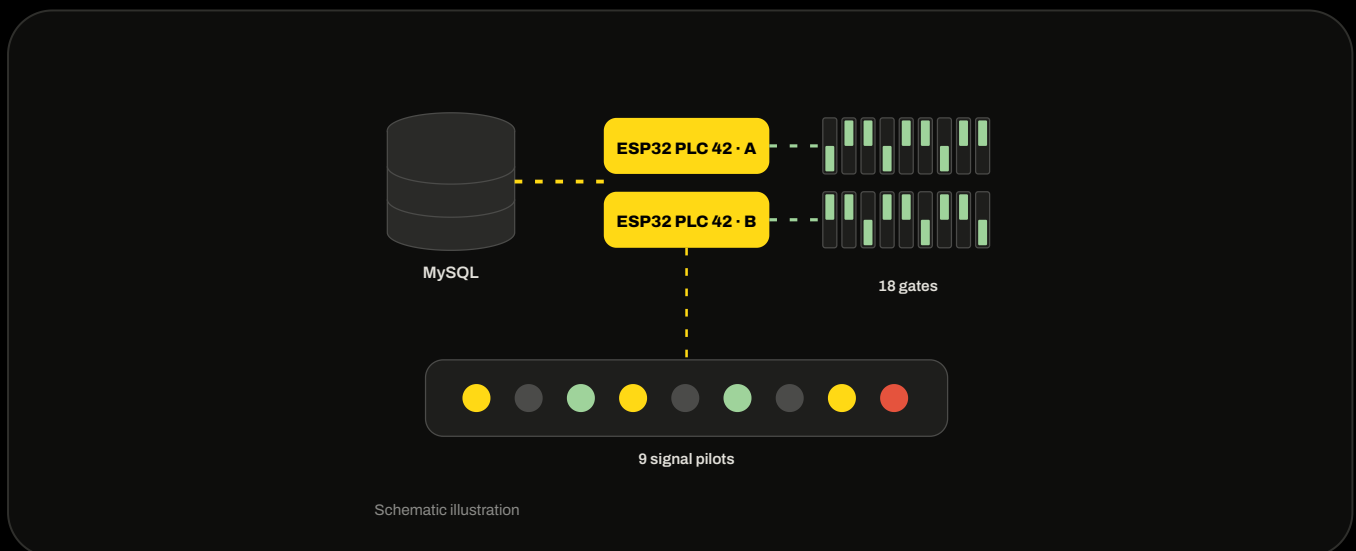


18 PNEUMATIC GATES · ESP32 PLC 42

# Your database knows which gate should open. **Two ESP32 PLC 42** move 18 pneumatic gates.

A real, anonymous deployment: 18 gates with pneumatic cylinders and 9 signal pilots, controlled by two ESP32 PLC 42 units. The open and close commands come over the network from a MySQL database. Here you see the pattern we found in it, and how you can test it on your own gates.

[Read the interactive case study online ›](#)

**18**

pneumatic gates, each moved by a cylinder

**9**

signal pilots on site

**2**

ESP32 PLC 42 units

**MySQL**

open and close commands sent over the network

## THE CHALLENGE

# Move 18 gates from commands that live in a database.

You do not write these orders on a local panel. You write them in a MySQL database, and your gates are on site, away from the cabinet. Your control system has to bring each command to the right gate, show you the state on site, and do what you decided in advance when something fails. You know this problem.

01

### Many actuators

18 gates. 9 pilots. You have pneumatic cylinders on every gate and signal pilots on site. Two controllers drive them, and your commands come from one source.

02

### Commands from IT

Your open and close commands are rows in a MySQL database. Your PLCs get them over the network. No buttons.

03

### A default state

Your network or your database fails. Then what? Each gate goes to the default state you defined in advance.



### Built by the customer's own team

They programmed and commissioned the system themselves. Their material: our documentation and our MQTT and Modbus courses. You have the same material.

## HOW IT WORKS

### Two loops: the command, and the failure.

#### Loop 1 · Command

#### A row becomes a gate movement

Your management system writes each command as a row in your table: the gate, open or close, and a status. The PLC that drives that gate gets the row over the network, switches the output for its valve and writes the result back. So your table shows what really happened. Not only what you asked for. You can trust it.

- Row written with status PENDING
- The PLC takes the command: SENT
- The PLC confirms the movement: DONE

#### Loop 2 · Network loss

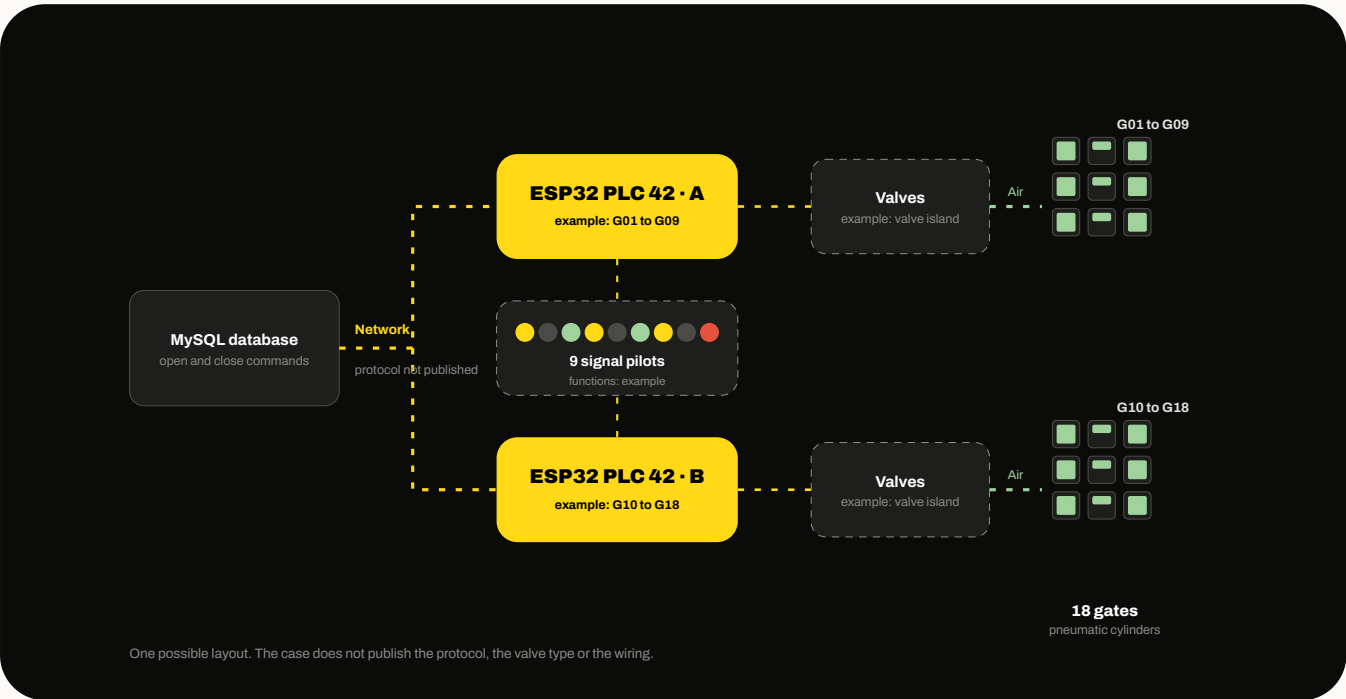
#### The link drops. Each gate goes to its default state.

Your PLC does not know by itself that the database is gone. So you add a watchdog to your program: it notices the silence. Then it moves each gate to the default state you chose, for example hold or close. This is a process default in your PLC program, not a safety function. If your people work near moving gates, you need a separate safety-rated circuit, such as an emergency stop, a pressure dump valve and guards.

- Watchdog timeout in your program (example: 5 s)
- Default state per gate: hold or close
- On reconnection, the PLC checks the table again before it accepts new commands

# How do two ESP32 PLC 42 units control 18 pneumatic gates?

Your database stores your commands. Two ESP32 PLC 42 units receive them over the network, drive the valves that move 18 pneumatic cylinders and switch 9 signal pilots. The case does not publish the protocol, the valve type or the wiring. The diagram shows one possible layout.



- Commands and status over the network
- Compressed air to the cylinders
- Example elements, not published by the case

## Signals and data (example layout)

| Flow              | Between                   | Type              | Purpose                                                                     |
|-------------------|---------------------------|-------------------|-----------------------------------------------------------------------------|
| Command row       | Management system → MySQL | Database row      | Stores which gate should open or close                                      |
| Command transfer  | MySQL → ESP32 PLC 42      | Network           | Brings each command to the PLC that drives the gate. Protocol not published |
| Status write back | ESP32 PLC 42 → MySQL      | Example           | Confirms SENT and DONE, or why a row did not run                            |
| Valve control     | ESP32 PLC 42 → valves     | Example           | Direct outputs, interface relays or a valve island bus. Not published       |
| Gate movement     | Valve → cylinder → gate   | Pneumatic         | Moves each of the 18 gates                                                  |
| Signal pilots     | ESP32 PLC 42 → pilots     | Example functions | Show the state on site with 9 pilots                                        |

# Two ESP32 PLC 42 units drive the gates and the pilots.



## ESP32 PLC 42

An industrial controller built on the dual core ESP32, with 26 inputs and 16 outputs, Ethernet, Wi-Fi and RS-485, for your DIN rail. In this case we count two units. They take the commands from the database and drive the gates and the signal pilots.

|              |                                                                         |
|--------------|-------------------------------------------------------------------------|
| Outputs      | 16: 10 digital optoisolated, 6 analog/digital/PWM (0-10 V), 5 to 24 Vdc |
| Inputs       | 26: 14 digital optoisolated, 12 analog/digital (0-10 V)                 |
| Network      | Ethernet 10/100 (RJ45), Wi-Fi 2.4 GHz, Bluetooth LE                     |
| Serial       | RS-485, RS-232, TTL                                                     |
| Power supply | 12 to 24 Vdc                                                            |
| Mounting     | DIN rail TS35, IP20                                                     |

The ESP32 PLC 42 has no relay outputs. If you wire your valves directly or through interface relays, each valve coil needs one output. If you use a valve island with a bus interface, your PLC writes the coil states over the network. These are example options. The case does not publish its wiring. Check the coil ratings in the family datasheet.

[See the ESP32 PLC 42 >](#)

### The database stays in charge

One place for orders. You write them there. Your PLCs execute them and write back what happened. After any failure, you still trust your table.

### A controller for your cabinet

DIN rail TS35, 12 to 24 Vdc supply, optoisolated I/O, Ethernet and Wi-Fi in one unit. You program it in C++ with the Arduino IDE, and you mount it next to your valves.

### Your team can build it

Proof? Here the customer's own team did it with our documentation and our MQTT and Modbus courses. You can use the same material for your project.

### SIMILAR PROJECT?

**Tell us your gates and your signals. An engineer will answer.**

Tell us how many gates and signals you have and where your commands come from: a database, an MES, a SCADA or a cloud service. We will tell you which controller fits your project and how you can connect it.

### READ IT ONLINE

[industrialshields.com/pneumatic-gate-control-case-study](https://industrialshields.com/pneumatic-gate-control-case-study)

### EMAIL

[info@industrialshields.com](mailto:info@industrialshields.com)

### PHONE

[\(+34\) 938 760 191](tel:+34938760191)

