



Gurukul Technicals Institute

Computer education · Robotics · Science projects · Delhi

FREE EBOOK

ROBOTICS AND ELECTRONICS

Arduino Starter Projects

Ten beginner projects with an Arduino Uno, with code explained

INSIDE THIS BOOK

- Meet the Arduino Uno
- Installing the IDE and your first upload
- Digital output, digital input and the Serial Monitor
- Analog input and PWM
- Lights and buttons
- ...and 5 more chapters, practice questions and a glossary

Beginner

Students of class 6 to 12 and hobbyists

engineergurukul.com

+91 97116 99511

Arduino Starter Projects

Get to know the Arduino Uno, install the free Arduino IDE and upload your first program. Then build ten projects, from a traffic light and a light-sensing lamp to a melody player, a distance meter, a thermometer, a servo, a soil moisture alarm and an LCD display, each with parts, pin-by-pin wiring, complete code and a clear explanation.

WHO IT IS FOR	Students of class 6 to 12 and hobbyists
LEVEL	Beginner
SUBJECT AREA	Robotics and electronics
WHAT IS INSIDE	10 chapters, practice questions with answers, and a glossary

CONTENTS

1	Meet the Arduino Uno	4
2	Installing the IDE and your first upload	5
3	Digital output, digital input and the Serial Monitor	7
4	Analog input and PWM	8
5	Lights and buttons	10
6	Light and brightness	12
7	Sound and distance	14
8	Temperature and movement	17
9	Soil and screens	20
10	Fixing problems and what to build next	22
	Practice questions	23

Free ebook: you may read, print and share it for learning; it may not be sold or changed. © 2026 Gurukul Technicals Institute. First edition, October 2026. Contact details are on the back cover.

Meet the Arduino Uno

An **Arduino Uno** is a small circuit board with a tiny computer on it, called a **microcontroller**. You write a set of instructions on a laptop, send them to the board through a USB cable, and the board follows them on its own: it can switch lights, read buttons and sensors, make sounds and move motors.

A program for an Arduino is called a **sketch**. You write sketches in a free program called the **Arduino IDE** (Integrated Development Environment), using a simple form of the C++ language.

A tour of the board

Hold the board with the USB socket on your left. These are the parts you will use most.

Part	What it is for
USB socket	power from the laptop, and the path for uploading sketches
Round power jack	power from a battery pack of 7 V to 12 V
Digital pins 0 to 13	switch things on and off, or read buttons
Pins marked ~ (3, 5, 6, 9, 10, 11)	can also give a "dimmed" output, called PWM
Analog pins A0 to A5	measure a voltage between 0 V and 5 V
5V and 3.3V pins	supply power to sensors
GND pins	ground: the negative side, 0 V
Reset button	restarts your sketch from the beginning
LED marked L	a built-in light joined to pin 13
TX and RX LEDs	flicker when data travels over USB

The main chip on the original Uno is the **ATmega328P**. It runs at 16 million steps per second and has 32 KB of memory for your sketch.

Rules that keep your board alive

- Each pin works at 5 V. Never connect a pin to more than 5 V.
- A pin can safely give about 20 mA. That is enough for one LED with a resistor, but not for a motor. Motors need a driver board.

- Never join the 5V pin directly to GND with a wire. That is a **short circuit**.
- Pins 0 and 1 carry the USB data. Leave them free, or uploads may fail.
- Change wiring only when the USB cable is unplugged.

SAFETY

Everything in this book runs from the laptop's USB port or from batteries of 12 V or less. Never connect any project to mains wiring.

DID YOU KNOW

Arduino started in Italy as a cheap board for design students. Its designs are **open source**, which means anyone may make a compatible board. That is why you will find many low-cost copies, often called clones, in Indian shops.

REMEMBER

This book uses the Uno R3 and its clones. A newer Uno R4 runs the same code, but its pins give less current, so use 1 k Ω resistors with your LEDs on an R4.

CHAPTER 2

Installing the IDE and your first upload

Installing the Arduino IDE

1. On a laptop or desktop, open arduino.cc and go to the Software page.
2. Download the Arduino IDE for your system (Windows, macOS or Linux) and install it like any other program.
3. Open the IDE. The big white area is where you type your sketch.
4. Plug the Uno into the laptop with its USB cable. The power LED marked ON lights up.

Choosing the board and the port

1. Open **Tools** → **Board** and choose **Arduino Uno**.
2. Open **Tools** → **Port** and pick the port that appeared when you plugged in the board. On Windows it is a name such as COM3; on Linux, something like `/dev/ttyACM0` or `/dev/ttyUSB0`.

If no port appears, try another USB cable. Some cheap cables only charge phones and carry no data. Many clone boards use a USB chip called CH340, which needs a driver on some older Windows computers.

Your first sketch: Blink

Type this sketch, or open the built-in version from **File → Examples → 01.Basics → Blink**.

CPP

```
// Blink: flash the built-in LED once a second
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);    // pin 13 on the Uno
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // LED on
  delay(1000);                     // wait 1000 ms, which is 1 second
  digitalWrite(LED_BUILTIN, LOW);  // LED off
  delay(1000);
}
```

Click the **tick** button (Verify) to check the sketch, then the **arrow** button (Upload). After a few seconds, "Done uploading" appears and the L LED blinks.

Reading the sketch

- Lines starting with `//` are **comments**: notes for humans. The board ignores them.
- `setup()` runs once, when the board starts.
- `loop()` runs again and again, for ever, until you unplug the board.
- `pinMode(pin, OUTPUT)` tells a pin that it will send out voltage.
- `digitalWrite(pin, HIGH)` puts 5 V on the pin; `LOW` puts 0 V.
- `delay(1000)` waits 1,000 **milliseconds** (thousandths of a second).
- Every instruction ends with a semicolon `;`. Curly brackets `{ }` hold a group of instructions together.

TRY THIS

Change both delays to 200 and upload again. The LED blinks fast. Now try 100 for on and 900 for off.

TIP

The Arduino language is **case-sensitive**: `digitalWrite` works, but `digitalwrite` gives an error. Copy capital letters exactly.

Digital output, digital input and the Serial Monitor

Digital means only two states: HIGH (5 V, on, 1) or LOW (0 V, off, 0). In this chapter the Arduino controls an LED on a breadboard and reads a push button.

An LED on a pin

An LED needs a resistor so the current stays safe for both the LED and the pin. On 5 V, a 220 Ω resistor gives a red LED about $(5 - 2) \div 220 = 0.014$ A, or 14 mA.

Wiring: pin 12 $\rightarrow$ 220 Ω resistor $\rightarrow$ LED long leg (anode). LED short leg (cathode) $\rightarrow$ GND. Change `LED_BUILTIN` to `12` in Blink and your breadboard LED blinks.

Reading a button with `INPUT_PULLUP`

A pin set as an **input** reads the voltage on it. But an input with nothing connected **floats**: it picks up stray electricity and reads HIGH and LOW at random. The cure is a resistor that holds it steady. The Uno has one built in, the **pull-up resistor**, which you switch on with `INPUT_PULLUP`.

Wiring: one leg of a push button to pin 2, the diagonally opposite leg to GND. With the pull-up on, pin 2 reads HIGH when the button is up and LOW when it is pressed, because pressing joins it to GND.

The Serial Monitor

The **Serial Monitor** is a window in the IDE where your sketch can print messages. It is the best way to see what the board is thinking.

```
// Print the button's state to the Serial Monitor
const int buttonPin = 2;

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  Serial.begin(9600);           // start talking at 9600 bits per second
}

void loop() {
  if (digitalRead(buttonPin) == LOW) {
    Serial.println("Pressed");
  } else {
    Serial.println("Not pressed");
  }
  delay(200);
}
```

Upload, then open **Tools → Serial Monitor**. Set the speed box in the monitor to 9600 baud, the same number as in `Serial.begin()`. Press the button and watch the messages change.

- `const int buttonPin = 2;` gives pin 2 a name. If you move the wire, you change one line.
- `digitalRead()` returns HIGH or LOW.
- `==` asks "is it equal?". A single `=` means "store this value", which is a different thing.
- `Serial.println()` prints and moves to a new line; `Serial.print()` prints without a new line.

REMEMBER

If the Serial Monitor shows strange symbols, its speed does not match the number in `Serial.begin()`.

EXAMPLE

Aarav in Bhopal could not tell why his project misbehaved. He added `Serial.println()` lines at three places in his sketch and saw at once that one part never ran. Printing values is how real programmers find faults too.

CHAPTER 4

Analog input and PWM

Many sensors do not give just on or off. They give a voltage anywhere between 0 V and 5 V. The analog pins measure it.

analogRead

`analogRead(A0)` measures the voltage on pin A0 and returns a number from 0 to 1023. 0 means 0 V and 1023 means 5 V, so each step is about $5 \div 1023 = 0.0049$ V, roughly 5 millivolts.

A **potentiometer** (pot) is a knob that gives any voltage you choose. **Wiring:** one outer leg to 5V, the other outer leg to GND, the middle leg to A0.

CPP

```
// Read a potentiometer and show the number and the voltage
void setup() {
  Serial.begin(9600);
}

void loop() {
  int value = analogRead(A0);           // 0 to 1023
  float volts = value * (5.0 / 1023.0); // change the number to volts
  Serial.print("Reading: ");
  Serial.print(value);
  Serial.print("   Volts: ");
  Serial.println(volts);
  delay(250);
}
```

Turn the knob and watch the numbers move. `int` stores whole numbers; `float` stores numbers with a decimal point.

EXAMPLE

The reading is 512. Volts = $512 \times 5.0 \div 1023$ = about 2.50 V, half of 5 V. So the knob is near the middle.

PWM: dimming with fast switching

A digital pin can only be fully on or fully off. To make an LED look half bright, the Uno switches the pin on and off hundreds of times a second. If the pin is on for half of each tiny cycle, the LED looks half bright. This is **PWM** (pulse width modulation), and it works only on the pins marked ~.

`analogWrite(pin, value)` takes a value from 0 (always off) to 255 (always on). 127 is about half.

```
// Fade an LED up and down on PWM pin 9
const int ledPin = 9;

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  for (int b = 0; b <= 255; b++) {    // brighter
    analogWrite(ledPin, b);
    delay(5);
  }
  for (int b = 255; b >= 0; b--) {    // dimmer
    analogWrite(ledPin, b);
    delay(5);
  }
}
```

Wiring: pin 9 → 220 Ω → LED → GND.

The `for` loop counts: start `b` at 0, keep going while `b <= 255`, and add 1 each time (`b++`).

DID YOU KNOW

PWM on most Uno pins switches about 490 times a second, and on pins 5 and 6 about 980 times. Your eye cannot follow such speed, so it sees the average brightness.

CHAPTER 5

Lights and buttons

Now the projects begin. Each one lists the parts, describes the wiring pin by pin and gives the complete sketch.

Project 1: Traffic light

Parts: Uno, breadboard, red, yellow and green LEDs, three 220 Ω resistors, jumper wires.

Wiring:

- Pin 10 → 220 Ω → red LED → GND
- Pin 9 → 220 Ω → yellow LED → GND
- Pin 8 → 220 Ω → green LED → GND

Use the breadboard's blue rail as a shared GND, joined to a GND pin.

```
// Project 1: traffic light
const int redPin = 10;
const int yellowPin = 9;
const int greenPin = 8;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(yellowPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
}

// Switch every light off, then one light on, then wait
void showLight(int pin, unsigned long ms) {
  digitalWrite(redPin, LOW);
  digitalWrite(yellowPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(pin, HIGH);
  delay(ms);
}

void loop() {
  showLight(redPin, 5000);    // stop
  showLight(greenPin, 5000);  // go
  showLight(yellowPin, 2000); // get ready to stop
}
```

How it works: `showLight()` is a **function** we wrote ourselves: a named block of steps that we can use again. `loop()` calls it three times with different pins and times. `unsigned long` is a type for large whole numbers that are never negative, the type the Arduino uses for times.

TRY THIS

Add a second set of LEDs on pins 5, 6 and 7 for the crossing road. When one road is green, the other must be red.

Project 2: Button-controlled LED

Press once for on; press again for off.

Parts: Uno, breadboard, a push button, an LED, a 220 Ω resistor.

Wiring:

- Pin 2 → button → GND (diagonally opposite legs)
- Pin 12 → 220 Ω → LED → GND

```
// Project 2: press once for on, press again for off
const int buttonPin = 2;
const int ledPin = 12;

bool ledOn = false;
int lastState = HIGH;      // HIGH means "not pressed" with INPUT_PULLUP

void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int state = digitalRead(buttonPin);
  if (state != lastState) {      // the button has changed
    if (state == LOW) {          // it was just pressed
      ledOn = !ledOn;           // flip true to false, or false to true
      digitalWrite(ledPin, ledOn ? HIGH : LOW);
    }
    delay(50);                  // let the contacts settle
    lastState = state;
  }
}
```

How it works: the sketch remembers the last state of the button. It acts only when the state changes from not pressed to pressed, so holding the button down does not flip the LED again and again. `bool` stores `true` or `false`, and `!` turns one into the other.

Metal contacts in a button **bounce**: for a few thousandths of a second they touch and separate many times. Without the 50 ms pause, one press could count as several. Waiting for the bounce to end is called **debouncing**.

TIP

`ledOn ? HIGH : LOW` is a short form of "if `ledOn` is true, use `HIGH`, otherwise use `LOW`".

CHAPTER 6

Light and brightness

Project 3: LDR automatic light

An **LDR** (light-dependent resistor) has low resistance in bright light and high resistance in the dark. With a fixed resistor it makes a **voltage divider**, which turns light into a voltage the Uno can measure.

Parts: Uno, breadboard, an LDR, a 10 k Ω resistor, an LED, a 220 Ω resistor.

Wiring:

- 5V → LDR → A0
- A0 → 10 k Ω → GND
- Pin 9 → 220 Ω → LED → GND

In bright light the LDR's resistance is low, so A0 is near 5 V and the reading is high. In the dark the reading falls.

CPP

```
// Project 3: LED turns on when it gets dark
const int ldrPin = A0;
const int ledPin = 9;
const int darkLevel = 300;    // turn on below this
const int brightLevel = 400;  // turn off above this

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

void loop() {
  int light = analogRead(ldrPin);
  Serial.println(light);

  if (light < darkLevel) {
    digitalWrite(ledPin, HIGH);
  } else if (light > brightLevel) {
    digitalWrite(ledPin, LOW);
  }
  // between the two levels, the LED stays as it was
  delay(200);
}
```

How it works: with one switching level, a reading wobbling around it would make the lamp flicker. Two levels with a gap between them stop this. The gap is called **hysteresis**; the thermostat in a fridge works the same way.

Setting it up: open the Serial Monitor and note the reading in normal room light and with your hand over the LDR. Choose `darkLevel` and `brightLevel` between those numbers.

EXAMPLE

In Shreya's room in Kochi, the reading is 750 in daylight and 120 under her hand. She sets `darkLevel` to 250 and `brightLevel` to 350.

Project 4: Potentiometer dimmer

Parts: Uno, breadboard, a 10 k Ω potentiometer, an LED, a 220 Ω resistor.

Wiring:

- Pot outer legs → 5V and GND; pot middle leg → A0
- Pin 9 → 220 Ω → LED → GND

CPP

```
// Project 4: turn the knob to set the brightness
const int potPin = A0;
const int ledPin = 9;      // a PWM pin

void setup() {
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int knob = analogRead(potPin);           // 0 to 1023
  int brightness = map(knob, 0, 1023, 0, 255); // change the range
  analogWrite(ledPin, brightness);
  delay(10);
}
```

How it works: `analogRead()` gives 0 to 1023, but `analogWrite()` wants 0 to 255. `map()` converts a number from one range to another, so 512 becomes about 127.

TRY THIS

Swap the last two numbers in `map()` to 255, 0. Now the knob works backwards.

REMEMBER

`analogWrite()` dims only on the ~ pins. On other pins, values below 128 give off and the rest give on.

CHAPTER 7

Sound and distance

Project 5: Buzzer melody

A **passive buzzer** clicks once each time its voltage changes. Click it hundreds of times a second and you hear a note. The `tone()` function does this for you. (An **active** buzzer has its own beeper inside and plays only one sound.)

Parts: Uno, a passive buzzer.

Wiring: pin 8 → buzzer + leg; buzzer other leg → GND.

Music notes have fixed frequencies. If we take the note C as Sa, the sargam Sa Re Ga Ma Pa Dha Ni Sa matches the notes C D E F G A B C.

CPP

```
// Project 5: play the sargam, then "Twinkle, twinkle, little star"
const int buzzerPin = 8;

// Note frequencies in hertz, with Sa taken as the note C
const int SA = 262, RE = 294, GA = 330, MA = 349;
const int PA = 392, DHA = 440, NI = 494, SA2 = 523;

int scale[] = {SA, RE, GA, MA, PA, DHA, NI, SA2};

// 0 means a rest (silence)
int tune[] = {SA, SA, PA, PA, DHA, DHA, PA, 0,
              MA, MA, GA, GA, RE, RE, SA, 0};
int beats[] = {1, 1, 1, 1, 1, 1, 2, 1,
               1, 1, 1, 1, 1, 1, 2, 1};

const int beat = 300;      // one beat, in milliseconds

void setup() {
  for (int i = 0; i < 8; i++) {
    tone(buzzerPin, scale[i], 250);
    delay(300);
  }
  delay(1000);
}

void loop() {
  for (int i = 0; i < 16; i++) {
    int length = beats[i] * beat;
    if (tune[i] > 0) {
      tone(buzzerPin, tune[i], length - 50);  // leave a small gap
    }
    delay(length);
  }
  delay(2000);
}
```

How it works: an **array** such as `scale[]` is a numbered list. `scale[0]` is the first item and `scale[7]` the eighth. `tone(pin, frequency, duration)` starts a note and lets the sketch carry on, so `delay()` waits for the note to finish.

DID YOU KNOW

Double any note's frequency and you get the same note one octave higher. Sa is 262 Hz here, and the upper Sa is 523 Hz.

Project 6: Ultrasonic distance meter

The **HC-SR04** sends out a burst of sound too high for humans to hear, then listens for the echo. The longer the echo takes, the farther away the object is.

Parts: Uno, an HC-SR04 sensor, an LED, a 220 Ω resistor.

Wiring:

- HC-SR04 VCC → 5V; GND → GND
- Trig → pin 9; Echo → pin 10
- Pin 13 → 220 Ω → LED → GND (or watch the L LED)

CPP

```
// Project 6: ultrasonic distance meter
const int trigPin = 9;
const int echoPin = 10;
const int ledPin = 13;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  // a 10-microsecond pulse on Trig starts one measurement
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // time the echo; give up after 30,000 microseconds
  unsigned long duration = pulseIn(echoPin, HIGH, 30000UL);

  if (duration == 0) {
    Serial.println("Out of range");
    digitalWrite(ledPin, LOW);
  } else {
    float cm = duration * 0.0343 / 2;
    Serial.print("Distance: ");
    Serial.print(cm);
    Serial.println(" cm");
    digitalWrite(ledPin, cm < 20 ? HIGH : LOW); // warn when closer than 20 cm
  }
  delay(100);
}
```

How it works: sound travels about 343 metres per second in air at 20 °C, which is 0.0343 cm per **microsecond** (millionth of a second). `pulseIn()` measures how long the Echo pin stays HIGH: the time to the object and back. So we multiply by 0.0343 and divide by 2.

EXAMPLE

The echo lasts 1,166 microseconds. Distance = $1,166 \times 0.0343 \div 2$ = about 20 cm.

TIP

The sensor works best from about 2 cm to a few metres, facing a flat, hard surface. Soft cloth and sloping surfaces send weak echoes.

CHAPTER 8

Temperature and movement

Project 7: Temperature display

Option A: the LM35. The LM35 is a small three-legged sensor whose output rises by 10 millivolts for every degree Celsius. At 25 °C it gives 0.25 V.

Parts: Uno, an LM35.

Wiring: hold the LM35 with its flat face towards you and legs down. Left leg → 5V; middle leg → A0; right leg → GND.

CPP

```
// Project 7a: thermometer with an LM35
const int sensorPin = A0;

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

void loop() {
  int reading = analogRead(sensorPin);
  float volts = reading * (5.0 / 1023.0);
  float tempC = volts * 100.0;      // 10 mV per degree
  Serial.print("Temperature: ");
  Serial.print(tempC, 1);          // one decimal place
  Serial.println(" C");
  delay(1000);
}
```

Each step of `analogRead()` is about 5 mV, so this thermometer moves in steps of about half a degree.

SAFETY

If the LM35 gets hot within seconds of plugging in, unplug the USB cable at once: it is wired the wrong way round. Let it cool before you touch it.

Option B: the DHT11. The DHT11 measures temperature and humidity (the amount of water vapour in the air) and sends both as digital data on one wire. It reads about 0 to 50 °C, to within about 2 °C.

Parts: Uno, a DHT11 on a small three-pin module (the module has the pull-up resistor the sensor needs).

Wiring: module + (VCC) → 5V; module - (GND) → GND; module data (OUT or S) → pin 2.

First install the library: open **Sketch → Include Library → Manage Libraries**, search for "DHT sensor library" by Adafruit and click Install. When the IDE asks to install the libraries it needs, agree.

CPP

```
// Project 7b: temperature and humidity with a DHT11
#include <DHT.h>

DHT dht(2, DHT11);    // data on pin 2, sensor type DHT11

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

void loop() {
  delay(2000);          // the DHT11 needs time between readings
  float h = dht.readHumidity();
  float t = dht.readTemperature();    // in Celsius
  if (isnan(h) || isnan(t)) {         // isnan means "is not a number"
    Serial.println("No reading. Check the wires.");
    return;
  }
  Serial.print("Temperature: ");
  Serial.print(t, 1);
  Serial.print(" C  Humidity: ");
  Serial.print(h, 0);
  Serial.println(" %");
}
```

A **library** is ready-made code that someone has written for a part, so you do not have to. `#include` adds it to your sketch. `return` ends this run of `loop()` early, and the next run starts.

DID YOU KNOW

On a monsoon day in Mumbai the air holds far more water vapour than on a dry winter afternoon in Jodhpur. Run your sensor in a steamy kitchen and then in a dry room, and watch the humidity change.

Project 8: Servo sweep

A **servo** is a motor that turns to an exact angle, from 0° to 180°, and holds it. The small SG90 is the usual choice for projects.

Parts: Uno, an SG90 servo.

Wiring: the servo has three wires. Brown → GND; red → 5V; orange (signal) → pin 9.

CPP

```
// Project 8: servo sweep
#include <Servo.h>

Servo myServo;

void setup() {
  myServo.attach(9);      // signal wire on pin 9
}

void loop() {
  for (int angle = 0; angle <= 180; angle++) {
    myServo.write(angle);  // go to this angle
    delay(15);
  }
  for (int angle = 180; angle >= 0; angle--) {
    myServo.write(angle);
    delay(15);
  }
}
```

How it works: the Servo library comes with the IDE. `attach(9)` connects the servo to pin 9, and `write(angle)` moves it. The short delay gives the arm time to reach each angle.

TIP

One small servo with nothing on its arm can run from the Uno's 5V pin. If the board resets or the servo shakes, give the servo its own supply of about 5 V, such as four rechargeable AA cells, and join that supply's - to the Uno's GND.

TRY THIS

Add a potentiometer on A0 and replace `loop()` with two lines: `int angle = map(analogRead(A0), 0, 1023, 0, 180);` then `myServo.write(angle);`. The servo now follows the knob.

Soil and screens

Project 9: Soil moisture alarm

A common soil moisture module has a fork with two metal legs that go into the soil, and a small board. Wet soil conducts electricity better than dry soil, and the board turns that into a voltage on its **AO** (analog out) pin.

Parts: Uno, a soil moisture fork and module, a passive buzzer, an LED, a 220 Ω resistor.

Wiring:

- Module VCC → 5V; GND → GND; AO → A0
- Pin 8 → buzzer + leg; buzzer other leg → GND
- Pin 13 → 220 Ω → LED → GND

CPP

```
// Project 9: soil moisture alarm
const int sensorPin = A0;
const int buzzerPin = 8;
const int ledPin = 13;
const int dryLevel = 700;    // change this after testing your own soil

void setup() {
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int moisture = analogRead(sensorPin);
  Serial.print("Soil reading: ");
  Serial.println(moisture);

  if (moisture > dryLevel) {      // a higher number means drier soil
    digitalWrite(ledPin, HIGH);
    tone(buzzerPin, 1000, 200);  // a short beep
  } else {
    digitalWrite(ledPin, LOW);
  }
  delay(1000);
}
```

Setting it up: with most of these modules the reading goes down as the soil gets wetter. Test yours: note the reading in dry soil and again just after watering, and set `dryLevel` between them.

EXAMPLE

Ishaan in Nashik tested his tulsi pot: 820 when dry and 410 after watering. He set `dryLevel` to 650, and now the buzzer reminds him to water the plant.

REMEMBER

The metal legs slowly corrode in damp soil, because a current flows through them. A **capacitive** soil sensor has no bare metal in the soil and lasts much longer.

Project 10: A 16x2 LCD display

A **16x2 LCD** shows 2 lines of 16 characters. On its own it needs six signal wires plus power and a contrast knob. With a small **I2C backpack** soldered on its back, it needs only four wires. **I2C** is a way for chips to talk over two shared wires, SDA (data) and SCL (clock). On the Uno, SDA is pin A4 and SCL is pin A5.

Parts: Uno, a 16x2 LCD with an I2C backpack.

Wiring: GND → GND; VCC → 5V; SDA → A4; SCL → A5.

Install the library: in **Manage Libraries**, search for "LiquidCrystal I2C" by Frank de Brabander and install it.

CPP

```
// Project 10: messages on a 16x2 LCD with an I2C backpack
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // address, columns, rows

void setup() {
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);           // column 0, row 0: the top line
  lcd.print("Namaste, India!");
}

void loop() {
  lcd.setCursor(0, 1);          // the bottom line
  lcd.print("Seconds: ");
  lcd.print(millis() / 1000);    // time since the board started
  delay(500);
}
```

How it works: every I2C device has an **address**, a number written here in hexadecimal. Most backpacks use 0x27; some use 0x3F. `setCursor(column, row)` chooses where the next text goes, counting from 0. `millis()` gives the milliseconds since the board started.

If the screen stays blank or shows only boxes: turn the small blue contrast knob on the backpack with a screwdriver. If that does not help, change `0x27` to `0x3F` and upload again.

TIP

Without a backpack, use the built-in LiquidCrystal library instead. Its examples use pins 12, 11, 5, 4, 3 and 2 for the LCD, plus a 10 kΩ potentiometer for contrast.

TRY THIS

Join Project 6 and Project 10: show the distance in centimetres on the LCD instead of the Serial Monitor.

CHAPTER 10

Fixing problems and what to build next

Every Arduino builder sees error messages. They are not a disaster; they are clues.

When the upload fails

What you see	Likely cause	What to do
No port in Tools → Port	charge-only cable, or a missing driver	try a data cable; install the CH340 driver on clone boards
"not in sync" or "programmer is not responding"	wrong board or port chosen, or wires on pins 0 and 1	choose Arduino Uno and the right port; free pins 0 and 1
"was not declared in this scope"	a spelling or capital-letter mistake	compare the name letter by letter
"expected ';' before"	a missing semicolon	look at the line above the one named
"No such file or directory" after <code>#include</code>	the library is not installed	install it in Manage Libraries

When the sketch runs but the project misbehaves

- **LED stays dark:** check that the long leg faces the pin, that the pin in the code matches the wire, and that GND is connected.
- **Readings jump about:** an input pin may be floating. Check that the sensor's wire is firmly in.
- **Serial Monitor shows rubbish:** set its speed to the number in `Serial.begin()`.

- **Board restarts when a motor moves:** the motor needs its own battery supply, with GNDs joined.
- **Nothing makes sense:** add `Serial.println()` lines to show values, and test one part at a time.

EXAMPLE

Neha in Patna built the distance meter, but it always said "Out of range". The code was correct, so she checked the wiring: Trig and Echo were swapped. She swapped them back and it worked.

Good habits

1. Build one small part, test it, then add the next part.
2. Save each working sketch with a new name, such as `traffic_light_v2`, before you change it.
3. Write comments that explain why, not just what.
4. Keep your parts in labelled boxes and your wires tidy.

What to build next

You can now join the projects into bigger ones:

- A smart dustbin lid that opens with a servo when the distance meter sees a hand.
- A weather station that shows temperature and humidity on the LCD.
- A plant guardian that beeps when the soil is dry and shows the reading on the LCD.
- A parking helper that beeps faster as a toy car gets closer to a wall.

REMEMBER

Every big project is small projects joined together. Test each part alone first, and you will always know where a fault is.

SAFETY

As your projects grow, keep to batteries of 12 V or less, add a driver board for every motor, and ask an adult to help with soldering, cutting and hot glue.

CHECK YOURSELF

Practice questions

1. What is a sketch, and what program do you write it in?
2. Which Uno pins can give a PWM output?

3. About how much current can one Uno pin safely give?
4. Why should you leave pins 0 and 1 free?
5. What is the difference between `setup()` and `loop()`?
6. What does `delay(500)` do?
7. Work out the current through a red LED with a 220 Ω resistor on a 5 V pin.
8. What does "floating" mean for an input pin, and how does `INPUT_PULLUP` fix it?
9. With `INPUT_PULLUP` and a button to GND, what does the pin read when the button is pressed?
10. The Serial Monitor shows strange symbols. What is the likely cause?
11. What range of numbers does `analogRead()` return, and what voltage does 1023 stand for?
12. An `analogRead()` gives 205. About what voltage is on the pin?
13. What does `map(knob, 0, 1023, 0, 255)` do?
14. What is button bounce, and how does Project 2 deal with it?
15. Why does Project 3 use two light levels instead of one?
16. What is the difference between an active and a passive buzzer?
17. The HC-SR04's echo lasts 2,332 microseconds. How far away is the object?
18. An LM35 gives 0.31 V. What is the temperature?
19. Which two libraries did you install in this book, and what is a library?
20. Name the three wires of an SG90 servo and where each goes.
21. With most soil moisture modules, does the reading go up or down as the soil gets wetter?
22. Which Uno pins are SDA and SCL?
23. An I2C LCD lights up but shows nothing. Name two things to try.

ANSWER KEY

Answers

1. A sketch is a program for an Arduino. You write it in the Arduino IDE.
2. Pins 3, 5, 6, 9, 10 and 11, marked with ~.
3. About 20 mA.
4. They carry the USB data, so parts on them can stop uploads.
5. `setup()` runs once when the board starts; `loop()` runs again and again for ever.
6. It makes the sketch wait 500 milliseconds, which is half a second.
7. $(5 - 2) \div 220 = \text{about } 0.014 \text{ A, or } 14 \text{ mA.}$

8. An input with nothing connected picks up stray electricity and reads at random.
`INPUT_PULLUP` switches on a built-in resistor that holds the pin HIGH until something pulls it LOW.
9. LOW, because pressing joins the pin to GND.
10. The monitor's speed does not match the number in `Serial.begin()`.
11. 0 to 1023; 1023 stands for 5 V.
12. $205 \times 5 \div 1023 = \text{about } 1.0 \text{ V}$.
13. It converts a number in the range 0 to 1023 into the matching number in the range 0 to 255.
14. Button contacts touch and separate many times when pressed. Project 2 waits 50 ms after a change before reading again.
15. So the lamp does not flicker when the reading wobbles around one level. The gap between the levels is called hysteresis.
16. An active buzzer has its own beeper inside and makes one sound on plain voltage. A passive buzzer needs a changing signal, such as from `tone()`, and can play different notes.
17. $2,332 \times 0.0343 \div 2 = \text{about } 40 \text{ cm}$.
18. $0.31 \text{ V} \div 0.010 \text{ V per degree} = 31 \text{ }^{\circ}\text{C}$.
19. The DHT sensor library and the LiquidCrystal I2C library. A library is ready-made code for a part, added to a sketch with `#include`.
20. Brown to GND, red to 5 V (or a separate 5 V supply), orange (signal) to a pin such as 9.
21. Down.
22. SDA is A4 and SCL is A5.
23. Turn the contrast knob on the backpack, and try the other address, 0x3F instead of 0x27.

WORD LIST

Glossary

Analog

a signal that can have any value in a range, such as 0 V to 5 V.

Array

a numbered list of values stored under one name.

Baud

the speed of serial data, in bits per second; both sides must use the same.

Debouncing

ignoring the quick on-off flickers of a button's contacts.

Digital

a signal with only two states, HIGH and LOW.

Function

a named block of code that can be used again and again.

GND

ground, the 0 V side of a circuit.

Hysteresis

using two switching levels with a gap, so a reading near one level does not cause flicker.

I2C

a way for chips to share data over two wires, SDA and SCL.

IDE

the program in which you write, check and upload sketches.

Library

ready-made code for a part or task, added with `#include`.

Microcontroller

a tiny computer on one chip that runs one program.

Millisecond

one thousandth of a second.

Potentiometer

a knob that gives an adjustable voltage.

Pull-up resistor

a resistor that holds an input HIGH until something pulls it LOW.

PWM

pulse width modulation, fast on-off switching that acts like a lower voltage.

Serial Monitor

the IDE window that shows messages from the board.

Servo

a motor that turns to an exact angle and holds it.

Sketch

a program written for an Arduino.

Variable

a named place in memory that stores a value, such as `int light`.

Voltage divider

two resistances in series that share a voltage, used to read sensors.



Gurukul Technicals Institute

Computer education institute in Delhi, with authorised member centres, robotics classes and science projects, working models and DIY kits.

COURSES

Computer & Office Skills · Accounting & Business Software · Programming & Web Development · Design & Digital Skills · Hardware & Networking · Robotics & Coding · Other Programmes

PROJECTS AND KITS

Ready-made school projects · DIY and STEM kits · Robotics and coding · Components and spares · Engineering projects

CONTACT

Head office: A-10, Chief Commissioner Colony, opposite Rana Pratap Bagh, Delhi 110007

Phone: 011 2721 1512 · **Mobile and WhatsApp:** +91 97116 99511

Email: engineergurukul@yahoo.co.in

Website: engineergurukul.com · **Instagram:** [@science_projects_club](https://www.instagram.com/science_projects_club)

More free ebooks for computer courses, robotics and science projects: engineergurukul.com/resources/ebooks/

Free to read, print and share. Not for sale.