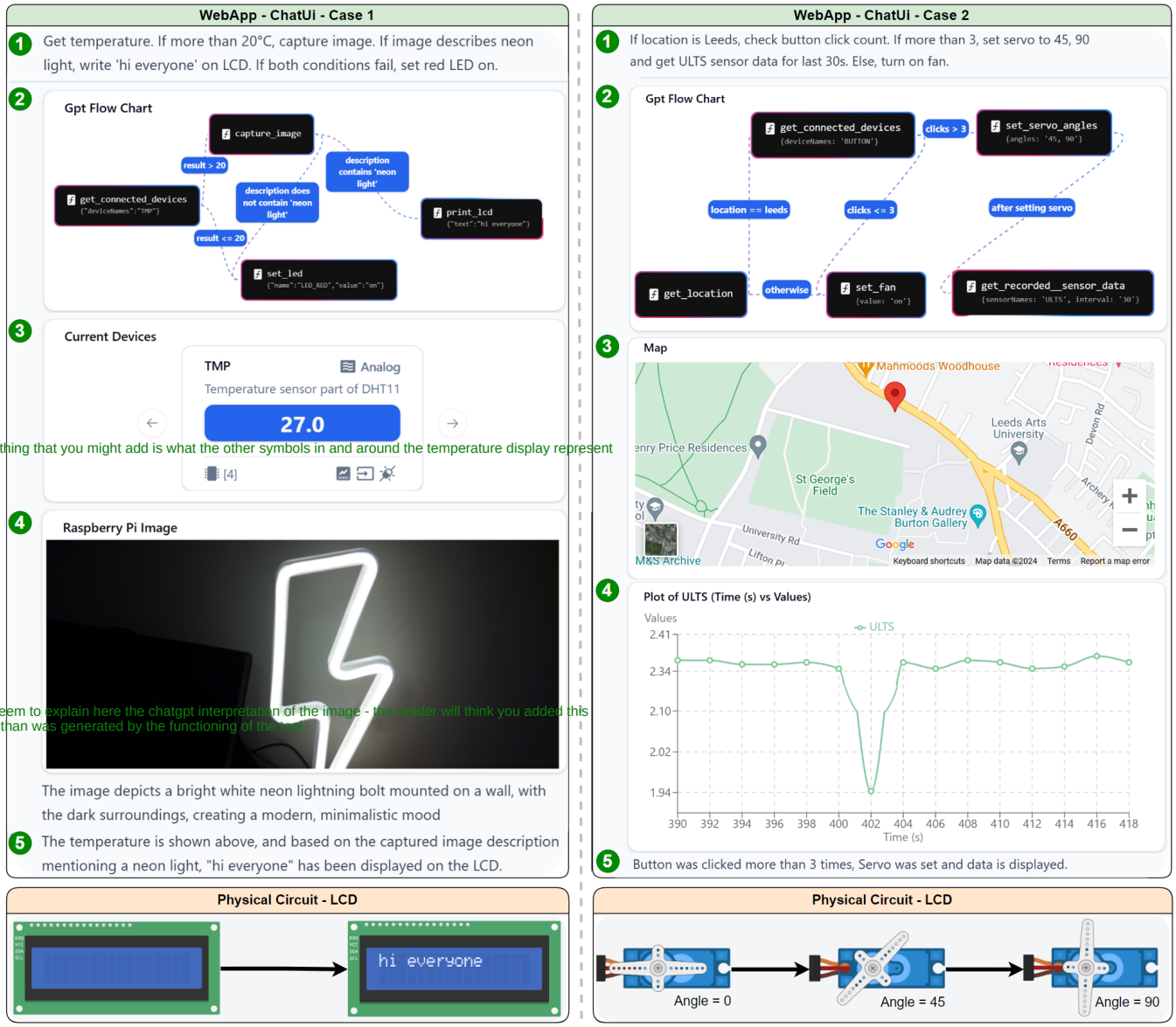


FIGURE 6. Execution of the LLM Chat Agent



only thing that you might add is what the other symbols in and around the temperature display represent

dont seem to explain here the chatgpt interpretation of the image - the reader will think you added this rather than was generated by the functioning of the tool

FIGURE 7. System case studies

action and outputs a JSON-formatted function call for the RPi.

This approach of using LLM agents for command interpretation and execution offers several advantages over traditional code generation methods. It enhances scalability and adaptability, allowing for easy integration of new sensors and data types without significant system modifications [45]. Additionally, it improves system security by limiting direct code execution on the RPi, instead relying on predefined functions interpreted by the LLM.

IV. EXPERIMENT AND RESULTS

A. COMPLEX COMMANDS IN ACTION

B. AUTOMATED EVALUATION

C. RESULT ANALYSIS

D. REAL LIFE APPLICABILITY

V. CONCLUSION AND FUTURE WORK

ACKNOWLEDGMENT

REFERENCES

[1] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3), mar 2024.

[2] Yongxin Liao, Eduardo de Freitas Rocha Loures, and Fernando Deschamps. Industrial internet of things: A systematic literature review and insights. *IEEE Internet of Things Journal*, 5(6):4515–4525, Dec 2018.

[3] Lukas A. Flohr, Sofie Kalinke, Antonio Krüger, and Dieter P. Wallach. Chat or tap? – comparing chatbots with 'classic' graphical user interfaces for mobile interaction with autonomous mobility-on-demand systems. In *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*, MobileHCI '21, New York, NY, USA, 2021. Association for Computing Machinery.

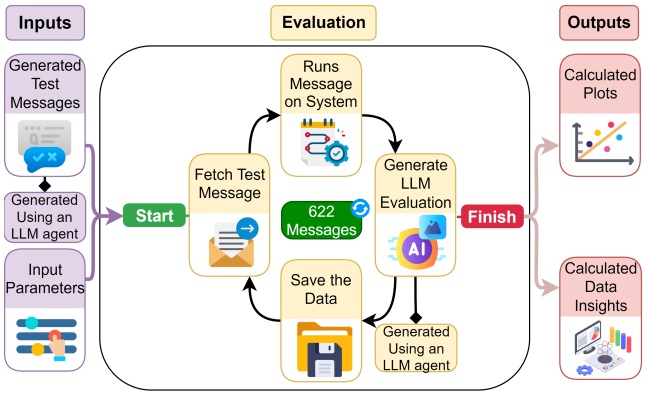


FIGURE 8. **Magnetization as a function of applied field. It is good practice to explain the significance of the figure in the caption.**

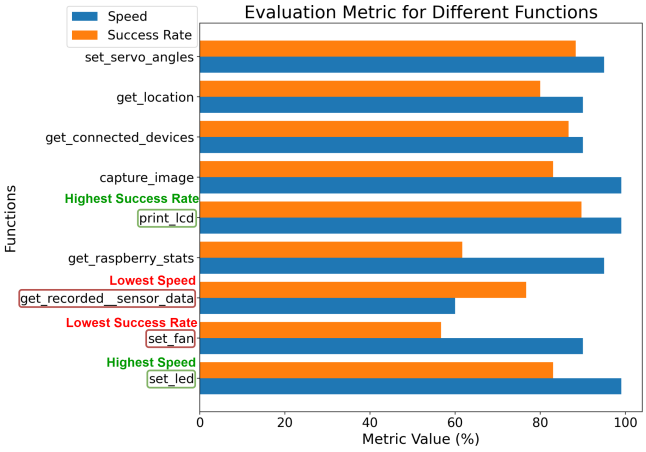


FIGURE 9. **Evaluation metrics for the functions defined earlier in .**

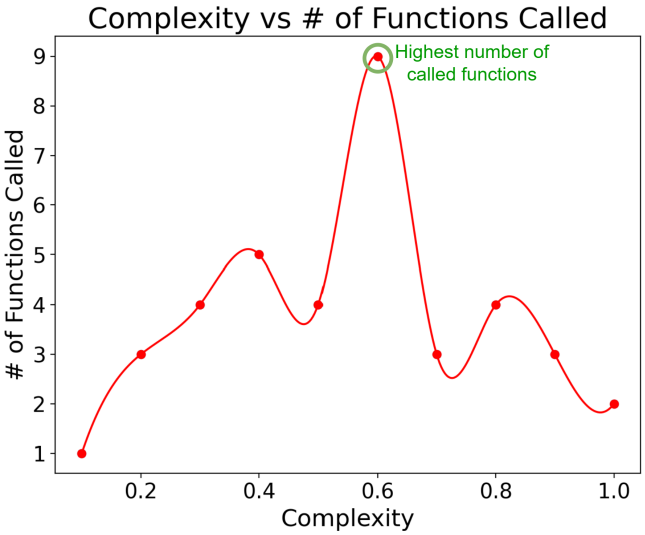


FIGURE 10. **Message complexity against the number of functions called per message.**

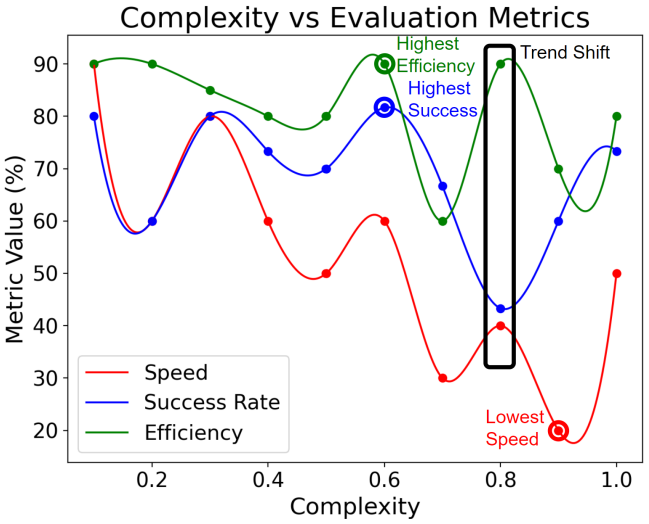


FIGURE 11. **Message complexity against all evaluation metrics and most importantly the success rate.**

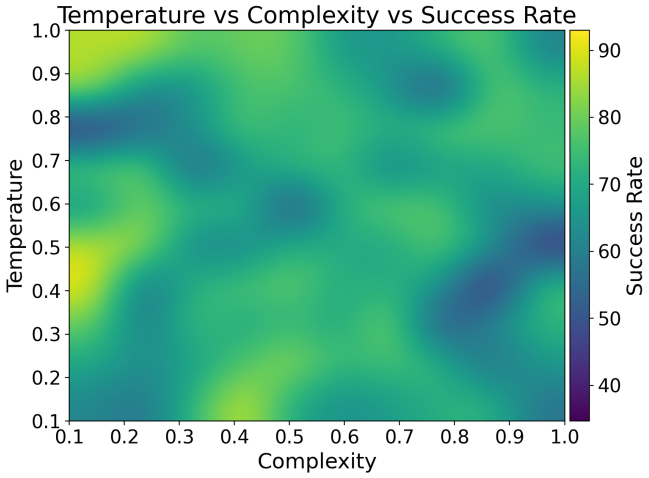


FIGURE 12. **Success rate against message complexity and temperature of the LLM.**

and recommendations. *Journal of Network and Computer Applications*, 163:102663, 2020.

[5] OpenAI. *Generative Pre-trained Transformer (GPT) Models*, 2023.

[6] Sarah A. Al-Qaseemi, Hajer A. Almulhim, Maria F. Almulhim, and Saqib Rasool Chaudhry. Iot architecture challenges and issues: Lack of standardization. In *2016 Future Technologies Conference (FTC)*, pages 731–738, 2016.

[7] Richard N. Taylor, Nenad Medvidovic, and Eric M. Dashofy. *Software Architecture: Foundations, Theory, and Practice*. John Wiley & Sons, 2010.

[8] Elizabeth Kadiyala, Shravya Meda, Revathi Basani, and S. Muthulakshmi. Global industrial process monitoring through iot using raspberry pi. In *2017 International Conference on Nextgen Electronic Technologies: Silicon to Software (ICNETS2)*, pages 260–262, 2017.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[10] Paula Maddigan and Teo Susnjak. Chat2vis: Generating data visualizations via natural language using chatgpt, codex and gpt-3 large language models. *IEEE Access*, 11:45181–45193, 2023.

[11] Maanak Gupta, Charankumar Akiri, Kshitiz Aryal, Eli Parker, and Lopamudra Praharaj. From chatgpt to threatgpt: Impact of generative ai in

FIGURE 13. Success rate against message complexity and Top P of the LLM.

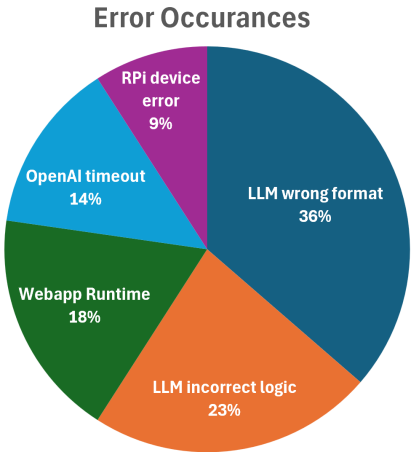


FIGURE 14. What types of errors occured throughout testing.

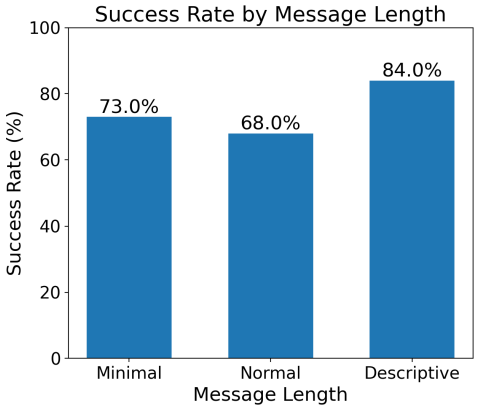


FIGURE 15. Success rate of different tones of the same message.

cybersecurity and privacy. *IEEE Access*, 11:80218–80245, 2023.

[12] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: program generation for situated robot task planning using large language models. *Autonomous Robots*, 47(8):999–1012, Dec 2023.

[13] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. Palm-e: an embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

[14] Shyam Sundar Kannan, Vishnunandan L. N. Venkatesh, and Byung-Cheol Min. Smart-llm: Smart multi-agent robot task planning using large language models, 2024.

[15] Jimmy Wu, Rika Antonova, Adam Kan, Marion Lepert, Andy Zeng, Shuran Song, Jeannette Bohg, Szymon Rusinkiewicz, and Thomas Funkhouser. Tidybot: personalized robot assistance with large language models. *Autonomous Robots*, 47(8):1087–1102, Dec 2023.

[16] Evan King, Haoxiang Yu, Sangsu Lee, and Christine Julien. Sasha: Creative goal-oriented reasoning in smart homes with large language models. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(1), mar 2024.

[17] N. Petrović, S. Koničanin, and S. Suljović. Chatgpt in iot systems: Arduino case studies. In *2023 IEEE 33rd International Conference on Microelectronics (MIEL)*, pages 1–4, 2023.

[18] Ningze Zhong, Yi Wang, Rui Xiong, Yingyue Zheng, Yang Li, Mingjun Ouyang, Dan Shen, and Xiangwei Zhu. Casit: Collective intelligent agent system for internet of things. *IEEE Internet of Things Journal*, 11(11):19646–19656, 2024.

[19] Paria Sarzaeim, Qusay H. Mahmoud, and Akramul Azim. A framework for llm-assisted smart policing system. *IEEE Access*, 12:74915–74929, 2024.

[20] Zhipeng Xu, Hao Wu, Xu Chen, Yongmei Wang, and Zhenyu Yue. Building a natural language query and control interface for iot platforms. *IEEE Access*, 10:68655–68668, 2022.

[21] Gwendal Daniel, Jordi Cabot, Laurent Deruelle, and Mustapha Derras. Xatkit: A multimodal low-code chatbot development framework. *IEEE Access*, 8:15332–15346, 2020.

[22] Javier Palanca, Andrés Terrasa, Vicente Julian, and Carlos Carrascosa. Spade 3: Supporting the new generation of multi-agent systems. *IEEE Access*, 8:182537–182549, 2020.

[23] Tzu-Yu Chen, Yu-Ching Chiu, Nanyi Bi, and Richard Tzong-Han Tsai. Multi-modal chatbot in intelligent manufacturing. *IEEE Access*, 9:82118–82129, 2021.

[24] Giovanni Almeida Santos, Guilherme Guy de Andrade, Geovana Ramos Sousa Silva, Francisco Carlos Molina Duarte, João Paulo Javidi Da Costa, and Rafael Timóteo de Sousa. A conversation-driven approach for chatbot management. *IEEE Access*, 10:8474–8486, 2022.

[25] Chen Li, Xiaochun Zhang, Dimitrios Chrysostomou, and Hongji Yang. Tod4ir: A humanised task-oriented dialogue system for industrial robots. *IEEE Access*, 10:91631–91649, 2022.

[26] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. Recipes for building an open-domain chatbot. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 300–325, Online, April 2021. Association for Computational Linguistics.

[27] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, Upper Saddle River, NJ, 1st edition, 2002.

[28] Jing Han, Haihong E, Guan Le, and Jian Du. Survey on nosql database. In *2011 6th International Conference on Pervasive Computing and Applications*, pages 363–366, 2011.

[29] Satish Bhosale, Miss Patil, and Pooja Patil. International journal of computer science and mobile computing sqlite: Light database system. *International Journal of Computer Science and Mobile Computing*, 44:882–885, 04 2015.

[30] Chunyue Bi. Research and application of sqlite embedded database technology. *WSEAS Transactions on Computers*, 8:83–92, 01 2009.

[31] Evgeny Nikulchev, Dmitry Ilin, and Alexander Gusev. Technology stack selection model for software design of digital platforms. *Mathematics*, 9(4), 2021.

[32] React Team. *React - A JavaScript library for building user interfaces*. Meta Platforms, Inc., 2024.

[33] WorkOS. *Radix UI*, 2022. Open-source UI component library for building high-quality, accessible design systems and web apps.

[34] Tailwind Labs. *Tailwind CSS*, 2023.

[35] Armin Ronacher. *Flask: Web Development, One Drop at a Time*, 2024.
[36] Electronic Wings. Sensors and modules, 2023.
[37] John Smith and Michael Davis. Testing and debugging iot projects with raspberry pi. *Journal of Internet of Things*, 8(2):123–135, 2020.
[38] Simon Monk. *Programming the Raspberry Pi, Second Edition: Getting Started with Python*. McGraw-Hill Education, 2019.
[39] Michael Brown and Susan Green. Integrating camera modules with raspberry pi for image capture applications. *IEEE Transactions on Consumer Electronics*, 64(2):145–152, 2018.
[40] Simon McManus. *Raspberry Pi Cookbook: Software and Hardware Problems and Solutions*. O'Reilly Media, Inc., 2021.
[41] Barry Wilkinson and Michael Allen. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*. Pearson/Prentice Hall, 2nd edition, 2005.
[42] Vijaya B. Surwase. Rest api modeling languages - a developer's perspective. *International Journal For Science Technology And Engineering*, 2:634–637, 2016.
[43] Jacek Wytrębowicz, Krzysztof Cabaj, and Jerzy Krawiec. Messaging protocols for iot systems—a pragmatic comparison. *Sensors*, 21(20), 2021.
[44] Taeho Kim, Yanming Wang, Vatshank Chaturvedi, Lokesh Gupta, Seyeon Kim, Yongin Kwon, and Sangtae Ha. Llmem: Estimating gpu memory usage for fine-tuning pre-trained llms, 2024.
[45] Hui Yang, Sifu Yue, and Yunzhong He. Auto-gpt for online decision making: Benchmarks and additional opinions, 2023.