Scientific Calculator Implementation on Arduino Microcontroller

Balaji B

March 24, 2025

Abstract

This report details the design and development of a scientific calculator powered by an Arduino microcontroller. The calculator supports a wide range of mathematical operations, including trigonometric, logarithmic, and exponential functions, along with numerical methods for solving differential equations. To ensure high precision, the implementation utilizes the fourth-order Runge-Kutta method for computations.

Contents

1 Introduction

The goal of this scientific calculator project is to execute various mathematical functions using an Arduino microcontroller with constrained computational power. Rather than using built-in math libraries, numerical methods have been employed to calculate functions such as sine, cosine, logarithms, and exponentials.

2 Hardware Components

- Arduino Microcontroller
- 16x2 LCD Display
- 4x5 Matrix Keypad for input
- Additional electronic components for support

3 Mathematical Methods

3.1 Runge-Kutta 4th Order Method

The Runge-Kutta 4th order method (RK4) is used to solve ordinary differential equations (ODEs). It provides a good balance between accuracy and computational complexity. For a first-order ODE of the form:

$$\frac{dy}{dx} = f(x, y) \tag{1}$$

The RK4 method approximates the solution using:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2}$$

$$k_1 = h \cdot f(x_n, y_n) \tag{3}$$

$$k_2 = h \cdot f(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \tag{4}$$

$$k_3 = h \cdot f(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \tag{5}$$

$$k_4 = h \cdot f(x_n + h, y_n + k_3) \tag{6}$$

where h is the step size.

3.2 Computing Logarithmic Functions

The natural logarithm is computed using the integral definition:

$$\ln(x) = \int_1^x \frac{1}{t} dt \tag{7}$$

This is solved using the RK4 method with the differential equation:

$$\frac{dy}{dx} = \frac{1}{x} \tag{8}$$

3.3 Computing Exponential Functions

The exponential function e^x is computed by solving the ODE:

$$\frac{dy}{dx} = y \tag{9}$$

With the initial condition $y(0) = 1$.

3.4 Computing Trigonometric Functions

Sine and cosine functions are computed by solving the second-order ODE:

$$\frac{d^2y}{dx^2} = -y \tag{10}$$

This is converted to a system of first-order ODEs:

$$\frac{dy_1}{dx} = y_2 \tag{11}$$

$$\frac{dy_2}{dx} = -y_1 \tag{12}$$

With initial conditions:

- For sine: $y_1(0) = 0, y_2(0) = 1$
- For cosine: $y_1(0) = 1, y_2(0) = 0$

3.5 Fast Inverse Square Root

For computing inverse square root $\frac{1}{\sqrt{x}}$, we use an optimized algorithm that provides a good approximation:

$$y_{n+1} = y_n \cdot \left(\frac{3 - x \cdot y_n^2}{2} \right) \tag{13}$$

4 Implementation Details

4.1 Code Structure

The code is structured into distinct functional modules, each responsible for a specific aspect of the implementation:

- **Numerical Methods:** Implements algorithms for mathematical computations.
- **LCD Interface:** Manages communication with the LCD display.
- **Keypad Interface:** Handles user input through the keypad.

4.2 Numerical Methods Implementation

The Runge-Kutta fourth-order (RK4) method is implemented to solve both first-order and second-order differential equations efficiently.

`https://github.com/Balaji29-code/Hardware_Project/blob/main/scientific_calculator/codes/range_kutta1.c`

`https://github.com/Balaji29-code/Hardware_Project/blob/main/scientific_calculator/codes/range_kutta2.c`

4.3 User Interface

The calculator supports three operational modes, each offering different functionalities:

- **Normal Mode:** Performs basic arithmetic operations.
- **Alpha Mode:** Provides access to scientific functions such as sine, cosine, and logarithm.
- **Shift Mode:** Enables advanced functions, including inverse trigonometric calculations and memory operations.

5 Mathematical Functions

The calculator implements the following mathematical functions:

5.1 Power and Root Functions

- $\sqrt{x}$: Computed using RK4 for the ODE $\frac{dy}{dx} = \frac{1}{2y}$ with initial condition $y(1) = 1$
- x^y: Computed using iterative multiplication
- $\sqrt[3]{x}$: Computed using Newton's method

5.2 Logarithmic and Exponential Functions

The calculator implements logarithmic and exponential functions using numerical methods:

- **Natural Logarithm** ($\ln(x)$): Computed via numerical integration of $\frac{1}{x}$.
- **Base-10 Logarithm** ($\log_{10}(x)$): Evaluated using the relation $\log_{10}(x) = \frac{\ln(x)}{\ln(10)}$.
- **Exponential Function** (e^x): Solved using the Runge-Kutta fourth-order (RK4) method for the differential equation $\frac{dy}{dx} = y$ with the initial condition $y(0) = 1$.
- **Power of 10** (10^x): Computed using the general power function.

5.3 Trigonometric Functions

The calculator computes trigonometric and inverse trigonometric functions using numerical methods:

- **Sine** ($\sin(x)$): Solved using the Runge-Kutta fourth-order (RK4) method for the differential equation $\frac{d^2y}{dx^2} = -y$ with initial conditions $y(0) = 0$, $y'(0) = 1$.
- **Cosine** ($\cos(x)$): Computed using RK4 for the equation $\frac{d^2y}{dx^2} = -y$ with initial conditions $y(0) = 1$, $y'(0) = 0$.
- **Tangent** ($\tan(x)$): Evaluated as $\frac{\sin(x)}{\cos(x)}$.
- **Inverse Sine** ($\sin^{-1}(x)$): Determined via numerical integration of $\frac{1}{\sqrt{1-x^2}}$.
- **Inverse Cosine** ($\cos^{-1}(x)$): Computed using the identity $\cos^{-1}(x) = \frac{\pi}{2} - \sin^{-1}(x)$.
- **Inverse Tangent** ($\tan^{-1}(x)$): Calculated using numerical integration of $\frac{1}{1+x^2}$.

6 Conclusion

The implementation of the scientific calculator showcases how complex mathematical functions can be efficiently computed on resource-limited microcontrollers using numerical methods. The use of the Runge-Kutta method allows for accurate approximations of differential equations, enabling the calculation of transcendental functions with high precision.

6.1 Future Improvements

Potential enhancements for the scientific calculator include:

- Expanding functionality to support advanced operations such as hyperbolic and statistical functions.
- Enhancing the expression parser to accommodate more complex mathematical expressions.
- Optimizing code efficiency for improved performance and reduced memory usage.

- Integrating graphing capabilities for visualizing functions and equations.

7 Full Source Code

The complete source code for the scientific calculator is available in the project repository.

https://github.com/Balaji29-code/Hardware_Project/tree/main/scientific_calculator/codes