LIBRARY · REFERENCE GUIDE

POWER MODES AND SLEEP

Sleep modes shut down parts of a microcontroller to cut current from milliamps to microamps. Why sleep current dominates battery life, with a runtime calculator.

ONE THOUSAND DRONES ENGINEERING TEAM · VERIFIED 2026-07

A microcontroller does not have to run flat out. Sleep modes shut down parts of the chip to cut its current from milliamps down to microamps, and on a battery that is the difference between hours and months. A board that spends almost all of its time asleep can run for a very long time on a small cell.

ACTIVE, LIGHT SLEEP, DEEP SLEEP

In active mode the CPU and radio run and the chip draws the most current. Light sleep pauses the CPU but keeps its state and memory alive, so it wakes quickly and picks up where it left off. Deep sleep powers down almost everything, keeping only a tiny always-on domain, which drops the current to microamps but loses most of the chip's state, so it effectively restarts on wake. Each step trades responsiveness for lower current.

- Espressif. ESP-IDF Programming Guide: Sleep Modes (light sleep, deep sleep, wake sources). docs.espressif.com

WHAT WAKES IT

A sleeping chip wakes on a source you configure ahead of time: a timer after a set interval, a pin changing level, or a peripheral event. A typical sensor node deep-sleeps on a timer, wakes, takes a reading, sends it over the radio, and goes back to sleep, spending a few milliseconds awake for every several seconds asleep.

WHY SLEEP DOMINATES BATTERY LIFE

When a board is awake for a few milliseconds and asleep for seconds at a stretch, the sleep current sets the average almost by itself, and the brief active current barely counts. Halving the deep-sleep current can nearly double the runtime, while shaving the active current barely moves the needle. That is why low-power design chases deeper, more frequent sleep; slowing the CPU down barely helps.

$$I_{avg} = (I_{active} \times t_{active} + I_{sleep} \times t_{sleep}) / (t_{active} + t_{sleep})$$

CALCULATOR · LIPO BATTERY RUNTIME CALCULATOR (ESP32 / MICROCONTROLLER)

Estimate runtime from battery capacity, average current draw, and usable capacity. Worked from a real ESP32-S3 board's measured budget.

Interactive calculator: academy.onethousanddrones.com/tools/lipo-battery-runtime

ESTIMATE RUNTIME FROM BATTERY CAPACITY AND AVERAGE CURRENT.

DEEP DIVE · WHAT STAYS POWERED IN DEEP SLEEP

On the ESP32 a small low-power domain stays alive through deep sleep: an RTC timer to keep time and wake the chip, a little RTC memory, and a few RTC-capable GPIOs that can serve as wake pins. Everything else, the main cores and most peripherals, is powered off, which is where the microamp current comes from. Because normal RAM is lost, you stash the handful of bytes you need to survive the restart (a counter, a state flag) in that RTC memory, and read them back on wake. (Espressif ESP-IDF sleep modes)

A WAKE-MEASURE-SLEEP CYCLE: THE LONG, LOW SLEEP FLOOR IS WHAT SETS THE BATTERY LIFE.

CHECKPOINT

1. What mostly determines a battery board's life if it sleeps most of the time?

a. Its sleep (idle) current

b. Its peak active current alone

c. Its CPU clock speed while awake

ANSWER · A

With the board asleep most of the time, the average current, and so the runtime, is set by the sleep current the board sits at almost all the time. The brief awake current and the clock speed barely add to it.

2. What is the main trade-off of deep sleep versus light sleep?

a. Deep sleep is always faster to wake from

b. Deep sleep uses far less current but loses most state and restarts on wake

c. Deep sleep keeps the radio fully powered

ANSWER · B

Deep sleep saves the most current by powering down almost everything, at the cost of losing state.

3. What can wake a sleeping ESP32?

a. Only physically unplugging and replugging it

b. Nothing, once it is asleep it stays asleep

c. A timer, a pin change, or a peripheral event

ANSWER · C

You configure a wake source before sleeping: a timer, a wake pin, or a peripheral event.

- Calculate it: the LiPo battery runtime calculator
- Next: reading the ESP32 pinout