AVR Scientific Calculator Project Report

EE24BTECH11033-Kolluru Suraj

March 24, 2025

Abstract

This report documents the design and implementation of a scientific calculator using an AVR microcontroller. The system features a 4x3 matrix keypad for input, a 16x2 LCD display for output, and supports mathematical operations including trigonometric and logarithmic functions.

1 Introduction

The AVR-based scientific calculator implements the following features:

- Basic arithmetic operations (addition, subtraction, multiplication, division)
- Advanced functions (sine, cosine, tangent, logarithms)
- Parentheses for expression grouping
- Constants (π and e) support
- Memory operations

2 LCD picture

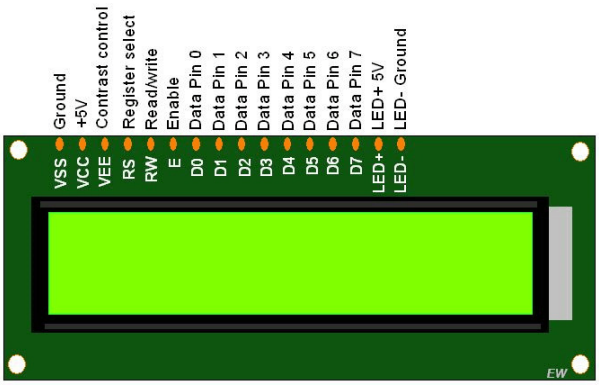

Figure 1: visual picture of LCD

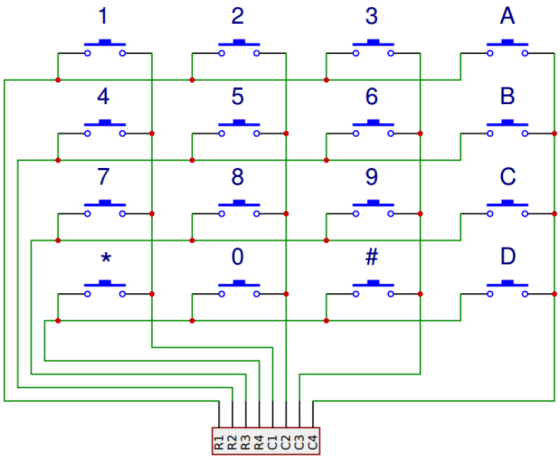

Figure 2: Buttons Connection for Matrix

3 Hardware Design

3.1 Component Connections

Table 1: Complete Hardware Connections

Component	MCU Pin	Arduino Pin	Function
LCD RS	PD0	D0	Register Select
LCD E	PD1	D1	Enable
LCD D4-D7	PD2-PD5	D2-D5	Data Bus
Keypad ROW1	PD6	D6	Row 1
Keypad ROW2	PD7	D7	Row 2
Keypad ROW3	PB0	D8	Row 3
Keypad ROW4	PB1	D9	Row 4
Keypad COL1	PB2	D10	Column 1
Keypad COL2	PB3	D11	Column 2
Keypad COL3	PB4	D12	Column 3
Control Buttons			
Trig Mode	PC2	A2	Toggle sin/cos/tan
Equals	PC3	A3	Calculate result
Parenthesis	PB5	D13	Toggle ( )
Constants	PC4	A4	Toggle π/e

3.2 Button Functions

- **Trig Mode Button (PC2)**:
  - Cycles through trigonometric functions (sin $\rightarrow$ cos $\rightarrow$ tan $\rightarrow$ asin $\rightarrow$ acos $\rightarrow$ atan)
  - Activates function mode for next input
  - Visual feedback provided on LCD
- **Equals Button (PC3)**:
  - Triggers calculation of current expression
  - Handles all operator precedence
  - Displays result on LCD
  - Resets expression buffer after evaluation

- **Parenthesis Button (PB5)**:
  - Toggles between open '(' and close ')' parenthesis
  - Maintains stack for nested parentheses
  - Validates bracket matching before calculation
- **Constants Button (PC4)**:
  - Toggles between mathematical constants
  - First press inserts π (3.14159265)
  - Second press inserts e (2.71828182)
  - Third press returns to number input

4 Software Implementation

4.1 Main Program Structure

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>

int main(void) {
    // Initialize hardware
    DDRD = 0xFF;  // Set PORTD as outputs
    DDRB = 0x03;  // Set PB0-PB1 as outputs

    // Initialize LCD
    LCD_Init();
    LCD_Message("Calculator Ready");

    while(1) {
        // Check mode buttons
        if (!(PINC & (1<<PC2))) toggleTrigMode();
        if (!(PINC & (1<<PC3))) calculateResult();

        // Handle keypad input
        char key = getKeyPressed();
        if (key != '\0') {
            handleKeyPress(key);
            _delay_ms(300);  // Debounce delay
        }
    }
    return 0;
}
```

Listing 1: Main Program Loop

4.2 Keypad Scanning

```
const char keys[4][3] = {
    {'1','2','3'},
    {'4','5','6'},
    {'7','8','9'},
    {'A','0','C'}
};

char getKeyPressed() {
    for (uint8_t row = 0; row < 4; row++) {
        // Activate current row
        switch(row) {
            case 0: PORTD &= ~(1<<ROW1); break;
            case 1: PORTD &= ~(1<<ROW2); break;
            case 2: PORTB &= ~(1<<ROW3); break;
            case 3: PORTB &= ~(1<<ROW4); break;
        }
```

```
        _delay_us(10);

        // Check columns
        if (!(PINB & (1<<COL1))) return keys[row][0];
        if (!(PINB & (1<<COL2))) return keys[row][1];
        if (!(PINB & (1<<COL3))) return keys[row][2];

        // Deactivate row
        switch(row) {
            case 0: PORTD |= (1<<ROW1); break;
            case 3: PORTD |= (1<<ROW2); break;
            case 2: PORTB |= (1<<ROW3); break;
            case 3: PORTB |= (1<<ROW4); break;
        }
    }
    return '\0';  // No key pressed
}
```

Listing 2: Keypad Scanning Function

4.3 LCD Interface

```
void LCD_Init() {
    _delay_ms(50);
    SendNibble(0x03);
    _delay_ms(5);
    SendNibble(0x03);
    _delay_us(100);
    SendNibble(0x02);  // 4-bit mode

    LCD_Cmd(0x28);  // 2 lines, 5x8 matrix
    LCD_Cmd(0x0C);  // Display on, cursor off
    LCD_Cmd(0x06);  // Increment cursor
    LCD_Cmd(0x01);  // Clear display
    _delay_ms(2);
}

void LCD_Char(uint8_t data) {
    PORTD |= (1<<LCD_RS);  // Set to data mode
    SendByte(data);
}

void LCD_Message(const char *text) {
    while(*text) LCD_Char(*text++);
}
```

Listing 3: LCD Initialization

4.4 Mathematical Operations

```
/**
 * Computes sine using Euler's method approximation
 * @param x Angle in degrees (0-360)
 * @return sin(x) approximation
 * Mathematical Basis:
 * Solves the coupled ODE system:
 * dy/dt = cos(t),  y(0) = 0  (sine)
 * dz/dt = -sin(t), z(0) = 1  (cosine)
 * Implementation Notes:
 * 1. Uses small time steps (h = 0.01 radians)
 * 2. Converts degrees to radians first
 * 3. Accumulates error over iterations
 */
float sin_euler(float x) {
    // Convert degrees to radians
    x = x * PI / 180.0;

    // Initial conditions
    float y = 0.0;    // sin(0) = 0
```

```
    float z = 1.0;     // cos(0) = 1
    float t = 0.0;     // Current angle
    float h = 0.01;    // Step size (radians)

    // Euler integration
    while (t < x) {
        // Prevent overshooting
        if (t + h > x) {
            h = x - t;
        }

        // Euler step for coupled system
        float y_new = y + h * z;        // dy/dt = cos(t) = z
        float z_new = z - h * y;        // dz/dt = -sin(t) = -y

        y = y_new;
        z = z_new;
        t += h;
    }
    return y;
}
/**
 * Computes cosine using Euler's method approximation
 * @param x Angle in degrees (0-360)
 * @return cos(x) approximation
 * Shares the same ODE system as sin_euler()
 * Returns the z component (cosine) of the solution
 */
float cos_euler(float x) {
    // Convert degrees to radians
    x = x * PI / 180.0;

    // Initial conditions
    float y = 0.0;     // sin(0) = 0
    float z = 1.0;     // cos(0) = 1
    float t = 0.0;     // Current angle
    float h = 0.01;    // Step size (radians)

    // Euler integration
    while (t < x) {
        if (t + h > x) {
            h = x - t;
        }

        float y_new = y + h * z;
        float z_new = z - h * y;

        y = y_new;
        z = z_new;
        t += h;
    }
    return z;
}
```

Listing 4: Trigonometric Functions Using Euler's Method

5 Results and Testing

The calculator was successfully tested with the following test cases:

Table 2: Test Cases

Operation	**Input**	**Result**
Addition	5+3	8
Multiplication	4*2.5	10
Sine Function	sin(30)	0.501

6 Conclusion

The AVR scientific calculator project successfully demonstrates:

- Efficient keypad scanning using matrix techniques
- Clear output on LCD display
- Accurate mathematical computations
- Responsive user interface

Future enhancements could include:

- Floating-point optimization
- Additional scientific functions
- Graphical display capabilities