AVR Scientific Calculator Project Report

Mokshith Kumar Reddy

March 24, 2025

Contents

1 Introductions

This report consists of the data how to make a scientific calculator(can handle algebraic, trigonometric, logarithmic functions) from the components shown below.

2 Components Required

- Arduino Uno/Nano
- LCD-Display
- Push Buttons
- Breadboard and Wires

3 Hardware Design

3.1 Component Connections

Table 0: 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

3.2 Matrix Scanning

Here we are implementing the matrix method keyboard scanning the 12 different push buttons using only 7 ports as follows, we arrange the buttons in the matrix form of 4 × 3 and we connect every lower wire of buttons in a row and every upper wire of buttons in the column is connected so if we enable a button a unique set of signal is given to Arduino. So now we need conserve the ports.

4 Software Implementation

4.1 Main Program Structure

```c
#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 1: 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

```
float sin_euler(float x) {
    x = x * PI / 180;  // Convert to radians
    float term = x, sum = x;

    for(int n = 3; n < 15; n += 2) {
        term *= -x*x/(n*(n-1));
        sum += term;
    }
    return sum;
}

float cos_euler(float x) {
    x = x * PI / 180;  // Convert to radians
    float term = 1, sum = 1;

    for(int n = 2; n < 15; n += 2) {
        term *= -x*x/(n*(n-1));
        sum += term;
    }
    return sum;
}
```

Listing 4: Trigonometric Functions

5 Conclusion

The AVR scientific calculator project successfully demonstrates: Efficient keypad scanning using matrix techniques , Clear output on LCD display, Accurate mathematical computations and Responsive user interface

Future enhancements could include:

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