جامعة الملك عبدالله للعلوم والتقنية
King Abdullah University of Science and Technology

أكاديمية كاوست
KAUST ACADEMY

Internet of Things (IoT)

DAY 3 Inside the device, and out to the cloud and the edge

August 2 to 6, 2026

KGSP Summer Enrichment Program 2026
KAUST Academy, KAUST

Salah Abdeljabar
salah.abdeljabar@kaust.edu.sa

🗺️ On Today's Agenda

1. **Components of an IoT application**: thing, internet, decisions
2. **In the cloud**: serverless functions & storage
3. **Edge computing**: processing data locally
4. **Security in IoT**: the "S" that isn't there
5. **Inside a microcontroller**: CPU, memory & I/O
6. **Programming microcontrollers**: RTOS & the Arduino framework
7. **Raspberry Pi**: single-board computer hardware
8. **SBC software**: Linux, Python & HATs
9. **MCU vs SBC**: choosing the right brain

Components of an IoT Application

The Thing, the Internet, and Decisions

The Thing + The Internet

🔌 The Thing

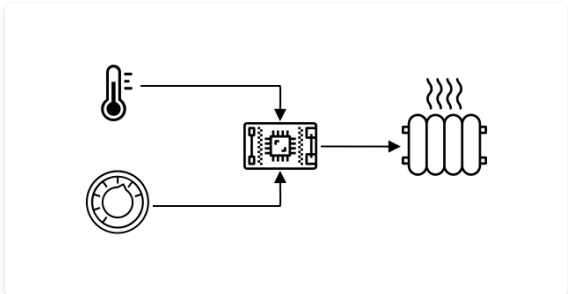
A device that interacts with the physical world, usually a small, low-power computer with **sensors** and **actuators**.

☁️ The Internet

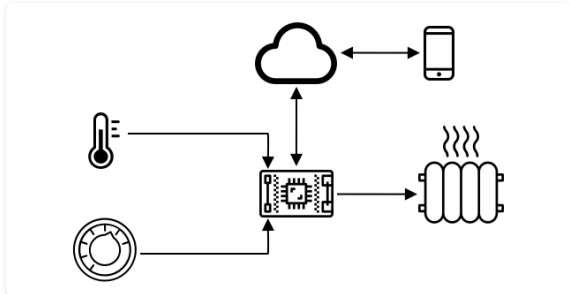
Cloud services the device connects to, in order to **receive** telemetry, **store & process** data, and **send commands** back.

> ⓘ **Note**
>
> Add **decision-making** (databases, AI, other data sources) and the loop is complete: **sense** → **send** → **decide** → **act**.

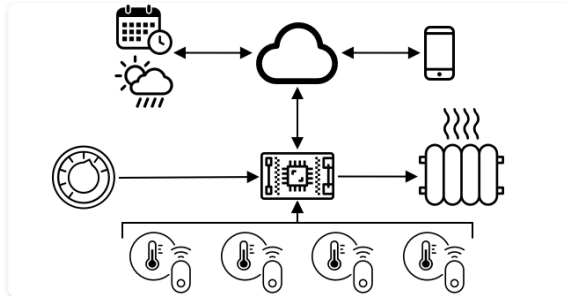
A Thermostat Gets Smarter



Basic: temperature + dial → heater



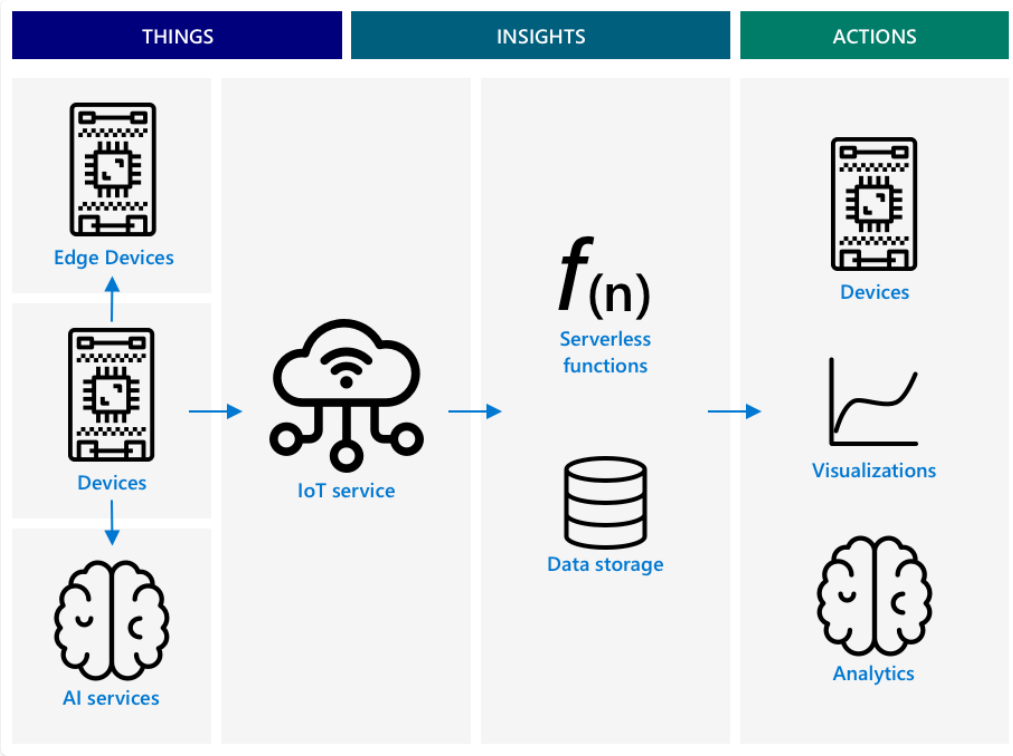
Connected: cloud + phone app



Smart: AI + calendar + weather

Each step adds more **data** and more **decision-making** in the cloud.

The Reference Architecture 🗺️



Things 🔌 Devices and edge devices sensing the world, some running AI at the edge.

Insights ☁️ An **IoT service** ingests the data, then **serverless functions** and **storage** turn it into meaning.

Actions ⚡ Commands back to devices, plus **dashboards** and **analytics** for people.

Tip

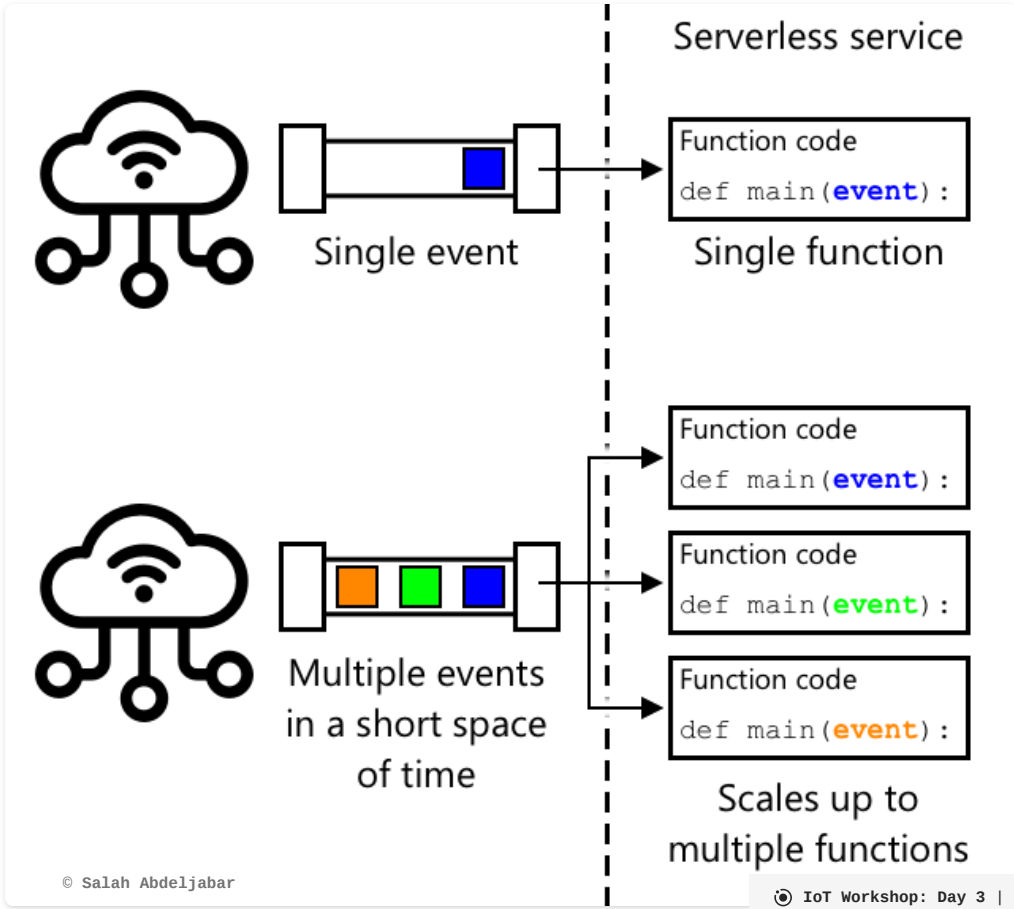
What the Cloud Does With It ☁️

Turning a stream of telemetry into insight

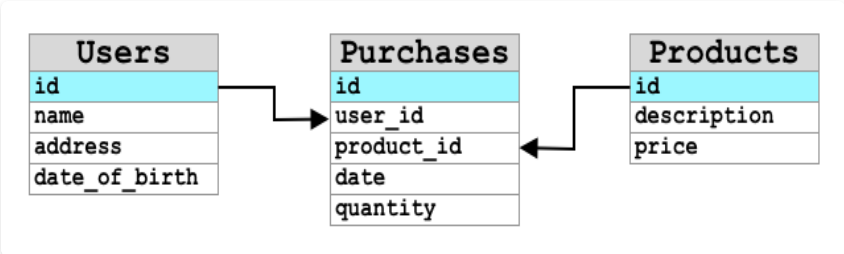
Serverless: Code Without a Server

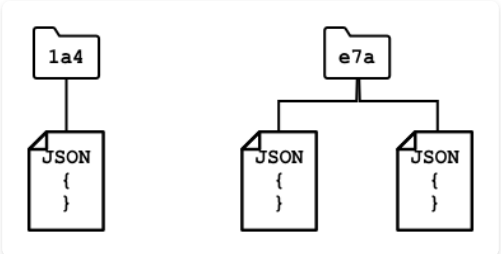
You write **one function** that handles **one message**. The platform runs it whenever a message arrives.

- No server to provision, patch, or keep running
- One message → one function run
- A burst of messages → many copies at once
- Pay only for the runs, so an idle device costs nothing



Where the Data Lands





SQL: fixed columns, rows linked between tables. Great when every reading has the same shape and you want to query across them.

NoSQL: documents keyed by id, each holding whatever JSON it likes. Great when devices send different fields.

> **Tip**
>
> IoT telemetry is often stored **raw** first and cleaned later: storage is cheap, and the question you'll want to ask next year hasn't been thought of yet.

Edge Computing 

Process data locally instead of always calling the cloud

What is the Edge?

Even though the **I** in IoT is "Internet," devices don’t always have to use it. **Edge devices** are gateways on your **local network** that process data without a round-trip to the cloud.

⚡ Speed

Handle lots of data or a slow connection without waiting on the Internet.

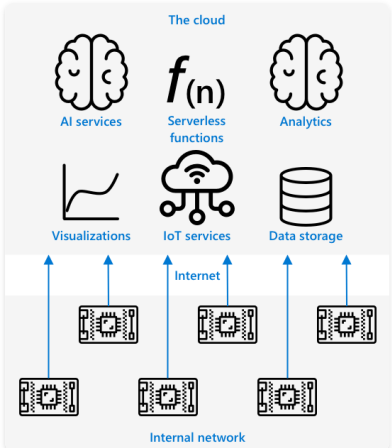
📴 Offline

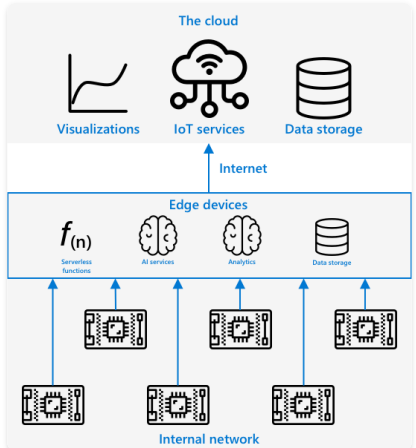
Keep working with no connectivity: on a ship, or in a disaster area.

🔒 Privacy

Sensitive data stays on the local device.

Without the Edge vs With the Edge





Without: every device crosses the Internet for every decision.

With: an edge device runs the same services on your network.

The cloud stack (functions, AI, storage) moves next to the devices. Only what matters goes up.

Edge in Action

A smart speaker listens for its **wake word** locally, only sending your speech to the cloud *after* it wakes.

Important

Everything said **before** the wake word (and **after** it stops listening) never leaves the device, keeping it private.

Security in IoT 

*"The **S** in IoT stands for Security"*

Only as Secure as the Cloud

An IoT device connects to a cloud service, so it's **only as secure as that service**. Open connections invite malicious data and attacks, with **real-world** consequences.

Stuxnet worm

Manipulated **valves in centrifuges** to physically damage them: a cyber attack with kinetic consequences.

Hacked baby monitors

Poor passwords let attackers access home cameras and monitors.

Air-gapping

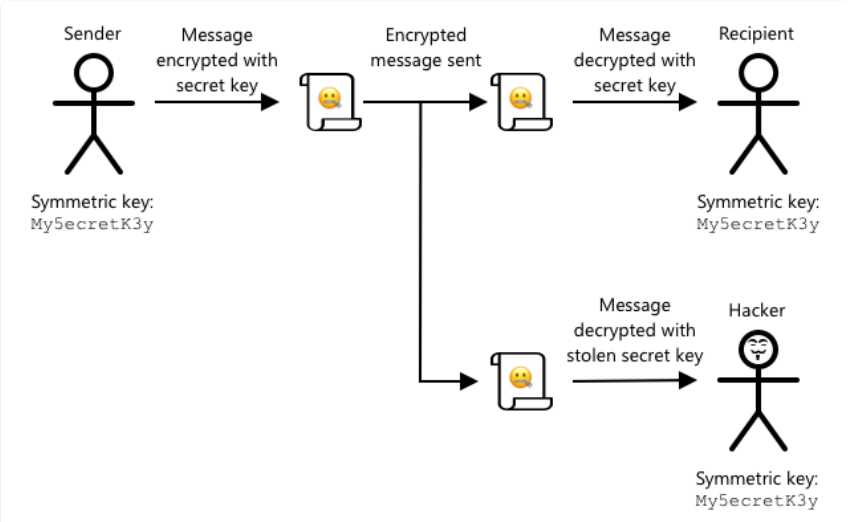
Some devices run on a network **completely isolated** from the Internet to stay private and secure.

Who Is Allowed to Connect?

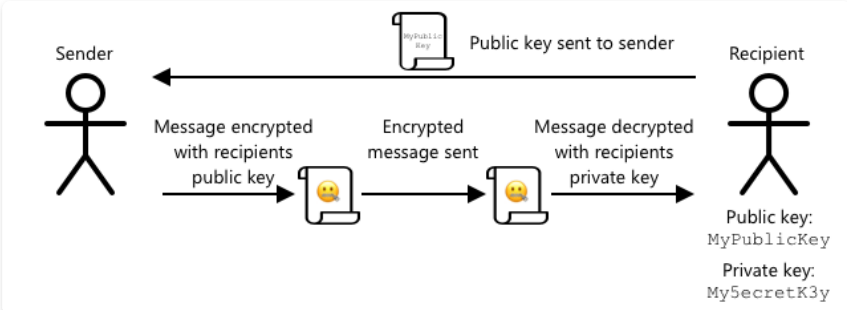
Every device gets its **own identity**: a secret key or certificate it presents when it connects. No valid credential, no connection.

> ⚠ **Warning**
>
> Per-device keys mean one compromised device can be `revoked` on its own, without locking out the whole fleet.

Keeping the Message Secret



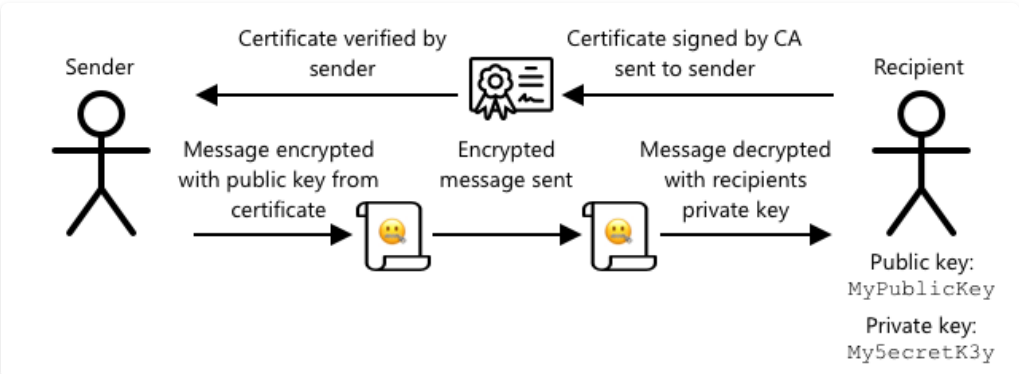
Symmetric: one shared key. Simple, but anyone who steals it can read everything.



Asymmetric: a **public** key encrypts, only the **private** key decrypts.

> **Note**
> Slower than symmetric, so real systems often use asymmetric keys just to agree on a symmetric one.

Certificates: Proving the Key Is Theirs



A **certificate** signed by a certificate authority proves a public key really belongs to who it claims to.

- The sender **verifies** it before trusting the key
- Devices can present certificates instead of secret keys
- The same idea behind the padlock in your browser

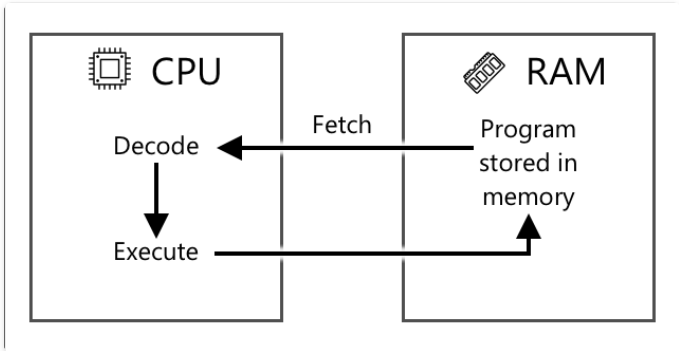
Inside a Microcontroller 

CPU · Memory · I/O

🧠 The CPU

The **brain** that runs your code, driven by a **clock** that ticks millions or billions of times a second.

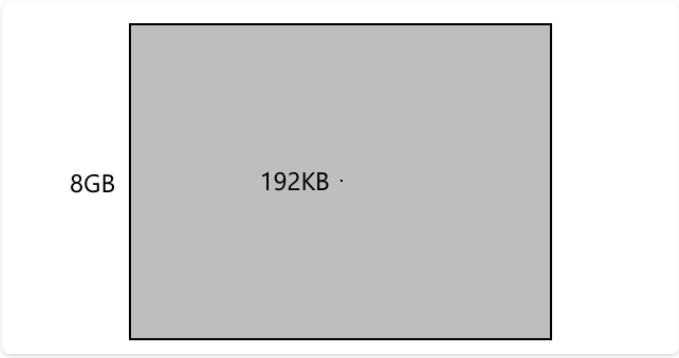
- Each tick = one step of the fetch–decode–execute cycle
- Speed in **Hz**: 1 MHz = 1M ticks/s, 1 GHz = 1B ticks/s
- An MCU runs at ~**120 MHz**; a PC at several **GHz**
- Slower = **cooler & lower power** → months on a battery



💾 Memory

Two kinds, both **tiny** next to a PC:

- **Program memory**: non-volatile, keeps code without power
- **RAM**: volatile, holds running data; lost on power-off
- An MCU: ~**192 KB** RAM & **4 MB** storage
- A PC: **8 GB** RAM & **500 GB** storage (*~40,000× more RAM*)



The dot in the center is 192 KB vs 8 GB

🔌 Input / Output

Microcontrollers talk to the world through **general-purpose I/O (GPIO)** pins, each configurable in software:

🧠⬅️ Input

Read values **from sensors**

🧠➡️ Output

Send signals **to actuators**

Programming Microcontrollers 🛠️

RTOS and the Arduino framework

No Full OS: Just Enough

MCUs are too small for a desktop OS. Two common approaches let you program them:

⚙️ RTOS

A lightweight **real-time OS** for handling peripherals on time.

- FreeRTOS, Zephyr, Azure RTOS
- Multi-threading, networking, GUI

🔧 Arduino framework

The most popular choice for learning and making.

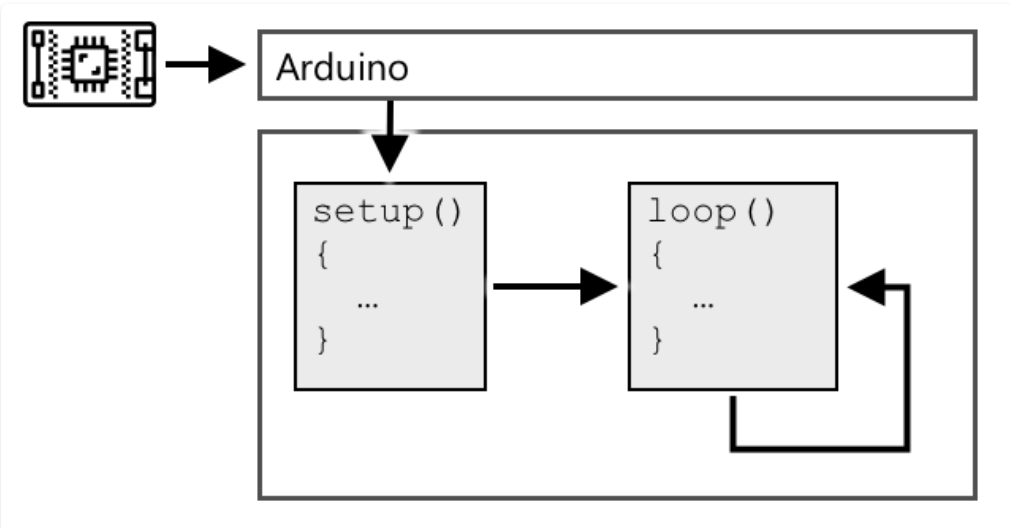
- Open-source, coded in **C/C++**
- Huge library ecosystem
- Portable across boards

The Arduino Sketch

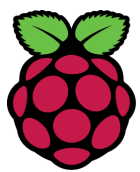
An Arduino program is built from just two functions:

- `setup()` : runs **once** at start (Wi-Fi, pins…)
- `loop()` : runs **forever**, again and again

> **Tip**
>
> This **event-loop** pattern quietly powers most desktop apps too.



Raspberry Pi 🍓



The most popular single-board computer

One Family, Many Sizes

Raspberry Pi 4B

- **Quad-core** 1.5 GHz CPU
- **2 / 4 / 8 GB** RAM
- Wi-Fi, dual 4K HDMI, USB 3.0
- **40 GPIO** pins, USB-C power
- From **US$35**

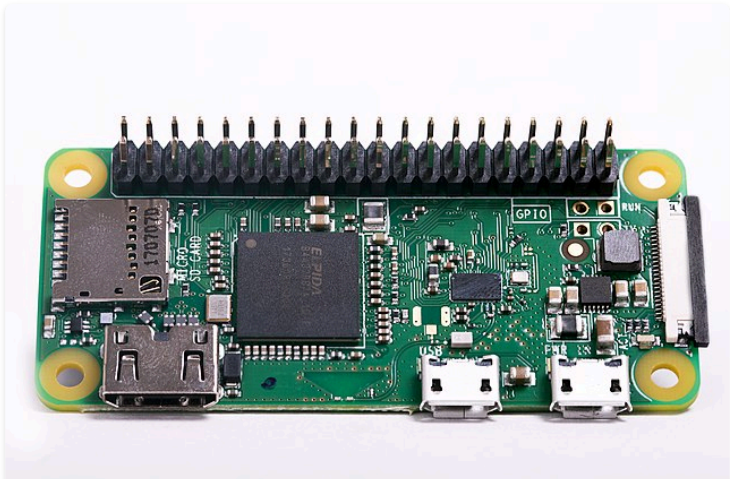
Raspberry Pi Zero

- **Single-core** 1 GHz CPU
- **512 MB** RAM
- Wi-Fi (Zero W model)
- **40 GPIO** pins
- 65×30×5 mm, **US$5–10**

One Detail Worth Knowing

> **Note**
>
> Pi CPUs are **ARM** processors, not the Intel/AMD **x86** chips in most PCs. The same ARM architecture powers nearly all **mobile phones** and **Apple Silicon** Macs.

All Pis run **Raspberry Pi OS**, a version of **Debian Linux**.



SBC Software 

Full operating systems, Python, and add-on boards

A Full OS Opens Doors

Because an SBC runs a complete operating system, you get a **full software ecosystem**.

🐧 OS & languages

- **Raspberry Pi OS** (Debian Linux): Lite (headless) or full desktop
- Program in **Python** with GPIO libraries
- Any Debian/ARM app runs

🎩 HATs

- Add-on boards that sit on the GPIO pins
- Screens, sensors, motor controllers
- Snap-in extra capabilities

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.output(18, GPIO.HIGH)   # turn a pin on
```

Beyond the Prototype

Important

SBCs aren’t just dev kits: they run **production IoT**, from controlling hardware to running **machine-learning models**.

Compute Module

The **Pi 4 Compute Module** packs Pi 4 power into a compact, cheaper board with fewer ports, designed to build into custom hardware.

MCU vs SBC 🆚

Choosing the right brain for the job

Side by Side

	🧠 Microcontroller	💻 Single-Board Computer
Cost	Cents to a few dollars	Tens of dollars
Power	Months on a battery	Usually mains-powered
Software	C/C++, no full OS	Linux + Python, full OS
Complexity	Single dedicated task	Many apps at once
Timing	Deterministic, real-time	OS-scheduled

Which Do I Pick?

Reach for an MCU

Battery-powered, **dedicated**, **real-time** tasks: a sensor node, a nightlight, a wearable.

Reach for an SBC

Complex logic, **multiple peripherals**, a **rich UI**, or **on-device AI**.

Tip

Many real products use **both**: an SBC coordinating several MCUs.