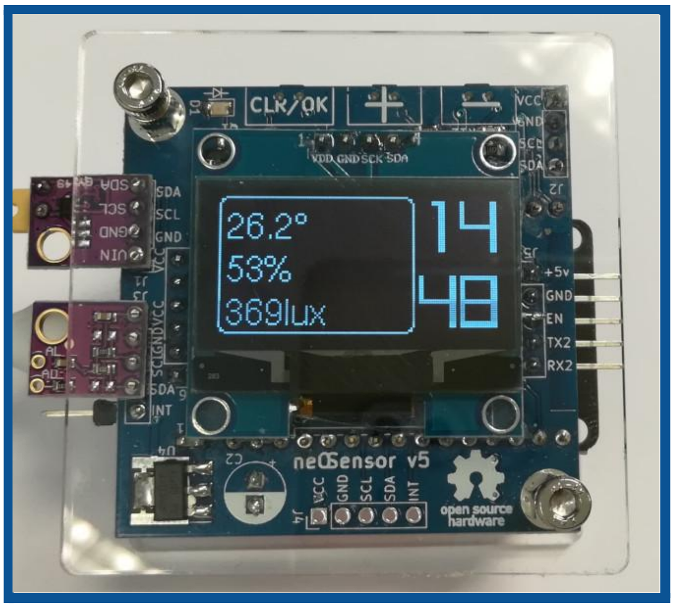


neOSensorV5 | setup guide

Dr Thiebolt François

Abstract

This document is related to the hardware and software setup of the ne**OS**ensor device. At the time of writing, we're talking about the setup of the neOSensor V5 and derivatives (i.e v5, v5.1).

Modifications table

Date	Note
apr.22	add 'User Guide' section
mar.22	added known issue about uniqueness of SHTxx and SCDxx per neOSensor
nov.21	added serial link @esp32 for CO2, PM sensors ...
aug.21	added PIR sensor HC-SR501 settings
jun.21	introducing neOSensor LoRaWAN, Heltec Cubecell based
feb.21	added neOSensor V5.1, esp32 based release
oct.20	generated new firmwares for neOSensor (esp8266 & esp32) and AirQuality (esp32)
sep.20	Upgrade to Arduino 1.8.13, ~~esp8266 core 2.7.4~~ (stay with SDK 2.7.1) First release to provide support for both esp32 and esp8266 :) Moved to github ;) added neOSensor and AirQuality boards in IDE's menu :)
may.20	Upgrade to arduino 1.8.12, esp8266 core 2.7.1
nov.19	Upgrade to arduino 1.8.10, esp8266 core 2.6.1, AP password
oct.18	Initial release

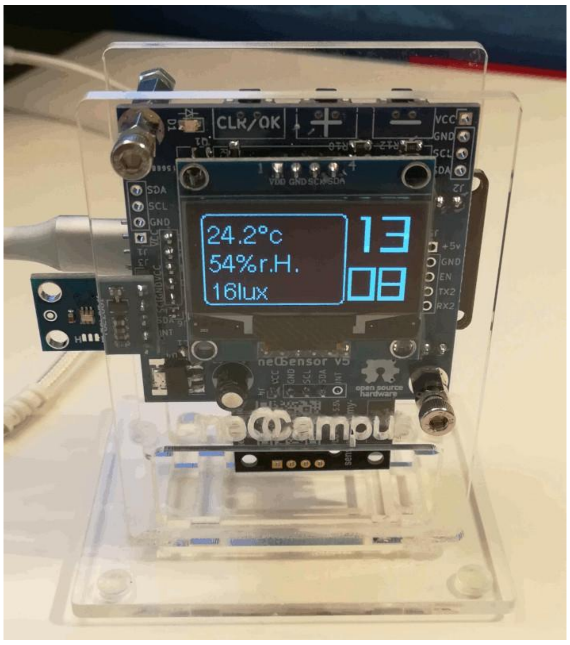
TABLE OF CONTENTS

Overview

15€ to build! open hardware WiFi LoRaWAN *(on way to)*

open source build your own

>>> if you're just interested in re-building the firmware on your own → Toolchains <<<



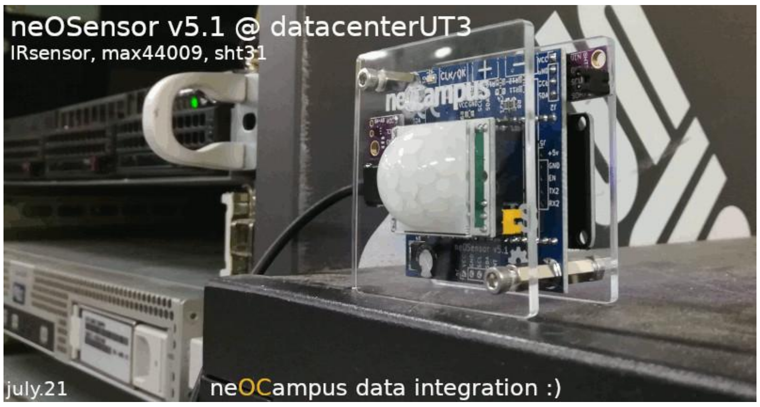
While starting in 2015, we were just searching for a simple (and cheap) way to monitor the ambient noise in the main library of our campus. Having been successful, we've quickly been asked to expand our ne**OS**ensor capabilities: temperature, humidity and luminosity digital sensors were added.

Nowadays, those cheap sensors spread across and above our campus while exhibiting support for a broader range of sensors. We now provide support for COV, PM and CO2 sensors, especially the latest sensirion SCD4x CO2 whose measurement is based on ultrasonic waves.

Last year, we started to add different kinds of sensors like PIR sensors and we also started to propose alternatives casing like a desktop version still made of 3mm acrylic elements.

At the same time we had undertaken a new, onboard data integration principle. This led to a dramatic decrease in the overall data produced while still maintaining the same high quality level of the data itself.

Latest developments geared us toward LoRaWAN networks. A first version based on CubeCell modules has been produced … more to come, stay tuned!



neOSensor v5.1 @ datacenterUT3
IRsensor, max44009, sht31
july.21
neOCampus data integration :)

onboard data integration

Each data of a sensor is acquired multiple times till it becomes stable according to a per class specific setup.

When a new data has been marked as stable, we'll check if it differs from the last one that has been sent to our MQTT broker:

data(new) - data(old) > x% ⇒ send

After having sent a new value, the sensor sleeps for a per class specified cooldown value (60s most of the time). Anyway, whatever happens, (e.g luminosity sensors in our datacenter are most of the time read at 0!) data is sent every 30mn at least.

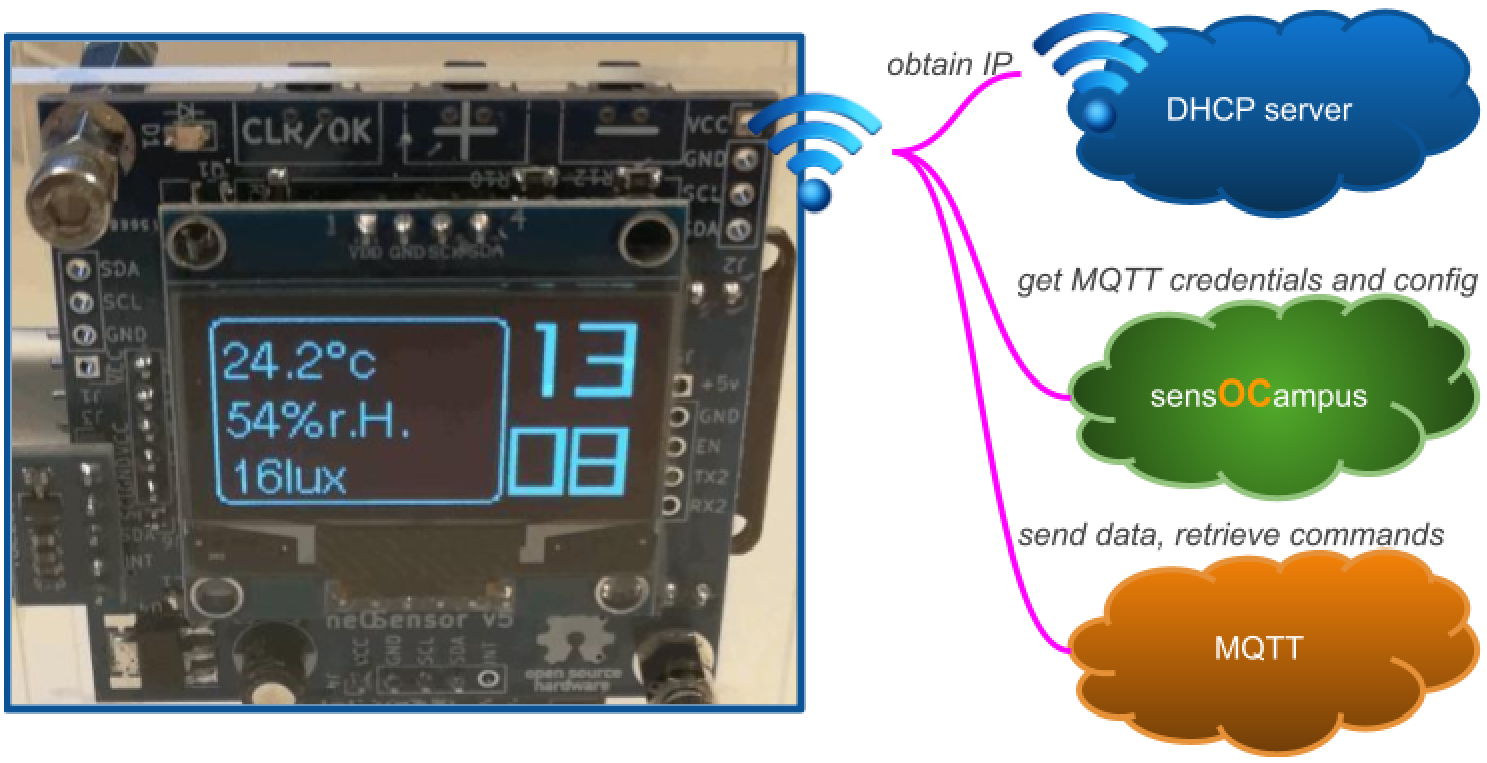
Services interactions

neOSensors end-devices send data and receive commands to/from a MQTT broker. More information about MQTT rules at:

https://neocampus.univ-tlse3.fr/_media/sensocampusv2_end-devices_api.pdf

Credentials allowing connection to this MQTT broker will be delivered by the sensOCampus web app. Alongside with its credentials, end-devices will get delivered their own specific configuration (if any); such config may encompass stuff like serial link to a sensor (speed, parity etc), analog sensors specifications … everything that the neOSensor is able to determine on its own.

It is worth mentioning that when you will configure your neOSensor, it will be possible to activate the **sandbox mode**; in this mode, no request to the sensOCampus will be issued and the device will be using the neOCampus MQTT sandbox.



Hardware releases

Since its inception in 2015, neOSensor has evolved over time :)

Date	Release	Module	Note
2015	neOSensor	esp8266 bare module	1st proto to BU
2017	neOSensor V2	esp8266 bare module	
2018	neOSensor V3	esp8266 bare module	
2018	neOSensor V4	esp8266 bare module	
2020	neOSensor V5	esp32 30 pin board	J.Maignan, removed noise sensor

2021	LoRaWAN	Heltec CubeCell dev. board	first LoRaWAN proto
2021	neOSensor V5.1	esp32 30 pin board	added support for IR sensors + analog + serial
2021	LoRaWAN V2	Heltec CubeCell dev. board	Faulty PCB
2021	LoRaWAN V2.1	Heltec CubeCell dev. board	

Links

[neOCampus] neOSensor project
https://neocampus.univ-tlse3.fr/projects/neosensor

[neOCampus] end-devices API and MQTT rules
https://neocampus.univ-tlse3.fr/_media/sensocampusv2_end-devices_api.pdf

[github] source code
https://github.com/fthiebolt/neOCampus-arduino

twitter
https://twitter.com/neOCampusUT3

neOSensorV5 hw setup

As shown on next page, neOSensors features the following interfaces:

- one I2C bus to connect all i2c sensors together,
- one serial link featuring an EN (i.e enable) command
- 3 digital inputs (buttons) on top,
- one led on top
- one digital input for a PIR sensor
- 4 analog inputs

I2c integrated circuits

You can connect several i2c ICs till you ensure a non overlapping of their i2c addresses. All devices are 3v3 powered. Some i2c interfaces feature an interrupt line used by smart sensors.

For example, the oled display (SH1106 128x64 monochrome) is an i2c device automatically detected at startup.

PIR sensor vs oled display

From a mechanical point of view, you ought to select between a PIR sensor (HC-SR501) or the oled display. From a configuration point of view, the PIR sensor can operate with the firmware default parameters (i.e without any additional configuration from sens**OC**ampus).

Serial link

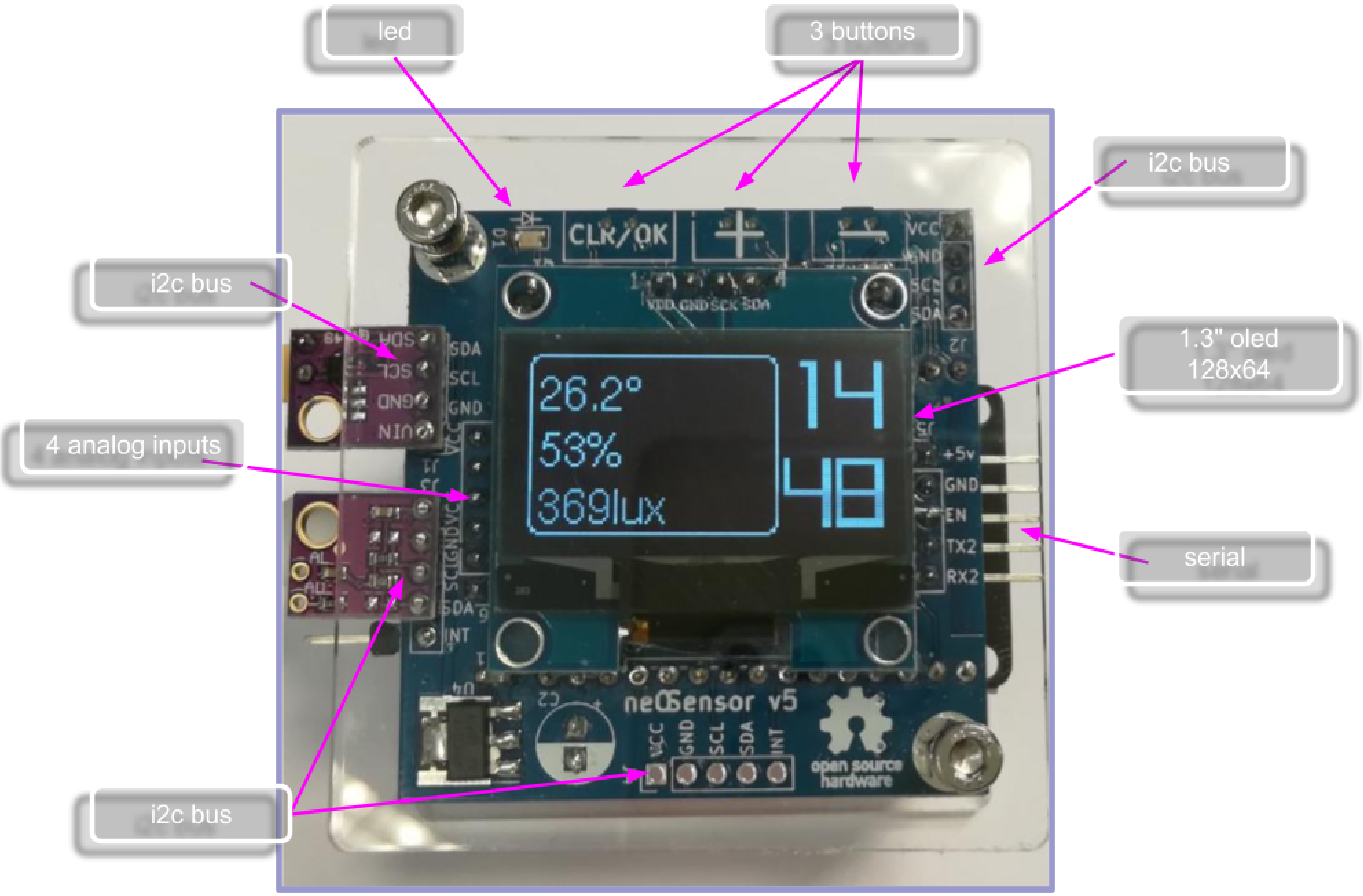
Used by some sensors like SDS011 or IKEA PM sensors, the interface provides a +5V along with an EN signal to activate or shutdown sensors that exhibit such capability.

Analog inputs

Only available on ESP32 boards, they are 3v3 max inputs. Hence, we provide a combination of aop + resistor divider to achieve a 5v → 3v3 seamless translation. Of course, you can change on your own the divider resistor bridge.

Power

An AMS1117-33 provides up to 800mA@3v3. It is unrelated to the (low power) 3v3 regulator located on the ESP32 devkit itself.


led
3 buttons
i2c bus
i2c bus
1.3" oled 128x64
4 analog inputs
serial
i2c bus
CLR/OK
26.2°
53%
369lux
14
48
neOSensor v5
VCC
GND
SCL
SDA
INT
+5v
EN
TX2
RX2
open source hardware

End-user setup

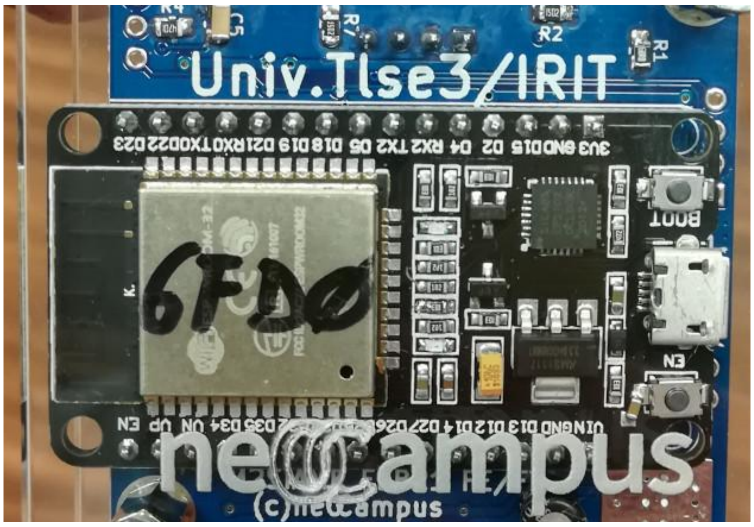
Roughly speaking, our neOSensors always follow these power up steps:

1. get connected to a WiFi network,
2. NTP servers connexion,
3. check for firmware upgrade at https://neocampus.univ-tlse3.fr/images/
4. grab MQTT credentials and configuration from https://sensocampus.univ-tlse3.fr
5. auto detect digital sensors,
6. [main loop] process sensors and send back the data to our MQTT broker.

As an end-user, you'll only get involved at the first step: setting up WiFi connection :)

Wifi connection

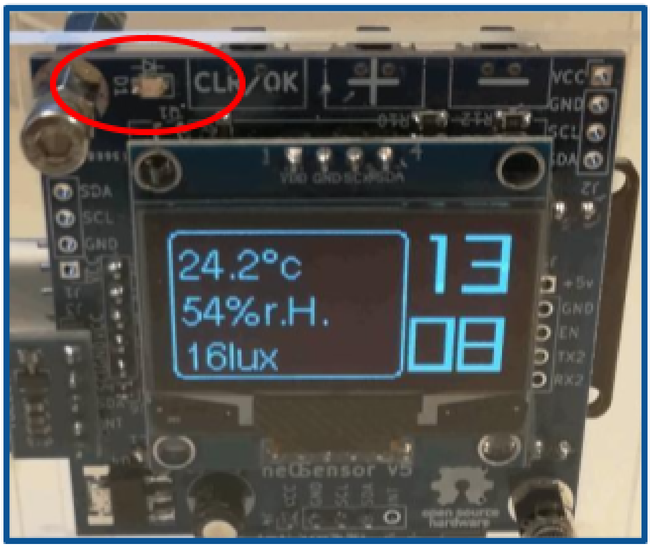
Probably the most important operation a user needs to carry out: setting up the network configuration of a neOSensor end-device.



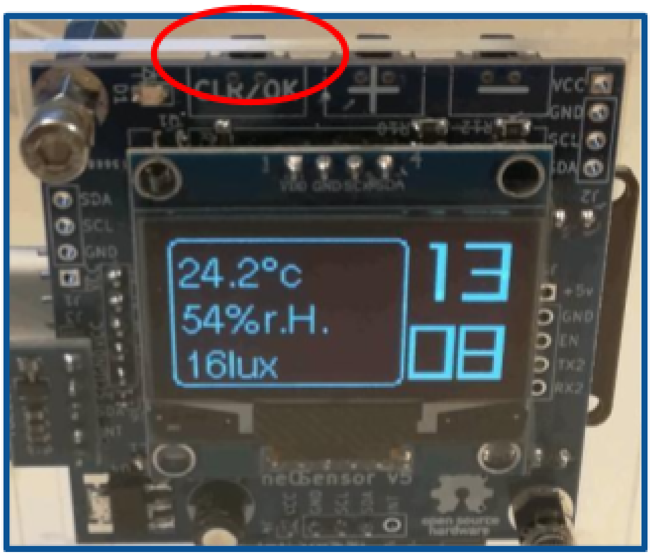
1. neOSensor activate a **WiFi Access Point** (i.e AP) named: *neOSensor_XXYY*
   The last four digits are the last two mac address bytes written on the esp itself. In this example your phone will discover a new WiFi hotspot named neOSensor_6fd0

   At this point, the front led will start fade in / fade out cycles indicating network connectivity is on the way.

led slowly fade in, fade out ⇒ network connectivity is on way

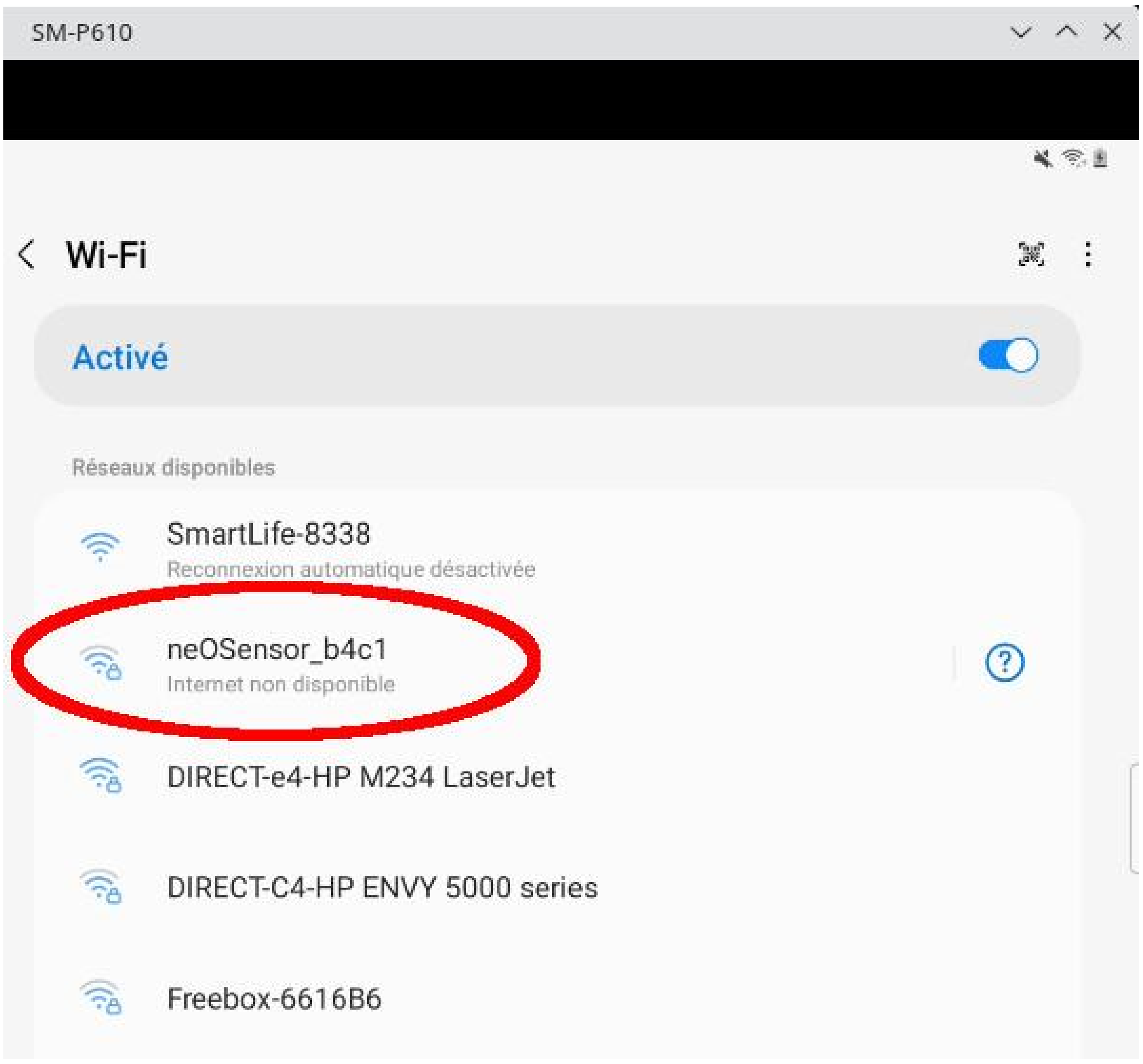


press CLR for 10s while powering the device ⇒ whole reset (i.e clear WiFi credentials, MQTT credentials, config etc)



*Note: if network credentials have already been set, neOSensor will try connection to the WiFi gateway during 90s. Once successfully connected, neOSensor_XXXX AP will stay open for **50s**. If the network connection fails, neOSensor_XXXX AP will get opened for **5mn**.*

Note: pressing the CLR button for 10s while powering up the device will clear everything related to both WiFi, MQTT and any configuration held onboard.


SM-P610

Wi-Fi

Activé

Réseaux disponibles

SmartLife-8338
Reconnexion automatique désactivée

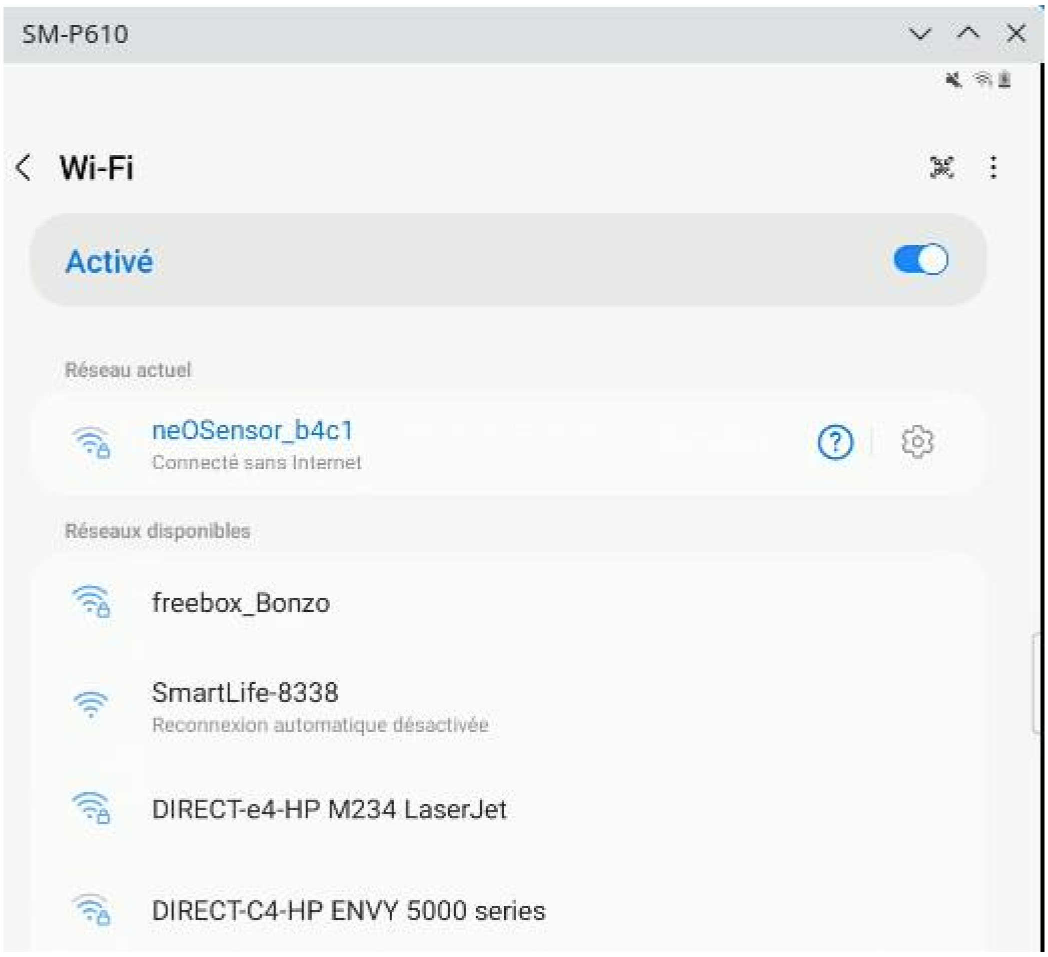
neOSensor_b4c1
Internet non disponible

DIRECT-e4-HP M234 LaserJet

DIRECT-C4-HP ENVY 5000 series

Freebox-6616B6

Scanning for WiFi networks, your smartphone just detected a neOSensor


SM-P610

Wi-Fi

Activé

Réseau actuel

neOSensor_b4c1
Connecté sans Internet

Réseaux disponibles

freebox_Bonzo

SmartLife-8338
Reconnexion automatique désactivée

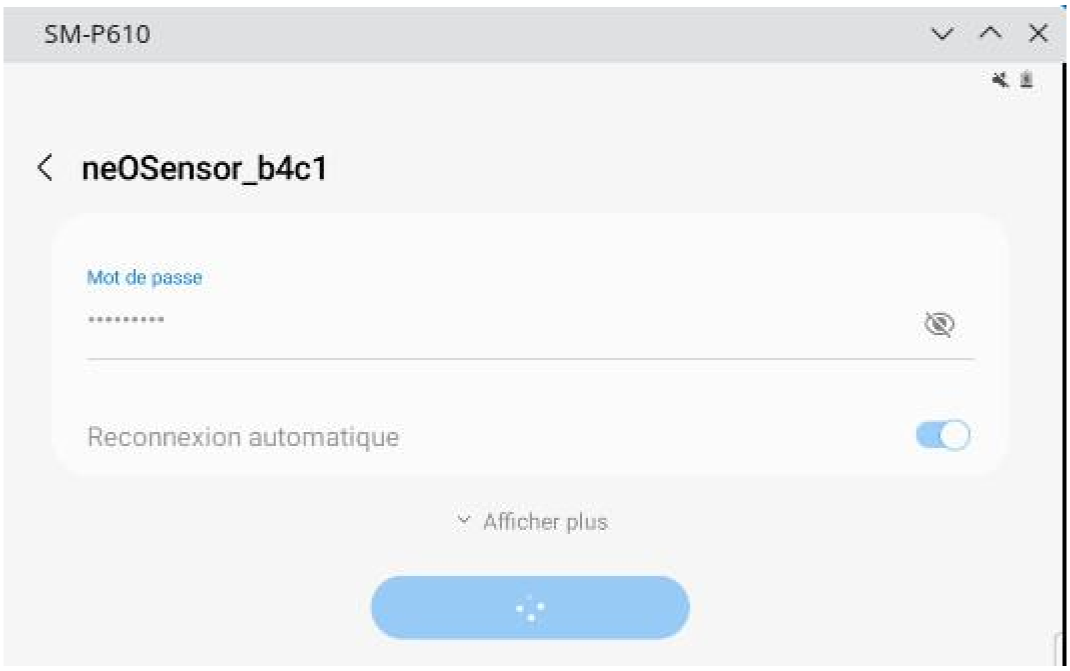
DIRECT-e4-HP M234 LaserJet

DIRECT-C4-HP ENVY 5000 series

Now you get connected to your neOSensor's AP.

Depending on smartphones, you may get automagically redirected to the captive portal located at https://192.168.4.1

2. **[optional]** You may get asked for a password if you selected AP protection at compilation time (disabled as default) [nov.19] AP mode password



SM-P610

neOSensor_b4c1

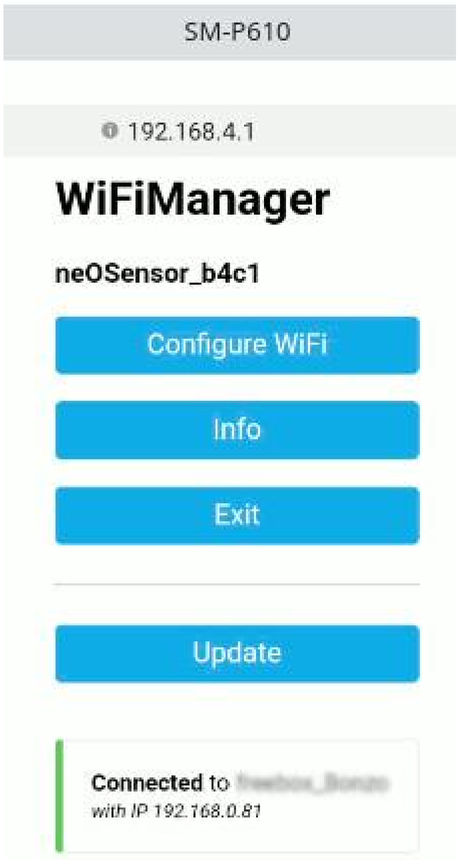
Mot de passe

•••••••••

Reconnexion automatique

Afficher plus

3. Once your phone will get connected to the neOSensor AP, you will be redirected to a **captive portal** enabling you to configure your end-device.



SM-P610

192.168.4.1

WiFiManager

neOSensor_b4c1

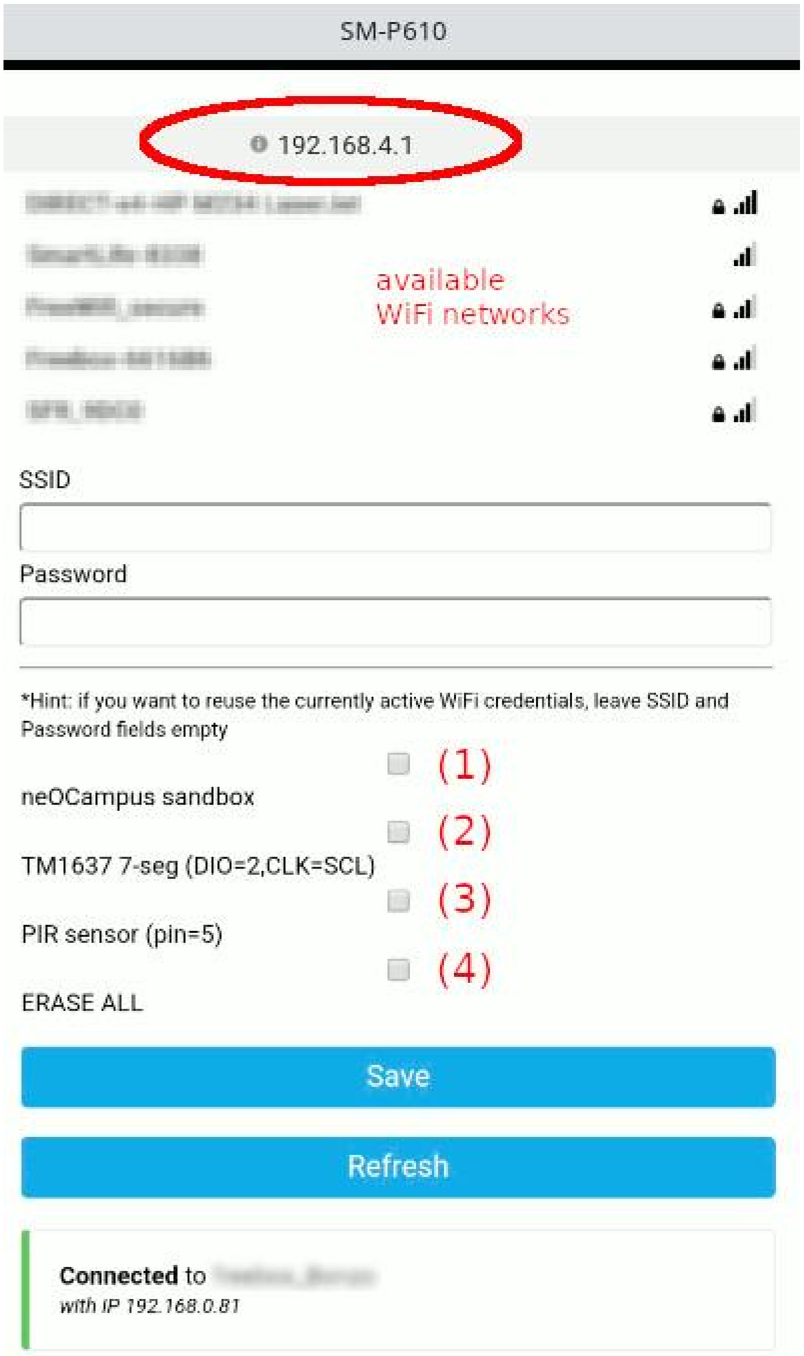
Connected to [redacted]
with IP 192.168.0.81

In case automatic redirection does not occur, head a browser to http://192.168.4.1

Click on 'Configure WiFi' to select the network you want your neOSensor to connect to.

4. This is the general neOSensor configuration page allowing you to select one of the listed WiFi networks and to select advanced options according to your needs.

   Note: for hidden WiFi network, just type the name and password in the corresponding text areas



SM-P610

192.168.4.1

available WiFi networks

SSID

Password

*Hint: if you want to reuse the currently active WiFi credentials, leave SSID and Password fields empty

neOCampus sandbox (1)

TM1637 7-seg (DIO=2,CLK=SCL) (2)

PIR sensor (pin=5) (3)

ERASE ALL (4)

Save

Refresh

Connected to [illegible]

with IP 192.168.0.81

5. **[optional]** advanced options

   (1) neOCampus sandbox

   In this mode, no access to the sensOCampus web server, instead the device will make use of the default neOCampus MQTT sandbox.

   (2) [internal use] only relevant to neOClock devices

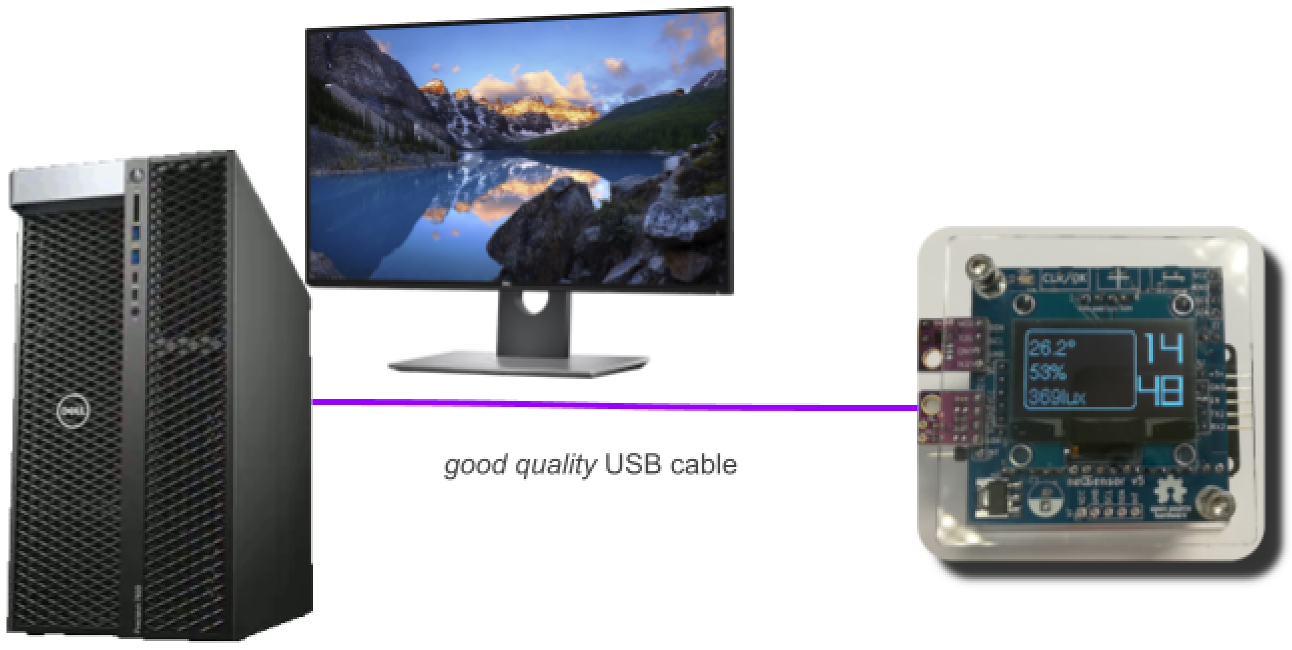
   (3) [internal use] DEPRECATED (nowadays, sensOCampus config will tell :))

   (4) will erase saved WiFi credentials along with a FLASH formatting hence deleting all locally held configurations (if any)

6. Now click on 'SAVE' and you can disconnect your smartphone from this WiFi AP
   Note: anyway, the WiFi AP will disappear on its own.

Serial monitoring

Especially with embedded systems, it may get frustrating now knowing what's happening … hence the serial monitor will come in handy :)



Serial link specifications:

- 115200 bauds
- 8N1

You can either monitor the serial link activity through some python application:

- in a tmux session, launch:
  pip3 install python3-pyserial
  `miniterm-3.py /dev/ttyUSB0 115200`

- **[recommended]** or just start your Arduino IDE and select *Tools → Serial monitor*

NTP servers

Your neOSensor will automatically connect a NTP server, either:

- (S)NTP server provided by your DHCP server,
- (S)NTP servers from pool.ntp.org

NTP sync will get automatically refreshed every 3600s and you will get some notice telling that NTP sync occurred.

Note: on its own, it is not really important being NTP sync'ed because data arriving on the MQTT *broker will get timestamped.*

(automatic) Firmware upgrades

Each time a neOSensor starts, after having successfully connected to a WiFi network, it will compare its current firmware version to the latest available on our servers.
Updating process is fully automatic and the neOSensor will reboot once its upgrade is over.

To ensure proper operation of the automatic firmware upgrade procedure, check that the network your neOSensor device is connected to has access to the URLs shown below.

Note: you can check on your own the latest revision of the firmwares your neOSensor will upgrade to:

- *[esp8266] https://neocampus.univ-tlse3.fr/images/esp8266/neOSensor/default.json*
- *[esp32] https://neocampus.univ-tlse3.fr/images/esp32/neOSensor/default.json*

As an example, at the time of writing, here is the content of the default.json for esp32

```
{
  "description": "esp8266 based neOSensor",
  "revision": 220328,
  "release_date": "2022-03-28",
  "image": "http://neocampus.univ-tlse3.fr/images/esp8266/neOSensor/firmware_latest.bin"
}
```

Note: right now, there's no option preventing an automatic firmware upgrade. One workaround is to flash your device with a higher revision than the one on the server. Fw revision is composed of two digits for <year><month><day>, eg. 220328 means 2022, March 28th

sensOCampus credentials and config

The sensOCampus web app is of internal use for all of our end-devices parts of our global infrastructure (i.e it is not intended for end-users access ---right now).

To ensure proper operation, check that the network your neOSensor device is connected to has access to the following URL:

- https://sensocampus.univ-tlse3.fr

Note: For the time being, a global rewrite of sensocampus2 is underway. This will include the ability for end-users to log in to check the status and configuration of their devices.

Changing MQTT data endpoint

As specified earlier, all of our devices send data to our MQTT broker which is part of the overall data acquisition chain.

Changing this data sink point can get achieved either ways:

- contact the neOCamps technical team so they will change in the sensOCampus configuration the MQTT target for your specific device,
- change in source code the default MQTT target and flash your custom firmware (pay attention to the (automatic) Firmware upgrades)

Henceforth, you need to ensure network connectivity to your custom MQTT broker. Also, get advised it will get requested to check for login/password along with topics ACLs (publish / subscribe policies).

Supported sensors

One of the most important features of our neOSensors is the ability to automatically determine the connected I2C integrated circuits. This way, we avoid most of the per-sensor configuration and we let the device itself discover them :)

Current list of available drivers:

IC ref	Type	Auto	Note
MCP9808	Temperature	✅	
MAX44009	Luminosity	✅	Great sensor that gives human perceived light
TSL2561	Luminosity	✅	
MCP47FEB, MCP47X6	DAC	✅	only for boards featuring noise detectors
SH1106	DISPLAY	✅	1.3" oled driven by the (great) U8g2 lib
PMSX003, SDS011, IKEA VINDRIKTNING	Particle Meters		**serial** sensors for PM2.5 and PM10
Sensirion SPS3X	*Particle Meters*		*[underway]* ***serial*** *sensors for PM2.5 and PM10*
MHZ1X	CO2		**serial** (optical) sensor
Sensirion SCD4x	T/RH/CO2	✅	Combined (fantastic) digital sensor calibrated
SHT2X, SHT3X	T/RH	✅	Temperature and humidity sensors. Beware of uncalibrated chinese clones :(
TM1637	7seg display		only used by the neOClock devices class
LCC's VOC sensors	various VOC		Research project from LCC laboratory

Toolchains

In order to download code to your neOSensor, you first need to set up the arduino toolchain along with support for ESP8266, ESP32, CubeCell, STM32 ...

- Arduino toolchain (from arduino website)

```
install Arduino toolchain
```

Arduino 1.8.19 at the time of writing.

According to the neOSensor version you own, it'll be best to follow the installation steps described at https://github.com/fthiebolt/neOCampus-arduino

- ESP8266 cross compiler

```
Launch Arduino → preferences → additional boards manager
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

Select ESP8266_Arduino version ~~***2.7.4***~~ ***2.7.1***

- ESP32 cross compiler

```
Launch Arduino → preferences → additional boards manager
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

Select ESP32_Arduino version ***1.0.6***

Then launch script:

```
./deploy.sh
```

Note: ensure you're located within the root of the git repository you've just downloaded.

Check for additional steps specified at https://github.com/fthiebolt/neOCampus-arduino#readme

neOSensor firmware compilation

At this point, your desktop/laptop feature the following:

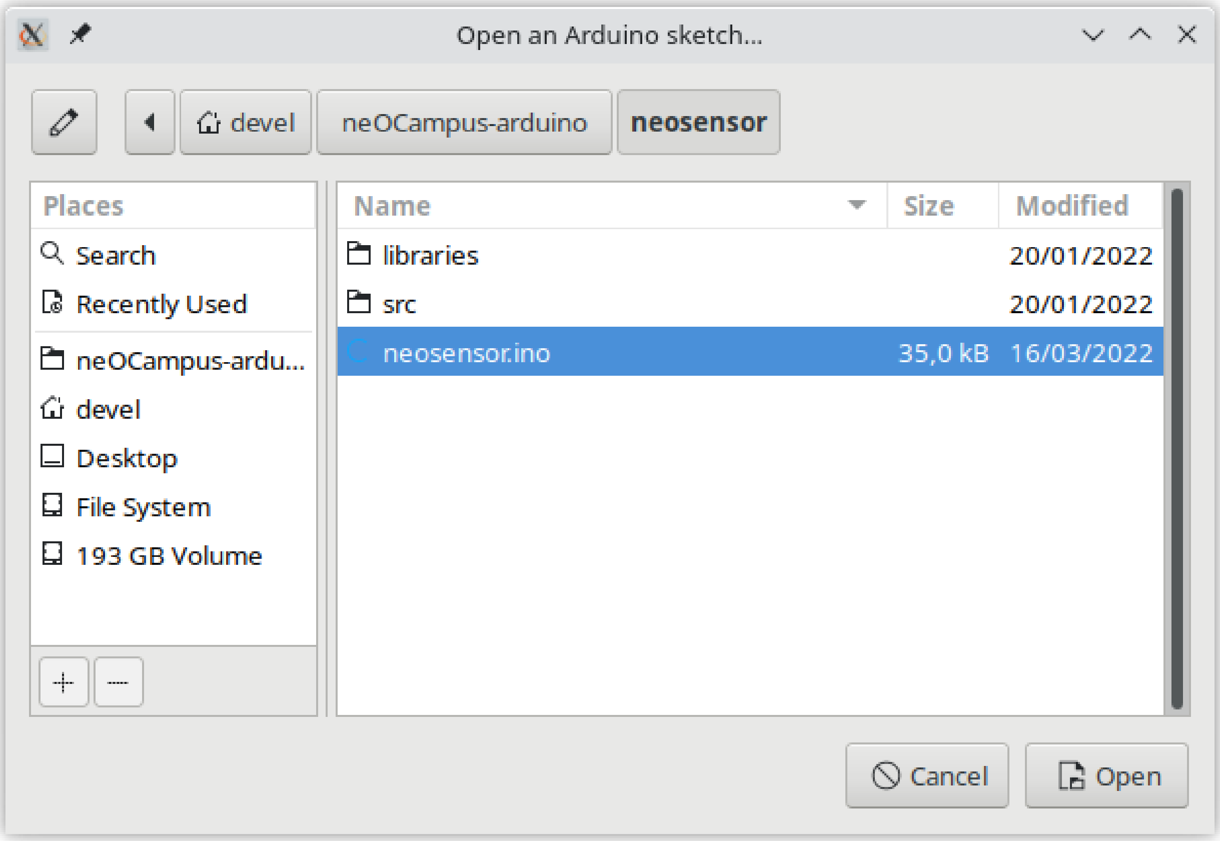
- arduino IDE installed (>= 1.8.19)
- python3 pyserial installed (used by arduino tools)
- ESP8266/ESP32 cross-compilers
- already defined path to your arduino libraries (i.e the neOSensor libs)
- deploy (bash) script launched

[git] neOSensor repository

```
git clone git@github.com:fthiebolt/neOCampus-arduino.git
```

open neOSensor.ino

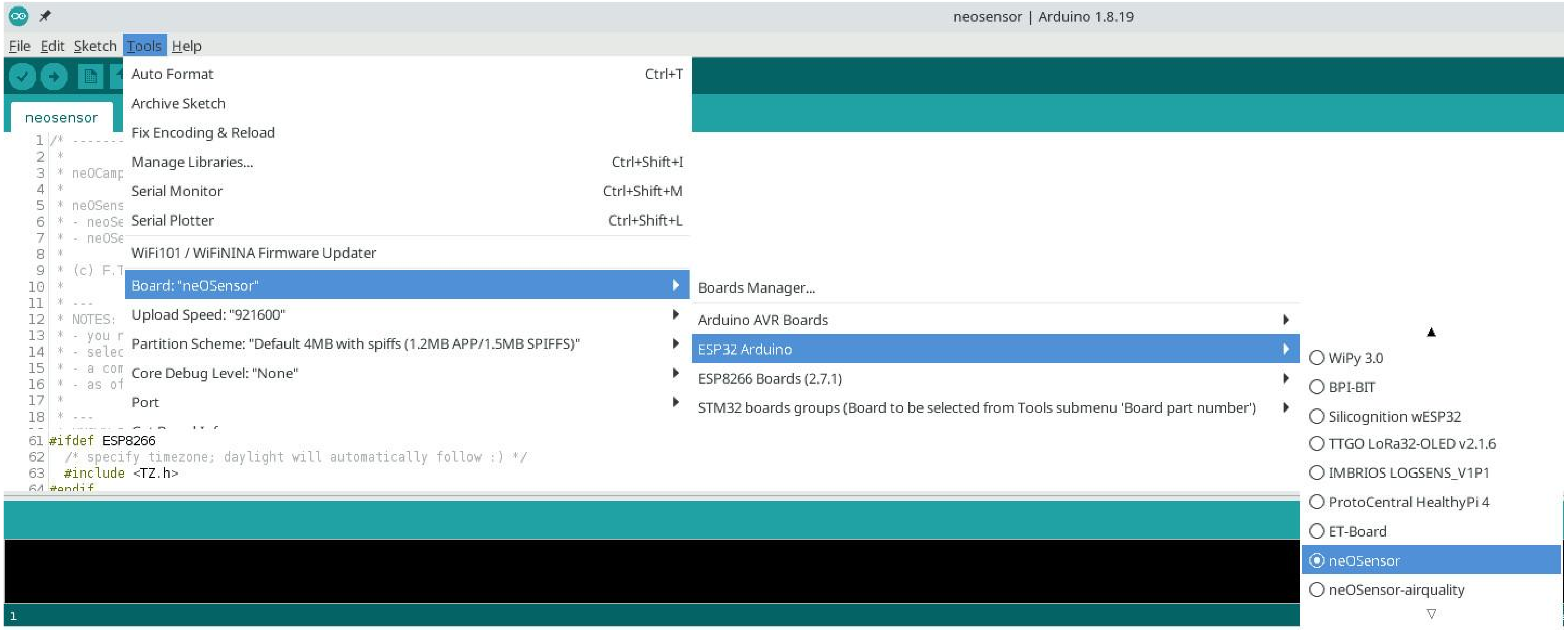
Select the '`neosensor.ino`' from `<path>/neOCampus-arduino/neosensor`



Name	Size	Modified
libraries		20/01/2022
src		20/01/2022
neosensor.ino	35,0 kB	16/03/2022

select proper hardware

In this example, we'll select the ESP32 version of neOSensor



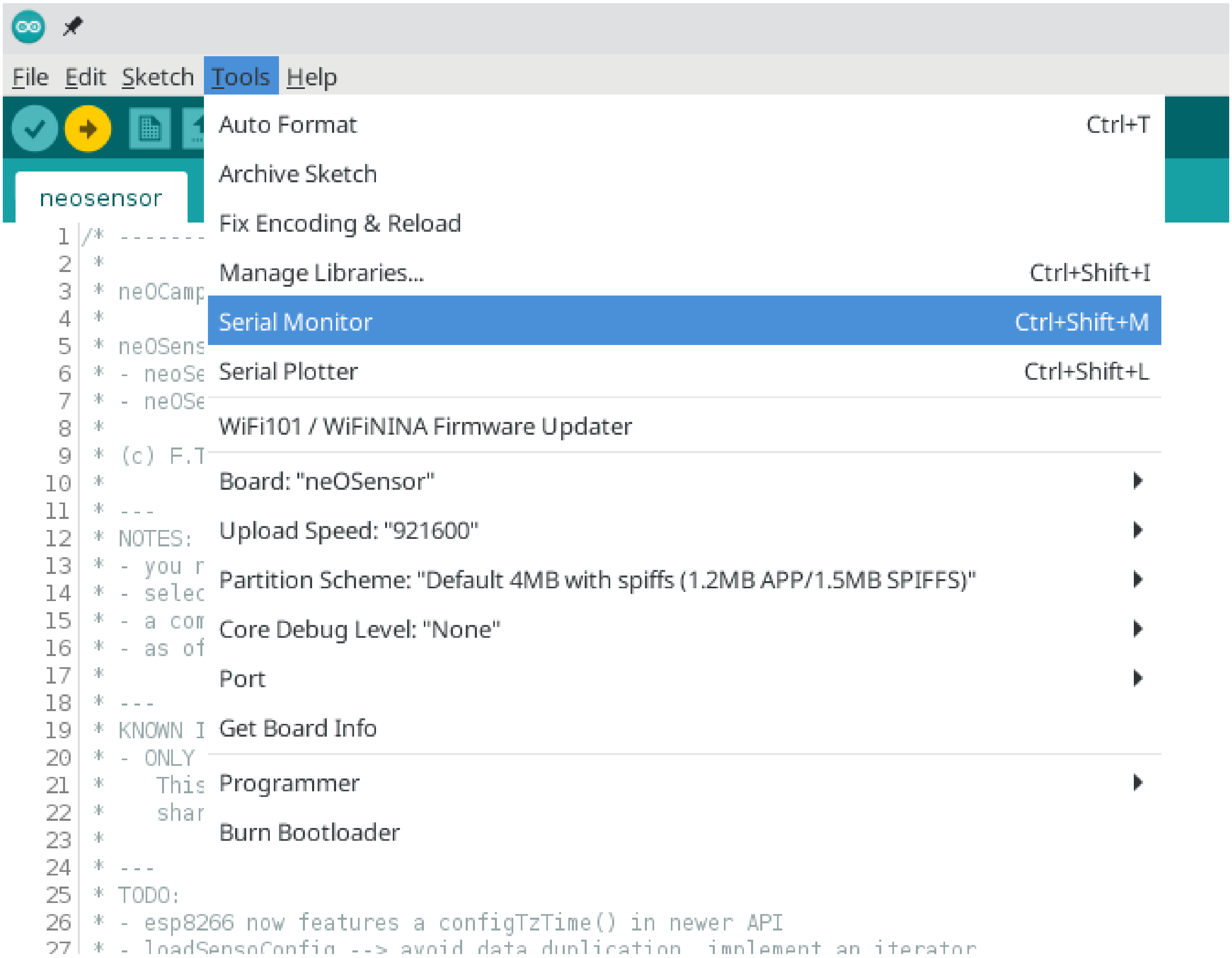
Board: "neOSensor"
Upload Speed: "921600"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"
[✓ neOSensor]

```
/* -------
 *
 * neOCamp
 *
 * neOSens
 * - neoSe
 * - neOSe
 *
 * (c) F.T
 *
 * ---
 * NOTES:
 * - you r
 * - selec
 * - a com
 * - as of
 *
 * ---
#ifdef ESP8266
  /* specify timezone; daylight will automatically follow :) */
  #include <TZ.h>
#endif
```

Note: the screenshot has been cropped to fit this page ;)

Upload and monitor

Now you just click on *'Upload'* and then select '*Tools → Serial Monitor*'



Board: "neOSensor"
Upload Speed: "921600"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "None"

```
/* -------
 *
 * neOCamp
 *
 * neOSens
 * - neoSe
 * - neOSe
 *
 * (c) F.T
 *
 * ---
 * NOTES:
 * - you r
 * - selec
 * - a com
 * - as of
 *
 * ---
 * KNOWN I
 * - ONLY
 *     This
 *     shar
 *
 * ---
 * TODO:
 * - esp8266 now features a configTzTime() in newer API
 * - loadSensoConfig --> avoid data duplication, implement an iterator
```

[optional] libraries update

All of our used libs are located within '`<project_dir>/neosensor/libraries`' … hence the reason why you OUGHT to set Arduino workdir to '`<project_dir>/neosensor`' in Arduino preferences.txt

Almost all of them are manageable through the '`Tools > Manage libraries`'

[dev. branch] WiFimanager

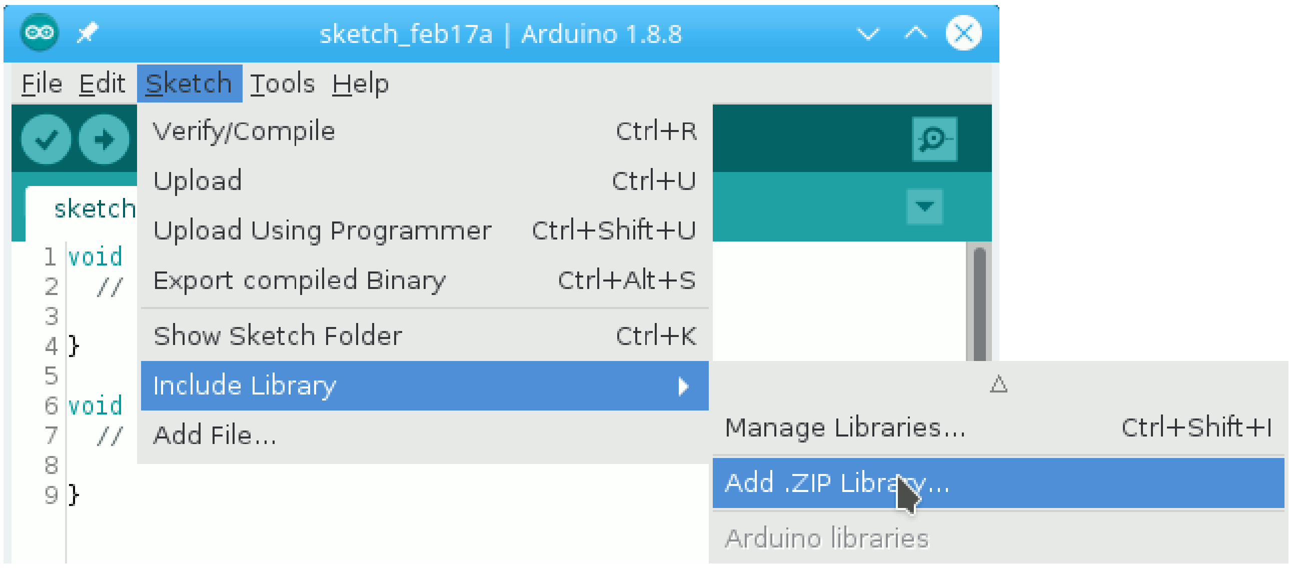
This one requires special care since it is both the dev. branch along with custom mods …

From URL

```
https://github.com/tzapu/WiFiManager/tree/development
```

… select download ZIP and save files somewhere (NOT in our project!)

From Arduino IDE '`Sketch > Include library > add .ZIP Library`'


sketch_feb17a | Arduino 1.8.8
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
sketch
void
//
}
void
//
}

… then you choose the previously saved 'WiFiManager-development.zip' file.

[custom mods] WiFimanager

- `neosensor/libraries/WiFiManager/WiFiManager.h`

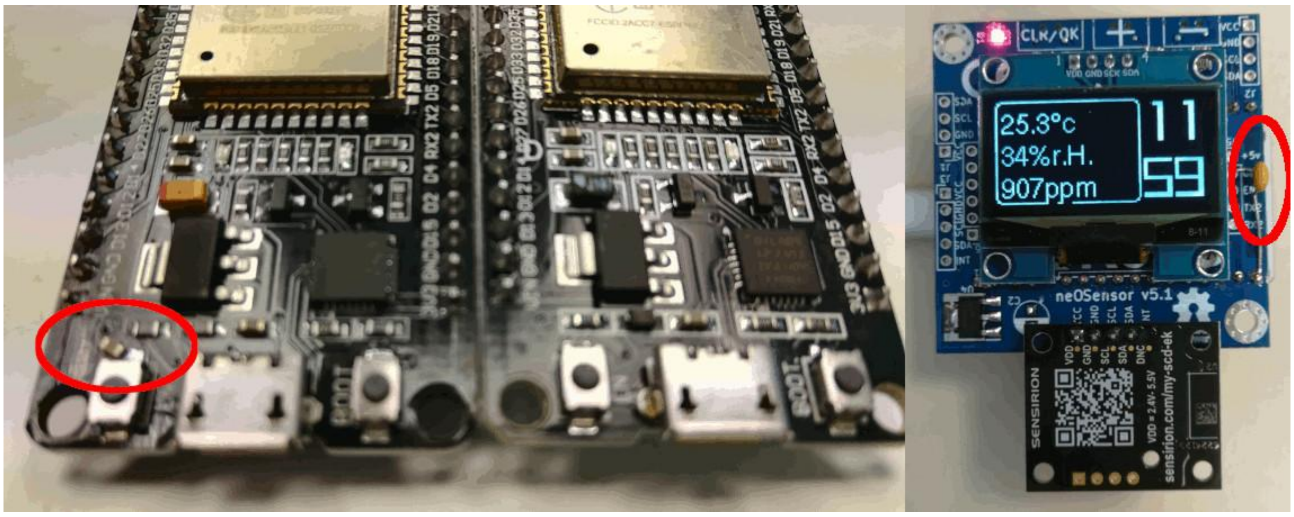
```
…………
class WiFiManagerParameter {
  public:
    …………
    // [feb.19] Francois: added as a public attr
    const char *_customHTML;
    …………

  private:
    …………
    // [feb.19] Francois: moved as public a attr
    // const char *_customHTML;
    …………
};
```

Known issues

We summarise all currently known issues.

esp32 devkit missing EN capacitor



Our colleagues from the Tr@nsNet project bought some ESP32 devkitV1. Unfortunately, these esp32 boards were missing a small cap that prevented them from booting ! ⇒ add 0805 cap footprint on PCB to enable RC circuit

i2c multi-sensors uniqueness

I2C sensors like SHT3x and SCD4x are shared across multiple modules due to the fact that they host several sensors onboard (e.g SHT31 feature both temp and hygro).

The problem arises for an i2c end-device shared across multiple modules (e.g 'temperature' and 'humidity' modules regarding SHT3x). SHT3x exhibits cached values as static variables i.e shared across all instances … even those featuring different i2c addresses :|
While it sounds reasonable for one i2c sensor, things turn to a dead-end when you have, let's say, 2 x SHT31 featuring two different I2C addresses … the same static vars are shared across ALL instances.

One solution could be the SHT3x and SCD4x drivers to implement cached values sensitive to the i2c addr (an array!)

multiple MQTT clients → max_sockets

Each of the neOSensors modules (i.e temperature, hygro, luminosity, noise etc etc), all of them feature a MQTT client. This situation has been overcome by extending the maximum number of available sockets in ESP8266 (8). ESP32 already features 16 TCP sockets.

esp software suites to upgrade

Due to the deprecation of the SPIFFS filesystem, we're still stuck to the following software suites:

Devices	Current	Latest	Notes
esp8266	**2.7.1**	3.0.2	We noticed that 2.7.4 has some WiFi limitations the 2.7.1 does not suffer from.
esp32	**1.0.6**	2.0.2	

password protected AP

Some Android smartphones do not redirect to the http://192.168.4.1 captive portal used for neOSensor configuration if an AP password has been enabled.

powering issues

neOSensor >= V5 enables you to get directly connected to your PC or any 5v PSU.
However, we discovered that USB ports on some Dell PCs (and probably others) exhibit a sub 5v USB power leading to issues with the neOSensor operations.

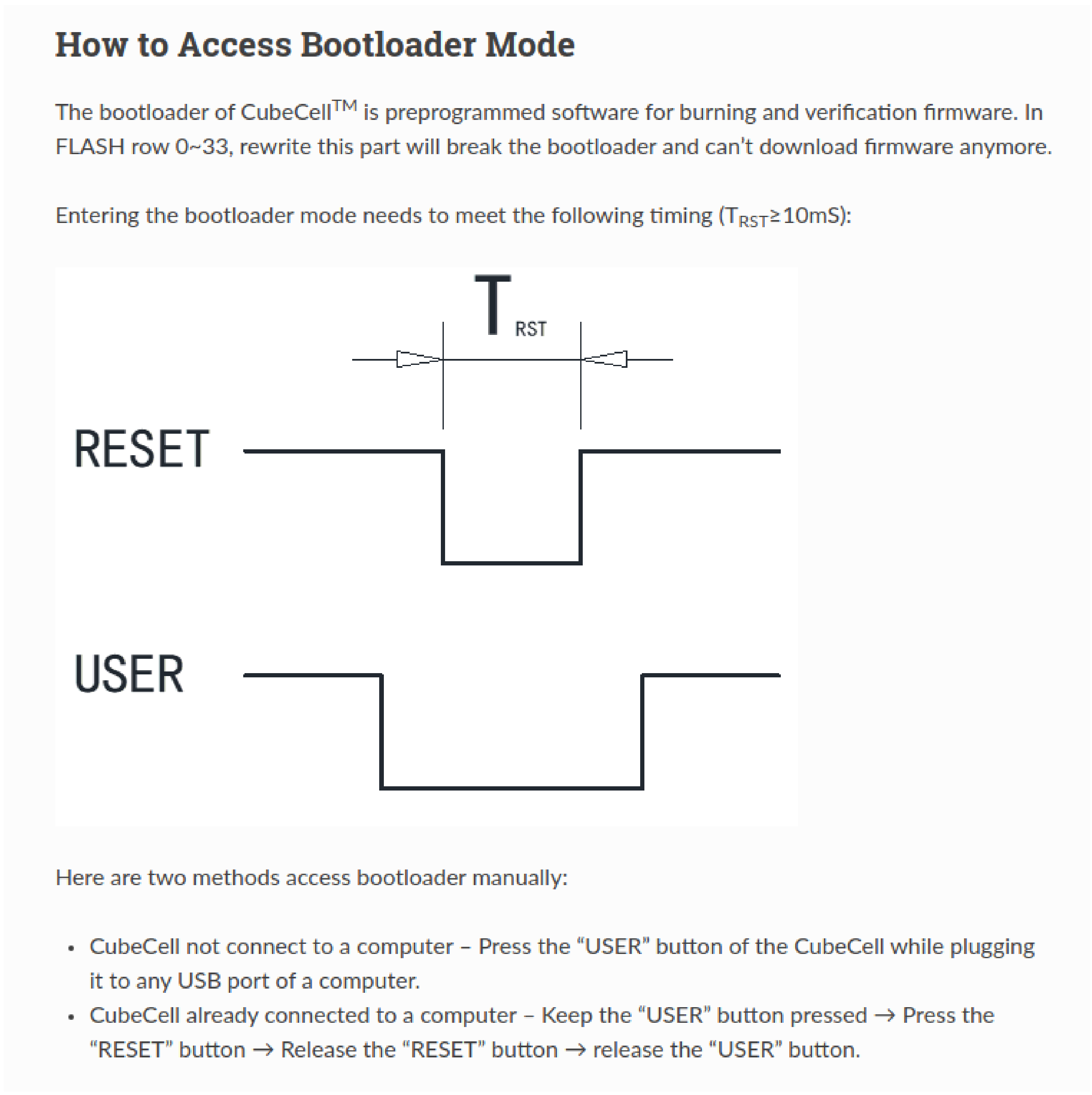
We either try to use very high quality (and short) USB cables … or just get powered via a Raspberry Pi power supply: this latter exhibits a **5.1v** output, perfect for our device :)

>>> to guarantee proper operation of your neOSensor, make use of a RPi power supply <<<

[jun.21][LoRaWAN] CubeCell boot mode

Since our failing neOSensor LoRaWAN pcb V2, we discovered how to enable the bootloader mode:

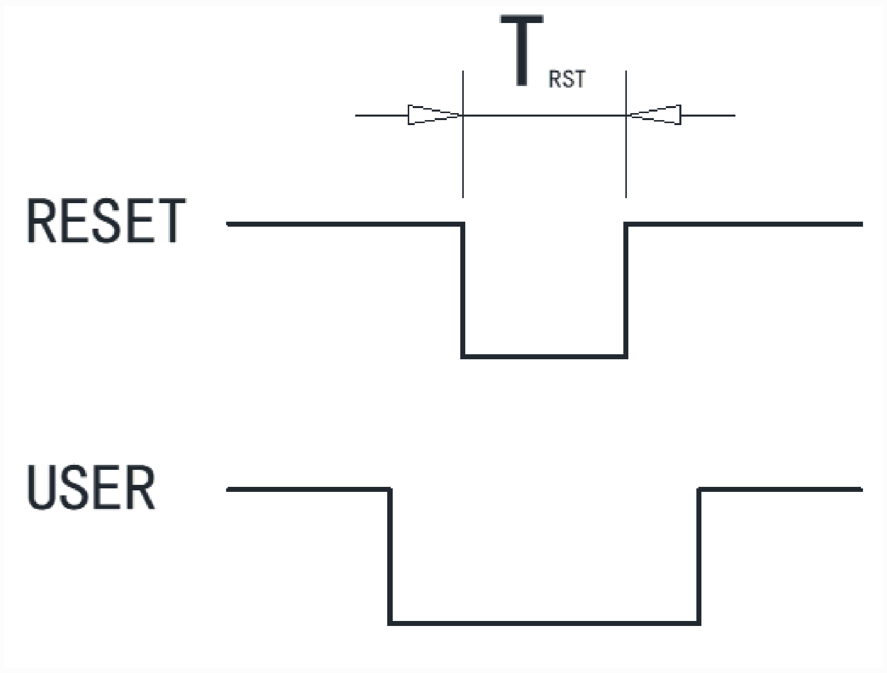
- **press USER button while pressing RST button** or powering up the board



How to Access Bootloader Mode

The bootloader of CubeCell^{TM} is preprogrammed software for burning and verification firmware. In FLASH row 0~33, rewrite this part will break the bootloader and can't download firmware anymore.

Entering the bootloader mode needs to meet the following timing (T_{RST}≥10mS):



Here are two methods access bootloader manually:

- CubeCell not connect to a computer – Press the "USER" button of the CubeCell while plugging it to any USB port of a computer.
- CubeCell already connected to a computer – Keep the "USER" button pressed → Press the "RESET" button → Release the "RESET" button → release the "USER" button.

- or **connect GPIO0 to GND** and power the module !!

[sep.20] esp8266 WiFi reduced range 2.7.4

We discover that new firmwares for ESP8266 based on SDK 2.7.4 exhibit a reduced WiFi connectivity as neOSensors deployed with BU tour floor 8 were unable to get in touch with the neOCampus infrastructure.

UNLESS OTHERWISE STATED … new firmwares intended to esp8266 neOSensors will make use of **SDK rev. 2.7.1**

[nov.19] AP mode password

This new release features a password to gain access to the configuration mode of our neOSensor.

(1) WiFi scan → look for neOSensor_xxxx AP

(2) Connect to WiFi device → captive portal → enter AP passwd

```
neOSensor
```

Note: in case captive portal does not come alone … open a browser with http://192.168.4.1

(3) configuration mode
as usual ...

[admin] FW upgrades deployment

Whenever you make improvements to your code, you'd like to push firmware upgrades to your devices.

neOSensor's firmware embeds a unique way to OTA upgrade (i.e Over The Air) based on current firmware comparison with those hosted on the neOCampus server.

>>> [oct.20] repeat the whole procedure for neOSensor (esp8266 & esp32) and AirQuality (esp32). Boards support added in Arduino IDE :) <<<

Retrieve code

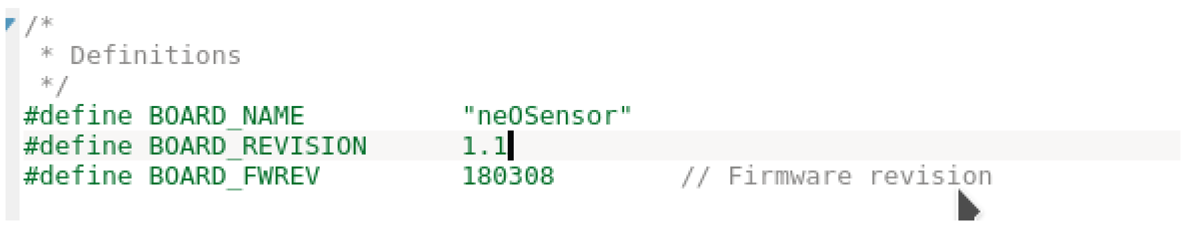
```
git clone https://github.com/fthiebolt/neOCampus-arduino
```

Generate new firmware

>>> Don't forget to select neOSensor board in the board setup <<<

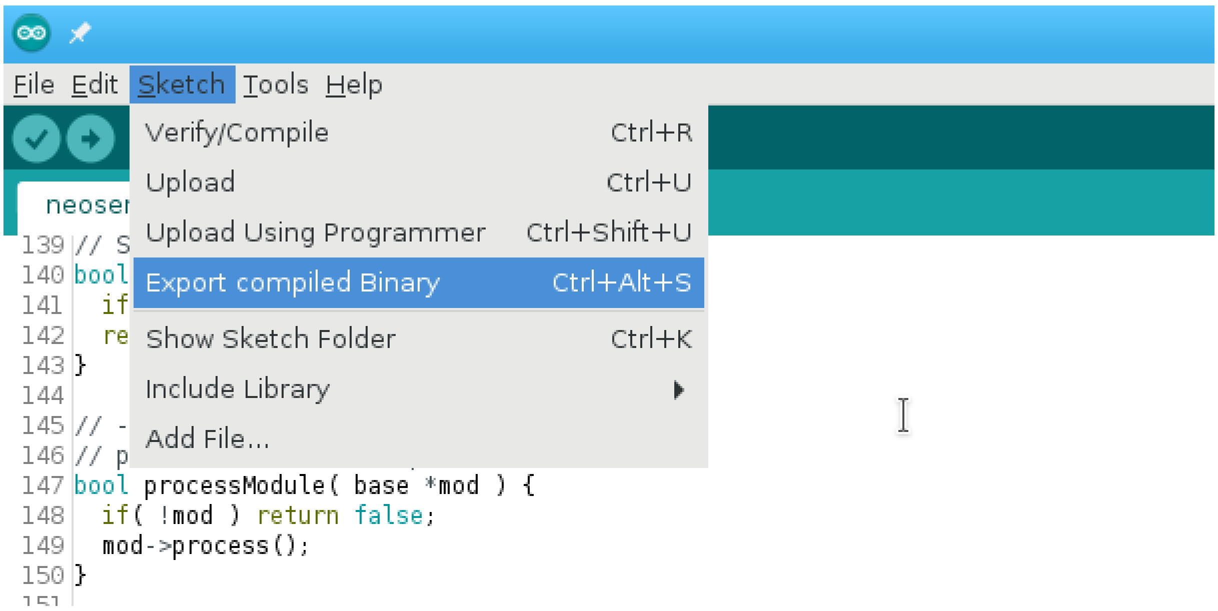
First, **update** your code's **firmware ver. → BOARD_FWREV**

- `<neOSensor_repo>/neosensor/libraries/boards/neOSensor.h`



```
/*
 * Definitions
 */
#define BOARD_NAME              "neOSensor"
#define BOARD_REVISION          1.1
#define BOARD_FWREV             180308          // Firmware revision
```

… then generate new binaries


File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
neoser

```
// S
bool
  if
  re
}

// -
// p
bool processModule( base *mod ) {
  if( !mod ) return false;
  mod->process();
}
```

that will generate files

```
<neOSensor_repo>/neosensor/neosensor.ino.generic.bin
```

[ESP8266][neOSensor] Upload new firmware to server

This time, we'll push the newly generated firmware to the server in order the other neOSensors devices may upgrade :)

- `/neocampus/images/<arch>/<device name>/firmware_latest.bin`

```
scp neosensor.ino.generic.bin
root@neocampus.univ-tlse3.fr:/nfs/images/esp8266/neOSensor/neosensor_220328.bin
```

Note: firmware files are named against their release number

now connect as **root** to neocampus server to modify JSON file describing firmware

```
ssh root@neocampus
cd /nfs/images/esp8266/neOSensor
rm -f firmware_latest.bin
ln -s neosensor_220328.bin firmware_latest.bin
```

- `/nfs/images/esp8266/neOSensor/default.json`

```
{
  "description": "esp8266 based neOSensor",
  "revision": 220328,
  "release_date": "2022-03-28",
  "image":  "http://neocampus.univ-tlse3.fr/images/esp8266/neOSensor/firmware_latest.bin"
}
```

… Now it's okay, other devices will get upgraded upon their next reboot :)

[ESP32][neOSensor] Upload new firmware to server

This time, we'll push the newly generated firmware to the server in order the other neOSensors devices may upgrade :)

- `/neocampus/images/<arch>/<device name>/firmware_latest.bin`

```
scp neosensor.ino.node32s.bin
root@neocampus.univ-tlse3.fr:/nfs/images/esp32/neOSensor/neosensor_220328.bin
```

Note: firmware files are named against their release number

now connect as **root** to neocampus server to modify JSON file describing firmware

```
ssh root@neocampus
cd /nfs/images/esp32/neOSensor
rm -f firmware_latest.bin
ln -s neosensor_220328.bin firmware_latest.bin
```

- /nfs/images/esp32/neOSensor/default.json

```
{
  "description": "esp32 based neOSensor",
  "revision": 220328,
  "release_date": "2022-03-28",
  "image":  "http://neocampus.univ-tlse3.fr/images/esp32/neOSensor/firmware_latest.bin"
}
```

… Now it's okay, other devices will get upgraded upon their next reboot :)

[ESP32][AirQuality] Upload new firmware to server

This time, we'll push the newly generated firmware to the server in order the other **AirQuality** devices may upgrade :)

>>> note: yes, it has the same filename but has been compiled with airquality flags <<<

- /neocampus/images/<arch>/<device name>/firmware_latest.bin

```
scp neosensor.ino.node32s.bin
root@neocampus.univ-tlse3.fr:/nfs/images/esp32/AirQuality/neosensor_201102.bin
```

Note: firmware files are named against their release number

now connect as **root** to neocampus server to modify JSON file describing firmware

```
ssh root@neocampus
cd /nfs/images/esp32/AirQuality
rm -f firmware_latest.bin
ln -s neosensor_201102.bin firmware_latest.bin
```

- /nfs/images/esp32/AirQuality/default.json

```
{
  "description": "esp32 based AirQuality",
  "revision": 201102,
  "release_date": "2020-11-02",
  "image":  "http://neocampus.univ-tlse3.fr/images/esp32/AirQuality/firmware_latest.bin"
}
```

… Now it's okay, other devices will get upgraded upon their next reboot :)

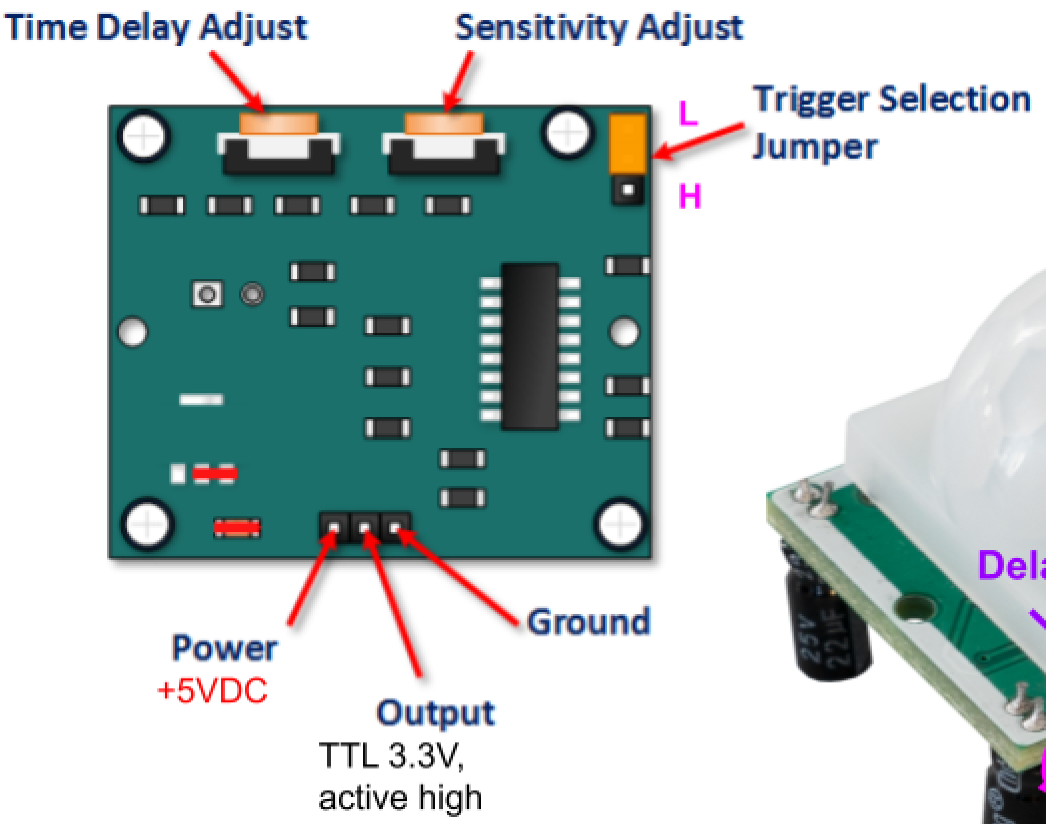
Annexe - A | Hardware addendum

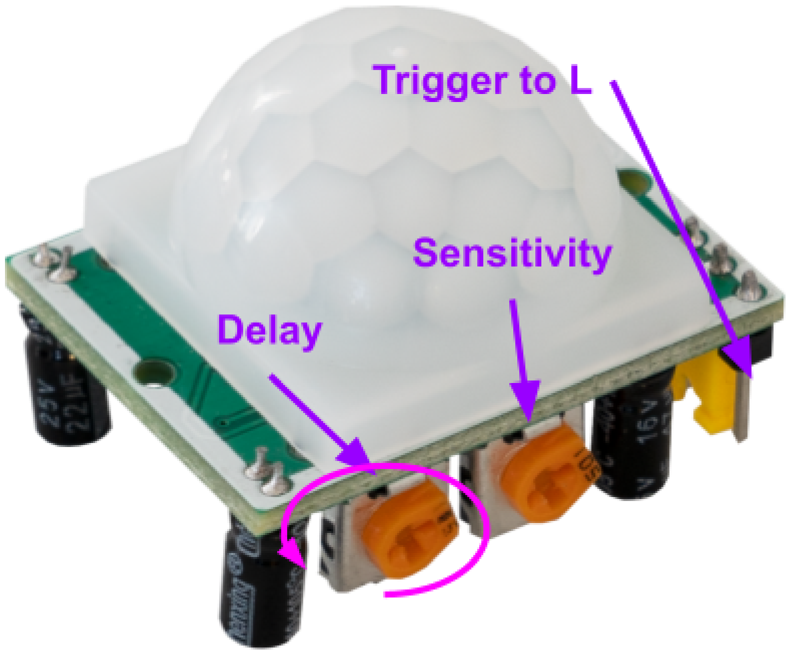
PCB assembling

Below are a few notes about soldering devices on our PCBs.

[aug.21] PIR sensor @ neOSensorV5 | HC-SR501

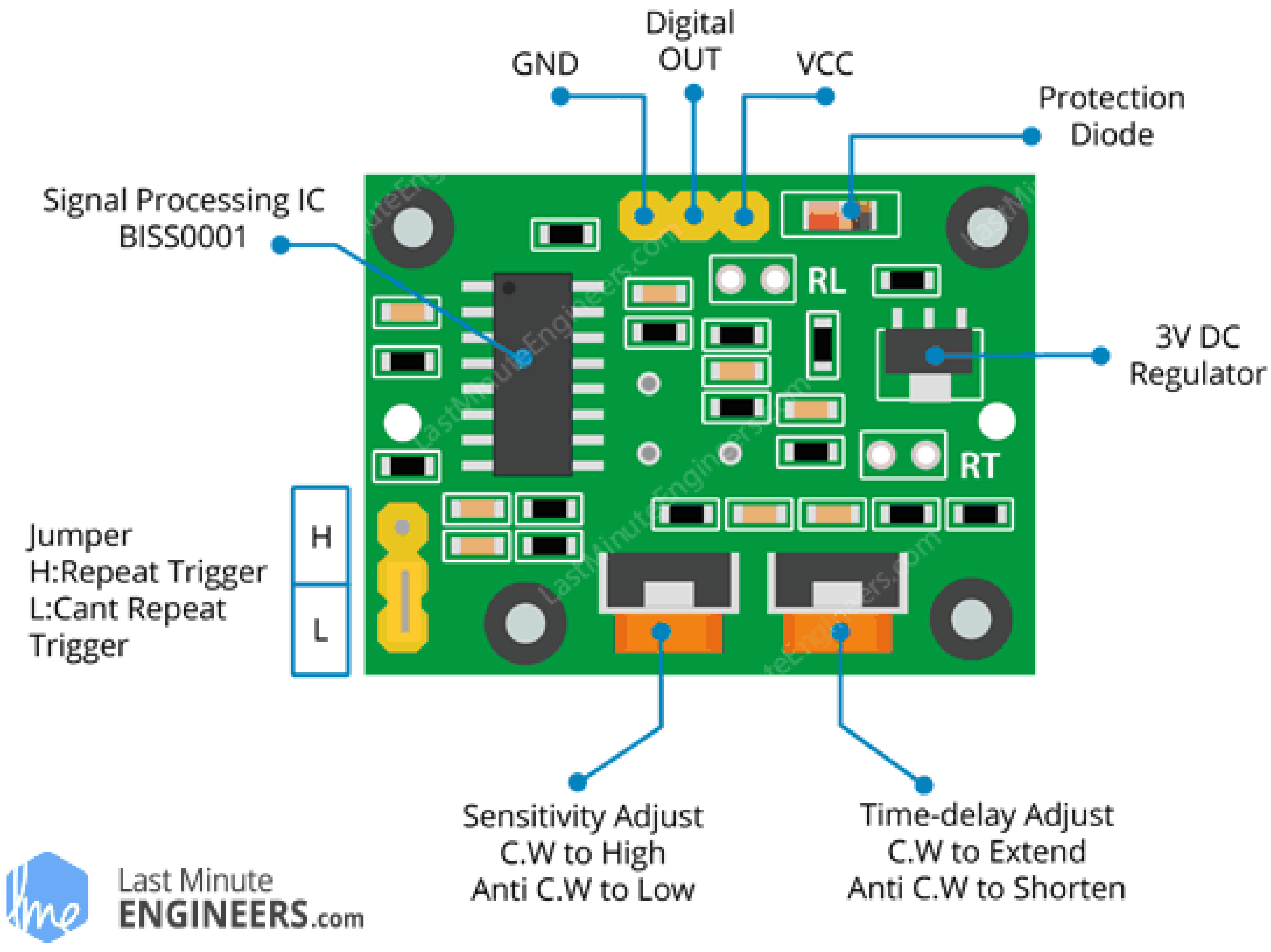
Only applies to neOSensor >= V5 boards
https://lastminuteengineers.com/pir-sensor-arduino-tutorial/





PIR sensor settings

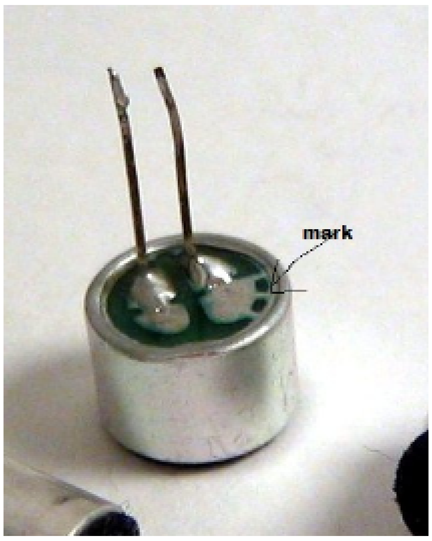
Set the yellow jumper to LOW (to the PCB's corner).
Set Delay anti C.W to the minimum (i.e pulse = 3s)
Set sensitivity according to the place



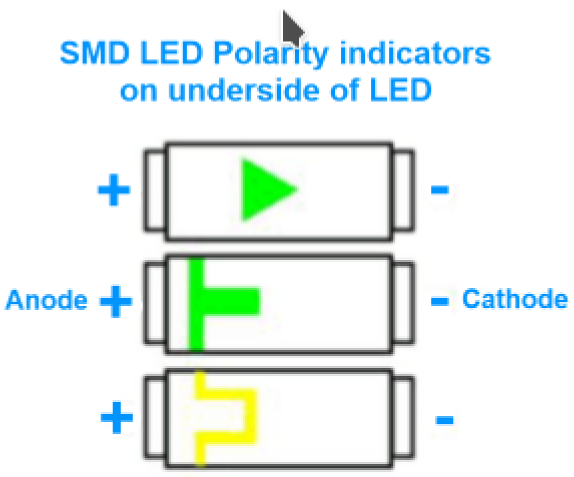
Electret microphone

Only applies to neOSensor < v5

electret micro has a gnd pin, connect it to the proper GND pin on the PCB



smd 0805 led polarity



WARNING: some 0805 leds feature different symbols on top and bottom … top symbol is for cathode !!

[neOSensorV4] PCB issues

- still exhibits power issues even with 2200 ohm RF-filter → set 1 x dedicated power supply based on AMS1117-33 + 1 x 100mA 3v3 power supply for all of the others components
- move microphone away from IR sensor
- temperature sensor board, replace C2 with 1µF 1210 format
- U3 (DAC) footprint pins are too long
- C6 too close to pcb border
- U1 (AOP) pins ought to be a bit longer
- R11 replace 330ohm → 470ohm
- near C14, useless *via* on L2 pad!
- mark '-' on microphone footprint

Panelization note: set µUSB connector on the edges of the PCB (to benefit from the PCB manufacturer's ultra precise cut!)

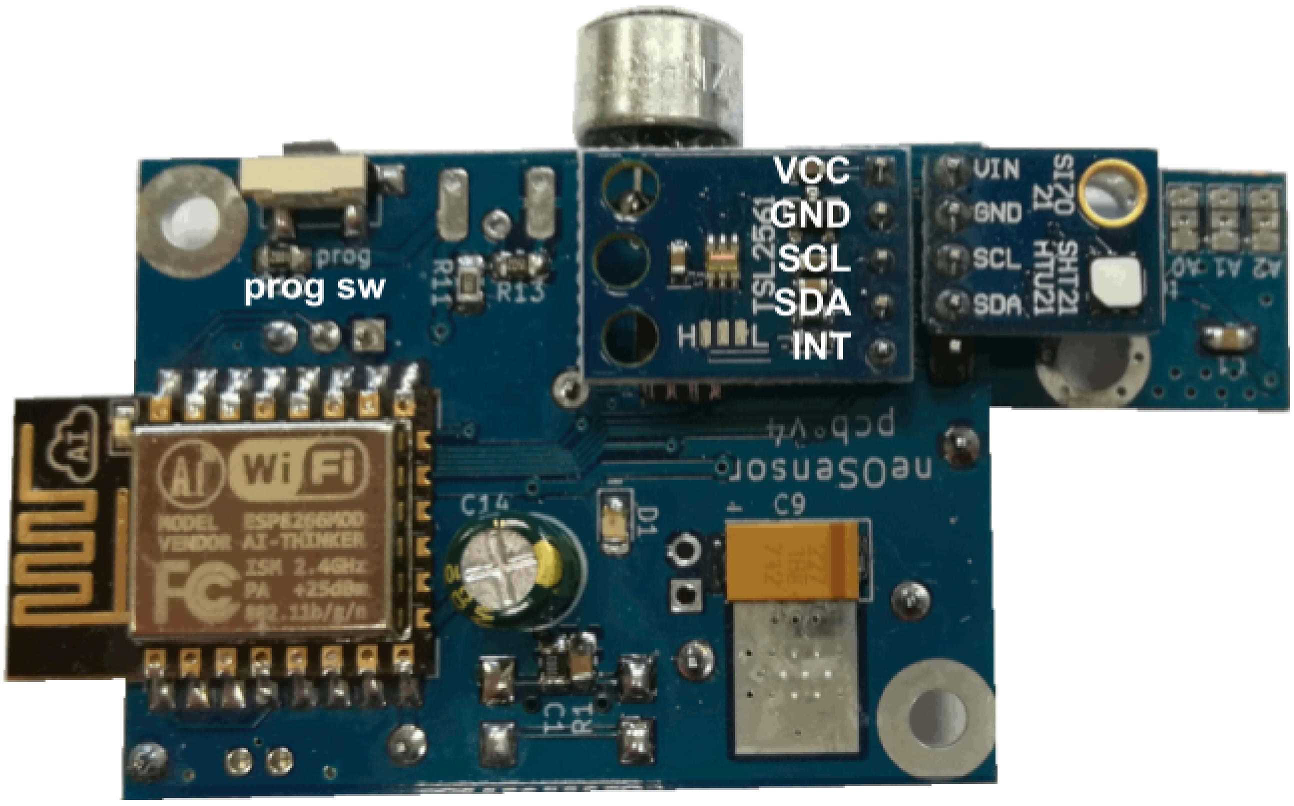
[neOSensorV4] HW test | led_blink

We now present below our boards featuring:

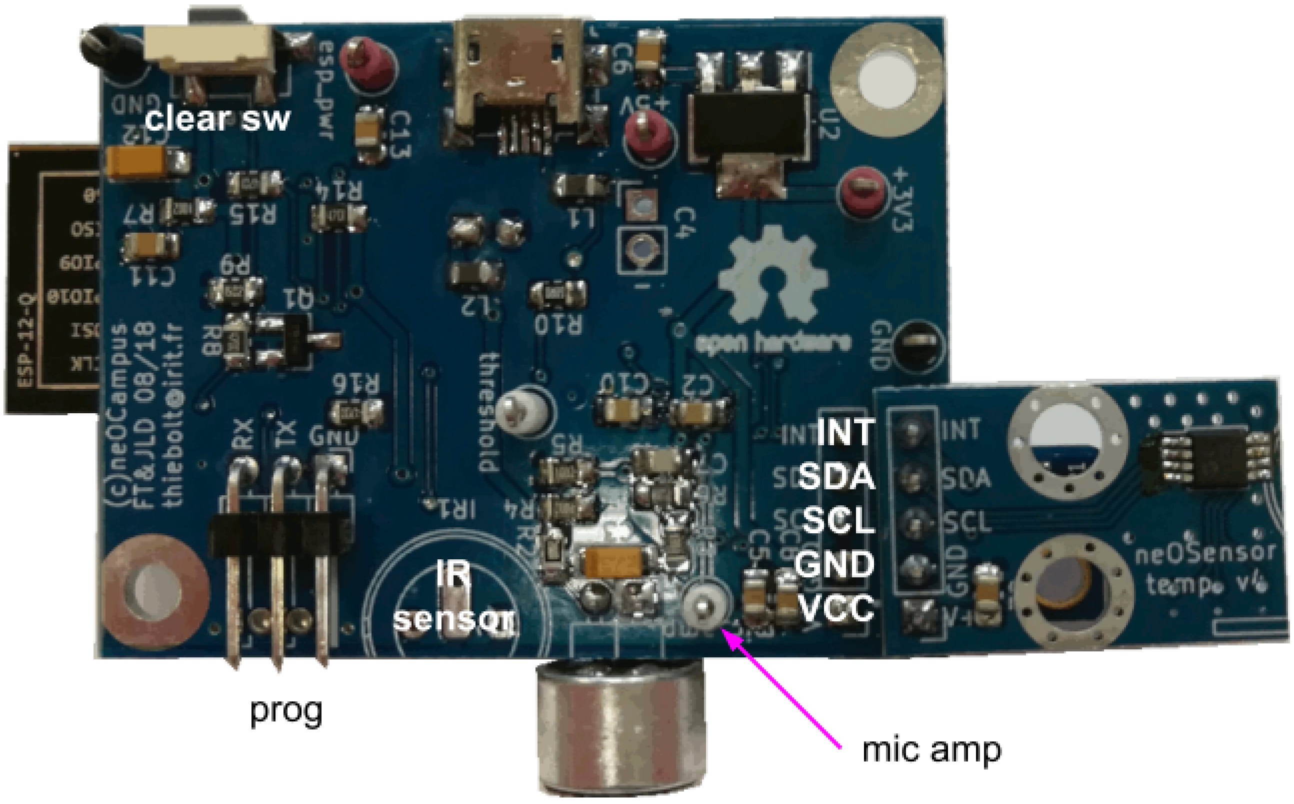
- noise detector with DAC (ESP8266 version only)
- MCP9808 temperature sensor
- available i2c pins with power supply for additional sensors (e.g TSL2561)

First of all, let's see what our PCB looks like:

- neOSensor PCBv4 TOP view



- neOSensor PCBv4 BOTTOM view



- MCP9808 temperature sensor based addon board PCB v4

(Note: board is self-describing with all pins explained)

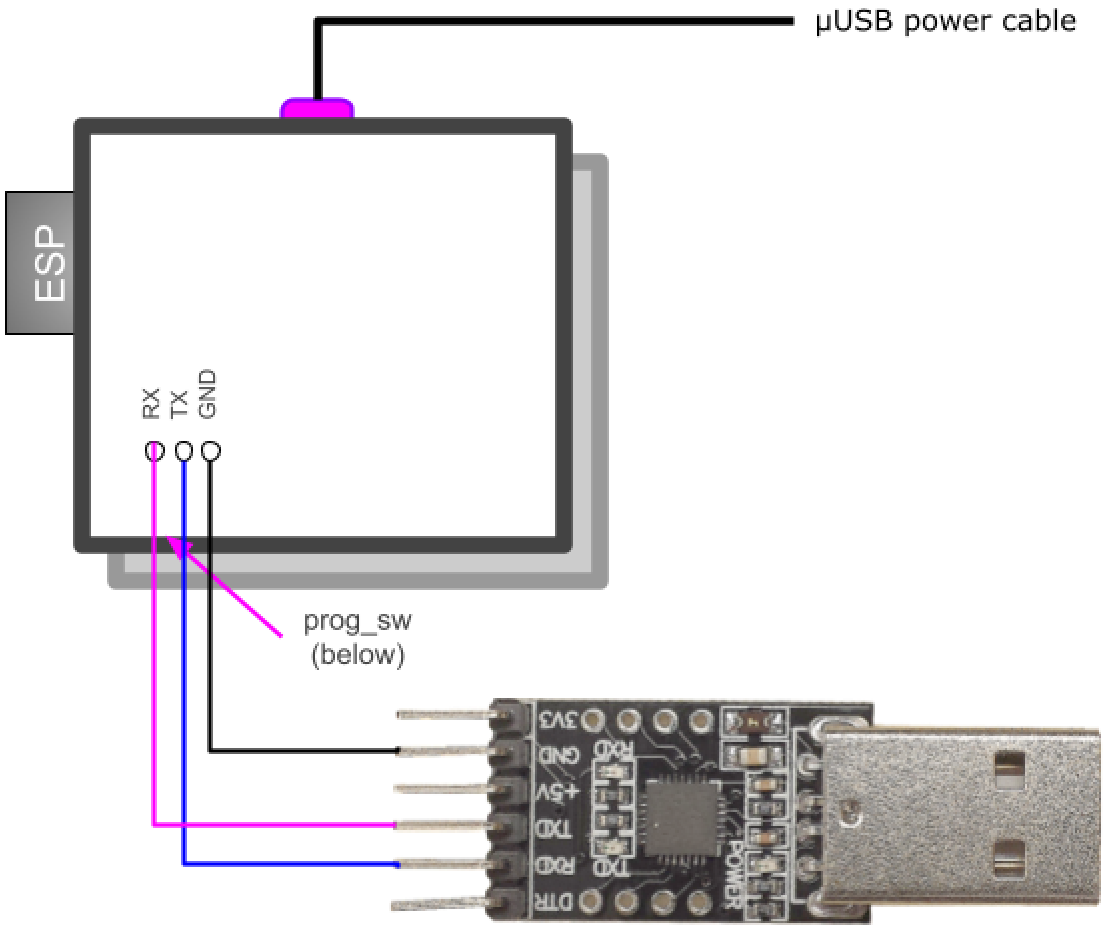
Retrieve code

```
git clone https://github.com/fthiebolt/neOCampus-arduino
```

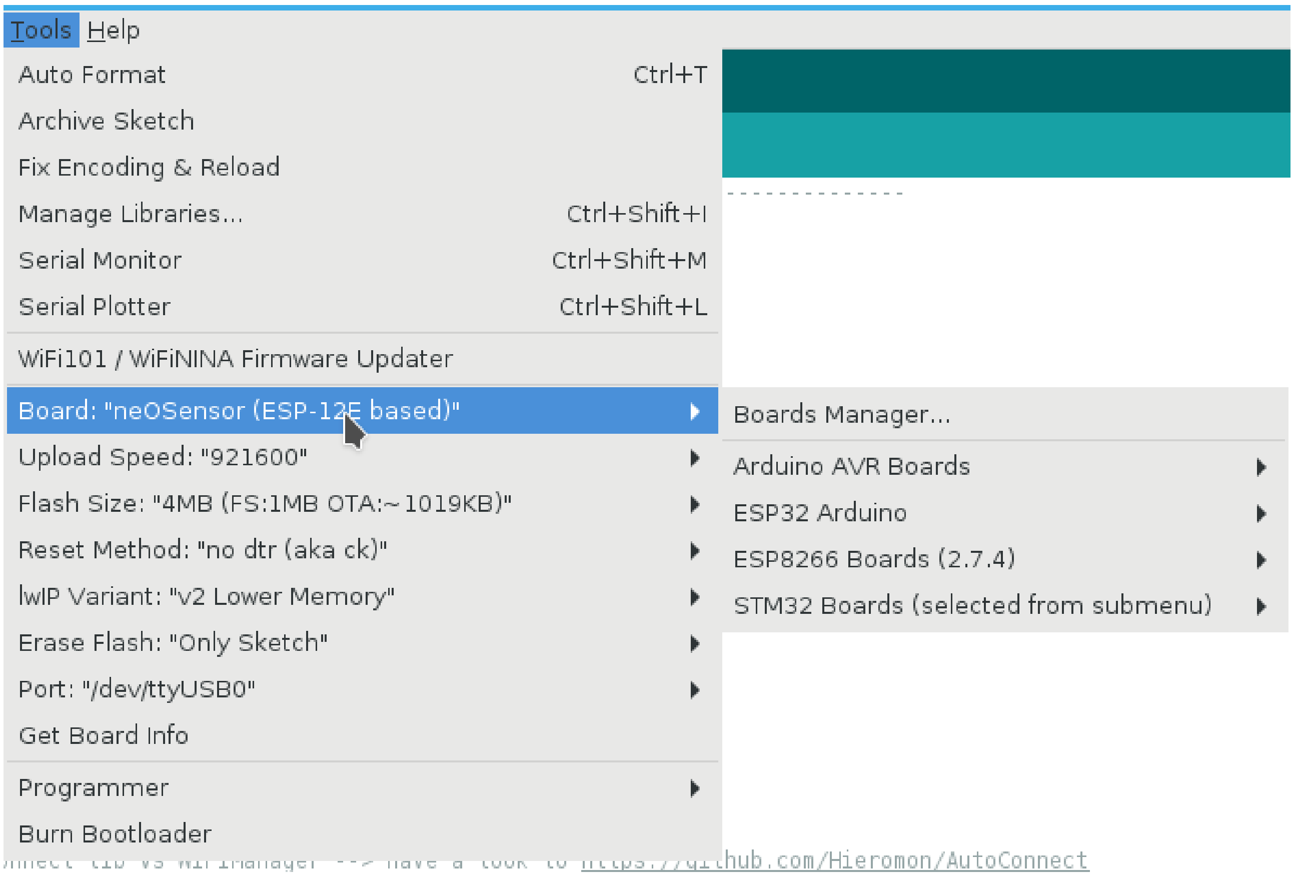
Activate programming mode

To activate programming mode, you need to press the 'prog_sw' while powering up your device

Note: it is very important to use an external power supply as those from USB-serial converters will not power your device properly.



- Set Arduino board as "neOSensor" from ESP8266 sub modules:



Tools Help

Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "neOSensor (ESP-12E based)"
Upload Speed: "921600"
Flash Size: "4MB (FS:1MB OTA:~1019KB)"
Reset Method: "no dtr (aka ck)"
lwIP Variant: "v2 Lower Memory"
Erase Flash: "Only Sketch"
Port: "/dev/ttyUSB0"
Get Board Info
Programmer
Burn Bootloader

Boards Manager...
Arduino AVR Boards
ESP32 Arduino
ESP8266 Boards (2.7.4)
STM32 Boards (selected from submenu)

Additionally, pay attention to the following parameters:

- Port = `/dev/ttyUSB0`
- Speed = 115200
- Flash Size = 4M (1M SPIFFS) (now as default since sep.20)

>>> ERASE FLASH <<<
when upgrading to a newer core library, a lot of things change and mapping will be described in a different way ⇒ select **All Flash Contents**.

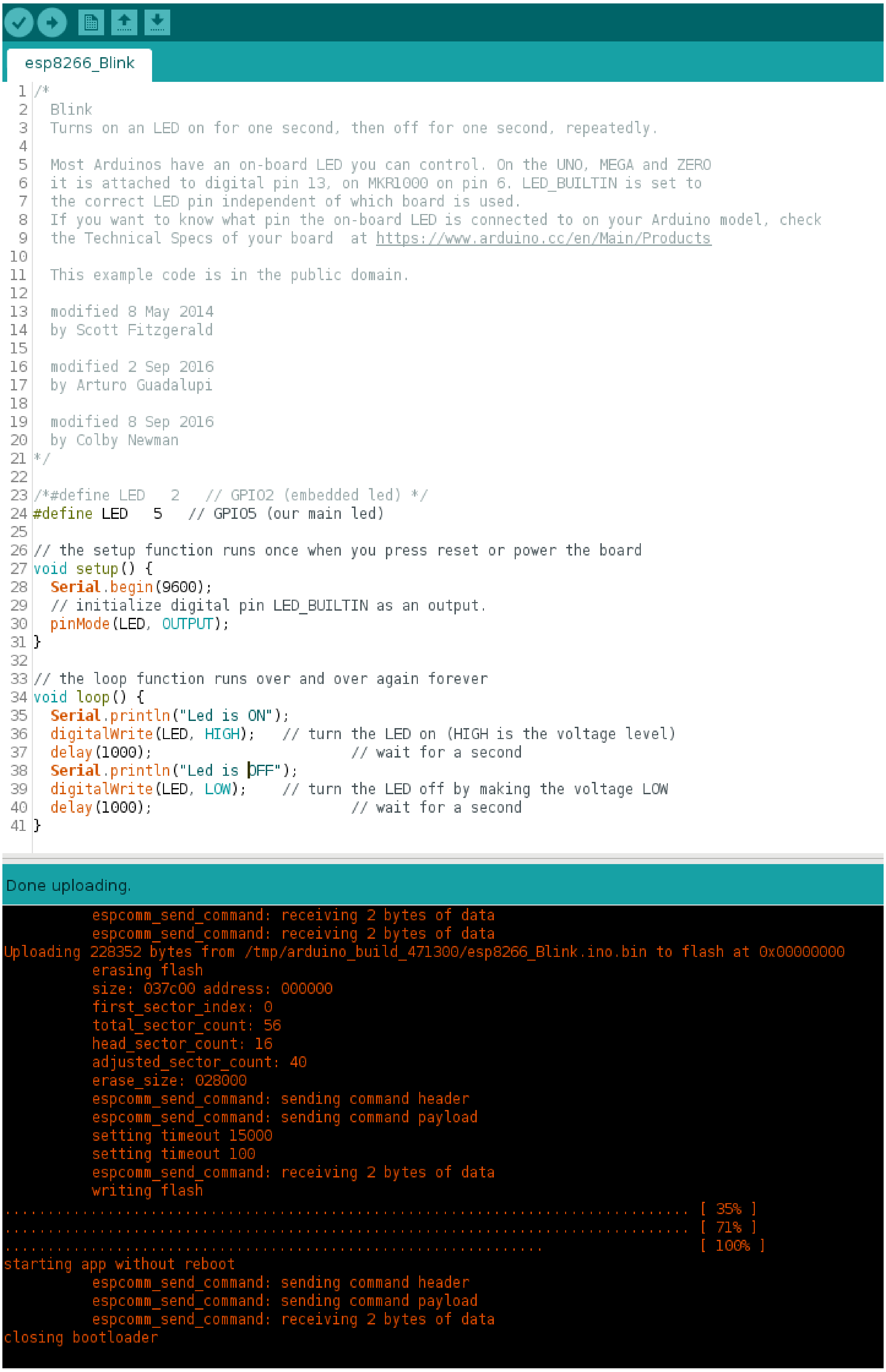
- Set Arduino default sketchbook

Launch Arduino → preferences → sketchbook location
*"<path>/neOCampus-arduino/**neosensor**"*

*This will enable the arduino toolchain to automatically find the **libraries** we're using.*

- open `blink` sketch in your Arduino IDE

This is the default one provided with the Arduino toolchain



```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino model, check
  the Technical Specs of your board  at https://www.arduino.cc/en/Main/Products

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi

  modified 8 Sep 2016
  by Colby Newman
*/

/*#define LED   2   // GPIO2 (embedded led) */
#define LED   5   // GPIO5 (our main led)

// the setup function runs once when you press reset or power the board
void setup() {
  Serial.begin(9600);
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  Serial.println("Led is ON");
  digitalWrite(LED, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  Serial.println("Led is OFF");
  digitalWrite(LED, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

Done uploading.

```
          espcomm_send_command: receiving 2 bytes of data
          espcomm_send_command: receiving 2 bytes of data
Uploading 228352 bytes from /tmp/arduino_build_471300/esp8266_Blink.ino.bin to flash at 0x00000000
          erasing flash
          size: 037c00 address: 000000
          first_sector_index: 0
          total_sector_count: 56
          head_sector_count: 16
          adjusted_sector_count: 40
          erase_size: 028000
          espcomm_send_command: sending command header
          espcomm_send_command: sending command payload
          setting timeout 15000
          setting timeout 100
          espcomm_send_command: receiving 2 bytes of data
          writing flash
.............................................................................. [ 35% ]
.............................................................................. [ 71% ]
..............................................................                 [ 100% ]
starting app without reboot
          espcomm_send_command: sending command header
          espcomm_send_command: sending command payload
          espcomm_send_command: receiving 2 bytes of data
closing bootloader
```

… then just click the '*upload*' icon and it will compile and upload your application …
… and LED ought to blink brightly :)

Reaching this step means that most of the embedded sensor is working :-)

… the next step is full application deployment!

monitoring serial line

```
miniterm.py /dev/ttyUSB0 115200
```

Annexe -B | Older releases & deprecation

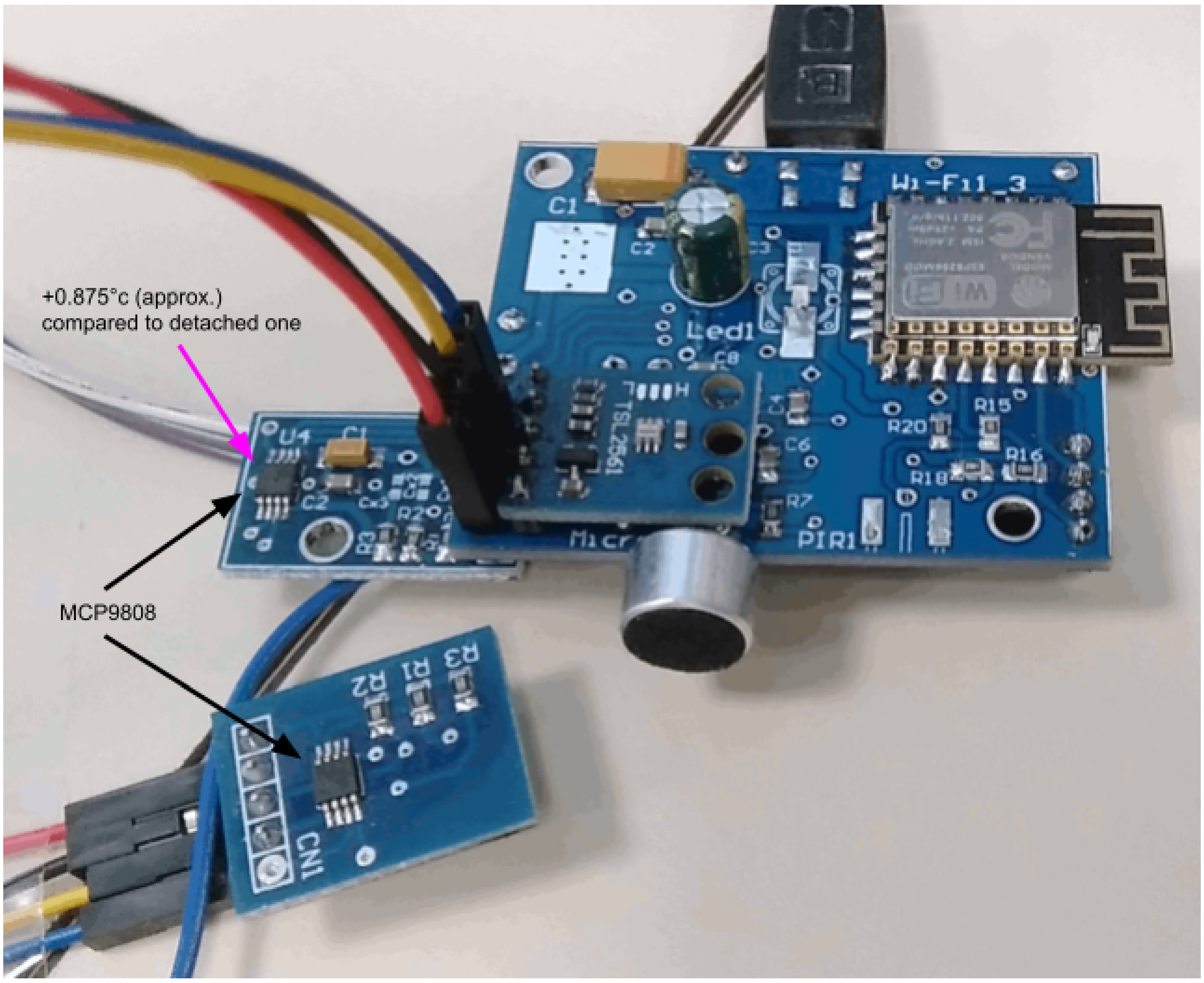
Temperature sensor | correction warning

See [Mar.18] Temperature correction **DEPRECATION**
When you add a temperature sensor tied to the main PCB, it OUGHT to feature an i2c addr that ought to be the last of its range, example:
MCP9808 i2c addr ranges from 0x18 to 0x1F, thus the sensor attached to the main PCB ought to feature CX1,2 and 3 opened.

[Mar.18] Temperature correction

>>> DEPRECATED: NOT enabled as default since sep.21 <<<

We found that temperature sensors (mainly MCP9808) whose PCB is tied to the main PCB exhibit a value #1°c above the value measured from the same sensor detached:



Thus we decided to compensate by software the measurement shift that was observed for a long time. Hence, we took as a convention that the temperature **sensor whose i2c addr is the last**

one will be the sensor tied to the main PCB, its **measures will be software compensated**.

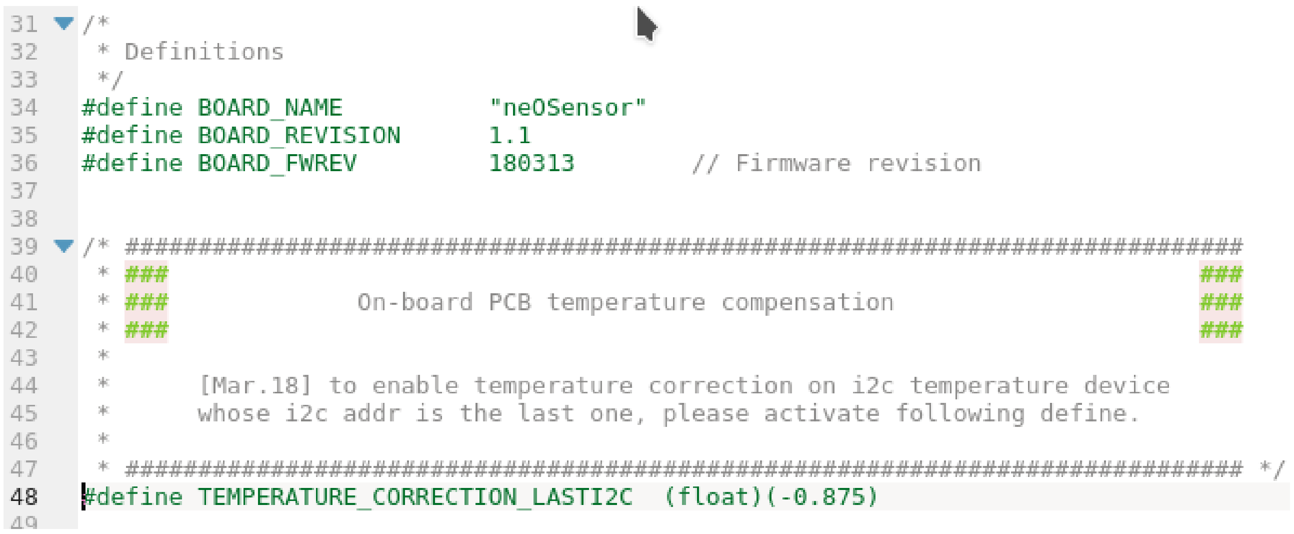
```
[base] MQTT msg = {"value":22,"value_units":"°c","subID":26,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":22.875,"value_units":"°c","subID":31,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)..................
[base] MQTT msg = {"value":67,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)..............................
[base] MQTT msg = {"value":66,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)..........
[base] MQTT msg = {"value":21.875,"value_units":"°c","subID":26,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":22.875,"value_units":"°c","subID":31,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)....................
[base] MQTT msg = {"value":66,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)..............................
[base] MQTT msg = {"value":66,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :).........
[base] MQTT msg = {"value":22,"value_units":"°c","subID":26,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":22.875,"value_units":"°c","subID":31,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :).....................
[base] MQTT msg = {"value":66,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)..............................
[base] MQTT msg = {"value":66,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :).......
[base] MQTT msg = {"value":21.875,"value_units":"°c","subID":26,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":23,"value_units":"°c","subID":31,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :).......................
[base] MQTT msg = {"value":66,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)..............................
[base] MQTT msg = {"value":66,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :).....
[base] MQTT msg = {"value":22,"value_units":"°c","subID":26,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":23.125,"value_units":"°c","subID":31,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :).........................
[base] MQTT msg = {"value":66,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)..............................
[base] MQTT msg = {"value":68,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)....
[base] MQTT msg = {"value":22,"value_units":"°c","subID":26,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":23.25,"value_units":"°c","subID":31,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)..........................
[base] MQTT msg = {"value":67,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)..............................
[base] MQTT msg = {"value":64,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
```

```
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)..
[base] MQTT msg = {"value":22,"value_units":"°c","subID":26,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":23.25,"value_units":"°c","subID":31,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)...........................
[base] MQTT msg = {"value":63,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :).............................
[base] MQTT msg = {"value":65,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :).
[base] MQTT msg = {"value":22,"value_units":"°c","subID":26,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":23.375,"value_units":"°c","subID":31,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)............................
[base] MQTT msg = {"value":63,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :)............................
[base] MQTT msg = {"value":22,"value_units":"°c","subID":26,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":23.5,"value_units":"°c","subID":31,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/temperature
[temperature] successfully published msg :)
[base] MQTT msg = {"value":62,"value_units":"lux","subID":57,"unitID":"auto_55a9"} -->
TestTopic/neOSensor_55a9/luminosity
[luminosity] successfully published msg :).............
```

activation

To activate temperature compensation of on-board i2c temperature sensor:

- `<neOSensor>/neosensor/libraries/boards/neOSensor.h`



```
/*
 * Definitions
 */
#define BOARD_NAME          "neOSensor"
#define BOARD_REVISION      1.1
#define BOARD_FWREV         180313        // Firmware revision


/* ##############################################################################
 * ###                                                                        ###
 * ###              On-board PCB temperature compensation                     ###
 * ###                                                                        ###
 *
 *       [Mar.18] to enable temperature correction on i2c temperature device
 *       whose i2c addr is the last one, please activate following define.
 *
 * ############################################################################## */
#define TEMPERATURE_CORRECTION_LASTI2C   (float)(-0.875)
```