Rapport du projet pratique

Circuits et machines électriques pour ingénieurs en mécanique

ELG3736 Automne 2023

Soumis par équipe A10

Ihab Achargui Afkir , 300217474

Prayot Sae-Khow, 300246953

Faculté de Génie de l'Université d'Ottawa

Date : 08-12-2023

Table des matieres

I) Introduction 4
II) Description détaillée du projet 5
A) Systeme de detection de place libre 5
1. Sous systeme d'occupation de place 5
B) Système de concentration et d'antivol 7
1. Sous-système d'accès à la boîte par carte étudiante 8
2. Sous-système de verrouillage et deverrouillage motorisé 8
C) Système de chronométrage de type Pomodoro 8
III) Liste des composantes matérielles et logicielles du projet 9
A) Composantes materielles 9
1) Systeme de detection de place libre 9
2) Système de concentration et d'antivol 11
B) Logicielles: 13
a) Arduino IDE 13
b) Serveur arduino : 13
c) Visual-Studio : 13
IV) Diagrammes, schémas et circuits électriques décrivant les différentes parties matérielles (Hardware) du projet. 14
V) Organigrammes et algorithmes décrivant les différentes parties logicielles (Software) du projet. 18
A)Microcontroleurs: 19
B)Programmation : 20
1. Ecran LCD avec timer pomorodoro: 20
a. Déclaration des variables et des constantes : 20
b. Déclaration des broches matérielles : 21
c. Configuration initiale dans la fonction setup() : 21
d. Boucle principale dans la fonction loop() : 21
i. Réinitialisation du compteur : 21
ii. Boucle principale (While) : 22
iii. Session d'étude : 22
iv. Pause : 23
v. Répétition : 23
2. Boîte de verrouillage : 24
a. Inclusions de Bibliothèques : 24
b. Initialisation des Objets et des Variables : 24
c. Configuration Initiale (setup) : 25
d. Boucle Principale (loop) : 25
e. Fonction de Vérification de l'Accès (checkAccess) : 26
3) Systeme de detection de place et d'environement de travail: 27
VI) Communication entre les différentes composantes du projet : 31
1. Fils reliant chaque composant : 31
2. Communication WiFi entre ESP32 et le serveur : 31

3. Écran d'affichage : 32
4. Interface IoT : 32
VII) Résolution de problèmes, maintenance: 32
VIII) Difficultés éventuelles. 33
Conclusion **34**
Annexe **35**

I) Introduction

Dans le cadre de notre cursus sur les circuits et les machines électroniques destiné aux ingénieurs mécaniques, notre équipe se consacre à la conception d'un système automatisé de détection des places libres sur le campus. Ce projet vise à simplifier la recherche d'emplacements propices à l'étude, offrant ainsi aux étudiants un moyen efficace d'économiser du temps et de l'énergie dans leur quotidien académique.

Notre système ne se contente pas uniquement de repérer les places disponibles, mais intègre également une fonctionnalité novatrice évaluant les conditions environnementales optimales pour l'étude. Trois facteurs clés sont pris en compte : le niveau de bruit ambiant, la température et la présence d'autres individus dans la zone d'étude. Cette approche permet aux étudiants de choisir des espaces propices à la concentration et à la productivité.

En résumé, notre projet a pour objectif de simplifier la quête de places libres tout en favorisant un environnement de travail idéal pour les étudiants, prenant en considération les nuisances sonores, la température et la présence d'autres personnes dans la zone. Cette solution promet d'apporter un gain significatif en termes de temps et d'efficacité pour les études sur le campus.

Afin d'optimiser davantage les conditions d'études et de mettre l'accent sur la concentration des étudiants, notre projet propose deux autres systèmes complémentaires : un dispositif de limitation des distractions et un système d'antivol et de chronométrage du temps d'études. L'objectif ultime est d'avertir l'utilisateur à l'avance sur les conditions de travail, évitant ainsi tout déplacement inutile.

Le projet repose sur un système IoT (Internet des objets), regroupant des dispositifs physiques dotés de capteurs et de logiciels, conçus pour une connectivité avec d'autres terminaux et systèmes via Internet. Cette architecture permet une communication en temps réel des données relatives à l'occupation des bureaux et à l'environnement sonore et thermique.

II) Description détaillée du projet

Notre projet se décompose en trois systèmes distincts, chacun soigneusement conçu pour répondre à des besoins spécifiques au sein de l'écosystème de détection des places libres et d'optimisation de l'environnement d'étude. Dans cette section, nous allons examiner en détail chaque sous-système, en fournissant une description exhaustive de ses composants soigneusement sélectionnés.

A) Systeme de detection de place libre

1. Sous systeme d'occupation de place

Ce sous-système est dédié à l'identification précise des places libres sur le campus. Il repose sur des capteurs de présence intelligents qui scrutent en temps réel l'occupation des espaces d'étude.

Des algorithmes de traitement de données sophistiqués sont implémentés pour interpréter les informations collectées et fournir une cartographie actualisée des emplacements disponibles.

L'une des complexité serait la prise en compte des événements parasitaires qui pourrait biaiser la fiabilité de notre programme, on parle ainsi de la gestion des exceptions. Lorsque nous pensons au système de détection de place libre, la première difficulté serait de faire comprendre au capteur que la personne est bien assise à l'endroit et non pas de passage , que cette même personne est toujours présente malgré le fait qu'elle soit juste distraite ou plus dans le champ de détection.Nous comprenons alors qu'un capteur de mouvement n'est pas suffisant en termes de précision et d'acuité pour une détermination de présence humaine dans le bureau. Nous avons donc eu l'idée d'introduire un capteur ultrason permettant de mesurer la distance entre la personne et le bureau , cette détection est dans notre système comme une boucle de retour permettant de confirmer constamment la présence. On passe de système d'intrusion à un système d'occupation qui sera plus stable mais d'autre part complexifie notre partie programmation.

La compréhension du fonctionnement électronique des capteurs et actionneurs est importante car elle permet de relever les defis liés au exeptions et erreurs de captures, passons sur l'ensemble de nos capteurs et du potentiel risque de défaillance.

Dans le cadre du sous-système de détection, nous avons implémenté une combinaison de capteurs de mouvement PIR (Passive Infrared), ultrasoniques, ainsi que des dispositifs ESP32 pour garantir une couverture complète des espaces d'étude.

- Capteurs de Mouvement PIR :

Les capteurs PIR utilisent des lentilles spéciales pour détecter les changements infrarouges causés par les mouvements d'une personne. Chacun est positionné de manière stratégique pour couvrir des zones spécifiques, assurant une détection fiable des mouvements dans l'espace.

- ❖ Capteurs Ultrasoniques :

Les capteurs ultrasoniques émettent des signaux sonores et mesurent le temps nécessaire pour que ces signaux rebondissent sur les surfaces, calculant ainsi la distance. Ces capteurs sont intégrés pour détecter la présence d'objets dans des zones où les capteurs PIR pourraient présenter des limitations.

- ❖ Dispositifs ESP32 :

Les microcontrôleurs ESP32 agissent comme des nœuds centraux de traitement des données provenant des capteurs. Leur puissance de calcul et leurs capacités de communication sans fil permettent une gestion efficace des informations collectées et le calcul numérique de nos données analogique ou digital.On comprend que l'esp32 ou encore l'arduino contiennent des transistors et multiplexeur et CAN , contribuant ainsi à la génération dynamique de la carte des places libres.

2. **Sous systeme d'Évaluation Environnementale :**

Concernant le sous-système d'évaluation environnementale, nous avons opté pour des capteurs de température DHT11 et des capteurs sonores exploitant un microphone en lien avec un potentiomètre pour une sensibilité ajustable.

- ❖ Capteur de Température DHT11 :

Le capteur DHT11 mesure la température ambiante avec précision néanmoins il faut prendre en compte un temps de stabilisation à froid ainsi que les simples perturbations comme le vent ou la présence de source de chaleur proche. Les données de température sont cruciales pour aider les étudiants à choisir des endroits confortables en fonction de leurs préférences thermiques.

- ❖ Capteur Sonore avec Potentiomètre :

Le capteur sonore, constitué d'un microphone, est équipé d'un potentiomètre qui permet d'ajuster la sensibilité aux niveaux sonores. Pour déterminer les seuils de bruits, nous avons opté pour une connexion analogique. Plutôt que d'investir dans un coûteux décibel mètre, nous avons adopté une approche économique en calibrant notre capteur sonore dans une salle bruyante.

Le code intégré au capteur sonore analyse ensuite les valeurs analogiques captées, les comparant aux données préalablement enregistrées lors de la calibration. Cette méthode permet d'établir des classifications en temps réel, indiquant si l'environnement est bruyant, modéré ou calme, offrant ainsi une évaluation précise de l'ambiance sonore. Cette fonctionnalité s'avère être un élément essentiel pour créer un espace propice à la concentration des étudiants, car elle contribue à orienter les choix d'emplacement en fonction du niveau de tranquillité recherché.

B) Système de concentration et d’antivol

L'une des principales sources de perturbation au cours d'une séance d'étude réside dans la présence des téléphones portables. La proximité de ces dispositifs dans l'espace d'étude accroît les risques de procrastination. Dans le but de remédier à cette problématique, nous avons notre deuxieme système

Notre proposition se matérialise sous la forme d'une boîte à accès contrôlé, dont l'ouverture et la fermeture dépendent exclusivement de la carte d'étudiant de l'utilisateur. Cette conception permet à l'étudiant de déposer son téléphone portable ainsi que tout autre objet de valeur susceptible de constituer une source de distraction. En instaurant cette barrière physique, le système stimule la concentration en restreignant l'accès aux distractions potentielles pendant les sessions d'étude, favorisant ainsi un environnement propice à la productivité.

De plus, cette boîte se révèle également être un dispositif antivol. Lorsque nous nous éloignons momentanément du lieu d'étude, que ce soit pour une pause aux toilettes ou pour prendre l'air, nos objets de valeur demeurent souvent exposés à la vue de tous, ce qui engendre un risque de perte. Ainsi, notre boîte sécurise efficacement nos biens pendant les périodes où nous ne sommes pas présents sur place.

Voici la liste des composantes que nous avons utilise pour la realisation de ce deuxieme systeme.:

- Nous avons d’abord la partie physique qui est le boitier. Nous avons concu notre boitier en MDF et nous avons colle des chanieres permettant l’ouverture et la fermeture
- Pour la partie hardware:
  - Nous avons une arduino uno classique fourni dans tout les kits de base. Nous utilisons ce type de calculateur car nous avons besoins d’une iteration simple.
  - Nous avons incorporé un capteur RFID de type **RC522** pour lire les cartes RFID (identification par radiofréquence). Nous avons choisi ce capteur parce que nous prévoyons que chaque étudiant de l'Université d'Ottawa utilisera ce boîtier individuellement. Ainsi, le système peut accéder à la base de données de l'université pour obtenir le numéro de série de chaque carte. Cette approche permet une personnalisation du système tout en facilitant la gestion des utilisateurs du boitier..
  - Nous utilisons un **servo-moteur** pour le verrouillage et le déverrouillage du boîtier. Nous avons choisi ce type de moteur en raison de notre nécessité d'un

moteur capable de s'arrêter à des positions précises. En outre, la sécurité offerte par ce type de moteur est adéquate pour notre application spécifique.

1. **Sous-système d'accès à la boîte par carte étudiante**

Pour accéder à la boîte, l'étudiant doit présenter sa carte d'étudiant à un emplacement spécifique sur la boîte. Lorsque la carte est reconnue, le système informe l'utilisateur par le biais d'une lumière bleue.

En cas de non-reconnaissance de la carte par le système, une lumière rouge s'allume pour signaler une erreur. Toutefois, si le système échoue à reconnaître la carte de l'utilisateur en question, nous avons envisagé l'ajout d'un système manuel supplémentaire. Celui-ci permettrait à l'utilisateur d'entrer les quatre derniers chiffres de son numéro d'étudiant pour déverrouiller la boîte.

2. **Sous-système de verrouillage et deverrouillage motorisé**

Pour sécuriser et ouvrir le boîtier, nous utilisons un servo-moteur, comme mentionné précédemment. Ce moteur est situé à l'extérieur du boîtier et adopte une position particulière, comme illustré dans la figure 1, lorsqu'il est verrouillé. Lorsque la carte est scannée, le moteur pivote vers une autre position, déverrouillant ainsi le boîtier, comme indiqué dans la figure 2.

Nous envisageons d'améliorer ce système de verrouillage et déverrouillage, car le positionnement actuel du moteur à l'extérieur pourrait poser des problèmes de sécurité. Nous pensons à déplacer le moteur à l'intérieur du boîtier et le relier à une serrure interne pour renforcer la sécurité globale du dispositif.

C) Système de chronométrage de type Pomodoro

La méthode Pomodoro est une technique de gestion du temps simple et efficace. Elle consiste à diviser le travail en périodes de 25 minutes appelées "Pomodori", suivies d'une pause de 5 minutes. Après avoir complété quatre Pomodori, il est recommandé de prendre une pause plus longue, d'environ 15 à 30 minutes.

L'idée est de se concentrer pleinement sur une tâche pendant la période de 25 minutes, puis de prendre une courte pause pour se détendre. Cela aide à maintenir la productivité en évitant la fatigue mentale et en favorisant une meilleure gestion du temps. L'utilisation d'une minuterie est courante pour suivre ces intervalles de temps.

Dans cette optique, nous avons développé un système de minuterie conforme à la méthode Pomodoro. L'étudiant déclenche le mécanisme en appuyant sur un bouton, déclenchant ainsi l'affichage du temps écoulé. Une lumière verte indique la période d'étude de 25 minutes, tandis qu'une lumière rouge s'allume pendant les 5 minutes de pause, signalant à l'utilisateur de faire une pause.
Voici la liste des composantes electroniques que nous avons utilises pour la realisation de ce systeme:

- Nous avons une arduino uno classique fourni dans tout les kits de base. Nous utilisons ce type de calculateur car nous avons besoins d'une iteration simple. Comme pour le cas du systeme **B**
- Nous avons intégré un écran LCD1602 pour afficher du texte ainsi que le minuteur. Ce choix a été motivé par notre désir d'offrir une expérience plus agréable à la lecture et à l'observation. Nous avons relié cet écran à un potentiomètre afin d'ajuster le contraste de l'écran.
- Nous avons un bouton standard pour démarrer le système.

III) Liste des composantes matérielles et logicielles du projet

A) Composantes materielles

Dans cette section, nous allons examiner en détail chaque composant, accompagné de sa fiche technique respective. Nous mettrons en lumière le choix de certains capteurs par rapport à d'autres, en fonction de nos critères de conception spécifiques.

1) Systeme de detection de place libre

- **HC-SR501 PIR sensor**

Specification	**Valeur**
Voltage d'entree (V) DC	4V-12V
Distance (m)	7 metre
Angle de detection	120 degre
Temperature de fonctionnement (Celsius)	-20 a 80 degre celsius

En analysant les spécifications du datasheet, notre attention s'est focalisée sur trois paramètres cruciaux : la distance de détection, l'angle de détection et la plage de température de fonctionnement. Pour adapter le capteur à nos exigences spécifiques, nous avons procédé à une réduction de sa sensibilité, le calibrant ainsi pour une distance de détection optimale de 2 mètres. L'alimentation requise a été fixée à 5 volts, et l'angle de détection a été précisément ajusté à 120 degrés. Ainsi le choix de ce capteur nous convient, car il est ajustable dans les plages souhaitees, il est compacte et facile a brancher.

Datasheet PIR sensor

❖ Capteur à Ultrasons HC-SR04

Specification	**Valeur**
Voltage d'entree (V) DC	5 V
Fréquence de travail	40Hz
Distance maximale de détection	4 m
Distance minimale de détection	2cm
Dimension	45*20*15mm

Suite à une analyse des spécifications, il ressort que ce capteur répond de manière optimale à nos besoins spécifiques. Nous avons besoins d'un composant compact, économe en énergie et facilement ajustable. Dans le contexte de notre espace de travail, notre objectif était que le capteur détecte la présence d'une personne assise dans une plage allant de 0 à 30 centimètres. Ce capteur satisfait ce critère essentiel.

Datasheet ultrasound sensor

❖ Capteur de Température DHT11

Specification	**Valeur**
Voltage d'entree (V)	5 V DC
Plage de mesure	0 a 50 degre celcius
Condition de mesure	25 degre celius
Precision	+ou- 1 degre

Suite à une analyse approfondie des spécifications du capteur de température, nous avons identifié certaines caractéristiques clés essentielles à notre contexte spécifique. Étant donné que

notre environnement de travail présente une faible humidité et des températures avoisinant les 20 degrés Celsius, nous n'avons pas besoin d'une précision extrême en matière de mesure de température.

Dans ces conditions, le choix de ce capteur s'avère convenable. Sa plage de précision convient parfaitement à nos besoins, offrant une mesure adéquate sans sacrifier la pertinence dans un environnement relativement stable sur le plan thermique. La sélection de ce capteur répond donc de manière appropriée aux exigences spécifiques de notre application.
Datasheet temperature and humidity senso

- **Capteur de son KY 038**

Specification	**Valeur**
Voltage d'alimentation (V)	5 V DC
Gamme de fréquences du microphone:	100 à 10000 Hz
Sensibilité du microphone:	46 ± 2.0, (0 dB = 1 V/ Pa) à 1 kHz
Temperature de fonctionnement (Celsius)	0°C à 70°C

À la suite de notre analyse approfondie de ce capteur, nous avons relevé plusieurs spécifications clés. Dans notre application particulière, la mesure du niveau sonore d'un environnement est essentielle. Ce milieu spécifique présente une plage relativement restreinte de décibels. En conséquence, le capteur sonore que nous avons choisi s'adapte parfaitement à nos besoins.

Nous avons soigneusement ajusté la sensibilité du microphone dans un environnement bruyant pour servir de référence. De cette manière, lorsque le microphone est positionné dans un milieu présentant un niveau sonore similaire, il est en mesure de nous fournir des informations pertinentes et précises. Cette adaptation fine du capteur renforce sa capacité à offrir des données significatives dans des conditions acoustiques spécifiques.
Datasheet Souhttps://www.handsontec.com/dataspecs/RC522.pdfnd sensor

2) **Système de concentration et d'antivol**

- **RC522 RFID**

Specification	Valeur
Operating Voltage	2.5V~3.3V
Operating Frequency	13.56MHz
Distance de fonctionnement	50 mm
Vitesse de transfert élevée	848 KBd

D’apres les specification presents, le capteur RFID est parfait pour notre application. Il fonctionne dans une plage de tension de 2.5V à 3.3V avec une consommation de courant de 13~26mA en mode opération et 10~13mA en mode veille, assurant une efficacité énergétique. La fréquence de fonctionnement de 13.56MHz assure la compatibilité avec les cartes étudiantes, tandis que la prise en charge de transferts à haute vitesse jusqu'à 848 KBd garantit une lecture rapide des données. Avec des interfaces polyvalentes telles que SPI, I2C, et RS232, le capteur offre une flexibilité d'intégration. Il est également compatible avec les normes MIFARE et ISO 14443A, assurant une lecture efficace des cartes. Enfin, avec une distance de fonctionnement typique de 50 mm, ce capteur répond aux exigences de la portée opérationnelle pour le scannage des cartes étudiantes.

Datasheet RFID sensor

- **Servo moteur SG90**

Specification	Valeur
Voltage d’alimentation (V)	4.8-6 V
Torque	2.5 Kg-cm
poid	14.7 g

Voici les spécifications du servo-moteur que nous avons sélectionné. Notre choix s'est porté sur ce moteur en raison de notre besoin spécifique : il doit être capable de tourner et de s'arrêter à un moment précis pour verrouiller le boîtier. Ainsi, nous recherchions un moteur compact et léger pour répondre à ces critères.

B) Logicielles:

a) Arduino IDE

L’arduino est une interface IDE simple permettant de coder et mettre des rapport logique entre les signaux , ceci est grâce aux multiplexeurs présents dans nos cartes arduinos, et aux transistors.Le signal est d'abord converti en nombre binaire puis convertit en langage C, c’est la partie compilation puis finalement on transfert le code et la logique sur l’arduino donc nous passons de langage C au langage machine.

b) Serveur arduino :

Dans le cadre de la mise en place d'un serveur web asynchrone sur notre ESP32, nous avons opté pour l'utilisation de la bibliothèque ESPAsyncWebServer. Cette solution offre plusieurs avantages, notamment la capacité à gérer simultanément plusieurs connexions et à traiter d'autres requêtes pendant que le serveur s'occupe d'envoyer une réponse en arrière-plan.

La page web résultante présente des informations telles que la température et l'humidité, avec une mise en forme attrayante grâce à l'utilisation d'icônes et de styles CSS. Pour garantir la réactivité de la page, une balise meta est utilisée, et les icônes sont chargées depuis un site externe via une balise de lien.

Dans la partie du corps HTML, des balises h2 sont employées pour l'en-tête, tandis que des balises p sont utilisées pour afficher la température et l'humidité, accompagnées d'icônes correspondantes. Le contenu HTML est stocké dans une variable pour une gestion plus aisée.

Enfin, afin d'assurer des mises à jour automatiques des données affichées, des scripts JavaScript ont été intégrés. Ces scripts utilisent la fonction setInterval() pour déclencher des requêtes asynchrones XMLHttpRequest toutes les 10 secondes. Ces requêtes sont dirigées vers le serveur pour obtenir les dernières données de température et d'humidité, assurant ainsi une mise à jour en temps réel de la page web sans nécessiter de rechargement.

c) Visual-Studio :

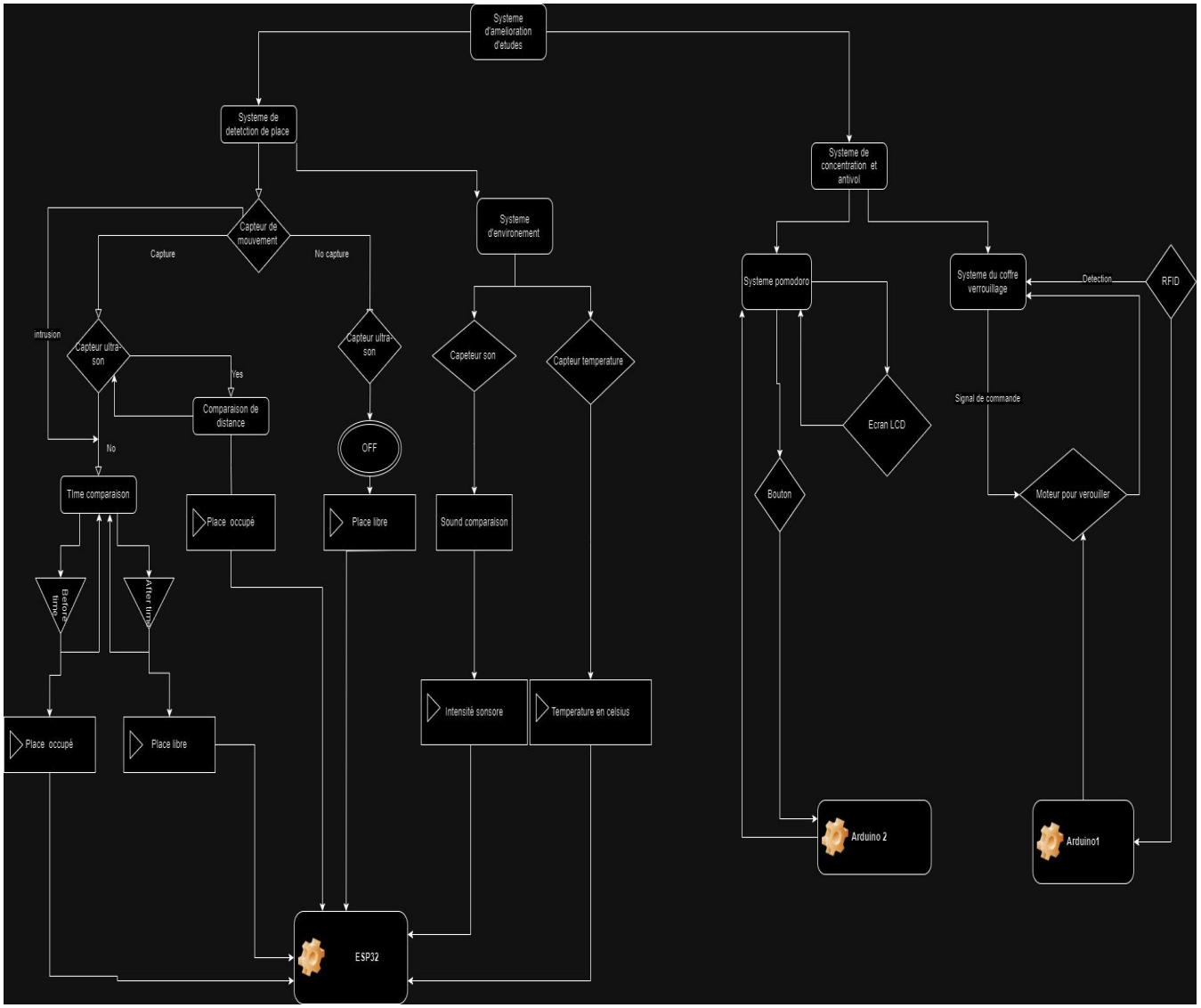
Dans le cadre de notre développement sur Visual Studio, nous avons créé un nouveau projet C# et élaboré l'interface utilisateur en utilisant les composants graphiques disponibles. Nous avons intégré des contrôles tels que TextBox et Label pour afficher les données d'un capteur.

La gestion des composants, notamment l'utilisation de la classe SerialPort pour établir une connexion série avec le capteur, a été mise en œuvre lors du chargement du formulaire. Nous

avons défini des événements, tels que TextChanged et Click, pour réagir aux interactions de l'utilisateur.

Nous avons également utilisé le composant Timer pour lire périodiquement les données du port série, que nous avons ensuite traitées et affichées dans les contrôles appropriés. Des mécanismes de gestion d'erreur ont été mis en place pour capturer d'éventuelles exceptions. Des corrections ont été apportées, notamment l'ajout de définitions manquantes et la résolution d'erreurs telles que les références ambiguës à la classe Timer.

IV) Diagrammes, schémas et circuits électriques décrivant les différentes parties matérielles (Hardware) du projet.



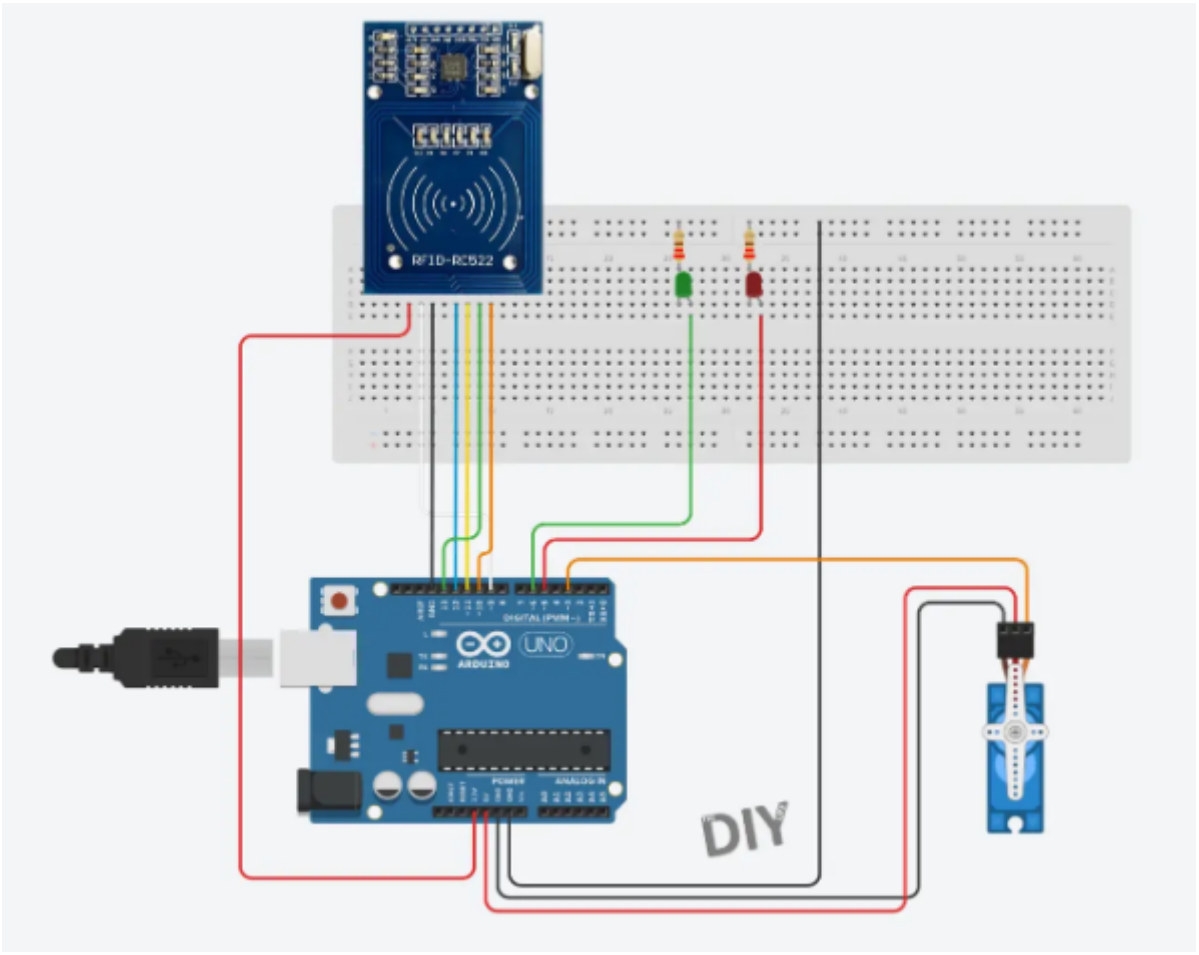


Figure : Circuit Boitier antivol

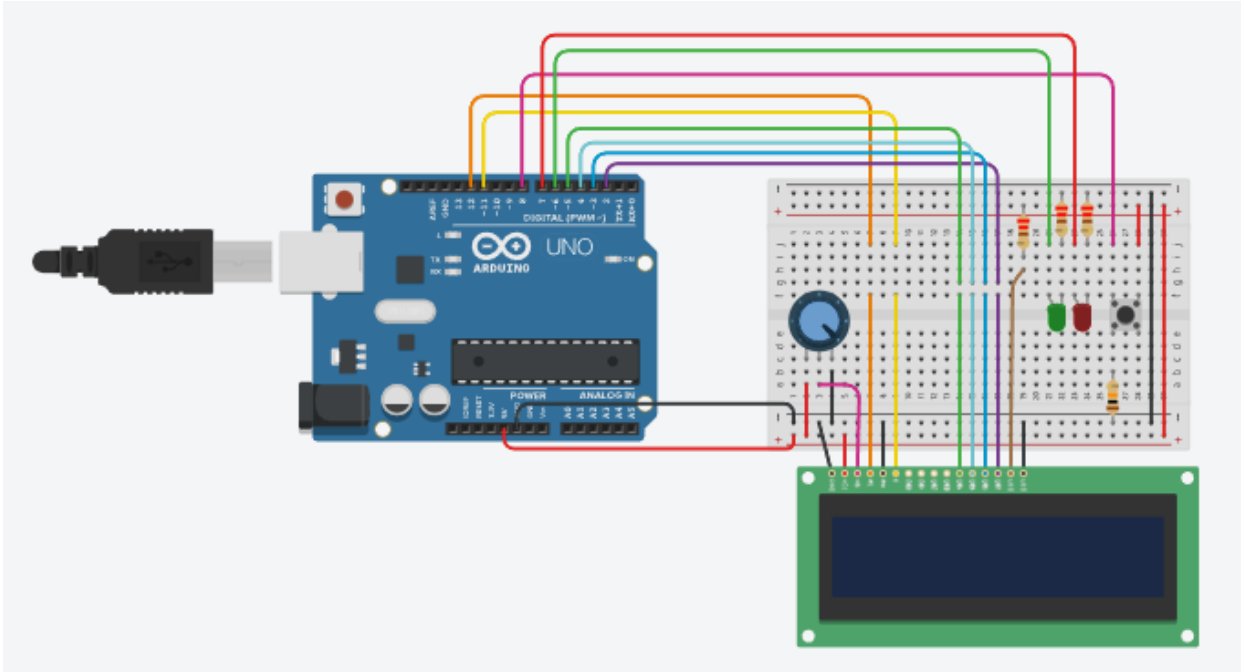


Figure : Circuit boitier pomodoro

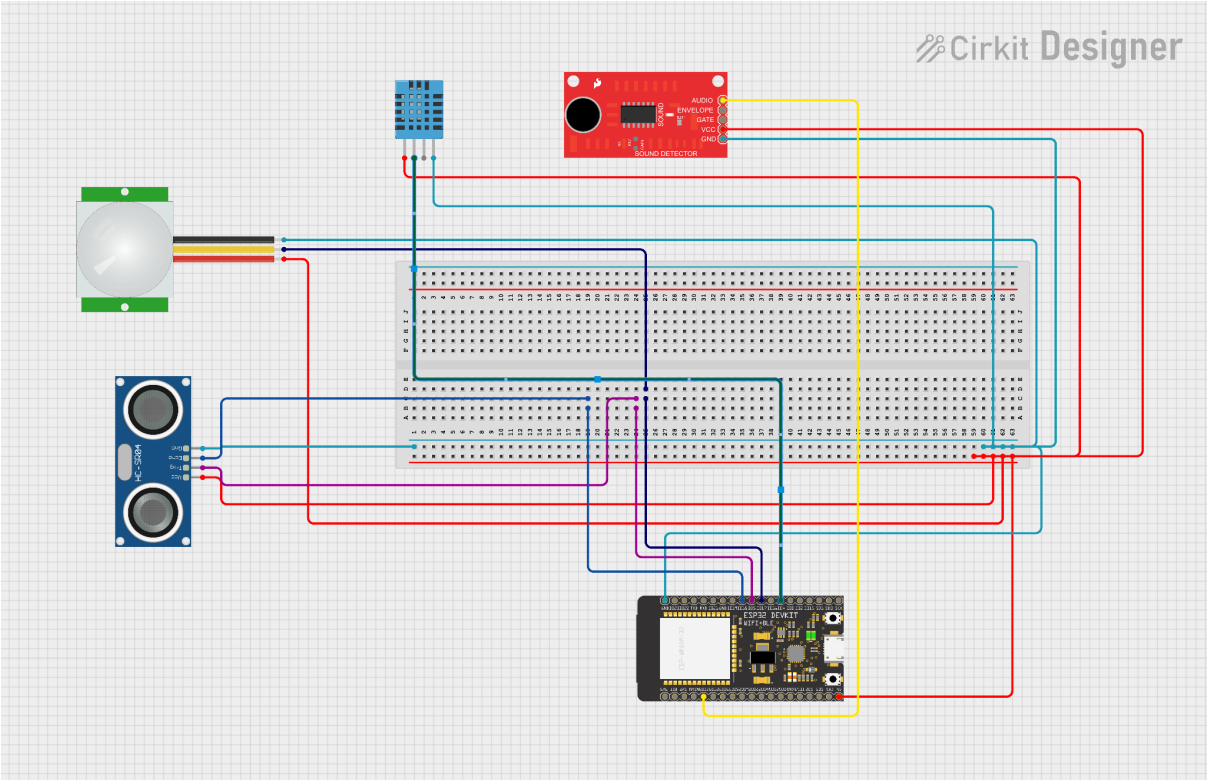


Fig: Circuit du systeme de detection de place avec l'analyse de l'environnement

Afin de tracer intégralement le canal de transmission de l'information et d'assurer le suivi complet de l'exécution, nous avons emprunté des concepts issus des diagrammes utilisés dans les modélisations d'automatismes. Cette approche nous offre une visualisation détaillée des boucles itératives, tant pour le contrôle des chaînes directes que pour les boucles de retour.
Les boucles de retour constituent un mécanisme essentiel, permettant le retour d'informations au sous-système. Ces données rétroactives sont ensuite traitées pour adapter la sortie du sous-système en fonction des besoins de l'utilisateur ou de l'environnement. Cette capacité d'adaptation est étayée par la composante logicielle, qui, à travers des calculs mathématiques et logiques sophistiqués, ajuste la réponse des capteurs. Le diagnostic de ces informations s'opère au travers d'une interface dédiée, que ce soit notre programme GUI ou notre système IoT.
Je comprends votre remarque concernant la répétition. Voici une version révisée, en éliminant la duplication et en améliorant la fluidité du texte :

Le système d'amélioration d'études intègre **un système de détection de places**, lequel, en fonction de la réponse du capteur de mouvement, déclenche deux actions distinctes. Si le capteur de mouvement est activé à distance, ou si le capteur de son ne détecte aucune mesure, la réponse est transmise à l'UCT (notre ESP32). En revanche, si le capteur de distance s'active, indiquant

qu'une personne est entrée dans le champ de travail (marquant le début de l'occupation), une boucle de retour est initiée par le capteur ultrasonique.

Cette boucle persiste tant que la personne demeure à une distance d'environ 30 cm du capteur ultrasonique. Pendant cette période, le capteur de son maintient l'état occupé de la place. Si la personne quitte le champ, un compte à rebours est enclenché. Si aucune réponse n'est reçue par le capteur sonore dans ce laps de temps, indiquant l'absence d'objet détecté à moins de 30 cm, la place se libère. En revanche, si une réponse est reçue, signifiant la présence continue d'un objet à proximité, la place reste occupée. Ainsi, une simple sortie du champ de l'utilisateur ne libère pas automatiquement l'espace.

Cette configuration permet de mettre en place deux boucles de retour dans le canal d'information. Ces boucles corrigent certaines erreurs potentielles et anticipent des situations exceptionnelles, contribuant ainsi à la robustesse et à la fiabilité du système dans des contextes d'utilisation variés.

D'autre part, nous avons **le système environnemental** composé des capteurs sonore et de température, deux systèmes indépendants, linéaires et directs, permettant simplement la transmission d’information brute a l'ESP 32, donc à l'utilisateur sans rétroaction de celle- ci.

Le système Pomodoro comporte deux boucles simples qui connectent deux actionneurs à notre Arduino. La complexité de ce système réside principalement dans la programmation. La première boucle relie le bouton à l'UCT pour effectuer l'action de pause ou de démarrage du chronomètre, tandis que l'écran est lié à l'UCT par une boucle de rafraîchissement des données affichées.

Le système de verrouillage de la boîte est conçu pour verrouiller une boîte à l'aide d'un moteur servo, et ce verrouillage est décidé ou actionné par une clé numérique, implémentée dans une carte NFC, en l'occurrence notre carte étudiante. Ainsi, l'accès à cette boîte est accordé à tous les étudiants de l'université dont la carte est répertoriée dans la base de données. Si le coffre est verrouillé, seul l'étudiant ayant scanné sa carte peut le rouvrir. Dans cette configuration, une boucle regroupe l'Arduino, le RFID (système de capture de clé) et le moteur. Le RFID alerte l'Arduino, qui agit directement sur le moteur pour verrouiller la boîte. Une fois celle-ci verrouillée, le système de détection de carte n'accepte plus qu'une seule clé.

Cette intégration astucieuse entre l'Arduino, le RFID et le moteur crée un système de verrouillage efficace, où l'interaction entre ces composants permet un contrôle sécurisé et sélectif de l'accès à la boîte.

- **Canal de Transmission de l'Information :**
  - ➢ Diagrammes d'Automatismes : Nous utilisons des diagrammes d'automatismes pour modéliser chaque étape du canal de transmission. Cela inclut la capture des signaux par les capteurs, leur traitement par les microcontrôleurs ESP32, et la transmission des données vers le hub central.
  - ➢ Boucles de Retour : Les boucles de retour s'avèrent cruciales pour optimiser le rendement. Les informations récoltées par les capteurs sont acheminées vers le sous-système, qui les analyse pour ajuster dynamiquement ses sorties. Cela garantit une réponse adaptative aux changements dans l'environnement ou aux préférences des utilisateurs.
- **Logique et mathématique discrète des informations :**
  - ➢ Calculs Mathématiques et Logiques : La partie logicielle effectue des calculs complexes pour interpréter les données des capteurs. Cela englobe des calculs mathématiques pour la précision des mesures, ainsi que des opérations logiques pour déclencher des actions spécifiques en réponse à certaines conditions.
  - ➢ Interface de Diagnostic : L'interface de diagnostic, qu'il s'agisse de notre programme GUI ou du système IoT ou même l'écran LCD, permet une analyse approfondie des données. Elle offre une représentation visuelle des informations captées, facilitant ainsi la détection d'anomalies, la prise de décision et la personnalisation des réglages.

Cette approche intégrée entre les composantes matérielles et logicielles, orchestrée par des boucles de retour dynamiques, renforce l'efficacité de notre système en le rendant adaptatif et résilient face aux variations de l'environnement et aux besoins spécifiques des utilisateurs.

V) Organigrammes et algorithmes décrivant les différentes parties logicielles (Software) du projet.

Nos calculateurs, composés d'Arduino et d'ESP32, jouent un rôle central dans le traitement de l'information numérique. Il est important de souligner que ces microcontrôleurs reçoivent l'information soit sous forme analogique, comme dans le cas des capteurs de température ou d'intensité sonore, soit sous forme numérique, comme pour les capteurs RFID ou les détecteurs de mouvement.

A)Microcontroleurs:

L'information reçue par ces microcontrôleurs est exprimée en langage machine binaire, étant la forme de langage compréhensible par ces calculateurs. La phase de compilation du code Arduino porte une importance cruciale, puisqu'elle permet la conversion du langage C, utilisé dans la programmation Arduino, en langage machine. Cela assure une interprétation précise et une exécution adéquate des instructions par les microcontrôleurs, facilitant ainsi le traitement efficace des informations captées par les capteurs.

Nous avons trois microcontroleurs ce qui veuti dire que nous avons trois codes respectifs , la raison du choix de trois microcontroleurs est l'insuffisance du nombre de pins que peut supporter l'arduino. Une des solutions que nous aurions pu choisir aurai ete soit:

- **Arduino Master-slave**: Deux cartes sont programmées pour communiquer en utilisant le protocole synchrone série I2C dans une configuration Maître Écrivain/Esclave Récepteur. Plusieurs fonctions de la bibliothèque Wire d'Arduino sont utilisées à cette fin. Arduino 1, le Maître, est programmé pour envoyer 6 octets de données toutes les demi-secondes à un Esclave adressé de manière unique. Une fois que ce message est reçu, il peut être visualisé dans la fenêtre de surveillance série de la carte Esclave ouverte sur l'ordinateur connecté en USB exécutant le logiciel Arduino (IDE).

  Le protocole I2C implique l'utilisation de deux lignes pour envoyer et recevoir des données : une broche d'horloge série (SCL) que la carte Maître Arduino impulse à intervalles réguliers, et une broche de données série (SDA) sur laquelle les données sont envoyées entre les deux dispositifs. Lorsque la ligne d'horloge change de bas à haut (connue sous le nom de front montant du signal d'horloge), un seul bit d'information - qui formera séquentiellement l'adresse d'un dispositif spécifique et une commande ou des données - est transféré de la carte vers le dispositif I2C via la ligne SDA.

  - **Multiplexeur:** Pour relier nos deux Arduinos, nous avons aussi le choix de l'utilisation d'un multiplexeur, un dispositif agissant comme un commutateur qui permet à notre Arduino principal de sélectionner lequel des deux autres Arduinos il souhaite interagir. En connectant les fils de communication des Arduinos esclaves aux canaux d'entrée du multiplexeur, nous pouvons, à l'aide des broches de commande du multiplexeur contrôlées par l'Arduino principal, choisir le canal correspondant à l'Arduino avec lequel nous voulons échanger des données à un

moment précis. En résumé, cette configuration nous offre la possibilité de partager des ressources ou de réaliser des échanges de données sélectifs entre nos deux Arduinos, ce qui s'avère particulièrement pratique pour notre application en laboratoire.

Le probleme majeur de ces deux methode serait la dependance de l'arduino master, ce qui rend la detection d'erreur fastidieuse et moins fiable le systeme.Deplus nous pouvons travailler en simultane lorsque nous avons plusieurs micro-controleurs

B)Programmation :

1. Ecran LCD avec timer pomorodoro:

a. Déclaration des variables et des constantes :

```
#include <LiquidCrystal.h>

// declare variables for keeping track of time
// and the number of study sessions
int seconds = 0;
int minutes = 0;
int count = 0;

// declare variables for the length of each
// period and the number of sessions before
// a longer break. Change these to customize
// your timer!
const int study_minutes = 25;
const int short_break_minutes = 5;
const int long_break_minutes = 15;
const int repeats = 4;

// a variable used to time the breaks later
int break_duration;
```

Ces variables stockent le temps écoulé, le nombre de minutes, le nombre de sessions d'étude, et la durée des pauses.

repeats indique le nombre de sessions d'étude avant une pause longue.
break_duration est utilisé pour déterminer la durée de la pause.
Déclaration des broches matérielles :

b. Déclaration des broches matérielles :

```
// declare pins used for hardware
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
const int led1 = 6;
const int led2 = 7;
const int button = 8;
```

Configuration des broches pour l'écran LCD, les LED et le bouton.

c. Configuration initiale dans la fonction setup() :

```
void setup() // setup code that only runs once
{
  lcd.begin(16, 2); // Set up the number of columns and rows on the LCD.
  // set LED pins as outputs and button pin as input
  pinMode(led1,OUTPUT);
  pinMode(led2,OUTPUT);
  pinMode(button,INPUT);

  // display initial message to user
  lcd.print("Press button");
  lcd.setCursor(0,1);
  lcd.print("to start");

  // wait for button press to start
  while(digitalRead(button)==LOW){
    // do nothing
  }
}
```

-Initialisation de l'écran LCD et configuration des broches.

-Attente d'appui sur le bouton pour commencer.

d. Boucle principale dans la fonction loop() :

i. Réinitialisation du compteur :

```
void loop()
{
  count = 0;  // set count to zero
```

- Avant de commencer les sessions, le compteur count est réinitialisé à zéro.

ii. Boucle principale (While) :

Cette boucle s'exécute tant que le compteur count est inférieur au nombre total de répétitions (repeats), représentant le nombre de sessions d'étude à effectuer.

```
// count up and display the timer during study period
while(minutes<study_minutes){ // keep counting until we've reached the time limit for a study session
  seconds = 0;
  // print out the timer value in mm:ss format
  while(seconds<60){
    lcd.setCursor(0, 1);
    if(minutes<10){  // if minutes is less than 10, we need to print an extra 0 to the display
      lcd.print("0");
    }
    lcd.print(minutes);
    lcd.print(":");
    if(seconds<10){  // if seconds is less than 10, we need to print an extra 0 to the display
      lcd.print("0");
    }
    lcd.print(seconds);
    // wait for one second then increment the second counter
    delay(1000);
    seconds++;
  }
  // increment the minute counter after 60 seconds have elapsed
  minutes++;

}
```

iii. Session d'étude :

```
// now repeat the process for a study break
lcd.clear();
lcd.setCursor(0, 0);
digitalWrite(led1,LOW);
digitalWrite(led2,HIGH);

// this is a good place to add your own code to control
// other hardware like buzzers or motors (make sure you
```

- L'écran LCD est effacé, et le message "Study time!" est affiché. Les LED sont configurées pour indiquer que c'est le temps d'étude.
- Les variables seconds et minutes sont réinitialisées à zéro.

iv. Pause :

```
    if(count==(repeats-1)){ // do a long break on the last repetition
      break_duration = long_break_minutes;
      lcd.print("Long break!");
    }
    else{  // otherwise do a short break
      break_duration = short_break_minutes;
      lcd.print("Short break!");
    }

    lcd.setCursor(0, 1);
    seconds = 0;
    minutes = 0;

    while(minutes<break_duration){
      seconds = 0;
      while(seconds<60){
        lcd.setCursor(0, 1);
        if(minutes<10){
          lcd.print("0");
        }
        lcd.print(minutes);
        lcd.print(":");
        if(seconds<10){
          lcd.print("0");
        }
        lcd.print(seconds);
        delay(1000);
        seconds++;
      }
      minutes++;
    }
    count++;  // increment the counter for the number of study sessions
  }
}
```

- Affichage du message "Short break!" ou "Long break!" sur l'écran LCD, activation d'une LED différente.
- Comptage du temps de la pause.

v. Répétition :

- Répétition du processus jusqu'à atteindre le nombre total de sessions spécifié.

2. Boîte de verrouillage :

Ce code Arduino utilise une carte RFID, un servomoteur et des LED pour créer un système de verrouillage qui peut être contrôlé par des cartes RFID spécifiques. Explorons chaque partie du code en détail :

a. Inclusions de Bibliothèques :

```
#include <SPI.h>
#include <RFID.h>
#include <Servo.h>
```

- Ces lignes incluent les bibliothèques nécessaires pour la communication SPI, la gestion RFID et le contrôle du servomoteur.

b. Initialisation des Objets et des Variables :

```
RFID rfid(10, 9);       //D10:pin of tag reader SDA. D9:pin of tag reader RST
unsigned char status;
unsigned char str[MAX_LEN]; //MAX_LEN is 16: size of the array
```

- Crée un objet RFID avec les broches de communication SDA (D10) et RST (D9).
- Initialise un tableau pour stocker les données lues de la carte RFID.

```
String accessGranted [2] = {"310988016", "19612012715"};  //RFID serial numbers to grant access to
int accessGrantedSize = 2;                                //The number of serial numbers
```

- Déclare un tableau de chaînes pour stocker les numéros de série RFID autorisés.

```
Servo lockServo;                  //Servo for locking mechanism
int lockPos = 15;                 //Locked position limit
int unlockPos = 75;               //Unlocked position limit
boolean locked = true;

int redLEDPin = 5;
int greenLEDPin = 6;
```

- Crée un objet Servo pour contrôler le servomoteur.
- Définit les broches pour les LED rouges et vertes.

c. Configuration Initiale (setup) :

```
void setup()
{
  Serial.begin(9600);     //Serial monitor is only required to get tag ID numbers and for troubleshooting
  SPI.begin();            //Start SPI communication with reader
  rfid.init();            //initialization
  pinMode(redLEDPin, OUTPUT);     //LED startup sequence
  pinMode(greenLEDPin, OUTPUT);
  digitalWrite(redLEDPin, HIGH);
  delay(200);
  digitalWrite(greenLEDPin, HIGH);
  delay(200);
  digitalWrite(redLEDPin, LOW);
  delay(200);
  digitalWrite(greenLEDPin, LOW);
  lockServo.attach(3);
  lockServo.write(lockPos);         //Move servo into locked position
  Serial.println("Place card/tag near reader...");
}
```

- Initialise la communication série, SPI, et l'objet RFID.
- Configure les broches pour les LED et le servomoteur.
- Effectue une séquence d'allumage des LED pour indiquer le démarrage du système.

d. Boucle Principale (loop) :

```
void loop()
{
  if (rfid.findCard(PICC_REQIDL, str) == MI_OK)   //Wait for a tag to be placed near the reader
  {
    Serial.println("Card found");
    String temp = "";                             //Temporary variable to store the read RFID number
    if (rfid.anticoll(str) == MI_OK)              //Anti-collision detection, read tag serial number
    {
      Serial.print("The card's ID number is : ");
      for (int i = 0; i < 4; i++)                 //Record and display the tag serial number
      {
        temp = temp + (0x0F & (str[i] >> 4));
        temp = temp + (0x0F & str[i]);
      }
      Serial.println (temp);
      checkAccess (temp);     //Check if the identified tag is an allowed to open tag
    }
    rfid.selectTag(str); //Lock card to prevent a redundant read, removing the line will make the sketch read cards continually
  }
  rfid.halt();
}
```

- Attend qu'une carte RFID soit placée près du lecteur.
- Lorsqu'une carte est détectée, lit son numéro de série RFID

e. Fonction de Vérification de l'Accès (checkAccess) :

```
void checkAccess (String temp)    //Function to check if an identified tag is registered to allow access
{
  boolean granted = false;
  for (int i=0; i <= (accessGrantedSize-1); i++)    //Runs through all tag ID numbers registered in the array
  {
    if(accessGranted[i] == temp)            //If a tag is found then open/close the lock
    {
      Serial.println ("Access Granted");
      granted = true;
      if (locked == true)          //If the lock is closed then open it
      {
          lockServo.write(unlockPos);
          locked = false;
      }
      else if (locked == false)   //If the lock is open then close it
      {
          lockServo.write(lockPos);
          locked = true;
      }
      digitalWrite(greenLEDPin, HIGH);    //Green LED sequence
      delay(200);
      digitalWrite(greenLEDPin, LOW);
      delay(200);
      digitalWrite(greenLEDPin, HIGH);
      delay(200);
      digitalWrite(greenLEDPin, LOW);
      delay(200);
    }
  }
  if (granted == false)     //If the tag is not found
  {
    Serial.println ("Access Denied");
    digitalWrite(redLEDPin, HIGH);      //Red LED sequence
      delay(200);
      digitalWrite(redLEDPin, LOW);
      delay(200);
      digitalWrite(redLEDPin, HIGH);
      delay(200);
      digitalWrite(redLEDPin, LOW);
      delay(200);
    }
}
```

- Vérifie si le numéro de série de la carte RFID fait partie des numéros autorisés.
- Si l'accès est autorisé, ouvre ou ferme le verrou en fonction de son état actuel.
- Active une séquence de LED pour indiquer l'état d'accès.

Le programme utilise un lecteur RFID pour détecter les cartes RFID spécifiques, et en fonction du numéro de série de la carte, il détermine si l'accès est autorisé. Si l'accès est autorisé, le servomoteur contrôlant le verrou se déplace entre les positions de verrouillage et de déverrouillage, et des séquences de LED indiquent l'état du système.

3) Systeme de detection de place et d'environement de travail:

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include <DHT_U.h>

#define DHTPIN 21     // Broche DHT11 (capteur de température) a changer en 21
#define DHTTYPE DHT11
DHT_Unified dht(DHTPIN, DHTTYPE);

const int motionSensorPin = 17; // Broche D17 pour le capteur de mouvement (PIR)
const int trigPin = 5;          // Broche D5 pour le capteur ultrasonique - Trigger
const int echoPin = 18;         // Broche D18 pour le capteur ultrasonique - Echo

bool placeOccupee = false;           // Statut de la place
unsigned long lastDetectionTime = 0;  // Temps de la dernière détection
unsigned long vacancyDuration = 10000; // Durée d'absence pour libérer la place (10 secondes)
const int distanceThreshold = 30;     // Seuil de distance pour considérer la place comme occupée (30 cm)

// SOUND SENSOR
int num_Measure = 128;  // Set the number of measurements
int pinSignal = 34;     // Pin connected to the sound sensor module
long Sound_signal;       // Store the value read from the sound sensor
long sum = 0;            // Store the total value of n measurements
long level = 0;          // Store the average value
int soundlow = 40;
int soundmedium = 500;
```

Ce code Arduino utilise un ESP32 pour surveiller différents capteurs, y compris un capteur de température DHT11, un capteur de mouvement PIR, un capteur ultrasonique, et un capteur de son.

Capteur de Température DHT11:

La bibliothèque DHT est utilisée pour lire la température à partir du capteur DHT11 connecté à la broche 21 (DHTPIN).
Les données de température sont affichées sur le moniteur série.

Capteur de Son :

Un capteur de son est connecté à la broche 34 (pinSignal).
Le code réalise 128 lectures du signal sonore et calcule ensuite la moyenne pour obtenir une valeur de niveau sonore.
En fonction de la valeur du niveau sonore, il est indiqué si l'intensité sonore est faible, moyenne ou élevée sur le moniteur série.

Capteur de Mouvement PIR:

Un capteur de mouvement PIR est connecté à la broche 17 (motionSensorPin).
Le code vérifie si le capteur de mouvement détecte une présence.
Si une personne est détectée, le code vérifie la distance avec le capteur ultrasonique.

Capteur Ultrasonique:

Le capteur ultrasonique est connecté aux broches 5 (trigPin) et 18 (echoPin).
La fonction getDistance mesure la distance en utilisant le capteur ultrasonique.
Si une personne est détectée à une distance inférieure à 30 cm, la place est considérée comme occupée.

- Configuration Initiale (setup)

```
void setup() {
  Serial.begin(9600);
  dht.begin();
  pinMode(motionSensorPin, INPUT);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}
```

La fonction setup() configure l'environnement initial. Elle initialise la communication série, active le capteur de mouvement, et configure les broches pour le capteur ultrasonique

- Fonction pour Mesurer la Distance:

```
float getDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  return pulseIn(echoPin, HIGH) * 0.0343 / 2;
}
```

Cette fonction utilise le capteur ultrasonique pour mesurer la distance. Elle envoie un signal, mesure le temps écoulé jusqu'à ce que le signal revienne, et calcule la distance en utilisant la vitesse du son.

```
void loop() {
  int motionState = digitalRead(motionSensorPin);  // Déclarer motionState ici

  // Lecture de la température
  sensors_event_t event;
  dht.temperature().getEvent(&event);
  if (!isnan(event.temperature)) {
    Serial.print("Température : ");
    Serial.print(event.temperature);
    Serial.println(" °C");
  } else {
    Serial.println("Erreur de lecture de la température !");
  }

  // SOUND SENSOR-----------------------------------------
  // Perform 128 signal readings
  for (int i = 0; i < num_Measure; i++) {
    Sound_signal = analogRead(pinSignal);
    sum = sum + Sound_signal;
  }

  level = sum / num_Measure;  // Calculate the average value
  // Adjust the offset based on your sensor
  if (level - 33 < soundlow) {
    Serial.println("Intensity of sound= Low");
  } else if (level - 33 > soundlow && level - 33 < soundmedium) {
    Serial.println("Intensity of sound = Medium");
  } else {
    Serial.println("Intensity of sound= High");
  }

  sum = 0;  // Reset the sum of the measurement values
//------------------------------
  // Si le capteur de mouvement détecte une personne
  if (motionState == HIGH) {
```

```
  sum = 0;  // Reset the sum of the measurement values
//------------------------------
  // Si le capteur de mouvement détecte une personne
  if (motionState == HIGH) {
    // Vérifier la distance avec le capteur ultrasonique
    float distance = getDistance();
    Serial.print("Distance : ");
    Serial.println(distance);

    // Si la personne est à proximité, la place est occupée
    if (distance < distanceThreshold) {
      // Si la place était libre, indiquer qu'elle est occupée
      if (!placeOccupee) {
        Serial.println("Présence détectée à proximité, la place est occupée.");
        placeOccupee = true;
        lastDetectionTime = millis();
      }
    } else {
      // Si la personne est loin, réinitialiser le temps de détection
      lastDetectionTime = millis();
    }
  } else {
    // Si la personne n'est pas détectée depuis un certain temps et la place est occupée, libérer la place
    if (placeOccupee && (millis() - lastDetectionTime > vacancyDuration)) {
      Serial.println("Pas de présence, la place est libre.");
      placeOccupee = false;
    }


  }
 if (placeOccupee == 0) {
  Serial.println("La place est libre.");
} else {
  Serial.println("La place est occupée.");
}
  delay(1000);  // Attendez une seconde avant de lire à nouveau
}
```

La boucle principale (loop) de ce programme Arduino est responsable de la surveillance continue des capteurs et de la gestion de l'état d'occupation d'une place. Elle fonctionne de manière itérative, réalisant plusieurs tâches à chaque itération.

Dans un premier temps, la boucle lit l'état du capteur de mouvement PIR, indiquant s'il détecte une présence à l'aide de la variable motionState. Ensuite, le capteur de température DHT11 est interrogé pour obtenir la température ambiante, dont les données sont affichées sur le moniteur série.

La section suivante de la boucle est dédiée au capteur de son. Elle effectue 128 lectures du signal sonore, calculant ensuite la moyenne pour déterminer le niveau sonore ambiant. En

fonction de cette valeur, le programme catégorise l'intensité sonore comme faible, moyenne ou élevée, et ces informations sont également affichées sur le moniteur série.

La boucle passe ensuite à la vérification du capteur de mouvement et du capteur ultrasonique. Si le capteur de mouvement détecte une présence, le programme mesure la distance entre l'objet détecté et le capteur ultrasonique. Si cette distance est inférieure à 30 cm, la place est considérée comme occupée.

Enfin, le programme gère la libération de la place si aucune présence n'est détectée pendant un certain laps de temps. Il vérifie si la place était occupée et si le délai spécifié par la variable vacancyDuration est écoulé. Si tel est le cas, la place est marquée comme libre.

L'état actuel de la place, la température, la distance et l'intensité sonore sont affichés sur le moniteur série à chaque itération de la boucle. L'ensemble de ces fonctionnalités permet à ce système de surveillance de créer une expérience interactive en temps réel, adaptée à la détection de la présence et à la gestion de l'environnement.

VI) Communication entre les différentes composantes du projet ：

1. Fils reliant chaque composant：

- Les connexions filaires entre les capteurs de température, son, mouvement ultrasonique, et les LEDs au module ESP32 permettent la transmission des données environnementales vers le microcontrôleur pour le traitement.
- Les fils connectant les modules RFID, le servomoteur et les LEDs au module ESP32 permettent la transmission des données d'identification, de contrôle de l'accès et d'état vers le microcontrôleur.
- Les connexions filaires entre le module ESP32 et le dispositif de limitation des distractions ainsi que le système d'antivol et de chronométrage du temps d'études permettent le contrôle et la gestion de ces fonctionnalités par le microcontrôleur.

2. Communication WiFi entre ESP32 et le serveur :

- L'ESP32 communique avec le serveur via une connexion WiFi pour envoyer des données des capteurs.

- Le serveur peut également envoyer des autorisations ou des instructions spécifiques à l'ESP32 via la connexion WiFi.
- La connexion WiFi permet une communication en temps réel des données relatives à l'occupation des bureaux, à l'environnement sonore, thermique et aux fonctionnalités complémentaires entre l'ESP32 et le serveur.

3. **Écran d'affichage :**
    - L'écran d'affichage est connecté à l'arduino et affiche le temps d'etudes et de pause.
    - L'Arduino envoie des commandes à l'écran d'affichage pour mettre à jour son contenu en fonction des données collectées par les capteurs.
4. **Interface IoT :**
    - L'interface IoT (Internet des Objets) fournie par le serveur permet d'afficher la disponibilité des bureaux, les données de température, de son, de mouvement ultrasonique,.
    - Les utilisateurs peuvent surveiller ces informations à distance et prendre des décisions éclairées sur l'utilisation des espaces de bureau en fonction des conditions environnementales et des fonctionnalités complémentaires.

L'ensemble de ces connexions assure une communication bidirectionnelle entre les composants du projet, permettant un contrôle efficace des capteurs environnementaux, du système RFID, des fonctionnalités additionnelles, tout en offrant une expérience d'utilisation intelligente et connectée pour les étudiants sur le campus.

VII) Résolution de problèmes, maintenance:

Pour connecter notre ESP32 au serveur, il suffit de copier l'adresse IP dans notre navigateur pour accéder au site et cela avec le même réseau WiFi. Le serveur diffuse les données en temps réel, mais présente une certaine latence en raison de la faible robustesse du site et des caractéristiques du canal de transmission d'informations utilisé par notre réseau WiFi. De plus, notre ESP32 ne se connecte pas au réseau hotspot de l'iPhone, probablement en raison de la fréquence WiFi de l'iPhone, qui est automatiquement réglée sur 5 GHz au lieu de 2,5 GHz.

Pour le capteur de mouvement, il ne détecte que l'entrée et la sortie de quelqu'un de la place. Cependant, lorsqu'une personne occupe la place, ce dernier reste immobile et le capteur envoie un signal indiquant que la place est disponible. Le capteur de mouvement est conçu

uniquement pour signaler l'entrée et la sortie de la place. Afin de déterminer si la place est toujours occupée, nous avons ajouté un capteur à ultrasons. Ainsi, en combinant ces deux capteurs, nous avons créé un détecteur de présence.

Concernant le capteur de son, nous l'avons connecté en mode analogique afin d'obtenir plusieurs niveaux de seuil, contrairement au mode numérique qui nous aurait limités à un seul seuil. En d'autres termes, pour capter le son ambiant, si nous l'avions configuré en mode numérique, le capteur aurait déclenché un signal uniquement lorsque le niveau sonore atteignait un seuil spécifique. Cependant, dans notre cas, nous souhaitions connaître la valeur du niveau sonore à différentes étapes.

VIII) Difficultés éventuelles.

Afin de pouvoir améliorer l'efficacité nous aurons pu implémenter du machine learning en definissant un valeur butoir comme pour une détection optimal d'espace vide , en récoltant des rétroactions (données) des utilisateurs après une séance d'études ou de recherche de place , puis les traiter par la machine afin d'améliorer l'acuité de la détection, c'est l'apprentissage par répétition.

Dans le cadre du projet, notre équipe a été confrontée à des défis techniques significatifs. Tout d'abord, l'intégration de l'ESP32 sur la protoboard s'est avérée complexe, avec des problèmes persistants sur certains pins malgré plusieurs tentatives de soudage et le remplacement des câbles. Ces difficultés ont requis une analyse approfondie des connexions physiques, des révisions du schéma de câblage et des ajustements pour assurer un fonctionnement stable du microcontrôleur en harmonie avec les autres composants.

Un autre défi s'est présenté lors de l'ajout du paramètre de température sur Arduino IoT. En intégrant le capteur de température au site Arduino IoT, une erreur persistante liée au capteur a été identifiée dans le terminal. Malgré des recherches approfondies et des ajustements dans le code, la cause de cette erreur n'a pas été immédiatement identifiée, nécessitant une recherche intensive pour résoudre efficacement le problème.

Ces défis, bien que complexes, ont finalement été surmontés grâce a la simplification du circuit arduino et la creation d’un serveur avec une page code en html. Ils soulignent l'importance de la planification minutieuse, de la flexibilité dans l'approche des problèmes et de la communication proactive pour résoudre efficacement les obstacles rencontrés lors du développement du système.

En ce qui concerne le système de verrouillage du boîtier, nous avons installé le servo-moteur à l'extérieur du boîtier, ce qui présente un risque de sécurité élevé. Nous

envisageons dans un avenir proche de modifier le système de verrouillage pour le rendre plus fiable.

Conclusion

En résumé, notre projet de système automatisé de détection des places libres sur le campus propose une solution novatrice pour simplifier la recherche d'emplacements propices à l'étude. En évaluant intelligemment les conditions environnementales, telles que le niveau de bruit, la température, et la présence d'autres individus, notre système offre aux étudiants la possibilité de choisir des espaces propices à la concentration et à la productivité.

En complément, notre proposition intègre des dispositifs limitant les distractions et assurant la sécurité des effets personnels, contribuant ainsi à créer un environnement idéal pour les études. Basé sur l'Internet des objets (IoT), notre système garantit une communication en temps réel des données relatives à l'occupation des bureaux et aux conditions environnementales, offrant une expérience efficace et personnalisée.

En réduisant les obstacles à la concentration et en maximisant l'efficacité des études, notre projet s'aligne sur notre engagement envers l'innovation et l'amélioration continue, offrant une solution intégrée pour répondre aux besoins concrets des étudiants en ingénierie mécanique sur le campus.

Annexe

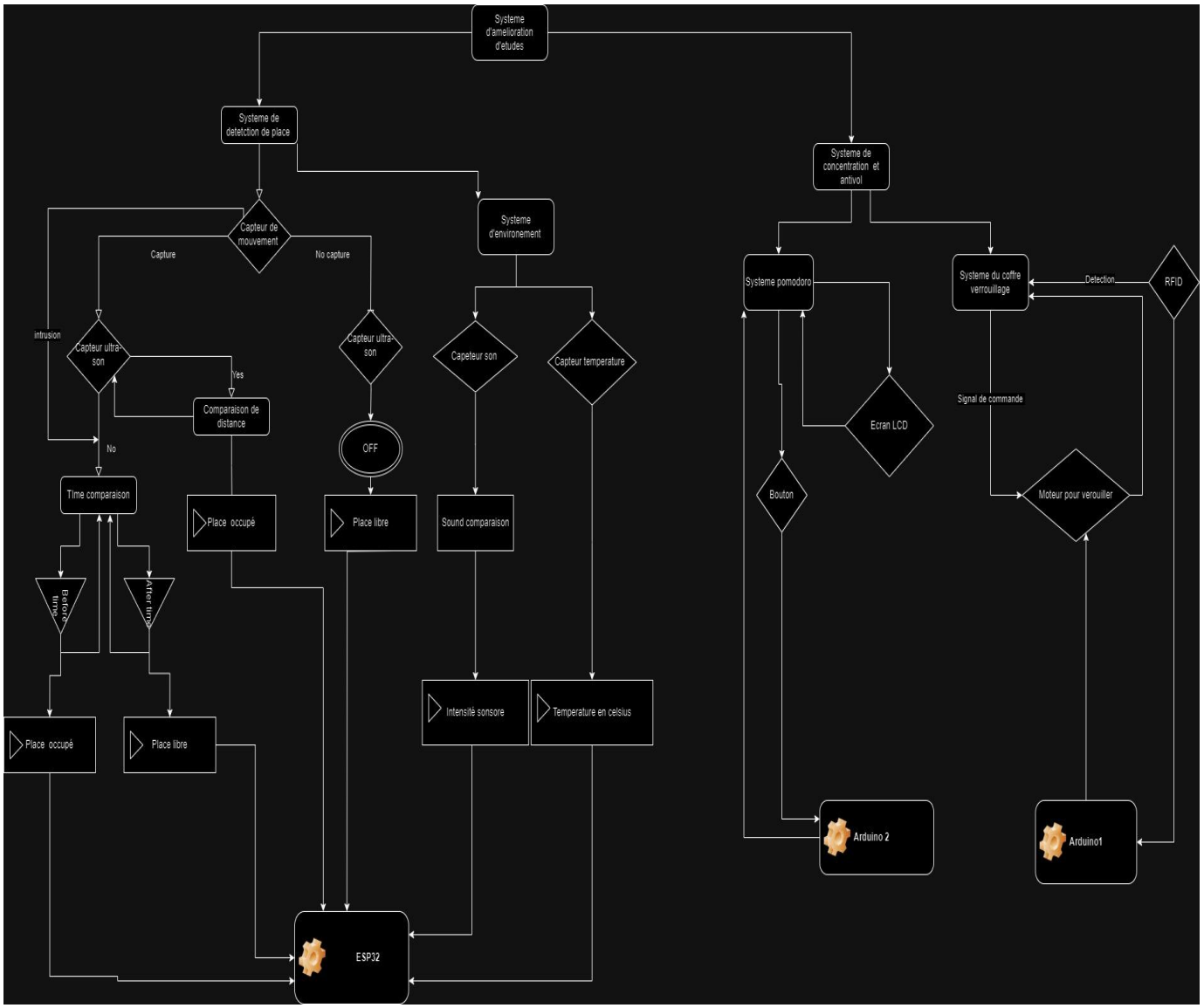
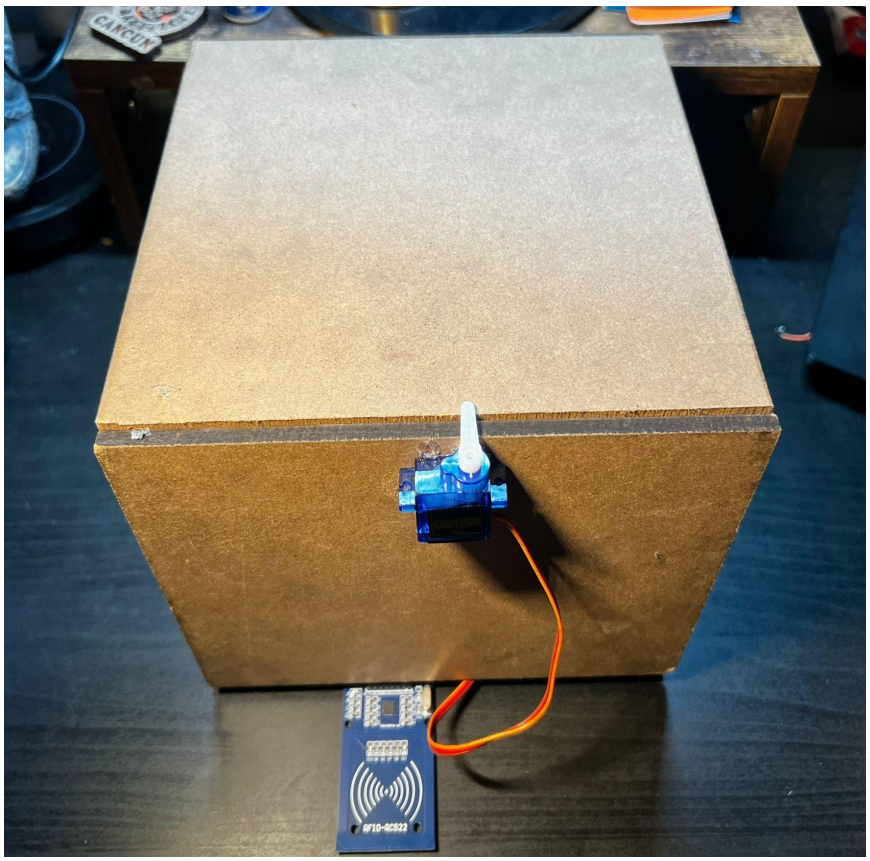


Fig : Modélisation de nos systèmes et boucles de transmission d'information.



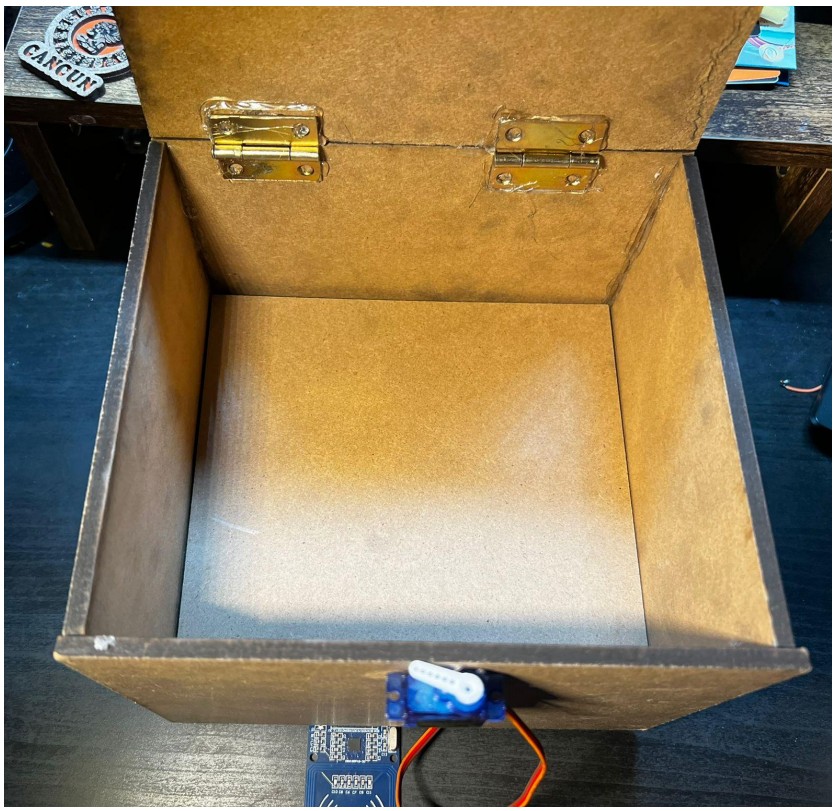
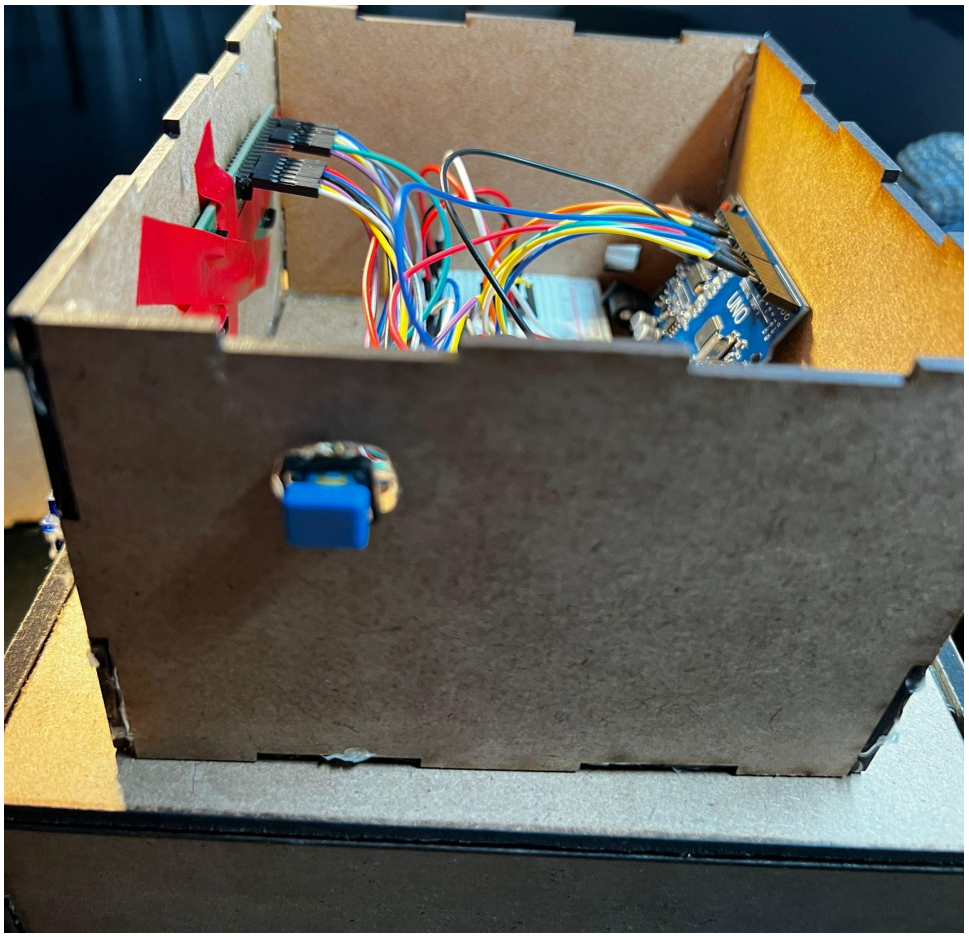
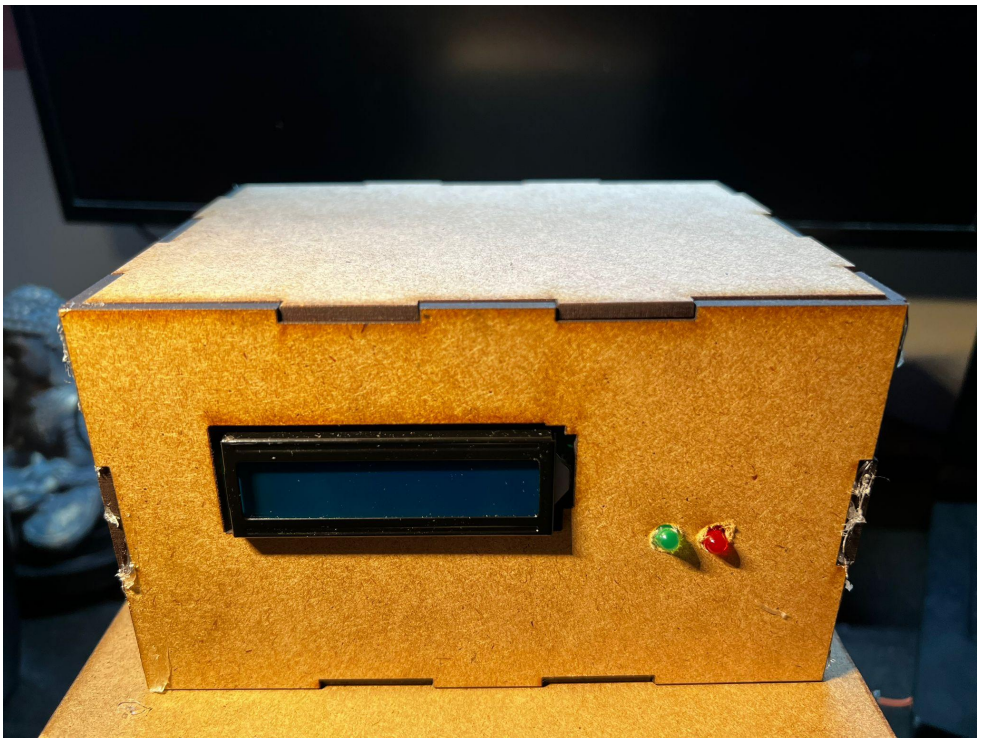


Fig: Boite pour verrouiller les effets personnels RFID





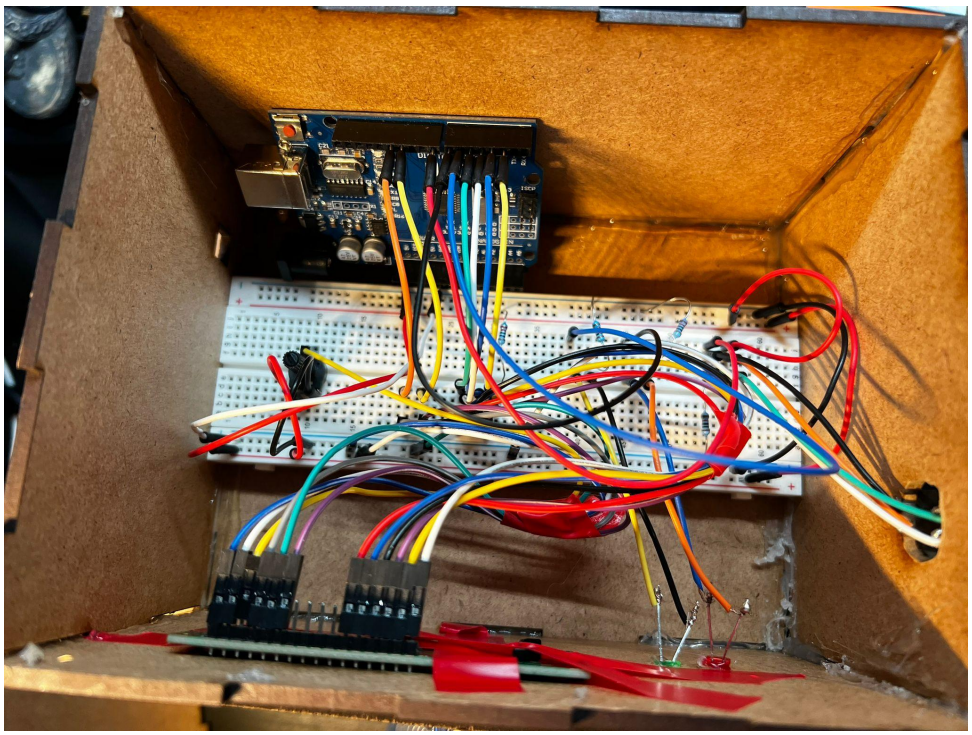
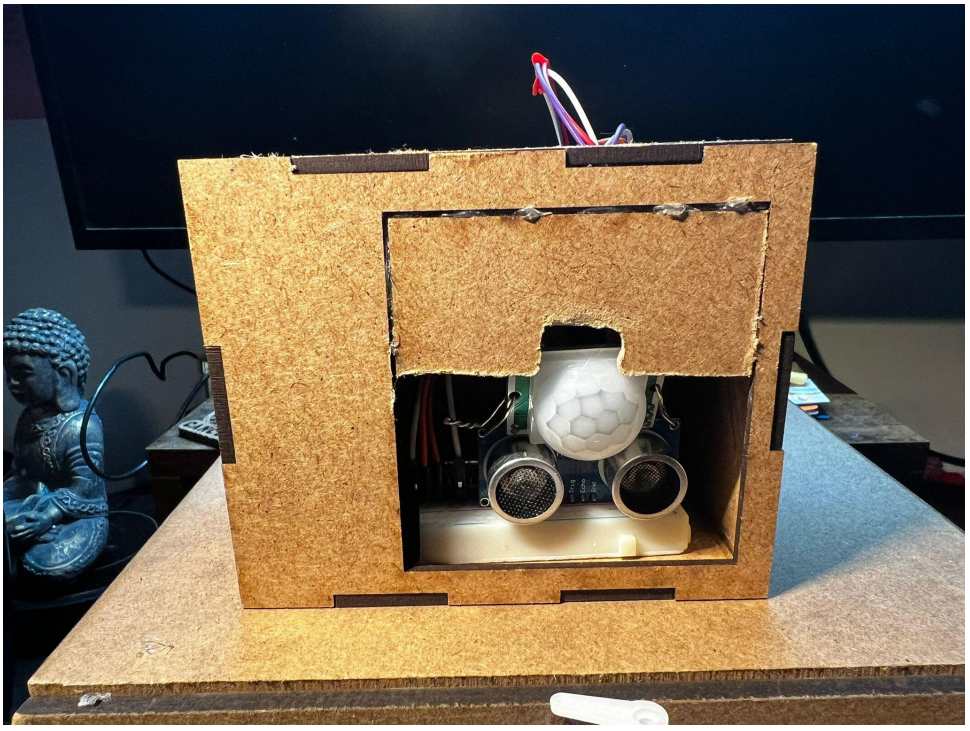


Fig : Systeme de chronometrage pomodoro



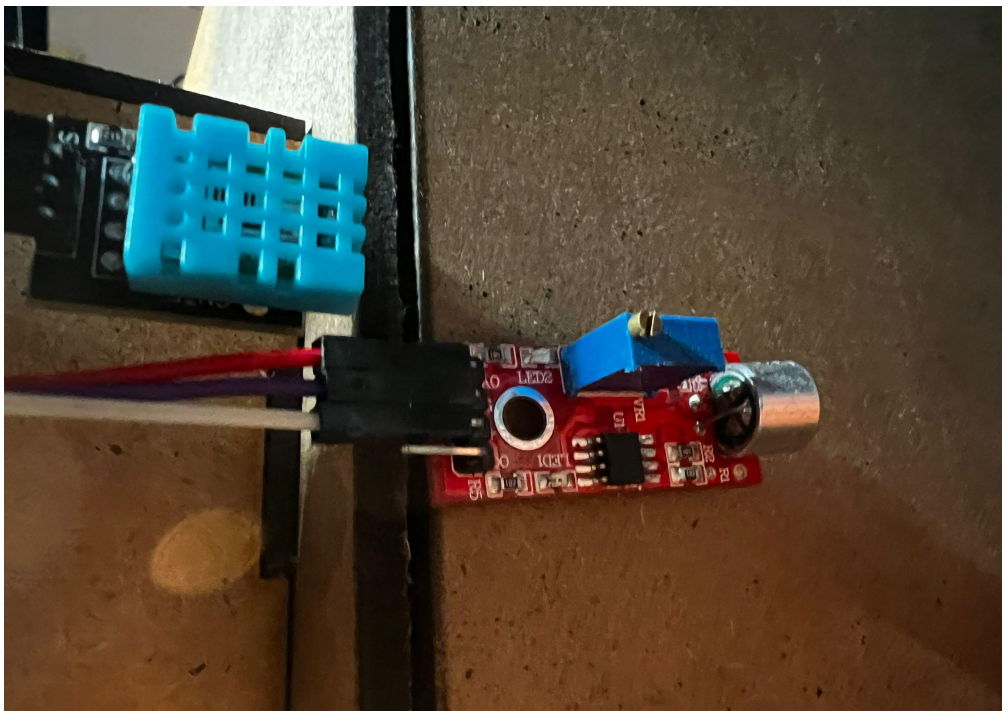


Fig : DDPL , capteur de temperature, et de son

Réference:

https://randomnerdtutorials.com/esp32-web-server-sent-events-sse/

https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2018/08/ESP32-DOIT-DEVKIT-V1-Board-Pinout-36-GPIOs-updated.jpg?quality=100&strip=all&ssl=1

https://www.youtube.com/watch?v=qdFjmaaB2Loà

https://www.handsontec.com/dataspecs/RC522.pdf

https://www.handsontec.com/dataspecs/RC522.pdf

https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

https://projecthub.arduino.cc/edison0215/pomodoro-with-arduino-8bc0b4

https://www.digikey.sk/en/maker/projects/how-to-make-an-arduino-based-rfid-box-lock/a57d9f8ad28043d1b56acbd34d8a55de