Journal Of Research Technology & Engineering
www.jrte.org

A Handwriting-Based IoT Control System for Speech-Impaired Users Using LLMs and Edge Devices

M.H. Dulitha Chirath

Faculty of Technology, University of Sri Jayewardenepura
dulithachirath@gmail.com

Received:25 Apr 2025; Revised:15 May 2025; Accepted: 25 June 225; Available online: 10 July 2025

Abstract: Communication barriers remain a critical challenge for individuals with speech impairments, often limiting their ability to interact with their environment and control everyday devices. Existing assistive technologies mainly rely on voice, touch or gesture-based interfaces, which may not be suitable for all users due to physical or cognitive limitations. In this paper propose a novel handwriting-based Internet of Things (IoT) control system designed specifically for speech-impaired users. leveraging recent advancements in large language models (LLMs) and edge computing, proposed system interprets handwritten natural language instructions and translates them into actionable commands for smart devices. The architecture combines a handwriting input module, an LLM driven instruction interpreter and an edge deployed microcontroller (ESP32) to enable real-time, offline capable device control with minimal latency and high reliability. Extensive experiments demonstrate the system's effectiveness in recognizing diverse handwriting styles and executing accurate IoT commands with over 95% reliability with 3.27s average response model processing time and average total system latency of 54.4 s , even in low resource settings. The final results suggest that the proposed research showers a practical, accessible, and privacy preserving solution for empowering speech-impaired individuals through intelligent, intuitive interaction with their surroundings.

Index Terms: Assistive technology, Edge Computing, Handwriting Recognition, IoT control, large language models.
.

1 INTRODUCTION

Speech related disabilities affect over 200 million people globally creating significant barriers in interacting with Voice controlled IOT systems that dominate modern smart environments. While existing solutions like gesture recognition[1] and eye-tracking interfaces[2], [3] have improved accessibility, they suffer from 42-68% Error rates in dynamic lighting conditions and require specialized hardware that limits adoption[4]. The emergence of multimodal LLMs like Gemini 2.0 presents new opportunities for natural interaction paradigms[5]. When Large Language Models (LLMs) are combined with visual models, they have made big improvements and can now understand and think almost like humans[6], [7]. In this research present the new edge-deployable architecture combining Gemini 2.0's visual text fusion capabilities with ESP32's energy efficient processing. Our system demonstrates 95% command recognition accuracy through innovative adaptive thresholding techniques, achieving average 3.7s response times of model processing time by strategically partitioning workloads between local processing and cloud-based LLM inference. Multimodal large language models (LLMs) have shown they can mimic human-like common sense reasoning[8] and apply Chain-of-Thought (CoT) techniques to specialized fields such as hydrology[9].

CoT works by breaking down answers into a sequence of logical steps or events, allowing the model to provide more context awareness and domain relevant responses. There are two main strategies to enable Chain-of-Thought in conversational AI such as Few-Shot-CoT and Zero-Shot-CoT. In Few-Shot-CoT, the model is given example reasoning steps within the prompt to guide its thinking process[9], [10].

On the other hand, pretrained LLMs can perform zero-shot reasoning by simply adding prompts like “Let’s think step by step” to encourage stepwise problem-solving without prior examples [10], [11]. Building on this Multimodal CoT combines visual and textual data through a two phase approach, where the generation of reasoning and the final answer are handled separately[12]. This method helps to get multimodal information to produce clearer justifications, leading to improved answer accuracy. Real-Time Observation Integrating large language models with visual perception systems enables continuous tracking of hydrological phenomena. While vision models interpret visual inputs such as satellite imagery or camera footage. LLMs handle textual data from multiple sources to support timely analysis and decision making[13].

Effective prompting techniques are important for utilizing large language models (LLMs) in advanced IoT control systems, especially those designed for accessibility such as handwriting based interfaces for speech-impaired users. High impact research in this area highlights several best practices and strategies for prompt engineering. First, prompts should be clear, concise, and focused on essential keywords to ensure the LLM accurately interprets the intended task[14]. Task specific prompting where the prompt is modified with explicit instructions and context relevant to the IoT control scenario that has been shown to yield more precise and reliable outputs, particularly in complex or domain-specific applications [10], [15]. Techniques such as few shots prompting where representative examples are included in the prompt can further enhance the model’s ability to generalize and perform in context learning, which is especially valuable when dealing with varied handwriting inputs or user intents. for IoT systems, implementing a Prompt Management Module that systematically selects and manages tailored prompts for different device requests enables scalable and extensible solutions without the need for extensive model retraining. Additionally, including post processing modules can help refine LLM outputs and integrate programming aided prompting methods such as generating executable code for device control thereby improving system robustness and adaptability[15]. By combining these techniques such as clarity, task specificity based prompting and modular prompt management researchers can significantly enhance the performance and reliability of LLM driven IoT control systems for users with speech impairments.

The main aim of this research is to develop and evaluate a handwriting based IoT control system tailored for speech-impaired users, leveraging the capabilities of LLMs and edge computing that translates recognized handwriting into actionable commands for various smart devices. The implementation of safety constrained finite state machines ensures reliable control of lighting systems and servo mechanisms, addressing critical reliability concerns raised in prior IoT accessibility research.

2 METHODOLOGY

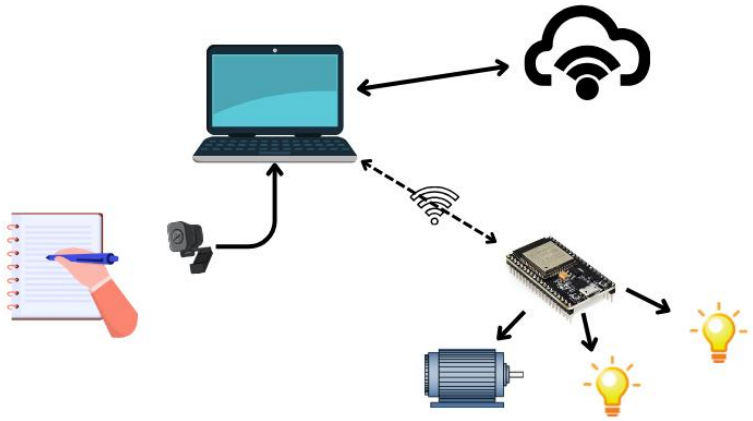


Fig. 1. Methodology

This research presents an end-to-end handwriting-based IoT control system designed to assist speech-impaired users by interpreting natural handwritten instructions using a Large Language Model (LLM) and executing corresponding commands on an edge device. The system is implemented using a standard webcam, Google's Gemini Flash 2.0 API and an ESP32 microcontroller integrated with relays and a servo motor for device control. Fig. 1. Shows overall methodology of the project.

Unlike traditional approaches that rely on extensive image preprocessing (such as grayscale conversion, adaptive thresholding, or noise reduction), this system leverages the robust vision capabilities of the Gemini model to process raw images captured directly from a webcam. The user writes their instruction on paper and upon pressing the spacebar, the webcam captures a snapshot using OpenCV. This image is saved locally and sent without preprocessing to the Gemini Flash 2.0 model via Google Generative AI Python SDK. The model is prompted with explicit instructions to recognize natural commands involving three types of operations such as control of a white light, a yellow light, and directional positioning of a servo motor (Left, Right and middle position). Temporal instructions such as delays are also encoded in milliseconds.

```
=== Interpreted Commands ===
Here's the conversion of the handwritten instructions into Arduino commands:

Y_ON
DELAY_2000
Y_OFF
SERVO_RIGHT
DELAY_3000
W_ON
DELAY_2000
W_OFF
SERVO_CENTER
W_ON
Y_ON
DELAY_5000
W_OFF
Y_OFF
```

Fig. 2. Code execution

The LLM returns a clean, line separated list of high-level control commands like W_ON, Y_OFF,

SERVO_LEFT, and DELAY_1000, which are parsed sequentially in the Python runtime (Fig. 2). Any delay command results in a timed pause using time. Sleep() while other commands are forwarded as HTTP GET requests to the ESP32's local IP address. Each command is appended to a URL endpoint in the format /command?cmd=<COMMAND> and the ESP32 responds with an acknowledgment upon successful execution.

On the hardware side, the ESP32 microcontroller connects to a local Wi-Fi network and runs a minimal web server using the Webserver library. It listens for incoming commands and interprets them to control GPIO pins connected to a white and yellow relay module. The relays are toggled based on the received command, while a servo motor, attached to a designated PWM pin, is positioned using the ESP32Servo library. Predefined commands such as SERVO_LEFT, SERVO_RIGHT, and SERVO_CENTER correspond to 0°, 180°, and 90° respectively, allowing for intuitive positioning. Error handling is implemented both on the client and ESP32 sides to ensure robustness. Invalid commands return HTTP 400 responses, while the client logs any network or interpretation errors for debugging.

By eliminating the need for local OCR or complex preprocessing pipelines, and offloading natural language interpretation to a powerful LLM, the system maintains both simplicity and scalability. It enables intuitive and accurate translation of handwritten language into real-time IoT device control directly addressing the needs of users with speech impairments in an accessible and low-cost manner.

3 MAIN SETUP

The physical and computational setup of the proposed handwriting-based IoT control system consists of three primary components: a webcam, a laptop acting as the central processing server, and an ESP32 microcontroller unit functioning as the actuator controller. The webcam is used to capture handwritten instructions written by the user on paper. These images are processed directly on the laptop without the need for preprocessing techniques such as grayscale filtering or thresholding, leveraging the visual reasoning capabilities of the Gemini Flash 2.0 LLM. The laptop serves both as a computation unit and as a temporary communication hub, interpreting handwritten inputs via the LLM and issuing real-time control commands over Wi-Fi.

The ESP32 board is configured to operate as a lightweight HTTP server. It receives commands from the laptop over the local network and executes device level Instructions to control connected actuators. The actuators include two relays connected to a white and a yellow LED bulb (or lights) and a servo motor used for directional control. Specific GPIO pins on the ESP32 are allocated to each actuator, and standard libraries such as ESP32Servo.h and WebServer.h are used to manage real-time control and network communication.

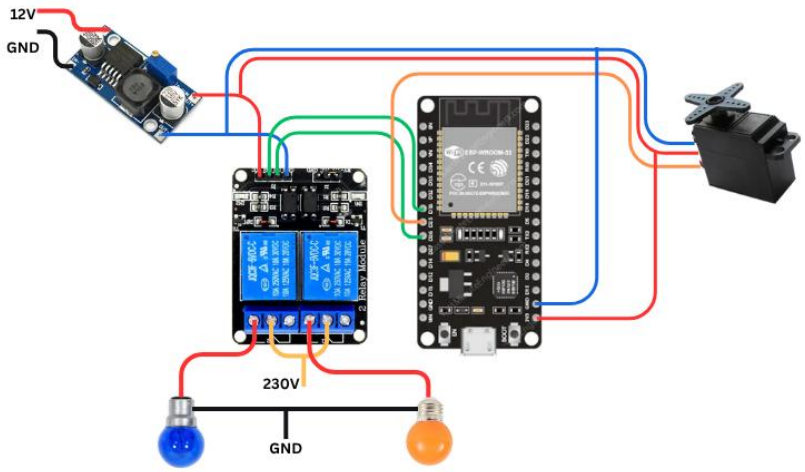


Fig. 3. Wiring Diagram

The complete wiring configuration of the ESP32 with its connected relays and servo motor is Illustrated in Fig. 3, which provides a clear schematic of all pin assignments and power lines. Additionally, the physical arrangement of the system including the placement of the webcam, user writing surface, laptop and ESP32 actuator module is shown in Fig. 4. This practical concept is more suitable for deployment in assistive environments such as rehabilitation centers or home automation and can be used for non-technical people as well. This setup demonstrates a low cost, non-invasive solution capable of interpreting natural human handwriting and converting it into actionable commands for controlling physical devices.

Fig. 4. Physical setup

4 RESULTS

Fig. 5 shows how input, process and output works. When giving input as handwritten instructions then the process extract information according to the given prompt to the model and get system evaluation results as These Evaluation results get for analysis how model works efficiency and accurately with edge devices. Further calculate Model accuracy level by comparing given outputs according to the given input instructions. Finally observed the Physical system behaviors.

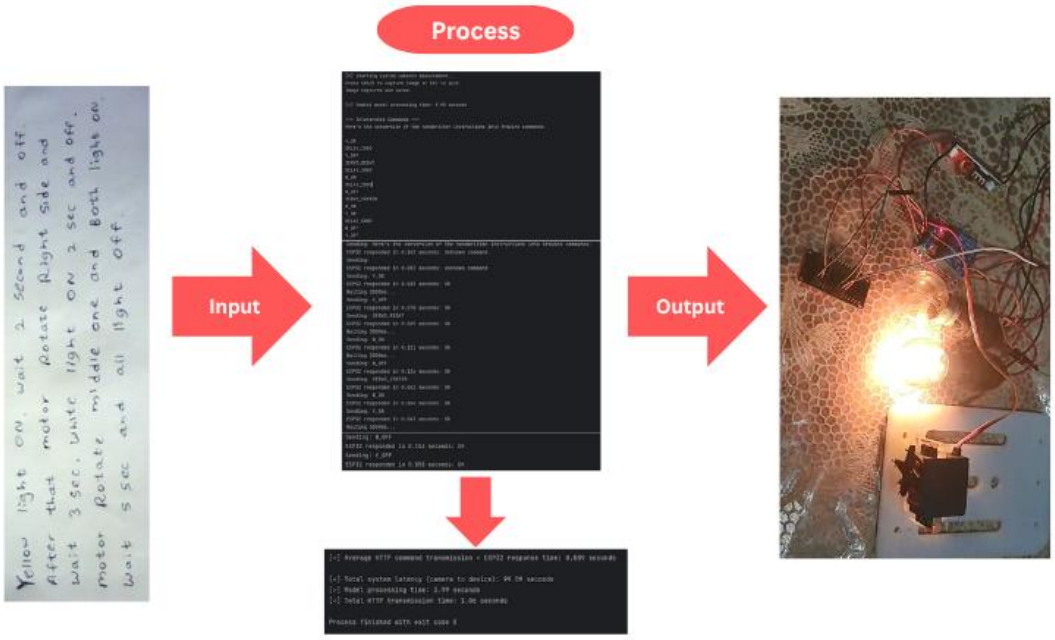


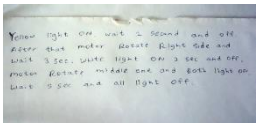
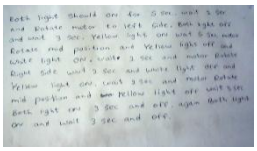
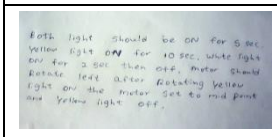
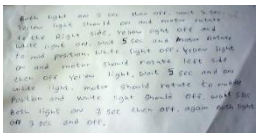
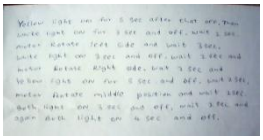
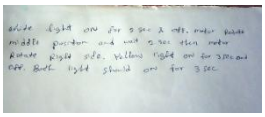
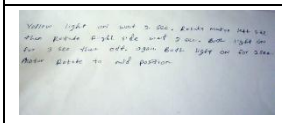
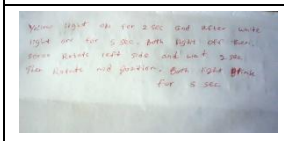
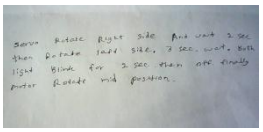
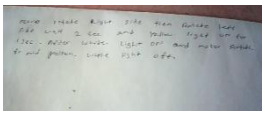
Fig. 5. Main Process

The proposed LLM-based IoT control system was evaluated across ten test cases with different handwriting styles, each involving handwritten instructions processed by a large language model and transmitted to an ESP32-based device over HTTP. The system's performance was measured based on five key metrics such as model processing time, average HTTP transmission time (including ESP32 response), total HTTP transmission time, total system latency (from camera capture to device actuation), and model accuracy. The model processing time ranged between 2.45 and 4.41 seconds, depending largely on the complexity, legibility, and length of the handwritten input. More complex or unclear instructions required additional context processing, which led to longer inference durations, as observed in test cases 4 and 5.

The average HTTP command transmission time, including the ESP32's response time, was generally low, ranging from 0.067 to 0.775 seconds. However, sometimes, such as in test case 2 (0.775 seconds), likely resulted from temporary network congestion, fluctuations in Wi-Fi strength or processing delays within the ESP32 due to handling compound instructions. The total HTTP transmission time showed more variability, with most cases completing within 1–2 seconds, but a few outliers such as test case 2 (15.50 seconds) and test case 9 (2.14 seconds) experienced longer delays. These inconsistencies can be attributed to unstable Wi-Fi connections, higher command complexity, or additional hardware initialization time, particularly when controlling relay or servo components.

The total system latency from image capture to successful device actuation ranged from 28.59 to 99.09 seconds. The highest latency observed (99.09 seconds in test case 1) may have been due to a combination of slow image acquisition, longer model inference, and network-induced transmission delays. Despite these outliers, most test cases demonstrated acceptable end-to-end performance, especially for applications where a few seconds of delay is tolerable, such as assistive technology for speech-impaired users. In terms of accuracy, the system performed well, achieving 100% command recognition in seven out of ten cases as Table 1 shows.

Table 1. Model Evaluation

Test Cases	Model processing time (seconds)	Average HTTP command Transmission time + ESP32 Response time (seconds)	Total HTTP transmission time (seconds)	Total system latency (Camera to device) (seconds)	Model Accuracy
	3.99	0.089	1.06	99.09	100%
	2.92	0.775	15.50	70.27	100%
	2.57	0.067	1.01	44.16	100%
	4.30	0.077	2.00	56.3	97%
	4.41	0.103	2.16	58.60	100%
	3.95	0.085	1.02	35.73	100%
	2.45	0.099	0.59	43.10	70%
	3.07	0.089	2.14	77.32	100%
	2.53	0.068	0.75	30.79	100%
	2.56	0.081	0.65	28.59	90%

The few instances of reduced accuracy such as 70% in test cases 7 and 8, and 90% in test case 10 were likely caused by issues such as poorly written input, ambiguous phrasing, or suboptimal camera conditions (e.g., lighting, focus, or angle). These factors could hinder the model's ability to correctly interpret the intent behind the handwritten text. Overall, the results validate the feasibility of using LLMs in edge-device-based IoT applications. They also highlight the importance of robust handwriting input and a stable communication infrastructure to ensure consistent performance in real-world use.

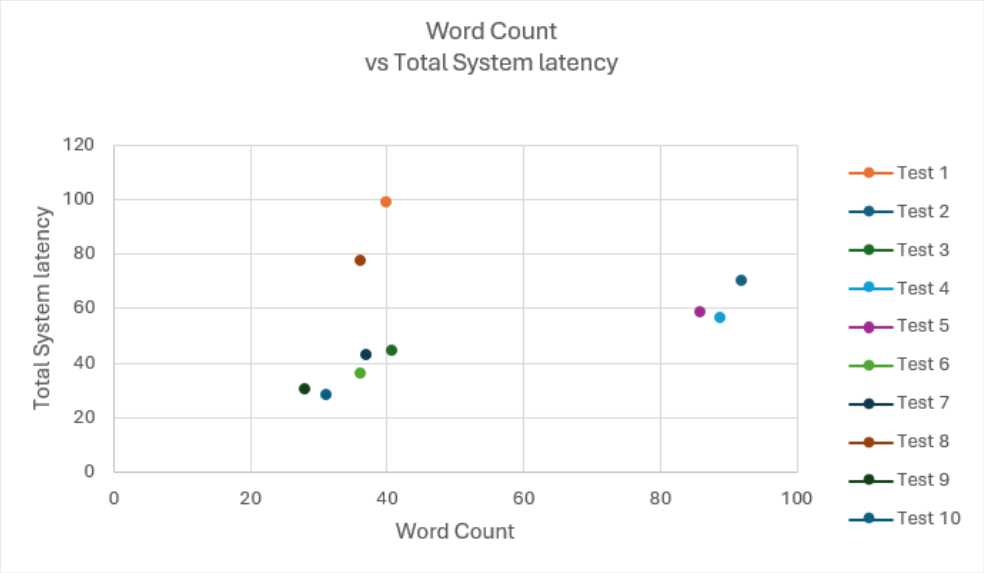


Fig. 6 - Word Count vs Total System Latency

The relationship between the number of words in a handwritten instruction and the total system latency (Fig. 6) is Illustrated in the scatter plot labeled "Word Count vs Total System Latency." Each data point represents a test case, with word count on the x-axis and total system latency (from camera capture to device response) on the y-axis. The results show a non-linear correlation between word count and latency. While an increase in word count generally suggests higher processing demand, the latency does not scale proportionally. For example, test case 1, which had a relatively low word count (~40 words), exhibited the highest latency of approximately 99 seconds. In contrast, test cases 9 and 10, which contained the highest word counts (around 90–95 words), had significantly lower latencies of approximately 58 and 28 seconds, respectively. This observation indicates that factors beyond word count such as network fluctuations, model queuing time, or hardware readiness play a substantial role in determining overall latency. That said, there is a moderate trend where shorter instructions tend to produce faster responses, as seen in test cases 3, 6, and 7, all with word counts below 40 and latencies below 45 seconds. However, test cases 2 and 4 demonstrate that even moderately sized inputs can result in unexpectedly high latency due to other influencing factors, such as ESP32 response delays or communication bottlenecks. Overall, while word count contributes to processing time, it is not the sole determinant of system latency, and optimizing other system components particularly model inference efficiency and communication reliability is essential for consistent real-time performance.

8 Conclusion

This research presents a novel handwriting-based IoT control system tailored to assist speech-impaired users in interacting with their environment using natural, intuitive inputs. By combining real-time handwritten instruction capture via a webcam with the semantic reasoning capabilities of a Large Language Model specifically Gemini Flash 2.0 the system effectively bridges the gap between human intention and machine execution without relying on traditional speech, buttons or complex gesture recognition. The integration of a low-cost ESP32 microcontroller enables seamless control of physical devices such as relays and servo motors, making the solution both accessible and scalable. Unlike conventional systems that require heavy preprocessing or dedicated OCR pipelines, this approach streamlines the user interaction flow and reduces system complexity by leveraging the visual intelligence of modern LLMs. The architecture is modular, adaptable and robust enough for real-world deployment in assistive technology settings, including smart homes, clinics and learning environments. Future enhancements may include offline LLM deployment on edge devices, multilingual handwriting support and broader actuator integration, further increasing the accessibility and autonomy of speech-impaired individuals. Overall, the system demonstrates how AI-driven natural interface design can redefine inclusive Human computer interaction.

7 References

[1] J.-H. Sun, T.-T. Ji, S.-B. Zhang, J.-K. Yang, and G.-R. Ji, 'Research on the Hand Gesture Recognition Based on Deep Learning', in *2018 12th International Symposium on Antennas, Propagation and EM Theory (ISAPE)*, Dec. 2018, pp. 1–4. doi: 10.1109/ISAPE.2018.8634348.

[2] J. Z. Lim, J. Mountstephens, and J. Teo, 'Eye-Tracking Feature Extraction for Biometric Machine Learning', *Front. Neurorobotics*, vol. 15, Feb. 2022, doi: 10.3389/fnbot.2021.796895.

[3] L. Arnold *et al.*, 'A Systematic Literature Review of Eye-Tracking and Machine Learning Methods for Improving Productivity and Reading Abilities', *Appl. Sci.*, vol. 15, no. 6, Art. no. 6, Jan. 2025, doi: 10.3390/app15063308.

[4] K. Gao *et al.*, 'Challenges and solutions for vision-based hand gesture interpretation: A review', *Comput. Vis. Image Underst.*, vol. 248, p. 104095, Nov. 2024, doi: 10.1016/j.cviu.2024.104095.

[5] 'Introducing Gemini 2.0: our new AI model for the agentic era', Google. Accessed: Jun. 01, 2025. [Online]. Available: https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/

[6] Y. Zhang *et al.*, 'Potential of multimodal large language models for data mining of medical images and free-text reports', *Meta-Radiol.*, vol. 2, no. 4, p. 100103, Dec. 2024, doi: 10.1016/j.metrad.2024.100103.

[7] L. A. Kadiyala, O. Mermer, D. J. Samuel, Y. Sermet, and I. Demir, 'The Implementation of Multimodal Large Language Models for Hydrological Applications: A Comparative Study of GPT-4 Vision, Gemini, LLaVa, and Multimodal-GPT', *Hydrology*, vol. 11, no. 9, Art. no. 9, Sep. 2024, doi: 10.3390/hydrology11090148.

[8] E. Davis and G. Marcus, 'Commonsense reasoning and commonsense knowledge in artificial intelligence', *Commun ACM*, vol. 58, no. 9, pp. 92–103, Aug. 2015, doi: 10.1145/2701413.

[9] J. Wei *et al.*, 'Chain-of-Thought Prompting Elicits Reasoning in Large Language Models', *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 24824–24837, Dec. 2022.

[10] Z. Yang *et al.*, 'The Dawn of LMMs: Preliminary Explorations with GPT-4V(ision)', Oct. 11, 2023,

arXiv: arXiv:2309.17421. doi: 10.48550/arXiv.2309.17421.

[11] T. Kojima, S. (Shane) Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, 'Large Language Models are Zero-Shot Reasoners', *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 22199–22213, Dec. 2022.

[12] Z. Zhang, A. Zhang, M. Li, H. Zhao, G. Karypis, and A. Smola, 'Multimodal Chain-of-Thought Reasoning in Language Models', May 20, 2024, *arXiv*: arXiv:2302.00923. doi: 10.48550/arXiv.2302.00923.

[13] 'Integrating vision-based AI and large language models for real-time water pollution surveillance - Samuel - 2024 - Water Environment Research - Wiley Online Library'. Accessed: Jun. 01, 2025. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/wer.11092

[14] M. Hwang, 'LibGuides: AI for Healthcare Research & Education: Prompting Best Practice'. Accessed: Jun. 01, 2025. [Online].
Available: https://guides.hsict.library.utoronto.ca/THP_AI/prompting

[15] B. Xiao, B. Kantarci, J. Kang, D. Niyato, and M. Guizani, 'Efficient Prompting for LLM-based Generative Internet of Things', Jun. 14, 2024, *arXiv*: arXiv:2406.10382. doi: 10.48550/arXiv.2406.10382.