



Gurukul Technicals Institute

Computer education · Robotics · Science projects · Delhi

FREE EBOOK

ROBOTICS AND ELECTRONICS

Arduino and Coding for Robots Study Book

The complete study notes, practice questions and glossary for Arduino and Coding for Robots

INSIDE THIS BOOK

- Meet the Arduino Uno and Its Pins
- The Arduino IDE and Your First Sketch
- Buttons, digitalRead and the Serial Monitor
- Variables, if/else and Loops
- ...and 3 more chapters, practice questions and a glossary

Beginner

Students of class 6 to 12 starting Arduino

engineergurukul.com

+91 97116 99511

Arduino and Coding for Robots Study Book

Program an Arduino Uno from your first blinking LED to a working robot: pins, the Arduino IDE, inputs and outputs, the Serial Monitor, variables, decisions, loops and functions, sensors, motor drivers, servos and debugging.

WHO IT IS FOR	Students of class 6 to 12 starting Arduino
LEVEL	Beginner
SUBJECT AREA	Robotics and electronics
WHAT IS INSIDE	7 chapters, practice questions with answers, and a glossary

CONTENTS

1	Meet the Arduino Uno and Its Pins	3
2	The Arduino IDE and Your First Sketch	5
3	Buttons, digitalRead and the Serial Monitor	7
4	Variables, if/else and Loops	9
5	Analog Input, PWM Output and Functions	12
6	Reading Sensors and Driving Motors with the L298N	15
7	The Servo Library and Debugging Your Robot	18
	Practice questions	20
	Answers	24
	Glossary	25

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 and Its Pins

What is an Arduino?

Arduino is an open-source electronics platform: a family of small boards built around a microcontroller, plus free software for writing programs and loading them onto the board. A **microcontroller** is a tiny computer on a single chip, with its own processor, memory and input/output pins. Once a program is uploaded, the board runs it every time it gets power, with no computer attached.

The Arduino Uno is the most widely used board for learning. The details below are for the long-running **Uno R3**. The newer Uno R4 boards keep the same shape and pin layout but use a faster chip with more memory.

Uno R3 at a glance

Feature	Uno R3
Microcontroller	ATmega328P
Clock speed	16 MHz
Operating voltage	5 V
Digital input/output pins	14 (D0 to D13)
PWM pins, marked ~	3, 5, 6, 9, 10 and 11
Analog input pins	6 (A0 to A5)
Flash memory, for programs	32 KB, of which 0.5 KB holds the bootloader
SRAM, for variables	2 KB
EEPROM, which survives power-off	1 KB

The pins and what they do

- **Digital pins 0 to 13** can each be an input or an output, and they work with two levels: HIGH (5 V) or LOW (0 V).
- **Pins 0 (RX) and 1 (TX)** are also the serial lines that carry data over USB. Leave them free while you are learning, or uploads can fail.
- **Pin 13** has a small LED on the board, called `LED_BUILTIN` in code.

- **PWM pins**, marked with a ~, can switch on and off very fast to imitate a voltage between 0 and 5 V. This dims LEDs and controls motor speed.
- **A0 to A5** measure voltages between 0 and 5 V. They can also be used as extra digital pins.
- **Power header**: 5V and 3.3V supply small modules, GND is ground (0 V), and VIN is the input used when the board is powered from a battery through that pin.

Every circuit connected to the Arduino must share its GND. A missing common ground is one of the most frequent reasons a project behaves strangely.

Powering the board

While you learn, the USB cable from the computer powers the board and carries your program. A project that runs on its own can use the DC barrel jack, for which 7 V to 12 V is recommended; the on-board regulator brings it down to 5 V.

Current limits

An Uno R3 pin can safely supply about 20 mA. That is enough for an LED with a resistor, but nowhere near enough for a motor, a relay coil or a strip of lights. Those loads need a transistor, a relay module or a motor driver. Drawing too much current from a pin can permanently damage the chip, so always check what a part needs before connecting it.

KEY POINTS

- The Uno R3 uses the ATmega328P microcontroller at 16 MHz and works at 5 V.
- It has 14 digital pins, 6 of them with PWM (marked ~), and 6 analog inputs, A0 to A5.
- Keep pins 0 and 1 free for USB serial; pin 13 has the built-in LED.
- Every part of a project must share a common GND.
- A pin can supply only about 20 mA, so motors and relays need a driver.

TRY THESE EXERCISES

1. On a real Uno, find the PWM pins (~), the analog pins, the GND pins, the 5V pin and the USB socket, and point them out to your partner.
2. Draw a labelled diagram of the board on paper showing each group of pins and what it is used for.
3. List three loads that need more than 20 mA and write down what you would use to switch each one.

The Arduino IDE and Your First Sketch

The Arduino IDE

The Arduino IDE (integrated development environment) is the free program in which you write, check and upload code. Version 2 runs on Windows, macOS and Linux. An Arduino program is called a **sketch**, and it is saved as a `.ino` file inside a folder with the same name.

Button	What it does
Verify (tick)	Compiles the sketch and reports any errors
Upload (right arrow)	Compiles the sketch and sends it to the board
Serial Monitor (magnifier, top right)	Shows text sent by the board

Before the first upload, connect the board with a USB data cable, then choose **Arduino Uno** and its port in the board selector or the Tools menu. On Windows the port appears as COM followed by a number.

Sketch structure

Every sketch has two functions:

- `setup()` runs once, when the board powers up or is reset. Put one-time settings here.
- `loop()` runs again and again, for ever, after `setup()` ends. Put the main job here.

Blink: the first sketch

CPP

```
// Blink: turns the built-in LED on and off every second
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);    // make pin 13 an output
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // LED on (5 V)
  delay(1000);                     // wait 1000 ms = 1 second
  digitalWrite(LED_BUILTIN, LOW);  // LED off (0 V)
  delay(1000);
}
```

- `pinMode(pin, mode)` sets a pin as `OUTPUT` or `INPUT` (or `INPUT_PULLUP`, which comes next).
- `digitalWrite(pin, value)` makes an output pin `HIGH` or `LOW`.
- `delay(ms)` pauses the program for the given number of milliseconds.

- Text after `//` is a comment, which the compiler ignores.

An external LED

To blink an LED on pin 8, connect pin 8 to a 220 Ω resistor, the resistor to the LED's longer leg (anode), and the shorter leg (cathode) to GND. In the code, add `const int LED_PIN = 8;` at the top and use `LED_PIN` in place of `LED_BUILTIN`. A named constant makes a sketch easier to read and to change later.

Rules that save time

1. Arduino code is C++, so it is **case-sensitive**: `digitalWrite` works, `digitalwrite` does not.
2. Each instruction, such as `delay(1000);`, ends with a semicolon. A line that opens a block, such as `void loop() {`, does not.
3. Every `{` needs a matching `}`. Use **Tools > Auto Format** to tidy the indentation, which makes a missing brace easy to spot.
4. Click **Verify** often, not only at the end.

If an upload fails, check three things first: the right board is selected, the right port is selected, and the USB cable carries data, because some cables only charge.

KEY POINTS

- A sketch is saved as a .ino file inside a folder of the same name.
- Verify compiles the sketch; Upload compiles it and sends it to the board.
- `setup()` runs once and `loop()` repeats for ever.
- `pinMode` sets a pin's mode, `digitalWrite` sets it HIGH or LOW, and `delay` pauses in milliseconds.
- Arduino code is case-sensitive, and each instruction ends with a semicolon.

TRY THESE EXERCISES

1. Upload the Blink sketch, then change both delays to 200 and watch the LED blink faster.
2. Wire an LED with a 220 Ω resistor to pin 8 and make it blink using a named constant for the pin.
3. Make the LED flash a pattern: on for 200 ms, off for 200 ms, on for one second, off for one second.

Buttons, digitalRead and the Serial Monitor

Reading a digital input

`digitalRead(pin)` returns `HIGH` or `LOW`. An input pin connected to nothing is said to be **floating**: it picks up electrical noise and reads at random. So a button circuit always needs a resistor that holds the pin at a known level while the button is not pressed.

The easiest way is the Uno's built-in pull-up resistor, switched on with the `INPUT_PULLUP` mode. Wire the button between the pin and GND.

Button	Pin reads
Not pressed	<code>HIGH</code> , pulled up to 5 V inside the chip
Pressed	<code>LOW</code> , connected to GND

The logic looks backwards at first, but it needs no extra parts.

The Serial Monitor

The board can send text back to the computer over the USB cable. Start serial communication in `setup()` with `Serial.begin(9600);`, where 9600 is the speed in bits per second, called the **baud rate**. Then:

- `Serial.print(x)` sends a value and stays on the same line.
- `Serial.println(x)` sends a value and then moves to a new line.

Open the Serial Monitor and choose the same baud rate in its drop-down. If the two do not match, you see garbage characters.

Example: a button that lights an LED

CPP

```
const int BUTTON_PIN = 2;
const int LED_PIN = 8;

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

void loop() {
  int state = digitalRead(BUTTON_PIN);
  if (state == LOW) {           // pressed
    digitalWrite(LED_PIN, HIGH);
    Serial.println("Button pressed");
  } else {
    digitalWrite(LED_PIN, LOW);
  }
  delay(50);
}
```

While the button is held down, the monitor prints "Button pressed" about twenty times a second, because `loop()` keeps running. That is a useful lesson in itself: a looping program reacts to a state, not to a single press.

Bouncing

A mechanical button's contacts bounce for a few milliseconds when pressed, so one press can look like several quick presses. The `delay(50)` above hides most of this. A more careful sketch checks that the reading has stayed the same for a short time before trusting it, which is called **debouncing**.

Using the Serial Monitor to check your thinking

The Serial Monitor is your window into the board. Print a variable's value, or a short message such as `Serial.println("reached step 3");`, to confirm that the program is doing what you think. You can also type into the monitor's input box and read the characters on the board with `Serial.read()`.

Serial communication uses pins 0 and 1, so connect nothing to them while your sketch uses `Serial`.

KEY POINTS

- `digitalRead` returns HIGH or LOW.
- A floating input reads at random; `INPUT_PULLUP` fixes this with a built-in resistor.
- With `INPUT_PULLUP` and a button wired to GND, a pressed button reads LOW.
- `Serial.begin(9600)` starts serial communication, and the Serial Monitor must use the same baud rate.
- `Serial.print` and `Serial.println` let you see inside a running program.

TRY THESE EXERCISES

1. Build the button-and-LED circuit and confirm the messages in the Serial Monitor.
2. Change the sketch so that the LED is on when the button is not pressed and off while it is held.
3. Write a sketch with `Serial.begin(9600)`; in `setup()` and `Serial.println(millis());` followed by `delay(1000)`; in `loop()`, and watch the number grow by about 1000 each time.

CHAPTER 4

Variables, if/else and Loops

Variables

A variable is a named box that holds a value. You declare it with a type:

Type	Holds	On the Uno R3
<code>int</code>	whole numbers	16 bits: -32,768 to 32,767
<code>long</code>	larger whole numbers	32 bits: about ± 2.1 billion
<code>unsigned long</code>	zero and positive whole numbers	0 to about 4.29 billion; used for <code>millis()</code>
<code>float</code>	numbers with a decimal point	about 6 to 7 significant digits
<code>bool</code>	<code>true</code> or <code>false</code>	1 byte
<code>char</code>	one character, such as <code>'A'</code>	1 byte

On 32-bit boards such as the Uno R4, an `int` is 32 bits. A variable declared outside every function is **global** and can be used anywhere; one declared inside a function is **local** to it. Use `const` for values that never change, such as pin numbers.

Doing arithmetic

Use `+`, `-`, `*` and `/` for arithmetic and `%` for the remainder. Dividing one whole number by another gives a whole number, so `7 / 2` is `3`, while `7.0 / 2` is `3.5`. Shortcuts save typing: `count++` adds one to `count`, `count--` takes one away, and `total += 5` adds five to `total`. Keep an eye on each type's range: on the Uno R3, adding 1 to an `int` that holds 32,767 cannot give 32,768, because the value overflows and the result is wrong.

Making decisions with if and else

Compare values with `==` (equal to), `!=`, `<`, `>`, `<=` and `>=`, and join conditions with `&&` (and), `||` (or) and `!` (not). A single `=` assigns a value; it does not compare.

CPP

```
const int LED_PIN = 8;
int lightLevel = 300; // imagine this came from a sensor

void setup() {
  pinMode(LED_PIN, OUTPUT);
  Serial.begin(9600);
  if (lightLevel < 200) {
    Serial.println("Dark");
    digitalWrite(LED_PIN, HIGH);
  } else if (lightLevel < 600) {
    Serial.println("Dim");
  } else {
    Serial.println("Bright");
  }
}

void loop() {
}
```

With 300 stored, this prints "Dim". Only the first true branch runs.

for loops

A `for` loop repeats a block a counted number of times. Its brackets hold a start, a condition and a step:

CPP

```
const int LED_PIN = 8;

void setup() {
  pinMode(LED_PIN, OUTPUT);
  for (int i = 1; i <= 5; i++) {
    digitalWrite(LED_PIN, HIGH);
    delay(300);
    digitalWrite(LED_PIN, LOW);
    delay(300);
  }
}

void loop() {
}
```

The LED blinks exactly five times after each reset and then stays off, because the loop is inside `setup()`.

while loops

A `while` loop repeats for as long as its condition is true, so it suits jobs where you do not know in advance how many times:

CPP

```
const int BUTTON_PIN = 2;

void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  Serial.begin(9600);
  Serial.println("Press the button to start");
  while (digitalRead(BUTTON_PIN) == HIGH) {
    // wait here until the button is pressed
  }
  Serial.println("Started!");
}

void loop() {
}
```

Remember that `loop()` already repeats for ever, so you rarely need an endless loop of your own. A `while` whose condition can never become false freezes the sketch.

KEY POINTS

- On the Uno R3 an int is 16 bits (–32,768 to 32,767); use long or unsigned long for bigger values.
- == compares two values, while = assigns a value.
- if, else if and else choose between actions, and && and || combine conditions.
- for repeats a counted number of times; while repeats as long as a condition is true.
- Use const for values, such as pin numbers, that never change.

TRY THESE EXERCISES

1. Write a sketch that blinks an LED the number of times stored in a variable called count.
2. Use if and else to print "Too close" when a variable called distance is below 20 and "Clear" otherwise, and test it with several values.
3. Make running lights on four LEDs connected to pins 4 to 7, each with its own resistor, using a for loop.

CHAPTER 5

Analog Input, PWM Output and Functions

analogRead: measuring a voltage

The Uno's analog pins A0 to A5 feed a 10-bit analog-to-digital converter. `analogRead(A0)` returns a number from 0 (0 V) to 1023 (5 V), so each step is about 4.9 mV. To turn a reading into volts, use `reading * 5.0 / 1023.0`.

A potentiometer, with its outer legs on 5V and GND and its middle leg on A0, gives a reading that changes smoothly as you turn the knob. An LDR in a voltage divider gives a reading that changes with the light.

analogWrite: PWM output

An Uno cannot output a true in-between voltage. Instead, `analogWrite(pin, value)` switches a PWM pin on and off very fast, about 490 or 980 times a second depending on the pin. The value runs from 0 (always off) to 255 (always on); 127 is on for about half the time. The fraction of time spent on is the **duty cycle**. A lower duty cycle makes an LED look dimmer and a motor run slower.

`analogWrite` works only on the PWM pins 3, 5, 6, 9, 10 and 11. It does not need `pinMode`, although adding it does no harm.

The map() function

`map(value, fromLow, fromHigh, toLow, toHigh)` rescales a number from one range to another using whole-number maths. Turning a 0 to 1023 reading into a 0 to 255 brightness is a typical use:

CPP

```
const int POT_PIN = A0;
const int LED_PIN = 9;  // a PWM pin

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

void loop() {
  int reading = analogRead(POT_PIN);
  int brightness = map(reading, 0, 1023, 0, 255);
  analogWrite(LED_PIN, brightness);
  Serial.println(reading);
  delay(20);
}
```

Writing your own functions

A function is a named block of code you can call whenever you need it. It keeps `loop()` short and saves copying the same lines.

```

const int LED_PIN = 8;

void blinkTimes(int times, int gapMs) {
  for (int i = 0; i < times; i++) {
    digitalWrite(LED_PIN, HIGH);
    delay(gapMs);
    digitalWrite(LED_PIN, LOW);
    delay(gapMs);
  }
}

float readVolts(int pin) {
  return analogRead(pin) * 5.0 / 1023.0;
}

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

void loop() {
  blinkTimes(3, 200);
  Serial.println(readVolts(A0));
  delay(1000);
}

```

`void` means a function returns nothing. `readVolts` returns a `float`, which the caller can print or store. The names in the brackets, such as `times` and `gapMs`, are **parameters**.

The Arduino IDE adds function declarations for you, so in a `.ino` sketch a function can be written above or below the place where it is called.

KEY POINTS

- `analogRead` returns 0 to 1023 for 0 V to 5 V on the Uno.
- `analogWrite` outputs PWM with a value from 0 to 255, only on the pins marked ~.
- The duty cycle is the fraction of time a PWM signal is on.
- `map()` rescales a value from one range to another.
- Functions take parameters, can return a value and keep `loop()` short.

TRY THESE EXERCISES

1. Connect a 10 kΩ potentiometer to A0 and print the readings while turning the knob from one end to the other.
2. Make an LED on pin 9 fade up from 0 to 255 and back down again using two for loops and analogWrite.
3. Write a function called isDark() that returns true when the reading from an LDR divider on A0 crosses a threshold you find by testing.

CHAPTER 6

Reading Sensors and Driving Motors with the L298N

Reading the HC-SR04 ultrasonic sensor

Connect VCC to 5V, GND to GND, Trig to pin 2 and Echo to pin 4. `pulseIn()` measures how long the Echo pin stays HIGH, in microseconds.

CPP

```
const int TRIG_PIN = 2;
const int ECHO_PIN = 4;

float readDistanceCm() {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH); // 10 microsecond trigger pulse
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    unsigned long duration = pulseIn(ECHO_PIN, HIGH, 30000UL);
    return duration * 0.0343 / 2; // there and back, so halve it
}

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

void loop() {
    Serial.println(readDistanceCm());
    delay(200);
}
```

The third argument to `pulseIn` is a time-out of 30,000 μ s. A result of 0 means no echo came back in time, usually because nothing was in range. An IR module is simpler: read its output with `digitalRead`, exactly like a button.

Why a motor driver?

A pin gives about 20 mA; even a small BO motor needs many times that while running, and far more as it starts or when its wheel is blocked. The **L298N** module contains two H-bridges, so it can run two DC motors, each forwards or backwards, from a separate battery.

An **H-bridge** is four electronic switches arranged around the motor like the letter H. Closing one diagonal pair sends current through the motor one way; closing the other pair sends it the opposite way. The Arduino only tells the bridge which pair to close, so its pins never carry the motor current. The L298N loses a couple of volts inside the chip, more at higher currents, so the motors always receive somewhat less than the battery voltage.

L298N terminal	Connect to
Motor supply (often marked 12V)	Positive of the motor battery
GND	Battery negative and Arduino GND
OUT1, OUT2	Motor A
ENA, jumper removed	Pin 5 (PWM)
IN1, IN2	Pins 7 and 8

IN1	IN2	Motor A
HIGH	LOW	Forward
LOW	HIGH	Backward
LOW	LOW	Stops

`analogWrite` on ENA sets the speed: 255 is full speed, and at low values the motor may hum without turning.

```

const int ENA = 5;
const int IN1 = 7;
const int IN2 = 8;

void motorA(int speed) { // -255 to 255; negative is backward
  if (speed > 0) {
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
  } else if (speed < 0) {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
  } else {
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
  }
  analogWrite(ENA, constrain(abs(speed), 0, 255));
}

void setup() {
  pinMode(ENA, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
}

void loop() {
  motorA(200);
  delay(2000);
  motorA(0);
  delay(1000);
  motorA(-200);
  delay(2000);
  motorA(0);
  delay(1000);
}

```

A second motor uses ENB, IN3 and IN4 in the same way, for example on pins 6, 11 and 12.

Join the L298N's GND to the Arduino's GND. Without a common ground, the IN and EN signals have no reference and the motors behave unpredictably.

KEY POINTS

- `pulseIn()` measures how long the HC-SR04 Echo pin stays HIGH, in microseconds.
- Distance in cm = duration × 0.0343 ÷ 2, and a result of 0 means no echo arrived in time.
- The L298N's two H-bridges run two DC motors forwards or backwards from a separate battery.
- IN1 and IN2 set the direction; `analogWrite` on ENA sets the speed.
- The motor driver and the Arduino must share a common ground.

TRY THESE EXERCISES

1. Wire the HC-SR04 and print distances to the Serial Monitor while moving your hand closer and further away, comparing the readings with a ruler.
2. Connect one BO motor through the L298N and run the motorA() sketch, then change the speeds and timings.
3. Add a second motor on ENB, IN3 and IN4 and write a motorB() function so that both wheels can be driven.

CHAPTER 7

The Servo Library and Debugging Your Robot

Controlling a servo

The Servo library comes with the Arduino IDE. It produces the control pulses for you, so you only give an angle.

CPP

```
#include <Servo.h>

Servo scanner;          // create a Servo object

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

void loop() {
  scanner.write(0);     // turn to 0 degrees
  delay(1000);
  scanner.write(90);    // centre
  delay(1000);
  scanner.write(180);
  delay(1000);
}
```

- `#include <Servo.h>` makes the library available to the sketch.
- `attach(pin)` connects the object to the signal pin.
- `write(angle)` takes an angle from 0 to 180 degrees.
- `read()` returns the last angle written.

Wire brown or black to GND, red to 5V and orange or yellow to the signal pin. One small SG90 with no load can run from the Uno's 5V pin for practice, but several servos, or a servo under load, need a separate 5 V supply with its GND joined to the Arduino's GND.

On the Uno R3, using the Servo library stops `analogWrite()` working on pins 9 and 10, whether or not a servo is connected to them. Keep the motor speed pins on 5 and 6, as in the previous lesson.

Debugging: finding and fixing mistakes

Bugs come in three kinds, and each needs a different approach.

Compile errors

The IDE refuses to build the sketch and shows the error in the Output panel at the bottom. Read the **first** error first, because one mistake can cause a list of others. Two common messages:

- `expected ';' before ...` means a semicolon is missing, usually at the end of the line before the one named.
- `'ledpin' was not declared in this scope` usually means a spelling or capital-letter mismatch, such as `ledpin` for `ledPin`, or a variable used outside the function where it was declared.

Upload errors

The sketch compiles but will not upload. Check the board and port, try a USB cable you know carries data, close any other program using the port, and disconnect anything on pins 0 and 1.

Logic errors

The program runs but does the wrong thing. This is where the Serial Monitor earns its place:

1. Print sensor readings and key variables to see what the board actually sees.
2. Print a short message in each branch, such as "turning left", to see which part of an `if` ran.
3. Test one part at a time: the motors alone, then each sensor alone, then everything together.
4. Check the wiring against a diagram. Loose jumpers, a missing common ground and swapped motor wires cause more faults than code does.

If the board resets whenever the motors start, the battery is too weak or is shared badly with the Arduino. Use fresh cells, a separate motor supply and a common ground.

KEY POINTS

- Include the library with `#include <Servo.h>`, create a Servo object, attach it to a pin and write an angle from 0 to 180.
- On the Uno R3 the Servo library disables `analogWrite` on pins 9 and 10.
- Power several servos, or a loaded servo, from a separate 5 V supply with a common ground.
- Compile, upload and logic errors each need a different approach; read the first compile error first.
- Use `Serial.println` to see what the board sees, and test one part at a time.

TRY THESE EXERCISES

1. Upload the servo sketch, then change it to sweep smoothly from 0 to 180 degrees one degree at a time using a for loop.
2. Delete a semicolon and misspell a variable on purpose, click Verify, and note the exact error messages the IDE shows.
3. Mount the HC-SR04 on the servo and print the distance at 0, 90 and 180 degrees to find the direction with the most space.

CHECK YOURSELF

Practice questions

Choose the best answer for each question. The answers, with a short explanation, are in the next chapter.

1. Which microcontroller is used on the Arduino Uno R3?

- a) ATmega2560
- b) ATmega328P
- c) ESP8266
- d) Intel 8085

2. Which pins of an Arduino Uno R3 can produce PWM with `analogWrite`?

- a) 0, 1, 2, 3, 4 and 5
- b) A0, A1, A2, A3, A4 and A5
- c) 3, 5, 6, 9, 10 and 11
- d) 2, 4, 7, 8, 12 and 13

3. Why should beginners leave digital pins 0 and 1 unconnected?

- a) They supply 12 V, which can damage most parts
- b) They are permanently connected to the built-in LED
- c) They can only be used as analog inputs
- d) They are the serial lines used for uploads

4. About how much current can one I/O pin of an Uno R3 safely supply?

- a) About 20 mA
- b) About 500 mA
- c) About 2 A
- d) About 5 A

- 5. Every module connected to an Arduino must share which connection with it?**
- a) The 3.3V pin
 - b) Pin 13
 - c) GND, a common ground
 - d) The reset button
- 6. What does the Verify button in the Arduino IDE do?**
- a) Uploads the sketch to the board without checking it
 - b) Compiles the sketch and reports any errors
 - c) Opens the Serial Monitor at 9600 baud
 - d) Erases the program stored on the board
- 7. How many times does the setup() function run?**
- a) Once, each time the board powers up or is reset
 - b) Once every second while the board is powered
 - c) Again and again, for as long as the board has power
 - d) Only when a button connected to pin 2 is pressed
- 8. Which line makes digital pin 8 an output?**
- a) `digitalWrite(8, OUTPUT);`
 - b) `pinMode(OUTPUT, 8);`
 - c) `analogWrite(8, OUTPUT);`
 - d) `pinMode(8, OUTPUT);`
- 9. What does delay(1000); do in a sketch?**
- a) Pauses the program for 1000 seconds
 - b) Repeats the next line 1000 times
 - c) Pauses the program for 1 second
 - d) Sets the clock speed to 1000 Hz
- 10. What file extension does an Arduino sketch use?**
- a) .ard
 - b) .ino
 - c) .exe
 - d) .uno
- 11. A button is wired between pin 2 and GND, and pin 2 is set to INPUT_PULLUP. What does digitalWrite(2) return while the button is pressed?**
- a) LOW
 - b) HIGH
 - c) 1023
 - d) It changes at random
- 12. What is a floating input?**
- a) An input pin that has been set to INPUT_PULLUP mode
 - b) An input wired straight to the 5 V pin of the board
 - c) An unconnected input that reads at random
 - d) An input that can read analog voltages only
- 13. The Serial Monitor shows strange, random characters. What is the most likely cause?**
- a) The USB cable is longer than one metre
 - b) Pin 13 is being used by the built-in LED
 - c) The sketch does not contain any comments
 - d) A baud rate mismatch with Serial.begin()
- 14. What is the difference between Serial.print() and Serial.println()?**
- a) Serial.print() works only with numbers, not text
 - b) println() adds a new line after the value
 - c) Serial.println() sends its data to a printer
 - d) Serial.print() is faster because it skips the baud rate

15. Why can one press of a mechanical button be read as several presses?

- a) The internal pull-up resistor is too strong
- b) The Serial Monitor repeats every message
- c) The contacts bounce for a few milliseconds
- d) The Arduino counts each press twice by design

16. What range of values can an int hold on an Arduino Uno R3?

- a) -32,768 to 32,767
- b) 0 to 255
- c) 0 to 1023
- d) -2,147,483,648 to 2,147,483,647

17. Which type should be used to store the value returned by millis()?

- a) int
- b) char
- c) bool
- d) unsigned long

18. Which operator checks whether two values are equal in an if statement?

- a) =
- b) ==
- c) =>
- d) !=

19. How many times does the body of for (int i = 1; i <= 5; i++) run?

- a) 4
- b) 5
- c) 6
- d) It never stops

20. When does a while loop stop repeating?

- a) After exactly ten repeats, by default
- b) When the board is reset only
- c) When its condition becomes false
- d) When loop() reaches its closing brace

21. What range of values does analogRead() return on an Arduino Uno?

- a) 0 to 255
- b) 0 to 100
- c) 0 to 5
- d) 0 to 1023

22. What range of values does analogWrite() accept?

- a) 0 to 255
- b) 0 to 1023
- c) 0 to 5
- d) -255 to 255

23. Roughly what duty cycle does analogWrite(9, 127) produce?

- a) 12.7%
- b) 25%
- c) 50%
- d) 100%

24. What value does map(512, 0, 1023, 0, 255) return?

- a) 127
- b) 512
- c) 255
- d) 2

25. What does the word void mean in front of a function's name?

- a) The function takes no parameters
- b) The function runs only once
- c) The function is empty
- d) The function returns no value

26. Which function measures how long the HC-SR04's Echo pin stays HIGH?

- a) analogRead()
- b) pulseIn()
- c) millis()
- d) digitalWrite()

- 27. In the distance calculation, why is the echo duration multiplied by 0.0343 and divided by 2?**
- a) 0.0343 converts volts to centimetres, and 2 is the number of pins
 - b) 0.0343 is the speed of sound in cm per μ s, and the sound travels there and back
 - c) 0.0343 is the sensor's error, and the reading is averaged over two pulses
 - d) Both numbers convert milliseconds into seconds
- 28. Why is a motor driver such as the L298N used between an Arduino and a DC motor?**
- a) It makes the motor run more quietly and smoothly
 - b) It converts the DC motor into a servo motor
 - c) It stores the sketch so the motor can run alone
 - d) It switches the motor current and can reverse it
- 29. IN1 HIGH with IN2 LOW makes motor A run forward on an L298N. What does IN1 LOW with IN2 HIGH do?**
- a) Runs motor A backward
 - b) Runs motor A forward at double speed
 - c) Turns on motor B
 - d) Disconnects the motor battery
- 30. How is the speed of a motor controlled through the L298N?**
- a) By changing the delay() value inside setup()
 - b) By swapping the OUT1 and OUT2 motor wires
 - c) With analogWrite on ENA, its jumper removed
 - d) By connecting the motor directly to pin 13
- 31. Which line creates a Servo object called scanner?**
- a) Servo.begin(scanner);
 - b) Servo scanner;
 - c) pinMode(scanner, SERVO);
 - d) #define scanner Servo
- 32. On an Uno R3, using the Servo library disables analogWrite() on which pins?**
- a) 3 and 11
 - b) 5 and 6
 - c) 9 and 10
 - d) 0 and 1
- 33. After scanner.attach(9);, what does scanner.write(90); do?**
- a) Spins the servo continuously at speed 90
 - b) Waits 90 milliseconds before moving
 - c) Moves the servo forward by 90 steps
 - d) Turns it to 90 degrees, the centre
- 34. The IDE reports "'ledpin' was not declared in this scope". What is the most likely cause?**
- a) The board is not connected to the computer
 - b) A semicolon is missing at the end of the sketch
 - c) The Serial Monitor window is still open
 - d) A spelling or capital-letter mismatch in the name
- 35. A sketch compiles and uploads, but the robot does the wrong thing. What kind of bug is this?**
- a) A logic error
 - b) A compile error
 - c) An upload error
 - d) A missing library error

Answers

1. **(b) ATmega328P.** The Uno R3 is built around the ATmega328P; the ATmega2560 is used on the larger Arduino Mega.
2. **(c) 3, 5, 6, 9, 10 and 11.** The six PWM pins are 3, 5, 6, 9, 10 and 11, marked on the board with a ~.
3. **(d) They are the serial lines used for uploads.** Pins 0 (RX) and 1 (TX) are the serial lines shared with USB, so anything connected to them can disturb uploads and serial communication.
4. **(a) About 20 mA.** Around 20 mA per pin is the safe working figure, which is enough for an LED with a resistor but not for a motor.
5. **(c) GND, a common ground.** Signals are measured against ground, so all parts of the circuit need a common GND to work reliably.
6. **(b) Compiles the sketch and reports any errors.** Verify compiles the code and lists any errors; Upload compiles and then sends the program to the board.
7. **(a) Once, each time the board powers up or is reset.** `setup()` runs once at start-up or after a reset, and then `loop()` repeats for ever.
8. **(d) `pinMode(8, OUTPUT);`.** `pinMode` takes the pin number first and the mode second.
9. **(c) Pauses the program for 1 second.** `delay` takes milliseconds, and 1000 milliseconds is one second.
10. **(b) `.ino`.** Sketches are saved as `.ino` files inside a folder with the same name.
11. **(a) LOW.** The internal pull-up holds the pin HIGH, and pressing the button connects it to GND, so it reads LOW.
12. **(c) An unconnected input that reads at random.** An unconnected input picks up electrical noise, so its readings jump between HIGH and LOW.
13. **(d) A baud rate mismatch with `Serial.begin()`.** The board and the Serial Monitor must use the same baud rate, or the received text is garbled.
14. **(b) `println()` adds a new line after the value.** Both send the value, but `println` adds a line ending so the next output starts on a new line.
15. **(c) The contacts bounce for a few milliseconds.** Mechanical contacts bounce open and closed briefly, so a sketch needs a short delay or a debounce check.
16. **(a) -32,768 to 32,767.** On the Uno R3 an `int` is 16 bits; the larger range belongs to a 32-bit `long`, which is also the size of an `int` on 32-bit boards.
17. **(d) unsigned long.** `millis()` returns an unsigned `long`, which can hold the large, ever-growing count of milliseconds.
18. **(b) `==`.** `==` compares two values; a `single =` assigns a value instead.
19. **(b) 5.** `i` takes the values 1, 2, 3, 4 and 5, so the body runs five times.
20. **(c) When its condition becomes false.** A `while` loop checks its condition before every repeat and stops as soon as it is false.
21. **(d) 0 to 1023.** The Uno's 10-bit converter gives 1024 steps, from 0 for 0 V to 1023 for 5 V.
22. **(a) 0 to 255.** `analogWrite` takes a value from 0 (always off) to 255 (always on).

- 23. (c) 50%.** 127 is about half of 255, so the pin is on for about half of each cycle.
- 24. (a) 127.** map rescales 512 from the 0 to 1023 range into 0 to 255; with whole-number maths the result is 127.
- 25. (d) The function returns no value.** void is the return type meaning that the function returns nothing to its caller.
- 26. (b) pulseIn().** pulseIn(pin, HIGH) waits for the pin to go HIGH and returns how many microseconds it stayed HIGH.
- 27. (b) 0.0343 is the speed of sound in cm per µs, and the sound travels there and back.** Sound travels about 0.0343 cm per microsecond, and the time covers the trip to the object and back, so it is halved.
- 28. (d) It switches the motor current and can reverse it.** A pin cannot supply a motor's current; the L298N's H-bridges switch the battery current and can reverse it.
- 29. (a) Runs motor A backward.** Swapping the two input levels reverses the current through the motor, so it turns the other way.
- 30. (c) With analogWrite on ENA, its jumper removed.** PWM on the enable pin switches the motor on and off rapidly; a higher value gives a higher speed.
- 31. (b) Servo scanner;.** After #include <Servo.h>, the declaration Servo scanner; creates an object that can be attached to a pin.
- 32. (c) 9 and 10.** On the Uno R3 the Servo library uses the timer behind pins 9 and 10, so PWM stops working on those two pins.
- 33. (d) Turns it to 90 degrees, the centre.** write() takes an angle from 0 to 180 degrees, so 90 sends a standard servo to the middle of its range.
- 34. (d) A spelling or capital-letter mismatch in the name.** The compiler cannot find a variable called ledpin, which usually means a spelling or case mismatch such as ledPin.
- 35. (a) A logic error.** The code is valid and reached the board, so the mistake is in what the program does, which is a logic error.

WORD LIST

Glossary

analogRead()

The function that reads the voltage on an analog pin and returns a number from 0 to 1023 on the Uno.

analogWrite()

The function that outputs a PWM signal with a value from 0 to 255 on a PWM pin.

Arduino IDE

The free program used to write, verify and upload Arduino sketches.

Baud rate

The speed of serial communication in bits per second, such as 9600.

Bootloader

A small program kept on the microcontroller that lets new sketches be uploaded over USB.

Bug

A mistake in a program or a circuit that makes it behave wrongly.

Compile

To translate a sketch's C++ code into machine code that the microcontroller can run.

Debugging

Finding and fixing the causes of errors in a program or a circuit.

digitalRead()

The function that returns HIGH or LOW from a digital input pin.

digitalWrite()

The function that sets a digital output pin HIGH or LOW.

Duty cycle

The fraction of each cycle for which a PWM signal stays on.

Floating input

An input pin connected to nothing, which gives random readings.

Function

A named block of code that can take parameters and return a value.

GND

Ground, the 0 V reference point that every part of a circuit must share.

H-bridge

A circuit of four switches that can drive a DC motor in either direction.

INPUT_PULLUP

A pin mode that switches on the microcontroller's built-in pull-up resistor.

Library

A ready-made collection of code, such as Servo, that adds features to a sketch.

loop()

The function that runs over and over after setup() has finished.

Microcontroller

A small computer on a single chip with its own processor, memory and input/output pins.

PWM

Pulse-width modulation, which switches a pin on and off rapidly to imitate an in-between voltage.

Serial Monitor

The IDE window that shows text sent by the board and can send text back to it.

setup()

The function that runs once when the board starts or is reset.

Sketch

An Arduino program, saved as a .ino file.

Upload

To send a compiled sketch from the computer to the board's memory.

Variable

A named storage place that holds a value of a given type.



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.