Experiment 7

Title: - Interfacing 3x3 Matrix Keypad to Arduino

Objectives:-

Interface 4x4 Matrix Keypad to Arudino. Write a program in Embedded C to display Key pressed by user on LCD.

Apparatus:-

1. Arduino UNO Board.
2. LCD, 4x4 keypad (SD543)
3. Connecting Wires
4. Arduino Software

Theory: -

Matrix Keypad Operation

When input keys are more than 4, matrix keyboard can save I/O. In following figure, 16 keys make up a 4 X4 matrix. Where 8 I/O are employed and one key can nestle on one intersection. Attention: the two lines that formed an intersection are electrically connected only when the key is pressed down. Among the 8 I/O lines, 4 (vertically arranged) are called scanning line (output) and 4 (horizontally arranged) are called feedback line (input). The signals of a matrix keyboard are read in time-sharing mode, once a line. Therefore, CPU can attain status of keyboard after reading four times.

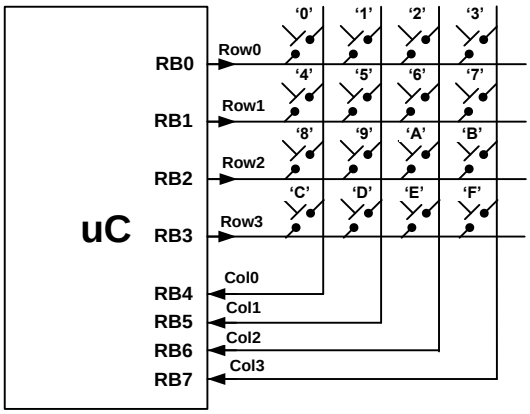

Figure 8.1: Matrix Keypad connection diagram

Key Detection

Software scanning sequence of the circuit shown in circuit diagram is as follows (Attention: RB3—RB0 are scanning lines and must be set to output mode, while RB7—RB4 are feedback lines and must be set to input mode.

1. Output 1110 to the scanning line RB3—RB0. Since RB0 output "0", when keys in the row0 that is connected to RB0 are pressed down, the status of the feedback line is "0".

However, when keys on the other three lines/rows are pressed down, status of the feedback line won't be changed.

2. Input status of the feedback line (RB7—RB4), and check if the four keys in the first row (the row controlled by RB0) are pressed down. The checking method is as follows. Checking the four bits of the feedback lines. If one of these feedback lines is "0", it indicates that the key connected to this line is pressed down.
3. If there is no key pressed down in the first row, and then output 1101 to RB3-RB0 to scan the line controlled by RB1 which is second row - Row1.
4. Input status of the feedback line again. The data attained is the status of the second row (the row controlled by RB1), and check if there is a key pressed down.
5. Output scanning signal of 1011 to scan the line controlled by RB2 which is third row – Row2
6. Input status of the feedback line. The data attained is the status of the third row (the row controlled by RB2), and check if there is a keystroke.
7. Output scanning signal of 0111 to scan the line controlled by RB3 which is fourth row – Row3.
8. Input status of the feedback line. The data attained is the status of the fourth row (the row controlled by RB3), and check if there is a keystroke.
9. Return to step 1 to start next scanning period.

The nine steps above are repeated ceaselessly until a keystroke of a certain row is detected (status of the feedback line is "0"), then this key will be coded by the program. There is no standard coding ways. The coding principle is to distinguish these keys by defining different codes for different keys.

As the keyboard shown in above figure 1.1 the keys are to be defined as the codes labeled in the figure ('0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F').

The relationship between codes of keys and the ranging positions can be expressed as follows:

Key coding = Row Number X 4 + Column number

For example, the key named 09 is arranged at Row 2 and Column 1, according to the formula:

Key code for key '9' = Row Number X 4+Column number = 2 X 4 + 1

Keyboard debounce:

When striking a keyboard key, the key oscillates against its contacts several times before settling. When released, it bounces again until it reverts to its rest state. Although it happens on such a small scale as to be invisible to the naked eye, it's sufficient for the computer to register multiple key strokes inadvertently.

To resolve this problem, the processor in a keyboard "debounces" the keystrokes, by aggregating them across time to produce one "confirmed" keystroke that (usually) corresponds to what is typically a solid contact. Early membrane keyboards limited typing speed because they had to do significant debouncing.

When any of the key is pressed i.e. zero on any of the key is detected, the controller waits for 20mS for the key debounce and then scans the columns again. The debounce is necessary because:

a. Waiting for key debounce help to detect if the key press was not erroneous viz due to spike.
b. 20ms to 100ms delay also prevents the same key press from being interpreted as a multiple key press.

Interfacing Diagram

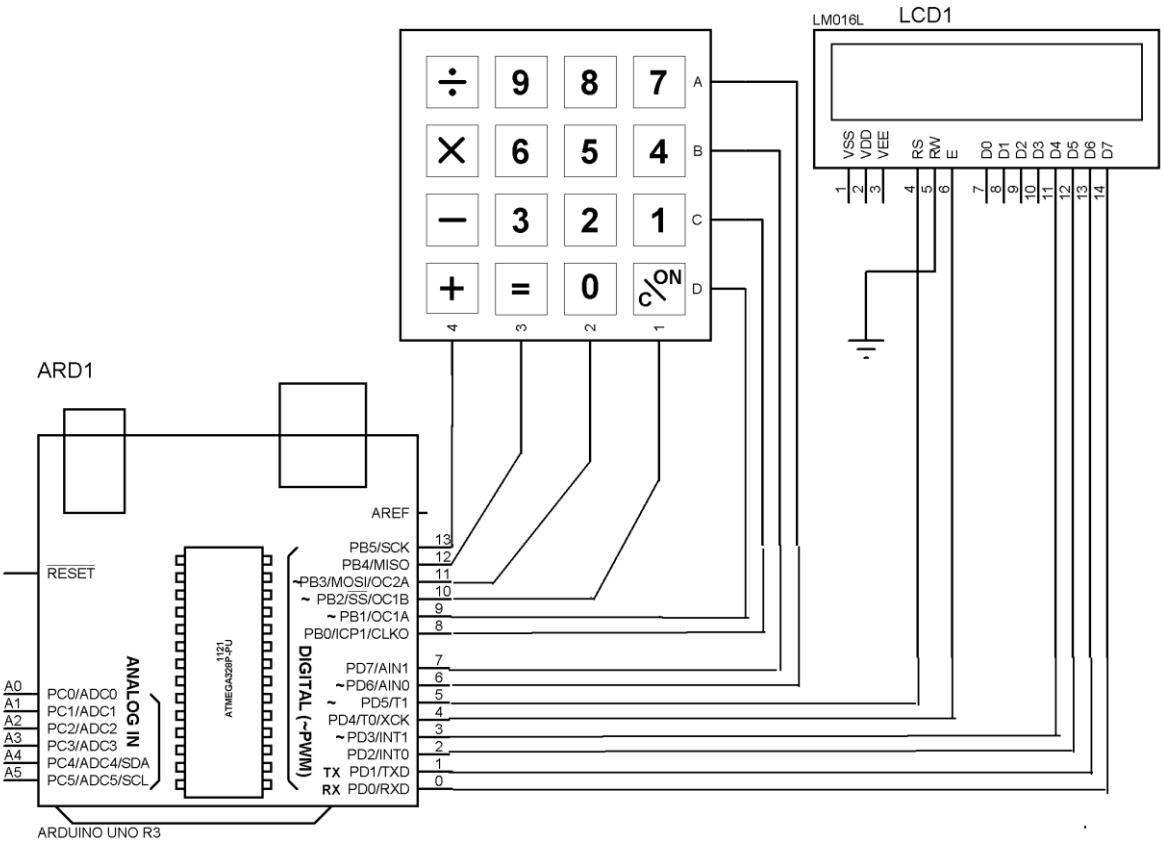

Figure 8.2 : Interfacing Matrix Keypad and 16x2 LCD to Arduino Microcontroller

Algorithm

1. Include LCD Library Functions (LiquidCrystal.h)
2. Include the Keypad library Functions (Keypad.h)
3. Configure the LCD Interfacing with Arduino Pins as given below

LCD Pins	Arduino Uno
Register Select (RS)	Digital 0
Enable (EN)	Digital 1
Data bit4 (D4)	Digital 2
Data bit5 (D5)	Digital 3
Data bit6 (D6)	Digital 4
Data bit7 (D7)	Digital 5

4. Define the number of rows as 3 and number of columns as 3
5. Define the keymap for each key pressed
6. Configure the Keypad Row pin and Column Pin Interfacing with Arduino Pins as given below

Key pad Pins	Arduino Uno
Row 0 (R0)	Digital 6
Row 1 (R1)	Digital 7
Row 2 (R2)	Digital 8
Row 3 (R3)	Digital 9
Column 0 (C0)	Digital 10
Column 1 (C1)	Digital 11
Column 2 (C2)	Digital 12
Column 3 (C3)	Digital 13

7. Initialize an instance of the Keypad class by passing keymap, row pins, column pins, number of rows and number of columns
8. In Setup function
    a. Set up the LCD's number of columns and rows.
    b. Print Message “Keypad Interface” on first line of LCD
    c. Set Cursor of LCD to 2nd line first column.
    d. Print Message “Key Pressed” on second line of LCD

9. In Loop function
    a. Declare the Char Variable to hold the key value
    b. Get the key pressed value using getKey() function
    c. Display the value of Key on LCD it valid key is pressed.

Result and conclusion: