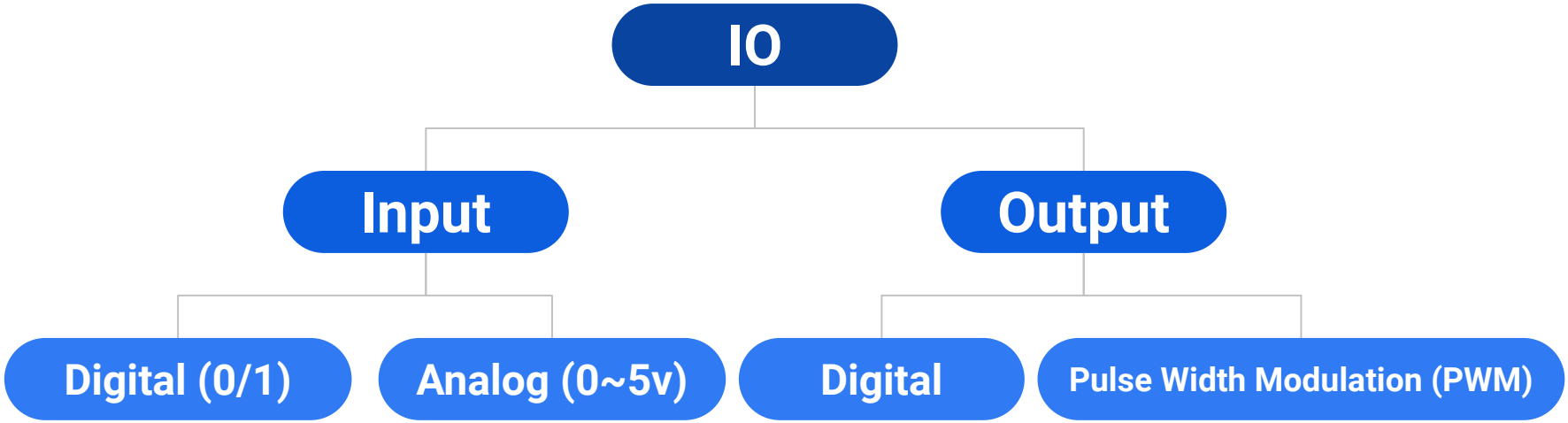
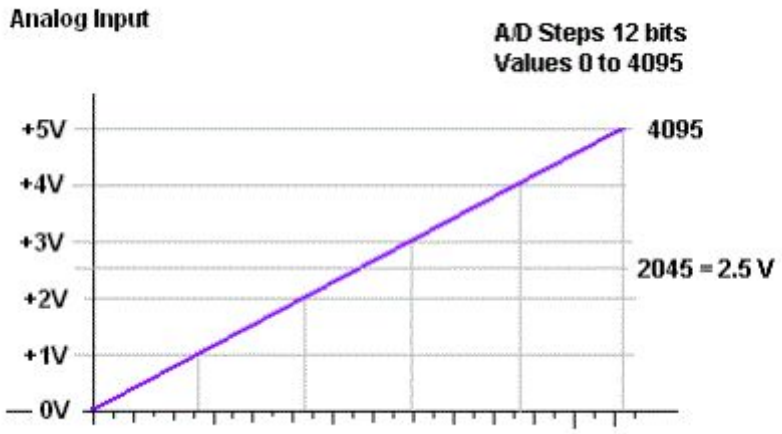
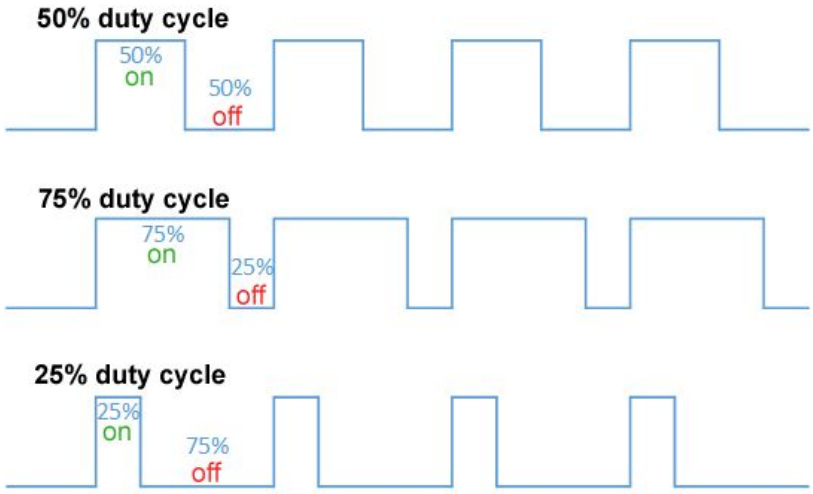
General Purpose IO (GPIO) (Low level IO)

Md. Khalilur Rhaman PhD
Associate Professor
CSE Department
BRAC University

Input Output



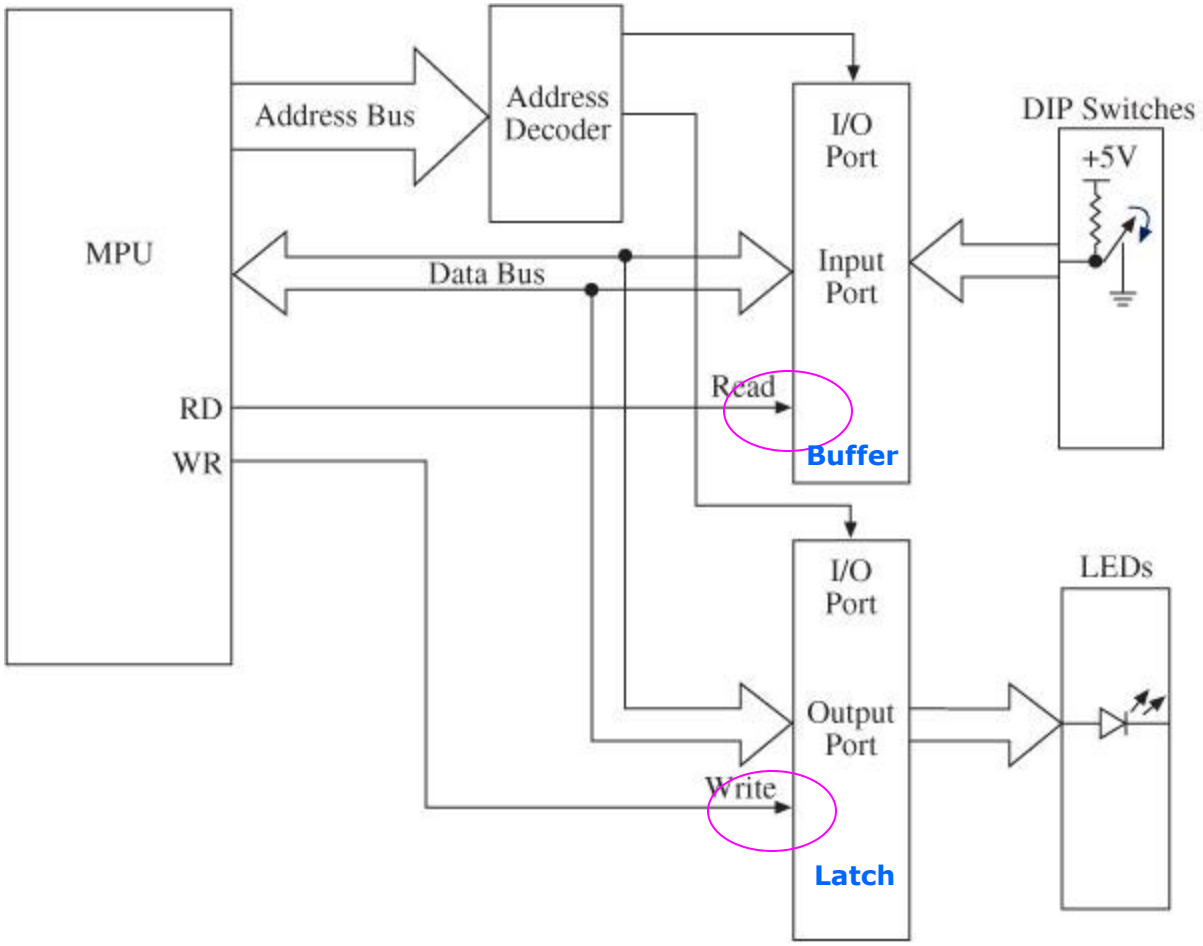




Basic Concepts in I/O Interfacing

- I/O devices (or peripherals) such as LEDs, LCDs, keyboards etc. are essential components of the microprocessor-based or microcontroller-based systems. Those can be classified into two groups
  - input devices
  - output devices

Block Diagram of I/O Interfacing



Buffer keep the level of 0 or 1 (to resolve fan in problem)
Latch hold the value (0/1) until it is cleared

I/O Ports: Interfacing and Addressing

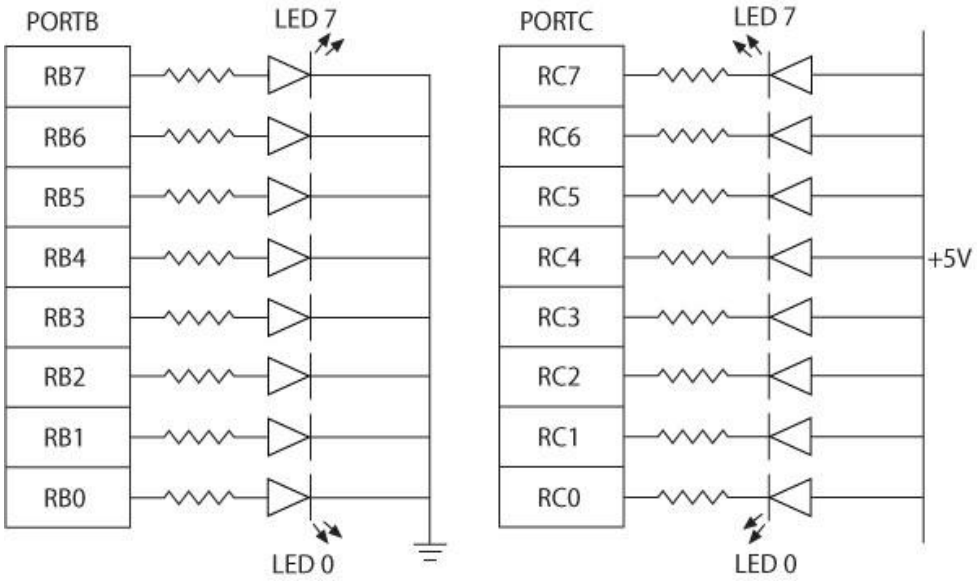
- To read (receive) binary data from an input peripheral
  - MPU places the address of an input port on the address bus, enables the input port by asserting the RD signal, and reads data using the data bus.
- To write (send) binary data to an output peripheral
  - MPU places the address of an output port on the address bus, places data on data bus, and asserts the WR signal to enable the output port.

Output

- Light Emitting Diode (LED)
- Seven segment display (Alphanumeric)
- Matrix Display
- Liquid Crystal Display (LCD)
- Buzzer
- DC Motor
- Servo Motor
- Stepper Motor

Interfacing LED

- 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.



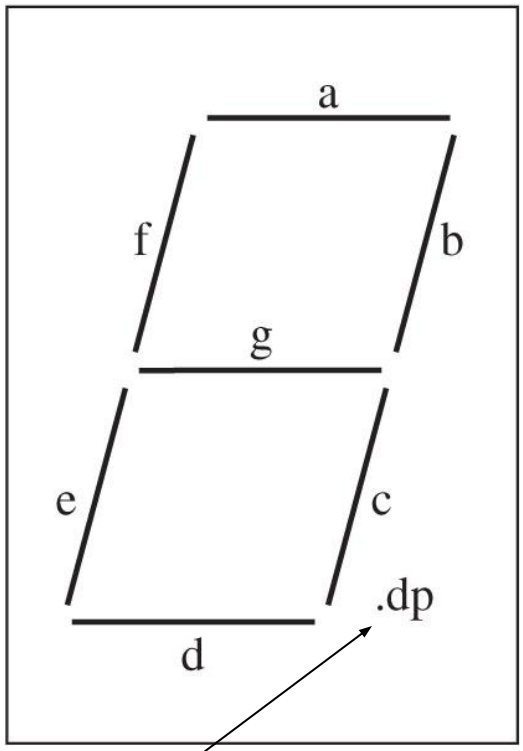
Common Cathode

Common Anode

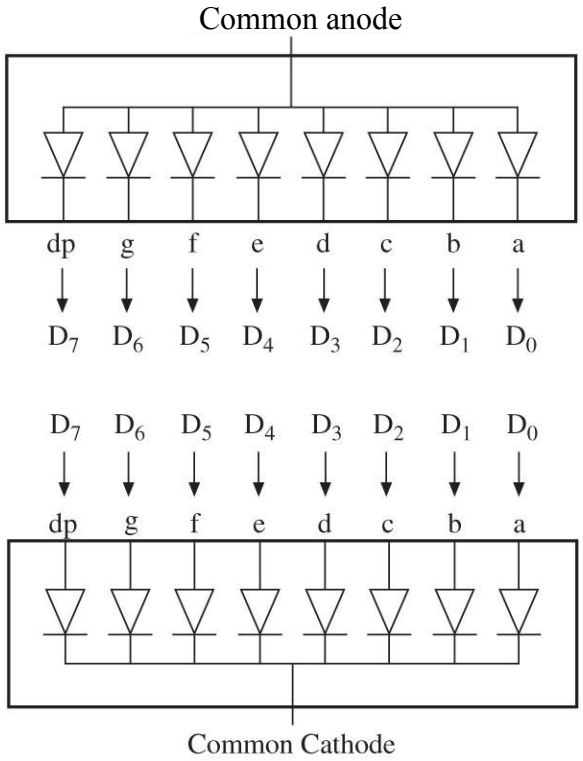
Interfacing Seven-Segment

- Seven-segment LEDs
  - Often used to display BCD numbers (0 through 9) and a few alphabets
  - A group of eight LEDs physically mounted in the shape of the number eight plus a decimal point
  - Each LED is called a segment and labeled as ‘a’ through ‘g’.

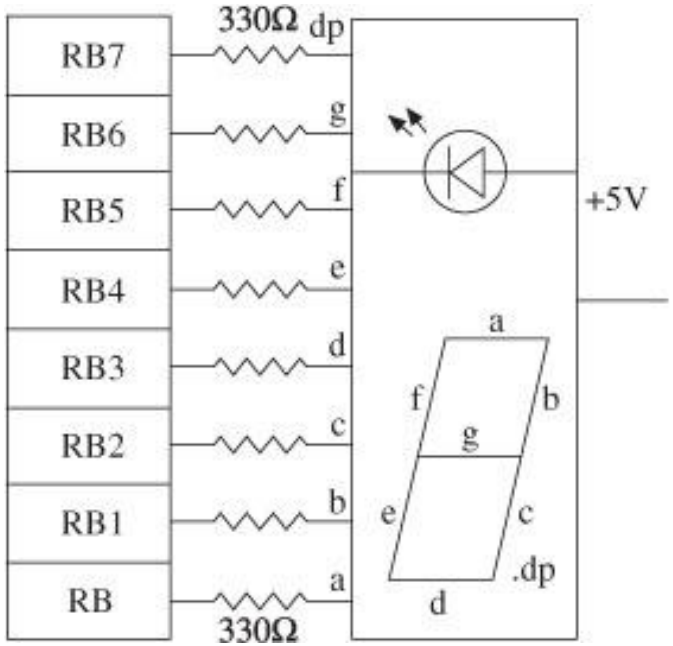
Interfacing Seven-Segment

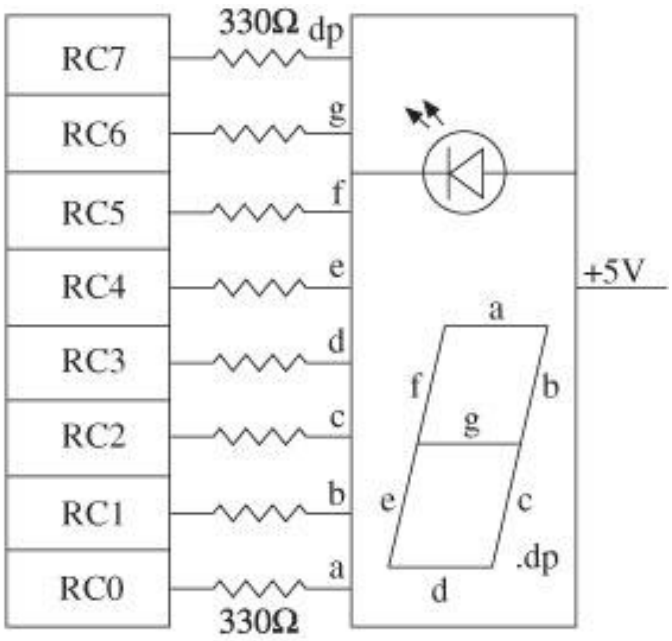


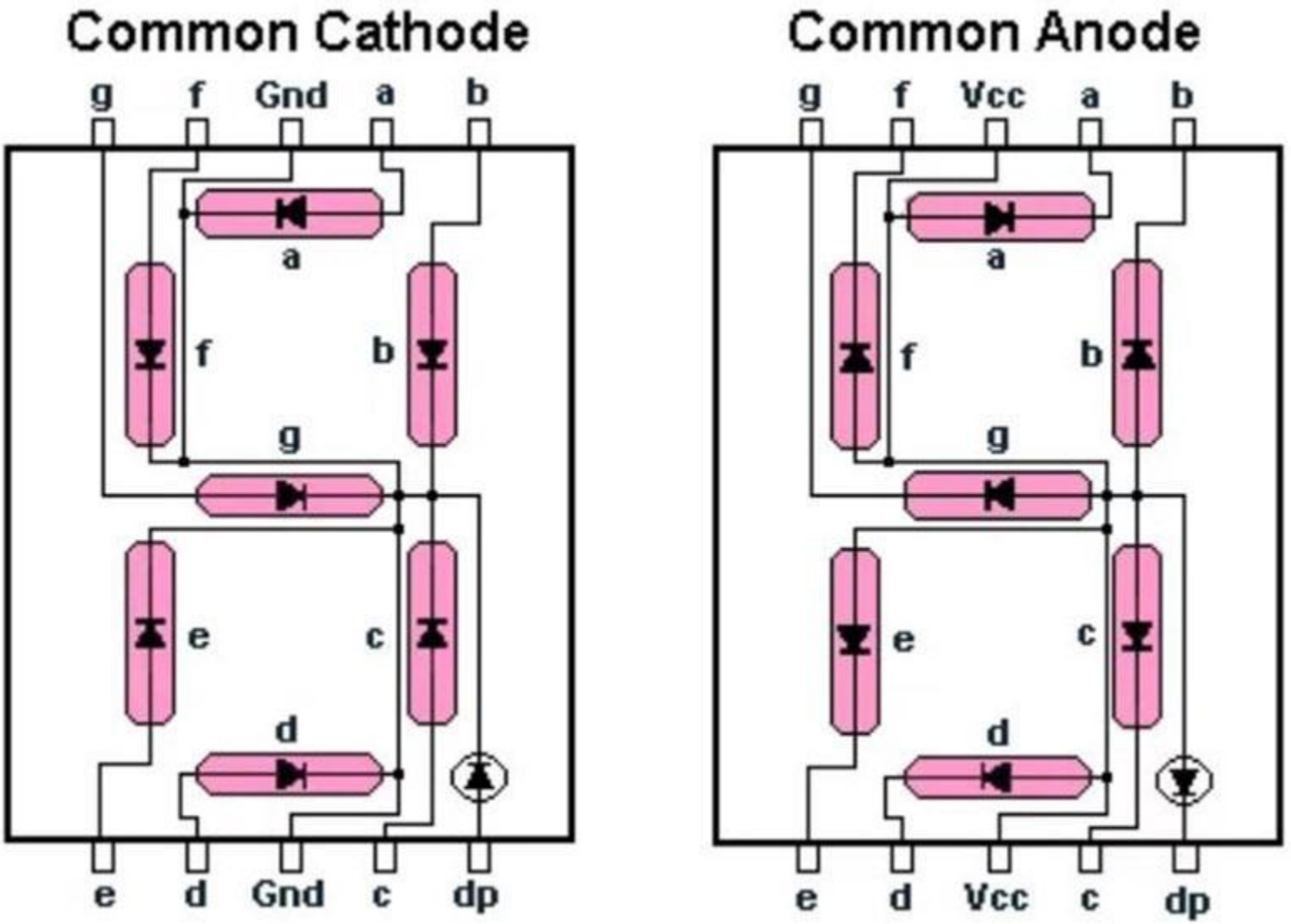
- Two types of seven-segment LEDs
  - Common anode
  - Common cathode



Interfacing Seven-Segment LEDS to PORTs



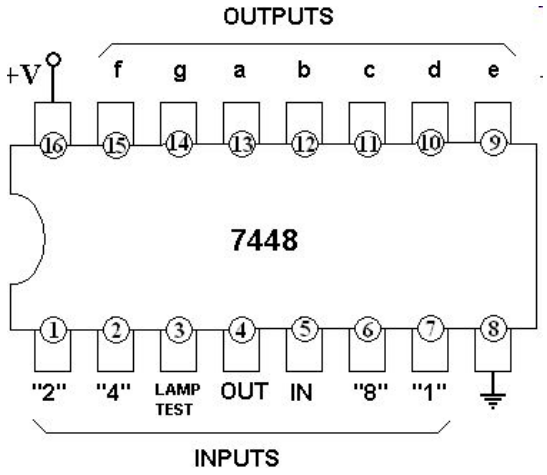


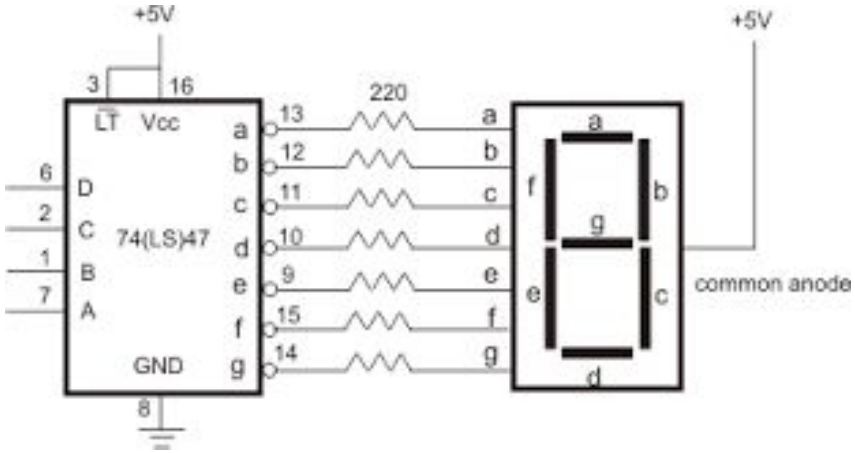


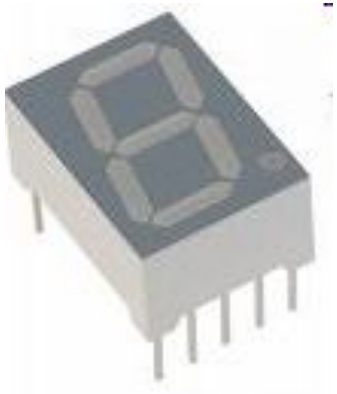
SSD Configuration

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	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

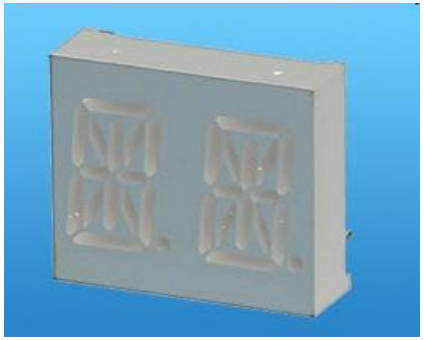
Seven-Segment Chips



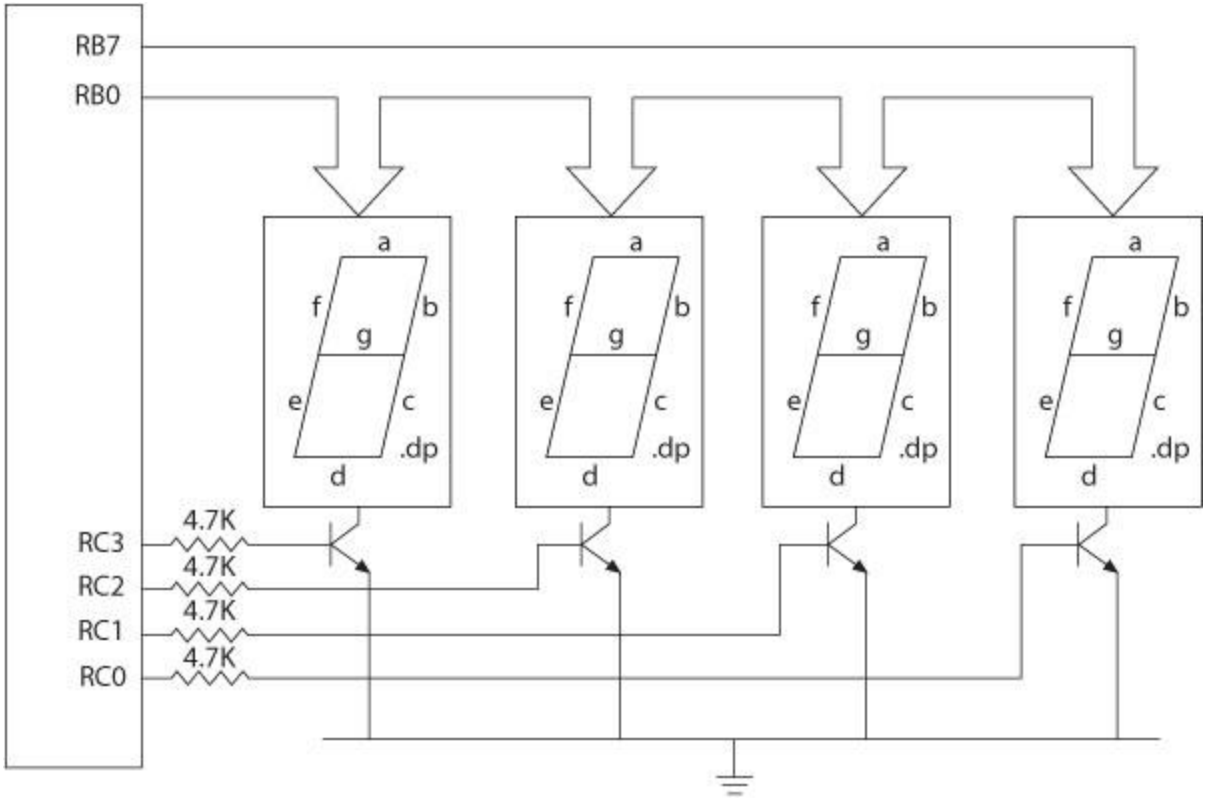




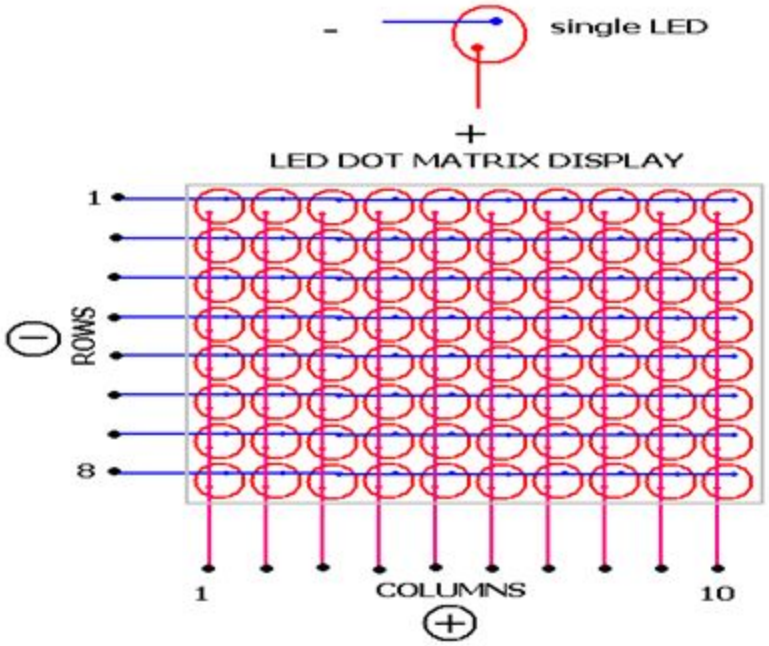
ALPHA/NUMERIC DISPLAY

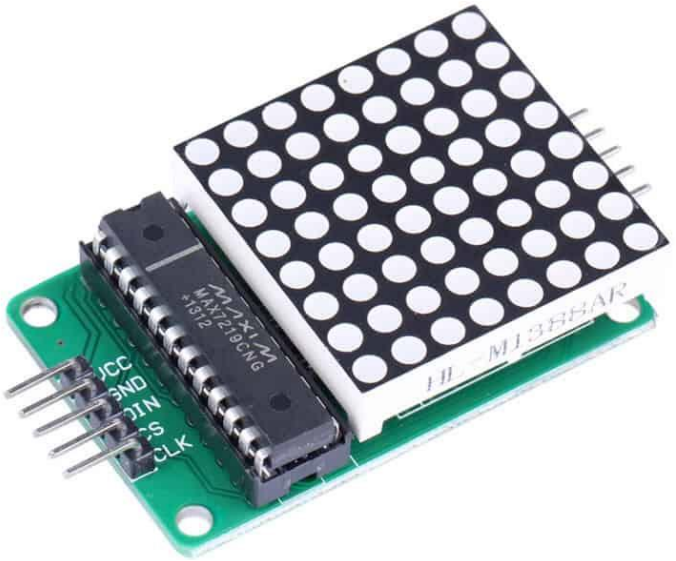


Interfacing to Multiple 7-Segments

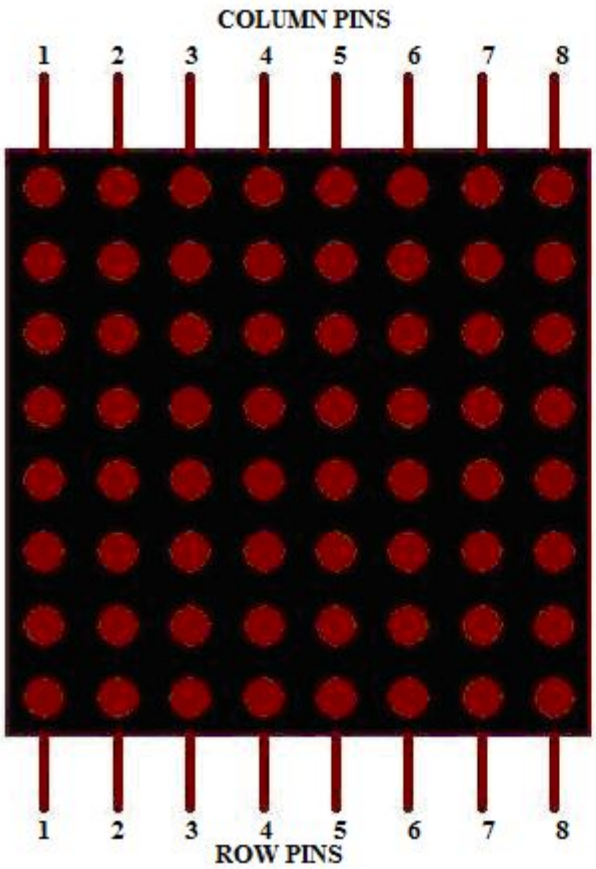


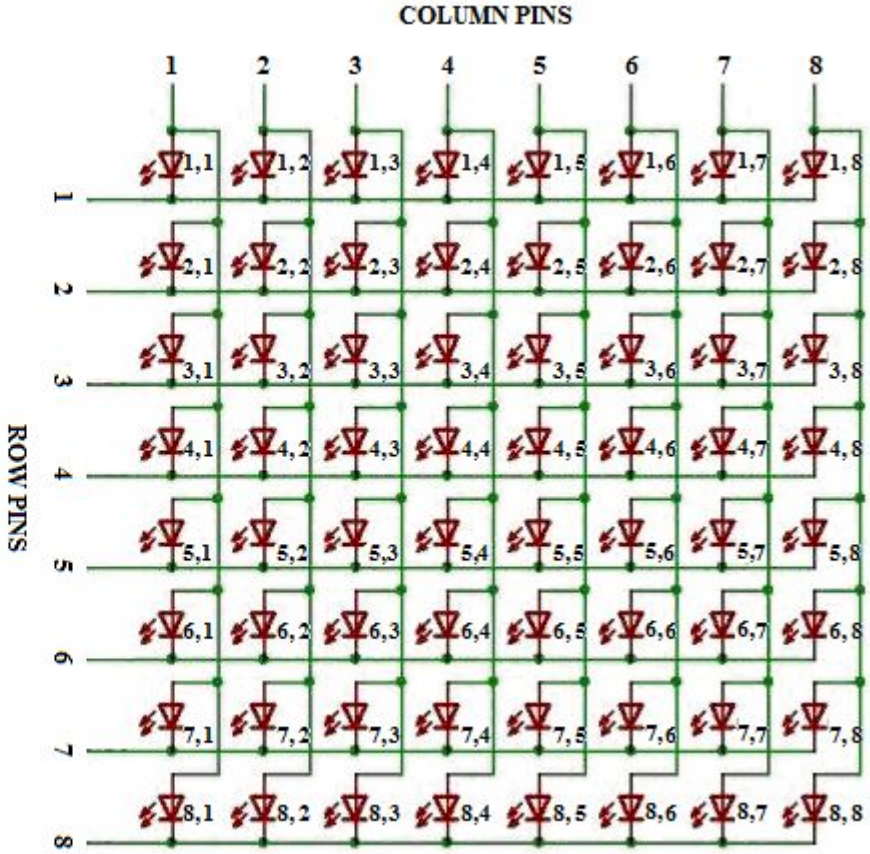
LED Matrix Display



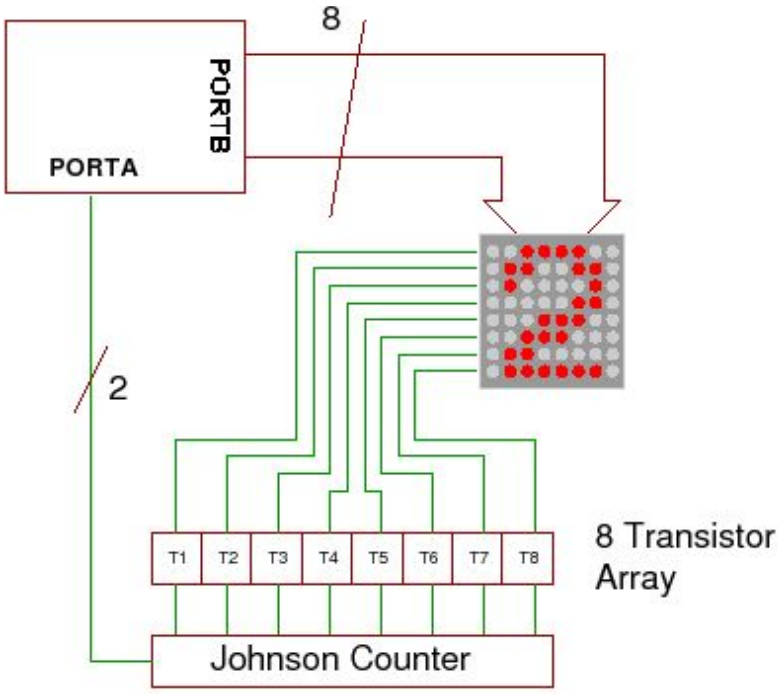


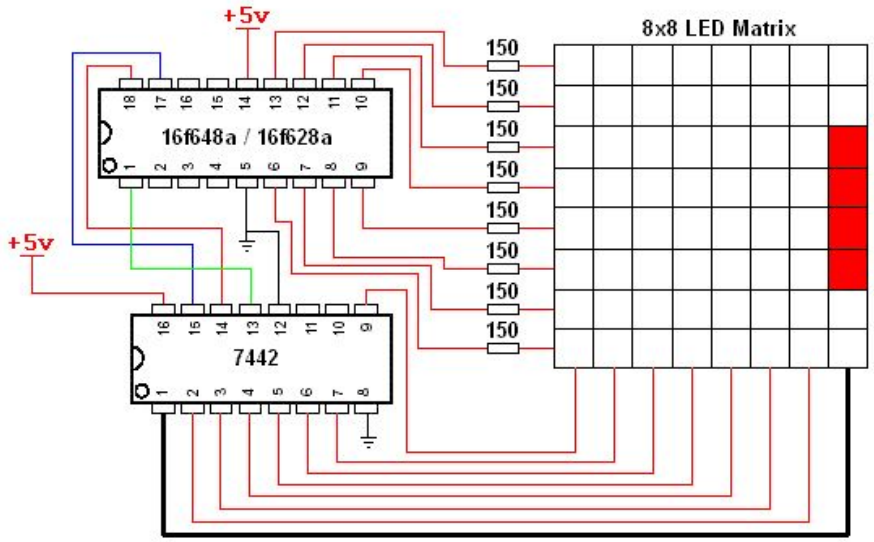
LED Matrix Display





LED Matrix Display





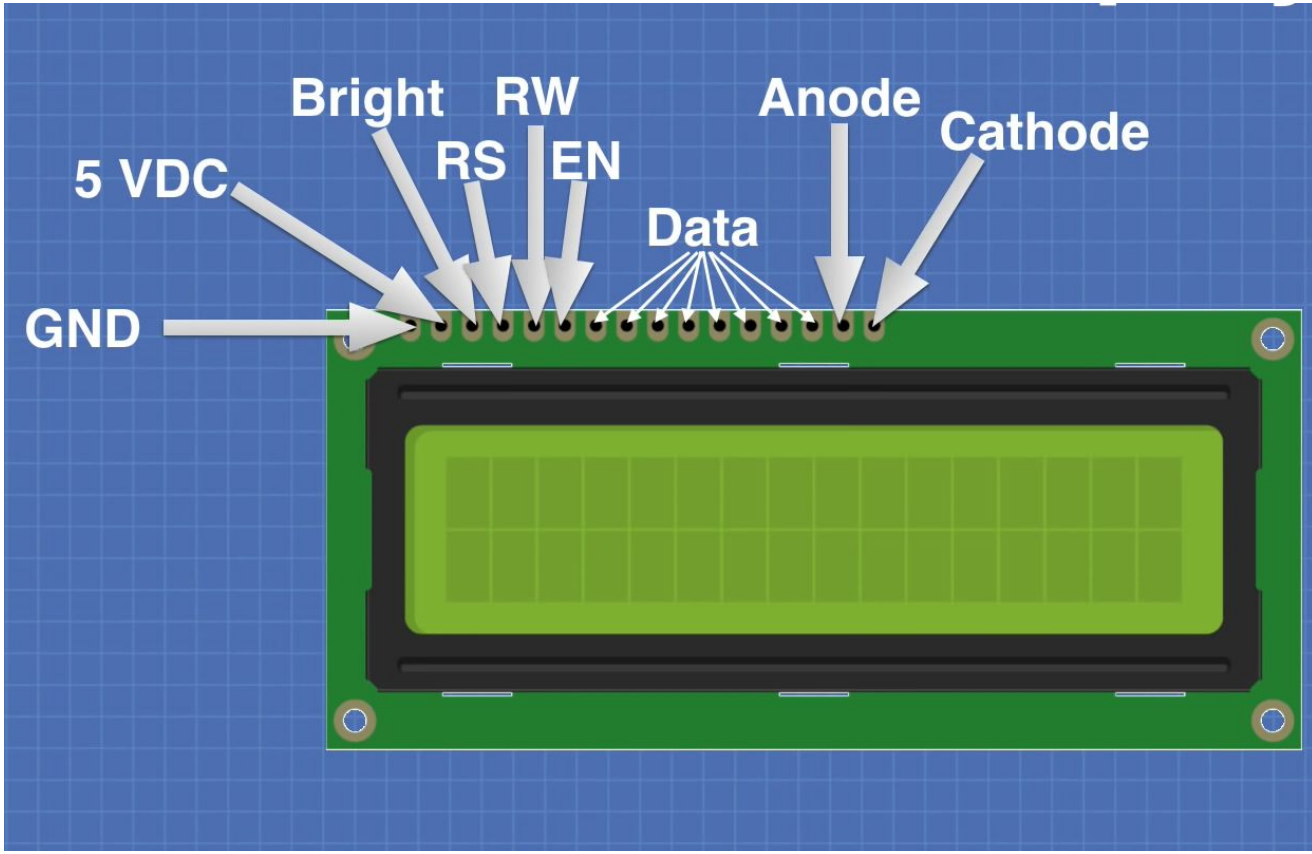
LCD

- Interface a 2-line x 16 character LCD module with the built-in controller to I/O ports of the PIC18 microcontroller
- LCD has a display Data RAM (registers) that stores data in 8-bit character code in ASCII.
  - Each register in Data RAM has its own address that corresponds to its position on the line.

LCD

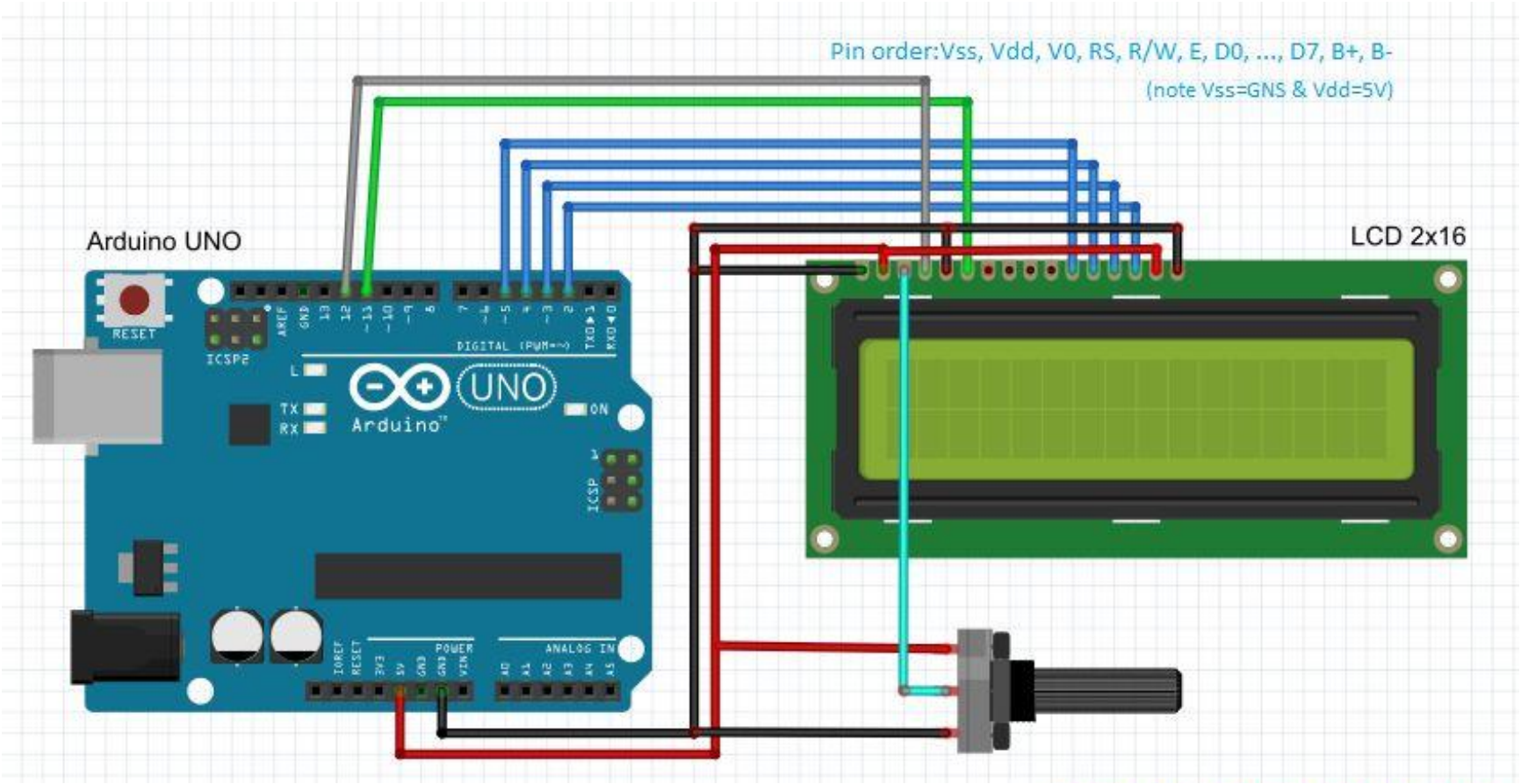
- Three control signals:
  - RS (Register Select)
  - RW (Read/Write)
  - E (Enable)
- Three power connections
  - Power (Vcc),
  - Ground
  - Variable register to control the brightness
- Driver has two 8-bit internal registers
  - Instruction Register (IR) to write instructions to set up LCD
  - Data Register (DR) to write data (ASCII characters)

LCD can be interfaced either in 8-bit or 4-bit mode

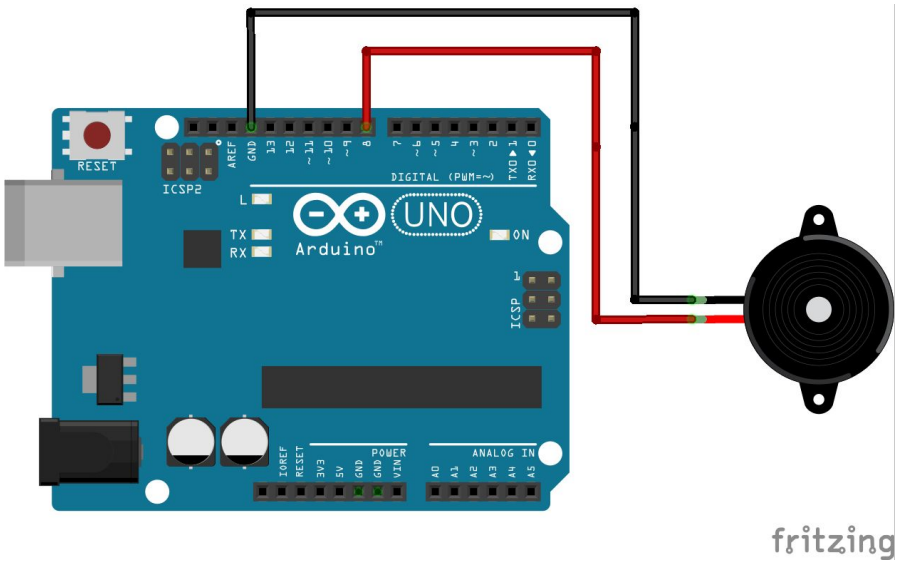


1	GND
2	5 VDC
3	Bright
4	RS
5	RW
6	EN
7	D0
8	D1
9	D2
10	D3
11	D4
12	D5
13	D6
14	D7
15	Anode
16	Cathode

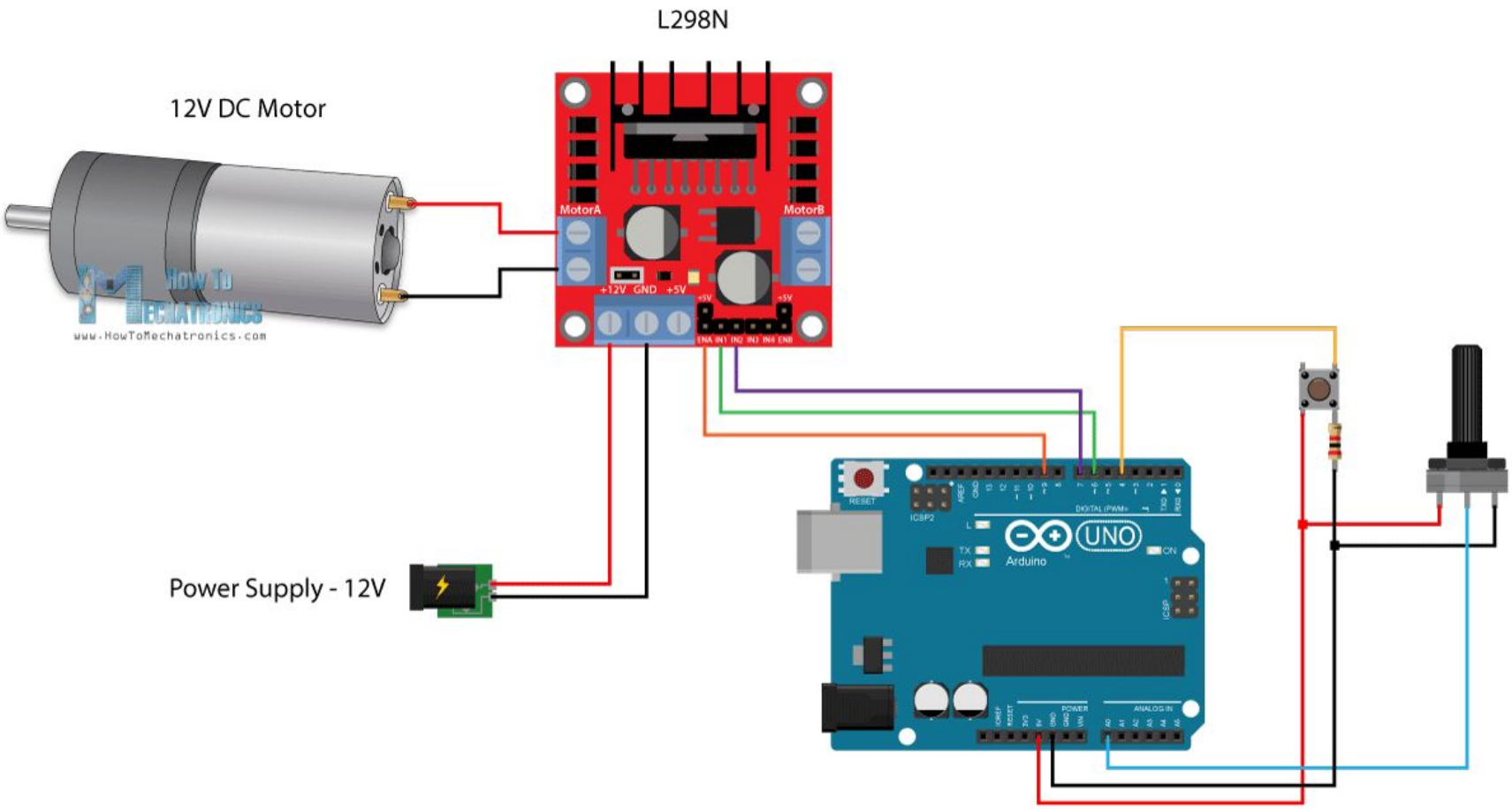
Connection with half byte mode



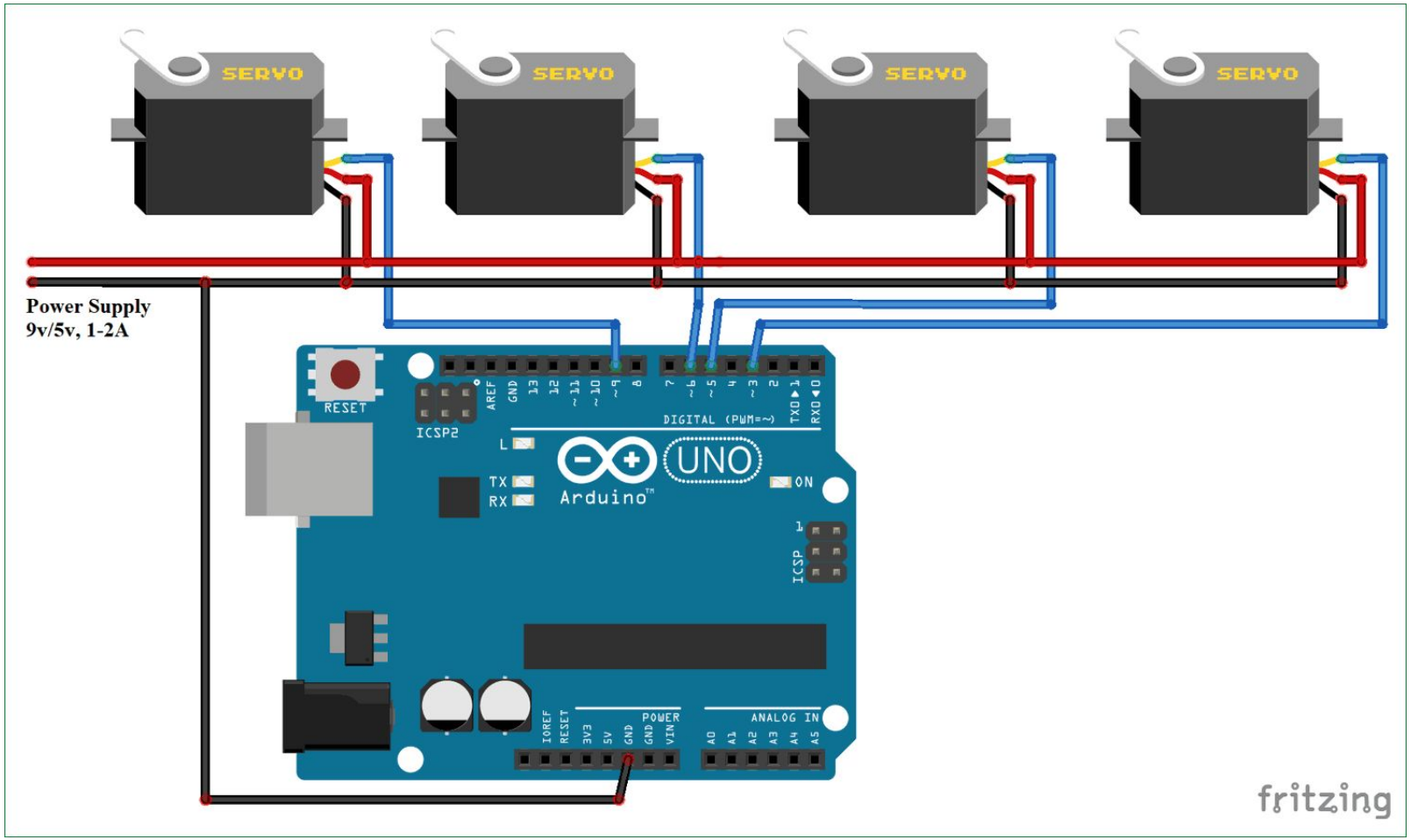
Buzzer


RESET
ICSP2
AREF
GND
DIGITAL (PWM=~)
TXD
RXD
L
TX
RX
UNO
Arduino
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
fritzing

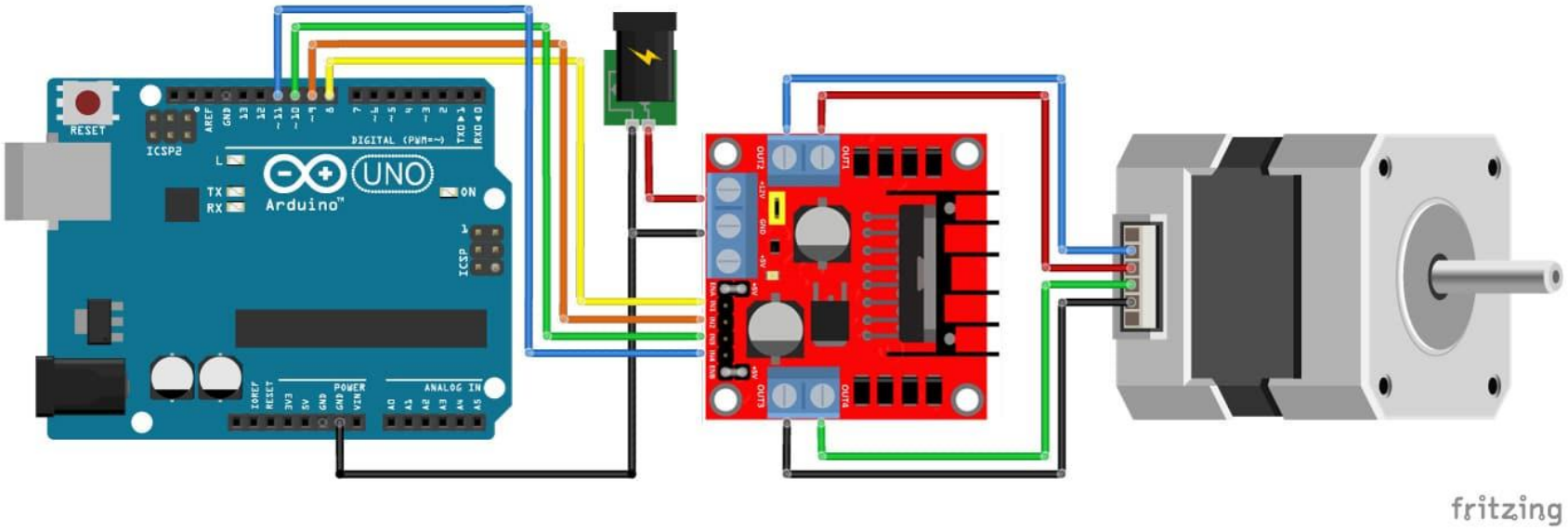
DC Motor


L298N
12V DC Motor
MotorA
MotorB
+12V GND +5V
ENA IN1 IN2 IN3 IN4 ENB
Power Supply - 12V
UNO
Arduino
www.HowToMechatronics.com

Servo Motor


SERVO
SERVO
SERVO
SERVO
Power Supply
9v/5v, 1-2A
RESET
ICSP2
DIGITAL (PWM=~)
UNO
Arduino™
TX
RX
ON
ICSP
POWER
ANALOG IN
fritzing

Stepper Motor

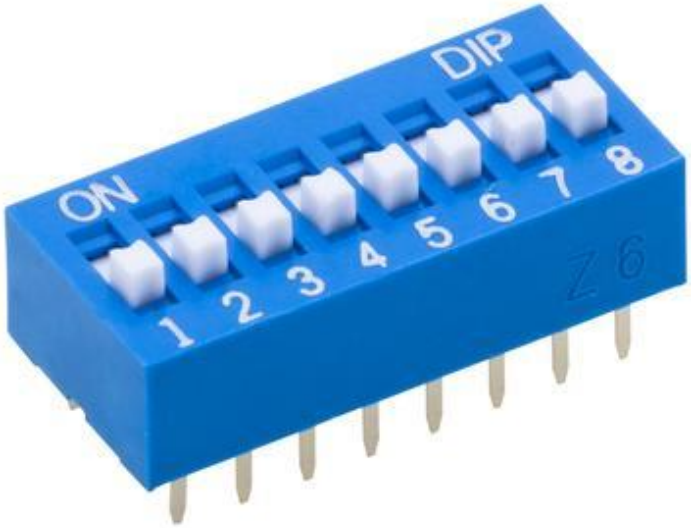


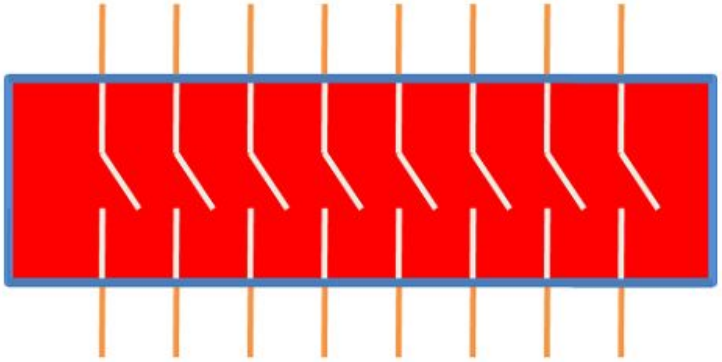
Input

- Dip Switch
- Push-Button Switch
- Matrix Keyboard
- Analog Input (Sensors)

DIP switch

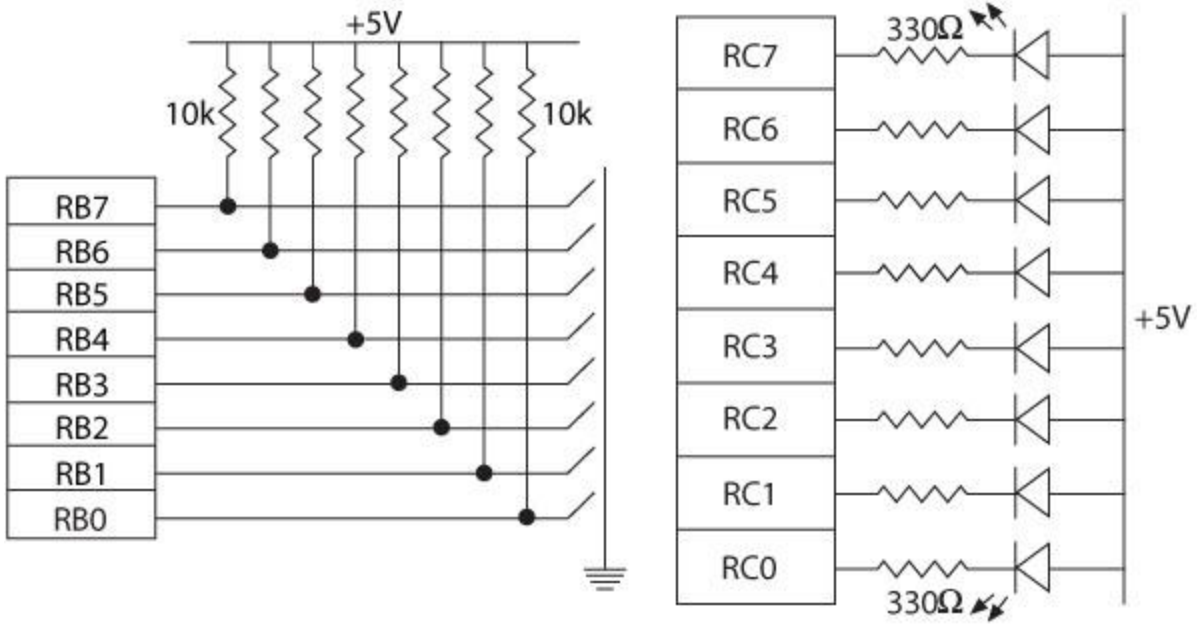
- One side of the switch is tied high (to a power supply through a resistor called a pull-up resistor), and the other side is grounded. The logic level changes when the position is switched.





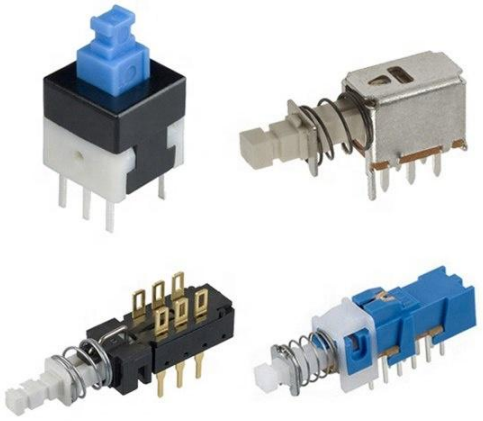
DIP Switch Schematic

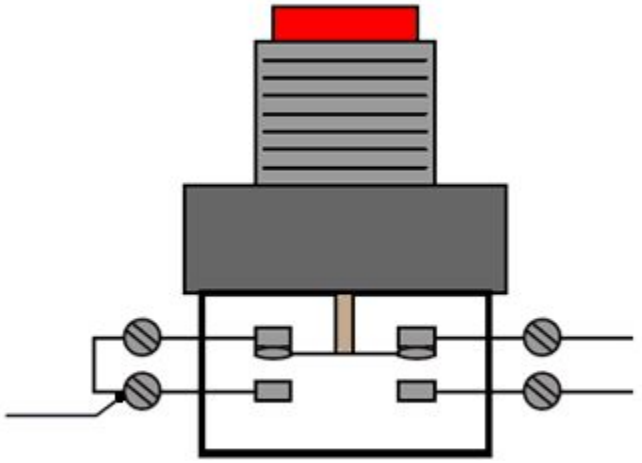
Interfacing Dip Switches and Interfacing LEDs



Push-button key

- 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.





Key Debouncing

- The reading of one contact as multiple inputs can be eliminated by a key-debounce technique, using either hardware or software.
- Software Key Debouncing
  - Wait for 20 ms.
  - Read the port again.
  - If the reading is still less than FFH, it indicates that a key is pressed.

Hardware Key Debounce Techniques

- Hardware technique is based on the principles of generating a delay and switching the logic level at a certain threshold level.
  - S-R latch: The output of the S-R latch is a pulse without a bounce.
  - Multiplexer that bounces the key internally and provides a steady output.

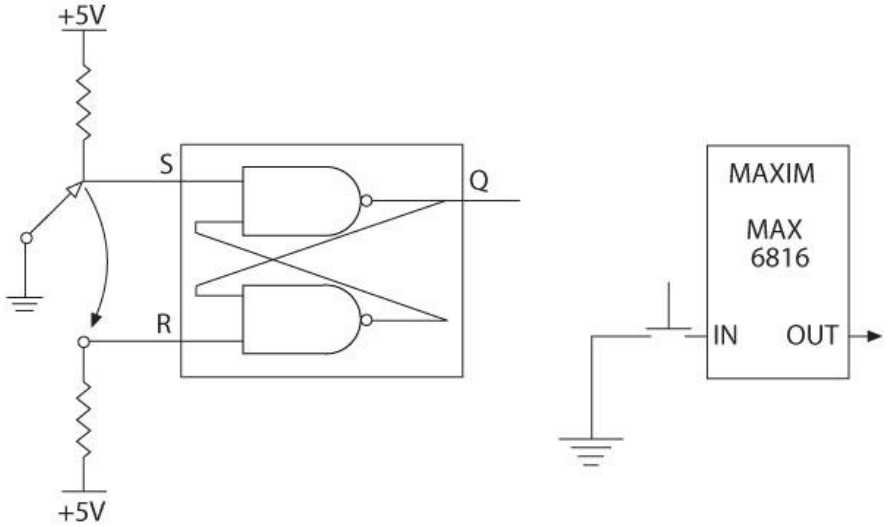
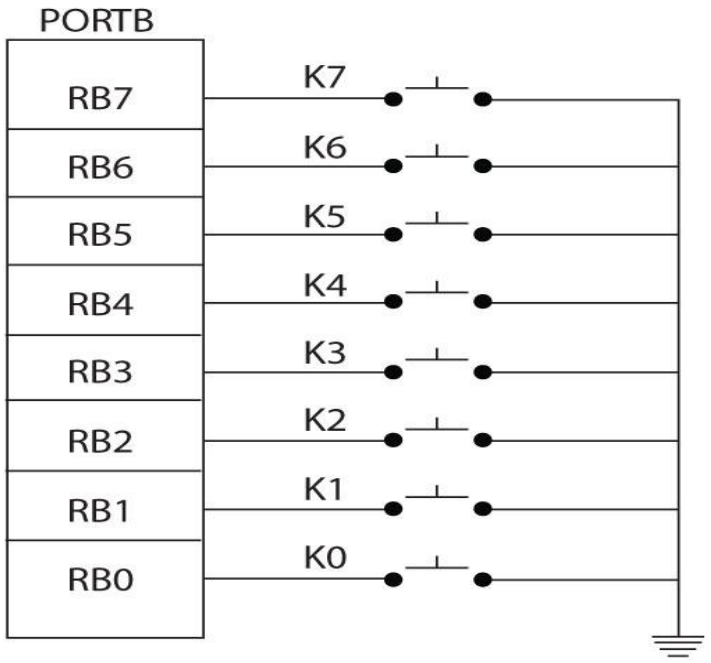
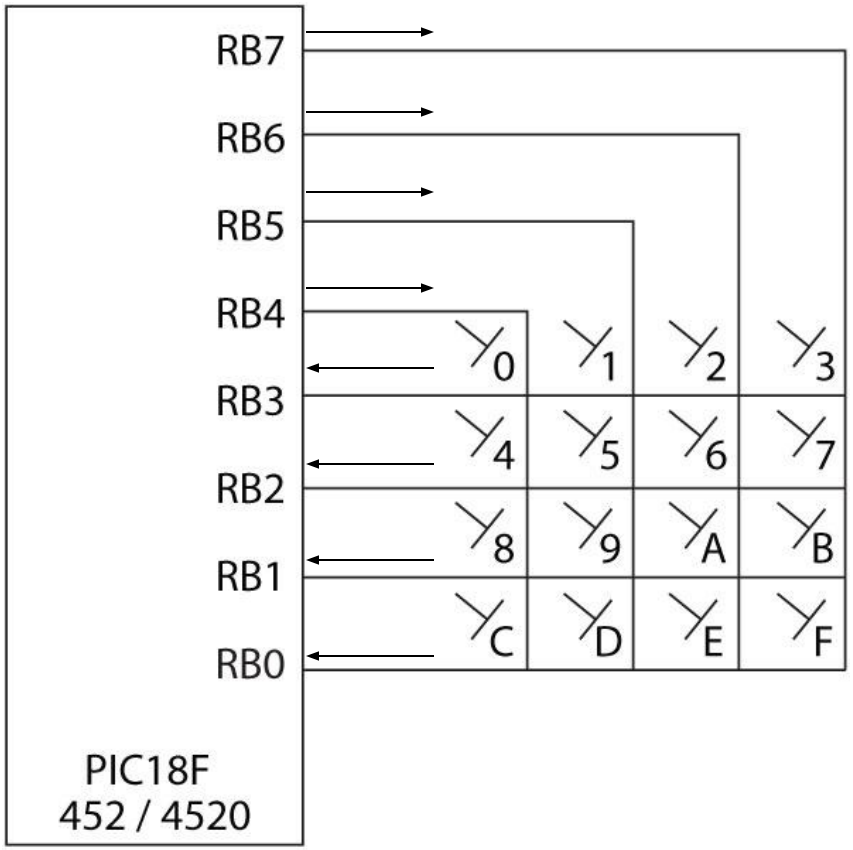


Illustration: Interfacing Push-Button Keys (1 of 6)

- Problem statement
  - A bank of push-button keys are connected as inputs to PORTB.
  - The pull-up resistors are internal to PORTB.
  - Write a program to recognize a key pressed, debounce the key, and identify its location in the key bank with numbers from 0 to 7.

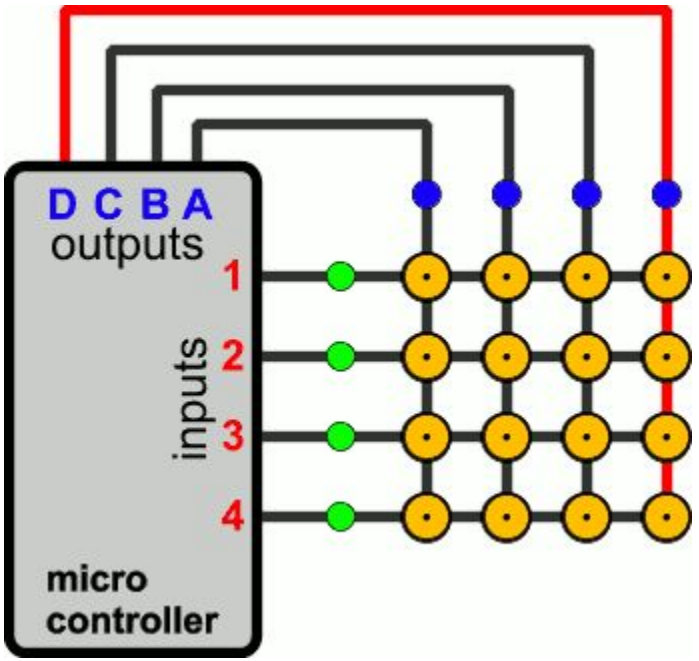


Interfacing a Matrix Keyboard



Interfacing a Matrix Keyboard

- Ground one column at a time and check all the rows in that column.
- Once a key is identified, it is encoded based on its position in the column.



Sensors will be discussed in next session…