



2 BTS - Electromécanique & Systèmes Automatisés

Rapport du projet de fin d'étude

Thème:

**Parking automatisé par RFID et
enregistrement sur Excel**

Encadrant :

Youssef IMEL

Etudiant(s) :

**Nada Bouchani
Noureddine Akouchah**

Remerciement

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce projet de fin d'étude.

Je tiens à remercier chaleureusement **Monsieur Youssef Imel**, mon encadrant de stage, pour son accompagnement, ses conseils et sa disponibilité tout au long de cette expérience.

J'adresse également mes sincères remerciements à **Monsieur le Professeur El Mahfoud Boulaoutaq** pour son appui dans la partie **communication réseau**, ainsi qu'à **Monsieur le Professeur Azrou Lahcen** pour son encadrement dans **l'étude mécanique du système**.

Nos remerciements s'adressent également à nos familles et amis pour leur soutien moral et leurs encouragements constants. Leur présence et leur compréhension ont été une source de motivation précieuse.

Enfin, nous tenons à remercier tous nos collègues et camarades de promotion pour leur aide, leurs échanges constructifs, et leur camaraderie. Leur collaboration et leur esprit d'entraide ont enrichi notre expérience et ont rendu ce projet encore plus stimulant et agréable à réaliser.

À tous, nous adressons nos plus sincères remerciements.

Table des matières

Chapitre 1.....	6
Conception d'un système de parking automatisé par RFID.....	6
I. Introduction Générale	7
1.1 Contexte du projet	7
1.2 Problématique	8
1.3 Objectifs du projet	8
1.4 Méthodologie adoptée.....	8
II. Étude Fonctionnelle	9
1. Introduction	9
2. Cahier des charges fonctionnel	9
3. Analyse fonctionnelle :.....	10
I. Partie Commande (Arduino + RFID).....	13
A. La Technologie RFID	13
B. Liste des composants utilisés avec caractéristiques	18
C. Schéma de câblage via logiciel fritzing	21
I. Étude opérative et mécanique de la barrière motorisée	22
A. Calcul du poids estimé de la porte de garage	23
Calcul du volume du métal :.....	23
Calcul du poids :.....	23
B. Étude mécanique simplifiée du système de déplacement de la porte.....	29
Forces en jeu	29
C. Conclusion technique :	30
Chapitre 2.....	31

Réalisation de la partie commande et communication Arduino – Modbus

I. Intégration d'un variateur de fréquence pour le contrôle moteur	32
A. Recherche du Module de Communication pour l'Interface Arduino–Variateur PowerFlex 4	33
B. Intégration du Variateur de Vitesse et Communication Modbus via Convertisseur TTL/RS485	34
C. Enregistrement sur Excel	35
Chapitre 3.....	38
Tests validation et présentation du prototype	38
I. Introduction	39
A. Objectifs des tests	39
B. Scénarios de test	39
C. Présentation du prototype utilisé.....	41
Conclusion Générale.....	42
ANNEXE	43

Liste des figures

Figure 1 Diagramme Bête à cornes.....	10
Figure 2 Diagramme Pieuvre:	11
Figure 3 Tableau des interactions fonctionnelles du système	11
Figure 4 Architecture système RFID - Parking automatisé	12
Figure 5 les différentes technologies d'identification	14
Figure 6 L'étiquette (tag)	15
Figure 7 Le lecteur RFID.....	16
Figure 8 Principe de fonctionnement de la RFID	17
Figure 9 Arduino Uno R3	18
Figure 10 Module RFID	18
Figure 11 Écran LCD 16×2.....	19
Figure 12 LED de signalisation	19
Figure 13 Résistances de précision utilisées.....	19
Figure 14 Buzzer piézoélectrique Modèle.....	19
Figure 15 Module complet RS485	20
Figure 16 de tags RFID compatibles RC522	20
Figure 17 Schéma de câblage via logiciel fritzing.....	21
Figure 18 la garage étude.....	22
Figure 19 variateur de fréquence PowerFlex 4	32
Figure 20 le variateur via le protocole Modbus RS485	33
Figure 21 Le schéma de câblage de connexion entre la carte Arduino et le convertisseur	34
Figure 22 le fichier Excel	37
Figure 23 Le système répond correctement aux badges.....	40
Figure 24 Présentation du prototype utilise.....	41

Chapitre 1

Conception d'un système de parking automatisé par RFID

I. Introduction Générale

1.1 Contexte du projet

Dans de nombreuses entreprises, établissements scolaires et administrations, la gestion de l'accès aux parkings reste une tâche assurée manuellement. Cette gestion repose souvent sur la présence de personnel de sécurité ou sur des registres papier, ce qui entraîne plusieurs limitations telles que la perte de temps, les erreurs humaines et un manque de traçabilité.

Avec l'augmentation du nombre de véhicules et le besoin croissant de sécurisation des lieux, il devient indispensable d'opter pour des solutions automatisées, fiables et intelligentes. Un système de contrôle d'accès automatisé permet non seulement de renforcer la sécurité des installations, mais aussi de simplifier le travail des agents, en réduisant leur charge liée à la vérification manuelle des accès.

Dans ce contexte, ce projet a pour objectif de concevoir et de mettre en œuvre un système de **gestion automatisée d'un parking** basé sur la technologie **RFID**. Chaque utilisateur (employé, enseignant, étudiant, visiteur, etc.) est muni d'une carte RFID personnalisée, lui permettant d'accéder au parking sans intervention humaine. Le système se charge d'ouvrir automatiquement la barrière et d'enregistrer les données de passage (identifiant, date, heure) dans un fichier Excel pour assurer un **suivi en temps réel**.

Cette solution vise à **améliorer l'efficacité, la sécurité et la traçabilité** du parking dans un environnement professionnel ou éducatif.

1.2 Problématique

Comment concevoir et mettre en œuvre un système de contrôle d'accès automatisé pour un parking d'entreprise, capable de sécuriser les entrées et sorties tout en enregistrant les données des utilisateurs en temps réel ?

1.3 Objectifs du projet

- Automatiser l'accès à un parking via la technologie RFID.
- Commander une barrière motorisée à l'aide d'un Arduino Uno.
- Enregistrer les entrées et sorties des utilisateurs sur un fichier Excel automatiquement.
- Assurer la communication entre l'Arduino et un variateur de vitesse via un convertisseur TTL RS485.

1.4 Méthodologie adoptée

Le projet a été mené en suivant les étapes suivantes :

- Étude fonctionnelle du système à mettre en place.
- Conception de la partie commande (Arduino + RFID).
- Mise en œuvre de la partie opérative (moteur + variateur + RS485).
- Intégration de l'enregistrement sur Excel.
- Réalisation de l'étude mécanique du système.
- Tests, évaluation et amélioration du système.

II. Étude Fonctionnelle

1. Introduction

L'étude fonctionnelle vise à analyser le système du point de vue de ses fonctions attendues, des besoins utilisateurs, et des contraintes techniques. Elle permet de décomposer le système en fonctions principales et secondaires, facilitant ainsi sa conception et sa mise en œuvre. Dans le cadre de ce projet, le système doit permettre l'accès automatisé à un parking à l'aide de la technologie RFID, tout en assurant la sécurité et la traçabilité via un enregistrement sur Excel.

2. Cahier des charges fonctionnel

Besoins identifiés

- Réduire le temps d'attente à l'entrée du parking.
- Supprimer la gestion manuelle des accès.
- Sécuriser les entrées et sorties.
- Enregistrer les passages dans un fichier consultable.
- Permettre une compatibilité avec les badges existants.

3. Analyse fonctionnelle :

A. Diagramme Bête à cornes

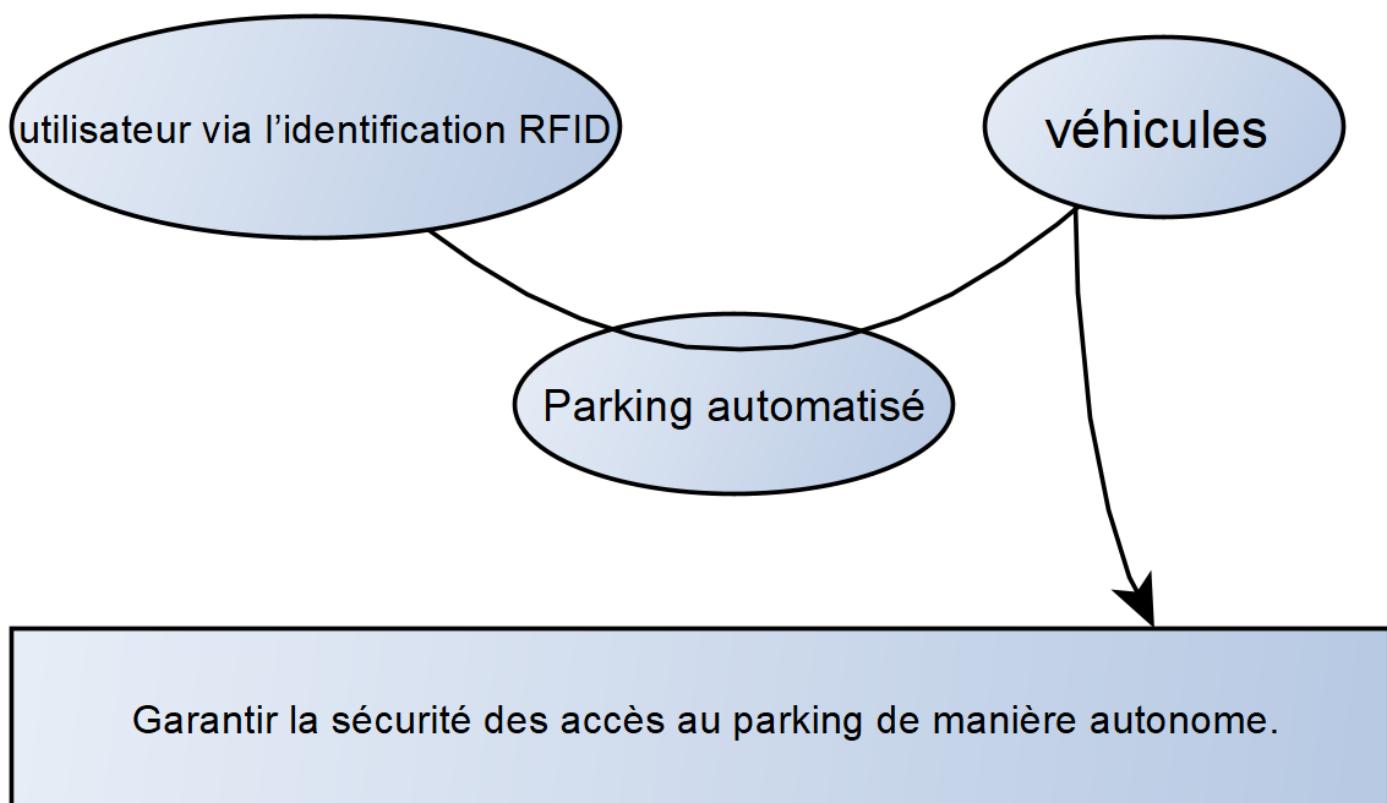


Figure 1 Diagramme Bête à cornes

B. Diagramme Pieuvre:

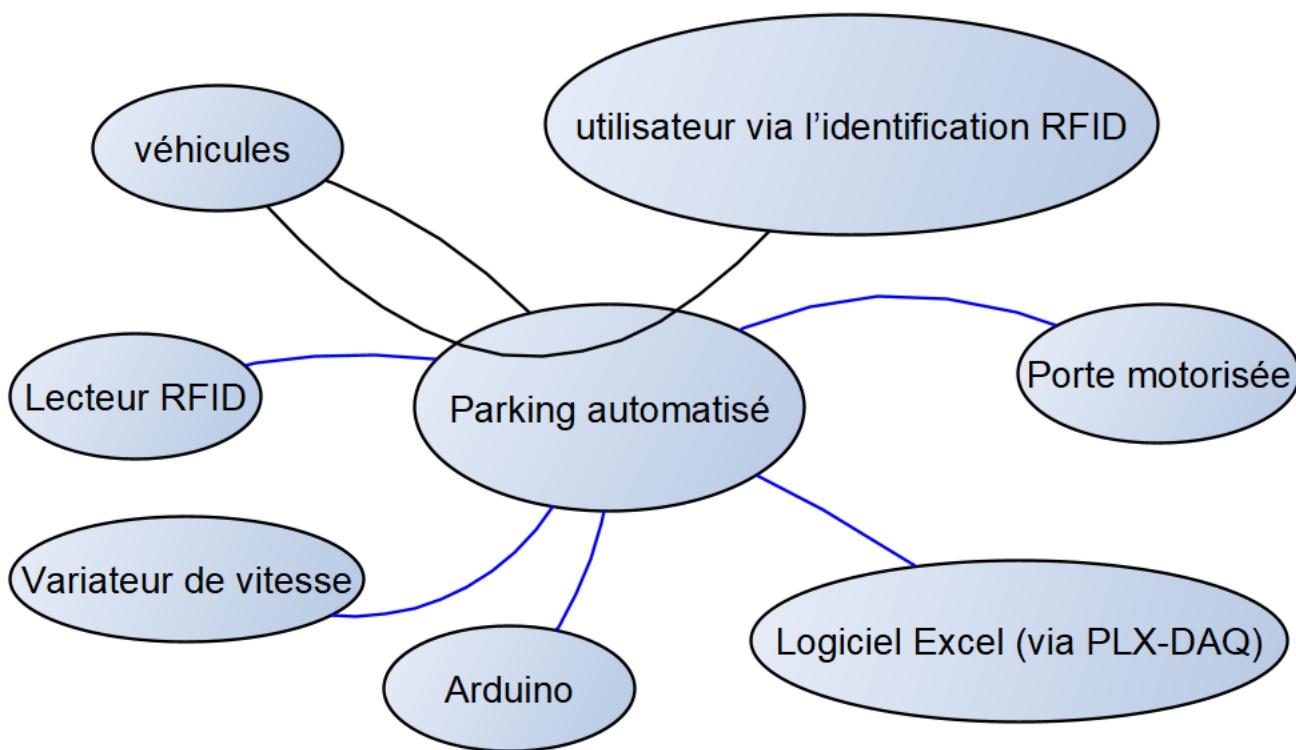


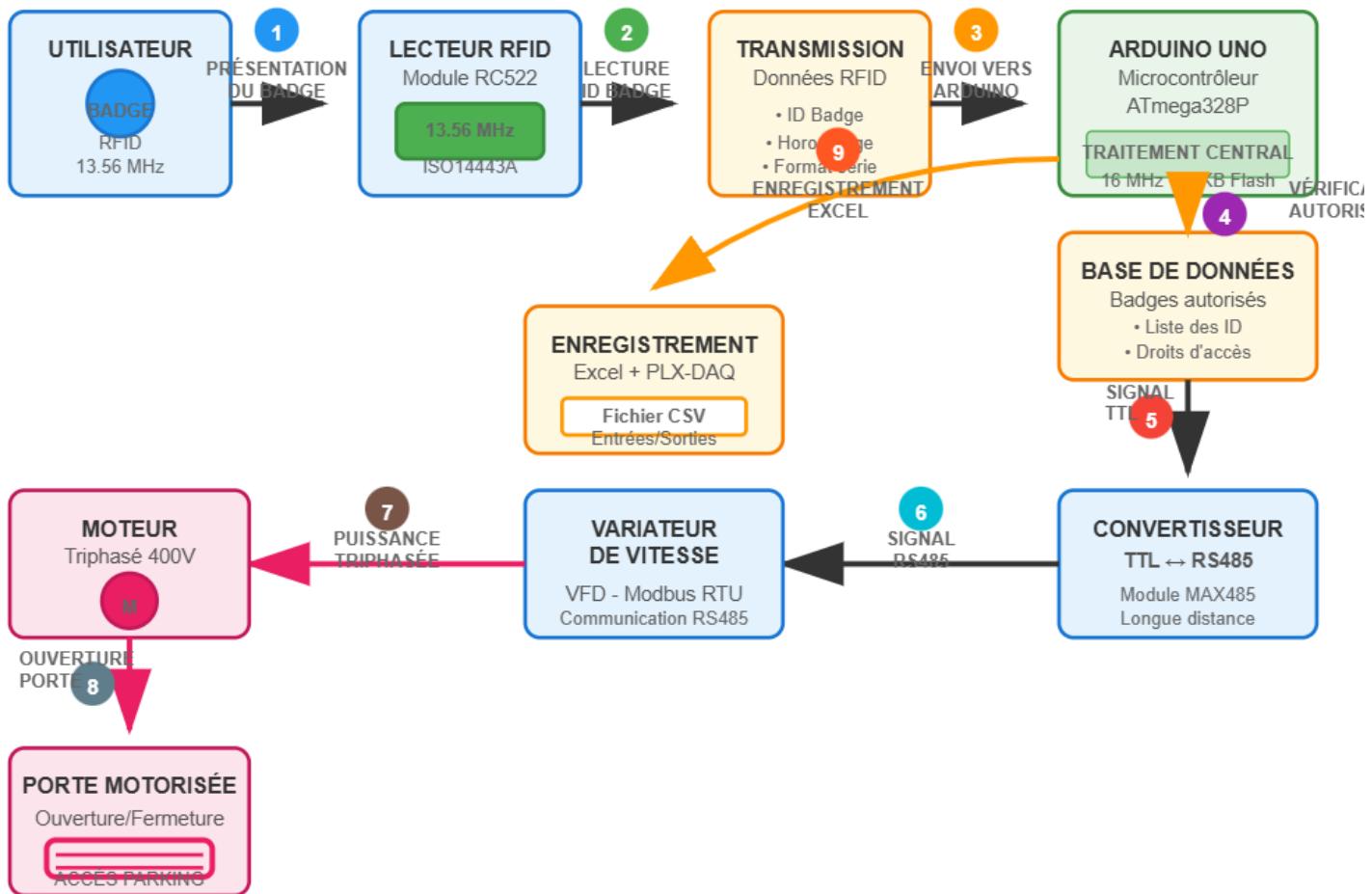
Figure 2 Diagramme Pieuvre:

Tableau des interactions fonctionnelles du système

Elément externe	Interaction avec le système	Type de fonction
Utilisateur	Présente un badge RFID	FP (principale)
Véhicule	Accède ou quitte le parking via la porte automatisée	FP (principale)
Lecteur RFID	Lit le badge et envoie l'identifiant à l'Arduino	FC (technique)
Arduino (microcontrôleur)	Gère la logique de commande	FC (technique)
Variateur de vitesse	Pilote le moteur selon les ordres de l'Arduino	FC (technique)
Porte motorisée	S'ouvre ou se ferme pour laisser passer un véhicule	FP (principale)
Logiciel Excel (via PLX- DAQ)	Enregistre les événements d'entrée/sortie	FP (enregistrement)

Figure 3 Tableau des interactions fonctionnelles du système

C. Architecture système RFID - Parking automatisé



SPÉCIFICATIONS TECHNIQUES

RFID: 13.56MHz, ISO14443A, portée 3-5cm
 Arduino: ATmega328P, 16MHz, 32KB Flash
 Communication: UART 9600 baud, RS485 Modbus

Convertisseur: MAX485, isolation galvanique
 Moteur: Triphasé 400V, commandé par VFD
 Logging: PLX-DAQ, format CSV avec horodatage

LÉGENDE

Électronique (Blue square) Traitements (Green square) Données (Orange square) Physique (Pink square) Contrôle (Black triangle) Données (Orange triangle) Puissance (Red triangle)

NOTES IMPORTANTES

Sécurité : Prévoir une alimentation de secours et une ouverture manuelle d'urgence
Maintenance Sauvegarder régulièrement la base de données des badges autorisés
Installation : Distance maximale RS485 : 1200m, blindage recommandé en environnement perturbé

Figure 4 Architecture système RFID - Parking automatisé

I. Partie Commande (Arduino + RFID)

Le système de commande constitue le cerveau de notre installation. Il est organisé autour d'un microcontrôleur Arduino Uno qui centralise toutes les informations et coordonne les actions. Cette architecture modulaire facilite le développement, les tests et la maintenance future.

L'Arduino communique avec quatre sous-systèmes principaux : le lecteur RFID pour l'identification, l'écran LCD pour l'interface utilisateur, le système moteur via relais pour l'action physique, et le PC via liaison série pour l'enregistrement des données. Cette approche distribuée améliore la fiabilité globale du système.

A. La Technologie RFID

La technologie d'identification par radiofréquence (Radio Frequency Identification RFID) est une technologie de capture automatique de données basées sur les ondes et rayonnements radiofréquence, elle est formée de trois composantes : une étiquette (ou plusieurs étiquettes), un lecteur ou un interrogateur ainsi que l'infrastructure de soutien correspondante (soit les composantes matérielles et les logiciels nécessaires)

A.1 Les différentes technologies d'identification

L'identification automatique se fait par plusieurs types de technologies. Dans ce présent travail, nous n'allons pas nous étendre sur tous les différents types de technologies, mais nous allons présenter quelques-uns à l'instar des codes à barres, des cartes intelligentes, des RFID passives et actives ainsi que les lecteurs d'empreintes.

- **Les codes à barres** : sont destinés à automatiser l'acquisition d'une information généralement numérique. Ils trouvent leurs applications dans différents domaines comme la gestion des prêts d'une bibliothèque, les caisses enregistreuses, etc.
- **L'étiquette RFID passive** : comme son nom l'indique, les tags passifs attendent un signal d'un lecteur RFID. Le lecteur envoie de l'énergie à une antenne qui convertit cette énergie en une onde RF qui est envoyée dans la zone de lecture. (Nous allons le détailler dans le paragraphe qui suit) [11].

- **L'étiquette RFID active :** comme les systèmes RFID passifs, les systèmes RFID actifs sont composés de lecteurs, d'étiquettes et d'antennes. Cependant, les systèmes passifs exigent que les étiquettes soient activées par le lecteur, les systèmes RFID actifs utilisent des étiquettes RFID alimentées par batterie qui ne nécessitent pas d'énergie pour envoyer un signal [12].
- **Le lecteur d'empreinte :** est parfait pour les gestionnaires RH (Ressource Humaine) de toute entreprise petite ou moyenne. Ce lecteur d'empreintes digitales fera des merveilles et facilitera le travail en permettant de suivre de plus près le temps des employés [13].

Sur le tableau ci-dessous, nous présentons une étude sur les différentes technologies d'identification automatique en faisant une comparaison sur leurs avantages et leurs inconvénients .

Technologies	Modification Des données	Sécurité de données	Volume de données	Coûts	Durée de vie
Codes à barres	Non modifiable	Minimale	Codes a barre linéaire contient 8-30 bytes. Certains codes a barre a 2D contient jusqu'a 7200 bytes.	Bas	Court
Etiquettes RFID active	Modifiable	Haute	Jusqu'à 8 MB	Très élevé	3-5 ans
Etiquettes RFID passive	Modifiable	Moyenne	Jusqu'à 64 KB	Moyen	Indéfini
Lecteurs d'empreintes	Non modifiable	Moyenne	certains lecteurs peuvent mémoriser jusqu'à 9000 empreintes.	Moyen	/
Carte intelligente	Modifiable	Haute	Jusqu'à 8 MB	Elevé	Long

Figure 5 les différentes technologies d'identification

A.2. Composants des systèmes RFID

Pour pouvoir mettre en place un système de traçabilité RFID, nous devons avoir un équipement spécifique composé de :

L'étiquette (tag)

C'est un dispositif récepteur, que l'on place sur des éléments (objet, animal...). Ils sont munis d'une puce contenant les informations et d'une antenne pour permettre les échanges d'informations. La figure ci-dessous, montre une étiquette à radiofréquence qui se compose d'une puce et d'une antenne.

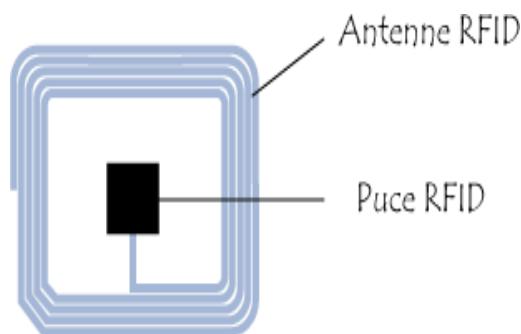


Figure 6 L'étiquette (tag)

Le lecteur RFID

Le lecteur/enregistreur est constitué d'un circuit qui émet une énergie électromagnétique à travers une antenne, et d'une électronique qui reçoit et décode les informations envoyées par l'étiquette et les envoie au dispositif de collecte des données. Le lecteur RFID est l'élément responsable de la lecture des étiquettes radiofréquence et de la transmission des informations qu'elle ; s contiennent.



Figure 7 Le lecteur RFID

Principe de fonctionnement de la RFID

Un système RFID est composé de deux entités qui communiquent entre elles :

- Un tag ou étiquette intelligente (aussi appelé transpondeur), associé à l'élément à identifier. Il est capable de répondre à une demande venant d'un lecteur.
- Une station de base ou lecteur RFID qui a pour mission d'identifier le tag. Le lecteur envoie une onde électromagnétique en direction de l'élément à identifier. En retour, il reçoit l'information renvoyée par le tag.

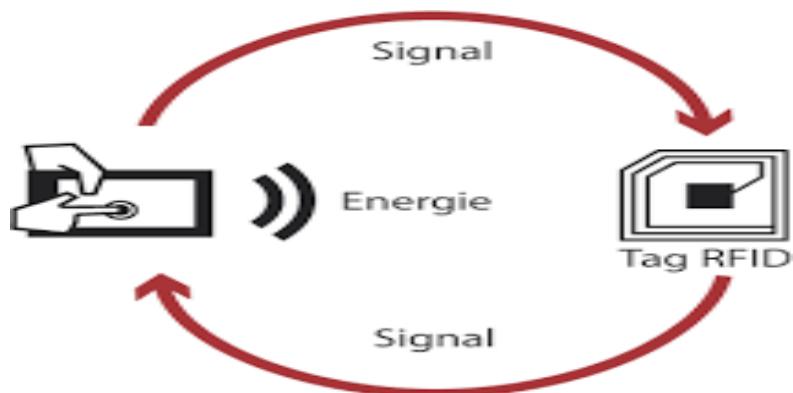
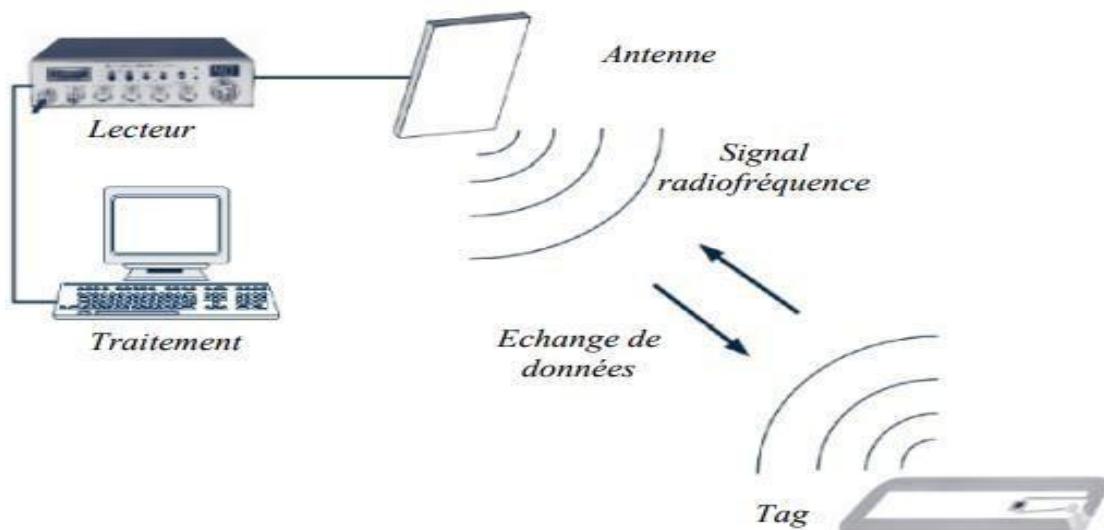


Figure 8 Principe de fonctionnement de la RFID

B. Liste des composants utilisés avec caractéristiques

- Arduino Uno R3 - Référence A000066**

Microcontrôleur : ATmega328P cadencé à 16 MHz

Alimentation : 7-12V DC via jack ou 5V USB

Entrées/Sorties : 14 pins numériques (6 PWM) + 6 pins analogiques

Mémoire : Flash 32 KB, SRAM 2 KB, EEPROM 1 KB

Communication : UART, SPI, I2C intégrés

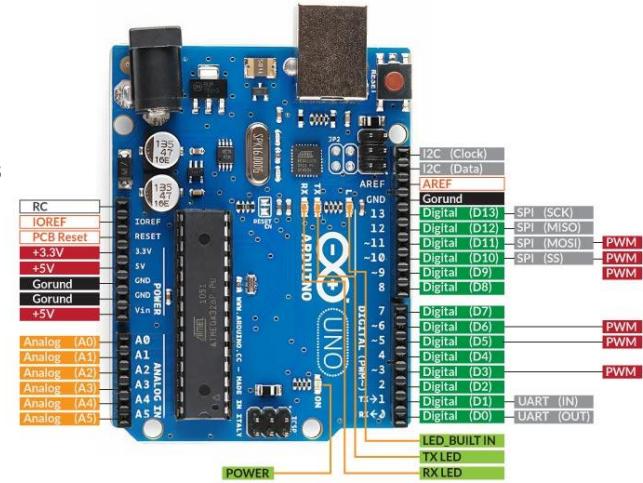


Figure 9 Arduino Uno R3

- Module RFID RC522 RFID Reader - Fréquence 13.56 MHz**

Chipset : MFRC522 de NXP Semiconductors

Protocole supporté : ISO14443A (MIFARE Classic, MIFARE Ultralight)

Fréquence porteuse : 13.56 MHz \pm 7 kHz

Distance de lecture : 0-60 mm (dépend de la taille de l'antenne)

Interface : SPI (4 fils + alimentation)

Tension d'alimentation : 3.3V \pm 5%

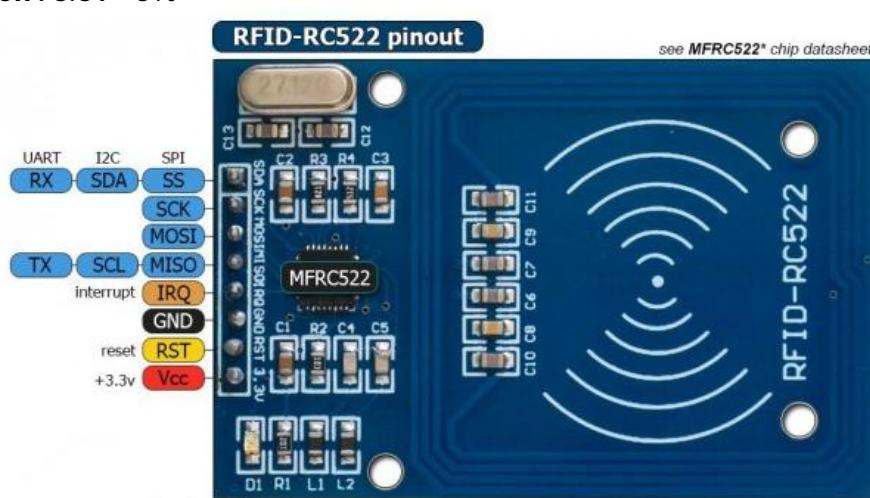


Figure 10 Module RFID

- **Écran LCD 16×2 avec module I2C - HD44780**

Affichage : 16 caractères × 2 lignes

Contrôleur : HD44780 compatible

Interface : I2C via PCF8574T (adresse 0x27)

Tension d'alimentation : 5V ±10%

Consommation : 30 mA avec rétroéclairage

Rétroéclairage : LED bleues avec contrôle PWM

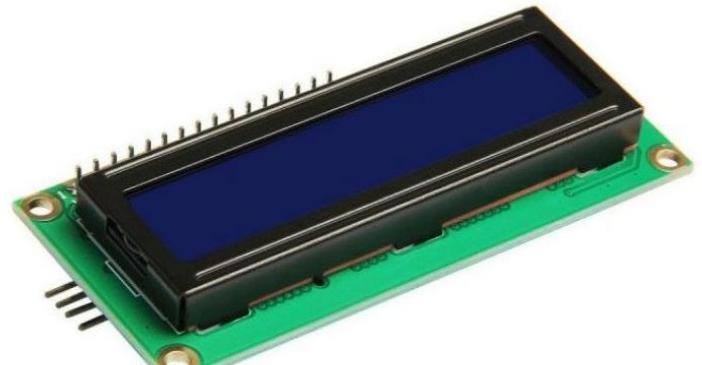


Figure 11 Écran LCD 16×2

- **LED de signalisation - 5mm haute luminosité**

Modèle : L-154A4SURKMVGPCBA (Kingbright)

Tension directe rouge : 2.0V typ @ 20mA

Tension directe verte : 3.2V typ @ 20mA



Figure 12 LED de

- **Résistances de précision utilisées**

Série : E12 (12 valeurs par décade)

Tolérance : ±1% (haute précision)



Figure 13 Résistances de précision utilisées

- **Buzzer piézoélectrique Modèle : PKM22EPPH4001-B0 (Murata)**

Type : Piézoélectrique avec oscillateur intégré

Tension : 3V-20V DC (optimal 5V)

Courant : 30mA @ 5V

Fréquence : 4000 Hz ± 500 Hz



Figure 14 Buzzer piézoélectrique Modèle

- **Module complet RS485 :**

Référence : MAX485-TTL-RS485-Module

Dimensions : 44 × 14 mm

Connecteurs : Borniers vis + pins Arduino

LEDs d'activité TX/RX

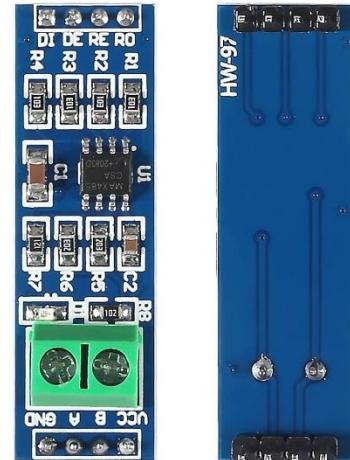


Figure 15 Module complet RS485

- **Types de tags RFID compatibles RC522**

Standard supporté : ISO14443A (13.56 MHz)

MIFARE Classic 1K

Spécifications techniques :

- **Référence :** NXP MIFARE Classic MF1S50yyX/V1
- **Mémoire :** 1024 bytes (1K)
- **Organisation :** 16 secteurs × 4 blocs × 16 bytes
- **UID :** 4 bytes (32 bits) - Numéro unique
- **Fréquence :** 13.56 MHz
- **Distance lecture :** 2-10 cm (selon antenne)



Figure 16 de tags RFID compatibles RC522

C. Schéma de câblage via logiciel fritzing

Le schéma de câblage ci-dessous illustre l'interconnexion entre les différents composants de la partie commande du système, notamment l'Arduino Uno, le lecteur RFID RC522, l'écran LCD I2C, le convertisseur TTL/RS485 ainsi que les périphériques d'affichage (LED, buzzer). Ce montage permet la lecture des badges RFID, le contrôle de la barrière motorisée et la transmission des données vers Excel.

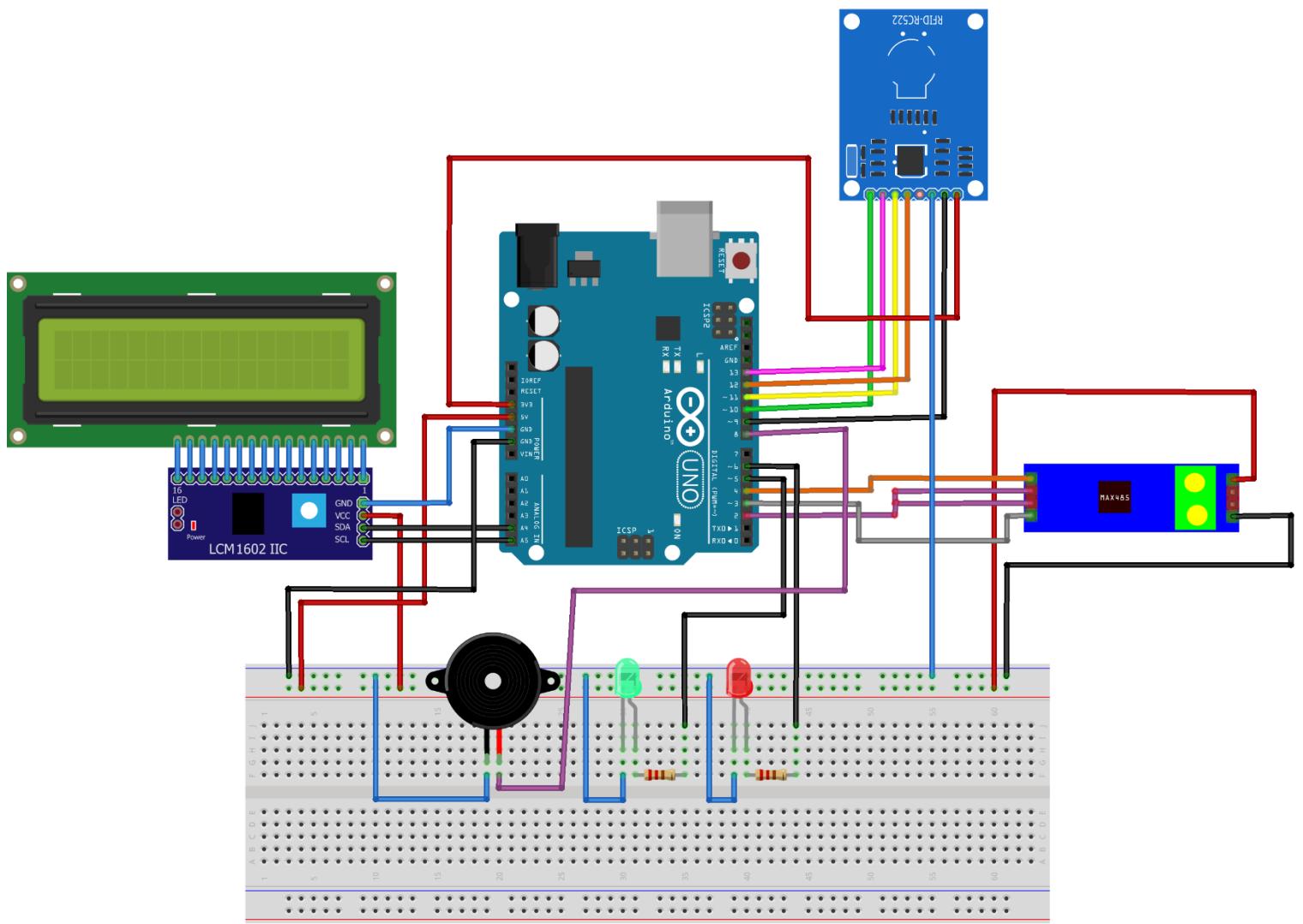


Figure 17 Schéma de câblage via logiciel fritzing

I. Étude opérative et mécanique de la barrière motorisée

L'étude opérative et mécanique constitue une étape cruciale dans la conception du système afin d'assurer son bon fonctionnement, sa fiabilité et sa durabilité. Cette étude permet d'analyser les contraintes mécaniques, les forces en présence, ainsi que les caractéristiques dynamiques liées au déplacement de la garage.

Le choix du moteur est directement lié au poids de la barrière, qui représente la charge principale à déplacer. Pour dimensionner correctement le moteur, il est indispensable de connaître précisément ce poids ainsi que les dimensions de la barrière, car ils influencent le couple et la puissance nécessaires. Une estimation précise du poids permet de calculer la force requise pour surmonter la gravité, les frottements au niveau des articulations, et l'inertie lors des mouvements d'ouverture et de fermeture.

En tenant compte de ces paramètres, l'étude mécanique inclut également le calcul des moments de force, la vitesse angulaire souhaitée, ainsi que la fréquence d'utilisation de la barrière. Ces éléments permettent de sélectionner un moteur adapté, capable de fournir la puissance nécessaire sans surconsommation d'énergie ni usure prématuée des composants.



Figure 18 la garage étude

Prise de mesures du garage

Dans le cadre de notre projet de PFE, mon collègue et moi avons procédé à la prise de mesures sur le site afin de recueillir les dimensions exactes de la porte existante.

A. Calcul du poids estimé de la porte de garage

Données :

- Longueur (largeur de la porte) : 4,51 m
- Hauteur : 2,35 m
- Épaisseur du fer : 3 mm = 0,003 m
- Densité de l'acier : 7850 kg/m³

Calcul du volume du métal :

Le volume d'une seule face de la porte est donné par la formule :

$$\text{Volume d'une face} = \text{Longueur} \times \text{Hauteur} \times \text{Épaisseur}$$

$$\text{Volume d'une face} = 4,51 \times 2,35 \times 0,003 = 0,0318 \text{ m}^3$$

Comme il y a deux faces métalliques (avant et arrière), le volume total est :

$$\text{Volume total} = 0,0318 \times 2 = 0,0636 \text{ m}^3$$

Calcul du poids :

Le poids de la porte est calculé en multipliant le volume total par la densité du métal :

$$\text{Poids} = \text{Volume total} \times \text{Densité}$$

$$\text{Poids} = 0,0636 \times 7850 = 499,86 \text{ kg}$$

Calculer la puissance mécanique nécessaire pour déplacer une porte de garage de 499,68 kg, puis proposer une puissance moteur triphasé adaptée.

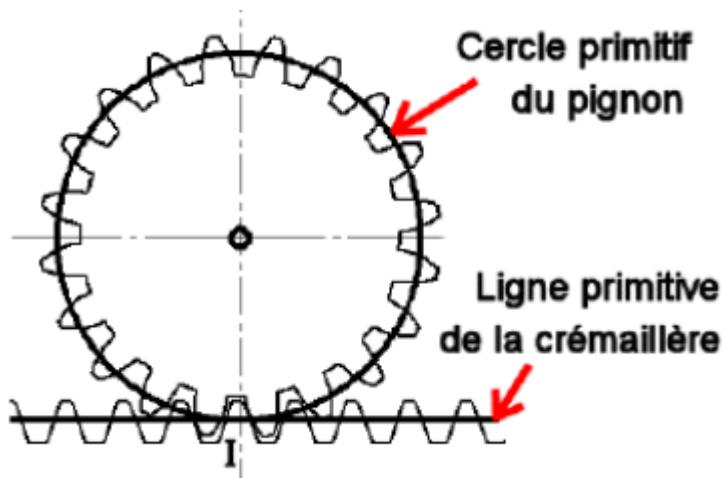
Données :

Masse de la porte (m) = 499,68 kg Gravité

(g) = 9,81 m/s²

$$F = m \times g = 499,68 \times 9,81 = 4902 \text{ N}$$

Par Frottements mécaniques :



Coefficient de frottement

$$F_t = m \times g \times f$$

$$= 499,68 \times 9,81 \times 0.04$$

$$F_t = 196.07$$

et f Coefficient de frottement

On prend f = 0.04

C'est l'effort que doit exercer le pignon sur la crémaillère pour déplacer la porte du garage à vitesse constante.

la puissance mécanique P, on utilise la formule suivante :

$$P = \frac{F \times V}{\eta}$$

Mais si le rendement η (efficacité du système) n'est pas donné, on considère :

$$P = F \times V$$

$$P=4900,57 \times 0,2=980,11W (\approx 0,98 \text{ kW})$$

On a $\varepsilon = F \cdot r$

$$P = F \cdot V$$

Avec P en W Ft en N V en m.s

$$V = 12 \text{ m/min} = 0,2 \text{ m/s}$$

$$\omega = V/r = 0,2$$

Avec ω la vitesse angulaire du pignon

r rayon de pignon

M

Reducteur

Npignon

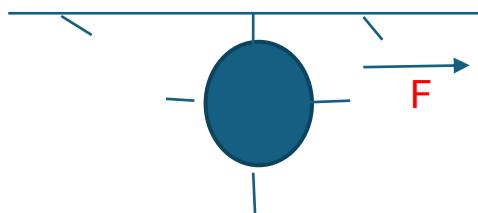
Calculer de puissance

$$P = F \cdot V$$

$$= 196.14.0,2 = 39,228 \text{ N}$$

$$\Omega = ?$$

$$Nm = N_{\text{pignon}} / Nm = 1000 / 1500 = 235605 / 1500 = 157,07$$

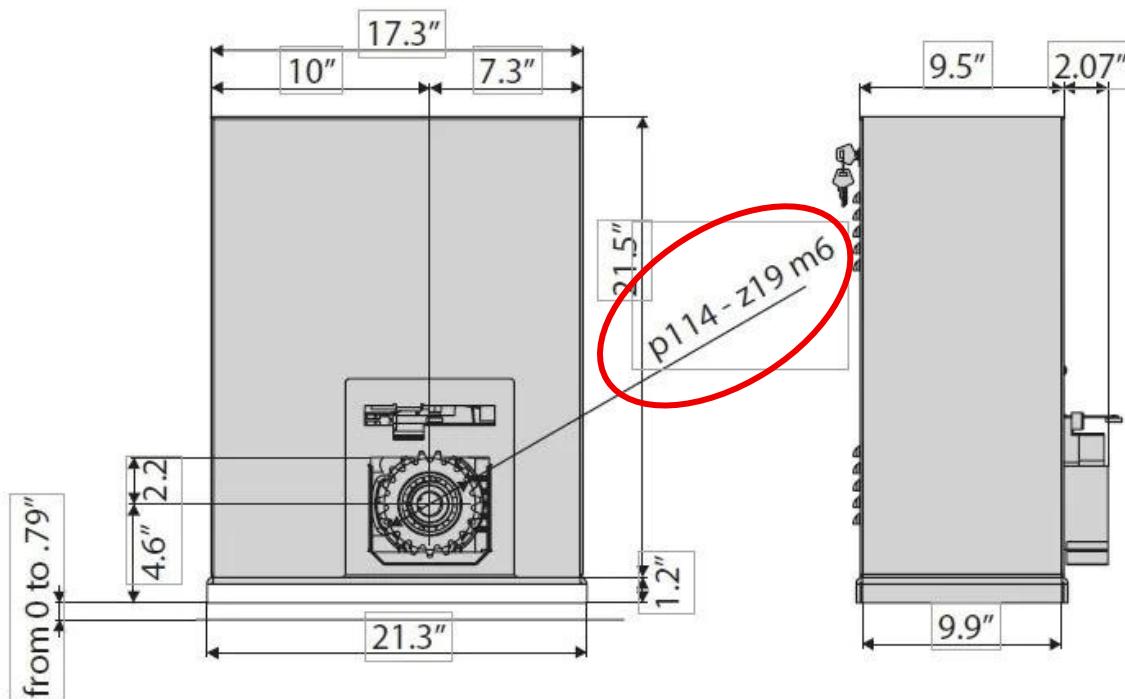


A. Calcul de RDM (résistance des matériaux)



« Pour garantir la solidité de la poutre, comment calculer le diamètre nécessaire de l'arbre en fonction des charges et contraintes appliquées ? »

À partir du dessin de définition de l'arbre moteur, il est nécessaire d'extraire le diamètre afin de procéder aux calculs de résistance et de dimensionnement.



Données visibles sur le schéma :

- Il est indiqué sur le pignon :

$$\text{Ø}114 - Z19 m6$$

Cela signifie :

- Ø114 mm** → diamètre du cercle primitif du pignon (pas le diamètre de l'arbre directement).
- Z19** → nombre de dents du pignon.
- m6** → ajustement de l'alésage du pignon sur l'arbre (ajustement serré).

Estimation du diamètre de l'arbre :

Pour un pignon de module mm avec $Z= 19$ dents :

$$\text{Diamètre primitif } d = m \times Z = 114 \Rightarrow m = \frac{114}{19} \approx 6$$

Donc c'est un pignon module 6.

Le diamètre d'alésage (donc celui de l'arbre) pour un pignon de module 6, 19 dents est normalement **≈ 35 à 40 mm**, mais dépend de la norme du fabricant.

Cependant, comme il y a une ajustement m6, et si on se base sur la norme ISO :

- Ø35 mm (m6) est une dimension standard.
- C'est souvent le diamètre de l'arbre sur lequel le pignon est monté.

Diamètre de l'arbre (d) = **35mm**

Calcul M_t

On a $M_t = P/\omega = 30P/\pi.N$

AN $M_t = 30.235,24/\pi.150 = 14,5 \text{ Nm}$

Calcul de Reg

$M_t \leq \text{Reg}$

$I_0/r = \pi.d^3/16 = \pi.15^3/16 = 662,67$

Calcul de S

On a $M_t.s/I_0$ avec $I_0 = \pi.d^3/16$

DONC $s = \omega \cdot \pi \cdot d^3/16P$

AN

$s = 2\pi.1500. \pi.35/60.16.39,228$ d ou s = **27, 51**

B. Étude mécanique simplifiée du système de déplacement de la porte

Après avoir défini les **dimensions** (4,51 m x 2,35 m) et le **poids estimé** de la porte ($\approx 499,86$ kg), on peut effectuer une **analyse mécanique** du système pour vérifier que la **solution retenue** (moteur + pignon-crémaillère) est adaptée.

Forces en jeu

Le système doit exercer une **force horizontale suffisante** pour :

- Surmonter l'inertie du poids de la porte (démarrage).
- Résister aux **frottements mécaniques** (rails, galets).
- Maintenir un mouvement régulier à **12 m/min.**
- **Choix du système pignon-crémaillère**

Ce système transforme le **couple moteur** en **force de translation linéaire**. Il est bien adapté aux portes lourdes



C. Conclusion technique :

Type : Moteur triphasé asynchrone

Puissance nominale : 1,5 kW (2 CV)

Tension d'alimentation : 380 V triphasé

Vitesse de rotation : adaptée à ton système (souvent 1400 à 1500 tr/min)

Commande : Variateur de fréquence PowerFlex 4

Chapitre 2

Réalisation de la partie commande et communication Arduino – Modbus

I. Intégration d'un variateur de fréquence pour le contrôle moteur

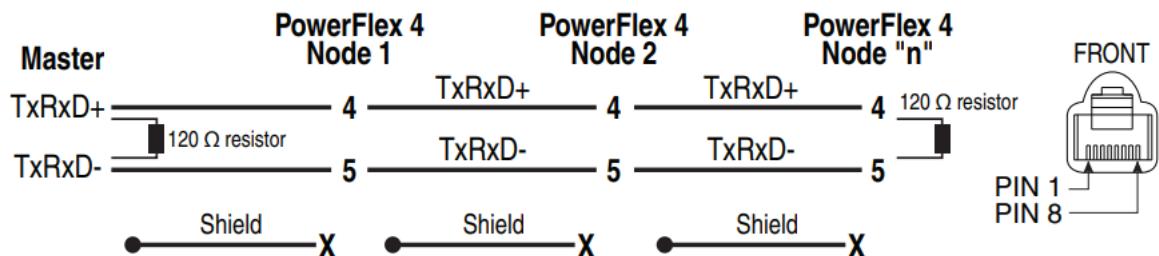
Nous avons intégré un variateur de fréquence PowerFlex 4 afin de piloter le moteur de manière précise et fiable. Ce variateur a été choisi pour sa compatibilité avec la communication Modbus, permettant un contrôle via Arduino à travers un adaptateur TTL/RS485. Cela assure une intégration fluide avec le système de commande basé sur Arduino, tout en garantissant la robustesse de la transmission des données.



Figure 19 variateur de fréquence PowerFlex 4

A. Recherche du Module de Communication pour l'Interface Arduino–Variateur PowerFlex 4

Actuellement, nous sommes en phase de recherche du module de communication adapté pour le variateur de vitesse PowerFlex 4, afin d'assurer une interface efficace entre l'Arduino (niveau TTL) et le variateur via le protocole Modbus RS485. Cette étape est essentielle pour garantir la compatibilité des échanges de données et la fiabilité du pilotage moteur.



NOTE: The shield is connected at ONLY ONE end of each cable segment.

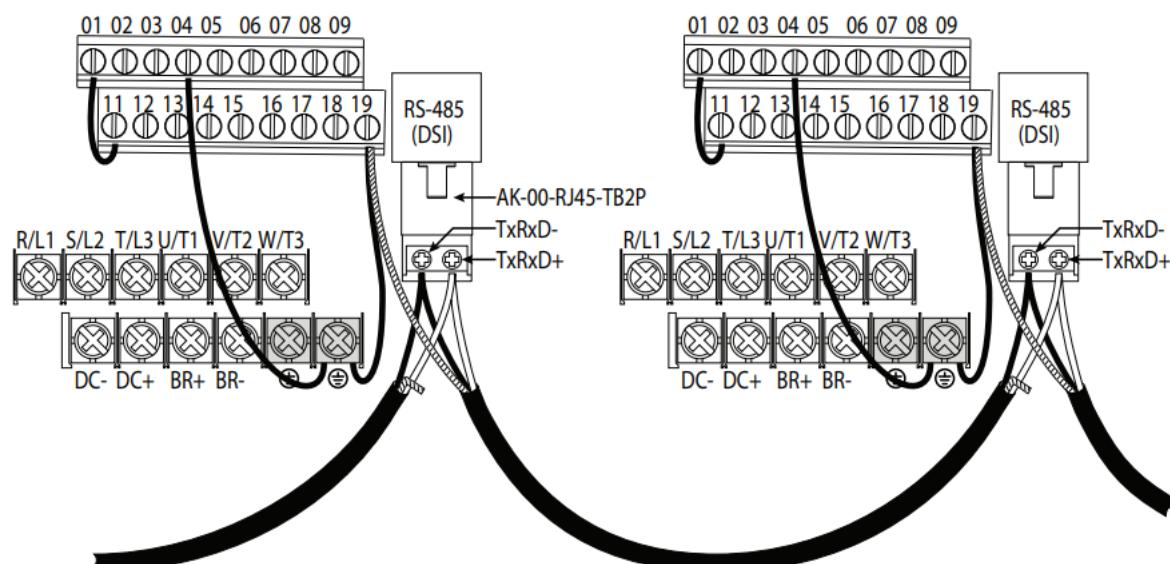


Figure 20 le variateur via le protocole Modbus RS485

B. Intégration du Variateur de Vitesse et Communication Modbus via Convertisseur TTL/RS485

Le schéma de câblage ci-dessous illustre la connexion entre la carte Arduino et le convertisseur TTL vers RS485, permettant la communication série avec le variateur de fréquence via le protocole Modbus. Cette liaison assure la transmission fiable des commandes de contrôle moteur.

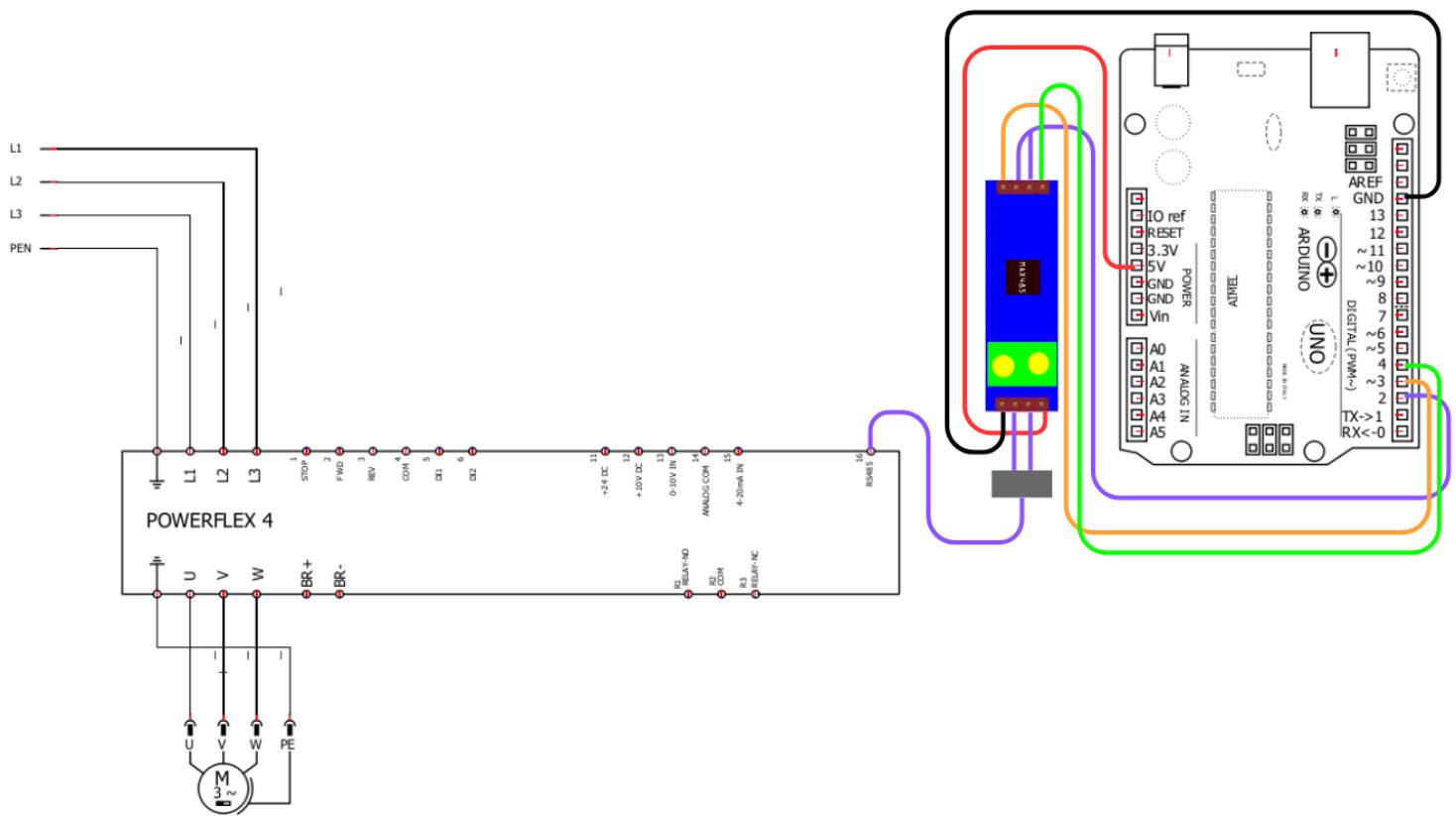
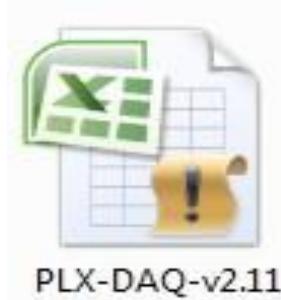


Figure 21 Le schéma de câblage de connexion entre la carte Arduino et le convertisseur

C. Enregistrement sur Excel

C.1 La macro PLX-DAQ

PLX-DAQ est une macro pour Excel mise au point par la société Parallax pour permettre aux microcontrôleurs qu'elle commercialise d'envoyer des données vers Excel. C'est gratuit, et ça fonctionne avec n'importe quel microcontrôleur capable de communication série, ce qui inclut bien sûr l'Arduino. Avec PLX-DAQ, nous pouvons envoyer les données en temps réel collectées par Arduino dans Excel, où il est beaucoup plus facile de traiter les données.



Remarque : Par défaut, Excel est plutôt paranoïaque en ce qui concerne les macros : en mode "Niveau de sécurité élevé", il refuse de les exécuter, par crainte qu'elles contiennent des virus. Pour vérifier le niveau de sécurité de votre exemplaire d'Excel (et pour le modifier s'il n'est pas adéquat), choisissez "Options" dans le menu "Outils".

C.2 Partie Arduino

Dans la fonction de configuration du sketch Arduino, incluez ces codes :

- Serial.begin (9600);
- Serial.println("CLEARDATA");
- Serial.println("LABEL,Acolumn,Bcolumn,...");
- Serial.println("RESETTIMER") ;

Il s'agit de directives de contrôle pour le PLX-DAQ.

- CLEARDATA efface les données précédentes dans la colonne,
- la directive LABEL spécifie le nom des colonnes,
- RESETTIMER réinitialise le timer interne.

Dans la fonction **loop()**, incluez d'abord cette directive de contrôle.

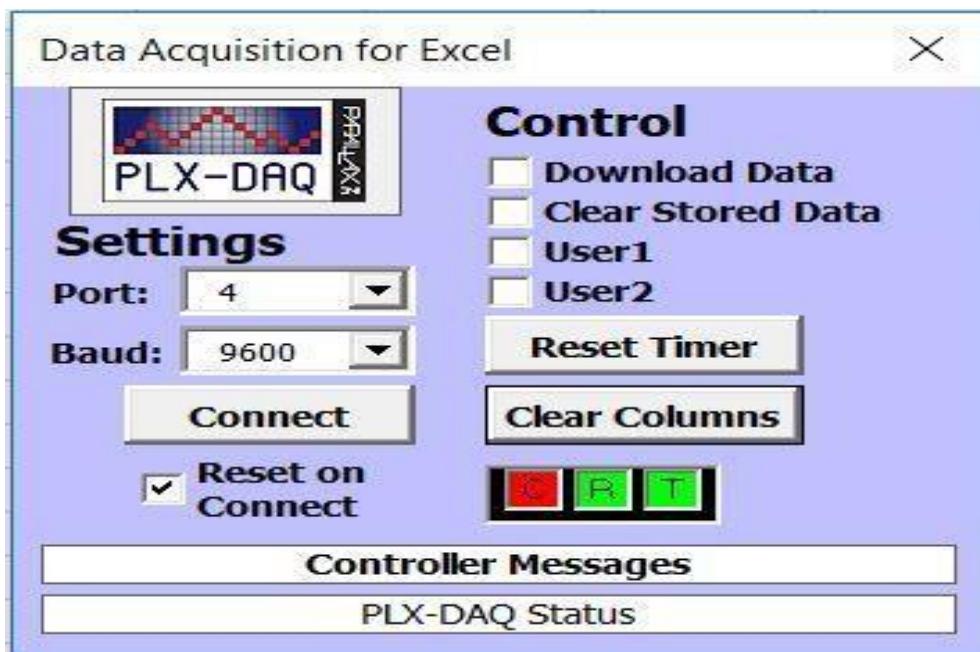
- `Serial.print("DATA,TIME,TIMER,")`: DATA spécifie que le reste de la sortie série sera étiqueté comme données et enregistré dans la colonne. Les deux premiers paramètres sont TIME qui spécifie l'heure actuelle et TIMER qui enregistre le temps écoulé depuis son démarrage.

Ensuite, il suffit d'imprimer les valeurs qui ont dû être imprimées. N'oubliez pas de commencer une nouvelle ligne et d'inclure un délai à la fin :

- `Serial.println();`
- `delay(100);`

C.3 Partie Excel

Lancer le fichier Excel ; cliquez sur OK dans la boîte de dialogue. Celle-ci apparaîtra alors :



Choisissez le bon port dans Arduino et le bon baud, cliquez sur "connecter", et les données devraient apparaître dans le fichier Excel. (N'oubliez pas de télécharger le code Arduino)

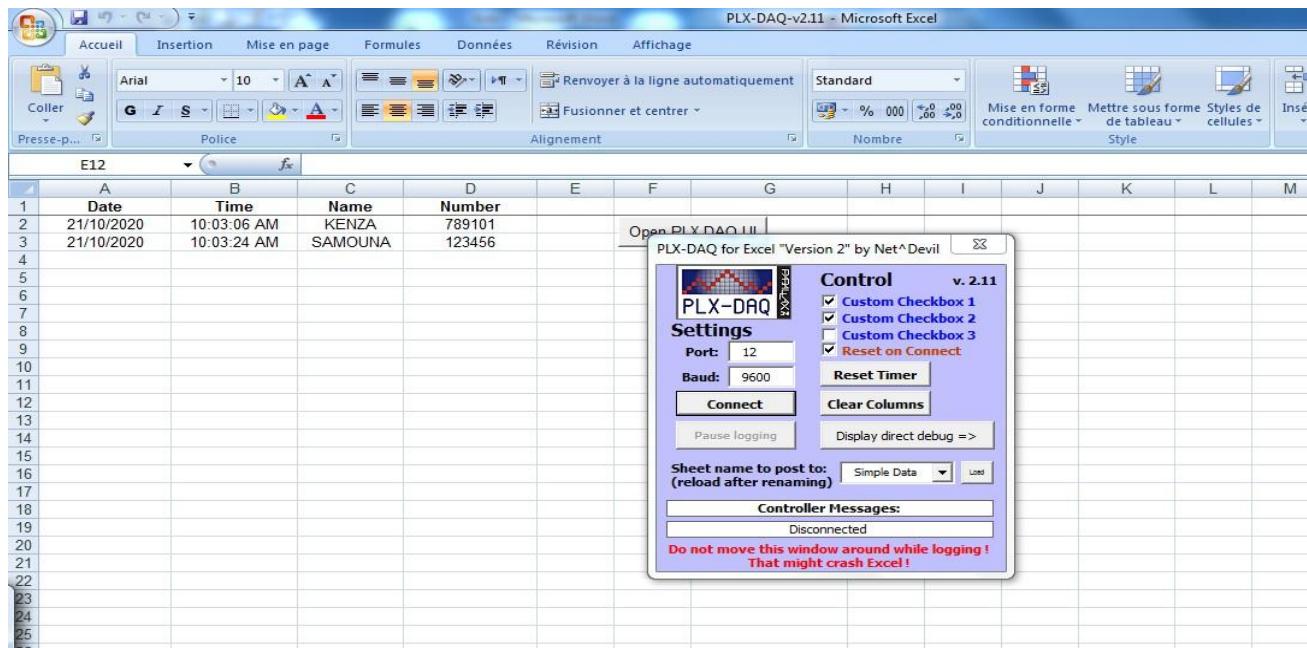


Figure 22 le fichier Excel

Le fichier Excel comporte trois onglets, il s'agit de données simples, de données simples avec tracés et d'un graphique à barres interactif. Le script s'exécutera sur la première position de l'onglet ; déplacez les onglets en conséquence pour répondre aux différents besoins.

Chapitre 3

Tests validation et présentation du prototype

I. Introduction

Après la phase de conception et de réalisation, une série de tests a été menée afin de valider le bon fonctionnement du système de parking automatisé. Ces tests ont pour objectif de vérifier la fiabilité de la lecture RFID, la réponse de la commande moteur, ainsi que l'enregistrement des données. Ce chapitre présente les scénarios de tests réalisés, les résultats obtenus, ainsi qu'une remarque importante concernant le prototype utilisé pour la soutenance.

A. Objectifs des tests

- ❖ Vérifier que le lecteur RFID reconnaît correctement les badges autorisés et refuse les non-autorisés.
- ❖ S'assurer que la barrière (ou moteur de simulation) réagit correctement aux autorisations.
- ❖ Tester la communication entre Arduino et le convertisseur TTL/RS485.
- ❖ Valider l'enregistrement des données dans un fichier Excel via PLX-DAQ.
- ❖ Identifier les éventuelles limites du système et proposer des améliorations.

B. Scénarios de test

Test n°	Description du test	Résultat attendu	Statut
1	Présenter un badge autorisé	La barrière s'ouvre, le badge est enregistré dans Excel	Réussi
2	Présenter un badge non autorisé	Accès refusé, aucune action sur la barrière	Réussi
3	Déconnexion du convertisseur RS485	Aucun mouvement moteur, message d'erreur possible	Réussi
4	Présenter plusieurs badges rapidement	Seuls les badges autorisés provoquent une ouverture	Réussi
5	Test d'enregistrement des données (heure, badge, action)	Enregistrement correct dans le tableau Excel	Réussi
6	Simulation moteur avec relais et disque dur (prototype)	Moteur simulé tourne brièvement pour représenter ouverture	Réussi

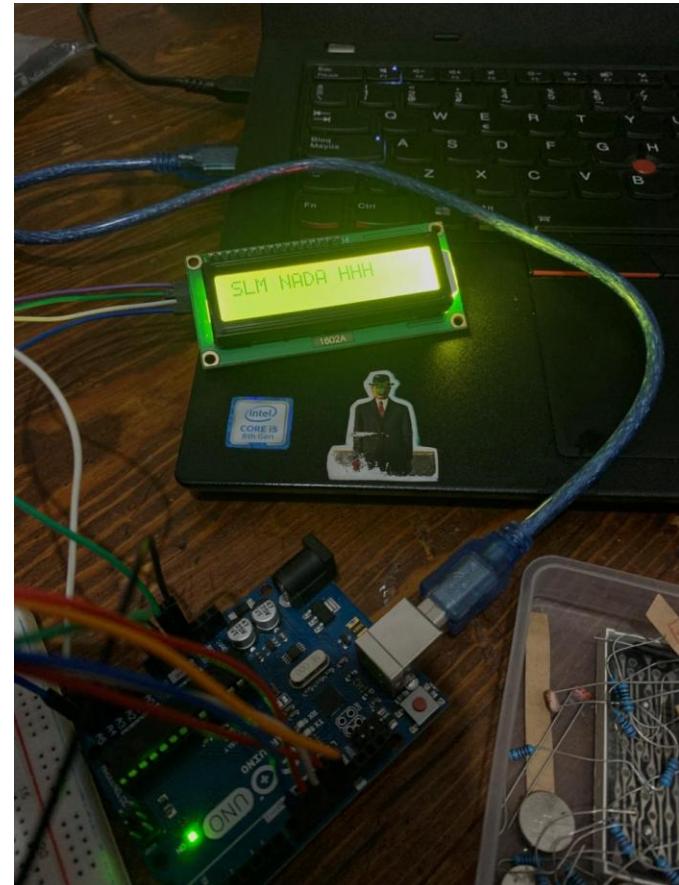
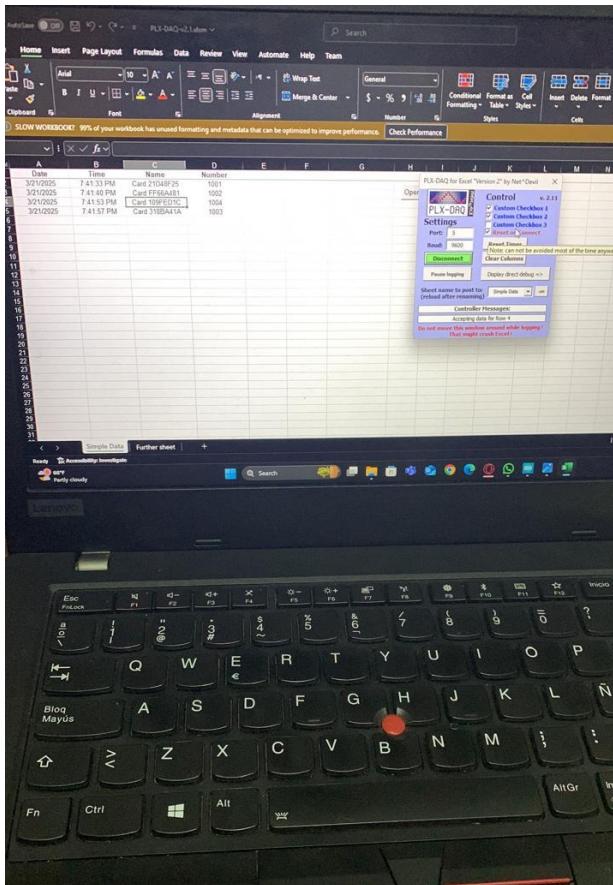


Figure 23 Le système répond correctement aux badges

C. Présentation du prototype utilisé

Afin de rendre le prototype plus accessible pour la soutenance et faciliter les tests en salle, une adaptation a été réalisée. Au lieu d'utiliser le variateur PowerFlex 4 et un moteur industriel, nous avons utilisé une **simulation de moteur basée sur deux relais** et un **petit moteur récupéré d'un lecteur de disque dur**. Cette configuration reproduit les commandes de montée et descente de la barrière, tout en permettant une démonstration pratique et sécurisée.

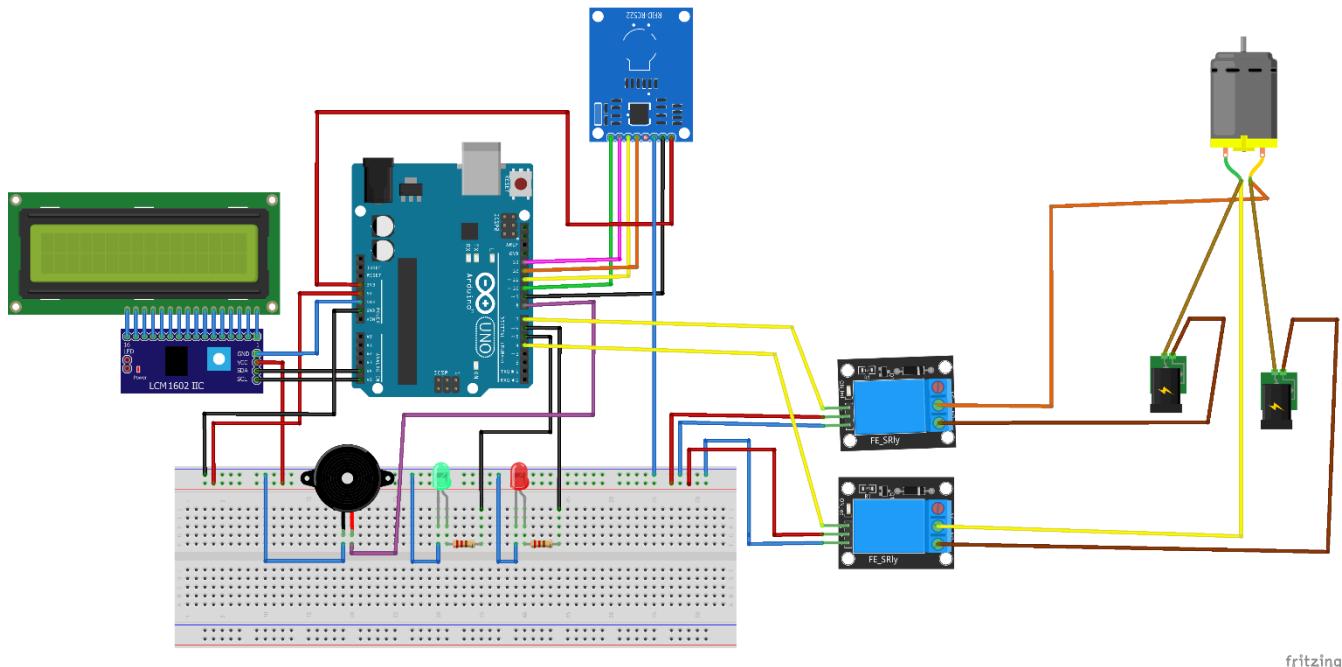


Figure 24 Présentation du prototype utilisé

Les résultats obtenus montrent que :

- Le système répond correctement aux badges autorisés et non autorisés.
- La commande moteur (réelle ou simulée) est déclenchée uniquement si les conditions sont remplies.
- La communication série entre Arduino et le convertisseur fonctionne correctement.
- L'enregistrement sur Excel est fiable et horodaté.

Le système est donc jugé **fiable, fonctionnel et reproductible** dans un cadre réel.

Conclusion Générale

Le projet de fin d'études intitulé « Parking automatisé par RFID et enregistrement sur Excel » a permis la conception, la réalisation et le test d'un système intelligent de gestion d'accès à un parking en utilisant la technologie RFID. Ce système est capable d'identifier automatiquement les véhicules autorisés à l'aide de badges RFID, d'actionner une barrière motorisée, et d'enregistrer les données d'entrée et de sortie dans un fichier Excel, via une communication série entre Arduino et un ordinateur.

Durant ce projet, nous avons développé plusieurs compétences techniques et méthodologiques. Nous avons appris à programmer un microcontrôleur Arduino, à manipuler divers composants électroniques (lecteur RFID, moteur, variateur, convertisseur RS485, écran LCD...), à réaliser un schéma de câblage fonctionnel, et à assurer l'enregistrement automatique des données en utilisant l'outil PLX-DAQ. Nous avons également pratiqué la gestion de projet en binôme, en organisant nos tâches et en respectant les étapes de développement prévues.

Cependant, certaines limites techniques ont été rencontrées. Le système dépend actuellement d'un ordinateur pour l'enregistrement des données, ce qui peut limiter son autonomie. De plus, l'absence d'interface utilisateur conviviale rend l'usage moins accessible en cas d'erreur ou de mauvaise lecture de badge. La gestion des cas comme la perte de badge ou la panne de communication reste à améliorer.

En conclusion, ce projet nous a permis de mettre en pratique nos connaissances théoriques dans un contexte concret et technique, tout en développant notre autonomie, notre rigueur, et notre esprit d'équipe. Il constitue une expérience formatrice en vue d'une future insertion professionnelle dans le domaine de l'électronique et de l'automatisation.

ANNEXE



979-179-p100.pdf | Arduino 1.8.19 (Windows Store 1.8.57.0)

File Edit Sketch Tools Help



979-179-p100.pdf §

```
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// I2C LCD Configuration

LiquidCrystal_I2C lcd(0x27, 16, 2); // Use 0x3F if address doesn't work

// RFID Configuration
#define SS_PIN 10
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);

// Replace with your card UIDs
byte card1[4] = {0x21, 0xD4, 0x8F, 0x25}; // Card 1
byte card2[4] = {0xFF, 0x66, 0xA4, 0x81}; // Card 2
byte card3[4] = {0x31, 0x8B, 0xA4, 0x1A}; // Card 3
byte card4[4] = {0x10, 0x9F, 0xED, 0x1C}; // Card 4

// System Configuration
const int redLed = 6;
const int greenLed = 5;
const int buzzer = 8;
bool registeredCards[4] = {false}; // Tracks registered cards

void setup() {
  Serial.begin(9600);
  SDT.begin();
```

979-179-p100.pdf | Arduino 1.8.19 (Windows Store 1.8.57.0)

File Edit Sketch Tools Help



The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** 979-179-p100.pdf | Arduino 1.8.19 (Windows Store 1.8.57.0)
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Save, Run, Open, Upload, and Download.
- Code Editor:** Displays the C++ code for an Arduino sketch. The code initializes the serial port, SPI, and LCD at 9600 baud. It sets up pin modes for red, green, and buzzer LEDs. It initializes Excel by sending "CLEAR SHEET" and "LABEL,Date,Time,Name,Number" commands via serial. The loop function resets the display every 2 seconds and prints "Scan Your Card" to the LCD.

```
void setup() {  
    Serial.begin(9600);  
    SPI.begin();  
    mfrc522.PCD_Init();  
  
    // LCD Initialization  
    lcd.init();  
    lcd.backlight();  
    lcd.print("Attendance System");  
    delay(2000);  
    lcd.clear();  
    lcd.print("Scan Your Card");  
  
    // Pin Modes  
    pinMode(redLed, OUTPUT);  
    pinMode(greenLed, OUTPUT);  
    pinMode(buzzer, OUTPUT);  
  
    // Initialize Excel  
    Serial.println("CLEAR SHEET");  
    Serial.println("LABEL,Date,Time,Name,Number");  
}  
  
void loop() {  
    // Reset display every 2 seconds  
    static unsigned long lastScan = 0;  
    if(millis() - lastScan > 2000) {  
        lcd.clear();  
        lcd.print("Scan Your Card");  
        lastScan = millis();  
    }  
}
```

979-179-p100.pdf | Arduino 1.8.19 (Windows Store 1.8.57.0)

File Edit Sketch Tools Help



979-179-p100.pdf §

```
lcd.setCursor(0, 1);
lcd.print(names[index]);

Serial.print("DATA,DATE,TIME, ");
Serial.print(names[index]);
Serial.print(",");
Serial.println(numbers[index]);

digitalWrite(greenLed, HIGH);
tone(buzzer, 2000, 200);
} else {
// Already registered
lcd.print("Already Logged!");
digitalWrite(redLed, HIGH);
tone(buzzer, 1000, 500);
}

delay(2000);
digitalWrite(greenLed, LOW);
digitalWrite(redLed, LOW);
}

void processInvalidCard() {
lcd.clear();
lcd.print("Invalid Card!");
digitalWrite(redLed, HIGH);
tone(buzzer, 1000, 1000);
delay(2000);
digitalWrite(redLed, LOW);
}
```