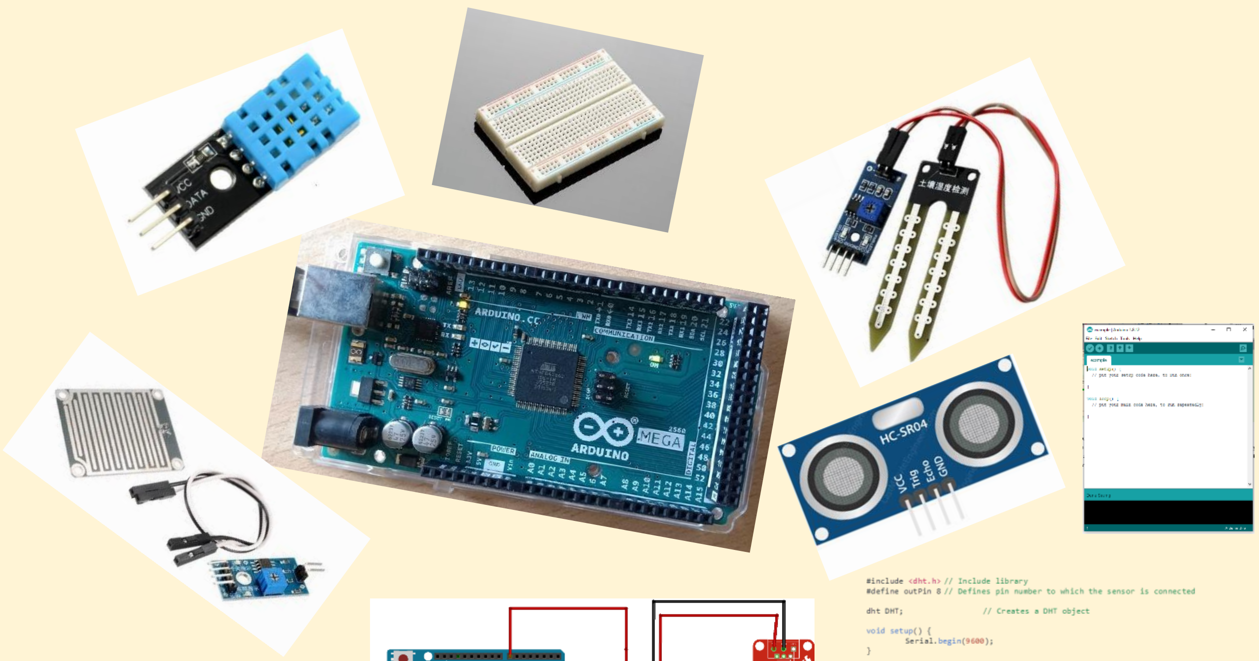
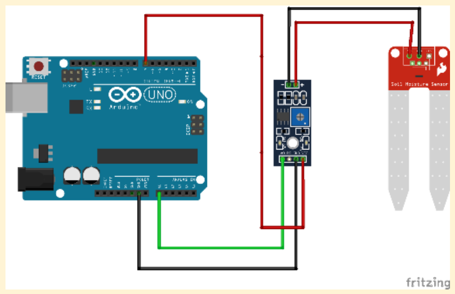
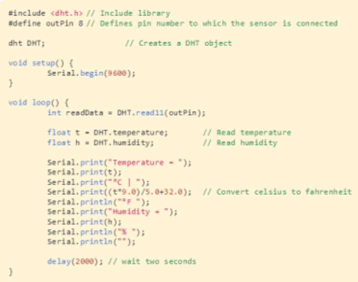
# Arduino Tutorial: Use of Sensors with Arduino for Agricultural Projects

An E-learning lesson

*(PME Publication no. ICARNEH-ML-EPUB-2022-49)*







```
#include <dht.h> // Include library
#define outPin 8 // Defines pin number to which the sensor is connected

dht DHT;                // Creates a DHT object

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

void loop() {
        int readData = DHT.read11(outPin);

        float t = DHT.temperature;        // Read temperature
        float h = DHT.humidity;           // Read humidity

        Serial.print("Temperature = ");
        Serial.print(t);
        Serial.print("°C | ");
        Serial.print((t*9.0)/5.0+32.0);  // Convert celsius to fahrenheit
        Serial.println("°F ");
        Serial.print("Humidity = ");
        Serial.print(h);
        Serial.println("% ");
        Serial.println("");

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

Prepared by:

Naseeb Singh, Debasish Chakraborty, Ajaykumar K., H. Dayananda Singh and S. Hazarika

Division of System Research and Engineering
ICAR Research Complex for NEH Region
Umiam-793103, Meghalaya (India)

# Arduino Tutorial: Use of Sensors with Arduino for Agricultural Projects

An E-learning lesson

***(PME Publication no. ICARNEH-ML-EPUB-2022-49)***

Prepared by:

Naseeb Singh
Debasish Chakraborty
Ajaykumar K.
H. Dayananda Singh
S. Hazarika

Division of System Research and Engineering
ICAR Research Complex for NEH Region
Umiam-793103, Meghalaya (India)

**Citation:** Singh, N., Chakraborty, D., Ajaykumar K., Singh, H. D., Hazarika, S., 2022. Arduino Tutorial: Use of Sensors with Arduino for Agricultural Projects. ICAR RC for NEH Region, Umiam, Meghalaya. PME Publication no. ICARNEH-ML-EPUB-2022-49.

# Table of Contents

# Chapter: 1 What is Arduino?

## 1.1 What is Arduino?

Arduino is an open-source platform that may be used to create electronic creations. Arduino is made up of two parts: a physical programmable circuit board (also known as a microcontroller) and software (called an Integrated Development Environment (IDE)) that runs on your computer and is used to create and upload computer code to the physical board. Fig. *1* shows the Arduino UNO board.

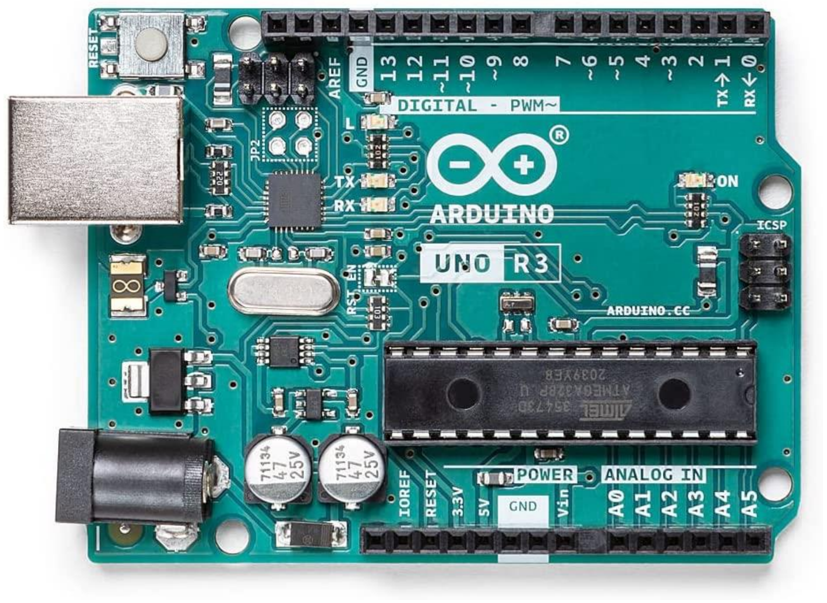


Fig. 1. Arduino UNO board.

## 1.2 Arduino Basic Components

On a single circuit board, Arduinos combine multiple distinct elements and interfaces. The design has evolved over time, and some iterations include additional components. However, the following pieces are likely to be seen on a basic board:

1. Several pins are available to link the Arduino to various components that you'll want to use. These pins come in two varieties:
    - Digital pins, which can read and write a single state, on or off. Most Arduinos have 14 digital I/O pins.
    - Analog pins, which can read a range of values, and are useful for more fine-grained control. Most Arduinos have six of these analog pins.

2. A power connector that supplies power to both the device and a low voltage that can power connected components such as LEDs and other sensors, as long as their power requirements are modest. An AC adaptor or a small battery can be connected to the power connector.
3. The Arduino's main component is a microcontroller, which allows you to programme it to perform commands and make decisions based on numerous inputs. The specific chip used in an Arduino varies widely depending on the model, although it's commonly an Atmel controller, such as an ATmega8, ATmega168, ATmega328, ATmega1280, or ATmega2560. The variations between these chips are minor, but the most noticeable distinction for a novice will be the differing amounts of onboard memory.
4. A serial connector, which is implemented on most newer boards as a standard USB port. This connector allows your computer to communicate with the board as well as load new programmes onto the device. Arduinos can frequently be powered via the USB port, eliminating the need for a separate power connection.
5. A variety of other minor components, like as an oscillator and/or a voltage regulator, contribute vital capabilities to the board, but you rarely deal with them; simply be aware that they exist.

# Chapter: 2 Components and pins description

## 2.1 Arduino Components

Fig. *2* shows the different components of a typical Arduino UNO board.

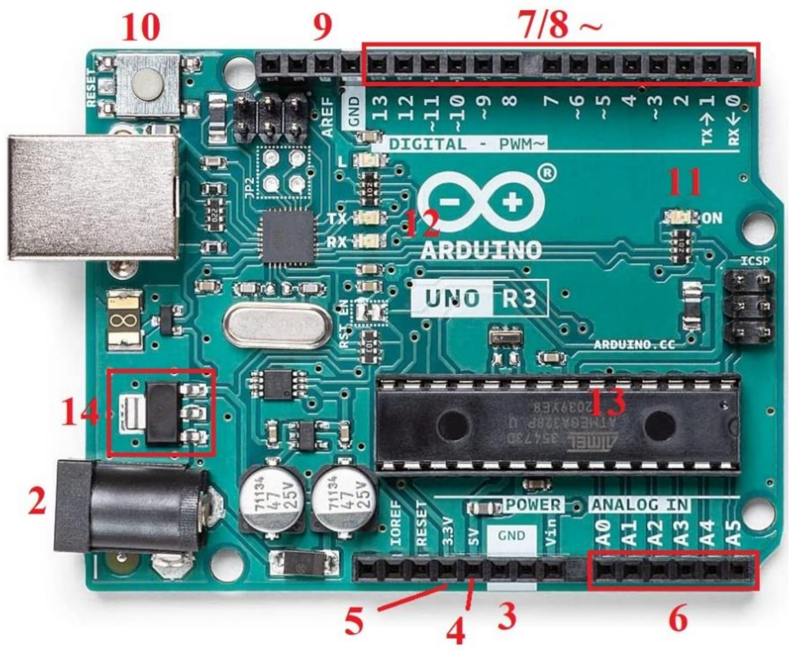


Fig. 2. Different components of a typical Arduino board.

## 2.2 Power (USB / Barrel Jack)

Every Arduino board needs a way to be connected to a power source. The Arduino UNO can be powered from a USB cable or a wall power supply that is terminated in a barrel jack. In Fig. 2, the USB connection is labeled as 1 and the barrel jack is labeled as 2.

*CAUTION: Most Arduino models require a voltage of between 6 and 12 volts. If you use a power supply with a voltage higher than 20 volts, you will overload (and thus destroy) your Arduino.*

## 2.3 Pins (5V, 3.3V, GND, Analog, Digital, PWM, AREF)

The pins on your Arduino are where you'll join wires to create a circuit. The Arduino contains a variety of pins, each of which is identified on the board and has a specific purpose.

### 2.3.1 GND (3)

Short for ‘Ground’. On the Arduino, there are multiple GND pins that can be utilized to ground the circuit.

### 2.3.2 5V (4) & 3.3V (5)

The 5V pin provides 5 volts of electricity, while the 3.3V pin provides 3.3 volts. Most of the Arduino's simple components are powered by 5 or 3.3 volts.

### 2.3.3 Analog (6)

The Analog pins are located under the 'Analog In' label (A0 through A5 on the UNO). These pins can read an analogue sensor's signal (such as a temperature sensor) and convert it to a digital value.

### 2.3.4 Digital (7)

Across from the analog, pins are the digital pins (0 through 13 on the UNO). These pins can be used for both digital input (like telling if a button is pushed) and digital output (like powering an LED).

### 2.3.5 PWM (8)

Some of the digital pins may have a tilde (~) next to them (3, 5, 6, 9, 10, and 11 on the UNO). These pins can be utilized for Pulse-Width Modulation as well as conventional digital pins (PWM). These pins can be used to replicate analogue output, such as fading an LED on and off.

### 2.3.6 AREF (9)

AREF stands for Analog Reference and is not commonly used. It is sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

## 2.4 Reset Button

There is a reset button on the Arduino (10). By pressing it, the reset pin will be briefly connected to the ground, and any code put on the Arduino will be restarted. If your code doesn't repeat yet you want to test it several times, this can be quite handy.

## 2.5 Power LED Indicator

Just beneath and to the right of the word “UNO” on your circuit board, there’s a tiny LED next to the word ‘ON’ (11). This LED should light up whenever you plug your Arduino into a power source. If this light doesn’t turn on, there’s a good chance something is wrong.

## 2.6 TX RX LEDs

TX is short for transmit while RX is short for receive. In electronics, these markings are commonly used to identify the pins responsible for serial transmission. TX and RX appear on the Arduino UNO in two places: once by digital pins 0 and 1, and again next to the TX and RX indicator LEDs (12). When Arduino receives or transmits data, these LEDs will provide some great visual clues.

## 2.7 Main IC

The black component with all of the metal legs is an Integrated Circuit, or IC (13), which serves as Arduino's brain. The main IC in the Arduino varies slightly depending on the board type; however, it is usually from the ATMEL ATmega line of ICs. Before loading a new programme from the Arduino software, it's vital to know the IC type (along with your board type). This information is often written on the top side of the IC.

## 2.8 Voltage Regulator

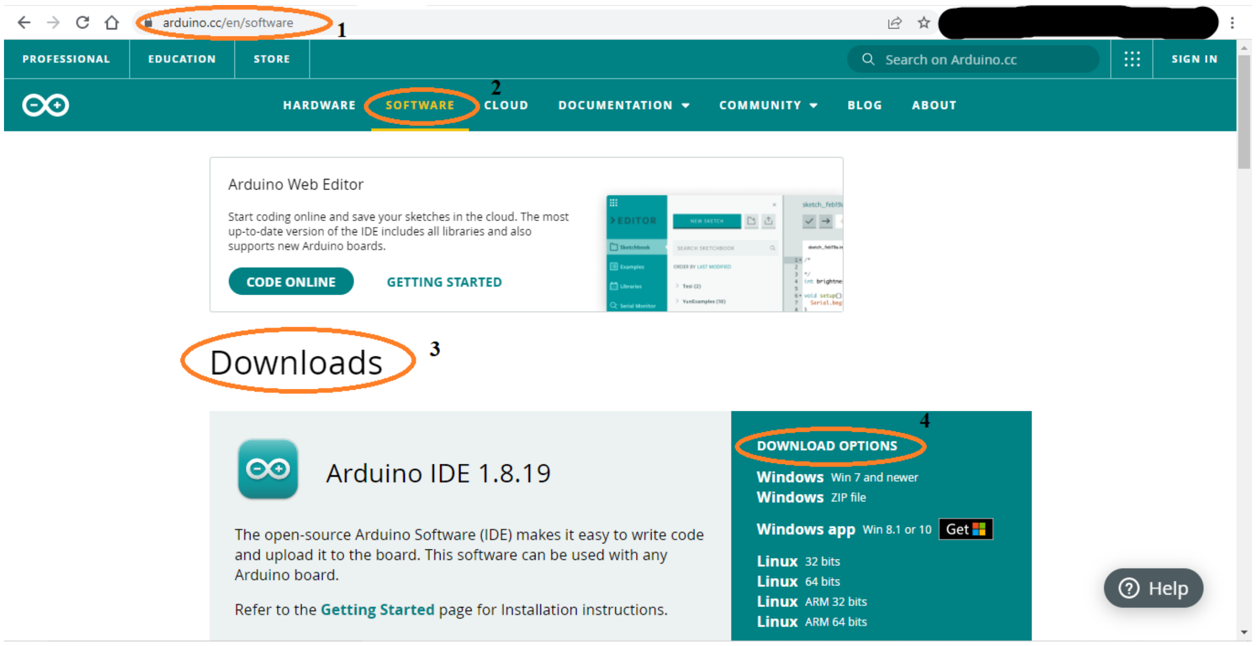
The voltage regulator (14) regulates the amount of voltage supplied to the Arduino board and rejects any excess voltage that could damage the circuit. Of course, there are some limitations, so don't connect Arduino to more than 20 volts.

# Chapter: 3 Installing the Arduino IDE

A text editor for writing code, a message box, a text console, a toolbar with buttons for common operations, and a series of menus are all included in the Arduino Integrated Development Environment (IDE). It connects to the Arduino hardware, allowing it to upload and communicate with programmes. The technique for downloading and installing the Arduino IDE on Windows is described in this chapter.

## 3.1 Download the IDE

To download the IDE, go to the official page of Arduino (*https://www.arduino.cc/en/software*) and click on the software icon in the site's upper navigation bar. Download the IDE that is compatible with your operating system from the download options (Windows 7, 10, or 11). All these options are encircled in Fig. *3* for better visualization.



Start coding online and save your sketches in the cloud. The most up-to-date version of the IDE includes all libraries and also supports new Arduino boards.

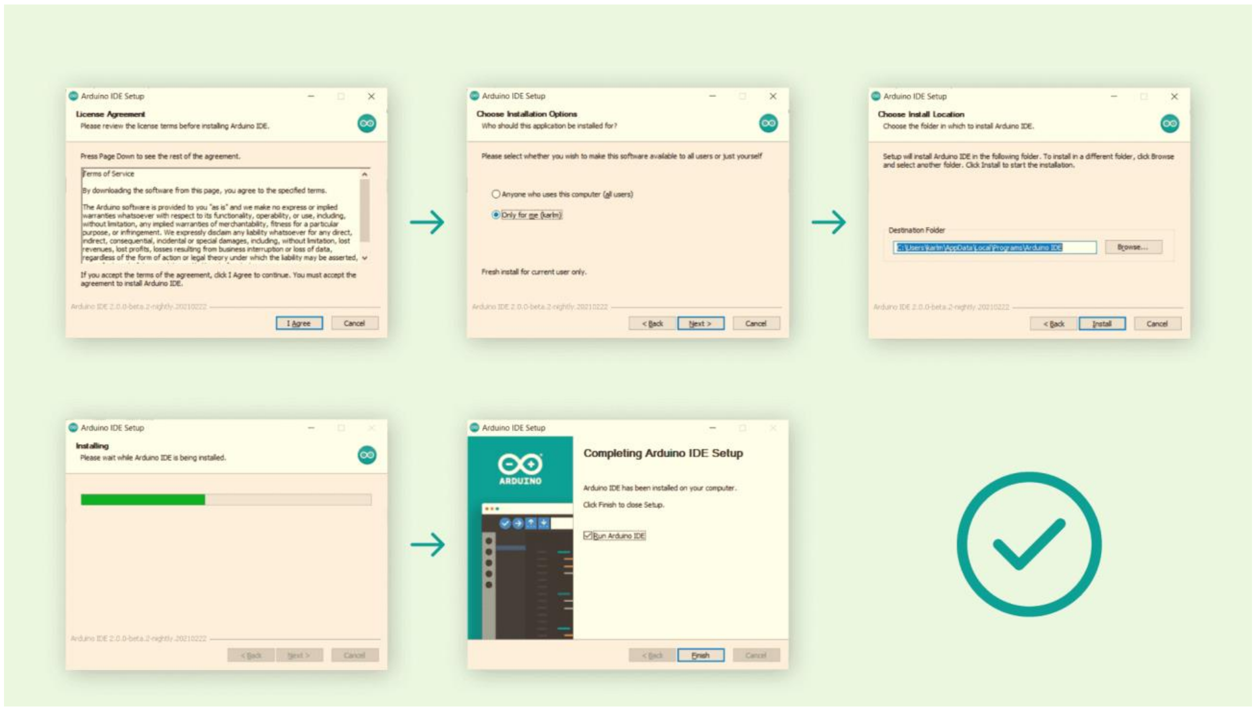
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the **Getting Started** page for Installation instructions.

Fig. 3. Arduino IDE downloading steps.

## 3.2 Installation

Simply execute the downloaded file and follow the directions in the installation guide to install the Arduino IDE on a Windows computer. It is likely that the installation will take several minutes. Fig. *4* illustrated the steps to follow during the installation.



**License Agreement**
Please review the license terms before installing Arduino IDE.

Press Page Down to see the rest of the agreement.

Terms of Service

By downloading the software from this page, you agree to the specified terms.

The Arduino software is provided to you "as is" and we make no express or implied warranties whatsoever with respect to its functionality, operability, or use, including, without limitation, any implied warranties of merchantability, fitness for a particular purpose, or infringement. We expressly disclaim any liability whatsoever for any direct, indirect, consequential, incidental or special damages, including, without limitation, lost revenues, lost profits, losses resulting from business interruption or loss of data, regardless of the form of action or legal theory under which the liability may be asserted,

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Arduino IDE.

Arduino IDE 2.0.0-beta.2-nightly.20210222

**Choose Installation Options**
Who should this application be installed for?

Please select whether you wish to make this software available to all users or just yourself

○ Anyone who uses this computer (all users)
● Only for me (karlm)

Fresh install for current user only.

**Choose Install Location**
Choose the folder in which to install Arduino IDE.

Setup will install Arduino IDE in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder: C:\Users\karlm\AppData\Local\Programs\Arduino IDE

**Installing**
Please wait while Arduino IDE is being installed.

**Completing Arduino IDE Setup**

Arduino IDE has been installed on your computer.

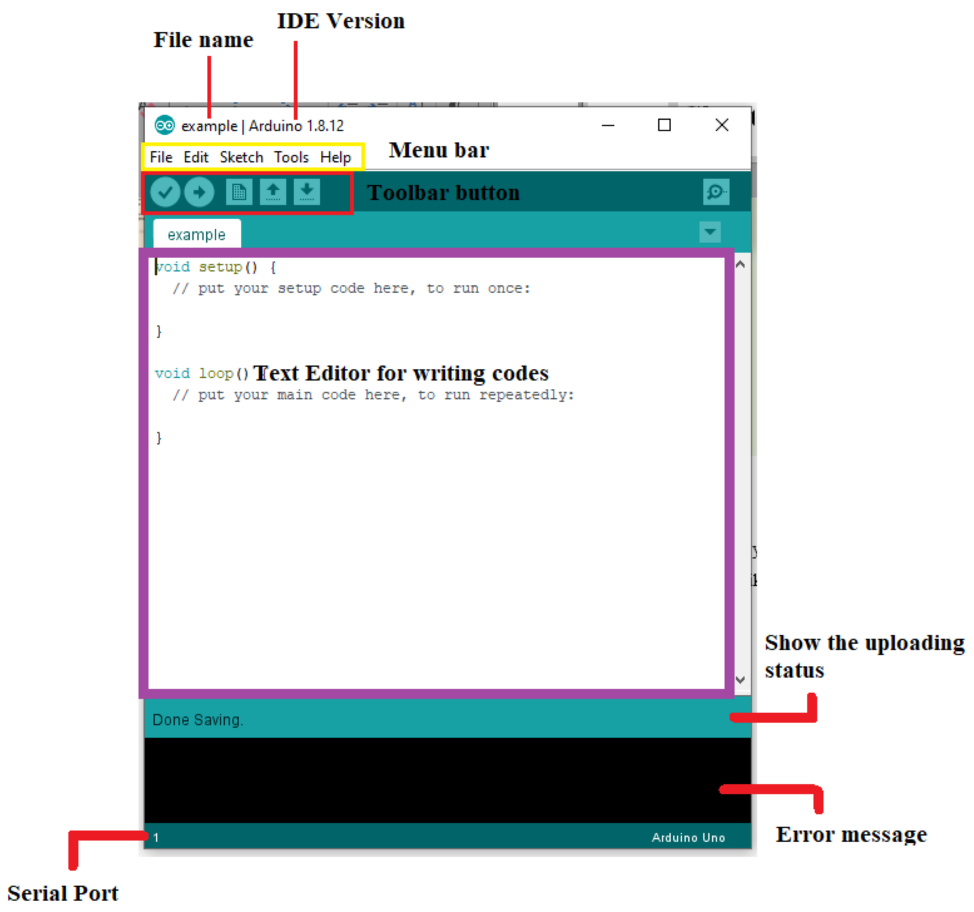
Click Finish to close Setup.

☑ Run Arduino IDE

Fig. 4. Instructions for installing the IDE.

## 3.3 Open the installed IDE

To open the Arduino IDE when it has been successfully installed, go to your system's start bar and search for Arduino. When you click on Arduino, an IDE will launch which will look like as shown in Fig. *5*.

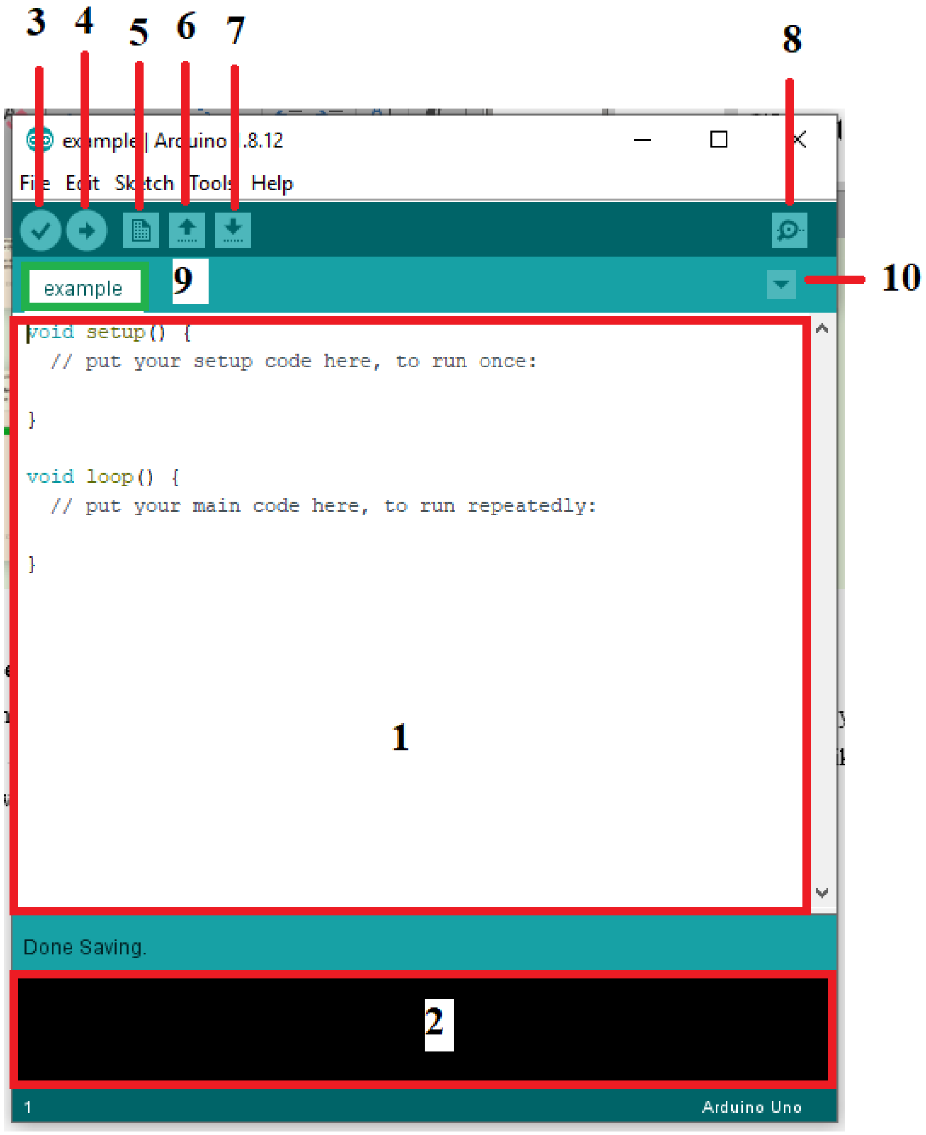


```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

Fig. 5. The interface of installed Arduino IDE.

# Chapter: 4 Screen structure of Arduino IDE

The screen structure of the Arduino IDE, as well as an example, will be explained in this chapter. A typical Arduino IDE is displayed in Fig. *6*.



```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

```
Done Saving.
```

Fig. 6. Screen structure of Arduino IDE.

## 4.1 Explanation of screen structure of Arduino IDE

### 4.1.1 Label 1

This section is used to create a programme: "void setup ()" and "void loop ()" appear when you create a new file. Describe the process that is only executed once when Arduino is started with "void setup ()." Describe the process that will be repeated using "void loop ()." Write settings to setup () and normal processing to loop (). Note that a message beginning with "//" will appear, but there will be no issues if you erase the memo.

### 4.1.2 Label 2

Checks and errors in the programme are displayed: If there's an issue with the software itself, look through the messages presented here to figure out what went wrong.

### 4.1.3 Label 3

Examine the programme you've created: Click to see if the programme you've generated is correct.

### 4.1.4 Label 4

To upload the developed programme to Arduino: Once the programme is sent, it is executed on the Arduino, allowing control of the electronic components. Before transferring, the application double-checks that everything is in order.

### 4.1.5 Label 5

Create a new program: When you create a new tab, it'll be displayed separately.

### 4.1.6 Label 6

To read the file.

### 4.1.7 Label 7

To save the program to a file.

### 4.1.8 Label 8

To display data from Arduino and display "Serial Monitor," which can send commands to Arduino.

### 4.1.9 Label 9

When multiple programs are open, it'll be displayed in separate tabs.

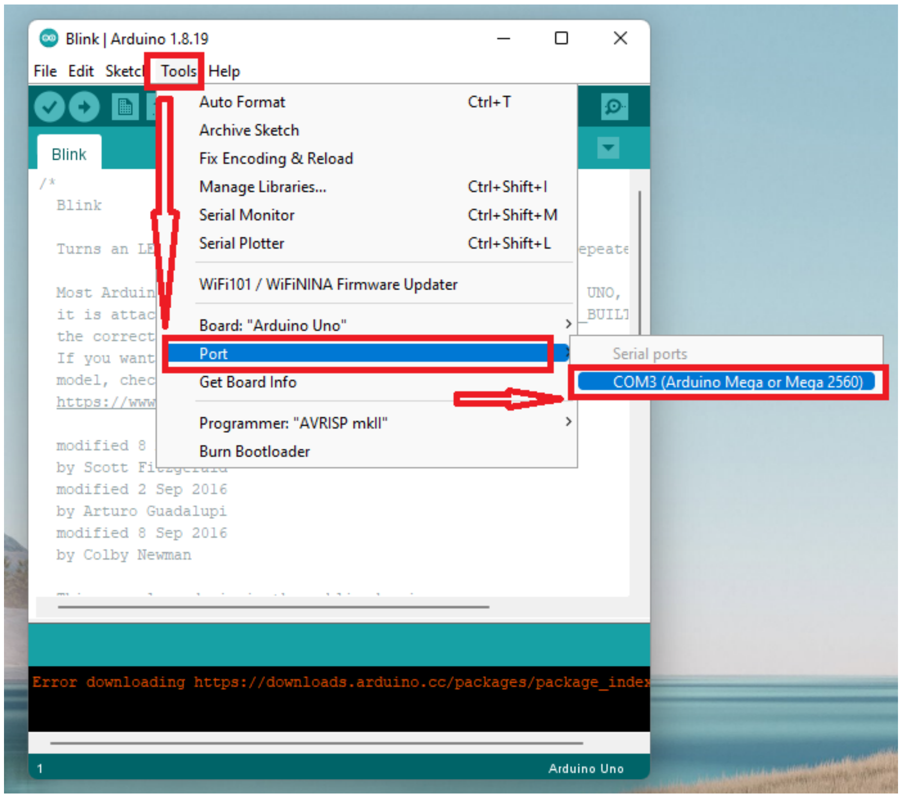
### 4.1.10 Label 10

Displays a menu for tab operations such as closing and switching tabs.

## 4.2 Example - Lighting up the LED equipped on Arduino

Arduino Uno/Mega has an LED that can be controlled by a program that can turn it on and off, thus creating a program that blinks the LED every second. Start the Arduino IDE and input the program as described below.

Connect Arduino and your computer through USB to transmit the software. Then Windows will be able to recognize Arduino and connect with it. Make communication settings in the Arduino IDE. To see a list of currently available ports, go to the "Tools" menu and select "Serial Port." Choose the port that shows the name of the connected Arduino, such as “COM3 (Arduino Mega or Mega 2560)” as shown in Fig. *7*.

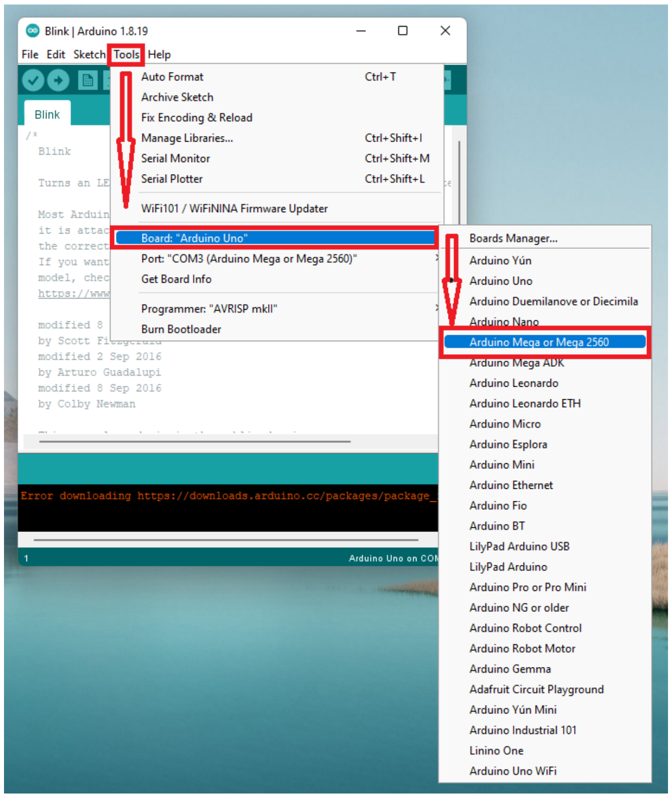

Blink | Arduino 1.8.19
File Edit Sketch Tools Help
Blink
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM3 (Arduino Mega or Mega 2560)
1
Arduino Uno

```
/*
  Blink

  Turns an LE                                        epeate

  Most Arduin                                         UNO,
  it is attac                                         _BUILT
  the correct
  If you want
  model, chec
  https://www

  modified 8
  by Scott Fi
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman
```

```
Error downloading https://downloads.arduino.cc/packages/package_index
```

Fig. 7.  Selection of the serial port.

Select the type of connection as well. In the "Tools" menu, select the Arduino that you want to use on the "Board." For Arduino Mega board, select “Arduino Mega or Mega 2560” as shown in Fig. *8*.



Board: "Arduino Uno"

Port: "COM3 (Arduino Mega or Mega 2560)"

Programmer: "AVRISP mkII"

```
/*
  Blink

  Turns an LE

  Most Arduin
  it is attac
  the correct
  If you want
  model, chec
  https://www

  modified 8
  by Scott Fi
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman
```

```
Error downloading https://downloads.arduino.cc/packages/package_
```

```
1                                    Arduino Uno on COM
```

Fig. 8. Select the Arduino board.

Now, click “Write to microcomputer board” (arrow icon) on the toolbar to transfer the program. When the transfer is complete, the program will run, and the inbuilt LED will start blinking as displayed in Fig. *9*.

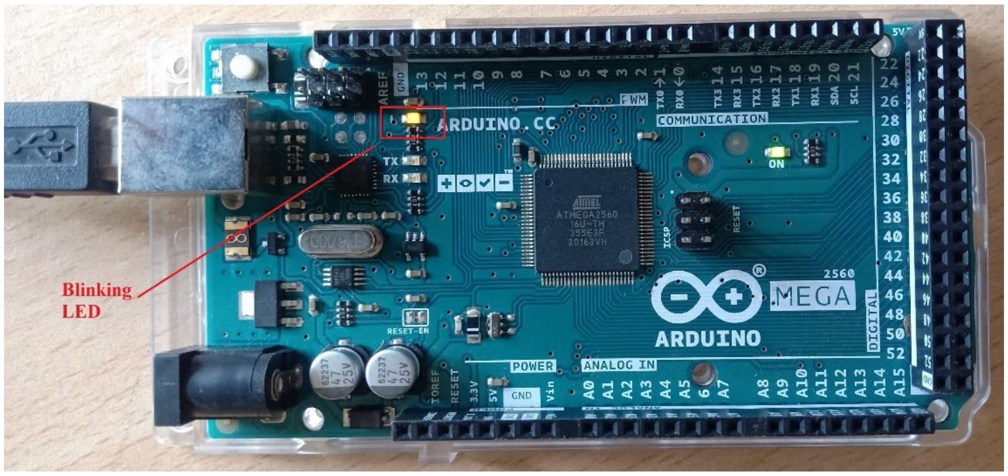


Fig. 9. Demonstration of blinking LED.

# Chapter: 5 Arduino Sensor – Types and Applications

The compatibility issue between hardware and accessible software IDEs is the key concern aspect while building any electronic project.

## 5.1 What is Arduino Sensor?

Arduino has a microcontroller as well as a software development environment (IDE) for uploading code to the hardware board. Due to the growing popularity of Arduino among amateurs, a slew of Arduino-compatible sensors has been released. There are many different types of Arduino sensors on the market. These sensors assist Arduino in interacting with the environment as well as developing new applications.

## 5.2 Working Principle

Before Arduino, microcontrollers didn't have a software IDE for uploading code to the hardware. To upload the code into the hardware, one needed to use a separate hardware device. It is simple to connect sensors to Arduino because of this flexibility feature. The only hardware necessary to interface these sensors with Arduino is the Breadboard and accompanying cables, as the microcontroller already includes a software IDE for programming. The Arduino IDE can be used to write code and upload it. For interface, you'll need a power supply, a ground, a breadboard, and some connecting wires.

## 5.3 Applications of Arduino Sensor

Many Arduinos sensor-based projects have been created for a variety of uses. Arduino is believed to have been utilized to bring a vision to life. For non-contact range detection, the ultrasonic module is used. It operates with the help of sonar. The IR obstacle avoidance sensor detects and emits a digital signal when it recognizes objects in front of it. It's a material that's found in robots. A soil hygrometer is a moisture sensor for the soil. When the moisture in the soil rises above a certain threshold, it generates a digital signal. This Arduino sensor is used to create an autonomous self-watering plant. Sound is detected using a microscope sensor. When the detected sound intensity exceeds a certain threshold, it generates a signal. The absolute pressure of the surroundings is measured using a digital barometric pressure sensor. This sensor can be used to determine the height of the robot or projectile. The Photoresistor sensor module is used to detect light. This sensor and Arduino are used in the night security light system. The ambient temperature is detected using the temperature sensor. To detect toxic gases such as LPG, i-Butane, Propane, Alcohol, and other similar substances. The gas sensor MQ-2 is utilized. The rain sensor is used to monitor the weather. The conventional light Flame sensor is used to detect flame. Human and pet motion is detected using a PIR sensor.

## 5.4 Examples of Arduino Sensor

There are many types of Arduino sensors available today. Some of them are given below.

- HC- SR04 Ultrasonic Module
- IR Infrared Obstacle Avoidance sensor
- Soil Hygrometer Detection Module
- Soil Moisture Sensor
- Microphone sensor
- Digital Barometric Pressure Sensor
- Photoresistor sensor
- Digital thermal sensor – Temperature sensor
- Rotary Encoder Module
- MQ-2 Gas sensor
- SW-420 Motion sensor
- Humidity and Rain Detection sensor
- Passive Buzzer Module
- Speed sensor Module
- IR Infrared Flame Detection sensor
- 5V 2- Channel Relay module
- Breadboard Power Supply Module 3.3V
- HC- SR501 Pyroelectric Infrared Sensor
- Accelerometer Module
- DHT11 Temperature and Humidity sensor
- RF 433MHz Transmitter/Receiver

Some of the most useful sensors like temperature sensors, humidity sensors, rain sensors, moisture sensors, etc. will be discussed in the next chapters.

# Chapter: 6 Measurement of temperature and humidity using the DHT11 module

DHT11 can measure temperature from 0°C to 50°C with ±2.0°C accuracy, and humidity from 20 to 80% with 5% accuracy.

## 6.1 DHT11 Module Pinout

The DHT11 module has four pins for connections which are explained in Fig. *10*.

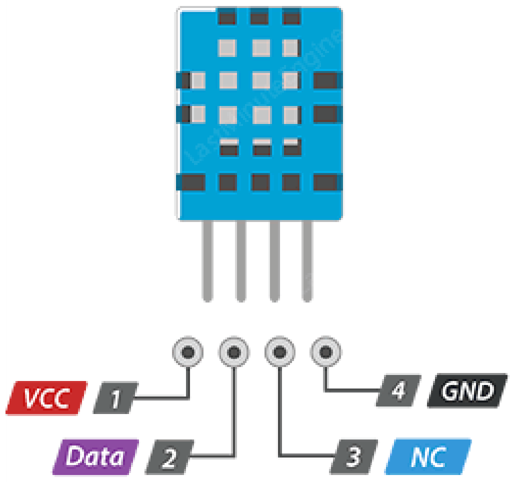


Fig. 10. DHT11 sensor pin connections.

- The sensor is powered by the VCC pin. Despite the fact that supply voltages range from 3.3 to 5.5 volts, a 5-volt supply is recommended. When using a 5V power supply, the sensor can be kept for up to 20 meters. With a 3.3V supply voltage, however, the cable length must not exceed 1 meter. Otherwise, the line voltage drop will cause measurement mistakes.
- A data pin is used to communicate between the sensor and the microcontroller.
- NC indicates Not connected.
- GND should be connected to the ground of Arduino.

## 6.2 Wiring DHT11 Module to Arduino

Begin by connecting the + (VCC) pin to the Arduino's 5V output and the – (GND) pin to the ground. Finally, connect the Out pin to the digital pin #8 as shown in Fig. *11*.

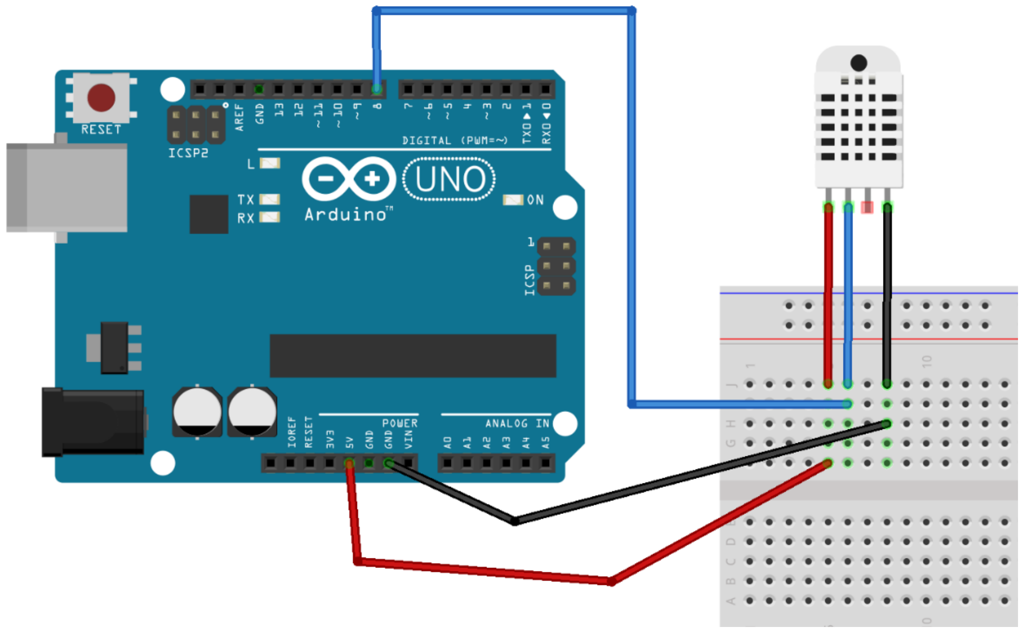


Fig. 11. Wiring DHT11 module with Arduino.

### 6.3 Installing DHT library

The data from DHT11 sensors is transferred using a single wire protocol. This procedure necessitates exact timing. Fortunately, the DHT Library was created to mask all of the intricacies, allowing us to read temperature and humidity data with simple commands.

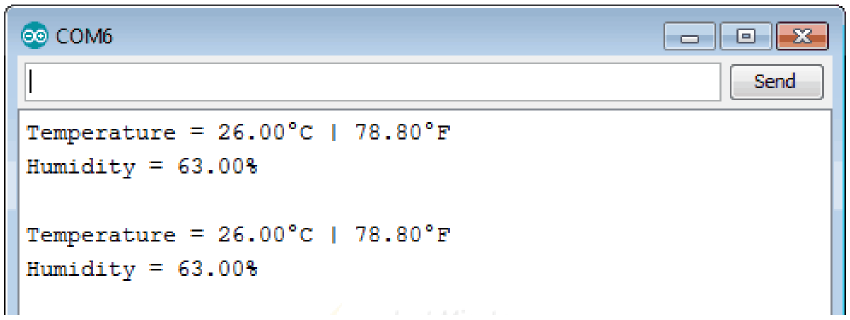
Download the library first, by visiting the GitHub repo link which is given here (https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib). To install it, open the Arduino IDE, and go to *Sketch* > *Include Library* > *Add .ZIP Library*, and then select the *DHTlib* ZIP file.

### 6.4 Arduino Code – Basic Example

After you've installed the library, you should paste the sketch below into the Arduino IDE.

```
#include <dht.h> // Include library
#define outPin 8 // Defines pin number to which the sensor is connected

dht DHT;                 // Creates a DHT object

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

void loop() {
        int readData = DHT.read11(outPin);

        float t = DHT.temperature;        // Read temperature
        float h = DHT.humidity;           // Read humidity

        Serial.print("Temperature = ");
        Serial.print(t);
        Serial.print("°C | ");
        Serial.print((t*9.0)/5.0+32.0);   // Convert celsius to fahrenheit
        Serial.println("°F ");
        Serial.print("Humidity = ");
        Serial.print(h);
        Serial.println("% ");
        Serial.println("");

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

Once the sketch is uploaded, open a Serial Monitor window to see the output from the Arduino as illustrated in Fig. *12*.



```
Temperature = 26.00°C | 78.80°F
Humidity = 63.00%

Temperature = 26.00°C | 78.80°F
Humidity = 63.00%
```

Fig. 12. DHT11 module output.

# Chapter: 7 Measurement of distance using Ultrasonic Sensor

HC-SR04 type ultrasonic sensor is used in this chapter to illustrate the working of the ultrasonic sensor for distance measurement specifications which are given in Table *1*.

Table 1. Specifications of HC-SR04 type ultrasonic sensor.

| **Model** | HC-SR04 |
|---|---|
| **Operating Voltage (VDC)** | 5 |
| **Average Current Consumption (mA)** | 2 |
| **Frequency (Hz)** | 40000 |
| **Sensing Angle** | 15° |
| **Max. Sensing Distance (cm)** | 450 |
| **Weight (gm)** | 9 |
| **Sensor Cover Dia. (mm)** | 16 |

## 7.1 HC-SR04 Ultrasonic Sensor Pins Layout

Fig. *13* Shows the ultrasonic sensor with pins layout.

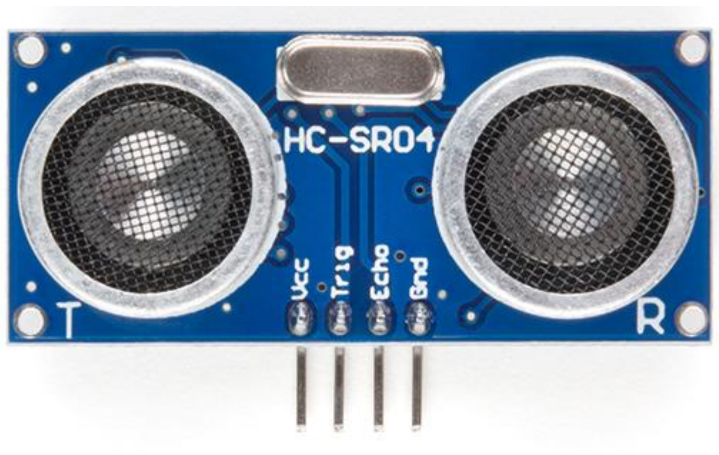


Fig. 13. Ultrasonic sensor with pins layout.

- VCC is the power supply for the HC-SR04 Ultrasonic distance sensor to which we connect the 5V pin on the Arduino.
- Trig (Trigger) pin is used to trigger the ultrasonic sound pulses.
- The echo pin produces a pulse when the reflected signal is received. The length of the pulse is proportional to the time it took for the transmitted signal to be detected.

- GND should be connected to the ground of Arduino.

## 7.2 Wiring – Connecting HC-SR04 to Arduino Uno

The HC-SR04 is simple to connect to the Arduino. Place the sensor on your breadboard first. Connect the VCC pin to the Arduino's 5V pin, while the GND pin is connected to the Arduino's Ground pin.

When you’re done you should have something that looks similar to the illustration shown in Fig. *14*.

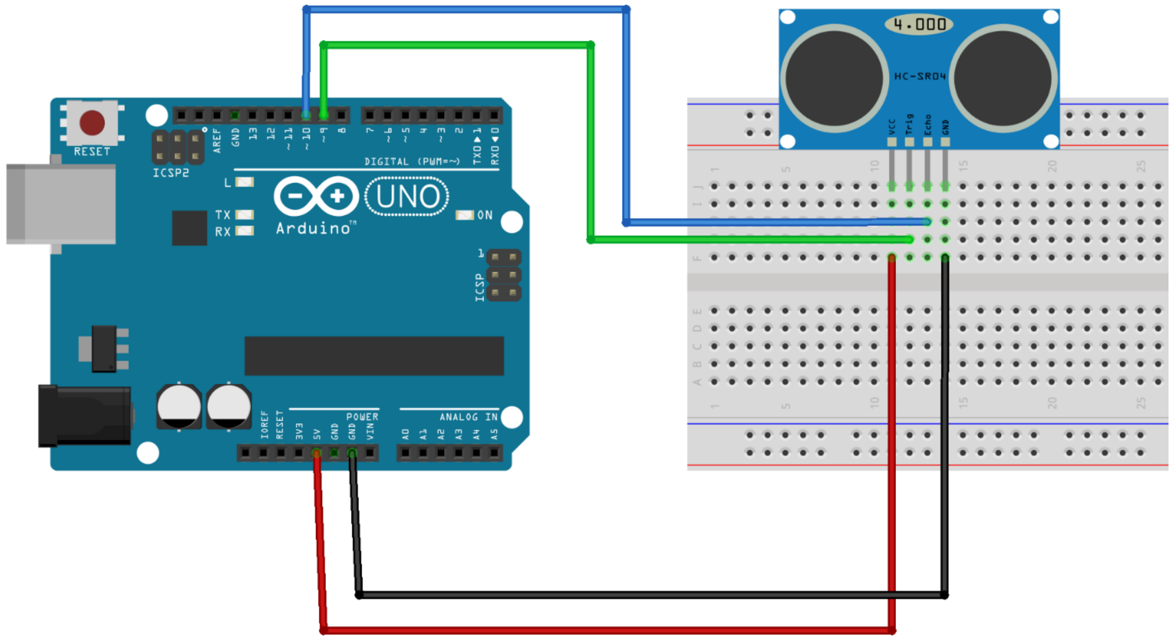


Fig. 14. Connections between ultrasonic sensor and Arduino.

We'll use a specific library instead of manually triggering the ultrasonic sensor and measuring the received signal pulse width. There are several to choose from, with "***NewPing***" being the most adaptable.

After you've installed the library, you should paste the sketch below into the Arduino IDE.

```
// This uses Serial Monitor to display Range Finder distance readings

// Include NewPing Library
#include "NewPing.h"

// Hook up HC-SR04 with Trig to Arduino Pin 9, Echo to Arduino pin 10
#define TRIGGER_PIN 9
#define ECHO_PIN 10

// Maximum distance we want to ping for (in centimeters).
#define MAX_DISTANCE 400

// NewPing setup of pins and maximum distance.
NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);
float duration, distance;

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

void loop()
{
    // Send ping, get distance in cm
    distance = sonar.ping_cm();

    // Send results to Serial Monitor
    Serial.print("Distance = ");

    if (distance >= 400 || distance <= 2)
    {
        Serial.println("Out of range");
    }
    else
    {
        Serial.print(distance);
        Serial.println(" cm");
    }
    delay(500);
}
```

# Chapter: 8 Measurement of soil moisture using LM393 comparator and probe

The soil moisture sensor is made up of two probes that measure the volumetric content of water in the soil. The two probes allow current to flow through the soil, and the resistance value is used to calculate the moisture value. When there is more water in the soil, it conducts more electrical, resulting in less resistance. As a result, the moisture content of the air will be increased. Dry soil conducts electricity poorly, therefore when there is less water, the soil conducts less electricity, resulting in increased resistance. As a result, the moisture level will be reduced. When exposed to moisture, one of these sensors' most well-known flaws is their limited lifespan. The rate of corrosion is greatly accelerated when power is applied to the probe on a continuous basis.

## 8.1 Components of soil moisture sensor

A typical soil moisture sensor has two components: a probe and an electronic module called LM393 comparator as shown in Fig. *15*.

The sensor has a fork-shaped probe with two exposed conductors that is inserted into the soil or anywhere else where water content is to be detected. It functions as a variable resistor, with resistance that varies depending on the moisture content of the soil.

The probe is connected to the Arduino via an electronic module in the sensor.

The Analog Output (AO) pin on the module creates an output voltage based on the resistance of the probe. The same signal is supplied into an LM393 High Precision Comparator, which digitizes it and outputs it to a Digital Output (DO) pin.

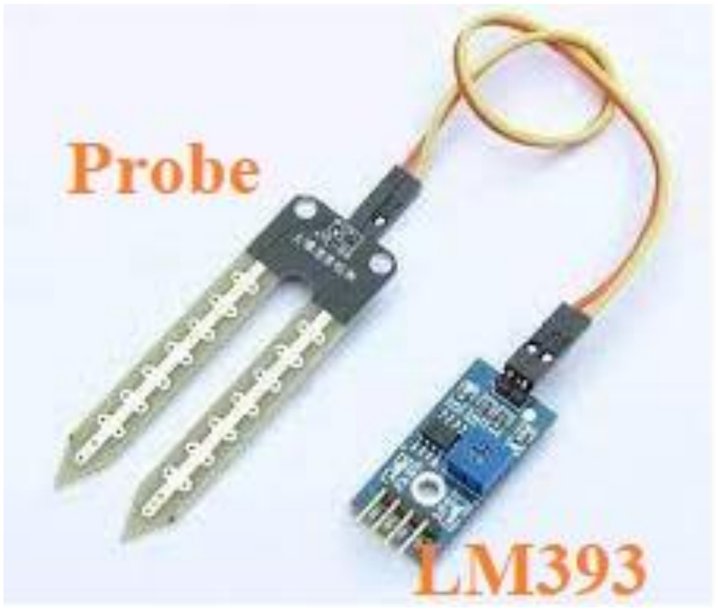


Fig. 15. Soil moisture probe and LM393 comparator.

**8.2 Connections between LM393 and Arduino**

- The AO (Analog Output) pin generates an analogue output between 0V and 5V and is connected to one of your Arduino's analogue inputs.
- The DO (Digital Output) pin provides the internal comparator circuit's digital output. It can be connected to any Arduino digital pin or directly to a 5V relay or other similar device.
- The sensor is powered via the VCC pin. The sensor should be powered between 3.3 and 5 volts. Please keep in mind that the analogue output will vary based on the sensor voltage.
- GND is a ground connection.

Fig. *16* shows the connections between Arduino and the soil moisture sensor.

*Note: To limit the rate of corrosion, only power the sensor of a digital pin when taking a reading.*

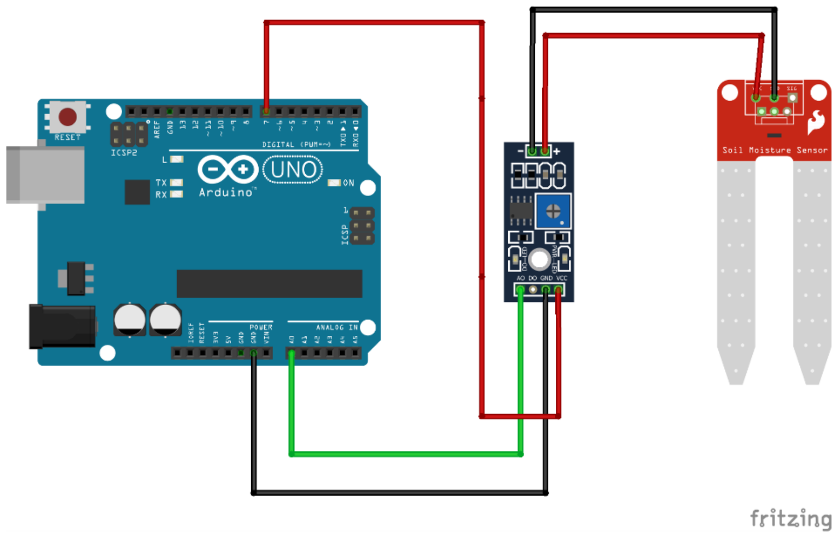


Fig. 16. Connections between Arduino and soil moisture sensor.

## 8.3 Calibration

It is recommended that you calibrate your soil moisture sensor for the specific type of soil you wish to monitor before using it to receive reliable readings. Different types of soil have different effects on the sensor, therefore depending on the soil you use, your sensor may be more or less sensitive.

Note what values your sensor produces when the soil is as dry as possible vs. when it is entirely saturated with moisture using the design below.

```
// Sensor pins
#define sensorPower 7
#define sensorPin A0

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

    // Initially keep the sensor OFF
    digitalWrite(sensorPower, LOW);

    Serial.begin(9600);
}

void loop() {
    //get the reading from the function below and print it
    Serial.print("Analog output: ");
    Serial.println(readSensor());

    delay(1000);
}

//  This function returns the analog soil moisture measurement
int readSensor() {
    digitalWrite(sensorPower, HIGH);    // Turn the sensor ON
    delay(10);                          // Allow power to settle
    int val = analogRead(sensorPin);    // Read the analog value form sensor
    digitalWrite(sensorPower, LOW);     // Turn the sensor OFF
    return val;                         // Return analog moisture value
}
```

## 8.4 Soil moisture output values using sensor

To obtain the soil moisture output values after calibration, use the sketch below.

```
/* Change these values based on your calibration values */
#define soilWet 500   // Define max value we consider soil 'wet'
#define soilDry 750   // Define min value we consider soil 'dry'

// Sensor pins
#define sensorPower 7
#define sensorPin A0

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

    // Initially keep the sensor OFF
    digitalWrite(sensorPower, LOW);

    Serial.begin(9600);
}

void loop() {
    //get the reading from the function below and print it
    int moisture = readSensor();
    Serial.print("Analog Output: ");
    Serial.println(moisture);

    // Determine status of our soil
    if (moisture < soilWet) {
        Serial.println("Status: Soil is too wet");
    } else if (moisture >= soilWet && moisture < soilDry) {
        Serial.println("Status: Soil moisture is perfect");
    } else {
        Serial.println("Status: Soil is too dry - time to water!");
    }

    delay(1000);    // Take a reading every second for testing
                    // Normally you should take reading perhaps once or twice
a day
    Serial.println();
}

//  This function returns the analog soil moisture measurement
int readSensor() {
    digitalWrite(sensorPower, HIGH);    // Turn the sensor ON
    delay(10);                          // Allow power to settle
    int val = analogRead(sensorPin);    // Read the analog value form sensor
    digitalWrite(sensorPower, LOW);     // Turn the sensor OFF
    return val;                         // Return analog moisture value
}
```

# Chapter: 9 Detection of rainfall using the rain sensor module

This sensor can be used to monitor rain and send closure requests to electronic shutters, windows, awnings, or skylights whenever the rain is detected.

**9.1 How does Rain Sensor work?**

The sensing pad, which is made up of a sequence of exposed copper traces, works as a variable resistor (similar to a potentiometer) whose resistance fluctuates depending on how much water is on its surface.

The amount of water has an inverse relationship with the resistance:

- The more water on the surface, the greater the conductivity and the lower the resistance;
- The less water on the surface, the worse the conductivity and the higher the resistance.

The sensor generates an output voltage based on the resistance, which can be used to assess whether or not it is raining.

**9.2 Components of the rain sensor**

A typical rain sensor has two components: a sensing pad and an electronic module called LM393 comparator.

**9.3 Connections between sensors and Arduino**

The connections between the sensing pad, LM393 comparator, and Arduino are the same as described in Chapter 8.

**9.4 Calibration of sensor**

To acquire accurate readings from your rain sensor, it's a good idea to calibrate it. For calibrating the digital output, the module contains a built-in potentiometer (DO). You can adjust the threshold by rotating the potentiometer's knob. The Status LED will illuminate and the digital output (DO) will output LOW when the amount of water exceeds the threshold value. Sprinkle little water on the detecting pad and turn the pot clockwise until the Status LED turns on, then counterclockwise till the LED turns off to calibrate the sensor. This sensor has been calibrated and is now ready to be used.

### 9.5 Detecting Rain – Arduino Code

Place the rain sensor in a location where rainwater can fall directly into the sensor, possibly over the roof, once the circuit is complete. It should also be slightly inclined to allow water to flow freely. Now, on the Arduino, upload the following sketch.

```
// Sensor pins
#define sensorPower 7
#define sensorPin 8

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

    // Initially keep the sensor OFF
    digitalWrite(sensorPower, LOW);

    Serial.begin(9600);
}

void loop() {
    //get the reading from the function below and print it
    int val = readSensor();
    Serial.print("Digital Output: ");
    Serial.println(val);

    // Determine status of rain
    if (val) {
        Serial.println("Status: Clear");
    } else {
        Serial.println("Status: It's raining");
    }

    delay(1000);    // Take a reading every second
    Serial.println();
}

//  This function returns the sensor output
int readSensor() {
    digitalWrite(sensorPower, HIGH);    // Turn the sensor ON
    delay(10);                          // Allow power to settle
    int val = digitalRead(sensorPin);   // Read the sensor output
    digitalWrite(sensorPower, LOW);     // Turn the sensor OFF
    return val;                         // Return the value
}
```

Open a Serial Monitor window after the programme has been uploaded to see the Arduino's output. When the weather is clear, you should observe a digital output HIGH. Sprinkle some water on the sensing pad to see it sense water.

# Chapter: 10 Measurement of water level using water level sensor

This sensor illustrated in Fig. *17* is used to measure the level of water in the sump.

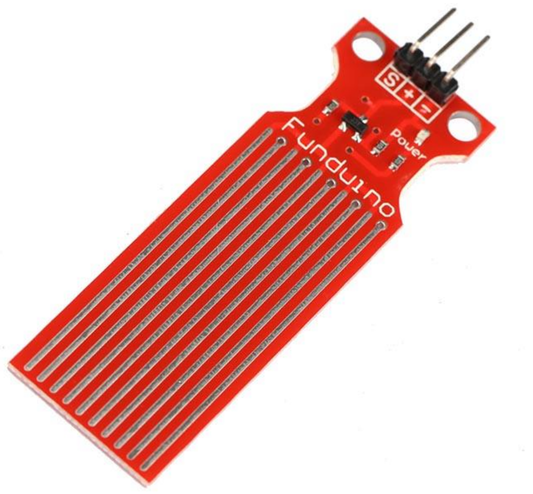


Fig. 17. Water level indicator sensor.

### 10.1 Connections to Arduino

- S (Signal) pin is an analog output that will be connected to one of the analog inputs on your Arduino.
- + (VCC) pin supplies power for the sensor. It is recommended to power the sensor between 3.3V and 5V. Please note that the analog output will vary depending on what voltage is provided for the sensor.
- -(GND) is a ground connection.

### 10.2 Water level measurement – Arduino code

Once the circuit is built, upload the following sketch to your Arduino.

```
// Sensor pins
#define sensorPower 7
#define sensorPin A0

// Value for storing water level
int val = 0;

void setup() {
    // Set D7 as an OUTPUT
    pinMode(sensorPower, OUTPUT);

    // Set to LOW so no power flows through the sensor
    digitalWrite(sensorPower, LOW);

    Serial.begin(9600);
}

void loop() {
    //get the reading from the function below and print it
    int level = readSensor();

    Serial.print("Water level: ");
    Serial.println(level);

    delay(1000);
}

//This is a function used to get the reading
int readSensor() {
    digitalWrite(sensorPower, HIGH);    // Turn the sensor ON
    delay(10);                          // wait 10 milliseconds
    val = analogRead(sensorPin);        // Read the analog value form sensor
    digitalWrite(sensorPower, LOW);     // Turn the sensor OFF
    return val;                         // send current reading
}
```

Open a Serial Monitor window after the programme has been uploaded to see the Arduino's output. When the weather is clear, you should observe a digital output HIGH. Sprinkle some water on the sensing pad to see it sense water.

## Important links as references:

https://www.arduino.cc/

https://www.arduino.cc/en/software

https://www.arduino.cc/en/Guide

https://www.arduino.cc/en/Tutorial/HomePage

https://www.arduino.cc/reference/en/libraries/adafruit-unified-sensor/

https://github.com/adafruit/Adafruit_Sensor

https://learn.adafruit.com/dht/downloads

https://www.ladyada.net/learn/arduino/

https://www.programmingelectronics.com/tutorial-3-arduino-ide-and-sketch-overview/

https://fritzing.org/