Design and Implementation of a Scientific Calculator

EE24BTECH11008 - Aslin Garvasis

Abstract

This report details the design and implementation of a scientific calculator using an Arduino Uno microcontroller. The system is designed to perform arithmetic, trigonometric, and inverse trigonometric operations. A minimalistic 15-button keypad is utilized, where one button cycles through all arithmetic operations, another cycles through all trigonometric and inverse trigonometric functions, and additional buttons are assigned for numerical inputs, enter (‘=‘), clear (‘C‘), and decimal point (‘.‘). The interface is implemented using an LCD display for output and push buttons for input selection. The system is programmed using AVR-GCC with efficient code execution techniques to ensure responsiveness.

I. Introduction

Scientific calculators are essential tools in engineering and mathematics. This project implements a scientific calculator using an Arduino Uno, focusing on cost-effectiveness, minimal hardware, and efficient software techniques. The objectives include:

- Implementing a compact user interface with minimal buttons.
- Efficiently handling arithmetic and trigonometric operations.
- Optimizing input interpretation to reduce hardware complexity.
- Ensuring quick response time using efficient programming techniques.

II. Hardware Components

The components used in the system are listed in Table I.

TABLE I
Hardware Components

S.No	Component	Quantity
1	Arduino Uno (ATmega328P-based)	1
2	16x2 LCD Display	1
3	Breadboard	1
4	USB A to USB B Cable	1
5	Jumper Wires	20
6	Resistors 10 kΩ	10
7	Push Buttons	15
8	10k Potentiometer	1

III. Circuit Design and Connections

The circuit is designed to optimize GPIO pin usage with a reduced number of input buttons:

- The LCD display is connected using a 4-bit interface to Arduino digital pins.
- A single button is used for all arithmetic operations (addition, subtraction, multiplication, division), cycling through each with multiple presses.
- Another button is used for trigonometric and inverse trigonometric functions, cycling through sine, cosine, tangent, and their inverses.
- Additional buttons include: numerical inputs (0-9), enter (‘=‘), clear (‘C‘), and decimal (‘.‘).

IV. Software Implementation

The firmware is developed in C using AVR-GCC. The software architecture consists of:

A. Input Handling

- The program scans button presses and determines the intended operation.
- A state machine cycles through arithmetic and trigonometric operations based on repeated presses of the respective button.

B. Arithmetic and Trigonometric Computation

- Arithmetic operations (`+,-,*,/`) are computed using integer and floating-point operations.
- Trigonometric functions utilize precomputed lookup tables to optimize execution speed.
- Inverse trigonometric calculations rely on standard AVR math library functions.

C. LCD Display Updates

- The LCD dynamically updates to reflect selected operations and computed results.
- Optimized character rendering ensures smooth transitions between screens.

V. Code Explanation

The Arduino code consists of:

- **Setup Phase:** Initializes the LCD, configures button pins as inputs, and displays an initial message.
- **Loop Handling:** Continuously scans for button presses and updates input accordingly.
- **Arithmetic Operations:** A single button cycles through ‘+‘, ‘-‘, ‘*‘, ‘/‘, and ‘=‘.
- **Trigonometric Operations:** Another button cycles through ‘sin‘, ‘cos‘, ‘tan‘, ‘asin‘, ‘acos‘, and ‘atan‘.
- **Decimal Handling:** A separate button allows floating-point calculations.
- **Enter Function:** The ‘=‘ button evaluates expressions and displays results.

- **Clear Function:** Pressing ‘C‘ resets the input string.

The program ensures responsive operation using **debounce techniques** to prevent unintended button presses. The evaluation logic converts user inputs into mathematical expressions and computes results using AVR-GCC math functions.

VI. Results and Observations

The implemented system successfully computes and displays results for all supported operations. The input system effectively cycles through operations using minimal buttons, reducing hardware complexity.

VII. Future Improvements

Potential enhancements include:

- Expanding functionality with additional mathematical operations.
- Implementing a graphical LCD for improved visualization.
- Adding a memory storage feature for saving and recalling results.

VIII. Conclusion

This project demonstrates a compact and efficient scientific calculator implementation using an Arduino Uno. The use of a minimal input interface without compromising functionality highlights the effectiveness of optimized hardware and software integration.

IX. References

The code used in this project is based on the implementation by Aravind (EE24BTECH11038).