# Arduino-based Calculator using 4x4 Keypad

**Mini project**

Name : P.Sridevi

192110522

Name : MKS.Thevaashree

192110557

Course Code : ECA1487

Course Name : Embedded System For IOT Application

# Aim:

To create a basic and efficient calculator utilizing an Arduino Uno, a 4×4 keypad, and an LCD I2C display, capable of performing fundamental arithmetic operations such as addition, subtraction, multiplication, and division.

# Apparatus Required:

1. **Arduino Uno:** Microcontroller board for controlling the project.
2. **4×4 Keypad:** For user input of numbers and operations.
3. **LCD I2C Display:** To display inputs and results.
4. **USB Cable:** To connect Arduino and Software.
5. **Jumper Wires:** For connecting the components.

# Principle:

The project operates based on the following concepts:

1. **Keypad Input**:
   - The 4×4 keypad is used to input numbers and operations.
   - Each key press is read as a specific input (e.g., digits 0–9, operators +, −, ×, ÷, and equals).
2. **Arduino Processing**:
   - The Arduino processes the input, interprets the sequence of numbers and operations, performs the calculations, and sends the result to the display.
3. **LCD I2C Display**:
   - The LCD I2C display shows the user inputs and outputs (results).
   - It operates with fewer pins due to the I2C communication protocol, making it easy to integrate with the Arduino.

# Procedure:

1. **Hardware Setup**:
    1. Connect the 4×4 keypad to the Arduino digital pins.
        - Keypad pins → Arduino digital pins 2–9.
    2. Connect the LCD I2C display to the Arduino.
        - SDA pin → Arduino A4 (SDA).
        - SCL pin → Arduino A5 (SCL).
        - VCC and GND to Arduino 5V and GND.

3. Connect the Arduino UNO using USB Cable.

2. **Software Setup**:

   1. Write the Arduino code using the following libraries:

      - `Keypad.h` for interfacing with the keypad.
      - `LiquidCrystal_I2C.h` for interfacing with the LCD.

   2. Upload the code to the Arduino using the Arduino IDE.

3. **Testing**:

   1. Press the keys on the keypad to input numbers and operators.
   2. Verify that the LCD displays the correct inputs and results.
   3. Test various calculations to ensure proper functionality.

# Working:

1. When powered on, the system initializes and displays a welcome message on the LCD.
2. The user enters numbers and operations using the keypad.
   - Example: "12 + 7 =".
3. The Arduino interprets the inputs, performs the specified operation, and calculates the result.
4. The result is displayed on the LCD.

# Library Used:

- **Keypad.h**:

  Used to interface the 4×4 keypad with the Arduino for capturing user input.

- **LiquidCrystal_I2C.h:**

  Used to interface the LCD I2C display with the Arduino for displaying inputs and results.

# Applications:

1. Demonstrates basic calculator functionality using embedded systems.

2. Provides an understanding of interfacing input devices (keypad) and output devices (LCD) with a microcontroller.

# Advantages:

1. Cost-effective and easy to assemble.
2. Simplifies learning about microcontroller-based projects.
3. Expandable to include additional features such as memory functions or trigonometric operations.

# Innovation:

This project introduces an additional feature that transforms the calculator into a memory game tool for children. For instance, pressing the first button twice displays its default value (0-9) as shown on the keypad. However, pressing the second or third button twice shuffles the button's values, creating a memory challenge where children must identify the shuffled numbers. This enhances cognitive development while making learning engaging.

# Future Scope:

In an era where children are increasingly reliant on mobile phones, this tool provides a dual-purpose solution: a functional calculator for learning basic arithmetic and an interactive memory game to boost memory power and cognitive skills.

It serves as an educational alternative, promoting both mental development and practical learning.

Adding advanced features such as square root, percentage, or memory storage.

Upgrading to a touchscreen input for enhanced user interaction.

Integrating wireless communication to send calculations to a mobile app or PC.

# Code:

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 2); // LCD address at 0x27 for a 16x2 display


#include <Keypad.h>
```

```
const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1', '2', '3', '+'},
  {'4', '9', '6', '-'},
  {'7', '8', '5', '*'},
  {'C', '0', '=', '/'}
};
byte rowPins[ROWS] = {10, 9, 7, 6};
byte colPins[COLS] = {5, 4, 3, 2};

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

boolean presentValue = false;
boolean next = false;
boolean final = false;
String num1, num2;
int answer = 0;
char op;

// Variables for handling double press
unsigned long lastPressTime = 0;
char lastKey = '\0';
int mode = 0; // Current mode (default = 0)

// Function to set up key mappings for each mode
void setKeyMappings(int mode) {
  char tempKeys[ROWS][COLS]; // Temporary array to hold the key mappings
```

```
switch (mode) {
  case 1: {
    char temp[ROWS][COLS] = {
      {'1', '2', '3', '+'},
      {'4', '5', '6', '-'},
      {'7', '8', '9', '*'},
      {'C', '0', '=', '/'}
    };
    memcpy(tempKeys, temp, sizeof(temp));
    break;
  }
  case 2: {
    char temp[ROWS][COLS] = {
      {'9', '7', '4', '+'},
      {'8', '1', '2', '-'},
      {'5', '3', '6', '*'},
      {'C', '0', '=', '/'}
    };
    memcpy(tempKeys, temp, sizeof(temp));
    break;
  }
  case 3: {
    char temp[ROWS][COLS] = {
      {'3', '6', '9', '+'},
      {'2', '8', '5', '-'},
      {'1', '4', '7', '*'},
      {'C', '0', '=', '/'}
    };
    memcpy(tempKeys, temp, sizeof(temp));
    break;
```

```
    }
    default: {
      char temp[ROWS][COLS] = {
        {'1', '2', '3', '+'},
        {'4', '9', '6', '-'},
        {'7', '8', '5', '*'},
        {'C', '0', '=', '/'}
      };
      memcpy(tempKeys, temp, sizeof(temp));
      break;
    }
  }

  // Copy from tempKeys to the global keys array
  memcpy(keys, tempKeys, sizeof(keys));
}


void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(2, 0);
  lcd.print("ARDUINO UNO");
  lcd.setCursor(3, 1);
  lcd.print("Calculator");
  delay(3000);
  lcd.clear();
  setKeyMappings(0); // Set default key mapping
}
```

```
void loop() {
  char key = myKeypad.getKey();

  if (key != NO_KEY) {
    unsigned long currentTime = millis();

    // Check for double press within 300 ms
    if (key == lastKey && (currentTime - lastPressTime <= 300)) {
      mode = key - '0'; // Set mode based on double-pressed key
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("Mode: ");
      lcd.print(mode);
      setKeyMappings(mode); // Update key mappings for the selected mode
      delay(1000);
      lcd.clear();
    }

    lastKey = key;
    lastPressTime = currentTime;

    // Process single key presses for calculator logic
    if (key >= '0' && key <= '9') {
      if (!presentValue) {
        num1 += key;
        lcd.setCursor(0, 0);
        lcd.print(num1);
      } else {
        num2 += key;
        lcd.setCursor(num1.length() + 1, 0);
```

```
      lcd.print(num2);
      final = true;
    }
  } else if (!presentValue && (key == '+' || key == '-' || key == '*' || key == '/')) {
    presentValue = true;
    op = key;
    lcd.setCursor(num1.length(), 0);
    lcd.print(op);
  } else if (final && 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.print("Answer: ");
    lcd.print(answer);
    delay(2000);
    lcd.clear();
    num1 = num2 = "";
    presentValue = final = false;
    answer = 0;
  } else if (key == 'C') {
    lcd.clear();
    num1 = num2 = "";
    presentValue = final = false;
    answer = 0;
  }
 }
}
```

# Output: