CogniDrive - Intelligent Vehicle-mounted Integrated Service System

Chengwei Liu*, Congyuan Yin, Yuxuan Han, Yumeng Zhang
School of Intelligence & Electronic Engineering, Dalian Neusoft University of Information, Dalian 116023, China
*Corresponding email: liuchengwei088@163.com

Abstract

Aiming at the problems of fragmented functions and single interaction mode of traditional vehicle-mounted terminals, an intelligent vehicle-mounted integrated service system based on the Android platform is designed and implemented. Taking “mobile terminal-hardware collaborative control” as the core, the system integrates six modules: vehicle status monitoring, Bluetooth communication, voice recognition, music playback, map navigation, and multi-interface collaborative management. The 51 single-chip microcomputer is adopted as the underlying control core, and full-duplex low-latency communication is realized through the Radio Frequency Communication (RFCOMM) protocol under Bluetooth. Baidu Speech Recognition Software Development Kit (SDK) and Amap SDK are integrated to construct a closed-loop interaction model of “voice command - function execution - visual feedback”. Tests indicate that the system meets the real-time requirements of vehicle-mounted scenarios in key indicators such as communication latency, recognition accuracy, and positioning precision. It achieves a low-cost and high-adaptability lightweight vehicle-mounted solution, providing an effective technical reference for human-computer interaction of intelligent connected vehicles.

Keywords

Intelligent vehicle-mounted system, Mobile terminal-hardware collaboration, Voice interaction, Bluetooth communication, Android

Introduction

With the rapid development of the intelligent connected vehicle industry, the vehicle-mounted information interaction system has become a core element to improve driving safety and convenience. However, traditional vehicle-mounted control terminals generally have problems such as fragmented functions, single interaction mode and poor hardware adaptability, which are difficult to meet the urgent needs of users for multi-modal interaction and equipment collaboration. In this context, vehicle-mounted applications (APPs) based on smartphones have gradually become a research hotspot in the field of intelligent vehicle connection due to their advantages of light weight, high adaptability and low cost.

In terms of vehicle control through mobile applications, scholars have carried out a series of explorations. Studies have shown that remote control of toy cars can be realized based on Android devices and Bluetooth communication technology, which verifies the feasibility of wireless control of embedded hardware by mobile terminals. Some researchers have designed an intelligent car control system based on Android smartphones and Peripheral Interface Controller (PIC) microcontrollers, using the Radio Frequency Communication (RFCOMM) protocol under Bluetooth to complete command transmission and realize basic motion control such as forward, backward and steering. In addition, some researchers proposed a mobile application scheme for Brushless Direct Current (BLDC) motor speed control through a secure Bluetooth channel, which has shown good applicability in industrial scenarios. However, the above studies mostly focus on a single motion control function, lacking the integration of comprehensive vehicle-mounted services such as real-time vehicle status monitoring, multimedia services and intelligent navigation.

Important progress has also been made in the field of vehicle-mounted interaction and status monitoring.

Monk et al. evaluated the visual and cognitive loads of manual and voice driving modes on smartphones through comparative experiments [1]. The results show that voice interaction can significantly reduce the degree of driver distraction, providing an important basis for the design of vehicle-mounted voice interaction.

Mohammed et al. designed an Internet of Things-based intelligent driver monitoring system, which uses PIC microcontrollers to collect parameters such as vehicle speed, engine speed and coolant temperature, and realizes remote graphical monitoring through WiFi modules, effectively improving the real-time performance of driving behavior recognition [2]. Some researchers proposed a low-cost vehicle tracking architecture integrated with Global Positioning System (GPS)/ Global System for Mobile Communications (GSM) modules, which triggers position feedback through smartphone calls, achieving a positioning accuracy of <5 meters and a response time of <3 seconds [3]. Nevertheless, existing studies either focus on a single monitoring function or rely on high-cost hardware platforms, and there are obvious deficiencies in the lightweight integrated solution of "mobile terminal-hardware collaboration" [4].

Compared with existing studies, the "Zhiyu Chexing" system designed in this paper achieves breakthroughs in the following aspects: First, it constructs a complete closed loop with in-depth integration of three modules: "vehicle collaborative control - intelligent multimedia service - multi-interface collaborative management". It integrates Bluetooth communication, voice recognition, music playback and map navigation into one, overcoming the limitation of fragmented functions. Second, it adopts STC51 single-chip microcomputer and HC-05 Bluetooth module to build a low-cost hardware platform, realizing millisecond-level status data feedback and User Interface (UI) synchronization through custom frame format, which significantly reduces the system cost while ensuring real-time performance. Third, it integrates Baidu Speech Recognition SDK and Amap API to construct a multi-modal interaction model of "voice command - function execution - visual feedback", realizing hands-free operation in driving scenarios. The above characteristics make the system have significant advantages in low cost, high integration and multi-modal interaction, providing an effective technical reference for human-computer interaction of intelligent connected vehicles.

Problem description

This paper designs and implements a lightweight integrated service system for intelligent connected vehicles, which adopts a hierarchical architecture of "mobile terminal-hardware collaboration" as a whole. Taking the Android application as the user interaction carrier and the STC51 single-chip microcomputer as the underlying control core, the system establishes a full-duplex wireless data link through the Bluetooth RFCOMM protocol to realize a closed loop of command issuance and status feedback. The overall architecture of the system is shown in Figure 1, and the core functions are integrated into three modules.

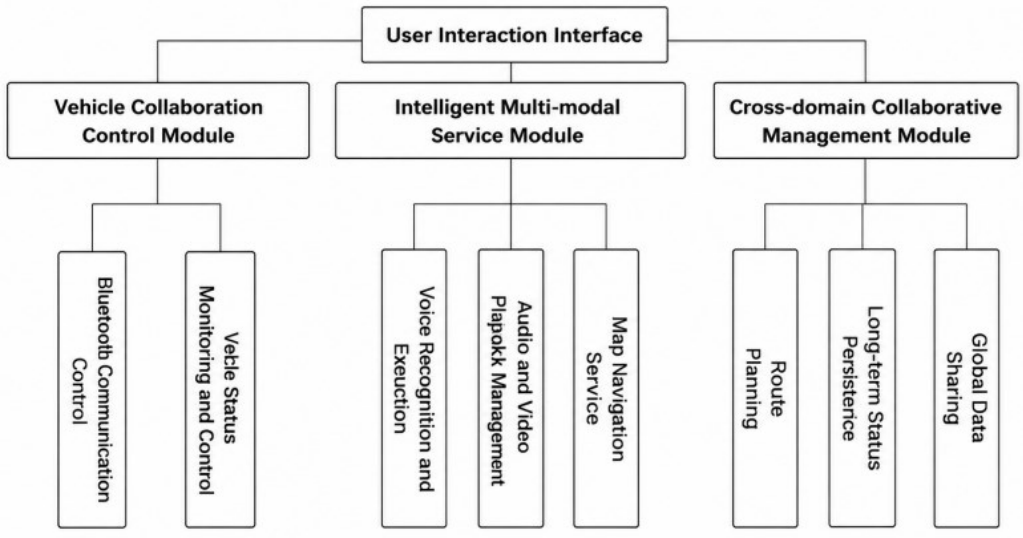

Figure 1. Overall system functional block diagram.

(1) Vehicle collaborative control module

It integrates Bluetooth communication and vehicle status monitoring functions. The former is responsible for device discovery, pairing and RFCOMM connection between the Android device and the STC51 single-chip microcomputer (equipped with HC-05 module), binds the serial port service with a fixed Universally Unique Identifier (UUID), and processes the sending of control commands and the receiving of status data concurrently through a multi-thread mechanism. The latter parses and displays information such as vehicle speed, temperature and power in real time, performs integrity verification and abnormal filtering on data frames, and ensures the synchronous update of the UI and hardware status.

(2) Intelligent multimedia service module

It integrates voice recognition, music playback and map navigation functions. For voice recognition, Baidu Speech Recognition SDK is integrated for voice activity detection, and a general Chinese Mandarin recognition model is adopted to realize intent parsing of commands such as “open navigation” and “play music” through keyword matching. Music playback calls the Android MediaPlayer framework, manages local audio resources through AssetManager or ContentResolver, and supports playback control and progress synchronization. Map navigation integrates Amap SDK, obtains the current position using a multi-source positioning fusion algorithm, converts addresses into coordinates based on geocoding, calculates the optimal driving route using Dijkstra/A* algorithm, and seamlessly calls third-party navigation clients through Uniform Resource Identifier (URI) Scheme.

(3) Multi-interface collaborative management module

It is responsible for routing between interfaces, state persistence and global data sharing. A bottom navigation bar is adopted to realize quick switching between the main view, control view and multimedia view. The Handler message passing mechanism is used to solve the communication security between sub-threads and UI threads. The Application singleton and observer pattern are used to manage global states such as Bluetooth connection and user login information, supporting state recovery after screen rotation or process termination.

The three modules operate collaboratively to effectively form a closed loop of “command input-data transmission-function execution-feedback presentation”, aiming to realize a low-cost, highly integrated multi-modal in-vehicle solution [5]. The overall system interface is shown in Figure 2.

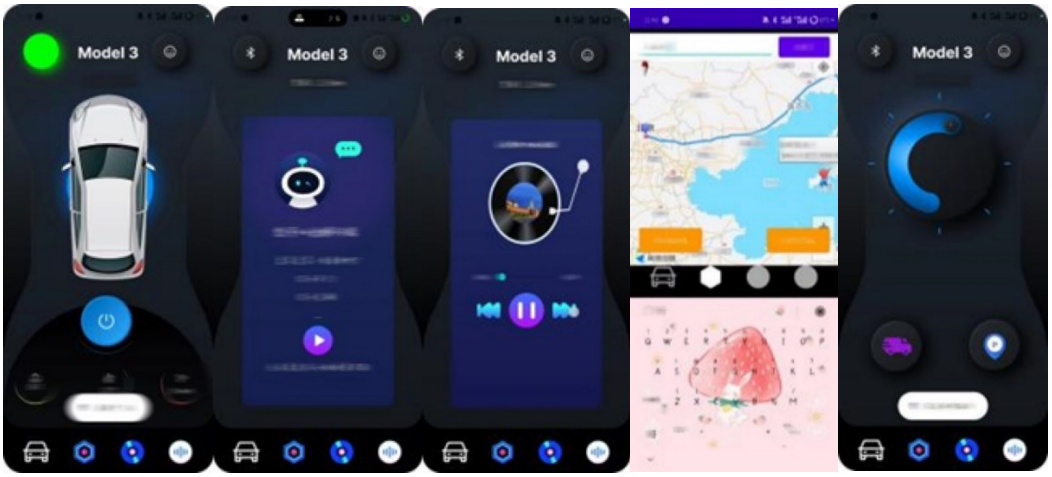

Figure 2. Overall system interface.

Hardware system design

Main control unit and communication interface

The system selects the STC51 series single-chip microcomputer (compatible with ATC89C51) as the main control chip of the lower computer. This chip has the advantages of low cost, low power consumption, simplified instructions and strong anti-interference ability, which meets the real-time motion control needs of intelligent cars.

The minimum system of the single-chip microcomputer includes a crystal oscillator circuit (11.0592 MHz), a reset circuit and a power filter circuit, ensuring stable system clock and reliable power-on reset. The overall architecture diagram of the hardware system is shown in Figure 3.

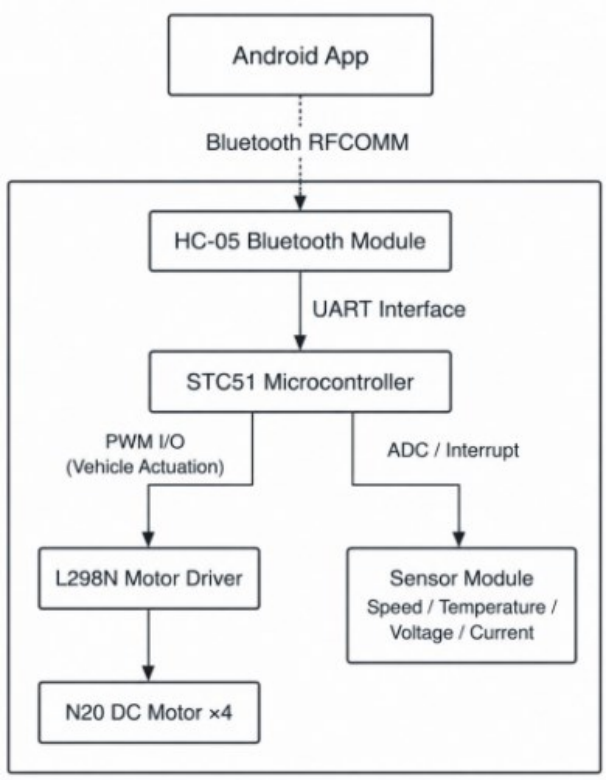

Figure 3. Overall architecture diagram of hardware system.

The Bluetooth communication module adopts HC-05, which supports master-slave integrated mode and works in slave mode by default. It is connected to the STC51 single-chip microcomputer through a Universal Asynchronous Receiver-Transmitter (UART) serial port (Transmit Data (TXD) → P3.1, Receive Data (RXD) → P3.0). The module is configured with a fixed baud rate of 9600bps, and the data format is 8 data bits, 1 stop bit and no parity check.

Motor drive and motion execution unit

The motion execution system is composed of four N20 DC gear motors, an L298N motor drive module and a multi-form mechanical chassis. The N20 motor has a built-in reduction gearbox, with large output torque and compact volume, which is suitable for the four-wheel independent drive scheme of the car. The L298N drive module integrates a dual H-bridge circuit, each channel can provide a maximum drive current of 2A. Pulse Width Modulation (PWM) signals are input through the Input/Output (I/O) ports (P1.0~P1.3) of the STC51 to realize motor speed regulation, and another four I/O ports are used to control forward and reverse rotation. The motor power is supplied by two 18650 lithium batteries connected in series (7.4 V), and the logic power is regulated to 5 V by LM2596 to supply the single-chip microcomputer and Bluetooth module.

The chassis is made of Acrylonitrile Butadiene Styrene (ABS) material, supporting three form switches: four-wheel differential, Ackermann and Mecanum wheel. This system takes the four-wheel differential chassis as the benchmark, with four motors controlled independently, and realizes in-place steering, forward and backward through the speed difference between the two sides of the wheels. A steering gear is configured in the steering module for front wheel guidance in the Ackermann form. The wheel axle and the motor are installed in a plane fit way to ensure that the power transmission does not slip [6].

Sensor and status monitoring module

To realize the self-perception of vehicle status, the system is equipped with three types of sensors: The speed sensor adopts a Hall effect encoder (installed on the motor output shaft), which outputs pulse signals to the external interrupt pin (P3.2) of the single-chip microcomputer, and converts the pulse count per unit time into real-time vehicle speed. The temperature sensor selects the DS18B20 single-bus digital sensor (connected to P2.0) to measure the temperature near the chassis motor. The power monitoring collects the battery voltage divider value through the Analog-to-Digital Converter (ADC) channel (P1.7) of the single-chip microcomputer, and converts it into the remaining power percentage through segmental linear calibration. All sensor data are collected and integrated by the single-chip microcomputer regularly and sent together with the status frame.

In terms of power management, the system is equipped with an independent 5 V/3.3 V dual-channel voltage stabilization circuit to ensure power isolation and ripple suppression between the single-chip microcomputer, Bluetooth module, sensors and steering gear. The overall hardware system takes STC51 as the core, and realizes a complete closed loop from command reception to motion execution and then to status feedback through resources such as UART, I/O, ADC and external interrupt, providing a stable and reliable physical foundation for upper computer control [7].

Software system design

Overall architecture design

The system software is divided into a four-layer system

from top to bottom: UI interaction layer, business logic layer, interface service layer and data persistence layer. The overall system architecture diagram is shown in Figure 4. The UI interaction layer is responsible for interface rendering, user operation response and status visualization display, providing a friendly vehicle-mounted interaction interface. The business logic layer processes core businesses such as user authentication, multimedia scheduling, navigation calculation and service routing, which is the core of system function scheduling; the interface service layer encapsulates third-party SDK calls and system component adaptation, providing standardized service interfaces for upper-layer logic.

The data persistence layer is responsible for the local storage and recovery of user information, interface status and configuration parameters. Data interaction between layers is realized via Android message mechanisms (i.e., Handler and Message) as well as asynchronous callbacks, in strict accordance with the development principle that time-consuming tasks are processed in sub-threads while the UI is updated in the main thread. This design avoids UI blocking and potential thread-safety risks, thereby guaranteeing low latency and high operational fluency for vehicle-mounted scenarios [8].

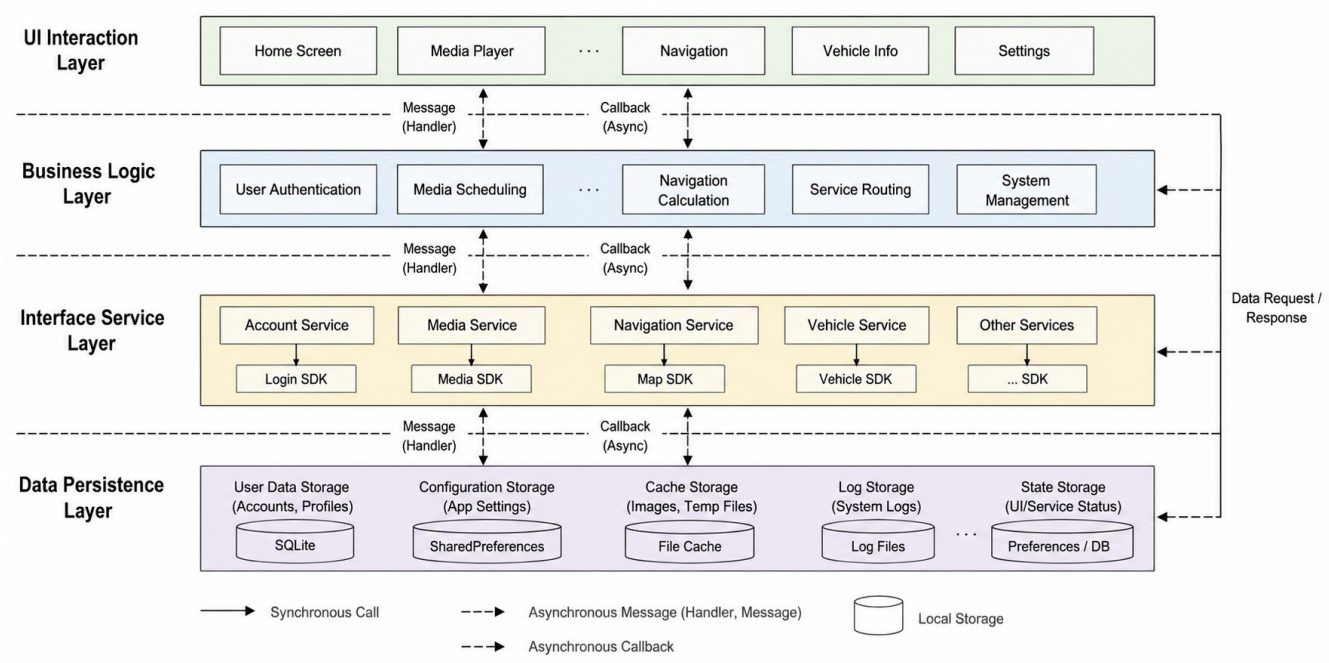

Figure 4. Overall system architecture diagram.

Functional module design

(1) User authentication management module

As the secure access entrance of the system, the user authentication management module realizes the functions of account registration, login verification and local persistence of information. Based on the Android built-in SQLite database, the module completes database creation, table structure management and version upgrade maintenance through the SQLiteOpenHelper class, and constructs a user data table including user identifier (ID), user name, email and password. In the registration process, the system performs legality verification on the uniqueness of the user name, email format, password length and consistency to improve information security. In the login process, the system performs matching verification on the input account and password. After successful verification, the user session information is stored in a global singleton mode, and the main interface is started through Intent, and the task stack is cleared to prevent the user from returning to the login interface. Through standardized error prompts and interface jump logic, the module realizes safe and simple account management functions, providing basic access control for the system.

(2) Vehicle-mounted service support module

The vehicle-mounted service support module provides upper-layer application services for the system, does not repeat the content of software and hardware communication interaction, and focuses on the integration and scheduling of software-side services. The multimedia service is implemented based on the

Android native MediaPlayer framework, reads local audio resources through AssetManager, constructs song lists and play queues, and supports play or pause, previous song or next song, progress drag and automatic song switching after playback. The module uses Handler timing tasks to realize the synchronous refresh of playback progress and time text, and combines life cycle callbacks to save and restore the playback state to ensure the continuity of playback after interface reconstruction.

The map navigation service integrates Amap SDK to complete location services and path planning, realizes multi-source fusion positioning through AMapLocationClient, and the outdoor positioning error does not exceed 5 meters. The GeocodeSearch class is used to convert text addresses into latitude and longitude coordinates, and the optimal driving route is calculated based on Dijkstra and A* path planning algorithms, and the route drawing, mileage and time consumption display are completed on the map interface. The module calls the third-party Amap client through URI Scheme to realize professional navigation, and strictly follows the privacy compliance requirements to complete SDK initialization and permission configuration, providing reliable location service support for vehicle travel.

(3) Interface and status collaboration module

The interface and status collaboration module is responsible for multi-interface management, global state unification and cross-interface data synchronization, ensuring the continuity of user operations and interface consistency. The module adopts BottomNavigationView to realize quick switching between the main interface, control interface, music interface and voice interface, and realizes cross-interface parameter transmission through Intent carrying Bundle data. The system adopts a three-layer state management mechanism of "lifecycle awareness + thread-safe synchronization + cross-interface sharing": Saving and restoring key interface states through onSaveInstanceState and onCreate methods to avoid data loss caused by screen rotation and memory recycling. Relying on the Handler message mechanism to complete the safe update of sub-thread data to the main thread UI, ensuring real-time synchronization of vehicle information, playback state and connection state. Using the Application global singleton mode to store shared data such as Bluetooth connection state and user information, and combines the observer pattern to realize unified refresh of multi-interface states, avoiding data inconsistency. Through optimization strategies such as anti-duplicate clicks and status visualization feedback, the module improves the operation reliability and interaction experience in the vehicle-mounted scenario.

System operation mechanism

After the system is started, it sequentially executes the processes of permission check, user authentication and core service initialization. After successful authentication, it enters the main interface and starts service monitoring. Users trigger service scheduling through interface operations or voice commands. Time-consuming businesses such as multimedia and navigation are executed asynchronously in sub-threads, and the processing results are synchronized to the UI layer through the callback mechanism. The system performs continuous state monitoring and stable service operation throughout its operation process, delivers standardized prompts, and implements fault-tolerant processing under abnormal scenarios, thereby forming an efficient interaction closed loop of "command input-logic processing-feedback presentation". This characteristic satisfies the lightweight, high-reliability, and expandable design objectives of the intelligent vehicle-mounted system.

Software and hardware communication interaction design

Bluetooth RFCOMM serial communication and command transmission

Bluetooth communication adopts RFCOMM protocol, and identifies the serial port service based on a fixed UUID (00001101-0000-1000-8000-00805F9B34FB). The Android terminal obtains the list of paired devices through BluetoothAdapter, screens the target device, and calls BluetoothSocket to establish a connection. The communication link adopts a multi-thread design: ConnectThread is responsible for initiating the connection, ConnectedThread continuously monitors the input stream and receives data, and the main thread processes messages asynchronously through Handler to avoid UI blocking.

Command transmission follows the "request-response" mode. When the user clicks the acceleration/braking

button or triggers control through voice, the software encodes the operation into a single-byte command (such as “1” for acceleration and “2” for braking). The command is written into the Bluetooth channel through the output stream of ConnectedThread, and the 51 single-chip microcomputer drives the motor module to execute the corresponding action after receiving it. At the same time, the single-chip microcomputer periodically returns status data in a custom frame format: “Speed, Temperature, Battery”. The Android terminal reads the data stream in ConnectedThread, extracts the complete frame by delimiter “#”, parses speed, temperature and power fields, and after range verification (speed 0~20,000, temperature 0~30,000, power 0~200), sends it to the main interface via Handler to update TextView component. This mechanism controls command response delay at the millisecond level and supports automatic reconnection within 5 seconds, ensuring link stability [9]. The Bluetooth communication timing diagram is shown in Figure 5.

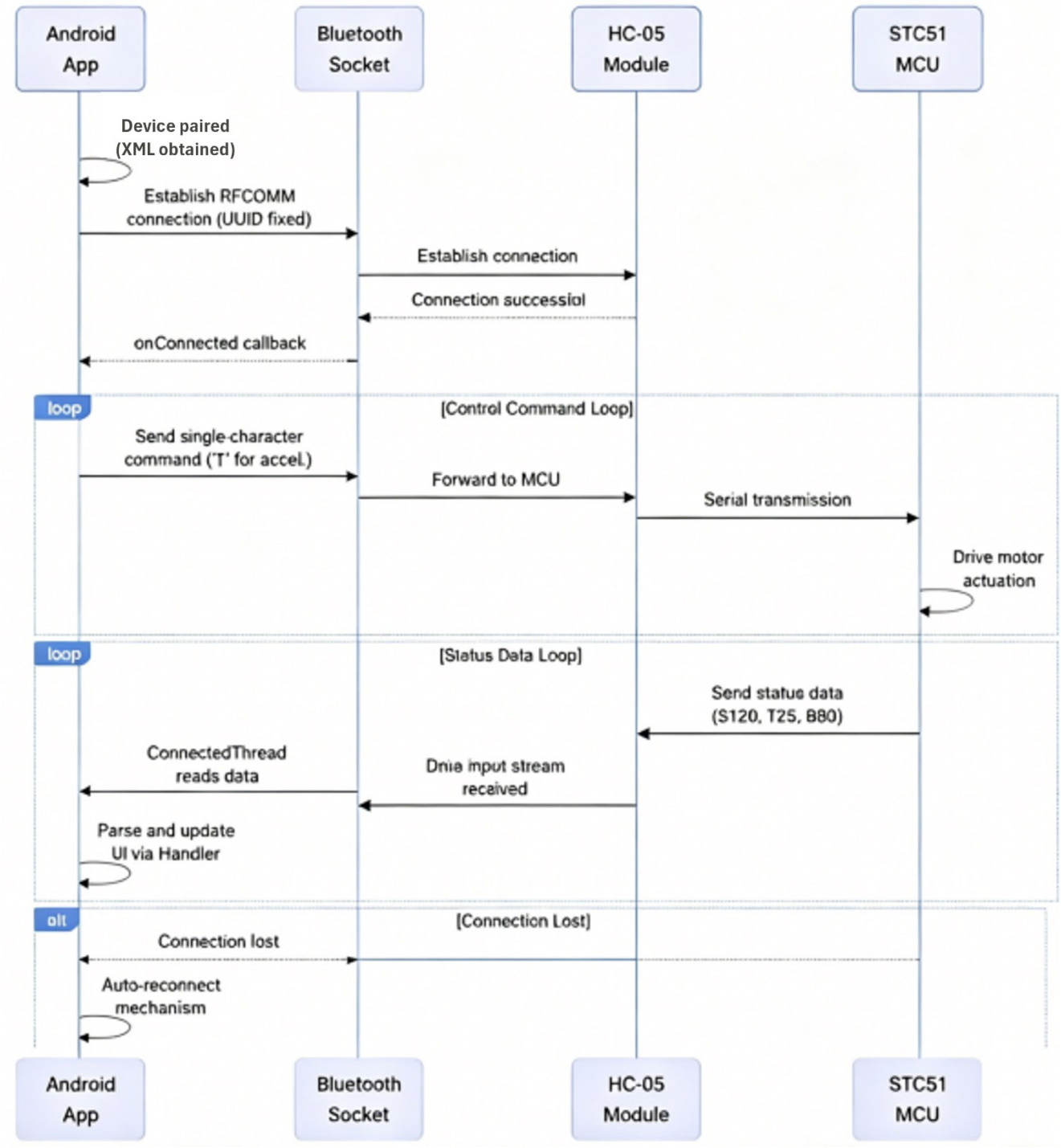

Figure 5. Bluetooth communication timing diagram.

Interactive implementation of voice control and hardware linkage

To realize hands-free interaction of "voice as command", the system integrates Baidu Speech Recognition SDK, parses natural language into hardware control commands, forming a closed loop of voice → text → command → hardware execution. When the user clicks the microphone button on the voice interface, the SDK starts audio collection, detects voice activity and filters environmental noise. The collected audio is compressed and uploaded to Baidu Cloud, and the text result is returned based on the Proportional-Integral-Derivative (PID)=1,537 Chinese Mandarin general recognition model. The software parses the intent through a keyword matching algorithm: If the text contains keywords such as "accelerate" and "forward", it is mapped to the acceleration command "1". If it contains "brake" and "stop", it is mapped to the brake command "2". The parsed command is issued to the 51 single-chip microcomputer for execution through the Bluetooth communication module.

In addition, voice commands also support cross-module linkage: When the user says "play music", the system starts the music playback Activity through Intent. When saying "open navigation", it calls the Amap SDK for path planning. For non-hardware commands, the system gives feedback through Toast or voice broadcast. This design unifies the scheduling of the voice entrance with Bluetooth control and multimedia services, allowing users to complete mixed operations without manually switching interfaces [10]. Its flow chart is shown in Figure 6.

As illustrated in Figure 6, the Bluetooth communication procedure standardizes the data transmission mechanism between mobile terminals and vehicle-side hardware. Following Bluetooth activation, the system performs permission authentication to comply with mobile-end privacy protocols. Upon successful device discovery and connection establishment, real-time voice-derived control signals are stably transmitted to the single-chip microcontroller, enabling low-latency command execution and ensuring reliable bidirectional interaction for in-vehicle voice-driven operations. This sequential authentication-connection-transmission workflow effectively mitigates communication interruption risks and enhances the robustness of human-vehicle interactive control under complex in-vehicle operational scenarios.

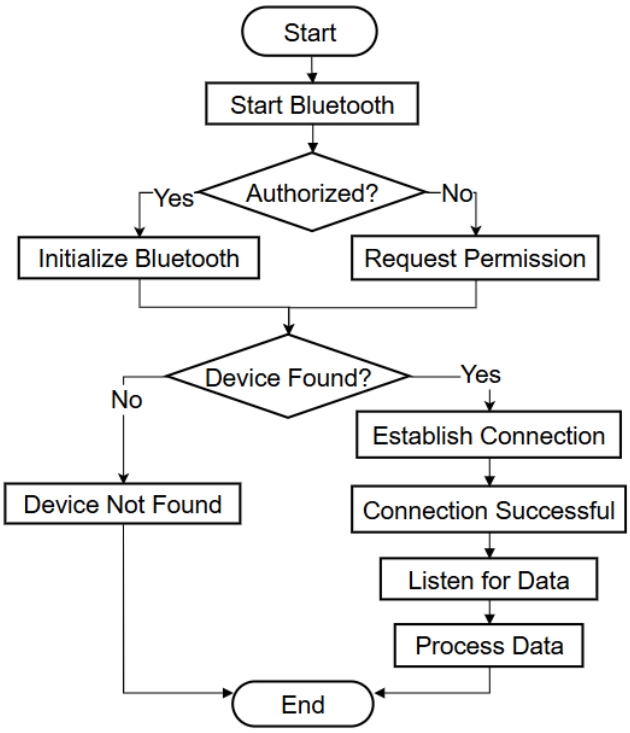

Figure 6. Flow chart of voice recognition and hardware linkage.

Status data parsing and UI synchronization mechanism

To ensure the real-time consistency of software and hardware status, the system designs a three-step parsing method of "frame recognition - field extraction - legality verification". After ConnectedThread receives the original byte stream, it assembles the complete data frame according to the delimiter "#". Taking the frame "S120, T25, B80#" as an example: First locating the keyword "S", extracting the value between it and the first comma as the speed; similarly extracting the temperature after "T" and the power after "B".

The extracted integer value is subjected to range check, and the frame is discarded if it exceeds the threshold to avoid display errors caused by abnormal hardware sensors. The parsed data is encapsulated into a Message object and sent to the main thread through Handler. In the handleMessage callback of the main thread, the UI components are updated: tvSpeedValue displays the speed (km/h), tvTempValue displays the temperature (°C), and tvBatteryValue displays the power (%). At the same time, the Bluetooth connection status (STATE_CONNECTED/STATE_DISCONNECTED) is also transmitted in real time through Message, and the

color of the icon on the main interface changes with the status (green indicates connected, gray indicates disconnected). If the connection is lost, the system automatically initiates reconnection and prompts the user. This mechanism controls the data refresh delay within 500ms, solves the thread safety problem of sub-threads operating the UI, and realizes the smooth visualization of hardware status on the mobile terminal. The Handler mechanism status synchronization is shown in Figure 7.

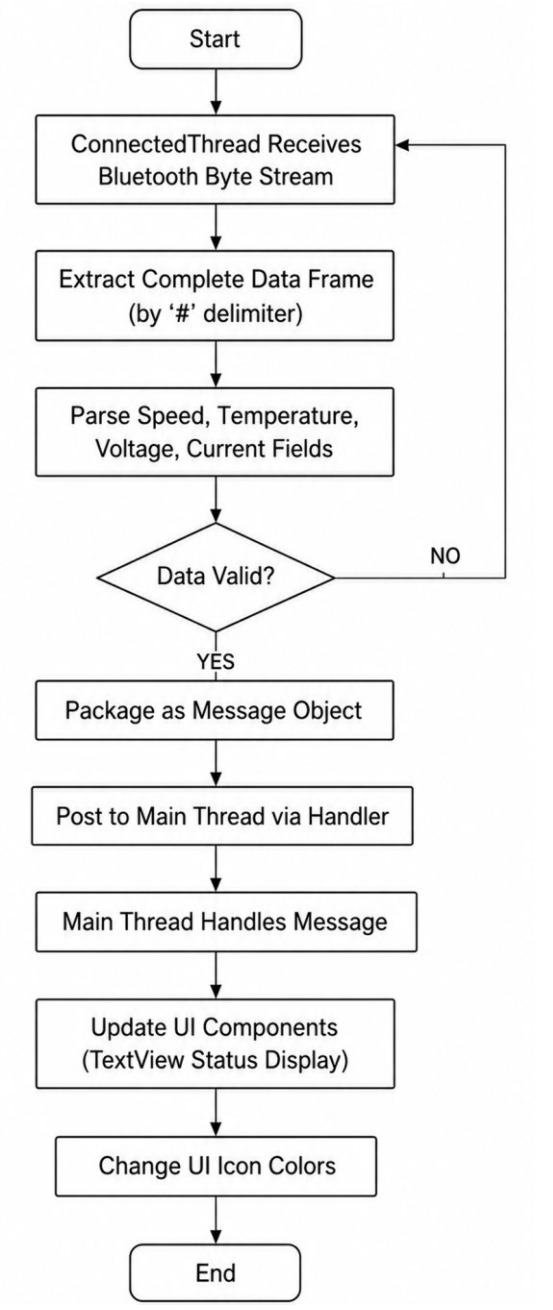

Figure 7. Flow chart of handler mechanism status synchronization.

Conclusion

This paper designs and implements the "Zhiyu Chexing" intelligent vehicle-mounted integrated service system based on the Android platform. Taking the STC51 single-chip microcontroller as the core hardware control unit, the system achieves full-duplex low-latency communication between mobile terminals and embedded devices via the Bluetooth RFCOMM protocol. It further integrates core functions including vehicle status monitoring, voice recognition, audio playback, and map navigation, thereby constructing a complete closed-loop interaction framework of "voice command - function execution - visual feedback".

Experiments show that the system meets the real-time requirements of vehicle-mounted scenarios in key indicators such as communication latency, recognition accuracy and positioning precision, verifying the feasibility of the low-cost and high-integration software and hardware collaborative scheme. In the future, deep learning can be introduced to optimize the robustness of voice recognition, Bluetooth 5.0 can be upgraded to improve communication stability, and vehicle condition parameters such as tire pressure monitoring can be expanded to further enhance the practical value and promotion potential of the system.

Funding

This work was not supported by any funds.

Acknowledgements

The authors would like to show sincere thanks to those techniques who have contributed to this research.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Monk, C., Sall, R., Lester, B. D., Higgins, J. S. (2023) Visual and cognitive demands of manual and voice-based driving mode implementations on smartphones. *Accident Analysis & Prevention*, 187, 107033.

[2] Mohammed, K., Abdelhafid, M., Kamal, K., Ismail, N., Ilias, A. (2023) Intelligent driver monitoring system: an Internet of Things-based system for tracking and identifying the driving behavior. *Computer Standards & Interfaces*, 84, 103704.

[3] Al-Wesabi, F. N., Alshahrani, A., Alanazi, R., Alshahrani, M. M., Sorour, S., Alghamdi, A. M.,

Alamri, M. Z., Alanazi, S. (2025) Integrated communication and location monitoring system for vehicle monitoring via smartphone calls. *Alexandria Engineering Journal*, 128, 1159-1167.

[4] Ma, J., Li, J., Gong, Z. (2022) Evaluation of driver distraction from in-vehicle information systems: a simulator study of interaction modes and secondary tasks classes on eight production cars. *International Journal of Industrial Ergonomics*, 92, 103380.

[5] Prabhakar, G., Biswas, P. (2021) A brief survey on interactive automotive UI. *Transportation Engineering*, 6, 100089.

[6] Mohammed, M. S., Abduljabar, A. M., Faisal, M. M., Mahmmod, B. M., Abdulhussain, S. H., Khan, W., Liatsis, P., Hussain, A. (2023) Low-cost autonomous car level 2: design and implementation for conventional vehicles. *Results in Engineering*, 17, 100969.

[7] Akanda, N. I., Hossain, M. A., Fahad, M. M. I., Rahman, M. N., Khairunnaher, K. (2022) Cost-effective and user-friendly vehicle tracking system using GPS and GSM technology based on IoT. *Indonesian Journal of Electrical Engineering and Computer Science*, 28(3), 1826-1833.

[8] Tan, Z., Dai, N., Su, Y., Zhang, R., Li, Y., Wu, D., Li, S. (2021) Human-machine interaction in intelligent and connected vehicles: a review of status quo, issues, and opportunities. *IEEE Transactions on Intelligent Transportation Systems*, 23(9), 13954-13975.

[9] Menon, V. G., Jacob, S., Joseph, S., Sehdev, P., Khosravi, M. R., Al-Turjman, F. (2022) An IoT-enabled intelligent automobile system for smart cities. *Internet of Things*, 18, 100213.

[10] Murali, P. K., Kaboli, M., Dahiya, R. (2022) Intelligent in-vehicle interaction technologies. *Advanced Intelligent Systems*, 4(2), 2100122.