Calculator Project Report

EE24BTECH11031-Jashwanth.Medamoni

1 Project Overview

This project presents the design and implementation of an advanced scientific calculator using the Arduino Uno. It supports basic arithmetic, trigonometric, logarithmic, and exponential functions, alongside complex expressions with operator precedence. User input is managed through push buttons, while a JHD162A LCD display presents the results. A recursive descent parser ensures accurate and efficient expression evaluation.

2 Project Objectives

- Develop a versatile calculator that handles:
  - Arithmetic: +, -, *, /, ^
  - Trigonometric: `sin`, `cos`, `tan` (degrees)
  - Inverse Trigonometry: `asin`, `acos`, `atan` (degrees)
  - Logarithmic/Exponential: `ln`, `log`, `exp`
  - Miscellaneous: `sqrt`, `abs`, `%`
- Implement a recursive descent parser with operator precedence and parentheses support.
- Deliver real-time results on an LCD display.
- Optimize Arduino Uno's resource usage for smooth performance.

3 Required Components

- Arduino Uno microcontroller
- JHD162A 16x2 LCD display
- 14 Push buttons:
  - 10 for digits (0–9)
  - Clear button (`C`) and Enter button (=)
  - Arithmetic Shift (Shift-A) and Scientific Shift (Shift-S) buttons
- Wires and breadboard for circuit assembly

4 Hardware Configuration

4.1 Connections

4.1.1 LCD Display

LCD Pin	Arduino pin	Notes
RS	D8 (PB0)	LOW: Command; HIGH: Data
E	D9 (PB1)	Latches data
D4	D10 (PB2)	4-bit interface
D5	D11 (PB3)	4-bit interface
D6	D12 (PB4)	4-bit interface
D7	D13 (PB5)	4-bit interface

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

4.1.2 Button Connections

Button	Arduino Pin	Notes
Digits 0–5	D2, D3, D4, D5, D6, D7	Active-low; internal pull-ups
Digits 6–9	A0, A1, A2, A3	Digital inputs; pull-ups enabled
Clear (C)	A4	Resets expression
Enter (=)	A5	Triggers computation
Extra Button 1	(Assign as needed)	
Extra Button 2	(Assign as needed)	

Note: The extra two push buttons can be used for further functions or for shift modes if needed.

4.1.3 Shift Connections

Button	Arduino Pin	Notes
Arithmetic Shift (Shift-A)	D0 (PD0)	Repurposed from serial RX
Scientific Shift (Shift-S)	D1 (PD1)	Repurposed from serial TX

5 System Configuration

5.1 LCD and Button Setup

The LCD operates in 4-bit mode, with data lines connected to digital pins D8 through D13 on the Arduino. Push buttons are linked to pins D2-D7 and A0-A5, using internal pull-ups for active-low detection.

5.2 Power Supply

The system runs on 5V DC, drawn from USB or an external power source connected to the Arduino.

6 Software Architecture

The software follows a modular design, comprising:

a) **Input Management**: Reads button presses with debouncing to ensure reliable detection.

b) **User Interface (UI)**: Updates the LCD dynamically with expressions, modes, and results.

c) **Expression Parser**: Implements recursive descent parsing with operator precedence and function handling.

7 Modes of Operation

The calculator supports two operational modes:

- **Arithmetic Mode**: Handles basic operations and parentheses.
- **Scientific Mode**: Supports trigonometric, inverse trigonometric, logarithmic, and exponential functions.

Mode switching is achieved using Shift-A and Shift-S buttons, with navigation through additional button presses.

8 Error Handling Mechanisms

The system detects and handles errors such as:

- Division by zero
- Mismatched parentheses
- Domain errors (e.g., `sqrt(-1)`)
- Invalid mode selections

9 Performance and Limitations

- **Voltage**: Operates on 5V
- **Precision**: Displays results up to 4 decimal places
- **Expression Length**: Supports up to 64 characters
- **Speed**: Basic operations complete within 10ms, complex calculations take 200ms

10 Code Breakdown

The code consists of separate modules for LCD control, button handling, expression evaluation, and shift mode management. The main loop continuously monitors inputs and updates the display accordingly.

11 AVR GCC

The firmware is compiled using **AVR GCC**, which is part of the AVR toolchain. AVR GCC provides code optimization for both size and speed, making it ideal for resource-constrained systems like the *Arduino Uno*. It supports inline assembly for performance-critical sections, enhancing execution speed. Additionally, it integrates seamlessly with `avr-libc` for standard C functions and `avrdude` for flashing the compiled code to the microcontroller.