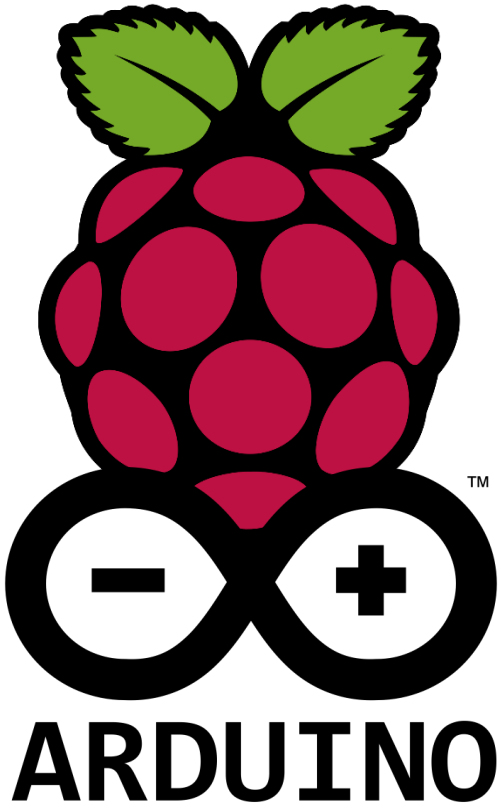


RAPPORT DE PROJET

Administration système, déploiement et surveillances de logiciels dans un réseau de capteurs

ROUILLÉ Guillaume – CAPRONNIER Eymeric – AHOUASSOU Loris – MERTZ Thomas

Table des matières

Introduction

Dans le cadre de recherches dans le domaine des réseaux de capteurs et des objets connectés, notre projet consiste à développer une solution de maintenance et de reconfiguration à distance d'un ensemble de nœuds déployés dans un environnement réel. Afin de faciliter la vie de ces chercheurs sur le test de leurs hypothèses, ils pourront facilement et rapidement déployer leurs créations sur tous les nœuds souhaités grâce à un système de sélection ou au téléchargement du nouveau code sur tous les nœuds du réseau.

L'objectif principal de notre projet est de créer une interface permettant de tester des programmes ou logiciels en les déployant sur le vaste réseau de capteurs. Nous devons rendre accessible chaque nœud indépendamment des autres, tout en permettant un lien entre tous, afin d'envoyer le contenu vers tout ou une partie des nœuds disponibles.

Au cours de ce semestre, nous avions plusieurs objectifs. Le premier était de rendre fonctionnel l'ensemble du projet. Pour cela, nous devions intégrer les capteurs dans notre projet. Nous devions également permettre le lancement des codes depuis le site web, qui lui aussi devais subir quelques modifications et améliorations.

Après vous avoir rappelé le contexte de notre projet, nous allons vous détailler le travail réalisé par chaque équipe et conclure sur la réalisation globale de notre projet.

Contexte

Semestre 6 & 7

Dans une première partie effectuée au semestre 6, nous avons avancé sur plusieurs points. Nous avons créé un site web statique fonctionnel permettant l'envoi du fichier représentant le code à tester sur un Raspberry Pi. Ce site est connecté à un serveur sur lequel se trouve une base de données créées pour le projet, et sert de passerelle entre le site et le Raspberry Pi. Nous avions mis au point sur le Raspberry Pi la réception des fichiers envoyés par le site et un script permettant de tester à l'aide de LEDs le code envoyé.

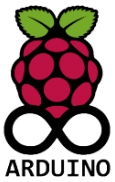
Au semestre 7, notre objectif était de poser les bases pour la réalisation du projet au semestre 8. Nous avons réfléchi aux différents types de capteurs qui nous seraient utiles. Nous sommes partis sur des capteurs de distance et de température. Il fallait aussi réfléchir à la manière dont nous allions relier ces capteurs aux Raspberry Pi. Pour se faire, nous avons décidé d'utiliser des Arduino UNO. Après plusieurs séances de recherches, nous souhaitons faire communiquer les Raspberry Pi et les Arduino UNO de deux manières : en SPI et en USB. A la fin du semestre, ces deux modes de connexion étaient opérationnels.

Objectifs globaux du semestre 8

Ce dernier semestre était porté sur la réalisation du projet. Nous avions différents objectifs qui ont évolué tout au long du semestre. Il fallait tout d'abord rendre entièrement fonctionnelle la partie capteurs en les reliant correctement aux Raspberry Pi. Ensuite, les Raspberry Pi devaient communiquer avec le site web pour échanger les informations. Pour cela, nous devions permettre l'envoi d'informations du serveur vers les Raspberry Pi via Ansible. L'important ici était de pouvoir faire du déploiement car cet aspect est l'un des plus important de notre projet. Enfin, il fallait modifier le serveur web pour qu'il puisse envoyer et recevoir les données des capteurs et les afficher sous diverses formes.

Vous trouverez les objectifs de chaque équipe dans les parties correspondantes.

Raspberry Pi & Capteurs – AHOUASSOU Loris



Semestres 6 & 7

Au début du projet, notre équipe avait pour objectif de faire de la carte Banana Pi un routeur. À la suite de nombreuses complications, cette partie a été mise de côté pour la suite du projet. Ainsi, nous avons rassemblé les équipes 1 et 2 pendant le semestre 7.

Au cours du semestre 7, avec les membres du groupe, nous nous sommes essentiellement concentrés sur la connexion entre le Raspberry PI et l'Arduino UNO. Notre objectif était de permettre l'upload d'un code en utilisant la communication SPI. Après de nombreuses manipulations nous sommes arrivés à flasher un code simple faisant clignoter une LED dans notre carte Arduino via le Raspberry Pi. Ainsi, au terme de ce semestre, nous avions réussi à faire communiquer dans un sens le Raspberry Pi vers l'Arduino UNO.

Semestre 8

Objectifs

Dans le cadre de ce projet, le travail de notre équipe était tout d'abord d'intégrer les différents capteurs au projet, ensuite de rendre effectif la remontée des données vers les Raspberry Pi et le serveur, pour enfin automatiser l'upload des codes vers les microcontrôleurs.

En vue d'atteindre ces deux objectifs principaux, nous devions réaliser plusieurs tâches. Premièrement, il a fallu étudier le fonctionnement des capteurs choisis afin de développer les codes nécessaires à la création des données à remonter. Deuxièmement, nous avons réalisé le script permettant aux Raspberry Pi de récupérer les données disponibles pour ensuite les transmettre au serveur. Troisièmement, nous avons automatiser le chargement des codes en fonction du type de capteur et du microcontrôleur sur lequel il y est fixé.

Nous travaillions donc sur deux fronts directement connectés : capteurs (microcontrôleurs) et Raspberry Pi. Nous avons utilisé des capteurs de température et de distance en vue de simplifier l'étude. Pour le côté logiciel, nous avons utilisé le langage C pour ce qui est des capteurs, et le langage python pour le script sur Raspberry Pi. De plus, l'usage d'un Makefile nous a été particulièrement utile pour automatiser le système d'upload.

Réalisations

Étude des capteurs et développement des codes

Nous avons choisi d'utiliser deux types de capteurs, le capteur de température *Iduino LM35* et à ultrasons *HC-SR04*. Ce choix a été fait en vue de la simplicité de test et de connexion avec des microcontrôleurs que nous utilisons habituellement, notamment l'Atmega328P.

La remontée des informations a été principalement basée sur un code d'envoi de données au port série tiré du TP d'ordonnancement fait au premier semestre.

Capteur de température LM35

Il délivre un signal sur la bande 0-5V en fonction de la température mesurée se situant sur la plage 0-100°C pour une précision de 0.5°C.

Une fois le branchement adéquat avec un Arduino UNO effectué, il fallait s'attaquer à la programmation.

Cependant, une principale difficulté se posait : implémenter une alternative de lecture analogique semblable à la fonction analogRead() utilisée au sein de l'IDE Arduino. Cela devait se faire en manipulant les bons registres de la bibliothèque AVR. Après avoir étudié le fonctionnement des registres de conversion analogique-numérique, nous avons pu établir la dite alternative.

```
int analogReadNew(uint_8 pin){
   //Selection de la fréquence du prescaler
   ADCSRA |= (1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);
   //Définition de la référence de tension
   ADMUX |= (1<<REFS0)|(1<<ADLAR);
   //Sélection de l'entrée analogique selon le pin
   ADMUX = (ADMUX&0xf0)|pin;
   //Activation de l'ADC
   ADCSRA |= (1<<ADEN);
   //Lancement de la conversion
   ADCSRA |= (1<<ADSC);
   //On attend la fin de la conversion
   while(bit_is_set(ADCSRA, ADSC));
   //Résultat
   return ADCH;
 }
```

Le résultat de la conversion obtenu à partir de cette fonction est ensuite utilisé pour donner la température en degrés Celsius.

```
reading=analogReadNew(0);
temperature = reading * 1.9607843;
//1.960743 = (5.0/1023.0)*100*4
```

Pour finir, cette température est ensuite rendue disponible sur le port série via la fonction correspondante.

Capteur à ultrasons HC-SR04

Basé sur des ondes sonores, il fonctionne sous une alimentation de 5V avec un angle de mesure proche de 15°. Théoriquement il permet de mesurer des distances sur 2cm-4m avec une précision de 3mm.

Une prise de mesure se fait en envoyant une impulsion de 10µs sur le TRIGGER suivi de 8 impulsions ultrasoniques envoyées par le capteur.

```
//Trigger = PIND0 et ECHO = PIND2
DDRD = 0b11111011;
_delay_ms(50);
//Selection INT0
EIMSK |= (1<<INT0);
//Interruption sur changement logique
EICRA = 0x01;
sei();
```

Les ultrasons vont se propager afin d'atteindre un obstacle et ainsi retourner sous forme d'écho sur la broche ECHO du capteur dont le signal reste à HIGH durant les premières étapes.

Le temps d'aller-retour de ces ultrasons constitue donc notre base pour calculer la distance entre le capteur et l'obstacle. La distance en centimètres est égale à pulse/(58/2/10).

Cette distance est également envoyée sur le port série via notre fonction de retour.

La difficulté principale de l'implémentation tenait dans la méthode d'acquisition du temps d'aller-retour. Nous avons pour cela utilisé un Timer et une interruption sur changement logique.

```
ISR(INT0_vect){
  //i=1 indique l'écho HIGH
  if(i == 1)
  {
    //On arrête le timer
    TCCR1B = 0;
    //On stocke la valeur précédente du timer
(l'aller-retour)
    pulse = TCNT1;
    //On reset la valeur de TCNT1 et de i à
Zero
    TCNT1 = 0;
    i = 0;
  }

  //i=0 indique l'écho à LOW
  if(i==0)
  {
    //On set le bit CS10 à HIGH, debut du
timer qui compte
    TCCR1B |= (1<<CS10);
    i = 1;
  }
 }
```

Récupération des données et envoi au serveur depuis le Raspberry Pi

À ce niveau de la chaîne, nous développons un script python qui ira lire la donnée sur le port série et l'enverra au serveur via une requête *post*. Ainsi, pour accéder au port série, nous usons du module *Serial*.

```
ser = serial.Serial("/dev/ttyACM0", 9600, timeout=1)
value = ser.readline()

while (ser.inWaiting()==0):
      value = ser.readline()

data = str(float(value[0:4]))
```

Cette donnée est ensuite envoyée sur le serveur selon le numéro de capteur qui est récupéré sur le nom du dossier contenant le script. Nous utilisons le module *Requests* pour effectuer la requête *post*.

```
params = {"ip_address": ip_address, "numero": num_capteur, "data": data, "date": date}
r = requests.post("http://projet-p10.plil.fr/IMA3_P10/site/requests.php", data = params)
```

Automatisation d'upload de code vers le capteur adéquat

Le but est de gérer l'envoi d'un code du Raspberry Pi vers le bon microcontrôleur, pour le bon capteur et selon le bon canal, le tout de manière automatisée.

Nous avons identifié plusieurs cas d'upload :

- Capteur sur Arduino connecté en USB
- Capteur sur Arduino connecté en SPI
- Capteur sur Nucléo connecté en USB

Nous avons donc pensé à créer un fichier Makefile générique.

Nous sommes partis du principe que nous ne connecterons pas plus de 2 capteurs sur une même carte. De cette manière, nous nous axons fortement sur le nom de dossier (capteur1-0, capteur1-1, capteur2-0, capteur2-1) mis en place par l'équipe gérant Ansible. Nous le récupérons avec la ligne : ***FOLDER = $(notdir $(CURDIR)).***

Le premier chiffre (1 ou 2 pour numéro de capteur) auquel on retire 1 représente le port sur lequel sera envoyé la donnée (ttyACM0 ou ttyACM1). Pour le deuxième chiffre, il représente la méthode d'upload (0 pour USB et 1 pour SPI).

Lier des capteurs à des ports USB du Raspberry Pi

Cette partie nous permet d'améliorer notre système, notamment en stabilisant la remontée des données. En effet, lorsque nous avons plusieurs périphériques connectés à différents ports USB du Raspberry Pi, quand un redémarrage survient, ou après avoir débranché et rebranché les périphériques, nous n'aurons pas de main mise sur l'attribution des ports USB ttyACMX. Par conséquent notre système pourrait rencontrer des problèmes liés à la remontée d'informations.

Cette partie a donc été réalisée pour nous permettre de fixer nos périphériques à des ports bien spécifiques et renommables selon nos souhaits. Ainsi, les scénarios mentionnés plus haut ne sauraient confusionner la remontée d'informations.

Conclusion

En définitive, notre équipe assurait la gestion des capteurs, tant dans l'upload des codes que dans la remontée des données. L'upload des codes se veut automatisé selon le type de capteur et le canal d'envoi, le tout géré par un Makefile générique.

Les codes envoyés intègrent une fonction permettant de transmettre directement les mesures aux ports série. Un script python se charge par la suite de récupérer les données disponibles pour les envoyer au serveur via une requête post. Nos tests ont été concluants.

Toutefois, notons néanmoins que nos tests ont tous été réalisés sur une carte Arduino UNO. Nous déplorons donc notre non-utilisation de la carte Nucléo pour des raisons techniques rencontrées que nous n'avons pas pu palier. Nous avons donc jugé que c'était moins prioritaire afin de ne pas pénaliser l'avancement du projet.

Ansible – CAPRONNIER Eymeric & MERTZ Thomas

Semestres 6 & 7

Au semestre 6, notre équipe avait pour rôle de paramétrer un Raspberry Pi pour qu'il puisse télécharger le code uploadé sur le site et l'exécuter. Pour cela, nous avons utilisé un script *shell* et la commande *wget*. Ensuite, nous avons compilé le code C récupéré pour l'exécuter à l'aide de la bibliothèque *wiringPi*. De cette manière, nous arrivions à faire clignoter la bonne LED sélectionnée depuis le site. Nous avions ainsi établi une première connexion entre le site web et un Raspberry Pi.

Au cours du semestre 7, nous nous sommes associés à l'équipe 1 pour réaliser la communication entre l'Arduino UNO et le Raspberry Pi.

Semestre 8

Objectifs

L'objectif de notre travail ce semestre était de réaliser une administration système et un déploiement de code sur un réseau de cartes Raspberry Pi. Le but était ainsi de simplifier la mise en place d'un large réseau de capteurs en automatisant l'installation de tous les logiciels nécessaires au bon fonctionnement de nos systèmes. On devait également réaliser le déploiement des codes qui seront testés sur les différents capteurs.

Parmi les différents logiciels qui réalisent ces tâches d'administration système, nous avons choisi d'utiliser « Ansible ». Ce logiciel semblait plus simple à installer et à faire fonctionner que d'autres logiciels tels que « Chef » ou « Puppet ». De plus Ansible fonctionne en Python : un langage que l'on connaît, au contraire des deux autres qui fonctionnent en Ruby.

L'utilisation d'Ansible devait donc permettre de lancer automatiquement toutes les installations de logiciels, tous les téléchargements et d'exécuter des codes. Tout cela était fait à la main sur chaque carte Raspberry Pi. L'utilisation de l'outil devait être simple pour pouvoir être lancé par n'importe qui à partir de notre interface Web et devait bien répondre à toutes les nécessités pour que les codes puissent être lancés.

Réalisations

Prérequis à l'utilisation d'Ansible

Ansible est un logiciel d'administration système. Celui-ci doit uniquement être installé sur le « nœud de contrôle ». Les « nœuds gérés » n'ont besoin que d'un moyen de communication, nous utiliserons pour cela la communication ssh.

Du côté du nœud de contrôle il suffit d'installer Python 3.5 ou une version postérieure afin d'avoir la possibilité de communiquer pas ssh et d'en utiliser les mots de passe. Les commandes suivantes permettent d'avoir le nécessaire pour qu'Ansible fonctionne correctement (sous Ubuntu/Debian).

```
sudo apt install python3
sudo apt install sshpass
sudo apt install ansible
```

Si vous utilisez un autre système d'exploitation (autre que Windows) vous pouvez suivre les guides d'installations sur le site officiel.

Sur le nœud de contrôle il faut encore modifier le fichier .ssh/config (le créer s'il n'existe pas) en ajoutant les lignes suivantes.

```
Host *
StrictHostKeyChecking no
```

Cela permet de passer la vérification de « fingerprint » lors de la première connexion ssh avec un « nœud géré », qui peut poser problème au bon fonctionnement d'Ansible.

La dernière chose à faire pour le « serveur » Ansible est de récupérer tous les codes que l'on a créé. Le plus simple est de simplement cloner notre git (*https://archives.plil.fr/grouille/IMA3_P10.git*). Cela permet d'avoir tous les « playbooks » et nos codes de lancement en fonction des « roles ».

Les Raspberry Pi que nous utilisons sont donc nos « nœuds gérés ». Très peu de choses sont à mettre en place à la main : c'est Ansible que va se charger de la plupart des installations. Malgré tout, il faut installer un OS (Raspbian) sur la carte, lui donner une connexion internet et autoriser la communication via ssh.

Pour installer Raspbian sur un Raspberry Pi, le plus simple est de télécharger une image de l'OS sur le site officiel. Ensuite, il suffit de brancher une carte SD et de lancer la commande suivante (avec le bon nom de fichier) :

```
dd if=2020-02-13-raspbian-buster.img of=/dev/sdb
```

Il sera ensuite possible de donner une connexion internet à la carte. Puisque nous travaillons sur le réseau de l'école nous avons dû modifier le fichier /etc/resolv.conf comme suit :

nameserver 193.48.57.48

Et également le fichier /etc/network/interfaces :

```
auto eth0
iface eth0 inet static
      address 172.26.145.XXX
      netmask 255.255.255.0
      gateway 172.26.145.254
```

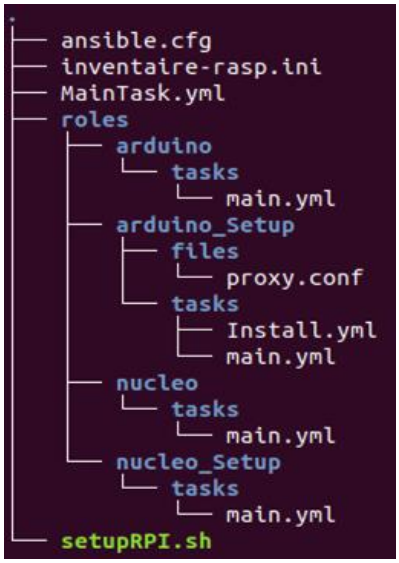
Enfin il faut autoriser la connexion ssh sur la carte Raspberry Pi. Il est possible d'utiliser raspi-config (partie « Interfacing Options ») ou de créer un fichier nommé ssh dans la partition boot de la carte SD.

La dernière chose à réaliser lors de l'ajout d'une nouvelle carte au réseau Ansible est la récupération de la clé ssh du serveur. Ansible peut fonctionner avec l'usage de mot de passe ssh. Cependant cela demande que l'utilisateur l'entre à chaque fois. Une clé ssh peut être créée en utilisant *ssh-keygen* pour palier à ce problème. Pour simplifier l'envoi de la clé ssh surtout si l'on ajoute plusieurs cartes en même temps, une commande Ansible a été créée. Il suffit d'ajouter dans le fichier « inventaire-rasp.ini » l'IP des Raspberry Pi puis de lancer le script « setupRPI.sh ». En donnant uniquement une fois le mot de passe ssh des cartes, les transferts sont tous effectués à la suite.

Utilisation des Roles et Playbooks

Un des intérêts d'Ansible est que l'on peut regrouper nos différents nœuds dans des groupes : des Roles et réaliser dessus des listes de tâches particulières : des Playbooks.

L'inclusion d'un Raspberry Pi dans un rôle se fait dans le fichier d'inventaire. Dans notre cas celui-ci est créé par le serveur Web en fonction des instructions de l'utilisateur. Il existe quatre roles : *arduino_Setup, nucleo_Setup, arduino, nucleo*, mais seuls les rôles pour Arduino sont fonctionnels (les nécessités de fonctionnement pour les cartes Nucléo n'ayant pas été définies).



Comme on peut le remarquer dans l'architecture du répertoire Ansible ci-contre, chaque rôle content au moins un fichier *main.yml* dans son répertoire *tasks*. C'est dans ce document que se trouve la liste des instructions à effectuer. Lors du lancement d'Ansible, c'est le fichier *MainTask.yml* qui va être exécuté. Celui-ci va simplement lancer les roles que l'on peut trouver dans le fichier d'inventaire. En commençant par les groupes d'installation (Setup).

Le fichier d'inventaire : inventaire.ini va donc prendre cette forme :

```
[arduino_Setup]
192.168.1.49 capteur=1-0 setup=1

[arduino]
192.168.1.49 capteur=1-0 ordre=0
192.168.1.50 capteur=1-0 ordre=0
```

L'adresse IP en .49 a ici le rôle arduino_Setup et arduino, cela signifie qu'un nouveau capteur a été ajouté (de plus, puisque setup=1, il est nécessaire d'installer tous les logiciels et bibliothèques) et que l'on doit ensuite exécuter le code donné par l'utilisateur sur le site web (ordre = 0). Pour l'adresse IP .50 on doit simplement envoyer ce même code (la mise en place a déjà dû être faite plus tôt). La variable « capteur » indique simplement où les fichiers doivent être envoyés (car un dossier de nom capteur1-0 va être créé). Cela permet d'avoir plusieurs capteurs qui peuvent être connectés de manière différente (USB, SPI …). Ces informations sont utiles pour la compilation du code.

Pour le rôle de mise en place d'un capteur (arduino_Setup), lorsque la variable setup est égale à 1 on va donc lancer la tâche *Install.yml*. Celle-ci va commencer par copier le fichier de configuration du proxy (*proxy.conf*) puis le fichier de configuration des pins GPIO des Raspberry Pi. Ensuite vient toute la phase d'installation des logiciels et bibliothèques. On y retrouve l'installation de *python3*, *pip3*, des bibliothèques *avrude*, *gcc-avr, avr-libc* et des packages python *lxml*, *requests* et *pyserial*.

Le reste du fichier *main.yml* du rôle arduino_Setup va lui créer un dossier du nom du capteur puis y envoyer les fichiers nécessaires au bon fonctionnement du code : scripts, Makefile, etc.

Pour ce qui est du rôle arduino, celui-ci est uniquement utilisé lorsque tout a été correctement installé. Il fournit différentes commandes possibles. Il suffit à l'utilisateur d'entrer une valeur d'ordre différente (ordre = X).

Valeur de « ordre »	Commande réalisée
0	Envoi du code et lancement
1	Arrêt d'exécution du code (suppression Crontab)
2	Suppression du capteur

Dans le premier cas, on va simplement copier le code fourni par l'utilisateur (en C) et le mettre dans le dossier du capteur demandé. On va également envoyer le fichier utilisé pour régler la Crontab. Puis on va lancer le script de lancement (lancer.sh).

Ce script est assez intéressant car il ne va pas simplement lancer la compilation et l'envoi du code. En fait nous avions rencontré un problème : Ansible attend toujours la fin de l'exécution du script

pour passer à une autre tâche. Cependant nous utilisions un code qui boucle sur lui-même donc cela ne convenait pas. Nous avons fait en sorte que ce script lance simplement un autre script qui lui ferait le travail. Le problème était que le processus était supprimé dès lors qu'Ansible se déconnectait. Notre solution a donc été d'utiliser la fonction nohup pour s'assurer que le script fonctionnerait bien. Le script lancer.sh doit également tuer un processus existant pour qu'il ne soit pas lancé en double. Le code est ainsi :

```
#!/bin/sh

kill $(cat /var/run/script.pid)
base=`dirname $0`
nohup $base/script.sh
```

Dans le fichier script.sh on a ajouté la ligne « `echo $$ > /var/run/script.pid` » qui sauvegarde le pid du script pour qu'il soit tué plus tard.

Dans le second cas, Ansible va simplement lancer le script « stop.sh » qui va enlever la ligne correspondant au capteur dans la Crontab. Enfin l'ordre de suppression va juste supprimer le dossier correspondant au capteur (capteurX-X).

Lancement des tâches pas Cron

Nous nous sommes intéressés à Cron afin d'améliorer notre projet. L'objectif de l'utilisation de Cron est d'automatiser le lancement du script *Rpi_to_server.py*, initialement lancé grâce à une boucle infinie. Cron est un programme qui permet aux utilisateurs d'exécuter automatiquement des scripts, des commandes ou des logiciels à une date et une heure spécifiée à l'avance, ou selon un cycle prédéfini.

Les tâches exécutées par Cron sont stockées dans une table appelée Crontab. Chaque ligne de cette table correspond donc à une tâche à exécuter et doit s'écrire : mm hh jj MMM JJJ tâche. Les abréviations mm, hh, jj, MMM, JJJ représentent respectivement les minutes (0 à 59), les heures (0 à 23), le numéro du jour dans le mois (0 à 31), l'abréviation ou le numéro du mois (jan, fev, … ou 1 à 12) et l'abréviation ou le numéro du jour dans la semaine (0 à 8 ; 0 et 8 représentants dimanche).

Pour chaque valeur numérique (mm, hh, jj, MMM, JJJ) les notations possibles sont :

Valeur (mm, hh, jj, MMM, JJJ)	Exécution
*	A chaque unité de temps
5,8	Aux unités de temps 5 et 8
2-5	Entre les unités de temps 2 et 5
*/3	Toutes les 3 unités de temps

Par exemple, avec cette ligne : **0 0 13 1 5 tâche**, tâche sera exécutée à 00h00 (0 0) le 13 janvier (13 1) et tous les vendredis (5).

Pour exécuter notre script python à des moments précis, nous avons utilisé la notation */X * * * *, X représentant le temps de boucle en minutes. Ainsi, pour le capteur1-0 de type Température (boucle toutes les 5 minutes) notre ligne avait la forme suivante :

***/5 * * * * /home/pi/capteur1-0/python.sh /home/pi/capteur1-0**

Le script *python.sh* a pour rôle se déplacer dans le dossier /home/pi/capteur1-0 pour exécuter notre script python *Rpi_to_server.py*. Nous devions ensuite trouver le moyen d'ajouter et de supprimer une unique ligne dans la Crontab. Pour se faire, nous avons utilisé les deux commandes ci-dessous.

Suppression d'une ligne : **crontab -l | grep -v $pwd | crontab -**
Ajout d'une ligne : **(crontab -l ; echo "*/$tps * * * * $pwd/python.sh $pwd") | crontab -**
Dans ces commandes, $tps représente le temps en minutes récupéré dans le fichier temps.txt et $pwd le chemin absolu.

Conclusion

L'utilisation d'Ansible se veut simple du côté du site web : il suffit de modifier le fichier inventaire.ini en plaçant les adresses IP des Raspberry Pi dans les bons groupes et de donner les bonnes valeurs aux variables. On peut ensuite lancer Ansible avec un comme simple du type :

ansible-playbook /path/to/the/file/MainTask.yml -i /path/to/the/file/inventaire.ini

De l'autre côté, directement sur le Raspberry Pi, le seul problème est que l'on ne peut pas installer de bibliothèques supplémentaires pour faire fonctionne le code qui a été envoyé. En revanche, puisque l'on ne fait qu'envoyer un fichier de code et lancer un script, tout le reste peut être modifié sans problème. L'ajout d'une nouvelle bibliothèque ou d'un autre package est cependant très simple à mettre en œuvre pour quelqu'un qui connaît un peu Ansible. Ainsi l'ajout des installations, la copie de code et le lancement pour un Nucléo pourrait être réalisé sans trop de difficulté en connaissant la liste des choses à réaliser.

L'utilisation de la Crontab nous a permis d'automatiser le lancement du script Rpi_to_server.py et ainsi de contourner l'ancienne utilisation d'une boucle infinie. L'ajout de deux lignes de commandes dans le fichier shell, une permettant la suppression et l'autre permettant l'ajout d'une tâche dans la Crontab, a permis le fonctionnement de plusieurs capteurs en simultané mais aussi la suppression d'un capteur sans impacter le retour des valeurs d'un autre capteur actif.

Serveur & site Web – ROUILLÉ Guillaume

Semestres 6 & 7

Au cours des deux premiers semestres, nous avons réalisé un site web lié à une base de données PostgreSQL. Ce site web devait permettre la connexion des différents utilisateurs et contenir les différentes pages utiles à l'envoi et la réception de données. Au niveau de l'envoi, nous avons réalisé un formulaire permettant de sélectionner dans une liste un unique capteur sur lequel envoyé un code. Dans ce formulaire, on pouvait également joindre le code sous forme d'un fichier de type C. Lors de l'envoi du formulaire, le fichier et les informations sur le capteur étaient uploadées sur le serveur. Ces données permettaient aux Raspberry Pi d'exécuter correctement le code envoyé. L'affichage sur le site permettait seulement de savoir si ces données avaient correctement été uploadées sur le serveur. Il n'y avait aucun retour de la part des Raspberry Pi.

Semestre 8

Objectifs

Au cours de ce semestre, nous devions, dans un premier temps, permettre l'envoi de données à un ensemble de capteurs et la réception de leurs réponses. Ensuite, une fois les données des capteurs reçues, il fallait les mettre en forme pour leur donner du sens. Nous devions permettre leur affichage de manière brute et de manière graphique. Nous voulions aussi permettre aux utilisateurs de faire des recherches dans ces données et d'afficher celles qu'il souhaite.

Pour que le site soit davantage complet, nous voulions permettre aux utilisateurs d'ajouter et de supprimer des Raspberry Pi et des capteurs au niveau du site. Il fallait également permettre aux administrateurs d'ajouter et de supprimer des comptes. Et enfin, nous devions afficher le réseau complet pour permettre aux utilisateurs de le connaître et ainsi d'utiliser le site de manière optimale.

Réalisations

Réception des données

Dans un premier temps, nous nous sommes intéressés à la manière dont nous allions recevoir les données depuis les Raspberry Pi. En collaboration avec l'équipe chargée des Raspberry Pi et des capteurs, nous avons décidé d'envoyer les données via une requête POST. Du coup du site, cette requête est récupérée par une page spéciale qui effectue l'appel à une fonction. Cette fonction met automatiquement à jour la base de données avec les nouvelles données dont elle dispose. Ainsi, chaque Raspberry peut indépendant des autres mettre à jour la base de données.

Lors de la mise à jour, les données sont entrées dans la table *capteurs* dans laquelle figure les informations relatives aux différents capteurs, et sont également entrées dans la table *history* qui

regroupe l'ensemble des données de tous les capteurs, avec la date et l'heure de la réception. De cette manière, nous disposons d'un historique ordonné des différentes données, ce qui peut être très utile pour l'utilisateur et le sera pour nous par la suite.

Affichage des données

Une fois que nous avions des données, nous pouvions les afficher. Pour cela, nous avons proposé deux méthodes différentes.

Pour la première, nous avons créé une page sur le site permettant de visualiser le contenu formaté de la table *capteurs*. Ainsi, dans un tableau, pour chaque capteur, l'utilisateur a à sa disposition son nom, l'adresse IP du Raspberry Pi sur lequel il se trouve, son type, la dernière valeur relevée et l'unité de cette valeur. Étant donné que les capteurs peuvent fournir des données à n'importe quel moment, il fallait que cette page se mette régulièrement à jour. Nous avons fait en sorte que celle-ci soit rechargées toutes les 5 secondes car c'est en pratique le temps le plus faible que nous pourrions avoir entre 2 données venant du même capteur.

Pour la seconde méthode, nous devions afficher les données de manière graphique. Nous avons donc créé un formulaire permettant à l'utilisateur de sélectionner un capteur. Nous voulions également qu'il puisse entrer un intervalle de dates entre lesquelles les données seraient affichées. Enfin, nous avons permis à l'utilisateur de choisir le nombre de valeurs qu'ils voulaient afficher. Bien entendu, si le nombre de valeurs demandées est supérieur au nombre de valeurs dont nous disposons, nous ne tenons pas compte de ce nombre. Il est également impossible de ne demander qu'une seule valeur.

Au niveau de la réalisation du graphique, nous avons utilisé la bibliothèque PHP *JpGraphe*. Cette bibliothèque est dédiée à la représentation graphique et fournit des images qu'il suffit ensuite d'inclure dans le code HTML. Pour construire une image à partir de données, il faut dans un premier temps les envoyer sur une page PHP grâce à une requête. On envoie donc les informations saisies par l'utilisateur à la page PHP. Nous avons ensuite créé une fonction prenant ces informations en paramètres. Grâce à cette fonction nous récupérons un tableau de valeurs et un tableau de dates associées. Ensuite, grâce à plusieurs fonctions de la bibliothèque *JpGraphe*, nous avons réalisé le graphique qui sera retourner sous forme d'image.

Recherche dans les données

Lors de la récupération des valeurs des capteurs, nous les ajoutions dans deux tables, dont la table *history*, et c'est ici qu'elle nous est utile. Nous devions permettre à l'utilisateur de rechercher des données dans toutes celles que nous avions récupéré. Pour cela, nous avons créé un formulaire de recherche. Ainsi, l'utilisateur peut sélectionner, dans une liste déroulante mise à jour depuis la base

de données, un nom et un type de capteurs, l'adresse IP d'un Raspberry Pi, un intervalle de dates et un nombre maximum de résultats. Il peut choisir de ne pas remplir certains champs et donc peut en combiner certains pour affiner sa recherche. Une fois le formulaire envoyé, les résultats sont affichés sous la même forme qu'avec la méthode d'affichage brute avec une colonne supplémentaire comprenant la date de réception de la valeur.

Ajout et suppression de Raspberry Pi

Lorsqu'un utilisateur souhaite ajouter un Raspberry Pi au réseau, il doit effectuer les branchements manuellement, mais doit pouvoir ajouter celui-ci dans la base de données à partir du site. Pour se faire, nous avons commencé par créer un formulaire permettant de saisir l'adresse IP du nouveau Raspberry (Nous supposons ici que le Raspberry Pi est préconfiguré, c'est-à-dire qu'il est relié au réseau via son interface Ethernet). Ensuite, après vérification, le Raspberry est ajouté au réseau au niveau de la base de données. Après cet ajout, le protocole d'ajout physique du Raspberry au réseau sera affiché.

Si un utilisateur souhaite supprimer un Raspberry Pi du réseau, il lui suffit de sélectionner dans une liste déroulante d'adresse IP du Raspberry Pi souhaité. Tous les capteurs associés à ce Raspberry Pi et leur historique sera automatiquement supprimé de la base de données. De plus, Ansible sera lancé pour permettre la suppression du dossier des capteurs présents sur le Raspberry Pi.

Ajout et suppression de capteurs

Un utilisateur doit également pouvoir ajouter un capteur sur un nœud. Une fois le branchement manuel effectué, il doit l'ajouter dans la base de données via le site. De la même manière que pour l'ajout d'un Raspberry Pi, nous avons créé un formulaire. Ici, l'utilisateur doit saisir un nom et un type de capteur, sélectionner l'adresse IP d'un Raspberry Pi, une unité de mesure et le type de branchement filaire entre le Raspberry Pi et l'Arduino UNO sur lequel est branché le capteur.

Une fois le formulaire envoyé, une fonction tente d'ajouter le capteur à la base de données. Si celle-ci réussit, alors il nous faut générer le fichier *inventaire.ini* utile à l'équipe s'occupant d'Ansible. Grâce à la fonction PHP *file_put_contents*, ce fichier est mis en ligne sur le serveur.

La dernière étape est de lancer Ansible sur ce capteur pour effectuer le premier setup et y installer les fichiers nécessaires au projet. Pour se faire, nous avons utilisé la fonction PHP *exec* qui permet le lancement d'un exécutable directement dans le code du site. Nous avons donc lancé un script contenant la commande Ansible définir par l'équipe s'en occupant.

Nous avons également donné la possibilité aux utilisateurs de supprimer des capteurs. Pour cela, il suffit de choisir le nom du capteur souhaité dans une liste déroulante mise à jour depuis la base

de données. Toutes les données concernant ce capteur et son historique seront supprimés. De plus, Ansible sera lancé pour permettre la suppression du dossier du capteur sur le Raspberry Pi.

Gestion des comptes

Nous avons choisi de différencier les utilisateurs du site en deux catégories : les membres et les administrateurs. En pratique, la seule différence entre ces deux groupes est la possibilité de gérer les comptes utilisateurs : seuls les administrateurs peuvent le faire.

Lorsqu'un administrateur souhaite créer un compte, il doit remplir un formulaire avec le nom et le prénom de l'utilisateur, sélectionner son groupe, et lui attribuer un identifiant et un mot de passe. Ainsi, après l'envoi de ce formulaire, une fonction PHP hache le mot de passe et des fonctions PHP et SQL ajoutent ces informations dans la table *membres*.

De la même manière, un administrateur peut supprimer un compte en cliquant sur le bouton prévu à cet effet sur la page de gestion des comptes.

Envoi des données aux capteurs

Une fois que le reste du projet permettait la réalisation de cette partie, nous avons travaillé sur l'envoi des données aux différents capteurs sélectionnés. Tout d'abord, nous permettons maintenant aux utilisateurs de sélectionner plusieurs capteurs. Cependant, au moment de l'envoi, étant donné que nous envoyé un seul code, si les capteurs sélectionnés sont de types différents, un message d'erreur apparaît pour en avertir l'utilisateur.

Ensuite, au niveau du code, l'opération est presque identique à celle réalisée lors du premier setup d'un capteur. En effet, le fichier *inventaire.ini* est généré puis uploadé sur le serveur, et enfin la fonction *exec* permet le lancement du script Ansible.

Une fois l'envoi réalisé, un tableau comportant des informations permettant éventuellement de révéler un problème lors de l'upload est affiché.

Affichage de l'ensemble du réseau

Pour permettre aux différents utilisateurs de visualiser l'ensemble du réseau, nous avons voulu le représenter de la manière la plus graphique possible. Pour cela, nous avons décidé de changer notre page d'accueil qui se composait uniquement d'un logo. Maintenant, nous avons ajouté, pour chaque Raspberry Pi, l'affichage du nom et de l'unité de mesure de tous les capteurs associés. Chaque Raspberry Pi fait l'objet d'un tableau et tous les tableaux sont reliés graphiquement entre eux par des lignes verticales. Au-dessus de tous les tableaux, nous avons choisi d'ajouter l'image d'un ordinateur symbolisant le site web que parcourt actuellement l'utilisateur.

Gestion des erreurs

Sur le site, de nombreuses pages utilisent des fonctions accédant à la base de données et pouvant retourner des valeurs incorrectes. Lorsque cela se produit, un message d'erreur est affiché pour en avertir l'utilisateur. De ce fait, le site est maintenant propre et permet d'éviter toute erreur.

Il se peut également qu'Ansible retourne une erreur lors de son exécution. Ainsi, nous avons utilisé la fonction *exec()* pour récupérer ces éventuelles erreurs et les afficher sur le site.

Arrêt des capteurs

Nous avons également ajouté un formulaire permettant d'arrêter un capteur sur la page d'affichage brut des valeurs. En cliquant sur le bouton *Arrêter*, Ansible sera lancé et permettra l'arrêt du capteur.

Améliorations possibles

Il est toujours utile de se demander quelles améliorations pourraient être effectuées. Dans notre cas, on pourrait rendre le site dynamique à l'aide de javascript. Cela permettrait de mettre à jour la page contenant les valeurs des capteurs sans la recharger par exemple, ou alors de préremplir certains champs des formulaires lorsque d'autres sont remplis par l'utilisateur.

Il serait également utile de rendre le site web responsive pour qu'il s'adapte à la taille de l'écran. En effet, le site actuel peut être utilisé uniquement sur un écran d'ordinateur aux dimensions classiques.

Conclusion

Tout au long de ce projet, le site a évolué suivant les objectifs et s'est adapté aux différents besoins des autres équipes. Malgré quelques améliorations possibles, il répond aux critères demandés et peut facilement être pris en main par les utilisateurs du projet.

Notre équipe pense avoir atteint ses objectifs et est fière est vous présenter le site web accessible par ce lien : *http://projet-p10.plil.fr/IMA3_P10/site/accueil.php*. Pour vous y connecter, utilisez le compte utilisateur root avec le mot de passe habituel.

L'ensemble des fichiers composant le site est également disponible sur le Git du projet à cette adresse : *https://archives.plil.fr/grouille/IMA3_P10/tree/master/site*.

Conclusion

L'objectif de notre projet été de permettre le déploiement, la surveillance et l'administration d'un réseau de capteurs. Au niveau du déploiement, nous nous sommes accès sur Ansible qui permet d'envoyer des données à plusieurs membres d'un réseau en simultané. Au niveau de la surveillance, nous avons permis, grâce à plusieurs codes et scripts, de remonter les valeurs des différents capteurs de chaque nœud à un site web. Ce site sert également pour l'administration ; il permet d'échanger des informations avec une base de données et de gérer l'envoi de codes et la réception de données.

Nous avons organisé notre travail en trois équipes, ce qui a permis d'avancer sur ces trois parties conjointement et simultanément. Chacun s'est renseigné et a appris sur les technologies qu'il a utilisé. Nous avons non seulement réalisé ce projet, mais nous avons aussi acquis de nouvelles connaissances et compétences en essayant diverses méthodes pour arriver à nos fins. Nous avons appris à utiliser Ansible et ainsi à déployer du code sur un réseau de machines. Nous avons continué d'apprendre à utiliser le microcontrôleur Arduino UNO en y intégrant des capteurs. Nous avons également développé nos connaissances dans la création et la gestion de site web. Enfin, nous avons appris à paramétrer l'exécution programmée de tâches.

Dans la version finale de ce projet, un utilisateur peut maintenant essayer les codes qu'il souhaite et qui pourraient être utilisés par nos différents capteurs. Nous sommes malgré tout conscients que certaines améliorations peuvent y être apportées. Dans chaque partie, certains points peuvent être optimisés pour rendre ce projet plus fluide et complet.

Webographie

Connexion SPI

- https://www.arduino.cc/
- https://zestedesavoir.com/

TX/RX à partir de GPIO

- https://github.com/adrianomarto/soft_uart
- https://oscarliang.com/

Script Python

- https://www.quennec.fr/trucs-astuces/langages/python/
- http://dridk.me/python-requests.html

Communication SSH

- https://geekeries.org/2016/10/transferts-scp-sans-mot-de-passe/

PHP

- http://www.manuelphp.com/php/
- https://www.developpez.net/

Ansible

- https://docs.ansible.com/

Liens utiles

Site

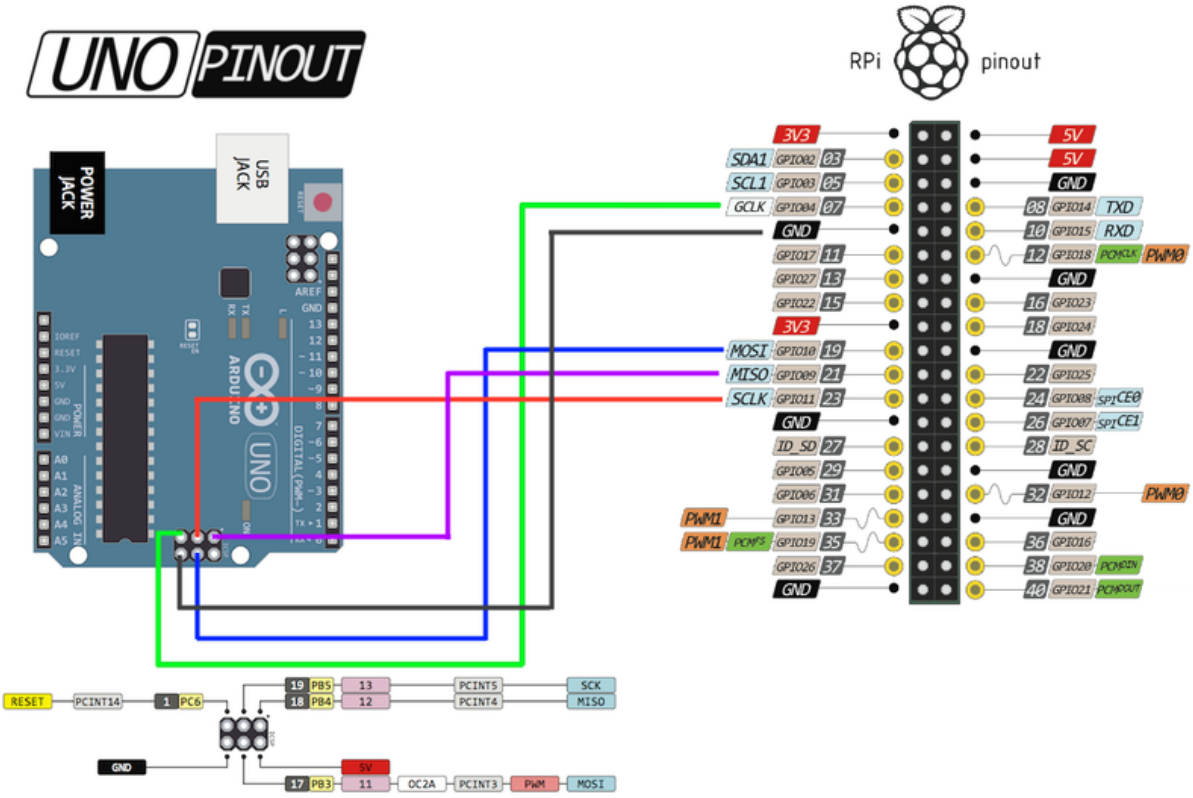
- http://projet-p10.plil.fr/IMA3_P10/site/accueil.php

Dépôt Git

- https://archives.plil.fr/grouille/IMA3_P10.git

Annexes

Schéma de connexion SPI Arduino UNO - Raspberry PI



Schémas de connexion des capteurs