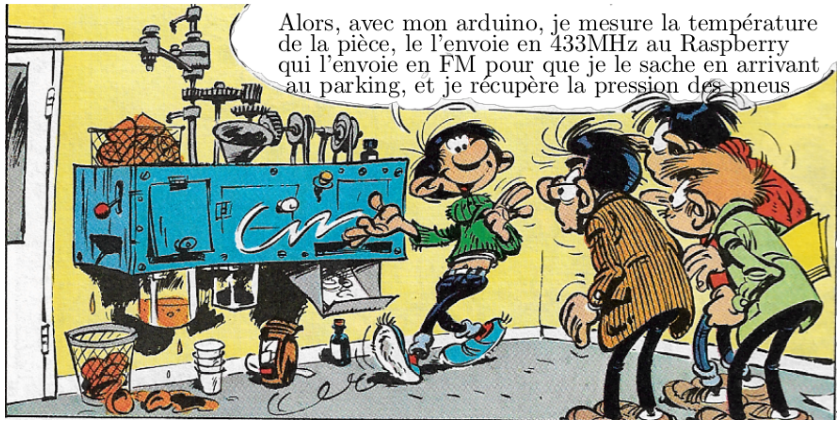
Architecture IoT

Arnaud Février — Yannick Marietti

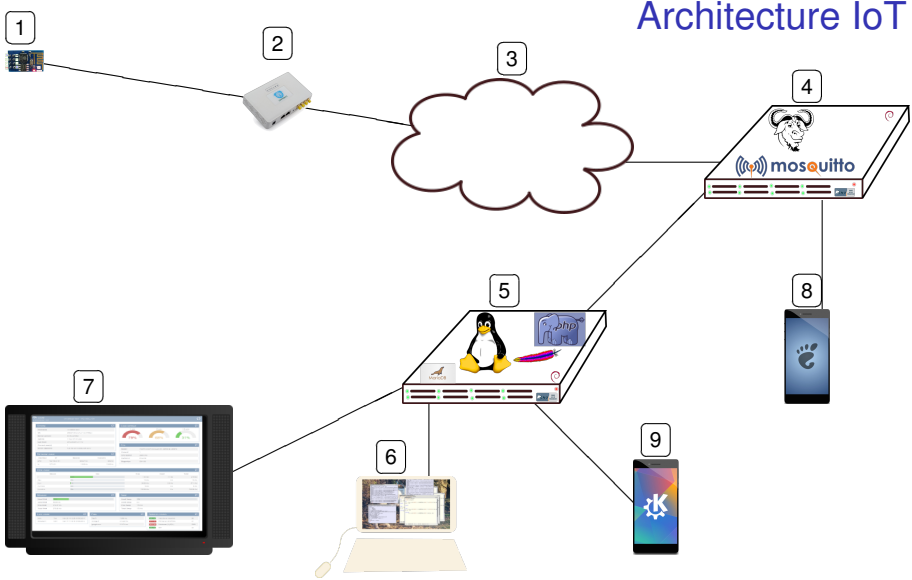


Plan

Application Météo

- Une ou plusieurs sondes environnementales
- Communication avec une Gateway en BLE
- Envoi des valeurs à un courtier MQTT
- Utilisation de plusieurs applications :
  - Affichage de la dernière valeur
  - graphique historique
  - valeurs sur une année
  - ...

Architecture IoT

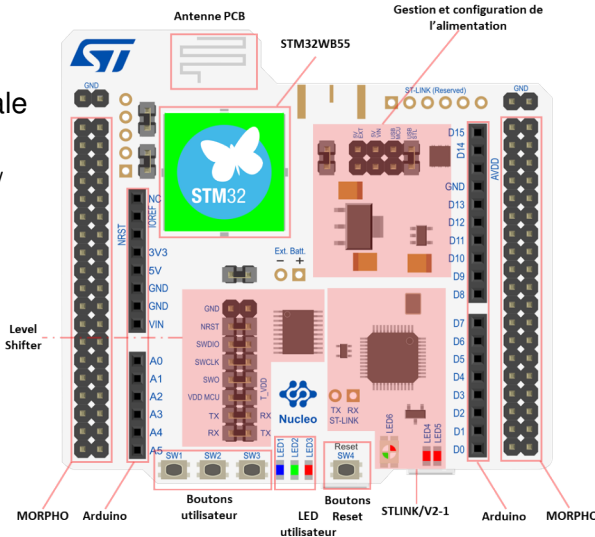


Architecture IoT

1. Sonde, capteur, afficheur, actionneur
2. Gateway (conversion vers IP)
3. Internet
4. Serveur réseau (Mosquitto, TTN)
5. Serveur(S) applicatif(S), Bases de données
6. Station de contrôle, développement
7. Station de visualisation, supervision
8. Application mobile MQTT
9. Application mobile

Nucleo WB55

- Solution mise en avant par ST pour l'éducation nationale
- Site web : https ://stm32python.gitlab.io/fr/

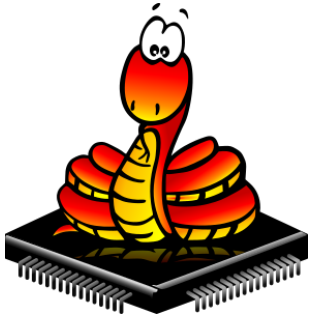


UNIVERSITÉ Grenoble Alpes

POLYTECH GRENOBLE

académie Grenoble
Région académique Auvergne-Rhône-Alpes

académie Aix-Marseille
Région académique PROVENCE-ALPES-CÔTE D'AZUR



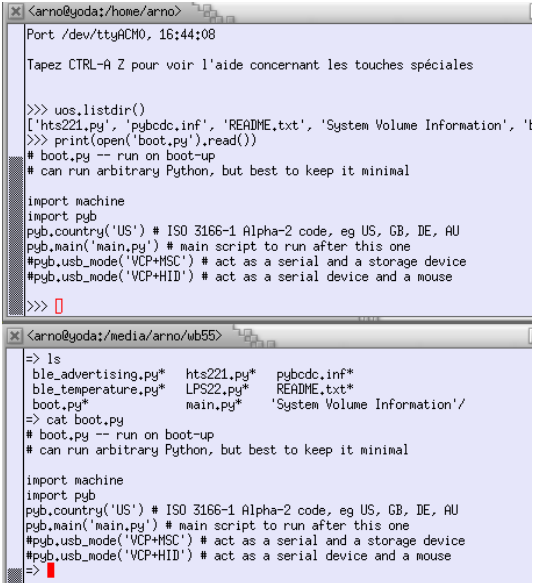
Python et MicroPython

- Langage interprété
- fichiers sur le microcontrôleur
  - facilité de modifications
  - beaucoup de librairies
  - langage à la mode dans l'éducation nationale française
- Compatibilité Python / MicroPython (presque)

Techniques de programmation

Microcontrôleurs grand public

- Branchement :
  - **USB**
  - UART (liaison série)
  - WiFi
- Système d'exploitation :
  - interface série (Modem)
  - Mémoire de masse
- Modification du firmware :
  - Complet
  - modification *d'un* fichier
  - Over The Air *(OTA)*



μPython et la température

- **HTS221.py**, disponible à plusieurs endroits et plusieurs versions.
- Capteur I2C, le bus ou adresse peuvent changer

Température du HTS221

```
from machine import I2C
import HTS221

i2c = I2C(1)      # I2C 1 du NUCLEO-W55
sensor1 = HTS221.HTS221(i2c)
t = sensor1.temperature()
```

Bluetooth Low Energy

Un protocole conçu pour l'IOT

- faible complexité $\Rightarrow$ faible coût
- faible bande passante (débit $<$ 2Mbps)
- faible consommation (courants $\sim \mu$A, pile bouton possible)
- BLE incompatible Bluetooth, mais puces *dual mode*
- bande ISM 2,4GHz
- portée 50m

Generic Access Profile

Rôles :

Périphérique (esclave)
- capteur
- peu de ressources

central (maître)
- Smartphone, ordinateur

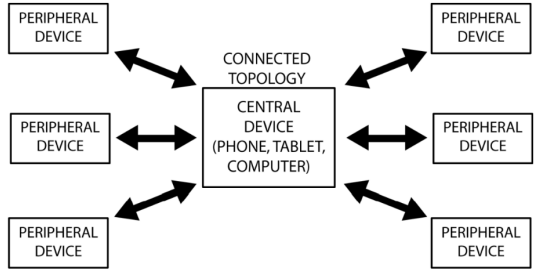
Advertiser broadcaster
- uniquement émission
- 31 octets
- pas de connexion

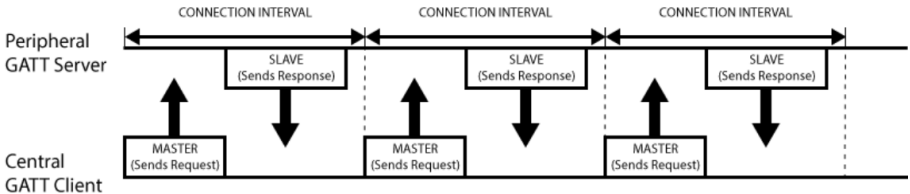
Scanner Observer
- uniquement réception
- pas de connexion

Connexion : ⇒ Protocole GATT

Generic Attributes Profile

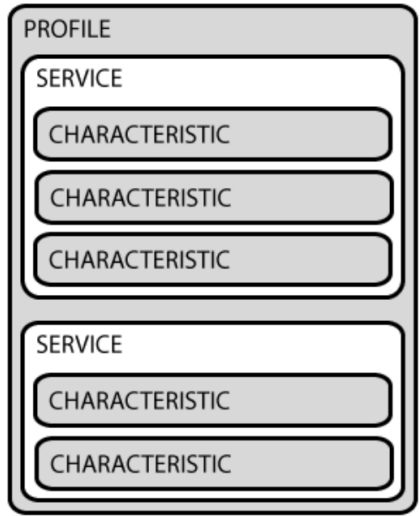
- Connexion exclusive (un seul central)
- Périphérique : serveur
- Central : client





Profils

- Profils : liste standard
  Ex : sonde
- Services : UUID
  Ex : environnement
  - 16 bits pour les standards
  - 128 personnalisés
- Caractéristiques UUID
  Ex : température



μPython et BLE

ble_temperature.py avec les sources de μPython

temp

```
import ble_temperature

ble = bluetooth.BLE()
temp = ble_temperature.BLETemperature(ble)

temp.set_temperature(t*100, notify=i == 0, indicate=False)
```

Linux Python et BLE

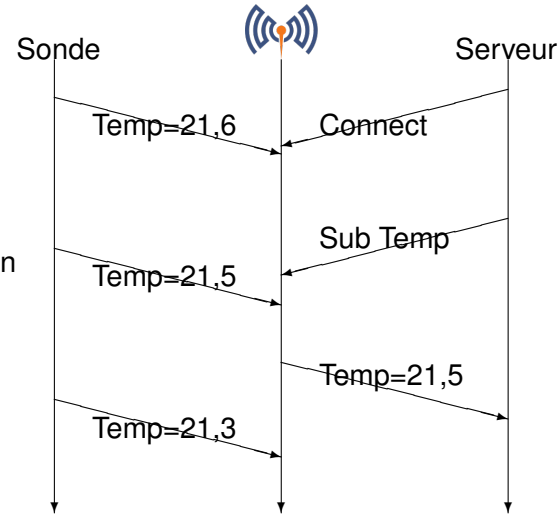
BLE WB55

```
from bluepy import btle

BTADDR="02:02:27:4E:21:92" # adresse par défaut
dev = btle.Peripheral(BTADDR)
HTS221=btle.UUID("0000181a-0000-1000-8000-00805f9b34fb")
Tservice=dev.getServiceByUUID(HTS221)
T=Tservice.getCharacteristics()[0]
Temperature=T.read()
```

Mosquitto

- Broker (courtier)
- Protocole léger
- Publication/ souscription
- `home/room1/temp`
- `home/room1/hum`



Le serveur Mosquitto

- Intégré dans les principales distributions GNU-Linux :

package Debian

```
# apt install mosquitto
```

- peu de configuration :
  - Écoute sur le réseau ou en local seulement (défaut)
  - Utilisateurs anonymes ou référencés

Création d'un utilisateur

```
# mosquitto_passwd -c /etc/mosquitto/pwfile
Password:
```

- Possibilité de chiffrement des communications (TLS)

Le serveur Mosquitto

- Écoute sur le réseau :

listen.conf

```
listener 1883 10.3.30.3
```

- Gestion des mots de passe :

users.conf

```
#allow_anonymous true
allow_anonymous false

password_file /etc/mosquitto/pwfile
```

Le client shell Mosquitto

Publication

```
mosquitto_pub -h 10.3.30.3 -u mqtt -P x
              -t sensor/temperature/chambre
              -m "20-06-11 09:26:26 27.6 62"
```

Souscription

```
mosquitto_sub -h 10.3.30.3 -u mqtt -P x
              -t sensor/temperature/chambre
20-06-11 09:26:26 27.6 62
...
```

Paquet `mosquitto-clients`

Le client python

Définitions

```
from paho.mqtt import client as mqtt_client
broker = '10.3.30.3'
port = 1883
topic = 'sensor/temperature/chambre'
client = mqtt.Client()
client.username_pw_set('mqtt','x')
client.connect(broker, 1883, 60)
```

Publication

```
client.publish('sensor/temperature/chambre','22.3')
```

souscription

```
client.publish('sensor/temperature/chambre','22.3')
```

Consignation des valeurs

crontab -l

```
*/5 * * * * /.../WB55env.py » /tmp/WB55.log
```

Souscription

```
mosquitto_sub -h 10.3.30.3 -u mqtt -P x
              -t sensor/temperature/chambre
              > Tch.log
```

Tch.log

```
21-03-16 00:00:03 17.9 46
21-03-16 00:05:03 17.0 45
21-03-16 00:10:03 17.9 44
...
21-03-16 17:20:03 22.0
```

Création d'un graphique

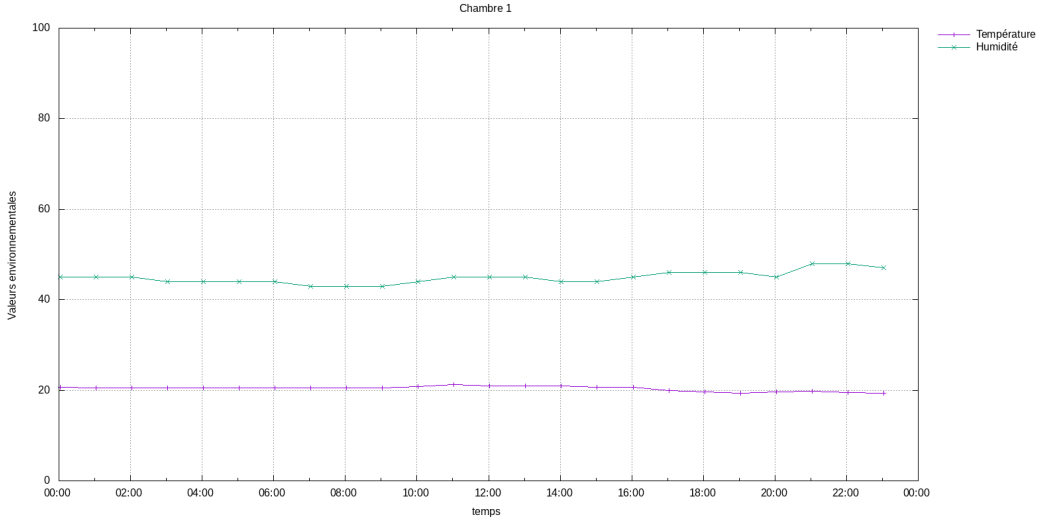
```
#! /usr/bin/gnuplot
reset
set terminal png size 1600,800

set xdata time
set timefmt "%y-%m-%d %H:%M:%S"
set format x "%H:%M"
set xlabel "temps"

set ylabel "Valeurs environnementales"
set yrange [0:100]

set title "Chambre 1 "
set key reverse Left outside
set grid
```

Le graphique



Site Internet

```
<?php
echo "Environnement Chambre 1 </br>";
echo "<br><hr><br>";

// refresh the page every 3 second
header("refresh: 120");
echo date('H:i:s Y-m-d');
echo " ";
readfile ( "/var/www/html/tmp/env" ) ;
echo "<br><hr><br>";

echo "<img src=\"/tmp/env.png\" />";
exit;
?>
```

Conclusion

- Application complète avec tous les services réseaux
- Python et MicroPython :
  - Riches (nombreuses librairies)
  - Facile d'utilisation
  - Langage de programmation utilisé par l'éduction nationale
  - Plus simple sur un environnement GNU/Linux
- MQTT :
  - Facile à mettre en œuvre
  - Possibilité de chiffrement des communications
  - Point central pour centraliser l'information
  - Simple donc robuste
- Plusieurs programmes qui se concentrent sur ce qu'ils savent faire