الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Ecole Nationale Supérieure de Technologie
Département : Génie Electrique et Informatique Industrielle

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'ingénieur d'état

Spécialité

Electronique option Systèmes Embarqués

Réalisation d'un système pour smart parking basé sur RFID

Réalisé par

SAIDI Walid BELKADI Daoud

Présenté et soutenu publiquement le (06/07/2022)

Composition du Jury :

Président	Mme Malika BOUTERFAS	MCA	ENST
Promoteur	Mme Khadidja ZELLAT	MCB	ENST
Examinateur	Mme Dalila BOUTOUTA	MAA	ENST
Examinateur	M. El Hadi KHOUMERI	MCB	ENST

Remerciement

Nous remercions Dieu qui nous a donné la force et la volonté d'accomplir ce modeste travail.

La réalisation de ce mémoire a été possible grâce à Dieu en premier lieu, et au concours de plusieurs personnes à qui nous voudrions témoigner toute notre reconnaissance.

Nous voudrions tout d'abord adresser toute notre gratitude à notre promotrice, ***Mme B.K Zellat****, pour sa patience, sa disponibilité et surtout ses judicieux conseils.*

Nous tenons également à exprimer notre immense gratitude envers tous les membres du jury, qui ont accepté d'évaluer ce travail.

Le travail sur ce mémoire durant toute une année n'aurait pas été aussi fructueux sans l'aide et le courage précieux de nos parents et toutes nos familles, ainsi que de nos amis. Pour cela, nous les remercions tous pour leur soutien moral, intellectuel, douaa et le courage qu'ils nous ont donné.

Enfin, nous adressons nos sincères remerciements à tous les enseignants, intervenants dans notre formation et à toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé notre chemin et ont accepté de nous rencontrer et répondre à nos questions à tout moment.

À tous ces intervenants, nous présentons nos remerciements, notre respect et notre gratitude.

Table des matières

Listes des figures VII

Listes des tableau IX

Introduction générale 1

I. Chapitre 01 3

Introduction 4

I.1 Définition 4

I.2 Plateformes matérielles 5

I.2.1 Microcontrôleur 5

I.2.2 Processeur 5

I.2.3 Mémoire 5

I.2.4 Les Bus 6

I.2.5 Les périphériques 6

I.2.6 Les entrées/sorties 6

I.3 Différents modèles de cartes à microcontrôleur 7

I.3.1 Cartes Raspberry Pi 7

I.3.2 Cartes Arduino 8

I.3.3 Comparaison entre carte à microcontrôleur et un PC standard 8

I.3.4 Les SOC (System-on-Chip) 9

I.3.5 Exemple d'application (SOC ARM) 9

I.3.6 Autres Systèmes Programmables 10

I.4 Plateforme logicielle 10

I.4.1 Définition 11

I.5 Le Système temps réel 11

I.5.1 Définition 11

I.5.2 Classes des systèmes en temps-réel 12

- Temps-réel strict/dur (hard real-time) 12

- Temps-réel souple/mou (soft real-time) 12
- Le temps réel ferme 12

I.6 Domaines d'application 12

I.6.1 Systèmes embarqués industriels 12

I.6.2 Électronique grand public 13

I.6.3 Objets connectés, IoT (Internet of Things) 13

I.7 Généralités sur les villes intelligentes et le stationnement intelligent 13

I.7.1 La ville intelligente 13

I.7.2 Les composants de la ville intelligente 14

a. Mobilité et déplacements 14

b. Gestion de l'énergie et de l'entretien 15

c. Accès sans-fil 15

d. Caméras de surveillance 15

e. Bornes de recharge de véhicules 16

f. Géolocalisation et applications mobiles 16

I.7.3 Stationnement intelligent 16

Conclusion 16

II. Chapitre 02 18

Introduction 19

II.1 Généralités sur le traitement d'image 19

II.1.1 Définition 19

II.1.2 Les images binaires 19

II.1.3 Les images couleur 20

II.1.4 Systèmes d'acquisition 20

a. Imagerie < grand public > : photographie numérique 20

II.1.5 Caractéristiques d'une image numériques 21

a. Pixel 21

b. Dimension 22

c. Bruit 22

d. Contour 23
II.1.6 Principales étapes de traitement d'une image 24
a. Pré-traitements 24
b. Classes de pré-traitement 24
c. Filtrage 25
d. Types de filtres 25
Linéaires 25
Non linéaires 25
II.1.7 Segmentation 25
Définition 25
II.2 L'internet des objets 26
II.2.1 Introduction 26
II.2.2 Définition 26
II.2.3 Les composants d'internet des objets [17] 26
II.2.4 Les avantages de l'IoT 27
II.2.5 Les inconvénients de l'IoT 27
II.2.6 Domaines d'application d'IoT 27
a. Les transports 28
b. La santé 28
c. Agriculture 28
II.2.7 Technologies de communication 28
Protocole 29
a. Protocole HTTP 29
II.3 Technologie RFID 30
II.3.1 Les différents composants d'un système RFID 30
a. Lecteur RFID 30
b. Le tag RFID 30
c. Fonctionnement de la RFID 31
Conclusion 32

III. Chapitre 03 33
Introduction 34
III.1 Partie Matériel 34
III.1.1 Raspberry Pi 34
a. Spécifications 34
III.1.2 Module camera Raspberry pi 36
a. Caractéristiques 36
III.1.3 Module RFID RC522 37
III.1.4 Carte d'essais (breadboard)/câbles 38
III.2 Partie logicielle et programmation 39
III.2.1 Langage de programmation 39
a. Python 39
b. Les bibliothèques utilisées 40
c. Kotlin 40
III.2.2 Système d'exploitation 40
a. Raspbian OS 40
b. Logiciels 40
• PyCharm 40
• Android studio 41
III.2.3 Gestion de base de données 41
a. Acces en temps reel 42
Conclusion 42
IV. Chapitre 04 43
Introduction 44
IV.1 Description du fonctionnement du projet 44
IV.2 Installation de la camera 46
IV.3 Préparation de l'environnement du traitement 46
IV.4 Les fonctions appliquées 47
IV.5 Présentation de l'application 51

IV.5.1 L'application mobile 51
a. Ecran de démarrage 51
b. Authentification 52
c. Page d'accueil 53
d. Information sur le parking 53
56
e. Page de profil 57
IV.5.2 Les Interfaces homme machine 58
a. L'interface administrateur 58
b. L'interface d'affichage 59
Conclusion 60
Conclusion générale 61
V. Références 63
VI. Annexe 65

Liste des Figures

Figure I-1 : système embarqué typique. 4
Figure I-2 : figure illustrative des composants d'un Raspberry Pi. 7
Figure I-3 : Model carte Arduino. 8
Figure I-4 : SOC ARM. 9
Figure I-5 : Ville intelligente. 14
Figure II-1 : Image binaire. 19
Figure II-2 : Image couleur. 20
Figure II-3 : Capture d'une image par un capteur photographique. 21
Figure II-4 : Lettre "E" en pixels. 22
Figure II-5 : Lettre "E" 22
Figure II-6 : image de dimension 20px*20px. 22
Figure II-7 : Système de traitement d'image 24
Figure II-8 : échange serveur client" HTTP" 29
Figure II-9 : Fonctionnement du tag/lecteur RFID. 31
Figure III-1 : Carte Raspberry Pi 3 35
Figure III-2 : Camera Ip V1.3. 37
Figure III-3 : Module RFID RC522. 38
Figure III-4 : Carte d'essai. 38
Figure III-5 : Logo de Python. 39
Figure III-6 : Logo de PyCharm. 41
Figure III-7 : Logo Android studio. 41
Figure IV-1 : Principe de fonctionnement du système de stationnement intelligent. 45
Figure IV-2 : Image originale prise par la caméra. 46
Figure IV-3 : Localisation des places de stationnement selon leur forme. 46
Figure IV-4 : Image floue 47
Figure IV-5 : Seuillage image flou/originale 48
Figure IV-6 : Filtre Médian. 48
Figure IV-7 : Image dilatée. 49
Figure IV-8 : Surveillance du parking. 50
FigureIV-9 : Ecran de démarrage. 51
Figure IV-10 : Inscription connexion. 52
Figure IV-11 : Parkings disponibles. 53
Figure IV-12 : Informations du parking sélectionné. 54
Figure IV-13: Configuration 55

Figure IV-14: l'information envoyee vers l'application..... 55
Figure IV-15 : Choix de date de réservation..... 56
Figure IV-16 : Informations d'utilisateur..... 57
Figure IV-17 : Interface d'administrateur..... 58
Figure IV-18 : Interface d'entrée. 59

Liste des tableaux

Tableau I.1 : table comparative 9
Tableau I.2: Circuits programmables 10
Tableau II.1: Composants de l'IoT 26
Tableau III.1 : Spécifications de la carte Raspberry 3 34
Tableau III.2 Caractéristique de la caméra 36
Tableau III.3 : Module RFID RC522 37

Liste d'abréviation

RFID	Radio Frequency IDentification
IOT	Internet Of Things
TIC	Technologie de l'Information et de Communication
RAM	Random Acces Memory
CAN	Convertisseur Analogique Numérique
CNA	Convertisseur Numérique Analogique
OS	Système d'Exploitation
GES	Gaz à Effet de Serre
CCD	Charge Coupled Device
RGB	Red Green Blue
TFD/DFT	Transformé de Fourier Directe/Direct Fourier Transformer
DCT	Discret Cosinus Tranform (cosinus discrète)
DWT	Discret Wave Transform
FPGA	Field Programmable Gate Array
IA	Intelligence Artificielle
GPS	Global Position System
ISO	International Standardization Organisation
UIT-T	Union Internationale des Télécommunications
OSI	Open Systems Interconnection
TCP	Transmission Control Protocol
IP	Internet Protocol
HTTP	HyperText Transport Protocol
Mo	Mega Octet
GPIO	General Purpose Input/Output
RF	RadioFréquence
CSI	Camera Serial Interface
SPI	Serial Peripheral Interface
ID	Numero d'Identification

Introduction générale

Le mode de vie du monde change rapidement et changera toujours. La technologie a dépassé la satisfaction des besoins militaires et des puissances mondiales, elle touche tous les secteurs qui entourent l'humain et tout ce qui lui est attaché. Partant du transport, de la santé, de la sécurité, de la communication, etc.

Par ailleurs, les développeurs et les constructeurs dans ce domaine continuent à laisser leurs traces dans notre vie et l'objectif devient maintenant de rendre tout intelligent : tout marche sans intervention manuelle ou force musculaire en assurant la sécurité, partant du téléphone intelligent puis du véhicule intelligent, vers la maison intelligente, le cartier puis la ville, le pays ; arrive-t-on à un monde intelligent ? Pourquoi pas ! Tout est possible avec cette évolution qui croit exponentiellement.

Dans le but de traiter puis de résoudre le problème d'encombrement dans les grandes villes, à titre d'exemple « Alger » qui est connue par son blocage routier insupportable, nous constatons que parmi les facteurs principaux provoquant ce problème est la mauvaise gestion des stationnements. Un conducteur passe plus de 20 min cherchant un emplacement vide pour qu'il puisse stationner. Cette durée augmente proportionnellement avec le nombre de véhicules qui circulent dans la ville et cherchent un emplacement de stationnement. Un problème de pollution se pose également : les automobilistes qui tournent dans la ville pour chercher une place de parking polluent la ville sans se rendre compte. Le temps de recherche entraine aussi l'énervement des usagers, ce qui n'aide pas à trouver une place

Plusieurs solutions ont déjà été mises en œuvre. Parmi elles on compte le stationnement automatique ou le stationnement intelligent. Cette technique est basée sur un ensemble de capteurs qui sont connectés entre eux et une unité de contrôle (contrôleur) qui fait les calculs nécessaires et assure la communication entre tout élément du système par le protocole adéquat. Celui que nous sommes en train de réaliser est spécifié par l'exploitation des caméras de surveillance au lieu des capteur analogiques habituels dans le but d'obtenir les emplacements disponibles par un calculateur qui traite les données (image, vidéo) reçues par la caméra de surveillance et de les transférer à une application mobile à travers un service nommé « cloud ». Le contrôleur doit assurer le bon traitement de données reçues par la caméra et aussi le système d'accès basé sur le module RFID, et l'interface homme- machine conçue pour l'orientation de l'utilisateur. Pour cela, il nous faut exploiter nos connaissances sur les microcontrôleurs, et surtout la notion du traitement d'image, ainsi que celle de l'internet des objets. Nous avons conçu ce mémoire en quatre chapitres, le premier qui sera une introduction aux systèmes embarqués, les

villes intelligentes et le stationnement intelligent. Le deuxième chapitre sera consacré aux notions fondamentales de traitement d'image, l'intelligence artificielle, l'internet des objets et la technologie RFID, ce chapitre combine les notions théoriques utiles dont on a besoin afin de réaliser ce projet sur terrain. Ensuite ; le troisième chapitre sera consacré aux matériel et environnements logiciels exploités. Enfin ; dans le dernier chapitre nous allons montrer les tests effectués et les résultats obtenus.

I. Chapitre 01

Généralités sur les systèmes embarqués

Introduction

Les systèmes embarqués nous accompagnent au quotidien et nous offrent des services qui nous facilitent la vie : smartphones, drones, robots jouets, appareils ménagers, systèmes d'aide à la conduite automobile, dispositifs de surveillance à distance du domicile et bien d'autres encore.

Ils sont présents dans le domaine de la santé, la domotique, la défense, le contrôle de processus industriel, les transports, les infrastructures critiques comme le nucléaire ou l'énergie, les loisirs ou encore l'agriculture.... Ils sont pour la plupart fortement connectés.

I.1 Définition

Un système embarqué est un système autonome, généralement hautement connecté. Il se concentre sur des tâches spécifiques, fonctionne souvent en temps réel et mélange des éléments matériels et logiciels. C'est ce qu'on appelle un "embarqué" (on l'appelle aussi un système "embarqué") car il est intégré à l'appareil physique qui lui fournit le contrôle ou les commandes, et il partage ses contraintes environnementales [1].

La figure I.1 représente l'architecture des systèmes embarqués.

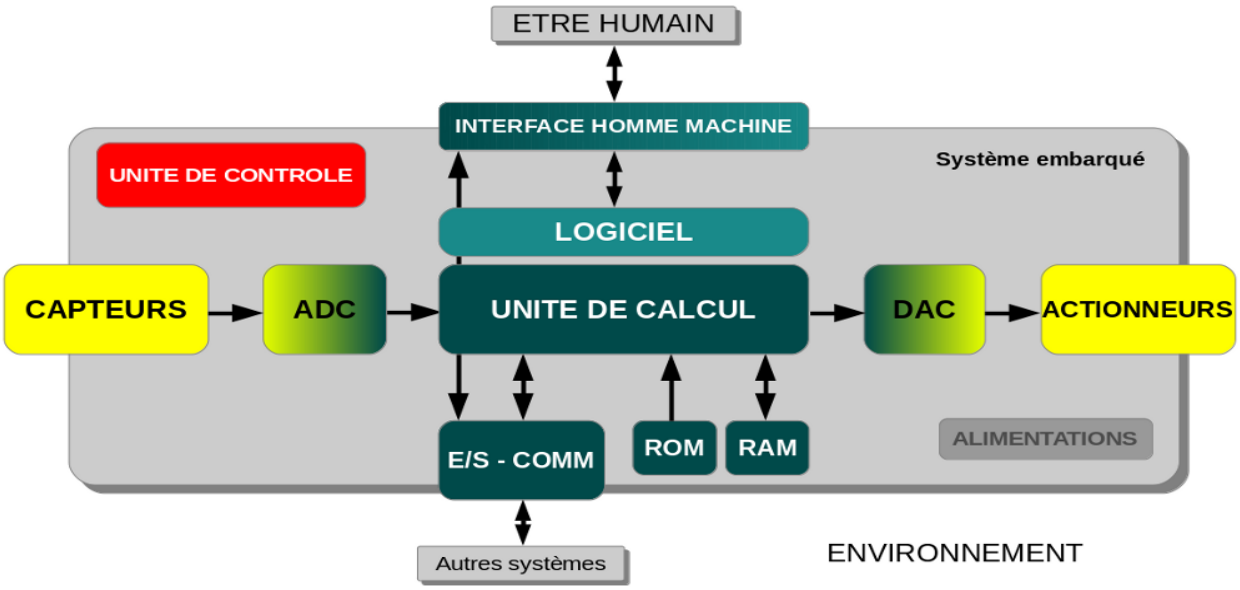


Figure I-1 : système embarqué typique.

I.2 Plateformes matérielles

I.2.1 Microcontrôleur

Est une unité de traitement optimisée pour les entrées/sorties. Typiquement, un microcontrôleur est directement associé à plusieurs types d'entrées/sorties, et il ne nécessite pas l'ajout de cartes d'extension supplémentaires. Par conséquent, il est plus compact qu'un microprocesseur équipé d'une carte de capture, mais est moins efficace en termes de calcul

L'unité de calcul est caractérisée par la fréquence d'horloge en Hertz (Hz). La fréquence correspond au nombre de cycles effectués par l'unité de calcul par seconde. Chaque instruction de bas niveau exécutée par l'unité de calcul prend un à plusieurs cycles d'horloge.

I.2.2 Processeur

Un microprocesseur est une unité de traitement optimisée pour le calcul. Généralement plus rapide qu'un microcontrôleur, c'est le cœur des micro-ordinateurs [2].

Il sert à exécuter les instructions et traiter les données des programmes. Il s'agit d'un élément semi-conducteur au silicium.

I.2.3 Mémoire

Les instructions d'un programme, ainsi que les données et le contexte d'exécution du programme sont stockés dans la mémoire centrale (mémoire vive comme la RAM pour Random Access Memory, ou mémoire FLASH plus lente que la RAM mais rémanente)

Une mémoire se caractérise par sa taille, la taille des mots mémoires (taille de données que le processeur et la mémoire échangent à chaque accès) et son temps d'accès. Le temps d'accès est donné en fréquence, soit en nombre d'accès par seconde.

Dans le cas des microcontrôleurs, la fréquence d'accès à la mémoire centrale est souvent la même que la fréquence du microcontrôleur : pour charger ou déposer un mot en mémoire, le processeur n'a pas à attendre plus d'un cycle. Dans le cas des microprocesseurs, la mémoire centrale est souvent 5 à 10 fois plus lente que le microprocesseur, cela implique qu'une communication avec la mémoire peut durer 5 à 10 cycles d'horloge du microprocesseur. Ainsi, lors du déroulement d'un programme le

chargement d'une instruction à partir de la mémoire centrale, ou l'accès à une donnée, peut durer plusieurs cycles processeur. Cependant, comparée à la mémoire de masse, comme le disque dur par exemple, avec un temps d'accès de l'ordre de 10 millisecondes, une mémoire centrale, pouvant fonctionner à plusieurs centaines de mégahertz (temps d'accès de l'ordre de quelques nanosecondes) fait figure de mémoire extrêmement rapide [2].

I.2.4 Les Bus

Un bus est un ensemble de fils qui transportent des informations codées en binaire. Chacune de ces lignes se voit attribuer un poids binaire. C'est par ces lignes que s'effectuent les échanges entre les différents éléments du système. Il existe 3 types de bus : bus de données, bus d'adresses et bus de contrôle.

I.2.5 Les périphériques

Sont des circuits électroniques intégrés au microcontrôleur capable d'effectuer des tâches spécifiques. On peut mentionner entre autres :

- Les convertisseurs analogiques-numériques (CAN) (donnent un nombre binaire à partir d'une tension électrique) ;
- Les convertisseurs numériques-analogiques (CNA) (effectuent l'opération inverse)
- Les générateurs de signaux à modulation de largeur d'impulsion (MLI, ou en anglais, PWM pour Pulse WidthModulation)
- Les timers (compteurs d'impulsions d'horloge interne ou d'événements externes)
- Les chiens de garde (Watch_dog)
- Les comparateurs (comparent deux tensions électriques)
- Les contrôleurs de bus de communication (UART, IIC, SSP, CAN, FlexRay, USB, Ethernet, etc.)

I.2.6 Les entrées/sorties

Permettent au microcontrôleur de communiquer avec le monde extérieur. C'est donc là que vont être connectés les claviers, afficheurs, poussoir, moteurs, relais, etc. que va utiliser l'application.

I.3 Différents modèles de cartes à microcontrôleur

Cette catégorie de cartes renferme un microcontrôleur complet ; c'est à dire l'association en un seul circuit d'une unité centrale de microprocesseur, de mémoire morte, de mémoire vive, de mémoire EEPROM, d'une interface d'entrée/sortie série et de toute la logique nécessaire pour faire fonctionner tout cela. Il existe plusieurs types de cartes à microcontrôleur, nous citons à titre d'exemple juste les deux suivants : Arduino et Raspberry Pi.

I.3.1 Cartes Raspberry Pi

Le Raspberry Pi est un mini-ordinateur muni d'un processeur ARM, il a été conçu par le créateur de jeux vidéo David Braden. Au début, le Raspberry Pi ne se déclinait qu'en 2 modèles : le A et le B. Puis s'en est suivi une amélioration de ces 2 modèles, à savoir le A+ et le B+. Depuis février 2015, un 5ème modèle a été commercialisé sous le nom de Raspberry Pi 2. Maintenant il existe un modèle plus récent qui est le Raspberry Pi 3 et Raspberry Pi 4 [3].

La figure I.2 représente les différents composants de la carte Raspberry Pi.

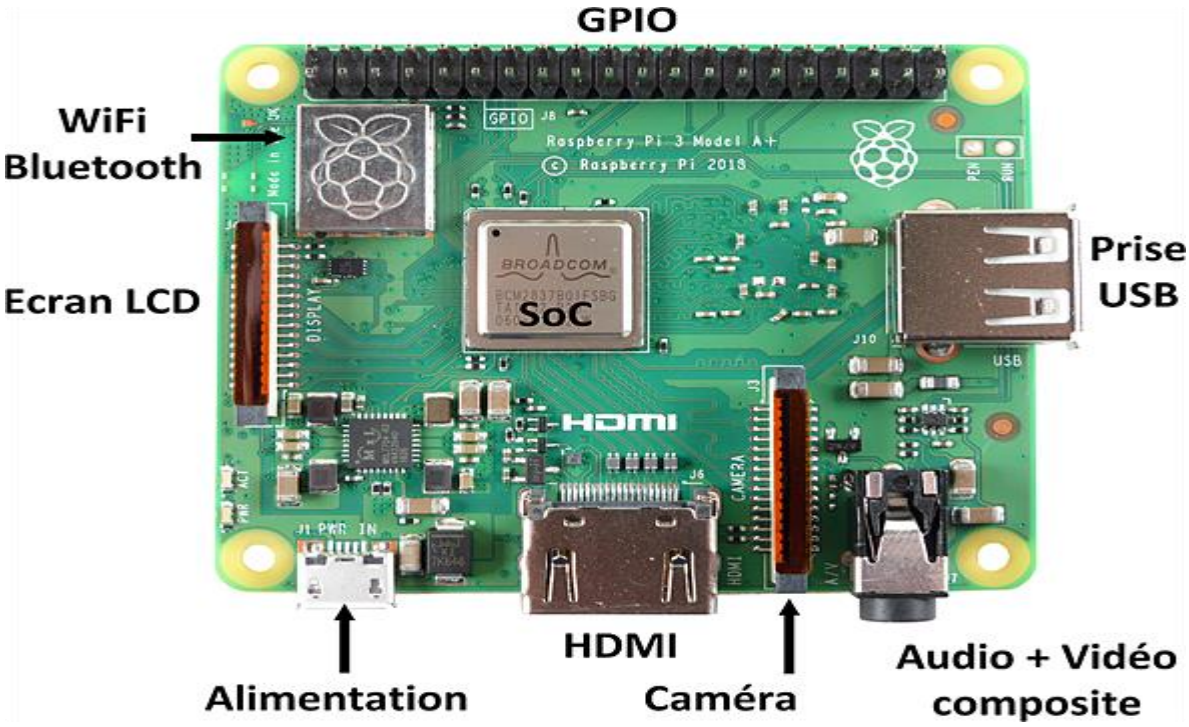


Figure I-2 : figure illustrative des composants d'un Raspberry Pi.

I.3.2 Cartes Arduino

Les cartes Arduino sont dotées de microcontrôleurs faciles à programmer et d'un grand nombre d'entrées et de sorties. Il existe plusieurs cartes Arduino qui diffèrent par la puissance du microcontrôleur ou la taille et la consommation de la carte. Le choix du type de carte Arduino est basé sur les besoins de votre projet. La carte Arduino UNO (voir Figure I.3) est la carte la plus couramment utilisée et un bon choix pour les débutants. Toutes les cartes Arduino sont programmées en C++ à l'aide d'un logiciel de programmation gratuit et open source fournie par Arduino.

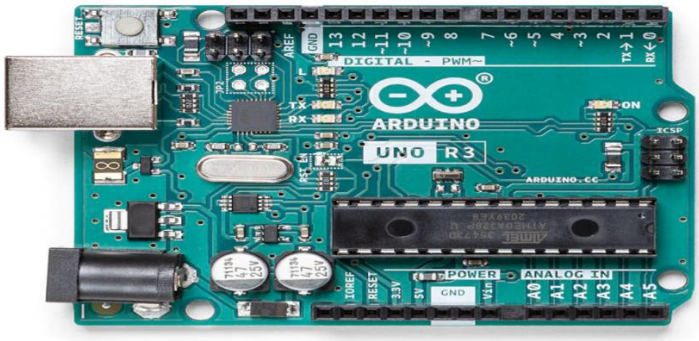


Figure I-3 : Model carte Arduino.

I.3.3 Comparaison entre carte à microcontrôleur et un PC standard

Pour le but de faire la différence entre les deux types de carte qu'on a étudié et un Pc standard on donne la comparaison indiquée dans le tableau I.1

Le choix du Raspberry Pi est effectué pour 2 raisons le premier c'est que la vitesse de traitement est plus performante par rapport à la carte Arduino, le volume est le deuxième critère de choix vu que le PC standard est plus volumineux cependant la carte Raspberry qui est similaire à une carte de crédit qui ne nécessite pas un espace important

Tableau I.1 : table comparative.

	PC standard	Raspberry PI 3 B+	ARDUINO Uno
Fréquence	2 GHz	1.4 GHz	16 MHz
Core/Architecture	4/64 bits	ARM-quadri/64 bits	1/8 bits
Consommation	100 à 500 W	4 W	$<$ 1 W
Entrées/Sorties	/	40 GPIO	6 Analog/13 Digital
Ports extension	USB, PCI...	USB, SPI, I2C, RS232, ETH	SPI, I2C, RS232
RAM	4 Go	1 Go	2 ko

I.3.4 Les SOC (System-on-Chip)

Sont des système complet embarqué sur une seule puce ("circuit intégré") sont regroupées les fonctionnalités du microprocesseur, de divers périphériques (contrôleur graphique, réseau, communications série, SPI, RS232, etc.), ainsi on peut intégrer une quantité plus ou moins importante de mémoire statique, dynamique, flash, ROM, PROM, EPROM, EEPROM), des dispositifs (capteurs) mécaniques, opto-électroniques, chimiques ou biologiques. Ou des et une puce radio pour les communications sans fil (Bluetooth, Wi-Fi) ou cellulaires.

I.3.5 Exemple d'application (SOC ARM)

Architecture ARM (Voir figure I.4) qui est surtout connu pour ses systèmes sur puce (SoC), intégrant sur une seule puce : microprocesseur, processeur graphique (GPU SIMD , et contrôleur de périphériques. Ceux-ci sont présents dans la majorité des smartphones et tablettes (Toutes les puces contenant des processeurs ou les microcontrôleurs ARM).



Figure I-4 : SOC ARM.

I.3.6 Autres Systèmes Programmables

- Systèmes en logique programmée et/ou en logique programmable

Les circuits logiques programmables ou matrices logiques programmables sont des circuits intégrés logiques qui peuvent être reprogrammés après fabrication. Il se composent de nombreuses unités logiques de base qui peuvent être assemblées librement. Ils sont connectés de manière permanente ou réversible par programmation pour réaliser la fonction numérique souhaitée. Fait intéressant, la même puce peut être utilisée dans de nombreux systèmes électroniques différents.

Des exemples sont mentionnés dans de tableau I.2

Tableau I.2: Circuits programmables.

FPGA	• (*Field-programmable gate array* réseau de portes programmables *in situ*) ;
PLD	• (*Programmable logic device* circuit logique programmable) ;
EPLD	• (*Erasable programmable logic device* circuit logique programmable et effaçable) ;
CPLD	• (*Complex programmable logic device* circuit logique programmable complexe) ;
PAL	• (*Programmable array logic* réseau logique programmable)

I.4 Plateforme logicielle

Sans le système d'exploitation (SE, Operating System, OS), le système embarqué n'est qu'une boîte inutilisable. Le système d'exploitation fait l'interface entre le matériel et le logiciel : il présente aux programmes une machine virtuelle, relativement indépendante du matériel sous-jacent.

I.4.1 Définition

Un système d'exploitation est défini comme un ensemble de programmes qui coopèrent à la gestion des ressources d'un ordinateur et fournit aux utilisateurs une machine virtuelle.

Pour les systèmes très réduits, comme les microcontrôleurs, il est fréquent d'employer des OS (Operating System – systèmes d'exploitation) très légers – voire aucun OS et de n'intégrer que le code développé spécifiquement pour le projet. Ces OS simplifiés, qui se résument souvent à un ordonnanceur et des mécanismes de synchronisation entre tâches, sont généralement propriétaires et vendus par le fabricant du microcontrôleur accompagnés d'un environnement de développement (compilateur, etc.) spécifique.

I.5 Le Système temps réel

On entend souvent parler de Temps Réel dès que l'on parle de système embarqué. En fait, un système embarqué doit généralement respecter des contraintes temporelles fortes *(Hard Real Time*) et l'on y trouve enfoui un système d'exploitation ou un noyau Temps Réel (*Real Time Operating System,* RTOS). Le Temps Réel est un concept un peu vague et chacun a sa propre idée sur la question.

I.5.1 Définition

Un système en temps réel doit réagir aux stimuli de l'objet contrôlé (ou de l'opérateur) dans des intervalles de temps dictés par son environnement. L'instant auquel un résultat doit être produit est appelé une date limite [4], ou bien on pourrait le définir comme : *Un système dont l'information après acquisition et traitement reste encore pertinente.* Plus précisément, cela veut dire que dans le cas d'une information arrivant de façon périodique (sous forme d'une interruption périodique du système), les temps d'acquisition et de traitement doivent rester inférieurs à la période de rafraîchissement de cette information.

Pour cela, il faut que le noyau ou le système Temps Réel soit :

- Déterministe : les mêmes causes produisent les mêmes effets avec les mêmes temps d'exécution.

- Préemptif : la tâche de plus forte priorité prête à être exécutée doit toujours avoir accès au processeur. Ce sont là des conditions nécessaires mais malheureusement pas suffisantes pour affirmer qu'un système embarqué est Temps Réel par définition.

I.5.2 Classes des systèmes en temps-réel

- **Temps-réel strict/dur (hard real-time)**

Le non-respect d'une contrainte de temps a des conséquences graves (humaines, économiques, écologiques) : besoin de garanties.

Ex : Applications de contrôle dans l'avionique, le spatial, le nucléaire, contrôle de production de matériaux toxiques.

- **Temps-réel souple/mou (soft real-time)**

Le résultat a une utilité même après l'expiration du délai, le délai est qualifié de souple, on peut tolérer le non-respect occasionnel d'une contrainte de temps (garanties probabilistes).

- Ex : Applications multimédia grand public (vidéo à la demande, TV)
- Applications interactives : temps de réaction invisible à l'usager.

- **Le temps réel ferme**

C'est un système pour lequel on peut tolérer quelques retards (peu souvent) dans la délivrance du résultat. C'est par exemple le cas d'un système multimédia type téléphone mobile.

I.6 Domaines d'application

Bien que les systèmes embarqués prennent des formes très différentes et représentent une activité transverse de l'informatique, on peut les ranger dans quelques catégories couramment rencontrées :

I.6.1 Systèmes embarqués industriels

Ces systèmes fonctionnent essentiellement dans le pilotage et la supervision des processus de production industrielle. Le cœur de la transaction est généralement exécuté

par des composants dédiés (système d'acquisition, FPGA, microcontrôleur, ASIC parlant à travers des protocoles de communication adaptés (CAN, SPI, etc.)), le processeur est intégré sur une carte embarquée qui assure des fonctions auxiliaires (réseau, interface utilisateur, stockage, etc.).

I.6.2 Électronique grand public

Les systèmes embarqués sont omniprésents dans les équipements technologiques actuels (smartphones, tablettes, liseuses...) et gagnent sans cesse du terrain dans de nombreux appareils domestiques (électroménager, domotique, etc.). L'interface utilisateur joue un rôle crucial dans le choix du public, les systèmes d'exploitation et environnements logiciels doivent assurer une cohérence d'utilisation entre les différentes plates-formes. C'est le domaine de prédilection des environnements comme Android, iOS, Tizen.

I.6.3 Objets connectés, IoT (Internet of Things)

Cette nouvelle branche des systèmes embarqués est en plein essor depuis plusieurs années. Ce modèle repose généralement sur la connexion des objets du quotidien à une plateforme de cloud computing (via une box internet locale). Les opérateurs collectent d'énormes quantités de données auprès de leurs utilisateurs, qu'ils peuvent traiter à l'aide d'algorithmes spécifiques et innovants (issus de la recherche dans le domaine du machine learning) pour en extraire des informations pertinentes (statistiques, prévisions, optimisation, etc.) pour les utilisateurs, les villes intelligentes, maisons et bâtiments intelligents, usines, transports, santé (télésurveillance des patients), télésurveillance, sécurité (caméras IP), télécommande, etc.

I.7 Généralités sur les villes intelligentes et le stationnement intelligent

I.7.1 La ville intelligente

Une ville intelligente est une zone urbaine qui utilise différents capteurs de collecte de données électroniques pour fournir des informations permettant de gérer efficacement les ressources et les actifs. Il s'agit d'une ville qui investit dans son capital humain et social, et privilégiant les technologies de l'information et de la communication (T.I.C.) afin de répondre aux problèmes publics, de réaliser un développement durable et favoriser une meilleure interaction avec ses citoyens et garantir à ses habitants d'amélioration de

la qualité de vie. Chaque ville numérique n'est pas nécessairement intelligente, mais chaque ville intelligente a des composants numériques [5].

Une ville peut être dite "intelligente" lorsque les investissements dans le capital humain et social et les infrastructures de communication traditionnelles et modernes alimentent une croissance économique durable et une qualité de vie élevée, avec une gestion avisée des ressources naturelles, par le biais d'une gouvernance participative [6].

I.7.2 Les composants de la ville intelligente

La technologie n'est qu'un facteur de l'équation d'un ville dite intelligente. Pour cela l'interconnexion entre les différents éléments de cette ville est la clé de cette nouvelle technologie.

Par le bon positionnement stratégique des capteurs et de tout type d'instrument de mesure dans un territoire donné, il est possible de récolter et d'analyser plusieurs éléments importants qui contribueront à la meilleure gestion des infrastructures (mobilité, consommation, gestion des risques, alertes diverses, etc).

La figure I.6 montre les différents composants d'une ville intelligente.



Figure I-5 : Ville intelligente.

a. Mobilité et déplacements

À l'heure où tout le monde parle du mot transport durable, et compte tenu des efforts des municipalités pour améliorer les déplacements sur leur territoire, certains capteurs peuvent planifier le trafic routier, les horaires de bus, les embouteillages potentiels et les prévisions en fonction des activités quotidiennes. Différents effets des événements, espaces de travail ou d'autres activités sur la circulation. La nouvelle

application guide les utilisateurs en temps réel dans la meilleure façon de se déplacer, rendant leur trajet plus fluide et réduisant les embouteillages. De même, l'emplacement et le nombre de places de stationnement gratuites dans la zone peuvent être facilement déterminés, augmentant ainsi l'accessibilité des services aux usagers et des diverses activités commerciales, sportives ou événementielles [7].

b. Gestion de l'énergie et de l'entretien

L'énergie consommée par les infrastructures urbaines telles que les luminaires de rue peuvent représenter jusqu'à 40 % de l'énergie totale consommée par la ville. Les dépenses liées à la planification de l'entretien et à la gestion de ses infrastructures sont également élevées, sans compter la surveillance de durée de vie et des GES émis par les activités entourant cette gestion.

L'objectif est de réduire considérablement les coûts de l'énergie et de la pollution lumineuse, mais aussi d'accroitre la sécurité des usagers grâce à des interventions plus rapides et mieux planifiées sur les luminaires défectueux.

Le même principe et le même système permettront également d'agir de la même façon sur l'ensemble des luminaires et des infrastructures de la ville.

c. Accès sans-fil

Puisqu'un réseau de communication est déjà déployé sur l'ensemble de la ville pour les besoins identifiés ci-haut, il devient d'autant plus facile de permettre, au choix de la ville, l'accès Internet gratuit à plusieurs endroits identifiés, comme les parcs et autres endroits publics. L'accès Internet étant très facilement contrôlable et les données d'utilisation récoltées.

d. Caméras de surveillance

Nul besoin de parler de la pertinence de l'installation de certaines caméras de surveillance. Rappelons simplement qu'il devient encore une fois plus facile de les intégrer, tout en évitant de payer des frais de gestion et de doubler les logiciels et leurs frais puisqu'elles seront intégrées au centre de données partagées.

e. Bornes de recharge de véhicules

Avec la commercialisation de la voiture électrique qui croît d'année en année, il est possible d'utiliser le même réseau pour installer des bornes, les gérer, les analyser et même ultimement vendre l'énergie consommée de façon simple et efficace.

f. Géolocalisation et applications mobiles

L'ensemble des capteurs installés et rattachés au centre de données partagées est géoréférencé, ce qui permet une surveillance de l'ensemble des infrastructures de façon intuitive et visuelle. Ces données non sensibles jouent un rôle important pour le développement d'applications utiles et évolutives pour la Ville et la collectivité. Pour maximiser l'utilisation des données géoréférencées partagées, la Ville a donc l'opportunité de rendre disponibles toutes les infrastructures et autres données géoréférencées sur son territoire.

I.7.3 Stationnement intelligent

Le stationnement intelligent est une solution de stationnement qui inclut de nouvelles technologies intégrées dans l'espace urbain. Elles sont installées sur les places de parkings ou juste à côté et au niveau des entrées et sorties du parking. Ces installations récoltent des données et les transmettent généralement à une application mobile intelligente disponible aux conducteurs. Ceux-ci ont alors la possibilité d'explorer toutes les options de stationnement à proximité qui s'offrent à eux. Ils peuvent consulter la disponibilité, les tarifs ou encore la localisation de places de parking aux alentours afin de faire facilement et rapidement leur choix de stationnement [8].

Conclusion

Dans ce chapitre introductif nous avons balayé plusieurs notions sur les systèmes embarqués et les villes intelligentes.

L'être humain est intelligent et son intelligence provient d'un organe miraculeux qui est le cerveau, ce dernier est le responsable de tous nos actes, nos gestes, nos mouvements et nos décisions. Tout système intelligent besoin d'un centre de traitement qui modélise le rôle du cerveau humain. Sans un processeur, on ne pourra plus dire intelligent sur un tel système. Ce dernier est le responsable de toute fonction réalisée par un tel actionneur.

Bien sûr ce processeur ne fonctionnera pas tout seul, il lui faut des mémoires pour stocker les données, des registres et des bus de communication etc.

Nous avons bien détaillé l'architecture d'un système embarqué, donné un large rappel sur les systèmes des villes intelligentes qui sont liées à un centre de traitement basé sur ces plateformes ; un processeur et microcontrôleur.

II. Chapitre 02

Notions fondamentales

Introduction

Dans ce chapitre nous allons introduire plusieurs notions qui feront l'objet de notre travail dans la société, l'industrie, la recherche scientifique etc.

Vu que notre travail est base sur le traitement d'image nous allons nous citer le principe de cette technique et ces éléments fondamentaux, par la suite nous allons parler sur la technologie RFID et l'internet des objets. Finir par l'intelligence artificiel en déployant son évolution ainsi que son importance dans notre projet.

II.1 Généralités sur le traitement d'image

II.1.1 Définition

Le traitement d'images est né de l'idée et de la nécessité de remplacer l'observateur humain par la machine. L'image ou les signaux provenant des capteurs ont alors été numérisés pour pouvoir être traités par l'ordinateur ou d'une autre façon c'est un Processus de modifier des images pour les améliorer ou les rendre exploitables par observateur humain ou un système électronique [9].

II.1.2 Les images binaires

C'est l'exemple des images les plus simples : un pixel peut prendre uniquement les valeurs noir ou blanc. C'est typiquement le type d'image que l'on utilise pour numériser du texte quand celui-ci est composé d'une seule couleur. (Voir figure II.1)



Figure II-1 : Image binaire.

II.1.3 Les images couleur

L'espace couleur est base sur la synthèse des couleurs, c'est-à-dire que le mélange de trois composantes donne une couleur. Un pixel est codé par trois valeurs numériques. La signification de ces valeurs dépend du type de codage choisi. Le plus utilisé pour le maniement des images numériques est l'espace couleur ! Rouge-Vert-Bleu "(R,V,B) (RGB en anglais). La restitution des couleurs sur écran utilise cette représentation. C'est une synthèse additive. [10]

La figure II.2 représente une image couleur



Figure II-2 : Image couleur.

II.1.4 Systèmes d'acquisition

Le principe de l'acquisition d'une image numérique, quelle qu'elle soit, consiste à transformer des quantités physiques (ou chimiques) en nombres que l'on stockera dans des tableaux.

L'œil est le système d'acquisition le plus sophistiqué que nous possédons. Le cerveau est l'ordinateur qui traite les données acquises et son fonctionnement est loin d'être compris. Toute méthode d'acquisition et de traitement des images ne peut que s'inspirer du système visuel [10].

a. Imagerie < grand public > : photographie numérique

Sur les appareils photographiques numériques, le film est remplacé par un capteur. Composé de plusieurs millions de cellules photosensibles recouvertes d'un filtre coloré rouge, vert ou bleu afin de restituer la couleur, le capteur CCD permet de transformer l'énergie lumineuse en signal électrique. Un convertisseur analogique/numérique se charge par la suite de convertir ce signal en données binaires. Ces données sont alors conservées

sur le support de stockage de l'appareil, représenté le plus souvent par une carte mémoire amovible. Afin de faciliter l'archivage des images, celles-ci sont compressées. Ce mode de compression peut être configuré avant la prise de vue et déterminera le nombre d'images qui pourront être stockées sur la carte. (Voir figure II.4)

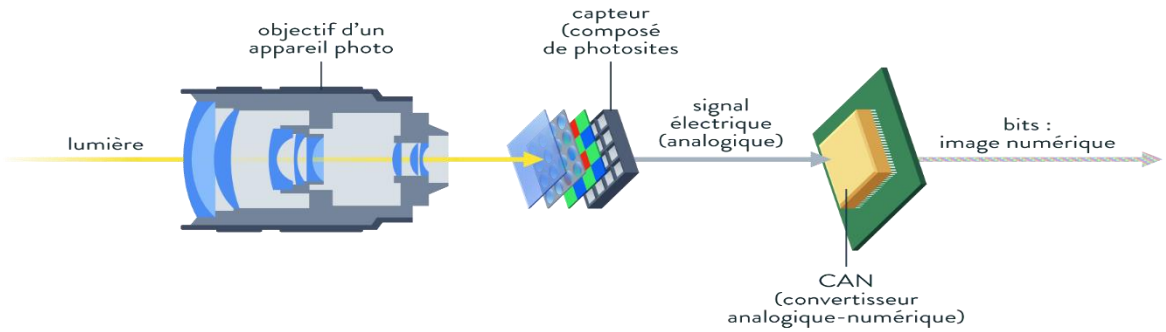


Figure II-3 : Capture d'une image par un capteur photographique.

II.1.5 Caractéristiques d'une image numériques

a. Pixel

Son nom provient de la locution anglaise « *Picture element* », qui signifie « élément d'image » [10], souvent abrégé p ou px, on peut le constater comme la plus petite composante d'une image comme on peut le définir par l'unité de base permettant de mesurer la définition d'une image.

Une image avec plus de détails implique un nombre plus de pixel qui la définit. Ce nombre de pixel va nous guider vers une autre notion dite « résolution ».

La figure II.6 à droite représente une image de la lettre E alors que celle qui est à gauche (figure II.5) représente la même lettre cette fois-ci en pixels.

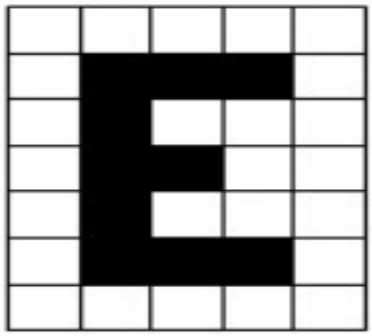


Figure II-4 : Lettre "E" en pixels.



Figure II-5 : Lettre "E".

b. Dimension

La définition d'une image correspond à sa dimension. C'est le nombre de points (ou pixels) que comporte une image numérique en largeur et en hauteur (le nombre de colonnes et nombre de lignes).

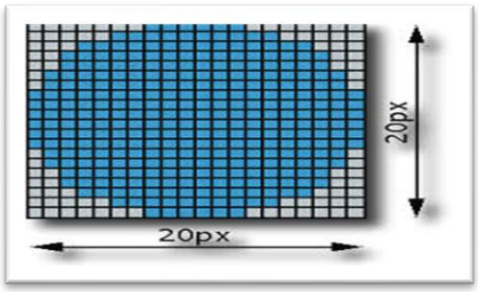


*Figure II-6 : image de dimension 20px*20px.*

c. Bruit

Dans toute image numérique, les valeurs de gris ou de couleur observées présentent une incertitude. Cette incertitude est due aux aléas du comptage des photons arrivant sur chaque capteur. Les valeurs de couleur mesurées sont perturbées car les capteurs reçoivent des photons parasites et subissent des fluctuations électrostatiques lors de leurs charges et décharges. Quand un capteur reçoit beaucoup de photons venant d'une scène bien éclairée, les parasites sont négligeables par rapport au flux de vrais photons. Mais, même dans une photo d'exposition suffisante, les pixels sombres reçoivent très peu de photons et sont donc bruité «. Visuellement, on distingue en général deux types de bruit d'image qui s'accumulent [9]

d. Contour

On parle sur le contour d'image les zone de forte variation de luminance, plusieurs modèles mathématiques le définirent afin de le détecter, c'est donc toute discontinuité qui n'est pas nécessairement située à la frontière.

La détection du contour est une technique qui consiste à transformer l'image en un ensemble de courbes pas forcément fermées formants des frontières significatives de l'image.

Deux approches souvent utilisées :

- Approximation du Gradient : son principe basé sur le calcul de gradient
- Approximation de Laplacien

Soit l'image $f(x,y)$;

Le gradient de f : $\overrightarrow{\nabla_f} = \begin{pmatrix} \frac{df}{dx} \\ \frac{df}{dy} \end{pmatrix}$

L'algorithme d'extraction des maximums locaux consiste à comparer la norme du gradient au site M [i, j] à celle de ses deux voisins M1 et M2 situés dans la direction du gradient à une distance unitaire de part et d'autre de M. [9]

II.1.6 Principales étapes de traitement d'une image

Un système de traitement d'image passe par les étapes suivantes

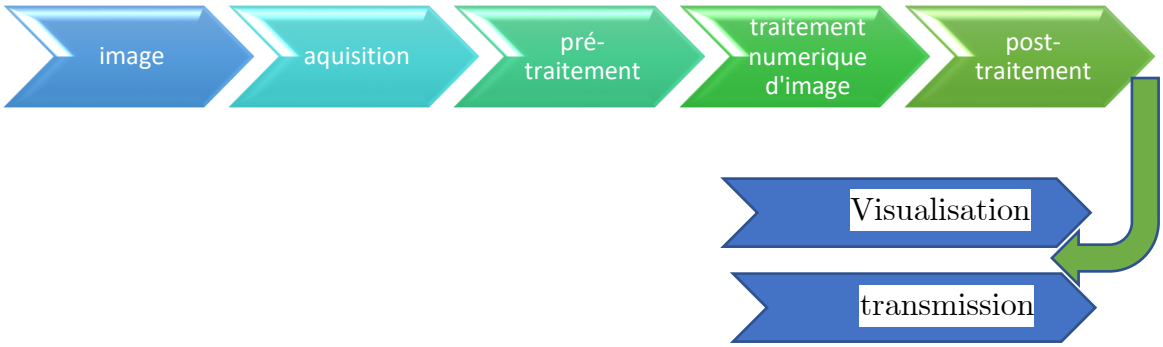


Figure II-7 : Système de traitement d'image.

a. Pré-traitements

Le pré-traitement sert à préparer les images brutes à des traitements postérieurs ou à l'utilisation directe (affichage, transmission, stockage, impression...), il est considéré comme un traitement bas niveau parce qu'il manipule les images sous leurs formats basiques. Cette technique est mise en œuvre pour les problématiques suivantes :

- Renforcement de certaines propriétés locales dans les images
- Correction des distorsions géométriques ou dégradations visuelles.
- Elimination des informations inutiles quant aux certaines applications.

b. Classes de pré-traitement

- Transformations ponctuelles : les 2 images ont les mêmes dimensions. La valeur d'un pixel de l'image prétraitée ne dépend que de celle de son homologue dans l'image originale.
  Exemples : seuillage, ajustement du contraste
- Transformations locales (ou de voisinage) : la valeur d'un pixel de l'image prétraitée dépend de celle du pixel homologue dans l'image originale et son voisinage.
  Exemples : filtrage, extraction de contours

- Transformations globales : la valeur d'un pixel de l'image prétraitée est calculée en fonction de tous (ou la majorité) des pixels de l'image originale. C'est le cas des transformées.

  Exemples : DFT (Fourier), DCT (cosinus discrète), DWT (ondelette).

c. Filtrage

Rentre essentiellement dans le cadre de restauration des images et sert à atténuer ou supprimer le bruit causé par : les conditions de prise d'images, le capteur, l'échantillonnage/quantification, contenu de scène.

d. Types de filtres

Il y on a plusieurs, nous allons citer les filtres basiques du traitement d'image et qui nous ont permis de réaliser notre système.

Linéaires : régis par des fonctions linéaires, chaque pixel est remplacé par une combinaison linéaire des valeurs pondérées de lui-même et ses voisins.

Non linéaires : régis par des relations empiriques ou des fonctions non linéaires.

Le filtre médian est l'un des filtres que nous avons exploité dans notre application.

II.1.7 Segmentation

Segmentation d'images est une tâche importante dans tout processus de traitement et analyse d'images. Cette tâche est présente dans plusieurs applications de vision par ordinateur telles que l'imagerie médicale, l'analyse vidéo, la télédétection, l'aide à l'exploitation des systèmes de vidéo-surveillance dans les transports, etc.

Définition

La segmentation d'images couleur est un traitement de bas niveau qui demeure un problème ouvert en analyse d'images. Cette tâche a pour but de créer une partition d'une image I en sous-ensembles disjoints et connexes, appelés régions et possédant des attributs communs. Au sein de cette partition, les régions doivent respecter des critères d'homogénéité et de connexité [11].

II.2 L'internet des objets

II.2.1 Introduction

L'Internet des objets (IoT) fait généralement référence à un réseau d'objets hétérogènes interconnectés (capteurs, actionneurs, dispositifs intelligents, objets intelligents, RFID (Radio Frequency Identification), ordinateurs embarqués, etc.) adressable de manière unique, basé sur des protocoles de communication standard.

II.2.2 Définition

Il n'existe pas de définition unique de l'Internet des objets qui soit acceptable par la communauté mondiale des utilisateurs. Toutes les définitions ont en commun l'idée que la première version de l'internet concernait des données créées par des personnes, tandis que la prochaine version concernera des données créées par des objets. La meilleure définition de l'Internet des objets serait la suivante :

Un réseau ouvert et complet d'objets intelligents qui ont la capacité de s'autoorganiser, de partager des informations, des données et des ressources, de réagir et d'agir face à des situations et des changements dans l'environnement".

II.2.3 Les composants d'internet des objets [12]

Un système IoT possède plusieurs composants résumés dans le tableau 1.

Tableau II.1 : composants de l'IoT.

Composant	Fonction
Objets connectées	On y ajoutant des capteurs afin de recueillir les données de fonctionnement.
Réseau de communication	Afin de connecter les objets du système et transférer les données reçus et envoyées par tout composant du système
Base de données	Une plateforme de collecte et d'hébergement des données. Ex : stockage sur le cloud
Unité de traitement	Des outils, plateformes ou logiciels capables de traiter les données

IHM (interface homme machine)	Des applications et services afin de visualiser et d'interagir avec son système connecté.

II.2.4 Les avantages de l'IoT

Dans chaque domaine où nous utilisons l'IoT on trouve des avantages et un effet très important de cette technologie, parmi eux nous citons :

- Efficacité : elle permit d'améliorer l'efficacité grâce à la prise de décisions plus intelligentes et plus éclairées.
- Reduction de couts : coûts de maintenance en remplaçant les maintenances inutiles par une maintenance prédictive plus efficace et personnalisée.
- Optimisation : optimise les processus avec une meilleure communication et un contrôle des opérations à distance pour identifier les risques, les sources de gaspillage et les éventuels goulots d'étranglement.
- Anticipation et sureté de fonctionnement : elle anticipe tout interventions de maintenance (chaîne de production) et réduit fortement les risques de pannes subites.

II.2.5 Les inconvénients de l'IoT

Pour l'IoT, le principal sujet tourne autour des enjeux liés à la sécurité des données et aux enjeux des cyberattaques. En effet, à partir du moment où un équipement est connecté, il peut être la source d'une attaque par des pirates informatiques.

Il est donc essentiel de mettre en place des protocoles de sécurité très stricts pour éviter qu'un individu malveillant n'accède à des données hautement sensibles sur le fonctionnement du système.

II.2.6 Domaines d'application d'IoT

Plusieurs domaines d'application sont touchés par l'IoT, et qui nous entoure souvent. Nous allons citer par la suite quelques principaux domaines touchés par cette technologie.

a. Les transports

De plus en plus utilisés par les communes, les objets servent à informer en temps réel d'une place qui se libère dans une rue, à piloter une voiture autonome, à recharger un forfait sur smartphone, et vont jusqu'à contrôler le trafic avec des drones.

b. La santé

L'Internet des Objets (IoT) a le potentiel de transformer les soins de santé, en changeant en profondeur la manière dont les hôpitaux, les cliniques et d'autres établissements de soins collectent et utilisent des données en réunissant les principales tendances techniques et commerciales liées à la mobilité, l'automatisation et l'analyse de données pour améliorer les soins aux patients, en offrant des prestations de soins nouvelles ou améliorées et des services pour permettre aux organismes de santé gérant des données de se différencier de la concurrence.

c. Agriculture

Plusieurs techniques sont utilisées dans ce domaine afin d'améliorer son rendement et superviser l'état des fermes et leurs conditions environnementaux, à titre d'exemple

A l'aide de drones munis d'un système GPS, les exploitants peuvent ainsi recueillir en temps réel des données ou des images pour vérifier l'humidité du sol ou encore l'état des cultures et des plantations, ceci leur permet de vérifier le taux de croissance ou encore l'hydratation. En plus de supervision l'intervention est encore possible avec certains robots intelligents sont également utilisés pour des missions de désherbage, grâce à un système de guidage contrôlé par l'Intelligence Artificielle (IA). Pompage automatique ajustement de l'aération et de dizaines actions possibles.

II.2.7 Technologies de communication

Une communication entre deux objets nécessite une certaine compatibilité entre ces deux derniers, cette compatibilité permet d'assurer le transfert de données entre ces deux objets connectés, pour assurer cette compatibilité il nous faut une norme standardisée qui soit compréhensible par tout type d'objets connectés. Cette norme en télécommunication est appelée protocole.

Protocole : est un ensemble de règles et de procédures de communication utilisées de part et d'autre par toutes les stations qui échangent des données sur le réseau pour satisfaire des objectifs bien déterminés. Son objectif principal est d'utiliser du canal de communication permettant le transfert fiable de données de bout en bout.

a. Protocole HTTP

HTTP (HyperText Transport Protocol) constitue le protocole de type question/réponse de transfert de fichiers hypertextes entre un serveur Web et un client Web. Nous l'avons mentionné car notre application mobile est basée sur ce protocole.

Les navigateurs fonctionnent selon le mode clients/serveur. En principe, le client n'est qu'un terminal passif chargé de l'affichage des pages HTML. Cependant, la puissance de calcul des machines clientes autorise le serveur à invoquer celles-ci pour l'exécution de certaines tâches. L'utilisateur n'a aucun contrôle sur ces petits programmes qui peuvent donc poser des problèmes de sécurité. Ces programmes ou routines dénommés « applet » (en français : appliquette) sont des scripts chargés par le navigateur et interprété par l'interpréteur « Java » (machine virtuelle). Cette approche assure l'indépendance totale entre le développement et la machine cliente. La figure II.18 illustre une session d'échange entre un serveur HTTP (port 80 par défaut) et un client HTTP [13].

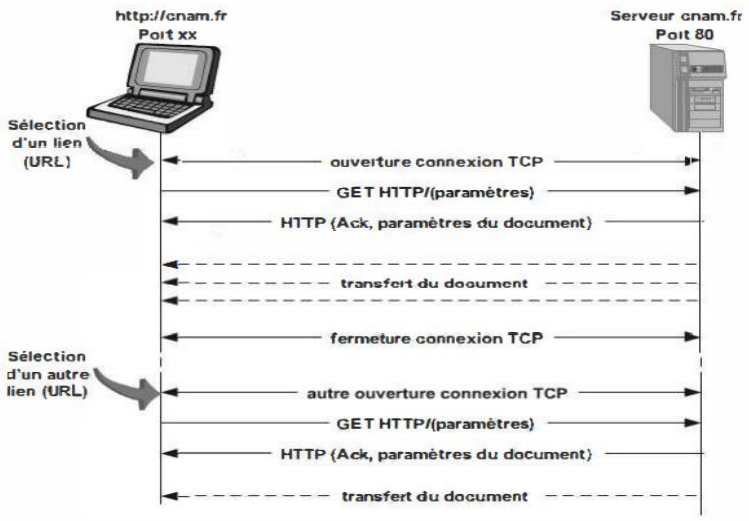


Figure II-8 : échange serveur client" HTTP".

II.3 Technologie RFID

L'identification par radiofréquence (RFID) est née pendant la Seconde Guerre mondiale, lorsqu'il était impératif de déterminer si les combattants étaient "amis ou ennemis". En substance, le système facilite l'identification automatique grâce à une combinaison d'étiquettes et de lecteurs. Aujourd'hui, les systèmes RFID sont appliqués avec succès dans les domaines de la fabrication, de la chaîne d'approvisionnement, de l'agriculture, du transport, des soins de santé et des services, pour n'en citer que quelques-uns. La recherche dans ce domaine s'est développée à un rythme rapide comme en témoigne le nombre d'articles publiés au cours des dernières années [14].

Ces dernières années, la technologie d'identification par radiofréquence est passée de l'obscurité à des applications courantes qui contribuent à accélérer la manutention des produits et matériaux manufacturés. La RFID permet l'identification à distance et, contrairement à la technologie antérieure des codes à barres elle le fait sans nécessiter une ligne de visée [15].

II.3.1 Les différents composants d'un système RFID

a. Lecteur RFID

Le lecteur RFID est un ensemble d'équipements fixes ou mobiles, constitué essentiellement d'une antenne et d'un module RF. Dès lors que le tag se retrouve dans la zone d'action du lecteur, une énergie est fournie au tag pour qu'il puisse fonctionner. Il envoie des commandes spécifiques et reçoit en retour des informations contenues dans la puce. Dans un autre cas de figure le lecteur peut lire et écrire, ainsi les informations reçues sont envoyées vers un autre dispositif qui va se charger du traitement des données (ordinateur).

b. Le tag RFID

Est un circuit électronique qui comprend une puce et une antenne et qui répond aux commandes émises par le lecteur. Il y a deux catégories : actives et passive. Le tag actif fournit lui-même son énergie à travers une batterie, le passif convertit le signal reçu en énergie afin de l'utiliser pour transmettre.

c. Fonctionnement de la RFID

Les tags comprennent une étiquette composée d'une antenne et d'un support d'information (généralement une puce en silicium) ; certaines contiennent une batterie (étiquette active) et d'autres non (étiquette passive). Le principe de fonctionnement est celui d'un dispositif de communication sans fil classique.

Le lecteur est utilisé pour établir une interface entre, d'une part, l'identifiant (ID) contenu dans l'étiquette et, d'autre part, une base de données permettant d'accéder à d'autres informations sur l'étiquette. Le lecteur transmet une trame selon un protocole normalisé, et le signal est ensuite modulé. Dans le cas de l'ultra haute fréquence (UHF), l'étiquette reçoit une partie de cette onde radiofréquence (RF). L'antenne de l'étiquette recueille alors le signal et le transmet à la puce RFID [16].

Pour les étiquettes RFID passives, la puissance transmise par l'onde générée par le lecteur permet au tag d'être alimentée, et donc de fonctionner. De plus, le lecteur transmet l'onde porteuse en continu, alors que la trame est envoyée périodiquement. Dans ce cas, l'antenne du tag remplit une sorte de double fonction :

- Assurer l'alimentation électrique du tag.
- Permettre la transmission du signal "utile" vers la puce à traiter.

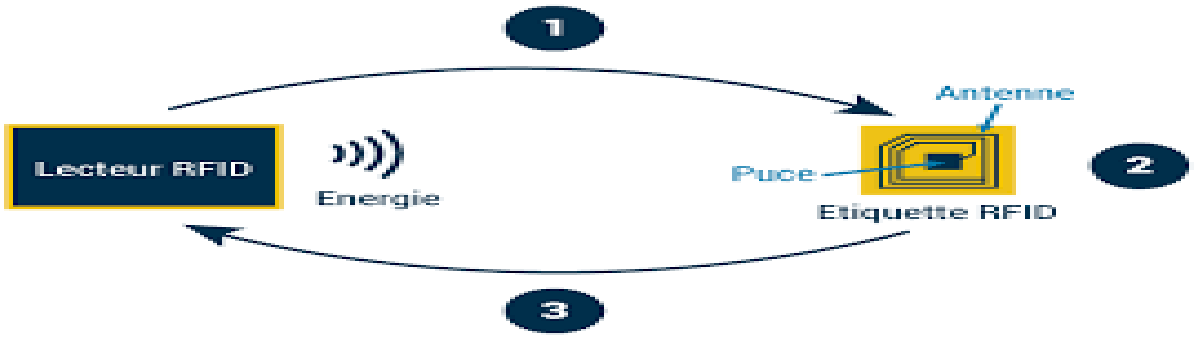


Figure II-9 : Fonctionnement du tag/lecteur RFID.

Conclusion

Après avoir compris le principe de la technologie RFID, et balayer l'essentiel du traitement d'image ainsi que l'internet des objets nous pourrons maintenant les projeter sur le chapitre suivant qui se fera l'objet de décrire tous les équipements dont on a matériel ou logiciel ; afin de préparer un meilleur environnement pour une réalisation d'un Smart parking.

III. Chapitre 03

Description élémentaire du projet

Introduction

Le travail à réaliser nécessite un ensemble d'équipements et de connaissances en programmation pour cela nous allons aborder dans ce chapitre les différents composants matériels utilisés, leurs caractéristiques ainsi que leurs principes de fonctionnement. La même chose pour la partie « soft » et tout ce qui relié à la programmation et au système de gestion de base de données.

III.1 Partie Matériel

Le système va contenir une/des cameras ip pour détecter les emplacements vides ou disponibles comme un capteur, et un Raspberry pi qui va traiter les données de la caméra et l'envoyer à la base de données du cloud. Un module RFID va être exploité afin de contrôler le système d'accès.

III.1.1 Raspberry Pi

Le Raspberry Pi est un ordinateur très bon marché qui fonctionne sous Linux, mais il fournit également un ensemble de broches GPIO (general purpose input/output), permettant de contrôler des composants électroniques pour différents utilisations. Il est fabriqué par la Fondation Raspberry Pi à base d'un processeur ARM. Ses versions ont connu une énorme évolution allant de sa première version dotée d'un processeur monocœur de 700 MHz et de seulement 256 Mo de RAM jusqu'au dernier modèle qui est doté d'un processeur quadricœur cadencé à plus de 1,5 GHz et de 4 Go de RAM.

La carte utilisée dans ce projet est le Raspberry Pi 3 modèle B+ dont ses caractéristiques sont indiquées en dessous.

a. Spécifications

Le Raspberry Pi 3 Modèle B+ est la dernière révision de la gamme Raspberry Pi 3.

Tableau III.1 : Spécifications de la carte Raspberry 3.

Processeur	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
Memoire	1 GO DE SDRAM LPDDR2

Connectivité	LAN sans fil 2,4 GHz et 5 GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE Gigabit Ethernet sur USB 2.0 (débit maximal de 300 Mbps) HDMI pleine grandeur 4 ports USB 2.0.
GPIO	Connecteur GPIO 40 broches étendu
Caméra	Port caméra CSI pour la connexion d'une caméra Raspberry Pi
Affichage	Port d'affichage DSI pour la connexion d'un écran tactile Raspberry Pi
Son	Sortie stéréo 4 pôles et port vidéo composite
Support carte SD	Port Micro SD pour charger votre système d'exploitation et stocker des données
Alimentation	Entrée d'alimentation CC 5V/2,5A

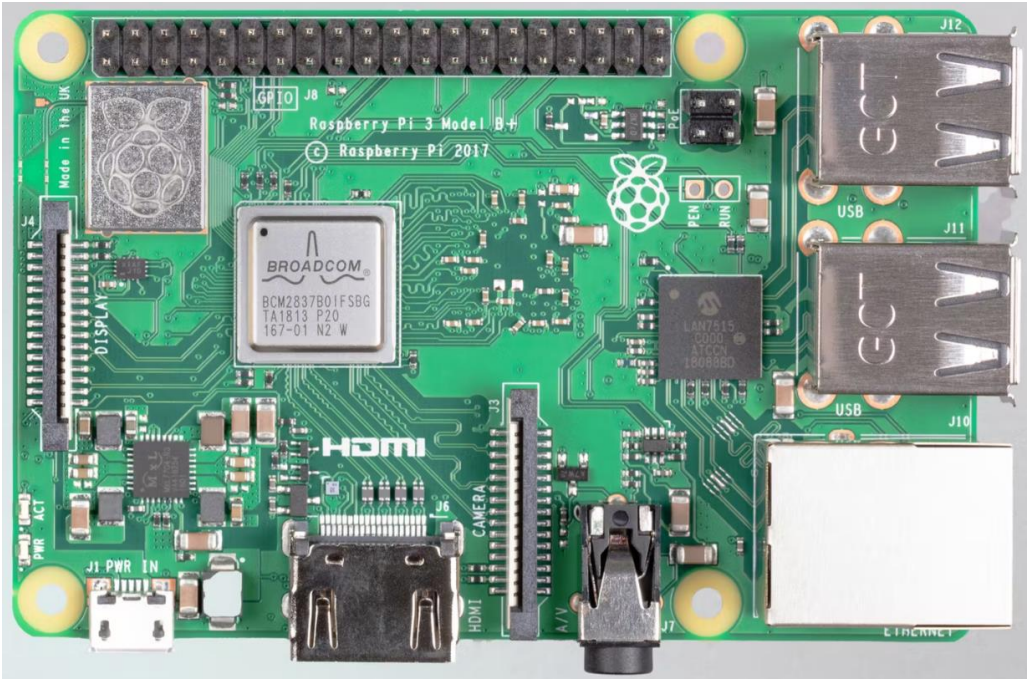


Figure III-1 : Carte Raspberry Pi 3 [3].

III.1.2 Module camera Raspberry pi

Module de caméra haute définition. Il donne une faible diaphonie, une haute sensibilité et un faible niveau de bruit et une capture d'image à faible bruit dans un design ultra léger et petit. Ce module se connecte à la carte Raspberry Pi via le connecteur CSI spécialement conçu pour l'interface avec les caméras. Le bus CSI est capable de débits de données extrêmement élevés et acheminer exclusivement les données de pixels au BCM2835.

a. Caractéristiques

Tableau III.2 Caractéristique de la caméra.

Compatibilité	Entièrement compatible avec les Raspberry Pi modèle A et modèle B.
Capteur d'images	Module caméra 5MP Omni vision 5647
Résolution	- 5-megapixel - Résolution images fixes : 2592 x 1944 - Taux de transfert d'image maximum : 1080p : 30fps / 720p: 60fps
Connection au Raspberry pi	Câble ruban 15 broches, vers l'interface série dédiée MIPI 15 broches pour caméra (CSI-2).
Taille	20 x 25 x 9 mm
Poids	3g
Taille de l'objectif	1/4"

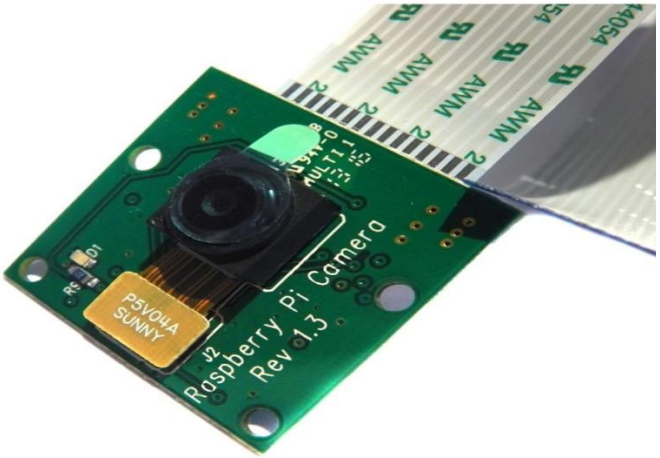


Figure III-2 : Camera Ip V1.3.

III.1.3 Module RFID RC522

Le module RFID-RC522 de Joy-it est une carte d'interface compatible Arduino et Raspberry Pi, basée sur le circuit MFRC-522 de NXP et est utilisée pour lire et écrire sur des cartes ou badges RFID de type Mifare.

La carte microprocesseur Raspberry Pi et le module RFID communiquent via le bus SPI permettant de laisser libres les autres ports de la carte pour d'autres applications.

Tableau III.3 : Module RFID RC522.

Alimentation	3,3 Vcc
Fréquence	13,56 MHz
Distance optimale	10 mm
Protocole	Mifare
Interface	SPI
Dimensions	(Sans les broches): 61 x 40 mm Hauteur avec les broches : 8 mm

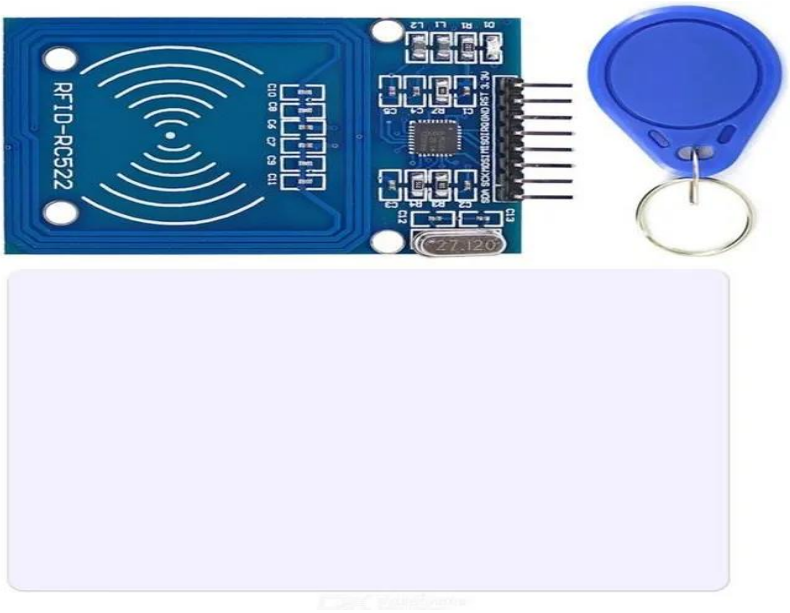


Figure III-3 : Module RFID RC522.

III.1.4 Carte d'essais (breadboard)/câbles

Pour connecter le lecteur RFID avec la carte Raspberry Pi



Figure III-4 : Carte d'essai.

III.2 Partie logicielle et programmation

III.2.1 Langage de programmation

Le projet englobe plusieurs taches qui sont traitées, commençant par le traitement numérique de vidéo de surveillance passant par le programme chargée de l'accès (module RFID) en terminant par l'application mobile.

a. Python

Est un langage de programmation informatique souvent utilisé pour une variété de taches (Analyse de données et apprentissage automatique développement Web, logiciels, Automatisation ou création de scripts, Test et prototypage de logiciels, analyses de données etc.

Le langage Python est placé sous une licence libre proche de la licence BSD et fonctionne sur la plupart des plates formes informatiques, des smartphones aux ordinateurs centraux, de Windows à Unix avec notamment GNU/Linux en passant par macOS, ou encore Android, iOS, et peut aussi être traduit en Java ou .NET.

Sa polyvalence et sa simplicité lui rendre un langage d'une vaste communauté active qui contribue au pool de modules et de bibliothèques de Python et qui agit comme une ressource utile pour les autres programmeurs. Grâce à la réactivité de membres de cette communauté des milliers de problèmes a été résolus et partagées à cause de ces avantages que nous avons choisi ce lagunage de programmation.



Figure III-5 : Logo de Python.

b. Les bibliothèques utilisées

- Opencv : spécialisé pour la vision par ordinateur, elle possède des centaines de fonctions qui aident au traitement numérique des images et d'apprentissage automatique.
- Numpy : cette bibliothèque est destinée pour exploiter des fonctions mathématiques qui s'opèrent sur des tableaux multidimensionnels ou matrices
- Tkinter : Module de base, destiné aux créations des interfaces graphique pour le développement logiciel. Son intérêt dans ce projet est de réaliser une interface graphique à l'entrée du parking pour l'utilisateur qui sera représentée dans le dernier chapitre.

c. Kotlin

Un langage de programmation informatique spécialisé pour les applications Android, développé par l'entreprise JetBrains. C'est un langage oriente objet plus pratique et simple à développer avec. Il est choisi pour le même raison, sa vaste communauté et le partage de problèmes résolus

III.2.2 Système d'exploitation

a. Raspbian OS

C'est le système d'exploitation officiel supporté, libre basé sur Debian, optimisé pour le matériel Raspberry Pi, et est le système d'exploitation recommandé pour une utilisation normale sur un Raspberry Pi. Le système d'exploitation est livré avec plus de 35 000 paquets : des logiciels précompilés regroupés dans un format agréable facile a installer sur le Raspberry Pi [17].

b. Logiciels

- **PyCharm**

Un environnement de développement intégré utilisée pour programmer en python, fonctionne sous Windows MacOs et GNU/Linux.



Figure III-6 : Logo de PyCharm.

- **Android studio**

Est un environnement de développement pour les applications mobile, basé sur Intellij IDEA, il utilise le moteur de production Gradle. Il fonctionne sous Windows, linux, MacOs...etc.



Figure III-7 : Logo Android studio.

III.2.3 Gestion de base de données

Le système contient un ensemble de données concernant les utilisateurs ou bien celles qui sont envoyé du centre du traitement à l'application mobile, on parle donc de la base de données.

Firebase : La base de données en temps réel Firebase est une base de données hébergée dans le nuage(cloud). Les données sont stockées et synchronisées en temps réel avec chaque client connecté. La création d'une application multiplateforme (SDK Apple, Android et JavaScript), tous les clients partagent une instance de la base de données en temps réel et reçoivent automatiquement les mises à jour avec les données les plus récentes.

a. Acces en temps reel

La base de données en temps réel de Firebase permet de créer des applications riches et collaboratives en autorisant un accès sécurisé à la base de données directement à partir du code côté client. Les données sont conservées localement et, même hors ligne, les événements en temps réel continuent de se produire, offrant à l'utilisateur final une expérience réactive.

Dans ce travail la base de données va contenir les informations personnelles du client (Nom, Prénom, email, crédit), ainsi que le nombre de places vide sur le parking et son état et d'autres données qui sont utiles.

Conclusion

Dans ce chapitre, nous sommes arrivés à donner une description des éléments constituants notre projet, ce qui permettra au lecteur de comprendre le principe et le fonctionnement, ainsi que les caractéristiques de notre propre projet. Le dernier chapitre se fera l'objectif d'implémenter l'application finale du projet.

IV. Chapitre 04

Réalisation et tests expérimentaux

Introduction

Les systèmes de stationnement intelligents représentent une tendance qui promet d'améliorer la qualité de vie dans les villes intelligentes en réduisant l'encombrement des parkings, en augmentant le chiffre d'affaires des propriétaires des zones de stationnement qui vont avec ces systèmes générer plus de revenu.

Dans ce chapitre, nous présenterons l'étude technique détaillée du projet. Nous allons parler de l'architecture logicielle ainsi que l'explication des interfaces utilisées dans ce projet.

IV.1 Description du fonctionnement du projet

Le principal objectif du projet est de construire un système de stationnement en accord avec l'évolution des villes intelligentes, pour faciliter le stationnement et la supervision des parkings.

La solution suggérée permettra à l'utilisateur de :

- Vérifier en temp réel combien de places vides restantes dans n'importe quel parking dans notre réseau de parkings.
- Réserver une place dans un parking par l'application mobile dans le plus proche parking, pour cela il faut que :
  - Chaque utilisateur soit enregistré dans une base des données par un unique ID.
  - Le chargement du crédit se fait dans le guichet de n'importe quel parking du réseau.

Le système sera composé en deux parties :

Une partie invisible qui consiste au traitement numérique de la vidéo de surveillance du parking.

Une deuxième partie apparente consacrée aux utilisateurs et aux administrateurs qui se compose de quatre sous-parties :

a) L'application mobile qui servira à :
   - Vérifier les places vides dans le parking.
   - Réserver une place

b) Les deux interfaces homme machine sont comme suit :
Une est réservée pour l'administrateur ; elle est conçue pour l'ajout des nouveaux utilisateurs :

- Ajouter de nouveaux clients avec l'ID de la carte à puce.
- Ajouter de crédit aux clients.

L'autre est pour l'affichage à l'entrée du parking.

c) Une base de données en temp réel pour stocker les informations des utilisateurs.

d) Regroupement de la carte à puce avec son lecteur RFID, un actionneur qui contrôle la barrière et une caméra pour détecter les places vides.

La figure illustre le principe de fonctionnement du système de stationnement intelligent

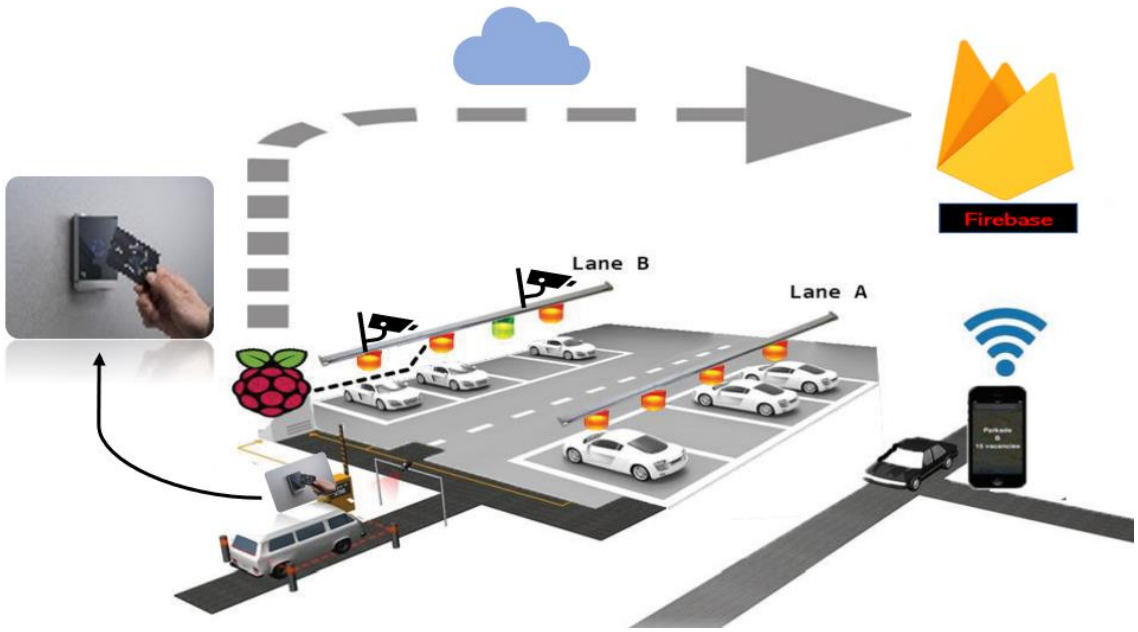


Figure IV-1 : Principe de fonctionnement du système de stationnement intelligent.

Le processus commence par le traitement numérique des images des caméras de surveillance qui sera traité par la carte Raspberry

Le traitement d'image utilisé est basé sur un traitement basique des pixels de chaque région ou segment qui représente un espace de stationnement. Pour cela, nous avons convertir notre image en image binaire et se basant sur le comptage des contours et les frontières nous pourrons connaitre si cet emplacement est vide ou non sans avoir besoin de détecter un objet et faire entrainer un modèle.

IV.2 Installation de la camera

Le choix de l'angle est un point très important afin de balayer le maximum de surface pour couvrir plus de places de stationnement.

IV.3 Préparation de l'environnement du traitement

Prise de photo du parking avec l'angle de vision qu'on possède. Voir figure IV.2



Figure IV-2 : Image originale prise par la caméra.

Selon la forme des emplacements de stationnement et l'angle de prise de photo nous essayons de les localiser. Voir figure IV.3

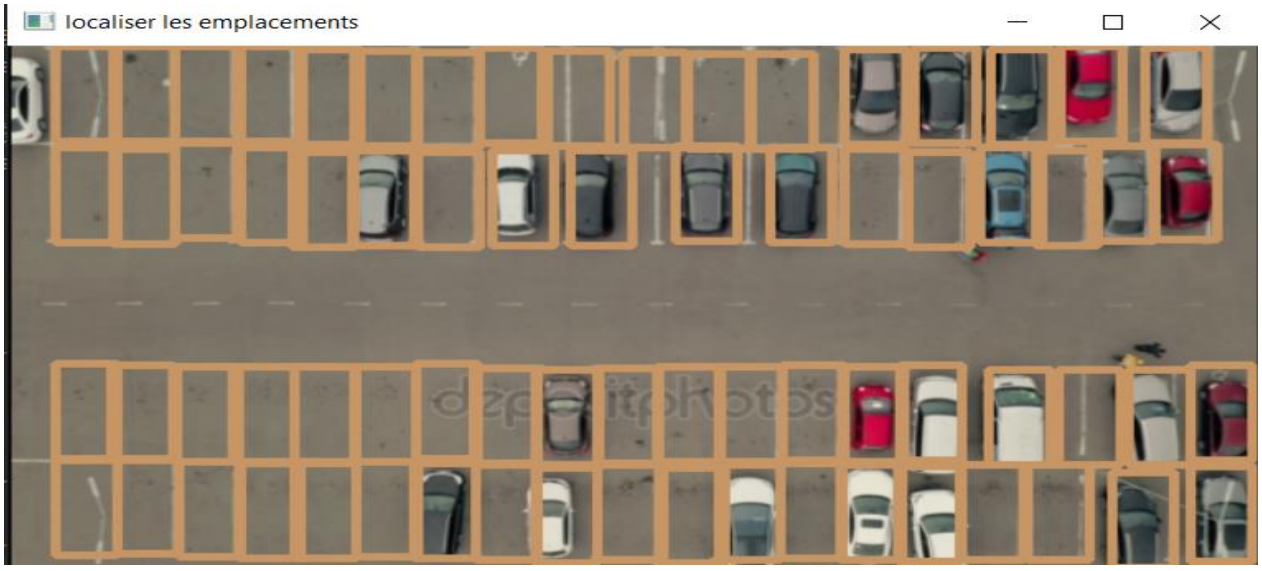


Figure IV-3 : Localisation des places de stationnement selon leur forme.

Vu que la réalisation est effectuée par une caméra immobile c'est-à-dire que le changement soit causé juste par le mouvement des véhicules.

L'idée est de projeter l'image du parking avec les emplacements localisés sur la vidéo générée par la caméra de surveillance puis en appliquant quelques fonctions de traitement d'image et de vision artificielle.

Nous considérons que chaque petit cadre de stationnement comme un segment que nous devons le subir aux quelques traitements numériques, beaucoup plus le filtrage, donc nous avons créé une fonction principale qui recadre chaque segment ou placement de stationnement et puis nous avons appliqué l'ensemble des fonctions expliquées en dessous afin de savoir si ce dernier contient un véhicule ou vide.

IV.4 Les fonctions appliquées

Fonction image-flou (Gaussian Blur) : c'est vrai qu'elle rend l'image floue, mais ce dernier est utile pour limiter le nombre de pixels à traiter, c'est un filtre passe-bas qui supprime les bruits de hautes fréquences.



Figure IV-4 : Image floue.

Seuillage (addaptiveThreshold) : considérée comme première forme de segmentation base sur une valeur de seuil et de comparer les valeurs de pixels à ce seuil.



Figure IV-5 : Seuillage image flou/originale.

Remarque :

On remarque une différence du bruit entre les deux images de la figure IV.5. Ça prouve que la fonction « Flou » réduit les bruits

Median-Blur : Cette opération traite les bords tout en supprimant le bruit. Le résultat est montré dans la figure IV.6

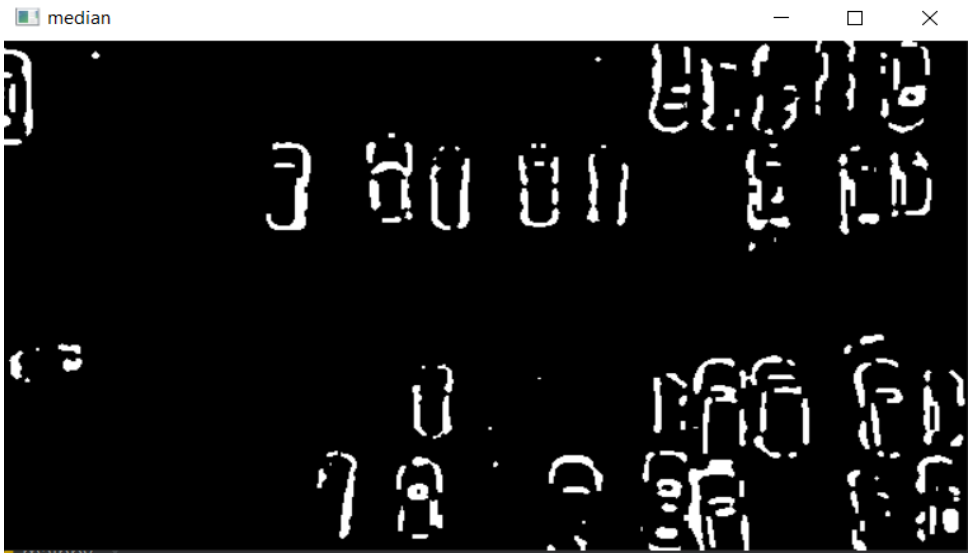


Figure IV-6 : Filtre Médian.

Dilatation (Dilate) : elle fait apparaitre plus les bords c'est-à-dire qu'elle ajoute des pixels au limites d'image. Voir figure IV.7.

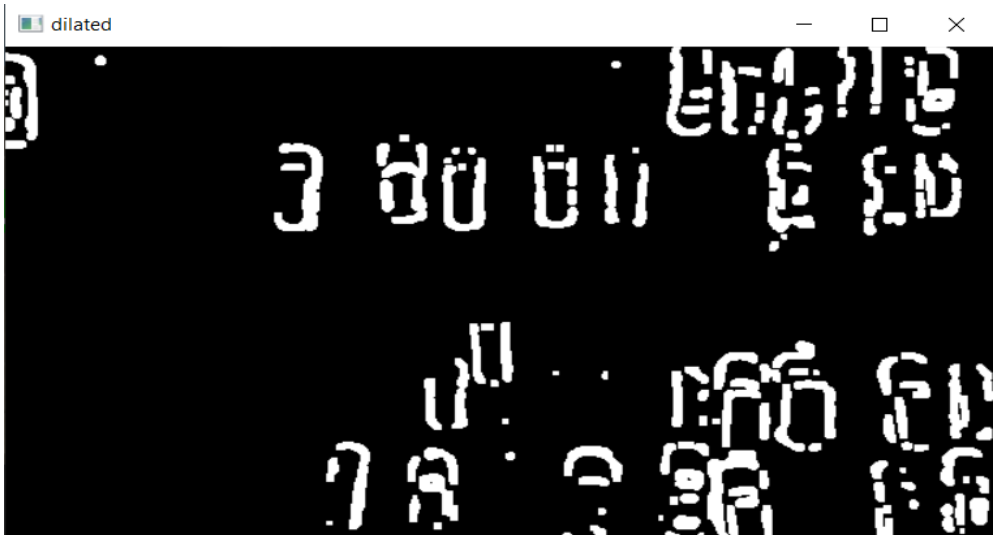


Figure IV-7 : Image dilatée.

On peut voir facilement l'apparence juste des véhicules en blanc, en exploitant ces pixels dans la suite nous obtenons les résultats suivants.

Count-Non-Zero : donne le nombre de pixels non nuls dans la matrice du tableau. Cette fonction peut être utile pour calculer le nombre de pixels noirs ou le nombre de pixels blanc. Les pixels nuls indiquent la couleur noire.

En pratique ce type de traitement n'est pas le plus performant, car il nécessite une lumière quasi constante pour que le traitement ne s'affecte pas trop. Pour remédier à ça, il est indispensable de développer un algorithme qui répond aux variations de lumière liée aux problèmes d'éclairages, brouillard etc.

La dernière fonction nous a aidé à distinguer les places ; si elles sont vides ou occupées comme montre la figure IV.8.



Figure IV-8 : Surveillance du parking.

A partir de ce traitement l'application enlève les informations nécessaires et dans la suite nous allons faire une démonstration sur la mise en marche du système.

IV.5 Présentation de l'application

IV.5.1 L'application mobile

a. Ecran de démarrage



FigureIV-9 : Ecran de démarrage.

Les écrans de démarrage ne servent à rien pratiquement, ils sont faits seulement pour le design et le logo de l'application.

b. Authentification

La figure IV.10 montre la page d'inscription ou de connexion au compte.

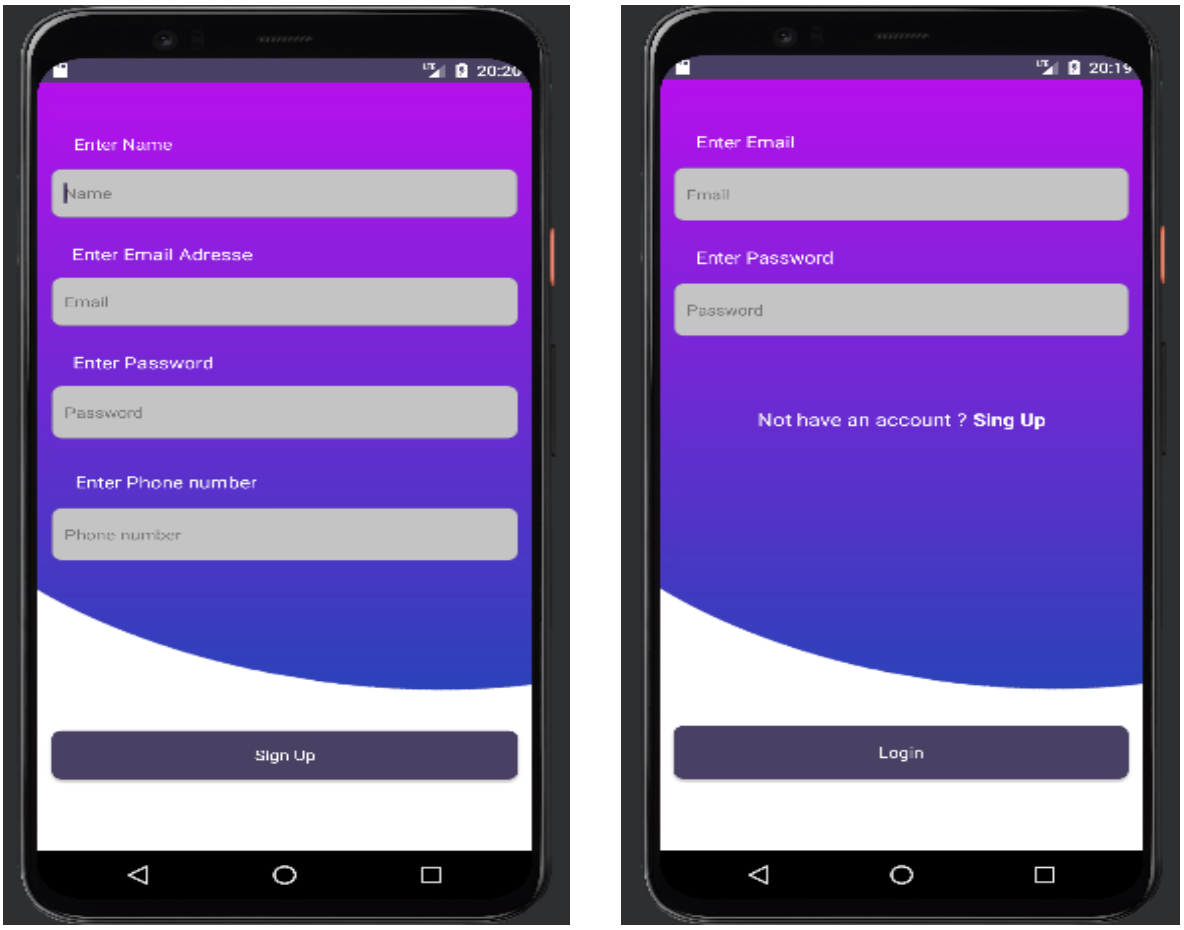


Figure IV-10 : Inscription connexion.

Pour utiliser l'application, il faut disposer d'un compte ouvert sur la page d'inscription « Sign Up ». A l'ouverture de l'application, ce compte sera enregistré automatiquement dans la base de données.

Après l'inscription, l'accès au compte se fait directement par l'email et le mot de passe de l'utilisateur par la fenêtre de connexion « Login ».

c. Page d'accueil

La figure IV.11 montre la page d'accueil de l'application.

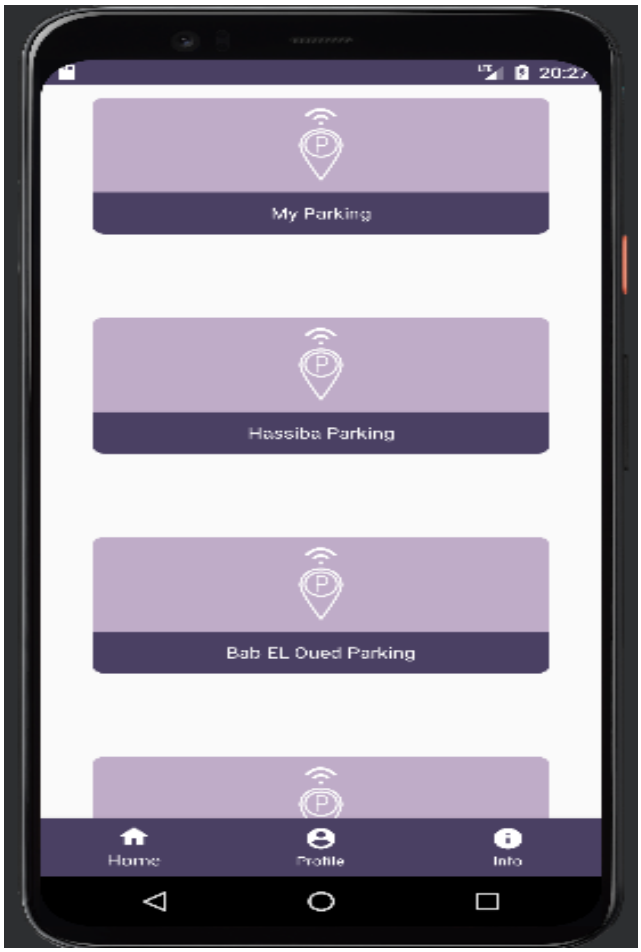


Figure IV-11 : Parkings disponibles.

La page d'accueil affiche tous les parkings connectés au réseau, en sélectionnant un parking nous aurons toutes les informations sur ce dernier comme la disponibilité des places de stationnement et le bouton de réservation ce qui nous permet de réserver une place.

d. Information sur le parking

Nous voyons ici le nombre de places vides et la date de la dernière réservation qu'on a effectuée.

Le bouton « RESERVE » permet de réserver une place dans la date choisie sachant que cette dernière ne doit pas être loin d'un mois et son stockage réel dans la base de données se fait avant deux heures de l'heure de la réservation. Voir figure IV.12.

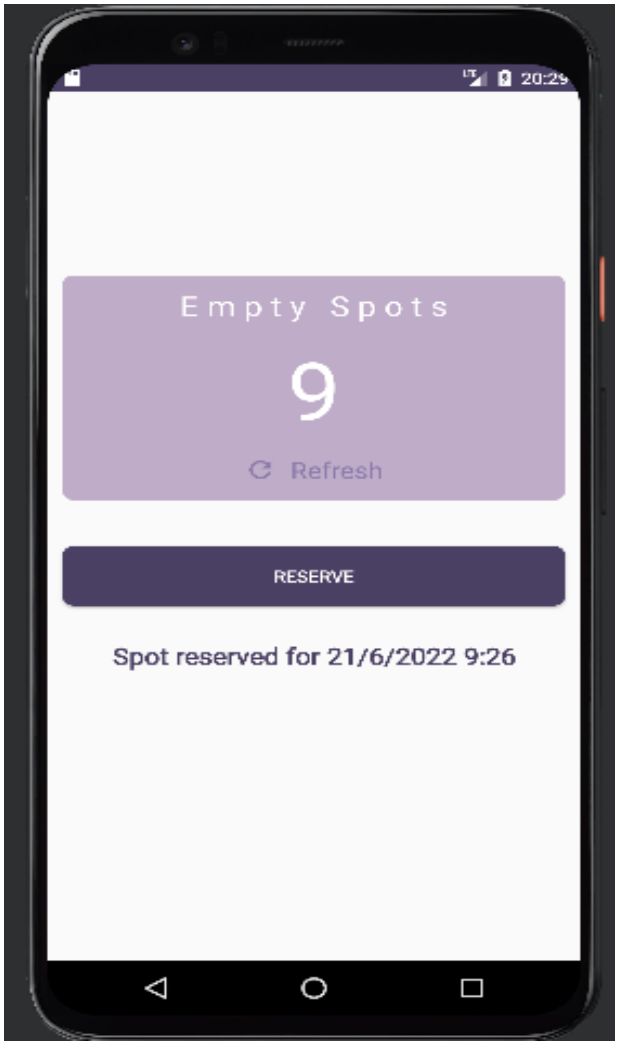


Figure IV-12 : Informations du parking sélectionné.

Le nombre de place disponible vient du centre de traitement de la vidéo vers l'application mobile, pour cela la configuration montrée dans la figure suivante assure la connexion en temps réel de la base de données de l'application et les informations extraits de la vidéo.

```
config = {
  "apiKey": "AIzaSyDKofilrzHQrGPCiXvj8gW6IrI6sLFmLwY",
  "authDomain": "smart-parking-f3e7f.firebaseapp.com",
  "databaseURL": "https://smart-parking-f3e7f-default-rtdb.firebaseio.com",
  "projectId": "smart-parking-f3e7f",
  "storageBucket": "smart-parking-f3e7f.appspot.com",
  "messagingSenderId": "1006665309832",
  "appId": "1:1006665309832:web:37d287615bd6a6a17dc93d",
  "measurementId": "G-B57QEMX8NC"
}

firebase = pyrebase.initialize_app(config)
auth = firebase.auth()
db = firebase.database()
```

Figure IV-13: Configuration

La figure IV-14 montre le nombre de places disponibles, cette information sera envoyée vers la base de données.



Figure IV-14: l'information envoyee vers l'application.

Après l'appui sur le bouton « RESERVE » nous s'orientons vers la page de sélection de date et heure de réservation comme le montre la figure IV.13.

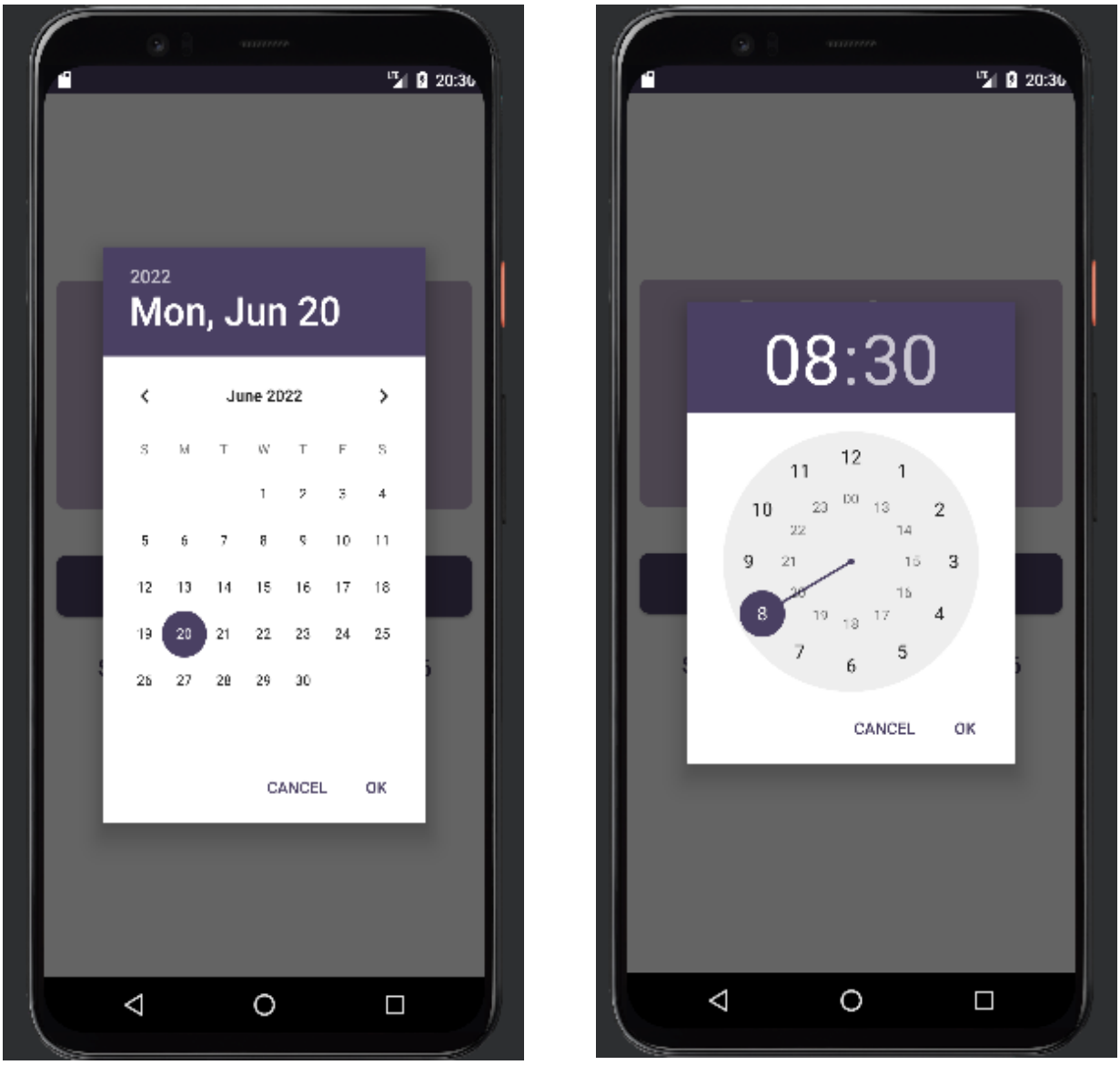


Figure IV-15 : Choix de date de réservation.

Le client aura la main de choisir la date du jour ensuite l'horaire.

e. Page de profil

Toutes les informations sur l'utilisateur s'affichent dans cette page et nous pouvons les modifier comme le montre la figure IV.14.

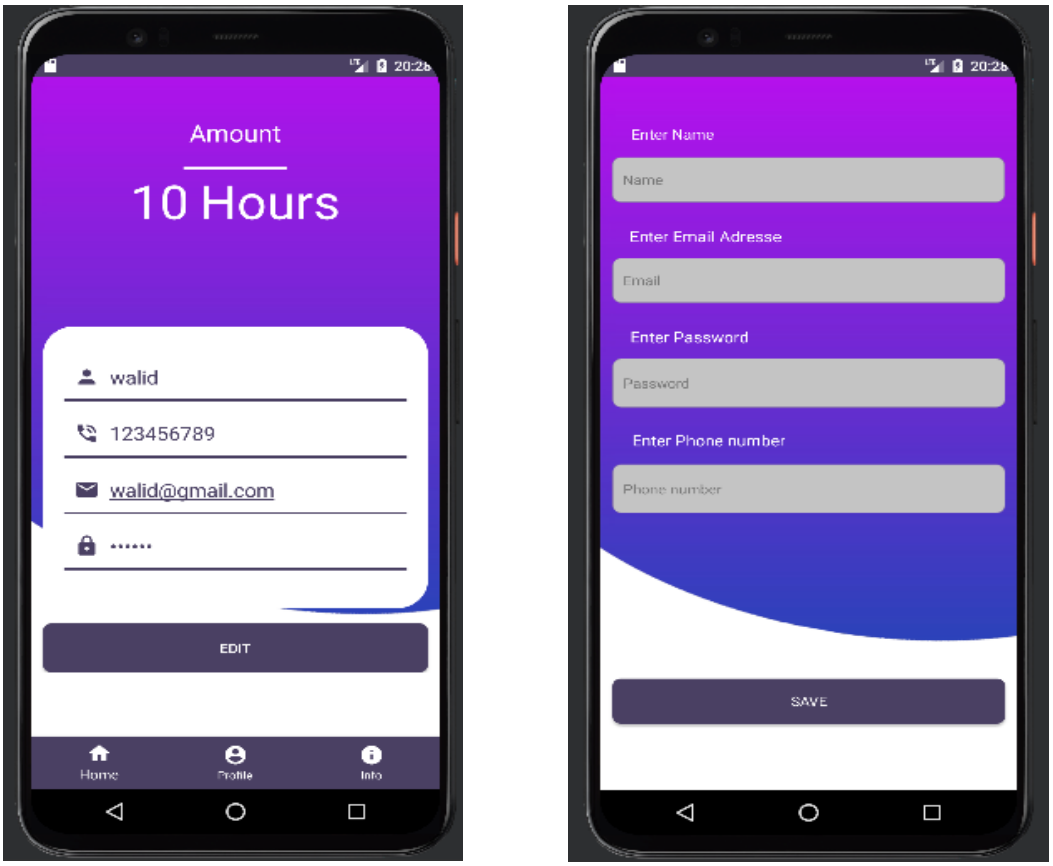


Figure IV-16 : Informations d'utilisateur.

Le profil comporte :

- Le nom d'utilisateur
- Le numero de telephone
- L'addresse email
- Le mot de passe

IV.5.2 Les Interfaces homme machine

a. L'interface administrateur

La figure IV.15 montre l'interface d'administrateur.

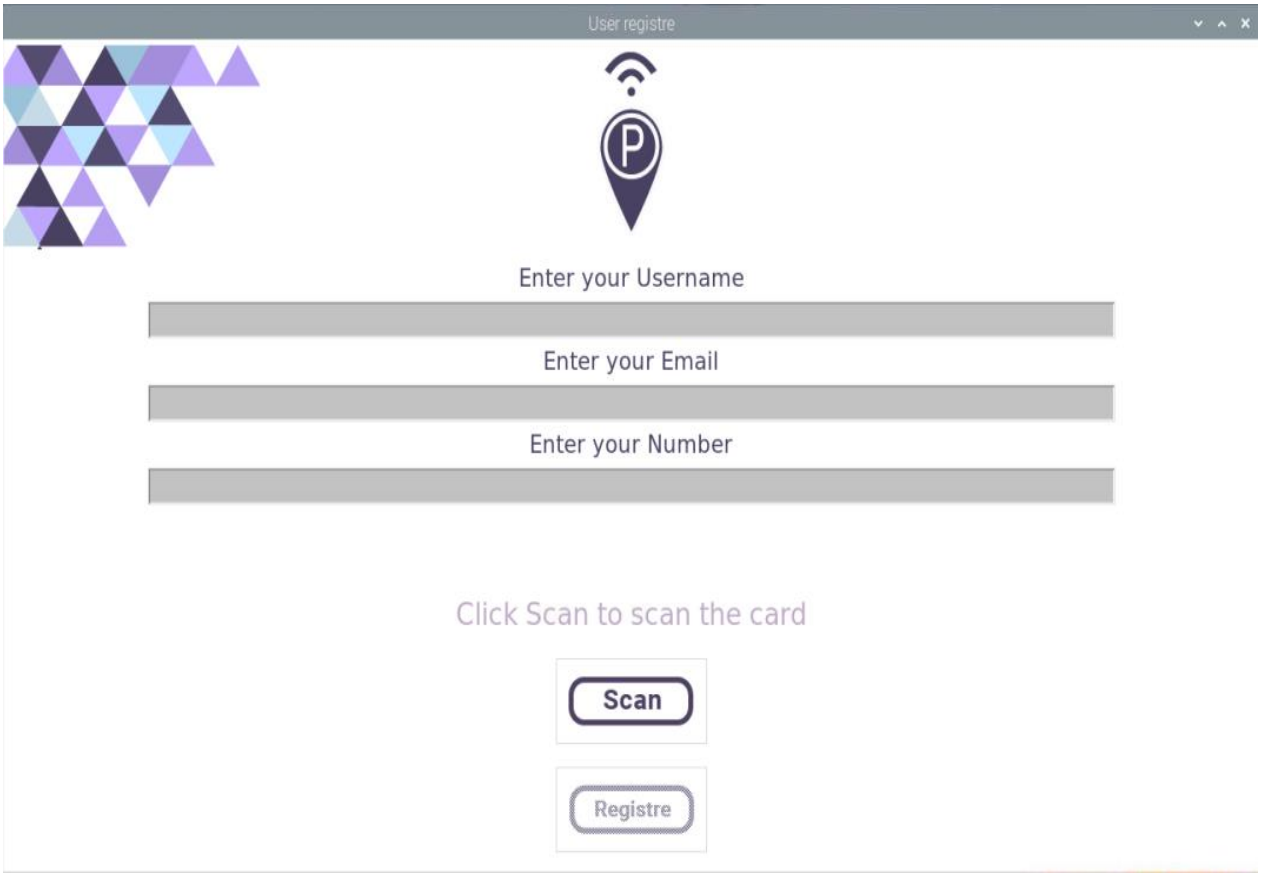


Figure IV-17 : Interface d'administrateur.

Cette interface est destinée à l'administrateur qui se charge d'ajouter les nouveaux utilisateurs dans la base des données en enregistrant les informations suivantes : Le nom, L'email, Le numéro de téléphone et un mot de passe initial généré automatiquement par le programme.

Les étapes d'enregistrement :

- Scanner une carte à puce et enregistrer son ID dans une variable lors de l'appui sur le bouton « Scan ».
- Remplir des informations nécessaires dans les champs appropriés.
- L'appui sur le bouton « Registre » qui envoie toutes les informations d'enregistrement vers la base de données.

b. L'interface d'affichage

La figure IV.16 montre l'interface affichée à l'entrée du parking.



Figure IV-18 : Interface d'entrée.

Cette interface est placée à l'entrée du parking avec le lecteur RFID pour l'affichage des places vides et du nom d'utilisateur lors de l'entrée.

Le client doit passer sa carte devant le lecteur à l'entrée du parking, une page lui apparaitre qui affiche les places vides et son numéro d'identification.

Si la carte scannée par le lecteur RFID n'est pas enregistrée dans la base des données, un message d'alerte s'affiche dans l'écran d'affichage et la barrière ne s'ouvre pas.

Conclusion

Cette partie est consacrée aux essais et à la réalisation pratique (conception, intégration, programmation et validation) de notre projet tout en passant par l'ensemble des composants et des modules utilisés constituants ce dernier, la réussite de cette réalisation pratique confirme notre bon choix du matériel. La programmation de notre projet représente un aspect spirituel vu sa grande difficulté, en fin, nous sommes arrivés à créer notre propre application pour un smart parking.

Conclusion générale

Incontestablement, le stationnement est un véritable casse-tête pour tous les conducteurs du monde. Avec les informations sur le stationnement, collectées soit par les capteurs de stationnement des villes, soit par des applications ou des applications de détection de foule, le service de stationnement intelligent peut fournir une plateforme avec le statut de stationnement en temps réel pour les conducteurs.

Grâce à cela, les conducteurs peuvent réduire leur temps de trajet et la ville peut augmenter les revenus provenant des frais de stationnement et des tickets. La plupart des applications de stationnement intelligent.

Se concentrent sur la technologie de détection et le développement d'applications mobiles. C'est pourquoi le déploiement de la ville crée un pont entre les deux. Le déploiement à grande échelle est une bonne occasion de tester la précision de la détection et la connectivité des capteurs.

Les informations détectées peuvent être stockées dans des serveurs et fournir des informations aux applications actuelles.

Ce qui rendre le travail spécial est d'exploiter les caméras de surveillance installées dans les parkings pour un autre bénéfice ou avantage qui nous permet de détecter les places vides dans ce parking. La détection des places de stationnement toute seule n'est pas suffisante, la connexion en temp réel d'une application mobile qu'on a développée et le contrôle d'accès au parking est l'ensemble de taches principales atteintes.

Le traitement de la vidéo de surveillance est effectué en exploitant les fonctions de base expliquées dans le deuxième chapitre, ces fonctions ont considérablement satisfait le besoin de segmenter les places de stationnement et de vérifier leur disponibilité.

Le développement de l'application mobile atteint 80% de ce qu'on a désiré, la vérification en temps réel de cas de parking est atteinte également la réservation du parking souhaité.

L'accès au parking en utilisant les carte RFID est pratiquement suffisant et n'apporte aucun problème ainsi l'interface située à l'entrée de chaque parking elle contient les informations nécessaires d'utilisateur a l'exception de localisation de place réservée dans le parking.

Perspective

Même si la détection été faite en temps réel mais le traitement effectué nécessite plus de développement, car l'image est toujours subie aux différentes contraintes liées à sa clarté à titre d'exemple le modèle que nous avons développé peut y avoir des males comptages de place dans un climat brouillé (si le parking est ouvert) ou dans le cas où il y a de mal stabilité de lumière dans les parkings.

Le positionnement des caméras a pour but de détecter le maximum d'espace, ce qui est plus difficile dans les parkings fermés par rapport aux parkings ouverts, car la hauteur des parkings fermes est limitée.

Du côté de l'application mobile, la localisation de parking dans la carte(map) est de notre perspective de la compléter.

L'orientation du client dans l'interface d'entrée ou dans l'application mobile vers la direction de sa place réservée et son numéro nécessite encore un développement du code de traitement de la vidéo vu que le projet ne se base pas sur les capteurs de position donc la détection et la localisation est purement par analyse et traitement d'image.

V. Références

[1] «openclassroom,» [En ligne]. Available: https://openclassrooms.com/fr/courses/5439096-concevez-un-detecteur-de-fumee-connecte/6132686-decouvrez-le-domaine-des-systemes-embarques-et-objets-connectes.consulté le 12/04/2022.

[2] F. C. E. Grolleau, SYSTÈMES TEMPS RÉEL DE CONTRÔLE-COMMANDE Conception et implémentation, Paris: Dunod, 2005.

[3] «ARDUINO. Le site officiel d'arduino < https://www.arduino.cc/ >.,» [En ligne]. Available: < https://www.arduino.cc/ >. consulté le 23/04/2022.

[4] H. Kopetz, Real-Time Systems, Design Principles for Distributed Embedded Applications, New York, Boston, Dordrecht, London, Moscow: Klower Academic Publishers, 1997.

[5] B. A. Sarah, *Mémoire master informatique thème :Conception et réalisation d'une application mobile pour un parking intelligent],* Bordj Bou Arreridj, 2020.

[6] «[Policy Department A: Economic and Scientific Policy, Mapping smart cities in the eu,,» 2014.,» 2014. [En ligne]. Available: https://www.europarl.europa.eu/thinktank/en/document/IPOL-ITRE_ET(2014)507480. Consulté le 14/04/2022.

[7] [En ligne]. Available: https://www.elecal.ca/ville-intelligente/les-composantes-d-une-ville-intelligente. Consulté le 22/04/2022.

[8] «MyflexiPark,» [En ligne]. Available: https://www.myflexipark.be/fr/smart-parking. Consulté le 22/04/2022.

[9] J.-M. C. J.-P. C. D. C. G. A. M. P. R. Z. J. Z. Ph. Bolon, Analyse d'images : Filtrage et segmentation.

[10] M. Bergounioux, Introduction au traitement mathématique des images - méthodes déterministes, Orléans Cedex.

[11] Y. E. MERABET, «Segmentation d'images couleur par combinaison,LPE-régions/LPE-contours et fusion de régions.Application à la segmentation de toitures à partir d'orthophotoplans,» UNIVERSITÉ DE TECHNOLOGIE BELFORT-MONTBÉLIARD, 2014.

[12] [En-ligne]-Available:Https://fr.rs-online.com/web/generalDisplay .html?id=discovery-conception-electronique/quels-sont-les-avantages-et-les-inconvenients-de-l-iot. Consulté le 21/05/2022.

[13] S. Claude, RESEAUX ET TELECOMS, Paris: DUNOD, 2003, p. 355.

[14] A. N. Nambiar, «RFID Technology: A Review of its Applications /Proceedings of the World Congress on Engineering and Computer Science 2009 Vol II WCECS 2009, 20-22, ,USA],» October 2009.

[15] R. Want, «An Introduction to RFID Technology,» 2006 .

[16] J. W. &. Sons, Inc Radio Frequency Identification and Sensors From RFID to Chipless RFID Etienne Perret.

[17] «Raspberry,» [En ligne]. Available: https//www.raspberrypi.com. Consulté le 20/05/2022.

[18] F. E.Grolleau, SYSTÈMES TEMPS RÉEL DE CONTRÔLE-COMMANDE Conception et implémentation, Paris: Dunod, 2005, p. 110.

VI. Annexe

L'application mobile

Page d'enregistrement

```
        progressDialog = ProgressDialog( context: this)
        progressDialog.setTitle("Please wait")
        progressDialog.setMessage("Creating account In ...")
        progressDialog.setCanceledOnTouchOutside(false)

        auth = FirebaseAuth.getInstance()

        binding.btnSignupPage.setOnClickListener {  it: View!
            validateData()
        }
    }


    private fun validateData() {
        Email = binding.inputEmailSignup.text.toString().trim()
        Password = binding.inputPasswordSignup.text.toString().trim()
        Number = binding.inputPhoneSignup.text.toString().trim()
        Name = binding.inputNameSinup.text.toString().trim()

        if (!Patterns.EMAIL_ADDRESS.matcher(Email).matches()) {
            binding.inputEmailSignup.error = "Invalid Email Format"
        } else if (TextUtils.isEmpty(Password)) {
            binding.inputPasswordSignup.error = "Please enter password"
        } else if (Password.length < 6) {
```

```
package com.example.smartparking

import ...

class SignUpActivity : AppCompatActivity() {

    private lateinit var binding: ActivitySignUpBinding

    private lateinit var auth: FirebaseAuth

    private lateinit var progressDialog: ProgressDialog

    private var Email: String = ""
    private var Password: String = ""
    private var Name: String = ""
    private var Number: String = ""
    private var Credit: String = ""

    private var database = FirebaseDatabase.getInstance()
    val myRef = database.reference

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivitySignUpBinding.inflate(layoutInflater)
        setContentView(binding.root)
```

```
            binding.inputPasswordSignup.error = "password length less then 6"
        } else {
            firebaseSignUp()
        }
    }
    private fun firebaseSignUp() {
        progressDialog.show()
        //creating account
        auth.createUserWithEmailAndPassword(Email, Password)
            .addOnSuccessListener {  it: AuthResult!
                progressDialog.dismiss()
                val firebaseUser = auth.currentUser
                val email = firebaseUser!!.email

                val usersclasse = Users(Name, Number, Email, Password, Credit,  CardId: "",  time: "")
                myRef.child( pathString: "users").child(auth.currentUser!!.uid).setValue(usersclasse)
                myRef.child( pathString: "users").child(auth.currentUser!!.uid).child( pathString: "credit").setValue(2)

                val myIntent = Intent( packageContext: this,ProfileFragment::class.java)
                myIntent.putExtra( name: "SignUpName", Name)

                Toast.makeText( context: this,  text: "Account created with email $email", Toast.LENGTH_LONG).show()
                startActivity(Intent( packageContext: this, ParkingsListActivity::class.java))
                finish()
            }
```

```
            .addOnFailureListener { e ->
                progressDialog.dismiss()
                Toast.makeText( context: this,  text: "SignUp failed due to ${e.message}", Toast.LENGTH_LONG).show()
            }
    }

    override fun onSupportNavigateUp(): Boolean {
        onBackPressed()
        return super.onSupportNavigateUp()
    }

}
```

Page d'accueil

```
package com.example.smartparking


import android.content.Intent
import android.os.Bundle
import android.view.MenuItem
import androidx.appcompat.app.AppCompatActivity
import android.view.View
import androidx.fragment.app.Fragment
import com.example.smartparking.databinding.ActivityParkingsListBinding
import com.example.smartparking.fragment.InfoFragment
import com.example.smartparking.fragment.ParkingsListFragment
import com.example.smartparking.fragment.ProfileFragment


class ParkingsListActivity : AppCompatActivity() {

    private lateinit var binding: ActivityParkingsListBinding
    private val ParkingsList = ParkingsListFragment()
    private val Profile = ProfileFragment()
    private val Info = InfoFragment()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        binding = ActivityParkingsListBinding.inflate(layoutInflater)
        setContentView(binding.root)
```

```
        replaceFragment(ParkingsList)
    }

    private fun replaceFragment(fragment: Fragment) {
        if (fragment != null){
            val transaction = supportFragmentManager.beginTransaction()
            transaction.replace(R.id.fragment_container, fragment)
            transaction.commit()
        }
    }

    fun go_to_parking_info(view: View) {
        val intent= Intent( packageContext: this,ParkingInfo::class.java)
        startActivity(intent)
    }

    fun go_to_profile(item: MenuItem) {
        replaceFragment(Profile)
    }

    fun go_to_home(item: MenuItem) {
        replaceFragment(ParkingsList)
    }

```

```
    fun go_to_info(item: MenuItem) {
        replaceFragment(Info)
    }

    fun go_to_edit(view: View) {
        val intent = Intent( packageContext: this,EditActivity::class.java)
        startActivity(intent)
    }


}
```

Page de profil

```kotlin
package com.example.smartparking.fragment

import ...


// TODO: Rename parameter arguments, choose names that match
// the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
private const val ARG_PARAM1 = "param1"
private const val ARG_PARAM2 = "param2"

class ProfileFragment : Fragment() {
    // TODO: Rename and change types of parameters
    private var param1: String? = null
    private var param2: String? = null

    private lateinit var auth: FirebaseAuth

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        arguments?.let { it: Bundle
            param1 = it.getString(ARG_PARAM1)
            param2 = it.getString(ARG_PARAM2)
        }
    }

```

```kotlin
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_profile, container, attachToRoot: false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        auth = FirebaseAuth.getInstance()

        var database = FirebaseDatabase.getInstance()
        val myRef = database.reference

        myRef.addValueEventListener(object:ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
```

```
        override fun onDataChange(snapshot: DataSnapshot) {

            if (snapshot.exists()) {
                val DBname:String = snapshot.child( path: "users").child(auth.currentUser!!.uid).child( path: "name").value as String
                val DBnumber:String = snapshot.child( path: "users").child(auth.currentUser!!.uid).child( path: "number").value as String
                val DBemail:String = snapshot.child( path: "users").child(auth.currentUser!!.uid).child( path: "email").value as String
                val DBpassword:String = snapshot.child( path: "users").child(auth.currentUser!!.uid).child( path: "password").value as String
                val DBcredit:String = snapshot.child( path: "users").child(auth.currentUser!!.uid).child( path: "credit").getValue<Long>().toString()

                var nameText:TextView = view.findViewById(R.id.profile_name)
                var numberText:TextView = view.findViewById(R.id.profile_number)
                var emailText:TextView = view.findViewById(R.id.profile_email)
                var passwordText:TextView = view.findViewById(R.id.profile_password)
                var creditText:TextView = view.findViewById(R.id.profile_credit)


                nameText.text = DBname
                numberText.text = DBnumber
                emailText.text = DBemail
                passwordText.text = DBpassword
                creditText.text = DBcredit
            }
        }
```

```
            override fun onCancelled(error: DatabaseError) {

            }
        })
    }
}
```

Page de réservation

```
package com.example.smartparking

import ...


private var auth: FirebaseAuth = FirebaseAuth.getInstance()
private var database = FirebaseDatabase.getInstance()
val myRef = database.reference


class ParkingInfo : AppCompatActivity() {

    lateinit var dateView:TextView
    lateinit var spot:TextView
    lateinit var binding: ActivityParkingInfoBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_parking_info)

        binding = ActivityParkingInfoBinding.inflate(layoutInflater)
        setContentView(binding.root)

        spot = findViewById(R.id.spot_nbr)
        dateView = findViewById(R.id.dateView)
```

```
    var handler : Handler = Handler()

    val runnable: Runnable = Runnable {
        myRef.addValueEventListener(object: ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                if (snapshot.exists()){
                    try {
                        var reserved = snapshot.child( path: "reserved").getValue<Long>()
                        val value = snapshot.child( path: "spots").getValue<Long>()
                        var date = snapshot.child( path: "users").child(auth.currentUser!!.uid).child( path: "date").getValue<String>()
                        var DDRS = snapshot.child( path: "users").child(auth.currentUser!!.uid).child( path: "complete_date").child( path: "time").getValue
                        handler.post(Runnable() {
                            run {  }
                        })
                        if (date != ""){

                            dateView.text = "Spot reserved for ${date.toString().trim()}"
                        }
                        spot.text = (value!! - reserved!!).toString().trim()
```

```
                    }catch (ex:Exception){
                        handler.post(Runnable {
                            Toast.makeText( context: this@ParkingInfo, text: "${ex.message}",Toast.LENGTH_SHORT).show()
                        })

                    }
                }
            }
            override fun onCancelled(error: DatabaseError) {
                Toast.makeText( context: this@ParkingInfo, text: "Failed to read value.",Toast.LENGTH_SHORT).show()
            }
        })

    }

    var thread: Thread = Thread(runnable)
    thread.start()
}
```

```
fun choseTime(view: View) {
    var c = Calendar.getInstance()

    DatePickerDialog( context: this,DatePickerDialog.OnDateSetListener { datePicker, yy, mm, dd ->
        var dt = "$dd/${mm+1}/$yy"
        TimePickerDialog( context: this, TimePickerDialog.OnTimeSetListener { timePicker, hh, mm ->
            dt += " $hh:$mm"
            calculateTime(dt)
        },c.get(Calendar.HOUR),c.get(Calendar.MINUTE), is24HourView: true).show()
    },c.get(Calendar.YEAR),c.get(Calendar.MONTH),c.get(Calendar.DAY_OF_MONTH)).show()

}

fun calculateTime(DOR:String){
    var today = Date()
    var sdf = SimpleDateFormat( pattern: "dd/MM/yyyy HH:mm")
    var DORS = sdf.parse(DOR)

    var mounth = (DORS.time - today.time)/2629800000
    var year = (DORS.time - today.time)/31556952000
    var days = (DORS.time - today.time)/86400000*24
    var hours = ((DORS.time - today.time)%86400000/3600000)+1
```

```
    try {
        if (year > 1 || mounth > 1 || hours < 0) {
            WarningDialog()
        }
        else{
            try {
                myRef.child( pathString: "users").child(auth.currentUser!!.uid).child( pathString: "date").setValue(DOR)
                myRef.child( pathString: "users").child(auth.currentUser!!.uid).child( pathString: "complete_date").setValue(DORS)
                //myRef.child("users").child(auth.currentUser!!.uid).child("complete_current_date").setValue(Date())
                myRef.child( pathString: "users").child(auth.currentUser!!.uid).child( pathString: "time").setValue(hours+days)

            }catch (ex:Exception){
                Toast.makeText( context: this@ParkingInfo,  text: "${ex.message}", Toast.LENGTH_LONG).show()
            }
        }
    }catch (ex:Exception){
        print(ex.message)
    }
}
```

```
156
157     private fun WarningDialog() {
158         var builder:AlertDialog.Builder = AlertDialog.Builder( context: this)
159         builder.setTitle("Info")
160         builder.setMessage("You have to choose a date in this month")
161         var alert:AlertDialog = builder.create()
162         alert.show()
163     }
164
165     fun actualiser(view: View) {
166         //update(dateView ,spot)
167     }
168
169     class MyThread:Thread{
170         var time:Long
171         var reserved:Long
172         var date:String
173         constructor(time:Long, date:String, reserved:Long):super(){
174             this.time = time
175             this.reserved = reserved
176             this.date = date
177         }
```

value-parameter **reserved**: Long

```
178         override fun run() {
179             try {
180                 var test = reserved
181
182                 while (test == reserved){
183                     if ((time - Date().time)%86400000/3600000+1 < 2){
184                         myRef.child( pathString: "reserved").setValue(reserved?.plus( other: 1))
185                     }
186                 }
187                 while (date != null){
188                     if (time == Date().time){
189                         myRef.child( pathString: "users").child(auth.currentUser!!.uid).child( pathString: "date").setValue("")
190                     }
191                 }
192             }catch (ex:Exception){
193                 print(ex.message)
194             }
195         }
196     }
197
198     override fun onBackPressed() {
199         finish()
200         super.onBackPressed()
201     }
202 }
203
```

ملخص

هدف العمل المنفذ هو استغلال أداة يمكن استخدامها لمزايا أخرى ، هذه الأداة هي كاميرا المراقبة في ساحات الانتظار ، ولهذا قمنا بتطوير نموذج يعالج البيانات الواردة من خلال هذه الكاميرات وتحويلها إلى تطبيق جوال. التي تتيح للمستخدم العثور على مكان لركن سيارته وحجز مكان في الوقت الذي يريده. تم استغلال العديد من المفاهيم مثل إنترنت الأشياء وتكنولوجيا RFID ومعالجة الصور.

كلمات مفتاحية : المواقف الذكية، معالجة الصور، تقنية RFID، إنترنت الأشياء.

Résumé

Le travail réalisé a pour but d'exploiter un outil qui peut nous servir à d'autres avantages, cet outil est la caméra de surveillance dans les parkings, pour cela on a développé un modèle qui traite les données reçues à travers ces caméras et les transférer à une application mobile qui permet à l'utilisateur de trouver son point de stationnement et de réserver une place pour le temp qu'il veut. Plusieurs notions ont été exploitées comme l'internet des objets, la technologie RFID et le traitement d'image.

Mots-clés : parking intelligent, traitement d'image, technologie RFID, internet des objets.

Abstract

The work carried aims to explore a tool that can be used for many benefits, this tool is the surveillance camera in parking lots, for this we have developed a model that processes the data received through these cameras and transfers them to a mobile application that allows the user to find his parking spot and reserve a place the time he wants. Several concepts have been used such as the Internet of Things, RFID technology and image processing.

Keywords: smart parking, image processing, RFID technology, internet of things.