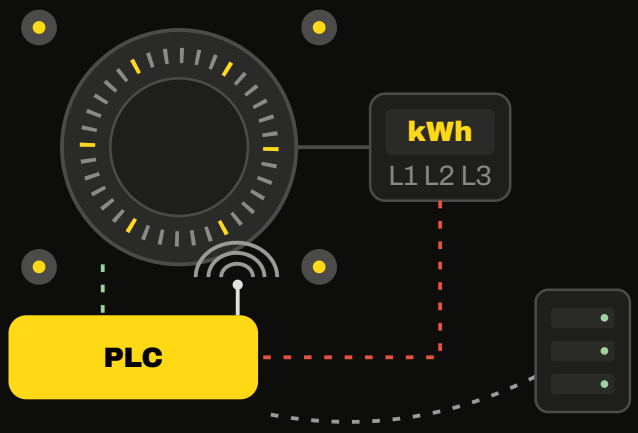
KNITTING PLANT · ESP32 PLC

Stops, alarms and kWh. One PLC per machine.

One ESP32 PLC per knitting machine: state, alarms and energy over MQTT.

Read the interactive case study online ›



Schematic illustration

6 machine contacts through coupling relays	**2** interrupt inputs: speed and shaft turns	**1** three-phase power meter per machine	**3** layers: machine, area, server

THE CHALLENGE

Many machines, different makers, one factory WiFi.

Maintenance wants to know when a machine stops and why. Management wants to know what each machine produces and what it uses. The machines come from different makers and give different signals. The factory WiFi does not always hold, and a lost event is a lost stop.

01

Different machines, different signals

One machine type gives six contacts: running, emergency stop, needle, oil, roll finished and broken yarn. Others give only the speed and the turns of the main shaft.

02

Machine circuits stay as they are

Coupling relays copy each machine signal to a clean 24 Vdc contact. The PLC does not share a circuit with the machine control.

03

Energy per machine, not per plant

A three-phase power meter in each machine. The plant meter gives the total. It cannot tell which machine used it.

04

A network that can drop

Events must not get lost when the WiFi fails. Each PLC keeps its data until the next layer has it.

One view for maintenance and management

A concentrator in each production area collects the data of its machines, keeps it until the server confirms it and passes it on to the business software. A heartbeat shows which PLC is silent.

HOW IT WORKS

Three stages: state, energy, delivery.

Stage 1 · State and alarms

A contact changes. The PLC publishes it at once.

Coupling relays copy the machine signals to the digital inputs of the PLC. Each change of an input is an event: the PLC publishes it over MQTT at once, with a time stamp from its real-time clock and a unique message ID. The speed and the turns of the main shaft come as pulses on the two interrupt inputs. Pick an event and watch the state, the stop count and the log.

Stage 2 · Energy per machine

A meter on each machine. The PLC reads it and adds the state.

A three-phase power meter measures each machine. A typical way to do it: the PLC, as Modbus RTU master, polls the meter (a slave) on its RS-485 port every few seconds and gets voltage, current, power and the energy counter. The PLC adds the machine state to each reading. Then the plant sees the energy a machine uses while it runs, and while it stands still. Move the load and stop the machine.

Stage 3 · Delivery

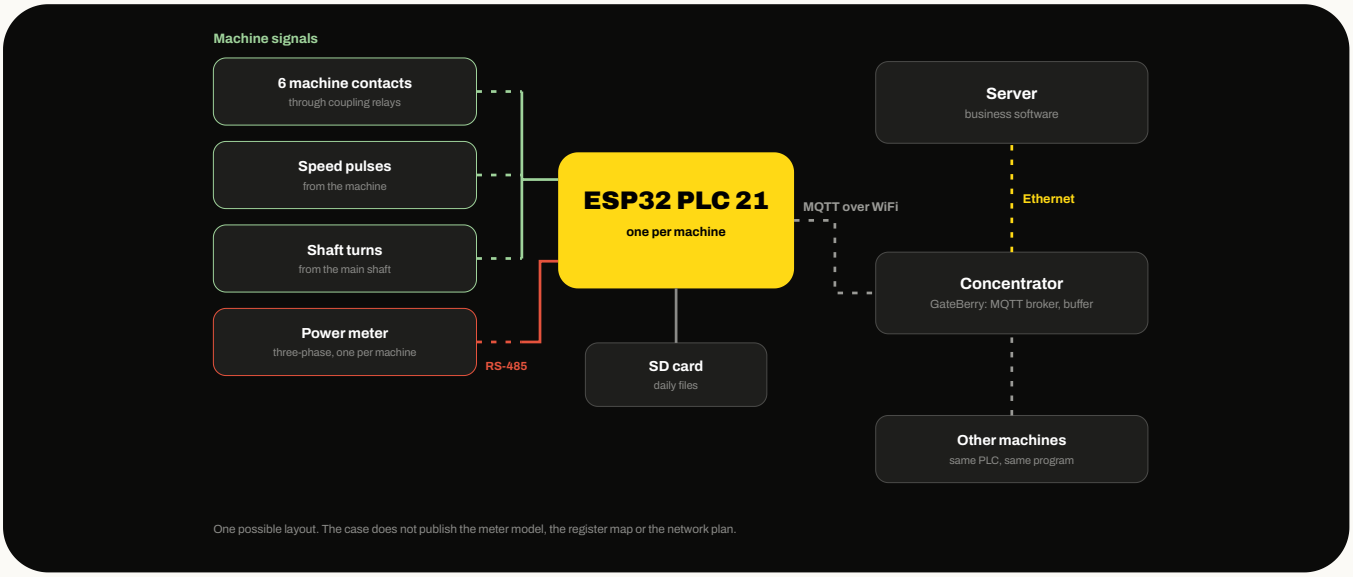
The WiFi drops. The data waits on the card.

A typical way to do it: the PLC writes every event and every reading to an SD card, one file per day, as in the offline-first logging example. When the WiFi or the concentrator is not there, the PLC keeps working and keeps writing. When the link comes back, it sends what is pending, and the message ID lets the concentrator find duplicates. The concentrator keeps the data until the server confirms it. Cut the WiFi, then restore it.

SYSTEM ARCHITECTURE

How do you monitor knitting machines and their energy with a PLC?

In each machine there is an ESP32 PLC 21 CORE, a three-phase power meter, coupling relays and a 24 Vdc supply. The PLC reads the machine contacts on digital inputs, the speed and the turns on two interrupt inputs, and the meter on RS-485. It keeps a local record and publishes it over MQTT on the factory WiFi. In each production area, a GateBerry concentrator runs the local MQTT broker, keeps and resends the data, gives the time and the firmware updates to the PLCs, and passes the data to the server and the business software.



Machine signals · RS-485 (power meter) · MQTT over WiFi · Ethernet

Signals and data

Signal	Between	Type	Purpose
Machine running	Machine contact → relay → PLC	Digital in	Running or stopped. Every change is an MQTT event
Emergency stop alert	Machine contact → relay → PLC	Digital in	Stop reason
Needle alert	Machine contact → relay → PLC	Digital in	Stop reason: needle
Oil alert	Machine contact → relay → PLC	Digital in	Stop reason: lubrication
Roll finished	Machine contact → relay → PLC	Digital in	A piece of fabric is complete
Broken yarn	Machine contact → relay → PLC	Digital in	Stop reason: yarn
Speed	Machine pulse → PLC	Interrupt in	Speed in rpm, on machines without the six contacts
Main shaft turns	Machine pulse → PLC	Interrupt in	Turn counter for the production
Power and energy	Power meter ↔ PLC	RS-485	Voltage, current, power, kWh. Modbus RTU is a typical choice
Events and readings	PLC → concentrator	MQTT over WiFi	An event on each change, a full snapshot at fixed times
Local record	PLC → SD card	Daily file	Everything is written first. Resent after a network cut
Heartbeat	PLC → concentrator	MQTT over WiFi	Shows which PLC is silent
Time and firmware	Concentrator → PLC	WiFi	Time for the real-time clock, firmware updates
Plant data	Concentrator → server	Ethernet	Kept until the server confirms it, then to the business software

HARDWARE USED

A PLC in each machine. A gateway in each area.



ESP32 PLC 21 CORE

An industrial controller built on the dual-core ESP32, with WiFi, Bluetooth LE, Ethernet, RS-485 and RS-232, 13 inputs and 8 outputs, for DIN rail. It is programmed with the Arduino IDE. In each machine it reads the contacts, the pulses and the power meter, keeps a local record and publishes over MQTT.

Inputs	**13: 6 analog/digital (0-10 Vdc analog, 0-24 Vdc digital), 7 digital isolated (5-24 Vdc, 2 with interrupts, up to 1 kHz)**
Outputs	**8: 3 analog/digital/PWM (0-10 V), 5 digital isolated**
Wireless	**WiFi 2.4 GHz, Bluetooth LE**
Wired comms	**Ethernet (RJ45), RS-485, RS-232, I2C, SPI**
Clock and memory	**RTC DS3231 with battery, 16 MB flash**

The CORE is the ready-to-ship version without an expansion board. For 4G, LoRa or CAN on board, see the ESP32 PLC 21 FLEX. Input count: the six contacts use 6 inputs and the pulses use the 2 interrupt inputs, so even a machine with both uses 8 of 13.

See the ESP32 PLC 21 CORE ›



GateBerry FLEX

The concentrator of each production area. An industrial gateway with a Raspberry Pi, for DIN rail, running Linux. It hosts the local MQTT broker and Node-RED, keeps the data of its PLCs until the server confirms it, gives them the time and the firmware updates, and watches their heartbeat.

Network	**Gigabit Ethernet, WiFi, Bluetooth LE**
Field bus	**RS-485 with switchable termination, I2C**

The project used one concentrator per production area. The GateBerry FLEX can be configured with more RAM, and with LoRa, 4G, CAN, M-Bus or RS-232. See how to build it in the Node-RED MQTT concentrator example.

See the GateBerry FLEX ›

An event, not a poll

The PLC publishes a change when it happens. A short stop does not hide between two polls. A full snapshot at fixed times (every 20 s, for example) confirms the state.

Relays between two systems

The datasheet asks for a common ground or for isolation when you connect other systems to the PLC. Coupling relays give the PLC a dry contact at 24 Vdc, and the machine circuits stay as the maker wired them.

One time base

The concentrator gives the time to all the PLCs, and the battery-backed real-time clock keeps it during a cut. Events from many machines line up in the order they happened on the floor.

SIMILAR PLANT?

Tell us your machines and your signals. An engineer will answer.

Tell us which machines you have, which signals each one gives, if you want energy per machine and where the data must go. We will tell you which controller fits and how to wire it.

READ IT ONLINE
industrialshields.com/case-study-knitting-machine-energy-monitoring-esp32-plc

EMAIL
info@industrialshields.com

PHONE
(+34) 938 760 191