Custom-Driver-Based UART Bluetooth Home Automation System Using LPC2138 ARM7 Microcontroller with LCD Feedback and Response Latency Evaluation

Vrushabh M. Ukhalkar
Department of Electronics and Telecommunication Engineering Deogiri Institute of Engineering and Management Studies, Chhatrapati Sambhajinagar, Maharashtra, India

***Abstract*—Bluetooth-based home automation provides a simple and low-cost method for short-range wireless control of electrical loads. This paper presents the design and implementation of a custom-driver-based UART Bluetooth home automation system using the LPC2138 ARM7TDMI-S microcontroller, HC-05 Bluetooth module, onboard LED load indicators, manual switches, and a 16×2 LCD display. The proposed system receives ASCII commands from an Android Bluetooth terminal application through the HC-05 Bluetooth module. The received Bluetooth data are converted into UART serial data and processed by the LPC2138 through UART1. Based on the decoded command, the firmware updates GPIO outputs and displays real-time load status on the LCD. Unlike library-dependent platforms, the system is implemented using bare-metal Embedded C with custom UART, GPIO, switch, command-decoder, output-manager, and 4-bit LCD drivers. The firmware also handles invalid commands and filters carriage-return and newline characters transmitted by mobile Bluetooth terminal applications. Prototype-level testing was performed for functional operation, response delay, command reliability, and operating distance. The average command response delay was approximately 56 ms, while command reliability was observed between 95% and 100% during repeated trials. The work demonstrates a compact, low-cost, educational, and reproducible ARM7-based Bluetooth automation prototype with UART protocol justification, LCD feedback, manual override, and response-latency evaluation.**

***Index Terms*—Bluetooth communication, Embedded C, HC-05, home automation, LCD interfacing, LPC2138, UART.**

I. INTRODUCTION

Embedded systems are widely used in automation, control, instrumentation, consumer electronics, and communication applications because they combine hardware, firmware, and real-time decision-making in a compact platform. In home automation, embedded controllers are used to receive user commands, process them, and control output devices such as lamps, fans, indicators, alarms, and other electrical loads. Conventional manual switching is simple, but it does not provide wireless control, flexible operation, or real-time electronic feedback.

Wireless automation improves usability by allowing users to control appliances from a short distance without direct physical access to switches. Among different wireless technologies, Bluetooth is suitable for small-range academic and prototype-level automation because it is low-cost, easy to configure, and supported by most Android smartphones. The HC-05 Bluetooth module is commonly used in embedded applications because it provides transparent serial communication through the Universal Asynchronous Receiver Transmitter (UART) protocol [4].

The LPC2138 microcontroller is based on the ARM7TDMI-S processor core and provides GPIO ports, UART channels, timers, ADC, DAC, SPI, I2C, PWM, and on-chip flash memory. These features make it suitable for embedded control and communication applications [1]–[3]. In the proposed system, UART1 is used for HC-05 Bluetooth communication, while UART0 is reserved for USB-

UART programming and debugging through the CP2102 interface. The board uses a 12 MHz crystal oscillator and is programmed using Keil µVision 5 and Flash Magic.

A 16×2 HD44780-compatible LCD is used as a local feedback display. The LCD is operated in 4-bit mode to reduce GPIO pin usage. In 4-bit mode, each byte is transferred as two 4-bit nibbles. This saves microcontroller pins but requires correct initialization, enable-pulse control, and timing management [5]. In this project, the LCD displays system readiness, Bluetooth/manual mode, bulb status, fan status, and invalid command messages.

The main contribution of this work is not only Bluetooth-based switching, but the development of a custom-driver-based embedded architecture. The firmware is written in Embedded C using direct register-level programming instead of ready-made Arduino-style libraries. This makes the system useful for ARM7 peripheral learning, driver development, hardware debugging, and academic embedded-system demonstration.

II. LITERATURE REVIEW

Smart home automation systems generally consist of a controller, communication interface, user interface, sensors or input devices, and output actuators. Review studies show that embedded devices and IoT-based platforms are widely used to improve convenience, monitoring, safety, and energy management in residential automation [7].

Bluetooth-based home automation has been implemented using Arduino, HC-05 modules, relay boards, and smartphone applications. Khare et al. presented a Bluetooth-controlled home automation system using Arduino and Android-based control [8]. Pal et al. discussed voice-controlled home automation using Arduino and HC-05, showing the usefulness of Bluetooth Serial Port Profile for short-range control applications [9].

Most such systems depend on ready-made development platforms and software libraries. In comparison, ARM7-based systems such as LPC2138 require register-level understanding of GPIO, UART, pin configuration, and peripheral initialization. Fewer prototype-level papers document LPC2138-based Bluetooth home automation with custom Embedded C drivers, LCD feedback, manual mode, command filtering, and response-delay evaluation.

LCD feedback is also important in embedded control. The HD44780-compatible 16×2 LCD supports 8-bit and 4-bit parallel communication. In this work, the LCD is not used only for fixed text; it is used as a real-time status and debugging display for system readiness, active mode, output status, and invalid command conditions.

III. RESEARCH GAP AND MOTIVATION

The literature shows that many Bluetooth home automation projects are based on Arduino or high-level libraries. Such systems are easy to implement, but they give limited exposure to register-level embedded programming. LPC2138/LPC2xxx-specific Bluetooth automation systems using custom UART, GPIO, switch, and LCD drivers are limited.

Many systems demonstrate only ON/OFF control without detailed command validation, LCD status feedback, response-delay measurement, reliability testing, or distance testing. Bluetooth terminal applications may also transmit extra carriage-return and newline characters, which can create invalid-command behavior if not filtered in firmware.

Another gap is protocol justification. HC-05 directly supports UART, whereas I2C is more suitable for short-distance chip-to-chip communication such as RTC, EEPROM, sensors, and I/O expanders [6]. Therefore, a clear UART-over-I2C justification strengthens the technical contribution of the proposed work.

IV. PROPOSED SYSTEM ARCHITECTURE

The proposed system consists of an Android smartphone, HC-05 Bluetooth module, LPC2138 ARM7 microcontroller, onboard LEDs, onboard switches, and a 16×2 LCD display. The Android Bluetooth terminal application sends ASCII commands A, B, C, D, E, and F. These commands are transmitted through the Bluetooth Serial Port Profile, converted into UART serial data by HC-05, received by UART1 of LPC2138, decoded by firmware, and used to update GPIO outputs and LCD feedback.

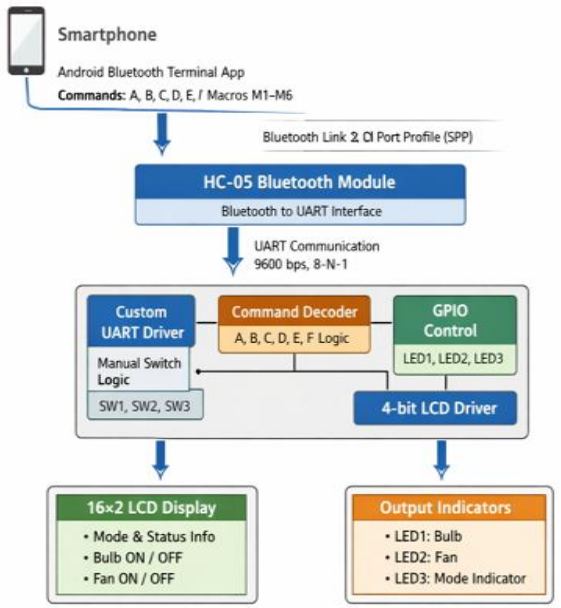

Fig. 1. Main system architecture of Bluetooth-based home automation using LPC2138 and HC-05.

The command path is Smartphone → HC-05 → UART1 → LPC2138 → Command Decoder → GPIO Output Manager → LEDs and LCD Feedback. The firmware is organized into hardware, driver, and application layers. This modular design improves readability, debugging, and portability across similar LPC2xxx boards.

Table I. Hardware and Software Summary

Item	Purpose
LPC2138 ARM7 board	Main controller and GPIO/UART processing unit
HC-05 Bluetooth module	Wireless serial command reception
16×2 LCD	Real-time mode and load-status feedback
Onboard LEDs	Prototype representation of bulb, fan, and mode status
Onboard switches	Manual load control and Bluetooth/manual mode selection
Keil µVision 5	Embedded C coding and HEX file generation
Flash Magic	Programming LPC2138 through UART/CP2102
Proteus 8	Simulation and virtual terminal testing

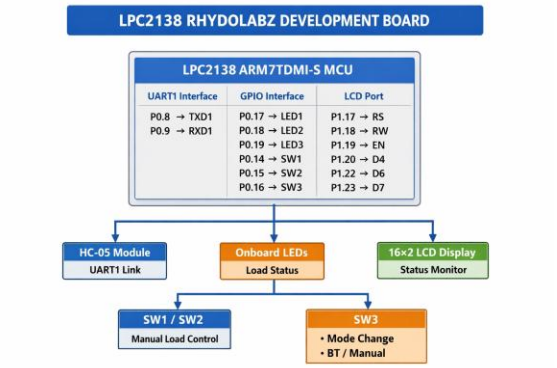

Fig. 2. Hardware interface mapping of LPC2138 RhydoLABZ development board.

V. HARDWARE AND SOFTWARE IMPLEMENTATION

The LPC2138 RhydoLABZ board is used as the main controller platform with a 12 MHz crystal oscillator. The HEX file is generated using Keil µVision 5 [10] and programmed using Flash Magic. Proteus 8 was used for simulation and virtual terminal testing [11]. During programming, the ISP jumper is installed, the programmer switch is kept in manual mode, and reset is pressed correctly to enter ISP mode. After flashing, the ISP jumper is removed and the board is reset again for application execution.

Table II. Important Interface Mapping

Interface	LPC2138 Mapping	Function
HC-05 TXD	P0.9 / RXD1	Bluetooth data to LPC2138
HC-05 RXD	P0.8 / TXD1	Data from LPC2138 to HC-05
LCD RS, RW, EN	P1.17, P1.18, P1.19	LCD control signals
LCD D4-D7	P1.20, P1.21, P1.22, P1.23	4-bit LCD data bus
LED1, LED2, LED3	P0.17, P0.18, P0.19	Bulb, fan, and mode indicators
SW1, SW2, SW3	P0.14, P0.15, P0.16	Manual control and mode selection

The HC-05 interface uses 9600 bps, 8 data bits, no parity, and 1 stop bit. A common ground between the Bluetooth module and LPC2138 board is necessary for reliable UART communication. The LCD is implemented in 4-bit mode using custom command, data, string, clear, and cursor-control functions. The actual LED ON/OFF logic is adjusted according to the active-low or active-high behavior of the board.

VI. WORKING METHODOLOGY

The system first initializes GPIO pins, UART1, LCD, output status variables, and switch inputs. After initialization, the LCD displays system readiness. The firmware then checks the operating mode. In Bluetooth mode, UART1 receives characters from HC-05. In manual mode, SW1 and SW2 are read for local load control. SW3 is used for Bluetooth/manual mode selection.

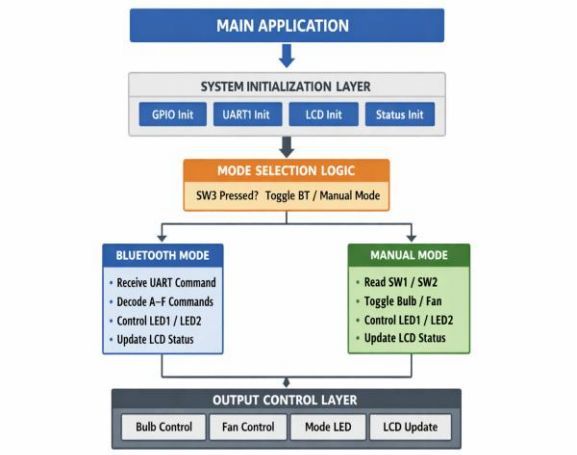

Fig. 3. Main application software flow with Bluetooth and manual modes.

In Bluetooth mode, macros M1 to M6 of the Android Bluetooth terminal application are configured as A, B, C, D, E, and F. The received character is filtered, decoded, and executed. Carriage-return and newline characters are ignored because some terminal applications send A + CR + LF instead of only A.

Table III. Bluetooth Command Mapping

Command	Operation	LCD Feedback
A	Bulb ON	Bulb ON
B	Bulb OFF	Bulb OFF
C	Fan ON	Fan ON
D	Fan OFF	Fan OFF
E	All outputs ON	All ON
F	All outputs OFF	All OFF
CR/LF	Ignored	No change
Other	Invalid command	Invalid Command

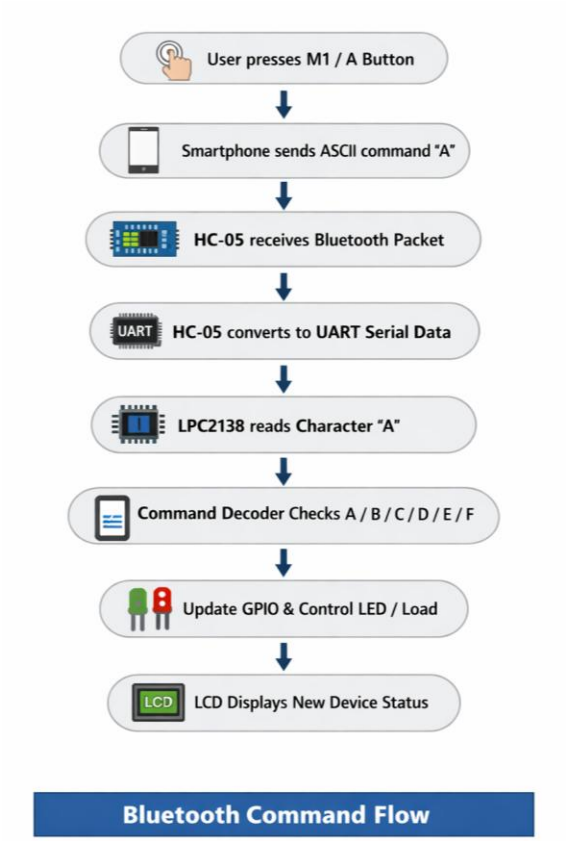

Fig. 4. Bluetooth command flow from smartphone to LPC2138 output control.

Table IV. UART and I2C Comparison

Parameter	UART	I2C
Type	Asynchronous	Synchronous
Lines	TX and RX	SDA and SCL
Clock	Not required	Required
Addressing	Not required	Required
HC-05 support	Directly supported	Not directly supported
Best use	Bluetooth command transfer	RTC, EEPROM, sensors
Project decision	Selected	Not selected

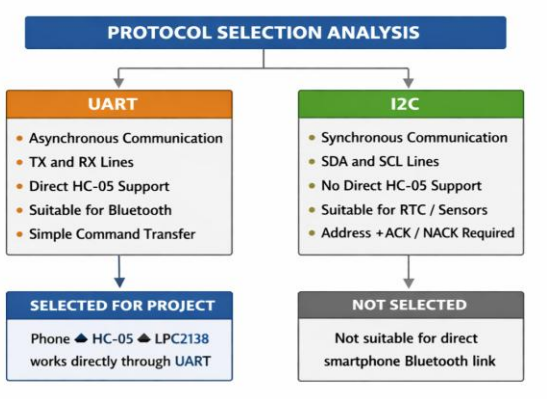

Fig. 5. Protocol selection analysis between UART and I2C for the proposed system.

VII. RESULTS AND PERFORMANCE ANALYSIS

The proposed system was tested in stages. First, a simple onboard LED blinking program was executed using P0.17, P0.18, and P0.19. This verified that the board, Keil-generated HEX file, Flash Magic programming process, and LED hardware were working correctly. Initial programming issues such as failed autobaud, reading signature failure, and verify mismatch were linked to ISP entry, reset timing, COM port selection, and jumper configuration. The major practical solution was to remove the ISP jumper after flashing and reset the board to run the application.

After board verification, the final Bluetooth home automation firmware was tested using HC-05, Android Bluetooth Terminal, LCD, onboard LEDs, and switches. Commands A-F were used for load control, invalid command handling, and status feedback.

Table V. Prototype Testing Summary

Test / Parameter	Observation
Power ON	LCD shows system-ready message
Command A/B	Bulb/load ON and OFF operation passed
Command C/D	Fan/load ON and OFF operation passed
Command E/F	All outputs ON and OFF operation passed
Invalid command	No output change; LCD shows invalid command
CR/LF characters	Ignored by command decoder
Manual mode	SW1/SW2 local control passed
Mode selection	SW3 used for Bluetooth/manual mode change

Table VI. Response Delay and Reliability

Command	Delay	Successful Trials
A - Bulb ON	52 ms	20/20
B - Bulb OFF	55 ms	20/20
C - Fan ON	58 ms	19/20
D - Fan OFF	54 ms	20/20
E - All ON	60 ms	20/20
F - All OFF	57 ms	20/20
Average	≈ 56 ms	95% to 100%

The average response delay of the proposed Bluetooth-based control system was found to be approximately 56 ms. This response time is suitable for a prototype-level human-controlled switching application, especially for basic home automation loads such as bulb and fan control, where millisecond-level precision is not critical. The observed delay includes the time required for command transmission from the Android Bluetooth terminal, reception by the HC-05 module, UART data transfer to the LPC2138, command decoding, GPIO output updating, and LCD status refreshing. Since the system is intended for manual user operation, the measured response delay provides sufficiently fast feedback for practical demonstration and academic prototype use.

The reliability of the command execution was observed between 95% and 100% during repeated trials. Commands A, B, D, E, and F responded successfully in all tested attempts, while command C showed one missed response during testing. This single missed response may be due to temporary Bluetooth link variation, smartphone terminal transmission behavior, UART reception timing, or measurement inconsistency. Therefore, the reliability results are presented as prototype-level experimental observations rather than industrial-grade reliability certification. Overall, the testing confirms that the system performs consistently for short-range wireless switching under normal laboratory conditions.

Distance testing showed stable operation at 1 m, 3 m, and 5 m, with commands being received and executed without noticeable difficulty. At 8 m, the system continued to operate, but a minor delay was observed in some cases. At approximately 10 m, the response became more dependent on practical environmental conditions. Factors such as physical obstacles, wall

interference, smartphone Bluetooth strength, HC-05 antenna orientation, power supply stability, and surrounding electromagnetic noise can affect the communication range and response behavior. Hence, the proposed system is most suitable for small-room, laboratory, and short-range indoor home automation applications.

VIII. ADVANTAGES AND APPLICATIONS

The proposed system is low-cost, does not require Internet connectivity, and is suitable for academic demonstration. It uses a simple Android Bluetooth terminal application for command transmission. The custom Embedded C drivers improve understanding of UART, GPIO, LCD, and switch interfacing. The system provides both Bluetooth and manual control and gives real-time feedback through the LCD.

Possible applications include home automation prototypes, embedded system laboratory experiments, ARM7 peripheral training, Bluetooth communication demonstration, LCD driver learning, wireless switching systems, and academic project and viva demonstrations.

IX. CONCLUSION AND FUTURE SCOPE

This paper presented a custom-driver-based UART Bluetooth home automation system using the LPC2138 ARM7TDMI-S microcontroller, HC-05 Bluetooth module, 16×2 LCD display, onboard LEDs, and onboard switches. The system successfully demonstrated wireless command reception from an Android smartphone, UART-based communication, command decoding, GPIO output control, manual switch operation, and real-time LCD feedback.

The firmware was developed using bare-metal Embedded C with custom UART, GPIO, LCD, switch-handler, command-decoder, and output-manager modules. The system also handled practical issues such as ISP jumper configuration, Flash Magic programming procedure, carriage-return/newline filtering, and LED logic correction. Experimental testing showed an average Bluetooth command response delay of approximately 56 ms and command reliability between 95% and 100% under prototype conditions.

Future improvements may include relay-driver interfacing for real appliances, opto-isolation for electrical safety, password-based authentication, a dedicated Android GUI application, interrupt-based UART reception, sensor integration, RTC/EEPROM interfacing using I2C, Wi-Fi/IoT upgrade, PCB design, enclosure development, and power optimization.

REFERENCES

[1] NXP Semiconductors, “LPC2131/32/34/36/38 single-chip 16/32-bit microcontrollers; 32/64/128/256/512 kB ISP/IAP flash with 10-bit ADC and DAC,” Datasheet, Rev. 05, 2011.

[2] NXP Semiconductors, “UM10120: LPC2131/2/4/6/8 user manual,” User Manual, 2002.

[3] ARM Ltd., “ARM7TDMI technical reference manual,” Technical Reference Manual, Revision 3.

[4] Guangzhou HC Information Technology Co., “HC-05 Bluetooth to serial port module,” Datasheet / AT command documentation.

[5] Hitachi, “HD44780U dot matrix liquid crystal display controller/driver,” Datasheet, 1999.

[6] NXP Semiconductors, “UM10204: I2C-bus specification and user manual,” User Manual, Rev. 7, 2021.

[7] A. Al Tareq, M. R. Mostofa, M. J. Rana, and M. S. Rahman, “A comprehensive review of intelligent home automation systems using embedded devices and IoT,” Control Systems and Optimization Letters, vol. 2, no. 2, pp. 198-203, 2024.

[8] P. Khare, Gangacharan, A. Singh, S. Mishra, and A. Kumar, “Research paper on Bluetooth-controlled home automation using Arduino,” International Journal of Novel Research and Development, vol. 9, no. 4, pp. i276-i280, Apr. 2024.

[9] V. Pal, S. Kumar, V. Kumar, G. Jha, and P. Pant, “Voice controlled home automation system using Arduino,” IITM Journal of Management and IT, vol. 12, no. 1, pp. 1-7, 2021.

[10] Arm Keil, “Keil µVision IDE and MDK-Arm documentation,” Technical Documentation.

[11] Labcenter Electronics, “Proteus Design Suite documentation,” Technical Documentation.