**Lab Exercise 15- Keypad with LCD Arduino**

Simple **Lab Exercise** using a **4x4 Keypad** and a **16x2 LCD** with an **Arduino**. In this exercise, you’ll practice reading input from a keypad and displaying the input on an LCD. The user will input numbers via the keypad, and the numbers will be shown on the LCD in real-time. This exercise introduces basic interfacing with the keypad and LCD, which is useful for building interactive systems like calculators, PIN-based security systems, or menu-driven applications.

**Components:**

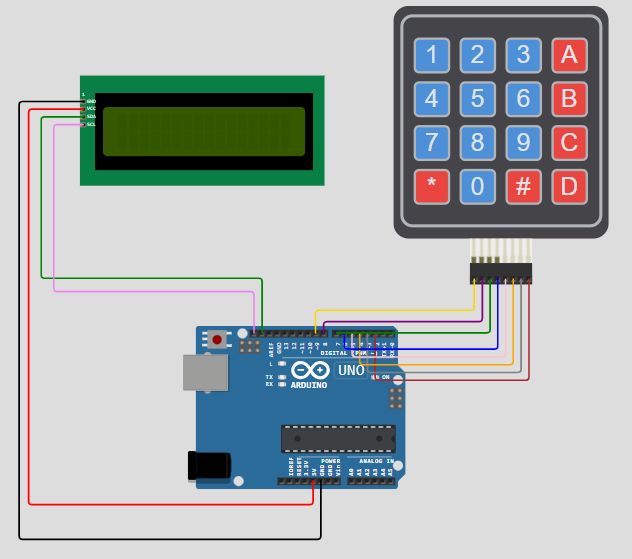
* Arduino Uno
* 4x4 Keypad (Matrix Keypad)
* 16x2 LCD (with I2C adapter for easier wiring)
* Jumper wires
* Breadboard
* Power supply (USB or battery)

**Learning Objectives:**

1. Understand how to interface a **4x4 keypad** with Arduino.
2. Learn how to display characters and numbers on a **16x2 LCD** using I2C.
3. Build a basic interactive system where keypad inputs are displayed on the LCD.

**Circuit Connections:**

1. **4x4 Keypad**:
   * The 4x4 keypad has 8 pins, which will be connected to Arduino digital pins.
   * Connect pins of the keypad to digital pins 2, 3, 4, 5, 6, 7, 8, and 9 on the Arduino.
   * Check your keypad’s pinout (rows and columns) to match the correct connections.
2. **16x2 LCD with I2C Adapter**:
   * VCC → 5V
   * GND → GND
   * SDA → A4
   * SCL → A5



**Code for Arduino:**

**Single Number Input:**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <Keypad.h>

// Initialize LCD (with I2C address 0x27) and 16x2 size

LiquidCrystal\_I2C lcd(0x27, 16, 2);

// Keypad setup

const byte ROWS = 4; // Four rows

const byte COLS = 4; // Four columns

char keys[ROWS][COLS] = {

{'1', '2', '3', 'A'},

{'4', '5', '6', 'B'},

{'7', '8', '9', 'C'},

{'\*', '0', '#', 'D'}

};

// Connect keypad to Arduino digital pins

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);

void setup() {

// Initialize LCD

lcd.init();

lcd.backlight(); // Turn on the backlight

// Display instructions on the LCD

lcd.setCursor(0, 0);

lcd.print("Enter Numbers:");

}

void loop() {

char key = keypad.getKey(); // Get the key pressed

if (key) {

lcd.setCursor(0, 1); // Move to the second line

lcd.print(key); // Display the key pressed on LCD

}

}

**How the Code Works:**

1. **Keypad Input**:
   * The code reads input from the 4x4 keypad using the getKey() function. This function returns the character of the pressed key (e.g., '1', '2', 'A', etc.).
   * The keypad has four rows and four columns, and each key is associated with a specific row-column pair.
2. **LCD Display**:
   * The **LCD** is initialized using the LiquidCrystal\_I2C library, which simplifies the communication with the LCD using the I2C protocol.
   * When the user presses a key on the keypad, it is displayed on the **second line** of the LCD.
   * The cursor is positioned on the second row (1st index) of the LCD, and each pressed key is printed in real-time.

**Steps for the Lab:**

1. **Set Up the Circuit**:
   * Connect the keypad and LCD to the Arduino according to the circuit diagram and the pinout of your keypad.
2. **Upload the Code**:
   * Copy the provided code into the Arduino IDE, select the correct board (Arduino Uno), and upload it to the Arduino.
3. **Test the Keypad and LCD**:
   * After uploading the code, press any key on the keypad.
   * The corresponding character will be displayed on the LCD’s second row.
   * Press several keys and observe how the LCD updates with each keypress.
4. **Experiment with Different Key Combinations**:
   * Try entering different sequences on the keypad (e.g., 1234, 5678) and observe the LCD display.
   * You can modify the code to display more characters or clear the LCD after certain conditions are met.

**Code 2: Multi-number input:**

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <Keypad.h>

LiquidCrystal\_I2C lcd(0x27, 16, 2);

const byte ROWS = 4; // four rows

const byte COLS = 4; // four columns

char keys[ROWS][COLS] = {

  {'1', '2', '3', 'A'},

  {'4', '5', '6', 'B'},

  {'7', '8', '9', 'C'},

  {'\*', '0', '#', 'D'}

};

byte rowPins[ROWS] = {9, 8, 7, 6}; // connect to the row pinouts of the keypad

byte colPins[COLS] = {5, 4, 3, 2}; // connect to the column pinouts of the keypad

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

const int maxInput = 32;

const int halfMaxInput = maxInput / 2;

char inputArray[maxInput]; // Array to store entered characters (changed to size maxInput)

int arrayIndex = 0; // Index to keep track of the current position in the array

char firstline[halfMaxInput];

char secondline[halfMaxInput];

void setup() {

  lcd.init();

}

void loop() {

  char key = keypad.getKey();

  if (key) {

    if (key == 'C') {

      // Clear the array

      clearArray();

      clearLCD();

    } else if (key == 'D') {

      // Delete the last character from the array

      deleteLastCharacter();

    } else {

      // Add the key to the array if it's not full

      if (arrayIndex < maxInput) {

        addCharacter(key);

      } else {

        displayErrorMessage("Input is full");

      }

    }

    splitArray(); // Split the input array into firstline and secondline

    printToLCD();

  }

}

void addCharacter(char key) {

  if (arrayIndex < maxInput) {

    inputArray[arrayIndex++] = key;

  }

}

void deleteLastCharacter() {

  if (arrayIndex > 0) {

    arrayIndex--;

    inputArray[arrayIndex] = '\0';

  }

}

void clearArray() {

  arrayIndex = 0;

  for (int i = 0; i < maxInput; i++) {

    inputArray[i] = '\0';

  }

}

void splitArray() {

  for (int i = 0; i < halfMaxInput; i++) {

    firstline[i] = inputArray[i];

    secondline[i] = inputArray[i + halfMaxInput];

  }

}

void printToLCD() {

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print(firstline);

  lcd.setCursor(0, 1);

  lcd.print(secondline);

}

void displayErrorMessage(const char \*message) {

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Error: ");

  lcd.setCursor(0, 1);

  lcd.print(message);

  delay(2000); // Display the error message for 2 seconds

  lcd.clear();

}

void clearLCD() {

  lcd.clear();

  lcd.setCursor(0, 0);

  lcd.print("Input");

  lcd.setCursor(0, 1);

  lcd.print("Cleared");

  delay(2000); // Display the error message for 2 seconds

  lcd.clear();

}