



وزارة البحث العلمي والتعليم العالي
MINISTERE DE L'ENSEIGNEMENT SUPEREUR
ET DE LA RECHERCHE SCIENTIFIQUE
Université Abdelhamid Ibn Badis Mostaganem
جامعة عبد الحميد بن باديس مستغانم
Faculté des Sciences et de la Technologie
كلية العلوم والتكنولوجيا
DEPARTEMENT DE GENIE ELECTRIQUE



Mémoire Pour l'obtention du diplôme de Master

Filière: Electronique

Spécialités: Electronique des systèmes embarquées

Sous thème :

Conception et réalisation d'un système multitâche de gestion intelligente du confort du climat d'une serre agricole.

Présenté et soutenu publiquement par:

-DAHDOUHI FATIMA ZAHRA

-BOUCEDRA FATMA

Devant le jury composé de:

Nom et Prénom	Grade	Établissement	Qualité
YAGOUBI Ben Abdallah	Professeur	Université de Mostaganem	Président
ABDELLAOUI Nasreddine	MAA	Université de Mostaganem	Encadrant
BENOUDNINE Hadjira	Professeur	Université de Mostaganem	Examineur
BERRADJA Khadidja	MCA	Université de Mostaganem	Co-encadreur

Année universitaire: 2023-2024

Remerciements

Tout d'abord, nous remercions **Allah** de nous avoir donné la capacité et la volonté et la patience d'aller jusqu'au bout de nos études.

Nous tenons à remercier de tout notre cœur Mr.**ABDELLAOUI Nasraddine**, notre encadrant de mémoire, pour son soutien, son expertise et ses conseils tout au long de notre parcours. Son dévouement et son engagement envers notre développement professionnel ont été inestimables.

Nous exprimons également notre gratitude aux **membres du jury** pour avoir accepté d'évaluer notre mémoire.

Nous n'oublierons pas **nos parents** pour leur contribution, leur soutien et leur patience. Nous tenons à exprimer notre reconnaissance envers eux.

Nous tenons à exprimer notre gratitude envers tous ceux qui ont contribué, de près ou de loin, à la réalisation de ce modeste travail.

Merci à toutes et à tous.

Dédicace

Je dédie ce modeste travail:

À mes parents, merci infiniment d'être toujours présents pour moi, de me féliciter
quand tout allait

bien et de me soutenir quand c'était nécessaire. Je n'en serais pas là sans eux.

A mes chères sœurs **Amina Imane Saadia Raziqa.**

A ceux qui m'ont soutenu **Abd Eldjallel**

À mon neveu **Ibrahim.**

À toute la famille **DAHDOUHI** et **Alola.**

À mes amies, en particulier à ma copine **Asma.**

À ma meilleur collègue **FATMA.**

À mon cher grand-père décédé.

Dahdouhi fatima

Dédicace

Je dédie ce mémoire :

À mes parents, qui m'ont toujours soutenue dans toutes mes entreprises et ont été une source constante d'inspiration et d'amour.

À mes frères, Djamel, Ammar et Walide, pour leur soutien indéfectible et leurs encouragements tout au long de ce chemin académique.

À toute la famille **BOUCEDRA** et **HAMCHERIF**.

À ma binôme et ma meilleure amie **Houda**.

À mes amies proches Chaima et Zineb , pour leur précieuse amitié, leur soutien inconditionnel et les moments de joie partagés qui ont rendu ce parcours encore plus mémorable.

Merci à vous tous.

BOUCEDRA Fatma

Résumé du Projet

Le projet vise à concevoir et réaliser un système multitâche pour la gestion intelligente du confort climatique d'une serre agricole. En intégrant des technologies avancées telles que l'IoT, l'Edge Computing, le Big Data et le Cloud Computing, le système collecte, traite et analyse les données environnementales en temps réel pour optimiser les conditions de croissance des plantes. Des capteurs mesurent des paramètres comme la température, l'humidité et la luminosité, tandis que des actionneurs ajustent les systèmes de chauffage, de ventilation et d'irrigation en conséquence. L'objectif est de maximiser la productivité agricole tout en minimisant les ressources utilisées, contribuant ainsi à une agriculture plus durable et efficace.

يهدف المشروع إلى تصميم وتنفيذ نظام متعدد المهام لإدارة ذكية لراحة المناخ في دفيئة زراعية. من خلال دمج تقنيات متقدمة مثل إنترنت الأشياء (IoT) والحوسبة الطرفية (Edge Computing) والبيانات الضخمة (Big Data) والحوسبة السحابية (Cloud Computing)، يقوم النظام بجمع ومعالجة وتحليل البيانات البيئية في الوقت الحقيقي لتحسين ظروف نمو النباتات. تقيس المستشعرات معلمات مثل درجة الحرارة والرطوبة والإضاءة، بينما تقوم المحركات بضبط أنظمة التدفئة والتهوية والري وفقاً لذلك. الهدف هو زيادة الإنتاجية الزراعية إلى أقصى حد مع تقليل استخدام الموارد، مما يساهم في زراعة أكثر استدامة وفعالية.

The project aims to design and implement a multitasking system for intelligent climate comfort management in an agricultural greenhouse. By integrating advanced technologies such as IoT, Edge Computing, Big Data, and Cloud Computing, the system collects, processes, and analyzes environmental data in real-time to optimize plant growth conditions. Sensors measure parameters like temperature, humidity, and light, while actuators adjust heating, ventilation, and irrigation systems accordingly. The objective is to maximize agricultural productivity while minimizing resource use, contributing to more sustainable and efficient farming.

SOMMAIRE

RESUME DU PROJET

SOMMAIRE

LISTE DES ABBREVIATIONS

LISTE DES FIGURES

LISTES DES TABLEAUX

INTRODUCTION GENERALE	1
CHAPITRE I : L'IOT, L'EDGE COMPUTING, LE BIG DATA, LE CLOUD COMPUTING ET LEURS RAPPORTS AVEC LA SERRICULTURE	3
I.1. INTRODUCTION	3
I.2. L'IOT :	3
I.2.1. Définitions de l'IoT	4
I.2.2. Caractéristiques d'un réseau IoT	4
I.2.2.1. La Connectivité des objets IoT	4
I.2.2.2. L'identité des objets IoT:	5
I.2.2.3. Le partage de données entre les objets IoT	5
I.2.2.4. L'Intelligence embarquée sur les objets IoT:	6
I.2.2.5. L'Évolutivité des objets de l'IoT :	6
I.2.2.6. L'Architecture de l'IoT :	6
I.2.2.7. La sécurité des objets IoT	7
I.2.3. Fonctionnement d'un réseau IoT	7
I.2.4. Quelques domaines applicatifs de l'IoT	7
I.2.5. Structure du réseau IoT	8
I.2.6. Les technologies de l'IoT et la serriculture	9
I.2.7. Domaines d'utilisation des composants IoT d'une serre IoT	9
I.3. LE BIG DATA	12
I.3.1. Définitions	12
I.3.2. Rapports avec la serriculture	12
I.4. L'EDGE COMPUTING	13
I.4.1. Définition	13
I.4.2. Rapport avec la serriculture intelligente	13
I.5. LE CLOUD COMPUTING	13
I.5.1. Définition	13
I.5.2. Rapport avec l'agriculture sous serre	13
I.6. CONCLUSION	13
CHAPITRE II : GENERALITES SUR LA LOGIQUE FLOUE	14
II.1. INTRODUCTION	14
II.2. DEFINITIONS	15
II.3. HISTORIQUE DE LA LOGIQUE FLOUE	16
II.4. AVANTAGES DE LA LOGIQUE FLOUE	16
II.5. INCONVENIENTS DE LA LOGIQUE FLOUE	17
II.6. LES DOMAINES D'APPLICATIONS	17
II.7. LES DOMAINES OU IL NE FAUT PAS UTILISER LA LOGIQUE FLOUE	17
II.8. LES CONCEPTS DE LA LOGIQUE FLOUE	17
II.8.1. Le concept de valeur linguistique	17
II.8.2. Le concept de degré de vérité	17
II.8.3. Le concept de variable linguistique	18
II.8.4. Le concept de sous-ensembles flous:	18
II.8.5. Le concept d'ensembles de discours :	18

II.8.6. <i>Le concept de fonction d'appartenance</i> :	18
II.8.7. <i>Fonction d'appartenance d'entrée et fuzzification</i>	18
II.8.8. <i>Fonction d'appartenance de sortie, règles, inférence floue et dé fuzzification</i>	20
II.9. STRUCTURE ET FONCTIONNEMENT D'UN CONTROLEUR FLOU	27
II.10. STRUCTURE D'UN SYSTEME DE CONTROLE FLOU EN BOUCLE OUVERTE ET EN BOUCLE FERMEE	27
II.12. CONCLUSION	28
CHAPITRE III : LE MULTITACHE ET LES DIFFERENTES APPROCHES DE CONCEPTIONS D'APPLICATIONS EMBARQUEES	29
III.1. INTRODUCTION	29
III.2. DEFINITIONS	29
III.3. ARCHITECTURE D'UNE APPLICATION EMBARQUEE	29
III.4. LES DIFFERENTES APPROCHES DE CONCEPTION D'UNE APPLICATION EMBARQUEE MULTITACHE	30
III.4.1. <i>Approche de la boucle principale plus routines de service d'interruption</i>	30
III.4.2. <i>Approche basée sur les ordonnanceurs coopératifs</i>	31
III.4.3. <i>Approche basée sur l'utilisation d'un système d'exploitation type RTOS</i>	34
III.4.3.1. Les différents types de systèmes d'exploitation	34
III.4.3.2. Principe de fonctionnement d'un RTOS:	34
III.5. CONCLUSION	37
CHAPITRE IV : REALISATION MATERIELLE	37
IV.1. INTRODUCTION	37
IV.2. ARCHITECTURE MATERIELLE DU SYSTEME DE CONTROLE REALISE	37
IV.3. DESCRIPTION FONCTIONNELLE	38
IV.4. LES COMPOSANTS MATERIELS UTILISES POUR LA REALISATION DU SYSTEME DE CONTROLE	38
IV.4.1. <i>Le contrôleur flou de proximité et l'unité de commande des actionneurs</i>	38
IV.4.2. <i>Les capteurs</i> :	38
IV.4.2.1. <i>Le capteur de température et d'humidité relative de l'air</i> :	38
IV.4.2.2. <i>Le capteur humidité du sol</i>	39
IV.4.2.3. <i>Le capteur de lumière</i> :	40
IV.4.3. <i>Les actionneurs</i> :	40
IV.4.3.1. <i>L'irrigation</i>	40
IV.4.3.2. <i>L'éclairage</i>	42
IV.4.3.3. <i>l'aération et le refroidissement par ventilation</i>	43
IV.4.3.4. <i>L'ombrage</i>	43
IV.4.3.5. <i>Le chauffage</i>	43
IV.5. SCHEMAS DE L'INTERFAÇAGE DU CONTROLEUR FLOU AVEC LES CAPTEURS ET LES ACTIONNEURS UTILISES	44
IV.5. 1. <i>Schéma de l'interfaçage du contrôleur flou avec les capteurs utilisés : (figure IV.12)</i>	44
IV.5. 2. <i>Interfaçage du contrôleur flou avec les actionneurs utilisés : (Figure IV.13)</i>	45
IV.6. VUE DE LA MAQUETTE	46
IV.7. CONCLUSION :	47
CHAPITRE V : REALISATION LOGICIELLE	48
V.1. INTRODUCTION	48
V.2. LES FONCTIONS D'APPARTENANCES UTILISEES	48
V.3.1. <i>Mise en œuvre de FreeRTOS dans une application embarquée</i>	49
V.3.2. <i>Les tâches dans freeRtos</i>	49
V.3.3. <i>L'ordonnancement des tâches dans freeRtos</i>	49
V.3.4. <i>La concurrence des tâches</i>	50
V.4. METHODE DE DEVELOPPEMENT D'UNE APPLICATION EMBARQUEE A L'AIDE D'UN RTOS	50
V.4.1. <i>Le nombre de tâches d'une application embarquée</i>	50
V.5. L'APPLICATION EMBARQUEE MULTITACHE REALISEE	51
V.5.1. <i>L'aération</i>	51
V.5.2. <i>Le refroidissement</i>	52
V.5.3. <i>Le chauffage</i>	53
V.5.4. <i>L'irrigation</i>	53

<i>V.5.5. L'éclairage</i>	54
<i>V.5.6. Les taches de l'application embarquée réalisée</i>	58
<i>V.5.7. Programme de la centrale :</i>	60
<i>V.5.8. Les règles :</i>	61
V.6. L'INTERFACE HOMME MACHINE REALISEE	63
V.7. CONCLUSION	66
CONCLUSION GENERALE	67
BIBLIOGRAPHIE :	68

Liste des abbreviations

1. IOT : INTERNET OF THINGS
2. M2M : MACHINE TO MACHINE
3. IP : INTERNET PROTOCOL
4. WIFI : WIRELESS FIDELITY
5. BLE : BLUETOOTH LOW ENERGY
6. ZIGBEE : ZIGBEE (UN STANDARD POUR LES RESEAUX DE CAPTEURS SANS FIL)
7. ZWAVE : Z-WAVE (UN PROTOCOLE DE COMMUNICATION POUR LA DOMOTIQUE)
8. LORA : LONG RANGE
9. SIGFOX : SIGFOX (UN RESEAU POUR L'INTERNET DES OBJETS)
10. RPMA : RANDOM PHASE MULTIPLE ACCESS
11. NB_IOT : NARROWBAND INTERNET OF THINGS
12. 3G : THIRD GENERATION (RESEAUX MOBILES)
13. 4G : FOURTH GENERATION (RESEAUX MOBILES)
14. 5G : FIFTH GENERATION (RESEAUX MOBILES)
15. MQTT : MESSAGE QUEUING TELEMETRY TRANSPORT
16. COAP : CONSTRAINED APPLICATION PROTOCOL
17. HTTP : HYPERTEXT TRANSFER PROTOCOL
18. IHM : INTERFACE HOMME-MACHINE
19. ETC : ELECTRONIC TOLL COLLECTION (OU SIMPLEMENT "ET CETERA" EN ANGLAIS)
20. ISR : INTERRUPT SERVICE ROUTINE
21. CPU : CENTRAL PROCESSING UNIT
22. GPOS : GENERAL PURPOSE OPERATING SYSTEM
23. OSS : OPEN SOURCE SOFTWARE
24. RTOS : REAL-TIME OPERATING SYSTEM

25. API : APPLICATION PROGRAMMING INTERFACE
26. ECOS : EMBEDDED CONFIGURABLE OPERATING SYSTEM
27. FREERTOS : FREE REAL-TIME OPERATING SYSTEM
28. LINUX EMBARQUÉ : EMBEDDED LINUX
29. LITEOS : LIGHTWEIGHT OPERATING SYSTEM
30. LYNXOS : LYNX OPERATING SYSTEM
31. MENUETOS : MENUET OPERATING SYSTEM
32. NUTTX : NUTTX REAL-TIME OPERATING SYSTEM
33. OS-9 : OPERATING SYSTEM 9
34. PIKEOS : PIKE OPERATING SYSTEM
35. QNX : QNX REAL-TIME OPERATING SYSTEM
36. RTEMS : REAL-TIME EXECUTIVE FOR MULTIPROCESSOR SYSTEMS
37. RTLINUX : REAL-TIME LINUX
38. RT-THREAD : REAL-TIME THREAD
39. RTX : REAL-TIME EXECUTIVE
40. μ C/OS-II : MICRO-CONTROLLER OPERATING SYSTEM II
41. VXWORKS : VXWORKS REAL-TIME OPERATING SYSTEM
42. ZEPHYR : ZEPHYR PROJECT REAL-TIME OPERATING SYSTEM
43. SOC : SYSTEM ON CHIP
44. KIB : KIBIBYTE
45. RAM : RANDOM ACCESS MEMORY
46. ROM : READ-ONLY MEMORY
47. PWM : PULSE WIDTH MODULATION
48. ESP32 : ESP32 (UN MICROCONTROLEUR AVEC WI-FI ET BLUETOOTH INTEGRES)
49. DHT22 : DIGITAL HUMIDITY AND TEMPERATURE SENSOR

50. LDR : LIGHT DEPENDENT RESISTOR

51. MCU : MICROCONTROLLER UNIT

52. GND : GROUND

53. VCC : VOLTAGE COMMON COLLECTOR

54. LED : LIGHT EMITTING DIODE

55. FREERTOS : FREE REAL-TIME OPERATING SYSTEM (RÉPÉTÉ, MAIS INCLUS POUR COMPLÉTUDE)

56. FREERTOS_AVR.H : FREERTOS AVR HEADER

57. FREERTOS_ARM.H : FREERTOS ARM HEADER

58. HPS : HIGH-PERFORMANCE SYSTEM

Liste des figures

FIGURE I. 1 : les trois types de connectivites m2m	5
FIGURE I.2 : architecture de base de l'iot.....	7
FIGURE I.3 : structure du reseau iot	8
FIGURE I. 4 : capteurs et actionneurs connectes a une unite de traitement de proximite via un reseau sans fil	9
FIGURE I. 5 : capteurs et actionneurs connectes a une unite de traitement distante.	10
FIGURE I. 6 : capteurs et actionneurs connectes physiquement a l' unite de traitement	11
FIGURE I. 7 : capteurs connectes physiquement a l' unite de traitement et actionneurs a cette derniere via un reseau sans fil	11
FIGURE I.8 : les 3 v du big data	12
FIGURE II.1 : demi-trapeze gauche	18
FIGURE II.2 : demi-trapeze droit	19
FIGURE II.3 : triangle symetrique ou asymetrique	19
FIGURE II.4: trapeze symetrique ou asymetrique	19
FIGURE II.5 : exemple de fuzzification d'une valeur mesuree de temperature	19
FIGURE II. 6: exemple de sous-ensembles flous d'entree, ensemble de discours d'entree, variable linguistique d'entree.	20
FIGURE II.7 : singletons ou rectangles etroits	20
FIGURE II.8 : triangle / trapeze	20
FIGURE II.9 : exemple de regle floue avec une conclusion floue	22
FIGURE II.10 : l'aggregation avec methode de coupure	22
FIGURE II.11 : exemple de regle floue avec une conclusion egale a une fonction mathematique des entrees	25
FIGURE II.12 : l'aggregation avec la methode de sugeno	26

FIGURE II.13 : les composants d'un contrôleur flou	27
FIGURE II.14 : fonctionnement d'un contrôleur flou	27
FIGURE II.16 : système de commande en boucle fermée ou rétroactif	28
FIGURE II.15 : système de commande en boucle ouverte ou directe	28
FIGURE III.1 : représentation schématique de l'architecture multitâche d'une application embarquée	29
FIGURE III.2 : application embarquée développée a l'aide de l'approche « boucle infinie + services d'interruption »	30
FIGURE III.3 : application embarquée développée a l'aide d'un ordonnanceur coopératif	32
FIGURE III.4 : exemple d'implémentation d'une application embarquée développée a l'aide d'un ordonnanceur coopératif, en langage arduino.....	33
FIGURE III.5 : les différentes classes de systèmes d'exploitation	34
FIGURE III.6 : exécution pseudo-parallèle.....	34
FIGURE III.7: application embarquée développée a l'aide d'un rts	35
FIGURE IV.1 : structure globale du système réalise.....	37
FIGURE IV.2 : brochage et interfaçage du dht22 a un microcontrôleur.....	39.
FIGURE IV.3 : brochage du capteur d'humidité du sol utilise.	39
FIGURE IV.4 : interfaçage a un microcontrôleur.....	39
FIGURE IV.5 : interfaçage d'une ldr a un mcu.....	40
FIGURE IV.6: exemple d'ev a commande proportionnelle	41
FIGURE IV.7 : principe de fonctionnement d'un servo moteur a commandé pwm	41.
FIGURE IV.8 : le servo-moteur sg90 et son interfaçage a un mcu	42
FIGURE IV.9: led blanche 12v	42
FIGURE IV.10 : interfaçage d'une led blanche a un mcu	42.
FIGURE IV.11 : ventilateur 5v.....	43
FIGURE IV.12 : interfaçage avec un mcu	43

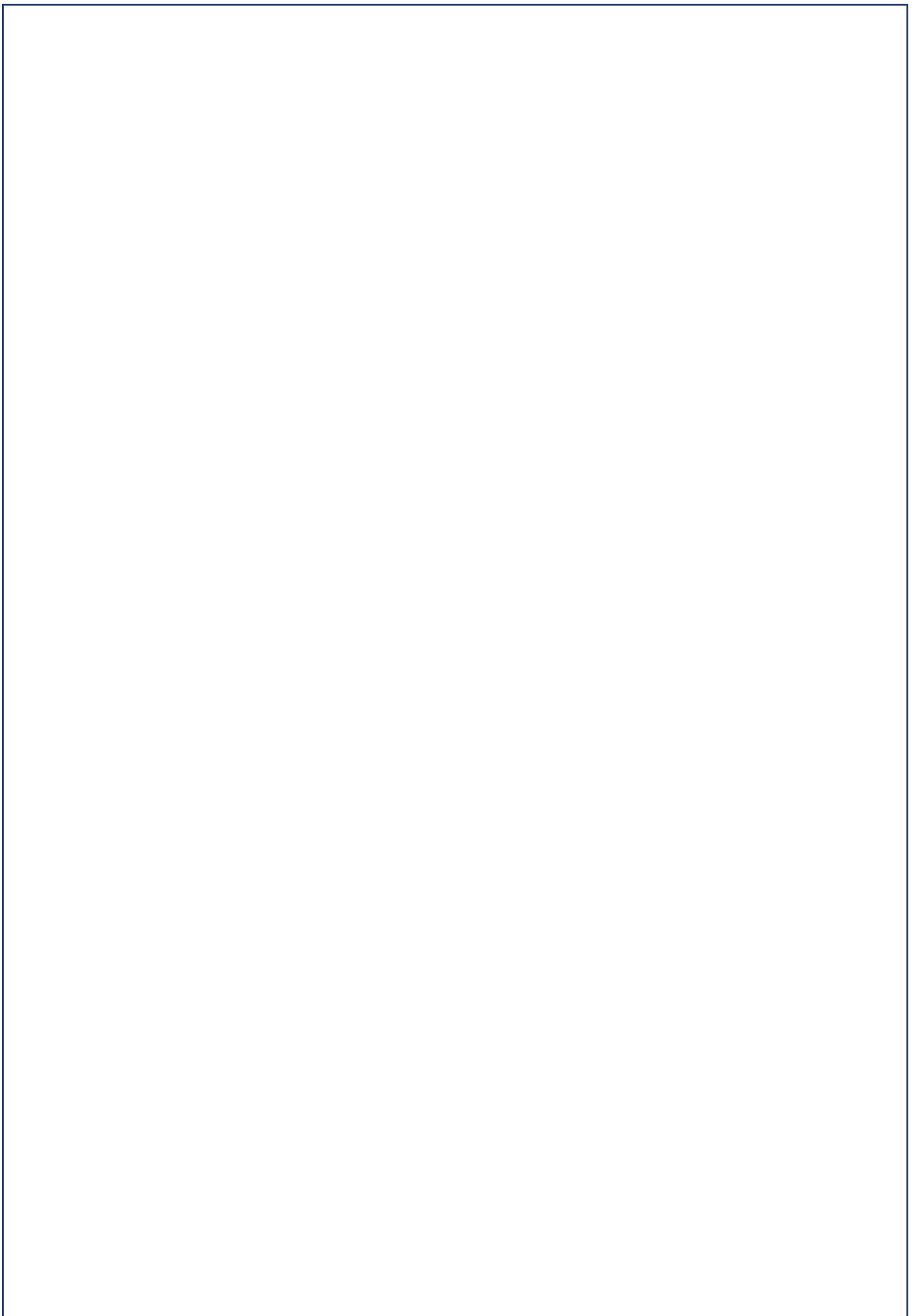
FIGURE IV.13 : schema electrique de l'interfaçage du controleur flou avec les capteurs utilises ..	44
FIGURE IV.14 : schema electrique de l'interfaçage de l'unite de commande avec les actionneurs utilises	45
FIGURE V.1 : fonctions d'appartenance	48
FIGURE V.2 : structure générique d'une tache dans freertos	49
FIGURE V.3 : principe de fonctionnement de l'exclusion mutuelle appliquée a un capteur partage par les taches d'une application embarquée	50
FIGURE V.4 : exemple de système d'aération naturelle constitue de fenêtres rabattables ou basculantes	51
FIGURE V.5 : exemple de système d'aération forcée constitue de ventilateurs/extracteurs	51
FIGURE V.6 : refroidissement par brumisateu	52
FIGURE V.7 : refroidissement par ombrage.	52
FIGURE V.8 : exemple de système de chauffage constitue de générateurs d'air chaud	53
FIGURE V.9 : quelques exemples de systèmes d'irrigation	54
FIGURE V.10 : exemple d'éclairage avec des lampes hps	54
FIGURE V.11 : exemple d'éclairage avec des lampes led	55
FIGURE V.12 : diagramme de contexte du système réalise	56
FIGURE V.13 : les taches de l'application embarquée du système réalise	57
FIGURE V.14 : les taches de l'application embarquée réalisée	58
FIGURE V.15 : contrôle du système de chauffage.....	59
FIGURE V.16 : contrôle du système de ventilateur.....	59
FIGURE V.17 : contrôle du système d'irrigation	60
FIGURE V.18 : contrôle du système d'ombrage	60
FIGURE V.19 : contrôle du système d'éclairage	60
FIGURE V.20: organigramme de la commande des actionneurs	60
FIGURE V.21 : vue de l'ihm de supervision réalisée	63

FIGURE V.22 : vue de l'affichage des paramètres sur la console de l'ide arduino64

FIGURE V.23 : vue de l'ihm de supervision réalise et de l'affichage des paramètres sur la console de l'ide arduino65

LISTES DES TABLEAUX

TABLEAU V.1: REGLES DE CONTROLE FLOU DU SYSTEME DE CHAUFFAGE	61
TABLEAU V.2: REGLES DE CONTROLE FLOU DU SYSTEME DE REFROIDISSEMENT...	61
TABLEAU V.3: REGLES DE CONTROLE FLOU DU SYSTEME D'IRRIGATION	61
TABLEAU V.4: REGLES DE CONTROLE FLOU DU SYSTEME D'OMBRAJE.....	62
TABLEAU V.5: REGLES DE CONTROLE FLOU DU SYSTEME D'ECLAIRAGE.....	62



Introduction Générale

Procurer du confort à des plantes cultivées sous serre pour bien croître, ne peut être obtenu qu'en agissant simultanément sur tous les paramètres qui entrent dans leur croissance. Alors une gestion multitâche s'impose. Parmi ces paramètres, les plus importants sont la température, l'humidité relative de l'air, l'humidité du sol et l'intensité de la lumière. Or ces paramètres en plus d'être interdépendants (i.e : une action sur l'un, impacte l'autre ou les autres systématiquement), sont représentés en agriculture, pour une croissance optimale d'une plante donnée par des plages et par conséquent un contrôle intelligent de ces paramètres par logique floue ou par réseaux de neurones est beaucoup plus adapté.

Par ailleurs, si l'on cherche une gestion efficace et économique de la consommation de l'énergie, de la lumière et de l'eau dans la serre, alors celle-ci ne peut être obtenue qu'à l'aide d'un contrôle lui aussi intelligent basé de préférence sur la logique floue ou les réseaux de neurones.

De même, si l'on veut éviter à l'agriculteur de passer son temps à se déplacer pour réaliser des relevés dans la serre et lui offrir une surveillance et un contrôle distants à travers Internet des différents paramètres climatiques et hydriques de sa serre, l'utilisation de la technologie IoT s'impose.

Egalement, si l'on veut mettre en œuvre un traitement en temps réel de proximité, de gestion et de supervision du confort climatique de la serre, alors l'utilisation de la technologie 'Edgecomputing' s'impose elle aussi.

Et en fin si l'on veut offrir à l'agriculteur des services lui permettant de faire des prédictions et par conséquent prendre des mesures à temps, c'est aux technologies du « Big Data » et du « cloud computing » qu'il faut faire appel.

C'est autour de ce contexte que se situe le sujet de notre projet de fin d'étude : il s'agit de concevoir et de réaliser un système de contrôle multitâche, basé sur la logique floue, pour une gestion intelligente, du confort climatique d'une serre agricole. Dans le contexte de notre projet ce qu'on veut dire par confort climatique de la serre, les besoins optimaux en chaleur, en intensité de lumière, en humidité relative de l'air et humidité du sol pour une croissance optimale des plantes à l'intérieur de la serre.

Le projet étant relativement complexe et nécessitant d'abord l'acquisition d'un minimum de compétences pour pouvoir le réaliser, nous avons adopté pour le mener à terme la démarche suivante : Dans un premier temps, une recherche bibliographique approfondie a été effectuée sur

des projets similaires. Nous avons par la suite, sélectionné le matériel avec lequel le réaliser, l'approche de conception de l'application embarquée, ensuite, nous avons fait les choix quand aux « outils de conception » : « langages et environnement de développement ».

Ceci étant dit, le mémoire de notre projet est organisé en cinq chapitres. Le 1^{er} chapitre est consacré à la présentation des technologies de l'IOT, de l'Edgecomputing, du Big Data et du cloud computing et leurs rapport avec la serriculture, le 2^{ème} entièrement consacré à la présentation détaillée de la logique floue, l'approche que nous avons utilisée pour embarquer de l'intelligence artificielle dans notre projet, et ceci vu son importance, le 3^{ème} au multitâche et aux différentes approches de conceptions d'applications embarquées multitâches, le 4^{ème} à la réalisation matérielle et le 5^{ème} à la réalisation logicielle du projet.

Chapitre I : L'IOT, l'Edgecomputing, le big data, le cloud computing et leurs rapports avec la serriculture

I.1. Introduction

L'essor des technologies de l'information et de la communication a révolutionné de nombreux secteurs, et l'agriculture n'y fait pas exception. Dans le cadre de la serriculture, l'intégration de l'Internet des Objets (IoT), de l'EdgeComputing, du Big Data et du Cloud Computing ouvre la voie à une gestion plus intelligente et efficace des environnements de culture sous serre. Ces technologies permettent de collecter, analyser et utiliser des données en temps réel pour optimiser les conditions climatiques, améliorer les rendements et réduire les coûts opérationnels.

Dans ce chapitre, nous explorerons en détail comment ces technologies se combinent pour transformer la serriculture, offrant des solutions innovantes pour répondre aux défis actuels et futurs de l'agriculture sous serre.

Les serres intelligentes représentent une autre génération développée des serres basée sur l'introduction des technologies avancées telles que Big Data, Internet des objets (iot), etc. Ces serres sont dotées d'un système de contrôle qui permet de collecter et d'analyser en temps réel des données sur les plantes et leur environnement et les transmet via internet à d'autres plateformes. L'Internet des objets représente une infrastructure qui remplit le gap entre les capteurs simples qui fournit des données bruts, bruités et imparfaites et la virtualisation basée sur des applications de haut niveau qui fournissent des services sophistiqués. Dans ce chapitre vont être présentés et discutés les points suivants :

- l'IoT avec tout ce qui s'y rapporte car nous jugeons que c'est très important de le faire, et son rapport avec la serriculture ;
- le big data et son rapport avec la serriculture ;
- l'informatique de proximité et la serriculture ;
- Le nuaging et la serriculture ;

I.2. L'IoT :

Evolution naturelle des technologies, l'IoT est un lien inévitable entre le monde numérique et le monde physique.

Maintenant, et parce qu'un objet compatible IoT est un objet qui incarne les caractéristiques de ce dernier, on ne peut pas alors parler de 'serriculture intelligente' sans parler de l'IoT. Cette partie est donc consacrée à ce réseau.

Chapitre I

Les points suivants sont par conséquent discutés : nous allons commencer par définir ce qu'est l'IoT, donner ses caractéristiques, décrire son fonctionnement, citer quelques exemples de domaines applicatifs, donner sa structure générale et montrer son rapport avec la serriculture intelligente.

I.2.1. Définitions de l'IoT

L'IoT est un réseau qui interconnecte Internet et les objets physiques du monde dans lequel nous vivons. Autrement dit, c'est une extension du réseau internet aux objets du monde physique.

Les objets dans l'IoT sont constitués par des systèmes embarqués associés aux objets physiques du monde dans lequel nous vivons.

Les systèmes embarqués des objets IoT sont dotés de :

- Capteurs et actionneurs, leurs permettant d'interagir avec l'environnement physique ;
 - Connectivités M2M (Machine To Machine) locales ou distantes, leurs permettant de communiquer entre eux sans ou avec une intervention humaine très limitée ;
 - Connectivités IP (Internet Protocole), permettant de les connecter à Internet ;
- et ce, à des fins d'automatisation, de supervision, de contrôle, de suivi et de prédiction.

I.2.2. Caractéristiques d'un réseau IoT

Ces caractéristiques sont : la connectivité des objets, l'identité des objets, le partage des données entre les objets, l'Intelligence embarquée sur les objets, l'évolutivité des objets, l'architecture de l'IoT et enfin la sécurité objets.

I.2.2.1. La Connectivité des objets IoT

- La connectivité est la capacité de communiquer et de partager des informations entre deux ou plusieurs objets.
- L'IoT crée un monde où tout est connecté grâce à la connectivité des objets. Ces objets peuvent être n'importe quoi. La connectivité permet aux objets de l'IoT, en plus d'échanger des données, d'être contrôlés à distance via Internet.
- La connectivité dans l'IoT est double : M2M et IP.
- La M2M est mise en œuvre grâce à des réseaux câblé ou sans fil.
- Un réseau M2M est un réseau dans lequel les objets peuvent communiquer directement.
- La connectivité IP, quand à elle, est mise en œuvre à l'aide de protocoles basés sur le protocole IP de Internet.

Chapitre I

On donne à la figure 1ci-dessous, les différents types de réseaux M2M sans fil les plus utilisés actuellement dans l'IoT et sur lesquels s'appuie une connectivité IP, classés en fonction de leur portée.

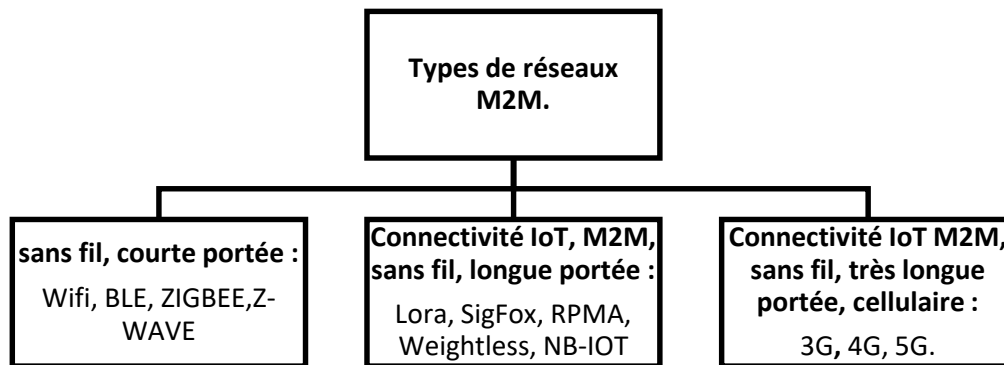


Figure I. 1 : Les trois Types de connectivités M2M .

1.2.2.2. L'identité des objets IoT:

L'identité est la caractéristique unique d'un objet. C'est ce qui permet d'identifier de manière unique un objet. C'est la seule chose qui rend un objet unique et identifiable. Elle peut être utilisée pour distinguer les objets et de les contrôler.

Par exemple, une adresse IP peut être utilisée comme identifiant unique représentant un objet dans un réseau. Elle peut être adresse IP dynamique publique, une adresse IP fixe publique ou une adresse IP fixe privée.

1.2.2.3. Le partage de données entre les objets IoT

Le partage des données entre les objets IoT fait référence à la capacité des objets connectés à échanger des informations entre eux de manière transparente et sécurisée.

Le partage de données permet aux objets IoT de coopérer, de prendre des décisions et d'agir en fonction des données collectées.

Grâce à l'IoT, les données peuvent être partagées par les objets IoT (objet IoT pris au sens le plus large) et ainsi permettre de mieux gérer, surveiller, suivre, contrôler, ... etc et de prendre de meilleures décisions.

Le partage de données peut être réalisé via des protocoles de communication standardisés tels que MQTT, CoAP, http, ... etc et le plus souvent, implique des technologies telles que les passerelles IoT, les services Cloud et les réseaux de capteurs.

1.2.2.4. L'Intelligence embarquée sur les objets IoT:

L'intelligence embarquée sur les objets IoT fait référence à la capacité des objets connectés à traiter, analyser et prendre des décisions localement, sans nécessiter une connexion constante à un réseau externe ou à un serveur distant. Cette intelligence embarquée permet aux objets IoT de fonctionner de manière autonome, ce qui peut améliorer la réactivité, réduire la latence et économiser de la bande passante en ne transmettant que les données pertinentes vers une plate forme distante (cloudcomputing). Ceci peut être particulièrement utile dans les environnements où la connectivité est limitée ou peu fiable.

Cette intelligence embarquée peut être mise œuvre à l'aide d'algorithmes développés en utilisant la logique floue ou les réseaux de neurones.

1.2.2.5.L'Évolutivité des objets de l'IoT :

L'évolutivité des objets IoT se réfère à leur capacité à s'adapter et à évoluer pour répondre aux besoins changeants des utilisateurs et des environnements. Cela inclut la capacité des objets à être mis à niveau avec de nouvelles fonctionnalités, à s'intégrer à de nouveaux systèmes et à être facilement extensibles pour prendre en charge un plus grand nombre d'applications et de cas d'utilisation. Pour garantir une évolutivité efficace, les objets doivent être conçus compatibles avec des normes ouvertes et des interfaces flexibles, permettant une intégration aisée dans l'écosystème IoT.

L'évolutivité des objets IoT est donc la capacité de ces derniers à se développer sans que leur performance soit affectée. Techniquement parlant, elle peut être réalisée en ajoutant des ressources matérielles ou en ajoutant des couches logicielles supplémentaires à un objet IoT ou système existant.

1.2.2.6. L'Architecture de l'IoT :

C'est une structure globale ou représentation schématique qui définit comment les différents composants du réseau IoT interagissent entre eux pour collecter, traiter, stocker et partager des données.

C'est l'enchevêtrement de composants tels que : capteurs, actionneurs, services en « edgecomputing », « en cloud computing », protocoles ouverts ou propriétaires et couches qui composent les systèmes de mise en réseau IoT.

C'est une architecture en couche.

Chapitre I

On donne à la figure 2 ci-dessous, l'architecture de base à titre illustratif car il n'existe pas en pratique jusqu'à l'heure actuelle un consensus sur le nombre de couches que devrait avoir cette architecture.

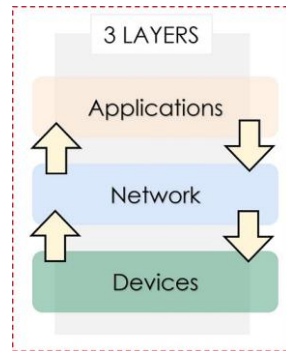


Figure I.2 : Architecture de base de l'IoT

1.2.2.7. La sécurité des objets IoT

On le rappelle : l'idée de l'IoT, est que tout ce qui nous entoure sera connecté à Internet pour nous faciliter la vie. Cependant, et en contre partie, ceci implique des risques : les cyber-attaques. Aussi et afin d'être à l'abri de ces derniers, il faut prendre des mesures de sécurité lors du développement d'un objet IoT.

1.2.3. Fonctionnement d'un réseau IoT

Le fonctionnement de l'IOT, commence au niveau des objets capteurs (couche « devices ») qui collectent des données et les envoient vers une plateforme locale : « edgecomputing » (couche « Applications ») ou distante : « cloud computing » (couche « Applications »). Cette dernière après traitements et analyses, envoie des commandes à des objets actionneurs (couche « Devices »).

Les capteurs, les actionneurs, les plateformes locales et distantes, peuvent être contrôlés par l'utilisateur à travers une IHM.

1.2.4. Quelques domaines applicatifs de l'IoT

- Ville intelligente : circulation routière intelligente, transports intelligents, collecte des déchets, cartographies diverses (bruit, énergie, etc.).
- Environnements intelligents : prédiction des séismes, détection d'incendies, qualité de l'air, etc.

Chapitre I

- Contrôle industriel : mesure, pronostic et prédiction des pannes, dépannage à distance.
- Santé : suivi des paramètres biologiques à distance.
- **Agriculture intelligente**, domotique, applications ludiques.
- Etc... pour ne citer que ceux là et la liste est encore plus longue.

I.2.5. Structure du réseau IoT

Avoir une vue de la structure de ce réseau, permet de mieux voir de quoi il s'agit quand on parle de l'IoT. Aussi, on donne à la figure 3 ci-dessous, la composition de ce type de réseau.

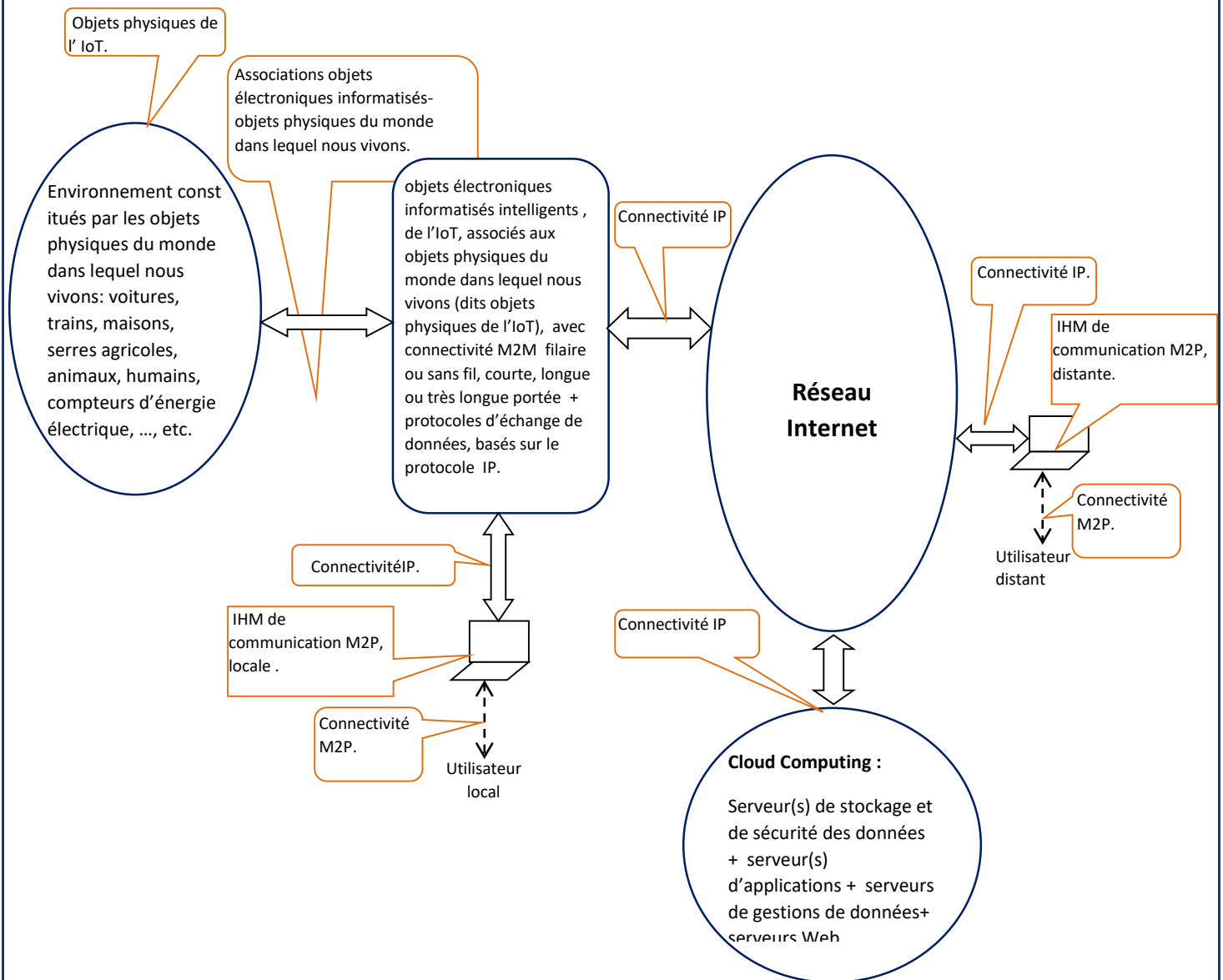


Figure I.3 : Structure du réseau IoT

1.2.6. Les technologies de l’IoT et la serriculture

En serriculture les technologies de l’IoT sont utilisées pour doter une serre avec un ou plusieurs réseaux IoT de courte ou longue portée, connectés à Internet. Le but étant l’automatisation de la serre, la surveillance de ses paramètres climatiques et leur contrôle, le suivi des semis et la prédiction entre autres de la croissance des plantes.

1.2.7. Domaines d’utilisation des composants IoT d’une serre IoT

En serricultureIoT, les composants sont utilisés pour :

- La surveillance de la température, de l’humidité relative de l’air, de l’humidité du sol, de l’intensité de la lumière, des ravageurs, des maladies, de la croissance des plantes, ... etc.
- Le contrôle du chauffage, de l’éclairage, de l’ombrage, du refroidissement, de l’irrigation, ... etc.
- Le suivi des semis, .
- La prédiction de la croissance des plantes, des maladies... etc.

On donne aux figures 4, 5, 6 et 7, suivantes, les différentes configurations possibles de réseaux IoT que l’on peut utiliser dans les pratiques de contrôle, de surveillance, de suivi et de prédiction dans une serre intelligente.

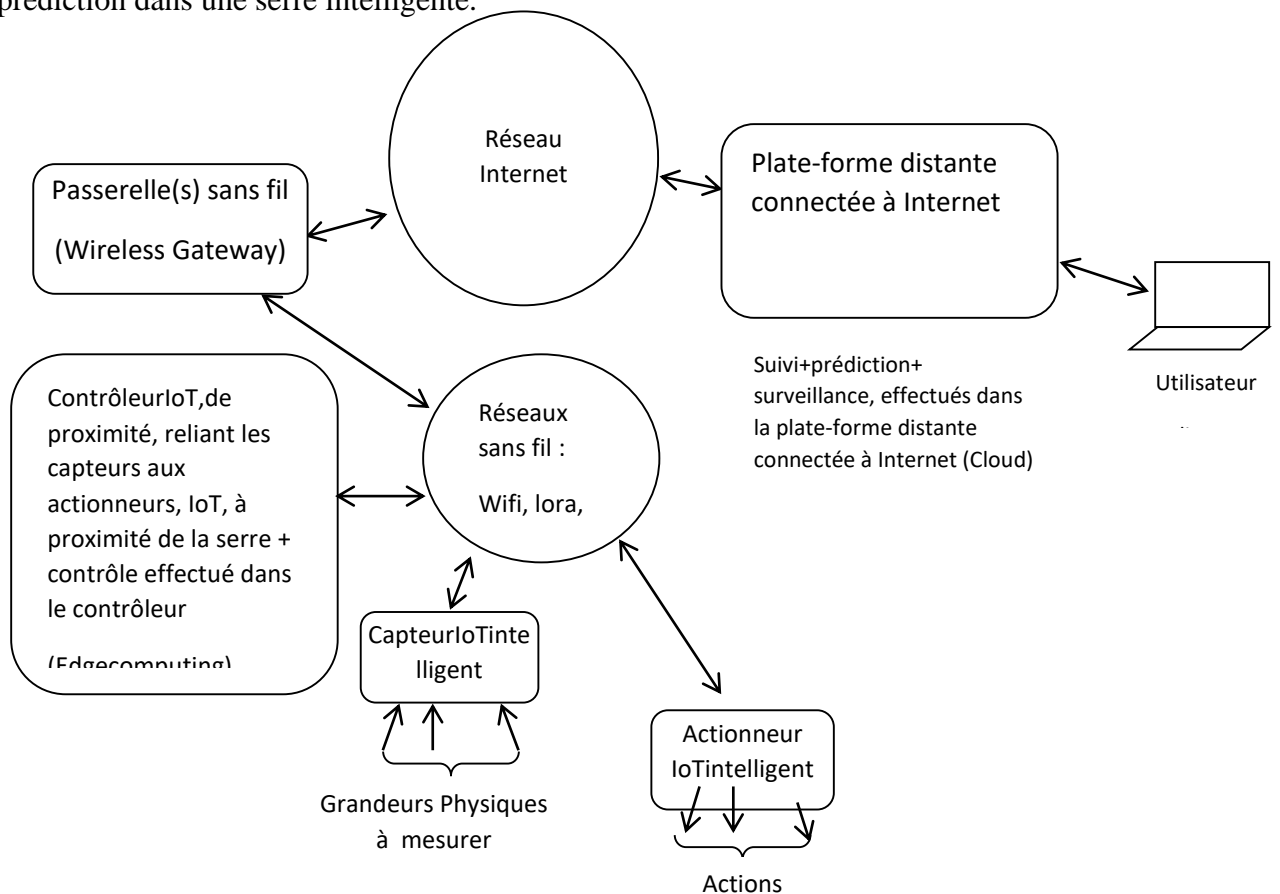


Figure I. 4 : capteurs et actionneurs connectés à une unité de traitement de proximité via un réseau sans fil

Chapitre I

Dans cette configuration les capteurs se contentent d'envoyer les valeurs de mesures au contrôleur de proximité et c'est dans ce dernier que le traitement et le contrôle se produit. L'IHM peut être un client léger (développée en HTML+javascript+CSS et embarquée dans le contrôleur et exécutée dans un navigateur de l'ordinateur de supervision distant) ou lourde (développée avec un langage de programmation de haut niveau et exécutée dans l'ordinateur de supervision de l'utilisateur distant).

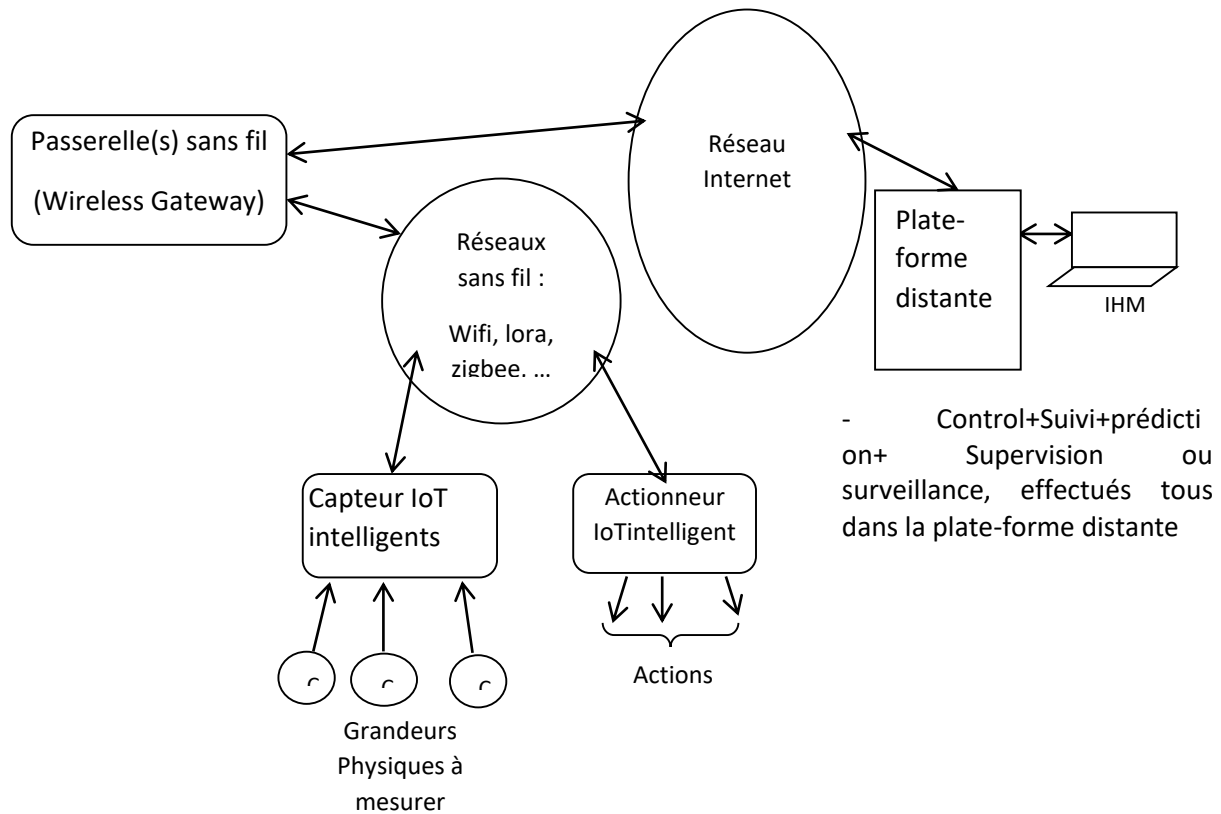


Figure I. 5 : capteurs et actionneurs connectés à une unité de traitement distante.

Chapitre I

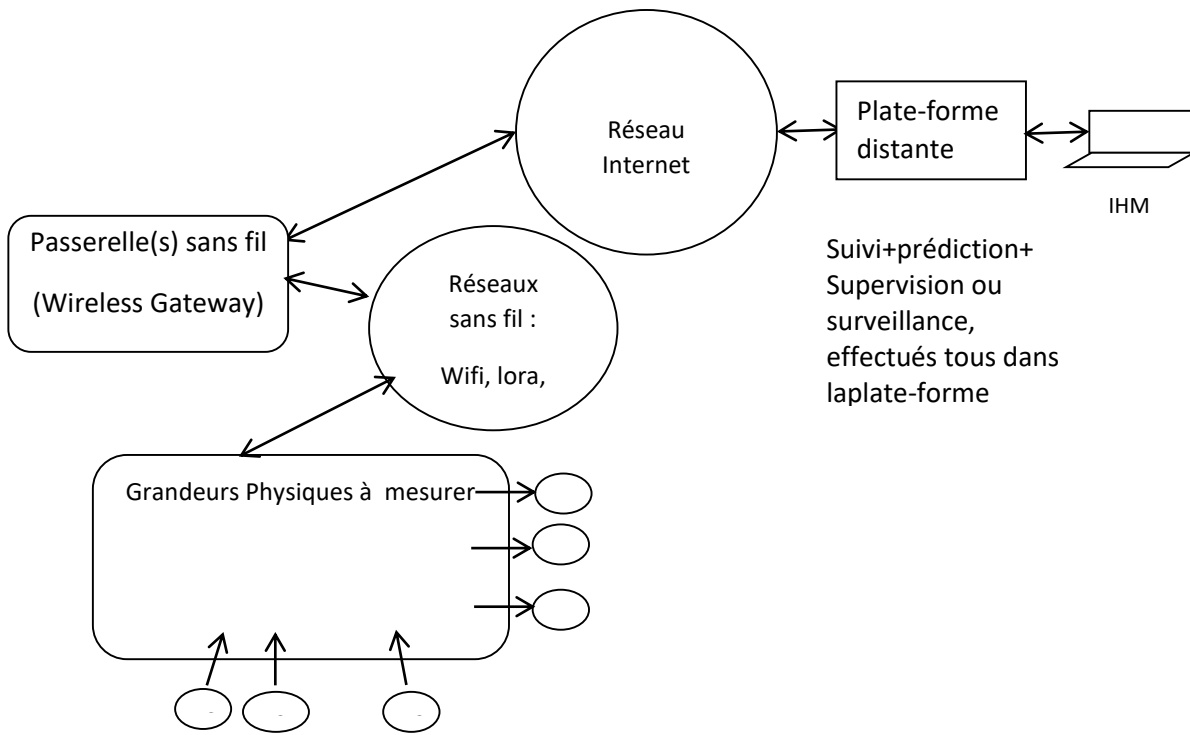


Figure I. 6 : capteurs et actionneurs connectés physiquement à l'unité de

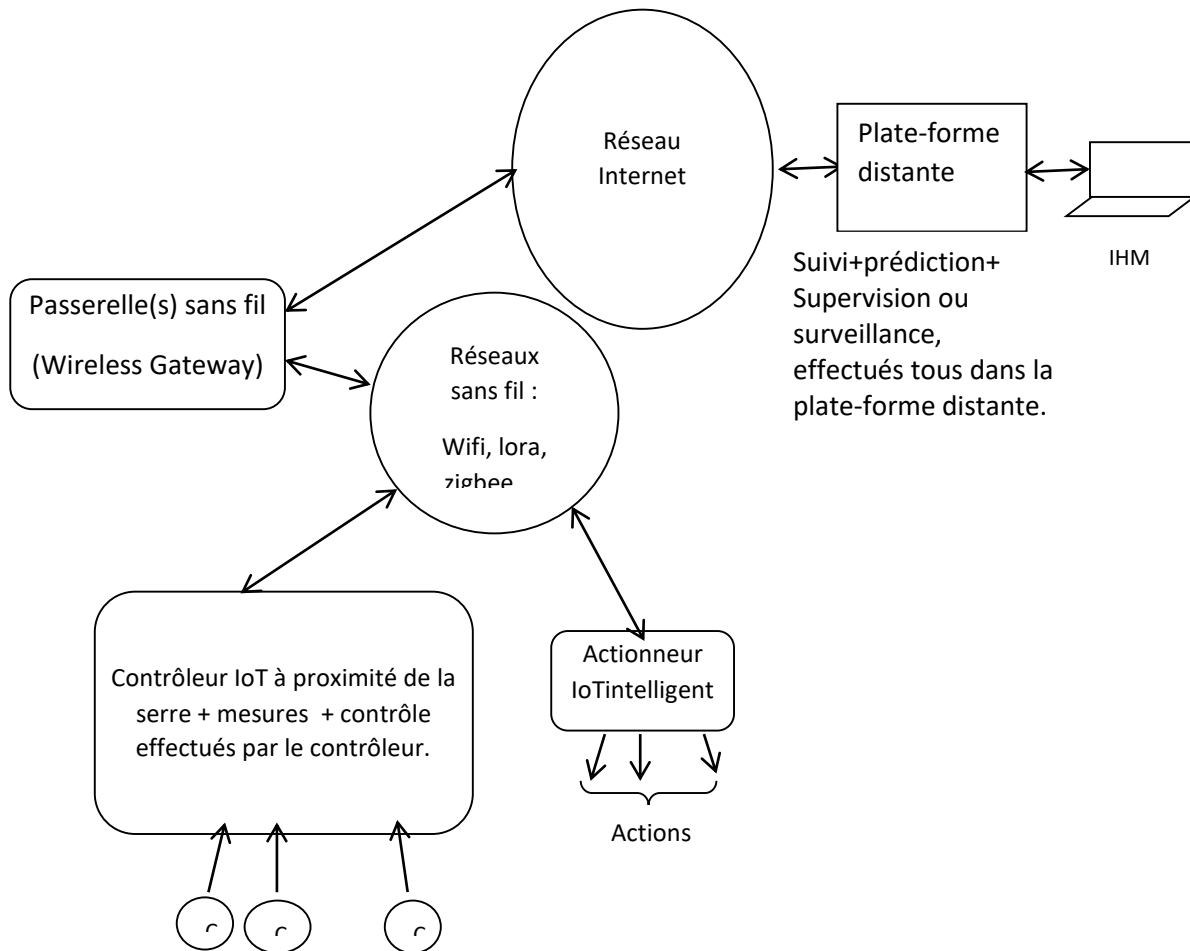


Figure I. 7 : capteurs connectés physiquement à l'unité de traitement et actionneurs à cette dernière via un réseau sans fil.

I.3. Le Big Data

I.3.1. Définitions

Le Big Data, également appelé « mégas données », « grosses données » ou encore « données massives », désigne un ensemble de données très volumineux difficile à travailler avec les outils classiques de gestion de base de données et de gestion de l'information.

Il est caractérisé par ses 3V que sont : le volume, la vitesse et la variété (Figure 1.8).

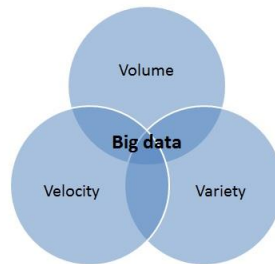


Figure 1.8 : Les 3 V du Big Data

Concernant le volume : la quantité de données astronomiques générées à travers le monde est en constante augmentation. Donc seul le Big Data est capable de traiter un nombre aussi conséquent de données et d'informations.

Concernant la vitesse : c'est la rapidité à laquelle les données affluent à travers le monde. Autrement dit, la fréquence à laquelle elles sont générées, capturées et partagées.

Concernant la variété : les types de données et leurs sources sont de plus en plus diversifiés supprimant ainsi les structures nettes et faciles à consommer des données classiques. Ces nouveaux types de données incluent un grand nombre de contenus très diversifiés tels que : géo-localisation, connexions, mesures, processus, flux, réseaux sociaux, texte, web, images, vidéos, mails, livres, tweets, enregistrements audio...pour ne citer que ceux là !

1.3.2. Rapports avec la serriculture

Son rapport avec la serriculture intelligente, est qu'une grande quantité de données, peut être collectée à l'intérieur d'une serre ou serres, pouvant facilement constituer un Big Data qui peut être utilisé dans les pratiques de contrôle, de surveillance, de suivi, de prédiction et dans la découverte de nouvelles connaissances.

I.4. L'edgecomputing

I.4.1. Définition

L'edge computing désigne le traitement informatique qui s'effectue à l'emplacement physique de la source des données, ou à proximité.

En rapprochant le calcul de la source des données, l'edge computing permet de profiter de services plus rapides donc sans latence et fiables.

I.4.2. Rapport avec la serriculture intelligente

En serriculture, l' 'Edge computing' couvre le domaine de la surveillance et du contrôle, des paramètres climatiques les plus importants de la serre à savoir : la température, l'humidité relative de l'air, l'humidité du sol, l'intensité de la lumière. Il est temps réel et permet donc le traitement des données des capteurs à la source.

I.5. Le Cloud Computing

I.5.1. Définition

Le « *cloud computing* », l'**informatique en nuage** ou l'**infonuagique**, est la pratique consistant à utiliser des serveurs informatiques à distance, hébergés dans des centres de données connectés à Internet pour stocker, gérer et traiter des données, plutôt qu'un serveur local ou un ordinateur personnel⁴.

I.5.2. Rapport avec l'agriculture sous serre

En serriculture, « le cloud computing » couvre les domaines de la surveillance des ravageurs, des maladies, de la croissance des plantes, ...etc, du suivi et de la prédiction, domaines qui nécessitent pour bien rendre service une grande quantité de données : du « Big Data »

I.6. Conclusion

Dans ce chapitre nous avons présenté les nouvelles technologies de manière plus ou moins détaillée, et leurs rapports avec la serriculture.

Chapitre II : Généralités sur La logique floue

II.1. Introduction

Parmi les récents développements des techniques de commande, l'introduction de la logique floue, a suscité un intérêt sans cesse croissant depuis les quelques dernières décennies. Il suffit de voir les nombreuses applications industrielles qui en découlent et de consulter l'abondante littérature sur ce sujet pour s'en convaincre.

L'intérêt de la logique floue réside dans sa capacité à traiter, l'imprécis, l'incertitude et le vague. Elle est issue de la capacité de l'homme à décider et agir de façon pertinente malgré le flou des connaissances disponibles et a été introduite dans le but d'approcher le raisonnement humain à l'aide d'une représentation adéquate des connaissances. Aussi, le succès de la

ntérêt de la logique floue réside dans sa capacité à traiter, l'imprécis, l'incertitude et le vague. Elle est issue de la capacité de l'homme à décider et agir de façon pertinente malgré le flou des connaissances disponibles et a été introduite dans le but d'approcher le raisonnement humain à l'aide d'une représentation adéquate des connaissances.

intérêt de la logique floue réside dans sa capacité à traiter, l'imprécis, l'incertitude et le vague. Elle est issue de la capacité de l'homme à décider et agir de façon pertinente malgré le flou des connaissances disponibles et a été introduite dans le but d'approcher le raisonnement humain à l'aide d'une représentation adéquate des connaissances.

L'intérêt de la logique floue réside dans sa capacité à traiter, l'imprécis, l'incertitude et le vague. Elle est inspirée de la capacité de l'homme à décider et agir de façon pertinente malgré le flou des connaissances disponibles et a été introduite dans le but d'approcher le raisonnement humain à l'aide d'une représentation adéquate des connaissances. Aussi, le succès de la commande floue trouve en grande partie son origine dans sa capacité à traduire une stratégie de contrôle d'un opérateur qualifié, expert, en un ensemble de règles linguistiques « si ... alors » facilement interprétables.

L'utilisation de la commande floue est particulièrement intéressante lorsqu'on ne dispose pas de modèle mathématique précis du processus à commander ou lorsque ce dernier présente de trop fortes non linéarités ou imprécisions.

Aussi, l'objectif de cette partie est de présenter les notions principales de la théorie de la logique floue, ainsi que de la commande floue. D'abord, nous en donnons quelques définitions, nous présentons ensuite les concepts de base de cette logique, ses avantages et ses inconvénients, indiquons

Chapitre II

quand il faut et quand il ne faut pas utiliser cette logique et enfin donnons la structure d'un contrôleur flou, celle d'un système de contrôle flou en boucle ouverte et celle d'un système de contrôle flou en boucle fermé.

II.2. Définitions

La logique floue (ou fuzzylogic en anglais) est une approche de l'informatique basée sur des « degrés de vérité » plutôt que sur la logique booléenne habituelle « vrai ou faux » (1 ou 0), et sur laquelle repose l'informatique moderne.

Le mot « **flou** » désigne des choses qui ne sont pas claires ou vagues. En effet, parfois, il nous arrive de ne pouvoir décider si un problème ou une affirmation donnée est vraie ou fausse. Ce concept fournit donc de nombreuses valeurs entre le vrai et le faux et nous donne la flexibilité nécessaire pour trouver la meilleure solution à un problème.

Un exemple pour mieux appréhender ce concept : répondre à la question « est ce que c'est de l'eau chaude ? » en le comparant au raisonnement booléen. En logique booléenne ou binaire : seulement 2 réponses sont possibles : oui ou non et ces réponses sont quantifiées par 1 ou 0 respectivement. Donc, en logique binaire, une affirmation est vraie ou fausse et son degré de vérité vaut 1 ou 0.

En logique floue par contre: plusieurs réponses sont possibles, 3 par exemple : beaucoup, un peu, très peu et ces réponses peuvent être quantifiés par des valeurs comprises entre 0 et 1. Par exemple : 0.9, 0.25, 0.1 respectivement. Donc, en logique floue, une affirmation est plus ou moins vraie (donc, plus ou moins fausse) et son degré de vérité varie entre 0 et 1.

La logique floue est une extension de la logique booléenne classique qui permet de prendre en compte l'incertitude et l'imprécision : alors que la logique classique ne considère que deux valeurs (« vrai » et « faux »), la logique floue considère un ensemble infini de valeurs réelles entre 0 et 1. Autrement dit, contrairement à la logique classique, qui énonce que toute proposition est soit vraie, soit fausse, la logique floue permet d'exprimer une proposition avec un certain degré de certitude ou d'incertitude.

La logique floue est fondamentalement une logique à valeurs multiples (i.e : non binaire) qui permet de définir des valeurs intermédiaires entre des évaluations binaires comme oui/non, vrai/faux, noir/blanc, etc. par exemple des notions comme plutôt chaud ou assez froid peuvent être formulées mathématiquement avec cette logique et traitées par calculateurs. De cette manière, avec cette logique, on tente d'appliquer une façon de penser humaine dans la programmation des calculateurs.

Chapitre II

La logique floue est un système de résolution de problèmes qui imite la logique de contrôle, humaine, en créant des décisions « raisonnées » et ce, en utilisant une logique multivaluée, un degré d'appartenance dit aussi degré de vérité et un système basé sur des règles.

La logique floue est une méthode de **traitement de l'information** », permettant de traiter des informations imprécises ou incertaines. Elle est capable de représenter des nuances et des degrés de concordance, en utilisant des valeurs intermédiaires comprises entre 0 (pas de similitude) et 1 (lorsque la similitude est totale).

II.3. Historique de la logique floue

Lorsqu'Aristote et ses prédécesseurs ont élaboré leurs théories de la logique et des mathématiques, ils ont élaboré ce qu'on appelle la loi du milieu ou de la tierce exclu qui stipule que toute proposition doit être vraie ou fausse. Par exemple: l'herbe est soit verte, soit non verte; il est clair qu'il ne peut pas être à la fois vert et non vert. Mais tout le monde à l'époque, n'était pas d'accord, et Platon par exemple et pour ne citer que lui, a indiqué qu'il existait une troisième région intermédiaire à ces opposés.

Dans la vision aristotélicienne du monde, la logique concernait deux valeurs seulement. Au XIXe siècle, George Boole a créé un système d'algèbre et de théorie des ensembles capable de traiter mathématiquement une telle logique à deux valeurs, en mappant respectivement le vrai et le faux sur 1 et 0. Puis, au début du XXe siècle, Jan Lukasiewicz a proposé une logique à trois valeurs (vrai, possible, faux), qui n'a jamais été largement acceptée.

Ceci dit, la logique floue est une approche issue d'une théorie mathématique qui a été développée dans les années 60 par le mathématicien Lotfi Zadeh [1]. En 1965, Lotfi A. Zadeh de l'Université de Californie à Berkeley a publié « Fuzzy Sets », qui présentait les mathématiques de la théorie des ensembles flous et, par extension, de la logique floue. Zadeh avait observé que la logique informatique conventionnelle dite aussi binaire, ne pouvait pas manipuler des données représentant des idées subjectives ou vagues. Il a donc créé une logique floue pour permettre aux calculateurs de déterminer les distinctions entre les données avec des nuances de gris, similaire au processus de raisonnement humain.

II.4. Avantages de la logique floue

- Modèle mathématique du système contrôlé non nécessaire ;
- Système pouvant être très non linéaire ;
- Temps de développement du contrôleur flou réduit ;
- Connaissances en automatique réduites.

II.5. Inconvénients de la logique floue

Analyse qualitative très poussée du système à contrôler, nécessaire; pour faire un recueil d'expertise.

II.6. Les domaines d'applications

On peut utiliser des contrôleurs flous pour contrôler des systèmes flous. En effet, la plupart des algorithmes de contrôle traditionnels nécessitent un modèle mathématique du processus ou système à contrôler. Cependant, de nombreux systèmes physiques sont difficiles, voire impossibles à modéliser mathématiquement. De plus, de nombreux processus ou systèmes à contrôler sont soit non linéaires, soit trop complexes pour que l'on puisse les contrôler avec les stratégies traditionnelles. Par conséquent, si l'on peut décrire qualitativement une stratégie de contrôle, on peut alors utiliser la logique floue pour créer un contrôleur flou qui émule une stratégie heuristique basée sur des règles d'expert.

II.7. Les domaines où il ne faut pas utiliser la logique floue

- La théorie conventionnelle du contrôle donne un résultat satisfaisant.
- Un modèle mathématique facilement résoluble et adéquat existe déjà.
- Le problème n'est pas résoluble.

II.8. Les concepts de la logique floue

Dans les paragraphes qui suivent, nous allons présenter ces différents concepts, et pour faciliter leurs compréhensions, nous allons le faire à travers les exemples donnés dans les paragraphes précédents.

II.8.1. Le concept de valeur linguistique

Dans le vocabulaire de la logique floue, les réponses (**beaucoup, peu, très peu**) de l'exemple (**est ce que c'est de l'eau chaude ?**) ci-dessus, sont appelées « **valeurs linguistiques** » dites aussi, « **valeurs symboliques** » ou « **labels flous** ».

II.8.2. Le concept de degré de vérité

De même, en logique floue, les valeurs de quantification (**0.9, 0.25, 0.1**) des réponses de ce même exemple, sont appelées « **degrés de vérité** ».

II.8.3. Le concept de variable linguistique

De même, la proposition « **eau Chaude** », utilisée dans la question de cet exemple, et que l'on peut écrire sous la forme « **eauChaude** » dans cette logique, est appelé « **variable linguistique** ».

II.8.4. Le concept de sous-ensembles flous:

En logique floue, une variable linguistique d'entrée/sortie peut être associée à plusieurs labels flous ou valeurs linguistiques (**Fuzzification**).

Ces valeurs linguistiques ou labels flous, d'entrée/sortie définissent les valeurs floues de la variable linguistique d'entrée/sortie et sont appelés en logique floue, **sous-ensembles flous (FUZZY SET)**.

II.8.5. Le concept d'ensembles de discours :

En logique floue, les labels ou les sous-ensembles flous, correspondent chacun à une partie d'un ensemble appelé dans le vocabulaire de cette logique : **ensemble de discours**.

II.8.6. Le concept de fonction d'appartenance :

En logique floue, un sous-ensemble flou A d'un ensemble de discours X est totalement caractérisé par une fonction appelée « **fonction d'appartenance** ». Elle est utilisée aussi bien en entrée comme en sortie.

Les fonctions d'appartenance peuvent théoriquement prendre n'importe quelle forme. Toutefois, elles sont souvent définies, pour des raisons de facilité d'utilisation, par des segments de droites, et donc sont « linéaires par morceaux ». Elles sont notées par le symbole μ . Pour exprimer le degré de vérité d'une valeur numérique dite en anglais « crisp value ») d'une variable linguistique, d'appartenir à un label, la fonction d'appartenance est utilisée comme suit :

$$\mu_{\text{Label}}(\text{variable linguistique} = \text{valeur numérique})$$

II.8.7. Fonction d'appartenance d'entrée et fuzzification

En pratique, on utilise les fonctions d'appartenance d'entrée montrées aux figures II.1, II.2, II.3 et II.4 suivantes:

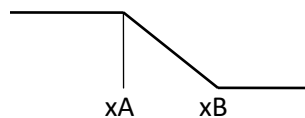
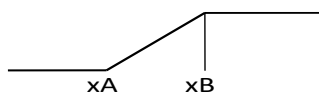


Figure II.1 : **Demi-trapèze gauche**



Chapitre II

Figure II.2 : Demi-trapèze droit

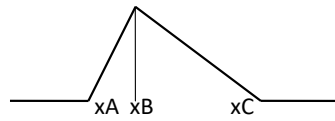


Figure II.3 : Triangle symétrique ou asymétrique

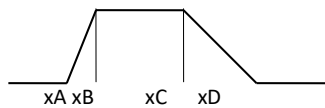


Figure II.4: Trapèze symétrique ou asymétrique

En entrée, cette fonction sert à quantifier le degré d'appartenance de la valeur numérique (mesurée, calculée, ..., etc), d'une variable linguistique d'entrée, à un label parmi les labels associés à cette variable.

En logique floue, l'association de labels ou de sous-ensembles flous d'entrée à une variable linguistique d'entrée est appelée «*fuzzification* ».

Pour illustrer cette dernière (fuzzification), on donne à la **figure II.5** ci-dessous, comment elle fonctionne, sur l'exemple d'une valeur de température mesurée (crisp value) et sa fuzzification en labels : FAIBLE, MOYENNE et ELEVEE.

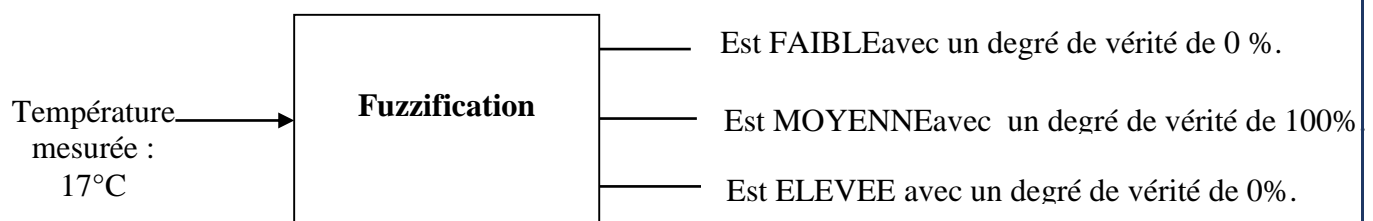


Figure II.5 : Exemple de fuzzification d'une valeur mesurée de température

Concrètement, cette opération (fuzzification en entrée), se fait en prenant la valeur numérique « crisp value » d'une entrée et lui faisant correspondre des valeurs floues en effectuant sa projection orthogonale depuis l'axe des températures sur ses labels et en évaluant ses degrés de vérité ou d'appartenance à ces labels à l'aide de la fonction d'appartenance associée.

Exemple :

Chapitre II

Une valeur d'une variable d'entrée linguistique ici de température, mesurée, peut appartenir à plusieurs labels, avec des degrés de vérité divers. : Degré d'appartenance de l'entrée T aux sous ensembles flous ou au label Froid ou Chaud (Figure II.6).

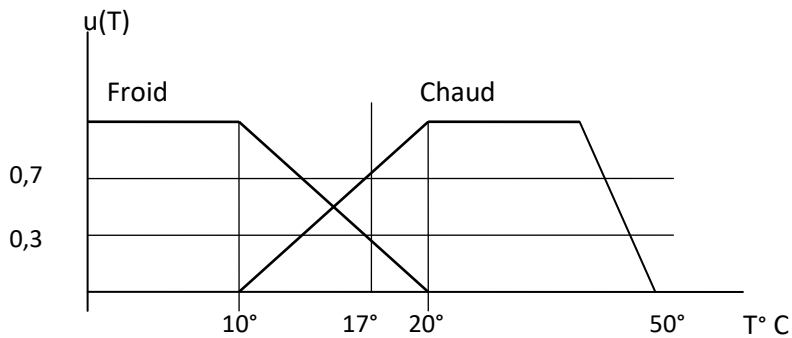


Figure II. 6:Exemple de sous-ensembles flous d'entrée, ensemble de discours d'entrée, variable linguistique d'entrée.

Sur le graphique de l'exemple ci-dessus, les sous ensembles flous « Froid » et « Chaud » correspondent respectivement aux sous ensembles de valeurs de température $[0,20]$ et $[10,50]$. L'ensemble flou est égal à $\{\text{Froid}, \text{Chaud}\}$

D'après ce graphique, on constate, que pour une valeur de température $T = 17^\circ$, le degré d'appartenance au sous ensemble flou « froid » vaut $\mu_{\text{Froid}}(T=17^\circ) = 0,3$ et le degré d'appartenance au sous ensemble « chaud » vaut $\mu_{\text{Chaud}}(T=17^\circ) = 0,7$.

II.8.8. Fonction d'appartenance de sortie, règles, inférence floue et dé fuzification

En pratique, on utilise les fonctions d'appartenance de sortie montrées aux figures II.7 et II.8 suivantes:

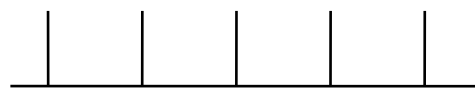


Figure II.7 : Singletons ou Rectangles Etroits



Figure II.8 : Triangle / Trapèze

Chapitre II

En sortie, cette fonction sert à calculer la valeur numérique réelle (Crisp Value) de la sortie qui va être utilisée pour commander ou contrôler un système. Et cette opération est appelée en logique floue **déffuzzification**.

Maintenant, cette opération (**déffuzzification**) se fait en calculant les différents degrés de vérité des différentes affirmations (dite en logique floue : conclusions ou déductions) sur une variable de sortie puis les projeter orthogonalement sur les différents labels associés à cette variable depuis l'axe des fonctions d'appartenance de la sortie, ensuite les exploiter pour calculer la valeur numérique (crisp value) de la sortie.

Le calcul des différents degrés de vérité des différentes affirmations est appelé en logique floue : **l'inférence floue**.

Elle est dite **inférence floue** car elle est basée sur un raisonnement flou (i.e : règles floues). Par exemple :

Plus la température est basse plus il faut chauffer fort, que l'on peut exprimer sous forme de **règle floue** d'expert comme suit :

Si température est très basse **ALORS** chauffer fort.

Et dans ce cas, la valeur de la conclusion de la règle floue (**chauffer fort**), est l'appartenance d'une variable floue de sortie **chauffer** à un sous ensemble flou ou label **fort**.

En logique floue, les variables linguistiques d'entrées, de sorties, et les labels sont utilisés pour définir les connaissances d'experts humains, sous forme d'expressions floues dans un ensemble de règles. Ces règles sont basées sur une combinaison de propositions floues élémentaires a. Une proposition floue élémentaire est de la forme : «*Variable linguistique est Label* » par exemple « **Température est Froide** ».

En logique floue, une combinaison de propositions floues élémentaires est réalisée à l'aide d'opérateurs logiques : négation (NON), conjonction (ET) et disjonction (OU).

Par exemple : « *V est A et W est B* », « *V est A ou non W est B* » sont des expressions floues.

Les propositions floues élémentaires sont des cas particuliers d'expressions floues.

Ces expressions floues sont utilisées pour exprimer les connaissances des experts humains sous forme de **règles floues**.

En logique floue, une règle floue est composée d'une **prémisse** (dite aussi **antécédent**) et d'une **conclusion** (dite aussi **conséquence**) et est de la forme :

« SI prémisse ALORS conclusion ».

Chapitre II

Une prémisses est donc une expression floue alors qu'une conclusion est une déclaration ou affirmation floue de nature différente.

Une conclusion peut être une proposition floue. Et dans ce cas là, ces types de règles, sont dits de « **Mamdani** ».

Exemple 1 :

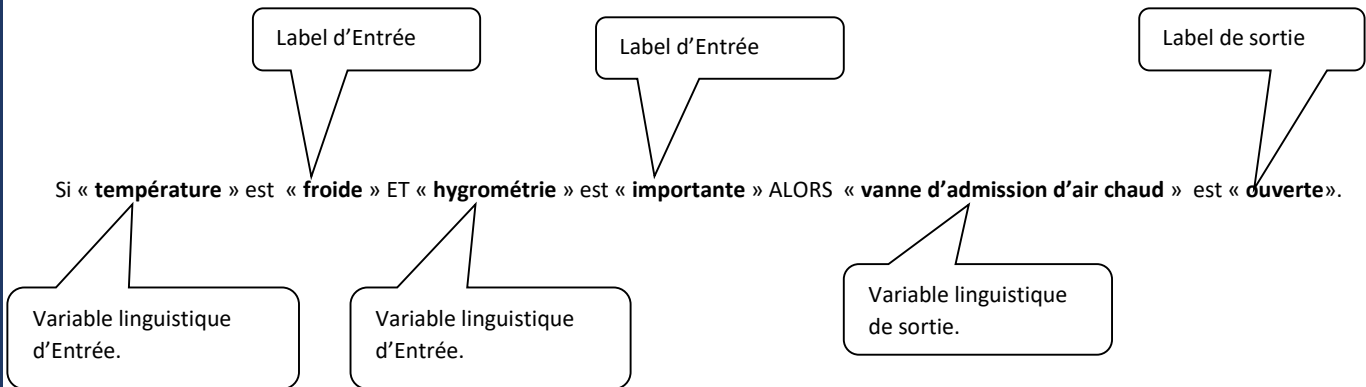


Figure II.9 : Exemple de règle floue avec une conclusion floue

Exemple 2:

SI Température est Froide Alors Vitesse est Minimale.

De même, une conclusion peut être une fonction mathématique des entrées des prémisses. Dans ce cas là, ces types de règles sont dits de « **Takagi-Sugeno** ».

Exemple 3 :

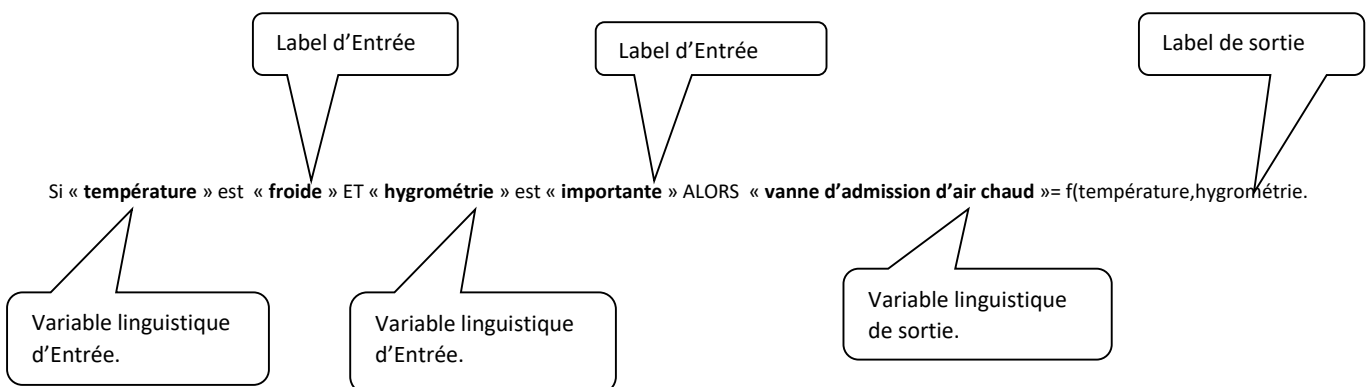


Figure II.10 : Exemple de règle floue avec une conclusion égale à une fonction mathématique des entrées

Exemple 4 :

SI Température est Froide Alors Vitesse = $J+k1*Température$.

Ainsi les règles associent les entrées du système aux sorties à l'aide d'une implication floue (dénotée par ALORS). En logique floue, ces types de règles sont dits « règles aux déductions ».

Par conséquent, le raisonnement en logique floue implique de faire un recueil d'expertise : C'est-à-dire définir un ensemble de règles aux déductions, reliant entre elles les entrées du système par des opérateurs OU ou ET aux sorties à l'aide de l'implication floue ALORS.

Dans les paragraphes qui suivent nous allons donner plus de détails quand aux opérations d'inférence floue et de ffuzification.

L'inférence floue est le calcul des différents degrés de vérité des propositions élémentaires et affirmations des règles reliant entre elles les entrées aux sorties du système.

Par exemple :

❖ **Cas de Mamdani :**

- Si Température Est Faible (degré de vérité ?) Ou Humidité est faible (degré de vérité ?) Alors Vitesse du ventilateur Est Faible (avec quel degré de vérité ?).
- Si température est Moyenne (degré de vérité) Et Humidité est Moyenne (avec quel degré de vérité ?) Alors Vitesse du ventilateur Est Moyenne (avec quel degré de vérité ?).
- Si Température Est Elevée (quelque soit l'humidité) Alors Vitesse du ventilateur Est Elevée (avec quel degré de vérité ?).

❖ **Cas de sugeno :**

- Si Température Est Faible (avec quel degré de vérité ?) Ou Humidité Est Sec (avec quel degré de vérité ?) Alors Vitesse du ventilateur = 30 (avec quel degré de vérité?).
- Si température est Moyenne (avec quel degré de vérité) Et Humidité est Moyenne (avec quel degré de vérité) Alors Vitesse du ventilateur = 50 (avec quel degré de vérité).
- Si Température Est Elevée (quelque soit l'humidité) Alors Vitesse du ventilateur = 70 (avec quel degré de vérité ?)

Donc l'inférence floue consiste à évaluer ces règles en fonction des valeurs floues ou des degrés de vérité de leurs prémisses.

Pour l'évaluation des degrés de vérité des prémisses d'une règle voir § **fuzzification** ci-dessus.

Chapitre II

Maintenant pour l'évaluation du degré de vérité des expressions floues et selon la théorie de cette logique, on applique les opérateurs flous Min, Max, de Mamdani[1] sur les valeurs de vérité de leurs prémisses comme suit :

$$\mu_{A \cap B}(x) = \text{Min}(\mu_A(x), \mu_B(x))$$

$$\mu_{A \cup B}(x) = \text{Max}(\mu_A(x), \mu_B(x))$$

Où A et B sont deux sous ensembles flous, x une variable linguistique et μ la fonction d'appartenance correspondante.

Soit : en considérant l'exemple des règles ci-dessus avec les valeurs de leurs prémisses suivantes :

- $\mu_F(T=18^\circ\text{C})=0,5$.
- $\mu_M(T=18^\circ\text{C})=0,33$.
- $\mu_E(T=18^\circ\text{C})=0,0$.
- $\mu_F(H=80\%)=0,25$.
- $\mu_M(H=80\%)=0,75$.

➤ Pour Mamdani:

Si T Est F(0,5) Ou H Est F(0,25) Alors V Est F(0,5)

Si T Est M(0,33) Et H est M(0,75) Alors V Est M(0,33).

Si T Est E (quelque soit l'humidité) Alors V Est E(0,0).

➤ Et pour Sugeno :

Si T Est F Ou H Est F Alors V = 30(0,5).

Si T Est M Et H est M Alors V = 50(0,33).

Si T Est E (quelque soit l'humidité) Alors V = 70(0,0).

La déffuzification :

les méthodes de déduction (Min, Max), fournissent des degrés d'appartenance résultants μ_R de la variable de sortie xR. Il s'agit donc d'informations floues.

Il faut donc transformer cette information floue en une valeur réelle déterminée (Crisp value) qui sera appliquée à l'interface de commande du processus. Dans le vocabulaire de la logique floue, cette transformation est appelée « **déffuzification** ».

Chapitre II

La Défuzzification, se fait à partir de données floues obtenues par l'opération inférence floue et passe par deux étapes : l'**agrégation** des données floues obtenues lors de l'inférence floue ensuite le **calcul de la sortie crisp** à l'aide de la méthode du centre de gravité ou celle de Sugeno.

Dans les paragraphes qui suivent, nous allons détailler chacune de ces étapes.

L'agrégation :

Deux méthodes peuvent être utilisées dans cette étape : La méthode de coupure et la méthode Sugeno.

- **La méthode de coupure :**

Pour expliquer le fonctionnement de cette méthode, considérons l'exemple d'un contrôleur flou (Figure II.11) dont le fonctionnement est régi par les trois règles d'expert suivantes:

Si x Est A Alors n Est D.

Si y Est B Alors n Est E.

Si z Est C Alors n Est F.

Où :

x, y, z sont les variables linguistiques d'entrée et n la variable linguistique de sortie.

A, B et C les labels ou fonctions d'appartenance associés respectivement aux entrées x, y et z.

D, E et F les labels ou fonctions d'appartenances associés à la sortie n.

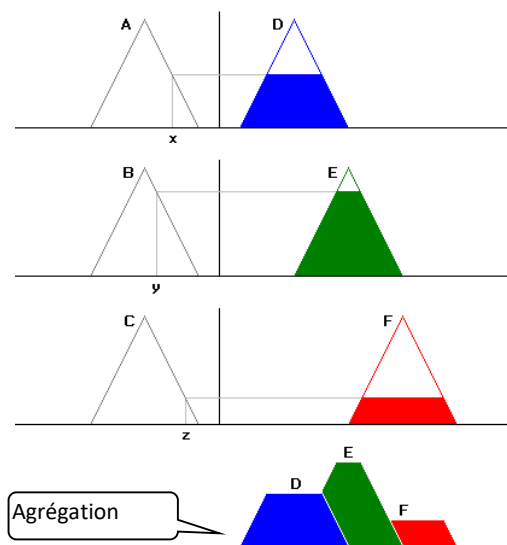


Figure II.11 : l'Aggrégation avec méthode de coupure

Chapitre II

Dans cet exemple, on peut voir que l'agrégation consiste donc à combiner les surfaces en dessous des différentes droites de coupure des données floues des inférences des règles avec les fonctions d'appartenance de la sortie n.

- **La méthode de Sugeno :**

De même, pour expliquer le fonctionnement de cette méthode, considérons ce même exemple de contrôleur flou (Figure II.12).

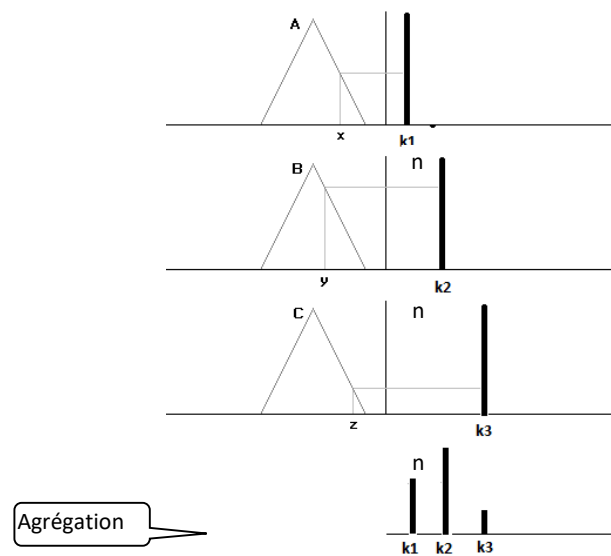


Figure II.12 : l'Aggrégation avec la méthode de Sugeno

Dans ce cas, on peut voir que l'agrégation consiste à combiner les singletons pondérés par les valeurs de coupure correspondant aux données floues des inférences des règles avec les fonctions d'appartenance (les singletons unitaires) de la sortie n.

Calcul de la sortie numérique (crisp value)

- **Méthode du centre de gravité**

Avec cette méthode, la valeur numérique de la sortie du contrôleur est calculée avec la formule suivante :

$$C.G = \frac{\sum_{x=a}^b \mu A(x) \cdot x}{\sum_{x=a}^b \mu A(x)}$$

• **Méthode de Sugeno :**

De même, avec cette méthode, la valeur numérique de la sortie du contrôleur est calculée à l'aide de la formule suivante :

$$C. GSugeno = \frac{\sum_{i=0}^n \mu A(ki) \cdot ki}{\sum_{i=0}^n \mu A(ki)}$$

II.9. Structure et fonctionnement d'un contrôleur flou

Pour résumer on donne à la figure II.13, la structure d'un contrôleur flou et à la figure II.14 suivante, les détails quand à son fonctionnement :

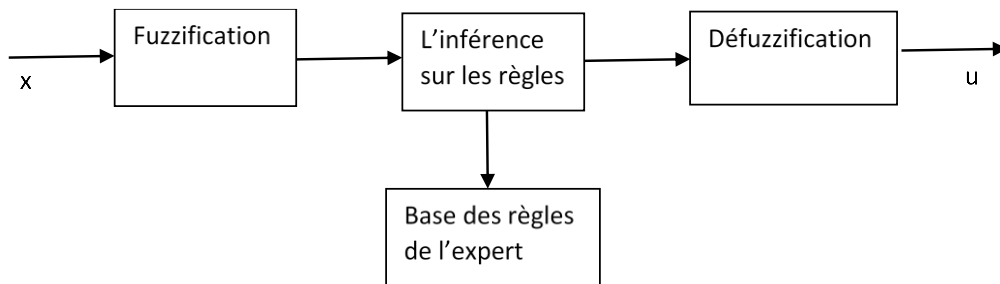


Figure II.13 : Les composants d'un contrôleur flou

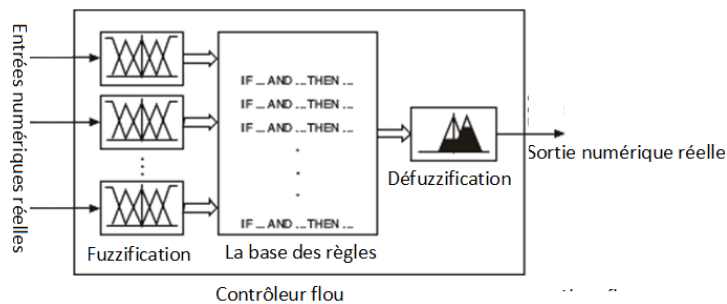


Figure II.14 : Fonctionnement d'un contrôleur flou

II.10. Structure d'un système de contrôle flou en boucle ouverte et en boucle fermée

On le rappelle : un système de contrôle est un ensemble de composants matériels et logiciels conçus pour modifier, changer ou réguler par une action de contrôle un autre système physique afin qu'il présente certaines actions ou comportements souhaités. On le rappelle : les systèmes de contrôle sont de deux types : les systèmes de contrôle en boucle ouverte (Figure II.15), dit aussi système de contrôle directe, dans lesquels l'action de contrôle est indépendante de la sortie physique du système,

Chapitre II

et les systèmes en boucle fermée (Figure II.16) également connu sous le nom de systèmes de contrôle par rétroaction dans lesquels l'action de contrôle dépend de la sortie physique du système

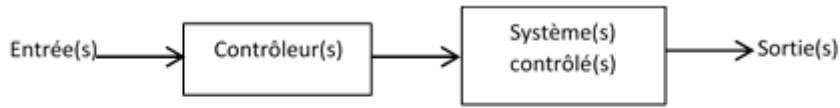


Figure II.15 : Système de commande en boucle ouverte ou directe

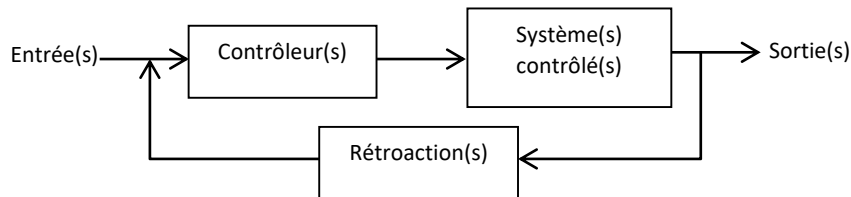


Figure II.16 : Système de commande en boucle fermée ou rétroactif

En pratique, on utilise une commande en boucle ouverte quand le système commandé ne change pas de sortie suite à une perturbation et on utilise une commande en boucle fermée quand le système contrôlé change de sortie quand il est perturbé.

II.12. Conclusion

Nous avons présentés dans ce chapitre de manière très détaillée et pratique les concepts de la logique floue. Nous les avons présentés de façon à ce qu'ils soient facilement exploitables et c'était le but de ce chapitre.

Chapitre III : Le multitâche et les différentes approches de conceptions d'applications embarquées

III.1. Introduction

La conception et la réalisation d'un système multitâche pour la gestion intelligente du confort climatique d'une serre agricole représentent un défi technologique majeur, un système multitâche permet de gérer simultanément des divers paramètres tels que la température, l'humidité et la luminosité

Dans ce chapitre nous discuterons également des architectures d'une application embarquée et les différentes approches de conception d'une application embarquée multitâche avec mention d'exemples, les avantages et les inconvénients de chaque approche.

III.2. Définitions

Le multitâche est une méthode permettant à un système informatique d'exécuter plusieurs tâches ou processus de manière simultanée ou quasi-simultanée.

III.3. Architecture d'une application embarquée

Le comportement concurrent des évènements et grandeurs physiques externes nous contraint à décrire l'environnement comme un système fortement parallèle. Ce qui conduit naturellement à ne voir comme architecture la mieux adaptée pour répondre à ce comportement parallèle de l'environnement externe qu'une architecture multitâche (Figure III.1).

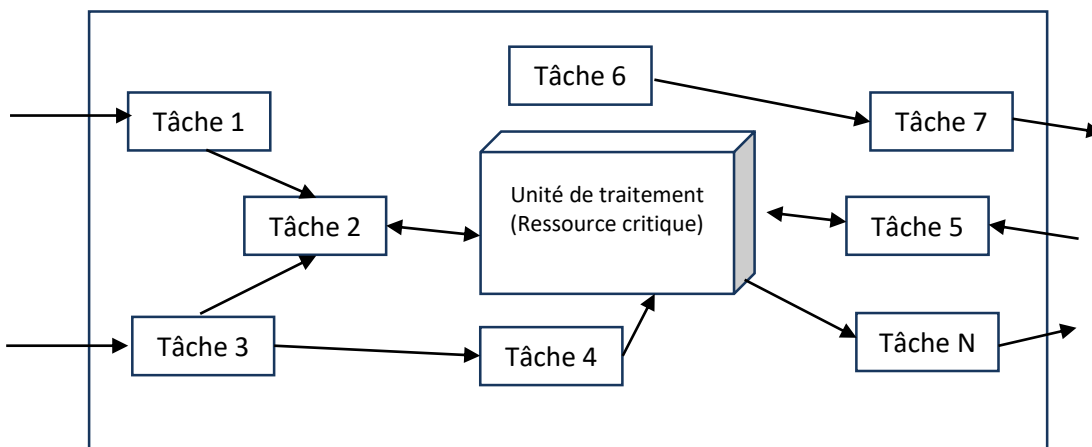


Figure III.1 : Représentation schématique de l'architecture multitâche d'une application embarquée

Chapitre III

III.4. Les différentes approches de conception d'une application embarquée multitâche

En pratique, il existe plusieurs approches :

- La boucle principale plus routines de service d'interruption.
- Les ordonnanceurs coopératifs.
- Les RTOS avec leur ordonnanceur hybride (tourniquet, coopératif + priorisé et préemptifs).
- Les machines virtuelles, les machines d'état et la conception pilotée par des délais réactifs.

Dans les paragraphes qui suivent, nous allons nous limiter aux trois premières, les autres étant relativement complexes et sortant du cadre de ce travail.

III.4.1. Approche de la boucle principale plus routines de service d'interruption

Cette approche est standard pour les petites implémentations logicielles et constitue une bonne solution lorsqu'il n'y a pas besoin d'exécution parallèle bien séparée. Le parallélisme est obtenu grâce aux routines de service d'interruption (ISR), qui sont déclenchées par des interruptions logicielles ou matérielles. La boucle principale est une boucle infinie qui répète indéfiniment les mêmes activités. Son flux est suspendu lorsqu'une interruption se produit pendant que l'interruption est traitée par l'ISR. Le pseudo-code de la figure III.2 ci-dessous, illustre cette solution.

```
variableGlobale1
variableGlobale2
...
variableGlobaleN
intmain()
{
  InitSystem() ;
  For( ;;)
  {
    Tache1();
    Tache2();
    ...
    TacheN() ;
  }
}
Isr1()
{
  //.....
}
Isr2()
{
```

Chapitre III

```
//.....  
}  
...  
IsrN()  
{  
//.....  
}  
}
```

Figure III.2 : application embarquée développée à l'aide de l'approche « boucle infinie + services d'interruption »

❖ **Avantage** : mise en œuvre facile.

❖ **Inconvénients** :

Il y a plusieurs problèmes avec cette approche :

Premièrement, l'itinérance conduit inévitablement à l'utilisation de variables à portée globale.

Deuxièmement, il est difficile de séparer l'application en modules indépendants qui ont leurs propres contraintes de temps. Les dépendances d'une telle implémentation sont trop nombreuses et les changements sont donc difficiles à maintenir.

Troisièmement, les ISR doivent s'exécuter rapidement et revenir une fois terminés. Cela crée des retards pour certaines activités. Le traitement réel doit se produire lorsque le contrôle revient à ce point dans la boucle principale. Selon l'instant où l'interruption s'est produite, un cycle de boucle complet peut s'écouler avant que l'action demandée par l'ISR ne soit exécutée par le logiciel de boucle principale. En d'autres termes, cette solution est la plus facile à créer, mais c'est le choix le moins flexible. Il convient aux petites applications bien définies où des modifications et une maintenance, minimales sont prévues tout au long de la durée de vie de la conception.

III.4.2. Approche basée sur les ordonnanceurs coopératifs

Les ordonnanceurs coopératifs sont implémentés le plus souvent et pour des raisons de facilité d'utilisation sous forme de bibliothèques.

Avec cette méthode, les tâches d'une application coopèrent pour s'exécuter.

Cette coopération consiste en des appels à l'ordonnanceur pour lui rendre le contrôle afin de passer la main à une autre tâche.

Chapitre III

Autrement dit, avec cette méthode, une tâche en cours d'exécution continue de s'exécuter tant qu'elle ne rend pas la main de façon explicite à l'ordonnanceur.

Le passage de la main à la tâche suivante est réalisé par un appel d'une fonction spéciale de l'ordonnanceur.

Avec cette méthode, le rôle de l'ordonnanceur se limite à démarrer les tâches et à leur permettre de lui rendre volontairement le contrôle pour passer la main à une autre tâche.

On donne à la figure III.3 suivante l'organigramme d'une application embarquée réalisée à l'aide d'un ordonnanceur coopératif

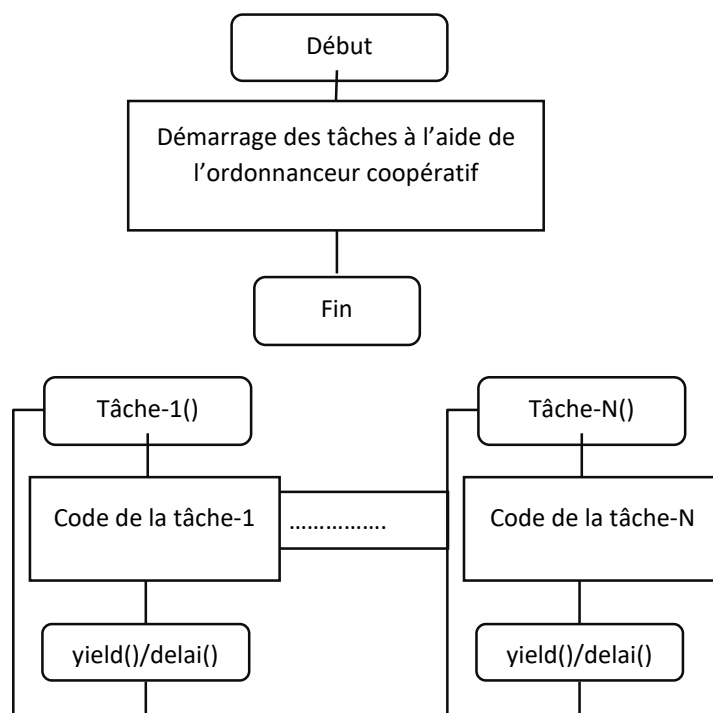


Figure III.3 : Application embarquée développée à l'aide d'un ordonnanceur coopératif

Chapitre III

Exemples d'ordonnanceurs coopératifs :

L'ordonnanceur coopératif « scheduler » de arduino.

On donne en donne à la figure III.4, un exemple d'implémentation en langage arduino :

```
// Inclusion de l'ordonnanceur multitache.
#include<Scheduler.h>
//Déclarations des variables globales et constantes

void setup() {
//Configuration du matériel

// Ajout de deux tâches "loop1" et "loop2" à l'ordonnanceur.
// "loop" est toujours lancée par défaut.

Scheduler.start(setup1,loop1);
Scheduler.start(setup2,loop2);
}

// tâche no.1
//Déclarations des variables globales et constantes
Void setup1(){
//Configuration du matériel
}

voidloop() {
//traitement de la tâche 1
Yield(); // pour passer la main à la tâche suivante
}

// tâche no.2
//Déclarations des variables globales et constantes
Void setup2(){
//Configuration du matériel
}

void loop2() {
//traitement de la tâche 2
yield();// pour passer la main à la tâche suivante
}
```

Figure III.4 : Exemple d'implémentation d'une application embarquée développée à l'aide d'un ordonnanceur coopératif, en langage Arduino

Avantages

Le multitâche coopératif permet une mise en œuvre beaucoup plus simple des applications embarquées car leur exécution n'est jamais interrompue de manière inattendue par l'ordonnanceur.

Chapitre III

Inconvénients

Comme un système multitâche coopératif repose sur le fait que chaque tâche cède régulièrement du temps à d'autres tâches du système, une application embarquée mal conçue peut consommer tout le temps CPU pour elle-même, soit en effectuant des calculs étendus, soit en attendant occupée ; les deux entraîneraient le blocage de l'ensemble du système.

III.4.3. Approche basée sur l'utilisation d'un système d'exploitation type RTOS

III.4.3.1. Les différents types de systèmes d'exploitation

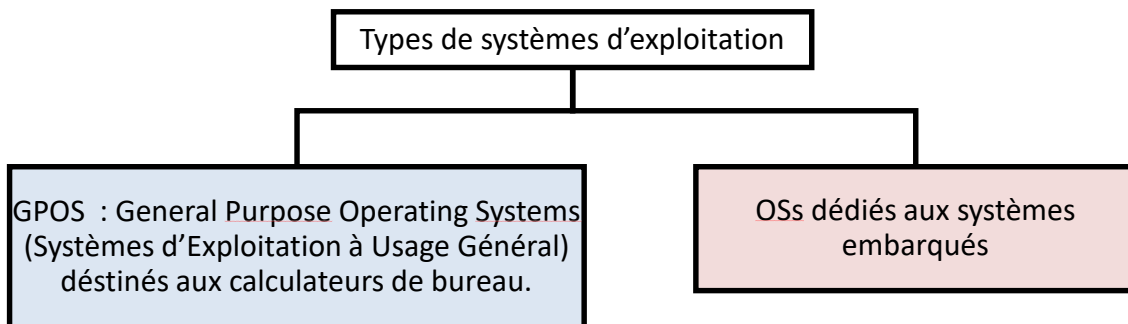


Figure III.5 : Les différentes classes de systèmes d'exploitation

III.4.3.2. Principe de fonctionnement d'un RTOS:

Dans les applications embarquées, un microcontrôleur ne peut exécuter qu'une seule tâche à un instant donné. C'est le planificateur ou l'ordonnanceur du RTOS, qui est chargée de décider quand exécuter une tâche et laquelle. Il donne l'illusion d'une exécution simultanée en basculant rapidement entre les tâches (Figure III.6).

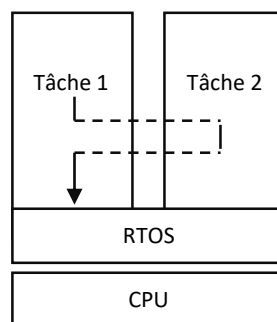


Figure III.6 : Exécution pseudo-parallèle

Chapitre III

Les RTOS sont des systèmes d'exploitation destinés à servir des applications en temps réel via un microcontrôleur qui traite les données au fur et à mesure qu'elles arrivent, généralement sans délai. Les exigences de temps de traitement sont signalées en dixièmes de seconde ou par incréments plus courts.

Les RTOS, facilitent énormément le développement multitâche. Ce dernier, se limite uniquement à la création des tâches et à leurs exécutions respectivement à l'aide de l'API de l'ordonnanceur du RTOS.

On donne à la figure III.7 suivante l'organigramme d'une application embarquée réalisée à l'aide d'un RTOS.

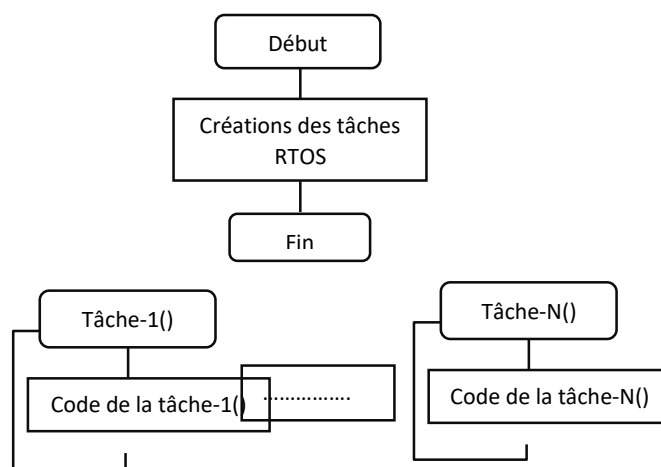


Figure III.7: Application embarquée développée à l'aide d'un RTOS

Avantages

La réalisation d'une application embarquée à l'aide d'un RTOS, présente plusieurs avantages dont on peut citer entre autres :

- Concentration sur l'application, et non sur l'exécution des tâches.
- Gestion automatique du temps de CPU entre les tâches, suivant un critère de sélection prédéfini (e.g : priorité)
- Synchronisation de l'accès aux ressources
- Communication entre les tâches
- Ajouts/retraits de tâches sans modifier l'application existante

Inconvénients

Chapitre III

Nécessite d'abord l'acquisition de compétences qui peut prendre quelques semaines voire quelques mois avant de pouvoir le déployer dans la réalisation d'une application embarquée.

Chapitre III

Exemples de systèmes d'exploitation temps réel dédiés à l'embarqué :

eCos, FreeRTOS, Linux embarqué, LiteOS, LynxOS, MenuetOS, NuttX, OS-9, PikeOS, QNX, RTEMS, RTLinux, RT-Thread, RTX, μ C/OS-II, VxWorks, Zephyr.

III.5. Conclusion

Ce chapitre a exploré les différentes approches de conception des applications embarquées multitâches, en mettant en lumière les avantages et les inconvénients de chaque méthode.

Chapitre IV : Réalisation matérielle

IV.1. Introduction

Dans ce chapitre les deux points suivants vont être présentés et discutés : Tout d'abord le schéma synoptique du système de contrôle que nous avons réalisé va être présenté avec description fonctionnelle. Ensuite vont être présentées les différentes solutions matérielles utilisées pour réaliser les différentes parties du système et enfin les différents interfaçages des capteurs et actionneurs utilisés avec le contrôleur flou.

IV.2. Architecture matérielle du système de contrôle réalisé.

On donne à la figure IV.1 ci-dessous, le schéma bloc du système de contrôle-réalisé.

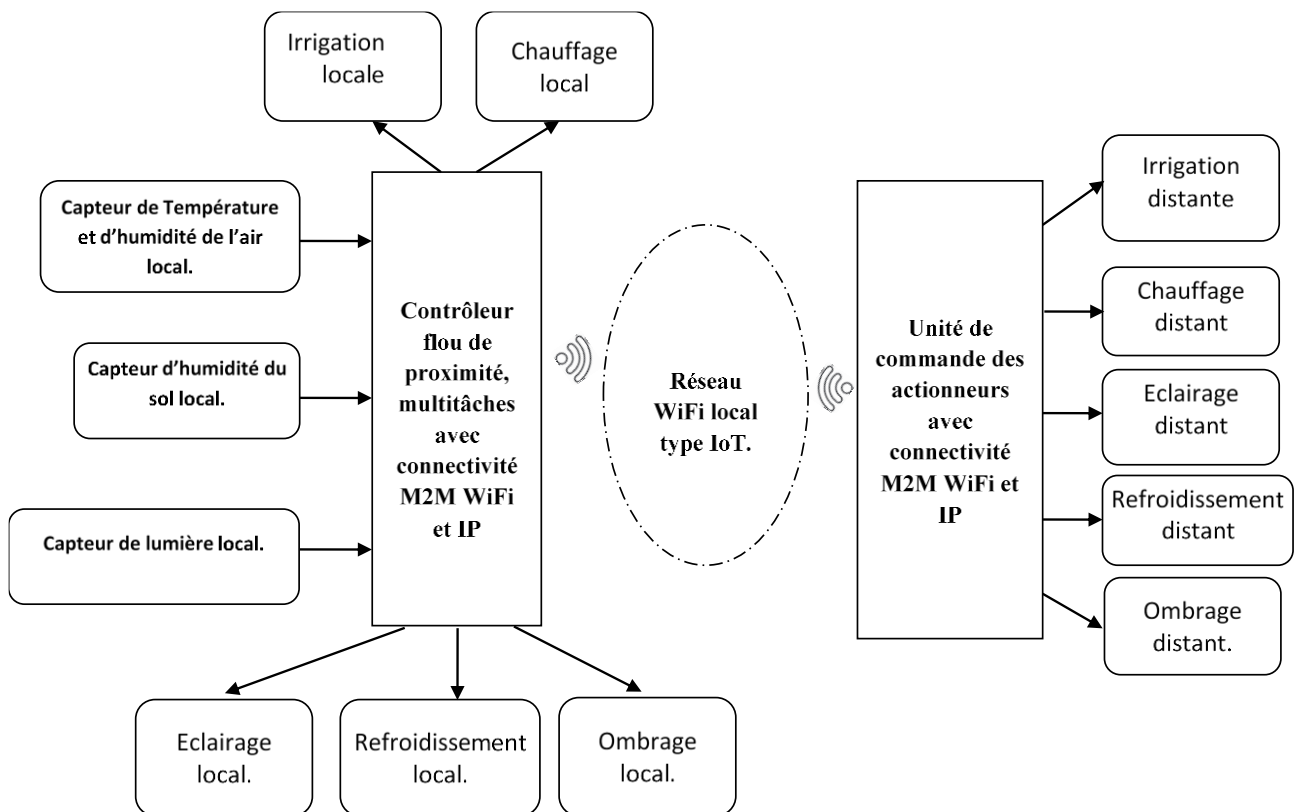


Figure IV.1 : Structure globale du système réalisé

Chapitre IV

IV.3. Description fonctionnelle

Comme le montre la figure IV.1 ci-dessus, le système que nous avons réalisé est constitué principalement d'un contrôleur flou de proximité et d'une unité de commande d'actionneurs.

Le contrôleur flou, surveille en permanence et calcule en temps réel, les paramètres et les valeurs des actions dans la mini-serre, en exécutant des algorithmes basés sur la logique floue (pour les raisons que nous évoquées en introduction).

L'unité de commande des actionneurs quant à elle, fait changer le climat de la mini serrée en fonction des valeurs des commandes qui lui sont envoyées par le contrôleur flou, à travers le réseau WiFi, en agissant sur les actionneurs.

IV.4. Les composants matériels utilisés pour la réalisation du système de contrôle

IV.4.1. Le contrôleur flou de proximité et l'unité de commande des actionneurs

Nous avons utilisé pour réaliser ces deux composantes, le microcontrôleur de type S.O.C : l'esp32, car il satisfait à nos besoins :

- Supporte le wifi pour transformer le contrôleur et l'unité de commande en objets IoT,
- Capacité mémoire assez suffisante pour embarquer et le système d'exploitation free Rtos que nous avons utilisé et le logiciel que nous avons développé :520 KiB RAM, 448 KiB ROM.
- Supporte la PWM que nous utilisons dans la commande des actionneurs car plus adaptée aux commandes que nous calculons pour actionner nos actionneurs.

IV.4.2. Les capteurs

IV.4.2.1. Le capteur de température et d'humidité relative de l'air:

Pour mesurer la température et l'humidité relative de l'air, notre choix s'est porté sur le DHT22 pour les raisons suivantes :

- Il est facile à utiliser car il produit des données de température et d'humidité dans un format numérique, ce qui facilite son interfaçage avec les microcontrôleurs.
- Il est précis : $\pm 0.5^{\circ}\text{C}$ pour la température et $\pm 1\%$ pour l'humidité relative.
- Il s'alimente avec une tension pouvant varier entre 3.3V et 5V donc compatible avec la famille TTL

Chapitre IV

- Il peut être utilisé sans être de nouveau calibré car il est pré-étalonné dès sa fabrication. Maintenant on peut toujours le ré-étalonner si l'on veut s'assurer de sa précision car cette dernière peut varier en fonction de chaque capteur.

On donne à la figure IV.2, son brochage et comment l'interfacier à un microcontrôleur.

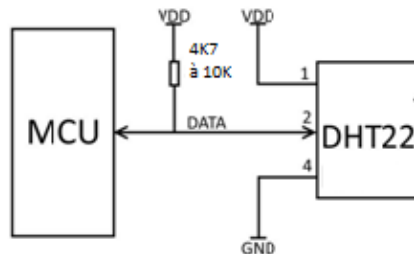


Figure IV.2 : Brochage et Interfaçage du DHT22 à un microcontrôleur.

IV.4.2.2. Le capteur humidité du sol

Pour mesurer l'humidité du sol, nous avons choisi d'utiliser un capteur capacitif parce qu'il est fabriqué à partir de matériaux résistants à la corrosion et par conséquent, permettant des mesures fiables et garantissant une durée de vie prolongée.

On donne à la figure IV.3, suivante le model que nous avons utilisé avec son brochage et on montre à la figure IV.4, comment le brancher à un microcontrôleur.



Figure IV.3 : Brochage du capteur d'humidité du sol utilisé.

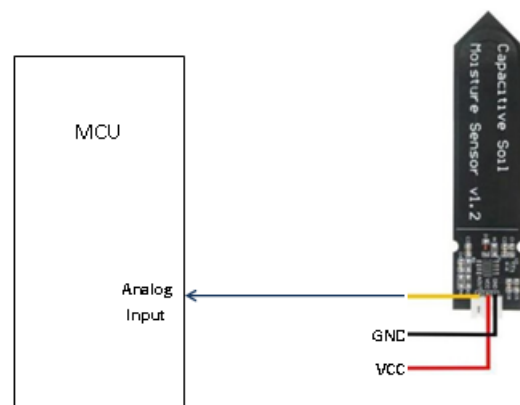


Figure IV.4 : Interfaçage à un microcontrôleur

Chapitre IV

IV.4.2.3. Le capteur de lumière

Pour ce qui est de la mesure de l'intensité de lumière, nous avons préféré d'utiliser une LDR plutôt qu'un autre type de capteur car les LDR sont adaptées aux environnements où la robustesse est requise et c'est notre cas.

La figure IV.5 suivante montre comment l'interfacer à un MCU.

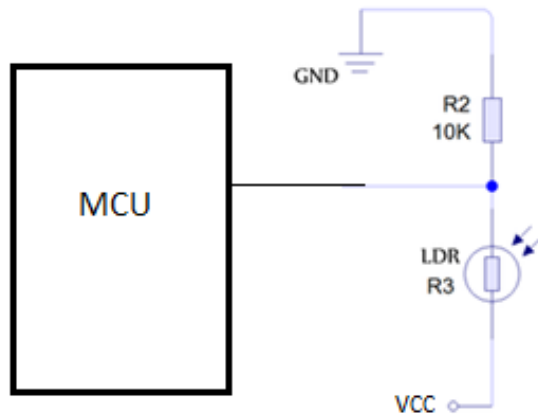


Figure IV.5 : Interfaçage d'une LDR à un MCU

IV.4.3. Les actionneurs

IV.4.3.1. L'irrigation

Pour l'irrigation, il était prévu dans le cahier des charges d'utiliser une électrovanne de régulation proportionnelle (PWM valve dc ou ac) pour contrôler le débit à travers la vanne car adaptée à la commande calculée par le contrôleur flou que nous avons réalisé.

On le rappelle : une électrovanne à commande proportionnelle, est une vanne utilisée pour réguler le débit d'un fluide en faisant varier la taille du passage par l'intermédiaire d'un étrangleur. Le débit régulé permet ensuite d'ajuster les paramètres affectant un processus dans un système, principalement le niveau, la pression dans notre cas et la température.

Une électrovanne à commande proportionnelle utilise un solénoïde comme actionneur pour le positionnement variable de la vanne.

Chapitre IV

On donne à la figure IV.6, et à titre illustratif, un exemple d'un tel type d'ev.



Figure IV.6 : Exemple d'ev à commande proportionnelle

Maintenant, n'ayant pas pu se procurer un tel dispositif, nous l'avons remplacé pour les besoins du projet par un servomoteur, remplacement tout à fait légitime étant donné que les deux se commandent par PWM et surtout que le but est de faire varier la sortie de l'actionneur proportionnellement à la valeur moyenne de la commande (dans notre cas la sortie du contrôleur flou).

Les servomoteurs sont des moteurs à courant continu à engrenages dotés d'un servomécanisme intégré avec une boucle de rétroaction pour permettre un positionnement précis de l'arbre du moteur. La plupart des servomoteurs sont limités en rotation à 180 ou 270 degrés, les servomoteurs à 180 degrés étant plus courants. En revanche, il existe, des servomoteurs spéciaux qui peuvent tourner au-delà de 360 degrés. Les servomoteurs sont disponibles dans une large gamme de tailles et peuvent être contrôlés soit avec un signal PWM analogique, soit avec un signal d'E/S numérique.

Dans le cas d'une commande par pwm, les servo moteurs sont conçus pour être commandés à l'aide d'un signal modulé en largeur d'impulsion (PWM) de 50 Hz, qui produit une impulsion toutes les 20 ms. La position du servo moteur peut être ajustée en modifiant la durée d'impulsion entre 1 ms et 2 ms (Figure IV.7).

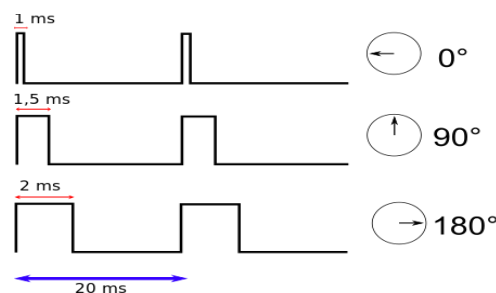


Figure IV.7 : Principe de fonctionnement d'un servo moteur à commande pwm.

Chapitre IV

On montre à la figure IV.8, le type que nous avons utilisé (SG90) et comment l'interfacer à un MCU.

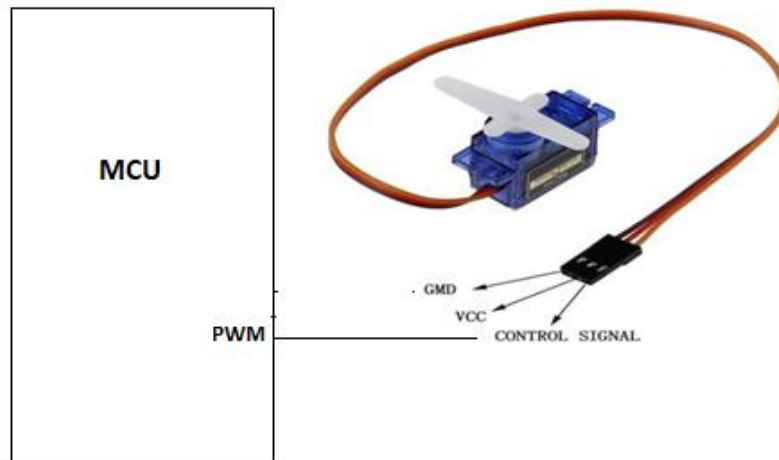


Figure IV.7 : Le servo-moteur sg90 et son interfaçage à un MCU

IV.4.3.2. L'éclairage

Pour ce qui est de l'éclairage, il était aussi question d'utiliser des lampes LED de culture. Maintenant vu leurs prix, et surtout leur non disponibilité, nous avons testé l'éclairage à l'aide de LEDs blanches 12V.

On montre à la figure IV.8, le modèle que nous avons utilisé et à la figure IV.9, comment l'interfacer à un MCU.



Figure IV.8 : LED blanche 12V

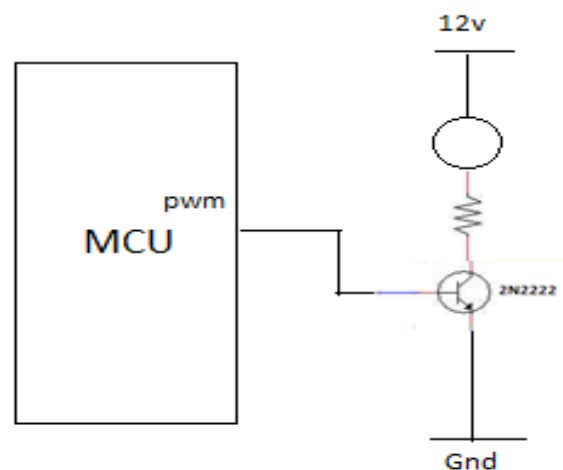


Figure IV.9 : interfaçage d'une LED blanche 12 à un MCU.

Chapitre IV

IV.4.3.3. L'aération et le refroidissement par ventilation

De même, pour ces deux actions, il était question dans le cahier des charges et pour rester dans les conditions réelles, d'utiliser des **ventilateurs qui fonctionnent en 220VAC avec commande pwm** pour contrôler le flux d'air entrant dans la mini-serre ou sortant de celle dernière.

Maintenant et pour les mêmes raisons que celles de l'év proportionnelle, nous avons utilisé un ventilateur 5Vdc, et nous avons réalisé le circuit l'interfaçant avec l'esp32 pour pouvoir le commander en pwm. Le but étant de montrer que la commande calculée par le contrôleur flou fonctionne correctement.

La figure IV.10, montre le modèle que nous avons utilisé et la figure IV.11, comment l'interfacer à un MCU.



Figure IV.10 : ventilateur 5V

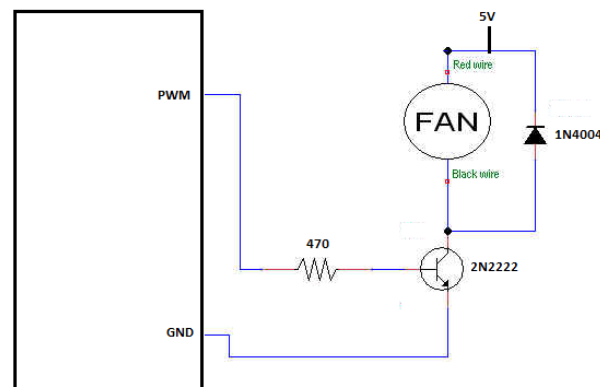


Figure IV.11 : interfaçage avec un mcu

IV.4.3.4. L'ombrage

En pratique les écrans d'ombrage sont motorisé avec commande PWM pour pouvoir les enrouler ou les dérouler en fonction de l'intensité de l'ensoleillement.

Maintenant concernant notre projet, et pour vérifier le bon fonctionnement de notre contrôleur intelligent, nous avons utilisé tout comme pour l'irrigation un servo-moteur du même type et par conséquent avec un interfaçage au MCU identique.

IV.4.3.5. Le chauffage

N'ayant pas pu se procurer un système de chauffage, nous avons utilisé une led blanche 12V pour vérifier le bon fonctionnement de cette action.

Chapitre IV

IV.5. Schémas de l'interfaçage du contrôleur flou avec les capteurs et les actionneurs utilisés

IV.5. 1. Schéma de l'interfaçage du contrôleur flou avec les capteurs utilisés : (figure IV.12)

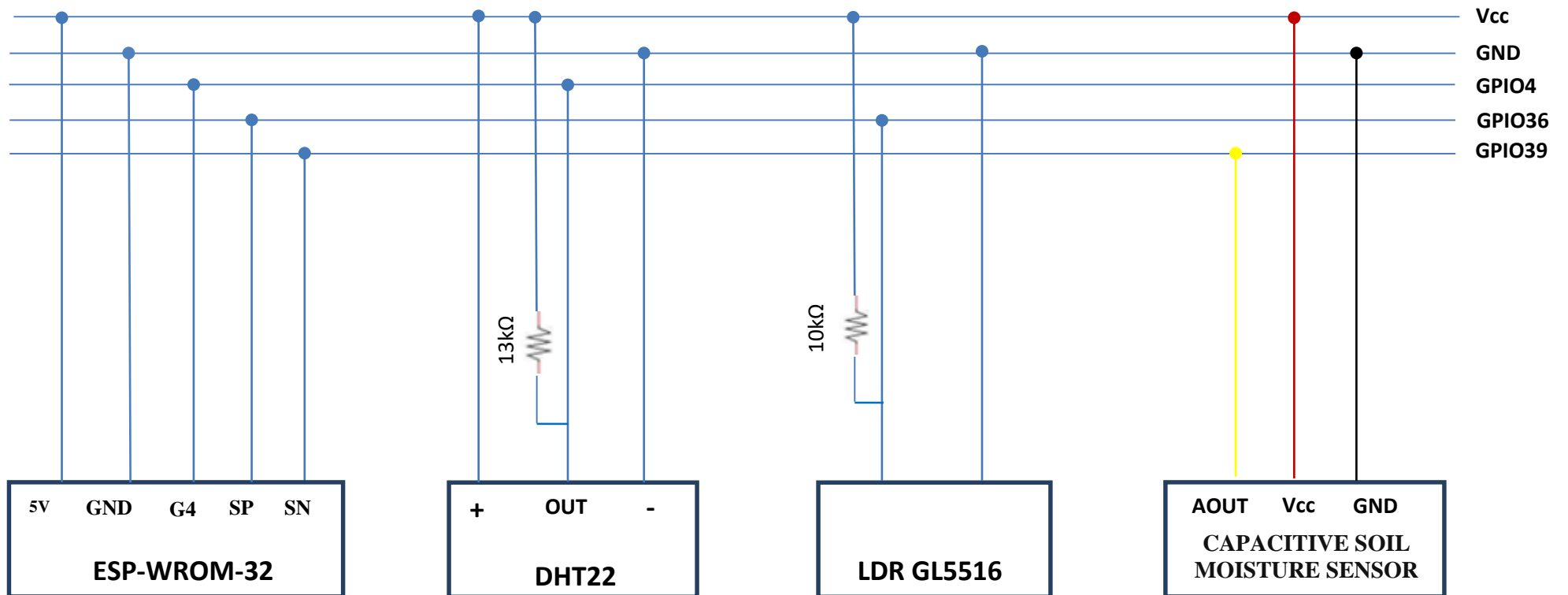


Figure IV.12 : Interfaçage des capteurs utilisés avec le contrôleur flou

Chapitre IV

IV.5. 2. Interfaçage du contrôleur fluo avec les actionneurs utilisés : (Figure IV.13)

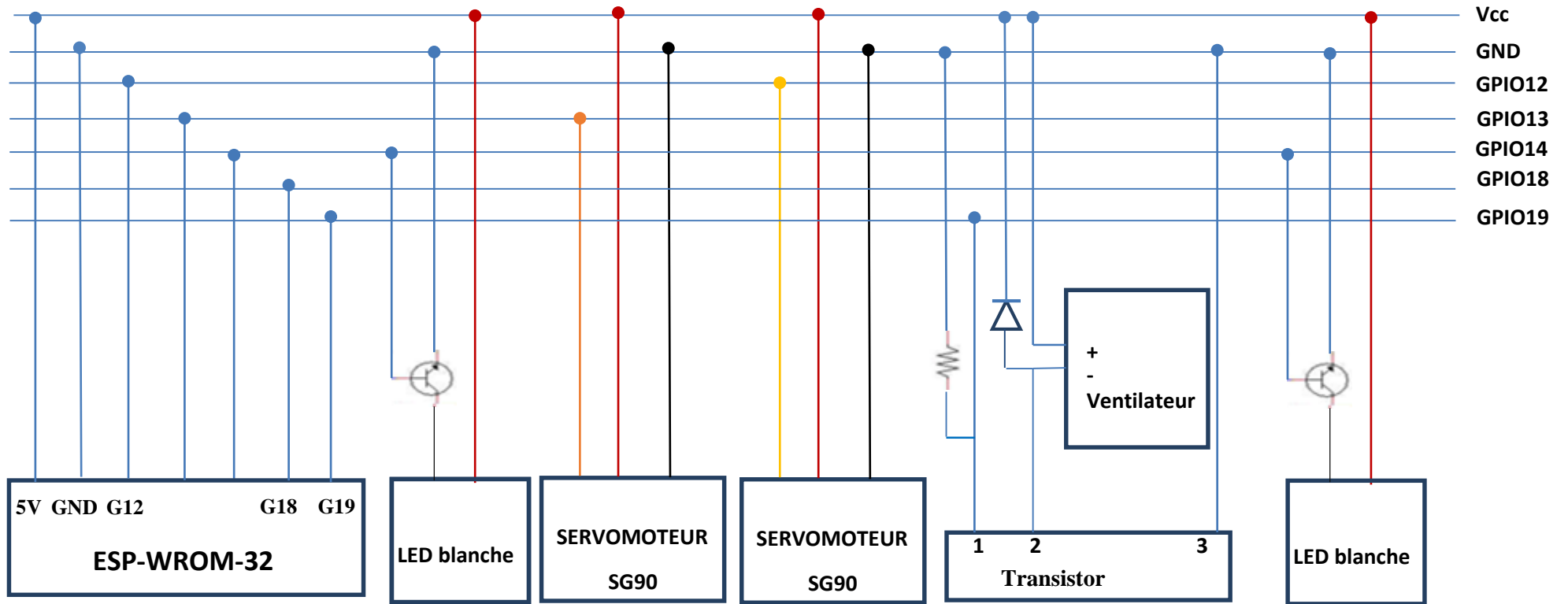


Figure IV.13 : Interfaçage des actionneurs utilisés avec le contrôleur fluo

IV.6. Vue de la maquette

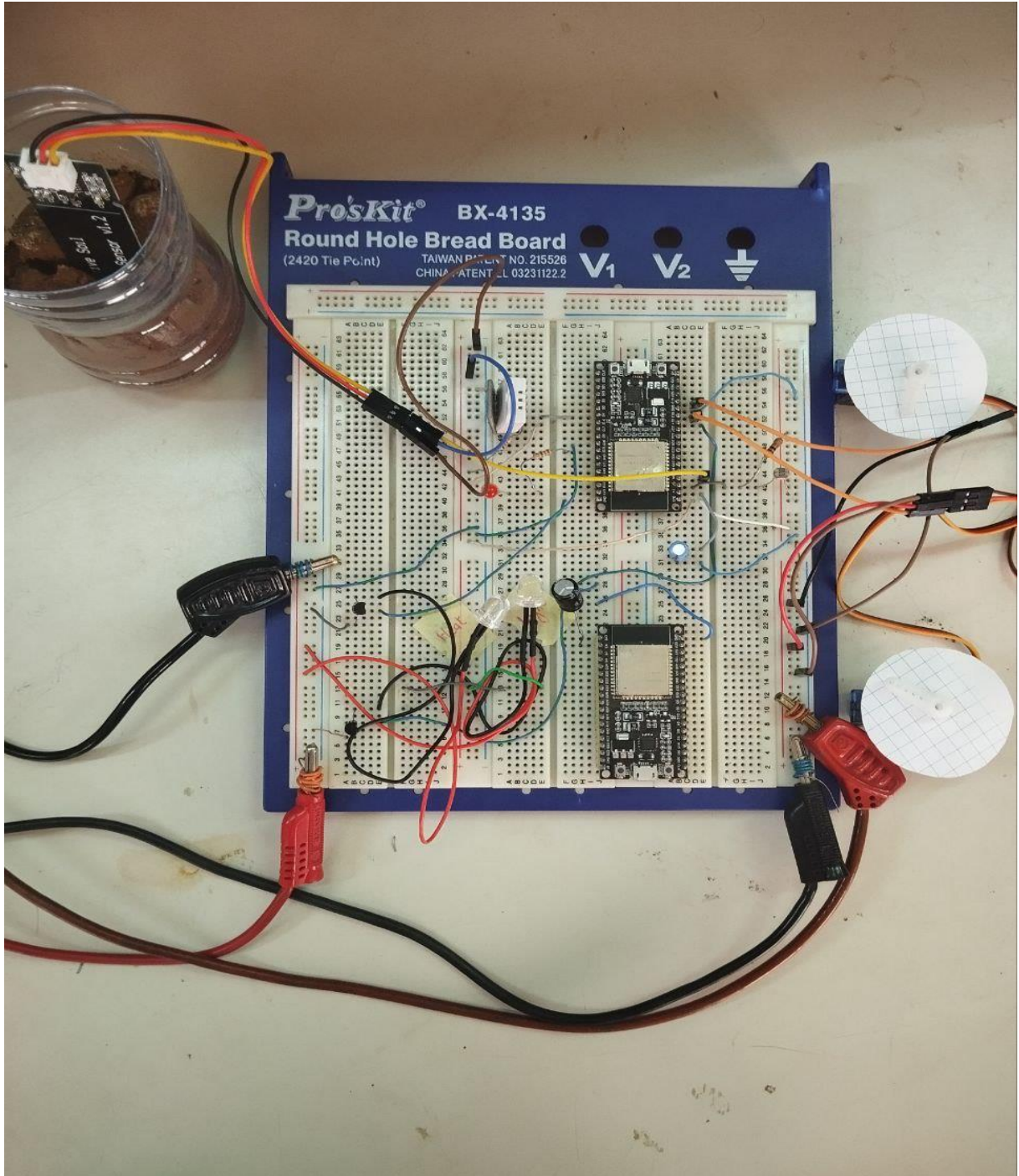


Figure IV.12: vue de la réalisation de projet

Chapitre IV

IV.7. Conclusion :

Dans ce chapitre nous avons présenté le schéma synoptique du système que nous avons réalisé. Ensuite nous avons donné les choix matériels que nous avons faits pour la réalisation de notre système tout en montrant comment les interfacier à un MCU. Nous avons ensuite donné les schémas globaux d'interfaçage des différents composants matériels utilisés au microcontrôleur autour duquel nous avons bâti notre système (l'esp32) ainsi qu'une vue de la maquette sur plaque d'essai de ce dernier.

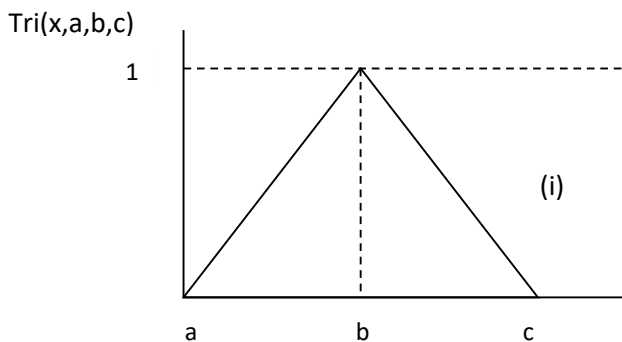
Chapitre V : Réalisation logicielle

V.1. Introduction

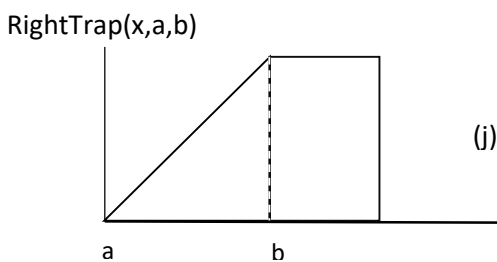
Dans ce chapitre les points suivant vont être présentés et discutés : les fonctions d'appartenance utilisées, le système d'exploitation utilisé, la méthode utilisée pour développer une application basée sur un RTOS dans notre cas freeRtos, l'application embarquée multitâches et l'IHM, réalisées

V.2. Les fonctions d'appartenances utilisées

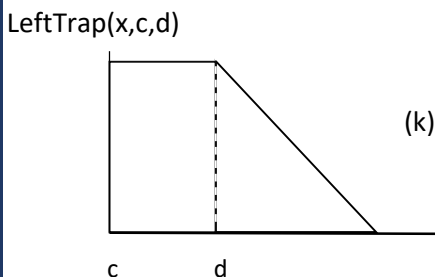
Pour vérifier le bon fonctionnement de l'application embarquée réalisée (i.e : le contrôleur flou), nous nous sommes basées sur les données de ÖzlemAlpay et EbubekirErdem [2]. Ces deux derniers utilisent dans leurs travaux, la fonction triangulaire (Figure V.1) définie par l'équation 1 et les fonctions demi-trapèze droit et gauche définies respectivement par les équations 2 et 3 pour construire les fonctions d'appartenance des entrées et les fonctions d'appartenance des sorties.



$$Tri(x, a, b, c) = \begin{cases} \frac{x-a}{b-a} & \text{si } x \in [a, b] \\ \frac{c-x}{c-b} & \text{si } x \in [b, c] \\ 0 & \text{ailleurs} \end{cases} \quad (1)$$



$$RightTrap(x, a, b) = \begin{cases} 0 & \text{si } x \leq a \\ \frac{x-a}{b-a} & \text{si } x \in [a, b] \\ 1 & \text{si } x \geq b \end{cases} \quad (2)$$



$$LeftTrap(x, c, d) = \begin{cases} 1 & \text{si } x \leq c \\ \frac{d-x}{d-c} & \text{si } x \in [c, d] \\ 0 & \text{si } x \geq d \end{cases} \quad (3)$$

V.3. Le système d'ex

Figure V.1 : fonctions d'appartenance (i) triangulaire, (j) demi-trapèze droit et (k) demi-trapèze gauche.

Chapitre V

De nos jours, de nombreux RTOS conçus avec open source sont disponibles au téléchargement et à l'utilisation gratuitement. Dans ce travail, nous avons sélectionné FreeRTOS. FreeRTOS, est une classe de RTOS conçus pour être suffisamment petits (faible empreinte) pour fonctionner sur un microcontrôleur pour des applications dans les domaines éducatifs et commerciaux. C'est un RTOS qui fournit une solution unique et indépendante pour de nombreux outils d'architecture et de développement différents (<https://freertos.org>).

V.3.1. Mise en œuvre de FreeRTOS dans une application embarquée

Dans une application embarquée, pour faire du multitâche à l'aide de freeRtos, il suffit d'inclure FreeRTOS_AVR.h ou FreeRTOS_ARM.h, selon l'architecture du microcontrôleur, ensuite exploiter les fonctions de l'API de FreeRTOS.

Dans les paragraphes qui suivent nous allons nous limiter à la présentation des points suivants ; les taches dans freeRtos, l'ordonnancement et la concurrence des taches mis en œuvre dans freeRtos étant donné que se sont ceux la que nous avons utilisés dans notre application.

V.3.2. Les tâches dans freeRtos

Ce sont de simples fonctions qui s'exécutent en boucle infinie (Figure V.2) et qui suivent la structure générique suivante :

```
VoidvATaskFunction( void *pvParameters )
{
while(1)
{
    //code de la tâche
}
}
```

Figure V.2 : Structure générique d'une tâche dans

V.3.3. L'ordonnancement des tâches dans freeRtos

L'ordonnancement mis en œuvre dans freeRtos est basé sur le modèle du tourniquet (Round-Robin) avec gestion de priorités. Aussi, c'est ce modèle que nous avons utilisé et sur lequel nous nous sommes basées pour attribuer les priorités aux différentes tâches de l'application embarquée que nous avons réalisée.

Chapitre V

Dans FreeRTOS Il n'y a aucun mécanisme automatique de gestion des priorités. La priorité d'une tâche ne pourra être changée qu'à la demande explicite du développeur.

V.3.4. La concurrence des tâches

FreeRTOS synchronise les tâches en utilisant principalement deux mécanismes : les Sémaphores et les Mutex.

Parmi les tâches de l'application que nous avons développée, il y a des tâches qui partagent certains capteurs. Mutex étant plus facile à utiliser, c'est ce mécanisme que nous avons mis en œuvre pour réaliser l'exclusion mutuelle entre nos tâches concurrentes.

On donne à la figure V.3 suivante, un exemple de bout de code en langage algorithmique, montrant comment fonctionne l'exclusion mutuelle d'une ressource constituée par un capteur entre les tâches qui la partagent.

```
1:while (Capteurxx_occupé); // si Capteurxxoccupé,attendre jusqu'à libre
2:Capteurxx_occupé = 1; // utiliser le Capteurxx
3:FaireLaMesure;
4:Capteur_occupé = 0; // libérer le Capteurxx
```

V.4. Méthode de développement d'une application embarquée à l'aide d'un RTOS

Pour écrire correctement une application qui utilise un RTOS, un développeur est confronté à trois difficultés: Déterminer correctement le nombre de tâches, attribuer à ces tâches des priorités, gérer leurs communications, leurs synchronisations et leurs concurrences.

Dans les paragraphes qui suivent nous allons développer chacun de ces points.

V.4.1. Le nombre de tâches d'une application embarquée

Pour déterminer le nombre correct de tâches que va comprendre l'application embarquée nous avons procédé à sa décomposition.

Pour ce faire, nous avons adopté la méthode « Outside-in ».

Cette méthode comprend principalement deux étapes : identifier les entrées et les sorties du système embarqué et les exprimer dans ce qu'on appelle dans la terminologie de cette

approche, un diagramme de contexte, ensuite attribuer à chacune, éventuellement à plusieurs selon la complexité de leur gestion, une tâche pour obtenir la décomposition cherchée.

Un diagramme de contexte est un diagramme qui représente un système embarqué avec ses périphériques d'entrées et ses périphériques de sorties.

V.5. L'application embarquée multitâche réalisée

Nous avons réalisé cette composante en utilisant l'intelligence artificielle car et comme nous l'avons souligné en introduction, c'est l'approche la plus adaptée à ce genre de réalisation. Les actions sur lesquels nous nous sommes intervenues, sont : l'aération, le refroidissement, l'ombrage, l'irrigation, l'éclairage, et le chauffage.

V.5.1. L'aération

La protection qu'offre la toiture de la serre aux cultures évite les dommages causés par les mauvaises conditions climatiques extérieures : ensoleillement excessif, basses températures, fortes pluies, vents forts, etc., mais elle limite le renouvellement de l'air et ralentit son mouvement à l'intérieur.

Or l'air est utilisé pour la transpiration, la respiration et la photosynthèse. Un taux approprié de renouvellement de l'air intérieur ainsi qu'un mouvement adéquat peuvent fournir des niveaux optimaux de température, d'hygrométrie et de concentration de dioxyde de carbone uniformément dans tout le volume de la serre. En pratique elle peut être naturelle (Figure V.4) ou forcée (Figure V.5).



Figure V.4 : Exemple de système d'aération naturelle constitué de fenêtres rabattables ou basculantes



Figure V.5 : Exemple de système d'aération forcée constitué de ventilateurs/extracteurs

Chapitre V

Une gestion intelligente du système d'aération, consiste à procurer à la plante un confort atmosphérique optimal en agissant sur l'ouverture des fenêtres dans le cas d'un système d'aération naturelle ou sur la vitesse de ventilation dans le cas d'un système d'aération forcée.

Dans l'application embarquée que nous avons réalisée cette gestion est prise en charge par une tâche que nous avons baptisée : aération _Refroidissement.

V.5.2. Le refroidissement

La production en serre pendant l'été peut être un défi. En effet, les niveaux de rayonnement solaire et Les températures de l'air plus élevés qui en résultent, rendent très difficile le maintien de conditions de croissance optimales. En conséquence, l'utilisation de stratégies spécifiques de contrôle environnemental pour maintenir des conditions de production optimales, s'impose.

En pratique ils existent plusieurs techniques de refroidissement dont on peut citer entre autres : refroidissement par brouillard (Figure V.6), refroidissement par ventilation tout comme pour l'aération, refroidissement par ombrage (Figure V.7), ...etc.



Figure V.6 : Refroidissement par brumisateurs

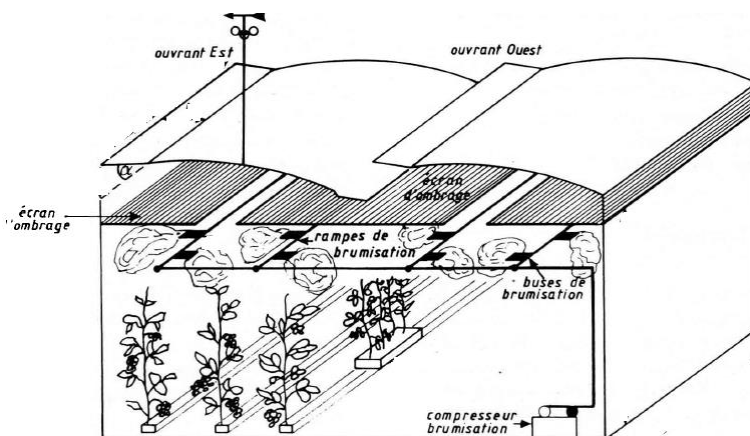


Figure V.7 : Refroidissement par ombrage.

Chapitre V

Une gestion intelligente du système de refroidissement, consiste donc à procurer à la plante un refroidissement optimal en agissant dans le cas où ce dernier est effectué par ombrage, sur la position des écrans ou sur la vitesse de brumisation dans le cas d'un refroidissement par brouillard ou sur la vitesse de ventilation dans le cas d'un refroidissement par ventilation.

Dans l'application embarquée que nous avons réalisée cette gestion est prise en charge par la même tâche que celle que nous utilisons pour l'aération.

V.5.3. Le chauffage

La plupart des cultures sous serre ont des cycles qui surviennent pendant la saison froide. Une baisse de la température des plantes en dessous d'un minimum optimal provoque différents troubles physiologiques, affections, arrêt végétatif, etc., et l'apparition d'un temps glacial avec des températures extrêmement basses d'une durée suffisante peut produire la mort de la plante ou la perte totale de la récolte.

En pratique il existe plusieurs types. On en donne à la figure V.8 suivante, un exemple :



Figure V.8 : Exemple de système de chauffage constitué de générateurs d'air chaud

Donc, une gestion intelligente du système de chauffage consiste à procurer à la plante un confort thermique optimal.

Dans l'application embarquée que nous avons réalisée cette gestion est prise en charge par la tâche que nous avons baptisée : chauffage.

V.5.4. L'irrigation.

Dans une serre, on ne peut pas compter uniquement sur la pluie. On doit donc installer un système d'irrigation.

En pratique il existe plusieurs types. On en donne à la figure V.9 suivante, quelques exemples.



Figure V.9 : Quelques exemples de systèmes d'irrigation

Aussi, une gestion intelligente de l'irrigation consiste par exemple à contrôler la position d'une électrovanne pour fournir la quantité optimale d'eau par jour dont a besoin une plante.

Dans l'application embarquée que nous avons réalisée cette gestion est prise en charge par la tâche que nous avons baptisée : Irrigation.

V.5.5. L'éclairage

De même, dans une serre, on ne peut pas compter uniquement sur l'éclairage naturel. On doit donc installer un système d'éclairage.

En pratique il existe deux techniques d'éclairage :

- L'éclairage à Lampes HPS (high-pressure-sodium) figure V.10.
- L'éclairage à Lampes LED (figure V.11).



Figure V.10 : Exemple d'éclairage avec des lampes HPS

Chapitre V



Figure V.11 :Exemple d'éclairage avec des lampes LED

L'option lampe à LED dépassant de très loin l'éclairage HPS, en raison de son efficacité énergétique améliorée et de sa contrôlabilité avancée, ainsi que du nombre croissant de recherches soulignant ses avantages, c'est ce type d'éclairage qui est le plus utilisé actuellement[<https://www.agritecture.com/blog/2023/9/5/smart-greenhouse-with-dynamic-led-lighting>].

Par conséquent, une gestion intelligente du système d'éclairage consiste à procurer à la plante un confort d'éclairage optimal.

Dans l'application embarquée que nous avons réalisée cette gestion est prise en charge par la tâche que nous avons baptisée : Eclairage

Chapitre V

On donne à la figure V.12 suivante, le diagramme de contexte du système que nous avons réalisé.

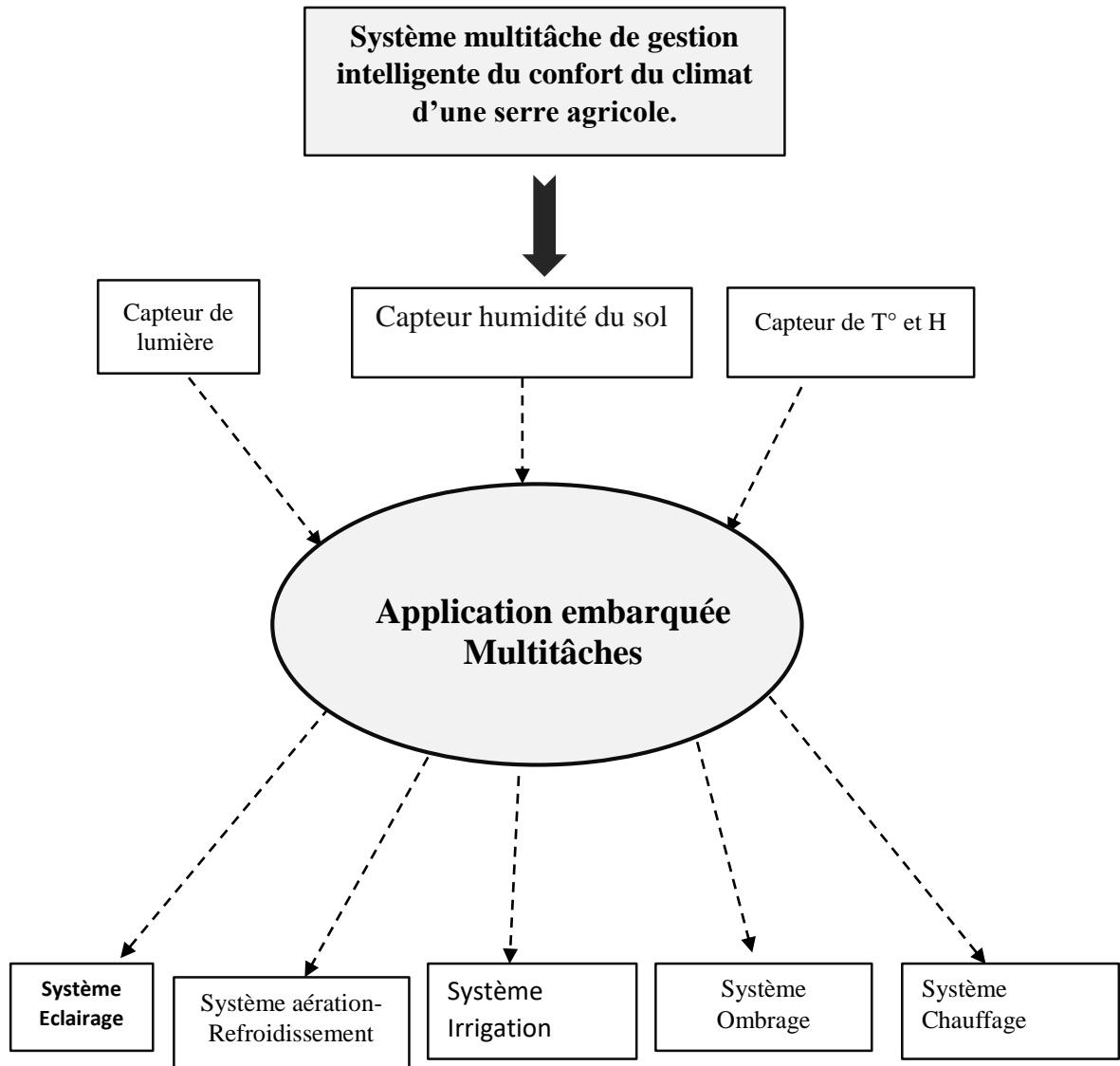


Figure V.12 : Diagramme de contexte du système réalisé

Le cercle au centre du diagramme représente l'application embarquée à réaliser. Les cases rectangulaires représentent les périphériques d'entrée et de sortie. Les flèches représentent les flux des communications d'entrée et de sortie.

De même on donne à la figure V.13 suivante, la décomposition en tâches, à laquelle nous avons abouti, de l'application que nous avons développée, déduite directement du diagramme de contexte :

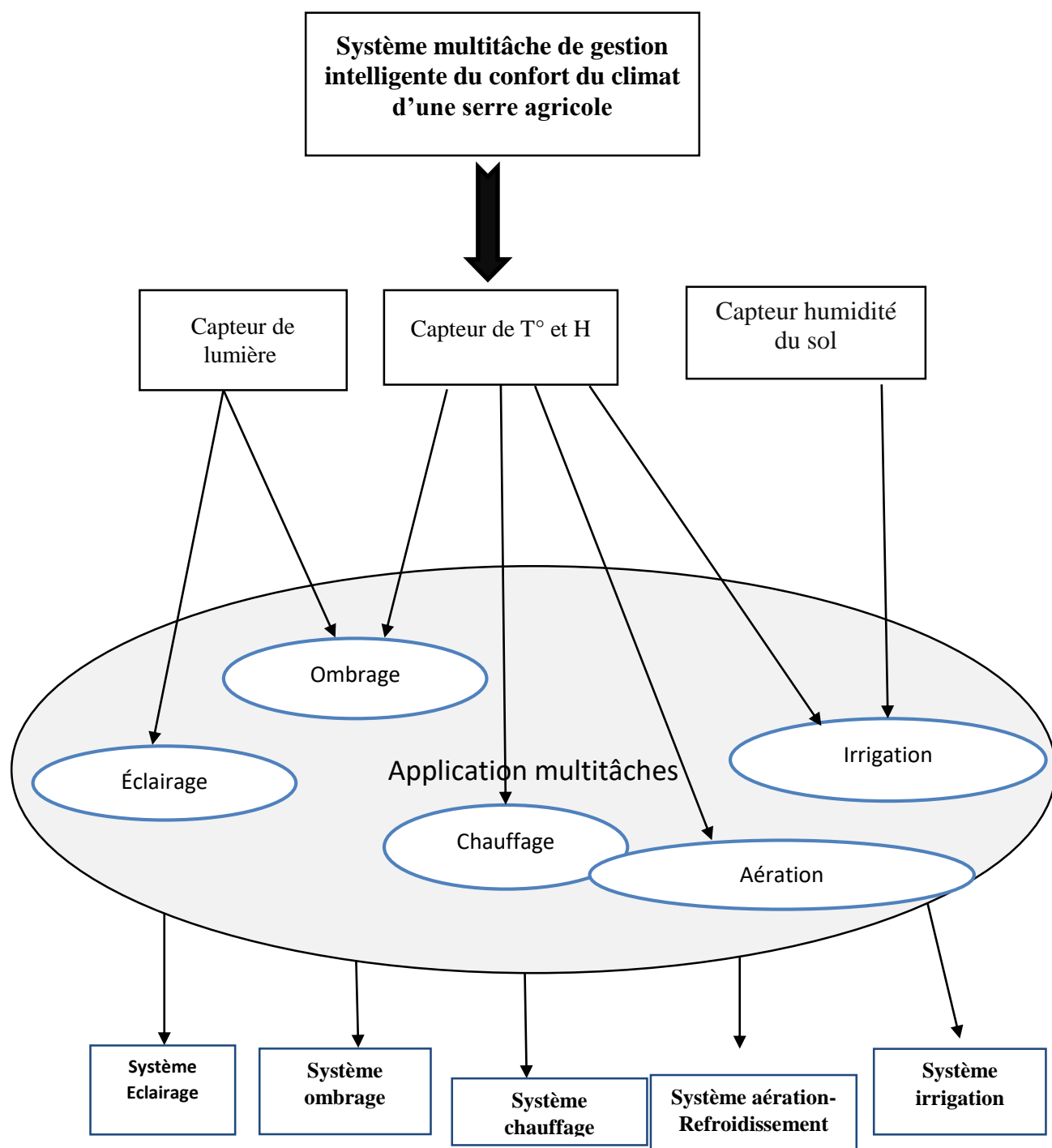


Figure V.13 : Les tâches de l'application embarquée du système réalisé

Dans cette approche, les tâches sont représentées par des ellipses.

Chapitre V

V.5.6. Les taches de l'application embarquée réalisée

On montre à la figure V.14 suivante, comment ces taches sont-elles mises en œuvre dans le contrôleur flou réalisé et aux figures V15, 16, 17 et 18, les organigrammes correspondants.

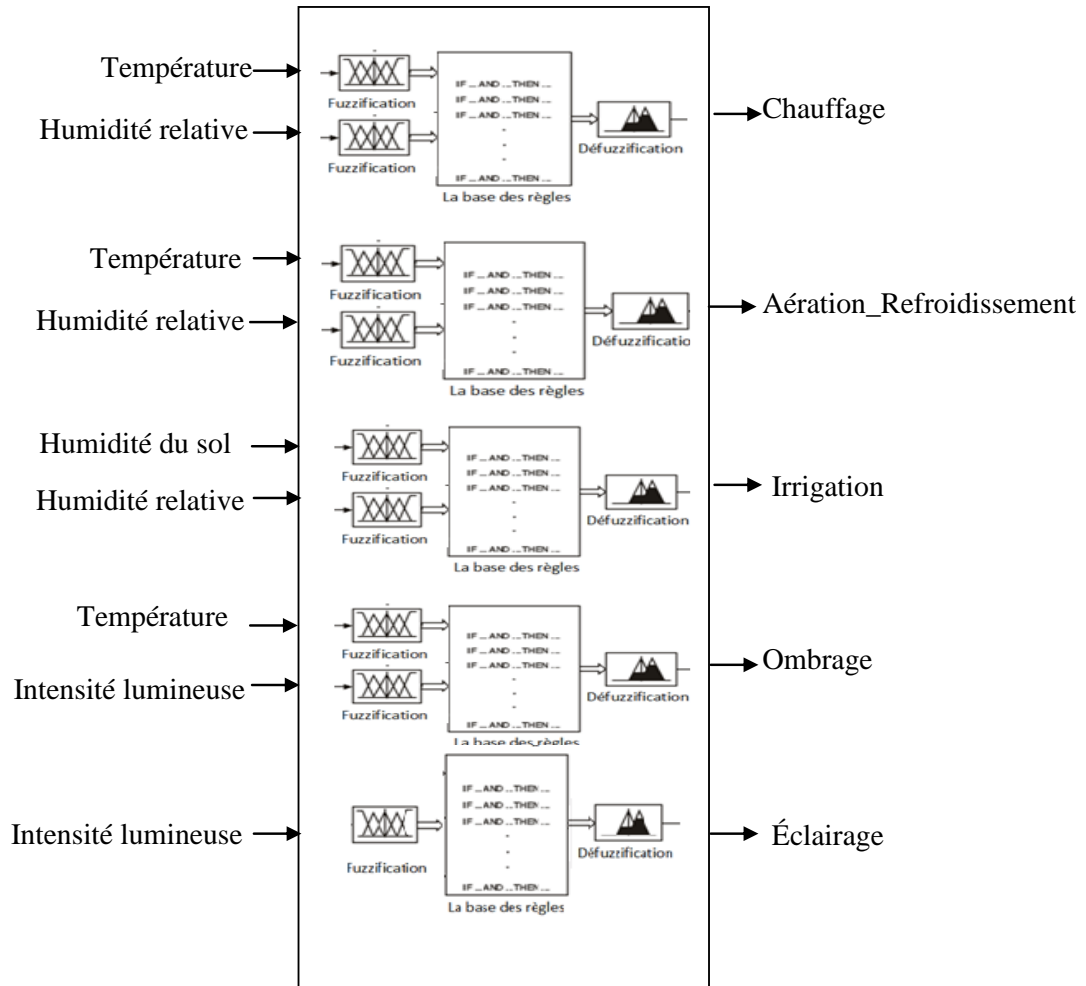


Figure V.14 : Les taches de l'application embarquée réalisée

Chapitre V

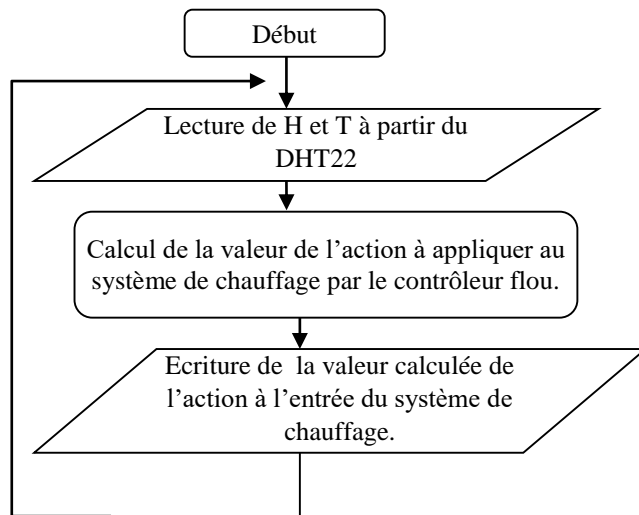


Figure V.15 : Contrôle du système de chauffage.

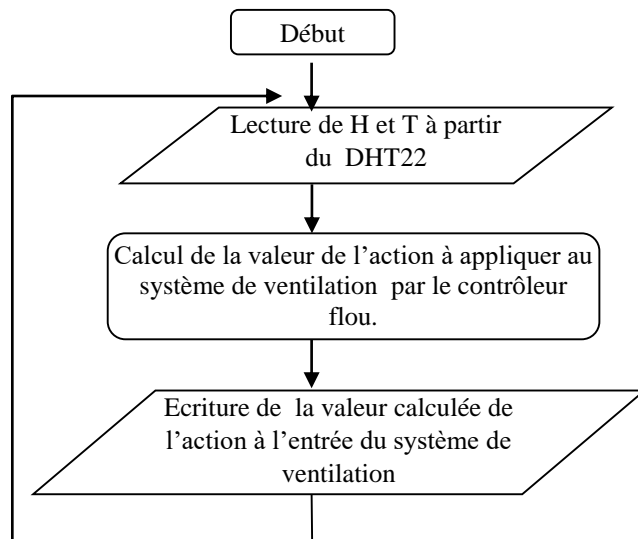


Figure V.16 : Contrôle du système de ventilateur.

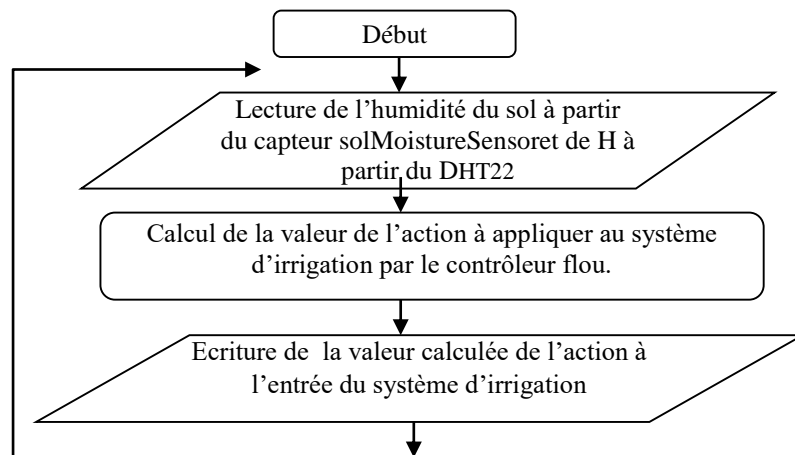


Figure V.17 : Contrôle du système d'irrigation

Chapitre V

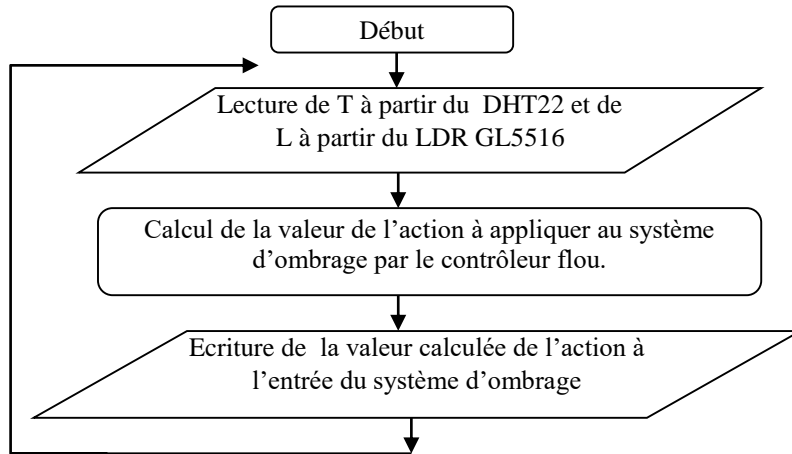


Figure V.18 : Contrôle du système d'ombrage

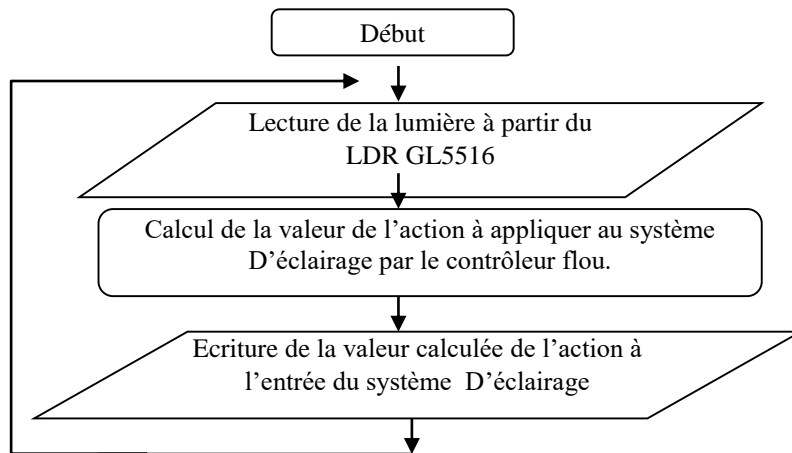


Figure V.19 : Contrôle du système D'éclairage

V.5.7. Programme de la centrale :

La figure V.20 suivante en montre l'organigramme :

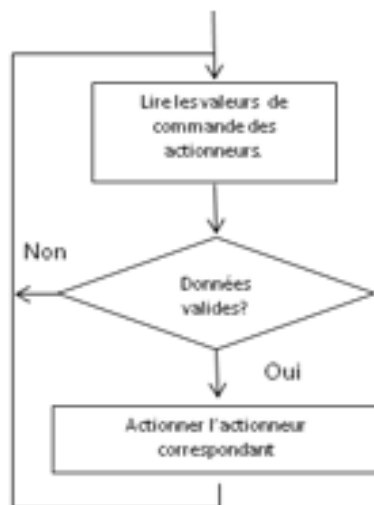


Figure V.20: Organigramme de la commande des actionneurs

Chapitre V

V.5.8. Les règles :

N'étant pas expertes en agriculture, pour vérifier le bon fonctionnement de notre système, nous avons utilisé les règles d'expertise des travaux de ÖzlemAlpay et EbubekirErdem[2]. On résume aux tableaux V.1, 2, 3, 4 et 5 suivants ces dernières.

T\RH	VL	L	M	H	VH
VL	VH	VH	H	H	H
L	H	H	H	M	M
M	M	M	M	M	L
H	L	L	L	L	L
VH	VL	VL	VL	VL	VL

Tableau V.1: Règles de contrôle flou du système de chauffage

T\RH	VL	L	M	H	VH
VL	VL	VL	VL	VL	VL
L	L	L	L	L	L
M	L	M	M	M	M
H	M	M	H	H	H
VH	H	H	H	VH	VH

Tableau V.2: Règles de contrôle flou du système de refroidissement

SM\RH	VL	L	M	H	VH
VL	VH	H	H	M	VL
L	VH	H	M	M	L
M	H	H	M	M	M
H	H	M	M	L	VL
VH	M	M	L	L	VL

Tableau V.3: Règles de contrôle flou du système d'irrigation

Chapitre V

T\LI	VL	L	M	H	VH
VL	VL	VL	L	M	M
L	VL	L	L	M	H
M	VL	L	M	M	H
H	L	L	M	H	H
VH	L	M	M	H	VL

Tableau V.4: Règles de contrôle flou du système d'ombrage

LI	VL	L	M	H	VH
	VH	H	M	L	VL

Tableau V.5: Règles de contrôle flou du système d'éclairage

Chapitre V

V.6. L'Interface Homme Machine réalisée

On donne à la figure V.21 , suivante, comment se présente cette dernière.

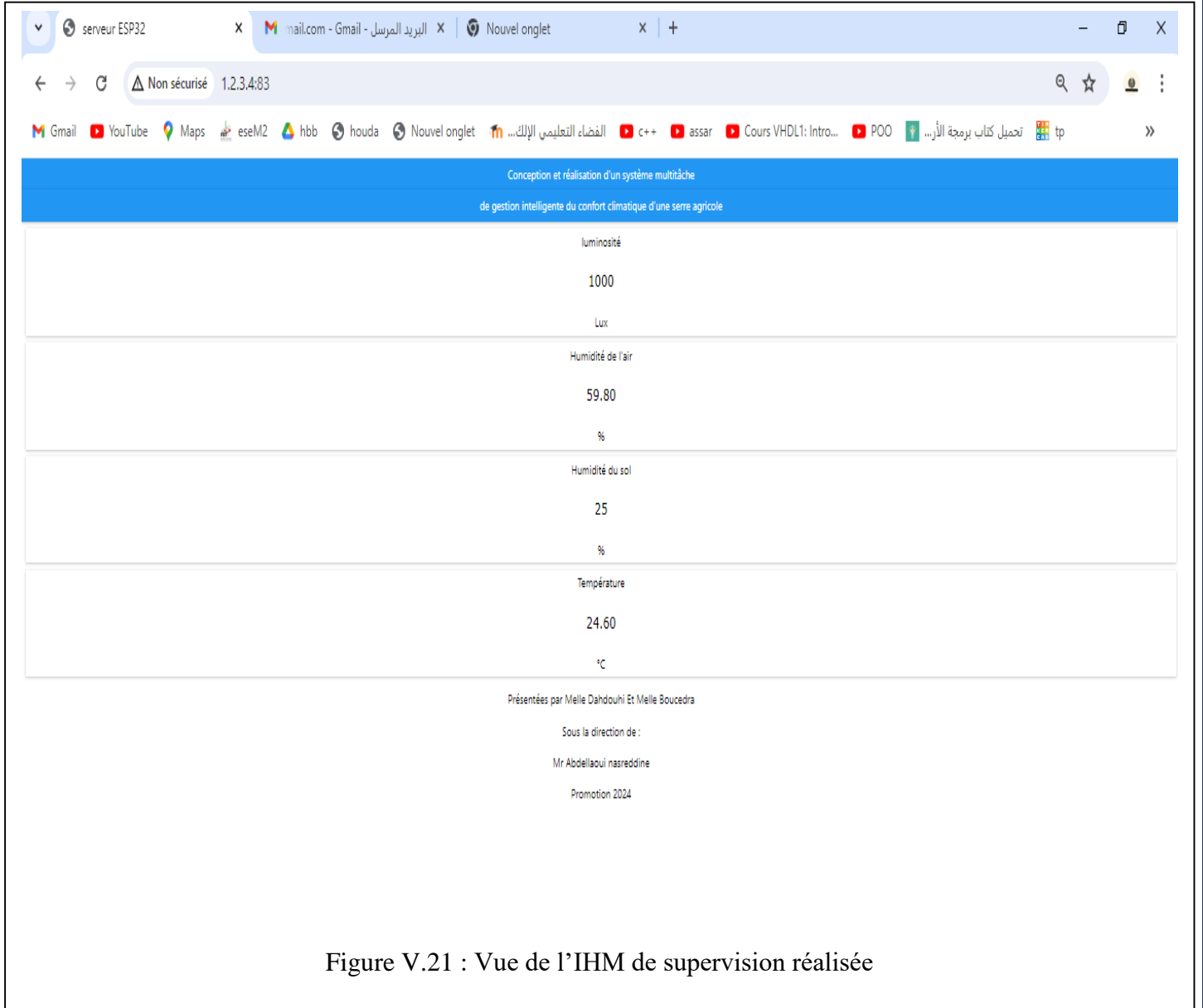



Figure V.21 : Vue de l'IHM de supervision réalisée

Chapitre V



```
COM5
-----Inside HeatingFLC Task-----
HFLC_Humidité: 49.90 %
HFLC_Température: 26.90 °C
HeatingComf: 295 %
Chauffage: 254
-----Inside CoolingFLC Task-----
CFLC_Humidité: 49.90 %
CFLC_Température: 26.90 °C
CoolingComf: 17
Fan: 166
-----Inside LightingFLC Task-----
LFLC_luminosité : 1451 Lux
comfortLighting: 550
LedLighting: 910
-----Inside ShadingFLC Task-----
SFLC_Température 26.90 °C
SFLC_luminosité 1444 Lux
comfortShading: 173
ServoShading: 173
-----Inside IrrigationFLC Task-----
IFLC_sm: 25 %
IFLC_Humidité: 49.90 %
rComfIrrigation: 21
ServoIrrigation: 22
-----Inside HeatingFLC Task-----
HFLC_Humidité: 50.00 %
HFLC_Température: 26.90 °C
 Défilement automatique  Afficher l'horodatage
Les deux, NL et CR v 115200 baud v Effacer la sortie
```

Figure V.22 : Vue de l'affichage des paramètres sur la console de l'IDE Arduino

Chapitre V

The image shows a split-screen view. On the left is the IDE Arduino console window, displaying the output of a program. On the right is a web browser window showing a web application interface for a smart greenhouse climate management system.

IDE Arduino Console Output:

```
-----Inside HeatingFLC Task-----  
HFCLC_Humidité: 58.50 %  
HFCLC_Température: 24.40 °C  
HeatingComf: 346 %  
Chauffage: 385  
-----Inside IrrigationFLC Task-----  
IFLC_sm: 26 %  
IFLC_Humidité: 58.50 %  
rComfIrrigation: 21  
ServoIrrigation: 22  
-----Inside CoolingFLC Task-----  
CFCLC_Humidité: 58.50 %  
CFCLC_Température: 24.40 °C  
CoolingComf: 16  
Fan: 159  
-----Inside LightingFLC Task-----  
LFCLC_luminosité : 1004 Lux  
comfortLighting: 944  
LedLighting: 1925  
-----Inside ShadingFLC Task-----  
SFCLC_Température 24.40 °C  
SFCLC_luminosité 1007 Lux  
comfortShading: 155  
ServoShading: 155  
-----Inside HeatingFLC Task-----  
HFCLC_Humidité: 58.50 %  
HFCLC_Température: 24.40 °C
```

Défilement automatique Afficher l'horodatage

Web Browser Interface:

Conception et réalisation d'un système multitâche
de gestion intelligente du confort climatique d'une serre agricole

luminosité	978
Lux	
Humidité de l'air	59.30
%	
Humidité du sol	26
%	
Température	24.50
°C	

Présentées par Melle Dahdouhi Et Melle Boucedra
Sous la direction de :
Mr Abdellaoui nasreddine
Promotion 2024

Figure V.23 : Vue de l'IHM de supervision réalisé et de l'affichage des paramètres sur la console de l'IDE Arduino

V.7. Conclusion

Dans ce chapitre, nous avons présenté le logiciel que nous avons développé. Nous avons indiqué les tâches du contrôleur que nous avons développé ainsi que l'IHM de suivi à distance des paramètres d'une serre.

Conclusion générale

Nous avons réalisé dans notre projet, d'un système multitâche et de supervision via Internet avec une gestion intelligente à l'aide de la logique floue, confort du climat atmosphérique destiné à une serre agricole.

Ce travail nous a été très bénéfique car il nous a permis de consolider nos connaissances, de beaucoup apprendre et de se familiariser davantage avec des techniques de développement qui nous ont permis d'améliorer nos compétences et nos acquis dans la programmation en poo et des interfaces graphiques.

Enfin, ce projet peut constituer une très bonne assise et un très bon prélude pour des travaux se faisant dans ce domaine car nous y avons réunis toutes les bases nécessaires.

Chapitre V

Bibliographie :

- [1] Fuzzy Controllers, https://research.iaun.ac.ir/pd/naghsh/pdfs/UploadFile_4810.pdf
- [2] Özlem Alpay, Ebubekir Erdem
- [3] The Control of Greenhouses Based on Fuzzy Logic Using Wireless Sensor Networks
- [4] International Journal of Computational Intelligence Systems Vol. **12(1)**; 2019,
- [5] <https://www.atlantis-pess.com/journals/ijcis/>
- [6] Sheetal Vatari, Aarti Bakshi, Tanvi Thakur Green House by using IoT and Cloud Computing 2016 IEEE International Conference on Recent Trends in Electronics Information & Communication Technology
- [7] Ravi Kishore Kodali, Vishal Jain and Sumit Karagwal —IoT based Smart Greenhouse| 2016 IEEE Region 10 Humanitarian Technology Conference
- [8] P.Rajalakshmi, S.Devi Mahalakshmi —IoT based crop-field monitoring and irrigation automation| 2016 IEEE International Conference on Intelligent System and Control
- [8] Yin Jie, Ji yong Pei, LI Jun,guo Yun,Xu Wei, “Smart Home System based on IOT Technologies” International conferences on computational and Information Science Issue: November-2013
- [9] D.D.Chaudhary, S.P.Nayse, L.M.Waghmare, “Application of wireless sensor network for greenhouse parameter in precision agriculture”,International Journal of Wireless & Mobile Networks (IJWMN) Vol. 3, No. 1, February 2011 PP: 140-149
- [10] Rahul Belsare, Komal Deshmukh, Mayuri Patil, Prof. Hattarge A.M., “Smart Green House Automation” International Journal of Computer Science & Engineering Technology (IJCSET) PP: 1127-1129

Chapitre V

[11] Moataz Soliman¹, Tobi Abiodun¹, Tarek Hamouda, Jiehan Zhou¹, Chung-Horng Lung, “Smart Home: Integrating Internet of Things with Web Services and Cloud Computing” IEEE International Conference on Cloud Computing Technology and Science Issue: 2015

[12] Feijs, L. “Multi-tasking and Arduino: Why and How? Design and semantics of form and movement”. Retrieved May 1, 2018, from https://ccrma.stanford.edu/~gurevich/Feijs_ArduinoMultitasking.pdf, 2013.

[13] Buonocunto, P., Biondi, A., & Loreface, P. “Real-Time Multitasking in Arduino”. Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES), <https://doi.org/10.1109/SIES.2014.7087331>. Pisa, Italy, 2014.

[14] Barry, R. “Multitasking on AVR”. Retrieved May 3, 2018, from d1.amobbs.com/bbs_upload782111/files_19/ourdev_488657.pdf, 2004.

[15] Pantano, N., & Timmerman, M.P. “Real-Time Operating System on Arduino. OSSEC mini project”. Retrieved May 5, 2018, from <https://id.scribd.com/document/222365331/Nicolas-Pantano-Report>, 2011.

[16] Principes fondamentaux du noyau FreeRTOS

[17] https://docs.aws.amazon.com/fr_fr/freertos/latest/userguide/dev-guide-freertos-kernel.html