1. INTRODUCTION

The Arduino calculator is a project that utilizes the Arduino platform to create a calculator-like device. It can perform basic arithmetic operations such as addition, subtraction, multiplication, and division. The calculator can be built using various components, including buttons, an LCD display, and the Arduino board itself. By programming the Arduino board, I can define and even alter the behaviour of the calculator and customize its functionality.

OVERVIEW OF COMPONENTS:

Arduino Board: The Arduino board serves as the main controller for the calculator. It receives input from the user, performs calculations, and displays the results.

Power Supply: The Arduino board and other components require a power supply to operate. This can be provided through a USB connection, a battery, or an external power source.

Input Interface: The calculator requires an input interface for users to enter numbers and select operations. This can be achieved using buttons, a keypad, or a touchscreen.

Calculation Logic: The Arduino board runs a program that implements the logic for performing arithmetic operations. This program interprets the user's input and performs the corresponding calculations.

Display: The calculator needs a display to show the numbers entered by the user and the results of calculations. An LCD display is commonly used for this purpose.

1. PROBLEM DEFINITION

Supposing a hypothetical situation arises where there is a need for a calculator-like device that can perform basic arithmetic operations and provide a user-friendly interface for input and display.

Existing calculators in the market may not always meet the specific requirements of buyers. Some may prefer a customized calculator with specific features or functionalities. Additionally, traditional calculators may not be easily programmable or adaptable to different needs.

1. METHODOLOGY:

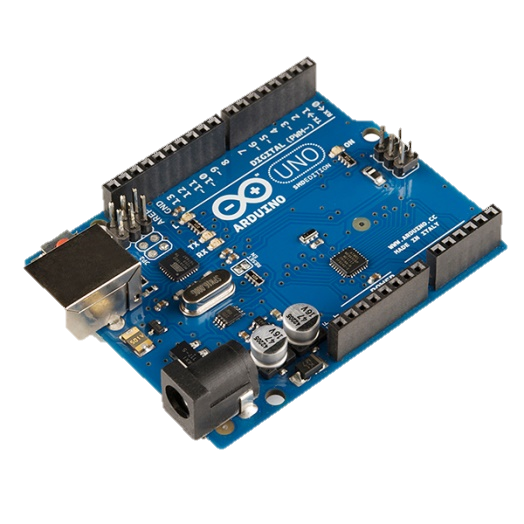
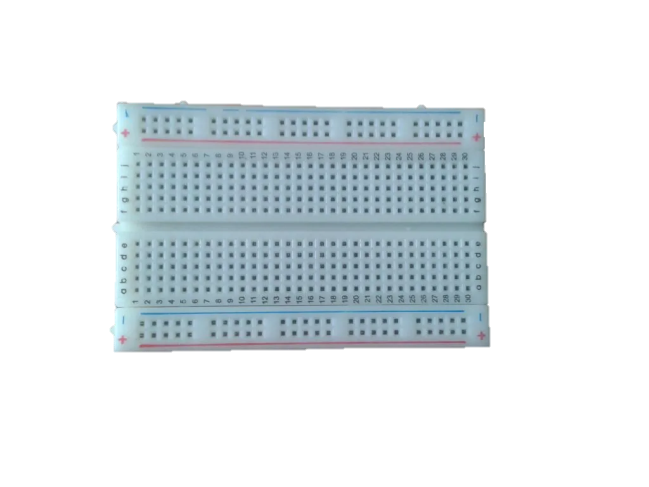
SETUP LAYOUT: Start by setting up the circuit on the breadboard, connect the potentiometer, lcd display, and Arduino uno using jumper wires. By following the circuit diagram provided in the 4x4 calculator blueprint and ensure all connections are secure and correct.

PROGRAM CODE: once the circuit is set up, you need to write the code to program the Arduino uno. Code includes the function for reading input from the potentiometer, performing arithmetic operations, and displaying the results on the lcd display. By utilizing Arduino IDE (integrated development environment) to write and upload the code to the Arduino uno.

TESTING AND DEBUGGING: after uploading the code to the Arduino uno, test the calculator by adjusting the potentiometer to input numbers and operations. verify that the calculator performs the desired arithmetic operations and displays the results accurately on the lcd display. If any issues or errors occur, debug the code and circuit connections to identify and resolve the problems.

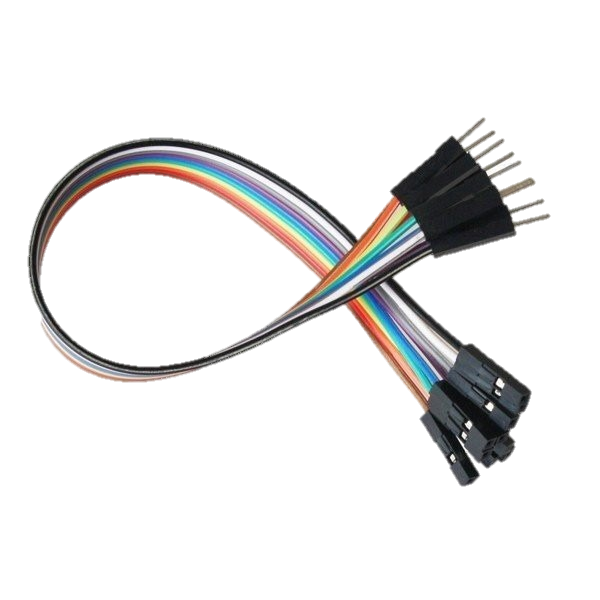
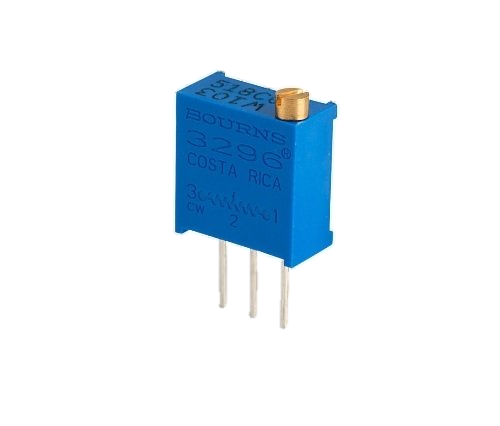
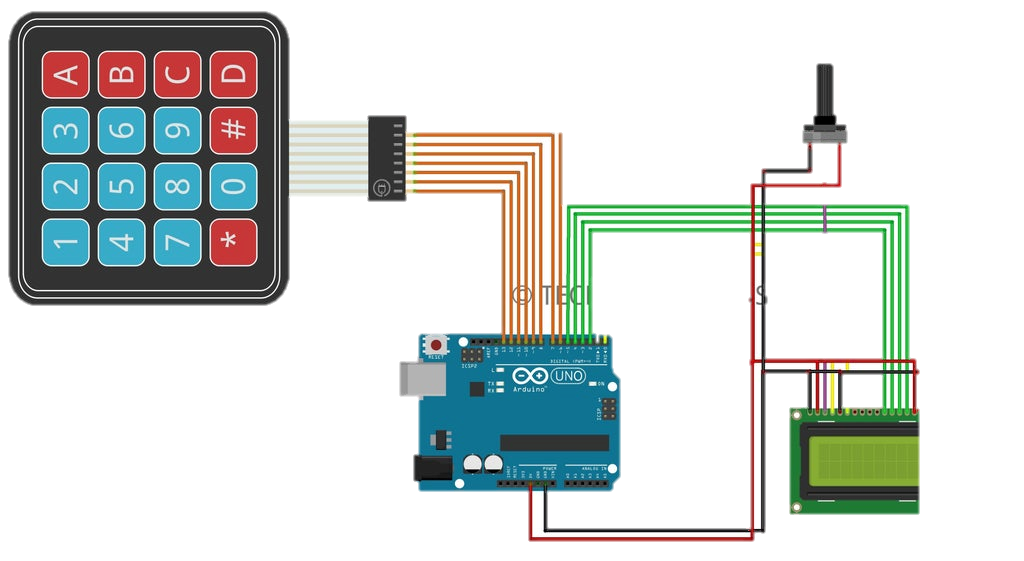
REFINING AND CUSTOMIZING: Once the basic functionality of the Arduino calculator is working correctly, I can refine and customize it further. This may include adding additional features, improving user interface design, or incorporating advanced functionalities. For example, exponentiation which is recursive multiplication.

IMAGES OF COMPONENTS BEING UTILIZED:

- Breadboard

- Arduino Uno board

- Jumper wires

-10k potentiometer

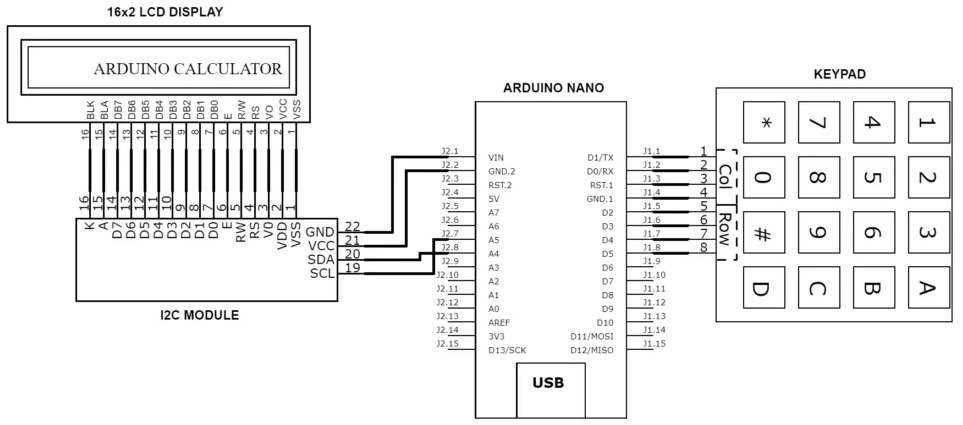
Arduino cable

-4x4 calculator blueprint

- LCD display

1. DETAILED DESIGN AND ARCHITECTURE

BLUEPRINT OF CIRCUIT + BLOCK DIAGRAM



HARDWARE COMPONENTS WITH RESPECTED FUNCTIONALITIES:

* Arduino Uno board: Center of the machine. Main microcontroller board that serves as the brain of the calculator. A perfect example would be a CPU of the calculator.
* Breadboard: It provides a platform for connecting and prototyping the circuit.
* 10k potentiometer: It can be used as an input device to adjust the contrast of the LCD display.
* Jumper wires: These wires are used to establish connections between different components.
* LCD display: A 16x2 LCD display is commonly used to show the numbers and results of calculations.
* 4x4 calculator blueprint: This is a keypad module that allows users to input numbers and perform calculations.
* Arduino cable: It is used to connect the Arduino board to a computer or power source.

SOFTWARE IMPLEMENTATION: The software implementation of the Arduino calculator involves writing code that runs on the Arduino board. The code is responsible for handling user input, performing calculations, and displaying the results on the LCD display. The code can be written using the Arduino IDE (Integrated Development Environment) and uploaded to the Arduino board.

ALGORITHM/DRY RUN FOR CALCULATIONS: The specific algorithm used for calculations in the Arduino calculator can vary depending on the implementation. Common algorithms for basic arithmetic operations like addition, subtraction, multiplication, and division can be used. These algorithms involve carrying, borrowing, and iterative operations to perform the calculations accurately.

SYSTEM ARCHITECTURE**:** The system architecture of the Arduino calculator can be divided into several components:

COMPONENT DIVISION

Input Component: The 4x4 calculator blueprint keypad module serves as the input component, allowing users to input numbers and perform calculations.

Processing Component: The Arduino Uno board acts as the processing component, executing the code that handles user input and performs calculations.

Display Component: The LCD display is used as the output component, showing the numbers and results of calculations to the user.

Communication Component: The Arduino board communicates with the keypad module and the LCD display to receive input and display output.

1. IMPLEMENTATION, TESTING AND PROGRAMMING CODE:

Software Development:

ARDUINO IDE: The Arduino Integrated Development Environment (IDE) was used as the primary software tool for writing and uploading code to the Arduino board [1](https://www.arduino.cc/en/Guide/Environment/). It provides a text editor, message area, console, and various functions and menus for programming and communication with the hardware.

LIBRARIES: Utilized to extend the functionality of the Arduino board. These libraries provide additional features for working with hardware components or manipulating data [1](https://www.arduino.cc/en/Guide/Environment/). In the provided code, the "Keypad" and "LiquidCrystal" libraries are used for keypad input and LCD display control, respectively.

CODE IMPLEMENTATION: The code provided implements the core functionalities of the calculator. It uses a keypad to input numbers and operators, an LCD display to show the input and result, and logic to perform arithmetic operations based on the user's input.

The software development for the Arduino calculator involves writing code to handle the keypad input, perform calculations, and display the results on the LCD display. The code uses the Keypad and Liquid Crystal libraries.

The code begins with including the necessary libraries:

#include <Keypad.h>

#include <LiquidCrystal.h>

The pin connections for the keypad and LCD display are defined:

const byte ROWS = 4;

const byte COLS = 4;

char keys[ROWS][COLS] = {

{'1', '2', '3', '+'},

{'4', '5', '6', '-'},

{'7', '8', '9', '\*'},

{'C', '0', '=', '/'}

};

byte rowPins[ROWS] = {13, 12, 11, 10};

byte colPins[COLS] = {9, 8, 7, 6};

The setup function initializes the LCD display and shows a welcome message:

void setup() {

lcd.begin(16, 2);

lcd.setCursor(0, 0);

lcd.print("DLD PROJECT 2024");

lcd.setCursor(0, 1);

lcd.print("Arduino Integrated Calculator");

delay(3000);

lcd.clear();

}

The loop function continuously checks for keypad input and performs the corresponding actions:

void loop() {

char key = myKeypad.getKey();

// Handle numeric input

if (key != NO\_KEY && (key == '1' || key == '2' || key == '3' || key == '4' || key == '5' || key == '6' || key == '7' || key == '8' || key == '9' || key == '0')) {

// Handle first number input

if (presentValue != true) {

num1 = num1 + key;

int numLength = num1.length();

lcd.setCursor(15 - numLength, 0);

lcd.print(num1);

}

// Handle second number input

else {

num2 = num2 + key;

int numLength = num2.length();

lcd.setCursor(15 - numLength, 1);

lcd.print(num2);

final = true;

}

}

// Handle operator input

else if (presentValue == false && key != NO\_KEY && (key == '/' || key == '\*' || key == '-' || key == '+')) {

if (presentValue == false) {

presentValue = true;

op = key;

lcd.setCursor(15, 0);

lcd.print(op);

}

}

// Handle calculation and display result

else if (final == true && key != NO\_KEY && key == '=') {

if (op == '+') {

answer = num1.toInt() + num2.toInt();

}

else if (op == '-') {

answer = num1.toInt() - num2.toInt();

}

else if (op == '\*') {

answer = num1.toInt() \* num2.toInt();

}

else if (op == '/') {

answer = num1.toInt() / num2.toInt();

}

lcd.clear();

lcd.setCursor(15, 0);

lcd.autoscroll();

lcd.print(answer);

lcd.noAutoscroll();

}

// Handle clear input

else if (key != NO\_KEY && key == 'C') {

lcd.clear();

presentValue = false;

final = false;

num1 = "";

num2 = "";

answer = 0;

op = ' ';

}

}

1. RESULTS/ SOFTWARE SIMULATION AND DISCUSSION

The Arduino calculator project has been successfully implemented, providing basic arithmetic operations through both hardware and software simulation. The software simulation was conducted using Arduino development tools, allowing for testing and validation of the calculator's functionality before deploying it onto hardware.

SOFTWARE SIMULATION RESULTS

During the software simulation phase, the Arduino code was developed and tested using an integrated development environment (IDE) such as Arduino IDE or similar platforms. Through simulation, the following results were observed:

ARITHMETIC OPERATIONS: Addition, subtraction, multiplication, and division operations were implemented and tested for accuracy and efficiency. The simulator accurately performed these operations based on user input and displayed the results appropriately.

USER INTERFACE INTERACTION: The user interface elements, including keypad input and LCD display output, were simulated to ensure proper interaction and feedback. Users were able to input numbers and select operations through the keypad, with corresponding results displayed on the LCD screen.

DISCUSSION

The software simulation phase provided valuable insights into the functionality and usability of the Arduino calculator project. Through testing and validation in a simulated environment, several aspects were discussed and analyzed:

ACCURACY AND PRECISION: The accuracy of arithmetic operations and the precision of numerical calculations were evaluated to ensure reliable results in both simulation and hardware implementation.

PERFORMANCE OPTIMIZATION: Optimization techniques were explored to improve code efficiency and reduce resource consumption, thereby enhancing the overall performance of the calculator application.

FUTURE ENHANCEMENTS: Potential areas for future enhancements and development were identified based on insights gathered from the software simulation phase. These include advanced mathematical functions, memory functionality, and graphical user interface improvements, among others.

1. CONCLUSION, COST AND FUTURE WORK

This device should be able to perform basic arithmetic operations such as addition, subtraction, multiplication, and division. It should also provide a user-friendly interface for input and display, allowing users to easily enter numbers and select operations.

By addressing this problem, the Arduino calculator project aims to provide a solution that offers flexibility, customization, and programmability in a calculator-like device. My project seeks to leverage the capabilities of the Arduino platform to create a calculator that can be adapted to diverse needs and preferences.

COST EXPENSES FOR MATERIALS

Arduino Nano/Uno Rs: 950

4\*4 Calculator Keypad Rs: 250

10K Potentiometer Rs: 450

Breadboard Rs: 340

LCD Display Rs: 580

Jumper Wire Rs: 230

Arduino UNO Cable Rs: 250

FUTURE WORK

While the current version of the Arduino calculator project focuses on implementing basic arithmetic operations, there are several avenues for future enhancement and development. Some potential areas for future work include:

ADVANCED MATHEMATICAL FUNCTIONS: Expand the calculator's capabilities to include more advanced mathematical functions such as trigonometric functions (sine, cosine, tangent), logarithms, exponentials, and square roots.

MEMORY FUNCTIONALITY: Implement memory functions such as memory recall, memory store, and memory clear, allowing users to store and reuse values during calculations.

SCIENTIFIC MODE: Introduce a scientific mode that enables users to access a broader range of mathematical functions and constants commonly used in scientific calculations.

CUSTOMIZABLE THEMES AND SETTINGS: Allow users to customize the appearance and settings of the calculator, including themes, font sizes, and button layouts, to suit their preferences.

UNIT CONVERSION: Incorporate unit conversion functionalities for converting between different units of measurement (e.g., length, mass, temperature) to make the calculator more versatile.

By pursuing these avenues for future work, the Arduino calculator project can evolve into a more versatile, feature-rich, and user-friendly tool for mathematical calculations and experimentation.