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

Digital Object Identifier 10.1109/ACCESS.2024.0429000

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

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

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."

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.

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.

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 IoT systems often requires specialized knowledge and programming skills, creating significant barriers for non-technical users [3].

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

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.

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 in the literature [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 a 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 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 rest 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

The foundation for LLMs in natural language processing was laid by [9] with the introduction of the Transformer architecture. These models, trained on vast amounts of text data, have shown potential in applications ranging from code generation to robotic control.

Recent research has explored diverse applications of LLMs. [10] demonstrated the use of LLMs like ChatGPT and GPT-3 for generating data visualizations from natural language queries. In cybersecurity, [11] investigated both defensive and offensive applications of GenAI tools, highlighting potential risks and opportunities. [12] compared the performance of phishing emails created by GPT-4 against those manually designed using traditional methods, revealing the potential of AI-enabled phishing attacks.

The integration of LLMs with robotics has emerged as a promising field. [13] showed how ChatGPT could convert natural language instructions into executable robot actions using few-shot prompts. [14] introduced PromptCraft, a platform for collaborative development of prompting schemes for robotics applications. [15] presented ProgPrompt, a method using programmatic LLM prompts to enable plan generation across different environments and robot capabilities.

[16] demonstrated a system for summarizing and answering questions about a virtual robot's past actions using egocentric video frames. [17] introduced COWP, a framework for open-world task planning that augments robot action knowledge with task-oriented commonsense knowledge from LLMs. [18] proposed SMART-LLM, a framework for multi-robot task planning using LLMs to convert high-level instructions into multi-robot task plans.

[19] developed TidyBot, a system combining language-based planning and perception with LLMs to infer generalized user preferences for household cleanup tasks. [20] introduced GenSim, an approach using LLMs to generate rich simulation environments and expert demonstrations for robotic tasks. This work demonstrates the potential of LLMs in enhancing task-level generalization in robotic policy training.

In the realm of education, [21] explored the use of LLMs for robot code explanation, generation, and modification in teaching coding to children. Their work highlighted the superior performance of GPT-4V in both traditional coding tasks and robot code generation. In the context of embodied AI, [22] proposed PaLM-E, an embodied multimodal language model incorporating real-world sensor data, showcasing its potential in robotic manipulation planning and visual question answering.

These advancements in LLM applications for robotics provide a 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. 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 [23]. 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, a large language model (LLM), in IoT systems, focusing on Arduino-based applications [24]. 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 [25]. 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 [23].

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 [26]. 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 [27]. 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

The integration of large language models (LLMs) with Internet of Things (IoT) systems has emerged as a promising frontier, enabling natural language interaction and intelligent control of physical devices. While this specific application domain remains relatively unexplored, several recent works have pioneered the fusion of LLMs and robotics, laying the groundwork for their extension to IoT scenarios [28].

Chatbots have gained significant traction across various industries, serving as direct communication channels between companies and end-users [29]. 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 [29].

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 [29]. 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 [30].

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 [31]. 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 [32].

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 [33]. 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 [34]. 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, and enhances overall maintenance. The architecture also implements separation of concerns, where different aspects such

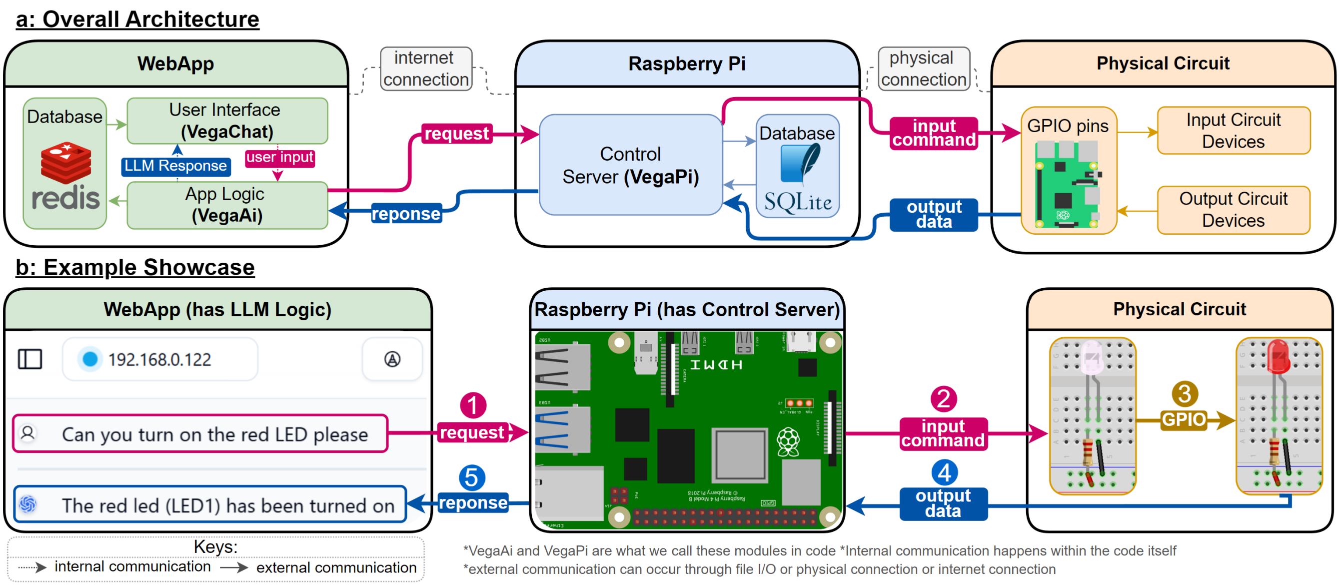

FIGURE 1. Overall architecture of Vega alongside a simple example.

as user interface, core functionality, and data management are segregated into distinct layers, improving code organization and enabling independent development.

Vega's architecture comprises three main modules: a Web Application, a RPi, and a Physical Circuit. These modules interact in a client-server model [35], 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.

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 [36] 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 [37], [38] 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.

A typical use case 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.

The technology stack for Vega has been carefully selected to ensure robustness, scalability, and accessibility [39]. The Web Application is built using React [40] with TypeScript, employing RadixUI [41] for accessible components and TailwindCSS [42] 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 [43], 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 a Raspberry Pi (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.

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 [44]. These components collectively provide a rich set of data sources, enabling the system to respond to complex, context-aware queries and commands.

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

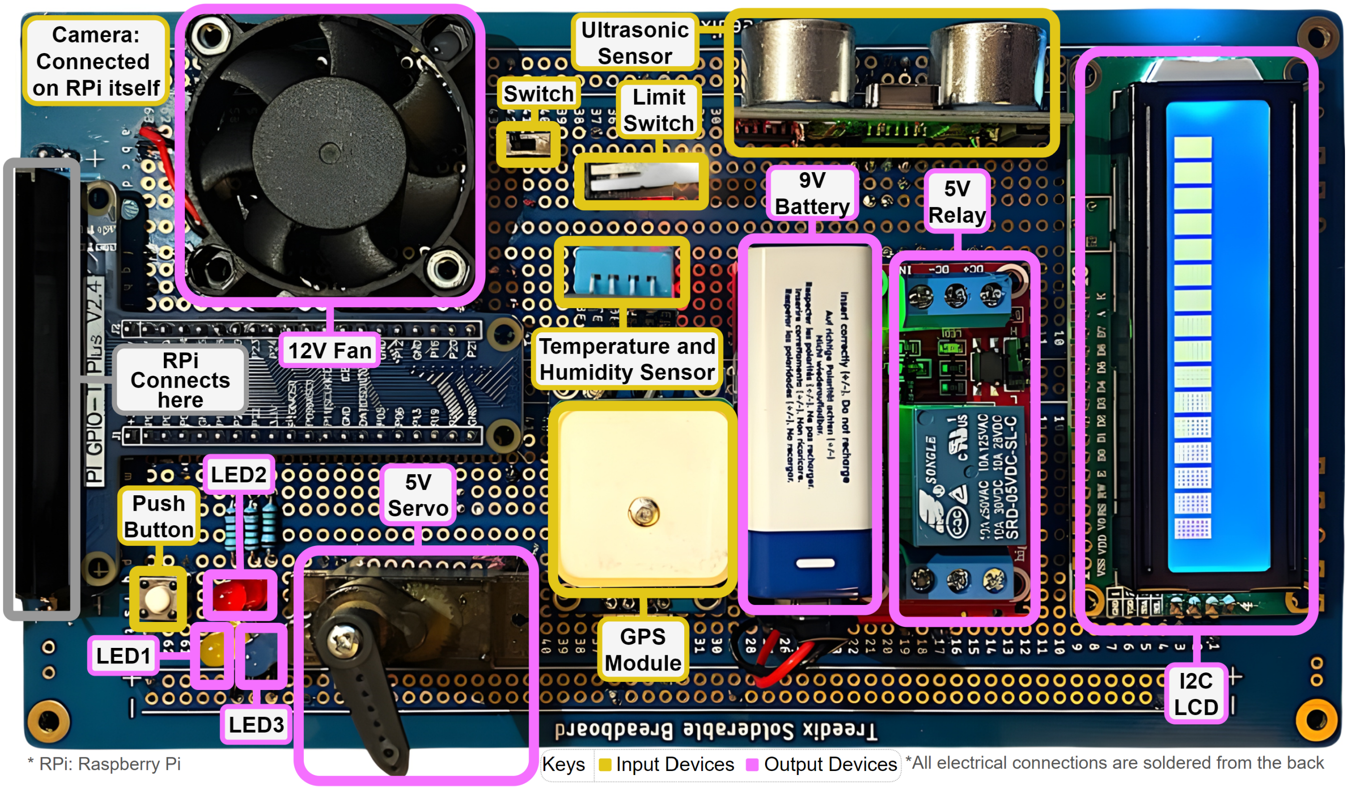

FIGURE 2. Soldered physical circuit connected to the RPi

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 [45]. 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.

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 [46]. This setup illustrates the system's ability to accommodate components with varying power requirements within a unified control structure.

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 [47].

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 [48].

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 Raspberry Pi (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 [49].

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 Representational State Transfer (REST) API, facilitating communication between different software systems over the internet [50].

The Database Thread retrieves sensor data at two-second

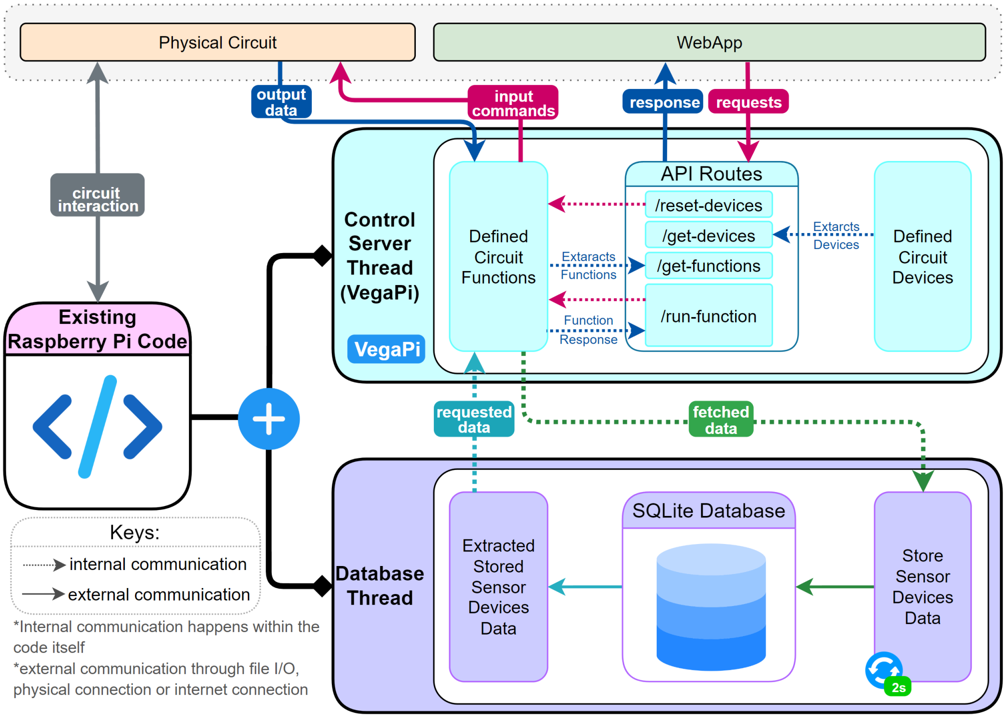

FIGURE 3. Architecture design of the RPi control server

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

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

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 current 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"

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 Large Language Models (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 JavaScript Object Notation (JSON) format via the "get-devices" REST API endpoint [35]. 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 [51]. It also provides a uniform interface, enabling easier integration with various client applications. While MQTT excels in low-bandwidth, high-latency environments, the current system architecture prioritizes the flexibility and widespread support offered by REST APIs in web

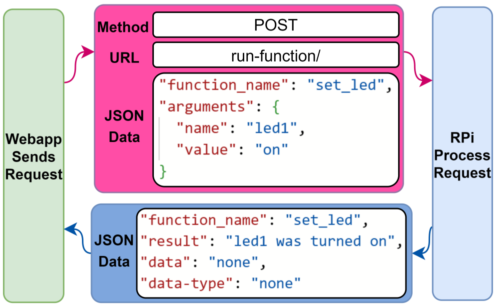

FIGURE 4. Webapp user interface implementation

development ecosystems.

The communication flow between the web application and the RPi follows a request-response pattern as shown in Figure 4. 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

E. WEB APP LLM LOGIC

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.

[4] Wafa'a Kassab and Khalid A. Darabkh. A–z survey of internet of things: Architectures, protocols, applications, recent advances, future directions 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 cybersecurity and privacy. *IEEE Access*, 11:80218–80245, 2023.

[12] Fredrik Heiding, Bruce Schneier, Arun Vishwanath, Jeremy Bernstein, and Peter S. Park. Devising and detecting phishing emails using large language models. *IEEE Access*, 12:42131–42146, 2024.

[13] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Chatgpt empowered long-step robot control in various environments: A case application. *IEEE Access*, 11:95060–95078, 2023.

[14] Sai H. Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities. *IEEE Access*, 12:55682–55696, 2024.

[15] 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.

[16] Chad DeChant, Iretiayo Akinola, and Daniel Bauer. Learning to summarize and answer questions about a virtual robot's past actions. *Autonomous Robots*, 47(8):1103–1118, Dec 2023.

[17] Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Andy Kaminski, Chad Esselink, and Shiqi Zhang. Integrating action knowledge and llms for task planning and situation handling in open worlds. *Autonomous Robots*, 47(8):981–997, Dec 2023.

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

[19] 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.

[20] Lirui Wang, Yiyang Ling, Zhecheng Yuan, Mohit Shridhar, Chen Bao, Yuzhe Qin, Bailin Wang, Huazhe Xu, and Xiaolong Wang. Gensim: Generating robotic simulation tasks via large language models, 2024.

[21] Peng Shu, Huaqin Zhao, Hanqi Jiang, Yiwei Li, Shaochen Xu, Yi Pan, Zihao Wu, Zhengliang Liu, Guoyu Lu, Le Guan, Gong Chen, and Xianqiao Wang Tianming Liu. Llms for coding and robotics education, 2024.

[22] 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.

[23] 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.

[24] 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.

[25] Ningze Zhong, Yi Wang, Rui Xiong, Yingyue Zheng, Yang Li, Mingjun Ouyang, Dan Shen, and Xiangwei Zhu. Casit: Collective intelligent

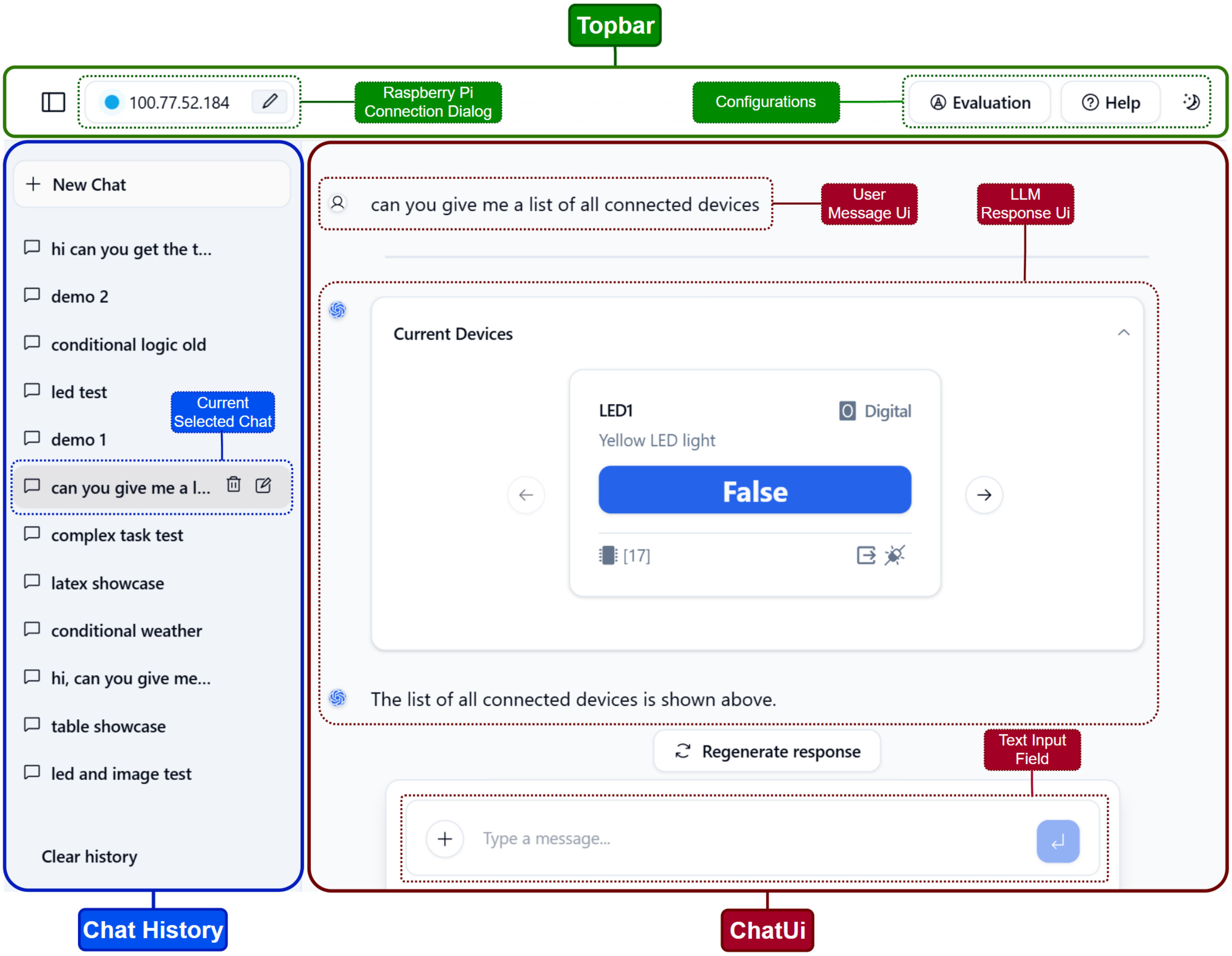

FIGURE 5. **Webapp user interface implementation**

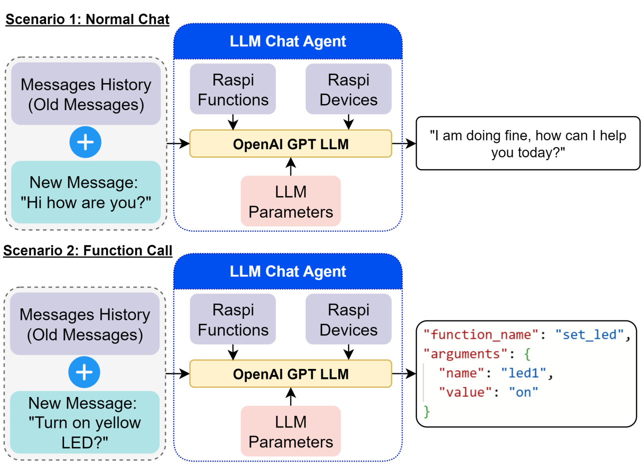

FIGURE 6. **Evaluation metrics for the functions defined earlier in .**

agent system for internet of things. *IEEE Internet of Things Journal*, 11(11):19646–19656, 2024.

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

[27] 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.

[28] Callie Y. Kim, Christine P. Lee, and Bilge Mutlu. Understanding large-language model (llm)-powered human-robot interaction. In *Proceedings of the 2024 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '24, page 371–380, New York, NY, USA, 2024. Association for Computing Machinery.

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

[30] 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.

[31] 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.

[32] 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.

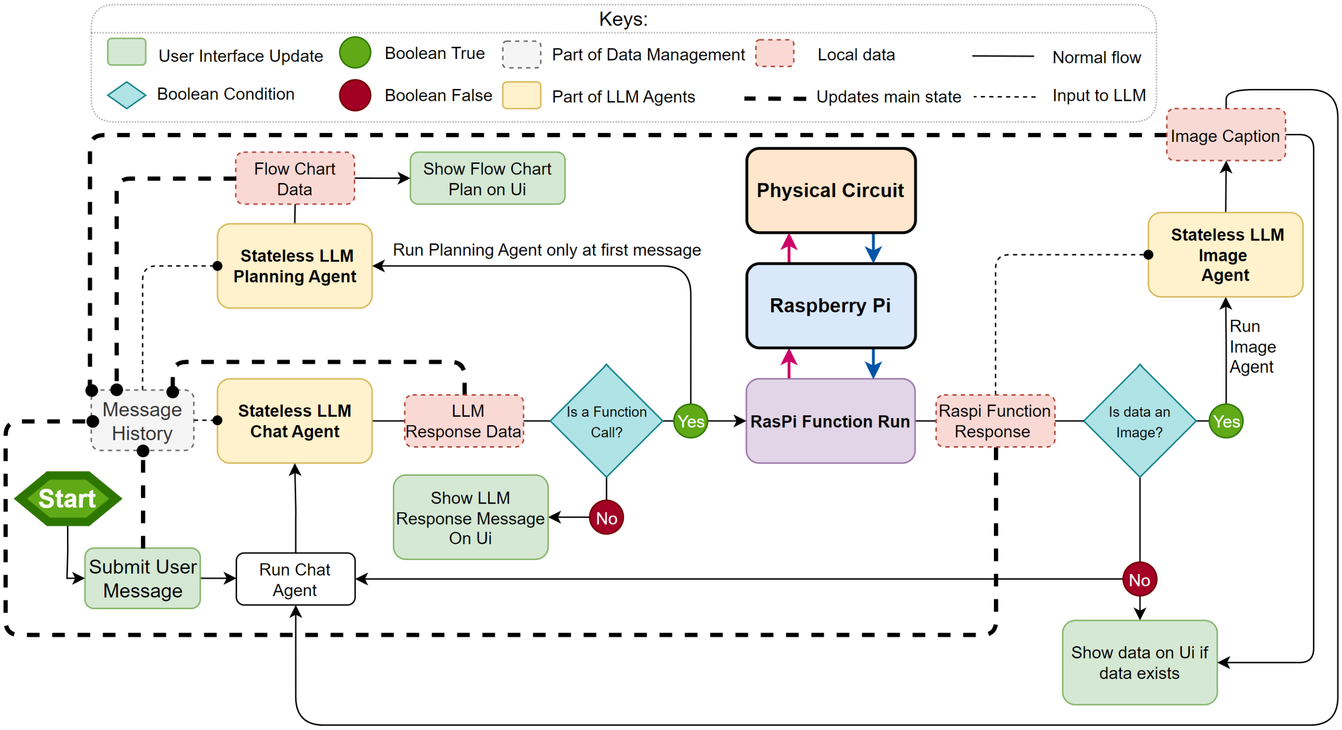

FIGURE 7. **Webapp logic design**

[33] 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.
[34] 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.
[35] Andrew S. Tanenbaum and Maarten Van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall, Upper Saddle River, NJ, 1st edition, 2002.
[36] 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.
[37] 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.
[38] Chunyue Bi. Research and application of sqlite embedded database technology. *WSEAS Transactions on Computers*, 8:83–92, 01 2009.
[39] Evgeny Nikulchev, Dmitry Ilin, and Alexander Gusev. Technology stack selection model for software design of digital platforms. *Mathematics*, 9(4), 2021.
[40] React Team. *React - A JavaScript library for building user interfaces*. Meta Platforms, Inc., 2024.
[41] WorkOS. *Radix UI*, 2022. Open-source UI component library for building high-quality, accessible design systems and web apps.
[42] Tailwind Labs. *Tailwind CSS*, 2023.
[43] Armin Ronacher. *Flask: Web Development, One Drop at a Time*, 2024.
[44] Electronic Wings. Sensors and modules, 2023.
[45] John Smith and Michael Davis. Testing and debugging iot projects with raspberry pi. *Journal of Internet of Things*, 8(2):123–135, 2020.
[46] Simon Monk. *Programming the Raspberry Pi, Second Edition: Getting Started with Python*. McGraw-Hill Education, 2019.
[47] 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.
[48] Simon McManus. *Raspberry Pi Cookbook: Software and Hardware Problems and Solutions*. O'Reilly Media, Inc., 2021.
[49] Barry Wilkinson and Michael Allen. *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*. Pearson/Prentice Hall, 2nd edition, 2005.
[50] Vijaya B. Surwase. Rest api modeling languages - a developer's perspective. *International Journal For Science Technology And Engineering*, 2:634–637, 2016.
[51] Jacek Wytrębowicz, Krzysztof Cabaj, and Jerzy Krawiec. Messaging protocols for iot systems—a pragmatic comparison. *Sensors*, 21(20), 2021.

HARITH AL-SAFI received the BEng industrial degree in Electronics and Computer Engineering from the University of Leeds, Leeds, in 2024. From 2022 to 2023, he was an IoT Software Engineer at Johnson Controls. He is currently pursuing a Graduate Software Developer role at BT Group. His research interests include IoT, embedded systems, and machine learning.

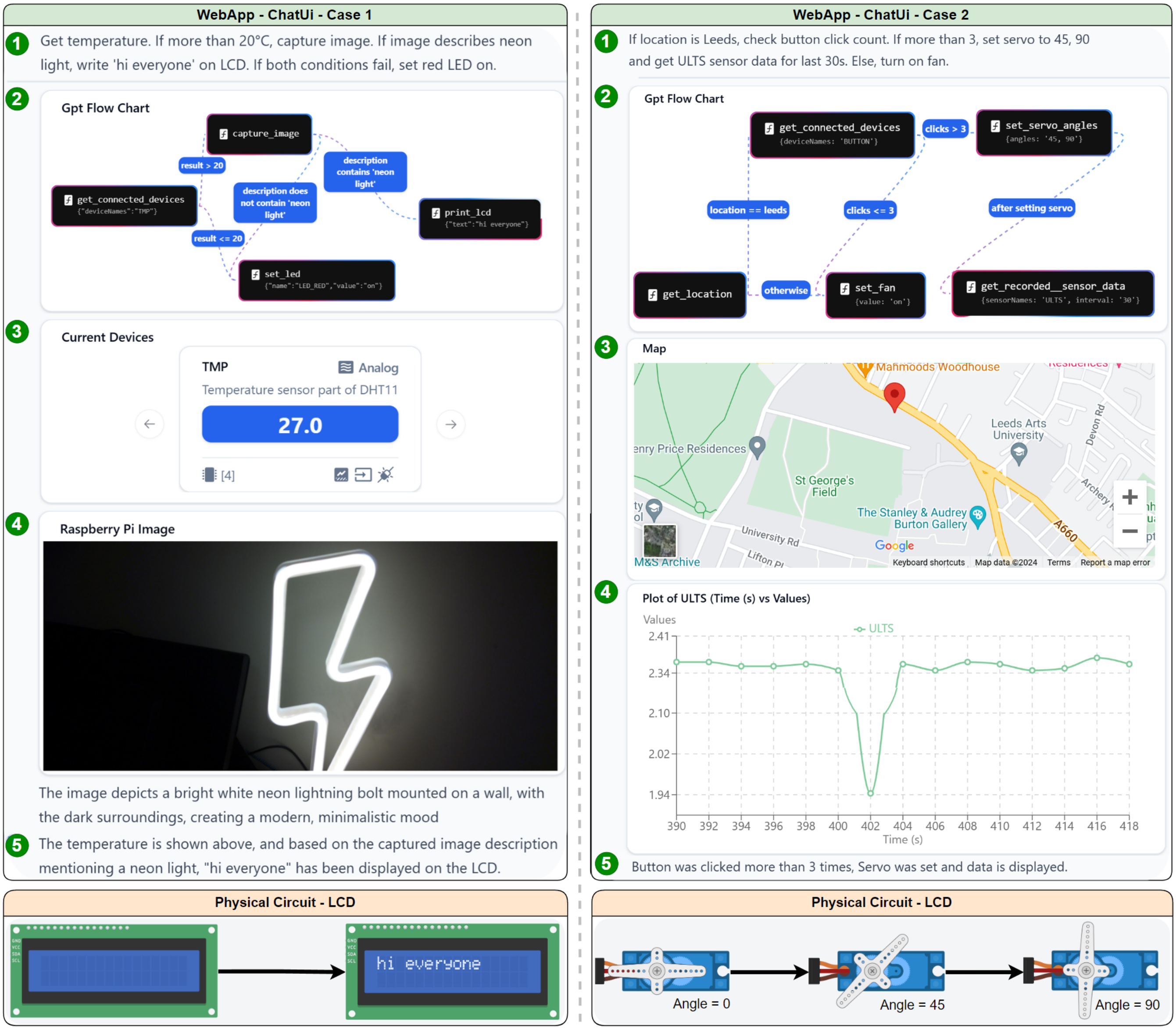

FIGURE 8. System case studies

SECOND B. AUTHOR (M'76–SM'81–F'87) and all authors may include biographies. Biographies are often not included in conference-related papers. This author became a Member (M) of IEEE in 1976, a Senior Member (SM) in 1981, and a Fellow (F) in 1987. The first paragraph may contain a place and/or date of birth (list place, then date). Next, the author's educational background is listed. The degrees should be listed with type of degree in what field, which institution, city, state, and country, and year the degree was earned. The author's major field of study should be lower-cased.

The second paragraph uses the pronoun of the person (he or she) and not the author's last name. It lists military and work experience, including summer and fellowship jobs. Job titles are capitalized. The current job must have a location; previous positions may be listed without one. Information concerning previous publications may be included. Try not to list more than three books or published articles. The format for listing publishers of a book within the biography is: title of book (publisher name, year) similar to a reference. Current and previous research interests end the paragraph.

The third paragraph begins with the author's title and last name (e.g., Dr. Smith, Prof. Jones, Mr. Kajor, Ms. Hunter). List any memberships in professional societies other than the IEEE. Finally, list any awards and work for IEEE committees and publications. If a photograph is provided, it should be of good quality, and professional-looking. Following are two examples of an author's biography.

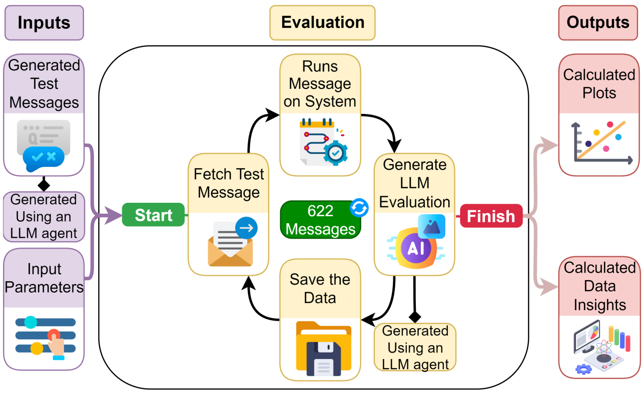

FIGURE 9. **Magnetization as a function of applied field. It is good practice to explain the significance of the figure in the caption.**

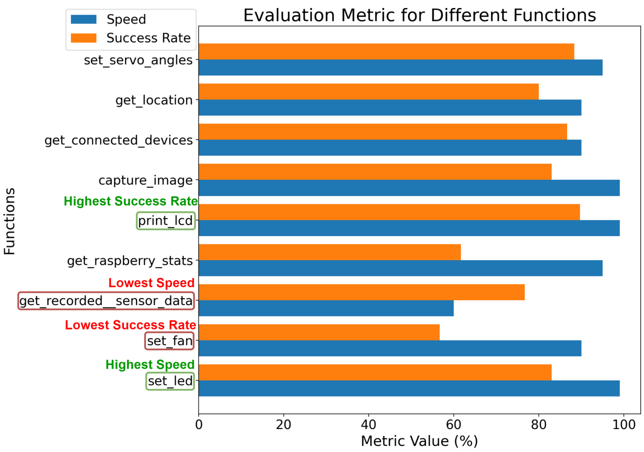

FIGURE 10. **Evaluation metrics for the functions defined earlier in .**

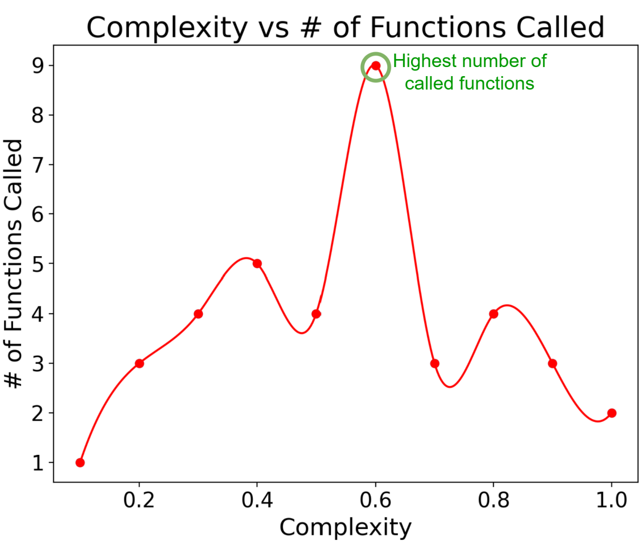

FIGURE 11. **Message complexity against the number of functions called per message.**

THIRD C. AUTHOR, JR. (M'87) received the B.S. degree in mechanical engineering from National Chung Cheng University, Chiayi, Taiwan, in 2004

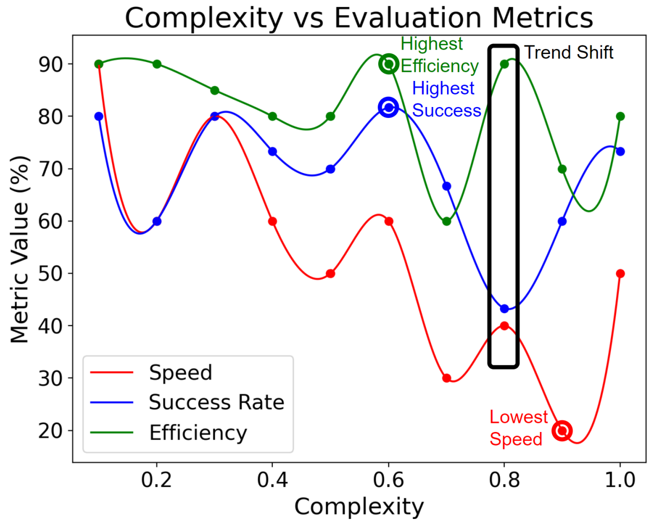

FIGURE 12. **Message complexity against all evaluation metrics and most importantly the success rate.**

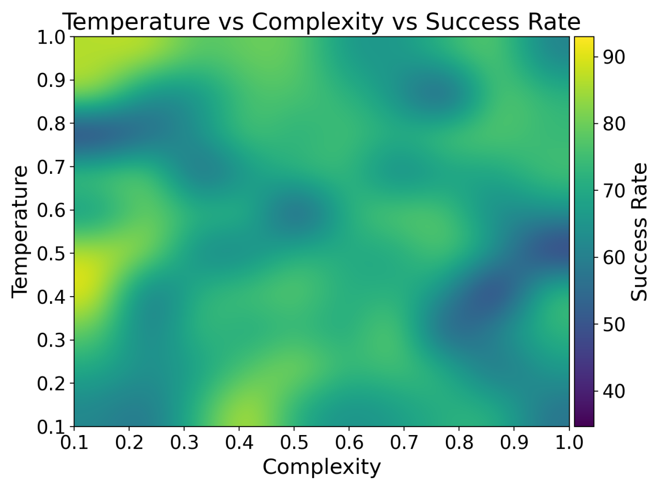

FIGURE 13. **Success rate against message complexity and temperature of the LLM.**

and the M.S. degree in mechanical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2006. He is currently pursuing the Ph.D. degree in mechanical engineering at Texas A&M University, College Station, TX, USA.

From 2008 to 2009, he was a Research Assistant with the Institute of Physics, Academia Sinica, Tapei, Taiwan. His research interest includes the development of surface processing and biological/medical treatment techniques using nonthermal atmospheric pressure plasmas, fundamental study of plasma sources, and fabrication of micro- or nanostructured surfaces.

Mr. Author's awards and honors include the Frew Fellowship (Australian Academy of Science), the I. I. Rabi Prize (APS), the European Frequency and Time Forum Award, the Carl Zeiss Research Award, the William F. Meggers Award and the Adolph Lomb Medal (OSA).

• • •

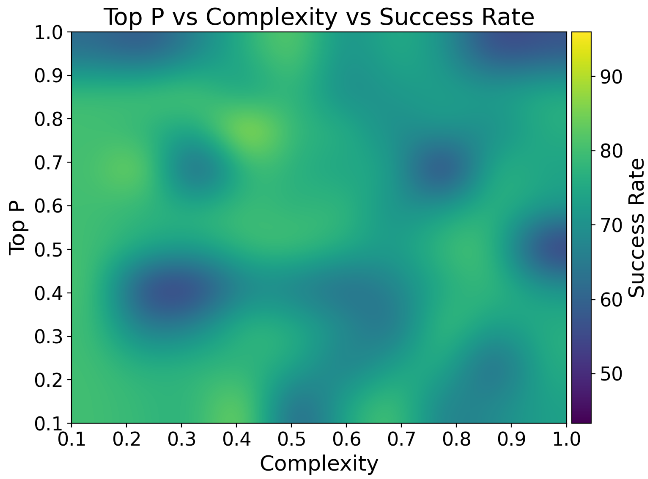

FIGURE 14. Success rate against message complexity and Top P of the LLM.

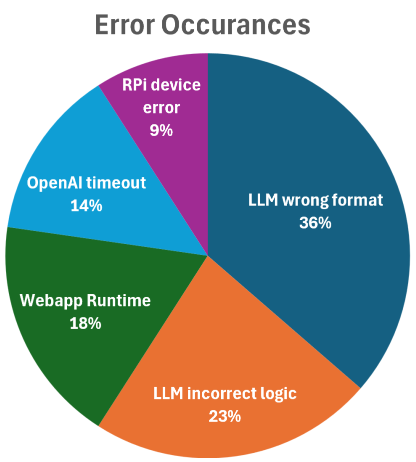

FIGURE 15. What types of errors occured throughout testing.

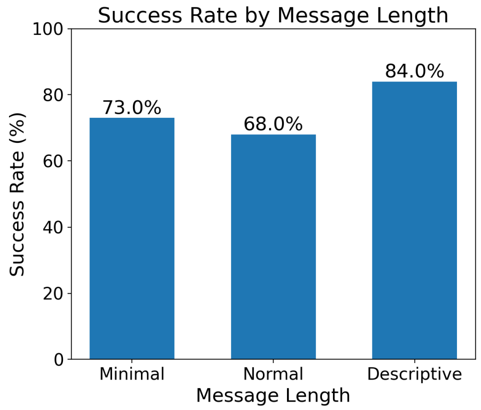

FIGURE 16. Success rate of different tones of the same message.