Morse Code Communication System

K B Ramesh[1], Shreya Srivastava[2], Esha Sharma[3], Yashmitha Desai[4]
[1]*Dept. of EIE, R V College of Engineering, Bengaluru, India*
[2,3,4] *Dept of CSE-CY, R V College of Engineering, Bengaluru, India*

***Abstract*—This paper describes a Morse code transmitter using the LPC2148 ARM7 microcontroller. The system converts button-pressed characters into Morse code, displayed via LED blinks, buzzer sounds, and an LCD. It also decodes Morse back to text and supports voice recognition for accessibility. Additional features like EEPROM storage and wireless communication allow future expansion. Designed in Keil and Proteus, this project serves as an educational tool for embedded systems and digital communication.**

***Index Terms*—Morse Code, embedded systems, microcon-troller, digital communication, accessibility**

I. INTRODUCTION

Morse code was discovered in the early nineteenth century and was used to transfer messages over the telegraph. It still remains a fundamental form of communication in critical times due to its simplicity and low bandwidth. With evolution of embedded systems, it brings an opportunity to explore the applications of Morse Code system as a hand-held device which can be made use of as an assistive tool or other practical applications. This project leverages the LPC2148 ARM7 micro-controller, programmed using Keil µVision and simulated through Proteus software to demonstrate a compact version of Morse code transmitter. The system facilitates character input through a keypad, converts it into Morse code, and relays the encoded signal through an LED, buzzer, and LCD display for multifaceted feedback. To further enhance interactivity, Morse-to-text decoding has been incorporated, allowing received signals to be interpreted back into charac-ters. Voice integration adds onto this allowing user to encode their spoken words. Furthermore, provisions for EEPROM storage, UART communication, and future wireless transmis-sion extend the applicability of the system beyond simulation to real-time deployment. This paper explores the architecture, implementation and future potential of the system, particularly in areas such as education, assistive communication, and emergency response systems.

II. OBJECTIVE

The primary objective of this research is to design and simulate a Morse code transmission system using the LPC2148 ARM7 micro-controller, focusing on efficient text-to-morse conversion, signal generation and making the system compact

. The study aims to develop a reliable embedded system capable of generating both visual and auditory Morse code outputs in the form of LED and buzzer with precise timing control. Additionally, the project explores different simulation techniques in Proteus software to validate the usefulness of the system before hardware implementation, ensuring proper accuracy in dots and dashes durations, UART communication, and correctly displaying output. It also evaluates performance data on metrics such as conversion speed, power efficiency, and resource utilization to optimize the system for real world applications like emergency signaling, accessibility device for the disabled.

III. BACKGROUND

Morse code, developed in the 1830s, remains a vital com-munication method in scenarios where traditional transmission systems fail, such as military operations, maritime signaling, and amateur radio. With the advancement of embedded sys-tems, microcontrollers such as the LPC2148, equipped with GPIO, timers, and UART, provide an efficient platform for the generation of Morse code. Previous implementations have used various microcontrollers, but the LPC2148 ARM7 architecture offers enhanced processing power, low power consumption, and

flexible peripheral integration, making it ideal for real time Morse code applications. Simulation tools like Proteus enable virtual testing of the system, reducing development costs and allowing for rapid prototyping. This research builds on existing Morse code transmission methods while introduc-ing optimizations in timing accuracy, signal modulation, and simulation-based validation.

IV. SYSTEM DESIGN

The Morse Code Communication System is built on the LPC2148 ARM7 microcontroller, which was chosen for its high processing performance, low power consumption, and flexibility. Because of the system's modular design, the in-put, processing, and output components can be seamlessly integrated to execute real-time Morse code encoding and decoding. The following is a discussion of key ideas.

A. Hardware Architecture

The hardware setup includes the following components:

- 4x4 Matrix Keypad: Used as input devices, each button corresponds to a specific number (0-9).
- LED Indicator: Used to visually represent Morse code sequences.
- Buzzer: Provides audio feedback, beeping in sync with the LED to mirror the Morse timing.
- 16x2 LCD Display: Displays both the input characters and their Morse equivalents in real time.
- LCD Display Handling: The entered character is dis-played on the first LCD screen. Simultaneously, its Morse code equivalent is shown on the second LCD in real-time as it is processed.

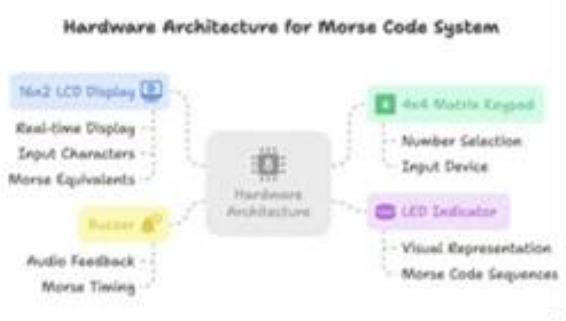

Fig. 1. Hardware Components for proposed solution

B. Software Design

The software architecture of the Morse Code Communica-tion System is created to efficiently manage input processing, signal encoding, and output generation using the LPC2138 ARM7 microcontroller. Developed in Embedded C within the Keil Vision environment, the architecture is modular, allowing for flexibility, scalability, and ease of debugging.

- System Initialization: Upon startup, the system performs the following initialization routines:

– GPIO Configuration: GPIO pins are configured us-ing the PINSEL and IODIR registers. Rows of the keypad and output pins for the LED (e.g., P0.0) and buzzer (e.g., P0.1) are set as outputs.

– LCD Initialization: The LCDs are configured via the I2C protocol using driver functions. Initialization includes setting display mode, clearing the display, and setting the cursor position.

– I2C Communication Setup: The I2C modules (I2C0 and I2C1) are initialized using appropriate frequency settings to enable communication with LCDs and other I2C peripherals.

– Buzzer Initialization: The buzzer pin is set as a digital output. It is controlled using IOSET and IOCLR instructions during Morse signal generation.

- Input Handling and Debouncing: The system continu-ously scans the 4x4 keypad using a row-column matrix method. Debouncing is handled in software using delay loops to prevent false triggering. Each key press is converted to a digit using a lookup table.
- Character-to-Morse Encoding: Each valid input character is mapped to its corresponding Morse code using a switch-case based lookup function. The Morse code is stored as a string of dots and dashes.
- Signal Generation Logic: The Morse code string is parsed symbol by symbol. For a dot, the LED and buzzer are turned on briefly. For a dash, they are activated for a longer duration. Proper delays are introduced between symbols and characters to maintain timing standards.

V. PROTOTYPE AND IMPLEMENTATION

The Morse Code Communication System prototype was developed using Keil Vision for embedded software devel-opment and Proteus Design Suite for simulating the hardware configuration. The implementation process focused on efficient code design and accurate signal timing. This section describes the complete prototyping methodology, development tools, simulation strategies, and feature integration.

• Development Environment and Tools:

– Microcontroller: LPC2138 ARM7TDMI-S, chosen for its high-speed processing, low power consumption, and flexible peripheral interfaces.[2]
– IDE: Keil Vision 5 was used for embedded C code development, compilation, and debugging.
– Simulation: Proteus 8 Professional enabled hardware simulation, facilitating virtual testing of the micro-controller's interaction with LEDs, buzzer, LCD, push buttons.

• Implementation Workflow: The implementation was di-vided into distinct stages, each addressing one subsystem of the overall design.

– Keypad input and character mapping: A 4x4 matrix keypad was interfaced with the LPC2148 microcon-troller using GPIO pins. Rows were set as outputs and columns as inputs. The system continuously scanned the keypad using polling, and each valid key press was assigned a numeric character (0–9) through a lookup array. The debouncing was handled in software using simple delay-based filtering to eliminate mechanical switch noise.
– Text-to-Morse Encoding: Each digit obtained from the keypad was converted into Morse code using a switch-case based lookup function. The correspond-ing Morse code string, consisting of dots and dashes, was parsed one symbol at a time. For a dot, the LED and buzzer were turned ON for a short duration. For a dash, they were activated for a longer duration. Delays were implemented using software delay loops to maintain consistent intra-symbol and inter-symbol timing.
– Morse Signal Output via LED and Buzzer: The LED connected to P0.0 and the buzzer to P0.1 were used for Morse code output. A dot was represented by a 200 ms ON signal, and a dash by a 600 ms ON signal. Delays between symbols (200 ms), characters (600 ms), and words (1400 ms) were included to match standard Morse timing. The signal output logic was implemented in the 'led ()' function, which handled both LED and buzzer activation.
– LCD Output Display: Two I2C-based 16x2 LCDs were used for visual feedback. The first LCD displayed the input character from the keypad, while the second displayed its Morse code equivalent. LCDs were initialized via the I2C protocol using custom 'sendchar0()' and 'sendchar1()' functions for communication. This dual-display system improved user understanding by providing both character and Morse representations.

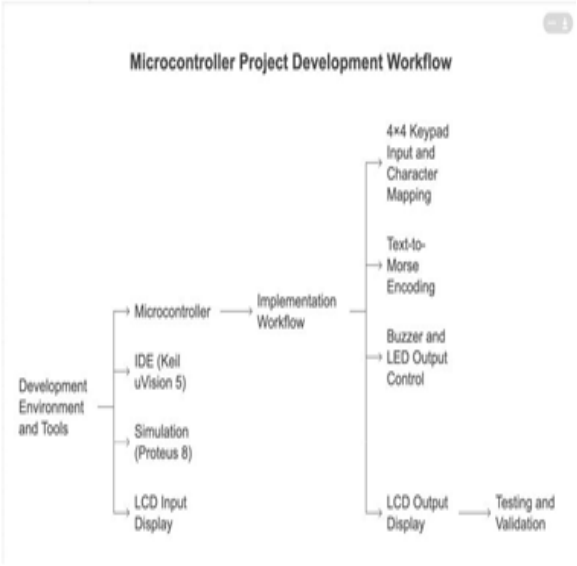

Fig. 2. Implementation flowchart

– Testing and Validation: Each functional module was independently verified in Proteus simulation. The keypad was tested for correct character mapping and debounce reliability. LCD outputs were monitored for accuracy in both character and Morse display. LED and buzzer output timing was validated us-ing virtual oscilloscopes. UART communication was tested for reliable data transfer. Optional modules, including voice input and Morse decoding, were simulated via UART and validated for correct func-tionality under different test cases, including rapid keypresses and varied pulse widths.

VI. RESULTS AND DISCUSSIONS

The Morse Code Communication System was successfully developed and validated through simulation using the Proteus Design Suite and programming in Keil Vision. The integration of hardware and software components, including the 4x4 keypad, I2C LCD, buzzer, and LED, demonstrated accurate and synchronized Morse code transmission.

A. Keypad Input and Morse Encoding Accuracy

All keys from the 4x4 matrix keypad were successfully mapped to numeric inputs (0–9). The system reliably detected and debounced keypresses using software delays. Each key-press triggered the correct Morse code sequence based on the implemented lookup logic, confirming accurate character-to-Morse conversion.

B. LED and Buzzer Synchronization

LED and buzzer outputs were synchronized precisely ac-cording to Morse timing rules:

- Dot: 200 ms ON
- Dash: 600 ms ON
- Intra-symbol delay: 200 ms
- Inter-character delay: 600 ms
- Inter-word delay: 1400 ms

Oscilloscope analysis in Proteus confirmed the timing consis-tency, with clear distinctions between dots and dashes. The buzzer tone effectively mirrored the LED flashes, providing accurate auditory feedback.

C. LCD Output Verification

Two I2C LCDs were used: one displayed the character entered, and the other displayed its Morse equivalent. The displays updated in real-time and remained stable across multiple keypresses. The custom driver functions ‘sendchar0() ‘and ‘sendchar1() ‘ensured smooth I2C communication and correct command/data transmission.

D. Simulation Results in Proteus

The complete setup was simulated successfully in Proteus. Inputs from the virtual keypad triggered all corresponding events—Morse conversion, LED/buzzer blinking, and LCD updates. The simulation allowed for stepwise debugging of logic, verifying signal timing, and testing edge cases (e.g., rapid keypresses or invalid inputs).

E. System Robustness

Stress testing included:

- Rapid sequential keypresses to evaluate debounce logic.
- Long simulation runs to test LED/buzzer stability.
- Multiple simultaneous signals to observe LCD perfor-mance.

All modules performed reliably with minimal timing drift and no functional errors observed.

F. Limitations and Future Enhancements

Although the current prototype functions effectively in simulation, some limitations exist:

- Hardware testing is pending for real-world validation.
- Limited character set support (only 0–9) in current im-plementation.
- Real-time decoding and bidirectional communication not yet implemented.

Future improvements may include:

- Support for full alphanumeric character set.
- Real-time Morse-to-text decoding using GPIO signal analysis.
- Integration with wireless modules (e.g., Bluetooth, RF) for remote transmission.
- Compact PCB design for standalone usage in emergency or assistive communication devices.

REFERENCES

[1] Rajpure, R., Pandekar, P., and Shinde, S., “Morse Code Communication System Using Microcontroller,” International Journal of Microelectron-ics and Digital Integrated Circuits, Vol. 10, Issue 1, pp. 10–17, June 2024.

[2] X. Wei, Z. Li, and S. Han, “YFDM: YOLO for detecting Morse code,” Sci. Rep., vol. 13, no. 1, Art. 20614, 2023.

[3] F. Xiao, J. Mu, L. He, and Y. Wang, “How to use one surface electromyo-graphy sensor to

recognize six hand movements for a mechanical hand in real time: a method based on Morse code," Med. Biol. Eng. Comput., vol. 62, no. 9, pp. 2825–2838, Sep. 2024.

[4] A. Penubarthi, S. M. Sai Supreeth, and G. Sivashankar, "Eye blink-based Morse code detection and real-time translation into text," Int. J. Eng. Res. Comput. Sci. Eng., vol. 11, no. 12, pp. 30–35, 2024.

[5] N. Tarek et al., "Morse Glasses: an IoT communication system based on Morse code for users with speech impairments," Computing, vol. 104,
pp. 789–808, 2022.

[6] V. Stangaciu, C. Stangaciu, B. Gus¸ita,˘ and D.-I. Curiac, "Integrating real-time wireless sensor networks into IoT using MQTT-SN," J. Network Syst. Manage., vol. 33, art. 37, 2025.

[7] K. Niu, F. Zhang, Y. Jiang, W. Yi, and W. Zhao, "WiMorse: a contactless Morse code text input system using ambient WiFi signals," in Proc. IEEE INFOCOM Workshop on Machine Learning for Communications, 2022,
pp. 1–6.

[8] M. Mathankumar and P. Thirumoorthi, "DRDE: Dual run distribution-based encoding scheme for sustainable IoT applications," IEEE Access, vol. 11, pp. 170–187, 2023.

[9] I. Kumaraguru et al., "Implementation of Morse code in underwater acoustic communication," in Proc. 2023 Int. Conf. Advances in Intell. Communication and AI, 2023, pp. 1–5.

[10] P. Seshadri and S. Dananjayan, "Hand detection and Morse code translation for alternative communication," in Proc. 2023 Int. Conf. Innov. Syst. Embedded Comput. (ICISEC), 2023, pp. 1–6.

[11] K. V. Singh et al., "CNN-based personal assistive system for deaf-blind individuals," in Proc. 2023 IEEE Int. Conf. Signal Inf. Process. Applic. (ICSIPA), 2023, pp. 1–5.

[12] R. Kumar and P. Singh, "Design and implementation of a smart home system using LPC2148 microcontroller," in Proc. 2022 Int. Conf. Commun., Electron. Embedded Syst. (ICCEES), 2022, pp. 1–6.

[13] B. Gupta, S. Sharma, and N. Patel, "LPC2148-based wireless sensor design for environmental monitoring," IEEE Trans. Instrum. Meas., vol. 71, 2022, Art. 125012.