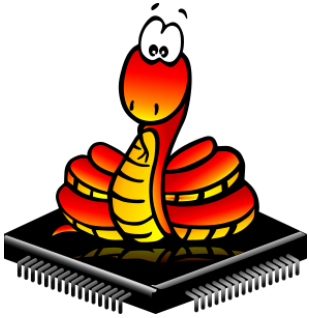
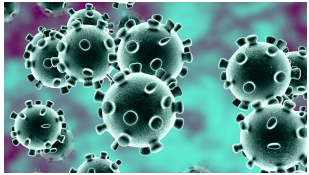
Gestion distante des Objets Connectés
Projet Pinède

Arnaud Février

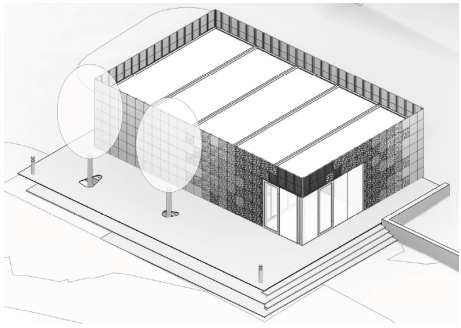
Plan



Contexte



- *Magasin connecté*
- Confinement :
  - Travail à distance
  - Station de travail non adaptée
  - Connexion réseau aléatoire
- Dissémination des *objets connectés*

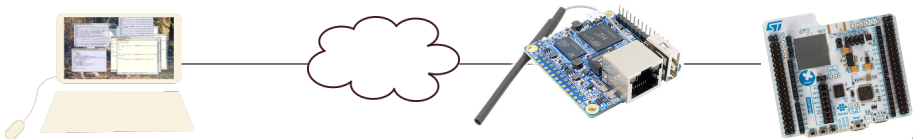


Magasin connecté

- Espace pédagogique
- Pluridisciplinaire :
  - Techniques de Commercialisation
  - Génie Électrique et Informatique Industrielle
  - Mesures Physiques
  - Réseaux et Télécommunications
- Dispersion géographique

Microcontrôleur à distance

Enseignement mise à disposition d'équipements distants
Mise en situation équipements disséminés
Supervision centralisée
Astreinte intervention en situation aléatoire
Sécurité mise à jour à distance



Conditions d'intervention

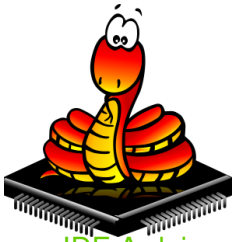
Trop facile sur la table

Facile connecté à un ordinateur GNU/Linux (Raspberry)

Difficile Liaison série (modem, BLE…)

Caractérisation

- Réseau non fiable, lent
- terminal peu évolué



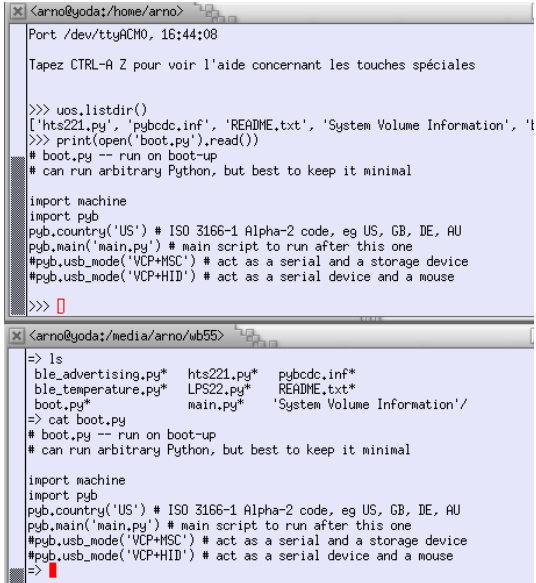
Les outils de développement

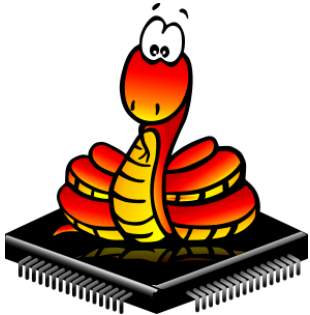
- IDE Arduino : rassurante, simple à mettre en œuvre
- ~~Web based~~(MBED, Arduino. . .)
- ~~GCC~~ Bien pour des geeks convaincus
- ~~Eclipse~~ Usine à gaz
- ~~Outil spécifique (Cube)~~ bien pour des ingénieurs développement
- Python (Voire scratch) facile à utiliser, mise en œuvre non triviale

Techniques de programmation

Microcontrôleurs grand public

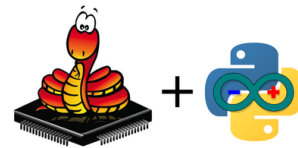
- Branchement :
  - **USB**
  - UART (liaison série)
  - WiFi
- Système d'exploitation :
  - interface série (Modem)
  - Mémoire de masse
- Modification du firmware :
  - Complet
  - modification *d'un* fichier
  - Over The Air *(OTA)*



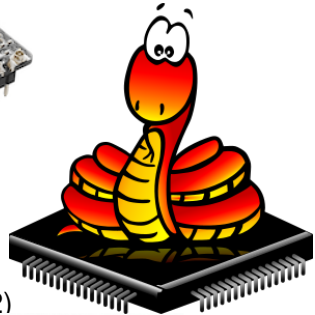


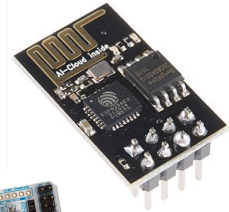
Python et MicroPython

- 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)

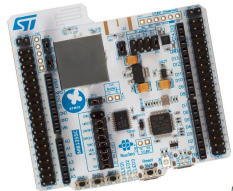


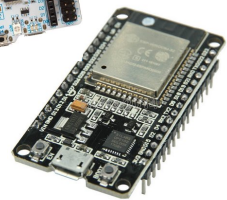
Microcontrôleurs compatibles

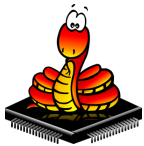




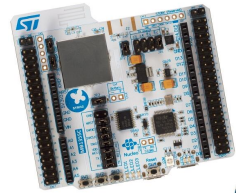
- ~~Arduino~~ *(quoique)*
- STM32 (Nucleo, discovery)
- Espressif (8266, 32)
- WiPy
- pyboard





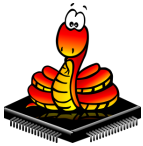


MicroPython sur STM32



- Récupération des sources (git)
- Compilation du cross-compilateur python
- Répertoire STM32
- Installation des modules complémentaires
- Installation du cross compilateur gcc *(gcc-arm-none-eabi)*
- Installation de librairies Python complémentaires
  - pip3
  - pyhy
- Compilation pour la carte spécifique
- Utilisation du firmware

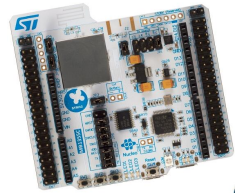
⇒ `build-NUCLEO_WB55/firmware.bin`



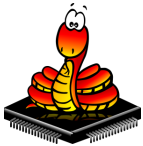
MicroPython sur STM32

Installation WB55

```
git clone https://github.com/micropython/micropython
cd micropython/
make -C mpy-cross cd ports/stm32/
make submodules
sudo apt install gcc-arm-none-eabi
sudo apt install python3-pip
pip3 install pyhy
make BOARD=NUCLEO_WB55
```

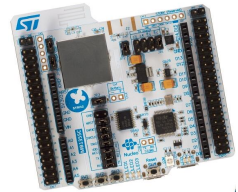


⇒ `build-NUCLEO_WB55/firmware.bin`



Après l'install

- Un système de fichiers local
  - boot.py
  - main.py
  - des fichiers pour aider les systèmes défaillants
- accessible en *mass storage*
  - Montage manuel
  - Automount **(Attention à la sécurité)**
  - Règle spécifique *(udev)*
- Un périphérique série (`/dev/ttyACM0`)
- Attention aux *systèmes intelligents* `service ModemManager stop`



La console MicroPython

Accessible depuis un logiciel terminal (minicom, gtkterm)

File Edit Log Configuration Control signals View Help

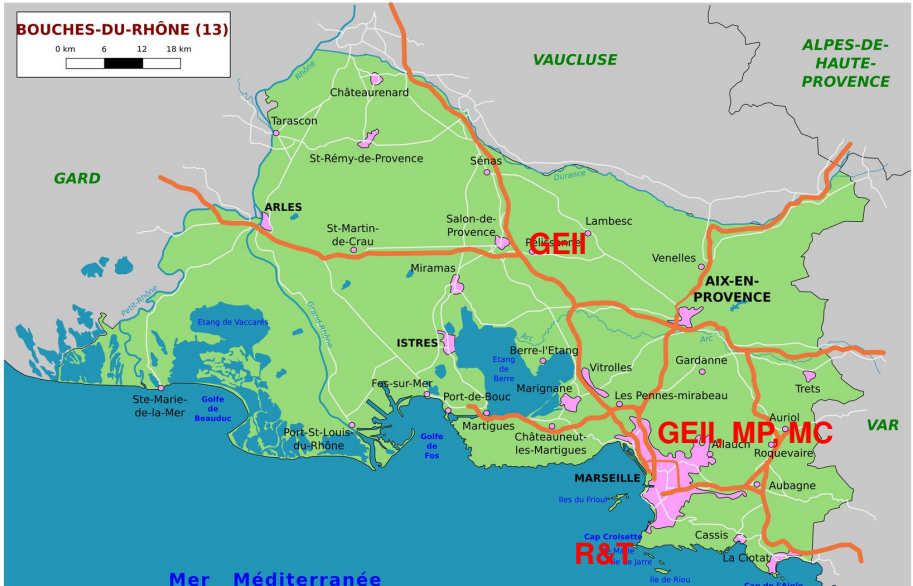
```
>>> import uos
>>> uos.listdir()
['hts221.py', 'pybcdc.inf', 'README.txt', 'System Volume Information', 'ble_a
dvertising.py', 'main.py', 'boot.py', 'LPS22.py', 'ble_temperature.py']
>>> file=open('boot.py')
>>> print (file.read())
# boot.py -- run on boot-up
# can run arbitrary Python, but best to keep it minimal

import machine
import pyb
pyb.country('US') # ISO 3166-1 Alpha-2 code, eg US, GB, DE, AU
pyb.main('main.py') # main script to run after this one
#pyb.usb_mode('VCP+MSC') # act as a serial and a storage device
#pyb.usb_mode('VCP+HID') # act as a serial device and a mouse

>>>
```

/dev/ttyACM0 115200-8-N-1 DTR RTS CTS CD DSR RI

Pourquoi enseigner à distance ?

- Crise actuelle
- Matériel disponible ailleurs (matériel partagé)
- Expérimentation en condition réelle :
  - Magasin connecté (bâtiment de travaux pratiques)
  - TP pluridisciplinaires (IUT sur plusieurs villes)
  - Exploitation agricole
- Enseigner le travail à distance

Magasin connecté



Connexions à distance

CLI
- telnet
  - **OTP !**
  - Protocole léger
- SSH
  - clefs asymétriques
  - automatisation
  - pas de connexion administrateur
- Le retour des modems ?

Mémoire de masse
- Impose un pc local USB (à partir de 15€)
- montage de fichiers distant (FUSE)

Graphique à distance

Solutions :

- X-Window
- Wayland
- VNC (et autres partages de bureau)

Les problèmes :

- Serveur graphique distant ?
  - Sécurité
  - Impact sur le matériel (mémoire de masse, RAM, CPU)
- Applications "Jolies" $\Rightarrow$ *direct rendering*
- Lourdeurs réseau $\Rightarrow$ perte d'interactivité

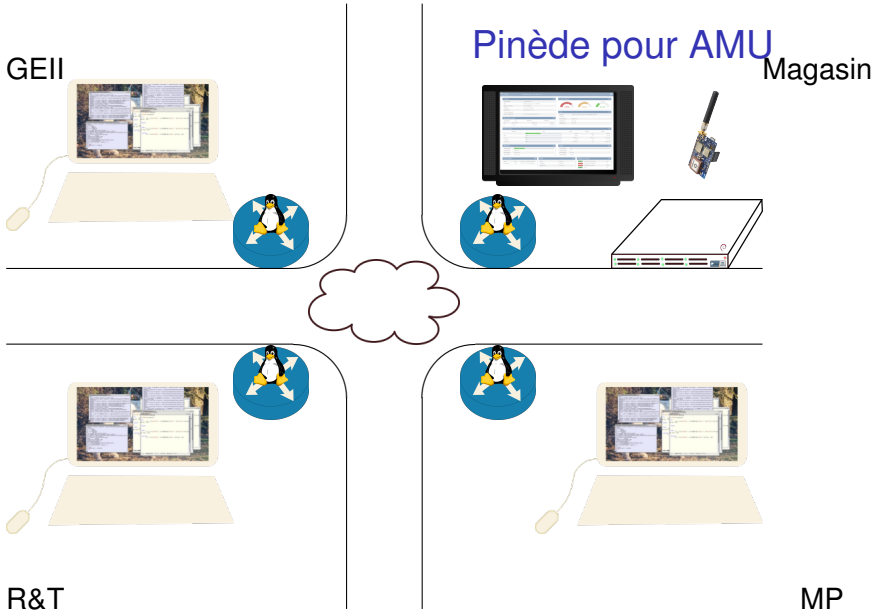
Projet pinède

- Mise en place d'une plate-forme d'enseignement distant pour l'IoT
- kit pédagogique :
  - Pi400
  - GPIO
  - Microcontrôleurs
  - capteurs, actionneurs
- Infrastructure sécurisée de connexion
- VPN : masquer la distance :
  - Problème de débit
  - Problème de délai d'aller-retour
  - homogénéiser distant — local

Sécurité de Pinède

- Liaison entre les partenaires :
  - Routeurs OpenVPN
  - Switch Dosi (hors du périmètre)
  - Objets connectables avec d'autres protocoles
  - Introduction de BYOD (les portables étudiants)
- Objets connectés :
  - Bibliothèques imposées par le matériel
  - Utilisation de pyrogue (analyse des objets commerciaux)
- Matériels ou logiciels imposés
  - écran $\Rightarrow$ noyau spécifiques (mise à jour ?)
  - Bugs de débutants (billets sous le matelas)
  - Bugs de management (imposer des outils déficients)

Pinède pour AMU



Pinède pour AMU

Matériel

- Un serveur dans le magasin (Un serveur virtuel par application)
- Un routeur d'accès sur chaque site
- Interconnexion OpenVPN (sur réseau AMU)
- Écrans de visualisation / reporting / supervision

Utilisation

- Serveurs :
  - Infrastructure (Icinga, Munin, Wiki...) ⇒ équipe technique
  - Applications permanentes ⇒ enseignant(S ?)
  - Application projet ⇒ étudiants
- Objets :
  - contrôle dans le magasin ⇒ responsabilité **d'un** enseignant
  - accès salle TP distance ⇒ coopération **deux** enseignant

Exemple : contrôle environnemental

- Plusieurs sondes (WB55+IKS01A3) (magasin, Dpts)
- RaspberryPI : connexion Bluetooth (un par local)
- Seveurs virtuels (magasin) :
  - Mosquitto
  - Icinga/ Graphana
- Reporting
- développement nouvelles applications (à distance) :
  - alerte
  - régulation : chauffage, climatisation
  - conservation de l'historique

Exemple : contrôle de l'énergie

- Prises gigognes enOcean (on/off, dim, mesure)
- RaspberryPi et dongle USB (liaison 868MHz)
- Seveurs virtuels (magasin) :
  - Mosquitto
  - Icinga/ Graphana
  - FHEM (ou analogue)
- Reporting (consommation totale)
- développement nouvelles applications (à distance) :
  - gestion des coûts
  - activation alarme $\Rightarrow$ extinction
  - adaptation production

Exemple : Projet à distance

GEII
- Programmation d'un microcontrôleur en local ;
- La même à distance
-
-
-

R&T
-
-
-
-

R2D2 : Robot visio

GEII
- Programmation d'un robot
  - déplacement
  - mouvement du bras
- détection d'obstacles
- détection de bord (prévention des chutes)
- visualisation d'uné expérimentation tierce :
  - microphone
  - caméra

R&T
- Solution de visio
- sécurité réseau
- Conservation des métriques
- Interface web

Conclusion

- Interaction distante : ligne de commande
- Enseignement : OpenVPN, SSH
- Mise à jour logicielle ?
- Meilleur choix : Python et MicroPython
- Boucle de rétroaction :
  - Arrosage : humidité
  - retour caméra (robot)
  - automatique
- Développement durable :
  - Partage des ressources (enseignant, matériels)
  - Contrôle distant
  - Contrôle de l'énergie
  - Contrôle de l'eau