

ARDUINO TUTORIAL

What is Arduino?

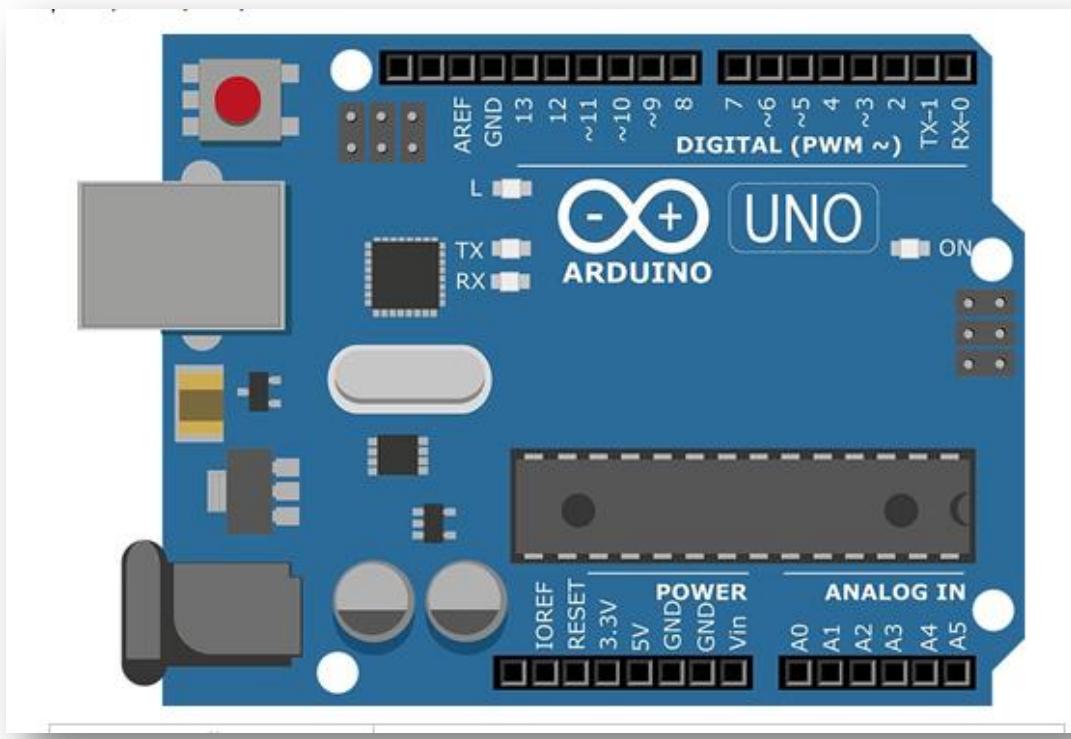
Arduino is a software as well as hardware platform that helps in making electronic projects. It is an open source platform and has a variety of controllers and microprocessors. There are various types of Arduino boards used for various purposes.

The Arduino is a single circuit board, which consists of different interfaces or parts. The board consists of the set of digital and analog pins that are used to connect various devices and components, which we want to use for the functioning of the electronic devices.

HARDWARE- ARDUINO UNO R3 BOARD.

SOFTWARE- ARDUINO IDE

Arduino.cc>> Download>>choose OS



Different Types of Arduino Boards

Arduino Uno (R3)

Arduino Nano

Arduino Micro

Arduino Due

LilyPad Arduino Board

Arduino Bluetooth

Arduino Diecimila

RedBoard Arduino Board

Arduino Mega (R3) Board

Arduino Leonardo Board

Arduino Robot

Arduino Esplora

Arduino Pro Mic

Arduino Ethernet

Arduino Zero

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

Parts List

UNO R3



USB Cable



RFID Module



4X4 Keypad



5V Step Motor



Moto Drivers



7segment



4 7segment



8*8 matrix



Jumper Wire



F-MDupont wire



Water



B10K Variable



2.54mm pin



Clock

ARDUINO TU

TURER GPK

www.quadstore.in

LED



5V Relay



Big Sound



Key Switch*4



9V battery holder



830 breadboard

www.quadstore.in

74HC595



SW-520D



Flame



LM35



IR Receiver



CDS

ARDUINO TU

TURER GPK



Installing IDE

STEP-1: Download the Arduino IDE (Integrated Development Environment)

Access the Internet:

In order to get your Arduino up and running, you'll need to download some software first from

www.arduino.cc (it's free!). This software, known as the Arduino IDE, will allow you to program the

Arduino to do exactly what you want. It's like a word

~~ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK~~

internet-capable computer, open up your favorite browser and type in the following URL into the

address bar:

www.arduino.cc/en/Main/Software

the following links: Windows User :

<http://www.arduino.cc/en/Guide/Windows> Mac OS X

User : <http://www.arduino.cc/en/Guide/MacOSX>

Linux User :

<http://playground.arduino.cc/Learning/Linux> For more detailed information about Arduino IDE,

please refer to the following link:

<http://www.arduino.cc/en/Guide/HomePage>

STEP-2: Connect your Arduino Uno to your Computer:

Use the USB cable provided in the kit to connect the Arduino to one of your computer's USB inputs.

STEP-3: Install Drivers

Depending on your computer's operating system, you

~~ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK~~

Please go to the URLs below for specific instructions on how to install the drivers onto your Arduino

Uno.

Windows Installation Process:

Go to the web address below to access the instructions for installations on a Windows-based computer.

<http://arduino.cc/en/Guide/Windows>

Macintosh OS X Installation Process:

Macs do not require you to install drivers. Enter the following URL if you have questions. Otherwise proceed to next page.

<http://arduino.cc/en/Guide/MacOSX>

Linux:

32 bit / 64 bit, Installation Process Go to the web address below to access the instructions for installations on a Linux-based computer.

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK
<http://www.arduino.cc/playground/Learning/Linux>

STEP-4: Open the Arduino IDE

Open the Arduino IDE software on your computer.

Poke around and get to know the interface.

We aren't going to code right away, this is just an introduction. The step is to set your IDE to identify your Arduino Uno.

GUI (Graphical User Interface)

Verify

Checks your code for errors compiling it.

Upload

Compiles your code and uploads it to the configured board. See uploading below for details.

Note: If you are using an external programmer with your board, you can hold down the "shift" key on your computer when using this icon. The text will change to "Upload using Programmer"

New

Creates a new sketch.

Open

Presents a menu of all the sketches in your sketchbook. Clicking one will open it within the current window overwriting its content.

Note: due to a bug in Java, this menu doesn't scroll; if you need to open a sketch late in the list,

use the File | Sketchbookmenu instead.

Save

Saves your sketch.

Serial Monitor

Opens the serial monitor.

STEP-5: Select your board: Arduino Uno

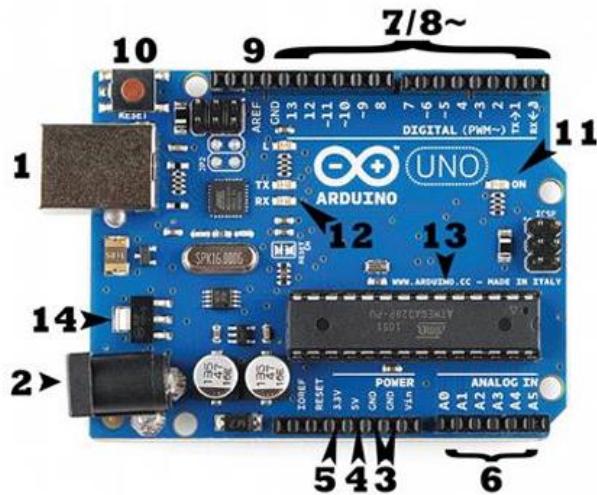
STEP-6: Select your Serial Device

Windows: Select the serial device of the Arduino board from the Tools | Serial Port menu. This is likely to be com3 or higher (COM1 and COM2 are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

About Arduino Uno R3 board

What's on the board?

There are many varieties of Arduino boards that can be used for different purposes. Some boards look a bit different from the one below, but most Arduino have the majority of these components in common:



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** coming from your computer or a wall power supply that is terminated in a **barrel jack**. In the picture above the USB connection is labeled (1) and the barrel jack is labeled (2).

NOTE: Do NOT use a power supply greater than 20 Volts as you will overpower (and thereby destroy) your Arduino.

The recommended voltage for most Arduino models is between 6 and 12 Volts.

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

The pins on your Arduino are the places where you connect wires to construct a circuit (probably in conjunction with a breadboard and some wire). They usually have black plastic ‘headers’ that allow you to just plug a wire right into the board. The Arduino has several different kinds of pins, each of which is labeled on the board and used for different functions.

- **GND (3):** Short for ‘Ground’. There are several GND pins on the Arduino, any of which can be used to ground your circuit.

- **5V (4) & 3.3V (5):** As you might guess, the 5V pin supplies 5 volts of power, and the 3.3V pin supplies 3.3 volts of power. Most of the simple components used with the Arduino run happily off of 5 or 3.3 volts.
- **Analog (6):** The area of pins under the ‘Analog In’ label (A0 through A5 on the UNO) are Analog In pins. These pins can read the signal from an analog sensor (like a temperature sensor) and convert it into a digital value that we can read.
- **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).
- **PWM (8):** You may have noticed the tilde (~) next to some of the digital pins (3, 5, 6, 9, 10, and 11 on the UNO).

These pins act as normal digital pins, but can also be used for

something called Pulse-Width Modulation (PWM). We have a tutorial on PWM, but for now,

think of these pins as being able to simulate analog output (like fading an LED in and out).

• **AREF (9): Stands for Analog Reference.** Most of the time you can leave this pin alone. It is

sometimes used to set an external reference voltage (between 0 and 5 Volts) as the upper limit for the analog input pins.

Reset Button

Just like the original Nintendo, the Arduino has a reset button (10). Pushing it will temporarily

connect the reset pin to ground and restart any code that is loaded on the Arduino. This can be

very useful if your code doesn't repeat, but you want to test it multiple times. Unlike the original

Nintendo however, blowing on the Arduino doesn't usually fix any problems.

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. Time to re-check your circuit!

TX RX LEDs

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

TX is short for transmit, RX is short for receive. These markings appear quite a bit in electronics to indicate the pins responsible for serial communication. In our case, there are two places on the Arduino UNO where TX and RX appear – once by digital pins 0 and 1, and a second time next to the TX and RX indicator LEDs (12). These LEDs will give us some nice visual indications whenever our

Arduino is receiving or transmitting data (like when we're loading a new program onto the board).

Main IC

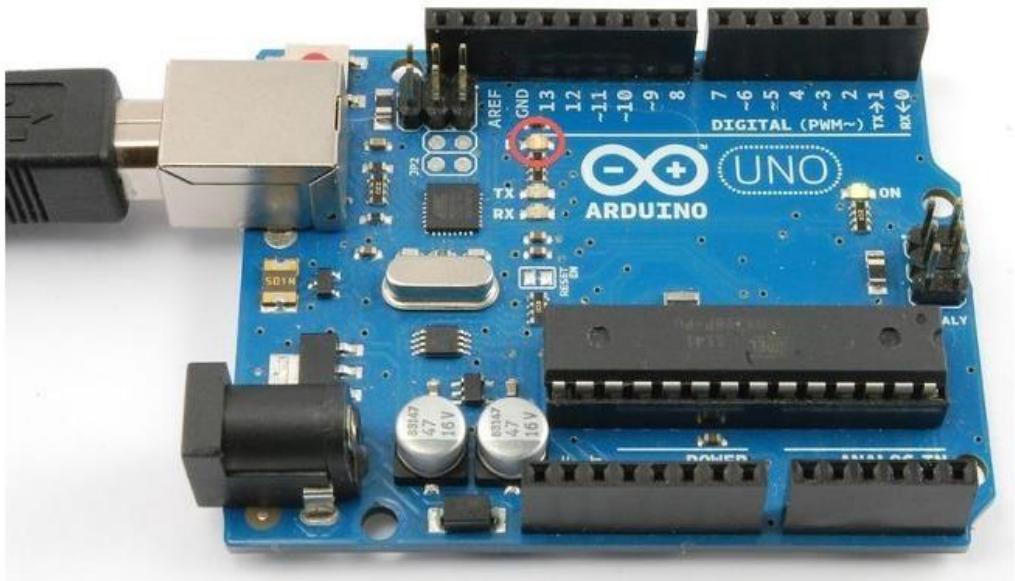
The black thing with all the metal legs is an IC, or Integrated Circuit (13). Think of it as the brains of our Arduino. The main IC on the Arduino is slightly different from board type to board type, but is usually from the ATmega line of IC's from the ATMEL company. This can be important, as you may need to know the IC type (along with your board type) before loading up a new program from the Arduino software. This information can usually be found in writing on the top side of the IC. If you want to know more about the difference between various IC's, reading the datasheets is often a good idea.

Voltage Regulator

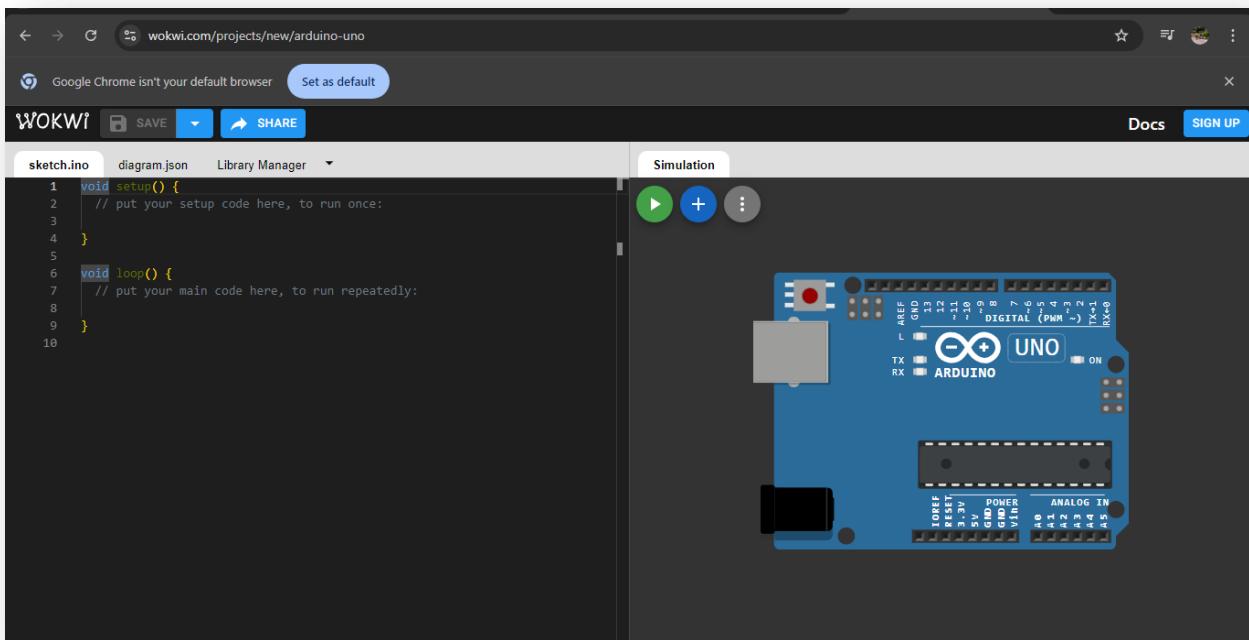
The voltage regulator (14) is not actually something you can (or should) interact with on the

Arduino. But it is potentially useful to know that it is there and what it's for. The voltage regulator does exactly what it says – it controls the amount of voltage that is let into the Arduino board. Think of it as a kind of gatekeeper; it will turn away an extra voltage that might harm the circuit. Of course, it has its limits, so don't hook up your Arduino to anything greater than 20 volts.

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK



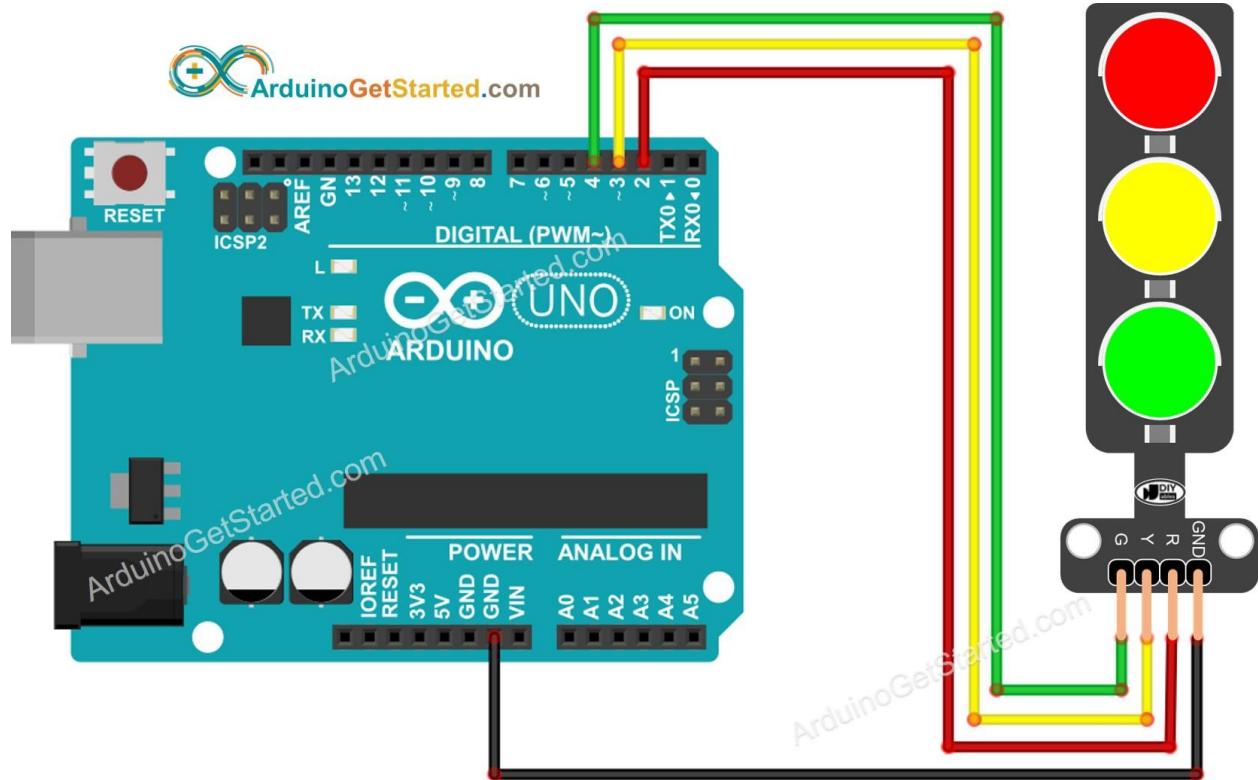
HELLO WORLD ARDUINO PROGRAM

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
void setup() {  
    // put your setup code here, to run once:  
    Serial.begin(9600);  
  
}  
  
void loop() {  
    // put your main code here, to run repeatedly:  
    Serial.println("Hello World!");  
    delay(1000);  
  
}
```

LED

TRAFFIC LIGHT MODULE



```
#define PIN_RED  2
// The Arduino pin connected to R pin of traffic light
module

#define PIN_YELLOW 3
// The Arduino pin connected to Y pin of traffic light
module

#define PIN_GREEN  4
// The Arduino pin connected to G pin of traffic light
module

#define RED_TIME   2000
// RED time in millisecond

#define YELLOW_TIME 1000
// YELLOW time in millisecond

#define GREEN_TIME 2000
// GREEN time in millisecond
```

```
#define RED  0
// Index in array

#define YELLOW 1
// Index in array

#define GREEN  2
// Index in array

const int pins[] = {PIN_RED, PIN_YELLOW, PIN_GREEN};

const int times[] = {RED_TIME, YELLOW_TIME,
GREEN_TIME};

void setup()
{
    pinMode(PIN_RED, OUTPUT);
    pinMode(PIN_YELLOW, OUTPUT);
    pinMode(PIN_GREEN, OUTPUT);
}
```

```
// the loop function runs over and over again forever
```

```
void loop()
```

```
{
```

```
    for (int light = RED; light <= GREEN; light++) {
```

```
        trafic_light_on(light);
```

```
        delay(times[light]); // keep light on during a period of  
time
```

```
}
```

```
}
```

```
void trafic_light_on(int light) {
```

```
    for (int i = RED; i <= GREEN; i++) {
```

```
        if (i == light)
```

```
            digitalWrite(pins[i], HIGH); // turn on
```

```
        else
```

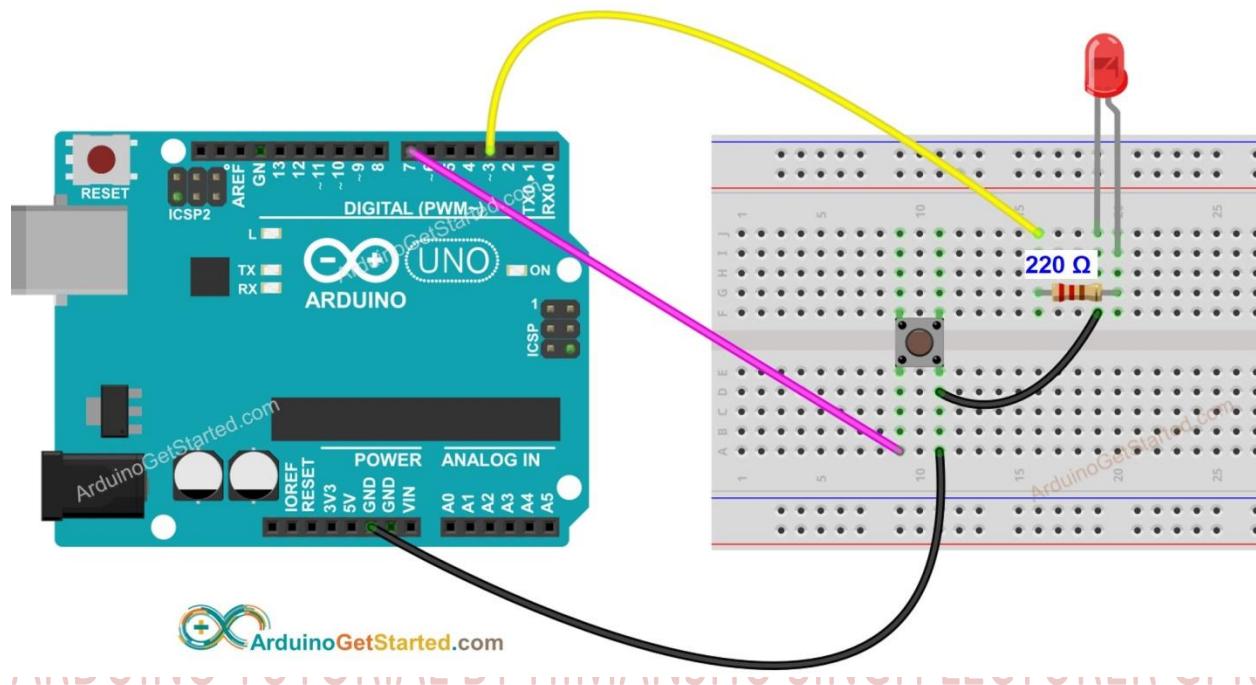
```
digitalWrite(pins[i], LOW); // turn off
```

```
}
```

```
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

ARDUINO BUTTON LED



```
// constants won't change. They're used here to set pin
numbers:
```

```
const int BUTTON_PIN = 7;
```

```
// the number of the pushbutton pin
```

```
const int LED_PIN = 3;
```

```
// the number of the LED pin
```

```
// variables will change:  
  
int buttonState = 0;  
  
// variable for reading the pushbutton status  
  
void setup()  
{  
  
    // initialize the LED pin as an output:  
  
    pinMode(LED_PIN, OUTPUT);  
  
    // initialize the pushbutton pin as an pull-up input:  
  
    // the pull-up input pin will be HIGH when the switch is  
    open and LOW when the switch is closed.  
  
    pinMode(BUTTON_PIN, INPUT_PULLUP);  
  
}  
  
  
void loop() {  
  
    // read the state of the pushbutton value:  
  
    buttonState = digitalRead(BUTTON_PIN);
```

```
// control LED according to the state of button

if(buttonState == LOW)

// If button is pressing

digitalWrite(LED_PIN, HIGH);

// turn on LED

else

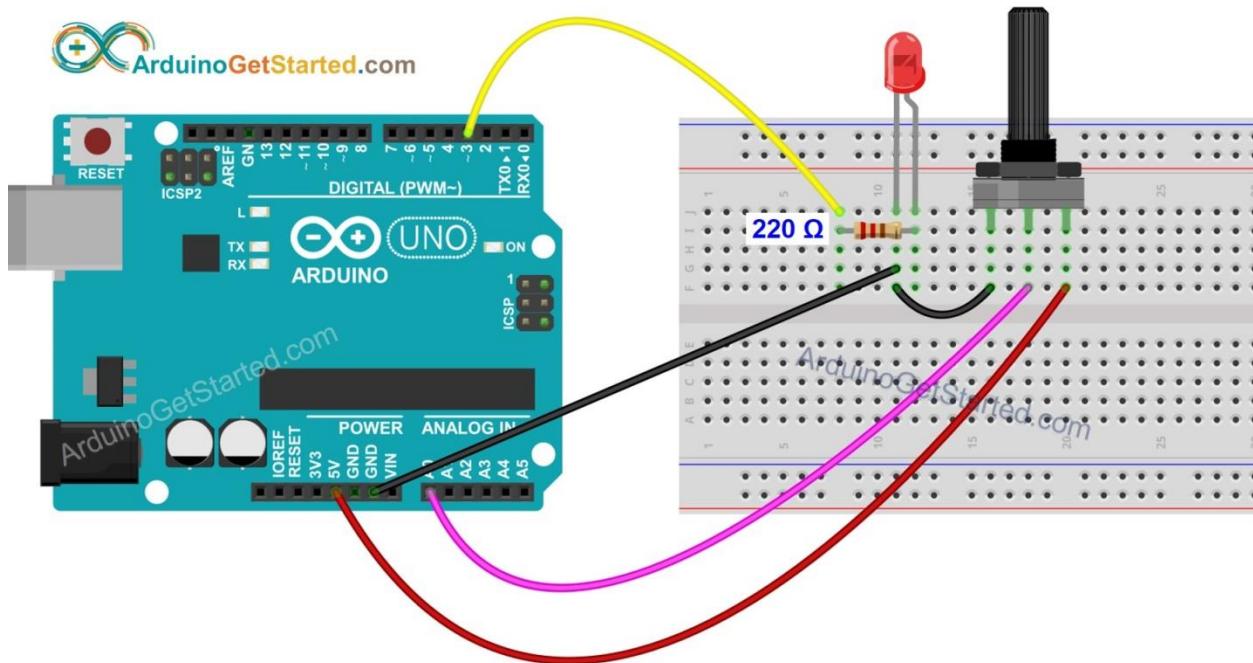
// otherwise, button is not pressing

digitalWrite(LED_PIN, LOW);

// turn off LED

}
```

POTENTIOMETER FADE LED



```
int LED_PIN = 3;
```

```
// the PWM pin the LED is attached to
```

```
// the setup routine runs once when you press reset:
```

```
void setup()
```

```
{
```

```
// initialize serial communication at 9600 bits per second:
```

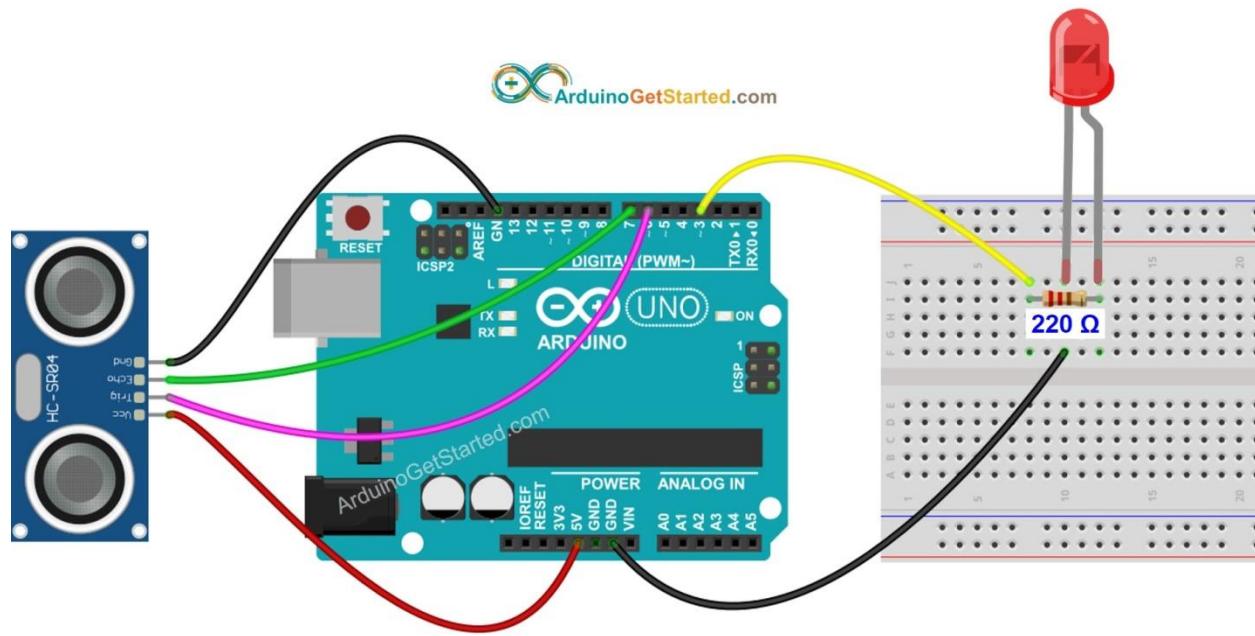
```
Serial.begin(9600);
```

```
// declare LED pin to be an output:  
  
pinMode(LED_PIN, OUTPUT);  
  
}  
  
  
// the loop routine runs over and over again forever:  
  
void loop()  
  
{  
  
    // reads the input on analog pin A0 (value between 0  
    and 1023)  
  
    int analogValue = analogRead(A0);  
  
  
  
    // scales it to brightness (value between 0 and 255)  
  
    int brightness = map(analogValue, 0, 1023, 0, 255);  
  
  
  
    // sets the brightness LED that connects to pin 3  
  
    analogWrite(LED_PIN, brightness);  
  
}
```

```
// print out the value  
Serial.print("Analog: ");  
Serial.print(analogValue);  
Serial.print(", Brightness: ");  
Serial.println(brightness);  
delay(100);  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

ARDUINO ULTRA SONIC SENSOR LED



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

// constants won't change

const int TRIG_PIN = 6;

// Arduino pin connected to Ultrasonic Sensor's TRIG pin

const int ECHO_PIN = 7;

// Arduino pin connected to Ultrasonic Sensor's ECHO pin

const int LED_PIN = 3;

// Arduino pin connected to LED's pin

```
const int DISTANCE_THRESHOLD = 50;
```

```
// centimeters
```

```
// variables will change:
```

```
float duration_us, distance_cm;
```

```
void setup()
```

```
{
```

```
Serial.begin(9600);
```

```
// initialize serial port
```

```
pinMode(TRIG_PIN, OUTPUT);
```

```
// set arduino pin to output mode
```

```
pinMode(ECHO_PIN, INPUT);
```

```
// set arduino pin to input mode
```

```
pinMode(LED_PIN, OUTPUT);
```

```
// set arduino pin to output mode
```

```
}
```

```
void loop()

{

// generate 10-microsecond pulse to TRIG pin
digitalWrite(TRIG_PIN, HIGH);

delayMicroseconds(10);

digitalWrite(TRIG_PIN, LOW);

// measure duration of pulse from ECHO pin
duration_us = pulseIn(ECHO_PIN, HIGH);

// calculate the distance
distance_cm = 0.017 * duration_us;

if(distance_cm < DISTANCE_THRESHOLD)

    digitalWrite(LED_PIN, HIGH); // turn on LED

else

    digitalWrite(LED_PIN, LOW); // turn off LED

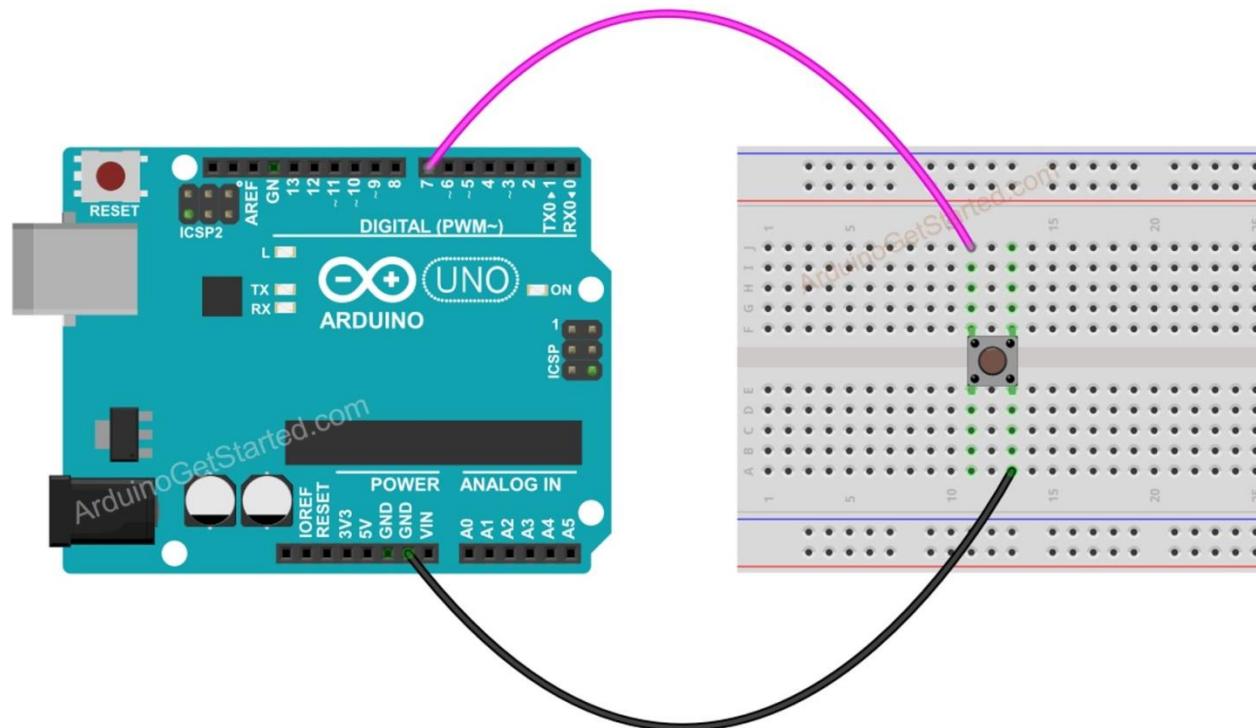
// print the value to Serial Monitor
Serial.print("distance: ");

Serial.print(distance_cm);
```

```
Serial.println(" cm");  
delay(500);  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

ARDUINO BUTTON LONG PRESS SHORT PRESS



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

// constants won't change. They're used here to set pin numbers:

```
const int BUTTON_PIN = 7; // the number of the pushbutton pin
```

```
const int SHORT_PRESS_TIME = 500; // 500 milliseconds
```

// Variables will change:

```
int lastState = LOW; // the previous state from the  
input pin
```

```
int currentState; // the current reading from the input  
pin
```

```
unsigned long pressedTime = 0;
```

```
unsigned long releasedTime = 0;
```

```
void setup() {
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
Serial.begin(9600);
```

```
pinMode(BUTTON_PIN, INPUT_PULLUP);
```

```
}
```

```
void loop() {
```

```
// read the state of the switch/button:
```

```
currentState = digitalRead(BUTTON_PIN);
```

```
if(lastState == HIGH && currentState == LOW)      //  
button is pressed  
  
pressedTime = millis();  
  
else if(lastState == LOW && currentState == HIGH) { //  
button is released  
  
releasedTime = millis();  
  
  
  
long pressDuration = releasedTime - pressedTime;  
  
  
if( pressDuration < SHORT_PRESS_TIME )  
    Serial.println("A short press is detected");  
  
}  
  
  
  
// save the the last state  
  
lastState = currentState;  
  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
// constants won't change. They're used here to set pin  
numbers:  
  
const int BUTTON_PIN = 7; // the number of the  
pushbutton pin  
  
const int LONG_PRESS_TIME = 1000; // 1000  
milliseconds
```

```
// Variables will change:
```

```
int lastState = LOW; // the previous state from the  
input pin
```

```
int currentState; // the current reading from the input  
pin
```

```
unsigned long pressedTime = 0;
```

```
unsigned long releasedTime = 0;
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
    pinMode(BUTTON_PIN, INPUT_PULLUP);
```

```
}
```

```
void loop() {
```

```
    // read the state of the switch/button:
```

```
    currentState = digitalRead(BUTTON_PIN);
```

```
if(lastState == HIGH && currentState == LOW)      //  
button is pressed  
  
    pressedTime = millis();  
  
else if(lastState == LOW && currentState == HIGH) { //  
button is released  
  
    releasedTime = millis();  
  
  
  
long pressDuration = releasedTime - pressedTime;  
  
  
  
if( pressDuration > LONG_PRESS_TIME )  
  
    Serial.println("A long press is detected");  
  
}  
  
  
  
// save the the last state  
  
lastState = currentState;  
  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
// constants won't change. They're used here to set pin  
numbers:
```

```
const int BUTTON_PIN = 7; // the number of the  
pushbutton pin
```

```
const int LONG_PRESS_TIME = 1000; // 1000  
milliseconds
```

```
// Variables will change:
```

```
int lastState = LOW; // the previous state from the  
input pin
```

```
int currentState; // the current reading from the input  
pin
```

```
unsigned long pressedTime = 0;
```

```
bool isPressing = false;
```

```
bool isLongDetected = false;
```

```
void setup() {
```

```
  Serial.begin(9600);
```

```
pinMode(BUTTON_PIN, INPUT_PULLUP);

}

void loop() {

    // read the state of the switch/button:
    currentState = digitalRead(BUTTON_PIN);

    if(lastState == HIGH && currentState == LOW) {      // button is pressed
        pressedTime = millis();
        isPressing = true;
        isLongDetected = false;
    } else if(lastState == LOW && currentState == HIGH) {
        // button is released
        isPressing = false;
    }
}
```

```
if(isPressing == true && isLongDetected == false) {  
    long pressDuration = millis() - pressedTime;  
  
    if( pressDuration > LONG_PRESS_TIME ) {  
        Serial.println("A long press is detected");  
        isLongDetected = true;  
    }  
}  
  
// save the the last state  
lastState = currentState;  
}
```

DETECTION OF SHORT PRESS AND LONG PRESS

// constants won't change. They're used here to set pin numbers:

```
const int BUTTON_PIN = 7; // the number of the pushbutton pin
```

```
const int SHORT_PRESS_TIME = 1000; // 1000 milliseconds
```

```
const int LONG_PRESS_TIME = 1000; // 1000 milliseconds
```

// Variables will change:

```
int lastState = LOW; // the previous state from the input pin
```

```
int currentState; // the current reading from the input pin
```

```
unsigned long pressedTime = 0;
```

```
unsigned long releasedTime = 0;
```

```
bool isPressing = false;
```

```
bool isLongDetected = false;

void setup() {
    Serial.begin(9600);
    pinMode(BUTTON_PIN, INPUT_PULLUP);
}

void loop() {
    // read the state of the switch/button:
    currentState = digitalRead(BUTTON_PIN);

    if(lastState == HIGH && currentState == LOW) {      //
        button is pressed
        pressedTime = millis();
        isPressing = true;
        isLongDetected = false;
    }
}
```

```
 } else if(lastState == LOW && currentState == HIGH) {  
 // button is released  
  
 isPressing = false;  
  
 releasedTime = millis();  
  
  
  
  
 long pressDuration = releasedTime - pressedTime;  
  
  
  
  
 if( pressDuration < SHORT_PRESS_TIME )  
  
 Serial.println("A short press is detected");  
  
 }  
  
  
  
  
 if(isPressing == true && isLongDetected == false) {  
  
 long pressDuration = millis() - pressedTime;  
  
  
  
  
 if( pressDuration > LONG_PRESS_TIME ) {  
  
 Serial.println("A long press is detected");  
  
 isLongDetected = true;
```

```
}
```

```
}
```

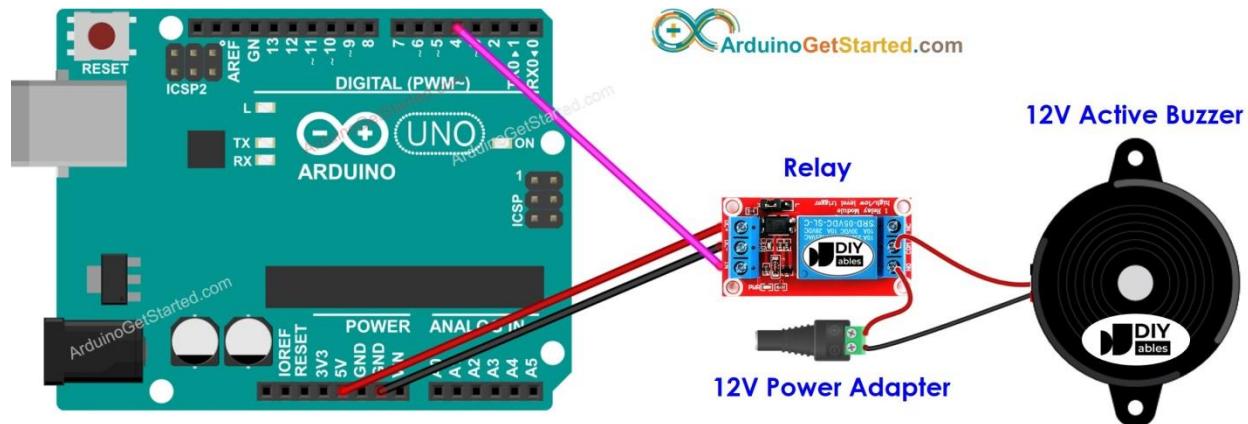
```
// save the the last state
```

```
lastState = currentState;
```

```
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

Arduino Buzzer



```
#define RELAY_PIN 4 // the Arduino pin that controls the
buzzer via relay

// the setup function runs once when you press reset or
power the board

void setup() {

    // initialize digital pin D4 as an output.

    pinMode(RELAY_PIN, OUTPUT);

}

// the loop function runs over and over again forever

void loop() {

    digitalWrite(RELAY_PIN, HIGH); // turn on buzzer 2
seconds

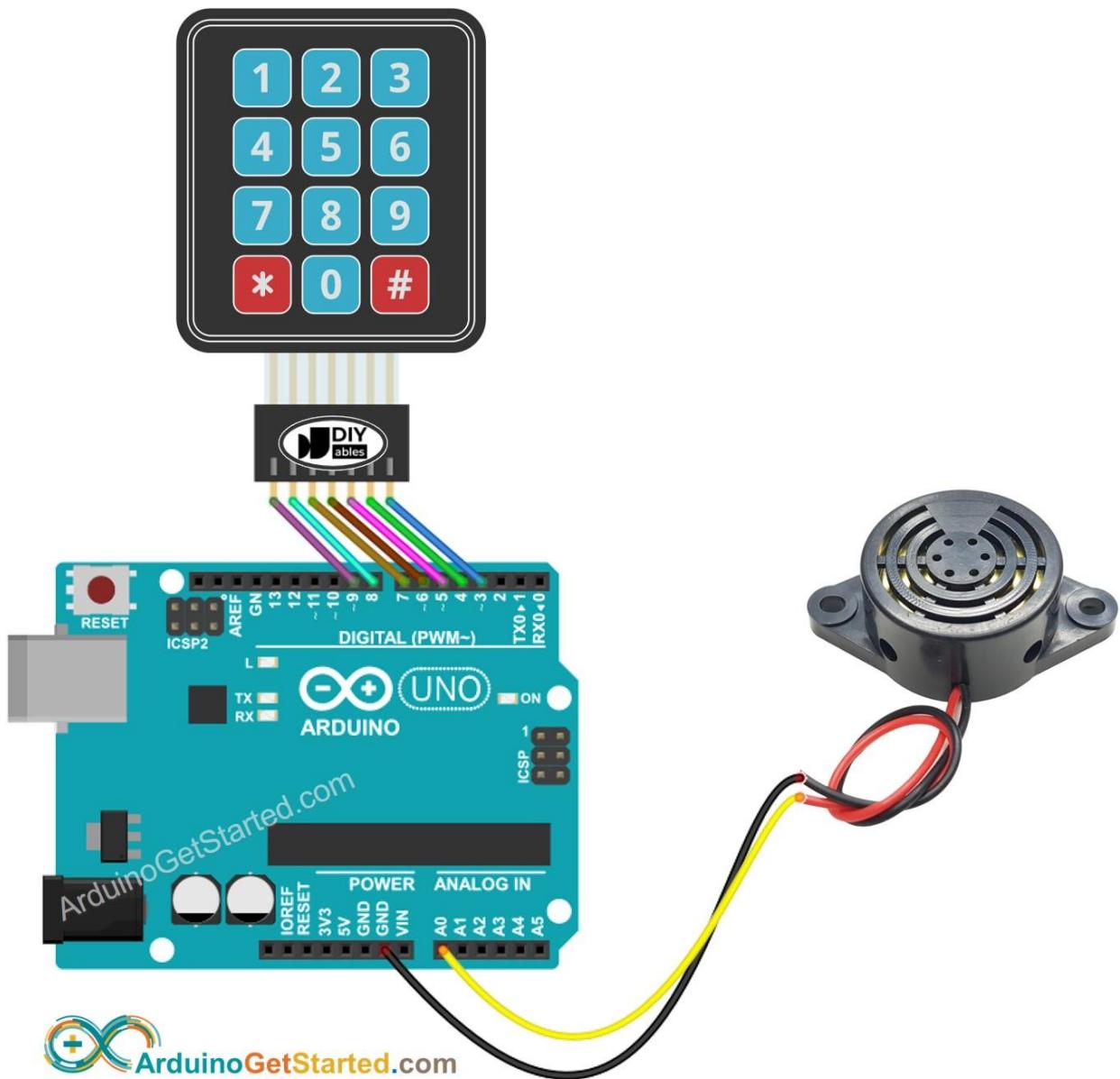
    delay(2000);

    digitalWrite(RELAY_PIN, LOW); // turn off buzzer 5
seconds

    delay(5000);

}
```

ARDUINO KEYPAD BUZZOR



```
#include <Keypad.h>

#include <ezBuzzer.h>

#define BUZZER_PIN A0

#define ROW_NUM 4 // four rows

#define COLUMN_NUM 3 // three columns

char keys[ROW_NUM][COLUMN_NUM] = {

    {'1','2','3'},
    {'4','5','6'},
    {'7','8','9'},
    {'*','0','#'}
};

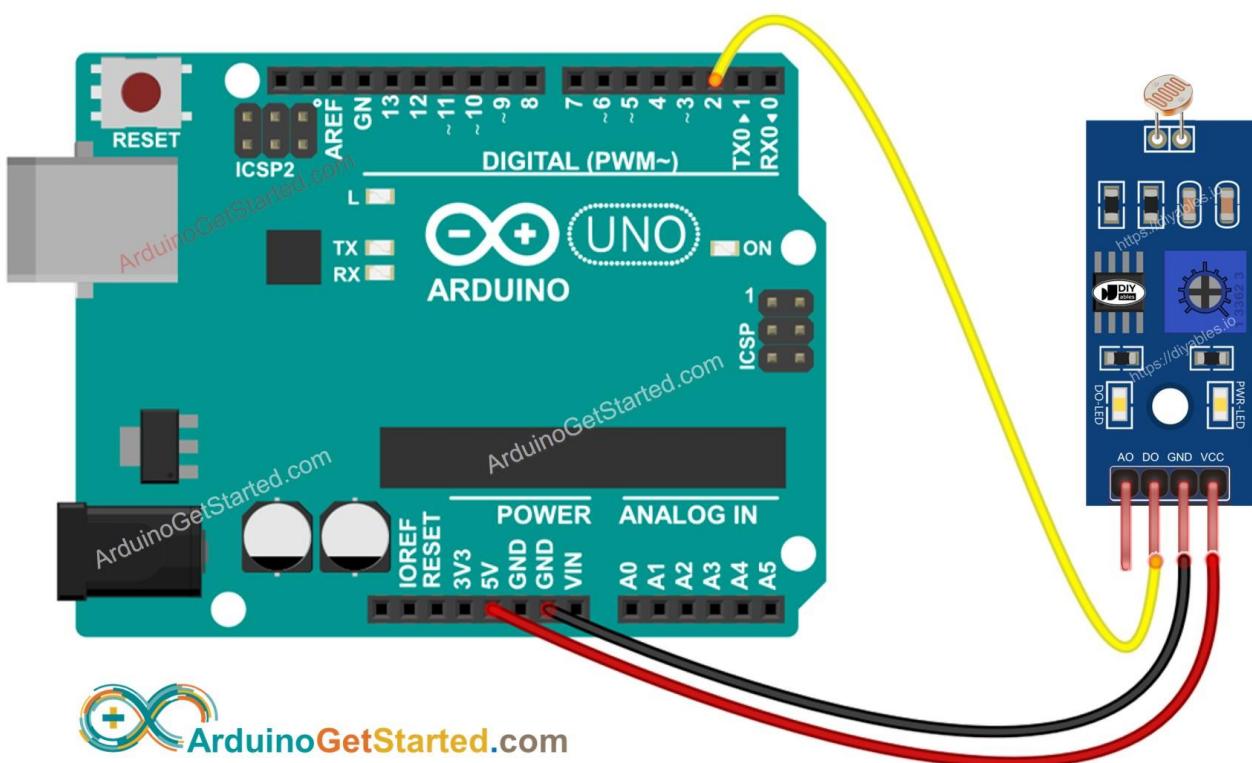
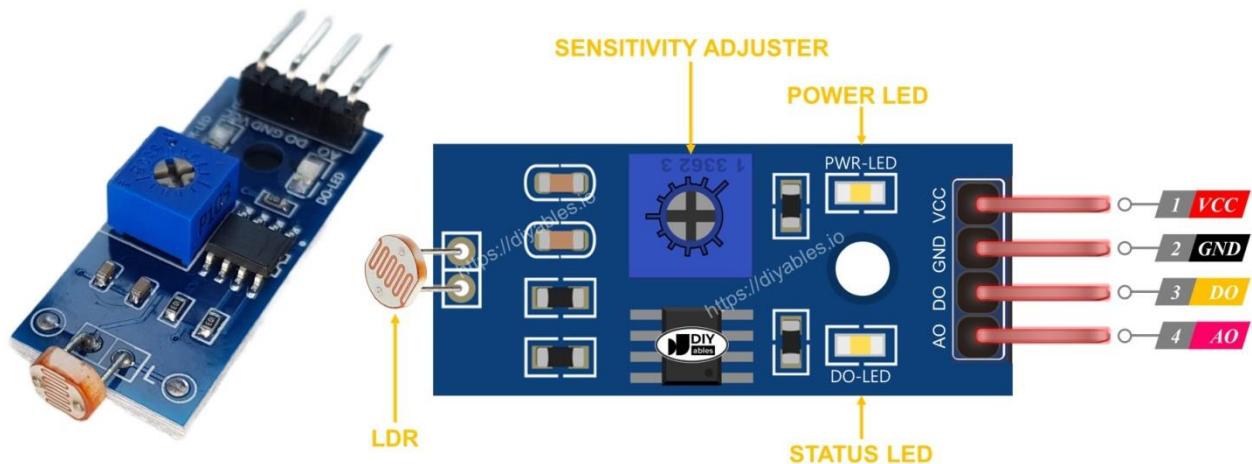
byte pin_rows[ROW_NUM] = {9, 8, 7, 6}; // connect to
the row pinouts of the keypad

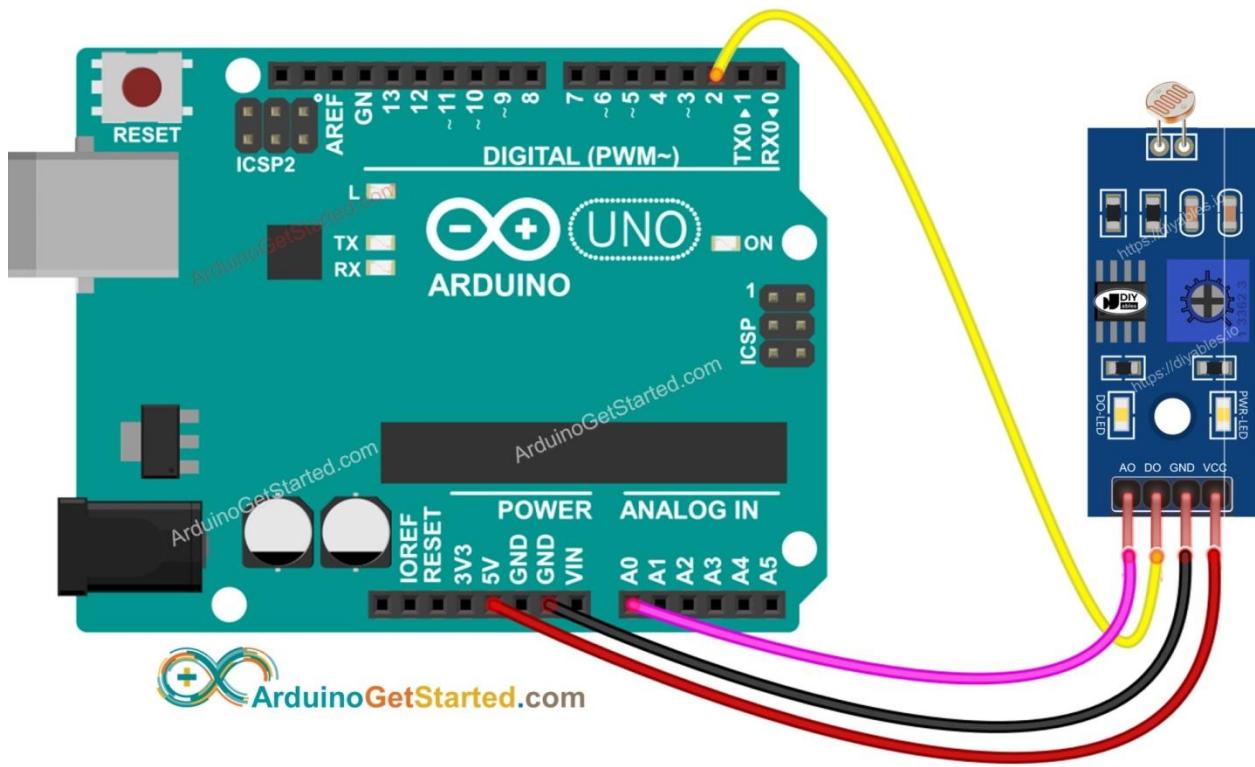
byte pin_column[COLUMN_NUM] = {5, 4, 3}; // connect
to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), pin_rows,
pin_column, ROW_NUM, COLUMN_NUM );
```

```
ezBuzzer buzzer(BUZZER_PIN); // create ezBuzzer object  
that attach to a pin.  
  
void setup()  
{  
    Serial.begin(9600);  
}  
  
void loop() {  
    buzzer.loop(); // MUST call the buzzer.loop() function in  
loop()  
    char key = keypad.getKey();  
    if (key) {  
        Serial.print(key); // prints key to serial monitor  
        buzzer.beep(200); // generates a 200ms short sound  
    }  
}
```

ARDUINO LDR MODULE





ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
#define DO_PIN 2 // Arduino's pin connected to DO pin
of the ldr module

void setup()
{
    // initialize serial communication
    Serial.begin(9600);

    // initialize the Arduino's pin as an input
    pinMode(DO_PIN, INPUT);
```

```
}
```

```
void loop()
```

```
{
```

```
    int lightState = digitalRead(DO_PIN);
```

```
    if (lightState == HIGH)
```

```
        Serial.println("The light is NOT present");
```

```
    else
```

```
        Serial.println("The light is present");
```

```
}
```

```
#define AO_PIN A0

// Arduino's pin connected to AO pin of the ldr module

void setup()

{

    // initialize serial communication

    Serial.begin(9600);

}

void loop()

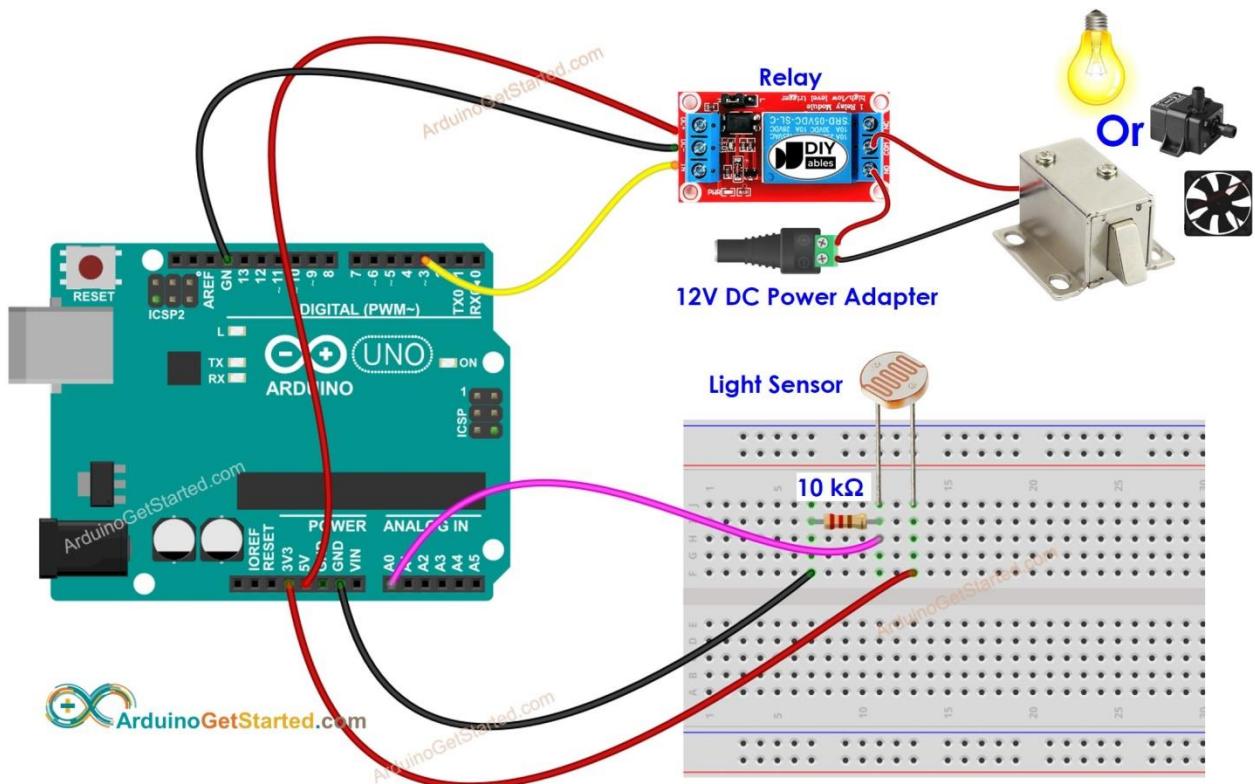
{

    int lightValue = analogRead(AO_PIN);

    Serial.println(lightValue);

}
```

ARDUINO LIGHT SENSOR TRIGGERS RELAY



```
// constants won't change

const int LIGHT_SENSOR_PIN = A0; // Arduino pin
connected to light sensor's pin

const int RELAY_PIN      = 3; // Arduino pin connected to
Relay's pin

const int ANALOG_THRESHOLD = 500;

// variables will change:

int analogValue;

void setup() {

    pinMode(RELAY_PIN, OUTPUT); // set arduino pin to
output mode

}

void loop()

{

analogValue = analogRead(LIGHT_SENSOR_PIN); // read
the input on analog pin

if(analogValue < ANALOG_THRESHOLD)
```

```
digitalWrite(RELAY_PIN, HIGH); // turn on Relay
```

```
else
```

```
digitalWrite(RELAY_PIN, LOW); // turn off Relay
```

```
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

TEMPERATURE SENSOR

EXAMPLE – LM35

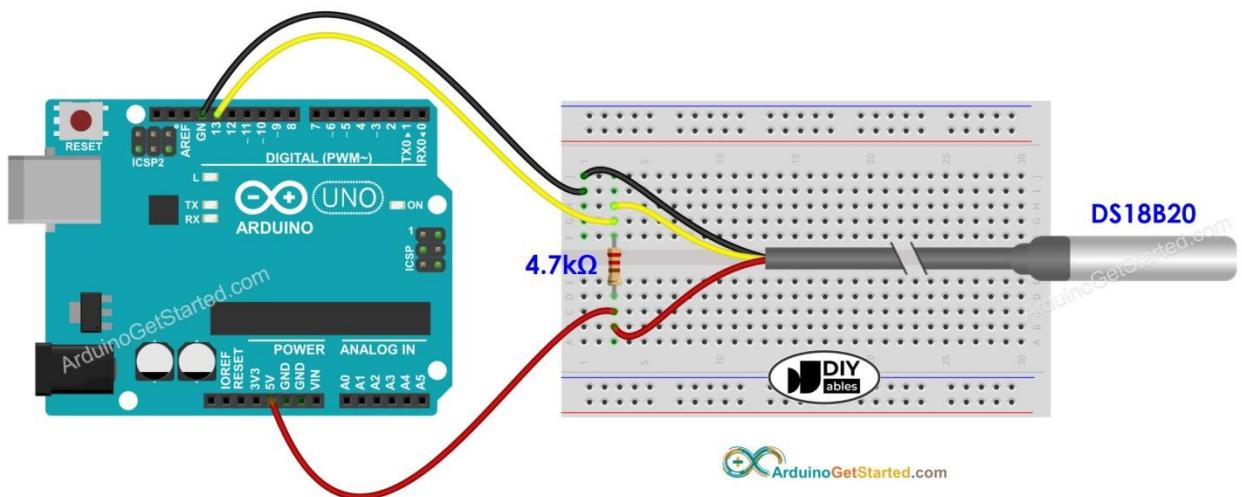
TH02

HDC1000HTS221

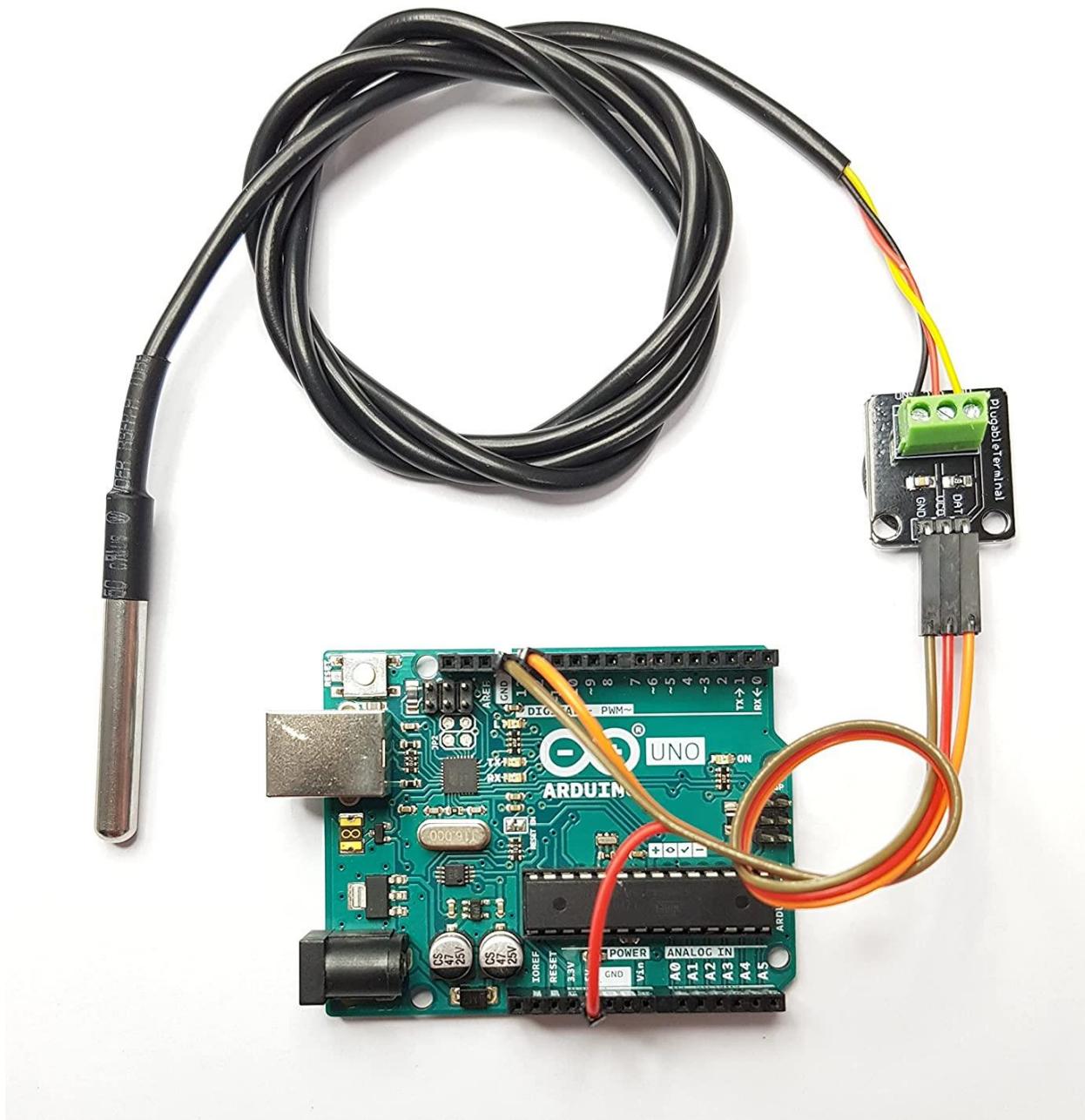
DHTxx

DS18B20(WATERPROOF TEMPERATURE SENSOR)





ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK



```
#include <OneWire.h>

#include <DallasTemperature.h>

const int SENSOR_PIN = 13;

// Arduino pin connected to DS18B20 sensor's DQ pin

OneWire oneWire(SENSOR_PIN);

// setup a oneWire instance

DallasTemperature tempSensor(&oneWire);

// pass oneWire to DallasTemperature library

float tempCelsius; // temperature in Celsius

float tempFahrenheit; // temperature in Fahrenheit

void setup()

{

    Serial.begin(9600); // initialize serial

    tempSensor.begin(); // initialize the sensor

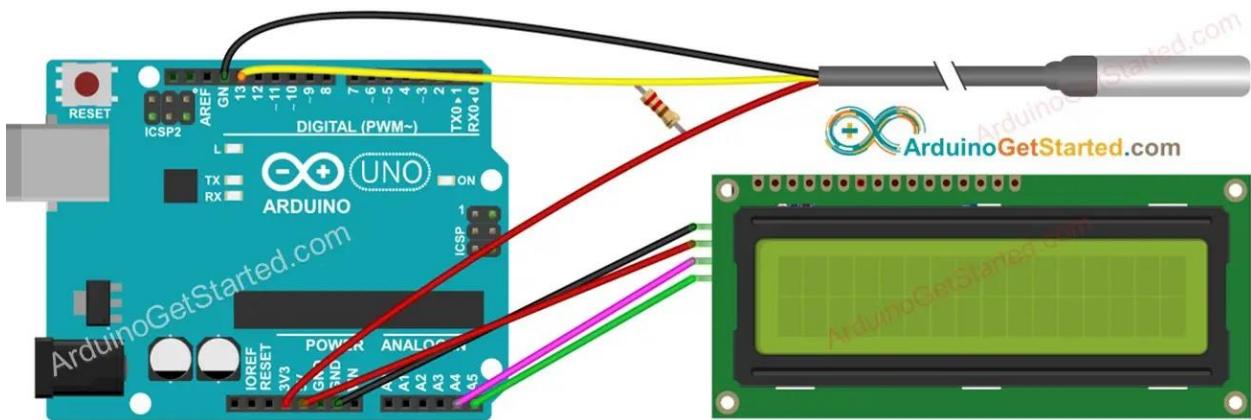
}

void loop()

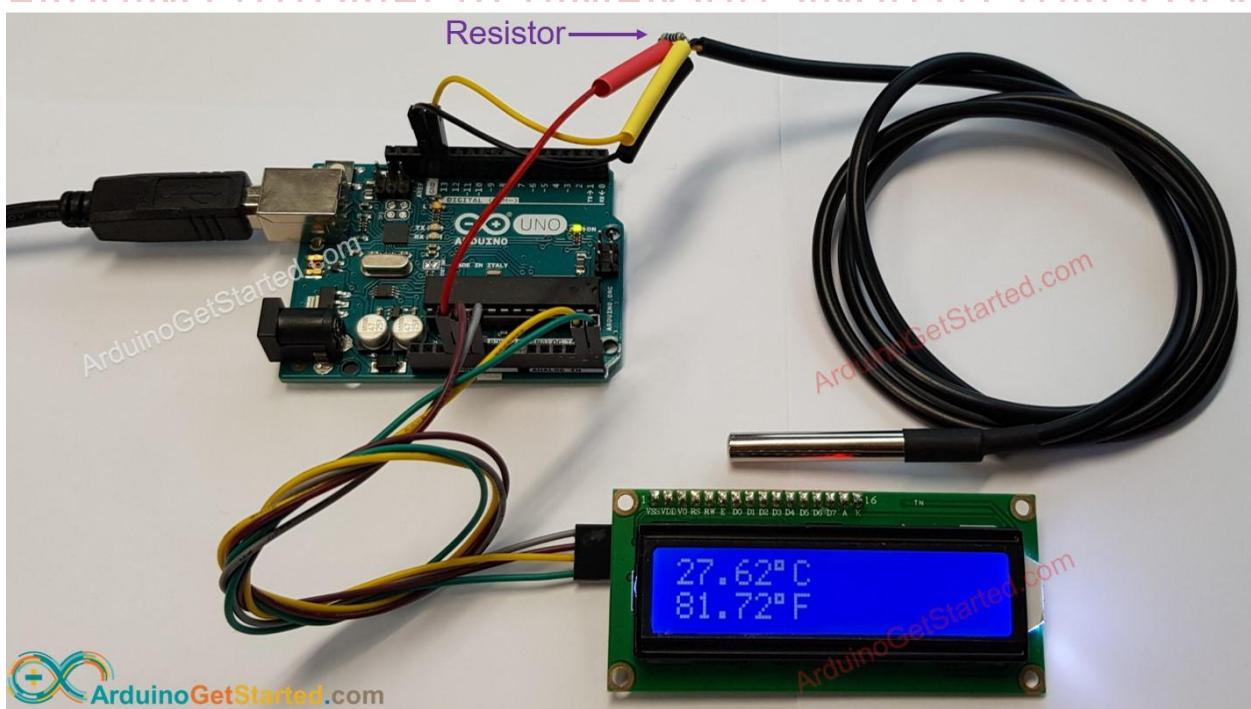
{
```

```
tempSensor.requestTemperatures();  
  
// send the command to get temperatures  
  
tempCelsius = tempSensor.getTempCByIndex(0);  
  
// read temperature in Celsius  
  
tempFahrenheit = tempCelsius * 9 / 5 + 32;  
  
// convert Celsius to Fahrenheit  
  
Serial.print("Temperature: ");  
  
Serial.print(tempCelsius);  
  
// print the temperature in Celsius  
  
Serial.print("°C");  
  
Serial.print(" ~ ");  
  
// separator between Celsius and Fahrenheit  
  
Serial.print(tempFahrenheit);  
  
// print the temperature in Fahrenheit  
  
Serial.println("°F");  
  
delay(500);  
  
}
```

TEMPERATURE SENSOR LCD



ARDUINO TUTORIAL BY HIMANSHU SINGH FOR GPK



```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal_I2C.h>

const int SENSOR_PIN = 13;
// Arduino pin connected to DS18B20 sensor's DQ pin

OneWire oneWire(SENSOR_PIN);
// setup a oneWire instance

DallasTemperature sensors(&oneWire);
// pass oneWire to DallasTemperature library

LiquidCrystal_I2C lcd(0x27, 16, 2);
// I2C address 0x27 (from DIYables LCD), 16 column and
2 rows

float tempCelsius;
// temperature in Celsius
```

```
float tempFahrenheit;  
// temperature in Fahrenheit  
  
void setup()  
{  
    sensors.begin();  
    // initialize the sensor  
    lcd.init();  
    // initialize the lcd  
    lcd.backlight();  
    // open the backlight  
}  
  
void loop()  
{  
    sensors.requestTemperatures();  
    // send the command to get temperatures
```

```
tempCelsius = sensors.getTempCByIndex(0);  
  
// read temperature in Celsius  
  
tempFahrenheit = tempCelsius * 9 / 5 + 32;  
  
// convert Celsius to Fahrenheit  
  
  
lcd.clear();  
  
lcd.setCursor(0, 0);  
  
// start to print at the first row  
  
lcd.print(tempCelsius);  
  
// print the temperature in Celsius  
  
lcd.print((char)223);  
  
// print ° character  
  
lcd.print("C");  
  
lcd.setCursor(0, 1);  
  
// start to print at the second row  
  
lcd.print(tempFahrenheit);  
  
// print the temperature in Fahrenheit
```

```
lcd.print((char)223);  
// print ° character  
lcd.print("F");  
delay(500);  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

ARDUINO TEMPERATURE SENSOR DHT11

About DHT11 Temperature and Humidity Sensor

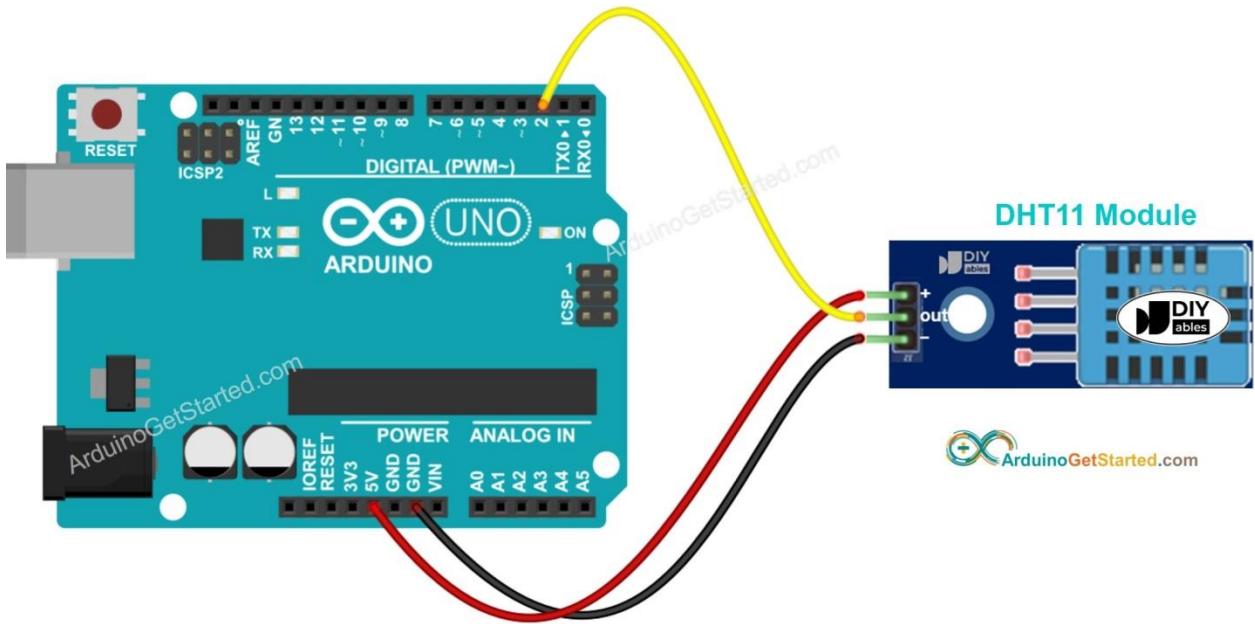
DHT11	
Operating Voltage	3 to 5V
Temperature Range	0°C to 50°C
Temperature Accuracy	± 2°C
Humidity Range	20% to 80%
Humidity Accuracy	5%
Reading Rate	1Hz (once every second)

- ◆ **GND pin:** needs to be connected to **GND** (0V)
- ◆ **VCC pin:** needs to be connected to **VCC** (5V, or 3.3V)
- ◆ **DATA pin:** the pin is used to communicate between the sensor and Arduino
- ◆ **NC pin:** Not connected, we can ignore this pin

DHT11 module has three pins:

- ◆ **GND pin:** needs to be connected to **GND** (0V)
- ◆ **VCC pin:** needs to be connected to **VCC** (5V, or 3.3V)
- ◆ **DATA pin:** the pin is used to communicate between the sensor and Arduino

Some manufacturers provide DHT11 sensor in module form with three pins: **GND**, **VCC** and DATA pins (or alternatively: -, +, and OUT pins).



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
#include "DHT.h"

#define DHT11_PIN 2

DHT dht11(DHT11_PIN, DHT11);

void setup()
{
    Serial.begin(9600);
    dht11.begin();
    // initialize the sensor
}

void loop()
{
    // wait a few seconds between measurements.
    delay(2000);
    // read humidity
    float humi = dht11.readHumidity();
    // read temperature as Celsius
```

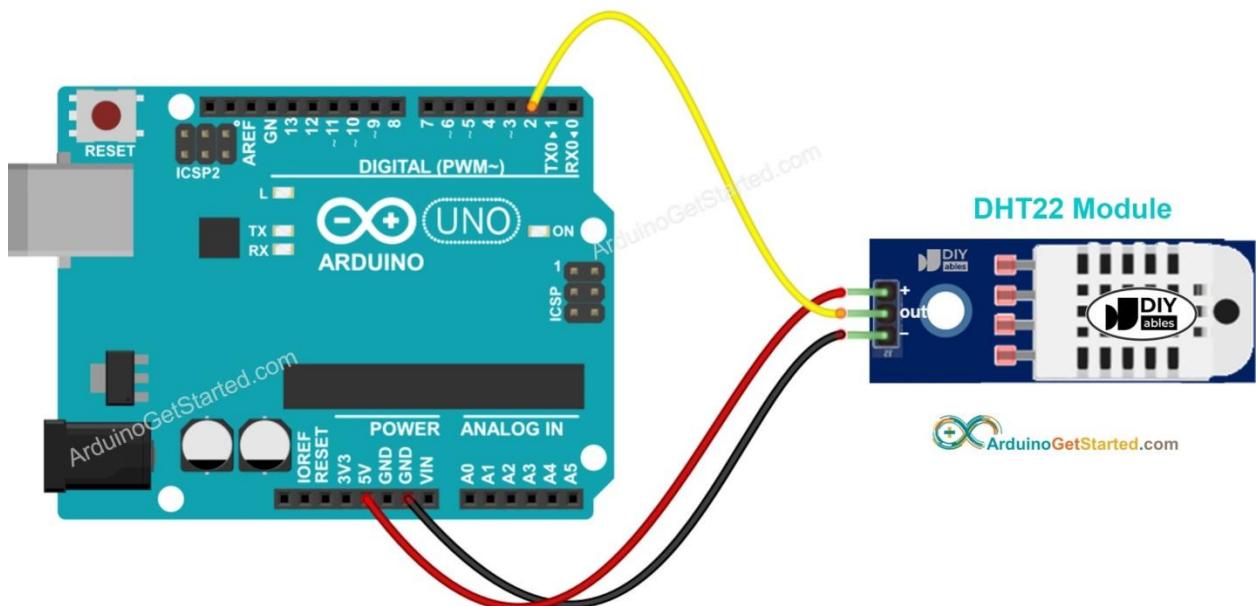
```
float tempC = dht11.readTemperature();  
  
// read temperature as Fahrenheit  
  
float tempF = dht11.readTemperature(true);  
  
// check if any reads failed  
  
if (isnan(humi) || isnan(tempC) || isnan(tempF))  
{  
    Serial.println("Failed to read from DHT11 sensor!");  
}  
  
Else  
{  
    Serial.print("DHT11# Humidity: ");  
    Serial.print(humi);  
    Serial.print("%");  
    Serial.print(" | ");  
    Serial.print("Temperature: ");  
    Serial.print(tempC);  
    Serial.print("°C ~ ");
```

```
Serial.print(tempF);  
Serial.println("°F");  
}  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

The differences between DHT11 and DHT22

	DHT11	DHT22
Price	ultra low cost	low cost
Temperature Range	0°C to 50°C	-40°C to 80°C
Temperature Accuracy	± 2°C	± 0.5°C
Humidity Range	20% to 80%	0% to 100%
Humidity Accuracy	5%	± 2 to 5%
Reading Rate	1Hz (once every second)	0.5Hz (once every 2 seconds)
Body size	15.5mm x 12mm x 5.5mm	15.1mm x 25mm x 7.7mm
Operating Voltage	3 to 5V	3 to 5V



```
#include "DHT.h"

#define DHTPIN 2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

void setup()
{
    Serial.begin(9600);
    dht.begin();
    // initialize the sensor
}

void loop()
{
    // wait a few seconds between measurements.
    delay(2000);
    // read humidity
```

```
float humi = dht.readHumidity();  
  
// read temperature as Celsius  
  
float tempC = dht.readTemperature();  
  
// read temperature as Fahrenheit  
  
float tempF = dht.readTemperature(true);  
  
  
  
// check if any reads failed  
  
if (isnan(humi) || isnan(tempC) || isnan(tempF))  
  
{  
  
    Serial.println("Failed to read from DHT sensor!");  
  
}  
  
Else  
  
{  
  
    Serial.print("Humidity: ");  
  
    Serial.print(humi);  
  
    Serial.print("%");  
  
}
```

```
Serial.print(" | ");
Serial.print("Temperature: ");
Serial.print(tempC);
Serial.print("°C ~ ");
Serial.print(tempF);
Serial.println("°F");
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
#include "DHT.h"

#define DHTPIN 2

#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);

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

    dht.begin(); // initialize the sensor

}

void loop()
{
    // wait a few seconds between measurements.

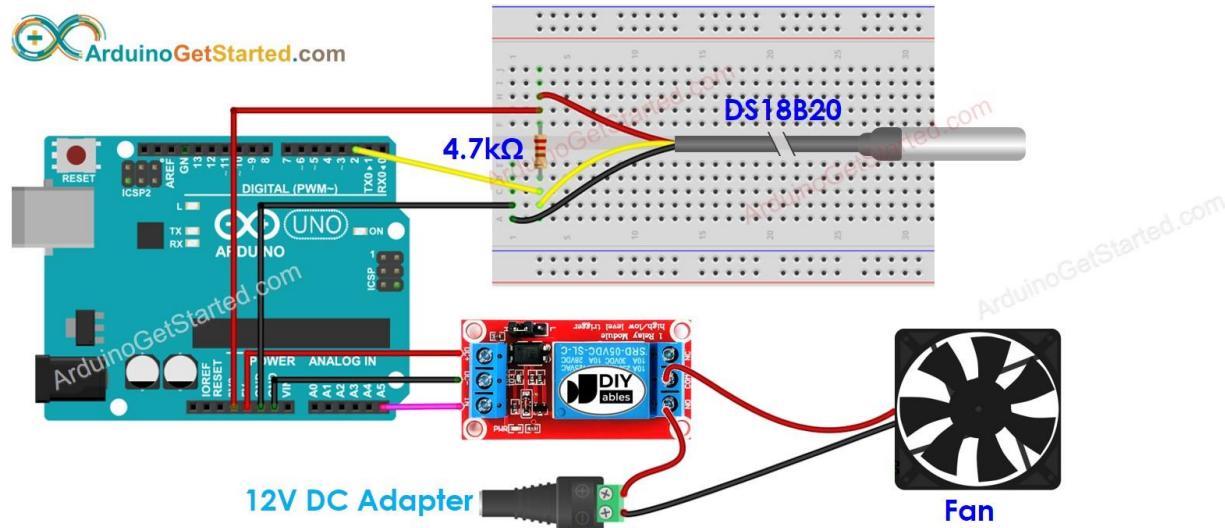
    delay(2000);
}
```

```
// read humidity  
  
float humi = dht.readHumidity();  
  
// read temperature as Celsius  
  
float tempC = dht.readTemperature();  
  
// read temperature as Fahrenheit  
  
float tempF = dht.readTemperature(true);
```

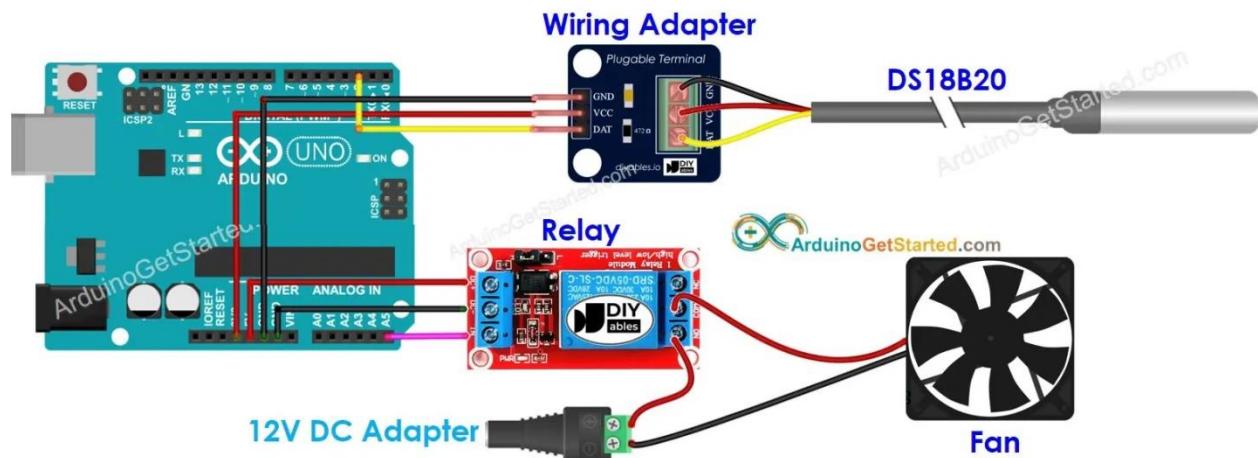
```
// check if any reads failed  
  
if (isnan(humi) || isnan(tempC) || isnan(tempF)) {  
  
    Serial.println("Failed to read from DHT sensor!");  
  
}  
  
Else  
  
{  
  
    Serial.print("Humidity: ");  
  
    Serial.print(humi);  
  
    Serial.print("%");
```

```
Serial.print(" | ");  
  
Serial.print("Temperature: ");  
  
Serial.print(tempC);  
  
Serial.print("°C ~ ");  
  
Serial.print(tempF);  
  
Serial.println("°F");  
  
}  
  
}
```

COOLING SYSTEM USING DS18B20 TEMPERATURE SENSOR



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK



```
/*
```

* Created by ArduinoGetStarted.com

*

* This example code is in the public domain

*

* Tutorial page:

<https://arduinogetstarted.com/tutorials/arduino-cooling-system-using-ds18b20-temperature-sensor>

*/

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURED GDK

```
#include <OneWire.h>
```

```
#include <DallasTemperature.h>
```

```
const int TEMP_THRESHOLD_UPPER = 25;
```

```
// upper threshold of temperature, change to your  
desire value
```

```
const int TEMP_THRESHOLD_LOWER = 20;
```

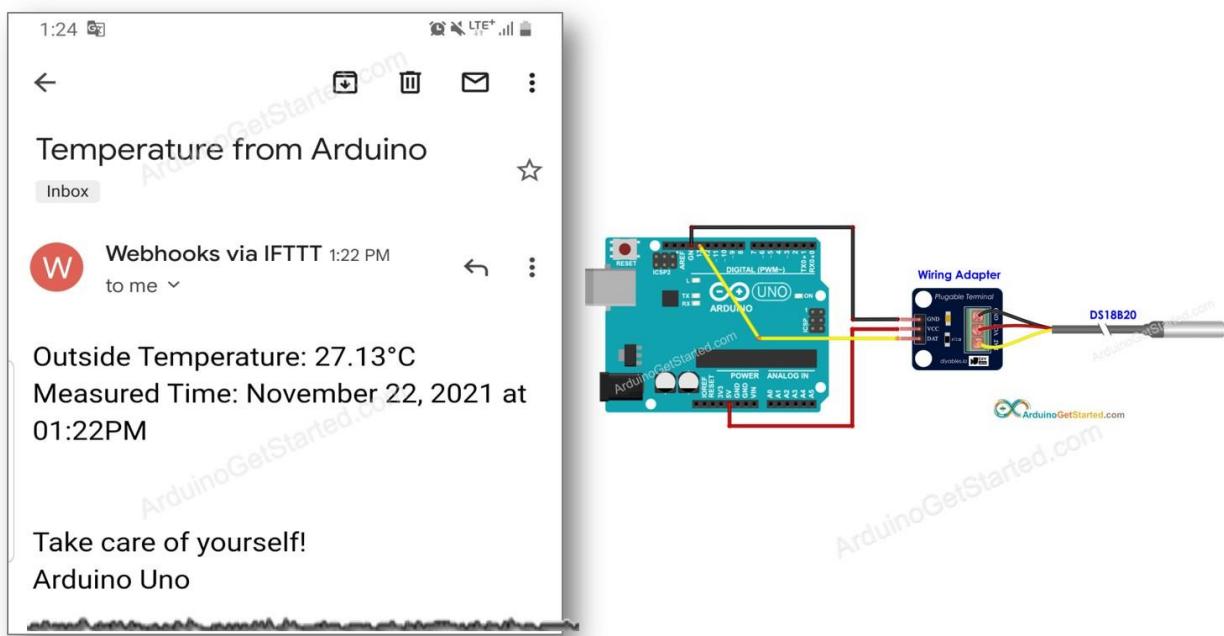
```
// lower threshold of temperature, change to your  
desire value
```

```
const int SENSOR_PIN = 2;  
// Arduino pin connected to DS18B20 sensor's DQ pin  
  
const int RELAY_FAN_PIN = A5;  
// Arduino pin connected to relay which connected to fan  
  
  
  
OneWire oneWire(SENSOR_PIN);  
// setup a OneWire instance  
  
DallasTemperature sensors(&oneWire);  
// pass oneWire to DallasTemperature library  
  
  
  
float temperature;  
// temperature in Celsius  
  
  
  
void setup()  
{  
    Serial.begin(9600); // initialize serial
```

```
sensors.begin(); // initialize the sensor  
  
pinMode(RELAY_FAN_PIN, OUTPUT);  
  
// initialize digital pin as an output  
  
}  
  
void loop()  
{  
  
    sensors.requestTemperatures();  
  
    // send the command to get temperatures  
  
    temperature = sensors.getTempCByIndex(0);  
  
    // read temperature in Celsius  
  
  
    if(temperature > TEMP_THRESHOLD_UPPER)  
  
    {  
  
        Serial.println("The fan is turned on");  
  
        digitalWrite(RELAY_FAN_PIN, HIGH);  
  
        // turn on  
  
    }
```

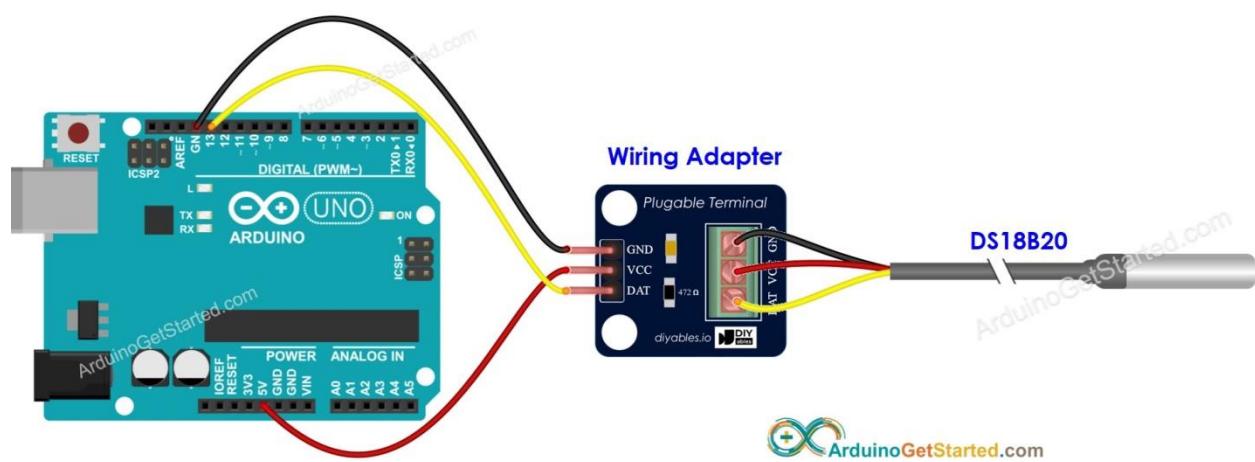
```
else
{
    if(temperature < TEMP_THRESHOLD_LOWER)
    {
        Serial.println("The fan is turned off");
        digitalWrite(RELAY_FAN_PIN, LOW);
        // turn on
    }
    delay(500);
}
```

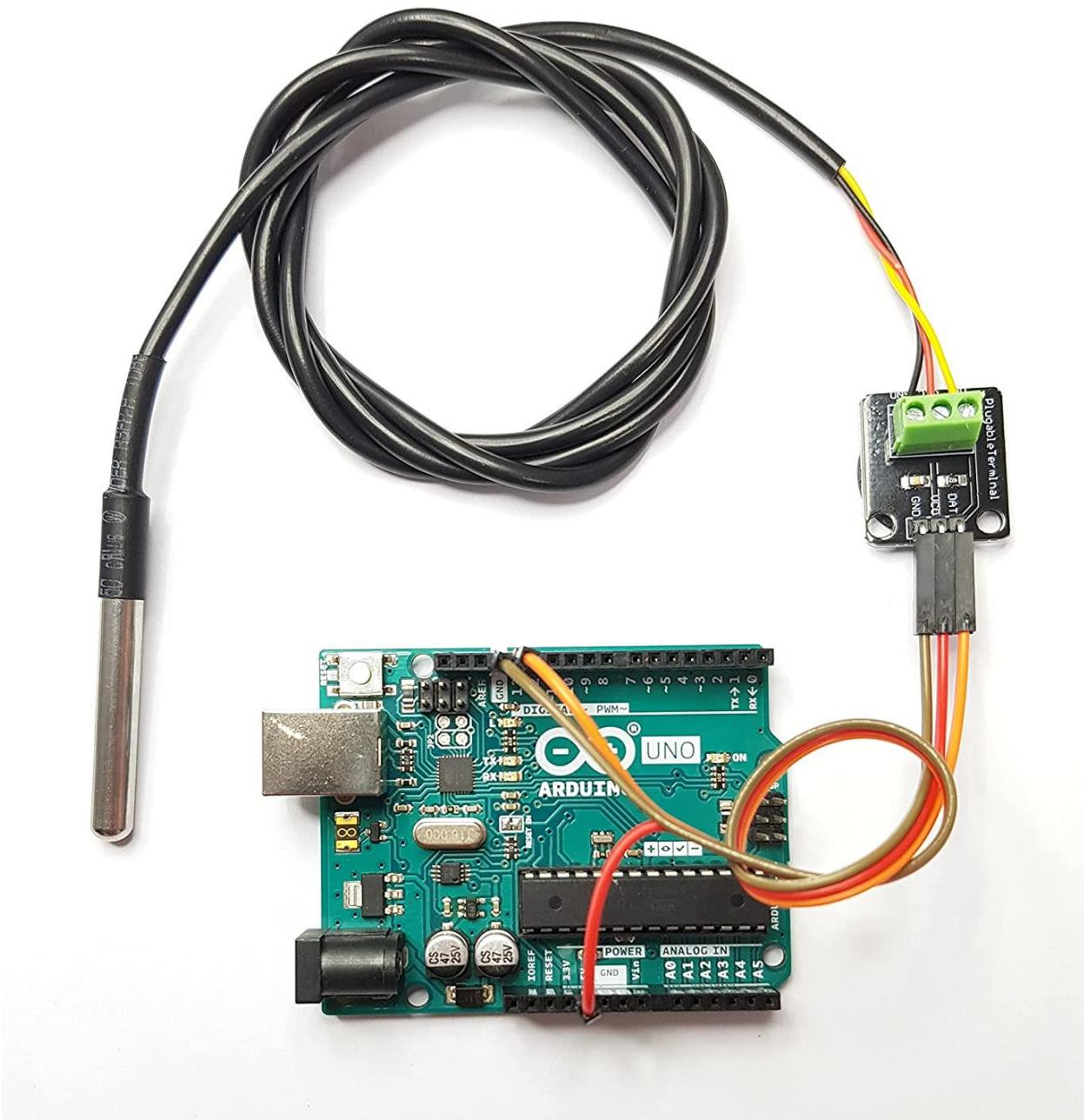
ARDUINO TEMPERATURE SEND E MAIL NOTIFICATION



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

ARDUINO UNO R4 WIFI





CREATE A APPLET ON IFTTT WEBSITEE

**WRITE ARDUINO CODE THAT ACCEPTS HTTP AS REQUEST
TO APPLET**

How it works

When the temperature exceeds a threshold:

- ◆ Arduino makes an HTTP request to IFTTT Applet we created,
- ◆ The IFTTT Applet will send an email to the email address.

The email address, email subject, and email content are specified when we create the IFTTT Applet.

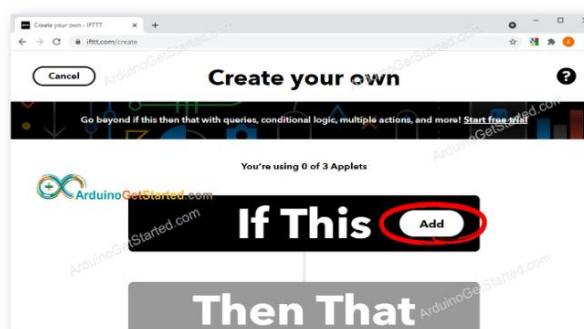
How To Create an IFTTT Applet

- ◆ Create an IFTTT account and Login to IFTTT.
- ◆ Go to IFTTT Home page and click the Create button



- ◆ Click the Add button

- ◆ Click the Add button



- ◆ Search for Webhooks, and then click the Webhooks icon

Arduino - Software Installation
Arduino - Hardware Preparation
Arduino - Hello World
Arduino - Code Structure
Arduino - Serial Monitor
Arduino - Serial Plotter

Arduino - LED - Blink
Arduino - LED - Blink Without Delay
Arduino - Blink multiple LED
Arduino - LED - Fade
Arduino - RGB LED
Arduino - Traffic Light

Arduino - Button
Arduino - Button - Debounce
Arduino - Button - Long Press Short Press
Arduino multiple Button

◆ Search for Webhooks, and then click the Webhooks icon

◆ Click the Receive a web request icon

ARDUINO TU

LECTURER GPK

◆ Click the Receive a web request icon

Choose a trigger

Webhooks

Receive a web request with a JSON payload

Receive a web request

Connect service

Webhooks

Connect

Type an event name. You can give any name you want and memorize it to use later. The event name is a part of URL that Arduino will make request to. In this tutorial, we use a name **send-email**. And then click Create trigger button

Complete trigger fields

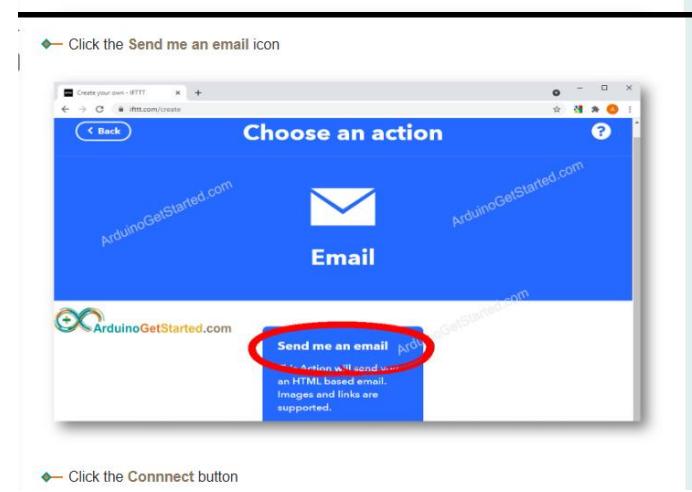
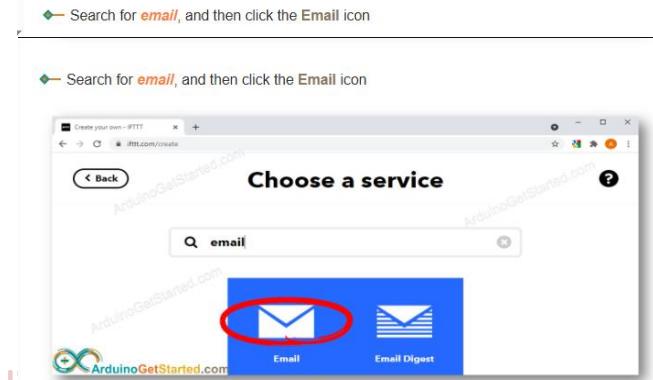
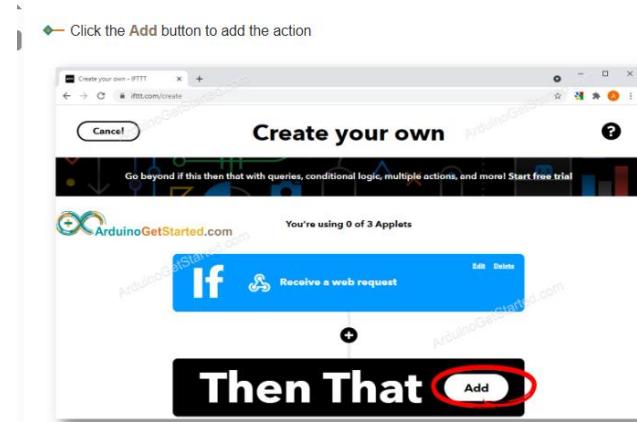
Receive a web request

Event Name

send-email

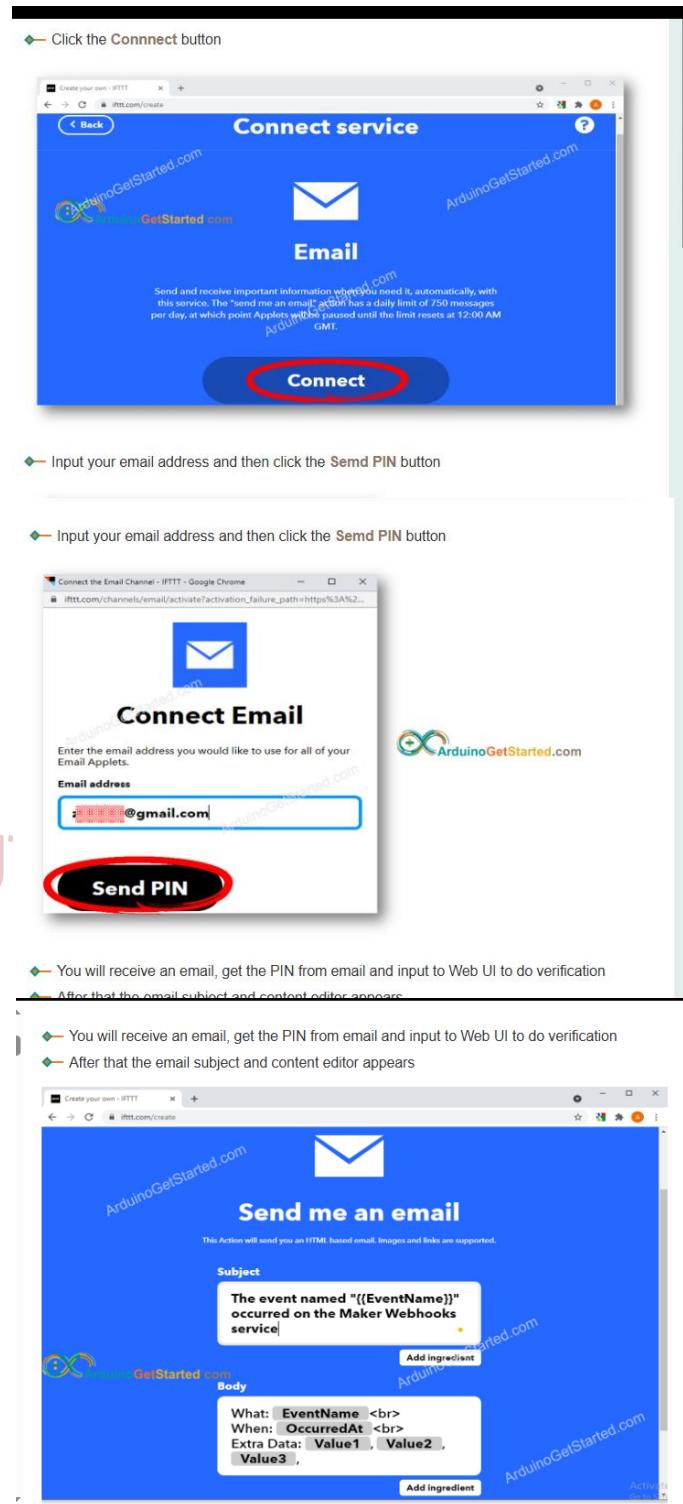
Create trigger

Arduino - Software Installation
Arduino - Hardware Preparation
Arduino - Hello World
Arduino - Code Structure
Arduino - Serial Monitor
Arduino - Serial Plotter
Arduino - LED - Blink
Arduino - LED - Blink Without Delay
Arduino - Blink multiple LED
Arduino - LED - Fade
Arduino - RGB LED
Arduino - Traffic Light
Arduino - Button
Arduino - Button - Debounce
Arduino - Button - Long Press Short Press
Arduino multiple Button

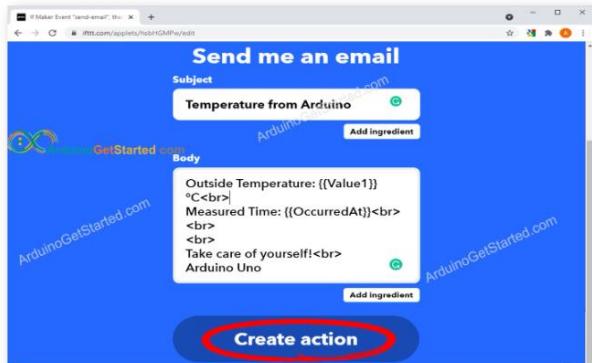


ARDUINO TU

LECTURER GPK



◆ You can give any subject and content for the email. The below are example



■ Subject:
■ Subject:

Temperature from Arduino

■ Content:

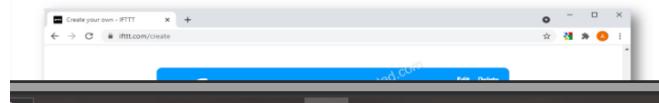
Outside Temperature: {{Value1}} °C

Measured Time: {{OccurredAt}}

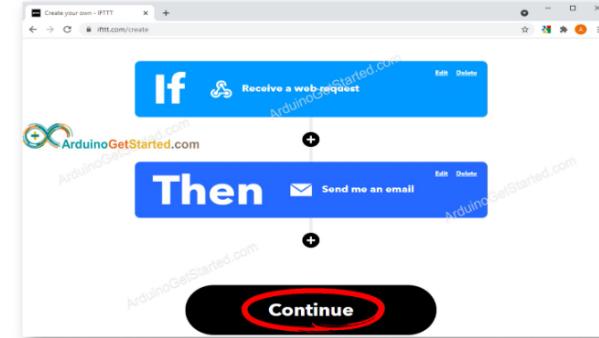
Take care of yourself!

Arduino Uno

- ◆ Click the Create Action button
- ◆ Click the Continue button



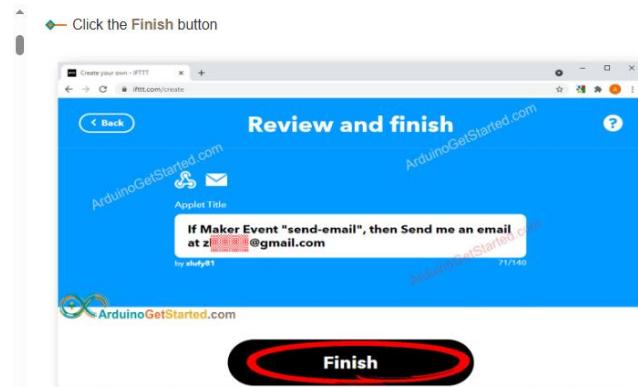
- ◆ Click the Create Action button
- ◆ Click the Continue button



- ◆ Click the Finish button

ARDUINO TU

ECTURER GPK



Now you succeeded to create an IFTTT [Applet](#) for your Arduino. The next step is to get the IFTTT [Webhooks key](#), which is used to authenticate and identify your Arduino.

Now you succeeded to create an IFTTT [Applet](#) for your Arduino. The next step is to get the IFTTT [Webhooks key](#), which is used to authenticate and identify your Arduino.

- ◆ Visit [IFTTT Webhooks page](#)
- ◆ Click the Documentation button



- ◆ You will see the [Webhooks key](#) as below

ARDUINO TU

LECTURER GPK



Copy and wire down your Webhooks to use on Arduino code.

Now, We just need to write Arduino code that makes an HTTP request to the below URL:

<https://maker.ifttt.com/trigger/send-email/with/key/XXXXXXXXXXXXXXXXXXXX>

Now, We just need to write Arduino code that makes an HTTP request to the below URL:

<https://maker.ifttt.com/trigger/send-email/with/key/XXXXXXXXXXXXXXXXXXXX>

* NOTE THAT:

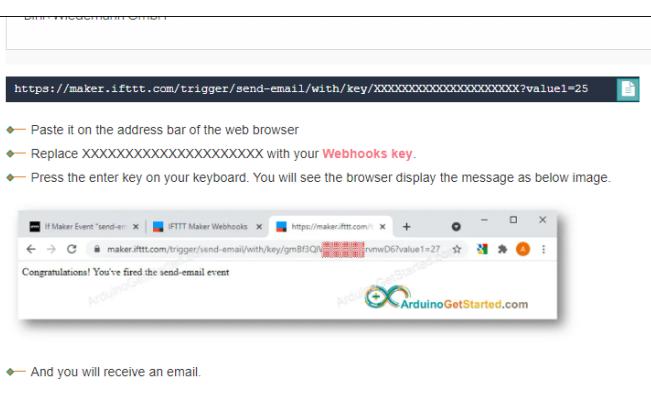
- ◆ In the URL, the **send-email** is the event name we used:
 - If you already used this name for another **Applet**, you can give it any other name.
 - If you use another name, please replace the **send-email** by your event name
- ◆ **Webhooks key** is generated by IFTTT. Please keep it secret. You can change it by regenerating it on the IFTTT website.

ARDUINO TU

LECTURER GPK

Before writing Arduino code, We can test the IFTTT **Applet** by doing:

- ◆ Open a web browser
- ◆ Copy the below link:



ARDUINO CODE

```
/*
 * Created by ArduinoGetStarted.com
 *
 * This example code is in the public domain
 *
 * Tutorial page:
 https://arduinogetstarted.com/tutorials/arduino-temperature-send-email-notification
 */
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
#include <WiFiS3.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define TEMP_THRESHOLD 27 // °C
#define SENSOR_PIN 13
```

```
// Arduino pin connected to DS18B20 sensor's DQ pin
```

```
const char ssid[] = "YOUR_WIFI_SSID";
```

```
// change your network SSID (name)
```

```
const char pass[] = "YOUR_WIFI_PASSWORD";
```

```
// change your network password (use for WPA, or use  
as key for WEP)
```

```
OneWire oneWire(SENSOR_PIN);
```

```
// setup a oneWire instance
```

```
DallasTemperature DS18B20(&oneWire);
```

```
// pass oneWire to DallasTemperature library
```

```
WiFiClient client;
```

```
int status = WL_IDLE_STATUS;
```

```
int HTTP_PORT = 80;
```

```
String HTTP_METHOD = "GET";
```

```
char HOST_NAME[] = "maker.ifttt.com";  
  
String PATH_NAME = "/trigger/send-  
email/with/key/XXXXXXXXXXXXXXXXXXXXXX";  
  
// change your Webhooks key  
  
bool isSent = false;  
  
void setup()  
{  
    Serial.begin(9600);  
  
    Serial.println("TEMPERATURE MONITORING via EMAIL  
STARTED!");  
  
    // check for the WiFi module:  
  
    if (WiFi.status() == WL_NO_MODULE)  
    {  
        Serial.println("Communication with WiFi module  
failed!");  
    }  
}
```

```
// don't continue

while (true)

;

}

String fv = WiFi.firmwareVersion();

if (fv < WIFI_FIRMWARE_LATEST_VERSION)

{

    Serial.println("Please upgrade the firmware");

}

// attempt to connect to WiFi network:

while (status != WL_CONNECTED)

{

    Serial.print("Attempting to connect to SSID: ");

    Serial.println(ssid);
```

```
// Connect to WPA/WPA2 network. Change this line if  
using open or WEP network:  
  
status = WiFi.begin(ssid, pass);  
  
// wait 10 seconds for connection:  
  
delay(10000);  
  
}  
  
// print your board's IP address:  
  
Serial.print("IP Address: ");  
  
Serial.println(WiFi.localIP());  
  
DS18B20.begin(); // initialize the sensor  
  
}  
  
void loop()  
  
{
```

```
DS18B20.requestTemperatures();  
  
// send the command to get temperatures  
  
float temperature = DS18B20.getTempCByIndex(0); //  
read temperature in Celsius  
  
Serial.print("Temperature: ");  
  
Serial.print(temperature); // print the temperature in  
Celsius  
  
Serial.println("°C");  
  
  
  
if (temperature >= TEMP_THRESHOLD) {  
  
    if (isSent == false) { // to make sure that Arduino does  
not send duplicated emails  
  
        sendEmail(temperature);  
  
        isSent = true;  
  
    }  
  
} else {
```

```
    isSent = false; // reset to send if the temperature  
    exceeds threshold again  
}  
}  
  
void sendEmail(float temperature) {  
    // connect to IFTTT server on port 80:  
    if (client.connect(HOST_NAME, HTTP_PORT)) {  
        // if connected:  
        Serial.println("Connected to server");  
        // make a HTTP request:  
        String queryString = "?value1=" + String(temperature);  
        // send HTTP header  
        client.println("GET " + PATH_NAME + queryString + "  
HTTP/1.1");  
        client.println("Host: " + String(HOST_NAME));  
        client.println("Connection: close");
```

```
client.println(); // end HTTP header

while (client.connected()) {

    if (client.available()) {

        // read an incoming byte from the server and print it
        to serial monitor:

        char c = client.read();

        Serial.print(c);

    }

}

// the server's disconnected, stop the client:

client.stop();

Serial.println();

Serial.println("disconnected");

} else { // if not connected:

    Serial.println("connection failed");
```

```
}
```

<https://arduinogetstarted.com/tutorials/arduino-temperature-send-email-notification>

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

GAS SENSOR

MQ2

MQ135



 ArduinoGetStarted.com



MQ135 Air Quality Sensor

A device that is used to detect or measure or monitor the gases like ammonia, benzene, sulfur, carbon dioxide, smoke, and other harmful gases are called as an air quality gas sensor. The MQ135 air quality sensor, which belongs to the series of MQ gas sensors, is widely used to detect harmful gases, and smoke in the fresh air.

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK
The alternatives for the MQ135 air quality sensor/detector are

- MQ-2 (methane, LPG, butane, and smoke),
- MQ-3 (alcohol, smoke, and ethanol),
- MQ-4 (CNG gas and methane),
- MQ-5 (natural gas, and LPG),
- MQ-6 (butane and LPG),
- MQ-7 (CO), MQ-8 (Hydrogen),

MQ-9 (CO, and flammable gases),

MQ131 (ozone),

MQ136 (Hydrogen sulfide gas),

MQ137 (ammonia),

MQ138 (benzene, alcohol, propane, toluene, formaldehyde gas, and hydrogen),

MQ214 (methane, and natural gas),

MQ303A (alcohol, smoke, Ethanol),

MQ306A (LPG and butane),

MQ307A(CO),

MQ309A(CO and flammable gas).

It operates at a 5V supply with 150mA consumption.

Preheating of 20 seconds is required before the operation, to obtain the accurate output.

MQ135 Air Quality Sensor

It is a semiconductor air quality check sensor suitable for monitoring applications of air quality. It is highly sensitive to NH₃, NO_x, CO₂, benzene, smoke, and other dangerous gases in the atmosphere.

It is available at a low cost for harmful gas detection and monitoring applications.

If the concentration of gases exceeds the threshold limit in the air, then the digital output pin goes high.

The threshold value can be varied by using the

potentiometer of the sensor. The analog output voltage is obtained from the analog pin of the sensor, which gives the approximate value of the gas level present in the air.

Pin Configuration:

The MQ135 air quality sensor is a **4-pin** sensor module that features both analog and digital output from the corresponding pins.

Pin 1: **VCC:** This pin refers to a positive power supply of 5V that power up the MQ135 sensor module.

Pin 2: **GND (Ground):** This is a reference potential pin, which connects the MQ135 sensor module to the ground.

Pin 3: **Digital Out (Do):** This pin refers to the digital output pin that gives the digital output by adjusting the threshold value with the help of a potentiometer. This pin is used to detect and measure any one particular gas and makes the MQ135 sensor work without a microcontroller.

Pin 4: Analog Out (Ao): This pin generates the analog output signal of 0V to 5V and it depends on the gas intensity. This analog output signal is proportional to the gas vapor concentration, which is measured by the MQ135 sensor module. This pin is used to measure the gases in PPM. It is driven by TTL logic, operates with 5V, and is mostly interfaced with microcontrollers.

H-pins: There are 2 H-pins, where one is connected to the voltage supply and the other is connected to the ground.

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

A-pins: Here A-pins and B-pins can be interchanged. These are connected to the voltage supply.

B-pins: Here A-pins and B-pins can be interchanged. One pin is used to generate output while the other pin is connected to the ground.

The operating voltage: +5V.

Measures and detects NH₃, alcohol, NOx, Benzene, CO₂, smoke etc.

Range of analog output voltage: 0V-5V.

Range of digital output voltage: 0V-5V (TTL logic).

Duration of preheating: 20 seconds.

Used as an analog or digital sensor.

The potentiometer is used to vary the sensitivity of the digital pin.

~~ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK~~
Heating Voltage: 5V±0.1.

Load resistance is adjustable.

Heater resistance: 33ohms±5%.

Heating consumption:<800mW.

Operating temperature: -10°C to -45°C.

Storage temperature: -20°C to -70°C.

Related humidity: <95%Rh.

Oxygen concentration: 21% (affects the sensitivity).

Sensing resistance: 30kiloohms to 200kiloohms.

Concentration slope rate: ≤ 0.65 .

Preheat time: over 24 hrs.

Simple drive circuit.

How to Detect and Measure Gases using the MQ135 Air Quality Sensor:

To measure or detect the gases, use analog pins or digital pins.

Just apply 5V to the module and you can observe that the module's power LED turns ON (glows) and the output LED turns OFF when no gas is detected by the module. This means that the output of the digital pin is 0V.

Note that the sensor must be kept for preheating time for 20seconds (as mentioned in the specifications) before the actual operation.

Now, once when the MQ135 sensor is operated to detect, then the LED output goes high along with the digital output pin. Otherwise, use the potentiometer until the output increases. Whenever the sensor detects

a certain gas concentration, the digital pin goes high (5V), otherwise it stays low (0V).

We can also use analog pins to get the same result. The output analog values (0-5V) are read from the microcontroller. This value is directly proportional to the gas concentration detected by the sensor. By the experimental values, we can observe the working and reaction of the MQ135 sensor with different gas concentrations and the programming developed accordingly.

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

How to Measure PPM (parts per million) using the MQ135 Air Quality Sensor:

The MQ-135 gas sensor uses SnO_2 , a gas-sensitive material that has higher resistance in clean air. An increase in the number of harmful gases decreases the resistance of the gas MQ135 sensor.

To measure PPM with the MQ-135 air quality sensor, observe the graph between (R_s/R_o) and PPM shown below.

We can calibrate the MQ135 sensor by determining the Rs value from the below formula,

$$\text{Resistance of sensor } R_s = (V_c/V_{RL} - 1)R_L$$

After calculating the Ro and Rs values, the ratio is found, and using the above graph we can calculate the PPM value of the particular gas, which is to be measured.

How to Interface the MQ135 Air Quality Sensor with Arduino:

Connecting wires.

When the Ao (analog output) of the MQ135 is higher than the 400, then the LED turns ON, which is connected to pin 2 of the Arduino board. Else the LED turns OFF. Observe the readings of both digital and analog outputs of the sensor on the LCD or monitor.

```
int sensorValue;
```

(int variable to read analogue output reading)

```
int digitalValue;
```

(int variable to read digital output reading)

```
void setup()
```

```
{
```

```
Serial.begin(9600);
```

// sets the serial port to 9600 (sets the serial communication to 9600 baud rate)

```
pinMode(13, OUTPUT);
```

(pin 13 is connected to the anode terminal of the LED as an output)

```
pinMode(2, INPUT);
```

(pin 2 of Arduino is connected to the Do pin of the MQ135 as an input)

```
}
```

```
void loop()
```

```
{
```

```
sensorValue = analogRead(0); // read analogue input pin  
0 (to read the analogue input on Ao)
```

```
digitalValue = digitalRead(2); (to read and save the digital  
output on pin 2 of Arduino)
```

```
if (sensorValue > 400)
```

```
{
```

```
digitalWrite(13, HIGH); (if the analogue reading is greater  
than 400, then the LED turns ON)
```

```
}
```

```
else
```

```
digitalWrite(13, LOW); (if the analogue reading is less  
than 400, the LED turns OFF)
```

```
Serial.println(sensorValue, DEC); // prints the value read
```

```
Serial.println(digitalValue, DEC);
```

```
delay(1000);  
  
// wait 100ms for the next reading (analogue and digital  
output readings are displayed on the monitor)  
  
}
```

Where to Use/Applications of MQ135 Air Quality Sensor:

The applications of the MQ135 quality sensor are,

Used in the detection of excess or leakage of gases like nitrogen oxide, ammonia, alcohol, aromatic compounds, smoke, and sulfide.

Used as air quality monitors.

Used in air quality equipment for offices and buildings.

Used as a domestic air pollution detector.

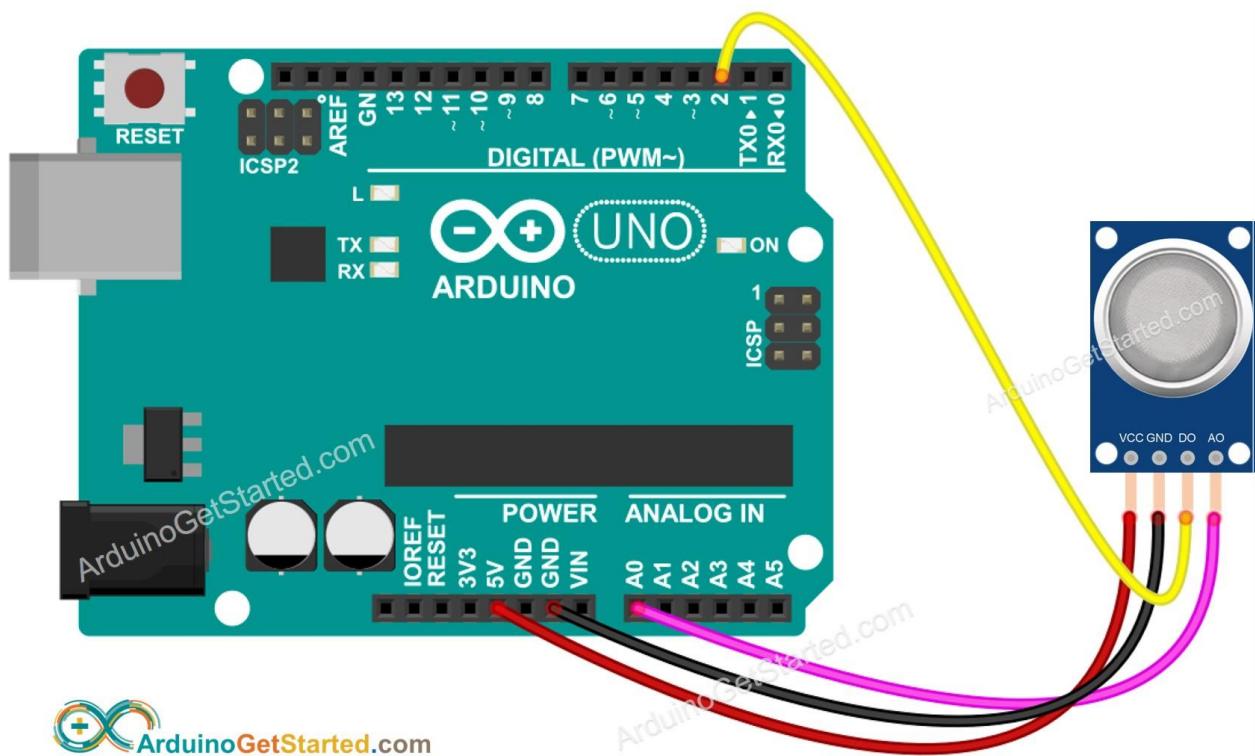
Used as an industrial air pollution detector.

Works as a portable air pollution detector.

On serial monitor you can see values of analog pin being detected. Currently in my case they are

around about 150 which indicate normal air.

- Normal air returns approximately 100-150
- Alcohol returns approximately 700
- Lighter gas returns approximately 750



```
#define AO_PIN A0 // Arduino's pin connected to AO pin  
of the MQ2 sensor  
  
void setup()
```

```
{  
    // initialize serial communication  
    Serial.begin(9600);  
  
    Serial.println("Warming up the MQ2 sensor");  
  
    delay(20000);  
  
    // wait for the MQ2 to warm up  
  
}  
  
  
  
  
  
  
void loop()  
{  
    int gasValue = analogRead(AO_PIN);  
  
    Serial.print("MQ2 sensor AO value: ");  
  
    Serial.println(gasValue);  
}
```

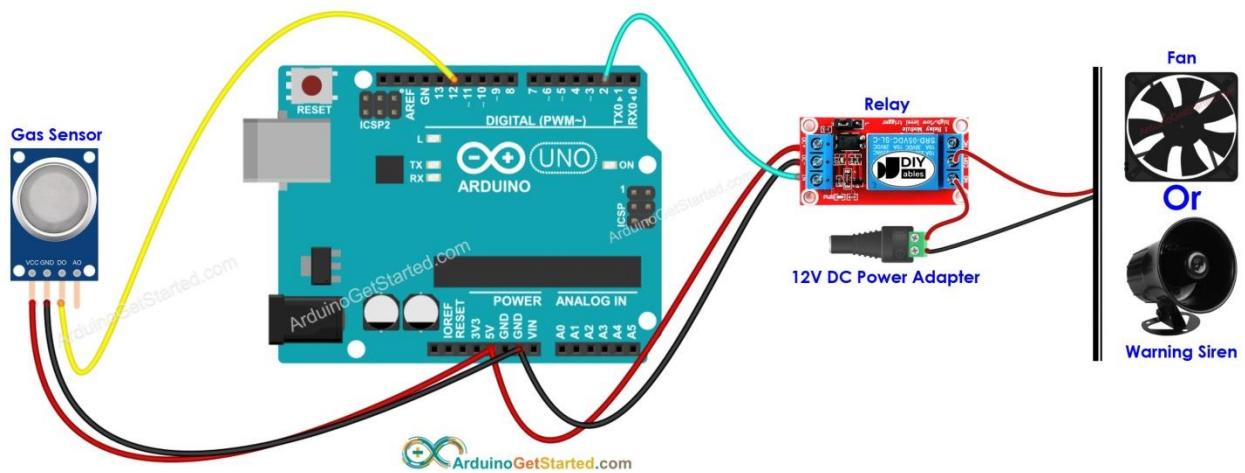
```
#define DO_PIN 2 // Arduino's pin connected to DO pin  
of the MQ2 sensor  
  
void setup() {  
  
    // initialize serial communication  
  
    Serial.begin(9600);  
  
    // initialize the Arduino's pin as an input  
  
    pinMode(DO_PIN, INPUT);  
  
    Serial.println("Warming up the MQ2 sensor");  
  
    delay(20000); // wait for the MQ2 to warm up  
  
}  
  
void loop()  
{
```

```
int gasState = digitalRead(DO_PIN);

if (gasState == HIGH)
    Serial.println("The gas is NOT present");
else
    Serial.println("The gas is present");
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

ARDUINO SENSOR GAS RELAY



```
#define DO_PIN 12
```

```
// Arduino's pin connected to DO pin of the MQ2 sensor
```

```
#define RELAY_PIN 2
```

```
// Arduino's pin connected to relay
```

```
void setup()
```

```
{
```

```
// initialize serial communication
```

```
Serial.begin(9600);
```

```
// initialize the Arduino's pin as an input
```

```
pinMode(DO_PIN, INPUT);
```

```
pinMode(RELAY_PIN, OUTPUT);
```

```
Serial.println("Warming up the MQ2 sensor");

delay(20000); // wait for the MQ2 to warm up

}

void loop()
{
    int gasState = digitalRead(DO_PIN);

    if (gasState == HIGH) {

        Serial.println("The gas is NOT present");

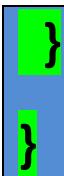
        digitalWrite(RELAY_PIN, LOW);

        // turn off
    }

    else

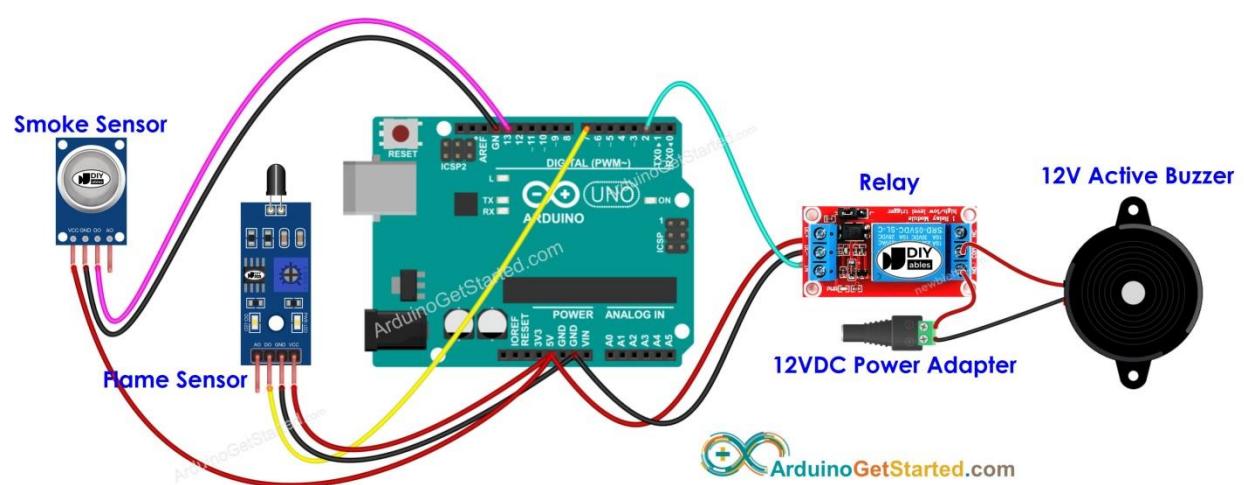
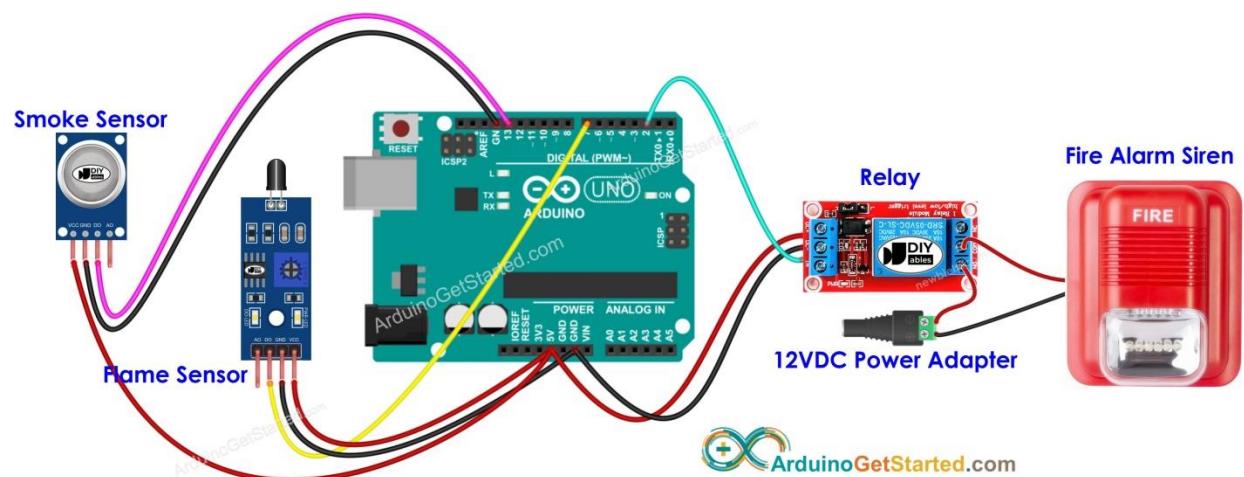
    {
        Serial.println("The gas is present");

        digitalWrite(RELAY_PIN, HIGH); // turn on
    }
}
```



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

GAS SENSOR FIRE ALARM SYSTEM



```
#define FLAME_PIN 7  
// Arduino's pin connected to DO pin of the flame  
sensor  
  
#define SMOKE_PIN 13  
// Arduino's pin connected to DO pin of the smoke MQ2  
sensor  
  
#define RELAY_PIN 2  
// Arduino's pin connected to relay  
  
  
void setup()  
{  
    // initialize serial communication  
    Serial.begin(9600);  
    // initialize the Arduino's pin  
    pinMode(FLAME_PIN, INPUT);  
    pinMode(SMOKE_PIN, INPUT);
```

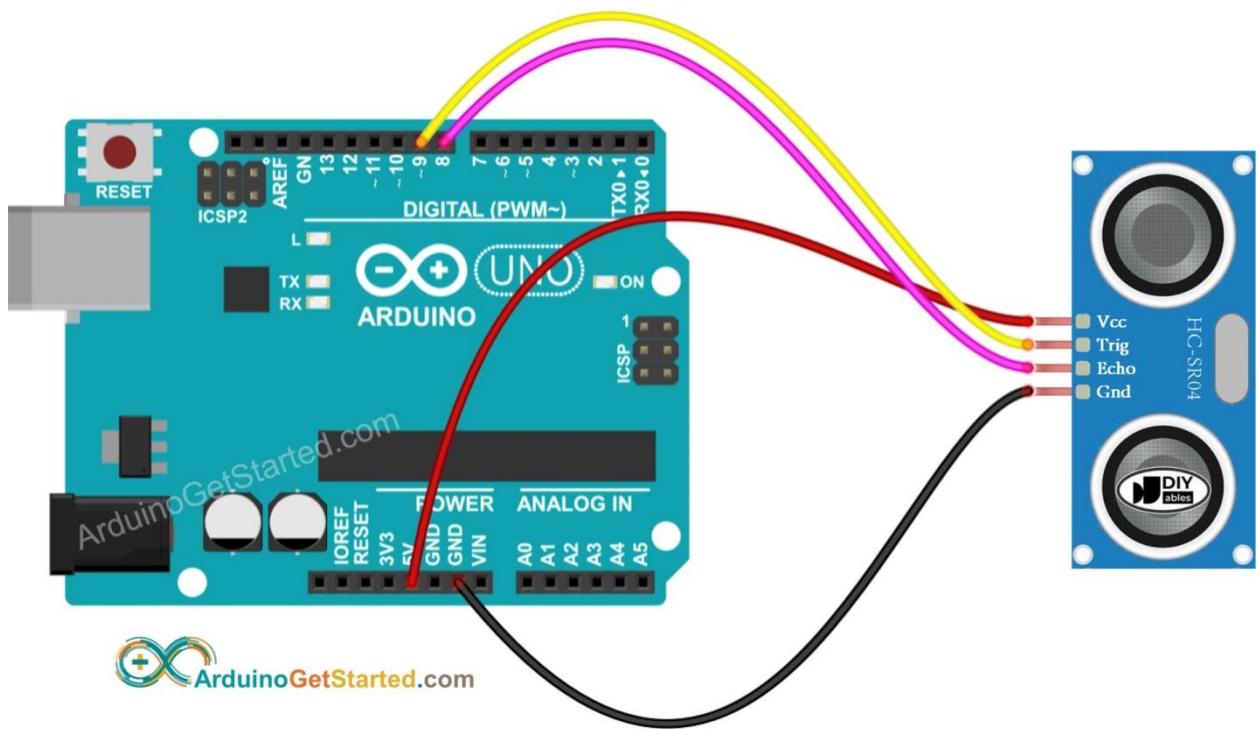
```
pinMode(RELAY_PIN, OUTPUT);  
}  
  
void loop()  
{  
    int flame_state = digitalRead(FLAME_PIN);  
    int smoke_state = digitalRead(SMOKE_PIN);  
  
    if (flame_state == LOW) {  
        Serial.println("Fire is detected based on the flame  
sensor => alarming");  
        digitalWrite(RELAY_PIN, HIGH);  
    } else if (smoke_state == LOW) {  
        Serial.println("Fire is detected based on the smoke  
sensor => alarming");  
        digitalWrite(RELAY_PIN, HIGH);  
    }  
}
```

```
Else  
{  
    Serial.println("No fire detected => great!");  
    digitalWrite(RELAY_PIN, LOW);  
}  
}
```

ARDUINO ULTRASONIC SENSOR

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK





ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

How to Get Distance From Ultrasonic Sensor

To get distance from the ultrasonic sensor, we only need to do two steps (1 and 6 on **How It Works** part)

- Generates a 10-microsecond pulse on TRIG pin
- Measures the pulse duration in ECHO pin, and then calculate the distance between sensor and obstacle

Distance Calculation

We have:

- The travel time of the ultrasonic wave (μs): `travel_time = pulse_duration`
- The speed of the ultrasonic wave: `speed = SPEED_OF_SOUND = 340 m/s = 0.034 cm/ μs`

So:

- The travel distance of the ultrasonic wave (cm): `travel_distance = speed * travel_time = 0.034 * pulse_duration`
- The distance between sensor and obstacle (cm): `distance = travel_distance / 2 = 0.034 * pulse_duration / 2 = 0.017 * pulse_duration`

* Arduino - Ultrasonic Sensor HC-SR04

* - VCC -> 5VDC

* - TRIG -> Pin 9

* - ECHO -> Pin 8

* - GND -> GND

*

```
int trigPin = 9; // TRIG pin
```

```
int echoPin = 8; // ECHO pin
```

```
float duration_us, distance_cm;
```

```
void setup()
```

```
{
```

```
// begin serial port
```

```
Serial.begin (9600);
```



```
distance_cm = 0.017 * duration_us;
```

```
// print the value to Serial Monitor
```

```
Serial.print("distance: ");
```

```
Serial.print(distance_cm);
```

```
Serial.println(" cm");
```

```
delay(500);
```

```
}
```

NOISE REDUCTION PROGRAM ULTRASONIC SENSOR

```
#define TRIG_PIN 9 // TRIG pin  
  
#define ECHO_PIN 8 // ECHO pin  
  
float filterArray[20];  
  
// array to store data samples from sensor  
  
float distance;  
  
// store the distance from sensor  
  
  
  
void setup()  
{  
    // begin serial port  
    Serial.begin(9600);  
  
    // configure the trigger and echo pins to output mode  
    pinMode(TRIG_PIN, OUTPUT);  
  
    pinMode(ECHO_PIN, INPUT);
```

```
}
```

```
void loop()
```

```
{
```

```
// 1. TAKING MULTIPLE MEASUREMENTS AND STORE  
IN AN ARRAY
```

```
for (int sample = 0; sample < 20; sample++) {
```

```
    filterArray[sample] = ultrasonicMeasure();
```

```
    delay(30); // to avoid ultrasonic interfering
```

```
}
```

```
// 2. SORTING THE ARRAY IN ASCENDING ORDER
```

```
for (int i = 0; i < 19; i++) {
```

```
    for (int j = i + 1; j < 20; j++) {
```

```
        if (filterArray[i] > filterArray[j]) {
```

```
            float swap = filterArray[i];
```

```
            filterArray[i] = filterArray[j];
```

```
            filterArray[j] = swap;
```

```
}
```

```
}
```

```
}
```

// 3. FILTERING NOISE

```
// + the five smallest samples are considered as noise -
```

```
> ignore it
```

```
// + the five biggest samples are considered as noise -
```

```
> ignore it
```

```
// -----
```

```
// => get average of the 10 middle samples (from 5th  
to 14th)
```

```
double sum = 0;
```

```
for (int sample = 5; sample < 15; sample++) {
```

```
    sum += filterArray[sample];
```

```
}
```

```
distance = sum / 10;
```

```
// print the value to Serial Monitor
Serial.print("distance: ");
Serial.print(distance);
Serial.println(" cm");
}

float ultrasonicMeasure() {
// generate 10-microsecond pulse to TRIG pin
digitalWrite(TRIG_PIN, HIGH);
delayMicroseconds(10);
digitalWrite(TRIG_PIN, LOW);

// measure duration of pulse from ECHO pin
float duration_us = pulseIn(ECHO_PIN, HIGH);

// calculate the distance
```

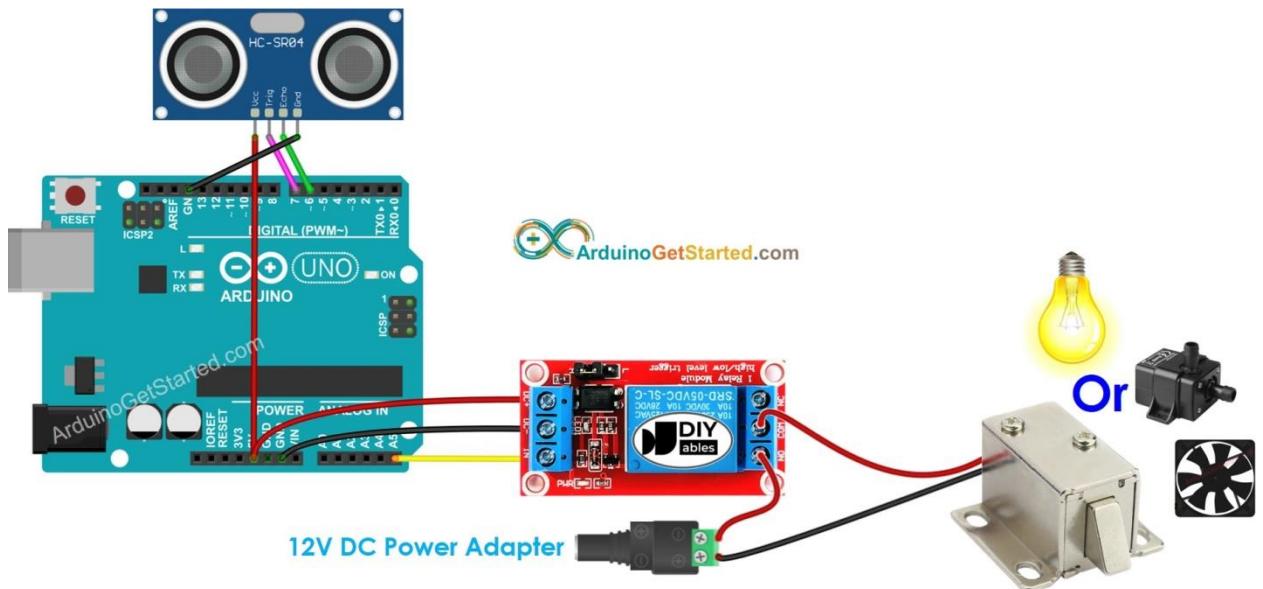
```
float distance_cm = 0.017 * duration_us;
```

```
return distance_cm;
```

```
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

ULTRASONIC RELAY



```
// constants won't change
```

```
const int TRIG_PIN = 7;
```

```
// Arduino pin connected to Ultrasonic Sensor's TRIG  
pin
```

```
const int ECHO_PIN = 6;
```

```
// Arduino pin connected to Ultrasonic Sensor's ECHO  
pin
```

```
const int RELAY_PIN = A5;
```

```
// Arduino pin connected to Relay's pin
```

```
const int DISTANCE_THRESHOLD = 50;  
  
// centimeters  
  
// variables will change:  
  
float duration_us, distance_cm;  
  
  
void setup()  
{  
    Serial.begin (9600);  
    // initialize serial port  
    pinMode(TRIG_PIN, OUTPUT);  
    // set arduino pin to output mode  
    pinMode(ECHO_PIN, INPUT);  
    // set arduino pin to input mode  
    pinMode(RELAY_PIN, OUTPUT);  
    // set arduino pin to output mode  
}  
}
```

```
void loop()
{
    // generate 10-microsecond pulse to TRIG pin
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);

    // measure duration of pulse from ECHO pin
    duration_us = pulseIn(ECHO_PIN, HIGH);

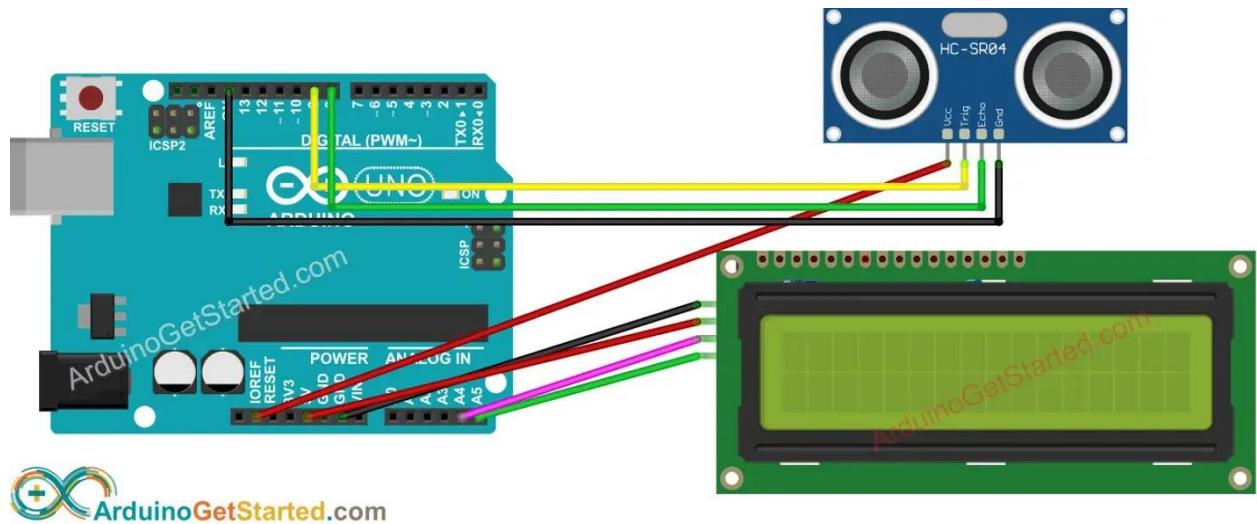
    // calculate the distance
    distance_cm = 0.017 * duration_us;

    if(distance_cm < DISTANCE_THRESHOLD)
        digitalWrite(RELAY_PIN, HIGH); // turn on Relay
    else
        digitalWrite(RELAY_PIN, LOW); // turn off Relay
}
```

```
// print the value to Serial Monitor  
Serial.print("distance: ");  
Serial.print(distance_cm);  
Serial.println(" cm");  
  
delay(500);  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

ULTRASONIC LCD



```
#include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27  
(from DIYables LCD), 16 column and 2 rows
```

```
int trigPin = 9; // TRIG pin
```

```
int echoPin = 8; // ECHO pin
```

```
float duration_us, distance_cm;
```

```
void setup() {
```

```
lcd.init();          // initialize the lcd

lcd.backlight();    // open the backlight

pinMode(trigPin, OUTPUT); // config trigger pin to
output mode

pinMode(echoPin, INPUT); // config echo pin to input
mode

}

void loop() {

// generate 10-microsecond pulse to TRIG pin

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

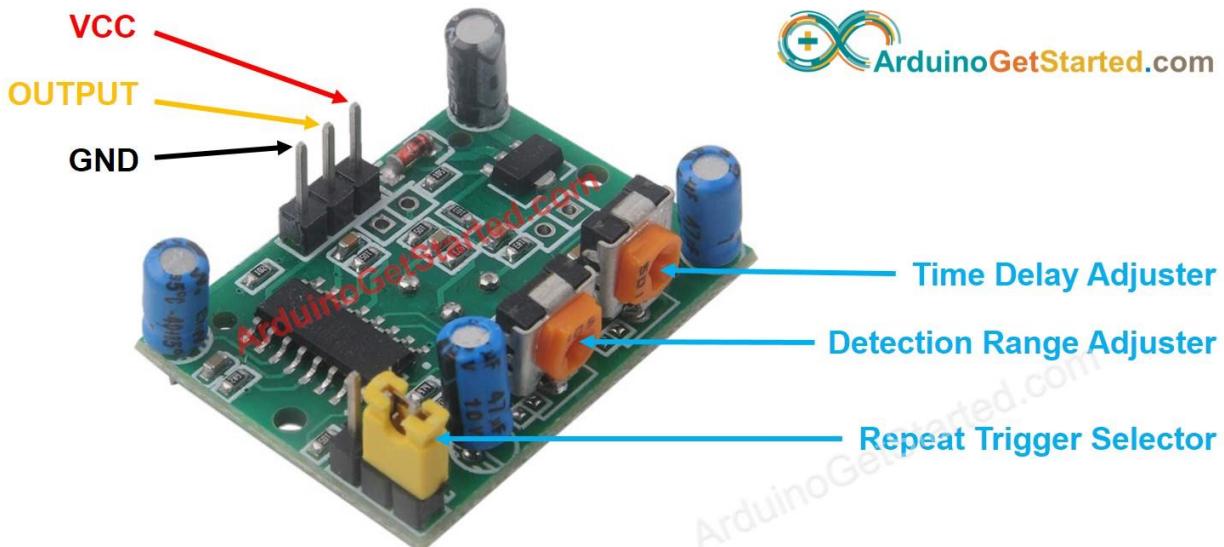
// measure duration of pulse from ECHO pin

duration_us = pulseIn(echoPin, HIGH);
```

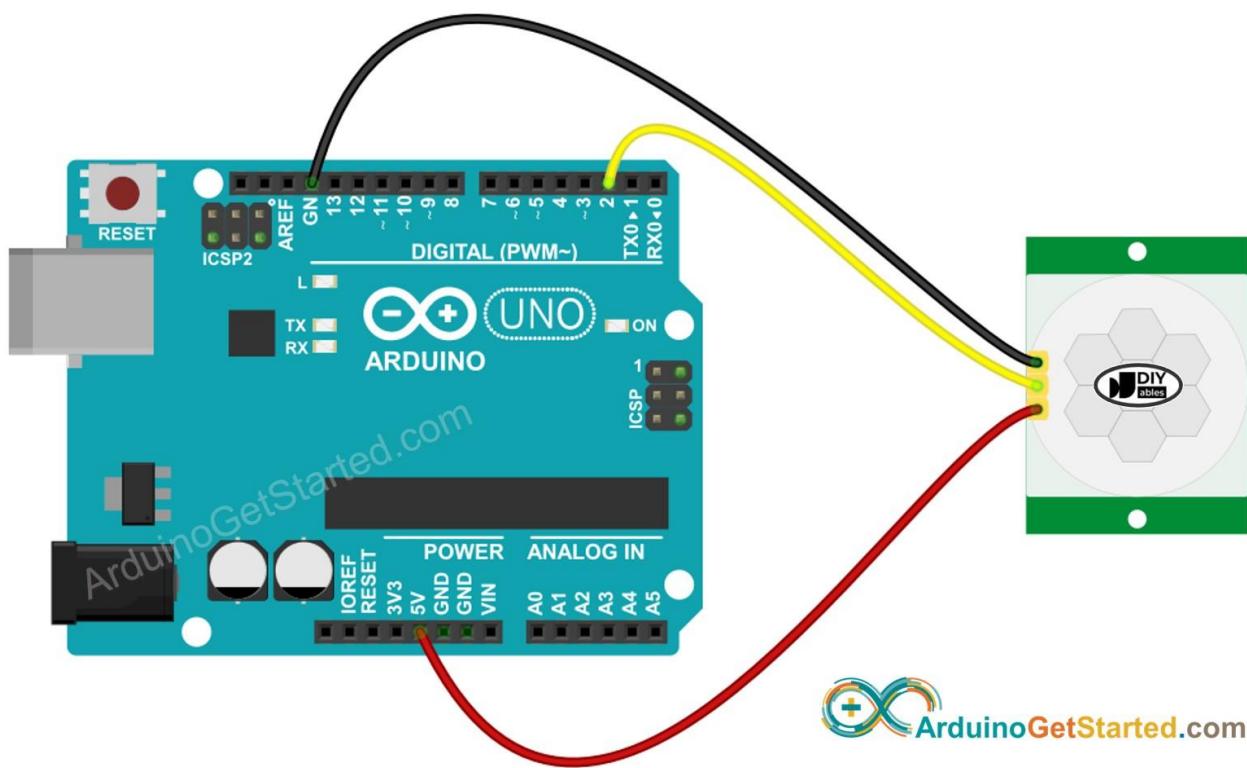
```
// calculate the distance  
  
distance_cm = 0.017 * duration_us;  
  
  
lcd.clear();  
  
lcd.setCursor(0, 0); // start to print at the first row  
  
lcd.print("Distance: ");  
  
lcd.print(distance_cm);  
  
  
delay(500);  
  
}
```

MOTION SENSOR

HC SR501



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK



```
const int PIN_TO_SENSOR = 2; // the pin that OUTPUT  
pin of sensor is connected to  
  
int pinStateCurrent = LOW; // current state of pin  
  
int pinStatePrevious = LOW; // previous state of pin  
  
  
void setup() {  
  
    Serial.begin(9600); // initialize serial  
  
    pinMode(PIN_TO_SENSOR, INPUT); // set arduino pin to  
input mode to read value from OUTPUT pin of sensor  
  
}  
  
  
  
void loop() {  
  
    pinStatePrevious = pinStateCurrent; // store old state  
  
    pinStateCurrent = digitalRead(PIN_TO_SENSOR); //  
read new state  
  
  
    if (pinStatePrevious == LOW && pinStateCurrent ==  
HIGH) { // pin state change: LOW -> HIGH
```

```
Serial.println("Motion detected!");

// TODO: turn on alarm, light or activate a device ...
here

}

else

if (pinStatePrevious == HIGH && pinStateCurrent ==
LOW) { // pin state change: HIGH -> LOW

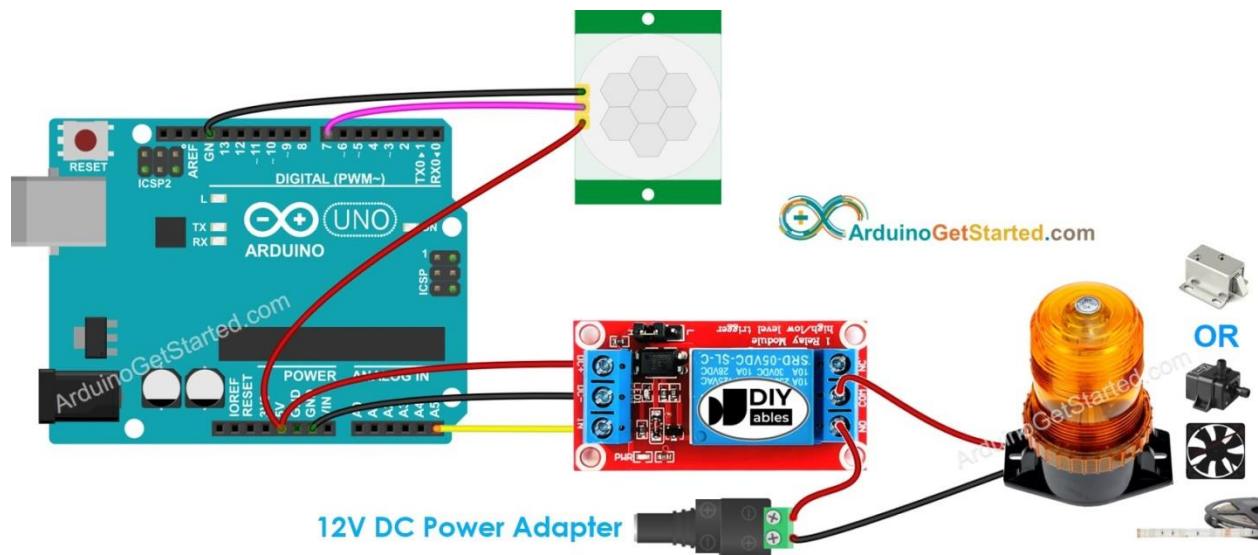
    Serial.println("Motion stopped!");

    // TODO: turn off alarm, light or deactivate a device ...
here

}

}
```

MOTION SENSOR RELAY



```
const int MOTION_SENSOR_PIN = 7; // Arduino pin
```

```
connected to the OUTPUT pin of motion sensor
```

```
const int RELAY_PIN      = A5; // Arduino pin connected  
to the IN pin of relay
```

```
int motionStateCurrent    = LOW; // current state of  
motion sensor's pin
```

```
int motionStatePrevious   = LOW; // previous state of  
motion sensor's pin
```

```
void setup() {
```

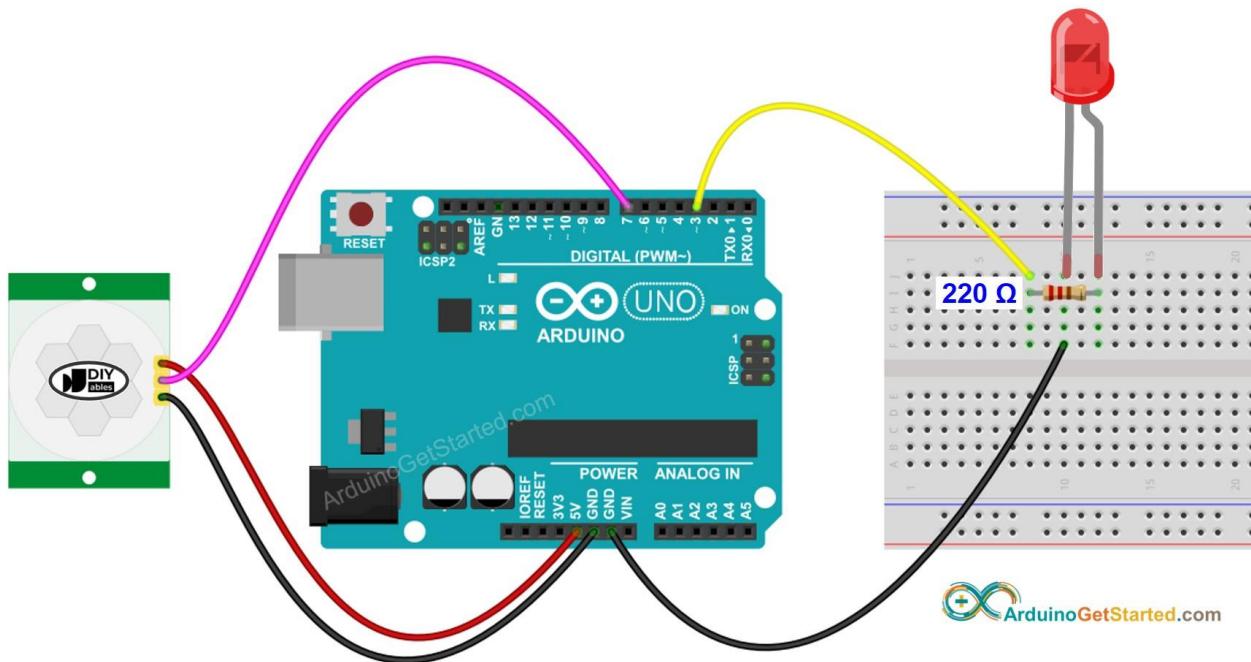
```
  Serial.begin(9600); // initialize serial
```

```
pinMode(MOTION_SENSOR_PIN, INPUT); // set arduino  
pin to input mode  
  
pinMode(RELAY_PIN, OUTPUT);      // set arduino pin to  
output mode  
  
}  
  
void loop() {  
  
    motionStatePrevious = motionStateCurrent;      //  
store old state  
  
    motionStateCurrent =  
digitalRead(MOTION_SENSOR_PIN); // read new state  
  
  
    if (motionStatePrevious == LOW &&  
motionStateCurrent == HIGH) { // pin state change: LOW  
-> HIGH  
        Serial.println("Motion detected!");  
        digitalWrite(RELAY_PIN, HIGH); // turn on  
    }  
}
```

```
else  
{  
    if (motionStatePrevious == HIGH &&  
        motionStateCurrent == LOW) { // pin state change: HIGH  
        -> LOW  
  
        Serial.println("Motion stopped!");  
  
        digitalWrite(RELAY_PIN, LOW); // turn off  
  
    }  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

MOTION SENSOR LED



```
const int MOTION_SENSOR_PIN = 7; // Arduino pin
```

```
connected to the OUTPUT pin of motion sensor
```

```
const int LED_PIN = 3; // Arduino pin connected  
to LED's pin
```

```
int motionStateCurrent = LOW; // current state of  
motion sensor's pin
```

```
int motionStatePrevious = LOW; // previous state of  
motion sensor's pin
```

```
void setup() {
```

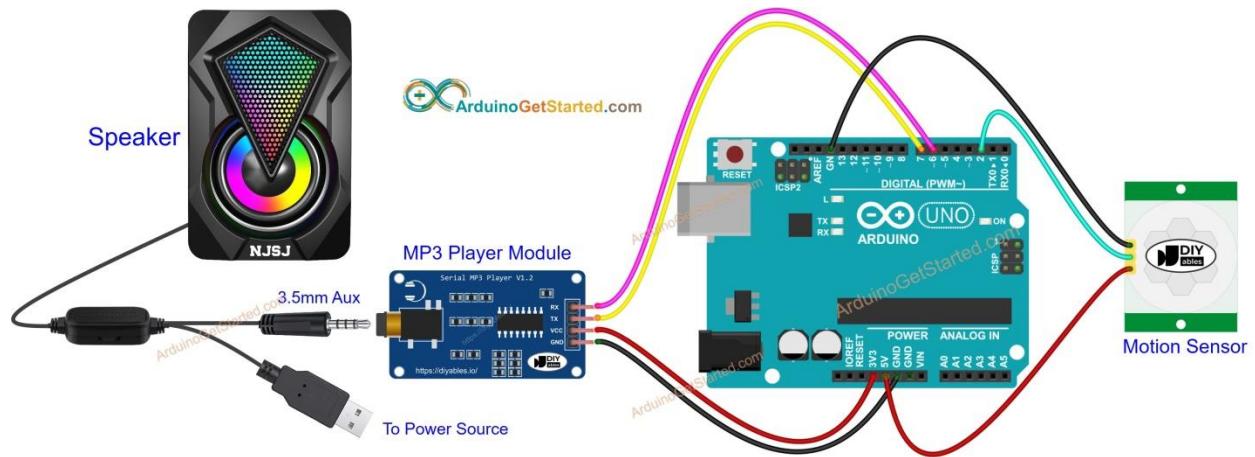
```
Serial.begin(9600); // initialize serial
```

```
pinMode(MOTION_SENSOR_PIN, INPUT); // set  
arduino pin to input mode  
  
pinMode(LED_PIN, OUTPUT);      // set arduino pin to  
output mode  
  
}  
  
void loop() {  
  
    motionStatePrevious = motionStateCurrent;      //  
store old state  
  
    motionStateCurrent =  
digitalRead(MOTION_SENSOR_PIN); // read new state  
  
  
    if (motionStatePrevious == LOW &&  
motionStateCurrent == HIGH) { // pin state change:  
LOW -> HIGH  
  
        Serial.println("Motion detected!");  
  
        digitalWrite(LED_PIN, HIGH); // turn on  
  
    }  
}
```

```
else  
if (motionStatePrevious == HIGH &&  
motionStateCurrent == LOW) { // pin state change:  
HIGH -> LOW  
  
Serial.println("Motion stopped!");  
  
digitalWrite(LED_PIN, LOW); // turn off  
  
}  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

MOTION SENSOR MP3 PLAYER



```
#include <SoftwareSerial.h>
#include <Servo.h>

#define CMD_PLAY_NEXT 0x01
#define CMD_PLAY_PREV 0x02
#define CMD_PLAY_W_INDEX 0x03
#define CMD_SET_VOLUME 0x06
#define CMD_SEL_DEV 0x09
#define CMD_PLAY_W_VOL 0x22
#define CMD_PLAY 0x0D
#define CMD_PAUSE 0x0E
#define CMD_SINGLE_CYCLE 0x19
```

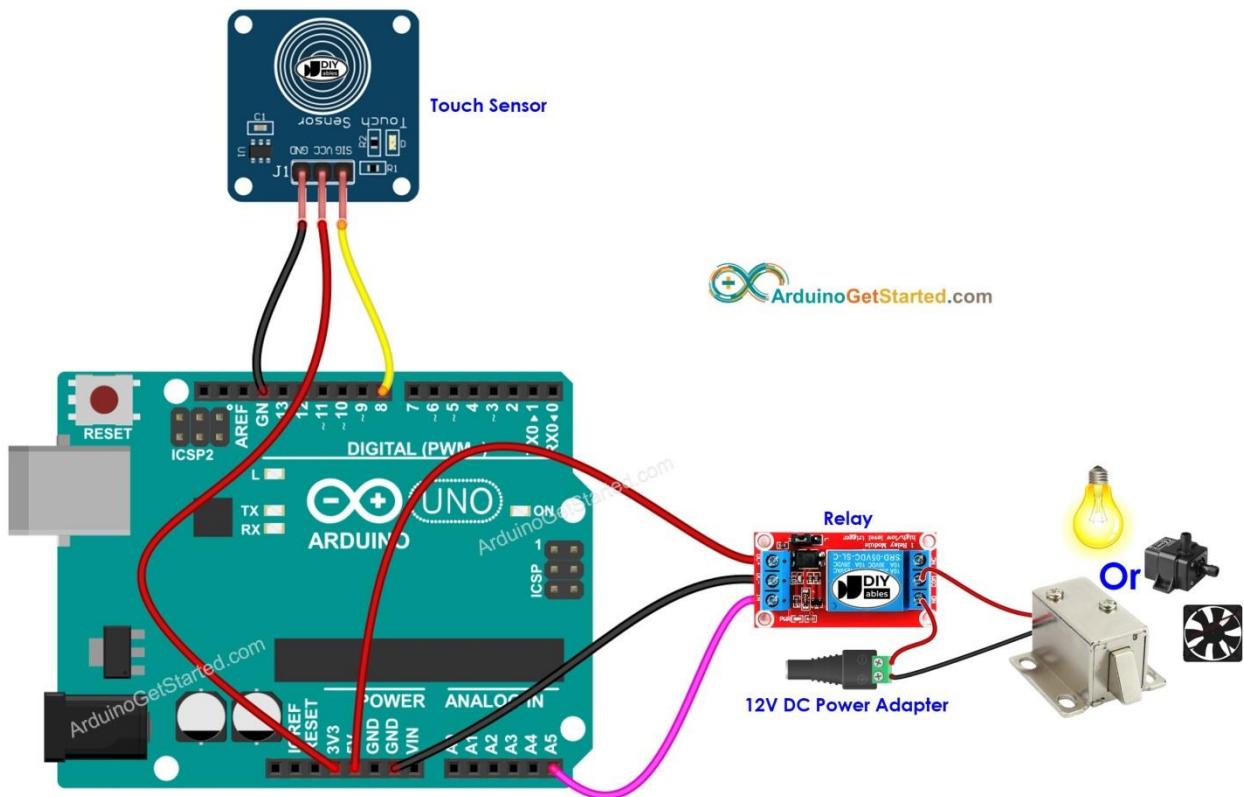
```
#define DEV_TF 0x02  
  
#define SINGLE_CYCLE_ON 0x00  
  
#define SINGLE_CYCLE_OFF 0x01  
  
#define ARDUINO_RX 7 // Arduino Pin connected to the  
TX of the Serial MP3 Player module  
  
#define ARDUINO_TX 6 // Arduino Pin connected to the  
RX of the Serial MP3 Player module  
  
#define MOTION_SENSOR_PIN 2 // Arduino pin  
connected to motion sensor's pin  
  
SoftwareSerial mp3(ARDUINO_RX, ARDUINO_TX);  
  
int prev_motion_state; // the previous state of motion  
sensor  
  
int motion_state; // the current state of motion  
sensor
```

```
void setup() {  
    Serial.begin(9600);  
    mp3.begin(9600);  
    delay(500); // wait chip initialization is complete  
  
    mp3_command(CMD_SEL_DEV, DEV_TF); // select the  
    TF card  
    delay(200); // wait for 200ms  
    pinMode(MOTION_SENSOR_PIN, INPUT); // set arduino  
    pin to input mode  
    motion_state = digitalRead(MOTION_SENSOR_PIN);  
}  
  
void loop() {  
    prev_motion_state = motion_state; // save the  
    last state
```

```
motion_state = digitalRead(MOTION_SENSOR_PIN); //  
read new state  
  
if (motion_state == LOW && prev_motion_state ==  
HIGH) { // pin state change: LOW -> HIGH  
    Serial.println("Motion detected!");  
    mp3_command(CMD_PLAY, 0x0000); // Play the first  
mp3 file  
}  
  
else  
  
if (motion_state == HIGH && prev_motion_state ==  
LOW) { // pin state change: HIGH -> LOW  
    Serial.println("Motion stopped!");  
}  
}  
  
void mp3_command(int8_t command, int16_t dat) {  
    int8_t frame[8] = { 0 };
```

```
frame[0] = 0x7e;          // starting byte  
frame[1] = 0xff;          // version  
frame[2] = 0x06;          // the number of bytes of the  
command without starting byte and ending byte  
frame[3] = command;       //  
frame[4] = 0x00;           // 0x00 = no feedback, 0x01 =  
feedback  
frame[5] = (int8_t)(dat >> 8); // data high byte  
frame[6] = (int8_t)(dat);    // data low byte  
frame[7] = 0xef;           // ending byte  
  
for (uint8_t i = 0; i < 8; i++) {  
    mp3.write(frame[i]);  
}  
}
```

TOUCH SENSOR REL



```
const int TOUCH_SENSOR_PIN = 8; // Arduino pin
```

connected to touch sensor's pin

```
const int RELAY_PIN = A5; // Arduino pin connected
```

to the IN pin of relay

```
void setup() {
```

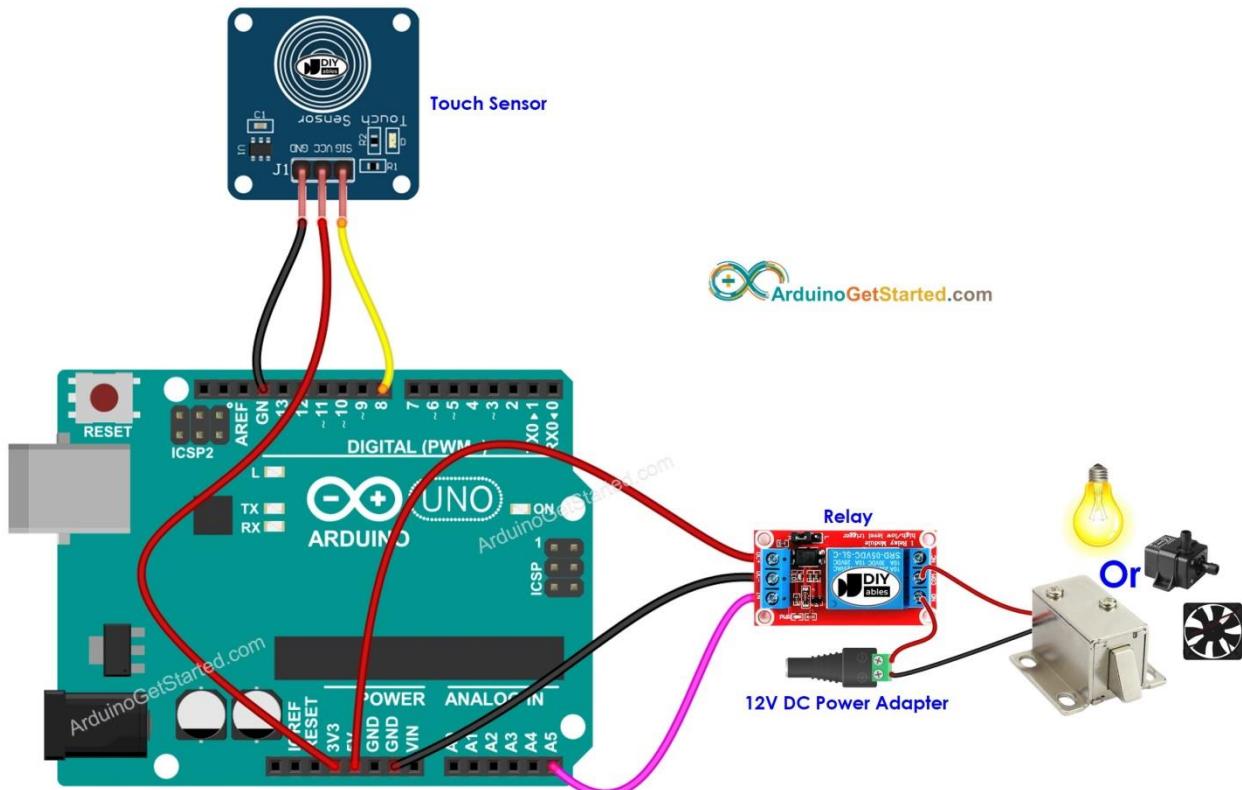
```
    Serial.begin(9600); // initialize serial
```

```
    pinMode(TOUCH_SENSOR_PIN, INPUT); // set arduino  
pin to input mode
```

```
pinMode(RELAY_PIN, OUTPUT); // set arduino pin  
to output mode  
  
}  
  
void loop() {  
  
    int touchState = digitalRead(TOUCH_SENSOR_PIN); //  
read new state  
  
    if (touchState == HIGH) {  
  
        Serial.println("The sensor is being touched");  
  
        digitalWrite(RELAY_PIN, HIGH); // turn on  
  
    }  
  
    else  
  
        if (touchState == LOW) {  
  
            Serial.println("The sensor is untouched");  
  
            digitalWrite(RELAY_PIN, LOW); // turn off  
  
        }  
}
```

}

TOUCH SENSOR TOGGLE RELAY



```
const int TOUCH_SENSOR_PIN = 8; // Arduino pin  
connected to touch sensor's pin
```

```
const int RELAY_PIN      = A5; // Arduino pin connected  
to relay's pin  
  
// variables will change:  
  
int relayState = LOW; // the current state of relay  
  
int lastTouchState; // the previous state of touch  
sensor  
  
int currentTouchState; // the current state of touch  
sensor  
  
void setup() {  
  
    Serial.begin(9600); // initialize serial  
  
    pinMode(TOUCH_SENSOR_PIN, INPUT); // set arduino  
pin to input mode  
  
    pinMode(RELAY_PIN, OUTPUT); // set arduino pin  
to output mode  
  
    currentTouchState =  
    digitalRead(TOUCH_SENSOR_PIN);
```

```
}

void loop() {

    lastTouchState = currentTouchState;      // save
    the last state

    currentTouchState =
    digitalRead(TOUCH_SENSOR_PIN); // read new state

    if(lastTouchState == LOW && currentTouchState ==
    HIGH) {

        Serial.println("The sensor is touched");

        // toggle state of relay

        relayState = !relayState;

        // control relay arccoding to the toggled state

        digitalWrite(RELAY_PIN, relayState);

    }

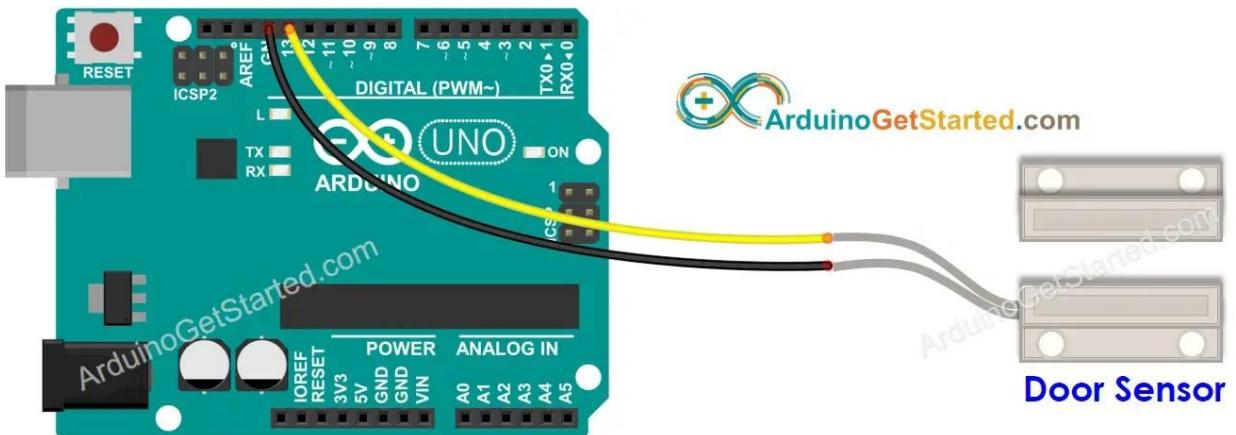
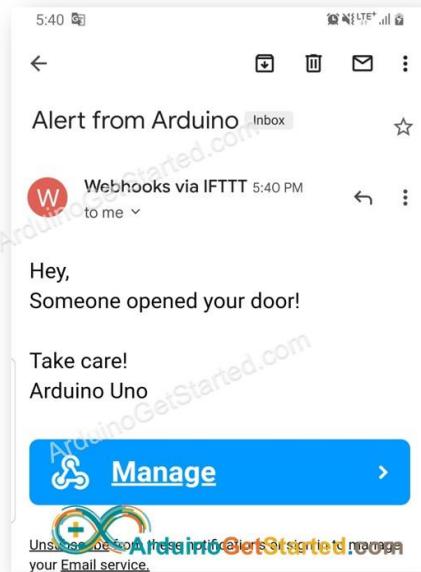
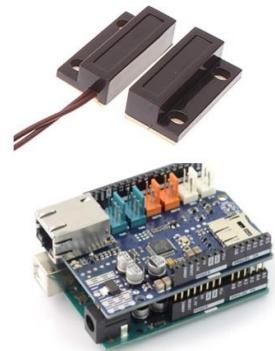
}
```

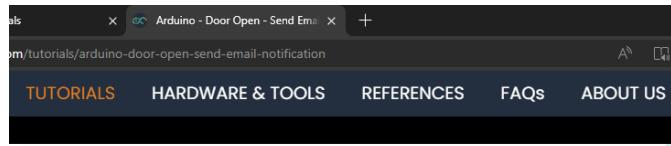
ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

DOOR OPEN E MAIL NOTIFICATION



If door is opened
Send Email Alert





What we need to do

We need to do two main tasks:

- ◆ Create an [Applet](#) on the IFTTT website
- ◆ Write Arduino code that makes an HTTP request to the [Applet](#) when the door is opened.

How it works

When the door is opened:

- ◆ Arduino makes an HTTP request to IFTTT [Applet](#) we created,
- ◆ The IFTTT [Applet](#) will send an email to the email address.

How it works

When the door is opened:

- ◆ Arduino makes an HTTP request to IFTTT [Applet](#) we created,
- ◆ The IFTTT [Applet](#) will send an email to the email address.

The email address, email subject, and email content are specified when we create the IFTTT [Applet](#).

ARDUINO TUTORIALS | LECTURER GPK

How To Create an IFTTT Applet

- ◆ Create an [IFTTT account](#) and Login to IFTTT.
- ◆ Go to [IFTTT Home page](#) and click the [Create](#) button

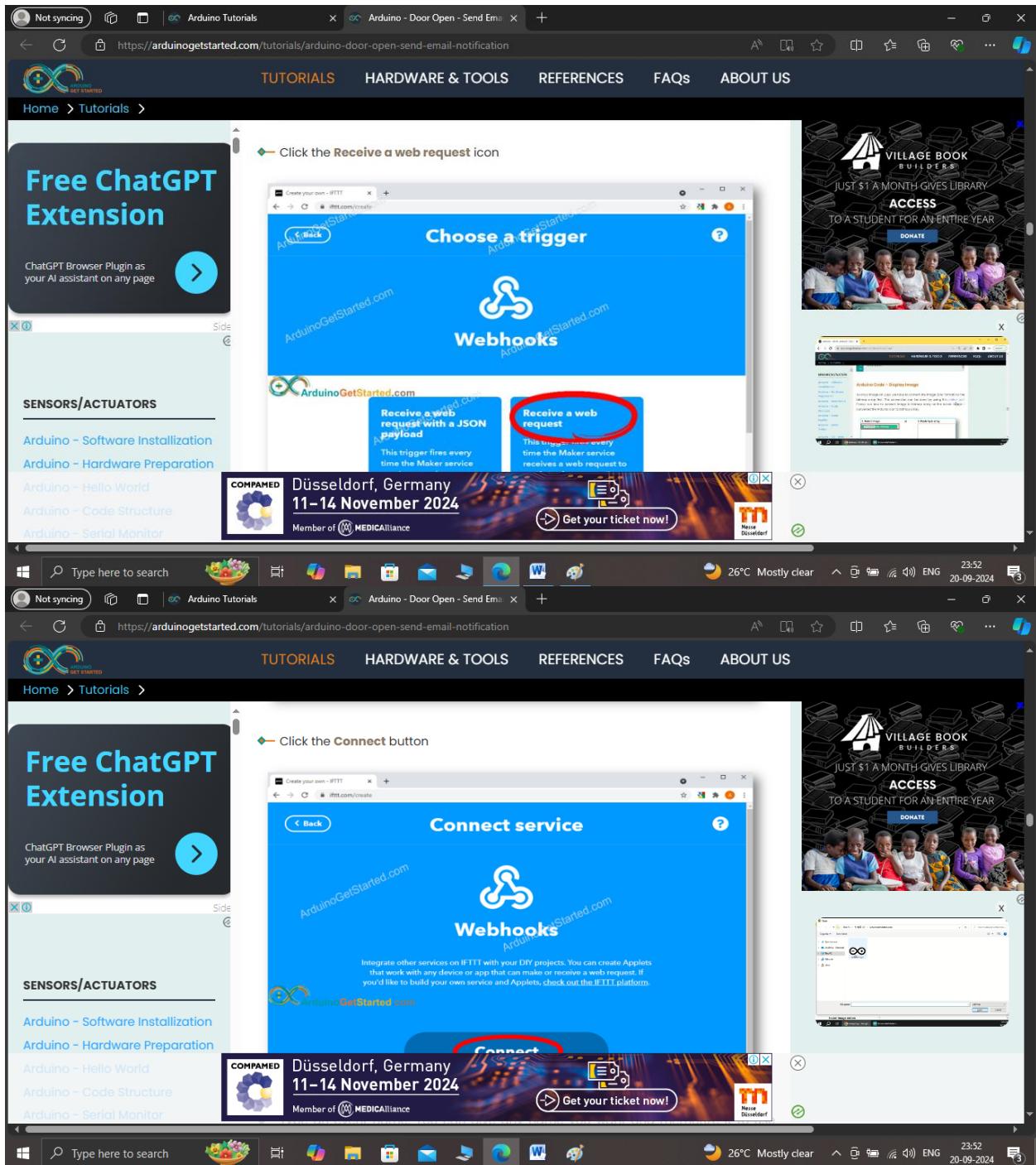


The image displays two screenshots of a Microsoft Edge browser window, showing the process of creating an IFTTT webhook for an Arduino project.

Screenshot 1: The browser shows the "Create your own" screen in the IFTTT interface. A callout arrow points to the "Add" button on the "If This" card, which is circled in red. The URL in the address bar is <https://arduinogetstarted.com/tutorials/arduino-door-open-send-email-notification>.

Screenshot 2: The browser shows the "Choose a service" screen in the IFTTT interface. A callout arrow points to the "Webhooks" search input field, which is circled in red. Below it, the "Webhooks" service icon is also circled in red. The URL in the address bar is <https://arduinogetstarted.com/tutorials/arduino-door-open-send-email-notification>.

Both screenshots include a sidebar with "Free ChatGPT Extension" and "SENSORS/ACTUATORS" sections, and a banner for COMPAMED Düsseldorf, Germany, 11-14 November 2024.

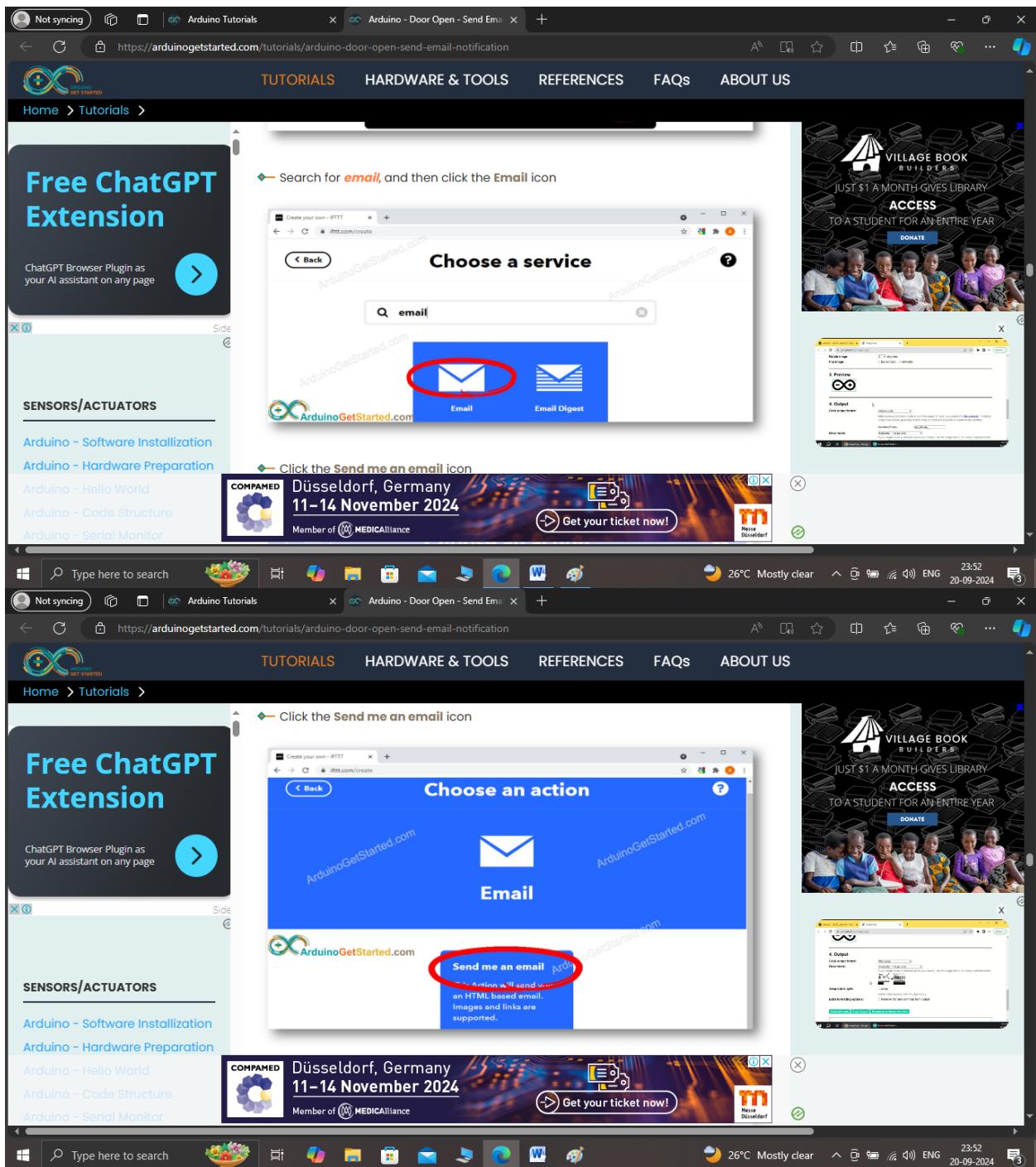


The screenshot shows a Windows desktop environment with a web browser open to the ArduinoGetStarted.com website. The main content area displays a tutorial titled "Free ChatGPT Extension". Below the main content, there is a sidebar with "SENSORS/ACTUATORS" and a list of links: "Arduino - Software Installation", "Arduino - Hardware Preparation", "Arduino - Hello World", "Arduino - Code Structure", and "Arduino - Serial Monitor".

The central part of the screen shows the IFTTT "Create your own" interface. A tooltip says: "Type an event name. You can give any name you want and memorize it to use later. The event name is a part of URL that Arduino will make request to. In this tutorial, we use a name `send-email`. And then click Create trigger button". A red circle highlights the text "send-email" in the "Event Name" input field.

Below this, another tooltip says: "Click the Add button to add the action". A red circle highlights the "Add" button in the "Then That" section.

The status bar at the bottom of the screen shows: "Not syncing", "Type here to search", "Arduino Tutorials", "Arduino - Door Open - Send Email", "26°C Mostly clear", "ENG 20-09-2024", and a battery icon.



The screenshot displays a Microsoft Edge browser window with two tabs open. The top tab shows a guide titled "Free ChatGPT Extension" from "Arduino Get Started". It instructs users to click the "Connect" button on the IFTTT service. A red circle highlights the "Connect" button on the IFTTT interface. The bottom tab shows the IFTTT step where an email address is being entered, with a red circle highlighting the "Send PIN" button. The browser's taskbar at the bottom shows various pinned icons and the date/time as 20-09-2024.

The screenshot shows a Microsoft Edge browser window with two tabs open. The active tab is titled "Arduino - Door Open - Send Email" and displays a tutorial from "arduinogetstarted.com". The tutorial explains how to receive an email PIN and verify it in a web UI. It includes a screenshot of a browser window titled "Send me an email" with a subject line "The event named \"{{EventName}}\" occurred on the Maker Webhooks service". The second tab is titled "arduinogetstarted.com/tutorials/arduino-door-open-send-email-notification" and shows a sidebar for "Free ChatGPT Extension" and an advertisement for "VILLAGE BOOK BUILDERS". The taskbar at the bottom shows various pinned icons and the date/time as 20-09-2024.

The screenshot shows a Microsoft Edge browser window with the following details:

- Address Bar:** https://arduinogetstarted.com/tutorials/arduino-door-open-send-email-notification
- Header:** Arduino Tutorials, Arduino - Door Open - Send Email
- Left Sidebar:**
 - Free ChatGPT Extension:** ChatGPT Browser Plugin as your AI assistant on any page.
 - SENSORS/ACTUATORS:** Arduino - Software Installation, Arduino - Hardware Preparation, Arduino - Hello World, Arduino - Code Structure, Arduino - Serial Monitor.
- Middle Content:**
 - A modal window titled "Free ChatGPT Extension" shows a sample email message with the subject "Alert from Arduino" and content "Hey, Someone opened your door! Take care! Arduino Uno". Below it, instructions say: "Click the Create Action button" and "Click the Continue button".
 - An IFTTT integration window titled "Create your own - IFTTT" shows a sequence of steps: "If" (Receive a web-request) followed by "Then" (Send me an email). A large red circle highlights the "Continue" button at the bottom of the IFTTT interface.
 - A banner for "COMPAMED Düsseldorf, Germany 11-14 November 2024" is visible.
 - A sidebar for "VILLAGE BOOK BUILDERS" with the text "JUST \$1 A MONTH GIVES LIBRARY ACCESS TO A STUDENT FOR AN ENTIRE YEAR" and a "DONATE" button.
- Bottom:** Windows taskbar with various pinned icons and system status.

Free ChatGPT Extension

ChatGPT Browser Plugin as your AI assistant on any page

SENSORS/ACTUATORS

- Arduino - Software Installation
- Arduino - Hardware Preparation
- Arduino - Hello World
- Arduino - Code Structure
- Arduino - Serial Monitor

TUTORIALS **HARDWARE & TOOLS** **REFERENCES** **FAQS** **ABOUT US**

Click the Finish button

Review and finish

If Maker Event "send-email", then Send me an email at z*****@gmail.com

by study81 7/146

Finish

VILLAGE BOOK BUILDERS
JUST \$1 A MONTH GIVES LIBRARY ACCESS TO A STUDENT FOR AN ENTIRE YEAR DONATE

Type here to search

Not syncing

Arduino Tutorials

Arduino - Door Open - Send Email

COMPAMED Düsseldorf, Germany 11-14 November 2024 Member of MEDICA Alliance Get your ticket now! Messe Düsseldorf

26°C Mostly clear 23:53 20-09-2024

Free ChatGPT Extension

ChatGPT Browser Plugin as your AI assistant on any page

SENSORS/ACTUATORS

- Arduino - Software Installation
- Arduino - Hardware Preparation
- Arduino - Hello World
- Arduino - Code Structure
- Arduino - Serial Monitor

TUTORIALS **HARDWARE & TOOLS** **REFERENCES** **FAQS** **ABOUT US**

Now you succeeded to create an IFTTT Applet for your Arduino. The next step is to get the IFTTT Webhooks key, which is used to authenticate and identify your Arduino.

- Visit IFTTT Webhooks page
- Click the Documentation button

Webhooks

Integrate other services on IFTTT with your DIY projects. You can create Applets that work with any device or app that connects to IFTTT. If you'd like to build your own service and Applets, check out the IFTTT API.

Create Documentation

VILLAGE BOOK BUILDERS
JUST \$1 A MONTH GIVES LIBRARY ACCESS TO A STUDENT FOR AN ENTIRE YEAR DONATE

Type here to search

Not syncing

Arduino Tutorials

Arduino - Door Open - Send Email

COMPAMED Düsseldorf, Germany 11-14 November 2024 Member of MEDICA Alliance Get your ticket now! Messe Düsseldorf

26°C Mostly clear 23:53 20-09-2024

You will see the **Webhooks key** as below

Your key is: **qmBf3Q*****nwD6**

To trigger an Event
Make a POST or GET web request to:
https://maker.ifttt.com/trigger/{}/with/key/qmBf3Q*****nwD6

Copy and wire down your Webhooks to use on Arduino code.

Now, We just need to write Arduino code that makes an HTTP request to the below URL:

<https://maker.ifttt.com/trigger/send-email/with/key/xxxxxxxxxxxxxxxxxxxx>

*** NOTE THAT:**

- In the URL, the **send-email** is the event name we used:
 - If you already used this name for another **Applet**, you can give it any other name.
 - If you use another name, please replace the **send-email** by your event name.
- Webhooks key** is generated by IFTTT. Please keep it secret. You can change it by regenerating it on the IFTTT website.

Before writing Arduino code, We can test the IFTTT **Applet** by doing:

- Open a web browser
- Copy the below link:

Düsseldorf, Germany
11-14 November 2024
Member of MEDICA Alliance

Before writing Arduino code, We can test the IFTTT **Applet** by doing:

- Open a web browser
- Copy the below link:

<https://maker.ifttt.com/trigger/send-email/with/key/XXXXXXXXXXXXXXXXXXXXXX>

Paste it on the address bar of the web browser
Replace XXXXXXXXXXXXXXXXX with your **Webhooks key**.
Press the enter key on your keyboard. You will see the browser display the message as below image.

Congratulations! You've fired the send-email event

And you will receive an email.

Manual Torque Wrenches

VILLAGE BOOK BUILDERS
JUST \$1 A MONTH GIVES LIBRARY
ACCESS
TO A STUDENT FOR AN ENTIRE YEAR
DONATE

WREN INDIA
Professional manufacturer of hydraulic equipment from 70Mpa to 300Mpa pressure.

COMPAMED Düsseldorf, Germany 11-14 November 2024 Member of MEDICA Alliance Get your ticket now!

```
#include <WiFiS3.h>

#include <ezButton.h>

const char ssid[] = "YOUR_WIFI_SSID"; // change your
network SSID (name)

const char pass[] = "YOUR_WIFI_PASSWORD"; // 
change your network password (use for WPA, or use as
key for WEP)

ezButton doorSensor(13); // create ezButton object
that attach to pin 13, which connected to door sensor's
pin

WiFiClient client;

int status = WL_IDLE_STATUS;

int HTTP_PORT = 80;

String HTTP_METHOD = "GET";
```

```
char HOST_NAME[] = "maker.ifttt.com";  
  
String PATH_NAME = "/trigger/send-  
email/with/key/XXXXXXXXXXXXXXXXXXXXXX"; //  
change your Webhooks key  
  
void setup() {  
  
    Serial.begin(9600);  
  
    Serial.println("DOOR MONITORING via EMAIL  
STARTED!");  
  
    // check for the WiFi module:  
  
    if (WiFi.status() == WL_NO_MODULE) {  
  
        Serial.println("Communication with WiFi module  
failed!");  
  
        // don't continue  
  
        while (true)  
  
        ;  
    }  
}
```

```
String fv = WiFi.firmwareVersion();  
  
if (fv < WIFI_FIRMWARE_LATEST_VERSION) {  
  
    Serial.println("Please upgrade the firmware");  
  
}  
  
  
  
// attempt to connect to WiFi network:  
  
while (status != WL_CONNECTED) {  
  
    Serial.print("Attempting to connect to SSID: ");  
  
    Serial.println(ssid);  
  
    // Connect to WPA/WPA2 network. Change this line if  
using open or WEP network:  
  
    status = WiFi.begin(ssid, pass);  
  
  
  
    // wait 10 seconds for connection:  
  
    delay(10000);  
  
}
```

```
// print your board's IP address:  
Serial.print("IP Address: ");  
Serial.println(WiFi.localIP());  
  
doorSensor.setDebounceTime(50); // set debounce  
time to 50 milliseconds  
}  
  
void loop() {  
    doorSensor.loop(); // MUST call the loop() function  
    first  
  
    if (doorSensor.isReleased()) { // LOW to HIGH  
        Serial.println("The door is opened!");  
        Serial.println("Arduino is sending email");  
        sendEmail();
```

```
}

}

void sendEmail() {

    // connect to IFTTT server on port 80:

    if (client.connect(HOST_NAME, HTTP_PORT)) {

        // if connected:

        Serial.println("Connected to server");

        // make a HTTP request:

        // send HTTP header

        client.println("GET " + PATH_NAME + " HTTP/1.1");

        client.println("Host: " + String(HOST_NAME));

        client.println("Connection: close");

        client.println(); // end HTTP header

    }

    while (client.connected()) {

        if (client.available()) {
```

```
// read an incoming byte from the server and print  
it to serial monitor:
```

```
char c = client.read();
```

```
Serial.print(c);
```

```
}
```

```
}
```

```
// the server's disconnected, stop the client:
```

```
client.stop();
```

```
Serial.println();
```

```
Serial.println("disconnected");
```

```
} else { // if not connected:
```

```
Serial.println("connection failed");
```

```
}
```

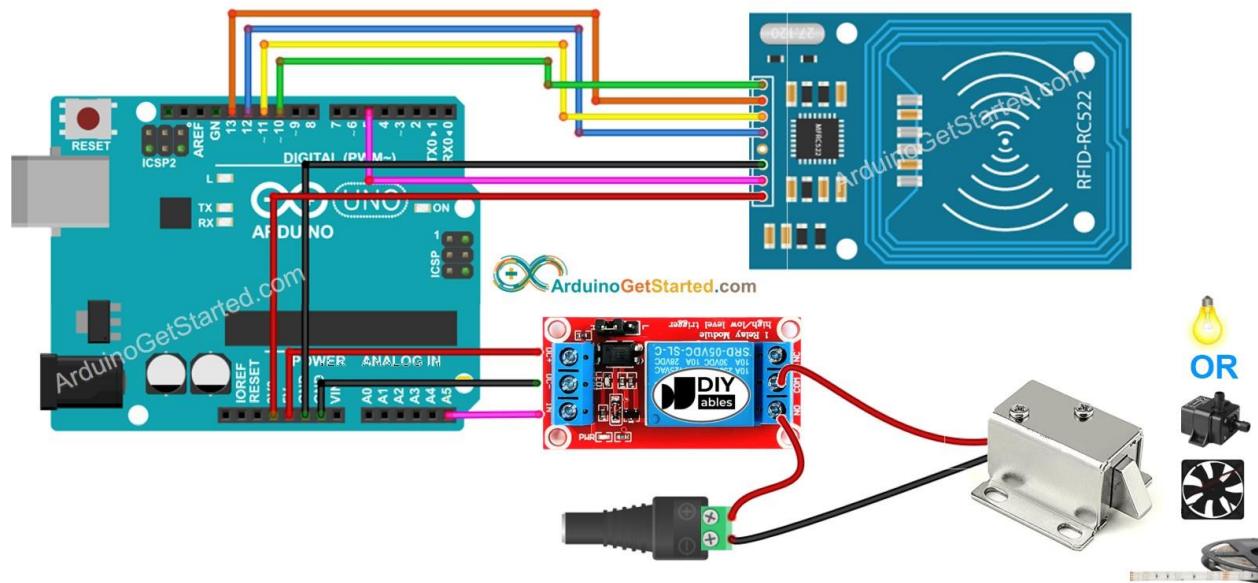
```
}
```

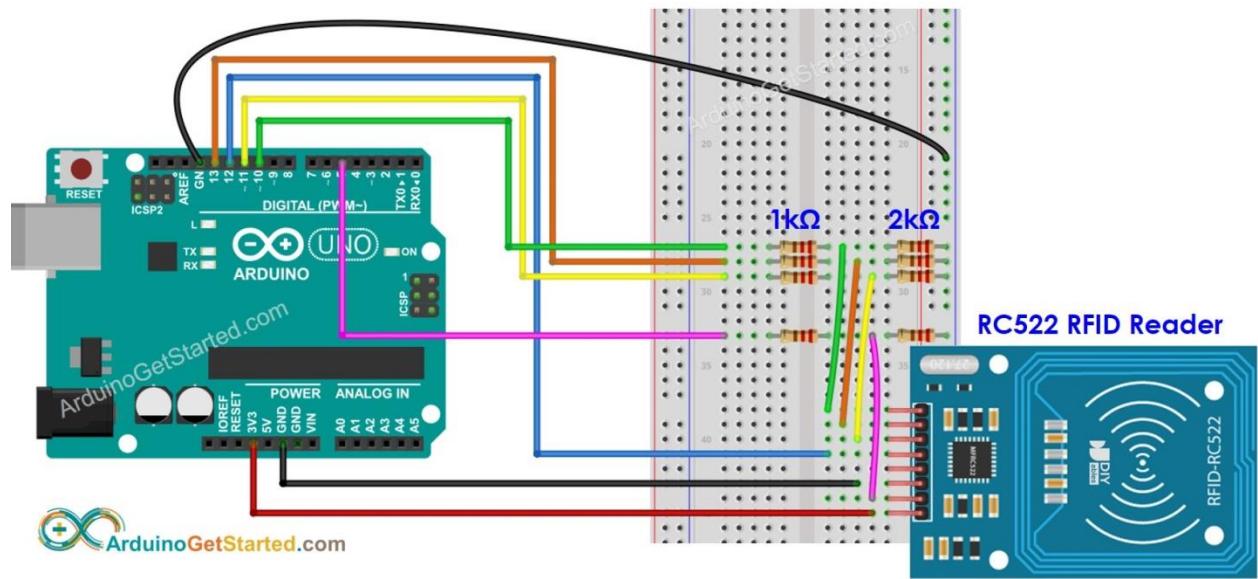
RFID NFC RELAY

RC522



ArduinoGetStarted.com





```
#include <SPI.h>
```

```
#include <MFRC522.h>
```

```
#define SS_PIN 10
```

```
#define RST_PIN 5
```

```
#define RELAY_PIN A5 // the Arduino pin connects to  
relay
```

```
MFRC522 rfid(SS_PIN, RST_PIN);
```

```
byte authorizedUID[4] = {0xFF, 0xFF, 0xFF, 0xFF};
```

```
void setup() {  
    Serial.begin(9600);  
    SPI.begin(); // init SPI bus  
    rfid.PCD_Init(); // init MFRC522  
    pinMode(RELAY_PIN, OUTPUT); // initialize pin as an  
    output.  
    digitalWrite(RELAY_PIN, LOW); // deactivate the relay  
  
    Serial.println("Tap RFID/NFC Tag on reader");  
}  
  
void loop() {  
    if (rfid.PICC_IsNewCardPresent()) { // new tag is  
    available  
        if (rfid.PICC_ReadCardSerial()) { // NUID has been  
        readed
```

```
MFRC522::PICC_Type piccType =  
rfid.PICC_GetType(rfid.uid.sak);  
  
if (rfid.uid.uidByte[0] == authorizedUID[0] &&  
    rfid.uid.uidByte[1] == authorizedUID[1] &&  
    rfid.uid.uidByte[2] == authorizedUID[2] &&  
    rfid.uid.uidByte[3] == authorizedUID[3] ) {  
    Serial.println("Authorized Tag");  
    digitalWrite(RELAY_PIN, HIGH); // activate the  
    relay for 2 seconds  
    delay(2000);  
    digitalWrite(RELAY_PIN, LOW); // deactivate the  
    relay  
}  
else  
{  
    Serial.print("Unauthorized Tag with UID:");  
    for (int i = 0; i < rfid.uid.size; i++) {
```

```
Serial.print(rfid.uid.uidByte[i] < 0x10 ? " 0" : " ");

Serial.print(rfid.uid.uidByte[i], HEX);

}

Serial.println();

}

rfid.PICC_HaltA(); // halt PICC

rfid.PCD_StopCrypto1(); // stop encryption on PCD

}

}

}
```

MULTIPLE RFID TAGS

```
#include <SPI.h>
#include <MFRC522.h>

#define SS_PIN 10
#define RST_PIN 5
#define RELAY_PIN A5 // the Arduino pin connects to
relay

MFRC522 rfid(SS_PIN, RST_PIN);

byte authorizedUID1[4] = {0x3A, 0xC9, 0x6A, 0xCB};
byte authorizedUID2[4] = {0x30, 0x01, 0x8B, 0x15};

void setup() {
    Serial.begin(9600);
    SPI.begin(); // init SPI bus
    rfid.PCD_Init(); // init MFRC522
```

```
pinMode(RELAY_PIN, OUTPUT); // initialize pin as an
output.

digitalWrite(RELAY_PIN, LOW); // deactivate the relay

Serial.println("Tap RFID/NFC Tag on reader");

}

void loop() {

if (rfid.PICC_IsNewCardPresent()) { // new tag is
available

if (rfid.PICC_ReadCardSerial()) { // NUID has been
readed

MFRC522::PICC_Type piccType =
rfid.PICC_GetType(rfid.uid.sak);

if (rfid.uid.uidByte[0] == authorizedUID1[0] &&
rfid.uid.uidByte[1] == authorizedUID1[1] &&
rfid.uid.uidByte[2] == authorizedUID1[2] &&
```

```
rfid.uid.uidByte[3] == authorizedUID1[3] ) {  
    Serial.println("Authorized Tag 1");  
    digitalWrite(RELAY_PIN, HIGH); // activate the  
    relay for 2 seconds  
    delay(2000);  
    digitalWrite(RELAY_PIN, LOW); // deactivate the  
    relay  
}  
  
else  
  
if (rfid.uid.uidByte[0] == authorizedUID2[0] &&  
    rfid.uid.uidByte[1] == authorizedUID2[1] &&  
    rfid.uid.uidByte[2] == authorizedUID2[2] &&  
    rfid.uid.uidByte[3] == authorizedUID2[3] ) {  
    Serial.println("Authorized Tag 2");  
    digitalWrite(RELAY_PIN, HIGH); // activate the  
    relay for 2 seconds  
    delay(2000);
```

```
    digitalWrite(RELAY_PIN, LOW); // deactivate the
    relay
}

else {
    Serial.print("Unauthorized Tag with UID:");
    for (int i = 0; i < rfid.uid.size; i++) {
        Serial.print(rfid.uid.uidByte[i] < 0x10 ? " 0" : " ");
        Serial.print(rfid.uid.uidByte[i], HEX);
    }
    Serial.println();
}

rfid.PICC_HaltA(); // halt PICC
rfid.PCD_StopCrypto1(); // stop encryption on PCD
}
```

}

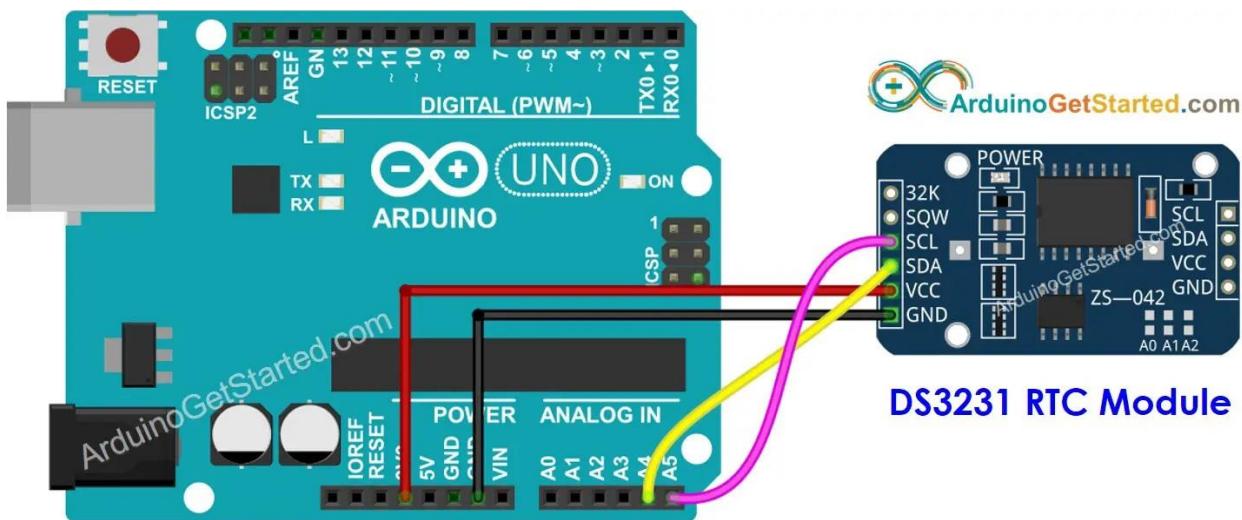
REAL TIME CLOCK

TIME CLOCK DS3231 MODULE



DS3231 RTC Module

- ◆ **32K pin:** outputs the stable(temperature compensated) and accurate reference clock.
- ◆ **SQW pin:** outputs a nice square wave at either 1Hz, 4kHz, 8kHz or 32kHz and can be handled programmatically. This can further be used as an interrupt due to alarm condition in many time-based applications.
- ◆ **SCL pin:** is a serial clock pin for I2C interface.
- ◆ **SDA pin:** is a serial data pin for I2C interface.
- ◆ **VCC pin:** supplies power for the module. It can be anywhere between 3.3V to 5.5V.
- ◆ **GND pin:** is a ground pin.



Arduino - DS3231 RTC Module

DS3231 RTC Module	Arduino Uno, Nano	Arduino Mega
Vin	3.3V	3.3V
GND	GND	GND
SDA	A4	20
SCL	A5	21

```
#include <RTClib.h>
```

```
RTC_DS3231 rtc;
```

```
char daysOfTheWeek[7][12] = {
```

```
    "Sunday",
```

```
    "Monday",
```

```
    "Tuesday",
```

```
    "Wednesday",
```

```
    "Thursday",
```

```
    "Friday",
```

```
    "Saturday"
```

```
};
```

```
void setup () {
```

```
    Serial.begin(9600);
```

```
// SETUP RTC MODULE
```

```
if (! rtc.begin()) {  
    Serial.println("Couldn't find RTC");  
    Serial.flush();  
    while (1);  
}
```

```
// automatically sets the RTC to the date & time
```

on PC this sketch was compiled

```
rtc.adjust(DateTime(F(__DATE__)), F(__TIME__)));
```

```
// manually sets the RTC with an explicit date &  
time, for example to set
```

```
// January 21, 2021 at 3am you would call:
```

```
// rtc.adjust(DateTime(2021, 1, 21, 3, 0, 0));
```

```
}
```

```
void loop () {  
  
    DateTime now = rtc.now();  
  
    Serial.print("Date & Time: ");  
  
    Serial.print(now.year(), DEC);  
  
    Serial.print('/');  
  
    Serial.print(now.month(), DEC);  
  
    Serial.print('/');  
  
    Serial.print(now.day(), DEC);  
  
    Serial.print(" (");  
  
    Serial.print(daysOfTheWeek[now.dayOfTheWeek()]);  
  
    Serial.print(") ");  
  
    Serial.print(now.hour(), DEC);  
  
    Serial.print(':');  
  
    Serial.print(now.minute(), DEC);  
  
    Serial.print(':');  
  
    Serial.println(now.second(), DEC);
```

```
delay(1000); // delay 1 seconds  
}
```

DAILY SCHEDULE

```
// Date and time functions using a DS3231 RTC  
connected via I2C and Wire lib  
  
#include <RTCLib.h>  
  
// event from 13:50 to 14:10  
  
uint8_t DAILY_EVENT_START_HH = 13; // event start  
time: hour  
  
uint8_t DAILY_EVENT_START_MM = 50; // event start  
time: minute  
  
uint8_t DAILY_EVENT_END_HH = 14; // event end  
time: hour
```

```
uint8_t DAILY_EVENT_END_MM = 10; // event end  
time: minute  
  
RTC_DS3231 rtc;  
  
char daysOfTheWeek[7][12] = {  
    "Sunday",  
    "Monday",  
    "Tuesday",  
    "Wednesday",  
    "Thursday",  
    "Friday",  
    "Saturday"  
};  
  
void setup () {  
    Serial.begin(9600);
```

```
// SETUP RTC MODULE

if (! rtc.begin()) {

    Serial.println("Couldn't find RTC");

    while (1);

}

// sets the RTC to the date & time on PC this sketch
// was compiled

rtc.adjust(DateTime(F(__DATE__)), F(__TIME__)));



// sets the RTC with an explicit date & time, for
// example to set

// January 21, 2021 at 3am you would call:
// rtc.adjust(DateTime(2021, 1, 21, 3, 0, 0));

}
```

```
void loop () {  
    DateTime now = rtc.now();  
  
    if (now.hour()  >= DAILY_EVENT_START_HH &&  
        now.minute() >= DAILY_EVENT_START_MM &&  
        now.hour()  < DAILY_EVENT_END_HH   &&  
        now.minute() < DAILY_EVENT_END_MM) {  
  
        Serial.println("It is on scheduled time");  
  
        // TODO: write your code"  
  
    } else {  
  
        Serial.println("It is NOT on scheduled time");  
  
    }  
  
    printTime(now);  
}  
  
void printTime(DateTime time) {
```

```
Serial.print("TIME: ");
Serial.print(time.year(), DEC);
Serial.print('/');
Serial.print(time.month(), DEC);
Serial.print('/');
Serial.print(time.day(), DEC);
Serial.print(" (");
Serial.print(daysOfTheWeek[time.dayOfTheWeek()]);
Serial.print(") ");
Serial.print(time.hour(), DEC);
Serial.print(':');
Serial.print(time.minute(), DEC);
Serial.print(':');
Serial.println(time.second(), DEC);
}
```

WEEKLY SCHEDULE

```
// Date and time functions using a DS3231 RTC  
connected via I2C and Wire lib
```

```
#include <RTCLib.h>
```

```
// UNCHANGABLE PARAMATERS
```

```
#define SUNDAY 0
```

```
#define MONDAY 1
```

```
#define TUESDAY 2
```

```
#define WEDNESDAY 3
```

```
#define THURSDAY 4
```

```
#define FRIDAY 5
```

```
#define SATURDAY 6
```

```
// event on Monday, from 13:50 to 14:10
```

```
uint8_t WEEKLY_EVENT_DAY = MONDAY;
```

```
uint8_t WEEKLY_EVENT_START_HH = 13; // event start  
time: hour  
  
uint8_t WEEKLY_EVENT_START_MM = 50; // event start  
time: minute  
  
uint8_t WEEKLY_EVENT_END_HH = 14; // event end  
time: hour  
  
uint8_t WEEKLY_EVENT_END_MM = 10; // event end  
time: minute  
  
RTC_DS3231 rtc;  
  
char daysOfTheWeek[7][12] = {  
    "Sunday",  
    "Monday",  
    "Tuesday",  
    "Wednesday",  
    "Thursday",  
    "Friday",
```

```
"Saturday"  
};  
  
void setup () {  
    Serial.begin(9600);  
  
    // SETUP RTC MODULE  
    if (! rtc.begin()) {  
        Serial.println("Couldn't find RTC");  
        while (1);  
    }  
  
    // sets the RTC to the date & time on PC this sketch  
    // was compiled  
    rtc.adjust(DateTime(F(__DATE__)), F(__TIME__)));
```

```
// sets the RTC with an explicit date & time, for  
example to set
```

```
// January 21, 2021 at 3am you would call:
```

```
// rtc.adjust(DateTime(2021, 1, 21, 3, 0, 0));
```

```
}
```

```
void loop () {
```

```
    DateTime now = rtc.now();
```

```
    if (now.dayOfTheWeek() == WEEKLY_EVENT_DAY  
        &&
```

```
        now.hour()      >= WEEKLY_EVENT_START_HH &&
```

```
        now.minute()     >= WEEKLY_EVENT_START_MM
```

```
        &&
```

```
        now.hour()      < WEEKLY_EVENT_END_HH  &&
```

```
        now.minute()     < WEEKLY_EVENT_END_MM) {
```

```
    Serial.println("It is on scheduled time");
```

```
    // TODO: write your code"
```

```
} else {  
    Serial.println("It is NOT on scheduled time");  
}  
  
printTime(now);  
}  
  
void printTime(DateTime time) {  
    Serial.print("TIME: ");  
    Serial.print(time.year(), DEC);  
    Serial.print('/');  
    Serial.print(time.month(), DEC);  
    Serial.print('/');  
    Serial.print(time.day(), DEC);  
    Serial.print(" (");  
    Serial.print(daysOfTheWeek[time.dayOfTheWeek()]);  
    Serial.print(") ");
```

```
Serial.print(time.hour(), DEC);
Serial.print(':');
Serial.print(time.minute(), DEC);
Serial.print(':');
Serial.println(time.second(), DEC);
}
```

SCHEDULE FOR SPECIFIC DATE

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
// Date and time functions using a DS3231 RTC
// connected via I2C and Wire lib

#include <RTCLib.h>

// UNCHANGABLE PARAMATERS

#define SUNDAY 0
#define MONDAY 1
#define TUESDAY 2
```

```
#define WEDNESDAY 3  
  
#define THURSDAY 4  
  
#define FRIDAY 5  
  
#define SATURDAY 6  
  
  
  
#define JANUARY 1  
  
#define FEBRUARY 2  
  
#define MARCH 3  
  
#define APRIL 4  
  
#define MAY 5  
  
#define JUNE 6  
  
#define JULY 7  
  
#define AUGUST 8  
  
#define SEPTEMBER 9  
  
#define OCTOBER 10  
  
#define NOVEMBER 11  
  
#define DECEMBER 12
```

```
// event from 13:50 August 15, 2021 to 14:10
```

```
September 29, 2021
```

```
DateTime EVENT_START(2021, AUGUST, 15, 13, 50);
```

```
DateTime EVENT_END(2021, SEPTEMBER, 29, 14, 10);
```

```
RTC_DS3231 rtc;
```

```
char daysOfTheWeek[7][12] = {
```

```
    "Sunday",
```

```
    "Monday",
```

```
    "Tuesday",
```

```
    "Wednesday",
```

```
    "Thursday",
```

```
    "Friday",
```

```
    "Saturday"
```

```
};
```

```
void setup () {  
    Serial.begin(9600);  
  
    // SETUP RTC MODULE  
    if (! rtc.begin()) {  
        Serial.println("Couldn't find RTC");  
        Serial.flush();  
        while (1);  
    }  
  
    // sets the RTC to the date & time on PC this sketch  
    // was compiled  
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));  
  
    // sets the RTC with an explicit date & time, for  
    // example to set
```

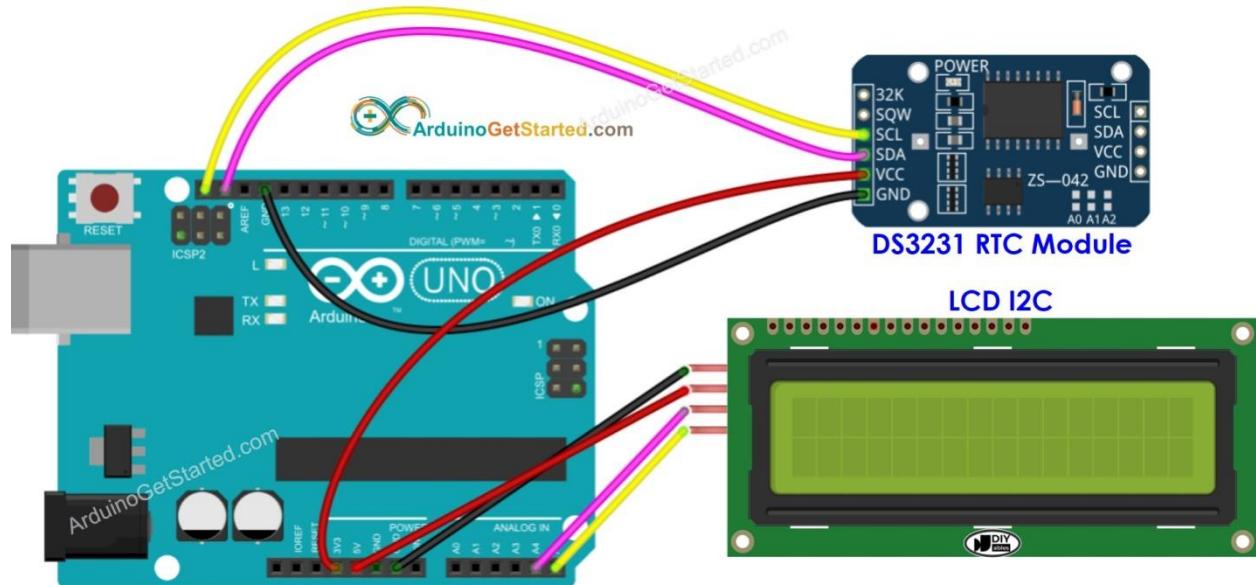
```
// January 21, 2021 at 3am you would call:  
  
// rtc.adjust(DateTime(2021, 1, 21, 3, 0, 0));  
  
}  
  
void loop () {  
  
    DateTime now = rtc.now();  
  
    if (now.secondstime() >= EVENT_START.secondstime()  
&&  
        now.secondstime() < EVENT_END.secondstime()) {  
  
        Serial.println("It is on scheduled time");  
  
        // TODO: write your code"  
  
    } else {  
  
        Serial.println("It is NOT on scheduled time");  
  
    }  
}
```

```
printTime(now);  
}  
  
void printTime(DateTime time) {  
    Serial.print("TIME: ");  
    Serial.print(time.year(), DEC);  
    Serial.print('/');  
    Serial.print(time.month(), DEC);  
    Serial.print('/');  
    Serial.print(time.day(), DEC);  
    Serial.print(" (");  
    Serial.print(daysOfTheWeek[time.dayOfTheWeek()]);  
    Serial.print(") ");  
    Serial.print(time.hour(), DEC);  
    Serial.print(':');  
    Serial.print(time.minute(), DEC);  
    Serial.print(':');
```

```
Serial.println(time.second(), DEC);
```

```
}
```

LCD CLOCK



```
#include <LiquidCrystal_I2C.h>
```

```
#include <RTCLib.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27  
(from DIYables LCD), 16 column and 2 rows  
  
RTC_DS3231 rtc;  
  
void setup() {  
    Serial.begin(9600);  
  
    lcd.init();      // initialize the lcd  
    lcd.backlight(); // open the backlight  
  
    // SETUP RTC MODULE  
    if (!rtc.begin()) {  
        Serial.println("Couldn't find RTC");  
        Serial.flush();  
        while (true)  
            ;  
    }  
}
```

```
// automatically sets the RTC to the date & time on PC  
this sketch was compiled  
  
rtc.adjust(DateTime(F(__DATE__)), F(__TIME__)));  
}  
  
void loop() {  
    DateTime now = rtc.now();  
  
    int year = now.year();  
    int month = now.month();  
    int day = now.day();  
    int hour = now.hour();  
    int minute = now.minute();  
    int second = now.second();  
  
    lcd.clear();
```

```
lcd.setCursor(0, 0); // start to print at the first row
```

```
lcd.print("Date: ");
```

```
lcd.print(year);
```

```
lcd.print("/");
```

```
lcd.print(month);
```

```
lcd.print("/");
```

```
lcd.print(day);
```

```
lcd.setCursor(0, 1); // start to print at the second row
```

```
lcd.print("Time: ");
```

```
lcd.print(hour);
```

```
lcd.print(":");
```

```
lcd.print(minute);
```

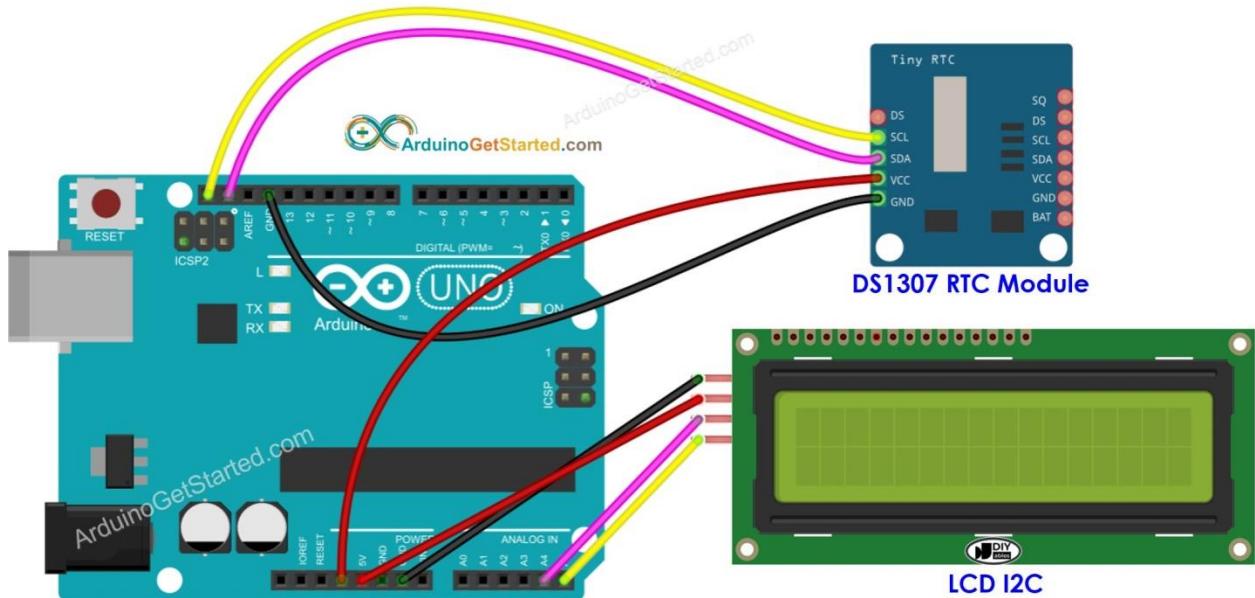
```
lcd.print(":");
```

```
lcd.print(second);
```

```
delay(1000); // Update every second
```

}

READING DATA FROM RTC AND DISPLAY IT TO LCD



```
#include <LiquidCrystal_I2C.h>
```

```
#include <RTCLib.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27
```

```
(from DIYables LCD), 16 column and 2 rows
```

```
RTC_DS1307 rtc;
```

```
void setup() {
```

```
Serial.begin(9600);

lcd.init();      // initialize the lcd

lcd.backlight(); // open the backlight

// SETUP RTC MODULE

if (!rtc.begin()) {

    Serial.println("Couldn't find RTC");

    Serial.flush();

    while (true)

    ;

}

// automatically sets the RTC to the date & time on PC
this sketch was compiled

rtc.adjust(DateTime(F(__DATE__)), F(__TIME__)));

}
```

```
void loop() {  
    DateTime now = rtc.now();  
  
    int year = now.year();  
    int month = now.month();  
    int day = now.day();  
    int hour = now.hour();  
    int minute = now.minute();  
    int second = now.second();  
  
    lcd.clear();  
    lcd.setCursor(0, 0); // start to print at the first row  
    lcd.print("Date: ");  
    lcd.print(year);  
    lcd.print("/");  
    lcd.print(month);  
}
```

```
lcd.print("/");
lcd.print(day);

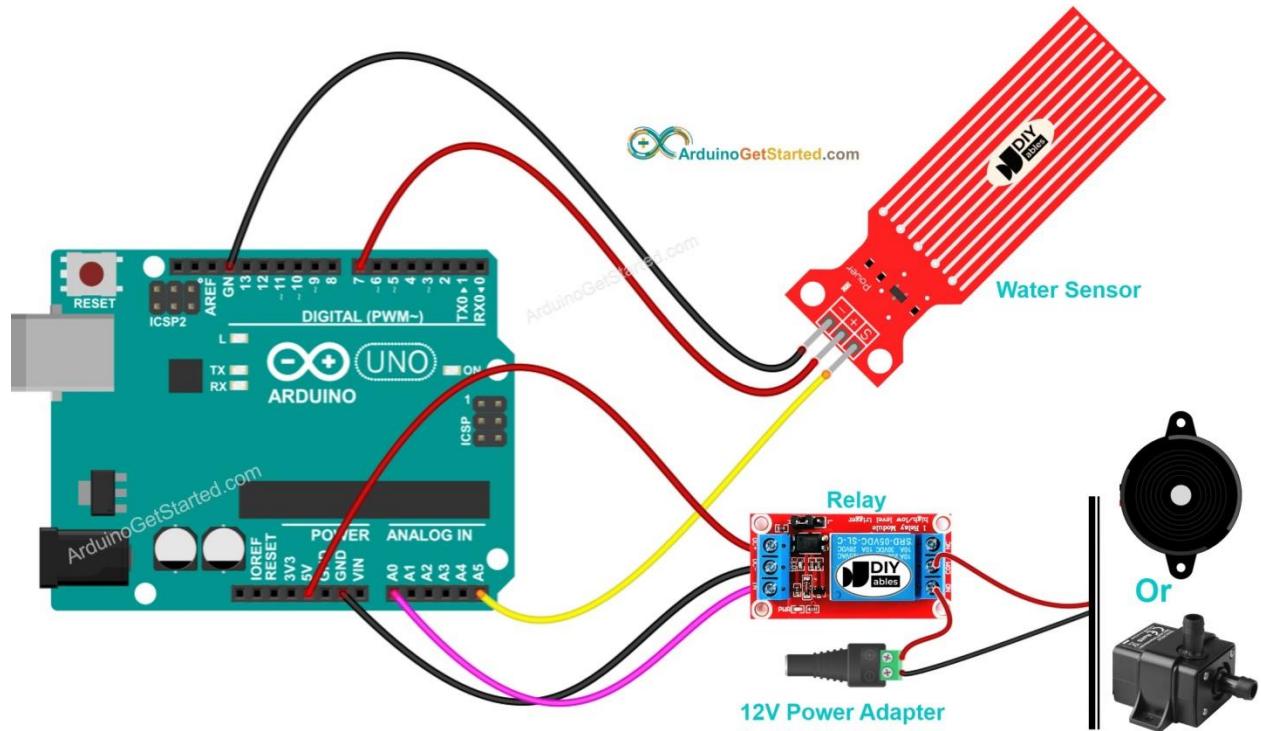
lcd.setCursor(0, 1); // start to print at the second row

lcd.print("Time: ");
lcd.print(hour);
lcd.print(":");
lcd.print(minute);
lcd.print(":");
lcd.print(second);

delay(1000); // Update every second

}
```

WATER SENSOR



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
#define RELAY_PIN A0 // The Arduino pin that
connects to the relay
```

```
#define POWER_PIN 7 // The Arduino pin that
provides the power to the water sensor
```

```
#define SIGNAL_PIN A5 // The Arduino pin that reads
the value from the water sensor
```

```
#define THRESHOLD 500 // The threshold for water  
detectiion  
  
void setup() {  
  
Serial.begin(9600);  
  
pinMode(RELAY_PIN, OUTPUT); // configure D2 pin as  
an OUTPUT  
  
pinMode(POWER_PIN, OUTPUT); // configure D7 pin  
as an OUTPUT  
  
digitalWrite(POWER_PIN, LOW); // turn the water  
sensor OFF  
  
digitalWrite(RELAY_PIN, LOW); // dectivates relay  
}  
  
  
void loop() {  
  
digitalWrite(POWER_PIN, HIGH); // turn the water  
sensor's power ON  
  
delay(10); // wait 10 milliseconds
```

```
int value = analogRead(SIGNAL_PIN); // read the
analog value from sensor

digitalWrite(POWER_PIN, LOW); // turn the water
sensor's power OFF

if (value > THRESHOLD) {
    Serial.print("The water is detected");
    digitalWrite(RELAY_PIN, HIGH); // activates relay
} else {
    digitalWrite(RELAY_PIN, LOW); // deactivates relay
}

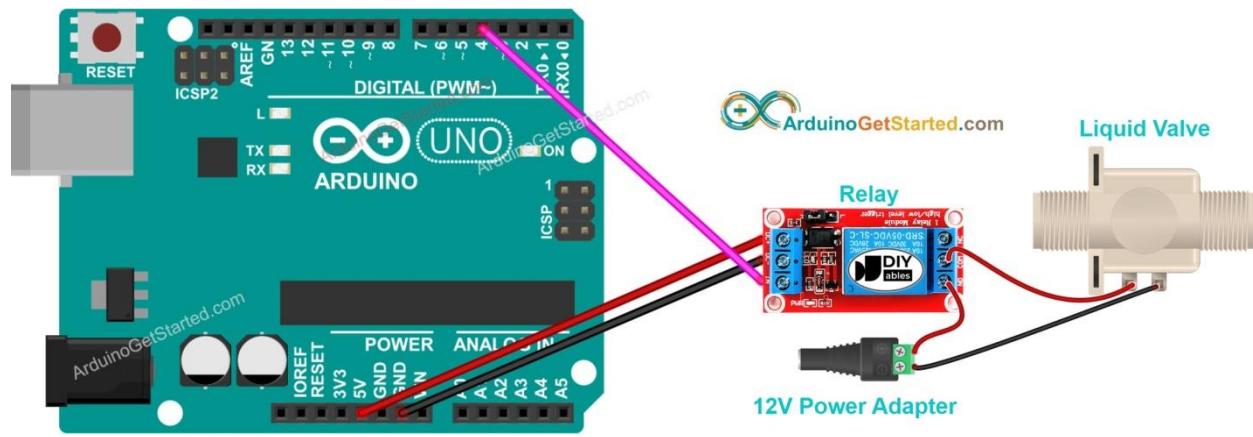
delay(1000);

}
```

WATER SENSOR LIQUID VALVE



 ArduinoGetStarted.com



arduino-water-liquid-valve

// constants won't change

```
const int RELAY_PIN = 4; // the Arduino pin D4, which
connects to the IN pin of relay
```

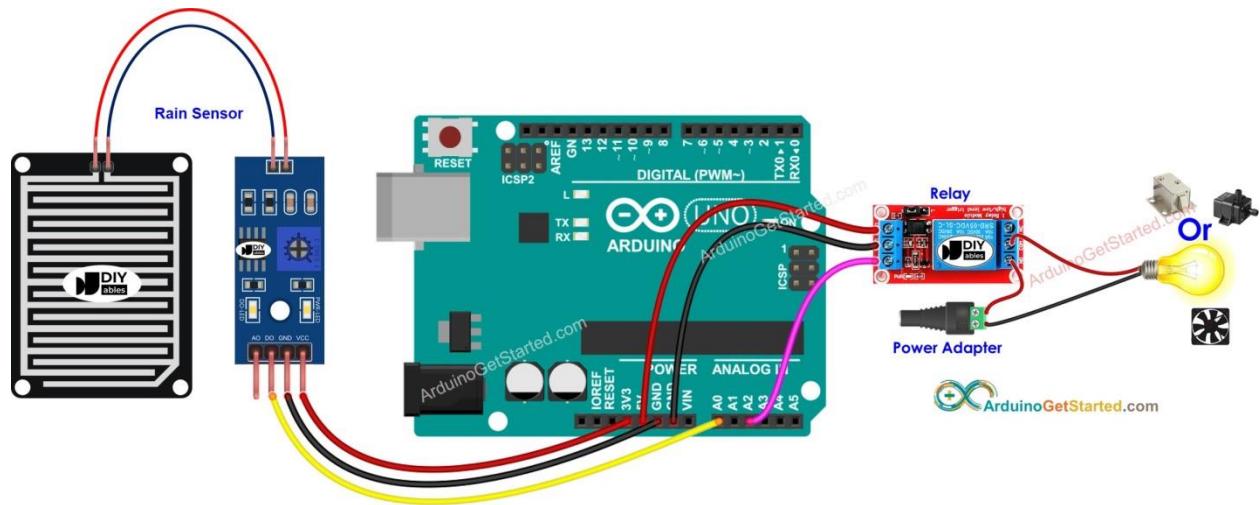
```
// the setup function runs once when you press reset or  
power the board
```

```
void setup() {  
    // initialize digital pin D4 as an output.  
    pinMode(RELAY_PIN, OUTPUT);  
}
```

```
// the loop function runs over and over again forever
```

```
void loop() {  
    digitalWrite(RELAY_PIN, HIGH); // open valve 5  
    seconds  
    delay(5000);  
  
    digitalWrite(RELAY_PIN, LOW); // close valve 5  
    seconds  
    delay(5000);  
}
```

RAIN SENSOR RELAY



```
#define RAIN_SENSOR_PIN A0 // Arduino pin
```

connected to the OUTPUT pin of rain sensor

```
#define RELAY_PIN      A2 // Arduino pin connected to  
the IN pin of relay
```

```
int rain_state      = LOW; // current state of rain sensor's  
pin
```

```
int prev_rain_state = LOW; // previous state of rain  
sensor's pin
```

```
void setup() {
```

```
    Serial.begin(9600);           // initialize serial
```

```
    pinMode(RAIN_SENSOR_PIN, INPUT); // set arduino pin  
to input mode
```

```
pinMode(RELAY_PIN, OUTPUT);      // set arduino pin to  
output mode  
  
}  
  
void loop() {  
  
    prev_rain_state = rain_state;      // store old state  
  
    rain_state = digitalRead(RAIN_SENSOR_PIN); // read  
new state  
  
    if (prev_rain_state == LOW && rain_state == HIGH) { //  
pin state change: LOW -> HIGH  
        Serial.println("Rain detected!");  
  
        digitalWrite(RELAY_PIN, HIGH); // turn on  
    }  
  
    else  
  
        if (prev_rain_state == HIGH && rain_state == LOW) { //  
pin state change: HIGH -> LOW  
            Serial.println("Rain stopped!");
```

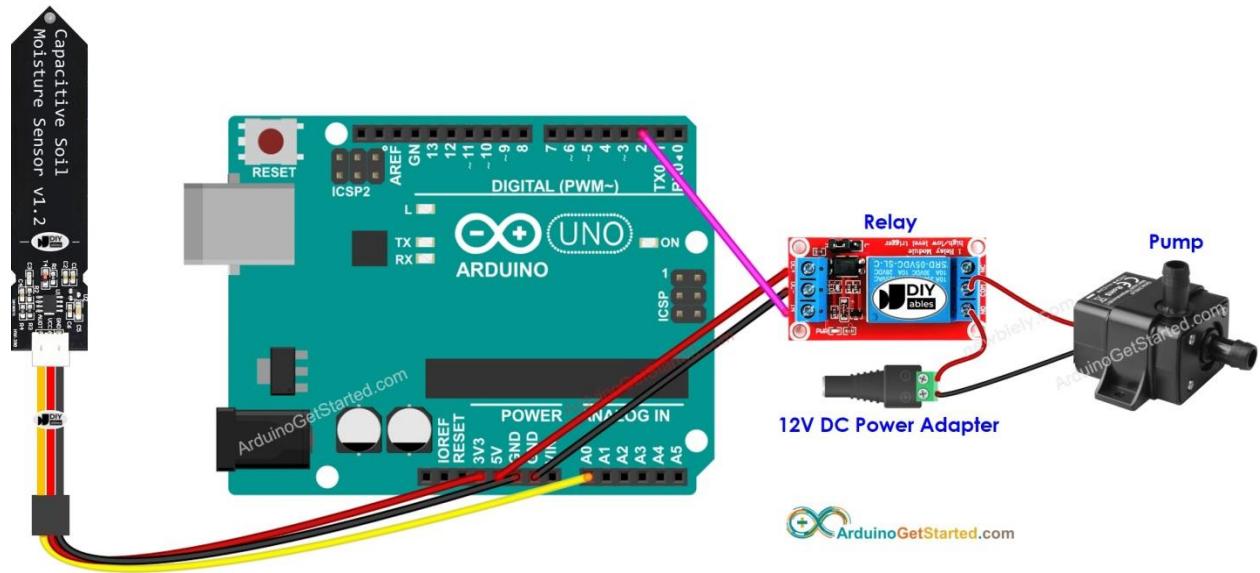
```
digitalWrite(RELAY_PIN, LOW); // turn off
```

```
}
```

```
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

SOIL MOISTURE RELAY PUMP



```
#define RELAY_PIN 2 // Arduino pin that controls the
pump via relay
```

```
#define MOISTURE_PIN A0 // Arduino pin that connects
to AOUT pin of moisture sensor
```

```
#define THRESHOLD 530 // => CHANGE YOUR
THRESHOLD HERE
```

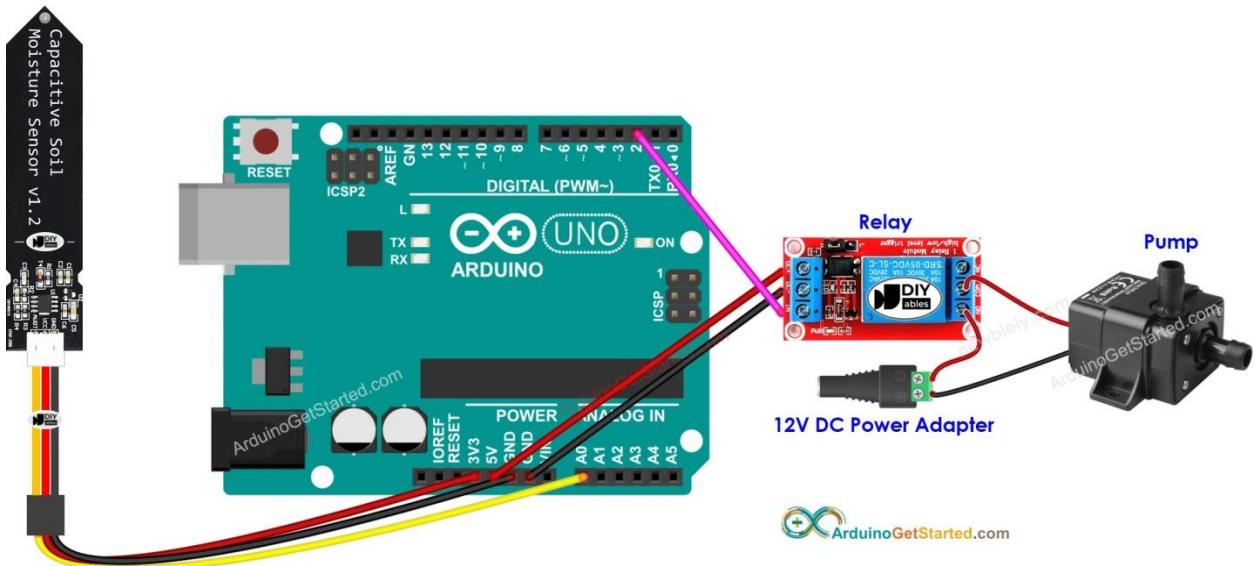
```
void setup() {
  Serial.begin(9600);
  pinMode(RELAY_PIN, OUTPUT);
}
```

```
void loop() {  
  
    int value = analogRead(MOISTURE_PIN); // read the  
    analog value from sensor  
  
    if (value > THRESHOLD) {  
  
        Serial.print("The soil moisture is DRY => activate  
pump");  
  
        digitalWrite(RELAY_PIN, HIGH);  
  
    } else {  
  
        Serial.print("The soil moisture is WET => deactivate the  
pump");  
  
        digitalWrite(RELAY_PIN, LOW);  
  
    }  
  
    Serial.print(" (");  
  
    Serial.print(value);  
  
    Serial.println(")");
```

```
delay(1000);  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

WATER SENSOR BASED AUTOMATIC IRRIGATION SYSTEM



```
#define RELAY_PIN 2 // Arduino pin that connects to  
relay
```

```
#define MOISTURE_PIN A0 // Arduino pin that connects  
to AOUT pin of moisture sensor
```

```
#define THRESHOLD 530 // CHANGE YOUR THRESHOLD  
HERE
```

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

```
pinMode(RELAY_PIN, OUTPUT);

}

void loop() {

    int value = analogRead(MOISTURE_PIN); // read the
analog value from sensor

    if (value > THRESHOLD) {

        Serial.print("The soil is DRY => turn pump ON");

        digitalWrite(RELAY_PIN, HIGH);

    } else {

        Serial.print("The soil is WET => turn pump OFF");

        digitalWrite(RELAY_PIN, LOW);

    }

    Serial.print(" (");

    Serial.print(value);

}
```

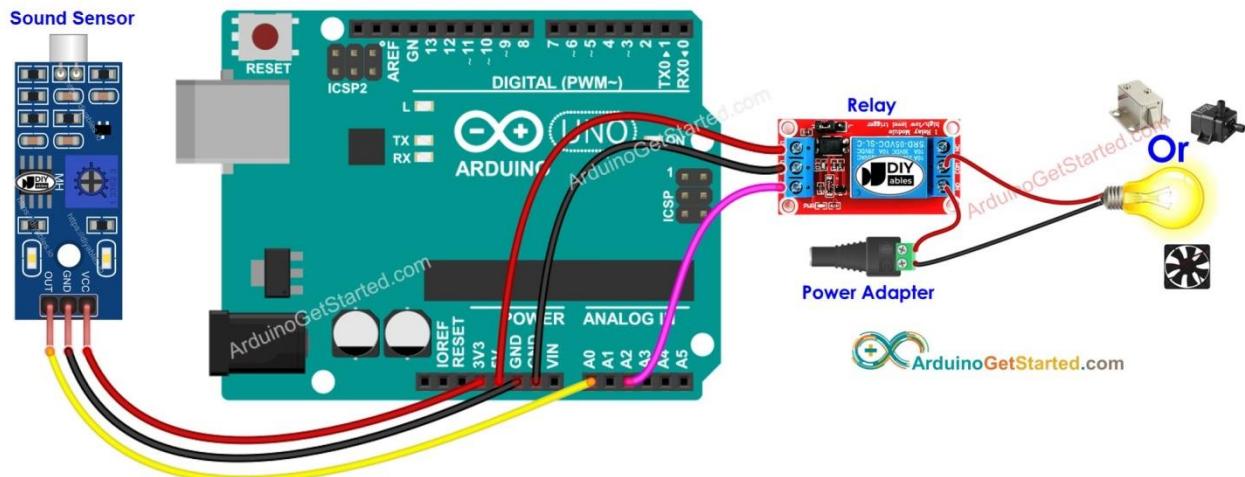
```
Serial.println(")");
```

```
delay(500);
```

```
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

SOUND SENSOR RELAY



```
#define SENSOR_PIN A0 // Arduino pin connected to
sound sensor's pin
```

```
#define RELAY_PIN A2 // Arduino pin connected to
LED's pin
```

```
int lastSoundState; // the previous state of sound
sensor
```

```
int currentSoundState; // the current state of sound
sensor
```

```
int relayState = LOW; // the current state of relay
```

```
void setup() {
```

```
Serial.begin(9600);      // initialize serial

pinMode(SENSOR_PIN, INPUT); // set arduino pin to
input mode

pinMode(RELAY_PIN, OUTPUT); // set arduino pin to
output mode

currentSoundState = digitalRead(SENSOR_PIN);

}

void loop() {

lastSoundState = currentSoundState;      // save the
last state

currentSoundState = digitalRead(SENSOR_PIN); // read
new state

if (lastSoundState == HIGH && currentSoundState ==
LOW) { // state change: HIGH -> LOW

Serial.println("The sound has been detected");
```

```
// toggle state of relay
```

```
relayState = !relayState;
```

```
// control relay according to the toggle relay state
```

```
digitalWrite(RELAY_PIN, relayState);
```

```
}
```

```
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

SOUND ACTIVATED RELAY FOR A PERIOD OF TIME

```
#define SENSOR_PIN A0 // Arduino pin connected to  
sound sensor's pin
```

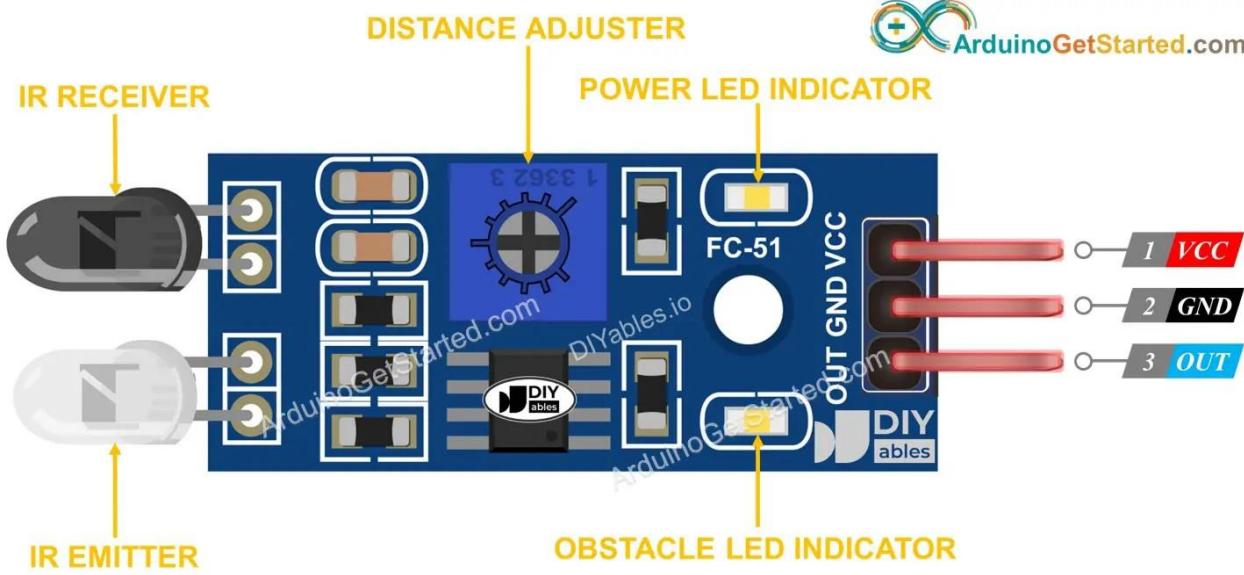
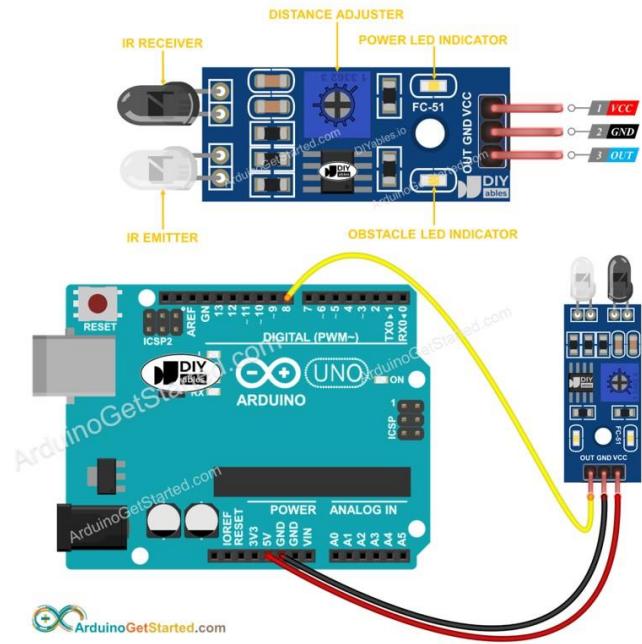
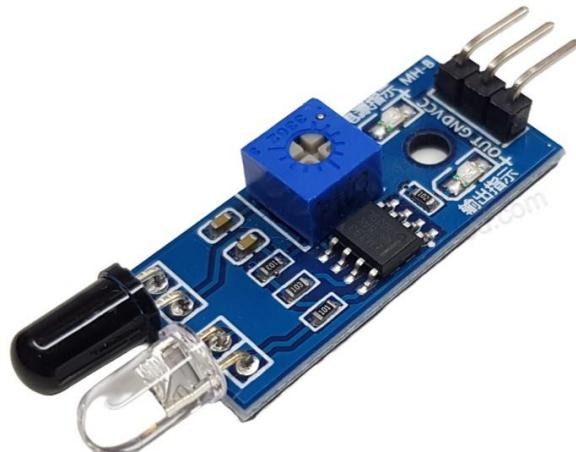
```
#define RELAY_PIN A2 // Arduino pin connected to  
the relay's pin
```

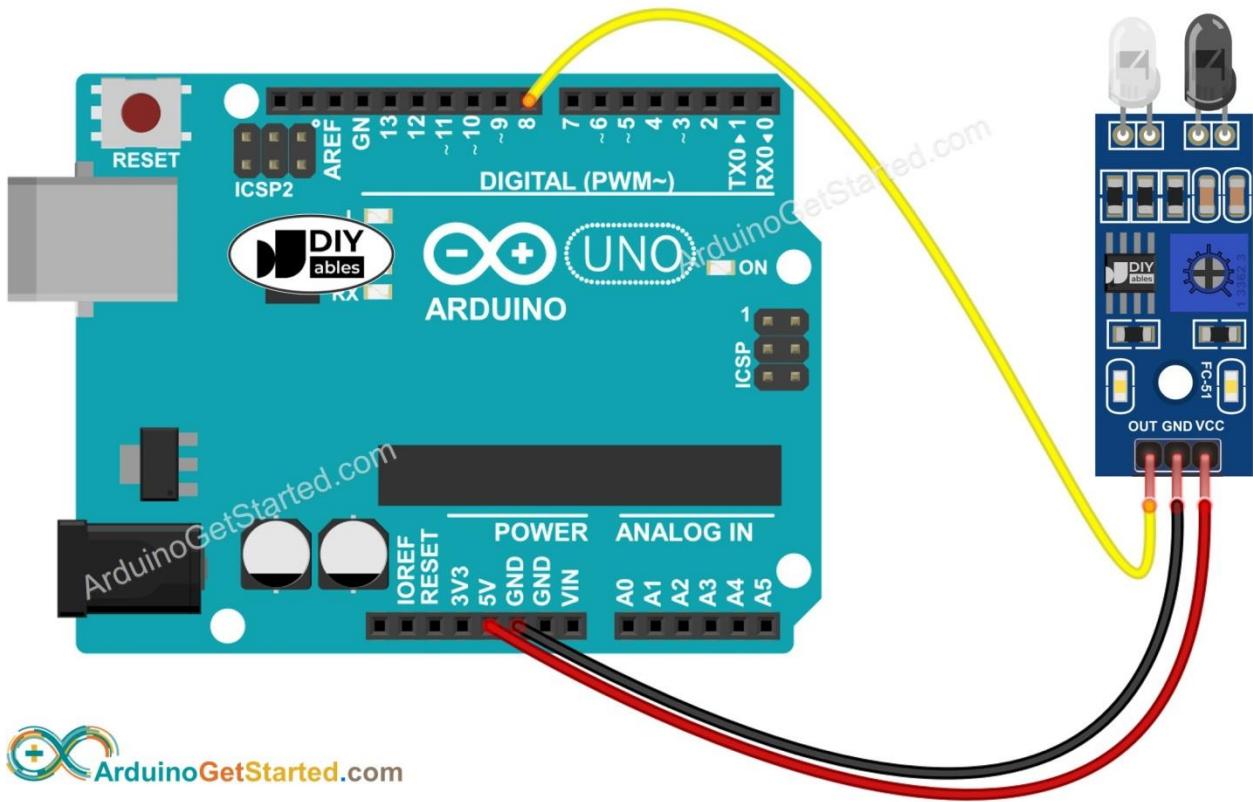
```
#define TIME_PERIOD 5000 // in milliseconds
```

```
int lastSoundState; // the previous state of sound  
sensor  
  
int currentSoundState; // the current state of sound  
sensor  
  
void setup() {  
  
    Serial.begin(9600); // initialize serial  
  
    pinMode(SENSOR_PIN, INPUT); // set arduino pin to  
input mode  
  
    pinMode(RELAY_PIN, OUTPUT); // set arduino pin to  
output mode  
  
    currentSoundState = digitalRead(SENSOR_PIN);  
}  
  
void loop() {  
  
    lastSoundState = currentSoundState; // save the  
last state
```

```
currentSoundState = digitalRead(SENSOR_PIN); //  
read new state  
  
if (lastSoundState == HIGH && currentSoundState ==  
LOW) { // state change: HIGH -> LOW  
  
    Serial.println("The sound has been detected");  
  
    // turn on relay  
  
    digitalWrite(RELAY_PIN, HIGH);  
  
    delay(TIME_PERIOD);  
  
    // turn off relay  
  
    digitalWrite(RELAY_PIN, LOW);  
}  
}
```

INFRARED OBSTACLE AVOIDENCE SENSOR





ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
// Arduino's pin connected to OUT pin of IR obstacle
```

```
avoidance sensor
```

```
const int SENSOR_PIN = 8;
```

```
int lastState = HIGH; // the previous state from the input
```

```
pin
```

```
int currentState; // the current reading from the input
```

```
pin
```

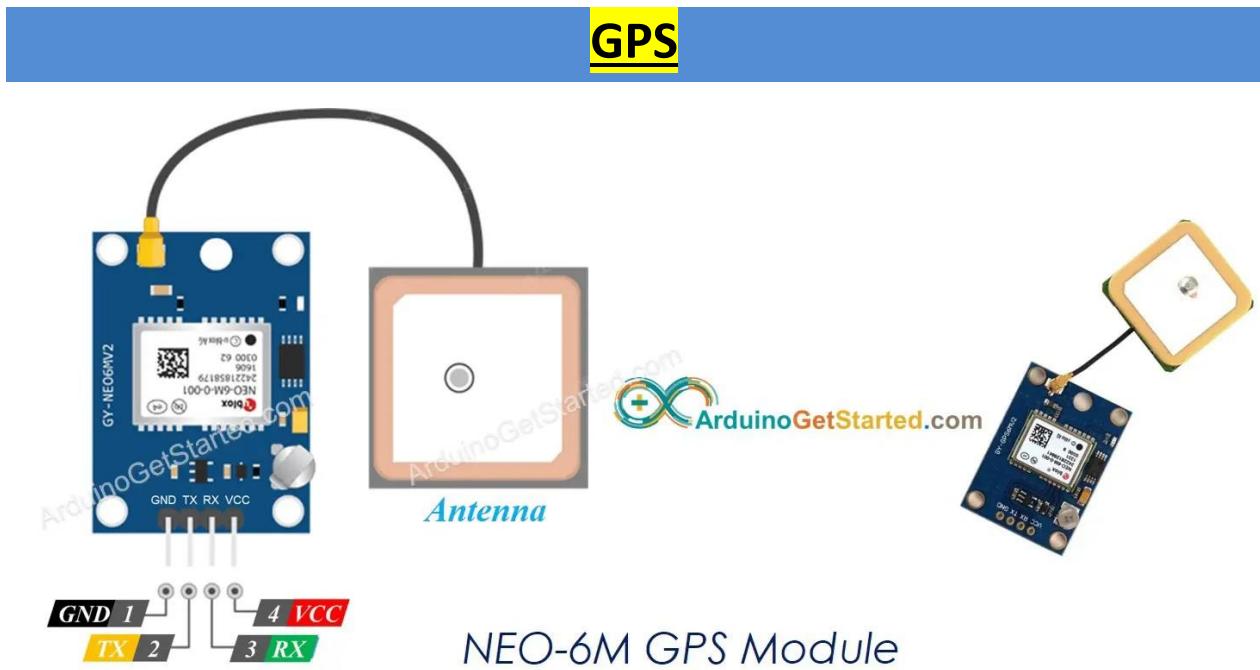
```
void setup() {  
    // initialize serial communication at 9600 bits per  
    // second:  
    Serial.begin(9600);  
  
    // initialize the Arduino's pin as an input  
    pinMode(SENSOR_PIN, INPUT);  
}  
  
void loop() {  
    // read the state of the the input pin:  
    currentState = digitalRead(SENSOR_PIN);  
  
    if (lastState == HIGH && currentState == LOW)  
        Serial.println("The obstacle is detected");  
    else if (lastState == LOW && currentState == HIGH)  
        Serial.println("The obstacle is cleared");  
}
```

```
delay(50);  
// save the the last state  
lastState = currentState;  
}
```

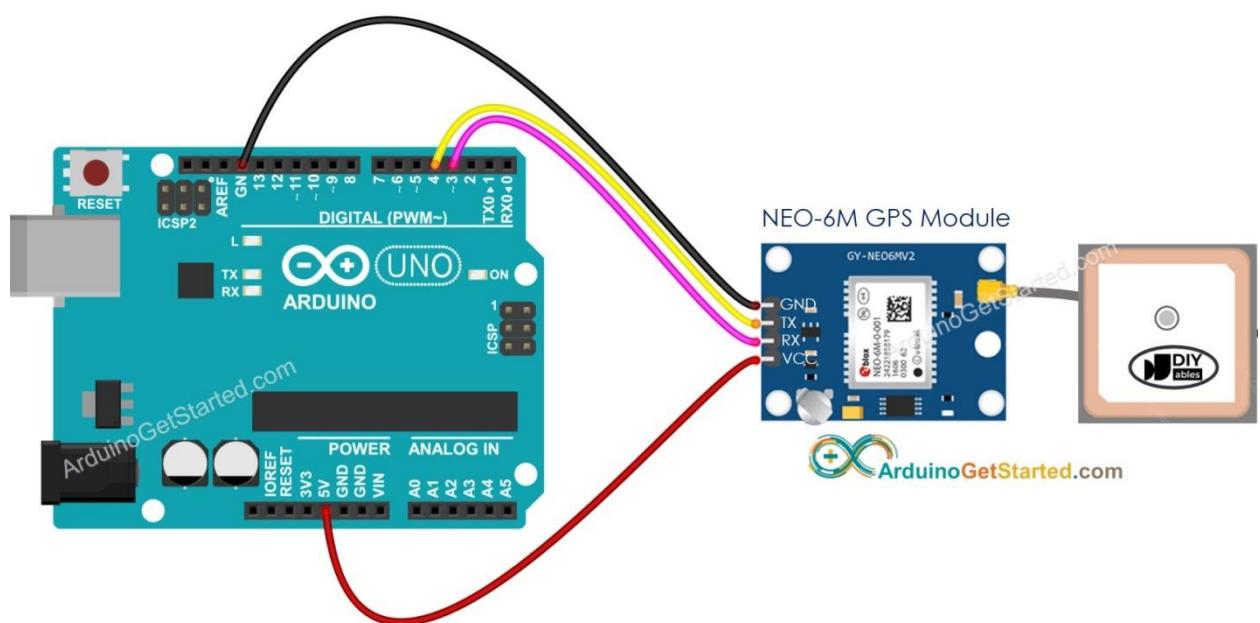
CHECK TO OBSTACLE IS PRESENT OR NOT

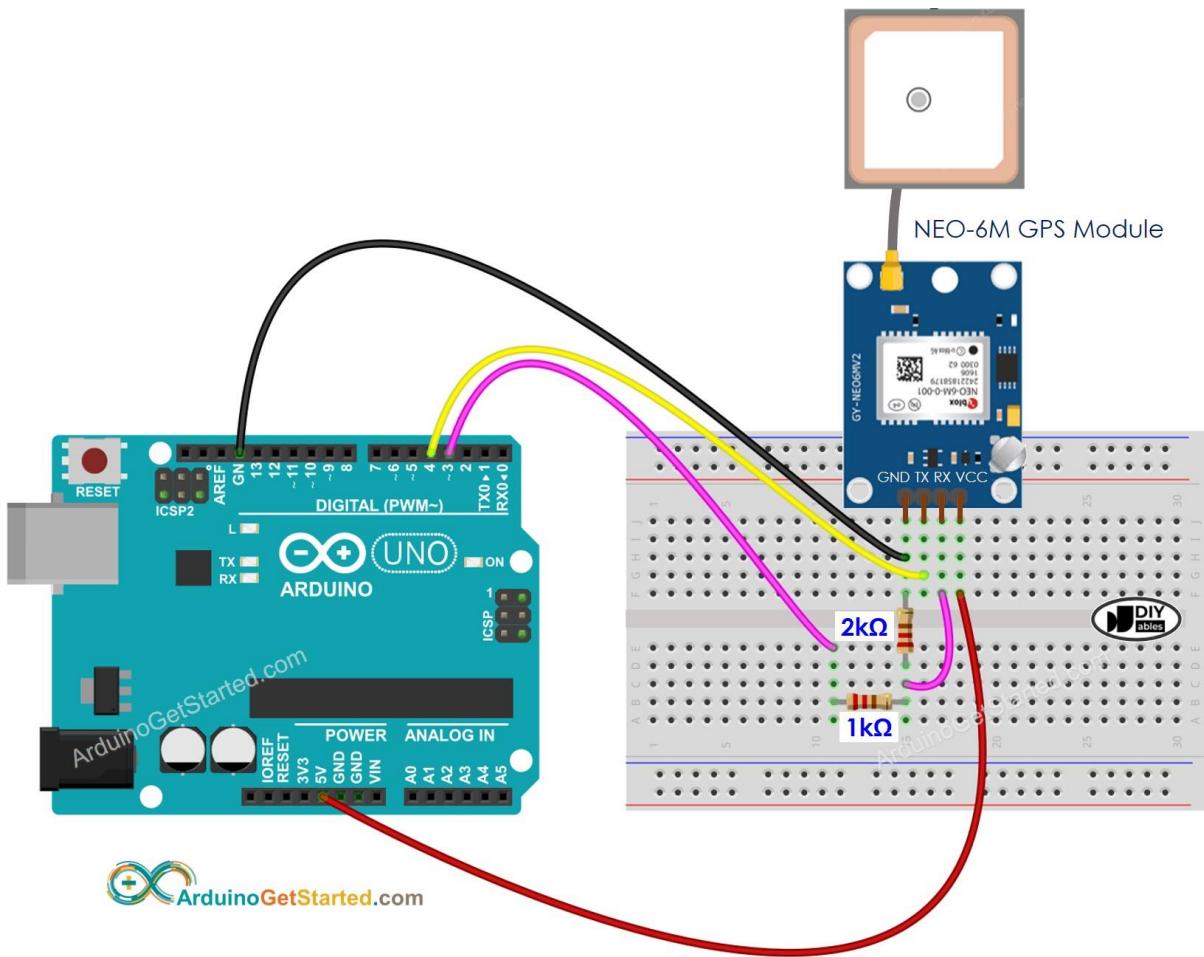
```
// Arduino's pin connected to OUT pin of IR obstacle  
avoidance sensor  
const int SENSOR_PIN = 8;  
  
void setup() {  
    // initialize serial communication at 9600 bits per  
    second:  
    Serial.begin(9600);
```

```
// initialize the Arduino's pin as an input  
pinMode(SENSOR_PIN, INPUT);  
  
}  
  
void loop() {  
    // read the state of the the input pin:  
    int state = digitalRead(SENSOR_PIN);  
  
    if (state == LOW)  
        Serial.println("The obstacle is present");  
    else  
        Serial.println("The obstacle is NOT present");  
  
    delay(100);  
}
```



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK





READING GPS CO ORDINATE SPEED DATE TIME

```
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
```

```
#define RX_PIN 4 // Arduino Pin connected to the TX of  
the GPS module
```

```
#define TX_PIN 3 // Arduino Pin connected to the RX of  
the GPS module
```

```
TinyGPSPlus gps; // the TinyGPS++ object
```

```
SoftwareSerial gpsSerial(RX_PIN, TX_PIN); // the serial  
interface to the GPS module
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
    gpsSerial.begin(9600); // Default baud of NEO-6M GPS  
module is 9600
```

```
    Serial.println(F("Arduino - GPS module"));
```

```
}
```

```
void loop() {
```

```
if (gpsSerial.available() > 0) {  
    if (gps.encode(gpsSerial.read())) {  
        if (gps.location.isValid()) {  
            Serial.print(F("- latitude: "));  
            Serial.println(gps.location.lat());  
  
            Serial.print(F("- longitude: "));  
            Serial.println(gps.location.lng());  
  
            Serial.print(F("- altitude: "));  
            if (gps.altitude.isValid())  
                Serial.println(gps.altitude.meters());  
            else  
                Serial.println(F("INVALID"));  
        } else {  
            Serial.println(F("- location: INVALID"));  
        }  
    }  
}
```

```
Serial.print(F("- speed: "));  
if (gps.speed.isValid()) {  
    Serial.print(gps.speed.kmph());  
    Serial.println(F(" km/h"));  
} else {  
    Serial.println(F("INVALID"));  
}  
  
Serial.print(F("- GPS date&time: "));  
if (gps.date.isValid() && gps.time.isValid()) {  
    Serial.print(gps.date.year());  
    Serial.print(F("-"));  
    Serial.print(gps.date.month());  
    Serial.print(F("-"));  
    Serial.print(gps.date.day());  
    Serial.print(F(" "));
```

```
Serial.print(gps.time.hour());  
Serial.print(F(":"));  
Serial.print(gps.time.minute());  
Serial.print(F(":"));  
Serial.println(gps.time.second());  
} else {  
    Serial.println(F("INVALID"));  
}  
  
Serial.println();  
}  
}  
  
if (millis() > 5000 && gps.charsProcessed() < 10)  
    Serial.println(F("No GPS data received: check wiring"));  
}
```

CALCULATING DISTANCE FROM CURRENT LOCATION TO PREDEFINED LOCATION

```
#include <TinyGPS++.h>

#include <SoftwareSerial.h>

#define RX_PIN 4 // Arduino Pin connected to the TX of
the GPS module

#define TX_PIN 3 // Arduino Pin connected to the RX of
the GPS module

TinyGPSPlus gps; // the TinyGPS++ object

SoftwareSerial gpsSerial(RX_PIN, TX_PIN); // the serial
interface to the GPS module

const double LONDON_LAT = 51.508131;

const double LONDON_LON = -0.128002;
```

```
void setup() {  
    Serial.begin(9600);  
  
    gpsSerial.begin(9600); // Default baud of NEO-6M GPS  
    module is 9600  
  
    Serial.println(F("Arduino - GPS module"));  
}  
  
void loop() {  
    if (gpsSerial.available() > 0) {  
        if (gps.encode(gpsSerial.read())) {  
            if (gps.location.isValid()) {  
                double latitude = gps.location.lat();  
  
                double longitude = gps.location.lng();  
  
                unsigned long distanceKm =  
TinyGPSPlus::distanceBetween(latitude, longitude,  
LONDON_LAT, LONDON_LON) / 1000;  
    }
```

```
Serial.print(F("- latitude: "));  
Serial.println(latitude);  
  
Serial.print(F("- longitude: "));  
Serial.println(longitude);  
  
Serial.print(F("- distance to London: "));  
Serial.println(distanceKm);  
} else {  
    Serial.println(F("- location: INVALID"));  
}  
  
Serial.println();  
}  
}  
  
if (millis() > 5000 && gps.charsProcessed() < 10)
```

```
Serial.println(F("No GPS data received: check wiring"));
```

```
}
```

IR REMOTE CONTROL

IR CONTROLLER



OR



IR RECEIVER



OR



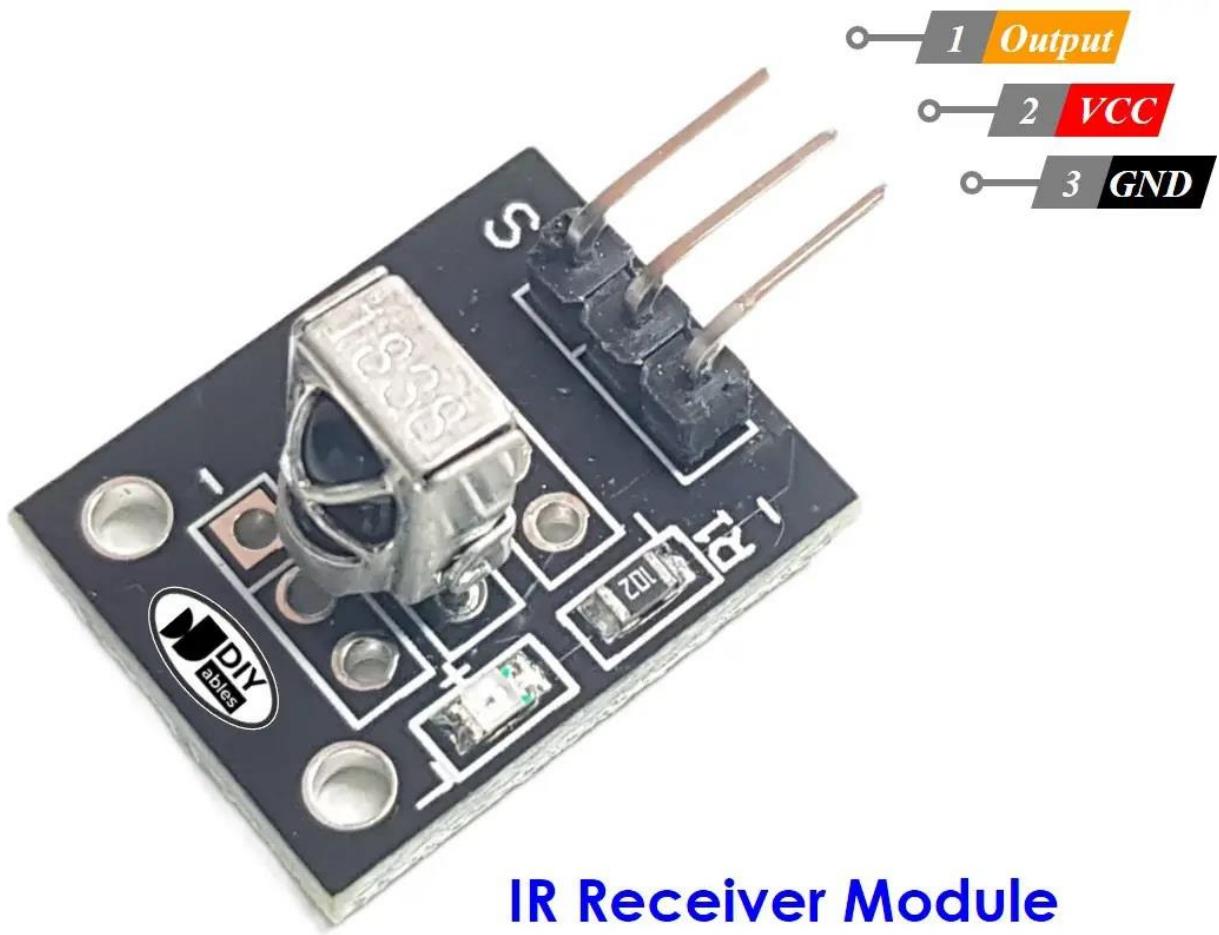
ArduinoGetStarted.com



17-key IR Controller



21-key IR Controller



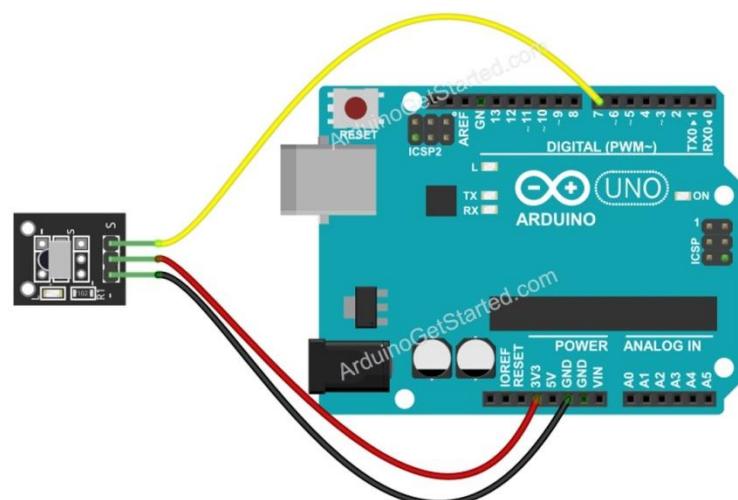
IR Receiver Module

**IR Receiver Sensor****Wiring Adapter**

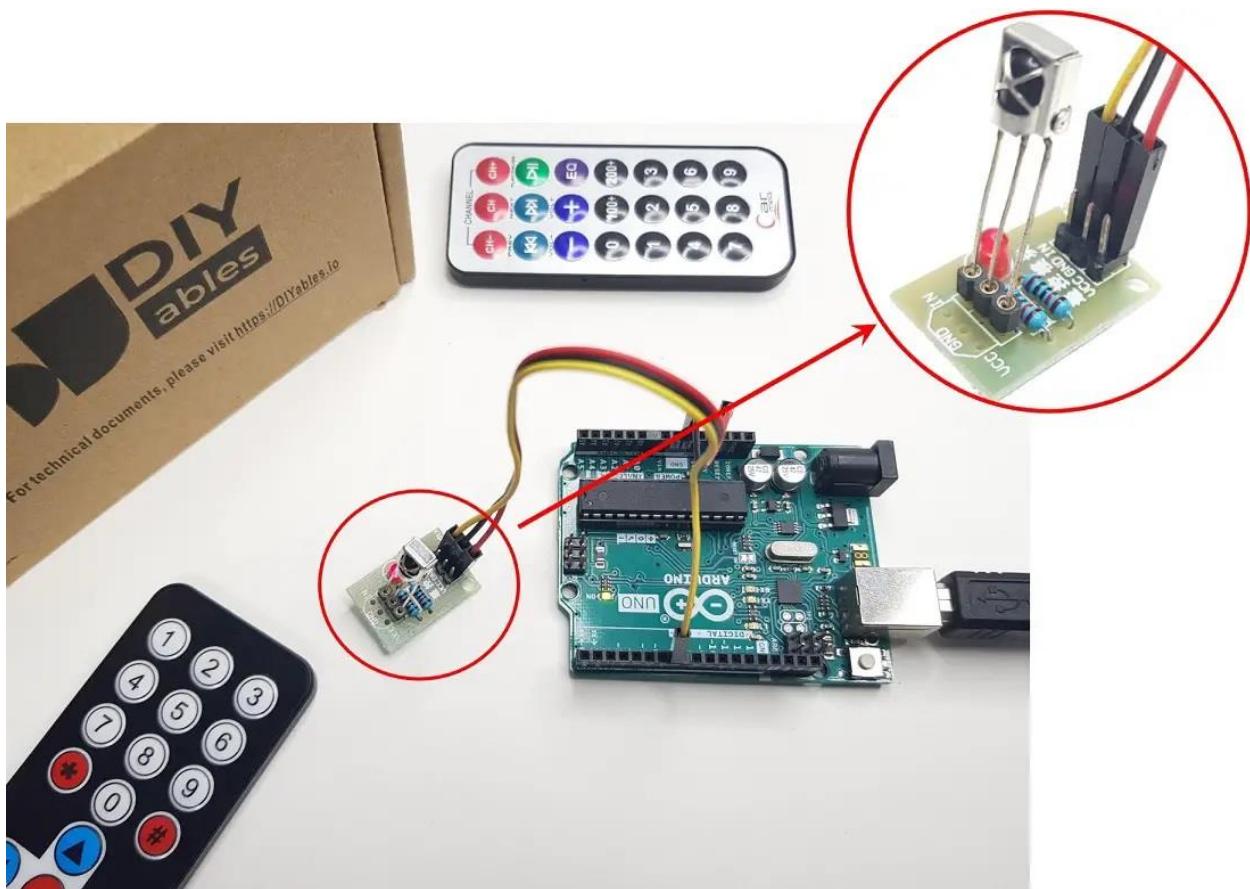
OR



ArduinoGetStarted.com







```
#include <DIYables_IRcontroller.h> //  
DIYables_IRcontroller library  
  
#define IR_RECEIVER_PIN 7 // The Arduino pin  
connected to IR controller  
  
DIYables_IRcontroller_17 irController(IR_RECEIVER_PIN,  
200); // debounce time is 200ms  
  
void setup() {
```

```
Serial.begin(9600);

irController.begin();

}

void loop() {

    Key17 key = irController.getKey();

    if (key != Key17::NONE) {

        switch (key) {

            case Key17::KEY_1:

                Serial.println("1");

                // TODO: YOUR CONTROL

                break;

            case Key17::KEY_2:

                Serial.println("2");

                // TODO: YOUR CONTROL

                break;
        }
    }
}
```

```
case Key17::KEY_3:  
    Serial.println("3");  
    // TODO: YOUR CONTROL  
    break;  
  
case Key17::KEY_4:  
    Serial.println("4");  
    // TODO: YOUR CONTROL  
    break;  
  
case Key17::KEY_5:  
    Serial.println("5");  
    // TODO: YOUR CONTROL  
    break;  
  
case Key17::KEY_6:
```

```
Serial.println("6");

// TODO: YOUR CONTROL

break;

case Key17::KEY_7:

Serial.println("7");

// TODO: YOUR CONTROL

break;

case Key17::KEY_8:

Serial.println("8");

// TODO: YOUR CONTROL

break;

case Key17::KEY_9:

Serial.println("9");

// TODO: YOUR CONTROL
```

```
break;
```

```
case Key17::KEY_STAR:
```

```
    Serial.println("*");
```

```
    // TODO: YOUR CONTROL
```

```
    break;
```

```
case Key17::KEY_0:
```

```
    Serial.println("0");
```

```
    // TODO: YOUR CONTROL
```

```
    break;
```

```
case Key17::KEY_SHARP:
```

```
    Serial.println("#");
```

```
    // TODO: YOUR CONTROL
```

```
    break;
```

```
case Key17::KEY_UP:  
    Serial.println("UP");  
    // TODO: YOUR CONTROL  
    break;  
  
case Key17::KEY_DOWN:  
    Serial.println("DOWN");  
    // TODO: YOUR CONTROL  
    break;  
  
case Key17::KEY_LEFT:  
    Serial.println("LEFT");  
    // TODO: YOUR CONTROL  
    break;  
  
case Key17::KEY_RIGHT:  
    Serial.println("RIGHT");
```

```
// TODO: YOUR CONTROL
```

```
break;
```

```
case Key17::KEY_OK :
```

```
Serial.println("OK");
```

```
// TODO: YOUR CONTROL
```

```
break;
```

```
default:
```

```
Serial.println("WARNING: undefined key:");
```

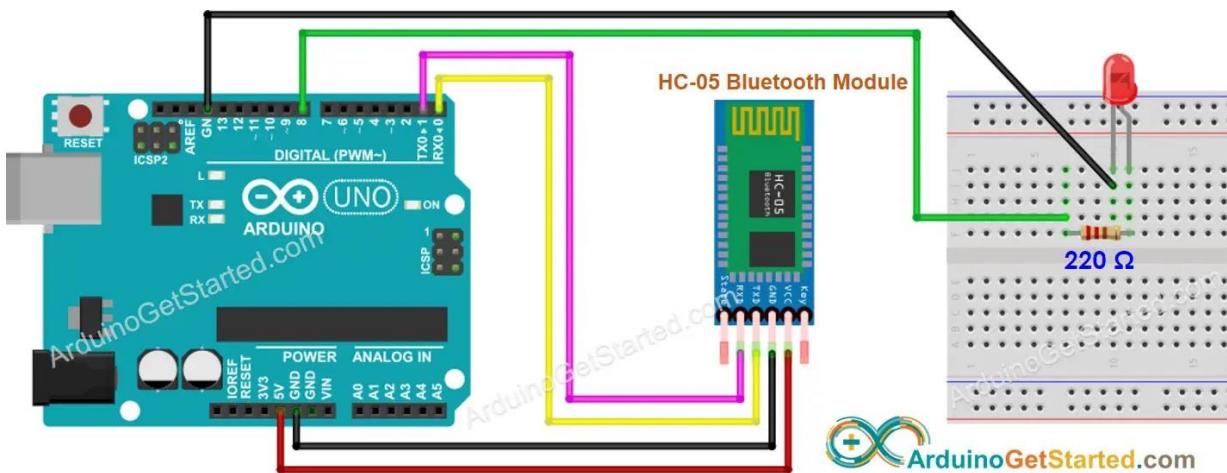
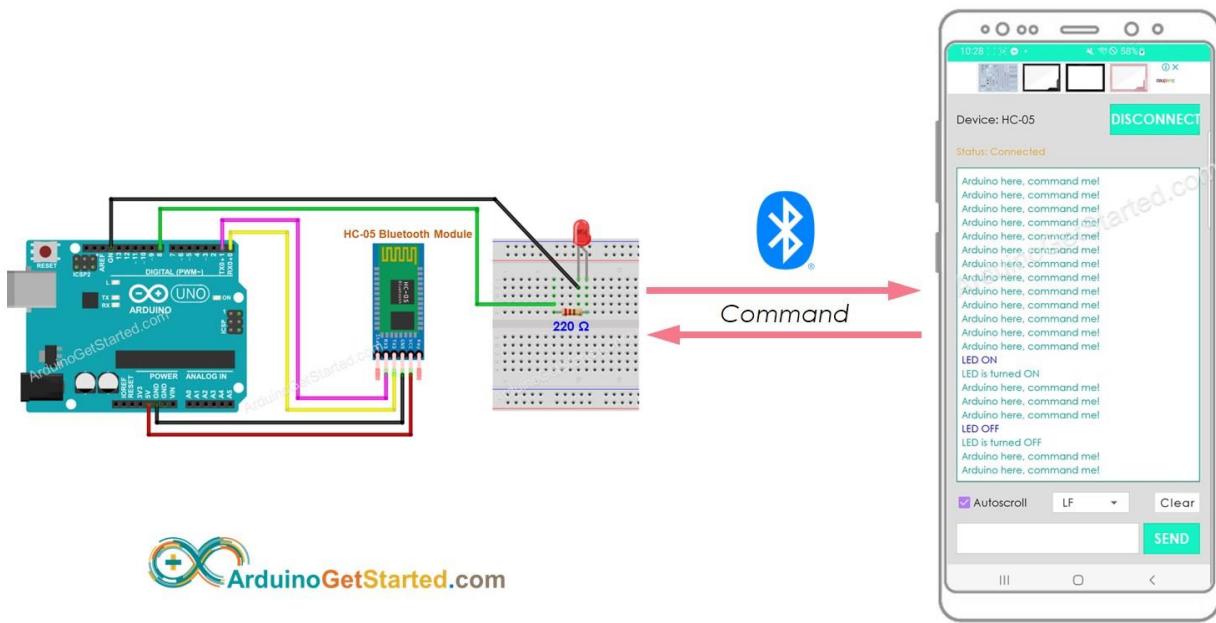
```
break;
```

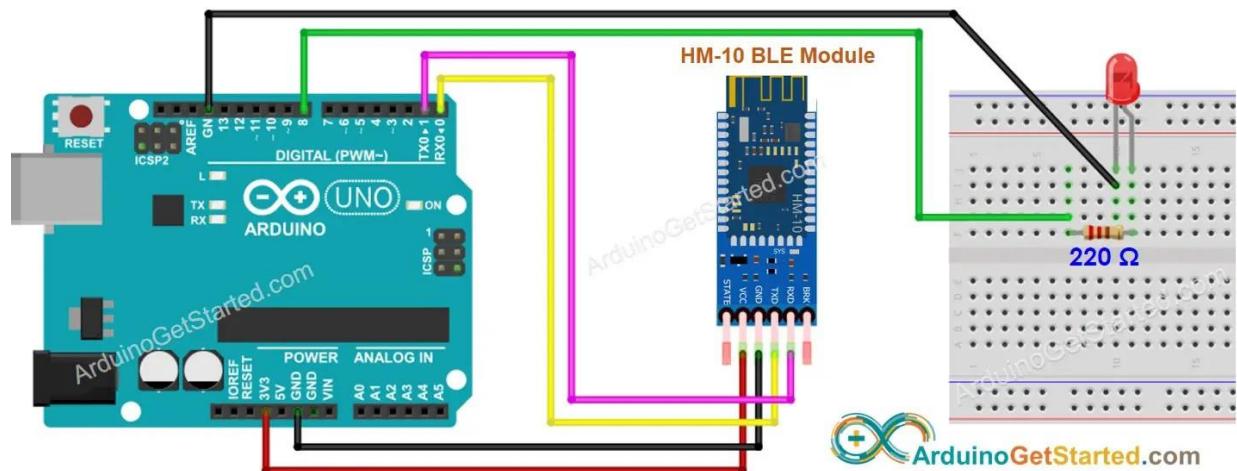
```
}
```

```
}
```

```
}
```

CONTROL LED VIA BLUETOOTH





// NOTE: change the Serial to other Serial/Software Serial
if you connects Bluetooth module to other pins

```
#define LED_PIN 8

void setup() {
    Serial.begin(9600);
    pinMode(LED_PIN, OUTPUT); // set the digital pin as
    output;
}
```

```
void loop() {  
  
    if (Serial.available()) { // if there is data comming  
  
        String command = Serial.readStringUntil('\n'); // read  
        string until meet newline character  
  
        if (command == "OFF") {  
  
            digitalWrite(LED_PIN, LOW); // turn off LED  
  
            Serial.println("LED is turned OFF"); // reports action  
            to smartphone app  
  
        } else if (command == "ON") {  
  
            digitalWrite(LED_PIN, HIGH); // turn on LED  
  
            Serial.println("LED is turned ON"); // reports action to  
            smartphone app  
        }  
    }  
}
```

Quick Steps

- Install [Bluetooth Serial Monitor App](#) on your smartphone
- Copy the above code and open with Arduino IDE and upload the code to Arduino
- Click **Upload** button on Arduino IDE to upload code to Arduino. If you are unable to upload code to your Arduino, try disconnecting the TX and RX pins from the Bluetooth module, uploading the code, and then reconnecting the RX/TX pins again.
- Open Bluetooth Serial Monitor App on your smartphone
- Select the Classic Bluetooth or BLE according to the module you used

SENSORS/ACTUATORS

- Arduino - Software Installation
- Arduino - Hardware Preparation
- Arduino - Hello World
- Arduino - Code Structure
- Arduino - Serial Monitor
- Arduino - Serial Plotter
- Arduino - LED - Blink
- Arduino - LED - Blink Without Delay
- Arduino - Blink multiple LED
- Arduino - LED - Fade
- Arduino - RGB LED
- Arduino - Traffic Light
- Arduino - Button
- Arduino - Button - Debounce
- Arduino - Button - Long Press Short Press
- Arduino multiple Button
- Arduino - Switch
- Arduino - Limit Switch
- Arduino - DIP Switch
- Arduino - Button - LED
- Arduino - Button - Relay
- Arduino - Button Toggle LED
- Arduino - Button Toggle Relay
- Arduino - Button - Piezo Buzzer
- Arduino - Button - Servo Motor

Pair the Bluetooth App with HC-05 Bluetooth module or HM-10 BLE module

HC-05
98:D3:C1:FE:1C:18

STOP SCAN

Viksit Bharat

Type "ON" or "OFF" and click Send button

Device: HMSoft

Status: Connected

Arduino here, command me!
ON
LED is turned ON
Arduino here, command me!
Arduino here, command me!

OFF

SEND

Autoscroll LF Clear

1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m

Bluetooth Serial Monitor

developed by [ArduinoGetStarted.com](#)

SCAN CLASSIC BLUETOOTH

SCAN BLE

Tutorial

Device: HMSoft

Status: Connected

Arduino here, command me!
ON
LED is turned ON
Arduino here, command me!
Arduino here, command me!

OFF

SEND

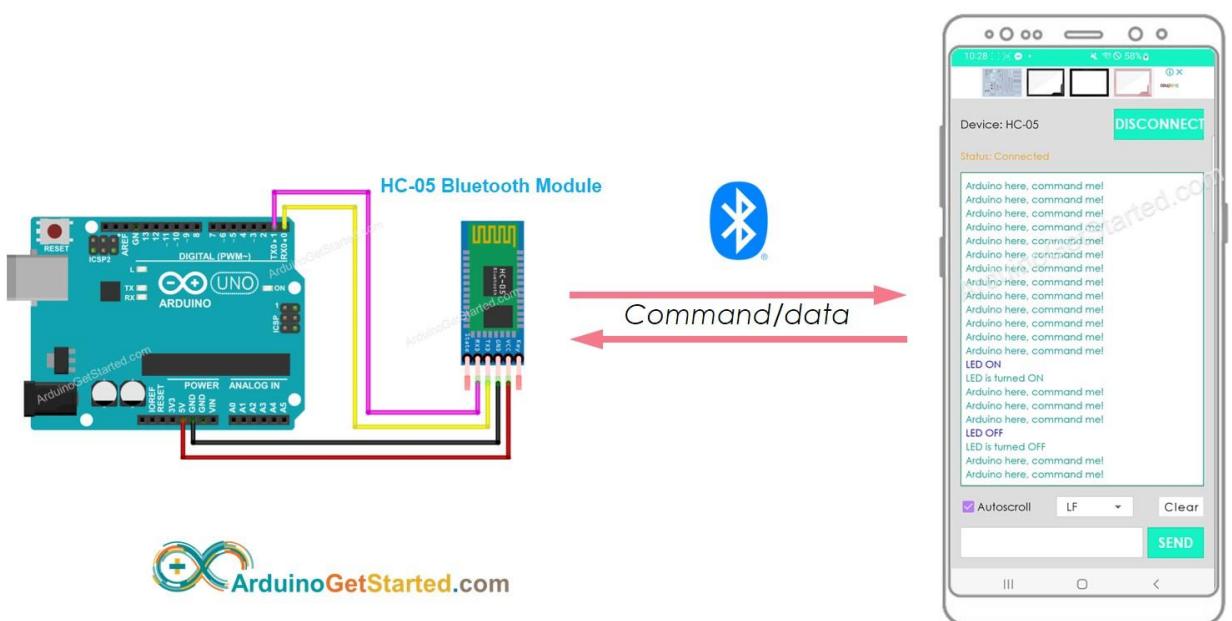
Autoscroll LF Clear

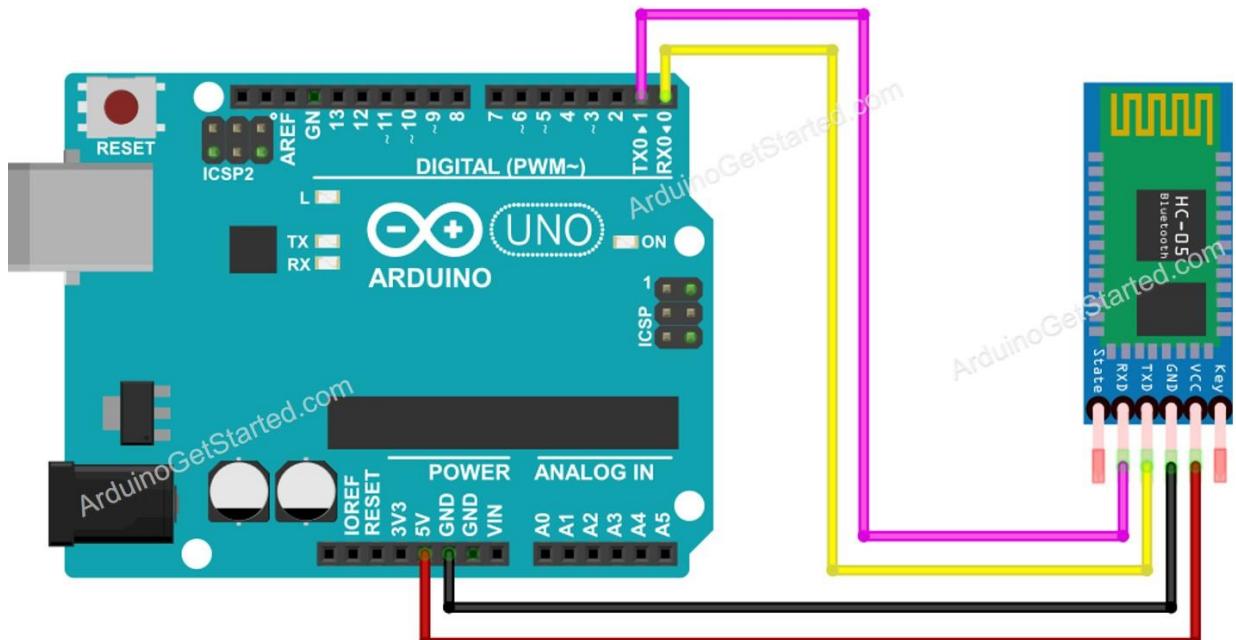
1 2 3 4 5 6 7 8 9 0
q w e r t y u i o p
a s d f g h j k l
z x c v b n m

Quick Steps

- Install [Bluetooth Serial Monitor App](#) on your smartphone
- Copy the above code and open with Arduino IDE and upload the code to Arduino
- Click **Upload** button on Arduino IDE to upload code to Arduino. If you are unable to upload code to your Arduino, try disconnecting the TX and RX pins from the Bluetooth module, uploading the code, and then reconnecting the RX/TX pins again.
- Open Bluetooth Serial Monitor App on your smartphone
- Select the Classic Bluetooth or BLE according to the module you used

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK





Arduino – HC-05 Bluetooth Module



```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.println("Arduino here, command me!");  
    delay(1000);  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

KEYPAD

Keypad 4x4



Keypad 3x4



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK





ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

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

```
const int ROW_NUM = 4; //four rows
```

```
const int COLUMN_NUM = 3; //three columns
```

```
char keys[ROW_NUM][COLUMN_NUM] = {
```

```
    {'1','2','3'},
```

```
    {'4','5','6'},
```

```
{'7','8','9'},  
{'*','0','#'}  
};
```

```
byte pin_rows[ROW_NUM] = {9, 8, 7, 6}; //connect to the  
row pinouts of the keypad
```

```
byte pin_column[COLUMN_NUM] = {5, 4, 3}; //connect  
to the column pinouts of the keypad
```

```
Keypad keypad = Keypad( makeKeymap(keys), pin_rows,  
pin_column, ROW_NUM, COLUMN_NUM );
```

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

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

```
if (key){  
    Serial.println(key);  
}  
}
```

KEYPAD WITH PASSWORD

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

```
#include <Keypad.h>  
  
const int ROW_NUM = 4; //four rows  
const int COLUMN_NUM = 3; //three columns  
  
char keys[ROW_NUM][COLUMN_NUM] = {  
    {'1','2','3'},  
    {'4','5','6'},
```

```
{'7','8','9'},  
{'*','0','#'}  
};
```

```
byte pin_rows[ROW_NUM] = {9, 8, 7, 6}; //connect to the  
row pinouts of the keypad
```

```
byte pin_column[COLUMN_NUM] = {5, 4, 3}; //connect  
to the column pinouts of the keypad
```

```
Keypad keypad = Keypad( makeKeymap(keys), pin_rows,  
pin_column, ROW_NUM, COLUMN_NUM );
```

```
const String password = "1234"; // change your password  
here
```

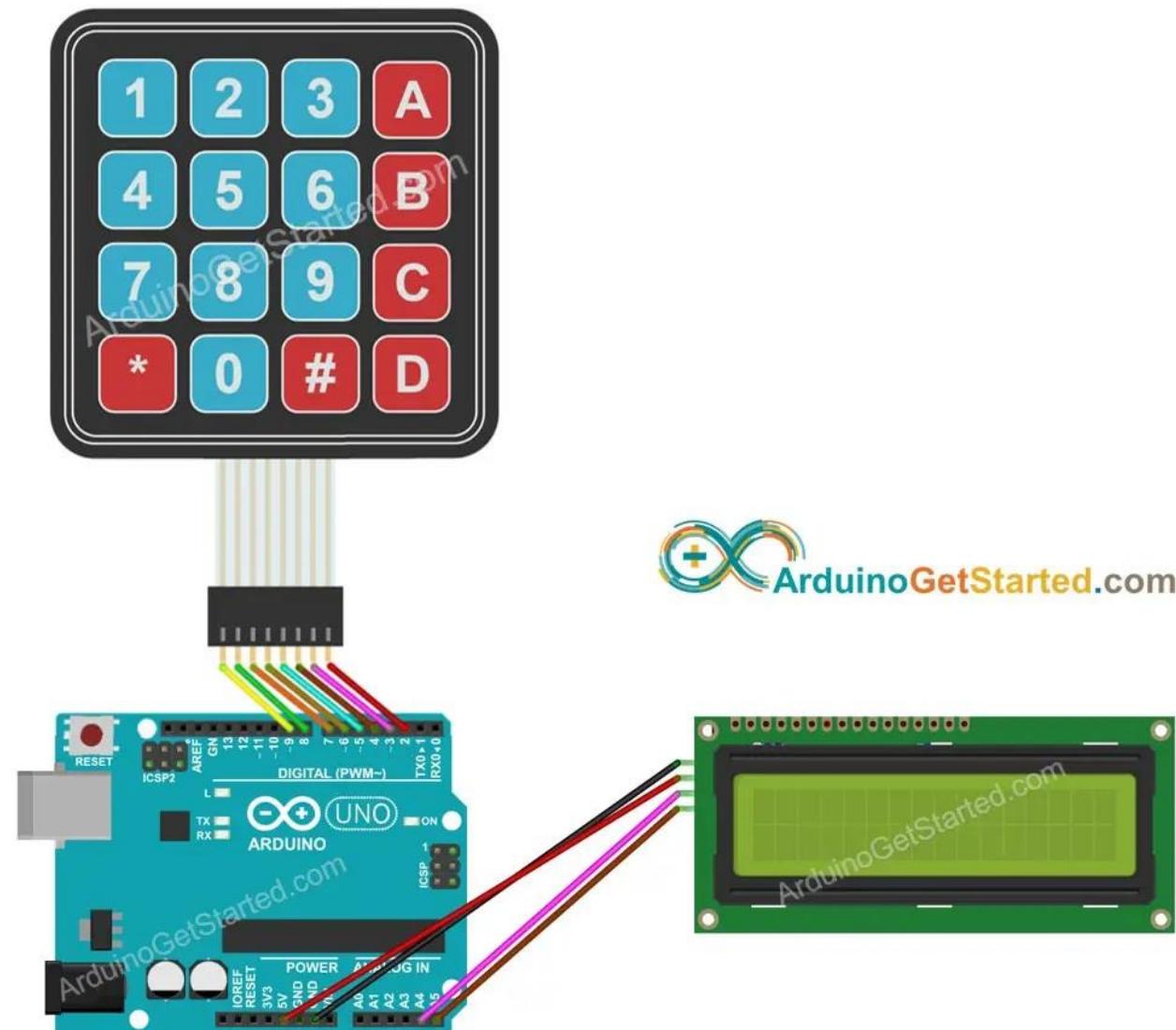
```
String input_password;
```

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

```
input_password.reserve(32); // maximum input  
characters is 33, change if needed  
  
}  
  
void loop(){  
    char key = keypad.getKey();  
  
    if (key){  
        Serial.println(key);  
  
        if(key == '*') {  
            input_password = ""; // clear input password  
        } else if(key == '#') {  
            if(password == input_password) {  
                Serial.println("password is correct");  
                // DO YOUR WORK HERE  
            }  
        }  
    }  
}
```

```
        } else {  
  
            Serial.println("password is incorrect, try again");  
        }  
  
        input_password = ""; // clear input password  
    } else {  
  
        input_password += key; // append new character to  
        input password string  
    }  
}  
}
```

KEYPAD LCD



```
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>

const int ROW_NUM = 4; // four rows
const int COLUMN_NUM = 4; // four columns
```

```
char keys[ROW_NUM][COLUMN_NUM] = {  
    {'1','2','3', 'A'},  
    {'4','5','6', 'B'},  
    {'7','8','9', 'C'},  
    {'*','0','#', 'D'}  
};
```

```
byte pin_rows[ROW_NUM] = {9, 8, 7, 6}; // connect to  
the row pinouts of the keypad
```

```
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2}; //  
connect to the column pinouts of the keypad
```

```
Keypad keypad = Keypad(makeKeymap(keys), pin_rows,  
pin_column, ROW_NUM, COLUMN_NUM );
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16  
column and 2 rows
```

```
int cursorColumn = 0;

void setup(){
    lcd.init(); // initialize the lcd
    lcd.backlight();
}

void loop(){
    char key = keypad.getKey();

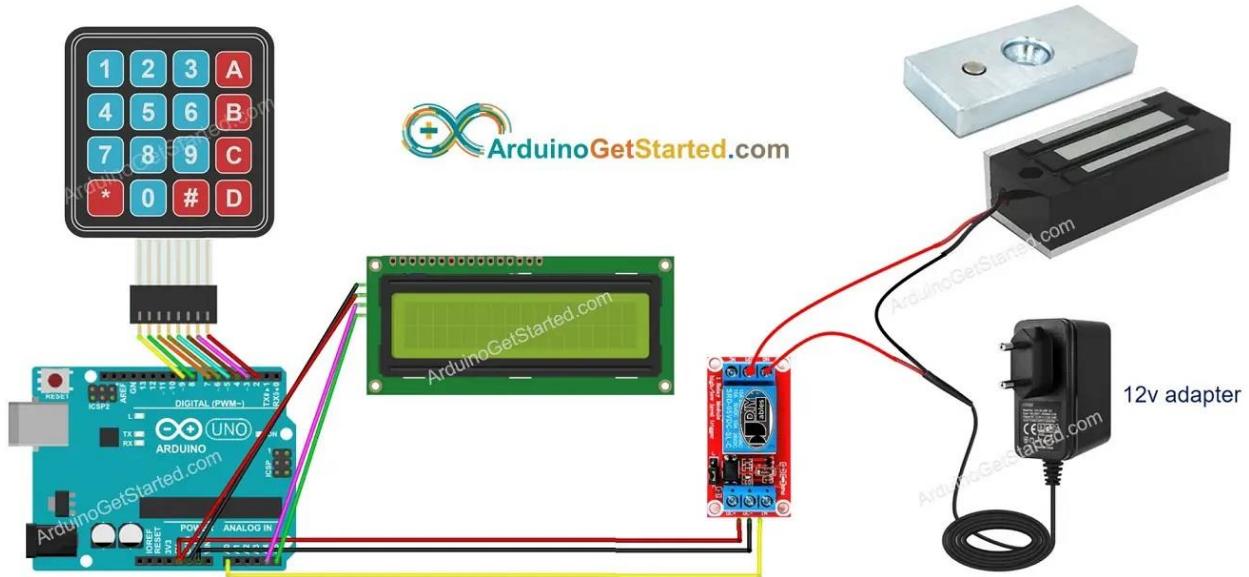
    if (key) {
        lcd.setCursor(cursorColumn, 0); // move cursor to
        (cursorColumn, 0)
        lcd.print(key); // print key at (cursorColumn,
        0)

        cursorColumn++; // move cursor to next
        position
    }
}
```

```
if(cursorColumn == 16) {      // if reaching limit, clear  
LCD  
  
lcd.clear();  
  
cursorColumn = 0;  
  
}  
}  
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

DOOR LOCK SYSTEM USING PASSWORD



```
#include <Keypad.h>
#include <LiquidCrystal_I2C.h>

const int RELAY_PIN = A0; // the Arduino pin, which
connects to the IN pin of relay

const int ROW_NUM = 4; //four rows

const int COLUMN_NUM = 4; //four columns

char keys[ROW_NUM][COLUMN_NUM] = {

    {'1','2','3', 'A'},
    {'4','5','6', 'B'},
```

```
{'7','8','9', 'C'},  
{'*','0','#', 'D'}  
};  
  
byte pin_rows[ROW_NUM] = {9, 8, 7, 6}; //connect to the  
row pinouts of the keypad  
  
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2};  
//connect to the column pinouts of the keypad  
  
  
  
Keypad keypad = Keypad( makeKeymap(keys), pin_rows,  
pin_column, ROW_NUM, COLUMN_NUM );  
  
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27  
(from DIYables LCD), 16 column and 2 rows  
  
  
  
const String password_1 = "1234ABC"; // change your  
password here  
  
const String password_2 = "5642CD"; // change your  
password here
```

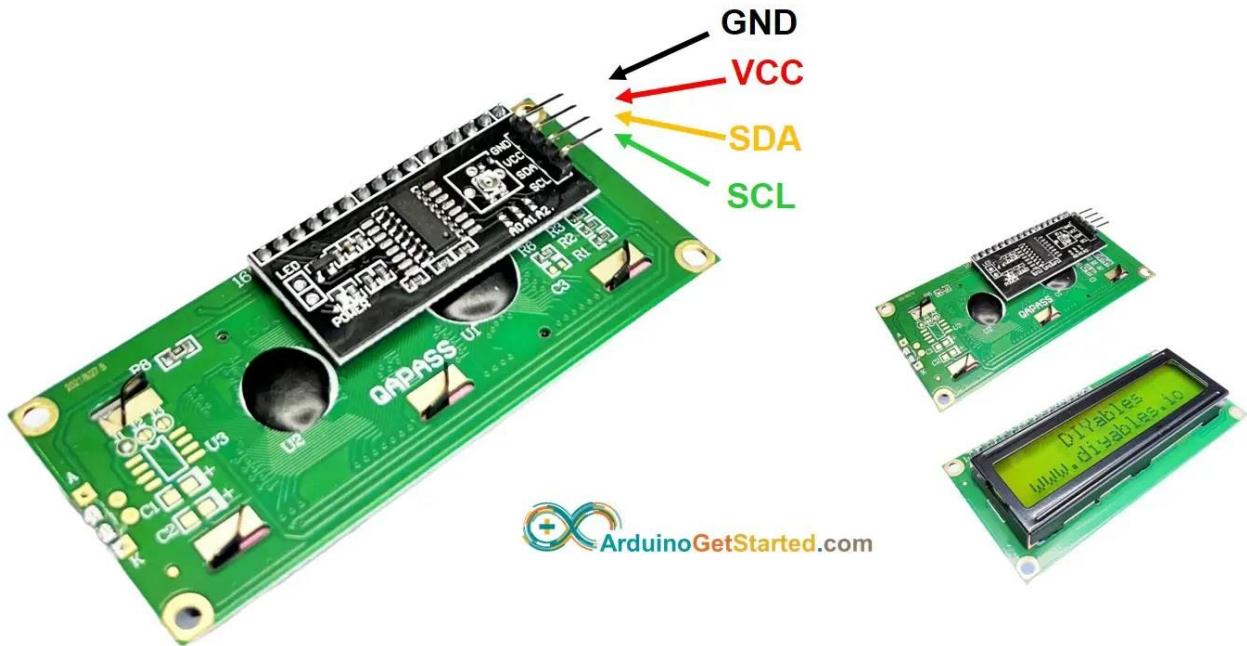
```
const String password_3 = "4545B"; // change your  
password here  
  
String input_password;  
  
void setup(){  
    Serial.begin(9600);  
  
    input_password.reserve(32); // maximum input  
characters is 33, change if needed  
  
    pinMode(RELAY_PIN, OUTPUT); // initialize pin as an  
output.  
  
    digitalWrite(RELAY_PIN, HIGH); // lock the door  
  
    lcd.init(); // initialize the lcd  
  
    lcd.backlight();  
}  
  
void loop(){  
    char key = keypad.getKey();
```

```
if (key){  
    Serial.println(key);  
  
    if(key == '*') {  
        input_password = ""; // reset the input password  
        lcd.clear();  
    } else if(key == '#') {  
        lcd.clear();  
        if(input_password == password_1 || input_password  
        == password_2 || input_password == password_3) {  
            Serial.println("password is correct");  
            lcd.setCursor(0, 0);  
            lcd.print("CORRECT!");  
            lcd.setCursor(0, 1);  
            lcd.print("DOOR UNLOCKED!");  
        }  
    }  
}
```

```
digitalWrite(RELAY_PIN, LOW); // unlock the door  
for 20 seconds  
  
delay(20000);  
  
digitalWrite(RELAY_PIN, HIGH); // lock the door  
  
} else {  
  
    Serial.println("password is incorrect, try again");  
  
    lcd.setCursor(0, 0);  
  
    lcd.print("INCORRECT!");  
  
    lcd.setCursor(0, 1);  
  
    lcd.print("ACCESS DENIED!");  
  
}  
  
  
  
input_password = ""; // reset the input password  
  
} else {  
  
    if(input_password.length() == 0) {  
  
        lcd.clear();  
  
    }  
}
```

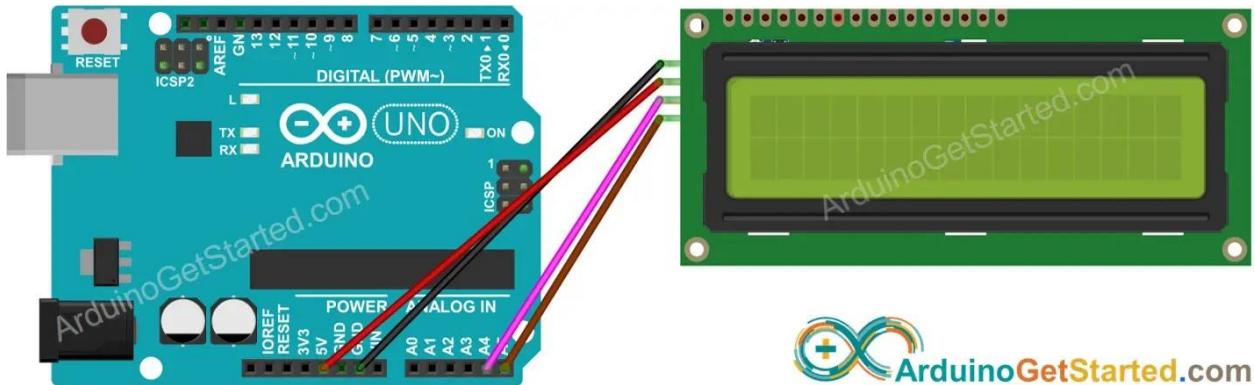
```
    input_password += key; // append new character to  
input password string  
  
    lcd.setCursor(input_password.length(), 0); // move  
cursor to new position  
  
    lcd.print('*');           // print * key as hiden  
character  
}  
}  
}
```

I2C LCD



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK





 ArduinoGetStarted.com

```
#include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16  
column and 2 rows
```

```
void setup()
```

```
{
```

```
lcd.init(); // initialize the lcd
```

```
lcd.backlight();
```

```
}
```

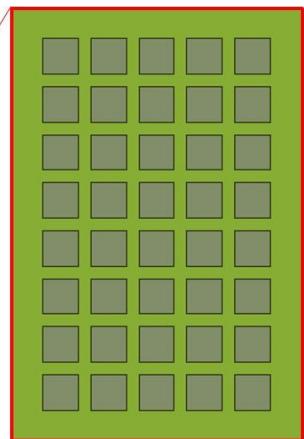
```
void loop()
{
    lcd.clear();          // clear display
    lcd.setCursor(0, 0);  // move cursor to (0, 0)
    lcd.print("Arduino"); // print message at (0, 0)
    lcd.setCursor(2, 1);  // move cursor to (2, 1)
    lcd.print("GetStarted.com"); // print message at (2, 1)
    delay(2000);          // display the above for two
    seconds

    lcd.clear();          // clear display
    lcd.setCursor(3, 0);  // move cursor to (3, 0)
    lcd.print("DIYables"); // print message at (3, 0)
    lcd.setCursor(0, 1);  // move cursor to (0, 1)
    lcd.print("www.diyables.io"); // print message at (0, 1)
    delay(2000);          // display the above for two
    seconds
```

}



ArduinoGetStarted.com



ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK



The character generator represents a character (40 pixels). You just need to do the following steps:

1. Click on each pixel to select/deselect

2. Copy below custom character code

```
byte customChar[8] = {  
    0b00000,  
    0b01010,  
    0b11111,  
    0b11111,  
    0b01110,  
    0b01000,  
    0b00000,  
    0b00000  
};
```

Clear

```
#include <LiquidCrystal_I2C.h>
```

```
LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16  
column and 2 rows
```

```
byte customChar[8] = {  
  
    0b00000,  
  
    0b01010,  
  
    0b11111,  
  
    0b11111,  
  
    0b01110,  
  
    0b00100,  
  
    0b00000,  
  
    0b00000  
};
```

```
void setup()  
{
```

```
lcd.init(); // initialize the lcd  
lcd.backlight();  
  
lcd.createChar(0, customChar); // create a new custom  
character  
  
lcd.setCursor(2, 0); // move cursor to (2, 0)  
lcd.write((byte)0); // print the custom char at (2, 0)  
}  
  
void loop()  
{  
}
```

- ◆ Create custom character and assign to an index value (from 0 to 7) in setup() function

```
1 lcd.createChar(index, customChar);
```

- ◆ Print the custom character in LCD anytime, anywhere (in setup() or loop() function)

```
1 lcd.setCursor(column, row); // move cursor to a desired position  
2 lcd.write((byte)index); // print the custom char at the desired position
```

Other functions

Add the below functions into loop() function one by one. And add delay(5000) after each function

- ◆ Clear LCD screen

```
1 lcd.clear();
```

- ◆ Move the cursor to the upper-left of the LCD

```
1 lcd.home();
```

- ◆ Move the cursor to the a position (column, row)

```
1 lcd.setCursor(column, row);
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK

- ◆ lcd.setCursor(column, row);

- ◆ Display the LCD cursor

```
1 lcd.cursor();
```

- ◆ Hides the LCD cursor.

```
1 lcd.noCursor();
```

- ◆ Display the blinking LCD cursor

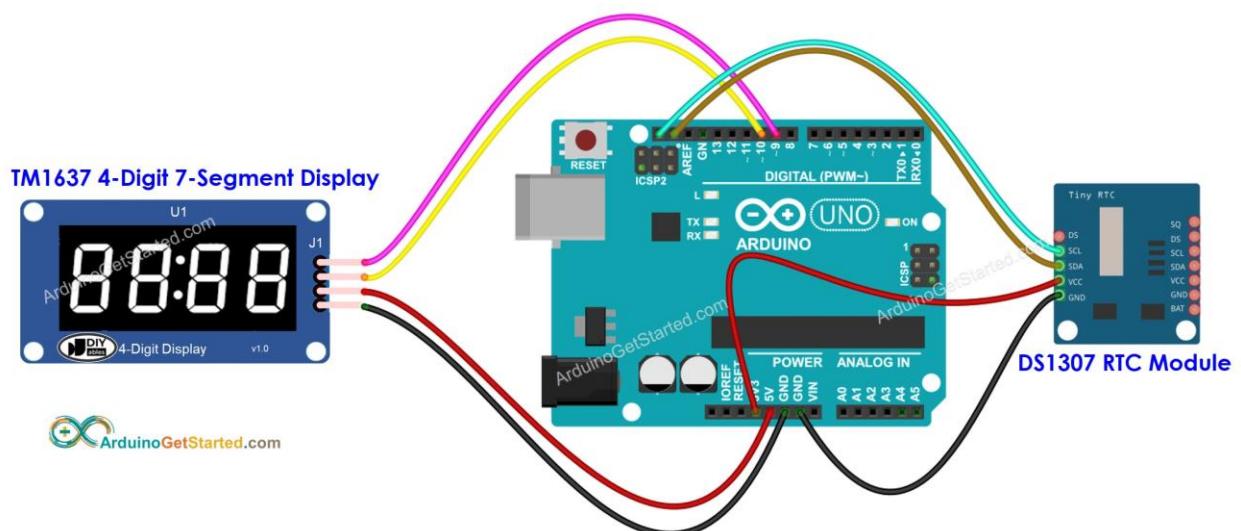
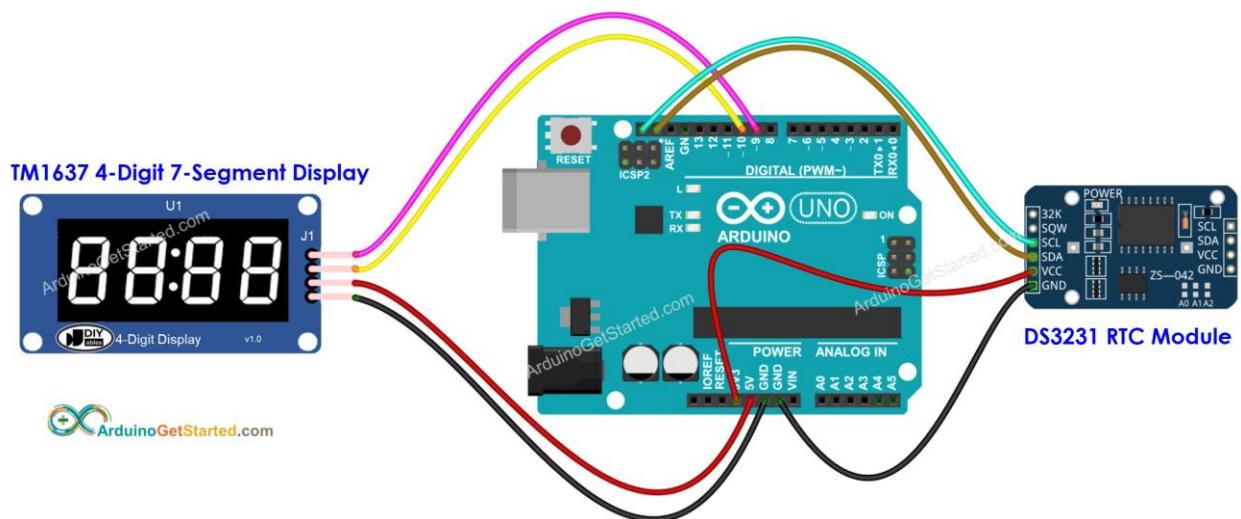
```
1 lcd.blink();
```

- ◆ Turns off the blinking LCD cursor.

```
1 lcd.noBlink();
```

- ◆ And more at [LiquidCrystal Library Reference](#)

7 SEGMENT CLOCK



DISPLAY MINUTE AND SECONDS IN 7 SEGMENT DISPLAY

```
#include <TM1637Display.h>

#include <RTCLib.h>

// define the connections pins

#define CLK 9

#define DIO 10

// create a display object of type TM1637Display

TM1637Display display = TM1637Display(CLK, DIO);

RTC_DS1307 rtc;

// RTC_DS3231 rtc; // uncomment this line and comment
the above line if using DS3231 module

unsigned long time_h = 0; // the variable to store hour
```

```
unsigned long time_m = 0; // the variable to store minute  
unsigned long last_m = 0; // the variable to store the last  
updated hour  
  
void setup() {  
    Serial.begin(9600);  
    display.clear();  
    display.setBrightness(7); // set the brightness to 7  
(0:dimmest, 7:brightest)  
  
    // SETUP RTC MODULE  
    if (!rtc.begin()) {  
        Serial.println("Couldn't find RTC");  
        Serial.flush();  
        while (true)  
        ;  
    }  
}
```

```
// automatically sets the RTC to the date & time on PC  
this sketch was compiled  
  
rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));  
  
}  
  
void loop() {  
  
    DateTime now = rtc.now();  
  
    time_h = now.hour();  
  
    time_m = now.minute();  
  
    if (time_m != last_m) { // only update if changed  
        unsigned long time = time_h * 100 + time_m;  
        display.showNumberDecEx(time, 0b11100000, false, 4,  
0);
```

```
Serial.print(time_h);  
Serial.print(":");  
Serial.println(time_m);
```

```
last_m = time_m;
```

```
}
```

ARDUINO TUTORIAL BY HIMANSHU SINGH LECTURER GPK