Scientific Calculator using AVR GCC

MANOGNYA K - EE24BTECH11037

March 24, 2025

Abstract

This report documents the design and implementation of a scientific calculator using AVR microcontrollers programmed with AVR GCC. The calculator features a 16x2 LCD display, 10-digit input buttons, and multiple operation modes including basic arithmetic, trigonometric functions, logarithmic functions, and more. The implementation includes custom numerical approximations for mathematical functions using the Forward Euler method.

Contents

1 Introduction

The AVR-based scientific calculator is designed to provide a wide range of mathematical operations while maintaining a simple hardware interface. The project demonstrates the integration of multiple components including:

- AVR microcontroller (ATmega328P)
- 16x2 LCD display
- Push buttons for Input
- Custom mathematical function implementations
- Multiple operation modes

2 Hardware Design

The calculator uses the following hardware components:

2.1 Microcontroller

The project is designed for AVR microcontrollers (ATmega328P) running at 16MHz clock frequency.

2.2 LCD Interface

The 16x2 LCD is connected in 4-bit mode to conserve I/O pins:

- VSS - GND
- VDD - 5V
- V_0 - middle pin of potentiometer
- RS (Register Select) - 12
- RW (Read/Write) - GND
- EN (Enable) - 11
- Data pins D4-D7 - PD5-PD2

- A(Backlight) - 5V
- K(Backlight) - GND

The reference diagram for the LCD pins and their functions is shown below :

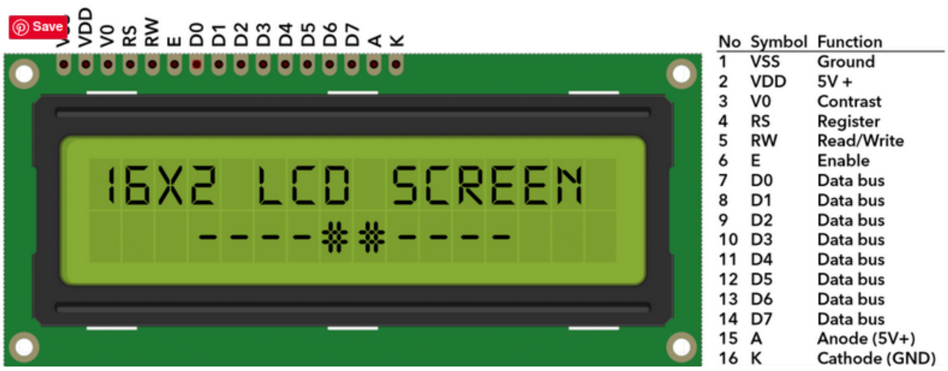

Figure 1: 16x2 LCD for reference

2.3 Input Buttons

- 10 digit buttons (0-9) connected to pins 6-10 and 14-18 (A0-A4)
- Shift button (A5) for accessing secondary functions
- Extra mode button (digital pin 13) for advanced functions
- Other terminal of the buttons is grounded

- The table for the buttons is shown below;

Button	**Right Toggle**	**Left Toggle**
2	$+$	sin^{-1}
3	-	cos^{-1}
4	$*$	tan^{-1}
5	$/$	log_{10}
6	$=$	ln
7	backspace	
8	sin	
9	cos	
10	e^x	
11	$\surd$	

2.4 Potentiometer

- The middle wire is connected to V_0 via 220Ω resistor
- One end is grounded and other is connected to $5V$

- The overall circuit connections are summarized in the table figue;
- The below is the circuit that I have set up;

Table 1: Connection Details

LCD	Connection
1	GND
2	5v
3	potentiometer (middle)
4	12
5	gnd
6	11
7	unused
8	unused
9	unused
10	unused
11	5
12	4
13	3
14	2
15	5v
16	gND

Buttons	Connection
1	13
2	6
3	7
4	8
5	9
6	10
7	A0
8	A1
9	A2
10	A3
11	A4
12	A5

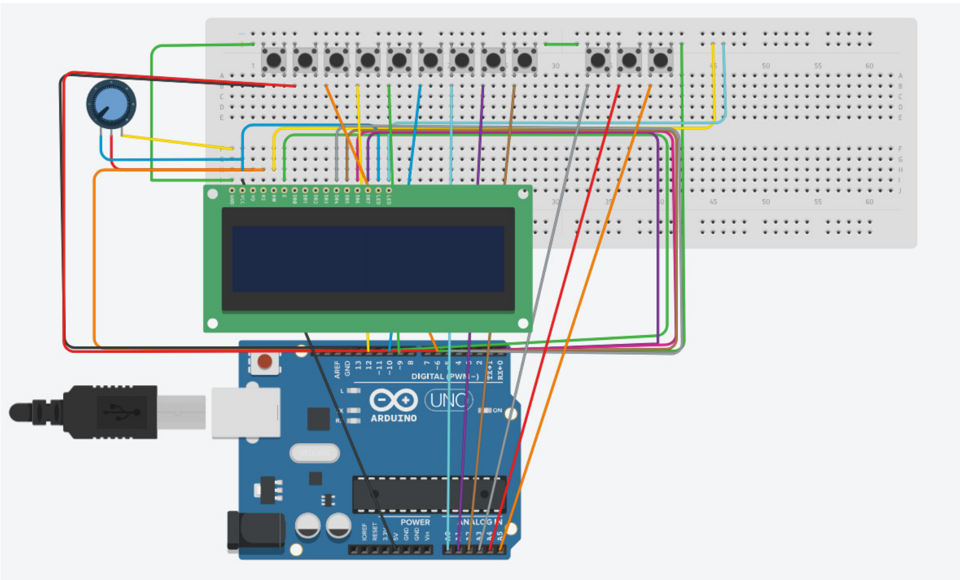

Figure 2: Circuit Setup

3 Software Architecture

The software is structured into several key components:

3.1 Main Program Flow

The program follows the standard Arduino-style structure with `setup()` and `loop()` functions:

- `setup()` initializes the LCD and button inputs
- `loop()` continuously checks button states and processes input

3.2 Input Processing

The calculator implements three operation modes:

- Normal mode (digits 0-9)

- Shift mode (arithmetic operations and basic functions)
- Extra mode (advanced mathematical functions)

3.3 Display Management

The LCD display shows:

- Current input expression (up to 32 characters across two lines)
- Calculation results with 3 decimal places precision

4 Mathematical Function Implementations

The calculator implements several mathematical functions using numerical approximations:

4.1 Trigonometric Functions

The sine and cosine functions are implemented using the Forward Euler method on the coupled ordinary differential equations:

$$\frac{dy}{dx} = z \quad (\text{where } y = \sin(x))$$
$$\frac{dz}{dx} = -y \quad (\text{where } z = \cos(x))$$

4.2 Exponential Function

The exponential function uses Forward Euler on:

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

4.3 Square Root

Implemented using Newton's method:

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$$

4.4 Logarithmic Functions

The natural logarithm uses:

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

with initial condition $y(1) = 0$.

5 Code Structure

The main components of the code are:

5.1 LCD Interface

```
void LCD_Command(unsigned char cmnd);
void LCD_Char(unsigned char data);
void LCD_Init();
void LCD_String(const char* str);
void LCD_Clear();
```

5.2 Button Handling

```
void pinMode(int pin, int mode);
int digitalRead(int pin);
void handleSpecial(char op);
```

5.3 Mathematical Functions

```
float mySin(float x);
float myCos(float x);
float myExp(float x);
float mySqrt(float x);
float myAsin(float x);
float myAcos(float x);
float myAtan(float x);
float myLn(float x);
float myLog10(float x);
```

5.4 Expression Evaluation

```
float evaluateFullExpression(const char* expr);
float evaluateExpression(const char* expr);
```

6 Challenges and Solutions

6.1 Numerical Approximation Accuracy

The Forward Euler method provides reasonable accuracy for the calculator's purposes, though more advanced methods (like Runge-Kutta) could improve precision.

6.2 Button Debouncing

Software debouncing is implemented with a 50ms delay after detecting a button press.

6.3 Memory Constraints

The AVR's limited RAM requires careful management of string buffers and intermediate calculation results.

7 Conclusion

The AVR-based scientific calculator successfully demonstrates:

- Integration of multiple hardware components
- Implementation of complex mathematical functions on limited hardware
- User interface design for embedded systems
- Numerical methods for function approximation

Future improvements could include:

- Implementation of operator precedence
- Additional mathematical functions
- Hardware PCB design for a standalone device
- Power management for battery operation