Pclika

Developer Guide

From Board to AI — Complete Setup & Workflow

DOCUMENT · Pclika Developer Guide v0.1

PLATFORM · ESP32-S3 + pclika-bridge + MCP

VERSION · 0.1.0 · June 2026

CLIENTS · Claude Code · Codex · Cursor · VS Code · OpenCode

STACK · Python 3.10+ · ESP-IDF v5.x

CONTACT · pclika.com · starinvc@gmail.com

SEAL · PCK-MMXXVI-C4A32096

This guide walks you step by step from purchasing the hardware to issuing your first AI-driven hardware command. No hardware experience required.

00 OVERVIEW

Pclika turns an ESP32-S3 board into an AI-controlled hardware platform. AI coding assistants call hardware functions via standard MCP tool calls (JSON-RPC 2.0) over USB in real time — no firmware writing required after setup.

STEP	TITLE	DESCRIPTION
01	**Buy the Kit**	Order from official store or MCP self-service
02	**Connect Hardware**	USB-C to computer, I2C sensors to JST SH headers
03	**Flash Firmware**	One command: idf.py flash — ~90 seconds
04	**Install Bridge**	pip install pclika-bridge — done
05	**Verify Connection**	pclika-bridge --list-ports, then test device_info
06	**Configure AI Client**	Claude Code / Codex / Cursor / VS Code / OpenCode
07	**Load Tool Instructions**	Paste system prompt once into your AI client
08	**Describe Your Project**	Write a plain-language requirement — AI starts coding
09	**Iterate & Deploy**	AI writes firmware + MCP calls — you review and flash
10	**Scale Up**	Add modules, switch to SSE multi-client mode

01 BUY THE KIT

The easiest way to start is the **Pclika Starter Kit** — board and sensor pack, pre-tested with all examples. Order from either official channel below.

OFFICIAL PURCHASE CHANNELS

User Store (Official Shop)	**https://mall.pclika.com/**
MCP Self-Service Purchase	**https://hc.pclika.com/**

Starter Kit ¥299 (Recommended)

ITEM	QTY	NOTES
Pclika ESP32-S3 Baseboard v0.1	1 pc	Main MCU · USB-C · Wi-Fi · BLE
DHT22 Temp/Humidity Sensor	1 pc	I2C · env-monitor example
BH1750 Light Sensor	1 pc	I2C · lux measurement
USB-C Cable (1 m)	1 pc	Data cable — not charge-only
JST SH 4P Cable x3	3 pcs	I2C sensor connections

DIY / BUILD YOUR OWN

KiCad design files and JLCPCB order guide are in the GitHub repo. Minimum order: 5 boards.

OPTIONAL MODULES

MODULE	USE CASE
SG90 Servo x4	PWM headers — servo-control example
SSD1306 OLED 128x64	I2C display — display_text tool
OV2640 Camera	Camera — vision-snapshot example
RS-485 Adapter	Modbus RTU — industrial-gateway example
iCE40UP5K M.2 FPGA	FPGA expansion — HDL Bridge

02 CONNECT THE HARDWARE

PHYSICAL CONNECTIONS

PERIPHERAL	CONNECTION	NOTE
USB-C cable	Board J1 to computer USB	Power + serial link
DHT22/BME280	SDA=GPIO21 SCL=GPIO22 JST SH 4P	env-monitor example
SSD1306 OLED	Same I2C bus, address 0x3C	display_text tool
Servo	PWM0-PWM3 headers	servo_move tool

BOARD LED STATUS

LED STATE	MEANING
Solid blue	Power on — firmware running
Slow teal pulse	Bridge idle — waiting for MCP commands
Fast teal flash	Active MCP session — receiving tool calls
Red LED on	Error — check serial_log_read

>> Use a USB DATA cable. Charge-only cables have no data lines.

03 FLASH THE FIRMWARE

PREREQUISITES

- Python 3.10+ (python3 --version)
- ESP-IDF v5.x (docs.espressif.com/projects/esp-idf)
- idf.py in PATH (idf.py --version)
- Pclika board connected via USB-C data cable

CLONE & FLASH

```
git clone https://github.com/Pclika/mcp-platform
cd mcp-platform/firmware
idf.py set-target esp32s3
idf.py flash monitor
```

PRE-BUILT FIRMWARE (no ESP-IDF needed)

Download the pre-built .bin from GitHub Releases and flash with esptool:

```
pip install esptool
esptool.py --port /dev/ttyUSB0 --baud 460800 \
write_flash -z 0x0 pclika-firmware-v0.1.bin
```

>> Windows: if idf.py flash reports port busy, close Arduino IDE / PuTTY first.

04 INSTALL THE BRIDGE

```
pip install pclika-bridge

# Verify
pclika-bridge --version
pclika-setup --list
```

WHAT GETS INSTALLED

COMMAND / PACKAGE	PURPOSE
pclika-bridge	CLI — start MCP server (STDIO or SSE mode)
pclika-setup	CLI — auto-install configs for all AI clients
pclika_bridge	Python library — importable in custom scripts

CONNECTION MODES

MODE	DESCRIPTION	BEST FOR
STDIO (default)	Each AI client spawns its own bridge process	Simple — works out of the box
SSE / Adaptive	One bridge process shared by all clients	Multi-client concurrent use

05 VERIFY THE CONNECTION

LIST AVAILABLE PORTS

```
pclika-bridge --list-ports
# /dev/ttyUSB0 Pclika ESP32-S3 (USB VID:PID=303A:1001)
```

QUICK FUNCTIONAL TEST

Run bridge in debug mode and call device_info manually:

```
pclika-bridge --debug
> {"jsonrpc":"2.0","id":1,"method":"tools/call",
"params":{"name":"device_info","arguments":{}}}
< {"mcu":"ESP32-S3","firmware":"0.1.0","ip":"192.168.1.x","mac":"AA:BB:CC:..."}
```

>> DeviceNotConnected: check cable is a data cable, board LED is on, port matches --list-ports.

06 CONFIGURE YOUR AI CLIENT

Run **pclika-setup** to auto-configure all detected clients at once:

```
pclika-setup
# Detected: Claude Code, Cursor, VS Code
# Writing ~/.cursor/mcp.json ... OK
# Writing .vscode/mcp.json ... OK
# Running: claude mcp add pclika... ... OK

# SSE mode (multi-client):
pclika-bridge --serve &
pclika-setup --sse
```

CLIENT	CONFIG FILE / COMMAND	LOCATION
Claude Code	claude mcp add pclikaPlatform pclika-bridge	run once in terminal
Cursor	cursor.mcp.json	~/.cursor/mcp.json
VS Code	vscode.mcp.json	.vscode/mcp.json
Codex	codex.config.toml	~/.config/codex/config.toml
OpenCode	opencode.jsonc	~/.config/opencode/config.json

07 LOAD THE TOOL INSTRUCTIONS

Paste the following system prompt into your AI client at the start of a session (or as a persistent system prompt). This tells the AI what tools exist and how to use them safely.

SYSTEM PROMPT — copy and paste this

```
You are an AI assistant with direct hardware control via MCP tools.
Connected: Pclika ESP32-S3 board on local USB serial port.

ALWAYS call device_info() first to confirm connection.

Available MCP tools (server: pclikaPlatform):
sensor_read - read I2C/UART sensor (temp, humidity, light, distance, IMU)
display_text - write text to SSD1306 OLED or ST7789 TFT
servo_move - move servo channel (0-3) to angle (0-180 degrees)
gpio_write - set GPIO pin HIGH (1) or LOW (0)
gpio_read - read GPIO pin state
led_control - set R/G/B (0-255) on 4x WS2812B LEDs
wifi_scan - list nearby Wi-Fi networks (SSID, dBm, channel)
button_read - read BOOT/RESET/USER button state
device_info - MCU model, firmware version, IP, MAC
serial_log_read - UART diagnostic log (default 20 lines, max 200)

Safety: GPIO is 3.3V. Servos need 5V via PWM header.
On DeviceNotConnected error -- stop and report, do not retry in a loop.
```

>> Claude Code: add to CLAUDE.md or --system-prompt. Cursor/VS Code: Settings > AI > System Prompt. Codex/OpenCode: --system flag or system_prompt field in config.

08 DESCRIBE YOUR PROJECT

Once the system prompt is loaded, just describe what you want to build. The AI will plan the implementation, call hardware tools to verify readings, generate firmware code, and walk you through flashing.

EXAMPLE REQUIREMENTS

TITLE	REQUIREMENT	SKILLS COVERED
Hello World	Blink onboard LED green 3 times, then blue, then off.	Full stack test in 10 s
Env Monitor	Read BME280 every 5s. Show temp & humidity on OLED. Red LED above 28 C	Sensor + display + logic
Servo Sweep	Move servo 0 from 0 to 180 deg in 10-deg steps, pause 500ms. Log angle.	PWM + feedback
Wi-Fi Survey	Scan visible Wi-Fi. Sorted SSID/dBm/channel table. Highlight open networks.	RF data to AI analysis
Plant Monitor	Read ADC0 every minute. Below 30% warn + flash red x5. Log with timestamps.	ADC + LED + logging

09 THE FULL DEVELOPMENT LOOP

After submitting a requirement, the AI follows this workflow automatically:

#	PHASE	WHAT HAPPENS
1	Connect check	AI calls device_info() to confirm board is live before anything else.
2	Sensor verify	For sensor projects, AI reads once to confirm response and value range.
3	Plan	AI explains implementation in plain language and asks for confirmation.
4	Firmware code	AI writes commented ESP-IDF C code with exact GPIO pins for your board.
5	Flash instruction	AI tells you which idf.py command to run and what to watch in monitor.
6	Live test	AI calls MCP tools to verify running firmware: sensor, LED, servo.
7	Iterate	On errors, AI reads serial_log_read() and proposes fix. Usually one round.

COMPLETE SESSION EXAMPLE (hello-mcp)

```
You: Make the LED pulse cyan 5 times then show 'HELLO' on the OLED.

[AI] device_info() => ESP32-S3 online, firmware 0.1.0
[AI] led_control(r=0,g=200,b=160) x5 => cyan pulse
[AI] led_control(r=0,g=0,b=0) => off
[AI] display_text('HELLO', line=0) => OLED updated
AI: Done. LED pulsed cyan 5x. 'HELLO' is shown on OLED line 0.
```

10 SCALE UP

MULTI-CLIENT SSE MODE

When Cursor and VS Code are both open, run one shared bridge so both share the same hardware connection:

```
# Terminal 1 — keep running
pclika-bridge --serve
# => Listening on http://localhost:3141/sse

# Terminal 2 — write SSE config to all clients
pclika-setup --sse

curl http://localhost:3141/health
# {"status":"ok","port":"/dev/ttyUSB0","connected_clients":2}
```

ADD MORE MODULES

MODULE	FUNCTION	MCP CALL
BME280	Temp · humidity · pressure	sensor_read("bme280")
MPU6050	Accelerometer · gyroscope	sensor_read("imu")
VL53L0X	Time-of-flight distance	sensor_read("distance")
SHT31	High-accuracy temp/humidity	sensor_read("sht31")
PCA9685	16-channel PWM (more servos)	servo_move(ch, angle)
OV2640	Camera — JPEG capture	camera_capture()

FPGA EXPANSION (advanced)

Plug an iCE40UP5K FPGA module into the M.2 slot. Install the HDL bridge for 6 additional MCP tools:

```
pip install pclika-hdl-bridge
pclika-setup # adds pclikaHDL server automatically
```

APPENDIX A — MCP TOOL QUICK REFERENCE

TOOL	PARAMETERS	RETURNS / ACTION
`sensor_read`	sensor_id: "temp_humidity"|"bme280"|"light"|"distance"|"imu"|"sht31"	Returns sensor readings as JSON
`display_text`	text: str · line: int (0-7 OLED rows)	Write to SSD1306 / ST7789 display
`servo_move`	channel: int (0-3) · angle: float (0-180)	Move servo to angle in degrees
`gpio_write`	pin: int · value: int (0 or 1)	Set GPIO HIGH or LOW
`gpio_read`	pin: int	Read GPIO pin state (0 or 1)
`led_control`	r/g/b: int (0-255)	Set colour of 4x WS2812B LEDs
`wifi_scan`	(no params)	Returns nearby Wi-Fi list
`button_read`	"BOOT"|"RESET"|"USER"	Button state (0=released, 1=pressed)
`device_info`	(no params)	MCU, firmware version, IP, MAC
`serial_log_read`	lines: int (default 20, max 200)	Read UART log for diagnostics

APPENDIX B — TROUBLESHOOTING

DeviceNotConnected error	Hint: 90% of cases: charge-only USB cable

1. Board powered? Blue LED should be on.
2. Swap to a known data cable (charge-only has no data).
3. Run --list-ports — does the port appear?
4. Linux: sudo usermod -aG dialout $USER

Port appears but bridge hangs	Hint: idf.py monitor shows firmware boot log

1. Wrong firmware — reflash with idf.py flash.
2. Wrong baud rate — use --baud 115200.
3. Another process has the port — close Arduino IDE, PuTTY.

AI client: pclikaPlatform server not found	Hint: Claude Code: type /mcp to check server status

1. Run pclika-setup again.
2. Restart the AI client after config change.
3. Confirm pclika-bridge in PATH: which pclika-bridge

sensor_read returns null	Hint: I2C scanner sketch finds your device address

1. Check wiring: SDA=GPIO21, SCL=GPIO22.
2. Call serial_log_read(50) — AI diagnoses I2C error.
3. Check sensor I2C address (BME280=0x76 or 0x77).

servo_move — no movement	Hint: SG90 rated 4.8-6V — power directly from USB 5V

1. Servos need 5V, not 3.3V — power from PWM header.
2. Channels: 0=GPIO38, 1=39, 2=40, 3=41.
3. Angle must be float 0.0 to 180.0.

SSE mode — clients not connecting	Hint: pclika-setup --sse rewrites all client configs

1. Confirm pclika-bridge --serve is running.
2. Check http://localhost:3141/health in browser.
3. Firewall may block port 3141 — add exception.

SUPPORT & FEEDBACK

Encountered an issue? Have a feature request or improvement idea? Our support portal is always open.

https://pclika.com/Support

CHANNEL	CONTACT
Web Portal	https://pclika.com/Support
Email	starinvc@gmail.com
GitHub Issues	github.com/Pclika/mcp-platform/issues
User Store	https://mall.pclika.com/
MCP Self-Service	https://hc.pclika.com/

Your feedback directly shapes the next release. Every bug report, feature request, and review matters.

github.com/Pclika/mcp-platform

pypi.org/project/pclika-bridge

Shop: https://mall.pclika.com/ · MCP Self-Service: https://hc.pclika.com/

Support: https://pclika.com/Support

Pclika