SentinAir

User Guide

Project designer and developer

[Dr. Domenico Suriano](), domenico.suriano@enea.it

This user guide version (v. 1.41) refers to the SentinAir system version 1.4 available at the Github SentinAir repository.

Description

SentinAir is a device designed and developed for data acquisition from diverse types of instruments, sensors, or devices (see figure 1). The core of the system is the *Raspberry 3 B+* board; therefore, the software presented in this guide can be installed either on a *SentinAir* device or a *Raspberry 3 B+* board. Devices have to be plugged into *SentinAir* by USB, Ethernet, serial UART, SPI, or I2C port. Currently, the system has been tested only with devices connected by the USB, I2C, Ethernet, and serial UART port available on the *Raspberry* board, which is the "brain" of *SentinAir*. The system is based on command line interfaces, except for the web pages served by the webserver installed inside.

Figure 1: on the left, *SentinAir* device. On the right, some devices currently tested by *SentinAir*.

The system automatically recognizes the devices or instruments plugged into SentinAir through their specific driver, written in Python. The drivers currently developed are listed in table 1.

Table 1: device drivers developed currently for *SentinAir* and available in the github repository.

Sensor or device	Device driver	Connection interface	Supplier or manufacturer
BME280 (atmospheric pressure, temperature, and relative humidity)	bme280.py	I2C	Bosch Sensortec
BH1750 (luxmeter)	bh1750.py	I2C	ROHM semiconductor
MCP342x on ADC Pi board (to use devices having analog output signals)	mcp342x.py	I2C	Microchip
IRC-A1 (CO_2 sensor)	irca1.py	USB	Alphasense
PMS3003 (PM sensor)	pms3003.py	TTL serial port	Plantower
PMS5003 (PM sensor)	pms5003.py	TTL serial port	Plantower
SPS30 (PM ssensor)	sps30.py	TTL serial port	Sensirion
MHZ19 (CO_2 sensor)	mhz19.py	TTL serial port	Winsen
Multisensor board (to use devices having analog output signals)	multisensor_board.py	USB	Tecnosens
106L GO3 PRO package (CO_2 and O_3 monitor)	go3.py	USB	2B technologies
405nm (NOx monitor)	nox405.py	USB	2B technologies
LCSS USB adapter (to use devices having analog output signals)	lcss_adapter.py	USB	Designed and built in our lab
CO12M (CO chemical analyzer)	co12m.py	Ethernet port	Environnement
AF22M (SO_2 chemical analyzer)	af22.py	Ethernet port	Environnement
AC32M (NO_x chemical analyzer)	ac32.py	Ethernet port	Environnement
O342M (O_3 chemical analyzer)	o342.py	Ethernet port	Environnement
VOC72M (VOC chemical analyzer)	v72m.py	Ethernet port	Environnement

SentinAir hardware

SentinAir is a modular and flexible system. The system hardware can be assembled in a minimal (see figure 2), or a typical configuration (see figure 3). The minimal hardware is composed of:

- a *Raspberry 3 B+* board
- a class 10 SD card featured by a memory capacity of at least 2 GB (4 GB recommended),.
- a 5V power source capable of providing a direct current of 3A maximum. At the moment, the system has been tested with a switching ac/dc adapter (see figure 2).

The minimal configuration allows the user to use sensors/devices/instruments featured by a USB, serial UART, I2C, or Ethernet output interface, while analog output interfaces devices must be used with one of the analog-to-digital converter boards featuring the SentinAir typical hardware configuration. Ethernet or USB devices must be connected through the Ethernet and USB sockets (see figure 2), while I2C and serial devices must be connected to the Raspberry GPIO connector following the indications of figure 3.

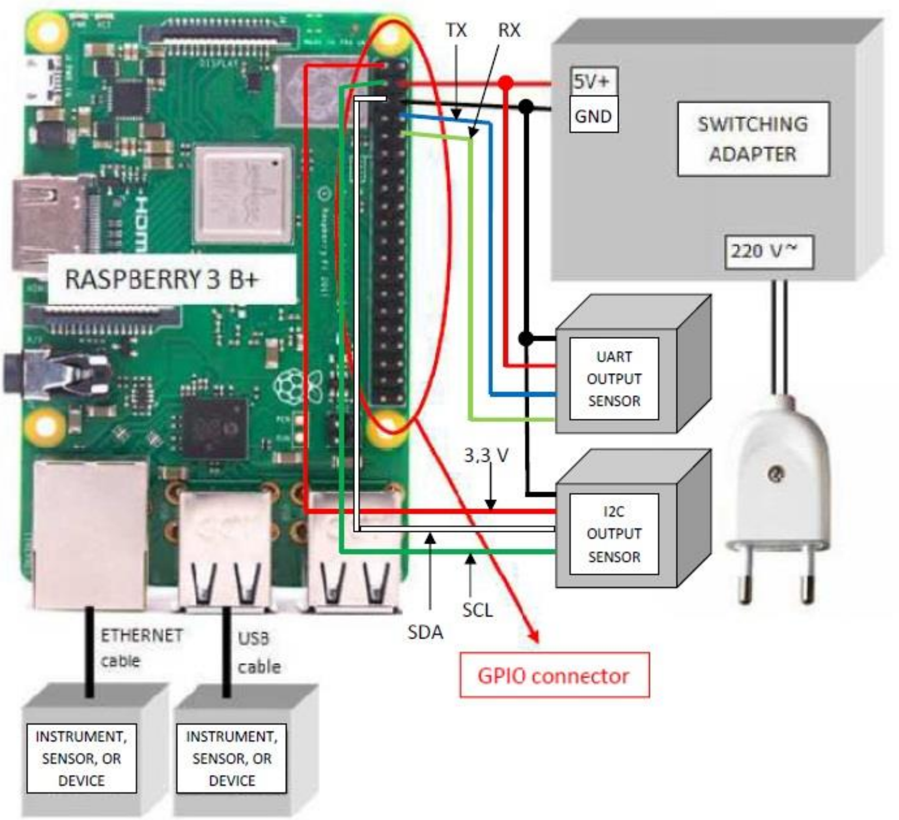

Figure 2: *SentinAir* hardware minimal configuration. Sensors/devices/instruments can be attached through the USB, Ethernet, Serial UART, or I2C port.

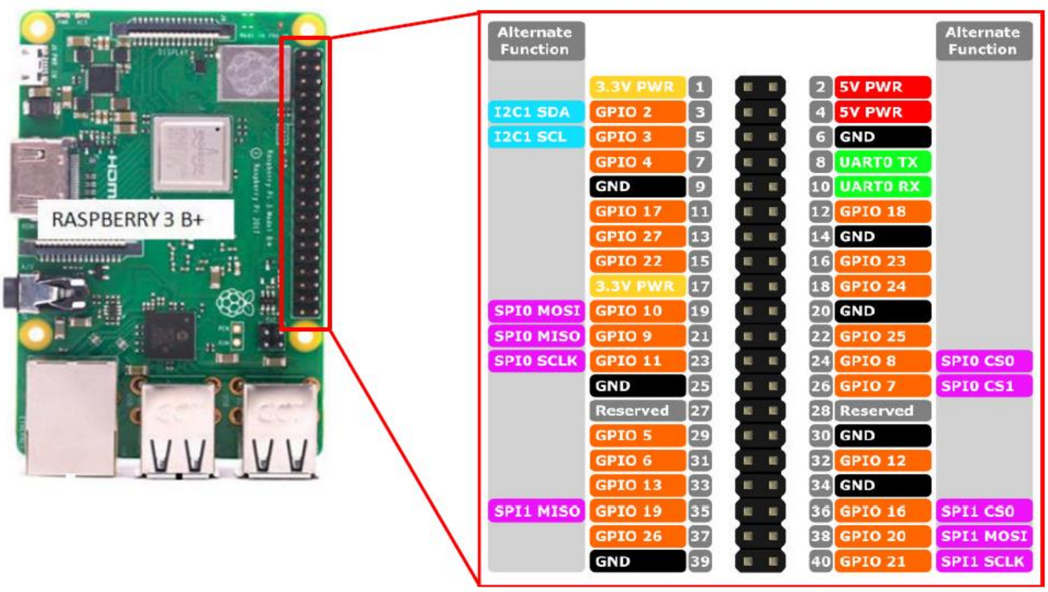

Figure 3: GPIO connector pinout. I2C devices to be used with SentinAir must connect their output to pins 3 and 5 (which is the I2C bus input). Devices having serial UART output must be connected to pins 8, 10. They can be powered through pin 2 or 4, if they need 5 Volts, otherwise, through pins 1, or 17. Device ground pins must be connected to one of the following pins: 6,9,14,20,25,30,34,39.

The typical hardware configuration of a *SentinAir* system is shown in figure 4, and it is composed of:

- A *Raspberry 3 B+* board,
- a class 10 SD card featured with a memory capacity of at least 2 GB (4GB recommended),
- The check light system made of three LEDs and three 120 Ohm resistors. They are going to indicate the three possible states of the device: standby (yellow LED off), active monitoring (yellow LED on), and fault (red LED blinking),
- A push-button to properly shut down the whole system,
- A USB stick modem (the system has been tested with a Huawei E303),
- The expansion board, which is necessary to enable the simultaneous multiple connections of sensors. This board is provided by header pin connectors through which devices can be wired for the power supply or I2C bus interfacing. Three header pin connector rows provide 5, 3,3 Volts power supply, and ground connections; in addition to them, the other two pin rows offer a quick way to connect the devices SCL and the SDA signal to the mainboard I2C bus (see figure 4).
- An analog-to-digital converter board capable of giving output data on the USB, I2C, or Ethernet port. This board is necessary to convert analog outputs of sensors into digital data. In fact, the Raspberry board, on which the SentinAir system is based, has not got a built-in analog-to-digital converter; therefore, without it, the use of sensors/devices having analog output interfaces could not be possible. The user can use for this purpose any board that requires a power of 5 V and 0,2 A maximum. For example, the ADC Pi board based on the MCP3424 microprocessor by Microchip and supplied by ABelectronics can fit the purpose. Another option is given by the LCSS adapter board (LCSS stands for Low-Cost Small commercial gas Sensors), which has been designed and developed to make the analog-to-digital conversion in an optimum way. The LCSS adapter board has been designed to minimize as much as possible the effects of the additional unwanted electronic noise, which may affect the conversion. Moreover, it has a 16-bit analog-to-digital converter that ensures a high resolution.

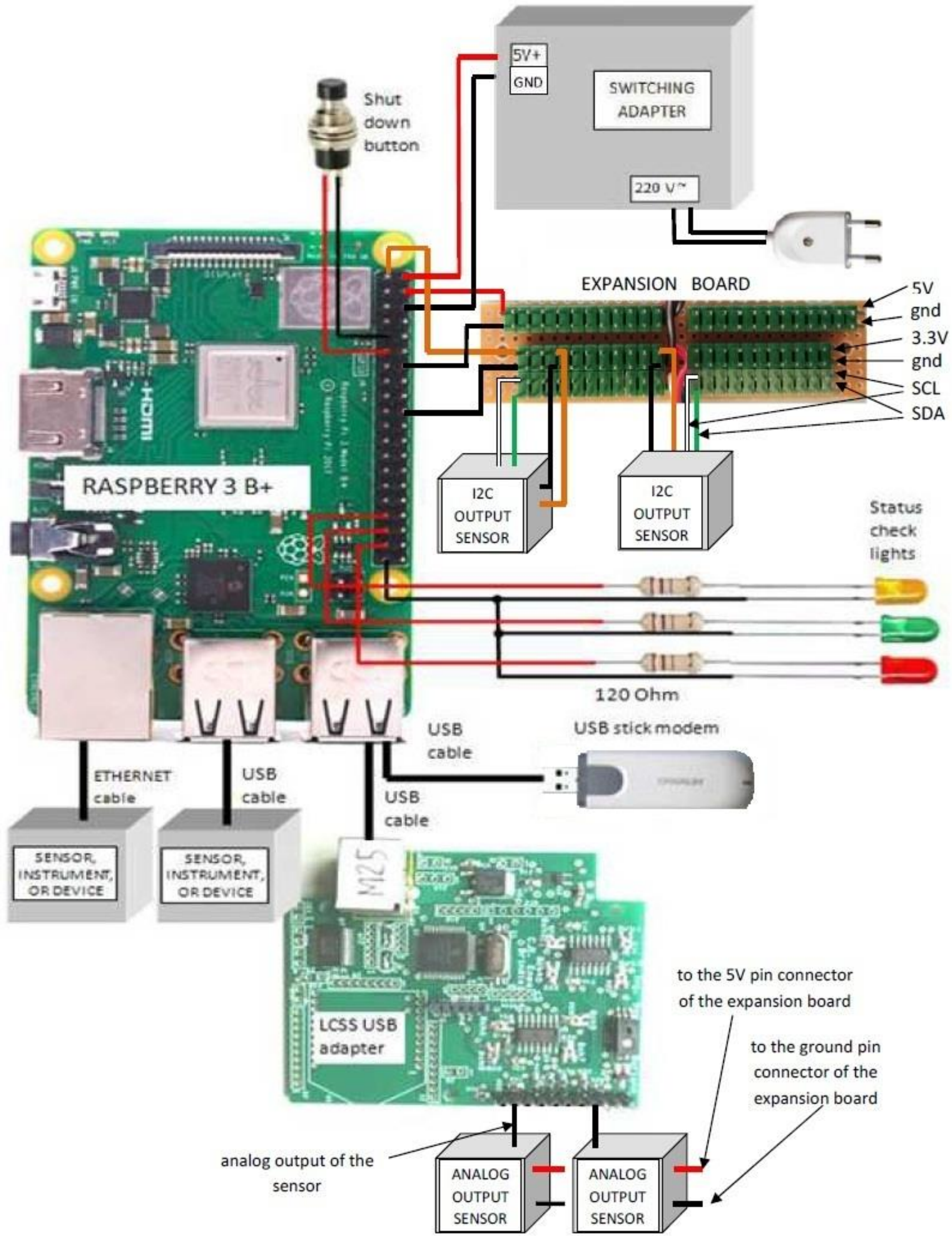

Figure 4: *SentinAir* hardware typical configuration. It includes an analog-to-digital converter board which enables the use of devices/sensors having analog output interfaces. This board can be the LCSS adapter or the ADC Pi by ABelectronics.

LCSS adapter assembly and setup

The LCSS adapter board allows the use of devices having analog output interfaces, and it is powered through its USB port. It converts the input analog signals into digital data that are given as output through the USB port. The analog outputs of the devices to be used with the LCSS adapter have to be wired to the J22 connector of the LCSS board (pins 4 to 11, see figure 6). These signals must be in the range 0 - 3,3 Volts. The LCSS adapter has a built-in temperature and relative humidity sensor. Eight analog-to-digital (ADC) converters feature it. They are high resolution (16 bits) converters, each of them having a signal amplifier and electronic noise filter. The user can set

the gain of each amplifier through the resistive trimmers R14, R15, R23, R22, R30, R31, R38, R39 placed on the LCSS board (see figure 6). LCSS board can be used with the RN42XV Bluetooth adapter, even though its use is not necessary for the SentinAir system. The LCSS board is also designed to be used with rechargeable Li-Ion batteries. They have to be wired to the J15 connector (see figure 5), and they are automatically charged through the USB port. The use of batteries is not necessary for the SentinAir system.
Printed Circuit Board (PCB) files for LCSS assembly and setup can be found in the folder "lcss adapter". They were created by the "ORCAD 10.0 Layout Plus" CAD software. LCSS adapter electronic schematics are present in the same folder. The project files of the schematics were created by "ORCAD 10.0 Capture". The file named "multisensore2_0.opj" is the main project file; by opening it by "ORCAD 10.0 Capture", it is possible to view and modify all the schematics files. Schematics and PCB files are also available as "pdf" documents in the "lcss adapter/pdf files/schematics" and "lcss adapter/pdf files/pcbs" subfolders respectively. LCSS adapter board is shown fully assembled in Figures 6 and 7. When the board is fully assembled, it is necessary to wire in short-circuit the two pins of the J11 and two pins of J13, as shown in figure 5. For the first use of the LCSS board, it is advisable to set the gain of the built-in amplifiers to the unit value. This operation can be carried out by using a little screwdriver and an ohm-meter. The resistive trimmers R14, R15, R23, R22, R30, R31, R38, R39, have to be set at their minimum possible value by rotating their little top screw (see figure 6).

Figure 5: The red arrows indicate the two pins of J11and J13 to short-circuit.

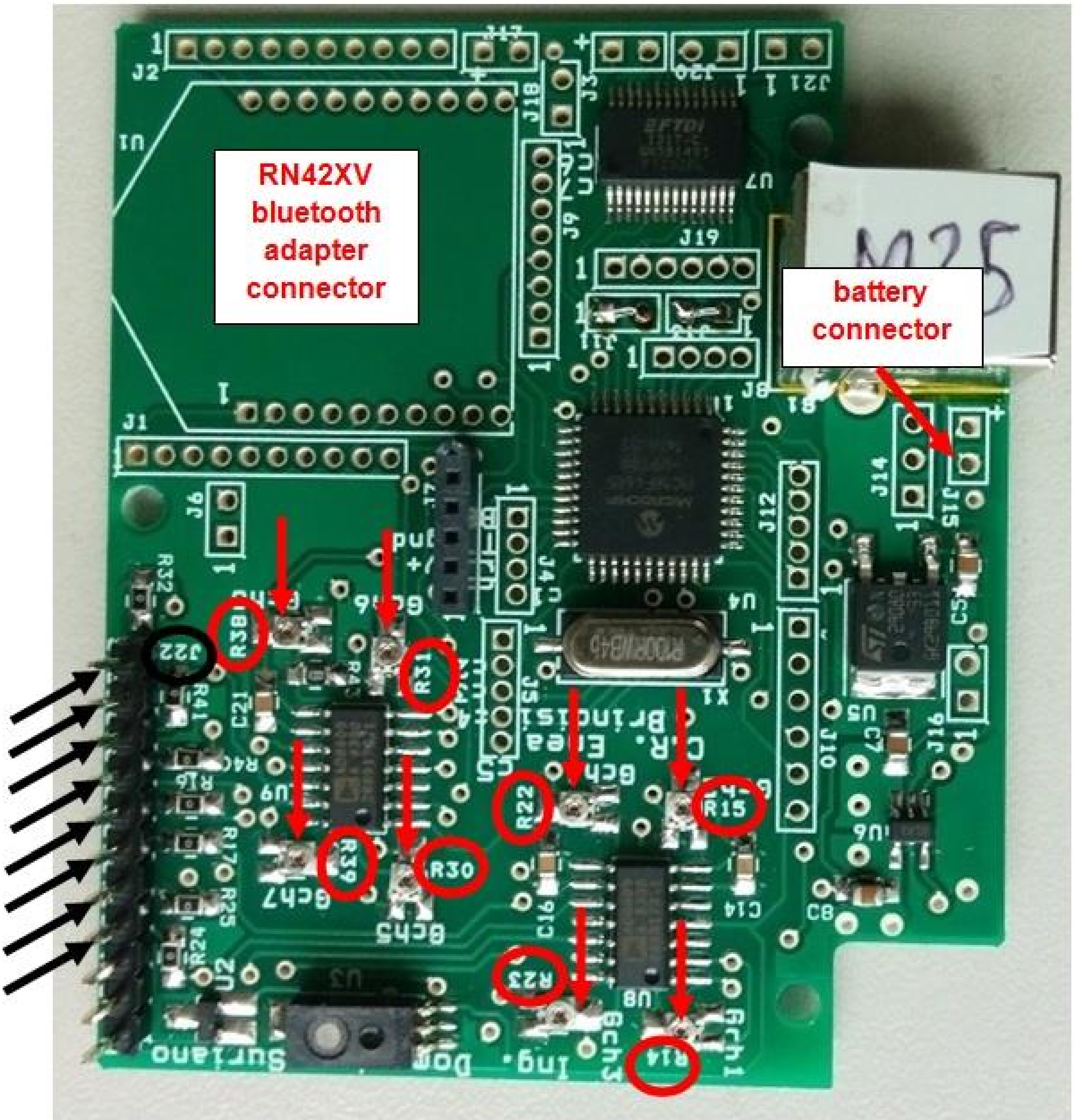

Figure 6: the black arrows and the black circle show the pins and the connector where the analog signals of the sensors have to be wired; while the red arrows and red circles indicate where the R14, R15, R23, R22, R30, R31, R38, R39 resistive trimmers are placed.

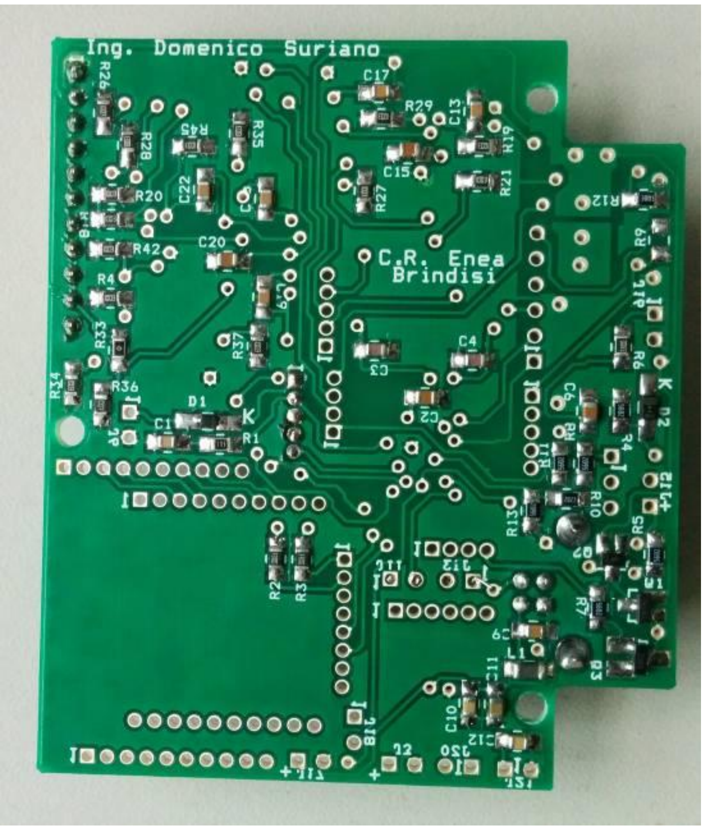

Figure 7: *LCSS adapter bottom view.*

The core of the LCSS adapter board is a PIC 18F4685 microcontroller by Microchip, which has to be programmed for the correct board operation. The tool needed to program the board is the "ICD2" programmer by Microchip. The necessary cable to connect the "ICD2" programmer to the LCSS board has to be assembled, as shown in figure 8. The cable is composed of an RJ11 female connector, a five positions male connector, and the wires. Optionally, the RJ11 connector can be fixed on a piece of a prototype board. The final appearance of the fully assembled cable is shown in figures 9 and 10. The electronic components needed for assembling both the LCSS adapter board and its programming cable are listed in the files placed in the subfolders "lcss adapter/bom/board" and "lcss adapter/bom/cable". In those files are also indicated the suppliers that sell all the required components. Once the programming cable and the LCSS board have been assembled, it is possible to burn the LCSS firmware on the board by using the "ICD2" programmer and the MPLAB IDE v. 8.66 distributed by Microchip (see https://www.microchip.com/development-tools/pic-and-dspic-downloads-archive). The program to burn on the LCSS board microcontroller is the "lcss_adapter.hex", located in the subfolder "lcss_adapter/firmware". When the user is going to connect the programming cable to the J7 socket of the LCSS board, he has to be careful to insert the male connector correctly: pin n.1 of it has to be inserted in the pin n.1 of the J7 socket. All the files related to the LCSS firmware are available in the subfolder "lcss_adapter/firmware/project_files" for modifications. The code of the LCSS firmware is written in "C" language; PCH libraries distributed by CCS are required to modify and compile the source code (see: https://www.ccsinfo.com/product_info.php?products_id=PCH_full).

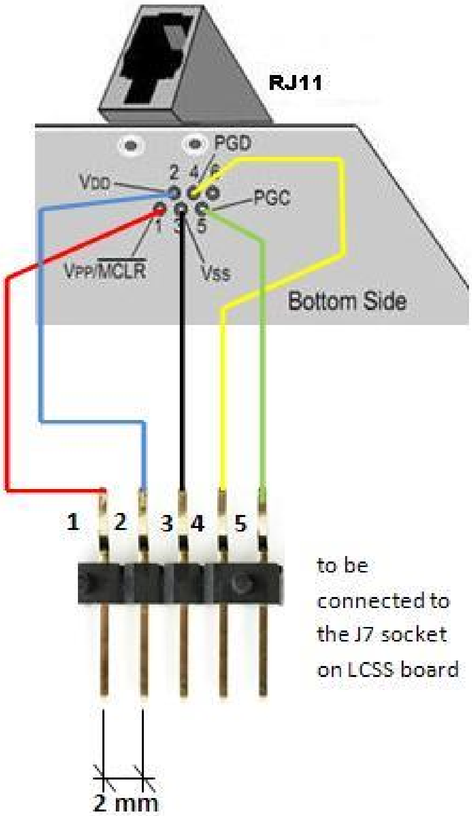

Figure 8: *connections to make for assembling the cable to program the LCSS board.*

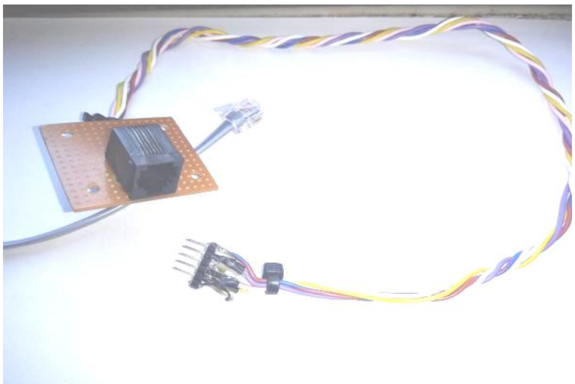

Figure 9: top view of the *programmer cable for LCSS board*.

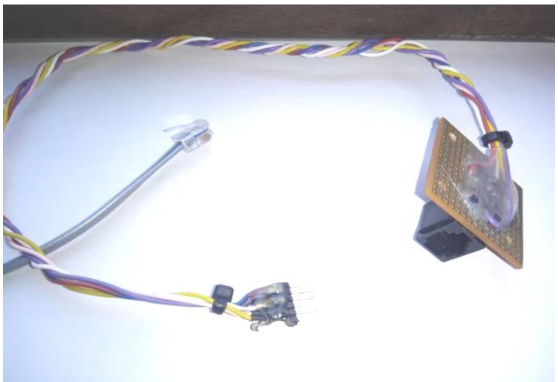

Figure 10: bottom view of the *programmer cable for LCSS board*.

Using the ADC Pi

The second option for using analog output interface devices with SentinAir is given by mounting the ADC Pi board on SentinAir. This board is provided by ABelectronics (see figure 11, 12) and it is based on the MCP3424 microprocessor, which is an analog-to-digital converter. The software driver for MCP3424, and therefore for the ADC Pi board, is already available in the repository of SentinAir in the folder "devices". This board is featured by two MCP3424 microprocessors each containing four analog inputs. Therefore, the total number of signals for each ADC Pi board is eight. The maximum analog-to-digital conversion accuracy provided by MCP3424 is 17 bits. The

ADC Pi is attached to the Raspberry board through the Raspberry GPIO connector. This board is interfaced with the Raspberry through the I2C port. No electrical wiring is necessary to connect the ADC Pi board to Raspberry, because the power and the I2C bus connections are provided by the GPIO connector to insert directly on the Raspberry board (see figure 12).

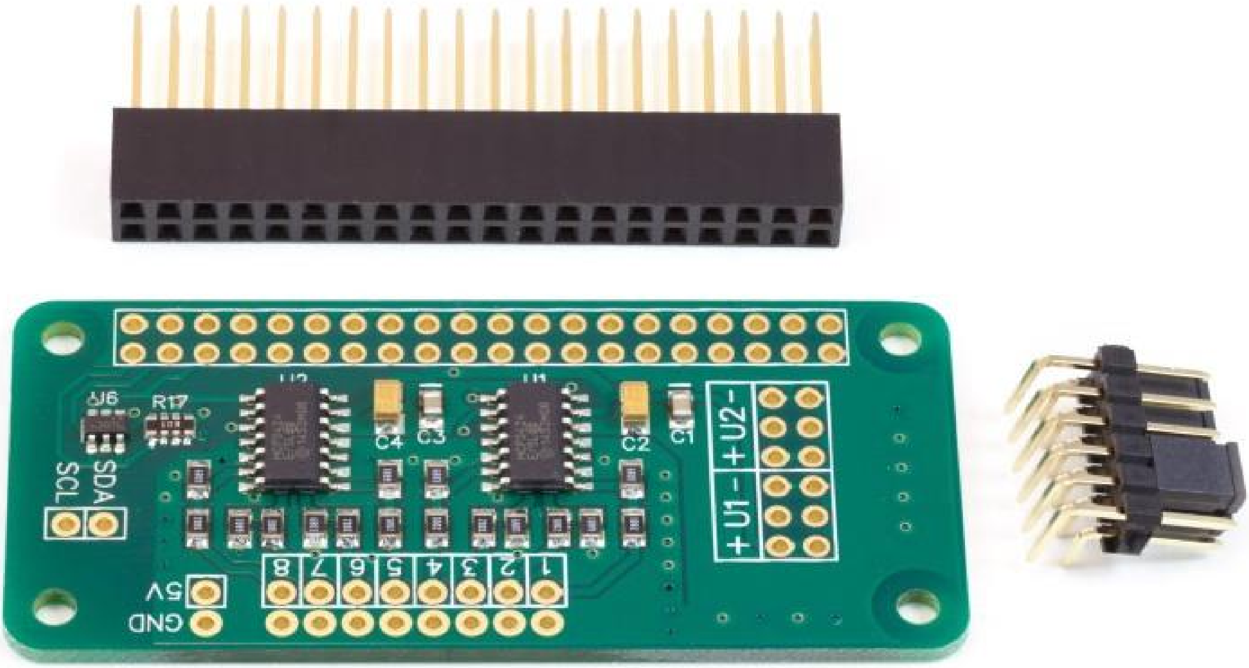

Figure 11: ADC Pi board layout. The V1 and V2 terminals are devoted to the I2C address selection of the two MCP3424 microprocessors of the board. The jumpers of the connector on the right side of the picture set the I2c bus address of each microprocessor. The pins numbered from 1 to 10 are the analog inputs.

Figure 12: the ADC Pi connected to the Raspberry Pi through the GPIO connector.

SentinAir software installation (basic procedure)

This is a quick procedure to install the SentinAir software making the minimum steps. If you desire a higher level of customization, you can refer to the “SentinAir software installation”chapter of this guide. The procedure is composed of two steps:

a) Installing the SentinAir sd card image

Download the latest SentinAir SD card image file. Write it in an SD card having at least 2 GB (4 GB recommended) of memory space by using, for example, *Win32diskimage.exe* program (for *Windows* operative systems).

b) Enabling the use of the IMAP-SMTP module (optional)

To use the IMAP-SMTP module (see the chapters “SentinAir start up”, and “controlling SentinAir by e-mail”), please, follow the steps described in points 10 and 11 of the section “Installing the SentinAir software modules”.

SentinAir start-up after the basic installation procedure

Supposing you have finished the a) point of the basic installation procedure, insert the SD card in the *SentinAir* device, plug one or more devices among the ones listed in table 1 into one of the system ports, and give power to *SentinAir*. After a few seconds, the green light starts blinking. Following this stage, the yellow light starts blinking. This means that the system is performing the port scanning. If one or more devices/sensors/instruments are found, the system automatically starts to acquire data (entering in the active monitoring mode and starting a new measurement session), and the yellow light turns on. To stop the measurement session, you have to use the “SentinAir user interface” and send the on-purpose command to the system (please, refer to the “Using the SentinAir user interface” chapter of this guide).

SentinAir software installation (advanced procedure)

The procedure described below allows skilled users to customize the *SentinAir* system during the installation process. All the software necessary to operate a *SentinAir* device runs on *Raspbian Stretch Lite* Operative System (OS); therefore, the first step is downloading and installing it on a *SentinAir* device or a *Raspberry 3 B+* board. For a correct *SentinAir* software installation, please, follow in sequence the below procedures. For the first software installation, the user has to plug a keyboard into the USB port and a monitor into the HDMI port. Subsequently, it is possible to access the system via SSH connections by using programs such as *putty.exe* or *WinSCP.exe* (for *Windows* operative systems).

a) Installing Raspbian Stretch Lite operative system

1. Download the *Raspbian Stretch Lite* OS image into a computer from the link: https://www.raspberrypi.org/downloads/raspbian/ and flash it in an SD card memory featured by at least 2 GB of room by using, for example, *Win32diskimage.exe* program (for *Windows* operative systems).
2. Create a blank file called *ssh* (no extension). Save this file in the root of the MicroSD card with the flashed image. This operation is for enabling SSH connections at start-up.

3. In order to allow *SentinAir* to connect to a Wi-Fi network when it turns on, create a *wpa_supplicant.conf* file and write inside the following lines:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP =netdev
update_config=1
country=<your country code>
# home network; allow all valid ciphers
network={
        ssid="<your network name>"
            scan_ssid=1
        key_mgmt=WPA-PSK
        psk="<your network password>"
}
```

   Save and close the file. If you are using the *Windows* operative system to write the file, you have to convert it to the *Linux/Unix* standard. To do this, it is possible to use the program *dos2unix.exe*. Save *wpa_supplicant.conf* in the root of the MicroSD card with the flashed image.
4. Plug the Sd card in the *Raspberry* board and power it.
5. Find the *SentinAir* IP address by using programs to scan the network set in the *wpa_supplicant.conf;* for example, for this purpose, there is an "app" for the Android system called *FindPi*. Once found which is the *SentinAir* device IP address, it will be possible to continue the installation process by connecting to *SentinAir* from another computer via SSH connections. To do this, for example, it is useful the program *putty.exe,* if the computer has a *Windows* operative system running on it.

b) Preparing Raspbian for SentinAir installation

1. Power the *SentinAir* device or the *Raspberry 3 B+* board, log in it, and type the command: *sudo raspi-config*.
2. To set the device name, select *Network Options/Hostname* and insert the device name. It is mandatory to enter a name with a "hyphen" inside, for example, *sentinair-S1*. This is for the correct functioning of *SentinAir* software. The part of the name following the hyphen is the identity number you choose for the device. Do not exit from *raspi-config* yet.
3. To use the serial port for connecting devices, select *Interfacing options/Serial* and then press in sequence "NO" and "YES". If everything is fine, you should see the message "*The serial login shell is disabled. The serial interface is enabled*". Then press "OK".
4. To use I2C ports for connecting devices, select *Interfacing options* and then *P5 I2.* Select *"yes"* to enable the I2C ports.
5. To expand the use of the memory on your SD card, select *Advanced Options/ expand Filesystem*. Select then *FINISH* and reboot the system.
6. Run the command *sudo nano /etc/hosts* and update the file by modifying the line: "*127.0.0.1 raspberrypi"* in the line: "*127.0.0.1 <device-name>",* where <device-name> has to be the same name selected in the step 2 (which is, as for example *sentinair-S1*). Reboot the system.

7. To complete the configuration for the serial port called */dev/ttyAMA0*, we have to disable the Bluetooth service that uses this port and set it as a general-purpose serial port. To do this, type *sudo nano /boot/config.txt,* and then add at the end of the file, the row: *dtoverlay=pi3-disable-bt*. Save and close the file. After doing this, prevent Bluetooth service to use serial port by typing the command: *sudo systemctl disable hciuart.* Then reboot the system by running *sudo reboot*.
8. Update the *Raspbian* repository by typing the command: *sudo apt-get update*.
9. From now on, to perform the next steps of installation, it is needed to log into the system with the help of a keyboard and a monitor. This is necessary because the built-in Wi-Fi modem will turn in a Wi-Fi access point, and therefore it is impossible to continue while *SentinAir* is connected to a pre-existing Wi-Fi LAN. Moreover, to connect the device to the Internet, it is going to be necessary a USB stick modem with a valid SIM card enabled by an Internet Service Provider to connect to the Internet (for example, the model *Huawey E303* does not require installation procedures). This is why the Ethernet port will be reserved to connect instruments to form a local "stand-alone" LAN. Therefore run *sudo halt* to stop the system, plug the USB stick modem, the keyboard, and a monitor, then log in again.
10. Download and install *dnsmasq* and *hostapd* by typing the command *sudo apt-get install -y dnsmasq hostapd.*
11. Stop the execution of the software just installed by running the commands: *sudo systemctl stop dnsmasq* and *sudo systemctl stop hostapd*
12. Set a static IP for the system and configure *dhcpcd* by opening the file *dhcpcd.conf* with the command *sudo nano /etc/dhcpcd.conf* and add at the end of the file the lines:

    ```
    interface wlan0
    static ip_address=192.168.4.1/24 (or an alternative IP address you choose)
    nohook wpa_supplicant
    dhcp-range=192.168.4.2,192.168.4.100,255.255.255.0,24h
    interface eth0
    static ip_address=192.168.20.1/24
    static routers=192.168.20.1
    nogateway
    ```

    Save and close the file.
13. Configure the *hostapd* software by creating the file *hostapd.conf* , to do this, run the command *sudo nano /etc/hostapd/hostapd.conf* . Then insert in the file the following lines:

    ```
    interface=wlan0
    driver=nl80211
    ssid=device-name     (for example: sentinair-S1)
    hw_mode=g
    channel=7
    wmm_enabled=0
    macaddr_acl=0
    auth_algs=1
    ignore_broadcast_ssid=0
    wpa=2
    ```

wpa_passphrase=sentinair1
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
Then save and close the file.

14. Complete *hostapd* configuration by typing *sudo nano /etc/default/hostapd.* Then modify the line:
    #DAEMON_CONF=""
    in the new line
    DAEMON_CONF="/etc/hostapd/hostapd.conf "
    Save and close the file.
15. Run the command *sudo nano /etc/init.d/hoastapd* and replace the line
    DAEMON_CONF=
    whit the line
    DAEMON_CONF=/etc/hostapd/hostapd.conf.
    Save and close the file.
16. Run the command *sudo nano /etc/dnsmasq.conf* and add at the end of the file the following lines:
    interface=wlan0
    dhcp-range=192.168.4.2,192.168.4.100,255.255.255.0,24h.
    Save and close the file.
17. Finally, run the commands:
    sudo systemctl restart dhcpcd.
    sudo systemctl unmask hostapd
    sudo systemctl enable hostapd
    sudo systemctl start hostapd
    sudo service dnsmasq start
    sudo reboot

c) Installing Lighttpd webserver

1. Install *lighttpd* by the command
   sudo apt-get install –y lighttpd.
2. Enable the *cgi-bin* module of the web server by the command
   sudo lighty-enable-mod cgi
3. Open */etc/lighttpd/conf-available/10-cgi.conf* and modify the lines:

```
$HTTP["url"] =~ "^/cgi-bin/" {
  cgi.assign = ( "" => "" )
}
```

   in

```
$HTTP["url"] =~ "^/cgi-bin/" {
  alias.url += ("/cgi-bin/"=>"/var/www/cgi-bin/")
  cgi.assign = ( "" => "" )
}
```

4. Run the command *sudo service lighttpd force-reload*

5. Create the folders: */var/www/html/data*, */var/www/html/log* and */var/www/html/img*. Give them the read, write, and execute rights for all the users.
6. Rename the file */var/www/html/index.lighttpd.html* in */var/www/html/index.lighttpd.html_*. Copy the files *index.html*, *sentinair.jpg*, and *btnplt.png* in the folder */var/www/html/*.
7. Copy the files *fileinspector.py* and *surianocliweb.py* in the folder */var/www/cgi-bin/* and give them read, write, and execution rights for all the users. Then reboot the system.

d) Installing the SentinAir software modules

1. Install *python3-serial* by running the command *sudo apt-get install python3-serial*.
2. Install *pip3* by running the command *sudo apt-get install python3-pip*
3. Install the library *RPi.GPIO* by running the command *sudo apt-get install python3-RPi.GPIO*.
4. Install the library *libscrc* by running the command *sudo pip3 install libscrc*.
5. Install the library *smbus2* (for i2c manging) by typing the command *sudo pip3 install smbus2*
6. Install the library *python3-matplotlib* by running the command *sudo apt-get install –y python3-matplotlib*
7. Reboot the system.
8. Create the folder */home/pi/sentinair*. SentinAir software files have to be mandatorily copied into this folder
9. Copy in */home/pi/sentinair/* the following files and folders:
   - *sentinair.py*
   - *sentinair_system_manager.py*
   - *sentinair_system_installer.py*
   - *imap-smtp-interface.py*
   - *imap-smtp-monitor.sh*
   - *mail-config.sentinair*
   - *devices*

   The folder "*devices*" is the folder where devices drivers have mandatorily to be placed.
10. In order to get started SentinAir software when the system boots, insert the following lines in the file */etc/rc.local* before the line “exit 0”:
    python3 /home/pi/sentinair/sentinair_system_manager.py&
    WARNING: if you are editing rc.local on a Windows OS, remember to convert it in Unix/Linux format through the “*dos2unix.exe*” utility.
11. Open the “*mail-config.sentinair”* file and put your e-mail account data in the lines without the comment character (#) avoiding inserting the “space” character, like the example below:
    ```
    MAIL_ADDRESS="my-email_address@my_company.com"
    MAIL_PWD="my-email-account-password"
    SMTP_SERVER="my-smtp-server-url"
    IMAP_SERVER="my-imap-server-url"
    ```
12. Reboot the system

Setting up SentinAir internet connection

The USB stick modem Internet plugged into one of the USB ports available gives the channel to reach a SentinAir system from the. If a *Huawey E303* is used for the purpose, no installation procedures are required. Other USB stick modem models have not been tested yet. Internet Service Providers (ISP) usually do not give public IP addresses to the user; therefore, *SentinAir* could not be reachable from anywhere. To overcome this issue, it is possible to use the "IP tunneling" services that give the possibility to get around this hurdle. Prices of these services can vary from free to a few Euros per month. Examples of web companies offering these services are www.dataplicity.com, or www.pitunnel.com or. https:\\remote.it.

SentinAir start-up

SentinAir gets started when the electronic hardware is powered, alternatively, by typing the command: *sudo python3 /home/pi/sentinair/sentinair_system_manager.py*. After few seconds the system is powered, you should see the green light asynchronous blinking, which indicates that the CPU is exchanging data with the SD memory and the system is setting up (see table 3).

Table 3: check light behaviors and their meaning

LED	LED STATUS	MEANING
RED	OFF	System turned off
	BLINKING	Sensor fault (the user must see the log file to get more information)
GREEN	ASYNCHRONOUS BLINKING	Data exchange activity between CPU and SD memory
YELLOW	OFF	System active, monitoring not active
	FAST BLINKING	The system is performing the port scanning to find which devices are connected
	SLOW BLINKING	The shut-down procedure is going on, following the shut-down button pressing
	ON	Monitoring active

When SentinAir gets started, before getting ready, it performs some preliminary operations to work appropriately. Firstly, it makes blink the yellow check light to indicate that the system is performing the scanning of the system ports to find which devices, among the ones whose driver is already installed in the system, are plugged into. When a device is found, it got connected to the system. This operation might last a few seconds, and during this time, you should see if you have a monitor plugged into the HDMI port, system messages on the screen. From now on, all the relevant operations and events are logged in the on-purpose log file of the system manager (the default name is *sentinair-log.txt*). During this period, the system cannot accept commands from the user; sometimes, it can last a few minutes. As soon as this operation is completed, *SentinAir* reads the file *status.sentinair*, where there is information about the device status: "standby", or "active monitoring". If the system is in "active monitoring" status, a new measurement session gets started; therefore, the yellow led check light turns on, and a new file containing the measurement data is created. The last operation of the start-up phase is the sending of an e-mail to the e-mail address indicated in the file *mail-config.sentinair* if the USB stick modem is plugged into the system (see figure 14). In the current version of the software, *SentinAir* configuration is featured by a static IP on the cable Ethernet port. Devices connected to this interface will form a "stand-alone" net that is not connected to the Internet. If the user needs to connect more devices, for example, it is possible to use an Ethernet switch.

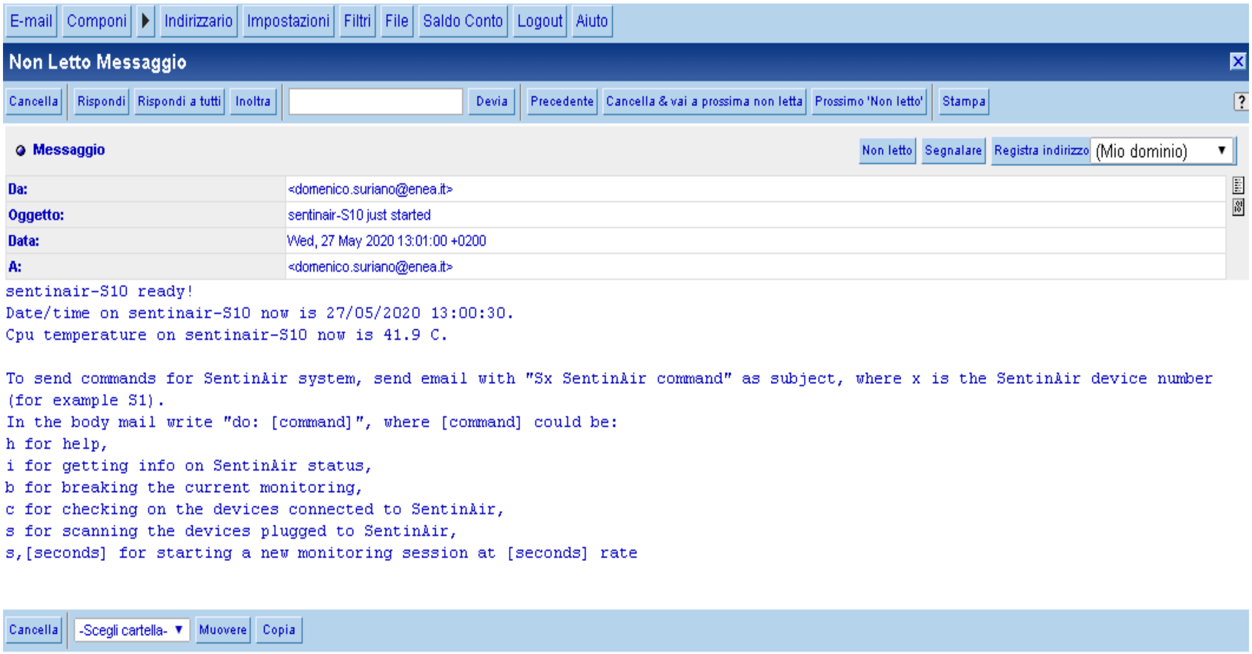

Figure 14: E-mail sent by SentinAir at start-up time.

Installing devices drivers in SentinAir

SentinAir can read data from any device as long as the specific driver of the device is installed in the system. It is mandatory that the device driver files have to be in the folder */home/pi/sentinair/device*. In order to easily install drivers, the user can log into the system by SSH connection via the Wi-Fi link through the LAN set up by *SentinAir*. Then the user has to use the module *sentinair_system_installer.py* by opening it through the command:
sudo python3 /home/pi/sentinair/sentinair_system_installer.py.
Once opened the installer, the commands available are listed in table 4.

Table 4: commands available for the SentinAir system installer

Command function	Command syntax	Example	Effect
Device driver installation	i,device_name.py	i,pms3003.py	Installs the driver for the device pms3003
Device driver uninstalling	u,device_name.py	u,pms3003.py	Uninstalls the driver for the device pms3003
Checking the current drivers installed	c	c	Displays a list of the drivers currently installed
Modifying the current path of the system manager module	m	m	Writes in the file "manager_dir.sentinair" the new path of the manager system module

Asking for the commands list available	h	h	Displays a brief manual where are shown the commands
Quit the program	q	q	Quits the programs

An example of *sentinair_system_installer.py* use is shown in figure 15. After updating the system with new installations, it has to be rebooted or restarted.

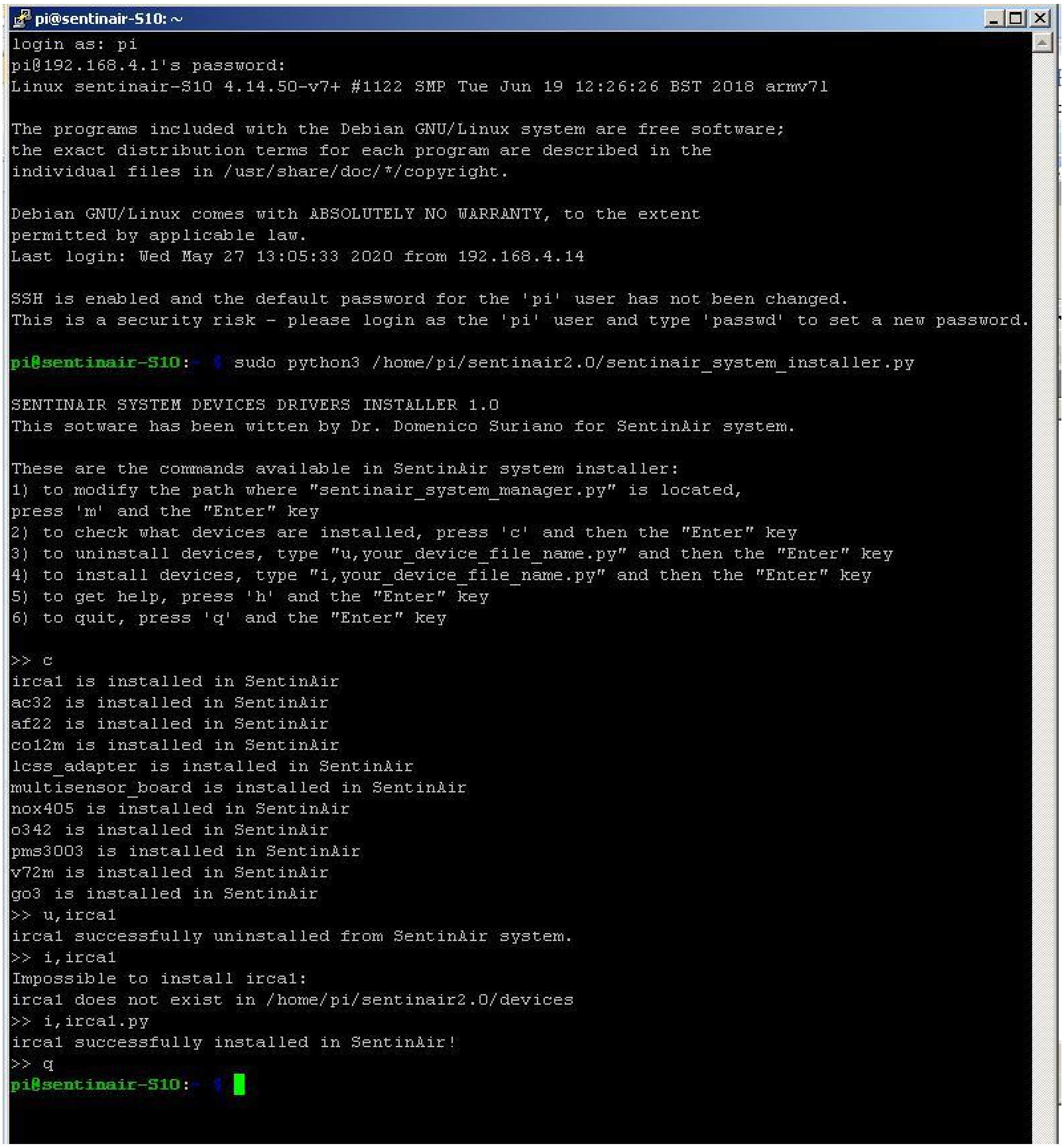

Figure 15: the installer user interface

Using the SentinAir user interface

SentinAir system main interface is provided by the *sentinair.py* module. The user has to log into the system by SSH connection via the Wi-Fi link through the LAN set up by *SentinAir*. To open the *SentinAir* user interface, type the command:
python3 /home/pi/sentinair/sentinair.py.
Once opened the user interface, the commands available are listed in table 5.

Table 5: commands available for the SentinAir user interface

Command function	Command syntax	Example	Effect
Starting a new monitoring session	s,seconds	s,60	A new monitoring session gets started. Every 60 seconds, device measurements data are read and stored in a file
Stopping a measurement session	b	b	Stops the current measurement session
Retrieving information about the system status	i	i	Displays information such as the devices currently connected to the system, the sampling rate of the measurement session, or if the system is on standby.
Retrieving information about the devices connected to the system	c	c	The system returns the information about the connected devices: identity, units of measurements, current measurements
Asking for the commands list available	h	h	Displays a brief manual where are shown the commands
Scanning the ports and connecting the devices plugged into *SentinAir* ports	s	s	The system scans all the ports (USB, Ethernet, etc.) and recognizes if one of the installed devices is plugged into
Quit the program	q	q	Quits the programs

An example of the use of the *SentinAir* user interface is shown in figure 16. If the system is not in active monitoring mode (you can see this by checking the yellow light, which should be off), the message “No measurement session ongoing” is shown. If you want to start acquiring data from sensors (putting the system in active monitoring mode), you have to type the command “s,60”. This way a monitoring session will start at 60 second sampling rate and the yellow check light will turn on.

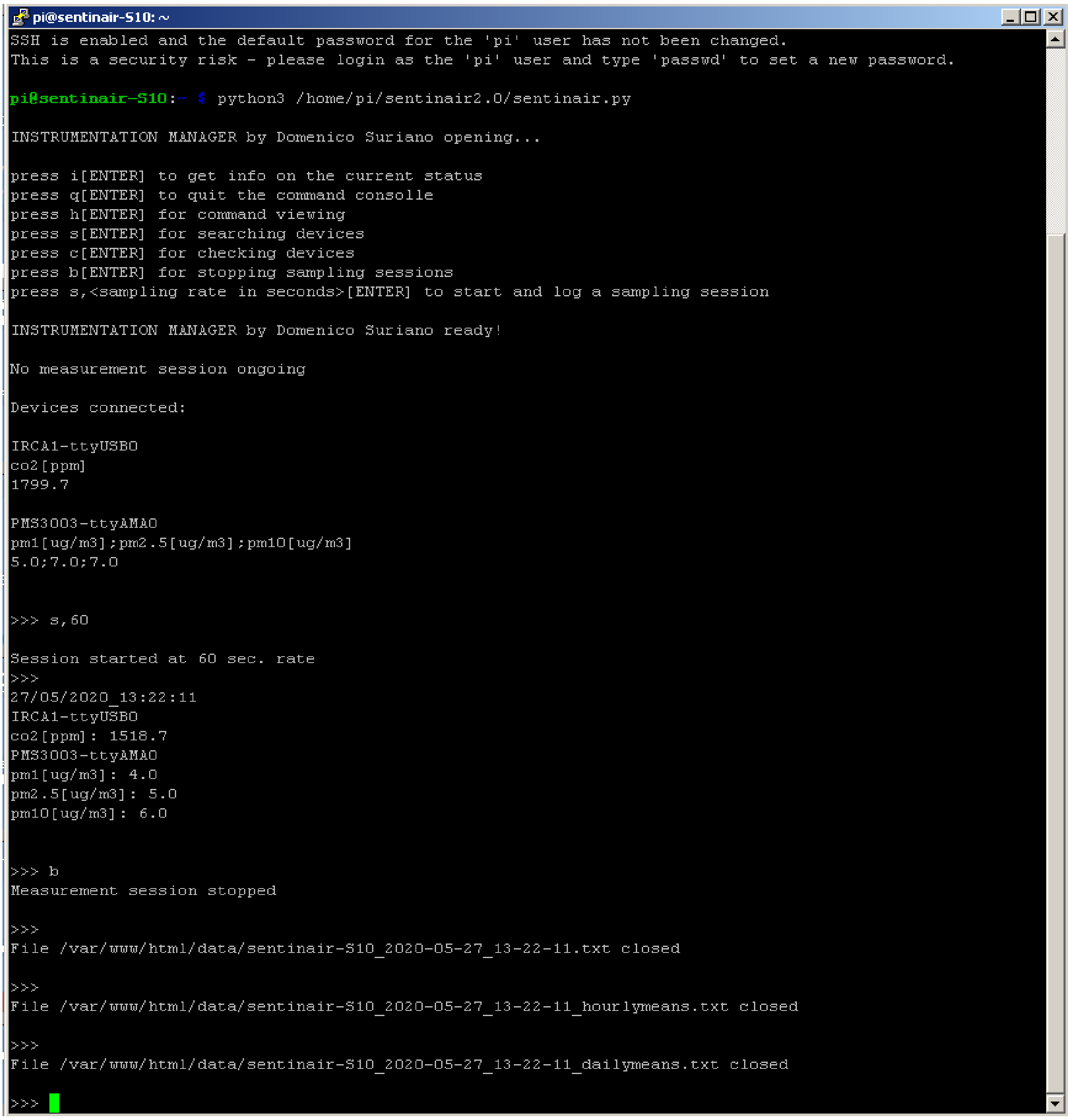

Figure 16: the *SentinAir* user interface.

Setting SentinAir system date and time

SentinAir system has not a hardware Real Time Clock (RTC). If the device is connected to internet, it automatically updates date and time of the system, otherwise, if necessary, the user must update manually it. To carry out this operation, no measurement session must be active; in this case, before setting correct date and time, the user must stop the current measurement session by opening the SentinAir user interface and by typing the command "b" (see table 5). Once done this operation, the user must quit the program by launcing the "q" command (see table 5), and subsequently, he can set the current date and time by running the Linux command "date" as follows: *"sudo date –s="YYYY-mm-dd hh:mm:ss"*. For example, if you want to set the date and time on the 23th of February 2022 at 23:44:21, you must type the command: *"sudo date –s="2022-02-23 23:44:21"*.

Controlling SentinAir by e-mail

SentinAir has been designed for being used in all that cases where it is necessary to perform experiments or measurements in remote areas, where possibly the internet link is weak or unstable, and therefore, where it is difficult the remote control and the data download from the device. For this purpose, commands and data can be exchanged by e-mail sending and receiving. *SentinAir* checks periodically on the IMAP server configured in the *imap-smtp-interface.py* for the presence of e-mails containing commands, and when the radio signal allows it, sends back the response. The *imap-smtp-interface.py* module is optional, and it is not essential for *SentinAir* operation. If it is present, it is launched by *sentinair_system_manager.py* at start-up. The user can send commands through e-mail for both the *SentinAir* system or for the operative system programs and services. A typical *SentinAir* command is, for example, *"s,60"*, which causes the start of a new measurement session at 60 seconds sampling rate. A command directed for the operative system could be, for example, *"ls –l"*, which returns the list of files and directories contained in the current system directory. E-mails containing commands for the operative system must have as subject the text: "*device_ID system command*",
where the *"device_ID"* is the part of the device name which follows the hyphen. For example: if the device name was set as *"sentinair-S1"* during the software installation process, then the *device_ID* will be *"S1"* (see figure 17 and 18). Moreover, e-mails containing commands directed to *SentinAir* software must have as subject the text:
"*device_ID SentinAir command*" (please, mind the correct case of letters).
The body of the e-mail must have a line where there is "*do: command*". For example: *"do: s,60"* (please, mind the "space" character between "do:" and the command). It is possible to download every kind of file present in the device by sending an e-mail having the subject as: "*device_ID system command*", and the body having the line as: "*do: fget /path_of_the_file/file_to_download"* (mind the space characters between "do:" and "fget" and between "fget" and "/path_of_the_file/file_to_download"). For example: "*do: fget /var/www/html/log/sentinair-log.txt"*.

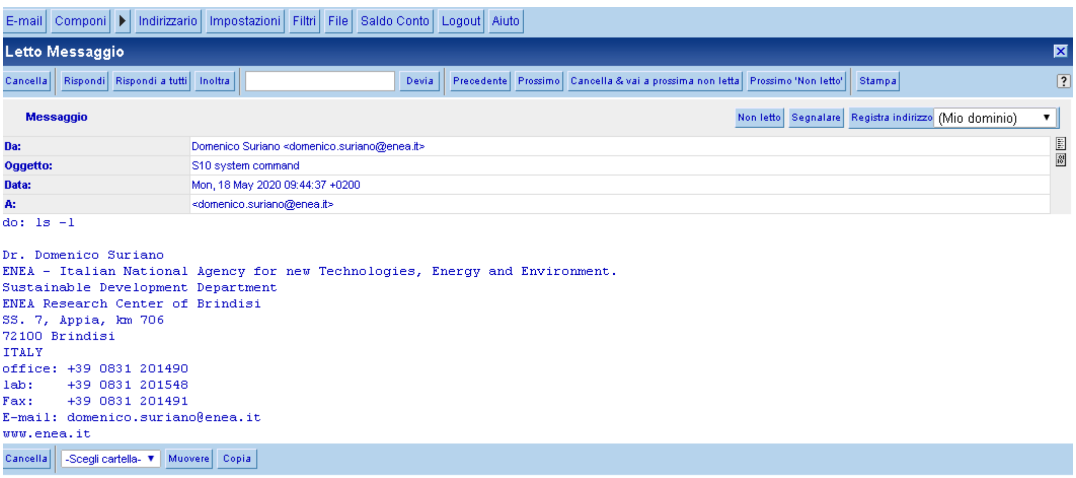

Figure 17: e-mail sent to *SentinAir* containing a command for the operative system

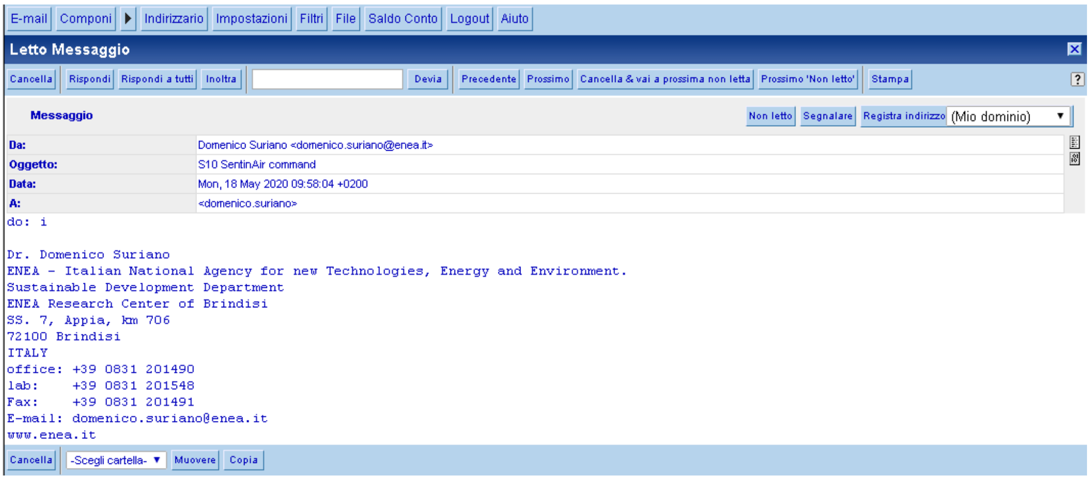

Figure 18: -mail sent to *SentinAir* containing a command for SentinAir system

SentinAir operations featuring the measurement session

When a new measurement session gets started (for example, by the command "*s,60*"), the yellow led check light turns on. Three new files are created: the file containing the records of the measurements along with the timestamps (see figure 19) and named: "*device_name_date_time.txt*" (for example *sentinair-S1_2020-02-03_09-30-15.txt*), the file containing hourly averages of measurement and named: "*device_name_date_time_hourlymeans.txt*" (for example: "*sentinair-S1_2020-02-03_09-30-15_hourlymeans.txt*"), and the file containing the daily averages of the measurements, named: "*device_name_date_time_dailymeans.txt*" (for example: "*sentinair-S1_2020-02-03_09-30-15_dailymeans.txt*"). Hourly and daily averages are computed in real-time by the module *sentinair_system_manager.py,* and they are stored in the path indicated in that module (which is for default: *"/var/www/html/data*"). Another operation performed by the *sentinair_system_manager.py* module during the measurement session is the plotting of each magnitude measured by each of the devices connected to *SentinAir*. The plots are stored in jpg files placed in the path *"/var/www/html/img*" (which is their default path). Those files are encapsulated in web pages and served to the user when he connects by a web browser to *SentinAir*.

```
Date/time;IRCA1-ttyUSB0_co2[ppm];PMS3003-ttyAMA0_pm1[ug/m3];PMS3003-ttyAMA0_pm2.5[ug/m3];PMS3003-ttyAMA0_pm10[ug/m3]
27/05/2020_15:28:22;1228.9;5.0;6.0;7.0
27/05/2020_15:28:52;2156.7;5.0;7.0;7.0
27/05/2020_15:29:22;4035.6;4.0;6.0;6.0
27/05/2020_15:29:52;2188.8;5.0;7.0;8.0
27/05/2020_15:30:22;1676.5;5.0;8.0;8.0
27/05/2020_15:30:52;1955.0;5.0;7.0;7.0
27/05/2020_15:31:22;2026.3;5.0;6.0;7.0
27/05/2020_15:31:52;2185.1;5.0;6.0;6.0
27/05/2020_15:32:22;3281.6;4.0;6.0;7.0
27/05/2020_15:32:52;2805.8;4.0;5.0;5.0
27/05/2020_15:33:22;1940.0;5.0;5.0;5.0
27/05/2020_15:33:52;2222.1;5.0;6.0;7.0
27/05/2020_15:34:22;2477.7;4.0;5.0;5.0
27/05/2020_15:34:52;1438.0;4.0;6.0;6.0
27/05/2020_15:35:22;2571.1;5.0;7.0;7.0
27/05/2020_15:35:52;2544.4;4.0;6.0;6.0
27/05/2020_15:36:22;1882.4;4.0;6.0;6.0
27/05/2020_15:36:52;3862.6;5.0;7.0;7.0
```

Figure 19: a file containing records of a *SentinAir* measurement session.

SentinAir web pages

The user can monitor *SentinAir* activity by connecting through a web browser to *SentinAir* (see figure 20). As for the other user interfaces given by the *sentinair.py* module and the *sentinair_system_installer.py* module, the webserver of *SentinAir* can be reached by connecting through a Wi-Fi link at the address set in the */etc/dhcpcd.conf* file (which default is 192.168.4.1), or through internet connections via the selected "IP tunneling" service. Web pages of *SentinAir* allow the user to download all data and log files that are in the system.

Figure 20: web pages from *SentinAir*

Shutting down SentinAir

To shut down the system through the button shown in figure 4, the user has to press it and hold it pressed until the yellow light starts blinking. At that moment, it is possible to release the button. Before taking off the power from the system, you have to wait that the yellow light stops blinking and after that, wait the green light stops blinking and the red light turns off. Optionally, you have to type the “sudo halt” command to the operative system running on the Raspberry board to properly shut-down the system.

How to write device drivers for SentinAir

The creation of new device drivers for *SentinAir* is highly expected. It is desirable that researchers, or users, write drivers for their own devices to use with the *SentinAir* system. In order to facilitate this task, driver templates have been written and stored in the folder “devices/device-driver-templates” of the SentinAir Github repository. When a new driver is going to be written, the user must follow some rules in order to get it fully compatible and operating with the pre-existing software. Some of these rules are common to all types of devices (USB, serial, I2C, Ethernet), while there are rules specific for each type of sensor/device. We recall that sensors or devices featured by analog output interfaces must be used with the LCSS adapter board (having a USB output interface), or with the ADC Pi (having an I2C output interface). A good way to getting started with writing new device drivers is by modifying the template file related to your device type ("ethernet-device-template.py", "usb-device-template.py", "serial-device-template.py", or "i2c-device-template.py").

Here below it is shown a step by step procedure which hopefully will help the user to correctly write new device drivers. In doing this, we will report some examples referring to the files of some drivers already developed ("nox405.py", " ac32.py", "bh1750.py"), and to the files of the driver templates.

1. Import the necessary library. For I2C devices, "smsbus2" is mandatory (see "bh1750.py" and "i2c-device-template.py"), for USB and serial devices "serial" is mandatory (see "nox405.py" and "serial-device-template.py"), for Ethernet devices "socket" is mandatory (see " ac32.py" and "ethernet-device-template.py"). Also import other libraries specific for your device operation (see for example "time" and "_thread" in "nox405.py")
2. Set the mandatory constant strings *CONNECTION_TYPE, DEVICE_IDENTITY*, and *DEVICE_SENSORS* common for all the device types following the indications in the driver template files ("ethernet-device-template.py", "usb-device-template.py", "serial-device-template.py", or "i2c-device-template.py"). The *DEVICE_SENSORS* string contains the magnitude names measured by the device, separated by a semicolon character; for example (see "nox405.py"): *"no2[ppb];no[ppb];nox[ppb]"*. Then set the constants specific for each type of device*: I2C_ADDRESSES*, which must be a Python list of integer values (for I2C devices, see "bh1750.py" and "i2c-device-template.py"), *ETH_ADDRESSES*, which must be a list of strings (for Ethernet devices, see " ac32.py" and "ethernet-device-template.py"), *DEVICE_BAUD_RATE*, which must be an integer value (for serial and usb devices, see "nox405.py" and "serial-device-template.py"). Set the other constants (if any) specific for your device operation.
3. Create the device class name. It is mandatory that the class name must be the same as the driver file name with the first letter capitalized. For example: the driver "nox405.py" contains the class *Nox405*.
4. In the *"__init(self)*" method of the device class, you can find the mandatory variables common for all the device types:
   self.identity = DEVICE_IDENTITY
   self.connection_type = CONNECTION_TYPE
   self.sensors = DEVICE_SENSORS.
   There are also variables specific for a device type. For I2C device (see "bh1750.py" and "i2c-device-template.py"), you will find:
   self.addresses = I2C_ADDRESSES
   self.device_address = 0x00.
   For Ethernet devices you will find (see " ac32.py" and "ethernet-device-template.py"):
   self.addresses = ETH_ADDRESSES
   self.device_address = None
   self.sk = None
   For serial or usb devices you will find (see "nox405.py" and "serial-device-template.py"):

self.baud_rate = DEVICE_BAUD_RATE
self.portname = ""
self.port = None.

The user must not modify the above-mentioned lines, but he can add variables necessary and specific for his device operation (see, for example, the "__init(self)" method in the file "nox405.py").

5. The device class must contain the following methods common for all the device types; they are:

```
def getConnectionType(self):
    return self.connection_type
def getIdentity(self):
    return self.identity
def getSensors(self):
    return self.sensors
def setIdentity(self,idstring):
    self.identity = idstring
```

Moreover, for a specific device type, there are other mandatory methods. For I2C devices device (see "bh1750.py" and "i2c-device-template.py"), we have:

```
def getConnectionParams(self):
    return self.addresses
def terminate(self):
    self.device_address = 0x00.
```

For Ethernet devices (see " ac32.py" and "ethernet-device-template.py"), we have:

```
def getConnectionParams(self):
    return self.addresses
def terminate(self):
    try:
        self.sk.close()
    except:
        return
def __del__(self):
    try:
        self.sk.close()
    except:
        return
```

For usb or serial devices (see "nox405.py" and "serial-device-template.py"), we have:

```
def getConnectionParams(self):
    return [self.portname,self.baud_rate]
def terminate(self):
    try:
        self.port.close()
    except:
        return
```

```
def __del__(self):
    try:
        self.port.close()
    except:
        return.
```

The user must not modify the above-listed methods, but he can add other ones specific for his device operation (see, for example, the “nox405.py”, ” ac32.py”, “bh1750.py”).

6. The class must have the method called “*connect(self,address)”* in case of I2C or Ethernet devices, or “*connect(self,portname)”* in case of serial or usb devices. This method detects if the device is connected and finds the connection parameters, for example, if it is searching for an Ethernet or I2C device, it finds the device address; or, if it is searching for a USB or serial device, it finds the USB port. If no device is found, it returns the "0" value, otherwise, it returns "1". For implementing the "*connect*" method, I suggest to request data from the device and see if it responds. If this operation succeeds, it means that the device is correctly connected, and in this case, the 1 value will be returned (see, for example, the “nox405.py”, ” ac32.py”, “bh1750.py”).
7. The device class must have the method "*sample(self)*" that must return a string containing the last measurements separated by a semicolon mark. An example of a string returned by this method could be: *“27,76;0,1234;456,9”*. It is mandatory that the fields contained in the string must be in the same number as the ones set in the *DEVICE_SENSORS* string. For example: if *DEVICE_SENSORS=“no2[ppb];no[ppb];nox[ppb]”* (see *“nox405.py”*), then the "*sample(self)*" method must return *“27,76;0,1234;456,9".*