RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ IBN-KHALDOUN DE TIARET

FACULTÉ DES SCIENCES APPLIQUEES

DÉPARTEMENT DE GENIE ELECTRIQUE

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du diplôme de Master

Domaine : Sciences et Technologie

Filière : Génie Electrique

Spécialité : Automatique et Informatique Industriel

THÈME

Parking souterrain
Cas1 : Etude, simulation et réalisation avec une carte Arduino
Cas2 : Etude et simulation avec un API

Préparé par :

- **BENSEDDIK Ichrak**

Devant le Jury :

Morsli SBAA	MCA	Président
Rahal OUARED	MAA	Examinateur
Hamid BOUMEDIENNE	MCB	Examinateur
Mohamed KOUADRIA	MAA	Encadreur

Année universitaire 2019/2020

Dédicace

Je dédie ce mémoire à ……..

Mes chers parents

Que nulle dédicace ne peut exprimer mes sincères sentiments

Pour leur patience illimitée, leur encouragement continu, leur aide, leur profond amour et respect pour leurs grands sacrifices. Sans oublier mes chers grands parents pour leur assistance.

Mes chers frères

Pour leur grand amour et leur soutien, qu'ils trouvent ici l'expression de ma haute gratitude.

Mes chers amis

Qui sans leur encouragement ce travail n'aura jamais vu le jour.

Et à toute la famille benseddik et à tous ceux que j'aime

Remerciements

Je remercie tout d'abord mes chers parents qui m'ont soutenu tout au long de mes cursus scolaire et universitaire jusqu'à la réalisation de mon mémoire.

J'adresse mes vifs remerciements à mon encadreur Dr KOUADRIA Mohamed pour leurs qualités scientifiques qui ont été très précieuses pour mener à bien mon modeste travail.

Enfin je remercie toute personne ayant contribué de près ou de loin à la réalisation de ce projet de fin d'études.

J'exprime mes sincères remerciements à Dr M. SBAA pour l'honneur qu'il me fait en acceptant de présider le jury de soutenance.

Je remercie également Dr H. BOUMEDIENNE et Dr R. OUARED pour l'intérêt qu'il a porté à ce travail en acceptant de l'examiner.

Enfin je remercie toute personne ayant contribué de près ou de loin à la réalisation de ce projet de fin d'études.

دراسة ، محاكاة وانجاز موقف سيارات ذكي تحت الأرض

ملخص بالنظر إلى عدد المركبات الذي يستمر في النمو ، فإن المشكلة الكبيرة بين تلك التي نواجهها في المدينة هي وقوف السيارات. في السنوات الأخيرة ، بفضل التطور التكنولوجي ، ظهر مفهوم أحدث ثورة في مجال وقوف السيارات. يتعلق الأمر بمواقف السيارات الذكية. وبالتالي ، فإن العمل المقترح في هذه الأطروحة يتعلق بدراسة نظام ذكي مع لوحة Arduino ، وبالتالي ، فإن العمل المقترح في هذه الأطروحة يتعلق بدراسة نظام ذكي مع لوحة Arduino ، والتي تسمح ، بشكل أساسي ، بالتحكم في مداخل ومخارج موقف سيارات تحت الأرض. وبالتالي ، من خلال هذا النظام ، هناك إمكانية عرض عدد الأماكن المتاحة ، وذلك خارج ساحة الانتظار وداخلها على حد سواء لإراحة سائق السيارة من سهولة الوصول إلى المساحة الخالية. بالإضافة إلى ذلك ، تم إجراء دراسة أخرى ، بناءً على مبدأ وحدات التحكم المنطقية القابلة للبرمجة ، وبالتالي يتم استخدام جهاز محاكاة من نوع PLC Siemens S7-200وبرنامج Step 7 Micro / WIN للبرمجة. نتائج المحاكاة التي تم الحصول عليها مرضية وحالة انجاز باستعمال Arduino مرضية، مما يدل على حسن سير البرنامج.

الكلمات الجوهرية اردوينو، سائق السيارة، ذكي، موقف السيارات، حيز، محاكي ، تحكم المنطق.

Study, simulation and realization of an intelligent underground car park

Abstract Considering the number of vehicles which continues to grow, the big problem among those encountered in a city is parking. In recent years, thanks to technological development, a concept has emerged to revolutionize the field of parking. It's about smart parking. Thus, the work proposed in this thesis concerns the study of a smart system with an Arduino board, which allows, mainly, to control the entries and exits of an underground car park. So, through this system, it is possible to display the number of places available, and this outside as well as inside the parking lot to alleviate the motorist of having easy access to the vacant space. In addition, another study was carried out, based on the principle of programmable logic controllers, therefore the simulator of the PLC type Siemens S7-200 and the Step 7 Micro / WIN software for programming are used. The results obtained from the simulations and the case of realization with Arduino are satisfactory are satisfactory, which shows the correct functioning of the program.

Keywords Arduino, Motorist, Smart, Parking, Place, Simulator, logic controller .

Etude, simulation et réalisation d'un parking intelligent souterrain

Résumé Vu le nombre de véhicule qui ne cesse de croitre, le gros problème parmi ceux rencontrés dans une ville est le stationnement. Depuis quelques années, grâce au développement technologique, un concept a émergé pour révolutionner le domaine du stationnement. Il s'agit du stationnement intelligent. Ainsi, le travail proposé dans ce mémoire concerne l'étude d'un système intelligent à carte Arduino, ce qui permet, principalement, de contrôler les entrées et les sorties d'un parking souterrain. Ainsi, à travers ce système, il y a possibilité d'affichage du nombre de places disponibles, et cela à l'extérieur comme à l'intérieur du parking pour soulager l'automobiliste d'accéder facilement à l'espace vacant. De plus, une autre étude a été effectuée, en se basant sur le principe des automates programmables, donc le simulateur de l'automate de type **Siemens S7-200** et le logiciel **Step 7 Micro/WIN** pour la programmation sont utilisés. Les résultats obtenus des simulations et du cas de réalisation avec Arduino sont satisfaisants, ce qui montre le bon fonctionnement des programmes.

Mots-clés Arduino, Automobiliste, Intelligent, Parking, Place, Stationnement, Simulateur, Automate.

Liste des figures

Chapitre 1

Chapitre2

Chapitre 3

Chapitre 4

Liste des tableaux

Tables des matières

Introduction générale

Le nombre de véhicule ne cesse de croître sur la surface du globe. Les statistiques sur ce nombre en circulation sont par définition imprécises, parce que ce nombre fluctue en permanence. Tous les jours de nouvelles voitures, de nouveaux bus, de nouveaux camions, de nouveaux utilitaires légers sont sur les routes et d'autres envoyés à la casse **[1].** Ce qui a causé la congestion urbaine du trafic. Ce problème, qui fait partie du quotidien, n'est pas nouveau mais il a gagné considérablement de l'espace **[2].** Cela démontre clairement la nécessité d'une gestion du stationnement.

Aujourd'hui, grâce à la révolution technologique, les recherches ont abouti à l'invention du parking intelligent, où les automobilistes peuvent trouver un parking rapidement et facilement. Ce concept n'existe encore que dans les pays développés **[3].**

Notre projet de fin d'étude marque la fin de nos études. Il s'attarde sur l'étude et la simulation d'un parking intelligent souterrain commandé par une carte arduino et un automate programmable industriel.

Comme le sujet a fait l'objet de plusieurs études et n'a jamais été traité au sein de notre université, il a été décidé de porter notre choix dans ce sens.

L'exploitation de ce type de parking, essentiellement due au gain de temps pour l'automobiliste, économie du carburant,..etc **[3].**

Le contenu de ce projet de fin d'étude est scindé en quatre chapitres.

Le premier chapitre est consacré à une généralité sur le thème de notre travail, où nous présentons dans un premier temps quelques définitions et types suiveurs de ligne. Dans un deuxième temps, nous donnons le principe de fonctionnent et l'organisation générale d'un suiveur de ligne. Nous terminons ce chapitre par les nouvelles technologies mises en œuvre en Algérie par rapport à certains pays étrangers.

Le cadre général de la carte à microcontrôleur « ARDUINO » et les automates programmables industriels (API) feront l'objet du deuxième chapitre. Dans ce cas, nous allons souligner les différentes cartes ARDUINO ainsi que quelques types d'automates. Ensuite, nous portons un intérêt sur les avantages et les inconvénients, et plus

particulièrement une description détaillée et les caractéristiques de la carte ARDUINO et l'API impliqués dans notre projet.

Le troisième chapitre concerne l'étude détaillée du système de notre projet dans les cas deux cas de situations : avec ARDUINO et avec un API. La suite de ce chapitre sera consacrée à la présentation du simulateur (ISIS Proteus) utilisé pour la conception des schémas électroniques correspondant aux différents circuits étudiés, et à l'illustration des différents organigrammes correspondant aux cas de figures des simulations effectuées.

Enfin, le dernier chapitre concerne la maquette à carte Arduino pour la simulation du fonctionnement du parking intelligente et l'évaluation du coût de sa réalisation.

Nous terminons ce mémoire par une conclusion portant sur les travaux effectués et par une présentation des perspectives de recherche pouvant être envisagées.

Chapitre1 :

Généralités sur le parking de stationnement

1.1 Introduction

Un milieu de stationnement réservé à un usage privé ou étatique est considéré comme le lieu le plus répandu dans toute région d'un pays. Les avancées technologiques ont pu créer aujourd'hui le parking intelligent et digital. Mais malheureusement certains pays restent toujours en dessous des normes internationales avec des niveaux technologiques relativement dépassés au regard des standards technologiques en vigueur dans le monde.

Dans ce cadre, le présent chapitre sera abordé par une présentation des différents types de lieux intelligents et de leur cadre général dans le but de citer les avantages et les inconvénients. Ensuite, un contexte sera établi pour donner l'organisation générale du système d'un parking intelligent. Après cela, un autre contexte sera présenté pour montrer le cas de nouvelles technologies exploitées au niveau local (en Algérie) et niveau de certain pays étrangers.

1.2 Définitions

Une définition des principaux termes auxquels nous aurons éventuellement recours dans notre étude est donnée dans ce qui suit :

1.2.1 Intelligence

Qualité de quelqu'un qui manifeste dans un domaine donné un souci de comprendre, de réfléchir, de connaître et qui adapte facilement son comportement à ces finalités [4].

1.2.2 Lieu intelligent

Un lieu intelligent (en anglais **smart** city) est un lieu utilisant les technologies de l'information et de la communication (TIC) pour améliorer la qualité d'usage ou réduire leurs coûts [5].

1.2.3 Système

Un **système** est un ensemble d'éléments interagissant entre eux selon certains principes ou règles [6].

1.2.4 Capteur

Dispositif permettant de capteur un phénomène physique et de le restituer sous forme de signal [7]. Un capteur assure l'interface de mesure entre le système et le processus [8].

1.2.5 Actionneur

Appareil ou organe permettant d'agir sur une machine ou un processus en vue de modifier

son comportement ou son état [9]. .

1.2.6 Stationnement intelligent, ou smart parking

Une application de technologies avancées pour améliorer la rapidité et l'efficacité avec lesquelles un automobiliste peut localiser, réserver et payer pour obtenir un espace de stationnement [10].

1.2.7 Parking souterrain

Un parking souterrain est un parking qui se trouve généralement en dessous des bâtiments tels que les aéroports, les centres commerciaux ou encore les bureaux d'entreprises. C'est un parking très sécurisé avec de nombreuses installations modernes [11].

1.2.8 Automate programmable

Un automate programmable industriel (API) est une forme particulière de contrôleur à microprocesseur qui utilise une mémoire programmable pour stocker les instructions et qui implémente différentes fonctions, qu'elles soient logiques, de séquencement, de temporisation, de comptage ou arithmétiques, pour commander les machines et les processus [12].

1.3 Types de parking

Plusieurs types de parkings sont touchés par la technologie intelligente, quelques exemples courants sont présentés dans les figures suivantes [13]:

(a)

(b)

(C)

(d) (e)

Figure 1.1 *Exemple de parking de stationnement de voitures*
(a) Parking en surface
(b) Parking souterrain intelligent
(c) GPS pour trouver en place de stationnent et parking connecté
(d) Parking à étage
(e) Smart parking

1.4 Contexte historique et son évolution

- **En 2004** : pour évaluer la faisabilité du concept de stationnement intelligent dans un contexte de transport en commun, des partenaires publics et privés ont lancé conjointement un essai sur le terrain de **stationnement intelligent** à la station Rockridge BART à Oakland, en Californie [14].
- **Du 2004 à 2005** : le service de stationnement intelligent était gratuit, lorsque BART a mis en place des frais de service.

- A partir de 2005: le système de stationnement intelligent mis en œuvre principalement en Europe, aux États-Unis et au Japon (Shaheen et al., 2005) est développé avec l'incorporation de technologies de pointe et de recherches issues de diverses disciplines universitaires.

- **A partir de 2008** : de nombreux spécialiste s'intéressent à ce concept du parking intelligent pour permettre un meilleur soutien possible pour la réalisation des tâches au quotidian. comme le cas en Europe, Réalisation d'un système de stationnement intelligent capable d'indiquer en temps réel aux automobilistes les places disponibles, tout en offrant de nouvelles solutions de paiement [15].

- **En 2017** : le déploiement des technologies de capteurs continue d'être au cœur du développement du stationnement intelligent, une grande variété d'autres innovations technologiques permettent également des systèmes plus adaptables, notamment des caméras, des communications sans fil [16].

- **En 2019:** des solutions sont proposées par plusieurs entreprises à Genève (Suisse). Comme, *IEM*, un acteur européen incontournable dans le monde du stationnement sur voirie, active dans la conception d'une technologie que les autres pays n'ont pas à disposition, et qui permet de repèrer les places de parc libre par guidage internet, l'essence même du parking intelligent, voir figure I.2 [17].

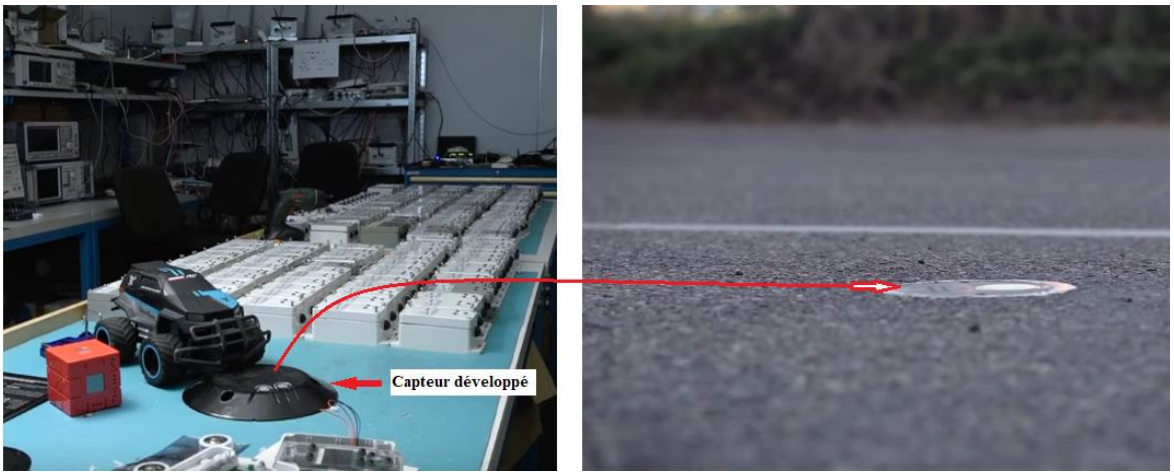


Figure 1.2 *Capteur pour le repérage des places de stationnement*

1.5 Cadre général d'un parking intelligent [18]

- Sécurité**:**
  - ✓ la sécurité est assurée grâce à un contrôle serré des entrées et des sorties.
  - ✓ En prévention contre les risques d'accident
  - ✓ Les caméras de sécurité, les panneaux d'avertissement informatifs, les bornes et les rampes d'accès offrent une surveillance supplémentaire et rehaussent d'un cran la sécurité.
- Les économies d'énergie :
  - ✓ En évitant le gaspillage et les dépenses inutiles. Par exemple, Des capteurs intégrés dans le sol ou des caméras montées sur des poteaux d'éclairage ou des structures de bâtiments déterminent si les places de stationnement sont occupées ou disponibles. Ces données sont acheminées sans fil vers une passerelle et relayées vers une plate-forme centrale de stationnement intelligent basée sur le cloud. Il est combiné avec les données d'autres capteurs pour créer une carte de stationnement en temps réel.

Comme toute installation récente, la technologie intelligente compte un certain nombre d'inconvénients :

- L'un des inconvénients des parkings intelligents est leur complexité perçue ; certaines personnes ont de la difficulté avec la technologie. Ils éviteront de se rendre dans ces endroits et iront vers d'autres destinations avec un stationnement plus facile [19].
- Dépendance à la qualité du signal internet. Ainsi si le signal est mauvais, les objets fonctionnement mal.
- Le coût d'installation d'un parking intelligent est très cher mais également son entretien.

1.6 Avantages et inconvénients d'un parking fermé et souterrain [20]

- Pour une ville, le parking souterrain n'a pas d'impact sur le paysage urbain. Il permet d'économiser le foncier.
- C'est un parking discret et très sécurisé.
- Il dispose de nombreuses places de stationnement, séparées par des piliers qui supportent l'ensemble de la structure.
- L'endroit est assez fréquenté en plus du personnel de nettoyage, agents de maintenance, etc.

Inconvénients

- La qualité de l'air est la principale problématique du parking souterrain, mais des normes ont été établies afin de minimiser la pollution dans ce type de parking.

1.7 Principe de fonctionnement

L'intelligence des parkings a pour principe de programmer et de contrôler à distance ou localement différents appareils électriques qui auront préalablement été intégrés dans un réseau. Ainsi, les équipements électriques peuvent communiquer entre eux grâce à une émission d'informations entre les unités de commandes et les appareils.

Les informations ainsi envoyées circulent aussi bien dans le sens "unité de commande-appareils" (afin d'envoyer les informations nécessaires à la réalisation d'une tâche) que dans le sens "appareils-unité de commande" (afin de faire part des informations sur leurs états).

Donc, le principe de fonctionnement d'un système à base de la technologie intelligente s'articule sur trois éléments suivants :

- ✓ Partie matériel : carte de développement, capteurs, actionneurs.
- ✓ Partie logiciel : les procédures de programmation.
- ✓ Un mode de transmission : pour garantir la communication entre les parties Hard et Soft.

Matériel	Algorithmique Logiciel	Mode de transmission
• Unité de traitement Les capteurs • Les pré-actionneurs et les Actionneurs	• Algorithme • Organigramme • Programme • Langage	• Liaison filaires • Onde radio • Bluetooth • Wifi • Infrarouge

Tableau 1.1 *Les éléments de base pour le fonctionnement d'un parking intelligent*

1.8 Organisation générale d'un parking intelligent

Pour automatiser intelligemment un parking, il faudra mettre en place un système à base d'une technologie intelligente. Un tel système est toujours constitué des mêmes équipements, quelque soit la technologie utilisée.

Figure 1.3 *Représentation schématique d'une organisation générale d'un parking intelligent*

1.8.1 Les circuits d'entrées

Ils sont de types très différents, et la plupart des capteurs délivrent des tensions ou des courants analogiques. Les capteurs les plus employés sont ceux qui mesurent [21] :

- Des grandeurs électriques (tension, courant, puissance),
- Les températures
- Les forces
- Les déplacements

Il est à noter que, certains de ces capteurs délivrent des tensions très faibles, de l'ordre du millivolt, il faut prendre des précautions spéciales pour que leur connexion au circuit de commande (par exemple, Arduino) n'introduise pas d'erreurs inacceptables.

La tendance actuelle en matière de capteurs consiste à utiliser des techniques numériques. La figure suivante illustre un exemple de cas de circuits introduits dans les applications à Arduino.

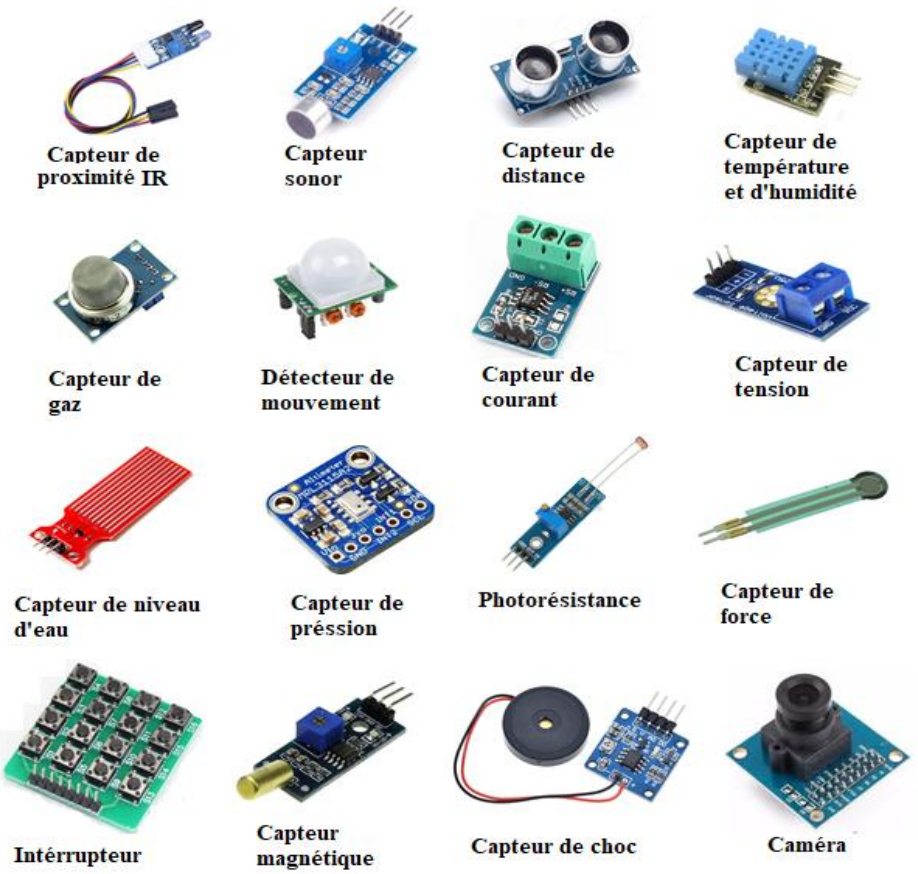


Figure 1.4 *Liste de quelques circuits d'entrées*

- **Les signaux d'entrées**

Sont des signaux envoyés par un circuit d'entrée dont les valeurs sont imposées à la partie de commande.

1.8.2 La partie de commande

Est un système (ensemble de circuits électroniques) qui génèrent des signaux destinés au fonctionnement de différents circuits de sorties. A travers la figure1.5, nous donnons un exemple de cartes de développements électroniques à base de microcontrôleurs. Celles-ci permettent d'exécuter des tâches en prenant les entrées de composants électroniques externes (capteurs) et de générer des signaux à leurs sorties. Le cas détaillée sur les types de cartes Arduino est étudié au chapitre 2.

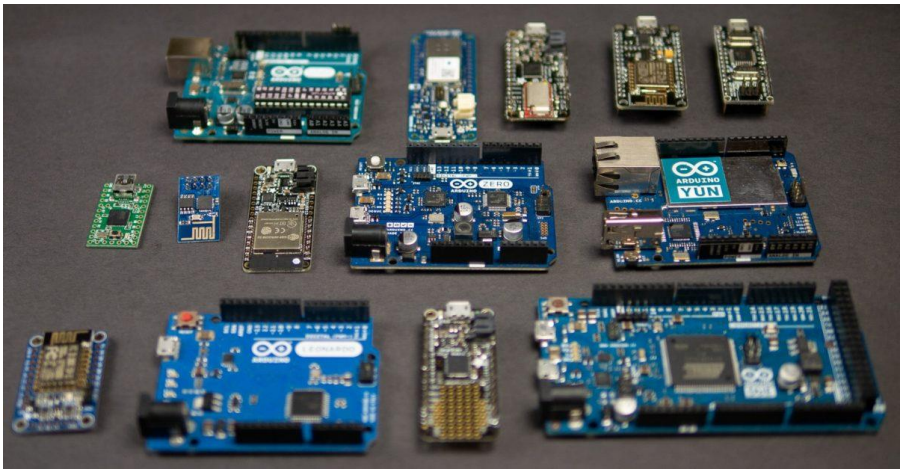


Figure 1.5 *Exemple de cartes de développements électroniques à base de microcontrôleurs*

- **Les signaux de sorties**

Représentent l'activité utile de la partie de commande. Ils en sont les seuls points observables, et ce sont eux qui permettent d'apprécier l'état de la partie commande.

1.8.3 Les circuits de sorties

Ce type de circuits sert à convertir un signal de commande généré par la partie de traitement en une action effective au niveau du système commandé.

Les circuits de sorties sont de types très divers et peuvent être des LEDs, des moteurs,..etc. La figure 1.6 illustre quelques circuits de sorties les plus utilisés.

Les circuits de sorties nécessitent en général une commande de puissance, et il est donc nécessaire de disposer circuits de puissance à l'interface entre le système informatique et les circuits de sorties [21].

Figure 1.6 *Liste de quelques circuits de sorties*

1.9 Nouvelles technologies mises en œuvre

Les technologies numériques constituent l'innovation majeure de ces dernières décennies et le principal vecteur de la nouvelle révolution industrielle dans les pays développés. Ces technologies ont radicalement transformé les modes de vie, les façons de produire, les rapports au temps et à l'espace, l'environnement culturel. De même, dans les pays en développement, notamment africains, où ces technologies connaissent une progression fulgurante [22].

1.9.1 En Algérie

En Algérie, à Alger, depuis plusieurs années les autorités locales ont décidé à lutter contre l'absence d'espaces de stationnements. Et pour ce faire, une seule solution est possible et imaginable : les **parkings intelligents**.

- En 2013 Lors d'une journée d'étude sur la mobilité urbaine organisée par l'Assemblée populaire de wilaya (APW) d'Alger, il a été annoncé qu'une expérience pilote sera tentée. L'objectif est de doter la capitale algérienne du premier parking "intelligent". Ce type de parkings, très répandu dans les grands centres urbains en Asie, consiste en une construction métallique en hauteur d'une capacité de 16 à 24 véhicules, démontable, adapté aux besoins des quartiers en matière de stationnement et dont le coût de la mise en œuvre serait de 150 millions DA.

La figure 1.7 montre un type de système de guidage pour parking qui a été proposé par une entreprise chinoise [23].

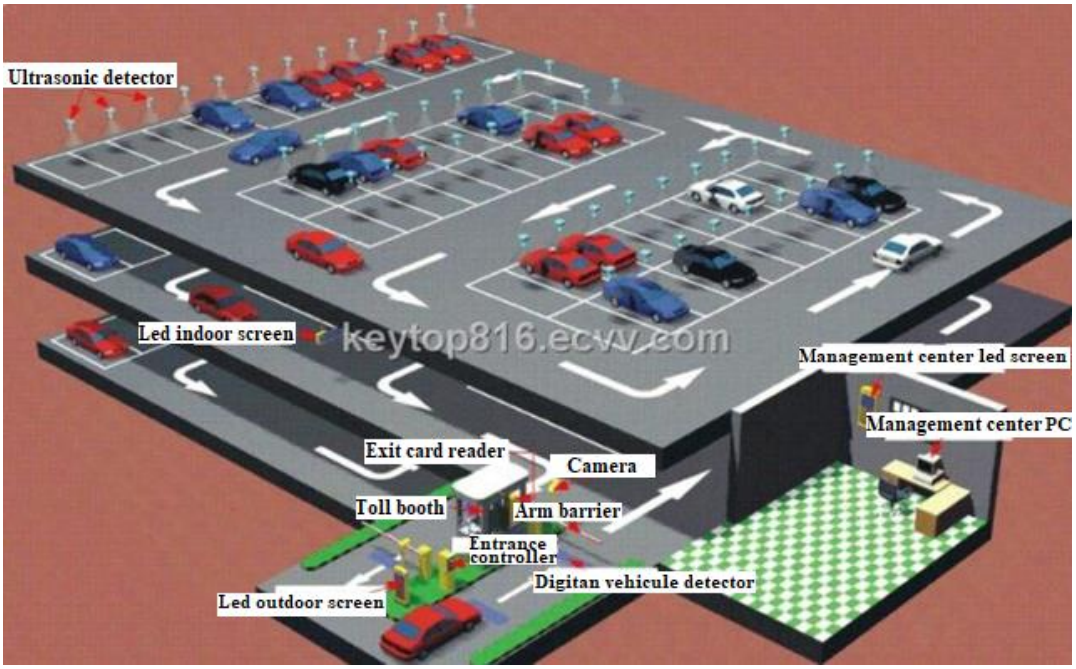


Figure 1.7 *Type de parking intelligent proposé en Algérie (2013)*

- Ainsi, un nouveau type de parking fait son apparition dans la commune de **Sidi M'hamed à Alger**. Il s'agit du premier **parking intelligent**, ou **Smart Parking**, implanté à la place du 1er mai, et qui est opérationnel depuis **1er novembre 2014**.

Ce projet unique au niveau national, a été lancé en coordination entre **l'APC de Sidi M'hamed** et l'**EGCTU** (Etablissement de gestion de la circulation et de transport urbain). L'équipement composé d'une structure en hauteur est capable de supporter jusqu'à **16 véhicules** sur une plateforme rotative. Le concept du **Smart Parking** a déjà montré son efficacité dans des pays tels que le Japon et la Russie. La figure 1.8 illustre le premier parking intelligent construit en Algérie [24].

Figure 1.8 *Premier parking intelligent en Algérie*

1.9.2 Aux pays étrangers

L'industrie évolue elle aussi rapidement en équipant ses parkings intelligents d'une puissance de calcul. Par exemple :

➢ Europe- En 2020, Here Technologies (Société basée aux Pays-Bas qui fournit de logiciels de planification d'itinéraires et de cartographie en ligne.) cherche à développer de nouvelles applications et services. Allant dans ce sens, le cartographe s'est rapproché du gestionnaire de parking européen APCOA pour réaliser des cartes HD et 3D des parkings couverts et souterrains européens. Ces cartes permettront de développer de nouveaux services, comme la pré-réservation de places de parkings [25].

➢ Asie- Depuis 2017, les parkings intelligents souterrains en chine sont équipés de robots qui se chargent de garer les voitures à la place de leurs propriétaires. Ainsi l'avantage dans ce cas, c'est optimiser l'espace et réduire le stress des chauffeurs. Donc, c'est l'une des coûteuses solutions mises en œuvre par des entreprises chinoises.

Figure 1.9 *Premier parking intelligent en Algérie*

➢ Occident- En 2020, Aversan labs (société d'ingénierie au Canada) propose une solution Smart Parking pour optimiser l'utilisation de l'espace de stationnement, améliorer l'efficacité des opérations de stationnement et faciliter la fluidité du trafic avec la prochaine génération de stationnement intelligent. Les nœuds de capteurs détectent la présence de voitures stationnées, ce qui permet de collecter la durée du stationnement, l'heure et d'autres analyses afin de prendre en charge le stationnement et la planification urbaine [26].

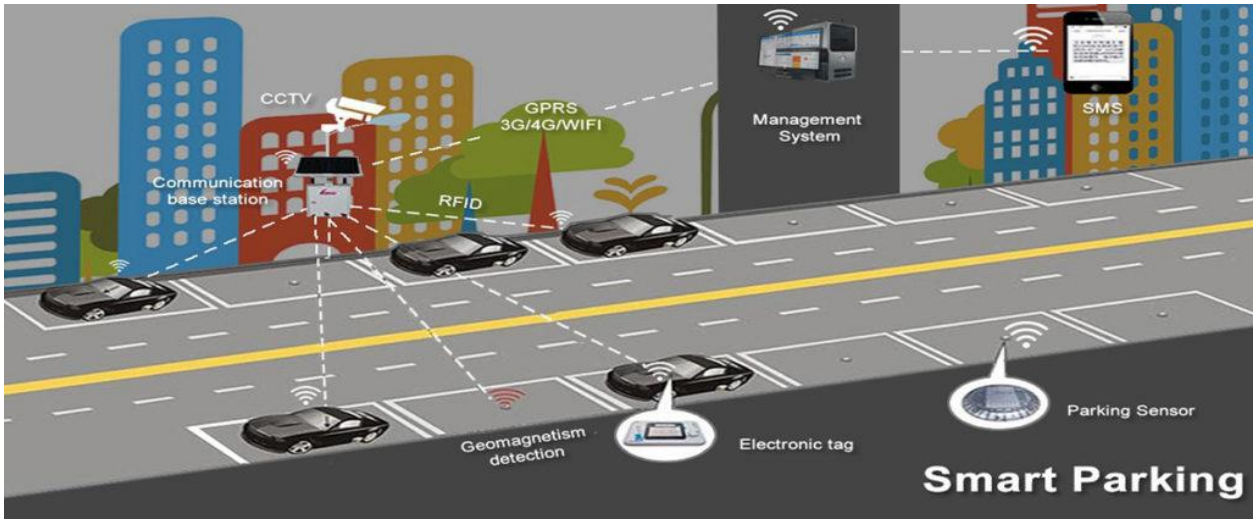


Figure 1.10 *Smart Parking proposé par une société Canadienne.*
Optimisation de l'utilisation de l'espace de stationnement.

1.10 Conclusion

À travers le présent chapitre, nous avons passé en revue les avantages et inconvénients d'un lieu intelligent ainsi que son organisation sous l'aspect le plus général. Après cela, nous

avons présenté les technologies numériques en Algérie par rapport à certains pays développés. C'est qui a montré que ces derniers ont pris plus de places dans la recherche pour l'optimisation du fonctionnement des parkings. Dans le prochain chapitre, nous allons voir une synthèse portant, sur la carte Arduino, ses types et les caractéristiques des cartes spécialement choisies pour notre travail.

Chapitre2 :

Cartes de développement à microcontrôleur et l'API

Chapitre 2 : Cartes de développement à microcontrôleur et l'API

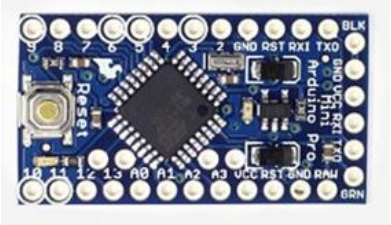
2.1. Introduction

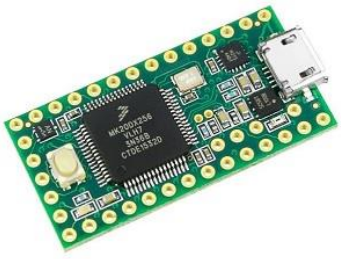
La commande des circuits électroniques à base des cartes à microcontrôleur est de plus en plus utilisée ces dernières années. Donc, différentes applications pour divers domaines ont recours à une multitude de cartes de développement, soit pour une autonomie de fonctionnement destiné à des taches bien précises, soit comme dispositifs d'interface entre les circuits d'entrées et ceux de sorties pour une meilleure efficacité de commande.

Ainsi, différentes cartes sont présentées dans le présent chapitre. Aussi, une autre intention est portée sur les caractéristiques de certaines cartes de type ARDUINO le plus communément utilisé. Encore, le principe d'utilisation de la carte ARDUINO UNO est bien particulièrement souligné. Donc, un choix sera porté sur une carte à microcontrôleur afin de l'appliquer dans le cas du système à étudier de ce projet.

2.2. Carte à microcontrôleur pour système embarqué

De nombreux modèles de cartes à microcontrôleur existent sur le marché.

Cartes microcontrôleur pour systèmes embarqués		
Carte Arduino permet aux utilisateurs de créer des applications électroniques interactifs à partir de cartes électroniques matériellement libres sur lesquelles se trouve un microcontrôleur [27].	**Nano** **12 €** **1823,15 DA**	
	Mini Pro **6,58 €** **1000 DA**	

	UNO **25 €** **3798,02** DA	
	UNO WiFi **40,20 €** **6107,54 DA**	
	Yen2 **58,80 €** **8933,43 DA**	
	Mega **35 €** **5 317,22 DA**	
Carte ChipKIT Spécialement conçu pour les Raspberry Pi sur la base du microcontrôleur 32 bits [28].	**WI-FIRE** **97.83 €** **14863,21 DA**	
Carte Teensy est un système complet de développement de microcontrôleurs basé sur USB, dans un très faible encombrement, capable de mettre en œuvre de nombreux types de projets. Toute la programmation se fait via le port USB [28].	**Teensy 3.2** **33,65 €** **5112,41 DA**	

Carte Raspberry Pi Elle est destinée à encourager l'apprentissage de la programmation informatique [28].	**3B- 1GB** **72,40 €** **11000 DA**	
	4 B – 4 GB **111,89 €** **17000DA**	
Carte LattePanda est un ordinateur monocarte compatible Arduino, fonctionnant sous Windows 10 ou Linux [28].	**DFR04182-GB/32GB** **152.50 €** **23169,17 DA**	
	DFR0545-Alpha 800s- 8GB **449,70 €** **68322,47 DA**	
Carte ArbotiX-M est un système de contrôle à distance évolué pour des robots de petites et moyennes tailles [29].	**ArbotiX-M** **51,90 €** **7885,12 DA**	

Tableau 2.1 *Quelques types de cartes à microcontrôleur pour les systèmes embarqués*

Généralement, les cartes à microcontrôleurs se caractérisent par [30] :

✓ Un plus haut degré d'intégration,

✓ Une plus faible consommation électrique,

✓ Une vitesse de fonctionnement plus faible (de quelques mégahertz jusqu'à plus d'un

gigahertz),

✓ Un coût réduit par rapport aux microprocesseurs utilisés dans les ordinateurs personnels.

Les systèmes intelligents sont en augmentation dans tous les domaines de la vie quotidienne. Voici des exemples d'utilisation des microcontrôleurs :

- **Télécommunications** : cartes FAX et MODEM, Minitel, téléphones portables (interfaces homme machine, gestion d'écrans graphiques)...

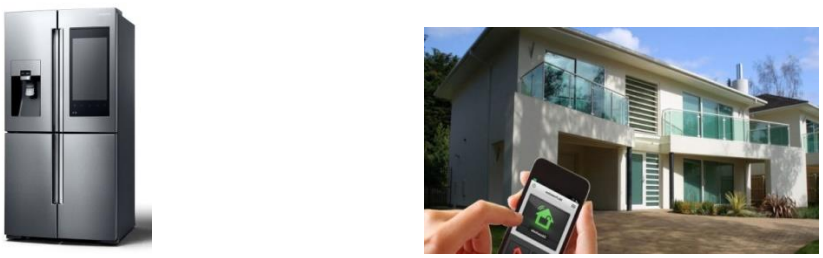
Figure 2.1 *Domaines de télécommunication intelligents*

(a) *Antenne intelligent* (b) *Téléphone portable intelligente*

- **Industriels** : automates programmables, contrôle de processus divers, supervision...

Figure 2.2 *Industrie intelligente*

- **Commercial** : électroménager, domotique...



(a) (b)

Figure 2.3 *Utilisation des microcontrôleurs dans le commercial*

(b) *Réfrigérateur intelligent* (b) *Maison intelligente*

- **Automobile** : ABS, tableau de bord, contrôle des sièges, des vitres...

Figure 2.4 *Voiture intélligente*

- **Militaire et spatial** : sonde, lanceurs de fusées, missile, robots...

Figure 2.5 *Les armes intelligentes*

- **Domaine scientifique :** Contrôle des sorties Arduino avec le PC. Dans ce cas, il y a une communication directe entre des plateformes externes et ARDUINO, à savoir Matlab, Visual basic, Delphi,..etc. La figure suivante illustre le cas d'asservissement de vitesse d'un moteur à courant continu [31].

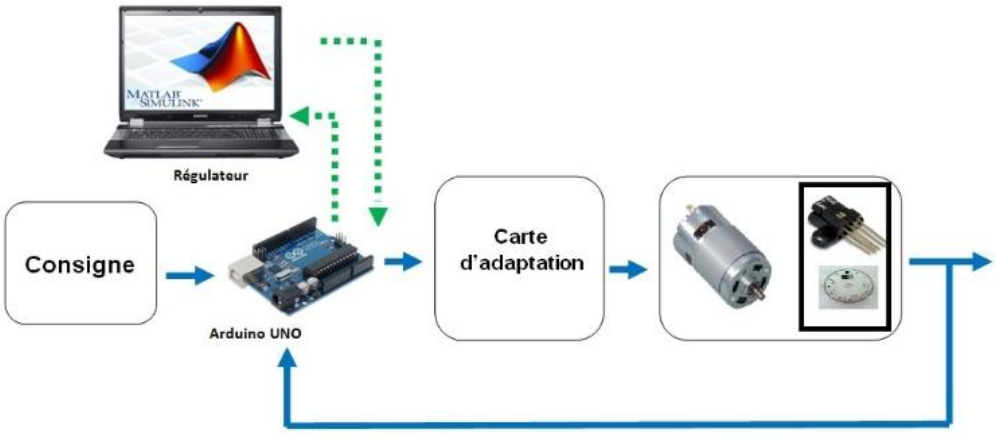


Figure 2.6 *Communication entre matlab et ARDUINO*

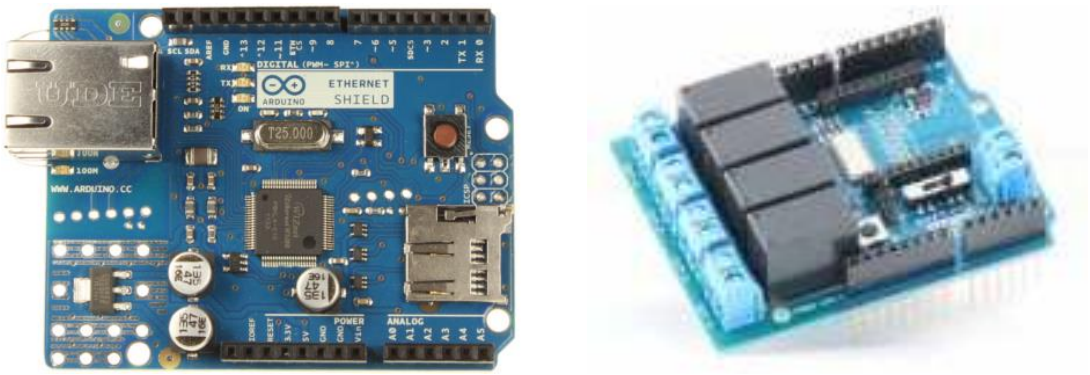
2.3 Cartes à microcontrôleur de type Arduino

- En 2004 : l'Arduino UNO a été développé par des enseignants et élèves d'une école de Design italienne, qui avaient pour vocation de démocratiser la programmation et de la rendre plus libre d'accès.

Les cartes à microcontrôleur de type ARDUINO sont disponibles avec de nombreux types de modules qui peuvent être reliés avec celui-ci. Ces modules sont connus sous le nom de « Shield ». Certains des shield les plus couramment utilisés sont:

- **Shields** Moteurs et servos : interface dédiée au pilotage des moteur
- **Shields** Afficheurs : permet de contrôler un afficheur
- **Shields** Relais : pour actionner des dispositifs
- **Shields** Ethernet : pour connecter un projet Arduino à Internet par le biais d'un câble RJ45
- **Shields** WIFI : permettant de connecter la carte à Internet
- **Shields** GSM : permet d'envoyer des SMS, MMS, GPRS et d'établir des communications audios.
- **Shields** Bluetooth : permet de commander un dispositif branché sur ARDUINO

- Exemple de cas de shileds



(a) (b)

Figure 2.7 *Shields pour carte ARDUINO*

(a) *Shield pour connexion à internet* (b) *Shield à relais isolés galvaniquement*

2.3.1 Caractéristiques des cartes de type Arduino

Caractéristiques	Carte ARDUINO					
	Nano	**Mini Pro**	**UNO**	**Leonardo**	**Yen**	**Méga**
Microprocesseur	ATMega328	ATmega328	ATmega328	ATmega32U4	ATmega32U4	ATMega2560
Mémoire flash	32 ko	32 Ko	32 ko	32 ko	32 ko	256 Ko
Mémoire SRAM	2 ko	2 Ko	2 ko	2,5 ko	2,5 ko	8 Ko
Mémoire EEPROM	1 ko	1 Ko	1 Ko	1 Ko	1 ko	4 Ko
broches d'entrées/ sorties	20	14	14	20	20	54
broches PWM	6	6	6	7	7	14
broches d'entrées analogiques	6	6	6	12	12	16
Courant par entrées-sorties	40 mA	40 mA	40 mA	40 mA	50 mA	40 mA
Fréquence d'horloge	16 MHz	16Mhz	16Mhz	16Mhz	16Mhz	16 MHz
Prise USB	mini-USB B	-	USB	USB	USB	USB
Dimensions	45 x 18 x 18 mm	33.8mm x 18mm	6.86cm x 5,3 cm	6.86cm x 5,3 cm	7cm x 5,3 cm	101,52 x 53,3 mm
Poids	5 g	2g	25g	20g	41g	37 g
Tension de fonctionnement	5V	7V – 9V	5V	5V	5V	5 V

Tableau 2.2 *Caractéristiques de quelques cartes de développement de type Arduino les plus utilisées*

2.3.2 Avantages et inconvénients

Dans cette partie, nous passons en revue quelques différentes cartes d'Arduino en mettant l'accent principalement sur leurs avantages et leurs inconvénients, Tableau 2.3.

Généralement, le principal avantage d'une carte à microprocesseur est le traitement rapide et une interface facile.

Type de carte Arduino	Avantages	Inconvénients
Uno	• Nombreux exemples de montage sont disponibles sur internet. • Nombres suffisant de broches d'E/S pour des projets élémentaires. • Vaste choix de **shields** • Bon marché	• Nombre de broche insuffisant pour les gros projets • La mémoire disponible risque d'être un peu juste pour projets ambitieux. • Ne peut pas être utilisée comme hôte USB pour simuler périphérique tel qu'un clavier ou une souris. • Acquiert des modules externes qui augmenteront inévitablement le coût total d'achat
Nano	• Encombrement réduit • Possibilité d'être enfichée directement sur la plaque d'essai	• Il n'est pas possible d'utiliser des **shields**
Mini Pro	• Taille réduite et son faible coût • Fournie sans connecteurs pour être facilement intégrée à tout projet final.	• Ne possède de convertisseur USB / série. • Ne possède pas non plus de sortie 3.3V (disponible sur les autres cartes) • Conçu pour les applications où l'espace est très petit • Conçu pour être intégré en permanence dans un projet. • Destiné à un public plus expérimenté.
Mega	• Nombreuses entrées et sorties pour raccorder des capteurs ou des actionneurs. • Capacité de mémoire suffisante pour les gros projets. • Plus de broches UART (4ports de communication série) • Plus de broches MLI(15sorties numériques peuvent être utilisées comme MLI. • Compatible avec la plupart des shields conçues pour Arduino	• Deux fois plus chère que l'Arduino Uno
Yun	• Module wifi préinstallé • Interface Ethernet préinstallé. • Nombreuses possibilités d'extension.	• Consomme plus d'énergie de sorte que le port USB 2.0 peut vite atteindre ses limites.

Tableau 2.3 *Avantages et Inconvénients de différentes cartes de type Arduino*

2.4 Eléments principaux d'une carte de développement Arduino

Une carte Arduino peut être classée en deux parties :

Carte Arduino	
Partie matériel	**Partie logiciel**

2.4.1 Partie Matériel

La carte de développement Arduino se compose de nombreux composants. Dans ce cas, nous allons nous appuyer sur le type de carte « Arduino UNO » pour montre les principaux blocs de composants qui aident à son fonctionnement[reff]:

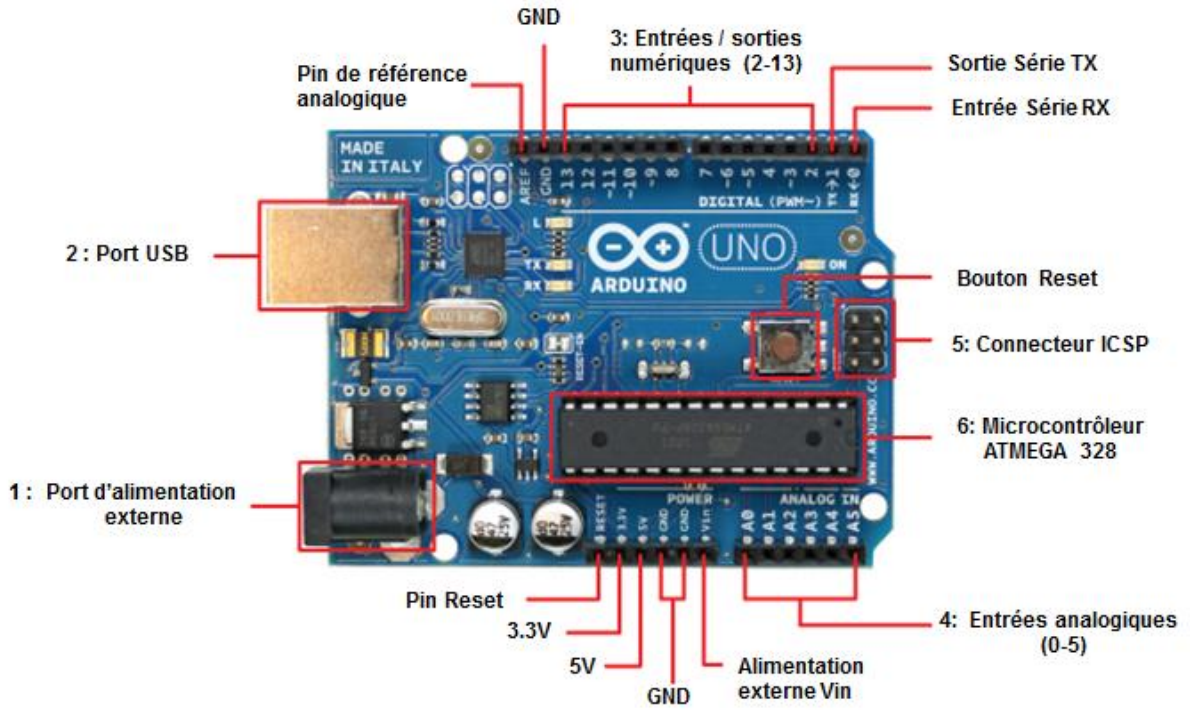


Figure 2.8 *Brochage d'une carte Arduino de type UNO* technologuepro.com

➢ **Description**

1 : Le port d'alimentation externe pour fonctionner, la carte a besoin d'une alimentation qui est conseillée en général entre 7 V et 12V pour garder une marge en basse tension et éviter que le circuit ne chauffe trop. Cette tension doit être continue et peut par exemple être fournie par une pile 9V.

2 : Le port USB Permet de communiquer avec la carte et de l'alimenter en 5V.

3: Les entrées/sorties numériques :

- 4 entrées/sorties numériques
- 6 (**broches 4, 5, 6, 7, 8, 9**) peuvent assurer une sortie PWM, et peuvent actionner de nombreux composants (LED, transistor, etc.) mais elles ne peuvent pas fournir beaucoup de courant (40 mA pour une carte Arduino UNO). Pour piloter des circuits de plus forte puissance, il faut passer par des transistors ou des relais.

4: Les entrées analogiques lui permettent de mesurer une tension variable (entre 0 et 5 V) qui peut provenir de capteurs ou d'interfaces divers (potentiomètres, etc).

5: Connecteur In-Circuit Serial Programming ICSP pour le téléchargement du programme.

6: Microcontrôleur ATMega328 est un microcontrôleur ATMEL de la famille AVR 8bits. Pour son brochage, voir la partie annexe.

2.4.2 Partie logiciel

Le code de programme écrit pour Arduino est connu sous le nom de croquis (sketch). Le logiciel utilisé pour développer de tels croquis pour un Arduino est communément appelé Arduino IDE. Cet IDE contient les parties suivantes [33]. L'ensemble de ces points seront traités et vus dans le chapitre 4 :

ARDUINO IDE	**Éditeur de texte**	C'est là que le code simplifié peut être écrit à l'aide d'une version simplifiée du langage de programmation C ++.
	Zone de message	Il affiche une erreur et donne également un retour sur l'enregistrement et l'exportation du code.
	Texte	La console affiche la sortie de texte par l'environnement Arduino, y compris les messages d'erreur complets et d'autres informations.
	Barre d'outils de la console	Cette barre d'outils contient divers boutons tels que Vérifier, Télécharger, Nouveau, Ouvrir, Enregistrer et Serial Monitor…

➢ **Principe d'utilisation d'une carte de développement ARDUINO**

Cette partie explore comment une carte ARDUINO peut être utilisée comme un outil pour les travaux d'étude et de recherche [35]. De ce fait, la communication entre le PC et la carte se fait via le port USB, moyennant installation d'un driver adapté (fourni par ARDUINO).

Figure 2.9 *Brochage d'une carte Arduino de type UNO* technologuepro.com

▪ **Les étapes de programmation ARDUINO**

Après la préparation du programme, il y'a des étapes à faire afin de terminer la programmation, qui sont les suivantes :

- ✓ Obtention d'une carte ARDUINO et un câble USB
- ✓ Téléchargement de l'environnement ARDUINO
- ✓ Raccordement de la carte ARDUINO au PC
- ✓ Installation des pilotes du périphérique Série-USB
- ✓ Lancement de l'application ARDUINO

▪ **Chargement du programme dans la carte**

Etape 1 : lancement du logiciel.

Etape 2 : ouverture et modification du programme.

Etape 4 : connexion de la carte au PC avec le cordon USB.

Etape 5 : transfert du programme vers la carte.

Etape 6 : vérification du fonctionnement.

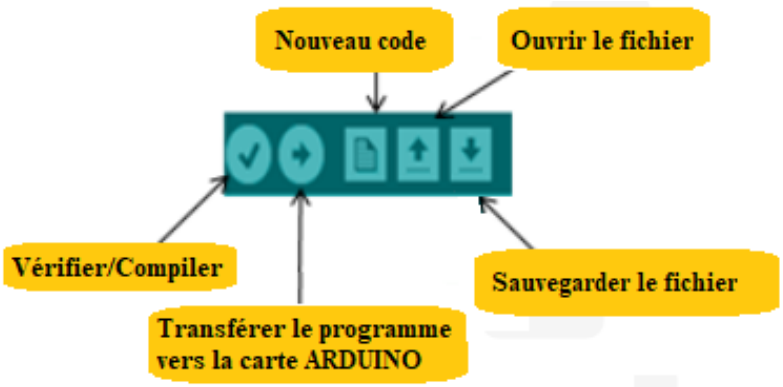


Figure 2. 10 *Application sur le programme de la carte*

- **Structure d'un projet ARDUINO**

Un programme ARDUINO est composé de 3 parties:

Partie1 : déclaration des variables (optionnelle)

Partie2 : représentée par **la Fonction setup ()**. C'est une partie initialisation et configuration des entrées / sorties. Elle est appelée une seule fois lorsque le programme commence.

Partie3 : représentée par la **Fonction loop ().** C'est la partie principale contenant le programme. Elle est répétée indéfiniment en boucle infinie

- **Notions du langage ARDUINO**

Structure

Fonctions de base

- void setup()
- void loop()

Structures de contrôle

- if
- if...else
- for
- switch case
- while
- do... while
- break
- continue
- return
- goto

Syntaxe de base

- ; (point virgule)
- {} (accolades)
- // (commentaire sur une ligne)
- /* */ (commentaire sur plusieurs lignes)
- #define
- #include

Opérateurs arithmétiques

- = (égalité)
- + (addition)
- - (soustraction)
- * (multiplication)
- / (division)
- % (modulo)

Opérateurs de comparaison

- == (égal à)
- != (différent de)
- < (inférieur à)
- > (supérieur à)
- <= (inférieur ou égal à)
- >= (supérieur ou égal à)

Opérateurs booléens

- && (ET booléen)
- || (OU booléen)
- ! (NON booléen)

Opérateurs composés

- ++ (incrémentation)
- -- (décrémentation) (à revoir)
- += (addition composée)
- -= (soustraction composée)
- *= (multiplication composée)
- /= (division composée)
- &= (ET bit à bit composé)
- |= (OU bit à bit composé)

Opérateurs bit à bit

- & (ET bit à bit)
- | (OU bit à bit)
- ^ (OU EXCLUSIF bit à bit)
- ~ (NON bit à bit)
- << (décalage à gauche)
- >> (décalage à droite)

Pointeurs

- * pointeur
- & pointeur

Fonctions

Entrées/Sorties Numériques

- pinMode(broche, mode)
- digitalWrite(broche, valeur)
- int digitalRead(broche)

Entrées analogiques

- int analogRead(broche)
- analogReference(type)

Sorties "analogiques" (génération d'impulsion)

- analogWrite(broche, valeur) - PWM

Entrées/Sorties Avancées

- tone()
- noTone()
- shiftOut(broche, BrocheHorloge, OrdreBit, valeur)
- unsigned long pulseIn(broche, valeur)

Communication

- Serial

Temps

- unsigned long millis()
- unsigned long micros()
- delay(ms)
- delayMicroseconds(us)

Math

- min(x, y)
- max(x, y)
- abs(x)
- constrain(x, a, b)
- map(valeur, toLow, fromHigh, toLow, toHigh)
- pow(base, exposant)
- sq(x)
- sqrt(x)

Pour davantage de fonctions mathématiques, voir aussi la librairie math.h : log, log10, asin, atan, acos, etc...

Nombres randomisés (hasard)

- randomSeed(seed)
- long random(max)
- long random(min, max)

Trigonométrie

- sin(rad)
- cos(rad)
- tan(rad)

Bits et Octets

- lowByte()
- highByte()
- bitRead()
- bitWrite()
- bitSet()
- bitClear()
- bit()

Interruptions Externes

- attachInterrupt(interrupti on, fonction, mode)
- detachInterrupt(interrupt ion) Interruptions
- interrupts()
- noInterrupts()

Variables et constantes

Types des données

Les variables peuvent être de type variés qui sont décrits cidessous.

Synthèse des types de données Arduino

- boolean
- char
- byte
- int
- unsigned int
- long
- unsigned long
- float (nombres à virgules)
- double (nombres à virgules)
- Les chaînes de caractères
- objet String NEW
- Les tableaux de variables
- le mot-clé void (fonctions)
- word
- PROGMEM

Conversion des types de données

- char()
- byte()
- int()
- long()
- float()
- word()

Portée des variables et qualificateurs

- Portée des variables
- static
- volatile
- const

Utilitaires

- sizeof() (opérateur sizeof)

Référence

- Code ASCII

2.5 Automate programmable industriel API

Les API ou PLC sont particulièrement appréciés dans le secteur industriel pour leur robustesse, leur réactivité et leur simplicité en ce qui concerne la maintenance [36].

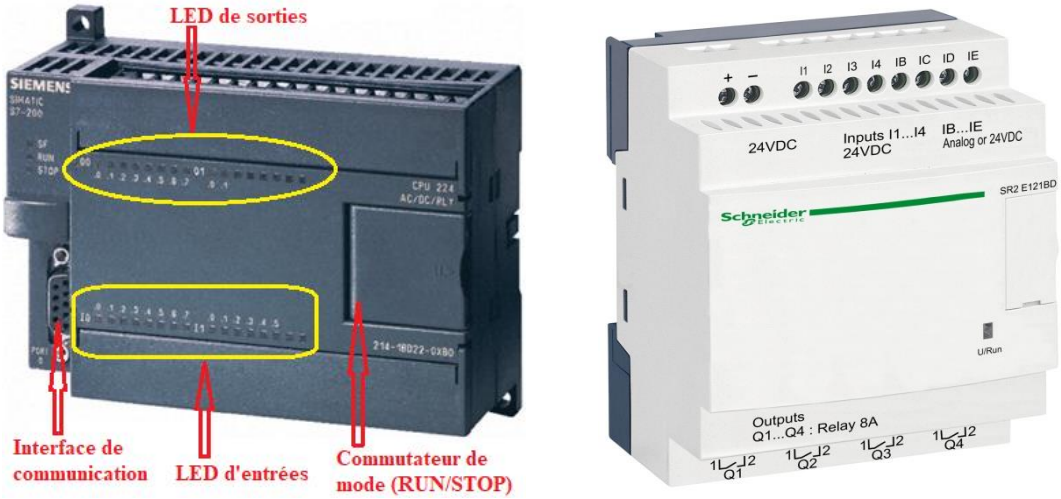
L'API a été conçu pour remplacer la logique câblée par une autre dite programmée. Il existe différents types d'automates distingués principalement par leur forme, leur taille, leur apparence, leur mode d'alimentation, leur puissance, leur langage de programmation [36].

2.5.1 Types d'APIs

Les automates programmables industriels sont construits à la base de différentes structures : API de type compact et API de type modulaire

- **API compact**

Ces automates de fonctionnement simple, sont généralement destinés à la commande de petits automatismes. Dans notre cas d'étude, nous avons considéré l'API de type siemens.



(a) (b)

Figure 2.11 *API de type compacte*

(a) Siemens (b) *Schneider*

- **API modulaire**

Ces automates sont intégrés dans les automatismes complexes où la puissance, capacité de traitement et flexibilité sont nécessaires. Dans ce cas, le processeur, l'alimentation et les

interfaces d'entrées / sorties résident dans des unités séparées (modules) et sont fixées sur un ou plusieurs racks contenant le "fond de panier" (bus plus connecteurs) [37].

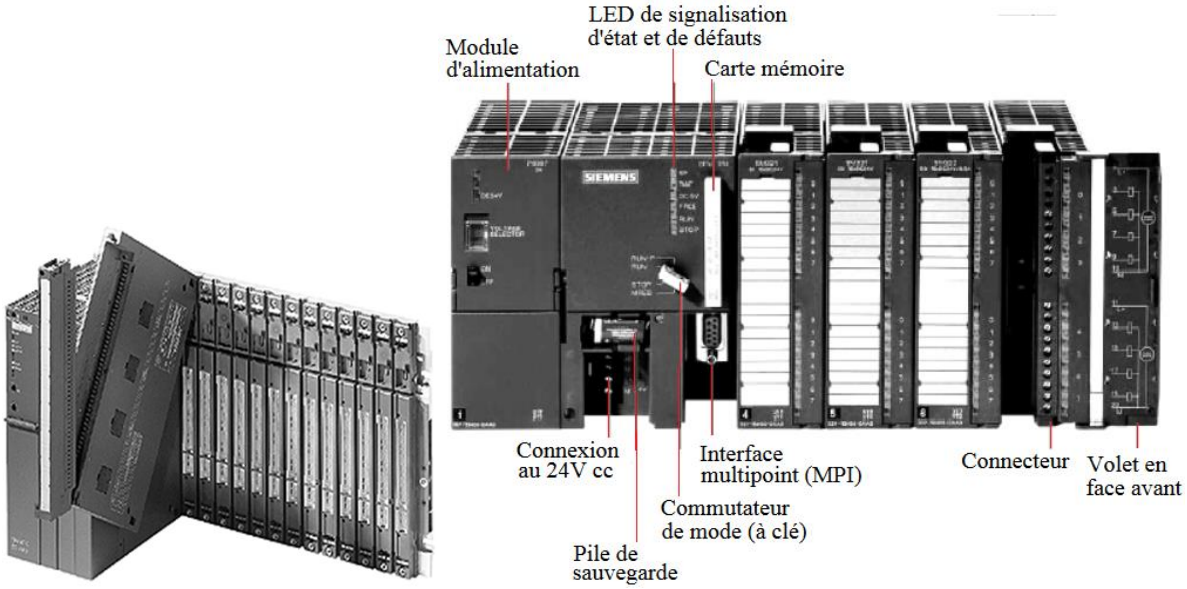


Figure 2.12 *Automate modulaire*

2.5.2 Les caractéristiques principales d'un API [38]

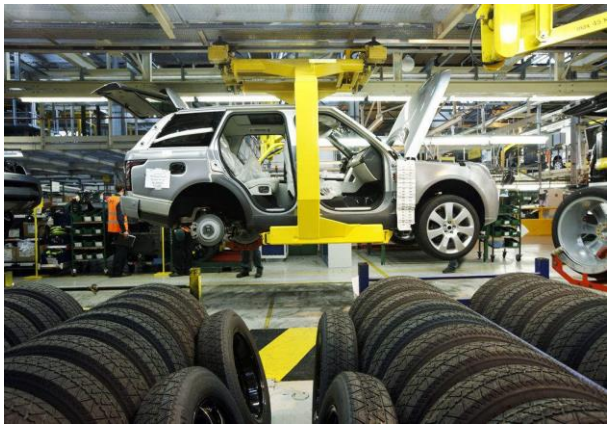
- ✓ Compact ou modulaire.
- ✓ Tension d'alimentation.
- ✓ Nombres d'entrées/sorties intégrés.
- ✓ Taille mémoire.
- ✓ Sauvegarde (EPROM, EEPROM, pile, ...).
- ✓ Modules complémentaires (analogique, Communication).
- ✓ Nombre de compteurs et de temporisateurs.
- ✓ Langage de programmation.

2.5.3 Domaines d'emploi des automates

On utilise les API dans tous les secteurs industriels pour la commande des machines. Les cas de figures donnés ci-dessous, illustrent quelques exemples sur l'usage des APIs.

(a)

(b)



(c)

(d)

(e)

(f)

Figure 2.13 *Quelques exemples de domaines ou l'API est appliqué*
(a) *Convoyage* (b) *Emballage* (c) *Chaines de production automobile*
(d) *Chaines de production agroalimentaire* (e) *Métallurgie* (f) *Parking*

L'API est de plus en plus utilisé dans le domaine du bâtiment (tertiaire et industriel) pour le contrôle du chauffage, de l'éclairage, de la sécurité ou des alarmes. De plus, l'API

convient parfaitement pour des systèmes de sécurité ferroviaire, des machineries d'ascenseur,…

Ce chapitre présente une large formation sur des deux APIs utiles dans l'industrie algérien qui sont les automates de type Schneider et de type Siemens.

2.5.4 Critères du choix d'un API [39]

Pour choisir un PLC adapté aux besoins d'une fonction d'usage, il sera nécessaire de considérer un certain nombre de critères importants:

- ✓ Le nombre et la nature d'entrées / sorties;
- ✓ Le temps de réponse (performance de processeur);
- ✓ Les modules complémentaires (analogique, communication...);
- ✓ La nature du traitement (temporisation, comptage, ...) ;
- ✓ La communication avec les autres systèmes ;
- ✓ Les moyens de sauvegarde du programme (capacité de la mémoire) ;
- ✓ La fiabilité, robustesse, immunité aux parasites ;
- ✓ Les moyens de dialogue et le langage de programmation ;
- ✓ La documentation.

2.5.5 Langages de programmation des API ou PLC

La norme industrielle CEI 61131-3 définit plusieurs langages de programmation pour les automates programmables industriels [40]:

- ➢ **LD (Ladder Diagram) - Le langage à contacts**

Il se base sur une approche visuelle évoquant des schémas électriques. Nous tenons à souligner que ce type de langage fait l'objet de notre travail.

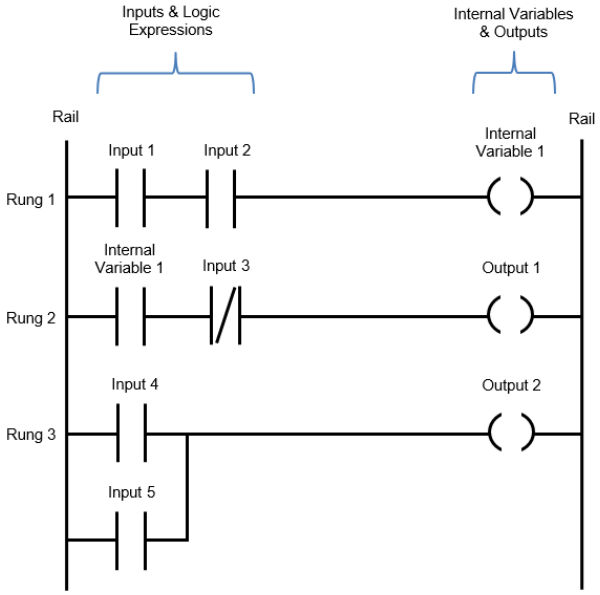


Figure 2.14 *Le langage à contacts*

- **IL (Instruction List) - Les listes d'instructions**

  Ce langage est très proche du langage informatique dit assembleur.

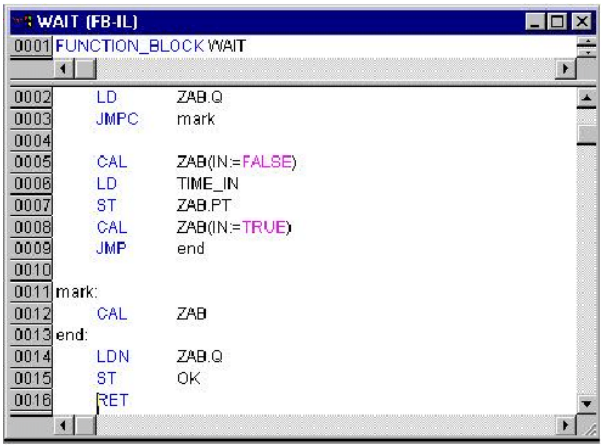


Figure 2.15 *Les listes d'instructions*

- **FBD (Function Block Diagram)** - Les diagrammes de schémas fonctionnels. C'est un langage graphique qui permet la construction d'équations complexes.

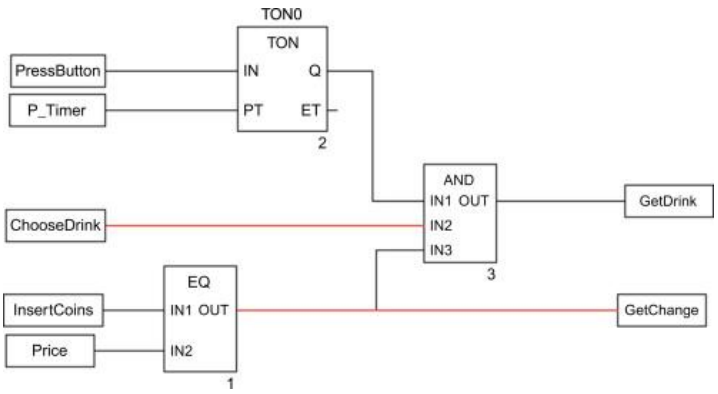


Figure 2.16 *Diagramme de blocs fonctionnels*

- **ST (Structured Text) - Le texte structuré**

Il s'agit d'un langage textuel de haut niveau qui est utilisé pour décrire des procédures complexes.

```
 x : BOOL;

 P_STEP : INT;
END_VAR

P_STEP := 3;

CASE PROGRAM_STEP OF
 1: P_STEP := P_STEP+1;
 2: P_STEP := P_STEP+2;
 3: P_STEP := P_STEP+3;
ELSE
 PROGRAM_STEP := PROGRAM_STEP+10;
END_CASE;

LIMIT_SWITCH1 := TRUE;
LIMIT_SWITCH2 := FALSE;

IF LIMIT_SWITCH1 OR LIMIT_SWITCH2 THEN
 OUTPUT5 := FALSE;
 P_TRIGGER := TRUE;
END_IF;
```

Figure 2.16 *Le texte structuré*

- **SFC (Sequential Function Charts) - Les graphes de fonction séquentielle**

Ce langage est issu du langage GRAFCET.

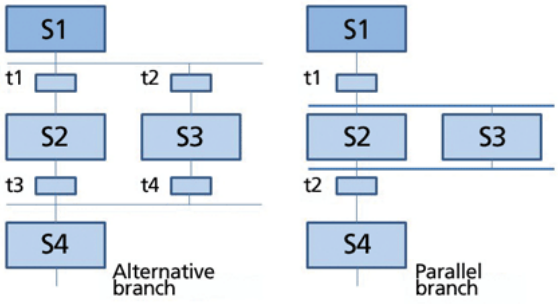


Figure 2.17 *graphes de fonction séquentielle*

2.5.6 Avantages des APIs

L'automate programmable se distingue de l'ordinateur sur un certain nombre de points [41] :

- ✓ Sa mémoire est programmable par un non informaticien, par le recours à un langage adapté (pas de langage de programmation complexe).
- ✓ Il dispose d'entrées et de sorties industrielles qui lui permettent d'être directement connecté aux capteurs.
- ✓ Il est conçu de façon à pouvoir fonctionner dans des conditions difficiles en termes de températures, poussière, humidité, vibrations, micro-coupures de courant,…

2.6 Conclusion

Dans le présent chapitre, nous avons vu un ensemble de cartes à microcontrôleur pouvant être appliqué dans les systèmes embarqués. A travers ces cartes, nous avons opté pour le choix d'utiliser la carte ARDUINO MEGA et ce, pour un fonctionnement approprié non rencontré avec un système à base de composants électroniques uniquement.

Par conséquent, le chapitre suivant sera consacré à l'étude et la conception du système proposée dans ce travail pour le suivi d'une ligne bien déterminée. Le système à carte Arduino sera comparé à celui non concerné par la carte à travers différentes circuiteries afin de monter la nécessité d'utilisation des circuits programmés.

Chapitre3 :

Etude et conception du système de parking intelligent

3.1. Introduction

Dans ce chapitre nous allons étudier notre système du parking intelligent, dont nous allons considérer les cas de situation avec deux cartes ARDUINO (Méga et Uno). Ainsi, nous avons déjà vu que le type de carte Méga peut être exploité dans le cas où les systèmes utilisés préconisent plus de sorties comparativement à une carte Arduino de type UNO dont ses caractéristiques portent plus d'intérêt à notre deuxième système étudié dans ce chapitre.

3.2 Cas du premier système de parking

3.2.1 Schéma de principe

La figure 3.1 montre les éléments principaux du système qui concerne mon étude. Ainsi, différents points sont décrits comme suit :

1 : Lorsqu'un véhicule désire entrer, l'ouverture de la barrière se produit automatiquement grâce au capteur qui détecte la présence du véhicule et envoient l'ordre d'ouvrir la lisse, mais dans le cas où le nombre de places est disponible. Dans le cas contraire, la barrière reste toujours fermée.

2 : lorsqu'un véhicule désire sortir du parking, il lui suffit de s'avancer vers la deuxième barrière de sortie pour que celle-ci s'ouvre automatiquement.

3 : l'affichage externe permet d'indiquer le nombre total de places disponibles au niveau du parking souterrain et celui de chacun de ses blocs. En effet, quand quelqu'un libère une place, par exemple, et quitte complètement le parking, l'affichage se met à jour automatiquement.

4 : toutes les places à l'intérieur du parking sont munies d'un capteur.

5 : l'affichage interne permet d'afficher le nombre de places restantes au bout de chaque bloc.

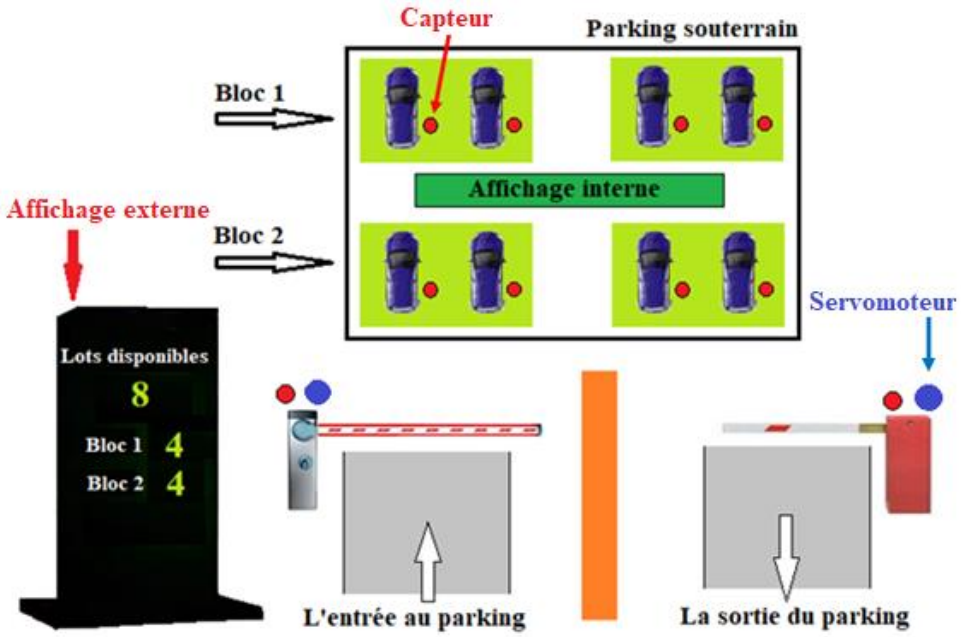


Figure 3.1 *Principe du premier système de parking à étudier*

3.2.2 Etude des circuits

Il s'agit ici d'étudier les différents circuits électroniques avec carte Arduino qui entrent dans la conception du notre système.

3.2.2.1 Schéma synoptique

Le procédé utilisé pour cette étude repose sur la différence d'éclairement incident sur les capteurs de lumière photorésistance ou LDR (Light-Dépendent-Résistor). Cela permet :

- La détection de présence d'un véhicule, à l'entrée, à la sortie comme à l'emplacement du stationnement à l'intérieur du parking.
- La commande des servomoteurs pour l'ouverture des barrières.
- Lacommande des afficheurs LCD pour l'affichage du nombre de placesdisponibles au niveau du parking.
- L'allumage des LEDs pour représenter un véhicule. Dans ce cas, deux types de LEDs sont concernés :

  - **LED rouge allumée** : voiture stationnée.
  - **LED verte allumée** : la place vide.

La figure 3.2 illustre l'ensemble des entrées et des sorties au circuit de commande représenter par la carte à microcontrôleur.

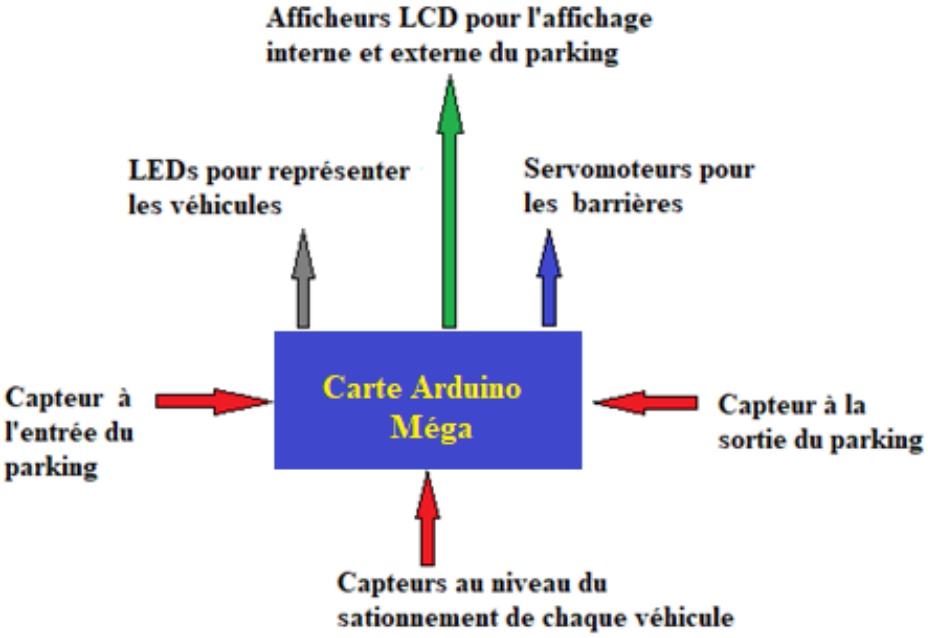


Figure 3.2 *Schéma synoptique général*

3.2.2.2 Analyse des circuits

A- <u>Parie 1 : circuit de commande</u>

La figure 3.3 illustre le bloc de commande en fonction de l'ensemble des circuits qui devraient être gérés à ses entrées et à ses sorties. Donc, nous avons :

- ✓ deuxLDR qui sont utilisées comme des capteurs de détection de présence d'un véhicule, soit pour son entrée comme pour sa sortie.
- ✓ Deux paires de LED, dont chacune est formée de deux LED (rouge et verte) pour représenter le véhicule.
- ✓ Huit paires de LED, chacune peut représenter le véhicule à l'emplacement du stationnement au niveau du parking.
- ✓ Huit LDR, chacune est utilisée pour détecter la présence d'un véhicule à l'emplacement du stationnement au niveau du parking.
- ✓ Deux afficheurs de type LCD pour l'affichage interne et externe du parking. Ce qui permet l'affichage du nombre total des places disponibles et du nombre de chaque bloc de stationnement au niveau du parking.

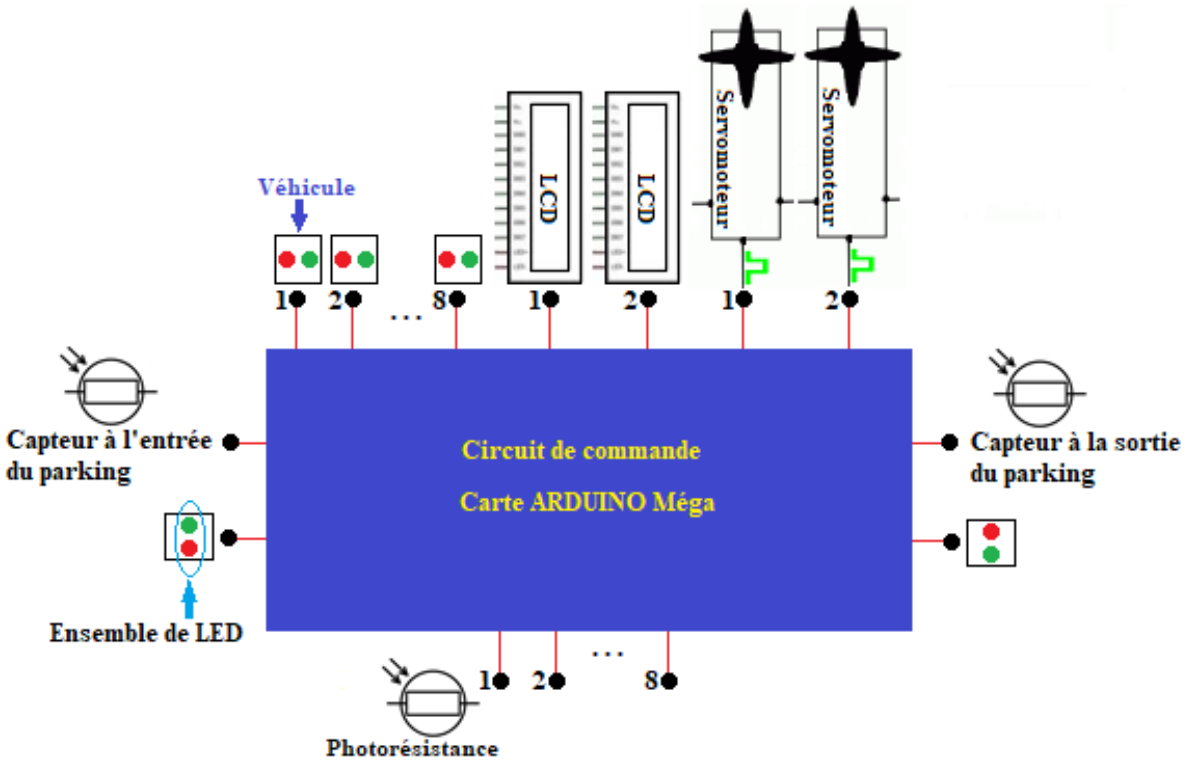


Figure 3.3 *Système en fonction des entrées et de sorties envisagées*

B- Parie 2 : circuits d'entrées

➢ A l'entrée au parking

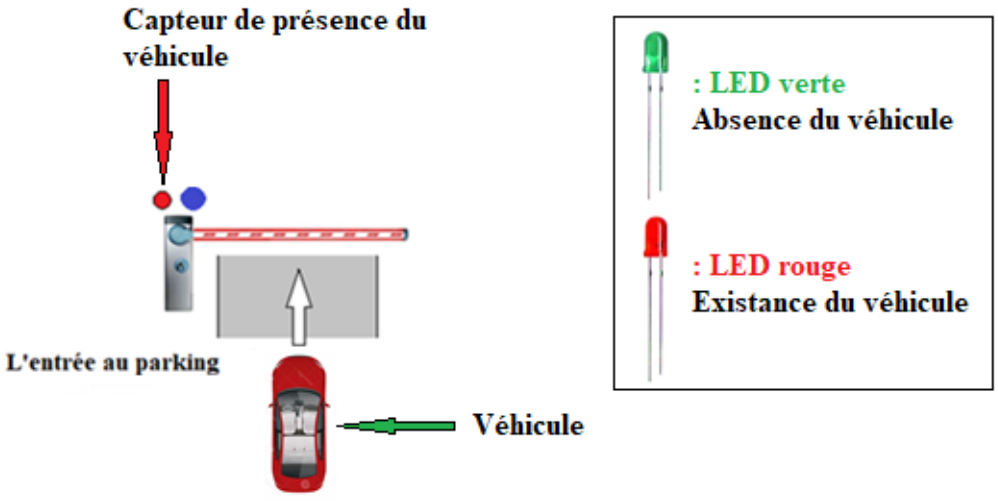


Figure 3.4 *Schéma de principe. L'entrée au parking*

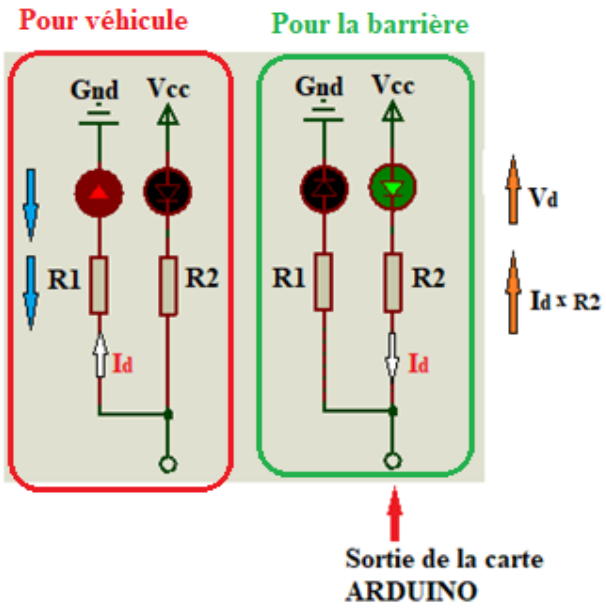


Figure 3.5 *Circuit électronique. L'entrée au parking*

- **Calcul des résistances R1 et R2**

Dans ce cas, la valeur de chaque résistance est en fonction du courant consommé par la LED correspondante.

Les caractéristiques des LEDs rouge et verte sont comme suit:

LED rouge :

La plage d'alimentation : 2,1V-2,3V

Le courant nominal : 0.02 A

LED verte :

La plage d'alimentation : 3,1V-3,3V

Le courant nominal : 0.02 A

Connaissant les caractéristiques ces caractéristiques, on peut déterminer les valeurs des résistances. Donc, en ce qui concerne la résistance R1, si la sortie de l'ARDUINO fournit un niveau haut à sa sortie (une tension de 5V), on peut procéder comme suit :

$Vs(ARDUINO) = Vd + R1x\ Id$ (1)

Ainsi, selon l'équation (1), et en prenant Vd égale à 2.2V, la valeur calculée de R1 est 140Ω. Dans ce cas, on peut prendre R1=150Ω.

De même, la valeur calculée de R2 est 100Ω.

- **Le choix du capteur**

Dans ce cas, le capteur utilisé est de type LDR, figure 3.6. C'est un composant dont la valeur en ohms dépend de la lumière à laquelle il est exposé. La principale utilisation de la photorésistance est la mesure de l'intensité lumineuse. Nous avons utilisé ce type de capteurs à cause de son temps de réponse qui est court **[34]**. Le tableau 3.1 montre les caractéristiques d'un modèle type de photorésisance LDR.

(a) (b)

Figure 3. 6 *Capteur LDR.*
(a) *Composant réel* (b) *de simulation*

Par exemple, la photorésistance de type PGM5506 est caractérisée par :

Model	Vmax (VDC)	Pmax (mW)	Ambient Temp (°C)	Spectral Peak (nm)	Photo Resistance (10Lx) (KΩ)	Dark Resistance (MΩ)min	γ min	ResponseTime (ms)	
								Rise	Decay
PGM5506	100	90	-30 ~ +70	540	2 ~ 6	0.15	0.6	30	40

Tableau 3.1 *Tableau des caractéristiques de la photorésistance.*

- **L'utilisation du capteur LDR avec la carte ARDUINO**

Dans ce cas, le capteur ne peut être connecté directement, une résistance R est branchée dans le circuit, figure 3.7, ce qui donne un diviseur de tension. Donc, la tension de sortie Vout est donnée comme suit :

$$Vout = Vcc\frac{R}{(R+R(LDR))} \quad (2)$$

De l'équation (2), on peut déduire l'équation de la résistance R :

$$R = \frac{Vout}{Vcc-Vout} R(LDR)$$

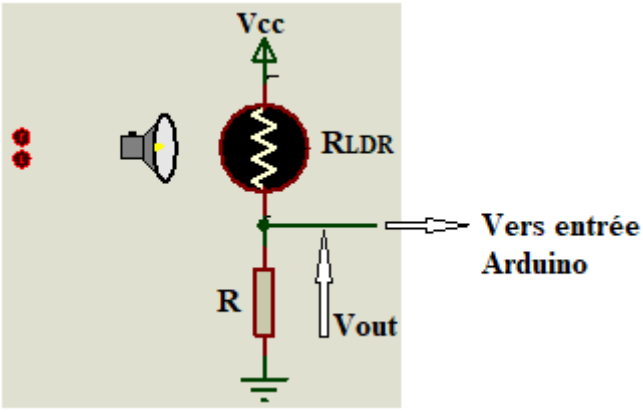


Figure 3.7 *Circuit à base de LDR pour le calcul de la résistance R*

Généralement, la valeur de R(LDR) est donnée suivant les cas suivant :

R(LDR) 10 fois plus lumineux ➔Vout =4.54 Volts, ce qui correspond à

10kΩ ≤ R(LDR)≤60kΩ

R(LDR) 10 fois moins lumineux ➔Vout =0.54 Volts, ce qui correspond à

R(LDR)≥ 0.1 MΩ

- Calcul de la résistance R du circuit :

Pour la valeur de la tension Vcc = 5 volts, on peut avoirR = $\frac{4.5}{5-4.5}\times$ 10kΩ

➔R = 90kΩ, donc la résistance R peut prendre 100kΩ

- **<u>Au niveau des emplacements de stationnement du parking</u>**

Comme nous avons déjà mentionné que le lieu de stationnement du parking est scindé en deux blocs, chacun est caractérisé par quatre lots ou emplacement pour le stationnement des véhicules, figure 3.8.

Un emplacement « E » représente l'ensemble formé par :

✓ Un capteur LDR permettant la détection de la présence d'un véhicule à l'emplacement du stationnement.

✓ Une paire de LED représentant ainsi le véhicule. Ce qui indique le stationnement ou non du véhicule au niveau d'un bloc du parking.

✓ Donc, nous avons huit capteurs LDR,puisqu'il s'agit de huit emplacements de stationnement, et 16 LED, puisque nous avons huit paires de LED

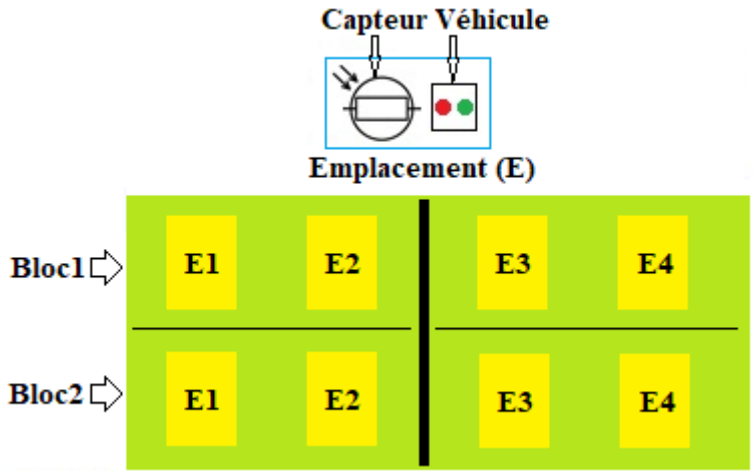


Figure 3.8 *Circuit à base de LDR pour le calcul de la résistance R*

La figure 3.9 montre le schéma électrique d'un emplacement. Nous constatons la même représentation des composants comme celle étudiée plus haut

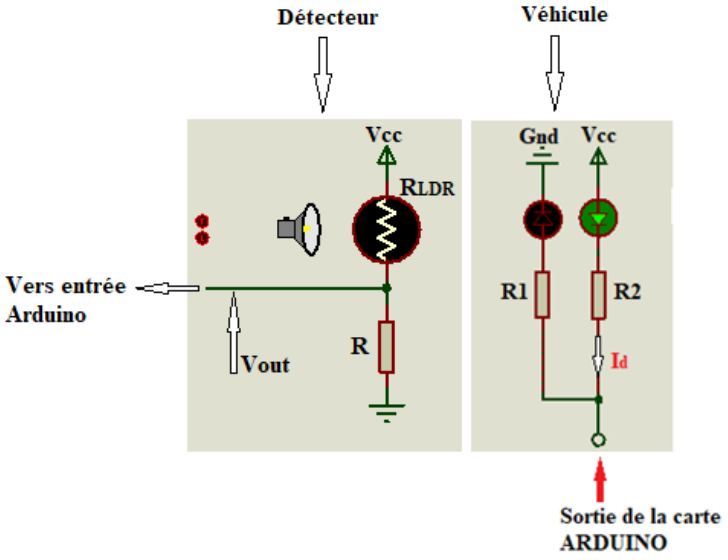


Figure 3.9 *Circuit à base de LDR pour le calcul de la résistance R*

C- Parie 3 : circuits de sorties

Dans cette partie, nous allons nous intéresser aux différents circuits de sorties qui devraient être commandés par la carte Arduino. Il s'agit de :

- ✓ deux servomoteurs
- ✓ deux afficheurs de type LCD :

1. l'afficheur principal (extérieur) LCD1 pour afficher le nombre total de places disponibles dans le parking et le nombre de places disponibles dans chaque bloc, puisqu'il s'agit de deux blocs de stationnement du parking.

2. L'afficheur intérieur LCD2 pour afficher le nombre de places vides des deux Blocs.

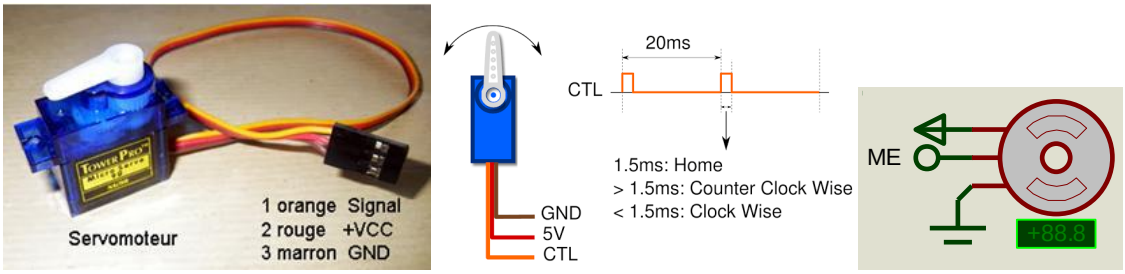
Donc, tout changement d'affichage est dû à un changement sur des capteurs qui sont branchés à la carte ARDUINO, ce qui va manipuler toute action du montage via un code bien défini.

✓ Des LED. Ces dernières sont déjà vues et étudiées dans la partie des circuits d'entrées, qui dans sa description, nous avons eu recours à leurs utilisations(LED).

➢ **Servomoteur**

Dans notre travail, nous avons besoin que le mouvement des moteurs soit précis. Dans cas nous optons pour le servomoteur, du fait que c'est est un moteur assez commun utilisé dans les projets d'ingénierie. De plus, la raison en est que nous pouvons déplacer le servomoteur à n'importe quel angle souhaité, ce qui n'est pas possible dans le cas du moteur à courant continu utilisé dans le cas des barrières. Donc, nous allons contrôler les servomoteurs avec Arduino.

La figure 3.10 montre la photo d'un type de servomoteur utilisé dans des projets de réalisation.Pour sa connexion, le servomoteur est équipé d'une prise de type Graupner à 3 fils.



(a) (b) (c)

Figure 3.10 *Servomoteur de type*
(a) *Exemple de servomoteur* (b)*Brochage d'un servomoteur*
(c) *Servomoteur Type de LDC :LM016L sous Proteus pour simulation*

➢ **Afficheur LCD**

Les afficheurs à cristaux liquides LCD sont des modules compacts et nécessitent peu de composants externes pour un bon fonctionnement. Ils consomment relativement peu de 1 mA à 5 mA, et ils nécessitent une alimentation de 5 V pour pouvoir alimenter leur pilote interne et ainsi permet l'affichage des caractères sur l'écran.

L'afficheur LCD que nous avons utilisé permet d'afficher des caractères ASCII sur 16 colonnes et de 2 lignes, figure 3.11.

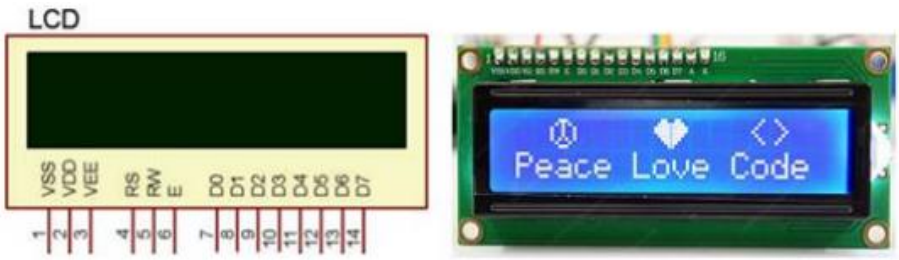


Figure 3.11 *Afficheur LCD à logique intégré 02 lignes x 16 colonnes*

Les différentes broches externes de l'afficheur LCD sont les suivantes :

- VCC, masse : Alimentation de l'afficheur LCD avec 5 V, 0V respectivement.
- Contraste : Entrée permettant de régler le contraste de l'afficheur LCD. Il faut appliquer une tension continue réglable entre 0 V et 5 V à l'aide d'un potentiomètre.

- VLED : Différence de potentiel permettant de commander le rétro éclairage.
- E (ENABLE) : Entrée de validation, elle permet de valider les données sur un front descendant.
- RS (Register Select) : Cette entrée permet d'indiquer à l'afficheur si l'on souhaite réaliser une commande (RS=0) par des instructions spécifiques ou écrire une donnée sur le bus (RS=1).
- R/W : Entrée de lecture (R/W=1) et d'écriture (R/W=0). Lorsqu'on commande l'afficheur LCD, il faut se placer en mode écriture.
- D7…D0 : Bus de données bidirectionnel, il permet de transférer les instructions ou les données à l'afficheur LCD. Puisque l'afficheur LCD sera commandé par un Arduino. Il faut donc penser aux mises en œuvre :

Matériel : *Connexion des broches d'Arduino à l'afficheur LCD.*

Logiciel : *Utilisation de sous programmes permet de commander l'afficheur LCD (initialisation, effacement de l'afficheur, affichage d'un caractère, affichage d'une variable…).*

3.2.3 Simulations des circuits

Ce chapitre sera consacré aux différents tests de simulation de différents circuits électroniques étudiés de notre système dans le troisième chapitre. Donc, différents organigrammes correspondant aux deux situations, sans et avec carte Arduino seront établis. De plus, les programmes concernés seront présentés, ce qui permettra de constater leurs différences.

3.2.3.1 Partie logicielle

Cette partie est dédiée à la représentation des informatiques utilisées dans le développement de notre projet. Ainsi, nous allons exploiter deux plateformes principales :

A- Plateforme de programmation Arduino

Dans ce cas, il y a une interface de l'IDEARDUINO. Ce qui nous offre la possibilité pour développer nos programmes sur les cartes Arduino. La figure IV. 1 montre la forme de la structure de programmes ARDUINO.

Nous remarquons deux fonctions principales que nous devions respecter pour l'écriture et l'exécution de nos programmes :

- ✓ **Setup** : contiendra toutes les opérations nécessaires à la configuration de la carte (directions des entrées sorties, débits de communications série, etc.).
- ✓ **Loop** : elle est exécutée en boucle après l'exécution de la fonction setup. Elle continuera de boucler tant que la carte n'est pas mise hors tension. Cette boucle est absolument nécessaire sur les microcontrôleurs étant donné qu'ils n'ont pas de système d'exploitation. En effet, si l'on omettait cette boucle, à la fin du code produit, il sera impossible de reprendre la main sur la carte Arduino qui exécuterait alors du code aléatoire.

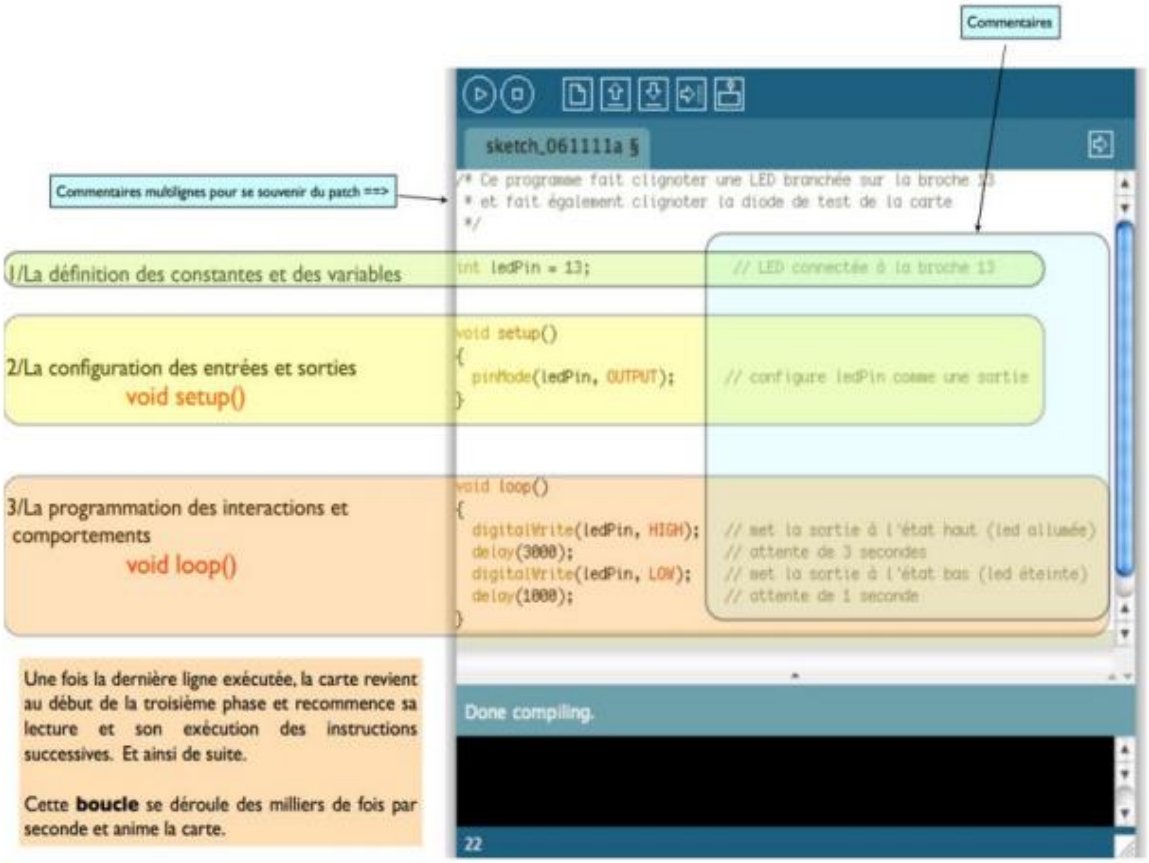


Figure 3.12 *Structure de programme ARDUINO*

B- Plateforme de développement et de simulation Proteus

Avant de passer à toute réalisation pratique d'un système donné, il est recommandé de passer à la l'établissement des différentes parties électroniques du système, pour cela on utilise le logiciel Proteus. Ce qui permettra pour notre cas, d'ajuster, de modifier le circuit et de vérifier les résultats obtenus correspondant aux différentes fonctions du système de parking.

Figure 3.13 *Représentation par une photo du logiciel Proteus*

Deux logiciels principaux composent cette suite logicielle: ISIS, ARES :

ISIS : pour éditer nos schémas électriques.

ARES : pour réaliser le PCB (**de** l'anglais printed circuit board ou circuit imprimé) de la carte électronique.

3.2.3.2 Organigrammes fonctionnels

A- Organigramme d'entrée

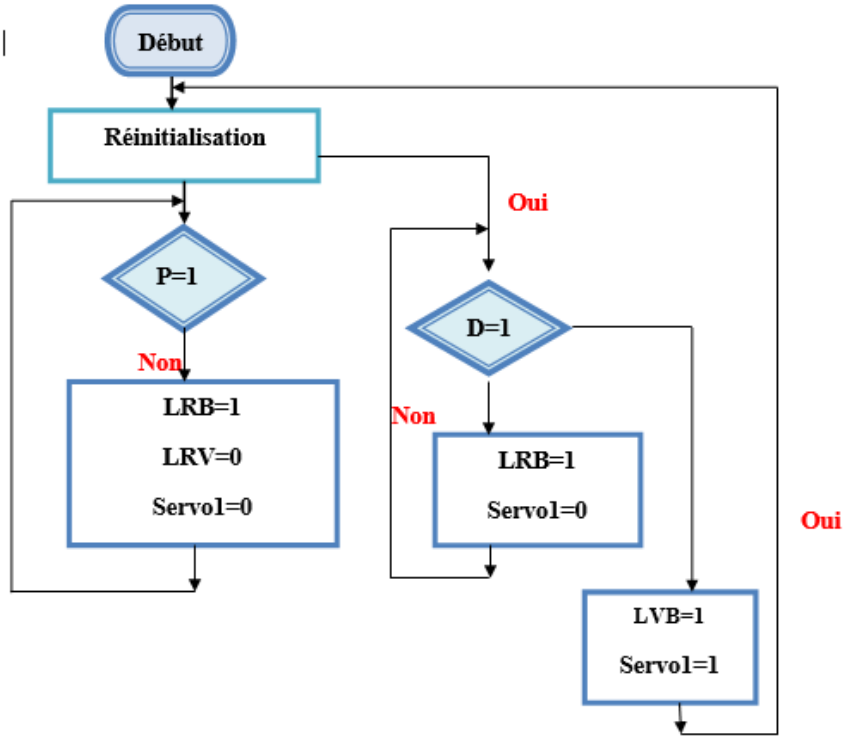


Figure 3.14 *Organigramme d'entrée*

P=1 présence de voiture
Absence de voiture

D=place disponible
LRB=1 LED rouge barrière allumée
LVB=1 LED verte barrière allumée
Servo 1=1 servo moteur 1 en marche
LRV =1 LED rouge voiture allumée

- **Circuit électronique :**

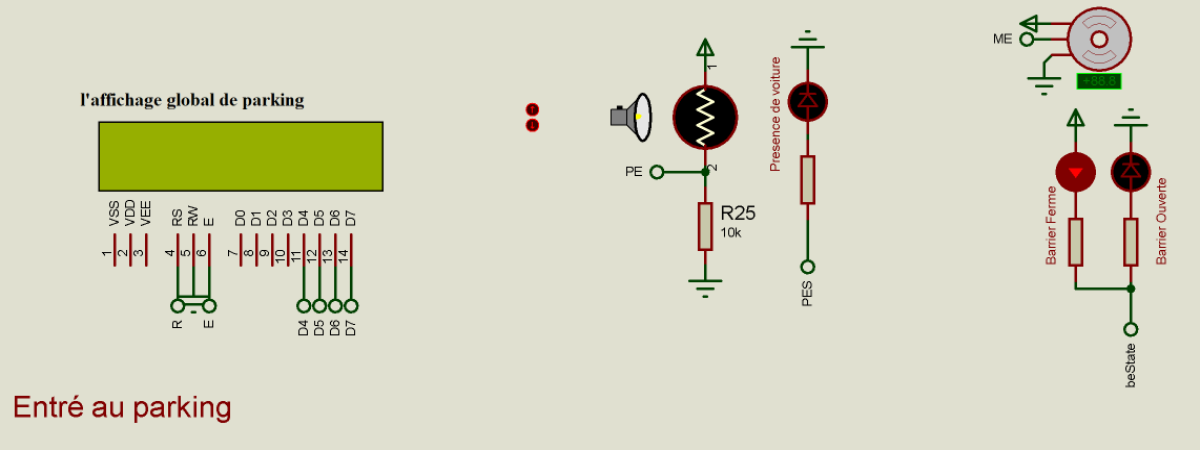


Figure 3.15 *Circuit électronique d'entrée*

B- Organigramme de stationnement :

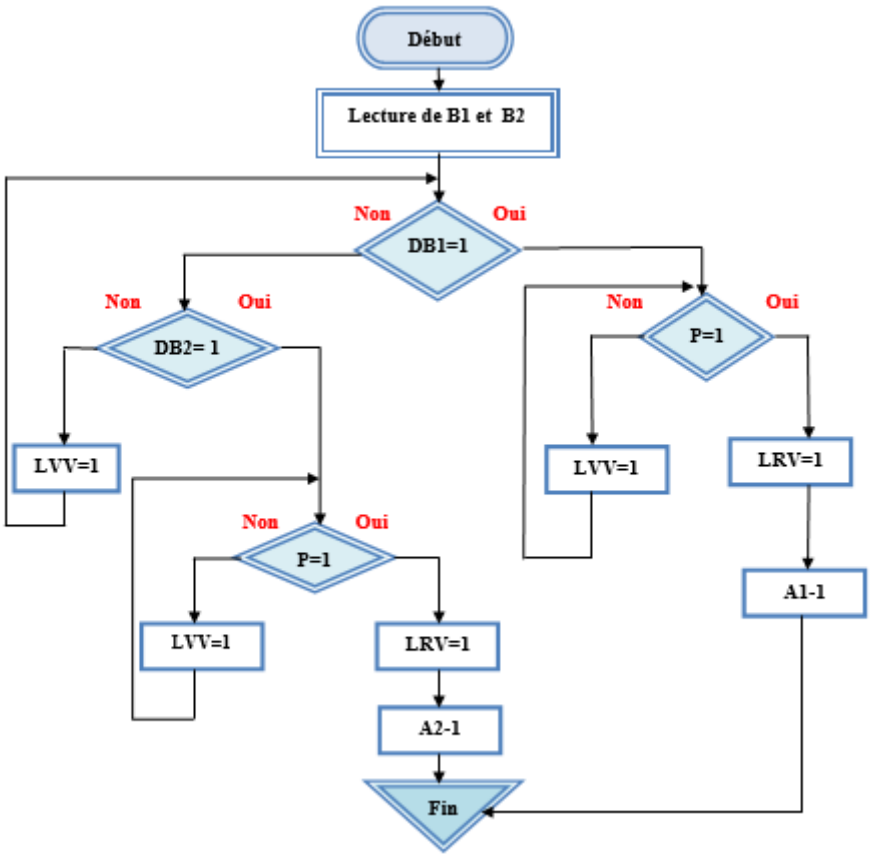
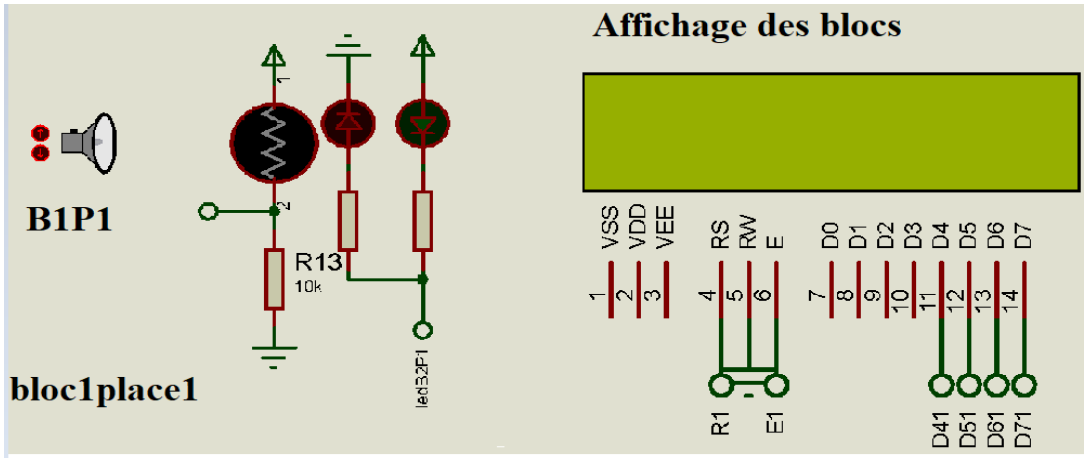


Figure 3.16 *Organigramme de stationnement*

B : bloc
DB : disponible de place au niveau d'un bloc
A : affichage
P : présence
LVV :led verte voiture allumée
LRV :led rouge voiture allumée

- **Circuit électronique :**



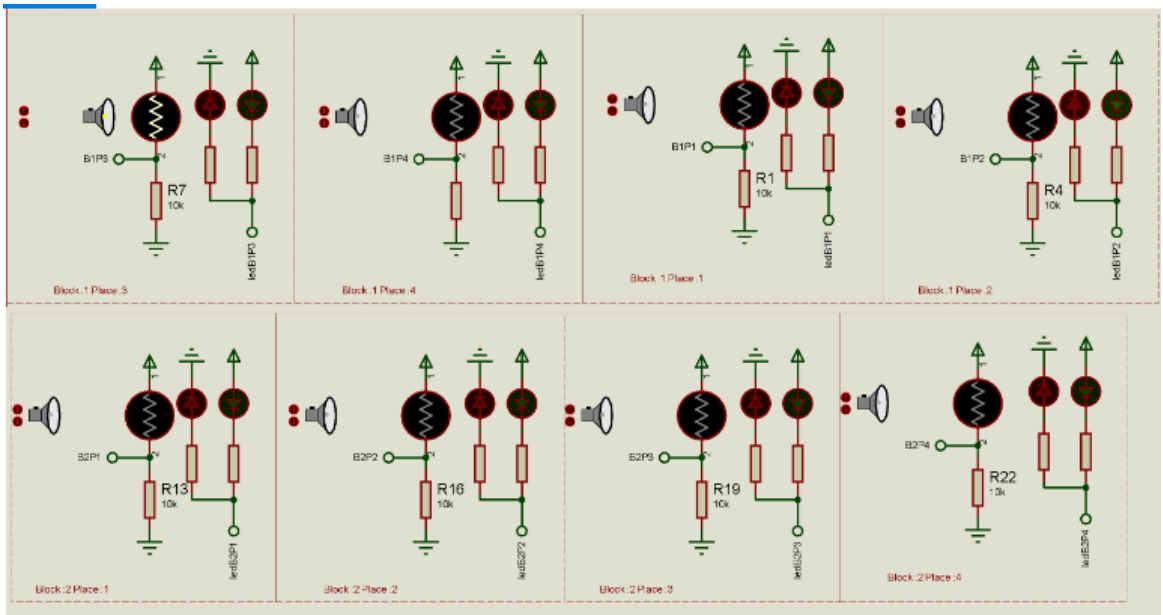


Figure 3.17 *Circuit électronique de stationnement*

C- Organigramme de sortie :

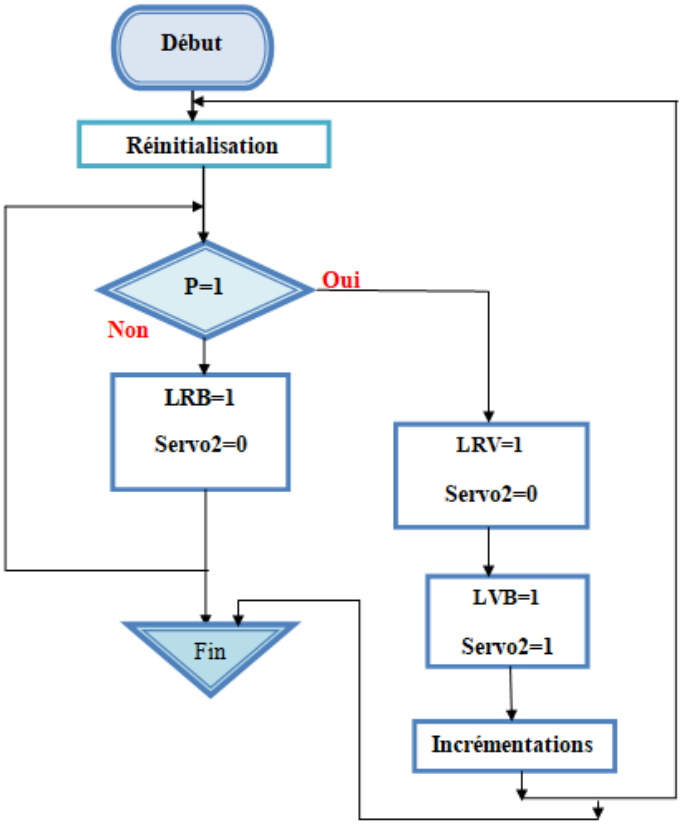


Figure 3.18 *Organigramme de sortie*

P=1 présence de voiture
LRB=1 LED rouge barrière allumée
LVB=1 LED verte barrière allumée
Servo 2=1 servo moteur 2 en marche
LRV =1 LED rouge voiture allumée (présence de voiture)

- **Circuit électronique :**

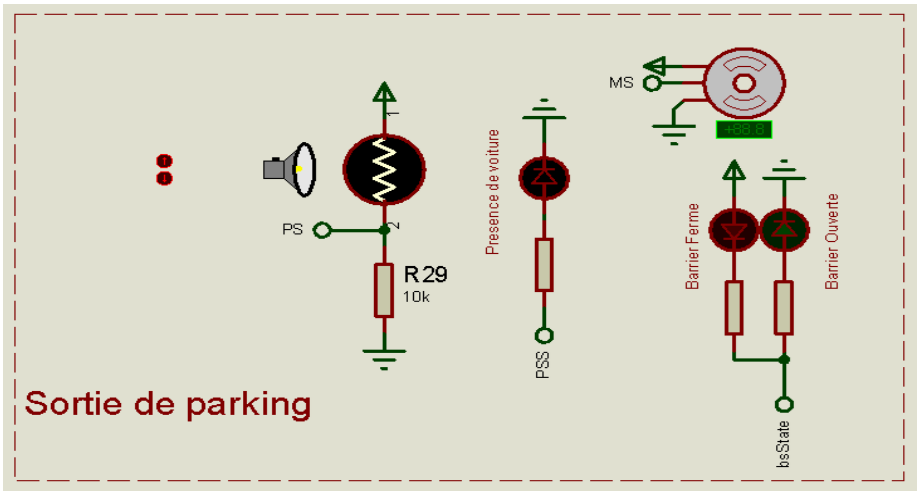


Figure 3.19 *Circuit électronique de sortie*

- **Carte Arduino méga utilisée**

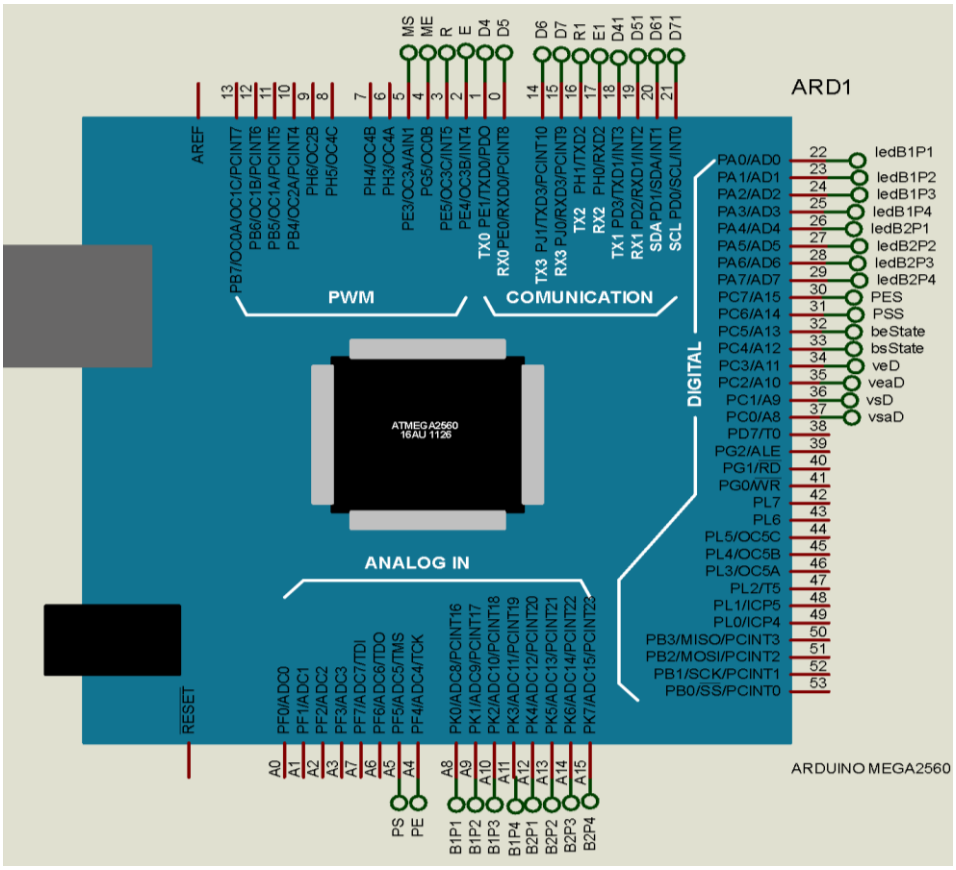


Figure 3.20 *Entrées/Sorties utilisées de la carte Arduino Méga*

3.3 Cas du deuxième système de parking

3.3.1 Cas du parking de stationnement avec l'API

Dans cette partie, nous allons voir une autre possibilité pour la commande du parking. Il s'agit d'un programme PLC pour le contrôle d'entrée / sortie du sous-sol ou du parking souterrain.

Dans ce cas, un compteur peut être utilisé pour suivre le nombre de véhicules dans un parking. Donc, trois principaux points seront considérés.

- Lorsque les véhicules entrent dans le parking par une porte d'entrée, le compteur compte.
- Lorsque les véhicules sortent du parking par une porte de sortie, le compteur compte à rebours.

- Lorsque les emplacements du stationnement sont complètements utilisés, un panneau à la porte d'entrée s'allume pour indiquer que le parking est plein.

3.3.1.1 Principe de fonctionnement

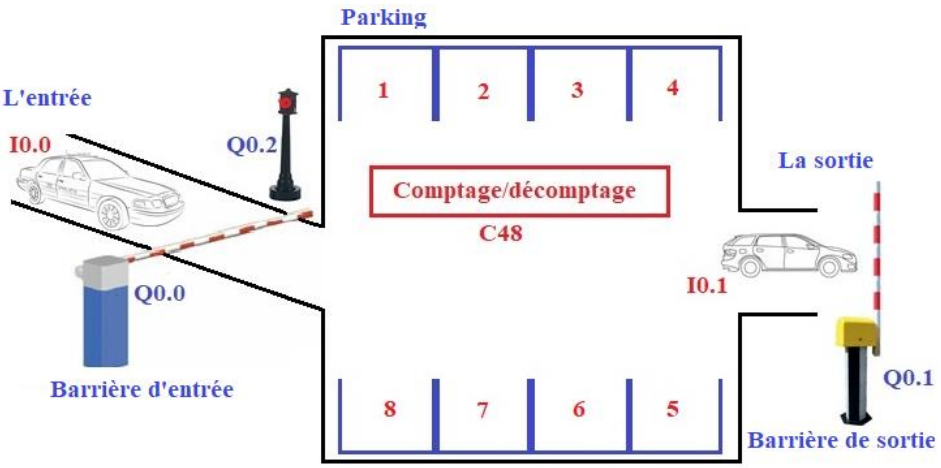


Figure 3.21 *Schéma principe du deuxième système de parking à étudier*

Le compteur / décompteur C48 est utilisé dans notre cas.

- Un interrupteur, connecté à la porte d'entrée, a été câblé à l'entrée I0.0.
- Un interrupteur, connecté à la porte de sortie, a été câblé à l'entrée I0.1.
- Un interrupteur de réinitialisation, situé sur la cabine de collecte, a été câblé à l'entrée I0.2. dans notre cas, le parking dispose de 8 places de parking. Cette valeur a été enregistrée dans la valeur prédéfinie (PV) du compteur. La sortie du compteur a été dirigée vers la sortie Q0.2. La sortie 2 est reliée à un panneau «Parking plein». Lorsque les voitures entrent dans le parking, la barrière d'entrée s'ouvre. L'entrée I0.0 passe d'un 0 logique à un 1 logique, incrémentant le comptage de un. Lorsque les voitures quittent le parking, la barrière de sortie s'ouvre. L'entrée I0.1 passe d'un 0 logique à un 1 logique, décrémentant le comptage de 1. Lorsque le comptage a atteint 8, la sortie Q0.2 passe d'un 0 logique à un logique 1. Le signe «Parking plein» s'allume. Lorsqu'une voiture sort, en décrémentant le compte à 7, le signe s'éteint.

3.3.1.2 Programme de stationnement de voiture PLC

Dans cette application, nous avons utilisé l'automate **Siemens S7-200** et le logiciel **Step 7Micro/WIN** pour la programmation.

Le progiciel de programmation STEP 7--Micro/WIN fournit un environnement pour concevoir, éditer et surveiller la logique nécessaire à la commande des applications. Comme nous remarquons d'après la figure suivante, STEP 7--Micro/WIN comprend trois éditeurs de programme, ce qui s'avère très pratique et efficace pour la mise au point du programme de commande des applications.

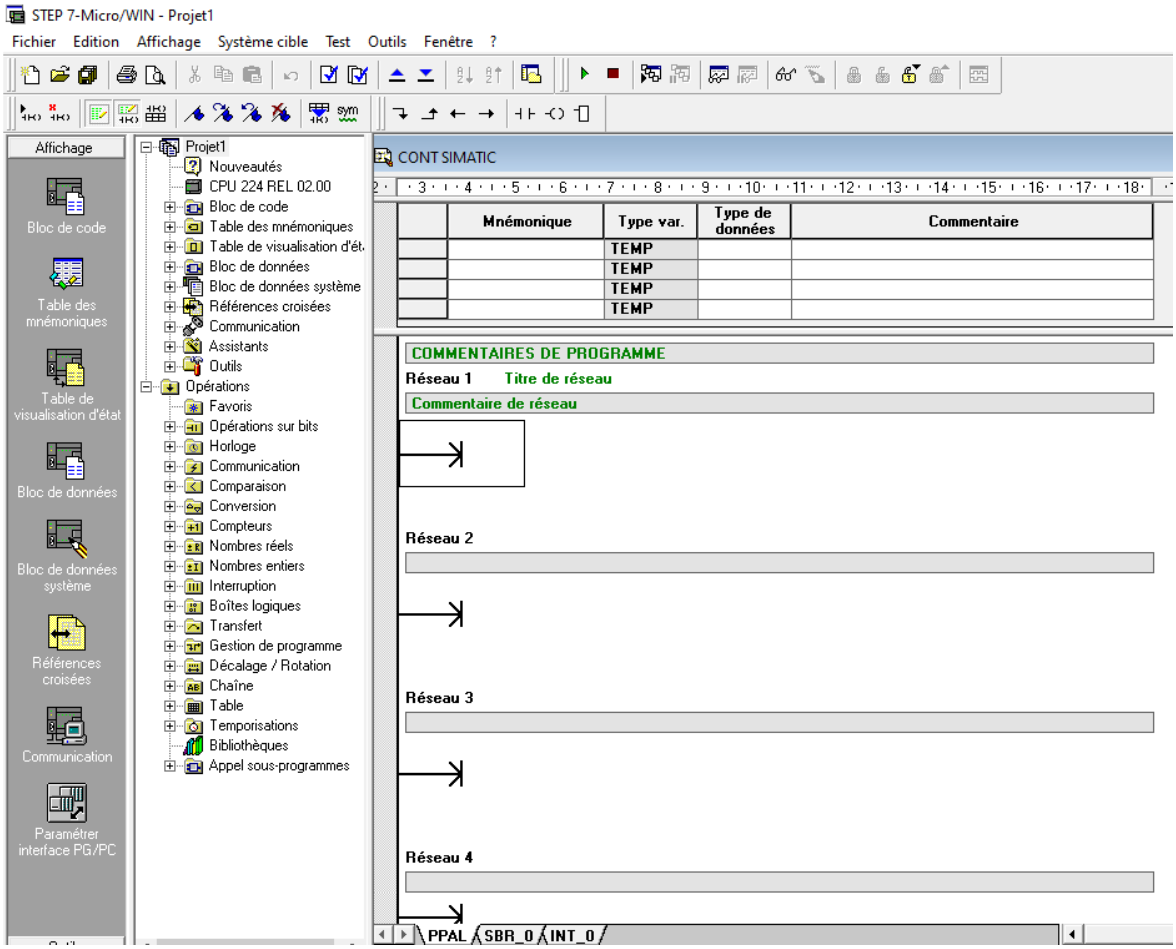


Figure 3.22 *L'interface du step 7-Micro/WIN*

- **Les types d'opérations utilisées**

Dans ce qui suit, nous montrons les différentes opérations sur lesquelles sera basé notre programme. Pour notre programme, nous avons besoin de quatre types d'opérations, figure 3. 3 :

- ✓ Opérations sur bits
- ✓ Opérations de comparaison
- ✓ Opérations de temporisation
- ✓ Opérations de comptage

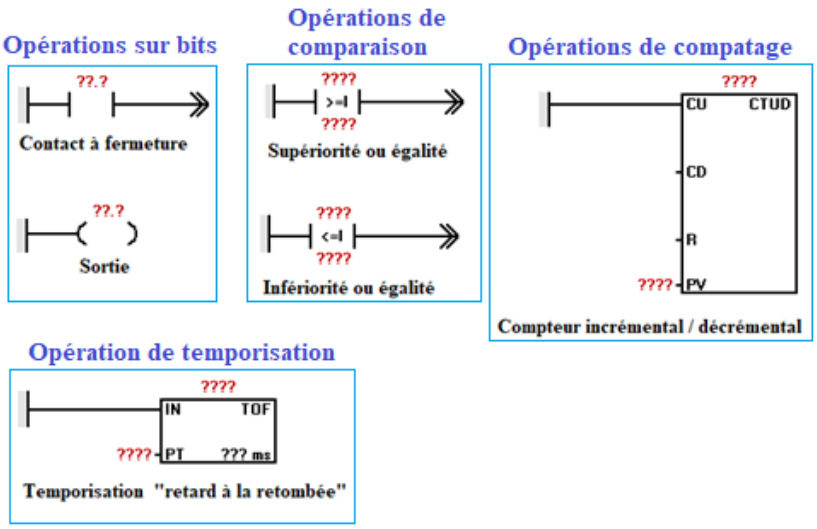


Figure 3.23 *Types d'opérations utilisées dans le programme*

- **Utilisation de la table des mnémoniques**

Le tableau 3.2 représente la table des mnémoniques. Ainsi, nous avons affecté dans cette table un nom symbolique à toutes les adresses absolues que nous voulons appeler dans le programme, par exemple pour l'entrée I0.1 le mnémonique Voiture_E. Ces noms valent pour toutes les sections du programme, et seront considérés comme des variables globales. Dee plus, leur utilisation permet d'avoir une clarté dans le programme.

Table des mnémoniques

	Mnémonique	Adresse	Commentaire
1	Voiture_E	I0.0	Voiture_E
2	Voiture_S	I0.1	Voiture_S
3	Reset	I0.2	Reset
4	Barrière_E	Q0.0	Barrière_E
5	Barrière_S	Q0.1	Barrière_S
6	Disponibilité	Q0.2	Disponibilité

Tableau 3.2 *La table des mnémoniques*

- **Section instructions du programme**

Dans le programme « Parking de stationnement », la section instructions se compose de plusieurs réseaux, qui eux-mêmes contiennent des opérations reliées entre elles, comme le montre la figure 3.

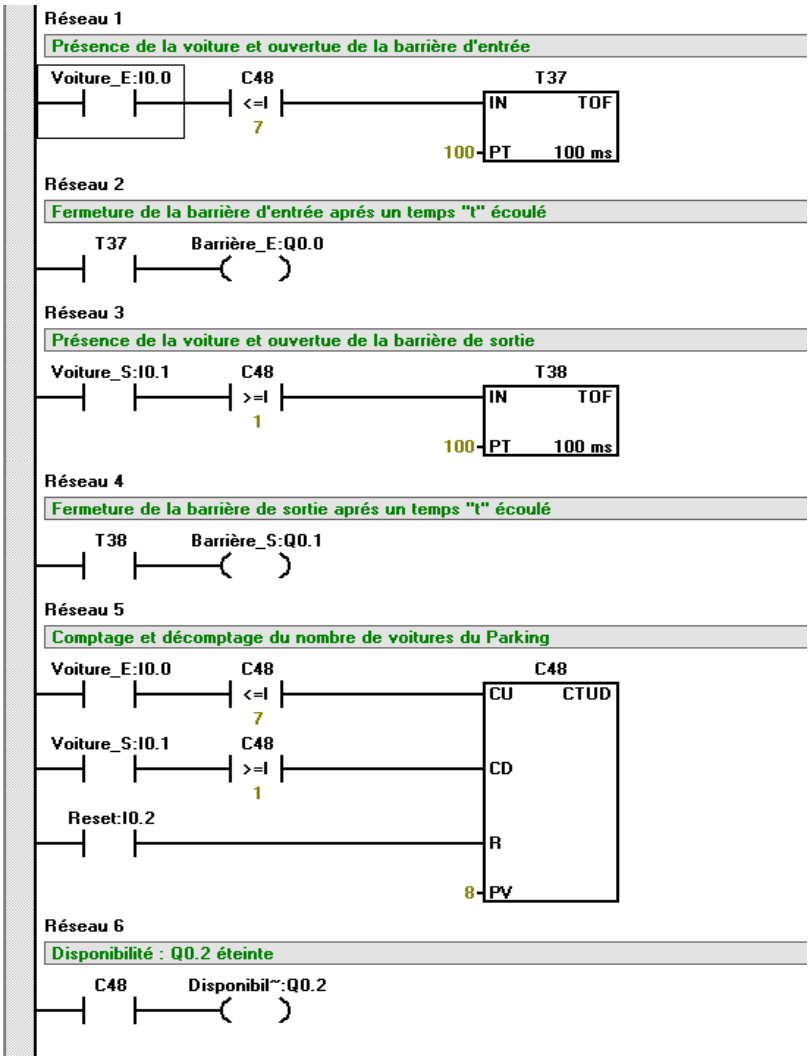


Figure 3.24 *Structure de la section instructions du programme*

- **Description du programme**

Réseau 1 :

Conformément à l'explication ci-dessus dans le premier réseau lorsque le système est allumé (I0.0) et la condition de disponibilité est vraie (C48<=7), ainsi la temporisation (T37) sera activée. Ce qui permet de calculer le temps pendant lequel I0.0 a été remis à 0.

Réseau 2 :

Comme expliqué ci-dessus dans le premier réseau, l'opération TOF retarde la désactivation de la sortie (Q0.0) de la barrière d'entrée pour un intervalle de temps donné après que l'entrée (I0.0) a été désactivée.

Réseau 3 :

Lorsque la voiture se présente à la sortie du parking, l'entrée (I0.1) sera déclenchée tant que la condition de disponibilité est vérifiée (C48>=1). Ici, nous avons le même principe que celui du réseau1 en ce qui concerne le temporisateur (T38).

Réseau 4 :

Lorsque l'entrée de validation (I0.1) est désactivée Lorsque le temps écoulé est égal au temps prédéfini (PT), le bit de sortie de la temporisation (Q0.1) de la barrière de sortie est désactivé.Donc, Si la durée est plus courte que le temps prédéfini, (Q0.1) reste activé.

Réseau 5 :

Continuer le comptage pour faire varier la valeur entière C48.Donc, la comparaison d'entiers vérifie si C48<=7 est vraie.

De même, continuer le décomptage tant que la valeur C48>=1

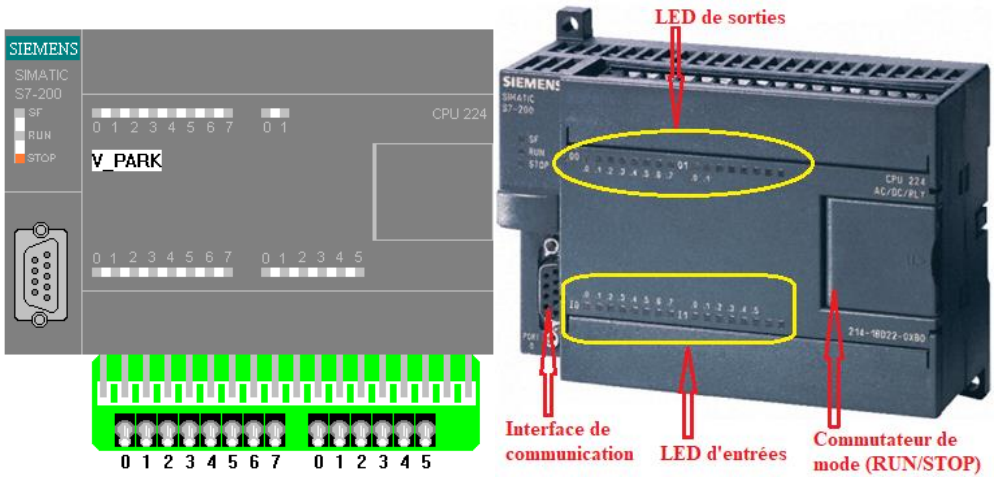
Réseau 6:

Dans ce réseau, la sortie (Q0.2) sera activée lorsquele compteur atteindra le nombre total de places du parking indiqué en référence.

La sortie (Q0.0) permet de conserver la désactivation du fonctionnement de la barrière d'entrée si aucune voiture ne se présentera en sortie.

4 Simulation

Pour pouvoir tester notre programme décrit ci-dessus, nous devons d'abord le charger dans la CPU du simulateur S7-200, voir figure 2. ? (a)



(a) (b)

Figure 3.25 *Micro-automate S7-200*

(a) *Modèle de simulation* (b) *Modèle physique*

La figure 3.26 Visualise le cas de figure après l'exécution du programme. Certaines opérations ont pris d'autres couleurs, ce qui prouve la circulation du courant dans les réseaux du bloc. Nous avons la situation où le nombre de places est disponible.

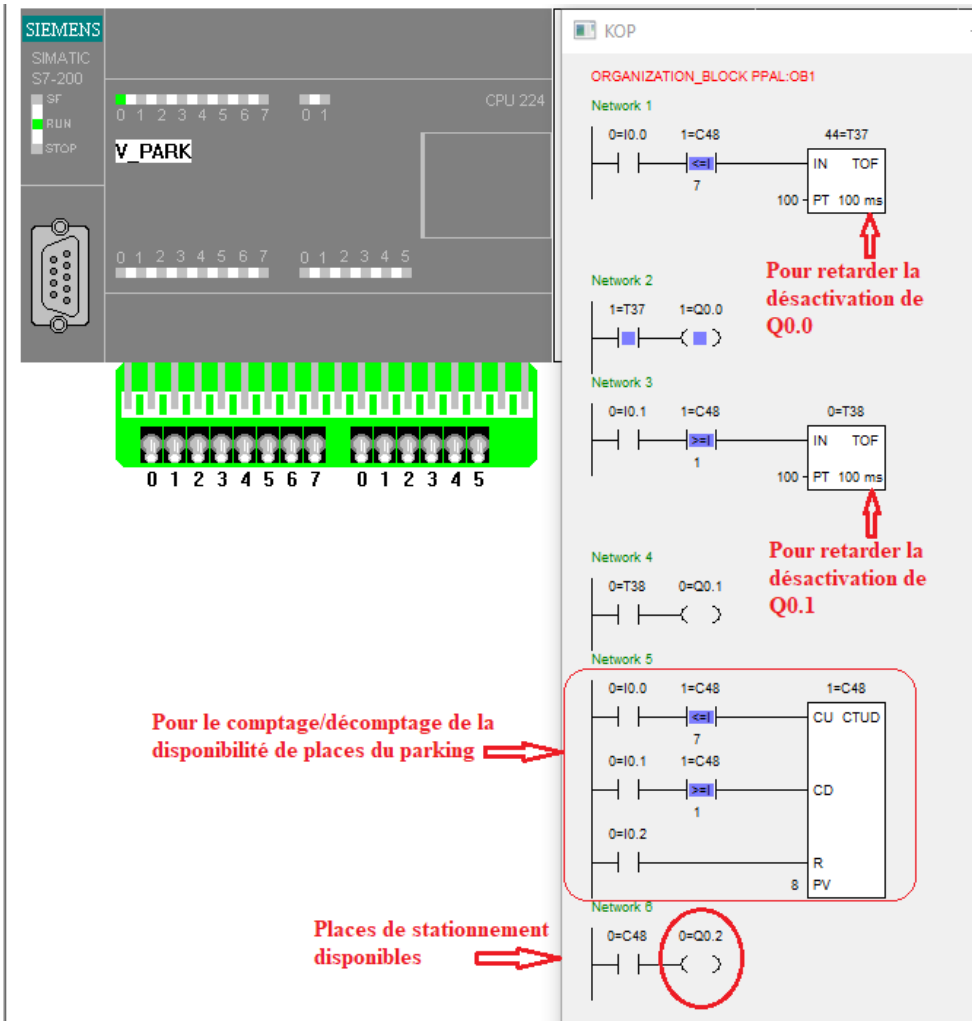


Figure 3.26 *Fonctionnement du parking. Places disponibles*

La figure 3.27 illustre le cas où le nombre de places n'est plus disponible. Dans cette situation, le compteur est programmé de telle sorte qu'il ne poursuivra plus le comptage, et il restera bloqué tant que la barrière de sortie (Q0.1) restera inactive.

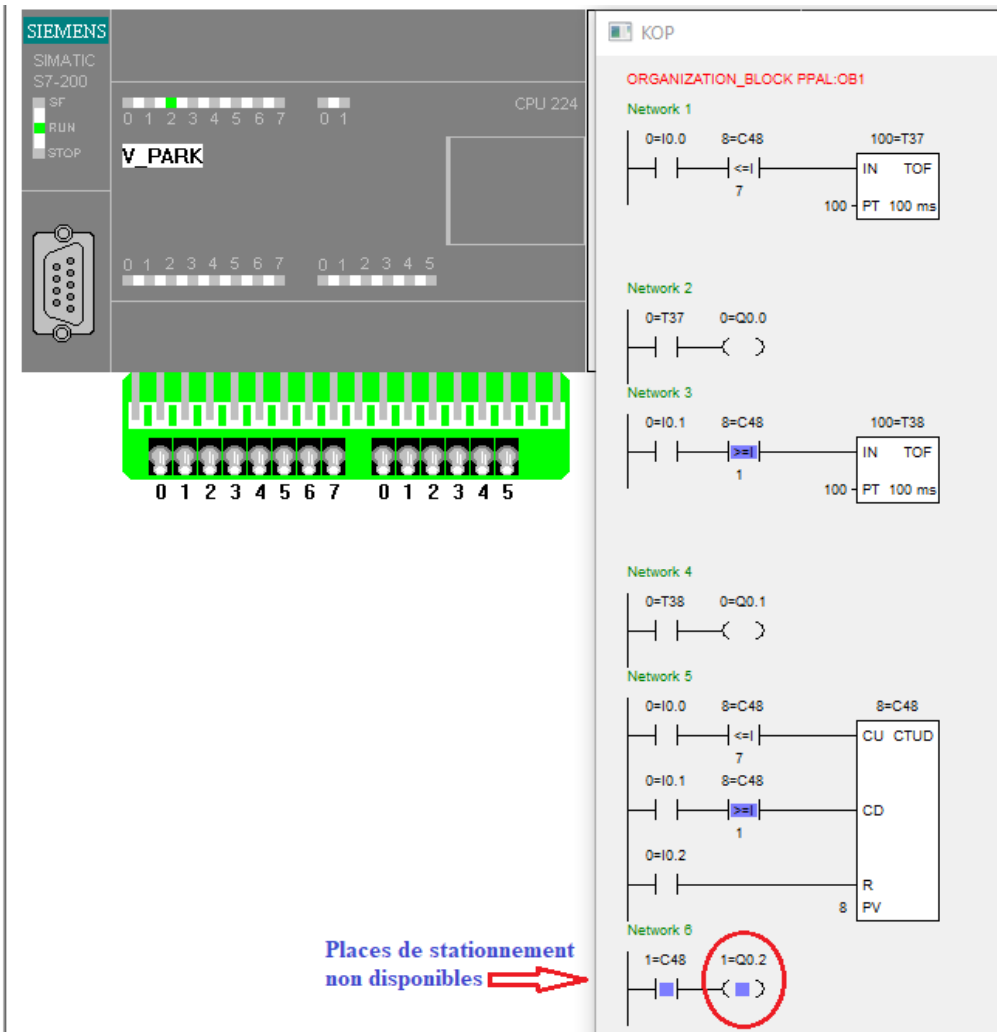


Figure 3.27 *Fonctionnement du parking. Places non disponibles*

3.3.2 Cas du parking de stationnement avec Arduino « Uno »

Dans ce cas, nous allons reprendre le même circuit traité ci-dessus avec API, mais cette fois-ci avec l'Arduino Uno pour la simulation.

3.3.2.1 Système complet de simulation

Le circuit suivant illustré à la figure 3.28 montre sa réalisation obtenue avec Proteus. Nous remarquons :

- à l'entrée une indication qui devrait s'allumer dans le cas où les places au niveau du parking sont complètement occupées, et un circuit pour simuler le fonctionnement d'une barrière pour le passage des véhicules à l'entrée de ce parking.

- A la sortie un circuit pour simuler le fonctionnement d'une barrière pour le passage des véhicules sortantes de ce parking.
- La partie commande assurée par une carte Arduino Uno. Ce qui permet l'affichage du nombre de places du parking en fonction des entrées/sorties des véhicules du parking.

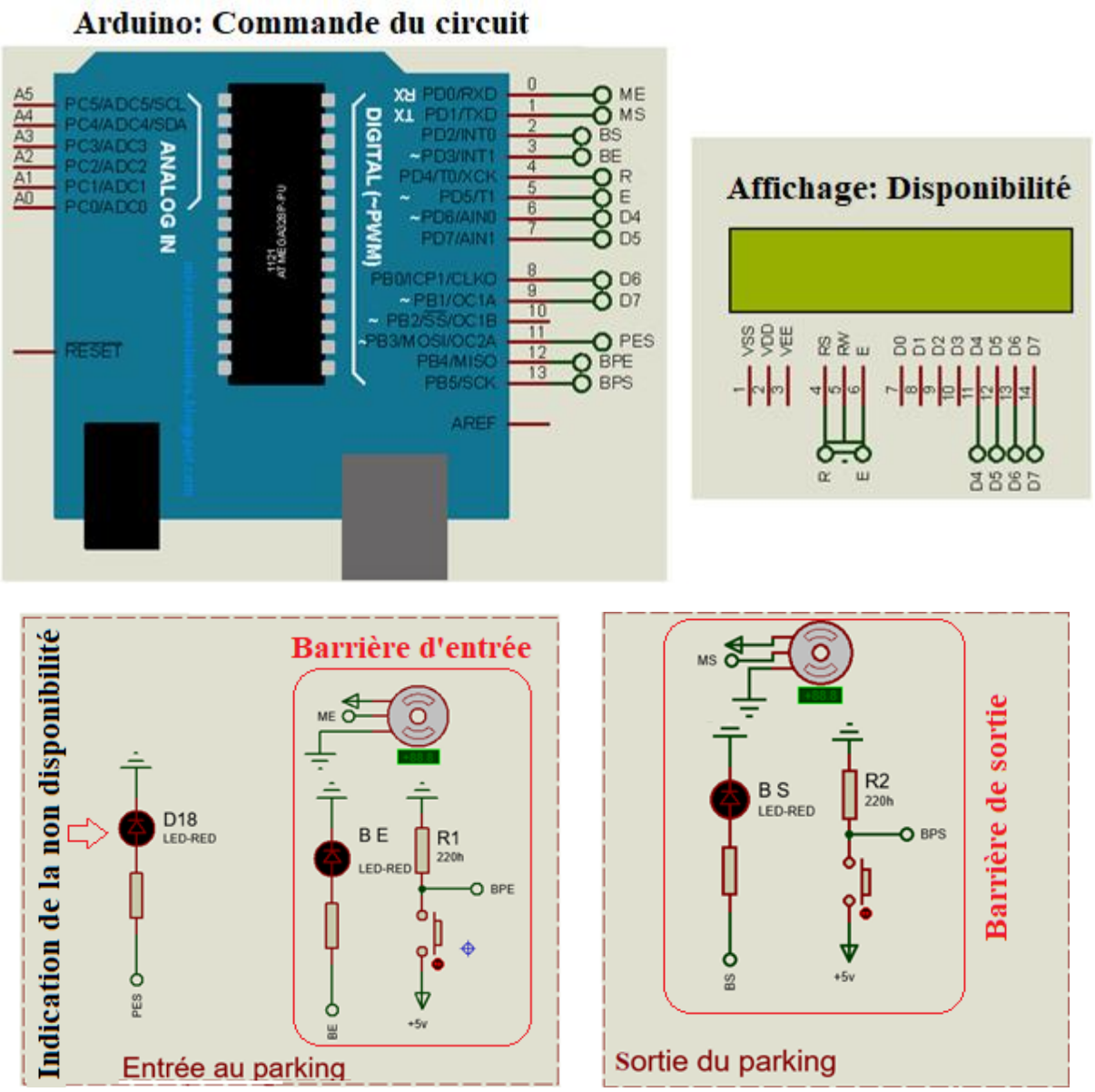


Figure 3.28 *Schéma complet de simulation du parking avec Arduino*

A- Organigramme1 : entrée au parking

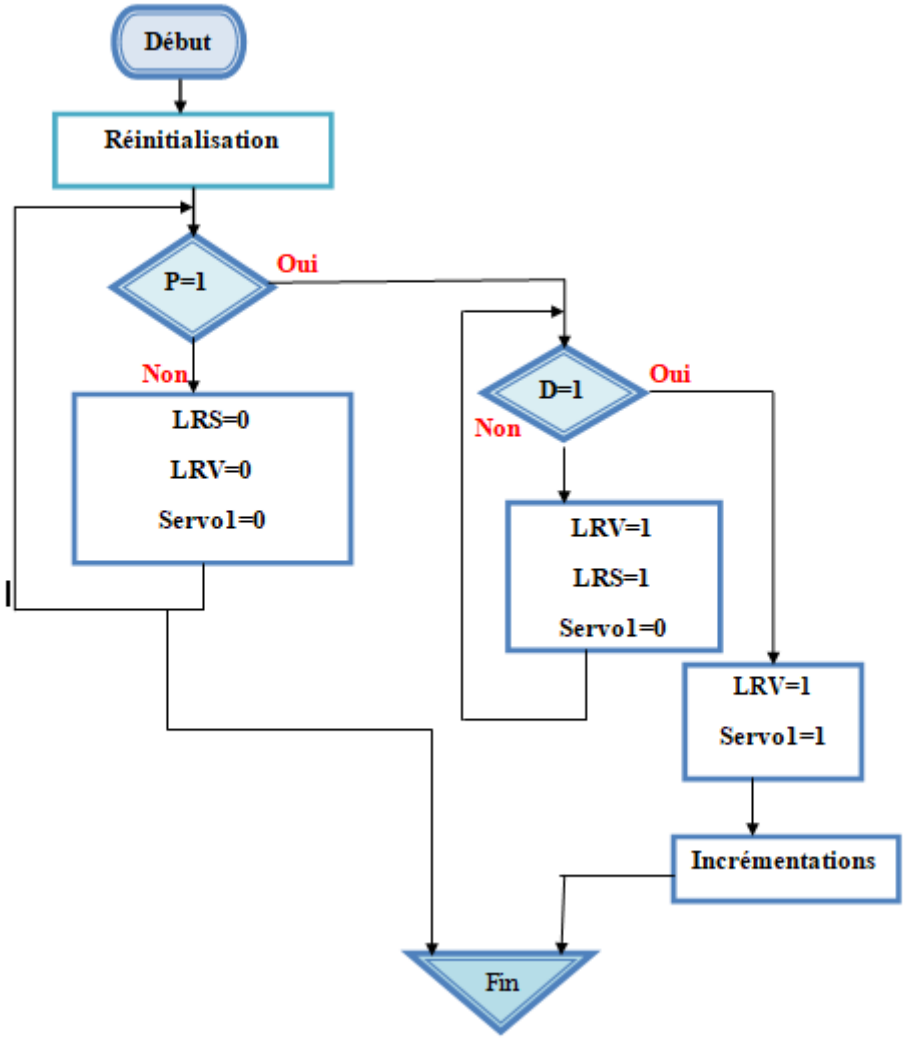


Figure 3.29 *Organigramme d'entrée*

P=1 présence de voiture
Absence de voiture
D=place disponible
LRB=1 LED rouge barrière allumée
LRS=LED rouge de saturation

B- Organigramme2 : sortie du parking

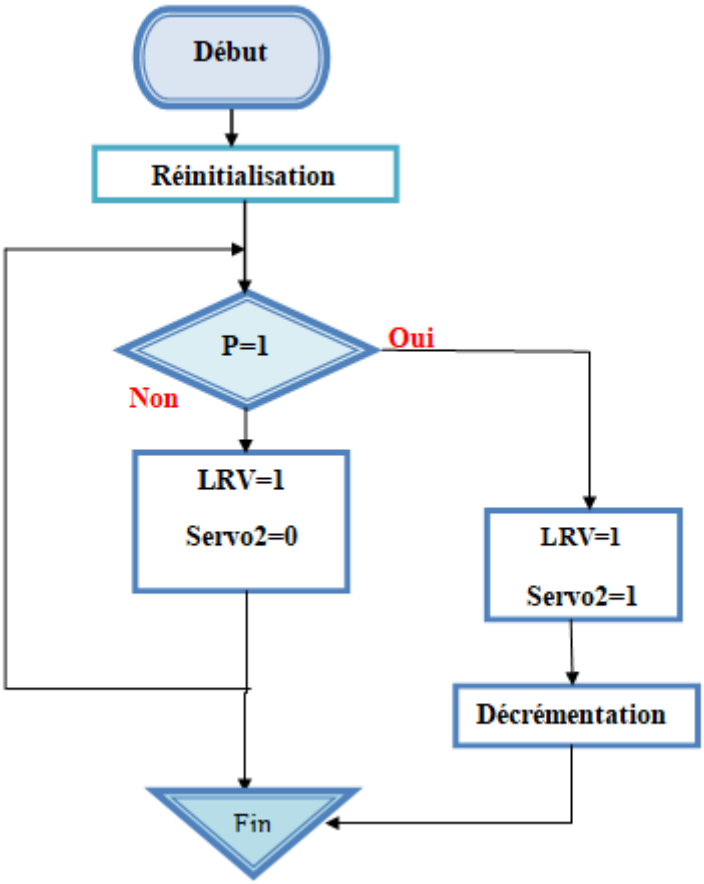


Figure 3.30 *Organigramme de sortie*

P=1 présence de voiture
LRV =1 LED rouge voiture allumée (présence de voiture)

3.4 Conclusion

Dans ce chapitre nous avons effectué plusieurs simulations des différents circuits électroniques étudiés. Ainsi, les résultats obtenus sont satisfaisants, du fait que les sorties du système répond aux différentes commandes selon les circuits d'entrées, et cela dans les deux cas de situations avec carte Arduino. De plus, à travers cette étude, nous avons constaté l'avantage d'un système à base d'une carte à microcontrôleur. Donc, l'étape de simulation est nécessaire avant toute réalisation pratique, ce qui fera l'objet de notre dernier chapitre.

Chapitre4 :

Réalisation pratique

Cas avec carte Arduino

4.1 Introduction

Notre étude a été vérifiée à l'aide de simulations effectuées dans le cas du deuxième système de parking dans le chapitre précédent. Dans ce qui suit, cela va être vérifié par sa réalisation pratique pour montrer son fonctionnement. Donc, différentes étapes seront illustrées dans cette partie.

4.2 Organigramme de fonctionnement du deuxième système de parking

L'organigramme suivant représente le fonctionnement de notre système de parking.

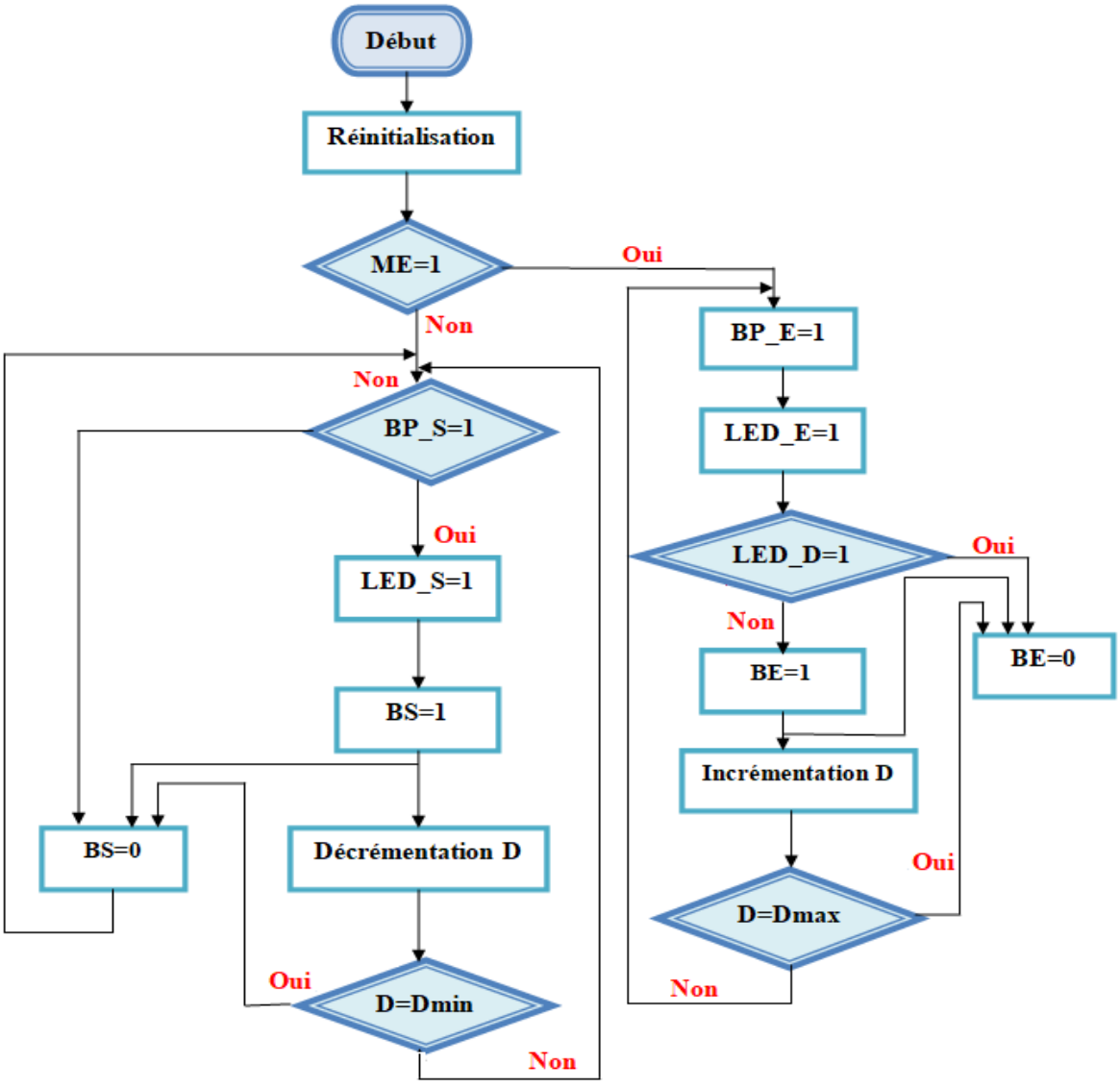


Figure 4. 1*Organigramme représente le fonctionnement général du système*

S : sortie
D : disponibilité
ME :mode entrée
BP : bouton poussoir
BE : barrière d'entrée
BS :barrière sortie

4.3 Chargement du programme principal

Apres l'écriture de code principal, il y a des étapes principales pour charger le programme sur Arduino :

4.3.1 Choix du type de carte Arduino

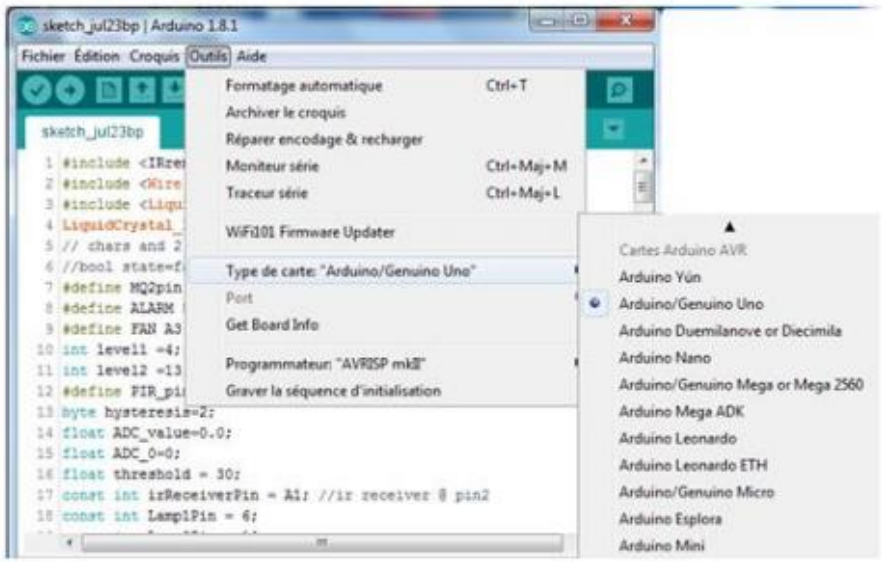


Figure 4.2*Choix de type de carte Arduino*

4.3.2 Choix du Port série (COM Port)

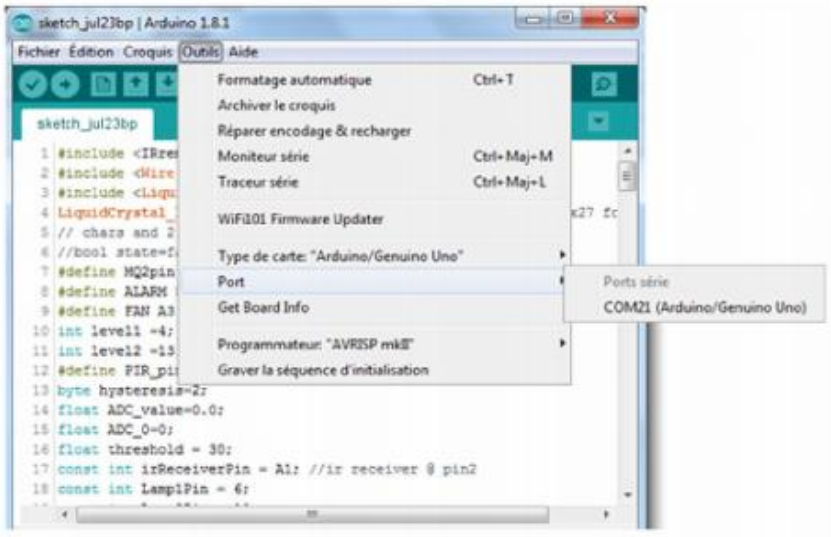


Figure 4.3 Choix du port série

4.3.3 Connexion entre le PC et la carte Arduino

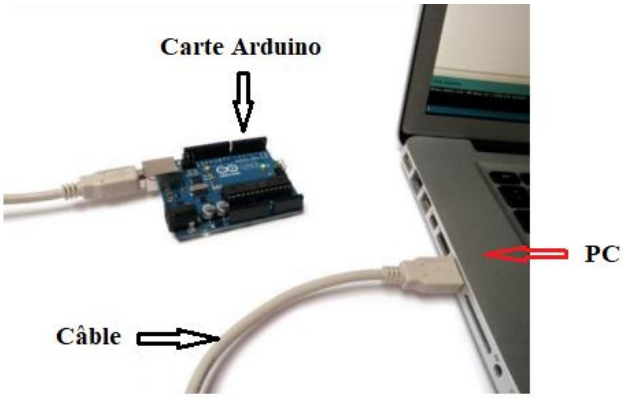


Figure 4.4*Câblage entre le PC et la carte Arduino*

4.3.4 Chargement du programme sur l'Arduino

La figure 4. 4, illustre l'étape pour le téléversement du programme déjà écrit et sauvegardé au niveau du PC vers la carte Arduino afin d'assurer le bon fonctionnement du système réalisé.

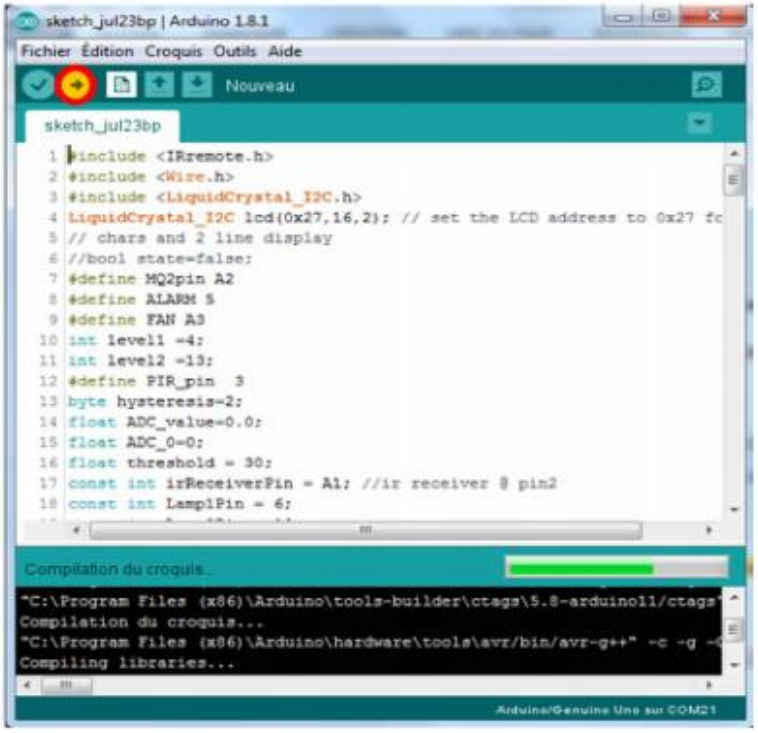


Figure 4.5 *Téléversement du programme vers la carte Arduino*

4.4 Coût du circuit réalisé

Le tableau suivant donne le coût des différents composants utilisés.

Désignation	Prix unitaire (DA)	Quantité	Total ligne (DA)
Arduino UNO	3000.00	1	3000.00
Servomoteur	800.00	2	1600.00
Résistance (220Ω)	05.00	5	25.00
Led rouge	10.00	3	30.00
Bouton poussoir	300.00	2	600.00
Afficheur LCD (16x2	800.00	1	800.00
breadboard	800.00	1	800.00
Total			**6 855.00**

Tableau 4.1 *Le coût total du circuit selon les différents composants utilisés*

4.5 Circuit complet du deuxième système réalisé

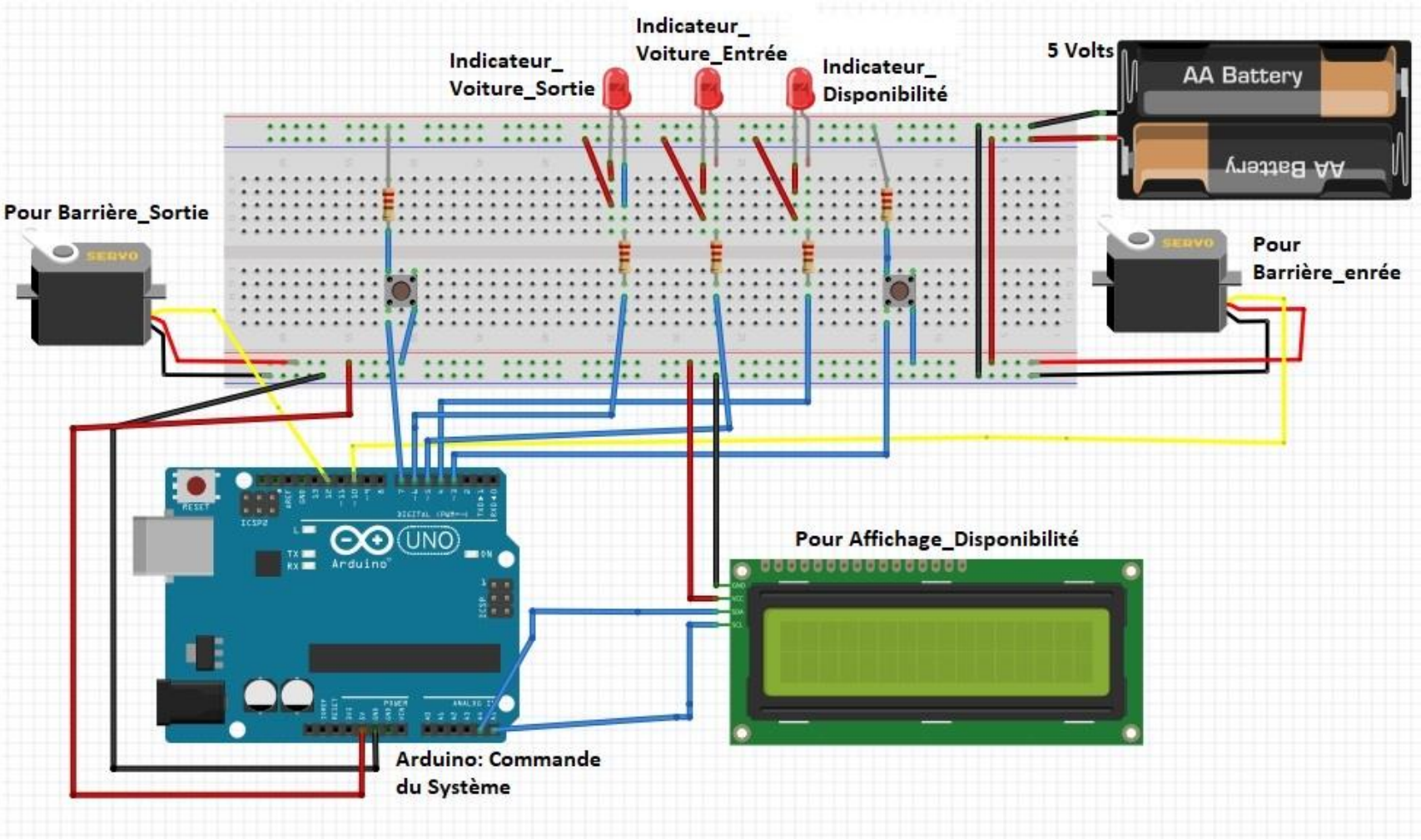


Figure 4.6 *Schéma du circuit complet du parking réalisé*

4.6 Mise en œuvre

Nous donnons quelques exemples de photos du circuit réel correspondant au parking intelligent à base de la carte Arduino.

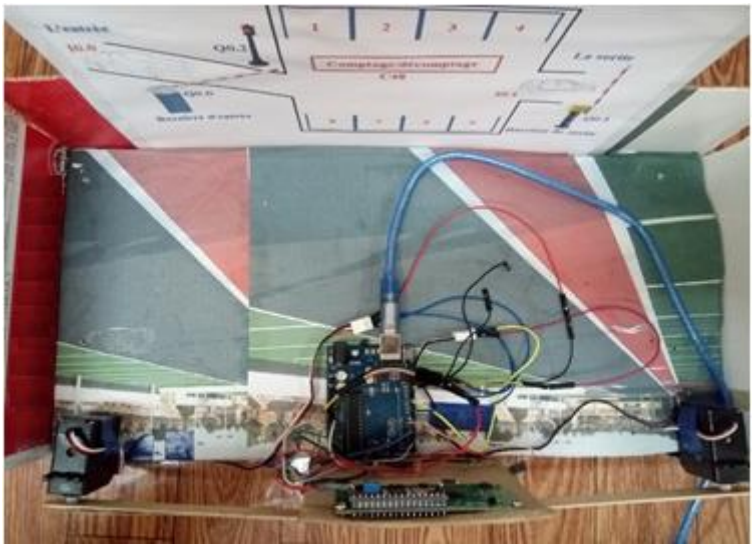


Figure 4.7 *Photo du circuit complet réalisé. Vue de dessus*

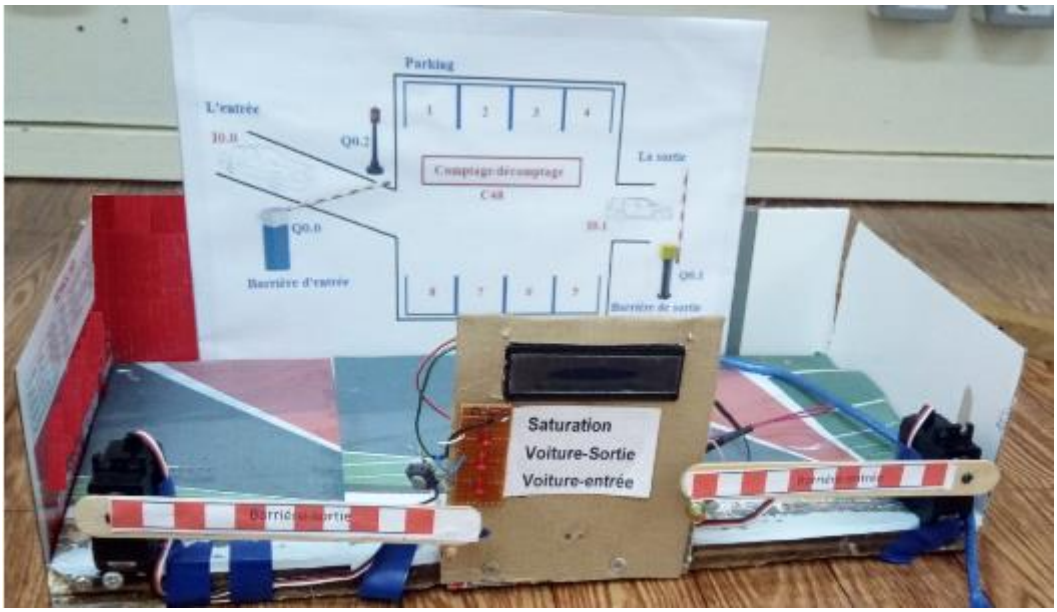


Figure 4.8 *Photo du circuit complet réalisé. Vue de face*

4.7 Conclusion

Dans cette partie, les tests des différentes sorties du système réalisé pratiquement, nous ont donné la possibilité de montrer le bon fonctionnement des circuits en entrées formellement en relation avec le circuit de commande géré à partir d'une carte Arduino UNO. Donc, plusieurs éventualités peuvent être considérées pour améliorer cecircuit ainsi que les programmes.

Conclusion générale

L'objectif de mon projet était d'étudier et de simuler un parking de stationnement souterrain à base d'une carte ARDUINO et d'un API.

D'abord, l'étude bibliographique menée a permis de se rendre compte de l'importance d'un système actif donné vis-à-vis du poids des nouveautés technologiques. Se passer de ce développement permettrait le fonctionnement des systèmes sans avoir recours à des améliorations.

Pour cela, plusieurs simulations ont été effectuées pour tester le fonctionnent des différents circuits étudies. Dans ce cas, j'ai exploité le logiciel de conception assistée par ordinateur pour l'électronique « Proteus » pour la réalisation des schémas électroniques, et l'utilisation de la plateforme step7 micro/win pour la réalisation du programme à contacts. Donc, deux résultats sont obtenus, le premier avec un système à base de l'ARDUINO et le second en considérant le simulateur siemens S7 200.

De plus, les deux cas de situation m'ont donné un point avantageux et motivant, et qui m'ont poussé à avoir un certain savoir-faire dans le domaine de notre spécialité. Bien sûre, durant la période consacrée à l'étude de mon sujet, j'ai rencontré pas mal d'obstacles, voire la contrainte liée au manque de mes compétences sur l'application de l'ARDUINO en l'exploitant dans la réalisation pratique avec des composants électroniques ; domaine très nouveau pour moi, mais à l'issu de ce projet, j'ai pu saisir son importance et son utilisation.

Le travail développé dans le cadre de ce mémoire ouvre des voies d'amélioration que l'on peut classer comme suit :

- Application du système réalisé avec d'autres types de microcontrôleurs.
- Développement du système réalisé en se basant sur le principe des objets connectés
- Réalisation pratique de l'application du système réalisé
- Réalisation pratique du système réalisé en se basant sur un API

Références bibliographiques

[1] https://www.transitionsenergies.com/combien-voitures-monde/

[2] https://books.openedition.org/pur/16232?lang=fr

[3] https://www.a-vos-moteurs.fr/auto-quen-stationnement-autonome-intelligent-292.html

[4] https://www.larousse.fr/dictionnaires/francais/intelligence/

[5] https://fr.wikipedia.org/wiki/Ville_intelligente

[6] https://fr.wikipedia.org/wiki/Système

[7] https://fr.wikiversity.org/wiki/Capteur/

[8] Henri Nussbaumer, 'Informatique industrielle I', Représentation et traitement de l'information, collection informatique, 1994.

[9] https://www.larousse.fr/dictionnaires/francais/actionneur/

[10] Jean-Simon BOURDEAU, 'Méthodologie d'analyse automatisée des stationnements', Maîtrise ès sciences appliquées, Université de Montréal, Août 2014

[11] https://www.onepark.fr/wiki/120-tout-sur-les-parkings/143-parking-ferme-ou-souterrain

[12] William Bolton,'automates programmables industriels', DUNOD ,2015

[13] https://www.auto-moto.com/actualite/high-tech/parking-connecte-les-4-solutions-futuristes-video-57814.html#item=1

[14] Susan A. Shaheen et Caroline Rodier,' Smart Parking Management Field Test: A Bay Area Rapid Transit (BART) District Parking Demonstration', California PATH Working PaperUCB-ITS-PWP-2006-10.

[15] https://www.banquedesterritoires.fr/nice-lance-son-systeme-de-stationnement-intelligent

[16] https://www.forbes.com/sites/pikeresearch/2017/01/26/smart-parking/#51e980f462f6

[17] https://www.youtube.com/watch?v=ffgX-Sdpbg0

[18] https://iiot-world.com/smart-cities-buildings-infrastructure/smart-cities/smart-parking-innovations-look-beyond-parking/

[19] http://www.jardaqua.com/avantages-inconvenients-maison-intelligente.htm

[20] https://www.onepark.fr/wiki/120-tout-sur-les-parkings/143-parking-ferme-ou-souterrain

[21] Henri Nussbaumer, informatique industrielle, représentation et traitement de l'information, collection informatique, 1994.

[22] Ahmed Dahmani, 'Les technologies numériques dans les pays en développement. Quel paradigme ?', Technologies mobiles, innovation et développement, Dossiers, 6/2018.

[23] https://www.algerie-focus.com/2013/12/alger-bientot-des-parkings-intelligents-pour-remedier-a-labsence-despaces-de-stationnements/

[24] https://www.autobip.com/fr/actualite/1er_parking_intelligent_a_alger/935

[25] https://www.usine-digitale.fr/article/here-technologies-veut-realiser-des-cartes-hd-des-parkings-d-interieur-europeens.N969536

[26] https://www.aversanlabs.com/our-work/

[27] https://fr.wikipedia.org/wiki/Arduino

[28] https://www.redohm.fr/2020/01/carte-microcontroleur-2/

[29] https://www.generationrobots.com/fr/401684-carte-controleur-robotique-arbotix-m.html

[30] https://fr.wikipedia.org/wiki/Microcontr%C3%B4leur

[31] https://phychim.ac-versailles.fr/spip.php?article1076

[32] Jean-François THULLIER ,'microcontrôleur arduino pour les plp mathématiques –physique chimie ',4ème version, Mars 2020.

[33] Leo Louis,'Working principle of arduino and using it as a tool for study and research', International Journal of Control, Automation, Communication and Systems (IJCACS), Vol.1, No.2, April 2016.

[34] https://www.researchgate.net/publication/314674254_FORMATION_ARDUINO_MATLABSIMULINK_Commande_d%27un_systeme_thermique_a_l%27aide_de_la_carte_Arduino_UNO

[35] https://www.technologuepro.com/microcontroleur-2/arduino/programmer.html

[36] https://www.etudier.com/dissertations/Les-Differentes-Types-Des-Automates/74341039.html

[37] https://www.geea.org/IMG/pdf/LES_AUTOMATES_PROGRAMMABLES_INDUSTRIELS_pour_GEEA.pdf

[38] https://www.technologuepro.com/cours-automate-programmable-industriel/Les-automates-programmables-industriels-API.htm

[39] https://www.technologuepro.com/cours-automate-programmable-industriel/Les-automates-programmables-industriels-API.htm

[40] https://www.se.com/fr/fr/work/products/product-launch/guides/plc.jsp

[41] https://www.se.com/fr/fr/work/products/product-launch/guides/plc.jsp

Annexe A

Programme pour le fonctionnement du premier système de parking étudié

```
#include <Servo.h>
#include <LiquidCrystal.h>
                //rs, en, d4, d5, d6, d7
LiquidCrystal lcd1(16, 17, 18, 19, 20, 21);
LiquidCrystal lcd(3, 2, 1, 0, 14, 15);
Servo ME;
Servo MS;
bool P11,P12,P13,P14,P21,P22,P23,P24,E,S;
int Block01,Block02,totalPlace,oldtotalPlace;
int beState=32;
int bsState=33;
int veD=34;
int veaD=35;
int vsD=36;
int vsaD=37;
//======= Class Place=========
class Place {
   int dPin;
   int aPin;
 public:
   Place(int analogPin,int digitalPin){  //Constricteur
     dPin=digitalPin;
     aPin=analogPin;
     }
   // Fonction pour verifier la Place
```

```
bool valable(){
      int readValue=analogRead(aPin);
      if(readValue>=15){
         digitalWrite(dPin,LOW);
        return true;
        }else{
          digitalWrite(dPin,HIGH);
        return false;
          }}
};
```

//======= entrée –sortie =========

```
class gate {
int dgPin;
    int agPin;
  public:
    gate(int analoggPin,int digitalgPin){  //Constricteur
      dgPin=digitalgPin;
      agPin=analoggPin;
      }
    // Fonction pour verifier la Place
    bool valable(){
      int readgValue=analogRead(agPin);
      if(readgValue>=15){
         digitalWrite(dgPin,LOW);
        return false;
        }else{
```

```
          digitalWrite(dgPin,HIGH);
        return true;
          }}
};
void setup()
{
  for (int i=22;i<=31;i++)
{
pinMode(i,OUTPUT);}
pinMode(beState,OUTPUT);
pinMode(bsState,OUTPUT);
pinMode(veD,OUTPUT);
pinMode(veaD,OUTPUT);
pinMode(vsD,OUTPUT);
pinMode(vsaD,OUTPUT);
    ME.attach(4);
    ME.write(180);
    MS.attach(5);
    MS.write(180);
    lcd1.begin(16, 2);
    lcd.begin(16, 2);

}

void loop() {
  // definition d'entre
gate Entre(A4,30);
```

```
 // definition de sortie
   gate Sortie(A5,31);
 // definition des Places Block 01(rédiger les objet )
   Place B1P1(A8,22);
   Place B1P2(A9,23);
   Place B1P3(A10,24);
   Place B1P4(A11,25);
 // definition des Places Block 02
Place B2P1(A12,26);
   Place B2P2(A13,27);
   Place B2P3(A14,28);
   Place B2P4(A15,29);
 // Valable des places Block 01
P11=B1P1.valable();
   P12=B1P2.valable();
   P13=B1P3.valable();
   P14=B1P4.valable();
   P21=B2P1.valable();
   P22=B2P2.valable();
   P23=B2P3.valable();
   P24=B2P4.valable();
 Block01=checkBlock(P11,P12,P13,P14);
Block02=checkBlock(P21,P22,P23,P24);
 totalPlace=Block01+Block02;
 Display();
 while(1){
     E=Entre.valable();
```

```
oldtotalPlace=totalPlace;
    if(E && (totalPlace>0)){
          while(E){
            ME.write(90);
            digitalWrite(beState,HIGH);
E=Entre.valable();
            digitalWrite(veD,HIGH);
            }
        digitalWrite(veD,LOW);
        digitalWrite(veaD,HIGH);
        delay(500);
        digitalWrite(veaD,LOW);
        digitalWrite(beState,LOW);
        ME.write(180);
          while((totalPlace>=oldtotalPlace)){
          // Check Apres entre(verifier et calculer )
            P11=B1P1.valable();
            P12=B1P2.valable();
            P13=B1P3.valable();
            P14=B1P4.valable();
            P21=B2P1.valable();
            P22=B2P2.valable();
            P23=B2P3.valable();
            P24=B2P4.valable();
          Block01=checkBlock(P11,P12,P13,P14);
          Block02=checkBlock(P21,P22,P23,P24);
          totalPlace=Block01+Block02;
```

```
                    }
oldtotalPlace=totalPlace;
          Display();
     }else{
       digitalWrite(beState,LOW);
       digitalWrite(veaD,LOW);
       digitalWrite(veD,LOW);
      }
  Display();
    S=Sortie.valable();
    if(S && (totalPlace<=8)){
          while(S){
            MS.write(90);
            digitalWrite(bsState,HIGH);
            S=Sortie.valable();
            digitalWrite(vsD,HIGH);
            }
       digitalWrite(vsD,LOW);
       digitalWrite(vsaD,HIGH);
       delay(500);
       digitalWrite(vsaD,LOW);
       digitalWrite(bsState,LOW);
       MS.write(180);
          while((totalPlace<=oldtotalPlace)){
          // Check apres entre
            P11=B1P1.valable();
            P12=B1P2.valable();
```

```
            P13=B1P3.valable();
            P14=B1P4.valable();
            P21=B2P1.valable();
            P22=B2P2.valable();
            P23=B2P3.valable();
            P24=B2P4.valable();
          Block01=checkBlock(P11,P12,P13,P14);
          Block02=checkBlock(P21,P22,P23,P24);
          totalPlace=Block01+Block02;
           //==================
           }
       oldtotalPlace=totalPlace;
           Display();
     }else{
       digitalWrite(bsState,LOW);
       digitalWrite(vsaD,LOW);
       digitalWrite(vsD,LOW);
      }
    Display();
  }
}
```

//Fonction pour verifier le nmbre des voitures dans les block

```
int checkBlock(bool a,bool b,bool c,bool d){
  int blockValue=d+c*2+b*4+a*8;
  String bin_data;
  bin_data = int2bin(blockValue);
//pour Calculer combien un '1' dans une chaine de character
```

```
  char checkCharacter = '1';
  int count = 0;
  for (int i = 0; i < 5; i++)
  {
     if (bin_data[i] ==  checkCharacter)
     {
        ++ count;
     }
  }
  //------------------
 return count;
 }

//Fonction pour convertir int to bin
String int2bin (int nb)
 {
 int   i = 4;
 String  bin = "";
 while (i >= 0) {
  if ((nb >> i) & 1)
    bin = bin + "1";
  else
    bin = bin + "0";
  --i;
 }
 return bin;
}
```

// Fonction pour l’affichage

```
void Display(){
    //Affichage Total
      lcd.setCursor(2, 0);
      lcd.print("Disponible: ");
      lcd.print(totalPlace);
      lcd.setCursor(0, 1);
      lcd.print("B01: ");
      lcd.print(Block01);
      lcd.setCursor(7, 1);
      lcd.print("|| B02: ");
      lcd.print(Block02);
    //Affichage par Block
      lcd1.setCursor(0, 0);
      lcd1.print("Block 01: ");
      lcd1.setCursor(11, 0);
      lcd1.print(Block01);
      lcd1.setCursor(0, 1);
      lcd1.print("Block 02: ");
      lcd1.setCursor(11, 1);
      lcd1.print(Block02);
   }
```

Annexe B

Programme pour le fonctionnement du deuxième système de parking réalisé

```
#include <Servo.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27,16,2);
Servo ME ;
Servo MS  ;
int i = 0;
int PES = 2;
int BS = 5;
int BE = 6;

const int UP = 7;     // the number of the pushbutton pin
const int DOWN =  3;

void setup() {

  pinMode(PES, OUTPUT);
  pinMode(UP, INPUT);
  pinMode(DOWN, INPUT);
  pinMode(BS, OUTPUT);
  pinMode(BE, OUTPUT);

  ME.attach(12);
  //ME.write(0);
  MS.attach(10);
  //MS.write(0);
```

```
 lcd.init();                  // initialize the lcd Print a message to the LCD.
 lcd.backlight();
lcd.begin(16, 2);
 lcd.print("park open");
   }

void loop() {
 digitalWrite(BS, LOW);
 digitalWrite(BE, LOW);
digitalWrite(PES, LOW);
 if (i>8){
 i=8;}
 else if (i<=8) {
  i=i;
 }
if ((digitalRead(UP) == 1) && (digitalRead(DOWN) == 0)) {

 i++;
 digitalWrite(BE, HIGH);
}
     else if ((digitalRead(UP) == 0) && (digitalRead(DOWN) == 1)){

  i--;
  digitalWrite(BS, HIGH);
 }
 if (i>8){
 digitalWrite(PES, HIGH);
 digitalWrite(BE, HIGH);
  lcd.clear();
```

```
lcd.print("");
lcd.print("Park sature");


}
else if  (i<=8) {
  digitalWrite(PES, LOW);
}
if ((i>8) ||  (digitalRead(UP) == 0)){
  ME.write(90);
}
else if ((i<=8) && (digitalRead(UP) == 1)){


  delay(1000);
lcd.clear();
    ME.write(120);
    delay(450);
    ME.write(90);
    delay(500);
    ME.write(-113);
    delay(450);


lcd.print("Place occupee");
lcd.print(":");
lcd.print(i);
}
if ((i==0) || (digitalRead(DOWN) == 0)) {
  MS.write(90);}


   else if ((i>0) && (digitalRead(DOWN) == 1)) {
```

```
delay(1000);
lcd.clear();
   MS.write(-120);
   delay(450);
   MS.write(90);
   delay(700);
   MS.write(130);
   delay(450);

 lcd.print("Place occupee");
lcd.print(":");
lcd.print(i);
} }
```