Rysera Intermediate Course

Week 1 Booklet

Session 1: Icebreaker

Saturday, 14 March 2026 • 3:00 PM

What Are We Building?

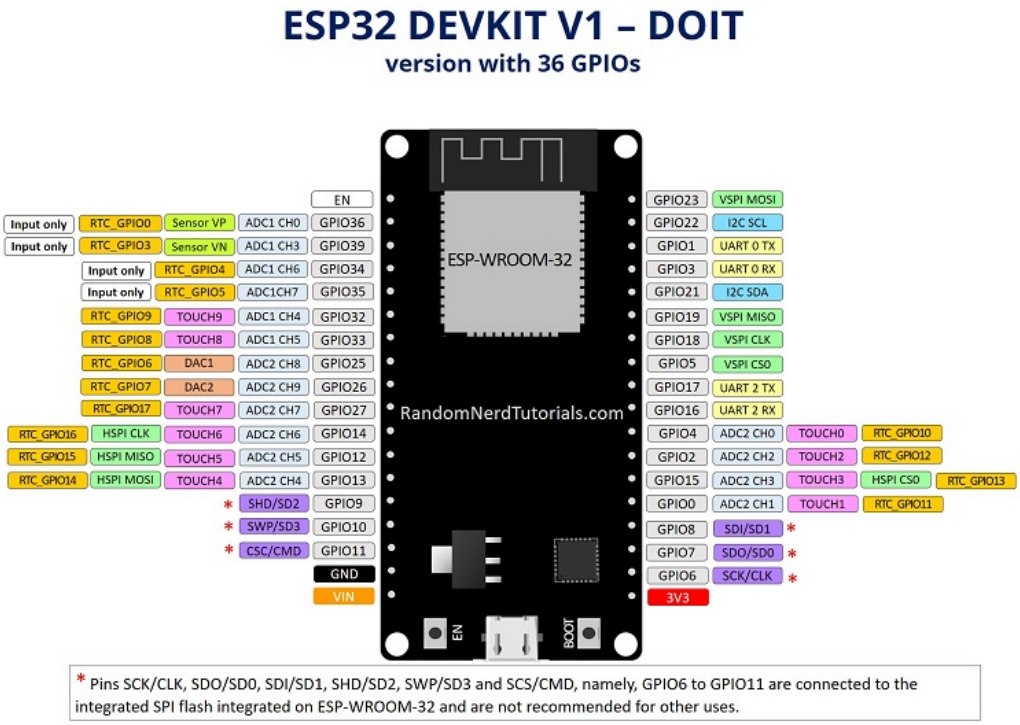
The centrepiece of this course is the **Medibox** — a smart medicine dispenser that you will design, build, and program yourself, from scratch.

The problem: People often forget to take their medicine at the right time. This can be dangerous, especially for older people or those managing multiple medications every single day.

Our solution: A device that reminds you when it is time to take your medicine, shows you the current time, monitors the environment your medicine is stored in, and can even be controlled from your phone.

By the end of this course you will have:

- → Wired real electronic components onto a breadboard and a PCB
- → Designed your own PCB (Printed Circuit Board)
- → Written C++ code to control your device
- → Connected your device to the internet
- → 3D-designed and printed an enclosure for your Medibox



The **ESP32 Dev Board** — the brain of your Medibox. Wi-Fi + Bluetooth + 38 programmable GPIO pins.

Medibox Requirements

For our Medibox to work properly, it needs to be able to do the following:

1. **Two push buttons** — used to navigate through the menus on the OLED screen.
2. **Show current time** — the time is fetched from the internet via Wi-Fi and displayed on screen.
3. **Connect to Wi-Fi** — so the device stays updated, syncs time, and can communicate with your phone.
4. **Set medicine alarms** — schedule exactly when each medicine needs to be taken.
5. **Ring an alarm** — a buzzer sounds when it is medicine time, so you never miss a dose.
6. **Multiple menu screens** — an OLED display lets you navigate through settings and information.
7. **Monitor temperature & humidity** — tracks the storage environment to keep your medicine safe.
8. **Connect with your phone** — control and monitor via the Blynk app (covered in a later session!).

Software You Need

Over this course you will be using three pieces of free software. Here is a quick overview:

Software	What it's for	Where to get it
Arduino IDE	Write and upload code to the ESP32 microcontroller	`arduino.cc`
KiCad	Design printed circuit boards (PCBs)	`kicad.org`
FreeCAD	Design a 3D enclosure for your finished device	`freecadweb.org`

Your instructor will help you install all of these during today's session. You do not need to have anything set up in advance.

Component 1 — Push Buttons

A push button is simply a switch. When you press it, it closes a circuit — the two sides of the button connect — and the ESP32 can detect that as a signal.

Understanding the 4 Pins

A standard push button has four legs (pins). They are arranged in two pairs:

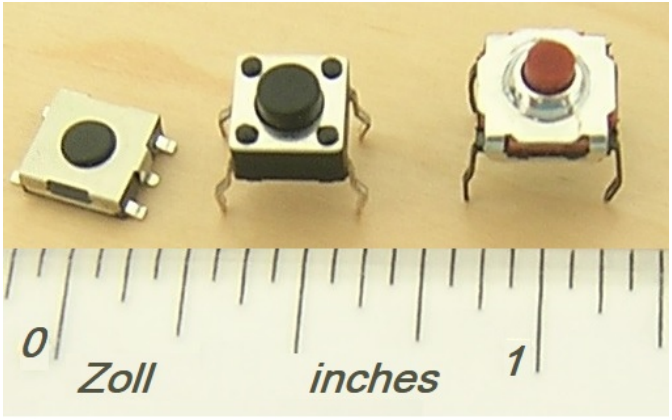
→ The two pins on the **left side** are always connected to each other.

→ The two pins on the **right side** are always connected to each other.

→ When you **press the button**, the left side and right side join — completing the circuit.

Tip: use one pin from each side when wiring — one connects to your GPIO pin, the other goes to GND.

We use **two** buttons on the Medibox: one for navigating **Left / Up** and one for **Right / Down**.



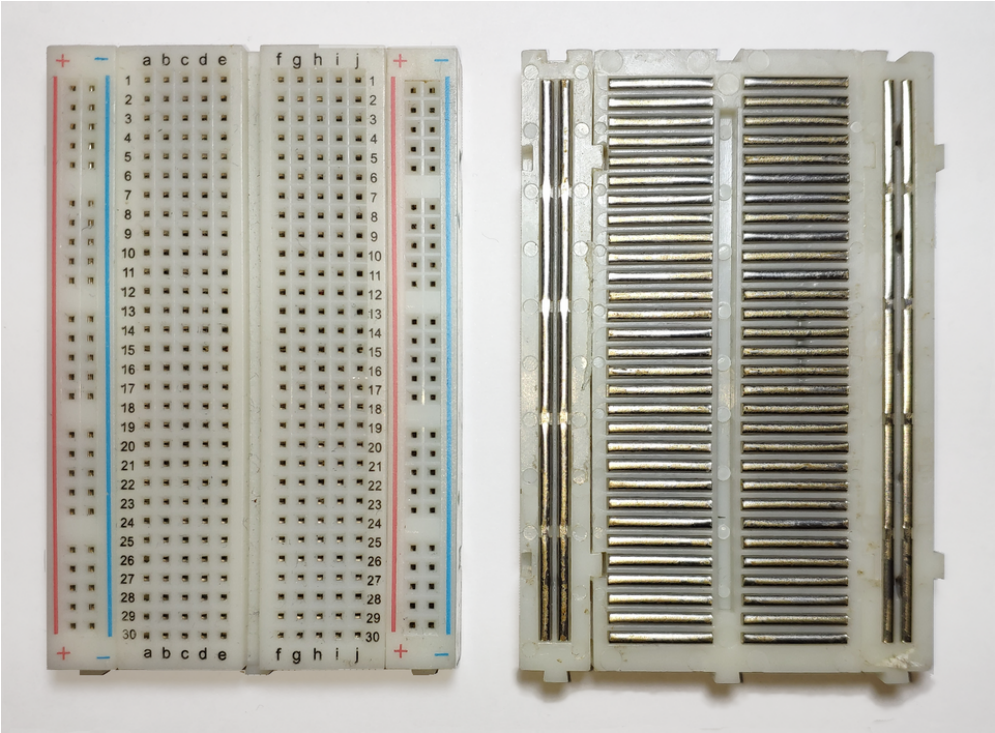
Tactile push button — 4 pins, two pairs. Use one pin from each side: one to GPIO, one to GND.

Button	GPIO Pin	Other Leg
Button 1 (Left / Up)	GPIO 34	GND
Button 2 (Right / Down)	GPIO 35	GND

Activity 1 — Wire Two Buttons 5 minutes

1. Place both push buttons on your breadboard, straddling the centre gap.
2. Wire **Button 1**: one leg to **GPIO 34**, the opposite leg to **GND**.
3. Wire **Button 2**: one leg to **GPIO 35**, the opposite leg to **GND**.
4. Open the starter sketch from Moodle and upload it to your ESP32.
5. Open the **Serial Monitor** — press each button and watch `"Left!"` or `"Right!"` appear.

Challenge: Can you make it print your name when you press *both* buttons at the same time?



A **breadboard** lets you build circuits without soldering. Holes in the same row are connected. The red/blue rails carry power (+) and ground (–).

Component 2 — Keeping Time

Here is something surprising: the ESP32 has **no built-in clock**. If you cut the power and restart it, it has absolutely no idea what time it is. So how does our Medibox know the correct time?

The answer is NTP (Network Time Protocol) — a worldwide service that provides the exact current time over the internet. Your ESP32 connects to a time server, sends a simple request, and gets back the current time in milliseconds.

How does NTP work?

1. Your ESP32 connects to Wi-Fi.
2. It sends a request to a time server (e.g. `pool.ntp.org`).
3. The server replies with the current time as milliseconds since 1 January 1970.
4. The ESP32 converts that number into hours, minutes, and seconds and displays it on screen.

The whole handshake takes less than a second!

There are two ways to think about time in your code:

→ **Counting up:** Start from zero when the device turns on and count seconds — like a stopwatch, but it does not know the real time.

→ **Real time:** Connect to Wi-Fi, fetch the time from an NTP server, and display the actual current time.

Activity 2 — Make a Counter

3 minutes

1. Open the counter sketch from Moodle.
2. Upload it to your ESP32.
3. Open the **Serial Monitor** and watch the numbers count up from zero.

Challenge: Can you make it count *down* from 60, like a timer?