Rapport technique final:

Conception du système de contrôle domotique

ELE795

Remis à

Ricardo Izquierdo

Fait par

Nicolas Gauvin-Gingras

Marc-Olivier Lanthier

Toma Gosselin-St-Pierre

Maxime Vallée-Chéné

Remis le

10 août 2021

LISTE DES TABLEAUX

TABLE DES ILLUSTRATIONS

LISTE DES ABRÉVIATIONS SIGLES ET ACRONYMES

AC	Alternating current -> Courant alternatif
API	Application Programming Interface
CFM	Cubic Foot per Minute
DEL	Diode électroluminescente
DC	Direct current -> Courant continu
FreeRTOS	Free Real Time Operating System
GCC	GNU Compiler Collection
GNU	GNU's Not Unix
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
PCB	Printed Circuit Board
PSU	Power Supply Unit -> Bloc d'alimentation
DNS	Système de noms de domaine

LISTE DES SYMBOLES ET UNITÉS DE MESURE
(SYSTÈME INTERNATIONAL)

UNITÉS DE BASE

m	mètre (unité de longueur)
s	seconde (unité de temps)
A	ampère (unité d'intensité de courant électrique)
mA	Milliampère, 1mA = 0.001A
C	Celsius (unité de température)
V	Volt (unité d'énergie potentielle)

PUISSANCE

W	watt
mW	milliwatt

UNITÉS GÉOMÉTRIQUES

Longueur

m	mètre
dm	décimètre
cm	centimètre
mm	millimètre
µm	micromètre

TABLE DES MATIÈRES

MISE EN CONTEXTE DU PROJET

La domotique est un sujet en vogue depuis maintenant quelques années dans le monde de l'électronique. Avec les solutions des géants de l'industrie telles que Google, Apple et Amazon, la domotique possède encore des atouts à offrir aux utilisateurs. Cependant, les techniques développées au sein de ces multinationales requièrent des appareils parfois coûteux pour un rendement médiocre si l'équipement rattaché est incompatible.

Alors, notre projet est une opportunité de démontrer qu'un système peut être à prix modeste et opérer sans problème autant à distance que localement. Notre projet n'est pas une fin en soi puisqu'il pourra toujours être amélioré. Cependant, ce projet permettra d'affirmer qu'il est possible de réaliser un système de contrôle domotique à bas prix et de réussir à l'utiliser avec l'équipement standard d'une maison. Tout cela dans l'optique de réduire et gérer efficacement l'énergie utilisée dans nos maisons.

Les objectifs du projet étaient de réussir à améliorer les capacités d'un système domotique en utilisant de l'équipement peu onéreux et capable de contrôler des lumières et le chauffage d'une maison. Tout cela dans le but de gérer efficacement ce qui coûte le plus cher en électricité dans une maison : le chauffage.

Alors, les objectifs peuvent être résumés en trois sections, soit les objectifs principaux, spécifiques et secondaires.

Objectifs principaux :

- Contrôler la température dans une pièce de maison entre 18° et 23° Celsius.
- Contrôler l'éclairage dans une pièce de maison.
- Obtenir un temps maximal de mise à jour de 120 secondes.

Objectifs spécifiques :

- Contrôler la température et l'éclairage par un seul contrôleur.
- Accéder et contrôler à distance le système domotique.
- Communiquer avec les capteurs de température de manière sans-fil.

Objectifs secondaires (optionnels) :

- Concevoir un système permettant de réduire les coûts et la consommation électrique.
- Optimiser le contrôle des lumières et de la température en fonction de l'environnement, des prix de l'électricité selon la région et de la détection de présence.

MÉTHODOLOGIE DE TRAVAIL

Pour développer notre système de contrôle domotique, nous avons utilisé plusieurs techniques d'ingénierie pour orienter notre démarche de façon efficace et efficiente. Notre projet pouvait se fragmenter en trois tâches, soit l'interface de communication, le contrôle de la logique et de la communication, ainsi que le montage matériel.

Afin d'utiliser nos ressources et connaissances au meilleur de nos capacités, nous avons divisé les tâches entre les membres de l'équipe :

Tableau 1 : Assignation des tâches

Tâches	Assignation
Interface	Nicolas
Contrôle	Toma
Acquisition	Maxime
Montage matériel	Marc-Olivier

Pour la communication au sein de l'équipe nous avons priorisé le logiciel Discord, puisqu'il est connu de tous les membres et facile d'utilisation. De plus, ce logiciel étant gratuit, il permet de clavarder et de réaliser des rencontres à distance par vidéoconférence.

Liste détaillée des tâches :

1. Conception de la maquette

   Construire une maquette contenant le système de domotique pour tester à petite échelle le projet.

2. Programmation du microcontrôleur

   Programmer le microcontrôleur pour transmettre les valeurs captées à l'unité centrale et effectuer les changements nécessaires selon l'utilisateur. Il permet également d'enregistrer la température voulue.

3. Conception de l'interface graphique

   Programmer une interface graphique afin d'afficher la température et les autres paramètres captés par l'unité centrale.

4. Test de l'interface graphique

   Tester si l'interface graphique est capable de communiquer avec l'unité centrale en transmettant ou en affichant différentes données.

5. Programmation de l'unité centrale

   Programmer l'unité centrale pour lui permettre de traiter les données captées et transmettre les données pertinentes à l'interface.

6. Test de l'unité centrale

   Tester si l'unité centrale est capable de recevoir les données de la maquette, de les traiter et de les envoyer à l'interface.

7. Branchement électrique

   Réaliser les branchements électriques de la maquette, du contrôleur et installer les capteurs dans la maquette.

8. Test des branchements électriques

   Tester les branchements pour vérifier si chaque capteur, élément chauffant et microcontrôleur fonctionne correctement.

9. Conception du système de contrôle de température.

   Construire un PID permettant de fixer la température à une valeur précise. La température devra varier entre 18° et 23° Celsius.

10. Test du système de contrôle de température.

    Tester le système de contrôle de température pour vérifier le temps nécessaire pour atteindre la chaleur désirée.

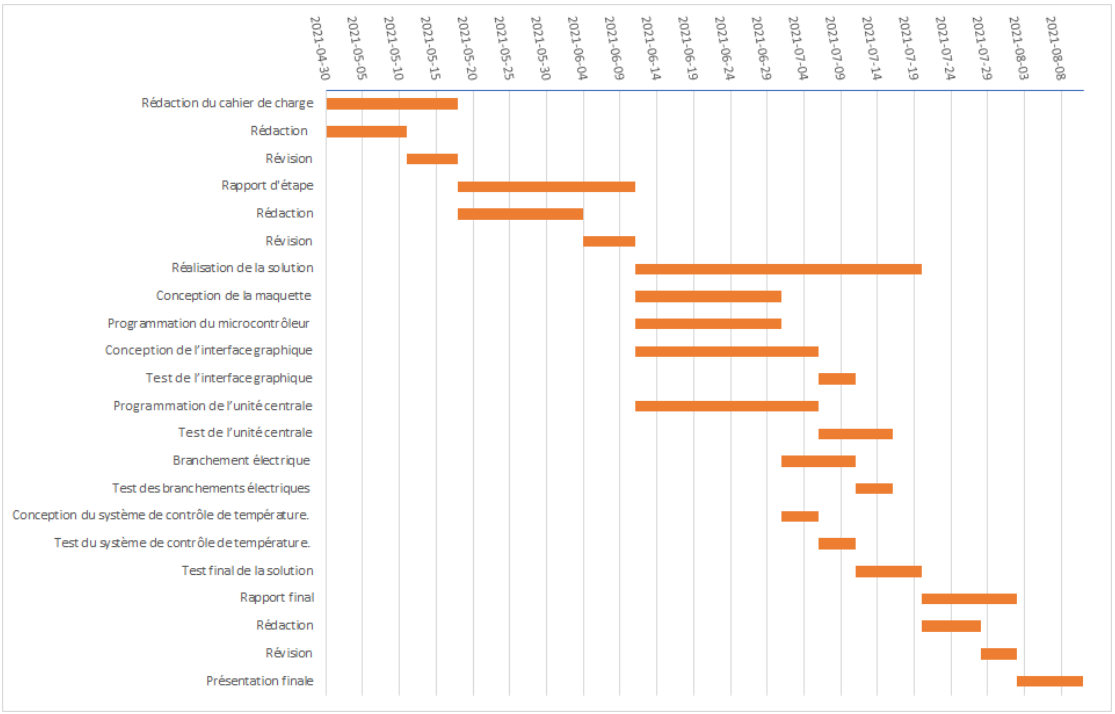


Figure 1: Diagramme de Gantt

Développement de l'interface

Le développement de l'interface était une partie purement logicielle. Il n'y avait donc pas d'outils ou d'équipements matériels nécessaires aux développements. Cependant, l'équipe utilisait la même interface de programmation, soit Visual Studio Code afin d'éviter les problèmes de compatibilité. De plus, le code était sauvegardé sur Git. Celui-ci permet une traçabilité des versions, une sauvegarde sur l'infonuagique, ainsi que la revue par les pairs. Nous pouvions donc, en tout temps, regarder l'avancement du travail de nos coéquipiers afin de donner des commentaires ou suggestions. Pour respecter les objectifs du projet, l'interface devait posséder au minimum une page d'accueil, ainsi qu'une page pour chacun des modules. Pour chaque module, le contrôle d'une lumière et de la température devait être présent.

Dans un premier temps, pour le développement de la page web proprement dit, des connaissances avec les langages de programmation HTML, JavaScript et CSS étaient nécessaires. Ces connaissances permettent la construction de la structure de la page, c'est-à-dire le titre, les paragraphes, les hyperliens, etc. D'un autre côté, le langage CSS a permis d'ajuster le format des éléments, ainsi que l'apparence. Finalement, le JavaScript établit le lien entre l'affichage sur la page web et le serveur.

Par la suite, le serveur devait être programmé pour accueillir la page web. Ce serveur doit envoyer de l'information et des requêtes afin d'établir une communication avec l'utilisateur de l'interface web. Le langage Python a été sélectionné pour programmer cette partie du projet, puisqu'il existe de nombreuses fonctions bien documentées qui facilitent le développement d'un serveur. Ensuite, il suffit d'ajouter les fonctions à son projet et de prendre connaissance de la documentation. Des connaissances en Python étaient requises pour réaliser la mise sur pied du serveur.

Finalement, l'idée derrière l'interface est d'obtenir la plus grande flexibilité possible pour ajouter, retirer ou modifier les modules initialement présents. Avec l'architecture actuelle, il suffit de quelques modifications afin d'obtenir une interface qui supporte davantage de modules.

Développement du contrôleur

La partie contrôleur a été scindée en deux, puisque des tâches différentes devaient être réalisées. En effet, la partie contrôleur est le cœur du système, cette section prend toutes les décisions relatives au contrôle des lumières et de la température et exécute, en plus, la communication. Tandis que la partie acquisition est utilisée pour obtenir l'information sur la température et l'état des lumières.

Contrôleur

Pour la partie contrôleur, nous avons utilisé la méthode Agile pour développer les fonctionnalités du projet. Cette méthode consiste principalement à faire de petites itérations de développement très rapides avec une validation constante.

Toutes les parties du projet ont été séparées en tâches qui peuvent être implémentées et tester séparément. Pour simplifier ce développement, nous avons créé une tâche de prototypage qui permet de développer les différentes fonctions indépendamment des autres parties du système et de les déplacer vers leur emplacement final.

Lorsque les fonctions principales du contrôleur étaient terminées, nous avons fait une réunion pour déterminer quelles autres fonctionnalités pourraient être bénéfiques au projet. Nous avons pris la décision d'implémenter un contrôle en deux points, c'est-à-dire, de l'interface web et de bouton physique.

Nous avons commencé par créer la tâche qui modifie l'interface, lorsqu'un changement est fait dans le programme. Ensuite, une fois cette fonctionnalité implémentée, nous avons modifié les tâches existantes, afin d'aller modifier les valeurs du contrôle.

La tâche de contrôle et de capteur a été combinée, cette décision d'équipe rendit la tâche globale pour chaque pièce de la maquette. Donc, cela se traduisit par l'affectation du ESP32 au contrôle et captation d'une pièce. Cette décision fut prise en groupe, lors des réunions quinzomadaires, dues aux restrictions de temps et de matériel.

La tâche de communication entre l'interface, les fichiers de contrôles et les capteurs ont été séparés. Le plan initial était d'avoir un seul fichier pour les deux. Cependant les restrictions imposées par le serveur web nous ont fait dévier du plan d'origine. Cela a été facilement modifiable, car nous avons appliqué le principe de modularité au projet et n'avions seulement qu'une tâche à modifier pour adapter le programme avec la nouvelle configuration.

De plus, toutes les fonctions possèdent des commentaires et des entêtes qui expliquent les fonctionnalités du programme. Cette méthode a été utilisée pour faciliter la modification par d'autre développeur pour un meilleur travail d'équipe.

Acquisition

L'acquisition est un élément clef du projet, car sans lui il serait impossible d'obtenir des informations sur l'état des lumières et de la température. Les modules d'acquisition sont opérés par des microcontrôleurs. On retrouve un appareil ESP32 par pièce pour réaliser la partie acquisition. De plus, un capteur d'humidité et de température ainsi que des relais contrôlant le chauffage, la ventilation et la lumière sont reliés à chaque microcontrôleur. Enfin, trois interrupteurs ont été reliés afin de contrôler la lumière et le chauffage.

Pour parvenir à brancher adéquatement les appareils mentionnés précédemment, plusieurs techniques ont été nécessaires. Pour débuter, le capteur d'humidité et de température a nécessité une recherche approfondie sur son fonctionnement, car celui-ci est de type numérique. Ainsi, le capteur ne pouvait être branché sur une broche analogique du ESP32 pour lire une variation de tension, puisque celui-ci est de type numérique. Afin de parvenir à décoder les données envoyées par le capteur, il est nécessaire de lire le train de bits envoyés[1]. Ensuite, le chauffage, la lumière, et le ventilateur sont branchés aux relais contrôlés par le microcontrôleur. Finalement, les interrupteurs de contrôle du chauffage et de la lumière ont aussi été branchés dans les broches numériques. Ces interrupteurs sont tous branchés au ESP32 par un circuit de rappel au niveau bas afin d'éviter des broches flottantes sur le microcontrôleur.

[1] Voir Annexe 1 sur le fonctionnement du capteur de température DHT11

Lors de la mise en place de l'environnement de travail logiciel, une embûche imprévue est survenue. La programmation des microcontrôleurs à l'aide de l'environnement développé par le manufacturier Espressif était très instable. Des problèmes au niveau de la compilation et programmation apparaissaient aléatoirement. Alors, en approfondissant les recherches, une solution a été trouvée. Cette solution a été d'utiliser une autre extension au sein de l'IDE qui se basait sur l'environnement Arduino. Donc, l'extension Platform.IO fut utilisée.

À ce moment, le schéma bloc de la partie acquisition a été créer pour aider à orienter l'écriture du code informatique.

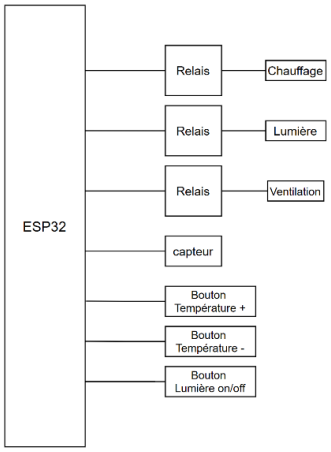


Figure 2: Schéma bloc acquisition

Pour réaliser la programmation des microcontrôleurs, des connaissances en langage C et C++ étaient requises. L'environnement de développement intégré (IDE) utilisé pour cette section était Visual Studio Code pour obtenir une versatilité et uniformité au sein de l'équipe, puisque ce logiciel gère plusieurs langages de programmation. De plus, pour effectuer un suivi de l'avancement du développement du programme le logiciel de gestion de version Git fut employé.

Construction de la maquette

La maquette est un élément très important du projet. Celle-ci permet de visuellement observer les actions réalisées par l'utilisateur. De plus, elle permet de contrôler physiquement le système de contrôle à l'aide des interrupteurs sans passer par l'interface.

La structure de la maquette est faite de bois avec une devanture en plastique transparent afin d'observer l'intérieur de la maquette. La partie inférieure est un rectangle de 36x22.86 cm divisé en deux pièces identiques, séparé par un mur central. L'électronique est installée sur le plafond de la maquette. Pour protéger l'électronique et augmenter l'esthétisme, un toit de forme triangulaire est ajouté par-dessus les circuits.

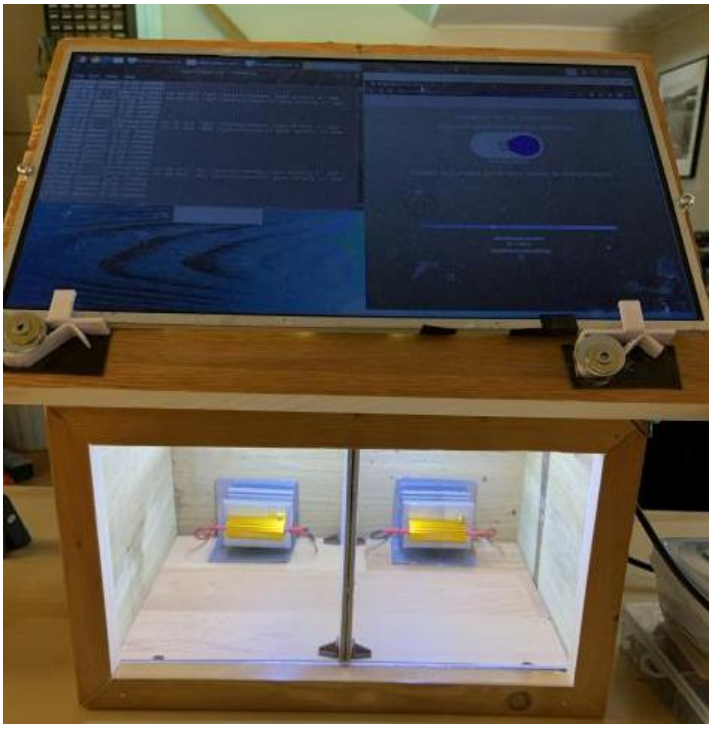


Figure 3 : Photo de la maquette finale

La section électrique est divisée en deux circuits, le circuit de contrôle et de puissance. Le circuit de contrôle permet d'activer les relais à l'aide des ESP32 (un pour chaque pièce) suivant les commandes envoyées par l'utilisateur. Le circuit de puissance alimente les appareils nécessitant des courants élevés. Ceux-ci seront actionnés par l'entremise des relais.

Un total de six composants pour les deux pièces seront contrôlables par l'utilisateur. En effet, chaque salle possède une résistance qui fait office de chauffage, un ventilateur et une lumière. Chacune de ces composantes sera contrôlée par un relais. Une autre composante du circuit est le capteur de température DHT11 qui permet de mesurer la température et l'envoyer à l'ESP32.

Le tableau des spécifications ci-dessous donne le détail sur les composants utilisés.

Tableau 2 : Spécification sur la maquette

Composante	Alimentation	Puissance absorbée	Rôle
Résistance	12 Volts	18 Watt (1.5A)	Chauffer la pièce
Ventilateur	12 Volts	0.36 Watt (0.03A)	Refroidir la pièce
LED	12 Volts	6W (0.5A)	Éclairer la pièce
DHT11	5 Volts	0.0015W (0.3 mA)	Mesurer la température
ESP32	5 Volts	2.5W (0.5A)	Contrôler les relais et transmettre les différentes données
Relais	5Volts	5W (1A)	Piloter les différents éléments contrôlables.

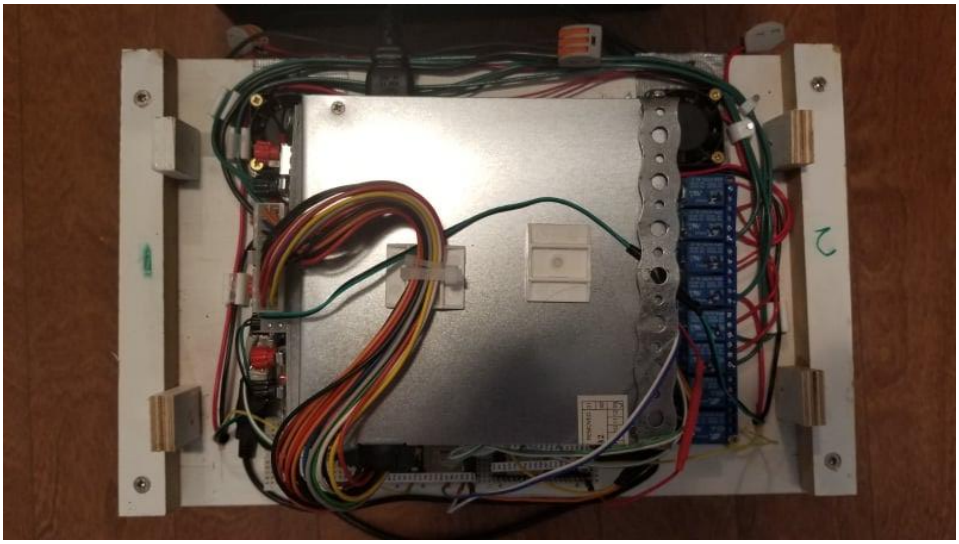


Figure 4 : Vue du dessus de l'alimentation et des relais

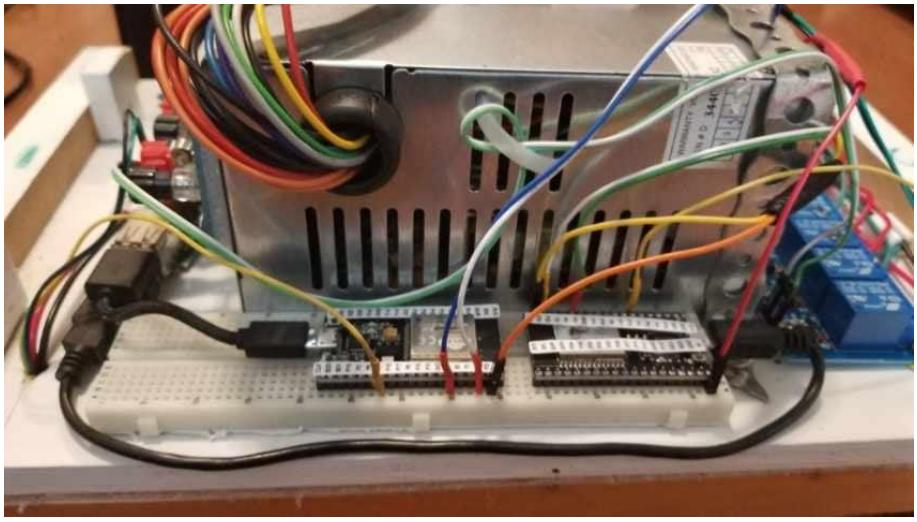


Figure 5 : Vue de côté de l'alimentation et des deux capteurs

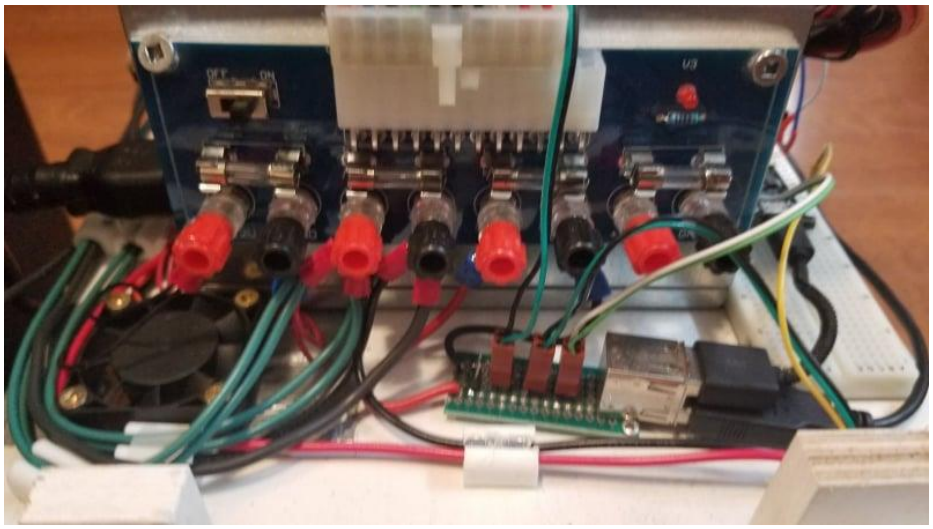


Figure 6 : Vue de la sortie de l'alimentation

Un PCB composé de huit relais est utilisé pour effectuer le contrôle de la lumière, du chauffage et de la ventilation. Ils ont été choisis pour plusieurs raisons. Premièrement ils sont activables avec 5 Volts, ce qui les rend directement contrôlables par l'ESP32. Deuxièmement, ils résistent à un courant électrique de 10A, ce qui est très primordial pour alimenter les résistances. Un autre avantage est l'isolation galvanique qu'ils procurent, puisqu'ils sont équipés d'optocoupleurs. Alors, advenant un problème le reste du circuit est protégé.

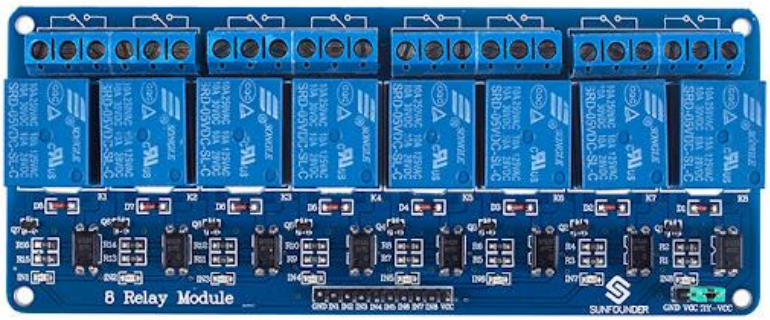


Figure 7 : PCB composé de 8 relais

Pour alimenter cette maquette, une source d'alimentation (PSU) d'ordinateur est employée. Ce choix a été retenu, car le bloc d'alimentation possède différents niveaux de tension. Il peut également fournir énormément de courant selon le tableau 3. Ce bloc d'alimentation aura comme rôle d'alimenter toutes les composantes. Pour ce faire, le PSU a été modifié afin d'offrir quatre tensions différentes : 3.3V, 5 V, 12V et –12V.

Tableau 3 : Spécification du PSU

Tension (V)	Courant (A)	Puissance (W)
3.3V	28A	92.4
5V	40A	200
12V	18A	216
-12V	1A	12

Tableau 4 : Composantes associées à chaque alimentation

Tension (V)	Composantes alimentées
3.3V	Non applicable
5V	ESP32, relais, DHT11, bouton
12V	Élément chauffant, DEL
-12V	Ventilation

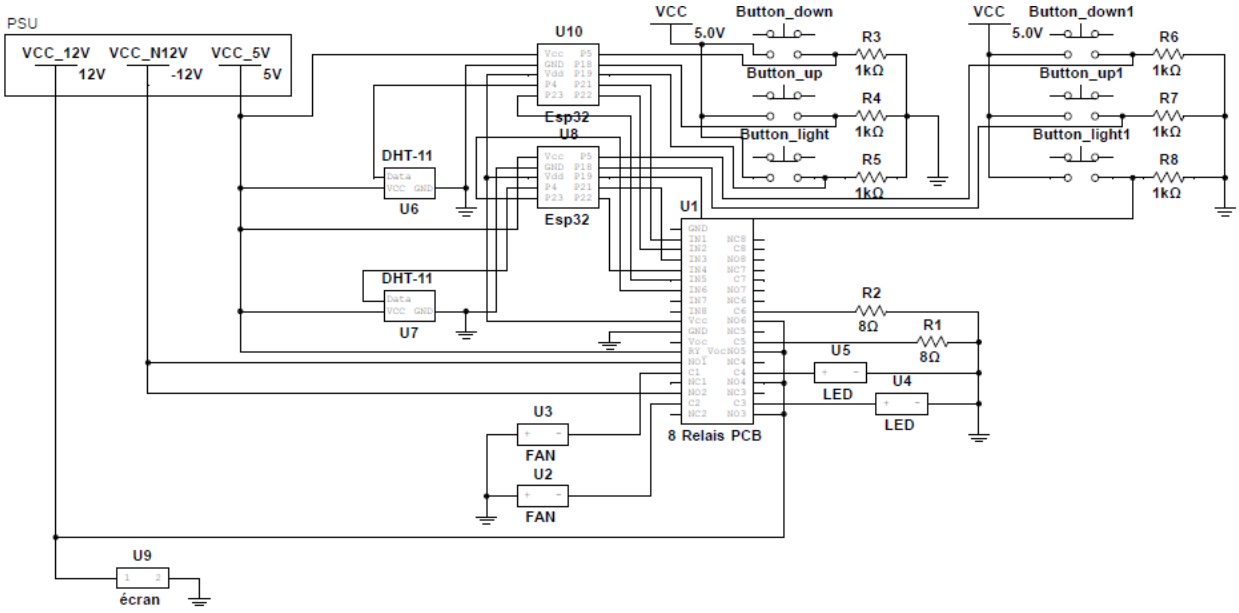


Figure 8 : Schéma électrique des branchements du PSU aux relais

Un autre facteur important à tenir compte lors de ce projet est la circulation de l'air dans la maquette. En effet, pour bien contrôler la température dans une pièce il est important de toujours avoir une rotation de l'air. Si l'air de la pièce est simplement immobile, une stratification de l'air risque de se créer. Ce qui signifie qu'il fera beaucoup plus froid dans le bas que dans le haut de la pièce. Pour permettre à l'air de bien circuler et bien réguler la température dans la pièce, il est important d'avoir une entrée et une sortie d'air. Pour la maquette, l'entrée passe par le ventilateur et la sortie est un trou positionné sur le côté en haut de chaque pièce.

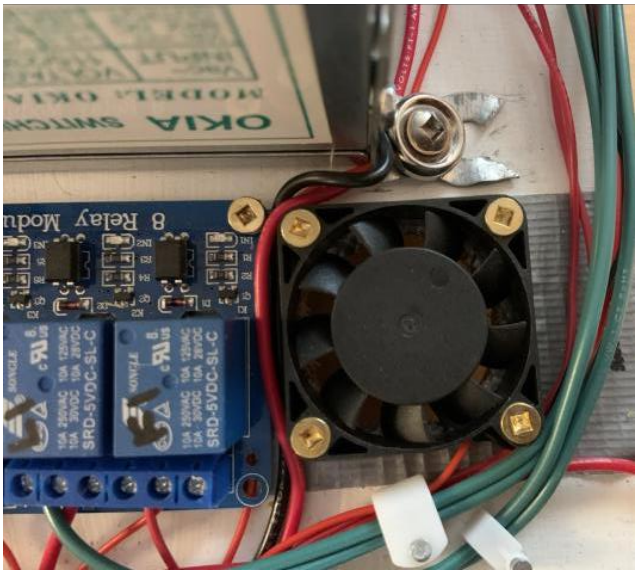


Figure 9 : Entrée d'air par le ventilateur

Figure 10 : Sortie d'air sur le côté de la maquette

Dans cette maquette, le risque d'électrocution est faible. En effet, la tension la plus élevée est de 12 Volts. En revanche, le risque d'incendie est important, puisqu'il y a un élément chauffant à l'intérieur d'une maquette en bois. Si la résistance entre en contact avec un mur, un incendie peut se déclarer. Cette unité de chauffage peut atteindre des températures avoisinant les 170 °C, et ce, en 5 minutes.

Le bois possède un point d'inflammation d'environ 200 °C, il est dangereux de fixer les éléments chauffants directement sur le bois. Pour contourner ce problème, les résistances ont été fixées sur des radiateurs d'aluminium, dont ceux-ci ont été placés en suspension par des vis. De cette façon, la chaleur est en grand parti diffusé par le radiateur et empêche un contact direct avec les parois.

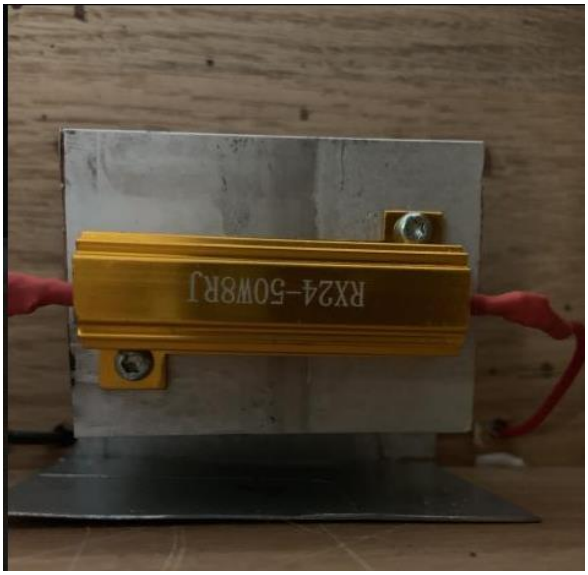


Figure 11 : Résistance en suspension pour l'écoulement de l'air

Les vis avec lesquels le radiateur est fixé au bois constituent encore un risque d'incendie. Pour empêcher la transmission de chaleur, les vis ont été entourées de téflon. Ce matériau résiste à des températures de 300 °C et est un bon isolant thermique. De cette façon, le risque d'incendie dans la maquette est réduit de manière considérable.

L'écran utilisé pour afficher l'interface locale provient d'un ordinateur portable recyclé. Conjointement avec une carte pilote, l'écran est utilisé comme moniteur et affiche l'interface de contrôle. Il y a plusieurs avantages à utiliser cet écran. Premièrement, elle est également légère et a une épaisseur de 0.7cm. Ensuite, la tension d'alimentation pour faire fonctionner l'écran est de 12 Volts. Elle peut donc directement se brancher dans le bloc d'alimentation.

Finalement, plusieurs mesures de sécurité ont été mises en place pour obtenir un prototype sécuritaire. La première mesure de sécurité est le positionnement des éléments chauffants. En effet, ils sont positionnés de sorte à ne pas avoir de contact direct avec le bois. Ces mesures ont été prises pour éviter un risque d'incendie. Une autre mesure est l'ajout de fusible sur le filage du bloc d'alimentation. Les fils des éléments chauffants ont un diamètre de 16 AWG pour éviter la surchauffe. Les relais sont également équipés d'optocoupleur. L'alimentation est donc électriquement séparée avec les entrées du microcontrôleur.

COMBINER, ADAPTER OU CRÉER DES OUTILS ET TECHNIQUES

Pour développer notre système de contrôle domotique, plusieurs outils ont été évalués pour chacun des aspects. Notre projet pouvait se fragmenter en trois tâches, soit l'interface de communication, le contrôle de la logique et de la communication, ainsi que le montage matériel. Cette section présentera l'analyse et la comparaison des différents outils pour chacune de ces trois tâches.

Interface

Tout d'abord, autant pour le développement de l'interface et la logique de contrôle, un environnement de développement pour écrire et compiler le code était indispensable. Il y a amplement de possibilités pour des logiciels gratuits facilement accessibles sur le marché : Visual Studio, VS Code, Éclipse, Code Lite, etc. L'équipe a choisi le logiciel VS Code, car il offre une flexibilité que beaucoup n'offrent pas. En effet, avec ce logiciel, il suffit d'aller dans l'onglet Extension pour ajouter des fonctionnalités. Ce produit offre la possibilité de coder en plusieurs langages de programmation au sein d'une même plate-forme. De plus, cela résout les problèmes de compatibilité et garde l'intégralité des fonctions IntelliSense. Ceci offrait une simplicité, puisque notre projet devait être développé à l'aide de plusieurs langages de programmation, soit Python, C, C++, HTML, CSS et JavaScript.

Par la suite, pour l'écriture du HTML et du CSS, il existe plusieurs outils pour simplifier le développement et offrir des gabarits simples : Pure CSS, Bootstrap et Boilerplate. Ces outils offrent de l'aide pour la mise en page du code, la flexibilité de la page, ajouter des d'animation ou toute autre fonction utile au développement. L'outil Bootstrap fut retenu, puisque certains membres de l'équipe possédaient des connaissances sur l'utilisation de ce logiciel. Ces connaissances simplifiaient l'écriture de l'interface et accéléraient le développement de notre produit, puisque nous avions une personne-ressource vers qui nous tourner. De plus, notons la florissante communauté d'utilisateur de Bootstrap ainsi que sa documentation abondante.

Ensuite, nous devions créer une interface de programmation d'application (API) ainsi qu'un serveur. Dans le but d'obtenir une communication entre la page web (l'utilisateur) et le code source de contrôle du microcontrôleur. Pour ces applications, deux langages de programmation ont été évalués, soit le C++ et le Python. Le grand avantage du C++ est le contrôle de l'espace mémoire. Il est beaucoup plus facile de gérer les ressources en C++ qu'en Python. Cependant, la communauté derrière le Python, ainsi que les paquets disponibles sur le web simplifiaient le développement. Puisque notre programme est peu gourmand en ressource, le contrôleur pouvait amplement supporter les ressources supplémentaires qu'amenait l'utilisation du Python. Ainsi, le serveur et l'API ont été créés à l'aide de Python et de paquets disponibles en ligne.

Contrôleur

POSIX Threads aussi appelé « pthreads » permet aux programmes sous Linux d'exécuter des tâches qui sont superposées dans le temps. Chaque tâche est appelée un « thread » et permet d'avoir accès à des variables partagées entre les différentes tâches. Les commandes de « pthreads » sont principalement divisées en quatre catégories soit la gestion des tâches, les mutex, les variables de condition et la synchronisation entre tâches. De plus, le programme utilise les fonctions de sémaphores qui se trouvent dans la librairie « sem » et non « pthread ». Cependant, cette librairie respecte toutes les normes POSIX.

Il aurait été possible de ne pas utiliser de fonction de gestion de tâche, mais dans le cadre du projet, il est important de garder le temps de réponse faible et de ne pas exécuter des fonctions inutiles.

Pour effectuer la compilation du programme en C, nous utilisons Cmake pour indiquer au compilateur les fichiers à traiter. Cet outil nous permet d'indiquer le nom du programme, le langage de programmation et les librairies qui sont liées au projet. Tous ces paramètres doivent être indiqués dans le fichier « CMakeLists.txt », afin que le compilateur prenne en compte les paramètres entrés.

Il aurait aussi été possible de coder le Makefile directement (qui est le type de fichier créer par Cmake pour compiler le C), mais celui-ci est beaucoup plus complexe à utiliser et possède moins de fonctionnalité intégrer, ce qui complexifie grandement la compilation du programme.

Nous utilisons GCC pour compiler notre code en C. Ce compilateur permet de compiler plusieurs langages de programmation, dont le C, C++ et le Go. Il fait partie du projet GNU qui possède l'objectif de fournir une solution gratuite et respecter la liberté des utilisateurs. Comme celui-ci est un environnement de développement ouvert, cela en fait un compilateur de choix s'alignant à la philosophie du projet.

Dans les choix de compilateur de C que nous aurions pu utiliser au lieu de GCC, Clang était une option, cependant l'intégration de GCC avec les extensions de VS Code permettait un développement plus rapide qu'avec Clang.

L'outil Gdbserver permet de débugger à distance un autre programme. Pour ce projet, nous utilisons « GNU gdb (Raspbian 8.2.1) », celui-ci va s'attacher au programme backend. Cet outil va permettre d'insérer des points d'arrêt dans le programme. Ils vont permettre d'obtenir l'état des variables et les exceptions qui se produisent. En plus, cela permet de visualiser les tâches en cours d'exécution.

L'affichage en ligne de commande aurait été possible pour trouver les problèmes du programme, mais cela amenait un grand désavantage. Il est impossible de mettre le programme sur pause et d'analyser les états des variables à un moment précis.

Afin d'obtenir les différents messages qui sont reçus et envoyés entre le backend et les capteurs, le logiciel utilisé est Wireshark. Celui-ci permet d'observer tous les paquets qui circulent par Wi-Fi ainsi que tous les détails sous-jacents à ceux-ci. Pour faciliter l'analyse des paquets, des filtres sur les données ont été appliqués pour limiter l'affichage des messages. Ils ont éliminé les paquets qui n'étaient pas pertinents au projet. Wireshark a été retenu, car il est facile d'utilisation et la configuration est simple. De plus, il est disponible sur Linux.

Pour l'IDE, nous utilisons VS Code comme expliquer à la section interface. Ce logiciel à plusieurs extensions qui permettent d'améliorer les capacités des outils décrits précédemment. Par exemple, certaines extensions permettent d'utiliser Cmake pour compiler ou bien lancer le programme. Il fournit aussi l'option de lancer directement Gdbserver. De plus, il inclut un formateur de code automatique qui est appliqué à la compilation des fichiers. Dans son interface de base, toutes les opérations principales de Git peuvent être réalisées.

Un logiciel payant, CLion, semblable à Cmake aurait pu être employé, puisqu'il possède des fonctionnalités similaires. En revanche, celui-ci étant payant cela bifurquait de l'idéologie du projet et réduisait l'accessibilité au grand publique.

Les capteurs se connectent au contrôleur via Wi-Fi, puisque celui-ci est configuré en mode point d'accès. Pour cela, il faut installer différents programmes et configurer l'interface réseau du Raspberry Pi afin d'émettre un réseau sans-fil. Un nom et un mot de passe fixes ont été utilisés comme identifiant. Le contrôleur est configuré en pont, cette méthode permet de faire passer le trafic circulant par le réseau dans l'interface sans-fil et câblé. Cela permet d'accéder au système à partir d'une connexion Internet, lorsque les ports du routeur sont ouverts.

Il aurait été possible d'utiliser le Bluetooth pour communiquer avec les capteurs, mais ce type de technologie sans-fil est limité. De plus, sa vitesse de transmission est moins élevée que le Wi-Fi.

Pour synchroniser le lancement de l'interface et du programme en arrière-plan, Bash est utilisé. Cet outil fait également partie de GNU. Bash est principalement le programme de l'interface en ligne de commande de Linux. Il permet d'accéder à tous les programmes installés sous Linux et les exécuter à partir d'une ligne de commande. Dans notre cas, des scripts ont été créés pour lancer les programmes nécessaires au projet en une seule commande.

Il aurait été possible d'utiliser un autre logiciel en C pour démarrer l'interface et le programme en arrière-plan. Cela aurait été efficace pour démarrer les deux, en revanche l'investissement de temps était substantiel, alors ce choix fut rejeté.

Acquisition

Au sein de l'environnement de programmation VS Code, l'extension Platform.IO a été utilisée pour programmer les microcontrôleurs. Cette extension fut préférée à l'IDE d'Arduino, puisqu'avec VS Code il était simple d'intégrer les différents langages de programmation des autres sections du projet. Alors, pour obtenir une plus grande versatilité et gagner en efficacité l'extension a été sélectionnée afin de compiler et téléverser le code créé pour les ESP32.

Afin de communiquer avec le capteur de température, nous avons utilisé et adapté la librairie Adafruit Unified Sensor créer par Adafruit. En effet, cette librairie a été créée pour faciliter l'acquisition de la température par les modèles DHTxx. Alors, en adaptant la librairie à nos besoins, cela nous a permis de communiquer avec le capteur DHT11 sans problème.

Le choix du ESP32 comme module d'acquisition avait été murement réfléchi et analyser. De cette analyse est ressortie la possibilité d'utiliser les deux cœurs d'un ESP32 pour réaliser des tâches en simultanées. Cela est rendu possible grâce à FreeRTOS. Ce système d'opération en temps réel (Real Time Operating System) permet de juxtaposer deux

instructions. Alors, il a été possible d'acquérir une lecture de température et de communiquer en même temps, en dédiant chaque tâche à un cœur. Cependant, FreeRTOS n'est pas le seul système qui permet à la puce d'Espressif de réaliser ces tâches. En revanche, lors de nos recherches et analyses nous avons découvert que FreeRTOS est en code ouvert et qu'il est parfaitement compatible avec le microcontrôleur utilisé. Alors, pour nous aligner à l'esprit du projet qui se voulait une solution ouverte et augmenter la compatibilité nous avons écarté les systèmes Zephyr et NuttX.

Maquette

Choix des matériaux :

Les matériaux ont été judicieusement choisis pour répondre au différent besoin du projet. Les murs et le plafond de la maquette sont en bois, car ce matériau est facile à travailler, peu onéreux et possède un point d'allumage élevé. Il est également un bon isolant et est représentatif d'une vraie maison. Le mur avant est réalisé en Plexiglas. Ce matériau a comme avantage d'être résistant aux chocs et transparent. Il est donc idéal pour regarder à l'intérieur de la maquette sans influencer la température intérieure. Pour le filage, la maquette utilise un code de couleur selon la tension associé. Le 12V est en vert, le 5 Volts est en rouge et la mise à la terre est en noir.

DOCUMENTATION TECHNIQUE DU PROTOTYPE

Interface

Joint à ce rapport se trouve tout le code nécessaire à la reproduction de l'interface. La structure du code se sépare en trois types de fichiers : les fichiers CSS pour l'apparence de l'interface, les fichiers HTML pour le contenu de l'interface et finalement, le fichier PY pour le serveur de l'interface. La section autre contient les fichiers JSON où l'interface écrit les différentes valeurs, ainsi que les photos sur la page d'accueil.

En résumé il y a les fichiers suivants :

Tableau 5 : Table résumée des fichiers d'interface

accueil.html	accueil.css	controlChambre.json
moduleChambre.html	modules.css	controlCuisine.json
moduleCuisine.html	tempratureSlider.css	controlSalon.json
moduleSalon.html	toggleSwitch.css	server.py

Pour lancer l'interface, aller dans le dossier ou le fichier binaire compiler précédemment se trouve et lancer avec la commande python « python server.py ». Dans ce cas spécifique, le dossier se trouve dans le dossier global qui peut être accédé grâce à la commande suivante « cd /home/pi/Documents/Projet_Final_Domotique/Interface/ ». Dans le cas spécifique du projet, un script Bash a été écrit et nommé « startInterface.sh » qui peut être lancé avec la commande « ./startInterface.sh ».

L'interface se décompose en deux sections : la page d'accueil et les pages des modules. Dans un premier temps, la page d'accueil présente sommairement l'équipe, les rôles, les contacts, ainsi que notre projet.

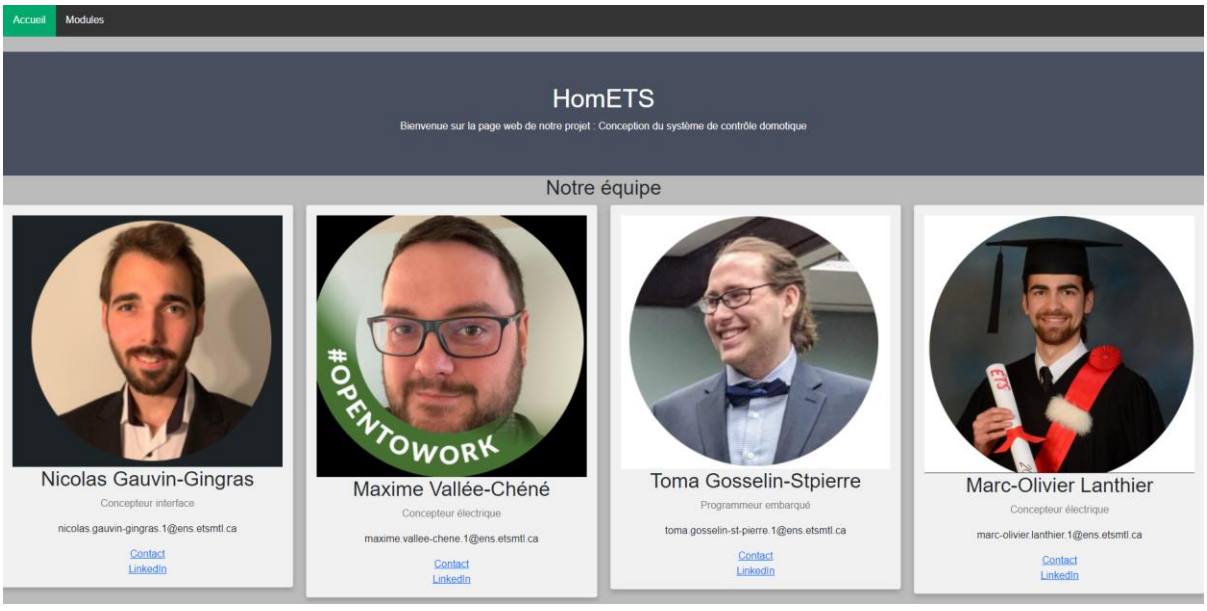


Figure 12 : Page d'accueil de l'interface web

Dans un deuxième temps, la barre de navigation permet de naviguer dans l'interface afin de contrôler les différents modules ajoutés au système de domotique. Le prototype possède deux pièces, donc deux modules. La barre de navigation contient ces deux modules, Chambre et Cuisine, permettant le contrôle de ces pièces. Sur chaque page des modules, on retrouve l'interrupteur pour allumer et éteindre la lumière dans la moitié supérieure de la page. L'état physique de la lumière apparait aussi sous l'interrupteur sous la forme de 1 ou 0. Dans la moitié inférieure de la page, on retrouve le curseur permettant de sélectionner la température désirée. La température sélectionnée, ainsi que la température réelle de la pièce apparaît sous la barre de sélection. L'unité utilisée est le degré Celsius.

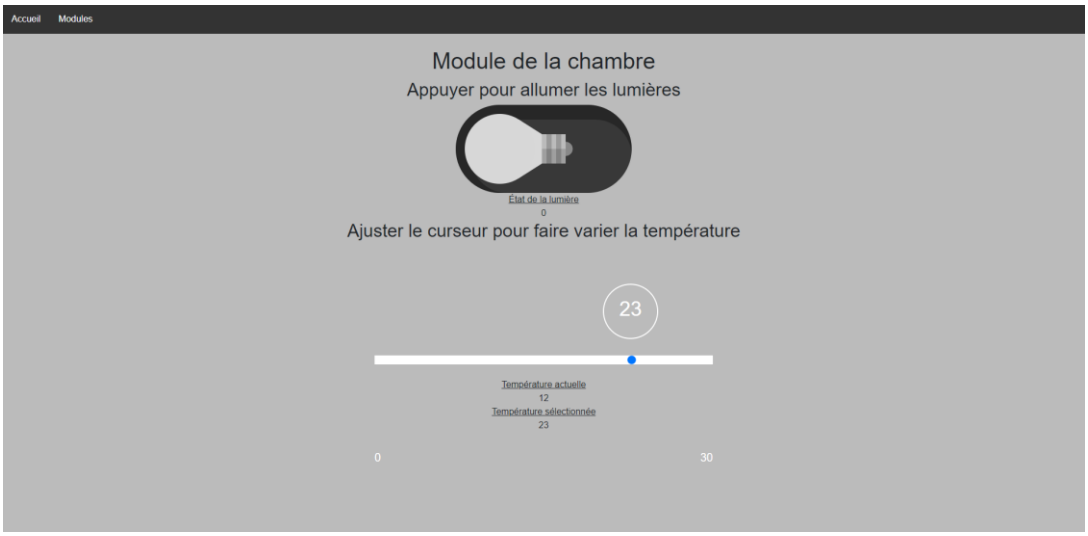


Figure 13 : Page du module Chambre de l'interface web

Pour permettre d'accéder à l'interface de partout sur Internet, il faut aussi faire l'ouverture du port 5000 sur l'adresse IP du Raspberry Pi qui est branché dans l'Ethernet. Cette procédure est différente pour tous les routeurs.

Contrôleur

Configuration

Préalablement à la programmation, il est nécessaire configurer le système d'exploitation du Raspberry Pi. Dans notre cas, nous avons choisi d'utiliser Raspberry Pi OS qui est préinstallé sur le Raspberry Pi qui est une distribution de Linux basée sur Debian, celui-ci peut être téléchargé sur le site officiel de Raspberry Pi.

Configuration du Wi-Fi

Pour la configuration de l'émetteur Wi-Fi, il faut commencer par une mise à jour du système Linux avec les commandes suivantes :

- sudo apt-get update
- sudo apt-get upgrade

Pour configurer le Raspberry Pi en point d'accès Internet sans fil, on utilise cette commande

- sudo apt-get install hostapd

Ensuite, pour installer dnsmasq on saisit cette commande pour créer un serveur DNS.

- sudo apt-get install dnsmasq

Pour configurer une adresse IP statique, il faut ouvrir le fichier de configuration avec :

- sudo nano /etc/dhcpcd.conf

Alors, il est important d'ajouter les lignes suivantes à la fin du fichier :

- interface wlan0
- static ip_address=10.10.10.10/24
- denyinterfaces eth0
- denyinterfaces wlan0

Pour sauvegarder, il faut appuyer sur Ctrl+X et Y.

Pour configurer le masque du DNS on utilise :

- sudo nano /etc/dnsmasq.conf

Après on ajoute ces lignes dans ce fichier. :

- interface=wlan0
- dhcp-range=10.10.10.150,10.10.10.200,255.255.255.0,12h

Pour configurer le point d'accès, il faut modifier le fichier avec :

- sudo nano /etc/hostapd/hostapd.conf

Puis, ajouter les lignes suivantes :

- interface=wlan0
- bridge=br0
- hw_mode=g
- channel=7
- wmm_enabled=0
- macaddr_acl=0
- auth_algs=1
- ignore_broadcast_ssid=0
- wpa=2
- wpa_key_mgmt=WPA-PSK
- wpa_pairwise=TKIP
- rsn_pairwise=CCMP
- ssid=HOMETS
- wpa_passphrase=VerySecurePassword3!

Afin que le système reconnaisse le nouveau fichier, il faut modifier le fichier avec la commande suivante :

- sudo nano /etc/default/hostapd

Ensuite, décommenter la ligne :

- DAEMON_CONF="/etc/hostapd/hostapd.conf"

Pour que le système transfère le trafic vers le câble Ethernet, il faut modifier le fichier avec :

- sudo nano /etc/sysctl.conf

Puis décommenter la ligne :

- net.ipv4.ip_forward=1

Pour avoir le masquage IP des adresses, il faut saisir les trois commandes suivantes :

- sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
- sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
- iptables-restore < /etc/iptables.ipv4.nat

Pour permettre aux appareils connectés d'accéder à Internet il faut faire quelques commandes de plus, soit installer le paquet de pont avec :

- sudo apt-get install bridge-utils

Ensuite, on ajoute un nouveau pont avec :

- sudo brctl addbr br0

Connecter notre interface au pont avec :

- sudo brctl addif br0 eth0

Enfin, pour éditer les interfaces avec :

- sudo nano /etc/network/interfaces

Puis ajouter les lignes suivantes :

- auto br0
- iface br0 inet manual
- bridge_ports eth0 wlan0

Finalement, il reste à redémarrer le Raspberry Pi pour appliquer toutes les modifications.

Backend

Compilation

Pour être en mesure de compiler le code informatique à l'aide de l'outil Cmake deux librairies doivent être présentes sur l'ordinateur.

Premièrement, la librairie PthRead doit être importée avec l'outil Cmake. Ensuite, la librairie JSON-C, devra être installée au préalable. Pour ce faire, la commande suivante est utilisée dans le terminal :

- sudo apt install libjson-c-dev

Pour faciliter la compilation, il est suggéré d'utiliser VS Code avec les extensions suivantes :

- C/C++ de Microsoft
- CMake de Twxs
- CMake Tools de Microsoft
- Markdown Preview Enhanced de Yiyi Wand
- Git Graph de Mhutchie

Enfin, pour créer des tâches sur Linux, nous utilisons Pthread.

Lancement

Pour lancer le backend, atteindre le dossier où le fichier binaire compiler précédemment se trouve et lancer la commande Bash suivante :

- ./backend

Pour ce projet, le dossier se situe dans le dossier global qui peut être accédé grâce à la commande suivante :

- cd /home/pi/Documents/Projet_Final_Domotique/backend/build/

Dans le cas spécifique du projet, un script Bash a été écrit et nommer « startBackend.sh » qui peut être lancé avec la commande suivante

- ./startBackend.sh

Structure

Pour ce qui est de la structure du programme, elle est séparée en tâche qui va partager certains éléments entre elles.

Tâche d'enregistrement

La tâche d'enregistrement permet à toutes les autres tâches de sauvegarder les erreurs et les actions importantes. Elle va ajouter des étampes de temps aux messages avant de les envoyer dans un tampon circulaire. Quand la tâche s'exécute, les valeurs de la mémoire tampon sont écrites dans un fichier log.

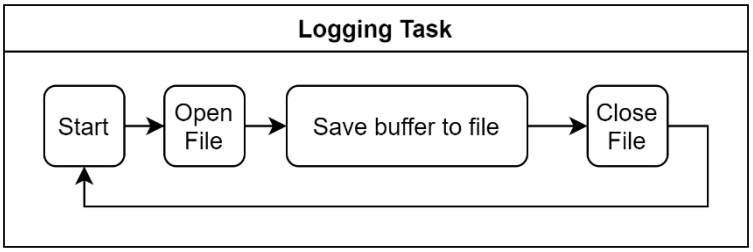


Figure 14 : Tâche d'enregistrement

Tâche d'interface

Cette tâche permet de communiquer avec l'interface grâce à des fichiers. Il est possible pour celle-ci de lire et écrire les fichiers de contrôle. Le fichier des capteurs doit être écrit avec les valeurs reçues. Alors, elle communique avec les tâches de sauvegarde de capteur, de lecture, de contrôle et de sauvegarde de contrôle. Ensuite, elle en extrait les valeurs nécessaires. Enfin, elle écrit les valeurs requises par les autres tâches.

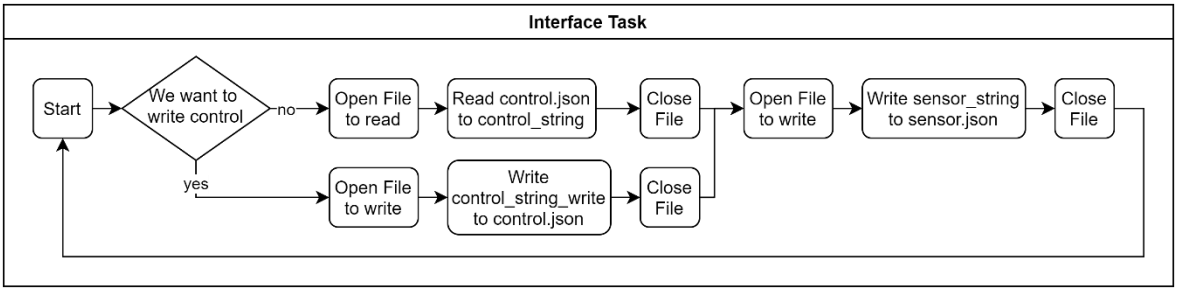


Figure 15 : Tâche d'interface

Tâche de sauvegarde de capteur

La tâche de sauvegarde de capteur recherche les capteurs en utilisation. Pour chaque capteur en utilisation, elle va l'ajouter à la variable « sensor_json », lorsque tous les capteurs sont parcourus le « sensor_json » est modifié en texte pour que la tâche interface puisse l'écrire dans un fichier.

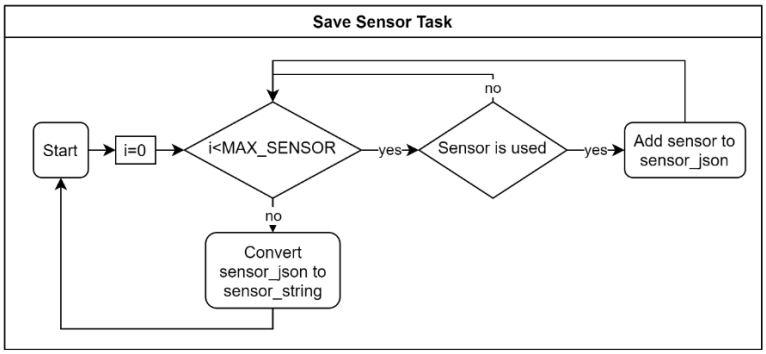


Figure 16 : Tâche de sauvegarde de capteur

Tâche de lecture de contrôle

Cette tâche prend le « control string » de la tâche interface et met à jour les contrôles existants.

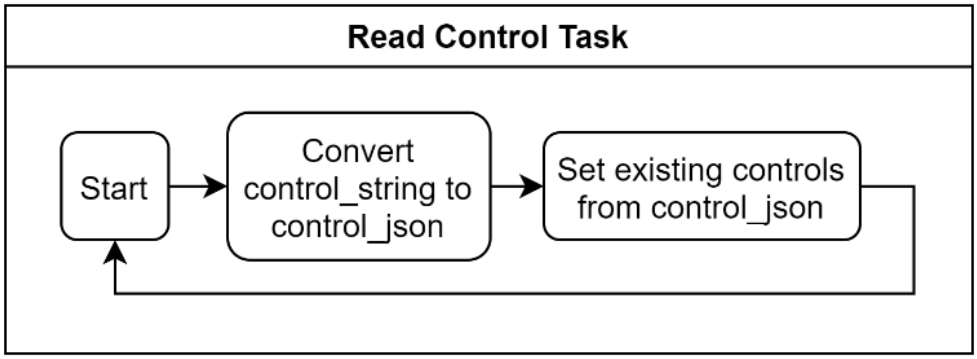


Figure 17 : Tâche de lecture de contrôle

Tâche de sauvegarde de contrôle

La tâche de sauvegarde de contrôle va parcourir tous les contrôles et vérifier ceux en utilisation. Pour les capteurs en utilisation, la tâche va l'ajouter à la variable « control_json » puis quand tous les capteurs sont parcourus le « control_json » est modifié en texte. Ensuite, la tâche interface l'écrit dans un fichier. Pour gérer l'écriture du fichier de contrôle, nous utilisons un sémaphore, qui écrit lorsqu'une valeur arrive des boutons des capteurs.

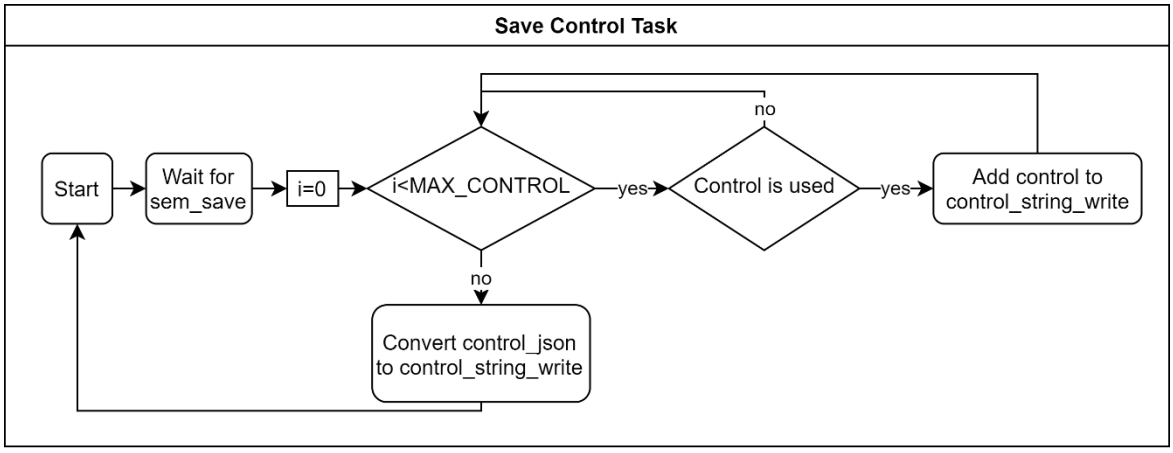


Figure 18 : Tâche de sauvegarde de contrôle

Tâche de recherche de capteurs

Pour la tâche de recherche de capteur, un serveur sera ouvert dans le programme qui va attendre une nouvelle connexion d'un des capteurs. Lorsqu'un capteur est découvert, la tâche crée une tâche capteur pour chaque capteur trouvé.

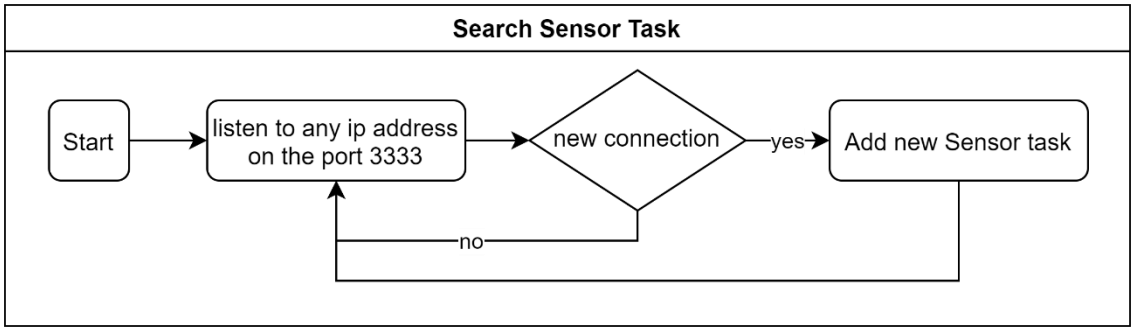


Figure 19 : Tâche de recherche de capteurs

Tâche de capteurs

Finalement, cette tâche essaie de se connecter aux capteurs. Lorsqu'elle réussit à se connecter, l'information des boutons du capteur sera lue. Alors, le sémaphore de l'écriture du contrôle est activé. Ensuite, la tâche lit l'information des capteurs et applique les valeurs de contrôle en fonction des capteurs et envoi les commandes de contrôle aux capteurs.

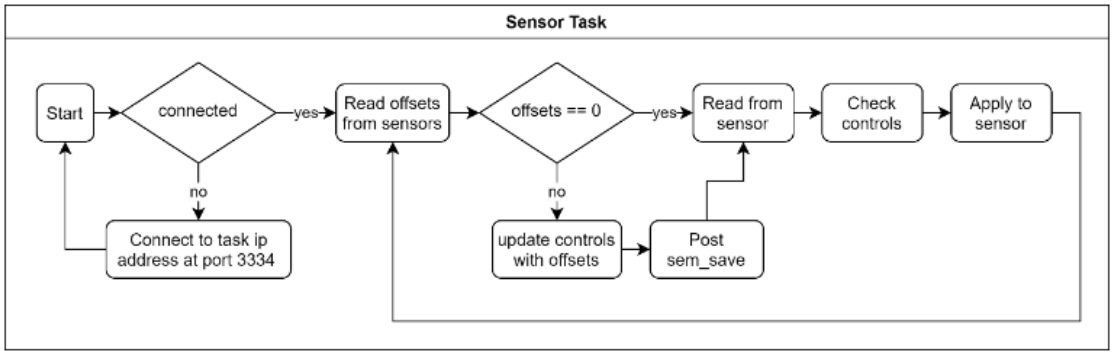


Figure 20 : Tâche de capteurs

Structure globale

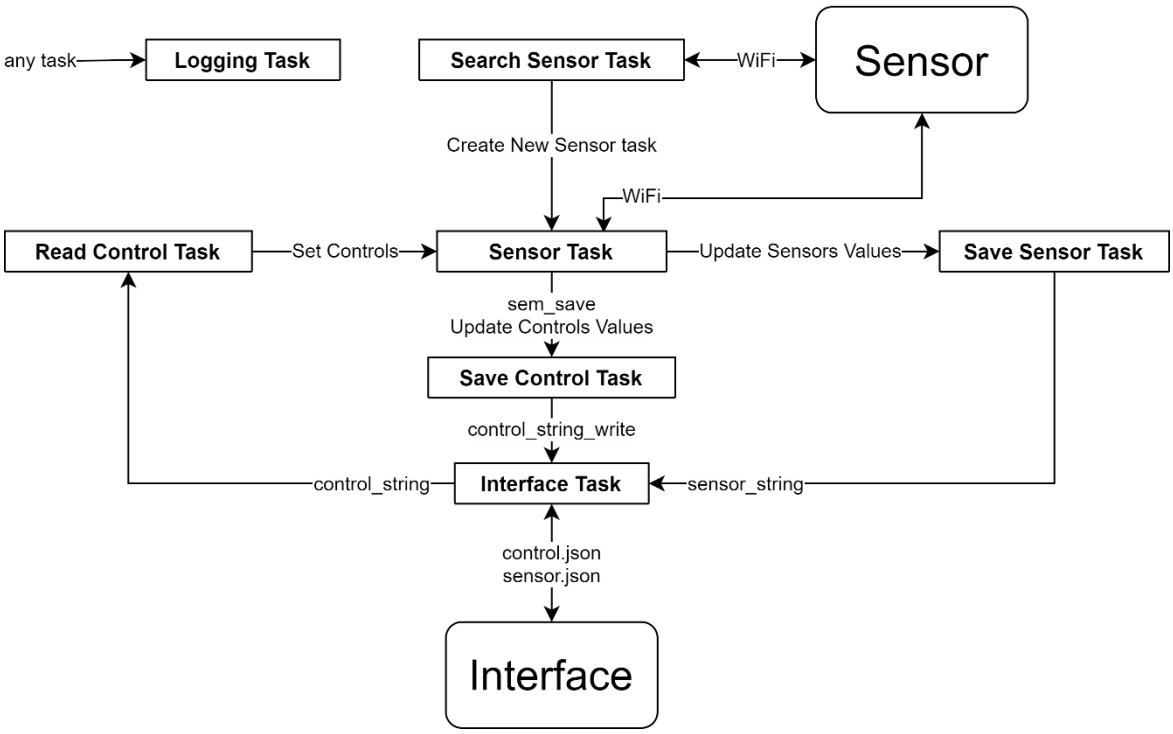


Figure 21 : Structure globale du contrôleur

Utilisation

Une fois que le programme est lancé, l'utilisateur n'a plus rien à faire, le programme va s'exécuter de lui-même et ajouter ses propres capteurs. S'il y a un problème, il est possible de voir les tâches que le programme exécute grâce à la fonction d'enregistrement. Cela permet de garder une trace des opérations exécutées pour identifier les problèmes qui sont survenus.

Capteurs

Compilation

Pour compiler le programme des esp32, nous utilisons l'extension PlatformIO. Il faut installer la plate-forme Espressif 32 dans PlatformIO, ensuite, ouvrir le projet dans l'extension. Le fichier « platformio.ini » configure le reste du projet. Les librairies utiliser sont Arduino_JSON version 0.1.0, Adafruit Unified Sensor version 1.1.4 et DHT sensor library version 1.4.2. Une fois que le projet est bien configuré, dans l'extension, il devrait y avoir une tâche qui apparait nommer « esp32doit-devkit-v1 », il est possible de voir que certaines options en dessous de « general » permettre de compiler avec l'option Build, de téléverser avec l'option Upload et de surveiller le port série avec Monitor. Pour créer de nouvelles tâches dans le ESP32, nous utilisons freeRTOS qui nous permet d'exécuter des tâches en parallèle. Pour téléverser le programme, il faut appuyer sur un des deux boutons physiques du microcontrôleur qui permet d'activer la programmation du ESP32. Le deuxième bouton permet de redémarrer.

Lancement

Pour les modules, une fois le programme compiler et téléverser dans les ESP32 le programme démarre automatiquement une fois que celui-ci est alimenté.

Structure

Le capteur comprend plusieurs tâches séparer qui sont créées grâce à freeRTOS. Pour commencer, la tâche wifi_client essaie de se connecter au contrôleur.

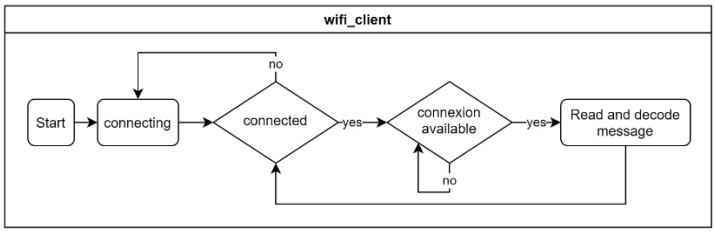


Figure 22 : Tâche client Wi-Fi

Lorsque le capteur est reconnu, le contrôleur va se connecter au « wifi_server », celui-ci va aller chercher les valeurs du DHT11 et des boutons, en plus d'écrire les valeurs des contrôles dans « apply_gpio ». La tâche par elle-même ne fait que décoder les messages entrants et répond les valeurs demandées ou applique les valeurs incluses dans celui-ci.

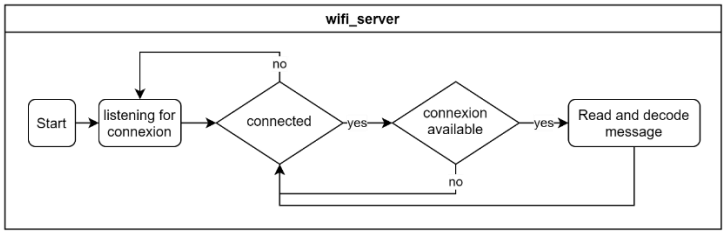


Figure 23 : Tâche serveur Wi-Fi

La tâche « read_temperature » lit le capteur en boucle et met sa valeur dans une variable partagée avec le « wifi server ».

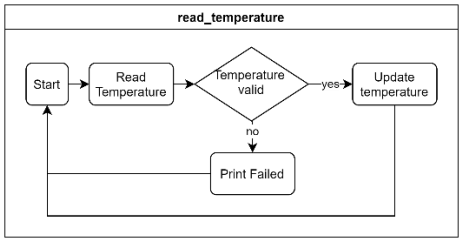


Figure 24 : Tâche lire température

Pour finir, la tâche « apply_gpio » va appliquer la valeur des contrôles aux bonnes sorties de l'ESP32 et regarder les valeurs des boutons actuels.

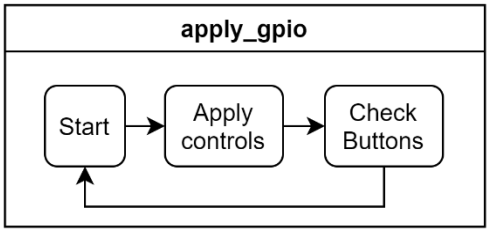


Figure 25 : Tâche appliquer sorties

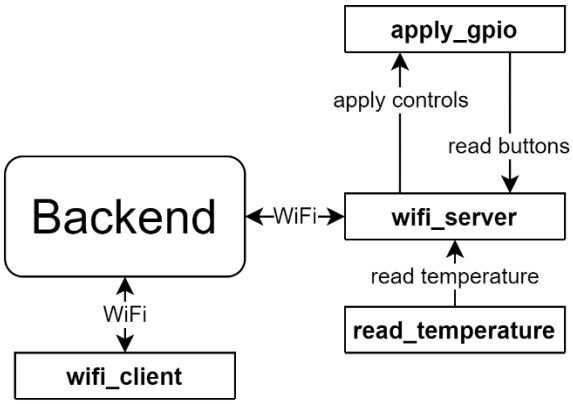


Figure 26 : Structure globale du capteur

Utilisation

Pour l'utilisation des capteurs, lorsque le programme en arrière-plan s'exécute sur le Raspberry Pi, les capteurs devraient se connecter automatiquement et se faire contrôler par l'interface web.

Il est aussi possible d'utiliser les boutons sur la maquette qui sont identifiés en fonction de leur fonctionnalité soit monter la température, baisser la température et changer l'état de la lumière.

Il est aussi possible de se connecter manuellement pour envoyer des commandes aux capteurs. Pour faire cela, il faut un client Telnet et envoyer les commandes du tableau 6. Le terme « TYPE » indique le type de capteur que nous voulons lire, dans notre cas nous avons les types « TEMP » pour température et « LIGHT » pour la lumière.

Les valeurs seront retournées en format texte, donc il est possible de recevoir des nombres entiers et des décimales par la même commande. Pour les contrôles, ils sont exécutés par des relais, donc les états possibles sont binaires. Pour le capteur, il y a un numéro d'identification qui est fixe pour permettre des identifications différentes pour la page web.

Tableau 6 : Table résumée des commandes du capteur

Commande	Fonction
SENSOR:TYPE:GET	Prends la valeur du capteur
CONTROL:TYPE:SET:value	Lis la valeur du contrôle
CONTROL:TYPE:GET	Prends la valeur du contrôle
BUTTON:TYPE:GET	Prends la valeur des boutons
SYSTEM:START	Commence l'acquisition
SYSTEM:RESTART	Redémarre le ESP32
SYSTEM:GET_ID	Prends le numéro du capteur

Maquette

Pour la construction de la maquette, des recherches étaient nécessaires. Les principales informations recherchées étaient lors de la création du circuit électrique de la maquette. Pour commencer, des recherches pour choisir l'élément chauffant ont été réalisées. Cette décision était délicate, car s'il chauffait trop la maquette pouvait prendre feu.

Il a donc fallu réaliser des tests thermiques pour évaluer la puissance dissipée. Ces tests ont permis de prendre la décision d'utiliser une résistance de huit ohms. Avec cette résistance, le système utilise 24 W, ce qui permet de chauffer la pièce à un rythme d'approximativement 0,5 °C/min.

L'alimentation contient des tensions de 3.3, 5 et 12 Volts, les composantes comme les lumières, les relais, la ventilation, les capteurs et le contrôleur fonctionnent avec ces tensions. En nous référant aux fiches techniques des pièces, nous avons confirmé les tensions d'alimentation.

Une autre information à surveiller dans les fiches techniques est le courant utilisé par chacun des composants. Cette information a permis de calculer la capacité du bloc d'alimentation à fournir suffisamment de courant à chaque composante. En plus, cela a défini le diamètre nécessaire pour les fils électriques. Pour ajouter un facteur de sécurité, une valeur plus élevée de diamètre de fils a été employée pour les éléments chauffants.

Pour la ventilation, un ventilateur d'ordinateur fut sélectionné. Les raisons étant qu'elles sont petites et fonctionnent à 12 V. Les ventilateurs choisis ont également une circulation d'air suffisante pour changer l'air de la pièce en moins d'une minute. La métrique associée à ces ventilateurs est de 5.2 CFM. Cette force est nécessaire pour refroidir la pièce et empêcher la stratification de l'aire dans la maquette.

Les relais devaient avoir une tension d'enclenchement à la bobine de 5 V pour être compatibles avec le contrôleur qui les entraîne. Il était également utile qu'ils aient une protection par optocoupleur et qu'il puisse supporter une tension de 12 V DC ainsi qu'un courant de 5A.

LISTE DES PIÈCES ET COÛT

Pièces/Composantes	Coût	Utilité
Bois	Recyclé	Structure de la maquette
Plexiglass	Recyclé	Structure de la maquette
Bloc d'alimentation	Recyclé	Alimentation
Écran LCD	Recyclé	Affichage
Pilote de l'écran	30$	Affichage
Fils électriques	Recyclé	Relier et alimenter les composantes
Boutons	6$	Commande locale
Vis	Recyclé	Fixer la structure
ESP32	27$	Microcontrôleur
Radiateur	Recyclé	Dissiper la chaleur dans la maquette
RPI	100$	Reçois les commandes
DEL	2$	Lumières dans la maquette
Connecteur	4$	Connecter les différentes composantes
Résistance	8$	Réchauffer la pièce
DHT11	10$	Capteur de température
Ventilateur	2$	Refroidis la pièce
Relais	12$	Permets de contrôler chaque composante
Câble d'alimentation	Recyclé	Alimente le PSU
Plaquette trouée	5$	Permets de fixer les connecteurs et les boutons
Breadboard	Recyclé	Permets de relier les fils au microcontrôleur
Support	Recyclé	Permets de supporter l'écran
Câble micro HDMI	10$	Permet de relier l'écran au Raspberry pi
TOTAL	**216$**	

Pour ce projet, plusieurs composantes étaient nécessaires afin de concevoir la maquette. Le coût total de ce prototype est de 216$. Les composantes les plus dispendieuses étaient le Raspberry Pi, le pilote de l'écran et les ESP32. On peut également lire que plusieurs éléments ont été recyclés. Le bois et le Plexiglas ont été récupérés sur d'anciens meubles. Le bloc d'alimentation a été récupéré sur un ordinateur. L'écran LCD provient d'un portable. Les fils, les vis et tous les câbles ont été récupérés sur divers appareils électriques.

RÉSULTATS ET DISCUSSION

Afin d'analyser les résultats et la performance du projet, chacun des objectifs établis lors du cahier des charges sera revu. Dans tous les cas, il est important de rappeler que le but était d'obtenir une preuve de concept sur un système de contrôle domotique. Afin d'économiser des coûts inutiles à la démonstration, l'achat du matériel ne donnant aucune valeur ajoutée au produit fut omis.

Tests de performance du prototype

Pour confirmer les métriques de notre prototype, deux tests ont été effectués. Dans un premier temps, il fallait obtenir le temps de réaction entre l'action sur l'interface et la réaction de la maquette. Dans un deuxième temps, il fallait obtenir un graphique de la variation de la température en fonction du temps.

Test du temps d'acquisition et de réaction de la maquette

Pour faire ce test, nous avons filmé un vidéo ayant en objectif un chronomètre, l'interface et la maquette. Le résultat du vidéo démontre que le temps de réaction est inférieur à 10ms entre l'action sur l'interface et la réaction de la maquette. L'objectif initial de notre projet était d'obtenir un temps de réaction inférieur à 120 secondes. L'objectif est atteint.

Variation de la température en fonction du temps

Pour ce deuxième test, nous démontrons que notre système de chauffage est opérationnel. Nous avons démarré le test à partir de la température ambiante de 24.5 °C. Par la suite, le chauffage a été activé en sélectionnant 26 °C sur la maquette. Afin de mesurer la variation de la température et de confirmer l'exactitude des résultats, nous avons inséré un thermomètre externe dans la maquette. Ci-dessous se retrouvent les résultats de l'expérience.

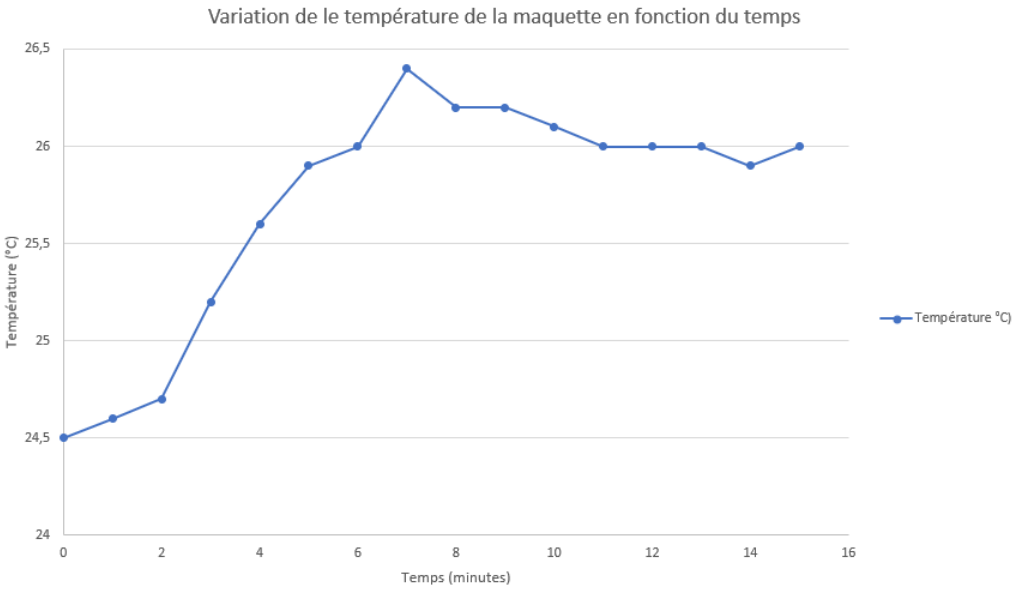


Figure 27 : Variation de la température de la maquette en fonction du temps

Avec la courbe, on aperçoit que la température a correctement augmenté jusqu'à la température sélectionnée, soit 26 °C. Il y a un léger dépassement de la température sélectionnée, car aucun contrôleur de type PID ou à retour d'état pour le réduire a été implanté. De plus, même lorsque le système est à l'arrêt, la résistance continue son dégagement de chaleur jusqu'à avoir épuisé son énergie résiduelle. Finalement, l'erreur en régime permanent de notre système est de 0,1 °C. Notre objectif initial était une erreur en régime permanent sous 0.5 °C.

Voici un tableau présentant les résultats selon chacun de nos objectifs.

Tableau 7 : Résultats des objectifs

Objectifs	Résultats
Contrôler la température dans une pièce de maison, entre 18 et 23 °C	La température de **deux pièces de la maison, et plus au besoin**, peuvent être contrôlés entre **18 et 23 °C**. Dans le cadre du projet, le chauffage est effectué par une simple résistance dans la maquette. De plus, aucun moyen de refroidir la maquette n'a été ajouté puisqu'il n'y aurait eu aucun défi de conception supplémentaire. Aussi, l'objectif du contrôle de la température était déjà maîtrisé. L'ajout d'un climatiseur ou d'un meilleur chauffage aurait tout simplement augmenté les coûts de réalisation du prototype. Sachant qu'en situation de produits commercialisables, le matériel de chauffage et de climatisation dépend de l'installation dans la résidence et non du contrôleur, il est donc possible d'aller au-delà des bornes énoncées dans l'objectif pour le contrôle de la température.
Contrôler l'éclairage dans une pièce de la maison	L'éclairage peut être contrôlé dans **deux pièces, et plus de la maison**. Puisque le prototype est versatile, nous pouvons contrôler l'éclairage d'autant de pièces qu'il y a dans la maison. Il suffit d'ajouter des capteurs dans chaque pièce et d'ajouter ces pièces à l'interface web.
Obtenir un temps maximal de mise à jour de 120 secondes	Le prototype possède un temps de mise à jour inférieur à 10 ms, entre la commande sur l'interface et la réaction sur le produit.
Contrôler l'éclairage et la température par un seul microcontrôleur	Tous les programmes s'exécutent sur **un seul microcontrôleur, soit un Raspberry Pi qui fait office de contrôleur central** pour toutes les pièces dans la maison. Dans notre prototype, le contrôleur contrôle deux pièces.
Accéder et contrôler à distance le système domotique	Le prototype peut être contrôlé à distance à l'aide d'une page web. La page web peut être accédée via tout appareil capable d'opérer un fureteur Internet. Ainsi, l'accès à son système est grandement simplifié, peu importe la plate-forme utilisée. Puisque la page web est active en tout temps sur le contrôleur, le système peut être accédé de n'importe où, sans contrainte de distance.
Communication avec les capteurs de température de manière sans-fil	Les capteurs de température communiquent par Wi-Fi via l'ESP32 avec le contrôleur central.

En bref, tous les objectifs principaux et secondaires ont été atteints. Il est important de comprendre que le prototype n’est qu’une version diluée d’un produit qui pourrait être déployé à grande échelle. Pour des raisons académiques, une maquette avec deux pièces simulées fut utilisée durant les démonstrations. Toutefois, on peut facilement augmenter la quantité de pièces à contrôler par l’interface, puisque peu de modifications au prototype développé sont nécessaires.

CONCLUSION ET RECOMMENDATIONS

Le projet de contrôle domotique avait pour vocation d'offrir une solution à la gestion d'énergie au sein d'une maison. Il devait réaliser cet objectif en effectuant un meilleur contrôle à bas prix et en étant compatible avec les équipements existants. Le prototype devait être capable d'augmenter et réduire la température ainsi qu'allumer et éteindre une lumière, dans une pièce de maison. En plus, il devait être capable de réaliser le tout en 120 secondes et moins. Enfin, le système devait être accessible à distance et localement et communiquer avec les modules d'acquisition de manière sans-fil.

Au cours de notre réalisation, les objectifs principaux et spécifiques ont tous été atteints. En créant le contrôleur à partir d'un Raspberry Pi, celui-ci communique avec un module d'acquisition, qui lui envoie sur demande, les données de température d'une pièce de maison. Ensuite, le contrôleur, suivant la température désirée par l'utilisateur, active en conséquence le chauffage ou le ventilateur afin d'ajuster la température de la pièce via le module d'acquisition. L'utilisateur peut opérer le système localement via des boutons situés sur le module d'acquisition ou via une interface web accessible par tout appareil disposant d'un fureteur de page web. De plus, l'activation d'une lumière peut être réalisée de la même manière que la température.

Le contrôleur communique de manière sans-fil avec le module d'acquisition via une connexion sans-fil, ce qui permet d'avoir une seule unité centrale prenant les décisions sur l'activation des équipements de la maison. Enfin, notre système réalise une mise à jour globale en moins de deux secondes ce qui est largement au-delà de notre objectif initial de 120 secondes. Cette mise à jour permet une interaction fluide avec le système.

De plus, une sécurité a été implantée advenant une coupure d'Internet. Le contrôleur crée son propre réseau local sans-fil. Alors, advenant un arrêt de service Internet, le contrôle pourra toujours être réalisé si l'utilisateur reste à distance de communication sans-fil du contrôleur. Cependant, en tout temps le contrôle pourra être réalisé via les boutons situés sur le module d'acquisition.

Alors, avec les fonctionnalités implantées au cœur de notre projet nous avons rempli tous nos objectifs principaux et spécifiques.

En contrepartie, il est possible d'augmenter l'efficacité du système. Afin d'améliorer la gestion des ressources énergétiques et dans un souci de développement durable, le système pourrait être couplé à de l'intelligence artificielle. Avec l'ajout de cette fonctionnalité, il serait possible de contrôler la température en fonction des habitudes de vie des utilisateurs. Le système intelligent serait capable d'abaisser automatiquement la température lorsque les occupants du domicile sont absents, par exemple. De plus, un système de calcul de coût d'énergie selon l'emplacement géographique pourrait être mis en place. En utilisant le prix du kW/h ainsi que la région des utilisateurs le système pourrait calculer un modèle d'utilisation basé sur le prix du chauffage et de climatisation. Enfin, le système étant relié à Internet pourrait se voir implanter des alertes via message texte directement sur le téléphone d'un utilisateur.

RÉFLEXION SUR LES NOTIONS DE DÉVELOPPEMENT DURABLE EN LIEN AVEC LE PROJET

En tant qu'étudiant et future ingénieur nous serons des acteurs de premier ordre en matière de développement durable. Dans le but de suivre cette ligne directrice, nous avons dirigé notre projet dans ce sens. En réalisant un système de contrôle domotique abordable, notre espérance était de pouvoir fournir un outil en matière de gestion d'énergie. En effet, en installant un système de gestion de lumière et de température, il est attendu que les coûts reliés à l'énergie baissent. Cependant, la gestion des ressources n'est qu'un des aspects lorsqu'on parle de développement durable.

En tant que concepteur de produits nous devons prendre conscience qu'ils ont une espérance de vie et qu'ensuite ils pourront se retrouver dans les décharges électroniques et possiblement contaminer les eaux et les sols. Alors, nous avons décidé en tant qu'équipes que nous devions réaliser notre projet avec le plus de matière recyclée possible. De ce fait est survenue la création d'une maquette en bois, puisque ce matériau est recyclable et non dommageable pour l'environnement. De plus, afin de réduire nos coûts liés au projet et dans le cadre d'une démarche environnementale nous avons été capables d'extraire une majorité de composants requis sur des machines en fin de vie. En effet, en démontant un ordinateur, une unité de chauffage, un éclairage et une télé nous avons extrait la majorité des pièces requises pour accomplir le projet.

De plus, les matériaux utilisés pour construire la maquette simulant une maison proviennent de matériaux de construction recyclés. En retaillant les matériaux, nous avons réussi à obtenir les dimensions désirées.

Essentiellement, les achats réalisés pour le projet furent les microcontrôleurs (ESP32), le contrôleur (Raspberry Pi) et les capteurs de température (DHT11). Les choix réalisés pour l'achat des pièces ont été faits avec la notion d'écoconception à l'esprit. Afin d'obtenir des appareils moins dommageables pour l'environnement, nous avons opté pour des produits contenant la certification ROHS. Cette norme « limite l'utilisation du plomb et autres substance potentiellement dangereuses dont le cadmium, le mercure et le chrome VI, PBB, PBDE, dans les produits électriques et électroniques » (SGS, s.d.). Toutes ces substances dommageables sont contrôlées au sein des produits portant la certification ROHS.

Enfin, lors de la création de notre système nous avions à l'esprit la fin de vie de notre produit. Nous avons l'objectif de participer à l'Urban Mining (Perrotta, 2017) en apportant toutes les composantes électroniques à une compagnie pratiquant cette méthode afin de recycler les métaux précieux contenus dans ces produits. Ce nouveau type de minage est prometteur, car il permet de réduire l'excavation de grand site à ciel ouvert et de recycler les appareils électroniques en circulation. Le recours à cette technique a permis aux Jeux olympiques de Tokyo d'amasser suffisamment de déchets électroniques pour confectionner les médailles remises aux vainqueurs.

BIBLIOGRAPHIE

Amazon Web Services, Inc. (2021). *FreeRTOS*. Récupéré sur FreeRTOS Kernel Developer Docs: https://www.freertos.org/features.html

Code Pen. (2021). Récupéré sur Code Pen: https://codepen.io/

David Lord, T. M. (2021). Récupéré sur Flask web development, one drop at a time: https://flask.palletsprojects.com/en/2.0.x/

DHT11-Temperature and Hmidity Sensor. (2021, Juillet). Récupéré sur Components101: https://components101.com/sensors/dht11-temperature-sensor

ESP-32 Datasheet. (s.d.). Récupéré sur espressif: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Garrels, M. (2008). *Bash Guide for Beginners*. Récupéré sur The Linux Documentation Project: https://tldp.org/LDP/Bash-Beginners-Guide/html/

Handson Technology. (2018). *8 Channel 5V Optical Isolated Relay Module*. Récupéré sur https://handsontec.com/dataspecs/module/8Ch-relay.pdf

Hawicz, E. (2021). *JSON-C - A JSON implementation in C*. Récupéré sur GitHub: https://github.com/json-c/json-c/wiki

Member #23999. (s.d.). *Pull-up Resistors*. Récupéré sur Sparkfun: https://learn.sparkfun.com/tutorials/pull-up-resistors/all

Perrotta, T. (2017, Mars 2). *What is Urban Mining?* Récupéré sur Greentec: https://greentec.com/what-is-urban-mining/

SGS. (s.d.). *Biens de consommation et distribution*. Récupéré sur SGS: https://www.sgs.ca/fr-fr/consumer-goods-retail/electrical-and-electronics-total-solution-services/audio-video-and-household-appliances/rohs

Systems, E. (2016). *ESP-IDF Programming Guide*. Récupéré sur https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

The Open Group. (1997). *pthread.h*. Récupéré sur The Single UNIX ® Specification, Version 2: https://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread.h.html

The Open Group. (1997). *sys/socket.h*. Récupéré sur The Single UNIX ® Specification, Version 2: https://pubs.opengroup.org/onlinepubs/7908799/xns/syssocket.h.html

The Pi. (2021). *How to use your Raspberry Pi as a wireless access point.* Récupéré sur The Pi: https://thepi.io/how-to-use-your-raspberry-pi-as-a-wireless-access-point/

tutorialspoint. (2021). *C - File I/O*. Récupéré sur tutorialspoint: https://www.tutorialspoint.com/cprogramming/c_file_io.htm

W3Schools. (2021). *w3schools*. Récupéré sur How TO - Range Sliders: https://www.w3schools.com/howto/howto_js_rangeslider.asp

W3Schools. (2021). *W3Schools*. Récupéré sur How TO - Toggle Switch: https://www.w3schools.com/howto/howto_css_switch.asp

W3Schools. (2021). *W3Schools*. Récupéré sur How TO - Top Navigation: https://www.w3schools.com/howto/howto_js_topnav.asp

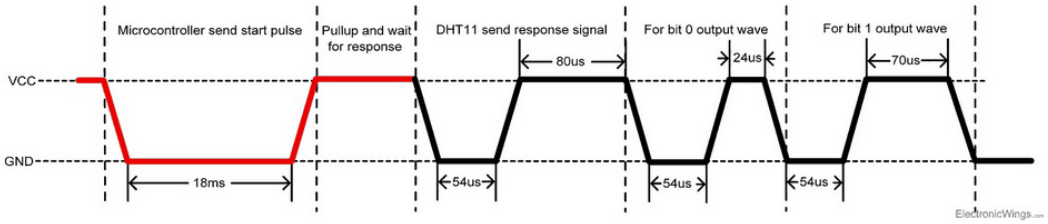
W3Schools. (2021). *W3Schools*. Récupéré sur How TO - Four Column Layout: https://www.w3schools.com/howto/howto_css_four_columns.asp

ANNEXE 1 : FONCTIONNEMENT DU DHT11

Fonctionnement du capteur de température DHT11

Communication with Microcontroller

- DHT11 uses only one wire for communication. The voltage levels with certain time value defines the logic one or logic zero on this pin.
- The communication process is divided in three steps, first is to send request to DHT11 sensor then sensor will send response pulse and then it starts sending data of total 40 bits to the microcontroller.



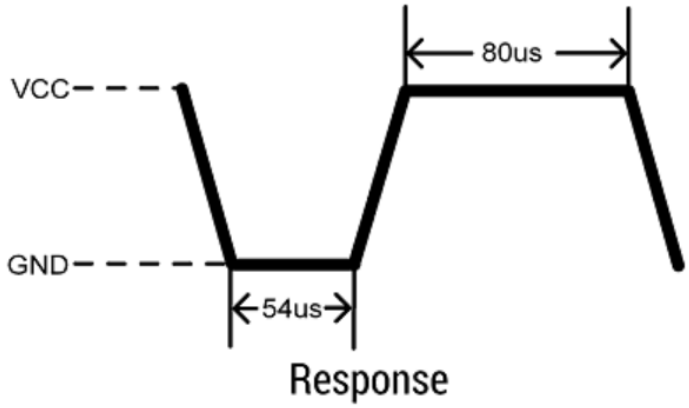
Communication process

Start pulse (Request)

Start Pulse

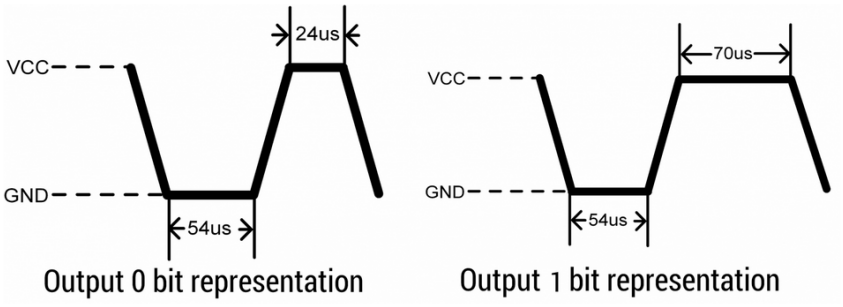
- To start communication with DHT11, first we should send the start pulse to the DHT11 sensor.
- To provide start pulse, pull down (low) the data pin minimum 18ms and then pull up, as shown in diag.

Response



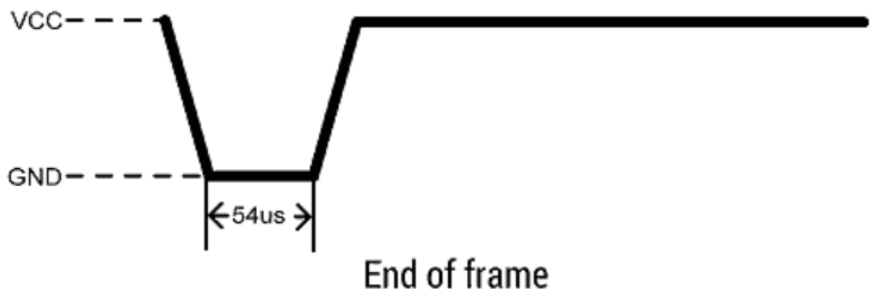
- After getting start pulse from, DHT11 sensor sends the response pulse which indicates that DHT11 received start pulse.
- The response pulse is low for 54us and then goes high for 80us.

Data



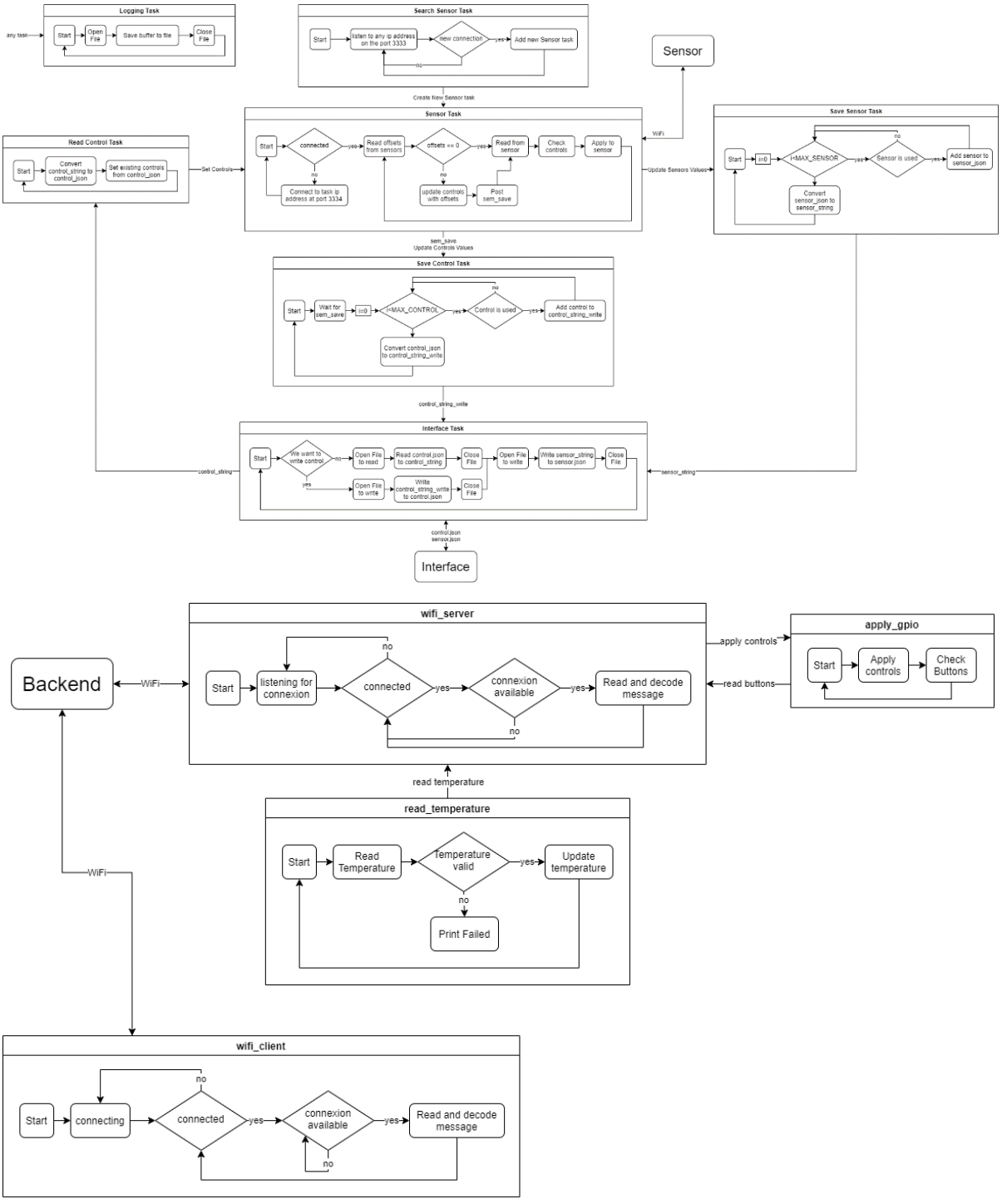
- After sending the response pulse, DHT11 sensor sends the data, which contains humidity and temperature value along with checksum.
- The data frame is of total 40 bits long, it contains 5 segments (byte) and each segment is 8-bit long.
- In these 5 segments, first two segments contain humidity value in decimal integer form. This value gives us Relative Percentage Humidity. 1st 8-bits are integer part and next 8 bits are fractional part.
- Next two segments contain temperature value in decimal integer form. This value gives us temperature in Celsius form.
- Last segment is the checksum which holds checksum of first four segments.
- Here checksum byte is direct addition of humidity and temperature value. And we can verify it, whether it is same as checksum value or not. If it is not equal, then there is some error in the received data.
- Once data received, DHT11 pin goes in low power consumption mode till next start pulse.

End of frame



- After sending 40-bit data, DHT11 sensor sends 54us low level and then goes high. After this DHT11 goes in sleep mode.

ANNEXE 2 : STRUCTURE DU LOGICIEL



ANNEXE 3 : BROCHES ESP32