Project Name: **Automatic Irrigation System**

- ❖ **Components Required**

1. Arduino UNO
2. Soil Moisture Sensor
3. 1-Channel Relay Module
4. Submersible Water Pump
5. Water container
6. External power supply/battery for pump
7. Jumper wires
8. Breadboard

- ❖ **Connections**

Soil Moisture Sensor → Arduino UNO

Soil Sensor	**Arduino UNO**
VCC	5V
GND	GND
AO	A0

Relay Module → Arduino UNO

Relay	**Arduino UNO**
VCC	5V
GND	GND
IN	D7

Pump → Relay

For a DC submersible pump:

Relay Terminal Connection

COM	Pump power supply +
NO	Pump +
Pump –	Pump power supply –

Use an **external power supply suitable for your pump**. Do not power the pump directly from the Arduino 5V pin.

Working

- Soil is **dry** → Arduino turns **ON the relay** → pump starts watering.
- Soil becomes **wet** → Arduino turns **OFF the relay** → pump stops.
- The system automatically maintains soil moisture without manually operating the pump.

Arduino Code

```
#define SOIL_PIN A0
#define RELAY_PIN 7

int soilValue;
int dryThreshold = 700;

void setup() {
  pinMode(SOIL_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);

  // Relay OFF initially
  digitalWrite(RELAY_PIN, HIGH);

  Serial.begin(9600);
}
```

```
void loop() {

  soilValue = analogRead(SOIL_PIN);

  Serial.print("Soil Moisture Value: ");
  Serial.println(soilValue);

  if (soilValue > dryThreshold) {
    // Soil is dry
    digitalWrite(RELAY_PIN, LOW);

    Serial.println("Soil is DRY - Pump ON");
  }
  else {
    // Soil is wet
    digitalWrite(RELAY_PIN, HIGH);

    Serial.println("Soil is WET - Pump OFF");
  }

  delay(1000);
}
```

Note

The 700 threshold is an example. Soil moisture sensor values vary depending on the soil and sensor. Check the value in the Serial Monitor for dry and wet soil, then adjust dry Threshold.

Project Name: **Smart Water Pump Controller**

- ❖ **Components Required**

1. Arduino UNO
2. Water Level Sensor
3. Traffic Light LED Module (Red, Yellow, Green)
4. Buzzer
5. Relay Module
6. Water Pump
7. External power supply for pump
8. Jumper wires
9. Breadboard

- ❖ **Connections**

Water Level Sensor → Arduino UNO

Water Level Sensor	Arduino UNO
VCC	5V
GND	GND
SIG/AO	A0

Traffic Light LED → Arduino UNO

LED	Arduino UNO
Red	D8
Yellow	D9
Green	D10
GND	GND

Buzzer → Arduino UNO

Buzzer Arduino UNO

+	D11
–	GND

Relay Module → Arduino UNO

Relay Arduino UNO

VCC	5V
GND	GND
IN	D7

Pump → Relay

- Relay **COM** → Pump power supply +
- Relay **NO** → Pump +
- Pump – → Pump power supply –

⚠️ Use a separate suitable power supply for the pump. Do not power the pump directly from the Arduino.

Working

Water Level	LED	Buzzer	Pump
Low	🟢 Green	OFF	ON
Medium	🟡 Yellow	OFF	ON
High	🔴 Red	ON	OFF

The system monitors the water level continuously. When the tank reaches a **high level**, the pump is automatically switched OFF and the buzzer gives an alert. When the water level is low, the pump turns ON.

Arduino Code

```
#define WATER_SENSOR A0

#define RELAY_PIN 7
#define RED_LED 8
#define YELLOW_LED 9
#define GREEN_LED 10
#define BUZZER 11

int waterLevel;

void setup() {
  pinMode(WATER_SENSOR, INPUT);

  pinMode(RELAY_PIN, OUTPUT);
  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  // Pump OFF initially
  digitalWrite(RELAY_PIN, HIGH);

  Serial.begin(9600);
}
```

```
void loop() {

  waterLevel = analogRead(WATER_SENSOR);

  Serial.print("Water Level: ");
  Serial.println(waterLevel);

  // LOW WATER LEVEL
  if (waterLevel < 300) {

    digitalWrite(GREEN_LED, HIGH);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(RED_LED, LOW);

    digitalWrite(BUZZER, LOW);

    // Pump ON
    digitalWrite(RELAY_PIN, LOW);

    Serial.println("Water Level LOW - Pump ON");
  }

  // MEDIUM WATER LEVEL
  else if (waterLevel >= 300 && waterLevel < 600) {

    digitalWrite(GREEN_LED, LOW);
```

```
  digitalWrite(YELLOW_LED, HIGH);
  digitalWrite(RED_LED, LOW);

  digitalWrite(BUZZER, LOW);

  // Pump remains ON
  digitalWrite(RELAY_PIN, LOW);

  Serial.println("Water Level MEDIUM - Pump ON");
}

// HIGH WATER LEVEL
else {

  digitalWrite(GREEN_LED, LOW);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(RED_LED, HIGH);

  digitalWrite(BUZZER, HIGH);

  // Pump OFF
  digitalWrite(RELAY_PIN, HIGH);

  Serial.println("Water Level HIGH - Pump OFF");
}
```

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

Note

The values **300 and 600 are example thresholds**. Check your water-level sensor readings in the Serial Monitor and adjust these values according to your sensor and tank.

Relay note: The code assumes an **active-LOW relay** (LOW = ON, HIGH = OFF). If your relay works oppositely, reverse these relay values.

Project Name: Temperature Controlled Car

- ❖ **Components Required**

1. Arduino UNO
2. DHT11 Temperature & Humidity Sensor
3. 0.96" OLED Display (I2C)
4. 1-Channel Relay Module
5. DC Fan / Motor
6. External power supply
7. Breadboard
8. Jumper wires

- ❖ **Connections**

DHT11 → Arduino UNO

DHT11	Arduino UNO
VCC	5V
GND	GND
DATA	D2

OLED Display → Arduino UNO

OLED	Arduino UNO
VCC	5V
GND	GND
SDA	A4
SCL	A5

Relay → Arduino UNO

Relay Arduino UNO

VCC 5V

GND GND

IN D7

Fan/Motor → Relay

- Relay **COM** → External supply +
- Relay **NO** → Fan/Motor +
- Fan/Motor – → External supply –

Working

The DHT11 continuously measures the temperature.

- **Temperature below 30°C** → Fan OFF
- **Temperature 30°C or above** → Relay ON → Fan ON
- OLED displays the **temperature and humidity**.

Arduino Code

```
#include <DHT.h>

#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>


#define DHT_PIN 2

#define DHT_TYPE DHT11


#define RELAY_PIN 7


#define SCREEN_WIDTH 128
```

```
#define SCREEN_HEIGHT 64
#define OLED_RESET -1

DHT dht(DHT_PIN, DHT_TYPE);

Adafruit_SSD1306 display(
  SCREEN_WIDTH,
  SCREEN_HEIGHT,
  &Wire,
  OLED_RESET
);

float temperature;
float humidity;

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

  dht.begin();

  pinMode(RELAY_PIN, OUTPUT);

  // Relay OFF initially
  digitalWrite(RELAY_PIN, HIGH);

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
```

```
  Serial.println("OLED not found!");
  while (1);
 }

 display.clearDisplay();
 display.setTextColor(SSD1306_WHITE);

 display.setTextSize(1);
 display.setCursor(20, 25);
 display.println("Temperature Car");

 display.display();
 delay(2000);
}

void loop() {

 temperature = dht.readTemperature();
 humidity = dht.readHumidity();

 if (isnan(temperature) || isnan(humidity)) {
  Serial.println("DHT11 Error");
  return;
 }

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

```
Serial.print(temperature);
Serial.print(" C  Humidity: ");
Serial.print(humidity);
Serial.println(" %");

// Temperature control
if (temperature >= 30) {
  digitalWrite(RELAY_PIN, LOW);

  Serial.println("Temperature HIGH - Fan ON");
}
else {
  digitalWrite(RELAY_PIN, HIGH);

  Serial.println("Temperature NORMAL - Fan OFF");
}

// OLED display
display.clearDisplay();

display.setTextSize(1);
display.setCursor(0, 0);
display.println("TEMPERATURE CONTROL");

display.setTextSize(2);
display.setCursor(0, 18);
```

```
  display.print(temperature, 1);
  display.println(" C");

  display.setTextSize(1);
  display.setCursor(0, 45);
  display.print("Humidity: ");
  display.print(humidity, 0);
  display.println(" %");

  display.setCursor(80, 45);

  if (temperature >= 30) {
    display.println("FAN ON");
  }
  else {
    display.println("FAN OFF");
  }

  display.display();

  delay(2000);
}
```

Note

The **30°C threshold** can be changed in the code according to your project requirement.

Project Name: **Motion Activated Light**

- **Components Required**

1. Arduino UNO
2. IR Sensor Module
3. Traffic Light LED Module (Red, Yellow, Green)
4. Jumper wires
5. Breadboard
6. USB cable

- **Connections**
- **IR Sensor → Arduino UNO**

IR Sensor Arduino UNO

VCC 5V

GND GND

OUT D2

- **Traffic Light Module → Arduino UNO**

LED Arduino UNO

Red D8

Yellow D9

Green D10

GND GND

Working

- **No motion detected** → Green LED ON.
- **Motion detected** → Red LED ON and Yellow/Green LEDs OFF.
- The IR sensor detects a person/object passing in front of the sensor and activates the light indication.

- **Arduino Code**

```
#define IR_SENSOR 2

#define RED_LED 8
#define YELLOW_LED 9
#define GREEN_LED 10

void setup() {

  pinMode(IR_SENSOR, INPUT);

  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);

  Serial.begin(9600);
}

void loop() {

  int motion = digitalRead(IR_SENSOR);

  if (motion == LOW) {

    // Motion detected
    digitalWrite(RED_LED, HIGH);
```

```
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(GREEN_LED, LOW);

  Serial.println("Motion Detected - Light ON");
 }
 else {

  // No motion
  digitalWrite(RED_LED, LOW);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(GREEN_LED, HIGH);

  Serial.println("No Motion - Light OFF");
 }

 delay(200);
}
```

Note

Most IR obstacle sensor modules give **LOW when an object is detected**. If your sensor works in the opposite way, change LOW to HIGH in the if condition. Use the small potentiometer on the IR module to adjust the detection distance.

Project Name: Water Tank Auto Refill System

- **Components Required**

1. Arduino UNO
2. HC-SR04 Ultrasonic Sensor
3. 1-Channel Relay Module
4. DC Submersible Water Pump
5. Water Tank/Container
6. External power supply for pump
7. Jumper wires
8. Breadboard

- **Connections**

HC-SR04 → Arduino UNO

HC-SR04	Arduino UNO
VCC	5V
GND	GND
TRIG	D9
ECHO	D10

Relay → Arduino UNO

Relay	Arduino UNO
VCC	5V
GND	GND
IN	D7

Pump → Relay

- Relay **COM** → External pump supply +
- Relay **NO** → Pump +

- Pump – → External supply –

Working

The ultrasonic sensor measures the distance between the sensor and the water surface.

- **Water level low** → distance is large → **Pump ON**
- **Water level reaches the required level** → distance becomes small → **Pump OFF**
- The system automatically refills the tank whenever the water level becomes low.

Arduino Code

```
#define TRIG_PIN 9

#define ECHO_PIN 10

#define RELAY_PIN 7


long duration;

float distance;


// Adjust these values according to your tank

int LOW_LEVEL = 25;     // Pump ON

int FULL_LEVEL = 10;    // Pump OFF


void setup() {

  pinMode(TRIG_PIN, OUTPUT);

  pinMode(ECHO_PIN, INPUT);


  pinMode(RELAY_PIN, OUTPUT);


  // Pump OFF initially

  digitalWrite(RELAY_PIN, HIGH);
```

```
  Serial.begin(9600);
}

void loop() {

  // Send ultrasonic pulse
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Read echo
  duration = pulseIn(ECHO_PIN, HIGH);

  // Calculate distance
  distance = duration * 0.034 / 2;

  Serial.print("Water Surface Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Water level LOW
  if (distance >= LOW_LEVEL) {
```

```
    // Pump ON
    digitalWrite(RELAY_PIN, LOW);

    Serial.println("Water Level LOW - Pump ON");
  }

  // Tank FULL
  else if (distance <= FULL_LEVEL) {

    // Pump OFF
    digitalWrite(RELAY_PIN, HIGH);

    Serial.println("Tank FULL - Pump OFF");
  }

  delay(1000);
}
```

Note

Mount the HC-SR04 **at the top of the tank facing downward**. The distance measured by the sensor decreases as the water level rises.

Project Name: Fire Protection System

- **Components Required**

1. Arduino UNO
2. Flame Sensor
3. 1-Channel Relay Module
4. Submersible Water Pump
5. Water container
6. External power supply for pump
7. Jumper wires
8. Breadboard

- **Connections**

Flame Sensor → Arduino UNO

Flame Sensor	Arduino UNO
VCC	5V
GND	GND
DO	D2

Relay → Arduino UNO

Relay	Arduino UNO
VCC	5V
GND	GND
IN	D7

Water Pump → Relay

- Relay **COM** → External pump supply +
- Relay **NO** → Pump +
- Pump – → External supply –

Working

- **No fire detected** → Pump OFF.
- **Fire detected** → Flame sensor sends a signal → Relay turns ON → Water pump starts.
- The pump sprays water to help extinguish the detected fire.

Arduino Code

```
#define FLAME_SENSOR 2
#define RELAY_PIN 7

void setup() {
  pinMode(FLAME_SENSOR, INPUT);
  pinMode(RELAY_PIN, OUTPUT);

  // Pump OFF initially
  digitalWrite(RELAY_PIN, HIGH);

  Serial.begin(9600);
}

void loop() {

  int flameStatus = digitalRead(FLAME_SENSOR);

  if (flameStatus == LOW) {

    // Fire detected
    digitalWrite(RELAY_PIN, LOW);
```

```
    Serial.println("FIRE DETECTED - WATER PUMP ON");
  }
  else {

    // No fire
    digitalWrite(RELAY_PIN, HIGH);

    Serial.println("NO FIRE - WATER PUMP OFF");
  }

  delay(500);
}
```

Note

Most flame sensor modules give **LOW when a flame is detected**. If your sensor gives HIGH during detection, change LOW to HIGH in the if condition.

Project Name : Smart Garden System

- **Components Required**

1. Arduino UNO
2. Soil Moisture Sensor
3. DHT11 Temperature & Humidity Sensor
4. 1-Channel Relay Module
5. DC Water Pump
6. Water Container
7. External Power Supply for Pump
8. Breadboard
9. Jumper Wires
10. USB Cable

- **Connections**

Soil Moisture Sensor

- VCC → 5V
- GND → GND
- AO → A0

DHT11

- VCC → 5V
- GND → GND
- DATA → D2

Relay Module

- VCC → 5V
- GND → GND
- IN → D7

Water Pump through Relay

- Relay **COM** → External supply +
- Relay **NO** → Pump +
- Pump – → External supply –

Important: Do not power the water pump directly from the Arduino 5V pin.

Working

- Soil is **dry** → Pump turns **ON**.
- Soil is sufficiently moist → Pump turns **OFF**.
- DHT11 measures and displays temperature and humidity in the Serial Monitor.

Arduino Code

```
#include <DHT.h>

#define SOIL_SENSOR A0
#define DHT_PIN 2
#define RELAY_PIN 7

#define DHT_TYPE DHT11

DHT dht(DHT_PIN, DHT_TYPE);

int soilValue;
float temperature;
float humidity;

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

```
  dht.begin();

  pinMode(SOIL_SENSOR, INPUT);
  pinMode(RELAY_PIN, OUTPUT);

  // Relay OFF initially
  digitalWrite(RELAY_PIN, HIGH);

  Serial.println("SMART GARDEN SYSTEM");
  Serial.println("--------------------");
}

void loop() {

  soilValue = analogRead(SOIL_SENSOR);

  temperature = dht.readTemperature();
  humidity = dht.readHumidity();

  Serial.print("Soil Moisture: ");
  Serial.println(soilValue);

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" C");
```

```
Serial.print("Humidity: ");
Serial.print(humidity);
Serial.println(" %");

// Soil is dry
if (soilValue > 700) {

  digitalWrite(RELAY_PIN, LOW);  // Pump ON

  Serial.println("Soil is DRY");
  Serial.println("Water Pump: ON");

}

// Soil is moist/wet
else {

  digitalWrite(RELAY_PIN, HIGH); // Pump OFF

  Serial.println("Soil is MOIST/WET");
  Serial.println("Water Pump: OFF");
}

Serial.println("--------------------");

delay(2000);
```

}

Note: The 700 value is an example threshold. Check the soil sensor value in the Serial Monitor for dry and wet soil and adjust the value if required. Most relay modules used with Arduino are **active LOW**; if your relay works opposite, the ON/OFF logic can be reversed.

Project Name : Smart Security Alarm

- **Components Required**

1. Arduino UNO
2. HC-SR04 Ultrasonic Sensor
3. Buzzer
4. Traffic Light Module (Red/Yellow/Green)
5. Breadboard
6. Jumper Wires
7. USB Cable

- **Connections**

HC-SR04 Ultrasonic Sensor

- VCC → 5V
- GND → GND
- TRIG → D9
- ECHO → D10

Buzzer

- → D11
- → GND

Traffic Light Module

- Red → D8
- Yellow → D7
- Green → D6
- GND → GND

Working

- If an object is **more than 50 cm away** → Green LED ON, alarm OFF.
- If an object is **between 20–50 cm** → Yellow LED ON, alarm OFF.

- If an object is **less than 20 cm** → Red LED ON and buzzer ON.

Arduino Code

```
#define TRIG_PIN 9

#define ECHO_PIN 10


#define BUZZER 11


#define RED_LED 8

#define YELLOW_LED 7

#define GREEN_LED 6


long duration;

float distance;


void setup() {

  Serial.begin(9600);


  pinMode(TRIG_PIN, OUTPUT);

  pinMode(ECHO_PIN, INPUT);


  pinMode(BUZZER, OUTPUT);


  pinMode(RED_LED, OUTPUT);

  pinMode(YELLOW_LED, OUTPUT);

  pinMode(GREEN_LED, OUTPUT);
```

```
  digitalWrite(BUZZER, LOW);
}

void loop() {

  // Send ultrasonic pulse
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIG_PIN, LOW);

  // Read echo
  duration = pulseIn(ECHO_PIN, HIGH);

  // Calculate distance
  distance = duration * 0.034 / 2;

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  // Object is very close
  if (distance < 20) {
```

```
  digitalWrite(RED_LED, HIGH);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(GREEN_LED, LOW);

  digitalWrite(BUZZER, HIGH);

  Serial.println("SECURITY ALERT!");

}

// Object is at medium distance
else if (distance >= 20 && distance <= 50) {

  digitalWrite(RED_LED, LOW);
  digitalWrite(YELLOW_LED, HIGH);
  digitalWrite(GREEN_LED, LOW);

  digitalWrite(BUZZER, LOW);

  Serial.println("WARNING");

}

// Area is safe
else {
```

```
   digitalWrite(RED_LED, LOW);
   digitalWrite(YELLOW_LED, LOW);
   digitalWrite(GREEN_LED, HIGH);

   digitalWrite(BUZZER, LOW);

   Serial.println("AREA SAFE");
  }

  delay(300);
}
```

Note: The 20 cm and 50 cm distances can be changed according to the required security range.

Project name : Automatic Night Security Light

- ❖ **Components Required**

1. Arduino UNO
2. IR Sensor Module
3. Traffic Light Module (Red/Yellow/Green)
4. Breadboard
5. Jumper Wires
6. USB Cable

- ❖ **Connections**

IR Sensor

- VCC → 5V
- GND → GND
- OUT → D2

Traffic Light Module

- Red → D8
- Yellow → D9
- Green → D10
- GND → GND

Working

- **No motion detected** → Green LED ON.
- **Motion detected at night** → Red LED ON, indicating security activity.
- Yellow LED remains OFF.

Code

```
#define IR_SENSOR 2


#define RED_LED 8
```

```
#define YELLOW_LED 9
#define GREEN_LED 10

void setup() {
  pinMode(IR_SENSOR, INPUT);

  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);

  Serial.begin(9600);
}

void loop() {

  int sensorState = digitalRead(IR_SENSOR);

  if (sensorState == LOW) {
    // Motion detected
    digitalWrite(RED_LED, HIGH);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(GREEN_LED, LOW);

    Serial.println("MOTION DETECTED - SECURITY ALERT");
  }
  else {
```

```
  // No motion
  digitalWrite(RED_LED, LOW);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(GREEN_LED, HIGH);

  Serial.println("NO MOTION - AREA SAFE");
 }

 delay(300);
}
```

Note: Most IR obstacle sensor modules give **LOW when an object is detected**. If your sensor works opposite, change LOW to HIGH in the if condition.

Environmental Monitoring Station

- **Components Required**

1. Arduino UNO
2. DHT11 Temperature & Humidity Sensor
3. Rain Sensor Module
4. 0.96" OLED I2C Display (128×64)
5. Traffic Light Module (Red/Yellow/Green)
6. Breadboard
7. Jumper Wires
8. USB Cable

- **Connections**

DHT11

- VCC → 5V
- GND → GND
- DATA → D2

Rain Sensor

- VCC → 5V
- GND → GND
- DO → D3

OLED Display

- VCC → 5V
- GND → GND
- SDA → A4
- SCL → A5

Traffic Light Module

- Red → D8

- Yellow → D9
- Green → D10
- GND → GND

Working

- OLED displays **temperature and humidity** from the DHT11.
- When **rain is detected** → Red LED turns ON.
- When there is **no rain** → Green LED turns ON.
- The OLED also displays the rain status.

Code

```
#include <Wire.h>

#include <Adafruit_GFX.h>

#include <Adafruit_SSD1306.h>

#include <DHT.h>


#define SCREEN_WIDTH 128

#define SCREEN_HEIGHT 64

#define OLED_RESET -1


Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);


#define DHT_PIN 2

#define RAIN_SENSOR 3


#define RED_LED 8

#define YELLOW_LED 9

#define GREEN_LED 10
```

```
#define DHT_TYPE DHT11

DHT dht(DHT_PIN, DHT_TYPE);

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

  dht.begin();

  pinMode(RAIN_SENSOR, INPUT);

  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println("OLED not found!");
    while (1);
  }

  display.clearDisplay();
  display.setTextColor(SSD1306_WHITE);

  digitalWrite(RED_LED, LOW);
  digitalWrite(YELLOW_LED, LOW);
```

```
  digitalWrite(GREEN_LED, HIGH);
}

void loop() {

  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  int rainStatus = digitalRead(RAIN_SENSOR);

  display.clearDisplay();

  display.setTextSize(1);
  display.setCursor(0, 0);
  display.println("ENVIRONMENT STATION");

  display.setCursor(0, 18);
  display.print("Temp: ");
  display.print(temperature);
  display.println(" C");

  display.setCursor(0, 32);
  display.print("Humidity: ");
  display.print(humidity);
  display.println(" %");
```

```
if (rainStatus == LOW) {

  // Rain detected
  digitalWrite(RED_LED, HIGH);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(GREEN_LED, LOW);

  display.setCursor(0, 48);
  display.println("Rain: DETECTED");

  Serial.println("RAIN DETECTED");

} else {

  // No rain
  digitalWrite(RED_LED, LOW);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(GREEN_LED, HIGH);

  display.setCursor(0, 48);
  display.println("Rain: NO RAIN");

  Serial.println("NO RAIN");
}

display.display();
```

```
  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" C | Humidity: ");
  Serial.print(humidity);
  Serial.print(" % | Rain: ");
  Serial.println(rainStatus == LOW ? "Detected" : "No Rain");


  delay(2000);
}
```

Note: Most rain sensor modules give **LOW when rain/water is detected**. If your module works opposite, change rainStatus == LOW to rainStatus == HIGH. The sensor module's potentiometer can be used to adjust rain sensitivity.

Project Name: **Obstacle Avoiding Car**

- **Components Required**

1. Arduino UNO
2. HC-SR04 Ultrasonic Sensor
3. L298N Motor Driver
4. BO Motors ×2
5. Robot Car Chassis
6. Wheels ×2
7. Caster Wheel
8. Battery
9. Jumper Wires

- **Connections**

HC-SR04 Ultrasonic Sensor

- VCC → 5V
- GND → GND
- TRIG → D9
- ECHO → D10

L298N Motor Driver

- IN1 → D4
- IN2 → D5
- IN3 → D6
- IN4 → D7
- GND → Arduino GND
- OUT1 & OUT2 → Left BO Motor
- OUT3 & OUT4 → Right BO Motor
- Motor power → External Battery

Important: Arduino GND and L298N GND must be connected together.

Working

- The ultrasonic sensor continuously measures the distance in front of the car.
- If the path is clear, the car moves **forward**.
- If an obstacle is detected within **20 cm**, the car stops, moves backward briefly, and turns.
- The car then continues moving forward.

Code

```
#define TRIG_PIN 9
#define ECHO_PIN 10

#define IN1 4
#define IN2 5
#define IN3 6
#define IN4 7

long duration;
int distance;

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

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
```

```
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
}

void loop() {

  distance = getDistance();

  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  if (distance > 20) {
    // Path clear
    forward();
  }
  else {
    // Obstacle detected
    stopCar();
    delay(300);

    backward();
    delay(400);

    stopCar();
    delay(200);
```

```
    // Turn right
    turnRight();
    delay(500);

    stopCar();
    delay(200);
  }

  delay(50);
}

// Measure distance
int getDistance() {

  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);

  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);

  digitalWrite(TRIG_PIN, LOW);

  duration = pulseIn(ECHO_PIN, HIGH);

  if (duration == 0) {
```

```
    return 400;
  }

  return duration * 0.034 / 2;
}

// Move forward
void forward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);

  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
}

// Move backward
void backward() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);

  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}

// Turn right
void turnRight() {
```

```
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);

  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
}

// Stop
void stopCar() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);

  digitalWrite(IN3, LOW);
  digitalWrite(IN4, LOW);
}
```

Note: If either BO motor rotates in the wrong direction, swap the two wires of that motor at the L298N output terminals. You can also change the 20 cm obstacle distance according to your requirement.