CAPTOR

Collective Awareness Platform for Tropospheric Ozone Pollution

Work package	WP2
Deliverable number	D2.2
Deliverable title	Release of electrochemical board design (a).M9
Deliverable type	R
Dissemination level	PU (Public)
Estimated delivery date	M9
Actual delivery date	15/10/2016
Actual delivery date after EC review	15/09/2017
Version	2.0
Comments	

Authors
Santiago Rodrigo-Muñoz, Oscar Trullols, Jose M. Barcelo-Ordinas, Jorge Garcia-Vidal (UPC)
Kun Mean Hou, Hongling Shi, Jianjin Li and Xunxing Diao (UCA)

Document History			
Version	**Date**	**Contributor**	**Comments**
V0.1	09/09/2016	Santiago Rodrigo-Muñoz, Oscar Trullols, Jose M. Barcelo-Ordinas, Jorge Garcia-Vidal (UPC)	Outline of contents. Captor description.
V0.2	14/09/16	Kun Mean Hou, Hongling Shi, Xunxing Diao, Jianjin Li (UCA)	Raptor description
V0.3	11/10/16	Roger Pueyo (guifi)	Peer review
V1.0	15/10/16	Jose M. Barcelo-Ordinas, Jorge García-Vidal (UPC)	Final version
V2.0	15/09/2017	Jose M. Barcelo-Ordinas, Jorge Garcia-Vidal, Kun Mean Hou (UCA)	Final version after EC review

Table of Contents

List of Figures

List of Listings

List of Abbreviations

AC	Alternating current
API	Application Programming Interface
DIY	Do It Yourself
GUI	Graphical User Interface
LAN	Local Area Network
mAh	mili-Ampere-hour
NO2	Nitrogen Dioxide
O3	Ozone
QoS	Quality of Service
QR code	Quick Response code
PVC	PolyVinyl Chloride
Vcc	IC power-supply pin

Executive Summary

Description of the work

This deliverable describes how to build the nodes we use for monitoring purposes following the DIY (Do It Yourself) philosophy. Each node is composed by a communications subsystem and a sensing subsystem that are packed in a waterproof box. The main target is that citizens with low knowledge on technology are able to build their own monitoring nodes.

We consider two versions of monitoring nodes:

- **Captors**: are based on the Arduino platform and use Metal Oxide sensing devices. We consider them easier to build, but probably not as accurate or reliable as the other model.
- **Raptors**: are based on the raspberry and use an electrochemical sensing device. The design is optimized for achieving a greater accuracy and reliability, but they need more professional building process.

Objectives

This deliverable covers the following topics and issues:

- Description of the building steps for Captor nodes
- Description of the main components of the Raptor node
- Description of the future work for improvements on both types of nodes

1. The DIY Captor v1.1 node

1.1 Building steps

In this section we describe the building steps for the DIY Captor nodes.

Needed tools:

- Hair dryer, Hot Air Gun or any other source (e.g., lighter) to heat the heat shrink tubes.
- 2cm hole saw with centring drill to making the sensor holes on the waterproof box.
- Insulating tape.
- Soldering iron and welding wire.

Sensor tubes:

We will start building the sensor tubes:

1. Cut the 1cm, 2cm and heat shrinking tubes into 3 parts of 10, 5 and 15cm respectively. You will need 6 tubes (5 for the O3 sensors and 1 more for the Temp + Humid sensor)

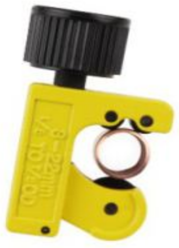


Figure 1. Sensor Tubes

2. Put both PVC tubes inside the heat shrinking tube, the smaller some mm inside the bigger one.

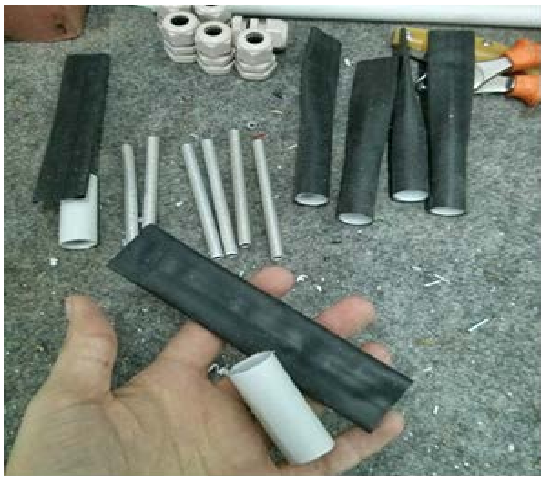


Figure 2. Insert sensor tubes

3. Use the hot air gun or hair dryer to heat the tube and make it shrink to the size of the smaller tube. Stretching the tube while it's still hot will help keeping it tighter.

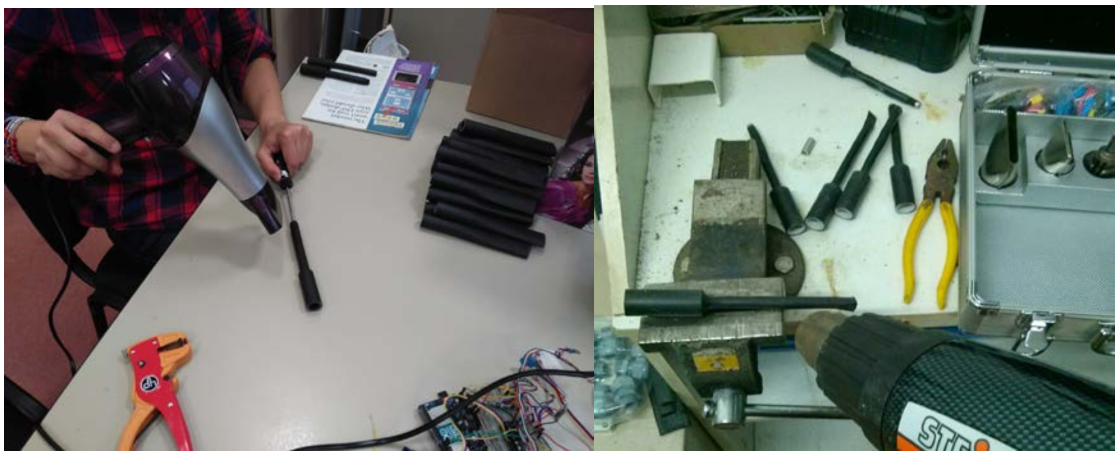


Figure 3. Heat the sensor tubes

4. Put the cable glands on the tubes and the sensor tubes will be finished.

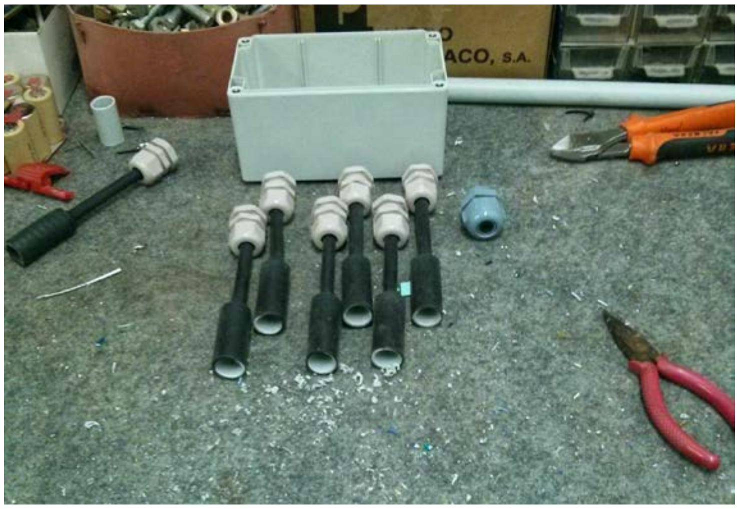


Figure 4. Put the cable glands

5. Our box without the cover is 7cm high. With a pencil and a ruler, make a line at 2cm from the top, and another one at 2.5cm from the bottom. This line is where our holes will be centered. As the holes are 2cm diameter, they will be 1/1.5cm from the top and bottom limits.
6. At the sides we need to keep in mind the screws and the glands' nuts: we will leave a certain margin, e.g. we can place the holes with a distance of 3.3 cm from each other. We need three holes on the upper line and four on the lower line.

7. Use the drill with the 2cm hole saw to make the 7 holes in the box. Also a 0.5 cm hole can be made in the box side to put a LED in it.

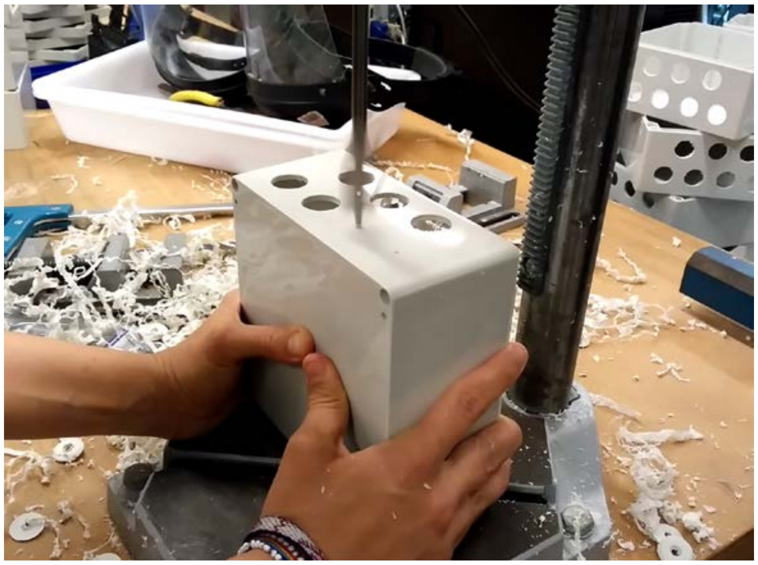


Figure 5. Drill the box

8. Clean the holes and put the 6 sensor tubes on them and the power cable gland, tightening them. To be able of tightening the power cable on the gland, you will need to add some extra insulating tape around the cable so that the cable diameter is incremented. Let's work now on the main course:

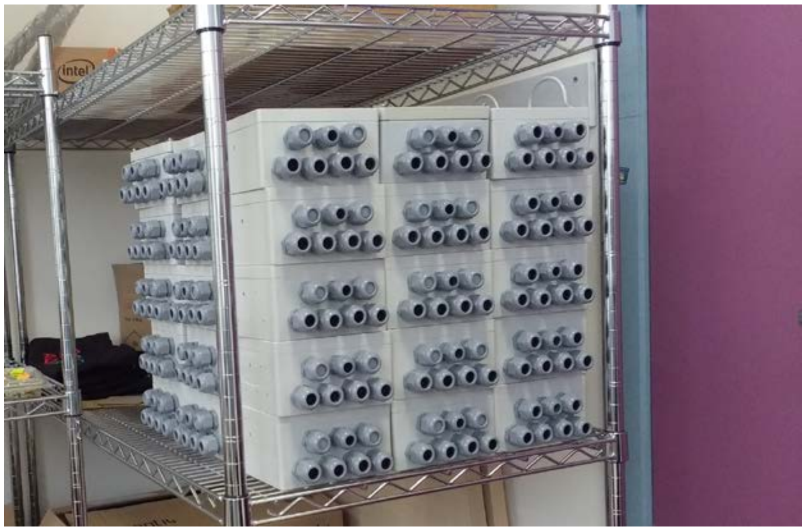


Figure 6. Clean the holes and add the tubes

<u>Soldering and preparing sensor cables:</u>

9. Plug the soldering iron, and while it gets warm, let's get the sensors ready for the soldering.

10. To make the MICS-2610 smaller and fit in the tubes, we will desolder the small pins below it, replacing them with coloured long wires (around 30 cm each).
11. Looking at the sensor as it's shown in the following picture, the bottom and top pins are from the "sensor's heater", while the other 2 pins are the "variable sensor resistance". If you want to follow our color code, we used:

- RED wire at the TOP (Vcc)

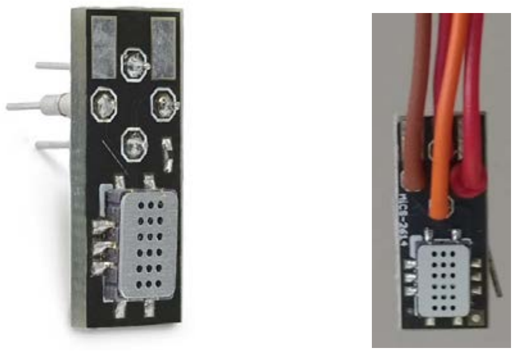


Figure 7. Sensor devices

- ORANGE at the BOTTOM (R_{heater} = 220 Ohms || 220 Ohms || 330 Ohms)
- BROWN at the LEFT (R_{load} = 1kOhm)
- RED at the RIGHT (Vcc)

12. We can insert the sensor with its new wires inside the sensor tubes, keeping them inside the tube so that no damage can occur on them.

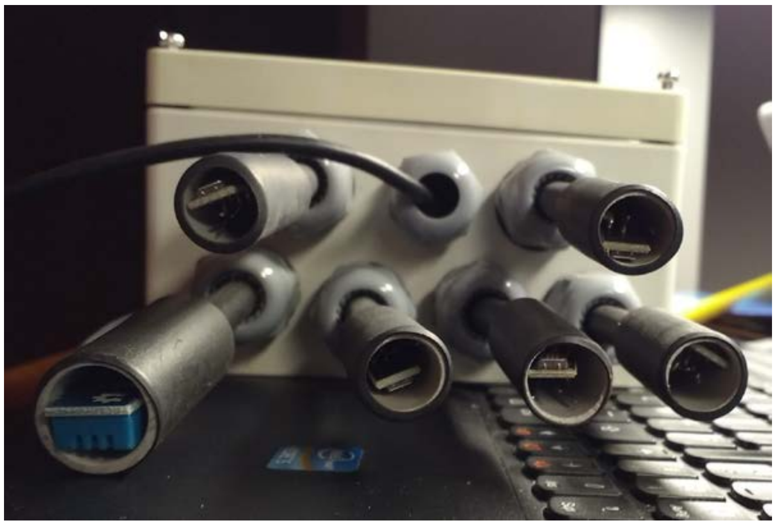


Figure 8. Insert sensors in sensor tubes

13. Respect to the humidity and temperature sensor, we had to remove some leftover shield to make it smaller. The 4-cable wire included with the sensor must be modified in order to be able

to connect it to the circuitry: we will cut out one of the side connectors and left the four separated cables stripped on its first centimeter or so.

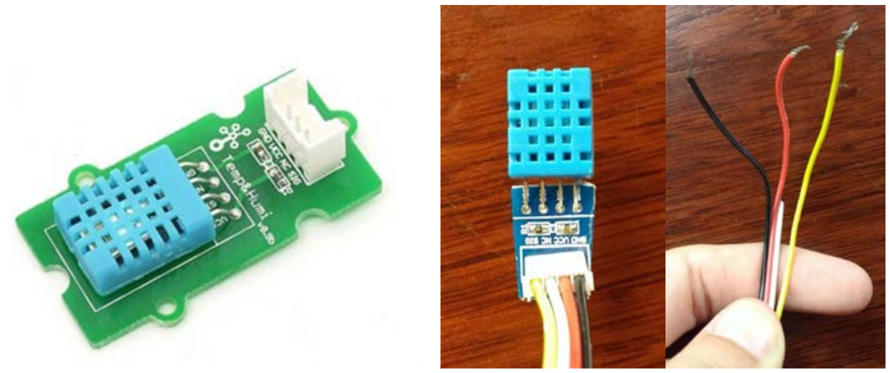


Figure 9. Add wires to the sensors

Breadbord sensor circuit:

14. The next step is to build up the sensor circuitry: for the sake of simplicity, we can simply make it with a small breadboard and common electrical components.

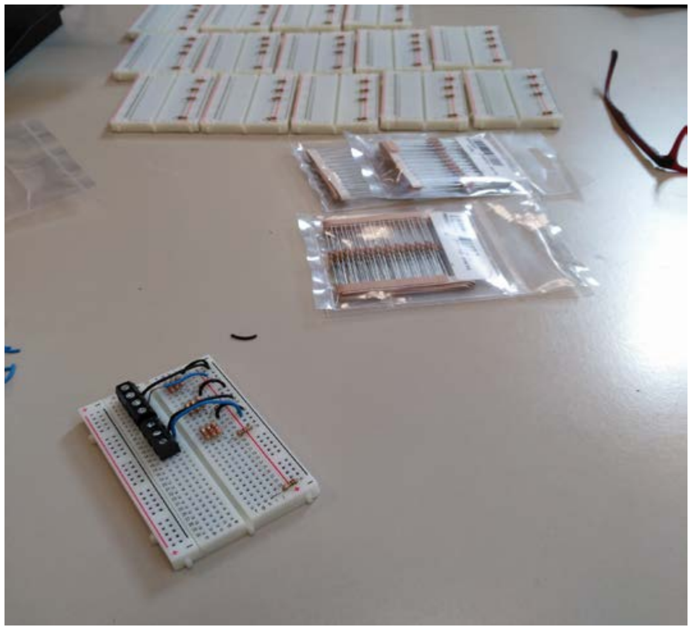


Figure 10. Build the breadboard

15. The breadboard will contain basically a couple of resistances for each sensor, 7 wire-to-breadboard connectors for the Real Time Clock and the Temp and Humi sensor, and the load resistance for the control LED (of course, the LED itself will be fixed in the box hole previously prepared for it, not in the breadboard):

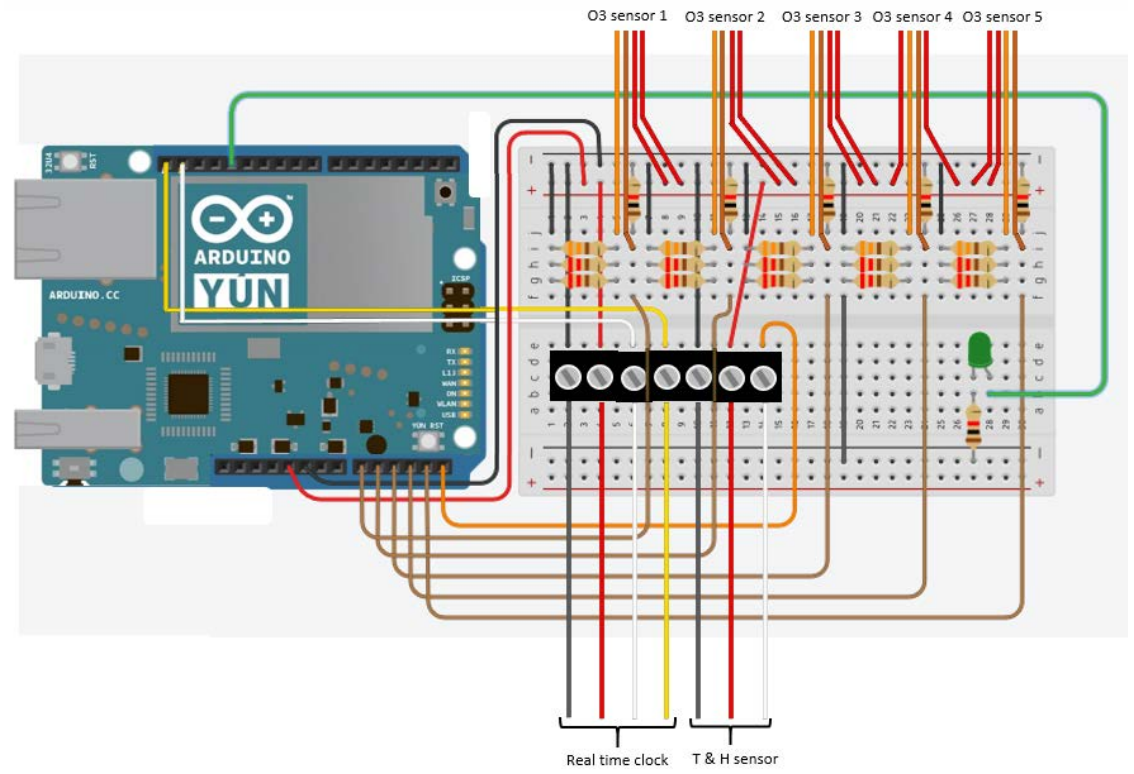


Figure 11. Breadboard circuit

16. One of the RTC 4-cable wire connector must be also cut out and stripped as we have done with the Temp and Humi sensor in order to connect it to the Arduino Yun through the wire-to-breadboard connectors. We will connect also the Temp and Humi sensor, but in this case the white wire can be ignored (just leave it disconnected without stripping it).
17. The Arduino Yun can be now connected to the circuit board, as shown in the schema presented before, and to the power cable.
18. The CR1225 battery can be placed in the RT clock.
19. The Hardware is ready. Now let's go for the software…

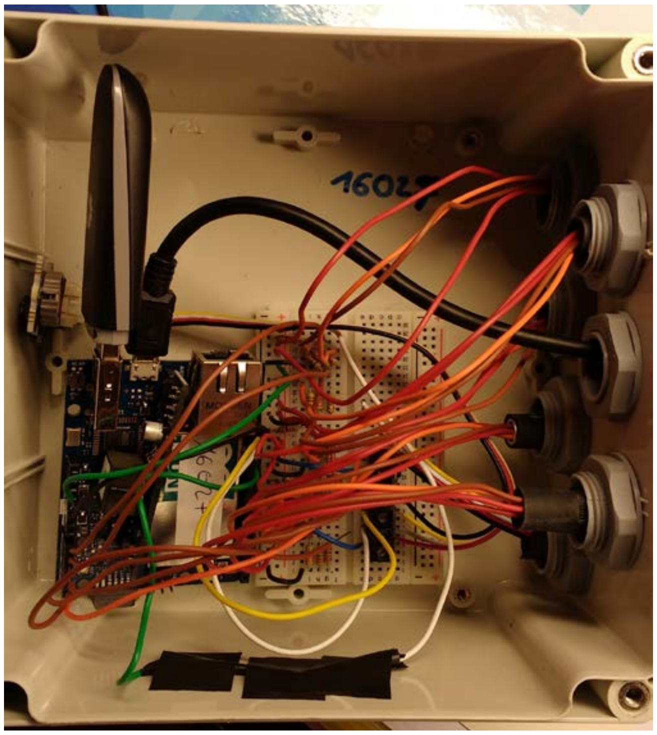
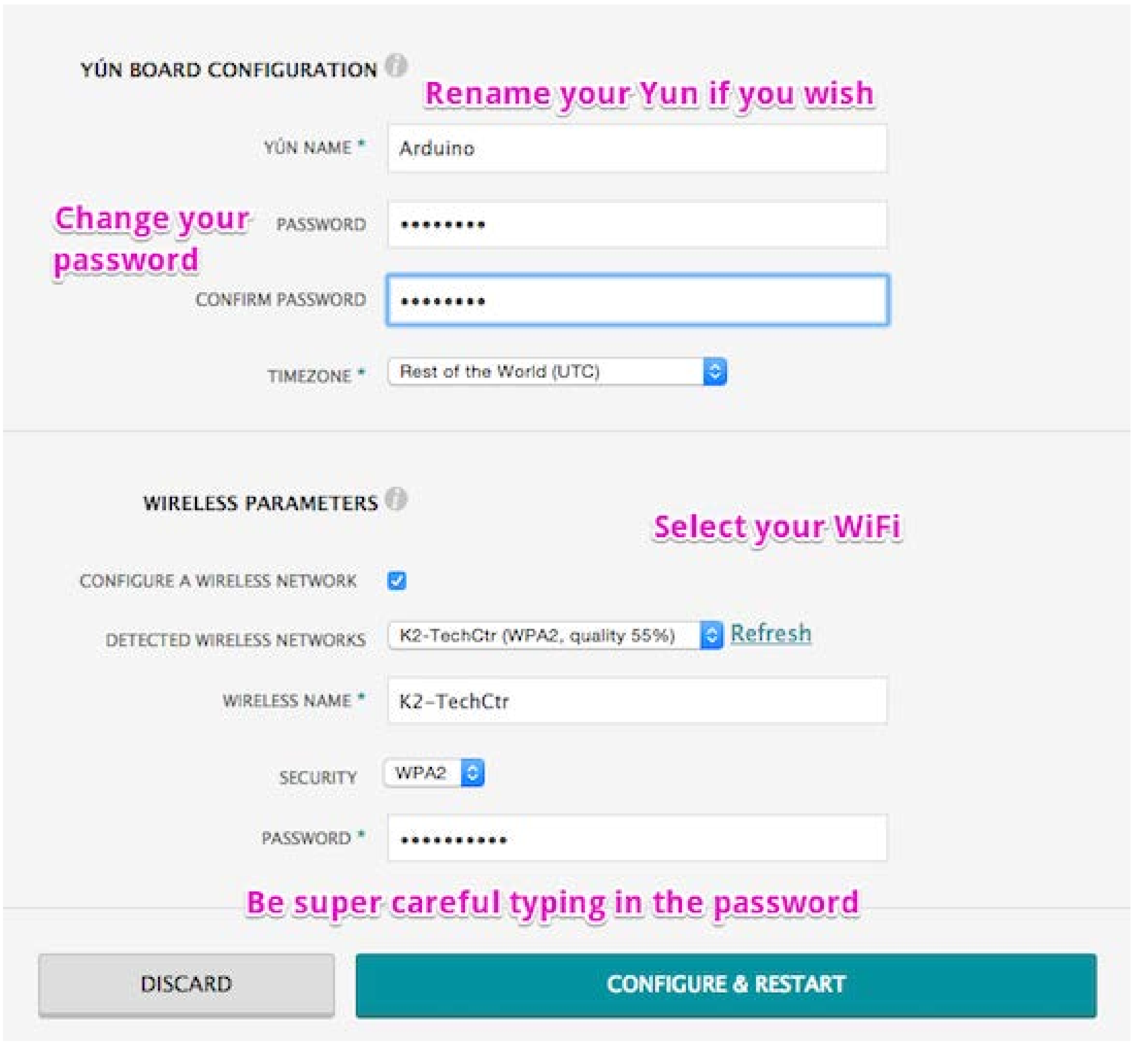


Figure 12. Box with breadboard

1.2 Programming the Arduino Yún

20. First of all, we should check that the firmware installed in the Arduino Yún board is the last available. To do so, go to https://www.arduino.cc/en/Tutorial/YunSysupgrade and follow the instructions, after connecting to the Wi-Fi generated by the Arduino and browsing to 192.168.240.1 (default password is ‘doghunter’ or ‘arduino’).
21. Once the Arduino has been upgraded, you can name it (e.g. captorYYSSS, where YY is the last two ciphers of the year of installation and SSS represent the last ciphers of the serial number of the CAPTOR) and set the password.
22. The Arduino can be configured with the WiFi which it will be connected to.



YÚN BOARD CONFIGURATION

Rename your Yun if you wish

YÚN NAME * Arduino

Change your password

PASSWORD ••••••••

CONFIRM PASSWORD ••••••••

TIMEZONE * Rest of the World (UTC)

WIRELESS PARAMETERS

Select your WiFi

CONFIGURE A WIRELESS NETWORK ✓

DETECTED WIRELESS NETWORKS K2-TechCtr (WPA2, quality 55%) Refresh

WIRELESS NAME * K2-TechCtr

SECURITY WPA2

PASSWORD * ••••••••••

Be super careful typing in the password

Figure 13. Arduino Yun configuration.

23. As the Linux default system has very little amount of disk space, it is recommended to follow the tutorial on (https://www.arduino.cc/en/Tutorial/ExpandingYunDiskSpace) ExpandingYunDiskSpace to partition the SD card on the Arduino and use one of the partitions as the root system of the Arduino.
24. Using the 'upload_captor.sh' script provided below, we will be able of installing everything we need to have the CAPTOR system on the Arduino Yún board. In a few words, this script copies the different executable files, installs the python packages needed and configures some system parameters. Its parameters are the captor serial numbers (in the form YYSSS commented before) to be configured (they shall be connected to the same network to which our computer is connected). All the files needed can be requested at SANS group, AC Department, UPC ({srodrigo,joseb,jorge}@ac.upc.edu).

A complete listing of the code can be found in the appendix.

2. The Raptor platform

In this section we describe the main components of the DIY Raptor nodes and first calibration results.

The Raptor Platform prototype contains tree main devices: Raptor node or Raptor end-device, Multisupport Raptor Local server and Raptor remote server.

2.1 Raptor devices

2.1.1 Raptor node

Raptor node hardware:

The Raptor node hardware is based on the uSu-Edu equipped with Alphasense electrochemical sensors: O3 and NO2 and an air temperature sensor. The Raptor node is designed to be powered by a standard battery and outdoor deployment. Therefore, to minimize energy consumption the sensory data is sent to the multisupport Raptor local server through the IEEE802.15.4 (ZigBee) wireless access medium. The lifetime of the current Raptor node prototype is 3 months when it's powered by a 9V 4000mAh battery (Figure 1).

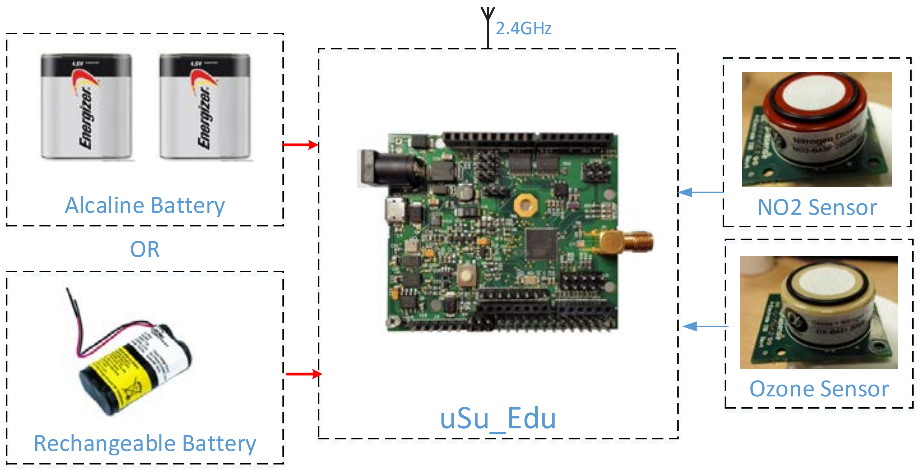


Figure 14. RAPTOR end-device hardware

The key features of uSu-Edu are (Figure 2):

- IEEE802.15.4 wireless Access medium
- 1 3-axis accelerometer
- 1 3-axis gyroscope
- 1 3-axis compass
- 1 barometric pressure
- 1 air Temperature Sensor
- 1 light Sensor
- 1 USB UART port
- 1 Extend port enables to connect with different Arduino Shield
- 1 port enables to directly connect with Raspberry Pi.

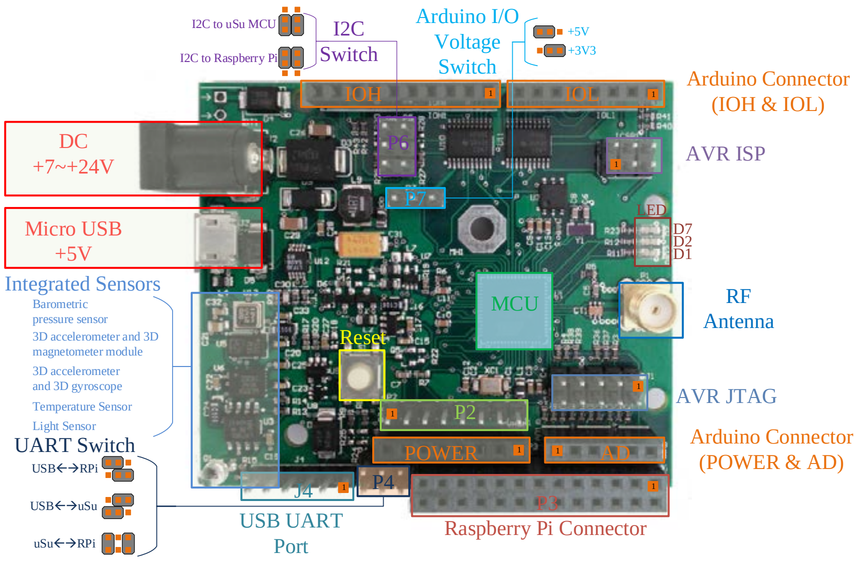


Figure 15. Different components of uSu-Edu board

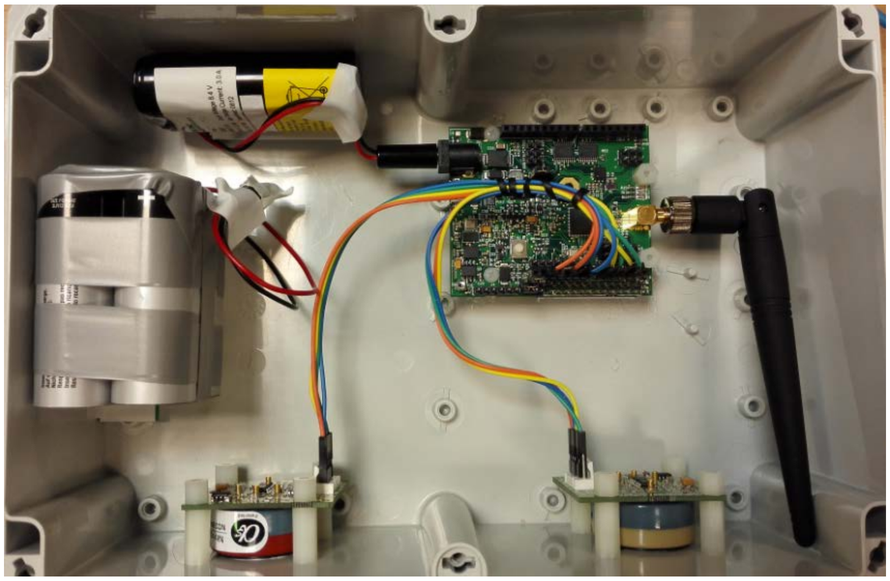


Figure 16.. Raptor node equipped with two Alphasense sensors: O3 and NO2

Raptor node software

The uSu-Edu board adopts the ATMEL wireless protocol stack firmware for the wireless communication with the multisupport Raptor local server. Moreover, SMIR firmware is also (e.g., driver) used to provide an autonomous Raptor node to fulfil environmental data collection and remote reconfiguration.

2.1.2 Multisupport Raptor Local server

Multisupport Raptor Local server hardware

To meet the different requirements of the different partners of CAPTOR project, it's important to implement a multisupport Raptor local server to connect Raptor nodes to internet. The multisupport Raptor local server node is implemented by using Rapberry Pi (pi2) and uSu-Edu boards. This design is modular and enables to adapt to the multisupport Raptor local server equipped with (Figure 4):

- o WiFi,
- o IEEE802.15.4 (ZigBee)
- o Ethernet
- o 3G/4G mobile network.

- The multisupport Raptor local server needs an AC plug.

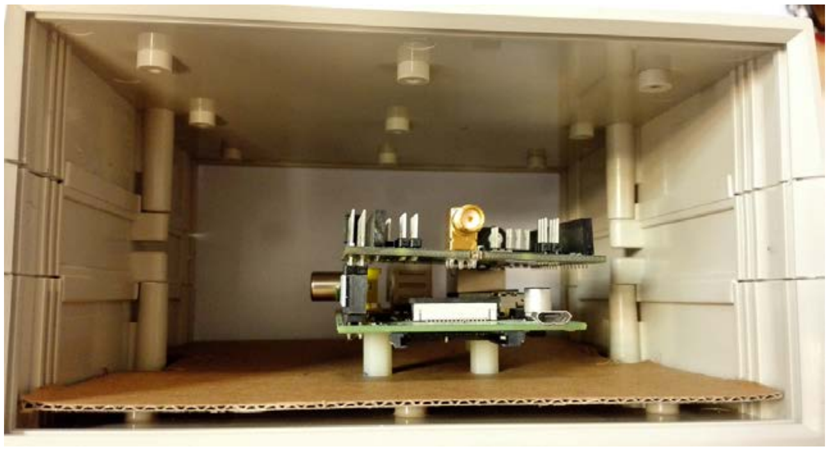


Figure 17. Multisupport Raptor Local server

Multisupport Raptor Local server software

We implement the specific firmware running on the uSu-Edu board to be served as coordinator to setup a star topology network. Each multisupport Raptor server can support 20 Raptor nodes within a range of 200m (low cost antenna). Notice that this range may be increased by using appropriate antenna.

2.1.3 Remote Linux Server

To test and evaluate the performance of the Raptor platform all the sensory data and the QoS of Raptor platform must be analysed. Thus, we implement a remote server having the following functionalities:

- Remote server connection through LAN
- Data and error Log file
- Sensory data display
- Remote Raptor node reconfiguration.

This remote server is based on a PC server running Linux.

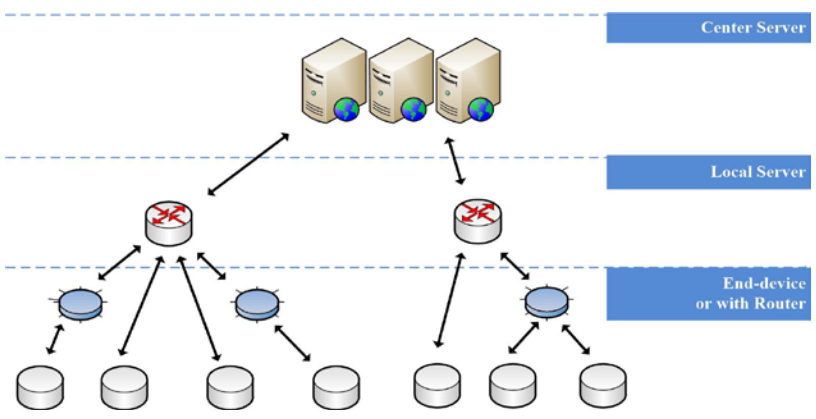


Figure 18. RAPTOR platform architecture

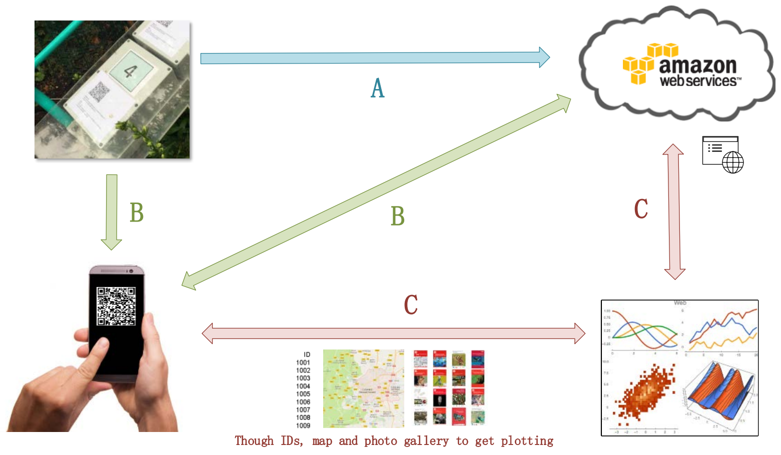


Figure 19. QR code scan to ease the O3 and NO2 data display

<u>Remote server software implementation</u>

Thanks to the data and error log file saved on the remote server, the performances of Raptor platform may be carried out:

- Sensor calibrations
- GUI: wireless sensor node management and Data display etc.
- API for data upload and display
- Smartphone QR code reading for displaying data.

Therefore, the above software is developed on the remote server.

To evaluate the end to end Raptor platform prototype (from sensor to decision support), in collaboration with ATMO Auvergne; two Raptor nodes, one multisupport Raptor local server connected to the remote server through ATMO Auvergne reference station local server were deployed from the 1st to 31st August 2016.

The two Raptor nodes (N° 8 and N° 4, Figure 8) were deployed on the roof of the ATMO Auvergne's reference station located in the Lecoq garden in Clermont-Ferrand (Figure 7).

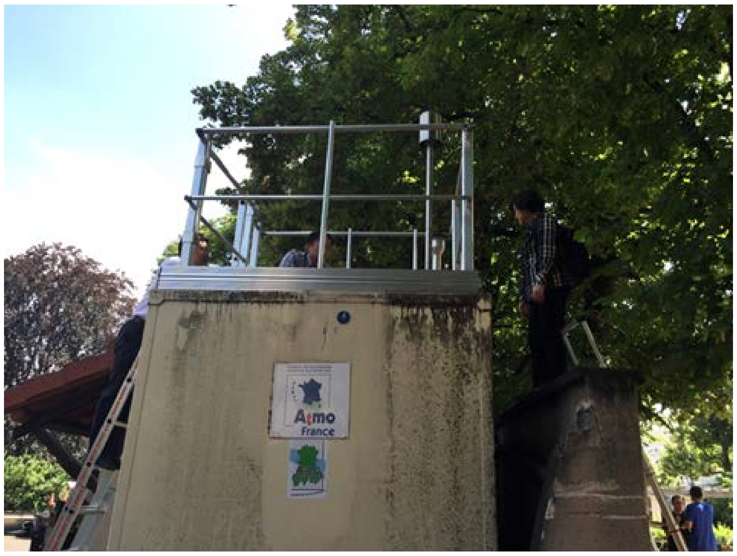


Figure 20. Deployment of Raptor platform in Clermont-Ferrand (Jardin Lecoq)

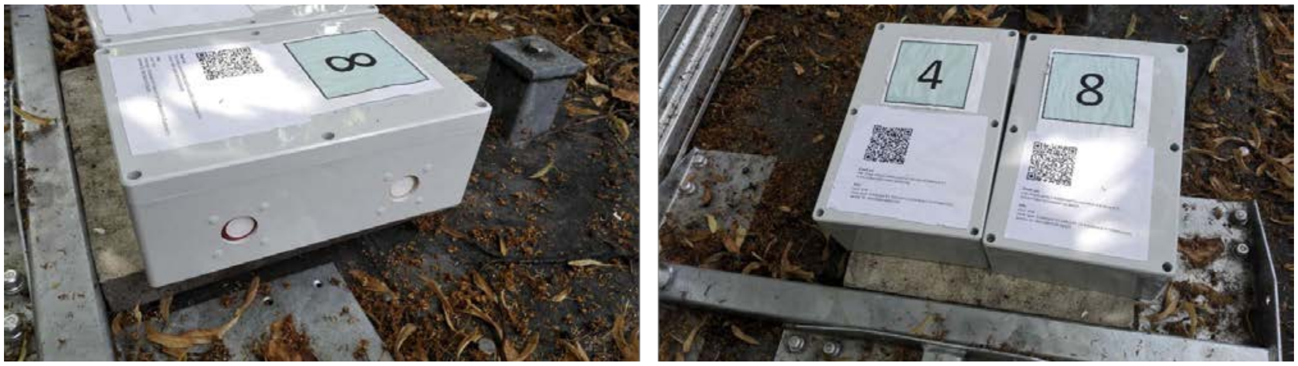


Figure 21. Deployment of Raptor nodes O3 and NO2 with QR code

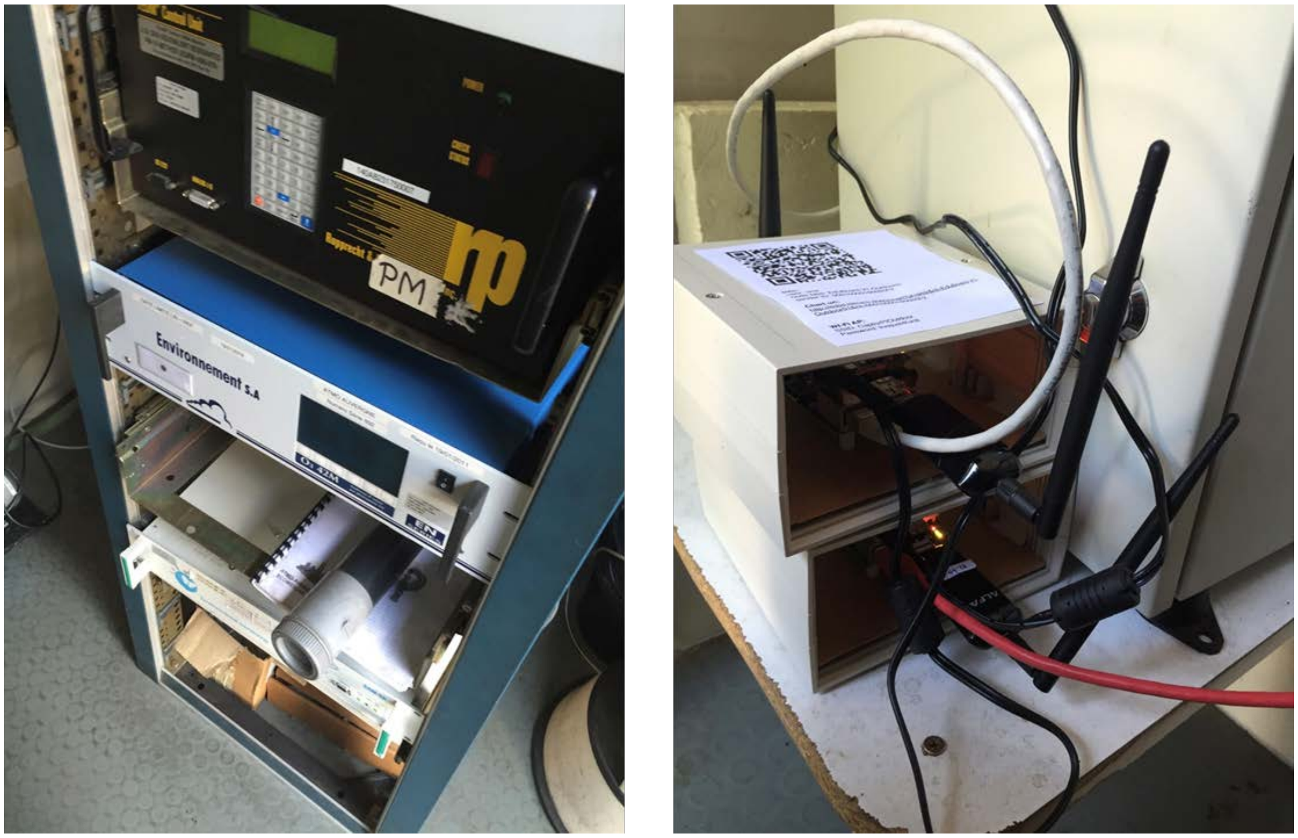


Figure 22. ATMO Auvergne reference station and Multisupport Raptor local server

2.2. First Calibration results

The following figures show first results obtained by the Raptor nodes after adjustment every hour.

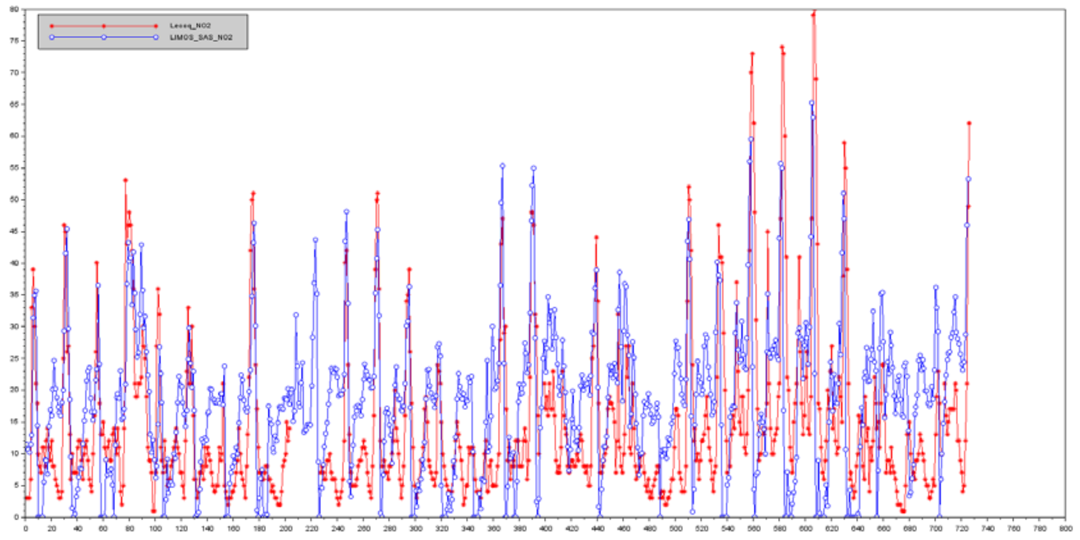


Figure 23. NO2 results of Raptor node N°8 (blue) comparing with the ATMO Auvergne ones (red)

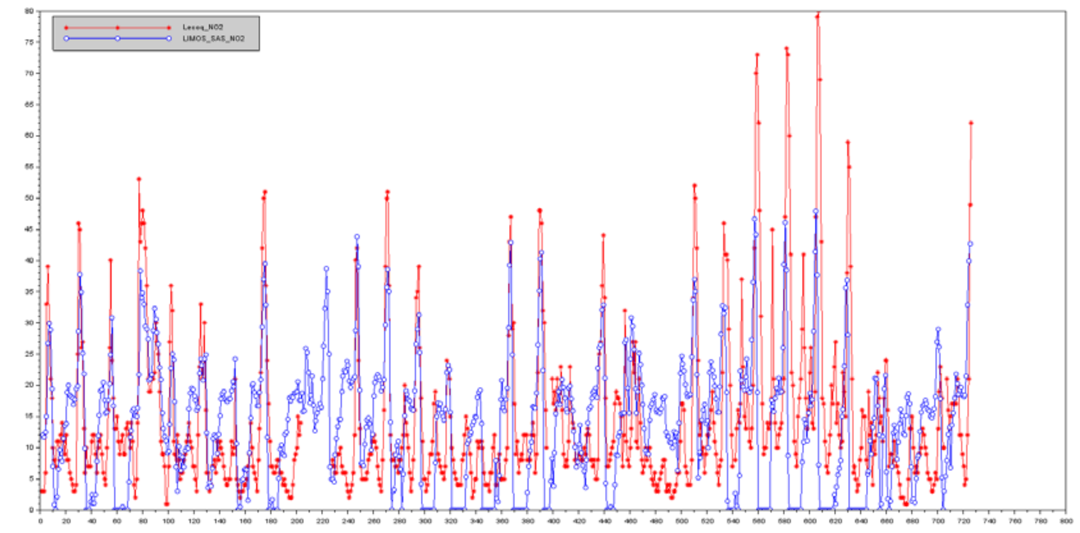


Figure 24. NO2 results of Raptor node N°4 (blue) comparing with the ATMO Auvergne ones (red)

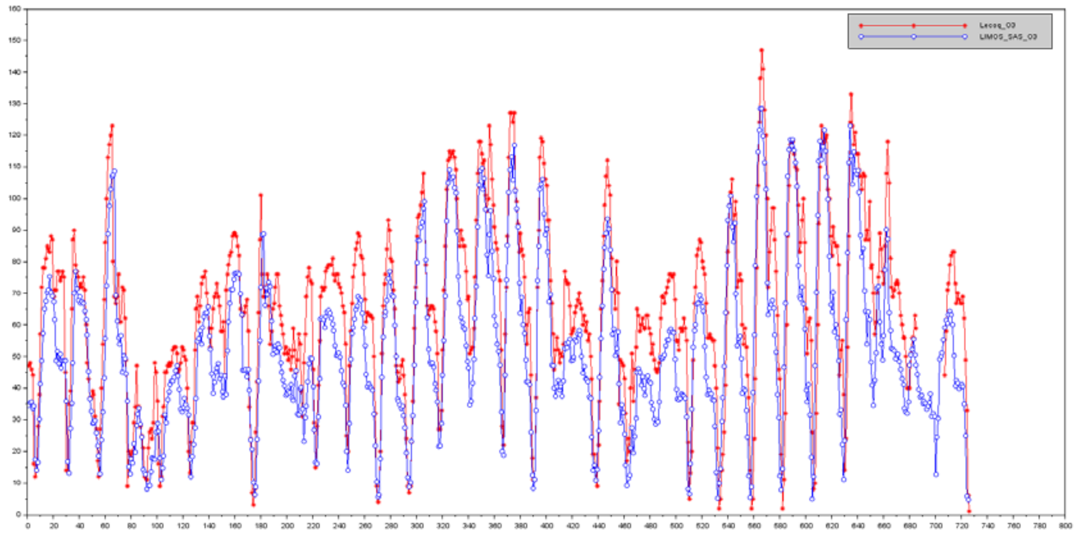


Figure 25. O3 results of Raptor node N°8 (blue) comparing with the ATMO Auvergne ones (red)

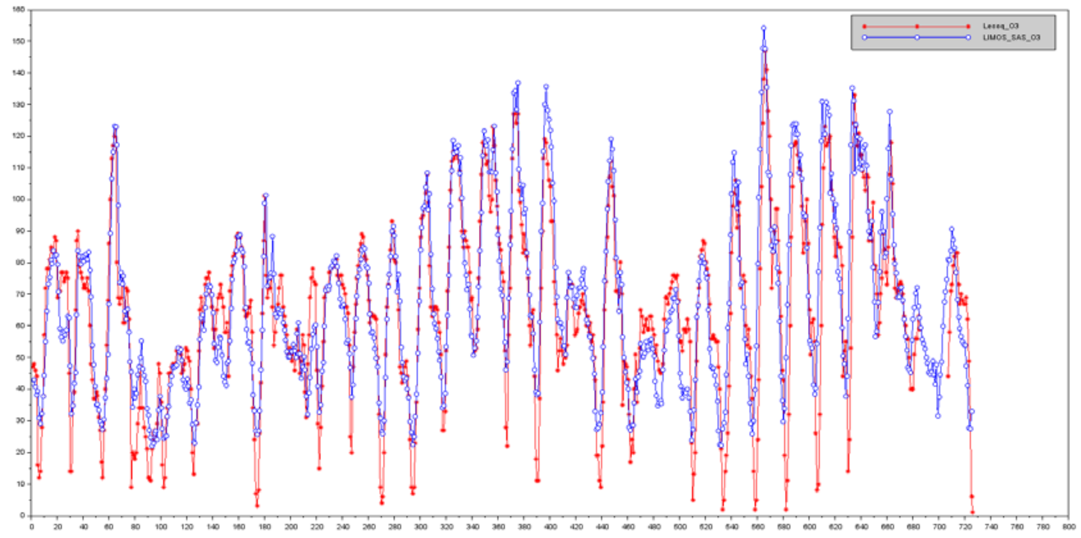


Figure 26. O3 results of Raptor node N°4 (blue) comparing with the ATMO Auvergne ones (red)

The O3 and NO2 field test data after processing are very close to the reference station ones. We conclude that Alphasense O_3 and NO_2 outdoor sensors can be used for O_3 (and NO_2) detection.

3. Future work

3.1 Future work in Captors

Captors have been used in the first testing campaign during summer 2016. After the lessons learned during this experience, we will introduce changes in the design for the next campaigns in summer 2017 and summer 2018.

3.2 Future work in Raptors

The first prototype version of the raptors node has been finished. During the following months we will work in order to have a design ready for the next monitoring campaigns. Some issues we plan to address are the following:

- Remote connection with 3G/4G
- Improve sensor calibrations: O3 and NO2
- Extend the test and validation in collaboration with ATMO Auvergne and ATMO Lyon
- Implementation Alphasense sensor (O3 and NO2) adaptor board:
    - Improvement of energy consumption
    - Improvement of the form factor and packaging of the wireless sensor node
- Remote Server sensor node management:
- Configuration and fail detection

4. Conclusions

In the deliverable we describe the building steps and main components of the captor and raptor nodes. The design of these nodes will be improved during the following nodes to serve as monitoring stations during the campaigns of summer 2017 and 2018.

Appendix A: Ardunio Yun code for the Captor node

```
#!/bin/bash
# Batch upload-to-Arduino script, to be run on a computer connected to the Arduino Yún network
# CAPTOR AC UPC 2016
# @author: srodrigo

if [ $# -eq 3 ]; then
      for i in `seq $1 $2`;
      do
      echo "===================="
      echo $i
      echo "===================="
      sed  's/16000/'$i'/g' captor16000_$3.py > captor$i.py
      sshpass -p "captorcsic" ssh -o StrictHostKeyChecking=no root@captor$i.local '/etc/init.d/captor stop'
      sshpass -p "captorcsic" ssh -o StrictHostKeyChecking=no root@captor$i.local 'mkdir -p /bin/captor'
      sed  's/16000/'$i'/g' scp_to_commsensum.sh > scp_to_commsensum_id.sh
      sshpass -p "captorcsic" scp -r -o StrictHostKeyChecking=no ./scp_to_commsensum_id.sh
root@captor$i.local:/root/scp_to_commsensum.sh
      sshpass -p "captorcsic" scp -r ./auto-3g.sh root@captor$i.local:/root/auto-3g.sh
      sshpass -p "captorcsic" scp -r ./ntplib.py root@captor$i.local:/usr/lib/python2.7/
      sshpass -p "captorcsic" scp -r ./captor root@captor$i.local:/etc/init.d/
      sshpass -p "captorcsic" scp -r ./captor$i.py root@captor$i.local:/bin/captor/captor.py
      sshpass -p "captorcsic" scp -r ./forever.sh root@captor$i.local:/bin/captor/
      sshpass -p "captorcsic" scp -r -o StrictHostKeyChecking=no ./install_captor.sh
root@captor$i.local:/bin/captor/install_captor.sh
      sshpass -p "captorcsic" ssh -o StrictHostKeyChecking=no root@captor$i.local 'chmod 755 /bin/captor/install_captor.sh
&& /bin/captor/install_captor.sh'
      done
else
      echo "Usage: ./upload captor mincaptorid maxcaptorid number-of-sensors"
      echo -e "\tThis program uploads captorv2 image to the Arduino Yun with\n\tID between mincaptorid and maxcaptorid,
both included"
fi
```

Listing 1. Code for installing the CAPTOR system on the Arduino Yún, supposing that the password by default is ‘captorcsic’

```
#!/bin/ash
# Batch install script, to be run on the Arduino Yún
# CAPTOR AC UPC 2016
# @author: srodrigo

# files captor, forever.sh and captor.py must be already copied in /etc/init.d and /bin/captor, respectively

# Installation of required python packages
opkg update
opkg install distribute
opkg install python-openssl
```

```
easy_install pip
pip install requests

# Providing permissions to executable files
chmod 755 /etc/init.d/captor
chmod 755 /bin/captor/captor.py
chmod 755 /bin/captor/forever.sh
chmod 755 /root/scp_to_commsensum.sh
chmod 755 /root/auto-3g.sh

# Generating secret for authentication purposes if needed
mkdir -p ~/.ssh
dropbearkey -t rsa -f ~/.ssh/id_rsa
dropbearkey -y -f ~/.ssh/id_rsa | grep "^ssh-rsa" >> authorized_keys

# Generating crontab with recurrent tasks
echo "0 1 * * * touch /etc/banner && reboot" >> /etc/crontabs/root
# copy logs every day if needed
echo "0 11 * * * touch /etc/banner && /root/scp_to_commsensum.sh" >> /etc/crontabs/root
# execute tasks if there are any
echo "5 11 * * * touch /etc/banner && /root/today_commands.sh" >> /etc/crontabs/root
# 3G connection daemon
echo "10 * * * * touch /etc/banner && /root/auto-3g.sh" >> /etc/crontabs/root

# Enabling CAPTOR daemon
/etc/init.d/cron enable
/etc/init.d/cron restart
/etc/init.d/captor enable
/etc/init.d/captor restart

# 3G required packages installation
opkg install kmod-usb-serial-option kmod-usb-serial-wwan luci-proto-3g usb-modeswitch-data usb-modeswitch
```

Listing 2. Code to be automatically run in the Arduino to complete the preparation of the CAPTOR system

Following with the steps mentioned in section 1.2:

25. To check that everything is OK, log in to the Arduino Yún through ssh and check that the captor process is running by executing the command ‘ps | grep captor’, which should have the following result:
26. If 3G is going to be used, it should be configured as follows:
    a. Connect to the Yun throught ssh
    b. open file /etc/usb_modeswitch.d/12d1\:1f01 and copy the following data

```
captorYYSSS: ~# vim /etc/usb_modeswitch.d/12d1\:1f01
########################################################################################
# Huawei E303
```

```
TargetVendor=  0x12d1
TargetProduct= 0x14dc

MessageContent="55534243123456780000000000000011063000000100010000000000000000"
##############################################################################
```

27. Disconnect (if it was connected) and/or reconnect the USB 3G modem
28. Check it's correctly seen by the Yun by typing:

```
captor3: ~# lsusb
... # Example, some data may vary, except the underlined text
Bus 001 Device 007: ID 12d1:1001 Huawei Technologies Co., Ltd. E169/E620/E800 HSDPA Modem
```

   a. Login into the Yun web interface and navigate to the Advanced configuration panel.
   b. Navigate to the Network–>Interfaces tab. Click on Add new interface
   c. Give the device the name: 3gHuawei
   d. Select the interface protocol as: UMTS/GPRS/EV-DO and click Accept
   e. Select the following values (or the ones corresponding to your provider):
      i. Modem device: /dev/ttyUSB0
      ii. Service Type: UMTS/GPRS
      iii. APN: movistar.es (this depends on the SIM you are using)
      iv. PIN : ????? (this depends on the SIM you are using)
      v. PAP/CHAP username: MOVISTAR
      vi. PAP/CHAP password: MOVISTAR
   f. and click on Save&Apply

29. The rest of the code is provided in the following listings. To solve any doubt, please contact with its author, srodrigo@ac.upc.edu:

```
#!/usr/bin/python
# CAPTOR main process
# CAPTOR AC UPC 2016
# @author: srodrigo
# -*- coding: utf-8 -*-
from subprocess import Popen, PIPE
import subprocess
import datetime
import time
import os
import httplib
import ntplib # Got version 0.3.3 from https://pypi.python.org/pypi/ntplib/
import requests # Installed through pip: (pip install requests)
# To install pip on the Yun:
    # ~:opkg update
    # ~:opkg install distribute
    # ~:opkg install python-openssl
    # ~:easy_install pip
import urllib
import json

# CAPTOR unit name
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
CAPTOR_ID=16000       # CHANGE THIS FOR EVERY CAPTOR!!!!!!!!!!!!!!!!!!!!!!!!!
CAPTOR_NAME="captor%d" %(CAPTOR_ID) # CHANGE THIS FOR EVERY CAPTOR!!!!!!!!!!!!!!
#!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
DEBUG = False
# URL allowing access to the Arduino "server" functions via Bridge
ARDUINO_URL = "localhost/arduino/"
# Arduino's returning date format
DATEFMT = "%Y-%m-%dT%H:%M:%SZ"
# CommSensum date format
DATESTRFMT = "%04d-%02d-%02dT%02d:%02d:%02d"
# Datafile template
FILENAMETEMPLATE = "/mnt/sda1/DATA%04d.txt"
# Data pending to be sent files
FILENAMEPENDING_1 = "/mnt/sda1/pending_1.tmp"
FILENAMEPENDING_2 = "/mnt/sda1/pending_2.tmp"
FILENAMETEMPPENDING = "/mnt/sda1/temppending.tmp"
# Log file
FILENAMELOG = "/mnt/sda1/log.txt"
# Status recovery file
FILENAMESTATUS = "/mnt/sda1/status.txt"
# Time between measures to be sent
SAMPLING_PERIOD = 30

CS_API_KEY = "Rrme1YvMeyi3Ftd69qc4Y2CHk2Yl4L06olUJaWdVVbM"
CS_USER_NAME = "otrullols"
CS_PROJECT_NAME = "CAPTOR"
CS_HTTP_ADDR_1 = "commsensum.pc.ac.upc.edu:3000" # UPC VM
CS_HTTP_ADDR_2 = "51.255.135.164:3000" # DEDICATED SERVER (REPLICATION)
CS_HEADER_CONTENT_TYPE = "application/json"

# Perform a call to an Arduino "server function", returning the output obtained
def arduinoCall(action):
    try:
    p = Popen(["curl", ARDUINO_URL + action], stdin=PIPE, stdout=PIPE, stderr=PIPE)
    output, err = p.communicate()
    rc = p.returncode
    if rc != 0: # Something went bad in the curl process
        return 1
    output = output.translate(None, "\r\n")
    return output
    except Exception as e:
    print "Something went wrong with Arduino call: "
    print e
    print e.message
    return 1

def createSDFileAsTemplate(template):
    i = 0
    while os.path.exists(template %(i)):
    i += 1
    filename = template %(i)
    return filename

def storeMeasurements(dtime,s1,s2,s3,temp,humid,filename):
    try:
    datafile = open(filename,"a")
    line = "%s\t%.04f\t%.04f\t%.04f\t%.02f\t%.02f\n" %(dtime,s1,s2,s3,temp,humid)
    datafile.write(line)
    datafile.close()
    return 0
    except Exception as e:
    print "Couldn't store measurements: "
    print e
    print e.message
    return 1

def sendPending(filename, url):
    try:
    pendingfile = open(filename, "r")
    lines = pendingfile.readlines()
    pendingfile.close()
    pendingfile = open(FILENAMETEMPPENDING, "w")
    error = 0
    for l in lines:
        err = popPending(url, l)
        if err != 0: # We have not been able of sending it
        error = error + 1
        if err != 2: # The line is correctly formatted
            pendingfile.write(l)
    pendingfile.close()
    os.remove(filename)
    os.rename(FILENAMETEMPPENDING, filename)
    return error
    except Exception as e:
    print "Error when send pending data: "
    print e
    print e.message
    return -1

def popPending(url, line):
    try:
    data = line.split(";") # Date;Ozone1;Ozone2;Ozone3;Temp;Humidity
    if len(data) < 6:
        return 2 # Parsing error
    dtime = data[0]
    o31 = float(data[1])
    o32 = float(data[2])
    o33 = float(data[3])
    temp = float(data[4])
    hum = float(data[5])
    return sendData(url, dtime, o31, o32, o33, temp, hum) # Redirect error code
    except Exception as e:
    print "Found some corrupted pending data: "
    print e
    print e.message
    return 2 # Parsing error

def pushPending(filename, dtime, s1, s2, s3, t, h):
    try:
    line = "%s;%f;%f;%f;%f;%f\n" %(dtime, s1, s2, s3, t, h)
    pendingfile = open(filename,"a")
    pendingfile.write(line)
    pendingfile.close()
    return 0
    except Exception as e:
    print "Couldn't store pending data: "
    print e
    print e.message
    return 1

def internet_available(url):
    conn = httplib.HTTPConnection(url)
    try:
    conn.request("HEAD", "/")
    return True
    except Exception as e:
    return False

def sendData(url, dtime, s1, s2, s3, t, h):
    headers = {"user":CS_USER_NAME ,"X-ApiKey": CS_API_KEY, "Content-type":CS_HEADER_CONTENT_TYPE}

    # Sensor 1
    dataJSON = createBody("O3r",dtime,"%.04f"%(s1),"KOhms")
    r = makePost(createUrl(url,CAPTOR_NAME+"01"),dataJSON,headers)
    if r != 0: # Retry
    r = makePost(createUrl(url,CAPTOR_NAME+"01"),dataJSON,headers)
    if r != 0: # Error
    return r

    # Sensor 2
    dataJSON = createBody("O3r",dtime,"%.04f"%(s2),"KOhms")
    r = makePost(createUrl(url,CAPTOR_NAME+"02"),dataJSON,headers)
    if r != 0: # Retry
    r = makePost(createUrl(url,CAPTOR_NAME+"02"),dataJSON,headers)
    if r != 0: # Error
    return r

    # Sensor 3
    dataJSON = createBody("O3r",dtime,"%.04f"%(s3),"KOhms")
    r = makePost(createUrl(url,CAPTOR_NAME+"03"),dataJSON,headers)
    if r != 0: # Retry
    r = makePost(createUrl(url,CAPTOR_NAME+"03"),dataJSON,headers)
    if r != 0: # Error
    return r

    # Temperature
    dataJSON = createBody("temperature",dtime,"%.02f"%(t),"ºC")
    r = makePost(createUrl(url,CAPTOR_NAME+"temp"),dataJSON,headers)
    if r != 0: # Retry
    r = makePost(createUrl(url,CAPTOR_NAME+"temp"),dataJSON,headers)
    if r != 0: # Error
    return r

    # Humidity
    dataJSON = createBody("humidity",dtime,"%.02f"%(h),"%")
    r = makePost(createUrl(url,CAPTOR_NAME+"humi"),dataJSON,headers)
    if r != 0: # Retry
    r = makePost(createUrl(url,CAPTOR_NAME+"humi"),dataJSON,headers)
    if r != 0: # Error
    return r

    return 0

def createBody(magnitude, dtime, value, unit):
    r = json.dumps({"date":dtime, "magnitude":magnitude, "value":value, "unit":unit})
    r = r.encode('utf-8')
    return r

def createUrl(url,stream):
    updatedUrl = "http://%(hostname)s/v1/%(project)s/%(sname)s" % {"hostname":url, "project":CS_PROJECT_NAME,
"sname":stream}
    if DEBUG == True:
    print "Url created: ", updatedUrl
    return updatedUrl

def makePost(url, dataJSON, headers):
    if DEBUG == True:
    print url, "-",dataJSON,"-",headers
    try:
    response = requests.post(url, data=dataJSON, headers=headers)

    if response.status_code!=requests.codes.ok:
        logfile.write(response)
        logfile.write(response.text)
        return 1
    else:
        logfile.write("OK\n")
        return 0

    except Exception as e:
    print "Exception on POST: "
    print e
    print e.message
    return 1

# Opening/creating log file
logfile = open(FILENAMELOG, "a")
logfile.write("===============================\n")
logfile.write("Hello there. Here we are again.\n")
logfile.write("===============================\n")
# Let's begin with some blinks
r = arduinoCall("blink/5")
if r == 1:
    logfile.write("Couldn't perform init blink\n")
    subprocess.call(["reset-mcu"])
    time.sleep(15)
# Now we will create the sequentially numbered datafile
datafile = createSDFileAsTemplate(FILENAMETEMPLATE)
logfile.write("We will write into datafile %s\n" % (datafile))
# Creating (if not existing already) pending file
pendingfile = open(FILENAMEPENDING_1, "a")
pendingfile.close()
pendingfile = open(FILENAMEPENDING_1, "r")
pending_1 = len(pendingfile.readlines())
pendingfile.close()

pendingfile = open(FILENAMEPENDING_2, "a")
pendingfile.close()
pendingfile = open(FILENAMEPENDING_2, "r")
pending_2 = len(pendingfile.readlines())
pendingfile.close()
logfile.write("Pending values are: %d (UPC VM) %d (Dedicated)\n" %(pending_1, pending_2))
logfile.close()

logfile = open(FILENAMELOG, "a")
# Compare with NTP hour and refresh RTC if possible
if internet_available("0.openwrt.pool.ntp.org"):
    try:
    ntpclient = ntplib.NTPClient()
    ntpd = datetime.datetime.utcfromtimestamp(ntpclient.request('0.openwrt.pool.ntp.org').tx_time)
    logfile.write("NTP server says it is now %s\n" % (ntpd))
    try:
        output = arduinoCall("setDate/%d/%d/%d/%d/%d/%d" %(ntpd.year, ntpd.month, ntpd.day, ntpd.hour, ntpd.minute,
ntpd.second))
        if output == 1:
        logfile.write("Couldn't set date\n")
        subprocess.call(["reset-mcu"])
        time.sleep(15)
        else:
        try:
            rtcd = datetime.datetime.strptime(output,DATEFMT)
            logfile.write("RTC set correctly to %s\n" % (rtcd))
        except Exception as e:
            print "No RTC available: "
            subprocess.call(["reset-mcu"])
            time.sleep(15)
            print e
            print e.message
        except NameError:
            print "Bad ntp datetime"

        except Exception as e:
        print "Exception when syncing with NTP: "
        print e
        print e.message

# We will now initialize the measure variables
meantemp = 0
meanhumid = 0
meano31 = 0
meano32 = 0
meano33 = 0
countmeasures = 0
stacksend = []

logfile.close()

first_measure = True # Send always the first measure to test connection
# Main loop
while (True):
        logfile = open(FILENAMELOG, "a")
        initsec = time.time()
        if pending_1 > 0 and pending_2 > 0:
        r = arduinoCall("blink/1")
        if r == 1:
            logfile.write("Couldn't perform loop blink\n")
            subprocess.call(["reset-mcu"])
            time.sleep(15)
        else:
        r = arduinoCall("blink/3")
        if r == 1:
            logfile.write("Couldn't perform loop blink\n")
            subprocess.call(["reset-mcu"])
            time.sleep(15)

        # Let's get Date and Time
        # Bad clock datetime: 2165-165-165T165:165:85Z

        dtime = arduinoCall("getDate")
        if dtime == 1:
        logfile.write("Couldn't get date from RTC\n")
        subprocess.call(["reset-mcu"])
        time.sleep(15)
        rtcd = datetime.datetime.now()
        else:
        try:
            rtcd = datetime.datetime.strptime(dtime,DATEFMT)
        except ValueError:
            rtcd = datetime.datetime.now()
```

```
logfile.write("Now it is %s\n" % (rtcd))
output = arduinoCall("getSensors")
if output == 1:
logfile.write("Couldn't get sensors data\n")
subprocess.call(["reset-mcu"])
time.sleep(15)
else:
try:
    [temp, humid, o31, o32, o33] = output.split("\t")
    if DEBUG == True:
    logfile.write("Sensor 1 value is " + o31 + "\n")
    logfile.write("Sensor 2 value is " + o32 + "\n")
    logfile.write("Sensor 3 value is " + o33 + "\n")
    logfile.write("Temp value is " + temp + "\n")
    logfile.write("Humidity value is " + humid + "\n")
    meano31 += float(o31)
    meano32 += float(o32)
    meano33 += float(o33)
    meantemp += float(temp)
    meanhumid += float(humid)
    countmeasures += 1
    print stacksend
    # Let's see if it's time to upload data
    # Each captor uploads data in a ID related instant of time, each 30 min
    if rtcd.minute%SAMPLING_PERIOD == 0 or countmeasures >= SAMPLING_PERIOD or first_measure == True:
    if countmeasures != 1 or first_measure == True:
        logfile.write("It is time to take measures\n")
        meano31 /= countmeasures
        meano32 /= countmeasures
        meano33 /= countmeasures
        meantemp /= countmeasures
        meanhumid /= countmeasures
        rtcstr = DATESTRFMT %(rtcd.year, rtcd.month, rtcd.day, rtcd.hour, rtcd.minute, rtcd.second)
        r = storeMeasurements(rtcstr,meano31,meano32,meano33,meantemp,meanhumid,datafile)
        listtemp = []
        print stacksend
        listtemp.append(meanhumid)
        listtemp.append(meantemp)
        listtemp.append(meano33)
        listtemp.append(meano32)
        listtemp.append(meano31)
        listtemp.append(rtcstr)
        stacksend.append(listtemp)
        print stacksend
        if r != 0:
        logfile.write("Couldn't store measurements\n")

    meantemp = 0
    meanhumid = 0
    meano31 = 0
    meano32 = 0
    meano33 = 0
    countmeasures = 0
```

```
        if ((rtcd.minute - (CAPTOR_ID%100))%SAMPLING_PERIOD == 0 or first_measure == True) and stacksend != []:
        first_measure = False
        logfile.write("It is time to communicate\n")
        for measure in stacksend:
            dt = measure.pop()
            ozone1 = measure.pop()
            ozone2 = measure.pop()
            ozone3 = measure.pop()
            te = measure.pop()
            hu = measure.pop()
            # CS_HTTP_ADDR_1 (UPC VM)
            err = sendData(CS_HTTP_ADDR_1, dt, ozone1, ozone2, ozone3, te, hu)

            if (err != 0):
            r = pushPending(FILENAMEPENDING_1, dt, ozone1, ozone2, ozone3, te, hu)
            if r != 0:
                logfile.write("Couldn't store pending data (UPC VM)\n")
            else:
            temp = sendPending(FILENAMEPENDING_1,CS_HTTP_ADDR_1)
            if temp != -1:
                pending_1 = temp

            # CS_HTTP_ADDR_2 (DEDICATED)
            err = sendData(CS_HTTP_ADDR_2, dt, ozone1, ozone2, ozone3, te, hu)

            if (err != 0):
            r = pushPending(FILENAMEPENDING_2, dt, ozone1, ozone2, ozone3, te, hu)
            if r != 0:
                logfile.write("Couldn't store pending data (DEDICATED)\n")
            else:
            temp = sendPending(FILENAMEPENDING_2,CS_HTTP_ADDR_2)
            if temp != -1:
                pending_1 = temp

        stacksend = []
    except Exception as e:
        logfile.write("Error when taking data\n")
        subprocess.call(["reset-mcu"])
        time.sleep(15)
        print "No data was token: "
        print e
        print e.message

    #We will sleep for the rest of the minute
    sleeptime = 60-(time.time()-initsec)
    if sleeptime < 0:
    sleeptime = 0
    time.sleep(sleeptime)
    logfile.close()
```

Listing 3. Code of the main CAPTOR process, in this case for three sensors onboard

```
#!/bin/sh
# 3G connectivity maintenance daemon
# CAPTOR AC UPC 2016
# @author: srodrigo

# This script checks the 3G connectivity and tries to recover it in case it is
# not working properly. If so, it saves the logread in a SD card file
# (for debug purposes). Always logs the status and the actions performed.

# Let's first obtain the date and hour right now
DATE=`date +%Y-%m-%d\|%H\:%M\:%S`

# Now we are going to check for the "logs" folder
# Due to possible problems in the Linino with mounting devices (the 3G USB
# sometimes messes up with the SD), we will first look for any SD mounted

ls /mnt/sda1 > /dev/null

if [ $? -ne 0 ]
then # There is no sd, let's save log in /root
   LOGFILE="/root/3G-connection-log.txt"
else # SD is mounted on /mnt/sda1
   LOGFILE="/mnt/sda1/3G-connection-log.txt"
fi

# Printing a general connectivity status message
echo "" >> $LOGFILE
echo "" >> $LOGFILE
echo "\||||||||||||||||||||||||||||||||||||||/" >> $LOGFILE
echo "-       3G CONNECTION LOG        -" >> $LOGFILE
echo -n "-   DATE: " >> $LOGFILE
echo -n "$DATE" >> $LOGFILE
echo "         -" >> $LOGFILE
echo "/||||||||||||||||||||||||||||||||||||||\\" >> $LOGFILE
echo "" >> $LOGFILE
echo "===============" >> $LOGFILE
echo "IFCONFIG STATUS" >> $LOGFILE
echo "===============" >> $LOGFILE
ifconfig >> $LOGFILE
echo "" >> $LOGFILE
echo "======" >> $LOGFILE
echo "ROUTES" >> $LOGFILE
echo "======" >> $LOGFILE
route -n >> $LOGFILE

# Let's check the connectivity to Internet
echo "" >> $LOGFILE
echo "==========" >> $LOGFILE
echo "PING CHECK" >> $LOGFILE
echo "==========" >> $LOGFILE
ping -q -w 10 www.google.es >> $LOGFILE
if [ $? -ne 0 ]
then
```

```
   echo "" >> $LOGFILE
   echo "======================" >> $LOGFILE
   echo "No Internet connection" >> $LOGFILE
   echo "======================" >> $LOGFILE
   CONNECTED=0
else
   echo "" >> $LOGFILE
   echo "===================" >> $LOGFILE
   echo "3G IFACE PING CHECK" >> $LOGFILE
   echo "===================" >> $LOGFILE
   ping -q -I 3g-3gHuawei -w 10 www.google.es  >> $LOGFILE
   if [ $? -ne 0 ]
   then
   echo "" >> $LOGFILE
     echo "======================================" >> $LOGFILE
     echo "Connected to Internet via other ifaces" >> $LOGFILE
     echo "======================================" >> $LOGFILE
      CONNECTED=2
   else
      echo "" >> $LOGFILE
      echo "============================" >> $LOGFILE
      echo "Connected to Internet via 3G" >> $LOGFILE
      echo "============================" >> $LOGFILE
      CONNECTED=1
   fi
fi

if [ $CONNECTED -ne 1 ]
then
   ifconfig | grep -q "3g-3gHuawei"
   if [ $? -eq 0 ]
   then
      echo "" >> $LOGFILE
      echo "=========================================================" >> $LOGFILE
      echo "It seems that the interface is up. Printing system log..." >> $LOGFILE
      echo "=========================================================" >> $LOGFILE
      logread >> $LOGFILE
      ifconfig 3g-3gHuawei down
      ifconfig 3g-3gHuawei up
      route add default gw 10.64.64.64 3g-3gHuawei
      ping -q -I 3g-3gHuawei -w 10 www.google.es
     if [ $? -ne 0 ]
      then
           echo "" >> $LOGFILE
      echo "=====================================" >> $LOGFILE
          echo "3G interface restarted without result" >> $LOGFILE
           echo "=====================================" >> $LOGFILE
           if [ $CONNECTED -ne 1 ] # 3G dongle is down; let's reconnect it
           then
                echo "" >> $LOGFILE
                echo "======================" >> $LOGFILE
                echo "Rebooting 3G dongle..." >> $LOGFILE
                echo "======================" >> $LOGFILE
                echo 0 > /sys/bus/usb/devices/1-1.1/authorized
                sleep 1
                echo 1 > /sys/bus/usb/devices/1-1.1/authorized
                sleep 5
                /etc/init.d/network restart
                sleep 15
                route add default gw 10.64.64.64 3g-3gHuawei
                ping -q -I 3g-3gHuawei -w 10 www.google.es
                if [ $? -ne 0 ]
                then
                    echo "" >> $LOGFILE
                    echo "=================================" >> $LOGFILE
                    echo "3G dongle rebooted without result" >> $LOGFILE
                    echo "=================================" >> $LOGFILE
                else
                    echo "" >> $LOGFILE
                    echo "====================================================" >> $LOGFILE
                    echo "3G dongle rebooted. Internet connection restablished" >> $LOGFILE
                    echo "====================================================" >> $LOGFILE
                fi
           fi
     else
           echo "" >> $LOGFILE
           echo "========================================================" >> $LOGFILE
           echo "3G interface restarted. Internet connection restablished" >> $LOGFILE
           echo "========================================================" >> $LOGFILE
      fi
   else
      echo "" >> $LOGFILE
      echo "=======================================================" >> $LOGFILE
      echo "The 3G interface does not exist. Printing system log..." >> $LOGFILE
      echo "=======================================================" >> $LOGFILE
      logread >> $LOGFILE
      ifconfig 3g-3gHuawei up
      route add default gw 10.64.64.64 3g-3gHuawei
      ping -q -I 3g-3gHuawei -w 10 www.google.es
     if [ $? -ne 0 ]
      then
           echo "" >> $LOGFILE
      echo "=====================================" >> $LOGFILE
         echo "3G interface restarted without result" >> $LOGFILE
           echo "=====================================" >> $LOGFILE
           if [ $CONNECTED -ne 1 ] # 3G dongle is down; let's reconnect it
           then
                echo "" >> $LOGFILE
                echo "======================" >> $LOGFILE
                echo "Rebooting 3G dongle..." >> $LOGFILE
                echo "======================" >> $LOGFILE
                echo 0 > /sys/bus/usb/devices/1-1.1/authorized
                sleep 1
                echo 1 > /sys/bus/usb/devices/1-1.1/authorized
                sleep 5
                /etc/init.d/network restart
                sleep 15
                route add default gw 10.64.64.64 3g-3gHuawei
                ping -q -I 3g-3gHuawei -w 10 www.google.es
```

```
                if [ $? -ne 0 ]
                then
                     echo "" >> $LOGFILE
                     echo "=================================" >> $LOGFILE
                     echo "3G dongle rebooted without result" >> $LOGFILE
                     echo "=================================" >> $LOGFILE
                else
                     echo "" >> $LOGFILE
                     echo "====================================================" >> $LOGFILE
                     echo "3G dongle rebooted. Internet connection restablished" >> $LOGFILE
                     echo "====================================================" >> $LOGFILE
                fi
           fi
     else
           echo "" >> $LOGFILE
           echo "========================================================" >> $LOGFILE
           echo "3G interface restarted. Internet connection restablished" >> $LOGFILE
           echo "========================================================" >> $LOGFILE
     fi
   fi
fi
```

Listing 4. Code of the 3G connectivity maintenance daemon that recovers the connectivity whenever there are network or device problems

```
#!/bin/sh /etc/rc.common
# Captor init script
# CAPTOR AC UPC 2016
# @author: srodrigo

START=10
STOP=15

start() {
     echo "starting captor service"
     reset-mcu
     sleep 15 # Wait for mcu
     /bin/captor/forever.sh &
     # commands to launch application
}

stop() {
     echo "stopping captor service"
     killall forever.sh
     killall captor.py
     # commands to kill application
}
```

Listing 5. CAPTOR daemon init script

```
#!/bin/ash

# run forever captor python daemon

while true; do
   /bin/captor/captor.py > /mnt/sda1/errorlog.txt 2> /mnt/sda1/errorlog.txt
   sleep 5
done
```

Listing 6. Run-forever simple approach for the CAPTOR daemon

Appendix B: Shopping list for a Captor node

Index	Item	Supplier	Count per Node	Unit Price (€)	Subtotal (€)	Comment (code in supplier)
1	HUAWEI 3G USB E303 (captor 3G)	Zoom Informatica	1	29,50	29,50	Huawei E303
2	O3 gas sensor MICS-2610	cookinghacks	5	35,00	175,00	mics2614
3	Grove Temp&Humi sensor	cookinghacks	1	8,00	8,00	grove-twig
4	Breadboard	cookinghacks	1	6,00	6,00	Self-adhesive breadboard
5	Grove-RTC clocks	cookinghacks	1	9,00	9,00	twig/grove RTC
6	Taps (10,7mm)	Farnell	6	0,12	0,74	1336184
7	cable gland M20 10-14 (stuffing box)	Farnell	6	7,31	43,86	2499302
8	cable gland M20 6-12 (stuffing box PG 13.5) power supply	Farnell	1	1,37	1,37	1174594
9	Box 160x160x90	Farnell	1	29,95	29,95	1554SGY
10	Arduino Yún with PoE	ondaradio.es	1	84,70	84,70	A000003
11	Heat shrink tube 19.1mm (termoretractile pipe)	ondaradio.es	6	1,92	11,54	TRS191N
12	Heat shrink tube 25.4mm (termoretractile pipe)	ondaradio.es	6	1,92	11,54	TRS254N
13	Tarjetas microSD (4GB)	ondaradio.es	1	4,39	4,39	1011967
14	Connector Cable DE 0,28 mm² Orange (10m) O3	ondaradio.es	1	2,53	2,53	CC2252NA
15	Cable Connector DE 0,28 mm² marrón (10m) O3	ondaradio.es	1	2,53	2,53	CC2252M
16	Cable Connector DE 0,28 mm² blue (10m) NO2	ondaradio.es	1	2,53	2,53	CC2252AZ
17	Cable Connector DE 0,28 mm² red (10m)	ondaradio.es	1	2,53	2,53	CC2252R
18	Cable Connector DE 0,28 mm² violet (10m)	ondaradio.es	1	2,53	2,53	CC2252V
19	Cable Connector DE 0,28 mm² grease (10m) NO2	ondaradio.es	1	2,53	2,53	CC2252G
20	Cable Connector DE 0,28 mm² black (10m)	ondaradio.es	1	2,53	2,53	CC2252N
21	Cable Connector DE 0,28 mm² green (10m)	ondaradio.es	1	2,53	2,53	CC2252V
22	Cable Connector DE 0,28 mm² yellow (10m)	ondaradio.es	1	2,53	2,53	CC2252A
23	Cable Connector DE 0,28 mm² white (10m)	ondaradio.es	1	2,53	2,53	CC2252B
24	batteries CR1225 (3V) for the RTC	ondaradio.es	1	1,20	1,20	CR1225
25	Cable USB-A a microUSB-B (+1m)	ondaradio.es	1	2,78	2,78	NI2765
26	Transformador AC-USB	ondaradio.es	1	7,47	7,47	PSUPUSB401
27	Pack 100 resistors 1K tolerance 5% (sensor load)	ondaradio.es	1	1,21	1,21	PR251KD
28	Pack 100 resistors 100K tolerance 5% (sensor load)	ondaradio.es	1	1,21	1,21	PR25100KD
29	Pack 100 resistors 330 tolerance 5% (O3)	ondaradio.es	1	1,21	1,21	PR25330HD
30	Pack 100 resistors 220 tolerance 5% (O3)	ondaradio.es	1	1,21	1,21	PR25220HD

31	Pack 100 resistors 1K tolerance 1% (sensor load)	ondaradio.es	1	1,21	1,21	MF251K
32	Pack 100 resistors 100K tolerance 1% (sensor load)	ondaradio.es	1	1,21	1,21	MF25100K
33	Pack 100 resistors 10K tolerance 1% (sensor load)	ondaradio.es	1	1,21	1,21	MF2510K
34	Pack 100 resistors 330 tolerance 1% (O3)	ondaradio.es	1	1,21	1,21	MF25330H
35	Pack 100 resistors 220 tolerance 1% (O3)	ondaradio.es	1	1,21	1,21	MF25220H
36	Tub of PVC 2cm external diameter	servei estació	1	1,10	1,10	TUBO PVC GRIS
37	Tub of PVC 1cm external diameter	servei estació	1	3,15	3,15	TUBO PVC TRANSPARENTE
				Total Price	**467,47**	

Appendix C: Shopping list for a Raptor node

	Item	Supplier	Node Type	Count per Node	Unit Price (€)	Sub total (€)	Comment
1	2.4G IEEE802.15.4 Antenna	Farnell	RAPTOR End-Device	1	5,41	5,41	http://at.farnell.com/rf-solutions/ant-24g-whj-sma/antenna-whip-sma-90d-2-4ghz/dp/1304038
2	uSu_Edu Board	SMIR	RAPTOR End-Device	1	100,00	100,00	uSu_Edu by LIMOS, UBP, France
3	Box for End-Device	Farnell	RAPTOR End-Device	1	12,63	12,63	http://at.farnell.com/fibox/ta201-610/box-grau-ip65-201x163x98mm/dp/1422670
4	Alcaline Battery	Farnell	RAPTOR End-Device	2	4,61	9,22	http://at.farnell.com/energizer/e300116200/pile-alcaline-4-5v-3lr12/dp/2507368
5	Alphasense NO2 Sensor, NO2-B43F	Alphasense	RAPTOR End-Device	1	63,00	63,00	Arthur Burnley <awb@alphasense.com>
6	ISB for NO2 B-Series sensor	Alphasense	RAPTOR End-Device	1	96,00	96,00	Arthur Burnley <awb@alphasense.com>
7	Alphasense OZONE Sensor, OX-B431	Alphasense	RAPTOR End-Device	1	63,00	63,00	Arthur Burnley <awb@alphasense.com>
8	ISB for OX B-Series sensor	Alphasense	RAPTOR End-Device	1	96,00	96,00	Arthur Burnley <awb@alphasense.com>
9	Male-Female Cable	RS	RAPTOR End-Device	1	2,78	2,78	http://fr.rs-online.com/web/p/products/7916454/
10	Female-Female Cable	RS	RAPTOR End-Device	1	2,78	2,78	http://fr.rs-online.com/web/p/products/7916450/

11	CRC9 Connector 4G LTE Antenne 35dBi	Amazon.de	RAPTOR Local Server	1	19,99	19,99	https://www.amazon.de/gp/product/B01N7DWGVB/ https://www.amazon.fr/URANT-Connector-Antenne-Amplifier-EC5377u-872/dp/B01MU4LOH3/
12	Box for Local Server	Farnell	RAPTOR Local Server	1	23,87	23,87	http://at.farnell.com/fibox/ta292411/box-grau-ip65-289x239x107mm/dp/1422672
13	2.4G IEEE802.15.4 Antenna	Farnell	RAPTOR Local Server	1	5,41	5,41	http://at.farnell.com/rf-solutions/ant-24g-whj-sma/antenna-whip-sma-90d-2-4ghz/dp/1304038
14	+12V 30W AC-DC Power Supply	Farnell	RAPTOR Local Server	1	24,45	24,45	http://at.farnell.com/xp-power/afm30us12c2/alimentation-2-5a-12v-30w-iec/dp/2319724
15	AC Power Supply Cable	Farnell	RAPTOR Local Server	1	2,78	2,78	http://at.farnell.com/pro-elec/sh10167r/power-cord-euro-to-fig-8-2m/dp/1283799
16	uSu_Edu Board	SMIR	RAPTOR Local Server	1	100,00	100,00	uSu_Edu by LIMOS, UBP, France
17	Huawei E3272 LTE Surf-Stick	Amazon.de	RAPTOR Local Server	1	66,99	66,99	https://www.amazon.de/gp/product/B00HT2HP6E/ https://www.amazon.fr/Huawei-E3272-Surf-Stick-150Mbps-microSD/dp/B00HT2HP6E
18	RASPBERRYPI-3 & MicroSD 16Go	Farnell	RAPTOR Local Server	1	43,09	43,09	http://at.farnell.com/raspberry-pi/rpi3-modb-16gb-noobs/sbc-raspberry-pi-3-model-b-16gb/dp/2525227
					Total Price	**737,40**	