GPIO

Input Device:

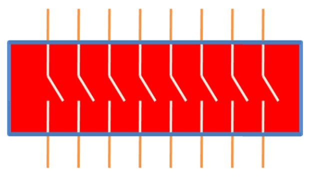
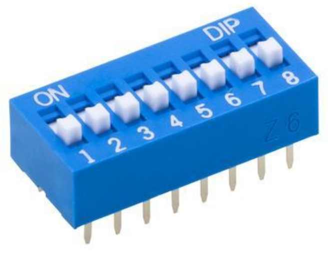
1. Dip Switch
2. Push-Button Switch
3. Matrix/Hex Keyboard
4. Analog Input (Sensors)

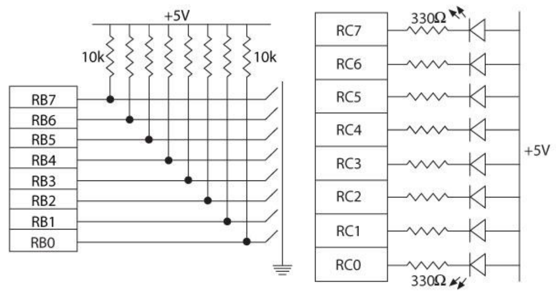
Output devices:

1. 7 Segment display
2. LED Display
3. LCD Display

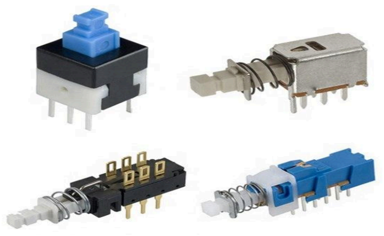
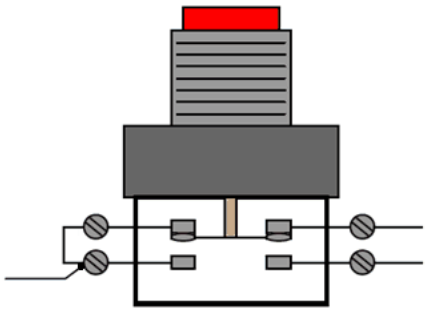
Input Devices

1. Dip switch:

One side of the switch is tied high (to a power supply through a resistor called a pull-up resistor), and the other side is grounded. The logic level changes when the position is switched.	 DIP Switch Schematic	

 +5V 10k 10k RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0 RC7 RC6 RC5 RC4 RC3 RC2 RC1 RC0 330Ω 330Ω +5V	Interfacing Dip Switches and Interfacing LEDs follow the same rules.

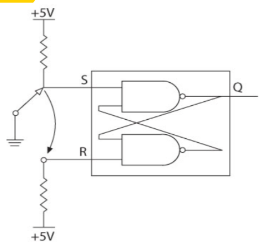
2. Push-Button switch

The connection is the same as in the DIP switch except that contact is momentary. When a key is pressed (or released), mechanical metal contact bounces momentarily and can be read.	
	

Note: Push-Button switches has a problem called Key-debouncing problem

Key debouncing problem: Once a key is pressed, after releasing that key we might get some distorted signals rather than complete 1 or 0. It can cause the reading of one contact as multiple inputs

Solution of key debouncing problem:

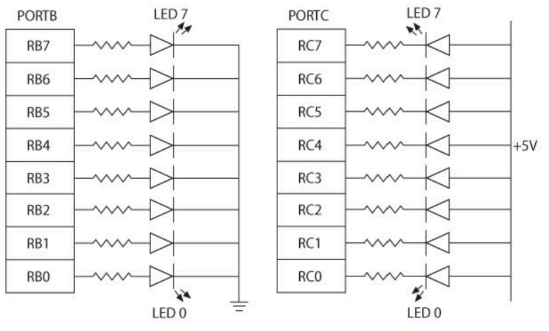
Software key debouncing solution: ● Wait for 20 ms. ● Read the port again.	**Hardware key debouncing solution:** ● Using a S-R latch along with the switch 

Output Devices

1. **7 Segment display:**

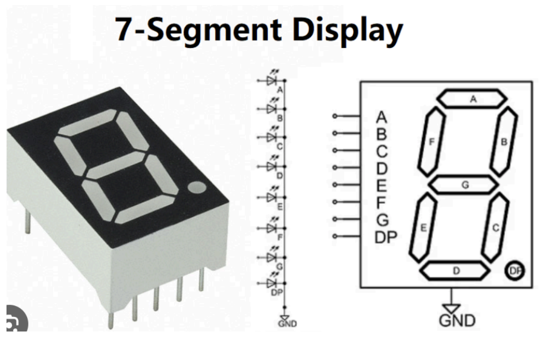
- First need to know LED interfacing

Two ways of connecting LEDs to I/O ports:
- Common Cathode - The current is supplied by the I/O port called current sourcing.
- Common Anode - The current is received by the chip called current sinking.

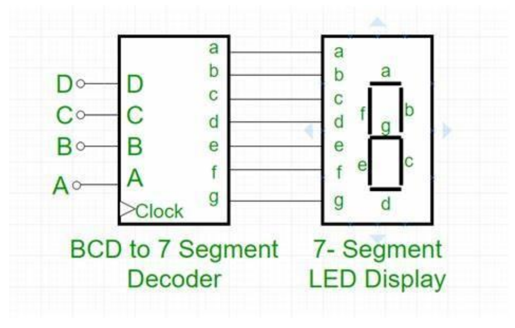


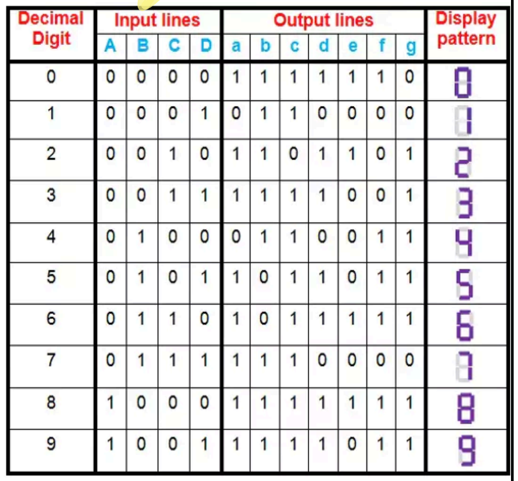
Common Cathode Common Anode

- 7 segment display:



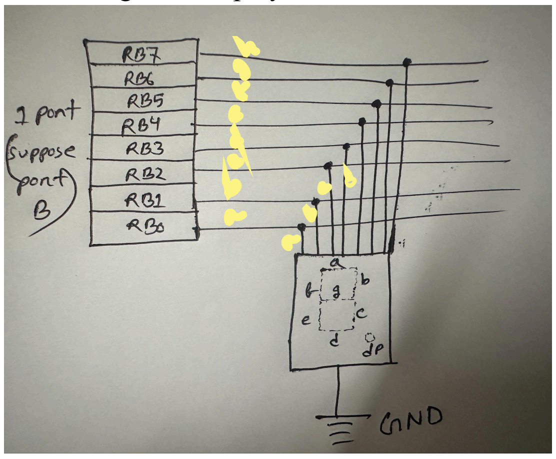
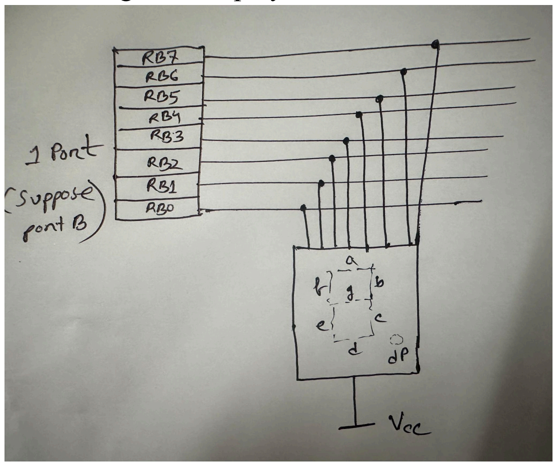
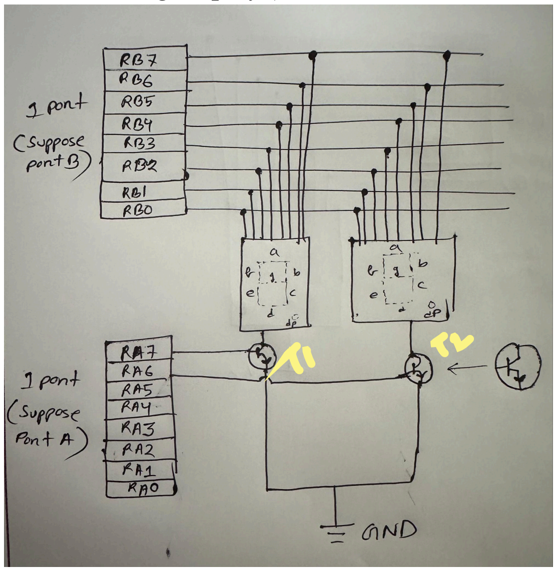
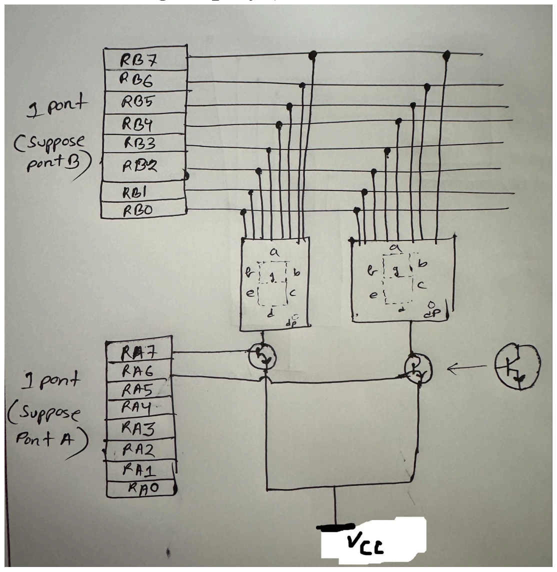
- BCD to 7-Segment Display table: [Common Cathode]



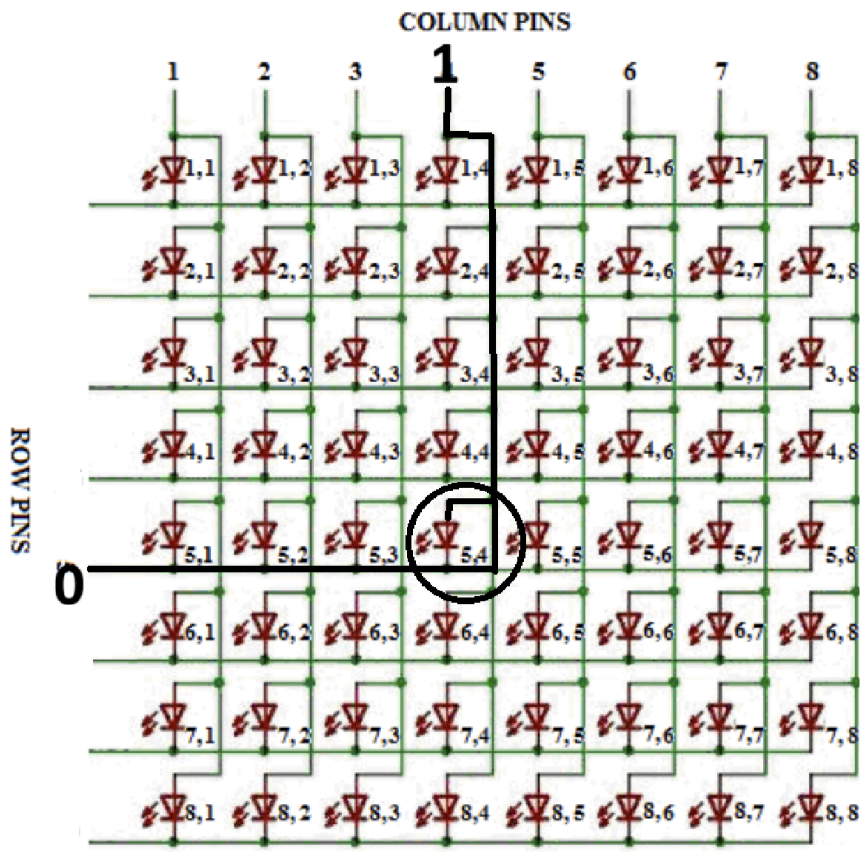


Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	1	0	1	1	9

- 2 ways to interface a 7-segment display: Common Anode & Common cathode

Common cathode mode	Common Anode mode
For 1 7-segment display  RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0 1 port (suppose port B) a b c d e f g dp GND	For 1 7-segment display  RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0 1 Port (Suppose port B) a b c d e f g dp Vcc
For multiple 7-segment display (Use transistor for controlling displays)  RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0 1 port (Suppose port B) RA7 RA6 RA5 RA4 RA3 RA2 RA1 RA0 1 port (Suppose Port A) T1 T2 GND	For multiple 7-segment display (Use transistor for controlling displays):  RB7 RB6 RB5 RB4 RB3 RB2 RB1 RB0 1 port (Suppose port B) RA7 RA6 RA5 RA4 RA3 RA2 RA1 RA0 1 port (Suppose Port A) Vcc

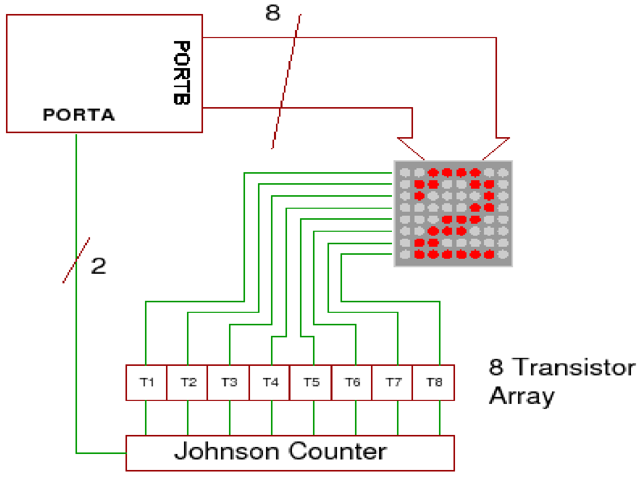
2. **LED Matrix Display:**
    - Has 16 pins [Row 8 & column 8]
    - Row pins serves as a sourcing/sinking point and actual data is sent through the column pins
    - If a row pin is grounded [0] and from a column data [1] is sent only that specific LED will lit up.
    - If a row pin is powered [1] and from a column data [0] is sent only that specific LED will lit up.



-

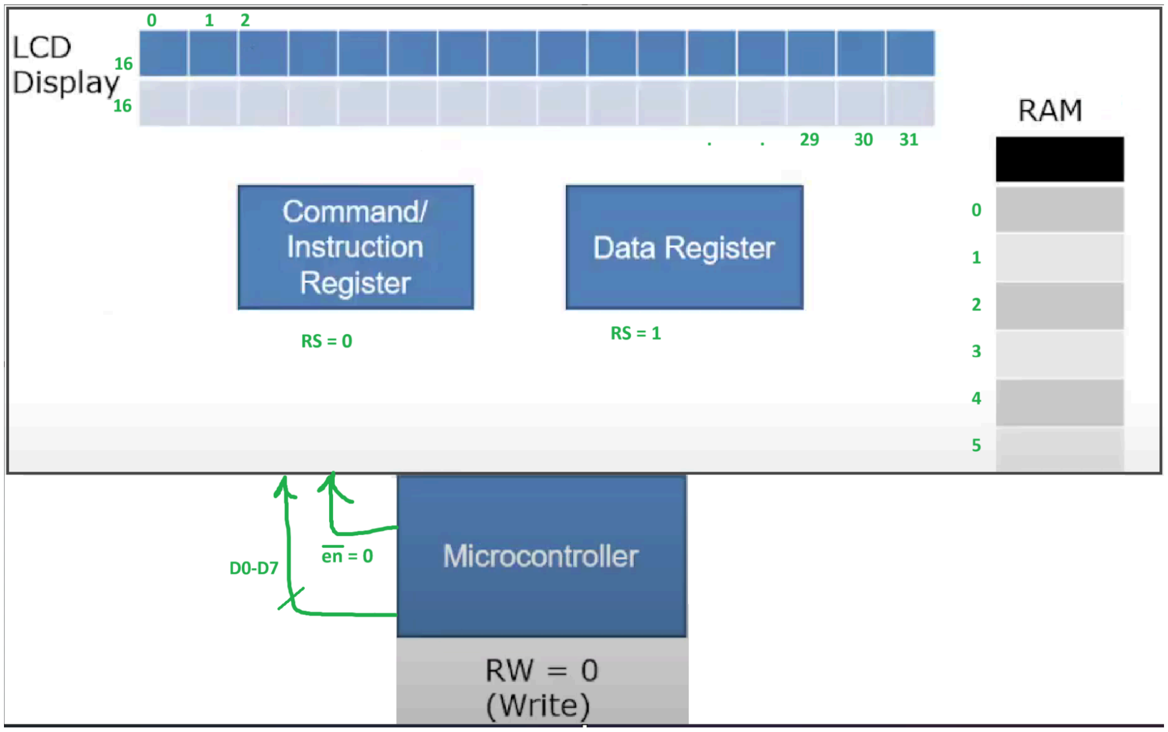
Note: 16 pins are too computationally heavy therefore we can use Johnson counters to reduce the pins needed.

- With the Johnson counter only 10 pins are needed.
- Columns will send data as usually [Through 8 pins]
- In the rows we will connect a Johnson counter [Which is basically an array-like structure of 8 transistors]. JC has 2 pins: 1 enable pin and 1 data pin to change clock pulse
- With each clock pulse the data sent [1/0] from the data pin will shift 1 bit and light up the LEDs according to the data sent through the columns.

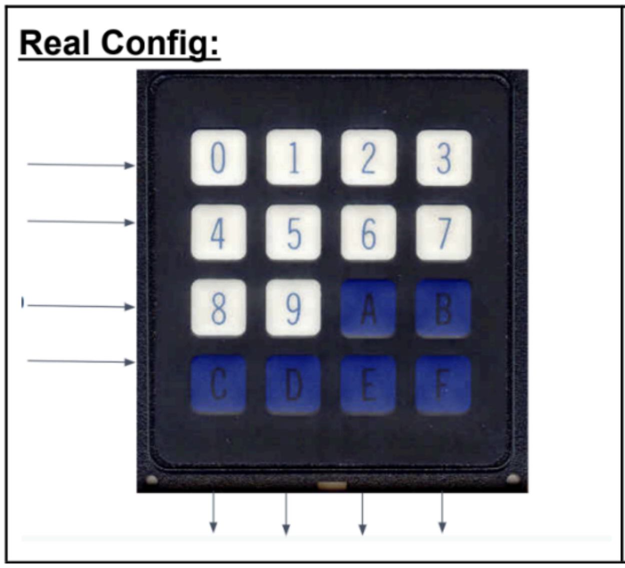


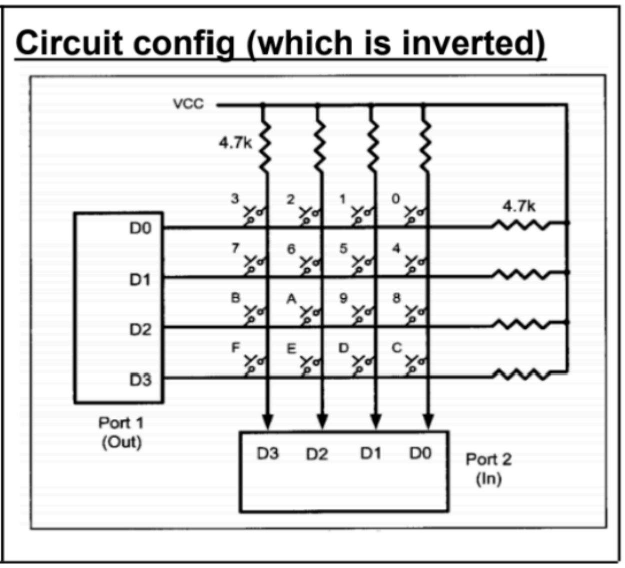
3. LCD display:

1. Microcontroller will activate the LCD first [en = 0]
2. Microcontroller will send 0 through D0-D7 which will work to activate the register select pin [RS = 0] to configure the LCD
3. Microcontroller will send 1 through D0-D7 which will work to activate the register select pin [RS = 1] to activate data register
4. Now microcontroller will send the data through D0-D7
5. The data will go to the data RAM through data register
6. Finally the data will be displayed on the LCD from data RAM



Matrix/ HEX Keyboard





How does it work?

1. **Column identification:**

a. Initially the column values remain 1 since connected to VCC. Meaning, D3-D2-D1-D0 = 1-1-1-1

b. Once any key is pressed the column value is changed to 0. E.g. for pressing "9" the value of D3-D2-D1-D0 = 1-1-0-1

c. The column is identified since the value has changed and it's not 1111 anymore

d. Now this value 1101 (D3-D2-D1-D0 = 1-1-0-1) gets saved in a register for later use (Key press identification)

2. **Row identification:**

a. Initially the row values remain 0 from the end of the microprocessor

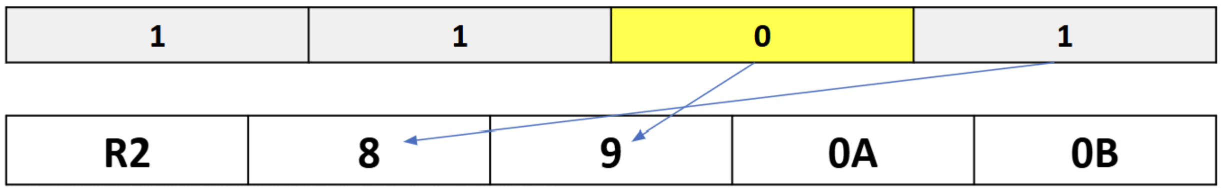
b. With each pulse microprocessor sends 0 from one pin (at first D0 = 0) and 1 from others (D3-D3-D1 = 1-1-1)

c. With each clock pulse and each change the new values are being compared with the saved value is the register

3. **Key press identification:**

a. We got the value of column in step 1 (1101 from Row 2 (R2))

b. Now we have to find the 0 by shift right (SHR) operation to find out the corresponding value and store that into the KEYPRESS variable.


1
1
0
1
R2
8
9
0A
0B