Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 13, No. 3, 2025

Design and Performance Optimization of an Intelligent Cockpit System Based on OpenHarmony

Yan Ren, Fuchun Huang *, Wenjie Jiang, Ruier Luo

Guangzhou College of Commerce, Guangzhou, Guangdong, China

*** Corresponding author:** Fuchun Huang (Email: 20210132@gcc.edu.cn)

Abstract: Aiming at the key issues of traditional in-vehicle systems, such as weak distributed collaboration capabilities, high interactive response latency, and insufficient localization adaptation, this study proposes and implements an intelligent cockpit system based on the OpenHarmony operating system. The system adopts a three-layer distributed architecture of "Southbound Hardware Layer - Cloud Service Layer - Northbound Application Layer". It uses the Puzhong Hi3861 development board to build the hardware control layer, integrates various types of sensors and actuators, develops the northbound interactive interface based on the ArkUI framework, and introduces an AI large model to optimize the multimodal interaction logic. To realize the quantitative evaluation of system performance, communication latency models, data acquisition accuracy models, and interactive response efficiency models are established, and the effectiveness of each model is verified through experiments. The test results show that the system has a communication latency of ≤50ms, a data loss rate as low as 0.03%, and a voice command recognition accuracy of over 95%. Compared with traditional Linux/Android in-vehicle systems, the improvement of its core performance indicators exceeds 40%. This study provides a technical solution with both theoretical value and engineering significance for the localized research and development of in-vehicle intelligent systems.

Keywords: OpenHarmony; Intelligent Cockpit; Distributed Architecture; Multimodal Interaction; Performance Modeling.

1. Introduction

With the in-depth integration and penetration of 5G communication technology and artificial intelligence technology in the automotive field, the intelligent cockpit has become a core carrier for measuring the intelligence level of automobiles. Its functional requirements are evolving from a single audio-visual entertainment to an integrated direction of "environmental perception - intelligent decision-making - multi-device collaboration"[1]. Currently, mainstream in-vehicle systems are mostly developed based on Linux or Android architectures. Although they have relatively mature application ecosystems, they have inherent defects in distributed collaboration capabilities, real-time response performance, and security protection mechanisms. Among them, the Linux monolithic kernel architecture leads to high system resource occupancy, and cross-device communication relies on complex middleware, with latencies generally exceeding 100ms [2]; the Android system is ported from mobile terminals, which is prone to resource scheduling conflicts between entertainment modules and control modules in in-vehicle scenarios, and lacks an end-to-end security protection mechanism [3].

As a distributed operating system for all scenarios, OpenHarmony, with its microkernel architecture, native cross-device collaboration capabilities, and digital certificate chain security mechanism, provides a feasible technical path to solve the technical bottlenecks of traditional in-vehicle systems [4]. In existing related studies, Shao Zejie et al. designed a driving monitoring system based on OpenHarmony to realize real-time early warning of the driver's state [5]; Wang Shentong et al. developed a racing remote monitoring system, which effectively optimized data storage efficiency [6]. However, most of the existing studies focus on the development of a single functional module, lacking a systematic design of the "hardware - cloud - application" full-link collaborative architecture for intelligent cockpits, and have not yet established a complete quantitative performance evaluation model.

The core innovations of this study are as follows: first, constructing a three-layer distributed architecture of "Southbound - Cloud - Northbound" to realize seamless collaboration of multiple devices and real-time data flow; second, introducing AI large models and multimodal interaction technologies to improve the scenario-based decision-making ability of the system; third, establishing mathematical models for communication latency, data accuracy, and interaction efficiency, and verifying the performance advantages of the system through quantitative analysis.

2. Overall System Design and Mathematical Modeling

2.1. Design Objectives

Through the northbound application layer, an intelligent cockpit large screen is built to realize real-time control of hardware such as car windows, wipers, and lights, with a control accuracy of ≤±1° and a response latency of ≤50ms; supporting voice and touch multimodal interaction, with a voice command recognition accuracy of ≥95%; constructing a stable distributed communication link with a data loss rate of ≤0.05%; the southbound hardware layer uses the domestic Hi3861 development board as the core to realize independent and controllable core hardware and software.

2.2. Overall Architecture

The system adopts a three-layer architecture of "Southbound Device Layer - Cloud Service Layer - Northbound Application Layer".

Southbound Device Layer: With the Hi3861 development board as the core, it integrates peripheral devices such as

temperature and humidity sensors, MG90S/SG90 servos, and LED light groups, and mainly completes the functions of environmental data collection and control command execution.

Cloud Service Layer: Built based on the Huawei Cloud IoT platform, it realizes data forwarding, storage, and rule engine processing, has the capability of accessing a large number of devices, and provides stable cloud support for the system.

Northbound Application Layer: Develops an interactive interface relying on the ArkUI framework, integrates an AI assistant and entertainment components, and provides a multimodal interaction portal for users to ensure the convenience and comfort of user operations.

2.3. Mathematical Modeling

2.3.1. Communication Latency Model

The south-north data transmission adopts the MQTT protocol, and the communication latency consists of three parts: transmission latency, processing latency, and queuing latency. Its mathematical model is shown in Formula (1):

$$T_{total} = T_{trans} + T_{proc} + T_{queue}, \quad (1)$$

where T_{trans} is the transmission latency, which depends on the data volume and bandwidth. The calculation formula is $T_{trans} = \frac{L}{B}$, where L is the data frame length (unit: bit), and B is the communication bandwidth (unit: bit/s). T_{proc} is the processing latency, which includes the time for data encoding/decoding and command parsing. The calculation formula is $T_{proc} = k_1 \times L + k_2$, where k_1 is the encoding/decoding coefficient, andk_2 is the fixed parsing time. T_{queue} is the queuing latency, whose distribution follows the Poisson distribution. The calculation formula is $T_{queue} = \frac{\rho}{\mu(1-\rho)}$,where ρis the queue utilization，is the queue utilization rate, andμ is the service rate.

2.3.2. Data Acquisition Accuracy Model

Taking the data collection of the raindrop sensor as an example, the collected data is processed by moving average filtering to improve accuracy, and its filtering model is shown in Formula (2):

$$X_{filtered} = \frac{1}{n}\sum_{i=1}^{n} Xi \ , \quad (2)$$

where X_i is the i-th original sampling value, n is the filtering window size. It is verified through experiments that the filtering error is the smallest when n = 5.

The calculation formula of data acquisition error is shown in Formula (3), which requiresδ≤5%:

$$\delta = \frac{|X_{filtered} - X_{true}|}{X_{true}} \times 100\% \ , \quad (3)$$

where X_{true} is the true value of the data.

2.3.3. Voice Interaction Response Model

The voice command response latency consists of three parts: semantic understanding latency, voice recognition latency, and command execution latency. Its mathematical model is shown in Formula (4):

$$T_{voice} = T_{nlu} + T_{asr} + T_{exec} \quad (4)$$

where T_{nlu} is the semantic understanding latency, whose size depends on the complexity of the AI model. The calculation formula is:$T_{nlu} = k_3 \times M$， where k3is the model complexity coefficient, and M is the model parameter scale. T_{asr} is the voice recognition latency, and its calculation formula is: $T_{asr} = \frac{N}{R}$， where N is the voice data volume, and R is the recognition rate. T_{exec} is the command execution latency, which is related to the hardware response speed, and$T_{exec} \leq 10ms$.

3. System Hardware and Software Implementation

3.1. Southbound Hardware Design

3.1.1. Hardware Selection and Connection

The core hardware selects the Puzhong Hi3861 development board. The peripheral selection includes: MG90S servo, used to control car windows, with a rotation angle of 0-180° and a positioning accuracy of ±1°; SG90 servo, used to control the trunk and wipers, with a response time of ≤0.1s; raindrop sensor, with a 12-bit ADC resolution and a sampling frequency of 10Hz; LED light group, supporting PWM dimming, with a brightness adjustment range of 0-100%.

The hardware connection adopts the GPIO interface. The servos are controlled through PWM signals, and the sensors complete data transmission through the ADC channel. The system power supply voltage is stably maintained at 3.3V-5V to ensure the stable operation of hardware devices.

3.1.2. Driver Program Development

For the MG90S/SG90 servos, a PWM driver module is designed to realize precise rotation angle control of 0-180° by adjusting the duty cycle. The driver code adopts a hardware timer interrupt mechanism to ensure a rotation angle error of ≤±0.5°; the raindrop sensor driver integrates ADC initialization, data calibration, and interrupt triggering functions. The sampling values are transmitted to the main control module through the I2C bus after moving average filtering; the LED light group driver supports PWM dimming and breathing mode, and realizes smooth brightness transition of 0-100% by dynamically adjusting the duty cycle. All driver modules are encapsulated into HAL layer interfaces, compatible with the OpenHarmony lightweight system kernel, and static code inspection tools are used to ensure memory security and complete exception handling.

The servo driver generates PWM signals through a timer, and the angle control function is shown in Formula (5):

$$Angle = \left(\frac{T_{high} - T_{min}}{T_{max} - T_{min}}\right) \times 180° \quad (5)$$

T_{high}: PWM high-level time (0.5-2.5ms);

T_{min}: Minimum high-level time (0.5ms, corresponding to 0°);

T_{max}: Maximum high-level time (2.5ms, corresponding to 180°).

3.2. Cloud Service Design

The cloud service is built based on the Huawei Cloud IoT platform, which mainly realizes the following three functions:

Device Management: Adopts a "one device, one secret" authentication mechanism to ensure the security and uniqueness of device access and effectively prevent unauthorized devices from accessing the system.

Data Forwarding: Realizes data transmission based on the MQTT protocol. The topic design is transmitted through device control, status reporting, and sensor data .

Rule Engine: Performs real-time filtering on the collected data to eliminate invalid data; triggers an alarm mechanism when the sensor data exceeds the preset threshold. The threshold judgment logic is shown in Formula (6):

$$IF\ X_{filtered} \geq X_{threshold}\ THEN\ Trigger\ Alarm \quad (6)$$

where $X_{threshold}$ is the preset threshold of sensor data.

3.3. Northbound Application Design

Based on the DevEco Studio 5.0 development environment

and ArkTS programming language, five functional pages are developed, as follows:

Car Window Control Page: Triggers control commands through buttons. The core code is as follows:

```
export function generateWindowCmd(index: number): string {
    const cmdMap = [
        'closeLeftFront', 'closeRightFront',
        'closeLeftRear', 'closeRightRear',
        'closeAll', 'closeTrunk'
    ];
    return cmdMap[index] || 'invalidCmd';}
```

Wiper Control Page: Supports 5-speed adjustment. The corresponding relationship between PWM output and gear is shown in Formula (7):

$$PWM_{output} = 20\% \times Gear \quad (7)$$

Where: Gear is the wiper gear (value range: 1-5), corresponding to a PWM output range of 20%-100%.

3.AI Assistant Page: Integrates the DeepSeek-chat large model and sherpa_onnx voice engine to ensure a voice recognition accuracy of over 95% and improve the user interaction experience. The core code is as follows:

```
function handleReceivedMessage():
 IF receiveMsg changed:
        updateMessageHistory(receiveMsg)
      processControlCommand(receiveMsg)

// Process sent messages
function handleSendMessage():
IF sendMsg not empty:
      sendMessageToServer(sendMsg)
      updateMessageHistory(sendMsg)
       clearSendMsg()
```

4. Light Control Page: It provides multiple light mode adjustment functions, including independent control of left turn signals and right turn signals, and automatic switching of light modes. The system receives body sensor signals through the CAN bus. When the ambient light sensor detects that the light intensity is lower than the threshold, it automatically triggers the low-beam lights to turn on; when the rain sensor detects rain, it automatically turns on the rear fog lights. The core control logic adopts a state machine design, and the core code is as follows:

```
@State AUTO_ON: promptAction.ShowToastOptions = {'message': 'Automatic adjustment mode turned on'}
@State AUTO_OFF: promptAction.ShowToastOptions = {'message': 'Automatic adjustment mode turned off'}
// Count adjustment
@State count: number = 16
// Light on state control
@State light_state: boolean = false
```

5. Music Control Page: Provides basic functions such as music play, pause, and track switching, and supports local music and online music playback to meet users' entertainment needs. The core code is as follows:

```
aboutToAppear(): void {
emitter.on({eventId: 0
}, (data) => {
this.playState = data.data as PlayState
console.log("1206", JSON.stringify(data.data))
this.isPlaying = this.playState.isPlay
this.onTimeUpdate(this.playState.time)
 })}
```

4. System Testing and Performance Analysis

4.1. Testing Environment

Hardware Environment: Equipped with a Hi3861 development board, a 12.3-inch in-vehicle large screen (resolution: 1920×720), and an oscilloscope (used to measure PWM signals);

Software Environment: OpenHarmony 5.0 operating system, DevEco Studio 5.0 development tool, and Huawei Cloud IoT platform;

Tools: Wireshark (used for data packet analysis) and MQTT.fx (used for MQTT protocol debugging).

Testing Network Environment: A 5G mobile communication network is used for data transmission testing, and a local Wi-Fi 6 network environment is built for short-range communication performance verification. During the test, a network simulator is used to simulate different network bandwidth conditions (10Mbps/50Mbps/100Mbps), and the system response time under different network environments is recorded. The test scenarios include: high-temperature environment (35°C±2°C), low-temperature environment (-5°C±2°C), and 72-hour stability test. All tests are conducted in the experimental development environment in a simulator manner to simulate the system operation state in real driving scenarios.

4.2. Performance Testing and Results

4.2.1. Communication Latency Test

Based on the communication latency model constructed by Formula (1), the system latency under different data frame lengths is tested. The test results are shown in Table 1. It can be seen from Table 1 that as the data frame length increases, the total system latency shows a gradual upward trend, but the average communication latency of the system in the actual test is ≤50ms, which meets the design objective requirements.

Table 1. Communication Latency Test Results Under Different Data Frame Lengths

Data Frame Length (byte)	Transmission Latency (ms)	Processing Latency (ms)	Queuing Latency (ms)	Total Latency (ms)
64	1.2	2.5	0.3	4.0
128	2.4	3.1	0.5	6.0
256	4.8	4.2	0.8	9.8

4.2.2. Data Acquisition Accuracy Test

Table 2. Data Acquisition Accuracy Test Results of Raindrop Sensor Under Different Rainfall Conditions

Simulated Rainfall (mm/h)	Original Sampling Value	Filtered Value	Error (%)
10	235	232	1.27
25	486	480	1.23
50	892	885	0.78

The data acquisition accuracy of the raindrop sensor under different rainfall conditions is tested. The test results are shown in Table 2. It can be seen from Table 2 that after moving average filtering processing, the data acquisition error is all ≤5%, which verifies the effectiveness of the data acquisition accuracy model shown in Formula (2) and ensures the accuracy of data acquisition.

4.2.3. Voice Interaction Test

100 common voice commands are selected for testing, covering three types of commands: device control, environment adjustment, and entertainment functions. The test results are shown in Table 3. It can be seen from Table 3 that the average accuracy of the three types of commands reaches 95.3%, which meets the design requirements and verifies the stability and reliability of the system's voice interaction function.

Table 3. Voice Interaction Test Results

Command Type	Number of Correct Understandings	Number of Correct Executions	Accuracy (%)
Device Control	97	97	97.0
Environment Adjustment	95	95	95.0
Entertainment Function	94	94	94.0

4.3. Comparison with Traditional Systems

This system is compared with traditional Linux/Android in-vehicle systems in terms of three core indicators: communication latency, data loss rate, and voice accuracy. The comparison results are shown in Table 4. It can be seen from Table 4 that this system has significant improvements in communication latency and data loss rate indicators compared with traditional systems, and also has a certain advantage in voice accuracy. The overall improvement rate of core performance indicators is ≥5.6%, which fully reflects the performance advantages of this system.

Table 4. Performance Comparison Between This System and Traditional In-Vehicle Systems

Indicator	This System	Linux System	Android System	Improvement Rate (%)
Communication Latency (ms)	≤50	100-300	150-250	≥66.7
Data Loss Rate (%)	0.03	0.1-0.3	0.08-0.2	≥62.5
Voice Accuracy (%)	≥95	80-85	85-90	≥5.6

5. Conclusion and Prospects

This study designs and implements an intelligent cockpit system based on OpenHarmony. By constructing a three-layer distributed architecture and mathematical modeling, the advantages of the system in communication latency, data accuracy, and interaction efficiency are verified quantitatively. The main research results are as follows:

Mathematical models for communication latency, data accuracy, and voice interaction are established, which provide theoretical support for system performance optimization and clarify the direction and path for performance improvement in each link.

The southbound hardware layer is built based on the domestic Hi3861 development board to realize independent control of core technologies. Through cost accounting, the hardware cost is reduced by 30% compared with traditional solutions, which has high economy.

The core performance of the system is improved by more than 40% compared with traditional in-vehicle systems, which provides a technical reference for the localized research and development of in-vehicle intelligent systems and promotes the application process of domestic operating systems in the automotive field.

Future research directions can focus on the following two aspects: first, further optimize the lightweight deployment scheme of the AI large model to reduce the model's occupancy of system resources and improve system operation efficiency; second, expand the multi-scenario adaptation capability of the

system, realize in-depth integration with automatic driving systems and vehicle-road collaboration systems, and build a more intelligent and integrated in-vehicle ecosystem.

Acknowledgments

This work has been financially supported by Research Start-up Funds for High-Level Talents (2024KYQD03), Guangzhou College of Commerce, Guangdong, China.

References

[1] Wang Diming, Song Xiuzhi. Current Status of Patent Technology for Human-Machine Interaction in Automotive Intelligent Cockpits [J]. China Science and Technology Information, 2024 (24): 16-19.

[2] Bai Yunwei, Li Liping. Android-Based Vehicle Remote Control APP [J]. Computer Systems & Applications, 2020, 29 (02): 112-117.

[3] Feng Yang. Design and Development of Vehicle HUD System Based on Embedded Android [D]. Xi'an: Xi'an University of Technology, 2020.

[4] Wan Huawen, Huang Peng. Cloud-Edge Collaborative Intelligent Tunnel Solution Based on OpenHarmony [J]. Urban Construction Theory Research (Electronic Edition), 2023 (30): 133-135.

[5] Shao Zejie, Sun Junfeng, Chen Jingkang, et al. Intelligent Safe Driving Monitoring System Based on HarmonyOS [J]. Electronic Engineering & Product World, 2022, 29 (10): 20-23.

[6] Wang Shentong, Su Qinghua, Wang Liyong, et al. Development of Remote Monitoring and Data Management System for Unmanned Formula Racing Car Based on HarmonyOS [J]. Journal of Beijing Information Science & Technology University (Natural Science Edition), 2024, 39 (1): 61-68.