SRM UNIVERSITY
DELHI-NCR, SONEPAT

MHRD'S
INNOVATION CELL
(GOVERNMENT OF INDIA)

INSTITUTION'S
INNOVATION
COUNCIL
(Ministry of HRD Initiative)

CENTER FOR INNOVATION INCUBATION & ENTREPRENEURSHIP

(C I I E)

ARDUINO WORKSHOP

What is Arduino?

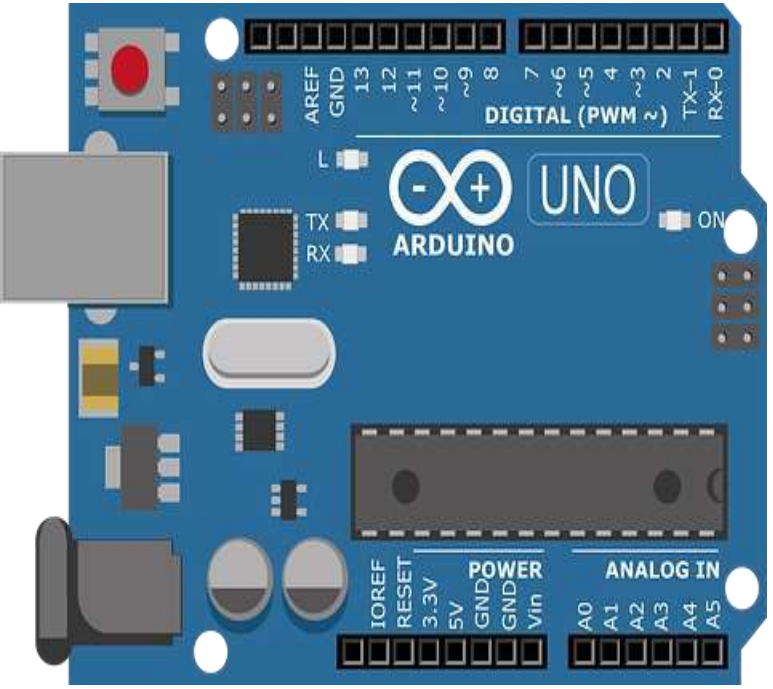
Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

Why Arduino ?

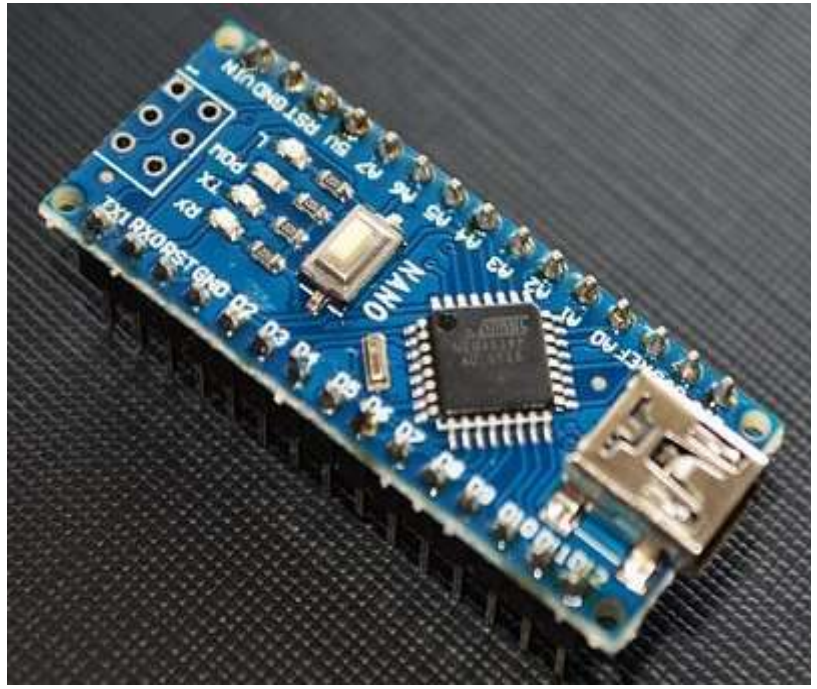
- Inexpensive
- Cross-platform
- Simple, clear programming environment
- Open source and extensible hardware

ARDUINO

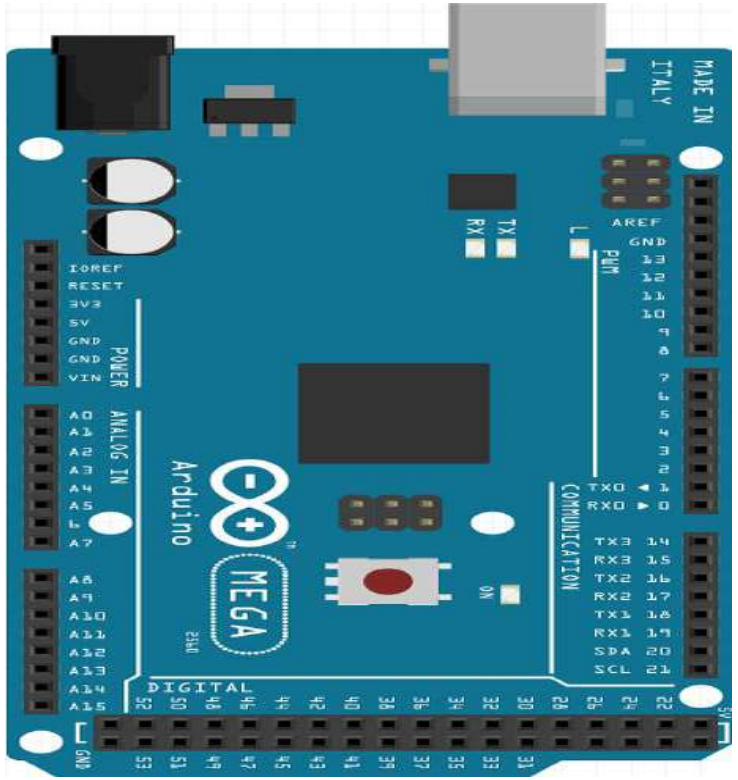
ARDUINO UNO



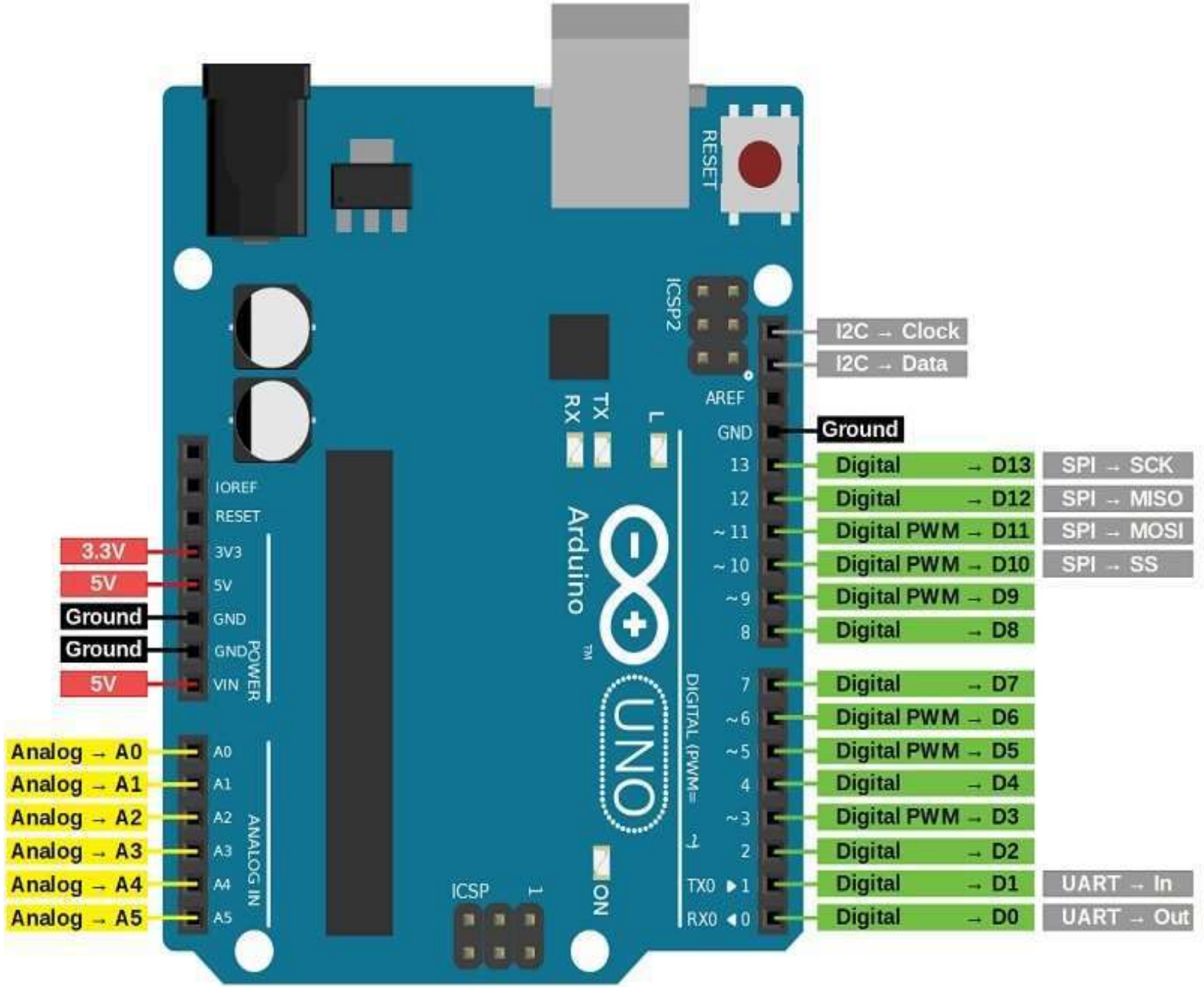
ARDUINO NANO



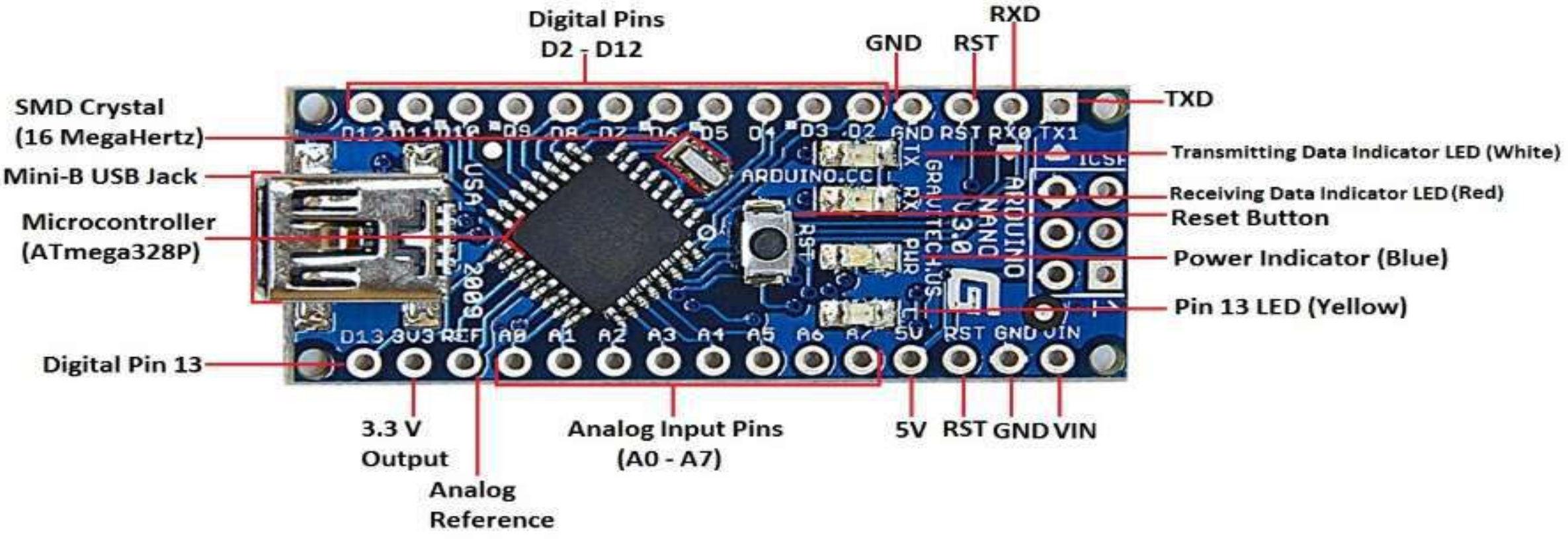
ARDUINO MEGA



ARDUINO UNO

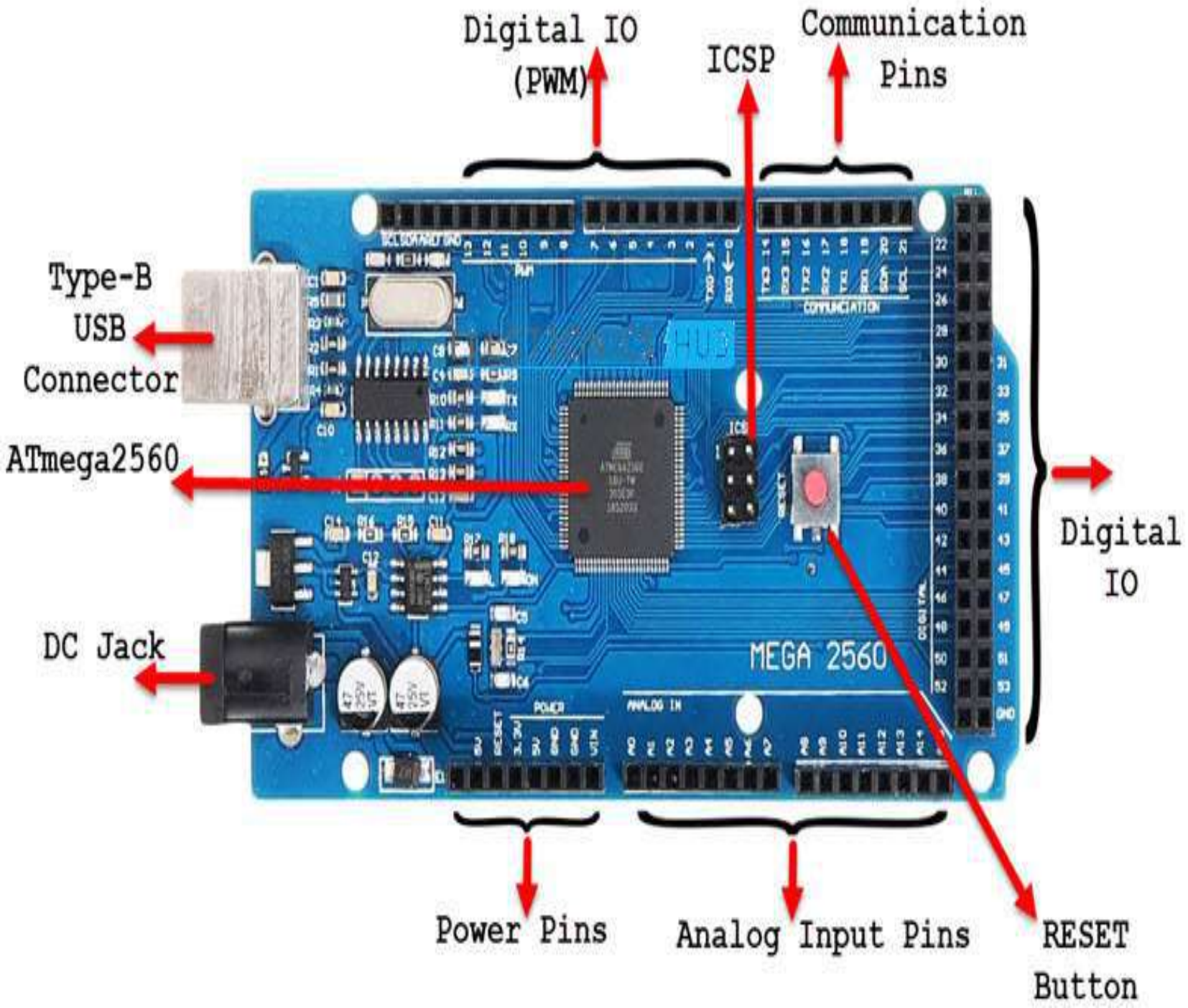

RESET
ICSP2
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TX0 1
RX0 0
DIGITAL (PWM=~)
I2C → Clock
I2C → Data
Ground
Digital → D13
SPI → SCK
Digital → D12
SPI → MISO
Digital PWM → D11
SPI → MOSI
Digital PWM → D10
SPI → SS
Digital PWM → D9
Digital → D8
Digital → D7
Digital PWM → D6
Digital PWM → D5
Digital → D4
Digital PWM → D3
Digital → D2
Digital → D1
UART → In
Digital → D0
UART → Out
TX
RX
L
ON
ICSP
Arduino
UNO
IOREF
RESET
3.3V
3V3
5V
5V
Ground
GND
Ground
GND
5V
VIN
POWER
Analog → A0
Analog → A1
Analog → A2
Analog → A3
Analog → A4
Analog → A5
A0
A1
A2
A3
A4
A5
ANALOG IN

ARDUINO NANO



Arduino Nano V3.0 Pinout

ARDUINO MEGA



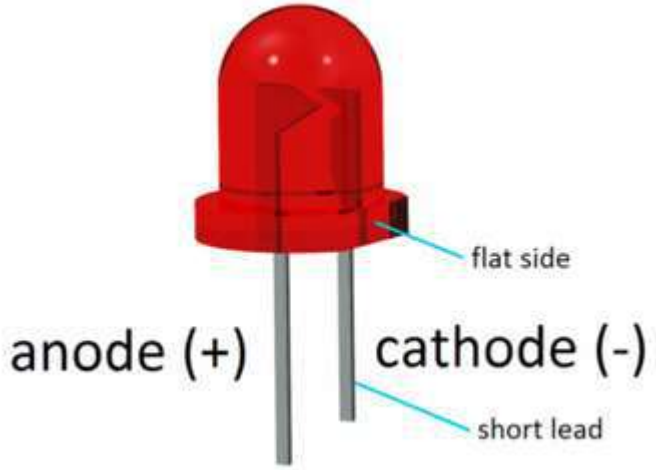
MALE TO MALE

FEMALE TO FEMALE

FEMALE TO MALE

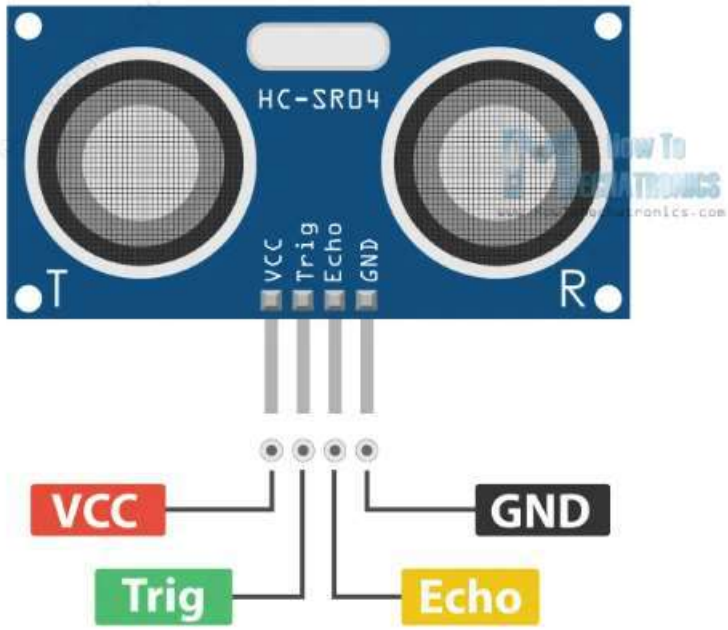
JUMPER WIRES

CABLE FOR ARDUINO UNO/MEGA (USB A TO B): 50 CM



LED

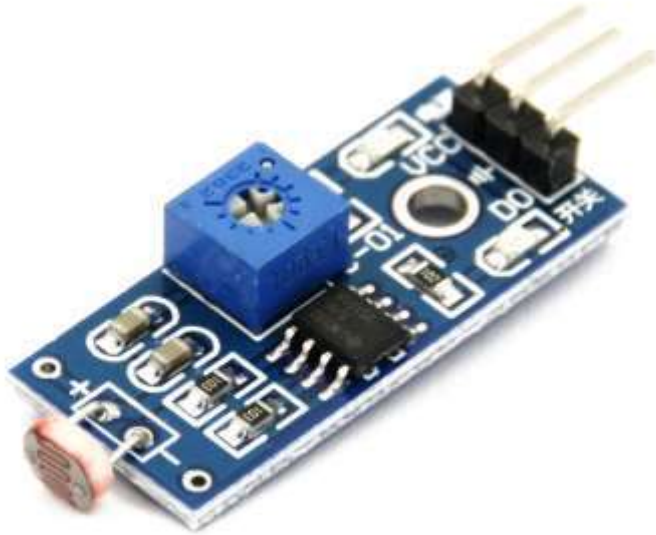
ULTRASONIC SENSOR

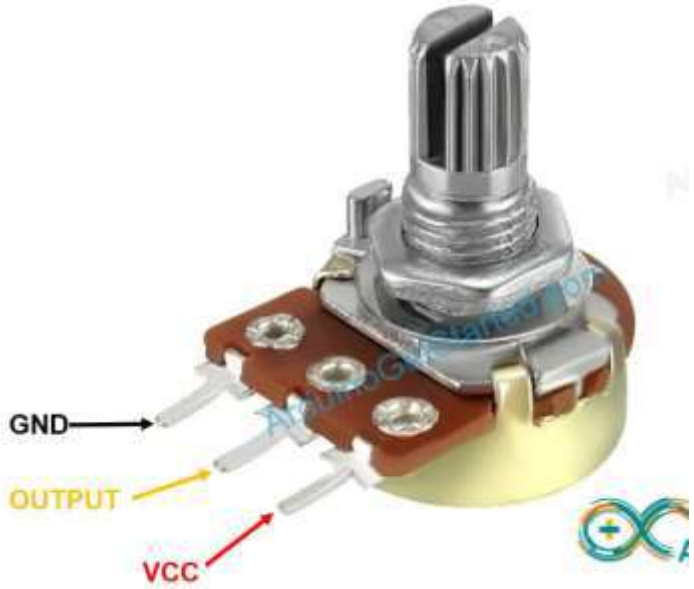


RGB

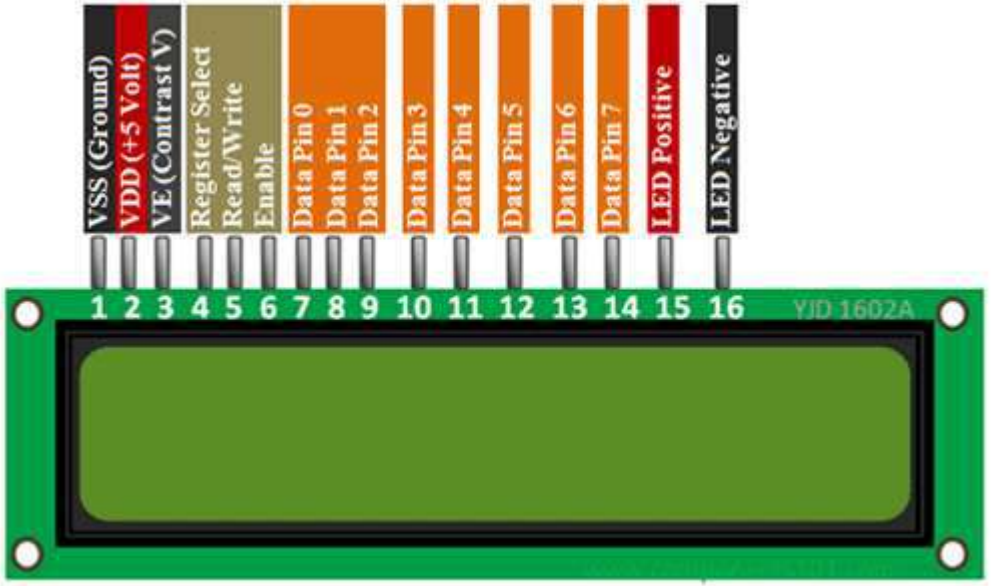
IR SENSOR

LDR SENSOR



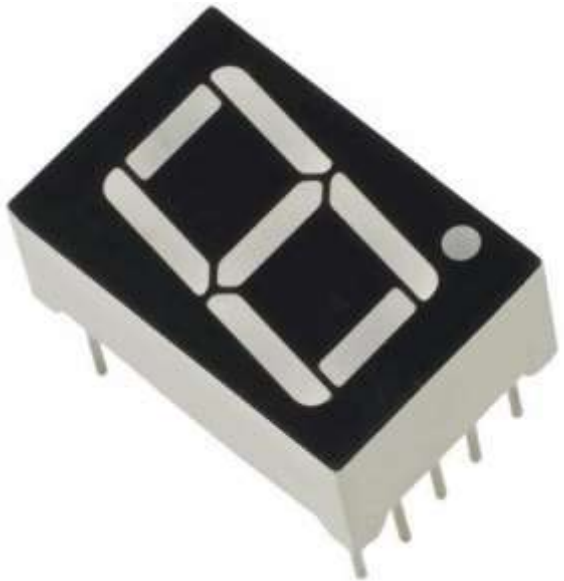


POTENTIOMETER



16 – PIN LCD

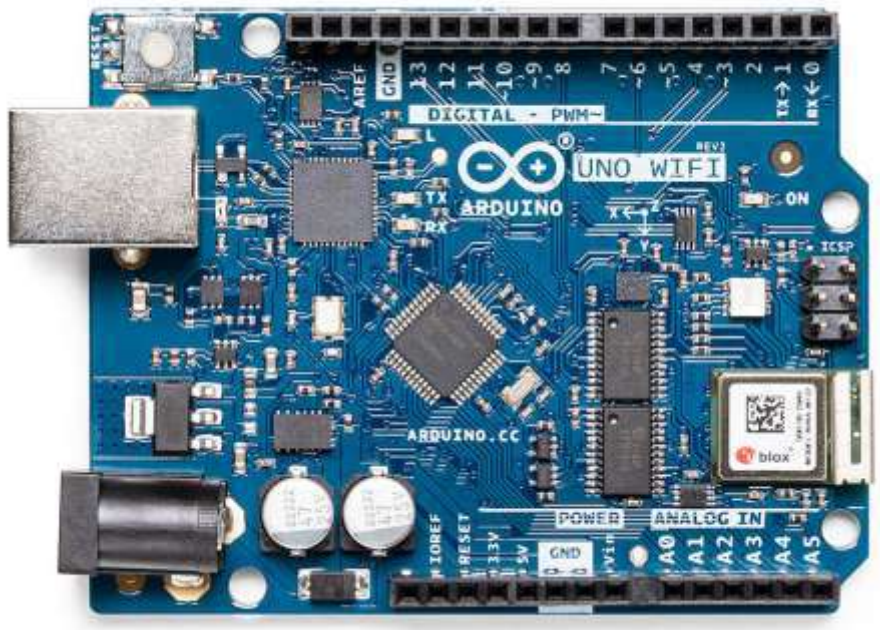
7 – SEGMENT DISPLAY



RFID

GAS
SENSOR

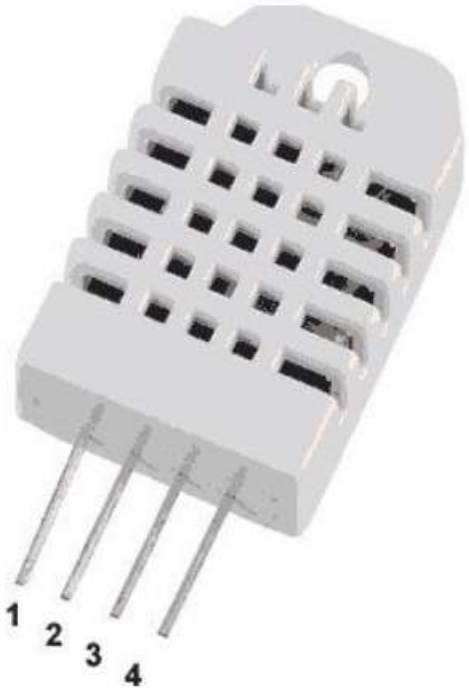
BUZZER



WIFI

BLUETOOTH

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



DHT
SENSOR

GEO
SENSOR

GYRO
SENSOR

MOTOR DRIVER

STEPPER
MOTOR

DC MOTOR

RELAYS

SERVO
MOTOR

Downloads

Arduino IDE 2.0.3

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the **Arduino IDE 2.0 documentation**.

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on **GitHub**.

DOWNLOAD OPTIONS

Windows Win 10 and newer, 64 bits
Windows MSI installer
Windows ZIP file

Linux AppImage 64 bits (X86-64)
Linux ZIP file 64 bits (X86-64)

macOS Intel, 10.14: "Mojave" or newer, 64 bits
macOS Apple Silicon, 11: "Big Sur" or newer, 64 bits

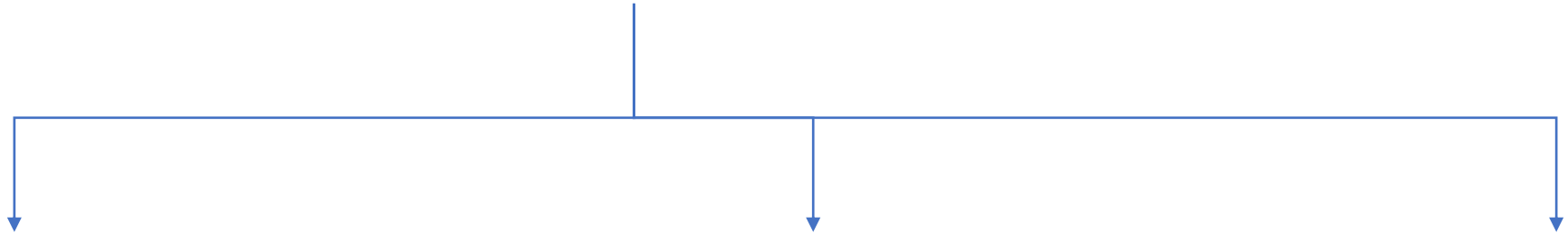
Release Notes

LANGUAGE FOR ARDUINO

C/C++

- General – purpose programming language.
- It has imperative, object-oriented and generic programming features.
- HIGH level language.

DATATYPES IN C/C++



Primitive Data Types: These data types are built-in or predefined data types and can be used directly by the user to declare variables.

- Integer
- Character
- Boolean
- Floating Point
- Double Floating Point
- Valueless or Void
- Wide Character

Derived Data Types: These data types are derived from the primitive or built-in datatypes are referred to as Derived Data Types.

- Function
- Array
- Pointer
- Reference

Abstract or User-Defined Data Types: These are defined by the user itself.

- Class
- Structure
- Union
- Enumeration

What is a Function?

A function is a self-contained block of statements that perform a coherent task of some kind. Every C program can be thought of as a collection of these functions. As we noted earlier, using a function is something like hiring a person to do a specific job for you. Sometimes the interaction with this person is very simple; sometimes it's complex.

Suppose you have a task that is always performed exactly in the same way—say a bimonthly servicing of your motorbike. When you want it to be done, you go to the service station and say, "It's time, do it now". You don't need to give instructions, because the mechanic knows his job. You don't need to be told how the job is done. You assume the bike would be serviced in the usual way, the mechanic does it and that's that. Let us now look at a simple C function that operates in much the same way as the mechanic. Actually, we will be looking at two things—a function that calls or activates the function and the function itself.

```
# include <stdio.h>
void message( ) ;  /* function prototype declaration */
int main( )
{
    message( ) ;  /* function call */
    printf ( "Cry, and you stop the monotony!\n" ) ;
    return 0 ;
}
void message( )  /* function definition */
{
    printf ( "Smile, and the world smiles with you...\n" ) ;
}
```

```
Smile, and the world smiles with you...
Cry, and you stop the monotony!
```

Here, we have defined two functions—main( ) and message( ). In fact, we have used the word message at three places in the program. Let us understand the meaning of each. The first is the function prototype declaration and is written as:

```
void message( ) ;
```

This prototype declaration indicates that message( ) is a function which after completing its execution does not return any value. This ‘does not return any value’ is indicated using the keyword void. It is necessary to mention the prototype of every function that we intend to define in the program. The second usage of message is…

```
void message( )
{
    printf ( "Smile, and the world smiles with you...\n" ) ;
}
```

This is the function definition. In this definition right now we are having only printf( ), but we can also use if, for, while, switch, etc., within this function definition. The third usage is…

```
message( ) ;
```

Here the function message( ) is being called by main( ). What do we mean when we say that main( ) ‘calls’ the function message( )? We mean that the control passes to the function message( ). The activity of main( ) is temporarily suspended; it falls asleep while the message( ) function wakes up and goes to work. When the message( ) function runs out of statements to execute, the control returns to main( ), which comes to life again and begins executing its code at the exact point where it left off. Thus, main( ) becomes the ‘calling’ function, whereas message( ) becomes the ‘called’ function

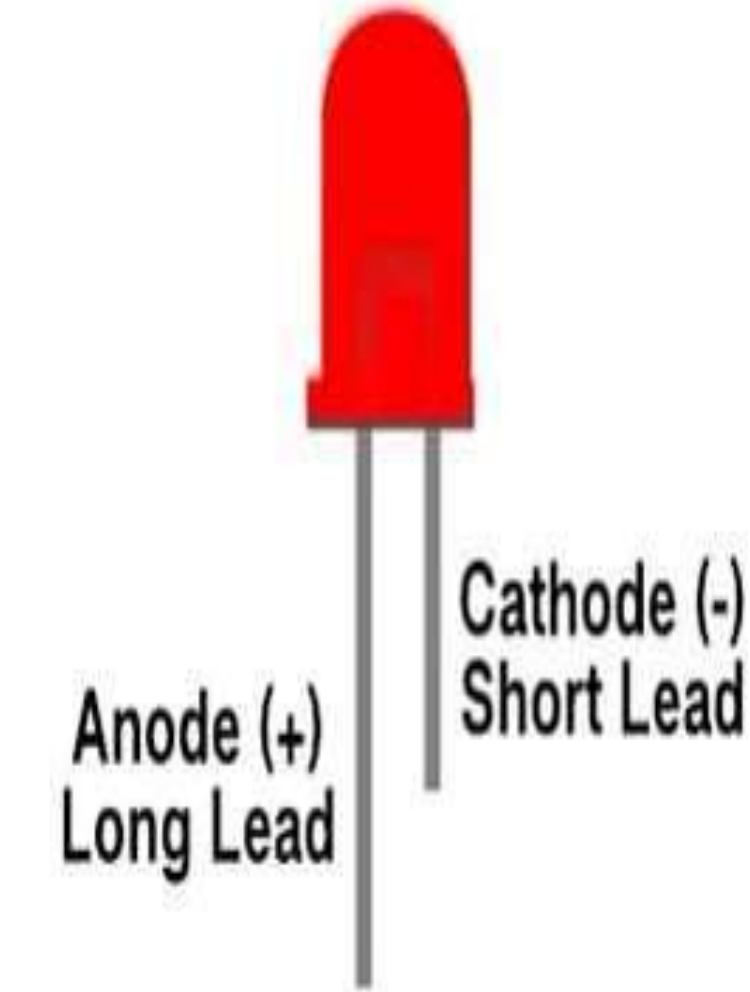
KEYWORD IN ARDUINO

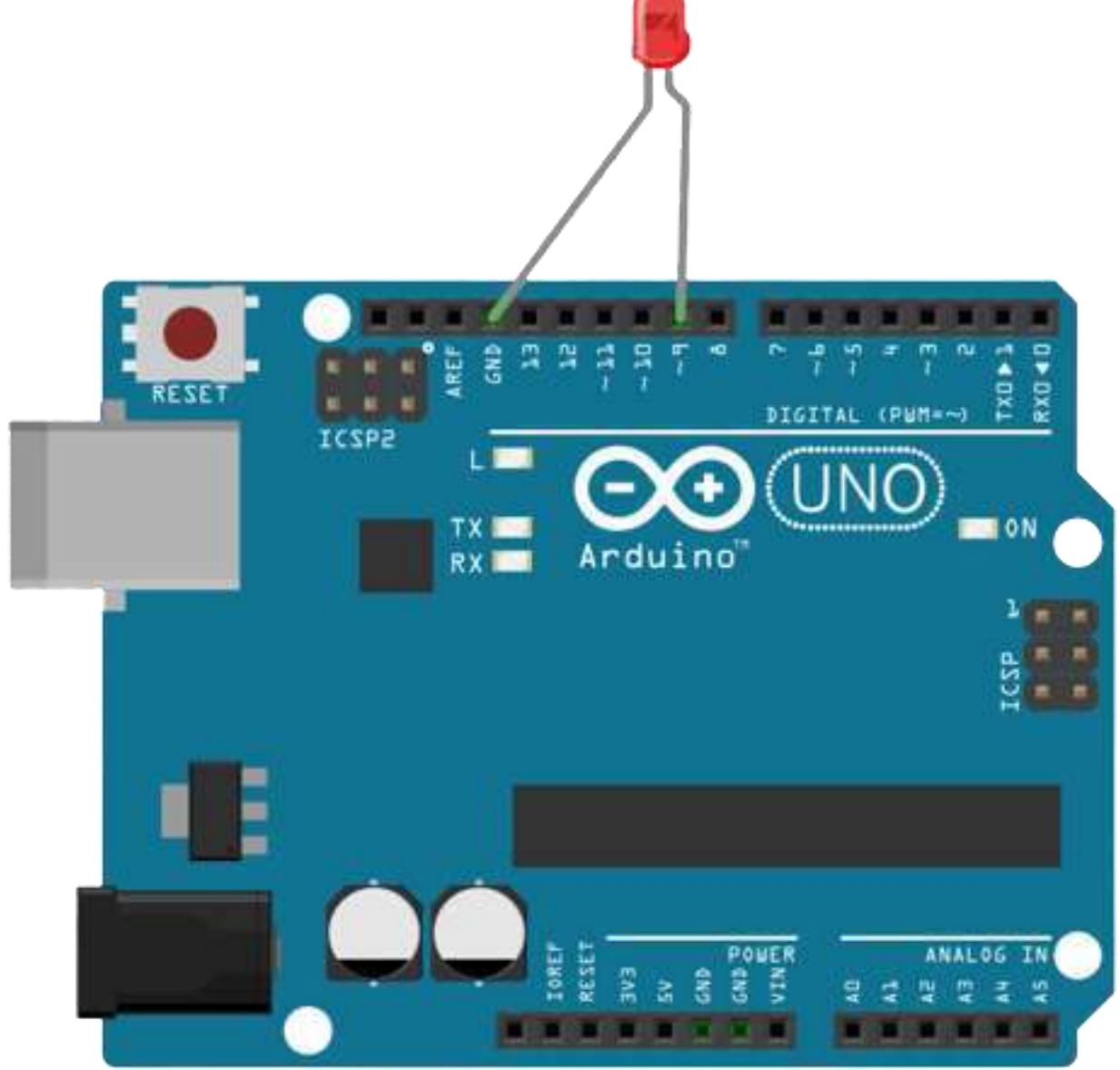
- void: This keyword is used to specify that a function does not return a value.
- setup(): This is a built-in function in Arduino that is run once at the start of the program. It is used to initialize the board and set up any necessary configurations.
- loop(): This is a built-in function in Arduino that is run repeatedly after the setup() function has completed. It contains the main code of the program.
- pinMode(): This function is used to set the mode of a pin as either input or output.
- digitalWrite(): This function is used to write a digital value (HIGH or LOW) to a pin that has been configured as an output.
- analogWrite(): This function is used to write an analog value (0-255) to a pin that supports pulse width modulation (PWM).

- digitalRead(): This function is used to read the value of a digital pin that has been configured as an input.
- analogRead(): This function is used to read the value of an analog pin.
- delay(): This function is used to pause the program for a specified amount of time in milliseconds.
- Serial.begin(): This function is used to initialize the serial communication at a specified baud rate.
- Serial.print(): This function is used to output data to the serial monitor.
- Serial.println(): This function is used to output data to the serial monitor and add a newline character at the end.

LED

Light-emitting diode (LED) is a widely used standard source of light in electrical equipment

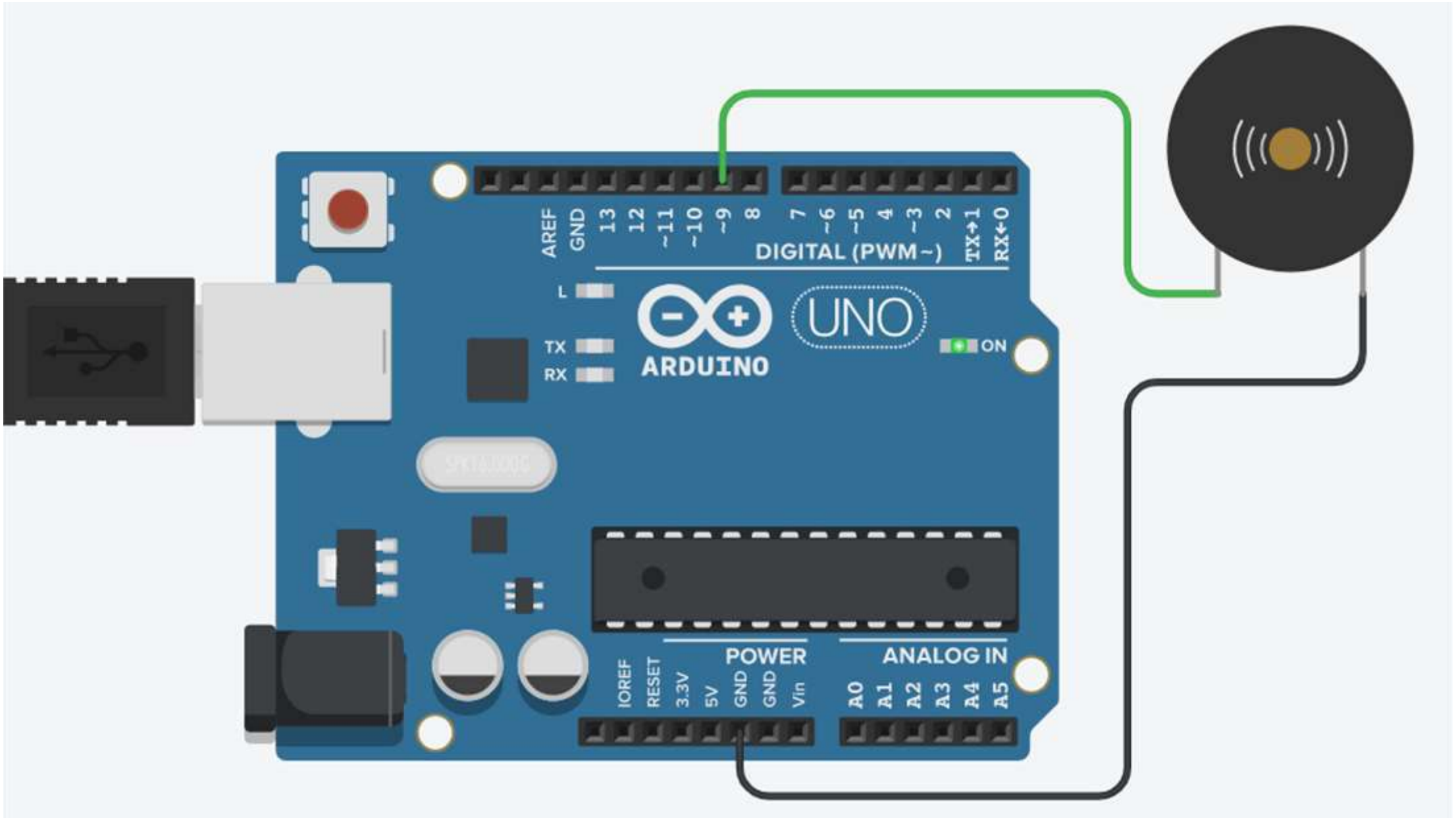

Anode (+)
Long Lead
Cathode (-)
Short Lead


RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5

```
int ledPin= 13;

void setup() {
  // put your setup code here, to run once:
  pinMode(ledPin, OUTPUT);

}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(ledPin, HIGH);

  delay(500);//in microseconds
  digitalWrite(ledPin, LOW);

  delay(1000);
}
```

BUZZER

An audio signaling device like a beeper or buzzer may be electromechanical or mechanical type. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.


AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
L
TX
RX
ARDUINO
UNO
ON
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5

```
int buzzerPin= 12;
void setup() {
  // put your setup code here, to run once:

  pinMode(buzzerPin, OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  digitalWrite(buzzerPin, HIGH);
  delay(500);//in microseconds

  digitalWrite(buzzerPin, LOW);
  delay(1000);
}
```


THANK YOU
FOR JOINING US