Scientific Calculator

J.KEDARANANDA
EE24BTECH11030

March 2025

Contents

1 Introduction

This project implements a scientific calculator using an Arduino microcontroller with a 5×5 keypad matrix and an LCD display. The calculator is capable of performing both basic arithmetic operations and advanced mathematical functions such as trigonometric calculations, logarithms, and exponentiation.

The scientific calculator serves as a practical application of embedded systems design and programming, combining hardware interfacing with mathematical computation algorithms. As processors become more powerful and interfaces more user-friendly, calculators continue to be essential tools in engineering, science, and education.

1.1 Project Objectives

The main objectives of this project are:

- Develop a functional scientific calculator using the Arduino platform
- Implement a user-friendly interface using a 5×5 key matrix and LCD display
- Support basic arithmetic operations: addition, subtraction, multiplication, and division
- Include advanced mathematical functions: trigonometry, logarithms, and power functions
- Create an efficient and responsive system using low-level programming techniques in C

1.2 Scope of the Project

The scientific calculator provides a comprehensive range of functions comparable to a standard scientific calculator, including:

- Basic operations: addition, subtraction, multiplication, division
- Power and square root functions
- Trigonometric functions: sine, cosine, tangent
- Inverse trigonometric functions (with shift key)
- Natural and base-10 logarithms
- Support for constants like Euler's number (e)

2 Hardware Components

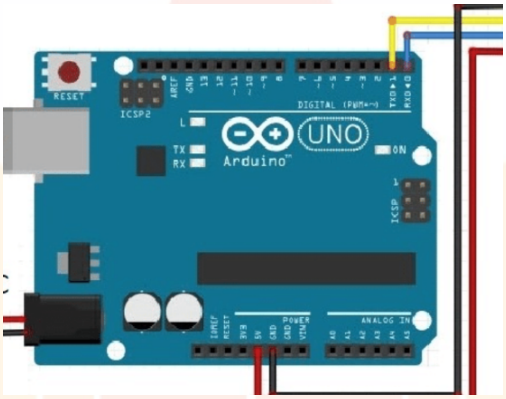
The project utilizes several key hardware components to build a functional calculator system. Each component plays a specific role in the overall design and functionality.

2.1 Arduino UNO

The Arduino UNO serves as the central microcontroller unit of the calculator. Based on the ATmega328P microcontroller, it provides:

- 6 analog inputs
- 16 MHz crystal oscillator
- USB connection for programming
- 32 KB Flash memory for storing program code
- 2 KB SRAM for runtime operations

The Arduino's ability to handle both the keypad input scanning and LCD display control simultaneously makes it ideal for this application.

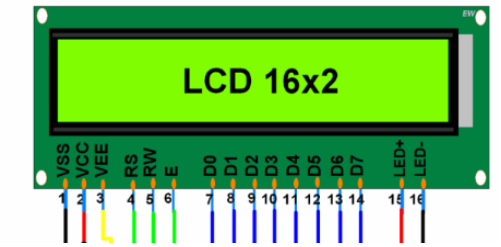


2.2 16×2 LCD Display

The 16×2 LCD (Liquid Crystal Display) module provides the visual interface for the calculator, with:

- 16 characters per line, 2 lines total
- HD44780 compatible controller
- 4-bit interface mode to minimize pin usage
- 5×8 pixel character display
- Contrast adjustment capability
- Backlight for better visibility

This display size provides sufficient space to show both the input expressions and calculation results while minimizing the project's physical footprint.



2.3 5×5 Matrix Keypad

The custom 5×5 matrix keypad provides the input interface with 25 keys total:

- Numeric keys: 0-9
- Basic operation keys: +, -, *, /, =
- Advanced function keys: sin, cos, tan, sqrt, log
- Special purpose keys: Shift, Clear, decimal point

The matrix arrangement allows for 25 keys while using only 10 I/O pins (5 rows + 5 columns), maximizing input capability with minimal pin usage.

2.4 Passive Components

Additional components used in the circuit include:

- Potentiometer for LCD contrast adjustment
- Jumper wires for connections
- Breadboard or PCB for mounting
- Power supply components

3 Circuit Connections

The calculator circuit integrates the Arduino, LCD display, and keypad matrix through carefully planned connections that maximize I/O efficiency.

3.1 LCD Connections

The 16×2 LCD is connected to the Arduino in 4-bit mode to save I/O pins:

LCD Pin	Arduino Pin	Function
RS	13 (PB5)	Register Select
E	12 (PB4)	Enable
D4	A2 (PC2)	Data Bit 4
D5	A3 (PC3)	Data Bit 5
D6	A4 (PC4)	Data Bit 6
D7	A5 (PC5)	Data Bit 7
VSS	GND	Ground
VDD	5V	Power
V0	Potentiometer	Contrast Control

3.2 5×5 Keypad Matrix Connections

The keypad is arranged in a matrix of 5 rows and 5 columns, requiring 10 I/O pins:

Keypad Line	Arduino Pin	Configuration
Row 1	6 (PD6)	Output (High with Pull-down)
Row 2	7 (PD7)	Output (High with Pull-down)
Row 3	8 (PB0)	Output (High with Pull-down)
Row 4	9 (PB1)	Output (High with Pull-down)
Row 5	A1 (PC1)	Output (High with Pull-down)
Column 1	2 (PD2)	Input (Pull-up)
Column 2	3 (PD3)	Input (Pull-up)
Column 3	4 (PD4)	Input (Pull-up)
Column 4	5 (PD5)	Input (Pull-up)
Column 5	A0 (PC0)	Input (Pull-up)

3.3 Keypad Matrix Layout

The 5×5 keypad matrix is organized as follows, with each key mapped to its corresponding character:

1	**2**	**3**	**4**	**5**
6	**7**	**8**	**9**	**0**
+	-	*	/	=
sin	**cos**	**tan**	**.**	**Clear**

3.4 Full Circuit Diagram

The complete circuit diagram showing all connections between the Arduino, LCD display, and keypad matrix is shown below:

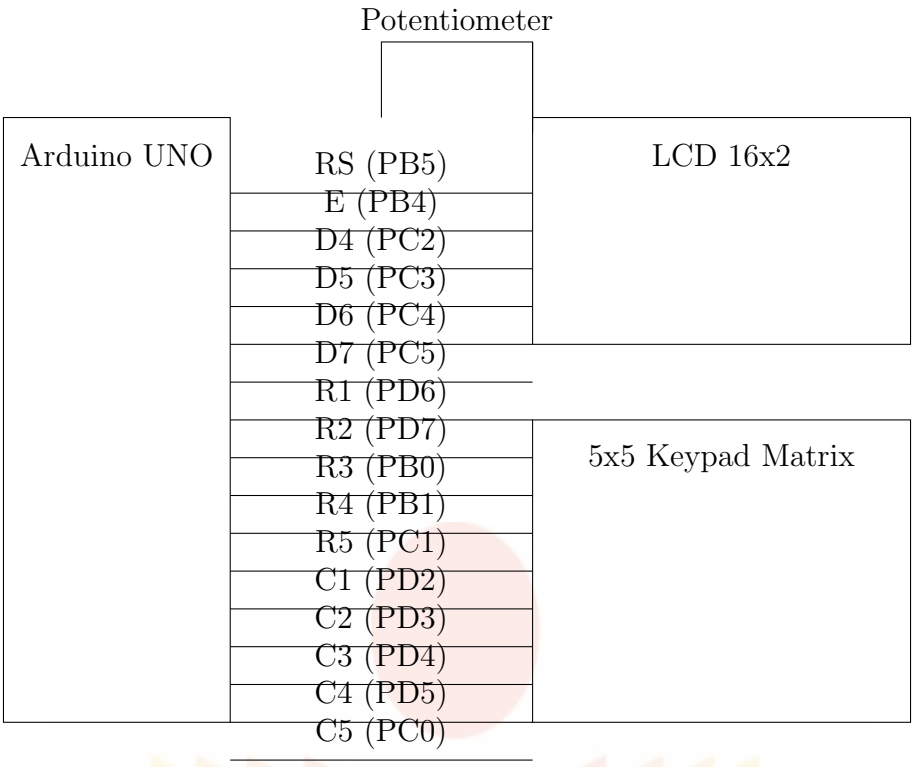
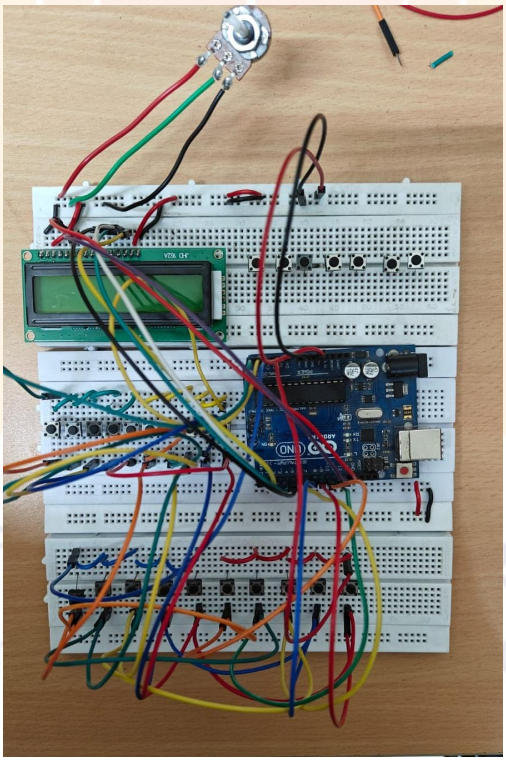


Figure 1: Complete circuit diagram of the scientific calculator



4 Hardware Setup

The physical assembly of the calculator requires careful attention to detail to ensure proper functionality and reliability.

4.1 Assembly Process

The hardware setup followed these steps:

1. Position the 16×2 LCD display and connect the power lines (VDD, VSS)
2. Connect the LCD control lines (RS, E) to the Arduino
3. Connect the LCD data lines (D4-D7) to the Arduino
4. Install the contrast potentiometer and connect it to the LCD's V0 pin
5. Arrange the 5×5 keypad matrix in a usable configuration
6. Connect the row pins to the designated Arduino output pins
7. Connect the column pins to the designated Arduino input pins with pull-up resistors
8. Verify all connections against the circuit diagram
9. Power the system via USB or external power supply

4.2 Troubleshooting Common Issues

During the assembly process, several common issues were addressed:

- LCD contrast adjustment: Finding the optimal potentiometer setting to ensure text visibility
- Key debouncing: Implementing software debounce in the code to prevent multiple key registrations
- Power requirements: Ensuring stable power supply for consistent operation

5 Final Setup

The completed calculator integrates all components into a functional system that provides both basic and advanced calculation capabilities.

5.1 Physical Arrangement

The final calculator arrangement prioritizes usability and accessibility:

- LCD positioned at the top for optimal viewing angle
- Keypad arranged in a logical layout with commonly used keys in easily accessible positions
- Numeric keys (0-9) placed in a familiar telephone-style arrangement
- Operation keys (+, -, *, /, =) positioned for convenient access
- Function keys grouped by category (trigonometric, logarithmic, etc.)
- Clear and Shift keys positioned for easy identification

5.2 Software Features

The implemented software provides a comprehensive set of calculator functions:

- Basic arithmetic: Addition, subtraction, multiplication, division
- Advanced functions: Powers, roots, trigonometry, logarithms
- Dual-mode operation through Shift key functionality
- Automatic display updating as expressions are entered
- Clear function for resetting calculations
- Error handling for invalid operations (e.g., division by zero)
- Memory of previous calculation result

5.3 Performance Characteristics

The completed calculator demonstrates the following performance characteristics:

- Response time: ¡ 100ms for key detection
- Calculation accuracy: Comparable to standard scientific calculators
- Display update speed: Nearly instantaneous (¡ 50ms)
- Power consumption: Approximately 100mA at 5V during operation
- Input capacity: Up to 32 characters in the expression buffer
- Memory usage: ¡ 70% of available SRAM on the Arduino

6 Working Principle

The scientific calculator operates based on several key principles and algorithms implemented in the embedded C code.

6.1 Keypad Scanning Algorithm

The calculator uses a polling method to scan the keypad matrix:

1. Set all rows to high (logical 1)
2. Sequentially set each row to low (logical 0), one at a time
3. While a row is low, check all columns
4. Map the row-column coordinates to the corresponding character using the key map
5. Implement debouncing by waiting approximately 50ms after detection
6. Wait for key release before continuing

This scanning method efficiently detects key presses while minimizing the number of required I/O pins.

6.2 LCD Control Protocol

The LCD is controlled using the industry-standard HD44780 protocol in 4-bit mode:

1. Initialize the LCD with specific configuration commands
2. Send commands with RS pin low, data with RS pin high
3. Split each byte into two 4-bit nibbles for transmission
4. Toggle the E (Enable) pin to latch data into the LCD controller
5. Use specific command codes for operations like clear screen, cursor positioning
6. Implement appropriate delays between commands per HD44780 specifications

6.3 Expression Parsing and Evaluation

Mathematical expressions are processed through a simplified parsing algorithm:

1. Capture input characters and build an expression string
2. Detect and handle special functions (trigonometric, logarithmic, etc.)
3. Apply operator precedence rules for basic arithmetic

4. Calculate results using appropriate mathematical library functions
5. Handle special cases (e.g., division by zero, invalid inputs)
6. Format and display the result with appropriate precision

The implementation uses the AVR libc mathematical functions for accurate calculation of trigonometric functions, logarithms, and other advanced operations.

6.4 Key Features Implementation

Several key features of the calculator required specific implementation approaches:

- **Shift Key Functionality:** A toggle flag tracks the shift state, changing the interpretation of function keys like trigonometric buttons (sin → asin, etc.)
- **Expression Building:** Characters are appended to a buffer as keys are pressed, with special handling for function names (e.g., "sin(")
- **Result Memory:** After calculation, the result becomes the new input, allowing for continuous calculations
- **Error Handling:** NaN (Not a Number) detection prevents the calculator from displaying meaningless results

6.5 Code Snippet: Keypad Scanning

The key detection algorithm is implemented as follows:

```
        char get_key(void) {
            uint8_t row_pins[5] = {ROW1_PIN, ROW2_PIN, ROW3_PIN,
               ROW4_PIN, ROW5_PIN};
            volatile uint8_t *row_ports[5] = {&ROW1_PORT, &ROW2_PORT
               , &ROW3_PORT, &ROW4_PORT, &ROW5_PORT};

            uint8_t col_pins[5] = {COL1_PIN, COL2_PIN, COL3_PIN,
               COL4_PIN, COL5_PIN};
            volatile uint8_t *col_pin_regs[5] = {&COL1_PIN_REG, &
               COL2_PIN_REG, &COL3_PIN_REG, &COL4_PIN_REG, &
               COL5_PIN_REG};

            for (uint8_t i = 0; i < 5; i++) {
                // Set current row low
                *row_ports[i] &= ~(1 << row_pins[i]);

                for (uint8_t j = 0; j < 5; j++) {
                    // Check if column is low (button pressed)
                    if (!(*col_pin_regs[j] & (1 << col_pins[j]))) {
                        _delay_ms(50); // Debounce
```

```
                    // Wait for button release
                    while (!(*col_pin_regs[j] & (1 << col_pins[j
                        ])));

                    // Set row back to high
                    *row_ports[i] |= (1 << row_pins[i]);

                    return keys[i][j];
                }
            }

            // Set row back to high
            *row_ports[i] |= (1 << row_pins[i]);
        }

        return '\0';
    }
```

7 Conclusion

The Arduino-based scientific calculator project successfully demonstrates the application of embedded systems principles to create a functional mathematical tool. Through the integration of a 5×5 keypad matrix, 16×2 LCD display, and carefully crafted embedded C code, the calculator provides a comprehensive set of mathematical functions in a compact and user-friendly package.

7.1 Project Achievements

Key achievements of this project include:

- Successful implementation of a fully functional scientific calculator
- Efficient I/O management, using only 10 pins to control a 25-key keypad
- Implementation of advanced mathematical functions through AVR libc
- User-friendly interface with shift functionality to double the number of available functions
- Reliable operation with debouncing and error handling

7.2 Learning Outcomes

This project provided valuable experience in several areas:

- Low-level microcontroller programming using embedded C

- Hardware interfacing techniques for LCD displays and keypad matrices
- Implementation of mathematical algorithms on resource-constrained systems
- Efficient memory usage and optimization techniques
- Practical application of debouncing and user interface design principles

7.3 Future Improvements

Several potential enhancements could be made to the calculator in future iterations:

- Implementation of a more sophisticated expression parser for handling complex equations
- Addition of bracket support for nested expressions
- Implementation of memory storage functions for saving and recalling values
- Addition of unit conversion capabilities
- Migration to a larger display for showing more of the expression and calculation history
- Development of a custom PCB and enclosure for a more professional finish

In conclusion, this project demonstrates that even with limited hardware resources, a microcontroller-based system can provide powerful calculation capabilities comparable to commercial scientific calculators. The modular design approach allows for future expansion and improvement while maintaining compatibility with the existing hardware platform.

Note

i referred this code from ROHITH (EE24BTECH11061)