BASIC CALCULATOR USING ARDUINO

BY:

ARQAM H WAHEEDI

SIMPLE CALCULATOR

A basic calculator was created using Arduino and a 4x4 keypad. The keypad's rows and columns were connected to Arduino pins. Code was written to initialize the keypad and execute arithmetic operations based on input. An LCD display was optionally added for output. Numbers and operators were input by users via the keypad, and operations were executed by Arduino. The result was displayed on the LCD or serial monitor. Enhancements, such as memory functions, and error handling, could be added. A beginner friendly project, provided a hands-on experience in Arduino programming, hardware interfacing, and arithmetic concepts. Challenges included debouncing keypad input and optimizing code. Educational benefits encompassed practical skills in microcontroller systems and electronics.

Table of Contents

INTRODUCTION

A simple calculator using an Arduino microcontroller and a keypad. This project aims to demonstrate basic arithmetic operations such as addition, subtraction, multiplication, and division in a user-friendly manner.

Calculators, ubiquitous tools in modern life, have revolutionized the way we perform mathematical calculations. From basic arithmetic to complex scientific computations, calculators simplify mathematical tasks, enhancing efficiency and accuracy. Originally mechanical devices, calculators evolved into electronic and digital forms, becoming compact, portable, and versatile. They find applications in various fields, including education, finance, engineering, and science. As technology advances, calculators incorporate advanced features like graphing, programmability, and connectivity. With their user-friendly interfaces and robust functionalities, calculators remain indispensable aids, empowering users to tackle mathematical challenges with ease and precision in everyday life and professional endeavors.

Arduino, while not as powerful as a full-fledged computer, behaves as a small computing device capable of executing programmed instructions. Here's how it takes input to function as a simple calculator:

1. **Input Interface**: Arduino can interface with various input devices like keypads, buttons, sensors, etc. In the case of a simple calculator, a keypad can be used to input numbers and operators.

2. **Data Processing**: Once the input is received, Arduino processes it according to the programmed logic. For a calculator, Arduino interprets the input to identify numeric values, operators, and commands.

3. **Arithmetic Operations**: Arduino performs arithmetic operations based on the input received. It can handle addition, subtraction, multiplication, and division operations using built-in mathematical functions or custom logic.

4. **Output Display**: Arduino can output the result of the calculations to various devices such as an LCD display, LED matrix, or serial monitor. This allows users to view the calculated results.

5. **Feedback and Interaction**: Arduino can provide feedback to users during input and calculation processes, such as displaying the entered numbers and operators on an LCD screen or providing audible feedback through a buzzer.

By leveraging its input/output capabilities and processing power, Arduino can effectively emulate the functionality of a simple calculator, making it a versatile platform for educational projects and practical applications.

Components Used

1. Arduino UNO
2. Bread Board
3. Jumper Wires(Male to Male(18))
4. LCD Display(16*2)
5. Keypad (4*4)

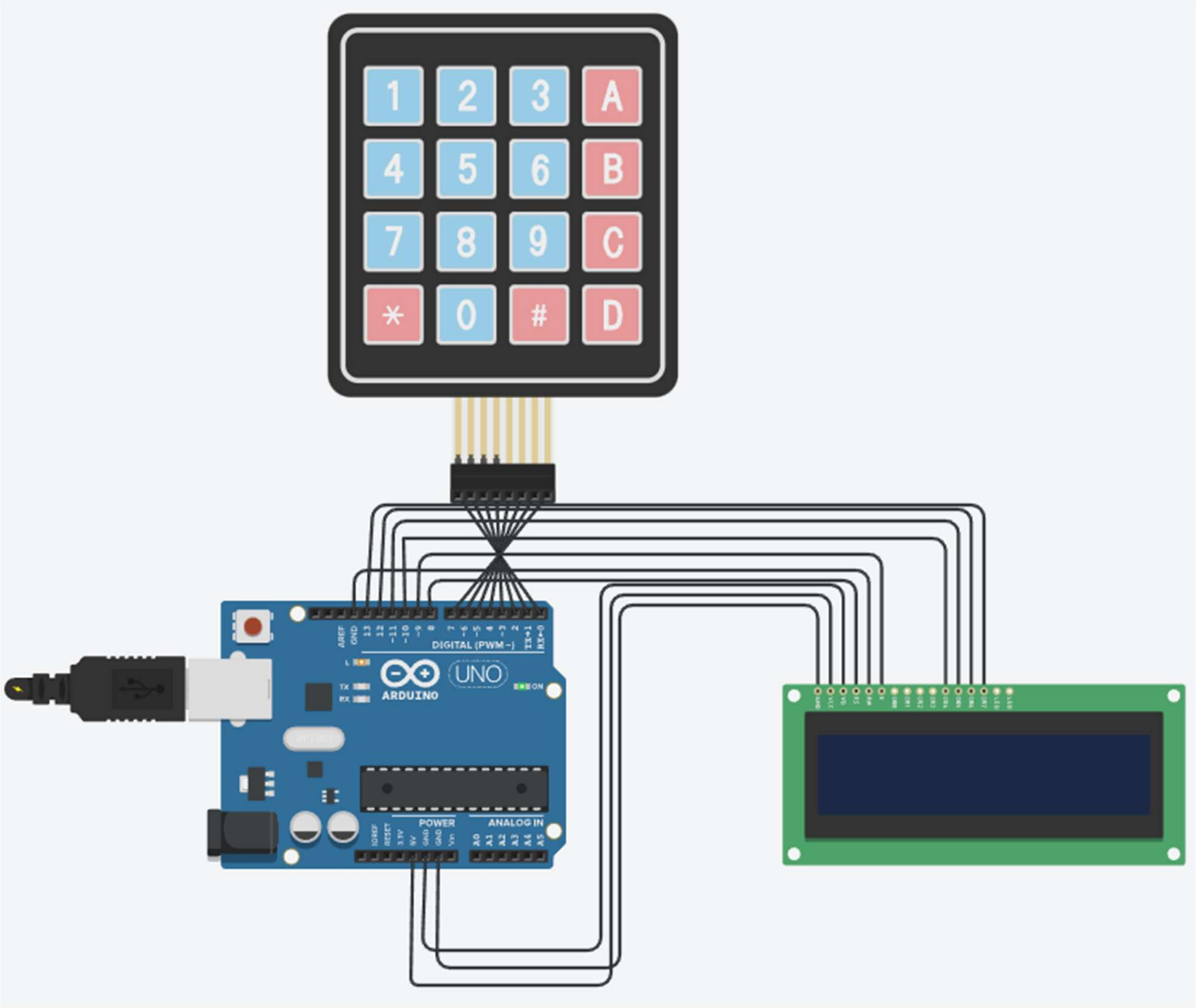


Fig 1.1 : Circuit Diagram(in TinkerCAD Simulation)

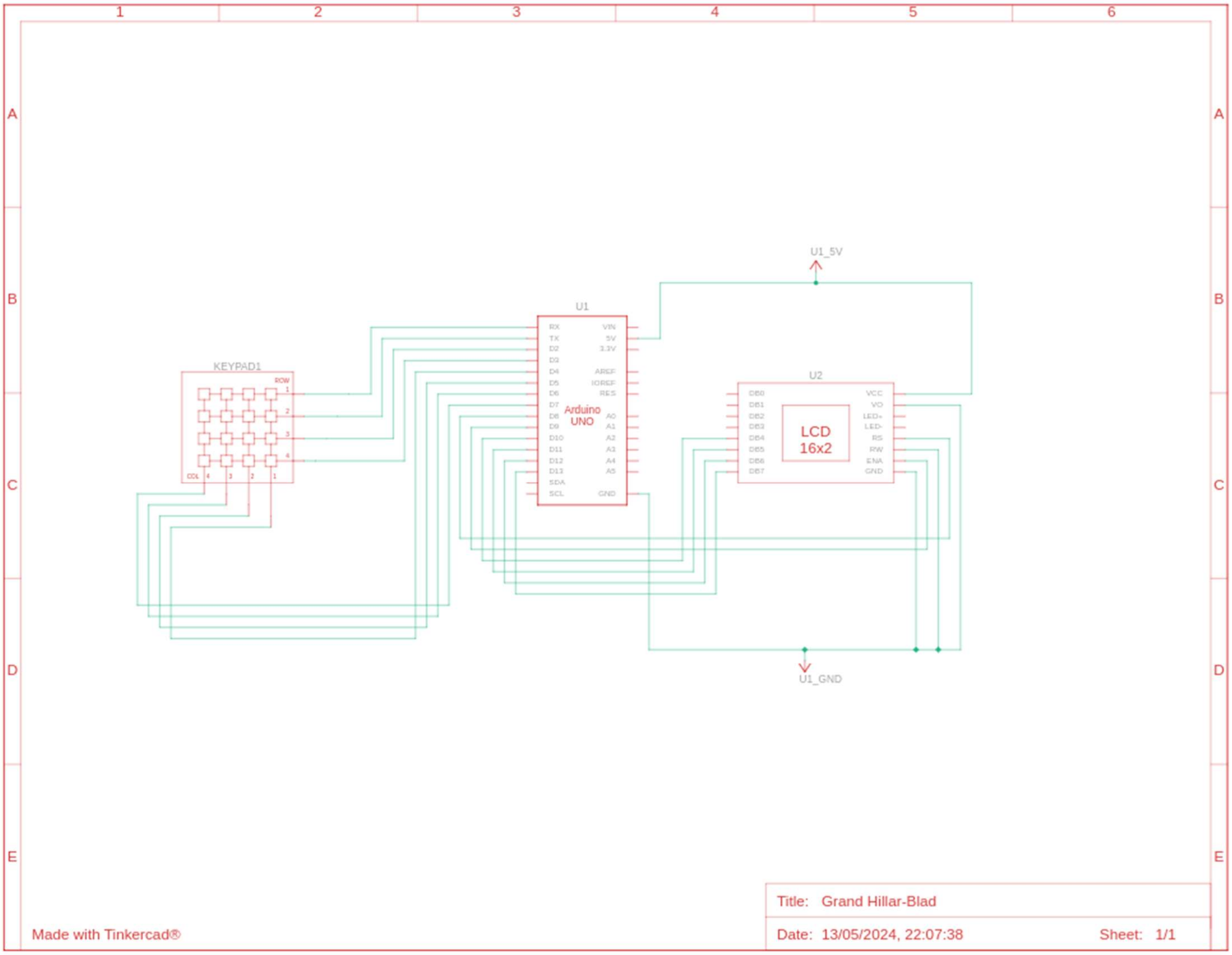


Fig 1.2 : Circuit Diagram With Connections

DESIGN

Working Principle:

- The 4x4 matrix keypad consists of 16 keys arranged in rows and columns. Each key press generates a unique combination of row and column signals, which can be decoded to determine the pressed key.

- The Arduino reads these signals from the keypad and interprets them as numeric values or operators (+, -, *, /).

- Based on the user input, the Arduino performs the corresponding arithmetic operation and displays the result on an LCD display, if available, or via serial communication to a computer.

Implementation Steps:

- Hardware Setup:
  - All the rows and columns of the keypad were connected to the digital pins of the Arduino using jumper wires
  - One can connect an LCD Display to the Arduino to show the input and output.
- Software Implementation:
  - Arduino code in c++ was written to initialize the keypad and LCD Display.Functions were implemented to read keypad input and perform arithmetic operations.

    - The LCD Screen was used as a display output or input
- User Interaction:
    - Users can input numeric values and select arithmetic operations using the keypad.
    - The Arduino performs the requested operation and displays the result.

CODE FOR ARDUINO :

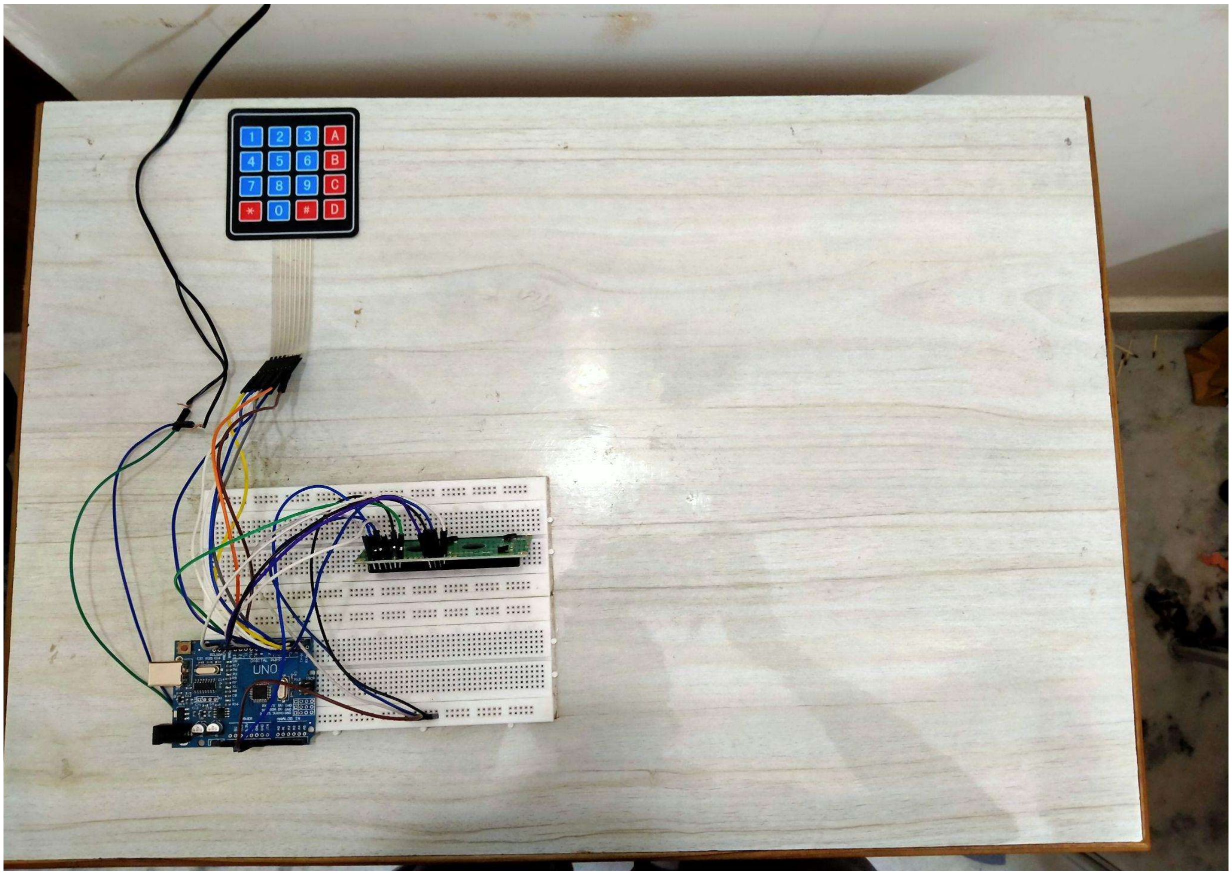
```
#include <Keypad.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// Define keypad pin connections
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] = {9, 8, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};

// Create keypad object
```

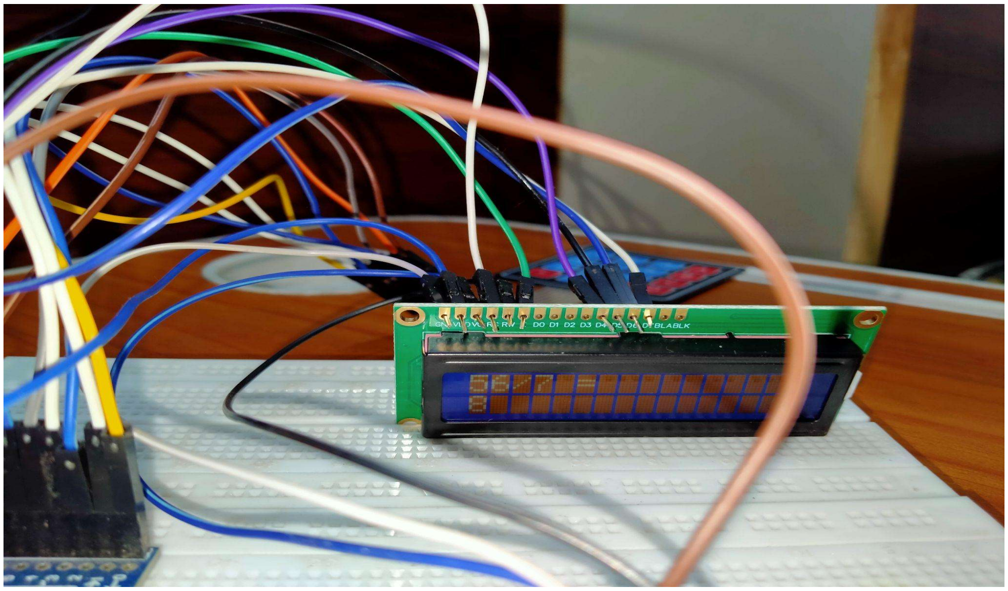
```
Keypad keypad = Keypad( makeKeymap(keys),
rowPins, colPins, ROWS, COLS );

// Define LCD display settings
LiquidCrystal_I2C lcd(0x27, 16, 2);

// Calculator variables
String input = ""; // String to store user
input
float num1 = 0; // First operand
float num2 = 0; // Second operand
char operatorSymbol = ' '; // Operator symbol
bool resultShown = false; // Flag to indicate
if result is shown

void setup() {
  // Initialize serial communication
  Serial.begin(9600);

  // Initialize LCD display
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("Calculator");
  delay(1000);
  lcd.clear();
}

void loop() {
  // Read keypad input
  char key = keypad.getKey();

  // If a key is pressed
  if (key != NO_KEY) {
    // Check if the result is shown, if so,
clear the display
    if (resultShown) {
      lcd.clear();
      resultShown = false;
    }

    // Check if the key is a digit or a
decimal point
    if (isdigit(key)  key == '.') {
      // Append the key to the input string
      input += key;
      lcd.print(key);
    }
    // Check if the key is an operator (+, -,
*, /)
```

```
    else if (key == '+'  key == '-'  key ==
'*'  key == '/') {
      // Store the input string as the first
operand
      num1 = input.toFloat();
      // Store the operator symbol
      operatorSymbol = key;
      // Reset the input string
      input = "";
      // Print the operator symbol on the LCD
display
      lcd.setCursor(15, 0);
      lcd.print(operatorSymbol);
    }
    // Check if the key is the equals sign
(=)
    else if (key == '=') {
      // Store the input string as the second
operand
      num2 = input.toFloat();
      // Reset the input string
      input = "";
      // Perform the calculation based on the
operator symbol
      float result;
      switch (operatorSymbol) {
        case '+':
          result = num1 + num2;
          break;
        case '-':
          result = num1 - num2;
          break;
        case '*':
          result = num1 * num2;
          break;
        case '/':
          result = num1 / num2;
          break;
        default:
          // Display error message if an
invalid operator is used
          lcd.setCursor(0, 1);
          lcd.print("Error");
          delay(1000);
          lcd.clear();
          break;
      }
      // Print the result on the LCD display
            lcd.setCursor(0, 1);
      lcd.print("Result: ");
      lcd.print(result);
      resultShown = true;
    }
    // Check if the key is the 'C' key for
clear
    else if (key == 'C') {
      // Clear the input string and reset the
calculator
      input = "";
      num1 = 0;
      num2 = 0;
      operatorSymbol = ' ';
      resultShown = false;
      lcd.clear();
    }
  }
}
```

RESULT

A Simple Calculator was Made using an Arduino , BreadBoard , Keypad and LCD Display





CONCLUSION

In conclusion, the development of a simple calculator using Arduino and a keypad underscores the versatility and accessibility of microcontroller-based systems in practical applications. Through this project, we have explored the fundamental principles of input interfacing, data processing, and output display, essential components of embedded systems design. By leveraging Arduino's capabilities and programming flexibility, we have successfully created a user-friendly calculator capable of performing basic arithmetic operations. This project not only serves as an educational endeavor, providing hands-on experience in electronics and programming, but also demonstrates the potential for Arduino to serve as a platform for innovation and creativity in solving everyday challenges. As we reflect on this project, we are inspired to continue exploring the vast possibilities of microcontroller technology in shaping the future of computing and automation.

FUTURE ENHANCEMENTS:

1. **Advanced Operations**: Expand the calculator's functionality to include advanced mathematical operations such as exponentiation, square root, trigonometric functions, and logarithms, catering to a wider range of mathematical needs.

2. **Decimal Input**: Implement support for decimal numbers, allowing users to perform calculations with greater precision and accuracy.

3. **Memory Functions**: Introduce memory functions such as storing and recalling values, enabling users to save intermediate results for later use in complex calculations.

4. **Error Handling**: Enhance error handling capabilities to detect and handle division by zero, overflow, and other common arithmetic errors gracefully, providing informative feedback to users.

5. **Graphical Interface**: Integrate a graphical user interface (GUI) using an LCD touchscreen or OLED display, providing a more intuitive and interactive user experience with graphical representations of calculations.

6. **Multiple Keypad Layouts**: Support multiple keypad layouts to accommodate different user preferences or regional variations, enhancing usability and accessibility.

7. **History Log**: Implement a history log feature to record and display a log of past calculations, allowing users to review and reference previous results.

8. **Custom Functions**: Allow users to define and execute custom mathematical functions, empowering them to tailor the calculator's capabilities to their specific needs and preferences.

9. **Wireless Connectivity**: Introduce wireless connectivity options such as Bluetooth or Wi-Fi, enabling remote control and data exchange with other devices or platforms, expanding the calculator's versatility and integration capabilities.

10. **Voice Control**: Integrate voice recognition technology to enable hands-free operation of the calculator, providing an alternative input method for users with limited dexterity or accessibility needs.

By incorporating these future enhancements, the calculator project can be elevated to new heights of functionality, usability, and versatility, catering to a broader range of user requirements and preferences.

REFERENCES:

https://forum.arduino.cc/t/calculator/360042

https://docs.arduino.cc/learn/starting-guide/whats-arduino/

https://circuitdigest.com/microcontroller-projects/arduino-calculator-using-4x4-keypad

https://projecthub.arduino.cc/123samridhgarg/arduino-calculator-bce0df

https://www.electronicwings.com/arduino/4x4-keypad-interfacing-with-arduino-uno

https://sparks.gogo.co.nz/ch340.html

https://github.com/Chris--A/Keypad