Design of a Voice Interaction Module Based on NanoPI NEO2

1st Chunpeng Wang
College of Art and Design
Beijing Institute of Fashion Technology
BeiJing, China
paladinstudio@126.com

2nd Kaixiang Wang*
College of Art and Design
Beijing Institute of Fashion Technology
BeiJing, China
wkx05@163.com

***Abstract*—With the deep integration of Internet of Things (IoT) and Artificial Intelligence (AI) technologies, voice interaction, as a natural and efficient mode of human-computer interaction, has widely penetrated various fields of social life and work. This paper mainly discusses the design process of a lightweight intelligent voice interaction module based on NanoPi NEO2. This module uses the NanoPi NEO2 development board as the core hardware platform, runs the Ubuntu operating system, and integrates four key technologies: voice wake-up, speech recognition, speech synthesis, and natural language processing. Through an API interface, it connects to large models, enabling a full-process interaction from voice input to command response. Based on the hardware architecture and software environment of the NanoPi NEO2 module, the performance advantages in voice interaction scenarios are analyzed. The overall system architecture, hardware interfaces, software functional modules, and large model integration process are described, and typical application cases are listed. By setting up a testing environment, tests are conducted from the perspectives of functional completeness, response latency, and stability, verifying the module's feasibility, demonstrating its stable application in smart home, portable intelligent assistant, and other scenarios, and providing a feasible solution for the wide application of voice interaction systems..**

Keywords—Voice interaction; Speech recognition and synthesis; Large model integration; Case Application

I. INTRODUCTION

With the deep integration of Internet of Things (IoT) and Artificial Intelligence (AI) technologies, voice interaction, as a natural and efficient mode of human-computer interaction, has widely penetrated multiple fields such as smart homes, intelligent terminals, and robotics. This design is based on the NanoPi NEO2 development module and follows an architecture approach of 'core module coordinating multiple components,' enabling multifunctional coverage, including voice wake-up, custom commands, IoT communication, sensor data collection, and integration with various large models. This process requires close alignment with the module's computational power characteristics and the adaptability of voice algorithms. The main program is developed in Python and runs on the Ubuntu system, completing the standardized design and development of both hardware and software.

Currently, voice interaction technology has formed a technical system centered around Speech Recognition (ASR), Natural Language Processing (NLP), and Text-to-Speech (TTS). Existing research on multi-scenario applications provides diversified references for the system design based on NanoPi NEO2: The online language translation interactive learning system designed by Deng Lijun [1] enhances the recognition accuracy in complex environments through noise reduction algorithms. This approach can be transferred to the NanoPi NEO2 scenario to compensate for accuracy limitations under low computational power and complement audio codec optimization. The 'dedicated voice module + core control unit' scheme used by Zhang Xuru et al. [2] for the AI voice interactive intelligent seat controller enriches the adaptation strategy of the voice module with NanoPi NEO2, offering diverse references for multi-scenario application research. In the robotics field, the voice control system for quadruped robots by Chen Teng et al. [3] and the voice system for reception robots based on ROS by Liao Chunlan [4] respectively verify the feasibility of voice interaction in mobile devices and software architecture design logic. In the smart home domain, the IoT-based home voice system by Li Hengbei et al. [5] and the smart lighting voice control system by Qiu Haiyan [6] establish a 'perception-processing-execution' feedback loop and scenarized adaptation scheme. The multi-modal interaction concept for intelligent terminals proposed by Kan Baoqiang et al. [7] provides new ideas for the hybrid design integrating voice, sensing, and large models in this system. Regarding the three core issues commonly faced by embedded voice interaction systems—computational power adaptation, low-power optimization, and multi-module collaboration—Guo Hongzhen et al.'s [8] distributed adaptive control algorithm can be applied to balance the computational power requirements and control accuracy of NanoPi NEO2, and achieve algorithm lightweighting through Opus codec optimization and voice model quantization. The low-power sleep strategy by Cao Xing et al. [9] can optimize the module's battery life and response balance by considering the intermittent nature of voice interaction. The monitoring terminal based on MQTT protocol by Zou Yingjie [10] provides technical support for multi-module communication adaptation.

II. PREPARATION FOR SOFTWARE AND HARDWARE DESIGN

The NanoPi NEO2 is an ultra-small ARM-based computer development module launched by FriendlyARM, featuring an Allwinner H5 processor. It is a compact embedded device with high performance, low power consumption, rich interfaces, and strong compatibility, making it naturally adaptable for embedded voice interaction system development. The hardware driving capabilities and data processing performance demonstrated by this series of modules in industrial control scenarios have been widely validated, accumulating integration experience for the incorporation of voice interaction modules. Therefore, existing technologies have laid a solid foundation for the fusion of voice interaction and terminal devices.

Correspondingauthor: KaixiangWang, School of Art and Design Beijing Institute of Fashion Technology,No.2EastYinghuaRoad, NorthEndofHepingStreet, ChaoyangDistrict,100029Beijing, China;Email:wkx05@163.com.

At the hardware level, the ARM Cortex-A53 architecture processor equipped in the NanoPi NEO2 is suitable for lightweight embedded applications. It can efficiently connect to various sensors through multiple interfaces for information acquisition, enabling voice signal capture and command transmission. Meanwhile, the module supports MQTT IoT communication protocol, allowing for remote control and data exchange with external software and hardware.

At the software and algorithm level, the voice wake-up function is implemented using the offline voice recognition engine 'pocketsphinx,' supporting custom wake-up word configurations. Speech recognition uses Alibaba's 'funasr' online tool, and speech synthesis is combined with the same platform's 'Tongyi TTS' tool, forming a complete voice interaction chain. Additionally, the module supports API integration with Alibaba's large models, including 'ASR' voice recognition, 'Tongyi Qianwen' and 'DeepSeek' text models, and Tongyi visual understanding image models. These models can be accessed via API for hardware control, creating a 'voice interaction - large model analysis - hardware execution' closed-loop system. Moreover, audio codec technology optimization ensures low latency and high precision under the module's limited computational power.

The overall design approach is to use the NanoPi NEO2 module as the main control device, which communicates with various sensor modules and controlled devices through limited connections. It processes information by connecting to cloud devices and corresponding large model APIs via the network. The schematic of this design is shown in Figure 1.

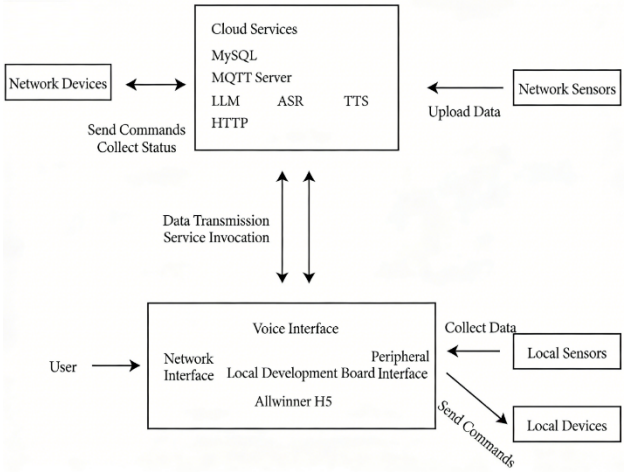

Fig.1. Framework diagram of voice interaction based on NanoPI NEO2 module

III. THE DESIGN PROCESS OF VOICE INTERACTION MODULE

A. Voice wake-up module

Successfully implementing the voice wake-up function is crucial for ensuring that the voice interaction system can begin operating. The core of this function is to detect a specific wake-up word, triggering the system to transition from standby to active mode. This requires balancing wake-up accuracy, anti-interference capability, and low power consumption. In this design, the Porcupine/PicoVoice offline wake-up engine is used, which can optimize custom wake-up words and utilize the TensorFlow Lite framework for lightweight deployment, thus achieving a match with the computational power requirements of the NanoPi NEO2 development module.

This engine is developed based on a Deep Neural Network (DNN) architecture, offering superior wake-up accuracy and anti-interference capabilities compared to traditional template-matching algorithms. It supports training custom wake-up words in multiple languages and allows the generation of model files for exclusive wake-up words through the PicoVoice console, thus reducing network latency and ensuring confidentiality. The implementation process is as follows: The wake-up word is customized through the PicoVoice console, and a model file suitable for embedded devices is trained and saved locally on the NanoPi NEO2 module. The microphone collects real-time audio data, which is preprocessed through sampling rate conversion and other steps before being input into the wake-up engine. The engine analyzes the audio data in real-time and, when the wake-up word's match exceeds a set threshold (90% in this design), it outputs a wake-up trigger signal to initiate the subsequent speech recognition process. If the match is below the threshold, no signal is triggered. The process is shown in Figure 2.

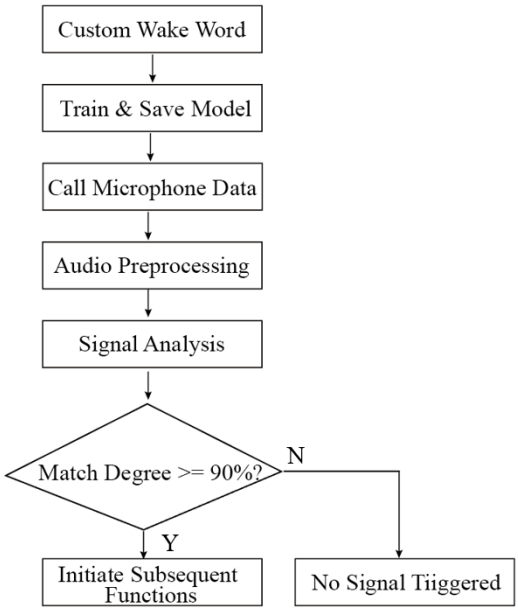

Fig.2. Voice wake-up function flowchart

Considering the improvement of the wake-up function's anti-interference capability and adaptability, and taking into account the computational power characteristics of the NanoPi NEO2, certain engine parameters can be optimized: For example, setting the audio sampling rate to 44.1kHz or 48kHz to enhance the quality of the wake-up voice. Matching the default input format of the speech engine and reducing unnecessary data format conversions to ensure data accuracy. Enabling the engine's built-in noise suppression module, which uses adaptive filtering algorithms to reduce environmental noise. Adjusting the length of the detection window appropriately to balance response speed and accuracy during the wake-up process, thereby effectively reducing false wake-ups and missed wake-ups, ensuring that the system can continuously and stably operate in various complex scenarios.

B. Implement a pre-processing module for speech signal recognition

The core of speech recognition technology is to convert the captured speech signals into text information, which is a key component of voice interaction. Considering the local computational power limitations of the NanoPi NEO2, this design adopts a hybrid approach of 'local preprocessing + cloud-based recognition.' The preprocessed data is sent to the cloud ASR service for high-precision recognition. The preprocessing process plays a role in performing spectral analysis, noise reduction, framing, and MFCC (Mel-Frequency Cepstral Coefficients) extraction on the captured speech signals, thereby reducing unnecessary data transmission and improving cloud recognition efficiency. The compensation formula for high-frequency attenuation of the speech signal through the high-pass filter is as follows:

$$y(n) = x(n) - \alpha x(n-1) \quad (1)$$

Where x(n) is the original speech signal, y(n) is the signal after pre-emphasis, and α is the pre-emphasis coefficient.

The compensated and processed speech signal is then divided into frames according to the predetermined frame length and time duration. A Hanning window is applied to each frame to modulate the signal, reducing spectral leakage. The Hanning window function is given by the formula:

$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right), 0 \le n \le N-1 \quad (2)$$

Where N is the frame length and nnn is the index of the sample point within the frame.

After framing, the time-domain signal undergoes Fast Fourier Transform (FFT) to convert it into the frequency domain, obtaining the frequency characteristics of the speech signal. The power spectrum is then weighted using a Mel filter bank to simulate auditory perception. The output formula of the filter is as follows:

$$M(m) = \sum_{k=f(m-1)}^{f(m+1)} P(k)H_m(k), 1 \le m \le 26 \quad (3)$$

This formula represents the Mel filter bank weighting, where M(m) is the weighted output of the m-th Mel filter, P(k) is the power spectrum at frequency index k, and Hm(k) is the Mel filter response. The summation is over the frequency indices k between the frequency range defined by the Mel filter corresponding to the mmm-th filter. f(m−1) and f(m+1) are the frequency boundaries of the m-th Mel filter.

Finally, the logarithm of the output from the Mel filter bank is taken, followed by a Discrete Cosine Transform (DCT), resulting in the corresponding MFCC (Mel-Frequency Cepstral Coefficients) values. The process is given by the following formula:

$$C(n) = \sum_{m=1}^{26} \log[M(m)]\cos\left(\frac{\pi n(m-0.5)}{26}\right), 1 \le n \le 12 \quad (4)$$

Where C(n) is the n-th MFCC value, and N is a positive integer, representing the upper limit for the number of selected coefficients.

After preprocessing, the speech signal data is sent over the network to the cloud-based ASR for recognition, and the recognized results are then sent back to the local system, completing the speech signal recognition process. The entire process flow is shown in Figure 3.

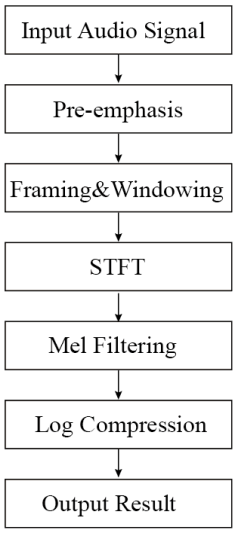

Fig.3. Flow chart of speech signal recognition structure

C. Text recognition adaptation and language synthesis/processing module

Considering the application needs of users in specific situations who recognize text and then restore it to speech, this feature uses TTS technology to recognize the text information processed by the system and synthesize it into natural and smooth speech information for output, thereby completing daily interactive needs such as reading manuscripts, books, and instruction manuals.

This module uses the external Alibaba Tongyi TTS large model to achieve this function. The large model provides a wide range of voice tone options, supports custom speech rate and intonation, and generates highly natural-sounding speech. By downloading and configuring the SDK, it can be embedded into the device. The process implemented on the NanoPi NEO2 module is as follows: After the system receives the input text, the Tongyi TTS SDK is used to call the cloud service for text-to-speech conversion, generating speech files in the specified format. The generated speech files are then played through the external speaker driver. To reduce playback latency, a preloading and caching method can be employed, where commonly used response texts are pre-generated and cached locally as speech files, thus reducing the number of cloud requests. The process flow is shown in Figure 4.

Additionally, the audio sampling rate (which can be set to 44.1kHz) and bitrate (which can be set to broadcast standard 96kbps or a lower 8-16kbps) can be optimized to balance speech quality and playback smoothness. The optimization process flow is shown in Figure 5.

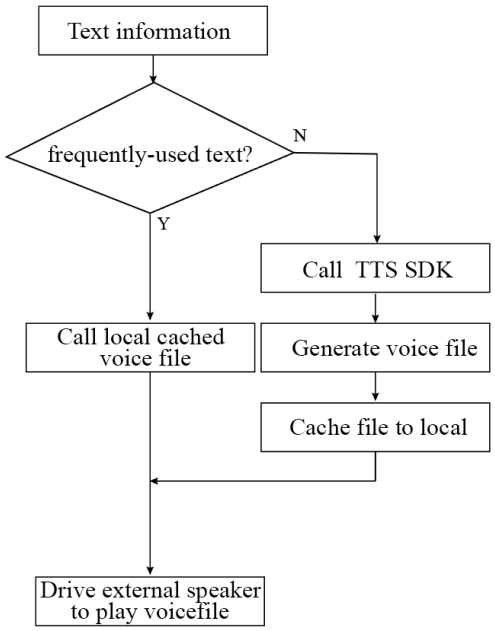

Fig.4. Flowchart of Text Recognition Adaptation, Language Synthesis, and Processing Modules

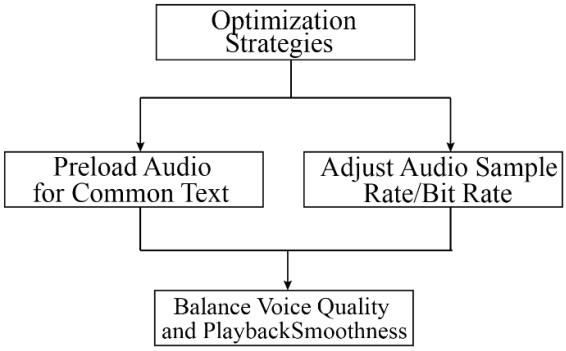

Fig.5. Flow chart of optimization process

D. Large model access module

Regarding the integration methods for large models, they can currently be divided into two approaches: local deployment and cloud-based invocation. The local deployment mode refers to deploying the large model on local servers or terminal devices, enabling local data processing and full autonomous control. The core advantages of this mode are that data security is controllable, sensitive data does not need to be leaked, and it has low latency, with response times that can be controlled within a very short period, meeting the needs of real-time interaction. However, local deployment requires high-performance hardware support, has a higher technical threshold, and faces challenges in terms of model updates, load monitoring, and scalability.

The method of calling the model through an API in the cloud is currently the most widely used approach for integrating large models. Its core logic is to call cloud-based model resources on demand through an API interface provided by the model service provider, without the need to focus on the underlying hardware or model maintenance. The advantages of this mode include zero initial deployment costs, strong performance scalability, and the elimination of the need for local device maintenance. Users can quickly integrate the model using an API key.

Taking the DeepSeek API integration as an example, developers can implement text generation functionality by writing a simple Python program. The process can essentially be broken down into two steps: First, identity authentication is completed through the 'Authorization' field in the request header, followed by passing the prompt (instructional text or multimodal instructions) and parameters to get the response from the large model.

Data exchange uses the HTTPS/HTTP protocol. The speech recognition request sends the preprocessed speech data in the specified format as a binary stream to the API interface, and the cloud returns a JSON response containing the recognized text, confidence level, and other information. The recognized text is then parsed through intent analysis and sent as parameters, with the cloud returning data such as intent type, control commands, and response text. The speech synthesis module sends the text, and the cloud returns speech data in the specified format.The response processing module parses the JSON data, extracts the key information, and converts it into a format that the system can recognize.

Additionally, an exception handling mechanism needs to be added. When an API call request fails, a preset prompt message is returned to ensure system stability. For example, a timeout duration can be set for the request. If the data transmission is not completed within the set time, the system will either retry or stop responding, in order to handle issues such as data loss caused by network fluctuations. The process of data exchange when integrating the large model is shown in Figure 6.

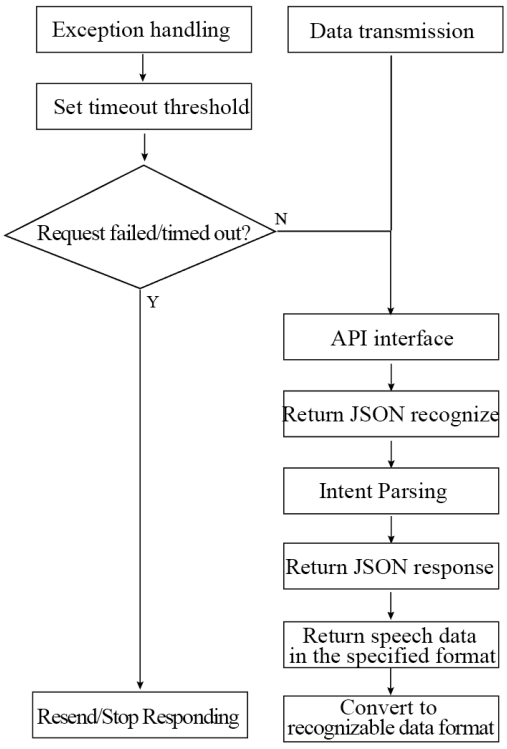

Fig.6. Process flowchart of data exchange for accessing large models

Currently, the technical implementation of integrating mainstream large models needs to address three key issues:

1)Interface compatibility: Most mainstream large models provide RESTful APIs or WebSocket interfaces that support JSON format for data exchange. Some models also offer SDK toolkits to simplify the development process. For example, DeepSeek's Python SDK allows quick implementation of code generation and logical inference function calls.

2)Parameter configuration capability: By adjusting relevant parameters, it is possible to control the randomness, length, and relevance of the generated content, adapting to different scene requirements.

3)Data format adaptation: When integrating multimodal models, it is necessary to support the encoding and transmission of non-text data such as images and audio. Models like GPT-4o and Wenxin Yiyan have already implemented end-to-end processing of multi-format data, which further lowers the barriers to integrating multimodal applications.

E. Functional Testing

Tests have shown that the system can accurately implement voice wake-up, speech recognition, intent analysis, speech synthesis, and peripheral control functions. It can respond to common commands such as 'wake up system,' 'stop operation,' 'turn on/off the light,' 'check the weather,' and 'set alarm,' and can synthesize reminder speech like 'distance too far' and 'posture deviation too large,' with no missing core functionalities.

The voice wake-up accuracy is 96.2% in a quiet environment, 92.3% in a normal noise environment, and 70.5% in a strong noise environment, indicating that the system has high wake-up reliability in daily environments, with performance slightly decreasing in strong noise environments. The average response delay is 1.3 seconds, of which local preprocessing takes 0.2 seconds, cloud requests take 0.9 seconds, and local playback takes 0.2 seconds. The delay is within an acceptable range with no noticeable lag. The specific test results are shown in Table 1.

TABLE I. Identify accuracy test data

Test metrics	⩽40dB	⩽70dB	⩽80dB
Awakening accuracy	96.2%	92.3%	70.5%
recognition accuracy	97.6%	91.1%	68.7%
Response delay (S)	1.2	1.4	1.3

IV. CASE STUDY OF PERIPHERAL DEVICE CONTROL APPLICATION

The voice interaction module can communicate extensively with other peripheral devices through the 'MQTT/TCP' protocol via the IoT network, enabling data exchange and control operations. A typical application scenario is voice control of various household items such as lights, curtains, humidifiers, and televisions in a home environment. It can also serve as a portable intelligent assistant in various aspects of life and work, by integrating functions like information inquiry, schedule management, and file operations, adapting to office, business, and other scenarios.

Users can query weather, news, or calculator results, set schedule reminders, and give commands like 'open file' or 'delete file' through voice. The system connects to cloud-based large models to obtain information, generate responses, and provide voice feedback.

Additionally, in industrial automation production scenarios, the system can be deployed in production workshops. By connecting industrial sensors and controllers, it can enable functions like equipment status queries and issuing simple control commands. After the workers wake up the system with voice commands, they can issue commands such as 'check assembly line speed,' 'pause equipment operation,' or 'restart control system.' The system analyzes these commands, communicates with the industrial controllers to obtain equipment status information or execute control instructions, and provides voice feedback with the corresponding response. This process effectively reduces the frequency of manual operations, significantly reduces operational risks, and improves production efficiency. Below are two specific application cases.

A. Case application of distance reminder

In daily life, the issue of children or elderly people getting lost is a very serious problem. This module can implement a voice alarm function by connecting to Bluetooth devices to obtain distance data and trigger the alarm based on preset values.

The voice module connects to the Bluetooth devices on both the guardian and the person being monitored. After obtaining the distance detection data from the Bluetooth devices, if the distance exceeds the safety threshold, the voice module will trigger a voice alarm. The distance between the two constantly changes as their positions shift. This is achieved through two Bluetooth modules, one as the master and the other as the slave, which form a data transmission relationship after pairing. The principle is to determine the distance based on the variation in the strength of the Bluetooth signal, specifically through the change in RSSI (Received Signal Strength Indicator) values. The relationship between Bluetooth signal strength and distance can be written as:

$$RSSI=A - 10n\cdot lg(d) \quad (5)$$

In the formula, A is the RSSI value at the receiver when the distance between the transmission nodes is 1 meter, n is the environmental attenuation factor, which represents the rate at which distance affects signal attenuation, and ddd is the distance between the nodes.

This step only requires determining if the distance has changed. Then, the duration of the change is used to decide whether to activate the voice alarm function. In the RSSI data detection experiment, the distance change was set within 10 meters, and 8 sets of data were tested, with each set tested 5 times. After removing the data with significant fluctuations, the average value was taken as the RSSI value. The results are shown in Table 2.

TABLE II. The relationship between RSSI value and distance testing

Distance interval (m)	RSSI(dBm)	Error(dBm)	ranging error (m)	Distance sensitivity
1 - 3	-45 ~ -55	±3	0.1-0.3	low
3 - 6	-60 ~ -70	±5-8	0.5-1	medium
6 - 10	-70 ~ -90	±10	Over 1.5	high

Thus, by obtaining the RSSI values from the Bluetooth devices, the distance data between the guardian and the person being monitored can be acquired. This data is then assessed by the module, and if the distance exceeds the preset safety threshold, the module will trigger a voice alarm.

B. Case application of posture reminder

In daily life, some objects need to maintain a fixed posture, even during the transportation process, where there are certain posture requirements. Therefore, this voice interaction module can be connected to a gyroscope module. The gyroscope module can real-time acquire the tilt angle data of the detected object in the X, Y, and Z directions of the spatial coordinates. These three angle data can be calculated using the following formula:

$$X\_Angle = ((RollH<<8)|RollL)/32768*180 \quad (6)$$

$$Y\_Angle = ((PitchH<<8)|PitchL)/32768*180 \quad (7)$$

$$Z\_Angle = ((YawH<<8)|YawL)/32768*180 \quad (8)$$

In the formula, RollH, PitchH, and YawH correspond to the high angle values along the X, Y, and Z axes, respectively. RollL, PitchL, and YawL correspond to the low angle values along the X, Y, and Z axes. Since the sensor records data using 2-byte hexadecimal numbers, the posture angles are obtained by converting the ratio to the corresponding angle values.

By calculating these three angle offsets, the current instantaneous posture of the moved object can be determined. The offset is then calculated to check if it exceeds the safety range. If it does, the data is fed back to the interaction module, and the voice module will issue a timely voice reminder. This is the logical process of determining whether a voice reminder is needed based on the angular change data from the gyroscope.The voice interaction module and the gyroscope can be connected via serial communication. The connection diagram between the two is shown in Figure 7.

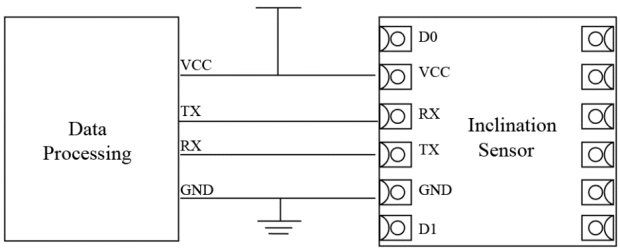

Fig.7. Schematic diagram of serial port connection between gyroscope and voice module

V. CONCLUSION

To sum up, the design and implementation of the voice interaction system based on the NanoPi NEO2 module is feasible. Through architecture design, hardware adaptation, software development, testing optimization, and application verification, the system's adaptation plan for voice interaction scenarios can be clearly defined. With hardware interface optimization and software environment configuration, efficient collaboration with peripheral devices and cloud services can be achieved.

However, some issues were also identified during actual testing: due to the local computational limitations of the module, more complex noise reduction and recognition systems could not be deployed. Therefore, the wake-up and recognition accuracy in noisy environments still has room for improvement. In the future, research can be conducted on lightweight algorithms, deploying small deep learning-based speech recognition models locally to further enhance the accuracy of speech recognition in noisy environments. Multimodal interaction fusion design can be explored, integrating various sensory recognition methods to achieve voice and visual multimodal interaction, improving the system's intent recognition accuracy and interaction experience. The application scope of the system can be further expanded by optimizing system functions for different industry scenarios, developing standardized hardware kits and software interfaces, and promoting the system's mass production and application in more lightweight scenarios.

REFERENCES

[1] Deng Lijun."Design and Implementation of an Online Language Translation Interactive Learning System Based on Speech Recognition Technology," Automation and Instrumentation, 2023, (06),pp. 199-203.

[2] Zhang Xuru, Ren Long, Chen Rui, et al, " Design and Development of an AI Voice Interaction Intelligent Seat Controller," IoT Technology, 2023, 13(02),pp.122-125+128.

[3] Chen Teng, Rong Xuewen, Li Yibin, "Experimental Design of Multimodal Control for a Quadruped Robot Based on Voice Interaction," Laboratory Research and Exploration, 2024, 43(10),pp. 65-69+106.

[4] Liao Chunlan, "Research and Design of a Voice Interaction System for a Welcoming Robot Based on ROS," Electromechanical Information, 2023, (23),pp.58-61.

[5] Li Hengbei, Lu Cuizhen, Lao Tianyuan, et al, "Design of the IoT-based Home Voice System for Small Household Devices, " IoT Technology, 2023, 13(11),pp. 112-114.

[6] Qiu Haiyan," Application of Voice Interaction in Intelligent Lighting Control Systems," IoT Technology, 2022, 12(11),pp. 55-572.

[7] Kan Baoqiang, Yu Rongseng. "Multimodal Interactive Intelligent Terminal," Journal of Changchun Normal University, 2024, 43(02),pp.84-91.

[8] Guo Hongzhen, Chen Mou, Dai Yongdong, et al, "Distributed Adaptive Event-triggered Quadrotor UAV Formation Control, " Acta Aeronautica et Astronautica Sinica, 2023, 44(S2),pp.491-500.

[9] Cao Xing, Zhang Lizeng, Yin Huijuan," Design of an Intelligent Refrigerator System Based on NanoPi2," Communications World, 2016, (12),pp. 243-244.

[10] Zou Yingjie, "Development of a Long-term Monitoring Terminal Based on MQTT Protocol and Gzip Compression Algorithm, " Guilin University of Technology, 2023.