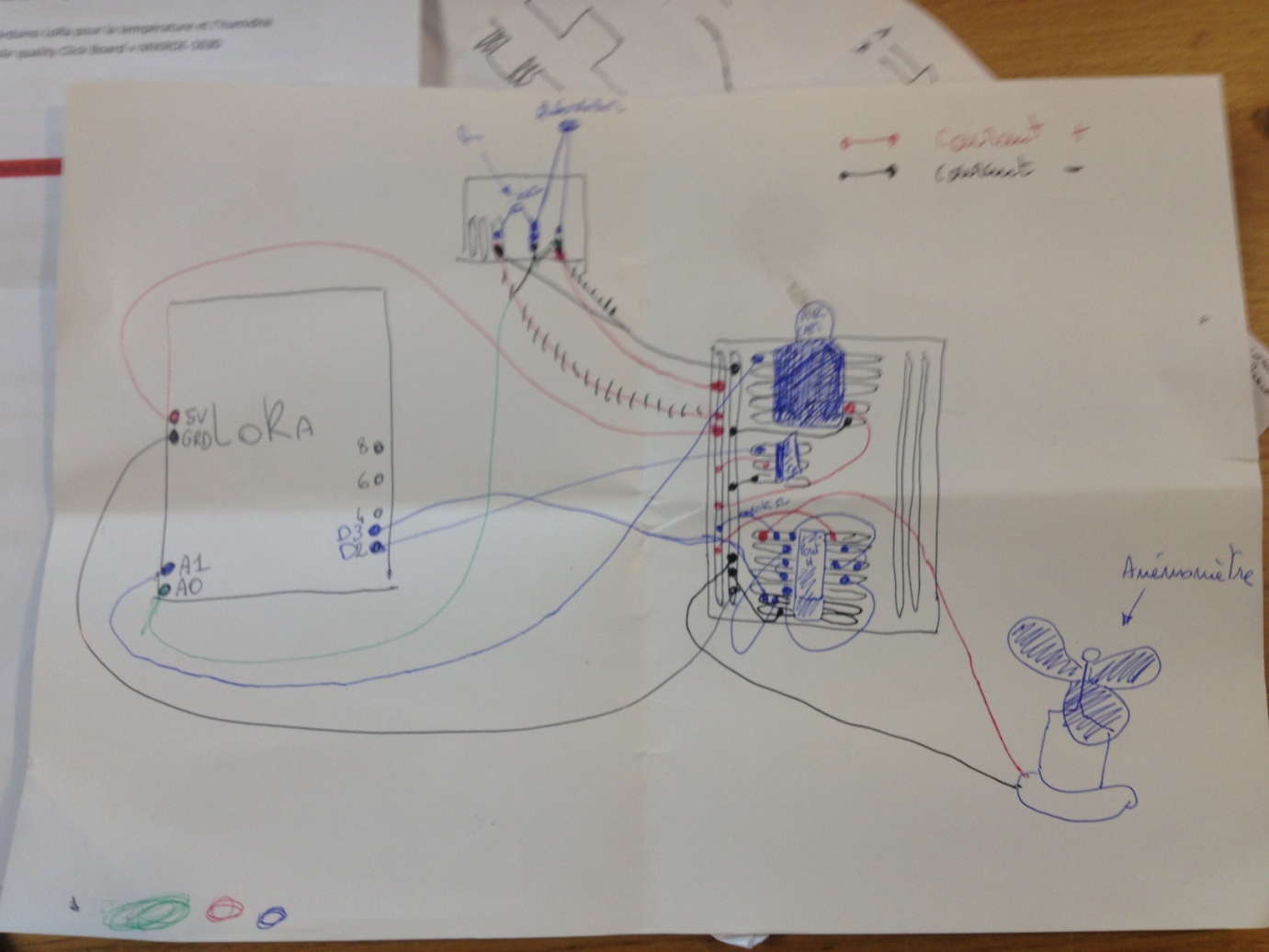
**Tuto SoundScape :**

**Station méteo :**

**Matériel :**

* Anémomètre Lextronic avec sortie digitale
* Photosenseur
* capteur H11 pour température et humidité
* capteur pour qualité de l’air

**Branchement :**



**Code fonctionnement du StarterKit Lora :**

D’abord faut installer Arduino 1.8.5 sur votre PC (logiciel OpenSource). Faut installer sur arduino les librairies OrangeRn2483 ; LpwaOrangeEncoder ; SimpleDHT. Ensuite faut téléverser ce programme :

//Déclaration librairies :

#include <OrangeRn2483.h>

#include <LpwaOrangeEncoder.h>

#include <SimpleDHT.h>

int pinDHT11 = 2; //Déclaration que le Pin 2 sur le STarteKit reçois info du DHT11

int anemometre = 3; //Déclaration que le Pin 3 sur le STarteKit reçois info du anémomètre

// Initialisation des variables utilisés

SimpleDHT11 dht11;

int sensorValue = 0;

int outputValue = 0;

int gaslevel = 0;

int gasvalue = 0;

unsigned long vent = 0;

// Définition de la clé de lecture pour envoyer les données sur LiveObjects.

const int8\_t appEUI[8] = { 0x45, 0x78, 0x70, 0x4C, 0x6F, 0x52, 0x65, 0x72 };

const int8\_t appKey[16] = { 0xff, 0xee, 0xdd, 0xcc, 0xbb, 0xaa, 0x99, 0x88, 0x00, 0x04, 0xA3, 0x0B, 0x00, 0x1E, 0xC8, 0xBE };

// Définition d’une fonction qui permettra au programme de communiquer avec le siteweb LiveObject

bool joinNetwork()

{

OrangeRN2483.setDataRate(DATA\_RATE\_1); // Set DataRate to SF11/125Khz

return OrangeRN2483.joinNetwork(appEUI, appKey);

}

// Déclaration des pins utilisées sur le Starter Kit

void setup() {

pinMode(A0,INPUT);// Utilisé pour photosenseur

pinMode(A1, INPUT);// Utilisé pour la pollution de l’air

pinMode(anemometre, INPUT); // utilisé pour l’anémomètre

OrangeRN2483.init();

}

// Définition d’une fonction qui permettra d’envoyer des données sur le siteweb LiveObject

bool SendLoRaMessage(int input1,int input2,int input3, int input4, int input5)

{

int8\_t size = 1;

int8\_t port = 5;

LpwaOrangeEncoder.flush();

String input;

input.concat(input1);input.concat(input2);input.concat(input3);input.concat(input4);input.concat(input5);

LpwaOrangeEncoder.addInt(input.toInt());

int8\_t\* frame = LpwaOrangeEncoder.getFramePayload(&size);

return OrangeRN2483.sendMessage(frame , size, port);

}

// Instructions executes en boucle infinit

void loop() {

delay(10000); // delay où le Starte kit ne fait rien

byte temperature = 0;

byte humidity = 0;

int err = SimpleDHTErrSuccess; //Variable pour verifier que le DHT11 fonctionne

if ((err = dht11.read(pinDHT11, &temperature, &humidity, NULL)) != SimpleDHTErrSuccess) { // Stockage valeurs de temperature et humidité

return;

}

bool res = joinNetwork();// 1 si communication avec Liveobject, 0 sinon

if(res)

{

OrangeRN2483.enableAdr(); // Etablissement du protocol de communication

sensorValue = analogRead(A0); // Lecture du courant analogique en milliVolt du photosenseur

outputValue = map(sensorValue, 0, 1023, 0, 10); // Conversion de millivolt en une valeur comprise entre 0 et 10

gaslevel = analogRead(A1); // Lecture du courant analogique en milliVolt du capteur de qualité de l’air

gasvalue = map(gaslevel, 0, 1023, 0, 99); // Conversion de millivolt en une valeur comprise entre 0 et 99

vent = Comptage(anemometre,HIGH,1000000); // Nombre de tours de l’anémomètre en 1O^6 millisecondes

SendLoRaMessage((int)temperature,(int)humidity,outputValue,gasvalue,vent); // envoie des données sur Liveobject

}

}

long count = 0;

// Définition d’une function qui compte le nombre de tours de l’anémomètre durant un laps de temps définit par « timeout »

unsigned long Comptage(uint8\_t pin, uint8\_t state, unsigned long timeout)

{

count =0;

unsigned long startMicros = micros();

while (micros() - startMicros < timeout)

{

// attendre la fin de toute impulsion précédente

while (digitalRead(pin) == state) {

if (micros() - startMicros > timeout)

return count;

}

count++;

// attendre que la pulsion arrive

while ((digitalRead(pin)) != state) {

if (micros() - startMicros > timeout)

return count;

}

}

return count;

}

**Station son pour le casque :**

**Matériel :**

* **Raspberry Pi**
* **carte Sd 16 gb**
* **casque audio**
* **Alimentation pour le Raspberry**

**Branchement :**

Il suffit de brancher le casque et l’alimentation au raspberry

**Configuration Raspberry :**

* Installer système exploitation Noob (Opensource)
* Installer Python3
* Installer librairies Pygame et urllib pour Python
* Mettre sur le bureau du raspberry les musiques fournis par le 3ème lieu en format ogg
* Mettre sur le bureau le code python ci-dessous

**Code python :**

#Déclaration lib utilisés

import pygame

import urllib

import sys

if sys.version\_info[0] == 3:

from urllib.request import urlopen

else:

from urllib import urlopen

while 1:#Boucle infinie

with urlopen("http://161.105.212.131/lora\_0004A30B001EC8BE/raw.php") as url:

s = url.read() #Lecture des données sur le siteweb Liveobject

PL = s.decode("utf8")

url.close()

# Récup des données de la station méteo

temp = int(PL[83:84],16)

humy = int(PL[85:86],16)

Lumy = int(PL[87],16)

gas = int(PL[88:89],16)

Vent = int(PL[90:91],16)

#Initialisation du son

pygame.mixer.init()

pygame.mixer.pre\_init(44100, -16, 2, 2048)

pygame.init()

sounds = []

if temp < 10 :#Différent sons pour la température en fonction de l’intervalle

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_temperature\_cold.ogg'))

elif temp > 20 :

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_temperature\_warm.ogg'))

else :

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_temperature\_medium.ogg'))

if humy < 30 : #Différent sons pour l’humidité en fonction de l’intervalle

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_rain\_light.ogg'))

elif humy >60 :

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_rain\_hard.ogg'))

else :

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_rain\_medium.ogg'))

if Vent < 8 : #Différent sons pour l’anémomètre en fonction de l’intervalle

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_wind\_low.ogg'))

elif Vent > 24 :

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_wind\_high.ogg'))

else :

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_wind\_medium.ogg'))

if gas < 40 : #Différent sons pour pollution de l’air en fonction de l’intervalle

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_air\_quality\_good.ogg'))

elif gas > 50 :

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_air\_quality\_bad.ogg'))

else :

sounds.append(pygame.mixer.Sound('/home/pi/Desktop/03\_air\_quality\_medium.ogg'))

for sound in sounds:

sound.play(60)# Execution du son crée en 60 boucles consécutives

**Exécution du code python au branchement de l’alimentation au raspberyy :**

1-Aller dans console Rasp

2-ouvrir terminal

3-taper : sudo nano /etc/rc.local

4-écrire à la première ligne la commande : python3 /chemin\_vers/face\_detect.py &

5-Sauver avec ctrl+o et quitter avec ctrl+q

6-Eteignez et rallumé la raspberry et le script python devraient s’exécuter automatiquement