WATER LEVEL SENSOR

Ever wondered how automatic water tanks know when to stop filling up or how flood alert systems can tell when water is rising to dangerous levels? That's all thanks to simple but clever sensors that can "feel" the presence of water.

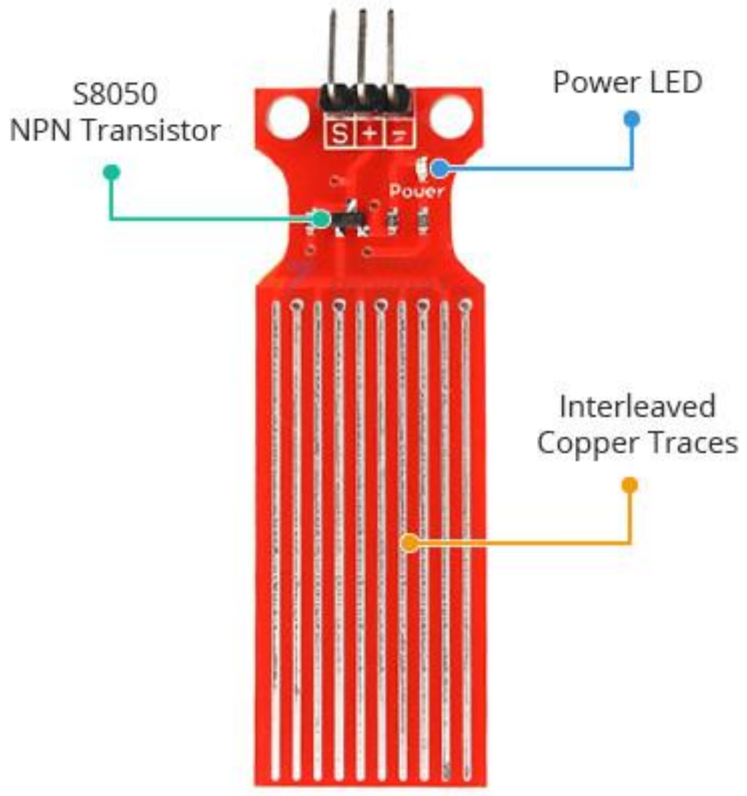
One of the easiest and most beginner-friendly sensors to experiment with is the Water Level Sensor—and when combined with an Arduino, it opens up a world of cool and practical projects!

This guide will walk you through how water level sensors actually work. You'll also find easy-to-follow Arduino code examples that will help you read water levels and use those readings to light up LEDs—giving you a visual way to monitor the water level.

Let's get started!

Hardware Overview

This water level sensor module is designed to give an analog output voltage that is proportional to the water level. In other words, the higher the water level, the higher the output voltage.



At the heart of this sensor is an S8050 NPN transistor, which works like a switch.

The sensing part of the module is made up of ten copper traces, arranged in an interleaved (alternating) pattern. These copper traces are split into two sets:

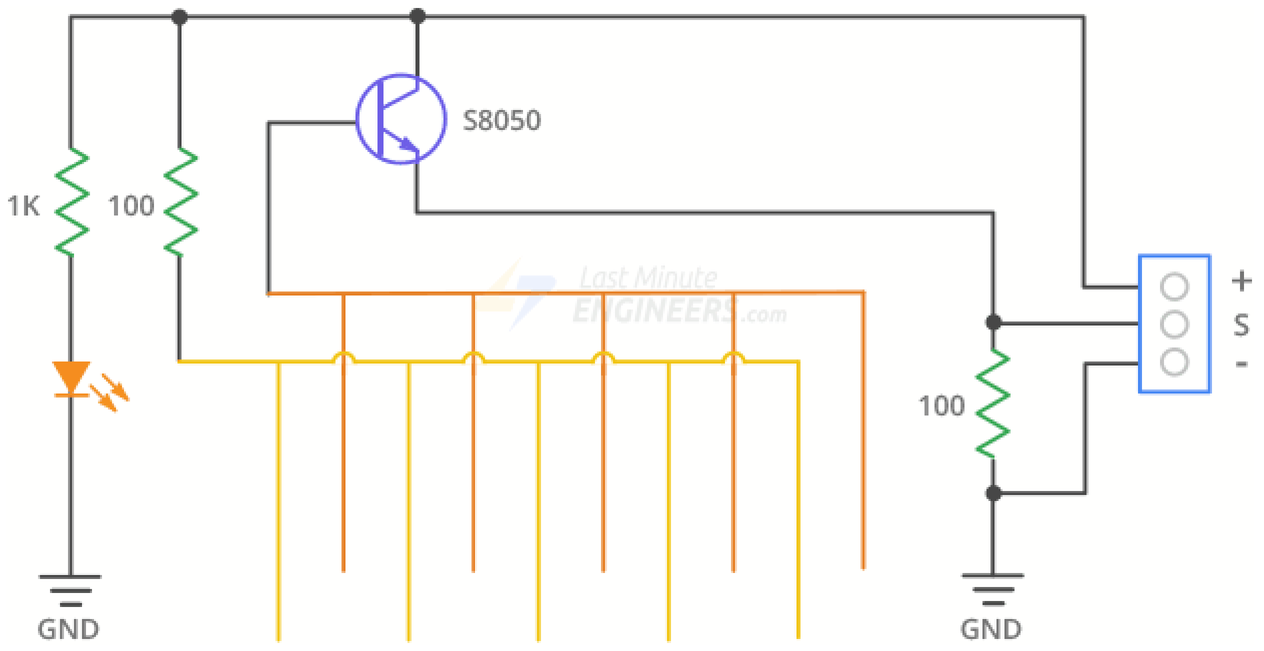
- One set of five traces is connected to the +5V power supply through a 100-ohm resistor.
- The other set is connected to the base of the transistor.

When water touches the sensor, it connects these two sets of traces, allowing a small electric current to flow into the transistor's base. This turns the transistor on, allowing a larger current to flow through it and produce a voltage at the output pin. More on this a little later.

The module also has a power indicator LED. This LED is not involved in the sensing process – it just lights up when the module is powered, letting you know the sensor is turned on and ready to work.

Circuit Diagram

Before we proceed, let's take a look at the circuit diagram of the module. This diagram will help you understand how all the parts are connected and how the sensor works.



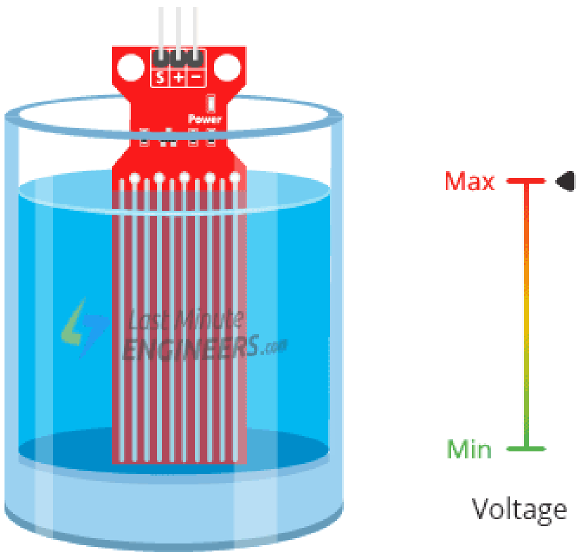
How Does a Water Level Sensor Work?

The water level sensor operates by using the electrical conductivity of water to complete a circuit between two sets of interleaved copper traces.

When the sensor is dry, the copper traces are separated from each other by air. This means no electric current flows between the traces. As a result, no current flows into the base of the NPN transistor, and the transistor remains switched off, similar to an open switch. Because the transistor is off, electricity cannot flow from its collector (connected to +5 volts) to its emitter (which is connected to ground through a 100-ohm resistor). Since the emitter is also connected to the sensor's output pin (OUT), the output reads 0 volts (LOW).

However, when water touches the copper traces, it creates a conductive bridge between them because water, especially with dissolved minerals and impurities, can conduct a small electric current. This tiny current from the traces flows into the base of the transistor. This turns the transistor on. Once the transistor is activated, it allows a much larger current to flow from the +5 volts through the transistor to the emitter and finally to ground through a 100-ohm resistor. Because the OUT pin is connected between the transistor's emitter and this resistor, the OUT pin voltage rises towards +5 volts.

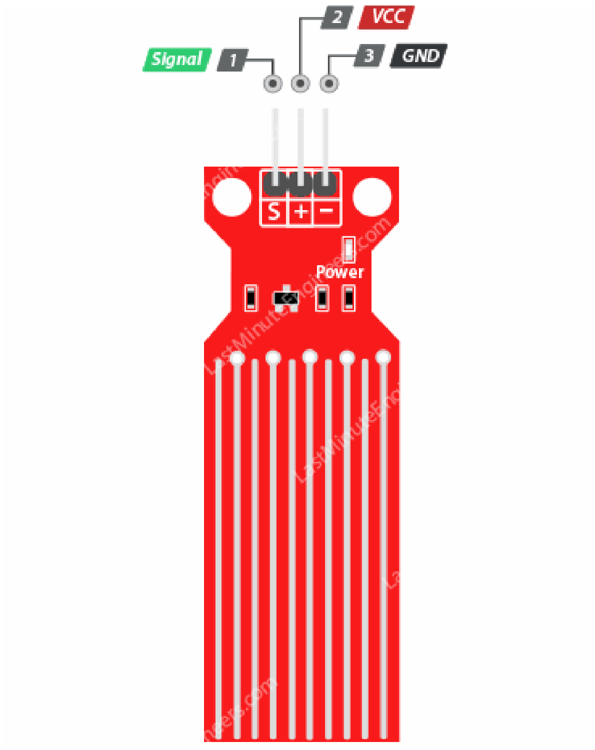
Importantly, the sensor doesn't just turn fully on or fully off—it provides different output voltages depending on how much water touches the sensor. If there's just a little water, the transistor partially activates, causing a small increase in the output voltage. More water increases conductivity, allowing more current into the transistor's base, and raising the output voltage even higher. Thus, the sensor generates an analog voltage signal directly proportional to the water level.



This analog output signal is typically fed to a microcontroller (like an Arduino), which uses its analog-to-digital converter (ADC) to convert the analog signal into a digital value. This digital value represents the water level.

Water Level Sensor Pinout

Let's look at the pinout:



S (Signal) is the analog output pin. It provides a voltage that is proportional to the water level. You connect this pin to one of the analog input pins on your Arduino.

+ (VCC) pin provides power to the sensor. It's best to power the sensor with 3.3V to 5V. Keep in mind that the analog output value will change based on the voltage you supply.

– (GND) is the ground pin.

Wiring a Water Level Sensor to an Arduino

Connecting the water level sensor to your Arduino is super easy! It only has three pins that you need to hook up:

First, connect the + (VCC) pin on the sensor module to the 5V pin on the Arduino, and connect the – (GND) pin on the sensor to one of the GND pins on the Arduino.

However, there's an important thing to keep in mind. One common issue with these sensors is that they don't last very long because they are always in contact with water. When the sensor is constantly powered while submerged, it tends to corrode faster, which reduces its lifespan.

To avoid this problem, it's a good idea to power the sensor only when you need to take a reading. A simple way to do this is by connecting the sensor's power pin to one of the Arduino's digital output pins, and then turning that pin HIGH or LOW in your code as needed.

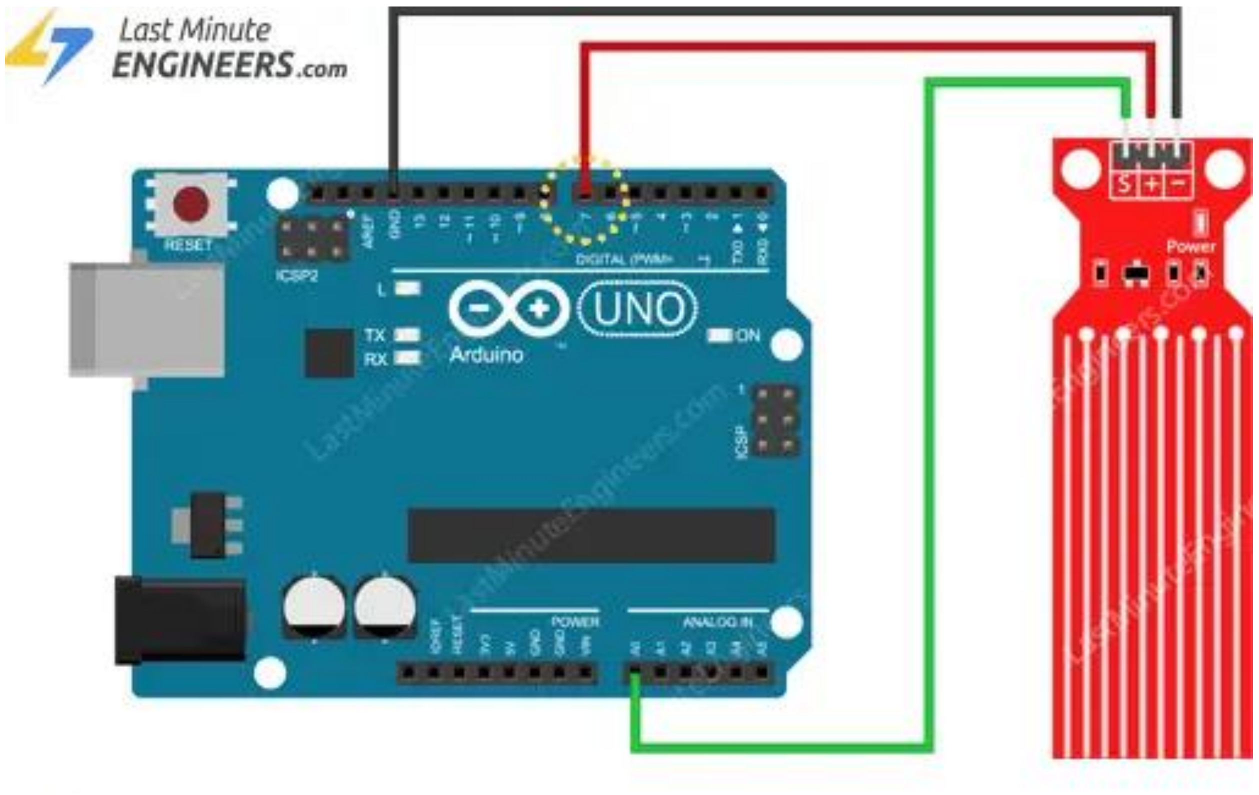
So instead of connecting the VCC pin directly to 5V, we'll connect it to digital pin #7 on the Arduino. This way, the Arduino can control when the sensor is powered.

Finally, connect the S (Signal) pin of the sensor to the A0 analog input pin on the Arduino.

The following table lists the pin connections:

WATER LEVEL SENSOR	**ARDUINO**
+	7
-	GND
S	A0

The wiring is shown in the image below.



Use a digital pin to power the sensor only when taking a reading to reduce corrosion.

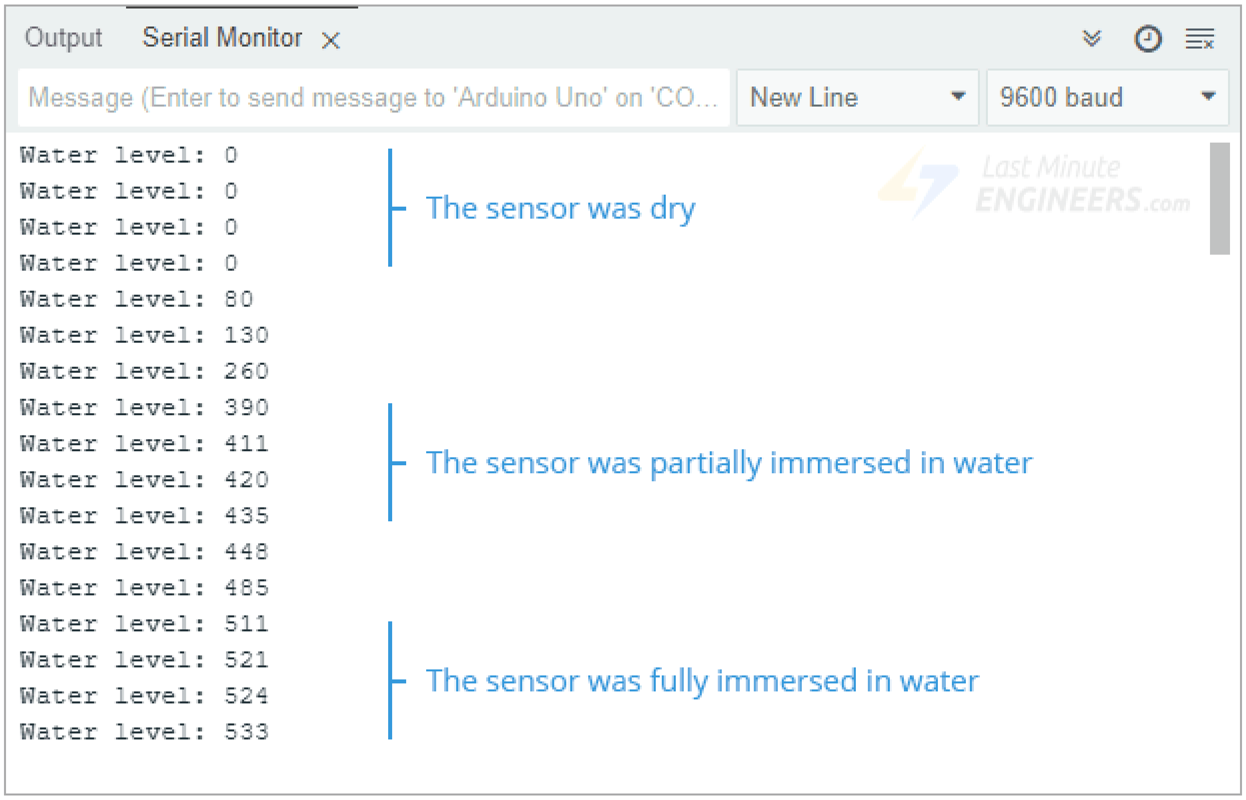
Basic Arduino Example

After building the circuit, the next step is to upload the sketch to your Arduino board.

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

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

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

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

  Serial.begin(9600);
}

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

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

  delay(1000);
}

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

```
return val;                        // send current reading
```

Once the code is uploaded, open the Serial Monitor to view the sensor's output. When the sensor is completely dry, it will show a value of 0. As the water level increases and more of the sensor is submerged, the output value will also increase gradually.



Output
Serial Monitor
Message (Enter to send message to 'Arduino Uno' on 'CO...
New Line
9600 baud
Last Minute ENGINEERS.com

```
Water level: 0
Water level: 0
Water level: 0
Water level: 0
Water level: 80
Water level: 130
Water level: 260
Water level: 390
Water level: 411
Water level: 420
Water level: 435
Water level: 448
Water level: 485
Water level: 511
Water level: 521
Water level: 524
Water level: 533
```

The sensor was dry
The sensor was partially immersed in water
The sensor was fully immersed in water

NOTE:

It's important to note that this sensor is not designed to be fully submerged in water. Only the lower part of the sensor, where the copper traces are exposed, should come into contact with water during use. Make sure that the rest of the sensor, especially the circuit area, stays dry.

Code Explanation

The sketch begins by declaring which Arduino pins are connected to the sensor. The pin labeled + (VCC) on the sensor is connected to pin 7 on the Arduino, and the S (signal) pin is connected to analog input pin A0.

```
#define sensorPower 7
#define sensorPin A0
```

Next, a variable named `val` is created to store the sensor's reading, which represents the current water level.

```
int val = 0;
```

In the loop() section, a custom function called `readSensor()` is called once every second. The result from this function is printed to the Serial Monitor as the current water level reading.

```
Serial.print("Water level: ");
Serial.println(readSensor());
delay(1000);
```

The `readSensor()` function is where the actual sensor reading takes place. It turns the sensor on by setting the power pin HIGH, waits for 10 milliseconds to give the sensor time to stabilize, reads the analog value from the signal pin, then turns the sensor off by setting the power pin LOW. Finally, it returns the value it read so the main program can use it.

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

Finding the threshold values

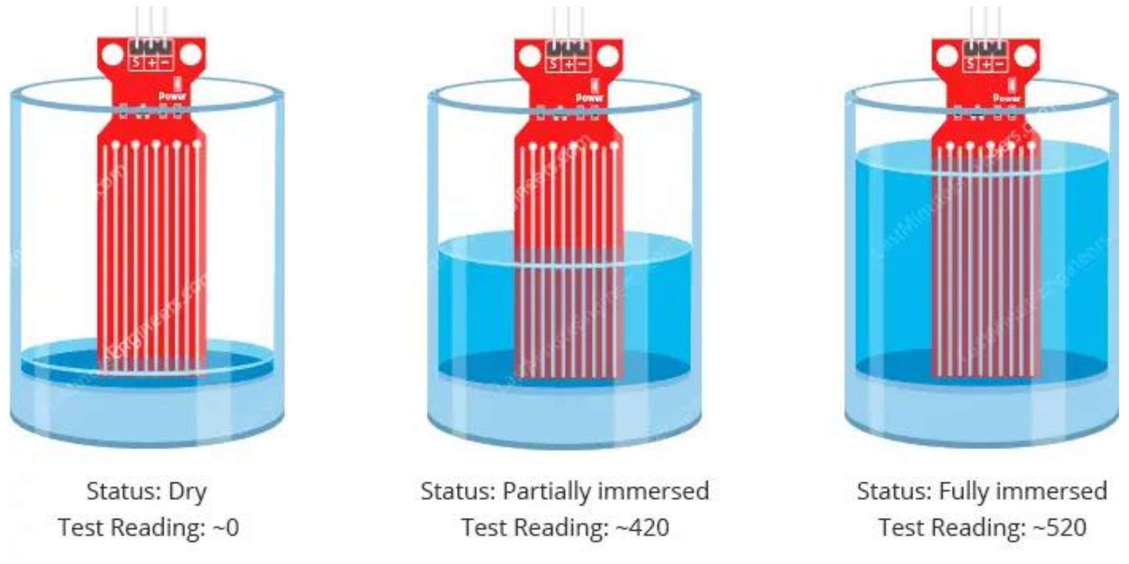
To estimate the water level using the sensor, you need to observe and record the sensor's output values under different conditions—when the sensor is completely dry, partially dipped in water, and fully immersed.

To do this, simply run the Arduino sketch you uploaded earlier and watch the output in the Serial Monitor.

Keep in mind that the sensor's readings may vary depending on the type of water you're using. For example, pure distilled water doesn't conduct electricity well because it lacks minerals; It's the dissolved minerals and impurities in water that allow electricity to pass through.

When you run the sketch, you should see values similar to these:

- When the sensor is dry, the reading is 0.
- When the sensor is partially immersed, the reading is around 420.
- When the sensor is fully immersed, the reading goes up to about 520.



These numbers may not be exactly the same for everyone, so this test may require a bit of trial and error. Once you've recorded your own readings, you can use them as threshold values in your program. These thresholds help your Arduino decide when to trigger an action—for example, turning on a pump or sending an alert—based on the water level.

REFERENCE:

https://lastminuteengineers.com/water-level-sensor-arduino-tutorial/