Scientific Calculator Project Report

EE224BTECH11044 - Muthyala koushik

1 Introduction

This project implements a scientific calculator using the Arduino Uno. It supports basic arithmetic, trigonometric (and inverse) functions, logarithmic/exponential operations, and complex expressions with proper operator precedence. A JHD162A LCD display shows the results, while push buttons serve as input. A recursive descent parser evaluates expressions accurately and efficiently.

2 Aim

- Design a calculator that performs:
  - Arithmetic: `+`, `-`, `*`, `/`, `^`
  - Trigonometry: `sin`, `cos`, `tan` (in degrees)
  - Inverse Trigonometry: `asin`, `acos`, `atan` (in degrees)
  - Log/Exp: `ln`, `log`, `exp`
  - Others: `sqrt`, `abs`, `%`
- Evaluate expressions using a recursive descent parser with proper precedence and parentheses.
- Provide real-time feedback via a JHD162A LCD display.
- Optimize resource use on the Arduino Uno.

3 Components Required

- Arduino Uno
- JHD162A LCD display
- 14 Push buttons:
  - 10 for digits (0–9)
  - Clear (`C`) and Enter (`=`)
  - 2 Shift buttons: Arithmetic Shift (Shift-A) and Scientific Shift (Shift-S)
- Wires, breadboard

4 Hardware Configuration

4.1 Pin Connections

4.1.1 LCD Display (4-bit Mode)

LCD Pin	Function	Arduino Pin
RS	Register Select	D8 (PB0)
E	Enable	D9 (PB1)
D4	Data Bit 4	D10 (PB2)
D5	Data Bit 5	D11 (PB3)
D6	Data Bit 6	D12 (PB4)
D7	Data Bit 7	D13 (PB5)

Additional connections: VSS (GND), VDD (+5V), VO (potentiometer), RW (GND), A (+5V backlight), K (GND).

4.1.2 Push Button Connections

Button	Function	Arduino Pin
Digits 0–5	Enter digits 0–5	D2, D3, D4, D5, D6, D7
Digits 6–9	Enter digits 6–9	A0, A1, A2, A3
Clear (`C`)	Clear input	A4
Enter (=)	Evaluate expression	A5
Extra Button 1	Additional function	(Assign as needed)
Extra Button 2	Additional function	(Assign as needed)

4.1.3 Shift Button Connections

Button	Function	Arduino Pin
Arithmetic Shift (Shift-A)	Toggle arithmetic mode	D0 (PD0)
Scientific Shift (Shift-S)	Toggle scientific mode	D1 (PD1)

4.1.4 Power Supply

Supply	Connection
VCC	+5V
GND	Ground

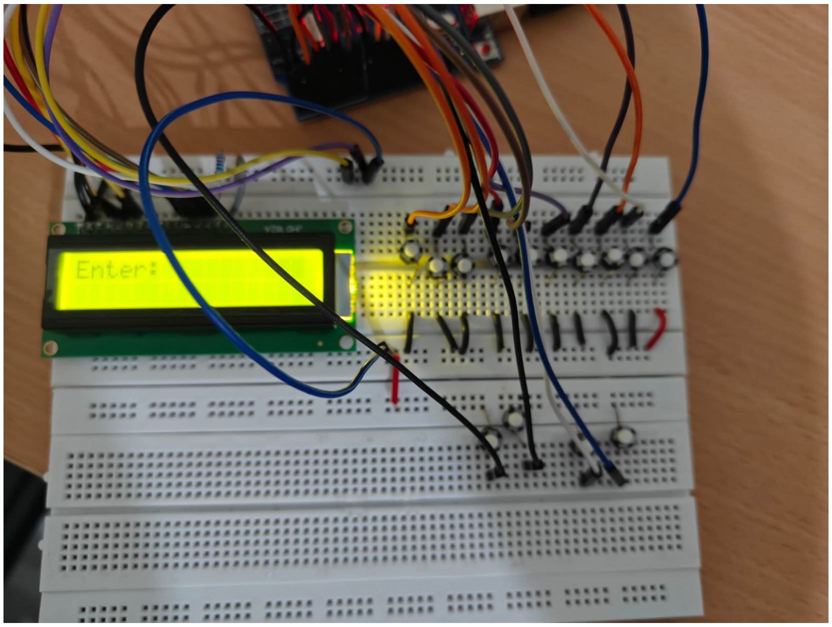

Figure 0: Circuit diagram of the scientific calculator using Arduino Uno and JHD162A LCD

5 Software Design

5.1 Overall Architecture

The software is modular, divided into:

a) **Input Handling**: Scans push buttons, applies a 30 ms debounce delay, and detects mode switches.

b) **User Interface (UI)**: Updates the LCD with the current expression, mode menus, and results. It initializes the LCD (4-bit mode), clears the screen, and sets the cursor.

c) **Expression Evaluation**: Implements a recursive descent parser that enforces operator precedence, handles parentheses, and supports function calls.

5.2 Input Handling

- **Button Scanning**: Uses `digitalRead()` to poll each push button.
- **Debouncing**: A 30 ms delay is applied to prevent false triggers.
- **Mode Switching**: Shift-A and Shift-S buttons toggle between arithmetic and scientific modes.

5.3 User Interface (UI)

- **LCD Initialization**: Functions like `lcd_init()`, `lcd_clear()`, and `lcd_setCursor()` prepare the display.
- **Dynamic Display**: The current expression and mode menus are shown in real time. Error messages and results (to 4 decimal places) are displayed as needed.

5.4 Expression Evaluation (Recursive Descent Parser)

- **Operator Precedence**:
  1. Level 1: `+` and `-`
  2. Level 2: `*`, `/`, and `%`
  3. Level 3: `^` (right-associative)
- **Parentheses**: Extraneous outer parentheses are removed.
- **Function Support**: Recognizes and evaluates:
  - Trigonometric: `sin`, `cos`, `tan` (in degrees)
  - Inverse Trigonometric: `asin`, `acos`, `atan` (in degrees)
  - Log/Exp: `exp`, `ln`, `log`
  - Others: `sqrt`, `abs`
- **Numerical Methods**: Utilizes Euler's method, Newton-Raphson, and Riemann sums for approximations.

5.5 Shift Modes

- **Arithmetic Shift Mode (Shift-A)**:
  - **Navigation**: Use buttons 8 (Next) and 9 (Prev) to switch pages.
  - **Mapping**:
    - Page 0: `+`, `-`, `*`, `/`
    - Page 1: `^`, `%`, `(`, `)`
    - Page 2: Backspace (BS)
- **Scientific Shift Mode (Shift-S)**:
  - **Navigation**: Use buttons 8 and 9 for page control.
  - **Mapping**:
    - Page 0: `sin`, `cos`, `tan`, `exp`
    - Page 1: `ln`, `sqrt`, `log`, `abs`
    - Page 2: `asin`, `acos`, `atan`

The LCD displays the current mode and page for user guidance.

5.6 AVR GCC

The firmware is compiled using AVR GCC, part of the AVR toolchain. AVR GCC enables code optimization for size and speed on resource-constrained systems like the Arduino Uno. It supports inline assembly for performance-critical sections and works in conjunction with avr-libc and avrdude for programming the microcontroller.

5.7 Error Handling

- Checks for division by zero, mismatched parentheses, and domain errors (e.g., `sqrt` of a negative number).
- Invalid selections in shift modes trigger brief error messages.

6 Operation Workflow

1. **Startup**: The LCD shows ”Simple Calc” for 2 seconds, then displays ”Enter:”.
2. **Normal Mode**: Users input digits and operators; pressing = evaluates the expression.
3. **Shift Modes**: Toggled by Shift-A (D0) or Shift-S (D1); navigation with buttons 8/9; selection with buttons 0–3.
4. **Result**: Displayed to 4 decimal places; new input clears the result.

7 Performance Specifications

- **Voltage**: 5V DC
- **Current**: ~50 mA (active), <10 mA (idle)
- **Precision**: IEEE 754 emulation; results to 4 decimal places.
- **Expression Limit**: 64 characters.
- **Speed**: Arithmetic <10 ms; trigonometric ~200 ms.

8 AVR Code Explanation

The embedded C code for the Arduino Uno is structured modularly to manage LCD interfacing, button inputs, expression parsing, and evaluation. The key components of the code are described below.

8.1 1. LCD Initialization and Display

The LCD used is a JHD162A, interfaced in 4-bit mode using Arduino Uno’s digital pins D8–D13. Initialization is done using `lcd_init()`, which sends the necessary commands for enabling display, clearing the screen, and setting cursor behavior.

- `lcd_cmd(uint8_t cmd)`: Sends control commands (e.g., clear, home).
- `lcd_data(char ch)`: Sends a character to be printed on screen.
- `lcd_print(char *s)`: Prints a full string on the LCD.

8.2 2. Button Scanning and Input

Buttons are connected to D2–D7 and A0–A5, and are scanned using the `read_buttons()` function. The button press is debounced and mapped to a character or command such as digit, operator, clear, or shift.

- Inputs are buffered into a character array `expr[64]` as the user types.
- Each character is appended until '=' is pressed to trigger evaluation.

8.3 3. Expression Parsing and Evaluation

A recursive descent parser is implemented to handle operator precedence, parentheses, and scientific functions. The input is parsed using the following hierarchy:

- `parse_expression()`: Handles '+' and '-'
- `parse_term()`: Handles '*', '/', and modulus
- `parse_factor()`: Handles numbers, parentheses, and powers
- `parse_function()`: Handles function names like `sin`, `log`, `sqrt`, etc.

The result is computed and returned to be displayed on the LCD.

8.4 4. Scientific Functions (Manual Implementation)

Common scientific functions are implemented manually using numerical approximations:

- `sin_deg(double x)`: Computes $\sin(x)$ using Taylor series in degrees.
- `log_custom(double x)`: Approximates $\log_{10}(x)$ using iterative or series methods.
- `sqrt_custom(double x)`: Uses Newton-Raphson method for square roots.

This approach avoids using standard math libraries, reducing code size and allowing better control on AVR.

8.5 5. Shift Modes for Multi-Function Input

To maximize button usage, shift modes are introduced:

- `Shift-A (Arithmetic)` and `Shift-S (Scientific)` buttons change input context.
- With 2 shift modes and multiple pages, a small number of buttons can cover all necessary operations.
- The mapping is handled using flags and arrays that define the current page and mode.

8.6 6. Display and Error Handling

The calculator continuously updates the display with current input. Upon pressing '=', the expression is evaluated and the result is shown with 4-digit precision.

- Errors like division by zero, invalid syntax, or domain errors (e.g., $\sqrt{-1}$) are detected and a message like `Error` is displayed temporarily.

8.7 7. Main Loop

The `main()` function initializes the LCD and runs an infinite loop to manage input and output.

```
int main() {
    lcd_init();
    show_welcome();
    while (1) {
        handle_input();
        update_lcd();
    }
}
```

- `handle_input()`: Reads buttons and updates the expression.
- `update_lcd()`: Refreshes the display based on the current state.