Scientific Calculator Using AVR-GCC

EE24BTECH11002 - Agamjot Singh

March 24, 2025

Features

- Basic operations: addition (+), subtraction (−), multiplication (∗), division (/), and power (ˆ)
- Trigonometric functions: sin, cos, tan, $\sin^{-1}$ or arcsin, $\cos^{-1}$ or arccos, $\tan^{-1}$ or arctan
- Logarithmic functions: natural log (ln), log base 10 (log)
- BODMAS implementation using the Shunting Yard Algorithm
- Constants: π, e
- Memory recall functionality (with EEPROM)
- Movable Cursor

Components and Circuit Schematic

Quantity	**Component**
1	Arduino Uno
25	Push buttons
1	LCD (16 x 2)
-	Wires
1	Potentiometer

Table 1: Materials Required

5×5 button matrix is built for input from the 25 buttons, which is made to reduce the amount of wires needed to mere 10 wires for 25 buttons. Button matrix connections to the arduino and connections are given below.

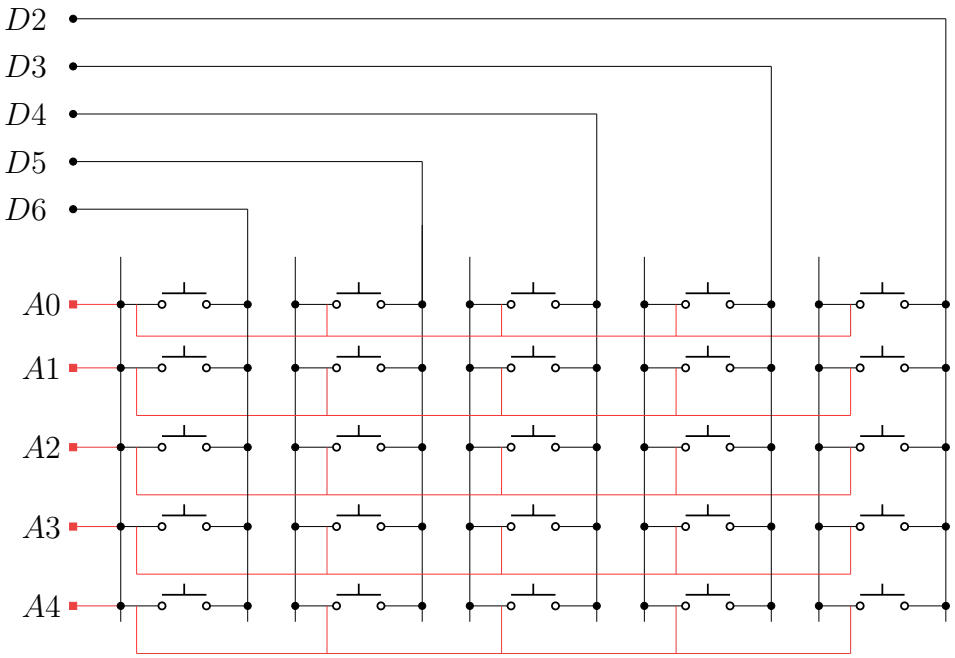


Figure 1: Button Matrix Circuit Schematic

Note that D2, D3, ... D6 and A0, A1, ..., A4 are the Arduino's Digital pins and Analog Pins respectively.

Arduino Pins	**LCD Pins**	**LCD Pin Label**	**LCD Pin Description**
GND	1	GND	
5V	2	Vcc	
GND	3	Vee	Contrast
D12	4	RS	Register Select
GND	5	R/W	Read/Write
D11	6	EN	Enable
D5	11	DB4	Serial Connection
D4	12	DB5	Serial Connection
D3	13	DB6	Serial Connection
D2	14	DB7	Serial Connection
5V	15	LED+	Backlight
GND	16	LED-	Backlight

Table 2: Arduino to LCD Pin Connections

The schematic for connections is as shown below,

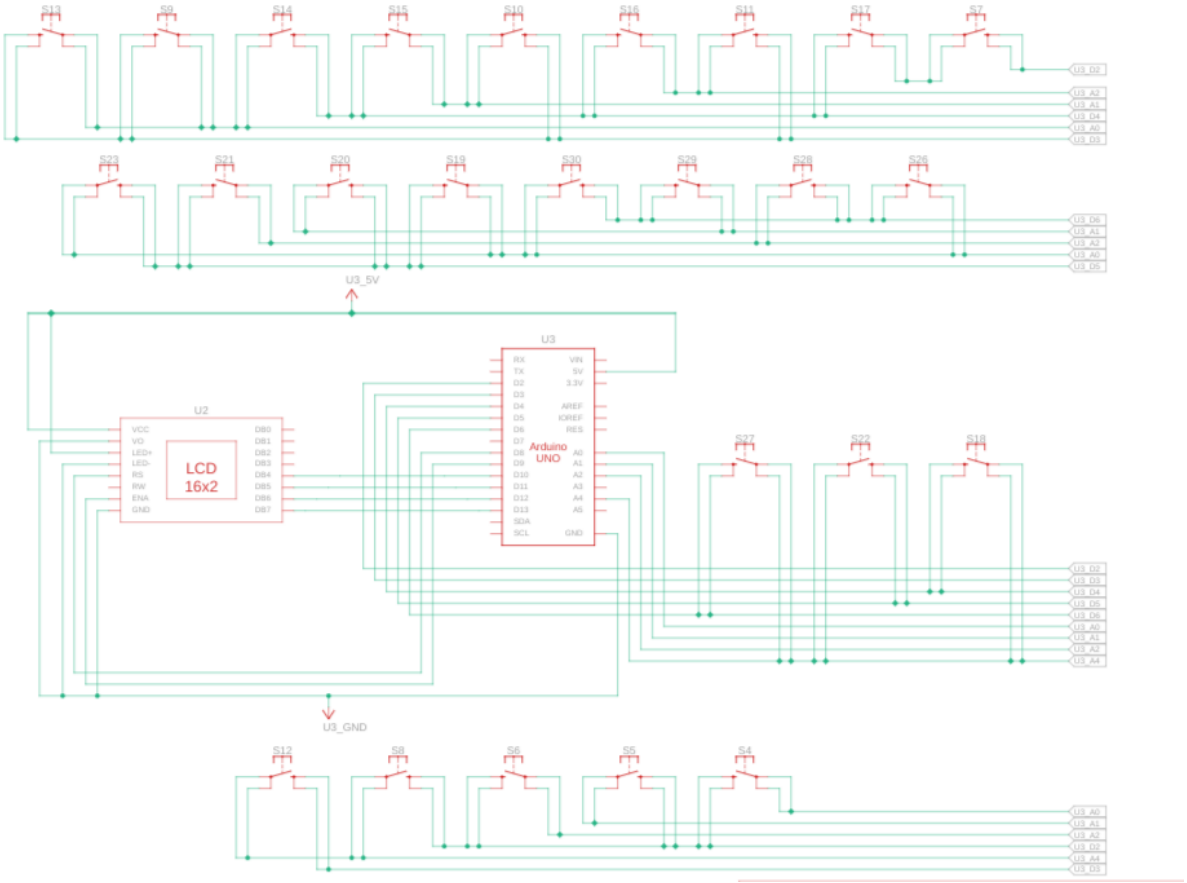


Figure 2: Schematic of Circuit.

Code implementation

The code is implemented using AVR GCC.

- 5×5 button matrix is implemented to check which key is pressed
- Parsing the input using the Shunting Yard algorithm
- Evaluating result Reversed Polish Expression to a result to display.
- Support for functions like trigonometric functions and inverse trigonometric functions, logarithms.
- All function implementations are done **without using math.h** library, solving differential equations by RK4 method.

Further Explanation on shunting Yard algorithm is found in the pdf in this directory.

Explanation of Implementation

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.

- 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 to a mere 10 pins of the microcontroller.

Numerical Methods

All the functions are implemented 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 $\frac{dy}{dx} = f(x, y)$:

$$k_1 = hf(x_n, y_n) \tag{1}$$

$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \tag{2}$$

$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \tag{3}$$

$$k_4 = hf(x_n + h, y_n + k_3) \tag{4}$$

$$y_{n+1} = y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \tag{5}$$

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

We solve the following differential equation whose solution is $y(x) = \sin(x)$,

$$\frac{d^2y}{dx} + y = 0; \tag{6}$$

Once we have $\sin x$, we can calculate $\cos x$ and $\tan x$,

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

Inverse Trigonometric Functions

We solve the differential equation whose solution is $y(x) = \arcsin(x)$,

$$\frac{dy}{dx} = \frac{1}{\sqrt{1-x^2}} \tag{7}$$

Once we have $\arcsin(x)$, we can calculate $\arccos x$,

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

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

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

Logarithmic Function

We solve the differential equation whose solution is $y(t) = \ln(t)$,

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

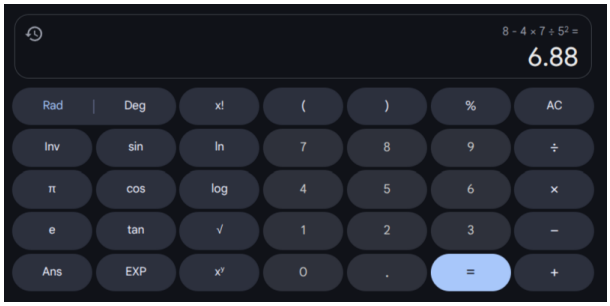
Power (Exponent) Function

We solve the differential equation whose solution is $y = x^a$,

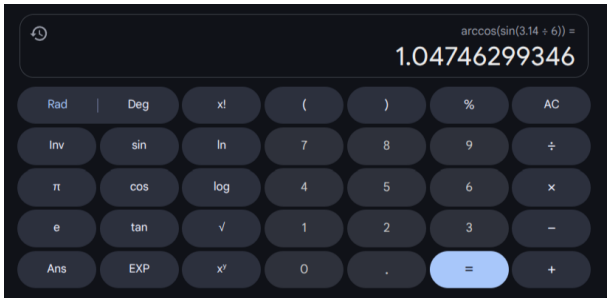
$$\frac{dy}{dx} = \frac{ay}{x} \tag{9}$$

Showcase of functions implemented

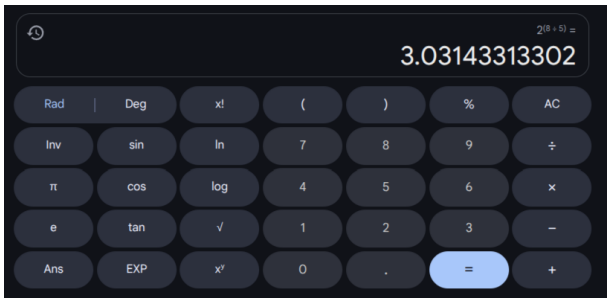
0.1 BODMAS Implementation



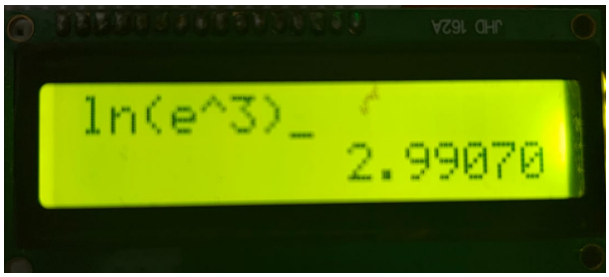
0.2 Trigonometric Functions and Inverse Trigonometric Functions Implementation

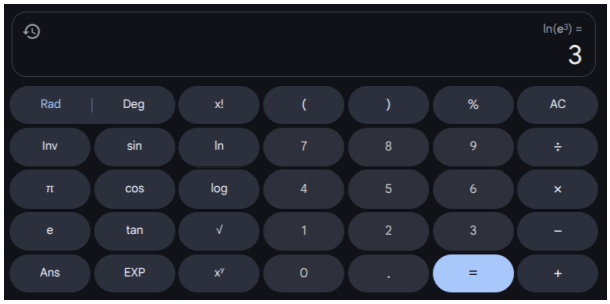


0.3 Power Implementation

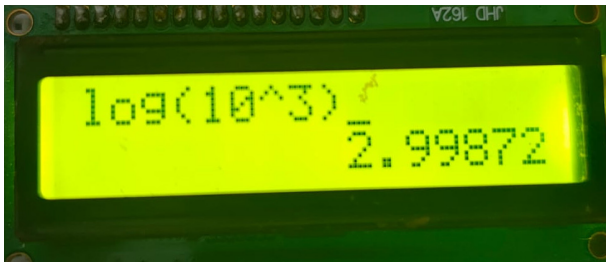


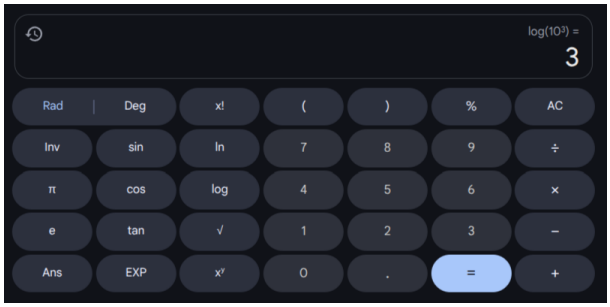
0.4 Natural Logarithm Implementation





0.5 Logarithm Base 10 Implementation





Conclusion

This project shows how I built a working calculator using a button matrix and ATMega328p microcontroller (Arduino Uno) using avr-gcc and differential equations.