

Scientific Calculator Using Arduino Uno

Your Name

May 25, 2025

Abstract

This project demonstrates the implementation of a scientific calculator using an Arduino Uno, a 16x2 LCD, and a 5x5 button matrix. The device evaluates complex mathematical expressions, including nested functions and operations with full BODMAS precedence, using custom numerical methods and parsing techniques.

Hardware Description

- **Microcontroller:** Arduino Uno R3
- **Display:** 16x2 LCD (JHD162A) connected to pins 7 to 12
- **Input:** 5x5 matrix keypad using tactile switches
- **Pins Used:**
 - Rows: Digital Pins 2 to 6
 - Columns: Analog Pins A0 to A4
 - LCD: Pins 7 (RS), 8 (EN), 9-12 (D4-D7)
- **Power:** USB or external adapter via Arduino

Software Overview

The calculator firmware is written in Arduino C++ using the `LiquidCrystal` library for display and custom logic for expression parsing. The code includes:

- Button matrix scanning
- LCD display handling
- Recursive expression parser with BODMAS logic
- Numerical implementations of sin, cos, log, sqrt, cbrt

Code Explanation

1. Library and LCD Setup

```
1 #include <LiquidCrystal.h>
2 LiquidCrystal lcd(7, 8, 9, 10, 11, 12);
```

This includes the LiquidCrystal library and defines the LCD pins connected to Arduino.

2. Keypad and Mapping

```
1 const int rowPins[5] = {2, 3, 4, 5, 6};
2 const int colPins[5] = {A0, A1, A2, A3, A4};
3 const char buttonMap[5][5] = {
4     {'0', '1', '2', '3', '4'},
5     {'5', '6', '7', '8', '9'},
6     {'+', '-', '*', '/', '='},
7     {'s', 'c', 'l', 'q', 'r'},
8     {'(', ')', '.', 'p', 'C'}
9 };
```

Each button in the 5x5 matrix is mapped to a character (digits, operators, functions, etc).

3. Initialization

```
1 void setup() {
2     lcd.begin(16, 2);
3     lcd.print("Sci Calculator");
4     delay(1500);
5     lcd.clear();
6     for (int i = 0; i < 5; i++) {
7         pinMode(rowPins[i], OUTPUT);
8         digitalWrite(rowPins[i], HIGH);
9     }
10    for (int j = 0; j < 5; j++) {
11        pinMode(colPins[j], INPUT_PULLUP);
12    }
13 }
```

Initializes the LCD and sets up the keypad: rows as outputs, columns with pull-ups.

4. Main Loop and Key Scan

```
1 void loop() {
2     char key = scanKeypad();
3     if (key != '\0') {
```

```
4     handleKeyPress(key);
5 }
6 }

1 char scanKeypad() {
2     for (int i = 0; i < 5; i++) {
3         digitalWrite(rowPins[i], LOW);
4         for (int j = 0; j < 5; j++) {
5             if (digitalRead(colPins[j]) == LOW) {
6                 delay(200);
7                 while (digitalRead(colPins[j]) == LOW);
8                 digitalWrite(rowPins[i], HIGH);
9                 return buttonMap[i][j];
10            }
11        }
12        digitalWrite(rowPins[i], HIGH);
13    }
14    return '\0';
15 }
```

Scans each row by pulling it low, reads the columns to detect which button is pressed.

5. Handling Key Press

```
1 void handleKeyPress(char key) {
2     if (key == 'C') {
3         inputExpression = "";
4         lcd.clear();
5     } else if (key == '=') {
6         float result = evaluateExpression(inputExpression);
7         lcd.clear();
8         if (result == -9999) lcd.print("Error");
9         else lcd.print(result);
10        inputExpression = "";
11    } else if (key == 'p') {
12        inputExpression += "3.14159";
13        lcd.print("3.14159");
14    } else {
15        inputExpression += key;
16        lcd.print(key);
17    }
18 }
```

Handles building the expression string, clearing it, evaluating it, or inserting π .

6. Expression Evaluation

```
1 float evaluateExpression(String expr) {
2     for (int i = 0; i < expr.length(); i++) {
3         if (expr[i] == 's' || expr[i] == 'c' || ...) {
4             int startIndex = i + 1;
5             if (expr[startIndex] != '(') return -9999;
6             int endIndex = findMatchingClosingParenthesis(expr,
7                 ↪ startIndex);
8             ...
9             float argument = evaluateExpression(argumentStr);
10            float computedValue = computeFunction(expr[i], argument);
11            expr = expr.substring(0, i) + String(computedValue) + expr.
12                ↪ substring(endIndex + 1);
13            i = -1;
14        }
15    }
16    return parseBODMAS(expr);
17 }
```

Handles recursive parsing for nested functions like `sin(log(90))`.

7. Compute Math Functions

```
1 float computeFunction(char func, float value) {
2     switch (func) {
3         case 's': return numericalSin(...);
4         case 'c': return numericalCos(...);
5         case 'l': return numericalLog(value);
6         case 'q': return numericalSqrt(value);
7         case 'r': return numericalCubeRoot(value);
8         default: return -9999;
9     }
10 }
```

8. Parsing BODMAS

```
1 float parseBODMAS(String expr) {
2     int startParen = expr.indexOf('(');
3     while (startParen != -1) {
4         int endParen = findMatchingClosingParenthesis(expr, startParen)
5             ↪ ;
6         String subExpr = expr.substring(startParen+1, endParen);
7         float subResult = parseBODMAS(subExpr);
8         expr = expr.substring(0, startParen) + String(subResult) + expr
9             ↪ .substring(endParen+1);
10        startParen = expr.indexOf('(');
11    }
```

```
9     }
10    return evaluateBasicBODMAS(expr);
11 }
```

Recursively evaluates expressions inside parentheses before computing the final result.

9. Operator Precedence Evaluation

```
1 float evaluateBasicBODMAS(String expr) {
2     // +, -, *, / handled with correct order
3     ...
4 }
```

Implements a simplified operator precedence evaluator without using stacks.

10. Numerical Math Implementations

```
1 float numericalSin(float x) { ... }
2 float numericalCos(float x) { ... }
3 float numericalLog(float x) { ... }
4 float numericalSqrt(float x) { ... }
5 float numericalCubeRoot(float x) { ... }
```

All functions are implemented using numerical methods like integration or Newton-Raphson.

Conclusion

This calculator is capable of evaluating highly complex expressions, supports function nesting, and mimics real scientific calculator behavior. With plans to port this to AVR-GCC, it also serves as a strong embedded systems and parsing logic demonstration.

Links

- GitHub: <https://github.com/mr21sk/my-projects/tree/main/Scientific-Calculator>

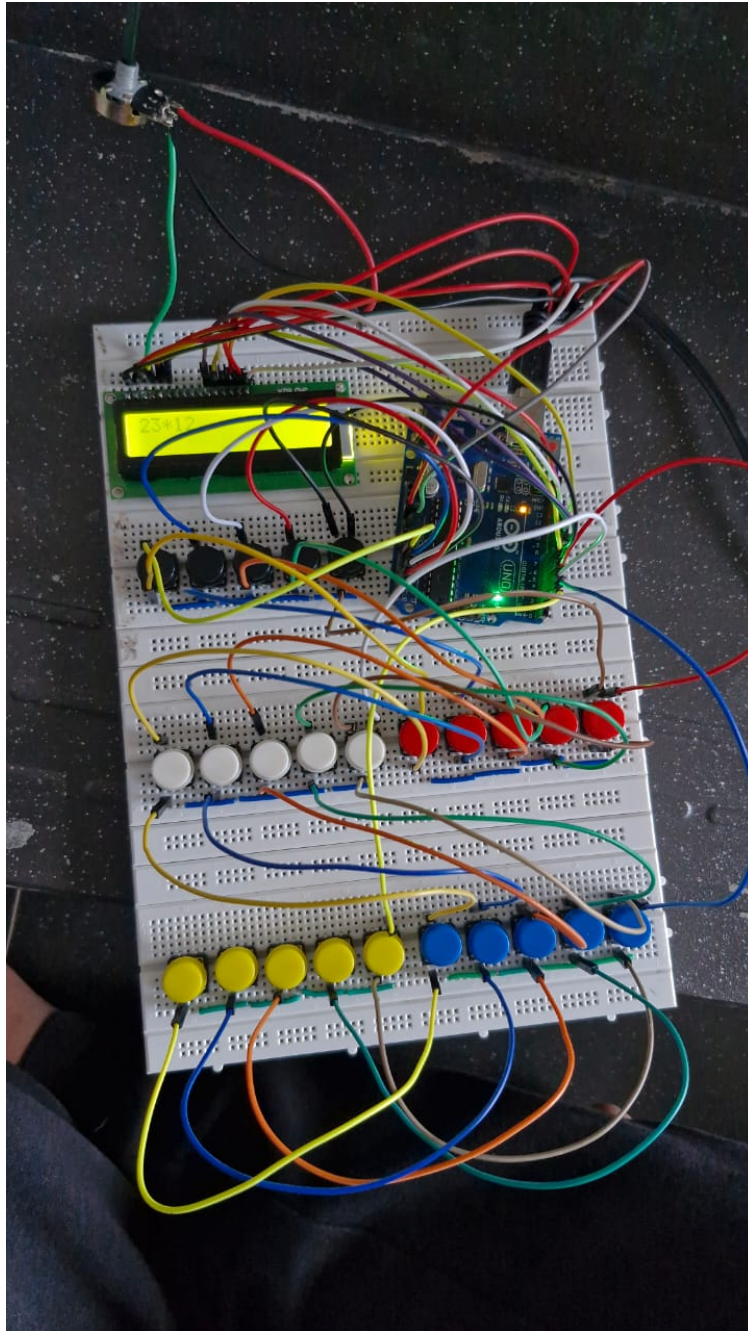


Figure 1: Scientific Calculator using Arduino