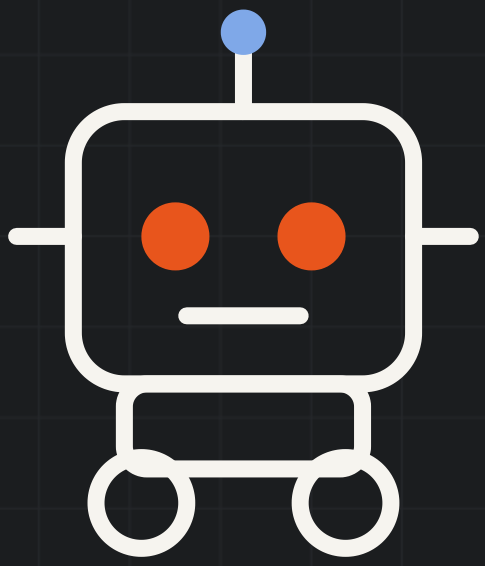
// TINYBOT FORGE · ESP32 ROBOTICS

ESP32 Robots
10 Projects

From a pair of breathing LED eyes to a phone-controlled robot arm.
Every sketch compile-checked. Every step with a wiring table.



TinyBot Forge · tinybotforge.com · First edition 2026

How to use this book

There are ten projects, easiest first. Projects 1 and 2 need only an ESP32 and a few small parts. Project 3 builds a small car, and the projects after it add to that same car — so **work through them in order**.

Every chapter follows the same pattern: **goal → parts → how it works → wiring → code → walkthrough → tuning → challenges → troubleshooting**. Open the sketches from the code bundle rather than copying from the PDF.

> **Safety:** every ESP32 pin is 3.3 V; more than that damages the pin. Every place that needs a voltage divider is marked — don't skip them. Use a protected holder for 18650 cells and never short them.

Contents

Toolchain (one-time setup)

1. Install Arduino IDE 2 (arduino.cc).
2. File → Preferences → Additional boards manager URLs:
   `https://espressif.github.io/arduino-esp32/package_esp32_index.json`
3. Boards Manager → search **esp32** → install Espressif Systems, version 3.0 or newer.
4. Tools → Board → **ESP32 Dev Module**; pick the serial port. No port? CP2102 boards need the Silicon Labs driver, CH340 boards need the WCH driver.

5. "Failed to connect" on upload: hold the BOOT button until *Connecting...* appears, then release.

> **Watch the version:** ESP32 core 3.x uses `ledcAttach(pin, freq, bits)` and `ledcWrite(pin, duty)`. Older tutorials use `ledcSetup` / `ledcAttachPin`, which no longer compile.

Complete parts list

Part	Qty	Used in
ESP32 DevKit V1 (30 or 38 pin)	1	All
USB cable that carries data (not charge-only)	1	All
Breadboard, jumper wires (M-M, M-F, F-F)	1 pack each	All
LEDs, 220 Ω resistors, push buttons	2–4 each	1
SG90 servos (MG90S is an upgrade)	2	2, 9, 10
HC-SR04 ultrasonic sensor	1	2, 3, 5, 9
1 kΩ, 2 kΩ, 10 kΩ resistors	2 each	2, 3, 5, 8, 9
L298N motor driver	1	3–8
2WD chassis kit (2 TT motors, caster)	1	3–8
2× 18650 Li-ion cells + holder	1 set	3–8
TCRT5000 IR line sensor modules	3	6, 7
Photoresistors (LDRs)	2	8
100–470 µF electrolytic capacitors	2	2, 3, 10
5 V 2 A external supply	1	10
Black electrical tape, white poster board	1	6, 7

1 Breathing Robot Eyes

LEDs, PWM and a button

Level ★☆☆☆☆ | Time ~30 min | Sketch breathing-eyes.ino

Build a pair of robot eyes that slowly brighten and dim as if breathing, and blink twice when you press a button. This is the foundation for everything after it: output and input.

Parts

Part	Qty
ESP32 DevKit V1	1
LEDs (any color)	2
220 Ω resistors	2
Push button	1
Breadboard and jumper wires	

How it works

PWM (pulse-width modulation): an ESP32 pin can only output 0 V or 3.3 V — never "half". But switch it thousands of times a second, on 50% of the time, and an LED looks half as bright. The on-fraction is the **duty cycle**.

ESP32 core 3.x sets up PWM in two lines:

```
ledcAttach(pin, frequency, resolution);   // 8-bit resolution → brightness 0–255
ledcWrite(pin, brightness);
```

INPUT_PULLUP: wire the button between a GPIO and GND. With the internal pull-up enabled, the pin reads HIGH when released and LOW when pressed. No external resistor.

Why square the brightness? Your eyes aren't linear: going from 0 to 50 looks like a big change, 200 to 250 barely registers. Squaring a 0–1 value before scaling to 255 makes the breathing look natural.

Wiring

From	To	Notes
GPIO 16	220 Ω → left LED long leg	Short leg to GND
GPIO 17	220 Ω → right LED long leg	Short leg to GND
GPIO 21	One side of the button	Other side to GND

Code

The complete file is in the code bundle: `breathing-eyes.ino`. Compiled with the Arduino ESP32 board package 3.3.12, all warnings enabled, 0 warnings.

```
/*
 * TinyBot Forge — "ESP32 Robots: 10 Projects", Project 1: Breathing Robot Eyes
 * Two LEDs are the robot's eyes: they slowly "breathe", and blink when you press the button.
 * You learn: digital output, PWM brightness, INPUT_PULLUP buttons, non-blocking timing with millis()
 * Toolchain: Arduino IDE 2.x + ESP32 board package 3.x
 */
const int PIN_EYE_L = 16;
const int PIN_EYE_R = 17;
const int PIN_BUTTON = 21;       // button between GPIO21 and GND
const int PWM_FREQ = 5000, PWM_BITS = 8;

const unsigned long BREATH_MS = 3000;   // one breath cycle
const unsigned long BLINK_MS = 120;     // eyes-closed time

void setEyes(int brightness) {
  ledcWrite(PIN_EYE_L, brightness);
  ledcWrite(PIN_EYE_R, brightness);
}

void setup() {
  pinMode(PIN_BUTTON, INPUT_PULLUP);
  ledcAttach(PIN_EYE_L, PWM_FREQ, PWM_BITS);
  ledcAttach(PIN_EYE_R, PWM_FREQ, PWM_BITS);
}

void loop() {
  unsigned long now = millis();

  if (digitalRead(PIN_BUTTON) == LOW) {   // pressed: blink twice
    for (int i = 0; i < 2; i++) {
      setEyes(0);   delay(BLINK_MS);
      setEyes(255); delay(BLINK_MS);
    }
    while (digitalRead(PIN_BUTTON) == LOW) delay(10);  // wait for release
    return;
  }

  // Breathing: a smooth cosine wave; eyes perceive brightness non-linearly, so squaring looks more n
atural
  float phase = (now % BREATH_MS) / (float)BREATH_MS;          // 0–1
  float s = (1.0 - cos(phase * 2.0 * PI)) / 2.0;               // 0–1–0
  int brightness = (int)(8 + 247 * s * s);
  setEyes(brightness);
```

```
  delay(10);
}
```

Code walkthrough

- `millis() % BREATH_MS` tells you where you are in the breathing cycle without a blocking `delay`.
- `(1 - cos(x)) / 2` produces a smooth 0 → 1 → 0 curve.
- The minimum brightness is 8, not 0, so the eyes never fully go dark — they look "alive".
- After a press, the `while` loop waits for release so one press gives one blink.

Tuning

If you want to…	Change
Breathing too fast or slow	`BREATH_MS` (3000 = one breath every 3 s)
Blink too quick	`BLINK_MS`

Challenges

1. Offset the two eyes by half a cycle so the robot appears to wink.
2. Hold the button for 2 s to toggle an "angry mode" with fast flashing.
3. Use an RGB LED and shift color as it breathes.

Troubleshooting

The LED never lights
LEDs are polarized: the long leg (anode) goes toward the resistor/GPIO side.

The button does nothing
Use diagonally opposite legs. On a 4-leg button, the two legs on the same side are already connected.