Project Name: Smart Thermometer

Required Components

1. Arduino UNO
2. DHT11 Temperature Sensor
3. 0.96" OLED Display (I2C, 128×64)
4. Breadboard
5. Jumper Wires
6. USB Cable

Simple Connection

Component	Pin	Arduino UNO
DHT11	VCC	5V
	GND	GND
	DATA	D2
OLED	VCC	5V
	GND	GND
	SDA	A4
	SCL	A5

Arduino Code

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>

#define DHTPIN 2
```

```
#define DHTTYPE DHT11

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
DHT dht(DHTPIN, DHTTYPE);

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

 dht.begin();

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

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

 display.setTextSize(2);
 display.setCursor(20, 10);
 display.println("SMART");

 display.setCursor(15, 35);
 display.println("THERMOMETER");
```

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

void loop() {
  float temperature = dht.readTemperature();

  if (isnan(temperature)) {
    display.clearDisplay();
    display.setTextSize(2);
    display.setCursor(10, 25);
    display.println("Sensor Error");
    display.display();
    delay(1000);
    return;
  }

  display.clearDisplay();

  display.setTextSize(1);
  display.setCursor(30, 5);
  display.println("TEMPERATURE");

  display.setTextSize(3);
  display.setCursor(15, 25);
  display.print(temperature, 1);

  display.setTextSize(2);
```

```
  display.print(" C");

  display.display();

  delay(2000);
}
```

Required Arduino libraries: DHT sensor library, Adafruit GFX Library, and Adafruit SSD1306.

Project Name: Distance Meter

Required Components

1. **Arduino UNO**
2. **HC-SR04 Ultrasonic Sensor**
3. **0.96" OLED Display (I2C, 128×64)**
4. **Breadboard**
5. **Jumper Wires**
6. **USB Cable**

Simple Connection

Component	Pin	Arduino UNO
HC-SR04	VCC	5V
	GND	GND
	TRIG	D9
	ECHO	D10
OLED	VCC	5V
	GND	GND
	SDA	A4
	SCL	A5

Arduino Code

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
```

```
#define TRIG_PIN 9
#define ECHO_PIN 10

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  Serial.begin(9600);

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    while (1);
  }

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

  display.setTextSize(2);
  display.setCursor(25, 10);
  display.println("DISTANCE");

  display.setCursor(35, 35);
  display.println("METER");

  display.display();
  delay(2000);
```

```
}

void loop() {
  long duration;
  float distance;

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

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

  duration = pulseIn(ECHO_PIN, HIGH);

  distance = duration * 0.0343 / 2;

  display.clearDisplay();

  display.setTextSize(1);
  display.setCursor(35, 5);
  display.println("DISTANCE");

  display.setTextSize(3);
  display.setCursor(15, 25);
  display.print(distance, 1);

  display.setTextSize(2);
```

```
  display.print(" cm");

  display.display();

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

  delay(500);
}
```

Project Name: Automatic Street Light

Required Components

1. **Arduino UNO**
2. **LDR (Light Dependent Resistor)**
3. **Traffic Light Module (Red, Yellow, Green LEDs)**
4. **Breadboard**
5. **Jumper Wires**
6. **10kΩ Resistor**

Simple Connection

Component	Pin	Arduino UNO
LDR	One leg	5V
	Other leg	A0
10kΩ Resistor	One leg	A0
	Other leg	GND
Traffic Light Module	Red	D8
	Yellow	D9
	Green	D10
	GND	GND

Arduino Code

```
#define LDR_PIN A0

#define RED_LED 8

#define YELLOW_LED 9

#define GREEN_LED 10
```

```
void setup() {
  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  int lightValue = analogRead(LDR_PIN);

  Serial.println(lightValue);

  // Dark condition
  if (lightValue < 500) {
    digitalWrite(RED_LED, HIGH);
    digitalWrite(YELLOW_LED, HIGH);
    digitalWrite(GREEN_LED, HIGH);
  }
  // Bright condition
  else {
    digitalWrite(RED_LED, LOW);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(GREEN_LED, LOW);
  }
delay(500);
```

Project Name: Fire Detection System

Required Components

1. **Arduino UNO**
2. **Flame Sensor**
3. **Traffic Light Module (Red, Yellow, Green)**
4. **Buzzer**
5. **Breadboard**
6. **Jumper Wires**

Simple Connection

Component	Pin	Arduino UNO
Flame Sensor	VCC	5V
	GND	GND
	DO	D2
Traffic Light Module	Red	D8
	Yellow	D9
	Green	D10
	GND	GND
Buzzer	+	D11
	–	GND

Arduino Code

```
#define FLAME_SENSOR 2

#define RED_LED 8

#define YELLOW_LED 9

#define GREEN_LED 10
```

```
#define BUZZER 11
void setup() {
  pinMode(FLAME_SENSOR, INPUT);

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

  Serial.begin(9600);
}

void loop() {
  int flame = digitalRead(FLAME_SENSOR);

  // Flame detected
  if (flame == LOW) {
    digitalWrite(RED_LED, HIGH);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(GREEN_LED, LOW);

    digitalWrite(BUZZER, HIGH);

    Serial.println("FIRE DETECTED!");
  }
  // No flame
```

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

    digitalWrite(BUZZER, LOW);

    Serial.println("No Fire");
  }

  delay(200);
}
```

Note: Most flame sensor modules give LOW when a flame is detected. If your module works opposite, change flame == LOW to flame == HIGH.

Project Name: Water Level Monitor

Required Components

1. **Arduino UNO**
2. **Water Level Sensor**
3. **Traffic Light Module (Red, Yellow, Green)**
4. **Breadboard**
5. **Jumper Wires**

Simple Connection

Component	Pin	Arduino UNO
Water Level Sensor	VCC	5V
	GND	GND
	SIG	A0
Traffic Light Module	Red	D8
	Yellow	D9
	Green	D10
	GND	GND

Arduino Code

```
#define WATER_SENSOR A0
#define RED_LED 8
#define YELLOW_LED 9
#define GREEN_LED 10
void setup() {
  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
```

```
  Serial.begin(9600);
}

void loop() {
  int 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);
  }

  // Medium water level
  else if (waterLevel < 600) {
    digitalWrite(GREEN_LED, LOW);
    digitalWrite(YELLOW_LED, HIGH);
    digitalWrite(RED_LED, LOW);
  }

  // High water level
  else {
```

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


  delay(500);
}
```

Note

Sensor values may vary depending on the water level sensor and the type of water/container used. Check the values in the Serial Monitor and adjust the values 300 and 600 in the code to get the desired Low / Medium / High levels.

Project Name: Soil Moisture Monitor

Required Components

1. **Arduino UNO**
2. **Soil Moisture Sensor**
3. **Traffic Light Module (Red, Yellow, Green)**
4. **Buzzer**
5. **Breadboard**
6. **Jumper Wires**

Simple Connection

Component	Pin	Arduino UNO
Soil Moisture Sensor	VCC	5V
	GND	GND
	AO	A0
Traffic Light Module	Red	D8
	Yellow	D9
	Green	D10
	GND	GND
Buzzer	+	D11
	–	GND

Arduino Code

```
#define SOIL_SENSOR A0


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

```
#define BUZZER 11

void setup() {
  pinMode(RED_LED, OUTPUT);
  pinMode(YELLOW_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(BUZZER, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  int soilValue = analogRead(SOIL_SENSOR);

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

  // Dry Soil
  if (soilValue > 700) {
    digitalWrite(RED_LED, HIGH);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(GREEN_LED, LOW);

    digitalWrite(BUZZER, HIGH);
  }
```

```
 // Moist Soil
 else if (soilValue > 400) {
  digitalWrite(RED_LED, LOW);
  digitalWrite(YELLOW_LED, HIGH);
  digitalWrite(GREEN_LED, LOW);

  digitalWrite(BUZZER, LOW);
 }

 // Wet Soil
 else {
  digitalWrite(RED_LED, LOW);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(GREEN_LED, HIGH);

  digitalWrite(BUZZER, LOW);
 }

 delay(500);
}
```

Note

The sensor values can vary depending on the soil type and sensor. Check the value in the Serial Monitor for dry and wet soil, then adjust 700 and 400 in the code according to your requirement.

Project Name: Rain Detection Monitor

Required Components

1. **Arduino UNO**
2. **Rain Sensor Module**
3. **Traffic Light Module (Red, Yellow, Green)**
4. **Breadboard**
5. **Jumper Wires**

Simple Connection

Component	Pin	Arduino UNO
Rain Sensor	VCC	5V
	GND	GND
	DO	D2
Traffic Light Module	Red	D8
	Yellow	D9
	Green	D10
	GND	GND

Arduino Code

```
#define RAIN_SENSOR 2


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


void setup() {
  pinMode(RAIN_SENSOR, INPUT);
```

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

  Serial.begin(9600);
}

void loop() {
  int rainStatus = digitalRead(RAIN_SENSOR);

  // Rain detected
  if (rainStatus == LOW) {
    digitalWrite(RED_LED, HIGH);
    digitalWrite(YELLOW_LED, LOW);
    digitalWrite(GREEN_LED, LOW);

    Serial.println("RAIN DETECTED");
  }

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

```
    Serial.println("NO RAIN");
  }


  delay(500);
}
```

Note

The rain sensor module usually gives LOW when water/rain is detected. If your sensor works opposite, change rainStatus == LOW to rainStatus == HIGH. You can also adjust the sensor's onboard potentiometer to change its rain-detection sensitivity.

Project Name : Oled Display Sensor

1. Rain Sensor → OLED

Connections

Component	Arduino UNO
Rain VCC	5V
Rain GND	GND
Rain DO	D2
OLED VCC	5V
OLED GND	GND
OLED SDA	A4
OLED SCL	A5

Rain Sensor + OLED Code

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1

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

#define RAIN_SENSOR 2

void setup() {
```

```
  pinMode(RAIN_SENSOR, INPUT);

  Serial.begin(9600);

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

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

  display.setTextSize(2);
  display.setCursor(15, 5);
  display.println("RAIN");

  display.setTextSize(1);
  display.setCursor(25, 30);
  display.println("Sensor Starting...");

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

void loop() {
```

```
int rainStatus = digitalRead(RAIN_SENSOR);

display.clearDisplay();

display.setTextColor(SSD1306_WHITE);

display.setTextSize(2);
display.setCursor(15, 0);
display.println("RAIN");

display.setTextSize(1);
display.setCursor(0, 22);
display.println("----------------");

display.setTextSize(1);
display.setCursor(0, 32);
display.print("Status: ");

if (rainStatus == LOW) {
  display.setCursor(0, 45);
  display.setTextSize(2);
  display.println("RAIN!");

  Serial.println("RAIN DETECTED");
}
else {
```

```
  display.setCursor(0, 45);
  display.setTextSize(2);
  display.println("NO RAIN");

  Serial.println("NO RAIN");
 }

 display.display();

 delay(500);
}
```

Note: Most rain sensor modules give LOW when rain/water is detected. If yours works opposite, change LOW and HIGH in the code.

2. Ultrasonic Sensor → OLED

For an ultrasonic sensor, showing the distance in cm is much better.

Connections

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

Component	Arduino UNO
OLED SDA	A4
OLED SCL	A5

Ultrasonic + OLED Code

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1

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

#define TRIG_PIN 9
#define ECHO_PIN 10

long duration;
float distance;

void setup() {

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

```
  Serial.begin(9600);

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

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

  display.setTextSize(2);
  display.setCursor(5, 5);
  display.println("DISTANCE");

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

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, 30000);

// Calculate distance
distance = duration * 0.0343 / 2;

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

// Title
display.setTextSize(1);
display.setCursor(20, 0);
display.println("DISTANCE METER");

display.drawLine(0, 12, 127, 12, SSD1306_WHITE);

// Distance
display.setTextSize(1);
display.setCursor(35, 20);
display.println("Distance:");

display.setTextSize(2);
display.setCursor(35, 32);
```

```
if (duration == 0) {
  display.println("---");
}
else {
  display.print((int)distance);
  display.println(" cm");
}

// Object status
display.setTextSize(1);
display.setCursor(30, 54);

if (duration == 0) {
  display.println("No Object");
}
else if (distance < 20) {
  display.println("Very Close!");
}
else if (distance < 100) {
  display.println("Object Nearby");
}
else {
  display.println("Object Far");
}
```

```
  display.display();


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


  delay(500);
}
```

Project Name: Multi-Sensor Dashboard

Sensors

1. DHT11 → Temperature + Humidity
2. Soil Moisture Sensor → Soil Moisture Status + Value
3. Flame Sensor → Fire Status

Components Required

Component	Quantity
Arduino UNO	1
0.96" OLED 128×64 I2C	1
DHT11 Sensor	1
Soil Moisture Sensor	1
Flame Sensor Module	1
Breadboard	1
Jumper Wires	As required
USB Cable	1

Connections

DHT11

DHT11	Arduino UNO
VCC	5V
GND	GND
DATA	D2

Soil Moisture Sensor

Soil Sensor	Arduino UNO
VCC	5V

Soil Sensor Arduino UNO

GND	**GND**
AO	**A0**

Flame Sensor

Flame Sensor Arduino UNO

VCC	**5V**
GND	**GND**
DO	**D3**

OLED

OLED Arduino UNO

VCC	**5V**
GND	**GND**
SDA	**A4**
SCL	**A5**

Install these libraries in Arduino IDE:

- **DHT sensor library**
- **Adafruit GFX Library**
- **Adafruit SSD1306**

Arduino 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);

// DHT11
#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

// Soil Moisture
#define SOIL_PIN A0

// Flame Sensor
#define FLAME_PIN 3

void setup() {

  Serial.begin(9600);

  dht.begin();

  pinMode(FLAME_PIN, INPUT);
```

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

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

 display.setTextSize(2);
 display.setCursor(15, 20);
 display.println("STARTING");

 display.display();

 delay(2000);
}

void loop() {

 // Read sensors
 float temperature = dht.readTemperature();
 float humidity = dht.readHumidity();

 int soilValue = analogRead(SOIL_PIN);

 int flameStatus = digitalRead(FLAME_PIN);
```

```
// ----------------------------
// SCREEN 1 - DHT11
// ----------------------------

display.clearDisplay();

display.setTextColor(SSD1306_WHITE);

display.setTextSize(1);
display.setCursor(20, 0);
display.println("MULTI SENSOR");

display.drawLine(0, 12, 127, 12, SSD1306_WHITE);

display.setCursor(0, 20);
display.println("TEMPERATURE:");

display.setTextSize(2);
display.setCursor(0, 30);

if (isnan(temperature)) {
  display.println("ERROR");
} else {
  display.print(temperature, 1);
  display.println(" C");
```

```
}

display.setTextSize(1);
display.setCursor(0, 52);
display.print("Humidity: ");

if (isnan(humidity)) {
  display.println("ERROR");
} else {
  display.print(humidity, 0);
  display.println(" %");
}

display.display();

delay(3000);
// ----------------------------
// SCREEN 2 - SOIL MOISTURE
// ----------------------------

display.clearDisplay();

display.setTextSize(1);
display.setCursor(20, 0);
display.println("MULTI SENSOR");
```

```
display.drawLine(0, 12, 127, 12, SSD1306_WHITE);

display.setCursor(0, 20);
display.println("SOIL MOISTURE");

display.setTextSize(2);
display.setCursor(25, 30);
display.print(soilValue);

display.setTextSize(1);
display.setCursor(0, 52);

if (soilValue > 700) {
  display.println("Status: DRY");
}
else if (soilValue > 400) {
  display.println("Status: MOIST");
}
else {
  display.println("Status: WET");
}

display.display();

delay(3000);
// ----------------------------
```

```
// SCREEN 3 - FLAME SENSOR
// ------------------------------

display.clearDisplay();

display.setTextSize(1);
display.setCursor(20, 0);
display.println("MULTI SENSOR");

display.drawLine(0, 12, 127, 12, SSD1306_WHITE);

display.setCursor(25, 22);
display.println("FLAME SENSOR");

display.setTextSize(2);
display.setCursor(20, 40);

if (flameStatus == LOW) {
  display.println("FIRE!");
  Serial.println("FIRE DETECTED");
}
else {
  display.println("SAFE");
  Serial.println("NO FIRE");
}
```

```
  display.display();

  delay(3000);


  // ----------------------------
  // Serial Monitor
  // ----------------------------

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.print(" C | Humidity: ");
  Serial.print(humidity);
  Serial.print(" % | Soil: ");
  Serial.print(soilValue);
  Serial.print(" | Flame: ");
  Serial.println(flameStatus);
}
```

Project Name: Smart Environment Monitor

This project combines temperature, humidity, light, and rain detection with a traffic light module and buzzer. The traffic light gives a quick visual indication of the environment.

Components Required

Component	Quantity
Arduino UNO	1
DHT11	1
LDR	1
Rain Sensor Module	1
Traffic Light Module	1
Buzzer	1
10kΩ Resistor	1
Breadboard	1
Jumper Wires	As required
USB Cable	1

Connections

DHT11

DHT11	Arduino UNO
VCC	5V
GND	GND
DATA	D2

LDR

LDR Circuit	Arduino UNO
LDR one leg	5V

LDR Circuit	Arduino UNO
LDR other leg	A0
10kΩ resistor	A0 → GND

Rain Sensor

Rain Sensor	Arduino UNO
VCC	5V
GND	GND
DO	D3

Traffic Light Module

Traffic Light	Arduino UNO
Red	D8
Yellow	D9
Green	D10
GND	GND

Buzzer

Buzzer	Arduino UNO
+	D11
–	GND

Arduino Code

Install the DHT sensor library before uploading.

```
#include <DHT.h>


#define DHTPIN 2
```

```
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

// Sensors
#define LDR_PIN A0
#define RAIN_PIN 3

// Traffic Light
#define RED_LED 8
#define YELLOW_LED 9
#define GREEN_LED 10

// Buzzer
#define BUZZER 11

void setup() {

  Serial.begin(9600);

  dht.begin();

  pinMode(RAIN_PIN, INPUT);

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

```
  pinMode(GREEN_LED, OUTPUT);

  pinMode(BUZZER, OUTPUT);

  // Start with green
  digitalWrite(GREEN_LED, HIGH);
  digitalWrite(YELLOW_LED, LOW);
  digitalWrite(RED_LED, LOW);
  digitalWrite(BUZZER, LOW);
}

void loop() {

  // Read DHT11
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Read LDR
  int lightValue = analogRead(LDR_PIN);

  // Read rain sensor
  int rainStatus = digitalRead(RAIN_PIN);

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

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

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

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

Serial.print("Light Value: ");
Serial.println(lightValue);

if (rainStatus == LOW) {
  Serial.println("Rain: DETECTED");
} else {
  Serial.println("Rain: NO RAIN");
}

// --------------------------------
// ENVIRONMENT DECISION
// --------------------------------

// Rain detected OR dark environment
if (rainStatus == LOW || lightValue < 400) {

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

```
  digitalWrite(BUZZER, HIGH);

  Serial.println("STATUS: ALERT");
}

// Moderate light
else if (lightValue < 700) {

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

  digitalWrite(BUZZER, LOW);

  Serial.println("STATUS: MODERATE");
}

// Normal environment
else {

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

  digitalWrite(BUZZER, LOW);
```

```
  Serial.println("STATUS: NORMAL");
 }


 delay(1000);
}
```

Important Note

The LDR values can vary depending on the room and lighting. Check the Serial Monitor and adjust these values if required:

lightValue < 400

lightValue < 700

Also, most rain sensor modules give LOW when rain is detected. If your sensor works opposite, change the rain condition accordingly.

Recommended project concept: 🟢 Normal → 🟡 Moderate → 🔴 Environmental Alert. This makes the project very easy for students to understand and demonstrate.

Project Name: Human Following Robot

Components Required

1. **Arduino UNO**
2. **HC-SR04 Ultrasonic Sensor × 1**
3. **Motor Driver L298N × 1**
4. **DC Geared Motors × 2**
5. **Robot chassis**
6. **Wheels × 2**
7. **Caster wheel × 1**
8. **Battery**
9. **Jumper wires**

Connections

HC-SR04 → Arduino UNO

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

L298N → Arduino UNO

L298N	Arduino UNO
IN1	D4
IN2	D5
IN3	D6
IN4	D7

L298N Arduino UNO

GND GND

Motors

- **Left motor → OUT1, OUT2**
- **Right motor → OUT3, OUT4**

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

Working Logic

With one front-mounted ultrasonic sensor:

- **Distance more than 80 cm → Robot moves forward.**
- **Distance 40–80 cm → Robot moves slowly/maintains following distance.**
- **Distance less than 40 cm → Robot stops.**
- **No person/object detected → Robot stops.**

Limitation: With only one ultrasonic sensor, the robot can follow a person in a straight direction, but it cannot reliably know whether the person moved left or right. For proper left/right human tracking, multiple sensors or a camera-based system is better.

Arduino Code

```
#define TRIG_PIN 9

#define ECHO_PIN 10


// L298N motor driver pins

#define IN1 4

#define IN2 5

#define IN3 6

#define IN4 7


long duration;
```

```
int distance;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);

  Serial.begin(9600);

  stopRobot();
}

void loop() {

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

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

```
// Receive echo
duration = pulseIn(ECHO_PIN, HIGH);

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

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

// Human following logic
if (distance > 80 && distance < 300) {
  // Person is far
  moveForward();
}
else if (distance >= 40 && distance <= 80) {
  // Person is at suitable distance
  moveSlow();
}
else if (distance < 40) {
  // Person is too close
  stopRobot();
}
else {
  // No object/person detected
  stopRobot();
```

```
  }

  delay(100);
}

// ---------------- MOTOR FUNCTIONS ----------------

void moveForward() {
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);

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

void moveSlow() {
  // Simple stop for maintaining distance
  // Can later be replaced with PWM speed control
  stopRobot();
}

void stopRobot() {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);

  digitalWrite(IN3, LOW);
```

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