الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire

وزارة التعليم العالي والبحث العلمي
جامعة أكلي محند أولحاج
-البويرة-

Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université Akli Mohand Oulhadj - Bouira -
Tasdawit Akli Muḥend Ulḥağ - Tubirett -

جامعة البويرة

كلية العلوم والعلوم التطبيقية
Faculté des Sciences et des Sciences Appliquées

المرجع:...../م م/ 2021
Référence : ...../MM/2021

Mémoire de Master

Présenté au

Département : Génie Électrique
Domaine : Sciences et Technologies
Filière : Electronique
Spécialité : Electronique des systèmes embarqués

Réalisé par :

Benabbas Abir

Et

Oukaci Kahina

Thème

Réalisation d'un supermarché avec tri et rangement des produits

Soutenu le: **30/11/2021**

Devant la commission composée de :

Mr Bouhedda Ali	M.A.A	Univ. Bouira	Président
Mr Benziane Mourad	M.A.A	Univ. Bouira	Rapporteur
Mr Bouzida Ahcene	M.C.A	Univ. Bouira	Examinateur

Année Universitaire: 2020-2021

Remerciements

Ce travail a été effectué au sein de la faculté des Sciences et sciences appliquées de l'Université de Bouira, département de génie électrique.

On tiens à remercier, en premier lieu, Dr. Benziane Mourad Directeur de ce mémoire pour sa disponibilité, son soutient et ses précieux conseils tout au long de l'élaboration de ce mémoire. On remercie également tous les membres du jury pour l'intérêt qu'ils ont porté à notre travail.

On adresse nos plus sincères remerciements à nos très chers familles et nos amis, qui ont toujours été là pour nous, leur soutien inconditionnel et leurs encouragements ont été d'une grande aide.

Enfin, on associe à ces remerciements tous ceux qui ont contribué à réaliser ce travail.

Abréviations

AREF Analog Reference
CAN Convertisseur Analogique-Numérique
CAO Conception Assisté par Ordinateur
DC Direct Current
E/S Entrées/sorties
EEPROM Electronically Erasable Programmable Read-Only Memory
IDE Integrated Development Environement
IR Infra-Rouge
LED Light Emitting Diode
PWM Pulse Width Modulation
TX/RX Transmission/Réception
USB Universal Serial Bus

Sommaire

Introduction générale **3**

1 Généralités sur les robots **5**
1 Introduction 5
2 Généralité sur les robots 5
2.1 Historique d'un robot 5
2.2 Définition d'un robot 6
2.3 Les composants d'un robot 6
2.4 Les différents types de robot 8
3 Robot mobile 8
3.1 Classification des robots mobiles 9
4 Domaine d'application de la robotique 10
5 les avantages et les inconvénients du robot 12
6 Conclusion 12

2 Description matérielle et logicielle **13**
1 Introduction 13
2 La partie matérielle 13
2.1 La carte Arduino 13
2.2 Le moteur à courant continue 17
2.3 Driver L298N 18
2.4 Moteur NEMA17 20
2.5 Driver A4988 21
2.6 Châssis de type voiture 22
2.7 Capteur infrarouge 23
2.8 Émetteur - Récepteur à ultrasons HC-SR04 24
2.9 Le module ESP32 DEVKIT V1 26
3 La partie logicielle 28

3.1 Logiciel IDE 28
3.2 Logiciel de conception de circuits Fritzing 31
3.3 Logiciel de simulation Proteus (isis) 32
3.4 App Inventor 33
3.5 Firebase Realtime 37
4 Conclusion 38

3 Réalisation pratique du projet 39
1 Introduction 39
2 Présentation synoptique du projet 39
3 Présentation de l'application Androïde réalisée 41
4 Montage de la réalisation sous Fritzing 46
4.1 Pour le robot suiveur de ligne arrangeur d'objet 46
4.2 Pour le système de contrôle des étagères 47
5 La simulation du robot suiveur de ligne sur Proteus 48
6 Résultats et principe de fonctionnement 49
7 Conclusion 55

Conclusion générale 56

Bibliographie 57

Annexe 59

Table des figures

1.1 Exemple d'un contrôleur 6
1.2 Exemple d'un capteur qui mesure la distance 7
1.3 Exemple d'un manipulateur 7
1.4 Exemple d'un actionneur 7
1.5 Exemple d'un effecteur 8
1.6 Robot Mobile à patte 9
1.7 Robot de type voiture [6]. 10
1.8 Exemple d'un Robot militaire 11
1.9 Exemple d'un Robot pour utilisation civil 11
1.10 Exemple d'un Robot pour l'usage domestique 12

2.1 Principe de fonctionnement d'une carte Arduino [9] 14
2.2 Quelques types de carte Arduino [10] 15
2.3 Description de la carte Arduino UNO [11] 15
2.4 Moteur à courant continue 18
2.5 Driver L298N 18
2.6 Schéma interne d'un driver L298N [14] 19
2.7 Le pont en H [15] 20
2.8 Moteur NEMA17 20
2.9 Driver A4988 21
2.10 Branchement d'un driver A4988 [19] 21
2.11 Schéma interne d'un driver A4988 [18] 22
2.12 Châssis à 4 roues motrices 22
2.13 Capteur infrarouge 23
2.14 Schéma interne d'un capteur infra-rouge [20] 23
2.15 Le spectre de longueur d'ondes 24
2.16 Principe du capteur infrarouge [20] 24
2.17 Ultrason HC-SR04 25

2.18 Schéma interne d'un ultrasons HC-SR04 [21] 25
2.19 Principe de mesure la distance d'un ultrason 26
2.20 Le module ESP32 DEVKIT V1 27
2.21 Pin out de ESP32 [22] 27
2.22 Interface du logiciel IDE. 29
2.23 Les boutons du logiciel IDE. 30
2.24 Structure d'un programme Arduino 31
2.25 Page d'accueil du logiciel fritzing 32
2.26 Présentation de l'interface 33
2.27 Le concept d'App Inventor [27] 34
2.28 Interface graphique de App Inventor 35
2.29 Interface des blocs d'AppInventor 36
2.30 Base de données Firebase Realtime 38

3.1 Schéma en bloc côté robot 40
3.2 Schéma en bloc côté étagères 40
3.3 Interface graphique écran 1 41
3.4 Interface de bloc de l'écran 1 42
3.5 Interface graphique de l'écran 2 42
3.6 Interface de bloc de l'écran 2 43
3.7 Interface graphique de l'écran 3 44
3.8 Interface de bloc de l'écran 3 45
3.9 Base de données utilisée 46
3.10 Schéma du robot sous Fritzing 47
3.11 Schéma de système de contrôle des étagères sous Fritzing . . . 48
3.12 Simulation de suiveur de ligne sur Proteus 49
3.13 Mécanisme de poussoir à vérin 50
3.14 Cas où l'étagère S1 pas vide et l'étagère S2 vide 51
3.15 Robot suiveur de ligne/porteur d'objets 51
3.16 Robot devant les étagères 52
3.17 Les étagères 53
3.18 Envoi du message1 à Firebase 53
3.19 Dépôt d'objet sur l'étagère S1 54
3.20 Dépôt d'objet sur l'étagère S2 54
3.21 Schéma interne de la carte UNO [30] 67

Introduction générale

Le concept de robot date de plusieurs siècles, mais le terme robot fut inventé par le tchèque Karel Capek dans une pièce de théâtre écrite en 1920 : "RUR ou les robots universels de Rossum". Ce terme est dérivé du verbe tchèque "robota" signifiant travail forcé ou corvée [1].

Depuis l'apparition des premiers robots industriels, leur essor a été considérable, et chaque année de nouveaux secteurs industriels s'ouvrent à la robotisation au fur et à mesure que les équipements se diversifient, deviennent de plus en adaptables aux taches industrielle et moins couteux. Les robots sont devenus tellement indispensables dans plusieurs secteurs industriels (l'automobile par exemple) au point que la survie économique de ses entreprises est devenue fortement liée aux robots. La robotique a permis à l'humanité d'accomplir des tâches difficiles, répétitives, pénibles et des fois même impossible. Prenant l'exemple du robot nommé Romeo [2], ou l'assistant du futur comme il est appelé. Il s'agit d'un robot humanoïde qui a été conçu pour aider les personnes âgées dépendantes chez elles. Il peut marcher, entendre, parler, et voir en trois dimensions.

Les robots peuvent être fixes ou mobiles selon l'application, les robots fixes sont généralement utilisés dans les usines, les hôpitaux...etc. Cependant les robots mobiles sont généralement utilisés dans les environnements qui nécessitent des mouvements.

Avec la diversité des taches et des travaux que les gens doivent faire quotidiennement de façon répétitive, ils peuvent nécessiter des grandes efforts ou un temps limité. Donc ces taches risquent de ne pas être bien achevé dans les délais.

Le rangement ou le triage des objets de manière répétitive dans leur places, fait partie de la catégorie des taches mentionnées ci-dessus, des fois les objets peuvent être lourds ou fragiles et il doivent être porter sur de long distance.

Pour mettre fin à cette problématique, un robot dédié à cette tache peut assurer l'accomplissement du rangement des objets dans leur étagères spécifiques.

Ce projet consiste à étudier et réaliser un robot prototype qui porte des objets et les ranger automatiquement dans leur places, tout en suivant une ligne noire. Ce travail est constitué de trois chapitres :

Le premier chapitre présente des généralités sur les robots. Le deuxième chapitre est consacrée à la présentation des éléments matériels et logiciels qui ont permis de réaliser ce projet. Le troisième chapitre décrit la commande et la réalisation pratique de robot arrangeur d'objet où il regroupe toutes les parties pour avoir un fonctionnement rigoureux.

Chapitre 1

Généralités sur les robots

1 Introduction

Les robots aujourd'hui ont un impact considérable sur de nombreux aspects de la vie moderne surtout dans la fabrication industrielle. Ce chapitre consiste à présenter des généralités sur les robots, leur historique, leurs composants et les différents type de robots, notamment les robots mobiles de type voiture.

2 Généralité sur les robots

2.1 Historique d'un robot

Le terme robot apparaît pour la première fois dans la pièce de théâtre (science-fiction) R. U. R (Rossum's Universal Robots), écrite en 1920 par l'auteur Karel Čapek. Le mot a été créé par son frère Josef à partir du mot tchèque « robota » qui signifie « travail, besogne, corvée ». Le premier robot industriel créé est 'unimate'. Il fallut attendre 1959 pour que Georges Devol invente une machine originale, polyvalente reprogrammable, ce qui a permis au robot d'acquérir une réalité industrielle. Mais en fait ce ne fut que vers la fin des années 1970 que les robots industriels de première génération ont vu le jour [1]. En 1970, le robot lunaire Lunokhod, envoyé par l'union soviétique, a voyagé sur une distance de 10 km et a transmis plus de 20 000 images.

2.2 Définition d'un robot

Un robot est un dispositif articulé mécatronique (alliant mécanique, électronique et informatique) accomplissant automatiquement des tâches qui sont généralement dangereuses, pénibles, répétitives ou impossibles pour les humains, capable d'imiter certaines fonctions humaines telles que la manipulation d'objets ou locomotion, dans le but de se substituer à l'homme pour la réalisation des certaines tâches matérielles, cette réalisation est plus ou moins autonome selon les facultés de perception de l'environnement dont est doté le robot.

Un robot peut être mobile et utiliser différents moyens de locomotion, il peut être anthropoïde s'il imite la forme humaine [3]. Les robots les plus évolués sont capables de se déplacer et de se recharger par eux-mêmes.

2.3 Les composants d'un robot

La plupart des robots contiennent les éléments suivants [4] :

1. Système de contrôle : tel qu'une carte contrôleur.

FIGURE 1.1 – Exemple d'un contrôleur

2. Capteurs : ils peuvent lire des informations sur l'environnement (capteurs extéroceptifs) ou le robot lui-même (capteurs proprioceptifs). Le robot utilise ses capteur pour prendre connaissance de son environnement.

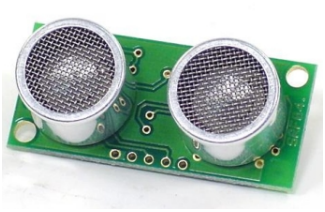

FIGURE 1.2 – Exemple d'un capteur qui mesure la distance

3. Manipulateur : un bras manipulateur est le bras d'un robot généralement programmable, avec des fonctions similaires à un bras humain, comprend les jonctions, les articulations et d'autres éléments de structure du robot.

FIGURE 1.3 – Exemple d'un manipulateur

4. Actionneurs : ils produisent un effet sur l'environnement du robot, qui permettent de transformer les donnés reçues des capteurs en un phénomène physique (déplacement, dégagement de chaleur...).

FIGURE 1.4 – Exemple d'un actionneur

5. Effecteur finale : est reliée à la dernière jonction d'un manipulateur qui

gère généralement les objets, établit des connexions à d'autres machines.

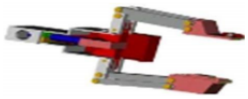

FIGURE 1.5 – Exemple d'un effecteur

2.4 Les différents types de robot

Il existe plusieurs de types, les plus importants sont :

1. Robots industriels : les robots industriels représentent la plus importante population robotique. Ils sont surtout retrouvés sur les chaînes de montage de l'industrie automobile.

2. Robots manipulateurs : robots ancrés physiquement à leur place de travail et généralement mis en place pour réaliser une tâche précise ou répétitive.

3. Robots mobiles : robots capables de se déplacer dans un environnement. Ils sont équipés ou non de manipulateurs suivant leur utilisation.

3 Robot mobile

A présent, la plupart des travaux de recherche portent sur les problèmes de perception, la planification de trajectoires, l'analyse et la modélisation de l'environnement de robot, appliqués sur des robots mobiles commerciaux.

La robotique mobile, c'est le système le plus étudiée, car c'est un système plus simple à réaliser par rapport les autres types de robots, ce qui permet d'en venir plus rapidement à l'étude de leur navigation. C'est un système mécanique, électronique et informatique agissant physiquement sur son environnement en vue d'atteindre l'objectif qui lui a été assigné. Ce type de robot est doté de moyens de locomotion qui lui permettent de se déplacer suivant son degré d'autonomie, il peut être doté de moyens de perception et de raisonnement [5].

3.1 Classification des robots mobiles

Robot mobile à pattes

Dans la situation où le terrain est encore plus incertain, avec des grandes différences de hauteur comme par exemple un escalier ou un terrain très accidenté, il fait recours aux robots mobiles à pattes. Ils ont des points d'appui discrets sur le terrain et sont donc la solution à ce problème de mouvement. Par contre, la conception et le contrôle d'un engin à pattes sont très complexes. En plus, la vitesse d'évolution est généralement très réduite. La commande est très difficile, dépend de la multiplicité des actionneurs utilisés.

Aibo de Sony est un exemple d'un robot mobile à pattes.

FIGURE 1.6 – Robot Mobile à patte

Robot mobile utilisant la chenille

Lorsque le terrain est accidenté, les robots à roues perdent leur efficacité de locomotion. Dans ces conditions, les chenilles sont plus intéressantes car elles permettent d'augmenter l'adhérence au sol et de franchir des obstacles plus importants.

Robot mobile à roues

Compte tenu de la simplicité du mécanisme de locomotion utilisé, ce type de robot est le plus répondu actuellement. La grande majorité des robots de ce type présente des contraintes de non holonomie qui limitent le mouvement instantané que le robot peut réaliser, car il existe pour toutes ces roues un point unique (centre instantané de rotation (CIR)) de vitesse nulle autour duquel le robot tourne de façon instantanée. La commande se fait par la motorisation des roues installées. Ces contraintes augmentent la complexité

du problème de planification de trajectoire et son contrôle.

Parmi les différents types des robots mobiles à roues, le robot prototype réalisé dans ce projet fait partie du type robot mobile à roues de type voiture.

FIGURE 1.7 – Robot de type voiture [6].

4 Domaine d'application de la robotique

La robotique actuelle est utilisée dans des différents domaines [7] :

1. L'industrie : le premier but des robots est de remplacer l'homme dans des activités difficiles et fatigantes pour l'employeur.

2. Le domaine militaire : les robots sont de plus en plus utilisés dans le domaine militaire. Aujourd'hui on peut facilement créer des robots discrets et dotés de nombreux capteurs, ce qui est idéal pour des missions d'espionnage ou d'éclairement. Le robot Mule de l'armée américaine est initialement conçu pour transporter les paquetages des soldats américains.

Ce véhicule autonome est également doté de radars, sonars, d'une caméra infrarouge.

FIGURE 1.8 – Exemple d'un Robot militaire

3. La santé : les robots dans ce domaine ne sont pas complètement autonomes mais ils assistent les médecins ou chirurgiens, jusqu'à permettre des opérations médicales à distance. Cette pratique de chirurgie assistée est émergente donc bien que peu répandue, elle est en phase de devenir la chirurgie du futur.

4. Utilisation civile : de plus en plus de tâches sont confiées aux robots. Ils servent à remplacer les personnes qui sont chargées de taches civiles (nettoyer la ville, aider la population, s'occuper des lieux publiques...).

FIGURE 1.9 – Exemple d'un Robot pour utilisation civil

Ce robot a été crée par le japonais Yukitaro. Il est équipé de deux caméras et d'un GPS. Il circule dans la rue de manière autonome en engouffrant la neige avec sa « bouche ». Puis il la rejette sous forme de blocs de glace.

5. L'usage domestique : comme le robot aspirateur, il fonctionne d'une manière autonome sur tous types de sol. Grâce à ses capteurs, il évite les escaliers et tout type d'obstacles. Il retourne à sa base pour se recharger sans assistance. Il est possible de délimiter une surface précise à nettoyer avec les accessoires fournis, telle qu'une application à reseaux WiFi.

FIGURE 1.10 – Exemple d'un Robot pour l'usage domestique

5 les avantages et les inconvénients du robot

Les avantages :

- Augmenter la sécurité.
- La capacité de production accélérée.
- La souplesse d'utilisation.
- La simplification des taches de production et la vie courante.
- Peuvent travailler dans un environnement dangereux.
- Travaillent continuellement, sans ressentir une fatigue ou l'ennui.

Les inconvénients :

- La suppression d'emploi.
- La complexité de la maintenance, elle doit être structurée.
- Les limites dans le pouvoir de prendre une décision.
- Consommation d'énergie.

6 Conclusion

Les définitions des concepts mises aux claires dans ce chapitre permettent de comprendre la robotique d'une façon générale, en citant d'une manière globale les diverses techniques et technologies utilisées dans les robots, ce qui donne une motivation pour se lancer dans ce projet de réalisation.

Chapitre 2

Description matérielle et logicielle

1 Introduction

L'objectif principal de l'évolution de l'électronique est de créer des systèmes de plus en plus puissants, miniaturisés et autonomes, tout en minimisant le coût d'étude et de production. Actuellement, les systèmes ne sont plus exclusivement électroniques mais impliquent la mise en œuvre d'autres technologies dans un même boitier, comme la mécanique, l'optique, la chimie...etc

Ce chapitre consiste à présenter deux parties, la partie matérielle (les différent composants électroniques utilisés dans le robot arrangeur d'objet) et la partie logicielle (logiciels de programmation, et d'application Androïde), qui vont être adaptés et assemblés pour le bon fonctionnement du robot.

2 La partie matérielle

2.1 La carte Arduino

Historique de la carte

Au début des années 2000, les outils de conception de projets dans le domaine de l'ingénierie et de la robotique étaient onéreux, proches d'une centaine d'euros. Maîtriser et utiliser ces outils demandait beaucoup d'argent et du temps d'apprentissage qui ralentissaient fortement le processus de création des projets.

Arduino est à l'origine un projet d'étudiants de l'école de Design d'interaction d'Ivrea en Italie. Arduino à été fondé en 2005 par un groupe d'étudiants Italiens : Massimo Banzi, David Cuartielles, Tom Lgoe, David Mellis et Nicholas Zambetti [8]. Leur but était de rendre la création et la conception des systèmes électroniques soient accessible et facile à tous.

Présentation de la carte

Une carte Arduino est une petite carte électronique équipée d'un microcontrôleur qui permet, à partir d'événements détectés par des capteurs, de programmer et commander des actionneurs.

FIGURE 2.1 – Principe de fonctionnement d'une carte Arduino [9]

Elle permet de connecter un bon nombre de composants (LEDs, moteurs, capteurs...etc) sur des broches appelées "PIN", et de les faire fonctionner de manière autonome en la programmant via le logiciel Arduino IDE ou d'autres comme Scratch, Arduiblock...etc. Cette carte est idéal pour réaliser des projets de domotiques, de robotiques ou d'informatique embarquée. C'est une carte de prototypage, car elle permet de créer des prototypes assez simplement.

Les types de carte Arduino

Il existe plusieurs types de carte Arduino, voici quelque uns :

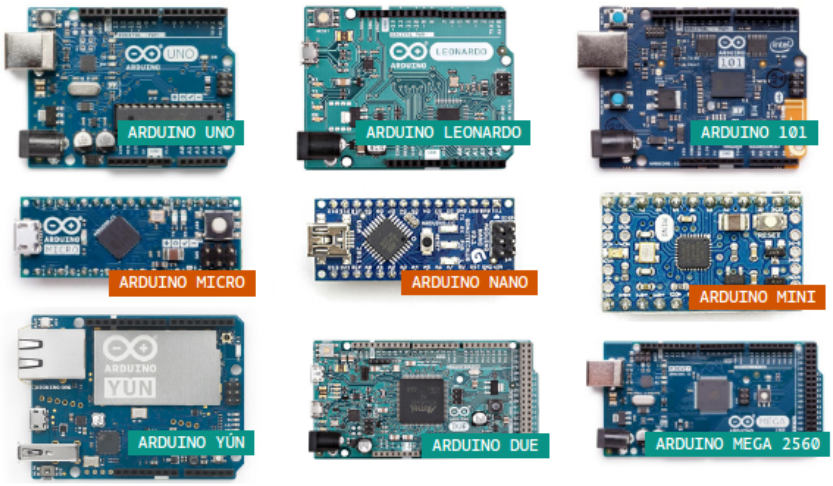

FIGURE 2.2 – Quelques types de carte Arduino [10]

Une carte Arduino UNO (carte basique) est utilisée pour réaliser et faire fonctionner le robot arrangeur d'objets.

Description de la carte Arduino UNO

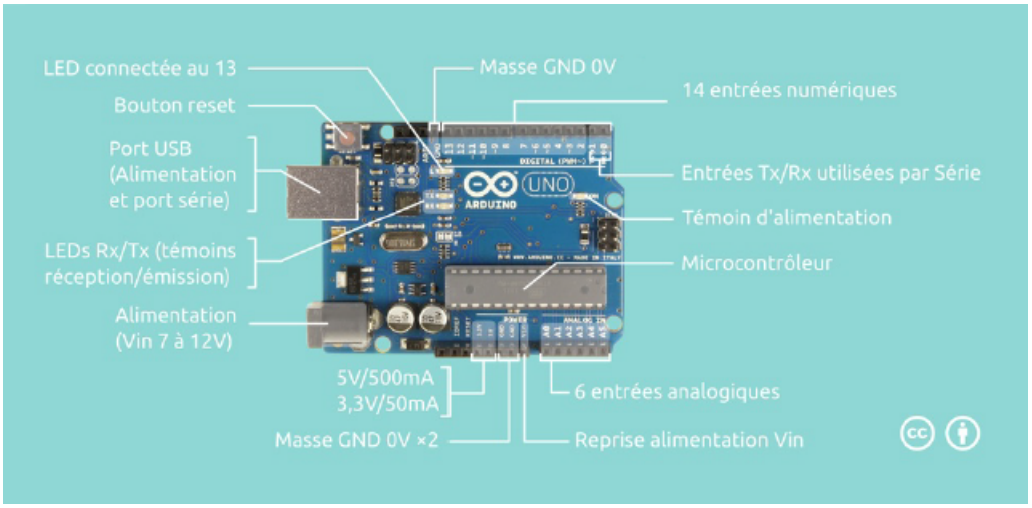

FIGURE 2.3 – Description de la carte Arduino UNO [11]

La carte Arduino UNO est sans doute la carte la plus populaire. Elle est

équipée d'un processeur (MEGA328P) fonctionnant à 16 MHz, comprend 32 Ko de mémoire programme (mémoire Flash), 1 Ko d'EEPROM, 2 Ko de RAM, 14 E/S numériques, 6 entrées analogiques et un rail d'alimentation de 5V et 3,3V. Les connexions numériques peuvent émettre ou recevoir une valeur numérique binaire 0 ou 1, et les connexions analogiques peuvent émettre ou recevoir une valeur comprend de 0 à 5 Volts. Les éléments essentiels de cette carte sont expliquées ci-dessous [12] :

1. Alimentation par USB : la carte Arduino peut être alimentée avec un câble USB relié à un ordinateur.

2. Alimentation via connecteur Jack DC : la carte Arduino peut être directement alimentée par un connecteur Jack DC. L'alimentation via ce connecteur doit être comprise entre 7v et 12 V.

3. Régulateur de tension : la fonction du régulateur de tension est de contrôler la tension d'alimentation de l'Arduino pour la stabiliser à la bonne tension du micro-contrôleur et de chaque éléments de la carte.

4. Reset : cette broche est utilisée pour redémarrer la carte Arduino. Cela aura pour effet de redémarrer le programme depuis le début.

5. Broches analogiques : Arduino UNO possède 6 broches d'entrées analogiques numérotée de A0 jusqu'à A5. Ces broches permettent de lire un signal analogique d'un capteur comme un capteur d'humidité ou de température. La carte Arduino utilise un convertisseur analogique/numérique (convertisseur CAN) pour permettre la lecture du signal par le micro-contrôleur. Un signal sera converti sur 10 bits. La valeur pourra être lue sur une échelle 1024 points.

6. Indicateur LED d'alimentation : ce voyant doit s'allumer lorsque on branche Arduino sur une source d'alimentation.

7. LEDs TX et RX : ils apparaissent à deux endroits sur la carte Arduino UNO. Tout d'abord, sur les broches numériques 0 et 1, pour indiquer les broches responsables de la communication série. Deuxièmement, les LEDs TX et RX.

• Le voyant TX : clignote à une vitesse variable lors de l'envoi des données série. La vitesse de clignotement dépend de la vitesse de transmission utilisée

par la carte.

• Le voyant RX : clignote pendant le processus de réception. La vitesse de transmission s'exprime en bauds, soit l'équivalent du bits/seconde si le signal est binaire.

8. Entrées/Sorties numériques : la carte Arduino UNO possède 14 broches d'entrées/sorties numériques, dont 6 peuvent fournir une sortie PWM. Ces broches peuvent être configurées pour fonctionner comme des broches numériques d'entrée pour lire des valeurs logiques (0 ou 1) ou numériques. Les broches étiquetées "∼" peuvent être utilisées pour générer des PWM.

9. Broche AREF : AREF est l'acronyme anglais de référence analogique. Cette broche est parfois utilisée pour définir une tension de référence externe (entre 0 et 5 Volts) comme limite supérieure pour les broches d'entrée analogiques.

10. Broches (3.3, 5, GND, Vin) :
• 3.3V : broche d'alimentation de tension 3.3 Volts.
• 5V : broche d'alimentation de tension 5 Volts La plupart des composants destinés à fonctionner avec Arduino fonctionnent bien en 3.3 Volts ou 5 Volts.
• GND (Ground / Masse) : Il y a trois broches de ce type présentées sur la carte Arduino, elles sont toutes communes et peuvent être utilisées comme masse (potentiel 0 Volts).
• Vin : Cette broche permet d'alimenter l'Arduino depuis une source de tension extérieure. Elle est reliée au circuit d'alimentation principale de la carte Arduino.

2.2 Le moteur à courant continue

C'est un actionneur qui permet à partir d'un courant électrique continu de faire tourner un mécanisme. Un moteur à courant continu fait partie d'une classe de moteurs électriques rotatifs qui convertit l'énergie électrique à courant continu en énergie mécanique. Les types les plus courants dépendent des forces produites par les champs magnétiques.

Dans ce projet, ce type de moteur est utilisé pour faire rouler les roues du châssis de type voiture.

FIGURE 2.4 – Moteur à courant continue

Remarque

Pour inverser le sens d'un moteur à courant continue, il faut inverser les fils positive et négative de l'alimentation du moteur. Pour éviter ça, il faut utiliser un composant électrique avec une structure électronique qui s'appelle "pont en H". Le driver L298N est choisi pour effectuer cette tache.

2.3 Driver L298N

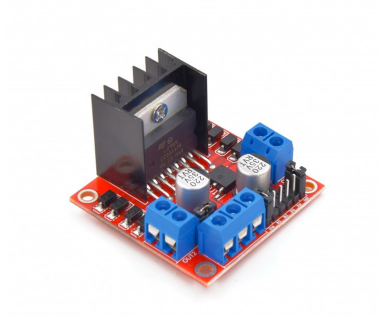

FIGURE 2.5 – Driver L298N

La puce L298N est un double Pont-H, elle permet de faire tourner les moteurs dans un sens ou dans l'autre sens sans avoir à modifier le branchement.

Cette puce est un module extrêmement utile pour contrôler des robots et ensembles mécanisés. L298N peut piloter des charges inductives comme des

relais, solénoïdes, moteurs continus et moteurs pas-à-pas. Les deux types de moteurs peuvent être contrôlés aussi bien en vitesse (PWM) qu'en direction. Toutes les sorties en puissance sont déjà protégées par des diodes anti-retour [13].

Dans ce projet, il est utilisé pour contrôler quatre moteurs à courant continu, chaque deux moteurs sont reliés ensemble en parallèle.

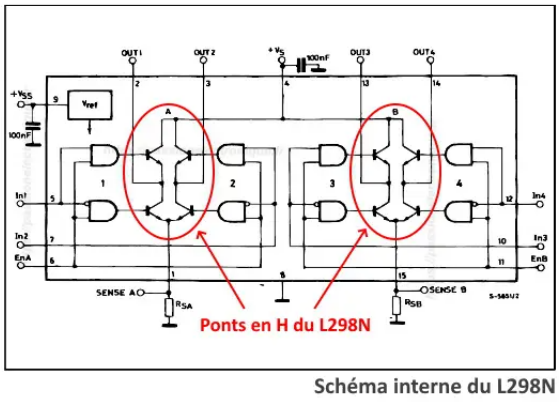

FIGURE 2.6 – Schéma interne d'un driver L298N [14]

Principe de fonctionnement d'un pont en H :

Un circuit pont en H contient quatre commutateurs qui sont généralement des transistors de puissance avec le moteur au centre formant un arrangement de type H.

La fermeture simultanée de deux interrupteurs particuliers inverse la polarité de la tension appliquée au moteur. Cela provoque un changement dans le sens de rotation du moteur [13].

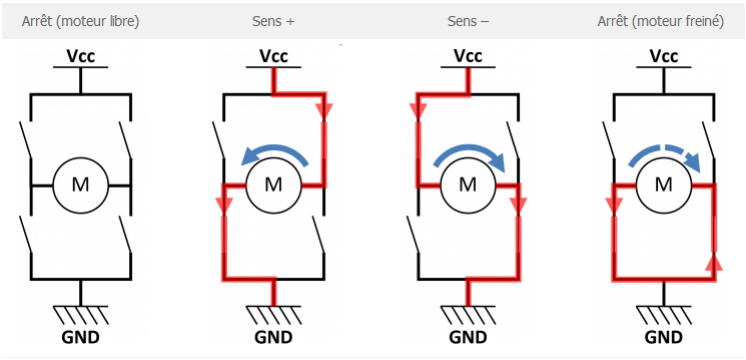

FIGURE 2.7 – Le pont en H [15]

2.4 Moteur NEMA17

Le moteur NEMA17 est un moteur pas à pas à quatre fils : Rouge, Bleu, Vert, et Noir. Il y a 4 demi-bobines, câblées en parallèles 2 à 2 pour former au final 2 bobines : 2 fils par bobine, les fils "rouge" et "bleu" étant les extrémités d'une bobine, les fils "noir" et "vert" les extrémités de l'autre bobine. En cas de doutes les extrémités des bobines peuvent être repérées par un multimètre en mode "continuité " ou " testeur". Ce moteur pas à pas bipolaire hybride a un angle de pas de 1.8°, ce qui fait 200 pas/tour [16].

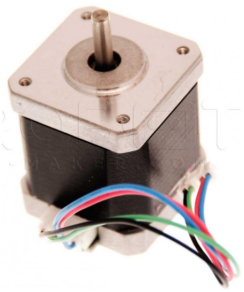

FIGURE 2.8 – Moteur NEMA17

Afin de contrôler le sens de rotation, la vitesse et le nombre de pas à faire, un driver de référence A4988 est utilisé.

2.5 Driver A4988

C'est une carte de dérivation pour le pilote des moteurs pas à pas bipolaire, comprend une limitation de courant réglable, une protection contre les surintensités et les surchauffes, et cinq résolutions de micro-pas différentes (jusqu'à 1/16 pas). Il fonctionne de 8 V à 35 V et peut fournir jusqu'à environ (1 ampère) par phase sans dissipateur thermique ni flux d'air forcé (il est calibré pour (2 ampère) par bobine avec un refroidissement supplémentaire suffisant) [17].

Il peut contrôler à la fois la vitesse et la direction de rotation d'un moteur pas à pas bipolaire comme NEMA17, avec seulement deux broches (une broche sert à réguler la direction des révolutions du moteur et l'autre à la régulation des pas du moteur).

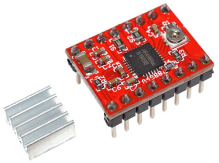

FIGURE 2.9 – Driver A4988

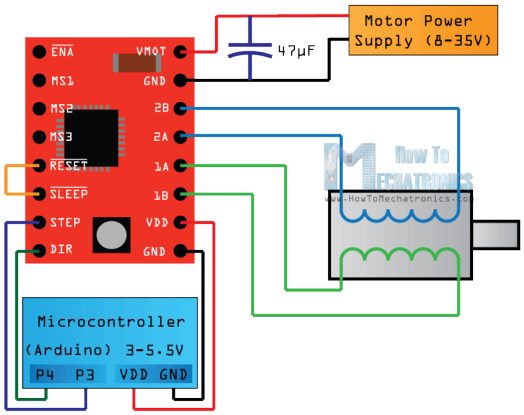

FIGURE 2.10 – Branchement d'un driver A4988 [19]

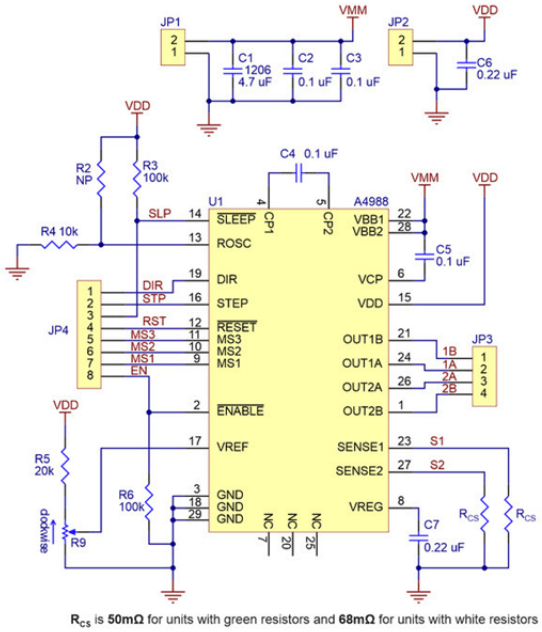

FIGURE 2.11 – Schéma interne d'un driver A4988 [18]

2.6 Châssis de type voiture

C'est un châssis voiture à 4 roues motrices, chaque roue est contrôlée par le moteur à courant continu mentionné précédemment.

FIGURE 2.12 – Châssis à 4 roues motrices

2.7 Capteur infrarouge

Le capteur infrarouge se compose essentiellement d'une LED IR et d'une photodiode, cette paire est généralement appelée paire IR ou photocoupleur. Le capteur IR fonctionne sur le principe dans lequel la LED IR émet un rayonnement infrarouge et la photodiode détecte ce dernier, en cas le rayonnement émis tombe à un obstacle. Le capteur IR est un capteur très populaire, qui est utilisé dans de nombreuses applications en électronique, il est utilisé dans les systèmes de télécommande, les détecteurs de mouvement et d'obstacle, les robots suiveurs de ligne, les alarmes, etc. [20].

FIGURE 2.13 – Capteur infrarouge

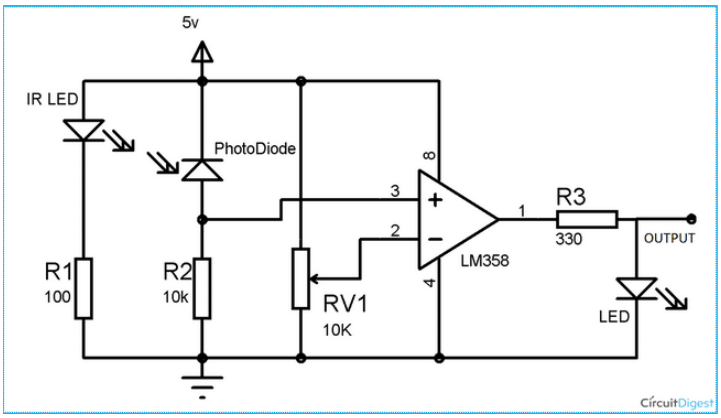

FIGURE 2.14 – Schéma interne d'un capteur infra-rouge [20]

Le rayonnement émis par ce module un rayonnement électromagnétique qui se situ dans une région spectrale invisible par les humain, car la fréquence

de ce type de rayonnement est inférieure à la fréquence du rouge visible où il vient le nom infrarouge.

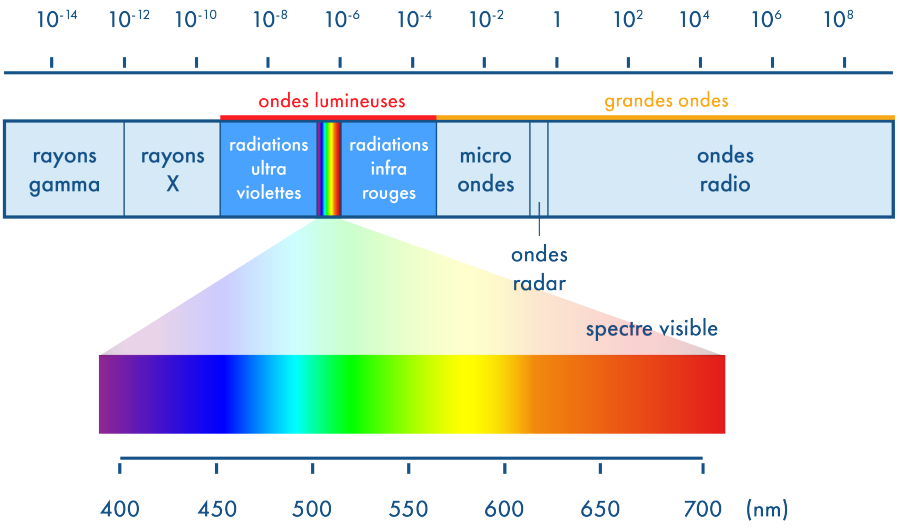

FIGURE 2.15 – Le spectre de longueur d'ondes

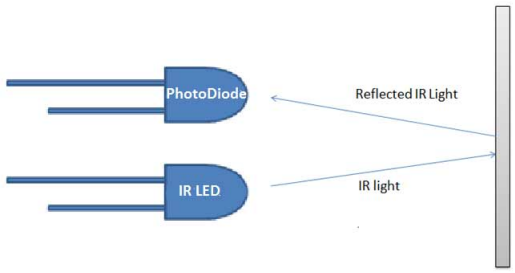

FIGURE 2.16 – Principe du capteur infrarouge [20]

2.8 Émetteur - Récepteur à ultrasons HC-SR04

Les capteurs HC-SR04 sont des émetteurs récepteurs d'ultrasons (40KHz) qui permettent de mesurer la distance entre le capteur et le premier élément

situé sur sa trajectoire, qui lui renvoie les ultrasons.

FIGURE 2.17 – Ultrason HC-SR04

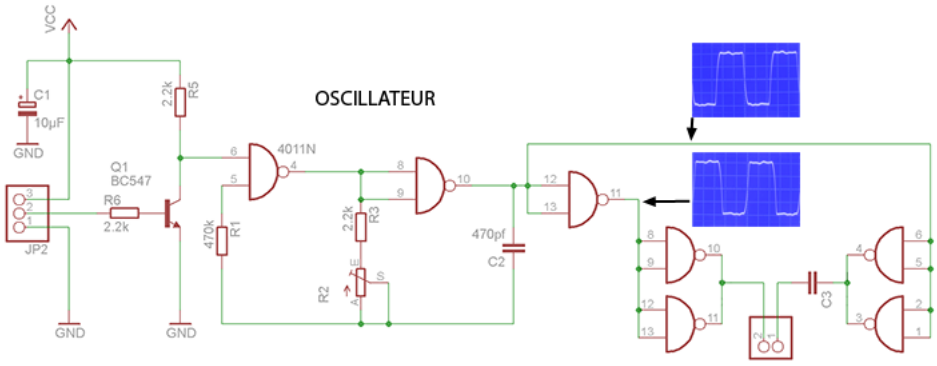

FIGURE 2.18 – Schéma interne d'un ultrasons HC-SR04 [21]

Le principe de mesure de distance avec ce type de capteur repose sur le temps mis par une onde sonore émise pour aller vers un objet, rebondir puis revenir vers le récepteur. Le capteur fournit une sortie de durée proportionnelle à la distance. Ce temps (t) est récupéré par un module de contrôle et en déduire une distance en multipliant par la vitesse du son (340 m/s) et en divisant par 2 (le son faisant un aller/retour) soit : d = ((340 x t) / 2) mètres.

Si t=2d/v Alors d=t*v/d

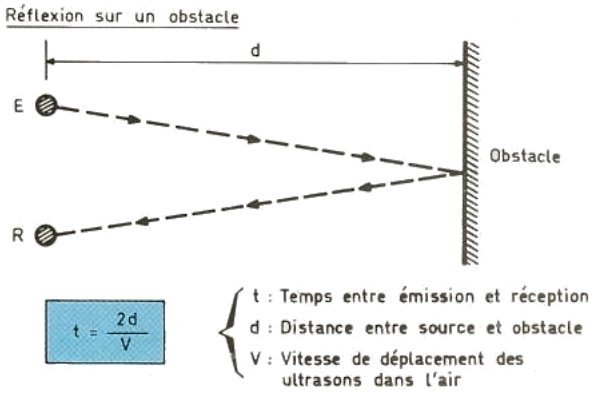

FIGURE 2.19 – Principe de mesure la distance d'un ultrason

2.9 Le module ESP32 DEVKIT V1

ESP32 est une série de micro-contrôleur à faible coût et à faible consommation d'énergie sur puce, avec Wi-Fi et Bluetooth intégrés. La série ESP32 utilise un microprocesseur Tensilica Xtensa LX6 dans des variantes à double cœur et à cœur unique et comprend des commutateurs d'antenne intégrés, un amplificateur de puissance, un amplificateur de réception à faible bruit, des filtres et des modules de gestion de l'alimentation.

Module ESP32 DEVKIT V1 est basé sur un ESP32 cadencé à 240 MHz et exécutant le firmware open source NodeMCU. Cette carte se programme via IDE Arduino (installation d'une extension nécessaire) et nécessite un cordon microUSB.

L'interface sans fil WiFi permet la création de point d'accès sans fil, l'hébergement d'un serveur, la connexion à internet et le partage des données par exemple.

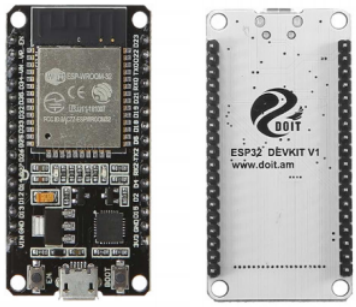

FIGURE 2.20 – Le module ESP32 DEVKIT V1

Brochage de la carte de développement ESP32

La carte de développement ESP32 possède au total 30 broches qui la connectent au monde extérieur. Les connexions sont les suivantes :

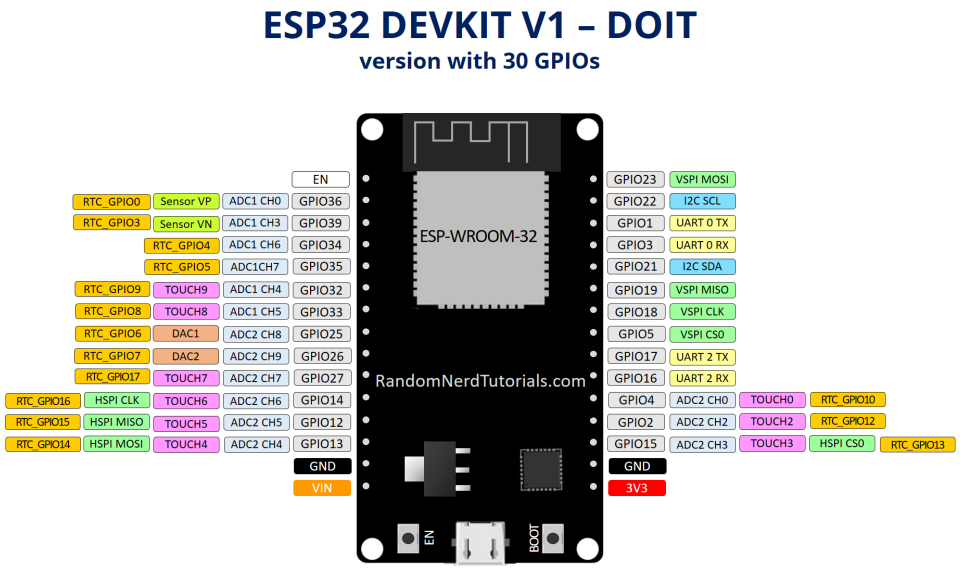

FIGURE 2.21 – Pin out de ESP32 [22]

Pour la programmation du module ESP32 dans ce projet, le logiciel IDE est utilisé. Voici quelques explications qui devraient permettre de réaliser

cette manipulation.

Utilisation de ESP32 avec logiciel IDE

Cette étude présente comment configurer un ESP32 depuis IDE Arduino [22].

• Pour configurer cette carte, la première chose à faire est d'ajouter un lien. Donc, d'abord il faut ouvrir le logiciel IDE Arduino, dans le menu "fichier" puis "Préférences" de l'interface IDE.
• Maintenant il est nécessaire de saisir un lien dans « URL de gestionnaire de cartes supplémentaires », c'est l'étape 2.
Voici le lien hypertexte :

https ://dl.espressif.com/dl/package esp32 index.json

• Choisir le type de carte, en cliquant sur "outils" après "type de carte" puis sur "gestionnaire de la carte", étape 3.

• Une fenêtre va alors apparaître, réaliser une recherche à l'aide du mot clé « esp32 » et le installer, étape 4.

• Après le lancement de l'installation et téléchargement, l'environnement de développement Arduino est maintenant équipé du nécessaire pour programmer la carte. L'étape suivante est la configuration. Pour cela, appuyant sur le menu "Outils" puis sur "Type de carte", puis il suffit de sélectionner la carte correspondante, c'est bien la carte "DOIT ESP32 DEVKIT V1" comme indiqué le module utilisé et puis sélectionné le port, étape 5.

3 La partie logicielle

3.1 Logiciel IDE

Pour programmer la carte Arduino UNO utilisée dans le robot réalisé, le logiciel IDE de Arduino est utilisé.

Un programme utilisateur Arduino est une suite d'instructions élémentaires sous forme textuelle, ligne par ligne. La carte lit puis exécute les instructions l'une après l'autre, selon l'ordre défini par les lignes de code.

Présentation de IDE

L'environnement de développement intégré Arduino est une application opensource qui est écrite dans des fonctions de C et C++.

L'interface du logiciel Arduino

L'interface du logiciel Arduino se présente de la façon suivante :

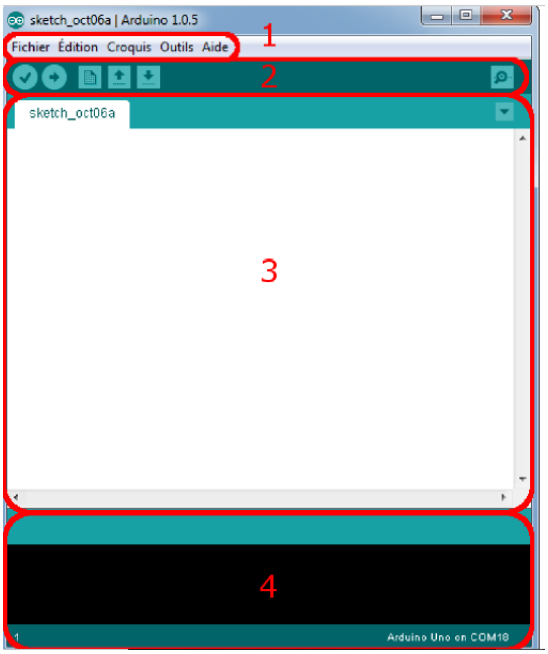

FIGURE 2.22 – Interface du logiciel IDE.

• Le cadre numéro 1 : ce sont les options de configuration du logiciel.
• Le cadre numéro 2 : il contient les boutons qui vont servir lorsque la programmation d'une carte.
• Le cadre numéro 3 : ce bloc va contenir le programme à créer.
• Le cadre numéro 4 : celui-ci est important, car il va aider à corriger les fautes dans un programme. C'est le débogueur.

Les boutons du logiciel IDE

Voyons à présent à quoi servent les boutons, encadrés en rouge et numérotés par les chiffres.

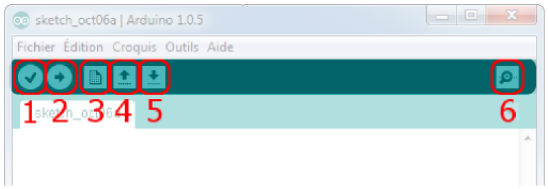

FIGURE 2.23 – Les boutons du logiciel IDE.

- Bouton 1 : ce bouton permet de vérifier le programme, il actionne un module qui cherche les erreurs dans le programme.
- Bouton 2 : charge (téléverse) le programme dans la carte spécifiée.
- Bouton 3 : crée un nouveau fichier.
- Bouton 4 : ouvre un fichier.
- Bouton 5 : enregistre le fichier.
- Bouton 6 : ouvre le moniteur série.

Réception du programme

Le programme rentre donc dans la carte en passant en premier par le connecteur USB de celle-ci. Il va alors subir une petite transformation qui permet d'adapter le signal électrique correspondant au programme (car le programme transite dans le câble USB sous forme de signal électrique) vers un signal plus approprié pour le micro-contrôleur. On passe ainsi d'un signal codé pour la norme USB à un signal codé pour une simple voie série. Puis ce nouveau signal est alors intercepté par le micro-contrôleur [23].

Structure d'un programme Arduino

Un programme Arduino est structuré en 3 grandes parties, exécutées successivement [24] :

1. Déclaration des librairies externes et initialisation des variables : cette partie permet de déclarer les variables globales, qui sont des variables accessibles et utilisables partout dans le programme. Et également permet de manipuler ces données à travers le nom de la variable qui les stocke.

2. Initialisation du programme : la fonction setup est appelée une seule fois lorsque le programme commence. C'est dans cette fonction que l'on va écrire le code qui n'a besoin d'être exécuté une seule fois.

3. Exécution du programme : la fonction loop. Il faut savoir que cette fonction est appelée en permanence. Les instructions écrites dans cette fonction s'exécutent un nombre infini de fois. [24]

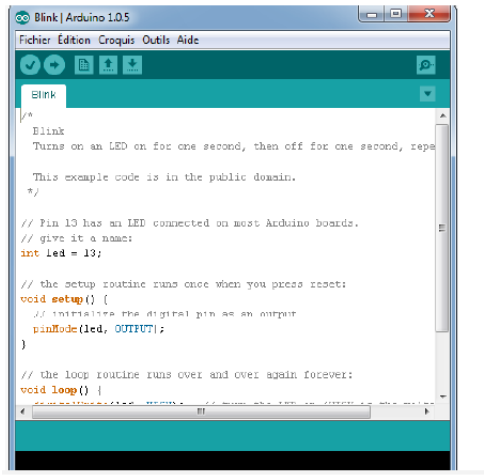

FIGURE 2.24 – Structure d'un programme Arduino

3.2 Logiciel de conception de circuits Fritzing

Fritzing est un logiciel opensource. Il permet de concevoir de façon simple et intuitive des circuits électroniques autant de façon schématique que physique (routage, typon), ainsi que la programmation d'un Arduino ou d'un Picaxe à partir du logiciel. Fritzing permet de couvrir tous les camps de la conception mais il ne permet pas de faire de simulation. La bibliothèque de composants Fritzing permet de créer des schémas de câblage utilisant les divers accessoires spécifiques.

Page d'accueil du logiciel fritzing

Voici la page d'accueil du logiciel fritzing

FIGURE 2.25 – Page d'accueil du logiciel fritzing

3.3 Logiciel de simulation Proteus (isis)

Proteus est une suite logicielle destinée à l'électronique. Développé par la société Labcenter Electronics. Les logiciels incluent dans Proteus permettent la CAO dans le domaine électronique, il permet de réaliser des schémas structurels et les simuler. Il regroupe des logiciels principaux : ISIS, ARES, PROSPICE et VSM [25].

Proteus possède d'autres avantages :

- Pack contenant des logiciels facile et rapide à comprendre et utiliser.
- Le support technique est performant.
- L'outil de création de prototype virtuel permet de réduire les coûts matériel et logiciel lors de la conception d'un projet.

Parmi les différents modules du logiciel proteus, le module ISIS est utilisé dans ce projet, pour éditer des schémas électriques. Par ailleurs, le logiciel permet également de simuler ces schémas ce qui permet de déceler certaines

erreurs dès l'étape de conception.

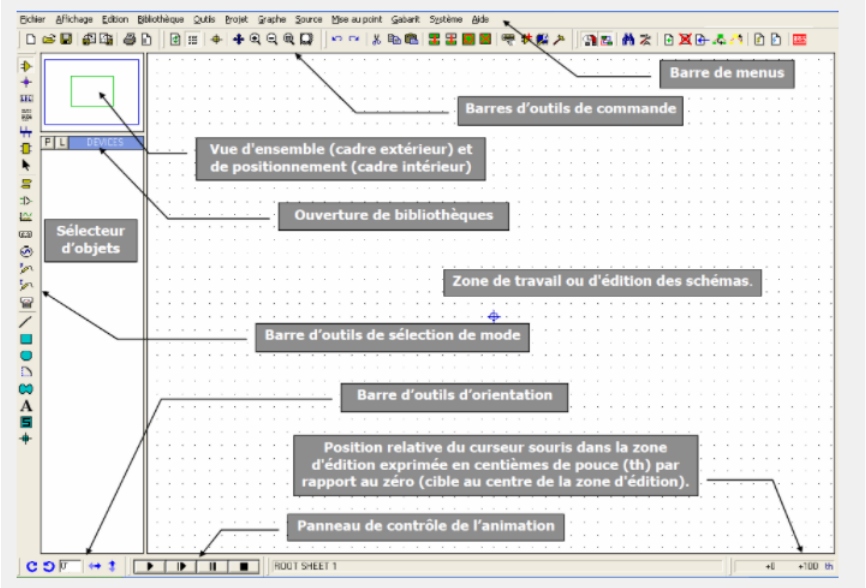

FIGURE 2.26 – Présentation de l'interface

3.4 App Inventor

App Inventor est une application développée par Google et qui est destiné aux utilisateurs d'androïde. C'est un logiciel en ligne qui permet de créer des applications pour appareils Androïde à travers une interface purement visuelle et de configurer les actions de l'application par un système de blocs logiques.

App Inventor est très simple à utiliser, il suffit de combiner certains bloc (ligne de code) avec d'autres afin d'avoir un programme assez satisfaisant [26].

La programmation se réalise en ligne. Seules contraintes : avoir un compte Gmail pour pouvoir y accéder, et un accès à internet évidemment. Les informations sont stockées sur des serveurs distants.

Le concept d'App Inventor

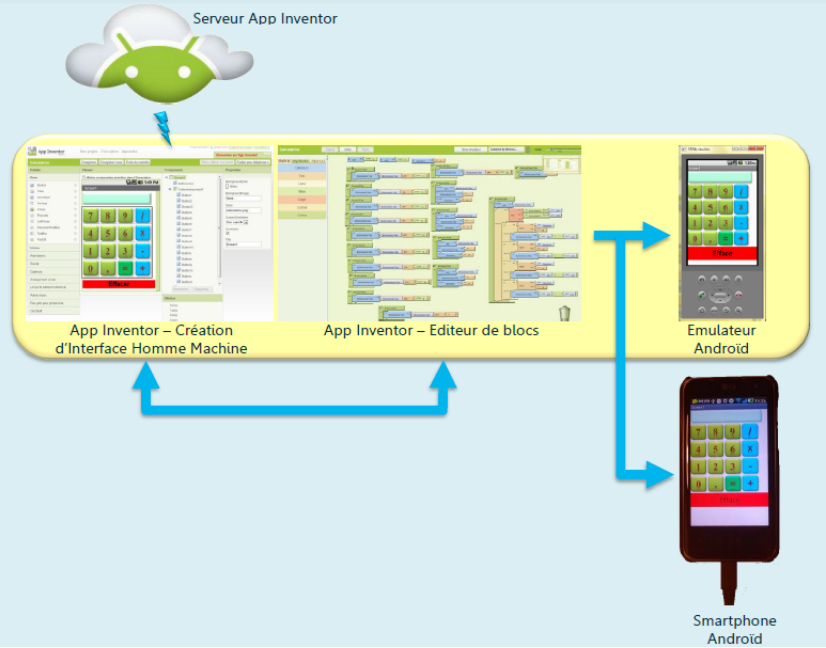

FIGURE 2.27 – Le concept d'App Inventor [27]

L'environnement de App Inventor contient trois fenêtres sont proposées pendant le développement [27] :

1-Une pour la création de l'interface de l'application : ce sera l'allure de l'application.

2-Une pour la programmation par elle-même : elle permettra, par l'assemblage des blocs de créer le comportement de l'application.

3-Une pour l'émulateur qui permettra de tester l'application. L'émulateur permet de remplacer un terminal réel pour vérifier le bon fonctionnement du programme.

La connexion d'un terminal réel sous Androïde permettra ensuite d'y télécharger le programme pour un test réel. Ce terminal pourra aussi bien être un téléphone qu'une tablette, le comportement du programme sera identique.

Historique de logiciel App Inventor

En 2009 c'est le début du développement du logiciel App Inventor par Google à partir des recherches dans l'enseignement de l'informatique faites par le MIT, Boston prés de New York. L'objectif de l'enseignement permet à des étudiants qui débutent en informatique d'apprendre à programmer sans se noyer sous le code Java. Après en 2011 Google rend App Inventor open source, le MIT poursuit le développement et en 2012 version Beta App Inventor diffusé par le MIT encore en version beta aujourd'hui [28].

Structure d'un App Inventor

App Inventor est composé de deux interfaces : interface graphique et interface de blocs.

1. L'interface graphique (design)

Pour créer une application, la première phase est la création de son interface.

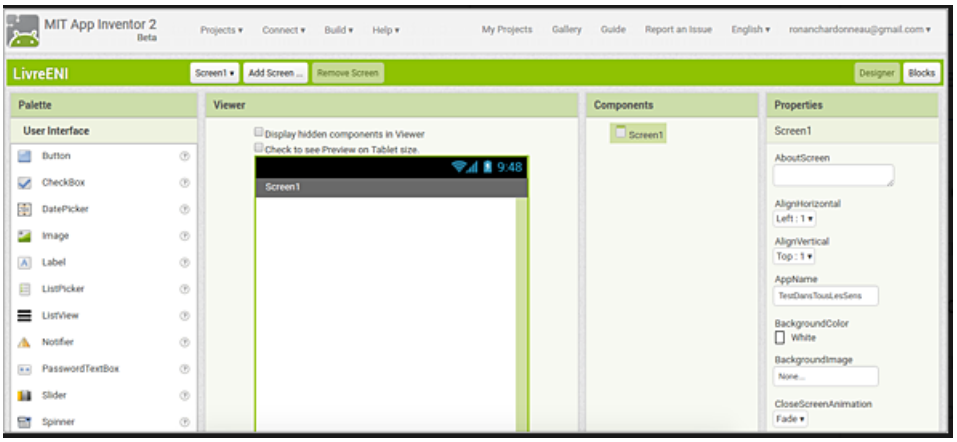

FIGURE 2.28 – Interface graphique de App Inventor

Cette interface graphique contient quatre parties :

1-La première partie : la palette contenant tous les éléments qui peuvent être positionnés sur l'écran du téléphone.

2-La deuxième partie : La zone Viewer Où l'écran s'affiche, il donne un aperçu visuel de l'application, il est possible d'ajouter plusieurs écrans en cliquant sur ce bouton «Add Screen».

La case « Display hidden components in Viewer » permet d'afficher ou les éléments de l'application ou pas afficher ces éléments, en cochant cette case.

3-La troisième partie : La liste des éléments et des médias utilisés sur l'écran.

4-La quatrième partie : Les propriétés des différents éléments utilisés.

2. Interface des blocs

Une fois l'allure de l'application est créée, il est nécessaire de lancer l'éditeur de bloc afin de mettre en œuvre la programmation associée aux différents objets.

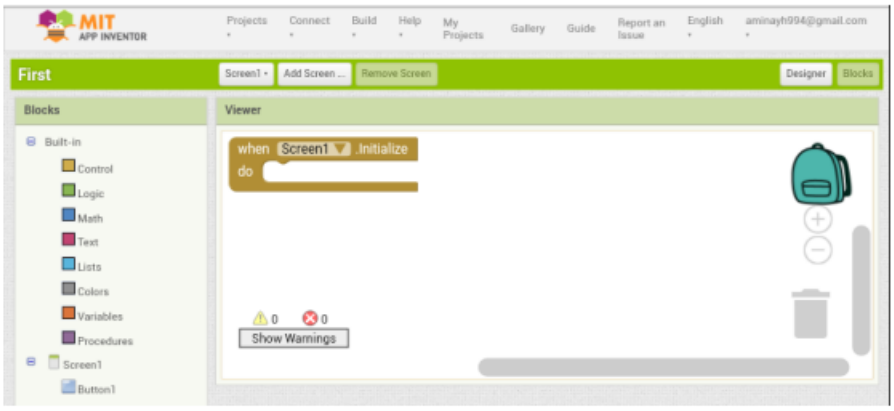

FIGURE 2.29 – Interface des blocs d'AppInventor

L'éditeur de bloc se compose de deux zones :

1-La zone Bloc : dans cette zone, il existe toutes les blocs nécessaires à la programmation, sont composées de deux parties :

- Built-in : c'est des fonctions prédéfinis (les tests, les boucles, les opérations logique...).

- Les éléments placés dans l'interface graphique.

2-La zone Viewer : c'est l'espace de travail pour agencer les blocs.

Il existe deux mode de connexion pour tester cette application, en cliquant sur "connect" :

• Connecter le téléphone en USB.

• Connecter au wifi, et le tester à travers un dispositif Androïde sur lequel il est installé l'application "AI Companion", ce choix AI Companion devrait apparaitre une fenêtre qui contient le QR code, lancer l'application sur le Smartphone, puis saisir ou scanner ou bien faire entrer manuellement ce code pour que l'application puisse démarrer.

Pour suivre le changement des états effectué par la page de commande, on a utilisé une base de données commune entre la page web et l'application Androïde, qui est le Firebase Realtime. La base de données Firebase Realtime utilise la synchronisation des données à chaque fois que les données sont modifiées.

3.5 Firebase Realtime

La base de données Firebase Realtime est une base de données hébergée dans le cloud. Les données sont stockées au format JSON et synchronisées en temps réel avec chaque client connecté. Lorsque on crée des applications multiplateformes avec SDK iOS, Android et JavaScript, tous les clients partagent une instance de base de données en temps réel et reçoivent automatiquement des mises à jour avec les données les plus récentes [29].

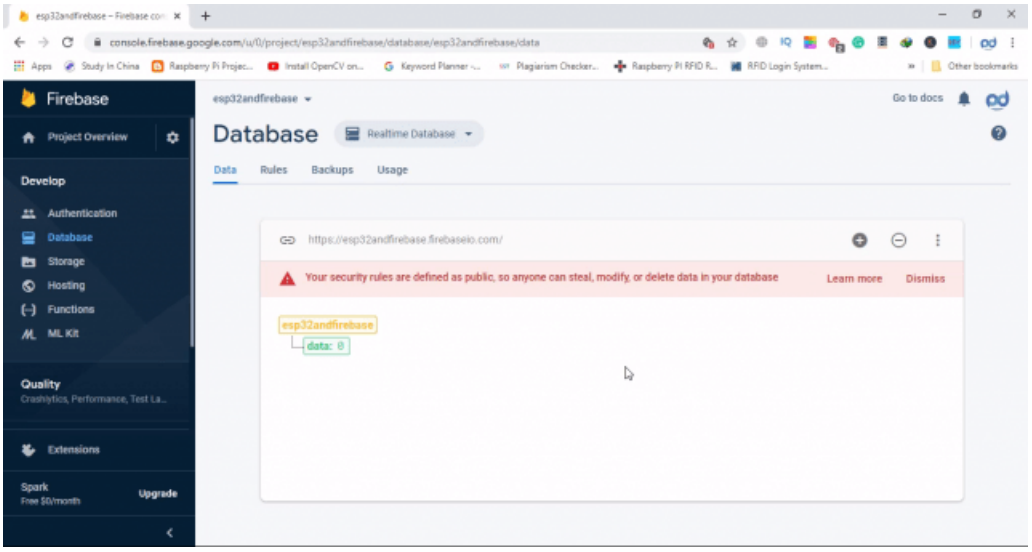

FIGURE 2.30 – Base de données Firebase Realtime

4 Conclusion

Ce chapitre a abordé les différents types de composants utilisé pour réaliser le robot et leurs fonctionnements. Ainsi que l'aspect logiciel qui permet de concevoir la partie commande du projet. Le chapitre qui suivra, sera consacré à la partie réalisation et la mise en œuvre du projet avec des tests réels.

Chapitre 3

Réalisation pratique du projet

1 Introduction

Après avoir mené une étude générale sur le projet, ce dernier chapitre va porter les différents schémas de circuits électriques du robot arrangeur d'objet, ainsi que la présentation de l'application androïde utilisée pour contrôler les étagères conçues comme places du rangement des objets portés par le robot. Et pour finir, présenter les résultats de la réalisation pratique du projet avec le principe de fonctionnement de ce dernier.

2 Présentation synoptique du projet

Ce schéma en bloc explique de manière globale comment fonctionner le projet réalisé.

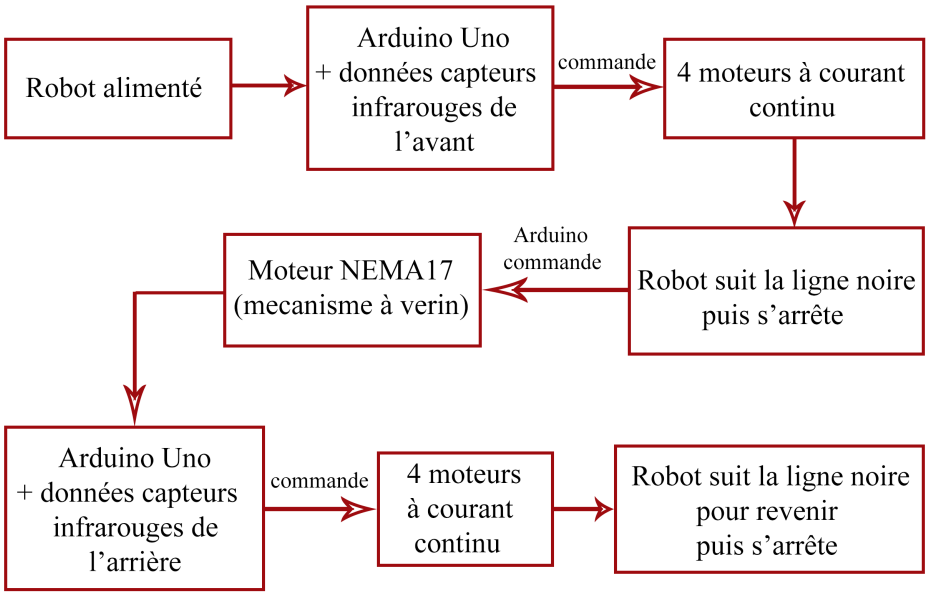

FIGURE 3.1 – Schéma en bloc côté robot

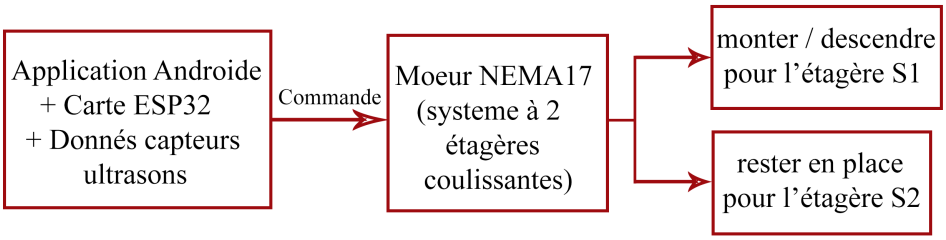

FIGURE 3.2 – Schéma en bloc côté étagères

3 Présentation de l'application Androïde réalisée

Cette application est réalisée à l'aide de l'outil de développement App Inventor pour créer et programmer l'interface de l'application, et également de la base de données en temps réel Firebase pour stocker les donnés qui viennent de la part des capteurs ultrasons installés sur chacune de deux étagères. Elle est composée de trois écrans (Screens) montrés ci-dessous :

- L'interface graphique du premier écran de l'application.

FIGURE 3.3 – Interface graphique écran 1

- L'interface de bloc de l'écran 1 :

Cet écran est programmé pour ouvrir le second écran après trois secondes d'attente.

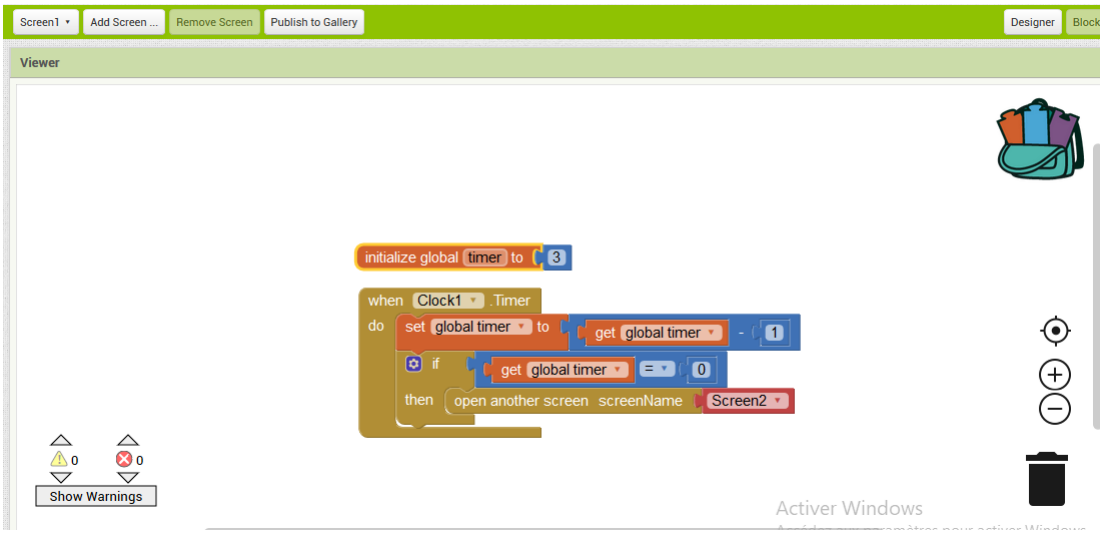

FIGURE 3.4 – Interface de bloc de l'écran 1

- L'interface graphique de l'écran 2 :

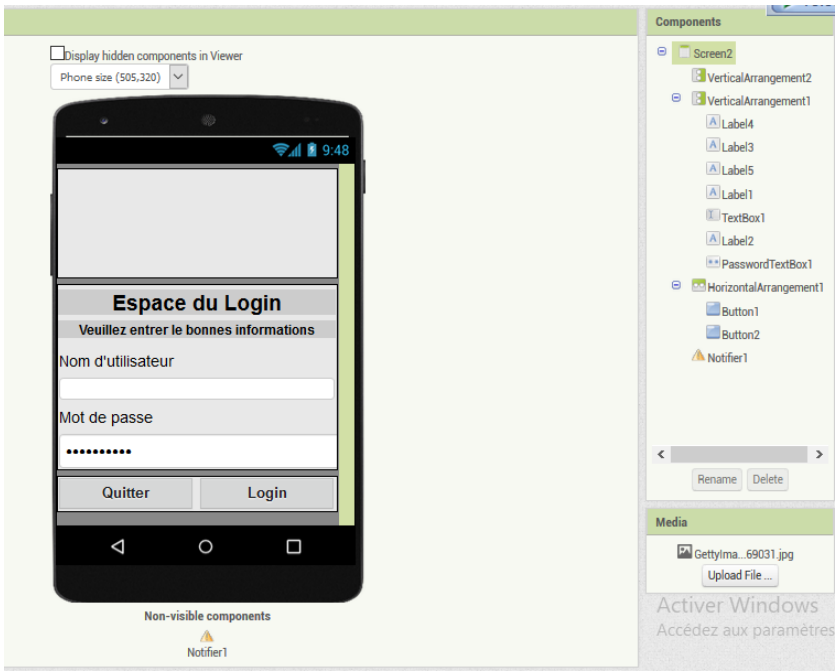

FIGURE 3.5 – Interface graphique de l'écran 2

- L'interface de bloc de l'écran 2 :
Cet écran est destiné pour l'authentification comme mesure de sécurité afin de protéger l'application. Pour que l'application soit seulement accessible par les personnes concernées, un nom d'utilisateur et un mot de passe doivent être insérés.

le non d'utilisateur et le mot de passe utilisés ici sont respectivement : "personne" et "610228". Pour entrer ces informations il faut cliquer sur le bouton "Login", si les informations sont juste, le troisième écran va s'ouvrir, si non, une notification va s'afficher sous le message "Erreur ! ! entrez à nouveau les bonnes infos".

Le bouton "Quitter" sert à quitter l'application.

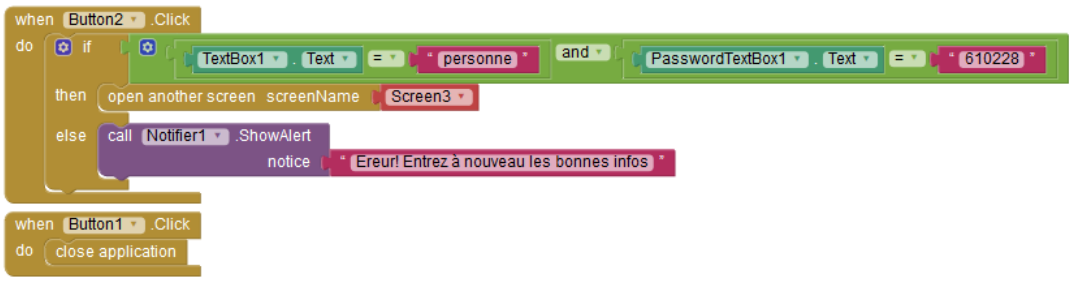

FIGURE 3.6 – Interface de bloc de l'écran 2

- L'interface graphique de l'écran 3 :

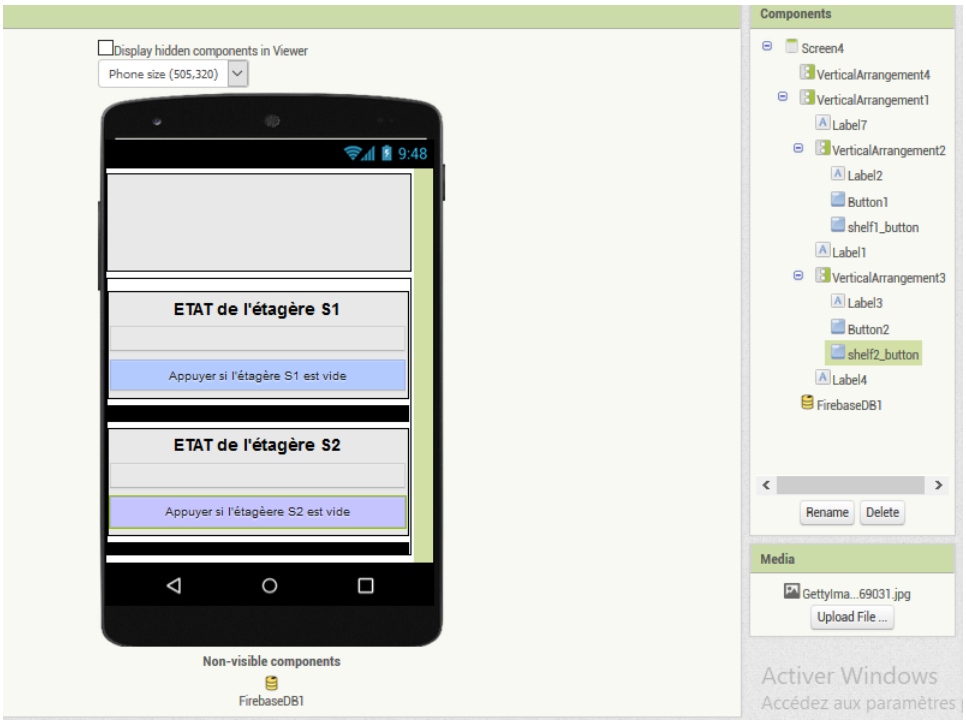

FIGURE 3.7 – Interface graphique de l'écran 3

- L'interface de bloc de l'écran 3 :

Cet écran est dédié pour afficher l'état (soit vide ou pas) des deux étagères coulissantes réalisées dans ce projet (première étagère S1 et la deuxième S2), et pour les contrôler (les faire monter ou descendre).

Ces deux étagères peuvent monter ou descendre pour pouvoir arriver au niveau du robot qui porte les objets. Initialement, le robot est au même niveau de l'étagère S2. Donc c'est l'étagère S1 qui va monter jusqu'au niveau du robot pour pouvoir installer les objets au dessus de celle-ci.

Si l'un des deux étagères est vide, ça va être indiqué sur l'écran "vide" dans une barre rouge, si non, il n'est pas vide et ça va être indiqué dans une barre verte. Les deux boutons bleus sont utiliser pour faire monter ou descendre les étagères.

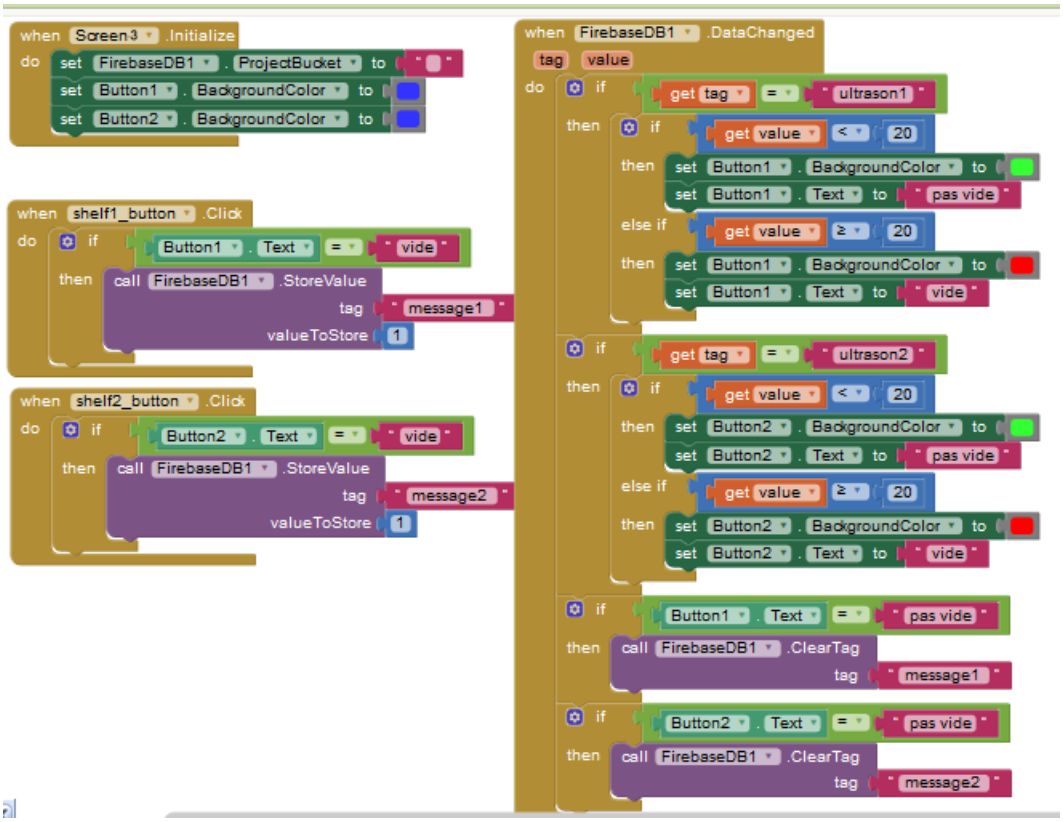

FIGURE 3.8 – Interface de bloc de l'écran 3

La base de données Firebase associée à cette application est montrée ci-dessous, elle est utilisée pour stocker les données qui viennent de la part des deux capteurs ultrasons en temps réel, ça veut dire à chaque moment les données changent, elles vont être enregistrées.

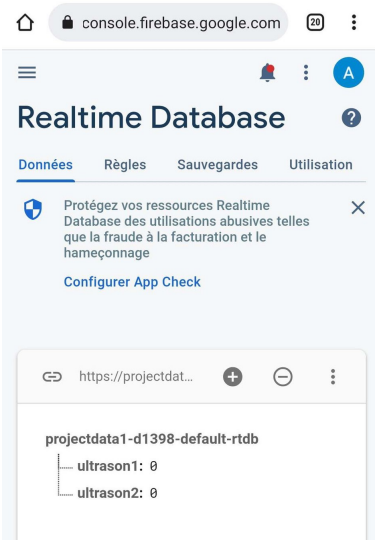

FIGURE 3.9 – Base de données utilisée

4 Montage de la réalisation sous Fritzing

4.1 Pour le robot suiveur de ligne arrangeur d'objet

Les composants mis en place pour réaliser le montage :

1. Pour le robot suiveur de ligne.

- Carte Arduino UNO.
- 4 moteurs a courant continue.
- Driver L298N.
- 4 capteurs infra-rouge (deux en avant, deux en arrière).
- Alimentation 9v.

2. Pour le mécanisme à vérin (poussoir d'objet)

- Moteur NEMA17.
- Driver A4988.
- Alimentation 9v.

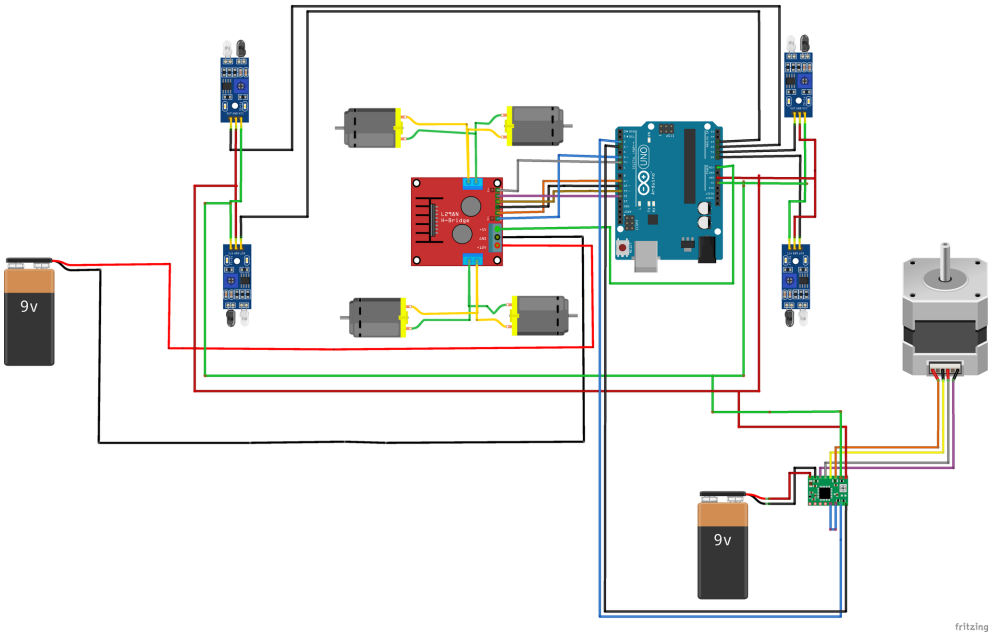

FIGURE 3.10 – Schéma du robot sous Fritzing

4.2 Pour le système de contrôle des étagères

Les composants mis en place pour réaliser le montage :

- Deux Émetteur - Récepteur à ultrasons HC-SR04.
- Module ESP32 DEVKIT V1.
- Alimentations 9v pour le moteur.
- Alimentations 5v pour les capteuru ultrasons
- Moteur NEMA17.
- Driver A4988.

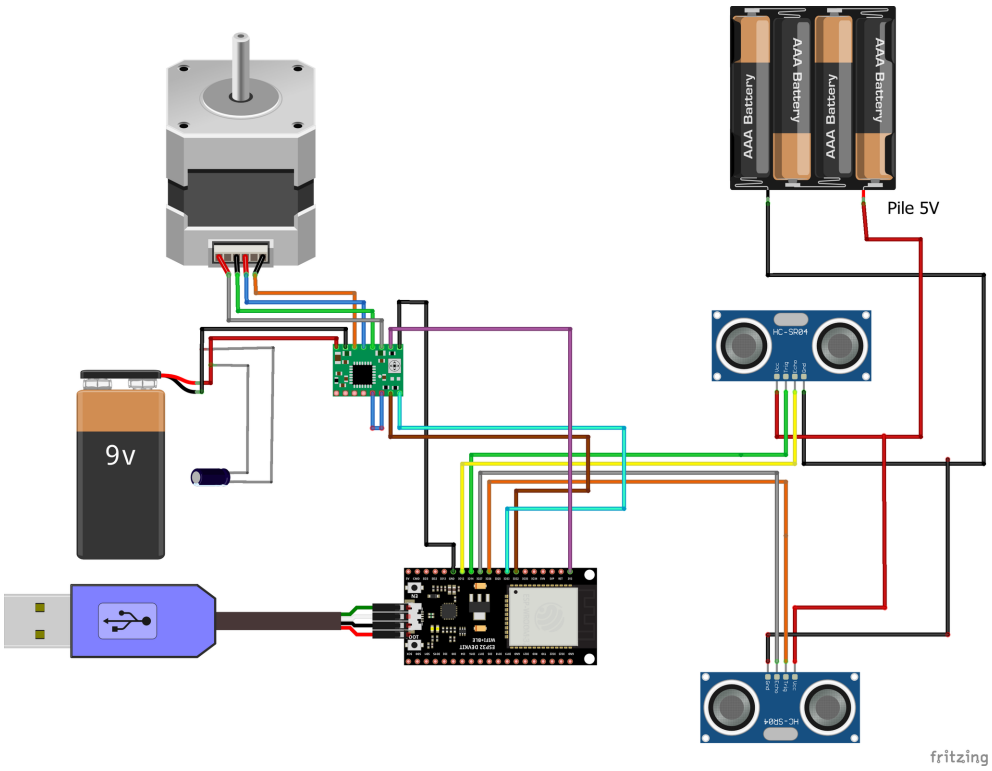

FIGURE 3.11 – Schéma de système de contrôle des étagères sous Fritzing

5 La simulation du robot suiveur de ligne sur Proteus

Le robot réalisé est conçu pour suivre une ligne noire à l'aide de deux capteurs infrarouges montés en avant, déposer les objets, et puis revenir en suivant la même ligne noire à l'aide de deux capteurs infrarouges montés en arrière.

La figure suivante montre la simulation de robot suiveur de ligne.

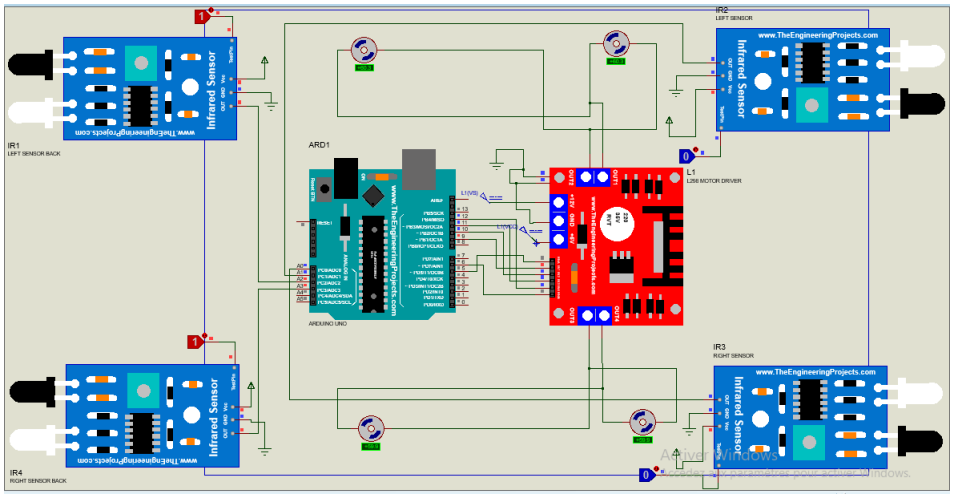

FIGURE 3.12 – Simulation de suiveur de ligne sur Proteus

La simulation de mécanisme à vérin (poussoir d'objet) et celui du système de contrôle des étagères sur Proteus ne sont pas réalisable à cause de l'indisponibilité de la librairie pour le driver A4988 et le module ESP32.

6 Résultats et principe de fonctionnement

Le système expliqué ci-dessous est conçu pour faciliter le processus du rangement des objets dans des étagères, afin de minimiser l'effort et le temps pour les gens destinés à cette tâche.

Le système du rangement réalisé est composé de deux étagères coulissantes contrôlées avec une application connectée à un réseau WiFi, et un robot suiveur de ligne avec un mécanisme qui transport les objets et les pousse jusqu'à l'étagère spécifiée.

Le mécanisme utilisé pour pousser les objets du robot à l'étagère spécifié est composé d'un moteur NEMA17 contrôlé par le driver A4988, ce moteur tourne en résultat le sort du vérin est donc, il va pousser les objets.

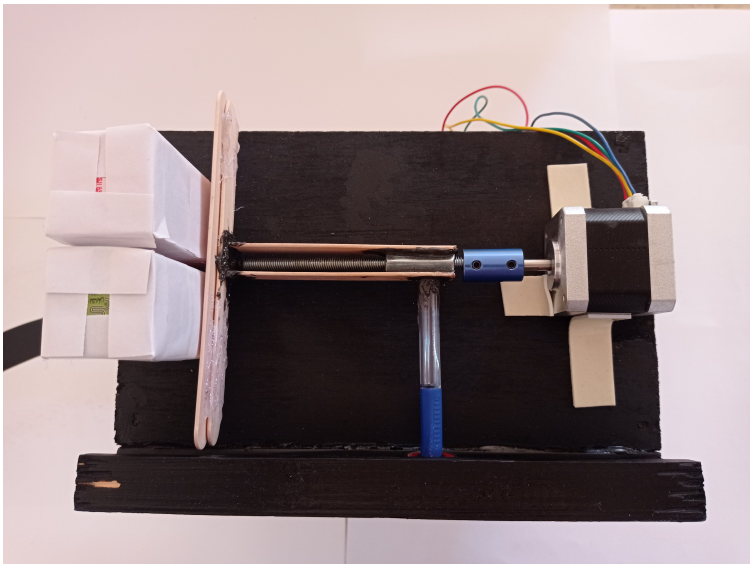

FIGURE 3.13 – Mécanisme de poussoir à vérin

L'application réalisée sur App Inventor, est faite pour indiquer l'état de chaque étagère, utilisé pour cela un capteur ultrason HCSR04 pour chacune d'eux, pour détecter la présence ou l'absence des objets sur chaque étagère, en mesurant la distance. Prenant la largeur de l'étagère comme distance de référence (27 cm), les deux distances mesurées sont envoyées à la base des données utilisées, qui est la base des données en temps réel de GOOGLE (Firebase). Si ces distances sont supérieurs ou égales à la distance de référence, cela signifie que les étagères sont vides et ça va être indiqué sur l'application par une barre rouge avec le mot "vide". Si non, les étagères ne sont pas encore vides et ça va être indiqué sur l'application par une barre verte avec les mots "pas vide".

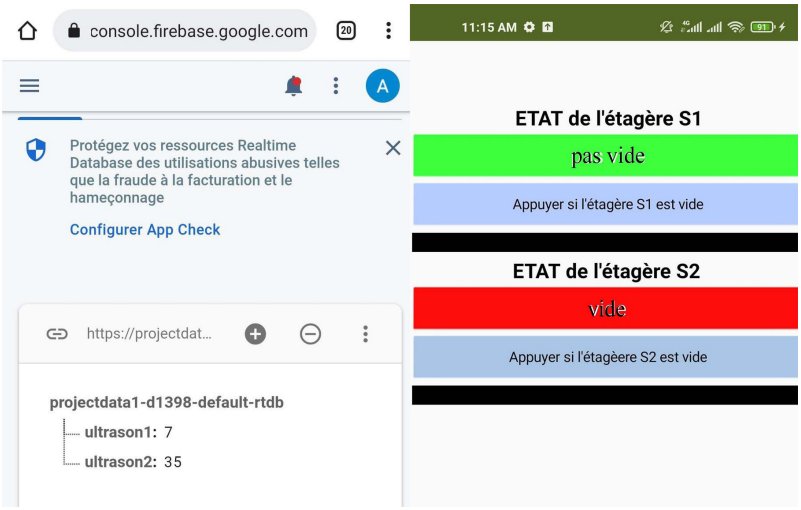

FIGURE 3.14 – Cas où l'étagère S1 pas vide et l'étagère S2 vide

Le châssis montré ci-dessous, alimentés par deux batteries rechargeables de référence 18650, et contrôlé par une carte Arduino UNO, Le châssis porte des objets et à l'appui sur l'interrupteur, ce dernier suit une ligne noire à l'aide de deux capteurs infrarouges monté sur l'avant.

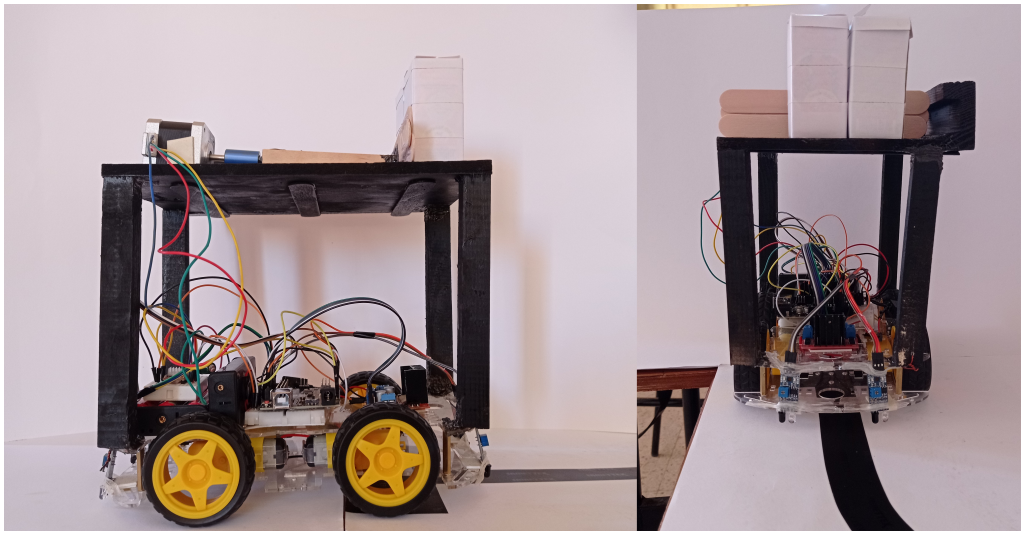

FIGURE 3.15 – Robot suiveur de ligne/porteur d'objets

A la fin, lorsque les deux capteurs détectent la ligne noir, le châssis va s'arrêter car il est devant les étagères.

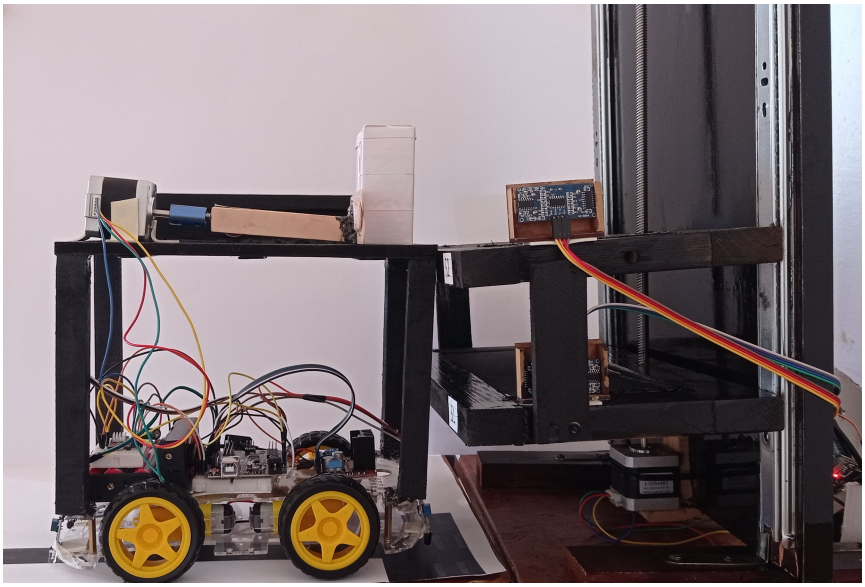

FIGURE 3.16 – Robot devant les étagères

Puisque il y a deux étagères, les objets peuvent être destinés à la première étagère ou bien la deuxième.

Ces deux étagères sont relié l'une avec l'autre pour pouvoir monter ou descendre ensemble. Elles peuvent aller en haut ou en bas selon le besoin. Il est utilisé pour cela un moteur type NEMA17 avec un driver A4988 qui est également contrôlé par l'application.

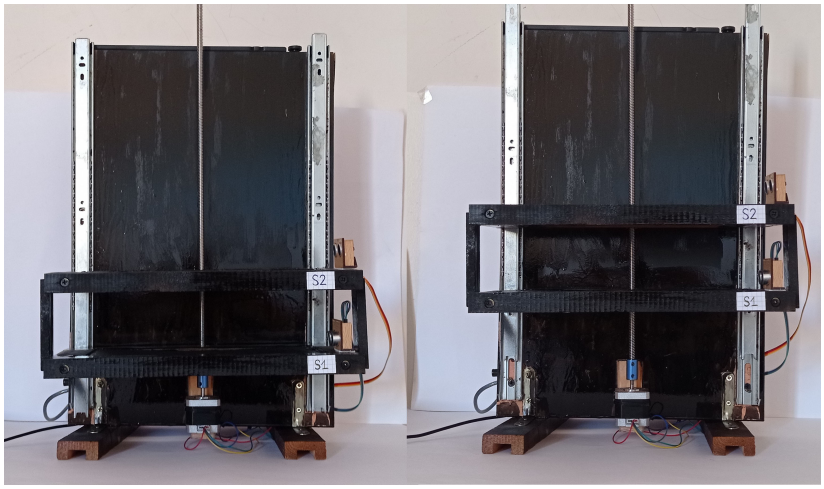

FIGURE 3.17 – Les étagères

Par l'appui sur le bouton nommé « Appuyez si l'étagère S1 est vide » et tant que la 1ère étagère est vide, l'application envoie un message tagué « message1 » de type String vers la base de données utilisée Firebase, la carte ESP32 va lire ce message, s'il est égale à « 1 » , elle va commander le moteur de tourner dans une certaine direction et donc l'étagère monte jusqu'au niveau du châssis, les objets sont déposés, après le moteur tourne dans le sens inverse et l'étagère descendant,et il revient à sa position initiale. L'étagère ne sera plus vide, alors le message envoyé à Firebase sera effacé pour ne plus tourner le moteur.

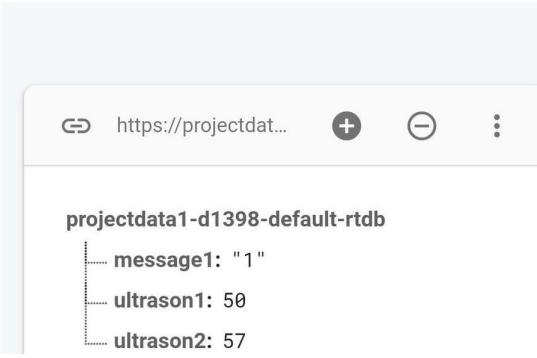

FIGURE 3.18 – Envoi du message1 à Firebase

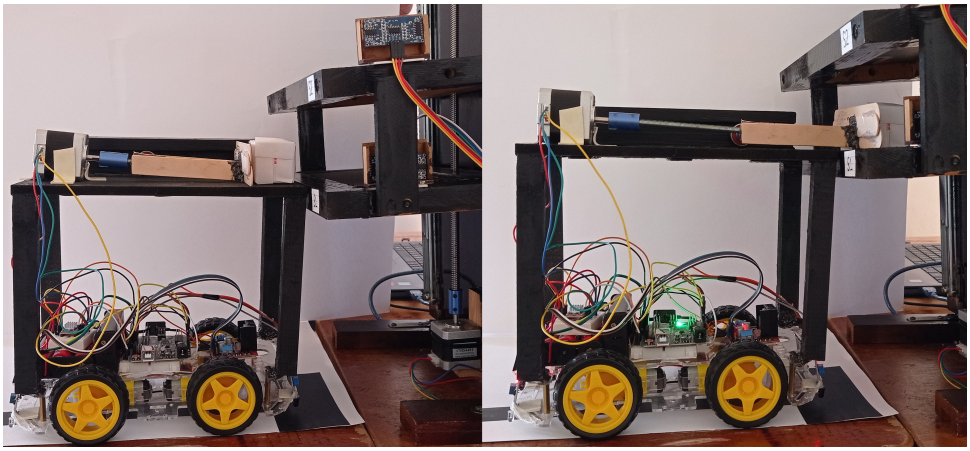

FIGURE 3.19 – Dépôt d'objet sur l'étagère S1

Si les objets sont destinés au deusième étagère qui est vide et le bouton nommé « Appuyez si l'étagère S2 est vide » est appui, l'application envoie un message tagué « message2 » de type String à la base des données, la carte ESP32 va lire ce message, s'il est égale à « 1 » le moteur ne tourne pas et l'étagère reste à sa position initiale parce qu'elle est déjà au niveau du châssis, et les objets sont déposés sur l'étagère.

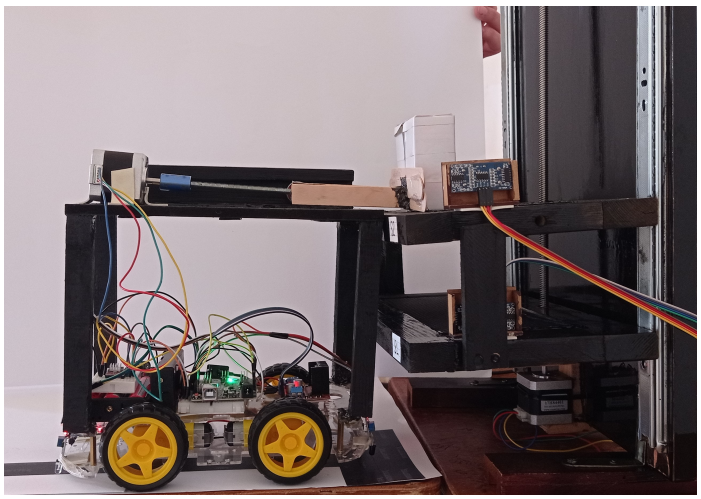

FIGURE 3.20 – Dépôt d'objet sur l'étagère S2

Comme l'étagère ne sera plus vide, alors le message envoyé à Firebase sera effacé.

Lorsque les objets sont déposés, l'application indique qu'elles sont remplies à nouveau (pas vide) pour l'une ou les deux étagères. Après le dépôt des objets, le châssis revient en suivant la même ligne noire à sa position initiale à l'aide des deux autres capteurs infrarouges monté sur l'arrière du ceclui-ci et s'arrête.

7 Conclusion

Ce dernier chapitre est dédié pour présenter la solution proposée concernant le rangement des objets, qui a été le problème majeur résultant la réalisation de ce projet, en expliquant également le principe de fonctionnement en détails après avoir présenter au premier lieu les différents montages électroniques du projet.

Conclusion générale

Dans le monde industriel, l'automatisation est l'un des éléments les plus importants pour le développement. Elle permet de réduire le besoin d'êtres humains en créant des systèmes d'aide supplémentaires qui peuvent augmenter l'efficacité et la productivité. Le domaine de l'automatisation consiste à fabriquer des équipements sophistiqués qui sont utilisés quotidiennement par les gens qui avaient besoin de machines auxiliaires en place où leur force n'était pas suffisante.

Ce mémoire a pour ambition de réaliser un système destiné à faciliter la tache du rangement des objets dans des étagères, permettant pour cela de minimiser le temps et l'effort des gens mis à cette tache. Alors, pour bien mener ce travail, il est présenté en trois parties. Il a fallu dans un premier temps présenter les robots d'une manière générale, leur historique, leurs types, et les avantages et les inconvénients de ces derniers. Le seconde chapitre cite les éléments et les composants matériels et logiciels qui en permis de réaliser le système du rangement, qui est composé d'un robot mobile suiveur de ligne, deux étagères coulissantes et une applications qui permet de contrôler le processus de système et afficher certaines paramètres importantes reliées à ce dernier. Au final, les résultats de la simulation et la réalisation pratique de chaque partie du système sont présentées et expliquées de manière détaillée qu'elles sont bien satisfaisantes.

Ce modeste travail, nous a mis en mesure d'appréhender le monde de la robotique et l'automatisme et d'en comprendre les enjeux et les techniques ainsi que de mettre en œuvre les compétences acquises au long de nos études.

Bibliographie

[1] H. Hamdi. *Introduction à la robotique*, Les éditions de l'université Mentouri, Constantine, 2003.

[2] www.maxongroup.fr/maxon/view/application/Romeo-Lassistant-du-futur. Consulté le 08 Octobre 2021.

[3] Futura-sciences.com/tech/definitions-robotique-robot-8433.

[4] diwo.bq.com/fr/qu-est-ce-qu-un-robot-apprendre-a-connaitre-les-capteurs-et-les-actionneurs. Consulté le 08 Octobre 2021.

[5] Stéphane Lens, *locomotion d'un robot mobile*, mémoire de fin d'études, l'université de liège, faculté des sciences appliquées, institut Montefiore, MAI 2008.

[6] J. D. Warren, J. Adams, H. Molle. *Arduino Robotics*, Springer, New York, 2011.

[7] sites.google.com/site/technovhugogassinc36fg5/la-robotique-actuelle. Consulté le 08 Octobre 2021.

[8] www.arduino-france.com/tutoriels/quest-ce-que-arduino. Consulté le 09 Octobre 2021.

[9] www.techmania.fr/arduino/Decouverte_arduino.pdf Consulté le 09 Octobre 2021.

[10] npoulain.fr/intro-arduino. Consulté le 09 Octobre 2021.

[11] sites.google.com/site/sciencesdunumerique/robotique/les-outils. Consulté le 09 Octobre 2021.

[12] bentek.fr/2-arduino-uno. Consulté le 09 Octobre 2021.

[13] www.lastminuteengineers.com/l298n-dc-stepper-driver-arduino-tutorial. Consulté le 09 Octobre 2021.

[14] www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf. Consulté le 09 Octobre 2021.

[15] arduino.blaisepascal.fr/pont-en-h-l298n. Consulté le 09 Octobre 2021.

[16] www.pololu.com/product/2267. Consulté le 09 Octobre 2021.

[17] www.pololu.com/product/1182. Consulté le 09 Octobre 2021.

[18] devenez-pro-en-electronique.com/comprendre-et-bien-utiliser-le-driver-a4988-pour-moteurs-pas-a-pas. Consulté le 10 Octobre 2021.

[19] howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino. Consulté le 09 Octobre 2021.

[20] circuitdigest.com/electronic-circuits/ir-sensor-circuit-diagram. Consulté le 10 Octobre 2021.

[21] www.reality.be/elo/labos2/ultrasons.htm. Consulté le 10 Octobre 2021.

[22] R. Santos, S. Santos. *ESP32 web server with Arduino IDE* (ebook). Disponible sur randomnerdtutorials.com/download. Consulté le le 10 Octobre 2021.

[23] S. Landrault, H. Weisslinger.*Arduino : Premiers pas en informatique embarquée* (ebook), 2020. Disponible sur : eskimon.fr/extra/ebooks/arduino-premiers-pas-en-informatique-embarquee.pdf. Consulté le 10 Octobre 2021.

[24] Astalaseven, Eskimon, Olyte. *Arduino pour bien commencer en électronique et en programmation* (ebook), 2012. Disponible sur : wiki.mdl29.net/lib/exe/fetch.php ?media=elec%3Aarduino-pour-bien-commencer-en-electronique-et-en-programmation.pdf Consulté le 10 Octobre 2021.

[25] www.elektronique.fr/logiciels/proteus.php. Consulté le 10 Octobre 2021.

[26] www.appinventor.mit.edu. Consulté le 10 Octobre 2021.

[27] sig.fgranotier.info/IMG/pdf/debuter_app_inventor.pdf. Consulté le 10 Octobre 2021.

[28] blogpeda.ac-poitiers.fr/lp2i-si/2013/01/23/developper-des-applications-android-avec-app-inventor. Consulté le 10 Octobre 2021.

[29] www.firebase.google.com. Consulté le 10 Octobre 2021.

[30] docs.arduino.cc/hardware/uno-rev3. Cosulté le 11 Octobre 2021.

Annexe

Le code IDE du robot arrangeur d'objet

chassis_prog §

```
//Définir les pins à relier avec le driver A4988.
const int stepPin = 2;
const int dirPin = 3;
//variable pour compter le temps en millisecondes.
long nowMillis;
//Définir les pins à relier avec le driver L298N.
int motorA_E1 = 9;
int motorA_E2 = 10;
int motorA_speed = 5;
int motorB_E3 = 11;
int motorB_E4 = 12;
int motorB_speed= 6;
//Définir les pins des quatres capteurs infrarouges
int left_sensor_pin = A1;
int right_sensor_pin = A0;
int left_sensor_back_pin=A2;
int right_sensor_back_pin=A3;

// intervalles(millisecondes) indiquent le temps
//quand le processus 1 ou 2 doit être effectuer à nouveau.

long intervall = 50;
long interval2=50000;


void input()
{
//pour compter le temps en millisecondes.
nowMillis=millis();
```

```
}

//variables utilisées pour savoir que le processus 1 ou deux est achevé.
boolean procldone,proc2done;

//fonction pour marcher en avant.
void forward(){
    digitalWrite(motorA_E1,HIGH);
    digitalWrite (motorA_E2,LOW);
    digitalWrite(motorB_E3,LOW);
    digitalWrite (motorB_E4,HIGH);


}
//fonction pour arreter les moteurs.
void off() {
    digitalWrite(motorA_E1,LOW);
    digitalWrite (motorA_E2,LOW);
    digitalWrite(motorB_E3,LOW);
    digitalWrite (motorB_E4,LOW);

}
//fonction pour tourner à droite.
void turnRight(){
    digitalWrite(motorA_E1,HIGH);
    digitalWrite (motorA_E2,LOW);
    digitalWrite(motorB_E3,HIGH);
    digitalWrite (motorB_E4,LOW);



 }
 //fonction pour tourner à gauche.
 void turnLeft() {
     digitalWrite(motorA_E1,LOW);
     digitalWrite (motorA_E2,HIGH);
     digitalWrite(motorB_E3,LOW);
     digitalWrite (motorB_E4,HIGH);

 }
 //fonction pour marcher en arrière.
 void backward(){

     digitalWrite(motorA_E1,LOW);
     digitalWrite (motorA_E2,HIGH);
     digitalWrite(motorB_E3,HIGH);
     digitalWrite (motorB_E4,LOW);
 }

 //foction pour dérouler les deux processus.
 void processing()
 {
   //définition d'un variable pour stocker la dernière fois(en temps millisecondes)
   //que le 1er processus a été effectué.
   static unsigned long lastProc1;
   //définition d'un variable pour stocker la dernière fois(en temps millisecondes)
   //que le 2eme processus été effectué.
   static unsigned long lastProc2;
```

```
// 1er processus
// à achaque fois intervall(50ms) est atteint, effectuer les instructions suivantes.
if (nowMillis-lastProc1>intervall)
{
  lastProc1=nowMillis;
  proc1done=true;
  //lire les valeurs des deux infrarouges montés en avant.
  boolean left_sensor_state=digitalRead(left_sensor_pin);
  boolean right_sensor_state=digitalRead(right_sensor_pin);

  if (left_sensor_state==0 and right_sensor_state==0)
   {
    analogWrite (motorA_speed,150);
    analogWrite (motorB_speed,150);
    forward();

   }
  else if (left_sensor_state==1 and right_sensor_state==0)
   {
    analogWrite (motorA_speed,250);
    analogWrite (motorB_speed,250);
    turnLeft();

   }


    else if (left_sensor_state==0 and right_sensor_state==1)
    {
      analogWrite (motorA_speed,250);
      analogWrite (motorB_speed,250);
      turnRight();

     }
    else if (left_sensor_state==1 and right_sensor_state==1)
     {
      analogWrite (motorA_speed,0);
      analogWrite (motorB_speed,0);
      off();
      delay(1000);
      //Régler le sens de rotation dans un sens.
      digitalWrite(dirPin,LOW);
       // Faire 200 impulsions(pas) pour faire un cycle complet de rotation.
       // faire 90 rotations complètes.
       for(int x = 0; x < 90*200; x++) {
          digitalWrite(stepPin,HIGH);
          delayMicroseconds(500);
          digitalWrite(stepPin,LOW);
          delayMicroseconds(500);
          }
     delay(1000); //1s pause
```

```
      //Régler le sens de rotation dans le sens inverse.
     digitalWrite(dirPin,HIGH);
      // Faire 200 impulsions(pas) pour faire un cycle complet de rotation.
      // faire 90 rotations complètes.
      for(int x = 0; x < 90*200; x++) {
         digitalWrite(stepPin,HIGH);
         delayMicroseconds(500);
         digitalWrite(stepPin,LOW);
         delayMicroseconds(500);
         }


     //rendre interval2 50ms pour sortir du 1er processus vers le 2eme
     //et le executer en boucle(chaque 50ms)
     interval1=50000;
     interval2=50;
      }
  }
  else proc1done=false;


 // 2eme processus
  if (nowMillis-lastProc2>interval2)
  {
    lastProc2=nowMillis;
    proc2done=true;
    //lire les valeurs des deux infrarouges montés en arrière.
    boolean left_sensor_back_state=digitalRead(left_sensor_back_pin);
    boolean right_sensor_back_state=digitalRead(right_sensor_back_pin);


    if (left_sensor_back_state==0 and right_sensor_back_state==0)
     {
       analogWrite (motorA_speed,130);
       analogWrite (motorB_speed,130);
       backward();

     }
    else if (left_sensor_back_state==1 and right_sensor_back_state==0)
     {
       analogWrite (motorA_speed,250);
       analogWrite (motorB_speed,250);
       turnRight();
     }
```

```
    else if (left_sensor_back_state==0 and right_sensor_back_state==1)
     {
       analogWrite (motorA_speed,250);
       analogWrite (motorB_speed,250);
       turnLeft();
     }

     else if (left_sensor_back_state==1 and right_sensor_back_state==1)
     {
       analogWrite (motorA_speed,0);
       analogWrite (motorB_speed,0);
       off();
       delay(3000);

     }
  }
  else proc2done=false;
}

void output()
{
  if (proc1done) Serial.println("Just finished task ONE");
  if (proc2done) Serial.println("Just finished task TWO");

}
```

```
void setup()
{
  //Initialisation communication entre la carte Arduino UNO et l'ordinateur
  Serial.begin(9600);
  //Configurer les pins en sorties ou en entrées.
  pinMode(motorA_E1, OUTPUT);
  pinMode(motorA_E2, OUTPUT);
  pinMode(motorB_E3, OUTPUT);
  pinMode(motorB_E4, OUTPUT);
  pinMode(motorA_speed, OUTPUT);
  pinMode(motorB_speed, OUTPUT);
  pinMode(left_sensor_pin, INPUT);
  pinMode(right_sensor_pin, INPUT);
  pinMode(left_sensor_back_pin, INPUT);
  pinMode(right_sensor_back_pin, INPUT);

  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);


}

void loop()
{
  input();
  processing();
  output();
}
```

Le code IDE du system de contrôle des étagères

etagere_programme

```
#include <FirebaseESP32.h>//Inclure la bibliothèque Firebase ESP32.

#include <WiFi.h>// inclure WiFi library.
#include <HCSR04.h>// inclure la bibliothèque de l'ultrason HCRS04.

//Définir les infos de FIREBASE et réseau WiFi.
#define FIREBASE_HOST "https://projectdata1-d1398-default-rtdb.firebaseio.com/"
#define FIREBASE_AUTH "MVPtEnlrelztQXWQv2rwX8vIRpdgyzXWUQprldp2"
#define WIFI_SSID "Redmi Note 9"
#define WIFI_PASSWORD "27c817e4f366"

FirebaseData firebaseData; //Définir l'objet de données Firebase.

HCSR04 hc1(26,27);//initialisation class HCSR04 (trig pin , echo pin)

HCSR04 hc2 (14,12);//initialisation class HCSR04 (trig pin , echo pin)

//Définir les pins step et direction du driver A4988.
const int stepPin = 32;
const int dirPin = 33;



void setup() {
  //Commencer la communication entre la carte et l'ordinateur.
  Serial.begin(115200);
  //Initialiser les variables de moteur driver A4988 en tant que sorties.
  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  //Connecter au réseau Wi-Fi avec SSID et mot de passe.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  //Afficher le message "Connecting to Wi-Fi" en moniteur série.
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(300);
  }
  Serial.println();
  Serial.print("Connected with IP: ");
  Serial.println(WiFi.localIP());//Afficher l'adresse IP locale en moniteur série.
  Serial.println();
  // Initialiser la bibliothèque avec Firebase Authentifacation code et Host link.
  Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  //Activer la reconnexion automatique du WiFi en cas de perte de connexion.
  Firebase.reconnectWiFi(true);}
```

```
void loop(){

 int distance1= hc1.dist();//Calculer la distance entre ultrason 1 et un obstacle.
 int distance2= hc2.dist();//Calculer la distance entre ultrason 2 et un obstacle.
 //Afficher les deux distances en moniteur série.
 Serial.print( "distance1 = " );
 Serial.println( distance1 );
 Serial.print( "distance2 = " );
 Serial.println( distance2 );

//Envoyer les valeurs de les deux distances vers Firebase.
if (distance1>=27){

   Firebase.setInt( firebaseData,"ultrason1", distance1);

   }
   else{

    Firebase.setInt( firebaseData,"ultrason1", distance1);

   }

   if (distance2>=27){
       Firebase.setInt( firebaseData,"ultrason2", distance2);
      }
      else{
       Firebase.setInt( firebaseData,"ultrason2", distance2);
      }
   //Lire la valeur de l'élément de type String "message1" en Firebase.
   if (Firebase.getString(firebaseData, "message1")){

         //if (fbdo.dataTypeEnum() == fb_esp_rtdb_data_type_integer)
         Serial.println(firebaseData.to<String>());
         }
   else{
       Serial.println(firebaseData.errorReason());
     }
       // Si la valeur de "message1" égale à "1"
       if(firebaseData.to<String>()== "1"){
       //Régler le sens de rotation dans le sens des aiguilles d'une montre.
       digitalWrite(dirPin,LOW);
          // Faire 200 impulsions(pas) pour faire un cycle complet de rotation.
          // faire 10 rotations complètes.
          for(int x = 0; x < 12*200; x++) {
             digitalWrite(stepPin,HIGH);
             delayMicroseconds(500);
             digitalWrite(stepPin,LOW);
             delayMicroseconds(500);
             }
       delay(20000); //20s pause
```

```
  //Régler le sens de rotation dans le sens inverse des aiguilles d'une montre.
  digitalWrite(dirPin,HIGH);
    // Faire 200 impulsions(pas) pour faire un cycle complet de rotation.
    // faire 10 rotations complètes.
    for(int x = 0; x < 12*200; x++) {
       digitalWrite(stepPin,HIGH);
       delayMicroseconds(500);
       digitalWrite(stepPin,LOW);
       delayMicroseconds(500);
       }
       Serial.println("près pour mettre les objets sur l'étagère 1");
  delay(600);// 600ms pause
 }
  else {
      }

if (Firebase.getString(firebaseData, "message2")){
  //if (fbdo.dataTypeEnum() == fb_esp_rtdb_data_type_integer)
    Serial.println(firebaseData.to<String>());
     }
else{
  Serial.println(firebaseData.errorReason());
}
// Si la valeur de "message2" égale à "1"
  if(firebaseData.to<String>()== "1"){
    Serial.println("près pour mettre les objets sur l'étagère 2");
  }}
```

Le schéma interne de la carte Arduino UNO

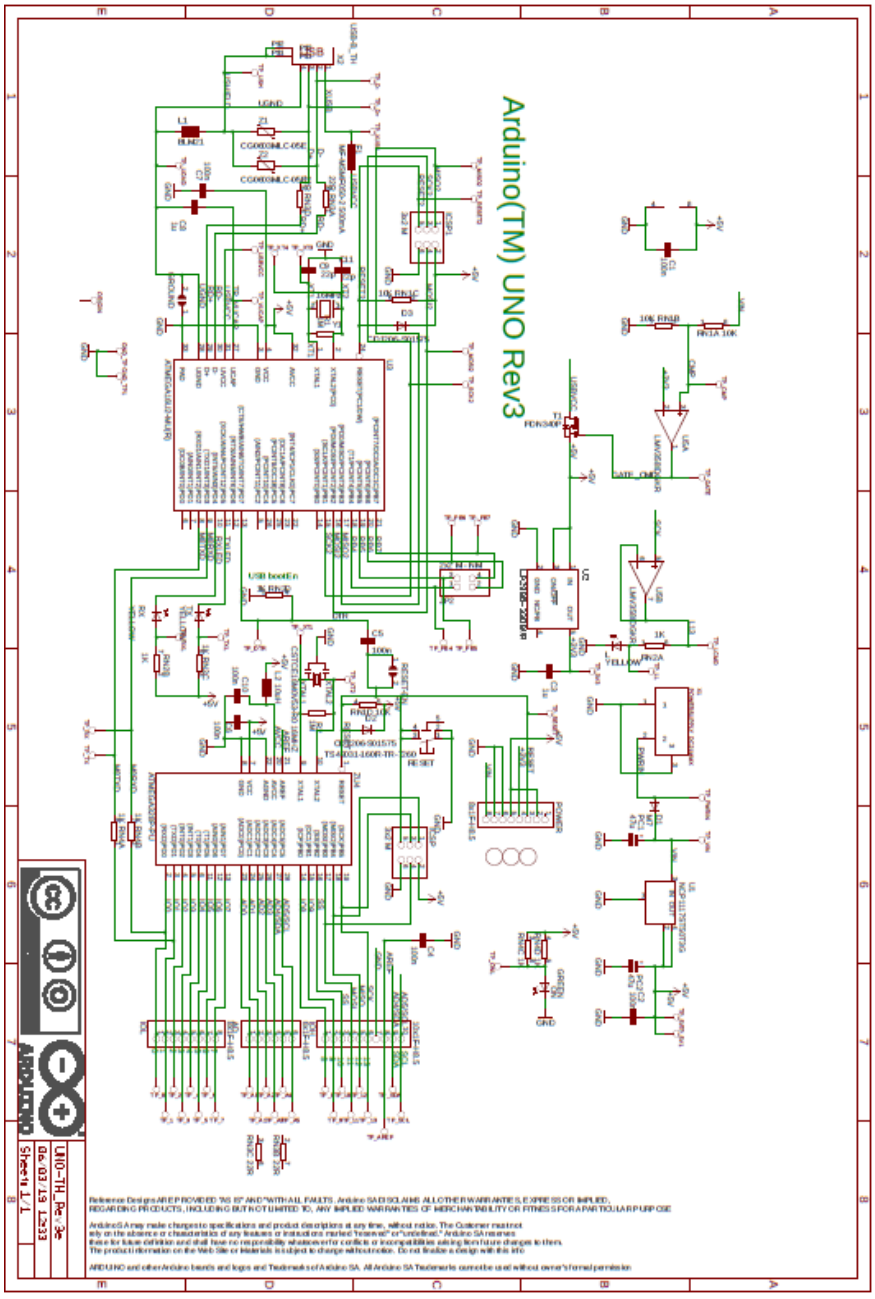

FIGURE 3.21 – Schéma interne de la carte UNO [30]

ملخص

أصبحت الروبوتات لا غنى عنها في مختلف جوانب الحياة، فهي تعد معدات متطورة أصبحت تستخدم يوميًا لتحل محل القوة البشرية، من أجل زيادة كفاءة وإنتاجية العمل. يتضمن هذا المشروع صنع روبوت عن طريق استخدام لوحات إلكترونية قابلة للبرمجة (Arduino UNO, ESP32) مما يجعل مهمة الترتيب أكثر سهولة، فهو يحمل الأشياء ويضعها على الرف المحدد لها، مع العلم بحالة الرف (فارغ أو لا) بالاستعانة بتطبيق اندرويد مبني على App Inventor .

الكلمات المفتاحية. روبوت، ترتيب، أردوينو، تطبيق اندرويد.

Résumé

Les robots sont devenus indispensables dans les différents aspects de la vie, ce sont des équipements sophistiqués qui sont utilisés quotidiennement pour remplacer la force humaine, afin d'augmenter l'efficacité et la productivité du travail. Ce projet consiste à réaliser un robot suiveur de ligne arrangeur d'objet à l'aide des cartes électroniques programmables (Arduino UNO, ESP32), et qui permet de rendre la tache du rangement moins pénible, il porte des objets et les installe sur l'étagère spécifié tout en sachant l'état de l'étagère (vide ou pas) en utilisant une application Androïde réalisée sur App Inventor.

Mots clés : Robot, Rangement, Suiveur de ligne, Arduino, Application Androïde.

Abstract

Robots have become essential in every aspect of life, they are now used daily to replace human strength, in order to increase the efficiency and productivity of work. This project consists in making a line follower robot to arrange objects, using programmable electronic boards (Arduino UNO, ESP32), and which makes it easier to store objects. It carries the objects and installs them on the specified shelf, all with knowing the state of the shelf (empty or not) using an Android application made on App Inventor.

Keywords: Robot, Object storage, Line follower, Arduino, Android Application.