Hardware Assignment-2

EE24BTECH11003 - Akshara Sarma Chennubhatla

March 2025

Assignment:
To make a calculator
with implementation of some
scientific functions and
regular BODMAS operations

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Bachelor of Technology

Department of Electrical Engineering

Contents

1 Introduction

This document details the hardware connections, working principles, and implementation of an Arduino-based calculator. The calculator supports arithmetic operations, scientific functions, and features a scrolling mechanism for selecting trigonometric and logarithmic functions.

2 Hardware Connections

2.1 Microcontroller

- Arduino Uno

2.2 Display

A 16x2 LCD is used for displaying input and results. The connections are as follows:

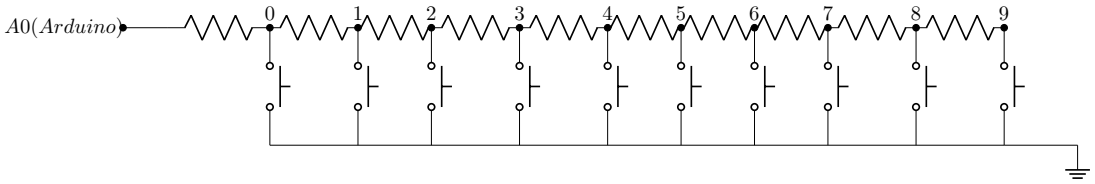
- RS: Pin 2
- Enable: Pin 3
- D4, D5, D6, D7: Pins 4, 5, 6, 7

2.3 10 -> 1 : Resistor Ladder

The calculator uses a resistor ladder to read multiple button inputs for digits from 0-9 using a **SINGLE** analog pin. The principle behind this approach is as follows:

- Each button is connected in series with resistors, forming a voltage divider.
- When a button is pressed, it changes the resistance in the circuit, altering the voltage read by the analog pin.
- The Arduino reads this analog value and maps it to a specific button.

This method reduces the number of pins required compared to connecting each button to a separate digital pin. The circuit of the implementation is as follows:



2.4 Power Supply

The Arduino can be powered via USB or an external 5V power supply.

3 Circuit Image

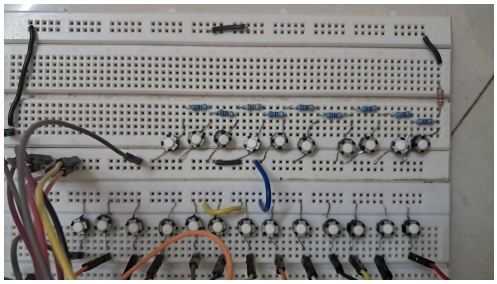


Figure 1: Result-2

4 Working of the Calculator

- Uses push buttons as input.
- LCD displays numbers and operations.
- Supports basic arithmetic $(+, -, \times, \div)$ and scientific functions $(\sin, \cos, \tan, \ln, \log_{10}, pow())$.
- Includes a temporary answer storage.
- Implements operator precedence using stacks.

5 Functions and Implementation

5.1 Stack Operations

Stacks are used for handling operator precedence and evaluating expressions.

- ‘initStack()‘, ‘initNumStack()‘, ‘push()‘, ‘pop()‘, ‘peek()‘, ‘isEmpty()‘ for operator stack.
- ‘pushNum()‘, ‘popNum()‘ for numerical evaluations.

5.2 Expression Evaluation

- Converts infix expressions to postfix using the **Shunting Yard Algorithm**.
- Evaluates postfix expressions using a number stack.
- Handles exponentiation using 'pow()'.
- Implements a parsing function to read input expressions character by character.
- Detects operator precedence dynamically.
- Uses a loop to traverse the expression and determine operand boundaries.

5.3 Trigonometric functions Using RK4

- Trigonometric functions (sin, cos, tan) are implemented using the Runge-Kutta 4th Order Method (RK4).

5.3.1 Runge-Kutta 4th Order Method for $\sin(x)$ Using a Second-Order Differential Equation

The function $\sin(x)$ satisfies the second-order differential equation:

$$\frac{d^2y}{dx^2} + y = 0. \tag{1}$$

To solve this using the Runge-Kutta 4th Order Method (RK4), we rewrite it as a system of first-order equations. Let:

$$y_1 = y = \sin(x), \tag{2}$$

$$y_2 = \frac{dy_1}{dx} = \cos(x). \tag{3}$$

Thus, the system becomes:

$$\frac{dy_1}{dx} = y_2, \tag{4}$$

$$\frac{dy_2}{dx} = -y_1. \tag{5}$$

Applying RK4, we calculate four intermediate values for each step:

$$k_1^1 = hy_2, \tag{6}$$
$$k_1^2 = h(-y_1), \tag{7}$$
$$k_2^1 = h\left(y_2 + \frac{k_1^2}{2}\right), \tag{8}$$
$$k_2^2 = h\left(-\left(y_1 + \frac{k_1^1}{2}\right)\right), \tag{9}$$
$$k_3^1 = h\left(y_2 + \frac{k_2^2}{2}\right), \tag{10}$$
$$k_3^2 = h\left(-\left(y_1 + \frac{k_2^1}{2}\right)\right), \tag{11}$$
$$k_4^1 = h\left(y_2 + k_3^2\right), \tag{12}$$
$$k_4^2 = h\left(-\left(y_1 + k_3^1\right)\right). \tag{13}$$

The updated values for the next step are:

$$y_1^{n+1} = y_1^n + \frac{1}{6}(k_1^1 + 2k_2^1 + 2k_3^1 + k_4^1), \tag{14}$$
$$y_2^{n+1} = y_2^n + \frac{1}{6}(k_1^2 + 2k_2^2 + 2k_3^2 + k_4^2). \tag{15}$$

Since $y_1 = \sin(x)$ and $y_2 = \cos(x)$, this method numerically solves $\sin(x)$ with high accuracy.

To calculate $\cos(x)$ we calculate $\sin(\frac{\pi}{2} - x)$
To calculate $\tan(x)$ we calculate $\frac{\sin(x)}{\sin(\frac{\pi}{2} - x)}$

5.4 Logarithmic Functions

- The differential equation used is:

$$\frac{d}{dx}\ln(x) = \frac{1}{x}. \tag{16}$$

  and the similar RK4 method is applied for finding the value of $\ln(x)$

- To calculate $\log_{10}(x)$ we use $\frac{\ln(x)}{\ln(10)}$

5.5 Power Function Implementation via Integration

The function x^n is implemented using the differential equation:

$$\frac{dy}{dx} = ny \tag{17}$$

which follows from the definition of exponentiation.

5.5.1 Derivation

Rearranging the equation:

$$\frac{dy}{y} = n\,dx \tag{18}$$

Integrating both sides from $x = 0$ to $\ln x$:

$$\int \frac{dy}{y} = \int n\,dx \tag{19}$$

$$\ln y = nx + C \tag{20}$$

Setting the initial condition at $x = \ln x$ (since we are computing x^n),

$$\ln y = n \ln x \tag{21}$$

Exponentiating both sides:

$$y = e^{n \ln x} \tag{22}$$

which simplifies to:

$$x^n = e^{n \ln x} \tag{23}$$

5.5.2 Implementation in Code

The function is computed using the natural logarithm and exponentiation:

```
double power(double x, double n) {
    if (x <= 0) return -1;  // Only valid for positive x

    double y = 1.0;  // Initial condition: y(0) = 1
    double log_x = log(x);  // Compute ln(x)
    int steps = abs(log_x / STEP_SIZE);  // Compute number of steps dynamically

    double dx = log_x / steps;  // Adjust step size to match ln(x)


    for (int i = 0; i < steps; i++) {
        y += dx * (n * y);  // Euler's method: y_k+1 = y_k + h*f(x, y)
    }

    return y;
}
```

5.5.3 Handling Edge Cases

- If $x = 0$ and $n > 0$, return 0.
- If $x = 0$ and $n < 0$, return an error (division by zero).
- If $x < 0$ and n is non-integer, return an error (complex result).

This method efficiently computes the power function using logarithms and exponentiation while maintaining accuracy.

5.6 Decimal to Fraction converter

On pressing the button for $' ='$, we get the result of the entered expression in decimal. On pressing it again, the decimal is converted to a fraction of the form $\frac{p}{q}$ where p and q are co prime numbers.

5.7 Digit Parsing and Display Handling

- Reads input via button presses and converts them into numeric values. - Stores user input as a string to allow multi-digit numbers. - Implements a cursor movement system to allow input correction. - Dynamically updates the LCD display as the user enters expressions.

6 Temporary Answer Storage

- A variable ‘temp‘ is used to store previous results. - This allows reuse of the last computed value in new calculations. - Implements memory retrieval for quick calculations. - Stores intermediate values in case of complex operations.

7 Scrolling Functionality for Scientific Functions

7.1 Trigonometric Function Scrolling

- A cyclic button-based scrolling mechanism allows selection of ‘sin, cos, tan‘. - Pressing a button cycles through the options.

7.2 Logarithmic Function Scrolling

- A separate scroll cycle for ‘ln, $\log_{10}$‘. - Ensures intuitive navigation without interference with trigonometric functions.

7.3 Constant Selection Scrolling

- ‘e‘ and ‘π‘ have their own separate scrolling selection.

8 Support Functions

- **Backspace:** Used to clear the last entered input.
- **Clear:** Used to clear the entire screen on the LCD still keeping the temp result value.

9 Error Handling and Edge Cases

- Prevents division by zero errors by checking denominators before evaluation.
- Ensures that logarithm and square root functions do not operate on negative numbers.
- Implements overflow detection to prevent extremely large or small numbers from causing computational errors.
- Displays error messages on the LCD when invalid operations are attempted.

10 Images of results of some expressions

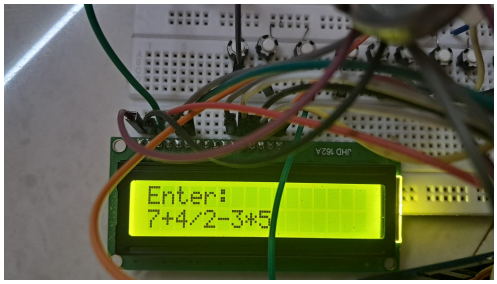


Figure 2: Expression-1

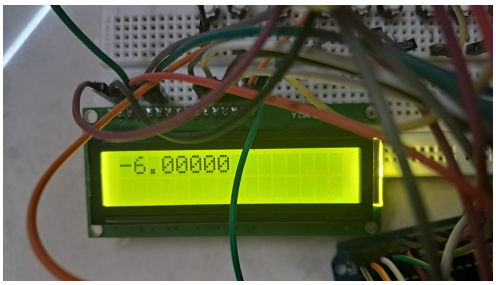


Figure 3: Result-1

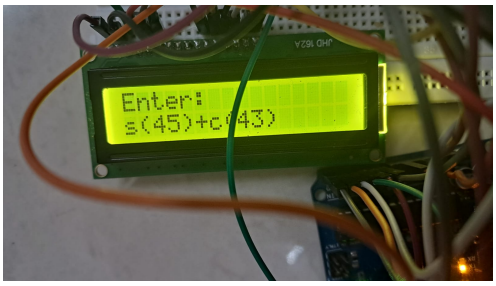


Figure 4: Expression-2

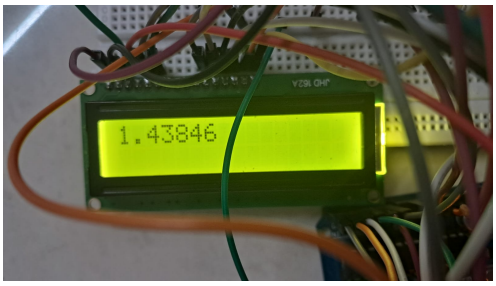


Figure 5: Result-2

11 Conclusion

This Arduino-based calculator efficiently evaluates arithmetic and scientific expressions using stacks. The resistor ladder reduces pin usage, and the scrolling feature improves usability. The RK4 method is implemented for trigonometric evaluations, ensuring high accuracy. The calculator ensures efficient parsing, evaluation, and error handling, making it both functional and user-friendly.