الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

Mémoire de Projet de Fin d'Études

Présenté par

Imene Feriel FERHAH
&
Sarah NAHMAR

Pour l'obtention du diplôme de Master en Électronique

Spécialité : Electronique des Systèmes Embarqués

Thème

Interface de Communication intelligente entre Parkings

Sujet Proposé par :

Dr. Djamel BOUCHAFFRA, Directeur de Recherche, CDTA, Alger
Dr. Abdelkrim ALLAM, Maître de Recherche A, CDTA, Alger
Dr. Fayçal YKHLEF, Maître de Recherche A, CDTA, Alger

Co-promoteur:
Pr. Farid YKHLEF, Université SAAD DAHLAB, BLIDA

Année universitaire : 2021-2022

REMERCIEMENTS

En préambule à ce mémoire, nous remercions *ALLAH* qui nous a aidé et donné la patience et le courage durant cette longue d'année d'étude.

Nos remerciements vont tout d'abord à nos promoteurs du CDTA : Dr. Djamel BOUCHAFFRA, Directeur de Recherche, Dr. Abdelkrim ALLAM, Maitre de Recherche A, et Dr. Fayçal YKHLEF, Maitre de Recherche A pour nous avoir proposé ce sujet, pour leurs encadrements, le temps qu'ils ont consacré à la correction et à la relecture de ce document ainsi que leurs conseils précieux qui nous ont permis d'aboutir à l'élaboration de ce mémoire.

Nous remercions très chaleureusement notre Co-promoteur Pr. Farid YKHLEF de l'Université SAAD DAHLEB pour sa confiance et ses conseils.

Nous remercions aussi Monsieur Abdeladhim DERBAL, ingénieur au CDTA, pour son aide technique et conseils précieux.

Nous présentons nos respects et nos sincères remerciements aux membres du jury qui nous ont fait l'honneur d'avoir accepté de faire partie du jury et de nous avoir consacré de leur temps précieux.

Merci à tous qui ont participé de près ou de loin à l'aboutissement de ce travail.

DEDICACE

Je dédie ce modeste travail

A mes chers parents,

Que nulle dédicace ne puisse exprimer ce que je leur dois, pour leur bienveillance, leur affection et leur soutien.

Trésors de bonté, de générosité et de tendresse, en témoignage de mon profond amour et ma grande reconnaissance « que DIEU vous garde ».

A ma chère sœur Lina et A mon cher frère Anis,

Je leur dédie ce modeste travail en témoignage de mon grand amour et gratitude infinie.

A toutes les personnes que j'aime et qui m'aiment,

Pour leur aide et leur soutien moral durant l'élaboration du travail de fin d'études.

A mon binôme Sarah,

Avec qui j'ai partagé des moments inoubliables et avec qui j'ai pu réaliser ce travail grâce à son aide, sa compréhension et sa présence.

Imene Feriel

DEDICACE

C’est avec un grand plaisir que je dédie ce travail :

A mon cher père Abdellah et ma chère mère Yamina,

Qui sont les meilleurs parents dans ce monde, c’est grâce à leur douaa, leur soutien moral, leur encouragement et leur confiance en moi que j’ai pu mener à terme ce projet. Tout en témoignant de ma profonde gratitude et de mon incontestable reconnaissance pour tous leurs sacrifices et tout l’amour dont ils m’entourent.

A ma chère sœur Assia et A mon cher frère Abderrahim,

Je leur dédie ce modeste travail en témoignage de mon grand amour et ma gratitude infinie.

A la famille Nahmar, la famille Ouaad et mes chers proches,

Qui étaient toujours à mes cotes et m’ont été d’un soutien précieux. Je leur souhaite à eux aussi plein de succès dans leur vie.

A mon binôme Feriel,

Avec qui j’ai partagé des moments inoubliables et avec qui j’ai pu réaliser ce travail grâce à son aide sa compréhension et sa présence.

Que DIEU les protège tous.

Sarah

RESUME

ملخص :يتمثل الهدف من عملنا في تصميم وتحقيق منصة الاتصال الذكي بين مواقف السيارات. هندسة الاتصالات التي اخترناها هي شبكة GSM لأنها ملائمة أكثر لبلدنا. يتم استخدام لوحات Arduino وواجهات GSM/SIM900كوسيلة نقل للبيانات من مواقف السيارات المختلفة .
تم أخذ ثلاث مواقف للسيارات كمثال في دراستنا. تعمل هذه المنصة في وضع يدوي وآلي. كما يتم استخدام شاشة LED كبيرة مقاومة للماء لعرض الأماكن الفارغة في المواقف الثلاث. وتم استخدام برنامج MATLAB R2020bفي تطبيقنا.
الكلمات الرئيسية: إدارة وقوف السيارات، شاشة LED مقاومة للماء، شبكة GSM، نظام الخبراء، Arduino، Matlab

Résumé :
Le but de notre travail consiste à concevoir et à réaliser une plateforme de communication intelligente entre parking. L'architecture de communication des données que nous avons choisie est le réseau GSM vu qu'elle est plus adéquate pour notre pays. Des cartes Arduino et des interfaces GSM/SIM900 sont utilisées comme support de transmission des données provenant des différents parkings. Trois parkings ont été pris comme exemple dans notre étude. Ce nombre peut être étendu selon le besoin. Cette plateforme fonctionne en mode manuel et automatique. Les règles de décision ont été utilisées pour automatiser notre système. Un écran étanche LED de grande dimension est utilisé pour l'affichage des places vides dans les trois parkings. Le logiciel MATLAB R2020b a été utilisé dans notre application.
Mots clés : Gestion de Parkings, Ecran LED étanche, réseau GSM, système expert, Arduino, Matlab

Abstract:

The aim of our work is to design and implement a smart parking communication platform. The data architecture we have chosen is the GSM network since it is more appropriate for our country. Arduino boards and GSM/SIM900 interfaces are used to transmit and receive data. Three parkings were used in our study. This number can be extended if needed. The platform operates in manual and automatic mode. A rule-based strategy is adopted to automatize the process. A large LED waterproof screen is used to display empty spaces in three car parks. MATLAB R2020b software has been used in our study.
Keywords: Parking management, waterproof LED screen, GSM network, expert system, Arduino, MATLAB
Keywords: Parking management, waterproof LED screen, GSM network, expert system, Arduino, MATLAB

LISTE DES ACRONYMES ET ABREVIATIONS

GSM: *Global System for Mobile communications*

IDE: *Integrated Development Environment*

IOT: *Internet Of Things*

LPR: *License Plate Recognition*

SBR : *Système à Base des Règles*

SIM: *Subscriber Identity Module*

USB: *Universal Serial Bus*

TABLE DES MATIERES

LISTE DES FIGURES

LISTE DES TABLEAUX

Introduction générale

La gestion des parcs de stationnement est une tâche très importante. Plusieurs méthodologies de gestion ont été proposées dans la littérature. Cette gestion inclue, la sécurité des parcs, la détection des places disponibles et la localisation des parkings dans les grandes villes. Cependant, la communication entre parkings, qui consiste à échanger les informations des lieux de stationnement entre les postes de contrôle, a rarement été abordée dans la littérature. Nous proposons et implémentons dans ce projet une méthodologie de communication entre trois parkings en utilisant le réseau téléphonique.

La plupart des méthodologies proposées en pratique pour la gestion des parcs de stationnement se basent sur l'internet des objets (en anglais IOT : Internet of Things) pour transmettre et recevoir les informations des lieux de stationnement. Cependant, en Algérie, nous avons remarqué que les technologies IOT sont rarement exploitées par les propriétaires des parcs de stationnement. Ceci est dû à plusieurs raisons. On peut citer :

- La nature anarchique des lieux de stationnement dans la plupart des villes ne permet pas d'installer des technologies de pointes pour gérer les lieux.
- L'exploitation des technologies IOT nécessite une connexion internet stable qui n'est pas toujours assurée en Algérie.
- Les technologies IOTs sont très couteuses. Certaines propriétaires des parcs n'acceptent pas d'investir sur ce genre de solution.

Les solutions de communication entre plusieurs parkings en Algérie doivent tenir compte de ces contraintes. De ce fait, nous proposons une solution de communication à base du réseau téléphonique mobile. Les composants utilisés pour implémenter solution ne sont pas couteuses. Des règles de gestions ont été proposées pour automatiser notre système.

Le travail réalisé dans ce mémoire est un module important de la plateforme de gestion des parkings qui est en cours d'élaboration au CDTA. La solution que nous avons proposée va être testée en pratique et ensuite intégrée dans la plateforme.

Le module que nous avons implémenté a pour objectif de communiquer entre parkings l'état des places de stationnement en temps réel.

Chapitre 1 : PARKINGS INTELLIGENTS

1.1 Introduction

Dans ce chapitre nous présentons des généralités sur les parcs de stationnement et les technologies incorporées dans les parkings intelligents. Nous présentons les différences entre les lieux de stationnement ordinaires et intelligents. Nous donnons un historique succin sur l'évolution des parcs de stationnement. Nous citons par la suite quelques travaux sur les méthodes de gestion des parkings.

1.2 Définitions

1.2.1 Parking ordinaire

Un parking est un air aménagé, destiné pour le stationnement des véhicules à l'intérieur ou à l'extérieur de la ville pour une durée déterminée ou indéterminée et ce dans la sécurité absolue.
Les avantages et les inconvénients des parkings ordinaires sont données comme suit [1]:
Avantages :

- Moins de circulation dans la ville,
- Pas de stationnement anarchique,
- Moins de pollution,

Inconvénients :

- Perte du temps pour la localisation de l'endroit exacte du parking,
- Une gestion qui ne fournit aucun détail sur le nombre des places de stationnement libres,
- Les agents de sécurité assurent le fonctionnement des espaces de stationnement d'une façon permanente,

Les inconvénients de ces parcs ont obligé les managers des villes à réfléchir à des solutions automatiques de gestion. Selon une étude menée dans le monde entier, il s'est avéré qu'environ 30 à 50 conducteurs passent plus de 14 minutes dans la recherche d'une place pour stationner. Cela mène à la frustration, aux accidents et à la perte d'opportunité professionnelle [1]. C'est dans ce concept et avec l'arrivé de l'internet des objets, les chercheurs ont pu instaurer un nouveau système plus performent et efficace qui permet de changer la gestion archaïque de ces anciens sites en proposant des solutions intelligentes.

1.2.2 Parking intelligent

C'est un espace de stationnement des voitures qui utilise des technologies avancées pour assurer une gestion intelligente de ces lieux. De nombreuses applications de stationnement intelligent en ligne ont été développées. Nous citons l'exemple de détermination des directions de navigation pour la réservation des places de stationnement vides. Ces places peuvent être réservées par paiement en ligne via une application mobile ou une interface Web. Une fois le paiement fait, la place de parking est réservée et l'utilisateur recevra un code de référence qui peut être utilisé pour avoir une autorisation d'accès au parking [2].

De plus, l'utilisation de la mobilité intelligente pour la localisation du lieu exacte des parkings a été aussi exploitée pour le développement des technologies de pointes [3].

Tenant compte de ces technologies avancées, on peut assurer moins de pollution et beaucoup de fluidité dans la circulation. Les villes détenant ces parkings ultras modernes permettent d'attirer des touristes et ainsi offrent une économie positive.

Figure 1.1: Parking Intelligent [4]

1.3 Historique

Tout a commencé lorsque les gens ont appris à se déplacer en utilisant autre chose que leurs pieds. Il y a des milliers d'années, le besoin d'un endroit spécial pour stocker les moyens de transport s'est fait sentir. L'exemple le plus simple de stationnement ancien est le rail d'attelage que les cavaliers utilisaient pour attacher leurs chevaux fatigués. Au fil du temps, des voitures et des autocars automoteurs sont apparus, nécessitant plus d'espace pour leur stockage. En conséquence, des écuries ont été créées : l'équivalent médiéval des garages et des parkings.

A la fin du **19ème siècle**, les automobiles commencent à apparaître en masse dans les rues des grandes villes. Au début, ils étaient stationnés le long des rues. Cependant, la croissance démographique et le chaos résultant de la popularité croissante des automobiles nécessitaient des endroits spéciaux pour se stationner toute la journée pendant que les gens travaillaient au bureau ou dormaient. Initialement, des écuries vides ont été utilisées. Cependant, ils n'avaient pas assez d'espace pour entreposer les automobiles qui se sont avérées plus abordables et plus faciles à utiliser que les calèches. Les écuries n'étaient pas non plus situées là où les chauffeurs le souhaitaient.

Pour cette raison, des aires de stationnement spécialement conçues pour les automobiles ont commencé à apparaître au début du 20eme **siècle**, notamment des parkings extérieurs, à plusieurs étages et souterrains.

Le premier parking fermé à plusieurs étages a été introduit à Londres par **City and Suburban Electric Carriage Company** [5], qui fabriquait des véhicules électriques. En **1901**, l'entreprise construit un bâtiment de sept étages pour garer 100 voitures. Elle a ensuite construit deux autres parkings de 230 et 200 places de stationnement. Le premier parking souterrain a été construit à Barcelone sous la Casa Mila, conçue par l'architecte **Antoni Gaudi** [6]**.**

C'était la première tentative de mécaniser un processus de stationnement ; les voitures ont été soulevées aux niveaux supérieurs à l'aide d'ascenseurs de service. Cependant, des voituriers étaient utilisés pour garer une voiture dans la bonne place de stationnement [7].

Figure 1.2: Ancien Parking [8]

En **1928**, le premier panneau bleu [P] a été créé par monsieur **Lucien Bourier** [9] à- l'occasion des jeux Olympiques [1].

1.4 Technologies incorporées dans les parkings intelligents

Pour occuper les places de stationnement vacantes dans les parkings et concevoir des systèmes de stationnement, plusieurs technologies sont exploitées. Nous citons quelques exemples :

1.4.1 Système de positionnement global

Le système de positionnement global, nommée en anglais « *Global Positioning System »* (GPS), est une technologie très utile pour la conception des systèmes de stationnement. Il permet de choisir l'itinéraire le plus court et l'emplacement du conducteur en temps réel. Le GPS est assujetti aux erreurs lorsque les signaux sont bloqués à cause des murs séparateurs à l'intérieur du bâtiment. Ce système est adapté beaucoup plus pour les Parkings extérieurs ouverts car il y a moins de risques de blocage du signal par rapport aux stationnements intérieurs. Sa précision dépend complètement de la disponibilité du signal par satellite [1].

1.4.2 Vision par ordinateur et intelligence artificielle

La vision par ordinateur a été largement utilisée pour concevoir des technologies de gestion des parkings. On peut citer l'exemple de la reconnaissance des plaques d'immatriculations et la détection des places vides dans un grand parking. La reconnaissance des plaques d'immatriculation en utilisant la technologie LPR (*Lisence Plate Recognition*) permet d'identifier l'occupation du parking. Les caméras, placées près de l'entrée du parking, sont utilisées pour acquérir les images des matricules en temps réel. Des algorithmes de reconnaissance de formes et traitement d'images sont exploités pour reconnaitre les matricules [10].Cette technologie permet de sauvegarder les entrées-sorties des véhicules et ainsi estimer le nombre de véhicules occupant le parking à un temps donné. La détection des places vides dans un parking est généralement basée sur les techniques d'apprentissage profonde (*Deep Learning*). Nous citons l'algorithme YOLO (*You Only Look Once*) [11].

1.4.3 Systèmes multi-agents

Ce sont des systèmes de stationnement conviviaux qui permettent aux utilisateurs, selon leurs préférences, de sélectionner une place de stationnement à l'aide d'applications mobiles ou Web. Plusieurs supports sont utilisés pour concevoir ces systèmes ; nous citons, les caméras visuelles,

des capteurs, et les applications mobiles. L'utilisateur reçoit également des informations de navigation pour accéder à la place du parking [12]. Ces systèmes sont adaptés pour les parkings ouverts et fermés.

1.5 Etat de l'art sur les méthodes de gestion des parkings intelligents

Dans cette section, nous présentons un état de l'art sur les méthodes de gestion des parcs de stationnement. Nous abordons les solutions proposées dans la littérature pour la communication entre parkings, la sécurité des parcs et la détection des places disponibles dans les lieux de stationnement.

Yao-Huei Huang et all. [13] ont élaboré un système intelligent d'aide à la décision pour détecter les places de stationnement disponibles en bord des routes urbaines. L'objectif principal de cette solution est de réduire le temps de recherche des places de stationnement appropriées dans les rues animées de la ville. Le système proposé peut réduire les coûts d'installation et de maintenance par rapport aux systèmes existants. Le stationnement intelligent comprend les modules suivants : (i) Utilisation des capteurs pour acquérir les images de stationnement dans la rue puis le renvoi d'informations au serveur de la base de données, (ii) Exploitation d'un algorithme de détection d'objets (un réseau de neurones conventionnel) pour déterminer si les places de stationnement dans la rue sont disponibles ou non, (iii) une application de stationnement intelligent est développée pour que le conducteur obtienne les informations de stationnement en temps réel à l'aide d'un appareil mobile (iv) et le comptage du nombre de places de stationnement disponibles. Les expériences numériques confirment que les algorithmes intégrés dans le système sont efficaces par rapport aux systèmes commerciaux existants.

Robert Vincent Racunas, Jr. [14] a proposé un système de gestion des parcs qui inclut un dispositif de communication sans fils capable d'accéder à un serveur sur internet et à une application logicielle. Cette application est utilisée pour recevoir les données issues des parkings, tel que le nombre de places vides. Ces données sont transmises vers le dispositif de communication via internet. Le dispositif permet de communiquer les donnes des parcs entre plusieurs parkings en temps réel.

M. Venkata Sudhakar et al [15] ont élaborés une application mobile pour identifier les espaces de stationnement libres dans une ville intelligente. L'application mobile fournit aussi des alertes

en cas d'incendies ou de fuites de gaz. Un écran LCD (*Liquid Crystal Display*) est situé à l'entrée du parking pour afficher le nombre de places disponibles. Des capteurs de proximité infrarouges sont utilisés pour identifier l'existence d'un véhicule à la porte d'entrée du parc. Un LPR est exploité pour reconnaitre le matricule du véhicule et sauvegarder la date et l'heure de son accès.

S. Aravinth Kumar et all [16] ont élaboré un système qui permet la réservation des places de stationnement dans les centres commerciaux. Le système comprend deux modules. Le premier permet la réservation préalable des places de stationnement pour les clients. Le deuxième module propose une interface de gestion des places de stationnement pour les vendeurs/fournisseurs.

Le système permet la réservation des places après avoir effectué le paiement en ligne. L'utilisateur peut se connecter à l'interface de stationnement en utilisant son téléphone. Les statistiques de stationnement peuvent être mesurées pour chaque place de stationnement.

1.6 Conclusion

Dans ce chapitre, nous avons présenté une vue globale sur les parcs de stationnement et les technologies incorporées dans les parkings intelligents. Nous avons présenté un historique sur l'évolution des lieux de stationnement dans le monde. Un état de l'art sur les méthodes de gestion des parkings a été aussi présenté. Nous avons trouvé que peu de travaux sur la communication entre parkings ont été reportés dans la littérature. Cette remarque nous a motivé à proposer une méthodologie de communication intelligente entre plusieurs parcs de stationnement.

Chapitre 2 : Approche proposée pour la communication intelligente entre parkings

2.1 Introduction

L'objectif de ce chapitre est de présenter l'approche d'interaction entre plusieurs parkings. Cette tâche est réalisée en équipant chaque place de parking d'une interface de communication, une carte électronique GSM de type Sim 900, une carte ARDUINO UNO et un grand écran pour affichage. Les outils utilisés pour concevoir cette solution sont présentés dans ce chapitre. Une conclusion terminera ce chapitre.

2.2 Interface de communication

Nous proposons dans ce projet de fin d'étude de communiquer le nombre de places de stationnement disponibles entre trois parkings situés dans la même région. Le système que nous proposons est composé des cartes Arduino munis de circuits GSM. De plus, notre système se base sur le réseau mobile pour transmettre et recevoir les données entre parkings.

2.2.1 Problématique

La plupart des solutions proposées dans la littérature pour la gestion des parcs intelligents, incluant le volet de la communication entre parkings, se basent sur l'internet des objets (en anglais IOT : *Internet of Things*) pour transmettre et recevoir les informations des parkings [13], [14], [15], [16]. Cependant, dans le contexte Algérien, nous avons remarqué que les technologies IOT sont rarement exploitées par les propriétaires des parcs de stationnement. Ceci est dû à plusieurs raisons. On peut citer les raisons suivantes :

- Les parcs de stationnement des voitures n'existent pas dans toutes les villes en Algérie. Ce fait, obligent les conducteurs des voitures de garer sur les trottoirs. Ces lieux sont considérés comme des parcs de stationnement dans certaines villes en Algérie (ex. La ville de Boufarik). Ces parcs spéciaux sont nommés dans notre mémoires des Parkings anarchiques (Figure 3.1). L'incorporation des technologies de pointes est impossible dans ces lieux. La solution de communication entre plusieurs parkings en Algérie doit tenir compte de la structure des parking anarchiques.

- Les parcs de stationnement existants dans plupart des grandes villes en Algérie (ex. Alger ou Oran) doivent être connectés au réseau internet pour permettre l'exploitation des IOT. Cependant, le volet de sécurité des données doit être pris en considération. De plus, la connexion internet en Algérie n'est pas toujours fiable. Des perturbations peuvent surgir dans certaines circonstances et ainsi perturbent les solutions de communication entre parkings à base de internet. Nous citons les exemples suivants : (i) conditions météorologiques défavorables, (ii) les heurs ou l'utilisation du réseau internet est excessive, (iii) et les périodes des épreuves nationales (ex. baccalauréat).
- Les solutions IOT pour la gestion des parkings sont très couteuses. Certaines propriétaires des parcs n'acceptent pas d'investir sur ce genre de solution.

Figure 2.1: Parking anarchique en Algérie [17]

Nous proposons dans notre PFE une solution de communication intelligente entre parkings en utilisant le réseau téléphonique mobile. Les avantages d'un réseau téléphonique par rapport à d'autres solutions sont :

- L'utilisation du réseau téléphonique est moins couteuse si on la compare au réseau internet,
- Le réseau téléphonique couvre plusieurs régions en Algérie,
- Il est moins vulnérable au piratage par rapport à l'internet,
- Les solutions de communication entre parkings qui se basent sur le réseau téléphonique sont moins chères que celles basées sur les IOTs.

2.2.2 Solution proposée

2.2.2.1 Vue globale

Le schéma de communication que nous proposons est décrit comme suit :

- Le positionnement de trois ordinateurs au niveau des postes de contrôles de trois parkings situés dans des régions différentes,
- Le troisième parking peut être un parking anarchique,
- La communication entre ces parkings est basée sur des règles. Elle s'effectue par le billet de cartes électroniques de type ARDUINO UNO munis de circuits GSM de type SIM900,
- La communication avec le parking anarchique est effectuée en utilisant un téléphone portable,
- L'émission et la réception des informations recueillies portant sur la disponibilité de places vides au niveau des trois parkings est affiché sur l'interface graphique de l'utilisateur,
- Les messages sont envoyés selon la demande pour connaitre l'état des places disponibles dans les différents parkings,
- Le nombre de places vides est aussi affiché en temps réel sur un grand écran Led situé à l'entrée du parking,

Figure 2.2: Schéma de communication entre trois parkings

2.2.2.2 Règles de communication

L'organigramme explicatif du système de communication à base des règles est schématisé sur la Figure 2.3.

Il est important de rappeler que la communication avec un parking anarchique nécessite l'utilisation d'un téléphone portable pour la communication vu que ce type de parking ne possède pas de poste de contrôle officiel. L'utilisation d'un smartphone, muni la norme LTE (*Long Term Evolution*) et la connexion 4G, n'est pas obligatoire vu que la communication se base seulement sur le réseau téléphonique.

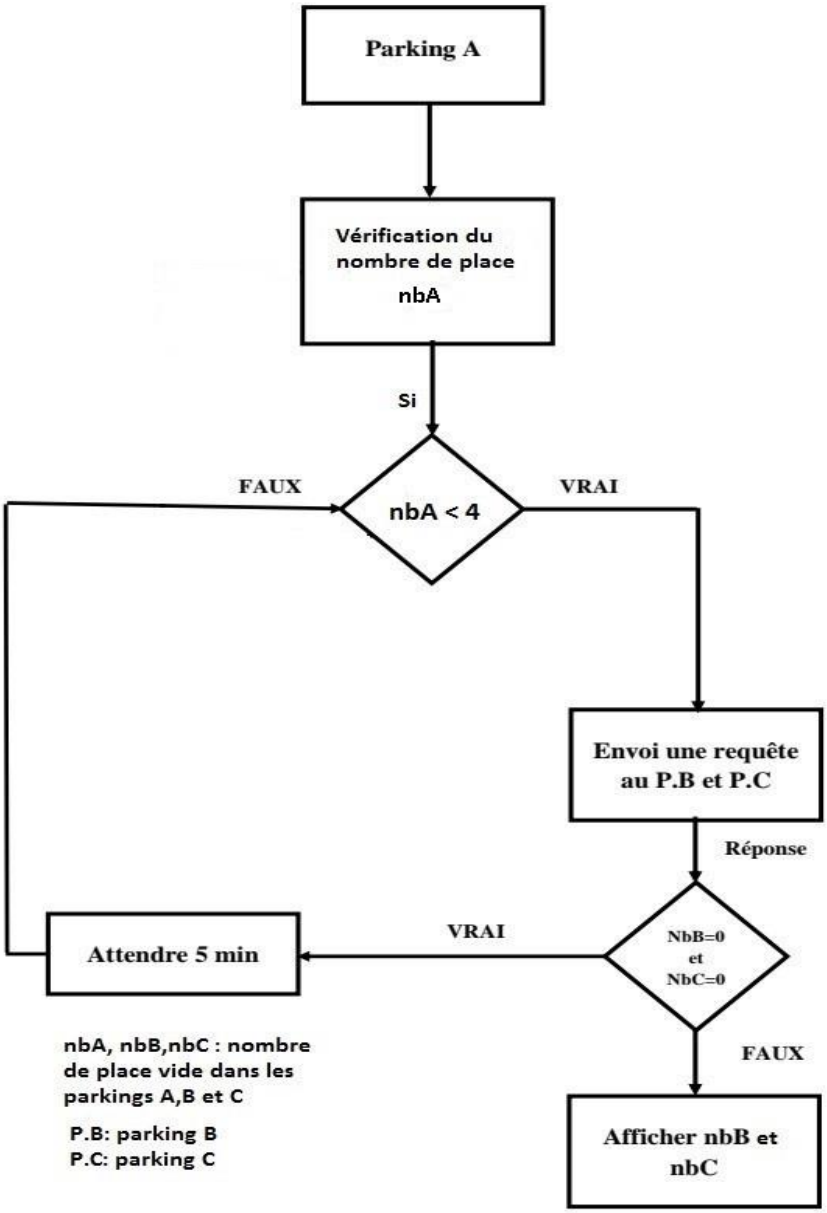

Figure 2.3: Organigramme explicatif du système de communication

Nous avons choisi les systèmes à base des règles (SBR) pour deux raisons. Premièrement, ce sont les systèmes à base de connaissances les plus répandus dans les applications pratiques. Deuxièmement, les représentations à base de règles mènent à une codification des connaissances en des termes assez simples, les règles si... alors... Il s'agit d'une façon d'exprimer les connaissances qui nous est familière. De plus, les SBR offrent des éditeurs de règles qui facilitent la codification des connaissances d'une application.

Dans cette section nous allons expliquer l'ensemble des méthodes et des règles permettant de collecter et mettre en forme la communication entre plusieurs parkings.

Toutes les parties d'un système à base des règles doivent travailler en collaboration pour que l'objectif de l'ensemble soit atteint. De ce point de vue, on peut définir la finalité d'un SBR comme étant la possibilité de remplacer un expert d'un domaine bien identifié.

Nous avons " 3 parkings " chaque parking est doté par une machine (PC ordinateur) relié avec une carte Arduino munie d'une carte GSM SIM900. Nous avons sélectionné un seul parking anarchique. Dans chaque écran, nous avons conçu une interface graphique en utilisant le logiciel Matlab. Cette interface permet d'afficher le nombre de places vides et la zone géographique de chaque parking. Ces données peuvent être analysés par les gestionnaires des parkings en vue gérer la communication entre parkings.

En application l'organigramme de la Figure 2.3., plusieurs scénarios sont envisagés.

Nous citons l'exemple suivant :

Dans le parking A, si trois places sont vidées, le système envoi une requête vers le parking B et C pour demander le nombre des places disponibles. Le système des parking B et C envoi automatiquement le nombre exact des places vides. Ces informations vont être affichées sur l'interface et l'écran LED. Si les parkings sont pleins, c.a.d. NbB=0 et NbC=0, le système attend 5 minutes puis on revoie une autre requête.

2.2.2.3 Description des composants et logiciels

a) Composants électroniques

a.1) Carte Arduino

Arduino est un projet créé par une équipe de développeurs, composée de six individus : Massimo Banzi, David Courteilles, Tom Igoe, Gianluca Martino, David Mellis et Nicholas Zambetti [18].

C’est un outil qui permet aux débutants, amateurs ou professionnels de créer des systèmes électroniques plus ou moins complexes.

Cette carte électronique constitue :

- **Le matériel :** cartes électroniques dont les schémas sont en libre circulation sur internet,
- **Le logiciel :** gratuit et open source, développé en Java, dont la simplicité d'utilisation relève du savoir cliquer sur la souris,

- Matériel

La carte Arduino est un circuit électronique qui peut être programmé et qui permet de faire le pont entre le monde virtuel de la programmation informatique (concepts formels et logiques) et le monde physique (interaction électromécanique des objets). Elle repose sur un circuit intégré (un mini-ordinateur appelé également microcontrôleur basée autour d'un microcontrôleur ATMEGA du fabricant (Atmel) associée à des entrées et sorties qui permettent à l'utilisateur de brancher différents types d'éléments externes :

- ✓ Entrées, des capteurs qui collectent des informations sur leur environnement comme la variation de température via une sonde thermique, le mouvement via un détecteur de présence ou un accéléromètre, le contact via un bouton-poussoir, etc.
- ✓ Sorties, des actionneurs qui agissent sur le monde physique telle une petite lampe qui produit de la lumière, un moteur qui actionne un bras articulé, etc.

Cette carte électronique peut être autonome et fonctionne sans ordinateur. Elle peut servir d'interface avec les ordinateurs.

Il existe plusieurs types de cartes Arduino. Dans notre projet, nous utilisons la carte Arduino UNO (Figure 2.4 et Tableau 2.1).

Figure 2.4: Carte Arduino UNO [18]

Tableau 2-1 : Caractéristiques de la carte Arduino Uno [18]

Désignation	Description
Microcontrôleur	ATmega328
Tension d'alimentation interne	5V
Tension d'alimentation (recommandée)	7 à 12V, limites =6 à 20V
Entrées/sorties numériques	14 dont 6 sorties PWM
Entrées analogiques	6
Courant max par broches E/S	40 Ma
Courant max sur sortie 3,3 V	50 Ma
Mémoire Flash	32 KB dont 0.5 KB utilisée par le bootloader
Mémoire SRAM	2 KB
Mémoire EEPROM	1 KB
Fréquence horloge	16 MHz
Dimensions	68.6mm x 53.3mm

La carte s'interface au PC par l'intermédiaire de sa prise USB (*Universel Serial Bus*). La carte s'alimente par le jack d'alimentation (utilisation autonome) mais peut être alimentée par l'USB (en phase de développement par exemple) [18].

- Logiciel

Le logiciel Arduino est gratuit (open source) et se télécharge sur le site officiel d'Arduino : http://Arduino.cc/en /Main/Software

Plusieurs fichiers différents sont proposés en téléchargement en fonction du système d'exploitation : Windows, MacOs, Linux. La dernière version officielle est généralement celle qu'il faut sélectionner, bien que pour les cartes d'Arduino les plus récentes il est parfois préférable d'opter pour les versions « beta » du logiciel, c'est-à-dire des versions non éprouvées et perfectibles, mais tout de même fonctionnelles [19].

L'IDE (*Integrated Development Environment*) est un programme spécial exécutable sur ordinateur qui permet d'écrire des esquisses pour la carte Arduino dans un langage simple mais évolué (langage C). C'est un éditeur qui permet d'écrire et de télécharger les esquisses à la carte. Le code écrit en langage C est traduit par le compilateur avr-gcc en langage machine compris par le microcontrôleur. Cette dernière étape est très importante car elle facilite la programmation des microcontrôleurs et les rends moins complexité à utiliser [20].

L'interface « IDE » d'Arduino se présente selon la fenêtre ci-dessous :

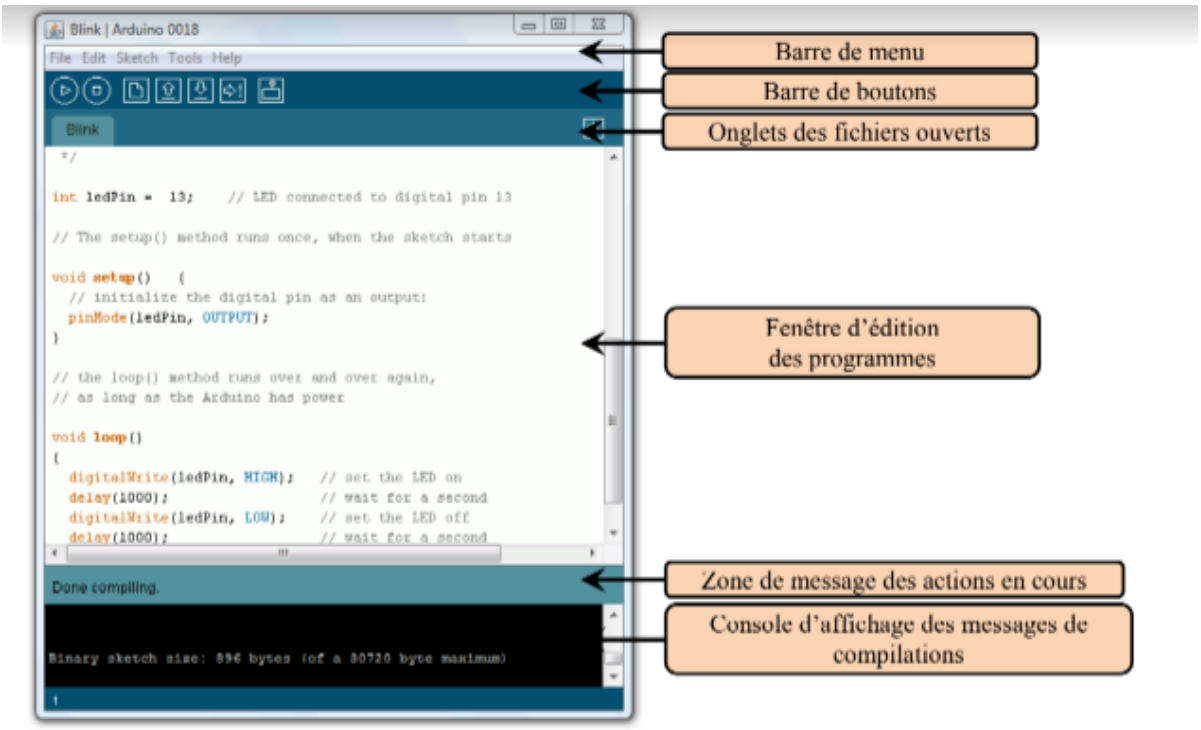

Figure 2.5: Interface du logiciel IDE pour Arduino [20]

On peut écrire du texte et des chiffres dans cette fenêtre. Celle-ci contient également des menus, des boutons, des alertes spéciales et des fonctions de contrôles. La liste des fonctions est :

Nouveau : crée une nouvelle esquisse.

Ouvert : présente un menu de toutes les esquisses contenues dans le dossier.

Enregistrer : enregistre l'esquisse.

Vérifiez : permet de vérifier que le programme est exempt d'erreurs de syntaxe.

Envoyez : permet de vérifier et de télécharger un programme à la carte l'Arduino si aucune erreur d'orthographe ou de mise en forme n'est trouvée.

Serial Monitor : permet d'ouvrir le moniteur de série et d'afficher les informations en provenance du port série sur l'Arduino. Le moniteur de série est un outil pour dialoguer avec la carte Arduino [46].

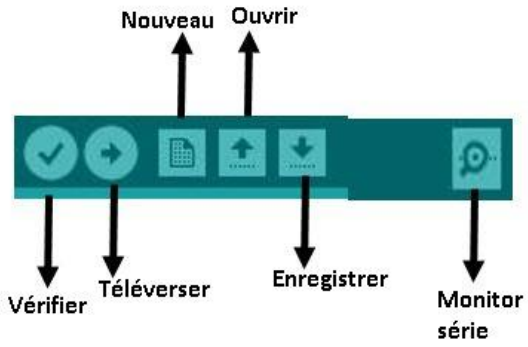

Figure 2.6: Détail de la barre de bouton

- **Avantages de la carte Arduino**
  - ✓ Pas cher,
  - ✓ Environnement de programmation (IDE) clair et simple,
  - ✓ Multiplateforme : tourne sous Windows, Macintosh et Linux,
  - ✓ Nombreuses librairies disponibles avec diverses fonctions implémentées,
  - ✓ Logiciel et matériel open source extensible,
  - ✓ Nombreux conseils, tutoriaux et exemples en ligne (forums, site personnel etc.…),

- ✓ Existence de « Shield » ou interface : ce sont des cartes supplémentaires qui se connectent sur le module Arduino pour augmenter les possibilités comme par exemple : afficheur graphique couleur, Interface Ethernet, GPS, etc...

a.2) Carte GSM

La carte GSM SIM900 est un modem GSM, qui peut être intégré dans un grand nombre de projets. Nous pouvons utiliser cette carte pour accomplir presque tout ce que le téléphone portable normal peut faire (Figure 2.7).

Figure 2.7: Carte GSM SIM900 [21]

La carte GSM SIM900 est conçue pour entourer la puce SIM900 avec tout le nécessaire pour interface avec la carte Arduino, des fonctions supplémentaires et des caractéristiques uniques de la puce.

Faisons connaissance des caractéristiques et des capacités de cette carte, un aperçu général faisant ressortir les détails bien déterminés est donné ci-dessous (Voir les Figures 2.8 et 2.9) :

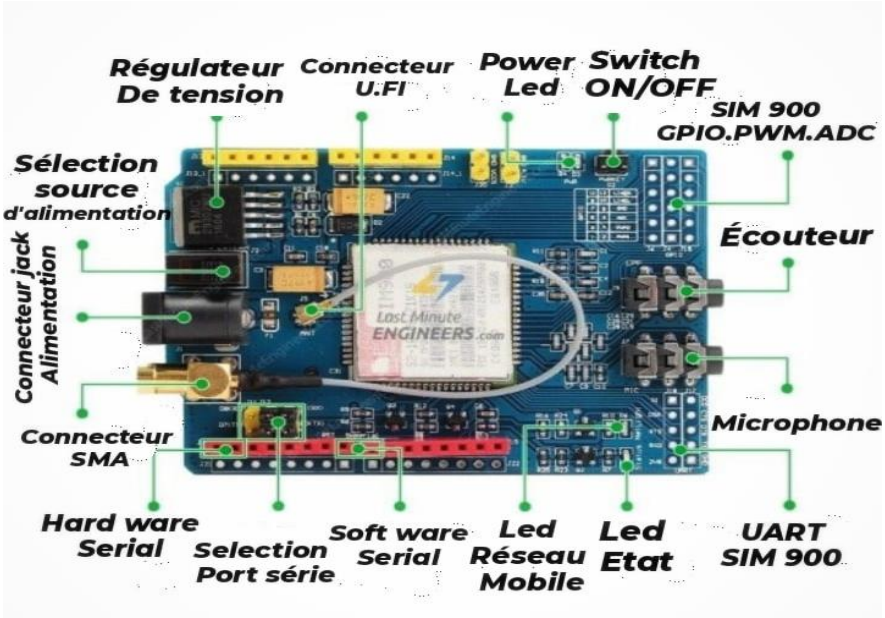

Figure 2.8: Carte GSM SIM900 (face) [21]

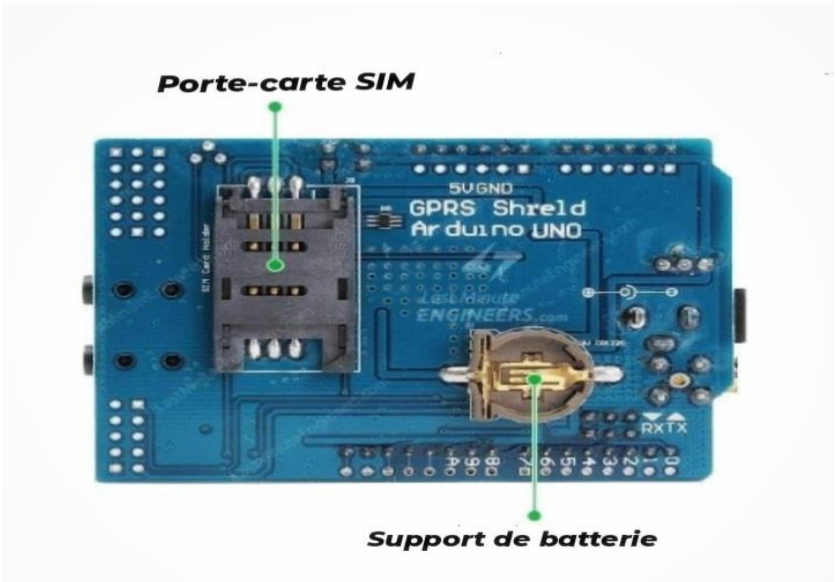

Figure 2.9: Carte GSM SIM900 (pile) [21]

Les fonctionnalistes de cette carte sont :

• Connecter à n'importe quel réseau GSM mondial avec n'importe quelle carte SIM,

• Passer et recevoir des appels vocaux à l'aide d'un écouteur externe et d'un microphone,

• Envoyer et recevoir des SMS,

• Scanner et recevoir des émissions de radio FM,

• Jeu de commandes AT basé sur série,

• Connecteurs U. FL et SMA pour antenne cellulaire,

• Accepte la carte SIM pleine taille.

Il y a une prise SIM à l'arrière. Toute carte SIM pleine grandeur 2G activée fonctionnerait parfaitement.

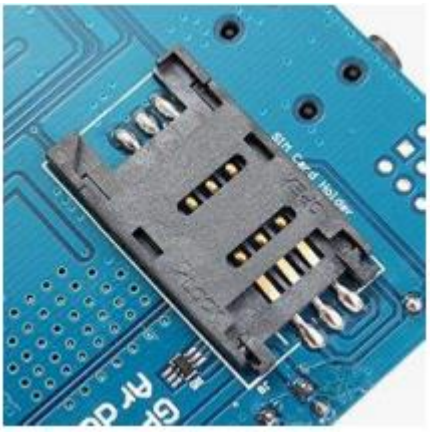

Figure 2.10: Prise SIM [21]

La carte SIM900 est accrocher à la carte Arduino [21].(Figure 2.11).

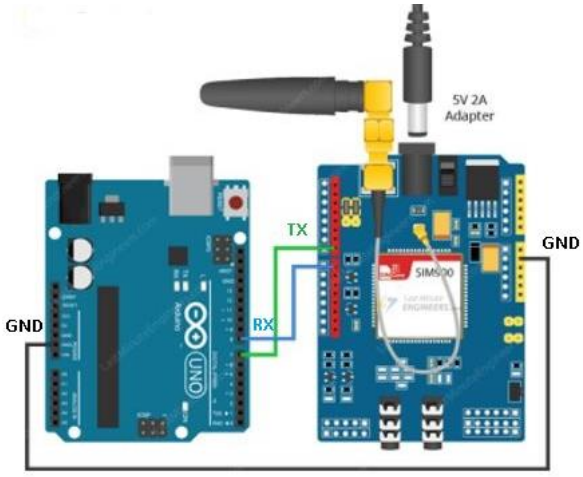

Figure 2.11: Montage Arduino + GSM [21]

a.3) Description de l'écran LED étanche

Les caractéristiques de l'écran LED, représenté sur la Figure 2.12, sont :

- Dimension 100cm (largeur) X 100cm (longueur)
- 120 pixels (lignes) X 120 pixels (colonnes)
- 18 matrices type « GKGD-P8-2727-5S »

- 40 pixels x 20 pixels (800 pixels) par matrice
- Distance entre deux pixels (LED) : 0.8 cm
- 03 matrices (colonne) x 06 matrices (ligne) :18 matrices au total
- 03 alimentations de 05 Volts ,40 A
- Alimentation 220 V
- Connexion réseau RJ45
- Carte électronique « Q1 plus 75 » du fabricant chinois ***listenvision***
- Logiciel LED Mplayer

Figure 2.12: Photo externe de l'écran LED

L'écran est constitué de : 18 matrices de type « GKGD-P8-2727-5S ». Ces matrices sont organisées en 6 lignes et 3 colonnes. Les 3 matrices de chaque ligne sont connectées en série (en cascade) par des nappes dont l'entrée finale communique avec l'un des 8 connecteurs de la carte électronique Q1 plus. Cette dernière reçois les 6 nappes correspond aux 6 lignes des matrices. 3 alimentations de 5V alimentent chaque colonne de matrice. Les caractéristiques de la carte d'alimentation sont :

- **Type** : A-200AF-5,
- **Input** : 200-240 Volts, 2.5 A, 50/60 Hz,
- **Output** : 5 Volts, 40 A.

La carte électronique possède un connecteur RJ45 et elle est alimenter en 5V. 8 connecteurs sont intégrés dans cette carte dont seulement 6 sont utilisés pour le cas de cet écran. La constitution interne de la carte Q1 plus est schématisé sur la Figure 2.13**.** [22].

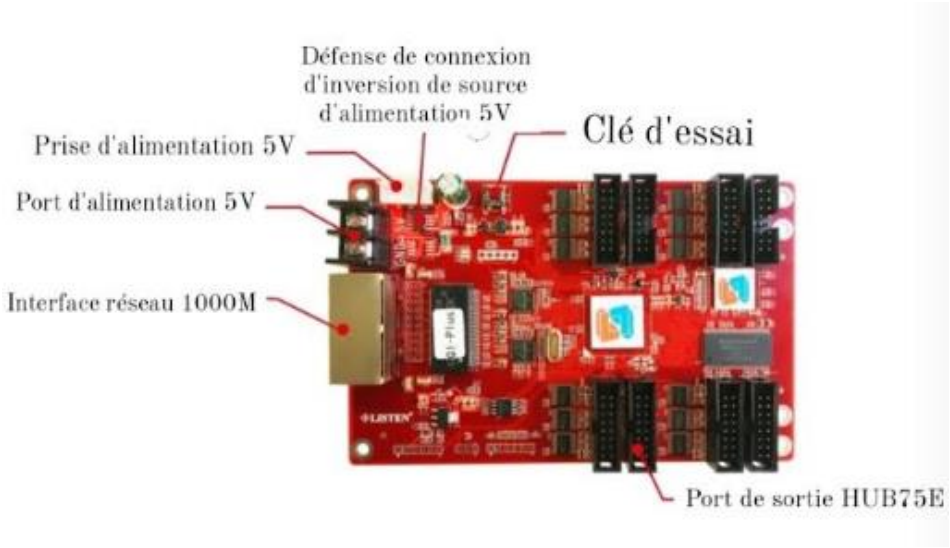

Figure 2.13: Carte électronique Q1 Plus type 75 [22]

Le fabriquant « *Listenvision* » met à la disposition des utilisateurs de la carte le logiciel « *LED Mplayer* » pour la configuration de l'écran. La constitution interne de L'écran est décrite sur la Figure 2.14.

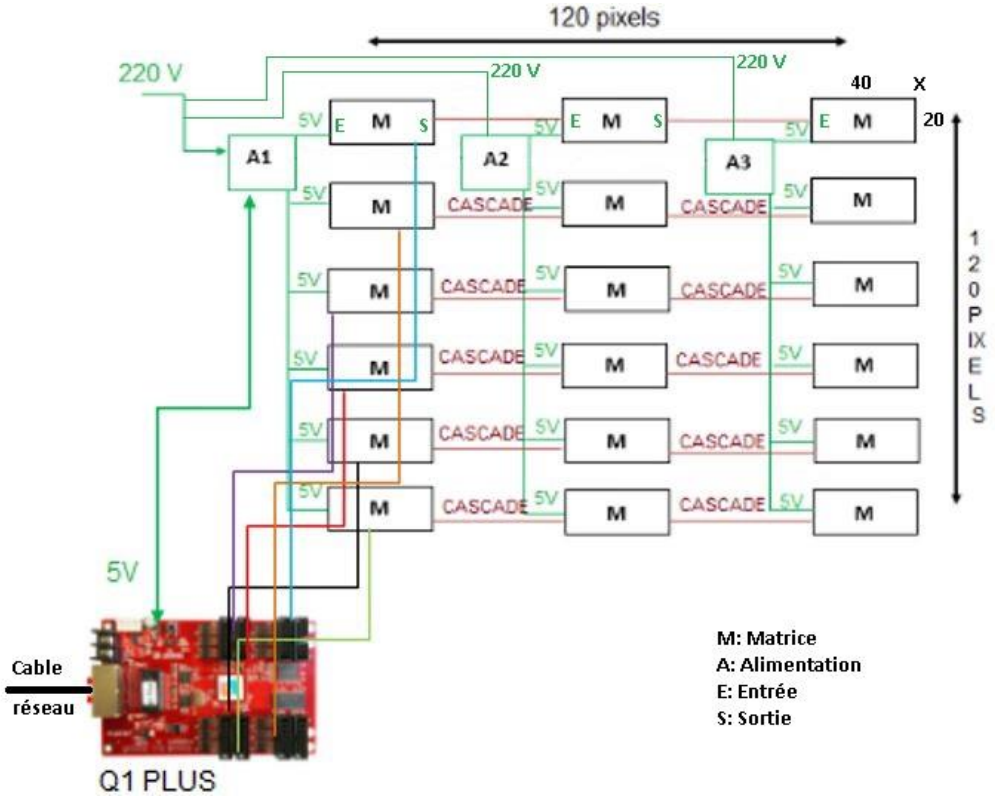

Figure 2.14: Schéma interne de l'écran

L'image interne de l'écran est montrée sur la figure 2.15.

Figure 2.15: Photo interne de l'écran LED

b) Logiciels

b.1) MATLAB

Ce projet a été réalisé à l'aide du logiciel MATLAB (Figure 2.16). C'est un langage de programmation orienté pour le calcul numérique, l'analyse de données et la simulation. Il intègre le calcul, la visualisation et la programmation dans un environnement facile à utiliser où les solutions sont exprimées dans une notation mathématique familière. Nous avons utilisé dans notre cas la version R2020b [23].

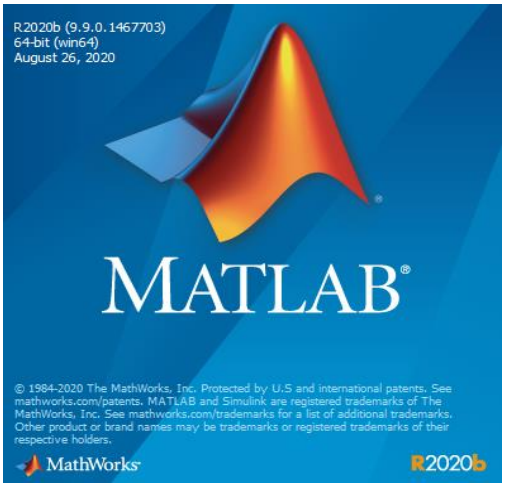

Figure 2.16: Logiciel Matlab 2020 [23]

Les cas d'applications de Matlab sont nombreux. Le langage de programmation est notamment utilisé dans :

- Systèmes de contrôle,
- Machine Learning et Depp Learning,
- Maintenance prédictive,
- Traitement du signal et les séries temporelles,
- Automatisation des tests,
- Systèmes de télécommunication,
- Robotique.

Avantages de Matlab R2020b :

- Fonction « getAvailable » : Donne la liste des « COM » utilisés,

- affichage et sélection des ports « COM »,
- Outil « Application Designer » doté d'une interface graphique améliorée,
- Commande DOS « CMD ».

b.2) LED Mplayer 2.0.5

Les caractéristiques de Mplayer sont :

- ✓ Support simple, double et pleine couleur, support RVB synchrone et asynchrone,
- ✓ Cadre de diffusion 30 images/s, vidéo fluide, animation claire,
- ✓ Attente d'allumage réduit (une seconde),
- ✓ Fonction « zoom » du processeur vidéo, on peut zoomer sur n'importe quelle taille de l'écran du PC et le LED,
- ✓ Excellente qualité d'affichage, fréquence de rafraîchissement élevée (16K Hz), niveau de gris élevé (max. 65536), luminosité élevée,
- ✓ Logiciel convivial, supportant le texte coloré, l'animation en arrière-plan, l'horloge flash et la vidéo,

Description du logiciel :

Le fabricant chinois « *LISTENVISION* » a mis à la disposition des utilisateurs des cartes Q1 plus ce logiciel pour configurer et paramétrer celle-ci en fonction de la résolution en pixels de l'écran LED et du nombre de matrices LED contenues à l'intérieur (voir Figure 2.17).

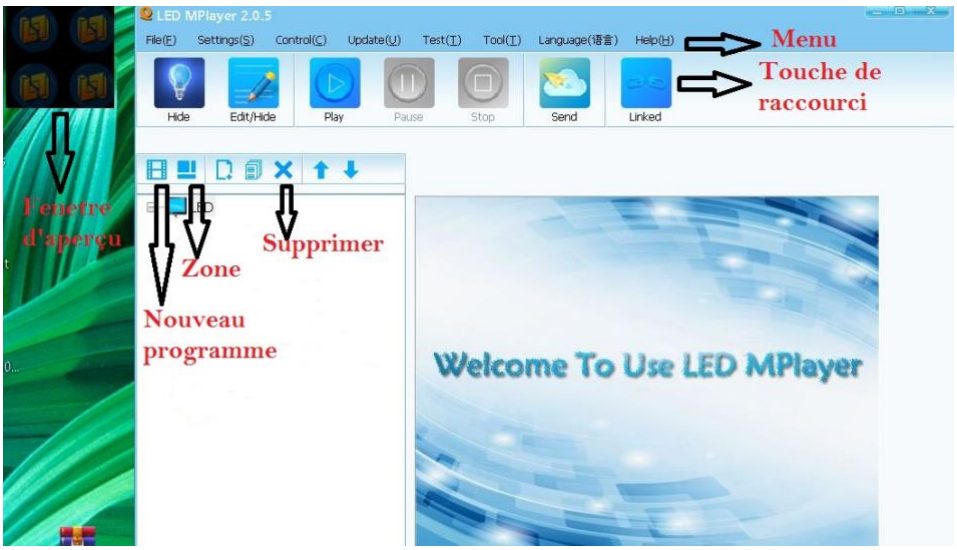

Figure 2.17: Interface du logiciel LED MPlayer [22]

Fonctionnement des modules :

- File(F) Comme le montre la figure 3.18 , cet onglet comprend les fonctions « Nouveau », « Ouvrir », « Enregistrer », « Enregistrer sous » et « Quitter » qui sont visualisées ci-dessous.

Figure 2.18: Onglet « File » [22]

New : Pour créer un nouveau fichier de programme.

Open : Pour ouvrir le fichier de programme enregistré.

Save : Pour enregistrer le fichier du programme actuel.

Save as : Pour enregistrer le fichier du programme actuel à un emplacement particulier avec un nom différent.

Exit : Pour quitter le logiciel LED MPlayer.

- Settings(S) Cet onglet comprend les fonctions de paramètrage des diodes LED, le réglage de la luminosité et les paramètres du logiciel.
- Control(C) Cet onglet comprend les fonctions « Play », « Pause » et « Stop ».
- Update(U) Cet onglet est utilisé pour mettre à jour le firmware de la carte de contrôle.
- Test(T) Cet onglet est utilisé pour le réglage et le test des motifs de l'affichage et des niveaux de gris.
- Tool(T) Onglet pour Utiliser divers outils comme Word, Excel, Bloc-notes, etc.
- Language(语言) Onglet pour modifier la langue par défaut du logiciel.
- Help(H) Onglet d'aide qui donne aussi la version actuelle du logiciel.

Les onglets permettant de lancer ou de tester un programme existant sont (Figure 2.19) :

Play - lancer le programme sur l'écran

Pause - mettre le programme en pause.

Stop – arrêter le programme.

Send – stocker le programme dans la mémoire de la carte de contrôle.

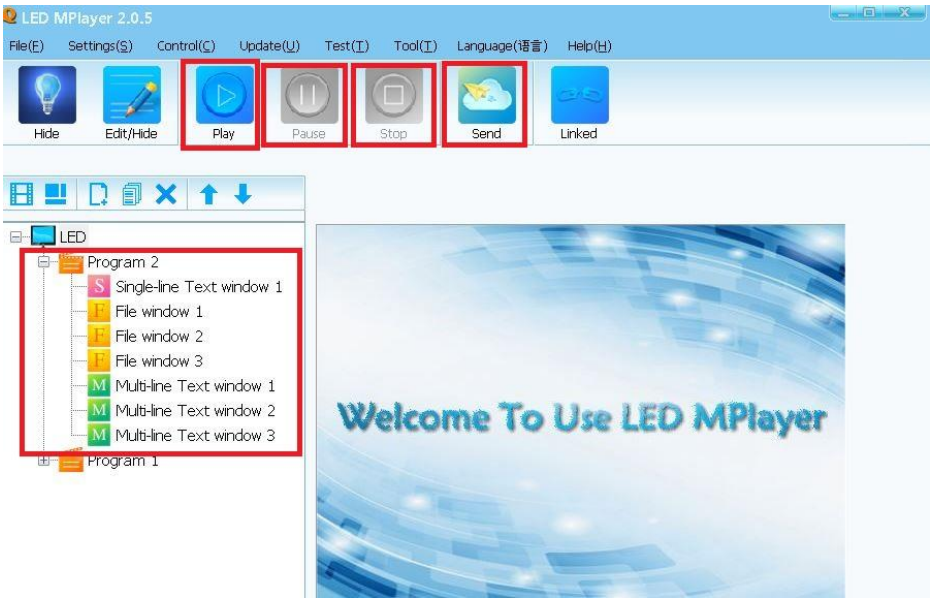

Figure 2.19: Onglets de test d'un programme [22]

2.3 Conclusion

Dans ce chapitre, nous avons présenté notre interface de communication intelligente entre parking. Nous avons détaillé la solution proposée tenant compte de : (i) la vue globale de la solution, (ii) et les règles de communication. Nous avons décrit les composants électroniques et logiciels nécessaires pour la mise en œuvre de notre solution. Le chapitre suivant sera consacré aux résultats obtenus ainsi que leurs interprétations.

Chapitre 3 : Résultats expérimentaux

3.1 Introduction

Dans ce chapitre, nous allons aborder trois volets importants : (i) la conception de l'application d'affichage sur écran LED, (ii) la conception de l'interface graphique de communications et (iii) les résultats obtenus. Nous clôturons ce chapitre par une conclusion.

3.2 Conception de l'application d'affichage sur l'écran LED

L'interface de communication que nous avons implémentée est conçue en utilisant le logiciel LED Mplayer 2.0.5 (Voir la Figure 3.1).

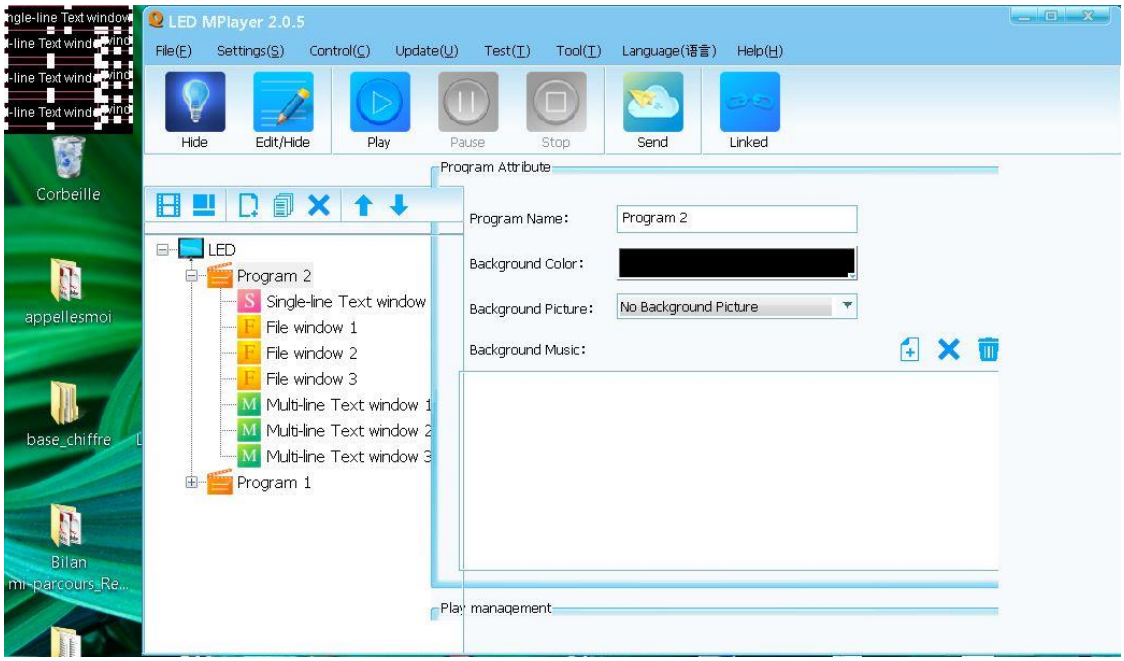

Figure 3.1: Affichage de l'application avec LED Mplayer

Le logiciel Led Mplayer 2.0.5 [19] dispose de 7 importantes icones nommées "hide", "edit/hide", "play"," pause", "stop", "send" et "linked" (Voir la Figure 3.2).

Figure 3.2: Menu de LED Mplayer

Notre application, correspondant au programme 2 (voir la Figure 3.3), est constituée de 7 fonctions importantes :

« Single-line Text Windows 1 », « File window 1 », « File window 2 », « File window 3 », « Multi-Line Text Windows 1 », « Multi -line Text Windows 2 » et « Multi -lie Text Windows 3 ».

Figure 3.3: Contenu du programme 2

Description du programme

- **Single-line Text Windows 1**

Lorsqu'on clique sur la fonction « Single-line Text Windows 1 », un fichier texte nommée « Single-line Text1 » est créé (voir la Figure 3.4) au niveau du chemin « C:\intra parking ». Quand Nous ouvrons ce fichier, le texte « *interconnexion entre parkings* » va apparaitre sur une partie de l'écran du PC et de l'écran LED étanche.

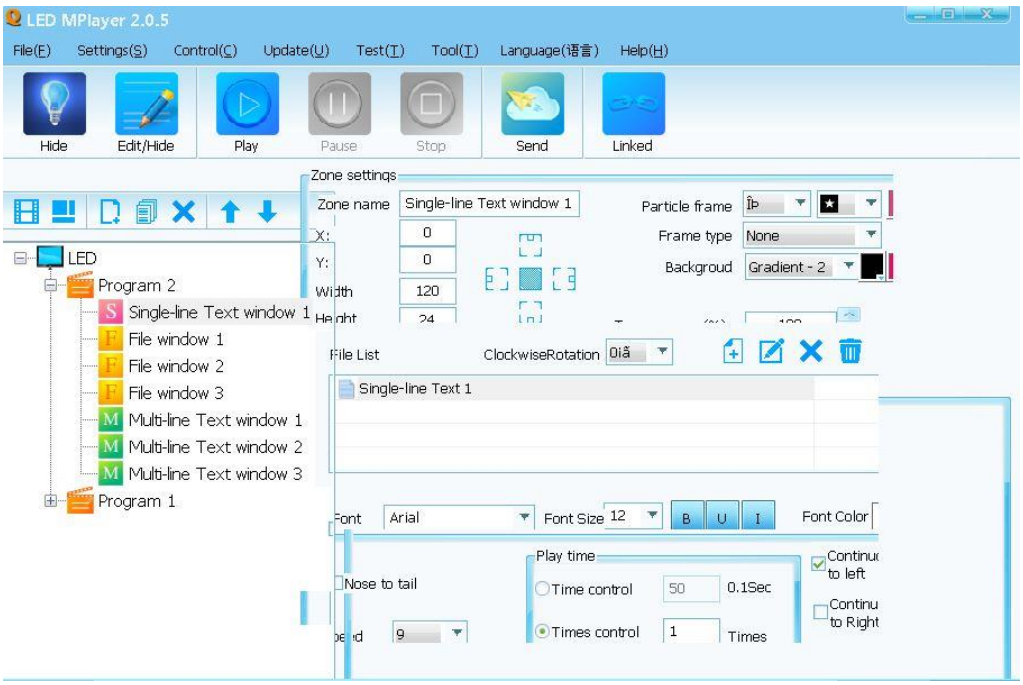

Figure 3.4: Affichage de la fonction « Single-line Text Windows 1 »

Le résultat obtenu s'affichera sur L'écran LED comme suit (Figure 3.5) :

Figure 3.5: Affichage sur l'écran LED de la fonction « Single-line Text Windows 1 »

Il est possible de programmer les paramètres du texte tels que : taille, police, animation, Couleurs, etc …).

- **File window 1**

Lorsqu'on clique sur la fonction « File Windows 1 », un fichier texte nommée « ParkingA.txt » est créé (voir la Figure 3.6) au niveau du chemin « C:\intra parking ». Ce chemin est variable et peut être changé par l'utilisateur à n'importante quel moment. Quand nous ouvrons ce fichier, le chiffre « 3 » indiquant le nombre de places vides dans le parking A va apparaitre sur une partie de l'écran du PC et de l'écran LED étanche.

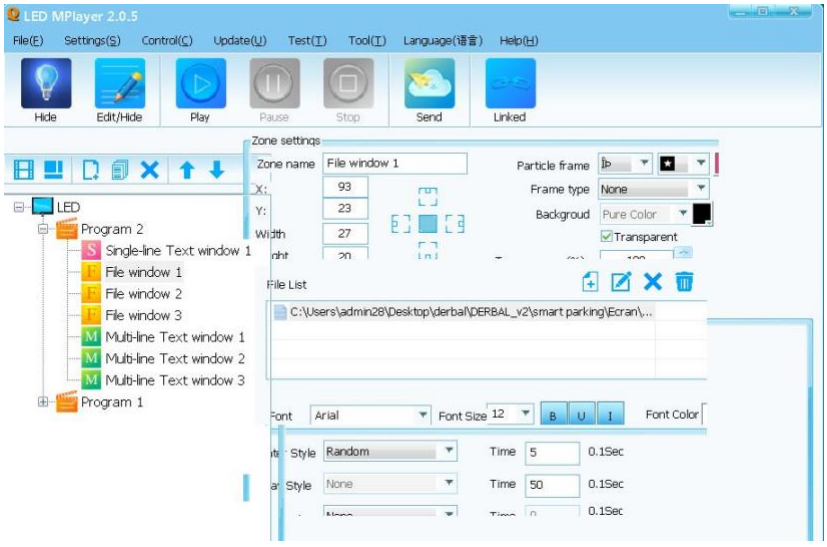

Figure 3.6: Affichage au niveau PC de la fonction « File Windows 1 »

Le résultat obtenu s'affichera sur L'écran LED comme suit (Figure 3.7) :

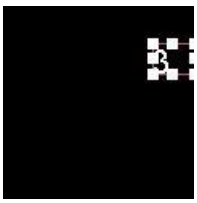

Figure 3.7: Affichage sur l'écran LED de la fonction « File Windows 1 »

- **File window 2**

Lorsqu'on clique sur la fonction « File Windows 2 », un fichier texte nommée « ParkingB.txt » est créé (voir la Figure 3.8) au niveau du chemin « C:\intra parking ». Ce chemin est variable et peut être changé par l'utilisateur à n'importante quel moment. Quand nous ouvrons ce fichier, le chiffre « 33 » indiquant le nombre de places vides dans le parking B va apparaitre sur une partie de l'écran du PC et de l'écran LED étanche.

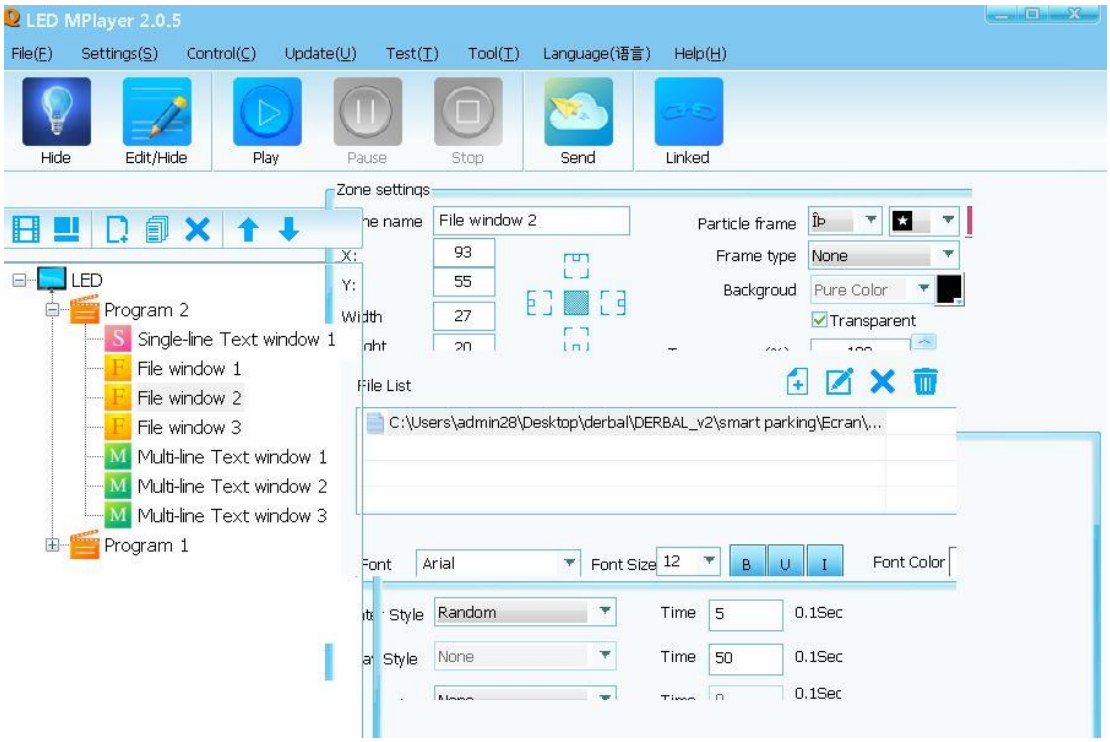

Figure 3.8: Affichage au niveau PC de la fonction « File Windows 2 »

Le résultat obtenu s'affichera sur L'écran LED comme suit (Figure 3.9) :

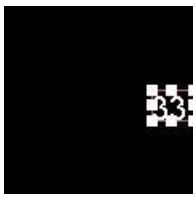

Figure 3.9: Affichage sur l'écran LED de la fonction « File Windows 2 »

- **File window 3**

Lorsqu'on cliquer sur la fonction « File Windows 3 », un fichier texte nommée « ParkingC.txt » est créé (voir la Figure 3.10) au niveau du chemin « C:\intra parking ». Ce chemin est variable et peut être changé par l'utilisateur à n'importante quel moment. Quand nous ouvrons ce fichier, le chiffre « 25 » indiquant le nombre de places vides dans le parking C, va apparaitre sur une partie de l'écran du PC et de l'écran LED étanche.

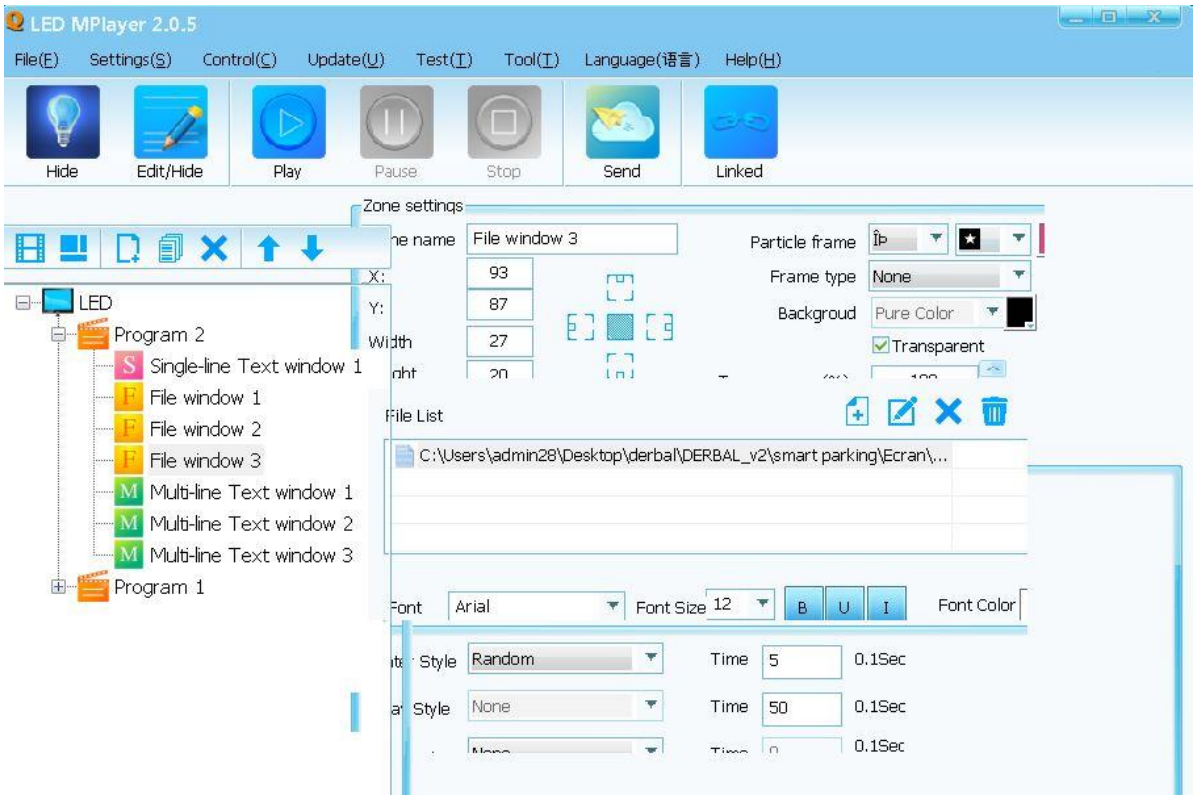

Figure 3.10: Affichage au niveau PC de la fonction « File Windows 3 »

Le résultat obtenu s'affichera sur L'écran LED comme suit (Figure 3.11) :

Figure 3.11: Affichage sur l'écran LED de la fonction « File Windows 3 »

- **Multi-line Text Windows 1**

Lorsqu'on clique sur la fonction « Multi-line Text Windows 1 », un fichier Word nommée « Multi-line Text 1 » est créé au niveau du chemin « C:\Program Files (x86)\LED Mplayer\UserData » (voir la Figure 3.12). Ce chemin est fixe et ne peut être changé par l'utilisateur. Quand nous ouvrons ce fichier, le mot « PARKING A » va apparaitre sur une partie de l'écran du PC et de l'écran LED étanche.

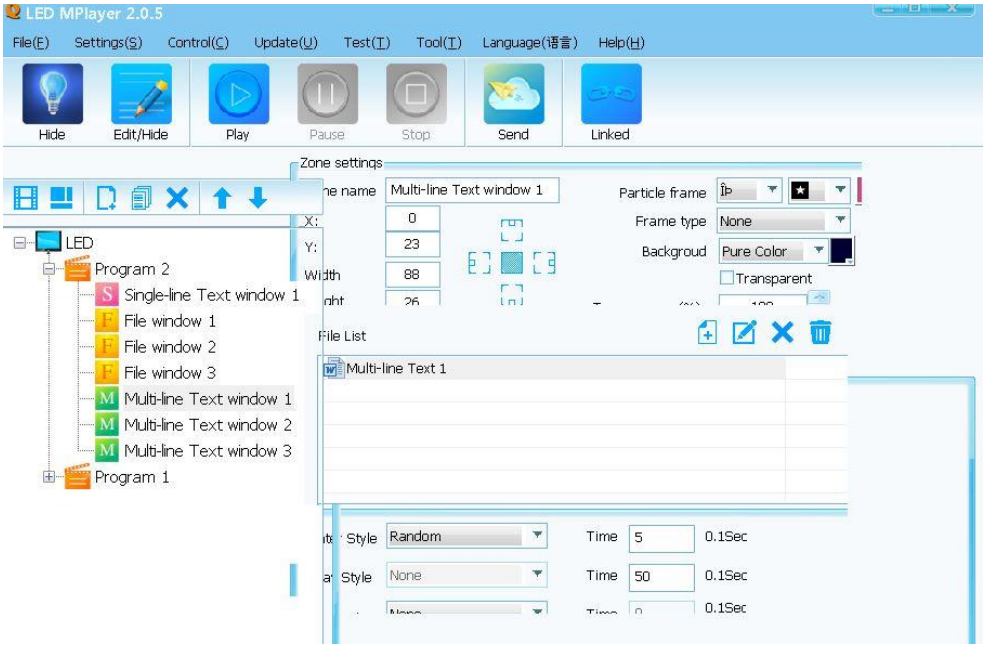

Figure 3.12: Affichage au niveau PC de la fonction « Multi-line Text Windows 1 »

Le résultat obtenu s'affichera sur L'écran LED comme suit (Figure 3.13) :

Figure 3.13: Affichage sur l'écran LED de la fonction « Multi-line Text Windows 1 »

- **Multi -line Text Windows 2**

Lorsqu'on clique sur la fonction « Multi-line Text Windows 2 », un fichier Word nommée « Multi-line Text 2 » est créé (voir la Figure 3.14) au niveau du chemin « C:\Program Files (x86)\LED Mplayer\UserData ». Ce chemin est fixe et ne peut être changé par l'utilisateur. Quand nous ouvrons ce fichier, le mot « PARKING B » va apparaitre sur une partie de l'écran du PC et de l'écran LED étanche.

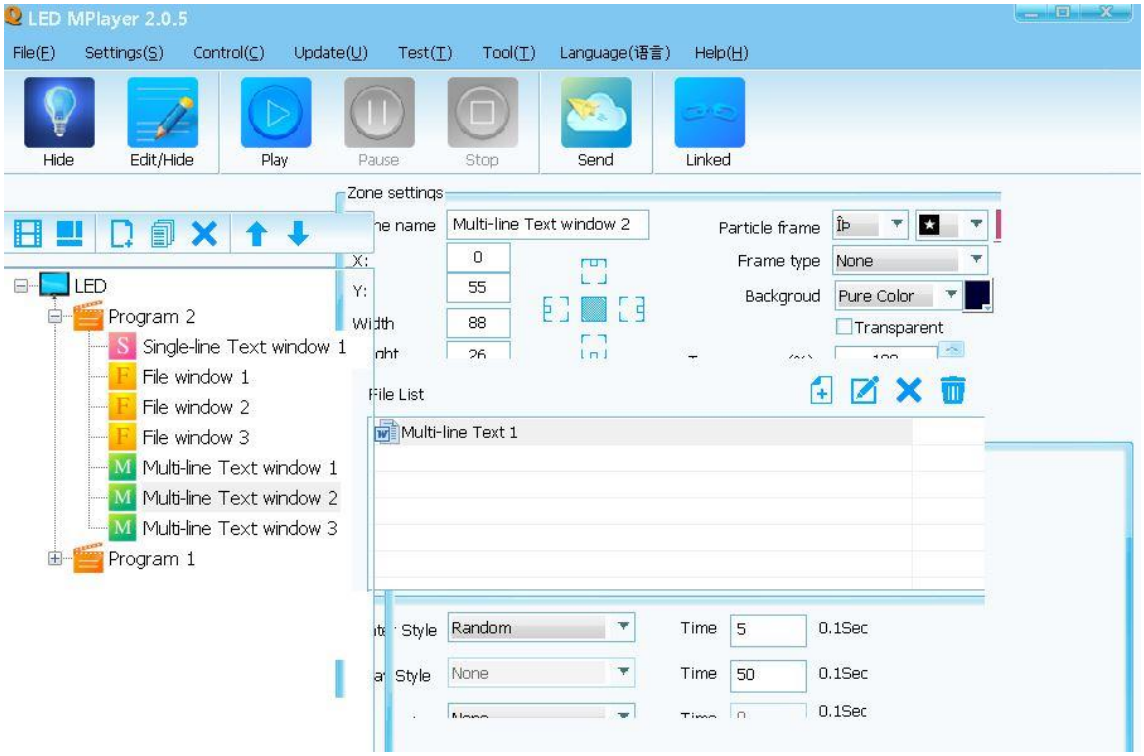

Figure 3.14: Affichage au niveau PC de la fonction « Multi-line Text Windows 2 »

Le résultat obtenu s'affichera sur L'écran LED (Figure 3.15) comme suit :

Figure 3.15: Affichage sur l'écran LED de la fonction « Multi-line Text Windows 2 »

- **Multi -line Text Windows 3**

Lorsqu'on clique sur la fonction « Multi-line Text Windows 2 », un fichier Word nommée

« Multi-line Text 3 » est créé au niveau du chemin « C:\Program Files (x86)\LED Mplayer\UserData » (voir la Figure 3.16). Ce chemin est fixe et ne peut être changé par l'utilisateur. Quand nous ouvrons ce fichier, le mot « PARKING C » va apparaitre sur une partie de l'écran du PC et de l'écran LED étanche.

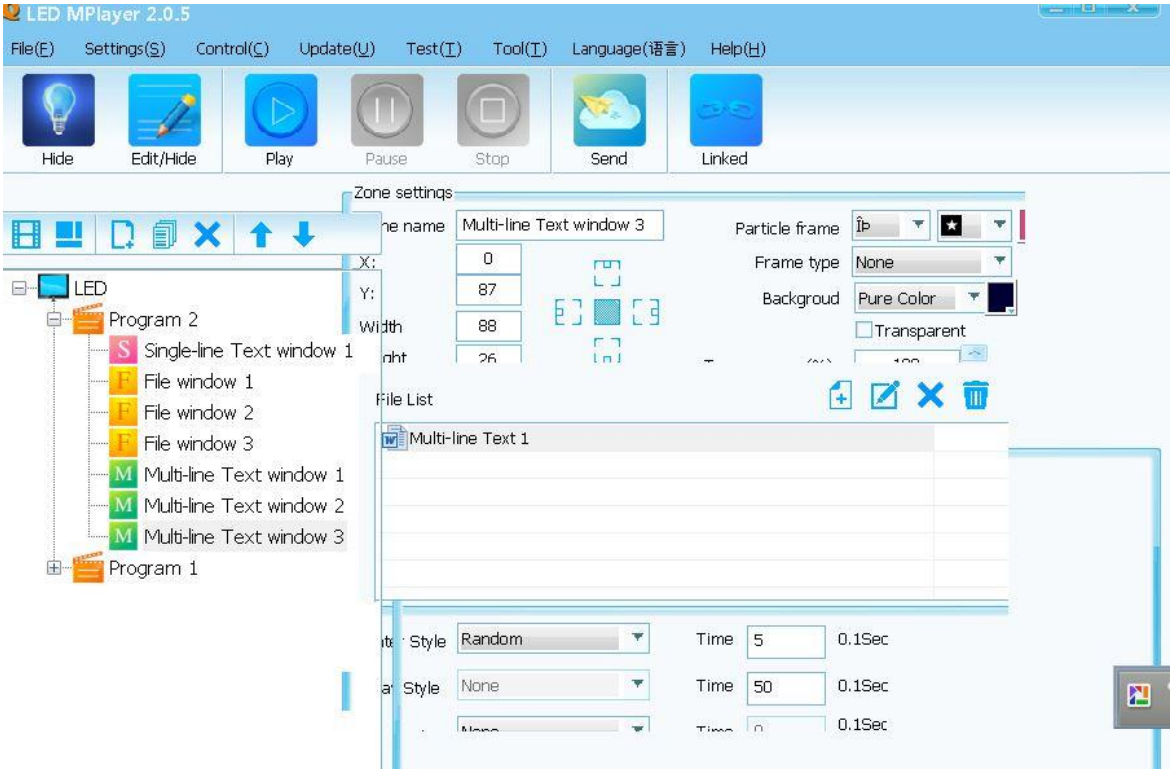

Figure 3.16: Affichage au niveau PC de la fonction « Multi-line Text Windows 3 »

Le résultat obtenu s'affichera sur L'écran LED comme suit (Figure 3.17) :

Figure 3.17: Affichage sur l'écran LED de la fonction « Multi-line Text Windows 3 »

La conception du « programme 2 » étant finalisée, il est possible de cliquer sur l'icône « Play » pour faire apparaitre le résultat de l'affichage du programme 2 sur une partie de l'écran du PC (voir la Figure 3.18).

Figure 3.18: Résultat final de l'Affichage au niveau PC du programme 2

Figure 3.19: Résultat final de l'Affichage au niveau de l'écran LED du programme 2

Il est possible de configurer l'affichage du texte en choisissant les paramètres suivants :
« None », « Random », « Move to left », « Move to right », etc…. Ces paramètres jouent un rôle important dans la façon d'afficher et de faire défiler le texte par des animations établies par le fabricant du produit sur l'écran LED (voir la Figure 3.20).

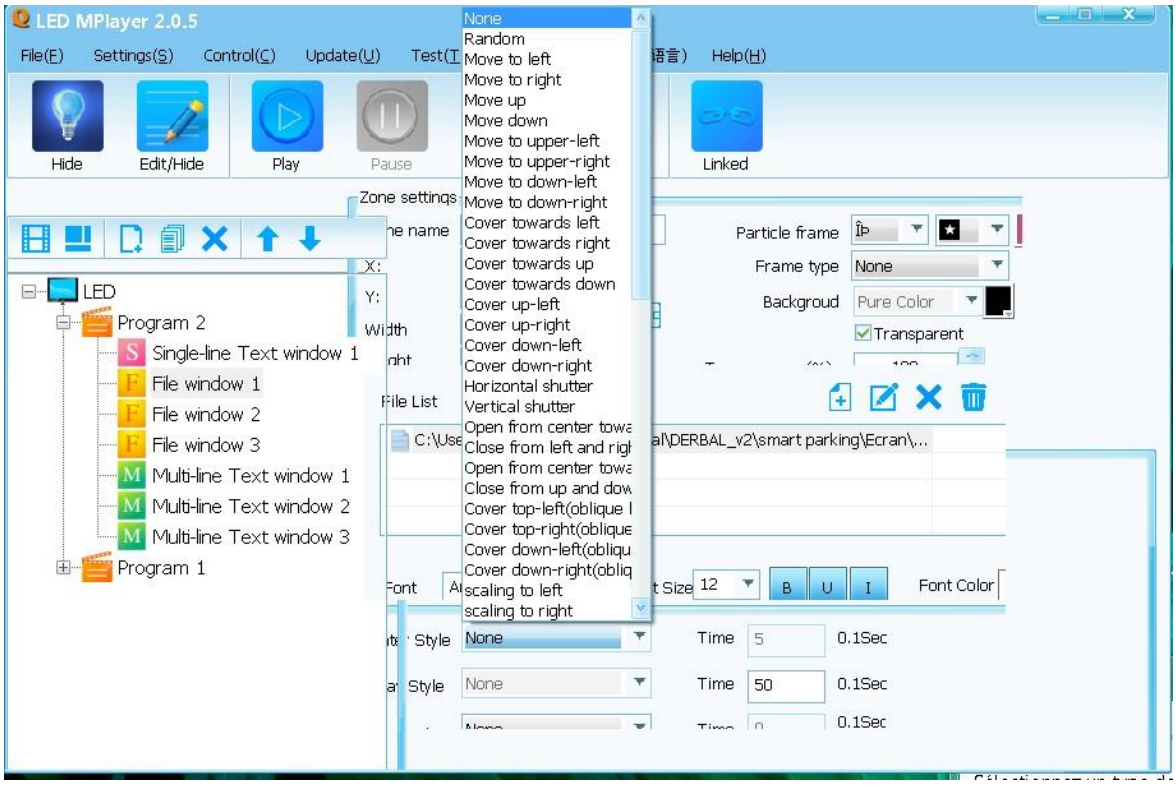

Figure 3.20: Paramétrage de l'affichage du texte du « programme 2 »

3.3 Conception de l'interface graphique de communication entre parkings

Dans cette section, nous discuterons du programme Matlab et de l'interface du projet :

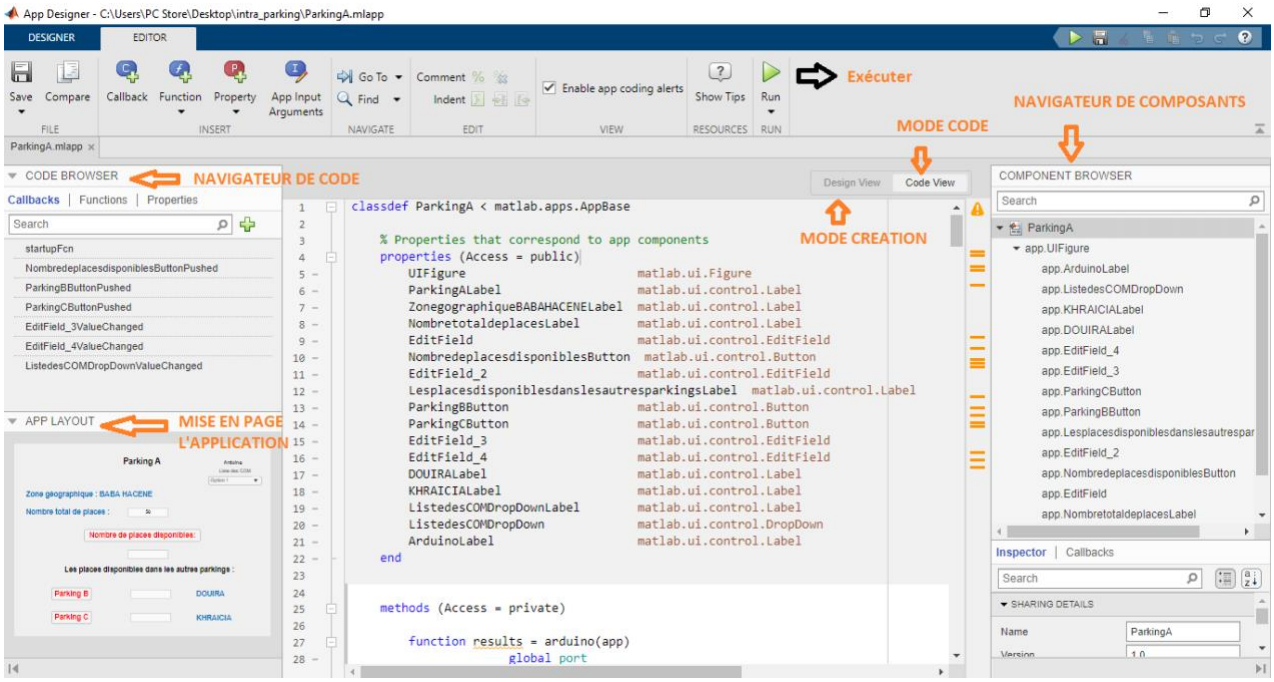

Figure 3.21: Interface de Matlab 2020 du programme parking A

Lorsqu'on a clique sur la fonction Callbacks correspondant au programme A on obtient les fonctions suivantes (Figure 3.22) :

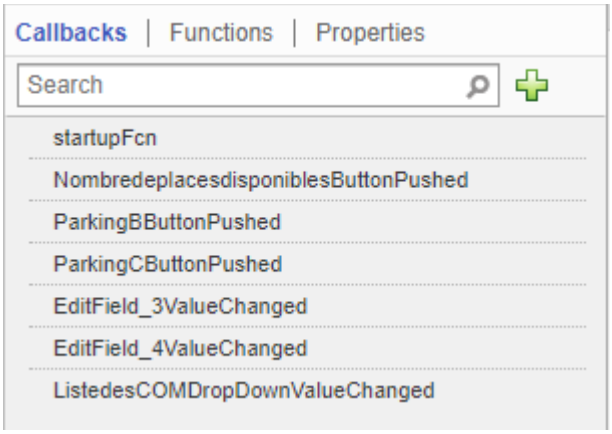

Figure 3.22: Fonctions du parking A

On agit sur l'option « code view » pour visualiser le code Matlab programme « parking A » (Figure 3.23) :

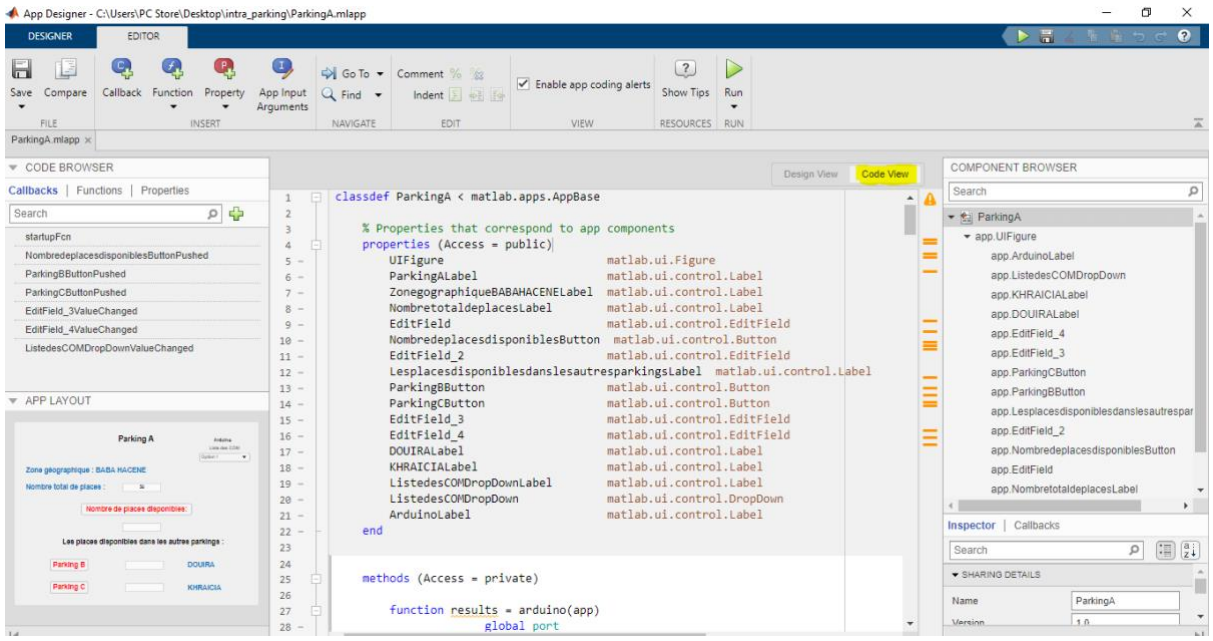

Figure 3.23: Interface graphique du programme avec option « Code view »

On peut utiliser l'option « design view » pour modifier l'interface graphique :

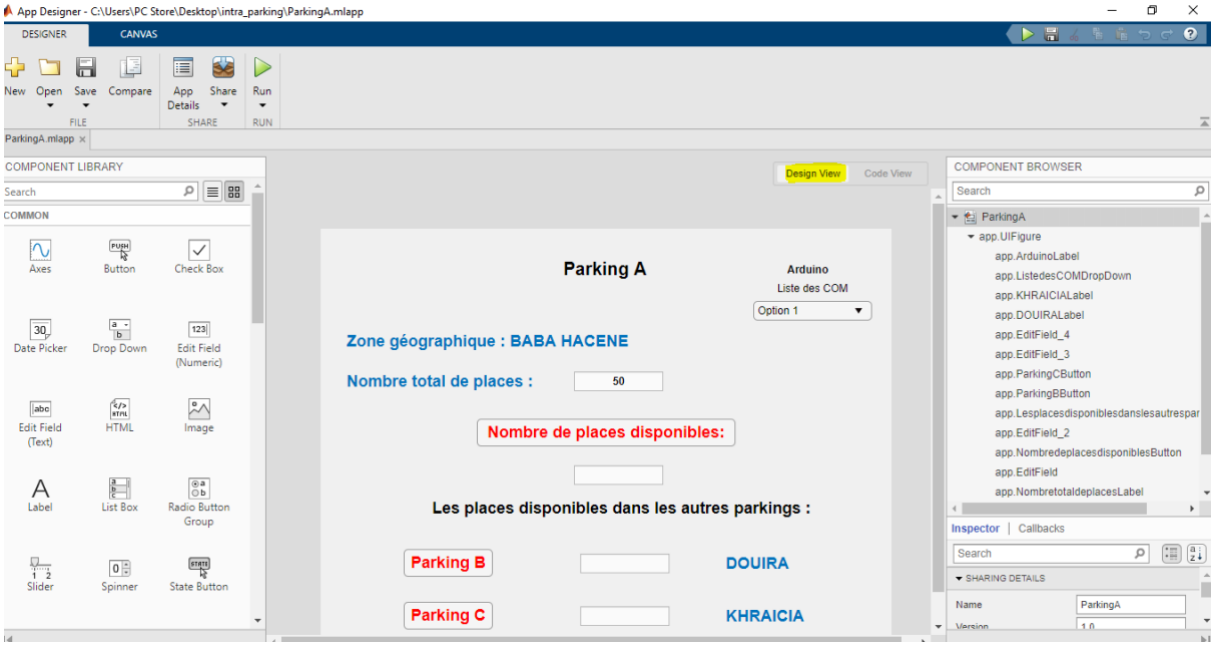

Figure 3.24: interface graphique du programme avec option "Design view"

Nous disposons d'une interface graphique pour le parking A. Cette interface visualise le nom, la zone géographique et le nombre total de places dans ce parking (500 pour cet exemple). Il dispose également de trois push Button nommés « nombre de places disponible », « parking B », « parking C » :

3.3.1 Parking A

- Code Matlab de « nombre de places disponible » :
  functionNombredeplacesdisponiblesButtonPushed(app, event)
- Code MATLAB de « Parking B »:
  functionParkingBButtonPushed(app, event)
- Code Matlab de « Parking C » :
  functionParkingCButtonPushed(app, event)

L'instruction de l'information correspondant à la liste des COM Arduino est :

value = app.ListedesCOMDropDown.Value;

En cliquant sur le bouton « parking B », le parking A interroge les parkings B et C pour savoir le nombre de leurs places vides. Le message envoyé aux parkings B et C est :

fileID = fopen('places.txt','r');

Ce message est envoyé via la carte Arduino et le réseau GSM. Le parking B répond par le code Matlab suivant :

if (length(idn)>3) & (idn(1:2)=='PB'),

et l'affiche dans la zone d'affichage nommée Edit Field 3 :

value = app.EditField_3.Value;

de l'interface graphique. Il en va de même pour le parking C.

Figure 3.25: Interface graphique du parking A

Dans cette interface, le parking A interrogé les parkings B et C. Ces derniers sont uniquement en mode d'attente et ne communiquent le nombre de leurs places vides que s'ils sont sollicités par le parking A.

Si on clique sur le bouton « Nombre de places disponibles », Matlab va lire le nombre qui est écrit dans le fichier de Mplayer du parking A suivant : « ParkingA.txt » au niveau du chemin « C:\intra parking » et l'affichera dans sa zone correspondante. Si ce nombre est supérieur à 3, les parkings B et C seront sollicités manuellement par l'utilisateur. Si ce nombre est inférieur ou égale à 3, le parking A envoie automatiquement un message qui contient le mot « place » pour les parkings B et C. La réponse du parking B au parking A est « PB33 » comme exemple. Le mot PB33 est pris en compte par le code Matlab et le chiffre 33 sera affiché dans la case du parking B.

La réponse du parking C au parking A est « PC25 » comme exemple.

Le mot PC25 est pris en compte par le code Matlab et le chiffre 25 sera affiché dans la case du parking C. L'envoie et la réception des messages SMS par le logiciel Matlab est conditionnée par l'utilisation des packages suivant situés dans le même dossier :

Figure 3.26: Librairies et codes nécessaires aux SMS

Nous pouvons visualiser cette opération grâce à l'instruction « idn ». Les figures ci-dessous illustrent le détail de ces opérations durant la phase « la communication entres les interfaces » :

```
idn =

    'Initializing...
     '


idn =

    '                                                    AT
     '


idn =

    '
     '


idn =

    'OK
     '


idn =
```

```
idn =

     'AT+CMGF=1
      '


idn =

     '
      '


idn =

     'OK
      '


idn =

     'AT+CNMI=1,2,0,0,0
      '


idn =
```

```
idn =

     'OK
      '


idn =

     'AT+CMGS="0555375888"
      '


idn =

     '
      '


idn =

     '>
      '


idn =
```

```
    '>
     '


idn =

    'place
     '


idn =

    '+CMGS: 1
     '


idn =

    '
     '


idn =

    'OK
     '
```

Figure 3.27: Phase de communication entre les parkings durant l'exécution du code

3.3.2 Parking B

Nous disposons d'une interface graphique pour le parking B. Cette interface visualise le nombre de places disponibles dans ce parking et la liste des COM pour connaitre l'état de la carte Arduino (connexion ou déconnexion). Au niveau de l'icône « Liste des COM » correspondant au code « puchboutton (ARDUINO/ Déconnexion) », un bouton Led s'allume en vert lorsqu'une carte Arduino est connectée et change de couleur en rouge quand celle-ci est déconnectée. L'instruction de l'information correspondant à liste des COM Arduino est :

value = app.ListedesCOMDropDown.Value;

Le nombre disponible des places dans le parking B est affiché dans la case des résultats correspondant.

Après la réception d'un message SMS (« place » dans notre cas) provenant du parking A, Le parking B répond par le code Matlab suivant :

if (length(idn)>3) & (idn(1:2)=='PB'),

Et l'affiche dans la zone d'affichage nommée Edit Field 3 :

value = app.EditField_3.Value;

Figure 3.28: Interface graphique du parking B

4.3.3 Parking C

Dans notre exemple on a un agent de parking C (parking anarchique), nous utilisons qu'un téléphone portable et une puces SIM, qu'il communique avec les parkings A et B.

Figure 3.29: Le parking C

3.3.3 Conception du programme en mode Desing View

Le tableau ci-dessous (3.1) résume les différents composants du mode « Desing View «

Utilisé dans notre application :

Tableau 3-1 : Caractéristiques de l'interface graphique

Fonctions et textes utilisé dans le programme	Composants (Design View)
Arduino/Liste des COM	List Box
Nombre de places disponibles	Push Button
Parking A	Text
Zone géographique	Text
Nombre total de places	Text
Les places disponibles dans les autres parkings	Text
Parking B	Push Button
Parking C	Push Button
Case correspondant au résultat obtenu de « nombre total de places » du parking A	Edit Field
Case correspondant au résultat obtenu de « Nombre de places disponibles » du Parking A	Edit Field
Case correspondant au résultat obtenu de Parking B	Edit Field
Case correspondant au résultat obtenu de Parking C	Edit Field

4.3.4 Description des phases de fonctionnement des trois parkings

La figure 4.30 illustre de façon simple les différentes opérations de communications entre les trois parkings A, B et C. Les parkings B et C sont en phase d'attente et ne communiquent leurs données que s'ils sont sollicités par le parking A. Ce dernier est en mode maitre par rapport au parking B et C.

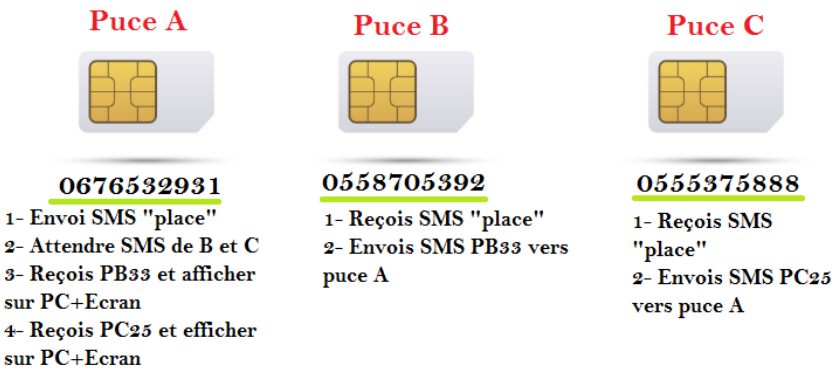

Figure 3.30: Etapes de fonctionnement des programmes de communications entre parking

3.4 Résultats des tests

Dans notre projet nous avons utilisé deux cartes Arduino et deux cartes interfaces GSM SIM900. L'ordinateur PC1 du parking A est connecté à une carte Arduino 1 munie de la carte GSM1 contenant une puce Mobilis.

L'ordinateur PC2 du parking B est connecté à la carte Arduino 2 munie de la carte SIM2 contenant une puce Ooredoo.

Le parking C ne dispose pas d'un ordinateur mais utilise un téléphone portable (simple ou smart) muni d'une autre puce Ooredooo.

L'ordinateur correspondant au parking A (mode maitre) est connecté par un câble RJ45 à un écran affichage étanche à LED de dimension 1 mètre x 1 mètre via un câble réseau de 20 mètres. Un exemple d'affichage des places disponibles dans les parkings A, B et C est illustré sur la Figure 3.31.

Figure 3.31: Affichage des résultats sur écran LED étanche

3.5 Conclusion

La plateforme de communication intelligente entre parking a été réalisée pour trois parkings A, B et C. L'architecture de communication des données choisie est le réseau GSM. Des cartes Arduino et des interfaces GSM/SIM900 ont été utilisées comme support de transmission de ces données provenant des différents parkings. On peut étendre le nombre de parking selon le besoin des utilisateurs. Cette plateforme fonctionne en mode manuel et automatique. Un écran étanche LED de grande dimension est utilisé pour l'affichage des places vides dans les trois parkings pour l'information des conducteurs. Un système par règle a été conçu pour assister et orienter les conducteurs selon le contenu des places vides disponibles dans les parkings. Le logiciel MATLAB R2020b a été utilisé dans ce projet pour ses avantages (fonction « getAvailable » qui donne la liste des « COM » utilisés, affichage et sélection des ports « COM », Outil « Application Designer » doté d'une interface graphique améliorée).

Conclusion générale

Le travail réalisé nous a permis d'avoir un aperçu sur le problème de communication intelligent entre parking. Les différentes approches et solutions utilisés dans notre application ont été décrites. Une vue d'ensemble sur les notions de base des systèmes experts comme outils intelligent a été abordée dans cette étude. L'intégration de cet outil dans notre application n'a pas été finalisée. La solution présentée utilise deux ordinateurs, un téléphone mobile, deux cartes Arduino Uno, deux interfaces (Shield) GSM/SIM900 et un écran LED étanche de grande dimension.

Annexe

1. Programme des cartes « Arduino » (Parking A, B et C)

• Programme de la carte « Arduino » connectée au parking A

```
#include <SoftwareSerial.h>
//Créer un objet série logiciel pour communiquer avec  SIM900
SoftwareSerial mySerial(7, 8); //SIM900 Tx & Rx est connecté a Arduino #7 & #8
char var ;
void setup()
{
 //Commencer la communication série avec Arduino et Arduino IDE (Serial Monitor)
 Serial.begin(9600);
 // Commencer la communication série avec Arduino et SIM900
 mySerial.begin(9600);
 Serial.println("Initializing..."); // affiche un retour de chariot et un saut de ligne
 delay(1000); // Suspend le programme pendant une durée de 1000 ms
 mySerial.println("AT"); //Prise de contact avec SIM900
 updateSerial();  // envoyer la commande AT via mySerial
 mySerial.println("AT+CMGF=1"); // Configuration du mode text
 updateSerial();  // envoyer la commande AT+CMGF=1 via mySerial
 mySerial.println("AT+CNMI=1,2,0,0,0"); // Décide comment les messages SMS
nouvellement arrivés doivent être manipulé
 updateSerial(); // envoyer la commande AT+CNMI=1,2,0,0,0 via mySerial
}
void loop () {
 delay(500); // Suspend le programme pendant une durée de 500 ms
while (Serial.available()) // function obtient les octets stockés du port série qui sont
disponible pour la lecture
{
   var = Serial.read(); // lire la commande
```

```
   If (var=='B') {
        mySerial.println("AT+CMGS=\"0555375888\""); //le numéro de la puce qui
est dans la carte SIM900 de PARKING B
        updateSerial();
        mySerial.print("place "); //contenu du text
        updateSerial();
        mySerial.write(26);  // envoyer un octet avec la valeur 26
     }
     if (var=='C') {
        mySerial.println("AT+CMGS=\"0558705392\""); // le numéro de la puce qui est
 dans la carte SIM900 de PARKING C
        updateSerial();
        mySerial.print("place “); //contenu du text
        updateSerial();
        mySerial.write(26);
     }
}
  updateSerial();
}
void updateSerial()
{ delay(500); // Suspend le programme pendant une durée de 500 ms
 while (Serial.available())
 {
  mySerial.write(Serial.read()); //Transférer ce que serial a réçu au port série du logiciel }
 while(mySerial.available())
 {
  Serial.write(mySerial.read()); // Transférer ce que logiciel serial a réçu au port série }
}
```

- **Programme de la carte « Arduino » connectée au Parking B**

```
#include <SoftwareSerial.h>
//Créer un objet série logiciel pour communiquer avec SIM900
SoftwareSerial mySerial(7, 8); //SIM900 Tx & Rx est connecté a Arduino #7 & #8
void setup()
{ //Commencer la communication série avec Arduino et Arduino IDE (Serial Monitor)
 Serial.begin(9600);
 // Commencer la communication série avec Arduino et SIM900
 mySerial.begin(9600);
 Serial.println("Initializing...");
 delay(1000);
 mySerial.println("AT"); //Prise de contact avec SIM900
 updateSerial();
 mySerial.println("AT+CMGF=1"); // Configuration du mode text
 updateSerial();
 mySerial.println("AT+CNMI=1,2,0,0,0"); // Décide comment les messages SMS nouvellement arrivés
doivent être manipulé
 updateSerial();
}
void loop()
{
 delay(500);
 while (Serial.available())
 {
  mySerial.println("AT+CMGS=\"0549169027\""); //le numéro de la puce qui est dans la carte
SIM900 de PARKING A
  updateSerial();
  mySerial.write(Serial.read()); //Transférer ce que logiciel serial a réçu au port série
  mySerial.print(Serial.read()); //contenu du text
  updateSerial();
```

```
    mySerial.write(26);
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read()); //Transférer ce que logiciel serial a reçu au port série
  }
}
void updateSerial()
{
  delay(500);
  while (Serial.available())
  {
    mySerial.write(Serial.read()); //Transférer ce que logiciel serial a reçu au port série
  }
  while(mySerial.available())
  {
    Serial.write(mySerial.read());
  }
}
```

Programme MATLAB de gestion du parking A

```
classdef ParkingA < matlab.apps.AppBase
 % Properties that correspond to app components
 properties (Access = public)
    UIFigure                matlab.ui.Figure
    ParkingALabel           matlab.ui.control.Label
    ZonegographiqueBABAHACENELabel  matlab.ui.control.Label
    NombretotaldeplacesLabel     matlab.ui.control.Label
    EditField               matlab.ui.control.EditField
    NombredeplacesdisponiblesButton  matlab.ui.control.Button
    EditField_2             matlab.ui.control.EditField
    LesplacesdisponiblesdanslesautresparkingsLabelmatlab.ui.control.Label
    ParkingBButton          matlab.ui.control.Button
    ParkingCButton          matlab.ui.control.Button
    EditField_3             matlab.ui.control.EditField
    EditField_4             matlab.ui.control.EditField
    DOUIRALabel             matlab.ui.control.Label
    KHRAICIALabel           matlab.ui.control.Label
    ListedesCOMDropDownLabel matlab.ui.control.Label
    ListedesCOMDropDown     matlab.ui.control.DropDown
    ArduinoLabel            matlab.ui.control.Label
    end
    methods (Access = private)
        function results = arduino(app)
            global port
 instrreset
     com = app.ListedesCOMDropDown.Value;
     port=serial(com,'BaudRate',9600);
     app.Lamp.Color=[0,1,0];
      fopen(port);
```

```
  pause(3);
  while (1)
  idn = fscanf(port);
  if ~isempty(idn)
     idn
if (length(idn)==7) & (idn(1:5)=='place')
                save('var.mat','idn');
                p=app.EditField.Value;
                fprintf(port,p);
                pause(2);
             end
end
          pause(2);
          end
          fclose(port);
       end
    end
% Callbacks that handle component events
    methods (Access = private)
       % Code that executes after component creation
       function startupFcn(app)
list = getAvailableComPort();   % for 2015
          app.ListedesCOMDropDown.Items=list;
       end
% Button pushed function: NombredeplacesdisponiblesButton
       function NombredeplacesdisponiblesButtonPushed(app, event)
fileID = fopen('places.txt','r');
          formatSpec = '%f';
  p = fscanf(fileID,formatSpec);
 app.EditField_2.Value=num2str(p);
         fileID = fopen('C:\Users\admin28\Desktop\derbal\DERBAL_v2\smart
parking\Ecran\ParkingA\places_libres.txt','w');nbytes = fprintf(fileID,'%d',str2num(app.EditField_2.Value))
```

```
if p <4
        ParkingBButtonPushed(app, event)
        ParkingCButtonPushed(app, event)
         end
       end
% Button pushed function: ParkingBButton
      function ParkingBButtonPushed(app, event)
         global port
        instrreset
com = app.ListedesCOMDropDown.Value;
        port=serial(com,'BaudRate',9600);
         fopen(port);
         pause(3);
for i=1:10
         idn = fscanf(port)
          end
         pause(1);
          fprintf(port,'B');
          pause(2);
          fprintf(port,'B');
         while (1)
           try
           fopen(port);
            end
         idn = fscanf(port);
         if ~isempty(idn)
            idn
            if (length(idn)>3) &(idn(1:2)=='PB')
              save('var.mat','idn');
              app.EditField_3.Value=idn(3:end);
fileID0 = fopen('C:\Users\admin28\Desktop\derbal\DERBAL_v2\smart parking\Ecran\ParkingB\places_libres.txt','w');nbytes
fprintf(fileID0,'%d',str2num(app.EditField_3.Value));
```

```
                pause(2);
                break
            end
        end
        pause(2);
        end
        fclose(port);
    end
% Button pushed function: ParkingCButton
    function ParkingCButtonPushed(app, event)
        global port
      instrreset
      com = app.ListedesCOMDropDown.Value;
      port=serial(com,'BaudRate',9600);
       fopen(port);
       pause(3);
        for i=1:10
       idn = fscanf(port);
        end
        pause(2);
        fprintf(port,'C');
        pause(2);
        fprintf(port,'C');
       while (1)
          try
         fopen(port);
          end
       idn = fscanf(port);
       if ~isempty(idn)
          idn
if (length(idn)>3) & (idn(1:2)=='PC')
```

```
            save('var.mat','idn');
             app.EditField_4.Value=idn(3:end);
fileID10 = fopen('C:\Users\admin28\Desktop\derbal\DERBAL_v2\smart parking\Ecran\ParkingC\places_libres.txt','w');nbyte
fprintf(fileID10,'%d',str2num(app.EditField_4.Value));
              pause(2);
              break
           end
        end
        pause(2);
        end
        fclose(port);
end
    % Value changed function: EditField_3
    function EditField_3ValueChanged(app, event)
        value = app.EditField_3.Value;
    end
% Value changed function: EditField_4
function EditField_4ValueChanged(app, event)
        value = app.EditField_4.Value;
    end
% Value changed function: EditField_2
    function EditField_2ValueChanged(app, event)
        value = app.EditField_2.Value;
    end
% Value changed function: ListedesCOMDropDown
function ListedesCOMDropDownValueChanged(app, event)
     value = app.ListedesCOMDropDown.Value;
    end
% Drop down opening function: ListedesCOMDropDown
    function ListedesCOMDropDownOpening(app, event)
    end
    % Button down function: UIFigure
```

```
        function UIFigureButtonDown(app, event)
        end
    % Value changing function: EditField_3
        function EditField_3ValueChanging(app, event)
            changingValue = event.Value;
        end
    end
    % Component initialization
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
    % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'MATLAB App';
        app.UIFigure.ButtonDownFcn = createCallbackFcn(app, @UIFigureButtonDown, true);
    % Create ParkingALabel
            app.ParkingALabel = uilabel(app.UIFigure);
            app.ParkingALabel.FontSize = 20;
    app.ParkingALabel.FontWeight = 'bold';
            app.ParkingALabel.Position = [273 422 96 27];
            app.ParkingALabel.Text = 'Parking A';
    % Create ZonegographiqueBABAHACENELabel
            app.ZonegographiqueBABAHACENELabel = uilabel(app.UIFigure);
            app.ZonegographiqueBABAHACENELabel.FontSize = 18;
            app.ZonegographiqueBABAHACENELabel.FontWeight = 'bold';
            app.ZonegographiqueBABAHACENELabel.FontColor = [0 0.4471 0.7412];
            app.ZonegographiqueBABAHACENELabel.Position = [30 343 328 24];
        app.ZonegographiqueBABAHACENELabel.Text = 'Zone géographique:BABA HACENE';
    % Create NombretotaldeplacesLabel
            app.NombretotaldeplacesLabel = uilabel(app.UIFigure);
```

```
            app.NombretotaldeplacesLabel.FontSize = 18;
            app.NombretotaldeplacesLabel.FontWeight = 'bold';
            app.NombretotaldeplacesLabel.FontColor = [0 0.4471 0.7412];
            app.NombretotaldeplacesLabel.Position = [30 298 220 24];
            app.NombretotaldeplacesLabel.Text = 'Nombre total de places :';
        % Create EditField
            app.EditField = uieditfield(app.UIFigure, 'text');
            app.EditField.Editable = 'off';
            app.EditField.HorizontalAlignment = 'center';
            app.EditField.FontWeight = 'bold';
            app.EditField.Position = [285 299 100 22];
            app.EditField.Value = '50';
        % Create NombredeplacesdisponiblesButton
            app.NombredeplacesdisponiblesButton = uibutton(app.UIFigure, 'push');
        app.NombredeplacesdisponiblesButton.ButtonPushedFcn = createCallbackFcn(app,
@NombredeplacesdisponiblesButtonPushed, true);
            app.NombredeplacesdisponiblesButton.FontSize = 18;
            app.NombredeplacesdisponiblesButton.FontWeight = 'bold';
            app.NombredeplacesdisponiblesButton.FontColor = [1 0 0];
            app.NombredeplacesdisponiblesButton.Position = [176 237 290 31];
         app.NombredeplacesdisponiblesButton.Text = 'Nombre de places disponibles:';
            % Create EditField_2
            app.EditField_2 = uieditfield(app.UIFigure, 'text');
            app.EditField_2.ValueChangedFcn = createCallbackFcn(app, @EditField_2ValueChanged, true);
            app.EditField_2.Editable = 'off';
            app.EditField_2.HorizontalAlignment = 'center';
            app.EditField_2.FontWeight = 'bold';
            app.EditField_2.Position = [285 194 100 22];
        % Create LesplacesdisponiblesdanslesautresparkingsLabel
        app.LesplacesdisponiblesdanslesautresparkingsLabel = uilabel(app.UIFigure);
        app.LesplacesdisponiblesdanslesautresparkingsLabel.FontSize = 18;
          app.LesplacesdisponiblesdanslesautresparkingsLabel.FontWeight = 'bold';
```

```
  app.LesplacesdisponiblesdanslesautresparkingsLabel.Position = [126 156 448 24];
app.LesplacesdisponiblesdanslesautresparkingsLabel.Text = 'Les places
disponibles dans les autres parkings : ';
        % Create ParkingBButton
        app.ParkingBButton = uibutton(app.UIFigure, 'push');
        app.ParkingBButton.ButtonPushedFcn = createCallbackFcn(app, @ParkingBButtonPushed, true);
        app.ParkingBButton.FontSize = 18;
        app.ParkingBButton.FontWeight = 'bold';
        app.ParkingBButton.FontColor = [1 0 0];
        app.ParkingBButton.Position = [94 91 100 31];
        app.ParkingBButton.Text = 'Parking B';
        % Create ParkingCButton
        app.ParkingCButton = uibutton(app.UIFigure, 'push');
app.ParkingCButton.ButtonPushedFcn = createCallbackFcn(app, @ParkingCButtonPushed, true);
        app.ParkingCButton.FontSize = 18;
        app.ParkingCButton.FontWeight = 'bold';
        app.ParkingCButton.FontColor = [1 0 0];
        app.ParkingCButton.Position = [94 32 100 31];
        app.ParkingCButton.Text = 'Parking C';
        % Create EditField_3
        app.EditField_3 = uieditfield(app.UIFigure, 'text');
        app.EditField_3.ValueChangedFcn = createCallbackFcn(app, @EditField_3ValueChanged, true);
        app.EditField_3.ValueChangingFcn = createCallbackFcn(app, @EditField_3ValueChanging, true);
        app.EditField_3.Editable = 'off';
        app.EditField_3.HorizontalAlignment = 'center';
        app.EditField_3.FontWeight = 'bold';
        app.EditField_3.Position = [292 95 100 22];
% Create EditField_4
        app.EditField_4 = uieditfield(app.UIFigure, 'text');
        app.EditField_4.ValueChangedFcn = createCallbackFcn(app, @EditField_4ValueChanged, true);
  app.EditField_4.Editable = 'off';
  app.EditField_4.HorizontalAlignment = 'center';
```

```
app.EditField_4.FontWeight = 'bold';
app.EditField_4.Position = [292 36 100 22];
% Create DOUIRALabel
app.DOUIRALabel = uilabel(app.UIFigure);
app.DOUIRALabel.FontSize = 18;
app.DOUIRALabel.FontWeight = 'bold';
app.DOUIRALabel.FontColor = [0 0.4471 0.7412];
app.DOUIRALabel.Position = [455 94 76 24];
app.DOUIRALabel.Text = 'DOUIRA';
% Create KHRAICIALabel
app.KHRAICIALabel = uilabel(app.UIFigure);
app.KHRAICIALabel.FontSize = 18;
app.KHRAICIALabel.FontWeight = 'bold';
app.KHRAICIALabel.FontColor = [0 0.4471 0.7412];
app.KHRAICIALabel.Position = [455 35 93 24];
app.KHRAICIALabel.Text = 'KHRAICIA';
% Create ListedesCOMDropDownLabel
app.ListedesCOMDropDownLabel = uilabel(app.UIFigure);
app.ListedesCOMDropDownLabel.HorizontalAlignment = 'right';
app.ListedesCOMDropDownLabel.Position = [506 403 86 22];
app.ListedesCOMDropDownLabel.Text = 'Liste des COM';
% Create ListedesCOMDropDown
app.ListedesCOMDropDown = uidropdown(app.UIFigure);
    app.ListedesCOMDropDown.DropDownOpeningFcn = createCallbackFcn(app, @ListedesCOMDropDownOpening, tru
    app.ListedesCOMDropDown.ValueChangedFcn = createCallbackFcn(app, @ListedesCOMDropDownValueChanged, tr
app.ListedesCOMDropDown.Position = [486 378 133 22];
% Create ArduinoLabel
app.ArduinoLabel = uilabel(app.UIFigure);
app.ArduinoLabel.FontWeight = 'bold';
app.ArduinoLabel.Position = [524 424 49 22];
app.ArduinoLabel.Text = 'Arduino';
```

```
    % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end
    % App creation and deletion
    methods (Access = public)
        % Construct app
        function app = ParkingA
            % Create UIFigure and components
            createComponents(app)

            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            % Execute the startup function
            runStartupFcn(app, @startupFcn)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```

Programme MATLAB de gestion du parking B

```
classdef ParkingB < matlab.apps.AppBase
% Properties that correspond to app components
properties (Access = public)
```

```
    UIFigure                matlab.ui.Figure
    ParkingBLabel           matlab.ui.control.Label
    PlacesdisponiblesLabel  matlab.ui.control.Label
    EditField               matlab.ui.control.EditField
    ArduinoPanel            matlab.ui.container.Panel
    ListedesCOMDropDownLabel  matlab.ui.control.Label
    ListedesCOMDropDown     matlab.ui.control.DropDown
    ArduinoConnexionButton  matlab.ui.control.Button
    Lamp                    matlab.ui.control.Lamp
    DeconnexionButton       matlab.ui.control.Button
end
    % Callbacks that handle component events
        methods (Access = private)
     % Code that executes after component creation
            function startupFcn(app)
                list = getAvailableComPort();% for 2015
                app.ListedesCOMDropDown.Items=list;
            end
    % Button pushed function: ArduinoConnexionButton
            function ArduinoConnexionButtonPushed(app, event)
             global port
             instrreset
             com = app.ListedesCOMDropDown.Value;
             port=serial(com,'BaudRate',9600);
             app.Lamp.Color=[0,1,0];
              fopen(port);
              pause(3);
              while (1)
              idn = fscanf(port);
              if ~isempty(idn)
                 idn
```

```
            if (length(idn)>6) & (idn(1:5)=='place')
                save('var.mat','idn');
                p=app.EditField.Value;
                p=['PB',p];
                fprintf(port,p);
                pause(2);
            end
        end
        pause(2);
        end
        fclose(port);
    end
% Button pushed function: DeconnexionButton
    function DeconnexionButtonPushed(app, event)
global port
        port
        fclose(port);
        port
        app.Lamp.Color=[1,0,0];
    end
  end
% Component initialization
   methods (Access = private)

  % Create UIFigure and components
function createComponents(app)
  % Create UIFigure and hide until all components are created
  app.UIFigure = uifigure('Visible', 'off');
  app.UIFigure.Position = [100 100 640 480];
  app.UIFigure.Name = 'MATLAB App';
% Create ParkingBLabel
```

```
app.ParkingBLabel = uilabel(app.UIFigure);
app.ParkingBLabel.FontSize = 20;
app.ParkingBLabel.FontWeight = 'bold';
app.ParkingBLabel.Position = [254 408 133 27];
app.ParkingBLabel.Text = 'Parking    B  ';
% Create PlacesdisponiblesLabel
app.PlacesdisponiblesLabel = uilabel(app.UIFigure);
app.PlacesdisponiblesLabel.FontSize = 14;
app.PlacesdisponiblesLabel.FontWeight = 'bold';
app.PlacesdisponiblesLabel.FontColor = [0 0 1];
app.PlacesdisponiblesLabel.Position = [61 66 141 22];
app.PlacesdisponiblesLabel.Text = 'Places disponibles ';
% Create EditField
app.EditField = uieditfield(app.UIFigure, 'text');
app.EditField.Position = [287 66 100 22];
% Create ArduinoPanel
app.ArduinoPanel = uipanel(app.UIFigure);
app.ArduinoPanel.TitlePosition = 'centertop';
app.ArduinoPanel.Title = 'Arduino';
app.ArduinoPanel.BackgroundColor = [0.9412 0.9412 0.9412];
app.ArduinoPanel.FontWeight = 'bold';
app.ArduinoPanel.Position = [45 163 194 178];
% Create ListedesCOMDropDownLabel
app.ListedesCOMDropDownLabel = uilabel(app.ArduinoPanel);
app.ListedesCOMDropDownLabel.HorizontalAlignment = 'right';
app.ListedesCOMDropDownLabel.Position = [46 128 92 22];
app.ListedesCOMDropDownLabel.Text = ' Liste  des COM';
% Create ListedesCOMDropDown
app.ListedesCOMDropDown = uidropdown(app.ArduinoPanel);
app.ListedesCOMDropDown.Items = {};
app.ListedesCOMDropDown.Position = [19 107 158 22];
```

```
        app.ListedesCOMDropDown.Value = {};
        % Create ArduinoConnexionButton
        app.ArduinoConnexionButton = uibutton(app.ArduinoPanel, 'push');
            app.ArduinoConnexionButton.ButtonPushedFcn = createCallbackFcn(app, @ArduinoConnexionButtonPushed, true);
        app.ArduinoConnexionButton.Position = [19 63 119 24];
        app.ArduinoConnexionButton.Text = 'Arduino Connexion';
        % Create Lamp
        app.Lamp = uilamp(app.ArduinoPanel);
        app.Lamp.Position = [157 47 20 20];
        app.Lamp.Color = [1 0 0];
        % Create DeconnexionButton
        app.DeconnexionButton = uibutton(app.ArduinoPanel, 'push');
     app.DeconnexionButton.ButtonPushedFcn = createCallbackFcn(app, @DeconnexionButtonPushed, true);
        app.DeconnexionButton.Position = [19 19 119 24];
        app.DeconnexionButton.Text = 'Deconnexion';
        % Show the figure after all components are created
        app.UIFigure.Visible = 'on';
    end
end
     % App creation and deletion
methods (Access = public)
    % Construct app
    function app = ParkingB
        % Create UIFigure and components
        createComponents(app)
        % Register the app with App Designer
        registerApp(app, app.UIFigure)
        % Execute the startup function
        runStartupFcn(app, @startupFcn)
        if nargout == 0
            clear app
```

```
    end
end
    % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
    end
```

Références

[1] A. Waheed, J. Shafi et V. K. P, Analyzing significant reduction in traffic by using restricted smart parking, Springer Singapore éd., vol. 1054, 2020, pp. 27- 40.

[2] W. Amatul, J. Shafi et P. Vinkata Krishna, analyzing significant reduction, Springer, Singapore éd., vol. 1054, 2020, pp. 27-40.

[3] A. Boumergued et S. Nasri, *Conception et réalisation d'une application mobile pour un parking intelligent,* Bordj Bou Arréridj: Mémoire de master, université de BBA, 2020.

[4] B. Michel , «laquotidienne.fr,» 08 01 2016. [En ligne]. Available: www.laquotidienne.fr/un-parking-intelligent-a-laeroport-de-nice/. [Accès le 20 05 2022].

[5] B. P. Association, «PPA,» [En ligne]. Available: britishparking.co.uk. [Accès le 17 07 2022].

[6] R. George, «Britannica,» 21 Juin 2022. [En ligne]. Available: https://www.britannica.com/. [Accès le 17 Juillet 2022].

[7] «Metinvest,» 09 Décembre 2020. [En ligne]. Available: metinvestholding.com. [Accès le 27 avril 2022].

[8] E. B. Joseph, «The mit press reader,» 1992. [Accès le 02 Mai 2022].

[9] A. Humbert et M. Jule, «Anecdotes et histoires sur le stationnement,» *Le figaro,* 2019.

[10] «Designa,» [En ligne]. Available: https://designa.com/fr-fr. [Accès le 16 Juillet 2022].

[11] K. Margrit, H. Nico, B. Stian, S. Tom, M. Oystien et K. Per Egil, «Short Communication: Detecting Heavy Goods Vehicles in Rest Areas in Winter Conditions Using YOLOv5,» *Algorithms,* vol. 14, p. 4, 13 Mars 2021.

[12] . B. Mustapha, P. Camille, R. Fano , P. Gauthier et B. Olivier , «Multi-agent based governance model for Machine-to-Machine networks in a smart parking management system,» International conference on communication ICC, Ottawa,Canada, 2012 IEEE.

[13] H. Yao-Huei et H. Cheng-Hung, «A decision support system for available parking slots on the roadsides in urban areas,» *Expert systems with applications,* vol. 205, n° %1117668, 2022.

[14] R. V. Racunas, Jr., «Systems and methods for internet». USA patent Brevet 6,750,786 B1, 15 June 2004.

[15] M. Venkata Sudhakar, A. Anoora Reddy, K. Mounika, M. Sai Kumar et T. Bharani, «Development of smart parking management system,» Materials Today: Proceedings, 2021.

[16] . A. S, M. Shreya et . A. A. Al-Absi, « Proceedings of International Conference on Smart Computing and Cyber Security,» chez *Conference proceedings*, India, Strategic Foresight, Security Challenges and Innovation (SMARTCYBER 2020).

[17] K. Hamoudi, «Algérie 360,» 04 septambre 2014. [En ligne]. Available: www.algerie360.com/stationnement-anarchique-des-vehicules-la-police-durcit-les-mesures/. [Accès le 09 septembre 2022].

[18] A. RAKOTONDRASOLO, *Systeme de getion de parking avec Arduino,* Antananarivo: mémoire université d'ANTANANARIVO, 2015.

[19] B. Affagard, J.-N. Lafargue et J.-M. Géridan, Projets créatifs avec Arduino, Pearson, Éd., Paris, 2014, p. 300.

[20] M. Banzi, Getting Started With Arduino, Springer, Singapore, 2018.

[21] E. Mai, «SCRIBD,» [En ligne]. Available: www.scribd.com/document/508773788/Module-GSM-SIM-900#. [Accès le 22 Juin 2022].

[22] «LISTENVISION,» [En ligne]. Available: http://en.listenvision.cn/Fullcolor/Q1plus.shtml. [Accès le 20 Juillet 2022].

[23] «Matlab : tout savoir sur le langage de calcul numérique,» *JDN,* 2022.

[24] «LISTENVISION,» [En ligne]. Available: http://en.listenvision.cn/softwaredownload/. [Accès le 04 Septembre 2022].