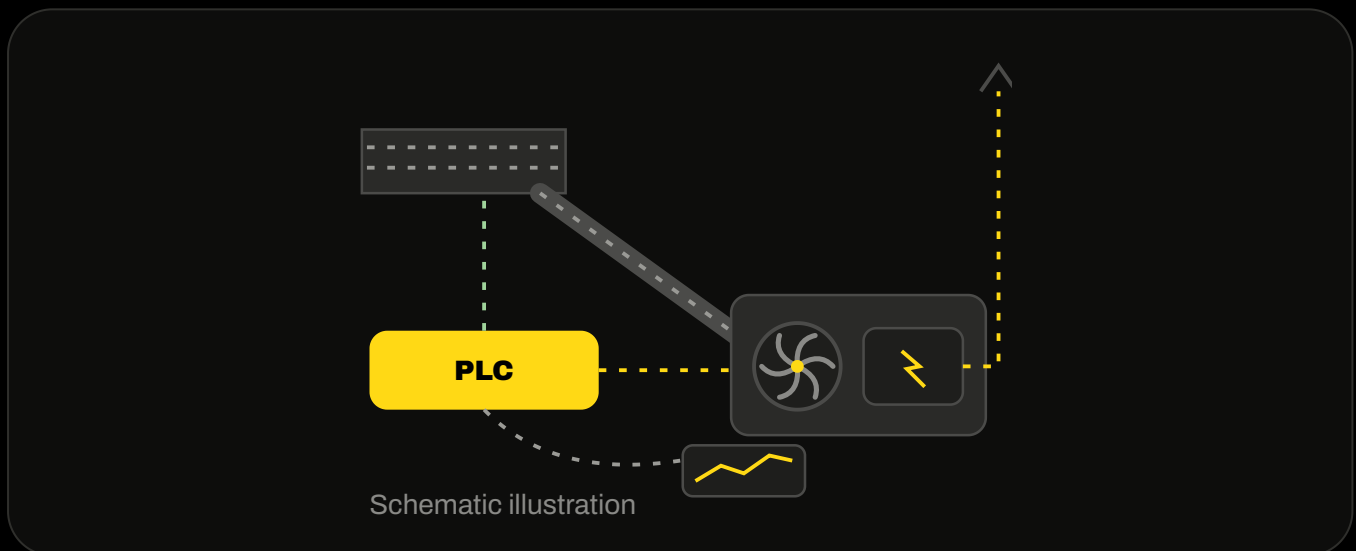


SMALL HYDRO PLANT · RASPBERRY PLC

Start, follow the river, stop. From far away.

Two Raspberry PLCs on a CAN bus run a small hydro plant.

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


21

I/O on the turbine cabinet
PLC: 13 inputs, 8 outputs

42

I/O on the plant cabinet
PLC: 26 inputs, 16 outputs

2

Raspberry PLCs joined by
CAN, one CAN board in
each

3

stages in the logic: start,
level control, safe stop

THE CHALLENGE

A small hydro plant is far away. Its control must not need a visit.

Many small hydro plants are in remote valleys, with no operator on site. The control has to start the turbine in the right order, follow the water that the river brings, and stop the unit safely when something goes wrong. The office must see what happens and act from far away. And each new plant should get the same tested cabinet.

01

Conditions before water

The turbine starts only when the level, the inlet valve and the protections allow it.

02

Speed before the grid

The unit speeds up with no load. A synchronizer closes the breaker at the right moment.

03

The river sets the power

In a run-of-river plant, the guide vanes follow the headwater level, not a power set point.

04

A stop that protects

On a trip, the vanes close along a ramp and then the inlet valve closes. A fast closing causes water hammer.



One cabinet design, many plants

The project builds its control cabinets in series: the same controller and the same wiring. A typical way to run them: the same program at each site, the office sees all the plants in one place, and each plant keeps its logic and stores its data when the link drops.

HOW IT WORKS

Three stages: start, follow the river, stop safely.

Stage 1 · Start sequence

Permissives, inlet valve, speed, synchronizer. In that order.

Before a start, the PLC checks the permissives: the headwater level is above the start level, the protection relay has no trip, the emergency stop chain is closed and the inlet valve is closed. Then it opens the inlet valve (where there is a bypass, it equalizes the pressure first), sets the guide vanes to a small start opening and lets the unit speed up with no load. Near the rated speed the voltage regulator (AVR) excites the generator and the PLC enables the synchronizer. The synchronizer sends raise and lower pulses to the PLC speed loop and to the AVR, and closes the generator breaker when voltage, frequency and phase match. Then the PLC opens the vanes to take load.

- Permissives on digital and analog inputs
- Shaft speed from a pulse sensor on an interrupt input

Stage 2 · Follow the river

The headwater level sets the opening. The river sets the power.

A run-of-river plant has little storage. The PLC reads the headwater level on an analog input and moves the guide vanes to hold the level at its set point: more water arrives, the vanes open; less water, they close. A turbine cannot run below a minimum opening. If the level falls to the stop level, the PLC unloads and stops the unit, so the intake does not run dry and the residual flow stays in the river. When the level comes back, the unit starts again.

- Headwater level transmitter on an analog input
- Guide vane set point on an analog output, position back from a CANopen encoder

Stage 3 · Trip and report

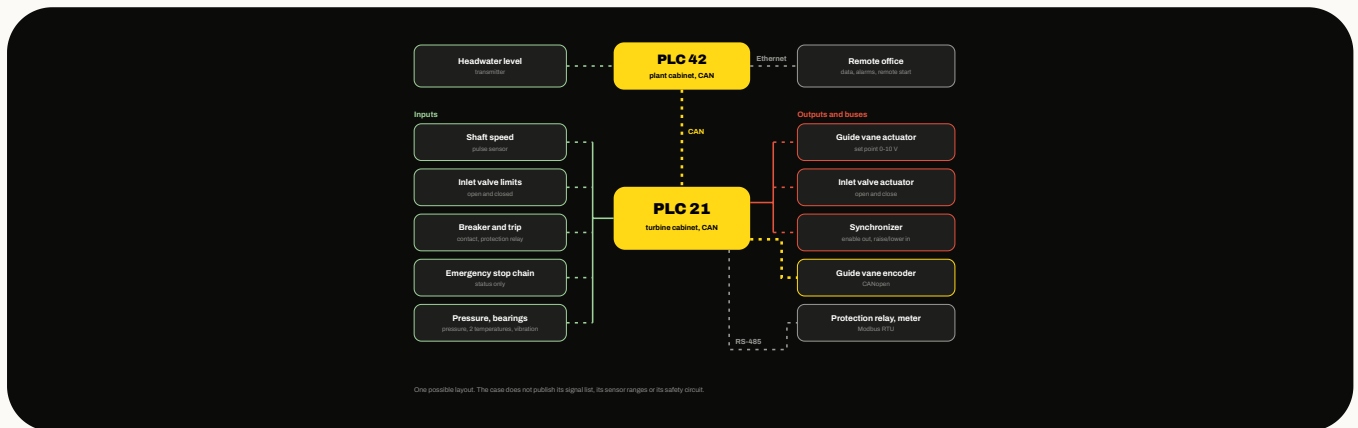
A trip closes the water along a ramp. The office gets the alarm.

The PLC watches the trips it is in charge of, such as overspeed, bearing temperature and a breaker opened by the grid protection. On a trip it closes the guide vanes along a closing ramp and then closes the inlet valve. A closing that is too fast raises the pressure in the penstock (water hammer). A closing that is too slow lets the speed rise after a load rejection. The turbine supplier sets the closing time. The alarm goes to the remote office.

- Bearing temperature and vibration on analog inputs
- Closing ramp first, inlet valve closed after

How does a PLC automate a small hydro plant?

Two Raspberry PLCs share the work. In the turbine cabinet, a Raspberry PLC 21 with a CAN board reads the unit: shaft speed, inlet valve limit switches, breaker status, protection trip, emergency stop chain, penstock pressure, bearing temperatures and vibration. It drives the inlet valve, the guide vane set point and the synchronizer enable, and it reads the guide vane position from a CANopen encoder. Over RS-485 it reads the electrical values from the protection relay or the energy meter with Modbus RTU. In the plant cabinet, a Raspberry PLC 42 reads the plant signals, such as the headwater level, and links the plant to the remote office over Ethernet. In this layout, one CAN bus joins the two PLCs and the encoder.



Inputs Outputs CAN bus Ethernet and RS-485

Signals and data (one possible layout)

Signal	Between	Type	Purpose
Shaft speed	Pulse sensor → PLC 21	Digital in, interrupt	Speed up, synchronizer enable, overspeed trip
Inlet valve open and closed	Limit switches → PLC 21	2 digital in	Confirms each step of the start and the stop
Breaker closed	Breaker contact → PLC 21	Digital in	Unit on the grid: take load. Opens on a grid trip
Protection trip	Protection relay → PLC 21	Digital in	Generator or grid protection tripped: stop the unit
Emergency stop chain	Safety relay → PLC 21	Digital in	Status only. The stop itself does not depend on the PLC program
Penstock pressure	Transmitter → PLC 21	Analog in	Start permissive, pressure during the closing ramp
Bearing temperatures	Transmitters → PLC 21	2 analog in	Alarm, then trip. Turbine and generator bearings
Vibration	Velocity transmitter → PLC 21	Analog in	Alarm, then trip
Guide vane position	CANopen encoder → PLC 21	CAN	Closed loop of the opening
Guide vane set point	PLC 21 → actuator	Analog out 0-10 V	Opening that follows the level
Inlet valve open and close	PLC 21 → valve actuator	2 digital out	Water to the turbine. Closed at the end of a stop
Synchronizer enable	PLC 21 → synchronizer	Digital out	Near rated speed. The synchronizer closes the breaker
Speed raise and lower	Synchronizer → PLC 21	2 digital in	Small changes of the guide vane opening while synchronizing. One on an analog/digital input in digital mode
Electrical values	Protection relay or meter → PLC 21	RS-485, Modbus RTU	Active power, voltage, frequency, energy
Headwater level	Level transmitter → PLC 42	Analog in	Start permissive, level control, stop level
Unit data	PLC 21 ↔ PLC 42	CAN	States, values and commands between the cabinets
Remote office	PLC 42 → router → office	Ethernet	Data, alarms, remote start and stop

A Raspberry PLC for the turbine. A bigger one for the plant.



Raspberry PLC 21 FLEX

An industrial controller built on a Raspberry Pi 4, with 13 inputs, 8 outputs, two Ethernet ports, two RS-485 ports and one expansion board. In this plant the expansion board is CAN. It runs the turbine: start sequence, level control, trips and the guide vane loop.

Role	Turbine cabinet: sequence, level control, trips
Inputs	13: 6 analog/digital (0-10 V, 12-bit), 7 digital optoisolated (5-24 Vdc), 2 with interrupt
Outputs	8: 3 analog/digital/PWM (0-10 V, 12-bit), 5 digital optoisolated (5-24 Vdc)
Expansion	One board. CAN in this plant (also LoRa, 4G or M-Bus master)
Comms	Ethernet 1 Gigabit + 1 10/100, 2 x RS-485, Wi-Fi, Bluetooth LE
Computer	Raspberry Pi 4, Debian 64-bit

The project uses a Raspberry PLC 21 with the CAN board: the FLEX version is the configuration that offers it. The outputs are transistor outputs, not relays: use interface relays where the valve actuator or the synchronizer need a dry contact. This layout uses 12 of the 13 inputs and 4 of the 8 outputs. For a 4-20 mA transmitter on a 0-10 V input, add a 500 Ω precision shunt.

[See the Raspberry PLC 21 FLEX ›](#)



Raspberry PLC 42

The same family with more I/O: 26 inputs and 16 outputs. In the plant cabinet it reads the plant signals, shares the unit data over CAN and links the plant to the remote office over Ethernet.

Role	Plant cabinet: plant signals, link to the office
Inputs	26: 12 analog/digital (0-10 V, 12-bit), 14 digital optoisolated (5-24 Vdc), 4 with interrupt
Outputs	16: 6 analog/digital/PWM (0-10 V, 12-bit), 10 digital optoisolated (5-24 Vdc)
Expansion	One board. CAN in this plant (also LoRa or 4G)
Comms	Ethernet 1 Gigabit + 1 10/100, 2 x RS-485, Wi-Fi, Bluetooth LE

The project has a second cabinet with a Raspberry PLC 42 and CAN. The case does not publish which plant signals go to it: the headwater level is an example. Spare inputs can take plant auxiliaries, such as intake gates or a trash rake.

[See the Raspberry PLC 42 ›](#)

Permissives, then actions

Each step starts only when the step before is confirmed by a signal: level, valve limit switch, speed, breaker contact. The PLC program can be written as a sequence of states with clear exits and a time limit on each step.

A ramp, not a slam

The guide vanes close along a ramp set by the turbine supplier. The ramp keeps the penstock pressure and the speed rise inside their limits.

Linux next to the control

The Raspberry PLC runs Debian. The same controller can log data, keep a local database and talk MQTT to the office. A typical way to do it: run the control program as a [systemd service](#) that starts at boot and restarts on a failure.

SIMILAR PLANT?

Tell us your plant and your sequence. An engineer will answer.

Tell us the turbine type, the signals you have, how the guide vanes or the nozzle move, and where the data must go. We will tell you which controllers fit and how to wire them.

FULL CASE STUDY ONLINE

industrialshields.com/case-study-small-hydro-plant-raspberry-plc

EMAIL

info@industrialshields.com

PHONE

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