EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

# Building an Intelligent Home Perception System Based on Multi-Modal Information Interaction

Guo Zhanmiao[1,*],Qian Zhongli[2]

[1] Department of Electronic Information and Communication Engineering, Applied Technology College of Soochow University, Jiangsu, Suzhou; 215000, China
[2]Faculty of Innovation Engineering, Macau University of Science and Technology, Macau; 999078, China

## Abstract

In response to the problems of single interaction modality and weak perception ability in traditional smart homes, this paper proposes a multi-modal information perception Artificial Intelligence(AI) model invocation framework. It schedules visual, voice, and sensor data through natural language prompts, and combines the zero-shot visual recognition method of the cloud-based visual-language hybrid large model workflow to achieve cross-scene generalization ability without labeled training. This framework can innovatively solve the problems of heterogeneous data fusion and insufficient computing power of edge devices. Experimental results show that the multi-modal smart home perception system designed in this paper achieves an accuracy rate of over 90% in environmental perception and a precision rate as high as 92% in user intention recognition, which can provide new ideas and practical foundations for the multi-modal perception of future smart home technology.

**Keywords:** Smart Home, Multi-modal Information Fusion, Multi-modal Large Model, Human-Computer Interaction, Internet of Things

Received on 22 September 2025, accepted on 19 March 2026, published on 31 March 2026



doi: 10.4108/eetsis.10349

## 1. Introduction

With the rapid development of Internet of Things(IoT) and multi-modal large model technologies, the combination of smart homes and multi-modal perception technology is attracting widespread attention. Internet of Things devices have become popular in homes. D.N. Mekuria et al. argue in their literature review on smart home reasoning that the development and popularization of IoT has brought many challenges and opportunities for the development of smart home systems [1], enhancing the development of smart homes. However, most such devices still require explicit manual control, and as the number of sensors and actuators continues to grow, the complexity of this task also increases. This surge is particularly evident in the voice assistant market, with over 320 million voice assistants installed in homes globally, and 35% of the US population owning such devices [2]. Although early boxology and its extensions were pioneering in researching visual representations for AI systems [3,4], their primary focus was on abstract representations to elicit and analyze design patterns that define internal architectures. A key limitation, however, was that these representations remained largely confined to visual symbolism, and the translation of these diagrams into machine-operable representations was inadequately supported. Multi-modal human-computer interaction can effectively identify and integrate various types of information, with multiple information input/output modes, providing smart home users with a more natural and efficient interaction experience [5,6]. Baidu's Enhanced Representation through Knowledge Integration(ERNIE) focuses more on Chinese language

*Corresponding author. Email: 15809285941@163.com

tasks, thus having local advantages in some aspects [7]. Joseph C. Kush studied in 2025 how real-time data from different sensors can improve AI models' understanding of the surrounding environment, user environment, and physical conditions, as well as interaction with conversational AI models such as ChatGPT 4.0 [8].

We adopt Tongyi Qianwen 2.5-VL to deeply integrate visual and language modalities, leveraging large-scale pre-training and cross-modal attention mechanisms to achieve semantic-level understanding of image content. We use Baidu Voice technology for speech recognition and speech synthesis, and utilize ERNIE for dialogue content generation. Addressing the passive state mode where interaction is limited to basic command-response, our designed system can build a "hearing-vision-environmental perception" trinity fusion architecture by integrating multi-dimensional perception technologies such as offline/online speech recognition, image processing, and environmental monitoring. The system innovatively introduces a multi-modal large model to achieve cross-modal feature recognition and collaborative calling on the ESP32-S3 processor, significantly improving the accuracy of interaction in complex home scenarios. At the same time, we explore how to use WiFi Mesh networks to build distributed sensing nodes, which can also be applied to fire safety early warning scenarios. The system can synchronously analyze sensor data, audio data, and image data, greatly reducing the false alarm rate through multi-modal evidence chains, integrating general ambient intelligence research into IoT fields beyond smart homes. This research result breaks through the traditional paradigm of "emphasizing control and neglecting perception" in the smart home field, providing a new path for smart homes to evolve towards proactive service systems.

The system design includes a cloud platform, a controller, and controlled devices. Controlled devices are divided into wired and wireless connection methods, all controlled by the Mesh gateway. The wired control domain uses a central controller STC15W4K32S4 to connect devices, while the wireless control domain consists of WiFi Mesh self-organizing network devices. The overall system design block diagram is shown in Figure 1.

The IoT cloud platform adopts Blinker cloud, providing APP, server, and device-side Software Development Kit(SDK), mainly used for remote operation of smart home devices on mobile phones and supporting voice recognition. The large model cloud platform utilizes the voice technology and multi-modal large model Application Programming Interfaces(APIs) provided by Alibaba Cloud and Baidu Cloud for speech recognition (STT), speech synthesis (TTS), image recognition (Qwen 2.5-VL), and combines a large language model (ERNIE) to process dialogue flows [9].

The multi-modal intelligent perception controller serves as the processing core of the system. It is equipped with ESP32-S3 for connecting various perception and interaction devices such as voice and image modalities. It supports the recording, sending, and playing of voice modalities, as well as the capture, display, and recognition of image modalities. In conjunction with the cloud platform, it can realize functions such as access control follow detection, fire risk investigation, voice operation of smart home devices, intelligent temperature control, intelligent timing, and user habit summarization.

Finally, the controlled devices are managed through wired and wireless perception control. Wired intelligent perception control is mainly managed by the central controller, integrating key components such as the 4G module SIM7020, Wi-Fi module ESP8266, offline voice recognition module ASR PRO, and the core processor STC15W4K32S4, enabling control via 4G or voice even when the Wi-Fi network is disconnected. Wireless intelligent perception control mainly relies on the Mesh gateway for remote or intelligent perception control. When the gateway is disconnected, the interconnected Mesh devices ensure the stable operation of the system, and users can control devices through a remote control or touch screen.

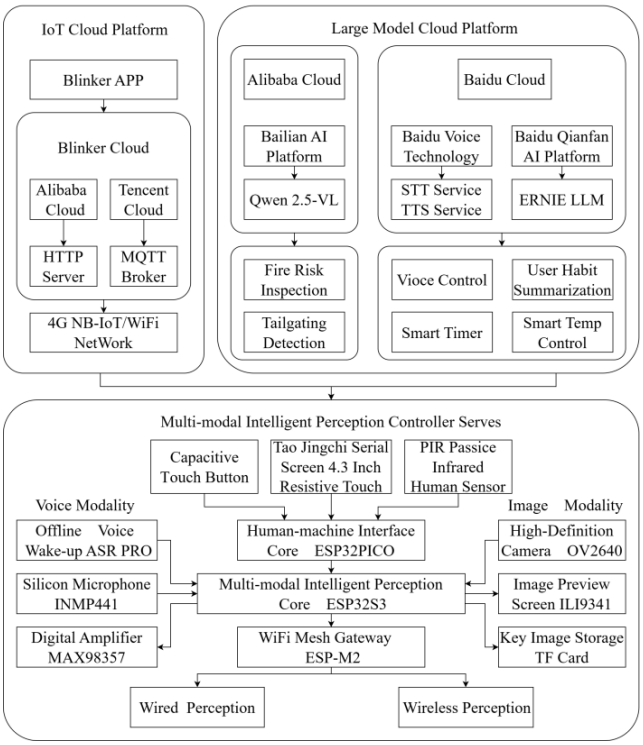



**Figure 1.** Overall System Design Block Diagram

## 2. Data description

### 2.1. Visual modality perception test of the multi-modal intelligent perception controller

For the visual modality perception test of the multi-modal intelligent perception controller, a total of 6 groups of data were verified, with 20 images in each group, and 5 rounds of shooting with different shooting angles in each

round, for a total of 100 images per group. The results are shown in Table 1. The average recognition accuracy is above 90%, and when the image is blurred or completely black and white, making it impossible to judge, a block alarm can be automatically pushed, meeting the smart home inspection needs.

Table 1. Visual Modality Perception Accuracy Test Results

| Group | Recognition Type | Round 1 | Round 2 | Round 3 | Round 4 | Round 5 | Average Accuracy |
|---|---|---|---|---|---|---|---|
| 1 | Fire Group 1 | 0.95 | 0.80 | 0.90 | 1.00 | 0.85 | 0.90 |
| 2 | Fire Group 2 | 0.80 | 0.80 | 0.85 | 0.95 | 1.00 | 0.88 |
| 3 | Smoke Group 1 | 1.00 | 0.95 | 0.95 | 0.80 | 0.90 | 0.92 |
| 4 | Smoke Group 2 | 0.85 | 0.90 | 0.90 | 0.85 | 1.00 | 0.90 |
| 5 | Access-Control Follow Group 1 | 0.90 | 1.00 | 0.85 | 0.80 | 0.90 | 0.89 |
| 6 | Access-Control Follow Group 2 | 0.95 | 0.95 | 0.80 | 0.85 | 0.90 | 0.89 |

## 2.2. Image modality perception design method

In terms of image modality interaction, we adopt the Tongyi 2.5-VL model, which possesses advanced technologies such as dynamic resolution support and Multi-modal Rotary Position Embedding (M-RoPE). This allows the model to dynamically adjust the number of image tokens (in the generative AI data processing, each piece of data divided into the smallest granularity is called a token) according to the original size of the input image, thereby fully preserving the detailed information in high-resolution images while ensuring efficient computation. At the same time, the M-RoPE, extended to video spatio-temporal modeling, can effectively capture spatio-temporal associations in text, images, and videos, achieving unified image and video understanding.

The image modality perception design utilizes model distillation and quantization techniques of the Tongyi 2.5-VL model, and is supported by hardware-algorithm co-design strategies that exploit token similarity to achieve high energy efficiency beyond mere token reduction[10] . Specifically, through knowledge distillation, the rich features extracted from the large model are transferred to a lightweight network, while the model is quantized to reduce the occupation of computing resources, enabling edge devices to achieve efficient inference with lower computing power. Hybrid precision technology is used during the training process, utilizing a combination of half-precision floating-point numbers (FP16) and single-precision floating-point numbers (FP32), which ensures model accuracy while reducing the consumption of computing resources and accelerating model inference speed. The adoption of a Multi-Scale Feature Integration Mechanism enables the model to extract effective information at different resolutions, thereby enhancing its ability to detect anomalies such as smoke and fire.

As shown in Figure 2, the image recognition steps are as follows: The system triggers image capture by the ESP32-S3 camera driver through serial port commands or timed intervals. The collected raw image is Base64 encoded using the Base64 library. After data encoding is complete, it is serialized into a JSON string along with the recognition requirement Prompt. The RESTful style API is then called to upload the JSON string to the cloud for image preprocessing [11]. The Tongyi 2.5-VL model will recognize the content according to the Prompt requirements and execute alarm and linkage control based on whether it contains risky images. If the image is unclear, the user will be prompted to adjust the camera direction and a forced retry will be performed. If there are no risky images, a status of "normal" will be broadcast. After the camera adjustment is complete, the user can close the broadcast and preview screen.

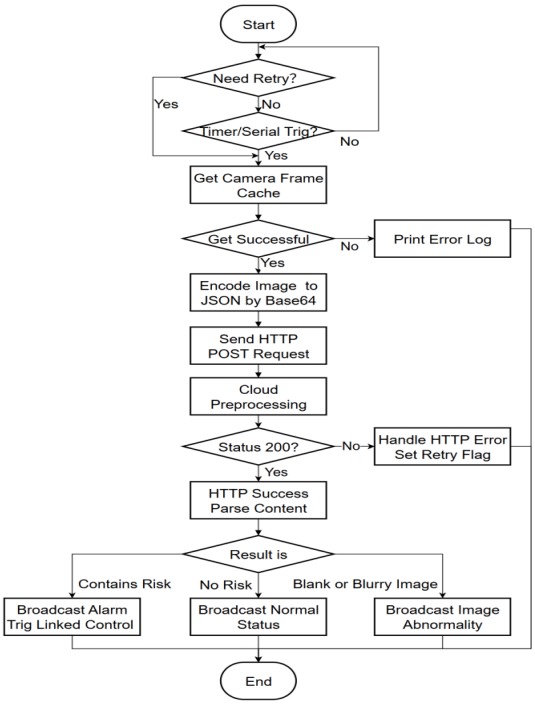


**Figure 2.** Image Modality Software Flowchart

### 2.2.1. Image preprocessing

After the cloud receives the image, it performs normalization, noise reduction, and size adjustment. Since the original grayscale or color value range of the image data is from 0 to 255, dividing it by 255 will convert it to a floating-point number between 0 and 1. Let the original image be represented by a matrix . The image preprocessing process is implemented using Equation (1).

$$I_p = \frac{I - I_{\min}}{I_{\max} - I_{\min}} \times 255 \quad (1)$$

Where $I_{min}$ and $I_{max}$ respectively represent the minimum and maximum grayscale values of the image. This step helps to eliminate the influence of different lighting conditions and ensures that the image data is within a uniform numerical range.

### 2.2.2 Structured output of image content recognition

To achieve structured output, the user-provided prompt is also used as a model input parameter. The processing is implemented using Equation (2).

$$R = f(I_p, \ P, \Theta) \quad (2)$$

Where $I_p$ represents the preprocessed image data matrix; P represents the prompt string given by the user, which requires the model to structurally output a JSON format string containing the key names code (operation code) and say (voice feedback); Θ represents the parameter set of the cloud large model; and R represents the structured JSON result returned by the model. The code key value can serve as a code indicating whether fire signs are present or as the number of access control personnel identified, and the say key value can serve as the voice broadcast content.

During the cloud processing, the model internally uses a dynamic resolution mechanism to adaptively convert input images of different sizes into fixed-length image token sequences [12]. Let the input image size be H × W. Then the resolution compression function used, g(·) , can be expressed as Equation (3).

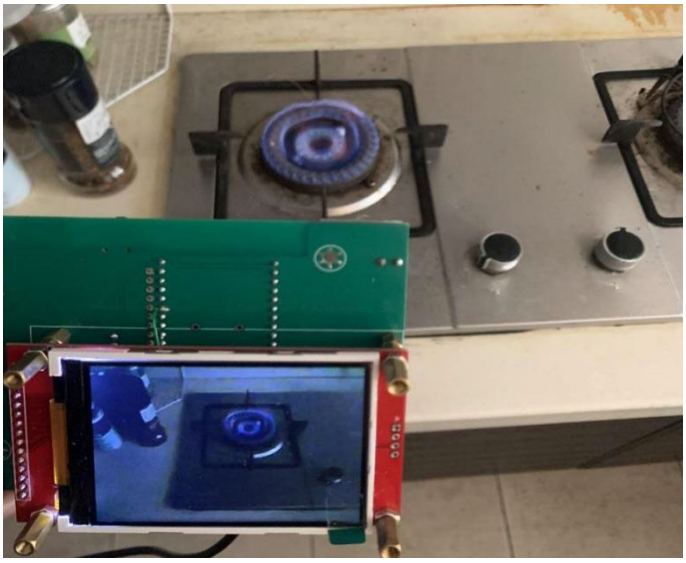
$$T = g(H, W) = \left\lceil \frac{H \times W}{s} \right\rceil \quad (3)$$

Where s is the compression factor, and the obtained T is the number of image tokens generated. Since large models usually divide images according to a fixed patch size (e.g., 16×16 or 14×14) and then perform feature extraction for each patch, without additional compression, the theoretical number of patches for a 1600×1200 image is

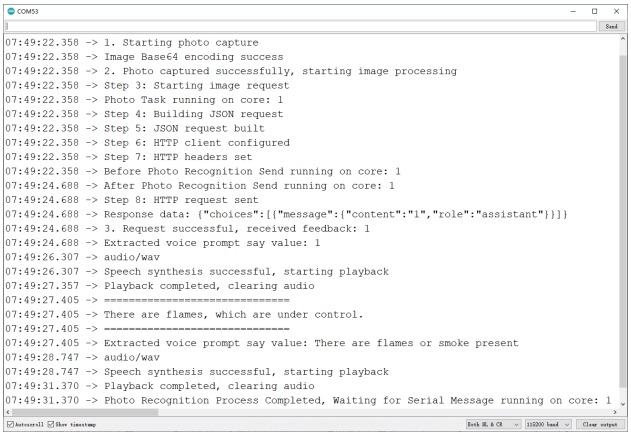
$$\left\lceil \frac{1600}{p} \right\rceil \times \left\lceil \frac{1200}{p} \right\rceil \quad (4)$$

Taking in Equation (4) with a patch size of 16 (p = 16), the number is approximately 7500 ($100 \times 75 = 7500$). However, due to the application of a dynamic resolution mechanism, feature compression, and MLP layers to merge tokens in practice, the number of tokens is significantly reduced. After these processes, each token actually represents 5 original patches, so 7500 divided by 5 gives 1500 tokens. Actual measurements show that an image with UXGA resolution occupies approximately 300KB, and the cloud consumes approximately 1500 tokens.

After cloud decomposition and recognition are complete, the returned string R is transmitted back to the controller via HyperText Transfer Protocol(HTTP). To further parse the cloud-returned result, the local processing module will extract the JSON data, where the code key value can serve as a code indicating whether fire signs are present or as the number of access control personnel identified, and the say key value serves as the voice broadcast content. After processing, the multi-modal intelligent perception controller can trigger corresponding operations based on the judgment, such as initiating a fire alarm or adding a follow-up record to the access control system. The fire risk identification web interface is shown in Figure 3(a), and the serial port results printing are shown in Figure 3(b) which are under control.



**(a)** ESP32-S3 web Server Interface




**(b)** ESP32-S3 recognition result serial port output

**Figure 3.** Recognition Output

### 2.2.3 Image display

Since the UXGA images (16001200 pixels) uploaded for platform recognition are large, to ensure that users can immediately know the specific shooting range during debugging, they are processed into 320240 pixel preview images that can be displayed on the preview screen. Because RGB format images are large, recording a 1600*1200 image requires approximately 3.8MB, while JPEG only requires about 100KB. Therefore, the OV2640 is configured for JPEG output. The OV2640 chip integrates a JPEG hardware encoder, which can quickly transmit data to the ESP32-S3 through the DVP interface. However, the screen based on the ILI9341 chip requires RGB565 format data, so software decoding is needed.

The pixel data generated by the JPEG decoding library is in little-endian format, meaning the low byte comes before the high byte. If this data is directly sent to the ILI9341, which inputs high bytes first, it will cause color channel misalignment and a garbled screen. Therefore, the program swaps the high and low bytes of the 16-bit color value of each pixel to prevent the garbled screen situation shown in Figure 4.

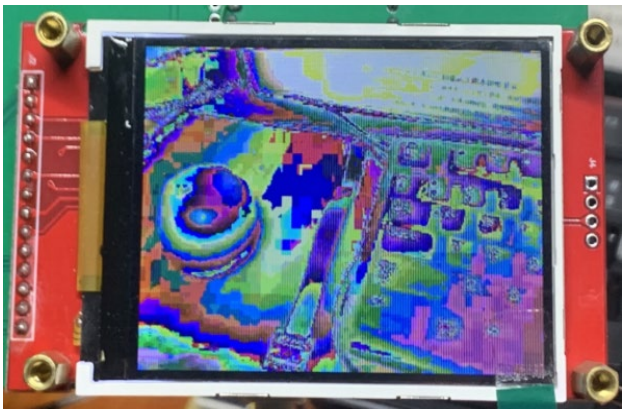


**Figure 4.** Garbled screen without Swapping high and low bytes

The preview screen image decoding function decomposes the JPEG image into several MCU (Minimum Coded Unit) blocks. JPEG uses Discrete Cosine Transform (DCT) to perform frequency domain transformation on image blocks, processing 8×8 pixel blocks. Decoding to the screen requires operation based on MCUs.

Let the width of each MCU block be mcu_w and the height be mcu_h. Let the overall width of the image (JPEG image width) be max_x and the height be max_y. For the right and bottom edges where a complete MCU is not filled, the code uses modulo and minimum value functions to calculate the width and height of the remaining parts, as shown in Equations (5) and (6):

$$\text{min\_w}=\min(\text{mcu\_w},\ \text{max\_x} \bmod \text{mcu\_w}) \quad (5)$$

$$\text{min\_h}=\min(\text{mcu\_h},\ \text{max\_y} \bmod \text{mcu\_h}) \quad (6)$$

If the current MCU block is completely within the image, the window size $\text{win}_w$ and $\text{win}_h$ are equal to mcu_w and mcu_h; otherwise, if it exceeds, it is forcibly set to min_w or min_h. The display position of each MCU block will be calculated by Equations (7) and (8):

$$\text{mcu\_x} = \text{JpegDec.MCUx} \times \text{mcu\_w} + \text{xpos} \quad (7)$$

$$mcu\_y = JpegDec.MCUy \times mcu\_h + ypos \quad (8)$$

In the above equations, xpos and ypos represent the offset values of the display start position, usually taken as 0 to indicate top-left alignment, or can be set according to user requirements to center the image. JpegDec. MCUx and JpegDec. MCUy respectively represent the index of the current MCU block in the horizontal and vertical directions. To improve speed and reduce screen tearing, DMA direct memory access and a double buffering mechanism are added to achieve a higher screen refresh rate. The processed screen display is shown in Figure 5.

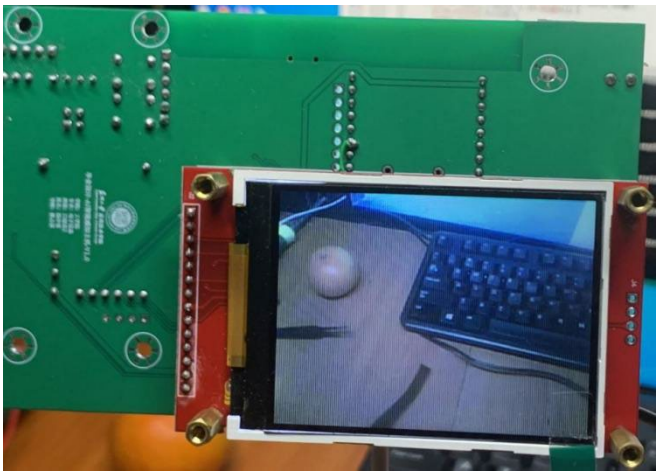
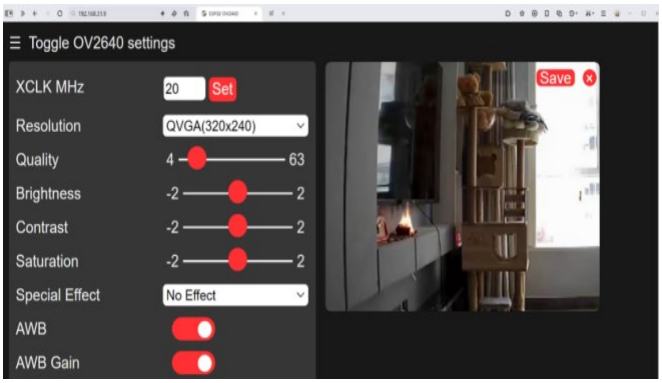


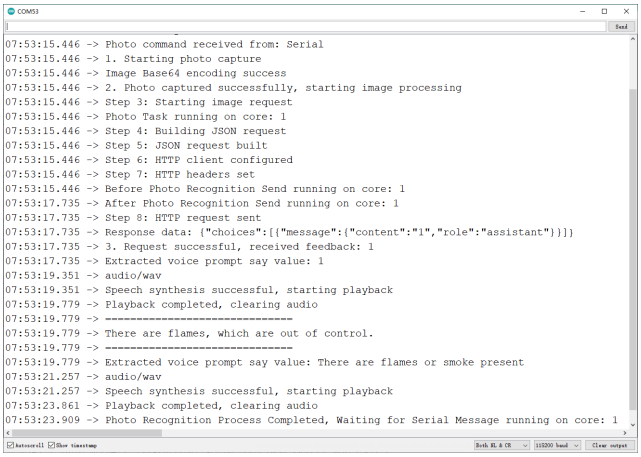
**Figure 5.** Processed Preview Screen

### 2.2.4 Image Recognition Modality Debugging

The debugging of the image recognition module mainly focuses on the stability of the Alibaba Cloud Bailian Large Model Platform interface program. As shown in Figure 6(a), when the independent image program is running, the system can clearly capture and recognize simulated fire, and the serial port can output the fire risk recognition results, as shown in Figure 6(b).




(a) Photo of the recognition scene




(b) Screenshot of serial port output

**Figure 6.** Image Recognition Debugging

When the image recognition modality and the voice recognition modality are running simultaneously, in some cases, the phenomenon of the photographing process not crashing but crashing before HTTP access occurs. After analysis, the reason is that the image buffer setting is unreasonable during the initialization of the ESP32CameraWebserver routine, which sometimes exceeds the array size and causes memory corruption problems. To this end, the image resolution and buffer size need to be dynamically adjusted according to the actual hardware conditions to ensure that memory resources are reasonably allocated. In the environmental safety risk monitoring experiment, the image model can accurately identify anomalies such as smoke and flames and trigger corresponding voice broadcasts and mobile push notifications, as shown in Figure 7.


**Figure 7.** Image Recognition Alarm

## 2.3 Voice modality perception test of the multi-modal intelligent perception controller

The voice modality test covered 6 types of typical control commands, 20 for each type, and 5 rounds of testing. Different ways of speaking were used in each round, such as expressing "a bit cold" for turning on the air conditioner to heat and "it's so dark" for turning on the lights, to test whether the model can correctly identify the user's intention. The test results are shown in Table 2. The system maintains an average intention recognition accuracy of over 92% in complex acoustic environments. When unidentifiable input is encountered, it will reply with "didn't hear clearly". When the large model backend is being upgraded, it can inform the user of the situation. When TTS broadcast generation fails, it can broadcast the locally stored "voice synthesis failed" prompt, meeting the smart home voice dialogue needs.

Table 2. Voice Dialogue Test Results

| Group | Recognition Type | Round 1 | Round 2 | Round 3 | Round 4 | Round 5 | Average Accuracy |
|---|---|---|---|---|---|---|---|
| 1 | Device Control Commands | 0.90 | 0.95 | 1.00 | 0.95 | 0.90 | 0.94 |
| 2 | Environment Query Commands | 0.95 | 0.90 | 0.90 | 1.00 | 0.85 | 0.92 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 3 | Scheduled Operation Commands | 1.00 | 0.95 | 0.95 | 0.90 | 0.90 | 0.94 |
| 4 | User Habit Summary | 0.85 | 1.00 | 0.95 | 0.90 | 0.95 | 0.93 |
| 5 | Visual Invocation Commands | 0.90 | 0.90 | 1.00 | 0.90 | 0.95 | 0.93 |
| 6 | Random Question Commands | 0.95 | 0.95 | 0.90 | 0.85 | 1.00 | 0.94 |

## 2.4 Voice modality perception design method

The voice modality interaction uses Baidu Voice technology for speech recognition and synthesis, and integrates the ERNIE4.5 Wenxin large model into the workflow to deeply analyze and record device operation intentions and user habits [13], which is applied to scenarios such as intelligent temperature control, intelligent timing, and visual linkage to achieve precise interaction between voice commands and device status.

The voice modality program starts with a button or serial port trigger. The serial port command is implemented by the ASRPRO offline voice recognition module. After the system plays the "I'm here" prompt, it allocates memory and continuously reads the Inter-IC Sound(I2S) bus for recording. When there is no sound for more than 1.5 seconds, the recording stops. After the recording stops, the audio data will be Base64 encoded and sent to the speech recognition API.

The recording sampling rate uses 16000Hz, generating approximately 32KB of data per second. The recording duration is 10 seconds to 15 seconds. After Base64 encoding, the volume increases by one-third, with an average size between 300KB and 400KB. Using a low-latency 100Mbps broadband to upload to the cloud for recognition takes approximately 20-80 milliseconds, meeting the requirements for voice dialogue. The audio volume is calculated by Equation (9) using the average amplitude $\bar{A}$ of the audio data collected by I2S.

$$\bar{A} = \frac{1}{N}\sum_{i=1}^{N}|x[i]| \quad (9)$$

Where x[i] represents the audio amplitude of each sampling point, with a size of two bytes and a range of 0-65535. N represents the number of data samples read each time, generally 1024, meaning 512 samples are used for each calculation, and the average value is calculated 32 times per second. The calculated result is compared with a threshold (generally below 30 in a quiet environment) to measure the average volume level of the current audio signal, determine whether the user has stopped speaking and entered a silent state, so as to subsequently determine if silence exceeds 1.5 seconds to pause recording and upload for cloud speech recognition. If the set maximum length of 15 seconds is exceeded, it will be forcibly stopped and uploaded. If the request is successful, the HTTP response status code will be 200. The system calls the large model API to generate a response based on the speech recognition content and system prompts. After extracting the operation code, it broadcasts the result through voice synthesis. If cloud recognition fails, response generation fails, or voice synthesis fails, a "synthesis failed" prompt is played. The voice modality software flowchart is shown in Figure 8. If successful PSRAM memory allocation is detected, the recording operation continues; if allocation fails, a memory allocation error is printed to the serial port. Both speech recognition STT failure and response TTS failure use audio broadcast to provide feedback to the user.

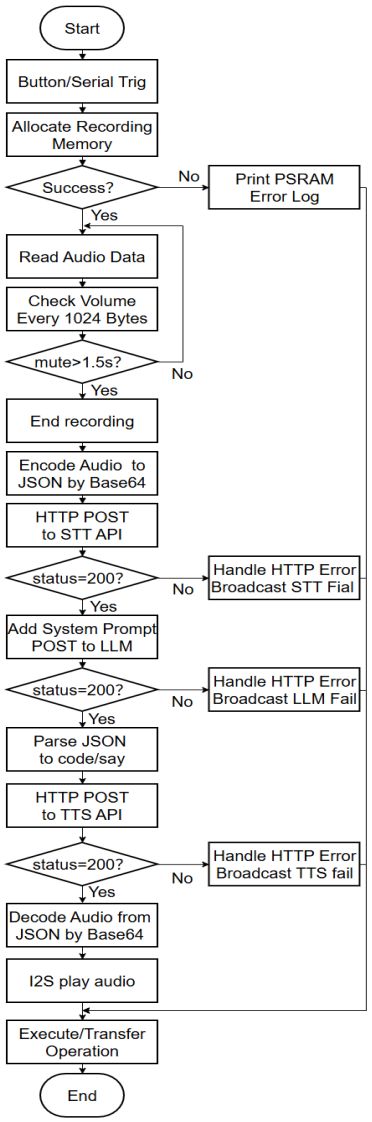



**Figure 8.** Voice Modality Software Flowchart

The INMP441 digital microphone recording preprocessing includes noise reduction, normalization, and framing of the speech signal. Let the collected raw speech signal be a discrete-time sequence x[n]. First, the Short-Time Fourier Transform (STFT) is used to obtain the frequency domain representation as shown in Equation (10).

$$X[k,m] = \sum_{n=0}^{N-1} x\,[n+mR]\,w[n]\,e^{-j2\pi kn/N} \quad (10)$$

Where w[n] is the window function, N is the frame length, and R is the frame shift. To reduce the interference of background noise on speech recognition, a noise suppression algorithm based on median filtering is adopted, as shown in Equation (11).

$$\tilde{X}[k,m] = \text{median}\{X[k,m-L],\ldots,X[k,m+L]\} \quad (11)$$

Where L is the set neighborhood size. The preprocessed frequency domain features are recovered to the time domain through inverse transformation, and the signal amplitude is distributed within a uniform interval through normalization operation, providing a stable input for subsequent speech recognition.

The preprocessed voice data is uploaded to the cloud by calling Baidu STT's API. The successfully recognized voice content (denoted as S) and the user-given prompt (denoted as P) are uploaded together to the large language model ERNIE for processing, denoted as h(·), where Φ represents the model parameters. The design of the voice processing API management function can be expressed by Equation (12) to obtain structured output.

$$Y = h(S, \mathrm{P}, \Phi) \quad (12)$$

In Equation (12), Y is the structured JSON result returned by the cloud, containing two key-value pairs: code represents the specific operation code, which determines the subsequent device control actions; say represents the voice feedback information for the user. After parsing using the ArduinoJSON library, the serial port feedback is shown in Figure 9.

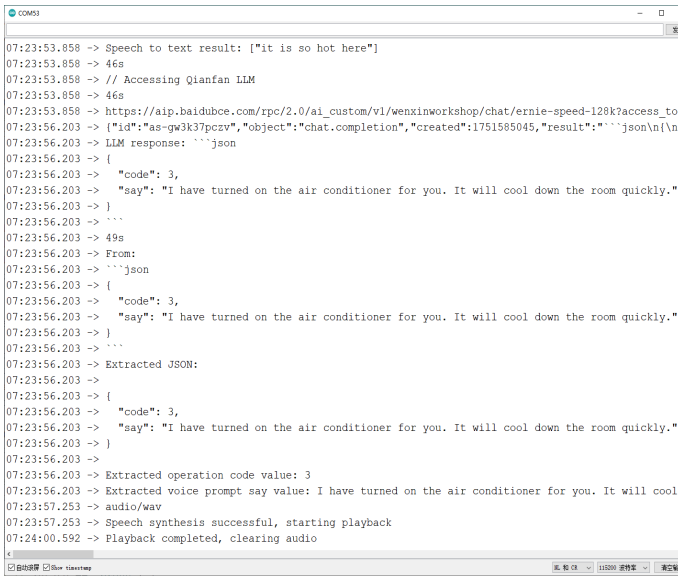



**Figure 9.** Voice Modality Return Value

Since the interface designs of various platforms are consistent with OpenAI's API, only the first half of the URL address and the API key need to be modified to achieve rapid switching between different models and platforms, such as ChatGPT, Qwen, DeepSeek, and other models can be accessed. Dense models can achieve faster dialogue and image recognition using ordinary APIs. Chain-of-thought models like DeepSeek generally require replacing the distillation model and the platform for the streaming dialogue API [14], improving the feedback speed to within 5 seconds to ensure a smooth user experience.

The system uses the LD3320 voice recognition chip to achieve the wake-up function [15]. The chip integrates a dual-core heterogeneous architecture and uses hardware acceleration to implement a multi-level signal processing pipeline: the front end uses an adaptive noise suppression (ANS) algorithm for acoustic preprocessing, coupled with an improved MFCC feature extraction module (26-channel Mel filter bank, frame length 32ms, frame shift 16ms). The feature vectors processed by the fixed-point FFT acceleration unit enter the hybrid recognition engine. This engine deeply integrates the DTW dynamic time warping algorithm (supporting 50 personalized templates) and the GMM-UBM universal background model. With the support of the 8MB Flash voice library embedded in the chip, it achieves an ultra-high wake-up word detection rate. It communicates with the main control via the UART interface. The recognition results are shown in Figure 10. The model accurately wakes up, determines the user's intention after voice recognition, and generates operation codes and reply statements.

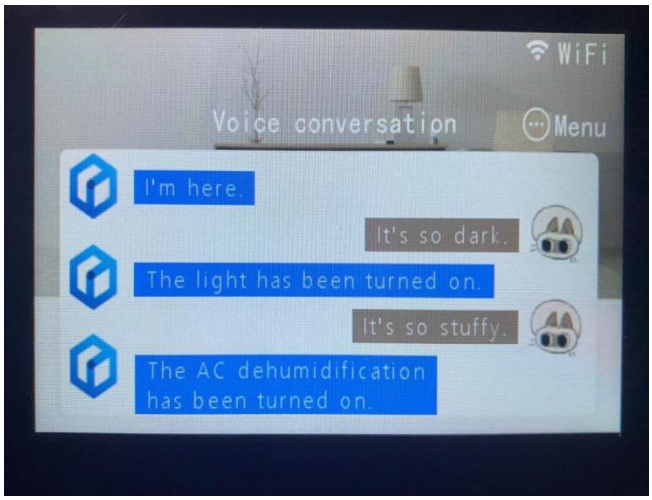



**Figure 10.** Voice Dialogue Test Results

User habit summarization primarily relies on the historical conversation log feature within the platform workflow settings. Through scheduled large model calls, weekly and monthly user habit analyses are generated and stored in the knowledge base

`DeviceControlCode_UserHabits`. This enables the model to retrieve operation codes and long-term habit summaries during voice interactions, rather than being limited to recent conversation history.

## 2.5 Multimodal Fusion Mechanism Experiment

### 2.5.1 Experiment Setup and Evaluation Metrics

This experiment built a dual-model test platform to compare the performance of a traditional object detection model (YOLOv8) and a multimodal large model (Qwen3-Omni-Flash) in fire detection tasks. In terms of hardware environment, the traditional model was deployed on an embedded development board to simulate the real-time monitoring scenario of a single-chip microcomputer; the large model was called through the Alibaba Cloud DashScope API Platform .

The dataset adopted a self-constructed fire scene dataset, including 95 real fire images (covering indoor kitchen fires, electrical short-circuit fires, smoke diffusion and other scenarios) and 5 normal scene images (living rooms, kitchen fire-free environments, outdoor streets, etc.). The image resolution ranged from 500×500 to 3000×2258, and the size of a single image was ≤10MB, which met the input specifications of the large model and the transmission requirements of embedded devices. The dataset was obtained through web crawlers and manually screened to ensure scene diversity and annotation accuracy. All fire images were labeled as positive samples (with fire), and normal scene images were labeled as negative samples (without fire) to verify the detection accuracy and generalization ability of the models. A selection of flame images is presented in Figure 11.

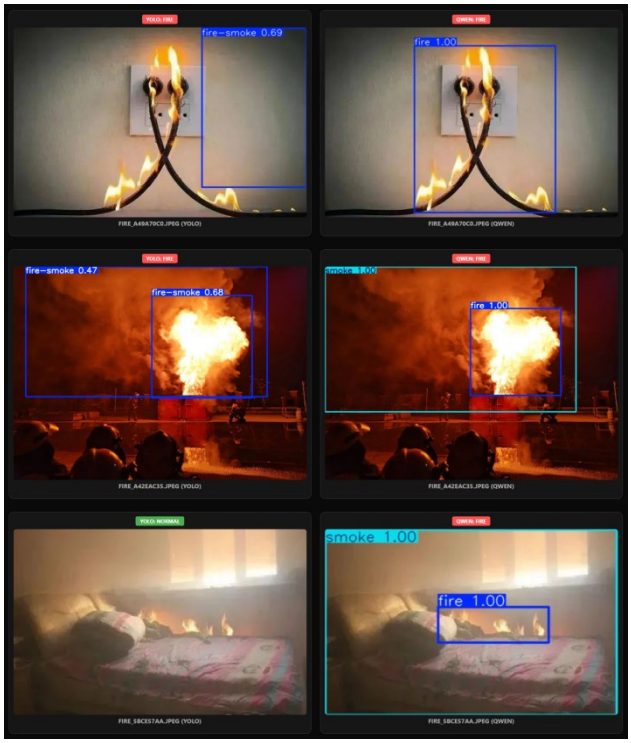



**Figure 11.** A Selection of Flame Images

### 2.5.2 Dual-Model Performance Comparison

The experiment tested YOLOv8 (traditional object detection model) and Qwen3-Omni-Flash (multimodal large model) respectively, and counted the detection results of 100 test images (95 fire images + 5 normal images). The experiment adopted core evaluation metrics for fire detection tasks, including Accuracy, Recall, Precision, F1-Score, and Average Response Time. The definitions of the metrics are as follows:

Accuracy: The proportion of test samples where the detection result is consistent with the true label, reflecting the overall detection ability of the model;

Recall: The proportion of real fire samples that are successfully detected as fires, reflecting the model's ability to control missed detections (the lower the value, the more serious the missed detections);

Precision: The proportion of samples judged as fires by the model that are actually real fires, reflecting the model's ability to control false alarms (the lower the value, the more serious the false alarms);

F1-Score: The harmonic mean of Recall and Precision, balancing missed detections and false alarms to comprehensively evaluate model performance;

Average Response Time: The average time consumed by the model to process a single image, reflecting the real-time performance of the model.

The dual-model detection results and total latency are shown in Figure12. From the test results, for the analysis of the 99/100 and 100/100 test images: when the Ground Truth (GT) is "True" (i.e., fire exists), the YOLO model failed in detection with two missed alarms, incorrectly judging them as "False" and taking 0.80 seconds and 1.08 seconds respectively; in contrast, the Qwen model correctly judged them as "True", with response times of 0.42 seconds and 0.53 seconds respectively. The performance comparison data is shown in Table 3, and the core statistical indicators (TP, FP, TN, FN) are shown in Table 4.

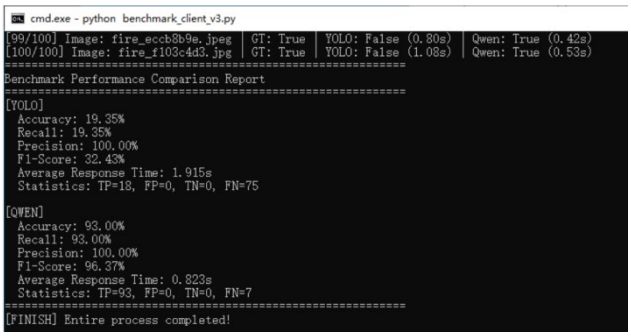



**Figure 12.** Dual-Model Detection Results and Total Latency

Table 3. Dual-Model Performance Comparison

| Evaluation Metrics | Traditional Object Detection Model (YOLOv8) | Multimodal Large Model (Qwen3-Omni-Flash) |
|---|---|---|
| Accuracy | 19.35% | 93.00% |
| Recall | 19.35% | 93.00% |
| Precision | 100.00% | 100.00% |
| F1-Score | 32.43% | 96.37% |
| Average Response Time | 1.915s | 0.823s |

Table 4. Dual-Model Detection Result Statistics

| Statistical Indicators | Traditional Object Detection Model (YOLOv8) | Multimodal Large Model (Qwen3-Omni-Flash) |
|---|---|---|
| True Positive (TP) | 18 | 93 |
| False Positive (FP) | 0 | 0 |
| True Negative (TN) | 0 | 40 |
| False Negative (FN) | 75 | 7 |
| Miss Rate | 80.65% | 7.53% |
| False Alarm Rate | 0% | 0% |

### 2.5.3 Comparative experimental Conclusions

By comparing the performance of traditional object detection models and multimodal large models in fire detection tasks, this experiment verified the absolute advantages of large models in detection rate, generalization ability, and false alarm control. Relying on pre-training with massive data, the fire detection rate of multimodal large models (93.00%) is much higher than that of traditional models (19.35%).

Single-model deployment schemes all have limitations: the pure large model scheme relies on the network, and the pure traditional small model scheme has low detection rate and difficult false alarm control; although the hybrid deployment scheme is theoretically optimal, practical issues such as threshold control and false alarm transmission have not been fully resolved. Therefore, in the actual deployment of fire early warning systems, if the scene has stable network conditions, the cloud-based large model scheme is preferred to ensure detection accuracy and reliability; if local offline deployment is required, the performance of traditional models needs to be improved by expanding training data, optimizing model structure (such as lightweight large model distillation), or waiting for the improvement of edge computing hardware computing power to realize the local deployment of lightweight large models.

## 2.6 Multi-modal Workflow Optimization

In addition to the features mentioned above,voice interaction can also ensure that the model can correctly execute operations when there are many smart home devices, achieving more personalized functions through model fine-tuning, Prompt engineering, historical data introduction, and workflow provided by the large model cloud platform:

Intent Recognition and Device Control: By setting up an intent recognition module and a knowledge base module [16] in the workflow, introducing knowledge bases such as smart home device control registers, and identifying whether the user's voice contains device operation intentions (such as "turn on the air conditioner", "turn off the lights", etc.), if it exists, a feedback with an operation code is generated, thereby achieving rapid control of all devices without training. For example, the operation code is directly fed back, forwarded by the gateway to drive the infrared emission node, to achieve control of the air conditioner.

Intelligent Timing: According to the time information and habit information contained in the user's voice, task scheduling and timed reminder functions are implemented to ensure that home devices operate according to pre-designed plans.

Intelligent Temperature Control: Combining environmental temperature data and user habits, the set temperature of the air conditioner is adjusted in real time to avoid situations such as the user getting sick due to continuous cooling all night.

Visual Linkage: Using voice commands to call the image modality function can check whether the access control camera has captured information such as people staying outside the door, and can also combine images to ask questions to the large language model to achieve cross-modal information fusion.

User Habit Summarization: Introducing chat history data, regularly updating user habits, and adjusting device control logic accordingly to improve the accuracy of automatic operations.

The request string input to the workflow enters the intent recognition module to determine whether the user is chatting with the controller or wants to operate a device. If device operation is required, the knowledge base and user commands are read. If only chatting is required, the large model's generalization ability is directly used to generate a response. The optimized workflow is shown in Figure 13.

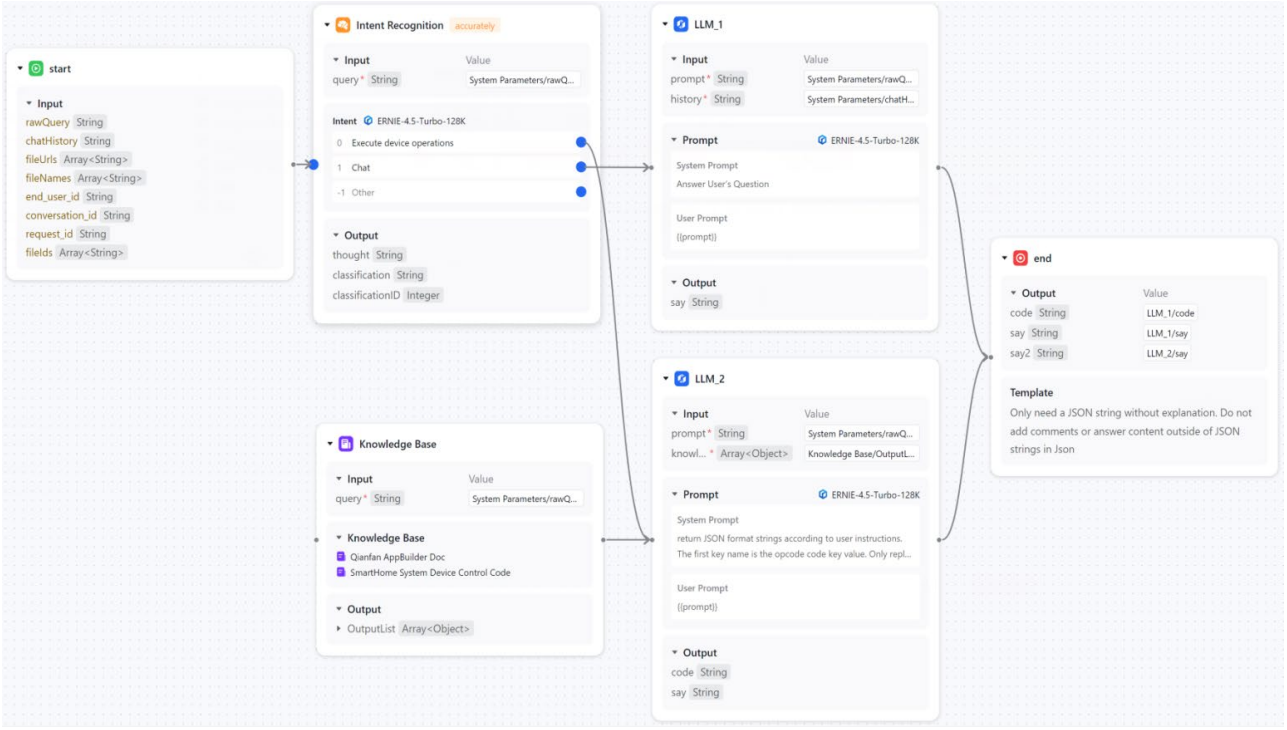



**Figure 13.** Optimized Workflow

The Baidu Qianfan large model platform supports fine-tuning pre-trained foundation models to enable the model to learn how to process actual datasets, but this approach requires providing a large amount of data, and the training effect depends on the data quality. Therefore, in smart home voice control and chat applications, users only need to upload a list of smart home device control registers to generate a vector knowledge base for the word embedding process, which can then be used for Retrieval-Augmented Generation to obtain more accurate operation codes and content feedback.

## 2.7 Sensor Layer Design and Data Acquisition

To facilitate the description of the hardware connection relationships and functional linkage logic among various sensors, controllers and actuators in the system, this section summarizes and explains the complete workflow of the "sensor–data link–actuator" chain in the multifunctional bathroom control system, whose network architecture diagram is shown in Figure 14. All sensor data is collected by ESP-01S/ESP8266 Mesh sub-nodes and transmitted to the Mesh gateway and main control chip via the Mesh network, whose internal structure is shown in Figure 15.

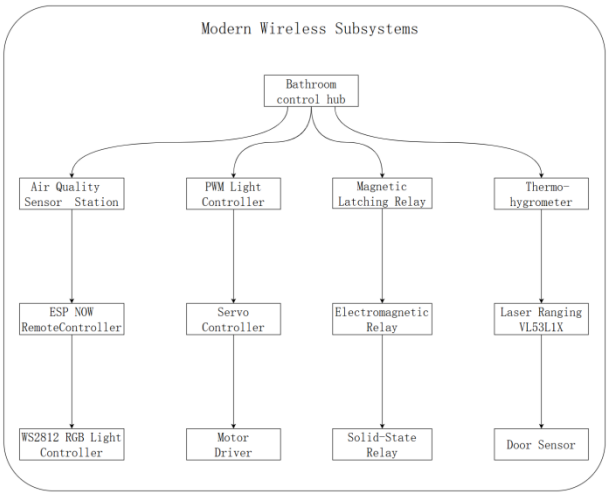



**Figure 14.** Network Architecture Diagram of the Bathroom Control System

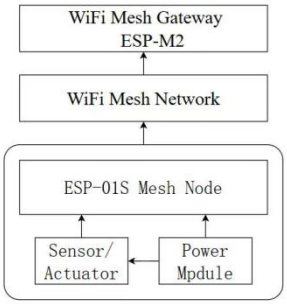



**Figure 15.** Internal Structure Diagram of the Bathroom Control System

Sensor Acquisition Layer: Diverse sensors gather environmental data through I²C, 1-Wire, UART, or GPIO interfaces. For instance, the VL53L1X laser ranging sensor performs distance measurement, the SHT-series temperature and humidity sensor monitors bathroom ambient temperature and humidity, and the BH1750 ambient light sensor detects light intensity—all utilizing the I²C protocol. The DHT11 temperature/humidity sensor and DS18B20 water temperature sensor adopt the 1-Wire communication protocol, while the NDIR-based $CO_2$ sensor uses the UART protocol. Additionally, MQ-series gas sensors and micro-switch door sensors detect smoke, gas conditions, and door open/closed status via GPIO on/off signals. The physical diagram is shown in Figure 16.

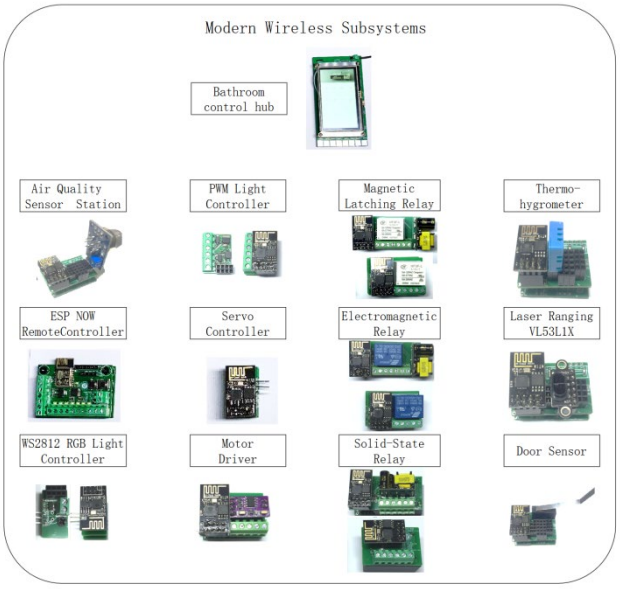



**Figure 16.** Physical Diagram of the Bathroom Control System

The bathroom control system includes five typical function trigger logics: First, the automatic flushing function collects distance data via the VL53L1X laser ranging sensor (I²C). When the distance is less than the threshold of approximately 30 cm (indicating a person is present) and the unoccupied state lasts for more than 30 seconds, the electromagnetic relay drives the solenoid valve, and the flushing duration is dynamically calculated based on the occupancy time to avoid false flushing and water waste. Second, the automatic lighting control relies on the BH1750 ambient light sensor (I²C) to collect illumination data, combined with distance measurement or door status to assist in judging human presence. When the light intensity is below the threshold and a person is detected, the latching relay or isolated PWM automatically turns on the light and adjusts the brightness, and the light is turned off immediately or with a delay after the person leaves. Third, the air quality linked ventilation collects data through the UART-based $CO_2$ sensor and GPIO-based MQ series gas sensors. When the $CO_2$ concentration exceeds the limit or a smoke/abnormal odor alarm is triggered, the solid-state relay or fan PWM controller starts ventilation and adjusts the fan speed according to the pollution level. Fourth, the door status auxiliary calibration logic collects door status via the GPIO-based door sensor. If the door is closed for a long time but the human body sensor is continuously triggered, the sensor is judged to be abnormal, and recalibration or shielding is performed to improve system reliability. Fifth, the user active control function allows users to operate via the remote control node (buttons, knobs, OLED). Buttons transmit instructions through GPIO, knobs through ADC, and OLED through I²C. The instructions are sent to actuators such as curtain motors, lights, fans, and relays via the Mesh network to achieve real-time control and host status synchronization.

Through the design of unified access for multi-interface sensors, interconnection via the Mesh network, centralized decision-making by the host, and device-level linkage control, the system constructs a complete architecture of "sensors perceiving the environment, Mesh network transmitting data, main control chip making logical judgments, and actuators executing actions". Each module has a clear division of labor and a well-defined structure, and the system also features good scalability and reliability, which can stably realize core functions such as environmental monitoring, intelligent linkage, and user active control.

## 3. Mesh Gateway and IoT Software Design

### 3.1 Mesh Gateway Software Design and Debugging

The Mesh gateway is used for cloud APP data forwarding and local interconnection coordination and scheduling of devices in wired and wireless control domains. Whether it

is the distribution of multi-modal perception results or the execution of instructions from the wired and wireless perception control domains, all wireless data transmission is carried out through Wi-Fi Mesh. The gateway software design mainly adopts the PainlessMesh library and self-built application layer protocol code to realize the following functions: 1) Multi-threaded task management: Utilizing the Real Time Operating System(RTOS) task management mechanism of ESP8285 ensures that data requests from multiple sub-devices can be processed simultaneously; 2) Self-healing mechanism and dynamic routing: When node failures, data congestion, or communication interruptions occur in the network, sub-nodes automatically reconstruct routes to ensure the continuity and stability of data transmission; 3) Data encryption and protocol management: A self-built high-performance communication protocol is used, which supports data encryption and integrity verification to ensure system security and reliability.

The design of the application layer communication protocol adopts JSON format, which has the characteristics of fast parsing speed and high readability. Example data is as follows:

{"data":{"get":"state","sender":{"type":"MeshRelay","id":1},"receiver":{"type":"CentralController","id":1},"time": 1743123489,"uuid":1}}

The protocol includes operation instructions, sender and receiver device identification information, UNIX timestamp, and operation user account, which can prevent data replay attacks and ensure communication security.

When a node joins or leaves the network, the system can automatically complete identity authentication and resource allocation, and adjust the routing according to the real-time status of the network to ensure the reliability of data transmission. The routing topology presents a star shape to prevent broadcast storms during data propagation, supports unicast and multicast. At the same time, due to its actual mesh topology structure, if any node fails, other nodes can quickly communicate and achieve self-healing.

The Mesh gateway can connect to a variety of devices, such as relays, PWM controllers, sensors, remote controls, etc. The data type of the received sub-node data is floating-point. The number of data points can be increased or decreased through macro definitions, and it can be parsed into other formats using the conversion functions provided by the JSON library. After the device is online, the Mesh gateway will upload the data list to the APP and print it on the serial port for easy debugging. As shown in Figure 17, the sensor attached to the relay node uploaded five types of data such as temperature and humidity, and the downlink status is all 0, indicating that the relay is in the off state.

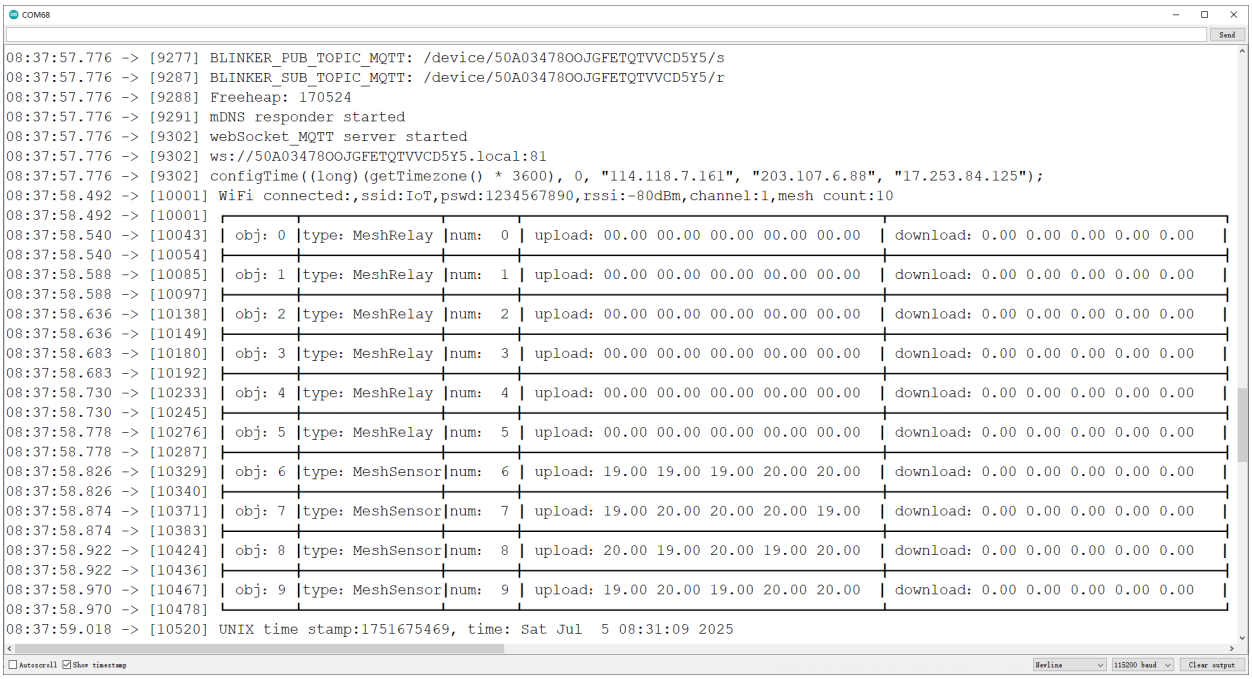



**Figure 17.** Sub-node Data Received by the Mesh Gateway

As an important bridge for the interconnection of various terminal devices in the system, the debugging of the Mesh network mainly includes: realizing multi-threaded task management through the Painless Mesh framework, and focusing on verifying the self-healing mechanism of data acquisition, forwarding, and routing reconstruction of each node during debugging [17]. Experiments show that when some nodes fail or the connection is interrupted, the system can automatically identify and recalculate the optimal routing path to ensure

uninterrupted data transmission. The debugging of Mesh network nodes is shown in Figure 18. Whether the devices are densely packed or distributed in various locations, they can send and receive information and execute instructions. The system is most stable when the node distance is less than 20 meters, which meets the usage needs of smart home scenarios.

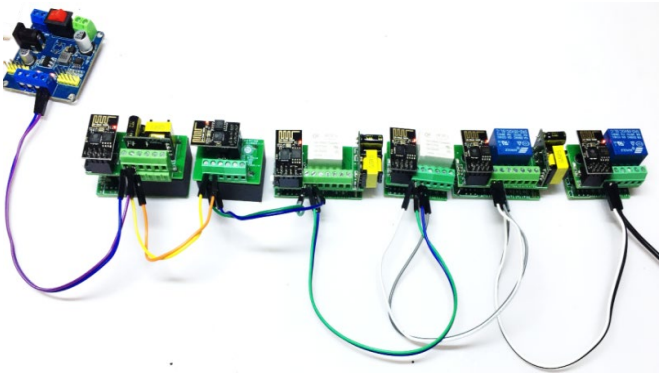
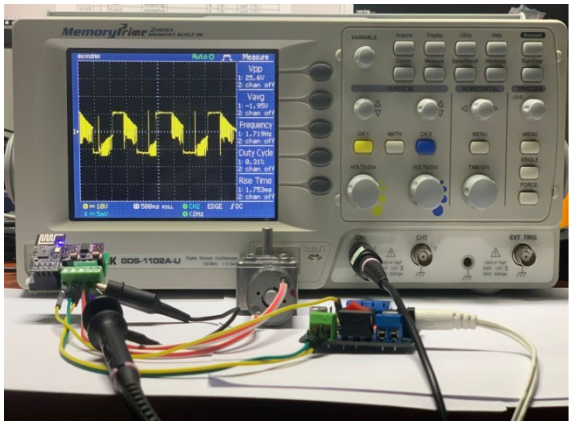
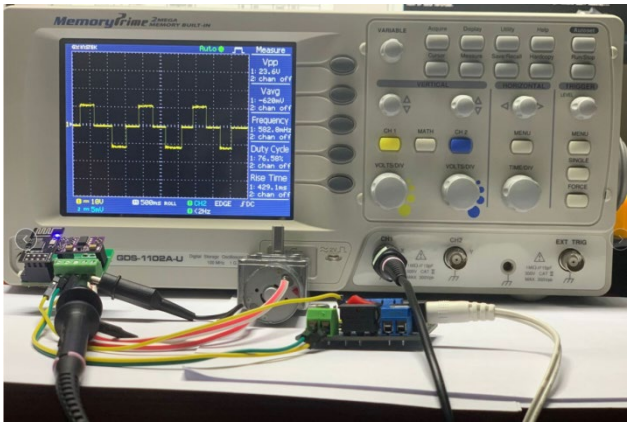


**Figure 18.** Mesh Network Node Debugging

To ensure the security of data during network transmission, the system adopts a self-built high-performance application layer communication protocol and has performed data encryption and integrity verification debugging. The test results show that under normal working conditions, the data transmission delay and packet loss rate are within a controllable range, and the device executes instructions immediately after receiving data. The hardware circuit design has been practically verified. As shown in Figure 19(a), when the DC motor forward and reverse rotation node in the Mesh network does not use braking, the ringing phenomenon is obvious. After using braking, the ringing caused by the back electromotive force of the motor is effectively suppressed, as shown in Figure 19(b), preventing voltage shock damage to the circuit.



(a) Waveform test without braking



(b) Waveform test with braking

**Figure 19.** Mesh Network Node Output Waveform Test

## 3.2 IoT Software Design

The system adopts the Blinker IoT open-source solution for remote control. This solution is provided by the blinker Technology operation and maintenance server and offers an APP and device-side SDK. Users can personalize the mobile APP interface and the device side. Users only need to develop the device-side program and configure the mobile APP interface components to complete the deployment. After deployment, users can initiate operations or schedule tasks to the controller through the Blinker APP mobile client, or change the controller's automatic hosting scheme, thereby realizing the controller's access to the cloud.

The blinker Message Queuing Telemetry Transport(MQTT) server (Blinker Broker) is deployed using the SaaS model [18], serving a large number of users and supporting device sharing. Therefore, each device and mobile APP account is assigned two topics: one for sending and one for receiving. The server is responsible for forwarding between device topic groups and mobile phone topic groups. Each device has independent sending and receiving topics. The advantage is that received messages will not be overwritten by the next sent message, avoiding the problem of unreceived messages being overwritten by newly published messages after a single topic is published. In addition, when a device has sharers, it only needs to include the sharer's Universally Unique Identifier(UUID) in the message to forward the message to other user APPs, achieving convenient one-to-many data transmission.

The blinker APP is built based on Ionic4, Angular8, and Cordova8. The interface uses the drag-and-drop layout function in Angular CDK to enable users to quickly perform secondary development. Its online process is the same as the device side. Each time it is opened, it needs to log in to MQTT after HTTP authentication [19-21]. When a user enters the device interface, the APP first sends {"get":"state"}. If the device side receives

{"state":"online"}, it determines that the device is online. After pressing buttons and other components, it automatically sends {"button key name":"key value"}, for example, the key value of a button component can be on/off (switch mode) or tap (button mode). In either mode, a long press can trigger press/pressup. The required function can be achieved by simply adding the processing method for the key value in the device-side user code. The IoT software architecture is shown in Figure 20.

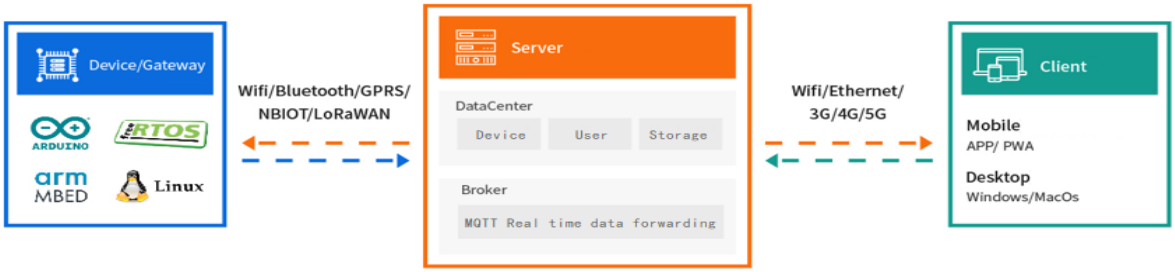



**Figure 20.** IoT Software Architecture

When the device is powered on, the boot program is executed first, and then the user's main program is started. In the main program initialization, the Blinker initialization function Blinker.Begin() creates component instances containing data key names, configures network parameters, starts Wi-Fi connection, and ends initialization. In the main program loop, Blinker.run() starts to execute cyclically, detects Wi-Fi configuration or NBIoT module SIM card information, and then connects to the Wi-Fi or 4G base station. After a successful connection, it accesses the blinker HTTP server for two-step authentication:

The first authentication is login authentication. The device inputs the authorization code parameter through the domain name, and the HTTP server returns the MQTT parameters for logging in to the Broker (the MQTT server is called Broker), including:

Device name "deviceName" and type "productKey"

Dynamic account "iotId" and dynamic password "iotToken" for logging in to the server (changes with each authentication)

Broker name, Broker domain name "host", and port "port"

Creator user ID "uuid" (mobile APP)

After that, it obtains the Network Time Protocol(NTP) network time and performs secondary authentication to confirm whether other users share this device. Then, it disconnects from the authentication HTTP server and logs in to the MQTT IoT server according to the obtained authentication information to execute the user IoT program. When the APP sends a message containing the data key name of a component, the message is placed in the callback function of the corresponding component, such as the on and off commands of a switch, to realize the control of the device side. The block diagram of the device online process is shown in Figure 21.

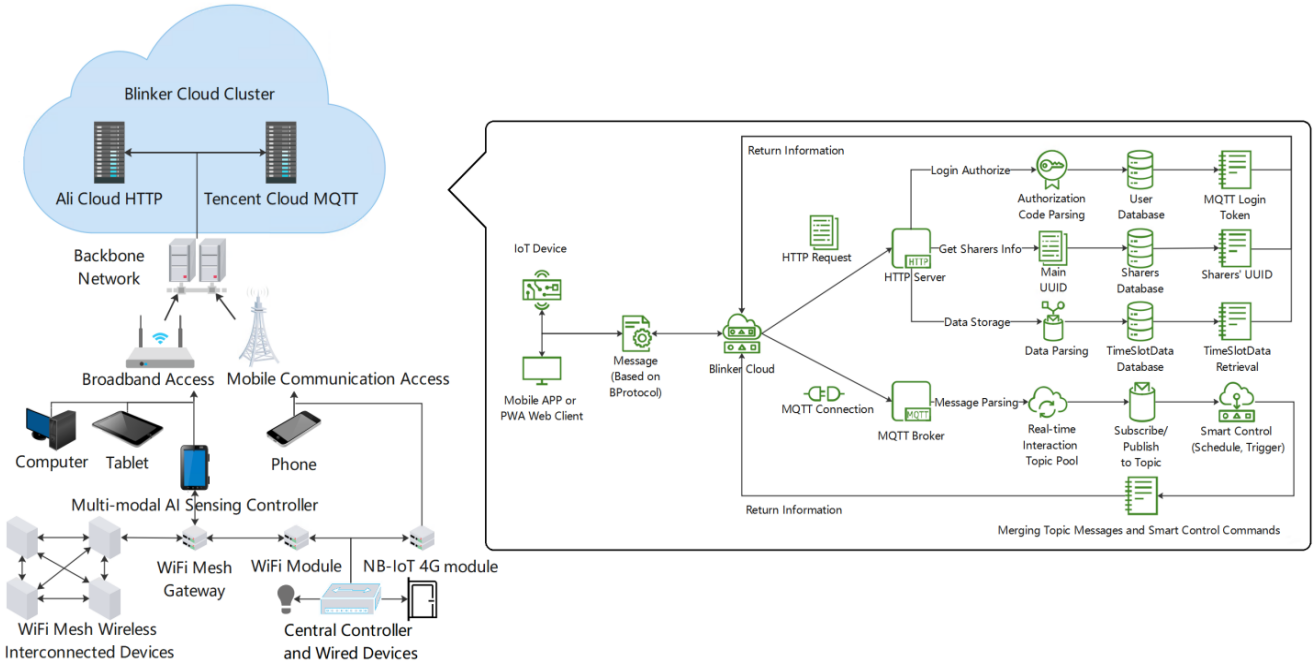



**Figure 21.** Device Online Process Block Diagram

## 4. Discussion and Limitations Analysis

This multimodal system leverages the principles of "prompt engineering" and "workflows" to achieve considerable scalability. A key advantage is that expanding its intelligent perception capabilities frequently requires only prompt adjustments, eliminating the computational expense of model retraining. The system's extensibility is further supported by a RAG-based knowledge base, which facilitates the addition of new devices. For widespread commercial adoption, however, critical impediments such as cost, power consumption, network dependency, and the paramount issues of user privacy and data security must be thoroughly addressed. Paramount among these concerns are the user privacy and data security challenges inherent to extensible systems, a well-documented issue highlighted by systematic research in the field[22].

## 5. Conclusion

The system has realized multiple functions such as voice interaction, visual monitoring, and Mesh self-organizing network for smart home perception systems. It uses the IoT embedded controller ESP32-S3 as the core perception unit and accesses Baidu Cloud Voice, Baidu Qianfan Large Model Platform, and Alibaba Cloud Bailian Large Model Platform to achieve multi-modal interactive functions such as speech recognition, smart home linkage, and visual intelligent environmental risk early warning. It has high real-time performance and uses the Mesh network to achieve efficient collaboration between wireless devices, ensuring the stability of data transmission and the flexibility of the system. The system utilizes the workflow orchestration function provided by the large model platform to optimize processing flows, encodes smart home device information into a knowledge base to improve operation execution accuracy, uses a front-end model to perform intent discrimination to distinguish between chat questions and operation commands, preventing model overfitting and ignoring chat content, and introduces dialogue history to achieve user habit summarization. This optimization method reduces the network transmission pressure on edge devices and improves the personalization of the system.

The author's work and innovations mainly include the following two points:

(1)Cross-scene generalization innovation of multi-modal information perception: This research introduces a multi-modal AI large model with a huge number of parameters to uniformly perform word embedding on various modal data such as vision, voice, and sensors, and encodes them into a high-dimensional vector space. Using this innovative multi-modal model invocation framework, multi-modal data fusion is driven by prompts, and cross-modal reasoning is performed using the pre-trained large language model platform, thereby achieving high-precision recognition of ambiguous semantics and unstructured information. This method effectively avoids the shortcomings of traditional single-modal perception that requires fine-tuning models one by one, and significantly reduces the video memory occupation and computational burden of edge devices. Furthermore, through vector retrieval based on the device-specific knowledge base, the optimization of the multi-modal perception workflow is realized, greatly improving the processing efficiency and accuracy of the smart home system for ambiguous semantic instructions.

(2)Workflow-based intelligent home control system architecture innovation: In terms of the smart home control system architecture, a smart home system prototype that deeply integrates technologies such as large model platforms, IoT cloud platforms, and Mesh self-organizing networks has been developed. Through workflow design, it automatically distinguishes between operation commands and ordinary chat content, and effectively manages the context during model reasoning, solving the problem of model hallucinations. In addition, the system architecture proposed in this paper adopts the inference mode after large model pre-training, combined with networking capabilities and retrieval-augmented generation technology, to achieve effective understanding and control of ambiguous and complex scene semantics, solving the problem of limited resources of terminal devices and providing a new and easily scalable technical route for the future smart home field.

## References


[1] D.N. Mekuria, P. Sernani, N. Falcionelli, A.F. Dragoni, Smart home reasoning systems: a systematic literature review, J. Ambient. Intell. Humaniz.Comput.12(4)(2021)4485–4502. http://dx.doi.org/10.1007/s12652-019-01572-z, URL https://link.springer.com/10.1007/s12652-019-01572-z.

[2] A. Mari, A. Mandelli, R. Algesheimer, Empathic voice assistants: Enhancing consumer responses in voice commerce, J. Bus. Res. 175 (2024) 114566, http://dx.doi.org/10.1016/j.jbusres.2024.114566, URL DOI: 10.1016/j.jbusres.2024.114566.

[3] F. Van Harmelen, A. Ten Teije, A boxology of design patterns for hybrid learning and reasoning systems, Journal of Web Engineering 18(1–3)(2019)97–123,URL https://doi.org/10.13052/jwe1540-9589.18133.

[4] M. Van Bekkum, M. De Boer, F. Van Harmelen, A. Meyer-Vitali, A.T. Teije, Modular design patterns for hybrid learning and reasoning systems: a taxonomy, patterns and use cases, Appl. Intell. 51 (9) (2021) 6528–6546, http://dx.doi.org/10.1007/s10489-021-02394-3,URL https://link.springer.com/10.1007/s10489- 021-02394-3.

[5] Seinfeld, S.; Feuchtner, T.; Maselli, A.; Müller, J. User representations in human-computer interaction. Hum.–Comput. Interact. 2021, 36, 400–438. DOI:10.1080/07370024.2020.1724790.

[6] Wang, H. Landscape design of coastal area based on virtual reality technology and intelligent algorithm. J. Intell. Fuzzy Syst. 2019, 37, 5955–5963. DOI:10.3233/JIFS-179177.

[7] Wangsa, K.; Karim, S.; Gide, E.; Elkhodr, M. A Systematic Review and Comprehensive Analysis of Pioneering AI Chatbot Models from Education to Healthcare: ChatGPT, Bard, Llama, Ernie and Grok. Future Internet 2024, 16, 219. https://doi.org/10.3390/fi16070219.

[8] ]Kush, J.C. Integrating Sensor Technologies with Conversational AI: Enhancing Context-Sensitive Interaction Through Real-Time Data Fusion. Sensors 2025, 25, 249. https://doi.org/10.3390/s25010249.

[9] Liu G, Bono CA, Pierri F. 2025. Comparing diversity, negativity, and stereotypes in Chinese-language AI technologies: an investigation of Baidu, Ernie and Qwen. PeerJ Computer Science 11:e2694 https://doi.org/10.7717/peerj-cs.2694.

[10] J. Oh et al., "An Energy-Efficient High Resolution Vision Transformer Processor Exploiting Token Similarity Beyond Token Merging," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 34, no. 1, pp. 118-129, Jan. 2026, doi: 10.1109/TVLSI.2025.3604745.

[11] Syed Mushhad Mustuzhar Gilani, Muhammad Usman, Saqib Daud, Asif Kabir, Qamar Nawaz, Oláh Judit.SDN-based multi-level framework for smart home services.Multimedia Tools and Applications . (2024) 83:327–347.https://doi.org/10.1007/s11042-023-15678-2.

[12] G. Yin, Y. Liu, T. Liu, H. Zhang, F. Fang, C. Tang, L. Jian g. Token-disentangling mutual transformer for multimodal emotion recognition. Eng. Appl. Artif. Intell., 133 (2024), https://doi.org/10.1016/j.engappai.2024.108348.

[13] Guo S, Wang Q. Application of Knowledge Distillation Based on Transfer Learning of ERNIE Model in Intelligent Dialogue Intention Recognition. Sensors. 2022; 22(3):1270. https://doi.org/10.3390/s22031270.

[14] Zhang, JX., Tao, CQ., Huang, ZQ. et al. Discovering API Directives from API Specifications with Text Classification. J. Comput. Sci. Technol. 36, 922–943 (2021). https://doi.org/10.1007/s11390-021-0235-1.

[15] Pang Z, Wang Q, Wang Y, Gong Z. A Novel Intelligent Rebound Hammer System Based on Internet of Things. Micromachines. 2023; 14(1):148. https://doi.org/10.3390/mi14010148.

[16] Li, Z., Peng, J., Lin, X. et al. Multimodal intent recognition based on text-guided cross-modal attention. Appl Intell 55, 690 (2025). https://doi.org/10.1007/s10489-025-06583-2.

[17] K Peng., Y Hu., H Ding.et al., "Large-Scale Service Mesh Orchestration With Probabilistic Routing in Cloud Data Centers," in IEEE Transactions on Services Computing, vol. 18, no. 2, pp. 868-882, March-April 2025, doi: 10.1109/TSC.2025.3526373.

[18] Zhu, J., Li, Q., Ying, S. et al. Research on Parallel Task Scheduling Algorithm of SaaS Platform Based on Dynamic Adaptive Particle Swarm Optimization in Cloud Service Environment. Int J Comput Intell Syst 17, 260 (2024). https://doi.org/10.1007/s44196-024-00666-7.

[19] Roldán-Gómez, J., Carrillo-Mondéjar, J., Castelo Gómez, J.M., Ruiz-Villafranca, S.: Security analysis of the mqtt-sn protocol for the internet of things. Appl. Sci. 12(21), 10991 (2022).

[20] Stangaciu, V., Stangaciu, C., Gusita, B. et al. Integrating Real-Time Wireless Sensor Networks into IoT Using MQTT-SN. J Netw Syst Manage 33, 37 (2025). https://doi.org/10.1007/s10922-025-09916-1.

[21] Roldán-Gómez, J., Carrillo-Mondéjar, J., Gómez, J.M.C., Martínez, J.L.M.: Security assessment of the mqtt-sn protocol for the internet of things. J. Phys: Conf. Ser. 2224(1), 012079 (2022). https://doi.org/10.1088/1742-6596/2224/1/012079.

[22] Thingnes, T. R., & Meland, P. H. (2025). Security Challenges for Users of Extensible Smart Home Hubs: A Systematic Literature Review. Future Internet, 17(6). https://doi.org/10.3390/fi17060238.