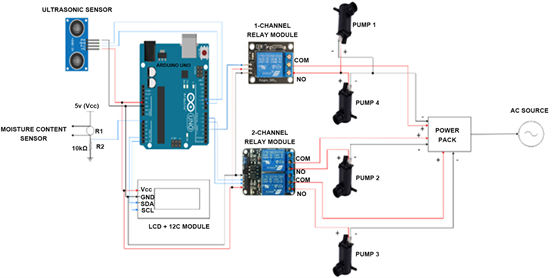
**IOT PHASE 3**

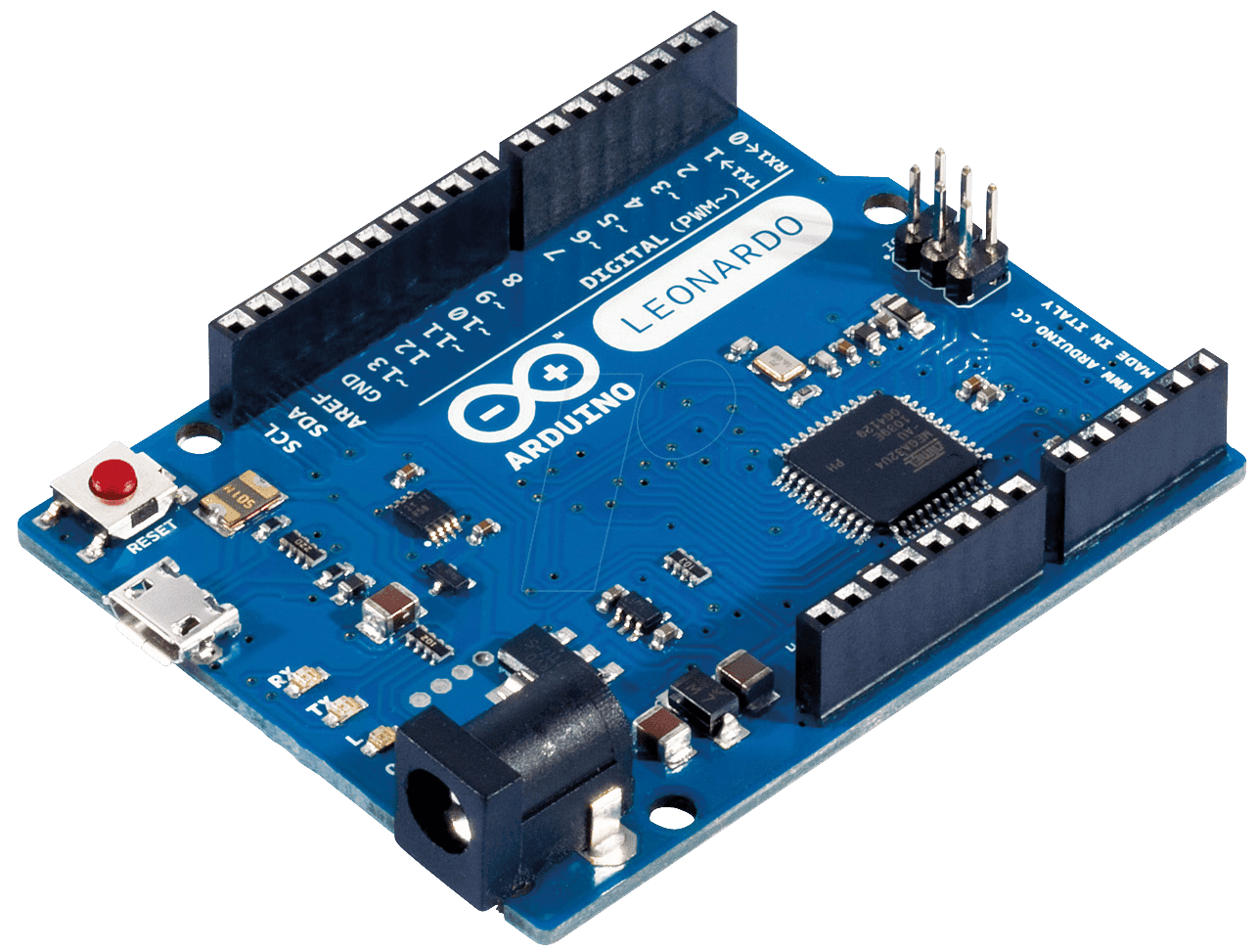
SMART WATER SYSTEM



# **Introduction:**

Modern methods for utilizing Arduino to manage water resources wisely include smart water-based systems. These systems can track and manage water use in a variety of applications by merging Arduino microcontrollers with sensors like moisture detectors, flow meters, and water level sensors. Arduino-based smart water systems offer real-time data and automation capabilities, making them ideal for controlling irrigation in agriculture, controlling water levels in tanks or reservoirs, and even finding problems in household plumbing. These systems not only conserve water but also cut costs and encourage sustainable methods of water management since they have the capacity to gather data, evaluate it, and make informed judgments. They are adaptable and available technology for both people and businesses interested in water conservation since they can be scaled and adjusted to meet varied needs.

# **Components Required:**

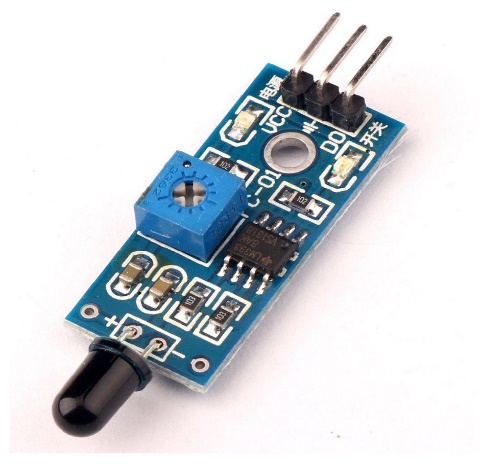


[This Photo](http://www.electronicaivanespinoza.com/2019/05/que-es-arduino-conceptos-basicos-y.html) by Unknown Author is licensed under [CC BY-SA-NC](https://creativecommons.org/licenses/by-nc-sa/3.0/)

1.Arduino Microcontroller:

As the system's central processing unit, select an Arduino board (e.g., Arduino Uno, Arduino Nano, or Arduino Mega).

## 2. Sensors:

 Soil Moisture Sensor: Measures soil moisture content for efficient irrigation.

Water Level Sensor: Monitors water levels in tanks, reservoirs, or bodies of water.

Flow Meter: Measures the flow rate of water in pipes or irrigation systems.

Temperature and Humidity Sensor: Provides data for weather and environmental conditions.

## 3. Actuators:

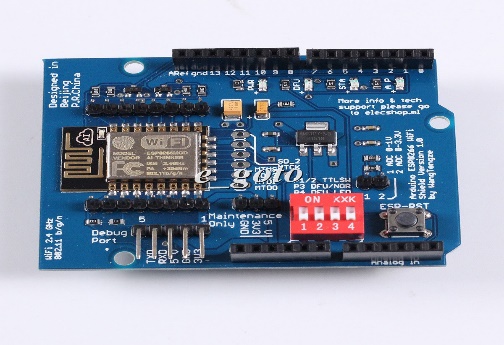
Water Pumps: Control water flow for irrigation or distribution.

Solenoid Valves: Regulate the flow of water in pipes or irrigation systems.

Relays: Switch high-voltage devices, like pumps or valves, on and off.

## 4. Communication Module:

Wi-Fi, GSM, LoRa, or Bluetooth module to enable data transmission and remote control.



[This Photo](http://arduino.stackexchange.com/questions/24919/how-to-connect-wi-fi-shield-esp-12e-esp8266-uart-wifi-wireless-shield-with-ardui) by Unknown Author is licensed under [CC BY-SA](https://creativecommons.org/licenses/by-sa/3.0/)

## 5.Power Supply:

Depending on your application, you may need a power source, such as a battery, solar panels, or an AC power supply.

# **Python script:**

A python script for the sensor and acutators to perform operation:

import RPi.GPIO as GPIO

import time

# Define GPIO pin numbers

trigPin = 18

echoPin = 24

# Set GPIO mode to BCM

GPIO.setmode(GPIO.BCM)

# Set GPIO pin directions

GPIO.setup(trigPin, GPIO.OUT)

GPIO.setup(echoPin, GPIO.IN)

def measure\_distance():

# Send a 10us pulse to trigger the ultrasonic sensor

GPIO.output(trigPin, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(trigPin, GPIO.LOW)

# Measure the time it takes for the echo to return

pulse\_start = time.time()

while GPIO.input(echoPin) == 0:

pulse\_start = time.time()

while GPIO.input(echoPin) == 1:

pulse\_end = time.time()

# Calculate the distance based on the echo time

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150

distance = round(distance, 2)

return distance

# Define water level thresholds

empty\_level = 10

low\_level = 20

full\_level = 30

# Continuously monitor the water level

while True:

distance = measure\_distance()

water\_level = 30 - distance

# Check water level status

if water\_level <= empty\_level:

print("Water tank is empty.")

elif water\_level <= low\_level:

print("Water level is low.")

elif water\_level <= full\_level:

print("Water level is sufficient.")

else:

print("Water tank is full.")

# Add water if level is low

if water\_level <= low\_level:

print("Adding water...")

# Implement water pump activation here

time.sleep(5) # Simulate water filling for 5 seconds

time.sleep(1) # Delay between measurements

# Clean up GPIO resources

GPIO.cleanup()

Additional for adding wifi module through HTTPS protocol

import serial

import time

import requests

# Define the serial port for communication with Arduino

serial\_port = 'COM3' # Change this to the correct port

# Define the IP address and port of the ESP8266 module

esp8266\_ip = '192.168.1.100' # Change this to the IP address of your ESP8266

esp8266\_port = 80 # Change this to the port your ESP8266 is listening on

# Create a serial connection to the Arduino

arduino = serial.Serial(serial\_port, 9600, timeout=1)

def read\_soil\_moisture():

# Send a request to the ESP8266 to get soil moisture reading

url = f'http://{esp8266\_ip}:{esp8266\_port}/moisture'

response = requests.get(url)

moisture\_reading = response.text

return int(moisture\_reading)

def control\_valve(valve\_state):

# Send a command to the ESP8266 to control the solenoid valve

url = f'http://{esp8266\_ip}:{esp8266\_port}/{valve\_state}'

response = requests.get(url)

try:

while True:

# Read soil moisture

moisture = read\_soil\_moisture()

print(f"Soil Moisture: {moisture}")

# Control the solenoid valve based on moisture level

if moisture < 300: # Adjust the threshold as needed

control\_valve('open')

else:

control\_valve('close')

time.sleep(3600) # Check moisture level every hour

except KeyboardInterrupt:

arduino.close()

# Conclusion:

Finally, combining Arduino, an ESP8266 module, and sensors with the provided Python script yields a versatile, cost-effective, and user-friendly smart water-based system. This technology improves water management techniques, minimizes water waste, and is adaptable to a variety of water-related applications. Individuals, small-scale farmers, and anybody interested in encouraging efficient water resource management will find it useful.