RASPIOT

Projet 1A

Electronique et IoT

Contrôle de micro électroniques à distance, au travers d'une interface web
et de boutons physiques

SOUBIRAN Estéban

MICHOUX Théo

MAUGER Pierre

PITHON Gabriel

Introduction

Les origines de l'idée

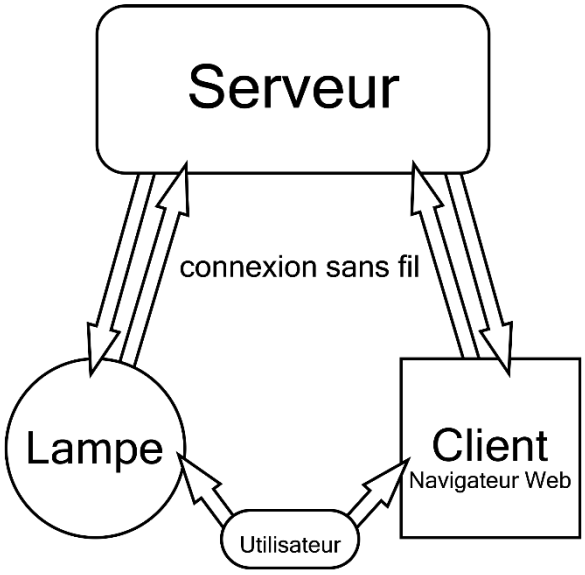
Après la présentation de Pépite[1] en décembre 2018, il est né une volonté commune de créer, ou tout au moins s'intéresser de très près au monde de l'entreprise. Nous nous sommes donc réunis tous les 4 et avons trouvé une idée qui nous semble viable. Nous sommes passés devant le jury Pépite et sommes ensuite devenus « étudiants-entrepreneurs ».

Pour trouver une idée, nous nous sommes réunis à de multiple reprises, avons passé des soirées entières à réfléchir. Puis, après avoir trouvé une idée qui nous semblait intéressante, il a fallu vérifier sa viabilité. Pour cela, nous avons réfléchi aux différents moyens qu'il allait falloir mettre en œuvre pour parvenir à réaliser ce projet. Nous nous sommes donc encore réunis à de multiple reprises pour savoir exactement où nous allions.

Le projet: objectif et volontés

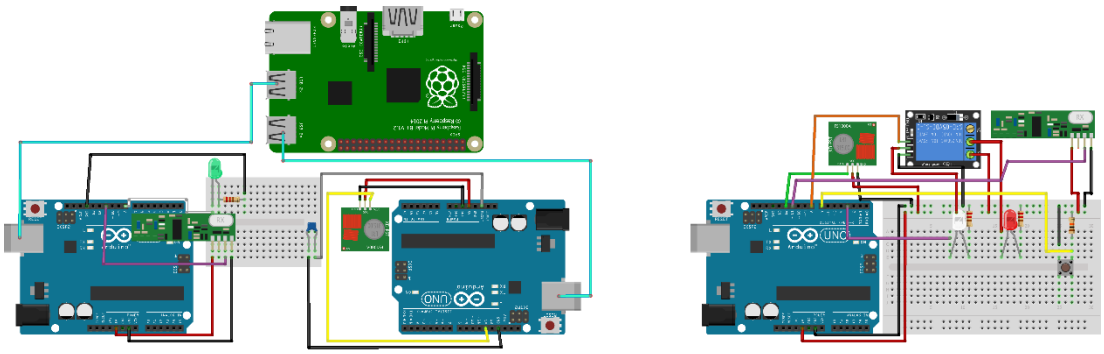
Objectif

Réussir à allumer une lampe, à distance, via une interface web (voir schéma ci-dessous)



1 Fonctionnement Globale du Système

[1] Réseau des étudiants-entrepreneurs



2 Schéma global du projet avec à gauche le serveur et à droite la lampe

Cahier des charges

L'utilisateur peut changer l'état d'une lampe via son navigateur web.

L'utilisateur peut changer l'état d'une lampe via un interrupteur classique.

L'utilisateur doit se voir informé du changement d'état de la lampe, lorsqu'un autre utilisateur interagit avec le système.

L'utilisateur doit être en mesure de faire des entrées et sorties de la base de données[2] via l'interface web pour ajouter ou supprimer des lampes à contrôlables.

Le système doit être performant, et ne pas allumer la lampe en 10 secondes, mais le plus rapidement possible à partir de l'émission du signal utilisateur

Le système doit être stable et donc planter le moins possible

Problématique

Comment gérer efficacement l'éclairage, et de manière plus général l'électronique, à l'échelle d'une maison comme d'un entrepôt ?

[2] Une base de données (en anglais database), permet de stocker et de retrouver l'intégralité de données brutes ou d'informations en rapport avec un thème ou une activité

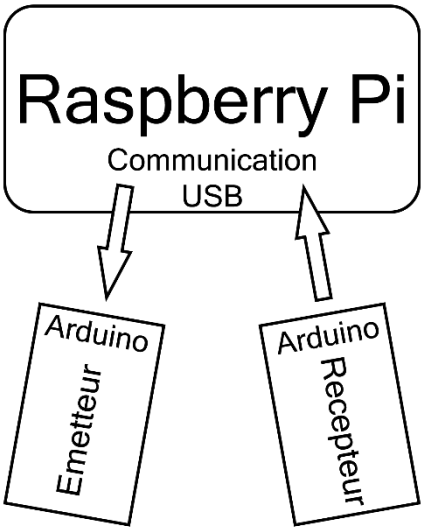
Développement

Web serveur

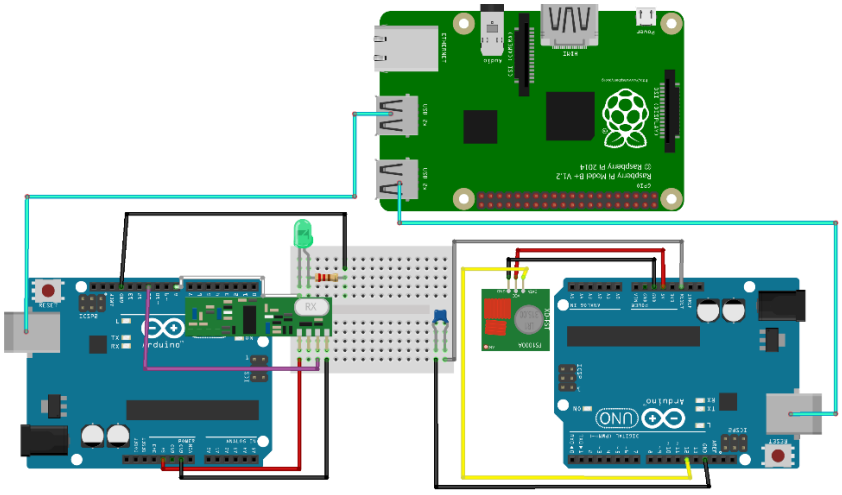
Objectifs

Le but du web serveur[3] est double. Il permet à la fois d'avoir une interface graphique pour interagir avec le système, mais permet aussi la manipulation de données pour l'ensemble du système. C'est-à-dire que c'est par lui que l'utilisateur va passer pour contrôler les différents appareils qui s'y trouvent, grâce à une interface graphique. Mais c'est aussi l'endroit où sont envoyées et traitées l'ensemble des données.

Il se compose de 3 éléments , un Raspberry Pi : le serveur, et 2 Arduino : interface entre le Raspberry Pi et différents objets connectés.



3 Schéma de Fonctionnement du serveur



4 Schéma de la partie serveur du projet

De la gauche vers la droite : Arduino (bleu), récepteur RF (vert en longueur), Raspberry Pi (vert avec la framboise), Emetteur RF (vert et orange), Arduino (bleu)

[3] Un serveur web est spécifiquement un serveur multi-service utilisé pour publier des sites web sur Internet ou un intranet, réseau interne.

Cas d'usages

Exemple 1: Connexion d'un client

Le serveur est branché, allumé et prêt à recevoir des requêtes HTML[4][5]. Un client se connecte alors au serveur via le navigateur web de son choix. Il envoie donc une requête au serveur qui lui renvoie, selon l'URL[6] la page correspondante. S'il se rend sur XXX.XXX.X.XX:3000/, le serveur lui renvoie alors la page d'accueil, s'il se rend sur une page qui n'existe pas, la page affichée sera une page d'erreur, 404 dans ce cas.

Exemple 2: Ajout et suppression d'une lampe

Pour ces 2 éléments, la méthode est la même. Il s'agit d'un formulaire qui utilise la méthode POST[7] pour envoyer les données au serveur. C'est à dire qu'une fois que l'utilisateur a rempli les différents champs du formulaire, il clique sur un bouton qui a pour conséquence d'envoyer les données. Cependant, celles-ci sont brutes et doivent être traitées avant d'être utilisées. En effet, malgré les contraintes mise en place dans l'HTML côté client, il est très facile de les surpasser et quelqu'un de mal intentionné peut envoyer n'importe quoi au serveur. C'est pour cela qu'un système de nettoyage, vérification (longueur de mots, correspondance) est mis en place dans le serveur. Si une erreur est détectée, alors rien ne se passe et l'utilisateur est averti. Sinon, en fonction du formulaire, la base de données est modifiée en conséquence et les autres clients qui seraient connectés en même temps voient la page de leur navigateur se rafraîchir pour correspondre aux nouvelles données, si cela n'est pas fait et que l'utilisateur essaie d'interagir avec une lampe qui n'existe pas, le serveur plante.

Exemple 3: Interaction avec une lampe

Lorsque sur la page d'accueil, là où se trouve l'ensemble des lampes, l'utilisateur interagit avec l'une d'elles, il se passe beaucoup de choses dans le serveur. D'abord, la base de données est mise à jour en conséquence du changement d'état. Ensuite, via les web sockets[8], une notification est envoyée en temps réel pour mettre à jour l'ensemble des autres pages web des clients connectés, sans devoir rafraîchir la page. Cela permet d'avoir l'ensemble des pages avec les mêmes données. Ensuite, des données sont envoyées via l'USB à l'Arduino pour agir sur les objets connectés via une transmission radio, mais cela est plus détaillé dans Python[9] et Arduino.

Conception

Technologies utilisées

Hardware

- 1 Raspberry Pi
- 2 Arduino

[4] L'HyperText Markup Language, généralement abrégé HTML, est le langage de balisage conçu pour représenter les pages web.

[5] Une requête HTTP est un ensemble de lignes envoyé au serveur par le navigateur.

[6] Le sigle URL (formé des initiales de l'anglais : Uniform Resource Locator, littéralement « localisateur uniforme de ressource »), désigne le nommage uniforme d'une ressource localisée, c'est-à-dire « une ressource identifiée par son emplacement ».

[7] Cette méthode est utilisée pour transmettre des données en vue d'un traitement à une ressource, le web serveur, (le plus souvent depuis un formulaire HTML).

[8] Web Socket est un standard du Web désignant un protocole réseau visant à créer des canaux de communication bidirectionnelle, client / serveur.

[9] Python est un langage de programmation interprété, multiparadigme et multiplateformes.

Software

NodeJS[10]

Dépendances majeurs:

- Express
- Socketio
- Chokidar
- Python-shell
- Pug
- Express-validator

Et plus de 400 autres dépendances mineures

Fonctionnement

Hardware

Le Raspberry Pi est un micro-ordinateur, comme votre PC, fonctionnant avec un système d'exploitation de PC, et non mobile. Sur architecture ARM[11], il est peu puissant mais largement suffisamment pour faire office de serveur et surtout consomme bien moins qu'un véritable serveur dédié. Son coût est également très intéressant pour ce type de projet. C'était donc un indispensable à notre projet.

L'Arduino, carte électronique programmable à faible coût, fait aussi partie des indispensables pour ce genre de projet. Il permet de contrôler différents composants assez simplement et de manière très efficace.

Software

NodeJS est un environnement javascript[12] open-source[13], cross-platform[14] en temps réel exécuté en dehors du navigateur, côté serveur. Pourquoi avoir choisi NodeJS pour réaliser le serveur plutôt que PHP[15] ou Python ? Pour plusieurs raisons. Tout d'abord, après avoir appris le javascript côté client, le fait de continuer sur NodeJS permet de ne pas avoir à éviter à réapprendre un nouveau langage. Ensuite, en ce qui concerne Python, nous ne savions pas comment il fonctionnait et surtout des capacités qu'il avait. L'utilisation de NodeJS est plus simple que celle de PHP. En effet, NodeJS crée son propre serveur alors que PHP a besoin d'un serveur pour fonctionner, Apache par exemple. En outre, le système d'exploitation du Raspberry Pi (appelé Raspbian) est assez particulier et donc les logiciels ne sont souvent pas optimisés pour fonctionner correctement. Et c'est le cas de PHP qui a énormément de mal à fonctionner sur Raspbian, à l'inverse de NodeJS qui est par définition cross-platform donc parfaitement compatible.

Les dépendances de NodeJS:

[10] Node.js est une plateforme logicielle libre et événementielle en JavaScript orientée vers les applications réseau.

[11] ARM est un type d'architecture de processeurs

[12] JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation (par exemple) de Node.js.

[13] La désignation open source signifie libre redistribution, d'accès au code source et de création de travaux dérivés.

[14] Un logiciel multiplateforme est un logiciel conçu pour fonctionner sur plusieurs plateformes, c'est-à-dire le couple liant ordinateur et système d'exploitation.

[15] HyperText Preprocessor, plus connu sous son sigle PHP , est un langage de programmation libre, principalement utilisé pour produire des pages Web.

- Express : permet de créer un serveur et de gérer ses composants efficacement
- Socketio : permet l'utilisation des web sockets, ou tout au moins une communication à double sens en temps réel
- Chokidar : permet de surveiller la modification de fichiers
- Python-shell : permet d'exécuter des scripts python via NodeJS
- Pug : moteur de Template [16]pour générer les pages HTML
- Express-validator : permet le traitement de données envoyée l'utilisateur

Problèmes rencontrés

Tout d'abord, le projet RaspIoT[17] ne provient pas d'internet. Il n'existe aucun tutoriel vous permettant de refaire ce qui a été fait, du début à la fin. Par conséquent, et "heureusement", sinon le projet perd de son sens, tout a été écrit et réfléchi par l'équipe du projet.

Cela implique une autonomie énorme dans le travail puisqu'il n'existe aucun support sur lequel se reposer. Mais c'est en réalité plus complexe.

En effet, chaque dépendance a sa documentation, permettant de comprendre et connaître les fonctions qui la compose. Cela permet de mettre en place facilement les fonctions voulues à la documentation existante. En revanche, dès lors que l'on commence à mixer différentes fonctions de différentes dépendances ensembles, les problèmes apparaissent. Tout simplement parce qu'il n'existe pas de tutoriels, on se retrouve à créer, imaginer et mettre au point un nouveau système fonctionnel. Lorsque le nombre de dépendances est faible, il est plus facile de rédiger le programme alors que pour un nombre élevé (ici 8) la phase de rédaction du projet est extrêmement compliquée. On se retrouve seul, face à un problème. Heureusement, on peut quand même compter sur StackOverflow[18].

Ressenti face à ce travail

Nous avons eu le sentiment de nous trouver face à un obstacle certes franchissable mais extrêmement complexe. Même si nous nous y connaissions un minimum, et si nous savions que c'était possible, dans les grandes lignes, c'est quelque chose que nous n'avions jamais fait et nous n'avions qu'une vague idée de comment y arriver. Le « comment » précis était très flou. Tellement que même si l'objectif final n'a pas changé, les moyens pour y arriver ont changé par quatre fois. Chaque ajout de lignes de codes provoque dans la majorité des cas une erreur, un problème qu'il faut corriger pour que le système fonctionne, mais c'est le jeu, un jeu fort agréable qui a été mené jusqu'au bout.

Arduino

Objectifs

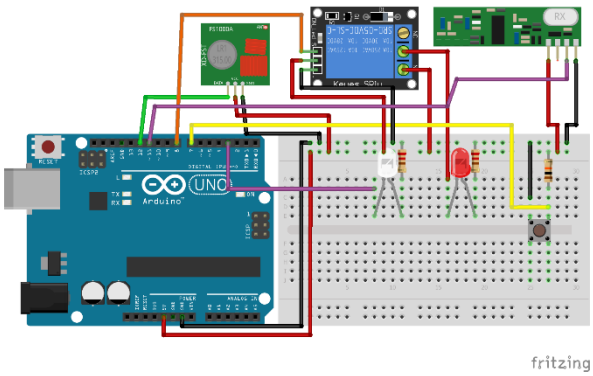
Transmettre un changement d'état de la lampe du serveur à la lampe, mais aussi de la lampe au serveur, via l'utilisation de modules RF[19].

[16] Un moteur de Template est un logiciel qui produit des documents en combinant des modèles avec une base de données

[17] Nom de notre projet, vient de la contraction de RasPi, signifiant Raspberry Pi, et IoT, signifiant Internet of Things

[18] Stack Overflow est un site web proposant des questions et réponses sur un large choix de thèmes concernant la programmation informatique.

[19] Radio Fréquences



5 Partie client du projet, c'est ici que les lampes pilotées vont s'allumer

De la gauche vers la droite: Arduino (bleu), Module Emetteur (vert et orange), Relais (bleu), Module Récepteur (vert en longueur)

Cas d'usages

Exemple 1: Du Web au Réel

Lorsqu'un client change l'état de la lampe via la web app, alors ce changement doit être envoyé à la lampe. Pour cela l'information passe par un script python, puis est envoyé à travers le port série et est reçu par l'Arduino connecté au serveur. Le programme est assez simple, lorsqu'il reçoit un message, il l'envoie grâce à un module RF. L'Arduino sur la lampe le capte et en fonction du message, change l'état de la lampe.

Exemple 2: Du Réel au Web

Lorsqu'un client change l'état de la lampe via l'interrupteur physique, alors cette dernière change d'état. Mais un message, en RF est aussi envoyé à un second Arduino connecté au serveur. Lorsque le second arduino, connecté au serveur reçoit un message, il le vérifie puis l'envoie via le port série à un script python qui s'occupe de le traiter.

Conception

Technologies utilisées

Hardware

3 Arduino

Software

C[20]/C++[21]

Virtual Wire

Fonctionnement

Hardware

Utilise les ports séries[22] pour communiquer avec le serveur et des modules RF (radio fréquences) pour communiquer avec les autres Arduino.

Software

Rien de particulier, si ce n'est la librairie Virtual Wire qui permet de gérer efficacement les modules RF.

---

[20] C est un langage de programmation impératif généraliste, de bas niveau

[21] C++ est un langage de programmation compilé permettant la programmation sous de multiples paradigmes (comme la programmation procédurale, orientée objet ou générique).

[22] Synonyme à port USB dans ce cas

Problèmes rencontrés

Les modules RF furent la partie la plus complexe de cette partie. Comment les faire fonctionner ? Les librairies sont très rares mais heureusement, pas les tutoriels. Après avoir lu entièrement la documentation et visionné de nombreux tutoriels, nous avons pu commencer.

Nous avons subi énormément d'échecs, que ce soit le fonctionnement même des émetteurs/récepteurs ou bien des erreurs dans l'écriture des programmes. Nous avons obtenu une première version fonctionnel mais le résultat ne n'était pas satisfaisant :e temps d'émission était trop long et l'envoi ne permettait que trop peu de possibilités. Nous avons donc tout recommencé, recherche, analyse, création. Puis nous sommes arrivés à une seconde version fonctionnelle.

Ressenti face à ce travail

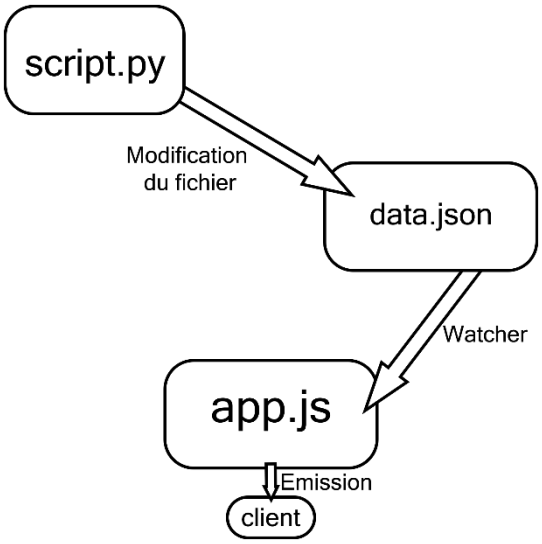
Le plus compliqué fut de faire fonctionner les modules RF. En effet, cela n'est vraiment pas aisé et nous avons dû lire et essayer des dizaines de tutoriels Cela fût très long mais très intéressant de voir l'évolution du système.

Python

Objectifs

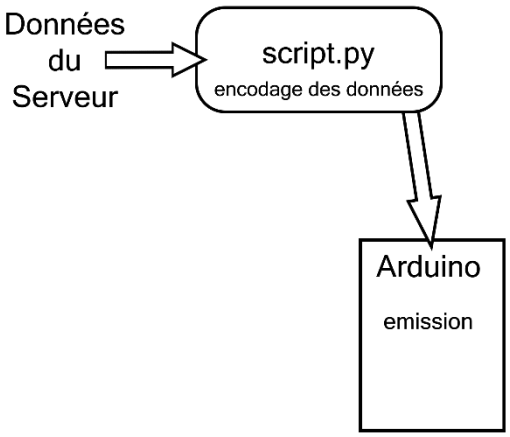
Le but des 2 scripts Python est de servir de passerelles entre le serveur écrit en NodeJS et fonctionnant sur un Raspberry Pi et les Arduino qui permettent d'utiliser les émetteur/récepteurs RF. Ils sont indispensables car le python facilite la communication via le port USB. Cela est possible sans le python mais devient extrêmement complexe.

L'image ci-dessous montre le fonctionnement du script récepteur de données. Après que l'Arduino a transmis ses données, ces dernières sont reçues par un script python qui va modifier la base de données. Ensuite, une fonction dans le serveur permet la détection de modifications dans la base de données. Cela permet d'envoyer des informations au client pour mettre à jour l'interface web.



6 Schéma de fonctionnement du script python récepteur

Cette seconde image montre l'envoi de données du serveur à l'Arduino par le biais d'un second script python. Ce dernier permet l'encodage des données et l'envoi de ces dernières. Il est indispensable puisqu'il sert de passerelle entre le serveur et l'émetteur.



7 Schéma de fonctionnement du script python émetteur

Cas d'usages

Exemple 1: Du Pi à Arduino

Lorsqu'un client se trouve sur la page de gestion des lampes, il peut interagir avec, sur la page et dans le réel, via le serveur. En effet, comme nous avons pu voir dans le cahier de conception Web Serveur, il y a tout un mécanisme permettant de stocker et de transmettre aux autres clients ce que le client vient de faire. Mais il se passe aussi autre chose. Lors d'un changement d'état d'une lampe, le web serveur exécute un script python en lui passant des paramètres, comme le nom de la lampe et son état. Le script est "très simple", seulement quelques lignes qui permettent de passer des données du serveur à l'Arduino au travers d'un port USB. L'Arduino reçoit les données et les traite en fonction.

Exemple 2: De l'Arduino au Pi

Lorsqu'un client appuie sur l'interrupteur physique, alors un message, en radiofréquence, est envoyé à l'Arduino récepteur côté serveur. Un second script python intervient juste après. Il permet de faire le lien entre l'information que transmet l'Arduino récepteur au serveur et la base de données. Le script reçoit et interprète les données pour les stocker dans la base de données. Cela tant que le serveur n'est pas éteint.

Conception

Technologies utilisées

Hardware

Raspberry Pi

Software

Python

Dépendance :

- Python-serial

Fonctionnement

Hardware

Le Raspberry Pi sert de support pour l'exécution du script et l'envoi de données en USB. Il permet aussi de relier le serveur à l'Arduino.

Software

En termes de compléments, seul le module python-serial est utilisé. C'est lui qui permet de faire simplement le transfert de données entre Raspberry Pi et Arduino.

Problèmes rencontrés

La communication directe entre différents langages de programmation est très complexe. Il y a principalement deux paramètres qui varient : le typage de la données, c'est-à-dire si c'est un nombre, un caractère, une chaine de charactères…, et l'encodage, c'est-à-dire comment est traduite l'information de notre langage à celui de la machine.

Du serveur à l'Arduino:

Comment fonctionne le lecteur du port série de l'IDE[23] Arduino ? Voilà une question bien complexe. Pour le comprendre, nous avons dû effectuer beaucoup de tests, d'autant que nous n'avons trouvé aucune documentation sur ce sujet.

De plus, la documentation des fonctions pour récupérer des données depuis le port série ne sont que très peu détaillées. Mais après de multiple tests, nous sommes parvenus à déterminer le type et l'encodage des données reçues.

Cependant, ce n'est pas le seul problème auquel nous avons dû faire face. En effet, lors de l'envoie de données, l'Arduino est sujet à un reset, c'est-à-dire que le programme recommence. Il nous a donc fallu comprendre d'où venait ce problème avant de réussir à le résoudre. Heureusement, il existe une solution simple et expliquée à ce problème.

De l'Arduino au serveur:

Pour transmettre une donnée de l'Arduino au Serveur, nous avons eu beaucoup de mal. En effet, l'encodage utilisé entre l'Arduino et le script python ne sont pas les mêmes. Par conséquent, il faut faire traduire à notre script python les données, mais cela fait alors apparaitre, sous forme de caractères, les retours ou les fins de lignes. Par conséquent, un traitement important de la donné doit être mis en place.

L'écriture de ce script, qui ne fait pourtant que quelques lignes, fût longue et périlleuse parce qu'il s'agit d'information que l'on ne trouve nul part. Nous avons dû effectuer beaucoup de tests pour comprendre le fonctionnement et s'apercevoir de ces différences.

Ressenti face à ce travail

Il semblait simple, mais s'est avéré complexe. En effet, nous ne nous attendions pas à tous ces problèmes de conversion de types et d'encodage. Il nous a donc fallu chercher longtemps pour d'abord bien comprendre le fonctionnement de ces différents langages et la façon dont il gère les données pour par la suite réussir à adapter notre programme. Cela fut donc très instructif.

[23] En programmation informatique, un environnement de développement (abrégé EDI en français ou IDE en anglais, pour integrated development environment) est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui développent des logiciels.

Raisons du choix de cette solution

Nous avons réfléchi et essayé plusieurs solutions. Celle qui a été mise en place est celle qui nous a semblé la plus pertinente mais aussi la plus efficace.

Pour chacune des idées que nous avons essayées, nous avons à chaque fois gardé le meilleur et changé ce qui n'allait pas. Cela permet d'avancer sans en permanence tout recommencer de zéro. Toutefois, cela a quand même était indispensable, notamment quand nous sommes passé d'un serveur Arduino à un serveur sur Raspberry Pi. Cependant, cela nous a permis de découvrir les possibilités offertes par l'Arduino et nous savons maintenant que ce n'est pas la bonne voie.

Ainsi, il est facile de penser que chaque piste de réflexion est inutile puisque cela nous fait perdre du temps. Mais il n'en est rien : c'est grâce à nos erreurs que nous avons pu savoir vers quoi nous orienter et savoir comment procéder.

Explications

Au départ, nous avons cherché comment arriver au résultat recherché, mais sans parler de moyens techniques pour y arriver, simplement d'un point de vu global, c'est la conception idéalisée initiale.

Nous avons ensuite fait des recherches sur internet pour voir ce qui existait. Nous avons vu qu'il était possible d'héberger un serveur sur un Arduino. Nous nous sommes rendus compte que les possibilités étaient extrêmement limitées, notamment pour la partie serveur.

Nous nous sommes donc tournés vers un autre moyen pour héberger le serveur mais nous gardons tout de même l'idée de la gestion des composants grâce à un Arduino, qui nous semblait efficace. Nous avons donc découvert que le Raspberry Pi pourrait faire office de serveur et que les possibilités offertes étaient très intéressantes. En effet, ce dernier supporte un bon nombre le langage et donc un choix intéressant pour le langage du serveur.

Méthodologie

Nous avions évidemment fixé notre objectif final mais pour y arriver, nous nous sommes fixés de nombreux objectifs intermédiaires, plus simple qui permettent de répartir le travail et surtout de voir le projet avancer sans trop s'éparpiller.

Pour arriver à ces divers objectifs, nous fonctionnons à chaque fois en quatre étapes.

Partons du principe que nous avons clairement défini l'objectif.

La première étape est celle des essais de découvertes. C'est à dire que nous testons, non pas dans le projet mais à part, sans lien direct avec le projet, la solution que nous souhaitons mettre en place. Cela permet de voir les capacités et le fonctionnement de ce que nous allons utiliser. Si cela ne nous paraît pas conforme à nos attentes, alors on recommence et on essaie autre chose. En revanche, s'il semble que cela nous convient alors on passe à l'étape suivante.

La seconde étape est de réfléchir sur papier comment intégrer un nouveau système dans le projet déjà créé. Une cela fait et que donc nous savons exactement ce que nous souhaitons faire, nous pouvons passer à l'étape suivante.

Cette troisième étape est celle des essais d'approfondissement, c’est à dire que nous créons notre système pour qu’il soit fonctionnel et utilisable avec le projet déjà fait mais en ne l’implantant pas encore au serveur. Cela permet notamment de se rendre compte si des erreurs de réflexions ont été faites et de les corriger le plus facilement possible.

Enfin, la quatrième étape est celle de l’implémentation et des derniers ajustements pour Assurer un fonctionnement optimal du système. Des étapes longues mais indispensable pour prendre en compte le plus de cas possible et donc s’assurer de la viabilité et de l'interopérabilité du système.

Conclusion

Apports

Ce projet nous a apporté de l'expérience pour gérer un travail en autonomie. Nous avons acquis les bases du travail en équipe.

En effet, après avoir décidé ce que nous allions faire et comment, nous nous sommes répartis sur chacune des parties du projet pour avancer en parallèle et donc plus efficacement. Nous avons également dû apprendre à travailler sur le projet tout en suivant d'autres enseignements et en réalisant le travail personnel correspondant, et donc à gérer notre temps.

L'aspect informatique nous a obligé à avoir rigueur et autonomie. Nous nous sommes lancés dans un projet où tout doit être parfait. Nous avons dû apprendre certains langages de programmation, lire et comprendre les documentations à disposition, essayer, tester, pour trouver une solution aux problèmes que nous avons rencontrés.

Et après ?

Nous sommes arrivés exactement là où nous voulions pour ce projet mais, ce n'est pas terminé. En effet, il est voué à évoluer pour devenir encore plus performant. Aussi, c'est un système qui a été pensé pour être le plus modulaire possible. C'est à dire que nous avons l'intention de créer des modules permettant d'ajouter des fonctionnalités, telles que des capteurs de températures, des détecteurs… L'application a aussi vocation à évoluer, en implémentant de nouvelles fonctionnalités.

Annexe

Nos pistes de réflexions

De la plus récente à la plus ancienne

Réflexion finale

Le serveur sera mis en place sur un Raspberry pi et écrit en NodeJS. Deux scripts Python permettrons de faire l'intermédiaire entre le serveur et les deux Arduino. Un Arduino permettra l'envoi de données en RF, tandis que le second recevra des données en RF.

Un troisième Arduino coté client sera en mesure de recevoir des données du serveur uniquement et d'en envoyer. Il lui sera branché un interrupteur permettant d'inverser l'état de la lampe (les systèmes multi-interrupteurs ne sont pas à réfléchir). La lampe sera actionnée par un relais mécanique, permettant le contrôle d'appareil utilisant du 220V.

Le script Python entre le serveur et le récepteur permet la récupération des données envoyées par l'Arduino récepteur branché au serveur pour les intégrer à la base de données.

Le serveur s'occupe de traiter les modification effectuée dans la base de données.

Réflexions intermédiaires

Serveur sur le Raspberry Pi et contrôle des modules RF grâce aux GPIO[24]

Solution

Le serveur sera mis en place sur un Raspberry pi et écrit en NodeJS. Deux scripts Python permettrons de faire l'intermédiaire entre le serveur et les GPIO. Il sera branché sur ces derniers les modules d'émission et de réceptions RF.

Un Arduino coté client sera en mesure de recevoir des données du serveur uniquement et d'en envoyer. Il lui sera branché un interrupteur permettant d'inverser l'état de la lampe (les systèmes multi-interrupteurs ne sont pas à réfléchir). La lampe sera actionnée par un relais mécanique, permettant le contrôle d'appareil utilisant du 220V.

Electronique nécessaire :

- Raspberry Pi
- 2 émetteurs/récepteurs radio
- Relais mécanique
- Interrupteur
- Fil
- Led
- Résistance

Librairies nécessaires :

PiGPIO, pour le contrôle des pins

Virtual Wire, pour le contrôle des modules RF

[24] Les ports GPIO (anglais : General Purpose Input/Output, littéralement Entrée-sortie à usage général) sont des ports d'entrées-sorties.

Viabilité de la solution

L'utilisation des modules RF par le biais des GPIO est trop complexe, il faut changer de technique. En revanche, le reste est bon.

Serveur sur le Raspberry Pi et un Arduino branché au port série

Solution

Le serveur sera mis en place sur un Raspberry Pi. Un Arduino permettra l'envoi et la réception des données en RF.

Un Arduino coté client sera en mesure de recevoir des données du serveur uniquement et d'en envoyer. Il lui sera branché un interrupteur permettant d'inverser l'état de la lampe (les systèmes multi-interrupteurs ne sont pas à réfléchir). La lampe sera actionnée par un relais mécanique, permettant le contrôle d'appareil utilisant du 220V.

Viabilité de la solution

Cela n'est pas possible car un Arduino ne peut pas envoyer et recevoir de données via son port série en même temps.

Serveur un Arduino

Solution

Le serveur sera mis en place sur un Arduino et écrit avec les fonctions disponibles sur l'Arduino. Les modules RF sont reliés directement à cet Arduino.

Un troisième Arduino coté client sera en mesure de recevoir des données du serveur uniquement et d'en envoyer. Il lui sera branché un interrupteur permettant d'inverser l'état de la lampe (les systèmes multi-interrupteurs ne sont pas à réfléchir). La lampe sera actionnée par un relais mécanique, permettant le contrôle d'appareil utilisant du 220V.

Viabilité de la solution

Mettre en place un serveur sur un Arduino est une mission impossible. De plus, les possibilités offertes étaient trop peu nombreuses pour satisfaire nos envies.

Concept Idéalisé Initial

Software[25]

Page Web avec un switch pour changer l'état de la lampe

Envoie de la modification à un serveur

Modification en conséquence d'une base de données

Hardware[26]

S'il y a une modification de la base de données, alors cette dernière est récupérée et traitée pour modifier le circuit électronique, l'état de la lampe, en conséquence

[25] Software est le mot anglais pour le logiciel, un ensemble d'instructions données à un appareil informatique.

[26] Un matériel informatique (en anglais : hardware) est une pièce détachée d'un appareil informatique.

Bibliographie

Site Web

Documentations

NodeJS

Mozilla Developer Network (Mars 2019). MDN Web Docs. [En ligne]. URL : *https://developer.mozilla.org/en-US/*

Socket.io (Mars 2019). Node.js documentation. [En ligne]. URL: *https://devdocs.io/node/*

NPM (Mars 2019). Express. [En ligne]. URL: *https://www.npmjs.com/package/express*

NPM (Mars 2019). Serial Port. [En ligne]. URL: *https://www.npmjs.com/package/serialport*

NPM (Mars 2019). Python-shell. [En ligne]. URL: *https://www.npmjs.com/package/python-shell*

CanIUse (Mars 2019). Can I Use. [En ligne]. URL*:* *https://caniuse.com/*

DevDocs (Mars 2019). DevDocs API Documentation. [En ligne]. URL: *https://devdocs.io/*

Python

DevDocs (Mars 2019). Python 3.7 documentation. [En ligne]. URL: *https://devdocs.io/python~3.7/*

Arduino

DevDocs (Mars 2019). C++ documentation. [En ligne]. URL: https://devdocs.io/cpp/

Arduino (Mars 2019). Arduino Reference. [En ligne]. URL: *https://www.arduino.cc/reference/en/*

Tutoriels

NodeJS

Mathieu NEBRA (Mars 2019). Ultra-Fast Applications using Node.js. [En ligne]. URL : *https://openclassrooms.com/fr/courses/2504541-ultra-fast-applications-using-node-js/2504696-node-js-what-is-it-for-exactly*

Mozilla Developer Network (Mars 2019). Express Nodejs. [En ligne]. URL : *https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs*

W3Schools (Mars 2019). Nodejs and Raspberry Pi. [En ligne]. URL : *https://www.w3schools.com/nodejs/nodejs_raspberrypi.asp*

Python

Vincent Le Goff (Mars 2019). Apprenez à programmer en Python. [En ligne]. URL : *https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python*

Patrick de AnotherMaker (Mars 2019). Communication entre un Raspberry Pi et un arduino. [En ligne]. URL : *http://anothermaker.xyz/iot/communication-entre-un-raspberry-pi-et-un-arduino-5319*

Arduino

Jean-Noël Rousseau (Mars 2019). Apprenez à programmer en Python. [En ligne]. URL : *https://openclassrooms.com/fr/courses/2778161-programmez-vos-premiers-montages-avec-arduino*

Gus de PiMyLifeUp (Mars 2019). Apprenez à programmer en Python. [En ligne]. URL : *https://pimylifeup.com/arduino-web-server/*

Miguel de AllAboutEE (Mars 2019). Apprenez à programmer en Python. [En ligne]. URL : *http://allaboutee.com/2014/12/30/esp8266-and-arduino-webserver/*

bobyAndCo de Locoduino (Mars 2019). Piloter son Arduino avec son navigateur web et Node.js. [En ligne]. URL : *http://locoduino.org/spip.php?article216*

Crasy Craft (Mars 2019). Alarme arduino par internet via page web. [Vidéo en ligne]. URL : *https://www.youtube.com/watch?v=URiXYVpVZvA*

U=RI (Mars 2019). U=RI. [Vidéo en ligne]. URL : *https://www.youtube.com/channel/UCVqx3vXNghSqUcVg2nmeqYA*

Grafikart (Mars 2019). Grafikart.fr. [Vidéo en ligne]. URL : *https://www.youtube.com/user/grafikarttv*

Julian Ilett (Mars 2019). ESP8266 Hack #1: Web Enabled LED WiFi Internet-of-Things IoT. [Vidéo en ligne]. URL : *https://www.youtube.com/watch?v=VvIoBFLj2Xo&feature=youtu.be*