# Arduino UNO R4 Minima Analog Sensor Tutorial

## Project of analog sensors

### Description

In this tutorial, we'll explore how to utilize analog sensors with the Arduino UNO R4 Minima microcontroller. Analog sensors are incredibly versatile and can be used for a variety of applications, from measuring soil moisture levels to detecting raindrops. We'll focus on two common analog sensors: the raindrop sensor and the soil moisture sensor. By the end of this tutorial, you'll have a solid understanding of how to interface these sensors with the Arduino UNO R4 Minima and use the data they provide for your projects.

* Raindrop Sensor

The raindrop sensor is a simple module that can detect raindrops. It works by measuring the resistance between two conductive probes on the sensor board. When raindrops fall on the probes, the resistance decreases due to the conductivity of water, allowing us to detect the presence of rain.

* Soil Moisture Sensor

The soil moisture sensor is used to measure the moisture content of soil. It typically consists of two probes that are inserted into the soil. The resistance between the probes changes depending on the moisture level of the soil. Dry soil has high resistance, while wet soil has low resistance.

## How-To Guide

### 1. Gather Components

* **1 x Arduino Uno R4 Minima**
* **1 x Breadboard**
* **1 x Raindrop Sensor**
* **1 x Soil Moisture Sensor**
* **10 x Jumper wires**

### 2. Assemble the Circuit

* Connect the raindrop sensor and soil moisture sensor to the breadboard.
* Connect the VCC and GND pins of both sensors to the 5V and GND rails on the breadboard, respectively.
* Connect the analog output pin of the raindrop sensor to analog pin A0 on the Arduino Uno R4.
* Connect the analog output pin of the soil moisture sensor to analog pin A1 on the Arduino Uno R4.

|  |  |
| --- | --- |
| Arduino UNO R4 Minima | Rain drop sensor |
| 3.3V | VCC |
| GND | GND |
| NC - Not Connect | DO |
| A0 | AO |

|  |  |
| --- | --- |
| Arduino UNO R4 Minima | Soil Moisture Sensor |
| 3.3V | VCC |
| GND | GND |
| A1 | AO |
| NC- Not Connect | DO |

### 3. Write the Arduino Code

Open the Arduino IDE on your computer.

// Define analog sensor pins

const int raindropSensorPin = A0;

const int soilMoistureSensorPin = A1;

void setup() {

  Serial.begin(9600); // Initialize serial communication for debugging

}

void loop() {

  // Read analog sensor values

  int raindropValue = analogRead(raindropSensorPin);

  int soilMoistureValue = analogRead(soilMoistureSensorPin);

  // Print sensor values to serial monitor

  Serial.print("Raindrop Sensor Value: ");

  Serial.println(raindropValue);

  Serial.print("Soil Moisture Sensor Value: ");

  Serial.println(soilMoistureValue);

  // Add your logic based on sensor readings here

  // Example: if (raindropValue > threshold) { // Do something }

  delay(1000); // Delay for stability

}

### 4. Understand the Code

The code initializes two variables to store analog sensor pin numbers.

In the **setup()** function, it initializes serial communication for debugging.

The **loop()** function continuously reads analog sensor values using **analogRead()**.

It prints the sensor values to the serial monitor for observation.

You can add your logic based on sensor readings to trigger specific actions.

### 5. Test the Project

* Upload the code to your Arduino Uno R4 by clicking the upload button in the IDE.
* Open the serial monitor to observe sensor values.
* Adjust environmental conditions around the sensors and observe changes in readings.

### 6. Experiment and Learn

Modify the code to include thresholds and trigger actions based on sensor readings.

Explore different ways to utilize sensor data, such as controlling actuators or displaying information on external devices.

Experiment with other analog sensors to expand your understanding of sensor interfacing with Arduino.

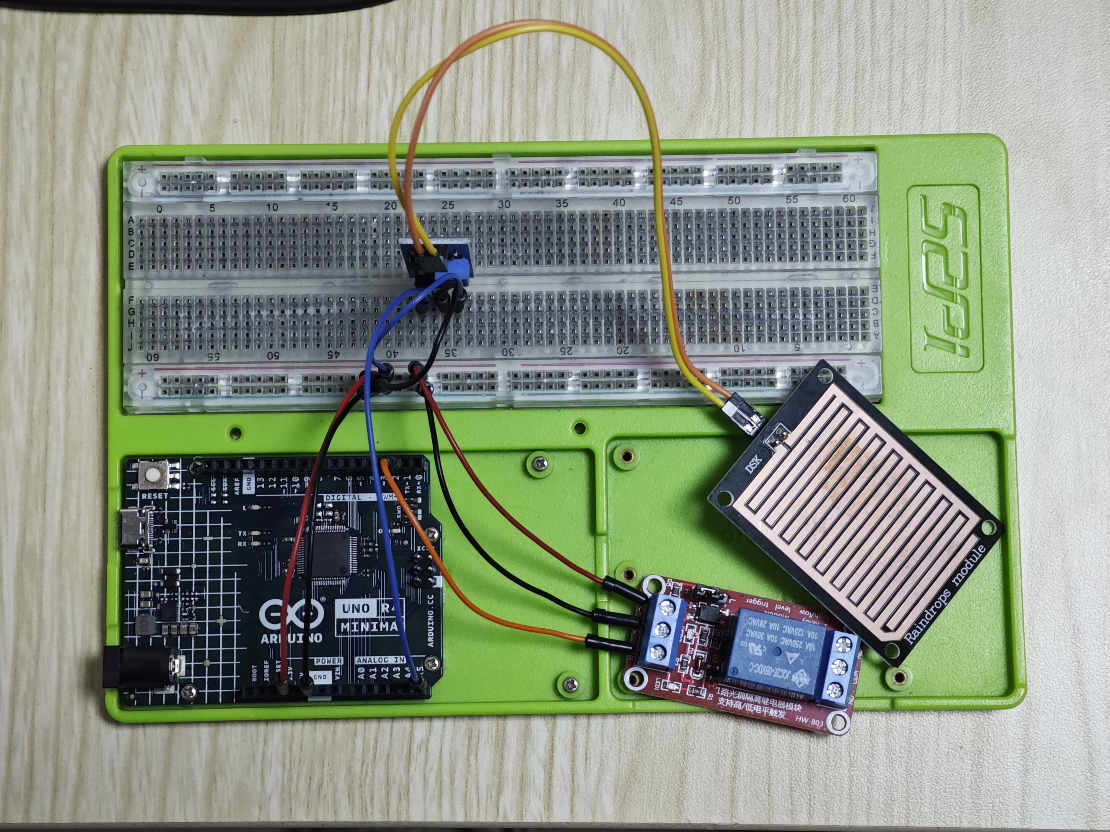
This project provides a foundation for building more complex systems that interact with the environment using analog sensors and Arduino Uno R4. Have fun experimenting and learning!

### 7. Rain drop sensor with Relay module

#### Wiring diagram

|  |  |
| --- | --- |
| Arduino UNO R4 Minima | Rain drop sensor |
| 3.3V | VCC |
| GND | GND |
| NC - Not connect | DO |
| A5 | AO |

|  |  |
| --- | --- |
| Arduino UNO R4 Minima | Relay |
| 5V | DC+ |
| GND | DC- |
| D2 | IN |



#### Demo code

#define SIGNAL\_PIN A5    // Define the output signal

#define RELAY\_PIN 2     // Define the RELAY pin

int value = 0; // variable to store the sensor value

void setup() {

  Serial.begin(9600);

  pinMode(RELAY\_PIN, OUTPUT);   // configure D2 pin as an OUTPUT

}

void loop() {

  delay(10);                      // wait 10 milliseconds

  value = analogRead(SIGNAL\_PIN); // read the analog value from sensor

  Serial.println(value);

  if(value > 800){        //If the value is greater than 800, we can consider it's dry

    Serial.println(" dry!!! ");

    digitalWrite(RELAY\_PIN, LOW);  // turn RELAY OFF

  }else {     //else, we can consider it's raining

    Serial.println(" raining!!! ");

    digitalWrite(RELAY\_PIN, HIGH);  // turn RELAY ON

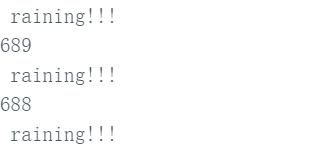
  }

  delay(1000);

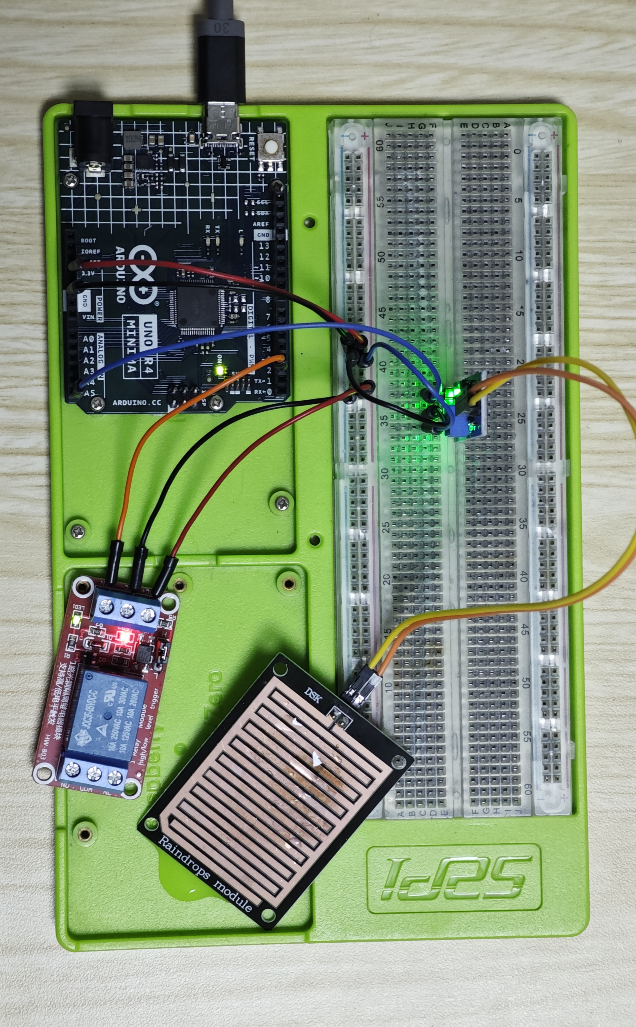
}

Upload the code and open serial monitor.

If you drop some water on the sensor pad.



and the Relay will be triggered.



## Project: Infrared Sensor and IR Controller with Relay

### Description

In this project, we'll use an infrared (IR) sensor to detect motion or proximity and an IR remote controller to trigger actions. We'll interface these components with the Arduino Uno R4 Minima and control a relay to switch external devices like lights, fans, or other appliances wirelessly.

## How-To Guide

### 1. Gather Components:

* 1 x Arduino Uno R4 Minima
* 1 x Breadboard
* 1 x Infrared Controller
* 1 x Infrared Receiver Module
* 1 x Relay Module
* Jumper wires

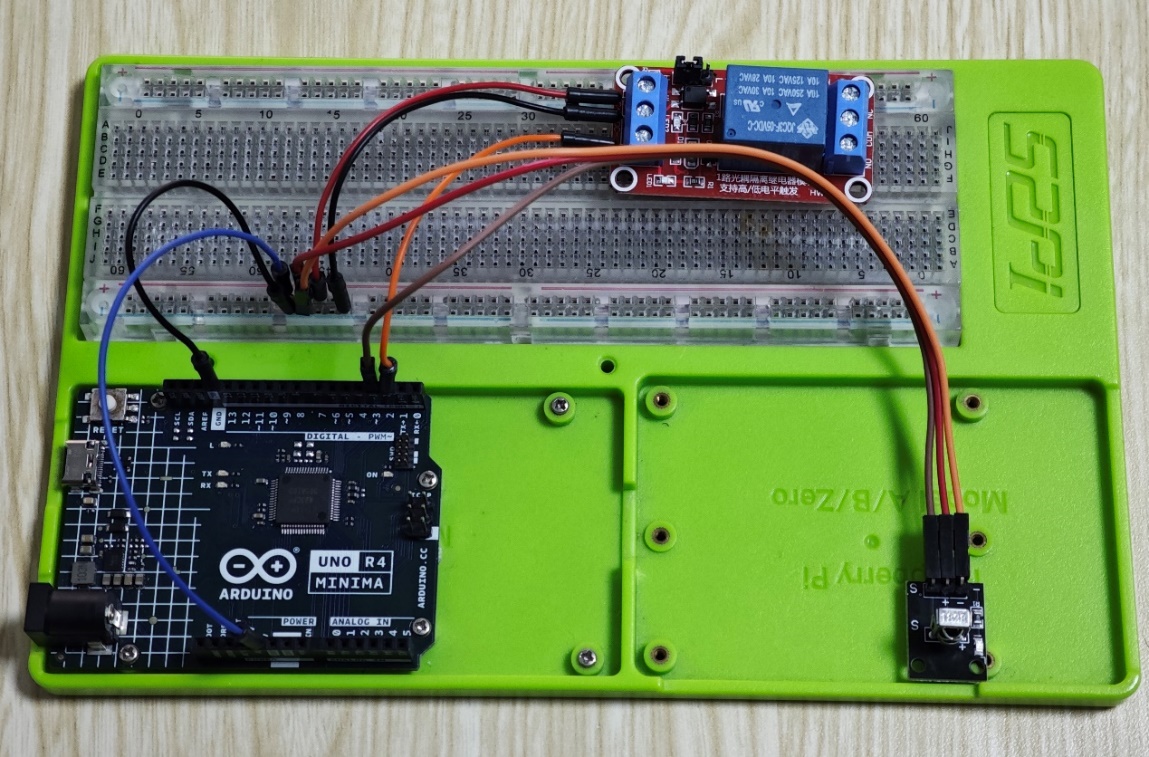
### 2. Assemble the Circuit

* Connect the VCC and GND pins of the infrared sensor, IR receiver, and relay module to the 5V and GND rails on the breadboard, respectively.
* Connect the output pin of the infrared sensor to digital pin 3 on the Arduino Uno R4.
* Connect the output pin of the IR receiver module to digital pin 2 on the Arduino Uno R4.
* Connect the control pin of the relay module to digital pin 4 on the Arduino Uno R4.

#### Wiring Diagram:

|  |  |
| --- | --- |
| Arduino UNO R4 Minima | IR Receiver Module |
| 3.3v | + |
| GND | - |
| 3 | S |

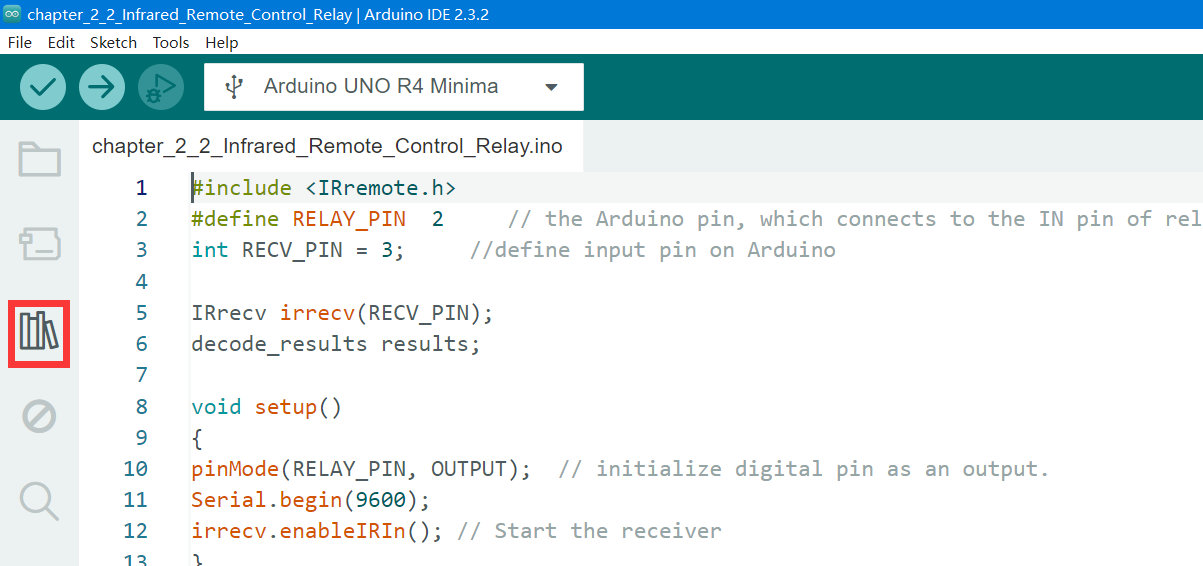
|  |  |
| --- | --- |
| Arduino UNO R4 Minima | Relay |
| 5v | DC+ |
| GND | DC- |
| 2 | IN |

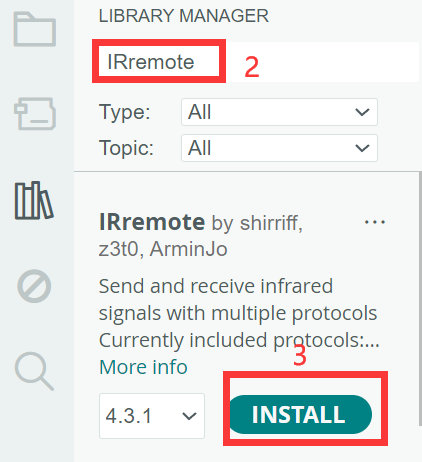


### 3. Install library IRremote

Open the Arduino IDE on your computer.

**Note**: When you include a library in your code, you should go to the library manager and download it.





### 4. Write the Arduino Code

#include <IRremote.h>

#define RELAY\_PIN  2     // the Arduino pin, which connects to the IN pin of relay module

int RECV\_PIN = 3;     //define input pin on Arduino

IRrecv irrecv(RECV\_PIN);

decode\_results results;

void setup()

{

pinMode(RELAY\_PIN, OUTPUT);  // initialize digital pin as an output.

Serial.begin(9600);

irrecv.enableIRIn(); // Start the receiver

}

void loop() {

if (irrecv.decode()) {

 if (irrecv.decodedIRData.protocol == UNKNOWN) {

      Serial.println(F("Received noise or an unknown (or not yet enabled) protocol"));

      // We have an unknown protocol here, print extended info

      irrecv.printIRResultRawFormatted(&Serial, true);

      irrecv.resume();  // Do it here, to preserve raw data for printing with printIRResultRawFormatted()

    } else {

      irrecv.resume();  // Early enable receiving of the next IR frame

      irrecv.printIRResultShort(&Serial);

      irrecv.printIRSendUsage(&Serial);

    }

    Serial.println();

    Serial.println(irrecv.decodedIRData.command);

    if (irrecv.decodedIRData.command == 0x45) {  // 1 Change status

      toggleRelay();

      delay(300);

    }else if(irrecv.decodedIRData.command == 0x46){  // 2 turn on the relay

      digitalWrite(RELAY\_PIN, HIGH);

    }else if(irrecv.decodedIRData.command == 0x47){   // 3 turn off the relay

      digitalWrite(RELAY\_PIN, LOW);

    }

  }

}

void toggleRelay() {

  digitalWrite(RELAY\_PIN, !digitalRead(RELAY\_PIN)); // change the status

}

Then you can modify the code yourself to achieve certain functions by pressing the infrared remote control.

if (irrecv.decodedIRData.command == 0x45) {

// if you press 1(0x45), it will do this.

      toggleRelay();

      delay(300);

    }

And you can open the serial monitor to see the command of the infrared remote control. Of course, we will also provide you with the code of the infrared remote control to facilitate your use.

#### Key Code of IR controller

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **IR Code** | **Key** | **IR Code** |
| 1 | 0x45 | 0 | 0x19 |
| 2 | 0x46 | \* | 0x16 |
| 3 | 0x47 | # | 0x0D |
| 4 | 0x44 | UP | 0x18 |
| 5 | 0x40 | RIGHT | 0x5A |
| 6 | 0x43 | DOWN | 0x52 |
| 7 | 0x07 | LEFT | 0x08 |
| 8 | 0x15 | OK | 0x1C |
| 9 | 0x09 | IR Controller Key code | |

### 5. Understand the Code

This code is written for an Arduino sketch that utilizes an infrared (IR) receiver module to receive signals from an IR remote controller. Based on the received IR commands, it controls a relay module connected to the Arduino. Let's break down the code step by step:

#### 1. Including Libraries and Defining Constants:

#include <IRremote.h>

#define RELAY\_PIN  2     // the Arduino pin, which connects to the IN pin of relay module

int RECV\_PIN = 3;     // define input pin on Arduino

* The code includes the IRremote library, which is necessary for working with IR signals.
* It defines the pin number for the relay module (RELAY\_PIN) and the pin number for the IR receiver module (RECV\_PIN).

#### 2. Initializing IR Receiver and Serial Communication:

IRrecv irrecv(RECV\_PIN);

decode\_results results;

void setup() {

  pinMode(RELAY\_PIN, OUTPUT);  // initialize digital pin as an output.

  Serial.begin(9600);

  irrecv.enableIRIn(); // Start the receiver

}

* An instance of the IRrecv class is created, specifying the pin connected to the IR receiver module.
* Serial communication is initialized for debugging purposes.
* The IR receiver is enabled to start receiving IR signals.

#### 3. Receiving and Processing IR Signals in the Loop:

void loop() {

  if (irrecv.decode()) {

    // Processing decoded IR data

    // Handling different commands

  }

}

* The loop continuously checks if there's any decoded IR signal available.
* When an IR signal is received and decoded, the program proceeds to process the decoded data.

#### 4. Processing Decoded IR Data:

if (irrecv.decodedIRData.protocol == UNKNOWN) {

     // Handling unknown or noisy IR signals

} else {

     // Handling known IR protocols

     // Performing actions based on received commands

}

* If the IR protocol is unknown or noisy, it prints a message indicating that it received noise or an unknown protocol.
* If the IR protocol is known, it prints the decoded IR result and performs actions based on the received commands.

#### 5. Handling Specific IR Commands:

if (irrecv.decodedIRData.command == 0x45) {  // Command to toggle relay status

  toggleRelay();

  delay(300);

} else if (irrecv.decodedIRData.command == 0x46) {  // Command to turn on the relay

  digitalWrite(RELAY\_PIN, HIGH);

} else if (irrecv.decodedIRData.command == 0x47) {   // Command to turn off the relay

  digitalWrite(RELAY\_PIN, LOW);

}

* The code checks the received IR command and performs corresponding actions:
* If the command is 0x45, it calls the toggleRelay() function to toggle the relay status.
* If the command is 0x46, it turns on the relay by setting the relay pin to HIGH.
* If the command is 0x47, it turns off the relay by setting the relay pin to LOW.

#### 6. Toggling Relay Status:

void toggleRelay() {

  digitalWrite(RELAY\_PIN, !digitalRead(RELAY\_PIN)); // Toggle the status of the relay pin

}

* The toggleRelay() function is responsible for toggling the status of the relay pin. It reads the current status of the pin and then changes it to the opposite status.

Overall, this code allows the Arduino to receive IR signals and control a relay module accordingly, enabling wireless control of connected devices.

### 6. Test the Project

Upload the code to your Arduino Uno R4 by clicking the upload button in the IDE.

Adjust the infrared sensor's sensitivity and observe its response to motion/proximity.

Use the IR remote controller to trigger actions and observe the relay's operation.

### 7. Experiment and Learn

Explore different configurations and placements of the infrared sensor for various applications.

Implement additional functionality, such as time delays or multiple relay controls based on different sensor inputs.

Integrate the project with other sensors or actuators to create more complex automation systems.

This project demonstrates how to use infrared sensors and IR controllers to wirelessly control devices via Arduino Uno R4. Experiment with different setups and functionalities to customize it according to your needs. Enjoy exploring the world of electronics and automation!