Scientific Calculator Using AVR-GCC

EE24BTECH11005: Credits EE24BTECH11001, EE24BTECH11002

March 24, 2025

Introduction

This project presents a scientific calculator built using the AtMega328p microcontroller with the avr-gcc framework. Instead of relying on standard libraries, an effort was made to implement mathematical functions using differential equations, staying true to the motto of the course. Special attention was given to creating a user-friendly interface with features like the ability to review and edit inputs, making the calculator both functional and easy to use.

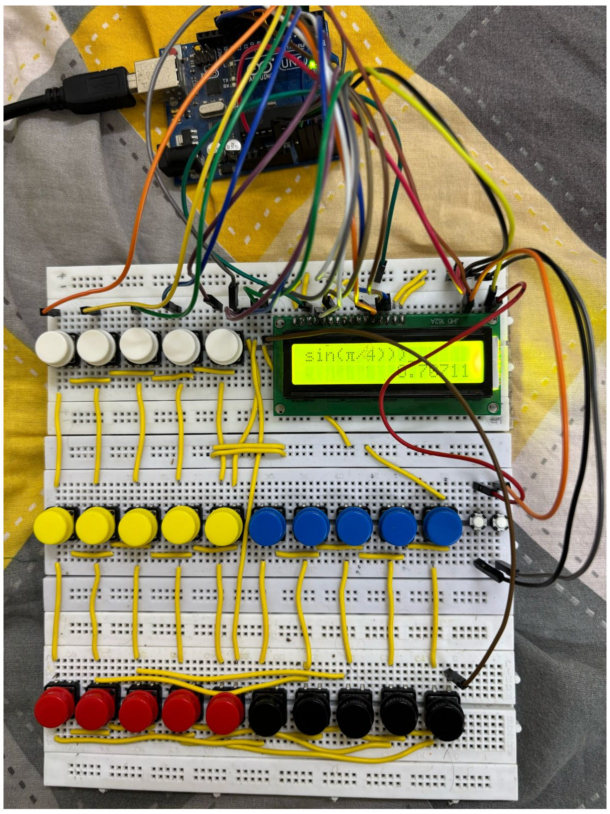


Figure 1:

Overview

Features

The calculator supports:

- Basic operations: addition ($+$), subtraction ($-$), multiplication ($*$), division ($/$), and exponentiation ($\hat{}$), factorial (!).
- Trigonometric functions: sin, cos, tan, arcsin, arccos, arctan.
- Logarithmic functions: natural log (ln), log base 10 ($\log_{10}$).
- Constants: π, e.
- Memory recall functionality.
- Autocompleting bracketis, movable cursor (to traverse code).

Hardware Overview

The hardware consists of:

Quantity	**Component**
25	Pushbuttons
1	LCD 16 x 2
1	Arduino Uno
-	Wires
1	Potentiometer

Table 1: Materials Required

- A button matrix for user input.
- An Arduino Uno microcontroller (AtMega328p) to process inputs and execute calculations.
- Connecting a 16x2 LCD to arduino for displaying results.
- Connections between the button matrix, LCD, and Arduino Uno as shown in Figure 2.

The schematic for connections is as shown below,

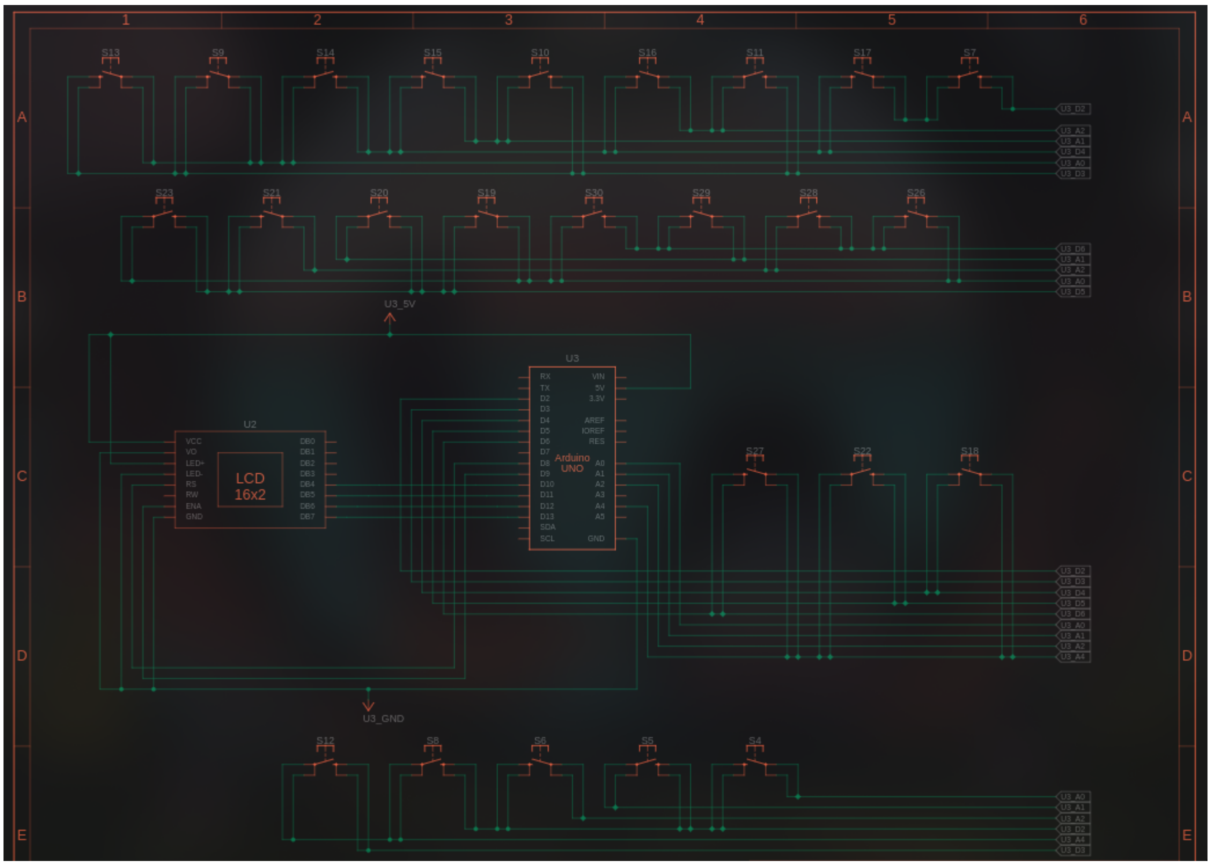


Figure 2: Schematic of Circuit

Software Overview

The software is implemented using avr-gcc. It includes:

- Implementing button matrix to check which key is pressed.
- Storing input as a string (infix)
- Converting infix to postfix using Shunting Yard algorithm.
- Evaluating postfix and returning answer.
- Support for functions like trigonometric functions (normal and inverse) logarithms, factorials, etc. All done without using **math.h** library, solving differential equations by RK4 method.

Now let us explore some of the features mentioned above in greater depth.

In Depth

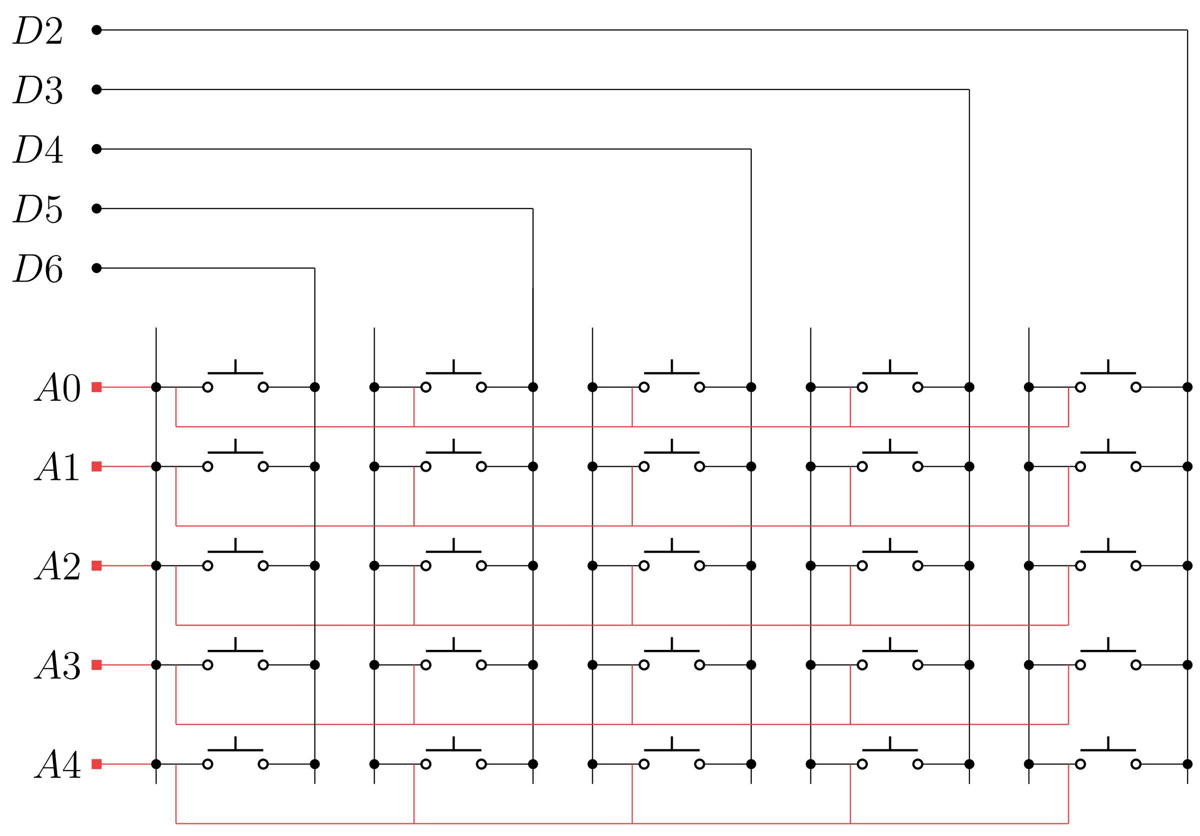
Button Matrix

The button matrix is a grid of push-button switches arranged in rows and columns. It allows multiple buttons to be connected to the microcontroller using fewer pins. In my circuit, the rows are connected to analog pins (A0 to A4), while the columns are connected

to digital pins (D2 to D6) of the Arduino.
Working Principle:

- The microcontroller scans the matrix by activating each row (setting it HIGH) one at a time while reading the columns for signals.
- When a button is pressed, it completes the circuit between its corresponding row and column.
- By identifying which row and column are active, the microcontroller determines which button was pressed.
- Suppose the first column is set to HIGH, and a button (unknown to the circuit) is pressed. When reading from the rows, the microcontroller detects a signal from the second row. This confirms that the button pressed is the second button in the first column.

This idea helped us reduce the number of pins required to implement a calculator with 25 buttons.



Numerical Methods

Key mathematical functions such as sine and logarithms are implemented using numerical methods like Runge-Kutta (RK4).

In this section, we discuss the application of the Runge-Kutta 4th order (RK4) method for solving differential equations. The RK4 method is a powerful and widely used technique to approximate solutions to ordinary differential equations (ODEs). Below, we list the differential equations used for various functions solved using the RK4 method.

Runge-Kutta 4th Order (RK4) Method

The RK4 method is given by the following iterative steps to solve the differential equation $\frac{dy}{dx} = f(x, y)$:

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$
$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$$
$$k_4 = hf(x_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where h is the step size, y_n is the value of the solution at time x_n, and $f(x, y)$ represents the differential equation.

Trigonometric Functions

For trigonometric functions, we solve the following differential equation whose solution is $y(x) = \sin(x)$:

$$\frac{d^2y}{dx} + y = 0;$$

Once we have the function to calculate sine, we can use the following facts to calculate cosine and tan.

$$\cos(x) = \sin\left(\frac{\pi}{2} - x\right)$$
$$\tan(x) = \frac{\sin(x)}{\cos(x)}$$

Inverse Trigonometric Functions

For inverse trigonometric functions, we solve the differential equation whose solution is $y(x) = \arcsin(x)$:

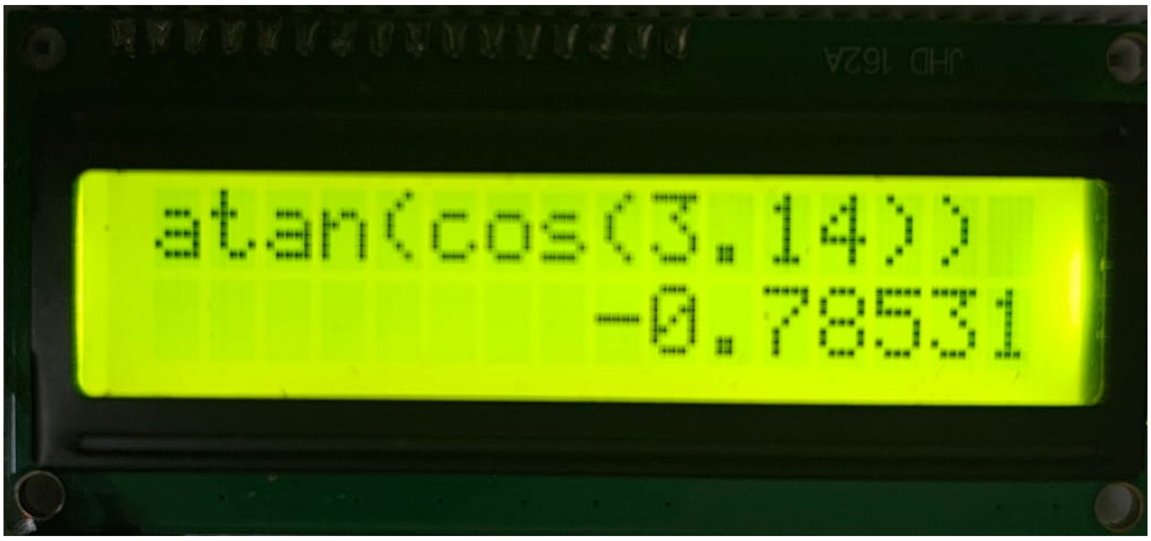
$$\frac{dy}{dx} = \frac{1}{\sqrt{1 - x^2}}$$

Once we obtain the method to calculate inverse sine, cosine inverse can be implemented by using the property,

$$\arccos(x) + \arcsin(x) = \frac{\pi}{2}$$

to obtain tan inverse we need to solve a seperate differential equation,

$$\frac{dy}{dx} = \frac{1}{1 + x^2}$$



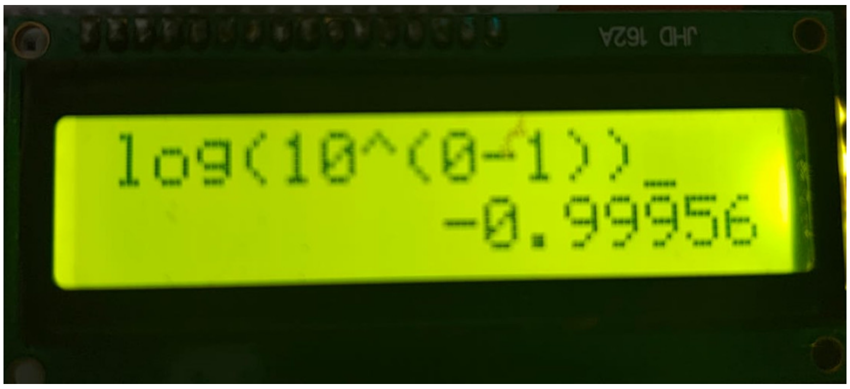
```
atan(cos(3.14))
       -0.78531
```

Figure 3: Trigonometric and Inverse Trigonometric Functions

Logarithmic Function

For the logarithmic function, we solve the differential equation whose solution is $y(t) = \ln(t)$:

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



```
log(10^(0-1))_
       -0.99956
```

Figure 4: Logarithm (Base 10)

Power (Exponent) Function

For the power function, we solve the differential equation whose solution is $y = x^a$:

$$\frac{dy}{dx} = \frac{ay}{x}$$

Information about shunting yard algorithm used can be found at *shuntingyard.pdf* file

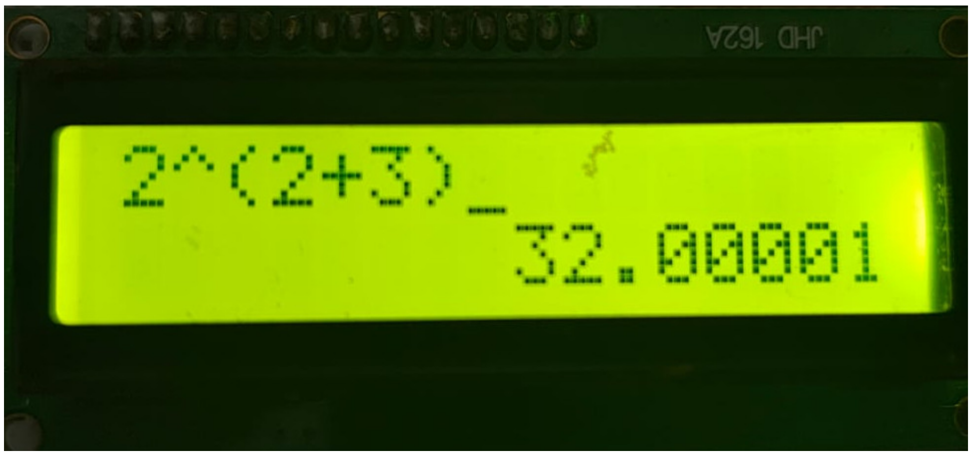


Figure 5: Power

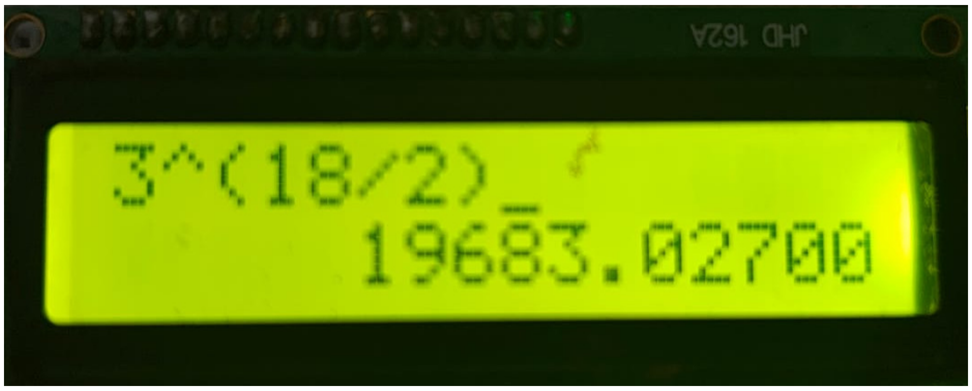


Figure 6: Exponent

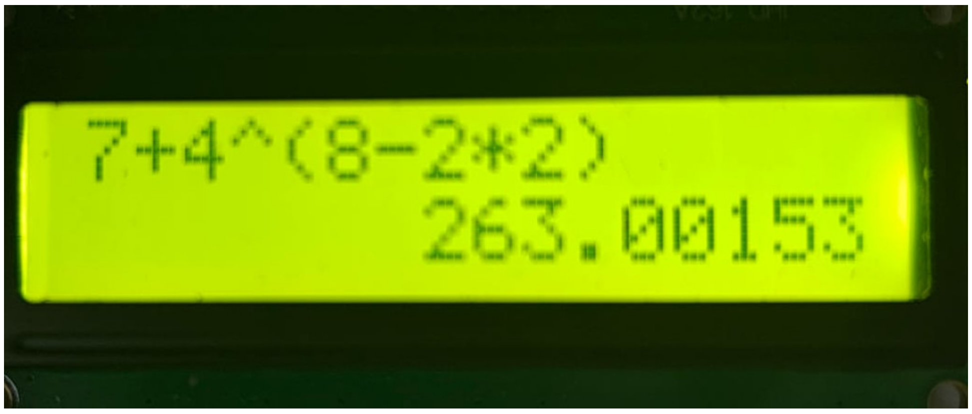


Figure 7: Bodmas

Conclusion

This project has demonstrated my attempt to implementation of a fully functional calculator using a button matrix and AtMega328p microcontroller (arduino uno) implemented using avr-gcc and differential equations.