وزارة التعليم العالي و البحث العلمي

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

جامعـة عبد الحميد بن باديس مستغانم

Université Abdelhamid Ibn Badis Mostaganem

كـلـيـة الـعـلـوم والـتـكـنـولـوجـيـا

Faculté des Sciences et de la Technologie

DEPARTEMENT DE GENIE ELECTRIQUE

N° d'ordre : M…../GE/2026

MEMOIRE

Présenté pour obtenir le diplôme de

MASTER EN TELECOMMUNICATIONS

Spécialité : Systèmes des télécommunications

Par :

Melle BENRADJA Marwa

Thème

Conception et réalisation pratique d'un système de stationnement intelligent basé sur IoT et application web

Soutenu publiquement le 28/06/2026 devant le jury composé de :

Président :	**M. BENBEKHTI Brahim**	**MCA**	**Université de Mostaganem**
Examinateur :	**M. HENNI Sid Ahmed**	**MCA**	**Université de Mostaganem**
Encadrant :	**M. ABED Mansour**	**Professeur**	**Université de Mostaganem**
Co-Encadrant :	**M. LARBI BEKLAOUZ Hadj**	**MCB**	**Université de Mostaganem**

Année Universitaire 2025/2026

Dédicace

Je tiens à dédier cet humble travail :

A ma tendre mère **Fatma** et mon très cher père **Sbaa**

A ma sœur : **Ahlam** et surtout mes très chère nièces **Haroun, Miral, Mayar**

A mon frère **BENRADIA Abdelkader**

Et toute ma famille

Je tiens aussi de dédier ce travail à **M. Abed Mansour**

A mes meilleures amies :

NAGHIB Romaissa , TITAOUI Soumia, et sa fille Soujoud , **FLIH Aya**

REMECIEMENT

Avant toute chose, je tiens à remercier Allah, le Tout-Puissant, pour Sa bénédiction, Sa miséricorde et Son soutien. C'est grâce à Lui que j'ai trouvé la force, la patience et la persévérance nécessaires pour accomplir ce travail et poursuivre mon parcours. Sans Sa volonté, rien n'aurait été possible.

J'adresse mes sincères remerciements à mon encadreur, Monsieur **ABED Mansour**, pour son accompagnement, sa disponibilité, ses précieux conseils, ses orientations et ses corrections qui ont grandement contribué à la réalisation de ce projet.

À ma très chère **mère TITAOUI Fatma**, je dédie une reconnaissance infinie. Merci pour tous les sacrifices que tu as consentis afin de nous voir réussir, pour ton amour inconditionnel, tes prières, ta patience, tes encouragements et tous les efforts que tu as fournis pour notre éducation. Aucun mot ne saurait exprimer toute ma gratitude envers toi.

À mon cher père, qu'Allah lui accorde Sa miséricorde et l'accueille dans Son vaste Paradis. Je n'ai peut-être pas eu la chance de partager beaucoup de souvenirs avec toi, mais le nom que je porte est pour moi la plus belle des fiertés et le plus précieux des héritages. Que ce succès soit une lumière pour ton âme.

À mon oncle TITAOUI Ahmed, qui a été pour moi bien plus qu'un simple oncle, mais une véritable figure paternelle. Merci pour tes conseils, ton soutien, ton accompagnement et ta présence à chaque étape importante de ma vie.

À mon cher frère Abdelkader, mon soutien après Allah, mon protecteur et mon compagnon de route. Merci d'avoir toujours été à mes côtés, de m'avoir soutenue dans les moments difficiles et d'avoir rendu plus faciles les obstacles qui se présentaient sur mon chemin. Je te serai éternellement reconnaissante.

À ma chère sœur **Ahlam** *et à son époux* **FEDIAR Ahmed**, *je vous remercie sincèrement pour votre soutien, vos conseils et vos encouragements qui m'ont aidée à avancer avec confiance.*

J'adresse également mes remerciements à tous les membres des familles **BENRADIA** *et* **TITAOUI**, *pour leur affection, leur soutien et leurs encouragements.*

Mes remerciements vont également à tous les enseignants qui ont contribué à notre formation et à l'acquisition de nos connaissances, ainsi qu'à toutes les personnes qui ont participé, de près ou de loin, à la réalisation de ce projet.

Nous tenons à exprimer notre sincère gratitude au Laboratoire de recherche Signaux et Systèmes (LSS) de l'Université de Mostaganem et à tous ses membres pour l'aide qu'ils nous ont apportée en termes d'équipement, d'espaces de travail, de commodités, de conseils, d'orientation et de suivi jusqu'à l'achèvement de nos projets de fin d'études master.

Enfin, je remercie du fond du cœur toutes les personnes qui ont cru en moi, m'ont soutenue et accompagnée tout au long de ce parcours. Que ce travail soit le témoignage de ma profonde reconnaissance envers chacun d'entre eux.

ملخص

تسعى المدن الذكية اليوم إلى إيجاد طريقة أفضل لإدارة مواقف السيارات الحضرية عبر الإنترنت، لضمان الكشف الفوري عن الأماكن المتاحة، وتأمين الوصول، والحد من الازدحام المروري. ويمكن حل هذه المشكلات باستخدام تقنية إنترنت الأشياء (IoT).
يقدم هذا المشروع نظامًا ذكيًا لمواقف السيارات قائمًا على إنترنت الأشياء، يستخدم أجهزة استشعار متصلة للكشف عن أماكن وقوف السيارات وإدارتها.
يستخدم هذا النظام الآلي أجهزة استشعار تعمل بالأشعة تحت الحمراء وفوق الصوتية للكشف عن وجود المركبات، ليحل محل الإدارة اليدوية التقليدية.
يهدف هذا المشروع إلى ابتكار جهاز ذكي لمواقف السيارات يتيح الحجز الفوري، والدفع، وإدارة الوصول بين تطبيق ويب ووحدة تحكم دقيقة ESP32، مع مراقبة حالة موقف السيارات والتحكم بها في الوقت الفعلي. ويتم ضمان التواصل بين هذه الأجهزة عبر بروتوكول HTTP وشبكة Wi-Fi. وقد طورنا نموذجًا أوليًا عمليًا. وتم اختبار نظامنا والتحقق من صحته باستخدام طرق وصول مختلفة: تطبيق ويب، وبطاقة RFID، وتذكرة رمز الاستجابة السريعة (QR).

الكلمات المفتاحية: إنترنت الأشياء، ESP32، مواقف السيارات الذكية، RFID، أجهزة الاستشعار، تطبيق الويب، Wi-Fi.

Résumé

De nos jours, les villes intelligentes cherchent à trouver un meilleur moyen pour gérer le stationnement urbain via internet, qui garantit une détection en temps réel des places disponibles, assure la sécurité des accès et élimine les problèmes de congestion du trafic. Ces problèmes pourraient être résolus en utilisant la technologie basée sur l'Internet des Objets (IoT).
Ce projet présente un système de stationnement intelligent basé sur l'IoT qui utilise des capteurs connectés comme support de détection et de gestion des places de parking.
Ce dernier est un système automatisé dans lequel des capteurs infrarouges et ultrasoniques sont utilisés pour détecter la présence des véhicules au lieu d'une gestion manuelle traditionnelle.
Le but de ce projet est de créer un dispositif de stationnement intelligent qui permet la réservation, le paiement et la gestion des accès instantanément, entre une application web et un microcontrôleur ESP32, afin de surveiller et contrôler l'état du parking en temps réel.
La communication entre ces appareils est assurée via le protocole HTTP et le Wi-Fi. Nous avons réalisé un prototype fonctionnel. Notre système a été testé et validé en utilisant différents modes d'accès : application web, carte RFID et ticket QR.

Mots clés : IoT, ESP32, Smart Parking, RFID, capteurs, application web, Wi-Fi.

Abstract

Today, smart cities are seeking a better way to manage urban parking via the internet, ensuring real-time detection of available spaces, secure access, and eliminating traffic congestion. These problems could be solved using Internet of Things (IoT) technology.
This project presents an IoT-based smart parking system that uses connected sensors to detect and manage parking spaces.
This automated system uses infrared and ultrasonic sensors to detect vehicle presence, replacing traditional manual management.
The goal of this project is to create a smart parking device that enables instant reservation, payment, and access management between a web application and an ESP32 microcontroller, monitoring and controlling the parking lot's status in real time. Communication between these devices is ensured via HTTP and Wi-Fi. We have developed a functional prototype. Our system has been tested and validated using different access methods: web application, RFID card, and QR code ticket.

Keywords: IoT, ESP32, Smart Parking, RFID, sensors, web application, Wi-Fi.

Liste des Acronymes

IoT	Internet of Things
NFC	Near Field Communication
ACH	Automated Clearing House
BNPL	Buy Now Pay Later
CIB	Carte Interbancaire
RFID	Radio Frequency Identification
PGS	Parking Guidance System
LCD	Liquid Crystal Display
Wi-Fi	Wireless Fidelity
LAN	Local Area Network
MQTT	Message Queuing Telemetry Transport
HTTP	HyperText Transfer Protocol
API	Application Programming Interface
GPIO	General Purpose Input Output
I2C	Inter-Integrated Circuit
SDA	Serial Data Line
SCL	Serial Clock Line
PWM	Pulse Width Modulation
NPM	Node Package Manager
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
SQL	Structured Query Language
JSON	JavaScript Object Notation

TCP/IP	Transmission Control Protocol / Internet Protocol
SPI	Serial Peripheral Interface
MOSI	Master Out Slave In
MISO	Master In Slave Out
SCLK	Serial Clock
CS	Chip Select
VCC	Voltage Common Collector
SSID	Service Set Identifier
TLS	Transport Layer Security
SSL	Secure Sockets Layer

Liste des figures

Chapitre I : Généralités sur les systèmes de stationnement intelligents basés sur IoT

Chapitre II : Architecture matérielle du projet et développement du système associé

Chapitre III : Développement logiciel du projet réalisé et application web

Chapitre VI : Résultats et discussion

Liste des tableaux

Chapitre I : Généralités sur les systèmes de stationnement intelligents basés sur IoT

Chapitre II : Architecture matérielle du projet et développement du système associé

Chapitre III : Développement logiciel du projet réalisé et application web

Table de matières

Introduction Générale

Avec l'évolution rapide des technologies numériques et l'augmentation du nombre de véhicules dans les zones urbaines, la gestion du stationnement est devenue un enjeu important pour les villes modernes. La recherche d'une place libre provoque souvent une perte de temps, une consommation excessive de carburant, une augmentation de la pollution et une congestion du trafic routier. Face à ces problèmes, les systèmes de stationnement intelligents représentent une solution efficace permettant d'améliorer la mobilité urbaine et d'optimiser l'utilisation des espaces de stationnement.[1]

Parmi ces technologies, l'Internet des Objets (IoT, En anglais *Internet of Things*) occupe une place de choix en raison de sa capacité à connecter des objets physiques afin de collecter, transmettre et traiter des données en temps réel. Cette technologie repose sur l'intégration de capteurs intelligents, de microcontrôleurs et de réseaux de communication, ce qui permet d'automatiser de nombreuses tâches et de réduire l'intervention humaine dans des domaines d'application variés. Son essor a ainsi favorisé le développement d'applications telles que les maisons connectées, l'agriculture de précision, les systèmes de surveillance, la gestion énergétique intelligente ou encore les systèmes de transport intelligents. Appliqué au stationnement, l'IoT permet notamment de surveiller l'occupation des places en temps réel, de faciliter la réservation à distance, d'automatiser l'accès aux parkings et, plus largement, d'améliorer l'expérience des usagers tout en optimisant l'utilisation des espaces disponibles.[2]

Dans ce contexte, ce travail porte sur la conception et la réalisation d'un prototype de système de stationnement intelligent basé sur l'IoT. Ce système repose sur une carte ESP32 associée à plusieurs capteurs et actionneurs assurant la détection des véhicules, le contrôle automatique des barrières d'accès ainsi que la gestion des places de stationnement. Il est complété par une application web permettant aux utilisateurs de consulter l'état du parking, d'effectuer des réservations, de gérer leurs abonnements RFID et de réaliser leurs paiements en ligne.

Le choix de la carte ESP32 comme plateforme de développement se justifie par plusieurs avantages : sa capacité de communication Wi-Fi ou Bluetooth intégrée, qui permet une connexion directe avec l'application web ; son faible coût et sa facilité de programmation ; sa compatibilité avec un large éventail de capteurs et de modules électroniques ; sa faible consommation énergétique ; ainsi que des performances bien adaptées aux applications IoT fonctionnant en temps réel.

Le système ainsi développé intègre plusieurs fonctionnalités complémentaires : la détection automatique de l'occupation des places de stationnement, la réservation des places via l'application web, le contrôle automatique des barrières d'entrée et de sortie, la gestion des abonnements à l'aide de cartes RFID, le paiement électronique avec consultation de l'historique des opérations, ainsi que la supervision et l'administration centralisées du parking.

Pour atteindre ces objectifs, le cahier des charges de ce projet s'articule autour de quatre axes principaux :

- Premièrement, d'étudier les technologies nécessaires à la réalisation d'un système de stationnement intelligent, notamment l'Internet des Objets, les capteurs de détection, la technologie RFID, les réseaux Wi-Fi ainsi que les protocoles de communication associés.
- Le deuxième axe consiste à concevoir et à développer une architecture matérielle et logicielle permettant la gestion automatisée d'un parking intelligent.
- Le troisième vise à réaliser une application web assurant la réservation des places, la gestion des utilisateurs, des paiements et des abonnements.
- Enfin, le quatrième axe porte sur le test et la validation du fonctionnement global du système à travers un prototype fonctionnel.

Ce mémoire est organisé en quatre chapitres :

- Dans le premier chapitre nous avons présenté les généralités sur les systèmes de stationnement intelligents ainsi que les concepts fondamentaux relatifs à l'Internet des Objets et aux technologies mobilisées dans ce travail.
- Le deuxième chapitre est consacré à l'étude et à la conception de l'architecture matérielle du système ; il décrit les différents composants utilisés, parmi lesquels la carte ESP32, les capteurs infrarouges et ultrasoniques, le module RFID, les servomoteurs et l'afficheur LCD.
- Le troisième chapitre porte sur le développement logiciel du système réalisé, en détaillant les outils de développement employés, l'architecture de l'application web, la communication entre l'ESP32 et le serveur, ainsi que les principaux programmes embarqués conçus.
- Le quatrième chapitre est dédié à la réalisation pratique du prototype, aux tests effectués, aux résultats obtenus et à leur analyse critique.

Enfin, nous concluons notre mémoire par une conclusion générale en incluant des recommandations pour les projets futurs.

CHAPITRE I

Généralités sur les systèmes de stationnement intelligents basés sur IOT

I.1. Introduction

Ce chapitre vise à présenter les généralités sur les systèmes de stationnement intelligents basés sur l'Internet des objets (IoT en anglais pour Internet of Things). Nous allons introduire les concepts fondamentaux de l'IoT, la mobilité intelligente, ainsi que les différents types de parkings intelligents. Nous présenterons également les architectures, les fonctionnalités, les systèmes de paiement et de guidage, ainsi que les avantages et les inconvénients offerts par ces solutions modernes dans le cadre des villes intelligentes.

I.2. Internet des objets (IoT)

I.2.1. Définition

L'Internet des objets fait référence au réseau d'objets physiques connectés à Internet à l'aide de logiciels, de capteurs, et d'autres technologies. Cela permet à ces appareils « intelligents » de collecter et d'échanger des données, facilitant la communication et l'automatisation. L'IoT s'étend au-delà des ordinateurs et des smartphones pour englober des objets du quotidien tels que des appareils électroménagers, des véhicules et même des vêtements. Cette interconnexion permet une surveillance, un contrôle et une analyse à distance, favorisant l'efficacité, la commodité et l'innovation dans divers secteurs, des maisons intelligentes aux soins de santé, en passant par la fabrication et le transport [3].

I.2.2. Domaines d'application :

L'IoT est utilisée dans plusieurs domaines à savoir [4]:

- Maison intelligente : L'IoT permet l'automatisation des équipements domestiques comme l'éclairage, le chauffage, les caméras de surveillance, et les appareils électroménagers connectés.
- Ville intelligente : Elle contribue à la gestion intelligente des infrastructures urbaines telles que l'éclairage public, la circulation routière, et la gestion des déchets.
- Agriculture de précision : Les capteurs IoT sont utilisés pour surveiller l'humidité du sol, la température, l'irrigation, et l'état des cultures afin d'optimiser la production agricole.
- Santé : L'IoT facilite le suivi médical à distance grâce aux objets connectés capables de surveiller les paramètres de santé des patients en temps réel.
- Industrie : Dans l'industrie, l'IoT permet l'autorisation des chaînes de production, la maintenance prédictive et le contrôle des machines à distance.
- Transport : Elle est utilisée dans la gestion du trafic routier, la localisation des véhicules, les systèmes de navigation, et les parkings intelligents afin d'améliorer la mobilité et la sécurité.

La Figure 1.1 ci-dessous résume les applications les plus répandues de l'IoT.

Figure 1.1 : Quelques domaines d'application des IoT .

I.3. Mobilité intelligente

La mobilité intelligente redéfinit les territoires en influençant leur attractivité économique et leur qualité environnementale, un enjeu crucial tant pour l'introduction de nouvelles solutions que pour l'optimisation des espaces publics par les gestionnaires. Elle se matérialise par une intégration holistique des besoins à travers des systèmes connectés. Des dispositifs de contrôle d'accès, tels que les lecteurs de plaques d'immatriculation, régulent l'entrée aux parkings fermés tout en diffusant des informations en temps réel.

Des panneaux dynamiques orientent vers les places de stationnement disponibles, tandis qu'une application mobile centralisée informe sur les options de transport, calcule des itinéraires optimaux en fonction du coût et du temps, et permet la réservation de moyens de transport en libre-service. La mobilité intelligente s'adapte ainsi aux usages et aux innovations, favorisant ainsi son expansion continue.[5]

Figure 1.2 : La mobilité intelligente [15].

I.4. Généralité sur le stationnement intelligent

I.4.1. Définition de stationnement

Le stationnement intelligent, ou le Smart Parking, est une application de technologies modernes qui permet d'éliminer beaucoup de besoins en peu de temps, et ceci est limité à la facilité de circulation et à la sécurité routière, aux places de réservation et aux différents moyens de paiement. Le principe consiste à équiper chaque place de stationnement d'un capteur intelligent capable de détecter la présence d'un véhicule et d'informer en temps réel que la place est libre ou occupée.

Le capteur est complètement autonome et ne nécessite donc aucune infrastructure à proximité, ce qui réduit les coûts d'investissement et surtout de maintenance. Il s'installe directement dans la chaussée, au centre de chaque place de stationnement, en moins de 10 minutes. Parmi ces technologies, il est possible de réserver et payer de différentes façons, de manière intelligente et parmi ces méthodes, cartes à puce (smart cardes), cartes sans contact, cartes de crédit/débit….etc. Ces technologies rendent le stationnement plus attrayant et populaire, encourageant le développement et le paiement. [6]

Figure 1.3 : Le stationnement intelligent [16].

I.4.2. Types de stationnement

- ✓ **Payant :** Soumis à un horodateur ou nécessitant un ticket.
- ✓ **Gratuit à durée limitée :** Souvent matérialisé par des marquages au sol bleus et l'usage d'un disque de stationnement (zone bleue).
- ✓ **Unilatéral alterné :** Le stationnement change de côté de la rue selon la période du mois (souvent du côté impair du 1er au 15, et pair du 16 à la fin du mois).
- ✓ **Résidentiel :** Réservé aux riverains disposant d'un macaron ou abonnement spécifique.
- ✓ **Interdit ou réglementé :** Stationnement interdit devant les bouches d'incendie, sur les passages piétons, les trottoirs, ou les emplacements réservés (livraison, personnes handicapées).

I.4.3. Définition de parking intelligent

Le parking intelligent est l'une des initiatives de ville intelligente les plus populaires à l'heure actuelle. Il s'appuie sur les données recueillies à partir d'une variété de capteurs pour mieux gérer les services, les ressources et les actifs de la ville afin d'améliorer la qualité de vie des citoyens. La technologie de stationnement intelligent répond à l'une des plus grandes frustrations des citoyens car elle aide les conducteurs à trouver des places de stationnement facilement et rapidement grâce à l'application. Elle permet aussi le transfert des informations aux responsables du stationnement pour identifier les infractions et d'inciter les gens à utiliser d'autres options de transport en cas de stationnement encombré.

Les initiatives de stationnement intelligent sont souvent combinées à la technologie des rues intelligentes telles que les feux de circulation intelligents et les lampadaires intelligents qui améliorent la sécurité, réduisent les temps de transport et réduisent la congestion du trafic.[7]

I.4.4. Types de parking

a) Classification selon l'emplacement

- **Parking en surface :** Situé à l'air libre, de plain-pied, souvent gratuit ou réglementé en voirie.
- **Parking souterrain :** Situé sous des immeubles ou des centres commerçants. Il est couvert et sécurisé.
- **Parking silo (aérien) :** Bâtiment construit en élévation à l'extérieur dédié uniquement au stationnement sur plusieurs étages. [8]

b) Classification selon l'accès et l'usage

- **Parking public :** librement accessible à tous les usagers de la route, souvent géré par la collectivité ou payant.
- **Parking privé :** réservé à l'usage exclusif d'un particulier, d'une copropriété ou d'une entreprise (nécessite souvent un badge).
- **Parking privé à usage public :** l'accès n'est toléré ou ouvert qu'à certains clients (ex: parkings de supermarchés).
- **Parc relais (P+R) :** situé aux entrées de ville pour inciter les usagers à utiliser les transports en commun. [8]

c) Comparaison entre le parking intelligent et le parking traditionnel

Le Tableau 1.1 récapitule les différences entre le parking intelligent et le parking traditionnel en termes de fonctionnement, utilisation de la technologie, fluidité de circulation, système de paiement, et coût d'installation.

Tableau 1.1 : Comparaison entre parking intelligent et parking traditionnel.

Critères	Parking traditionnel	Parking intelligent
Définition	Espace classique de stationnement où les véhicules se garent manuellement	Système moderne utilisant les technologies intelligentes pour gérer le parking.
Fonctionnement	Gestion manuelle qui souvent désorganisé	Gestion automatisée et optimiser en temps réel
Trouver une place	Prend beaucoup de temps	Recherche rapide grâce aux informations en temps réel
Utilisation de la technologie	Très faible ou inexistence	Utilisation de capteurs, caméras, applications mobiles et technologies IoT
Fluidité de circulation	Risque fréquent de congestion et de circulation désordonnée	Circulation plus fluide grâce à une meilleure gestion des flux de véhicules
Système de paiement	Paiement généralement manuel	Intégration de solutions de paiement électronique, abonnement mensuel et annuelle
Coût d'installation	Coût de mise en œuvre relativement faible	Coût plus élevé en raison de l'intégration des équipements technologiques avancés

I.5. Catégories des systèmes de stationnement intelligents

On distingue deux grandes catégories de systèmes de stationnement intelligents : les systèmes de guidage et d'information, et les systèmes de paiement intelligent.

I.5.1. Systèmes de guidage et d'information

I.5.1.1. Définition de système de guidage

Un système de guidage au stationnement (PGS en anglais pour Parking Guidance System) est une solution technologique conçue pour aider les conducteurs à trouver rapidement des places de stationnement disponibles dans un parking ou une zone désignée. Il vise à optimiser l'utilisation des espaces de stationnement et à améliorer l'efficacité globale des opérations de stationnement.

I.5.1.2. Composants d'un système de guidage

a) Capteurs de stationnement

Ces capteurs, souvent installés sur des places de stationnement individuelles ou à des endroits stratégiques du parking, détectent la présence de véhicules et déterminent si une place de stationnement est occupée ou vacante. Les capteurs fournissent ces données en temps réel.[9]

b) Système de gestion centralisé

Le système de guidage au stationnement est géré de manière centralisée par un logiciel qui collecte, traite et analyse les données des capteurs de stationnement. Il surveille l'état d'occupation de chaque place de stationnement, calcule la disponibilité du stationnement, et transmet ces informations aux conducteurs via la signalisation, les applications mobiles ou d'autres systèmes d'affichage.[9]

c) Applications mobiles et intégration

De nombreux systèmes de guidage au stationnement proposent des applications mobiles qui permettent aux conducteurs d'accéder à des informations de stationnement en temps réel, notamment la disponibilité, les itinéraires et la navigation. Le système peut également s'intégrer à d'autres applications ou systèmes, tels que des plateformes de paiement ou des systèmes de réservation, pour améliorer l'expérience globale de stationnement.[9]

I.5.2. Systèmes de paiement intelligent

Le paiement intelligent désigne des méthodes de paiement numérique utilisant des technologies avancées (NFC, biométrie, block Chain) pour réaliser des transactions rapides, sécurisées, et pratiques. Il inclut les portefeuilles mobiles, les paiements sans contact, et les crypto-monnaies. Ces systèmes améliorent la commodité et la sécurité en simplifiant le processus de paiement en réduisant les frais et les délais grâce à la suppression des intermédiaires et à l'automatisation des transactions.

I.5.2.1. Types de paiement intelligent

a) Paiement en ligne [10]

Les paiements en ligne désignent les transactions effectuées sur Internet pour régler des biens ou des services, qu'ils aient été achetés en ligne ou en magasin. Ces paiements sont traités via des plateformes sécurisées et peuvent être réalisés à l'aide de différentes méthodes, notamment :

- **Transactions par carte de crédit ou de débit :** Paiements effectués en entrant les détails de la carte dans une passerelle de paiement en ligne.
- **Portefeuilles numériques :** Services tels que PayPal, Google Pay ou Apple Pay qui permettent aux utilisateurs de stocker des informations de paiement pour des transactions rapides.
- **Virements bancaires :** Également connus sous le nom de virements télégraphiques, ils permettent d'envoyer des fonds directement entre des comptes bancaires.
- **Paiements par débit direct ou ACH :** Paiements récurrents automatisés débités directement d'un compte bancaire.
- **Services "Achetez maintenant, payez plus tard" (BNPL** en anglais pour **Buy Now Pay Later) :** Les options de paiement telles que Klarna ou Afterpay permettent aux clients de payer en plusieurs tranches, souvent sans frais.

Le paiement en ligne présente plusieurs avantages notamment [11] :

- **Disponibilité 24h/24 et 7j/7** sans dépendre des horaires d'ouverture des agences.

- **Sécurité et traçabilité** sans aucun risque de transporter de fortes sommes d'argent liquide. Chaque transaction par carte CIB ou Edahabia nécessite une authentification stricte (par exemple, via un code de confirmation reçu par SMS).
- **Gain de temps** en évitant les files d'attente.
- **Autonomie**

b) Paiement par horodateur

Un horodateur est un dispositif électromécanique permettant aux automobilistes de payer pour stationner leurs véhicules sur une place de parking payante à durée limitée. Ce type de dispositif se trouve uniquement dans les zones avec des places de stationnement payant à durée limitée, principalement dans les agglomérations. Pour réguler le stationnement en milieu urbain, de nombreux pays ont choisi d'installer des horodateurs. Ils constituent généralement un élément clé du système de gestion du stationnement soutenu par les pouvoirs publics. Dans le cadre des politiques écologiques, certains horodateurs sont alimentés par des panneaux photovoltaïques, permettant aux communes de réaliser des économies d'énergie significatives. La Figure 1.4 montre un exemple d'un horodateur.[12]

Figure 1.4 : Un horodateur [17].

Le payement par horodateur présente les avantages suivants :

- **Suivi du temps de stationnement :** Ils permettent aux autorités de savoir depuis combien de temps un véhicule est stationné à un emplacement donné.
- **Fourniture d'informations utiles :** Les horodateurs affichent diverses informations telles que le logo de l'agglomération concernée et la zone de stationnement.
- **Utilisation d'énergie verte :** Les horodateurs sont souvent alimentés par des panneaux photovoltaïques, utilisant ainsi de l'énergie renouvelable.

I.6. Fonctionnalités d'un parking intelligent

❖ **Détection automatique de véhicules :** grâce à l'utilisation des différents types de capteur (capteur ultrason, capteur IR), le système détecte automatiquement la présence ou l'absence de véhicule.

- **Surveillance de place de stationnement :** chaque place de stationnement contient un capteur spécifique qui détecte le stationnement de véhicule et envoie un signal au microcontrôleur pour actualiser l'état de parking.
- **Intégration de l'internet des objets :** cette technologie améliore nettement la communication entre les capteurs et le centre de contrôle pour une gestion plus réactive et plus précise.
- **Gestion des places disponibles :** à l'aide d'une application mobile ou un afficheur convenable à l'entrée de parking, le client peut connaitre tous les places disponibles et occupées.
- **Commande automatique de barrière :** après l'affectation correct de réservation ou paiement, le microcontrôleur envoie le signal au servomoteur pour ouvrir ou fermer la barrière.
- **Réservation à distance :** via une application web ou Android, les conducteurs peuvent faire une pré-réservation ainsi que le paiement via la carte Edhahabia ou CIB. Les clients peuvent donc réserver des places de stationnement en ligne, effectuer des paiements électroniques, et recevoir un code QR pour accéder automatiquement au parking.
- **Identification des utilisateurs via RFID :** l'utilisateur peuvent effectuer un abonnement mensuel ou annuel lui permettant d'accéder au parking avec sa carte RFID.
- **Gestion du stationnement par application web :** Un tableau de bord web permet aux administrateurs de surveiller les places de stationnement, de gérer les utilisateurs et les abonnements, et d'analyser efficacement les statistiques et les revenus.
- **Gestion en temps réel du stationnement :** le système intelligent installé met à jour instantanément la disponibilité des places et affiche le nombre de places libres/occupée en temps réel afin d'améliorer la fluidité du trafic et de réduire les embouteillages.

I.7. Avantages et inconvénients de stationnement intelligent

- **Les avantages :**
  - Obtenir des informations précises sur les emplacements occupés ou non occupés en temps réel.
  - Augmenter l'activité et se déplacer plus librement dans la ville en utilisant les technologies modernes.
  - Assurer la sécurité du trafic pour les conducteurs et les utilisateurs.
  - Profiter du temps de la recherche et de l'espace libre pour stationnement
  - Réduire la pollution et l'utilisation de l'essence et l'émission de gaz toxiques.
  - Améliorer la surveillance et la gestion en temps réel de l'espace de stationnement disponible, ce qui entraîne une génération de revenus significative.
  - Fournir un service excellent aux touristes.
  - Etablir des décisions intelligentes à l'aide de données, notamment des applications d'état en temps réel et des rapports d'analyse historique.
  - Travailler sur la communication d'informations aux usagers avant, pendant et après de stationnement. [13]

- **Les inconvénients :**
  - **Frais de déploiement :** l'installation des capteurs au sol, des caméras et des bornes de connexion nécessite un investissement massif de la part des municipalités ou des propriétaires de parkings

- **Pannes du système :** une coupure de réseau, un défaut logiciel ou une panne de capteur peuvent fausser les données, indiquant par erreur qu'une place est libre.
- Exclusion : Les personnes âgées ou moins à l'aise avec la technologie peuvent se retrouver pénalisées si le système ne propose pas d'alternative physique (comme un ticket ou un horodateur classique).
- Risque de piratage : les serveurs centralisés et les applications mobiles constituent des cibles potentielles pour les cyberattaques, posant des risques de fuite de données confidentielles. [13]

I.8. Défis des parkings intelligents

Les villes intelligentes utilisent les nouvelles technologies pour offrir des solutions pratiques aux utilisateurs, notamment à travers le stationnement connecté, qui doit relever plusieurs défis[14] :

- **Réduction de la pollution :** Contribuer à diminuer les émissions polluantes.
- **Amélioration du confort des usagers :** Fournir une expérience agréable pour fidéliser les clients.
- **Fluidité du trafic :** Réduire la congestion et améliorer la circulation.
- **Sécurité :** Assurer la surveillance des véhicules et des conducteurs contre le vol et les squatteurs.
- **Optimisation de l'occupation :** Gérer efficacement le taux d'occupation des places de stationnement.

I.9. Conclusion

Dans ce chapitre, nous avons présenté les concepts fondamentaux liés aux systèmes de stationnement intelligents basés sur l'Internet des objets. Nous avons également présenté les différents types de parkings, leurs fonctionnalités, leurs systèmes de paiement et de guidage, ainsi que leurs avantages et limites. Cette étude théorique constitue une base essentielle pour la conception et la réalisation du système de parking intelligent proposé dans ce mémoire conformément au cahier de charge élaboré au début de ce projet startup.

CHAPITRE II

Architecture matérielle du projet et développement du système associé

II.1. Introduction

Ce chapitre a pour objectif de présenter l'analyse structurelle du projet réalisé. Nous décrivons tout d'abord en détail le schéma fonctionnel du système de stationnement intelligent conçu, ainsi que les différents composants et équipements utilisés, leurs caractéristiques, leurs rôles et leurs interconnexions. Enfin, nous présentons le schéma électronique complet adopté pour la construction du prototype.

II.2. Schéma synoptique du système de parking intelligent réalisé

II.2.1. Architecture matérielle

Le système réalisé repose principalement sur une carte à microcontrôleur telle que l'ESP32, connectée à différents capteurs et modules intelligents. À l'entrée et à la sortie du parking, des capteurs spécifiques, à ultrasons en l'occurrence, détectent la présence des véhicules afin de déclencher automatiquement les différentes actions du système. Lorsqu'un véhicule arrive à l'entrée, le système propose plusieurs modes d'accès, notamment via une plateforme web qui permet la pré-réservation, une carte RFID pour les abonnés soit mensuelle ou annuel, ou encore un ticket contenant un code QR. Chaque méthode d'authentification est vérifiée par le système avant d'autoriser l'accès. En cas de validation, la barrière s'ouvre automatiquement. Chaque place de stationnement contient un capteur infrarouge (IR). Une fois le véhicule stationné, le compteur de temps démarre automatiquement et le système met à jour en temps réel l'état du parking grâce à la détection des places occupées. Ces informations sont affichées sur un écran LCD 16x2 I2C et synchronisées avec une application web permettant une supervision à distance. Une fois le véhicule sort de la zone de stationnement, le compteur de temps s'arrête automatiquement et l'information est envoyée au microcontrôleur.

À la sortie, un second processus de détection est activé. Le système vérifie l'autorisation de sortie via RFID, paiement électronique, ou ticket QR. Après validation, la barrière s'ouvre et la place est automatiquement libérée dans le système. La Figure 2.1 présente le schéma synoptique du système de parking intelligent réalisé.

II.2.2. Description de chaque bloc du système

a) Couche capteur et équipements terrain

Cette couche représente l'infrastructure physique du parking et assure la collecte des données et l'interaction directe avec les véhicules.

- Détection de la disponibilité des places de stationnement : Des capteurs IR sont utilisés au-dessus de chaque place pour déterminer si elle est occupée ou libre. Des caméras peuvent également être utilisées pour l'analyse des images.
- Gestion des accès : Des capteurs à ultrason détectent la présence d'un véhicule à l'entrée, des servomoteurs ou des moteurs à courant continu contrôlent l'ouverture et la fermeture des barrières automatiques, en association avec des lecteurs de cartes RFID.

- Panneaux d'affichage dynamiques (PAD) : Des écrans LCD I2C sont placés aux entrées et à chaque étage pour afficher en temps réel l'état de parking (par exemple : « Étage 1 :place1 disponible, place2 occupé ….. »).

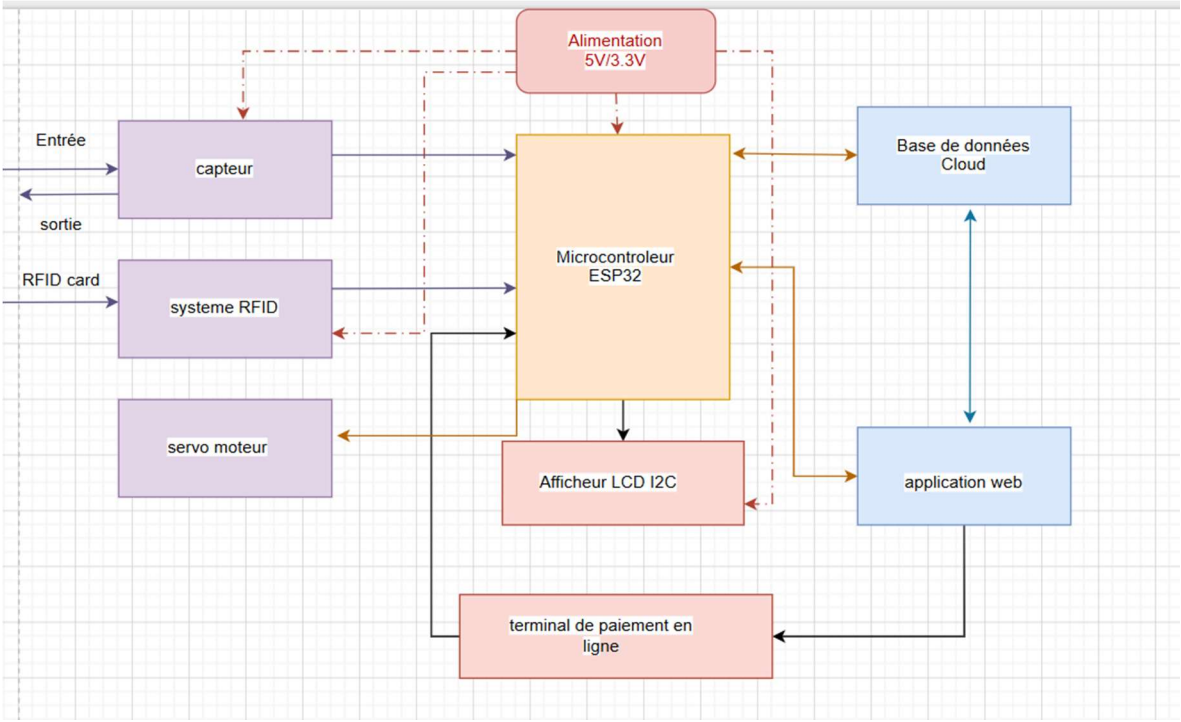

Figure 2.1 : Schéma synoptique du système de stationnement intelligent réalisé.

- Terminaux de paiement en libre-service : Des bornes de paiement permettent aux utilisateurs de payer en espèces, par carte bancaire ou avec leurs tickets de sortie ; ainsi que le paiement par abonnement mensuel à l'aide d'une carte RFID.

b) Couche communication et passerelles

Cette couche assure la communication entre les équipements physiques et les systèmes logiciels basés sur le Cloud, facilitant ainsi un transfert de données rapide et sécurisé.

- Passerelle : Un microcontrôleur puissant ou un petit ordinateur industriel (tel que ESP32 ou un Raspberry Pi configuré comme passerelle) collecte les données de toutes les composantes du système.
- Connexion locale : Protocoles et réseaux internes pour la transmission des données des capteurs à la passerelle, tels que Wi-Fi, Bluetooth, Ethernet (LAN) ou liaisons filaires, selon la distance.
- Connexion au Cloud : Une connexion Internet sécurisée (via un modem 4G/5G, la fibre optique ou le Wi-Fi) utilisant des protocoles légers et rapides tels que MQTT ou HTTP REST assure la transmission régulière de l'état de la situation à un serveur central.

c) Couche plateforme Cloud et logiciel

Cette couche représente le « cerveau » et le système dorsal qui traite d'importants volumes de données et gère l'intelligence globale du projet.

- Base de données centrale : Elle permet le stockage en temps réel d'historique du trafic des véhicules (entrées et sorties), des données utilisateurs, et de la gestion du système de tarification.

- Moteur de réservation et de paiement : Il s'agit d'interfaces de programmation d'applications (API) pour le traitement des paiements en ligne, la vérification des billets numériques, et la confirmation des pré-réservations.

d) Couche utilisateurs et opérateurs

Cette interface permet aux utilisateurs d'interagir avec le système et d'utiliser ses services. Dans notre projet, il s'agit de développer une application web pour les clients. C'est une sorte d'interface permettant de consulter l'état de parking, de rechercher des places disponibles de pré-réserver des places, de payer directement par téléphone mobile, et aussi d'être guidé jusqu'à l'emplacement désigné.

II.2.3. Connexions entre les différents composants

La connexion entre les différents composants du système s'effectue via plusieurs interfaces de transmission de données. Le module RFID communique avec l'ESP32 par le protocole SPI pour transmettre les informations d'identification des cartes RFID.

L'écran LCD quant à lui utilise le protocole I2C pour afficher les données envoyées par la carte ESP32, tout en réduisant le nombre de connexions nécessaires.

Les capteurs infrarouges et ultrasoniques transmettent leurs signaux directement aux broches d'entrée de l'ESP32. Après avoir traité les informations reçues, l'ESP32 commande le servomoteur pour gérer la barrière d'accès. Enfin, grâce à la connexion Wi-Fi intégrée, l'ESP32 échange des données avec l'application web afin de synchroniser l'état du parking et de permettre la consultation des informations, pré-réservation, ainsi que le paiement en ligne via le téléphone portable.

II.3. Choix d'équipements et de composants

II.3.1. Carte à microcontrôleur

Nous avons choisi d'utiliser le module ESP32 comme microcontrôleur en raison de ses caractéristiques et fonctionnalités très intéressantes.

L'ESP32 est une puce Wi-Fi et Bluetooth économique d'Espressif Systems. Elle intègre les solutions Wi-Fi (bande 2,4 GHz) et Bluetooth 4.2 sur une seule puce. Elle prend également en charge le Bluetooth classique pour les connexions existantes, notamment les profils L2CAP, SDP, GAP, SMP, AVDTP, AVCTP, A2DP (SNK) et AVRCP (CT). L'ESP32 est également compatible avec le Bluetooth Low Energy (BLE), qui couvre les profils L2CAP, GAP, GATT, SMP et les profils basés sur GATT.

L'ESP32 existe sous deux formes : puce et module. Ces deux formes diffèrent par leur taille et le nombre de broches. Le choix de la forme dépend de la conception et des objectifs du projet à développer. La taille peut également être un critère de choix lors de la conception d'une solution IoT, notamment pour le circuit imprimé [18]. La Figure 2.2 représente une carte ESP32 à 38 pins.

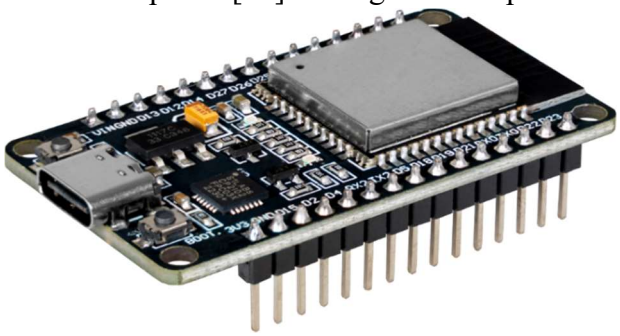

Figure 2.2 : La carte ESP32 à 38 broches [25].

Les caractéristiques principales de la carte ESP32 sont les suivantes [19]:

1. Microcontrôleur : Elle utilise un microcontrôleur à processeur à double-cœur Xtensa LX6 32 bits cadencé jusqu'à 240 MHz.

2. Mémoire : La carte dispose de mémoire flash entre 4 MB et 16 MB selon les modèles et 520 Ko de RAM pour les instructions et les données.

3. Broches d'E/S : Elle offre 23 Entrées/sorties numériques GPIO, et 18 Entrées/sorties analogiques GPIO.

4. Interfaces : La carte possède :

- Une interface à connecteur micro USB (type-C) pour la connexion à un ordinateur et pour l'alimentation externe de 2.2 V à 3.6 V, et des broches d'alimentation 3.3V.
- Bluetooth Low Energy (BLE, BT4.0, Bluetooth Smart) intégré.
- Puce Wifi 2.4 GHz (802.11 b/g/n) avec antenne intégrée.

5. Compatibilité : Le module ESP32 est compatible avec la plupart des shields Arduino.

6. Dimensions : Ses dimensions sont d'environ 51 mm x 25.4 mm.

7. Logiciel : La carte est programmable avec le logiciel Arduino IDE, compatible avec les systèmes d'exploitation Windows, macOS et Linux.

Les pins de la carte ESP32 utilisée dans notre projet sont illustrés sur la Figure 2.3.

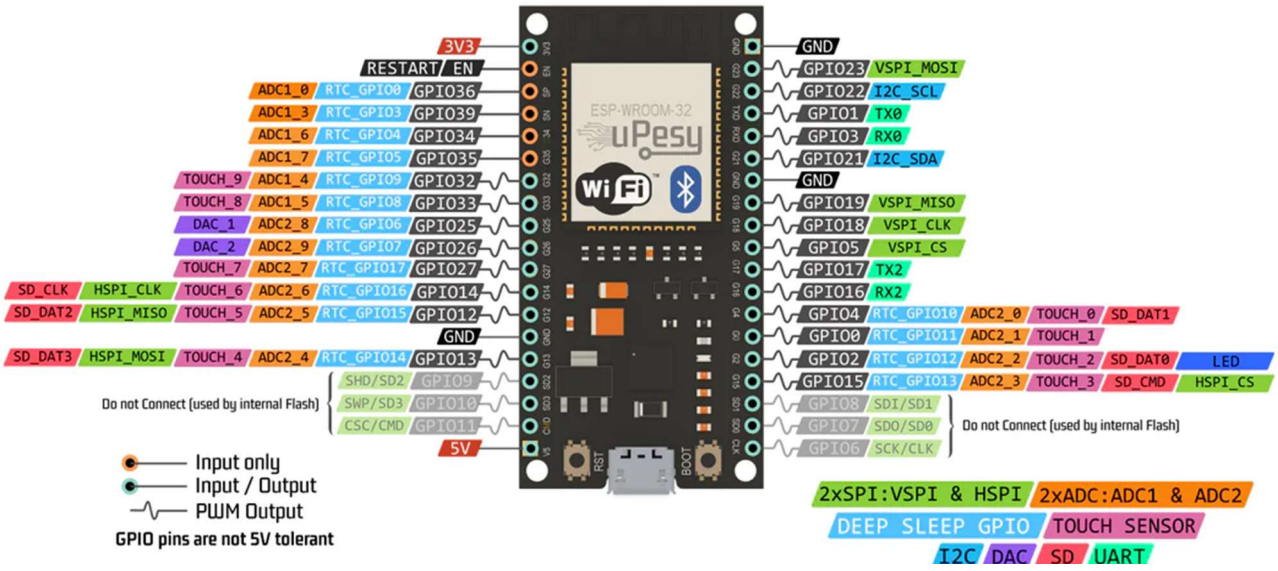

Figure 2.3 : Les pins de la carte ESP32 utilisée dans le projet [26].

II.3.2. Capteurs utilisés

II.3.2.1. Capteurs infra-rouges

Un capteur de proximité infrarouge (Figure 2.4) détecte la présence d'objets ou de personnes sans contact physique direct en émettant un faisceau de lumière infrarouge et en détectant les variations de ce faisceau réfléchi. Il est composé d'une LED infrarouge, d'un phototransistor, et comporte trois broches : VCC pour l'alimentation, GND pour la mise à la terre en plus de la sortie du signal.

Couramment utilisé dans l'automatisation des portes, les systèmes de sécurité et les robots mobiles, ce capteur est essentiel pour sa détection rapide et fiable [20].

Figure 2.4 : Capteur infra-rouge [27].

Le fonctionnement du capteur repose sur deux composants clés : une LED infrarouge (IR) et un phototransistor. La LED émet un rayonnement à une longueur d'onde de 920 nanomètres, invisible à l'œil humain.

Lorsque ce rayonnement est bloqué par un obstacle, il indique la présence de celui-ci. Le phototransistor, agissant comme un intercepteur, réceptionne le rayonnement réfléchi. Deux situations se présentent [21]:

a) Si aucun rayonnement n'est réfléchi (équivalent à zéro), cela signifie qu'aucun objet n'est présent dans le périmètre, et le transistor reste bloqué, agissant comme un interrupteur ouvert.
b) Si le rayonnement réfléchi est supérieur ou égal à un seuil prédéfini, le transistor devient conducteur, activant ainsi un interrupteur équivalent à un interrupteur fermé.

La Figure 2.5 schématise le fonctionnement du capteur infrarouge

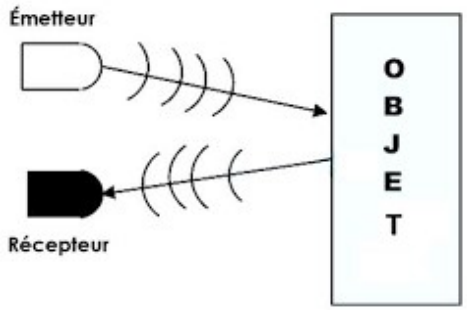

Figure 2.5 : Principe de fonctionnement d'un capteur IR [28].

II.3.2.2. Capteur à ultrasons (HC-SR04)

Les capteurs ultrasoniques (Figure 2.6) permettent la d´détection des emplacements de stationnement. Ces capteurs sont fixés au-dessus de chaque espace de stationnement. Ils fonctionnent en fonction de la localisation de l'´écho. Le temps entre l'impulsion envoyée et l'´écho renvoyé est utilisé pour calculer la distance. Dans un espace vacant, le temps entre le son transmis et la réflexion est plus long que dans un espace occupé [21].

Figure 2.6 : Capteur Ultrasonique (HC-SR04) [29].

Le principe de fonctionnement illustré sur la Figure 2.7 est comme suit [22] :

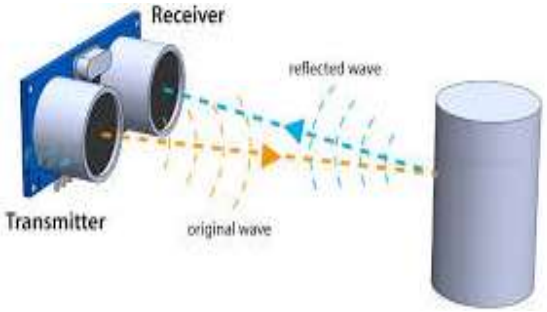

Figure 2.7 : Principe de fonctionnement du capteur à ultrasons HC-SR04 [30].

a) Un signal de 10 µs est envoyé sur la broche Trigger pour émettre des ultrasons à 40 kHz via la capsule émettrice du module. Ce signal est généré via une sortie digitale d'une carte compatible.
b) La seconde capsule, installée sur le module, permet de réceptionner les ultrasons retournés par l'objet. Cette information est envoyée vers une E/S digitale de la carte microcontrôleur via la broche Echo

II.3.3. Module RFID (Radio Frequency Identification)

La technologie RFID, ou Radio-Frequency Identification, est un système de communication sans fil qui permet de collecter et d'enregistrer des données à distance. Elle utilise des étiquettes électroniques, appelées tags RFID, qui contiennent une puce électronique et une antenne permettant de transmettre des informations. La Figure2.8 ci-dessous présente le module RFID MFRC522.

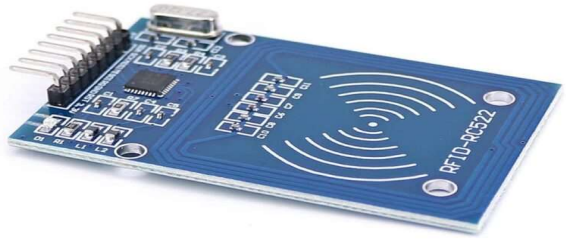

Figure 2.8: Module RFID MFRC522 [31].

Le principe de fonctionnement d'un module RFID est illustré sur la Figure 2.9 et comprend les éléments suivants [23] :

- Le système RFID permet l'identification d'étiquettes codées appelées TAG ou encore TRANSPONDEUR (TRANSmitter/resPONDER). Ces Tags "intelligents" sont constitués d'une puce électronique et d'une antenne émission/réception. Un émetteur envoie une onde radio de fréquence plus ou moins élevée.
- L'énergie rayonnée est suffisamment importante pour alimenter l'étiquette (passive), qui va dès lors envoyer de la même façon un code d'identification numérique. L'interaction entre les champs magnétiques émis et reçus permet au récepteur de décoder la trame émise par le Tag RFID.
- La distance de détection va de quelques centimètres jusque quelques mètres, en fonction de la fréquence utilisée, ce qui permet des identifications éloignées. La plupart du temps les Tags sont passifs et ne peuvent qu'émettre un code unique. Leur durée de vie est pratiquement illimitée, cependant, il existe des tags actifs, alimentés par pile, qui peuvent recevoir et stocker des données.

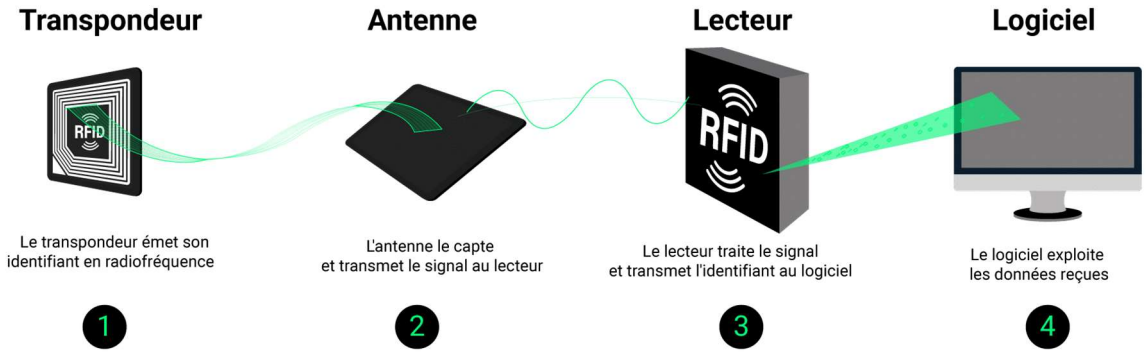

Figure 2.9 : Principe de fonctionnement d'un module RFID [32].

II.3.4. Afficheur LCD (Liquid Crystal Display)

Un écran **LCD I2C** est un afficheur à cristaux liquides (LCD) équipé d'un module de communication **I2C (Inter-Integrated Circuit)** permettant de communiquer avec un microcontrôleur comme Arduino ou Espressif Systems en utilisant seulement deux lignes de données **: SDA** (données) et **SCL** (horloge).

Figure 2.10: Afficheur LCD I2C.[33]

Le module I2C convertit les données série envoyées par le microcontrôleur vers le LCD. Ainsi, au lieu d'utiliser plusieurs broches comme avec un écran LCD classique, seules les broches suivantes sont utilisées :

- **VCC** : alimentation
- **GND** : masse
- **SDA** : données
- **SCL** : horloge

La Figure 2.11 montre les pins de branchement d'un afficheur LCD I2C.

Figure 2.11 : Branchement d'un afficheur type LCD I2C [34].

II.3.5. Système de servomoteur

Un servomoteur, comme celui illustré sur la Figure 2.11, est un dispositif électromécanique qui utilise un signal de modulation de largeur d'impulsion (MLI) ou PWM en anglais pour Pulse Width Modulation, et un système de rétroaction pour contrôler avec précision la position de son axe de rotation. Il se compose d'un moteur, d'un capteur de position, d'un circuit de commande, et d'un microcontrôleur. Le signal PWM détermine la position désirée de l'axe, et le microcontrôleur ajuste le moteur en conséquence, utilisant les données du capteur pour corriger toute différence entre la position réelle et la position souhaitée [24].

Figure 2.12: Le servomoteur SG09 [35].

II.4. Câblages et interconnexions

❖ **Bloc d'entrée:**

- Capteur d'entrée : à l'entrée on a relié un capteur ultrasonique (HC-SR04) avec un ESP32.
- Servomoteur : un servomoteur qui commande la barrière d'entrée est aussi relié à la carte à microcontrôleur ESP32.

Les connexions et le câblage correspondants de ces deux éléments sont illustrés respectivement dans le Tableau 2.1 et la Figure 2.13.

Tableau 2.1 : Connexions du capteur à ultrasons et du servomoteur avec l'ESP32.

VCC	5V
TRIG	GPIO17
ECHO	GPIO16
GND	GND
IN de servomoteur	GPIO13

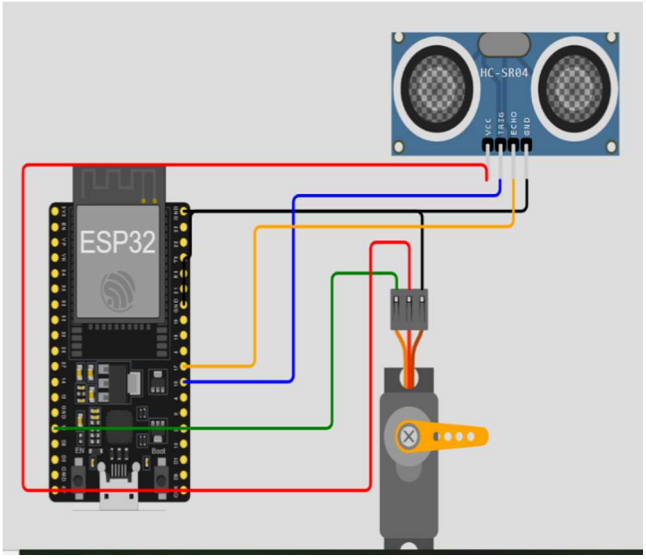

Figure 2.13 : Le câblage du capteur à ultrasons et du servomoteur avec l'ESP32.

- **Afficheur LCD :** Les connexions de l'écran LCD I2C avec l'ESP32 sont présentées dans le Tableau 2.2 tandis que le câblage est illustré dans la Figure 2.14.

Tableau 2.2 : Connexions de l'afficheur LCD I2C avec la carte à microcontrôleur.

VCC	5V
SDA	GPIO21
SCL	GPIO22
GND	GND

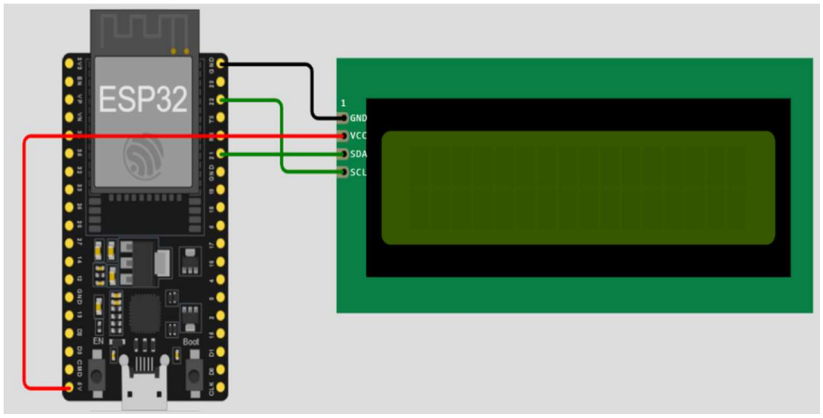

Figure 2.14 : Câblage de l'afficheur LCD avec la carte à microcontrôleur.

- **Module RFID (MFRC522):** Le câblage du module MFRC522 avec l'ESP32 est présenté dans la Figure 2.15 et les connexions sont illustrées dans le Tableau 2.3.

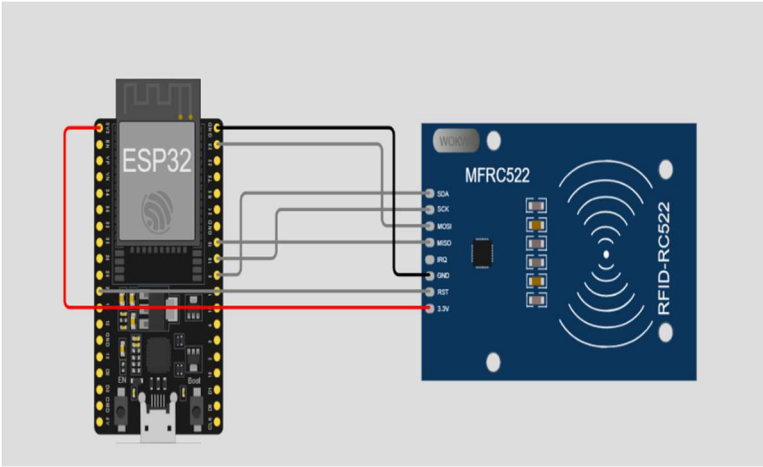

Figure 2.15 : Câblage du lecteur RFID à l'ESP32.

Tableau 2.3 : Connexions du lecteur RFID avec l'ESP32.

SDA	GPIO5
SCK	GPIO18
MOSI	GPIO23
MISO	GPIO19
3.3V	3.3V
RST	GPIO27
GND	GND

- **Capteurs infrarouges :** Dans notre prototype, il y a quatre places de parking, ce qui nécessite quatre capteurs infrarouges. Le câblage de chaque capteur au microcontrôleur est illustré dans la Figure 2.16 et les connexions correspondantes sont présentées dans le Tableau 2.4.

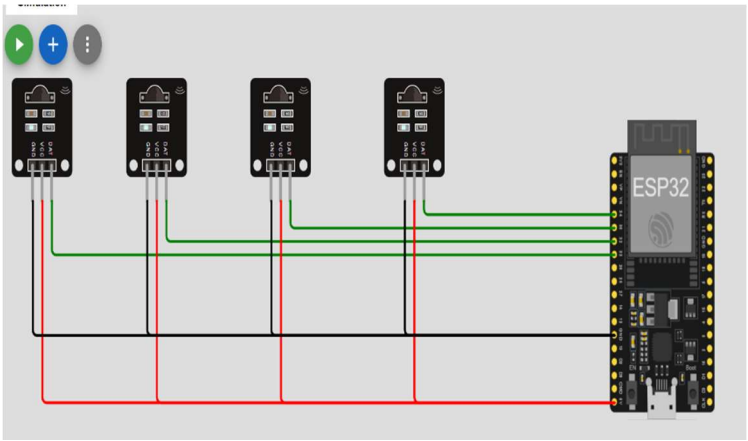

Figure 2.16 : Câblages des capteurs IR.

Tableau 2.4 : Connexions des capteurs IR au microcontrôleur.

	Capteur 1	**Capteur 2**	**Capteur 3**	**Capteur 4**
5V	**5V**	**5V**	**5V**	**5V**
OUT	GPIO32	GPIO33	GPIO34	GPIO35
GND	GND	GND	GND	GND

❖ **Bloc de sortie :**

- Capteur de sortie : A la sortie, nous relions un capteur ultrason de type HCSR04 avec l'ESP32.
- Servomoteur : on relier un servomoteur qui commande la barrière de sortie avec le microcontrôleur, le tableau ce dessus montre le câblage de HCSR04 ainsi que le servomoteur.

Les connexions et le câblage correspondants de ces deux éléments sont illustrés respectivement dans le Tableau 2.5 et la Figure 2.17.

Tableau 2.5 : Connexions du capteur HCSR04 et du servomoteur avec l'ESP32.

5V	5V
TRIG	GPIO25
ECHO	GPIO26
GND	GND
IN de servomoteur	GPIO12

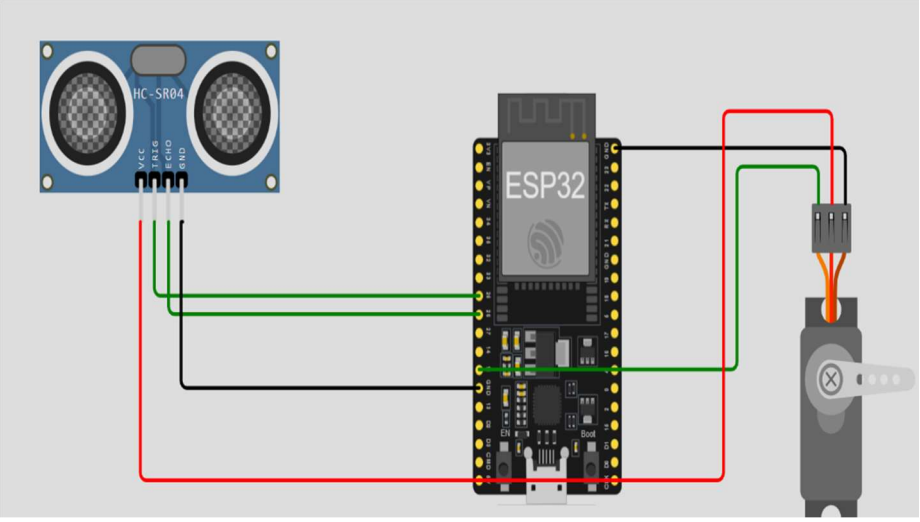

Figure 2.17 : Le câblage du bloc de sortie avec l'ESP32.

II.5. Schéma électrique

Le schéma électrique été réalisé à l'aide du logiciel Proteus 9.1 pour but de vérifier la cohérence des raccordements et d'assurer le bon fonctionnement du système avant sa mise en œuvre pratique.

L'objectif principal est de représenter l'ensemble de connexions entre les différents composants matériels utilisé dans notre système, tel que l'ESP32, l'afficheur LCD I2C, les capteurs IR, les capteurs à ultrasons, ainsi que les deux servomoteurs d'entrée et de sortie et le lecteur de carte RFID qu'on a simulé avec un terminal virtuel. Cette représentation facilite la compréhension du fonctionnement du système ainsi que la vérification des différentes interconnexions entre les composants constituant le prototype. La Figure 2.18 représente le schéma électrique complet du système de parking intelligent conçu.

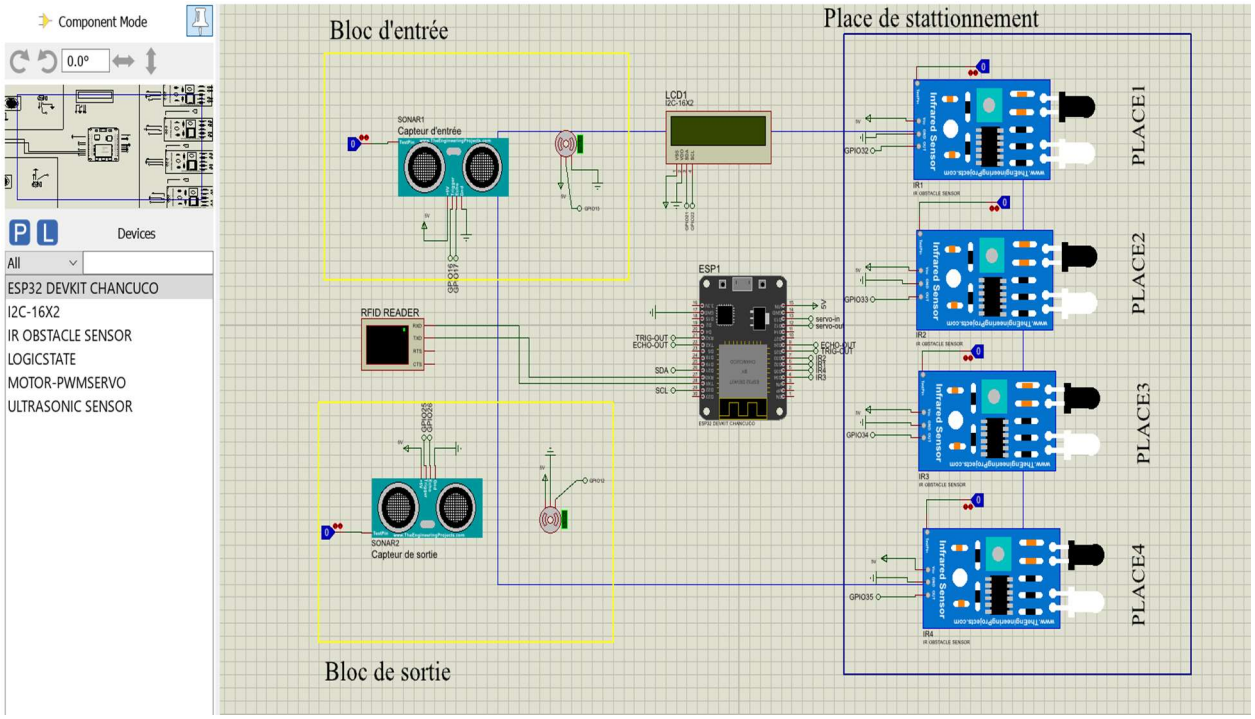

Figure 2.18 : Schéma électrique du système de parking intelligent réalisé.

II.6. Conclusion

Dans ce chapitre, nous avons présenté l'architecture matérielle du système de stationnement intelligent conçu ainsi que les principaux composants utilisés, notamment l'ESP32, les capteurs IR et HC-SR04, le module RFID, l'afficheur LCD I2C, et les servomoteurs. Nous avons également décrit leurs caractéristiques, leurs rôles, et leurs interconnexions au sein du système. Enfin, le schéma électrique réalisé sous Proteus a permis de valider le câblage et le bon fonctionnement du prototype avant sa mise en œuvre.

Le chapitre suivant sera consacré à l'analyse algorithmique et à la conception logicielle de notre projet, permettant la mise en œuvre pratique du système conformément aux spécifications exprimées dans le cahier de charges

CHAPITRE III

Développement logiciel du projet réalisé et application web

III.1. Introduction

Ce chapitre vise à présenter la phase développement logiciel système, après avoir étudié le développement matériel dans le chapitre précèdent, il est nécessaire d'étudie les différentes technologies et outils de développement utilisé.

Dans ce chapitre nous allons discutons sur les différents environnements de développement et logiciel notamment Arduino IDE qui est utilisé pour la programmation de carte ESP32 ainsi que les outils employé pour le développement de l'application web.

Ensuite l'architecture globale de l'application web sera décrite en mettant en évidence les différentes couches qui la composent : l'interface utilisateur (Frontend), le serveur web (Backend) et la base de données.

Il expose également le mécanisme de communication entre l'ESP32 et l'application web à travers le protocole HTTP et l'échange de données au format JSON. Enfin, les principaux programmes embarqués développés pour la gestion des différents composants du système (Wi-Fi, capteurs (IR et ultrasonique), RFID, les barrières d'entrées et de sortie, affichage LCD, gestion des entrées et sorties seront détaillés à l'aide d'extraits de code et d'organigrammes de fonctionnement.

III.2. Outils de développement utilisés

III.2.1. Arduino IDE

➢ *Définition*

Les créateurs d'Arduino ont développé un logiciel pour la programmation des différentes cartes ESP32, ESP86, Arduino UNO, etc. Arduino est un logiciel open source, permettant de programmer toute carte Arduino et propose une variété de fonctionnalités.

Le langage Arduino s'inspire de plusieurs langages de programmation, à savoir C, C^{+}..., et impose une structure particulière typique de l'informatique embarquée. Le programme est exécuté par le microcontrôleur de haut en bas. Une variable doit être déclarée avant d'être utilisée par une fonction [36].

➢ *Présentation de l'Arduino IDE*

Arduino IDE est un environnement de développement intégré (Integrated Development Enivrement) souple et facile à utiliser. Un programme Arduino IDE se compose de trois parties principales : la déclaration des variables, la fonction de configuration setup(), et la boucle principale loop(). La Figure 3.1 représente l'interface d'Arduino IDE.

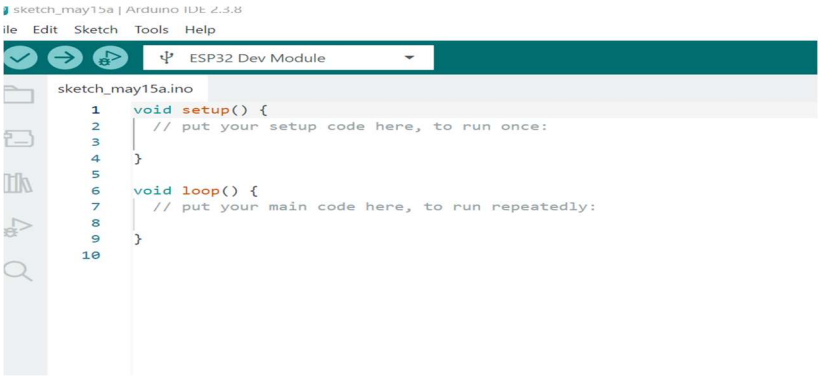

Figure 3.1 : Interface d'Arduino IDE.

- **Déclaration de variables :** permet de définir les variables utilisées es dans les programmes (en-tête du code).

- **Fonction setup () :** elle sert à initialiser les composants, configurer les broches en entrées ou en sortie, et démarrer la communication série.
- **Fonction loop () :** Cette partie est lue en boucle et contient les fonctions principales à réaliser en continu.

➢ *Étapes de programmation sur Arduino*

1. Choisir le type de carte à programmer ainsi que le port (Figure 3.2) :

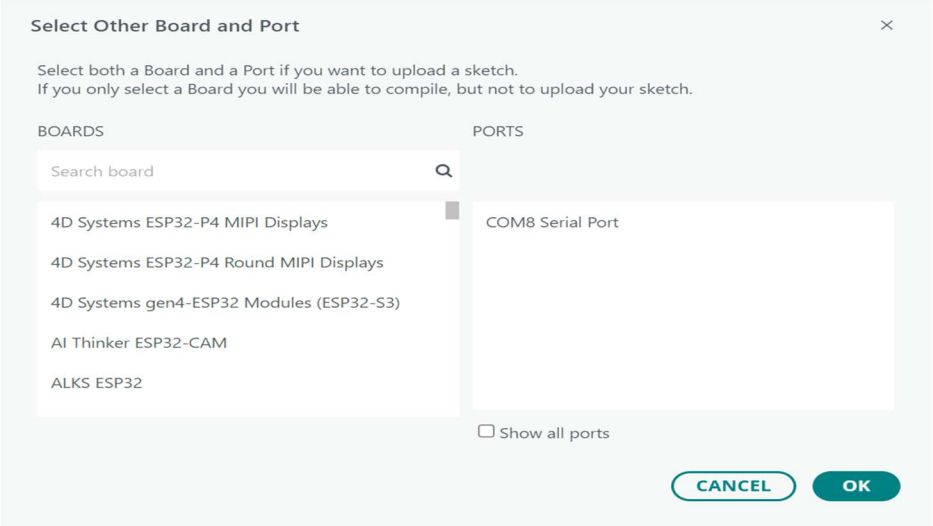

Figure 3.2 : Configuration de la carte ESP32 dans Arduino IDE.

2. Télécharger les bibliothèques du Library manager (Figure 3.3):

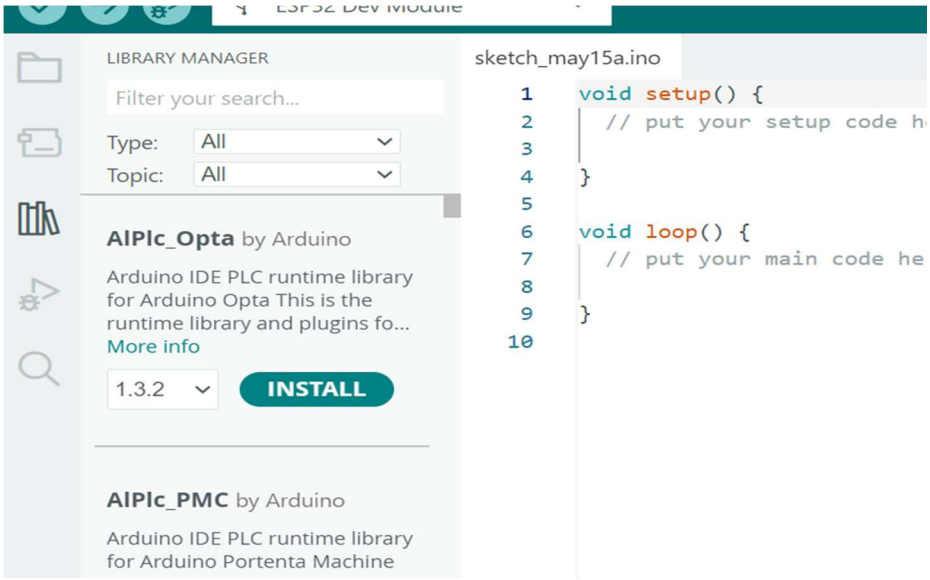

Figure 3.3 : Installation des bibliothèques nécessaires au développement du système dans Arduino IDE.

3. Écrire le programme dans la fenêtre d'édition de code.
4. Compiler le code pour vérifier s'il n'y a pas d'erreurs. Les erreurs éventuelles sont affichées dans la fenêtre de console.

5. Si le code est correct, l'envoyer vers la mémoire flash de l'Arduino. L'exécution démarre immédiatement après le téléversement.
6. Surveiller les informations et les erreurs à chaque étape sur la console.

III.2.2. Développement de l'application web

III.2.2.1. Poste de travail et de configuration

Le développement de l'application web du système de parking intelligent a été réalisé sur un ordinateur fonctionnant sous Windows 10 Professionnel 64 bits. Cet environnement a permis la conception de l'interface utilisateur, le développement du serveur web, la gestion de la base de données, ainsi que les différents tests de communication avec la carte ESP32. La configuration logicielle utilisée est présentée dans le Tableau 3.1 suivant.

Tableau 3.1 : Configuration logicielle du poste de développement.

Composants	***Outils***	***Rôle***
OS	Windows 10 Pro (64 bits)	Système d'exploitation
Navigateur	Google Chrome	Test et débogage de l'application web
Terminal	PowerShell	Exécution des commandes Node.js et npm

III.2.2.2. Editeur de code

Visual Studio Code (VS Code) a été choisi comme environnement principal de développement de l'application web. Cet outil offre une interface moderne et de nombreuses fonctionnalités facilitant la création d'applications web.

Les principales raisons de ce choix sont [37]:

- Sa légèreté et sa rapidité d'exécution.
- Son large catalogue d'extensions.
- Son terminal intégré permettant l'exécution des commandes Node.js et npm.
- Ses outils de débogage intégrés.
- Son intégration native avec Git pour le contrôle de versions.

III.2.2.3. Outils complémentaires utilisés

Le développement web repose sur un ensemble d'outils complémentaires : les éditeurs de code pour écrire les instructions, les Frameworks pour accélérer la création, le contrôle de version pour collaborer, et les outils de test. Le Tableau 3.2 ci-dessous rassemble les différents outils utilisés.

Tableau 3.2 : Outils complémentaires utilisés pour le développement web.

Outil/logiciel	***Rôle***
Visual Studio Code	Éditeur principal : rédaction de tout le code (HTML, CSS, JS, SQL)
Node.js	Environnement d'exécution du serveur backend (Express.js)
Npm	Gestionnaire de paquets : installation des dépendances backend
MySQL Server	Serveur de base de données relationnelle
MySQL Workbench	Interface graphique pour conception et administration de la BDD
Live Server (extension)	Serveur de développement frontend avec rechargement automatique
Postman	Test manual des routes API REST (POST, GET, PUT)
Git	Contrôle de version et sauvegarde du code source
Navigateur Chrome	Test et débogage de l'interface (DevTools, Network, Console)
Nodemon	Redémarrage automatique du serveur Node.js lors des modifications

III.3. Architecture de l'application web

III.3.1. Les composants de l'architecture

III.3.1.1. Interface utilisateurs (frontend)

La couche de présentation est construite en HTML, CSS et JavaScript purs, sans Framework comme React ou Vue. La Figure 3.4 ci-dessous représente l'interface d'authentification de l'application web *Smart Park*.

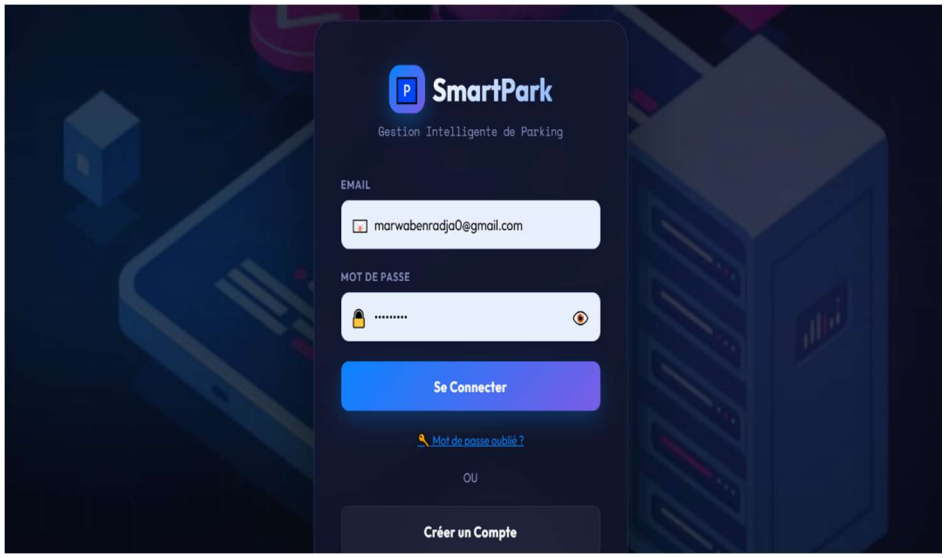

Figure 3.4 : L'interface de l'application web *Smart-Park*.

Ces principales fonctionnalités sont :

- Authentification à l'application Smart-Park.
- Consultation de l'état de parking.
- Réservation et libération d'une place.
- Renouvellement d'abonnement de cartes RFID.
- Paiement en ligne via carte Edhahabia ou paypal.

La Figure 3.5 illustre le tableau de bord principal de l'application web développée.

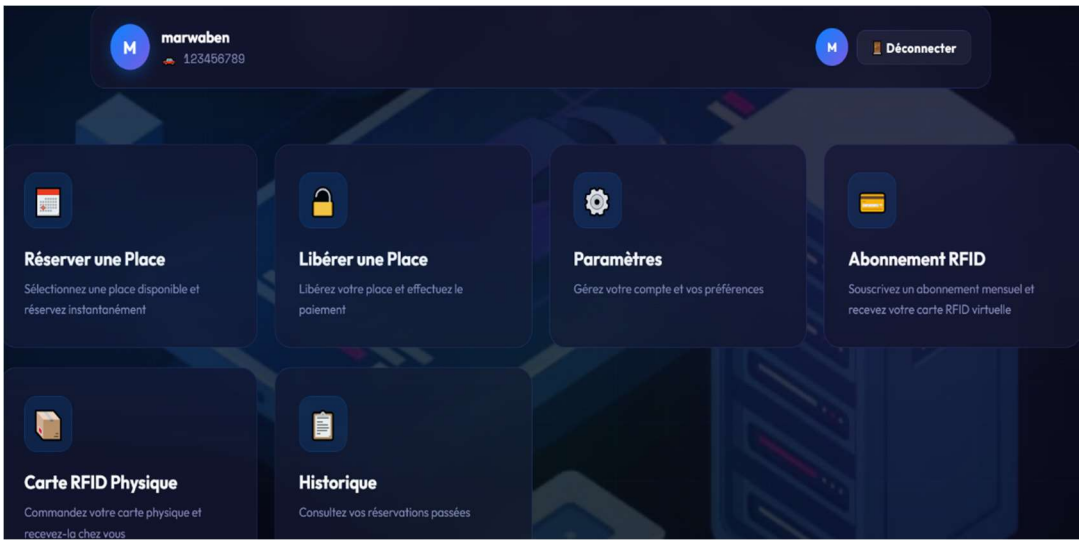

Figure 3.5 : Tableau de bord principal de l'application web *Smart Park*.

Chaque fonctionnalité de ce tableau de bord contient plusieurs opérations comme suite :

- **Réservation d'une place (Figure 3.6):** cette fonctionnalité permet aux utilisateurs de consulter l'état de parking (occupé en rouge, libre en vert), ainsi que sélectionner une place (en bleu) puis confirmer la réservation. Après validation, la réservation est enregistrée par le système et associée au compte de l'utilisateur.

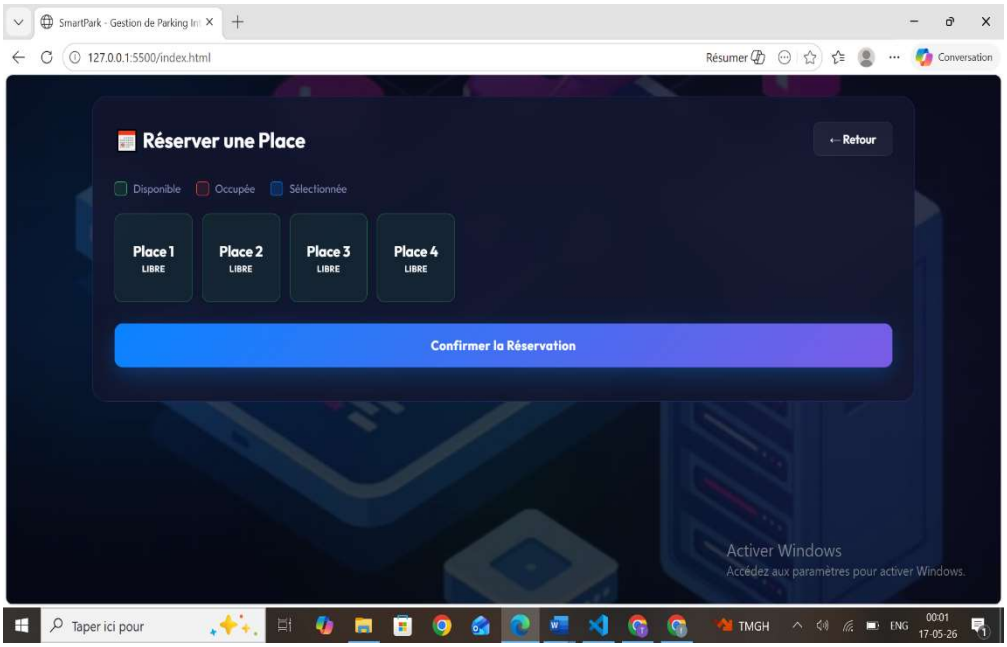

Figure 3.6 : Interface de réservation d'une place.

- **Libérer une place :** cette interface permet aux utilisateurs de consulter les informations relatives à leurs stationnements (la durée, le montant à payer). Elle offre donc la possibilité de libérer une place occupée (Figure 3.7), et de choisir le mode de paiement souhaité afin de régler les frais de stationnement avant la validation de la sortie (Figure 3.8).

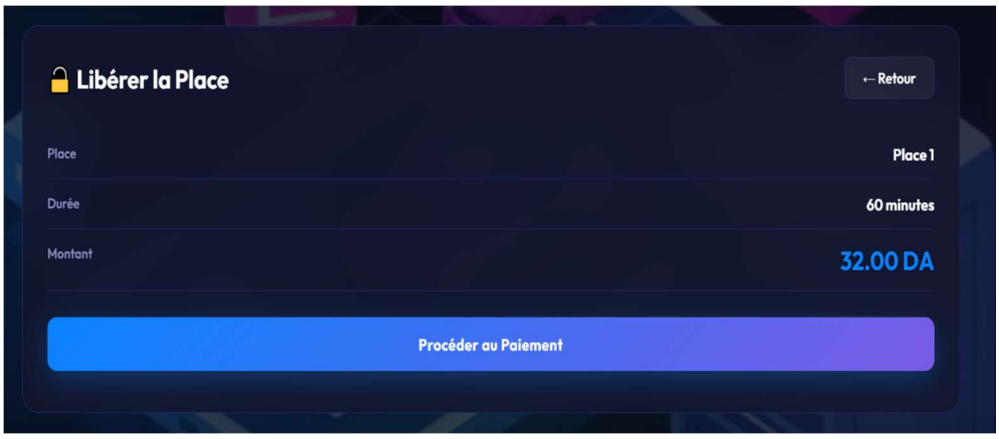

Figure 3.7 : Interface de libération d'une place.

Figure 3.8 : Modes de paiement disponibles sur l'interface de libération.

Il existe trois modes de paiement :

- Via application web : l'utilisateur peut effectuer le paiement en ligne à l'aide d'une carte Edhahabia ou une carte PayPal.
- Via carte RFID : ce mode est destiné aux abonnées. Chaque abonné dispose d'un identifiant unique associe à sa carte RFID ce qui permet l'accès et la gestion automatique de son abonnement.
- Via ticket : lors de son entrée au parking l'utilisateur reçoit un ticket contient les informations relatives de son stationnement telles que PIN unique heure d'entrée ainsi que la place attribue, à la sortie après la vérification et règlement de frais de stationnement l'administrateur peut autoriser l'ouverture de la barrière.

- **Abonnement RFID (Figure 3.9):** cette interface permet aux utilisateurs de de renouveler leurs abonnements annuels ou mensuels et payer en ligne avec Edhahabia ou PayPal.

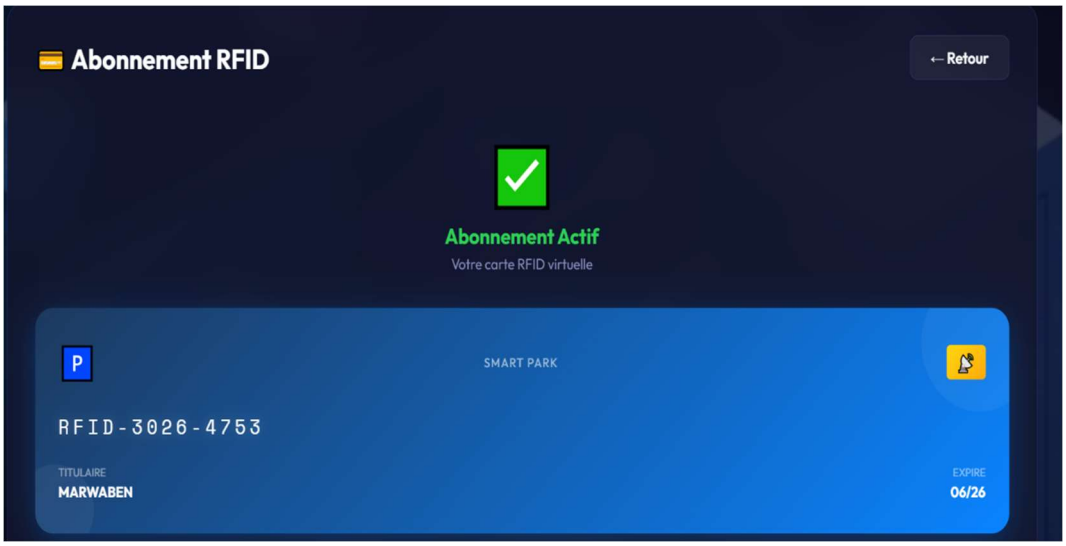

Figure 3.9 : Interface d'abonnement RFID.

- **Historique :** L'interface de la Figure 3.10 permet aux utilisateurs de consulter leurs historiques contenant toutes leurs opérations sur la plateforme.

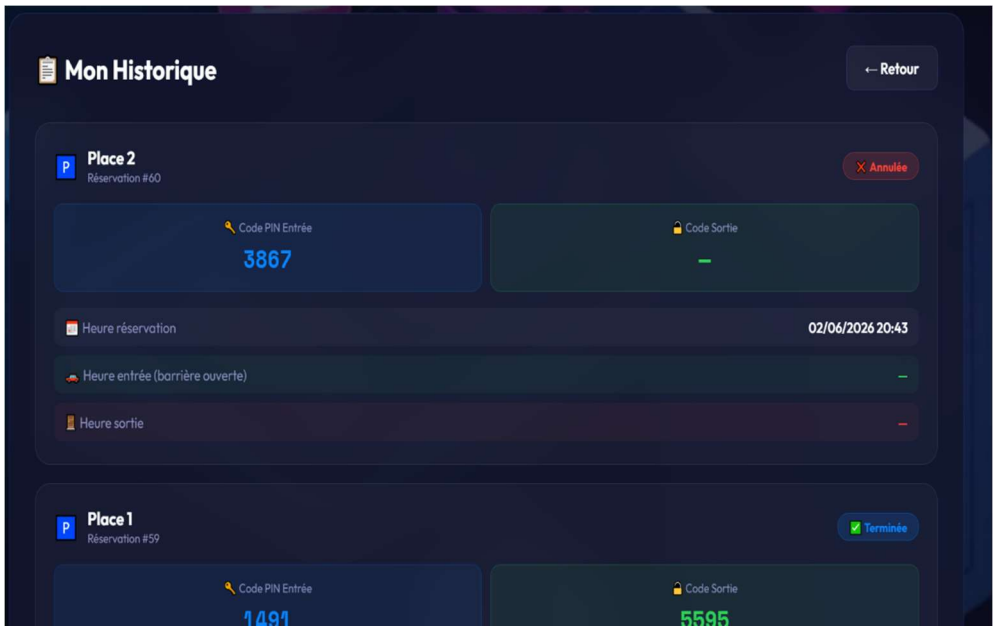

Figure 3.10 : Interface d'historique de clients.

- **Demande de carte physique (Figure 3.11):** cette interface permet aux utilisateurs de demander une carte RFID physique et de choisir le mode de récupération de la carte (livraison ou au bureau). L'utilisateur reçoit la carte dans 48 heures après la demande, et reçoit un email au moment d'expédition.

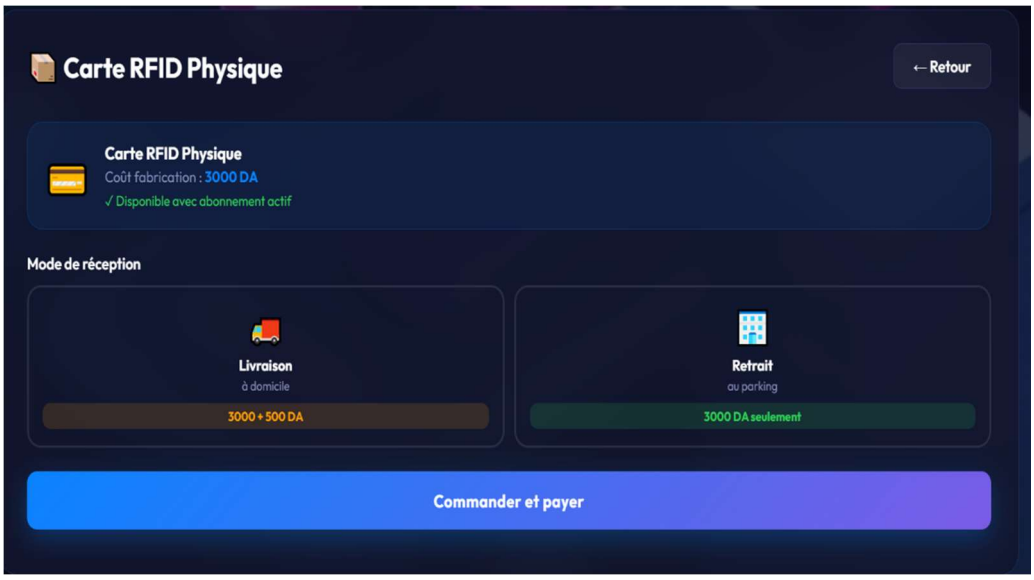

Figure 3.11 : Interface de demande de carte RFID physique.

- **Espace ADMIN (Figure 3.12):** cette interface concerne exclusivement les administrateurs qui peuvent consulter toutes les opérations relatives au parking telles que les réservations, les utilisateurs, les paiements, les demandes des cartes FRID, et les renouvellements d'abonnements.

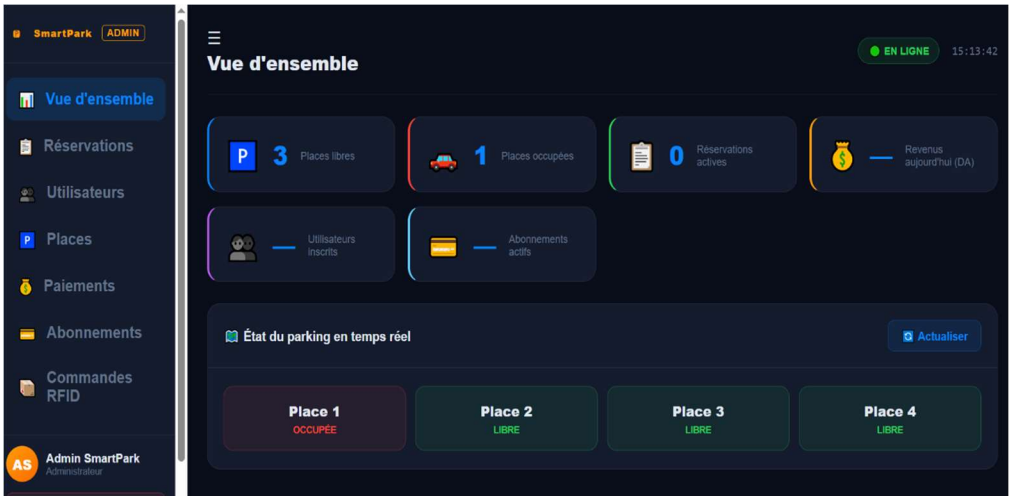

Figure 3.12 : Interface ADMIN de l'application *Smart Parking*.

III.3.1.2. Serveur web (Backend)

Le backend constitue la partie serveur de l'application. Il est responsable du traitement provenant de l'interface web, de la gestion logique métier, ainsi que la communication avec la base de données.

Dans notre projet, le serveur backend est développé avec Node.js (environnement d'exécution JavaScript côté serveur) et le Framework Express.js. Cette combinaison a été retenue pour sa légèreté, sa rapidité de mise en œuvre, et son excellente adéquation avec les API REST. Le Tableau 3.3 ci-dessous décrit les différents API utilisés dans notre système.

III.3.1.3. Base de données

La base de données SmartPark a été développée avec un système de gestion des bases de données MySQL utilisant un moteur de stockage InooDB. Elle est considérée comme le noyau central du système de parking intelligent réalisé en assurant le stockage, l'organisation, et la gestion des données nécessaire pour le fonctionnement correct de l'application. La Figure 3.13 ci-dessous représente la structure de la base de données SmartPark sous phpMyAdmin.

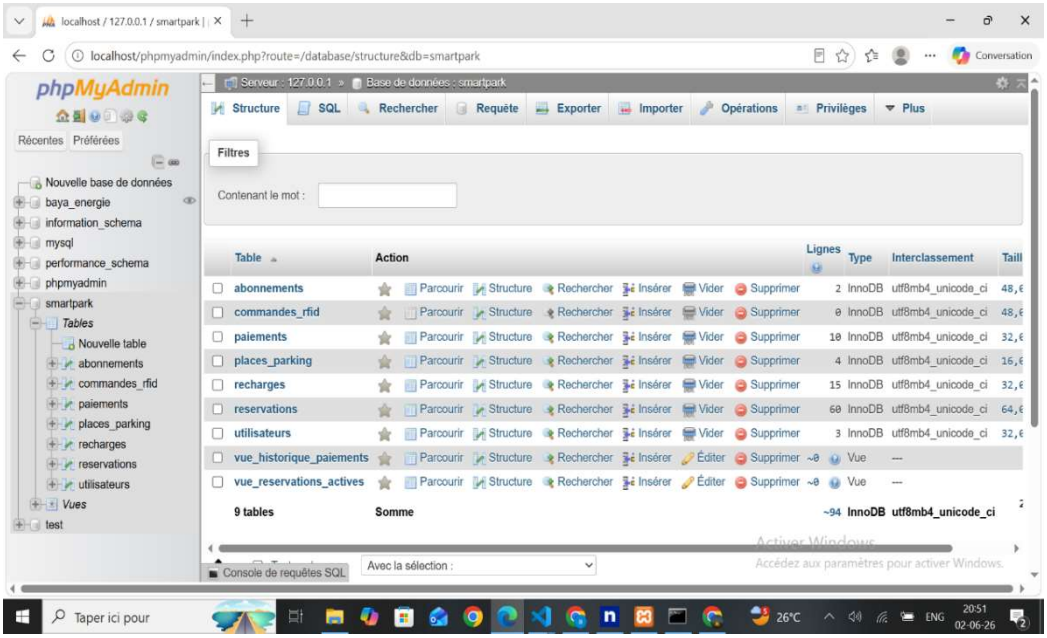

Figure 3.13 : Structure de la base de données SmartPark sous phpMyAdmin.

Tableau 3.3 : Les différents API utilisés dans notre système de parking intelligent.

API	Méthode HTTP	Rôle	Exemple d'utilisation
`/api /user /register`	POST	Créer un compte utilisateur	Inscription d'un nouveau client
`/api/user/login`	POST	Connexion	Authentifier un utilisateur
`/api/user/profile`	GET	Voir profil	Afficher les infos d'un utilisateur connecté
`/api/user/update`	PUT	Modifier profil	Changer les informations relatives du compte
`/api/reservation`	POST	Créer une réservation	Réserver une place pour un utilisateur
`/api/reservation/{id}`	GET	Consulter une réservation	Voir les détails d'une réservation
`/api/reservation/{id}`	DELETE	Annuler une réservation	Libérer automatiquement la place
`/api/place/available`	GET	Vérifier disponibilité	Afficher les places libres
`/api/place/assign`	POST	Attribuer une place	Associer une place à un utilisateur
`/api/place/release`	POST	Libération manuelle	Libérer une place après sortie
`/api/parking/status`	GET	État du parking	Nombre de places libres/occupées
`/api/user/reservations`	GET	Historique utilisateur	Voir les réservations d'un client
`/api/esp32/entry`	POST	Ouvrir barrière entrée	Si PIN ou RFID ou Ticket valide
`/api/esp32/exit`	POST	Ouvrir barrière sortie	Si paiement valide
`/api/esp32/status`	GET	Etat de capteur	Vérification de l'état des capteurs IR
`/api/esp32/update`	POST	Envoyer les données	ESP32 met à jour le parking
`/api/payment/method`	POST	Choisir le mode de paiement	L'utilisateur sélectionne Edahabia ou PayPal
`/api/payment/method/{userId}`	GET	Voir mode choisi	Vérifier le mode sélectionné avant paiement
`/api/payment/edahabia/pay`	POST	Paiement par carte Edhahabia	Paiement via ECCP
`/api/payment/card/pay`	POST	Paiement par carte bancaire	Paiement classique
`/api/payment/confirm`	POST	Confirmation de paiement	Débité le montant de la carte et donné un pin
`/api/admin/dashboard`	GET	Statistique	Nombre places libres/occupées
`/api/admin/users`	GET	Les des utilisateurs	Gestion client
`/api/admin/reservations`	GET	Toutes les réservations	Suivi global de système
`/api/admin/reset`	POST	Reset le système	Remise à zéro du système

Cette base de données est composée de 7 **tables** et de 2 **vues,** permettant de gérer les informations relatives aux utilisateurs, aux abonnements RFID, aux places de stationnement et réservations, aux accès, à la libération d'une place, aux paiements, ainsi qu'à l'historique des

opérations. Grâce à cette structure, le système garantit une gestion efficace des données et une synchronisation en temps réel entre l'application web et la carte ESP32.

III.3.2. Communication entre ESP32 et l'application web

La communication entre microcontrôleur et une application web repose sur une connexion Wifi utilisant le protocole http afin d'assurer cette communication plusieurs bibliothèques sur Arduino IDE intégrée au programme de système embarqué.

III.3.2.1. Bibliothèques utilisées pour la communication

- **Bibliothèque Wifi.h :** cette bibliothèque permet à une carte Arduino de se connecter à Internet. Elle peut servir de serveur (acceptant les connexions entrantes) ou de client (établissant les connexions sortantes). La bibliothèque prend en charge les protocoles de chiffrement WEP et WPA2 Personnel, mais pas WPA2 Entreprise. Il est à noter également que si le SSID n'est pas diffusé, la carte d'extension ne pourra pas se connecter. Arduino communique avec la carte d'extension Wifi via le bus SPI [38].
- **Bibliothèque WifiClientserver.h :** implémente la prise en charge des connexions sécurisées utilisant TLS (SSL). Elle hérite de Wifi Client et implémente donc un sur-ensemble de l'interface de cette classe [39].
- **Bibliothèque HTTP Client :** est utilisée pour envoyer des requêtes http au serveur et recevoir les réponses correspondantes.
- **Bibliothèque ArduinoJson** : permet de traiter les données échangées au format Json, et de faciliter l'interprétation des réponses reçues du serveur.

III.3.2.2. Technologies de communication utilisées

a) Réseaux Wifi (IEEE 802.11)

Le premier standard Wifi offrait des débits de l'ordre du Mbps. Il définissait une technique d'accès CSMA/CA (Carrier Sense multiple collision avoidance). Cette technique visait à privilégier la fiabilité de la transmission des données dans des environnements difficiles en sacrifiant une partie de la bande passante théoriquement disponible au profit de ces mécanismes de vérification. La norme initiale a connu des évolutions, qui correspondent à l'ensemble des normes de la famille IEEE802.11.

Il existe deux bandes de fréquence utilisées e, Wifi qui sont [40]:

- La bande 2.4 GHz pour les amendements IEEE 802.11b et IEEE 802.11g.
- La bande 5 GHz pour l'amendement IEEE 802.11a.

b) Modèle TCP/IP

Le modèle TCP/IP (Transport Control Protocol/Internet Protocol), dit aussi modèle DOD (Departement Of Defence) a été mis au point par une agence du ministère de la défense américaine, DARPA (Defense Advanced Research Agency) vers les années 70.

Les protocoles TCP/IP représentent un ensemble de règles de communication sur Internet et se basent sur la notion adressage IP, c'est-à-dire, le fait de fournir une adresse IP à chaque machine du réseau afin de pouvoir acheminer des paquets de données. Etant donné que la suite de protocoles TCP/IP a été créée à l'origine dans un but militaire, elle est conçue pour répondre à un certain nombre de critères, parmi lesquels [41]:

- Le fractionnement des messages en paquets
- L'utilisation d'un système d'adresses
- L'acheminement des données sur le réseau (routage)
- Le contrôle des erreurs de transmission de données

Le modèle TCP/IP peut en effet être décrit comme une architecture réseau à 4 couches comme le montre la Figure 3.15 suivante.

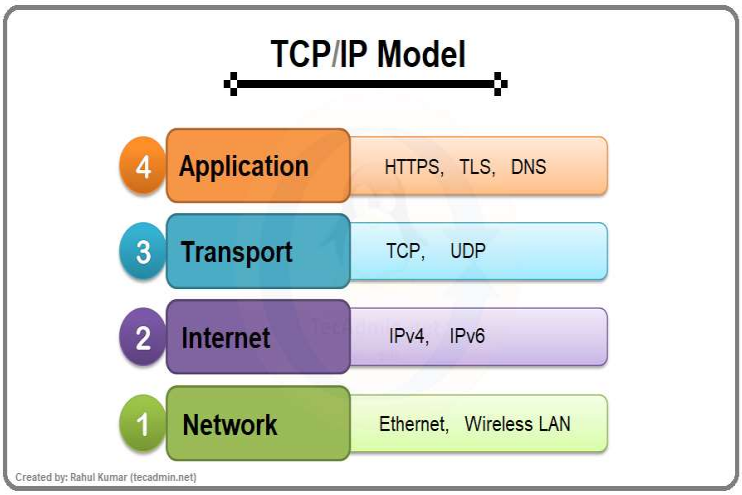

Figure 3.15 : Architecture du modèle TCP /IP [46].

c) Protocol HTTP (HyperText Transfert Protocol) [42]

HTTP est un protocole de la couche application permettant la transmission de documents hypermédia, tels que le HTML. Conçu initialement pour la communication entre navigateurs web et serveurs web, il peut également servir à d'autres fins, comme la communication machine à machine, l'accès programmatique aux API, etc.

Le protocol HTTP a plusieurs caractéristiques parmi eux nous citons **:**

- Sans état (stateless) : chaque requête est indépendante, le serveur ne conserve pas de contexte entre deux requêtes
- Basé sur TCP/IP (port 80 par défaut)
- Texte lisible : les messages sont clairs, donc interceptables.

Le Tableau 3.4 rapporte les codes statuts associés au protocole HTTP tandis que la Figure 3.16 illustre le principe de fonctionnement du protocole HTTP selon le modèle client-serveur.

Tableau 3.4 : Codes statuts associés au protocole HTTP [43].

Code	Signification
200	Accès succès
301	Redirection permanant
400	Mauvaise requête
401	Accès non autorisé
403	Accès interdit
404	Ressource introuvables
500	Erreur serveur internet

Figure 3.16 : Principe de fonctionnement du protocole HTTP selon le modèle client-serveur [47].

d) Format JSON (Java Script Object Notation) [44]

JSON est un format d'échange de données léger. Il est facile à lire et à écrire pour les humains, et facile à analyser et à générer pour les machines. Il est basé sur un sous-ensemble de la norme du langage de programmation JavaScript ECMA-262, 3ème édition (décembre 1999). JSON est un format texte totalement indépendant du langage mais utilise des conventions familières aux programmeurs de la famille de langages C, notamment C, C++, C#, Java, JavaScript, Perl, Python et bien d'autres. Ces propriétés font de JSON un langage d'échange de données idéal.

Voici les principaux éléments de la syntaxe JSON :

- Les données sont présentées sous forme de paires clé/valeur.
- Les éléments de données sont séparés par des virgules.
- Les accolades {} désignent les objets.
- Les crochets [] désignent des tableaux.

La Figure 3.17 montre la structure principale du format JSON.

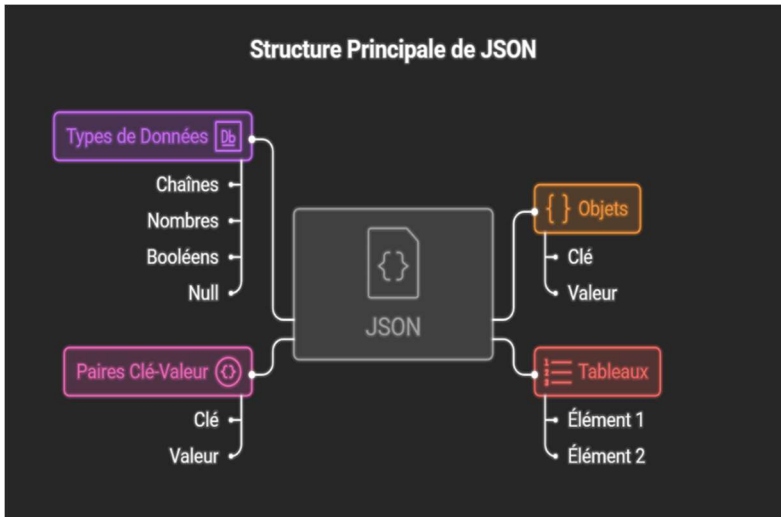

Figure 3.17 : Structure principale de JSON.

e) Protocole SPI (Serial Peripheral Interface) [45]

L'interface périphérique série (SPI) est un protocole de communication série synchrone utilisé pour permettre l'échange de données à grande vitesse entre un dispositif maître et plusieurs dispositifs périphériques. Il s'agit d'un système de communication full-duplex conçu pour transférer des données rapidement et efficacement sur de courtes distances. Couramment utilisé

dans les systèmes embarqués, les capteurs et les microcontrôleurs, le protocole SPI simplifie la communication entre les périphériques.

Le protocole SPI fonctionne à l'aide de quatre signaux primaires : **MOSI (Master Out Slave In), MISO (Master In Slave Out), SCLK (Serial Clock) et CS (Chip Select).** Le dispositif maître contrôle le signal d'horloge, coordonnant la transmission des données avec les périphériques. SPI est connu pour sa simplicité et sa rapidité, offrant une fréquence d'horloge personnalisable et une communication full-duplex, ce qui signifie que les données peuvent être envoyées et reçues simultanément. Contrairement à d'autres protocoles de communication, SPI ne nécessite pas d'adressage, ce qui réduit les frais généraux et accélère la communication. La Figure 3.18 ci-dessous résume le principe de fonctionnement de ce protocole.

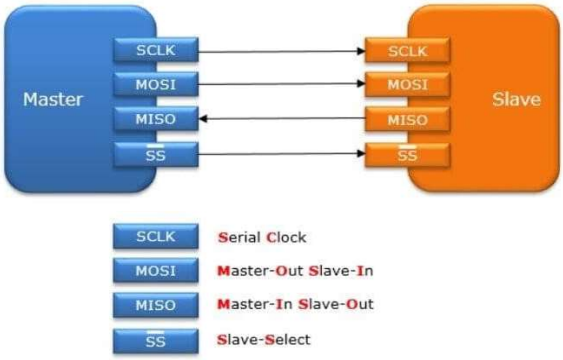

Figure 3.18 : Principe de fonctionnement du Protocol de communication SPI [48].

III.3.2.3. Fonctionnement de la communication

La communication entre ESP 32 et l'application web est basée généralement sur l'échange des requêtes HTTP et des réponses au format JSON. Lorsqu'un utilisateur effectue une réservation via l'application web, la demande est envoyée vers le backend qui enregistre les informations dans la base de données et traite la disponibilité de places. Ensuite, les données de réservation sont transmises vers l'ESP32 avec le protocole de communication HTTP. Une fois que l'ESP32 reçoit ces informations, il analyse les données et effectue l'action correspondante, qui consiste à mettre à jour l'état du parking intelligent qui gère.

Une fois le code PIN validé par l'application web, la session de stationnement débute par le déclenchant d'un compteur qui calcule en temps réel la durée d'utilisation et le montant dû. Ces informations, structurées sous forme de message JSON, sont envoyées au serveur web via une requête HTTP. Le backend reçoit ces données, les traite et les stocke dans la base de données avant de les afficher sur l'interface utilisateur.

Lorsqu'un utilisateur décide de libérer sa place de parking, il accède à l'interface web et sélectionne l'option de libération de la place. Une requête HTTP est ensuite envoyée au serveur pour mettre à jour le statut des places dans la base de données. Le backend transmet ces informations à l'ESP32, qui arrête le minuteur, finalise le calcul du montant dû et met à jour le statut de cette place, le marquant comme disponible.

Une fois la place de parking libérée, l'utilisateur est redirigé vers l'interface de paiement où il peut choisir son mode de paiement préféré. Après la validation du paiement, le serveur enregistre la transaction dans la base de données et met à jour l'historique de stationnement. Le système génère ensuite un code de sortie unique pour le client, qui lui permet d'ouvrir la barrière à la sortie.

Les informations relatives à la durée totale de stationnement, au montant payé, à la date et à l'heure de la transaction sont conservées afin d'assurer la traçabilité des opérations et de permettre à l'utilisateur de consulter les détails de sa session de stationnement.

Lorsqu'un utilisateur entre dans le parking à l'aide d'un ticket ou d'une carte RFID, l'ESP32 envoie une requête HTTP au serveur pour vérifier les informations correspondantes dans la base de données. Après validation, le backend met à jour l'état du parking et transmet la réponse à l'ESP32, qui exécute les actions nécessaires. Les données sont ensuite enregistrées afin d'assurer un suivi en temps réel de l'occupation des places.

Un compte administrateur dédié a été créé pour assurer la gestion complète du système de stationnement. L'accès à ce compte est sécurisé par une adresse e-mail et un mot de passe uniques. L'administrateur dispose de privilèges étendus lui permettant de consulter toutes les données du système, y compris l'état en temps réel du parking (places disponibles et occupées), les informations des utilisateurs et l'historique des opérations enregistrées. Il peut également gérer directement le parking en libérant une place si nécessaire et en mettant à jour les informations correspondantes dans la base de données.

III.3.2.3. Schéma de communication ESP32-application web-base de données

La Figure 3.19 illustre le schéma détaillé de communication entre l'ESP32, l'application web, et la base de données dans le cadre du projet Smart Parking réalisé.

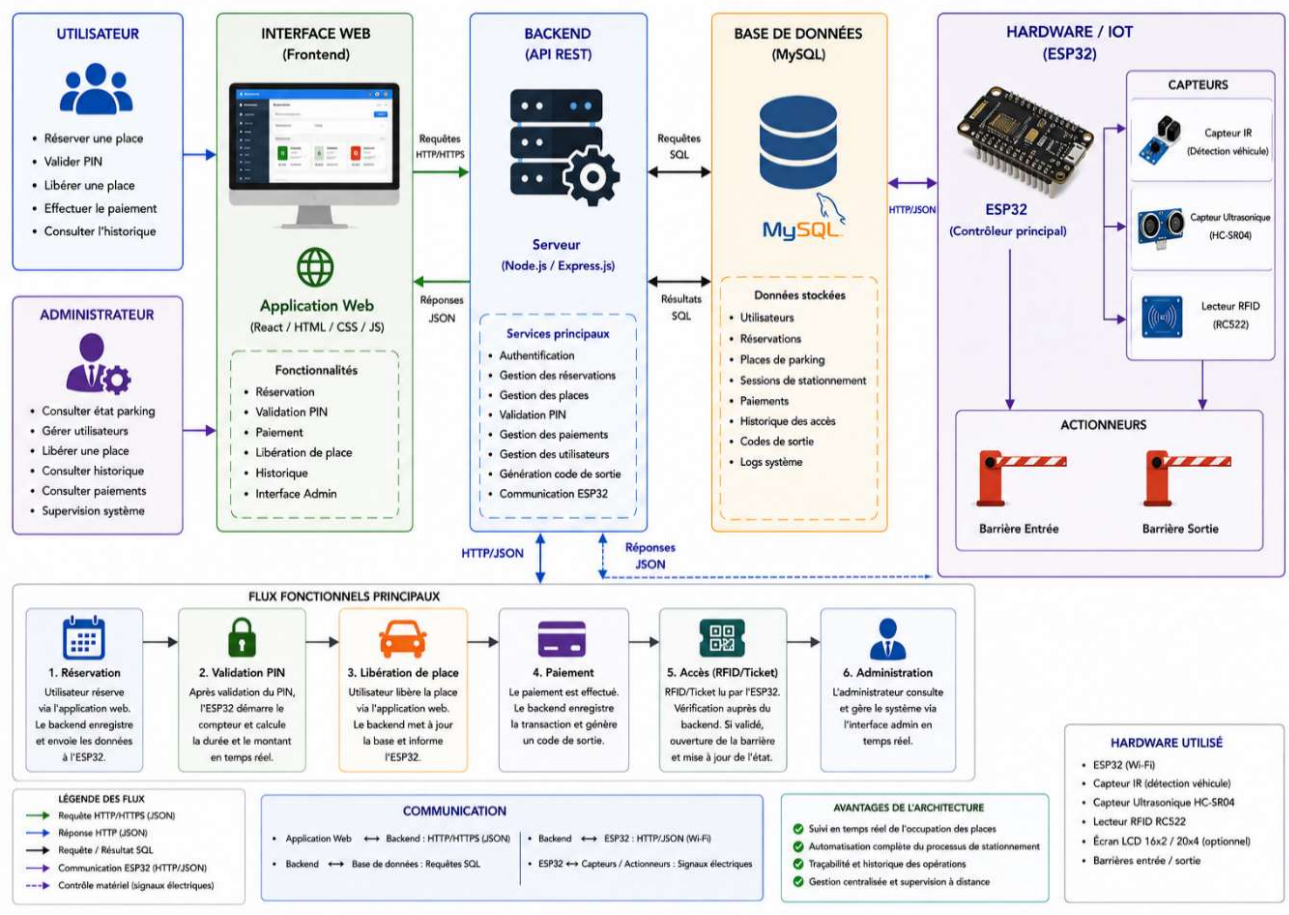

Figure 3.19 : Schéma de communication entre l'ESP32, l'application web et la base de données.

III.4. Développement de programmes embarqués sur ESP32

Les programmes implémentés sur la carte ESP32 permettent la gestion des capteurs ultrasoniques et IR, le contrôle les servomoteurs d'entrée et de sortie, la communication avec le serveur, ainsi que l'affichage des informations sur un écran LCD. Le code source est développé dans l'environnement Arduino IDE en utilisant plusieurs bibliothèques spécifiques.

III.4.1. Gestion de la connexion Wifi

a) Organigramme :

La Figure 3.20 montre l'organigramme de gestion de la connexion Wifi.

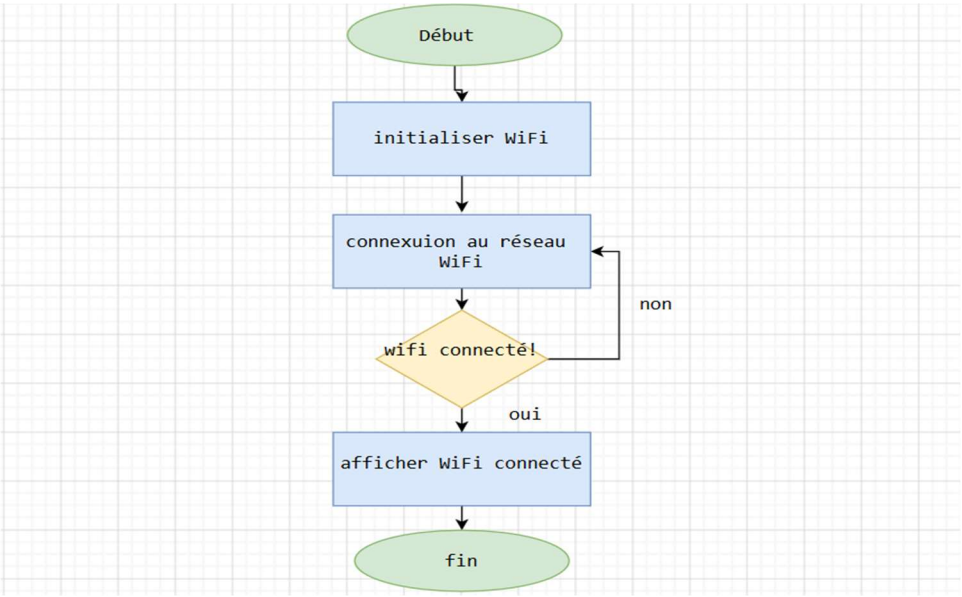

Figure 3.20 : Organigramme de gestion de la connexion Wifi.

III.4.2. Gestion de capteur HC-SR04

a) Organigramme :

La Figure 3.21 représente l'organigramme du fonctionnement du capteur ultrasonique HC-SR04.

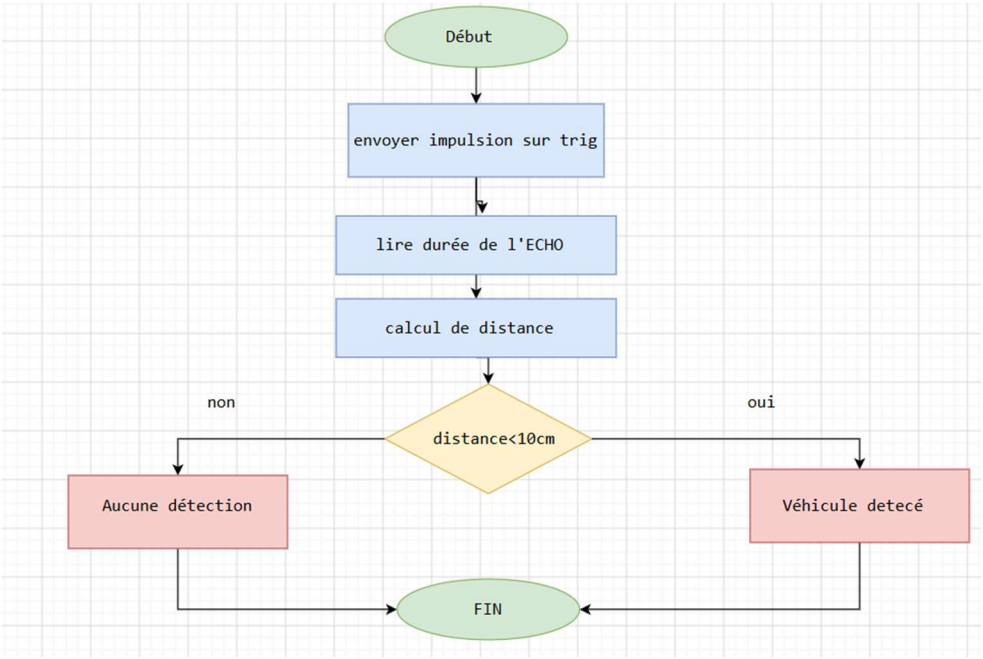

Figure 3.21 : Organigramme de fonctionnement du capteur ultrasonique HC-SR04.

III.4.3. Gestion des places de stationnement

a) Organigramme :

La gestion des places de stationnement est effectuée à l'aide de capteurs infrarouges comme illustré dans l'organigramme de la Figure 3.22.

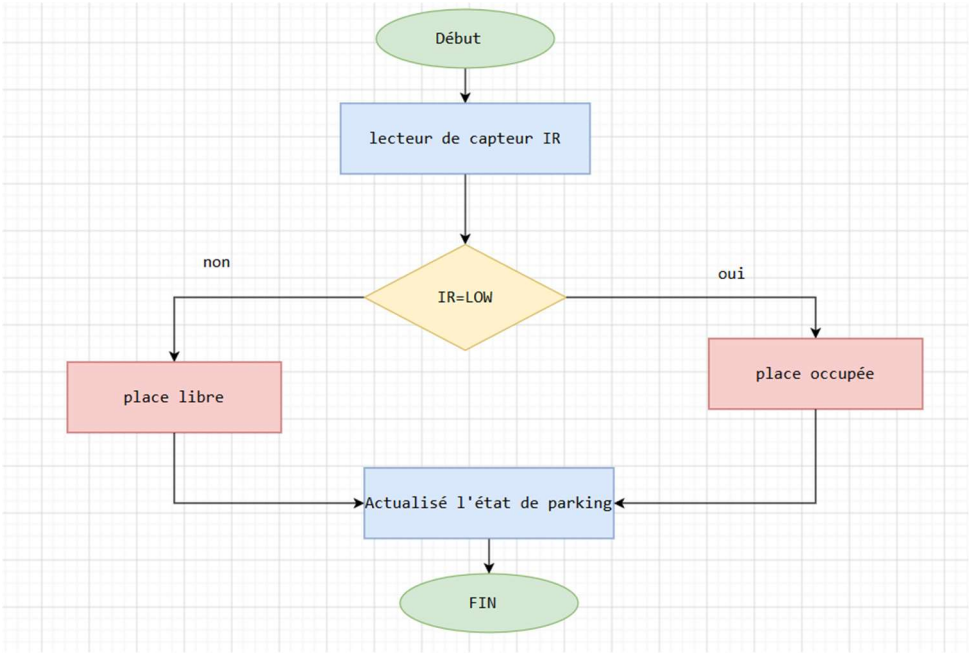

Figure 3.22 : Organigramme de gestion des places de stationnement via capteurs infrarouges.

III.4.4. Gestion du module RFID

a) Organigramme :

La gestion des places du module RFID est effectuée selon l'organigramme de la Figure 3.23.

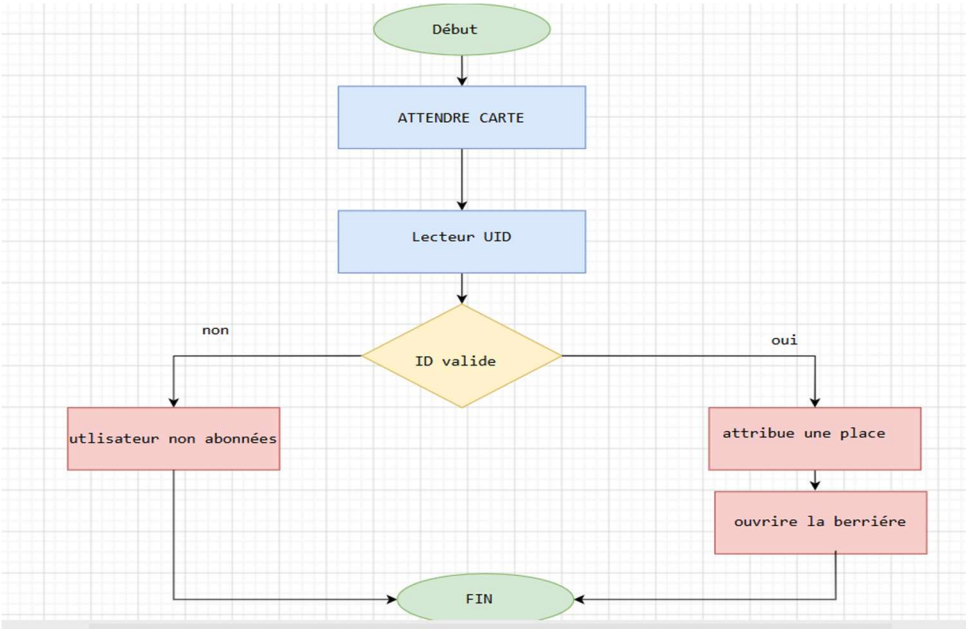

Figure 3.23 : Organigramme de gestion du lecteur de cartes RFID.

III.4.5. Gestion d'entrée

a) Organigramme :

La Figure 3.24 représente l'organigramme de gestion d'entrée.

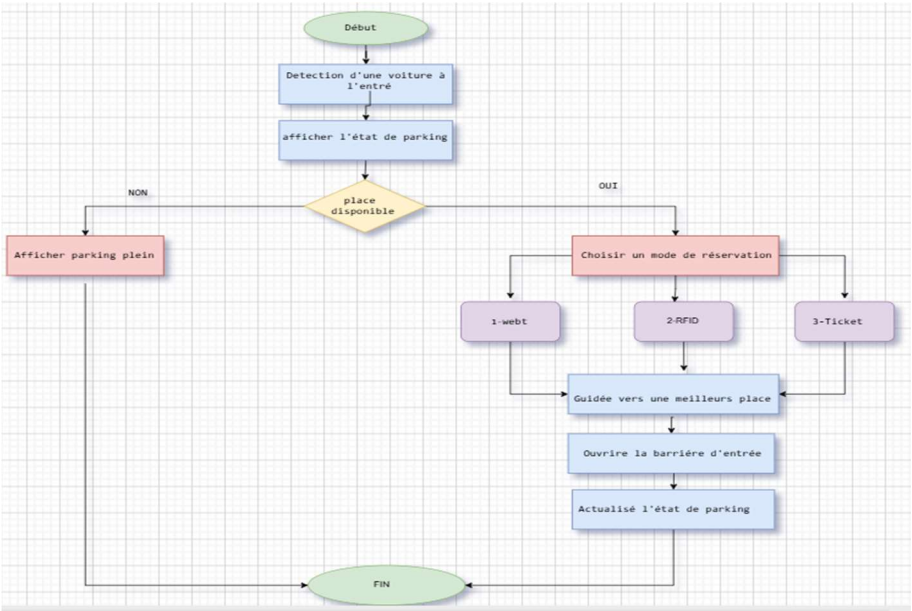

Figure 3.24 : Organigramme de gestion d'entrée**.**

III.4.7. Gestion de sortie

a) Organigramme :

L'organigramme de gestion de sortie est présenté dans la Figure 3.25.

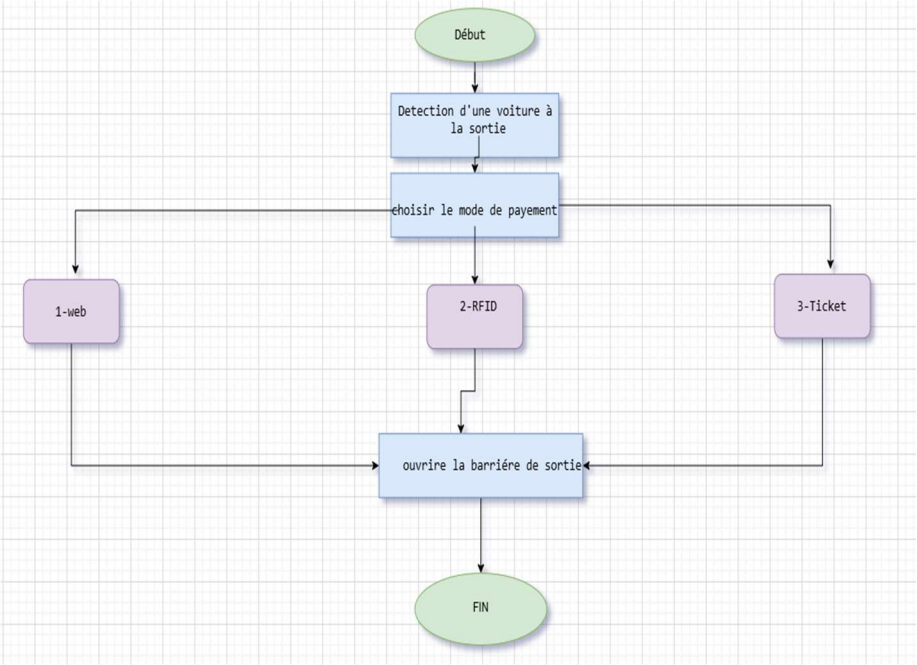

Figure 3.25 : Organigramme de gestion de sortie

III.5. Conclusion

Dans ce chapitre, nous avons présenté l'ensemble des aspects logiciels mis en œuvre pour la réalisation du système. Les différents outils de développement utilisés ont été décrits, notamment Arduino IDE pour la programmation de l'ESP32 ainsi que les technologies web employées pour le développement du frontend, du backend, et de la base de données.

Nous avons également détaillé l'architecture de l'application web ainsi que le mécanisme de communication entre l'ESP32 et le serveur via le protocole HTTP et l'échange de données au format JSON. Cette communication assure la synchronisation en temps réel des informations relatives aux places de stationnement, aux réservations, et aux opérations de paiement.

Enfin, les différents programmes embarqués développés pour la gestion des capteurs, des cartes RFID, des barrières automatiques, de l'affichage LCD et des procédures d'entrée et de sortie ont été présentés. L'intégration de ces différents modules a permis la mise en œuvre d'un système complet, capable d'automatiser la gestion du parking tout en offrant aux utilisateurs une interface web moderne et intuitive.

CHAPITRE IV

Résultats et discussions

IV.I. Introduction

Ce chapitre est consacré aux résultats de tests pratiques relatifs à notre projet de système de stationnement intelligent basé sur IoT.

Nous présenterons dans un premier temps la maquette réalisée ainsi que l'interface web développée. Ensuite, nous exposerons les différents tests effectués et les résultats obtenus. Une discussion des résultats permettra d'évaluer les performances et le bon fonctionnement du système. Enfin, ce chapitre se terminera par une conclusion récapitulant les principaux résultats de cette phase de réalisation et de validation.

IV.2. Présentation de la plateforme réalisée

Dans cette partie, nous présentons la réalisation pratique de notre système de stationnement intelligent, à travers la maquette matérielle mise en œuvre, et l'interface web développée pour les utilisateurs et l'administrateur.

IV.2.1. Prototype matérielle réalisé (Figure 4.1)

Figure 4.1 : Prototype matérielle réalisé.

IV.2.2. Interface web (Figure 4.2)

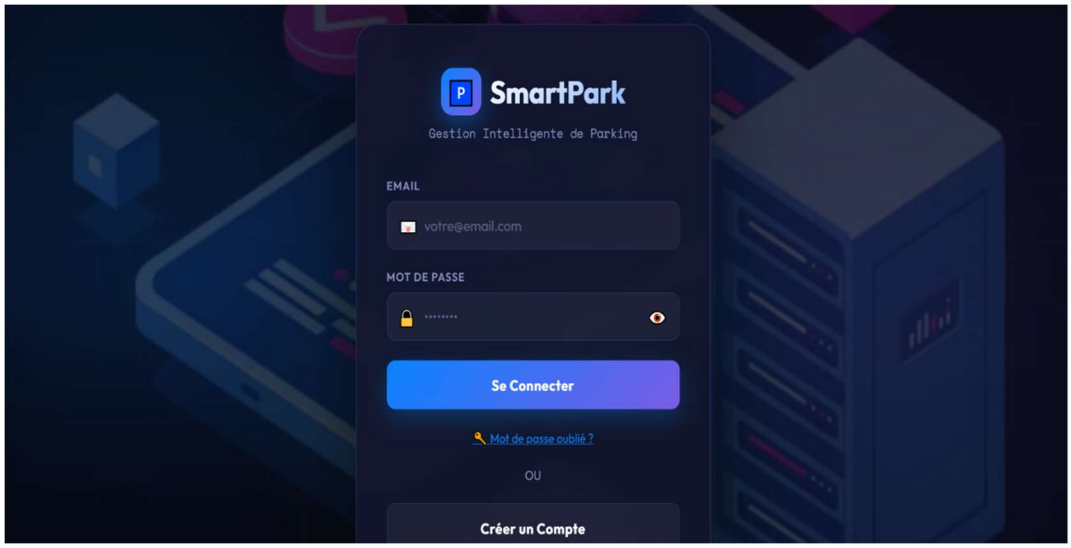

Figure 4.2 : Interface web réalisée.

IV.3. Tests pratiques et résultats obtenus

IV.3.1. Test de connexion d'ESP32 au serveur web

- Connexion Wi-Fi :

Ce test vérifie que l'ESP32 est capable de se connecter correctement à un réseau Wi-Fi. Il faut que l'ESP32 soit alimenté et configuré avec un ID et un mot de passe spécifique.

On observe les logs via le port série pour confirmer l'établissement de la connexion. La connexion est réussite si l'ESP32 obtient une adresse IP valide et affiche un message de confirmation tel que "WIFI OK ".

La Figure 4.3 ci-dessous représente la connexion d'ESP32 avec Wi-Fi.

Figure 4.3 : La connexion d'ESP32 avec Wi-Fi.

- Echange de données entre ESP32-application web :

a) Scenario 01 : Envoi d'une requête http GET au serveur

Ce scénario a pour objectif de vérifier la capacité de l'ESP32 à communiquer avec le serveur web à travers une requête HTTP GET. L'ESP32 doit être préalablement connecté au réseau Wi-Fi et le serveur web doit être opérationnel et accessible.

Une fois la connexion établie, l'ESP32 envoie une requête HTTP GET à l'adresse du serveur afin de récupérer des informations. Le serveur traite la requête puis retourne une réponse contenant les données demandées. Le test est considéré comme réussi si l'ESP32 reçoit un code de réponse HTTP 200 et que le contenu de la réponse correspond aux données attendues.

b) Scenario 02 : Envoi des données via http POST

Ce scénario a pour objectif de vérifier la capacité de l'ESP32 à transmettre des données vers le serveur à travers une requête HTTP. L'ESP32 collecte les informations nécessaires au système, telles que l'identifiant RFID de l'utilisateur, l'état des places de stationnement ou les événements d'entrée et de sortie des véhicules. Ces données sont ensuite formatées et envoyées au serveur via une requête HTTP POST.

Le serveur est configuré pour recevoir ces informations, les traiter et les enregistrer dans la base de données. Le test est réussi si les données sont correctement reçues et stockées par le serveur, et qu'une réponse de confirmation est retournée à l'ESP32 avec un code HTTP 200. Les résultats obtenus montrent que la communication entre l'ESP32 et le serveur est fonctionnelle.

IV.3.2. Test de création d'un compte sur l'application

Ce test a pour objectif de vérifier le bon fonctionnement de la fonctionnalité de création d'un compte sur l'application web. L'utilisateur accéde à l'interface et renseigne les informations associées telles que le nom, l'adresse email, le matricule, et le mot de passe. Après la création du compte, les données sont transmises au serveur et enregistrées dans la base de données.

La Figure 4.4 ci-dessous représente l'interface de création d'un compte.

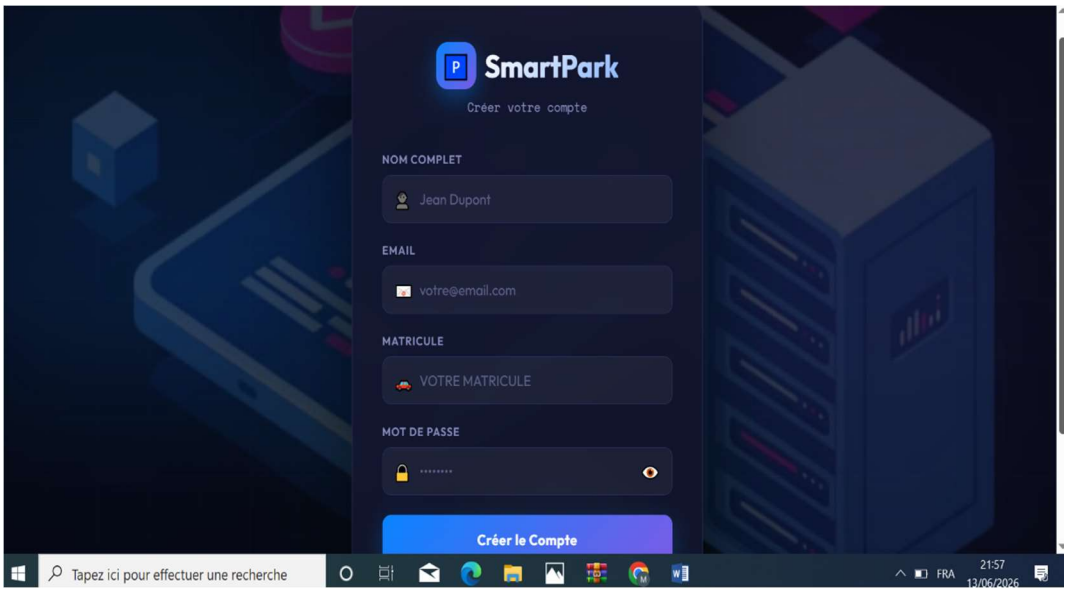

Figure 4.4 : Interface de création d'un compte.

Le test considère comme réussi si le compte de l'utilisateur est créé sans erreurs et les informations sont enregistrées correctement dans la base de données.

La Figure 4.5 ci-dessous représente la création correcte du compte.

Figure 4.5 : La création correcte du compte.

IV.3.3. Test de réservation d'une place en ligne

Ce test a pour l'objectif de vérifier la capacité de l'utilisateur à réserver une place en ligne via l'application web après qu'il soit authentifié à son compte. L'utilisateur peut consulter l'état de parking et choisir la meilleure place s'il y a des places libres, en suite, l'utilisateur confirme la réservation.

Ce test est considéré réussi si le serveur affiche une page contenant les informations relatives à la réservation effectuée telles que la place réservée, le code unique associé, et le temps d'ouverture. Notons qu'il faudra valider le code au parking avant 10 minutes sinon la réservation s'annule automatiquement.

La Figure 4.6 ci-dessous représente une réservation d'une place via l'application web.

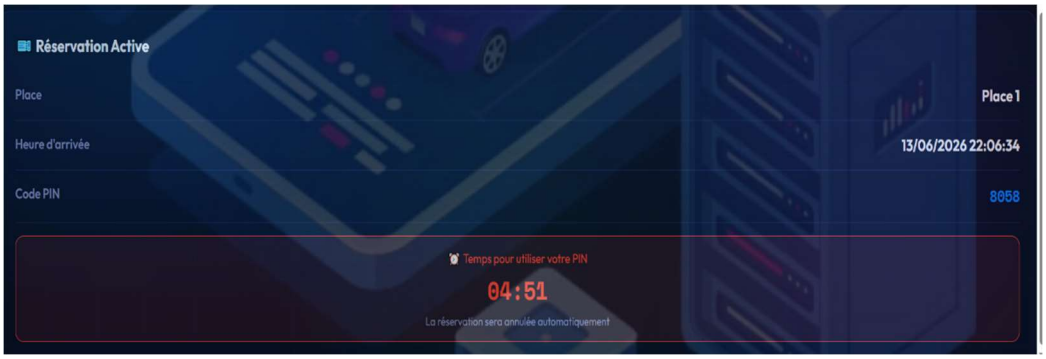

Figure 4.6 : Réservation d'une place via l'application web.

IV.3.4. Test de détection des véhicules

Ce test a pour objectif de vérifier le bon fonctionnement des différents capteurs au parking.

a) Scenario 01 : Détection de véhicules à l'entrée

Ce test vise à vérifier la capacité du capteur ultrasonique installé à l'entrée de parking à détecter l'arrivée de véhicules. Le capteur est installé à proximité de la barrière, détecte la présence d'un obstacle, puis transmet l'information à l'ESP32. Après la réception du signal, l'ESP32 analyse les données du capteur pour confirmer la présence d'un véhicule.

Le test est considéré comme réussi si le véhicule est détecté correctement et le système affiche sur LCD « VOITURE A L ENTREE » pendant 1 seconde puis il affiche l'état de parking, exemple : place libre :3, place occupée :1, et le mode d'accès au parking « 1-web 2-RFID 3-ticket ». La Figure 4.7 ci-après représente la détection d'un véhicule à l'entrée.

Figure 4.7 : Détection d'un véhicule à l'entrée.

La Figure 4.8 ci-dessous représente l'affichage de l'état de parking sur LCD.

Figure 4.8 : Affichage de l'état de parking sur LCD.

La Figure 4.9 représente l'affichage du mode d'accès au parking sur LCD.

Figure 4.9 : Affichage du mode d'accès au parking sur LCD.

b) Scenario 02 : Détection de véhicule à la place de stationnement

Ce test vise à vérifier la capacité des capteurs infrarouges installés à chaque place de stationnement à détecter le stationnement de véhicules. A chaque fois qu'un véhicule s'arrête définitivement à sa place, le capteur est mis à l'état bas et envoie un signal à l'ESP32, et un nouveau message tel que « Véhicule garé à la place 3 » est affiché.

Le test est jugé réussi si le véhicule est détecté correctement et le système affiche sur LCD que le véhicule a stationné à sa place. La Figure 4.10 représente un exemple de détection d'un véhicule par un capteur IR.

Figure 4.10 : Exemple de détection d'un véhicule par capteur IR.

c) Scenario 03 : Détection de véhicules à la sortie

Ce test vise à vérifier la capacité du capteur ultrasonique installé à proximité de la barrière de sortie du parking à détecter le véhicule. Après la réception du signal indiquant l'existence d'un obstacle de la part du capteur, l'ESP32 analyse les données pour confirmer la présence du véhicule.

Le test est considéré comme réussi si le véhicule est détecté correctement et le système affiche sur LCD un message comme par exemple « Voiture détectée à la sortie » pendant une seconde puis il visualise l'état de parking et le mode de paiement selon les variantes suivantes : « 1WEB 2RFID 3TICKET ».

La Figure 4.11 illustre le résultat de ce scénario.

Figure 4.11 : Détection d'un véhicule à la sortie.

IV.3.5. Test d'accès au parking

a) Scenario 01 : Accès via l'application web

Ce test a pour l'objectif de vérifier le fonctionnement du système de stationnement intelligent conçu lorsqu'un utilisateur accède au parking via une réservation effectuée via l'application web. Le véhicule présent à l'entrée du parking doit saisir le code donné par l'application.

Le test est validé si le code saisi correspond au code lié à la place réservée alors que la barrière d'entrée s'ouvre.

La Figure 4.12 montre un exemple pratique de vérification de ce scénario.

Figure 4.12 : Vérification d'accès via l'application web.

b) Scenario 02 : Accès par carte RFID

Dans ce cas, le véhicule présent à l'entrée du parking intelligent demande au conducteur le mot de passe de sa carte RFID sur le lecteur installé à proximité de la barrière d'entrée pour valider l'accès s'il est abonné.

Le test est validé si la carte est correctement reconnue permettant d'attribuer une place (exemple place 01) au véhicule, la barrière d'entrée s'ouvre, et le système met à jour les places disponibles.

La Figure 4.13 illustre un test réel de l'accès par la carte RFID.

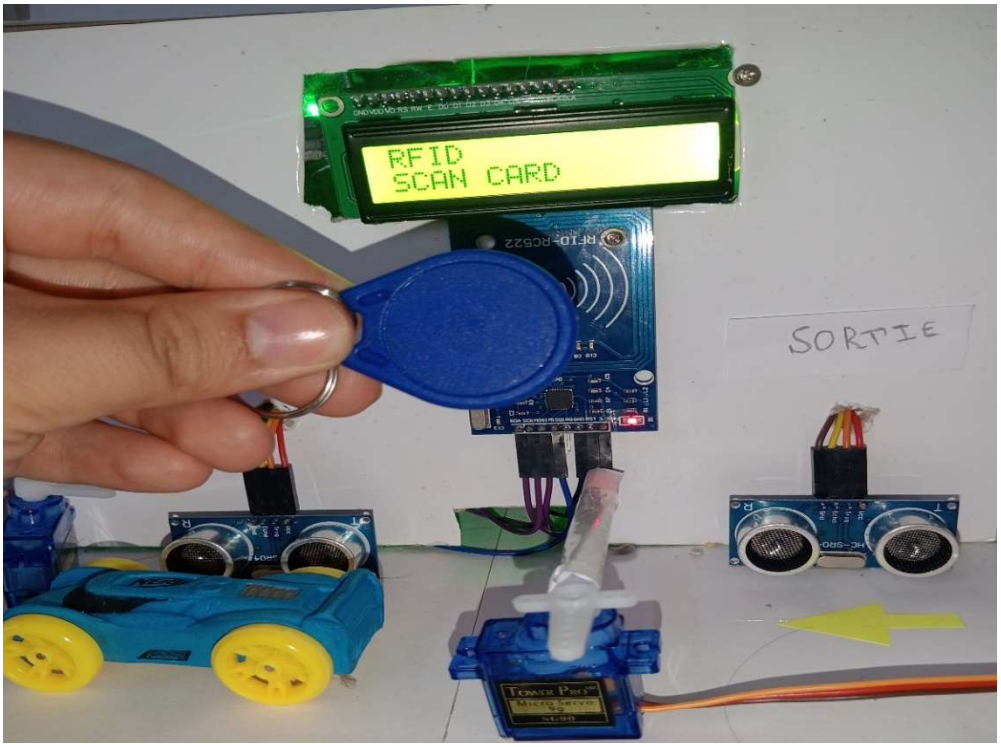

Figure 4.13 : Accès par la carte RFID.

c) Scenario 03 : Accès par ticket

Notre système de stationnement intelligent réalisé prend aussi en compte l'accès classique par le billet d'un ticket pour attribuer une place spécifique en incluant les différentes informations relatives telle que le numéro de la place réservée, le temps d'entrée, et un code particulier. Une fois le véhicule présent à l'entrée du parking, le système génère le ticket et le distribue.

Le test réussit si le système attribue un ticket qui contient toutes les informations utiles sur la place réservée.

La Figure 4.14 ci-dessous représente un exemple de l'accès par ticket.

Figure 4.14 : Accès par ticket.

IV.3.6. Test de libération et paiement d'une place

a) Libération et paiement en ligne

Ce test pour a objectif de vérifier le bon fonctionnement du système lorsqu'un utilisateur libère une place via l'application web. L'utilisateur connecté à son compte choisit l'option « libérer une place » et le système affiche les informations relatives de son stationnement telle que la durée de stationnement, la place, et le montant. L'utilisateur confirme la libération du parking et il se redirige vers l'interface de paiement.

Une fois le paiement confirmé, le système met à jour l'état des places disponibles et donne un code pin qui est utilisé à la sortie pour ouvrir la barrière.

Le test est jugé réussi si le paiement s'effectue correctement, le système actualise l'état du parking en temps réel, et l'utilisateur est capable de sortir du parking facilement et sans aucune contrainte.

Un exemple de ce scénario est reporté sur La Figure 4.15.

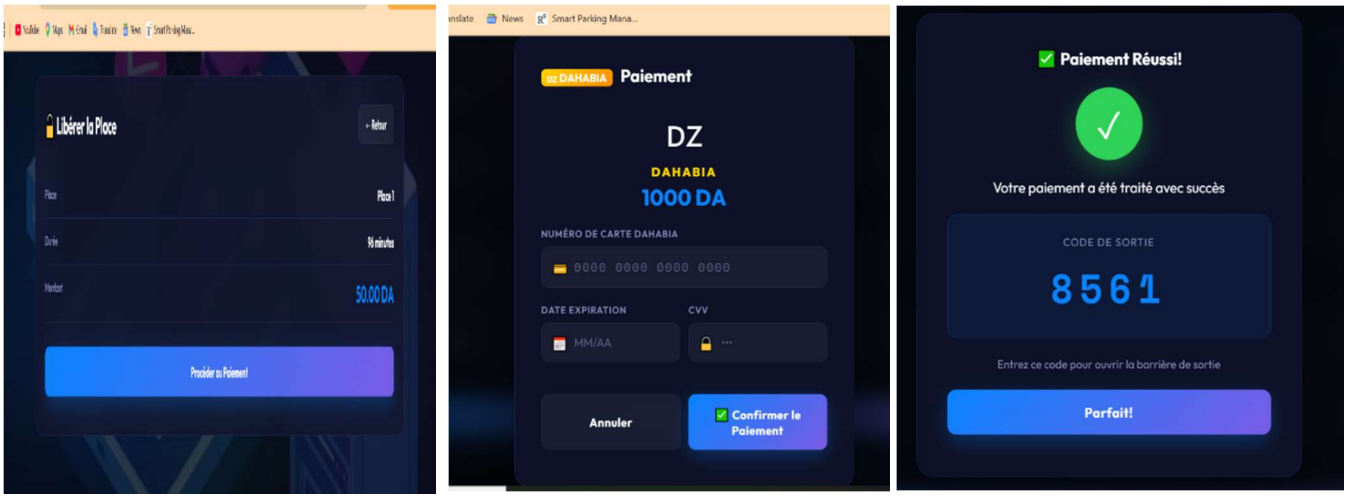

Figure 4.15 : L'opération de libération et paiement d'une place.

b) Paiement par carte RFID

Le même scénario a) est testé en utilisant une carte RFID et le lecteur installé à proximité de la barrière d'entrée permettant de lire l'UID du conducteur et vérifié son abonnement après avoir introduit le mot de passe.

On dit que le test est validé si la carte RFID est correctement identifiée et l'abonnement est vérifié. Ceci doit permettre à l'utilisateur de quitter le parking et au système d'actualiser son état conformément à la situation actuelle liée au stationnement.

c) Paiement par ticket

Dans ce scénario, il est question de tester la libération d'une place et le paiement par un ticket pour les utilisateurs non abonnés. Lors de l'entrée au parking, un ticket est généré et associé au véhicule. À la sortie, l'utilisateur présente son ticket afin que le système puisse récupérer les informations de stationnement et calculer automatiquement le montant à payer en fonction de la durée passée dans le parking. Les résultats obtenus montrent que le ticket est correctement identifié, que les frais sont calculés avec précision, et que la transaction est enregistrée dans la base

de données. Ce test confirme ainsi la fiabilité du système de paiement classique et son aptitude à gérer efficacement les utilisateurs occasionnels du parking.

La Figure 4.16 montre le ticket généré avec succès par le système à la sortie

```
===== FACTURE =====
Place  :  1
Temps  :  117 sec
Montant :  5.85
```

Figure 4.16 : Ticket généré par le système à la sortie.

IV.3.7. Test du compte administrateur

Ce test vise à vérifier le bon fonctionnement du compte admin et l'accès aux différentes fonctionnalités du parking. L'administrateur se connecte à l'application avec un email unique qui y est exclusivement réservé. Après authentification, l'administrateur dispose d'une vue globale sur l'ensemble des opérations réalisées dans le parking. À travers le tableau de bord, il peut consulter l'état des places de stationnement, suivre les entrées et les sorties des véhicules, gérer les réservations effectuées par les utilisateurs, visualiser les paiements réalisés ainsi que les informations des clients. Cette centralisation des données permet une gestion efficace du parking et facilite le suivi en temps réel de son fonctionnement.

La Figure 4.17 ci-dessous représente l'espace du compte admin.

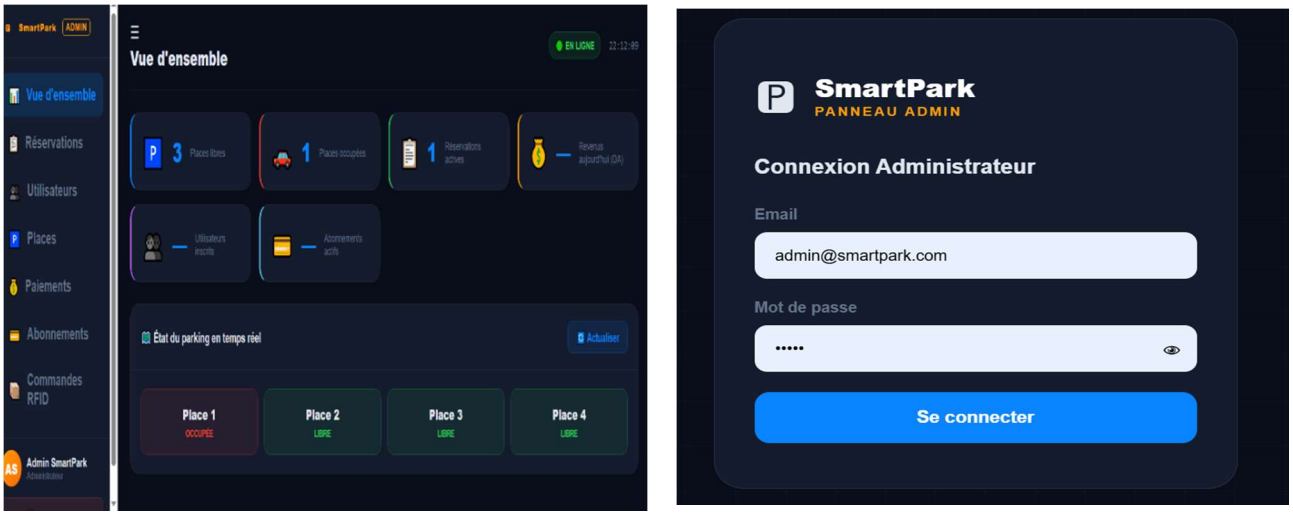

Figure 4.17 : L'espace du compte admin.

IV.3.8. Test de demande d'une carte d'abonnement RFID

Ce test pour l'objectif de vérifier le processus et l'attribution de carte d'abonnement RFID à l'utilisateur via l'application web. L'utilisateur connecter à son compte sur l'application Smart Park il choisir commander une carte RFID puis ou il veut prend son carte (à domicile ou au bureau), il remplir ses informations et il attente l'accepte par l'administrateur et peut suivi leur livraison de carte avec des emails envoyé par l'administrateur.

La Figure 4.18 illustre l'icône de demande de carte d'abonnement RFID.

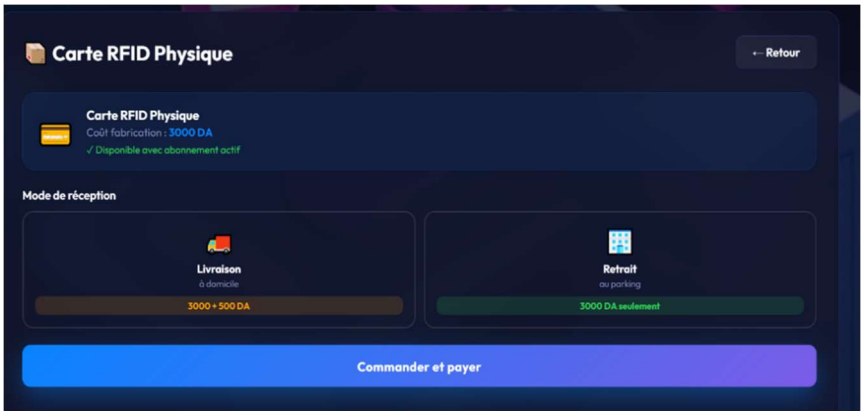

Figure 4.18 : L'icône de demande de carte d'abonnement RFID.

La Figure 4.19 montre la confirmation de la demande de la carte d'abonnement RFID, tandis que la Figure 4.20 illustre la demande au niveau de l'espace administrateur.

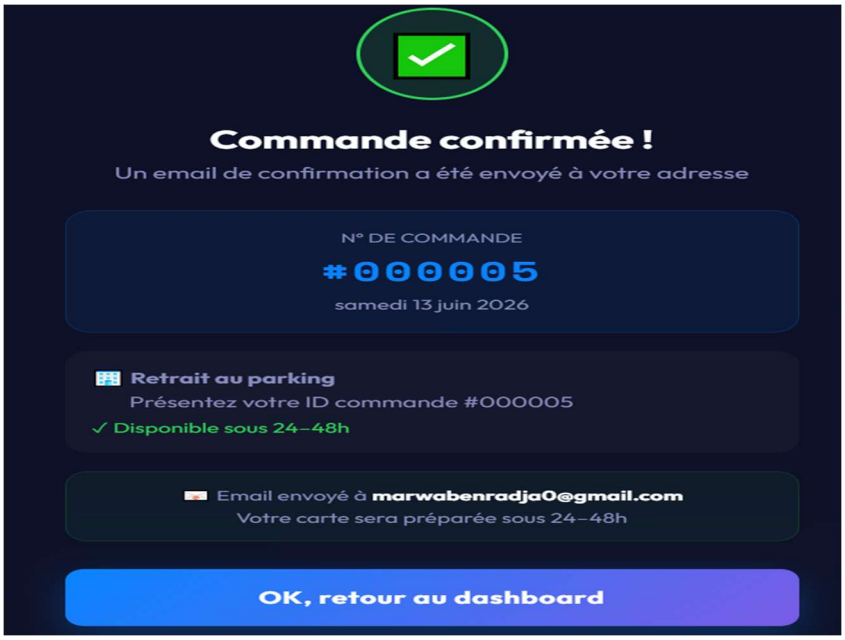

Figure 4.19 : Confirmation de la demande de la carte d'abonnement RFID.

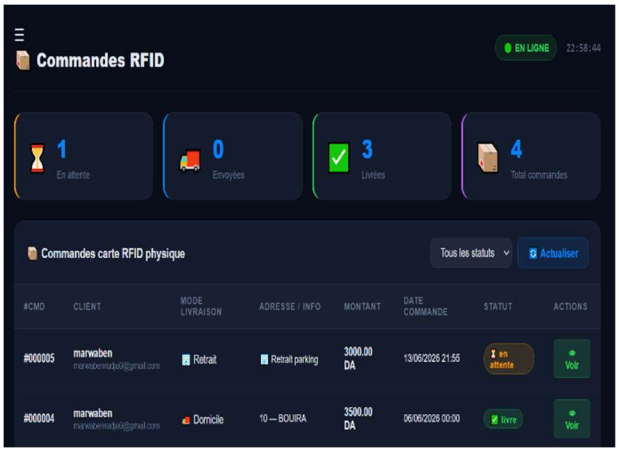

Figure 4.20 : Demande de carte RFID au niveau de l'espace administrateur.

La Figure 4.21 ci-dessous représente l'email de suivi envoyé au client.

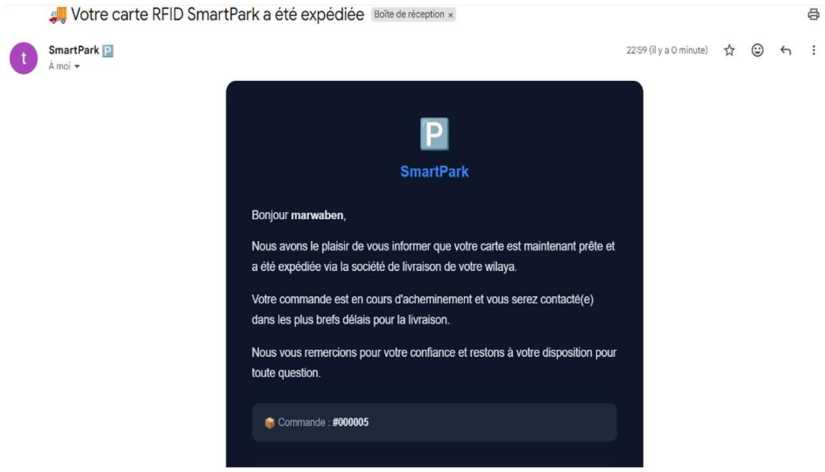

Figure 4.21 : L'email de suivi envoyé au client.

IV.4. Discussion des résultats

Les différents tests réalisés ont permis de valider le bon fonctionnement du système de gestion intelligente de parking proposé alors que :

- Les essais de connexion Wi-Fi ont montré que l'ESP32 établit une connexion stable avec le réseau et obtient correctement une adresse IP, ce qui garantit la communication avec le serveur web. Les tests d'échange de données via les requêtes HTTP GET et POST ont confirmé la fiabilité de la transmission des informations entre l'ESP32, le serveur, et la base de données.
- Les tests de l'application web ont démontré que les fonctionnalités de création de compte et de réservation de places sont opérationnelles. Les informations des utilisateurs sont correctement enregistrées dans la base de données et l'actualisation de l'état de parking en temps réel est bien fonctionnel.
- Concernant la détection des véhicules, les capteurs ultrasoniques installés à l'entrée et à la sortie ont détecté efficacement la présence des véhicules. Les capteurs infrarouges placés au niveau des places de stationnement ont également permis de déterminer avec précision l'état d'occupation des places. Les informations collectées sont immédiatement transmises à l'ESP32 puis vers la base de données et donc la mise à jour sur l'interface web.
- Les différents modes d'accès au parking (application web, carte RFID et ticket) ont été testés avec succès. Le système attribue correctement une place libre aux utilisateurs et contrôle l'ouverture des barrières en fonction du mode d'authentification choisi.
- Les opérations de libération de place et de paiement ont également donné des résultats très réussis. Le système met à jour automatiquement l'état des places après la sortie du véhicule et assure la synchronisation entre la base de données, l'application web, et les équipements électroniques du parking.
- Les résultats obtenus montrent que l'interface administrateur assure une supervision centralisée de l'ensemble des opérations du parking. Cette fonctionnalité permet un suivi en temps réel de l'activité du système et facilite les tâches de gestion et de contrôle. La centralisation des informations améliore l'efficacité de l'administration du parking et

permet une prise de décision rapide en cas de problème ou d'occupation élevée des places. De plus, l'accès sécurisé au tableau de bord garantit que seules les personnes autorisées peuvent gérer les données du système.

Toutefois, certaines limitations ont été observées lors des tests. La qualité de la connexion Wi-Fi peut influencer le temps de réponse du système. De plus, les performances peuvent être affectées dans le cas d'un nombre important d'utilisateurs connectés simultanément. Ces contraintes pourront être améliorées dans les travaux futurs par l'utilisation de protocoles de communication plus performants et par l'optimisation du serveur.

IV.5. Conclusion

Dans ce chapitre, nous avons présenté la réalisation pratique de notre système de stationnement intelligent ainsi que l'interface web développée pour sa gestion. Les différents tests effectués et les scénarios considérés ont permis de valider le bon fonctionnement des composants matériels et logiciels du système.

Les résultats obtenus montrent que la solution proposée répond aux exigences fixées initialement par le cahier de charge en termes de détection, de contrôle d'accès, de réservation, et de gestion des places de stationnement. Ces résultats confirment la faisabilité et l'efficacité du système développé sur des parkings réels.

Conclusion Générale

Conclusion générale

Dans ce travail, nous avons conçu et réalisé un prototype de système de stationnement intelligent basé sur l'Internet des Objets (IoT). L'objectif principal était de proposer une solution permettant d'améliorer la gestion des places de stationnement, de faciliter l'accès des utilisateurs au parking et d'assurer une supervision en temps réel à travers une application web.

La première partie de ce mémoire a permis de présenter les concepts fondamentaux liés à l'IoT, à la mobilité intelligente et aux systèmes de stationnement modernes. Cette étude a montré l'importance des technologies connectées dans l'amélioration de la gestion urbaine, notamment dans le domaine du transport et du stationnement.

Ensuite, l'architecture matérielle du système a été définie à partir de plusieurs composants essentiels, tels que la carte ESP32, les capteurs infrarouges, les capteurs ultrasoniques, les lecteurs RFID, les servomoteurs et l'écran LCD. Ces éléments ont permis d'assurer la détection des véhicules, l'identification des utilisateurs, l'ouverture automatique des barrières, et l'affichage en temps réel de l'état du parking.

La partie logicielle a également été développée afin d'assurer la communication entre le prototype matériel et l'application web. L'utilisation du protocole HTTP et du format JSON ont permis l'échange des données entre l'ESP32 et le serveur. L'application web réalisée offre plusieurs fonctionnalités, notamment la création de comptes, l'authentification, la réservation d'une place, la consultation de l'état du parking, et la gestion des informations utilisateurs.

Les tests réalisés ont confirmé le bon fonctionnement global du système. La connexion Wi-Fi de l'ESP32, l'échange de données avec le serveur, la détection des véhicules, la réservation des places, l'identification par RFID, la commande des barrières, et le paiement ont été validés avec succès. Les résultats obtenus montrent que le prototype proposé répond aux objectifs fixés par le cahier de charge et peut constituer une base fonctionnelle pour un système de stationnement intelligent.

Cependant, certaines améliorations peuvent être envisagées dans les travaux futurs parmi lesquelles nous citons :

- Intégration d'une application mobile Android/Ios permettant la réservation et le suivi du stationnement en temps réel.
- Mise en place d'un système de reconnaissance automatique des plaques d'immatriculation (ANPR) afin de remplacer ou compléter les cartes RFID.
- Ajout d'une caméra de surveillance pour renforcer la sécurité du parking et assurer la traçabilité des véhicules.

- Intégration de notifications en temps réel (SMS, e-mail ou notifications push) pour informer les utilisateurs concernant les réservations, paiements et expirations d'abonnement.
- Utilisation de technologies de communication IoT plus avancées telles que le **MQTT**, le **LoRa** ou la **5G** afin d'améliorer la couverture et les performances du système.
- Mise en œuvre d'algorithmes d'**intelligence artificielle** pour prédire les périodes de forte affluence et optimiser l'occupation du parking.

En conclusion, ce projet montre que l'intégration des IoT dans les systèmes de stationnement permet d'obtenir une solution pratique, automatisée et efficace. Le prototype réalisé constitue une contribution intéressante au développement des villes intelligentes et à l'amélioration de la mobilité urbaine.

Bibliographie

[1] Smart Parking Systems: A Survey Revathi, G., Dhulipala, V. S., **Smart Parking Systems and Sensors: A Survey**, IEEE, 2012.

[2]Internet of Things: A Hands-On Approach Arshdeep Bahga, Vijay Madisetti, *Internet of Things: A Hands-On Approach*, Universities Press, 2014.

[3] https://www.fortinet.com/fr/resources/cyberglossary/iot

[4] L. Atzori, A. Iera et G. Morabito, *"The Internet of Things: A Survey"*, Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010.

[5] ATEC ITS France, « les rencontres de la mobilité intelligente », 2016.

[6] Denden Nour El Houda et Derouiche Oussama, « Vision artificielle : Application à la conception d'un parking intelligent », mémoire de fin d'études master en Génie Industriel, Département de Maintenance en Instrumentation, Université Mohamed Ben Ahmed, 2021.

[7] Boumergued Ahlem, Nasri Sarah « Conception et réalisation d'une application mobile pour un parking intelligent », mémoire de fin d'études master en Informatique, Université Mohamed El Bachir El Ibrahimi Bordj Bou Arréridj,2020.

[8] Centre d'études et d'expertise sur les risques, l'environnement, la mobilité et l'aménagement. *Le stationnement : enjeux, politiques et aménagements*. CEREMA – Stationnement .

[9] https://www.realparkanpr.com

[10] https://www.weareplanet.com

[11] Mesloub Sliman et Saidani Mohand « le rôle de paiement électronique dans le développement de commerce électronique en Algérie » mémoire de fin d'études master en science commerciales, Département des Sciences Commerciales, Université Mouloud Mammeri Tizi-Ouzou.

[12] Denden Nour El Houda et Derouiche Oussama, « Vision artificielle : Application à la conception d'un parking intelligent », mémoire de fin d'études master en Génie Industriel, Département de Maintenance en Instrumentation, Université Mohamed Ben Ahmed, 2021.

[13] Idris M. Y. I., Leng Y. Y., Tamil E. M., Noor N. M., Razak Z.,"Car Park System: A Review of Smart Parking System and its Technology",Information Technology Journal, vol. 8, no. 2, pp. 101-113, 2009.

[14] Brahimi Nadia et Kerroum, « Vision artificielle : Etude et réalisation d'un système de stationnement intelligent basé sur l'ESP 32 », mémoire de fin d'études master en électronique des système embarqués, Département de D'Electronique, Université Mohamed Ben Ahmed, 2021.

[15] https://www.mobiliteintelligente.com/

[16] https://www.deep.eu/fr/ressources/articles-blog/telecom/solutions-iot/smart-parking-entreprise

[17] https://parking.brussels/fr/de-nouveaux-stickers-sur-les-horodateurs

[18] livre internet of things project with ESP32

[19] Espressive systems-ESP32 datasheets

[20] BOURENANE Sabrine, « Conception d'un habitat intelligent (smart house) : Domotique et confort labélisé à Souk Ahras, Algérie », Mémoire de master en architecture, Département d'Architecture, Faculté des Sciences et de Technologie, Université 08 mai 1945 de Guelma, Juin 2022.

[21] Capteur de proximité infrarouge | Comment ça marche, application et avantages (electricity-magnetism.org)

[22] : ETEME ONGUENE Jeanne Christelle, « Conception et réalisation d'un système de parking intelligent basé sur la technologie RFID», mémoire rédigé en vue de l'obtention du Diplôme de Professeur d'Enseignement Technique Deuxième Grade (DIPET II), Département de Génie Informatique, Université de Yaoundé I, 2020.

[23] https://www.gotronic.fr/pj2-guide-us-hc-sr04-compatible-arduino-

[24] T. Djerrai ? « Conception et réalisation d'un système de contrôle de serre agricole intelligente et autonome à base de la carte ESP 32 », Mémoire de master en électronique des systèmes embarqués, Département d'Electronique, Université Mouloud Mammeri Tizi-Ouzou.

[25] A. Boukoutaya, Réalisation d'une maison intelligente à base d'Arduino, Université MOHAMED V-Dep-physique 2015/2016

[26] https://www.gotronic.fr/art-module-nodemcu-esp32-usb-c-47499.htm?srsltid

[27] https://www.upesy.com/blogs/tutorials/esp32-pinout-reference-gpio-pins-ultimate guide?srsltid=AfmBOooo-DPmYvX-

[28] https://www.dzduino.com/module-de-capteur-

[29] https://abra-electronics.com/

[30] https://www.moussasoft.com/hc-sr04-capteur-ultrason-avec-arduino/

[31] https://fr.mfgrobots.com/mfg/mpm/1002020627.html

[32] https://www.circuits-diy.com/mfrc522-rfid-module

[33] https://cipam.com/comment-fonctionne-un-systeme-rfid/

[34] https://electronics.ma/product/i2c-1602-lcd-display-module/

[35] https://arduinogetstarted.com/tutorials/arduino-lcd-i2c

[36] https://www.aranacorp.com/fr/pilotez-un-servomoteur-avec-raspberry-pi/amp/

[37] Arduino,ArduinoDocumentation.Disponible:https://docs.arduino.cc/

[38] https://code.visualstudio.com/docs/getstarted/overview

[39] https://docs.arduino.cc/libraries/wifi/

[40]https://forum.arduino.cc/t/wificlient-versus-wificlientsecure-thingspeak-question/1407855/12

[41] Benchellal Amel, « Réseaux sans fil et réseaux mobiles », support de cours pour master 2 systèmes des télécommunications, Département de Génie Electrique, Université Abdelhamid Ibn Badis de Mostaganem.

[42] Computer Networking: A Top-Down Approach James F. Kurose, Keith W. Ross, *Computer Networking: A Top-Down Approach*, 8th Edition, Pearson, 2021.

[43] Mozilla Developer Network (MDN) – HTTP: HyperText Transfer Protocol

[44] https://www.rfc-editor.org/rfc/rfc9110.html

[45] https://www.json.org/json-en.html

[46] https://fr.digi.com/resources/definitions/spi

[47] https://medium.com/@noderin1/osi-and-tcp-ip-model-f12214e771ff

[48] https://fr.linkedin.com/pulse/what-http-protocol-cavliwireless-f6k6c?tl=fr

Annexe:

```
#include <SPI.h>
#include <MFRC522.h>
#include <ESP32Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <WiFiClientSecure.h>

// ================ RFID ================
#define SS_PIN 5
#define RST_PIN 27
MFRC522 rfid(SS_PIN, RST_PIN);

// ================ LCD I2C ================
LiquidCrystal_I2C lcd(0x27, 16, 2);

// ================ SERVOS ================
Servo servoEntree;
Servo servoSortie;
#define SERVO_ENTREE 13
#define SERVO_SORTIE 12

// ================ IR PARKING ================
int irPins[4] = {32, 33, 34, 35};
bool placeOccupee[4] = {false, false, false, false};
unsigned long tempsEntree[4];

// ================ HC-SR04 ================
#define TRIG_IN 16
#define ECHO_IN 17
#define TRIG_OUT 25
#define ECHO_OUT 26

// ================ WIFI ================
const char* ssid = "system telecom 2026";
const char* password = "30303005";

// ================ URL WEB ================
const char* SERVER_URL = "https://broadly-freebee-senator.ngrok-free.dev/api";

// ================ VARIABLES ================
int placesLibres = 4;
int placesOccupees = 0;
String pinPlaces[4];
bool entreeDetectee = false;
```

```
bool sortieDetectee = false;

// ================ DISTANCE ================
float lireDistance(int trigPin, int echoPin)
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH, 30000);
  if (duration == 0) return 999;
  return duration * 0.034 / 2;
}

// ================ LCD ================
void afficherLCD(String ligne1, String ligne2)
{
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(ligne1);
  lcd.setCursor(0, 1);
  lcd.print(ligne2);
}

//======================etat BDD=========================
void afficherEtatPlacesDB()
{
  WiFiClientSecure client2;
  client2.setInsecure();
  HTTPClient http2;

  http2.begin(client2, String(SERVER_URL) + "/places/status");
  http2.addHeader("Content-Type", "application/json");
  http2.addHeader("ngrok-skip-browser-warning", "true");

  int httpCode2 = http2.GET();

  if (httpCode2 == 200)
  {
    String payload2 = http2.getString();

    DynamicJsonDocument doc2(1024);
    deserializeJson(doc2, payload2);

    JsonArray places = doc2["places"].as<JsonArray>();

    for (JsonObject place : places)
    {
      String nom = place["nom_place"].as<String>();
```

```
      String statut = place["statut"].as<String>();

      if (statut == "disponible")
      {
        afficherLCD(nom.substring(0, 16), "LIBRE");
      }
      else
      {
        afficherLCD(nom.substring(0, 16), "OCCUPEE");
      }

      delay(2000); // Affiche chaque place pendant 2 secondes
    }

    afficherLCD(
      "Libres:" + String(placesLibres),
      "Occ:" + String(placesOccupees)
    );
  }
  else
  {
    afficherLCD("ERREUR", "CONNEXION");
  }

  http2.end();
}

// ================ BARRIERES ================
void ouvrirBarriereEntree()
{
  servoEntree.write(90);
  afficherLCD("ENTREE", "OUVERTE");
  delay(4000);

  servoEntree.write(0);
  afficherLCD("ENTREE", "FERMEE");
  delay(1000);
}

void ouvrirBarriereSortie()
{
  servoSortie.write(90);
  afficherLCD("SORTIE", "OUVERTE");
  delay(4000);

  servoSortie.write(0);
  afficherLCD("SORTIE", "FERMEE");
  delay(1000);
}
// ================ PLACE LIBRE ================
```

```
int trouverPlaceLibre()
{
  for (int i = 0; i < 4; i++)
  {
    if (!placeOccupee[i]) return i;
  }
  return -1;
}

// ================ RFID ================
void reservationRFID()
{
  afficherLCD("RFID", "SCAN CARD");
  unsigned long startTime = millis();
  while (!rfid.PICC_IsNewCardPresent())
  {
    if (millis() - startTime > 10000)
    {
      afficherLCD("RFID", "TIMEOUT");
      return;
    }
  }
  while (!rfid.PICC_ReadCardSerial()) {}

  int place = trouverPlaceLibre();
  if (place == -1)
  {
    afficherLCD("PARKING", "PLEIN");
    return;
  }

  pinPlaces[place] = String(random(1000, 9999));
  afficherLCD("PLACE:" + String(place + 1), "PIN:" + pinPlaces[place]);
  ouvrirBarriereEntree();
  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();
}

// ================ TICKET ================
void reservationTicket()
{
  int place = trouverPlaceLibre();
  if (place == -1)
  {
    afficherLCD("PARKING", "PLEIN");
    return;
  }

  pinPlaces[place] = String(random(1000, 9999));
  afficherLCD("PLACE:" + String(place + 1), "PIN:" + pinPlaces[place]);
```

```
  //D'ABORD envoyer au backend
  WiFiClientSecure clientT;
  clientT.setInsecure();
  HTTPClient httpT;
  httpT.begin(clientT, String(SERVER_URL) + "/barriere/entree-ticket");
  httpT.addHeader("Content-Type", "application/json");
  httpT.addHeader("ngrok-skip-browser-warning", "true");
  String bodyT = "{\"placeId\":" + String(place + 1) + "}";
  int codeT = httpT.POST(bodyT);
  if (codeT == 200) {
    String payloadT = httpT.getString();
    Serial.println("DB ticket: " + payloadT);
  } else {
    Serial.println("Erreur HTTP ticket: " + String(codeT));
  }
  httpT.end();}

// ================ WIFI ================
void connecterWiFi()
{
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
  }
}

// ================ SETUP ================
void setup()
{
  Serial.begin(115200);
  SPI.begin();
  rfid.PCD_Init();
  Wire.begin();
  lcd.init();
  lcd.backlight();
  randomSeed(analogRead(4));
  connecterWiFi();

  for (int i = 0; i < 4; i++) pinMode(irPins[i], INPUT);

  pinMode(TRIG_IN, OUTPUT);
  pinMode(ECHO_IN, INPUT);
  pinMode(TRIG_OUT, OUTPUT);
  pinMode(ECHO_OUT, INPUT);

  servoEntree.attach(SERVO_ENTREE);
  servoSortie.attach(SERVO_SORTIE);
  servoEntree.write(0);
  servoSortie.write(0);
```

```
  afficherLCD("SMART PARKING", "SYSTEM READY");
}

// ================ LOOP ================
void loop()
{
lcd.setCursor(0, 0);
lcd.print("Bienvenue au park");
  // ================ ENTREE ================
  float distanceEntree = lireDistance(TRIG_IN, ECHO_IN);

  if (distanceEntree < 10 && !entreeDetectee)
  {
    entreeDetectee = true;
    afficherLCD("VOITURE", "A L ENTREE");
    afficherEtatPlacesDB();
    if (placesLibres == 0)
    {
      afficherLCD("PARKING", "PLEIN");
    }
    else
    {
      afficherLCD("1WEB 2RFID", "3TICKET");
      while (Serial.available() == 0) {}
      int choix = Serial.parseInt();
      while (Serial.available()) Serial.read();

      // ================ WEB ================
      if (choix == 1)
      {
        afficherLCD("APP WEB", "ENTREZ PIN");
        while (Serial.available() == 0) {}
        String pinSaisi = Serial.readStringUntil('\n');
        pinSaisi.trim();
        Serial.println("PIN saisi : [" + pinSaisi + "]");
        WiFiClientSecure client;
        client.setInsecure();
        HTTPClient http;
        http.begin(client, String(SERVER_URL) + "/barriere/verifier-pin");
        http.addHeader("Content-Type", "application/json");
        http.addHeader("ngrok-skip-browser-warning", "true");

        String body = "{\"codePin\":\"" + pinSaisi + "\"}";
        Serial.println("Envoi : " + body);

        int httpCode = http.POST(body);
        Serial.println("HTTP Code : " + String(httpCode));

        if (httpCode == 200)
```

```
        {
          String payload = http.getString();
          Serial.println("Reponse : " + payload);

          StaticJsonDocument<256> doc;
          deserializeJson(doc, payload);

          if (doc["success"])
          {
            String place = doc["place"].as<String>();
            afficherLCD("PIN VALIDE", place);
            ouvrirBarriereEntree();
            int numPlace = -1;
            if (place.indexOf("1") >= 0)      numPlace = 0;
            else if (place.indexOf("2") >= 0) numPlace = 1;
            else if (place.indexOf("3") >= 0) numPlace = 2;
            else if (place.indexOf("4") >= 0) numPlace = 3;

            if (numPlace >= 0)
            {
              placeOccupee[numPlace] = true;
              placesOccupees++;
              placesLibres--;
              tempsEntree[numPlace] = millis();
              Serial.println("Place " + String(numPlace + 1) + " marquee
OCCUPEE");
            }



          }
          else
          {
            String msg = doc["message"].as<String>();
            afficherLCD("ERREUR", "PIN INVALIDE");
          }

        }   // ← fin if(httpCode == 200)
        else
        {
          afficherLCD("ERREUR", "SERVEUR");
        }

        http.end();
        delay(500);

      }   // ← fin if(choix == 1)

      // ================ RFID ================
      else if (choix == 2)
```

```
      {
        reservationRFID();
      }
      // ================ TICKET ================
      else if (choix == 3)
      {
        reservationTicket();
      }
    }
  }

  if (distanceEntree > 15) entreeDetectee = false;

  // ================ IR ================
  for (int i = 0; i < 4; i++)
  {
    int etat = digitalRead(irPins[i]);

    if (etat == LOW && !placeOccupee[i])
    {
      placeOccupee[i] = true;
      placesOccupees++;
      placesLibres--;
      tempsEntree[i] = millis();
      afficherLCD("PLACE " + String(i + 1), "OCCUPEE");
    }

    if (etat == HIGH && placeOccupee[i])
    {
      placeOccupee[i] = false;
      placesOccupees--;
      placesLibres++;
      afficherLCD("PLACE " + String(i + 1), "LIBRE");
    }
  }

  // ================ SORTIE  ================
float distanceSortie = lireDistance(TRIG_OUT, ECHO_OUT);
if (distanceSortie < 10 && !sortieDetectee)
{
  sortieDetectee = true;
  Serial.println("\nVehicule sortie");
  afficherLCD("SORTIE", "1WEB 2RFID 3PIN ");
  Serial.println("1-WEB");
  Serial.println("2-RFID");
  Serial.println("3-PIN");

  // Attente du choix du mode
  while (Serial.available() == 0) {}
  int mode = Serial.parseInt();
```

```
  // CRUCIAL : On vide TOUT le tampon (y compris les \n ou \r) après le parseInt
  while (Serial.available() > 0) {
    Serial.read();
  }

  // ================ MODE 1 : SORTIE WEB ================
  if (mode == 1)
  {
    afficherLCD("SORTIE WEB", "ENTREZ PIN");
    // Attente du PIN de l'application
    while (Serial.available() == 0) {}
    String pinSortie = Serial.readStringUntil('\n');
    pinSortie.trim(); // Nettoie les espaces et les retours chariots cachés (\r)

    Serial.println("Code saisi : [" + pinSortie + "]");

    // Si l'utilisateur a envoyé une ligne vide par erreur, on bloque le
traitement HTTP
    if (pinSortie.length() == 0) {
      Serial.println("Erreur : Code saisi vide !");
      afficherLCD("ERREUR", "CODE VIDE");
    }
    else {
      WiFiClientSecure clientS;
      clientS.setInsecure(); // Ignore la vérification stricte SSL pour Ngrok
      HTTPClient httpS;

      // Construction propre de l'URL (gestion du slash final de SERVER_URL)
      String urlCible = String(SERVER_URL);
      if (!urlCible.endsWith("/")) {
        urlCible += "/";
      }
      urlCible += "barriere/verifier-pin-sortie";

      httpS.begin(clientS, urlCible);
      httpS.addHeader("Content-Type", "application/json");
      httpS.addHeader("ngrok-skip-browser-warning", "true");

      String bodyS = "{\"codePin\":\"" + pinSortie + "\"}";
      int codeS = httpS.POST(bodyS);
      Serial.println("HTTP Code : " + String(codeS));

      if (codeS == 200)
      {
        String payloadS = httpS.getString();
        Serial.println("Reponse : " + payloadS);

        StaticJsonDocument<256> docS;
        DeserializationError error = deserializeJson(docS, payloadS);
```

```
      if (!error && docS["success"])
      {
        afficherLCD("CODE VALIDE", "BONNE ROUTE !");
        ouvrirBarriereSortie();
      }
      else
      {
        String msg = docS["message"].as<String>();
        if(msg == "null" || msg == "") msg = "INVALIDE";
        afficherLCD("CODE INVALIDE", msg.c_str());
      }
    }
    else
    {
      afficherLCD("ERREUR", "SERVEUR (" + String(codeS) + ")");
    }
    httpS.end();
  }
  delay(500);
}
// ================= MODE 2 : RFID SORTIE =================
else if (mode == 2)
{
  afficherLCD("RFID", "SCAN CARD");
  while (!rfid.PICC_IsNewCardPresent()) {}
  while (!rfid.PICC_ReadCardSerial()) {}
  afficherLCD("RFID", "VALIDE");
  ouvrirBarriereSortie();
  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();
}
// ================= MODE 3 : PIN SORTIE CLASSIQUE =================
else if (mode == 3)
{

  afficherLCD("ENTRER", "PIN");

  while (Serial.available() == 0) {}
  String pin = Serial.readStringUntil('\n');
  pin.trim();

  bool pinValide = false;

  for (int i = 0; i < 4; i++)
  {
    if (pin == pinPlaces[i])
    {
      pinValide = true;
      unsigned long duree = (millis() - tempsEntree[i]) / 1000;
```

```
        float montant = duree * 0.05;

        Serial.println("===== FACTURE =====");
        Serial.print("Place : "); Serial.println(i + 1);
        Serial.print("Temps : "); Serial.print(duree); Serial.println(" sec");
        Serial.print("Montant : "); Serial.println(montant);

        afficherLCD("MONTANT:", String(montant) + " DA");
        ouvrirBarriereSortie();
        break;
      }
    }

    if (!pinValide)
    {
      afficherLCD("ERREUR", "PIN INCORRECT");
    }


    while (Serial.available() > 0) Serial.read();
  }
}

if (distanceSortie > 15) sortieDetectee = false;

delay(1000);
}
```