République Algérienne Démocratique et Populaire
الجمهورية الجزائرية الديمقراطية الشعبية
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
وزارة التعليم العالي و البحث العلمي

المدرسة الوطنية العليا للإعلام الآلي
(المعهد الوطني للتكوين في الإعلام الآلي سابقا)
Ecole nationale Supérieure d'Informatique
ex. INI (Institut National de formation en Informatique)

Rapport Projet SCI

Réalisation d'une station météorologique connectée avec Arduino et Raspberry

Réalisé par :

- LAMDANI Wilem
- BOUROUINA Anfal
- MAKHLOUFI Kenza
- BELKESSA Linda

Enseignant : Mr. Sehad

Promotion 2021-2022

ESI (Ecole nationale Supérieure d'Informatique) B.P 68
M Oued Smar Alger Tel: +213 023 93 91 32 Fax :
+213 023 93 91 34
Site web: http://www.esi.dz

Table des matières

Introduction

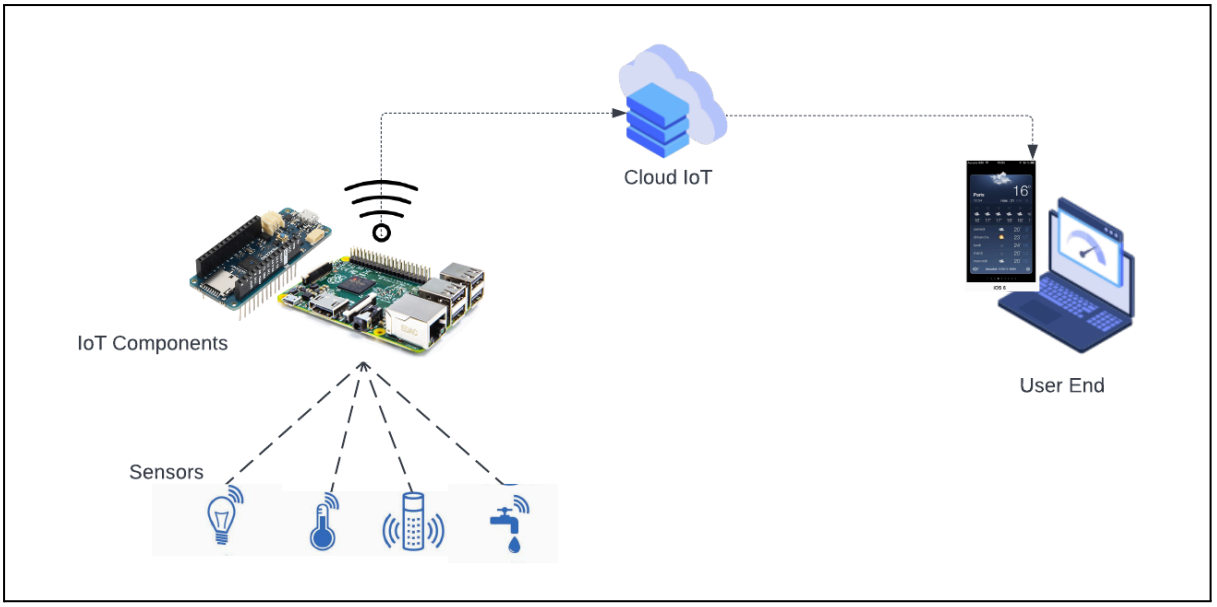
Aujourd'hui, l'électronique est de plus en plus remplacée par l'électronique programmée. On parle aussi de système embarquée ou d'informatique embarquée. Son but est de simplifier les schémas électroniques et par conséquent réduire l'utilisation des composants électroniques, réduisant ainsi le coût de fabrication d'un produit. Il en résulte des systèmes plus complexes et performants pour un espace réduit. Depuis quelques années l'internet des objets fit son apparition et le besoin d'observer et de contrôler des phénomènes physiques tels que la température, la pression, l'humidité ou encore la luminosité, est essentiel pour de nombreuses applications industrielles et scientifiques.

Dans notre travail nous allons réaliser une station météo de mesure des phénomènes physiques . Dans l'ensemble, la station comprend plusieurs capteurs tels que: Capteur de Température, Capteur d'Humidité, Capteur de Pression, Capteur d'Ensoleillement, capteurs de vitesse et direction de vent . Une carte d'acquisition à base d'Arduino MKR 1010 a pour but de transférer les données mesurées de ces capteurs vers le broker Mqtt (mosquitto) installé sur une carte raspberry. Deux principaux objectifs sont visés: Le premier objectif est de regrouper suffisamment d'informations sur les capteurs utilisés ainsi que leur fonctionnement. Le deuxième objectif consiste à réaliser une communication entre Node-red, le broker Mqtt et Arduino.

L'ensemble des données récoltées sont en temps réel et seront résumés sous forme d'un dashboard comprenant des widgets transmettant ces informations à l'utilisateur.

Description du projet

L'objectif principal du projet est de présenter l'étude des objets connectés (IoT) et leurs capacités à l'évaluation et le suivi des phénomènes météorologiques ainsi que le traitement et manipulation de ces mesures physiques liées aux variations du climat, avec la réalisation d'un système connecté au réseau internet, communiquant les données à temps réel à distance ainsi que l'historique des mesures prises, qui sont consultables via une plateforme IoT pour une interface adaptée à l'utilisateur. Un système évolutif avec un aspect dynamique, acceptant des modifications autant matérielles que logicielles. L'objectif final est la réalisation d'une station météo intelligente et connectée tout en étant autonome.



Branchement des composants

La station permet de capter et communiquer les différentes mesures météorologiques :

- Température
- Luminosité
- Direction du vent
- Vitesse de vent
- Humidité
- Qualité de l'air
- Quantité de pluie
- Pression atmosphérique
- Altitude

Pour ce faire, plusieurs composants ont été utilisés. La liste détaillée figure dans le tableau ci-dessous.

Liste des capteurs

N°	Capteur	Utilisation	Branchement
5	TSL 2591	Capteur de lumière.	I2C (0x29)
2	BMP280	Capteur de pression et d'altitude.	I2C (0x76)
6	HM3301	Capteur de qualité d'air.	I2C (0x40)
1	DHT11	Capteur de température et d'humidité.	D3
3	SEN-1901	Anémomètre (vitesse de vent).	D1
3	SEN-1901	Girouette (direction de vent).	A6
7	YL-83	Capteur de pluie.	D2
4	Pluviomètre	Capteur de quantité de pluie.	D0

Tableau 1. Liste des capteurs utilisés et leur branchement.

Autre matériel

8	LCD Screen	Affichage en local.	I2C (0x29)
9	Bouton Poussoir	Changement de l'affichage.	D4

Tableau 2. Liste des autres composants utilisés.

Schéma de branchement

La figure qui suit schématise le circuit réalisé, les composants sont numérotés selon leur numérotation dans les deux tableaux précédents.

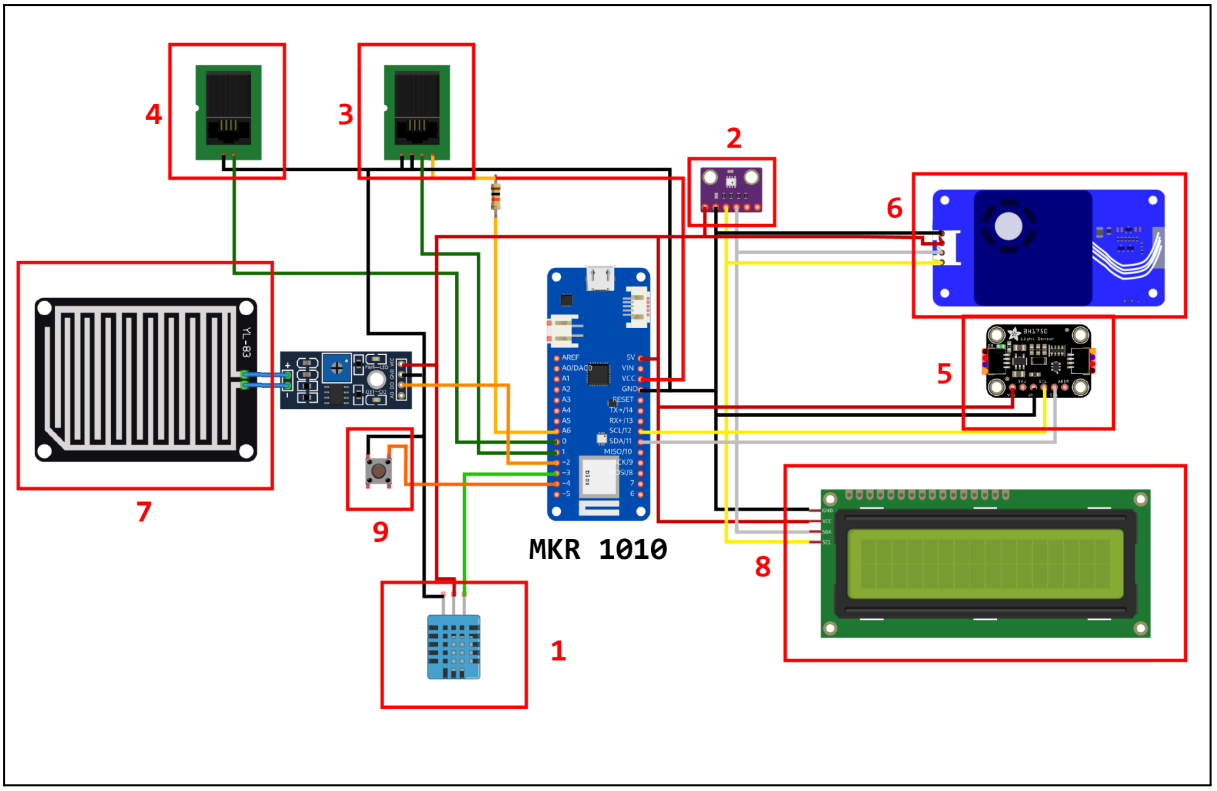


Figure 1. Schéma de branchement des composants utilisés.

Détails de branchement

Le branchement est plutôt classique, certains détails sont cependant à noter :

- L'Anémomètre et la Girouette partagent la même Sortie rj 11 : La pin 1 et 4 sont consacrées à la girouette, les pins 2 et 3 son pour l'anémomètre.
- La sortie de la girouette offre un signal analogique, devant être relié à une résistance de 10KΩ et a une alimentation 3v3 -Circuit anti-rebond-
- L'anémomètre, le pluviomètre et le bouton poussoir ont des sorties digitales servant à déclencher des interruptions : Ils sont donc utilisés en mode INPUT_PULLUP.
- Les différents capteurs I2C proviennent de fabricants différents, et ont leur propres résistances intégrées : Nous pouvons donc tous les relier au même bus I2C sans aucun conflit.

Challenges rencontrés lors du branchement

- Les capteurs de vitesse de vent (anémomètre) et de quantité de pluie (pluviomètre) ont des sorties RJ11 non exploitables directement par arduino.
    - Solution: l'utilisation d'un boîtier sur lequel se branche la sortie RJ11, puis l'adaptation de quelques câbles (4 pour l'anémomètre et 2 pour le pluviomètre) vers des sorties pin exploitables.

- Manque de documentation sur le pluviomètre, girouette et anémomètre. (Quel pin est le GND, la valeur de résistance adaptée pour chaque composant…)
- Le test du branchement de chaque composant individuellement et la vérification que les adresses I2C ne se chevauchent pas, grâce à un I2C Scanner (ceci a été évité en raison des constructeurs différents de chaque composant).
- La bonne exploitation des pins de la carte Arduino digitales et analogiques, interruptibles et non interruptibles.

Code Arduino

Afin de lier toutes les composantes du projet et les faire fonctionner correctement, nous devons programmer la carte arduino pour gérer et recevoir les données de tous les capteurs et les envoyer au broker mqtt sur les différentes topic à travers le wifi (connexion sur un réseau local).

Deux types de communication (filaire et non filaire) sont utilisés à travers I2c et wifi respectivement.

Librairies utilisées

Librairie	**Utilisation**
Tomoto_HM330X.h	Exploitation du capteur de poussière
Wire.h	Exploitation du bus I2C
Adafruit_Sensor.h	Exploitation des composants Adafruit
Adafruit_TSL2591.h	Exploitation du capteur de lumière (Adafruit)
DHT.h	Exploitation du capteur d'humidité
LiquidCrystal_I2C.h	Exploitation de l'écran LCD.
Adafruit_BMP280.h	Exploitation du capteur de pression & altitude.
ArduinoMqttClient.h	Etablissement des communications MQTT.
WiFiNINA.H	Exploitation du module WIFI en MKR 1010.
arduino_secrets.h	Stocke le SSID et mot de passe du WiFI

Tableau 3. Détails des librairies utilisées.

Partie Initialisation “setup”

```
void setup() {
  Serial.begin(9600);

//Initialisation de la communication WIFI
  while (WiFi.begin(ssid, pass) != WL_CONNECTED) {
    // failed, retry
    Serial.print(".");
    delay(5000);
  }

//Initialisation de la communication MQTT
  if (!mqttClient.connect(broker, port)) {
    Serial.print("MQTT connection failed! Error code = ");
    Serial.println(mqttClient.connectError());

    while (1);
  }

//Initialisation de la communication I2C
//Initialisation des capteurs sur le  bus I2C (lumière, poussière, pression)
//Initialisation du capteur d'humidité branché en D3
//Initialisation de l'afficheur LCD  avec un contraste d'affichage élevé
  Wire.begin();
  start_light_sensor();
  start_dust_sensor();
  start_pressure_sensor();
  dht.begin();
  lcd.init();
  lcd.backlight();

//Initialisation des capteurs météorologique + interruptions
  //Kit météo
  pinMode(PIN_VANE, INPUT);
  pinMode(PIN_ANEMOMETER, INPUT_PULLUP);
  pinMode(PIN_RAINGAUGE, INPUT_PULLUP);
  pinMode(PIN_PUSHBUTTON, INPUT_PULLUP);
  pinMode(PIN_RAINSENSOR, INPUT);
  attachInterrupt(digitalPinToInterrupt(PIN_ANEMOMETER) , countAnemometer, FALLING);
  attachInterrupt(digitalPinToInterrupt(PIN_RAINGAUGE) , countAnemometer, FALLING);
  attachInterrupt(digitalPinToInterrupt(PIN_PUSHBUTTON) , lcdChange, FALLING);
  nextCalcSpeed = millis() + MSECS_CALC_WIND_SPEED;
  nextCalcDir   = millis() + MSECS_CALC_WIND_DIR;
  nextCalcRain = millis() + MSECS_CALC_RAIN_FALL;

}
```

Partie Loop

```
float wind_speed, rain_fall, altitude, pressure, dust, light, temperature, humidity;
String wind_dir, isRaining;
int rain, wind_dir_mqtt,global_reading;

void loop() {
  time = millis();
//Récupération des valeurs captées à partir  des senseurs
  light = lightValue();
  dust = dustValue();
  temperature = temp_Value();
  humidity = Humidity_Value();
  altitude = alt_Value();
  pressure = pressure_Value();
  rain = digitalRead(PIN_RAINSENSOR);

//Evaluation de la valeur de capteur de pluie
  if (rain) {
    isRaining = "No";
  }else{
    isRaining = "Yes";
  }
//Evaluation de la valeur de capteur de vent (vitesse de vent)
   if (time >= nextCalcSpeed) {
      wind_speed = calcWindSpeed();
      nextCalcSpeed = time + MSECS_CALC_WIND_SPEED;
   }
//Evaluation de la valeur de capteur de niveau de pluie
   if (time >= nextCalcRain) {
      rain_fall = calcRainFall();
      nextCalcRain = time + MSECS_CALC_RAIN_FALL;
   }
//Evaluation de la valeur de capteur de vent (direction de vent)
  if (time >= nextCalcDir) {
    int indice = calcWindDir();
      wind_dir = strVals[indice];
       wind_dir_mqtt = dirVal[indice];
      nextCalcDir = time + MSECS_CALC_WIND_DIR;
   }
//Affichage des valeurs sur le moniteur série
   printValues();
//Envoie de valeurs vers le broker MQTT
   sendToMQTT();
//Affichage des valeurs sur le LCD
   afficherLCD();
  delay(500);
}
```

Challenges rencontrés lors de la réalisation du code

- Trouver les bonnes librairies pour chaque capteur.
- Se documenter sur l'utilisation de chaque capteur et garder que les fonctionnalités utiles au projet (eg. Le capteur de qualité d'air retournent

plusieurs valeurs, on a gardé que le taux des particules PM2.5 jugées dangereuses)

- Un mappage entre les valeurs produites par la girouette et les directions que signifie chaque valeur a nécessité beaucoup d'essais.
- L'intégration des codes de chaque composant dans un même code tout en gardant la lisibilité de ce dernier.

L'implémentation des fonctions utilisées dans la partie loop est présente sur github : Weather-station-iot.

Communication MQTT

La communication entre le dashboard Node-red et les composantes hardware est faite à travers le protocole MQTT à travers le broker Mosquitto soit en local à travers Raspberry pi où à travers le cloud à travers le serveur test.mosquitto.org.

la carte arduino MKR 1010 à travers le WIFI , communique avec le broker MQTT (raspberry pi où cloud) et publie les valeurs captées à partir des senseurs , les valeurs par la suite sont reçus à partir de Node-red où on les affiche et les interprète comme données météorologique.

Donc on aura 3 composantes connectée , arduino comme un publisher, Node-red comme subscriber et raspberry pi comme un broker, suivant ce schéma :

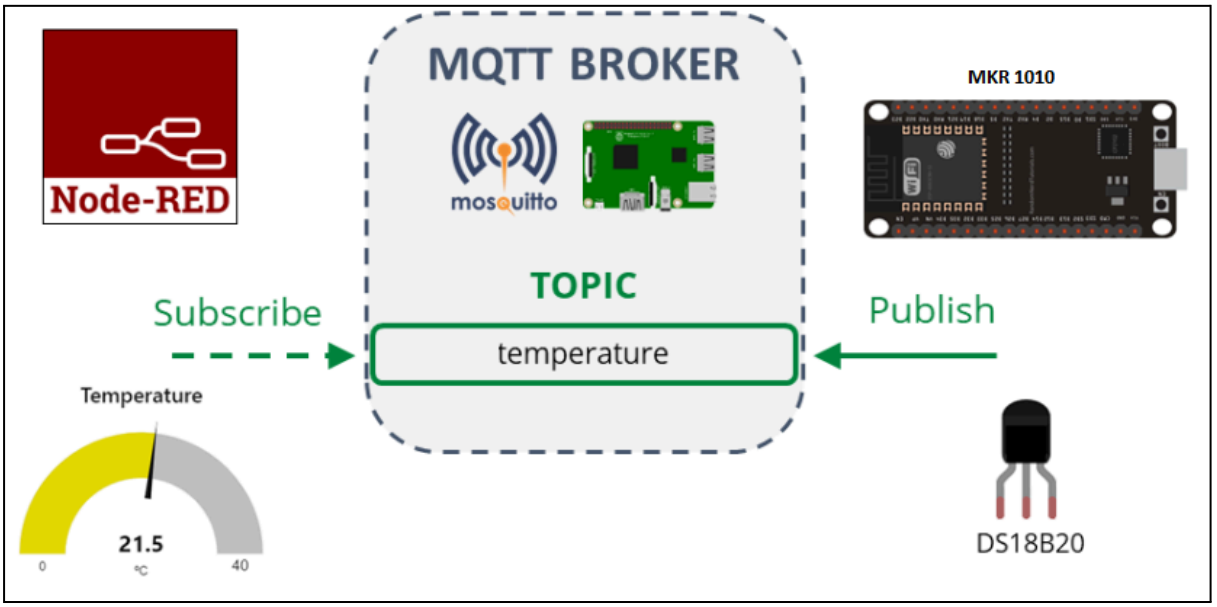


Figure 2. Schématisation de la communication MQTT.

Nous lançons node-red au niveau du pc pour contrôler et modifier les flux. Tandis que le broker mqtt tournera sur la carte raspberry en écoutant le port 1883. Une fois le flux déployé, les clients pourront accéder au dashboard à travers un navigateur web ou bien l'application Remote-RED.

Configuration Broker local (Raspberry pi)

Afin d'utiliser Raspberry pi 4 comme un broker mqtt, nous avons fait la configuration suivante:

1. Installer mosquitto
   sudo apt install -y mosquitto mosquitto-clients
2. Vérifier que le service mosquitto est activé
   sudo systemctl status mosquitto.service

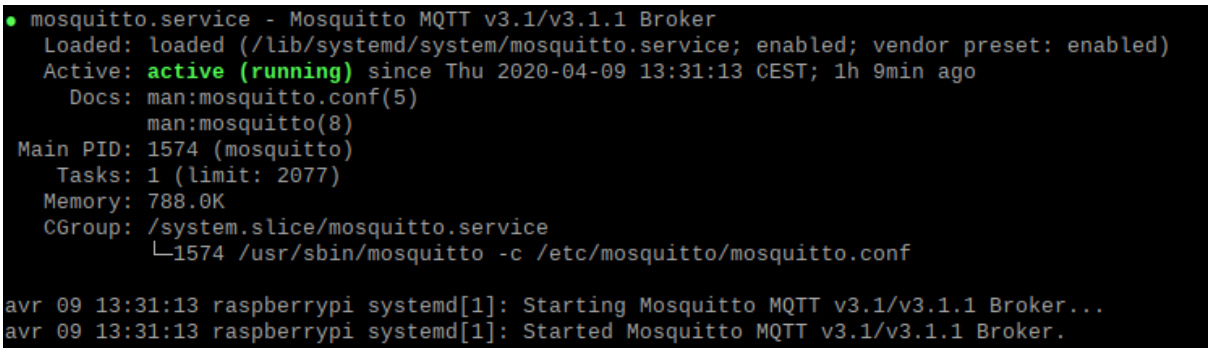


Figure 3. Lancement du broker MQTT.

Configuration des Topics

Pour capter les différentes mesures envoyées par les capteurs, nous avons utilisé un topic pour chaque valeur lue. Plus de détails dans le tableau suivant:

Nom du topic	**Capteur**	**Unité**	**Commentaires**
pressure	BMP280	Pascal	mesure la pression atmosphérique, les valeurs sont entre : 300 - 1100 hPa (hecto pascal)
dust	HM3301	µg/m^3	Interpretation valeurs: [0-30]: bon qualité d'air [30-50]: problème de santé (poussière) [50-100]: cas dangereux (l'air est très pollué)
altitude	BMP280	m	mesure la hauteur par rapport au niveau de mer
windDirection	SEN-1901	angle°	renvoie l'angle qui représente la déviation de direction depuis le nord
windDir	SEN-1901	/	renvoie la direction en text (Nord, Nord-est, Est, Sud-est, Sud, Sud-ouest, Ouest, Nord-ouest)

windSpeed	SEN-1901	km/h	renvoie vitesse de vent en km/h
rain	raingauge	mm	renvoie la quantité de pluie mesuré
raining	YL-83	/	renvoie une valeur boolean (0 -> pluie, 1-> pas de pluie)
humidity	DHT11	%RH	Humidité entre (10 et 90)
light	TSL 2591	Lux	Luminosité en lux , valeurs entre 0 et 2000 lux
temperature	DHT11	C°	Température en degré , valeurs entre 0 et 60° C.

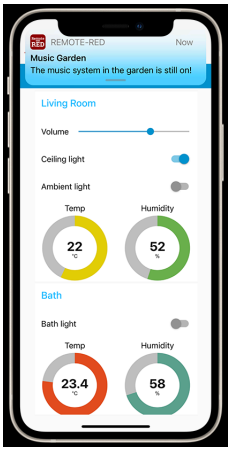
Tableau 4. Détails des topics MQTT.

Node-red

Nous avons installé et configuré node-red sur un pc dans le même réseau que le serveur du broker mqtt sur raspberry pi.
Installation de Node-red:

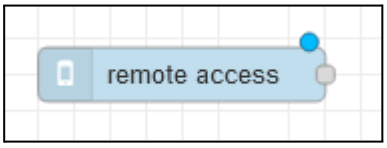
1. Télécharger et installer NodeJS depuis le site officiel[1].
2. Installer node-red à travers la gestionnaire des packages “npm” en utilisant la commande: npm install -g --unsafe-perm node-red.
3. Lancer node-red en utilisant la commande node-red et accéder à l’interface à travers http://127.0.0.1:1880 .

Application mobile Remote-RED



En plus de pouvoir visualiser le dashboard via un navigateur web, le client pourra tout de même exploiter cette application disponible sur les plateformes Android et IOS pour manipuler (ajouter, supprimer, modifier) les widgets de l’UI.

Elle nécessite uniquement l’ajout du node suivant au niveau du flux node-red :



[1] Download | Node.js (nodejs.org)

Flux node-red

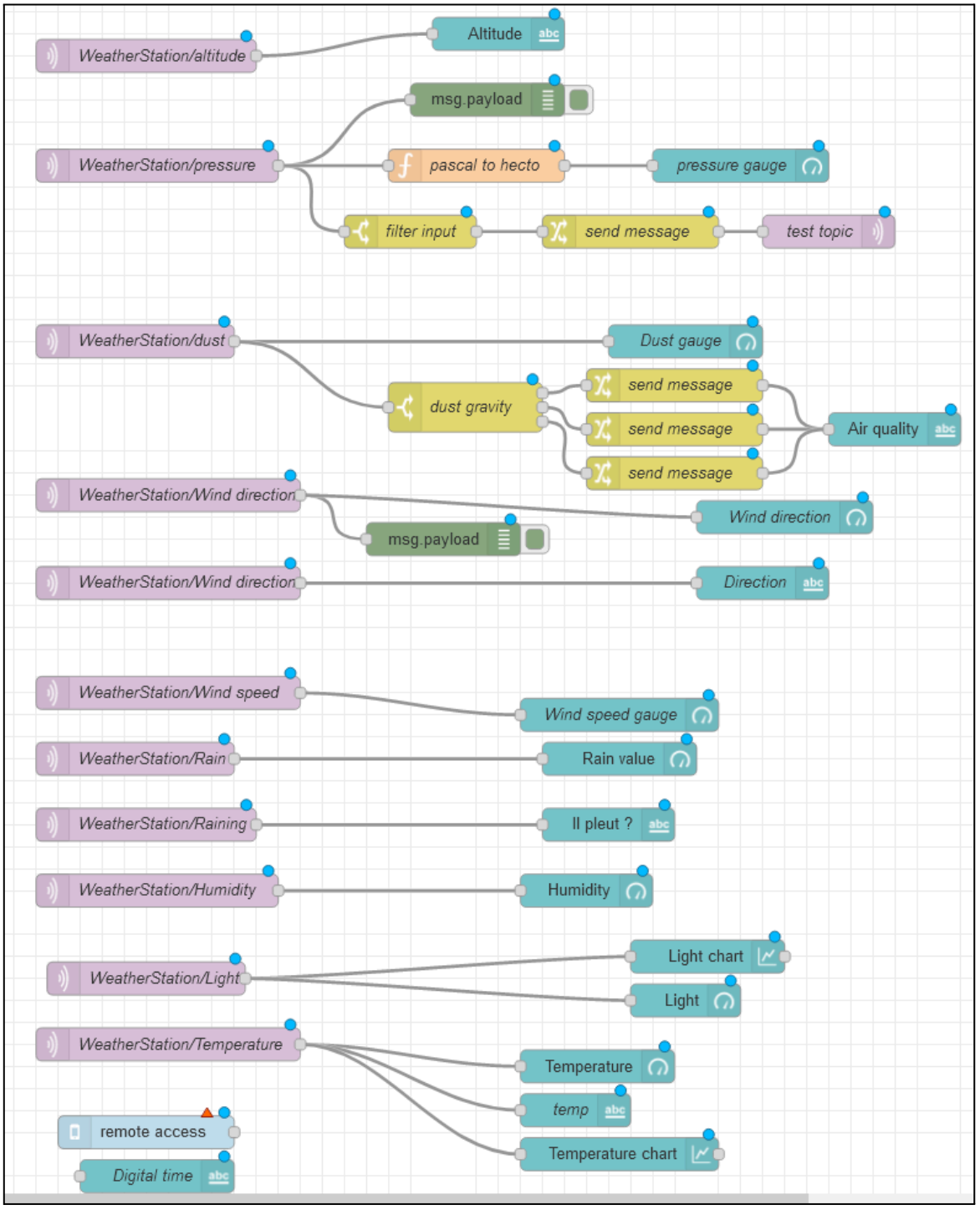


Figure 4. Capture d'écran sur les flux NodeRed.

Chaque node applicatif (agissant sur les widgets du dashboard) est précédé par des node de contrôle mqtt et/ou des nodes utilitaires.
Les nodes mqtt sont configurés pour écouter le port 1883 de l'adresse IP où se trouve le broker (dans notre cas Mosquitto) et en ajoutant aussi le topic cible.

Node	Type	Topic
Altitude	Text	altitude
pressure gauge	Gauge	pressure
Air quality	Text	dust
Wind direction	gauge	Wind direction
Wind speed gauge	gauge	Wind speed
Rain value	gauge	Rain
Il pleut ?	text	Raining
Humidity	gauge	Humidity
Light chart	chart	Light
Light	gauge	Light
Temperature	gauge	Temperature
Temperature	Text	Temperature
Temperature chart	chart	Temperature

Tableau 5. Liste des nodes sur NodeRed.

Nodes utilitaires

Ces nodes permettent d'exécuter certaines fonctions utiles comme la conversion d'unités, switchs et filtres, nous avons employé ce qui suit :

- pascal to hecto (fonction) : permet de passer au hPa (hectopascal) qui est 0.01*valeur en Pascal.
- dust gravity (switch) : ce node permet de choisir le message à envoyer selon un un intervalle de valeurs comme suit :
    - Si quantité de poussière <= 30 alors message = Healthy
    - Si quantité de poussière >30 et <= 50 alors message = Serious health issues
    - Sinon message = Dangerous

Le fichier qui contient les flux node-red est présent ici : Weather-station-iot.

Tests NodeRed

Dashboard web

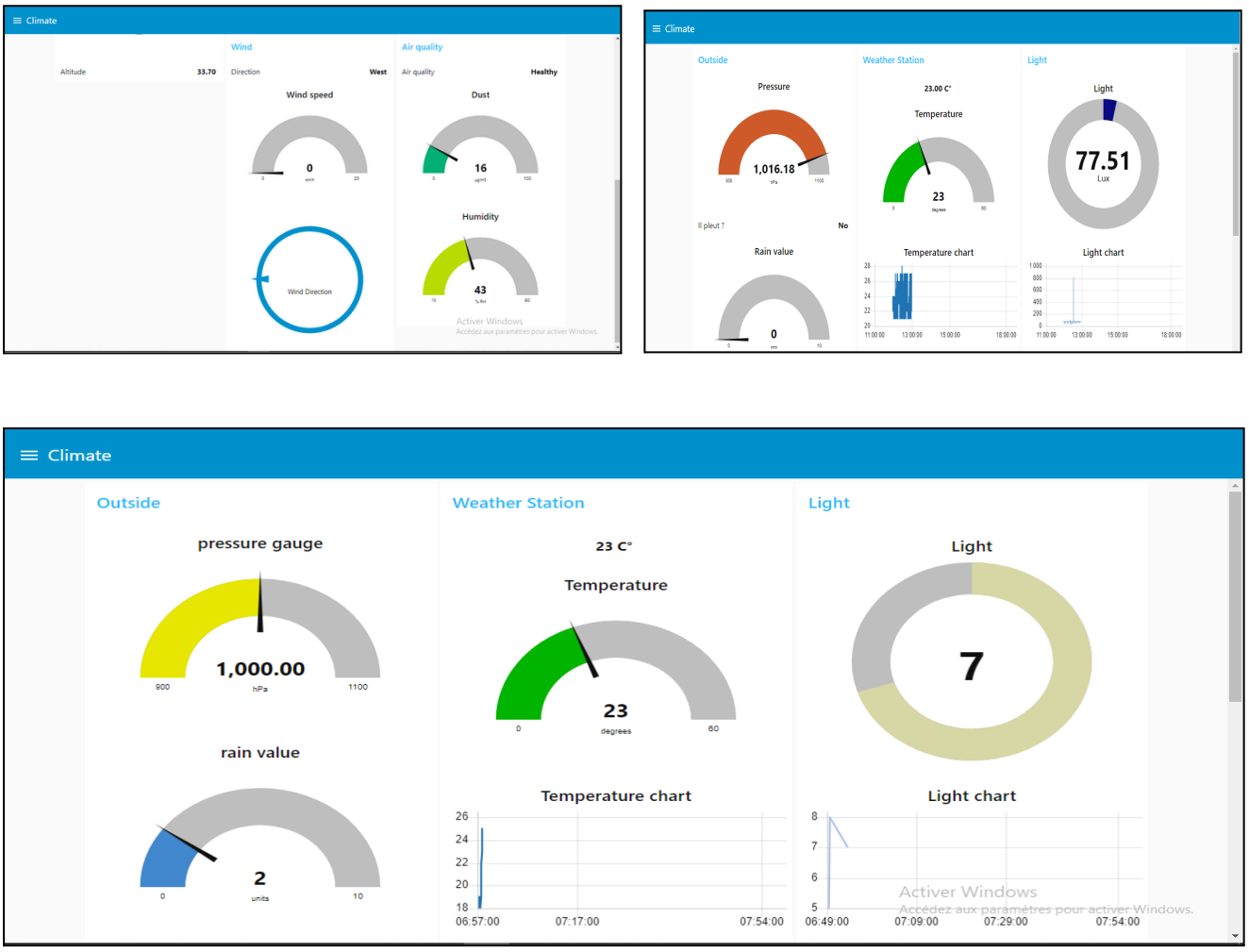


Figure 5. Captures d'écran sur le dashboard Web.

Dashboard client mobile (Remote-RED app)

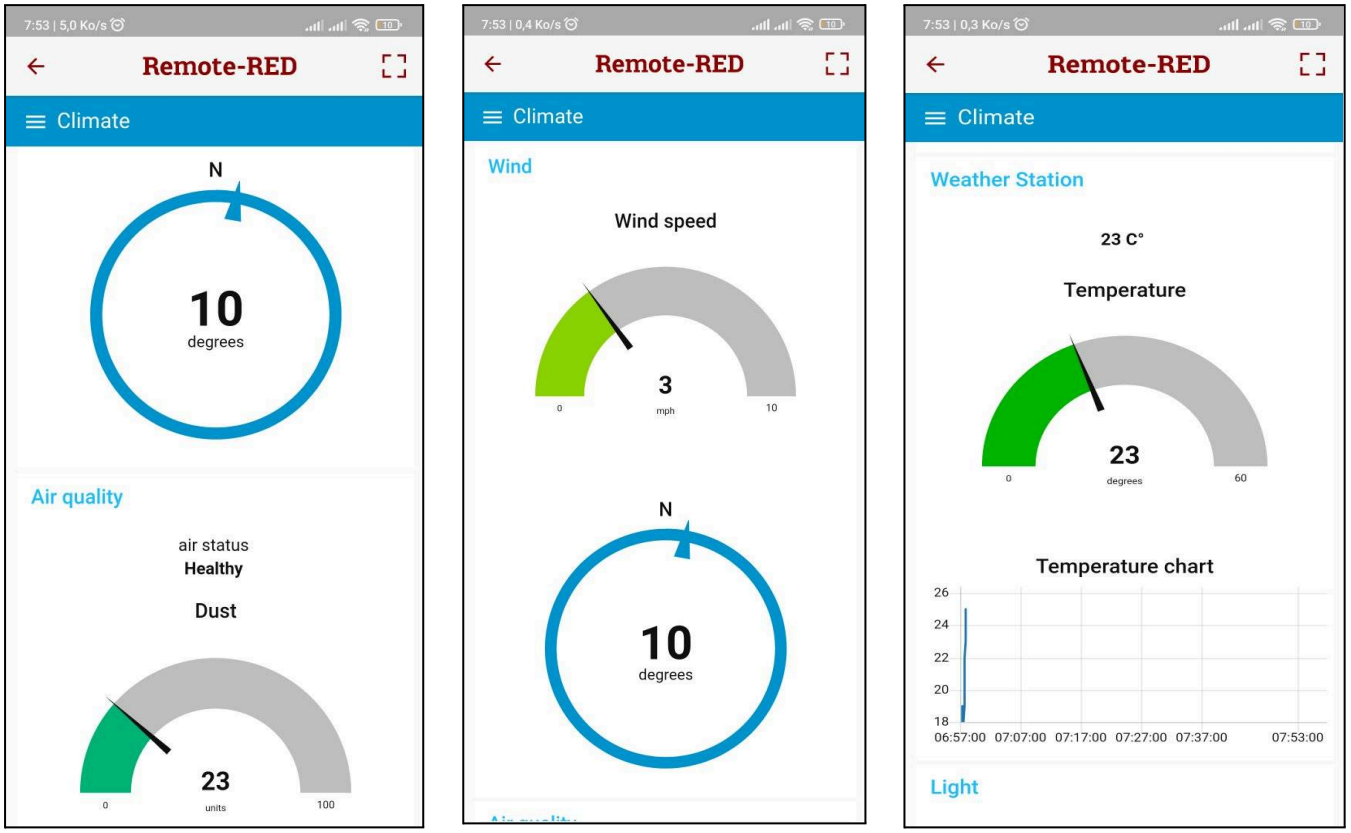


Figure 6. Captures d'écran sur le dashboard mobile.

Conclusion et perspectives

La station météo a pour but de renvoyer des données météo sur un réseau afin de permettre soit à toute une société soit à une famille d'avoir accès à tout moment aux données météorologiques locales. Les champs d'application sont très vastes, cela peut servir à gérer une serre, à s'informer sur le temps qu'il fait ou encore à gérer le chauffage et l'aération d'une maison afin par exemple de n'allumer le chauffage que si la température est en dessous d'un certain seuil ou encore de baisser les stores si la luminosité est trop forte. De même, les données renvoyées par la station peuvent servir pour la prévision météorologique et l'analyse des données à travers des protocoles à longue portée comme LoRaWan/Zigbee avec une alimentation solaire.