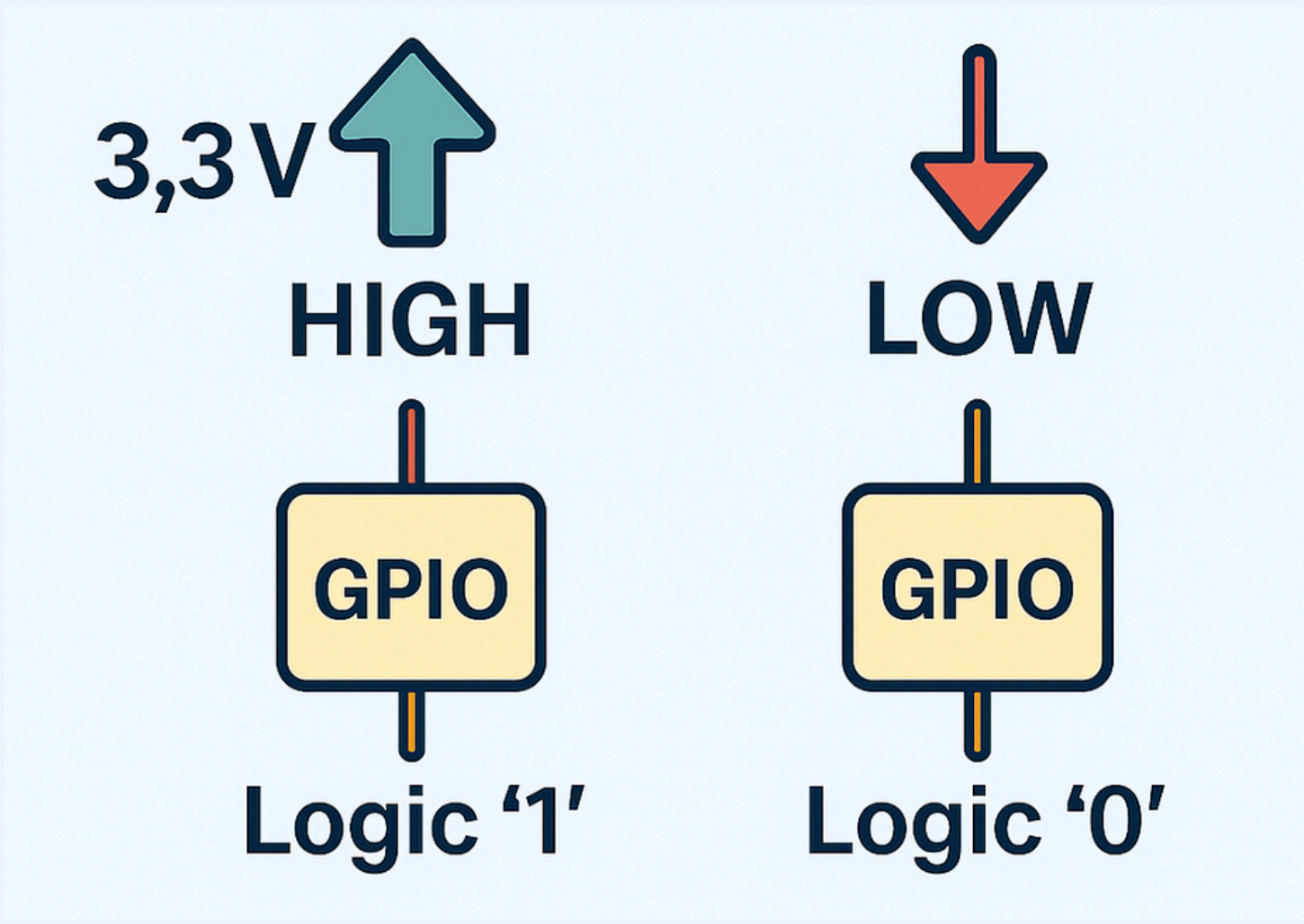
TA Embedded

#AWeekAPeripheral

General Purpose Input/Output (GPIO)

GPIO stands for **General Purpose Input/Output**. It refers to a digital pin on a microcontroller (MCU) or processor that can be programmatically configured as either:

- An input: to read a digital signal (HIGH or LOW, 1 or 0)
- An output: to write or generate a digital signal (set the pin HIGH or LOW



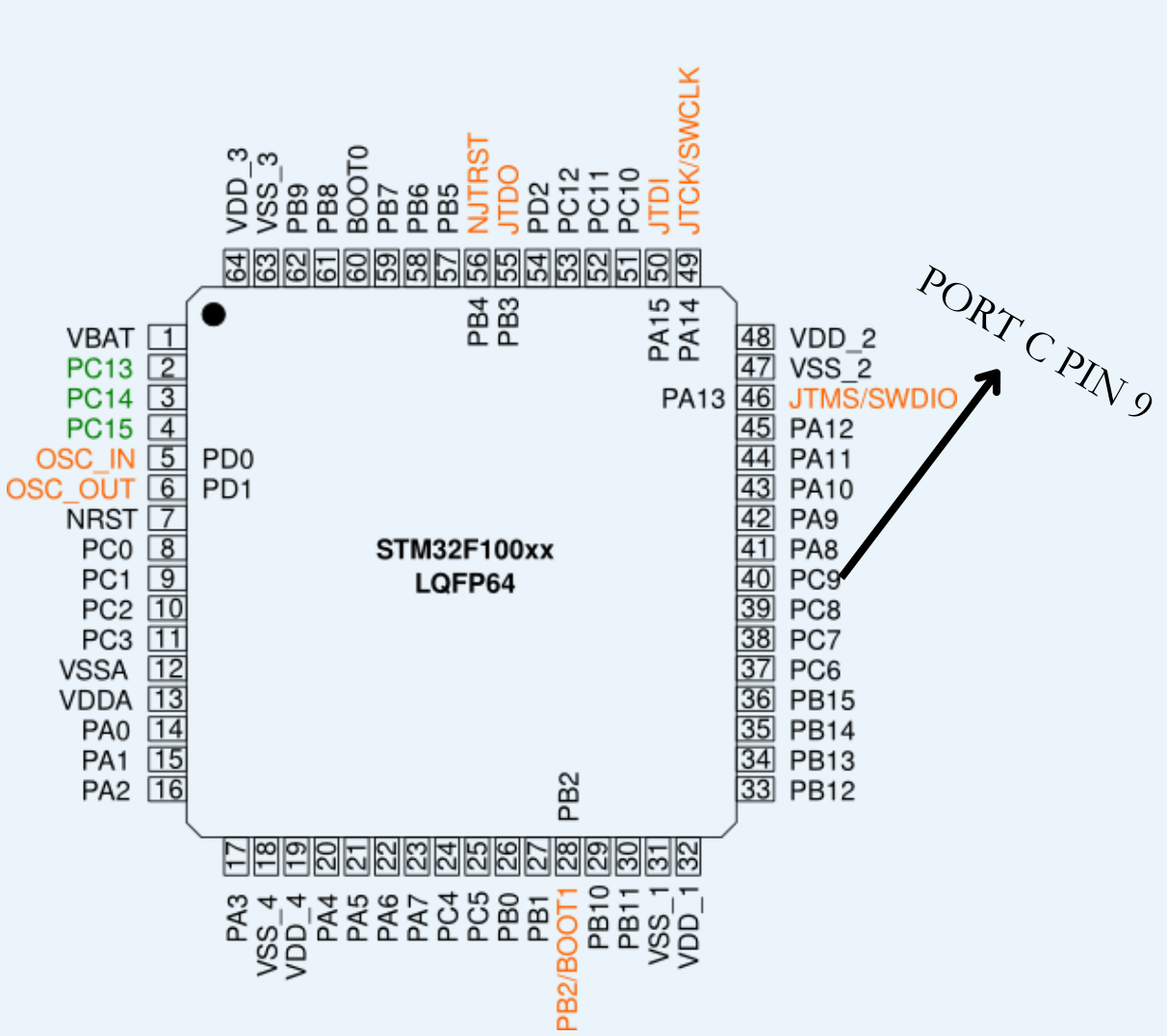
These pins are the primary interface through which a microcontroller interacts with the external world, reading buttons, sensing logic states, or driving LEDs, motors, and relays.

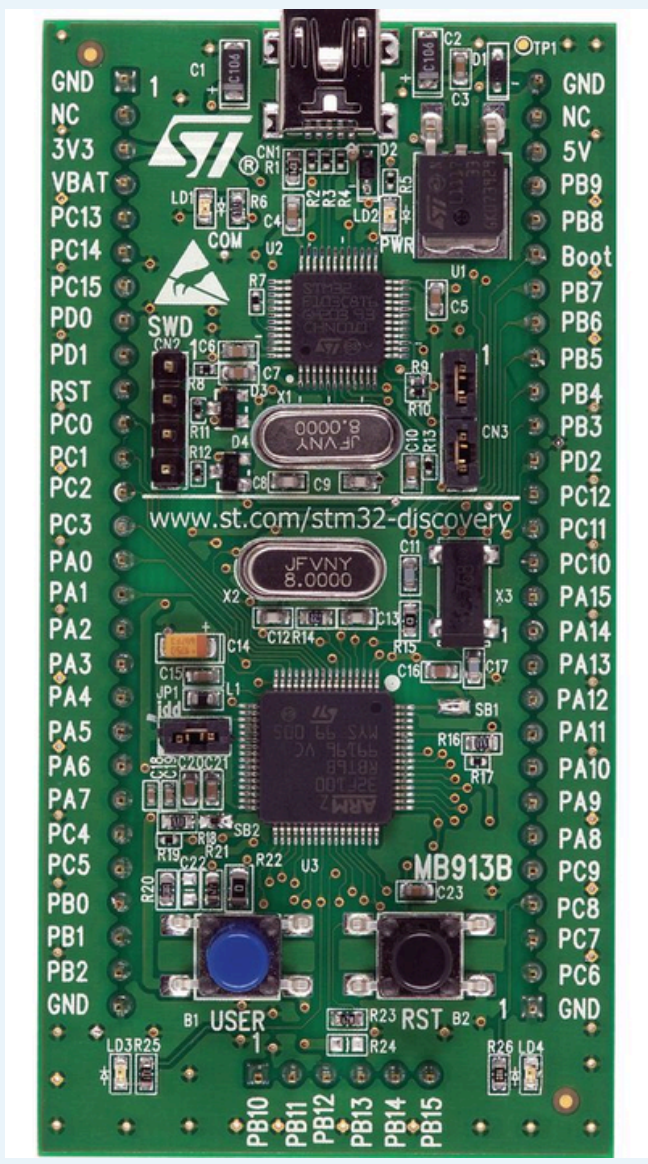
In most microcontrollers (like STM32, AVR, or PIC), GPIO pins are not isolated, they are organized into groups called ports.

What is a GPIO port?

A port is a logical grouping of GPIO pins that:

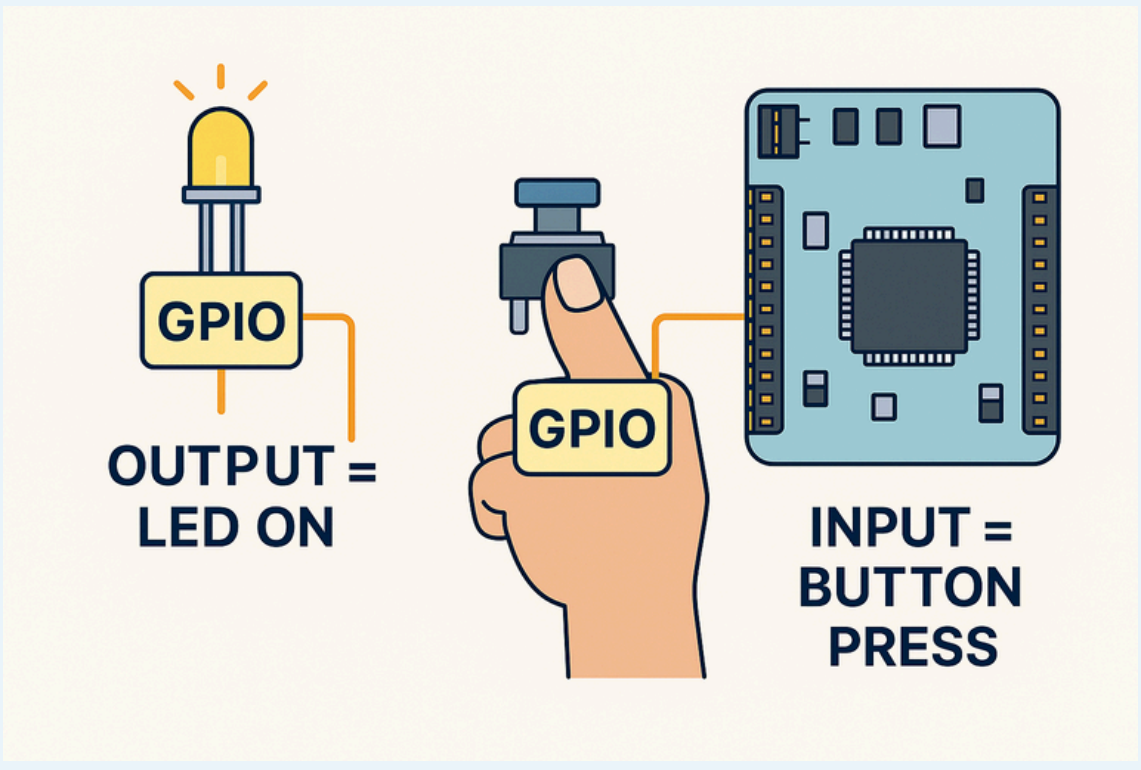
- Share a common base register (e.g., GPIOA, GPIOB, PORTC)
- Are accessed or configured collectively or individually
- Often contain 8, 16, or 32 pins (depending on the MCU)





NB : GPIOs are located on the microcontroller (μC) itself. The development board simply routes these GPIO pins to headers so you can connect external components easily.

GPIO in Action – Example



- When configured as input, the GPIO reads the button press, triggering logic in code.
- When configured as output, the GPIO drives an LED , turning it ON or OFF.

🔁 This is how your microcontroller sees and controls the world.

TA Embedded

#AWeekAPeripheral

🧱 GPIO is the building block of interaction , whether it's blinking an LED or sensing a button press, this is where software meets hardware.

❤️ Like, save, and share if GPIO was your first peripheral too.

👇 What was the first thing you built with GPIO?

🔁 Repost to help beginners and embedded communities grow

🔧 Stay tuned for upcoming posts with:

- GPIO configuration tips
- Real hardware examples using STM32 Nucleo boards