# LED, PWM und Zeitsteuerung

LED sicher anschliessen, ohne delay() blinken und mit dem aktuellen Arduino-ESP32-Core dimmen.

**ACHTUNG**

Eine LED braucht einen Strombegrenzungswiderstand. Fuer typische Unterrichtsaufbauten sind 220-330 Ohm und etwa 5-10 mA ein robuster Start. GPIO-Grenzwerte sind kein Betriebsziel.

## SCHALTUNG

```
GPIO18 -- 330 Ohm --|>|-- GND
                    LED

langes Bein  = Anode (+)
kurzes Bein = Kathode (-)

Widerstand kann vor oder nach
der LED liegen - entscheidend ist
die Reihenschaltung.
```

## BLINK MIT delay()

```
const int LED_PIN = 18;

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

void loop() {
  digitalWrite(LED_PIN, HIGH);
  delay(500);
  digitalWrite(LED_PIN, LOW);
  delay(500);
}
```

Gut zum Start, aber waehrend delay() wird keine andere Aufgabe bearbeitet.

| Ziel | Befehl | Hinweis |
|---|---|---|
| **Pin vorbereiten** | pinMode(LED_PIN, OUTPUT); | In setup() einmal ausfuehren. |
| **LED schalten** | digitalWrite(..., HIGH/LOW); | Bei externer LED gilt die dargestellte Schaltung; Onboard-LED kann invertiert sein. |
| **Warten** | delay(ms); | Blockiert. Nur fuer einfache Einzelaufgaben verwenden. |
| **Zeit lesen** | uint32_t now = millis(); | Seit Start vergangene Millisekunden; Differenzen bilden. |
| **PWM dimmen** | analogWrite(pin, 0...255); | 0 = aus, 255 = volle Einschaltdauer; aktuelle Arduino-ESP32-API. |

**0 %**

immer aus PWM-Tastgrad 0

**25 %**

kurz an, lange aus wirkt eher dunkel

**50 %**

gleich lang an/aus mittlere Helligkeit

**100 %**

immer an volle Einschaltdauer

**MERKEN**

PWM erzeugt keine beliebige Gleichspannung, sondern schaltet sehr schnell. Fuer Motoren, Relais oder LED-Streifen ist ein geeigneter Treiber erforderlich.

# LED, PWM und Zeitsteuerung

LED sicher anschliessen, ohne delay() blinken und mit dem aktuellen Arduino-ESP32-Core dimmen.

## BLINKEN OHNE BLOCKIEREN

```
const int LED_PIN = 18;
const uint32_t INTERVAL = 500;

uint32_t previous = 0;
bool ledState = false;

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

void loop() {
  uint32_t now = millis();

  if (now - previous >= INTERVAL) {
    previous = now;
    ledState = !ledState;
    digitalWrite(LED_PIN, ledState);
  }

  // Taster/Sensor/WLAN koennen hier weiterlaufen.
}
```

## PWM MIT analogWrite()

```
const int LED_PIN = 18;

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

void loop() {
  int poti = analogRead(34);
  int duty = map(poti, 0, 4095,
                 0, 255);
  analogWrite(LED_PIN, duty);
}
```

Bei anderer Core-Version die zugehoerige LEDC-API-Dokumentation pruefen.

| Fehlerbild | Ursache | Pruefung |
|---|---|---|
| **LED bleibt dunkel** | Polung, GND, falscher GPIO | LED drehen; Pinout; digitalWrite-Test |
| **LED sehr hell/heiss** | Widerstand fehlt/zu klein | Sofort trennen; Widerstand berechnen |
| **Andere Aufgaben stoppen** | Lange delay()-Kette | Zustandsautomat und millis()-Differenz |
| **PWM ohne Wirkung** | Pin/Core/API oder Onboard-LED | Externe LED an geeignetem Ausgang testen |
| **Board startet schlecht** | LED an Strapping-Pin | Fuer Einsteiger GPIO18/19/21/22/23 nutzen |

### SIGNAL

- Einzelne LED
- Statusausgang
- Widerstand Pflicht

### TREIBER

- Transistor/MOSFET
- Separate Lastversorgung
- Gemeinsame GND

### SCHUTZ

- Freilaufdiode bei Spulen
- Strom/Spannung messen
- Datenblatt beachten

**PRAXIS**

Erst digitales Ein/Aus testen, dann PWM. Bei mehreren parallelen Aufgaben jeden Ablauf als kleinen Zustand mit eigener Zeitmarke planen.