

**[Collège Ahuntsic](https://www.collegeahuntsic.qc.ca/" \t "_blank" \o "Retour au site général du Collège Ahuntsic)**

RAPPORT PROJET INTÉGRATEUR POUR AEC INTERNET DES OBJETS ET INTELLIGENCE ARTIFICIELLE

**THÈME**

Réalisation d’une maison intelligente

via un Raspberry Pi

Réalisé par :

Arfaoui Walid,

Henia Mohamed Haikel,

Krayem Manel,

Lemineux Jonathan

1. **Introduction**

L'avènement de la technologie a ouvert la voie à une nouvelle ère de confort et de commodité dans nos foyers. Dans ce contexte, notre projet se concentre sur la conception et la réalisation d'une maison intelligente exploitant les capacités du Raspberry Pi en tant que système de contrôle central. Notre objectif principal est de mettre en évidence l'efficacité de l'IoT (Internet des Objets) dans le domaine de l'automatisation résidentielle, offrant ainsi aux utilisateurs un moyen pratique et accessible de gérer leur environnement domestique à distance.

Aujourd'hui, la technologie est omniprésente dans notre quotidien, des smartphones aux ordinateurs en passant par les objets connectés. Les maisons intelligentes, intégrant des systèmes automatisés, répondent à une demande croissante de confort, de sécurité et d'efficacité énergétique. Notre projet vise à concrétiser cette vision en offrant une solution complète et fonctionnelle.

1. **Contexte, Objectifs & Portée**

Les principaux objectifs de notre projet incluent la conception et la mise en œuvre d'un système de contrôle domotique basé sur le Raspberry Pi. Nous cherchons à exploiter pleinement les capacités de cette plateforme pour surveiller et contrôler divers appareils électriques, capteurs et actionneurs dans la maison. En outre, nous visons à fournir une interface utilisateur conviviale, accessible via une application mobile, pour permettre aux utilisateurs d'interagir facilement avec le système.

En résumé, notre projet s'articule autour de la création d'un prototype fonctionnel et évolutif de maison intelligente, démontrant ainsi les avantages tangibles de l'automatisation résidentielle en termes de confort, de sécurité et d'efficacité énergétique. Nous aspirons à fournir une solution qui servira de référence pour des projets similaires dans le domaine de la domotique et de l'IoT, contribuant ainsi à façonner l'avenir des foyers connectés.

1. **Méthodologie**
   1. Présentation des méthodes de travail adoptées par l'équipe :

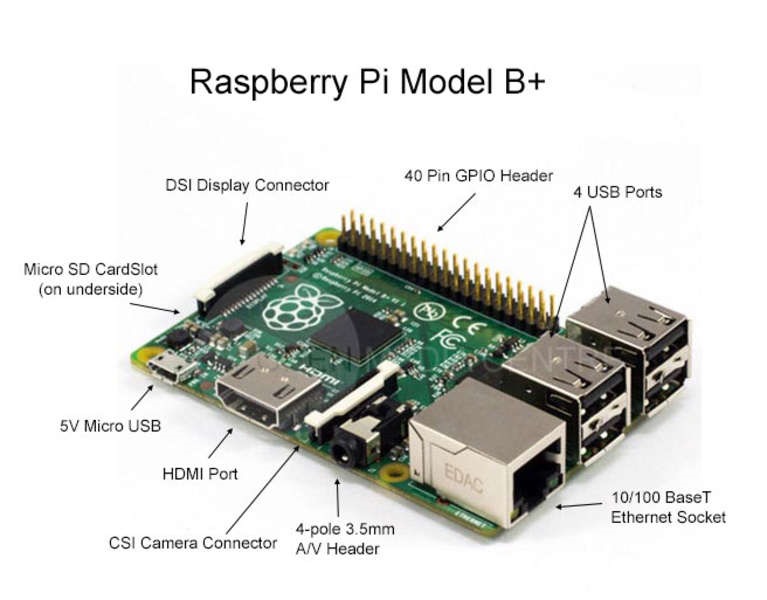
L’approche agile a été adoptée pour la gestion du projet, en utilisant des méthodes telles que Scrum et Kanban. Nous avons organisé des réunions quotidiennes de synchronisation sur Discord, où chaque membre de l'équipe partage son avancement, identifie les obstacles et discute des prochaines étapes. Ces réunions nous ont permis de rester alignés sur les objectifs du projet et de résoudre rapidement les problèmes rencontrés.

Nous avons également utilisé la fonctionnalité Kanban de GitHub pour gérer les tâches et suivre l'avancement du projet. Cela nous a permis de visualiser clairement les étapes de développement, de prioriser les tâches et de réagir rapidement aux changements de priorités. Les révisions de sprint régulières nous ont aidés à évaluer notre progression et à ajuster notre plan en conséquence.

* 1. Aperçu des technologies et outils utilisés :

Pour la communication et la collaboration, nous avons utilisé Discord comme plateforme principale, avec des canaux dédiés pour différents sujets tels que le développement, le contrôle qualité et les discussions générales. Pour la gestion du code et le suivi des versions, nous avons utilisé GitHub, en exploitant les fonctionnalités telles que les pull-requests et les issues tracking.

En ce qui concerne le développement du projet, nous avons utilisé le Raspberry Pi comme plateforme matérielle principale, avec une combinaison de langages JavaScript.



Ce qui nous a permis de concevoir un système en temps réel et qui peut interagir de manière transparente avec le monde physique.

Les bibliothèques JavaScript jouent un rôle crucial dans divers aspects de l'IoT, notamment la communication, le traitement des données, le contrôle à distance, et bien d'autres encore.

* Socket.io : Une bibliothèque de communication évènementielle bidirectionnelle en temps réel, Socket.io permet le contrôle en temps réel des dispositifs IoT sur le web. Elle est nécessaire pour créer des interfaces web réactives pouvant envoyer des commandes et recevoir des données des dispositifs IoT.
* Express.js : Au cœur, un Framework d’application web, Express.js peut être utilisé pour créer des API pour les dispositifs IoT, facilitant les opérations et la surveillance à distance.

Pour l'interface utilisateur, nous avons choisi d'utiliser des technologies web telles que HTML et CSS qui nous ont permis de créer une interface utilisateur conviviale et esthétiquement agréable pour notre application de contrôle domotique.

* 1. Explication de la répartition des tâches parmi les membres de l'équipe :

Chaque membre de l'équipe s'est vu attribuer des responsabilités spécifiques en fonction de ses compétences et de ses intérêts. Le Team Lead était chargé de la gestion des priorités du projet et de l'interface avec les parties prenantes externes. Le Scrum Master et QA était responsable de l'organisation des réunions SCRUM, du suivi de l'avancement des tâches et de l'assurance qualité du produit. Les développeurs ont collaboré étroitement sur le code via GitHub, en se concentrant sur le développement des différentes fonctionnalités et en assurant la cohérence et la qualité du code produit.

Dans l'ensemble, cette répartition des tâches a permis à chaque membre de l'équipe de se concentrer sur des domaines spécifiques, tout en favorisant la collaboration et la communication entre les membres pour atteindre les objectifs du projet de manière efficace et efficiente.

1. **Résultats et Discussions**
   1. Composants nécessaires de réalisation le prototype

* Carte Raspberry Pi 4
* Plaque d’essai
* Câbles de branchements
* Câble Ethernet
* Capteur de proximité et de luminosité APDS-9930 compatible avec Raspberry PI.
* Servo Moteur (utilisé comme un moteur pour un store électrique).
* Module Camera OV5647 avec Filtre Infrarouge Motorisé et 2 LEDs Infrarouges
* Bande LED WS2812B (PCB noir IP65, 1M 144LEDs/m DC5V)
* 3x LEDs (jaune, bleu, vert).
  1. Configuration des composants avec Raspberry Pi
     1. Le servomoteur

Une image contenant câble, outil, machine

Description générée automatiquementLe servomoteur utilisé dans la maquette est de modèle SG90 pour ouvrir ou fermer les stores et les fenêtres. Un servomoteur c'est un actionneur (système produisant une action) qui mélange l'électronique, la mécanique et l'automatique.

Les caractéristiques du SG90 sont les suivantes :

* Dimensions : 22 x 11.5 x 27 mm
* Poids : 9 gr.
* Tension d’alimentation : 4.8v à 6v.
* Vitesse : 0.12 s / 60° sous 4.8v.
* Couple : 1.2 Kg / cm sous 4.8v.
* Amplitude : de 0 à 180°.

Figure 4. 1 : servomoteur modèle SG90

Un fonction JavaScript est développé pour le contrôler.

function toggleComponent(component) {

componentStatusMap[component] = !componentStatusMap[component];

}

function setServoPosition(position) {

servomotor.servoWrite(position);

}

function openServoCompletely() {

setServoPosition(MAX\_SERVO\_POSITION);

console.log("Servo opened completely");

}

function closeServoCompletely() {

setServoPosition(MIN\_SERVO\_POSITION);

console.log("Servo closed completely");

}

function toggleServomotor() {

toggleComponent('toggleServomotor');

if (componentStatusMap.toggleServomotor) {

if (!componentStatusMap.toggleAutoServo) {

openServoCompletely();

console.log('Blinds open');

} else {

// Start auto mode

intervalServoAuto = setInterval(adjustServoBasedOnLight, 1000);

}

} else {

// Close servo and stop auto mode

closeServoCompletely();

clearInterval(intervalServoAuto);

console.log('Blinds closed');

}

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

function toggleAutoServomotor() {

toggleComponent('toggleAutoServo');

clearInterval(intervalServoAuto);

if (componentStatusMap.toggleAutoServo && componentStatusMap.toggleServomotor) {

intervalServoAuto = setInterval(adjustServoBasedOnLight, 1000);

}

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

4.2.2. Éclairage automatique et économique

On utilise pour ce dernier Capteur de proximité et luminosité **APDS-9930** compatible avec Raspberry Pi, ce capteur est numérique et estinterfaceable via I2C.

Le module**APDS-9930** de chez AVAGO Technologies est équipé d'un capteur de lumière ambiante ainsi que d'un capteur de proximité pouvant détecter des objets jusqu'à 100mm sans calibration. Très utile pour les applications de type IHM, il permet notamment de détecter le rapprochement d'unepersonne ainsi que l'éclairage d'un environnement pour gérer la luminosité. Le capteur est numérique et est ineffaçable via I2C. Il fonctionne avec une tension de 3V3.

|  |  |
| --- | --- |
| Broches :   1. VL (alimentation LED IR en option), 2. GND (masse), 3. VCC (entrée 3,3 V pour capteur APDS-9960), 4. SDA (données I2C), 5. SCL (horloge I2C), 6. INT (interruption). | Spécifications :   * Capteur de lumière ambiante   + Possibilité de programmer des interruptions selon les niveaux de luminosité   + Résolution de mesure de 16 bit * Capteur de proximité   + Détection d'objet jusqu'à 100mm sans calibration   + LED IR intégrées * Interface I2C jusqu'à 400KHz (Adresse 0x39 sur 7 bits) * Pin d'interruption dédiée et programmable * Alimentation  2V2 < VCC < 3V6 |

Une image contenant Composant de circuit, Composant électronique, Composant de circuit passif, Ingénierie électronique

Description générée automatiquement

# Figure 4. 2 : Capteur de proximité et luminosité APDS-9930

Script pour l’éclairage automatique et économique

const MIN\_LUMINOSITY = 0;

const MAX\_LUMINOSITY = 1000;

const MIN\_SERVO\_POSITION = 750;

const MAX\_SERVO\_POSITION = 2000;

const servomotor = new pigpio.Gpio(23, { mode: pigpio.Gpio.OUTPUT });

function readLuminosity() {

const luminosityData = i2c1.readWordSync(DEVICE\_ADDRESS, CDATAL);

return luminosityData;

}

function readProximity() {

const proximityData = i2c1.readWordSync(DEVICE\_ADDRESS, PDATAL);

return proximityData;

}

function adjustServoBasedOnLight() {

if (!componentStatusMap.toggleAutoServo) {

return; // If auto mode is not enabled, do nothing

}

const luminosity = readLuminosity();

const servoPosition = mapServoPosition(luminosity);

setServoPosition(servoPosition);

console.log(`Servo adjusted based on light. Luminosity: ${luminosity}, Servo Position: ${servoPosition}`);

}

function toggleAutoServomotor() {

toggleComponent('toggleAutoServo');

clearInterval(intervalServoAuto);

if (componentStatusMap.toggleAutoServo && componentStatusMap.toggleServomotor) {

intervalServoAuto = setInterval(adjustServoBasedOnLight, 1000);

}

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

function toggleProximity() {

toggleComponent('toggleProximity');

if (componentStatusMap.toggleProximity) {

clearInterval(intervalProximity);

intervalProximity = setInterval(adjustLightBasedOnProximity, 1000);

} else {

clearInterval(intervalProximity);

}

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

Une image contenant texte, diagramme, capture d’écran, Police

Description générée automatiquement4.2.3 Les LEDs

Une diode électroluminescente (abrégé en LED, de l’anglais : *light emitting diode*), est un dispositif opto-électronique capable d’émettre de la lumière lorsqu’il est parcouru par un courant électrique. Ces LED feront office d’éclairage dans notre maquette et servirons d’indicateur lumineux pour nos circuits.

Figure 4.3 – Une LED

Script utilisé pour l’éclairage

const yellowLed = new Gpio(19, 'out');

const greenLed = new Gpio(26, 'out');

const blueLed = new Gpio(13, 'out');

function setLedPins() {

yellowLed.writeSync(componentStatusMap.toggleYellow ? 1 : 0);

greenLed.writeSync(componentStatusMap.toggleGreen ? 1 : 0);

blueLed.writeSync(componentStatusMap.toggleBlue ? 1 : 0);

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

4.2.4. La sécurité

A close-up of a camera

Description automatically generatedOn a utilisé la caméra OV5647 avec filtre infrarouge motorisé et 2 LEDs infrarouges, afin de surveiller l’entourage devant la porte, jour et nuit. Ce module de caméra de vision nocturne est parfaitement adapté pour répondre aux besoins variés en matière de capture d'images et de vidéos haute résolution, que ce soit pour des applications de sécurité domestique, de surveillance ou pour des projets de développement technologique.

Figure 4.4 – Caméra OV5647

|  |  |
| --- | --- |
| Description **:**  Le module de caméra de vision nocturne avec 5 millions de pixels est un composant essentiel pour les projets impliquant la capture d'images et de vidéos de haute qualité, même dans des conditions de faible luminosité. Conçu pour être compatible avec Raspberry Pi B 3/2, ce module offre une résolution maximale de 2592 x 1944 pixels et un champ de vision de l'objectif de 75 degrés. Grâce à ses fonctionnalités avancées, telles que le commutateur automatique R-CUT entre les modes jour et nuit, et ses deux lumières de remplissage 850IR, ce module garantit une qualité d'image optimale dans divers environnements. | Spécifications :   * Résolution maximale : 5 millions de pixels (2592 x 1944) * Champ de vision de l'objectif : 75 degrés * Prise en charge des cadres : 1080p30 pour la capture vidéo, 720p60 pour l'enregistrement vidéo * Technologie du capteur photo : CMOS * Type d'objectif : Focale fixe * Fonctionnalité spéciale : Vision nocturne avec commutateur automatique R-CUT * Lumières de remplissage : 2 lumières de remplissage 850IR intégrées * Compatibilité : Raspberry Pi B 3/2 |

Script utilisé pour la caméra :

function toggleInfrared() {

toggleComponent('toggleInfrared');

infrared.writeSync(componentStatusMap.toggleInfrared ? 1 : 0);

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

function takeSnapshot() {

console.log('Fetching snapshot...');

fetch('http://jaunepi.local:8088/0/action/snapshot')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

return response.text();

})

.catch(error => console.error('Error fetching snapshot:', error));

}

function startMovie() {

console.log('Starting movie...');

fetch('http://jaunepi.local:8088/0/action/eventstart')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

})

.catch(error => console.error('Error starting movie:', error));

console.log('Movie started.');

}

function endMovie() {

console.log('Ending movie...');

fetch('http://jaunepi.local:8088/0/action/eventend')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

})

.catch(error => console.error('Error ending movie:', error));

console.log('Movie ended.');

}

function toggleRecording() {

toggleComponent('toggleRecording');

if (componentStatusMap.toggleRecording) {

startMovie();

} else {

endMovie();

}

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

function startDetection() {

console.log('Starting detection...');

fetch('http://jaunepi.local:8088/0/detection/start')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

})

.catch(error => console.error('Error starting detection:', error));

console.log('Detection started.');

}

function pauseDetection() {

console.log('Pausing detection...');

fetch('http://jaunepi.local:8088/0/detection/pause')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

})

.catch(error => console.error('Error pausing detection:', error));

console.log('Detection paused.');

}

function toggleDetection() {

updateDetectionStatus()

toggleComponent('toggleDetection');

if (componentStatusMap.toggleDetection) {

startDetection();

} else {

pauseDetection();

}

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

function updateDetectionStatus() {

console.log('Fetching detection status...');

fetch('http://jaunepi.local:8088/0/detection/status')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

return response.text();

})

.then(data => {

console.log(data);

const isActive = data.includes('status ACTIVE');

if (isActive) {

componentStatusMap.toggleDetection = true;

} else {

componentStatusMap.toggleDetection = false;

}

})

.catch(error => console.error('Error:', error));

}

function restartMotion() {

console.log('Restarting motion...');

fetch('http://jaunepi.local:8088/0/action/restart')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

})

.catch(error => console.error('Error restarting motion:', error));

}

4.2.5. La bande LED

A close-up of a light strip

Description automatically generatedOn a utilisé une bande LED de type WS2812B afin de créer une lumière multifonctionnelle.

Figure 4.5 – LED de type WS2812B

|  |  |
| --- | --- |
| Description **:**  La bande LED WS2812B est une solution d'éclairage polyvalente qui offre une gamme de fonctionnalités pour diverses applications. Cette bande LED est équipée de LEDs individuellement contrôlable, ce qui permet de créer des effets lumineux personnalisés et dynamiques. Son design flexible et découpable permet de l'adapter facilement à différentes longueurs et configurations, offrant ainsi une grande souplesse d'utilisation. Avec sa compatibilité avec une alimentation CC 5V et un contrôleur LED, elle peut être intégrée dans une variété d'environnements, que ce soit dans des lieux commerciaux, des espaces résidentiels ou des projets de bricolage. | Spécifications :   * Tension d'entrée : DC 5V * Ressource LED : WS2812B LED * Contrôle individuel des LEDs * Type d'IC : version améliorée ws2812 IC (intégré à l'intérieur de la LED SMD RVB 5050) * Échelle de gris : 256 * Couleur FPC : noir/blanc * Couleurs d'éclairage : RVB pleine couleur, changement de couleur de rêve * Puissance: 28 W/m * Largeur FPCB : 12 mm * IP65 avec revêtement en silicone |

Script utilisé pour la bande LED

function turnOnLights() {

console.log('Turning on lights...');

fetch('http://jaunepi.local:5000/on')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch /on');

}

// Chain the second fetch request

return fetch('http://jaunepi.local:5000/daylight');

})

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch /random\_color');

}

console.log('Lights turned on successfully.');

})

.catch(error => console.error('Error turning on lights:', error));

}

function turnOffLights() {

console.log('Turning off lights...');

fetch('http://jaunepi.local:5000/off')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

})

.catch(error => console.error('Error turning off lights:', error));

}

function stripRandomColor() {

console.log('Random light color...');

fetch('http://jaunepi.local:5000/random\_color')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

} else {

componentStatusMap.toggleStrip = true;

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

})

.catch(error => console.error('Error turning off lights:', error));

}

function toggleStrip() {

toggleComponent('toggleStrip');

if (componentStatusMap.toggleStrip) {

turnOnLights();

console.log('LED Strip turned on');

} else {

turnOffLights();

console.log('LED Strip turned off');

}

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

function adjustLightBasedOnLuminosity() {

if (readLuminosity() < 12) {

if (!componentStatusMap.toggleStrip) {

toggleStrip();

}

} else {

if (componentStatusMap.toggleStrip){

toggleStrip();

}

}

}

function toggleNight() {

toggleComponent('toggleNight');

if (componentStatusMap.toggleNight) {

clearInterval(intervalNight);

intervalNight = setInterval(adjustLightBasedOnLuminosity, 1000);

} else {

clearInterval(intervalNight);

}

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

function startRainbow() {

console.log('Rainbow!!!');

fetch('http://jaunepi.local:5000/rainbow/start')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

})

.catch(error => console.error('Error starting rainbow', error));

}

function stopRainbow() {

console.log('No more rainbow...');

fetch('http://jaunepi.local:5000/rainbow/stop')

.then(response => {

if (!response.ok) {

throw new Error('Failed to fetch');

}

})

.catch(error => console.error('Error stopping rainbow:', error));

}

function toggleRainbow() {

toggleComponent('toggleRainbow');

if (componentStatusMap.toggleRainbow) {

startRainbow();

console.log('Rainbow!');

} else {

stopRainbow();

console.log(" :'( ");

}

socket.emit('message-from-robot', { componentStatusMap, to: 'USER' });

}

4.3. Problèmes rencontrés

Pendant le développement du projet, plusieurs problèmes ont été rencontrés et résolus pour assurer le bon fonctionnement du système. Tout d'abord, pour contrôler les LEDs, il était nécessaire d'utiliser une bibliothèque spécifique (rpi-ws281x) qui n'était pas compatible avec le serveur existant. Par conséquent, une solution a été trouvée en utilisant un serveur Flask dédié pour la gestion de la bande LED, plutôt que de s'appuyer sur le serveur déjà en place qui était uniquement compatible avec JavaScript.

<https://github.com/JauneO/jaunepi/blob/main/strip-led-server/server.py>

De plus, des difficultés ont été rencontrées lors de l'installation de la caméra, nécessitant l'installation de libcamera-v4l2 et l'utilisation d'une commande spécifique (sudo libcamerify motion) trouvée dans une publication sur un forum Raspberry Pi.

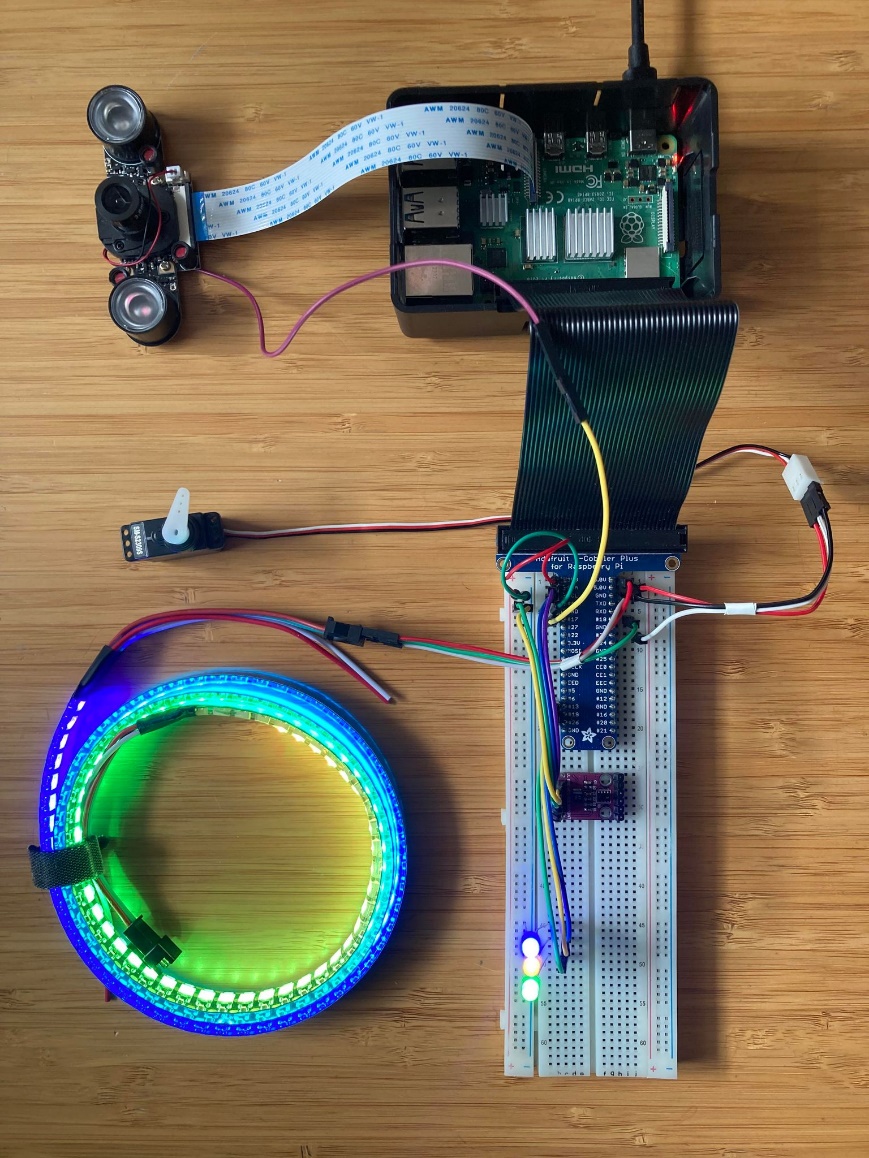
https://forums.raspberrypi.com/viewtopic.php?t=358211

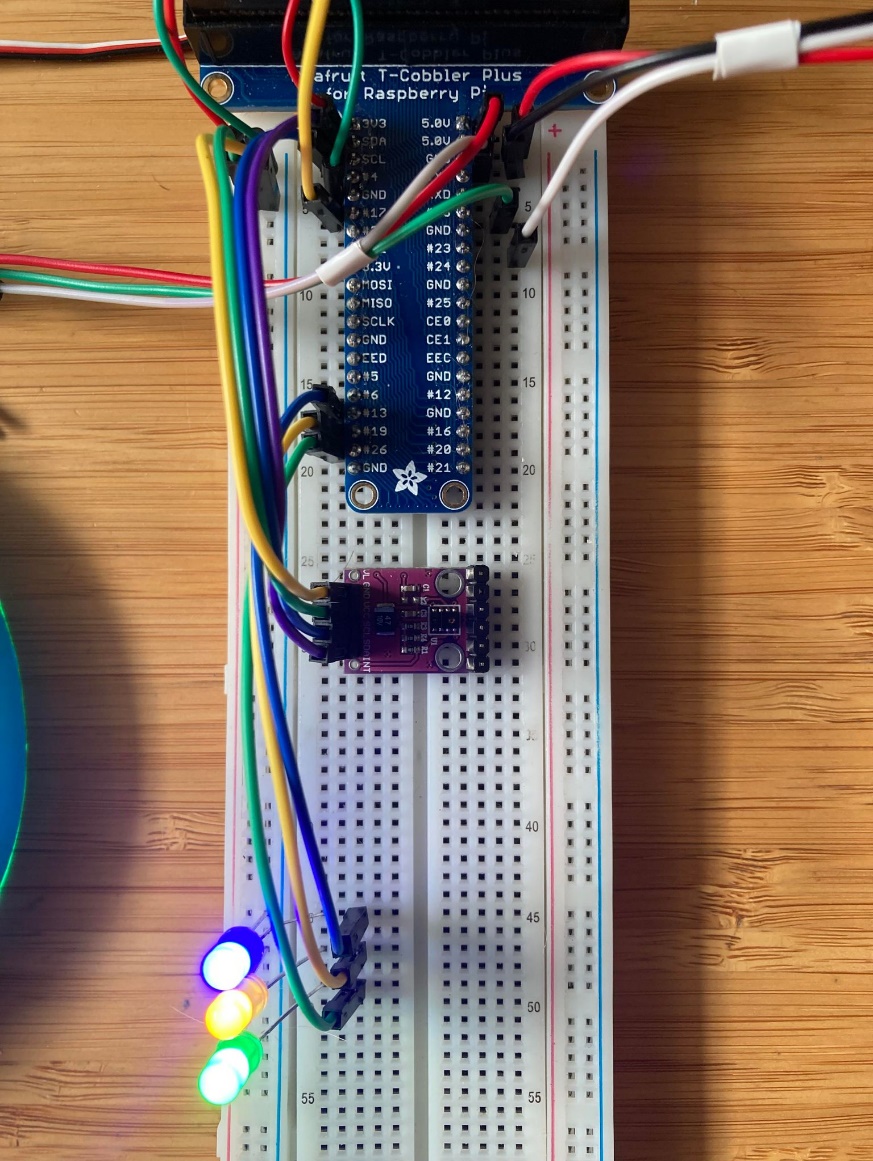
Une autre problématique a été liée à des erreurs CORS (Cross-Origin Resource Sharing) qui se sont produites en raison de requêtes effectuées à partir de domaines différents de celui où les ressources étaient hébergées. Cela est survenu en raison de l'utilisation de trois serveurs provenant de ports différents (8080, 8088, 5000). Les erreurs CORS ont été résolues en configurant correctement les entêtes CORS dans les réponses HTTP des serveurs pour autoriser l'accès aux ressources depuis d'autres domaines.

Enfin, des problèmes matériels ont été rencontrés avec le capteur de proximité et de luminosité APDS-9930 en raison de mauvais contacts. Ces problèmes ont été résolus en vérifiant et en assurant une connexion correcte du capteur pour garantir son bon fonctionnement.

En somme, malgré ces défis rencontrés, des solutions appropriées ont été mises en place pour garantir le bon fonctionnement et la fiabilité du système d'automatisation de bureau basé sur le Raspberry Pi et les technologies IoT.

* 1. Réalisation pratique





Commentaires

|  |  |  |
| --- | --- | --- |
| **Capteur de proximité et luminosité APDS-9930 :**  1. Le capteur de proximité et de luminosité APDS-9930 est utilisé pour détecter la luminosité ambiante et la proximité d'objets. Lorsque la luminosité ambiante est supérieure à un seuil prédéfini, l'action associée (par exemple, allumer une lumière) reste désactivée, indiquant ainsi l'absence de besoin d'éclairage supplémentaire.  2. En revanche, si la luminosité ambiante est inférieure à ce seuil, ou si un objet est détecté à proximité, l'action associée est activée, signalant ainsi la présence d'une condition nécessitant une intervention (par exemple, allumer une lumière en cas d'obscurité ou de détection de mouvement). | **Caméra OV5647 :**  1. La caméra OV5647 est utilisée pour surveiller l'environnement devant la porte, de jour comme de nuit. Lorsque la caméra détecte un mouvement ou un changement de luminosité significatif, elle active l'enregistrement vidéo ou prend des instantanés, signalant ainsi une potentielle activité devant la porte.  2. L’interface peut également être configurée pour afficher en temps réel les images ou vidéos capturées par la caméra OV5647 sur le moniteur série, offrant ainsi une surveillance en direct de l'entrée de la maison. | **Bande LED de type WS2812B :**  1. La bande LED de type WS2812B est utilisée pour créer un éclairage d'ambiance personnalisable et multifonctionnel dans la maison intelligente. Elle offre la possibilité de créer des effets lumineux dynamiques et colorés, adaptés à différentes occasions et préférences. Par exemple, la bande LED peut être programmée pour changer de couleur en fonction de la musique.  2. L’interface également être utilisé pour contrôler la bande LED en temps réel, en ajustant les couleurs, les motifs et la luminosité en fonction des besoins et des préférences des utilisateurs. Cela permet une personnalisation complète de l'éclairage de la maison, offrant ainsi une expérience visuelle immersive et unique. |

4.4 Interface graphique

Notre application sert à contrôler quelque objet dans notre smart home : elle assure la possibilité de surveiller l’entourage devant la porte d’entrée avec une caméra, Elle nous permet d’ouvrir et de fermer la Store et aussi elle assure le confort à partir de contrôler l’éclairage.

L’interface de contrôle graphique contient 15 boutons de contrôle :

1. Contrôle lumière
   1. Allumer et éteindre les 3 lumières (LEDs) : **Cuisine** (Jaune), **Salon** (Vert) et **Entrée** (Bleu)
2. Contrôle store
   1. **Ouvrir** et **fermer** **le** **store**
   2. **Automatisation** : Activer ou désactiver l’ajustement automatique du store selon la luminosité
3. Camera de sécurité
   1. **Capture d’écran** : Sauvegarder une photo
   2. **Enregistrer** : Commencer et interrompre un enregistrement vidéo
   3. **Détection** : Activer et désactiver la détection, c’est-à-dire l’enregistrement automatique de vidéo lorsqu’un mouvement est détecté
   4. **Redémarrer** le serveur vidéo
   5. **Infrarouge** : Activer et désactiver la vision **infrarouge**
4. Bande LED
   1. Allumer et fermer la **bande LED**
   2. Choisir une **couleur aléatoire** pour la lumière
   3. **Veilleuse** : Activer et désactiver le mode veilleuse, c’est-à-dire que la lumière s’allume ou s’éteint selon un seuil de luminosité
   4. **Proximité** : Activer et désactiver le mode de proximité, c’est-à-dire que la lumière s’allume ou s’éteint selon un seuil de proximité
   5. Activer et désactiver le mode **Arc-en-ciel**, c’est-à-dire que le cyclage de couleur

|  |  |
| --- | --- |
| Une image contenant texte, capture d’écran, conception  Description générée automatiquement  Figure 4.6: La page de bienvenue | Figure 4.7: La page principale Une image contenant texte, capture d’écran, Site web, Publicité en ligne  Description générée automatiquement |

1. **Conclusion**

Ce travail est basé sur la réalisation et la conception d’une maison intelligente. Ce projet permit d’évaluer et enrichir les connaissances théoriques ainsi que pratiques dans le domaine de l'intelligence artificielle et IoT. Ce dernier qui est devenu un grand domaine de recherche et un grand marché de travail au monde. En plus, la réalisation d'un tel projet permis d'apprendre l'un des importants langages de programmation (javascript, python, html, css) où ce dernier est choisi pour la programmation de la carte Raspberry pi pour contrôler les divers appareils électroniques et électriques. Le mémoire montre que les capacités d’une carte Raspberry Pi sont élevées malgré sa taille réduite. Où une maquette d’une maison est réalisée afin de la contrôler tenant compte plusieurs fonctionnalité ouverture et fermeture des portes et des fenêtres, éclairage automatique et économique, sécurité avec caméra, ainsi que l’utilisation d’une interface graphique pour contrôler divers appareils électroniques et électriques à partir d'un appareille mobile ou un ordinateur. Ce travail a envie de réaliser le projet pour une maison réelle et de développer une application web comme une interface de contrôle pour commander et visualiser l’état de la maison à distance.