الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

---

جامعة وهران 2 محمد بن أ حمد
Université d'Oran 2 Mohamed Ben Ahmed

معهد الصيانة والأمن الصناعي
Institut de Maintenance et de Sécurité Industrielle
Département de Maintenance en Instrumentation

MÉMOIRE

Pour l'obtention du diplôme de Master

Filière : Génie Industriel
Spécialité : Maintenance des Automates et Instrumentation Industrielle.

Thème

SYSTEM FEEDEER ANIMALS

Présenté et soutenu publiquement par :

HOUARI TAHAR **&** **BELKBIR AYOUB**

Devant le jury composé de :

Nom et Prénom	Grade	Etablissement	Qualité
Mme. AISSANI Nassima	MCA	Univ. Oran-2, IMSI	**Présidente**
Mr. CHAREF Djilali	MCB	Univ. Oran-2, IMSI	**Encadreur**
Mr. BENFEKIR Abderrahim	MCB	Univ. Oran-2, IMSI	**Co-Encadreur**
Mr. MEKKI Ibrahim El Khalil	MCB	Univ. Oran-2, IMSI	**Examinateur**

Septembre 2024

Remerciement

Tout d'abord, nous remercions le Dieu Tout Puissant de mous avoir donné la force et la patience, et mous a guidé sur le droit chemin et mous a donné le courage et la force de continuer ce travail.

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce mémoire.

Tout d'abord, nous remercions sincèrement notre encadreur de mémoire ***DJILALI Charef,*** pour son encadrement précieux, ses conseils avisés et son soutien constant tout au long de ce projet. Ses orientations ont été essentielles à la réussite de ce travail. Et pour le deuxième encadreur BENFEKIR Abderrahim pour son soutien et sa présence avec nous après la mort de notre encadreur.

Nous souhaite également remercier nous enseignants et professeurs pour leur enseignement, qui a constitué les bases solides de nous compétences et connaissances, ainsi que pour leur disponibilité et leur soutien.

Enfin, nous voudrons exprimer notre profonde reconnaissance à notre famille et nous amis pour leur patience, leur soutien infaillible et leur encouragement tout au long de cette aventure académique. Leur présence nous a donné la force de persévérer et d'atteindre cet objectif.

À toutes et à tous, je vous dis un grand merci.

Dédicace

Avant tous, nous remercions dieu de tout puissant de nous avoir donnée le courage et la patience pour réaliser ce travail malgré toutes les difficultés rencontrées.

Avec l'expression de notre reconnaissance, on dédie ce modeste travail a ceux qui, quels que soient les termes embrasses, nous n'arriverons jamais à leur exprimer notre amour sincère.

A nous grande mère et nous parents. Grace à leurs tendres encouragements et leurs grands sacrifices, ils ont pu créer le climat affectueux et propice à la poursuite de nous études.

Aucune dédicace ne pourrait exprimer notre respect, notre considération et nos profonds sentiments envers eux.

Nous prions le bon dieu de les bénir, de veiller sur eux, en espérant qu'ils seront toujours fiers de nous.

A vous nous frère et sœurs qui n'ont pas cessé de nous conseiller, encourager et soutenir tout au long de nos études.

A la famille.

A tous nos professeurs.

A tous mos amis et mos collège.

Merci !

RESUME

Nous avons abordé dans ce travail un thème prenant assez d'ampleur dans notre vie quotidienne. On vous présentera le système d'alimentation intelligent (Animal Feeder System), est une solution technologique avancée conçue pour automatiser et optimiser le processus de nourrissage des animaux. Grâce à l'intégration de capteurs, de dispositifs de commande et des logiciels de gestion, ces systèmes permettent de distribuer la nourriture de manière précise, tout en surveillant la santé et les besoins nutritionnels spécifiques des animaux.

L'objectif de cette étude est d'évaluer la possibilité d'augmenter la productivité aussi aider les participants pour lever le fonds de ces travaille.

Mots clés : Animal Feeder System, Système intelligent multifonctionnel, ESP 32.

Abstract

In this work, we addressed a topic that is gaining significant importance in our daily lives. We present the Intelligent Feeding System (Animal Feeder System), which is an advanced technological solution designed to automate and optimize the animal feeding process. Through the integration of sensors, control devices, and management software, these systems enable precise food distribution while monitoring the health and specific nutritional needs of the animals.
The objective of this study is to assess the possibility of increasing productivity and to assist participants in raising funds for this project.
Keywords : Animal Feeder System, Multifunctional Intelligent System, ESP 32.

Voici la traduction de votre texte en arabe :

في هذا العمل تناولنا موضوعًا يكتسب أهمية كبيرة في حياتنا اليومية. سنقدم لكم نظام التغذية الذكي، وهو حل تقني متقدم مصمم لأتمته وتحسين عملية تغذية الحيوانات. بفضل دمج المستشعرات، وأجهزة التحكم، وبرامج الإدارة، تُمكِّن هذه الأنظمة من توزيع الطعام بدقة مع مراقبة صحة الحيوانات واحتياجاتها الغذائية الخاصة .

تهدف هذه الدراسة إلى تقييم إمكانية زيادة الإنتاجية ومساعدة المشاركين في جمع التمويل لهذا المشروع.

الكلمات المفتاحية: نظام تغذية الحيوانات، نظام ذكي متعدد الوظائف، ESP 3

Sommaire

Remerciements…… *i*
Dédicace …….. *ii*
Résumé……. *iii*
Sommaire …… *iv*
Liste des figures ……. *vi*
Liste des tableaux ……... *viii*
Liste des abréviations……. *ix*
Glossaire …… *x*

Introduction Générale ……... 12

Chapitre I - Généralité sur les animaux domestiques et leur système alimentaire

I.1 Introduction..... 15
I.2. L'origine de la domestication des animaux : 15
I.3. Généralités sur les aliments et les alimentations automatise 17
I.4. Système d'alimentation des animaux et sa standardisation 21
I.5.Contexte du projet..... 21
I.6. Standardisation du système..... 22
I.7. Standardisation :..... 23
I .8. Sous-système :..... 23
I.9. Conclusion 24

Chapitre II – Application de l'étude du projet à l'élevage de vaches

II.1. Introduction..... 26
II.2. Etude de Notre projet..... 26
II.2.1. Les trois system important dans notre projet..... 26
II.2.2. Le system de commande..... 27
II.2.3. Les systèmes de la partie opérative 30
II.3. Le rôle des pièces : 34
II.4. Système de supervision 36
II.5. Le fonctionnement de notre système 36
II.6. Stratégie de contrôle de notre projet..... 37
II.7. Limite de notre projet 38
II.8. Conclusion 39

Chapitre III - Prototype programmé à l'aide de l'ESP32

III.1. Introduction 44
III.2. Les composants utilisés et leurs rôles : 44
III.3. Câblages des composants 50
III.4. Programmation 56
III.4.1. Système d'alimentation 56
III.4.2. Système d'eau 60
III.4.3. System de température est humidité 64
III.4.4. System détection de flamme 66
III.4.5. L'application d'ESP 32 72
III.5. Discussion de prototype proposé 80
III.6. Synthèse des résultats 80
III.7. Conclusion 81

Conclusion général……82
Les recommandations…… 83
Bibliographiques……84

LISTE DE FIGURE

Figure1.1 : vache dans l'industrie 17
Figure1.2 : le processus de fabrication d'aliments pour animaux 18
Figure 1.3 : utilisation moyenne d'alimentation animale dans le monde 19
Figure1.4 : les sept dimensions de la qualite des aliments d'origine animale 20
Figure 1.5 : schema systeme d'alimentation intelligent multifonctionnel 24
Figure 2.1 : composant de raspberry pi 27
Figure 2.2 : image de cpu 28
Figure 2.3 : schema de cpu 28
Figure 2.4 : image de ram 28
Figure 2.5 : image de rom 28
Figure 2.6 : image de gpio 29
Figure 2.7 : image d'api 29
Figure 2.8 : schema d'api 30
Figure 2.9 : capture de poids 30
Figure 2.10 : capture de niveau 31
Figure 2.11 : verin electrique 31
Figure 2.12 : reservoir 32
Figure 2.13 : capture de position 32
Figure 2.14 : capture de niveau 33
Figure 2.15 : electrovanne 33
Figure 2.16 : reservoir 34
Figure 2.17 : cablage de systeme d'alimentation intelligent multifonctionnel 37
Figure 3.1 : esp 32 gpio pins 44
Figure 3.2 : plaque d'essai 45
Figure 3.3 : carte d'alimentation (mb102) 45
Figure 3.4 : capteur de temperature (dht 11) 45
Figure 3.5 : capteur de flamme (lm 393) 46
Figure 3.6 : electrovanne 46
Figure 3.7 : relais 47
Figure 3.8. : capteur ultrason 47
Figure 3.9 : capteur infrarouge 48
Figure 3.10 : capteur d'eau 48
Figure 3.11 : Servo-moteur 49
Figure 3.12 : capteur de poids 49
Figure 3.13 : cablage de l'electrovanne, relais et esp32 50
Figure 3.14 : cablage de servo-moteur au uln2003 et esp32 50
Figure 3.15 : cablage de capture ultrason (hc-sr04) 51
Figure 3.16 : cablage de capture de temperature (dht 11) 52
Figure 3.17 : cablage de capture de poids (hx711 adc) 53
Figure 3.18 : cablage de capture infrarouge (rees52) 53
Figure 3.19 : cablage de capture capteur de flamme 54

Figure 3.20 : programme de systeme d'alimentation 56
Figure 3.20-a : explication de programme de systeme d'alimentation 57
Figure 3.20-b : explication de programme de systeme d'alimentation 57
Figure 3.20-c : explication de programme de systeme d'alimentation 57
Figure 3.20-d : explication de programme de systeme d'alimentation 57
Figure 3.20-e : explication de programme de systeme d'alimentation 58
Figure 3.20-f : explication de programme de systeme d'alimentation 59
Figure 3.21 : programme de systeme d'eau 60
Figure 3.21-a : explication de programme de systeme d'eau 61
Figure 3.21-b : explication de programme de systeme d'eau 61
Figure 3.21-c : explication de programme de systeme d'eau 61
Figure 3.21-d : explication de programme de systeme d'eau 61
Figure 3.21-e : explication de programme de systeme d'eau 62
Figure 3.21-f : explication de programme de systeme d'eau 62
Figure 3.21-g : explication de programme de systeme d'eau 62
Figure 3.21-h : explication de programme de systeme d'eau 62
Figure 3.21-i : explication de programme de systeme d'eau 63
Figure 3.21-j : explication de programme de systeme d'eau 63
Figure 3.21-k : explication de programme de systeme d'eau 63
Figure 3.22 : programme de systeme de temperature et humidite 64
Figure 3.22-a : programme de systeme de temperature et humidite 64
Figure 3.22-b : programme de systeme de temperature et humidite 65
Figure 3.22-c : programme de systeme de temperature et humidite 65
Figure 3.22-d : programme de systeme de temperature et humidite 65
Figure 3.22-e : programme de systeme de temperature et humidite 65
Figure 3.22-f : programme de systeme de temperature et humidite 65
Figure 3.22-g : programme de systeme de temperature et humidite 66
Figure 3.22-h : programme de systeme de temperature et humidite 66
Figure 3.22-i : programme de systeme de temperature et humidite 66
Figure 3.23 : programme de systeme de detection de flamme 67
Figure 3.23-a : programme de systeme de detection de flamme 68
Figure 3.23-b : programme de systeme de detection de flamme 68
Figure 3.23-c : programme de systeme de detection de flamme 68
Figure 3.23-d : programme de systeme de detection de flamme 69
Figure 3.23-e : programme de systeme de detection de flamme 69
Figure 3.23-f : programme de systeme de detection de flamme 69
Figure 3.23-g : programme de systeme de detection de flamme 70
Figure 3.23-h : programme de systeme de detection de flamme 70
Figure 3.23-i : programme de systeme de detection de flamme 70
Figure 3.23-j : programme de systeme de detection de flamme 70
Figure 3.23-k : programme de systeme de detection de flamme 71
Figure 3.23-l : programme de systeme de detection de flamme 71
Figure 3.24 : reçoit de message de capteur de flamme sur mai 72
Figure 3.25-a : interface de l'application et conception de bouton water level 73

Figure 3.25-b : conception de bouton temperature, humidite et food weight 73
Figure 3.25-c : conception de bouton camera 74
Figure 3.26-a : programme d'application 74
Figure 3.26-b : programme d'application 75
Figure 3.26-c : programme d'application 75
Figure 3.26-d : programme d'application 76
Figure 3.26-e : programme d'application 76
Figure 3.26-f : programme d'application 77
Figure 3.26-g : programme d'application 77
Figure 3.26-h : programme d'application 78
Figure 3.26-i : programme d'application 78
Figure 3.26-j : programme d'application 79
Figure 3.26-k : programme d'application 79

LISTE DES TABLEAU

Table 3.1 : câblage des capteurs 55
Table 3.2 : câblage des actionneurs 55
Table 3.3 : câblage de l'électrovanne 55

Liste des abréviations

ESP 32 : Espressif Systems 32-bit

GPIO : Entrée/Sortie à Usage Général**.**

SMTP : Protocole Simple de Transfert de Courrier
GSM : Système Global de Communications Mobiles

CPU : Unité Centrale de Traitement

RAM : Mémoire à Accès Direct

ROM : Mémoire Morte.

TIA Portal : Portail d'Automatisation Totalement Intégrée

HTML : Language de Marquage Hypertexte

CSS : Feuillet de Style en Cascades

API : Automate Programmable Industriel

Glossaire

Microcontrôleur : un petit circuit intégré qui fonctionne comme un mini-ordinateur

Processeur dual-core : un processeur qui contient deux unités de calcul, appelées "cœurs", dans une seule puce.

Tensilica LX6 : un processeur de type microcontrôleur développé par Cadence, basé sur l'architecture Xtensa.

Un capteur optique : détecte des changements dans la lumière ou les propriétés optiques.

La chaleur excessive : températures qui dépassent des niveaux normaux

Fluide : une substance qui peut s'écouler et qui n'a pas de forme fixe

Bobine électromagnétique : dispositif constitué d'un fil conducteur enroulé en spirale autour d'un noyau

Ondes sonores vibrations qui se propagent à travers un milieu

Rayons infrarouges : rayonnement électromagnétique avec des longueurs d'onde plus longues que celles de la lumière visible

Rotor : est une partie tournante d'une machine ou d'un dispositif

Nano-ordinateur : ordinateur de très petite taille

Ethernet : technologie de réseau utilisée pour connecter des dispositifs au sein d'un réseau local

Volailles : désignent les oiseaux domestiques élevés principalement pour leur viande

Ruminant : animaux herbivores

Actionneur : un dispositif qui transforme un signal de commande en mouvement physique.

Ondes sonores : des vibrations qui se propagent à travers un milieu sous forme de variations de pression

Introduction générale

Les animaux et les aliments jouent un rôle fondamental dans le maintien des écosystèmes et dans la vie quotidienne des humains. Les interactions complexes entre les différentes espèces animales et végétales sont essentielles pour préserver l'équilibre naturel, assurant ainsi la survie de toutes les formes de vie sur Terre. Dans ce contexte, la conception de systèmes intelligents pour l'alimentation des animaux se situe à la croisée de l'innovation technologique et de la praticité, révolutionnant la manière dont les propriétaires prennent soin de leurs animaux.

Ces systèmes intelligents, grâce à l'intégration de technologies avancées, apportent commodité, efficacité et contrôle à distance, tout en améliorant le bien-être animal. Ils permettent non seulement d'automatiser le processus d'alimentation, mais aussi de surveiller la santé des animaux en temps réel, et de personnaliser leur régime alimentaire en fonction de leurs besoins spécifiques.

Dans cette optique, le développement d'un système d'alimentation intelligent multifonctionnel constitue une avancée majeure dans la gestion quotidienne des animaux, s'adaptant à diverses espèces et à leurs exigences nutritionnelles. Ce projet met en avant l'importance de la standardisation dans la conception de tels systèmes pour garantir cohérence, interopérabilité et évolutivité, tout en facilitant leur adoption. L'ajout de sous-systèmes comme des caméras de surveillance ou des systèmes de nettoyage automatisé offre une solution complète pour le bien-être et la gestion des animaux.

L'automatisation des processus agricoles est de plus en plus nécessaire pour accroître l'efficacité, optimiser la gestion des ressources, et assurer le bien-être animal. Ce projet, axé sur le développement d'un système d'alimentation intelligent pour les vaches, illustre l'importance de ces technologies dans la gestion des fermes modernes. En combinant capteurs, actionneurs et unités de commande automatisées, ce système garantit une distribution efficace et précise de la nourriture et de l'eau, réduisant ainsi l'intervention humaine.

Basé sur l'utilisation d'un Raspberry Pi et d'un automate programmable industriel (API), ce système assure un contrôle automatique de l'alimentation et de l'hydratation des animaux, chaque composant jouant un rôle essentiel dans le bon fonctionnement global. L'objectif de ce projet est de démontrer l'importance et l'efficacité de l'automatisation dans le secteur de l'élevage, tout en améliorant les pratiques agricoles et le suivi des animaux.

Ce travail est organisé de la façon suivante :

Dans **Le chapitre 1** nous présenterons de façon générale l'importance de l'animal dans le maintien des écosystèmes et dans la vie quotidienne des humains. Ainsi que tous types d'aliment et la standardisation de notre système et tous c'est bénéfice.

Le chapitre 2 : nous présentons une application de l'étude du projet à l'élevage de vaches

Où la réalisation de projet sur le terrain réel en utilisant une vache comme exemple et tous les composants utilisés et comment sa fonction.

Le chapitre 3 : un prototype de la fonctionnalité de system, on donnant les pièces utilisé et leur rôle et comment faire le câblage et tout programme des pièces, à la fin on a réalisé une application pour le consommateur pour suivi son système.

Et nous nous finalisons par une conclusion générale de ce travail.

Chapitre I - Généralité sur les animaux domestiques et leur système alimentaire

I.1. Introduction :

Les animaux et les aliments sont essentiels au fonctionnement des écosystèmes, où ils interagissent de manière complexe pour maintenir l'équilibre naturel. Les animaux, en tant qu'herbivores, carnivores ou omnivores, influencent la répartition des plantes et la structure des communautés biologiques. Les herbivores régulent la croissance des plantes, tandis que les carnivores contrôlent les populations d'herbivores. Les chaînes alimentaires, qui montrent comment l'énergie et les nutriments se déplacent à travers différents organismes, sont au cœur de ces interactions. Ainsi, chaque élément joue un rôle crucial dans le maintien de la santé de l'écosystème, et une perturbation dans cette dynamique peut avoir des effets significatifs sur l'ensemble du système.

I.2 L'origine de la domestication des animaux : [5]

La domestication des animaux d'intérêt alimentaire est apparue très tard dans l'histoire de l'humanité : il y a 10 000 ans au Proche-Orient (porc, bœuf, mouton et chèvre), 9 000 ans en Chine (porc) et dans l'Indus (zébu), 7 000 à 6 500 ans dans les Andes (lama, alpaca) et l'Asie du Sud-est (poule), 5 500 ans en Asie centrale (cheval, chameau). Ce tournant majeur dans l'histoire de l'humanité, nommé « transition néolithique », se caractérise par d'importants changements alimentaires, mais aussi et surtout par l'émergence d'un attachement aux territoires, le développement de nouvelles techniques pour produire et transformer les aliments, l'accumulation de réserves voire de richesses, de profondes modifications de l'organisation sociale et l'émergence de nouvelles croyances.

a. Objectif d'alimentations animales :

Pour vivre et se développer, un animal doit s'alimenter. Le régime alimentaire d'un animal correspond à l'ensemble des aliments qu'il prélève dans son milieu. Le plus souvent, les animaux appartenant à une même espèce ont le même régime alimentaire, mais en fonction des espèces, les régimes rencontrés sont très divers.

On distingue trois principaux types de régimes alimentaires : les herbivores (animaux qui se nourrissent uniquement de végétaux), les carnivores (animaux qui se nourrissent uniquement d'autres animaux) et les omnivores (animaux qui mangent des végétaux et des animaux)

b. Le besoin des animaux :

Les animaux rendent plusieurs services aux humains tels que : être de la compagnie, apporter de la sécurité, être une source de nourriture, aider au recyclage des matières mortes, être pollinisateur, favoriser le bien-être des êtres humains.

Animal de compagne : Un animal permet d'apporter de la joie et de la bonne humeur. Il est même reconnu qu'avoir un animal domestique diminue l'anxiété et atténue la solitude. C'est aussi pour cette raison qu'ils sont sollicités, notamment dans les établissements médicosociaux, pour apporter du réconfort aux personnes âgées et seules.

L'alimentation animale joue un rôle central dans l'industrie alimentaire mondiale ; c'est un élément-clé pour assurer la production durable de protéines animales sûres et abordables, les animaux sont une source importante de nourriture pour les humains. Ils fournissent de la viande, du lait, des œufs et d'autres produits alimentaires qui sont essentiels pour nourrir de la population. Chaque espèce animale est utile et participe à l'équilibre des différents écosystèmes. Les animaux vivant dans les forêts protègent ces derniers et leur permettent de se développer. Chaque écosystème présent sur terre à besoin des animaux et la biodiversité permet aux humains de survivre.

Figure1.1 : Vache dans l'industrie [1]

I.3. Généralités sur les aliments et les alimentations automatise :

Un aliment est une substance ingérée par les animaux et apportant tout ce qui est nécessaire comme énergie et nutriments à leur ration pour couvrir les besoins de l'animal pour son entretien et pour sa production.

La disponibilité du fourrage tout au long de la journée a un impact direct sur la santé générale des vaches et sur la production laitière. Pour mélanger les bonnes portions dans les bonnes quantités et les distribuer au bon moment, il faut des connaissances, de l'expérience et du temps. L’alimentation automatisée ainsi que

d'autres équipements innovants prennent en charge l'alimentation quotidienne du troupeau, en assurant une efficacité optimale, tout en améliorant la santé animale.

a. Fabrication des aliments :

Le processus de production d'une meunerie d'aliments comporte plusieurs étapes critiques, notamment la sélection de la formulation des aliments, le broyage, le mélange, la granulation, le refroidissement, l'emballage et le contrôle qualité. L'utilisation du bon équipement et le respect des meilleures pratiques garantissent la production d'aliments de haute qualité qui répondent aux besoins nutritionnels des animaux.

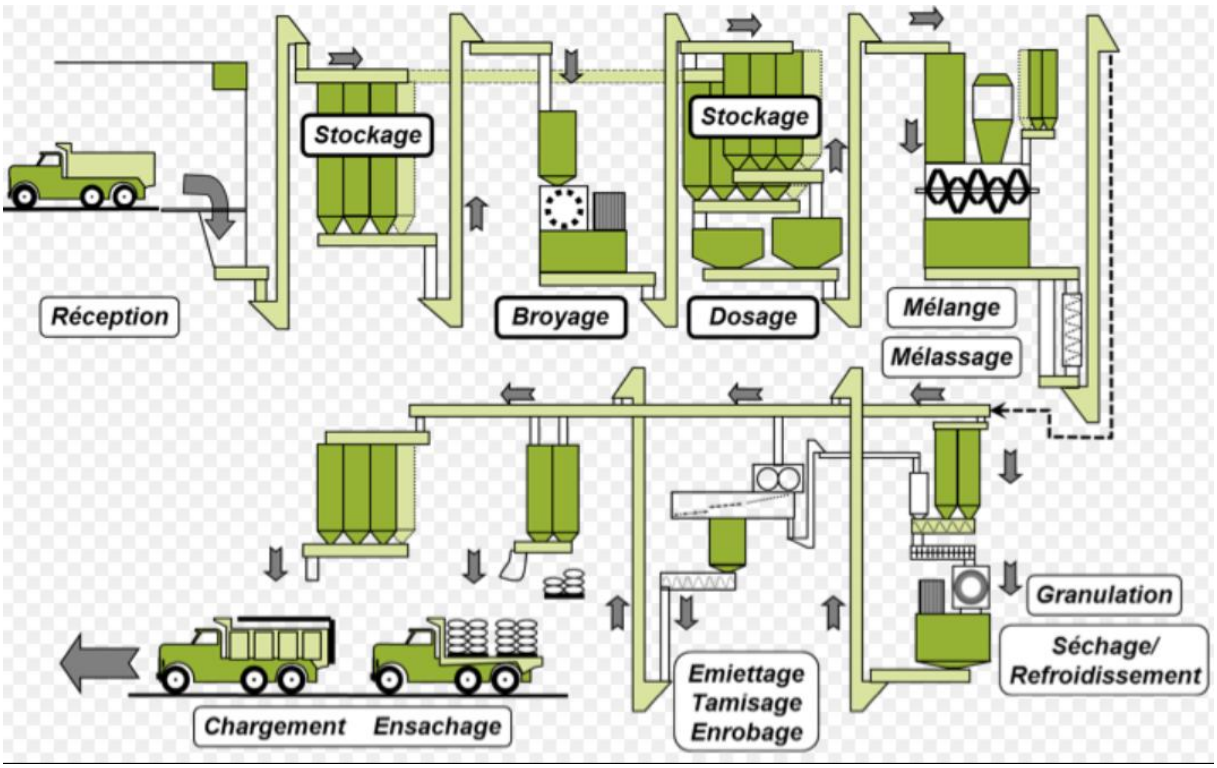


Figure 1.2 : le processus de fabrication d'aliments pour animaux [2]

b. Type d'aliments :

Nous vous présentons les 7 groupes d'aliments à contrôler dans votre alimentation

- Fruits et légumes.
- Produits céréaliers et légumes secs (légumineuses) Produits laitiers.
- Viande, poisson et fruits de mer.
- Matières grasses.

- Produits sucrés.
- Sel.
- Eau.

c. Les aliments pour animaux :

Les animaux doivent trouver dans leur alimentation des apports quotidiens en énergie, en protéines, en vitamines, en minéraux et en fibres végétales. Ils les trouvent dans les aliments composés où les différentes matières premières sont assemblées en fonction de ce qu'elles apportent dans un dosage équilibré. Pour tous les animaux d'élevage, les céréales constituent la base énergétique de la ration alimentaire. Elles représentent en moyenne 40 à 50% des matières premières mises en Œuvre dans les aliments composés

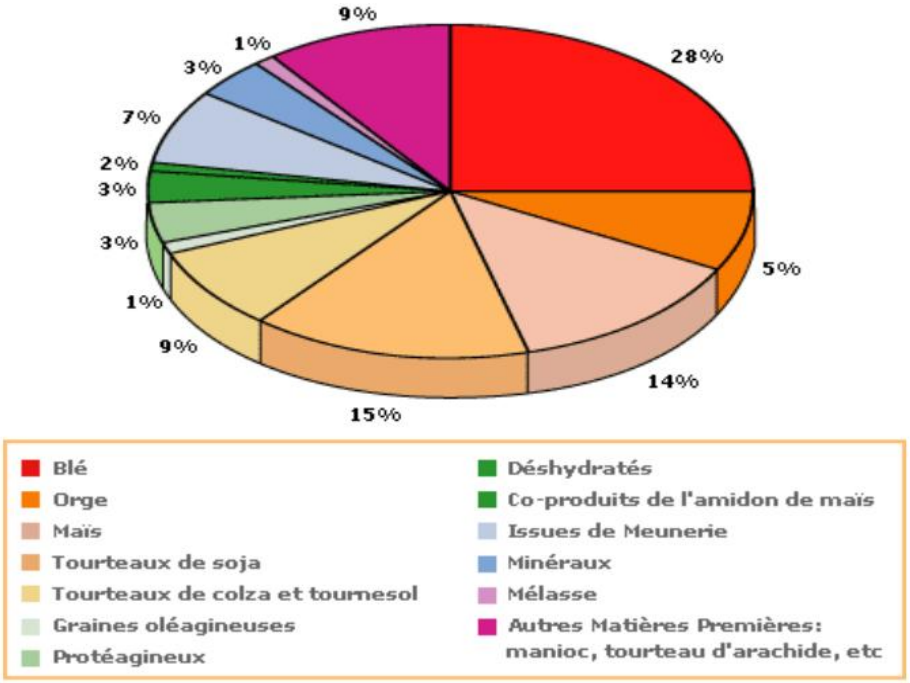


Figure 1.3 : utilisation moyenne d'alimentation animale dans le monde [3]

d. Risque lié à l'alimentation animale : [6]

Contamination par des substances dangereuses (résidus de médicaments, pesticides, mycotoxines).

Utilisation de sous-produits animaux, comme les farines animales, pouvant causer des maladies graves (ESB).

Additifs alimentaires aux effets néfastes sur la santé animale.

Pollution environnementale liée à la production de cultures pour l'alimentation animale.

Transmission de maladies via l'alimentation contaminée.

Alimentation déséquilibrée, entraînant des problèmes de santé animale.

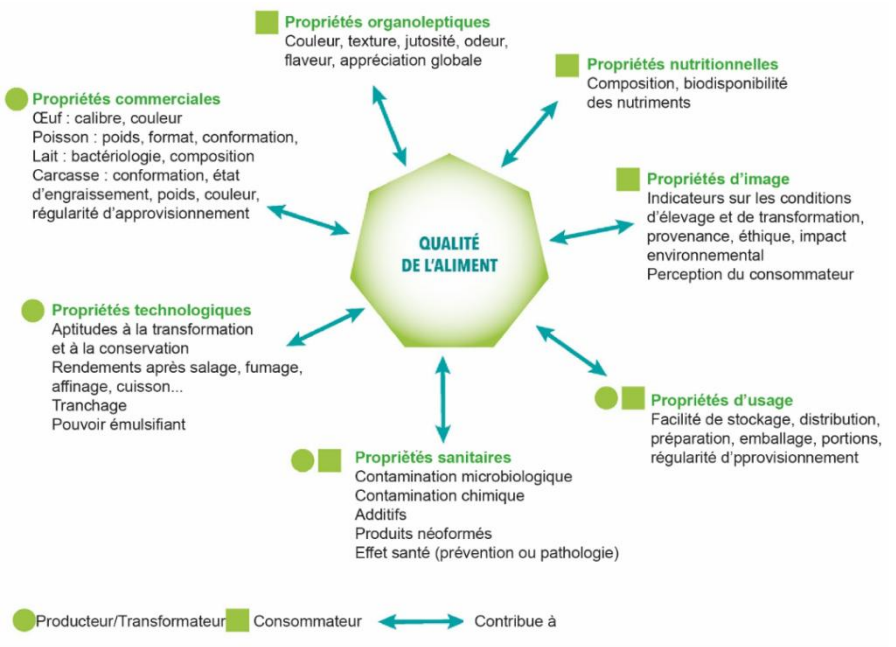


Figure 1.4 : Les sept dimensions de la qualité des aliments d'origine animale [4]

e. Transformation dans l'industrie des aliments :

La préparation et la transformation alimentaires peuvent être définies comme étant « tout changement qui modifie la qualité gustative ou la durée de conservation d'un aliment ». Les transformateurs de produits alimentaires doivent fabriquer des produits salubres pour éviter tout risque aux consommateurs.

Les méthodes de transformation des aliments pour animaux ?

Méthodes de traitement humide : Cela comprend : le broyage, laminage à sec, effritement.

f. Recyclage dans l'industrie des aliments :

Deux solutions existent : le compostage et la méthanisation

*** Le compostage** : la décomposition des déchets alimentaires permet de fabriquer de l'engrais naturel pour les sols de jardins et terrains agricole. Le retour

à la terre des ordures organiques fertilise les sols et optimise donc la pousse des végétaux.

***La méthanisation** : ce processus vise à extraire le méthane issu de la fermentation des produits organiques. Le biogaz extrait sert généralement à la production d'électricité, de chaleur ou de carburant.

I .4. Système d'alimentation des animaux et sa standardisation :

La conception allie à la fois innovation et praticité pour révolutionner la façon dont nous prenons soin de nos animaux. Ce système intègre une technologie de pointe avec une conception intuitive pour offrir commodité, efficacité et tranquillité d'esprit aux propriétaires d'animaux.

À la base, le système d'alimentation intelligent multifonctionnel est conçu pour rationaliser le processus d'alimentation des animaux tout en offrant une gamme de fonctionnalités supplémentaires pour améliorer leur bien-être.

Dans ce chapitre, on parlera des éléments standards de notre système et ses multiples fonctionnalités. On verra aussi comment est-ce qu'on pourra l'utiliser afin d'alimenter toute sorte d'animaux. Et bien plus que ça, on peut ajouter lui plusieurs sous-système.

I .5. Contexte du projet :

La conception d'un système d'alimentation intelligent multifonctionnel a pour objectif de révolutionner la façon dont nous prenons soin de nos animaux en intégrant de manière transparente une technologie de pointe à des fonctionnalités pratiques. Voici plusieurs objectifs clés de notre système :

a. **Commodité :** l'objectif principal d'un système d'alimentation intelligent est d'offrir une commodité aux propriétaires d'animaux en automatisant le processus d'alimentation. Grâce à des horaires d'alimentation programmés et à un contrôle précis des portions, les propriétaires d'animaux peuvent s'assurer que leurs animaux reçoivent des repas réguliers même lorsqu'ils sont loin de chez eux.

b. **Surveillance de la santé :** un autre objectif important est de promouvoir la santé et le bien-être des animaux. Les systèmes d'alimentation intelligents incluent souvent des fonctionnalités telles que le suivi des repas et la surveillance de la consommation, permettant aux propriétaires d'animaux de suivre les habitudes alimentaires de leurs animaux et de détecter tout changement pouvant indiquer des problèmes de santé.

c. **Personnalisation :** les systèmes d'alimentation intelligents offrent des programmes d'alimentation et des tailles de portions personnalisables, répondant aux besoins et préférences alimentaires spécifiques de chaque animal. Cette personnalisation garantit que les animaux reçoivent la bonne quantité de nourriture au bon moment, favorisant ainsi une nutrition et une gestion du poids optimal.

d. **Accès à distance :** La possibilité de surveiller et de contrôler à distance la mangeoire via des applications pour téléphone ou des interfaces Web est un objectif clé des systèmes de mangeoires intelligents. Cette fonctionnalité permet aux propriétaires d'animaux de vérifier l'état d'alimentation de leurs animaux, d'ajuster les horaires d'alimentation et de recevoir des notifications de n'importe où, offrant ainsi une tranquillité d'esprit et une flexibilité.

e. **Intégration aux écosystèmes de la maison intelligente :** les systèmes de mangeoires intelligents s'intègrent souvent à d'autres appareils et écosystèmes de la maison intelligente, permettant une automatisation et un contrôle transparents. L'intégration avec les assistants de caméra, les systèmes de sécurité domestique et d'autres appareils intelligents améliore la commodité et améliore l'expérience globale de soins aux animaux.

f.**Sûreté et sécurité :** Assurer la sécurité et la sûreté des aliments pour animaux est un autre objectif essentiel des systèmes de mangeoires intelligents. Des fonctionnalités telles que des mécanismes de verrouillage sécurisés, des systèmes antiblocages et des options d'alimentation de secours aident à empêcher l'accès non autorisé à la nourriture et à atténuer les perturbations potentielles.

Dans l'ensemble, l'objectif de la conception d'un système de mangeoire intelligent multifonctionnel est d'améliorer la qualité des soins aux animaux en offrant une commodité, en favorisant la santé et la nutrition et en s'intégrant parfaitement aux modes de vie modernes. En tirant parti de la technologie pour simplifier les routines d'alimentation et surveiller le bien-être des animaux, ces systèmes contribuent au bonheur et à la longévité des animaux et de leurs propriétaires.

I .6. Standardisation du système :

Le développement d'une conception standard pour un système de mangeoire intelligent multifonctionnel sert plusieurs objectifs importants :

a. **Cohérence :** une conception standard garantit la cohérence des fonctionnalités, de la facilité d'utilisation et des performances des systèmes de mangeoires intelligents pour différents animaux (vaches, moutons, chats, chiens…). Cette cohérence permet aux éleveurs d'animaux d'avoir des attentes claires et facilite la prise en charge des différents animaux.

b. **Interopérabilité :** la standardisation de la conception facilite l'interopérabilité entre les systèmes de mangeoires intelligents et d'autres appareils ou plateformes de maison intelligente. Cela signifie que les éleveurs d'animaux peuvent intégrer leur mangeoire intelligente de manière transparente à d'autres appareils tels que les Smartphones, les assistants photo ou les systèmes domotiques, améliorant ainsi l'expérience globale de l'utilisateur.

c. **Facilité d'adoption :** une conception standard simplifie l'adoption et la mise en œuvre de systèmes de mangeoires intelligents pour les fabricants et les éleveurs d'animaux. Les fabricants peuvent suivre les directives et les meilleures pratiques établies, rationalisant ainsi le processus de développement et réduisant les coûts. Les consommateurs, quant à eux,

bénéficient d'interfaces intuitives et d'expériences utilisateur cohérentes, ce qui facilite la configuration et l'utilisation du système d'alimentation intelligent.

d. **Évolutivité :** la standardisation de la conception permet l'évolutivité et la production en masse des systèmes d'alimentation intelligents, réduisant ainsi les coûts de production et les rendant plus accessibles à un public plus large. Cette évolutivité encourage également l'innovation et la concurrence sur le marché, ce qui conduit à des améliorations continues des fonctionnalités et des performances.

e. **Conformité réglementaire :** les conceptions standard s'alignent souvent sur les exigences réglementaires et les normes de l'industrie, garantissant que les systèmes d'alimentation intelligents répondent aux normes de sécurité, de qualité et d'interopérabilité. Cette conformité renforce la confiance des consommateurs et des régulateurs, favorisant une réputation positive pour l'industrie dans son ensemble.

f. **Assistance et maintenance :** une conception standard facilite l'assistance et la maintenance des systèmes d'alimentation intelligents. Les fabricants peuvent développer des procédures de dépannage standardisées et offrir une assistance client cohérente, améliorant ainsi la fiabilité et la longévité globales du produit.

En résumé, la création d'une conception standard pour les systèmes d'alimentation intelligents multifonctionnels favorise la cohérence, l'interopérabilité, la facilité d'adoption, l'évolutivité, la conformité réglementaire et la prise en charge, profitant en fin de compte à la fois aux fabricants et aux acheteurs.

I .7. Standardisation :

Il est notable que tout système d'alimentation existant des animaux est destiné à une seule espèce d'animaux.

Et c'est pour cela, dans notre système, on a opté à ajouter l'intelligence artificielle à données mieux améliorées pour que ce soit adapté à toute espèce que nous mettrons d'animaux ; donc en modifiant seulement des paramètres comme l'heure de manger , la quantité d'alimentation et les type d'aliment ce serait compatible à l'animal en question.

I .8. Sous-système :

Dans notre projet, on a parlé sur l'alimentation de notre animal mais n'empêche qu'on peut lui associer d'autres systèmes pour vous faciliter la vie comme :

a. **Système de caméra de surveillance :** à l'aide de l'informatique maintenant en peut programmer des caméras pour être plus fonctionnel, sert à la détection de mouvements et la mesure de la température.

b. **Système de nettoyage :** avec ce système le nettoyage se ferait seul, automatiquement et on n'aura plus besoin de l'être humain, ça va même permettre d'économiser de l'argent.

c. **WIFI :** ce système est utilisé pour être à jour avec votre animal et vos tâches. A l'aide du WIFI, on peut recevoir des notifications sur le téléphone, nous renseignant sur l'état de

l'animal et du système ; détection de mouvement anormal de l'animal ou une hausse de température (qui sera mesurée à l'aide de CPU), l'état des capteurs et les actionneurs du système : s'il y'a une pièce défaillante….dans l'unique but de s'informer à temps pour mieux et rapidement agir.

d. **GSM :** Le GSM est un système dont on peut s'en passer car c'est payant donc à sa place on utilise le wifi.

Et voici un schéma de notre système qui vous facilitera la vie à vous et à vos animaux.

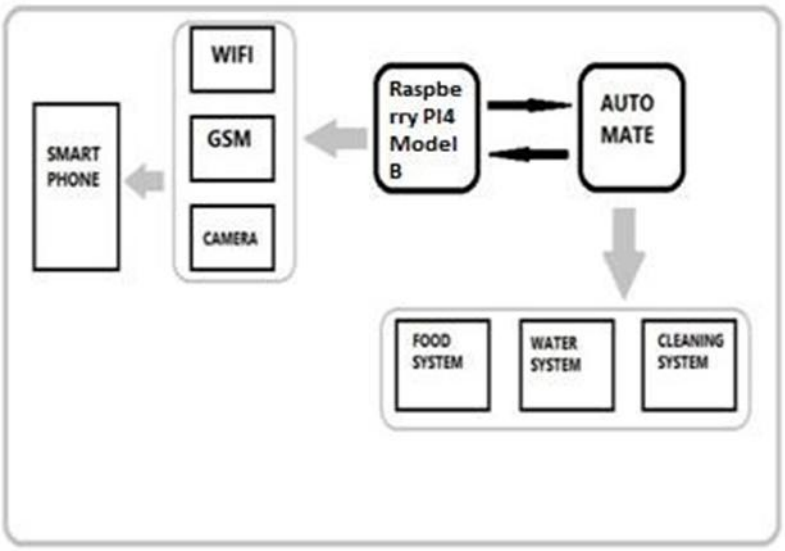


Figure 1.5 : Schéma système d'alimentation intelligent multifonctionnel

I.9. Conclusion :

La relation entre les animaux, leur alimentation et les écosystèmes est fondamentale pour le maintien de l'équilibre naturel et de la biodiversité. À travers l'histoire, la domestication des animaux a marqué une étape cruciale dans l'évolution humaine, influençant profondément les pratiques alimentaires, sociales et économiques. Aujourd'hui, avec l'intégration de technologies avancées telles que les systèmes d'alimentation intelligents, nous avons la capacité d'améliorer non seulement la gestion et le bien-être des animaux, mais aussi d'optimiser la production alimentaire de manière durable et efficace. Ces systèmes offrent commodité, personnalisation et surveillance à distance, tout en respectant les besoins des animaux et en contribuant à la sécurité alimentaire. La standardisation de ces systèmes, combinée à des sous-systèmes intelligents, promet une adoption plus large et une meilleure gestion des ressources dans le futur.

Chapitre II – Application de l’étude du projet à l’élevage de vaches

II.1. Introduction :

Dans ce chapitre, nous allons procéder à une étude détaillée du projet. Nous présenterons les systèmes impliqués ainsi que tous les composants constituent le projet qui sont utilisées.

II.2. Etude de Notre projet :

Après avoir mené des recherches approfondies dans l'industrie de l'alimentation animale, il est apparu que la majorité des systèmes existants sont spécifiquement conçus pour un seul type d'animal. Cette spécialisation limite leur efficacité et leur flexibilité. Dans le cadre de notre projet, nous visons à développer un système standardisé capable de s'adapter à différents types d'animaux, qu'il s'agisse de chiens, de chats, de volailles, ou même de ruminants.

Notre solution repose sur l'utilisation d'un Raspberry Pi, qui servira de plateforme centrale pour le contrôle et la gestion du système. Grâce à une interface conviviale et intuitive, l'utilisateur pourra, d'un simple clic, sélectionner l'animal concerné et ajuster les paramètres d'alimentation en conséquence.

En outre, nous intégrerons des API (interfaces de programmation d'application) pour permettre la communication entre le Raspberry Pi et d'autres appareils ou services, comme des capteurs de poids ou des applications mobiles. Cela garantira une gestion personnalisée et efficace de l'alimentation des animaux, tout en collectant des données précieuses pour optimiser leur nutrition.

II.2.1. Les trois system clés de notre projet

a. **Système de Commande :**

Considéré comme le cœur et le cerveau de l'ensemble du dispositif, ce système est chargé de coordonner et de superviser toutes les structures nécessaires pour le bien-être de notre vache. Grâce à des algorithmes intelligents et à des interfaces conviviales, il permet une gestion efficace des différentes fonctionnalités, assurant ainsi un fonctionnement harmonieux et réactif aux besoins de l'animal.

b. **Système d'Alimentation :**

Ce système est essentiel pour garantir que notre vache reçoit ses repas de manière régulière et appropriée. Il est conçu pour distribuer la nourriture en quantités contrôlées, en tenant compte des besoins nutritionnels spécifiques de l'animal. Ce système intègre des capteurs pour surveiller les niveaux de nourriture et peut être programmé pour adapter les portions en fonction de l'âge, du poids et des conditions de santé de la vache. Cela permet non seulement de prévenir le gaspillage, mais aussi d'optimiser la santé et la croissance de l'animal.

c. **Système d'Eau :**

Ce système est conçu pour fournir de l'eau fraîche à notre vache, en réagissant à ses besoins en hydratation. Équipé de capteurs de détection, il identifie quand l'animal s'approche

et active le mécanisme de distribution d'eau. Cela garantit que la vache a toujours accès à une source d'eau propre, essentielle pour sa santé. Ce système peut également inclure des fonctionnalités de surveillance pour s'assurer que les niveaux d'eau sont adéquats à tout moment.

II.2.2. Le system de commande :

a. Raspberry Pi :

Le Raspberry Pi 4 B est un nano-ordinateur pouvant se connecter à un moniteur, à un ensemble clavier/souris et disposant d'interfaces Wi-Fi, Bluetooth et Ethernet. Il fonctionne depuis une carte micro-SD et fonctionne avec un système d'exploitation basé sur Linux ou Windows 10 IoT.

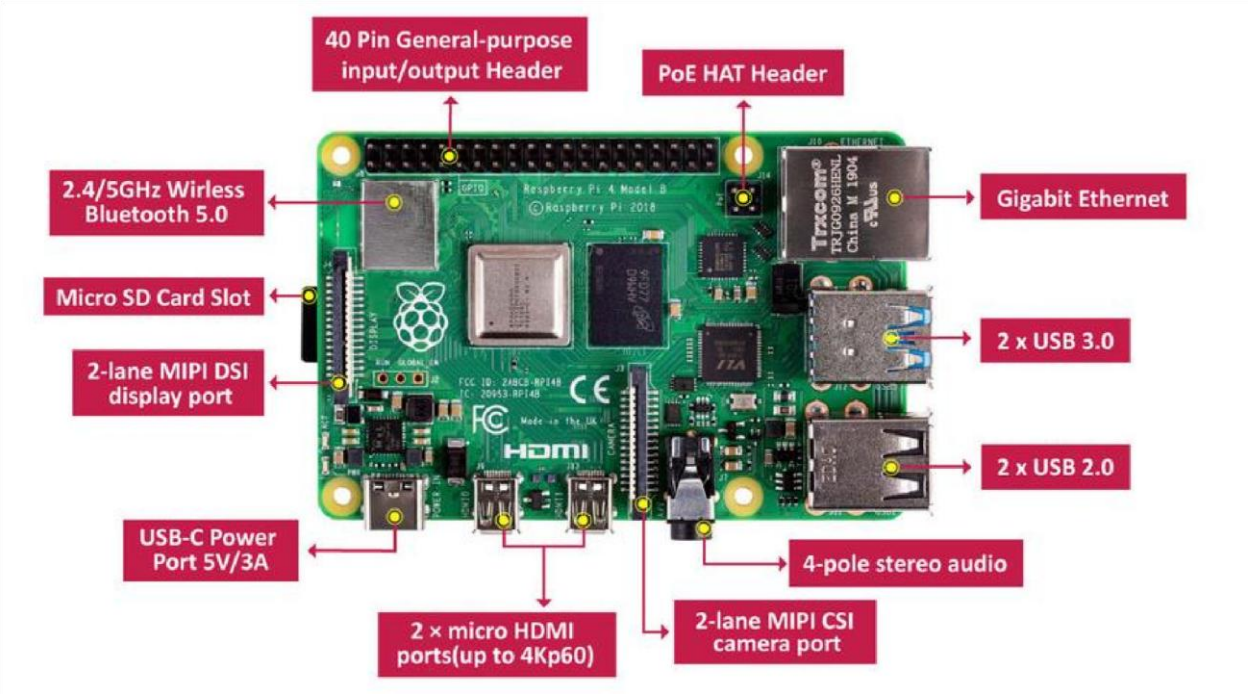


Figure 2.1 : Composant de raspberry pi [7]

Composant de raspberry pi :

a. Le CPU :

Un processeur (CPU) est un composant matériel qui constitue l'unité centrale de traitement d'un serveur. Les serveurs et autres appareils intelligents convertissent les données en signaux numériques et y effectuent des opérations mathématiques. Le CPU est le principal composant qui traite les signaux et rend le calcul possible. C'est le cerveau de tout appareil informatique. Il extrait les instructions de la mémoire, exécute les tâches requises et renvoie les résultats en mémoire. Il gère toutes les tâches informatiques nécessaires au fonctionnement du système d'exploitation et des applications.

Figure 2.2 : Image de cpu

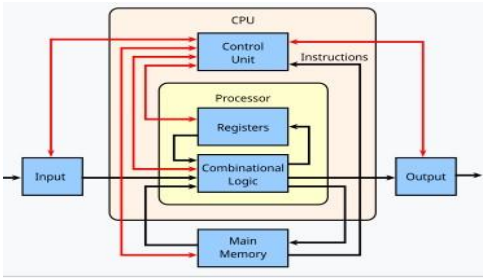


Figure 2.3 : Schéma de cpu

b. La ram :

La RAM est l'abréviation, en anglais, de Radom Access Memory (mémoire vive). C'est l'un des éléments fondamentaux de l'informatique. La RAM est une banque de mémoire temporaire où votre ordinateur stocke les données qu'il doit retrouver rapidement.

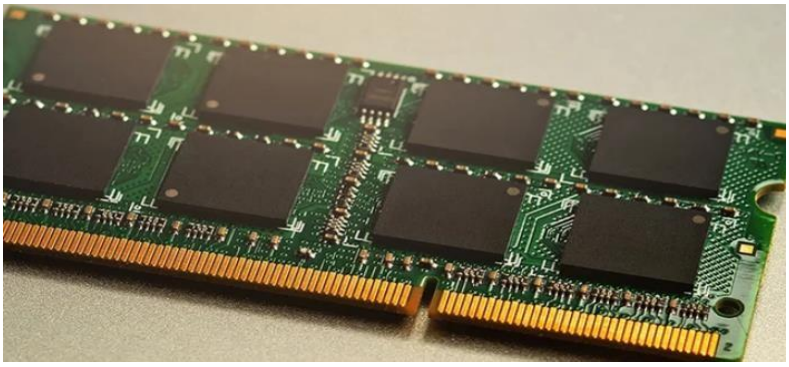


Figure 2.4 : Image de ram

c. La rom :

La mémoire morte, appelée ROM (Read Only Memory), contient des données qui ne peuvent être modifiées par l'utilisateur

Figure 2.5 : Image de rom

d. GPIO :

Les ports GPIO (anglais : General Purpose Input/Output, littéralement Entrée-sortie à usage général) sont des ports d'entrées-sorties très utilisés dans le monde des microcontrôleurs, en particulier dans le domaine de l'électronique embarquée.

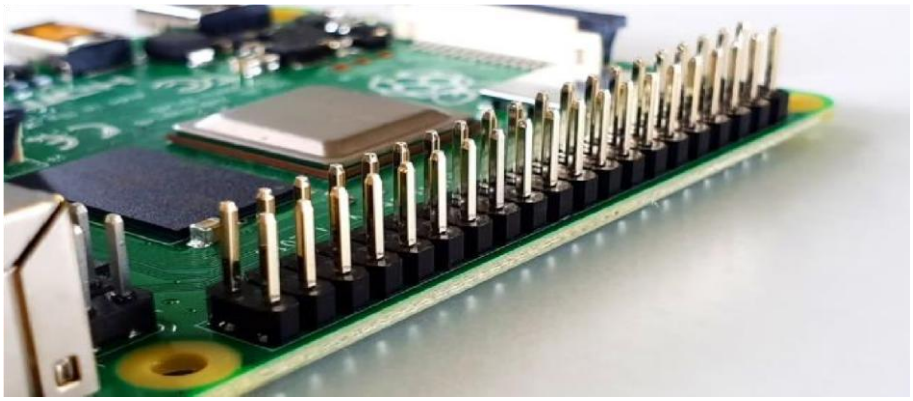


Figure 2.6 : image de gpio

API :

Un automate programmable industriel, ou API (en anglais programmable logique contrôler, PLC), est un dispositif électronique numérique programmable destiné à la commande de processus industriels par un traitement séquentiel. Il envoie des ordres vers les prés actionneurs (partie opérative ou PO côté actionneur) à partir de données d’entrées (capteurs) (partie commande ou PC côté capteur), de consignes et d’un programme informatique.

Figure 2.7 : image d’api

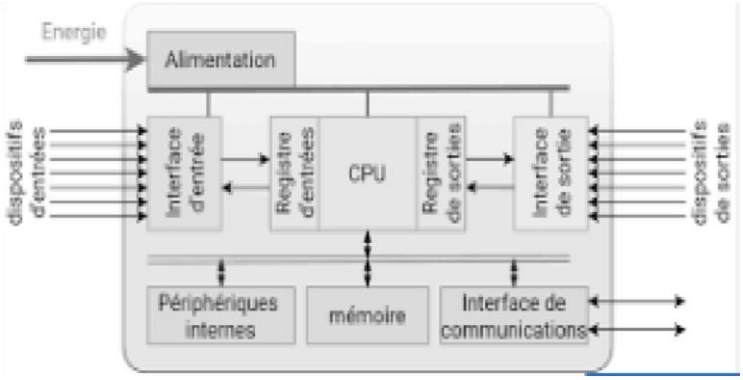


Figure 2.8 : Schéma d'api

II.2.3. Les systèmes de la partie opérative

1. Système d'alimentation :

a. Les capteurs :

Un capteur est un dispositif permettant d'interpréter l'état d'une grandeur physique observée en une grandeur exploitable, telle qu'une tension électrique, une hauteur de mercure, un courant électrique ou la déviation d'une aiguille.
Pour ce système on utilise capteur de poids, capteur de niveau :

i. Capteur de poids :

Un Capteur de Pesage ou un capteur de force est un dispositif très précis utilisé pour mesurer le poids ou la force dans un certain nombre d'applications différentes. Ils peuvent être utilisés pour mesurer les forces de compression, de tension, de flexion ou de cisaillement

Figure 2.9 : Capture de poids [8]

ii. Capteur de niveau :

Un capteur de niveau est un dispositif électronique qui permet de mesurer la hauteur du matériau, dans un réservoir ou un autre récipient.

Figure 2.10 : Capture de niveau [9]

Les actionneurs :

Dans une machine, un actionneur est un objet qui transforme l’énergie qui lui est fournie en un phénomène physique qui fournit un travail, modifie le comportement ou l’état d'un système[1].

Pour ce système on utilise un vérin électrique : Un vérin électrique est un dispositif qui convertit le mouvement rotatif d'un moteur en mouvement linéaire. Il peut être intégré dans n'importe quel équipement pour pousser, tirer, lever, baisser, positionner ou tourner une charge.

Figure 2.11 : vérin électrique [10]

b. Le réservoir :

Bassin naturel ou artificiel ; récipient d'une machine ou d'une installation où s'accumule une matière, pour être mis en réserve.

Figure 2.12 : Réservoir

2. Système d'eau :

a. Les capteurs :

Un capteur est un dispositif permettant d'interpréter l'état d'une grandeur physique observée en une grandeur exploitable, telle qu'une tension électrique, une hauteur de mercure, un courant électrique ou la déviation d'une aiguille.
Pour ce system on utilise capteur de position, capteur de poids, capteur de niveau :

i. Capteur de position :

Le capteur de position est un dispositif qui permet de recueillir des informations sur la position précise et le mouvement d'un objet. Les informations peuvent être obtenues par contact direct avec l'objet ou à distance à l'aide des capteurs magnétiques. Il permet également de détecter l'épaisseur d'une pièce, le niveau d'un fluide ou encore l'angle de rotation d'un arbre.

Figure 2.13 : Capture de position [11]

ii. Capteur de poids : revenir à (II.2.3.1.a.i)

iii. Capteur de niveau :

Un capteur de niveau est un dispositif électronique qui permet de mesurer la hauteur du matériau, dans un réservoir ou un autre récipient.

Figure 2.14 : capture de niveau [12]

b. Les actionneurs :

Dans une machine, un actionneur est un objet qui transforme l’énergie qui lui est fournie en un phénomène physique qui fournit un travail, modifie le comportement ou l’état d'un système[1].

Pour ce système on utilise électrovanne : Une électrovanne ou électrovalve est une vanne commandée électriquement. Grâce à cet organe, il est possible d'agir sur le débit d'un fluide dans un circuit par un signal électrique.

Figure 2.15 : électrovanne [13]

C. Le réservoir :

Bassin naturel ou artificiel ; récipient d'une machine ou d'une installation où s'accumule une matière, pour être mis en réserve.

Figure 2.16 : réservoir

II.3. Le rôle des pièces :

a. La partie commande :

La partie commande contrôle deux sections principales : la partie alimentation et la partie eau, grâce à une Raspberry Pi et un API (Automate Programmable Industriel).

i. L'utilisation de l'API :

L'API joue un rôle central en exécutant les programmes développés avec TIA Portal pour contrôler l'ensemble des actionneurs et capteurs du système. Il est responsible de la gestion :

- Des actionneurs : comme l'électrovanne et le vérin électrique.
- Des capteurs : comme les capteurs de position, de poids et de niveau.

ii. L'utilisation de la Raspberry Pi :

La Raspberry Pi est la carte principale qui héberge les composants électroniques tels que le CPU, la RAM, la ROM, ainsi que les modules Wi-Fi et Bluetooth.

CPU (Central Processing Unit) : Il s'agit du processeur central qui effectue tous les calculs nécessaires. Il reçoit les données des capteurs, les traite, et envoie les instructions appropriées aux actionneurs.

- RAM (Random Access Memory) : Mémoire vive qui stocke temporairement les informations nécessaires pour exécuter les programmes. Le CPU interagit constamment avec la RAM pour réaliser les tâches en temps réel.
- ROM (Read Only Memory) : Mémoire de stockage permanente qui contient le programme à exécuter, même après la mise hors tension.
- **Wi-Fi** : Utilisé pour connecter la Raspberry Pi au réseau, facilitant ainsi la communication avec un téléphone via l'ESP32 pour la supervision à distance.

b. La partie alimentation :

Cette partie est dédiée à la gestion des aliments dans le système.

i. Les capteurs :

- **Capteur de poids** : Mesure le poids des aliments qui descendent du réservoir. Lorsqu'une certaine quantité est atteinte, il envoie un signal à l'API pour déclencher la fermeture du vérin électrique, stoppant ainsi l'alimentation.
- **Capteur de niveau** : Détecte le niveau des aliments dans le réservoir. Si un seuil critique est atteint, il envoie un signal au CPU, qui transmet ensuite une notification au téléphone via le Wi-Fi pour demander une intervention (remplissage du réservoir).

ii. Les actionneurs :

- **Vérin électrique** : Actionné par l'API, il s'ouvre pour permettre le passage des aliments et se ferme lorsque la quantité requise est atteinte.

iii. Le réservoir :

Le réservoir alimentaire stocke les aliments destinés à être distribués. Sa gestion est contrôlée par les capteurs et le vérin électrique.

C. La partie eau :

Cette partie gère l'approvisionnement en eau du système.

i. Les capteurs :

- **Capteur de poids** : Mesure le poids de l'eau qui sort du réservoir. Une fois la quantité d'eau adéquate délivrée, le capteur envoie un signal à l'API pour fermer l'électrovanne, arrêtant ainsi l'écoulement de l'eau.
- **Capteur de niveau** : Surveille le niveau de l'eau dans le réservoir. Si le réservoir est presque vide, un signal est envoyé au CPU, qui alerte l'utilisateur via le téléphone pour qu'il remplisse le réservoir.
- **Capteur de position**: Détecte la présence de la vache. Une fois détectée, il envoie un signal à l'API pour ouvrir l'électrovanne et permettre l'écoulement de l'eau.

ii. Les actionneurs :

- **Électrovanne** : Contrôlée par l'API, elle s'ouvre pour laisser passer l'eau lorsque cela est nécessaire, et se ferme pour interrompre le flux d'eau.

iii. Le réservoir :

Comme pour la partie alimentation, un réservoir est utilisé pour stocker l'eau. Le niveau est surveillé en permanence par les capteurs pour assurer un approvisionnement adéquat.

II.4. Système de supervision

Un ensemble de capteurs utilisés pour surveiller et contrôler les infrastructures informatiques, les réseaux ou les équipements industriels.

a. Capteur ultrasons :

- **Utilisation :** Surveillance du niveau d'eau et de l'alimentation.
- **Description :** Les capteurs à ultrasons sont capables de mesurer des distances ou des niveaux sans contact physique, en utilisant des ondes sonores. Ils sont couramment utilisés pour surveiller le niveau d'eau dans les réservoirs ou les canalisations. Dans les systèmes industriels, ils peuvent aussi détecter la présence de liquides dans des conduits et alerter en cas de niveau anormal, que ce soit une fuite ou un débordement. Ce type de capteur est crucial pour la gestion des infrastructures hydrauliques ou de refroidissement.

b. Capteur de température et d'humidité :

- **Utilisation :** Contrôle de l'environnement pour assurer des conditions optimales.
- **Description :** Ce type de capteur surveille à la fois la température et le taux d'humidité dans un environnement donné. Dans les centres de données, ces capteurs permettent de vérifier que la température reste dans des limites sûres pour éviter la surchauffe des serveurs, ce qui pourrait entraîner des défaillances. Dans un cadre industriel, ils permettent également de surveiller les conditions ambiantes afin de garantir la sécurité des équipements et des processus. Une humidité excessive peut, par exemple, provoquer des courts-circuits ou la corrosion des composants.

c. Capteur de flamme :

- **Utilisation :** Détection des incendies ou anomalies thermiques dangereuses.
- **Description :** Le capteur de flamme est utilisé pour détecter la présence de flammes ou d'émissions infrarouges caractéristiques d'un incendie. En cas de détection, une alarme est déclenchée pour avertir les opérateurs, permettant une réponse rapide pour maîtriser l'incendie avant qu'il ne se propage. Dans des environnements critiques comme les centrales électriques ou les zones industrielles à risque, ce type de capteur est essentiel pour prévenir des incidents majeurs pouvant conduire à des pertes matérielles ou humaines.

II.5.Le fonctionnement de notre système

Notre système se divise en deux parties principales :

Partie 01 : Distribution d'aliment à la vache

- **Réservoir plein** : Lorsque le réservoir d'aliment est rempli, l'API, avec le programme chargé, commande l'ouverture du vérin électrique pour permettre à l'aliment de descendre et remplir un récipient. Un capteur de poids placé sous le récipient surveille la quantité d'aliment. Lorsque le récipient atteint le poids souhaité, le capteur envoie un signal à l'API, qui commande la fermeture du vérin électrique.
- **Réservoir presque vide** : Si le niveau d'aliment dans le réservoir devient faible, le capteur de niveau le détecte et envoie un signal à l'ESP32. Grâce à la connexion Wi-Fi, un

message est envoyé au téléphone pour notifier l'utilisateur qu'il doit intervenir et remplir le réservoir.

Partie 02 : Distribution d'eau à la vache

- **Réservoir plein** : Lorsque le réservoir d'eau est rempli, le capteur de position détecte la présence de la vache et envoie un signal à l'API. L'API, via son programme, commande l'ouverture de l'électrovanne pour laisser l'eau remplir le récipient. Un capteur de poids détecte le niveau de l'eau dans le récipient et, une fois rempli, envoie un signal à l'API pour fermer l'électrovanne et arrêter l'écoulement de l'eau.
- **Réservoir presque vide** : Si le réservoir d'eau est presque vide, le capteur de niveau le détecte et envoie un signal à la Raspberry Pi. Grâce à la connexion Wi-Fi, un message est envoyé au téléphone pour avertir l'utilisateur de la nécessité de remplir le réservoir.

Ce système permet de gérer efficacement l'alimentation et l'hydratation des vaches de manière automatisée, tout en alertant l'utilisateur en cas de réservoirs vides.

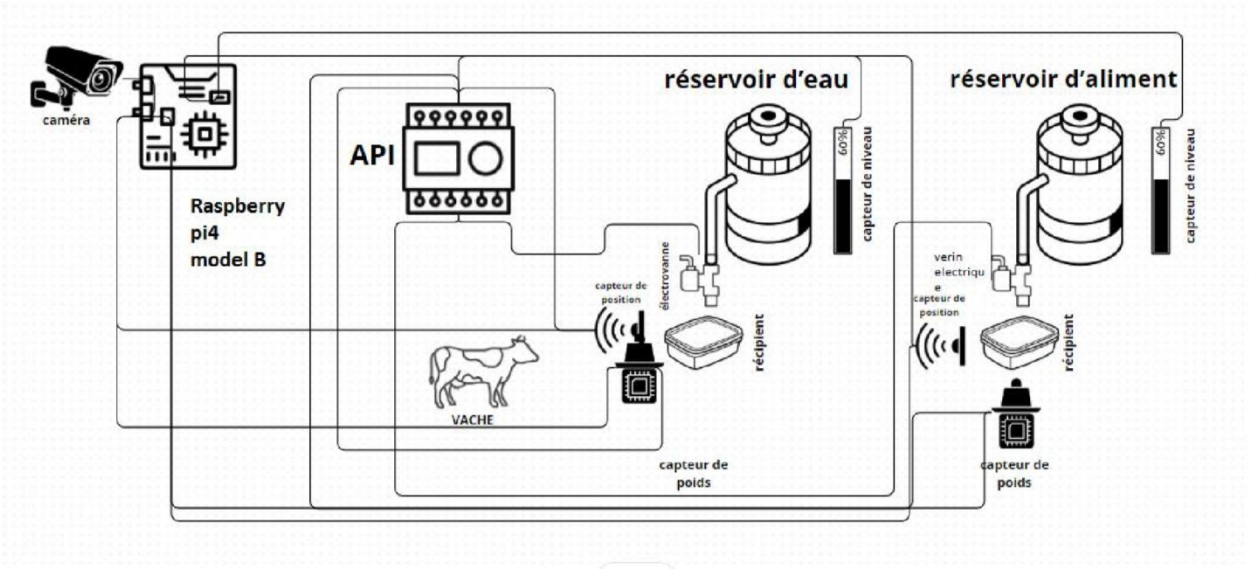


Figure 2.17 : câblage de système d'alimentation intelligent multifonctionnel

II.6. Stratégie de contrôle de notre projet

La stratégie de contrôle de notre projet repose sur l'automatisation et la supervision en temps réel des systèmes de distribution d'aliments et d'eau pour les vaches. L'objectif principal est d'assurer une alimentation et une hydratation optimales, tout en minimisant l'intervention humaine grâce à une gestion intelligente et des alertes en cas d'anomalies. Voici les différentes étapes et principes de cette stratégie de contrôle :

1. Automatisation basée sur des capteurs et actionneurs :

- **Capteurs intelligents :** Des capteurs de position, de poids et de niveau sont utilisés pour surveiller en permanence les réservoirs d'aliments et d'eau. Ces capteurs envoient des informations à l'API et à la Raspberry Pi pour déclencher des actions automatiques.

- **Actionneurs contrôlés :** Les actionneurs, tels que le vérin électrique et l'électrovanne, sont pilotés par l'API en fonction des signaux reçus des capteurs. Cela permet d'ouvrir ou de fermer le flux d'aliments ou d'eau de manière autonome et en temps réel.

2. Programmation et supervision via l'API et Raspberry Pi :

- **API (Automate Programmable Industriel) :** Le cœur de la gestion des actionneurs et des capteurs repose sur des programmes développés avec TIA Portal, qui contrôlent l'ouverture et la fermeture des électrovannes et vérins électriques en fonction des données des capteurs.
- **Raspberry Pi :** Cette carte assure une supervision globale du système en collectant les données des capteurs, en exécutant les calculs nécessaires via son CPU, et en envoyant des notifications via Wi-Fi en cas d'anomalie ou de besoin d'intervention (réservoirs presque vides).

3. Détection des situations critiques et réponse automatisée :

- **Réservoir plein :** Lorsqu'un réservoir est plein (que ce soit d'aliment ou d'eau), les capteurs de position et de poids déterminent si la vache est présente et surveillent la quantité distribuée. Le système ajuste automatiquement le flux en fonction des besoins.
- **Réservoir presque vide :** En cas de faible niveau détecté par les capteurs, une alerte est envoyée automatiquement à l'utilisateur via Wi-Fi pour signaler qu'une action est nécessaire, comme le remplissage du réservoir. Cela garantit une continuité dans l'alimentation et l'hydratation des vaches.

4. Système de notification et gestion à distance :

- **Notifications en temps réel :** À l'aide de l'ESP32 et de la connexion Wi-Fi, le système est capable d'envoyer des notifications sur le téléphone portable de l'utilisateur, l'informant d'une intervention nécessaire (comme le remplissage des réservoirs).
- **Contrôle à distance :** Cette architecture permet à l'utilisateur de surveiller l'état du système à distance et d'intervenir rapidement en cas de problème, améliorant ainsi l'efficacité du système et la qualité de service.

5. Optimisation des ressources et maintenance prédictive :

- **Prévention des pannes :** En surveillant en permanence les niveaux d'eau et d'aliment, et en anticipant les pénuries grâce aux capteurs, le système prévient les pannes potentielles, minimisant ainsi les interruptions dans l'alimentation des animaux.
- **Gestion efficace des ressources :** Grâce à l'automatisation, l'alimentation et l'eau sont distribuées avec précision, ce qui évite les gaspillages et garantit un approvisionnement optimal en fonction des besoins réels.

II.7. Limites de notre projet

Bien que notre projet présente de nombreux avantages en termes d'automatisation et de gestion efficace des ressources, il existe certaines limites et défis qui doivent être pris en compte :

- Connexion Wi-Fi requise : Le système repose sur la communication sans fil via Wi-Fi pour envoyer des alertes à distance. Si la connexion Wi-Fi est instable ou indisponible, les notifications ne pourront pas être transmises, ce qui peut retarder les interventions critiques, comme le remplissage des réservoirs.
- Zone de couverture limitée : Dans les environnements ruraux ou éloignés, la couverture réseau peut être insuffisante, rendant difficile l’utilisation efficace du système à distance.
- Fiabilité des capteurs : Les capteurs (de poids, de niveau, de position) sont essentiels pour le bon fonctionnement du système. Toutefois, ils peuvent être sensibles à l’usure, à l’encrassement ou aux conditions environnementales difficiles (humidité, poussière, variations de température). Une défaillance d’un capteur pourrait entraîner un mauvais fonctionnement du système ou des données erronées.
- Usure des actionneurs : Les actionneurs tels que les vérins électriques et les électrovannes subissent des sollicitations régulières. Une maintenance régulière est nécessaire pour éviter des pannes mécaniques, ce qui pourrait alourdir les coûts d'entretien.
- Intervention humaine requise : Même avec des notifications automatiques, l’utilisateur doit intervenir physiquement pour recharger les réservoirs. Cela peut poser problème en cas d'absence prolongée ou d’imprévus, créant une interruption dans l’approvisionnement des animaux.
- Compétences techniques nécessaires : La mise en place, la configuration et la maintenance du système nécessitent des compétences techniques en électronique, programmation (TIA Portal pour l'API), et réseau. Cela peut représenter une barrière pour les utilisateurs qui ne sont pas familiarisés avec ces technologies.

II.8. Conclusion

Notre projet propose une solution innovante et automatisée pour la gestion de l’alimentation et de l’hydratation des vaches, en utilisant une combinaison de capteurs, d’actionneurs, d’API, et de Raspberry Pi. Cette automatisation permet de réduire l’intervention humaine, d’optimiser les ressources et de garantir un suivi en temps réel grâce à des notifications via Wi-Fi

Chapitre III - Prototype programmé à l'aide de l'ESP32

III.1. Introduction :

Dans ce chapitre on vous montrera comment on a réalisé tout le projet à l'aide de l'ESP32 qu'on a programmé avec langage de programmation C++ avec logiciel ARDUINO IDE, et on a réalisé une petite application pour vous montrer les valeurs de tous les capteurs comme le niveau de l'eau, la température et l'humidité… à l'aide du langage de programmation HTML, CSS ET JAVASCRYPT avec logiciel VISUELSTUDIO.

Et on a réalisé le câblage des composants en utilisant le site web WOKWI.COM.

III.2. Composants utilisés et leurs rôles :

III.2.1. ESP 32 :

L'ESP32 est un microcontrôleur à faible consommation d'énergie, équipé d'un processeur dual-core basé sur l'architecture Tensilica LX6. Il est particulièrement apprécié pour sa connectivité intégrée, offrant à la fois des capacités Wifi (802.11 b/g/n) et Bluetooth (classique et BLE, pour Bluetooth Low Energy). Ce microcontrôleur est utilisé dans une grande variété de projets et d'applications, allant des objets connectés aux systèmes embarqués.

Son rôle est de commander et contrôler tous les autres composants qui seront connectés au GPIO pins.

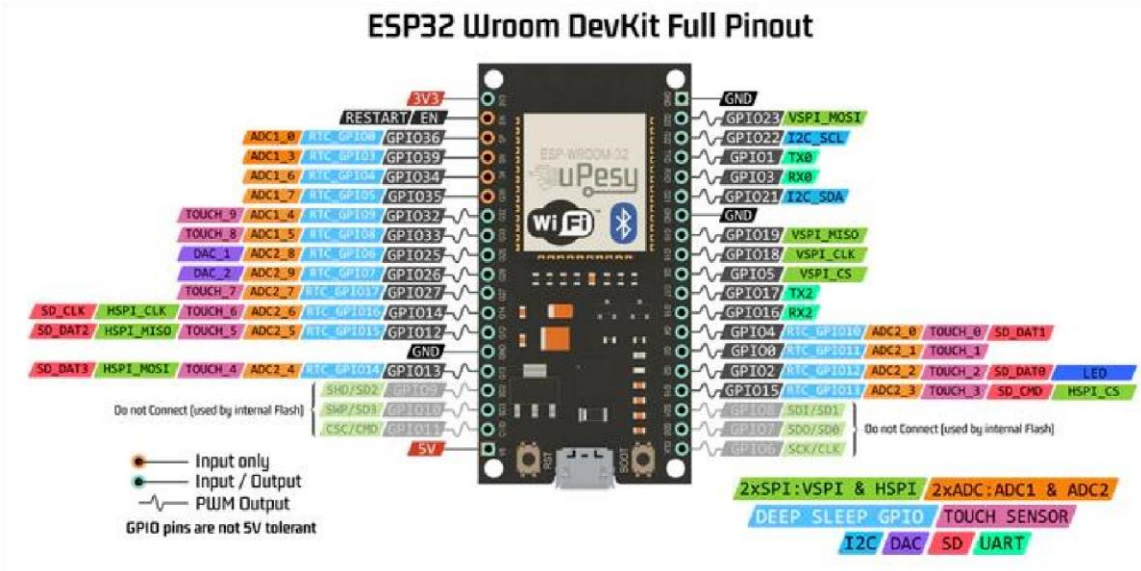


Figure 3.1 : ESP 32 GPIO pins [14]

III.2.2.Plaque d'essai :

Est un dispositif de construction de circuits électroniques temporaires, elle est généralement constituée d'une grille de trous conducteurs connectés de manière spécifique, facilitant la connexion des composants électroniques.

Son rôle est de nous permettre d'assembler, de tester et de modifier des circuits sans souder les composants.

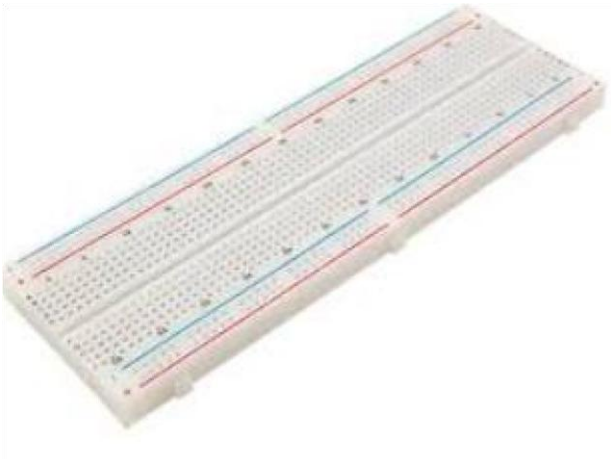


Figure 3.2 : plaque d'essai [15]

III.2.3. Carte d'alimentation (MB 102) :

Est un module de distribution d'alimentation conçu pour fournir des tensions stables à partir d'une source d'alimentation externe vers les composants d'un circuit monté sur une plaque d'essai.

Son rôle est d'alimenter l'esp 32 et les autres composants.

Figure 3.3 : carte d'alimentation (MB102) [16]

III.2.4.Capteur de température (DHT 11) :

Est un capteur électronique utilisé pour mesurer la température ambiante et l'humidité relative.

Son rôle est de mesurer la température et l'humidité de l'environnement de la vache.

Figure 3.4 : capteur de température (DHT 11) [17]

III.2.5. Capteur de flamme (LM 393) :

Est un circuit intégré comparateur de tension utilisé dans des systèmes de détection de feu. Il compare le signal d'un capteur de feu, tel qu'un capteur thermique ou optique, à un seuil de référence. Lorsque la mesure du capteur dépasse ce seuil, le LM393 génère un signal de sortie pour alerter de la présence d'un incendie ou de chaleur excessive.

Son rôle est de détecter si il y'a une flamme dans le système.

Figure 3.5 : capteur de flamme (LM 393) [18]

III.2.6. Les composants de système d'eau :

a. Electrovanne :

Est un dispositif électromécanique, elle fonctionne en utilisant un courant électrique pour activer ou désactiver une vanne, permettant ainsi de réguler le passage du fluide.

Son rôle est de contrôler le passage de l'eau.

Figure 3.6 : Electrovanne

b. Relais :

Un interrupteur automatique qui utilise une bobine électromagnétique pour activer ou désactiver un ou plusieurs contacts électriques. Lorsqu'un courant circule dans la bobine, il crée un champ magnétique qui déplace un noyau mobile, ouvrant ou

fermant les contacts du relais. Cela permet de contrôler un circuit électrique avec un autre circuit.

Son rôle est de connecter entre l'électrovanne et l'esp32.

Figure 3.7 : relais

C. Capteur ultra son (HC-SR04) :

Est un appareil de mesure qui émet des ondes sonores à haute fréquence (ultrasons) et mesure le temps nécessaire à ces ondes pour se réfléchir sur un objet et revenir au capteur. En utilisant le temps de vol des ondes sonores, le capteur peut calculer la distance entre lui-même et l'objet.

Son rôle est de mesurer le niveau de l'eau

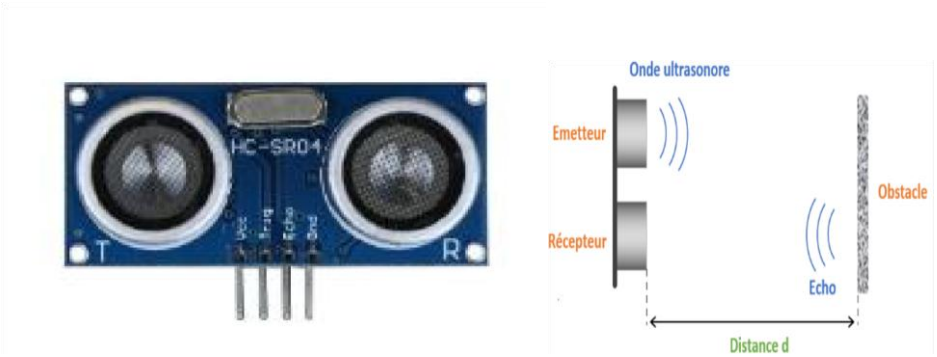


Figure 3.8. : Capteur ultrason

d. Capteur infrarouge (REES52) :

Est un capteur électronique qui émet des rayons infrarouges et détecte leur réflexion sur les objets à proximité. Il est utilisé pour mesurer des distances courtes ou pour détecter la présence d'objets dans un champ de détection spécifique.

Son rôle est de détecter la vache pour ouvrir l'électrovanne.

Figure 3.9 : Capteur infrarouge

e. Capteur d'eau :

Un dispositif de détection qui mesure la présence ou le niveau d'eau dans un réservoir.

Son rôle est de détecter si il y'a de l'eau dans le récipient pour fermer l'électrovanne.

Figure 3.10 : Capteur d'eau

III.2.7. Les composants de système d'aliment :

a. Servo-moteur:

Est un moteur électrique à courant continu qui se déplace en pas précis, chaque pas correspondant à une fraction du tour complet du rotor. Il est conçu pour fournir une rotation angulaire précise à chaque impulsion électrique reçue.

Son rôle est de laisser passer l'aliment pour l'animal.

Figure 3.11 : Servo-moteur

b. Capteur de poids (HX711 ADC) :

Est un dispositif de mesure qui détecte le poids ou la force appliquée et la convertit en un signal électrique proportionnel.

Son rôle est de mesurer le poids d'aliment pour commander le moteur pas à pas s'il s'ouvre ou non.

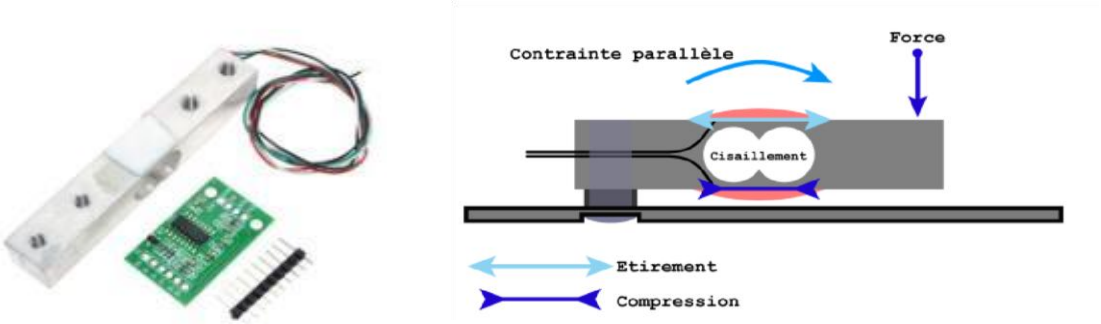


Figure 3.12 : Capteur de poids

c. Capteur ultra son (HC-SR04) : revenir à (III.2.6.c)

III.3. Câblages des composants : (WOKWI) [19]

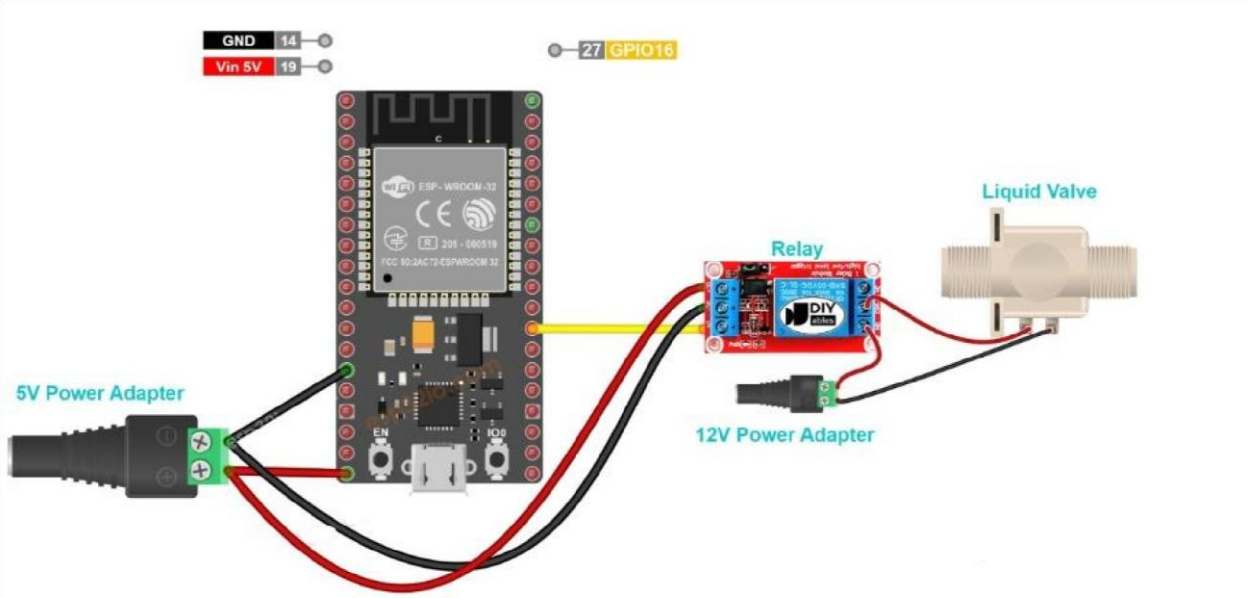


Figure 3.13 : Câblage de l'électrovanne, relais et ESP32

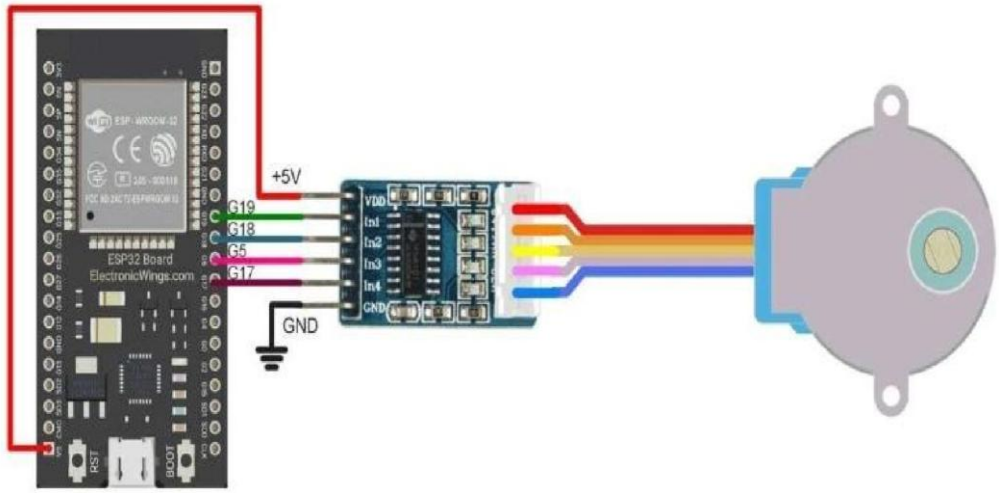


Figure 3.14 : Câblage de Servo-moteur au ULN2003 et ESP32

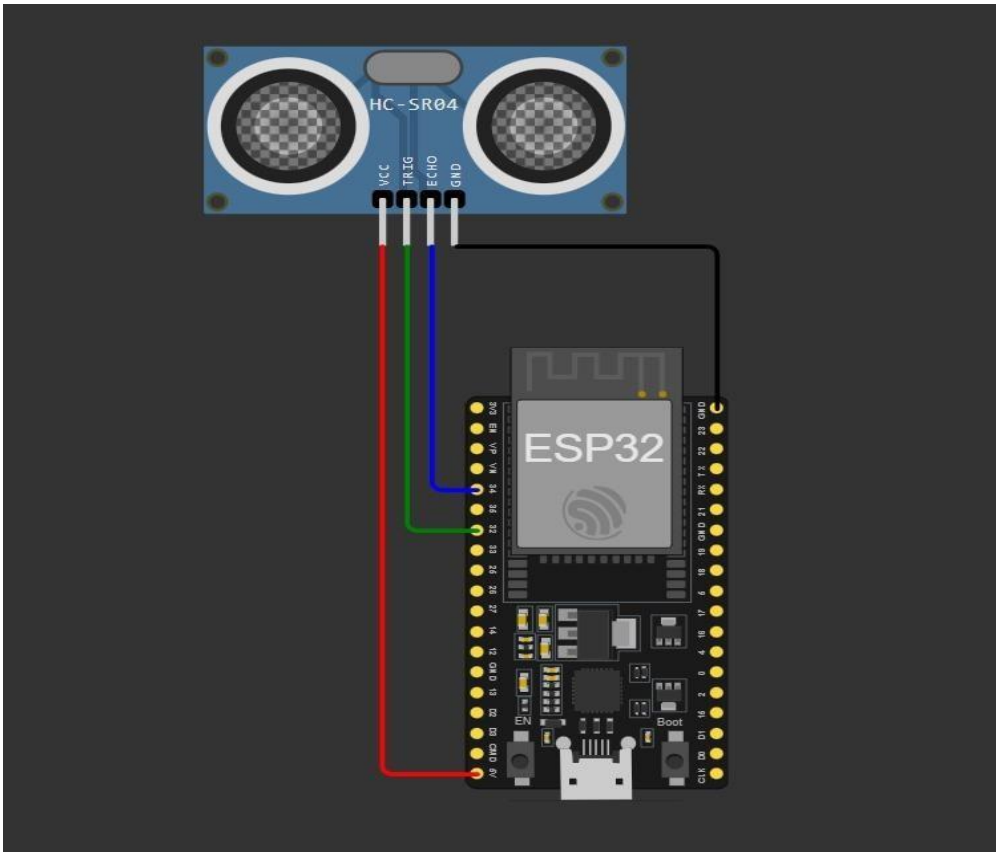


Figure 3.15 : Câblage de capture ultrason (HC-SR04)

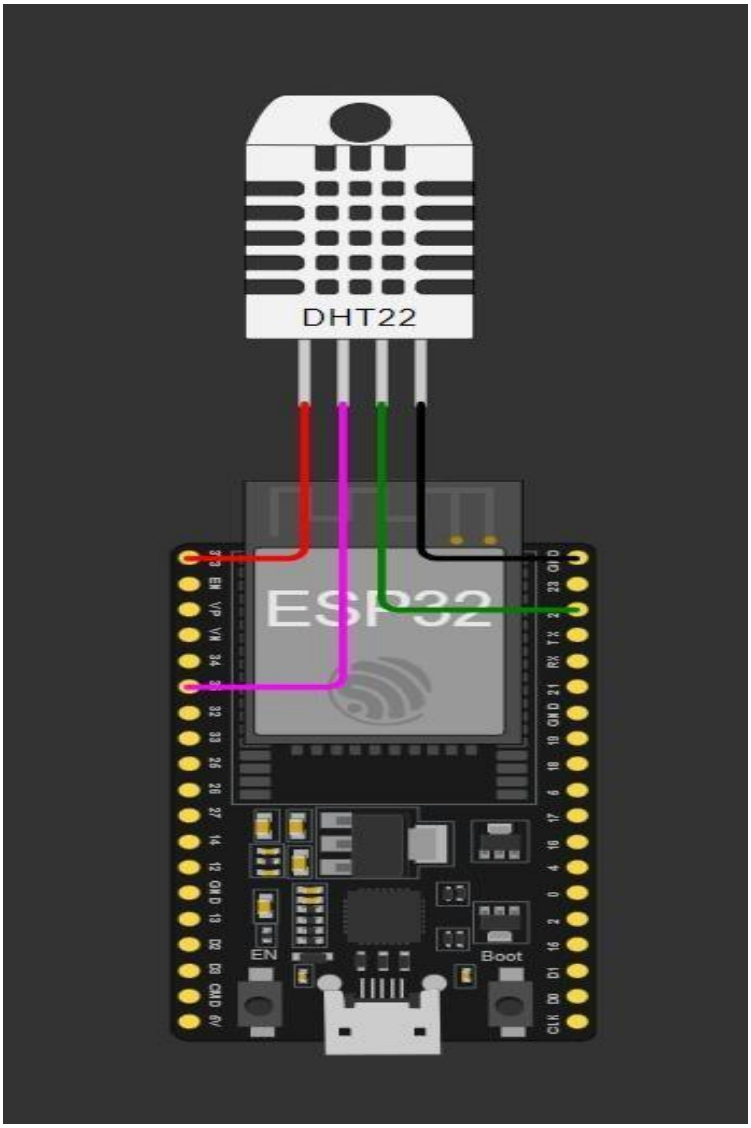


Figure 3.16 : Câblage de capture de température (DHT 11)

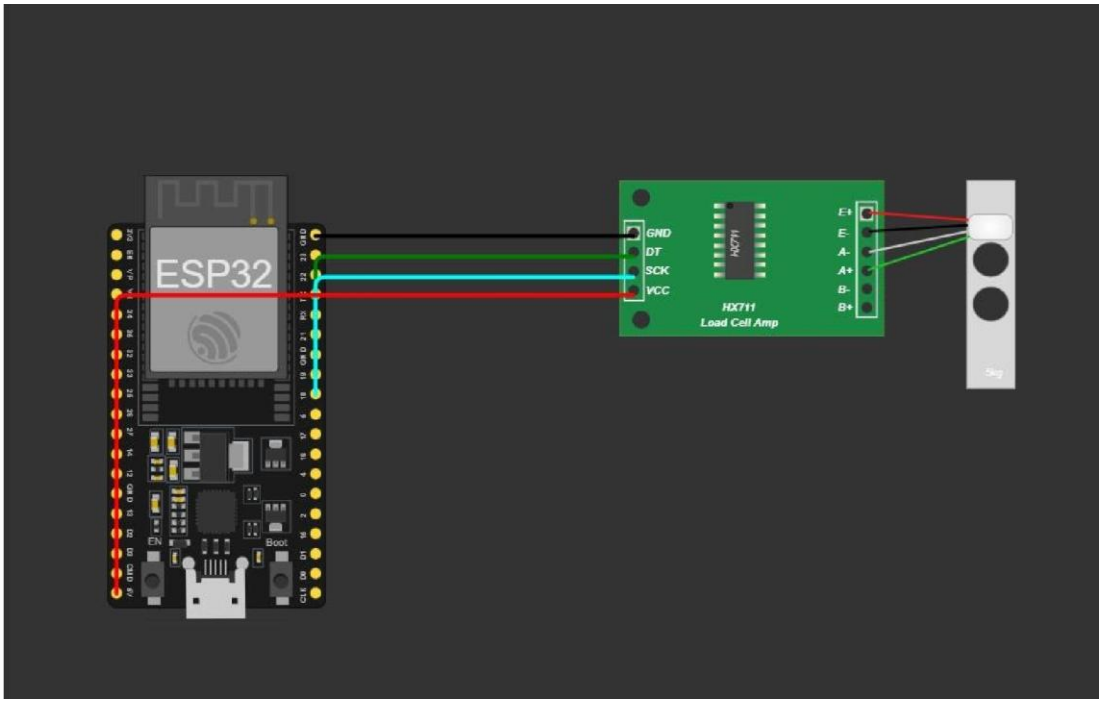


Figure 3.17 : Câblage de capture de poids (HX711 ADC)

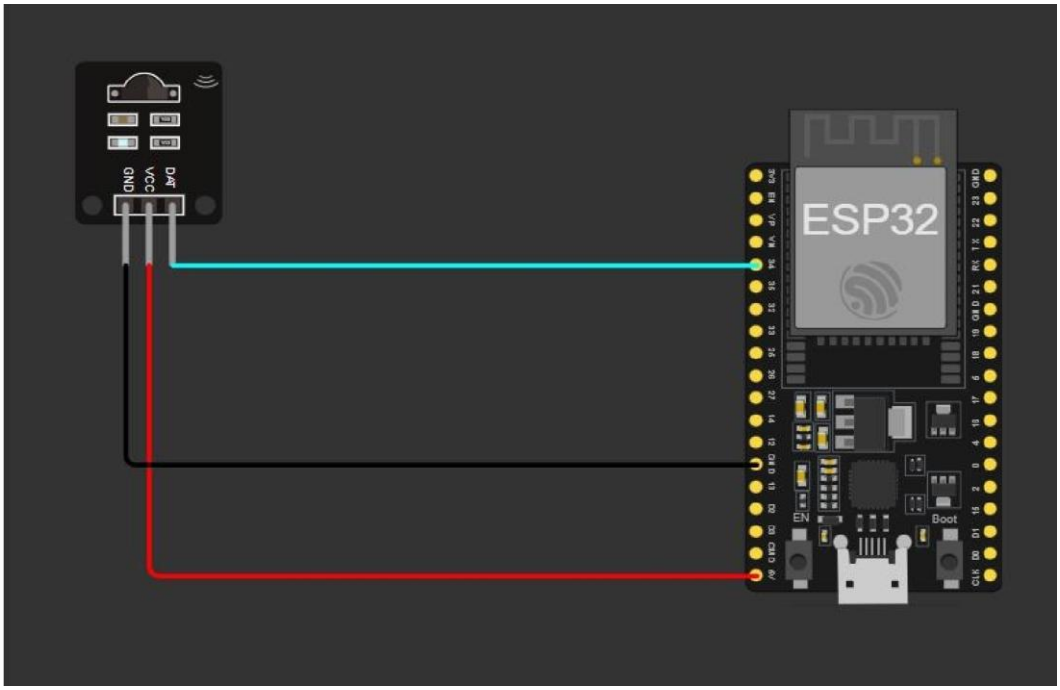


Figure 3.18 : Câblage de capture infrarouge (REES52)

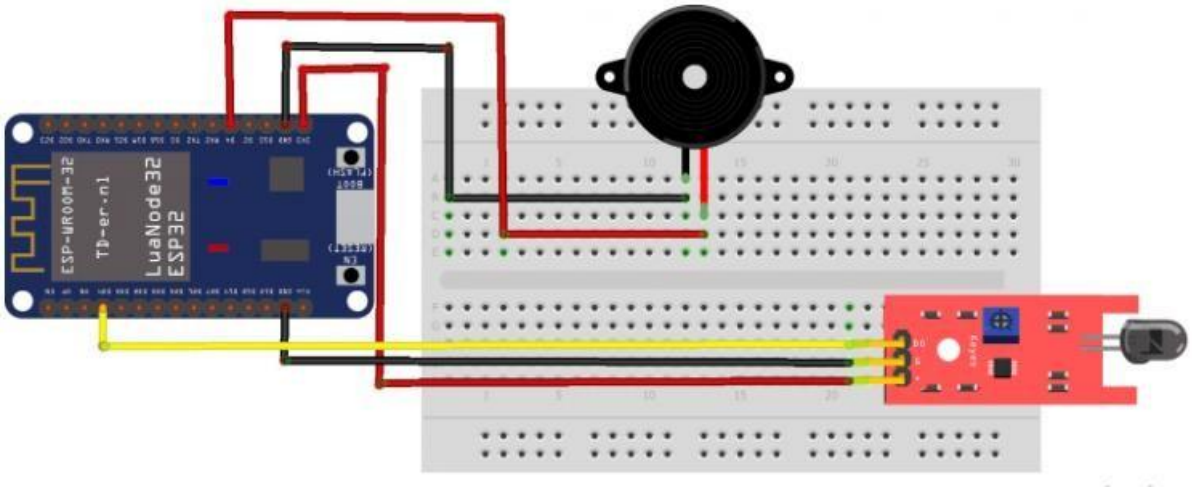


Figure 3.19 : Câblage de capture Capteur de flamme (LM 393)

CAPTEUR	CAPTEUR PIN	ESP 32 PIN	VCC 5V	VCC 3.3V	GND
Poids (HX711ADC)	DATA CLOCK	19 18	*		*
Ultrason (HC-SR04)	ECHO TRIG	32 33	*		
Flamme (LM 393)	DATA	15		*	
Infrarouge (REES52)	DATA	34		*	
Température (DHT 11)	DATA	12		*	
Eau	DATA	25		*	

Table 3.1 : câblage des capteurs

Actionneurs	Actionneur pin	ESP32 PIN	VCC 5V	VCC 3.3V	GND
Moteur pas à pas (28BYJ-48)	INT 1 INT 2 INT 3 INT 4	5 17 16 4	*		*
Relais	INT	13	*		

Table 3.2 : câblage des actionneurs

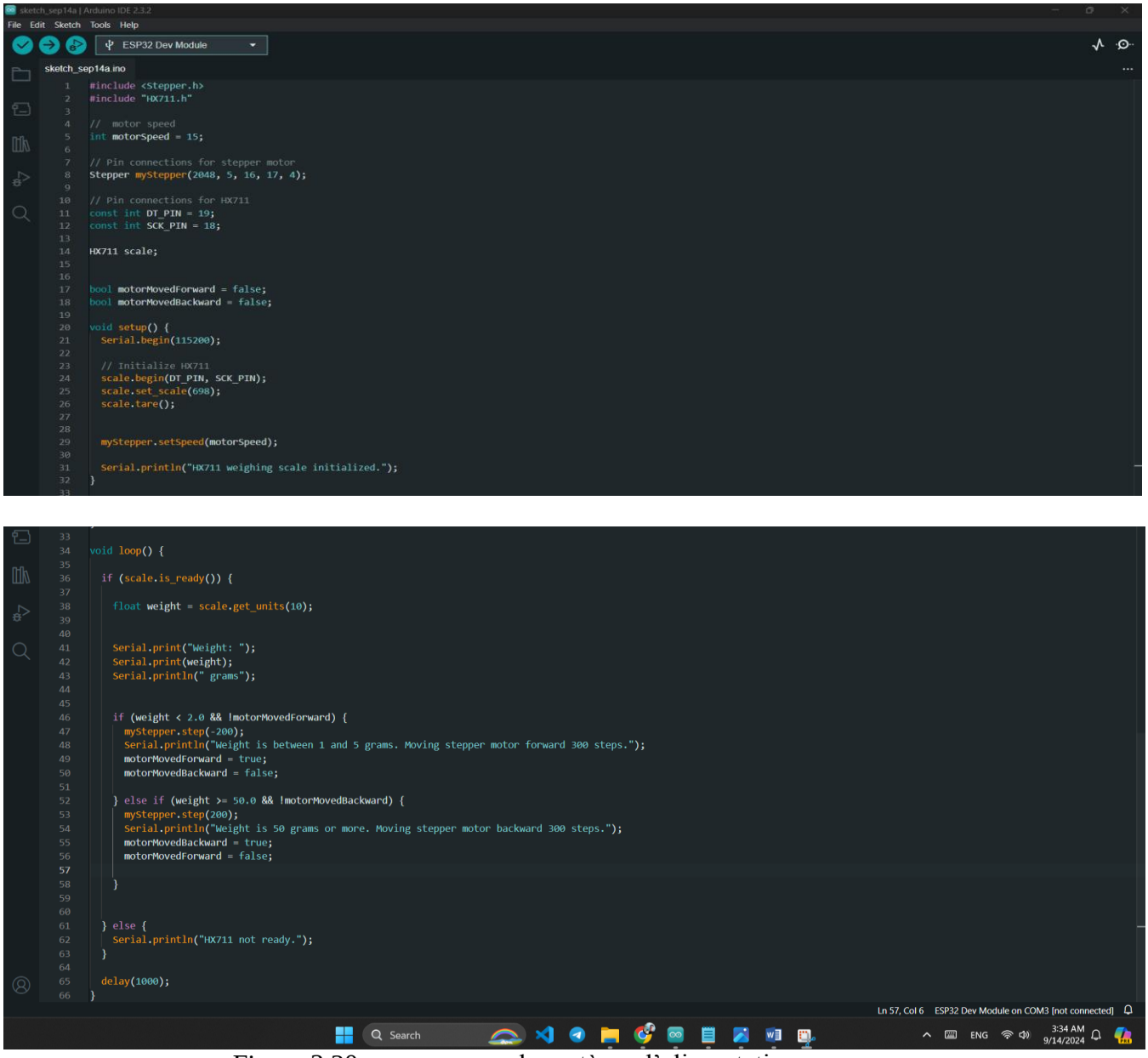
	électrovanne		relais		Power adapter	
PIN	+	-	com	no	+	-
Electrovanne			*			*
Relais	*				*	
Power adapter		*		*		

Table 3.3 : câblage de l'électrovanne

III.4. PROGRAMMATION : (ARDUINO IDE) [20]

III.4.1. SYSTEME D'ALIMENTATION :

a. PROGRAMME COMPLET :



```ino
#include <Stepper.h>
#include "HX711.h"

//  motor speed
int motorSpeed = 15;

// Pin connections for stepper motor
Stepper myStepper(2048, 5, 16, 17, 4);

// Pin connections for HX711
const int DT_PIN = 19;
const int SCK_PIN = 18;

HX711 scale;


bool motorMovedForward = false;
bool motorMovedBackward = false;

void setup() {
  Serial.begin(115200);

  // Initialize HX711
  scale.begin(DT_PIN, SCK_PIN);
  scale.set_scale(698);
  scale.tare();


  myStepper.setSpeed(motorSpeed);

  Serial.println("HX711 weighing scale initialized.");
}

void loop() {

  if (scale.is_ready()) {

    float weight = scale.get_units(10);


    Serial.print("Weight: ");
    Serial.print(weight);
    Serial.println(" grams");


    if (weight < 2.0 && !motorMovedForward) {
      myStepper.step(-200);
      Serial.println("Weight is between 1 and 5 grams. Moving stepper motor forward 300 steps.");
      motorMovedForward = true;
      motorMovedBackward = false;

    } else if (weight >= 50.0 && !motorMovedBackward) {
      myStepper.step(200);
      Serial.println("Weight is 50 grams or more. Moving stepper motor backward 300 steps.");
      motorMovedBackward = true;
      motorMovedForward = false;

    }


  } else {
    Serial.println("HX711 not ready.");
  }

  delay(1000);
}
```

Figure 3.20 : programme de système d'alimentation

b. EXPLICATION DE CODE :

```ino
#include <Stepper.h>
#include "HX711.h"
```

Figure 3.20-a : explication de programme de système d'alimentation

#include <Stepper.h> : Inclut la bibliothèque Arduino pour contrôler les moteurs pas à pas.
#include "HX711.h" : Inclut la bibliothèque HX711 pour interfacer avec le module amplificateur de cellule de charge HX711.

```
4   //  motor speed
5   int motorSpeed = 15;
6
7   // Pin connections for stepper motor
8   Stepper myStepper(2048, 5, 16, 17, 4);
```

Figure3.20-b : explication de programme de système d'alimentation

motorSpeed : Définit la vitesse du moteur pas à pas, ici à 15 tours par minute (RPM). Stepper myStepper(2048, 5, 16, 17, 4); : Crée un objet Stepper nommé myStepper avec 2048 pas par révolution. Le moteur est connecté aux broches GPIO 5, 16, 17 et 4 de l'ESP32.

```
10  // Pin connections for HX711
11  const int DT_PIN = 19;
12  const int SCK_PIN = 18;
13
14  HX711 scale;
15
```

Figure 3.20-c : explication de programme de système d'alimentation

DT_PIN et SCK_PIN : Définissent les broches de données et d'horloge pour le module de cellule de charge HX711, connectées aux broches GPIO 19 (DT) et GPIO 18 (SCK).
HX711 scale; Crée un objet HX711 nommé scale pour gérer les données de la cellule de charge.

```
17  bool motorMovedForward = false;
18  bool motorMovedBackward = false;
```

Figure 3.20-d : explication de programme de système d'alimentation

MotorMovedForward : Suivi si le moteur a avancé en fonction de la condition de poids.
MotorMovedBackward : Suivi si le moteur a reculé.

```
20  void setup() {
21    Serial.begin(115200);
22
23    // Initialize HX711
24    scale.begin(DT_PIN, SCK_PIN);
25    scale.set_scale(698);
26    scale.tare();
27
28
29    myStepper.setSpeed(motorSpeed);
30
31    Serial.println("HX711 weighing scale initialized.");
32  }
33
```

Figure 3.20-e : explication de programme de système d'alimentation

Serial.begin(115200); : Initialise la communication série à 115200 bauds pour le débogage.
scale.begin(DT_PIN, SCK_PIN); : Initialise le HX711 avec les broches de données et d'horloge spécifiées.

scale.set_scale(698); : Définit le facteur d'échelle pour la cellule de charge afin de calibrer les lectures de poids (ce chiffre dépend de votre capteur spécifique).

scale.tare(); : Remet la balance à zéro, considérant que la charge actuelle est nulle.

myStepper.setSpeed(motorSpeed); : Définit la vitesse du moteur pas à pas à motorSpeed (15 RPM).

Serial.println : Affiche un message indiquant que la balance HX711 est initialisé.

```
void loop() {

  if (scale.is_ready()) {

    float weight = scale.get_units(10);


    Serial.print("Weight: ");
    Serial.print(weight);
    Serial.println(" grams");


    if (weight < 2.0 && !motorMovedForward) {
      myStepper.step(-200);
      Serial.println("Weight is between 1 and 5 grams. Moving stepper motor forward 300 steps.");
      motorMovedForward = true;
      motorMovedBackward = false;

    } else if (weight >= 50.0 && !motorMovedBackward) {
      myStepper.step(200);
      Serial.println("Weight is 50 grams or more. Moving stepper motor backward 300 steps.");
      motorMovedBackward = true;
      motorMovedForward = false;

    }


  } else {
    Serial.println("HX711 not ready.");
  }

  delay(1000);
}
```

Figure 3.20-f : explication de programme de système d'alimentation

if (scale.is_ready()) : Vérifie si le HX711 est prêt à prendre une mesure.
float weight = scale.get_units(10); : Lit le poids moyen sur 10 échantillons depuis le HX711 et le stocke dans weight.
Serial.print("Poids : "); : Affiche le poids en grammes pour le suivi.
Logique de Contrôle du Moteur :

- **if (weight < 2.0 && !motorMovedForward) :** Si le poids est inférieur à 2 grammes et que le moteur n'a pas avancé, il avance de 200 pas (le signe négatif indique la direction), affiche un message, et met à jour les drapeaux motorMovedForward à true et motorMovedBackward à false.
- **else if (weight >= 50.0 && !motorMovedBackward) :** Si le poids est supérieur ou égal à 50 grammes et que le moteur n'a pas reculé, il recule de 200 pas, affiche un message, et met à jour les drapeaux.

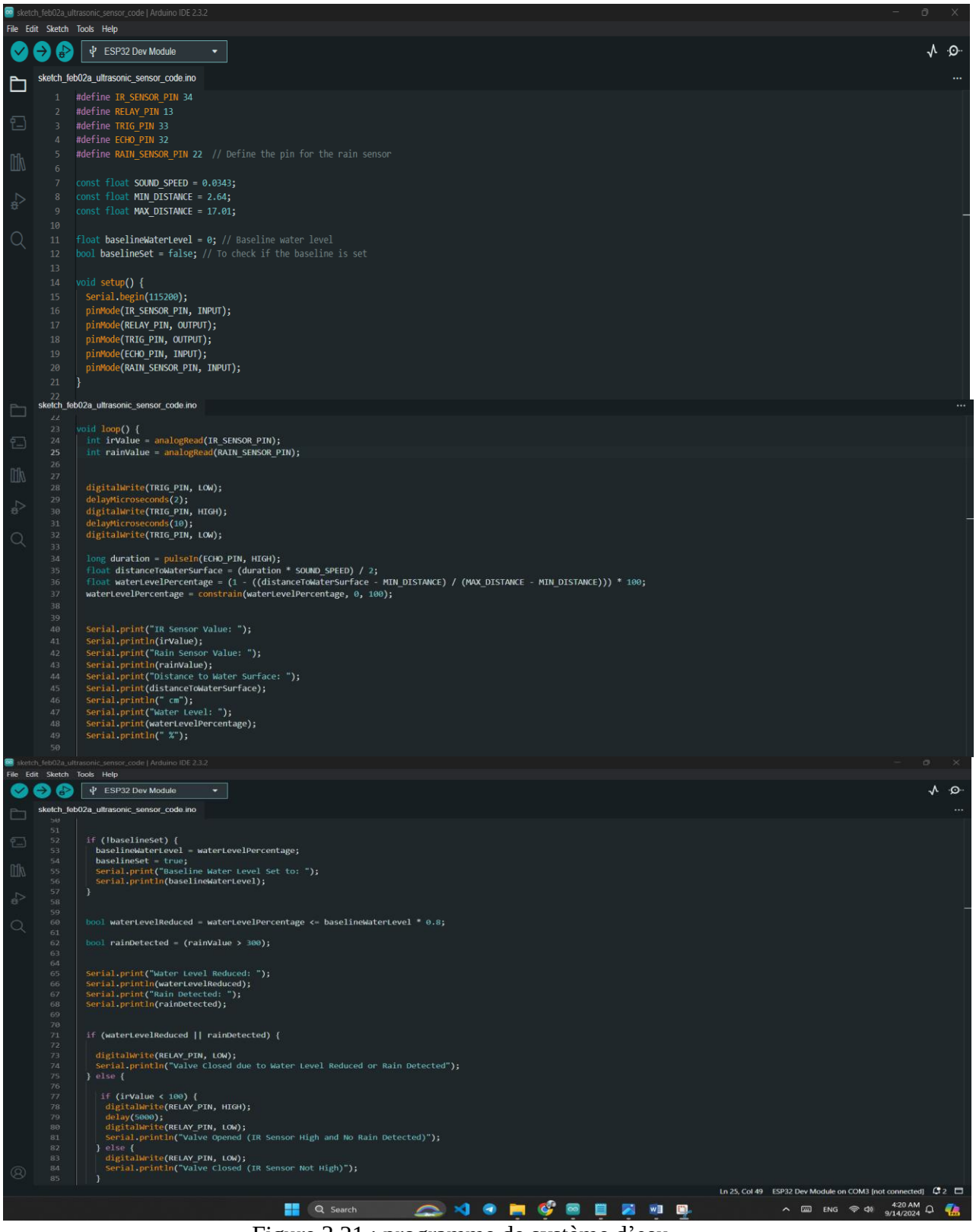
else : Affiche "HX711 non prêt" si le HX711 n'est pas prêt pour une lecture.
delay(1000); : Met en pause la boucle pendant 1 seconde avant la prochaine itération.

Résumé :
Le code lit le poids d'une cellule de charge HX711 et fait avancer ou reculer un moteur pas à pas selon certaines conditions de poids :

- Si le poids est inférieur à 2 grammes, le moteur avance.
- Si le poids est supérieur ou égal à 50 grammes, le moteur recule.

III.4.2. SYSTEME D'EAU :

a. PROGRAMME COMPLET



```
#define IR_SENSOR_PIN 34
#define RELAY_PIN 13
#define TRIG_PIN 33
#define ECHO_PIN 32
#define RAIN_SENSOR_PIN 22  // Define the pin for the rain sensor

const float SOUND_SPEED = 0.0343;
const float MIN_DISTANCE = 2.64;
const float MAX_DISTANCE = 17.01;

float baselineWaterLevel = 0; // Baseline water level
bool baselineSet = false; // To check if the baseline is set

void setup() {
  Serial.begin(115200);
  pinMode(IR_SENSOR_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(RAIN_SENSOR_PIN, INPUT);
}

void loop() {
  int irValue = analogRead(IR_SENSOR_PIN);
  int rainValue = analogRead(RAIN_SENSOR_PIN);


  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH);
  float distanceToWaterSurface = (duration * SOUND_SPEED) / 2;
  float waterLevelPercentage = (1 - ((distanceToWaterSurface - MIN_DISTANCE) / (MAX_DISTANCE - MIN_DISTANCE))) * 100;
  waterLevelPercentage = constrain(waterLevelPercentage, 0, 100);


  Serial.print("IR Sensor Value: ");
  Serial.println(irValue);
  Serial.print("Rain Sensor Value: ");
  Serial.println(rainValue);
  Serial.print("Distance to Water Surface: ");
  Serial.print(distanceToWaterSurface);
  Serial.println(" cm");
  Serial.print("Water Level: ");
  Serial.print(waterLevelPercentage);
  Serial.println(" %");


  if (!baselineSet) {
    baselineWaterLevel = waterLevelPercentage;
    baselineSet = true;
    Serial.print("Baseline Water Level Set to: ");
    Serial.println(baselineWaterLevel);
  }


  bool waterLevelReduced = waterLevelPercentage <= baselineWaterLevel * 0.8;

  bool rainDetected = (rainValue > 300);


  Serial.print("Water Level Reduced: ");
  Serial.println(waterLevelReduced);
  Serial.print("Rain Detected: ");
  Serial.println(rainDetected);


  if (waterLevelReduced || rainDetected) {

    digitalWrite(RELAY_PIN, LOW);
    Serial.println("Valve Closed due to Water Level Reduced or Rain Detected");
  } else {

    if (irValue < 100) {
      digitalWrite(RELAY_PIN, HIGH);
      delay(5000);
      digitalWrite(RELAY_PIN, LOW);
      Serial.println("Valve Opened (IR Sensor High and No Rain Detected)");
    } else {
      digitalWrite(RELAY_PIN, LOW);
      Serial.println("Valve Closed (IR Sensor Not High)");
    }
```

Figure 3.21 : programme de système d'eau

b. EXPLICATION DE CODE :

```
#define IR_SENSOR_PIN 34
#define RELAY_PIN 13
#define TRIG_PIN 33
#define ECHO_PIN 32
#define RAIN_SENSOR_PIN 22  // Define the pin for the rain sensor
```

Figure 3.21-a : explication de programme de système d'eau

IR_SENSOR_PIN (34) : La broche connectée au capteur IR, probablement utilisée pour détecter la présence d'un objet ou le débit d'eau.

RELAY_PIN (13) : La broche contrôlant un relais, qui pourrait être utilisée pour ouvrir ou fermer une vanne ou un autre dispositif de contrôle.

TRIG_PIN (33) & ECHO_PIN (32) : Broches utilisées pour un capteur à ultrasons pour mesurer le niveau d'eau. TRIG_PIN déclenche l'impulsion ultrasonore, et ECHO_PIN reçoit l'impulsion réfléchie pour calculer la distance jusqu'à la surface de l'eau.

RAIN_SENSOR_PIN (22) : La broche connectée au capteur de pluie, qui détecte s'il pleut ou non.

```
const float SOUND_SPEED = 0.0343;
const float MIN_DISTANCE = 2.64;
const float MAX_DISTANCE = 17.01;
```

Figure 3.21-b : explication de programme de système d'eau

SOUND_SPEED (0,0343 cm/µs) : La vitesse du son dans l'air est de 343 mètres par seconde, mais comme nous mesurons la distance en centimètres, elle est convertie en 0,0343 cm par microseconde.

MIN_DISTANCE (2,64 cm) & MAX_DISTANCE (17,01 cm) : Ces distances représentent la distance minimale et maximale entre le capteur et la surface de l'eau que le système est conçu pour mesurer.

```
float baselineWaterLevel = 0; // Baseline water level
bool baselineSet = false; // To check if the baseline is set
```

Figure 3.21-c : explication de programme de système d'eau

baselineWaterLevel : Le niveau d'eau initial mesuré lorsque le système démarre. Cela sert de point de référence pour déterminer les changements du niveau d'eau.

baselineSet : Un indicateur qui s'assure que le niveau de référence n'est défini qu'une seule fois, lors de la première itération de la boucle.

```
void setup() {
  Serial.begin(115200);
  pinMode(IR_SENSOR_PIN, INPUT);
  pinMode(RELAY_PIN, OUTPUT);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(RAIN_SENSOR_PIN, INPUT);
}
```

Figure3.21-d : explication de programme de système d'eau

Serial.begin(115200) : Initialise la communication série à un débit de 115200 pour le débogage et la journalisation.

pinMode : Définit le mode de chaque broche :

- **IR_SENSOR_PIN** et **RAIN_SENSOR_PIN** sont des entrées, recevant les données des capteurs.
- **RELAY_PIN** est une sortie, utilisée pour contrôler un relais (par exemple, ouvrir/fermer une vanne).
- **TRIG_PIN** est une sortie pour déclencher le capteur ultrasonique, et **ECHO_PIN** est une entrée pour lire l'impulsion réfléchie du capteur.

```
24    int irValue = analogRead(IR_SENSOR_PIN);
25    int rainValue = analogRead(RAIN_SENSOR_PIN);
```

Figure 3.21-e : explication de programme de système d'eau

irValue : Lit la valeur du capteur IR, qui pourrait indiquer un débit d'eau ou la présence d'un objet.

rainValue : Lit la valeur du capteur de pluie pour détecter la présence de pluie.

```
28  digitalWrite(TRIG_PIN, LOW);
29  delayMicroseconds(2);
30  digitalWrite(TRIG_PIN, HIGH);
31  delayMicroseconds(10);
32  digitalWrite(TRIG_PIN, LOW);
33
34  long duration = pulseIn(ECHO_PIN, HIGH);
35  float distanceToWaterSurface = (duration * SOUND_SPEED) / 2;
```

Figure 3.21-f : explication de programme de système d'eau

TRIG_PIN est pulsée pour envoyer une onde sonore à partir du capteur ultrasonique. Une impulsion haute de 10 microsecondes déclenche l'onde sonore.

pulseIn() : Mesure le temps que met le signal ultrasonore à revenir au capteur.

distanceToWaterSurface : La distance est calculée en fonction du temps (duration) et de la vitesse du son, divisée par 2 (puisque le son se déplace vers la surface et revient).

```
36  float waterLevelPercentage = (1 - ((distanceToWaterSurface - MIN_DISTANCE) / (MAX_DISTANCE - MIN_DISTANCE))) * 100;
37  waterLevelPercentage = constrain(waterLevelPercentage, 0, 100);
```

Figure 3.21-g : explication de programme de système d'eau

waterLevelPercentage : Calcule le niveau d'eau en pourcentage entre la distance minimale et maximale.

constrain() : S'assure que le pourcentage du niveau d'eau reste entre 0 % et 100 %.

```
39
40    Serial.print("IR Sensor Value: ");
41    Serial.println(irValue);
42    Serial.print("Rain Sensor Value: ");
43    Serial.println(rainValue);
44    Serial.print("Distance to Water Surface: ");
45    Serial.print(distanceToWaterSurface);
46    Serial.println(" cm");
47    Serial.print("Water Level: ");
48    Serial.print(waterLevelPercentage);
49    Serial.println(" %");
50
```

Figure 3.21-h : explication de programme de système d'eau

Ces lignes affichent les valeurs des capteurs (IR, pluie, distance, et pourcentage du niveau d'eau) sur le moniteur série pour le débogage.

```
52    if (!baselineSet) {
53      baselineWaterLevel = waterLevelPercentage;
54      baselineSet = true;
55      Serial.print("Baseline Water Level Set to: ");
56      Serial.println(baselineWaterLevel);
57    }
```

Figure 3.21-i : explication de programme de système d'eau

Définition de la référence : Lors de la première exécution, le niveau d'eau de référence est défini au pourcentage initial du niveau d'eau. Cela sert de point de référence pour détecter les changements du niveau d'eau.

```
59
60    bool waterLevelReduced = waterLevelPercentage <= baselineWaterLevel * 0.8;
61
62    bool rainDetected = (rainValue > 300);
63
```

Figure 3.21-j : explication de programme de système d'eau

waterLevelReduced : Vérifie si le niveau d'eau a baissé de 20 % ou plus par rapport à la référence.

rainDetected : Si la valeur du capteur de pluie dépasse 300, cela indique que de la pluie a été détectée.

```
71    if (waterLevelReduced || rainDetected) {
72
73      digitalWrite(RELAY_PIN, LOW);
74      Serial.println("Valve Closed due to Water Level Reduced or Rain Detected");
75    } else {
76
77      if (irValue < 100) {
78        digitalWrite(RELAY_PIN, HIGH);
79        delay(5000);
80        digitalWrite(RELAY_PIN, LOW);
81        Serial.println("Valve Opened (IR Sensor High and No Rain Detected)");
82      } else {
83        digitalWrite(RELAY_PIN, LOW);
84        Serial.println("Valve Closed (IR Sensor Not High)");
85      }
86    }
```

Figure 3.21-k : explication de programme de système d'eau

Réduction du niveau d'eau ou détection de pluie : Si le niveau d'eau a diminué ou si la pluie est détectée, le relais est éteint (LOW), ce qui probablement ferme une vanne, et un message est imprimé dans le moniteur série.

Vérification du capteur IR : Si aucune pluie n'est détectée et que la valeur du capteur IR est inférieure à 100, le relais est activé (HIGH) pendant 5 secondes, ouvrant la vanne temporairement. Sinon, la vanne reste fermée.

Résumé

- Le code lit les entrées d'un capteur IR, d'un capteur de pluie et d'un capteur de distance ultrasonique pour surveiller les niveaux d'eau et la présence de pluie.
- Il calcule le niveau d'eau en pourcentage et le compare à un niveau de référence initial.
- Si le niveau d'eau baisse de manière significative ou si la pluie est détectée, le relais (vanne) se ferme.
- Si les conditions sont normales et que le capteur IR détecte une activité, le relais ouvre la vanne pendant une courte période.
- Le système journalise toutes les données des capteurs et les actions dans le moniteur série pour

III.4.3. SYSTEM DE TEMPERATURE EST HUMIDITER :

a. PROGRAMME COMPLET

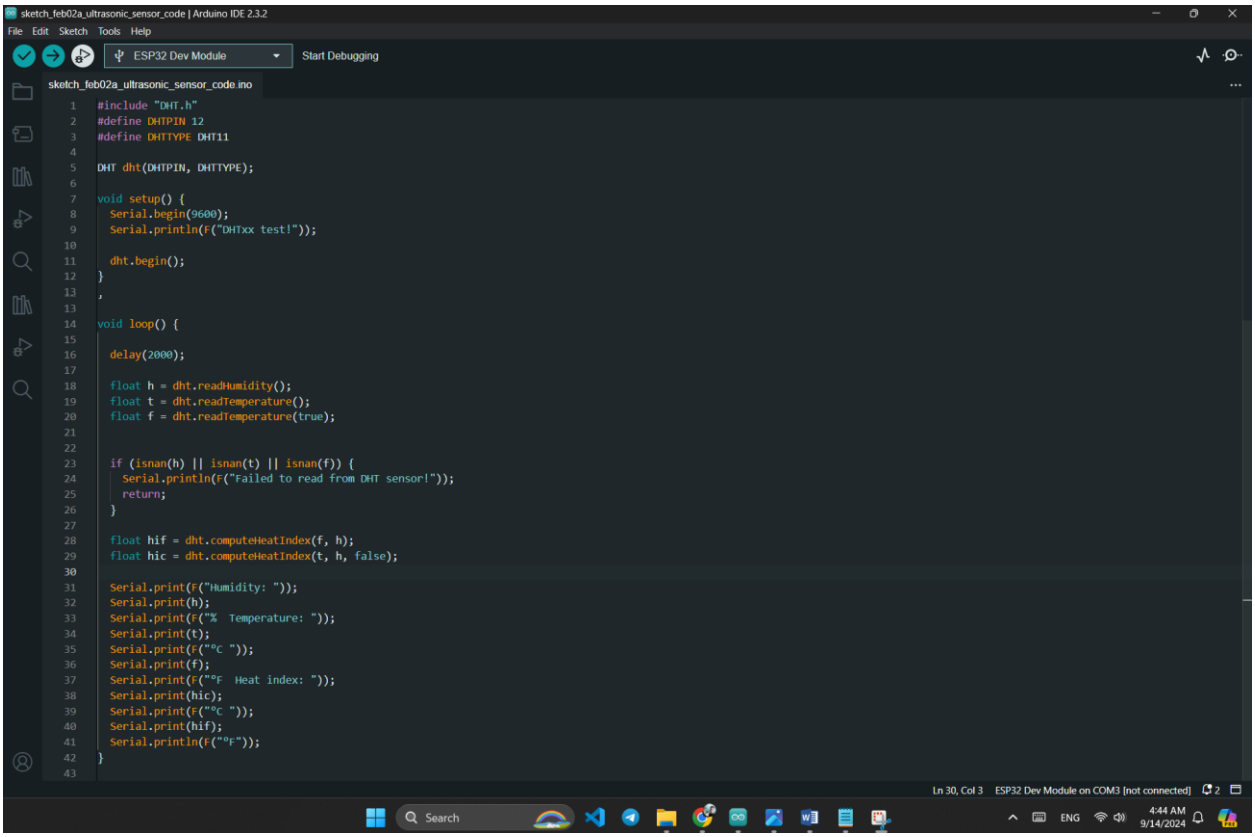


Figure 3.22 : programme de système de température et humidité

b. EXPLICATION DE CODE

```
1   #include "DHT.h"
```

Figure 3.22-a : programme de système de température et humidité

#include "DHT.h" : Cette ligne inclut la bibliothèque DHT, qui fournit des fonctions pour interagir avec les capteurs DHT comme le DHT11 et le DHT22. Ces capteurs sont utilisés pour mesurer la température et l'humidité.

```
2   #define DHTPIN 12
3   #define DHTTYPE DHT11
```

Figure 3.22-b : programme de système de température et humidité

DHTPIN (12) : Définit la broche à laquelle la broche de données du capteur DHT est connectée, dans ce cas, la broche GPIO 12 du microcontrôleur.

DHTTYPE (DHT11) : Spécifie le type de capteur DHT utilisé. Il s'agit d'un **DHT11**, un capteur basique de température et d'humidité.

```
5   DHT dht(DHTPIN, DHTTYPE);
```

Figure 3.22-c : programme de système de température et humidité

DHT dht(DHTPIN, DHTTYPE) : Cela crée un objet dht de la classe DHT. L'objet est initialisé avec la broche de données et le type de capteur, permettant au programme de lire les données du capteur DHT.

```
7   void setup() {
8     Serial.begin(9600);
9     Serial.println(F("DHTxx test!"));
10
11    dht.begin();
12  }
```

Figure 3.22-d : programme de système de température et humidité

Serial.begin(9600) : Initialise la communication série à un débit de 9600 bits par seconde. Cela permet d'afficher des messages dans le moniteur série.

Serial.println(F("Test du DHTxx !")) : Affiche le message "Test du DHTxx !" dans le moniteur série pour indiquer que le programme a démarré.

dht.begin() : Initialise le capteur DHT, le rendant prêt à lire des données.

```
14  void loop() {
15
16    delay(2000);
```

Figure 3.22-e : programme de système de température et humidité

delay(2000) : Le programme s'arrête pendant 2000 millisecondes (2 secondes) entre chaque cycle de boucle. Le capteur DHT a un taux d'échantillonnage lent (environ 1 lecture par seconde), donc ce délai permet de ne pas le surcharger.

```
18    float h = dht.readHumidity();
19    float t = dht.readTemperature();
20    float f = dht.readTemperature(true);
21
```

Figure 3.22-f : programme de système de température et humidité

float h = dht.readHumidity() : Lit la valeur de l'humidité du capteur DHT et la stocke dans la variable h. Le résultat est un nombre flottant représentant l'humidité en pourcentage.

float t = dht.readTemperature() : Lit la température en degrés Celsius du capteur DHT et la stocke dans la variable t.

float f = dht.readTemperature(true) : Lit la température en Fahrenheit en passant true comme paramètre. Cette valeur est stockée dans la variable f.

```
23  if (isnan(h) || isnan(t) || isnan(f)) {
24    Serial.println(F("Failed to read from DHT sensor!"));
25    return;
26  }
```

Figure 3.22-g : programme de système de température et humidité

isnan(h), isnan(t), isnan(f) : Ces fonctions vérifient si les lectures d'humidité ou de température renvoient **NaN** (Not-a-Number). Cela pourrait se produire si le capteur ne parvient pas à lire des données.

Si l'une des valeurs est NaN : Le programme affiche "Échec de la lecture du capteur DHT !" dans le moniteur série et sort de la boucle avec return.

```
28  float hif = dht.computeHeatIndex(f, h);
29  float hic = dht.computeHeatIndex(t, h, false);
```

Figure 3.22-h : programme de système de température et humidité

float hif = dht.computeHeatIndex(f, h) : Calcule l'**indice de chaleur** (ou température ressentie) en Fahrenheit, basé sur la température f (en Fahrenheit) et l'humidité h.

float hic = dht.computeHeatIndex(t, h, false) : Calcule l'indice de chaleur en Celsius, basé sur la température t (en Celsius) et l'humidité h. Le paramètre false indique que la température est en Celsius.

```
30
31    Serial.print(F("Humidity: "));
32    Serial.print(h);
33    Serial.print(F("%  Temperature: "));
34    Serial.print(t);
35    Serial.print(F("°C "));
36    Serial.print(f);
37    Serial.print(F("°F  Heat index: "));
38    Serial.print(hic);
39    Serial.print(F("°C "));
40    Serial.print(hif);
41    Serial.println(F("°F"));
42  }
```

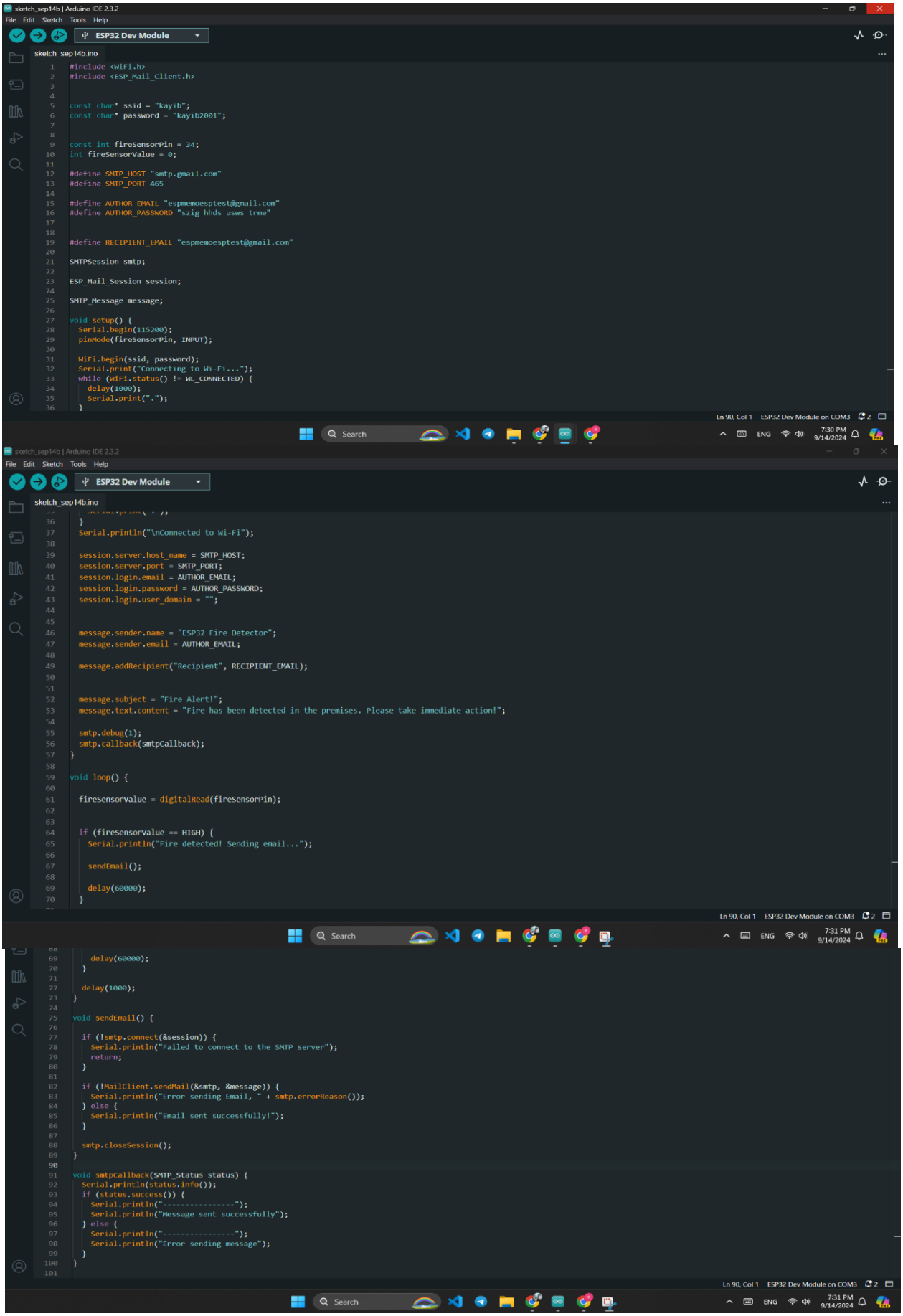
Figure 3.22-i : programme de système de température et humidité

Serial.print() et Serial.println() : Ces fonctions affichent l'humidité, la température (en Celsius et en Fahrenheit), et l'indice de chaleur (en Celsius et en Fahrenheit) dans le moniteur série.

F() est utilisé pour stocker les chaînes de caractères dans la mémoire **Flash** (au lieu de la RAM) afin d'économiser de l'espace.

III.4.4. System détection de flamme :

a. Programme complet :



```
#include <WiFi.h>
#include <ESP_Mail_Client.h>


const char* ssid = "kayib";
const char* password = "kayib2001";


const int fireSensorPin = 34;
int fireSensorValue = 0;

#define SMTP_HOST "smtp.gmail.com"
#define SMTP_PORT 465

#define AUTHOR_EMAIL "espmemoesptest@gmail.com"
#define AUTHOR_PASSWORD "szig hhds usws trme"


#define RECIPIENT_EMAIL "espmemoesptest@gmail.com"

SMTPSession smtp;

ESP_Mail_Session session;

SMTP_Message message;

void setup() {
  Serial.begin(115200);
  pinMode(fireSensorPin, INPUT);

  WiFi.begin(ssid, password);
  Serial.print("Connecting to Wi-Fi...");
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("\nConnected to Wi-Fi");

  session.server.host_name = SMTP_HOST;
  session.server.port = SMTP_PORT;
  session.login.email = AUTHOR_EMAIL;
  session.login.password = AUTHOR_PASSWORD;
  session.login.user_domain = "";


  message.sender.name = "ESP32 Fire Detector";
  message.sender.email = AUTHOR_EMAIL;

  message.addRecipient("Recipient", RECIPIENT_EMAIL);


  message.subject = "Fire Alert!";
  message.text.content = "Fire has been detected in the premises. Please take immediate action!";

  smtp.debug(1);
  smtp.callback(smtpCallback);
}

void loop() {

  fireSensorValue = digitalRead(fireSensorPin);


  if (fireSensorValue == HIGH) {
    Serial.println("Fire detected! Sending email...");

    sendEmail();

    delay(60000);
  }

  delay(1000);
}

void sendEmail() {

  if (!smtp.connect(&session)) {
    Serial.println("Failed to connect to the SMTP server");
    return;
  }

  if (!MailClient.sendMail(&smtp, &message)) {
    Serial.println("Error sending Email, " + smtp.errorReason());
  } else {
    Serial.println("Email sent successfully!");
  }

  smtp.closeSession();
}

void smtpCallback(SMTP_Status status) {
  Serial.println(status.info());
  if (status.success()) {
    Serial.println("----------------");
    Serial.println("Message sent successfully");
  } else {
    Serial.println("----------------");
    Serial.println("Error sending message");
  }
}
```

Figure 3.23 : programme de système de détection de flamme

b. **explication de code :**

```
1   #include <WiFi.h>
2   #include <ESP_Mail_Client.h>
```

Figure 3.23-a : programme de système de détection de flamme

#include <WiFi.h> : Cette bibliothèque permet à l'ESP32 de se connecter à un réseau Wi-Fi.

#include <ESP_Mail_Client.h> : Cette bibliothèque est utilisée pour envoyer des e-mails via l'ESP32 en se connectant à un serveur SMTP (Simple Mail Transfer Protocol).

```
5   const char* ssid = "kayib";
6   const char* password = "kayib2001";
```

Figure 3.23-b : programme de système de détection de flamme

ssid et password : Contiennent les identifiants du réseau Wi-Fi auquel l'ESP32 va se connecter.

```
9   const int fireSensorPin = 34;
10  int fireSensorValue = 0;
```

Figure 3.23-c : programme de système de détection de flamme

fireSensorPin = 34 : Le pin de l'ESP32 (GPIO 34) est connecté au capteur de feu.
fireSensorValue : Stocke la valeur lue depuis le capteur.

```
12  #define SMTP_HOST "smtp.gmail.com"
13  #define SMTP_PORT 465
14
15  #define AUTHOR_EMAIL "espmemoesptest@gmail.com"
16  #define AUTHOR_PASSWORD "szig hhds usws trme"
17
18
19  #define RECIPIENT_EMAIL "espmemoesptest@gmail.com"
```

Figure 3.23-d : programme de système de détection de flamme

- **SMTP_HOST** : Le serveur SMTP de Gmail, utilisé pour envoyer des e-mails.
- **SMTP_PORT = 465** : Le port utilisé pour le SMTP via SSL (Secure Socket Layer).
- **AUTHOR_EMAIL** : L'adresse e-mail de l'expéditeur (celle utilisée pour envoyer les alertes).
- **AUTHOR_PASSWORD** : Le mot de passe spécifique à l'application pour ce compte Gmail (nécessaire pour l'authentification).
- **RECIPIENT_EMAIL** : L'adresse e-mail du destinataire qui recevra l'alerte incendie.

```
21  SMTPSession smtp;
22
23  ESP_Mail_Session session;
24
25  SMTP_Message message;
```

Figure3.23-e : programme de système de détection de flamme

- **smtp** : Représente la session SMTP pour gérer l'envoi des e-mails.
- **session** : Contient les informations du serveur et de connexion pour la session d'e-mail.
- **message** : Contient le contenu de l'e-mail (expéditeur, destinataire, sujet, corps du message).

```
27  void setup() {
28    Serial.begin(115200);
29    pinMode(fireSensorPin, INPUT);
30
31    WiFi.begin(ssid, password);
32    Serial.print("Connecting to Wi-Fi...");
33    while (WiFi.status() != WL_CONNECTED) {
34      delay(1000);
35      Serial.print(".");
36    }
37    Serial.println("\nConnected to Wi-Fi");
38
```

Figure 3.23-f : programme de système de détection de flamme

- **Serial.begin(115200)** : Initialise le moniteur série pour le débogage avec un débit de 115200 bauds.
- **pinMode(fireSensorPin, INPUT)** : Définit le pin du capteur de feu (34) comme une entrée.
- **WiFi.begin(ssid, password)** : Démarre la connexion au réseau Wi-Fi avec le SSID et le mot de passe spécifiés.
- **while (WiFi.status() != WL_CONNECTED)** : Boucle jusqu'à ce que l'ESP32 soit connecté au Wi-Fi, en affichant des points (".") toutes les secondes.
- Une fois connecté, un message de confirmation est imprimé : "Connecté au Wi-Fi".

```
39    session.server.host_name = SMTP_HOST;
40    session.server.port = SMTP_PORT;
41    session.login.email = AUTHOR_EMAIL;
42    session.login.password = AUTHOR_PASSWORD;
43    session.login.user_domain = "";
44
```

Figure 3.23-g : programme de système de détection de flamme

Configuration de la session d'e-mail : Ces lignes configurent le serveur SMTP et les identifiants de connexion pour la session d'envoi d'e-mail.

```
46  message.sender.name = "ESP32 Fire Detector";
47  message.sender.email = AUTHOR_EMAIL;
48
49  message.addRecipient("Recipient", RECIPIENT_EMAIL);
50
51
52  message.subject = "Fire Alert!";
53  message.text.content = "Fire has been detected in the premises. Please take immediate action!";
```

Figure 3.23-h : programme de système de détection de flamme

Configuration du message : Ces lignes définissent le nom et l'e-mail de l'expéditeur, le nom et l'e-mail du destinataire, l'objet de l'e-mail, et le contenu du message (alerte incendie).

```
smtp.debug(1);
smtp.callback(smtpCallback);
```

Figure 3.23-i : programme de système de détection de flamme

smtp.debug(1) : Active la sortie de débogage pour le processus SMTP.
smtp.callback(smtpCallback) : Associe une fonction de rappel (smtpCallback) pour gérer les mises à jour du statut lors de l'envoi de l'e-mail.

```
59  void loop() {
60
61    fireSensorValue = digitalRead(fireSensorPin);
62
63
64    if (fireSensorValue == HIGH) {
65      Serial.println("Fire detected! Sending email...");
66
67      sendEmail();
68
69      delay(60000);
70    }
71
72    delay(1000);
73  }
```

Figure 3.23-j : programme de système de détection de flamme

- **fireSensorValue = digitalRead(fireSensorPin)** : Lit l'état du capteur de feu. Si la valeur est élevée (HIGH), cela indique une détection d'incendie.
- **Si un incendie est détecté** : L'ESP32 imprime un message et appelle la fonction sendEmail() pour envoyer l'alerte. La boucle attend alors 60 secondes (delay(60000)) avant de vérifier à nouveau le capteur.
- **Si aucun incendie n'est détecté** : La boucle attend 1 seconde (delay(1000)) avant de vérifier à nouveau la valeur du capteur.

```
75  void sendEmail() {
76
77    if (!smtp.connect(&session)) {
78      Serial.println("Failed to connect to the SMTP server");
79      return;
80    }
81
82    if (!MailClient.sendMail(&smtp, &message)) {
83      Serial.println("Error sending Email, " + smtp.errorReason());
84    } else {
85      Serial.println("Email sent successfully!");
86    }
87
88    smtp.closeSession();
89  }
```

Figure 3.23-k : programme de système de détection de flamme

- **smtp.connect(&session)** : Tente de se connecter au serveur SMTP en utilisant la configuration de la session. Si cela échoue, un message d'erreur est imprimé.
- **MailClient.sendMail(&smtp, &message)** : Envoie l'e-mail en utilisant la session SMTP et le message configurés. Si cela échoue, une erreur est imprimée.
- **smtp.closeSession()** : Ferme la session SMTP après l'envoi de l'e-mail.

```
91  void smtpCallback(SMTP_Status status) {
92    Serial.println(status.info());
93    if (status.success()) {
94      Serial.println("----------------");
95      Serial.println("Message sent successfully");
96    } else {
97      Serial.println("----------------");
98      Serial.println("Error sending message");
99    }
100 }
```

Figure 3.23-l : programme de système de détection de flamme

- **smtpCallback()** : Une fonction de rappel qui est appelée chaque fois qu'il y a une mise à jour du statut pendant l'envoi de l'e-mail. Elle affiche les informations sur le statut et indique si l'e-mail a été envoyé avec succès ou non.

Résumé

- Ce programme se connecte à un réseau Wi-Fi, lit les entrées d'un capteur de feu, et envoie un e-mail d'alerte en cas de détection d'incendie. Il utilise la capacité de l'ESP32 à envoyer des e-mails via un serveur SMTP et lit périodiquement le capteur pour vérifier la présence de feu.

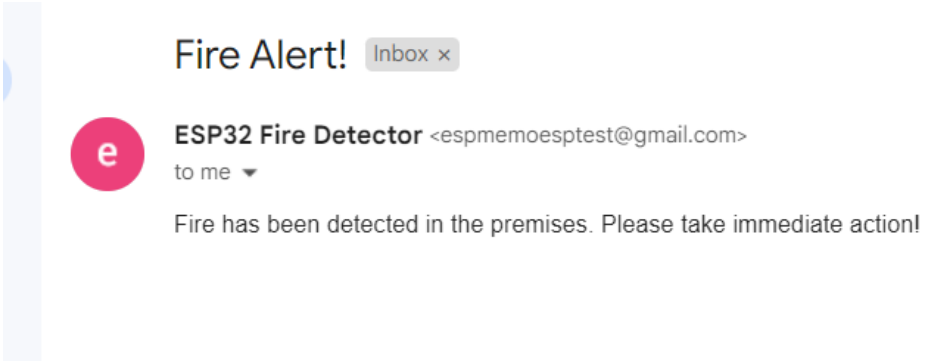


Figure 3.24 : reçoit de message de capteur de flamme sur mai [36]

III.4.5. L'APPLICATION D'ESP32 :(ESP 32.LOCAL) Visual studio [21]

a. Conception d'application :

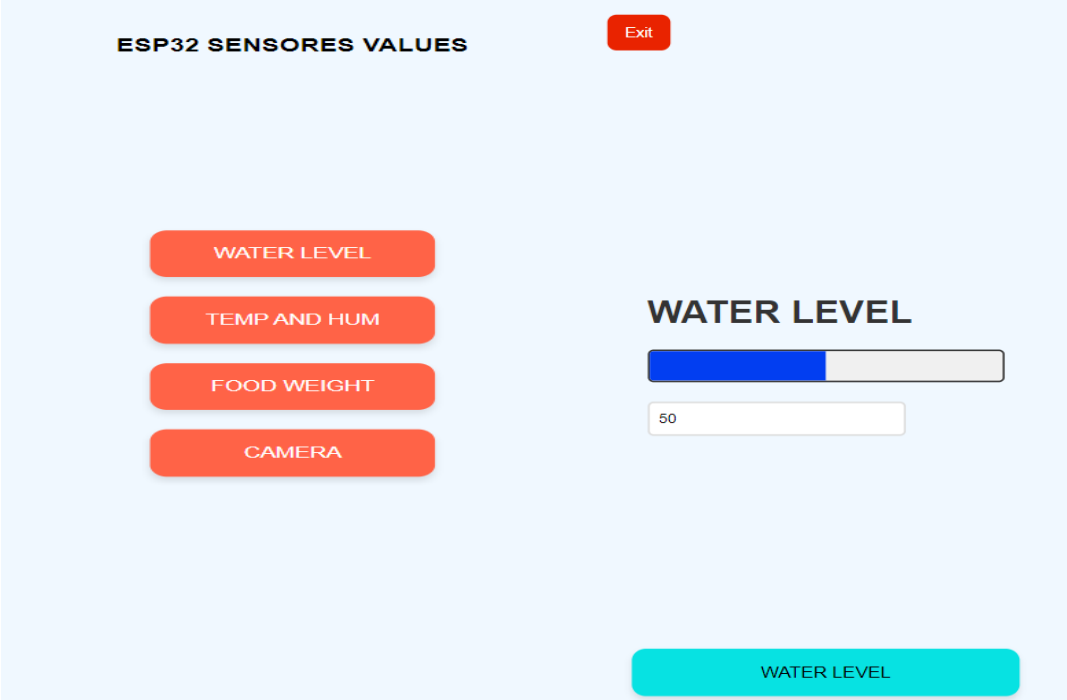


FIGURE 3.25-a : interface de l'application et conception de bouton water level

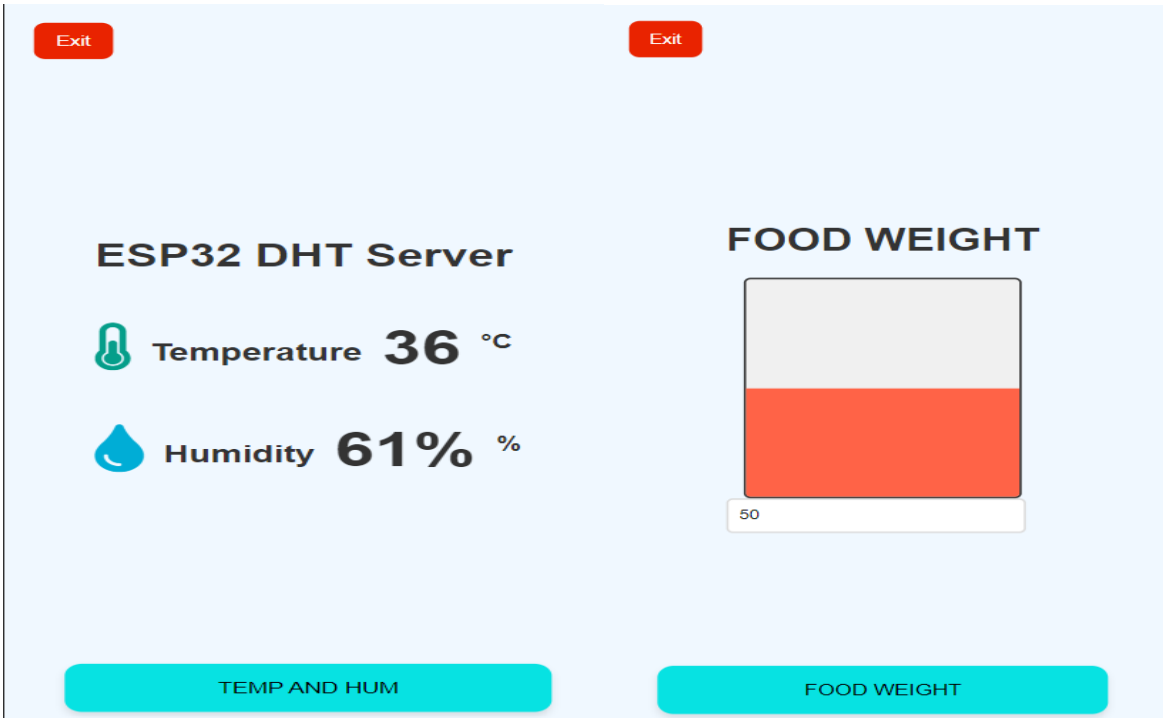


FIGURE 3.25-b : conception de bouton température, humidité et food weight

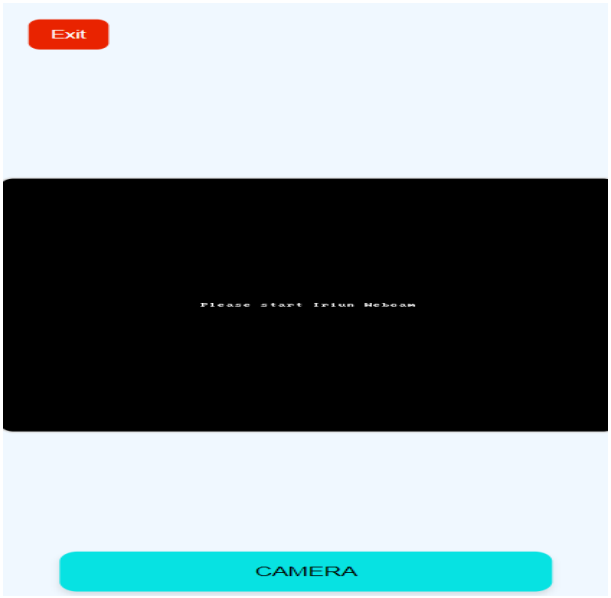
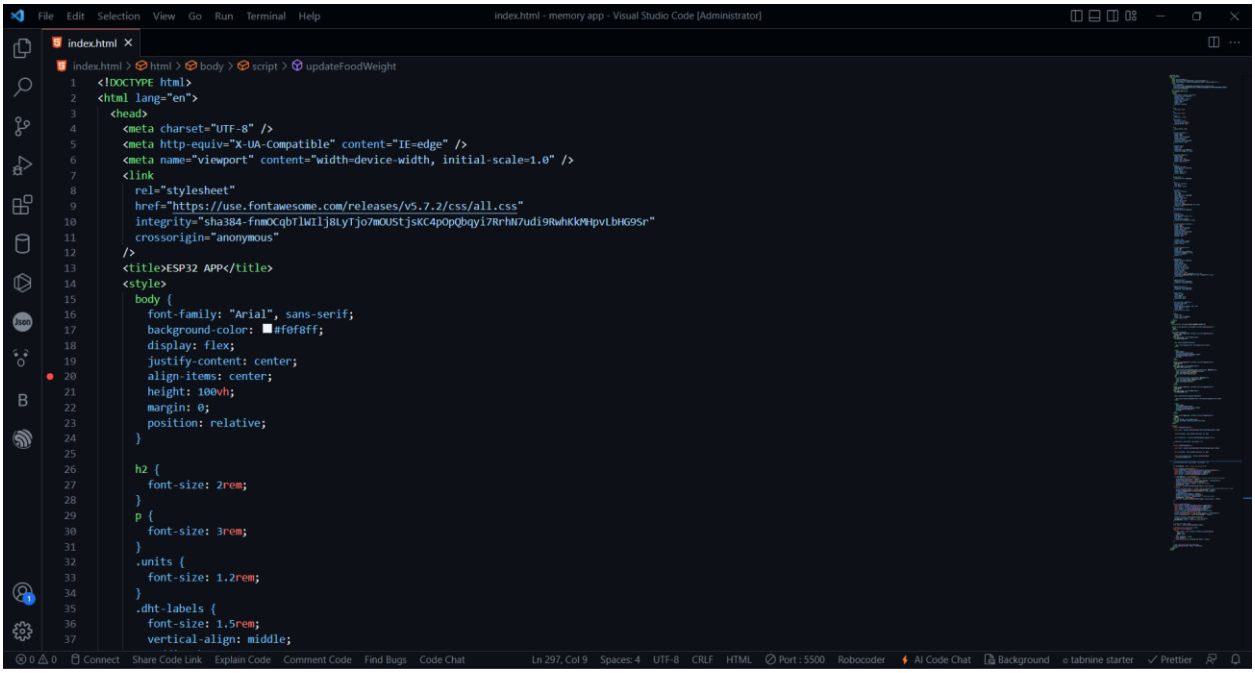


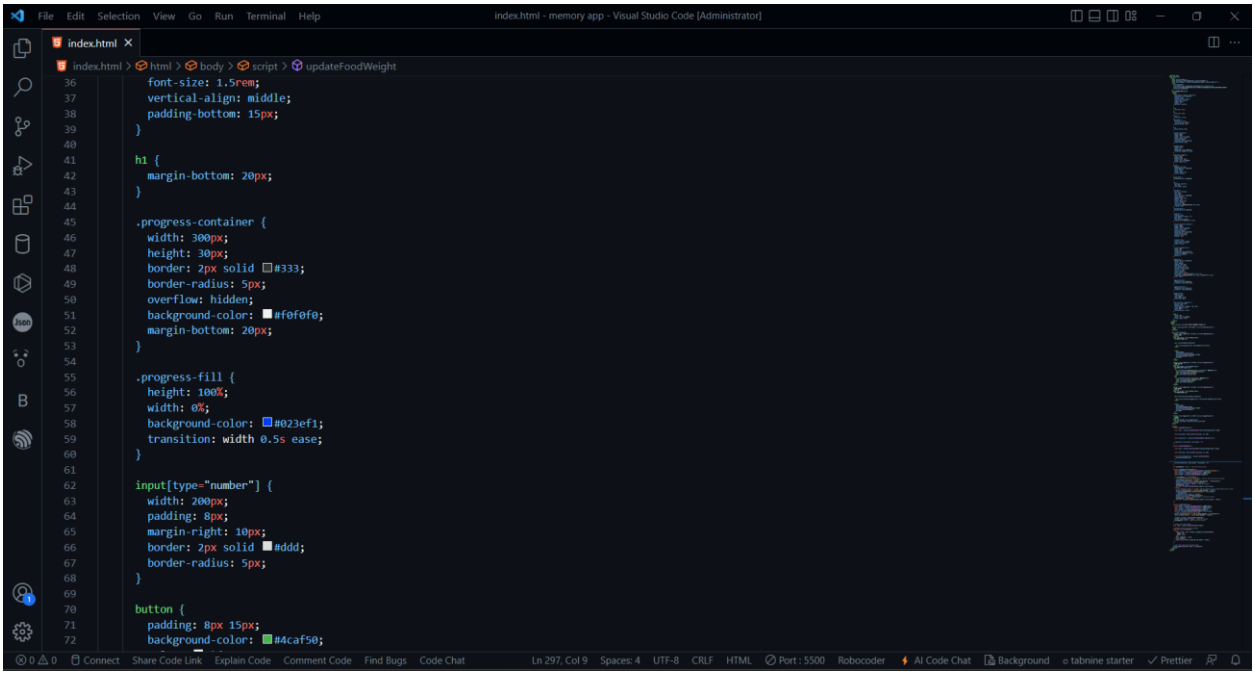
FIGURE 3.25-c : conception de bouton camera

b. LE CODE COMPLET DE L'APPLICATION :



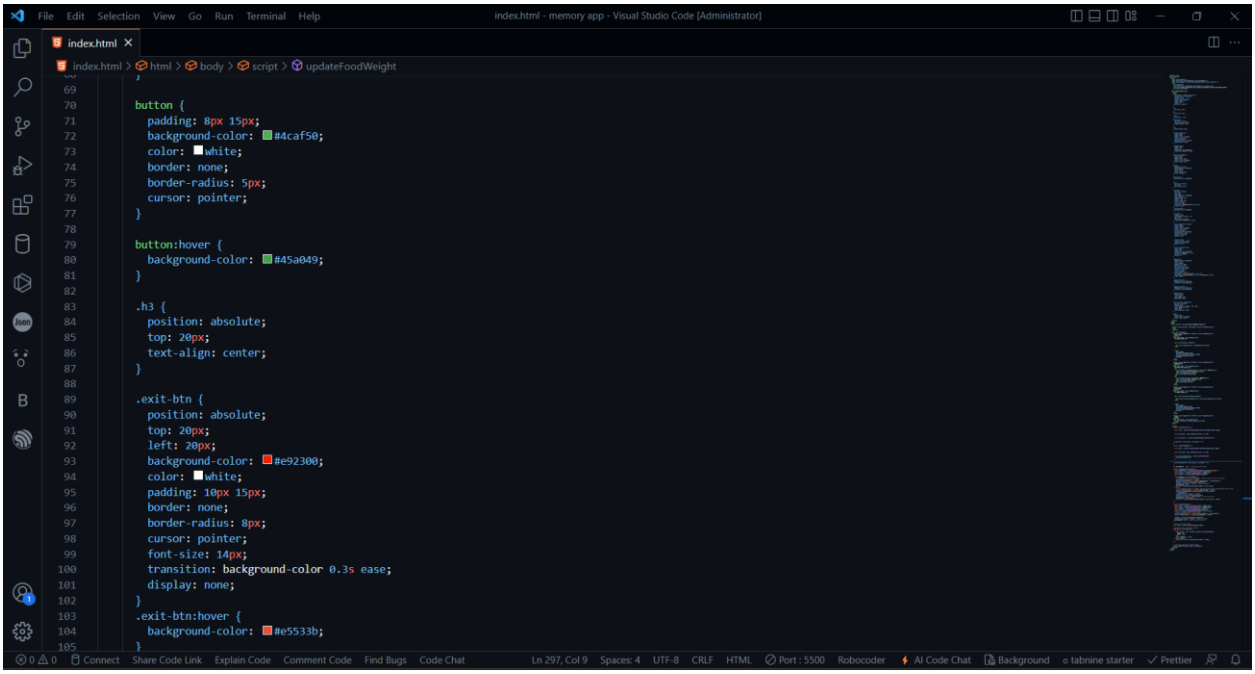
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link
      rel="stylesheet"
      href="https://use.fontawesome.com/releases/v5.7.2/css/all.css"
      integrity="sha384-fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr"
      crossorigin="anonymous"
    />
    <title>ESP32 APP</title>
    <style>
      body {
        font-family: "Arial", sans-serif;
        background-color: #f0f8ff;
        display: flex;
        justify-content: center;
        align-items: center;
        height: 100vh;
        margin: 0;
        position: relative;
      }

      h2 {
        font-size: 2rem;
      }
      p {
        font-size: 3rem;
      }
      .units {
        font-size: 1.2rem;
      }
      .dht-labels {
        font-size: 1.5rem;
        vertical-align: middle;
```

FIGURE 3.26-a : programme d'application



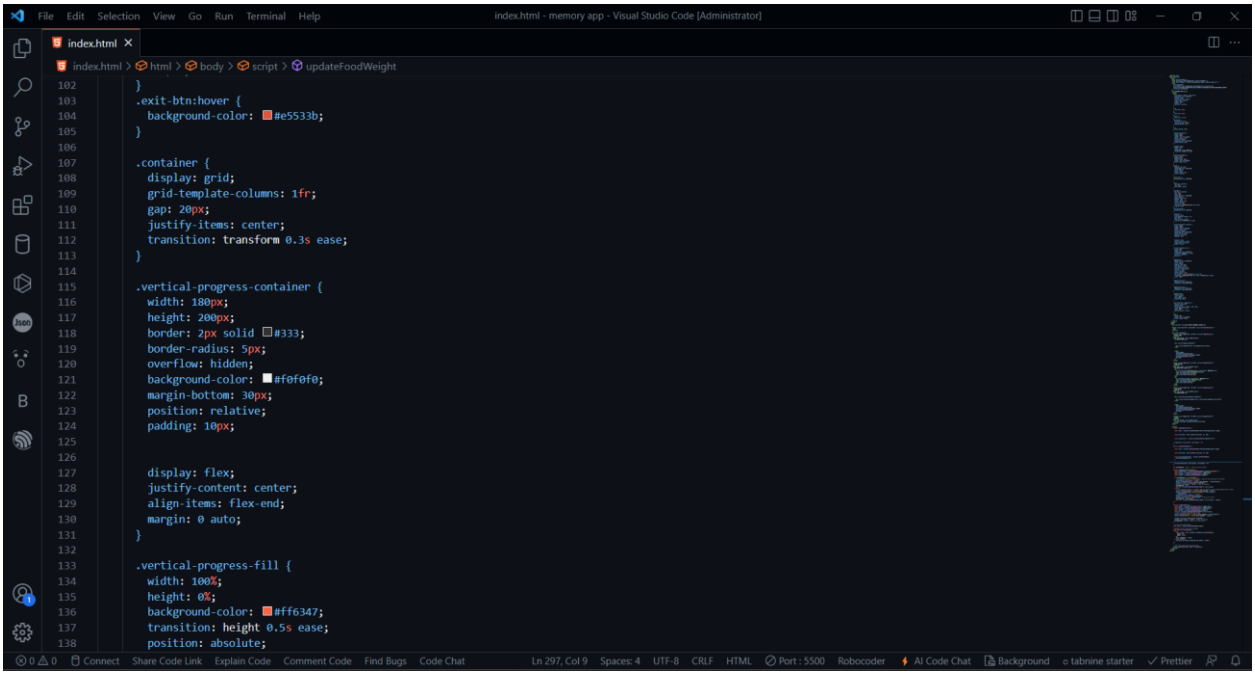
```
        font-size: 1.5rem;
        vertical-align: middle;
        padding-bottom: 15px;
      }

      h1 {
        margin-bottom: 20px;
      }

      .progress-container {
        width: 300px;
        height: 30px;
        border: 2px solid #333;
        border-radius: 5px;
        overflow: hidden;
        background-color: #f0f0f0;
        margin-bottom: 20px;
      }

      .progress-fill {
        height: 100%;
        width: 0%;
        background-color: #023ef1;
        transition: width 0.5s ease;
      }

      input[type="number"] {
        width: 200px;
        padding: 8px;
        margin-right: 10px;
        border: 2px solid #ddd;
        border-radius: 5px;
      }

      button {
        padding: 8px 15px;
        background-color: #4caf50;
```

FIGURE 3.26-b : programme d'application



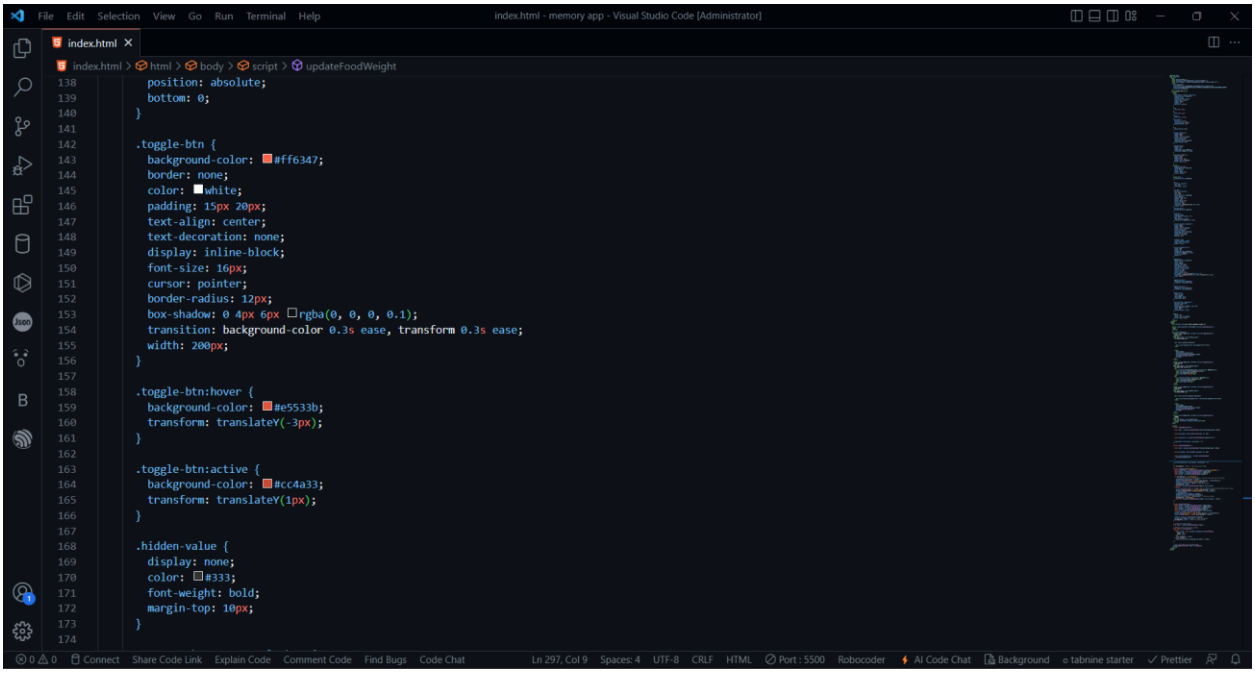
```
button {
  padding: 8px 15px;
  background-color: #4caf50;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #45a049;
}

.h3 {
  position: absolute;
  top: 20px;
  text-align: center;
}

.exit-btn {
  position: absolute;
  top: 20px;
  left: 20px;
  background-color: #e92300;
  color: white;
  padding: 10px 15px;
  border: none;
  border-radius: 8px;
  cursor: pointer;
  font-size: 14px;
  transition: background-color 0.3s ease;
  display: none;
}
.exit-btn:hover {
  background-color: #e5533b;
}
```

FIGURE 3.26-c : programme d'application



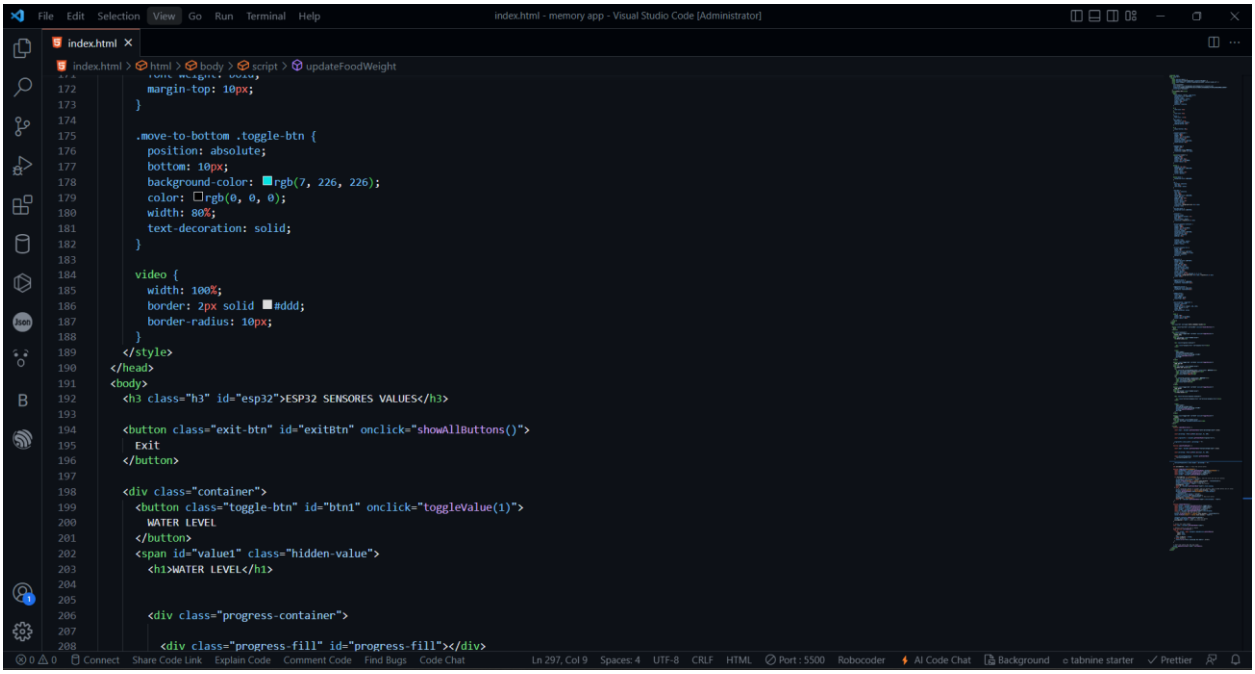
```
}
.exit-btn:hover {
  background-color: #e5533b;
}

.container {
  display: grid;
  grid-template-columns: 1fr;
  gap: 20px;
  justify-items: center;
  transition: transform 0.3s ease;
}

.vertical-progress-container {
  width: 180px;
  height: 200px;
  border: 2px solid #333;
  border-radius: 5px;
  overflow: hidden;
  background-color: #f0f0f0;
  margin-bottom: 30px;
  position: relative;
  padding: 10px;


  display: flex;
  justify-content: center;
  align-items: flex-end;
  margin: 0 auto;
}

.vertical-progress-fill {
  width: 100%;
  height: 0%;
  background-color: #ff6347;
  transition: height 0.5s ease;
  position: absolute;
```

FIGURE 3.26-d : programme d'application



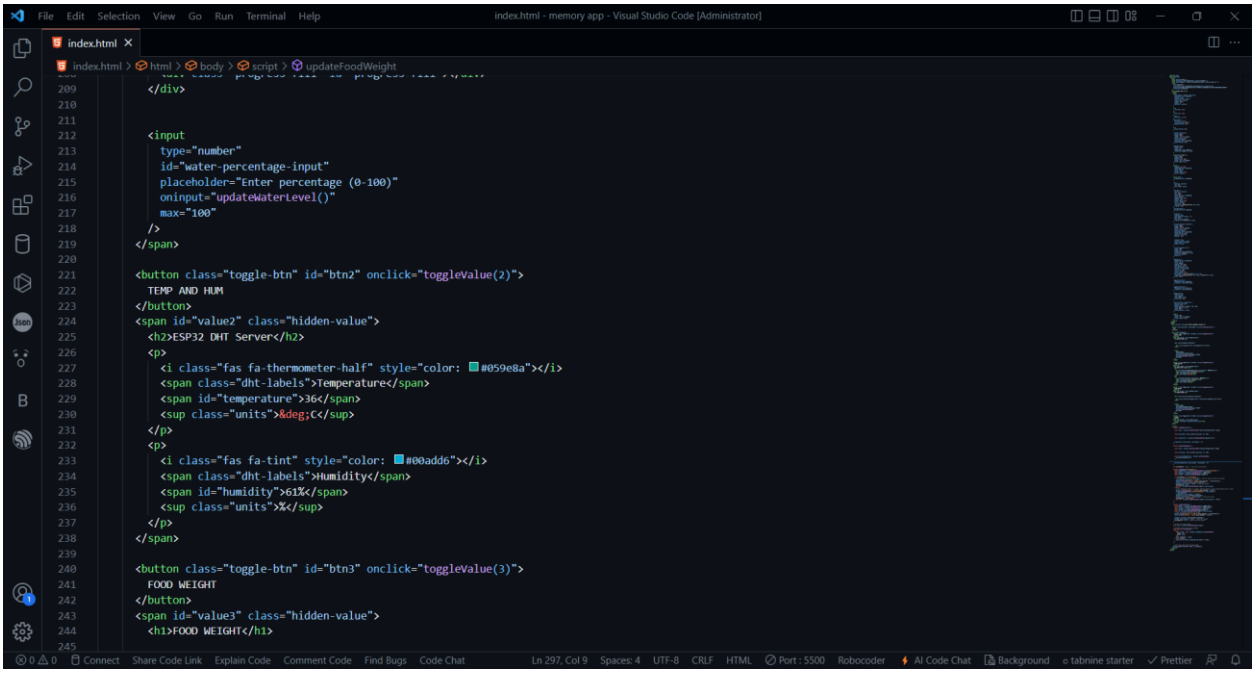
```
      position: absolute;
      bottom: 0;
    }

    .toggle-btn {
      background-color: #ff6347;
      border: none;
      color: white;
      padding: 15px 20px;
      text-align: center;
      text-decoration: none;
      display: inline-block;
      font-size: 16px;
      cursor: pointer;
      border-radius: 12px;
      box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
      transition: background-color 0.3s ease, transform 0.3s ease;
      width: 200px;
    }

    .toggle-btn:hover {
      background-color: #e5533b;
      transform: translateY(-3px);
    }

    .toggle-btn:active {
      background-color: #cc4a33;
      transform: translateY(1px);
    }

    .hidden-value {
      display: none;
      color: #333;
      font-weight: bold;
      margin-top: 10px;
    }
```

FIGURE 3.26-e : programme d'application



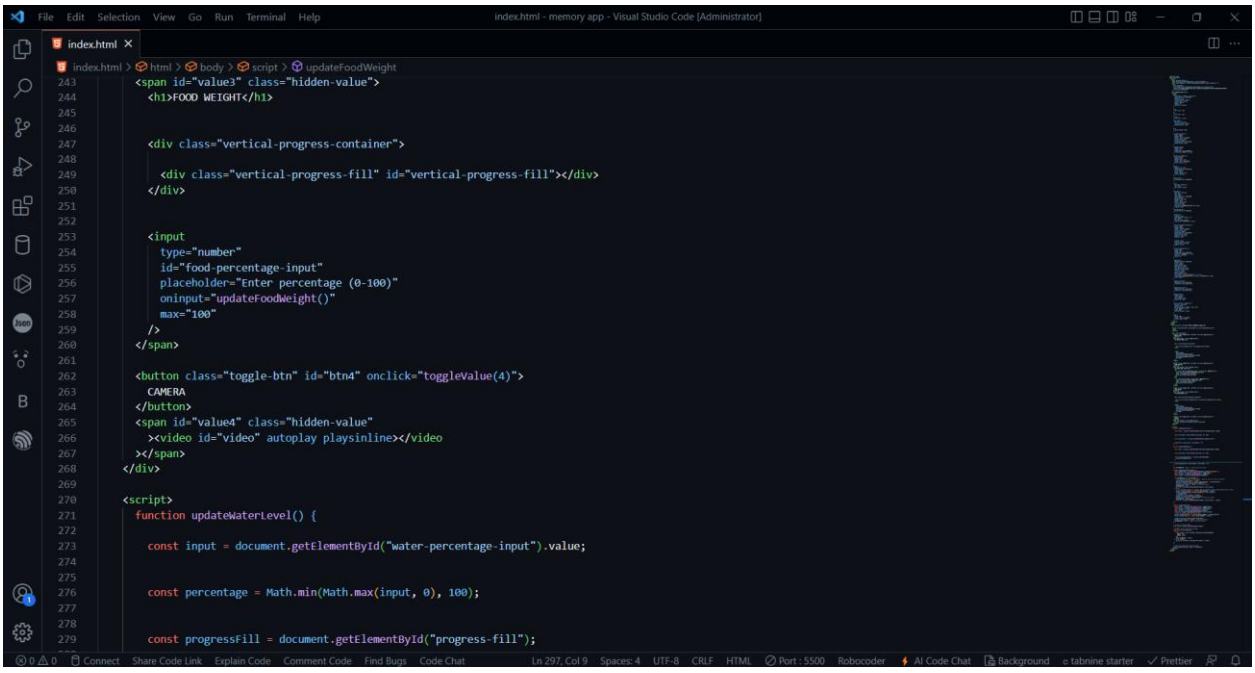
```
      margin-top: 10px;
    }

    .move-to-bottom .toggle-btn {
      position: absolute;
      bottom: 10px;
      background-color: rgb(7, 226, 226);
      color: rgb(0, 0, 0);
      width: 80%;
      text-decoration: solid;
    }

    video {
      width: 100%;
      border: 2px solid #ddd;
      border-radius: 10px;
    }
  </style>
</head>
<body>
  <h3 class="h3" id="esp32">ESP32 SENSORES VALUES</h3>

  <button class="exit-btn" id="exitBtn" onclick="showAllButtons()">
    Exit
  </button>

  <div class="container">
    <button class="toggle-btn" id="btn1" onclick="toggleValue(1)">
      WATER LEVEL
    </button>
    <span id="value1" class="hidden-value">
      <h1>WATER LEVEL</h1>


      <div class="progress-container">

        <div class="progress-fill" id="progress-fill"></div>
```

FIGURE 3.26-f : programme d'application



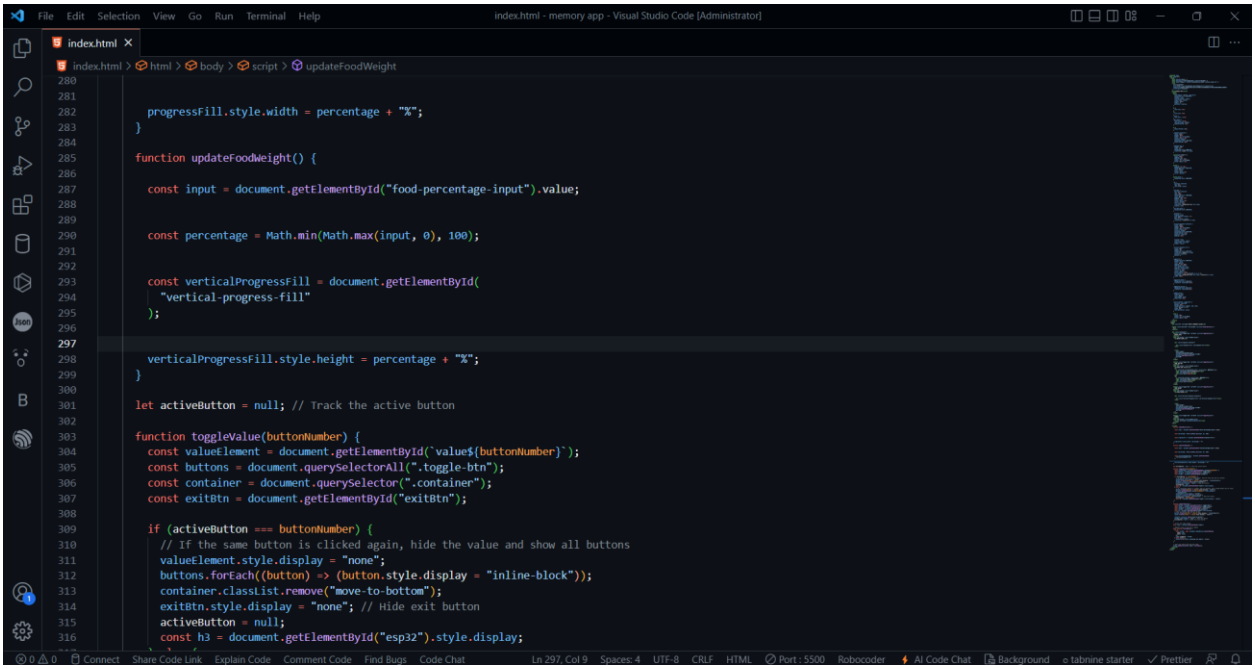
```
      </div>


      <input
        type="number"
        id="water-percentage-input"
        placeholder="Enter percentage (0-100)"
        oninput="updateWaterLevel()"
        max="100"
      />
    </span>

    <button class="toggle-btn" id="btn2" onclick="toggleValue(2)">
      TEMP AND HUM
    </button>
    <span id="value2" class="hidden-value">
      <h2>ESP32 DHT Server</h2>
      <p>
        <i class="fas fa-thermometer-half" style="color: #059e8a"></i>
        <span class="dht-labels">Temperature</span>
        <span id="temperature">36</span>
        <sup class="units">&deg;C</sup>
      </p>
      <p>
        <i class="fas fa-tint" style="color: #00add6"></i>
        <span class="dht-labels">Humidity</span>
        <span id="humidity">61%</span>
        <sup class="units">%</sup>
      </p>
    </span>

    <button class="toggle-btn" id="btn3" onclick="toggleValue(3)">
      FOOD WEIGHT
    </button>
    <span id="value3" class="hidden-value">
      <h1>FOOD WEIGHT</h1>
```

FIGURE 3.26-g : programme d'application



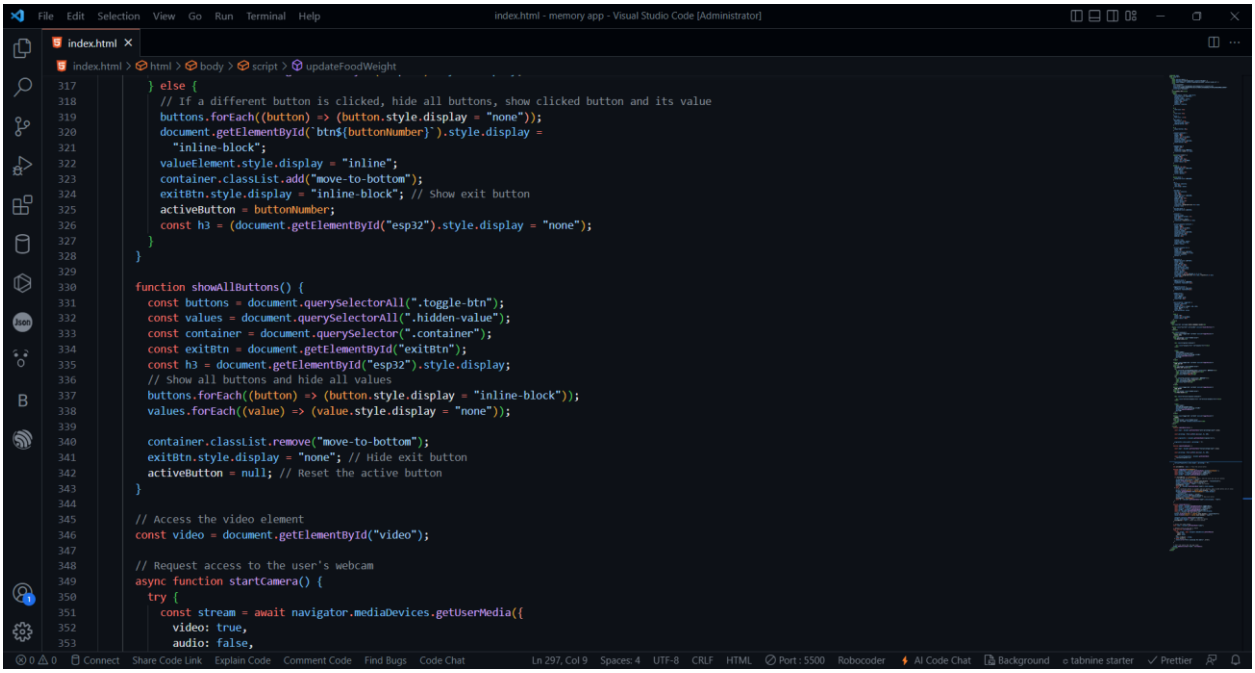
```
    <span id="value3" class="hidden-value">
      <h1>FOOD WEIGHT</h1>


      <div class="vertical-progress-container">

        <div class="vertical-progress-fill" id="vertical-progress-fill"></div>
      </div>


      <input
        type="number"
        id="food-percentage-input"
        placeholder="Enter percentage (0-100)"
        oninput="updateFoodWeight()"
        max="100"
      />
    </span>

    <button class="toggle-btn" id="btn4" onclick="toggleValue(4)">
      CAMERA
    </button>
    <span id="value4" class="hidden-value"
      ><video id="video" autoplay playsinline></video
    ></span>
  </div>

  <script>
    function updateWaterLevel() {

      const input = document.getElementById("water-percentage-input").value;


      const percentage = Math.min(Math.max(input, 0), 100);


      const progressFill = document.getElementById("progress-fill");
```

FIGURE 3.26-h : programme d'application



```
progressFill.style.width = percentage + "%";
}

function updateFoodWeight() {

  const input = document.getElementById("food-percentage-input").value;


  const percentage = Math.min(Math.max(input, 0), 100);


  const verticalProgressFill = document.getElementById(
    "vertical-progress-fill"
  );


  verticalProgressFill.style.height = percentage + "%";
}

let activeButton = null; // Track the active button

function toggleValue(buttonNumber) {
  const valueElement = document.getElementById(`value${buttonNumber}`);
  const buttons = document.querySelectorAll(".toggle-btn");
  const container = document.querySelector(".container");
  const exitBtn = document.getElementById("exitBtn");

  if (activeButton === buttonNumber) {
    // If the same button is clicked again, hide the value and show all buttons
    valueElement.style.display = "none";
    buttons.forEach((button) => (button.style.display = "inline-block"));
    container.classList.remove("move-to-bottom");
    exitBtn.style.display = "none"; // Hide exit button
    activeButton = null;
    const h3 = document.getElementById("esp32").style.display;
```

FIGURE 3.26-i : programme d'application



```
  } else {
    // If a different button is clicked, hide all buttons, show clicked button and its value
    buttons.forEach((button) => (button.style.display = "none"));
    document.getElementById(`btn${buttonNumber}`).style.display =
      "inline-block";
    valueElement.style.display = "inline";
    container.classList.add("move-to-bottom");
    exitBtn.style.display = "inline-block"; // Show exit button
    activeButton = buttonNumber;
    const h3 = (document.getElementById("esp32").style.display = "none");
  }
}

function showAllButtons() {
  const buttons = document.querySelectorAll(".toggle-btn");
  const values = document.querySelectorAll(".hidden-value");
  const container = document.querySelector(".container");
  const exitBtn = document.getElementById("exitBtn");
  const h3 = document.getElementById("esp32").style.display;
  // Show all buttons and hide all values
  buttons.forEach((button) => (button.style.display = "inline-block"));
  values.forEach((value) => (value.style.display = "none"));

  container.classList.remove("move-to-bottom");
  exitBtn.style.display = "none"; // Hide exit button
  activeButton = null; // Reset the active button
}

// Access the video element
const video = document.getElementById("video");

// Request access to the user's webcam
async function startCamera() {
  try {
    const stream = await navigator.mediaDevices.getUserMedia({
      video: true,
      audio: false,
```

FIGURE 3.26-j : programme d'application

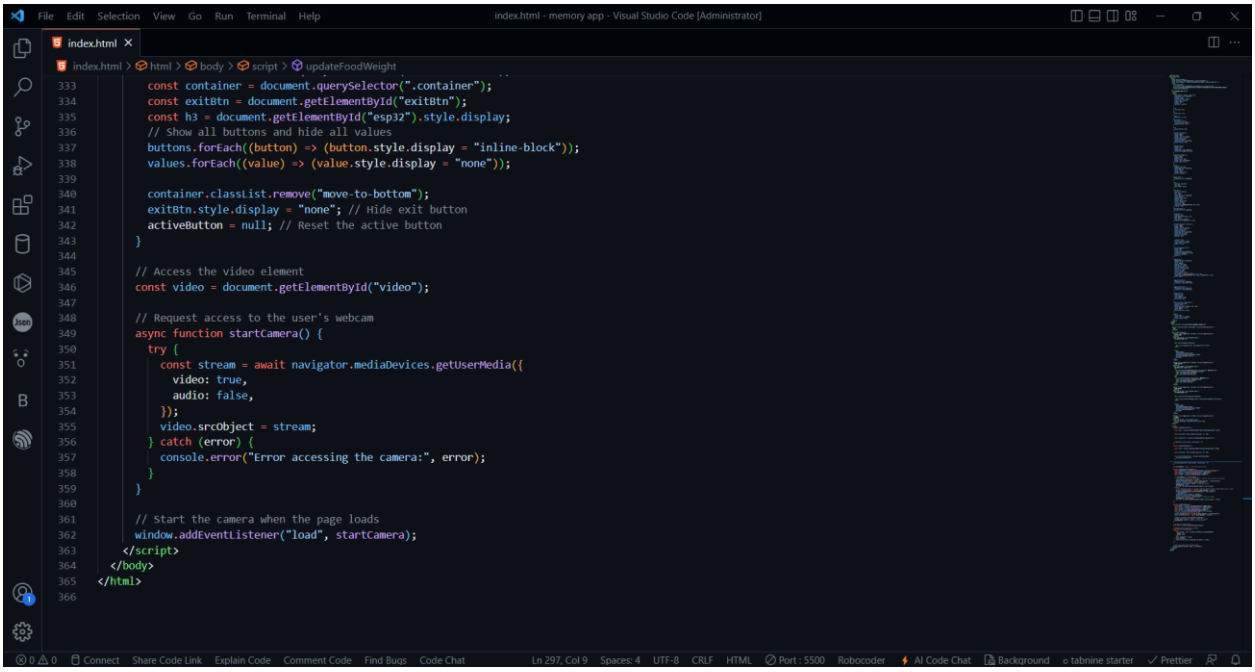


FIGURE 3.26-k : programme d'application

c. EXPLICATION DE CODE :

1. Structure HTML

- **Déclaration** : <!DOCTYPE html> définit le document comme HTML5.
- **Langue** : <html lang="fr"> indique que le contenu est en français.
- **En-tête** (<head>) : Contient les métadonnées, les liens vers les styles, et le titre de la page.
- **Corps** (<body>) : Contient le contenu visible, y compris les titres, boutons, et vidéos.

2. Styles CSS

- **Général** : Le style de la page est défini pour centrer le contenu, ajuster les tailles, couleurs, et les effets des boutons.
- **Barres de Progression** : Styles pour les barres de progression horizontales et verticales, y compris les transitions douces.

3. Contenu HTML

- **Boutons** : Quatre boutons permettent de basculer entre différents affichages (niveau d'eau, température/humidité, poids des aliments, caméra).
- **Affichages Cachés** : Chaque bouton révèle des informations spécifiques lorsque cliqué.

4. JavaScript

- **Mise à Jour des Valeurs** : Les fonctions updateWaterLevel et updateFoodWeight ajustent les barres de progression en fonction des entrées de l'utilisateur.
- **Basculement du Contenu** : toggleValue montre ou cache les informations en fonction du bouton cliqué.
- **Affichage des Boutons** : showAllButtons réaffiche tous les boutons et cache les informations.
- **Caméra** : startCamera accède à la webcam et affiche le flux vidéo.

En résumé, le code crée une interface utilisateur avec des boutons pour afficher différentes informations et une vidéo en direct depuis la webcam.

III.5. Discussion de prototype proposé :

Le prototype proposé représente une solution innovante pour automatiser la gestion des ressources pour les animaux à l'aide de l'ESP 32 qui vas commander tous les composants **:**

Partie Alimentation:

Le système d'alimentation se compose principalement d'un moteur pas à pas, conçu pour ouvrir un mécanisme de distribution d'aliments. Ce moteur est piloté par un capteur de poids, qui surveille en temps réel la quantité d'aliment disponible. Si le capteur indique que le poids est inférieur à 1 g, le moteur s'active automatiquement pour libérer une portion d'aliment pour la vache. Cette approche garantit que les animaux reçoivent leur nourriture de manière régulière et contrôlée. En revanche, lorsque le poids d'aliment dépasse 50 g, le moteur se ferme automatiquement, évitant ainsi une distribution excessive. Parallèlement, un capteur à ultrasons est intégré pour mesurer le niveau d'aliment dans le réservoir, assurant une gestion précise des stocks.

Partie Eau :

Dans le système d'hydratation, un capteur infrarouge est positionné pour détecter la présence de la vache. Dès que ce capteur confirme la présence de l'animal, l'électrovanne s'ouvre automatiquement, permettant l'accès à l'eau. Cependant, cette ouverture est conditionnée par un capteur de niveau d'eau, qui vérifie la disponibilité de l'eau dans le réservoir. Si le niveau d'eau est insuffisant, l'électrovanne ne s'ouvrira pas, prévenant ainsi tout gaspillage d'eau. De plus, un capteur à ultrasons est également utilisé pour mesurer le niveau d'eau, offrant une redondance dans le système de surveillance.

Partie Protection :

Le système de protection repose sur deux capteurs essentiels :

a. **Capteur de Température et d'Humidité** : Ce capteur surveille en permanence les conditions environnementales. En cas d'augmentation de la chaleur ou de fluctuations climatiques importantes, il envoie une alerte, permettant ainsi une intervention rapide pour protéger les animaux contre des conditions potentiellement dangereuses.

b. **Capteur de Flamme** : Ce capteur est crucial pour la sécurité des installations. Il détecte toute présence de feu dans l'usine et déclenche une alerte immédiate, informant le propriétaire des risques d'incendie. Cela permet de mettre en œuvre des mesures de sécurité adéquates, minimisant ainsi les pertes potentielles.

III.6. Synthèse des résultats

- **Intégration des Composants** :

L'utilisation de l'ESP32 permet une connectivité sans fil et une gestion centralisée des capteurs et actionneurs. Cela facilite l'interaction avec le système et la collecte de données en temps réel.

- **Efficacité Énergétique** :

Le choix de composants à faible consommation, comme le DHT11 et le HC-SR04, contribue à un fonctionnement durable. De plus, l'ESP32 est conçu pour être économe en énergie, ce qui est crucial pour les systèmes embarqués.

- **Surveillance et Sécurité** :

Le système de détection de flamme est un ajout important pour assurer la sécurité des animaux et des installations. L'envoi d'alertes par e-mail permet une réponse rapide en cas d'urgence.

- **Interface Utilisateur** :

L'interface conviviale permet aux utilisateurs de suivre et de contrôler facilement le système. La possibilité de visualiser des données en temps réel améliore l'expérience utilisateur et permet des ajustements rapides.

- **Scalabilité** :

Le prototype est conçu pour être évolutif. De nouveaux capteurs ou fonctionnalités peuvent être ajoutés en fonction des besoins futurs, comme des caméras pour la surveillance ou des capteurs de poids supplémentaires.

- **Applications Élargies** :

Au-delà des applications agricoles, ce système pourrait être adapté à d'autres contextes, comme la gestion des ressources en milieu urbain ou dans des installations de recherche.

- **Défis Techniques:**

Des défis peuvent survenir, notamment en termes de connectivité réseau dans des zones rurales ou éloignées. Il est essentiel d'évaluer la robustesse du système dans différents environnements.

III.7. Conclusion :

Ce projet met en place un système automatisé pour la gestion de l'alimentation et de l'eau des animaux, basé sur l'ESP32. Il utilise divers capteurs, tels que le DHT11 pour la température et l'humidité, et le HC-SR04 pour mesurer le niveau d'eau. Les relais et électrovannes contrôlent l'eau et les aliments, tandis qu'un moteur pas à pas gère leur distribution.

Une interface utilisateur permet de visualiser en temps réel les données et de contrôler le système à distance. En cas de détection de flamme, des alertes par e-mail sont envoyées, garantissant la sécurité. Ce projet illustre l'application des technologies modernes pour améliorer la gestion des ressources et le bien-être animal.

Conclusion général

En conclusion, la relation entre les animaux, leur alimentation et les écosystèmes est essentielle au maintien de l'équilibre naturel et de la biodiversité. À travers les siècles, la domestication des animaux a profondément influencé les sociétés humaines, tant sur le plan alimentaire qu'économique. Aujourd'hui, avec l'essor des technologies avancées, nous avons la possibilité d'améliorer la gestion des ressources animales de manière durable et efficace.

Le projet proposé, qui intègre capteurs, actionneurs et systèmes automatisés tels que l'ESP32 et Raspberry Pi, montre comment l'innovation technologique peut optimiser la gestion de l'alimentation et de l'hydratation des animaux. En réduisant l'intervention humaine et en garantissant un suivi en temps réel, ce système contribue à la sécurité alimentaire tout en améliorant le bien-être animal. De plus, la personnalisation et la surveillance à distance permettent une gestion des ressources plus intelligente, illustrant l'avenir de l'agriculture moderne axée sur l'efficacité et la durabilité.

De plus, nous avons pu réaliser un prototype concret, démontrant la faisabilité et l'efficacité de ce projet. Ce prototype met en avant l'importance d'une gestion automatisée de l'alimentation et de l'hydratation des animaux, et ouvre la voie à de futures améliorations et évolutions du système. La réussite de ce prototype montre qu'il est tout à fait possible d'intégrer davantage de fonctionnalités, telles que des capteurs plus sophistiqués ou des systèmes d'intelligence artificielle, afin d'optimiser encore davantage les performances et la gestion des ressources.

Les recherches approfondies que nous avons menées dans le cadre de ce projet nous ont permis d'acquérir une expertise précieuse dans ce domaine. Elles nous ont offert une meilleure compréhension des défis et des opportunités liés à l'automatisation dans le secteur agricole. De plus, elles nous ont ouvert des perspectives pour le développement de systèmes encore plus complexes et innovants, capables de résoudre divers problèmes pratiques que nous rencontrons au quotidien, que ce soit dans la gestion des fermes, l'amélioration des conditions de vie des animaux ou l'optimisation des ressources en eau et en nourriture.

Ainsi, ce projet ne se limite pas à la simple gestion de l'alimentation animale ; il représente une avancée vers une agriculture plus intelligente, plus durable et mieux adaptée aux besoins actuels. Les connaissances et l'expérience acquises nous incitent à poursuivre dans cette voie, en développant de nouvelles solutions technologiques capables d'améliorer l'efficacité et la durabilité des pratiques agricoles, tout en répondant aux enjeux de sécurité alimentaire et de bien-être animal à long terme.

Recommandations :

Recommandation 1 :

- **Problème ciblé** : WIFI pour le reçoit des messages
- **Action suggérée** : GSM
- **Justification** : utilisation de GSM donne la possibilité de recevait des messages dans n'importe quel place ou des problèmes dans le wifi

Recommandation 2 :

- **Problème ciblé** : défaillance des composants
- **Action suggérée** : utilisation des composants plus fiables
- **Justification** : malgré le cout des composants il vous donnera une langue durabilité

Recommandation 3 :

- **Problème ciblé** : l'électricité
- **Action suggérée** : utilisation des plaques solaires ou groupe électrique
- **Justification** : pour que les animaux reçoit leur repart à temps

Bibliographiques

[1] https://www.agrimaroc.net/ consulté le 25/08/2024

[2] Bou maza O., Houri M.M, « système de dosage automatique pour les aliments bétail » Mémoire de master en Automatique, Univ. Said Dahled, Bilida, Juin 2022.

[3] https://f2school.com/wp-content/uploads/2020/08/Capteurs-cours-01.pdf consulté le 25/08/2024

[4] Taibi D., Djebiri A., "Etude et réalisation d'un compteur d'énergie intelligent '', mémoire de Master, Département de Génie électrique, Univ. Ouargla 2020.

[5] https://wokwi.com/projects/new/esp32 consulté le 23/09/2024

[6] https://learn.microsoft.com/fr-fr/visualstudio/windows/?view=vs-2022 consulté le 24/09/2024

[7] https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022 consulté le 24/09/2024

[8] Rajput, Aditya & Magdum, Pushpadant & Kumbhar, Dinesh & Patil, Shweta & Rajput, Abhilash & Dhawale, Nandkishor. (2021). Automatic Animal Food Feeding System. 3. 69. 10.35629/5252-03106976.

[9] José O. Chelotti a b, Luciano S. Martinez-Rau 2024, Livestock feeding behaviour: A review on automated systems for ruminant monitoring" Elsevier jornal, Biosystems Engineering, Volume 246, 2024, Pages 150-177.