SSG EMBEDDED SOLUTIONS

SSG

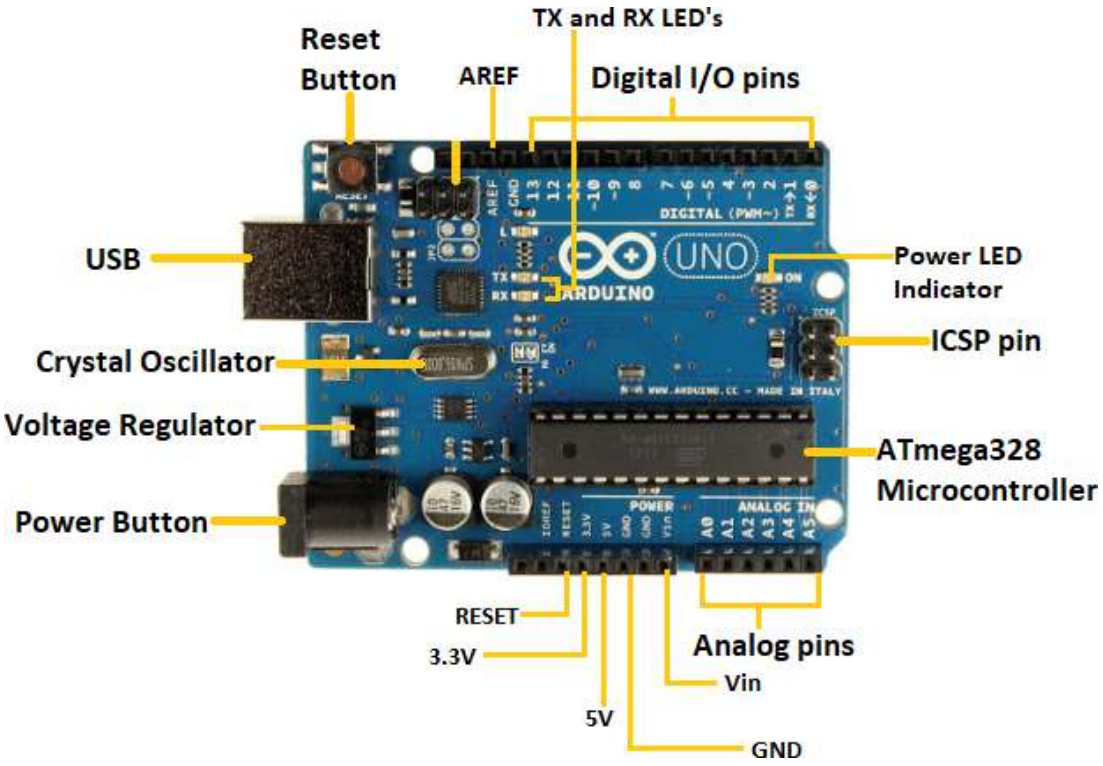
info@ssges.co.in

Development Kit of ARDUINO UNO Documentation

List of Contents

Introduction

Arduino UNO is one of the famous microcontroller boards of the Arduino family and is developed by Arduino.cc. Basically Arduino.cc is an open-source platform and is mainly based upon AVR microcontroller Atmega328. It is one of the most economical boards of Arduino family and is widely used because of its small number of input-output pins and reduced size as compared to Arduino mega which is the big brother of Arduino UNO. In this article, we are going to briefly discuss Arduino UNO. We will discuss its specifications, pin configuration, dimensions, shields compatibility, software for coding, applications and projects in which this board is used.

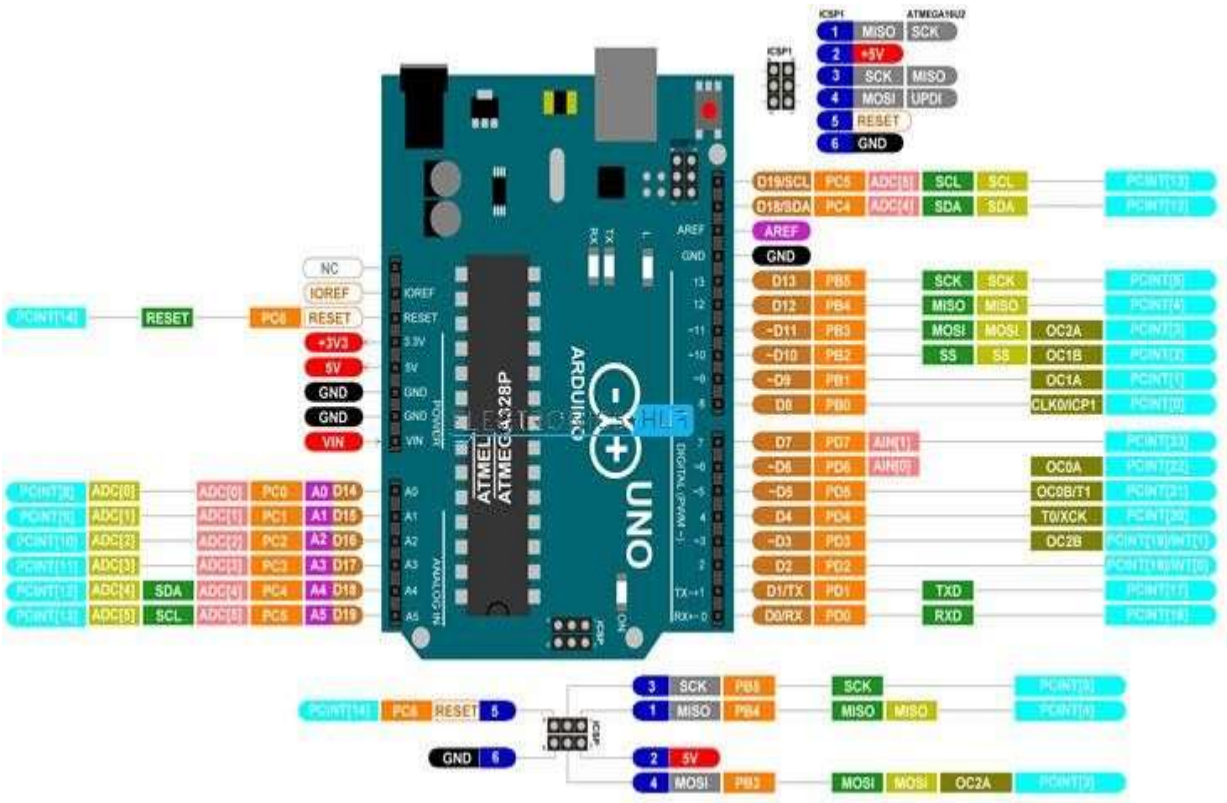


The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B USB cable. It can be powered by a USB cable or a barrel connector that accepts voltages between 7 and 20 volts, such as a rectangular 9-volt battery. It is similar to the Arduino NANO and Leonardo. The hardware reference design is distributed under a Creative Commons Attribution Share-Alike 2.5 license and is available on the Arduino website. Layout and production files for some versions of the hardware are also available.

Arduino Uno supports three different types of communication interfaces. They are:

- Serial
- I2C or I^2C
- SPI

Pin Diagram



Specifications

- Microcontroller: ATmega328
- Operating Voltage: 5V
- Input Voltage (recommended): 7-12V
- Input Voltage (limits): 6-20V
- Digital I/O Pins: 14 (of which 6 provide PWM output)
- Analog Input Pins: 6
- DC Current per I/O Pin: 40 mA
- DC Current for 3.3V Pin: 50 mA
- Flash Memory: 32 KB of which 0.5 KB used by bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Speed: 16 MHz

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

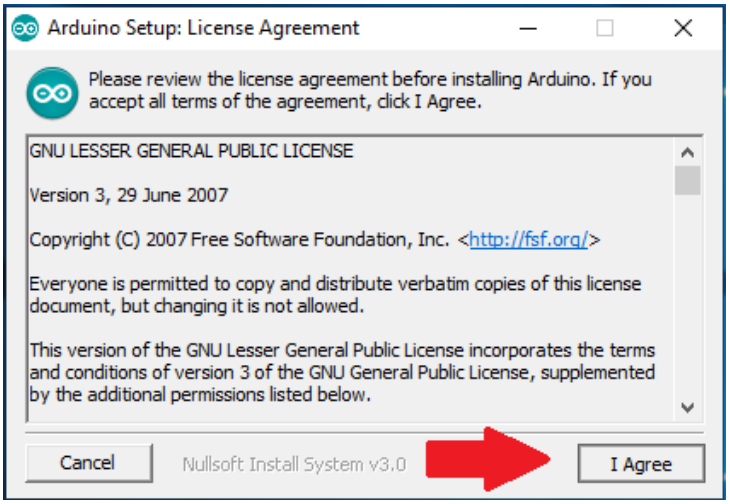
Material Required

- NANO board
- Cable

How to Install the Arduino IDE

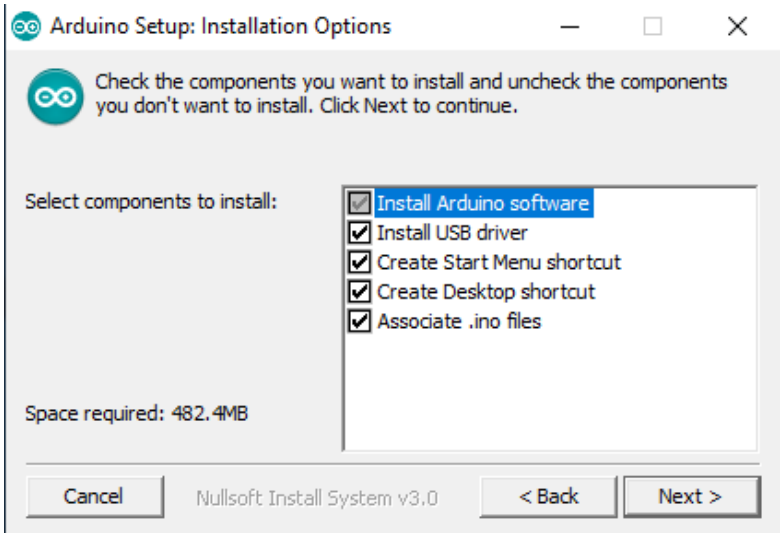
Step 1: Download File Arduino IDE

Step 2: License Agreement



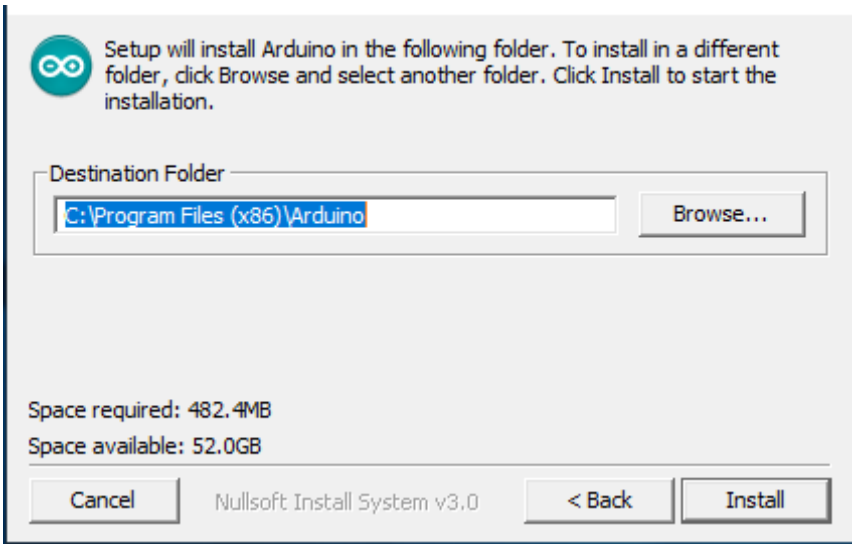
After the file is run, the "License Agreement" page will apper. You can read it, then click "*I Agree*" to continue.

Step 3: Installation Option



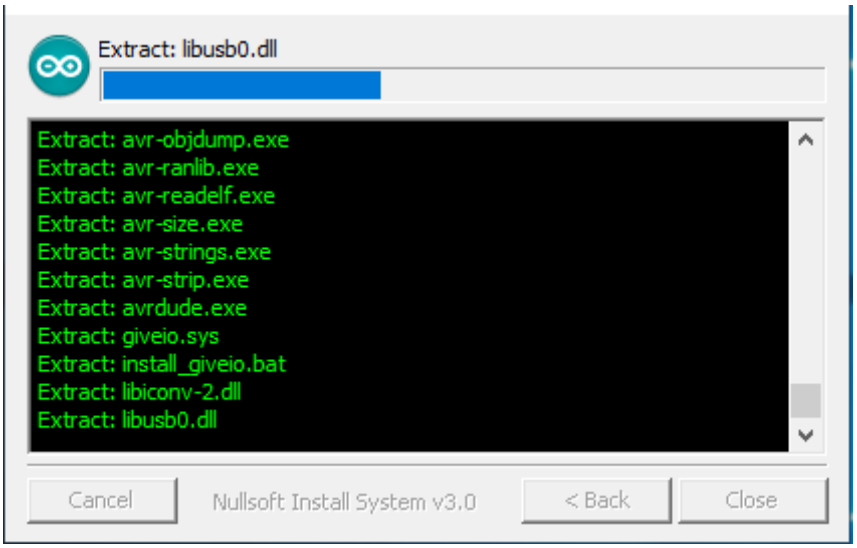
Check the component that you want to install and uncheck the components that you don't want to install. I suggest installing all componen. Click "*next*" to continue.

Step 4: Installation Folder



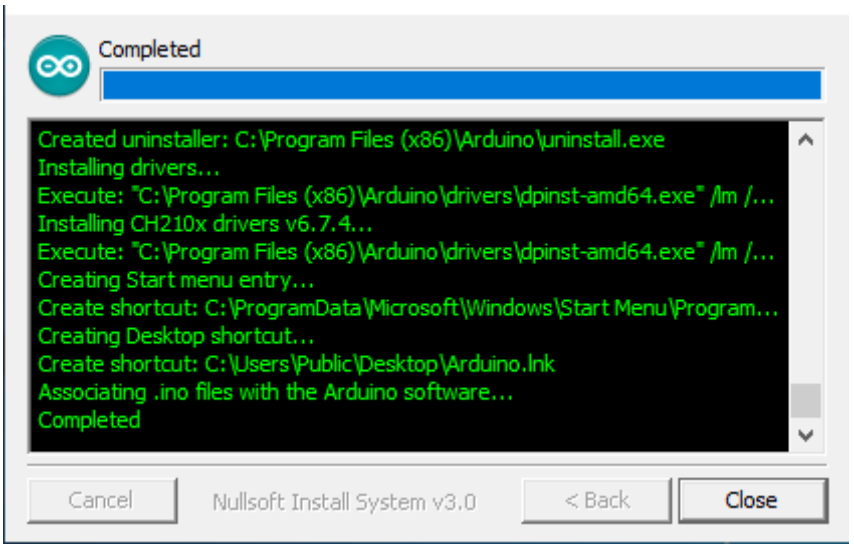
Arduino will automatically be installed in "**C:\Program Files (x86)\Arduino**". If you want to change the folder, click "*Browse*" and select the desired folder. Click install to start the installation.

Step 5: Installing Proses



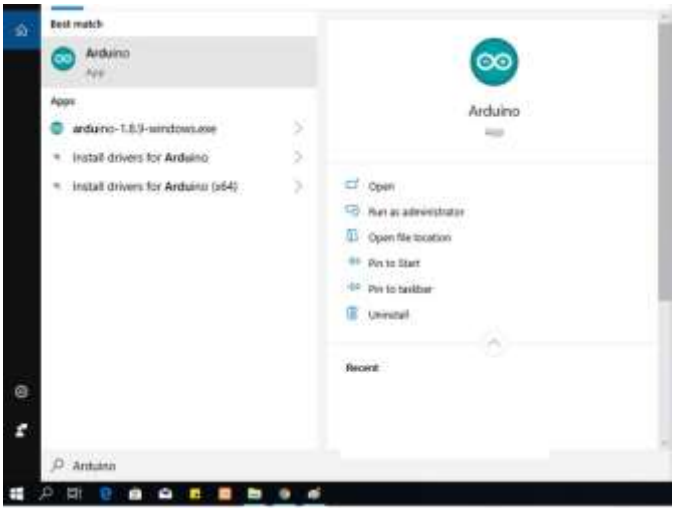
The installation process is ongoing.

Step 6: Installation Complete



If there is written "*complete*", it means that the isntallation process is complete. click "*Close*".

Step 7: Open Arduino IDE



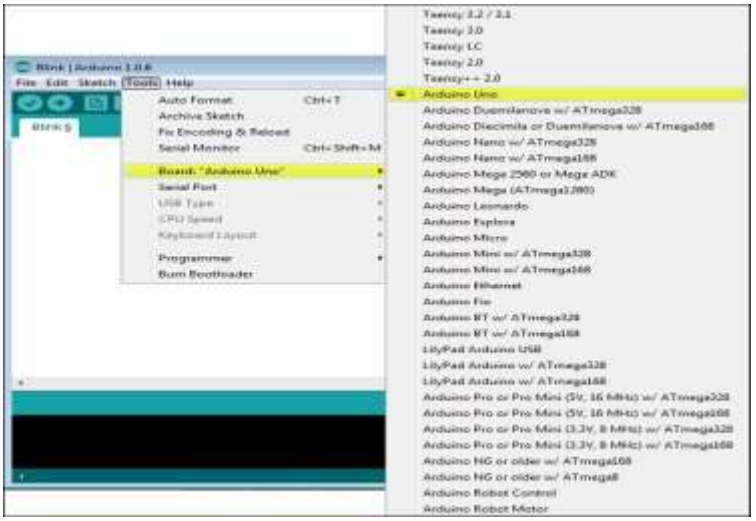
After the installation process is complete, there will be an Arduino icon on the Desktop. Or check on the search icon and write "arduino". If you have found the arduino icon, run the application.

Step 8: Display Arduino IDE

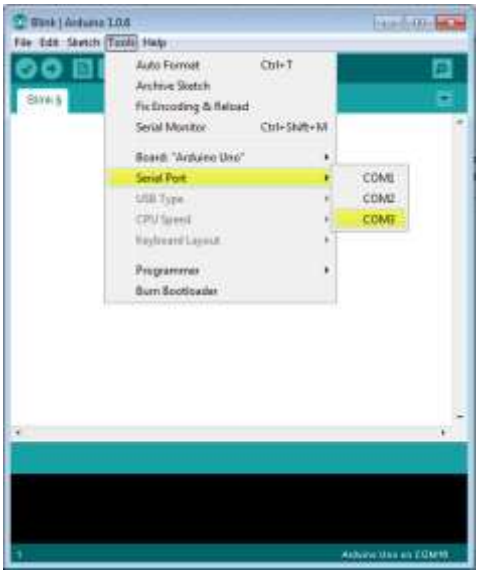
This is a display of the Arduino IDE Software. The application is ready to be used to create amazing projects.

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

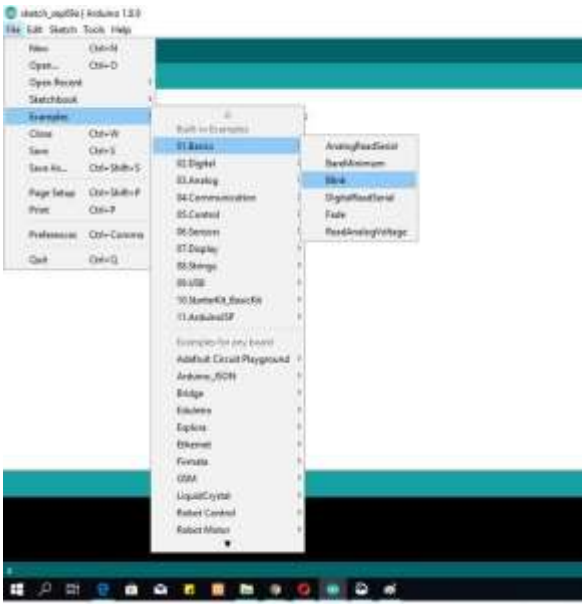
Step 9: Select the Board That Is Used



Step 10: Select your serial port



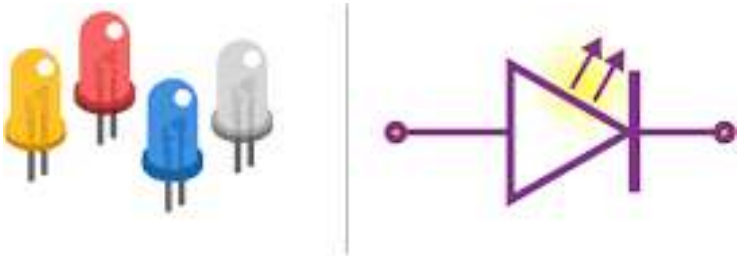
Step 11: Open and Upload Sketch



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

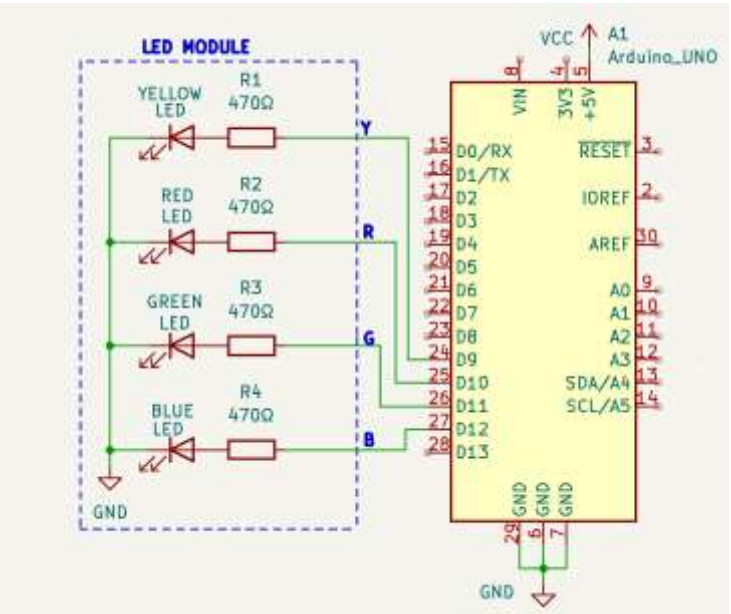
It is most widely used semiconductor which emit either visible light or invisible infrared light when forward biased. Remote controls generate invisible light. A Light emitting diodes (LED) is an optical electrical energy into light energy when voltage is applied.



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.

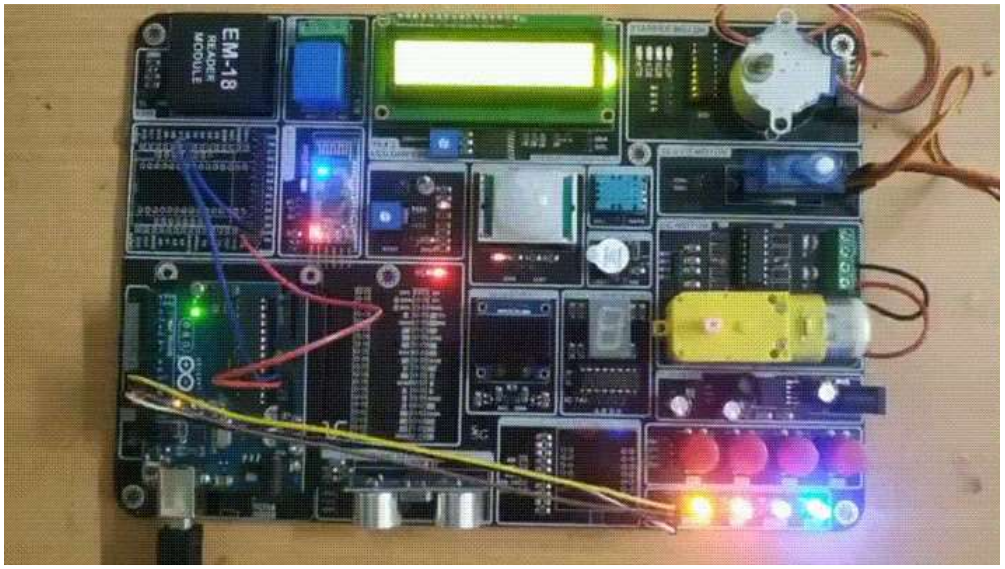
LED interfacing with Arduino Uno:



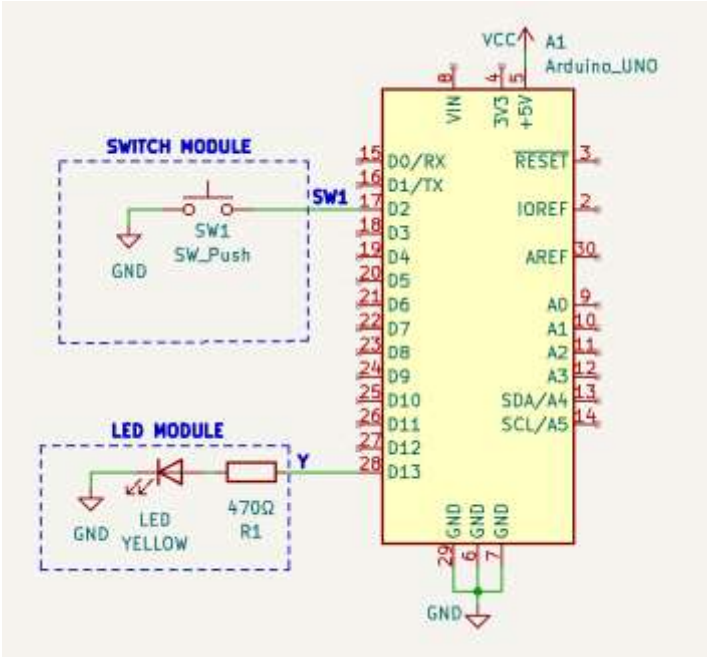
LED Blinking Code

```
//connect 9,10,11 & 12 to led pins yellow,red,green & blue
#define LED_PIN_4 12
#define LED_PIN_3 11
#define LED_PIN_2 10
#define LED_PIN_1 9

void setup()
{
  pinMode(LED_PIN_1,  OUTPUT);
  pinMode(LED_PIN_2,  OUTPUT);
  pinMode(LED_PIN_3,  OUTPUT);
  pinMode(LED_PIN_4, OUTPUT);
}

void loop()
{
  digitalWrite(LED_PIN_1, LOW);
  digitalWrite(LED_PIN_2, LOW);
  digitalWrite(LED_PIN_3, LOW);
  digitalWrite(LED_PIN_4, LOW);
  delay(1000);

  digitalWrite(LED_PIN_1, HIGH);
  digitalWrite(LED_PIN_2, HIGH);
  digitalWrite(LED_PIN_3, HIGH);
```

```
  digitalWrite(LED_PIN_4, HIGH);
  delay(1000);
}
```



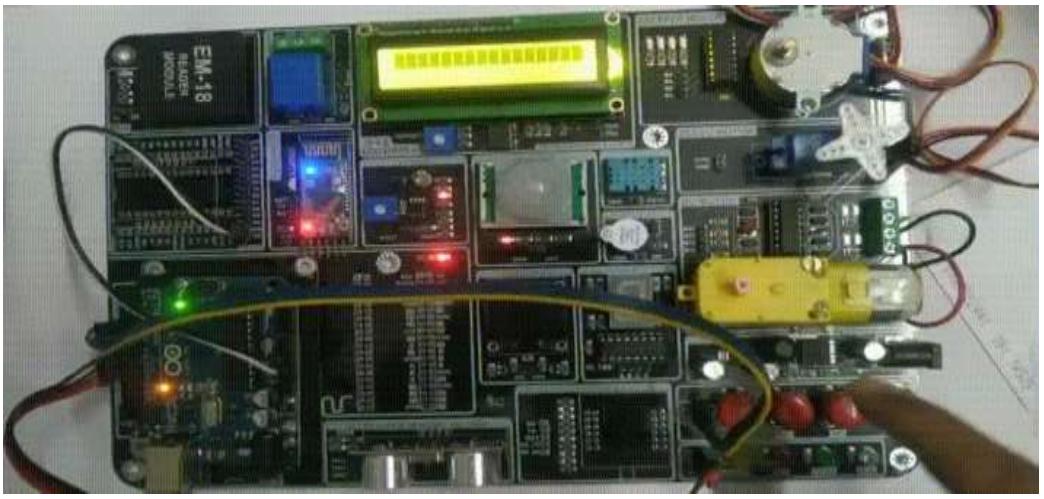
Interfacing of SWITCH with Arduino Uno



LED with Switch Code

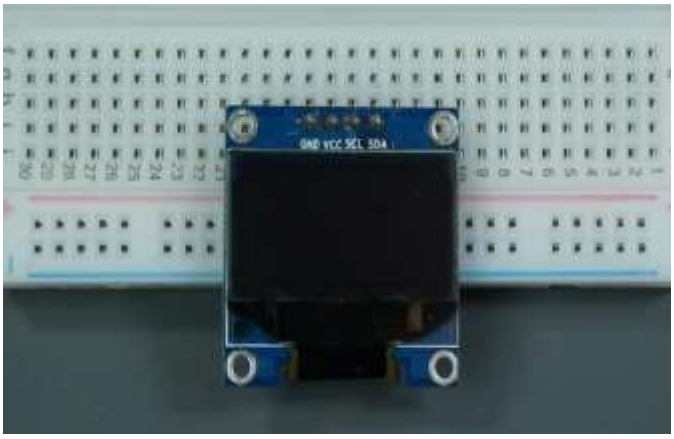
```
//connect pin 13 to led & pin 2 to switch
#define Y  8
#define R  9
#define G  10
#define B  11

#define buttonState1  4
#define buttonState2  5
#define buttonState3  6
#define buttonState4  7

void setup() {
  pinMode(buttonState1, INPUT_PULLUP);
  pinMode(buttonState2, INPUT_PULLUP);
  pinMode(buttonState3, INPUT_PULLUP);
  pinMode(buttonState4, INPUT_PULLUP);
  pinMode(Y, OUTPUT);
  pinMode(R,  OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(B, OUTPUT);

}

void loop() {
 if (digitalRead(buttonState1) == LOW)
```

```
  digitalWrite(Y, HIGH);
 else
  digitalWrite(Y, LOW);

if (digitalRead(buttonState2) == LOW)
  digitalWrite(R, HIGH);
 else
  digitalWrite(R, LOW);

if (digitalRead(buttonState3) == LOW)
  digitalWrite(G, HIGH);
 else
  digitalWrite(G, LOW);

if (digitalRead(buttonState4) == LOW)
  digitalWrite(B, HIGH);
 else
  digitalWrite(B, LOW);
}
```



OLED

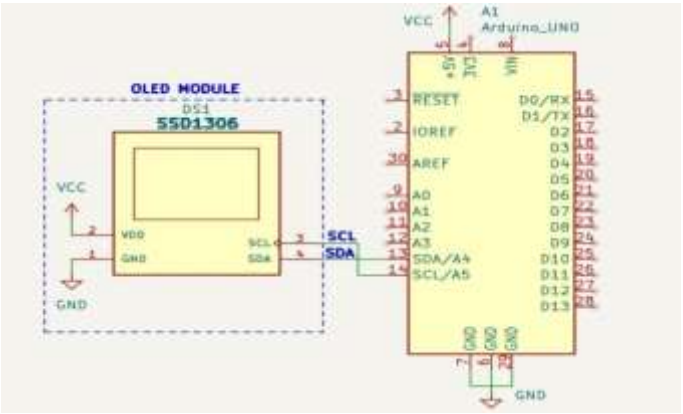
OLED is the acronym for **Organic Light Emitting Diode**. OLED is a modern display technology used in a wide range of electronic display devices, such as TVs, monitors, laptops, smartphones, bulletin boards, stadium screens, etc.



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.

Interfacing of OLED display with Arduino Uno

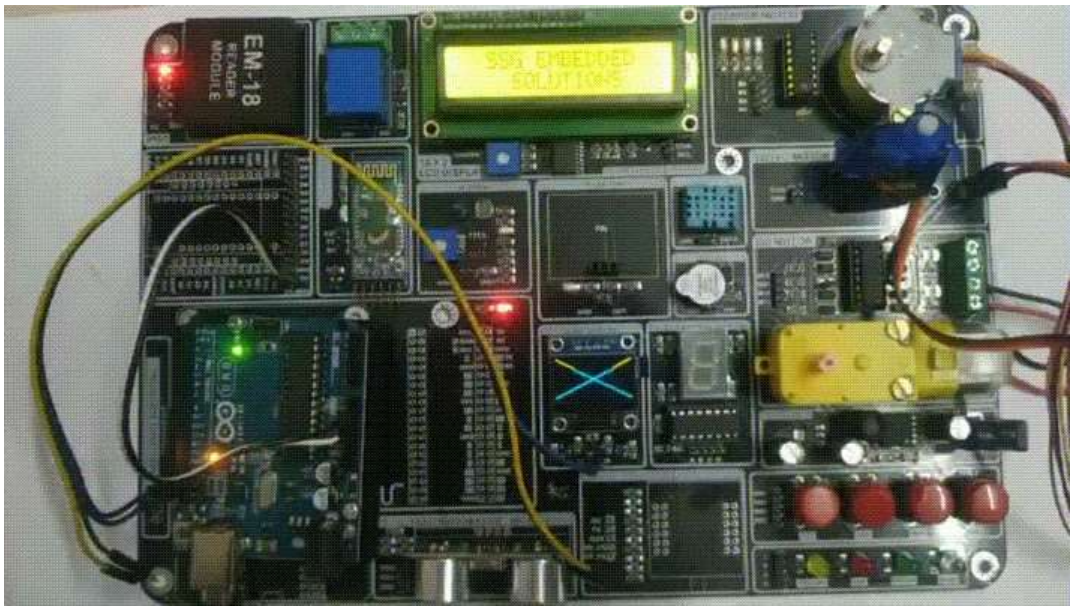


OLED Code

```
//connect SDA TO SDA of OLED & SCL TO SCL of OLED
#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 32 // OLED display height, in pixels
#define OLED_RESET 4
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
void setup()  {
  Serial.begin(9600);
  // by default, we'll generate the high voltage from the 3.3v line internally! (neat!)
  display.begin(SSD1306_SWITCHCAPVCC, 0x3C); // initialize with the I2C addr 0x3C (for the 128x32)// Check your I2C address and enter it here, in Our case address is 0x3C
  display.clearDisplay();
  display.display(); // this command will display all the data which is in buffer
}
void loop() {
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0,10);
  display.println("SSG EMBEDED SOLUTIONS");
  display.display();
  delay(3000);
// drawing diagonal lines
  display.clearDisplay();
```

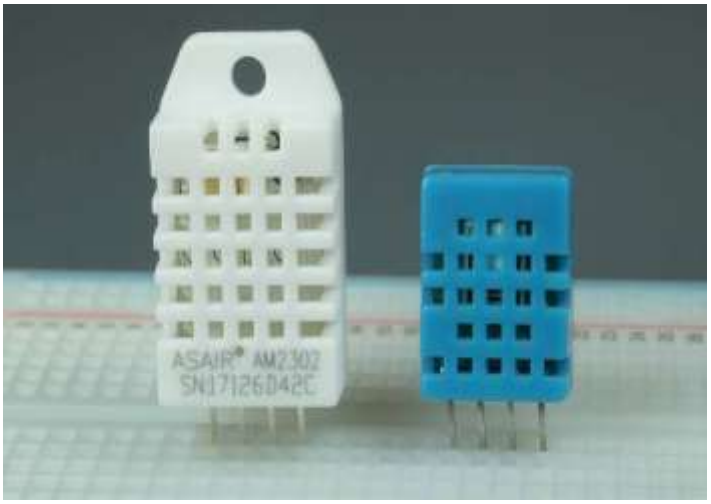
```
  display.drawLine(0,0,display.width() - 1, display.height() - 1, WHITE);
  display.drawLine(display.width() - 1,0,0, display.height() - 1, WHITE);
  display.display();
  delay(5000);
//drawing rectangles
  display.clearDisplay();
  display.drawRect(100, 10, 20, 20, WHITE);
  display.fillRect(10, 10, 45, 15, WHITE);
  display.drawRoundRect(60, 20, 35, 35, 8, WHITE);
  display.display();
  delay(5000);
//drawing circles
  display.clearDisplay();
  display.drawCircle(30, 15, 15, WHITE);
  display.fillCircle(25, 10, 2, WHITE);
  display.fillCircle(35, 10, 2, WHITE);
  display.display();
  delay(5000);
//drawing points
  display.clearDisplay();
  display.drawPixel(20, 35, WHITE);
  display.drawPixel(45, 12, WHITE);
  display.drawPixel(120, 59, WHITE);
  display.drawPixel(97, 20, WHITE);
  display.drawPixel(35, 36, WHITE);
  display.drawPixel(72, 19, WHITE);
  display.drawPixel(90, 7, WHITE);
  display.drawPixel(11, 29, WHITE);
```

```
  display.drawPixel(57, 42, WHITE);
  display.drawPixel(69, 34,  WHITE);
  display.drawPixel(108, 12, WHITE);
  display.display();
  delay(5000);
}
```



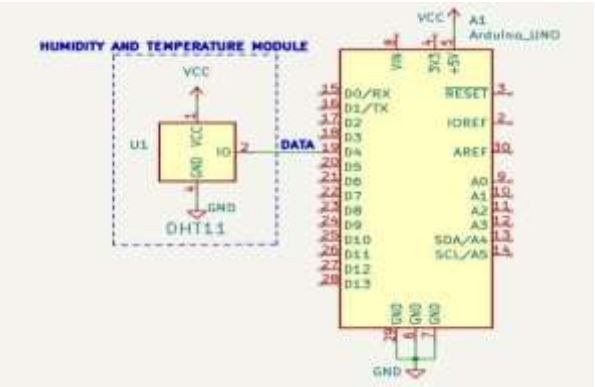
DHT11

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.

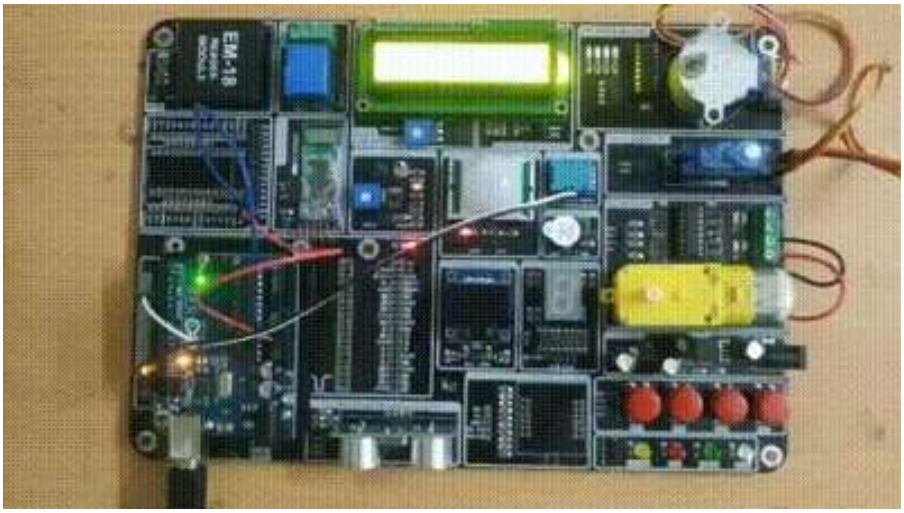
DHT11 interfacing with Arduino Uno



DHT11 Code

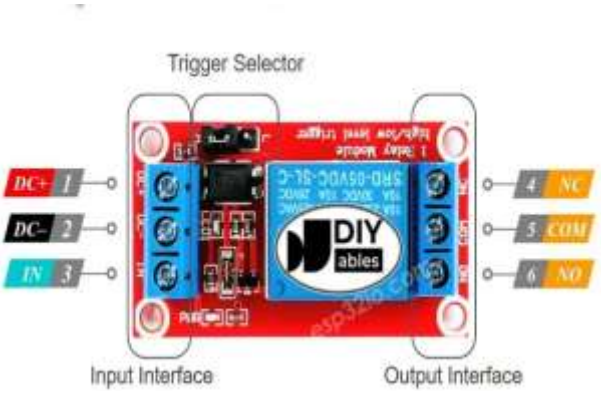
```
//connect 4 to DATA of dht11
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>
#define DHTTYPE    DHT11     // DHT 11
#define DHTPIN 4

DHT_Unified dht(DHTPIN, DHTTYPE);
uint32_t delayMS;

 void setup()
 {
  Serial.begin(9600);
  dht.begin();

  sensor_t sensor;
  //delayMS = sensor.min_delay / 1000;
  delay(500);
}
void loop()
{
  sensors_event_t event;
  dht.temperature().getEvent(&event);

  Serial.print(F("Temperature: "));
  Serial.print(event.temperature);
  Serial.println(F("°C"));
```

```
  dht.humidity().getEvent(&event);
  Serial.print(F("Humidity: "));
  Serial.print(event.relative_humidity);
  Serial.println(F("%"));
  //delay(delayMS);
  delay(500);
}
```



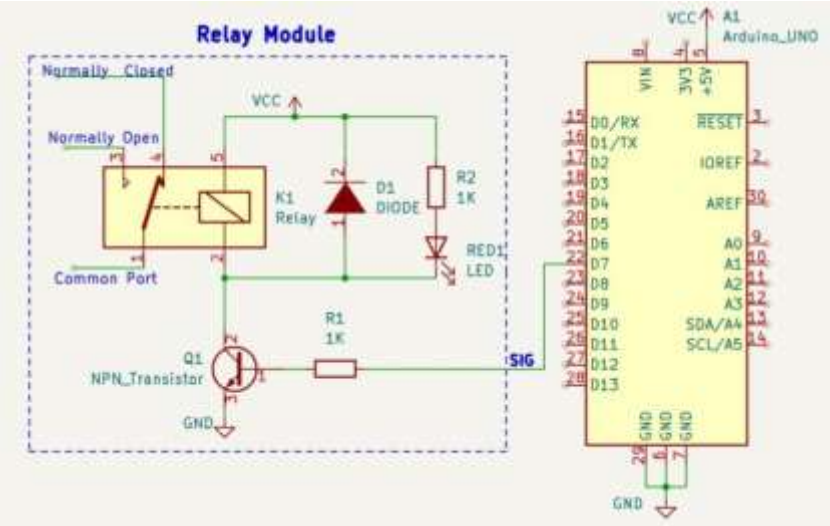
Relay

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.



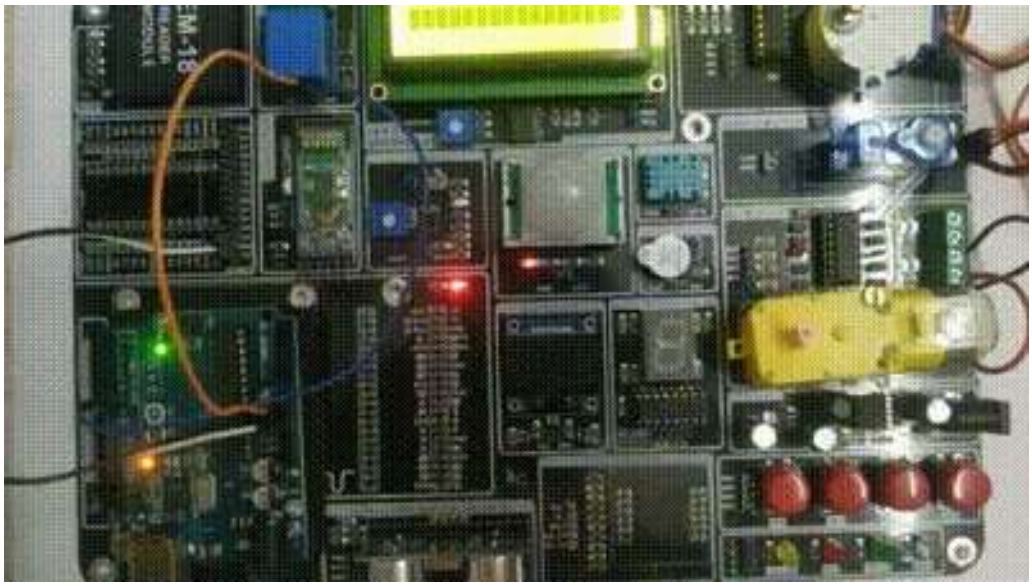
Relays are normally used in the control panels, manufacturing, and building automation to control the power along with switching the smaller current values in a control circuit. However, the supply of amplifying effect can help control the large amperes and voltages because if low voltage is applied to the relay coil, a large voltage can be switched by the contacts.

Relay interfacing with Arduino Uno



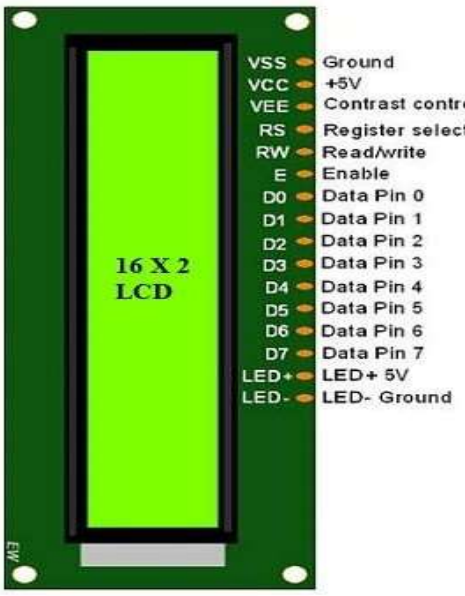
Relay Code

```
//connect pin 7 to relay pin
int relay_pin = 7;


void setup()
{
  pinMode(relay_pin,OUTPUT);


}
void loop()
{
  digitalWrite(relay_pin,HIGH);
  delay(500);
  digitalWrite(relay_pin,LOW);
  delay(500);
}
```



LCD

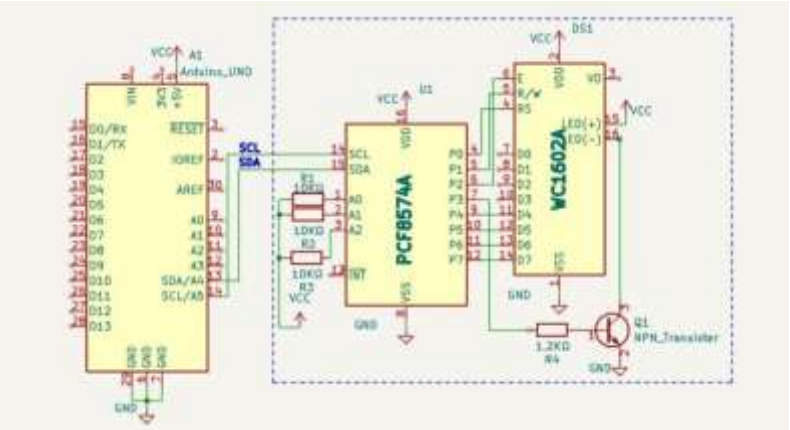
The term LCD stands for liquid crystal display. It is one kind of electronic display module used in an extensive range of applications like various circuits & devices like mobile phones, calculators, computers, TV sets, etc. These displays are mainly preferred for multi-segment light-emitting diodes 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

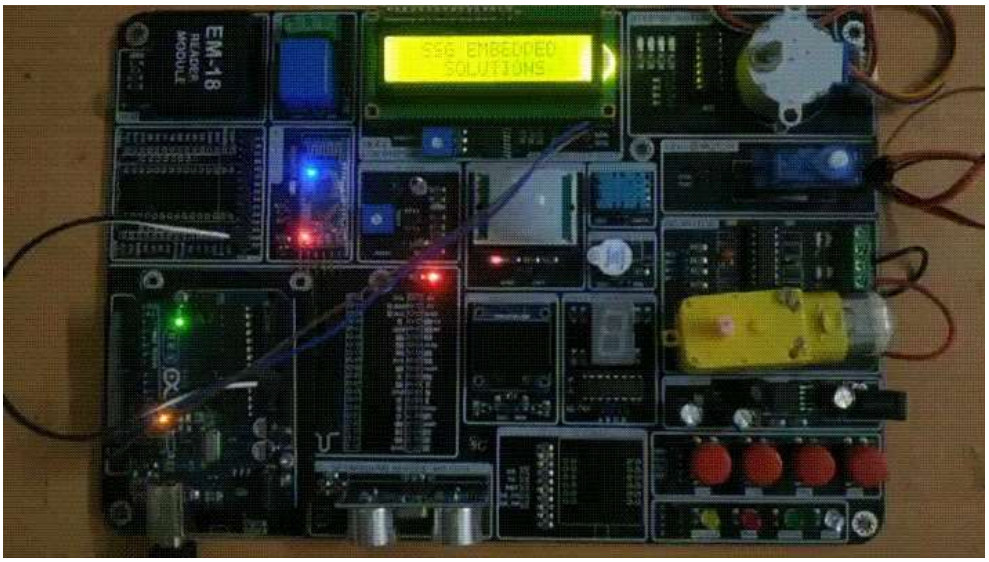
Interfacing of LCD display with Arduino Uno



LCD Code

```
//CONNECT SDA & SCL TO LCD DISPLAY
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup()
{

  lcd.init();



  lcd.backlight();
  lcd.clear();
  lcd.setCursor(2,0);
  lcd.print("SSG EMBEDDED ");
  lcd.setCursor(4,1);
```

```
  lcd.print("SOLUTIONS ");
}

void loop()
{
}
```

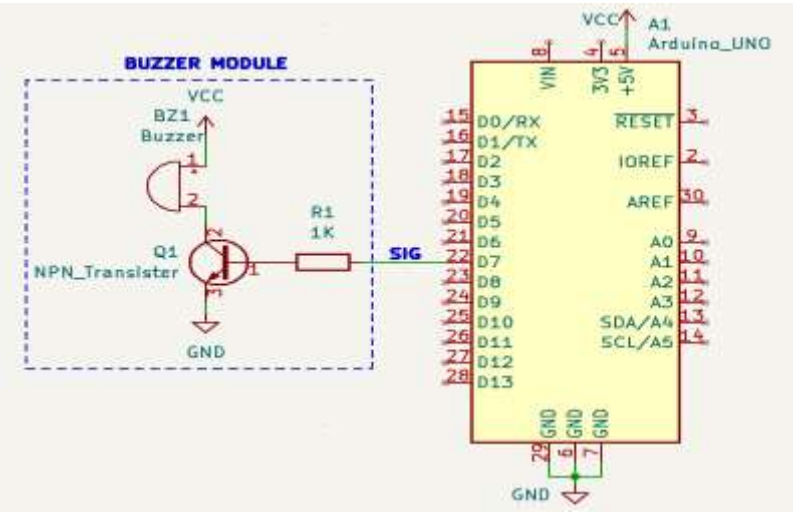


Buzzer

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.

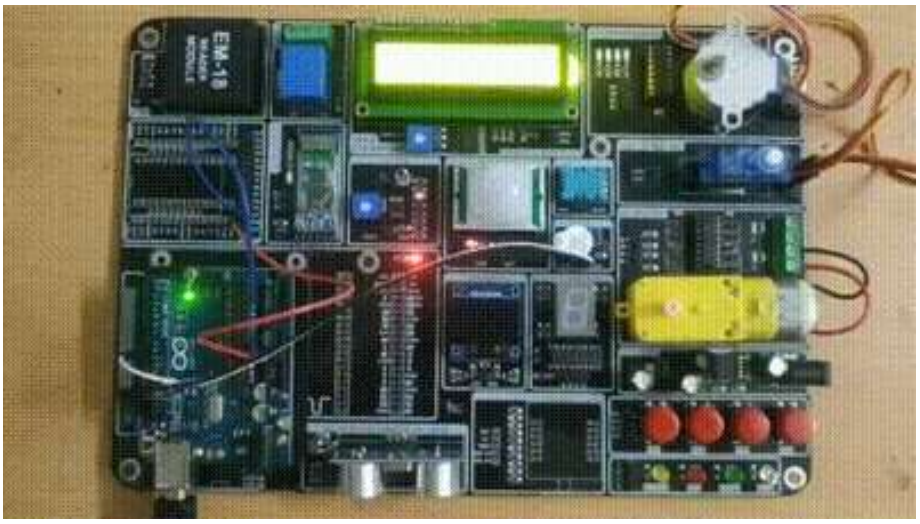
Buzzers are commonly used in a wide range of applications such as alarms, timers, and warning systems. They can also be used in electronic devices such as mobile phones, computers, and other electronic devices to generate different sounds and tones.

Interfacing of Buzzer with Arduino Uno

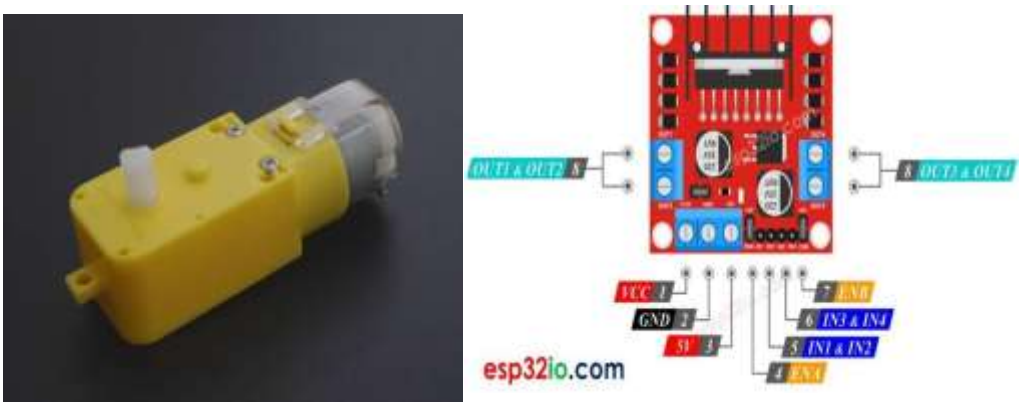


Code

```
//connect pin 9 to buzzer sin pin
int buzzer_pin = 9;


void setup()
{
  pinMode(buzzer_pin,OUTPUT);


}
void loop()
{
  digitalWrite(buzzer_pin,HIGH);
  delay(2000);
  digitalWrite(buzzer_pin,LOW);
  delay(2000);
}
```



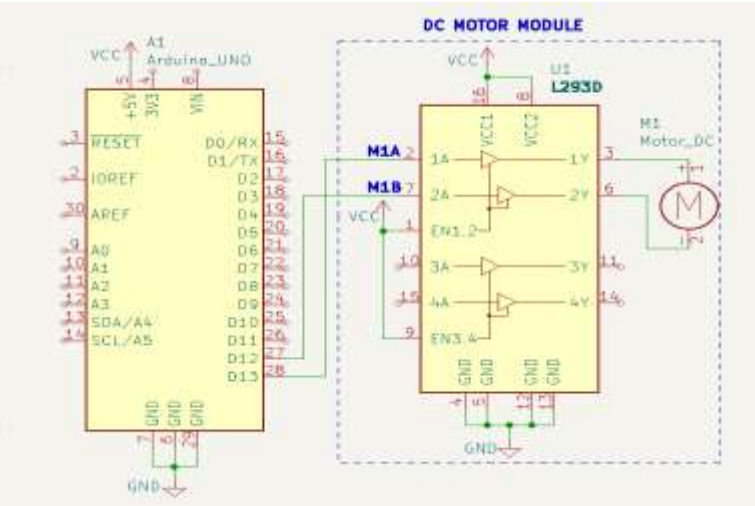
DC Motor



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.

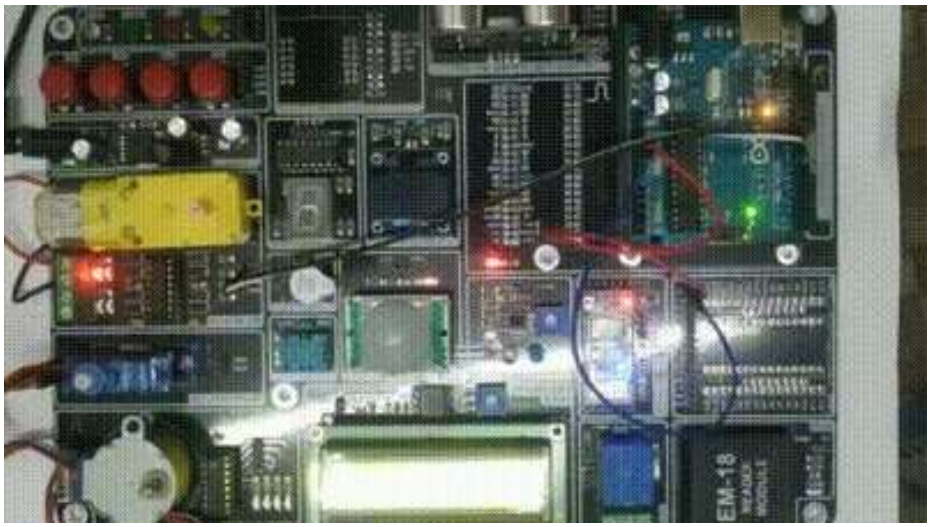
DC motor interfacing with Arduino Uno



DC Motor Code

```
const int in_1 = 8 ;
const int in_2 = 9 ;
void setup()
{
pinMode(in_1,OUTPUT) ;
pinMode(in_2,OUTPUT) ;
}

void loop()
{
//For Clock wise motion

digitalWrite(in_1,HIGH) ;
digitalWrite(in_2,LOW) ;

delay(3000) ;

digitalWrite(in_1,HIGH) ;
digitalWrite(in_2,HIGH) ;
delay(1000) ;

//For Anti Clock-wise motion
digitalWrite(in_1,LOW) ;
digitalWrite(in_2,HIGH) ;

delay(3000) ;
```

```
digitalWrite(in_1,HIGH) ;
digitalWrite(in_2,HIGH) ;
delay(1000) ;
 }
```

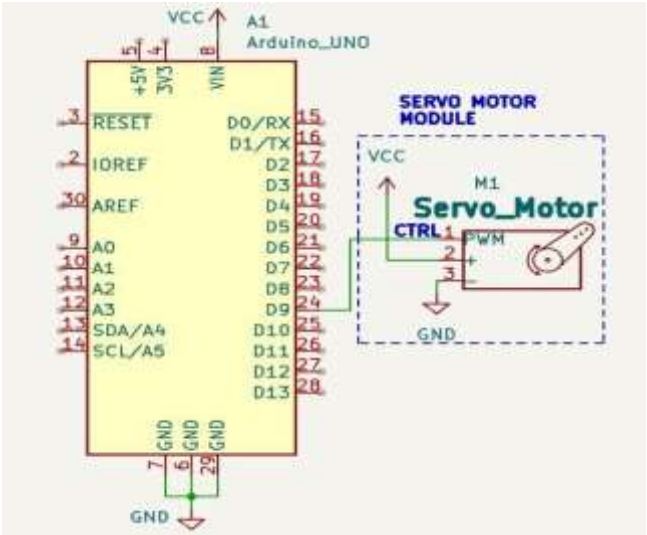


Servo Motor

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.

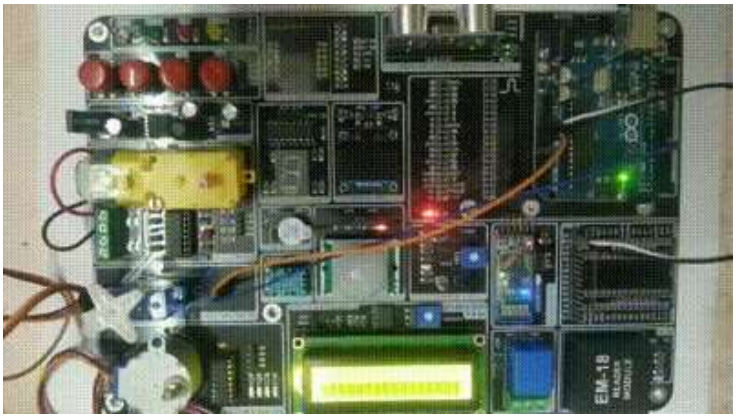
It is a closed loop system where it uses negative feedback to control motion and final position of the shaft.It is not used for continuous rotation like conventional AC/DC motors.It has rotation angle that varies from 0° to 380°.

Interfacing of stepper MOTOR with Arduino Uno



Servo Motor Code

```
//connect 7 to CTRL
#include <Servo.h>
int servoPin = 7;
Servo servo;
int angle = 0;  // servo position in degrees
void setup() {
   servo.attach(servoPin);
}
void loop() {
   // scan from 0 to 180 degrees
   for(angle = 0; angle < 180; angle++) {
      servo.write(angle);
      delay(15);
   }
   // now scan back from 180 to 0 degrees
   for(angle = 180; angle > 0; angle--) {
      servo.write(angle);
      delay(15);
   }
}
```



Stepper Motor

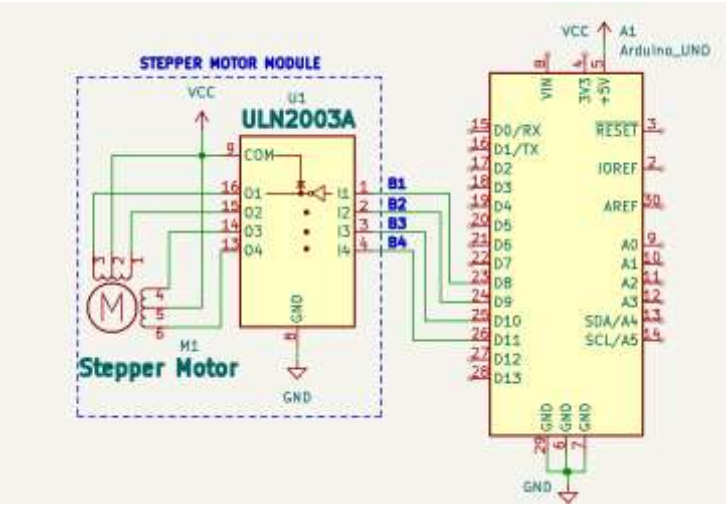
Stepper Motor is a brushless DC Motor. Control signals are applied to stepper motor to rotate it in steps.

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.

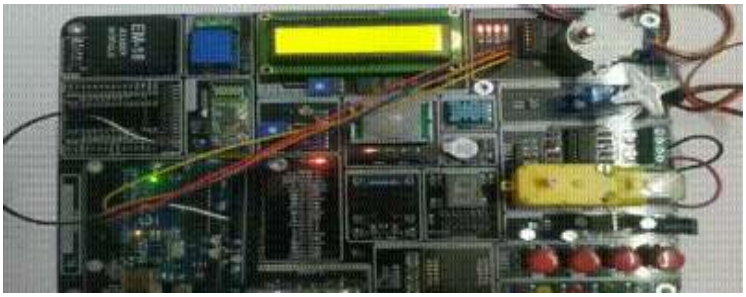
Step angle is the minimum angle that stepper motor will cover within one move/step. Number of steps required to complete one rotation depends upon step angle. Depending upon stepper motor configuration, step angle varies e.g. 0.72°, 3.8°, 3.75°, 7.5°, 35° etc.

Stepper motor interfacing with Arduino Uno



Stepper Motor Code

```
//Connect 8,9,10 & 11 to B1,B2,B3 & B4
#include <Stepper.h>
const int stepsPerRevolution = 200;  // change this to fit the number of steps per revolution
// for your motor
// initialize the stepper library on pins 8 through 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);
void setup() {
  // set the speed at 60 rpm:
  myStepper.setSpeed(60);
  // initialize the serial port:
  Serial.begin(9600);
}
void loop() {
  // step one revolution  in one direction:
  Serial.println("clockwise");
  myStepper.step(stepsPerRevolution);
  delay(500);
  // step one revolution in the other direction:
  Serial.println("counterclockwise");
  myStepper.step(-stepsPerRevolution);
  delay(500);
}
```

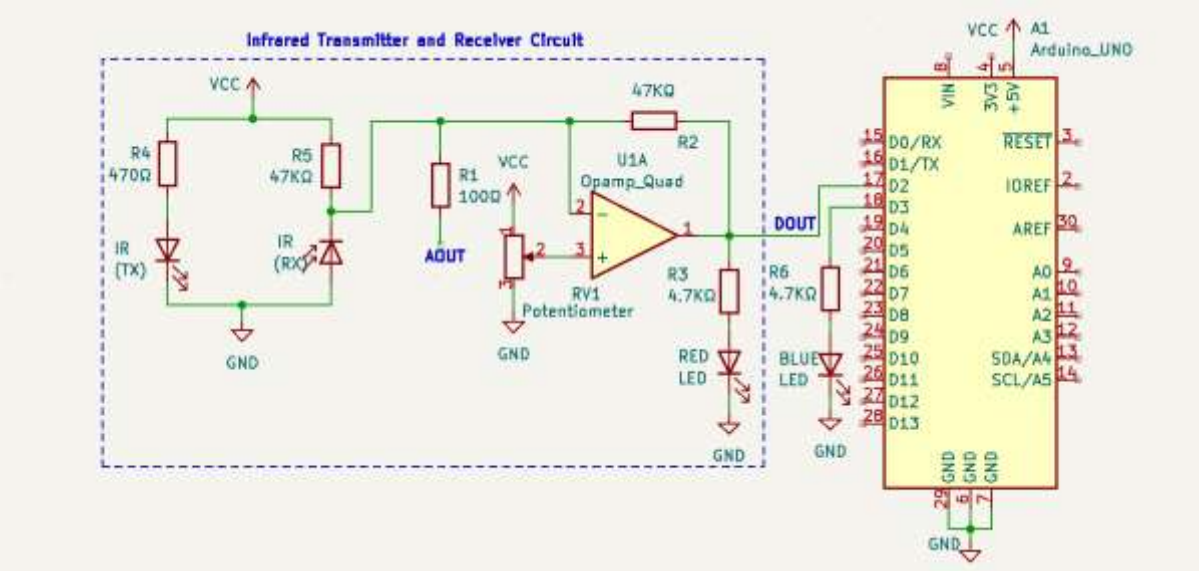


IR Sensor

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.

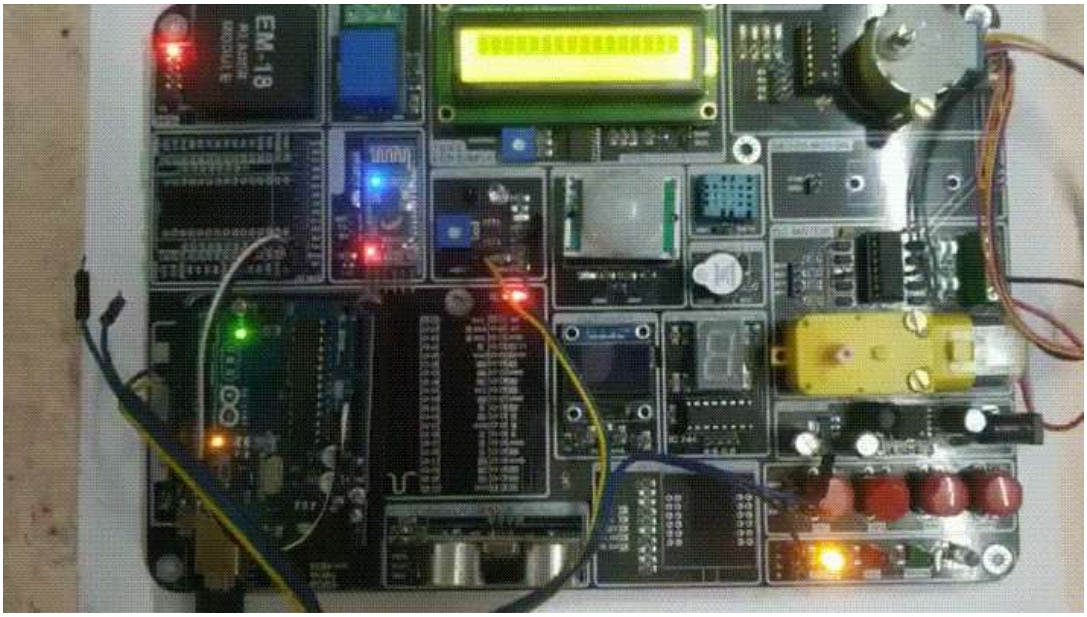
IR Sensor interfacing with Arduino Uno



IR Sensor Code

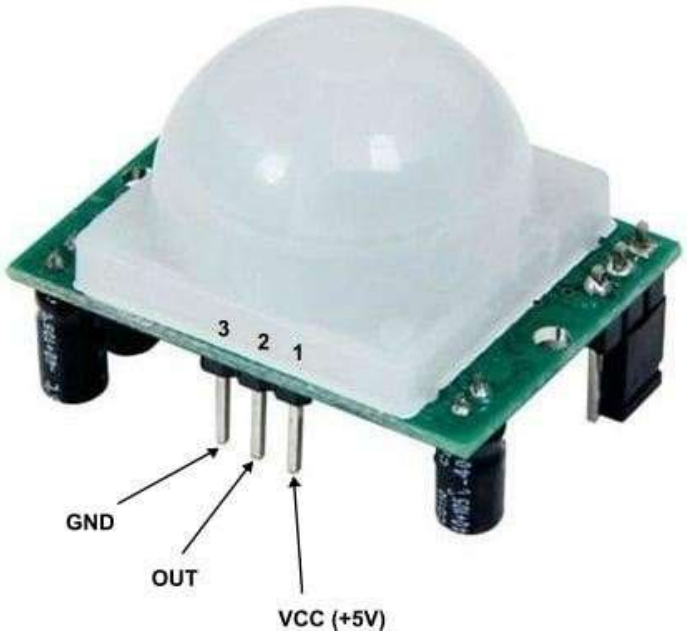
```
//connect 7 TO IR PROXI DOUT & 6 TO LED PIN
int IRSensor = 7;


int LED = 6;


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,LOW); // LED LOW
Serial.println("OBJECT DETECTED"); // print Motion Detected! on the serial monitor window
}
else
  {
    //else turn on the onboard LED
    digitalWrite(LED, HIGH); // LED High
```

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

  }

delay(500);

}
```



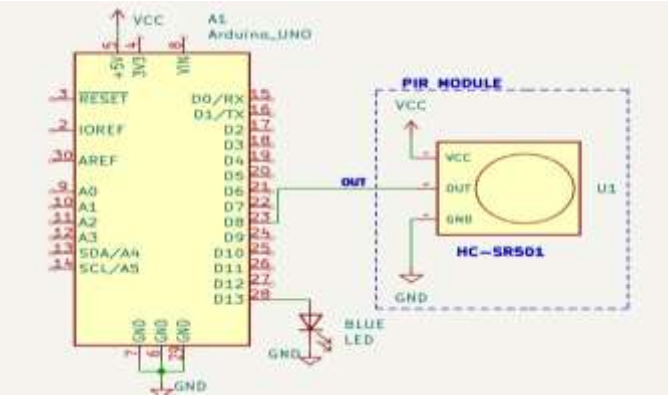
PIR Sensor

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 Uno



PIR Sensor Code

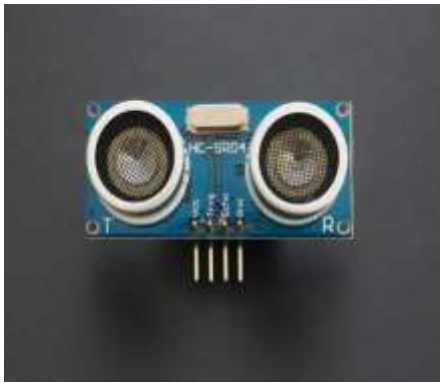
```
int led = 3;              // the pin that the LED is atteched to
int sensor = 2;           // the pin that the sensor is atteched to
int state = LOW;          // by default, no motion detected
int val = 0;              // variable to store the sensor status (value)

void setup() {
  pinMode(led, OUTPUT);     // initalize LED as an output
  pinMode(sensor, INPUT);   // initialize sensor as an input
  Serial.begin(9600);       // initialize serial
}

void loop(){
  val = digitalRead(sensor);  // read sensor value
  if (val == HIGH) {          // check if the sensor is HIGH
    digitalWrite(led, HIGH);  // turn LED ON



    if (state == LOW) {
```

```
    Serial.println("Motion detected!");
    state = HIGH;       // update variable state to HIGH
   }
  }
  else {
    digitalWrite(led, LOW); // turn LED OFF


    if (state == HIGH){
      Serial.println("Motion stopped!");
      state = LOW;       // update variable state to LOW
   }
  }
}
```

```
COM8

15:25:20.926 -> Motion detected!
15:25:22.092 -> Motion stopped!
15:25:41.514 -> Motion detected!
15:25:42.682 -> Motion stopped!
15:25:44.920 -> Motion detected!
15:25:46.086 -> Motion stopped!
15:25:48.020 -> Motion detected!
15:25:49.994 -> Motion stopped!
15:25:52.418 -> Motion detected!
15:25:53.587 -> Motion stopped!
15:25:56.710 -> Motion detected!
15:25:57.876 -> Motion stopped!
15:26:00.777 -> Motion detected!
15:26:01.946 -> Motion stopped!
```

Ultrasonic

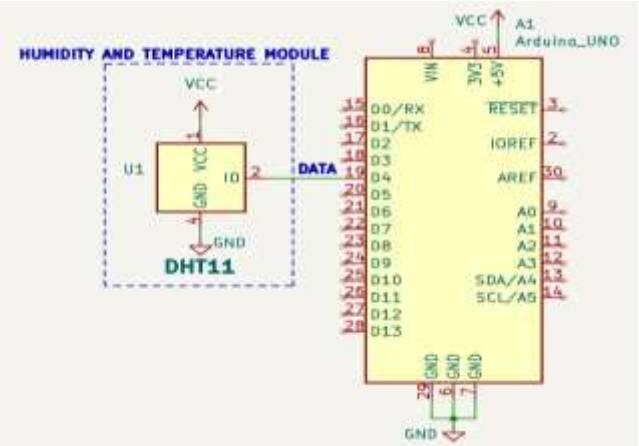
Ultrasonic Module HC-SR04 works on the principle of SONAR and RADAR systems. It can be used to determine the distance of an object in the range of 2 cm – 400 cm. An ultrasonic sensor generates high-frequency sound (ultrasound) waves. When this ultrasound hits the object, it reflects as echo which is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



Ultrasonic Code

```
const int pingPin = 7; // Trigger Pin of Ultrasonic Sensor

const int echoPin = 6; // Echo Pin of Ultrasonic Sensor


void setup() {

  Serial.begin(9600); // Starting Serial Terminal

}


void loop() {

  long duration, cm;

  pinMode(pingPin, OUTPUT);

  digitalWrite(pingPin, LOW);

  delayMicroseconds(2);

  digitalWrite(pingPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(pingPin, LOW);

  pinMode(echoPin, INPUT);

  duration = pulseIn(echoPin, HIGH);

  cm = microsecondsToCentimeters(duration);
```

```
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(100);
}
long microsecondsToCentimeters(long microseconds) {
  return microseconds / 29 / 2;
}
```

```
COM11

15:45:49.437 -> Distance: 2
15:45:49.953 -> Distance: 2
15:45:50.471 -> Distance: 2
15:45:50.944 -> Distance: 2
15:45:51.447 -> Distance: 2
15:45:51.954 -> Distance: 2
15:45:52.459 -> Distance: 2
15:45:52.967 -> Distance: 3
15:45:53.488 -> Distance: 3
15:45:54.007 -> Distance: 2
15:45:54.478 -> Distance: 2
15:45:54.997 -> Distance: 2
15:45:55.516 -> Distance: 2
15:45:55.987 -> Distance: 2
```

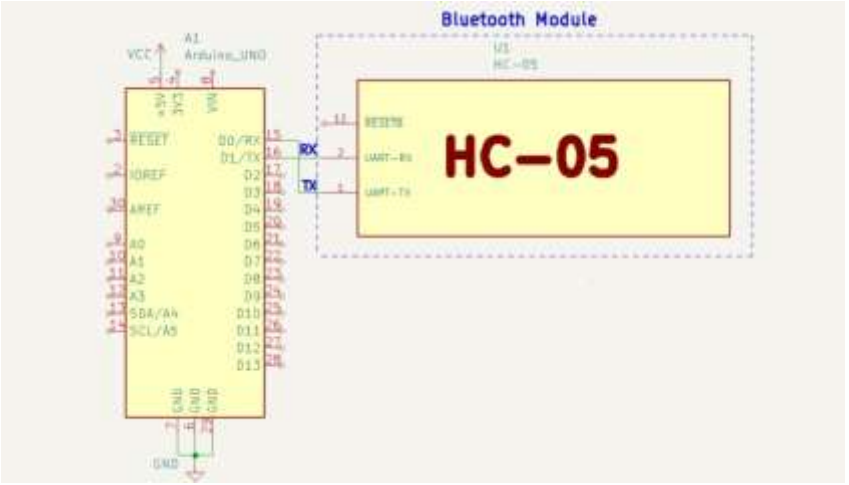
Bluetooth

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.

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.

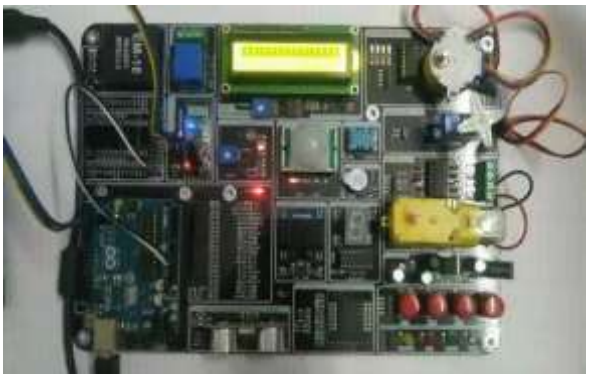
It uses serial communication to communicate with devices. It communicates with microcontroller using serial port (USART).

Interfacing of Bluetooth with Arduino Uno



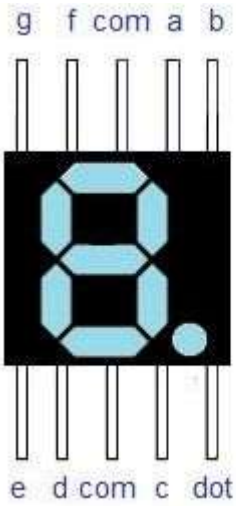
Bluetooth Code

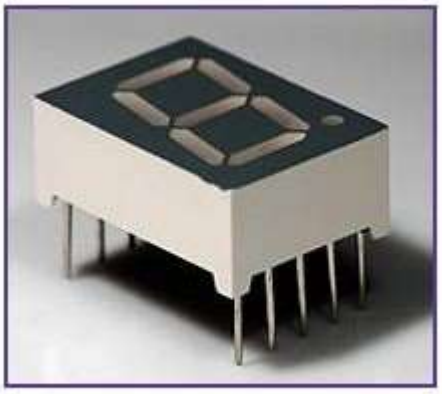
```
// Connect Tx of Bluetooh to pin 2 of uno and Rx of bluetooth to pin 3 of uno
#include<SoftwareSerial.h>

/* Create object named bt of the class SoftwareSerial */
SoftwareSerial bt(2,3); /* (Rx,Tx) */

void setup() {
  bt.begin(9600);         /* Define baud rate for software serial communication */
  Serial.begin(9600);    /* Define baud rate for serial communication */
}

void loop() {

    if (bt.available())      /* If data is available on serial port */
    {
     Serial.write(bt.read());       /* Print character received on to the serial monitor */
    }
}
```



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.



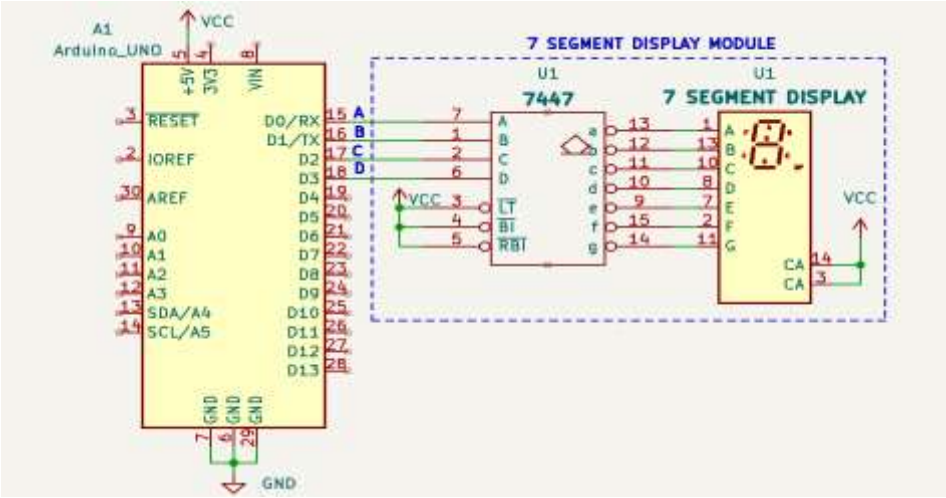


There are two types of 7 segment displays: Common Anode and Common Cathode:

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.

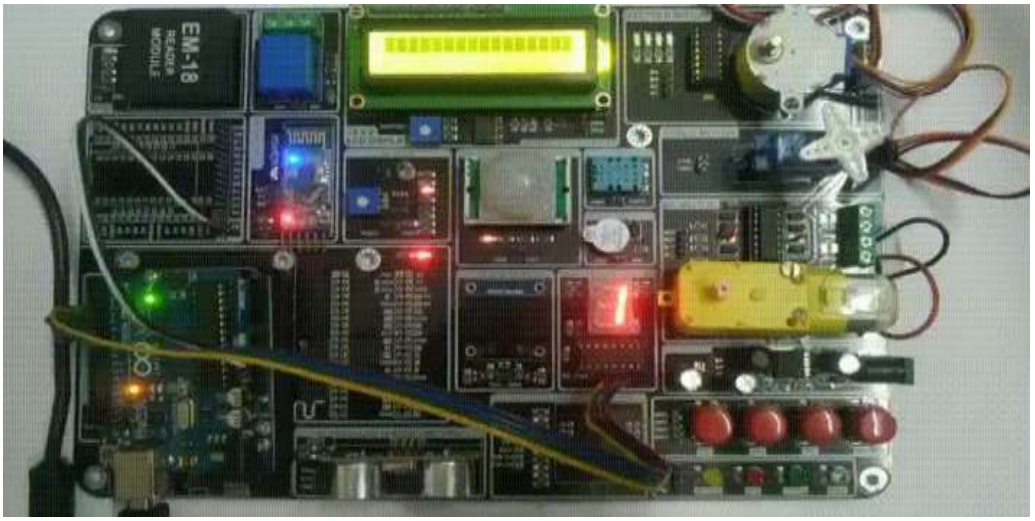
Interfacing of seven segment display with Arduino Uno



Seven Segment Code

```
const uint8_t BCD_A = 3;
const uint8_t BCD_B = 4;
const uint8_t BCD_C = 5;
const uint8_t BCD_D = 6;

const unsigned long DisplayPeriod = 1000;

void setup() {
  pinMode(BCD_A, OUTPUT);
  pinMode(BCD_B, OUTPUT);
  pinMode(BCD_C, OUTPUT);
  pinMode(BCD_D, OUTPUT);
}
```

```
void loop() {
  static uint8_t count = 0; // display count
  static unsigned long previousDisplayTime = 0;
  unsigned long currentTime = millis();



  if (currentTime - previousDisplayTime >= DisplayPeriod) { // check if time to update display
    displayWrite(count);  // update display
    count++;  // increase counter
    if (count == 10) count = 0; // reset to 0 if count exceeds 9
    previousDisplayTime = currentTime;
  }
}
void displayWrite(uint8_t value) {
  digitalWrite(BCD_A, bitRead(value, 0));  // BCD LSB
  digitalWrite(BCD_B, bitRead(value, 1));
  digitalWrite(BCD_C, bitRead(value, 2));
  digitalWrite(BCD_D, bitRead(value, 3));  // BCD MSB
}
```



RFID (EM18)

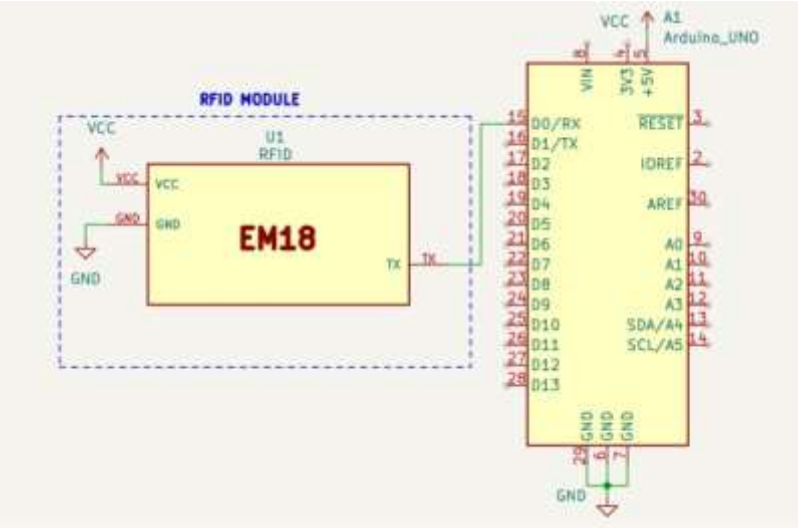
Radio frequency Identification i.e. RFID is a wireless identification technology that uses radio waves to identify the presence of RFID tags.

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 Uno



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);
  }
}
```