

ARDUINO UNO 5V

Sensors, Outputs & Practical Projects

7 Basic Experiments + 4 Advanced IoT Projects

A beginner-friendly practical handbook for Arduino UNO. Every circuit uses the Arduino UNO's 5V logic and is designed for breadboard-based classroom experiments.

Platform: Arduino UNO R3 **Logic:** 5V

Programming: Arduino IDE

Safety: MQ-2 gas/smoke readings here are demonstration values, not calibrated ppm or certified safety thresholds.

1. Complete Materials Checklist

Item	Qty	How to identify / find it
Arduino UNO R3	1	USB-B port, 14 digital pins, 6 analog pins, 5V/3.3V/GND labels
Breadboard	1-2	White board with connected rows and power rails
USB A-to-B cable	1	Printer-style USB cable for computer connection
Jumper wires M-M	10+	Male pins on both ends
Jumper wires M-F	10+	Male pin on one end, female socket on the other
Jumper wires F-F	5+	Female sockets on both ends
5mm LED	4+	Long leg usually anode (+), short leg cathode (-)
220Ω resistor	4+	Current-limiting resistor for LEDs
Active buzzer 5V	2	Usually marked + and -
HC-SR04 ultrasonic	2	VCC, TRIG, ECHO, GND
MQ-2 gas/smoke module	1-2	Metal mesh sensor; VCC, GND, AO, DO
LDR module OR LDR	1	Light-sensitive sensor; module commonly has AO
10kΩ resistor	1	Needed for a bare LDR voltage divider
DHT11	1	Blue sensor/module; VCC, DATA, GND
HC-SR501 PIR	1	White Fresnel dome; VCC, OUT, GND

2. Standard Arduino UNO Pin Map

Pin	Use
D2	DHT11 DATA
D7	PIR OUT
D8	Buzzer
D9	HC-SR04 TRIG
D10	HC-SR04 ECHO
D13	LED / built-in LED
A0	MQ-2 analog output
A1	LDR analog output
5V	Sensor power
GND	Common ground

Always connect a common GND between Arduino and every module. Never connect an LED directly to an Arduino pin without a resistor.

To find parts, search electronics shops or online stores by the exact component names above: Arduino UNO R3, HC-SR04, MQ-2, DHT11, HC-SR501, 5mm LED, 220Ω resistor, active buzzer 5V.

3. Setup Before Every Experiment

Install Arduino IDE. Select **Tools → Board → Arduino UNO** and the correct USB/COM port. Upload with the Upload button. Open **Tools → Serial Monitor** and use **9600 baud** for the programs below.

Disconnect power before changing wiring. Recheck VCC, GND and signal pins before reconnecting the USB cable.

BASIC 1 — Blinking LED & Buzzer

Materials: Arduino UNO, LED, 220Ω resistor, active 5V buzzer, breadboard and wires.

Circuit: LED anode (long leg) → 220Ω → D13; LED cathode → GND. Buzzer + → D8; buzzer – → GND.

```
const int LED_PIN = 13;
const int BUZZER_PIN = 8;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_PIN, HIGH);
  digitalWrite(BUZZER_PIN, HIGH);
  delay(1000);

  digitalWrite(LED_PIN, LOW);
  digitalWrite(BUZZER_PIN, LOW);
  delay(1000);
}
```

Expected: LED and buzzer turn ON for one second and OFF for one second.

BASIC 2 — HC-SR04 Distance Measurement

Materials: Arduino UNO, HC-SR04, breadboard and wires.

Circuit: VCC → 5V; GND → GND; TRIG → D9; ECHO → D10.

```
const int TRIG_PIN = 9;
const int ECHO_PIN = 10;

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH, 30000);

  if (duration == 0) {
    Serial.println("No object detected / out of range");
  } else {
    float distance = duration * 0.0343 / 2.0;
    Serial.print("Distance: ");
    Serial.print(distance, 1);
    Serial.println(" cm");
  }

  delay(500);
}
```

Expected: Example Serial Monitor: Distance: 24.6 cm.

BASIC 3 — MQ-2 Gas Sensor

Materials: Arduino UNO, MQ-2 module, breadboard and wires.

Circuit: VCC → 5V; GND → GND; AO → A0. Leave DO unused.

```
const int GAS_PIN = A0;

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

void loop() {
  int gasValue = analogRead(GAS_PIN);

  Serial.print("Gas Sensor: ");
  Serial.print(gasValue);
  Serial.print(" | Status: ");

  if (gasValue < 300) {
    Serial.println("GOOD / NORMAL");
  } else if (gasValue < 600) {
    Serial.println("WARNING");
  } else {
    Serial.println("HIGH / DANGER DEMO");
  }

  delay(1000);
}
```

Expected: Allow the MQ-2 to warm up and record its normal room baseline. Thresholds are classroom demo values, not ppm.

BASIC 4 — Smoke Level

Materials: Arduino UNO, MQ-2 module, breadboard and wires.

Circuit: VCC → 5V; GND → GND; AO → A0.

```
const int SMOKE_PIN = A0;

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

void loop() {
  int smokeValue = analogRead(SMOKE_PIN);

  Serial.print("Smoke Sensor: ");
  Serial.print(smokeValue);
  Serial.print(" | Status: ");

  if (smokeValue < 300) {
    Serial.println("SAFE / NORMAL");
  } else if (smokeValue < 600) {
    Serial.println("SMOKE WARNING");
  } else {
    Serial.println("HIGH SMOKE");
  }

  delay(1000);
}
```

Expected: Do not expose the sensor to flames or concentrated smoke.

BASIC 5 — LDR Light Level

Materials: Arduino UNO, LDR module (preferred) or bare LDR + 10kΩ resistor.

Circuit: LDR module: VCC → 5V; GND → GND; AO → A1. Bare LDR: make a voltage divider and connect its midpoint to A1.

```

const int LDR_PIN = A1;

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

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

  Serial.print("LDR Value: ");
  Serial.print(lightValue);
  Serial.print(" | Light: ");

  if (lightValue < 300) {
    Serial.println("DARK");
  } else if (lightValue < 700) {
    Serial.println("NORMAL");
  } else {
    Serial.println("BRIGHT");
  }

  delay(500);
}

```

Expected: Some LDR modules work in the opposite direction. Observe raw values and reverse the labels if necessary.

BASIC 6 — DHT11 Temperature & Humidity

Materials: Arduino UNO, DHT11, breadboard and wires. Install Adafruit DHT sensor library and its dependency when prompted.

Circuit: VCC → 5V; DATA → D2; GND → GND. Bare DHT11 sensors may need the manufacturer's recommended pull-up resistor.

```

#include <DHT.h>

#define DHTPIN 2
#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

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

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

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("DHT11 read failed!");
    delay(2000);
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(temperature, 1);
  Serial.print(" C | Humidity: ");
  Serial.print(humidity, 1);
  Serial.println(" %");

  delay(2000);
}

```

Expected: Example: Temperature: 25.4 C | Humidity: 68.0 %.

BASIC 7 — PIR Motion Sensor

Materials: Arduino UNO, HC-SR501 PIR, breadboard and wires.

Circuit: VCC → 5V; GND → GND; OUT → D7.

```

const int PIR_PIN = 7;

void setup() {
  Serial.begin(9600);
  pinMode(PIR_PIN, INPUT);
}

void loop() {
  int motion = digitalRead(PIR_PIN);

  if (motion == HIGH) {
    Serial.println("MOTION DETECTED");
  } else {
    Serial.println("NO MOTION");
  }

  delay(500);
}

```

Expected: Allow the PIR to stabilize after power-up and adjust its sensitivity/time-delay controls.

ADVANCED 1 — ROOM COUNTER

Use two HC-SR04 sensors at a doorway. Sensor A triggering first and then B means ENTER. B then A means EXIT. The current count appears in Serial Monitor.

Part	Connection
Sensor A TRIG/ECHO	D3 / D4
Sensor B TRIG/ECHO	D5 / D6
LED	D13 through 220Ω
Buzzer	D8
VCC / GND	5V / GND

```
const int TRIG_A = 3;
const int ECHO_A = 4;
const int TRIG_B = 5;
const int ECHO_B = 6;
const int LED_PIN = 13;
const int BUZZER_PIN = 8;

int people = 0;
const float DETECT_CM = 80.0;

float distanceCM(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH, 25000);
  if (duration == 0) return 999;
  return duration * 0.0343 / 2.0;
}

void beep() {
  digitalWrite(LED_PIN, HIGH);
  digitalWrite(BUZZER_PIN, HIGH);
  delay(150);
  digitalWrite(BUZZER_PIN, LOW);
  digitalWrite(LED_PIN, LOW);
}

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_A, OUTPUT); pinMode(ECHO_A, INPUT);
  pinMode(TRIG_B, OUTPUT); pinMode(ECHO_B, INPUT);
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
  float a = distanceCM(TRIG_A, ECHO_A);
  delay(30);
  float b = distanceCM(TRIG_B, ECHO_B);

  if (a < DETECT_CM && b >= DETECT_CM) {
    unsigned long start = millis();
    while (millis() - start < 1500) {
      b = distanceCM(TRIG_B, ECHO_B);
      if (b < DETECT_CM) {
        people++;
        Serial.print("ENTERED - People inside: ");
        Serial.println(people);
        beep();
        delay(700);
        return;
      }
    }
  }

  if (b < DETECT_CM && a >= DETECT_CM) {
    unsigned long start = millis();
```

```

while (millis() - start < 1500) {
  a = distanceCM(TRIG_A, ECHO_A);
  if (a < DETECT_CM) {
    if (people > 0) people--;
    Serial.print("EXITED - People inside: ");
    Serial.println(people);
    beep();
    delay(700);
    return;
  }
}
}
}

delay(50);
}

```

ADVANCED 2 — MOTION SECURITY LIGHT

The LED turns on only when PIR detects motion and the LDR says the environment is dark.

Part	Connection
PIR OUT	D7
LDR AO	A1
LED	D13 through 220Ω
VCC / GND	5V / GND

```

const int PIR_PIN = 7;
const int LDR_PIN = A1;
const int LED_PIN = 13;

void setup() {
  Serial.begin(9600);
  pinMode(PIR_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);
}

void loop() {
  int motion = digitalRead(PIR_PIN);
  int light = analogRead(LDR_PIN);
  bool dark = (light < 300);

  if (motion == HIGH && dark) {
    digitalWrite(LED_PIN, HIGH);
    Serial.println("DARK + MOTION -> SECURITY LIGHT ON");
  } else {
    digitalWrite(LED_PIN, LOW);
    Serial.println("LIGHT OFF");
  }

  delay(300);
}

```

ADVANCED 3 — GAS LEAK ALARM

MQ-2 AO → A0; LED → D13 through 220Ω; buzzer + → D8; all grounds common; MQ-2 VCC → 5V.

```

const int GAS_PIN = A0;
const int LED_PIN = 13;
const int BUZZER_PIN = 8;

const int WARNING_LEVEL = 300;
const int ALARM_LEVEL = 600;

void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
  int gasValue = analogRead(GAS_PIN);

```

```

Serial.print("Gas: ");
Serial.print(gasValue);

if (gasValue < WARNING_LEVEL) {
  Serial.println(" -> SAFE");
  digitalWrite(LED_PIN, LOW);
  digitalWrite(BUZZER_PIN, LOW);
} else if (gasValue < ALARM_LEVEL) {
  Serial.println(" -> WARNING");
  digitalWrite(LED_PIN, HIGH);
  digitalWrite(BUZZER_PIN, LOW);
} else {
  Serial.println(" -> GAS ALARM");
  digitalWrite(LED_PIN, HIGH);
  digitalWrite(BUZZER_PIN, HIGH);
}

delay(500);
}

```

Educational prototype only. Do not rely on these thresholds for real gas safety.

ADVANCED 4 — SMOKE ALARM

MQ-2 AO → A0; LED → D13 through 220Ω; buzzer + → D8; MQ-2 VCC → 5V; all grounds common.

```

const int SMOKE_PIN = A0;
const int LED_PIN = 13;
const int BUZZER_PIN = 8;

const int WARNING_LEVEL = 300;
const int ALARM_LEVEL = 600;

void setup() {
  Serial.begin(9600);
  pinMode(LED_PIN, OUTPUT);
  pinMode(BUZZER_PIN, OUTPUT);
}

void loop() {
  int smokeValue = analogRead(SMOKE_PIN);

  Serial.print("Smoke: ");
  Serial.print(smokeValue);

  if (smokeValue < WARNING_LEVEL) {
    Serial.println(" -> SAFE");
    digitalWrite(LED_PIN, LOW);
    digitalWrite(BUZZER_PIN, LOW);
  } else if (smokeValue < ALARM_LEVEL) {
    Serial.println(" -> WARNING");
    digitalWrite(LED_PIN, HIGH);
    digitalWrite(BUZZER_PIN, LOW);
  } else {
    Serial.println(" -> SMOKE ALARM");
    digitalWrite(LED_PIN, HIGH);
    digitalWrite(BUZZER_PIN, HIGH);
  }

  delay(500);
}

```

TROUBLESHOOTING

Problem	What to check
LED does not light	Polarity, 220Ω resistor, GND and pin number.
Buzzer silent	Check that it is an active buzzer and polarity.
HC-SR04 no reading	TRIG/ECHO pins, common GND and sensor orientation.
MQ-2 unstable	Warm-up time and baseline reading; thresholds depend on environment.
LDR backwards	Reverse the brightness conditions after observing raw values.
DHT11 read failed	DATA pin, library, power and sensor type.
PIR always HIGH	Warm-up and adjust sensitivity/time-delay potentiometers.
Serial output wrong	Set Serial Monitor to 9600 baud.
Upload fails	Select Arduino UNO and correct COM/USB port.

Recommended student workflow: identify → wire with power disconnected → check VCC/GND/signal → connect USB → upload → open Serial Monitor → test one variable at a time → record readings.

For advanced projects, students should first complete the corresponding Basic Experiment. This makes troubleshooting much easier.

QUICK PIN REFERENCE

Project	Inputs	Outputs
LED + Buzzer	—	D13, D8
Ultrasonic	D9, D10	Serial Monitor
Gas	A0	Serial Monitor
Smoke	A0	Serial Monitor
LDR	A1	Serial Monitor
DHT11	D2	Serial Monitor
PIR	D7	Serial Monitor
Room Counter	D3/D4, D5/D6	D13, D8
Security Light	D7, A1	D13
Gas Alarm	A0	D13, D8
Smoke Alarm	A0	D13, D8

END OF PRACTICAL HANDBOOK

Educational use only. Disconnect power before changing circuits. Gas and smoke projects are demonstrations and are not substitutes for certified safety equipment.