THE DIGITAL TWIN · VOLUME 1
BOOK 2 · PRACTICE

The Digital Twin Hands-On Projects

Design, Simulation, Implementation, and Validation of Digital Twin Systems Using MATLAB, Simulink, Python, Arduino, and PLC

Dr. Ahsan Rahman
University of Prince Mugrin · Al Madinah

Table of Contents

Part I — Foundations

Part I lays the groundwork. It explains what a digital twin is, what it is made of, what hardware and software you will use, and the workflow that every later build follows.

PART I — FOUNDATIONS

Chapter 1 — Introduction to Digital Twin Technology

Learning Objectives

By the end of this chapter you will be able to:

- Explain, in plain language, what a digital twin is and how it differs from an ordinary model.
- Trace the short history of the idea, from NASA's spare spacecraft to today's connected machines.
- Distinguish a digital model, a digital shadow, and a true digital twin by how their data flows.
- Identify the seven building blocks that every digital twin in this book is made from.
- Describe what you will build across the book and why one running example is used throughout.

1.1 What Is a Digital Twin?

Imagine you could keep a perfect, living copy of a machine — not a drawing of it, and not a one-off simulation, but a copy that breathes in step with the real thing. When the real motor speeds up, the copy speeds up. When a bearing starts to wear, the copy feels it too. And when the copy works out a better way to run the machine, it can quietly hand that advice back. That living, two-way copy is a digital twin.

More formally, a **digital twin** is a virtual representation of a physical asset that is continuously linked to that asset by a flow of data, and is faithful enough to be used in its place for monitoring, analysis, and decision-making. Two words in that sentence carry most of the weight. *Continuously* means the link is live, not a snapshot. *Faithful* means the model is trustworthy enough that we are willing to act on what it tells us.

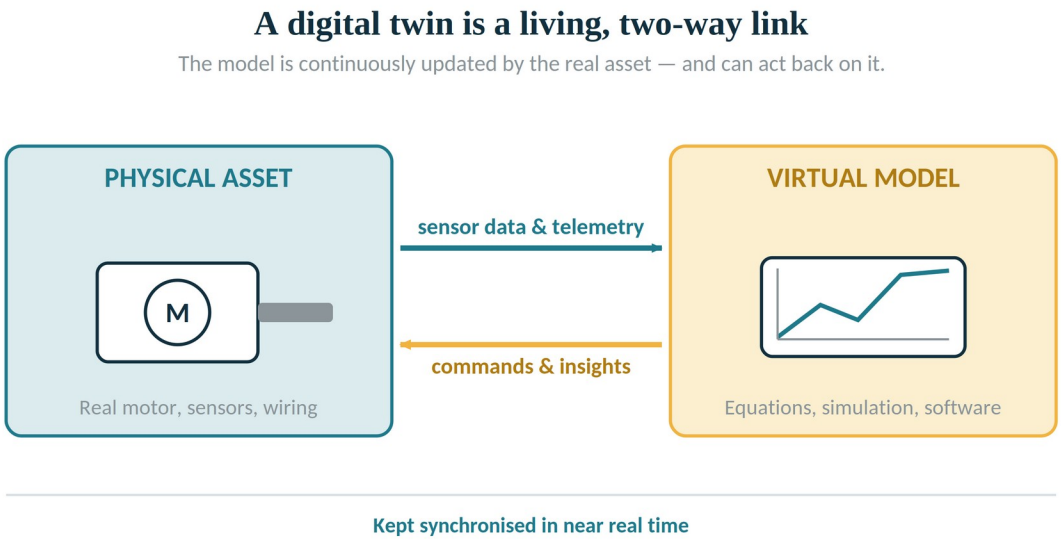

Figure 1.1. **A digital twin is a living, two-way link.** *Sensor data flows up to keep the model honest; commands and insights flow back down to the asset.*

The picture captures the whole idea on one page. On the left is the physical asset — in our case a real DC motor with its sensors and wiring. On the right is the virtual model — a set of equations, a simulation, or a piece of software that behaves like the motor. The two are joined by a loop. Data climbs from the asset to the model so the model always knows the true state of the machine; insights and commands flow back down so the model can influence what the machine does next. Take away that loop and you no longer have a twin; you simply have a model that happens to resemble the asset.

Reality Check

> *In industry the word "twin" is used loosely, and a great deal of marketing calls any 3-D model a digital twin. Throughout this book we hold to a stricter test: if there is no live data connection to a real asset, it is not yet a twin. Keeping that test in mind will save you a great deal of confusion when you read vendor brochures later.*

1.2 A Short History of the Idea

The instinct behind digital twins is older than the computers that now make them possible. During the Apollo programme in the 1960s, NASA kept fully built duplicates of its spacecraft on the ground. When something went wrong in orbit, engineers could reproduce the fault on the duplicate, try out fixes, and only then radio up instructions. Those were physical twins, but the principle is exactly ours: keep a faithful copy you can experiment on safely.

The modern, software-based notion took shape in 2002, when Dr. Michael Grieves described a “mirrored” pairing of a physical product and its virtual counterpart in his work on product lifecycle management. For most of the following decade the idea stayed within aerospace and high-end manufacturing, because connecting a model to a real machine was expensive. What changed everything was the arrival of cheap sensors, cheap microcontrollers such as the Arduino, and easy networking. The live data link — the hard part — suddenly became affordable for students and small workshops, not just for space agencies.

Today, under the banners of Industry 4.0 and the Internet of Things, digital twins have spread from jet engines and wind farms to buildings, factories, and individual machines on a lab bench. The technology you will use in this book to twin a thirty-dollar motor is, in spirit, the same technology used to twin a multimillion-dollar turbine.

1.3 Model, Shadow, or Twin?

Not every virtual copy deserves to be called a twin. The clearest way to tell them apart is to ask a single question: how does data move between the physical thing and its virtual copy? The answer sorts every system into one of three levels.

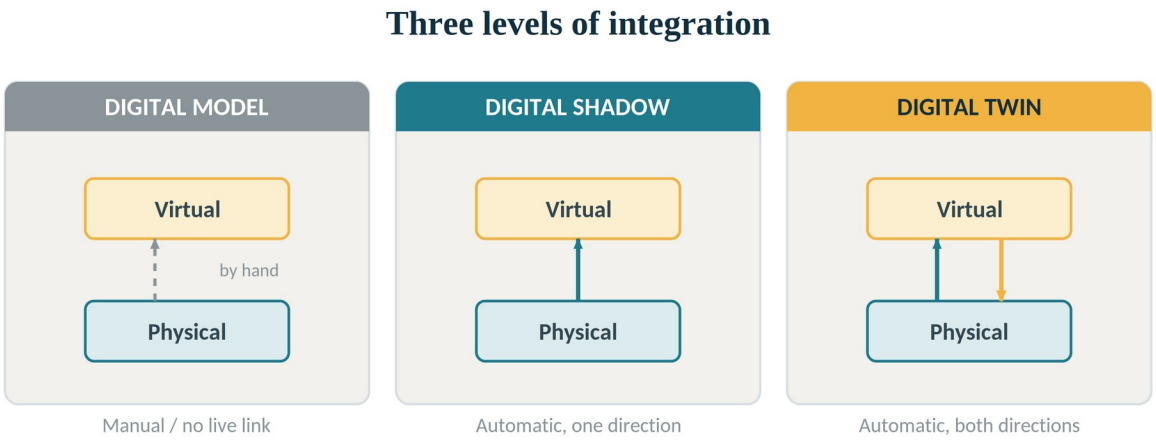

Figure 1.2. Three levels of integration. *The difference between a model, a shadow, and a twin is simply how data flows between physical and virtual.*

Digital model. A virtual copy with no automatic link to the real asset. You might run a Simulink model of a motor on your laptop, but if you have to type in the speed by hand, it is only a model. It is useful for design and study, yet it knows nothing about the actual machine today.

Digital shadow. A virtual copy fed automatically by the asset, but in one direction only. Sensor readings stream into the model so it reflects the live state of the

machine, but the model cannot act back. A live dashboard mirroring your motor's speed is a shadow: it watches faithfully, but it cannot change anything.

Digital twin. A virtual copy connected in both directions. Data flows up to keep the model honest, and decisions flow back down to tune, correct, or protect the machine. This closed loop is what makes a twin genuinely useful — and it is the level we reach in the hybrid builds later in the book.

Try It

> *Think of one device near you right now — a thermostat, a phone's battery manager, a car's cruise control. For each, decide whether it is a model, a shadow, or a twin, and write one sentence justifying your answer by naming the direction its data flows. You will be surprised how often everyday "smart" devices are only shadows.*

1.4 The Anatomy of a Digital Twin

Every digital twin you build in this book — whatever the toolchain — is assembled from the same seven building blocks. Learning them now means each later chapter is simply a matter of choosing which tools fill each block. The blocks stack from the real world at the bottom to intelligent action at the top, and a feedback path closes the loop from top back to bottom.

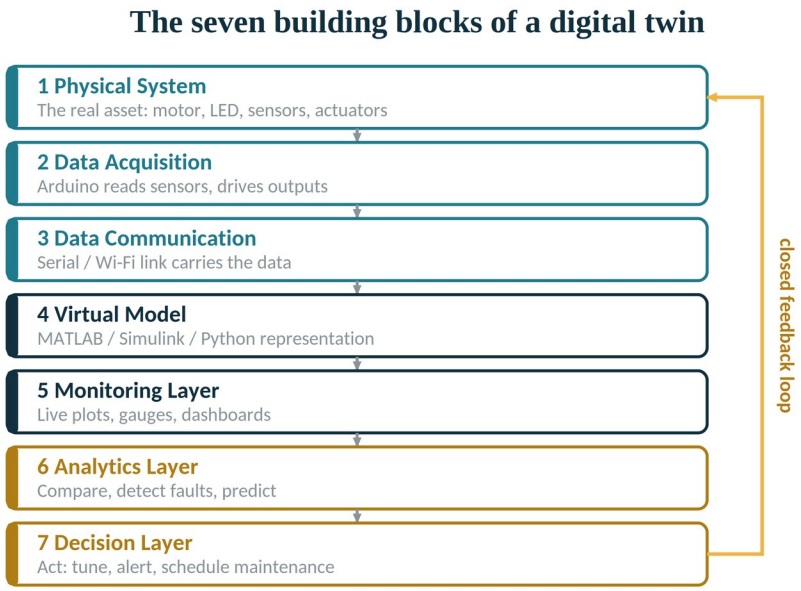

Figure 1.3. The seven building blocks. *From the physical system up to the decision layer, with a feedback path that closes the loop.*

1. Physical system. The real asset and everything attached to it: the motor or LED, its sensors, and its actuators. Nothing in the twin matters unless it ultimately serves this layer.

2. Data acquisition. The point where the physical world becomes numbers. An Arduino reads voltages and pulses from the sensors and drives the outputs that move the machine.

3. Data communication. The channel that carries those numbers onward — a USB serial cable in the simplest builds, or Wi-Fi when we go wireless.

4. Virtual model. The mathematical or software twin of the asset, written in MATLAB, Simulink, or Python. This is the block that can predict, not merely report.

5. Monitoring layer. The human-facing view: live plots, gauges, and dashboards that turn the data stream into something a person can read at a glance.

6. Analytics layer. The reasoning block. It compares the model against reality, measures the error, detects faults, and looks ahead to predict what comes next.

7. Decision layer. The block that acts: it retunes the controller, raises an alert, or schedules maintenance — and that action feeds straight back to the physical system, closing the loop.

1.5 Why Digital Twins Matter

It is fair to ask what all this machinery buys us. Four benefits recur across every application, large or small, and they are the reasons the technology has spread so quickly.

Benefit	What it means in practice
See inside	Read quantities you cannot easily measure directly — a hidden temperature, an internal stress, a true motor speed — by inferring them from the model.
Look ahead	Run the model faster than real time to predict what the asset will do next, before it actually does it.
Ask “what if?”	Test a risky change — a new setting, a heavier load — on the twin first, where a mistake costs nothing.
Catch faults early	Notice when reality and the model drift apart, often the first quiet sign that something is wearing out.

None of these benefits require an expensive machine. The very same advantages apply to the small motor on your bench, which is precisely why it makes such a good teacher.

1.6 Where Digital Twins Are Used

Digital twins now appear wherever a physical asset is valuable enough to be worth watching closely. In manufacturing, they mirror production lines to spot bottlenecks and predict tool wear. In energy, they track wind turbines and power transformers to schedule maintenance before failure. In aerospace and automotive engineering, they shadow engines and battery packs through their whole lives. Hospitals twin critical equipment; cities twin water networks and traffic; and researchers increasingly build twins of organs to test treatments in simulation. The scale ranges from a single component to an entire factory, but in every case the seven blocks of Section 1.4 are present in some form.

1.7 What This Book Builds

Rather than skim many systems, we go deep on two. The **DC motor speed-control bench** is our running example: a small brushed motor whose speed we will measure, model, control, and ultimately twin. It is the spine of the book, returned to again and again, each time with a sharper tool. Alongside it, the **LED monitoring station** provides a lighter second system for reinforcing each idea from a new angle, with a stronger emphasis on sensing, dashboards, and data logging.

Each system is built four ways — with MATLAB and Simulink, with Python, as a hybrid of both, and in Part V, with a Programmable Logic Controller using IEC 61131-3 Structured Text, Modbus TCP, and OPC-UA. By the final chapter you will have created eight complete digital twins and will understand, from direct experience, the trade-offs between toolchains. The goal is not to make you an expert in one piece of software, but fluent in the underlying pattern, so that you can twin a system you have never seen before using whatever tools you have to hand — whether that is Python on a laptop or a PLC on a factory floor.

Box 1.1 — One system, sharpened again and again

> *If you remember one design decision behind this book, remember this: we do not abandon a system once it is built. The same DC motor that begins as a sketch in Chapter 1 ends, many chapters later, as a validated, self-correcting digital twin. Every new technique is added as a layer on a machine you already understand.*

Chapter Summary

- A digital twin is a virtual copy of a physical asset, kept faithful by a continuous live link and trustworthy enough to act on.

- The idea began with NASA's physical spacecraft duplicates and became practical in software once cheap sensors, microcontrollers, and networking arrived.
- How data flows separates a digital model (no live link) from a digital shadow (one-way) and a true digital twin (two-way, closed loop).
- Every twin in this book is assembled from seven blocks: physical system, data acquisition, communication, virtual model, monitoring, analytics, and decision.
- We will build two systems — a DC motor and an LED station — four approaches each: MATLAB, Python, Hybrid, and PLC (Part V), using the DC motor as the running example throughout.

Key Terms

Term	Meaning
Digital twin	A virtual copy of an asset, continuously linked to it by data and used in its place for monitoring and decisions.
Digital model	A virtual copy with no automatic data link to the real asset; updated by hand.
Digital shadow	A virtual copy fed automatically by the asset in one direction only; it watches but cannot act.
Closed loop	An arrangement in which the output of a system is fed back to influence its own input.
Data acquisition	The process of converting physical signals into numbers a computer can use.
Validation	Checking how closely the virtual model's behaviour matches the real asset's measured behaviour.

Review Questions — Chapter 1

Recall

1.1 Give, in your own words, the strict test this book uses to decide whether something is truly a digital twin.

1.2 List the seven building blocks of a digital twin in order, and state in one phrase what each contributes.

1.3 Define digital model, digital shadow, and digital twin by the direction in which their data flows.

Application

1.4 A colleague shows you a live dashboard that displays a pump's pressure but cannot change the pump. Model, shadow, or twin? Justify your answer.

1.5 Pick one of the four benefits in Section 1.5 and describe a situation from your own field where it would be valuable.

Analysis

1.6 Explain why NASA's Apollo-era ground duplicates count as early twin thinking even though they used no software model.

1.7 Many "smart" consumer devices are shadows rather than twins. Choose one, explain which it is, and describe what would have to change to make it a true twin.

Design

1.8 Choose any everyday machine. Sketch it as a digital-twin diagram in the style of Figure 1.1, marking the sensors you would add, the data flowing up, and one decision that could flow back down.

Hands-On Task

No hardware or software is required yet — only observation. Choose one machine you can watch for a few minutes: a ceiling fan, a washing machine, a car, a 3-D printer. Sketch it as a digital-twin diagram in the style of Figure 1.1. On your sketch, mark (a) the sensors you would add to read its true state, (b) what the live data flowing up to a model would contain, and (c) one decision a twin could send back down to improve how the machine runs. Keep your sketch; in Chapter 2 you will compare it against the formal list of digital-twin components and see how close your instincts were.

PART V — THE PLC DIGITAL TWIN

Chapter 23 — PLCs and Digital Twins: Foundations

Learning Objectives

By the end of this chapter you will be able to:

Explain what a PLC is, how its scan cycle works, and why that determinism matters for twinning.

Name the five IEC 61131-3 languages and choose the right one for a given task.

Place the PLC correctly in the seven-layer digital twin model from Chapter 1.

Describe the key communication protocols — Modbus and OPC-UA — that bridge the PLC to a host model.

State why a PLC twin differs from an Arduino twin and when each approach is appropriate.

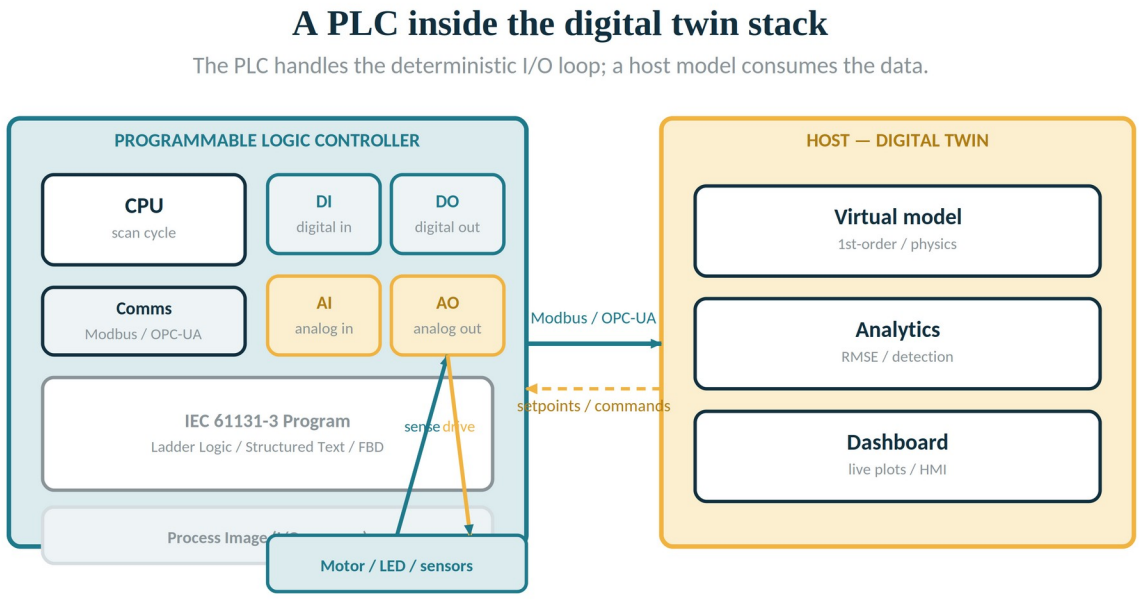

Figure 23.1. **A PLC inside the digital twin stack.** *The PLC handles deterministic I/O; a host model consumes the data over Modbus or OPC-UA.*

23.1 What Is a PLC, and Why Does It Matter for Twinning?

A Programmable Logic Controller is an industrial computer designed to control machinery reliably, in real time, and continuously — through vibration, dust, electrical noise, and temperature swings that would crash an ordinary laptop within a day. Unlike the Arduino, which runs one program from top to bottom with

no timing guarantee, the PLC repeats a fixed, deterministic scan cycle: read all inputs, execute the program, write all outputs, repeat. That cycle takes the same number of milliseconds whether the program is simple or complex, which means every measurement it produces carries an implicit, trusted timestamp. That determinism is exactly what a digital twin needs from its data source.

PLCs also carry communications built in. Every modern industrial controller speaks Modbus TCP, OPC-UA, or both, so connecting a host model is a matter of pointing a Python or MATLAB client at the right address — no serial port, no baud rate negotiation, no dropped packets. For a twin that must be production-ready, the PLC offers reliability guarantees no hobbyist board can match.

23.2 The Scan Cycle

Understanding the scan cycle is understanding the PLC. At the start of every scan the controller reads all its physical inputs into a memory snapshot called the process image. The program executes against this snapshot, so inputs cannot change mid-scan and create inconsistent state. At the end, the outputs are written from the image to the physical outputs. The whole cycle then repeats, typically one to ten milliseconds later, for the life of the machine. From the twin's perspective this means the data it reads always represents one coherent instant in the asset's history, which makes modelling and validation far cleaner than they would be with asynchronous polling.

Scan phase	What happens	Duration (typical)
Input scan	All physical inputs copied to the process image	< 1 ms
Program execution	Logic runs against the image, updates output image	1–10 ms (program-size dependent)
Output scan	Output image written to physical outputs	< 1 ms
Communications	Modbus / OPC-UA service handled in background	Asynchronous

23.3 IEC 61131-3: Five Languages, One Standard

The IEC 61131-3 standard gives PLC programmers five officially recognised languages, each suited to a different kind of problem. All five compile to the same instruction set and can coexist in one project, so a typical real application mixes them freely.

Language	Abbreviation	Best for
Ladder Diagram	LD	Relay-replacement logic, discrete switching, electricians comfortable with relay drawings
Structured Text	ST	Complex arithmetic, loops, string handling, data processing — closest to modern high-level code
Function Block Diagram	FBD	Signal processing chains, PID loops, calling standard blocks graphically
Sequential Function Chart	SFC	State machines, step-by-step sequences, start-up and shut-down procedures
Instruction List	IL	Low-level mnemonic code; largely obsolete, superseded by ST

This book uses Structured Text (ST) for all its PLC listings. ST reads like Pascal or modern pseudocode — variables, arithmetic, IF/THEN/ELSE, FOR/WHILE loops — which makes it the most transferable skill and the easiest to compare with the Python and MATLAB listings in earlier parts. Where Ladder Diagram is shown, it is for context only; the working listings are in ST.

23.4 The PLC in the Seven-Layer Twin Model

In Chapter 1 you met the seven building blocks of every digital twin: physical system, data acquisition, data communication, virtual model, monitoring, analytics, and decision. A PLC-based twin maps cleanly onto that stack, as Figure 23.1 shows. The PLC fills layers one through three in a single, ruggedised box: it is directly connected to the physical system, it acquires data on every scan cycle, and it communicates that data upstream via Modbus or OPC-UA. The host — MATLAB, Python, or a SCADA platform — fills layers four through seven. The interface between PLC and host is the only integration point you need to manage, and both protocols make it straightforward.

23.5 Modbus: the Universal Translator

Modbus is the simplest and most widely supported protocol in industrial automation. It defines a small set of function codes for reading and writing blocks of 16-bit registers. On a PLC you expose the variables you want the twin to see by mapping them to holding registers; on the host you open a TCP connection to the PLC's IP address and read those registers by number. The entire exchange is stateless and human-readable, which makes debugging trivial and implementation reliable even over unstable links.

The one limitation of Modbus is its flat, numeric address space: every variable is just a register number, and the host needs a separate map to know what each

number means. For small benches this is fine; for larger systems, OPC-UA's self-describing node tree is more maintainable.

23.6 OPC-UA: the Industrial Standard

OPC Unified Architecture is the modern, vendor-neutral information-exchange standard for industrial systems. Where Modbus gives you register numbers, OPC-UA gives you a named, typed, hierarchical node tree — so a variable might live at a path like ns=2;s=Motor.Speed and be self-describing in the units it returns. OPC-UA also provides built-in security (encryption, authentication, and certificates) and a subscription model: a client subscribes to a node and the server pushes updates only when the value changes, avoiding unnecessary polling. For a production twin talking to a real PLC, OPC-UA is the right choice.

Box 23.1 — Reality Check: simulation PLCs for learning

> *Physical PLCs cost hundreds to thousands of dollars, which is out of reach for a bench exercise. For learning, several free options replace hardware. CODESYS (free runtime on a Raspberry Pi or a Windows PC) speaks both Modbus and OPC-UA and supports all five IEC 61131-3 languages. OpenPLC runs on any Linux device. PLCopen simulators integrate with MATLAB Simulink. Any of these will exercise every listing in this part of the book without a hardware PLC in sight.*

23.7 PLC vs Arduino: Choosing the Right Edge

Both the Arduino (Parts II and III) and the PLC serve as the sensing-and-actuation edge of a twin, but they are not interchangeable. The Arduino is cheap, instantly programmable, and ideal for rapid experimentation where flexibility matters more than reliability. The PLC is industrial-grade, deterministic, and suited to any deployment that must run continuously and safely in a real environment. A good engineer knows when to use each: the Arduino for a student bench or a prototype, the PLC for a production machine, a teaching factory, or any asset whose downtime has a real cost.

Attribute	Arduino	PLC
Cost	< $10	$200–$2000+
Programming language	C/C++ (proprietary sketch environment)	IEC 61131-3 (ST, LD, FBD, SFC, IL)
Scan-cycle determinism	None (loop runs at variable speed)	Hard real-time, fixed scan period
Industrial	USB serial / SoftSerial	Modbus RTU/TCP, OPC-UA,

Attribute	Arduino	PLC
communications		PROFINET, EtherNet/IP
Durability	Consumer-grade	Industrial-rated (vibration, temperature, EMC)
Appropriate for	Experiments, student labs, prototypes	Production plant, teaching factory, industrial DT

Chapter Summary

- A PLC executes a fixed, deterministic scan cycle that gives every measurement an implicit, trustworthy timestamp.
- IEC 61131-3 defines five languages; this book uses Structured Text (ST) for clarity and comparability with Python and MATLAB.
- In the seven-layer twin model the PLC fills layers one through three; the host fills layers four through seven.
- Modbus is the simplest integration point; OPC-UA is the standard for production systems, adding security and self-describing nodes.
- Software PLCs (CODESYS, OpenPLC) let you exercise every listing in this part on a Raspberry Pi or PC at zero cost.

Key Terms

Term	Meaning
PLC	Programmable Logic Controller: an industrial computer built for real-time, deterministic machine control.
Scan cycle	The fixed read-execute-write loop a PLC repeats continuously.
Process image	The memory snapshot of all I/O taken at the start of each scan cycle.
Modbus TCP	A simple register-based protocol for reading and writing PLC variables over Ethernet.
OPC-UA	The modern industrial information-exchange standard: typed nodes, security, and subscriptions.
IEC 61131-3	The international standard defining the five PLC programming languages.

Review Questions — Chapter 23

Recall

23.1 List the three phases of a PLC scan cycle in order and say what happens in each.

23.2 Name three differences between Modbus and OPC-UA as integration protocols for a digital twin.

Application

23.3 Map the PLC, a SCADA system, and a Python twin model onto the seven-layer model from Chapter 1, stating which layers each one fills.

Analysis

23.4 Explain why the PLC's process image matters when a host is reading sensor data for a twin, and what could go wrong without it.

Design

23.5 For each of the following scenarios, choose Arduino or PLC and justify your choice: (a) a student lab bench, (b) a production conveyor belt, (c) a remote weather station.

Hands-On Task

Set up a software PLC. Install CODESYS or OpenPLC on a Raspberry Pi or a spare PC. Create a new project, write a ten-line ST program that increments a counter every scan cycle and exposes it as a Modbus holding register. From Python, open a Modbus TCP connection to the soft-PLC and read the counter; confirm it increments once per scan. This confirms your environment works before any motor or LED is involved.