CALCULATOR

EE1003 : Scientific Programming for Electrical Engineers
Indian Institute of Technology Hyderabad

Yellanki Siddhanth (EE24BTECH11059)

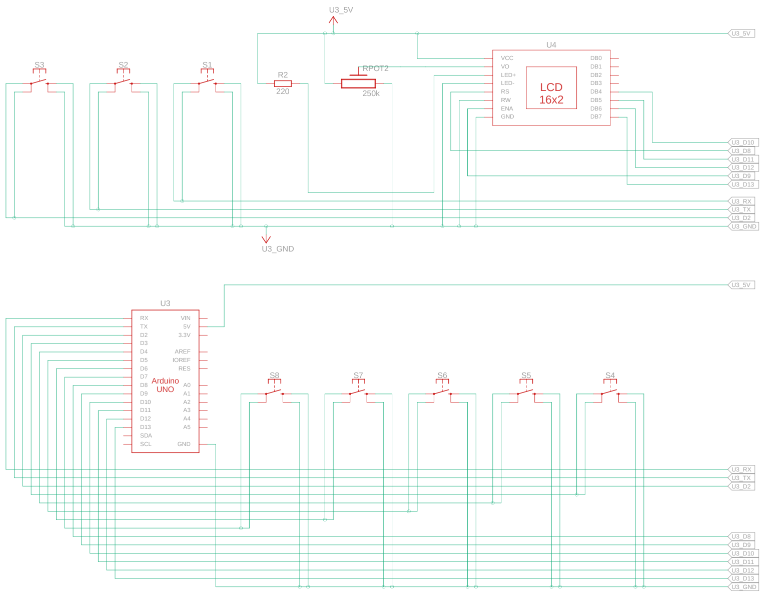
1 Introduction

This report describes a calculator system that utilizes a 16x2 LCD display and a pair of 4-button D-pads for user input. The system supports multiple modes for numbers, operators, functions, and inverse functions, providing a versatile and user-friendly interface.

2 System Components

The calculator system consists of the following components:

1) Arduino Atmel328P Microcontroller
2) 16x2 LCD Display
3) Two 4-Button D-Pads - 8 Push Buttons
4) Wires



Made with Tinkercad®

Title: Surprising Jaagub

Date: 3/24/2025, 10:49:12 PM

Sheet: 1/1

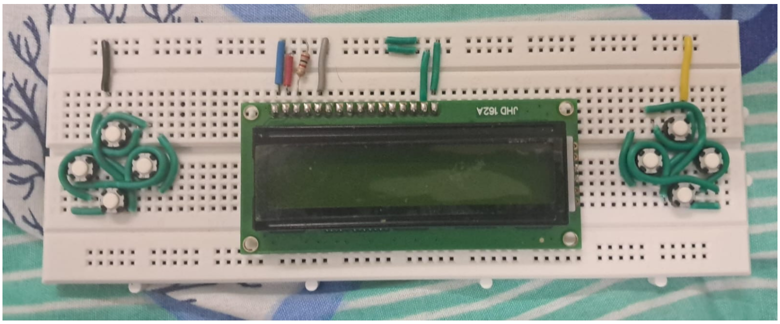


Fig. 4: Circuit Diagram

3 User Modes

The calculator features four primary modes:

1) Number Mode: Allows input of numerical values from 0 to 9.
2) Operators Mode: Includes basic arithmetic operators [+, -, *, /, e, pi,), (, . , ,].
3) Functions Mode: Offers common mathematical functions (sin, cos, tan, log, ln, pow,exp).
4) Inverse Functions Mode: Provides inverse operations for functions (asin, acos, atan).

4 Control Logic

The first D-pad controls cursor movement (left and right buttons) and expression selection buttons are up and down. In the second D-Pad's left button acts as a delete key which removes the previous element from the cursor position, whereas the right button will solve for the expression. The second D-Pad's up and down buttons allow you to scroll through the modes.

5 User Interface

First, we have to set the mode. Then we use the first d-pad to cycle through the respective mode's list. Then we just pressed the forward cursor to move to the next expression. The cursor will accurately display where current expression. Now repeating the following process will give us the required expression to be calculated. Even though this may seem to be redundant and slow, with usage and over time, this process can be done really fast. While scrolling through the modes, we can see which mode is currently set at the corner of the screen in the second line.

6 Expression Parser

I have used a library called tinyexpr.c , which allows you to parse the expression that we need to calculate. This tool allows you to calculate expressions with complex bodmas and nested function with ease and it comes with error handling. Ex. "sin($1/cos(\pi/3)$)/ log($pow(2,\pi/2)$)". This library doesn't innately support being uploaded on the arduino due to it's large size despite being called "tinyexpr". After removing

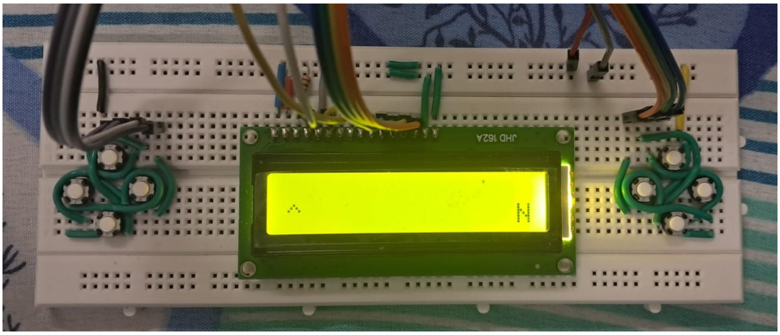


Fig. 4: Interface Example 1, Showing the cursor and the mode indicator

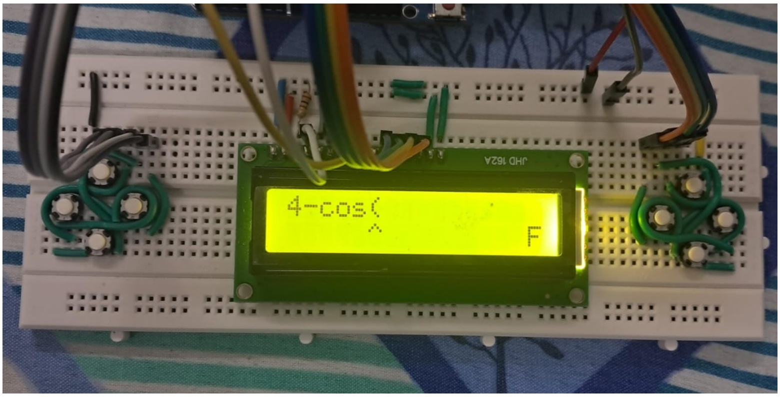


Fig. 4: Interface Example 2

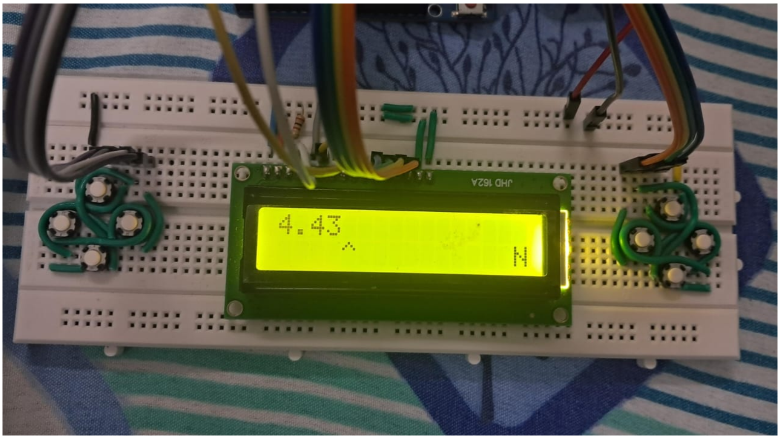


Fig. 4: Interface Example 3

redundant libraries and codes used by tinyexpr and heavily modifying, it works without a problem on the Arduino. The .hex file is now just below 18kb which is the average .hex file size of the code made by majority of the class excluding the fact that, this parser is better than theirs and also features error handling.

7 Error Handling

If any expression has a mistake in it, Ex. incomplete operators, missing parenthesis etc, the calculator will pop up an error screen showing the pointing to the error in the expression, and then it goes back into input mode.

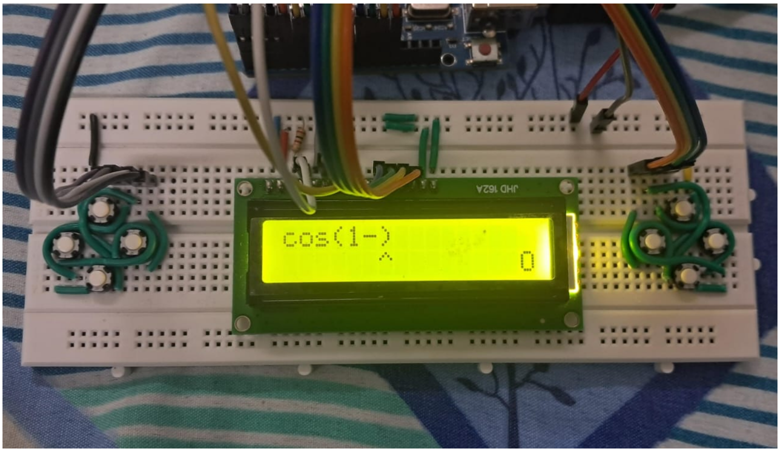


Fig. 4: Erred Prompt 1

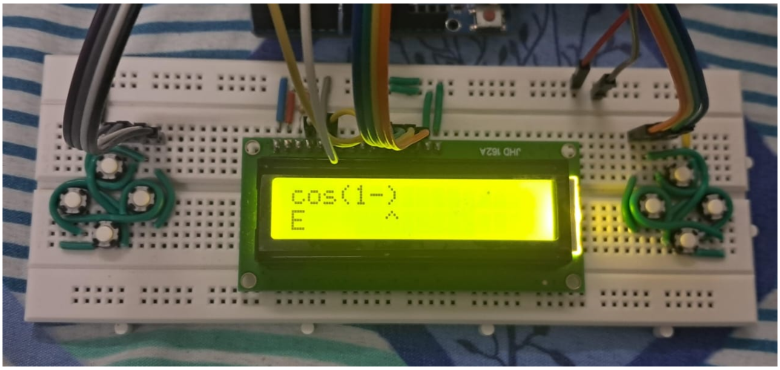


Fig. 4: Error Indicator 1

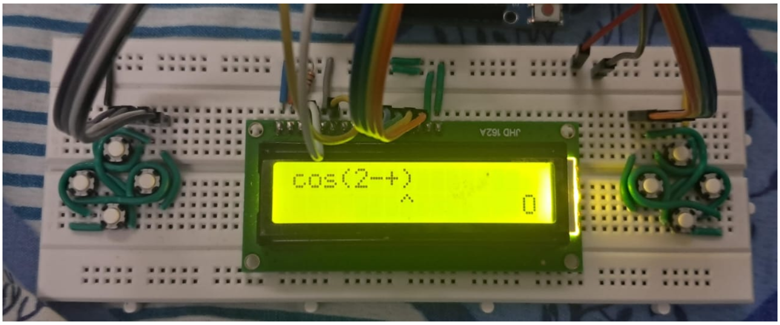


Fig. 4: Erred Prompt 2

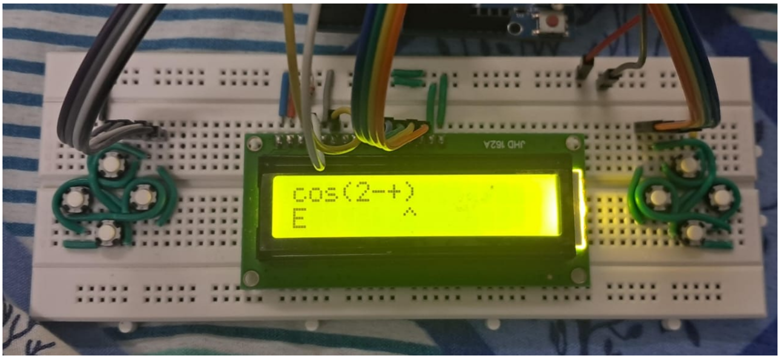


Fig. 4: Error Indicator 2

8 EEPROM Persistance

EEPROM persistence is implemented to store user data when the device is powered off. The expression, length of the expression and the cursor location is stored in the EEPROM.

9 Assembly Routines

To drive the LCD, I have implemented assembly routines. The following functions are present in the routines file.

1) SendNibble()
2) SendByte()
3) LCDCmd()
4) LCDInit()
5) LCDChar()

The file is located at,

```
/codes/final/routines.S
```

The makefile will automatically compile these routines with the embedded C code.

10 Function Implementations

I have implemented the following functions using the numerical methods.

10.1 sin(x)

I have used the Forward Euler method for estimating $\sin(x)$.

$$\frac{d^2y_n}{dx_n^2} = -y_n \tag{1}$$

Using the finite difference approximation for the second derivative,

$$\frac{y_{n+1} - 2y_n + y_{n-1}}{h^2} = -y_n \tag{2}$$

Rearranging for y_{n+1},

$$y_{n+1} = 2y_n - y_{n-1} - h^2 y_n \tag{3}$$

The initial points are $y_0 = 0, y_1 = 0.099, h = 0.01$

10.2 cos(x)

The same update equation for sin(x) is used, but this time $y_0 = 1, y_1 = 0.99, h = 0.01$.

10.3 tan(x)

I have used the second-order Runge-Kutta (RK2) method to approximate $\tan(x)$. The differential equation governing $\tan(x)$ is:

$$\frac{dy_n}{dx_n} = 1 + y_n^2 \tag{4}$$

Using the RK2 method, the update steps are:

$$k_1 = (1 + y_n^2)h, \tag{5}$$

$$k_2 = \left(1 + \left(y_n + \frac{k_1}{2}\right)^2\right)h, \tag{6}$$

$$y_{n+1} = y_n + \frac{k_2 + k_1}{2}. \tag{7}$$

The above is the update equation for tan(x), where the initial conditions are, $y_0 = 0, h = 0.01$.

10.4 ln(x)

I have used the second-order Runge-Kutta (RK2) method to approximate $\ln(x)$. The governing differential equation is:

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

Applying the RK2 method:

$$k_1 = \frac{1}{x_n}h, \tag{9}$$

$$k_2 = \frac{1}{x_n + \frac{h}{2}}h, \tag{10}$$

$$y_{n+1} = y_n + \frac{k_2 + k_1}{2} \tag{11}$$

The initial conditions are, $y_0 = -4.6, x_0 = 0.01, h = 0.01$

10.5 log(x)

Since $\log(x) = \log(e)\ln(x)$, we use the same update equation as $\ln(x)$ and then scale the function by $\log(e)$

10.6 exp(x)

The governing equation for e^x is:

$$\frac{dy}{dx} = y \tag{12}$$

Applying RK2:

$$k_1 = y_n h, \tag{13}$$

$$k_2 = (y_n + \frac{k_1}{2})h, \tag{14}$$

$$y_{n+1} = y_n + \frac{k_1 + k_2}{2}. \tag{15}$$

Here the initial conditions are, $y_0 = 1, x_0 = 0, h = 0.01$

10.7 pow(a, x)

Since $a^x = e^{x \ln a}$, we use the equation:

$$\frac{dy}{dx} = y \ln a \tag{16}$$

Applying RK2:

$$k_1 = y_n \ln ah, \tag{17}$$

$$k_2 = (y_n + \frac{k_1}{2}) \ln ah, \tag{18}$$

$$y_{n+1} = y_n + \frac{k_1 + k_2}{2}. \tag{19}$$

Here the initial conditions are the same as the conditions for $exp(x)$

10.8 asin(x)

I have used Newton's method to approximate $\sin^{-1}(x)$. The governing equation is:

$$f(y) = \sin(y) - x \tag{20}$$

Applying Newton's iteration formula:

$$y_{n+1} = y_n - \frac{\sin(y_n) - x}{\cos(y_n)} \tag{21}$$

This iterative formula refines the approximation of $\sin^{-1}(x)$. The initial condition taken is $y_0 = x$.

10.9 acos(x)

Since $\cos^{-1}(x)$ satisfies:

$$f(y) = \cos(y) - x \tag{22}$$

The Newton's update equation is:

$$y_{n+1} = y_n - \frac{\cos(y_n) - x}{-\sin(y_n)} \tag{23}$$

which ensures convergence to $\cos^{-1}(x)$. The initial condition taken is $y_0 = x$.

10.10 atan(x)

For $\tan^{-1}(x)$, the equation is:

$$f(y) = \tan(y) - x \tag{24}$$

Applying Newton's method:

$$y_{n+1} = y_n - \frac{\tan(y_n) - x}{\sec^2(y_n)} \tag{25}$$

This iterative approach provides an efficient method for computing $\tan^{-1}(x)$. The initial condition taken is $y_0 = x$.

11 Conclusion

This calculator system effectively integrates a user-friendly interface with versatile mathematical capabilities, making it a practical tool for educational and everyday use.

12 Code Requirements

The code is located in the following directory:

```
/codes/final/code.c
```

It can be compiled and uploaded directly using the Makefile.