# Automatic Serial Port Detection for ESP32 on Linux

When you plug an ESP32 board into a Linux PC, it appears as a serial port (e.g. /dev/ttyUSB0 or /dev/ttyACM0). This device name can change depending on the order of connection or other devices, making it inconvenient to hard-code. The goal is to reliably identify and connect to the ESP32’s serial interface **without manually specifying which port** it’s on. Below are solutions that work every time on Linux, using unique identifiers to target your one ESP32 device.

## Using a Stable Device Identifier (/dev/serial/by-id)

Linux’s udev system creates persistent symlinks for USB serial devices under /dev/serial/by-id. These symlinks incorporate the device’s vendor, product, and (if available) serial number, providing a unique identifier for each device. Using this path ensures you always connect to the correct ESP32 board, regardless of whether it enumerates as ttyUSB0, ttyUSB1, etc[[1]](https://www.reddit.com/r/esp32/comments/vqfzvt/just_a_little_tip_to_make_your_esp32_kinda_work/#:~:text=instance). For example:

* **Find the ESP32’s by-id name:** Plug in the ESP32 and list the directory:
* ls -l /dev/serial/by-id/
* You should see an entry like usb-...-if00-port0 corresponding to the ESP32’s USB-UART interface. You can filter by known text – for instance, if using an Espressif-native USB (ESP32-S2/S3), the identifier might contain "Espressif". Running ls -l /dev/serial/by-id | grep -E 'Espressif|ESP' will show any serial ports with "ESP" in their name[[2]](https://www.reddit.com/r/esp32/comments/vqfzvt/just_a_little_tip_to_make_your_esp32_kinda_work/#:~:text=On%20Linux%2C%20you%20can%20search,Product%20and%20Manufacturer%20strings%20with). (For boards with an external USB-UART chip, the entry will show that chip’s manufacturer, e.g. "usb-Silicon\_Labs\_CP210x\_...").
* **Use the by-id path in your software:** The symlink you found (e.g. /dev/serial/by-id/usb-Silicon\_Labs\_CP2102N\_USB\_to\_UART\_Bridge\_Controller\_0001-if00-port0) always points to the ESP32’s current tty device. Open this path in your program instead of /dev/ttyUSBX. This way, you don’t need to guess the port number – the OS will map that by-id name to the correct port every time[[3]](https://blog.smittytone.net/2025/02/13/udev-i-dev-unique-identifiers-and-aliases-for-usb-serial-ports-on-linux-and-macos/#:~:text=Of%20course%2C%20the%20point%20of,them%20by%20an%20index%20number).

## Creating a Persistent Udev Symlink (Custom Name)

Another option is to define your own stable name using a udev rule. This is useful if you want a shorter or more convenient alias (for example, /dev/esp32) for the device.

1. **Identify the device’s attributes:** Use udevadm to get unique properties of the ESP32’s USB-UART. For example, run:

* udevadm info -a -n /dev/ttyUSB0
* (Replace ttyUSB0 with whatever port the ESP is currently on.) Look for attributes like **ATTRS{idVendor}**, **ATTRS{idProduct}**, and **ATTRS{serial}** in the output. These correspond to the USB vendor ID, product ID, and the device’s serial number string. For instance, an ESP32 dev board with a Silicon Labs CP210x USB chip might show idVendor=="10c4", idProduct=="ea60" (the IDs for CP210x) and a unique serial string[[4]](https://stackoverflow.com/questions/60414655/udev-rule-generating-symlink-for-usb-device-using-attrsserial-in-link-name#:~:text=SUBSYSTEM%3D%3D,).

1. **Write a udev rule:** Create a file (e.g. /etc/udev/rules.d/99-esp32.rules) with a rule matching those attributes and assigning a symlink name. For example:

SUBSYSTEM=="tty", ATTRS{idVendor}=="10c4", ATTRS{idProduct}=="ea60", ATTRS{serial}=="S101", SYMLINK+="esp32"

This rule says: when a tty device with VID 10c4 and PID ea60 and serial "S101" is present, create a symlink named /dev/esp32 pointing to it[[4]](https://stackoverflow.com/questions/60414655/udev-rule-generating-symlink-for-usb-device-using-attrsserial-in-link-name#:~:text=SUBSYSTEM%3D%3D,). Adjust the values to match your device (you can omit the serial filter if you only ever have one such device attached). The symlink will be created in addition to the normal /dev/ttyUSB\* node.

1. **Apply and test:** Save the rule and reload udev with sudo udevadm control --reload-rules. Unplug and re-plug the ESP32. Now it should consistently create the alias (for example, /dev/esp32) every time that specific board is connected. You can use this symlink in your programs. This approach guarantees that you always know what name to use for that device, regardless of which physical USB port or order it enumerated in[[3]](https://blog.smittytone.net/2025/02/13/udev-i-dev-unique-identifiers-and-aliases-for-usb-serial-ports-on-linux-and-macos/#:~:text=Of%20course%2C%20the%20point%20of,them%20by%20an%20index%20number). (Ensure the symlink name is unique and doesn’t conflict with existing names.)

## Programmatic Port Detection via Unique IDs

You can also have your software automatically discover the ESP32’s port by matching a unique identifier, rather than relying on a fixed name. Since you only have one ESP32 connected, the program can simply find the first device that matches the expected ID. Here are two common methods:

* **By USB Vendor/Product ID (VID:PID):** Every USB device has a vendor ID and product ID. For example, the CP210x USB-UART (used on many ESP32 dev kits) has VID 10C4 and PID EA60[[4]](https://stackoverflow.com/questions/60414655/udev-rule-generating-symlink-for-usb-device-using-attrsserial-in-link-name#:~:text=SUBSYSTEM%3D%3D,). In code, you can enumerate serial ports and look for these values. For instance, using Python’s PySerial library (v3+):

import serial.tools.list\_ports as list\_ports  
esp\_port = None  
for port in list\_ports.comports():  
 if port.vid == 0x10C4 and port.pid == 0xEA60: # Silicon Labs CP210x USB IDs  
 esp\_port = port.device # e.g. "/dev/ttyUSB0"  
 break  
  
if esp\_port:  
 print(f"Found ESP32 on {esp\_port}")  
else:  
 print("ESP32 not found")

This loop finds the first serial port with the matching VID/PID and retrieves its device name[[5]](https://stackoverflow.com/questions/38661797/is-it-possible-to-refer-to-a-serial-device-by-vendor-and-device-id-in-pyserial#:~:text=So%20for%20those%20looking%20to,PID)[[6]](https://stackoverflow.com/questions/38661797/is-it-possible-to-refer-to-a-serial-device-by-vendor-and-device-id-in-pyserial#:~:text=from%20serial). (If you were dealing with multiple identical devices, you could also check port.serial\_number or other fields to pick a specific one, but since you have just one ESP, matching the VID/PID is sufficient to identify it.)

* **By device description or serial string:** You can also match the port using human-readable identifiers. PySerial provides a convenient list\_ports.grep() function that can filter ports by a substring in their description/HWID. For example, if you know the USB vendor/product ID combo or a unique part of the device’s name, you can do:

import serial.tools.list\_ports as list\_ports  
pattern = "0403:6014" # e.g. FTDI VID:PID in hex  
ports = list(list\_ports.grep(pattern))  
if len(ports) == 1:  
 esp\_port = ports[0].device

In this snippet, pattern is a string containing the VID:PID (0403:6014 is an example for an FTDI adapter) and list\_ports.grep() returns ports whose hardware ID string contains that pattern[[7]](https://stackoverflow.com/questions/38661797/is-it-possible-to-refer-to-a-serial-device-by-vendor-and-device-id-in-pyserial#:~:text=I%20have%20found%20that%20the,function%20comes%20in%20handy%2C%20like). We then select the device name. You could also grep for part of the manufacturer name or product string (for instance, "Espressif" if the ESP32’s native USB interface reports that). The result is that your code discovers the correct /dev/tty... device automatically.

Using these methods, your program can **auto-connect to the ESP32 without specifying a fixed port name**. The key is to leverage a unique identifier (hardware IDs or device metadata) that the ESP32’s USB connection provides. This ensures a reliable connection every single time, since the match will only succeed for the intended device[[1]](https://www.reddit.com/r/esp32/comments/vqfzvt/just_a_little_tip_to_make_your_esp32_kinda_work/#:~:text=instance). By targeting the device via its IDs, you eliminate any ambiguity about which serial port to open.

**Sources:** Persistent USB device naming on Linux[[2]](https://www.reddit.com/r/esp32/comments/vqfzvt/just_a_little_tip_to_make_your_esp32_kinda_work/#:~:text=On%20Linux%2C%20you%20can%20search,Product%20and%20Manufacturer%20strings%20with)[[1]](https://www.reddit.com/r/esp32/comments/vqfzvt/just_a_little_tip_to_make_your_esp32_kinda_work/#:~:text=instance); Udev rule example for stable symlink[[4]](https://stackoverflow.com/questions/60414655/udev-rule-generating-symlink-for-usb-device-using-attrsserial-in-link-name#:~:text=SUBSYSTEM%3D%3D,); PySerial port detection by VID/PID[[5]](https://stackoverflow.com/questions/38661797/is-it-possible-to-refer-to-a-serial-device-by-vendor-and-device-id-in-pyserial#:~:text=So%20for%20those%20looking%20to,PID) and by ID pattern[[7]](https://stackoverflow.com/questions/38661797/is-it-possible-to-refer-to-a-serial-device-by-vendor-and-device-id-in-pyserial#:~:text=I%20have%20found%20that%20the,function%20comes%20in%20handy%2C%20like); general device identification best practices[[3]](https://blog.smittytone.net/2025/02/13/udev-i-dev-unique-identifiers-and-aliases-for-usb-serial-ports-on-linux-and-macos/#:~:text=Of%20course%2C%20the%20point%20of,them%20by%20an%20index%20number).

[[1]](https://www.reddit.com/r/esp32/comments/vqfzvt/just_a_little_tip_to_make_your_esp32_kinda_work/#:~:text=instance) [[2]](https://www.reddit.com/r/esp32/comments/vqfzvt/just_a_little_tip_to_make_your_esp32_kinda_work/#:~:text=On%20Linux%2C%20you%20can%20search,Product%20and%20Manufacturer%20strings%20with) Just a little tip to make your ESP32 kinda work like a regular USB device : r/esp32

<https://www.reddit.com/r/esp32/comments/vqfzvt/just_a_little_tip_to_make_your_esp32_kinda_work/>

[[3]](https://blog.smittytone.net/2025/02/13/udev-i-dev-unique-identifiers-and-aliases-for-usb-serial-ports-on-linux-and-macos/#:~:text=Of%20course%2C%20the%20point%20of,them%20by%20an%20index%20number) udev, I dev: unique identifiers and aliases for USB serial ports on Linux and macOS | smittytone messes with micros

<https://blog.smittytone.net/2025/02/13/udev-i-dev-unique-identifiers-and-aliases-for-usb-serial-ports-on-linux-and-macos/>

[[4]](https://stackoverflow.com/questions/60414655/udev-rule-generating-symlink-for-usb-device-using-attrsserial-in-link-name#:~:text=SUBSYSTEM%3D%3D,) linux - udev rule generating symlink for USB-device using ATTRS{serial} in link name - Stack Overflow

<https://stackoverflow.com/questions/60414655/udev-rule-generating-symlink-for-usb-device-using-attrsserial-in-link-name>

[[5]](https://stackoverflow.com/questions/38661797/is-it-possible-to-refer-to-a-serial-device-by-vendor-and-device-id-in-pyserial#:~:text=So%20for%20those%20looking%20to,PID) [[6]](https://stackoverflow.com/questions/38661797/is-it-possible-to-refer-to-a-serial-device-by-vendor-and-device-id-in-pyserial#:~:text=from%20serial) [[7]](https://stackoverflow.com/questions/38661797/is-it-possible-to-refer-to-a-serial-device-by-vendor-and-device-id-in-pyserial#:~:text=I%20have%20found%20that%20the,function%20comes%20in%20handy%2C%20like) python 3.x - Is it possible to refer to a serial device by vendor and device ID in pySerial? - Stack Overflow

<https://stackoverflow.com/questions/38661797/is-it-possible-to-refer-to-a-serial-device-by-vendor-and-device-id-in-pyserial>