CPE Lyon - 4IRC - S7 - Année 2020/21

Architecture et Protocoles Réseaux pour l'IoT

Projet - Mise en place d'une mini-architecture IoT

Oscar Carrillo - Anthony Chomienne

Ce projet a pour objectif d'utiliser les différentes compétences que vous avez acquises pendant les TPs pour vous permettre de mettre en place une architecture de l'internet des objets, telles que :

1. Développer un protocole réseau pour établir une communication entre un objet et une passerelle.
2. Programmer un micro-contrôleur afin que ce dernier puisse collecter des données et les envoyer sous un format précis, ainsi que de recevoir des données de la passerelle afin de les afficher sur un écran OLED.
3. Programmer une passerelle afin qu'elle puisse recevoir les données d'un client, les manipuler, les stocker et les envoyer au client suivant en format précis.
4. Développer une application Android capable de se connecter à un serveur pour demander des données et les afficher, ainsi que renvoyer des configurations à un objet connecté.

Organisation

— Travail virtuel dans la plateforme *Teams*, le code pour joindre l'équipe du module est `d5sudcc`.
— Ce projet est à réaliser par équipes de 4 ou 5 élèves, veuillez inscrire votre groupe de travail dans la fiche partagé du module : `https://bit.ly/4irc-iot20`
— Afin de mieux répondre à vos questions, créez un canal sur Teams pour votre groupe de travail avec le numéro d'équipe pris dans le point précédent.

Introduction

Votre entreprise souhaite déployer des objets connectés avec des capteurs de température afin de pouvoir relever et afficher des informations dans chaque bureau possédant un de ces objets. De plus, l'affichage des données pourra être défini via une application Android communiquant directement avec le serveur. Cependant, les objets qui collectent les informations, et le serveur qui stockent et manipule les données ne sont pas configurés. C'est donc à vous de réaliser l'architecture objet-passerelle-serveur entre votre objet et le serveur qui vous sont fournis. De même, aucune application mobile pour l'affichage et configuration des objets n'existe actuellement, et vous devrez donc la créer pour réaliser les besoins de votre entreprise.

La figure 1 montre un schéma de l'architecture que vous allez mettre en place pour la configuration de votre architecture. Ici un smartphone Android en simulateur ou réel qui sera connecté par le réseau (WiFi dans le cas d'un smartphone physique ou interne de l'ordinateur si smartphone simulé) à une passerelle.

Votre ordinateur couplé avec une carte micro :bit par USB, fera la fonction de passerelle pour envoyer et recevoir des données via RF 2.4GHz et communiquer aussi avec votre smartphone par le réseau.

Pour finir vous aurez à programmer un des objets à déployer dans les bureaux de la société ; il est composé d'un micro-contrôleur micro :bit qui va obtenir des données de ses capteurs et afficher des informations sur un écran OLED (acteur).

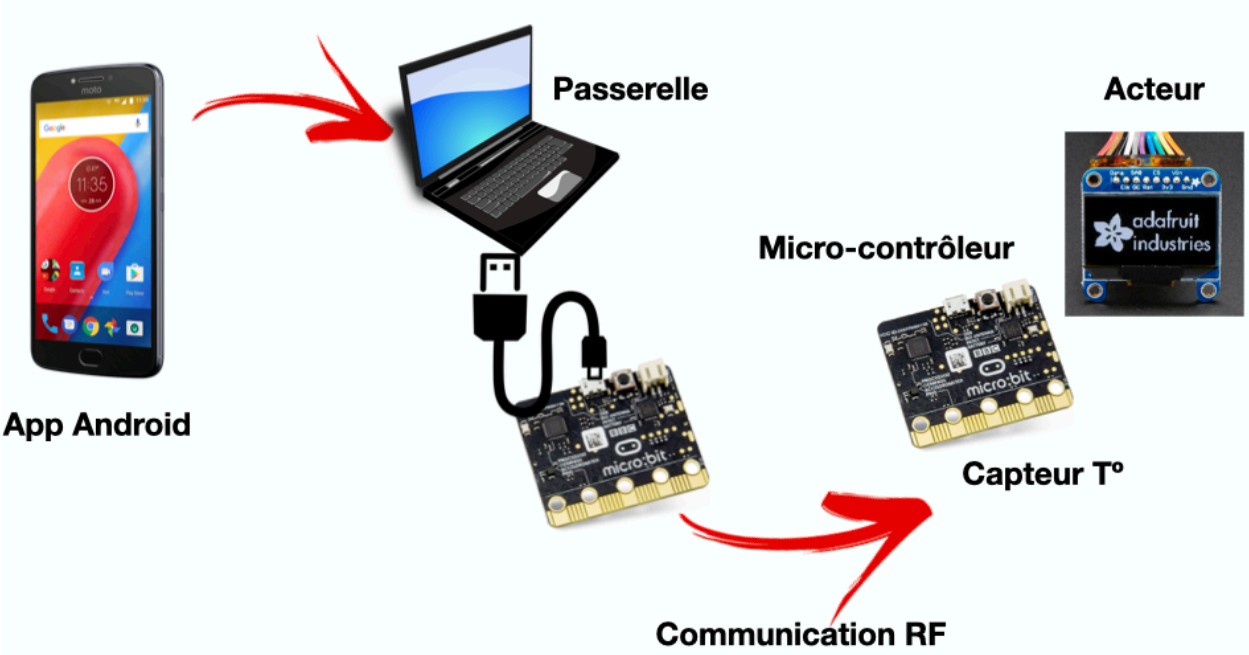

FIGURE 1 – Maquette de l'architecture du mini-projet

Mise en place d'une infrastructure objet-passerelle

Exercice 1 : Mise en place du réseau

Cette étape de ce projet consiste à mettre en place un canal de communication entre votre objet et votre passerelle afin que ces deux puissent communiquer et échanger des messages entre eux. Les communications sont bidirectionnelles, c'est à dire, que l'objet comme la passerelle sont en mesure de recevoir et d'envoyer des messages de l'un à l'autre.

Vous devez définir un protocole simple pour envoyer les messages entre les deux modules micro :bit, *pensez aussi à la sécurité des données envoyées.*

Exercice 2 : configuration des capteurs

Une étape importante de la mise en place de l'infrastructure côté objet consiste à configurer l'objet à déployer dans les bureaux, afin que ce dernier puisse collecter les informations souhaitées : température et luminosité.

Les informations doivent être envoyées au serveur (passerelle) et enfin afficher les informations dans l'ordre demandé par le serveur.

Pour ce faire, vous disposez d'un micro-contrôleur micro :bit qui compte plusieurs capteurs intégrés. De plus, l'objet dispose d'un afficheur OLED sur lequel les données pourront être affichées.

Affichage sur l'écran OLED

Une partie de la configuration de l'objet est l'affichage des données sur l'écran OLED qui accompagne le module micro :bit. Avant d'afficher directement les informations dans l'ordre défini par le serveur, assurez vous de comprendre comment afficher des informations sur cet écran. Ainsi, pour commencer, vous pouvez programmer votre module micro :bit afin que ce dernier affiche les données récupérées des capteurs.

Si vous programmez le micro :bit en micro-python, vous disposez du code contrôleur d'un écran OLED Adafruit sur le git : `https://github.com/CPELyon/microbit_ssd1306`. Un exemple d'utilisation de la bibliothèque est dans le sous-dossier `/samples/hello-world/`.

Dans le code donné, on utilise l'interface I^2C pour pouvoir envoyer les informations à afficher, on positionne les données en indiquant la ligne et colonnes à utiliser, la distribution des pins dans la carte est présenté dans la figure 2.

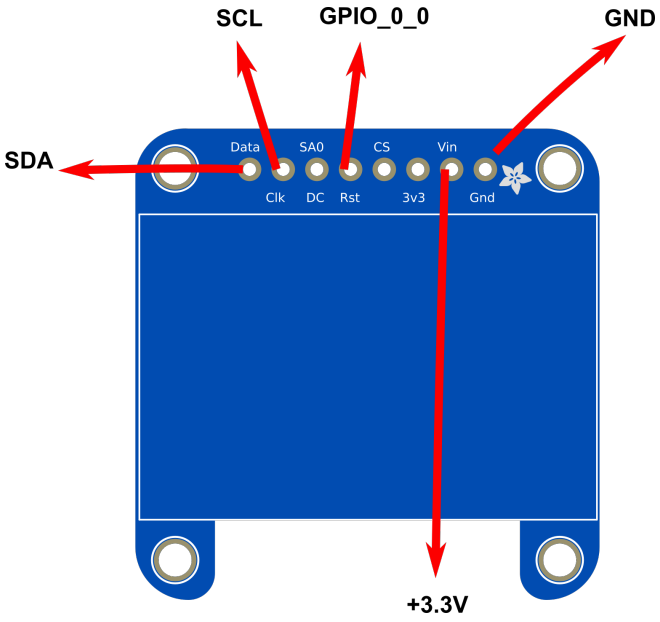

FIGURE 2 – Disposition des pins du module *OLED*

Communications avec la passerelle

Cette étape englobe deux rôles. En effet, il faut que votre objet soit capable d'envoyer les informations récoltées à la passerelle, mais qu'il soit également capable d'écouter les messages envoyés par la même afin de pouvoir afficher le contenu des messages sur l'écran OLED.

Le format des messages à envoyer à la passerelle est libre, ainsi les données peuvent être envoyées brutes (sans traitement côté objet), ou bien suivant un format précis (type ficher de configuration format JSON par exemple). Dans un premier temps, n'utilisez aucun format et envoyez les données brutes au serveur, le choix d'un format est optionnel.

Le micro :bit attaché au PC fera le rôle de récepteur RF 2.4GHz de la passerelle et recevra les données de la part des capteurs déployés, mais sera aussi en mesure d'envoyer la configuration d'affichage à chaque objet. Les données de configuration seront reçus à travers son interface USB (UART), indiquant l'ordre d'affichage des données par 2 lettres majuscules, ainsi :

— **TL** : Indique que la **T**empérature sera affiché en premier, puis la **L**uminosité

— **LT** : Indique que la **L**uminosité sera affiché en premier, puis la **T**empérature

Exercice 3 : configuration de la passerelle et serveur

Le PC, dans un premier temps, doit être configuré pour stocker les données reçues au format brut dans un fichier texte, de cette façon il fera aussi le rôle de serveur.

Puis ce dernier va écouter les requêtes du client Android par UDP :

— Message "`getValues()`" : le serveur répond à ce message en envoyant les données reçu dans sa passerelle au client Android sous le même format envoyé depuis l'objet (texte, JSON,...).

Exemple de code serveur

Le code d'un serveur effectuant les tâches demandées au point précédent est disponible sur :

https://github.com/CPELyon/4irc-aiot-mini-projet/blob/master/controller.py

Dans ce code python à exécuter dans un PC linux et à adapter pour d'autres systèmes d'exploitation, un serveur est démarré sur le port 10000 pour écouter les requêtes UDP selon le protocole décrit précédemment.

L'écriture et lecture dans la sortie UART est aussi géré dans ce code, faites attention au port qui est associé au micro :bit pour le faire correspondre à celui défini dans le code.

Évolutions du serveur

L'objectif de cet exercice est de faire évoluer le serveur, plusieurs tâches sont possibles :

— Remplacer le fichier texte par une vrai structure de base de données telle que MongoDB, InfluxDB ou SQLite, afin d'avoir une meilleure gestion des données côté serveur, pour une gestion plus facile, n'hésitez pas à regarder des solutions en conteneurs.
— Définir un format spécifique pour l'échange des données avec le client Android et le micro-contrôleur, comme par exemple un fichier de configuration au format JSON. Cependant, le choix du format reste libre
— Définir une interface web pour la visualisation des données reçus, par exemple Grafana
— Donner la possibilité de gérer plusieurs objets. Dans l'état actuel le serveur reçoit des données brutes sans faire un filtrage et de ce fait, il n'est pas possible de gérer plus d'un objet, vous pouvez donc implémenter un protocole pour pouvoir gérer plusieurs objets au même temps et les gérer depuis l'interface web et/ou l'application Android.
— ...

Création de l'application Android

La dernière partie de la mise en place de l'architecture IoT consiste à développer une application Android qui permet de contrôler l'ordre d'affichage des données collectées sur un des objets en particulier. Ainsi, l'application a deux fonctionnalités, le choix de l'affichage des données d'un des objets et le choix du serveur sur lequel cela s'applique.

Pour simplifier, on assume qu'il y a seulement un objet associé à chaque serveur et donc on doit choisir seulement l'adresse du serveur de destination et pas un des objets qui lui sont associés.

Pour faciliter le développement de votre application, les données seront envoyées via le protocole UDP, et votre Smartphone sera connecté au serveur via WiFi si vous êtes en physique ou via le réseau interne de votre PC si vous êtes en simulateur.

Exercice 1 : choix d'affichage

Dans un premier temps votre application devra être en mesure de définir un ordre d'affichage pour les 2 différentes données collectées. Vous pouvez, pour ceci, afficher les trois données à l'écran et par simple pression du doigt définir l'ordre d'affichage des données. *Ceci n'est qu'un exemple, et le choix d'interface visuel de votre application reste libre.*

Exercice 2 : définir le serveur de destination

En plus de choisir l'ordre d'affichage des données, votre application doit être en mesure de choisir le serveur de destination. En effet, le but est de pouvoir contrôler l'affichage des données pour chaque objet via le serveur avec lequel il communique. Ainsi, le choix du serveur dans l'application est indispensable. La configuration du serveur dans l'application Android se fait via l'adresse IP du serveur et son port d'écoute (par défaut : 10000). Donc dans votre application Android, vous devez disposer de deux champs dans lequel il sera possible de saisir une adresse IP et un port, qui permettront la communication avec le serveur souhaité.

De plus, la communication sera effectuée via le protocole UDP, et aucun ACK n'est demandé. Ainsi, votre application doit seulement faire de l'émission en direction du serveur, sans se préoccuper de devoir réceptionner des paquets. Les données envoyées par votre application seront les 2 lettres majuscules indiquant l'ordre d'affichage dans l'écran OLED.

Exercice 3 : Communication bidirectionnelle avec le Smartphone

Dans l'objectif d'être capable d'afficher les données également sur votre Smartphone, votre application doit être en mesure de recevoir des messages venant du serveur. Tout comme le serveur envoie des données à l'objet, ce dernier devra pouvoir envoyer les mêmes données avec le même format à votre Smartphone. Ce dernier devra être capable de réceptionner les données et de les afficher dans l'application créée précédemment.

De plus, le Smartphone ne doit réceptionner que les données émises depuis le serveur avec lequel il est connecté (défini dans l'exercice précédent).

Rendu final et démo

L'évaluation du mini-projet sera faite par une présentation de vos choix technologiques et une démonstration de votre implémentation d'une durée de 10 minutes maximum le 7 décembre après-midi.

De même, un rendu sur la plateforme e-campus devra être fait au plus tard le 4 décembre à 18h. Le rendu doit contenir :

1. Un rapport très synthétique listant les membres de l'équipe et décrivant les activités réalisées (au moins 2 pages)

2. L'application Android *documenté* créée pour le contrôle et accès à votre architecture de l'internet des objets
3. Votre projet ou code *documenté* créé pour gérer la capture et affichage des données dans l'objet
4. Votre projet ou code *documenté* créé pour le micro-contrôleur faisant le rôle de passerelle
5. L'application coté serveur *documenté*