N° d'ordre : 236/RC/TCO Années universitaire : 2015/2016

REPOBLIKAN'I MADAGASIKARA

Fitiavana – Tanindrazana – Fandrosoana

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR

ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE D'ANTANANARIVO

INSTITUT D'ENSEIGNEMENT SUPERIEUR D'ANTSIRABE – VAKINANKARATRA

MEMOIRE DE FIN D'ETUDE

En vue de l'obtention

Du DIPLOME de

LICENCE

En RADIOCOMMUNICATION

Par : **TOKY Rajaonarison Faniriharisoa Maxime**

« SYSTEME DE PARKING PAR UTILISATION DE LA TECHNOLOGIE DE RADIOFREQUENCE »

Soutenue le 11 Avril 2017, devant la commission d'examen composé de :

Président du jury :	Monsieur RAVONIMANANTSOA Ndaohialy Manda-Vy, Maître de Conférences
Examinateurs :	Monsieur RANDRIANANDRASANA Marie Emile, Assistant d'Enseignement et de Recherche
	Monsieur ANDY Marlon Bourgeon, Assistant d'Enseignement et de Recherche
Encadreur :	Monsieur RAKOTONDRAINA Tahina Ezechiel, Maître de Conférences

REMERCIEMENTS

Avant toute chose, je tiens à remercier Dieu tout puissant de m'avoir donné la force et de m'avoir soutenu durant cet ouvrage.

Ensuite, je tiens aussi à remercier les personnes suivantes qui m'ont encouragé et sans qui je n'aurais jamais pu accomplir l'étude que j'ai suivie à l'IES-AV, parmi lesquelles je cite :

- Monsieur RAMANOELINA Panja, Professeur Titulaire, Président de l'Université d'Antananarivo;
- Monsieur RAJAONARISON Eddie Franck, Maître de Conférences, Directeur de l'Institut d'Enseignement Supérieur Antsirabe Vakinankaratra ;
- Madame RANAIVOSOA Mamitiana Olivette, Assistant d'Enseignement et de Recherche, Responsable de la mention Automatisme Electrique Informatique Industriel à l'IESA-V,
- Monsieur RANDRIANANDRASANA Marie Emile, Assistant d'Enseignement et de Recherche, Responsable de parcours Réseau et Système RadioCommunication,;
- Monsieur RAKOTONDRAINA Tahina Ezechiel, Maître de Conférences, à qui j'exprime mes sincères reconnaissances et gratitudes, et qui, entant qu'Encadreur de ce mémoire s'est toujours montré à l'écoute et très disponible tout au long de sa réalisation.
- Tous les membres du Jury, également enseignants dans la filière Télécommunications, à savoir :
- Monsieur RAVONIMANANTSOA Ndaohialy Manda-Vy, Maître de Conférences
- Monsieur RANDRIANANDRASANA Marie Emile, Assistant d'Enseignement et de Recherche
- Monsieur ANDY Marlon Bourgeon, Assistant d'Enseignement et de Recherche

Qui ont accepté de sacrifier leur temps pour honorer de leurs présences à la présentation de ce mémoire ;

- Tous les enseignants et personnels de l'Institut d'Enseignement Supérieur Antsirabe Vakinankaratra, notamment ceux du département Télécommunication ;
- A toute ma grande famille, mes amis, mes collègues et surtout tous ceux qui ont participé, de près ou de loin, à la réalisation de ce travail ;

Je vous remercie tous et que Dieu vous bénisse et vous donnera tous le bonheur que vous souhaitez.

TABLE DES MATIERES

NOTATIONS ET ABREVIATIONS

1. Minuscule grec

ε_0 : Permittivité diélectrique du vide

μ_0 : Perméabilité magnétique du vide

C : Célérité de la lumière dans le vide

$\vec{j}_d$: Courant de déplacement.

$[\rho , \vec{j}]$: Source du champ dans le milieu considéré

2. Abréviations

BD : Base de Données

CIL : Code en Langage Intermédiaire

CLR : Common Languange Runtime

CPU : Central Processing Unit

DBMS : DataBase Managment System

EMI : Interférence électromagnétique

EPC : Electronic Product Code

EEPROM : Electrically-Erasable Programmable Read-Only Memory

FM : Frequency Modulation

FCC : Commission fédérale des communications

GPS : Global Positioning System

IDE : Integrated Development Environment

LAN : Local Area Network

LDD : Langage de Description de Données

LMD : Langage de Manipulation de Données

NFC : Near Field Communication

NIC : Network Interface Card

NN : Non Nul

PC : Personnal Computer

PK : Primary Key

PWM : Pulse Width Modulation

RF : RadioFréquence

RFID : Radio Fréquence Identification

SGBD : Système de Gestion de Base de Données

SQL : Structured Query Language

SRAM : Static Random Access Memory

UII : Unique Item Identifier

USB : Universal Serial Bus

WORM : Write Once Read Multiple

LISTES DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

Dans le monde entier, la circulation routière doit être bien ordonnée pour le bon fonctionnement de la vie des gens, et notamment celle des citadins. Ceci peut permettre d'éviter le stress et les énervements causé par les embouteillages et les accidents.

En même temps, la technologie ne cesse d'évoluer. Avec cela, notre domaine, ou plus précisément parlant, la télécommunication évolue lui aussi. Qu'est-ce qu'on peut rêver de mieux si on peut utiliser cela pour améliorer notre vie quotidienne. Par ici et par là, et notamment à Madagascar, les véhicules ne cessent de se garer sur la côté de la route, perturbant ainsi la circulation des véhicules en causant des embouteillages et des accidents; et bloquant aussi les espaces qui doivent être utilisés par les piétons, et surtout les trottoirs.

Mais comment la technologie de télécommunication peut y apporter son aide ? Ceci nous amène l'idée de concevoir ce projet portant le titre « Système de parking par utilisation de la technologie de la radiofréquence » Si on suppose qu'on ne peut pas vraiment éradiquer immédiatement ces évènements perturbateurs, on peut cependant mettre en place des systèmes de parking, qu'on va proposer d'utiliser à maximum, pour que les véhicules s'y garent et ne perturbent plus la circulation routière.

On va voir dans ce projet que cette technologie si impressionnant peut facilement apporter son coup de main. Pour cela, on va d'abord voir ce que l'on appelle technologie de radiofréquence, après, on verra les outils nécessaires pour le projet ; et viendra en fin la conception et réalisation.

CHAPITRE 1 : TECHNOLOGIE DE RADIOFREQUENCE

1.1 La radiofréquence

Comme dans ce projet, la radiofréquence peut être considérée comme l'élément de base qui nous permet de le mettre en œuvre, il serait judicieux de comprendre son fonctionnement.

1.1.1 Définition

Un signal RF est une onde électromagnétique que les systèmes de communication utilisent pour transporter l'information par l'air d'un point à un autre. Les signaux RF sont utilisés depuis de nombreuses années. Ils fournissent les moyens de transporter de la musique à des radios FM et vidéo à des téléviseurs. En fait, les signaux RF sont les moyens les plus courants pour transporter des données sur un réseau sans fil. [4]

1.1.2 Principe

Le signal RF se propage entre les antennes des émetteurs et des stations réceptrices. Comme la montre la figure ci-dessous, le signal qui alimente l'antenne a une amplitude, une fréquence et une phase. Ces attributs varient dans le temps afin de représenter l'information.

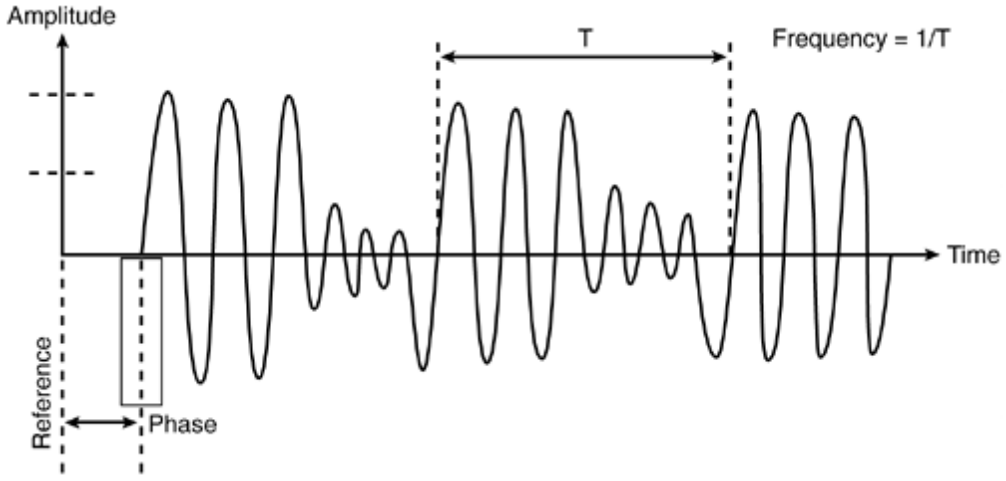


Figure1.01 *Eléments de base du signal RF : Amplitude, fréquence et phase*

L'amplitude indique la force du signal RF. La mesure d'amplitude est généralement la puissance, qui est analogue à la quantité d'effort qu'une personne a besoin d'exercer pour faire du vélo sur une distance spécifique. La puissance, en termes de signaux électromagnétiques, représente la quantité d'énergie nécessaire pour pousser le signal sur une distance particulière. À mesure que la puissance augmente, la portée augmente.

Comme un signal radio se propage à travers l'air, il subit une perte d'amplitude. Si la plage entre l'émetteur et le récepteur augmente, l'amplitude du signal diminue exponentiellement. Dans un environnement ouvert, à l'abri des obstacles, les signaux RF éprouvent ce que l'on appelle la perte d'espace libre, qui est une forme d'atténuation. L'atmosphère fait que le signal modulé s'atténue de façon exponentielle lorsque le signal se propage plus loin de l'antenne. Par conséquent, le signal doit avoir une puissance suffisante pour atteindre la distance désirée à un niveau de signal acceptable que le récepteur a besoin. Cependant, la capacité du récepteur à comprendre le signal dépend de la présence d'autres signaux RF voisins.

La fréquence décrit le nombre de fois par seconde que le signal se répète lui-même. L'unité de fréquence est Hertz (Hz), qui est le nombre de cycles se produisant chaque seconde. Par exemple, un LAN sans fil 802.11b fonctionne à une fréquence de 2,4 GHz, ce qui signifie que le signal comprend 2.400.000.000 cycles par seconde.

La phase correspond à la distance entre le signal et un point de référence. Comme convention, chaque cycle du signal s'étend sur 360 degrés. Par exemple, un signal peut avoir un déphasage de 90 degrés, ce qui signifie que la valeur de décalage est un quart (90/360 = 1/4) du signal. Une variation de phase est souvent utile pour transmettre des informations. Par exemple, un signal peut représenter un 1 binaire comme un déphasage de 30 degrés et un 0 binaire avec un décalage de 60 degrés. Un fort avantage de représenter des données en tant que déphasages est que les dégradations résultant de la propagation du signal dans l'air n'ont pas beaucoup d'impact. Les altérations affectent généralement l'amplitude et non la phase du signal.

1.1.3 Caractéristique du signal RF par rapport aux avantages et inconvénients

Par rapport à l'utilisation de signaux lumineux, les signaux RF ont les caractéristiques définies dans ce tableau.

Comparer les avantages et les inconvénients des signaux RF	
Caractéristiques avantageux	**Caractéristiques désavantageux**
Portée relativement longue, jusqu'à 20 milles lorsque la ligne de visée est possible	Réduction du débit, jusqu'à la gamme Mbps
Un bon fonctionnement dans les conditions de brume et de brouillard, à l'exception des fortes pluies, provoque de mauvaises performances	Haut potentiel pour les interférences RF provenant d'autres systèmes RF externes
Fonctionnement sans licence (uniquement pour les systèmes basés sur 802.11)	Sécurité limitée en raison de la propagation radio au-delà des installations

Tableau 1.01 *Avantage et inconvénients des signaux RF* [4]

Ces avantages rendent l'utilisation de signaux RF efficaces pour la plupart des applications de réseau sans fil. La plupart des normes de réseau sans fil, telles que 802.11 et Bluetooth, spécifient l'utilisation de signaux RF.

1.1.4 Gamme de fréquences RF

Le spectre RF est divisé en plusieurs gammes, ou bandes . A l'exception du segment de basse fréquence, chaque bande représente une augmentation de fréquence correspondant à un ordre de grandeur (puissance de 10). Le tableau suivant représente les huit bandes dans le spectre RF, montrant la fréquence et les bandes de largeur de bande. Les bandes de fréquence super haute fréquence (SHF) et de fréquence extrêmement élevée (EHF) sont souvent appelées spectre hyperfréquence.

La désignation	Abréviation	Fréquences	Longueurs d'onde en espace libre
Fréquence très basse	VLF	9 kHz à 30 kHz	33 km à 10 km
Basse fréquence	LF	30 kHz à 300 kHz	De 10 km à 1 km
Fréquence moyenne	MF	300 kHz à 3 MHz	1 km à 100 m
Haute fréquence	HF	3 MHz à 30 MHz	100 m à 10 m
Très haute fréquence	VHF	30 MHz à 300 MHz	10 m à 1 m
Ultra haute fréquence	UHF	300 MHz à 3 GHz	1 m à 100 mm
Super haute fréquence	SHF	3 GHz à 30 GHz	100 mm à 10 mm
Extrêmement Haute Fréquence	EHF	30 GHz à 300 GHz	10 mm à 1 mm

Tableau 1.02 *Les différentes fréquences RF* [3]

1.1.5 Détérioration du signal RF

Les signaux RF rencontrent des altérations, telles que l'interférence et la propagation par trajets multiples. Cela affecte les communications entre l'émetteur et le récepteur, entraînant souvent des performances inférieures et des utilisateurs malheureux.

Une interférence se produit lorsque les deux signaux sont présents à la station réceptrice en même temps, en supposant qu'ils ont la même fréquence et la même phase. C'est semblable à une personne essayant d'écouter deux autres parler en même temps. Dans cette situation, les récepteurs NIC sans fil font des erreurs lors du décodage de la signification des informations envoyées.

La Commission fédérale des communications (FCC) réglemente l'utilisation de la plupart des bandes de fréquence et des types de modulation pour éviter la possibilité d'interférence de signal entre les systèmes. Cependant, des interférences radio peuvent encore se produire, en particulier avec des systèmes fonctionnant dans des bandes exemptés de licence. Les utilisateurs sont libres d'installer et d'utiliser des équipements sans licence tels que des LAN sans fil sans coordination de l'utilisation et des interférences.

La figure ci-dessous illustre diverses formes d'interférences. L'interférence vers l'intérieur est l'endroit où les signaux externes interfèrent avec la propagation du signal radio d'un réseau sans fil. Cette interférence peut provoquer des erreurs dans les bits d'information envoyés. Le récepteur découvre finalement les erreurs, qui invoquent les retransmissions et entraîne des retards pour les utilisateurs. Une interférence significative vers l'intérieur peut se produire si un autre système radio fonctionne à proximité avec la même fréquence et le même type de modulation, tels que deux LAN radio fonctionnant dans les bandes sans licence à proximité.

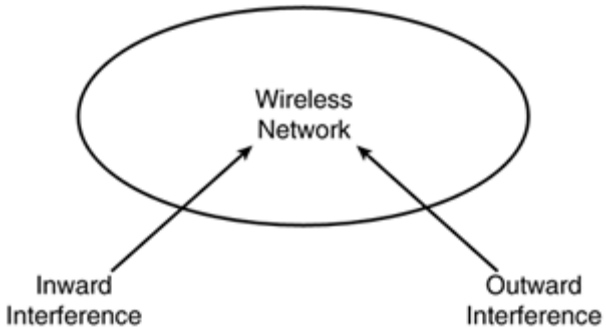


Figure1.02 *Source d'interférences : de l'intérieur ou aussi de l'extérieur*

D'autres sources d'interférences sont les téléphones sans fil, les fours à micro-ondes et les appareils Bluetooth. Lorsque ces types de dispositifs RF sont utilisés, les performances d'un réseau sans fil peuvent diminuer de façon significative en raison des retransmissions et de la concurrence sur le réseau pour l'utilisation du support. Cela nécessite une planification et la prise en compte d'autres dispositifs radio susceptibles d'interférer avec le réseau sans fil.

Une des meilleures façons de lutter contre les interférences RF est d'éliminer les sources d'interférences. Par exemple, une entreprise pourrait établir une politique pour ne pas utiliser des téléphones sans fil qui relèvent de la même bande de fréquences que le réseau sans fil. Le problème, cependant, est qu'il est souvent impossible de restreindre complètement l'utilisation d'interférences potentielles, comme les périphériques Bluetooth. Si l'interférence est un gros problème, il faut penser à choisir un réseau sans fil qui fonctionne dans une bande de fréquences qui n'est pas en conflit.

Les interférences externes se produisent lorsque les signaux provenant du système de signaux radio interfèrent avec d'autres systèmes. Comme avec l'interférence vers l'intérieur, une interférence significative vers l'extérieur peut se produire si un réseau sans fil est à proximité étroit avec un autre système. Étant donné que la puissance d'émission du réseau sans fil est relativement faible, les interférences extérieures causent rarement des problèmes importants.

1.1.6 Multipath

La propagation par trajets multiples se produit lorsque des parties d'un signal RF prennent des chemins différents lorsqu'ils se propagent depuis une source telle qu'une NIC radio vers un nœud de destination, tel qu'un point d'accès. Une partie du signal peut aller directement à la destination; Et une autre partie pourrait rebondir d'un bureau au plafond, puis à la destination. En conséquence, certaines des rencontres de signal retardent et parcourent des chemins plus longs vers le récepteur.

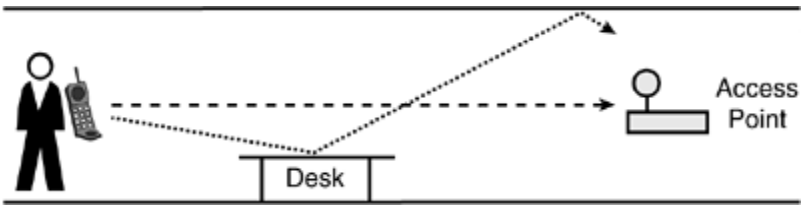


Figure1.03 *Directions différents du signal*

Les retards à trajets multiples provoquent le frottis des symboles d'information représentés dans le signal radio. Puisque la forme du signal transmet l'information, le récepteur fait des erreurs lors de la démodulation des informations du signal. Si les retards sont suffisamment importants, des erreurs de bit dans le paquet se produisent, en particulier lorsque les débits de données sont élevés. Le récepteur ne pourra pas distinguer les symboles et interpréter correctement les bits correspondants. Lorsque le multi-trajet frappe de cette manière, la station de réception détecte les erreurs par un processus de vérification d'erreur. En réponse aux erreurs de bit, la station émettrice retransmet finalement la trame de données.

1.2 RFID

1.2.1 Généralité

La radio-identification, plus souvent désignée par le sigle RFID est une méthode pour mémoriser et récupérer des données à distance.

On peut définir la RFID comme une technologie d'identification automatique qui utilise le rayonnement radiofréquence pour identifier les objets porteurs d'étiquettes lorsqu'ils passent à proximité d'un interrogateur.

La Radio Fréquence Identification est un système électronique qui permet de lire des données se trouvant sur des « étiquettes » ou « tags » en utilisant les fréquences radio. Les étiquettes RFID se distinguent des codes-barres par l'intégration d'une puce électronique associée à une antenne qui lui permet de communiquer des informations. [16]

Le terme RFID englobe toutes les technologies qui utilisent les ondes radio pour identifier automatiquement des objets ou des personnes. [12]

1.2.2 Les composants

Un système RFID se compose de transpondeurs (aussi nommés étiquettes, marqueurs, tags, identifiants...) et d'un ou plusieurs interrogateurs (aussi nommés coupleurs, base station...).

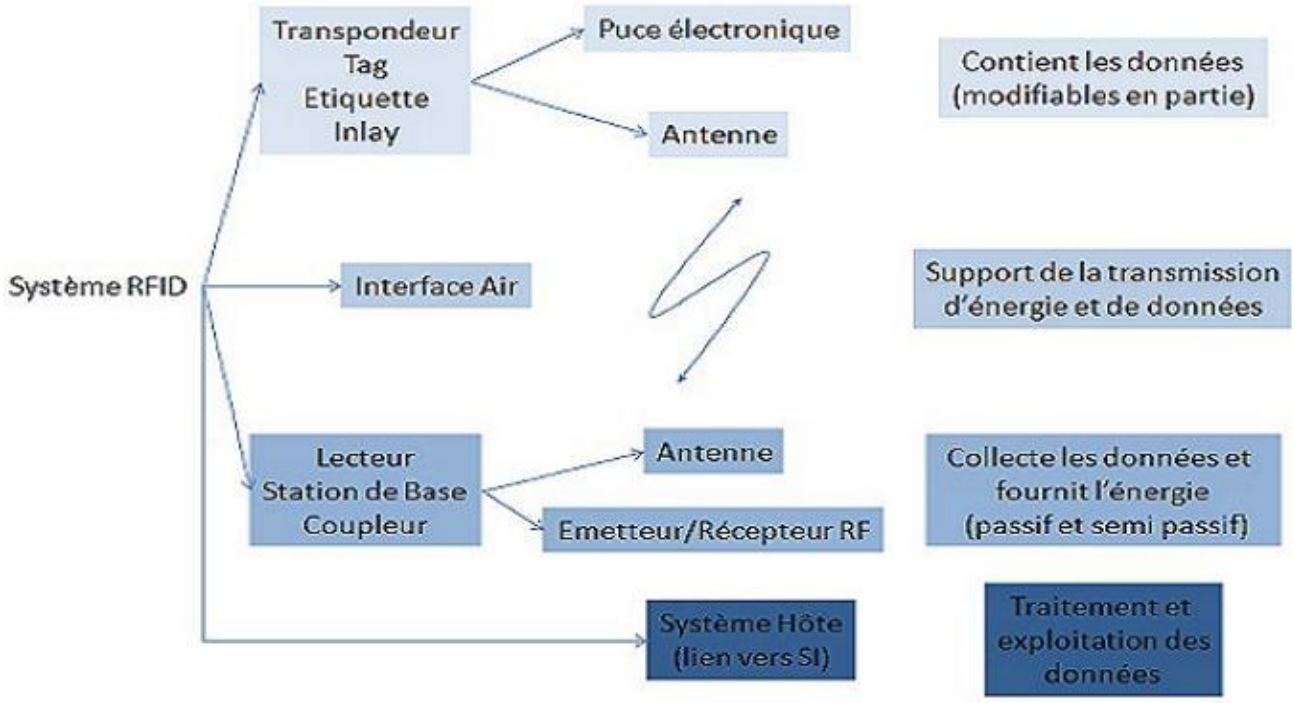


Figure1.04 *Différents composants du système RFID*

Interrogateurs RFID : Ce sont des dispositifs actifs, émetteurs de radiofréquences qui vont activer les tags qui passent devant eux en leur fournissant l'énergie dont ils ont besoin pour fonctionner. Outre de l'énergie pour l'étiquette, l'interrogateur envoie des commandes particulières auxquelles répond le tag. L'une des réponses les plus simples possibles est le renvoi d'une identification numérique. La fréquence utilisée par les interrogateurs est variable selon le type d'application visé et les performances recherchées. [12]

Tag RFID : C'est un dispositif récepteur, que l'on place sur les éléments à tracer (objet, animal...). Ils sont munis d'une puce contenant les informations et d'une antenne pour permettre les échanges d'informations.

Middleware : un système dont la fonction est d'assurer la gestion des données, des interrogateurs et de transférer les informations ad hoc aux applications de plus haut niveau.

Interface : L'interface est le support de transmission de l'énergie et des données. Dans le cadre des systèmes RFID, il s'agit de l'air. [12]

1.2.2.1 Le transpondeur

Le transpondeur, qui représente le support de données à proprement parler d'un système RFID, est généralement composé d'un élément de couplage ainsi que d'une micro-puce et la plupart du temps d'un boîtier. En dehors de la zone de réponse d'un appareil de lecture, le transpondeur, qui généralement ne possède pas d'alimentation en tension (batterie), se comporte de façon totalement neutre. Le transpondeur est activé seulement dans la zone de réponse. L'énergie nécessaire à l'exploitation du transpondeur ainsi que la cadence et les données sont transmises au transpondeur par l'unité de couplage (sans contact), la plupart du temps une bobine qui sert d'antenne, et les informations provenant de la micro-puce sont transmises à l'environnement via l'élément de couplage. La pièce essentielle du tag est la mémoire (micro-puce électronique) qui sert de support d'informations. [14]

Ces tags, sous forme de capsule de verre, de disque ou de film peuvent être accrochés à des objets (textiles, paquets) ou insérés (clé, unités logistiques réutilisables, conteneurs de transport, etc.) ou sont utilisés au sens large comme Smart Cards (cartes de téléphone, cartes bancaires ou badges).

Une autre caractéristique de différenciation des transpondeurs est la propriété d'écriture. Sur les systèmes très simples, les données, la plupart du temps un numéro (de série) sont écrites sur la puce à la fabrication et ne peuvent plus être modifiées. Sur les transpondeurs inscriptibles, les données peuvent être fournies par un appareil de lecture. Sur les systèmes programmables, l'accès en écriture et en lecture sur la mémoire ainsi qu'une éventuelle interrogation sur l'habilitation, doivent être pilotés via le support de données.

La quantité de données en mémoire va généralement de quelques octets à plusieurs Kilo-octets. Mais il existe également les transpondeurs appelés 1-bit : ce volume de données suffit pour

signaler l'appareil de lecture quand le transpondeur est dans le champ. Ceci est suffisant pour remplir des fonctions simples de surveillance. Ces transpondeurs 1-bit sont vraiment économiques, étant donné qu'aucune puce électronique n'est nécessaire à leur réalisation. C'est pour cette raison qu'ils peuvent être utilisés en grandes quantités comme sécurité antivol dans des entrepôts et des magasins.

L'étiquette RFID également nommée étiquette intelligente, étiquette à puce ou tag est un support d'identification électronique qui n'a pas besoin d'être vu pour être lu.

1.2.2.2 Etiquette RFID

Le principe de fonctionnement de la RFID consiste à donner à chaque objet porteur d'étiquette RFID une identité ou un numéro unique qui peut être communiqué à un lecteur par fréquence radio.

L'étiquette RFID est le support RFID le plus utilisé, il consiste à abriter un numéro de série ou une série de données sur une puce reliée à une antenne.

L'étiquette est activée par un signal radio émis par le lecteur RFID lui-même équipé d'une carte RFID et d'une antenne, les étiquettes transmettent les données qu'elles contiennent en retour. Cet ensemble puce/antenne qui constitue l'étiquette peut être apposé sur un objet ou inséré dans un objet.

Les catégories d'étiquette peuvent être classées de la manière suivante :

- les étiquettes passives (étiquettes en lecture seule) : elles ne possèdent pas de batterie, elles puisent leur énergie dans le signal électromagnétique du lecteur qui doit donc se situer à proximité de l'objet à identifier et à tracer ;
- les étiquettes actives (étiquettes en lecture-écriture) : elles transportent une source d'énergie comme une pile, une batterie :
    - Elles sont donc autonomes et transmettent au lecteur les informations en continu.
    - Le lecteur peut ainsi se situer à de plus grandes distances, les objets et le lecteur n'ont pas besoin d'être déplacés pour être lus.

En fonction de la puce intégrée à l'étiquette, celles-ci peuvent être :

- à usage unique : les données inscrites sur la puce ne peuvent être modifiées,
- réutilisables : la puce dispose d'une mémoire réinscriptible, les données peuvent donc être modifiées et l'étiquette peut donc servir plusieurs fois.

Le choix de l'étiquette RFID la plus appropriée pour l'identification et la traçabilité des objets/personnes visées repose sur :

- le choix de la fréquence d'utilisation : déterminé selon la nature de l'objet à identifier et de son environnement,
- l'utilisation de l'étiquette,
- le format d'étiquettes : déterminé selon les performances de lecture à avoir.

Dans notre cas, dans ce projet, on utilisera le type passif car ceci est le plus approprié. [13]

1.2.3 Fonctionnement

Rappelons que RFID est une méthode permettant de mémoriser et récupérer des données à distance. Le système est activé par un transfert d'énergie électromagnétique entre une étiquette radio et un émetteur RFID. L'étiquette radio composée d'une puce électronique et d'une antenne reçoit le signal radio émis par le lecteur lui aussi équipé d'une technologie RFID. Les composants permettent à la fois de lire et de répondre aux signaux.

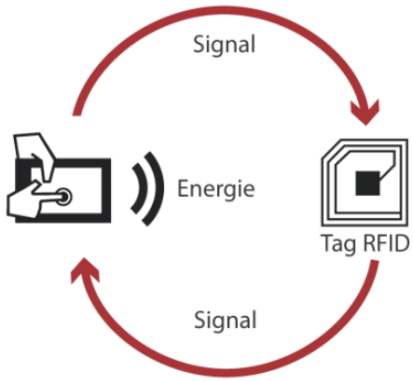


Figure1.05 *Transmission d'énergie lecteur-tag*

Aujourd'hui, la RFID se développe sous différents supports :

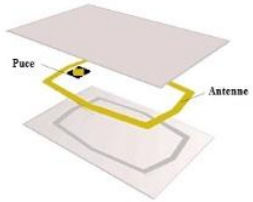
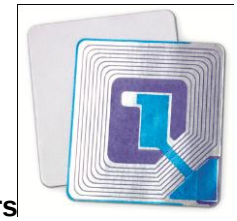
Types de support	Objectifs
Cartes RFID et badges 	Identification des personnes Paiement sans contact Contrôle d'accès en entreprise Transports Cartes de fidélité
Étiquettes et stickers 	Identification des biens Stockage et inventaire Lutte contre la contrefaçon Traçabilité des produits Promotion dans les événements
 **Bracelets**	Identification des personnes Paiement sans contact Promotion dans les événements
Porte-clés et tags 	Accès à des résidences, locaux et parking Badges d'accès en entreprise
Puces sous cutanés 	Identification d'animaux

Tableau 1.03 *Les différents types de supports RFID*

[2]

Pour chacun de ces supports, la puce peut être à usage unique (lecture seule) ou bien réinscriptible (lecture et écriture avec mémoire). Pour choisir le type puce, il faut tout d'abord choisir la fréquence en fonction de l'usage souhaité.

Pour transmettre des informations à l'interrogateur (encore appelé station de base ou plus généralement lecteur), une étiquette RFID est généralement munie d'une puce électronique associée à une antenne. Cet ensemble, appelé inlay, est ensuite packagé pour résister aux conditions dans lesquelles il est amené à vivre. L'ensemble ainsi formé est appelé tag, label ou encore transpondeur.
Les informations contenues dans la puce électronique d'un tag RFID dépendent de l'application. Il peut s'agir d'un identifiant unique (UII ou code EPC, etc.). Une fois écrit dans le circuit électronique, cet identifiant ne peut plus être modifié mais uniquement lu (WORM). Certaines puces électroniques disposent d'une autre zone mémoire dans laquelle l'utilisateur peut écrire, modifier, effacer ses propres données. La taille de ces mémoires varie de quelques bits à quelques dizaines de kilobits.

Plus précisément parlant, le système RFID fonctionne de la manière suivante :

- L'étiquette RFID (ou transpondeur ou tag) est elle-même équipée d'une puce reliée à une antenne, l'antenne permet à la puce de transmettre les informations (exemple : numéro de série) qui peuvent être lues grâce à un lecteur émetteur-récepteur.
- Une fois les informations transmises au lecteur RFID équipée d'une antenne intégrée ou externe, celui-ci n'a plus qu'à convertir les ondes-radios en données et celles-ci pourront être lues par un logiciel RFID. [13]

1.2.3.1 Le couplage tag RFID / lecteur RFID

Lorsque le lecteur composé d'un bobinage est alimenté en tension, il génère un champ magnétique. Et lorsqu'un tag (même principe pour la carte) s'en approche, par effet électromagnétique, cela génère un courant électrique, et donc une différence de potentiel**.**

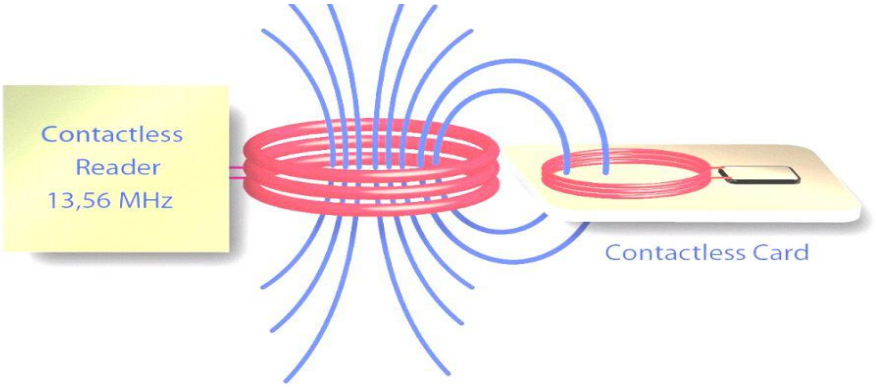


Figure1.06 *Couplage magnétique entre lecteur et carte*

Et c'est cette différence de potentiel qui permet à une puce électronique dans le tag ou dans la carte d'être alimentée en tension**.**

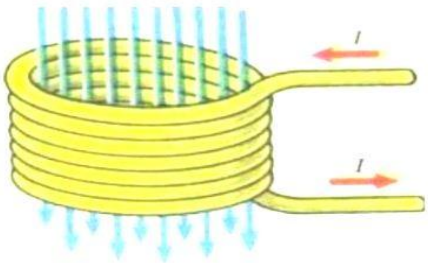


Figure1.07 *Génération de courant par effet électromagnétique*

A partir de cet instant, le lecteur et la puce utilisent leur antenne pour échanger des données, dont le numéro d'identification du tag ou bien celui d'une carte.

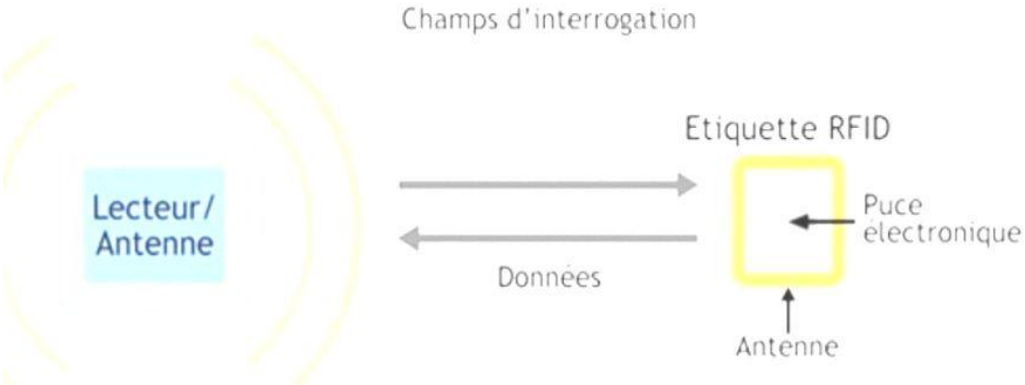


Figure1.08 *Echange de données entre tag et lecteur*

Certes, le principe de fonctionnement ci-dessus est celui qui nous intéresse car c'est celui-ci que nous établirons dans notre projet ; cependant il est nécessaire de connaître les autres liaisons entre tag et interrogateur dans les autres utilisations :

- Couplage magnétique dans le cas d'un champ proche (quelques cm à 1,5 m). L'interrogateur utilise alors des LF (Basses Fréquences) ou des HF (Hautes Fréquences). Les antennes sont alors constituées de boucles inductives.
- Couplage électrique dans le cas d'un champ lointain (jusqu'à 6m). L'interrogateur utilise alors des UHF (Ultra Hautes Fréquences) ou des SHF (Super Hautes Fréquences). Les antennes de base sont alors des dipôles ou des patchs. [10]

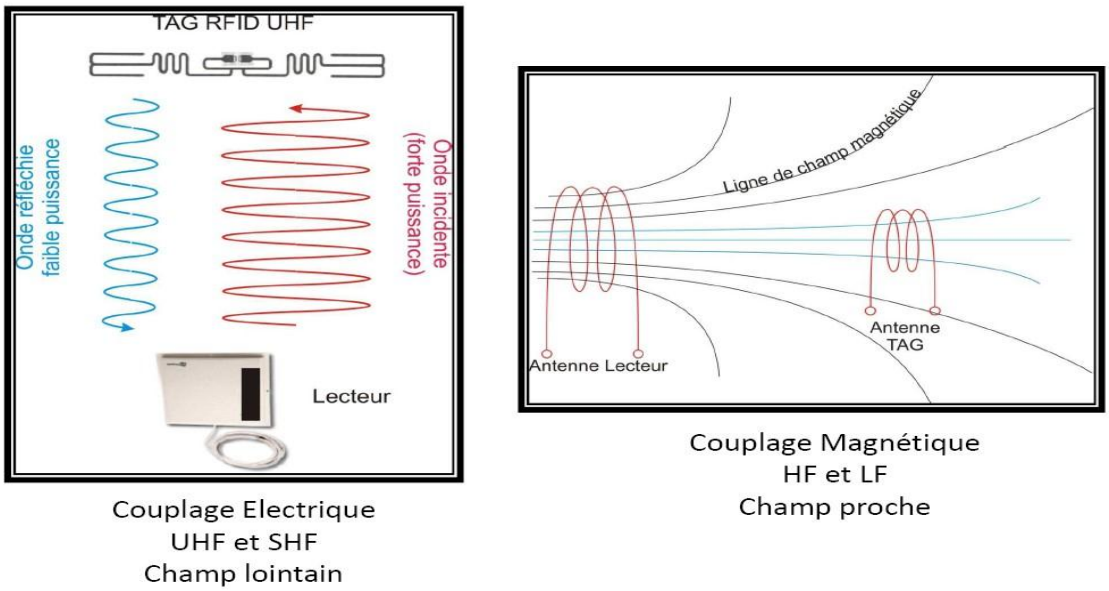


Figure1.09 *Couplage magnétique Champ lointain et champ proche*

1.2.3.2 Equations de Maxwell

En effet, les équations de Maxwell décrivent de façon mathématique comment sont liées et comment interagissent charges électriques, courants électriques, champs électriques et champs magnétiques. Pour le dire simplement, elles décrivent quantitativement les phénomènes électriques, magnétiques et lumineux.

Ces équations sont très importantes en physique et tirent leur grande élégance dans leur simplicité : seulement quatre équations pour décrire tous les phénomènes de l'électromagnétisme.

$$div\,\vec{B} = 0 \qquad (1.01)\quad \text{Maxwell-Thomson}$$

$$\overrightarrow{rot}\,\vec{E} = -\frac{\partial \vec{B}}{\partial t} \qquad (1.02)\quad \text{Maxwell-Faraday}$$

$$div\,\vec{E} = \frac{\rho}{\varepsilon_0} \qquad (1.03)\quad \text{Maxwell-Gauss}$$

$$\overrightarrow{rot}\,\vec{B} = \mu_0\left(\vec{J} + \varepsilon_0 \frac{\partial \vec{E}}{\partial t}\right) \qquad (1.04)\quad \text{Maxwell-Ampère}$$

Avec :

- La permittivité diélectrique du vide ε_0 et la perméabilité magnétique du vide μ_0 vérifient de façon exacte $\varepsilon_0\mu_0 c^2 = 1$ où c : célérité de la lumière dans le vide.

  On a fixé $\mu_0 = 4\pi 10^{-7}\ H.m^{-1}$

- Le terme $\vec{j}_d = \varepsilon_0 \frac{\partial \vec{E}}{\partial t}$, homogène à une densité volumique de courant, est appelé courant de déplacement.

- Les équations de Maxwell-Gauss et Maxwell-Ampère relie le champ électromagnétique $[\vec{E}, \vec{B}]$ à ses sources ρ et $\vec{j}$

- Les équations de Maxwell-Thomson et Maxwell-Faraday sont des relations de structure du champ électromagnétique, elles ne font pas intervenir les sources ρ et $\vec{j}$.

- Les équations de Maxwell sont valables dans n'importe quel milieu. Elles ne sont cependant utilisables sous cette forme que si l'on sait expliciter la source $[\rho, \vec{j}]$ du champ dans le milieu considéré, c'est le cas :

  - Dans le milieu non diélectrique et non magnétique, en particulier les conducteurs ohmiques.
  - Dans le vide
  - Dans les plasmas [1]

1.2.3.3 La communication par la puce

La fréquence est la caractéristique qui permet d'établir la communication entre la puce et l'antenne. Toutes les puces sur le marché n'ont donc pas la même fonctionnalité. [15]

Les puces se différencient en grande partie par la fréquence de fonctionnement et la distance de lecture. Plus la fréquence est élevée, plus la distance de lecture s'agrandit. En fonction de ces éléments, la puce sera plus ou moins puissante et plus onéreuse.

Trois types de fréquences sont utilisés pour les puces RFID :

- Basse fréquence (125Khz),
- Haute (13,56 Mhz)
- Très haute fréquence (UHF).

Types de fréquence	Fréquence de fonctionnement	Distance de lecture (m)	Taux de transfert	Normes
Basse fréquence	< 135 kHz	0.5	1kb/s	ISO 142231 ISO 18000-2
Haute fréquence	13,56 Mhz	1	25kb/s	ISO 14443 ISO 15693 ISO 18000-3
Très haute fréquence	863 à 915 Mhz	3 à 6	28kb/s	ISO 18000-6

Tableau 1.04 *Les Gammes de fréquences utilisées en RFID*

Plusieurs fabricants se partagent le marché et proposent des puces de plus en plus performantes. Cette technologie est aujourd'hui standardisée et présente dans beaucoup d'objets du quotidien.

a. Les capacités de la puce RFID

Cette technologie se décline en trois versions :

- La RFID passive
- La RFID semi-passive
- La RFID active

La RFID passive fonctionne en lecture seule puisque la puce ne possède pas de batterie et doit être déplacé vers le lecteur pour être lu. Un puissant signal électromagnétique lui est alors envoyé, ce qui permet d'activé la puce RFID et de lire les informations qu'elle contient.

En revanche, la RFID active fonctionne avec une source d'énergie telle qu'une petite pile ou une batterie, ce qui permet de lire la carte à plus longue distance. Cette technique est principalement utilisée pour la traçabilité de personnes, de véhicules ou encore pour la traçabilité logistique.

Tout comme la RFID active, la RFID semi-passive est alimenté par une source d'énergie. Cependant, la batterie alimente la puce RFID à des intervalles de temps réguliers. Celle-ci n'envoie pas de signal. Cette technologie s'avère utile pour la traçabilité alimentaire notamment pour enregistrer les changements de température durant le transport.

Il existe différentes classes concernant la RFID :

Classe	Tag	Fonction	Avantages / inconvénients
Classe 0 Classe 1	Passif	Lecture de l'identifiant unique	Moins onéreux que les tags actifs, utile pour un gros volume de marchandises pour être lues à courte distance. Cependant, la distance de lecture est aussi un frein car le lecteur doit se trouver à proximité.
Classe 2	Passif	Fonctions additionnelles : lecture, écriture avec mémoire	
Classe 3	Semi-passif	Tags assistés par une batterie	Plus performant et moins onéreux que la RFID active. En revanche, l'incertitude repose sur la fiabilité en cas de traçabilité.
Classe 4	Actif	Communication sans transiter par un serveur central	Technologie autonome grâce à son énergie propre qui permet une lecture à longue distance. Les inconvénients sont : le coût des étiquettes et leur durée limitée, la faible sécurité des ondes émisses et son impact sur la santé.
Classe 5	Interrogateur	Alimentent les tags de classe 0 à 3 et communiquent avec les tags de classe 4.	

Tableau 1.05 *Les différentes classes de RFID*

1.2.4 Avantages de l'identification par radiofréquence

Les avantages des transpondeurs se situent en particulier dans la grande fiabilité, même en cas d'influences environnementales extrêmes (plus grande solidité, robustesse et réutilisable, durée de vie plus longue), dans le contact visuel avec l'appareil de lecture qui n'est pas nécessaire, dans la possibilité d'une grande capacité de mémoire et de pouvoir enregistrer plusieurs supports de données simultanément dans un seul processus de lecture

Une exigence importante est de pouvoir saisir en bloc, c'est-à-dire, en une seule lecture, plusieurs tags se trouvant dans le champ électromagnétique. En ce qui concerne le nombre minimum de tags nécessaire pour une solution standard qu'un système RFID doit pouvoir traiter, une quantité de quelques centaines de tags est prévue comme ordre de grandeur. Il faut prévoir des règles anticollision du point de vue technique dans le protocole de transfert pour la réalisation de la saisie en bloc.

1.2.5 Inconvénients de l'identification par radiofréquence

Ces inconvénients se situent dans les coûts; absorption et réflexion par les métaux, les matériaux et liquides conducteurs ; sensibilité aux champs électromagnétiques (EMI), aucune compatibilité par des régulations nationales et régionales; influence et effets sur l'homme est la preuve que les données ont été lues correctement. Cependant des tags plus économiques sans puce sont produits entre temps, environ 7 à 10 cents US, sinon à partir de 20 cents, sans limite supérieure, selon l'utilisation, les données, la portée. [4]

1.3 Conclusion

Si on résume un peu, la radiofréquence est une onde électromagnétique, qui offre la possibilité d'établir une communication par l'air. Ses caractéristiques et ses gammes de fréquence définissent les avantages et les inconvénients de son utilisation. La RFID est une technologie qui utilise cette radiofréquence pour faire l'identification automatique des transpondeurs lorsque ceux-ci passent à proximité d'un interrogateur. Cette communication se fait grâce au transfert d'énergie électromagnétique entre les composants RFID utilisant la fonctionnalité des puces.

CHAPITRE 2 : OUTILS NECESSAIRES POUR LE PROJET

2.1 Base de Données

2.1.1 Définitions

A première vue, on peut définir une base de données comme étant un ensemble d'informations, ou fichiers si l'on veut, partagé par plusieurs utilisateurs. Ces informations sont interrogées et mises à jour par l'intermédiaire d'un logiciel. [17]

Exemples d'applications :

- Annuaire électronique
- Catalogue électronique d'une bibliothèque

Une base de données (son abréviation est BD, en anglais DB) est une entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondance possible. Ces données doivent pouvoir être utilisées par des programmes, par des utilisateurs différents.

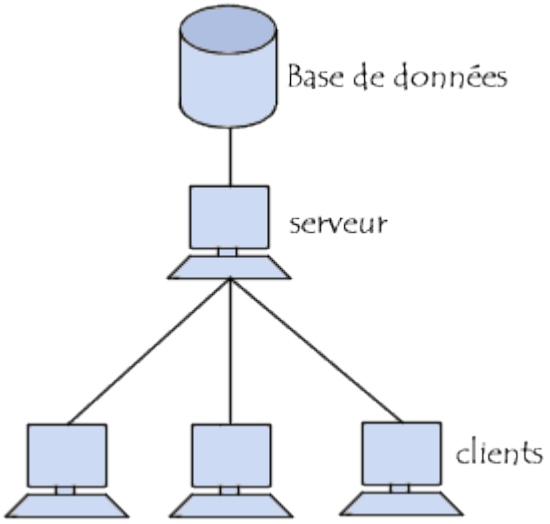


Figure 2.01 *Illustration de la relation entre BD et Clients*

Plus techniquement parlant, une base de données est un ensemble structuré de données enregistrées sur des supports accessibles par l'ordinateur (Stockage sur disque) pour satisfaire simultanément plusieurs utilisateurs (partage des données) de manière sélective (Confidentialité) en un temps opportun, ce qui fait sa performance. [5]

2.1.2 Utilité

Une base de données permet de mettre des données à la disposition d'utilisateurs pour une consultation, une saisie ou bien une mise à jour, tout en s'assurant des droits accordés à ces derniers. Cela est d'autant plus utile que les données informatiques sont de plus en plus nombreuses.

Une base de données peut être locale, c'est-à-dire utilisable sur une machine par un utilisateur, ou bien répartie, c'est-à-dire que les informations sont stockées sur des machines distantes et accessibles par réseau.

L'avantage majeur de l'utilisation de bases de données est la possibilité de pouvoir être accédées par plusieurs utilisateurs simultanément. [17]

2.2 SGBD

2.2.1 Description

Afin de pouvoir contrôler les données ainsi que les utilisateurs, le besoin d'un système de gestion s'est vite fait ressentir. La gestion de la base de données se fait grâce à un système appelé SGBD ou en anglais DBMS.

Le SGBD, comme son nom l'indique (Système de Gestion de Base de Données), est le logiciel qui permet d'interagir avec la base de données.

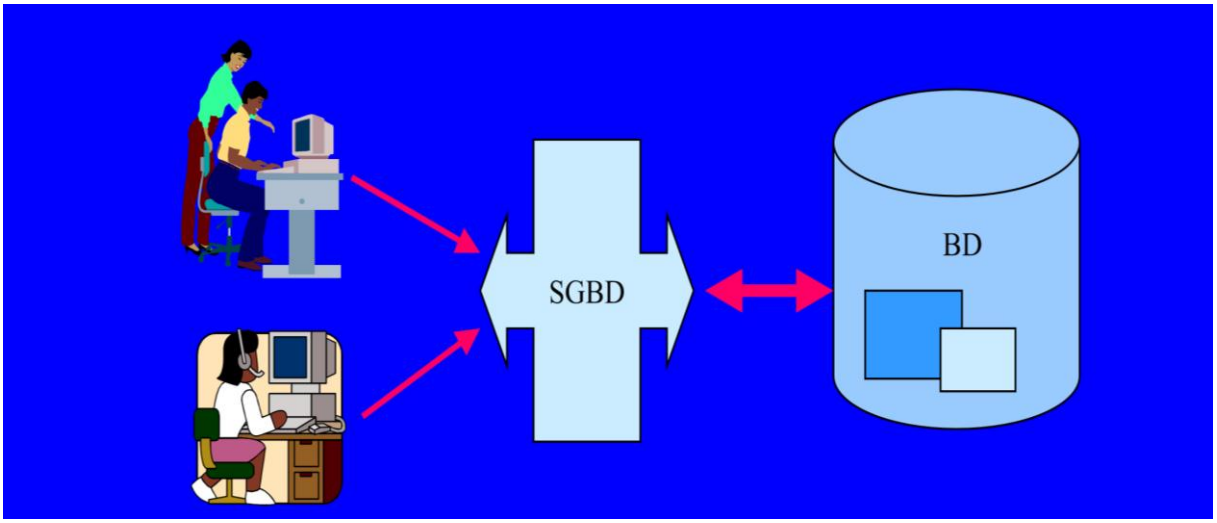


Figure 2.02 *Interaction entre utilisateurs et BD*

Généralement parlant, le SGBD est un ensemble de services (applications logicielles) permettant de gérer les bases de données, c'est-à-dire :

- permettre l'accès aux données de façon simple
- autoriser un accès aux informations à de multiples utilisateurs
- manipuler les données présentes dans la base de données (insertion, suppression, modification)

Le SGBD permet à des utilisateurs de créer et maintenir une base de données. Les activités supportées sont la définition d'une base de données (spécification des types de données à stocker), la construction d'une base de données (stockage des données proprement dites) et la manipulation des données (principalement ajouter, supprimer, retrouver des données).

2.2.2 Objectif du SGBD

L'objectif principal du SGBD est de faciliter la représentation et la description de données. Plus besoin de travailler directement sur les fichiers physiques (tels qu'ils sont enregistrés sur disque). Un SGBD nous permet de décrire les données et les liens entre elles d'une façon logique sans se soucier du comment cela va se faire physiquement dans les fichiers. On parle alors d'image logique de la base de données, ou aussi description logique ou conceptuelle ou encore de schéma.

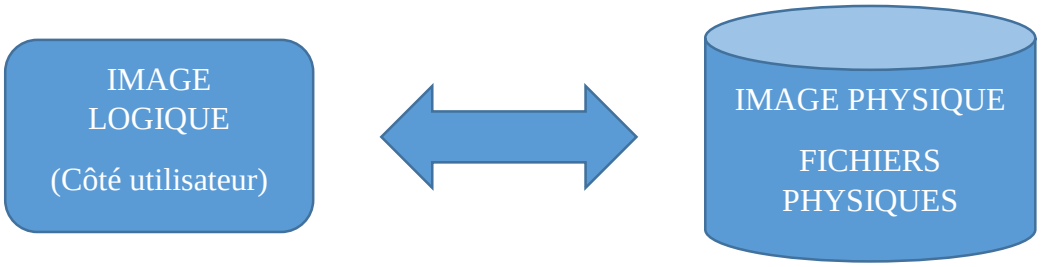


Figure 2.03 *Liens entre image logique et physique*

Le SGBD permet aussi, évidemment de faciliter la manipulation en travaillant directement sur le schéma logique. On peut insérer, supprimer, modifier des données directement sur l'image logique. Le SGBD va s'occuper de faire le travail sur les fichiers physiques.

Le SGBD permet aussi l'ajout des contraintes permettant d'avoir à tout instant des données cohérentes par exemple l'âge d'une personne supérieur à zéro, salaire supérieur à zéro, etc. Dès que l'on essaie de saisir une valeur qui ne respecte pas cette contrainte, le SGBD le refuse. Le SGBD nous fournit aussi une efficacité des accès (Temps de réponse & débit global)

2.2.3 Fonctions

Ce système de gestion permet la description des données ; c'est-à-dire la codification et la structuration, grâce au Langage de Description de Données (LDD). [5]

Il nous permet aussi de faire la manipulation et restitution des données comme l'insertion, les mises à jour et l'interrogation.

Le SGBD nous permet aussi le contrôle des données (partage, intégrité, confidentialité, sécurité).

En tout, le SGBD met alors à notre disposition 3 fonctions avantageuses. Ces fonctions peuvent être mise en œuvre à l'aide d'un Langage de Manipulation de Données (LMD) comme le langage S.Q.L (Structured Query Language) qui est le langage standard.

2.2.4 Définition et description des données

Il y a 3 niveaux de description des données :

- Au niveau logique
- Au niveau physique
- Au niveau externe

2.2.4.1 Définition et description des données au niveau logique (conceptuel)

C'est à ce niveau qu'on considère la description des objets, des propriétés des objets (attributs), des liens entre les objets, des contraintes.

Cette description est faite selon un modèle de données. Un modèle de données est un ensemble de concepts permettant de décrire la structure d'une base de données. La plupart des modèles de données incluent des opérations permettant de mettre à jour et questionner la base. Le modèle de données le plus utilisé est le modèle relationnel.

Cette description va donner lieu à un schéma de base de données. Un schéma de base de données se compose d'une description des données et de leurs relations ainsi que d'un ensemble de contraintes d'intégrité.

2.2.4.2 Définition et description des données au niveau physique :

Dans ce niveau, il y a la description informatique des données et de leur organisation : en terme de fichiers, d'index et de méthodes d'accès. Il nous forme le passage du modèle logique au modèle physique tend à être assisté par le SGBD : transparent et/ou semi-automatique. Ceci a pour objectifs d'optimiser les performances.

2.2.4.3 Définition et description des données au niveau externe

C'est dans ce niveau qu'on a la description des données vues par un utilisateur ou un groupe d'utilisateurs. Ce niveau a pour objectifs la simplification et la confidentialité des données.

2.2.5 Manipulation des données

En ce qui concerne la manipulation des données, on a la possibilité de faire :

- Une insertion
- Une suppression
- Une modification
- Une interrogation, c'est-à-dire rechercher des données via des requêtes.

La manipulation des données est mise en œuvre à l'aide d'un LMD ou bien Langage de Manipulation de Données. Comme c'est déjà dit ci-dessus, S.Q.L est le langage standard de manipulation de base de données.

2.2.6 Contrôle

Dans un SGBD, il y a un partage de données, c'est-à-dire que plusieurs utilisateurs ont accès à la même information en même temps. Le SGBD inclut un mécanisme de contrôle de la concurrence basé sur des techniques de verrouillage des données, pour éviter par exemple qu'on puisse lire une information qu'on est en train de mettre à jour.

L'intégrité des données est établie grâce à la définition de contraintes sur les données. Le SGBD veille à ce que toutes les contraintes soient vérifiées à chaque manipulation d'une donnée.

Cependant, quand plusieurs utilisateurs peuvent utiliser en même temps une base de données, le problème qui se pose est la confidentialité des données. Des droits doivent être gérés sur les données, droits de lecture, mise à jour, création.

Une base de données est souvent vitale dans le fonctionnement d'une organisation, et il n'est pas tolérable qu'une panne puisse remettre en cause son fonctionnement de manière durable. Les SGBD fournissent des mécanismes pour assurer cette sécurité.

2.2.7 Architecture général des SGBD

Comme on parle de base de données, le SGBD est basé sur une architecture Client-Serveur. Les données sur le serveur sont partagées entre les clients, le SGBD fournit des interfaces graphiques sur la station de travail personnelle des utilisateurs, la communication se fait par des protocoles standardisés, et enfin, les clients et le serveur communiquent par des requêtes avec réponses.

2.2.8 Démarche de construction d'une BD

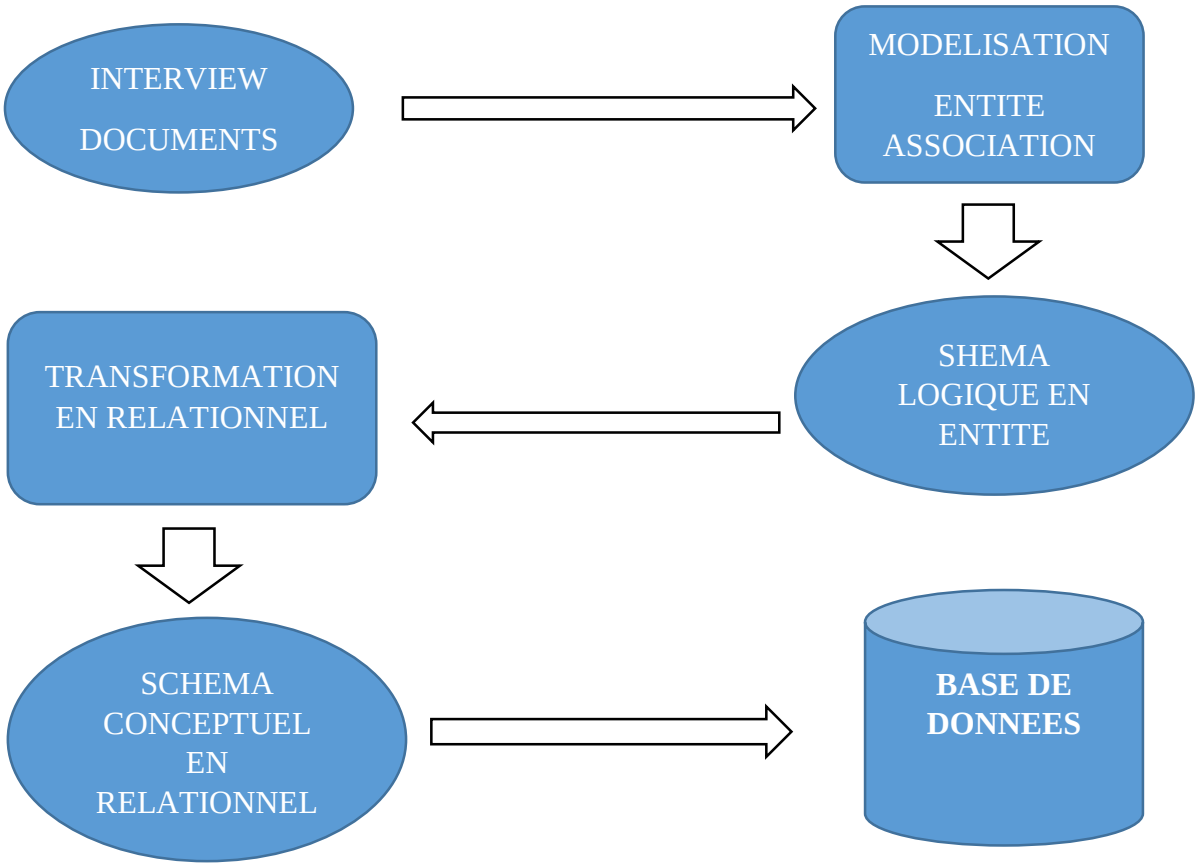


Figure 2.04 *Etape de mise en place d'une BD*

2.2.9 Les principaux SGBD

Les principaux systèmes de gestion de bases de données sont les suivants :

- Borland Paradox
- Filemaker
- IBM DB2
- Ingres
- Interbase
- Microsoft SQL server
- Microsoft Access
- Microsoft FoxPro
- Oracle
- Sybase
- MySQL
- PostgreSQL
- Msql
- SQL Server 11
- NoSQL

Il existe certainement plusieurs types de système de gestion de base de données, et chacun a sa particularité. Mais pour ce projet, on utilise MySql WorkBench car il a été particulièrement conçu pour être utilisé dans le domaine de travail, pour tout le monde. MySql WorkBench est aussi gratuit et plus adapté à ce projet.

2.2.10 Le modèle relationnel

Ils existent de nombreuses modèles de SGBD comme :

- modèle hiérarchique
- modèle réseau
- modèle relationnel
- modèle objet

Cependant, le modèle relationnel demeure le plus utilisé et est la base de tous les modèles. Il est préférable alors de s'informer un peu plus sur ce modèle.

2.2.10.1 Généralités sur le modèle relationnel

a. Modèle de données

Un modèle est un ensemble d'outils utilisés pour décrire et manipuler des données.

b. Description du modèle relationnel

Ce modèle a été créé par CODD. La majorité des SGBD actuels sont basés sur ce modèle. Il dispose d'un Langage de Description des Données (LDD) et d'un Langage de Manipulation des Données (LMD).

Le modèle relationnel est basé sur le principe simple qu'un seul concept (relation ou table) décrit les données et les liens entre ses données. Le langage de description et de manipulation des données est toujours le SQL.

2.2.10.2 Concepts du modèle

Le modèle décrit une vision tabulaire du relationnel. Ceci implique que les données (le schéma logique) sont représentées dans une table.

Avec :

- Attribut : nom donné à une colonne d'une table. C'est-à-dire que la première ligne de la table comporte ses attributs.
- Domaine : ensemble de valeurs possibles prises par les attributs.
- Nom de la table : nom de la relation qui est comme le titre de la relation qu'on veut représenter.
- Tuple (ou n-uplet) : nom donné à une ligne comportant des valeurs saisies.
- Extension d'une table : le contenu de la table ; c'est-à-dire tous les tuples.
- Cardinalité : nombre de tuples de la relation. La cardinalité peut alors être définie comme le nombre des lignes comportant des valeurs saisies.
- L'ordre des lignes et des colonnes n'est pas significatif
- Une case représente une valeur

a. Clé primaire

C'est le groupe d'attributs minimum qui détermine un tuple d'une manière unique dans la table.

Une clé est défini par un attribut qui peut déterminer, par lui seul et de façon unique, des lignes pour la table.

La clé se détermine par rapport à toutes les valeurs possibles de l'attribut (ou les attributs) formant la clé primaire, et surtout pas par rapport aux valeurs déjà saisies.

Il est important savoir que toute table doit obligatoirement avoir une clé primaire

b. Clé Étrangère

On appelle Clé étrangère toute clé primaire apparaissant dans une autre table. Une clé étrangère est soulignée en pointillé ou mise en italique ou les deux.

La notion de clé est toujours liée à une table, un attribut (ou groupe d'attributs) et clé primaire, ou clé étrangère dans une table donnée.

c. Schéma d'une table

Le schéma d'une table, appelé aussi le schéma en intention, comporte le nom de la relation, ses attributs, le format des valeurs des attributs et la clé primaire.

La clé primaire est souvent soulignée ou mise en gras, ou les deux en même temps.
Forme générale d'un schéma d'une table :

Nom de table (**clé primaire** : format, Attribut_1 : format, Attribut_2 : format, Attribut_3 : format, …)

Le schéma d'une base de données est composé de l'ensemble des schémas des tables (relations) définies dans cette base de données.

2.2.10.3 Problème de Redondance des données

On appelle redondance la répétition des informations dans la table. Un des objectifs des SGBD est (de nous permettre) de représenter les données avec le moins de redondance possible.
Pour comprendre l'élimination de la redondance, il est plus simple de prendre un exemple de table.

Prenons comme exemple concret, la table : « OUVRAGES »

OUVRAGES

Côte	Titre	NbExem.	Année	Thème	Editeur	NomAuteur	PrénomAuteur
12TA1	Réseaux informatiques	10	1998	Interconnexion, réseaux, Internet	Eyrolles	Tanenbaum	Henri
13GO1	Algorithmes génétiques	5	1994	AG, informatique évolutionniste,	Addison Wesley	Goldberg	Stephen
13GO1	Algorithmes génétiques	5	1994	AG, informatique évolutionniste,	Addison Wesley	Holland	John
15TA2	Système d'exploitation	6	1993	UNIX, SE, ordinateurs	Eyrolles	Cardy	Ronald
15TA2	Système d'exploitation	6	1993	UNIX, SE, ordinateurs	Eyrolles	Dumar	Eric
15TA2	Système d'exploitation	6	1993	UNIX, SE, ordinateurs	Eyrolles	Tannenbaum	Henri

Tableau 2.01 *Exemple de table*

Pour éliminer les répétitions nous allons, dans un premier temps, construire une table auteur comportant tous les auteurs.

La table auteur est décrite par AUTEURS (NumAuteur, NomAuteur,PrénomAuteur). Nous avons rajouté l'attribut NumAuteur pour représenter la clé. NumAuteur est un numéro qui peut être donné automatiquement par le SGBD(Auto-incrémentation).

AUTEURS

Num Auteur	Nom Auteur	Prénom Auteur
1	Tanenbaum	Henri
2	Goldberg	Stephen
3	Holland	John
4	Cardy	Ronald
5	Dumar	Eric

Tableau 2.02 *Exemple de table de suppressions de redondance*

La table OUVRAGES peut se réduire à cela :

OUVRAGES

Côte	Titre	NbExem.	Année	Thème	NomEditeur	NmAuteur
12TA1	Réseaux informatiques	10	1998	Interconnexion, réseaux, Internet	Eyrolles	1
13GO1	Algorithmes génétiques	5	1994	AG, informatique évolutionniste,	Addison Wesley	2
13GO1	Algorithmes génétiques	5	1994	AG, informatique évolutionniste,	Addison Wesley	3
15TA2	Système d'exploitation	6	1993	UNIX, SE, ordinateurs	Eyrolles	4
15TA2	Système d'exploitation	6	1993	UNIX, SE, ordinateurs	Eyrolles	5
15TA2	Système d'exploitation	6	1993	UNIX, SE, ordinateurs	Eyrolles	1

Tableau 2.03 *Exemple de table subissant une suppression de données*

Cette représentation nous permet effectivement de réduire la table OUVRAGES il n'y a que le numéro de l'auteur au lieu du nom et du prénom, mais Il y a toujours des redondances. La redondance provient, dans notre exemple, du fait qu'un OUVRAGE peut avoir plusieurs auteurs. Pour éliminer ces redondances, on va construire une autre table « ECRIT » qui permet de relier les OUVRAGES et leurs AUTEURS.

Rappelons qu'un des intérêts d'un SGBD est sa possibilité de créer des liens entre les objets. On va définir le schéma de la table ECRIT comme ceci : ECRIT (cote, NumAuteur)

Il suffit donc de prendre les clés primaires des tables OUVRAGES et AUTEURS et former une nouvelle nouvelle table, en l'occurrence ECRIT.

ECRIT

N u m A u t e u r	C o t e
1	1 2 T A 1
2	1 3 G O 1
3	1 3 G O 1
4	1 5 T A 2
5	1 5 T A 2
1	1 5 T A 2

Tableau 2.04 *Exemple de deuxième table permettant la suppression de données*

Après ces créations de ces deux tables, la base de données décrivant les OUVRAGES sera composée des tables suivantes :

- AUTEURS(**NumAuteur**, Nom, Prénom)
- OUVRAGES(**cote**, Titre, NbExemplaire, Année, Editeur, Thème)
- ECRIT (**cote, NumAuteur**)

Notons qu'on a supprimé l'attribut NumAuteur de la table OUVRAGES.

2.2.10.4 Contraintes d'intégrités

a. Définition

Un des avantages des bases de données par rapport à une gestion de fichiers traditionnelle est la possibilité d'intégrer des contraintes (conditions d'insertion de tuples) que doivent vérifier les données à tout instant. Ceci est possible grâce à la notion de contraintes d'intégrité.

Les contraintes d'intégrités sont donc des assertions qui doivent être vérifiées à tout moment par les données contenues dans la base de données (exemples : un salarié doit avoir un nom, un numéro et ce numéro doit être une valeur numérique sinon l'insertion de ce salarié dans la base de données est refusée).

b. Types

Il existe 3 types de contraintes d'intégrités obligatoires :

- Contrainte de clé : une relation doit posséder une clé primaire
- Contrainte d'entité : un attribut d'une clé ne doit pas posséder de valeurs nulles (vides)
- Contrainte de référence (pour les clés étrangères), c'est une contrainte exprimée entre deux tables. Tout tuple d'une relation faisant référence à une autre relation doit se référer à un tuple qui existe. Intuitivement, cela consiste à vérifier que l'information utilisée dans un tuple pour désigner un autre tuple est valide, notamment si le tuple désigné existe bien. En d'autre terme, quand on désigne un attribut comme clé étrangère, les seules valeurs que peut prendre cet attribut sont celles qui sont déjà saisies dans la table qu'il référence.

On peut considérer comme contrainte optionnelle la contrainte de domaine qui est liée au domaine de définition d'un attribut.

Les contraintes d'intégrité sont vérifiées, ou bien exécutées, à chaque mise à jour de la base de données (ajout, suppression ou modification d'un tuple). Si, lors d'une mise à jour une contrainte n'est pas satisfaite, cette mise à jour ne peut pas avoir lieu.

2.3 ARDUINO

2.3.1 Présentation de la carte

2.3.1.1 Présentation générale

Les modules arduino sont des plates-formes de prototypage microcontrôlées "open-source". L'arduino est principalement composé d'un microcontrôleur qui change selon le modèle utilisé.

C'est un circuit imprimé comportant tous les composants électroniques nécessaires pour faire fonctionner un microcontrôleur associé à une interface USB lui permettant de communiquer avec un ordinateur.

2.3.1.2 Microcontrôleur

Un microcontrôleur est l'équivalent d'un petit ordinateur. Il contient un microprocesseur (unité centrale) associé à différentes mémoires et des interfaces lui permettant de communiquer avec l'extérieur, et une horloge pour cadencer l'ensemble.

On peut aussi dire que c'est une petite unité de calcul accompagné de mémoire, de ports d'entrée/sortie et de périphériques permettant d'interagir avec son environnement. Parmi les périphériques, on recense généralement des Timers, des convertisseurs analogique-numérique, des liaisons Séries, etc. On peut comparer un microcontrôleur à un ordinateur classique, mais sans système d'exploitation et avec une puissance de calcul considérablement plus faible.

Les microcontrôleurs sont inévitables dans les domaines de l'informatique embarquée, de l'automatique et de l'informatique industrielle. Ils permettent de réduire le nombre de composant et de simplifier la création de cartes électroniques logiques. [6]

2.3.2 Utilisation dans le projet

Il y a de nombreuses versions de l'arduino, choisie en fonction des finalités qu'on veut exécuter avec le programme. Pour notre projet, on va utiliser le modèle Méga 2560 de cette carte. Cette version est architecturée autour d'un CPU ATmega2560 et est désignée pour des programmes plus complexes (surtout par rapport aux programmes réalisés avec l'arduino UNO). En effet, l'arduino méga possède 54 Pin entrée/sortie digitaux (dont 14 peuvent être utilisés comme des sorties PWN), 16 Pin analogiques, et un espace mémoire spacieux pour le programme. L'ensemble des entrées/sorties (PIN) de la platine sont disponibles sur des connecteurs femelles présents sur les bords de la platine.

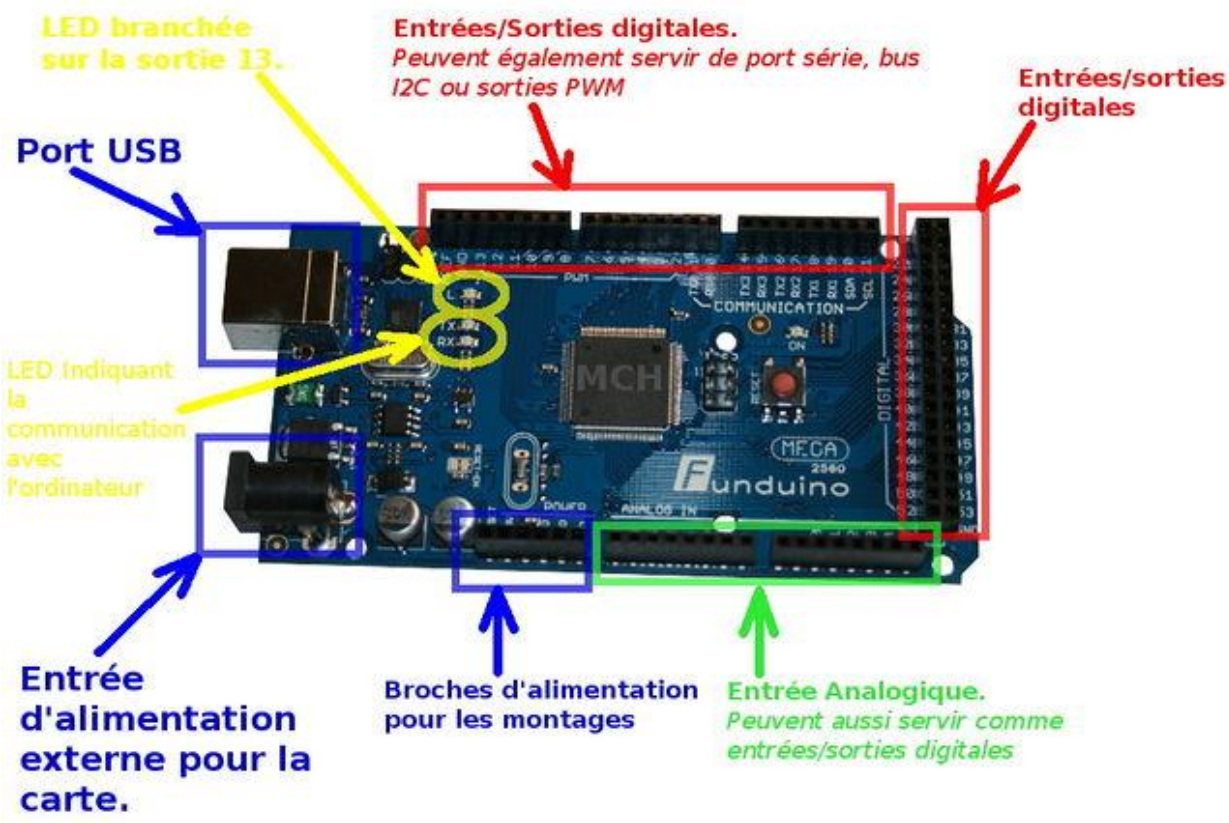


Figure 2.05 *Image légendée d'un Arduino*

Voici les caractéristiques de la carte arduino Méga 2560 :

- Microcontrôleur ATmega2560
- Tension de fonctionnement 5 V

– Tension d'alimentation (recommandée)	7- 1 2 V
– Tension d'alimentation (limites)	6 - 20V
– Nombre d'E/S	54 (dont 14 pouvant générer des signaux PWM)
– Nb ports "Analogique/Numérique"	16
– Courant max. par E/S	40 Ma
– Courant pour broches 3.3 V	50 Ma
– Mémoire Flash	256 KB (ATmega328) dont 8 KB utilisé par le bootloader
– SRAM	8 KB (ATmega328)
– EEPROM	4 KB (ATmega328)
– Vitesse horloge	16 MHz

2.3.3 Les shields

Pour la plupart des projets, il est souvent nécessaire d'ajouter des fonctionnalités aux cartes Arduino. Plutôt que d'ajouter soit même des composants extérieurs (sur une platine d'essai, circuit imprimé, etc.), il est possible d'ajouter des shields. Un shield est une carte que l'on connecte directement sur la carte Arduino qui a pour but d'ajouter des composants sur la carte. Ces shield viennent généralement avec une librairie permettant de les contrôler. On retrouve par exemple, des shields Ethernet, de contrôle de moteur, lecteur de carte SD, etc.

Le principal avantage de ces shields est leurs simplicités d'utilisation. Il suffit de les emboiter sur la carte Arduino pour les connecter, les circuits électroniques et les logiciels sont déjà faits et on peut en empiler plusieurs. C'est un atout majeur pour ces cartes pour pouvoir tester facilement de nouvelles fonctionnalités.

2.3.4 Présentation du logiciel

2.3.4.1 Le logiciel Arduino

Ce logiciel est classé dans la catégorie des IDE. Un IDE libre et gratuit est distribué sur le site d'Arduino.

L'interface de l'IDE Arduino est plutôt simple, il offre une interface minimale pour développer un programme sur les cartes Arduino.

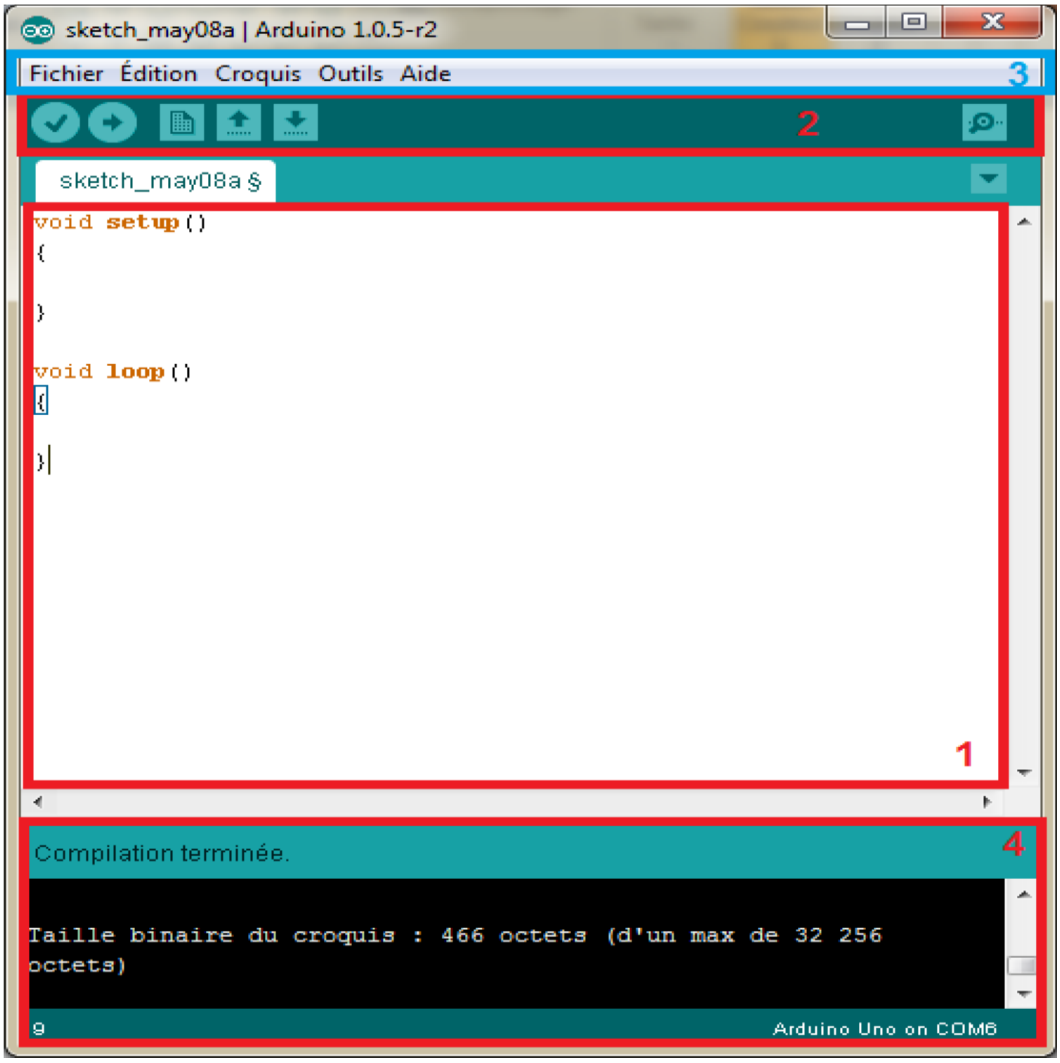


Figure 2.06 *Image de l'IDE Arduino*

Il est doté d'un éditeur de code avec coloration syntaxique et d'une barre d'outils rapide. Ce sont les deux éléments les plus importants de l'interface, c'est ceux que l'on utilise le plus souvent. On retrouve aussi une barre de menus plus classique qui est utilisé pour accéder aux fonctions avancées de l'IDE. Enfin, une console affichant les résultats de la compilation du code source, des opérations sur la carte et les erreurs.

Ce logiciel propose un menu composé de 5 boutons. Ces cinq boutons sont quasiment les seuls que nous aurons à utiliser pour lancer le programme. Nous allons donc les étudier un par un.

a. Le premier bouton à gauche : le V de vérifier.

Ce bouton permet de vérifier votre programme. C'est-à-dire que le logiciel Arduino va chercher si ce que vous avez écrit est conforme à ce qui est attendu.

Si lors de la compilation, les lignes de codes ne sont pas conformes à ce qui est attendu, un message d'erreur s'affichera dans une fenêtre en bas.

Le logiciel de l'arduino propose un petit débuggeur. Il ne va pas réparer tout seul l'erreur, mais il indique où il pense qu'elle se trouve. Pour comprendre un peu ce qui ce passe, il faut lire le message d'erreur :

- "sketch_feb20a.ino" c'est le nom de votre programme.
- "4:1" signifie que ligne 4 caractère 1, le programme bloque pour la compilation.
- "expected" un truc, indique le type d'erreur. Mais ce n'est pas toujours la bonne raison.

b. La flèche de téléverser

En fonction du système utilisé, l'Arduino va ou non être reconnu. Sur Windows, il faut parfois attendre qu'il installe un pilote de périphérique. Si on change de port USB, il va demander d'installer à nouveau le pilote. On a alors deux choix, soit on décide de ne connecter l'arduino qu'au même port USB, soit on les faites tous un par un jusqu'à ce que Windows ait tout installé.

Une fois que l'arduino est connecté à l'ordinateur par l'intermédiaire du câble USB. Dans les menus du logiciel, il y a un menu "Outils", dans ce menu, il y a un sous-menu "Port Série" qui propose plusieurs ports à utiliser. On choisit le port où la carte arduino est connectée. On clique sur l'icône avec la flèche pour téléverser le programme. Le logiciel va donc transférer le programme compilé (transformé en langage machine) dans la mémoire du microcontrôleur de l'Arduino. Une fois cette opération effectuée, l'Arduino gardera ce programme en mémoire et l'exécutera tant qu'il sera alimenté en électricité. Il sera donc autonome et ne dépendra plus de l'ordinateur.

c. Sauvegarde et transfert du programme

Lorsque le programme est televersé, il reste stocké dans le microcontrôleur. On peut débrancher ce dernier, le brancher à une pile 9V via la prise prévue à cet effet, ou à un autre ordinateur (sans lui envoyer de programme) et le programme téléversé va tourner encore et encore. Il est donc sauvegardé, mais il n'est plus accessible. C'est-à-dire que l'opération inverse, qui consisterait à sortir le programme de l'Arduino pour le mettre sur le logiciel de votre ordinateur, est impossible. C'est la raison pour laquelle il faut sauvegarder le travail. C'est le bouton le plus à droite (une flèche vers le bas) proposé par le logiciel :

Les fichiers sont stockés dans un dossier accessible depuis le logiciel par le menu en haut : on choisit "Fichier" puis "Carnet de croquis" et on voit apparaître les sauvegardes.

Pour ouvrir les programmes sauvegardés, on utilise le bouton à droite

d. Editeur de textes de l'IDE de l'arduino

Dans la console de programmation, on doit avoir le « void setup() » et le « void loop() »

```
1 void setup()
2 {
3
4 }
5
6 void loop()
7 {
8
9 }
```

Figure 2.07 *Programme principal de l'arduino*

Les déclarations se trouvant dans « void setup() » ne vont être exécutées qu'une seule fois, tandis que celles se trouvant dans « void loop() » vont être exécutées infiniment en boucle.

On remarque que les mots se colorent lorsqu'on les a saisis. C'est un petit plus très pratique qui s'appelle la coloration syntaxique. Elle très employée en informatique. Elle nous permet de faire la différence entre les mots clés du langage utilisé, et les autres. [7]

2.3.4.2 Langage Arduino

Le langage Arduino est inspiré de plusieurs langages. On retrouve notamment des similarités avec le C, le C++, le Java et le Processing. Le langage impose une structure particulière typique de l'informatique embarquée. La fonction « setup » contiendra toutes les opérations nécessaires à la configuration de la carte (directions des entrées sorties, débits de communications série, etc.). La fonction « loop » est exécutée en boucle après l'exécution de la fonction « setup ». Le programme téléversé continuera de boucler tant que la carte n'est pas mise hors tension, redémarré (par le bouton reset).

Au niveau de la syntaxe, on retrouve des similarités avec les langages précédemment cités. La déclaration des variables se fait généralement dans l'espace global, de façon à partager les variables les plus importantes entre les deux fonctions principales. On retrouve les types de base suivant :

Nom	Contenu	Taille (en octet)	Plage de valeurs
(unsigned) char	Entier ou caractère	1	(0->255) -128 -> 127
(unsigned) int	Entier	2	(0->65 535) -32 768 -> 32 767
(unsigned) long	Entier	4	(0 -> 4 294 967 295) -2 147 483 648 -> 2 147 483 647
float/double	Nombre à virgule flottante	4	-3,4028235E+38 -> 3,4028235E+38
String	Chaine de caractères (Objet)	variable	Aucune
boolean	Booléen	1	True / False

Tableau 2.05 *Tableau de présentation de variable*

On retrouve les opérateurs les plus courants pour les types de bases. Parmi eux : = (affectation), == (comparaison), != (différence), <, >, <=, >=, && (et logique), || (ou logique), ! (non logique).

On retrouve aussi les opérateurs mathématiques (+, -, *, /, %) et les opérateurs logiques bit à bit (^ (XOR), & (et), | (ou), ~(non), << (décalage logique à gauche), >> (décalage logique à droite)).

Les structures de contrôle sont elles aussi similaires aux langages de références cités précédamment. On y retrouve toutes les structures de contrôle standard, conditions, boucle, switch, fonctions, etc. Chaque structure de contrôle est suivie d'un bloc d'instructions délimitées par des accolades

Nom	Utilité	Syntaxe

If-else	Condition logique	If (<valeur booléenne>) {
		<instruction>
		} else {
		<instruction>
		}

If-else if – else	Condition logique multiples	If (<valeur booléenne>) {
		<instruction>
		} else if (<valeur booléenne>) {
		<instruction>
		} else {
		<instruction>
		}

Switch	Sélecteur	Switch (<variable>) { case <valeur> : <instruction> break ; default : <instruction> }

While	Boucle	While (<valeur booléenne>)
		{
		<instruction>
		}

For	Boucle itérative	For (<initialisation> ;
		<valeur booléenne> ;
		<évolution>) {
		<instruction>
		}

Tableau 2.06 *Liste des structures de contrôles*

On peut aussi écrire des structures et des classes. Les fonctions, les structures et les classes se déclarent de la même façon qu'en C++. C'est-à-dire, se déclarent sous les forment suivantes :

- Fonction : <type de retour><nom>(<paramètre>) {
  <instruction>
  }

- Structure : struct<nom>{
  <type><nom du champ> ;
  } ;
- Classe : class<nom>{
  public :
  <attributs et champs publics>
  private :
  <attributs et champs privés>
  Protected :
  <attributs et champs privés>
  } ;

a. Fonctionnalité de base : entrées/sorties

Le langage Arduino vient avec un nombre important de fonction de base permettant d'interagir avec son environnement. Les fonctions les plus utilisée sont les fonctions d'entrée/sorties. Ce sont elles qui permettent d'envoyer ou de mesurer une tension sur une des broches de la carte.

Dans un premier temps, avant d'effectuer une mesure ou d'envoyer une commande. Il est nécessaire de définir la direction des broches utilisées. Pour cela on fait appel à la fonction pinMode en lui donnant d'une part, la broche concernée, et d'autre part, la direction :

```
void setup() {
        pinMode(1,OUTPUT) ; // Broche 1 en sortie
        pinMode(2,INPUT) ;  // Broche 2 en entrée
        }
```

Une fois cette configuration faite, on peut procéder à l'utilisation des broches. Toutes les broches sont capables d'écrire et de lire des données numériques (c'est-à-dire des 0 (0V) ou des 1 (5V)). Mais, certaines disposent de fonctionnalité supplémentaire.

Toutes les cartes Arduino possèdent des entrées analogiques. Elles permettent de lire des tensions analogiques (comprise entre 0 et 5V) et de le convertir en entier (compris entre 0 et 1023) proportionnellement à la tension mesurée.

Certaines cartes Arduino possède des sorties analogique faisant l'opération inverse (met une tension sur la broche proportionnellement à l'entier donné).
Pour pouvoir tout de même contrôler des composants autrement qu'en « tout ou rien » il est possible d'utiliser des broches PWM. Les PWM sont utilisées pour synthétiser des signaux analogiques en modulant le temps passé à l'état 1 (5V).

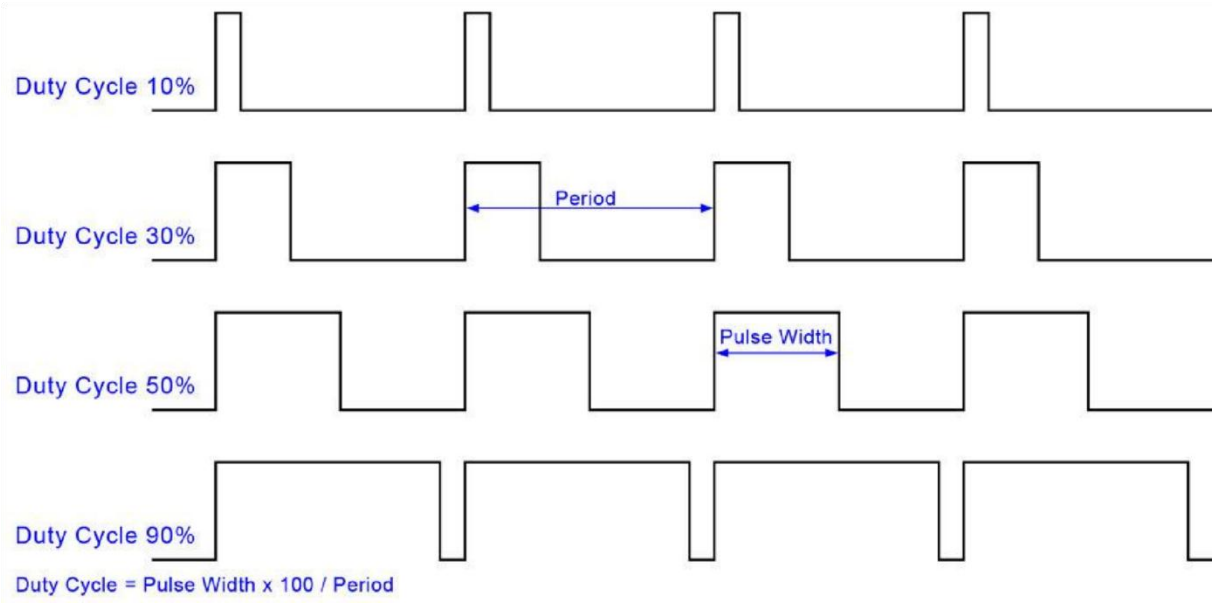


Figure 2.08 *Signal PWM*

En utilisant une fréquence relativement élevée, les PWM permettent de commander certains composants comme s'il recevait une tension analogique. Cela provient du fait que les composants utilisés dans l'électronique analogique, ne changes pas d'états instantanément. Grâce à cela, on pourra par exemple faire varier l'intensité d'une LED. La plupart des cartes Arduino utilisent des PWM cadencées à 490Hz environ.

Toutes ces fonctionnalités sur les broches d'entrées sorties sont utilisables par le biais de quatre fonctions :

- digitalRead(pin) : mesure une donnée numérique sur une des broches, la broche en question doit être réglée en entrée.
- digitalWrite(pin, value) : écrit une donnée numérique sur une des broches, la broche concernée doit être réglée en sortie. Le paramètre value doit être égal à HIGH (état 1 soit 5V) ou LOW (état 0 soit 0V). [9]

- analogRead(pin) : mesure une donnée analogique sur une des broches (compatible seulement), la broche doit être réglée sur entrée.
- analogWrite(pin, value) : écrit une donnée sous forme de PWM sur une des broches (compatible uniquement), la broche doit être réglée en sortie. Le paramètre value doit être compris dans l'intervalle [0;255] .

b. La gestion du temps

Pour la plupart des applications, il est nécessaire de faire intervenir des intervalles de temps. Par exemple, pour gérer le temps d'appui sur un bouton ou pour faire une sonnerie qui se répète un certain nombre de fois. Le langage Arduino fournis quelques fonctions permettant de gérer le temps.

Il est possible d'insérer une pause dans le programme pendant un instant. Pour cela, on utilise les fonctions « delay »et « delayMicroseconds » qui insère une pause suivant le paramètre passé (en milliseconde pour l'un, en microseconde pour l'autre). Cependant ces fonctions bloquent le microcontrôleur, on ne peut alors plus effectuer aucune action.

En plus d'insérer une pause, il est possible de mesurer le temps. De la même manière que les fonctions de délai, on utilise les fonctions « millis » et « micros » qui donnent le nombre de milliseconde, respectivement microseconde, depuis le lancement de la carte. Cependant, ces fonctions incrémentent une variable (interne). Ces variables se remettent à zéro une fois le maximum (overflow) atteint. La variable utilisée pour les millisecondes atteint son maximum au bout de 49 jours et 17 heures et la variable utilisée pour les microsecondes au bout de 71 minutes et 34 secondes environ. [8]

2.4 Conclusion

Si on met tout cela dans une coquille, la BD, qui est un ensemble d'information ou de données issues et peut être partagé de manière relative par plusieurs utilisateurs, est l'un des outils nécessaires pour le fonctionnement du projet. C'est le SGBD qui permet d'avoir tous les avantages lors de l'utilisation et la manipulation de la BD. Le deuxièmement considéré comme outil indispensable est l'arduino. Ce dernier est un circuit qui comporte tous les éléments et caractéristiques nécessaires pour fonctionner avec son microcontrôleur. Chaque version de l'arduino a ses caractéristiques et est choisie selon la finalité du projet estimé. C'est le langage arduino qui est utilisé pour communiquer avec cette carte.

CHAPITRE 3 : CONCEPTION ET REALISATION

3.1 Utilisation du module RC-522 avec l'arduino

3.1.1 Montage

L'émetteur RFID propose 8 broches qu'on peut brancher avec les Pin de l'arduino.

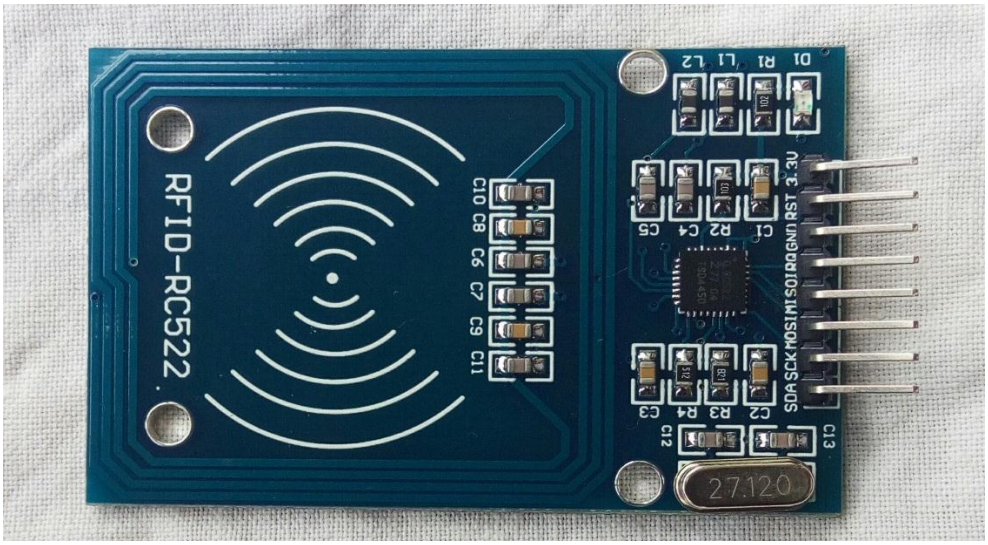


Figure 3.01 *Lecteur RC-522*

Le module RFID est monté sur une carte déjà câblée avec une antenne prête à l'emploi. Il permet d'identifier sans contact des puces RFID présentées à proximité placées dans une carte ou un badge. Ceci permet une lecture de badge sans contact.

Utilisations variées:

- Badges d'accès
- Identifier une personne ou un objet sans contact
- Cartes de visite numérique
- Payement direct
- Identifier une carte de transport
- Péage urbain rapide
- Identification de livre de bibliothèque, etc...

L'alimentation est de 3.3V mais pas 5V ; le lecteur utilise une interface SPI, et fonctionne sur une fréquence de 13.56 MHz, donc il lit tous les badges répondant à cette fréquence.

Le bus SPI se pilote par 4 signaux logiques et le brochage dépend des cartes (Uno, Mega, Nano…).

Carte Arduino	MOSI	MISO	SCK	SS Slave	SS Master
Uno Ou Duemilanove	11 ou ICSP-4	12 ou ICSP-1	13 ou ICSP-3	10	-
Mega 1280 Ou Mega 2560	51 ou ICSP-4	50 ou ICSP-1	52 ou ICSP-3	53	-
Leonardo	ICSP-4	ICSP-1	ICSP-3	-	-
Due	ICSP-4	ICSP-1	ICSP-3		4, 10, 52

Tableau 3.01 *Tableau de branchement sur communication SPI*

Comme la version de carte Arduino la plus utilisée est la carte Uno, Cela a pour conséquence qu'on présente généralement le schéma de câblage sur la carte Uno.

Brochage sur Carte Arduino **UNO** :

- SCLK Horloge (généré par le maître) : pin 13 ou ICSP-3
- MOSI Master Output, Slave Input (généré par le maître) : pin 11 ou ICSP-4
- MISO Master Input, Slave Output (généré par l'esclave) : pin 12 ou ICSP-1
- SS Slave Select, actif à l'état bas, (généré par le maître) : pin 10

Branchement à réaliser (avec la carte Uno) :

- Vcc : +3.3V alimentation
- Rst : pin 9 reset
- Gnd : -gnd masse
- Miso : pin 12
- Mosi : pin 11
- Sck : pin 13
- Nss : pin 10
- Irq : non connecté

[10]

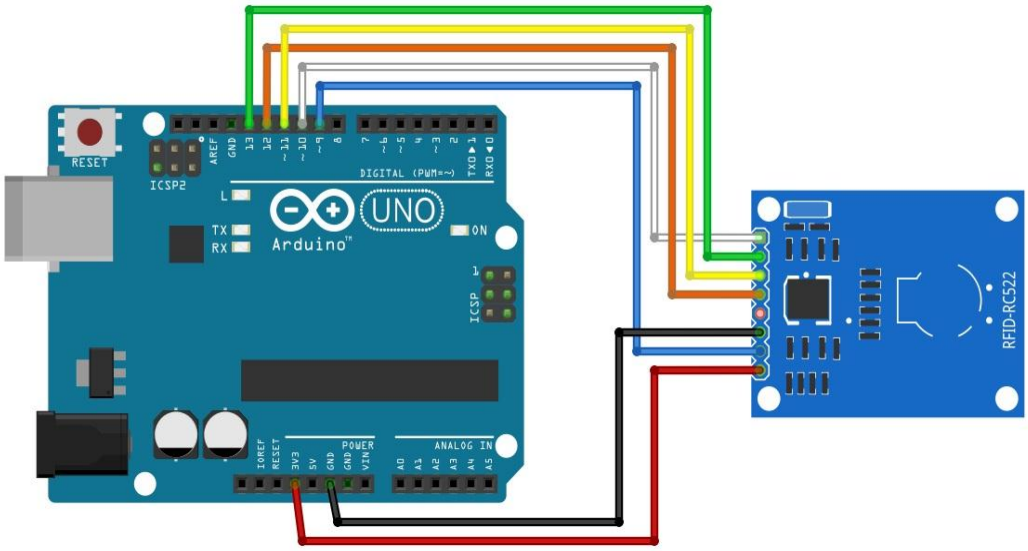


Figure 3.02 *Branchement du lecteur RC-522 sur la carte Arduino Uno*

Certes, la carte généralement utilisée est la version Uno ; cependant, dans ce projet, on utilise la version Mega. Donc, il suffit juste de suivre le tableau ci-dessus et réaliser ensuite le même branchement avec la carte Arduino Méga 2560 pour notre projet. L'image présentée sur la figure 3.03 suivante montre le lecteur RFID-RC522 monté sur la carte électronique arduino Méga.

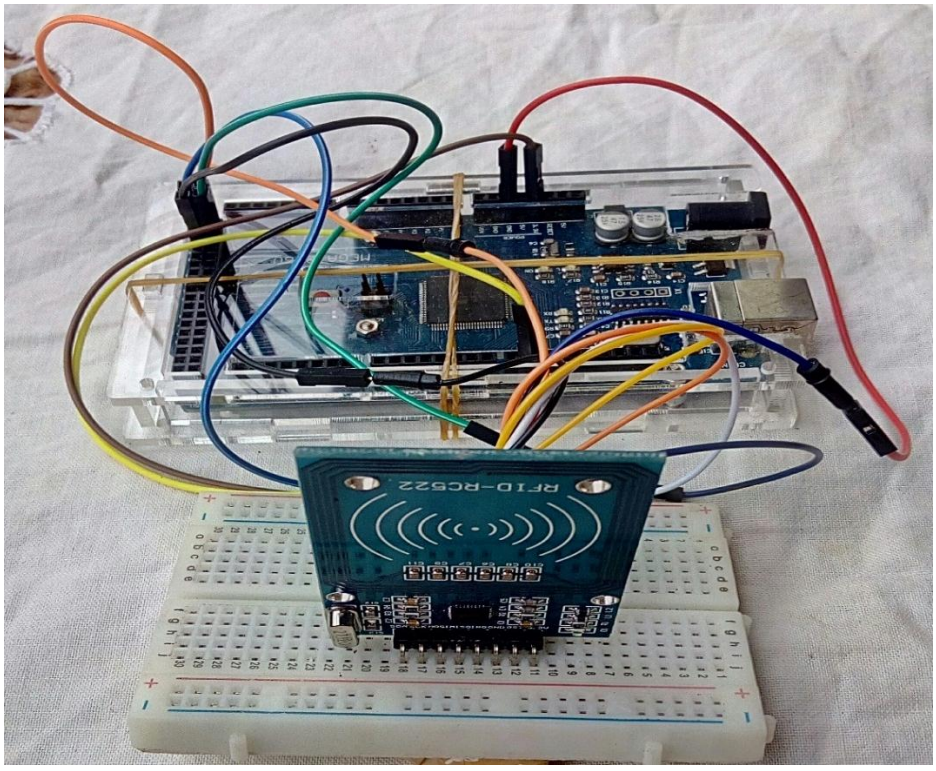


Figure 3.03 *Image du lecteur RFID RC-522 monté sur la carte arduino Méga*

3.1.2 Lignes de codes

Avant toute chose, on doit d'abord télécharger les 2 bibliothèques indispensables pour pour établir la bonne connexion entre l'Arduino et le module RC522. Une fois téléchargés, ces 2 bibliothèques doivent être ajoutées dans la liste des bibliothèques utilisables de l'IDE Arduino.

Figure 3.04 *Les bibliothèques nécessaires pour la réalisation*

Il faut inclure ces 2 bibliothèques dans le programme de lecture pour pouvoir indiquer à l'Arduino que nous les utilisons dans notre programme.

```
#include <SPI.h>

#include <RFID.h>
```

Figure 3.05 *Inclure les 2 bibliothèques*

Ensuite, on déclare l'utilisation de notre module RFID, avec la gamme de pins selon le branchement de la carte utilisée avec la communication SPI.

```
RFID monModuleRFID(53,9);
```

Figure 3.06 *Déclaration de l'utilisation du module RFID*

Dans la fonction « void setup », on doit initialiser le module RFID et aussi la communication SPI.

```
SPI.begin(); /*initialisation de la communnication SPI*/
monModuleRFID.init(); /*initialisation du module RFID*/
```

Figure 3.07 *Initialisations de la communication SPI et le module RFID*

Enfin, dans la fonction « void loop », qui est considérée comme la fonction principale, on ordonne au système s'il existe une carte ou un tag devant l'émetteur et qui doit être lu. Si l'émetteur détecte un signal venant d'une carte, on ordonne à notre carte Arduino de lire celle-ci. Evidemment, tout cela s'effectue à travers l'émetteur RC-522.

Et enfin, on dit à l'Arduino d'afficher les informations recueillis dans le moniteur série. Dans notre cas, ces informations doivent être des numéros uniques à chaque carte ou tag.

```
if (monModuleRFID.isCard()) {
      if (monModuleRFID.readCardSerial()) {
        Serial.print("L'UID est: ");
        for(int i=0;i<=4;i++)
        {
          UID[i]=monModuleRFID.serNum[i];
          Serial.print(UID[i],DEC);
          Serial.print(".");
        }
        Serial.println("");
```

Figure 3.08 *Détection et lecture de carte*

3.2 Développement d'un logiciel

Pour pouvoir effectuer la gestion du parking, avec le module RFID et la carte arduino, on a besoin de fabriquer un logiciel avec une interface qui va permettre de concrétiser notre gestion à partir du système.

3.2.1 Langage utilisé

Pour la réalisation de notre projet, nous utiliseront comme langage de programmation le langage C#. Il est alors judicieux de connaitre quelques points importants sur ce langage de programmation.

3.2.1.1 Présentation générale

Le C# est le langage de programmation phare de Microsoft, et créé par celui-ci en 2002. Utilisé par un nombre important et grandissant de professionnels, il permet de réaliser toutes sortes d'applications.

Un langage de programmation est un ensemble d'instructions, c'est-à-dire un ensemble de mots qui permettent de construire des applications informatiques.

Ces applications informatiques peuvent être de beaucoup de sortes, par exemple une application Windows, comme un logiciel de traitement de texte ou une calculatrice ou encore un jeu de cartes.

On les appelle également des clients lourds. Il est également possible de développer des applications web, comme un site d'e-commerce, un intranet, … Nous pourrons accéder à ces applications grâce à un navigateur internet que l'on appelle un client léger. Toujours grâce à un navigateur internet, nous pourrons développer des clients riches. Ce sont des applications qui se rapprochent d'une application Windows mais qui fonctionnent dans un navigateur. Bien d'autres types d'applications peuvent être écrites avec le C#, citons encore le développement d'applications mobiles sous Windows phone, d'applications pour tablettes, de jeux ou encore le développement de web services …

Le C# est un langage dont la syntaxe ressemble un peu au C++ ou au Java qui sont d'autres langages de programmation très populaires. Il fait partie d'un ensemble plus important. Il est en fait une brique de ce qu'on appelle le « Framework .NET ». [11]

3.2.1.2 Le framework .NET

Le nom DOTTE NETTE ou POINT NETTE peut être trompeur. Avec l'omniprésence d'internet, son abréviation (net) ou encore des noms de domaines (.net), on pourrait penser que le framework .NET est un truc dédié à internet alors ceci n'est pas vraie. Il est donc préférable de préciser un peu ce qu'est le framework .NET pour éviter les ambigüités.

Pour simplifier, on peut dire qu'un framework est une espèce de grosse boite à fonctionnalités qui va nous permettre de réaliser des applications informatiques de toutes sortes.

En fait, c'est la combinaison de ce framework et du langage de programmation C# qui va nous permettre de réaliser ces applications informatiques.

Le framework .NET est un framework créé par Microsoft en 2002, en même temps que le C#, qui est principalement dédié à la réalisation d'applications fonctionnant dans des environnements Microsoft. On peut par exemple réaliser des programmes qui fonctionnent sous Windows, ou bien des sites web ou encore des applications qui fonctionnent sur téléphone mobile.

On peut voir le framework .NET comme un ensemble de composants que l'on devra assembler pour réaliser notre application. Certains sont spécifiques pour la réalisation d'applications web, d'autres pour la réalisation d'applications Windows, etc. Pour assembler notre application, nous allons utiliser un langage de programmation : le C#.

3.2.1.3 Visual C#

Visual C#, utilisable dans l'IDE Visual Studio, est un outil de développement qui permet de concevoir des applications développé avec le langage C# qui est un langage conçu pour générer des applications d'entreprise. Il propose les outils pour développer des applications C# qui ciblent la plateforme nouvelle génération de Microsoft pour la programmation distribué et compatible Internet.

C'est dans Visual Studio 2002 que Visual C# est premièrement apparu. Aujourd'hui, dans sa version actuelle (version de 2008), on peut distinguer deux versions utilisables :

- La version professionnelle, et payante, qui est inclus dans l'IDE Visual Studio 2008
- La version gratuite qui est proposé comme application indépendante téléchargeable sur internet. C'est la version Visual Studio C# Express

3.2.1.4 Les applications informatiques

L'ordinateur exécute des applications informatiques pour effectuer des tâches. Ce sont des logiciels (comme un traitement de texte, un navigateur internet, un jeu vidéo…). Cependant, l'ordinateur ne peut exécuter ces applications informatiques que si elles sont écrites dans le seul langage qu'il comprend, le binaire. Techniquement, le binaire est représenté par une suite de 0 et de 1. Il n'est bien sûr pas raisonnablement possible de réaliser une grosse application en binaire, c'est pour ça qu'il existe des langages de programmation qui permettent de simplifier l'écriture d'une application informatique.

3.2.1.5 Langages traditionnels : la compilation

Avec des langages traditionnels comme le C et le C++, on écrit des instructions simplifiées, lisibles par un humain (comme : printf("Bonjour");). C'est beaucoup plus simple que le binaire et on comprend globalement avec cet exemple que l'on va afficher le mot Bonjour. Bien entendu, l'ordinateur ne comprend pas ces instructions car ce n'est pas encore du binaire. Pour obtenir du binaire à partir d'un code écrit en C ou C++, on doit effectuer ce qu'on appelle une compilation. Le compilateur est un programme qui traduit le code source en binaire exécutable.

Cette méthode est efficace et a fait ses preuves. De nombreuses personnes développent toujours en C et C++ aujourd'hui. Néanmoins, ces langages ont aussi un certain nombre de défauts dus à leur ancienneté. Par exemple, un programme compilé (binaire) ne fonctionne que sur la plateforme pour laquelle il a été compilé.

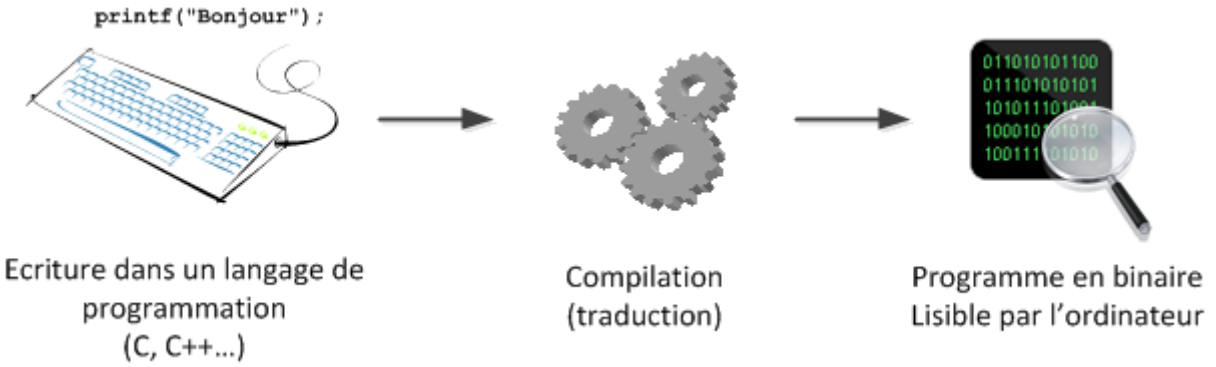


Figure 3.09 *Compilation en langages traditionnels*

Cela veut dire que si on compile sous Windows, on obtient un programme qui fonctionne sous Windows uniquement (et sur un type de processeur particulier). Impossible de le faire tourner sous Mac OS X ou Linux simplement, à moins de le recompiler sous ces systèmes d'exploitation (et d'effectuer au passage quelques modifications). [11]

3.2.1.6 Langages récents : le code managé

Les langages récents, comme le C# et le Java, résolvent ce problème de compatibilité tout en ajoutant de nombreuses fonctionnalités appréciables au langage, ce qui permet de réaliser des programmes beaucoup plus efficacement.

La compilation en C# ne donne pas un programme binaire, contrairement au C et au C++. Le code C# est en fait transformé dans un langage intermédiaire (appelé CIL ou MSIL) que l'on peut ensuite distribuer à tout le monde. Ce code, bien sûr, n'est pas exécutable lui-même, car l'ordinateur ne comprend que le binaire.

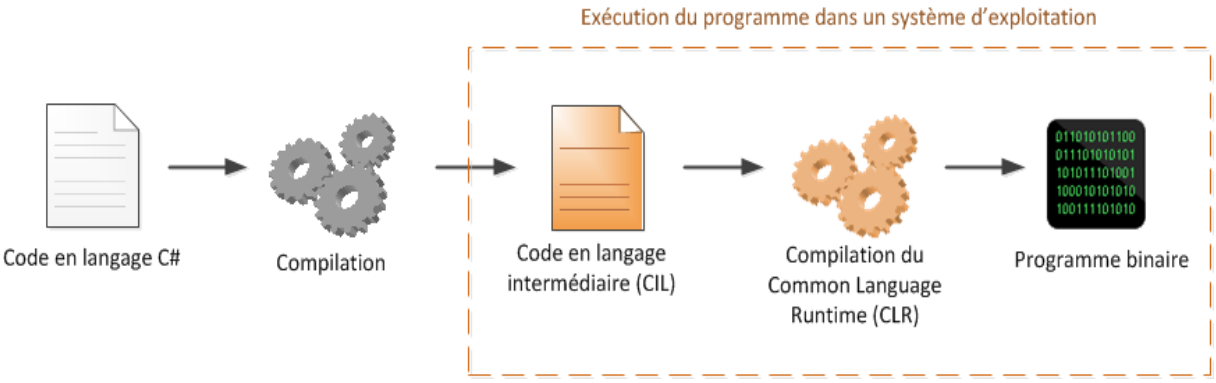


Figure 3.10 *Compilation en langage récents*

Le CIL correspond au programme que l'on distribue. Sous Windows, il prend l'apparence d'un .exe comme les programmes habituels, mais il ne contient en revanche pas de binaire.

Lorsqu'on exécute le programme CIL, celui-ci est lu par un autre programme (une machine à analyser les programmes, appelée CLR) qui le compile cette fois en vrai programme binaire. Cette fois, le programme peut enfin s'exécuter.

Certainement, cela complique bien les choses quand même. Mais cela offre beaucoup de souplesse au développeur. Le CIL peut être distribué à tout le monde. Il suffit d'avoir installé la machine CLR sur son ordinateur, qui peut alors lire les programmes en C# et les compiler "à la volée" en binaire. Ceci présente l'avantage que le programme est toujours adapté à l'ordinateur sur lequel il tourne.

La CLR vérifie aussi la sécurité du code ; ainsi en C, du code mal pensé (par exemple une mauvaise utilisation des pointeurs) peut entraîner des problèmes pour votre PC, ce qu'on risque beaucoup moins avec le C#. De plus, la CLR dispose du JIT debugger qui permet de lancer Visual Studio si une erreur survient dans un programme .NET pour voir ce qui a causé cette erreur. On parle de code managé. Cette complexité ralentit légèrement la vitesse d'exécution des programmes (par rapport au C ou au C++), mais la différence est aujourd'hui vraiment négligeable par rapport aux gains que cela apporte.

Donc, en théorie, il est possible d'utiliser n'importe quelle application compilée en langage intermédiaire à partir du moment où il y a une implémentation du CLR disponible. En réalité, il n'y a que sous Windows qu'il existe une implémentation complète du CLR. Il existe cependant une implémentation partielle sous Linux : Mono. Cela veut dire que si un programme utilise des fonctionnalités qui ne sont pas couvertes par Mono, il ne fonctionnera pas. En conclusion, dans la pratique, le .NET est totalement exploitable sous Windows, ailleurs non.

Il est possible de créer des programmes (.exe) qui pourront directement être exécuté par le CLR, mais il est également possible de créer des bibliothèques sous la forme d'un fichier possédant l'extension « .dll ». On appelle ces deux formes de programmes des assemblages, mais on utilise globalement toujours le mot anglais « assembly ».

- Les fichiers .exe sont des assemblys de processus
- Les fichiers .dll sont des assemblys de bibliothèques

Concrètement, cela signifie que le fichier .exe servira à lancer une application et qu'une dll pourra être partagée entre plusieurs applications .exe afin de réutiliser du code déjà écrit.
Il est à noter qu'un raccourci est souvent fait avec le terme assembly. On a tendance à voir que le mot assembly sert à désigner uniquement les bibliothèques dont l'extension est .dll.

3.2.1.7 Premier code d'application

Souvent, en débutant dans l'apprentissage d'un langage de programmation, un développeur a toujours tendance à réaliser, comme premier programme, l'affichage du fameux « Hello Word ». Voici comment c'est fait avec C# dans Visual Studio :

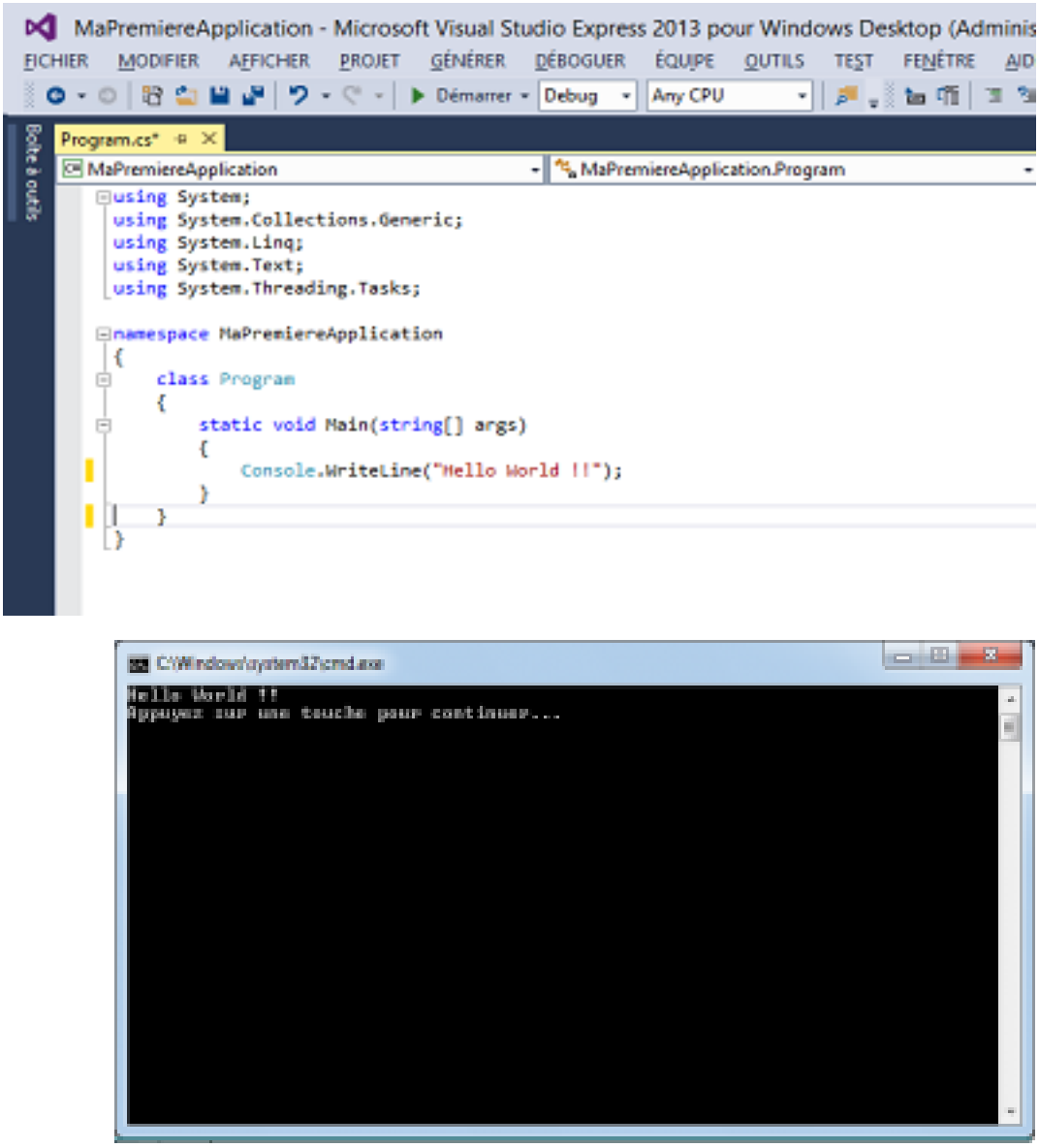


Figure 3.11 *Programme d'affichage du « Hello World »*

3.2.2 Interfaces

3.2.2.1 Base de données

Dans ce projet, pour mettre en place notre BD, on utilise l'IDE MySql Workbench. Tout d'abord, on va créer un schéma qui n'est autre que la BD elle-même. Pour cela, on a qu'à ouvrir le logiciel MySql Workbench ; puis, on va juste sur le menu « SHEMA » et sélectionner l'option « Create Schema » comme l'indique la figure suivante :

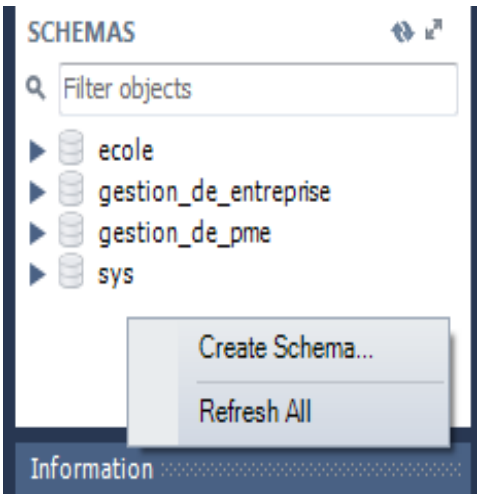


Figure 3.12 *Création de Base de données*

En faisant cela, on a accès à une fenêtre qui va nous permettre de définir le nom que va porter notre BD.

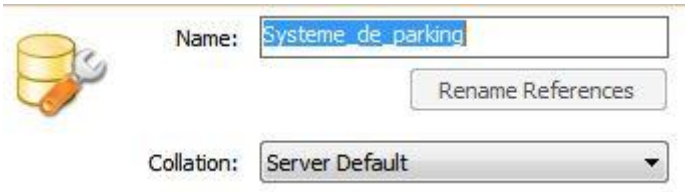


Figure 3.13 *Définition du nom du BD*

Après ceci, en cliquant sur « Apply », on a réussi à créer une base de données. Ensuite, on doit maintenant ajouter une Table à cette base de données. Pour le faire, on va sur le sous-menu « Table » et choisir l'option « Create Table » comme le montre la figure 3.14.

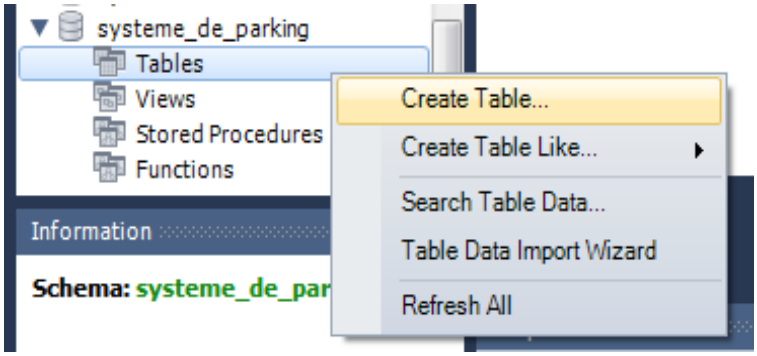


Figure 3.14 *Ajout de table*

Puis, exactement comme dans création de schéma, on définit le nom de la table dans la fenêtre qui s'ouvre après le choix de l'option « Create Table ». Ensuite on entre les attributs, ou colonnes, qu'on a besoin dans le projet, sans oublier d'indiquer si l'attribut est une clé primaire (PK) et s'il doit être non-nul (NN) ou aussi autre.

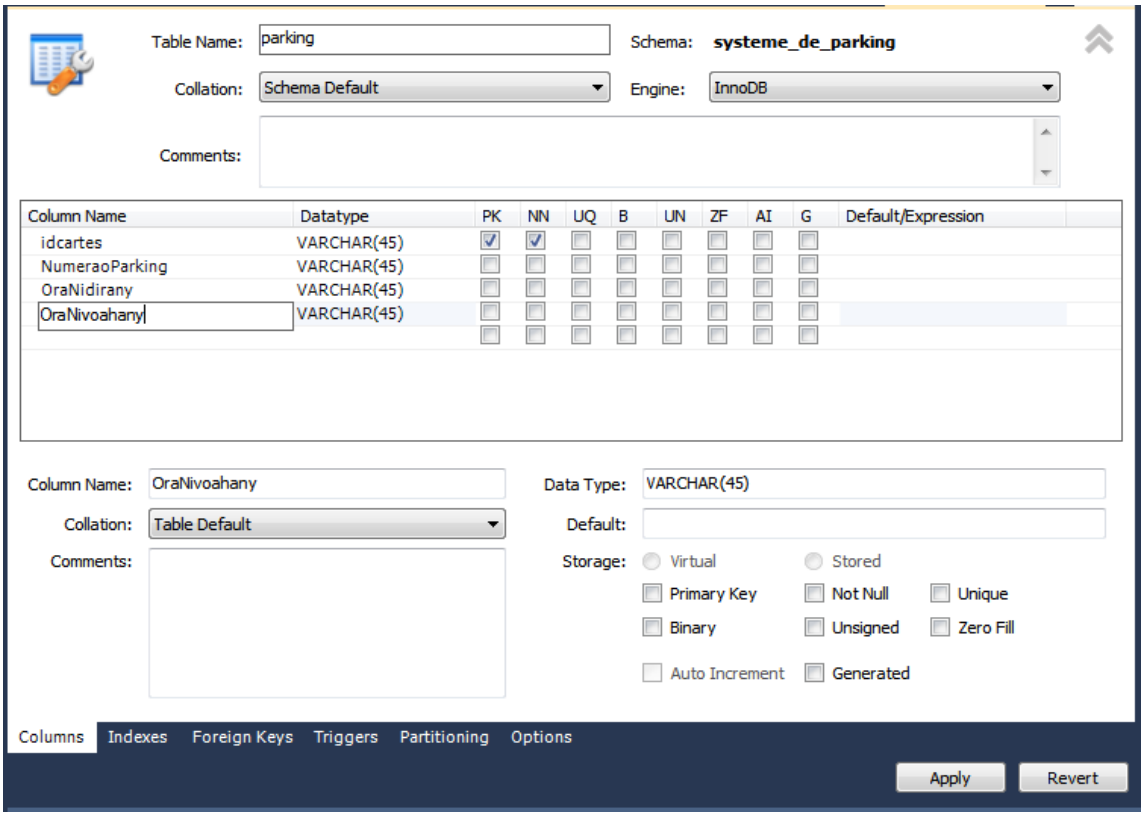


Figure 3.15 *Définition de nom de table et création des attributs*

En cliquant de nouveau sur « Apply », on réussit à créer une table avec les attributs souhaités. Pour remplir cette table, ou bien les attributs, on a qu'à saisir manuellement les valeurs désirées comme l'indique cette figure ci-dessous. Ici, on va y entrer les numéros des cartes qu'on lit avec le module RFID.

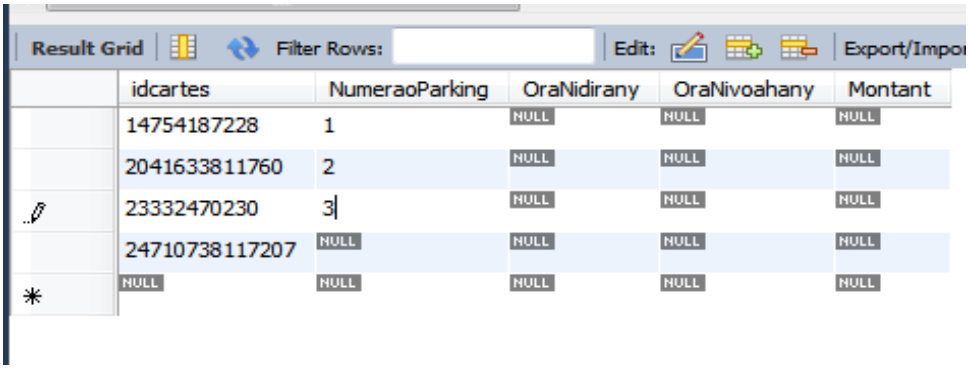


Figure 3.16 *Remplissage des valeurs des colonnes*

3.2.2.2 Récupération des informations du port série

Apres que cette base de données a été bien mise en place, on doit s'assurer que notre système composé de Arduino, de module RFID, de base de données et de logiciel soit bien interconnecté pour que l'on puisse bien mettre en œuvre le projet qui est la gestion de personnels par radiofréquence. Rappelons que le module RFID envoie des numéros d'identification au port série, selon la carte présentée devant le lecteur. On va alors récupérer ces informations, présentes dans le port série, et faire en sorte que l'on puisse les avoir dans l'interface de notre logiciel. Pour cela, on ajoute d'abord les outils nécessaires pour pouvoir définir le port comme où les informations sont envoyées. Ce port n'est autre que le port où est connectée la carte Arduino.

Figure 3.17 *Outils de définition du port série*

C'est alors dans cette zone de texte qu'on définit le port où est établie la communication. Quand ce port est saisi dans cette zone de texte, on appuie juste sur le bouton « Start » pour commencer à recevoir les informations présentes dans ce port. Bien sûr, contrairement à cela, le bouton « Stop » sert à stopper la réception des données présentes dans le port.

Une fois que cette définition de numéro de port est faite, on affiche, dans une autre zone de texte, l'information présente dans le port série qui est, dans notre cas, les numéros de carte que le lecteur RFID a lu.

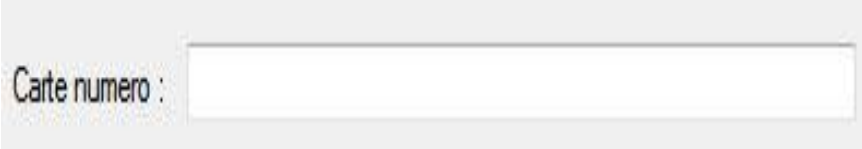


Figure 3.18 *Récupération de l'information dans le port série*

Tout ceci c'est du côté interface, mais on a besoin d'écrire des lignes de code pour que ceci fonctionne. D'abord, on indique la fonctionnalité qui doit être employée pour pouvoir récupérer les données du port série.

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;
```

Figure 3.19 *Indication de fonctionnalité*

Ensuite, il faut aussi déclarer les variables qu'on utilise dans les scripts qu'on va saisir.

```
private SerialPort myport;
private string in_data;
```

Figure 3.20 *Déclaration de variables*

Après, on entre les caractéristiques de lecture du port ; on écrit aussi les lignes de codes permettant de démarrer la réception des informations du port.

```
private void start_btn_Click(object sender, EventArgs e)
{
    myport = new SerialPort();
    myport.BaudRate = 9600;
    myport.PortName = port_name_tb.Text;
    myport.Parity = Parity.None;
    myport.DataBits = 8;
    myport.StopBits = StopBits.One;
    myport.DataReceived += myport_DataReceived;
    try
        {
            myport.Open();
            data_tb.Text = "";
        }
    catch (Exception ex10) {
        MessageBox.Show(ex10.Message, "Error");
    }
}
```

```
void myport_DataReceived(object sender, SerialDataReceivedEventArgs e)
{
    in_data = myport.ReadLine();

    this.Invoke(new EventHandler(displaydata_event));


}

private void displaydata_event(object sender, EventArgs e)
{
    data_tb.Text = in_data;
}
```

Figure 3.21 *Démarrage de la réception d'informations présente dans le port série*

Enfin, on entre les lignes de codes permettant de stopper la réception des informations.

```
private void button5_Click_1(object sender, EventArgs e)
{
    try
    {
        myport.Close();
    }
    catch (Exception ex11)
    {
        MessageBox.Show(ex11.Message, "Error");
    }
}
```

Figure 3.22 *Arrêt de la réception d'informations présente dans le port série*

3.2.2.3 Récupération de la BD dans l'interface

Certes, on a déjà une base de données à l'intérieur du logiciel MySql WorkBench ; cependant, il est préférable et plus pratique d'avoir la base de données dans l'interface de notre logiciel. Pour pouvoir faire cela, on a besoin d'un outil que l'on appelle « DataGriedView » pour y afficher la table de la base de données qu'on a créé ci-dessus. On a juste qu'à faire un « drag and drop » et importer un « DataGriedView » du « Tool Box » vers l'interface.

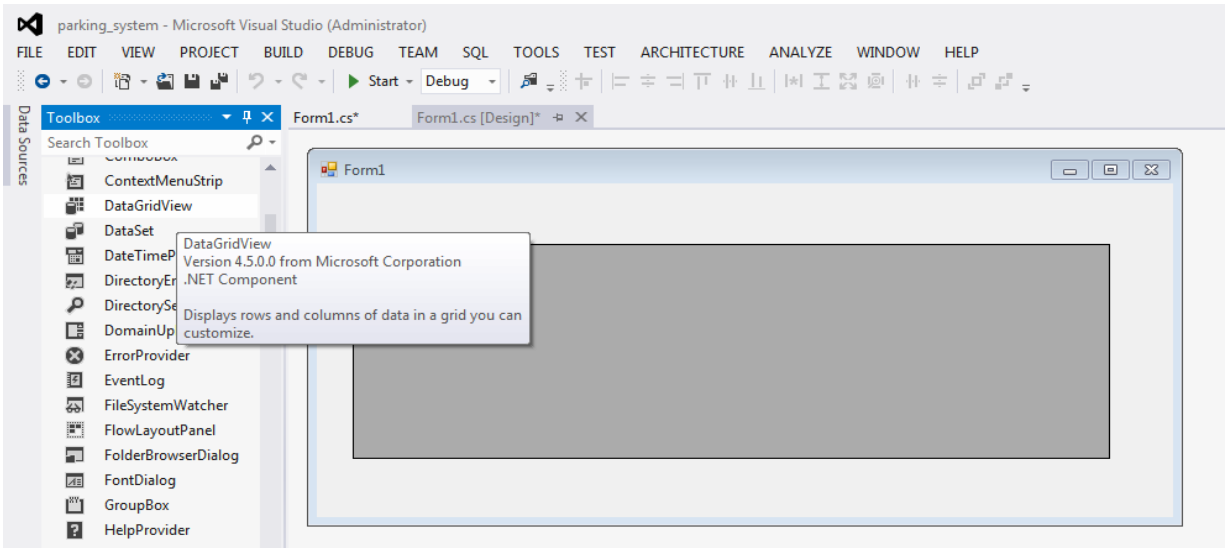


Figure 3.23 *Importation du DataGridView dans l'interface*

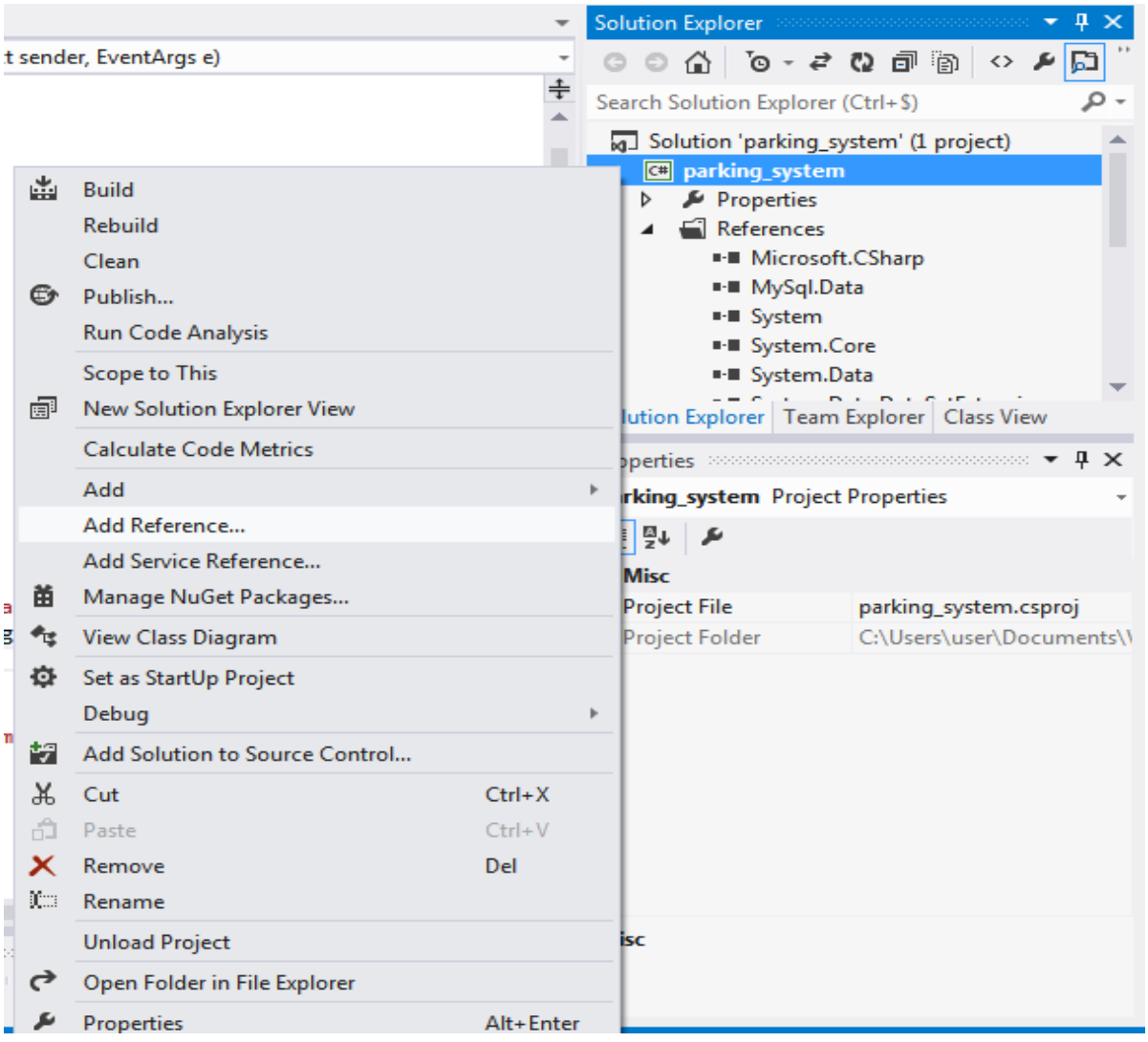


Figure 3.24 *Ajout de référence dans le projet*

Ensuite, on ajoute dans notre projet la référence « MySql.Data » pour pouvoir bien accéder à la base de données. Pour ajouter cette référence, on y procède comme l'est indiqué dans la figure 3.24.

Après, on indique, comme l'est présenté sur la figure suivante, qu'on utilise bien la référence « MySql.Data » qu'on a précédemment ajouté.

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using MySql.Data.MySqlClient;
```

Figure 3.25 *Indication de l'utilisation de référence*

Et puis, on indique, avec C#, l'adresse, l'utilisateur, le port et le mot de passe du BD pour que le logiciel puisse y avoir accès. Après, on établit l'objet de connexion. Et enfin, on y met la requête SQL qu'on veut exécuter. Ces étapes sont brièvement représentées dans la figure suivante :

```
/*Indication de l'adresse du BD, l'utilisateur du BD, le port et le mdp du BD*/
String constring = "datasource=localhost; port=3306; username=root; password=3210";

/*creation de l'objet de connexion*/
MySqlConnection conDataBase = new MySqlConnection(constring);

/*Commande requête SQL*/
MySqlCommand cmdDataBase1 = new MySqlCommand("select * from systeme_de_parking.parking;", conDataBase);
```

Figure 3.26 *Connexion et communication avec la BD*

Ensuite, on indique au logiciel d'afficher la table dans le « DataGridView » après avoir ordonner d'appliquer la requête SQL.

```
MySqlDataAdapter sda = new MySqlDataAdapter();
sda.SelectCommand = cmdDataBase1; //appliquer la commande SQL
DataTable dbdataset = new DataTable();//indiquer notre nouveau BD
sda.Fill(dbdataset);
BindingSource bsource = new BindingSource();
bsource.DataSource = dbdataset;
dataGridView1.DataSource = bsource;//affichage de la table dans le DataGridView
```

Figure 3.27 *Affichage de la Table dans l'interface du logiciel*

Après avoir fait tout ceci, on réussit à obtenir la table, qu'on a créé dans « MySql Worbench », dans l'interface du logiciel (qu'on crée). Ceci a pour bénéfice de pouvoir suivre, en temps réel, les modifications apportées sur la base de données. Ici, cela est très bénéfique car on peut alors ainsi voir les heures de passage des cartes.

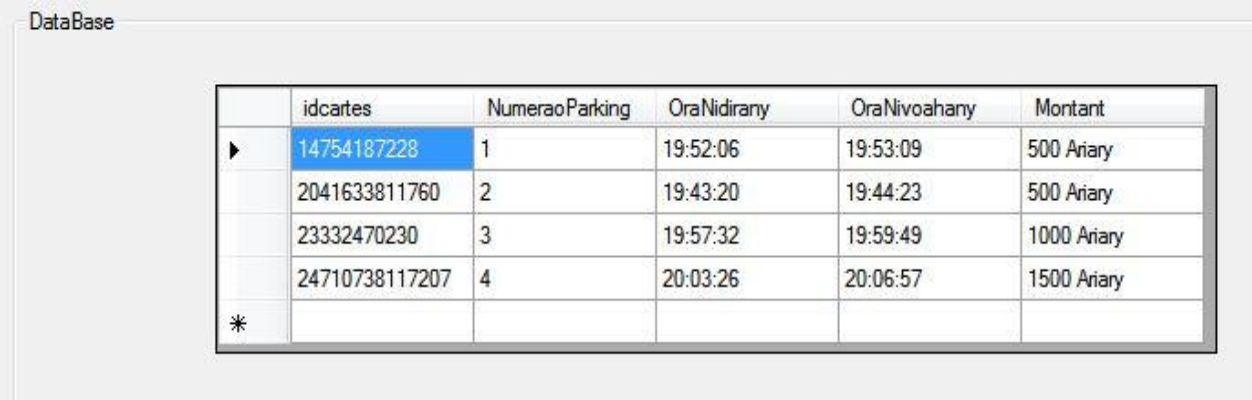


Figure 3.28 *Table de la Base de données obtenue dans l'interface du logiciel*

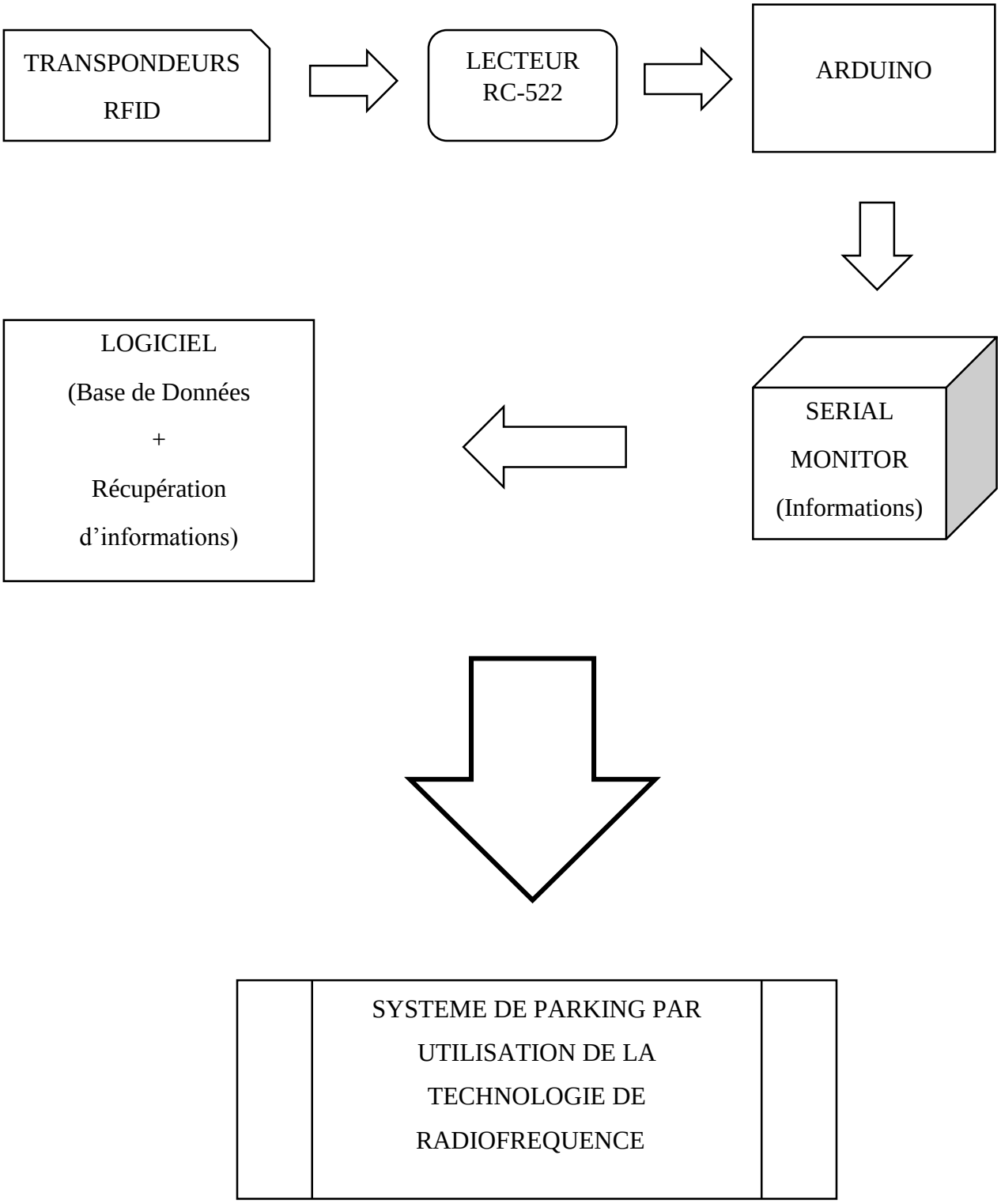
3.2.2.4 Interconnexion

Pour pouvoir effectuer le système de parking, on va enregistrer l'heure exacte où les cartes vont être présentées devant le lecteur. Ainsi, on va donc pouvoir définir l'heure qu'une voiture est entrée dans le parking et aussi l'heure de sortie. On part sur le principe que si une carte passe une fois devant le lecteur, l'heure exacte (selon l'horloge de l'ordinateur) va être ajouté dans la colonne heure d'entrée, et que quand cette même carte passe une deuxième fois devant le lecteur, à ce moment-là, l'heure va être ajouté dans la colonne heure de sortie de cette voiture.

Plus pratiquement parlant, le système de pointage du parking se déroule comme suit : le système va être installé à l'entrée du lieu de parking. Quand un véhicule va se garer dans un parking, qui utilise ce projet, un contrôleur va donner à son conducteur une carte RFID ; celui-ci va le présenter devant le lecteur et pointant ainsi son heure d'entrée et entrant cette heure dans la colonne « OraNidirany », dans la base de données du logiciel (qu'on a développé). Ce conducteur doit garder cette carte jusqu'à la sortie de son véhicule du parking. En sortant du parking avec son véhicule, le conducteur va de nouveau présenter sa carte devant le lecteur et pointant ainsi l'heure de sortie du véhicule et entrant aussi cette heure dans la colonne de sortie « OraNivoahany», dans la BD. Ainsi, on peut facilement avoir la montant, qui n'est rien d'autre que le droit de parking, en faisant la soustraction de l'heure d'entrée du véhicule à l'heure de sortie de celle-ci, et puis en multipliant ceci par le tarif du parking (pendant une heure par exemple). A ce moment, le contrôleur du parking n'aura plus qu'à utiliser le logiciel qu'on a développé pour afficher le droit de parking que le conducteur doit payer. Après avoir payé ce droit, le conducteur doit remettre la carte au contrôleur pour que cette même carte soit attribuée au prochain véhicule qui va occuper le parking et ainsi de suite.

3.2.2.5 Schéma synoptique

D’après toutes les étapes précédentes, on obtient le schéma synoptique suivant :



3.3 Conclusion

Le RC-522 est le module RFID utilisé dans ce projet. Le lecteur est monté avec la carte électronique embarqué arduino, selon le montage de la communication SPI. Des lignes de code, ayant chacun sa signification, sont ajoutés dans les fonctions de l'IDE arduino pour contrôler cet ensemble. Un logiciel a été développé pour fonctionner avec ce système. Ce logiciel a été développé avec le langage C# par l'utilisation de l'IDE Visual Studio. La BD, qui va aussi avec, est créé avec le logiciel MySql WorkBench et est utilisé avec le logiciel développé. L'interconnexion de l'ensemble permet de réaliser le projet intitulé « Système de parking par utilisation de la technologie de radiofréquence ».

CONCLUSION GENERALE

En se rendant compte que le stationnement des véhicules sur le bord des routes, ou des autoroutes, perturbent gravement la circulation routière, ce projet propose d'utiliser la technologie d'utilisation de radiofréquence, c'est-à-dire la technologie NFC, et plus précisément la technologie RFID pour y remédier. On a utilisé un shield RFID pour le mettre en place. On peut aussi considérer que l'évolution de l'électronique embarquée y a apporté son aide. On se permet de dire cela car on a utilisé la carte arduino, qui est une carte électronique embarqué, pour le contrôle de notre shield.

Après, on utilise l'IDE Arduino avec le langage associé pour bien faire fonctionner l'ensemble. Dans le domaine de la télécommunication, on a toujours à côtoyer le monde de l'électronique mais aussi celui de l'informatique. On a donc ainsi utilisé l'électronique ; mais dans ce projet, l'informatique a bien été aussi au rendez-vous. Ici, la connaissance en informatique nous permet de développer un logiciel via l'IDE Visual Studio de Microsoft. Ce logiciel qu'on a créé nous donne la possibilité d'avoir dans son interface les outils nécessaires pour faire la gestion du système de parckage. On a aussi eu recourt à la mise en place d'une base de donnée, qu'on a récupéré dans l'interface du logiciel, le support visuel qui est bien pratique pour visualiser les états du parking, c'est-à-dire les heures d'entrées et de sorties des véhicules.

C'est après tout ceci alors qu'on a mis en place un système de parckage qui, en le mettant en place, va permettre aux véhicules de s'y garer et évitant ainsi la perturbation routière. En guise d'amélioration de ce projet, on peut le modifier et, ensuite, y ajouter une reconnaissance numérique des numéros des immatriculations des véhicules. Ceci nous permet d'avoir tous les données et de savoir quelle voiture est passée sur le parking, à quelle heure, et le droit de parking que le conducteur a payé.

En sauvegardant tous ces données, le montant qui a été récolté peut être vérifié, et on connaitra aussi, à ce moment, toutes les informations sur le passage de chaque véhicule sur le parking. Ceci peut être utilisé pour des raisons quelconques. On peut aussi mettre en œuvre, avec ce projet, un système GPS que tout le monde peut utiliser et avec lequel on peut voir les emplacements des parking. On peut aussi substituer le support visuel et le mettre un écran embarqué compatible. A ce moment-là, le système aura le grand avantage d'être portable, mais il est quand même toujours utile de prendre en compte la situation du parking.

ANNEXES

ANNEXE 1 : CONTROLE DU MODULE RFID AVEC L'ARDUINO

Pour controller la carte électronique arduino, comme on l'a déjà dit dans les pages précedantes, on doit entrer des lignes de code nécessaire selon le projet que l'on veut réaliser. Dans ce projet, voici tous les extraits de lignes de code qu'on a besoin :

- Declaration de l'utilisation des librairies :

```
#include <SPI.h>
#include <RFID.h>
```

- Declaration de l'utilisation d'un module RFID :

```
RFID monModuleRFID(53,9); /*declarer-na oe mapiasa ny module RFID atsika isika*/
```

Dans la fonction qui ne serait exécutée qu'une seule fois :

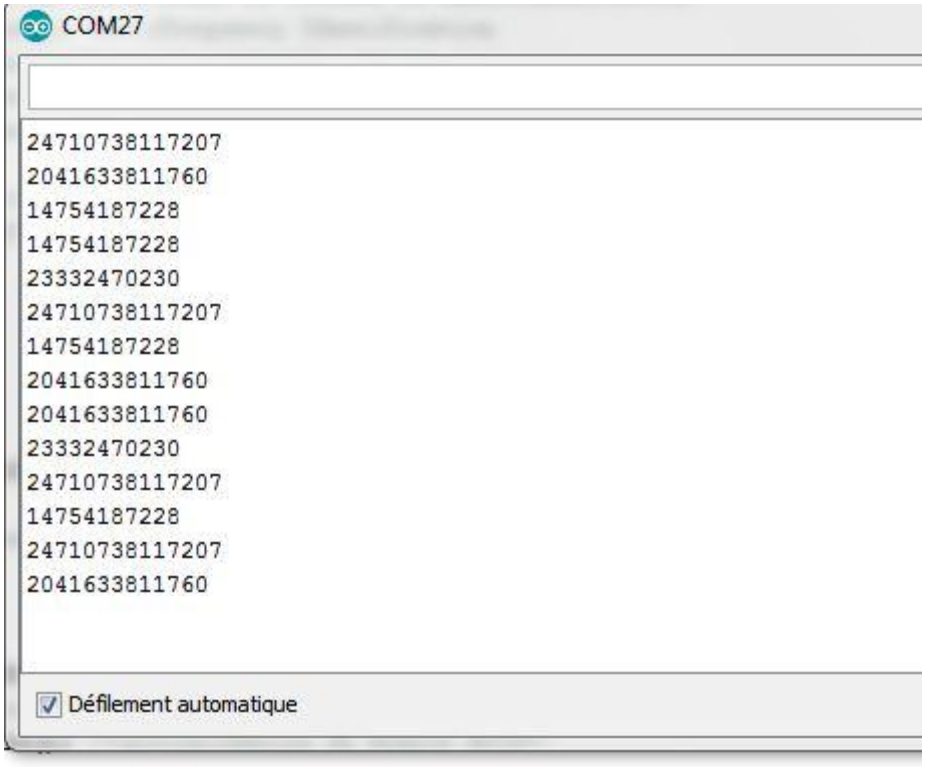
- Initialisation de la liaison via la communication SPI et aussi du module RFID :

```
SPI.begin(); /*initialisation de la communnication SPI*/
monModuleRFID.init(); /*initialisation du module RFID*/
```

Dans la fonction principale, voici les lignes de code qu'on va ordonner à la carte électronique de lire et exécuter en boucle infini :

```
if (monModuleRFID.isCard()) {  /*mijery raha misy tag na carte eo aloan'ny capteur*/
    if (monModuleRFID.readCardSerial()) { /*raha misy dia :*/
for(int i=0;i<=4;i++)   /*vakina ny numeroan'izay eo alohan'ny capteur*/
      {
       UID[i]=monModuleRFID.serNum[i];
       Serial.print(UID[i],DEC);
       }
      Serial.println("");
    }
    monModuleRFID.halt();
```

Voici comment se présentent les informations contenues dans le moniteur série durant l'exécution du programme :



ANNEXE 2 : LIGNE DE CODE LORS DU DEVELOPPEMENT DU LOGICIEL

Il a été déjà mentionné que ce projet met à notre disposition un logiciel développé avec le langage C#. Ci-dessous se trouvent les extraits de script qui ont été utilisés lors de la mise en œuvre de ce logiciel.

Pour effectuer certaines actions, il est nécessaire de charger des fonctionnalités. Les lignes de code suivantes indiquent dans quel lot de fonctionnalités, notre fichier va se baser pour pouvoir fonctionner :

```
using System.IO.Ports;
using MySql.Data.MySqlClient;
```

Ici, on indique tous les informations nécessaires pour la lecture des informations du moniteur série, puis on essaie de lire ces informations :

```
myport = new SerialPort();
myport.BaudRate = 9600;
myport.PortName = Anarana_port.Text;
myport.Parity = Parity.None;
myport.DataBits = 8;
myport.StopBits = StopBits.One;
myport.DataReceived += myport_DataReceived;

try
{
myport.Open();

}
catch (Exception ex)
{
MessageBox.Show(ex.Message, "Error");
}
}

void myport_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
in_data = myport.ReadLine();
this.Invoke(new EventHandler(displaydata_event));
}
private void displaydata_event(object sender, EventArgs e)
{
string a = in_data;
            long b = Convert.ToInt64(a);
            Information.Text = Convert.ToString(b);
 }
```

Pour arriver à établir une communication entre le logiciel et la BD, les lignes de code suivantes ont été ajoutées étape par étape :

- Dire tous les informations sur le BD pour que le logiciel obtienne l'autorisation de travailler avec lui :

```
String constring = "datasource=localhost; port=3306; username=root;
password=3210";
```

- Créer l'objet de connexion :

```
MySqlConnection conDataBase = new MySqlConnection(constring);
```

- Puis, on indique le requête SQL selon le résultat qu'on veut avoir. Voici un exemple de de commande requete SQL :

```
MySqlCommand cmdDataBase1 = new MySqlCommand("select * from
système_de_parking.parking;", conDataBase);
```

- Création de l'objet de lecture :

```
MySqlDataReader Myreader2;
```

- Quand on a réussi à établir l'objet de lecture, on peut ordonner des requêtes qu'on peut faire en manipulant les éléments du BD. Ici, par exemple, ce sont les lignes de code permettant d'essayer d'afficher une table du BD dans l'interface du logiciel :

```
try
        {
            MySqlDataAdapter sda = new MySqlDataAdapter();
            sda.SelectCommand = cmdDataBase1;
            DataTable dbdataset = new DataTable();
            sda.Fill(dbdataset);
            BindingSource bsource = new BindingSource();
            bsource.DataSource = dbdataset;
            dataGridView1.DataSource = bsource;
            sda.Update(dbdataset);
        }

        catch (Exception ex)
        {

            MessageBox.Show(ex.Message);


        }
```

BIBLIOGRAPHIE

[1] E.Saudrais, *« Electromagnétisme, Le champ électromagnétique variable, Equations de Maxwell* », CPGE PSI, Mars 2016

[2] D. Santos, « *Comprendre la RFID en 10 points* », SBE SAS – SAS 38457670800034 – 30 rue de Penthièvre – 75008 Paris – FRANCE, Fevrier 2016.

[3] M. Rouse, «*Radio frequency (RF, rf), RFID (radio frequency identification)* », TechTarget's IT and learning center, Mai 2008

[4] J. Geier, « *Wireless Networks first-step, Radio Frequency and Light Signal Fundamentals: The Invisible Medium, Understanding RF Signals* » Ltd. (www.wireless-nets.com) eTutorials, August 2004

[5] M. Boughanem, « *Cours Base de données relationnelle* », IUP STRI, Version Fevrier 2013

[6] S. Landrault et H. Weisslinger , « *Premier pas en informatique embarqué* », Le blog d'Eskimon, Juin 2016

[7] J.N. Rousseau, « *Créez votre premier programme sur Arduino* », Openclassroom, Nov. 2016

[8] J. LeChalupe, « *Cours d'initiation à Arduino* », ASTUPS Campus Fab., Nov. 2015

[9] A. T. Tiptop, « *Tutoriels pour Arduino PhPBB* », creating communities, Janvier 2017

[10] A. Pailhoux, « *Comment utiliser le module RFID RC-522* », Les électroniciens.com, Réseau communautaire de développeurs en électroniques, Octobre 2016

[11] N. Hilaire, « *Apprenez à developper en C#* », Expert .NET, Novembre 2016

[12] « *La RFID dans le spectre radio, Fonctionnement d'un système RFID* », Centre National de Reference RFID (CNRFID), Etablissement principal 5 avenue de Maneou 13790 ROUSSET, http://www.centrenational-rfid.com/introduction-a-la-rfid-article-15-fr-ruid-17.html

[13] « *Principe et systèmes RFID* », les experts Ooreka, https://rfid.ooreka.fr/comprendre/systeme-rfid

[14] « *Principe de fonctionnement des systemes RFID* », GS1, https://www.gs1.ch/fr/syst%C3%A8me-gs1/le-syst%C3%A8me-gs1/epcglobal/technologie-de-radiofr%C3%A9quence/principe-de-fonctionnement-des-syst%C3%A8mes-rfid.

[15] « *RFID (puces RFID*) », http://www.jobintree.com/dictionnaire/definition-rfid-puces-rfid-1024.html.

[16] « *Radio Fréquence Identification (RFID)* », http://economie.fgov.be/fr/consommateurs/Internet/Telecommunications/RFID/

[17] « *Base de données, Utilités d'une base de données* », http://www.commentcamarche.net/contents/104-bases-de-donnees-introduction,

FICHE DE RENSEIGNEMENT

Auteur :

Nom : TOKY

Prénoms : Rajaonarison Faniriharisoa Maxime

Adresse : Lot 112 T 43 Antsenakely

Antsirabe-110 Madagascar

Tel : +261340359352

Mail : tfaniriharisoa@gmail.com

Titre du mémoire :

«SYSTEME DE PARKING PAR UTILISATION DE LA TECHNOLOGIE DE RADIOFREQUENCE »

Nombre de pages : 63

Nombre de tableaux : 12

Nombre de figure : 44

Encadreur de mémoire :

Nom : RAKOTONDRAINA

Prénoms : Tahina Ezechiel

Tel : +261331176110

Mail : tahina.ezechiel@gmail.com