Project Documentation

"NumNinja"

Prepared By

Eng. Eslam Osama Saad

Table of Contents

(1) System Description:

(1.1) System Overview

An embedded calculator is a handy tool made to do basic math operations like addition, subtraction, multiplication, and division with two or more numbers. Unlike regular calculators, it's built right into various devices and apps, giving quick and easy access to math functions. This is especially helpful in areas like engineering, finance, and education, where fast and accurate calculations are needed. The user-friendly design makes it simple to enter numbers and get results quickly. Whether it's part of software, websites, or hardware, this calculator boosts productivity by making complex calculations easier and ensuring accuracy in everyday math tasks. The following Figure 1 is the implementation of our described embedded calculator on the simulation tool called proteus and it gives the initial look to our embedded calculator.

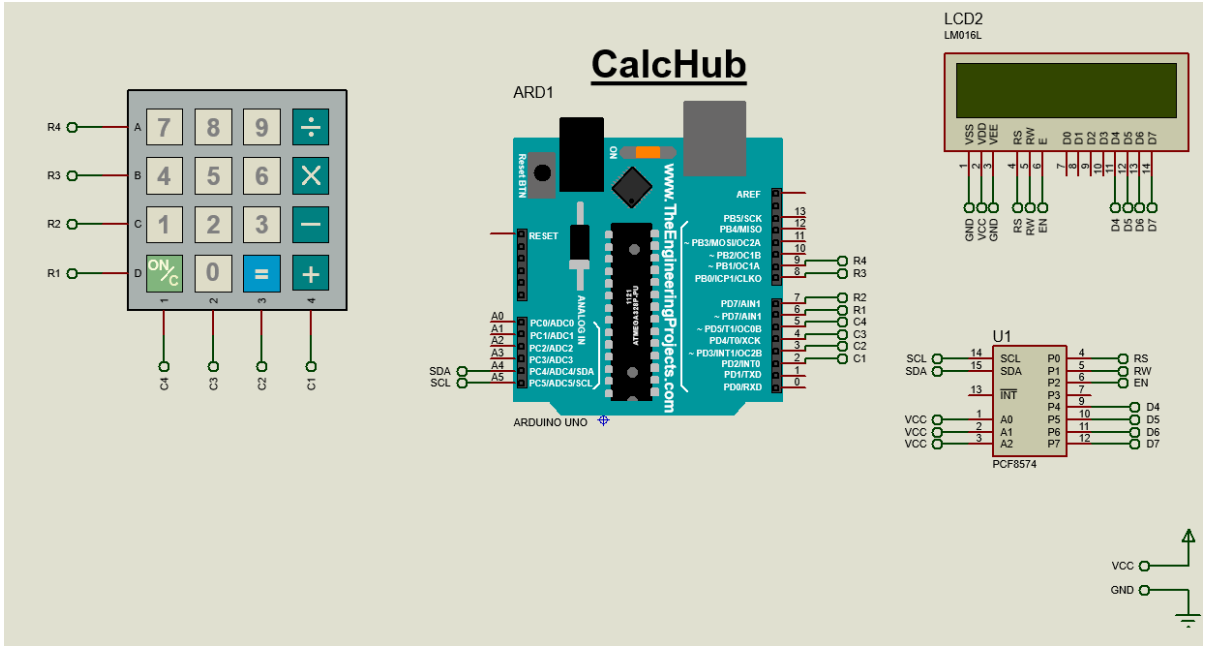

Fig.1 Proteus simulation of our NumNinja

(1.2) System Functionality

- Performs basic math operations: addition, subtraction, multiplication, and division.
- Handles calculations with two or more numbers.
- Integrated into various devices and applications.
- Provides quick and easy access to math functions.
- Useful in engineering, finance, and education.
- User-friendly design for easy number input and result retrieval.
- Enhances productivity by simplifying complex calculations.
- Ensures accuracy in everyday numerical tasks.

(2) System Design:

(2.1) Schematic Diagram

The following Figure 2 is the schematic diagram for a simple calculator that includes several key components: an Arduino Uno, a 4x4 Keypad, a breadboard, an I2C 2x16 LCD, and various connecting wires. The Arduino Uno acts as the brain of the calculator, receiving input from the 4x4 keypad. The keypad connects to the Arduino's digital pins. The Arduino processes these inputs to perform calculations and sends the results to the I2C 2x16 LCD for display. The breadboard is used to organize connections and distribute power. Male-to-male and male-to-female wires are used to make all necessary electrical connections between components. This setup allows for a functional calculator that can perform basic arithmetic operations and display results clearly.

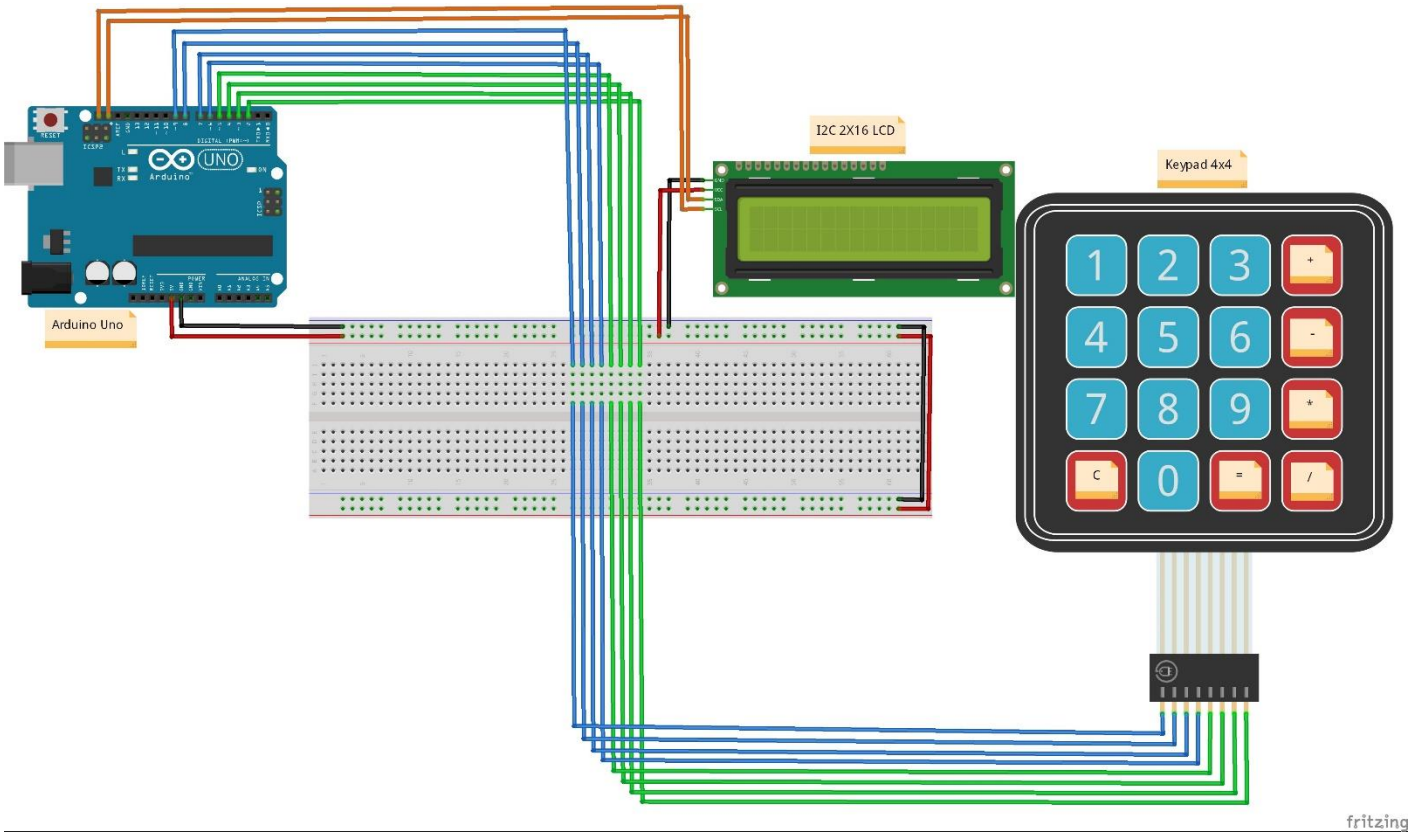

Fig. 2 Schematic Diagram of our NumNinja

(2.2) System Requirements

Components Description:

Arduino Uno: This is the main microcontroller that processes inputs and controls the display and calculations.

4x4 Keypad: This is used for inputting numbers and operations. It has 16 keys arranged in a 4x4 grid.

Breadboard: This is used to make temporary electrical connections between components.

I2C 2x16 LCD: This displays the numbers, operations, and results. The I2C interface simplifies wiring by reducing the number of pins needed.

Male to Male Wires: These connect components like KeyPad on the breadboard.

Male to Female Wires: These connect components like LCD to the Arduino Uno.

Connections:

- Keypad to Arduino Uno:
  - The 4x4 keypad has 8 pins (4 rows and 4 columns).
  - Connect these 8 pins to the digital pins on the Arduino (e.g., D2 to D9).

- LCD to Arduino Uno (via I2C):
  - The I2C LCD has 4 pins: VCC, GND, SDA, and SCL.
  - Connect VCC to the 5V pin on the Arduino.
  - Connect GND to a GND pin on the Arduino.
  - Connect SDA to the A4 pin on the Arduino.
  - Connect SCL to the A5 pin on the Arduino.

- Breadboard:
  - Use the breadboard to facilitate connections between the Arduino, keypad, and LCD.
  - Power and ground lines from the Arduino can be distributed using the breadboard.

- Power and Ground Connections:
  - Ensure all components share a common ground.
  - Use male-to-male and male-to-female wires to connect VCC and GND lines from the Arduino to the breadboard and then to other components as needed.

(2.3) System Constraints

Display Constraints:

Character Limit: The I2C 2x16 LCD can only display 32 characters at a time (16 characters per line), which limits the amount of information shown to the user.

Refresh Rate: The speed at which the LCD updates may not be fast enough for very rapid inputs or complex animations.

Keypad Limitations:

Debouncing: Mechanical keypads can have issues with bouncing, requiring software debouncing to ensure accurate input.

Limited Inputs: With only 16 keys, the keypad can only provide a limited number of input combinations, affecting the complexity of operations that can be performed.

Software Limitations:

Make sure to not entire variable more than max size of the variable's data type to avoid overflow. For example, we are using float data type and it contains 4 bytes which means not type value higher than $2^{(4*8)} - 1 = 4294967295$.

(2.4) Operating Environment

Temperature and Humidity: The operating environment should be within the acceptable range for all components to ensure reliable operation.

Electrical Noise: External electrical interference could affect the reliability of the keypad input or display output.

(2.5) Input & Output Formats

Inputs:

- we have variable inform of float datatype to store mathematical equation inputs, but whenever we want to print these variables on LCD we need to change it to array of chars or string.
- char datatype variable to take operator, equal sign, or ‘C’ which indicates clear LCD from user using Keypad.

Output:

- We have just one variable called result inform of float datatype to store result of mathematical operation, but before printing it on LCD, we need to apply explicitly typecasting to it from float to string or array of chars.

(3) FlowChart:

As shown in the following figure 3, the flowchart of an embedded calculator starts with setting up the Arduino, keypad, and LCD. It then waits for the user to enter the first number, digit by digit, displaying it on the first row of the LCD. The user can press 'C' to clear the display and start over. When an operator key is pressed, it appears on the second row of the LCD, and the user can press 'C' to clear it if needed.

Next, the system waits for 500 milliseconds, clears the LCD, and prompts the user to enter the next number, again digit by digit on the first row. The user can clear the display if needed. Pressing the equal sign clears the LCD and shows the result on the first row. For more complex operations, the user can press another operator key, clearing the

LCD and waiting for the next number. This process continues until the user presses the equal button to get the final result.

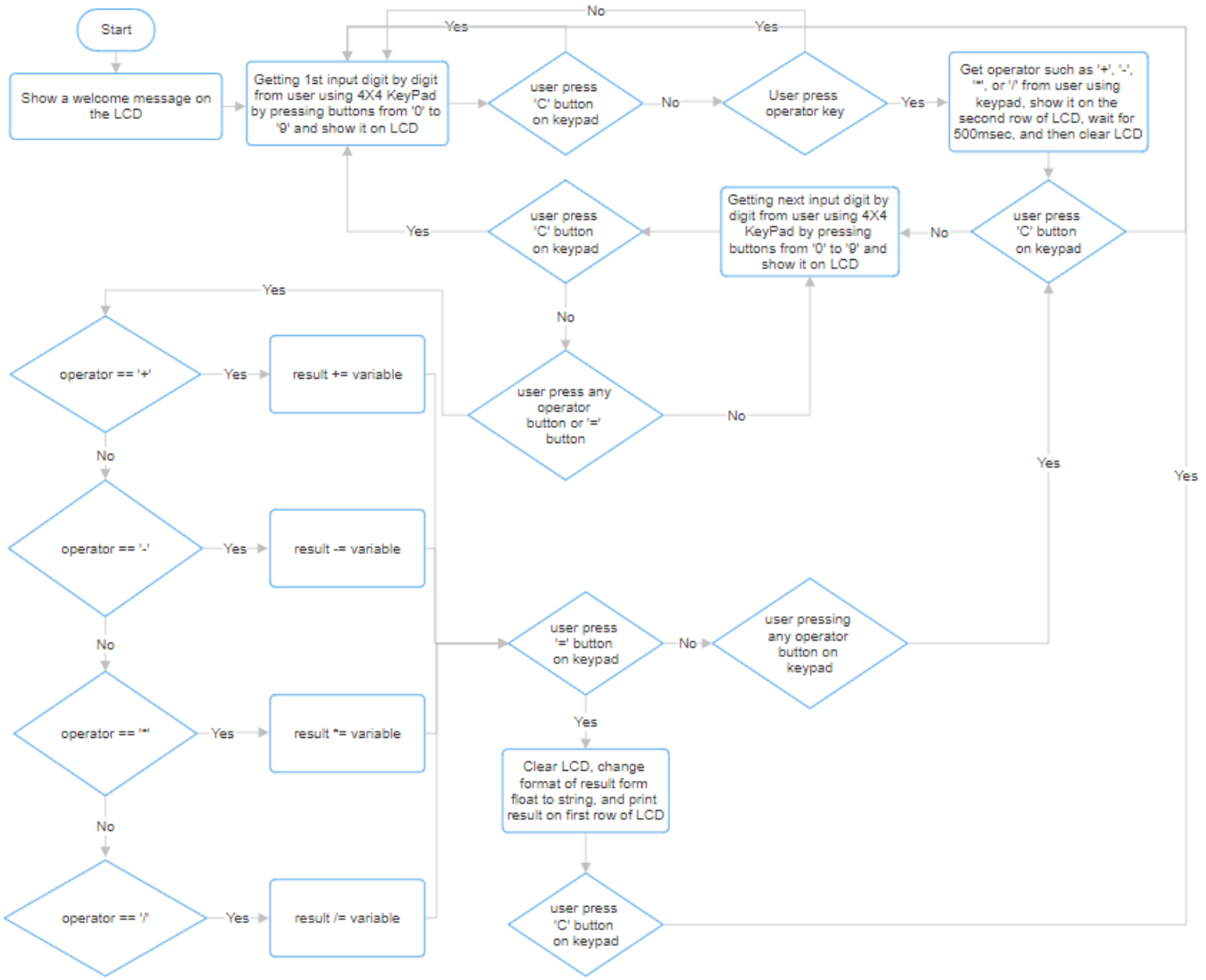

Fig. 3 Flowchart of our NumNinja