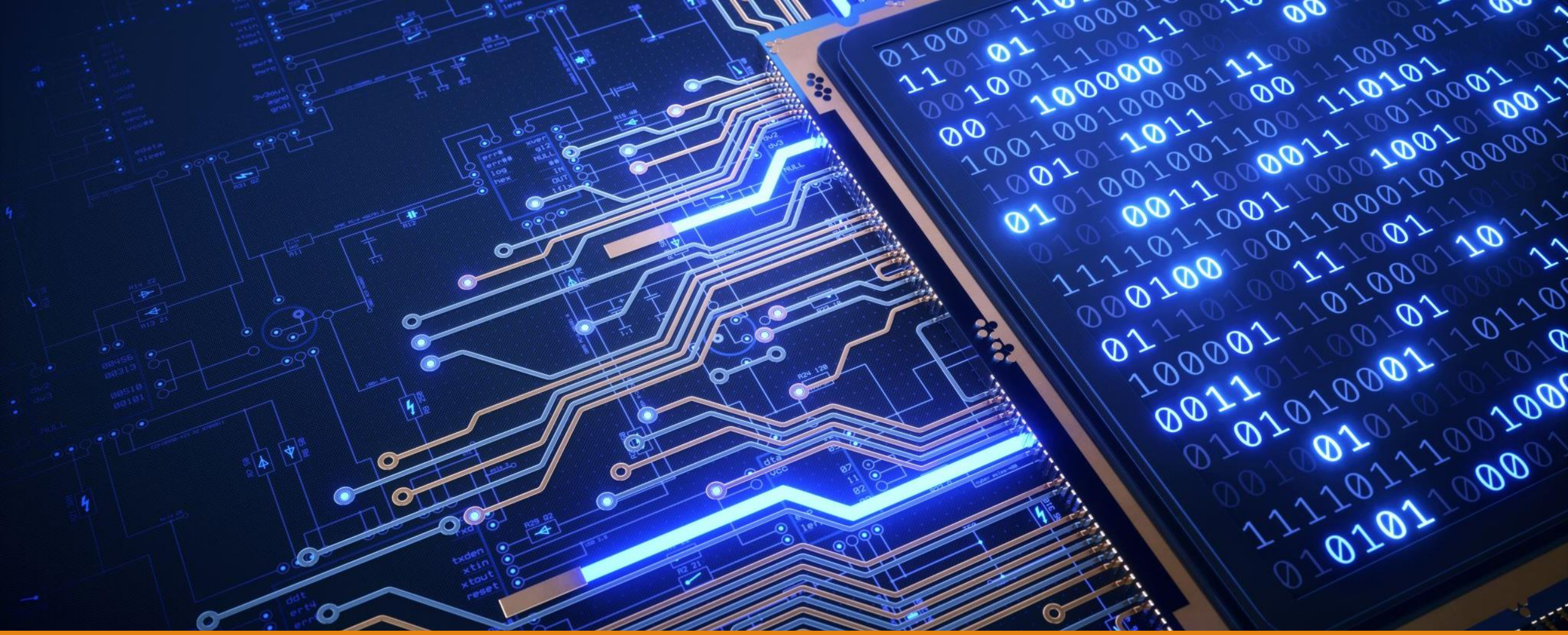


# Embedded Systems

Dr. Abdelhay Ali

# Lecture 2

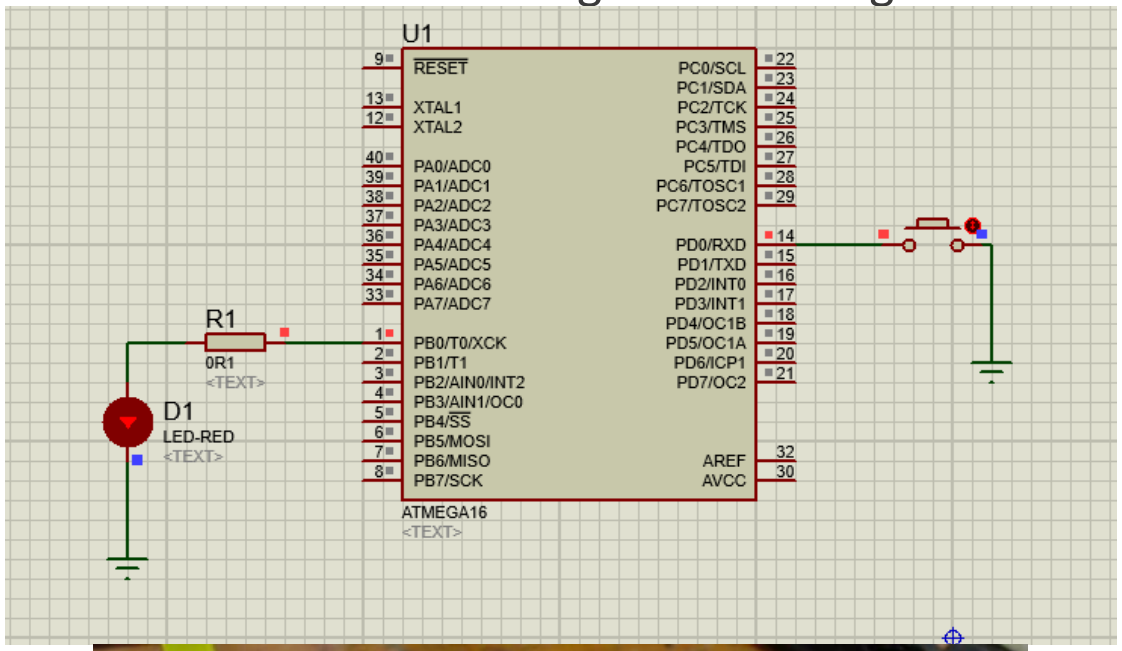
# I/O

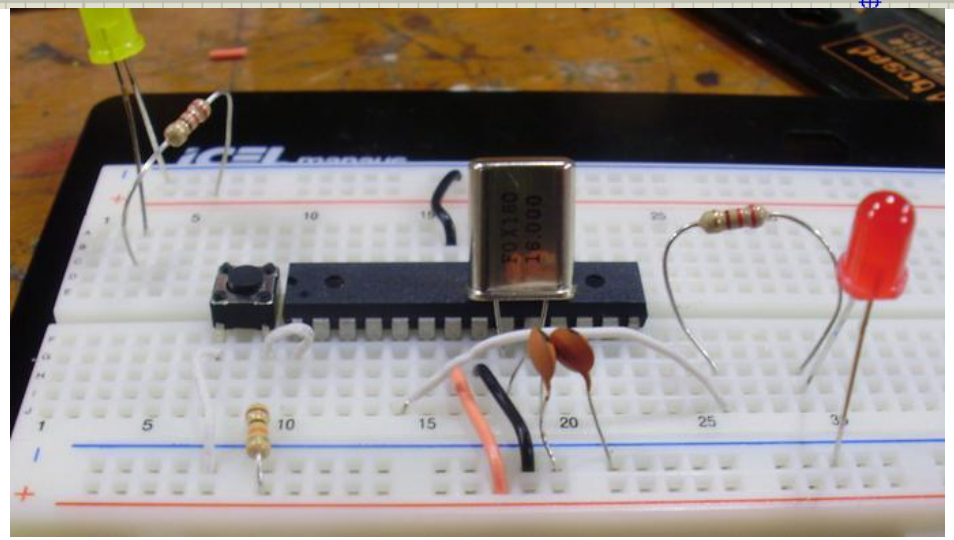
# Outlines

1. LED Blinking
2. Seven segment
3. Keypad
4. 16x2 LCD

# Example 1

## LED and Push Button Program in Atmega16-AVR

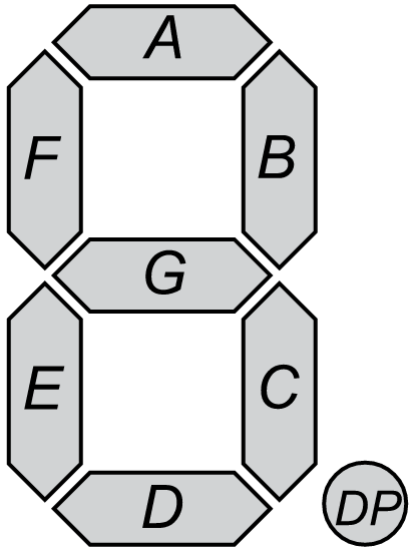


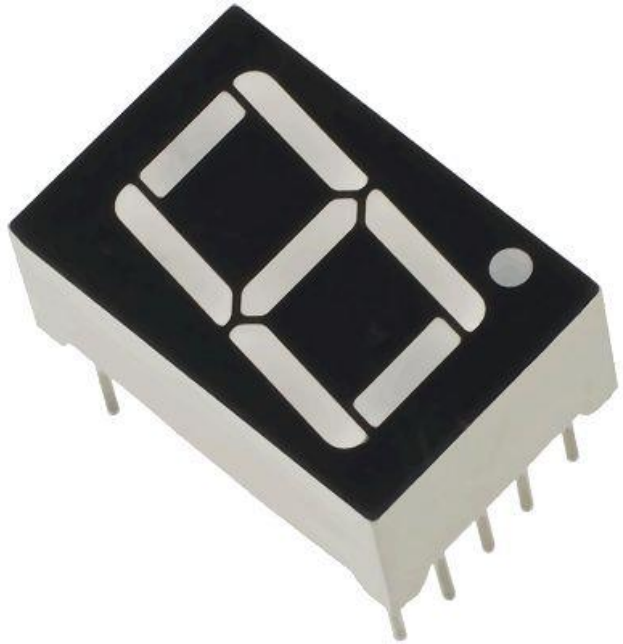



```
#include <mega16.h>
 #include <delay.h>
void main(void)
{
DDRB=0x01;       //Pin B0 Output
PORTB=0x00;

DDRD=0x00; //Pin D0 input
PORTD=0x01; //Pin D0 pull up

while (1)
    {
     if(PIND  & (1<<0) == 1)// checking the status of PIN
      {
         delay_ms(100); // for debouncing of switch
        PORTB=PORTB | (1<<0);//0x01;   // Turning on the LED
      }
        else
      {
        delay_ms(100); // For debouncing of switch
         PORTB=PORTB & !(1<<0);// Turning off the LED
      }
   }
}
```

# 7 Segment

7-segment LED displays are formed by LED segments. It is basically used to display numerical values from 0 to 9. One more segment is also present there which is used as decimal point.

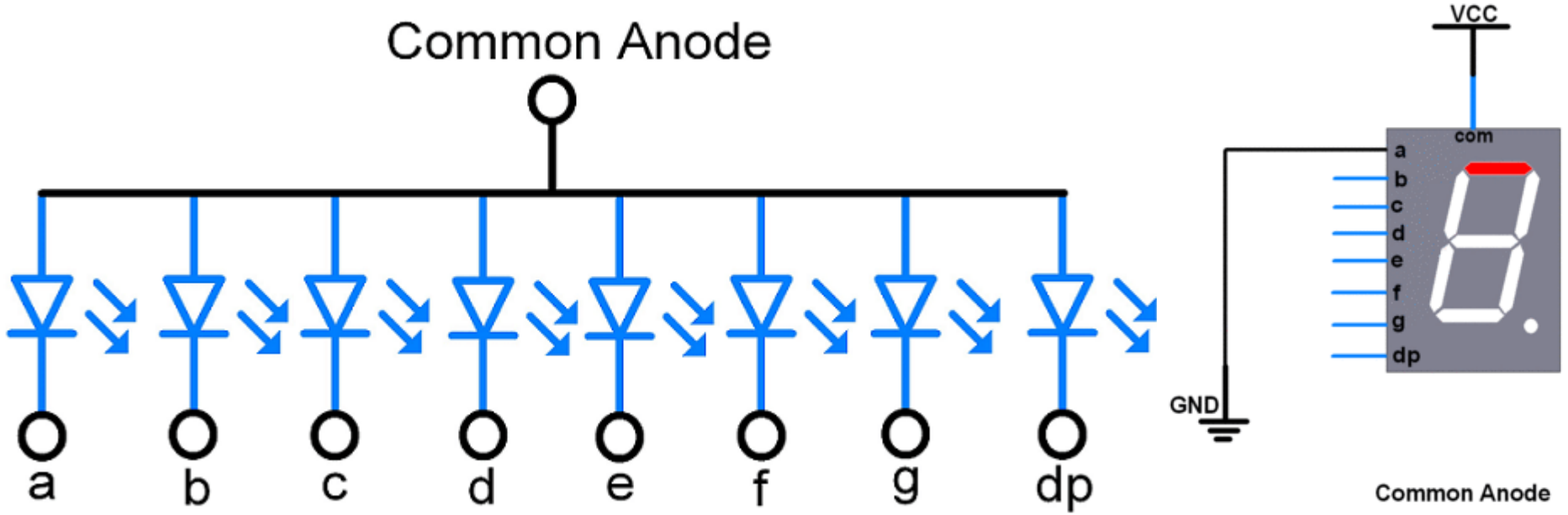


# 7 Segment

1. **Common Anode (CA)**

In common anode display, all anode pins are connected together to VCC and LEDs are controlled via cathode terminals. It means to turn ON LED (segment), we have to make that cathode pin logic LOW or Ground.





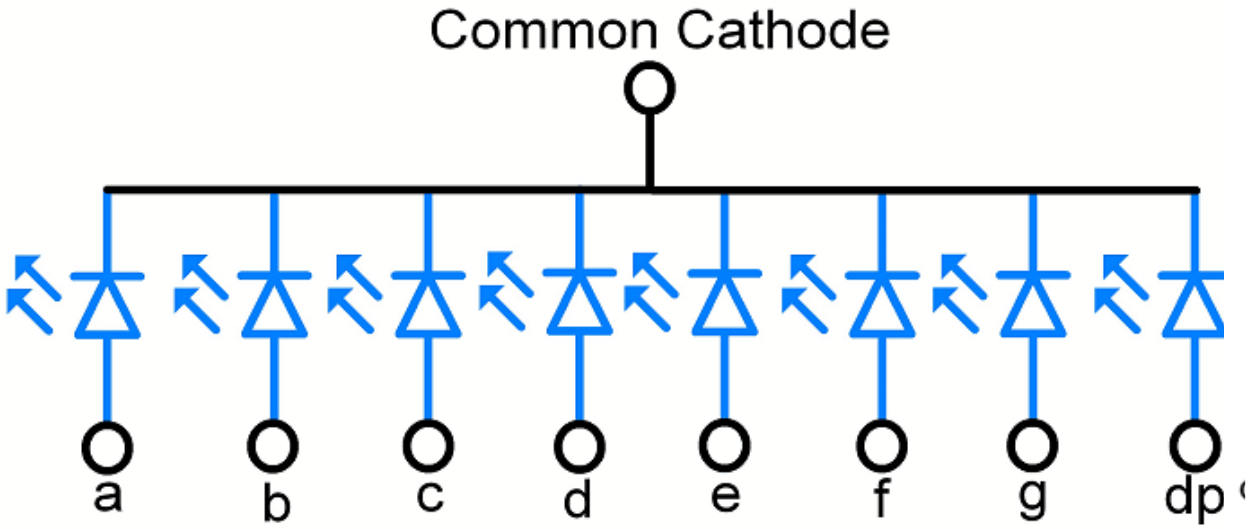


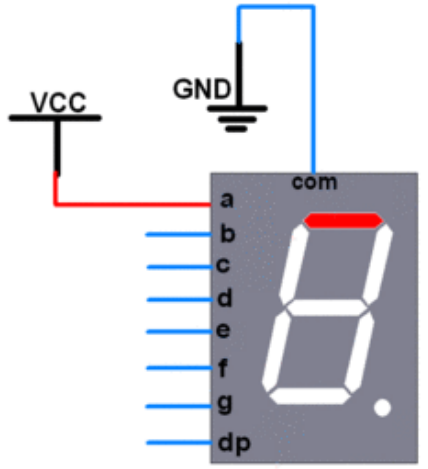
Common Anode

# 7 Segment

2. **Common Cathode (CC)**

In common cathode display, all cathode pins are connected together and led are controlled via anode terminal. It means to turn ON LED (segment), we have to apply proper voltage to the anode pin.



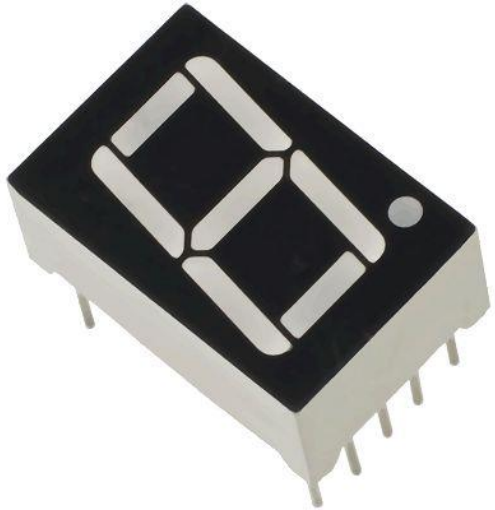



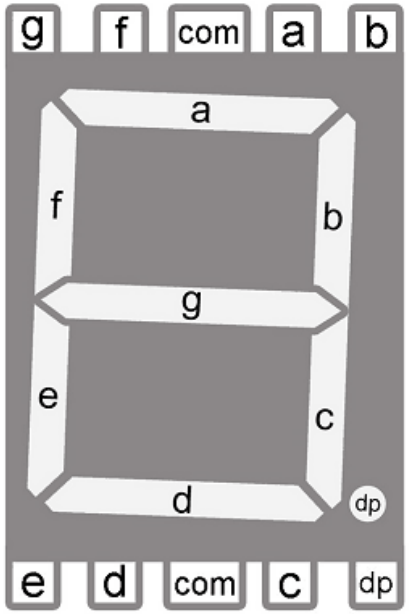

Common Cathode

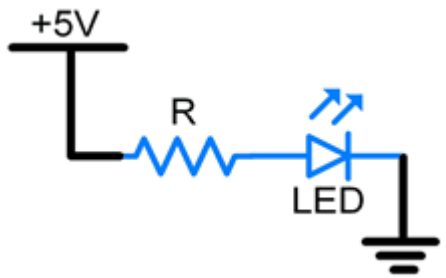
# 7 Segment

**Pin Diagram**

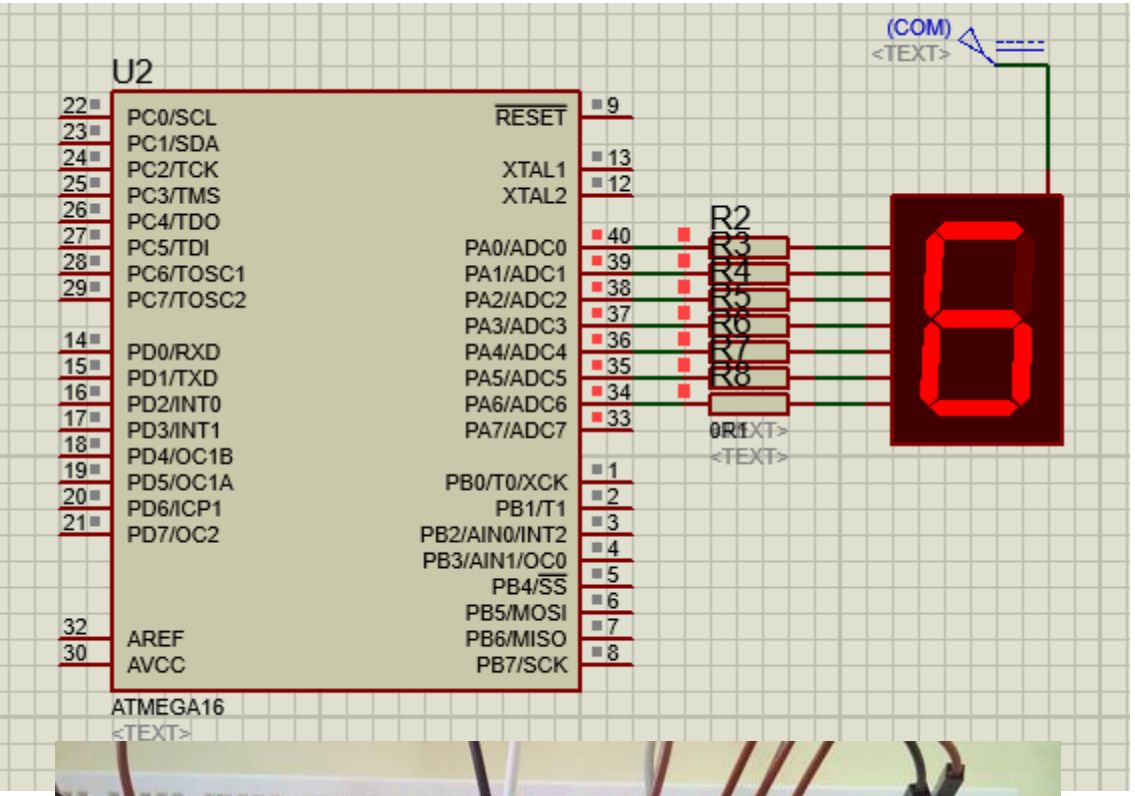
- 7-segment display has total 10 pins.
- The common pin (com) is connected either in common anode or common cathode configuration.

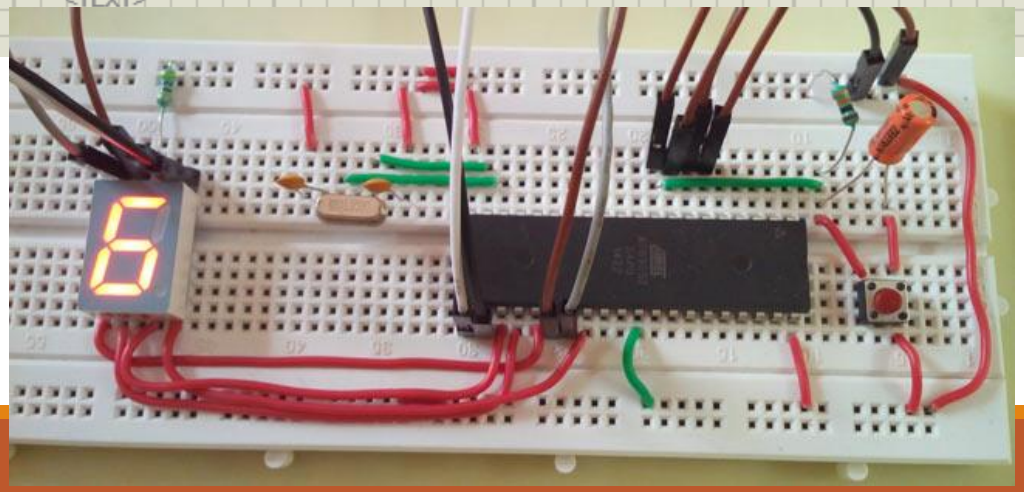




# 7 Segment



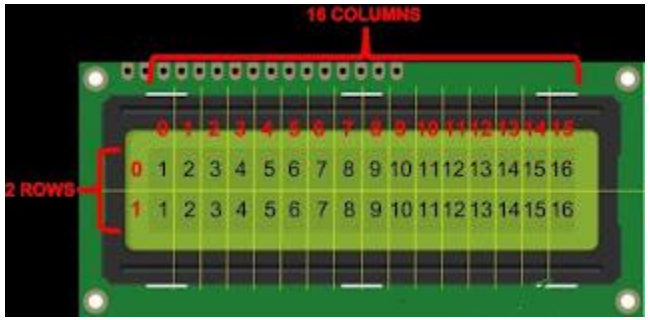



```
#include <mega16.h>
 #include <delay.h>
void main(void)
{
 char
array[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
   int i;

PORTA=0x00;
DDRA=0xFF;
while (1)
    {
      for( i=0;i<10;i++) {
              PORTA = array[i]; /* write data on to the LED port */
               delay_ms(1000); /* wait for 1 second */
       }

    }
}
```

# LCD16x2

LCDs (Liquid Crystal Displays) are used for displaying status or parameters in embedded systems.

LCD16x2 has 2 lines with 16 characters in each line. Each character is made up of 5x8 (column x row) pixel matrix.

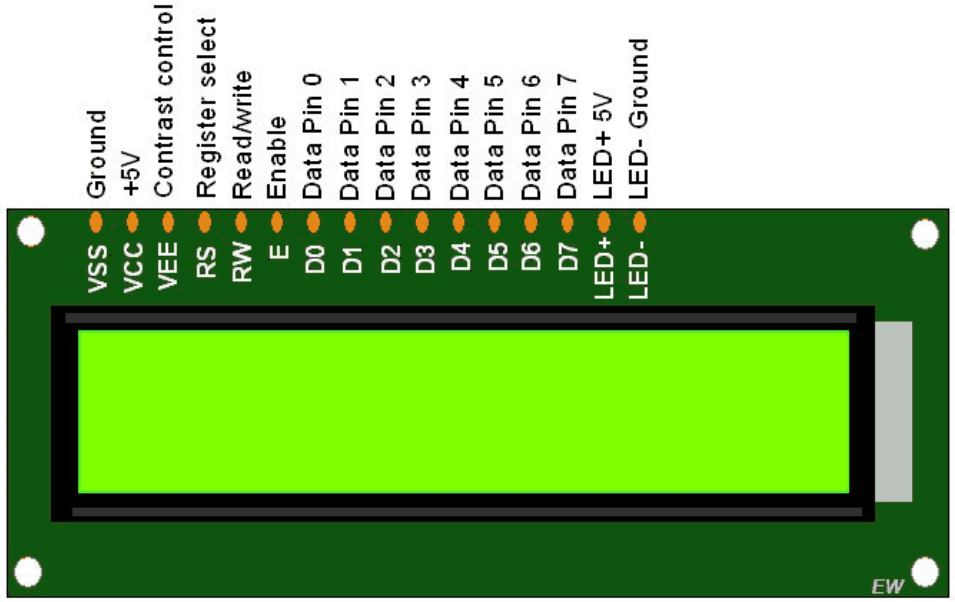
# LCD16x2

LCD 16x2 is a 16 pin device which has 8 data pins (D0-D7) and 3 control pins (RS, RW, EN). The remaining 5 pins are for supply and backlight for the LCD.

The control pins help us configure the LCD in command mode or data mode. They also help configure read mode or write mode and also when to read or write.

LCD 16x2 can be used in 4-bit mode or 8-bit mode depending on the requirement of the application.

In order to use it, we need to send certain commands to the LCD in command mode and once the LCD is configured according to our need, we can send the required data in data mode.

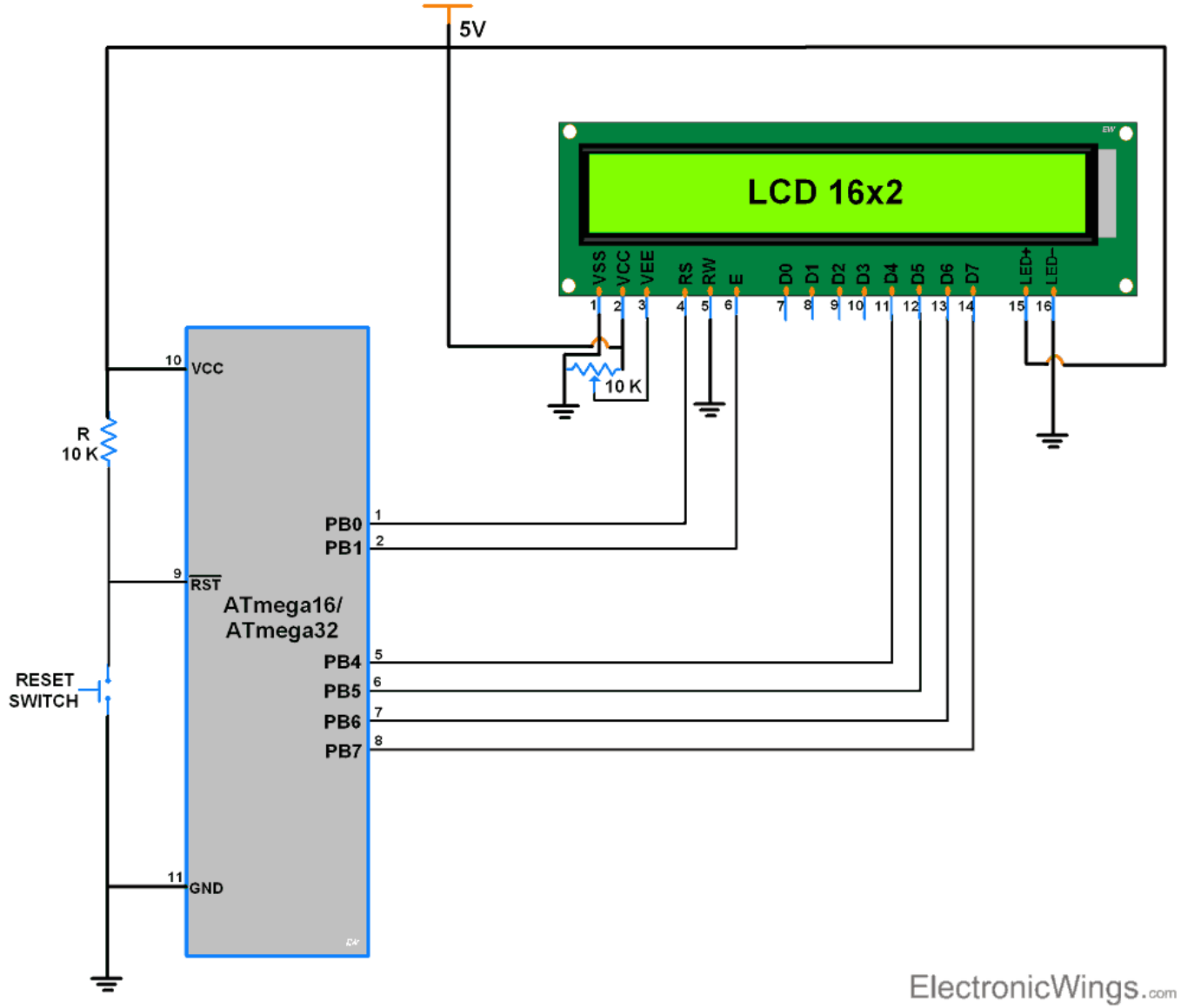


# LCD16x2

**Interfacing Diagram**

[LCD] [AVR Port]

RS (pin4) ------ bit 0
RD (pin 5) ------ bit 1
EN (pin 6) ------ bit 2
DB4 (pin 11) --- bit 4
DB5 (pin 12) --- bit 5
DB6 (pin 13) --- bit 6
DB7 (pin 14) --- bit 7



# LCD16x2
# #include <alcd.h>

**LCD Functions:**

## unsigned char lcd_init(unsigned char lcd_columns)

initializes the LCD module, clears the display and sets the printing character position at row 0 and column 0. The numbers of columns of the LCD must be specified (e.g. 16). No cursor is displayed. The function returns 1 if the LCD module is detected and 0 if it is not.
This is the first function that must be called before using the other high level LCD Functions.

## void lcd_clear(void)

clears the LCD and sets the printing character position at row 0 and column 0.

## void lcd_gotoxy(unsigned char x, unsigned char y)

sets the current display position at column x and row y. The row and column numbering starts from 0.

# LCD16x2
# #include <lcd.h>

---

**void lcd_putchar(char c)**

displays the character c at the current display position.

**void lcd_puts(char *str)**

displays at the current display position the string str, located in RAM.

**void lcd_putsf(char flash *str)**

displays at the current display position the string str, located in FLASH.

# LCD16x2
# #include <lcd.h>

**Example**

*/* after defining the port to which the LCD is connected and including the lcd.h file*/*

*/*/the value of the character variable 'P' is written in the 1st position of the first line (0,0)*/*

**lcd_putchar(P);**

*//move the cursor to the middle of the second line*

**lcd_gotoxy(8,1);**

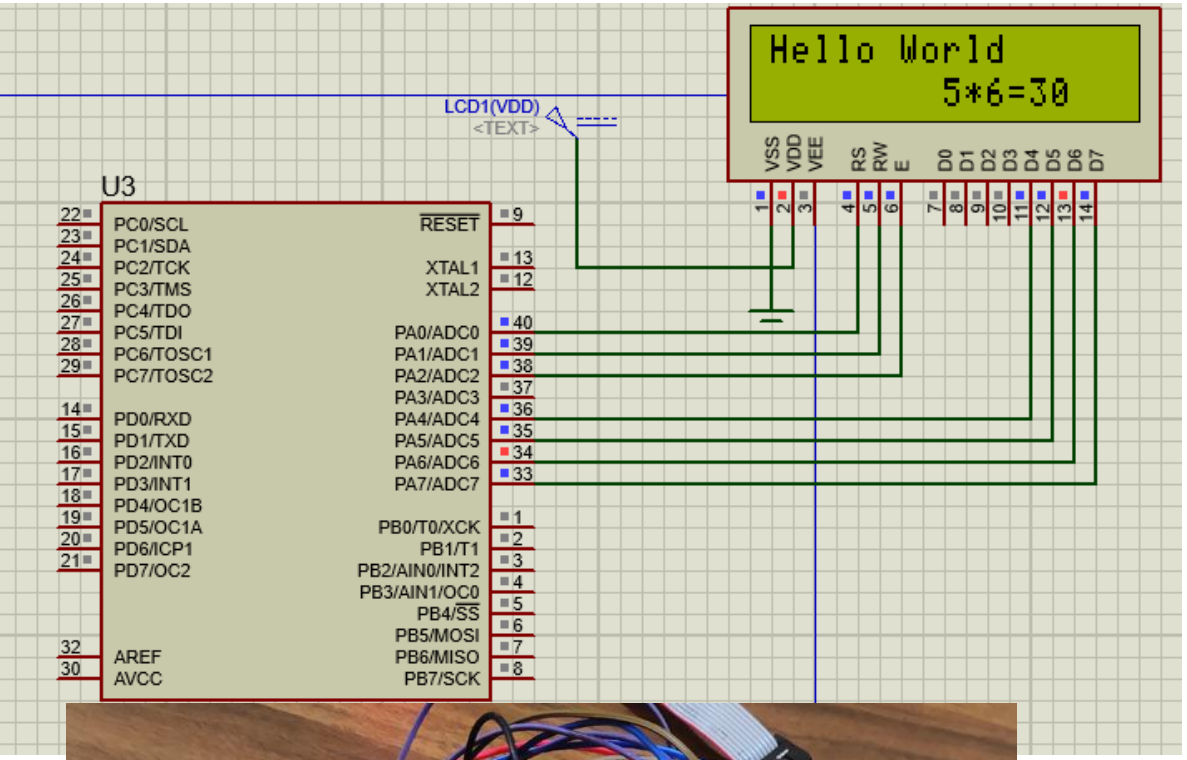
*// Write the words "Hi World"*

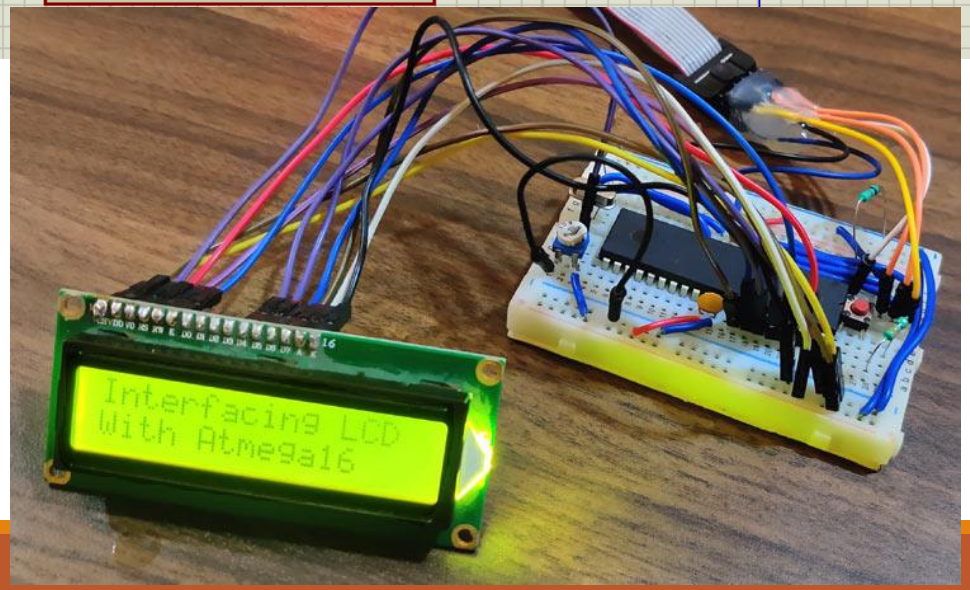
**lcd_putsf("Hi World");**

*//now clear the display*

**lcd_clear();**

# LCD Example




```
#include <mega16.h>
 #include <delay.h>
#include <alcd.h>
void main(void)
{
 lcd_init(16);
while (1)
      {
      lcd_gotoxy(0,0);
      lcd_putsf("Hello World");
      delay_ms(1000);
      lcd_clear();
      lcd_gotoxy(8,1);
      lcd_putsf("5*6=30");
      delay_ms(1000);        }

      }
}
```

# 4x4 Keypad

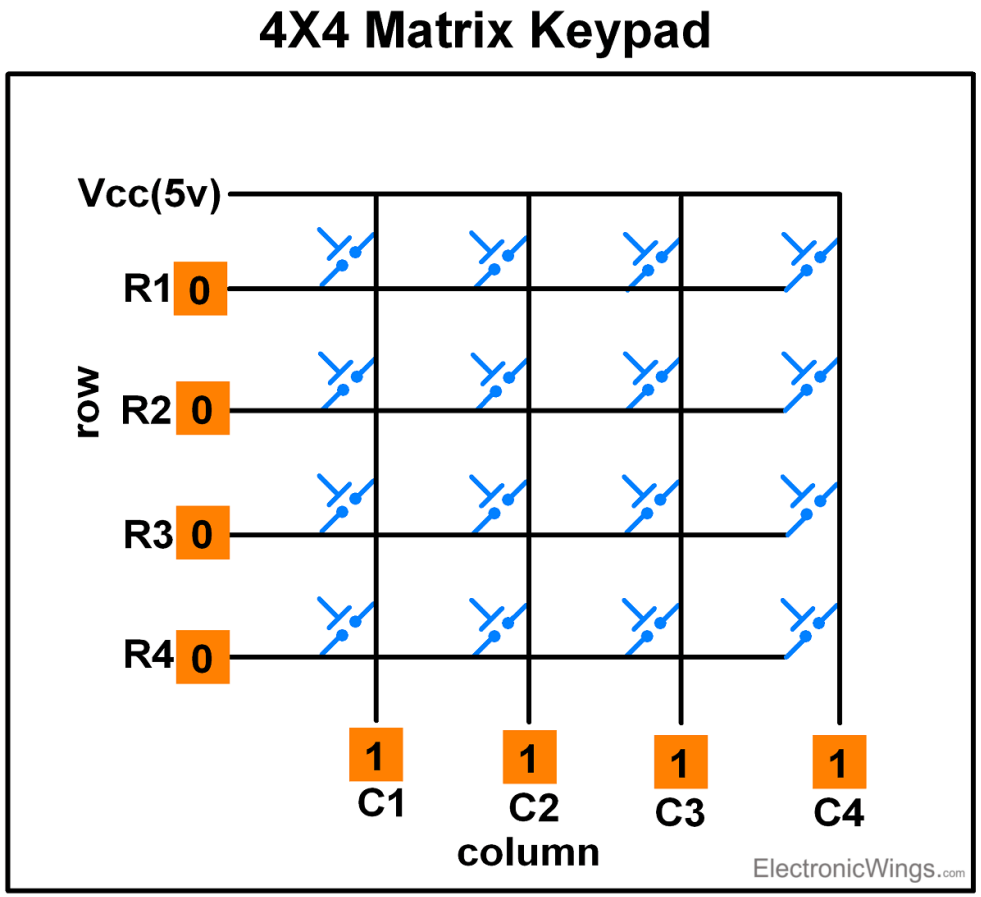
The keypad is used as an input device to read the key pressed by the user and to process it.

4x4 keypad consists of 4 rows and 4 columns. Switches are placed between the rows and columns. A keypress establishes a connection between the corresponding row and column between which the switch is placed

In order to read the keypress, we need to configure the **rows as outputs** and **columns as inputs**.

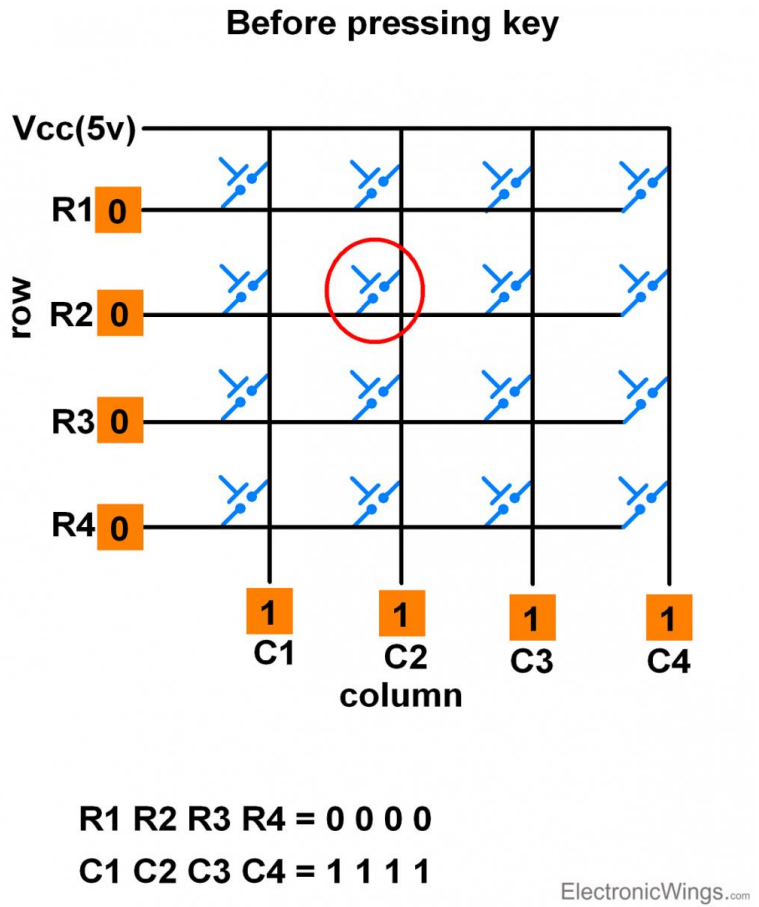
Columns are read after applying signals to the rows in order to determine whether or not a key is pressed and if pressed, which key is pressed.

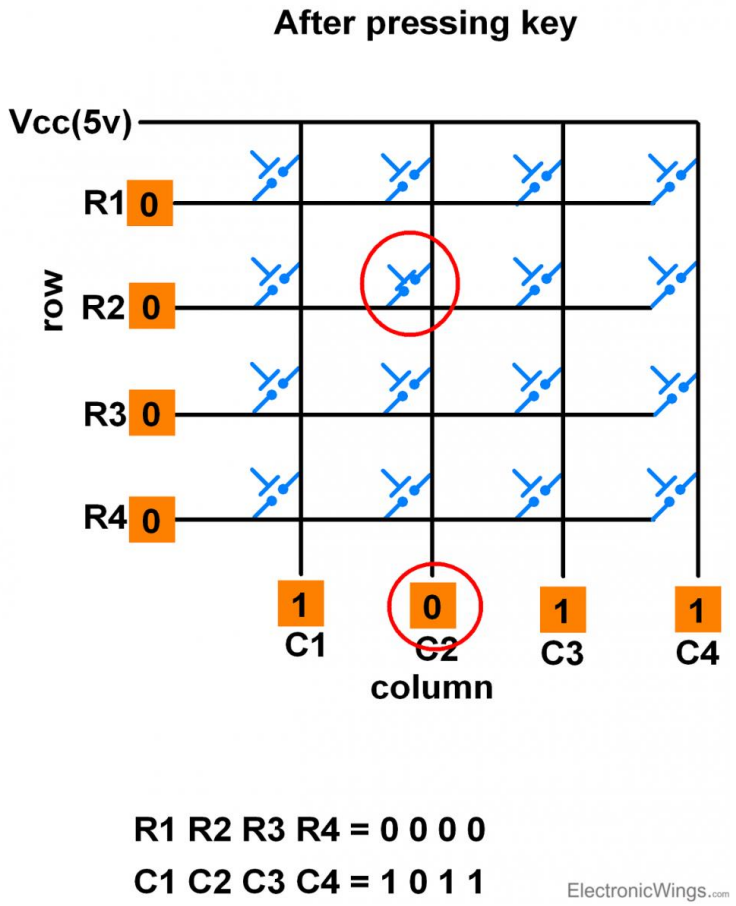
# 4x4 Keypad


4X4 Matrix Keypad
Vcc(5v)
R1 0
R2 0
R3 0
R4 0
row
1 C1
1 C2
1 C3
1 C4
column
ElectronicWings.com

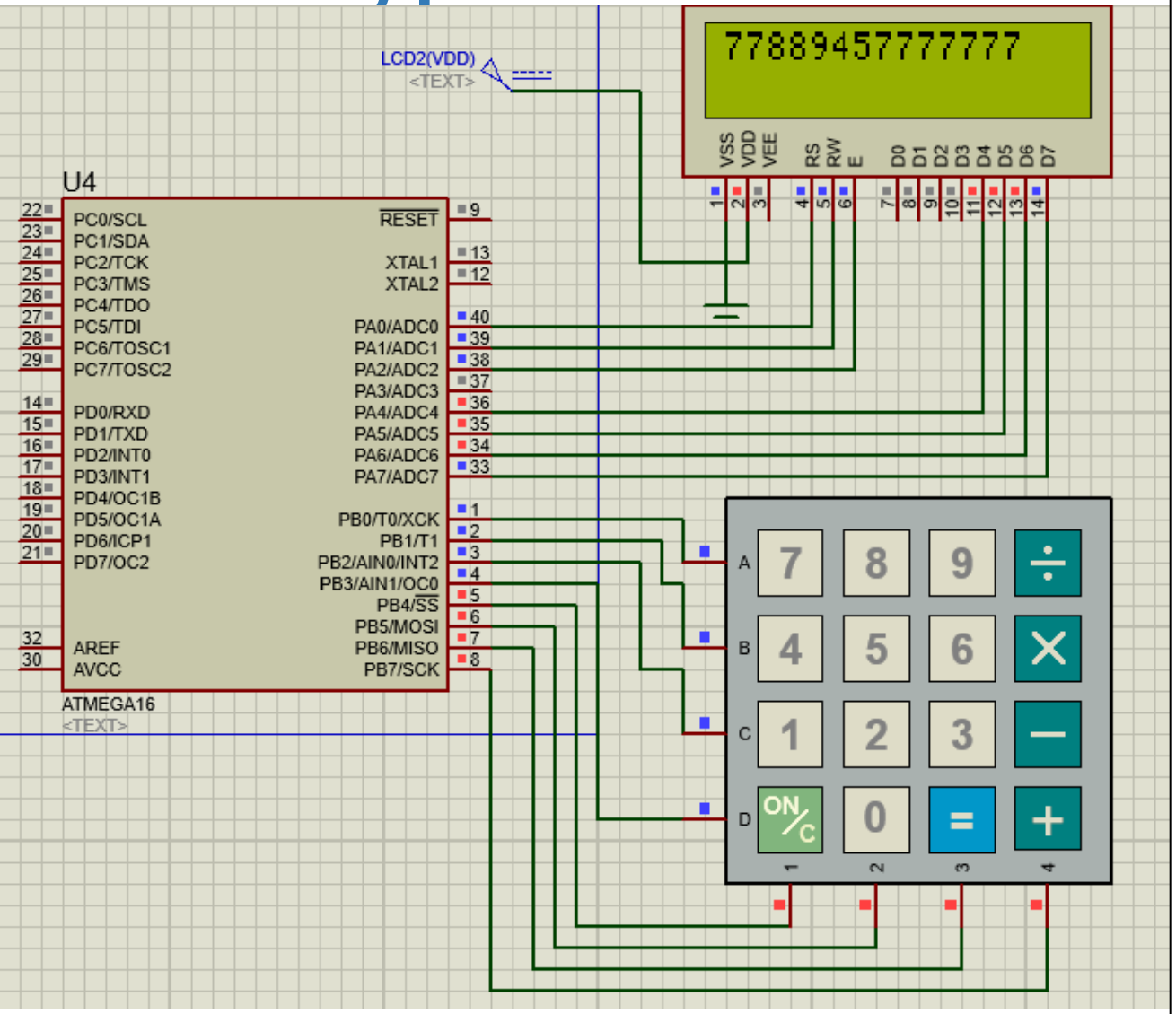

1 2 3 A
4 5 6 B
7 8 9 C
* 0 # D

# 4x4 Keypad


Before pressing key
Vcc(5v)
R1 0
R2 0
R3 0
R4 0
row
1 C1
1 C2
1 C3
1 C4
column
R1 R2 R3 R4 = 0 0 0 0
C1 C2 C3 C4 = 1 1 1 1
ElectronicWings.com


After pressing key
Vcc(5v)
R1 0
R2 0
R3 0
R4 0
row
1 C1
0 C2
1 C3
1 C4
column
R1 R2 R3 R4 = 0 0 0 0
C1 C2 C3 C4 = 1 0 1 1
ElectronicWings.com

# 4x4 Keypad

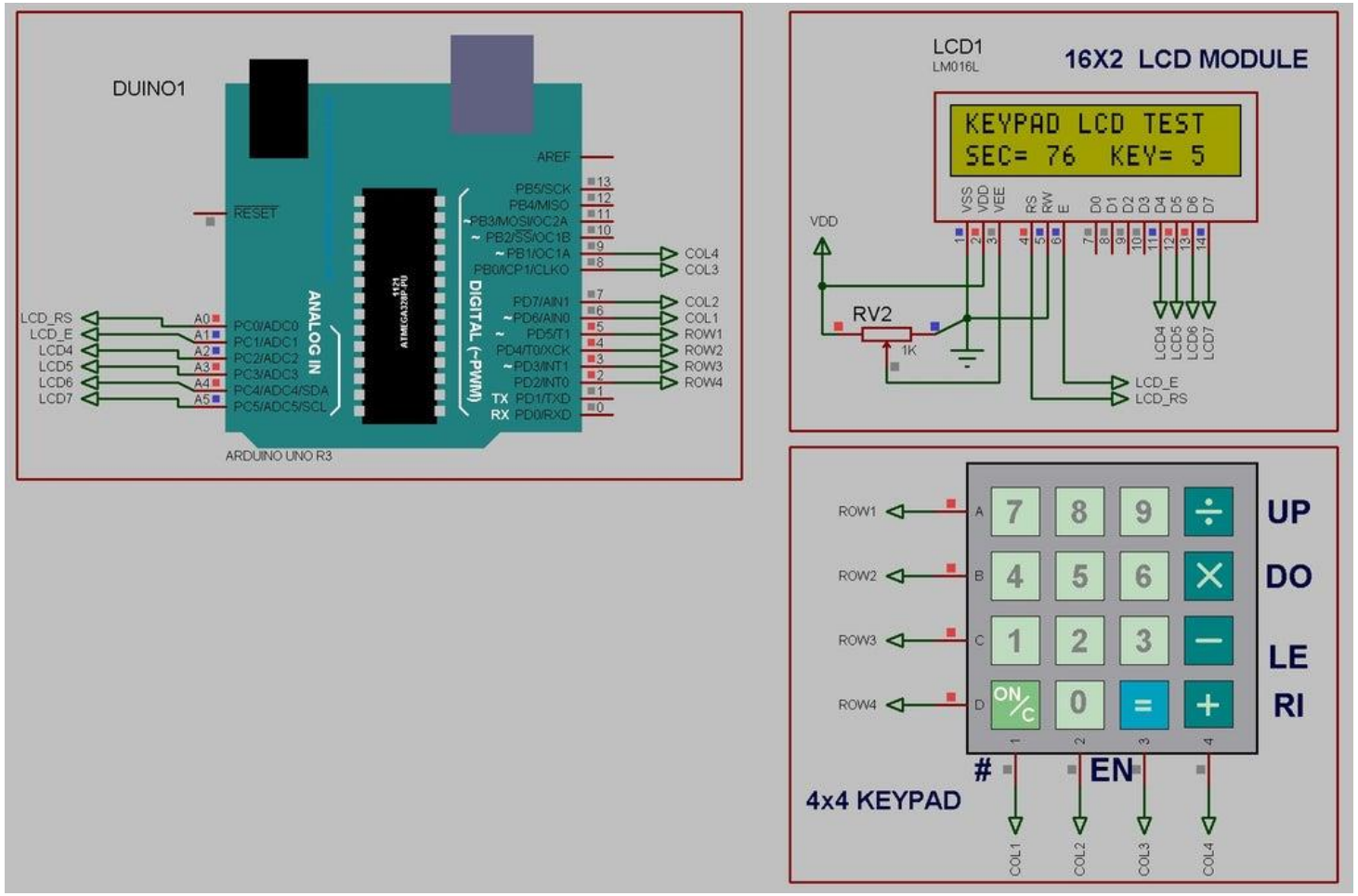



```
#include <mega16.h>
 #include <delay.h>
#include <alcd.h>

#define KEY_PRT          PORTB
#define KEY_DDR          DDRB
#define KEY_PIN          PINB

char keyfind();

unsigned char keypad[4][4] = {      {'7','8','9','/'},
                                    {'4','5','6','*'},
                                     {'1','2','3','-'},
                                     {' ','0','=','+'}};

unsigned char colloc, rowloc;
void main(void)
{
 lcd_init(16);
while (1)
    {
       lcd_putchar(keyfind());
    }
}
```

```
char keyfind()
{
   while(1)
   {
      KEY_DDR = 0x0F;      // set port direction as input-output
      KEY_PRT = 0xF0;
 do
      {
        delay_ms(20);            /* 20ms key debounce time */
        colloc = (KEY_PIN & 0xF0); /* read status of column */
      }while(colloc == 0xF0);       /* check for any key press */

          delay_ms(100);    /* 100ms key debounce time */

     /* now check for rows */
      KEY_PRT = 0xFE;           // check for pressed key in 1st row
      colloc = (KEY_PIN & 0xF0);
      if(colloc != 0xF0)
         {
             rowloc = 0;
             break;
          }
      KEY_PRT = 0xFD;  // check for pressed key in 2nd row

      colloc = (KEY_PIN & 0xF0);
      if(colloc != 0xF0)
      {
          rowloc = 1;
          break;
      }

      KEY_PRT = 0xFB; // check for pressed key in 3rd row

     colloc = (KEY_PIN & 0xF0);
      if(colloc != 0xF0)
      {
          rowloc = 2;
          break;
      }
```

```
KEY_PRT = 0xF7;          /* check for pressed key in 4th row */

        colloc = (KEY_PIN & 0xF0);
        if(colloc != 0xF0)
        {
            rowloc = 3;
            break;
        }
    }

    if(colloc == 0xE0)
      return(keypad[rowloc][0]);
    else if(colloc == 0xD0)
      return(keypad[rowloc][1]);
    else if(colloc == 0xB0)
      return(keypad[rowloc][2]);
    else
      return(keypad[rowloc][3]);
```

# 4x4 Keypad Arduino



# 4x4 Keypad Arduino

```
#include <Keypad.h>
#include <LiquidCrystal.h>

LiquidCrystal lcd(A0,A1,A2,A3,A4,A5);
//4x4 Matrix key pad
const byte ROWS = 4; // Four rows
const byte COLS = 4; // Four columns

 // Define the Keymap
  char keys[ROWS][COLS] =
  {
  {'7','8','9','/'},
  {'4','5','6','X'},
  {'1','2','3','-'},
  {'#','0','=','+'}
  };

 // Connect keypad ROW0, ROW1, ROW2 and ROW3 to
Arduino pins.
  byte rowPins[ROWS] = { 5, 4, 3, 2 };

  // Connect keypad COL0, COL1, COL2 and COL3 to  Arduino
pins.
  byte colPins[COLS] = { 6, 7, 8, 9 };

  // Create the Keypad

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

# 4x4 Keypad Arduino

```
void setup()
{
  // set up the LCD's number of columns and rows:

lcd.begin(16, 2);
  // Print  a message to the LCD.

   lcd.print("KEYPAD LCD TEST");

}
```

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

   // set the cursor to column 0, line 1
  lcd.setCursor(0, 1);

// Check  for a valid key
 if(key)
  {
              lcd.print("KEY= ");

              lcd.print(key);
  }

 }
```

# Thanks