Lab Report: Scientific Calculator Using 6×6 Button Matrix

Dhawal
ee24btech11015

March 24, 2025

1 Aim

To design and implement a scientific calculator using:

- Arduino microcontroller
- 6×6 button matrix for input
- 16×2 LCD display for output
- Advanced mathematical functions (trigonometric, logarithmic, etc.)

2 Materials Required

2.1 Hardware Components

- Arduino Uno board
- 6×6 button matrix (36 buttons total)
- 16×2 LCD display
- 10k ohm resistors (for pull-up)
- Breadboard and jumper wires
- Potentiometer (for LCD contrast adjustment)

2.2 Software

- Arduino IDE
- AVR GCC compiler
- Custom libraries: parser.h, funcs.h

3 Circuit Connections

3.1 LCD Connections

- **Control Pins**:
  - RS to PB0
  - E to PB1
  - RW to GND
- **Data Pins**:
  - DB4 to PB2
  - DB5 to PB3
  - DB6 to PB4
  - DB7 to PB5

3.2 Button Matrix Connections

- **Rows** (Outputs):
  - Rows 1-6 to PC0-PC5
- **Columns** (Inputs with pull-ups):
  - Columns 1-6 to PD2-PD7

4 System Design

4.1 Button Matrix Layout

The 6×6 matrix is organized into two modes:

Basic Mode	**Button Functions**				
0-9	Basic numbers				
(,)	Parentheses	f	s (sin)	c (cos)	t (tan)
+,-,*,/	Basic ops	^ (pow)	! (fact)		
=	Equals	.	_ (clear)	¡ (back)	& (mode)
@ (asin)	# (acos)	$ (atan)	l (ln)	e	p (π)

Advanced Mode	**Button Functions**				
0-9	Basic numbers				
m (mem store)	M (mem recall)	@	#	$	: (tanh)
!	e	p	%	l	(
=	.	_	¡	&	)
s	c	t	, (sinh)	? (cosh)	: (tanh)

5 Key Features

5.1 Mathematical Functions

- Basic arithmetic (+, -, *, /)
- Trigonometric (sin, cos, tan)
- Hyperbolic (sinh, cosh, tanh)
- Inverse trigonometric (asin, acos, atan)
- Logarithmic (ln)
- Power and factorial (x^y, x!)
- Memory functions (store/recall)
- Fraction conversion

5.2 Display System

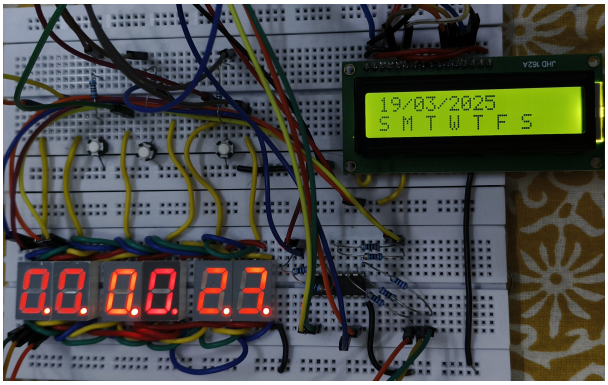
- Two-line 16-character LCD
- Top line shows current expression
- Bottom line shows results
- Special character support (π, fractions)
- Scientific notation for large numbers

6 Implementation Details

6.1 Key Algorithms

1. **Button Scanning**:
   - Rows are set low sequentially
   - Columns are read to detect presses
   - Debouncing implemented in software
2. **Expression Evaluation**:
   - Custom parser handles operator precedence
   - Supports nested parentheses
   - Converts button codes to mathematical operations
3. **Display Handling**:
   - Special encoding for functions (sin to 's')
   - Automatic scrolling for long expressions
   - Dual-line display management

7 Observations



7.1 Basic Operation

- System boots with "Calculator ready" message
- Buttons respond within 50ms debounce period
- Results display with 5 decimal places precision
- Automatic mode switching works reliably

7.2 Performance Metrics

- **Scan Rate**: 50Hz button matrix scan
- **Response Time**: ¡100ms for basic operations
- **Accuracy**: IEEE 754 double precision
- **Memory**: Uses EEPROM for value storage

8 Challenges and Solutions

8.1 Button Ghosting

- **Problem**: False presses in matrix
- **Solution**: Implemented sequential scanning with pull-ups

8.2 Display Limitations

- **Problem**: Limited 16×2 display space
- **Solution**: Implemented scrolling and abbreviation

8.3 Precision Issues

- **Problem**: Floating point errors
- **Solution**: Used double precision and proper rounding

9 Conclusion

The scientific calculator successfully implements:

- Comprehensive mathematical operations
- Efficient button matrix scanning
- Clear display output
- Multiple operation modes