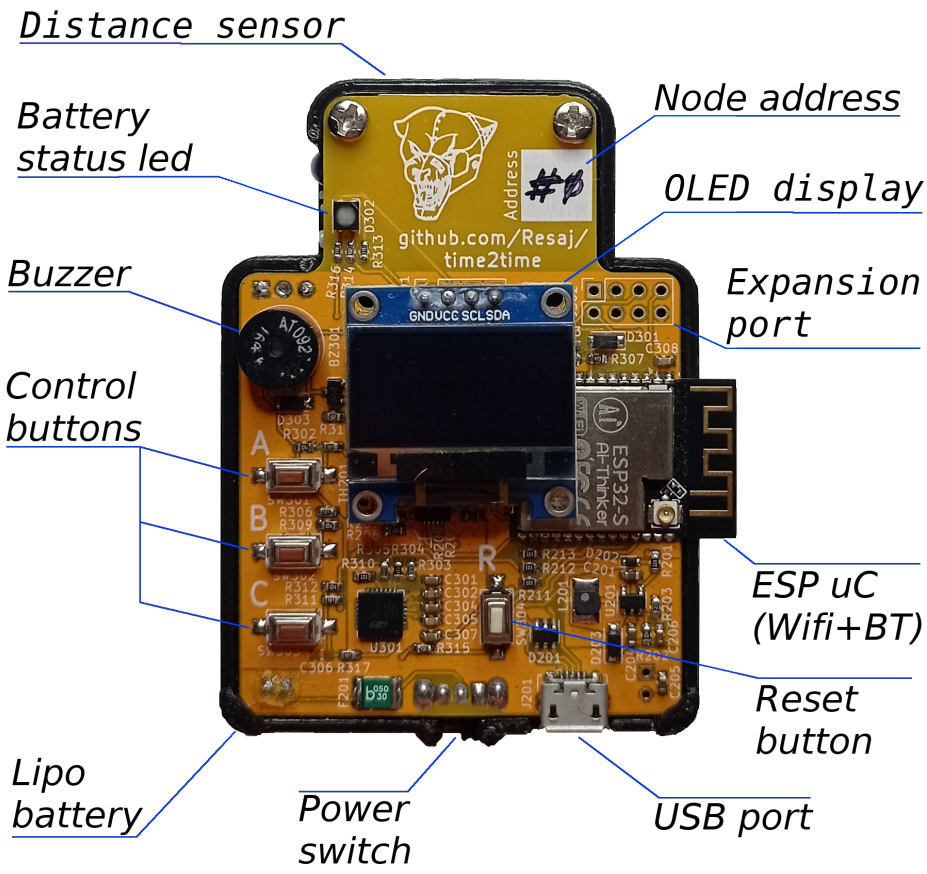
# USER GUIDE

## #time2time chronometer controls




- **Distance sensor**: detects the passage of the objects in front of the chronometer
- **Battery status led**: shows the battery status
- **Buzzer**: indicates through beeps the sensor detections and the record times according to the selected operation mode
- **Control buttons**: let controlling the chronometer's menu and selecting the different operation modes
- **Lipo battery**: is the chronometer power supply
- **Power switch**: lets switching the power supply between the battery and the USB port
- **USB port**: lets uploading the ESP32 firmware, charging the battery and supply the chronometer through USB
- **Reset button**: lets restarting the chronometer's firmware without disconnecting the power supply

- **ESP uC (Wifi + BT)**: ESP32 module with microcontroller which includes Wifi and bluetooth
- **Expansion port**: ESP32 available pins for another future purposes and #time2time improvements
- **OLED Display**: shows the chronometer's menu and the registered times
- **Node address**: node number to differ them easily while some of them are being used simultaneously

## ESP32 and firmware upload

The **ESP32 module** has been popularized in the maker's community because of its processing power, low cost, the easy programming and the integrated Bluetooth and Wifi for the wireless communication.

To program the module it's only necessary to have the **Arduino IDE** with the ESP32 libraries. Once connected to the computer through the microUSB connector, the firmware uploading is automatic by selecting the "Upload" option. Other modules maybe need to initialize the bootloader mode manually through a boot and reset buttons sequence. These buttons would be connected to the microcontroller, but #time2time incorporates this functionality yet, because the USB driver (CP2102) circuitry has been designed to facilitates the firmware uploading with this characteristic.

In this case, I have used Arduino because it's an easy IDE, but if you prefer another platform you can use another ones like PlatformIO.

The **main firmware** that contains all the #time2time functionality can be found at the project, in the folder **fw/t2t_fw**.

For a more detailed explanation about how to configure and upload the code into the #time2time nodes, read the **tutorial 2**.

## Buttons general functions (A, B, C and R)

The **A, B and C buttons** allow **selecting the functionality** that normally will be indicated in the menu, showed in the display. In the case of the user wants to get out of the actual operation mode, just press the button C.

The **R button**, or **reset** button, restarts the #time2time firmware.

## Lipo battery, status led and battery charge

The **Lipo battery** that #time2time includes is constantly monitored during the main firmware execution, which can be found at the project, in the folder fw/t2t_fw.

For the **battery charge**, connect #time2time through the USB to a external power supply. Also, turn on the power switch; in other cases the battery won't be connected to the device and won't be charged.

The **battery status led** shows the device supply status and the battery charging as long as #time2time is being powered by the battery and/or by USB. The possible states are as follows:

- Temperature fault or timer fault: fast red blink. Restart #time2time. If the fault persists, shutdown #time2time temporary in case the module has overheated or reprogram the module in case the firmware is corrupted. It rarely gets into this state.
- Preconditioning, constant current or constant voltage: slow yellow blink. It occurs when the battery is being charged. To get into this state the power switch has to be turned on and #time2time has to be powered by USB.
- Low battery output: slow red blink. It is necessary to charge the battery to follow with the normal operational mode and not to damage the battery with a too much low voltage.
- Charge complete: slow green blink. It occurs when the battery is totally charged, the power switch is turned on and the USB wire is connected, or when the battery is disconnected (or power switch turned off) and #time2time is being powered by USB.
- No battery present: fixed blue led. This is not a usual case, since the module must be powered in some way to turn on the led.
- USB disconnected: slow cyan blink. #time2time is being powered by the battery and it's voltage is right.

## Distance sensor adjustment

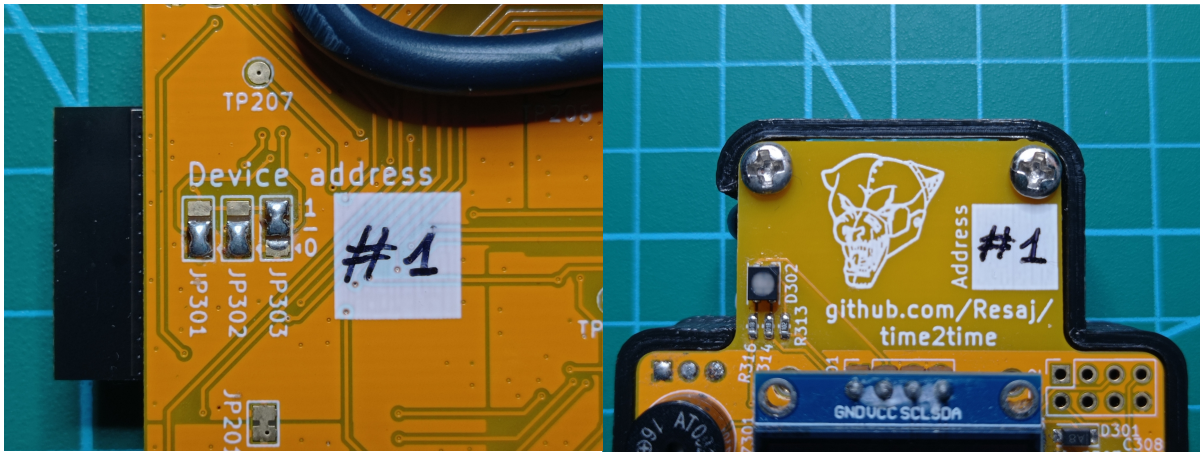
The **distance sensor** requires to be regulated for a good detection. As this is a more extensive topic, it will be covered in **tutorial 3** to explain it.

## Wireless communication and node addresses

Every #time2time node has two addresses:

- **MAC address**: is specific to each ESP32 module. It can be found on the rear sticker of #time2time.

- **Node address**: is specified with the rear jumpers on the #time2time PCB. It can take values between 0 and 7 for linking nodes in groups up to 8 nodes. This address is not essential, but it is used to make it easier for the user to sort the nodes.




Both addresses can be obtained by executing the test program that you can find at the project, in the folder fw/test.

In future tutorials the link between #time2time nodes will be explained in order to take partial times or times with different start and end of the route.

## Considerations and maintenance

As the general rule, it is important **not to expose the #time2time chronometer to high temperatures**, as this could damage the Lipo battery. In addition, the casing is printed of PLA, so it could start to deteriorate at 60ºC and above.