SSG EMBEDDED SOLUTIONS

SSG

info@ssges.co.in

Development Kit of ARDUINO NANO Documentation


List of Contents

     SSG EMBEDDED SOLUTIONS | Ph. No. 7123559635

Introduction

The Arduino Nano is a simple 8-bit microcontroller, ideal for beginners and simple embedded projects. It is programmed via USB and has a number of I/O pins. It uses the Atmel ATmega 328P microprocessor chip. Arduino NANO is the open-source microcontroller development board based on the ATMEGA328P microcontroller IC. The microcontroller IC on which the Arduino UNO and Arduino NANO is based is usually the same by the way sometimes the difference lies in the package type of the microcontroller IC. Having same microcontroller IC it follows that the crucial specifications of both the Arduino UNO and Arduino NANO are essentially the same. The Arduino NANO is sometimes preferred over the Arduino UNO when there is limitation on the space constraint. Arduino NANO is quite small in size as compared to the Arduino UNO and can easily be mounted on the Breadboard making it useful in Breadboard based prototypes. Arduino NANO has 14 Digital Input / Output pins and 8 analog pins. The Arduino NANO has two additional Analog to Digital converters as compare to the Arduino UNO so that NANO has two additional Analog pins. Arduino NANO has one UART, one Inter-Integrated Circuit (I2C) computer bus and one Serial Peripheral Interface (SPI) computer bus. Arduino UNO also has one UART, one SPI and one I2C interface on board. Out of the 14 digital input / output pins 5 pins are PWM (Pulse Width Modulation) enabled. The discussion on the PWM phenomenon and the peculiar use of these PWM enabled pins will be discussed later in the posts. Some differences that exist between the Arduino UNO and Arduino NANO will be pointed out later in the post.
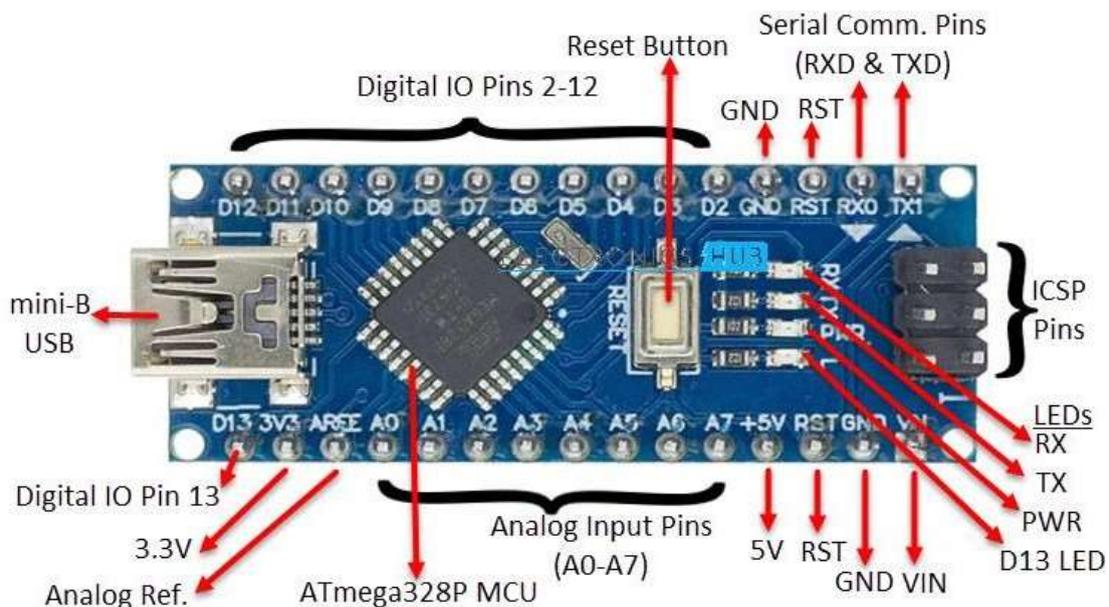
The microcontroller using in Nano board (ATmega328) offers <u>serial communication</u> (UART TTL). This can be accessible at digital pins like TX, and RX. The

Arduino software comprises of a serial monitor to allow easy textual information to transmit and receive from the board.

The TX & RX LEDs on the Nano board will blink whenever information is being sent out through the FTDI & USB link in the direction of the computer. The library-like SoftwareSerial allows serial communication on any of the digital pins on the board. The microcontroller also supports SPI & I2C (TWI) communication.

Pin Diagram
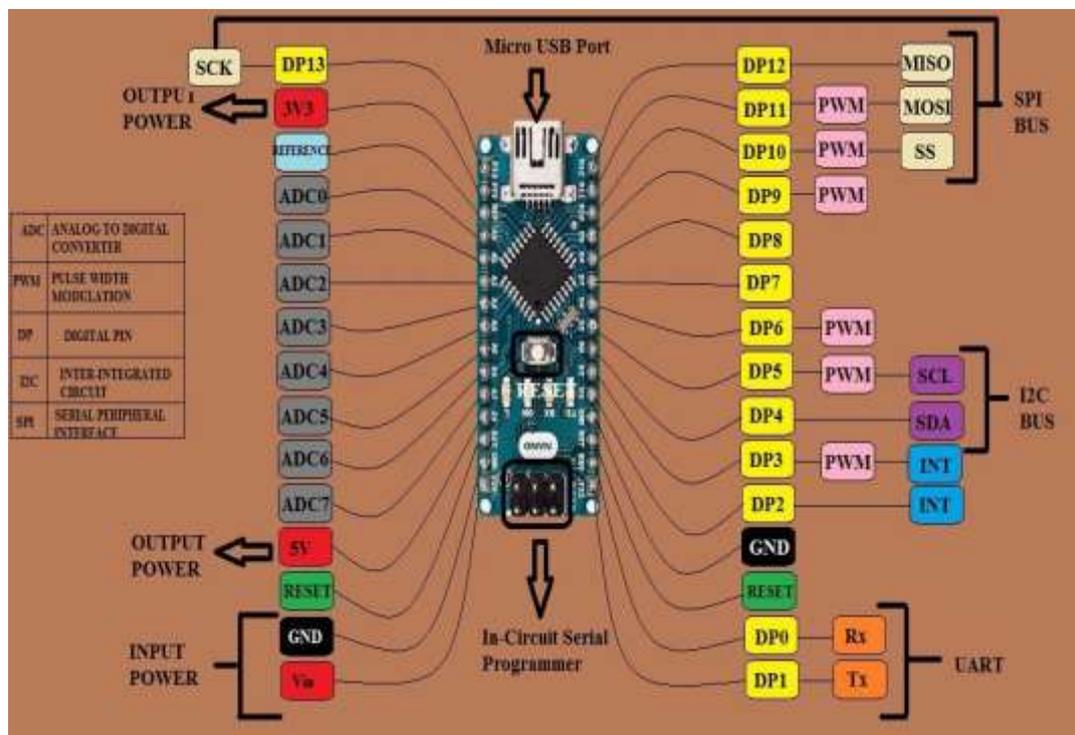
Specifications

- ATmega328P Microcontroller is from 8-bit AVR family
- Operating voltage is 5V
- Input voltage (Vin) is 7V to 12V
- Input/Output Pins are 22
- Analog i/p pins are 6 from A0 to A5
- Digital pins are 14
- Power consumption is 19 mA
- I/O pins DC Current is 40 mA
- Flash memory is 32 KB
- SRAM is 2 KB
- EEPROM is 1 KB
- CLK speed is 16 MHz
- Weight-7g
- Size of the printed circuit board is 18 X 45mm
- Supports three communications like SPI, IIC, & USART

Application

These boards are used to build Arduino Nano projects by reading inputs of a sensor, a button, or a finger and gives an output by turning motor or LED ON, or and some of the applications are listed below.

- Samples of electronic systems & products
- Automation
- Several DIY projects
- Control Systems
- Embedded Systems
- Robotics
- Instrumentation

Material Required

- NANO board
- Cable

How to Install the Arduino IDE


Step 1: Download File Arduino IDE

Step 2: License Agreement
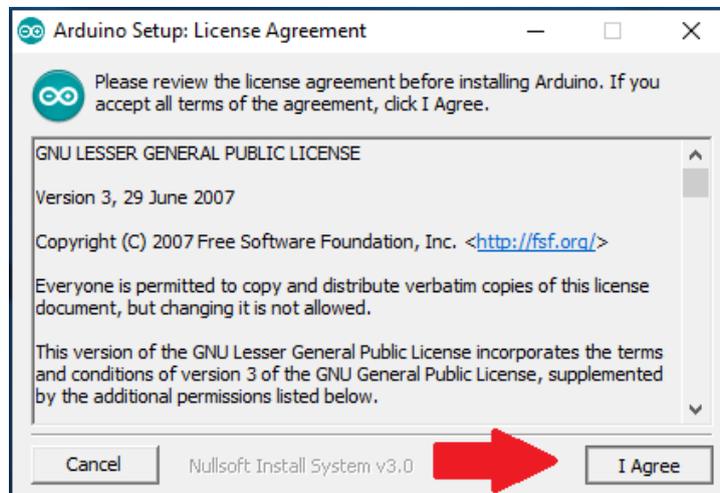
After the file is run, the "License Agreement" page will apper. You can read it, then click "*I Agree*" to continue.

Step 3: Installation Option
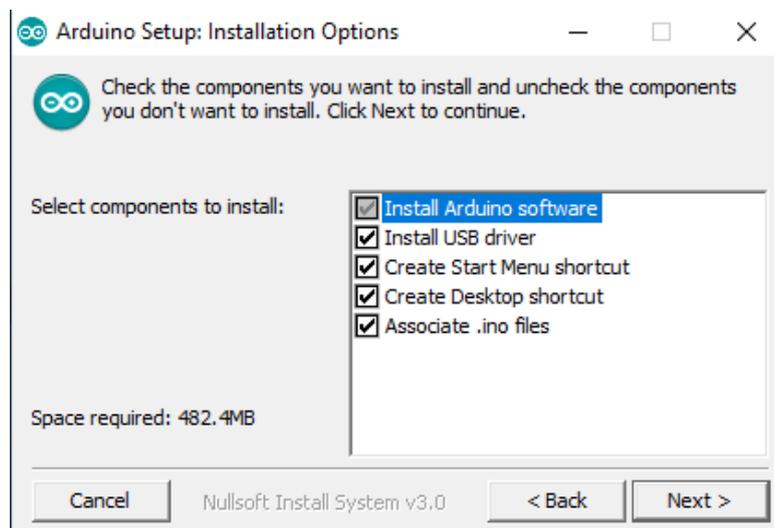
Check the component that you want to install and uncheck the components that you don't want to install. I suggest installing all componen. Click "*next*" to continue.

Step 4: Installation Folder
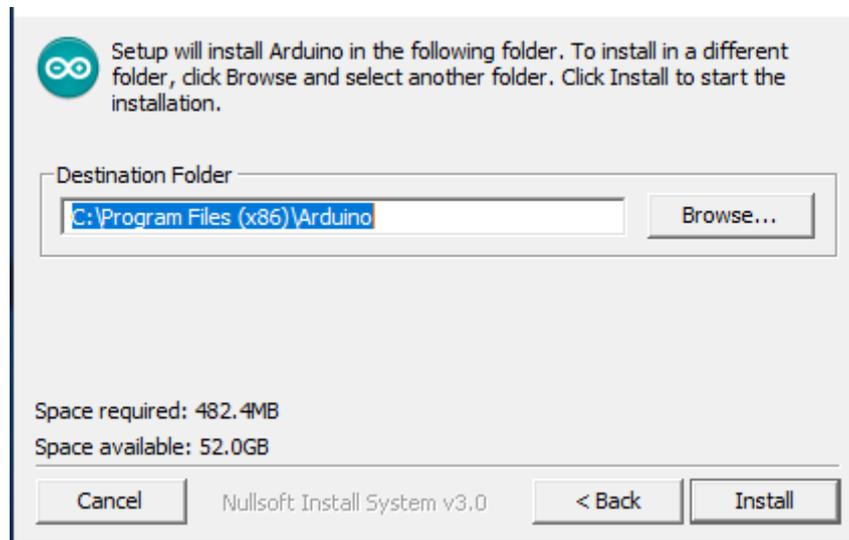
Arduino will automatically be installed in "**C:\Program Files (x86)\Arduino**". If you want to change the folder, click "*Browse*" and select the desired folder. Click install to start the installation.

Step 5: Installing Proses
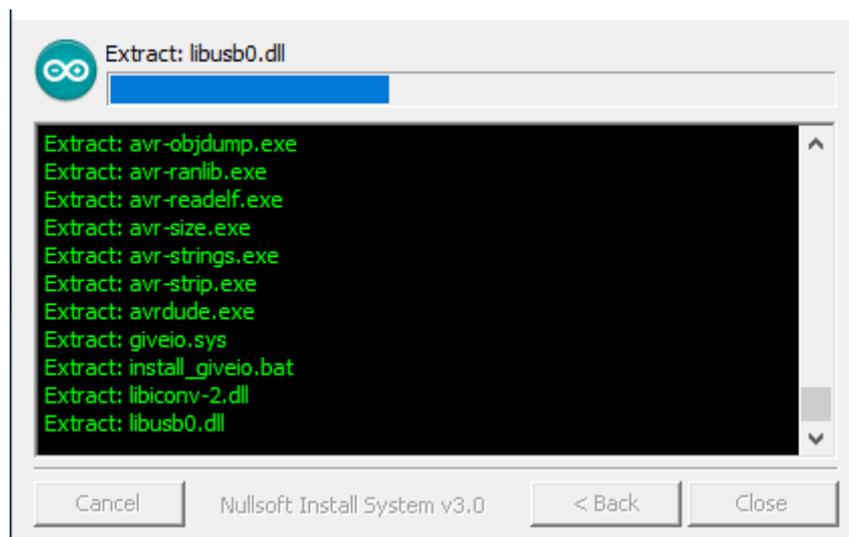
The installation process is ongoing.

SSG EMBEDDED SOLUTIONS | Ph. No. 7123559635

Step 6: Installation Complete
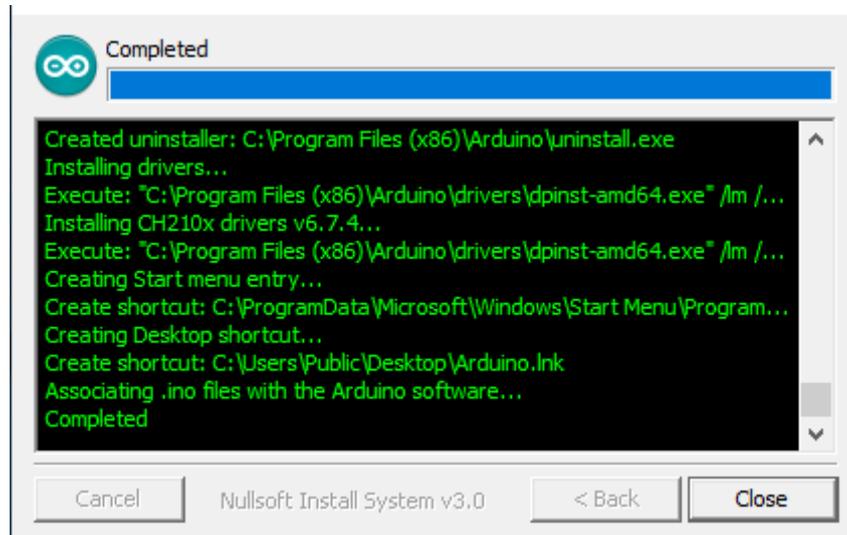
If there is written "*complete*", it means that the isntallation process is complete. click "*Close*".

Step 7: Open Arduino IDE
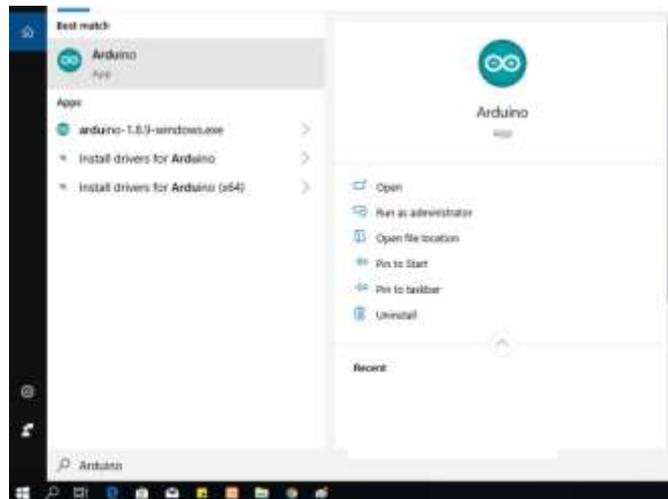
After the installation process is complete, there will be an Arduino icon on the Desktop. Or check on the search icon and write "arduino". If you have found the arduino icon, run the application.

Step 8: Display Arduino IDE

This is a display of the Arduino IDE Software. The application is ready to be used to create amazing projects.

This is a display of the Arduino IDE Software. The application is ready to be used to create amazing projects.

Step 9: Select the Board That Is Used

SSG EMBEDDED SOLUTIONS | Ph. No. 7123559635
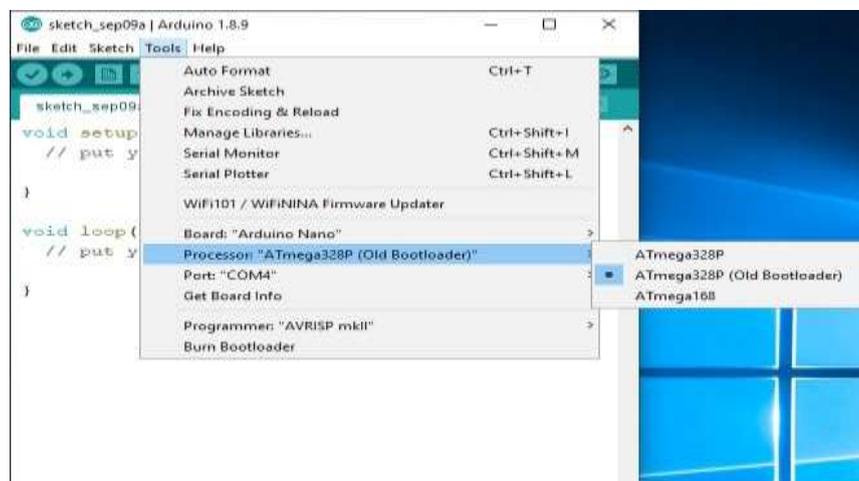
Step 4: Open and Upload Sketch
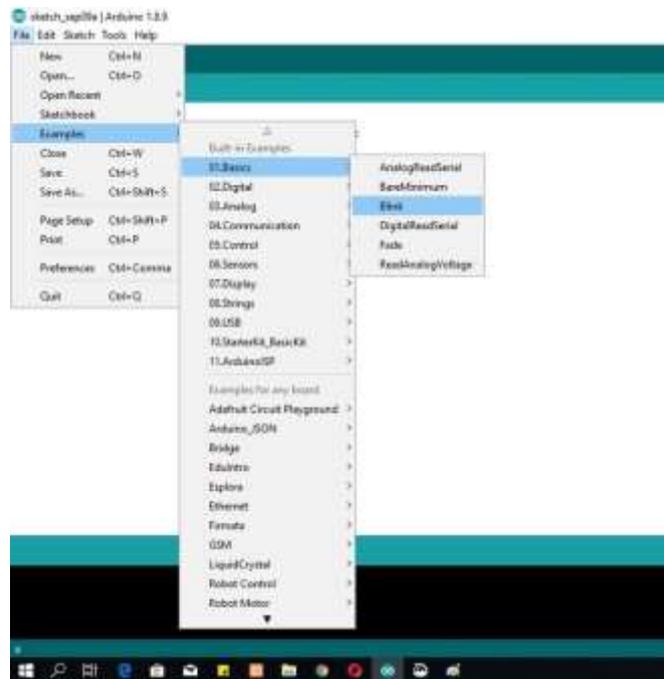
Open the LED blink example sketch: **File > Examples > 01.Basics > Blink**.

To upload the program. Click the upload button. Wait for a moment - During the upload process, the RX and TX LEDs will be flashing. If the upload is successful, the message "**Done uploading**" will appear in the status bar.

LED

It is most widely used semiconductor which emit either visible light or invisible infrared light when forward biased. Remote controls generate invisible light. A Light emitting diodes (LED) is an optical electrical energy into light energy when voltage is applied.
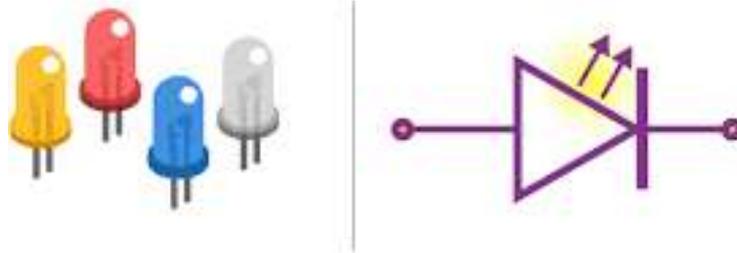
These are the applications of LEDs:

- Digital computers and calculators.
- Traffic signals and Burglar alarms systems.
- Camera flashes and automotive heat lamps
- Picture phones and digital watches.

LED Interfacing with Arduino Nano
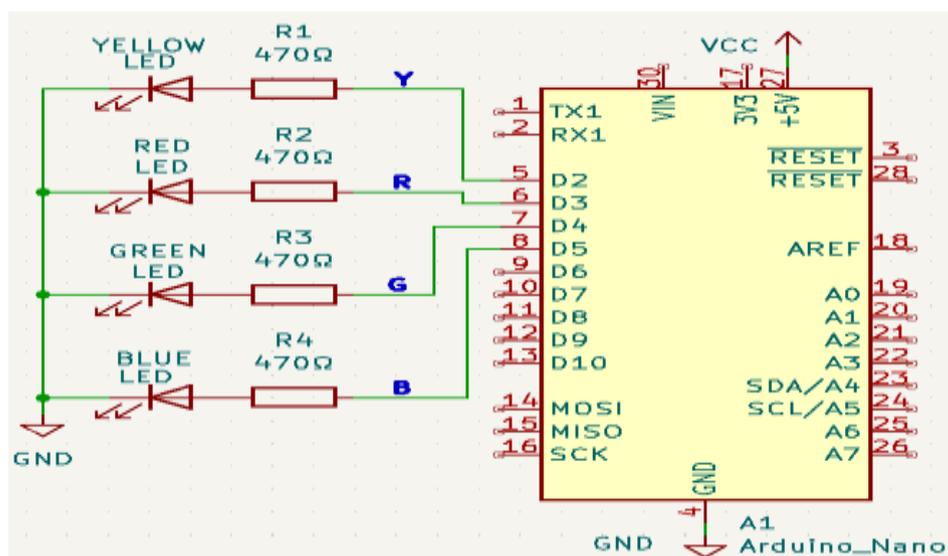
LED Blinking Code

```
//connect D2,D3,D4 & D5 WITH blue,green,red & yellow leds respectively
void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
}
void loop()
{
  digitalWrite(2, HIGH);
  delay(100);
  digitalWrite(2, LOW);
  delay(100);

  digitalWrite(3, HIGH);
  delay(100);
  digitalWrite(3, LOW);
  delay(100);

  digitalWrite(4, HIGH);
  delay(100);
  digitalWrite(4, LOW);
  delay(100);

  digitalWrite(5, HIGH);
```

```
delay(100);

digitalWrite(5, LOW);

delay(100);

}
```


Switch & LED Interfacing with Arduino Nano
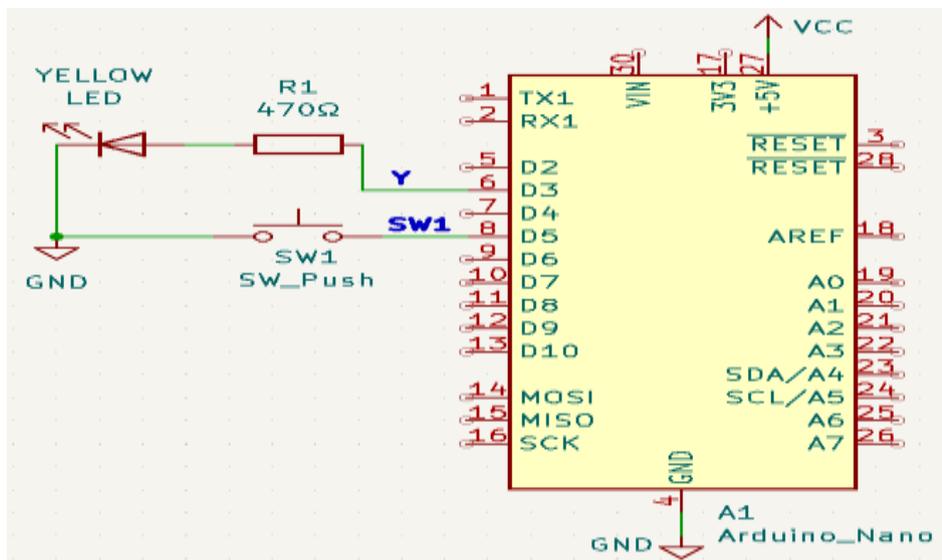
LED with Switch Code

```
//CONNECT D5 TO SWITCH & D3 TO LED PIN

const int LEDPIN1 = 2;

const int LEDPIN2 = 3;

const int LEDPIN3 = 4;

const int LEDPIN4 = 5;

const int PushButton1 =17;

const int PushButton2 =16;

const int PushButton3 =15;

const int PushButton4 =14;


// This Setup function is used to initialize everything

void setup()

{

// This statement will declare pin 5 as digital output

pinMode(LEDPIN1, OUTPUT);

pinMode(LEDPIN2, OUTPUT);

pinMode(LEDPIN3, OUTPUT);

pinMode(LEDPIN4, OUTPUT);


// This statement will declare pin 2 as digital input

pinMode(PushButton1, INPUT_PULLUP);

pinMode(PushButton2, INPUT_PULLUP);

pinMode(PushButton3, INPUT_PULLUP);

pinMode(PushButton4, INPUT_PULLUP);

}
```

```
void loop()

{

int Push_button_state1 = digitalRead(PushButton1);

int Push_button_state2 = digitalRead(PushButton2);

int Push_button_state3 = digitalRead(PushButton3);

int Push_button_state4 = digitalRead(PushButton4);

if ( Push_button_state1 == HIGH )

{

digitalWrite(LEDPIN1, HIGH);

}

else

{

  digitalWrite(LEDPIN1, LOW);

}

if ( Push_button_state2 == HIGH )

{

digitalWrite(LEDPIN2, HIGH);

}

else

{

  digitalWrite(LEDPIN2, LOW);

}

if ( Push_button_state3 == HIGH )

{

digitalWrite(LEDPIN3, HIGH);

}

else

{
```

```
  digitalWrite(LEDPIN3, LOW);

}

if ( Push_button_state4 == HIGH )

{

digitalWrite(LEDPIN4, HIGH);

}

else

{

  digitalWrite(LEDPIN4, LOW);

}

}
```
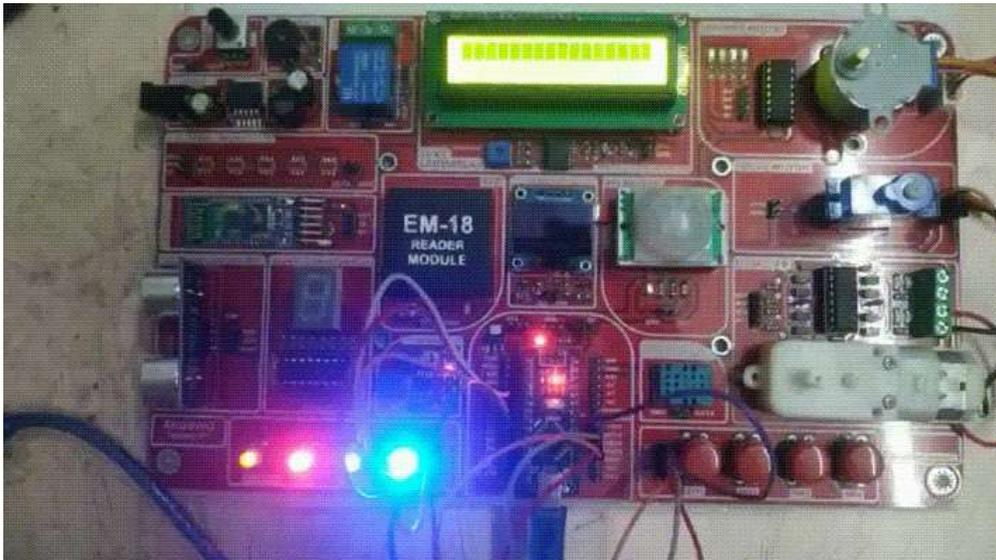
OLED

OLED is the acronym for **Organic Light Emitting Diode**. OLED is a modern display technology used in a wide range of electronic display devices, such as TVs, monitors, laptops, smartphones, bulletin boards, stadium screens, etc.
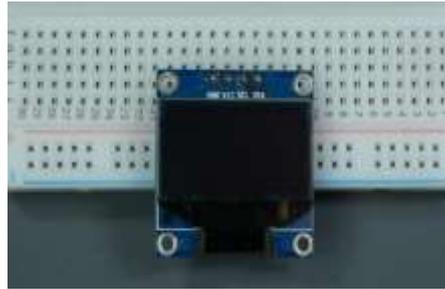
OLED displays consist of organic semiconductor compounds that emit a bright light on the passage of electric current through them, and hence it is termed as OLED. Since, OLED displays can emit light on their own, thus they are considered as self-emissive types of display. There is no need of backlight panel with LEDs to illuminate the screen.

The primary advantages of OLED displays include better picture quality, relatively wider viewing angles, greater flexibility in design, compact size, faster response time, and low power consumption.

OLED Display Interfacing with Arduino Nano
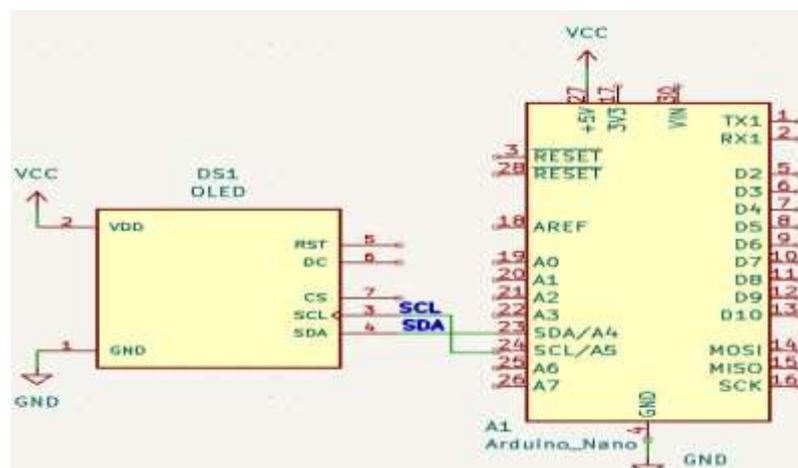
OLED Code

```
//connect SDA & SCL pin of NANO board with SDA & SCL of OLED Display

#include <SPI.h>

#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>


#define SCREEN_WIDTH 128 // OLED display width, in pixels

#define SCREEN_HEIGHT 64 // OLED display height, in pixels


// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)

// The pins for I2C are defined by the Wire-library.

// On an arduino UNO:       A4(SDA), A5(SCL)

// On an arduino MEGA 2560: 20(SDA), 21(SCL)

// On an arduino LEONARDO:  2(SDA),  3(SCL), ...

#define OLED_RESET     -1 // Reset pin # (or -1 if sharing Arduino reset pin)

#define SCREEN_ADDRESS 0x3C ///< See datasheet for Address; 0x3D for 128x64, 0x3C for 128x32

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);


#define NUMFLAKES     10 // Number of snowflakes in the animation example

#define LOGO_HEIGHT   16

#define LOGO_WIDTH    16

static const unsigned char PROGMEM logo_bmp[] =

{ 0b00000000, 0b11000000,

  0b00000001, 0b11000000,

  0b00000001, 0b11000000,

  0b00000011, 0b11100000,
```

```
0b11110011, 0b11100000,

0b11111110, 0b11111000,

0b01111110, 0b11111111,

0b00110011, 0b10011111,

0b00011111, 0b11111100,

0b00001101, 0b01110000,

0b00011011, 0b10100000,

0b00111111, 0b11100000,

0b00111111, 0b11110000,

0b01111100, 0b11110000,

0b01110000, 0b01110000,

0b00000000, 0b00110000 };


void setup() {

  Serial.begin(9600);


  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally

  if(!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {

    Serial.println(F("SSD1306 allocation failed"));

    for(;;); // Don't proceed, loop forever

  }

  // Show initial display buffer contents on the screen --

  // the library initializes this with an Adafruit splash screen.

  display.display();

  delay(2000); // Pause for 2 seconds


  // Clear the buffer

  display.clearDisplay();
```

```
// Draw a single pixel in white

display.drawPixel(10, 10, SSD1306_WHITE);


// Show the display buffer on the screen. You MUST call display() after

// drawing commands to make them visible on screen!

display.display();

delay(2000);

// display.display() is NOT necessary after every single drawing command,

// unless that's what you want...rather, you can batch up a bunch of

// drawing operations and then update the screen all at once by calling

// display.display(). These examples demonstrate both approaches...

testdrawline();     // Draw many lines

testdrawrect();     // Draw rectangles (outlines)

testfillrect();     // Draw rectangles (filled)

testdrawcircle();   // Draw circles (outlines)

testfillcircle();   // Draw circles (filled)

testdrawroundrect(); // Draw rounded rectangles (outlines)

testfillroundrect(); // Draw rounded rectangles (filled)

testdrawtriangle();  // Draw triangles (outlines)

testfilltriangle();  // Draw triangles (filled)

testdrawchar();     // Draw characters of the default font

testdrawstyles();   // Draw 'stylized' characters

// Invert and restore display, pausing in-between

display.invertDisplay(true);

delay(1000);

display.invertDisplay(false);

delay(1000);
```

```
}
void loop() {

}
void testdrawline() {
  int16_t i;

  display.clearDisplay(); // Clear display buffer
  for(i=0; i<display.width(); i+=4) {
    display.drawLine(0, 0, i, display.height()-1, SSD1306_WHITE);
    display.display(); // Update screen with each newly-drawn line
    delay(1);
  }
  for(i=0; i<display.height(); i+=4) {
    display.drawLine(0, 0, display.width()-1, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  delay(250);
  display.clearDisplay();

  for(i=0; i<display.width(); i+=4) {
    display.drawLine(0, display.height()-1, i, 0, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  for(i=display.height()-1; i>=0; i-=4) {
    display.drawLine(0, display.height()-1, display.width()-1, i, SSD1306_WHITE);
    display.display();
```

```
    delay(1);
  }
  delay(250);
  display.clearDisplay();


  for(i=display.width()-1; i>=0; i-=4) {
    display.drawLine(display.width()-1, display.height()-1, i, 0, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  for(i=display.height()-1; i>=0; i-=4) {
    display.drawLine(display.width()-1, display.height()-1, 0, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  delay(250);
  display.clearDisplay();


  for(i=0; i<display.height(); i+=4) {
    display.drawLine(display.width()-1, 0, 0, i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  for(i=0; i<display.width(); i+=4) {
    display.drawLine(display.width()-1, 0, i, display.height()-1, SSD1306_WHITE);
    display.display();
    delay(1);
  }
```

```
  delay(2000); // Pause for 2 seconds
}
void testdrawrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=2) {
    display.drawRect(i, i, display.width()-2*i, display.height()-2*i, SSD1306_WHITE);
    display.display(); // Update screen with each newly-drawn rectangle
    delay(1);
  }
  delay(2000);
}
void testfillrect(void) {
  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2; i+=3) {
    // The INVERSE color is used so rectangles alternate white/black
    display.fillRect(i, i, display.width()-i*2, display.height()-i*2, SSD1306_INVERSE);
    display.display(); // Update screen with each newly-drawn rectangle
    delay(1);
  }
  delay(2000);
}
void testdrawcircle(void) {
  display.clearDisplay();

  for(int16_t i=0; i<max(display.width(),display.height())/2; i+=2) {
    display.drawCircle(display.width()/2, display.height()/2, i, SSD1306_WHITE);
```

```
    display.display();

    delay(1);


    delay(2000);

}

void testfillcircle(void) {

  display.clearDisplay();


  for(int16_t i=max(display.width(),display.height())/2; i>0; i-=3) {

    // The INVERSE color is used so circles alternate white/black

    display.fillCircle(display.width() / 2, display.height() / 2, i, SSD1306_INVERSE);

    display.display(); // Update screen with each newly-drawn circle

    delay(1);

  }

  delay(2000);

}

void testdrawroundrect(void) {

  display.clearDisplay();

  for(int16_t i=0; i<display.height()/2-2; i+=2) {

    display.drawRoundRect(i, i, display.width()-2*i, display.height()-2*i,

      display.height()/4, SSD1306_WHITE);

    display.display();

    delay(1);

  }

  delay(2000);

}

void testfillroundrect(void) {

  display.clearDisplay();
```

```cpp
  for(int16_t i=0; i<display.height()/2-2; i+=2) {
    // The INVERSE color is used so round-rects alternate white/black
    display.fillRoundRect(i, i, display.width()-2*i, display.height()-2*i,
    display.height()/4, SSD1306_INVERSE);
    display.display();
    delay(1);
  }
  delay(2000);
}
void testdrawtriangle(void) {
  display.clearDisplay();
  for(int16_t i=0; i<max(display.width(),display.height())/2; i+=5) {
    display.drawTriangle(
      display.width()/2 , display.height()/2-i,
      display.width()/2-i, display.height()/2+i,
      display.width()/2+i, display.height()/2+i, SSD1306_WHITE);
    display.display();
    delay(1);
  }
  delay(2000);
}
void testfilltriangle(void) {
  display.clearDisplay();

  for(int16_t i=max(display.width(),display.height())/2; i>0; i-=5) {
    // The INVERSE color is used so triangles alternate white/black
    display.fillTriangle(
      display.width()/2 , display.height()/2-i,
```

```
      display.width()/2-i, display.height()/2+i,
      display.width()/2+i, display.height()/2+i, SSD1306_INVERSE);
    display.display();
    delay(1);
  }
  delay(2000);
}
void testdrawchar(void) {
  display.clearDisplay();
  display.setTextSize(1);      // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE); // Draw white text
  display.setCursor(0, 0);     // Start at top-left corner
  display.cp437(true);         // Use full 256 char 'Code Page 437' font
  // Not all the characters will fit on the display. This is normal.
  // Library will draw what it can and the rest will be clipped.
  for(int16_t i=0; i<256; i++) {
    if(i == '\n') display.write(' ');
    else          display.write(i);
  }
  display.display();
  delay(2000);
}
void testdrawstyles(void) {
  display.clearDisplay();
  display.setTextSize(1);           // Normal 1:1 pixel scale
  display.setTextColor(SSD1306_WHITE);      // Draw white text
  display.setCursor(0,0);           // Start at top-left corner
  display.println(F("SSG EMBEDDED SOLUTIONS"));
```

```
display.display();

delay(2000);

}
```

DHT11

The DHT11 and DHT22 sensors are used to measure temperature and relative humidity. These sensors contain a chip that does analog to digital conversion and spit out a digital signal with the temperature and humidity. This makes them very easy to use with any microcontroller.
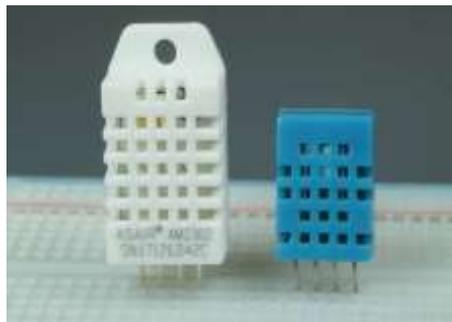
The DHT22 sensor has a better resolution and a wider temperature and humidity measurement range. However, it is a bit more expensive, and you can only request readings with 2 seconds interval. The DHT33 has a smaller range and it's less accurate. However, you can request sensor readings every second. It's also a bit cheaper.

DHT11 Interfacing with Arduino Nano
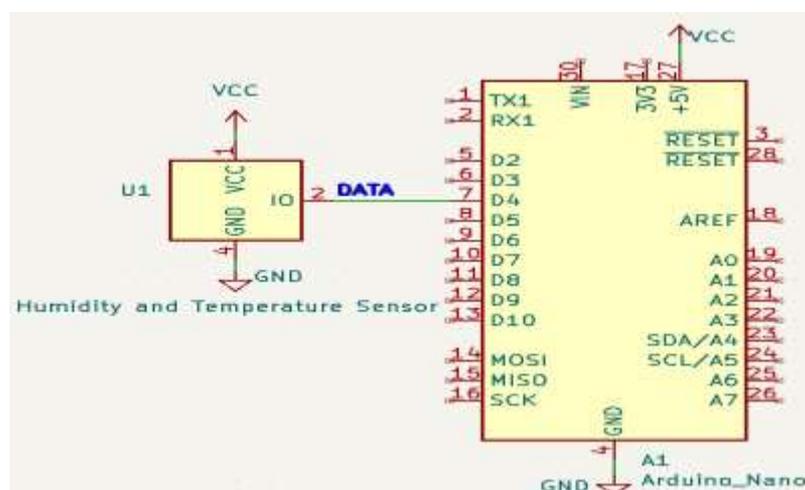
DHT11 Code

```
//D4 TO DHT11 PIN

#include <dht.h>  // Include DHT11 library

#define DATApin 4  // Defines Arduino pin which is connected to the sensor


dht DHT;     // Creates a DHT object


void setup()
 {
  //Sets the baud for serial data transmission between Arduino and your computer
  Serial.begin(9600);
 }


void loop()
 {
  // Read data from Sensor
  int readData = DHT.read11(DATApin);


  float TC = DHT.temperature;  // Read Temperature in Degree Celsius unit
  float TF = ((TC*9.0)/5.0+32.0); // Convert Celsius to Fahrenheit unit


  float h = DHT.humidity;  // Read humidity


  //Print Tempareture Value on Serial Monitor Window
  Serial.print("Temperature = ");
  Serial.print(TC); // Temperature value in Degree Celsius
  Serial.print("°C | ");
```

```
Serial.print(TF); // Temperature value in Fahrenheit
Serial.println("°F ");


//Print Humidity Value on Serial Monitor Window
Serial.print("Humidity = ");
Serial.print(h);
Serial.println("% ");
Serial.println("");


delay(2000); // wait two seconds
}
```

Relay

Relays are the switches that aim at closing and opening the circuits electronically as well as electromechanically. It controls the opening and closing of the circuit contacts of an electronic circuit. When the relay contact is open (NO), the relay isn't energized with the open contact. However, if it is closed (NC), the relay isn't energized given the closed contact. However, when energy (electricity or charge) is supplied, the states are prone to change.
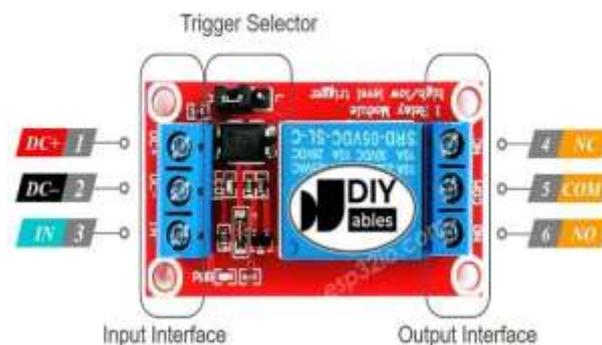
Relays are normally used in the control panels, manufacturing, and building automation to control the power along with switching the smaller current values in a control circuit. However, the supply of amplifying effect can help control the large amperes and voltages because if low voltage is applied to the relay coil, a large voltage can be switched by the contacts.

Relay Interfacing with Arduino Nano
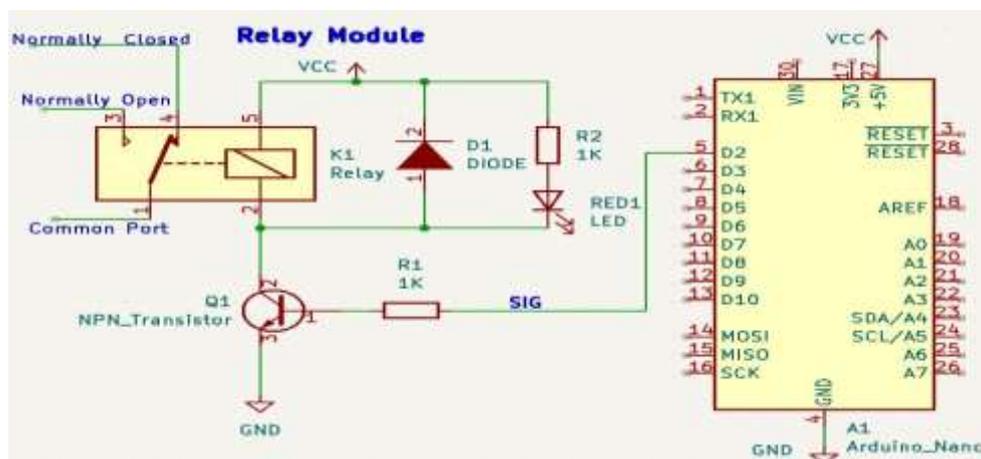
Relay Code

```
//connect D2 pin to Relay Signal pin

void setup()

{

  pinMode(2, OUTPUT);


}


void loop()

{

  digitalWrite(2, HIGH);

  delay(500);

  digitalWrite(2, LOW);

  delay(500);


}
```
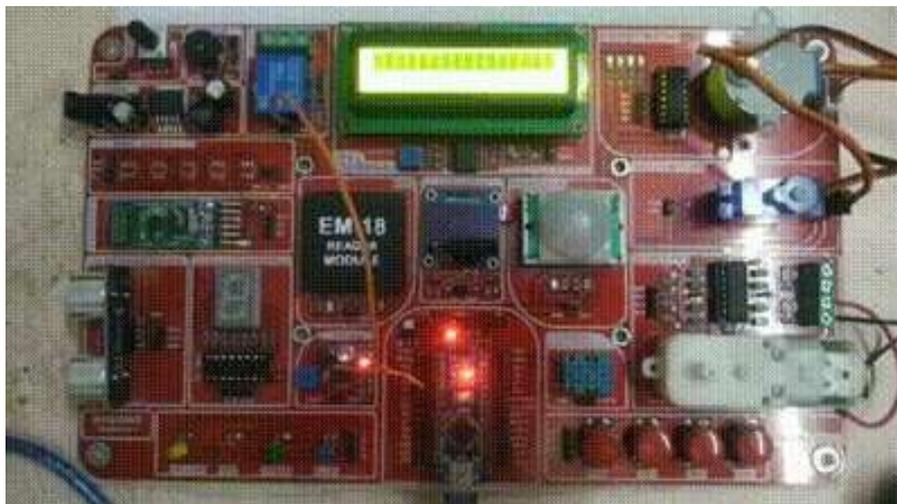
LCD

The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes and seven segments. The main benefits of using this module are inexpensive; simply programmable, animations, and there are no limitations for displaying custom characters, special and even animations, etc.

The features of this LCD mainly include the following.

- The operating voltage of this LCD is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8 pixel box
- The alphanumeric LCDs alphabets & numbers
- Is display can work on two modes like 4-bit & 8-bit

LCD Display Interfacing with Arduino Nano
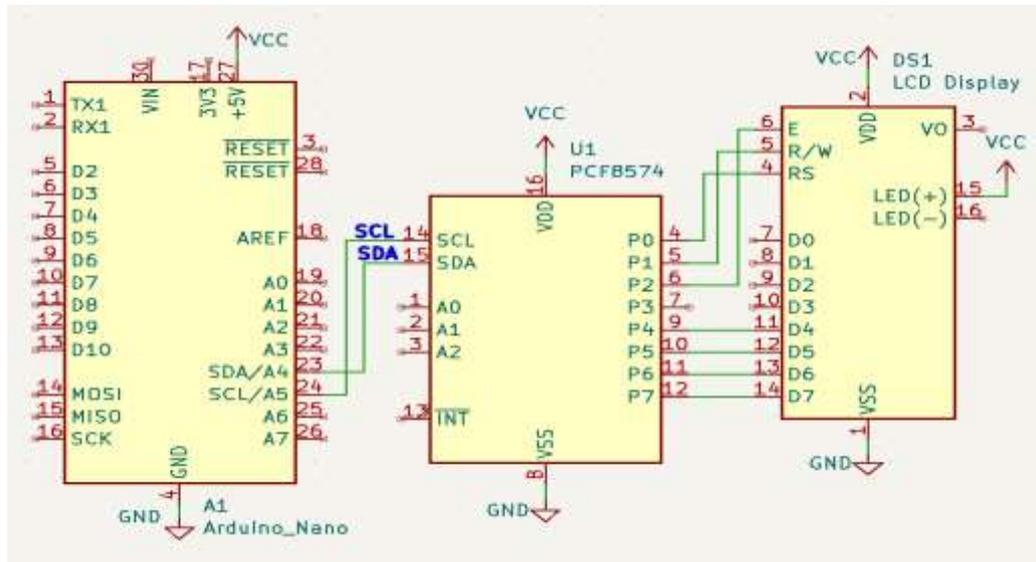
LCD Code

//Connect SDA & SCL of NANO board with SDA & SCL of LCD Display

```
#include <LiquidCrystal_PCF8574.h>

#include <Wire.h>

LiquidCrystal_PCF8574 lcd(0x27); // set the LCD address to 0x27 for a 16 chars and 2 line
display

int show = -1;

void setup()

{

  int error;

  Serial.begin(115200);

  Serial.println("LCD...");

  // wait on Serial to be available on Leonardo

  while (!Serial);

  Serial.println("Dose: check for LCD");
```

```
// See http://playground.arduino.cc/Main/I2cScanner how to test for a I2C device.
Wire.begin();
Wire.beginTransmission(0x27);
error = Wire.endTransmission();
Serial.print("Error: ");
Serial.print(error);
if (error == 0) {
  Serial.println(": LCD found.");
  show = 0;
  lcd.begin(16, 2); // initialize the lcd
} else {
  Serial.println(": LCD not found.");
} // if
} // setup()
void loop()
{
  if (show == 0) {
    lcd.setBacklight(255);
    lcd.home();
    lcd.clear();
    lcd.print("SSG LCD");
    delay(1000);
    lcd.setBacklight(0);
    delay(400);
    lcd.setBacklight(255);
}
else if (show == 1) {
    lcd.clear();
```

```
    lcd.print("Cursor On");

    lcd.cursor();


  } else if (show == 2) {

   lcd.clear();

   lcd.print("Cursor Blink");

   lcd.blink();


  } else if (show == 3) {

   lcd.clear();

   lcd.print("Cursor OFF");

   lcd.noBlink();

   lcd.noCursor();


  } else if (show == 4) {

   lcd.clear();

   lcd.print("Display Off");

   lcd.noDisplay();


  } else if (show == 5) {

   lcd.clear();

   lcd.print("Display On");

   lcd.display();


  } else if (show == 7) {

   lcd.clear();

   lcd.setCursor(0, 0);

   lcd.print("*** first line.");
```

```
    lcd.setCursor(0, 1);

    lcd.print("*** second line.");

  } else if (show == 8) {

    lcd.scrollDisplayLeft();

  } else if (show == 9) {

    lcd.scrollDisplayLeft();

  } else if (show == 10) {

    lcd.scrollDisplayLeft();

  } else if (show == 11) {

    lcd.scrollDisplayRight();


  } else if (show == 12) {

    lcd.clear();

    lcd.print("write-");


  } else if (show > 12) {

    lcd.print(show - 13);

  } // if

  delay(1400);

  show = (show + 1) % 16;

} // loop()
```

Buzzer

A **buzzer** is an electronic device that generates sound by converting electrical energy into sound energy. It typically consists of a piezoelectric crystal, which expands and contracts when an alternating current is applied to it, creating sound waves.
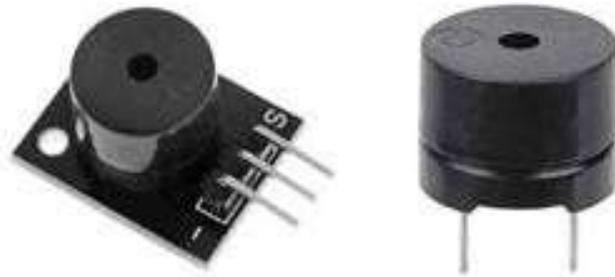
Buzzers are commonly used in a wide range of applications such as alarms, timers, and warning systems. They can also be used in electronic devices such as mobile phones, computers, and other electronic devices to generate different sounds and tones.

## Buzzer Interfacing with Arduino Nano
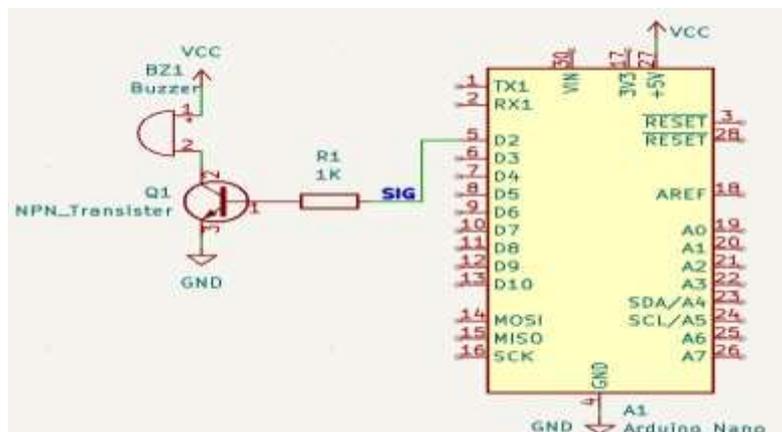
Buzzer Code

```
//connect D2 pin to Buzzer Signal pin

void setup()

{

  pinMode(2, OUTPUT);


}

void loop()

{

  digitalWrite(2, HIGH);

  delay(500);

  digitalWrite(2, LOW);

  delay(500);

}
```

DC Motor

The working principle of DC motor is based on the fact that when a current carrying conductor is placed in a magnetic field, it experiences a mechanical force and starts rotating. Its direction of rotation depends upon Fleming's Left Hand Rule.

DC motors are used in many applications like robot for movement control, toys, quadcopters, CD/DVD disk drive in PCs/Laptops etc.

DC Motor Interfacing with Arduino Nano
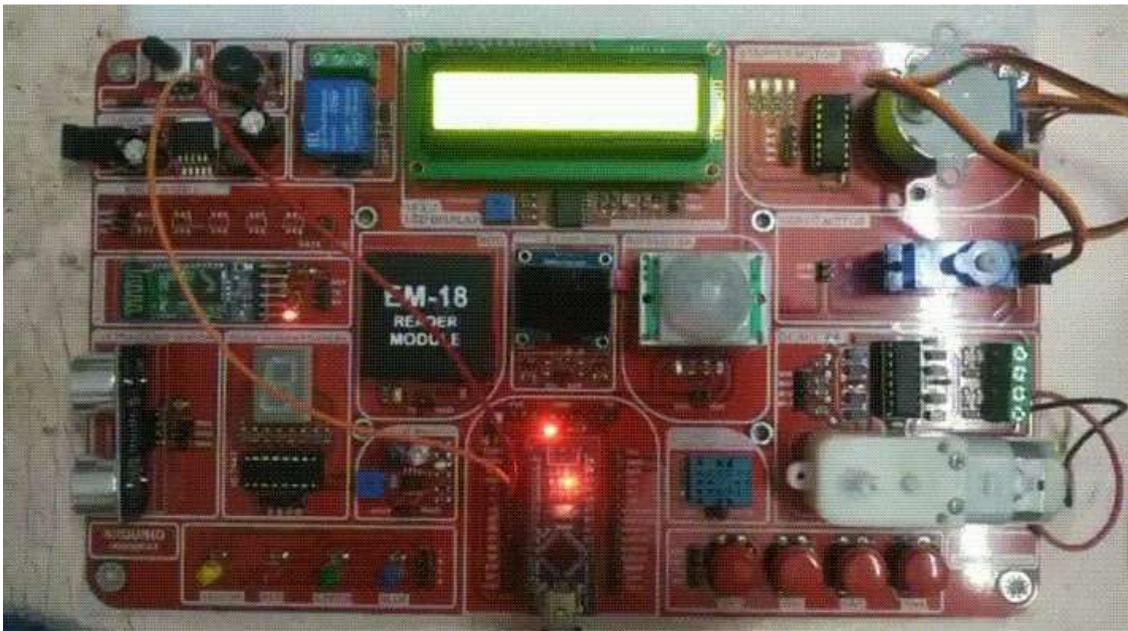
DC Motor Code

```
//connect D2,D3,D4 & D5 with M1A,M1B,M2A & M2B
byte M1A = 2;
byte M1B = 3;
void setup()
{
  pinMode(M1A, OUTPUT);
  pinMode(M1B, OUTPUT);
}
void loop()
{
  digitalWrite(M1A,  LOW);
  digitalWrite(M1B, HIGH);
  delay(2000);
  digitalWrite(M1A, HIGH);
  digitalWrite(M1B, HIGH);
  delay(2000);
  digitalWrite(M1A, HIGH);
  digitalWrite(M1B,  LOW);
  delay(2000);
}
```

Servo Motor

Servo motor is an electrical device which can be used to rotate objects (like robotic arm) precisely.Servo motor consists of DC motor with error sensing negative feedback mechanism. This allows precise control over angular velocity and position of motor. In some cases, AC motors are used.
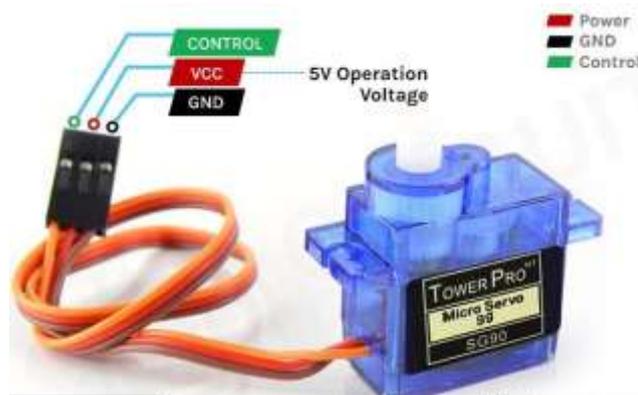
It is a closed loop system where it uses negative feedback to control motion and final position of the shaft.It is not used for continuous rotation like conventional AC/DC motors.It has rotation angle that varies from 0° to 380°.

Servo Motor Interfacing with Arduino Nano
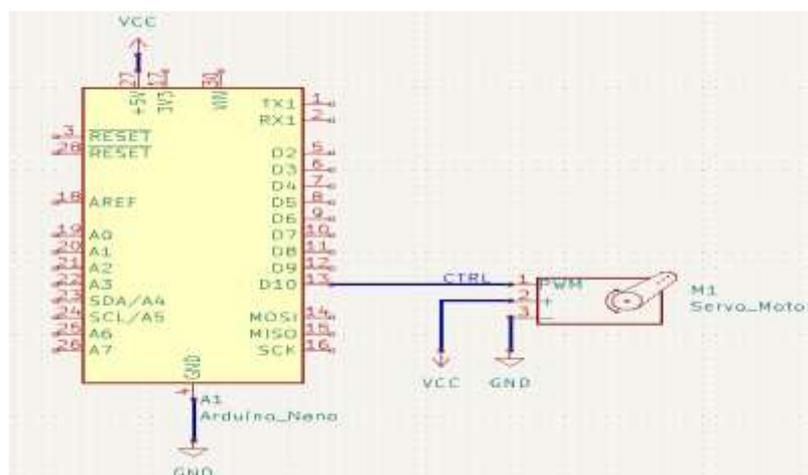
Servo Motor Code

```
//connect D10 pin to Servo motor CTRL pin

#include <Servo.h>

Servo myservo;  // create servo object to control a servo

// twelve servo objects can be created on most boards

int pos = 0;    // variable to store the servo position

void setup() {

 myservo.attach(10);  // attaches the servo on pin 9 to the servo object

}

void loop()

{

  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees

    // in steps of 1 degree

    myservo.write(pos);            // tell servo to go to position in variable 'pos'

    delay(15);                     // waits 15ms for the servo to reach the position

  }

  for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees

    myservo.write(pos);            // tell servo to go to position in variable 'pos'

    delay(15);                     // waits 15ms for the servo to reach the position

  }

}
```

Stepper Motor

Stepper Motor is a brushless DC Motor. Control signals are applied to stepper motor to rotate it in steps.

Its speed of rotation depends upon rate at which control signals are applied. There are various stepper motors available with minimum required step angle.

Stepper motor is made up of mainly two parts, a stator and rotor. Stator is of coil winding and rotor is mostly permanent magnet or ferromagnetic material.

 Step angle is the minimum angle that stepper motor will cover within one move/step. Number of steps required to complete one rotation depends upon step angle. Depending upon stepper motor configuration, step angle varies e.g. 0.72°, 3.8°, 3.75°, 7.5°, 35° etc.

Stepper Motor Interfacing with Arduino Nano
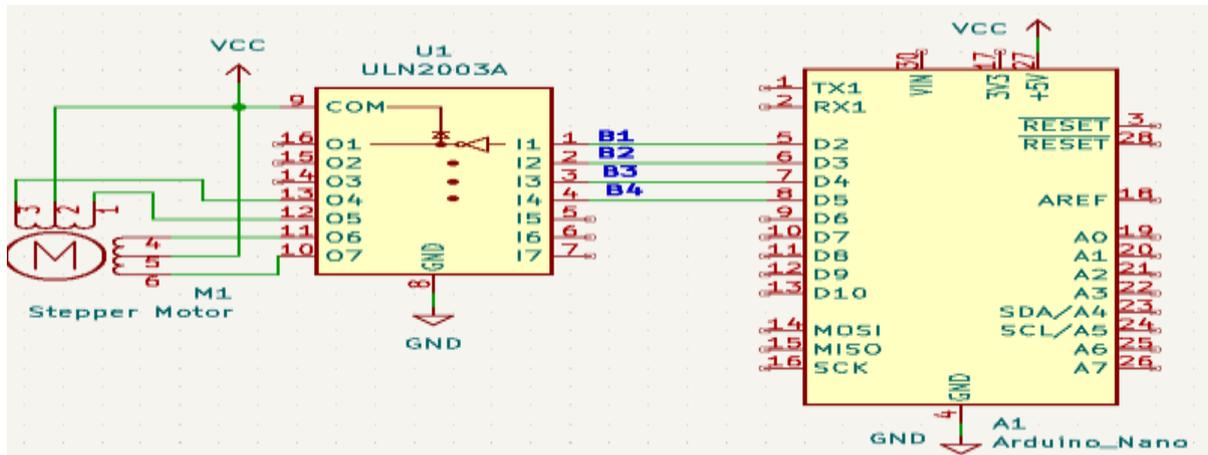
Stepper Motor Code

//connect D2,D3,D4 &D5 with B1,B2,B3 & B4 respectively

```
void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
}
// the loop function runs over and over again forever
void loop()
{
  digitalWrite(2,  LOW);
  digitalWrite(3,  LOW);
  digitalWrite(4,  LOW);
  digitalWrite(5, HIGH);
  delay(2);
  digitalWrite(2, LOW);
```

```
digitalWrite(3, LOW);
digitalWrite(4, HIGH);
digitalWrite(5, HIGH);
delay(2);
digitalWrite(2,  LOW);
digitalWrite(3,  LOW);
digitalWrite(4, HIGH);
digitalWrite(5,  LOW);
delay(2);


digitalWrite(2, LOW);
digitalWrite(3, HIGH);
digitalWrite(14, HIGH);
digitalWrite(5, LOW);
delay(2);
digitalWrite(2,  LOW);
digitalWrite(3, HIGH);
digitalWrite(4,  LOW);
digitalWrite(5,  LOW);
delay(2);
digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4,  LOW);
digitalWrite(5,  LOW);
delay(2);
digitalWrite(2, HIGH);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
```

```
digitalWrite(5, LOW);

delay(2);

digitalWrite(2, HIGH);

digitalWrite(3,  LOW);

digitalWrite(4,  LOW);

digitalWrite(5, HIGH);

delay(2);

}
```

IR Sensor

IR sensor is an electronic device, that emits the light in order to sense some object of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. Usually, in the **infrared spectrum**, all the objects radiate some form of thermal radiation. These types of radiations are invisible to our eyes, but infrared sensor can detect these radiations.

The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode . Photodiode is sensitive to IR light of the same wavelength which is emitted by the IR LED. When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received.

IR Sensor Interfacing with Arduino Nano
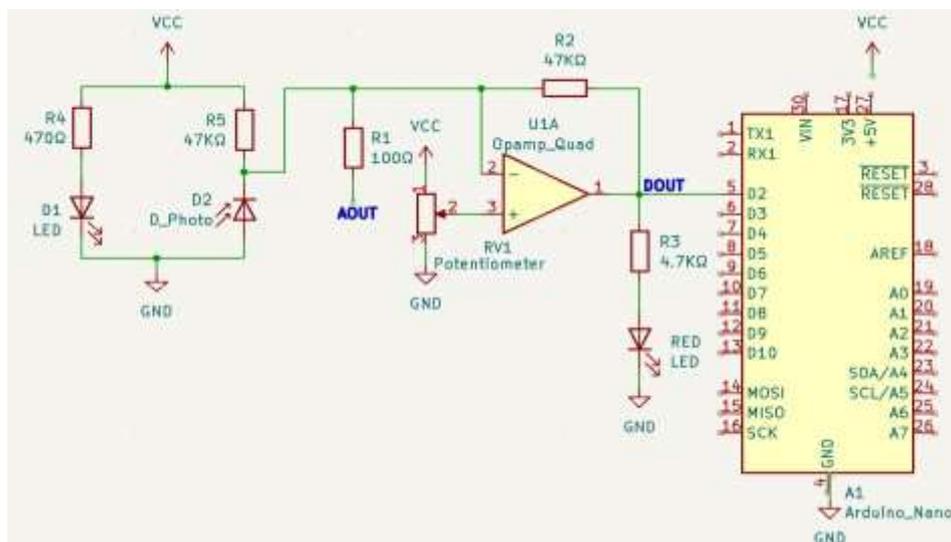
IR Sensor Code

```
//connect D2 TO IR PROXI DOUT & D3 TO LED PIN

int IRSensor = 2; // connect ir sensor module to Arduino pin 9

int LED = 3; // conect LED to Arduino pin 13

void setup()

{

Serial.begin(115200); // Init Serila at 115200 Baud

Serial.println("Serial Working"); // Test to check if serial is working or not

pinMode(IRSensor, INPUT); // IR Sensor pin INPUT

pinMode(LED, OUTPUT); // LED Pin Output

}

void loop()

{

int sensorStatus = digitalRead(IRSensor); // Set the GPIO as Input

if (sensorStatus == 1) // Check if the pin high or not

{

// if the pin is high turn off the onboard Led

 digitalWrite(LED,HIGH); // LED LOW

Serial.println("OBJECT  DETECTED"); // print Motion Detected! on the serial monitor window

}

else {

  //else turn on the onboard LED

  digitalWrite(LED, LOW); // LED High

  Serial.println(" OBJECT NOT DETECTED"); // print Motion Ended! on the serial monitor window

 }

delay(500);

}
```

PIR Sensor

[passive infrared sensor](#) is an electronic sensor that measures infrared light radiating from objects. PIR sensors mostly used in PIR-based motion detectors. Also, it used in security alarms and automatic lighting applications. The below image shows a typical pin configuration of the PIR sensor, which is quite simple to understand the pinouts. The PIR sensor consists of 3 pins,

- Pin1 corresponds to the drain terminal of the device, which connected to the positive supply 5V DC.
- Pin2 corresponds to the source terminal of the device, which connects to the ground terminal via a 100K or 47K resistor. The Pin2 is the output pin of the sensor. The pin 2 of the sensor carries the detected IR signal to an amplifier from the
- Pin3 of the sensor connected to the ground.

PIR Sensor Interfacing with Arduino Nano
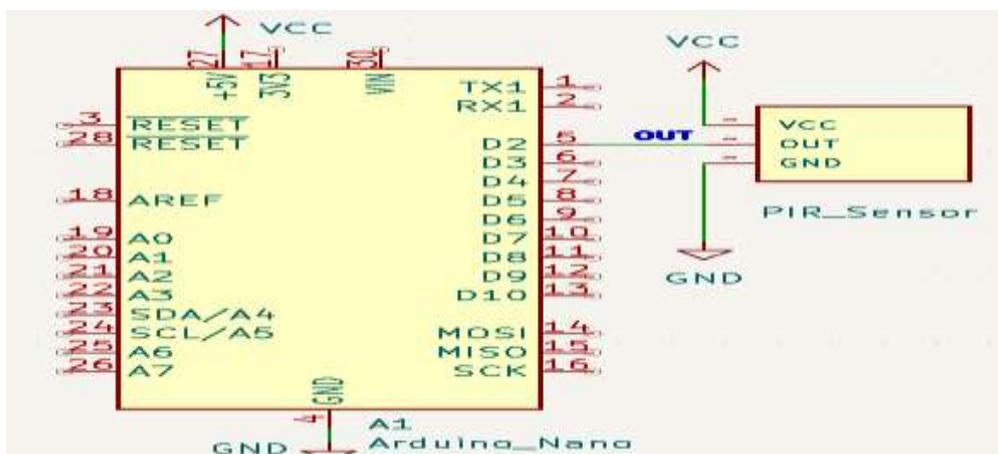
PIR Sensor Code

```
//CONNECT D2 TO PIR OUT D3 TO LED PIN
void setup() {
  pinMode(2, INPUT);//define arduino pin
  pinMode(3, OUTPUT);//define arduino pin
  Serial.begin(9600);//enable serial monitor
}
void loop() {
  bool value = digitalRead(2);//get value and save it boolean veriable
  if (value == 1) { //check condition
    digitalWrite(3,HIGH);//LED on
    Serial.println("MOTION DETECTED");//print serial monitor ON
  } else {
    //Serial.println("OFF");//print serial monitor OFF
    digitalWrite(3,LOW);//LED off
    Serial.println("MOTION NOT DETECTED");//print serial monitor ON
  }
  delay(1000);


}
```

Ultrasonic

Ultrasonic Module HC-SR04 works on the principle of SONAR and RADAR systems. It can be used to determine the distance of an object in the range of 2 cm – 400 cm. An ultrasonic sensor generates high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo which is sensed by the receiver

HC-SR-04 has an ultrasonic transmitter, receiver and control circuit.

In the ultrasonic module HCSR04, we have to give trigger pulse, so that it will generate ultrasound of frequency 40 kHz. After generating ultrasound i.e. 8 pulses of 40 kHz, it makes echo pin high. Echo pin remains high until it does not get the echo sound back. So the width of echo pin will be the time for sound to travel to the object and return back. Once we get the time we can calculate distance, as we know the speed of sound.

Ultrasonic Sensor Interfacing with Arduino Nano
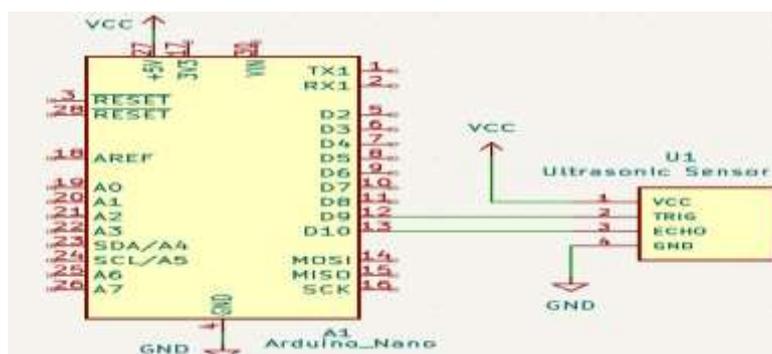
Ultrasonic Code

```
//CONNECT D9 TO TRIG & D10 TO ECHO

// defining the pins

const int trigPin = 9;

const int echoPin = 10;


// defining variables

long duration;

int distance;


void setup()

{

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

Serial.begin(9600); // Starts the serial communication

}


void loop()

 {

// Clears the trigPin

digitalWrite(trigPin, LOW);

delayMicroseconds(2);


// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);
```

// Reads the echoPin, returns the sound wave travel time in microseconds

duration = pulseIn(echoPin, HIGH);

// Calculating the distance

distance= duration*0.034/2;

// Prints the distance on the Serial Monitor

Serial.print("Distance: ");

Serial.println(distance);

}
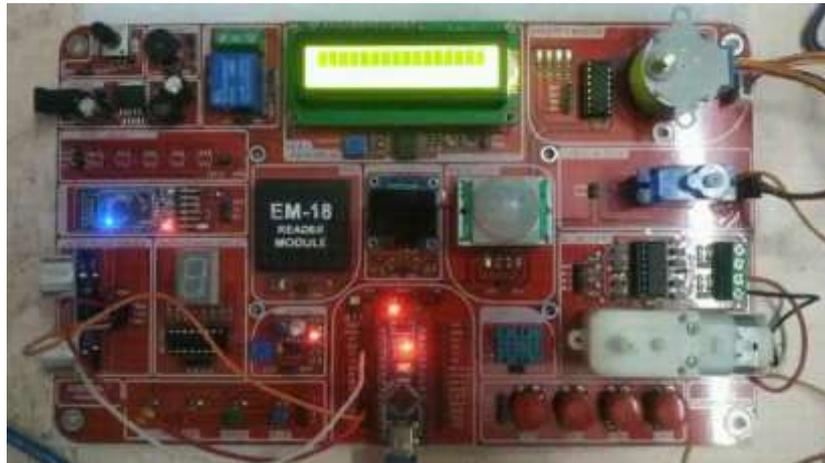

```
COM7

13:18:56.536 -> Distance: 112
13:18:56.582 -> Distance: 111
13:18:56.582 -> Distance: 112
13:18:56.629 -> Distance: 112
13:18:56.629 -> Distance: 112
13:18:56.675 -> Distance: 112
```

Bluetooth

It is used for many applications like wireless headset, game controllers, wireless mouse, wireless keyboard, and many more consumer applications.It has range up to <100m which depends upon transmitter and receiver, atmosphere, geographic & urban conditions.

ElectronicWings.com
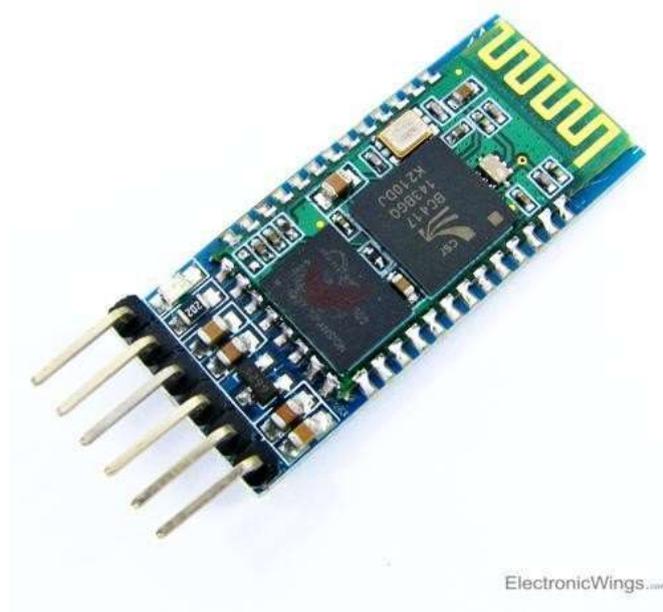
It is IEEE 802.15.1 standardized protocol, through which one can build wireless Personal Area Network (PAN). It uses frequency-hopping spread spectrum (FHSS) radio technology to send data over air.

It uses serial communication to communicate with devices. It communicates with microcontroller using serial port (USART).

Bluetooth Interfacing with Arduino Nano
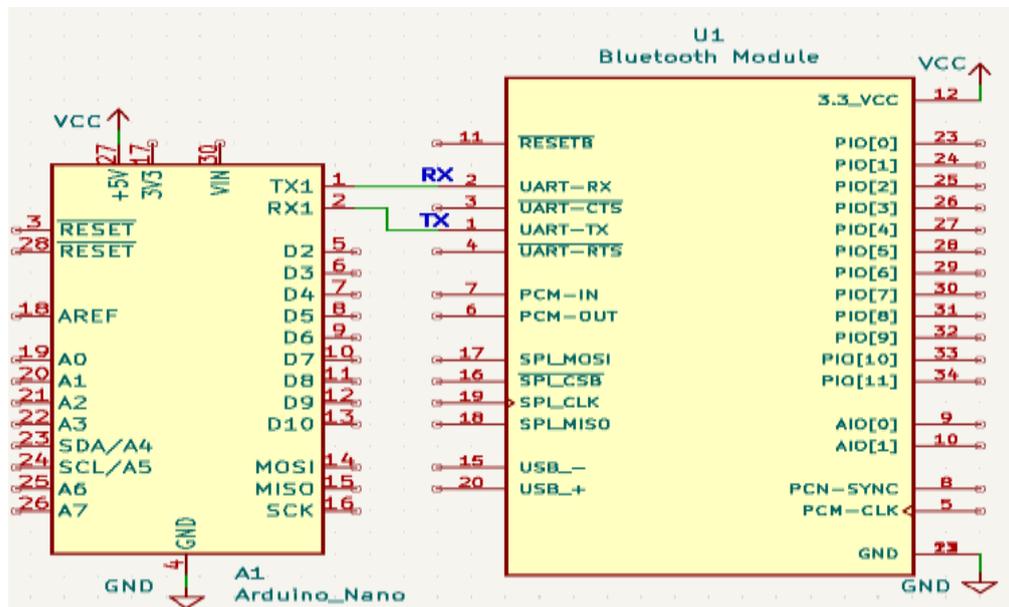
Bluetooth Code

//connect RX OF BLUETOOTH TO TX OF BOARD & TX OF BLUETOOYH TO RX OF BOARD AND D3 TO LED

char data = 0;   //Variable that store receive input

void setup()

{

  Serial.begin(9600); /*Baud Rate for serial communication*/

  pinMode(3, OUTPUT); /*D3 for LED*/

}

void loop()

{

 if(Serial.available() > 0)   /*check for serial data availability*/

  {

    data = Serial.read();   /*read data coming from Bluetooth device*/

    Serial.print(data);   /*print values on serial monitor*/

```
    Serial.print("\n");          /*print new line*/

    if(data == '1')              /*check data value*/

        digitalWrite(3, HIGH);  /*Turn ON LED if serial data is 1*/

    else if(data == '0')         /*check data value*/

        digitalWrite(3, LOW);  /*Turn OFF LED if serial data is 0*/

  }

}
```
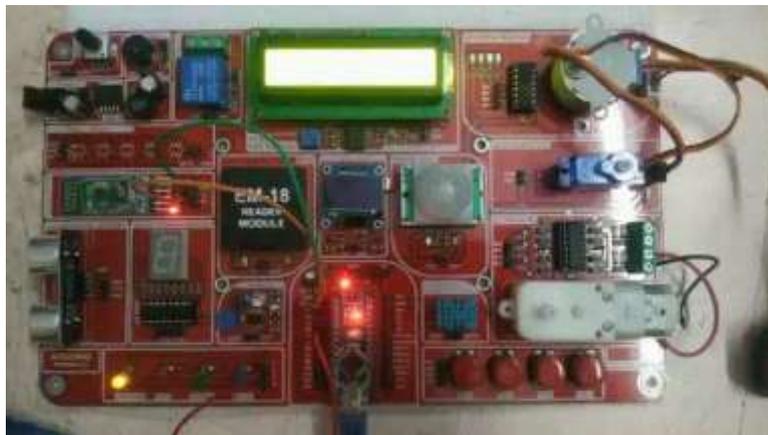
Seven Segment

Seven segment displays are important display units in Electronics and widely used to display numbers from 0 to 9. It can also display some character alphabets like A,B,C,H,F,E etc. It's the simplest unit to display numbers and characters. It just consists 8 LEDs, each LED used to illuminate one segment of unit and the 8th LED used to illuminate DOT in 7 segment display. We can refer each segment as a LINE, as we can see there are 7 lines in the unit, which are used to display a number/character. We can refer each line/segment "a,b,c,d,e,f,g" and for dot character we will use "h". There are 10 pins, in which 8 pins are used to refer a,b,c,d,e,f,g and h/dp, the two middle pins are common anode/cathode of all he LEDs. These common anode/cathode are internally shorted so we need to connect only one COM pin.
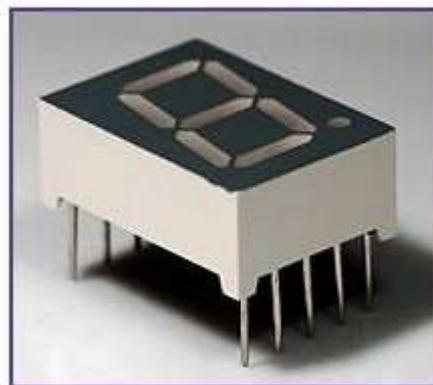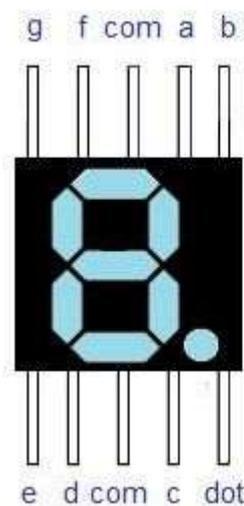
There are two types of 7 segment displays: Common Anode and Common Cathode:

Common Anode: In this all the Negative terminals (cathode) of all the 8 LEDs are connected together (see diagram below), named as COM. And all the positive terminals are left alone.

Common Cathode: In this all the positive terminals (Anodes) of all the 8 LEDs are connected together, named as COM. And all the negative thermals are left alone.

Seven Segment Display Interfacing with Arduino Nano
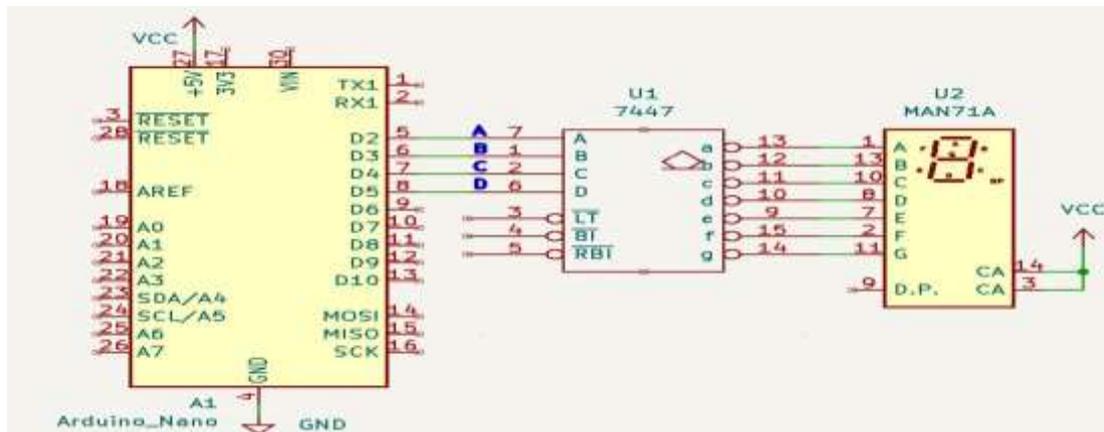
Seven Segment Code

//connect D2,D3,D4 & D5 to SEVEN SEGMENT DISPLAY A,B,C & D respectively

int inputs[4] = {2,3,4,5}; // A,B,C,D inputs

byte BCD[16][4] ={{0,0,0,0},

{1,0,0,0},

{0,1,0,0},

{1,1,0,0},

{0,0,1,0},

{1,0,1,0},

{0,1,1,0},

{1,1,1,0},

{0,0,0,1},

{1,0,0,1},

{0,1,0,1},

{1,1,0,1},

{0,0,1,1},

{1,0,1,1},

{0,1,1,1},

{1,1,1,1}}; //BCD code

```arduino
int number = 0; //which number in BCD code do you want to send
void setup() {
for(int a = 0; a < 4; a++){
pinMode(inputs[a], OUTPUT);} //set outputs
}
void loop() {
while(number < 10)
{
for(int c = 0; c < 4; c++)
{
digitalWrite(inputs[c], BCD[number][c]);
}
number++;
delay(500);
}
number = 0;
}
```

RFID (EM18)

Radio frequency Identification i.e. RFID is a wireless identification technology that uses radio waves to identify the presence of RFID tags.

Just like Bar code reader, RFID technology is used for identification of people, object etc. presence.

In barcode technology, we need to optically scan the barcode by keeping it in front of reader, whereas in RFID technology we just need to bring RFID tags in range of readers. Also, barcodes can get damaged or unreadable, which is not in the case for most of the RFID.

RFID is used in many applications like attendance system in which every person will have their separate RFID tag which will help identify person and their attendance. RFID is used in many companies to provide access to their authorized employees. It is also helpful to keep track of goods and in automated toll collection system on highway by embedding Tag (having unique ID) on them.

RFID Sensor Interfacing with Arduino Nano

RFID (EM18) Code

```
int count = 0;

char card_no[12];

void setup()

{

  Serial.begin(9600);

}

void loop()

{

  if(Serial.available())

  {

    count = 0;

    while(Serial.available() && count < 12)

    {

      card_no[count] = Serial.read();
```

```
        count++;

        delay(5);

    }

    Serial.println(card_no);

  }

}
```
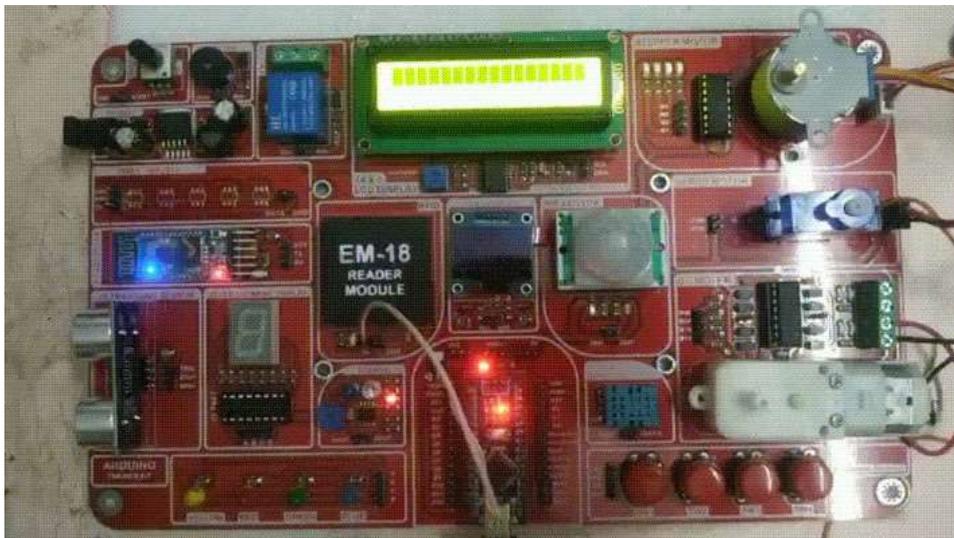


```
13:35:32.123 -> 0B0029213B38↑
13:35:35.003 -> 0B0029213B38↑
13:35:37.906 -> 30005138BDE4↑
13:35:39.125 -> 30005138BDE4↑
```

Potentiometer

A **potentiometer** is a three-terminal resistor with a sliding or rotating contact that forms an adjustable voltage divider. If only two terminals are used, one end and the wiper, it acts as a **variable resistor** or **rheostat**.

The measuring instrument called a potentiometer is essentially a voltage divider used for measuring electric potential (voltage); the component is an implementation of the same principle, hence its name.

Potentiometers are commonly used to control electrical devices such as volume controls on audio equipment. Potentiometers operated by a mechanism can be used as position transducers, for example, in a joystick. Potentiometers are rarely used to directly control significant power (more than a watt), since the power dissipated in the potentiometer would be comparable to the power in the controlled load.

Potentiometer Code

//CONNECT A0 TO POTENTIOMETWR OUT & D2 TO LED PIN

const int analogInPin = A0; // Analog input pin that the potentiometer is attached to

const int analogOutPin = 2; // Analog output pin that the LED is attached to

int sensorValue = 0;   // value read from the pot

int outputValue = 0;   // value output to the PWM (analog out)

void setup() {

```
// initialize serial communications at 9600 bps:
 Serial.begin(9600);
}


void loop() {
 // read the analog in value:
 sensorValue = analogRead(analogInPin);
 // map it to the range of the analog out:
 outputValue = map(sensorValue, 0, 1023, 0, 255);
 // change the analog out value:
 analogWrite(analogOutPin, outputValue);


 // print the results to the Serial Monitor:
 Serial.print("sensor = ");
 Serial.print(sensorValue);
 Serial.print("\t output = ");
 Serial.println(outputValue);


 // wait 2 milliseconds before the next loop for the analog-to-digital
 // converter to settle after the last reading:
 delay(1000);
}
```
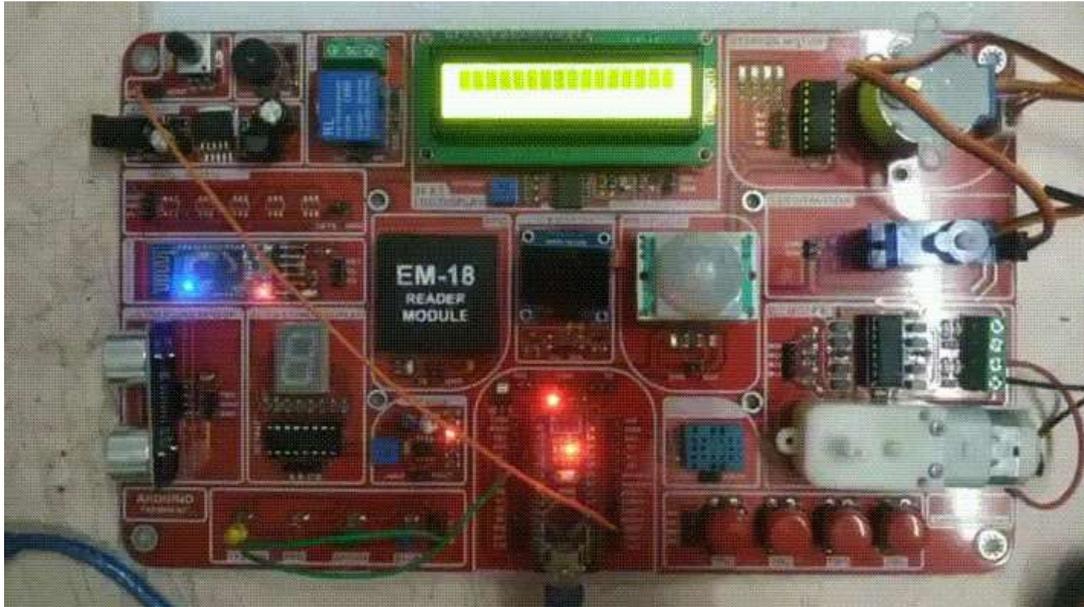
```
COM7

12:56:38.655 -> sensor = 114      output = 28
13:02:27.253 -> sensor = 117      output = 29
13:02:28.235 -> sensor = 114      output = 28
13:02:29.263 -> sensor = 116      output = 28
13:02:30.240 -> sensor = 115      output = 28
13:02:31.262 -> sensor = 115      output = 28
13:02:32.241 -> sensor = 117      output = 29
13:02:33.265 -> sensor = 114      output = 28
13:02:34.240 -> sensor = 196      output = 48
13:02:35.263 -> sensor = 338      output = 84
13:02:36.238 -> sensor = 572      output = 142
13:02:37.261 -> sensor = 696      output = 173
13:02:38.238 -> sensor = 808      output = 201
13:02:39.261 -> sensor = 998      output = 248
13:02:40.240 -> sensor = 1023     output = 255
13:02:41.262 -> sensor = 675      output = 168
13:02:42.237 -> sensor = 1        output = 0
13:02:43.266 -> sensor = 0        output = 0
13:02:44.248 -> sensor = 0        output = 0
13:02:45.270 -> sensor = 0        output = 0
13:02:46.247 -> sensor = 51       output = 12
13:02:47.266 -> sensor = 332      output = 82
13:02:48.247 -> sensor = 1021     output = 254
13:02:49.273 -> sensor = 1021     output = 254
13:02:50.244 -> sensor = 972      output = 242
13:02:51.265 -> sensor = 81       output = 20
13:02:52.235 -> sensor = 0        output = 0
13:02:53.258 -> sensor = 0        output = 0
13:02:54.235 -> sensor = 0        output = 0
13:02:55.259 -> sensor = 0        output = 0
13:02:56.235 -> sensor = 0        output = 0
13:02:57.258 -> sensor = 0        output = 0
13:02:58.233 -> sensor = 0        output = 0
13:02:59.254 -> sensor = 15       output = 3
13:03:00.280 -> sensor = 1020     output = 254
13:03:01.258 -> sensor = 1020     output = 254
```