

Université de Gafsa
Institut Supérieur des Sciences Appliquées et de Technologie de GAFSA
Département Informatique et Télécommunication



Réalisation d'un prototype de système d'irrigation à base de Raspberry PI et Node RED Red pour les zones vertes

Présenté et soutenu par :

Mariem Ben Mbarek

Dhouha Ben Amara

En vue de l'obtention de Diplôme Licence en

Ingénierie des systèmes informatiques

Sous la Direction de :

Mr. Mounir Telli

Mr. Ilyess Soualmia (CPG)

Soutenu le 05/06/2023

Devant le jury composé de :

Président :

Rapporteur :

A.U.2022-2023

Dédicaces

Dédicaces

Remerciement

Au terme de ce travail, nous tenons à exprimer notre profonde gratitude à notre cher professeur et encadrant ***M.Telli Mounir*** pour son suivi et pour son énorme soutien, qu'il n'a cessé de nous prodiguer tout au long de la période du projet .

Nous tenons à remercier également notre encadrant ***M. Ilyess Soualmia*** pour le temps qu'il a consacré et pour les précieuses informations qu'il nous prodiguées avec intérêt et compréhension.

Nous adressons aussi nos vifs remerciements aux membres des jurys pour avoir bien voulu examiner et juger ce travail.

Nous ne laisserons pas cette occasion passer sans remercier tous les enseignants et le personnel de **ISSAT Gafsa** et particulièrement pour leur aide et leurs précieux conseils et pour l'intérêt qu'ils portent à notre formation.

Enfin, mes remerciements à tous ceux qui ont contribué de près ou de loin au bon déroulement de ce projet.

Liste des figures

Figure 1 : Structure d'un système embarqué	4
Figure 2 : Classification des différents types de systèmes.	6
Figure 3 : Schéma synoptique d'un SE	6
Figure 4 : Les domaines d'application des systèmes embarqués	8
Figure 5 : Concept d'IoT.....	10
Figure 6 : L'interconnexion d'IoT avec différent objets	11
Figure 7 : Le protocole utilisé dans l'IoT	12
Figure 8 : Les composantes utilisé dans l'IoT	12
Figure 9 : Architecture de l'IoT.....	13
Figure 10 : L'utilisation d'un IoT dans le domaine agricole	14
Figure 11 : Prototype d'une serre intelligente	19
Figure 12 : Serre tunnel.....	20
Figure 13 : Serre multi chapelle.....	20
Figure 14 : Principe de la Photosynthèse.....	21
Figure 15 : Catre Arduino.	23
Figure 16 : Catre Wemos	23
Figure 17 : Catre Raspberry pi.....	24
Figure 18 : Capteur DHT11	24
Figure 19 : Capteur MQ2.....	25
Figure 20 : Ventilateur	25
Figure 21 : Capteur LM35	26
Figure 22 : Détecteur de lumière LDR	26
Figure 23 : Électrovanne d'eau	26
Figure 24 : Le contrôle à distance.....	27
Figure 25 : Carte Wemos connecter avec les capteurs	27

Liste des tableaux

Figure 26 : Connexion du capteur de température LM35 avec la carte Arduino	28
Figure 27 : Première partie du programme ‘carte Wemos’	29
Figure 28 : Deuxième partie du programme ‘carte Wemos’	29
Figure 29 : Troisième partie du programme ‘carte Wemos’	30
Figure 30 : Quatrième partie du programme ‘carte Wemos’	30
Figure 31 : La moniteur série du traitement du carte Wemos	31
Figure 32 : Première partie du programme de la carte Arduino	31
Figure 33 : Deuxième partie du programme de la carte Arduino	32
Figure 34 / Carte Raspberry Pi.....	34
Figure 35 : Évolution des cartes Raspberry pi.....	35
Figure 36 : Les composants standards de la carte Raspberry pi	36
Figure 37 : Chargeur de carte Raspberry Pi.....	37
Figure 38 : Carte micro SD	37
Figure 39 : Câble HDMI	38
Figure 40 : Boiter pour la Raspberry Pi.....	38
Figure 41 : SD Card Formatter	39
Figure 42 : Raspberry Pi Imager.....	39
Figure 43 : Fichier SSH	40
Figure 44 : Adresse IP du Raspberry	40
Figure 45 : Interface du PuTTY.....	40
Figure 46 : La page de connexion avec Raspberry	41
Figure 47 : Le serveur MQTT.....	42
Figure 48 : Les codes d’installation	42
Figure 49 : Installation de Mosquito.....	43
Figure 50 : Logiciel Node-red.....	43
Figure 51 : Installation de Node-Red.....	44

Liste des tableaux

Figure 52 : Lancement de Node-Red.....	44
Figure 53 : Fenêtre terminale de Node-Red dans Raspberry.....	45
Figure 54 : Interface de Node-Red.....	46
Figure 55 : 1ère étape d'implémenter une fichier JSON	50
Figure 56 : 2ème étape d'implémentation.	50
Figure 57 : Le programme de fonctionnement d'une motopompe d'irrigation et de contrôle température/ humidité/gaz.	51
Figure 58 : Programme du système de contrôle avec DHT11	52
Figure 59 : Programme de système d'activation de motopompe	53
Figure 60 : Programme pour la récupération des données météo depuis le site OpenWeatherMap.	53
Figure 61 : Installation de bibliothèque Remote-RED	54
Figure 62 : La bibliothèque Remote-RED	54
Figure 63 : Access node.	54
Figure 64 : Configuration de Remote-RED.....	55
Figure 65 : Installation de l'application Remote-RED sur smartphone.	55
Figure 66 : Connexion de l'application Remote-RED avec programme.....	56
Figure 67 : Création d'host Name.....	56
Figure 68 : Configuration au niveau de service virtuel.	57
Figure 69 : Configuration au niveau de DDNS.	57
Figure 70 : Programme no-ip.....	57
Figure 71 : Dashboard du programme développé.....	57
Figure 72 : Prototype réel d'une serre intelligente.	58

Liste des tableaux

Sommaire

<i>Remerciement</i>	IV
<i>Liste des figures</i>	VII
<i>Introduction générale</i>	1
<i>Chapitre 1 : Etat de l'art</i>	3
1. Introduction.....	4
2. Définition	4
3. Types des systèmes embarqués.....	5
4. Classification d'un système embarqué	5
5. Architecture générale de système embarqué	6
6. Caractéristiques principales d'un système embarqué.....	7
7. Domaines d'application d'un système embarqué	7
8. Fiabilité	8
9. Historique.....	9
10. Contraintes.....	9
11. L'internet des objets (IoT).....	10
12. Notion de base sur l'IoT	10
13. Définition d'IoT.....	11

Sommaire

14.	Protocoles applicatifs utilisées à l’IoT	11
15.	Cadre du projet	14
15.1.	Présentation de la CPG	14
15.2.	Contexte du projet	15
16.	Spécification des besoins.....	16
16.1.	Les besoins fonctionnels	16
16.2.	Les besoins non fonctionnels.....	16
17.	Conclusion	17
<i>Chapitre 2 : Approche d’une serre intelligente via une carte RaspBerry.....</i>		<i>18</i>
1.	Introduction.....	19
2.	Définition d’une serre intelligente	19
3.	Les types des serres.....	20
4.	Les besoins en lumière des plantes	21
5.	Performance d’une serre intelligente	22
6.	Matériels utilisés	23
7.	Contrôle du système à distance.....	27
8.	Câblage de la carte Wemos.....	27
9.	Câblage de la carte Arduino.....	28
10.	Programmes utilisés.....	28
11.	Conclusion	32
<i>Chapitre 3 : Expérimentations.....</i>		<i>33</i>
1.	Introduction.....	34

Sommaire

2. Carte Raspberry.....	34
3. Les étapes de création d'une interface de supervision sous le logiciel Node-Red 39	
4. Dashboard	57
5. Réalisation et conception	58
6. Conclusion	58
Conclusion Générale.....	59

Introduction générale

De nos jours, une difficulté apparaît que le secteur agricole doit faire face relative aux conditions météorologique instable et le réchauffement de la planète qui a un impact négatif sur les cultures.

Les ingénieurs et les scientifiques cherchent des solutions répondant aux besoins alimentaires suffisants et allant au-delà des menaces du changement climatique, l'agriculture intelligente utilise des technologies avancées telles que Big Data, l'internet des objets (iot), ...etc.

L'IOT facilite l'automatisation de l'agriculture, le collecte des données sur le terrain et les analyse, afin que les agricultures puissent prendre une décision précise en matière de production de culture de haute qualité.

L'agriculture intelligente face au climat semble être un moyen d'augmenter la production alimentaire tout en s'adaptant au changement climatique. L'agriculture sous serre vise à améliorer le rendement des fruits, des légumes, des cultures, ...etc. Mais les serres sont connues pour être énergivores. La construction de serres intelligentes leur permettrait d'économiser de l'argent et d'atteindre leurs objectifs agricoles.

Une serre intelligente dotée de la technologie IoT peut ainsi contrôler l'environnement ; des serveurs dans le cloud peuvent accéder à distance au système informatique de la serre lorsqu'elle est connectée. Le traitement des données et la suppression de la surveillance manuelle permanente permettent alors de réaliser des économies de coûts.

D'après certaines projections, un nombre croissant d'agriculteurs devrait faire appel à l'IoT dans les prochaines années. À la clé, des gains d'efficacité, des rendements en hausse, une réduction des tâches manuelles, et bien d'autres avantages.

L'agriculture en serre vise à augmenter la production de fruits, légumes, cultures, ...etc. Cependant, comme nous le savons tous, les serres sont énergivores. La construction de serres intelligentes leur permettra d'économiser de l'argent et d'atteindre leurs objectifs agricoles.

Introduction générale

Par conséquent, la serre intelligente utilisant la technologie de l'Internet des objets peut contrôler l'environnement. Après la connexion, le serveur dans le cloud peut accéder à distance au système informatique de la serre. Traitez les données et éliminez la surveillance manuelle permanente, réduisant ainsi les coûts.

Selon certaines prévisions, dans les prochaines années, de plus en plus d'agriculteurs devraient utiliser l'Internet des objets. Le résultat est une efficacité accrue, une production accrue, moins de tâches manuelles et plus de travail effectué.

Dans le système intelligent que nous allons développer, nous allons utiliser la technique d'IoT (internet des objet) qui va servir pour la Compagnie des phosphates de Gafsa comme outil d'aide à fertilisation et irrigation des zones vertes. Ce système permet de contrôler intelligemment un ensemble des plantes des zones vertes appartenant au nouveau projet de fertilisation de la CPG en vue de garantir une bonne performance

Ce rapport est composé de trois chapitres, qui sont organisés comme suit :

- Le premier chapitre : Etat de l'art, présente les systèmes embarqués et notion de base d'un IoT.
- Le deuxième chapitre présente une approche d'une serre intelligente contrôlée via une carte Raspberry y compris une petite introduction, l'identification des matériel utilisé, l'algorithme développé et les performances du système.
- Le troisième chapitre consiste à la mise en œuvre expérimentale via la carte Raspberry Pi, en touchant dans ce chapitre les étapes de création une interface de supervision et de contrôler le fonctionnement de motopompe.

Chapitre 1 : Etat de l'art

1. Introduction

Les systèmes embarqués trouvent de plus en plus leurs applications dans plusieurs domaines : militaire, médicales, loisir, transport, ...etc.

Un système embarqué (SE) est un système informatisé spécialisé, qui constitue une partie intégrante d'un système plus large ou une machine. Un (SE) est une combinaison entre la partie hardware et la partie software.

Un sous-système intelligent est capable d'exécuter un ensemble prédéfini de tâche. Il contient plusieurs capteurs lui permettant d'acquérir de l'information du monde qui l'entoure, il peut contenir des actionneurs ou des interfaces associées à ces capteurs et plusieurs liens de communications avec d'autre SE.

2. Définition

Un système embarqué est un système mixte. Il combine le hardware et le software pour accomplir une fonction particulière. Il est une partie d'un système plus complexe qui n'est pas forcément un « ordinateur ». D'autre part, c'est un système électronique et informatique autonome (il ne possède pas d'entrée et de sortie standard), souvent fonctionnant en temps réel, spécialisé dans une tâche bien précise.

Un système embarqué peut être défini comme étant un système de traitement de l'information qui répond aux stimulus externes dans un délai bien définie. Il peut être aussi défini comme un système électronique et informatique autonome ne possédant pas des entrées/sorties standards comme un clavier ou un écran d'ordinateur (PC).

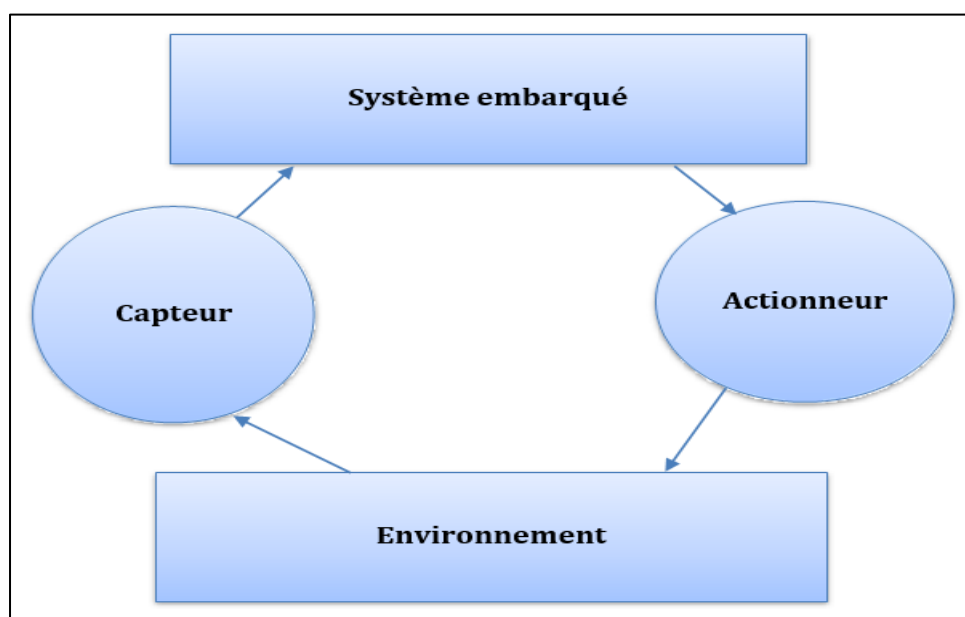


Figure 1 : Structure d'un système embarqué

3. Types des systèmes embarqués

Il existe quatre types d'un systèmes embarqué telles que pour l'utilisation de calcul généraux, systèmes de contrôle, traitement du signal et le communication réseaux.

3.1. Calculs généraux

Ce type de systèmes embarqué utiliser pour créer des applications similaires à une application de bureau mais empaquette dans un système embarqué comme les jeux vidéo, set-top box.

3.2. Systèmes de contrôle (automatisme)

Utiliser pour le contrôle de systèmes en temps réel, moteur d'automobile, processus chimique, systèmes de navigation, ...etc.

3.3. Traitement du signal

Calcul sur grosses quantité de données, Radar, Sonar, compression vidéo...

4. Classification d'un système embarqué

Il existe plusieurs classifications de systèmes embarqué essentiellement qui se devise enttrois formes. Systèmes transformationnel, systèmes interactif, systèmes réactifs ou en temps réel.

4.1. Systèmes transformationnels

C'est une activité de calcul, qui lit ses données et ses entrées lors de son démarrage, qui fournit ses sorties puis meurt.

4.2. Systèmes interactifs

Réagit aux stimuli en entrées à sa propre vitesse (aucune contrainte sur le temps de réponse).

4.3. Systèmes réactifs ou temps réel

Produit des sorties en fonctions des entrées et en réaction à des événement produit par

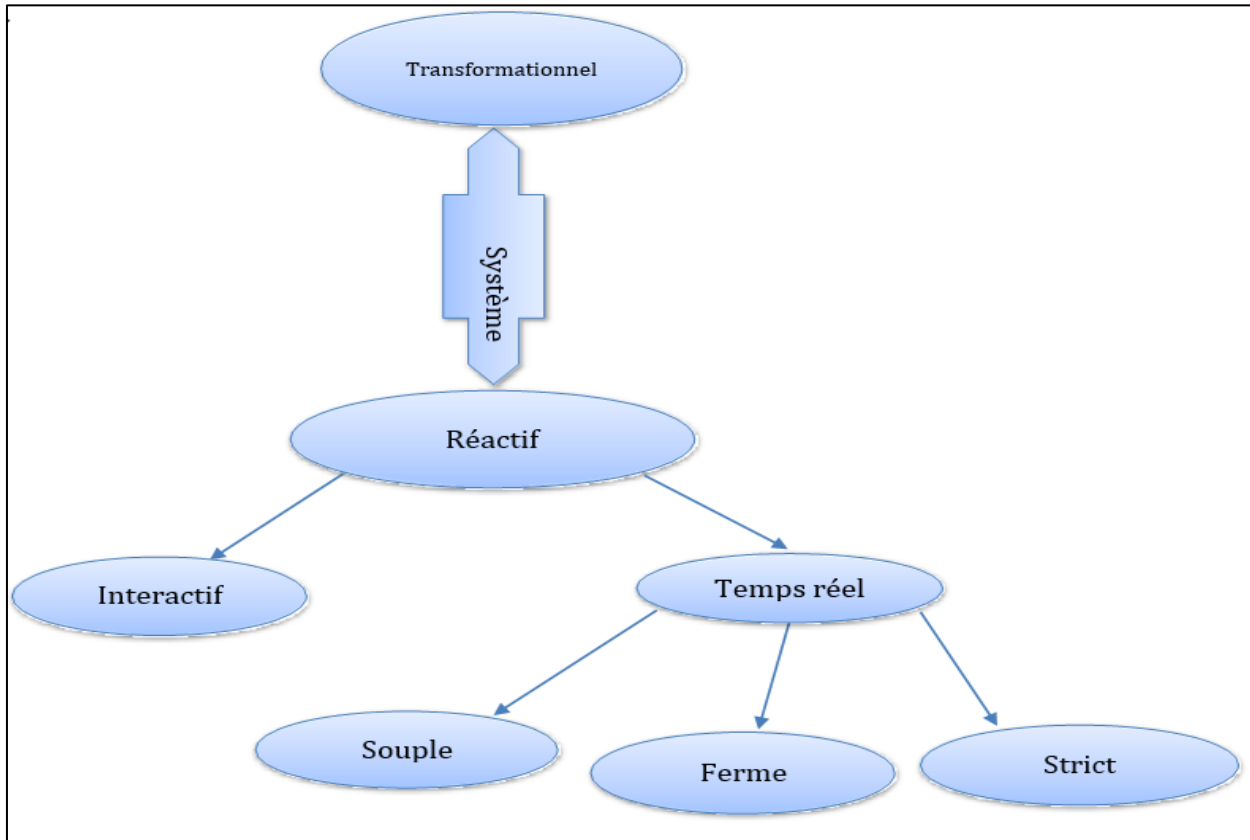


Figure 2 : Classification des différents types de systèmes.

5. Architecture générale de système embarqué

L'architecture d'un système embarqué se définit par le schéma présenté par la figure 3.

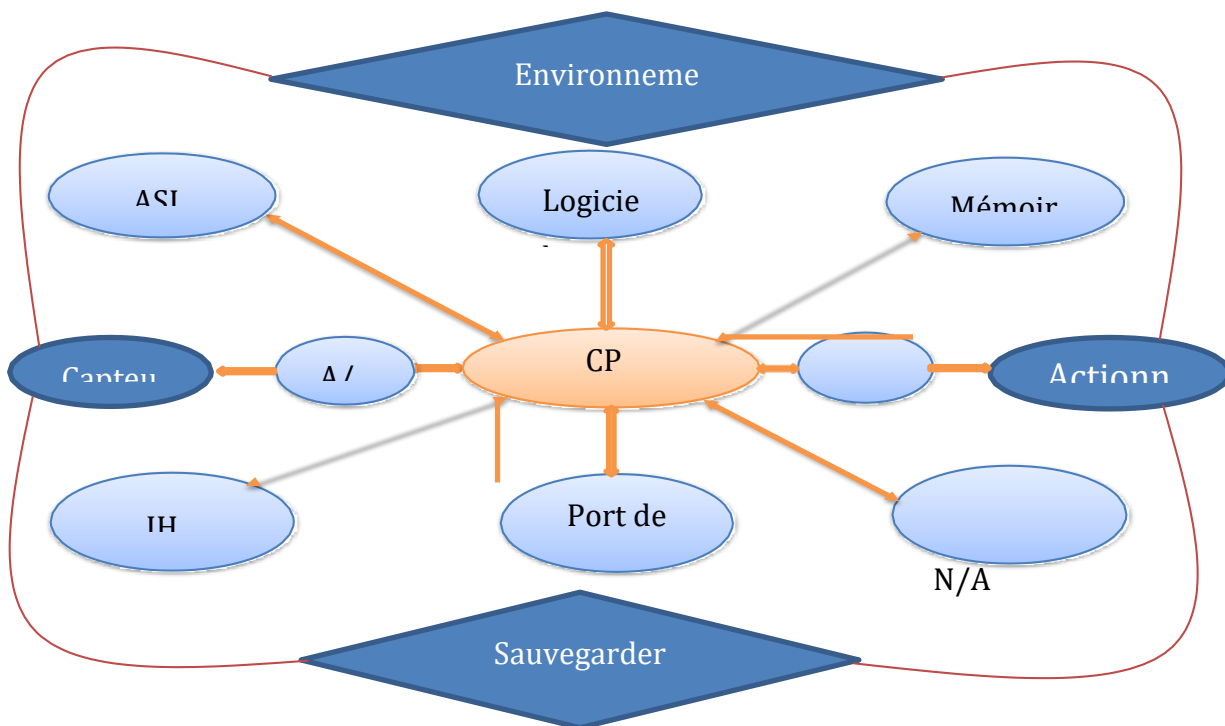


Figure 3 : Schéma synoptique d'un SE

Un système embarqué typique comme illustré par la figure 3, contient en entrée des capteurs généralement analogiques couplés à des convertisseurs Analogiques/Numériques(A/N). On trouve en sortie des actionneurs généralement analogiques couplés à des convertisseurs Numériques/Analogiques (N/A). Au milieu, on trouve le calculateur mettant en œuvre un processeur embarqué et ses périphériques d'Entrées/Sorties (E/S). Il est à noter qu'il est complété généralement d'un circuit FPGA (Field Programmable Gate Array) jouant le rôle de coprocesseur (est un circuit électronique destiné à ajouter une fonction à un processeur classique) afin de proposer des accélérations matérielles au processeur. Sur ce schéma théorique se greffe un paramètre important, c'est le rôle de l'environnement extérieur, l'environnement dans lequel opère le système embarqué n'est pas contrôlé ou contrôlable. Cela suppose donc dépendre en compte ce paramètre lors de conception. On doit par exemple prendre en compte les évolutions des caractéristiques électriques des composants en fonction de la température, des radiations.

- CAN : Convertisseur Analogique Numérique ;
- CPU : Processeur ;
- CNA : Convertisseur Numérique Analogique ;
- IHM : Interface Homme Machine ;
- ASIC : Application Spécifique Intégrée, il s'agit d'un circuit, qui intègre sur une même puce l'ensemble des éléments actifs indispensables pour qu'une fonction ou qu'un ensemble électronique puisse se réaliser.

6. Caractéristiques principales d'un système embarqué

Les caractéristiques principales d'un système embarqué sont :

- ✓ **Autonome** : une fois enfouis dans l'application ne sont plus accessibles ;
- ✓ **Temps-réel** : les temps de réponses de ces systèmes sont aussi importants que l'exactitude des résultats ;
- ✓ **Réactif** : il doit réagir à l'arrivée d'information extérieure non prévue.

7. Domaines d'application d'un système embarqué

L'électronique embarquée est de plus en plus utilisée dans notre quotidien. En plus des ordinateurs et micro-ordinateurs qui sont les systèmes embarqués les plus connus, l'électronique embarquée est utilisée sur de nombreux objets comme téléphone, les agendas électroniques et les

voitures. L'objectif de l'utilisation des systèmes embarqués est de donner aux objets usuels la possibilité de réagir à l'environnement l'électronique embarquée est introduite dans divers domaines à savoir :

- Le domaine de grand public : Smart phone, console de jeux, appareil photo, lecture audio ;
- Les moyens de transport : Ordinateur de bord, GPS, système de navigation, avion ;
- Les équipements médicaux : Imagerie (rayons x, ultra-sons, IRM, endoscopie, caméra, monitoring, perfusion, lasers ...) ;
- Les équipements de télécommunication : Station mobile, routeur, Gateway, satellite ;
- Les équipements industriels : Production automatisées, système de commande d'énergie, équipement de stockage ;

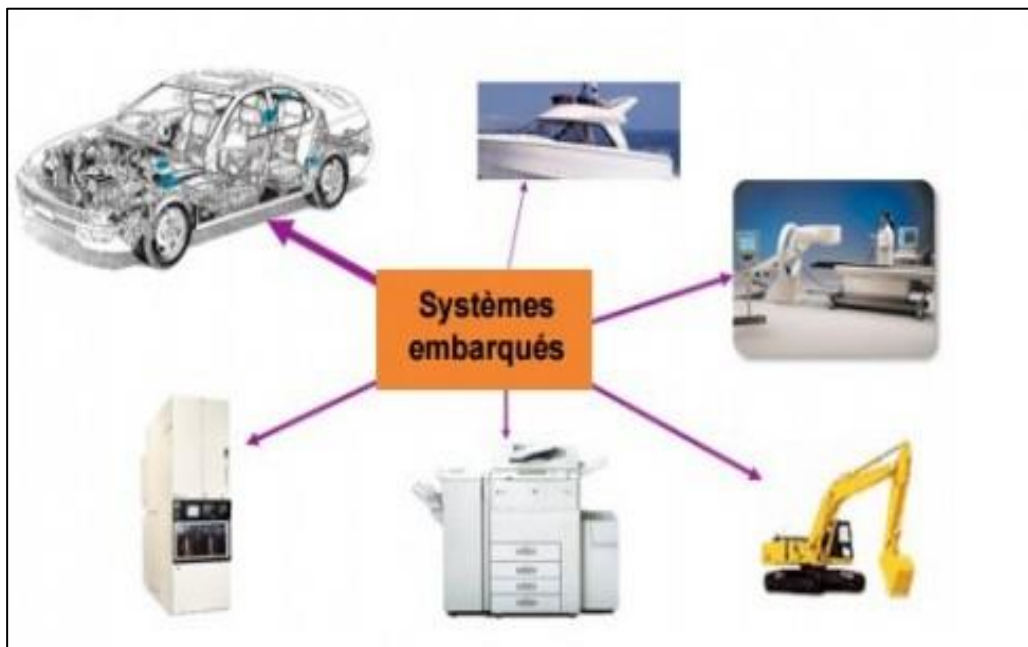


Figure 4 : Les domaines d'application des systèmes embarqués

8. Fiabilité

Les systèmes embarqués sont la plupart du temps dans des machines qui doivent fonctionner en continu pendant de nombreuses années, sans erreurs et, dans certains cas, réparer eux-mêmes les erreurs quand elles arrivent. C'est pourquoi les logiciels sont toujours développés et testés avec plus d'attention que ceux pour les PC. Les pièces mobiles non fiables (par exemple les lecteurs de disques, boutons ou commutateurs) sont proscrites.

La question de la fiabilité peut inclure :

- Le système ne peut pas être éteint pour des réparations ou ce sont des réparations

inaccessibles ;

- La solution peut être des pièces détachées supplémentaires ou un "mode mou" du logiciel qui fournit un fonctionnement partiel. Par exemple : les câbles sous-marins, les balises de navigation, les puits de forage, ...etc. ;
-
- Le système doit rester en marche pour des raisons de sécurité. Souvent, les sauvegardes sont effectuées par un opérateur. Dans ce cas, le « mode mou » est toléré. Par exemple : les systèmes de contrôle des réacteurs, les usines chimiques, ...etc. ;
- Un arrêt du système peut provoquer des pertes monétaires énormes s'il s'éteint. Comme exemple : les systèmes de ponts ou d'ascenseurs, les transferts de fonds.

9. Historique

Dans l'un des premiers systèmes modernes embarqués reconnaissables a été l'Apollo Guidance Computer en 1967, le système de guidage de la mission lunaire Apollo, développé par Charles Stark Draper du Massachusetts Institute of Technology. Chaque mission lunaire était équipée de deux systèmes D'Apollo Guidance Computer (AGC). Au commencement du projet, l'ordinateur AGC d'Apollo était considéré comme l'élément le moins fiable du projet. Cependant grâce à l'utilisation de nouveaux composants qu'étaient à l'époque les circuits intégrés, des gains substantiels sur la place utile et la charge utile ont été réalisés, avec une diminution supposée des risques déjà nombreux des missions.

10. Contraintes

Les contraintes de développement sur un système embarqué sont nombreuses et se rapprochent assez de celles rencontrées en informatique en général. La complexité des algorithmes utilisés doit être relativement faible d'autant plus que les performances et la puissance des systèmes est moindre, l'absence de MMU limite la taille des applications. L'absence de media de stockage permanente, aussi de nombreuses applications des systèmes embarqués sont des applications temps-réel ; on doit souvent ajouter une contrainte temps-réel.

10.1. Complexité des algorithmes

Du fait d'une puissance de calcul et d'une mémoire limitée, on préfère des algorithmes peu complexes sur un système embarqué. Par exemple, l'utilisation de la récursivité (Une fonction récursive est une fonction qui s'appelle elle-même. Chaque appel à la fonction est indépendant des autres, avec ses propres variables.) sur des systèmes à faible performances est peu recommandée.

10.2. Absence de MMU

L'absence de Memory Management Unit (MMU) rend impossible la gestion de la mémoire virtuelle. Les processus se retrouvent donc très vite limité en mémoire car ils doivent tous partager le même espace mémoire, y compris la pile.

10.3. Absence de medium de stockage permanent

Les systèmes embarqués sont généralement dépourvus d'unité de stockage telle qu'un disque dur ou une bande magnétique.

La sauvegarde des données est généralement faite en ROM ou en mémoire FLASH.

L'utilisation de ces mémoires est relativement onéreuse et peu fiable.

11. L'internet des objets (IoT)

Pour définir le concept de l'Internet of Thing (IoT) évoqué tout au long de notre rapport, les différentes définitions de l'IoT existant dans la littérature pour ensuite mettre l'accent sur son fonctionnement sont présentées comme suit :

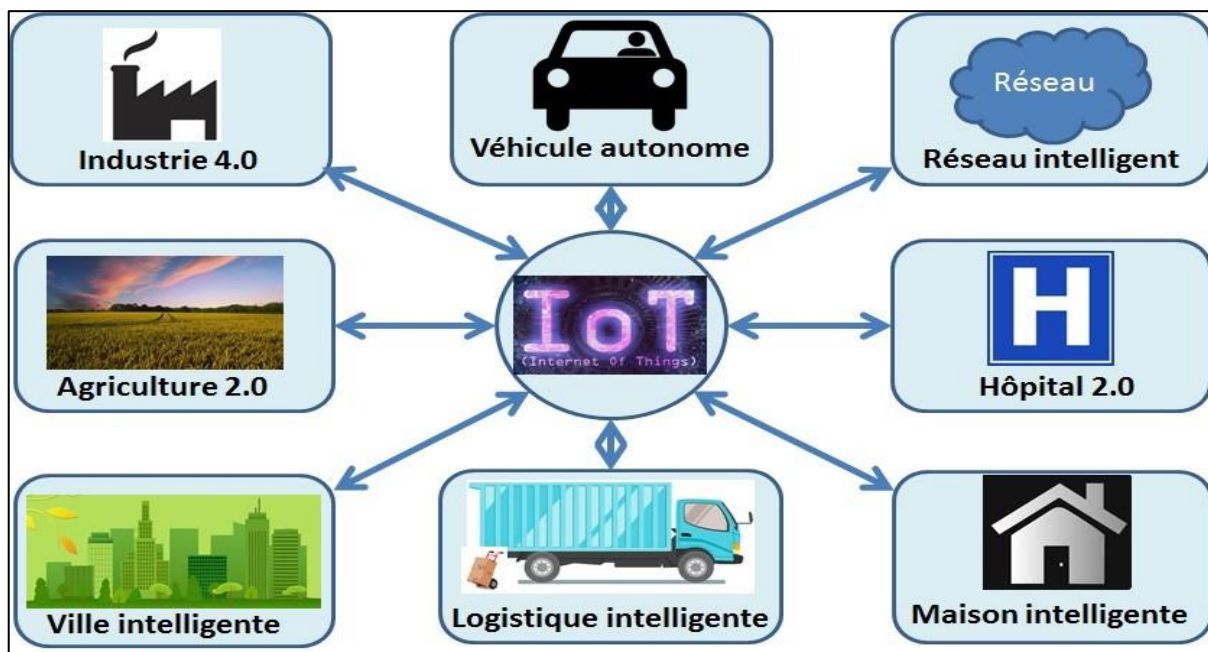


Figure 5 : Concept d'IoT

12. Notion de base sur l'IoT

Selon l'UIT (union internationale des télécommunication), l'internet des objets est défini comme une infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets (physique ou virtuels) grâce aux technologies de

l'information et de la communication interopérables existantes ou en évolution.

L'Internet des objets, ou IoT (Internet of Thing), est un scénario dans lequel les objets, les animaux et les personnes se voient attribuer des identifiants uniques, ainsi que la capacité de transférer des données sur un réseau sans nécessiter aucune interaction humain-à-humain ou humain-à-machine.

On peut définir l'IoT comme l'acronyme de l'internet of thing (internet des objets). Le terme IoT est apparu la première fois en 1999 dans un discours de Kevin ASHTON, un ingénieur britannique. Il servait à désigner un système où les objets physiques sont connectés à internet. Il s'agit également d'un système capable de créer et transmettre des données afin de créer de la valeur pour ses utilisateurs à travers divers services (agrégation, analytique...).



Figure 6 : L'interconnexion d'IoT avec différents objets

13. Définition d'IoT

L'internet des objets est l'interconnexion entre l'internet et des objets, des liens des environnements physiques. L'internet des objets est en partie responsable d'un accroissement exponentiel du volume de données générées sur le réseau, à l'origine du big data.

14. Protocoles applicatifs utilisés à l'IoT

Un protocole applicatif est un ensemble des règles définissant le mode de communication entre deux applications informatiques. Ils se basent sur les protocoles de transport (TCP/UDP), pour établir dans un premier temps des routes et échanger les données selon l'ensemble des règles du protocole applicatif sélectionné.

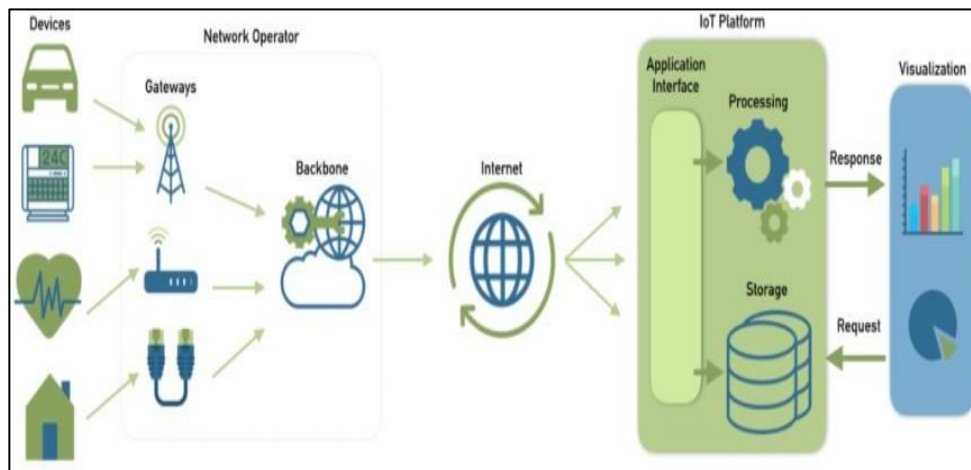


Figure 7 : Le protocole utilisé dans l'IoT

Pour la suite, nous avons identifié plusieurs familles, et pour chacune d'elle nous avons sélectionné les protocoles applicatifs les plus utilisés.

✚ Protocoles de messagerie : MQTT, XMPP, et AMQ ; ✚ Protocoles de transfert web : CoAP, API REST ;

✚ Protocole réseaux : Web socket.

14.1. Les composants d'un IoT

Une solution d'IoT s'articule autour de 5 composants essentiels que sont :

- ✚ Les objets (capteurs) ;
- ✚ Le réseaux (connectivité) ;
- ✚ Les données ;
- ✚ Les informations ;
- ✚ Les applications d'exploitation.



Figure 8 : Les composants utilisés dans l'IoT

14.2. Architecture d'un IoT

L'architecture d'un système IoT est composée de plusieurs niveaux qui communiquent entre eux pour relier le monde tangible des objets au monde virtuel des réseaux et du cloud.

Tous les projets n'adoptent pas une architecture formellement identique, néanmoins il est possible de schématiser le parcours de la donnée.

Il existe plusieurs formes d'architecture d'un système IoT (internet des objets), il n'y a pas une définition formellement identique d'une architecture d'un système IoT adopté pour tous les projets. Dans le présent projet, on a adopté celle présentée dans illustré par la figure 9.

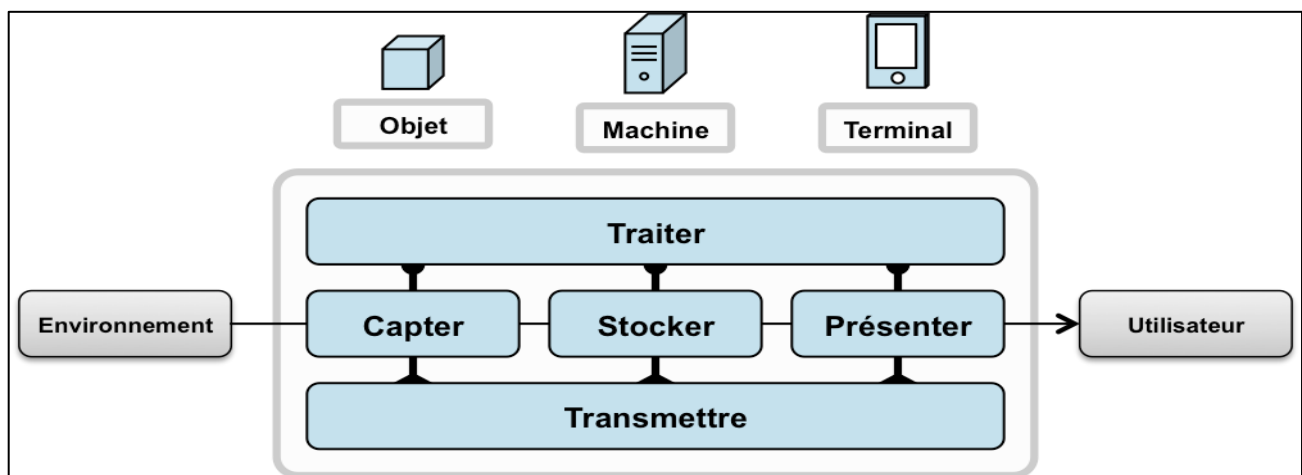


Figure 9 : Architecture de l'IoT

14.3. Domaines d'utilisation de l'IoT

Plusieurs domaines d'application sont touchés par l'IoT, parmi ces principaux domaines nous citons :

- ✚ Le domaine de sécurité ; ✚ Le domaine de transport ;
- ✚ L'environnement et l'infrastructure et service public ; ✚ Le domaine d'agriculture.

14.4. IoT dans le milieu agricole

Le monde de l'agriculture n'échappe pas aux évolutions de l'internet des objets. En effet, il n'est pas d'utiliser des technologies robotiques ou d'intelligence artificielles pour assister les agriculteurs, on trouve, par exemple des robots qui peuvent venir en aide aux agriculteurs dans la réalisation de tâches répétitives comme le désherbage, le transport de cageotage ou de fourniture. L'utilisation d'un IoT dans le milieu agricole s'explique selon la figure 10.

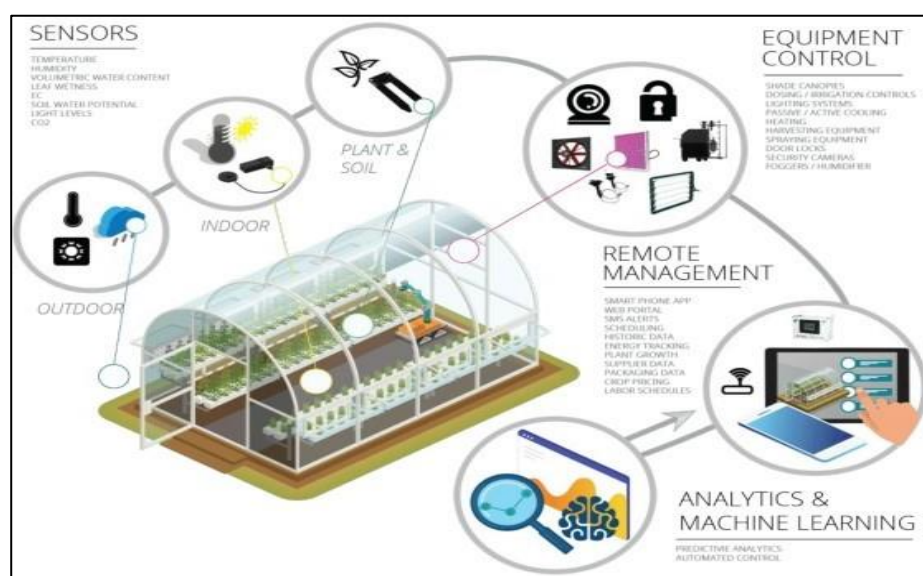


Figure 10 : L'utilisation d'un IoT dans le domaine agricole

15. Cadre du projet

15.1. Présentation de la CPG

La Compagnie des Phosphates de Gafsa (CPG) est une entreprise publique anonyme à caractère industriel et commercial, qui a pour objet l'exploitation des gisements de phosphate en Tunisie. C'est une entreprise publique ayant la forme de société anonyme et placée sous la tutelle du ministère de l'Industrie, de l'Énergie et des Mines. Son capital social s'élève à 268 MDT et il est détenu à 99,99% par l'État Tunisien.

La CPG fait partie du groupe CPG-GCT (Groupe Chimique Tunisien). Le groupe a été créé en 1994 et résulte de la fusion de la CPG et de 5 sociétés de transformation des phosphates à savoir la Société Industrielle d'Acide Phosphorique et d'Engrais à Sfax, les Industries Chimiques Maghrébines à Gabès, la Société Arabe des Engrais Phosphatés et Azotés à Gabès, les Engrais de Gabès et l'Industrie Chimique de Gafsa. L'objet social de la CPG est l'exploitation des gisements de phosphate en Tunisie, et de plus précisément :

- l'exploitation des réserves de phosphate de la Tunisie ;
- l'enrichissement du minerai extrait pour obtenir une qualité commercialisable ;
- la commercialisation du phosphate produit ;
- les prospections et recherches géologiques.


15.2. Contexte du projet

En matière de respect de l'environnement, CPG donne une attention particulière en menant des actions continues visant à réduire la pollution industrielle et développement d'espaces verts dans la région minière.

Les principaux projets environnementaux sont :

CONSTRUCTION DE DIGUES POUR LES EFFLUENTS LIQUIDES ACCUMULATION

Ce projet consiste à construire 6 digues dans le bassin minier ayant différentes superficies (30-300 hectares (à stocker les effluents liquides provenant des stations de lavage et arrêter les détournant dans les oueds voisins.)

 Premier pas :

- Pour construire les digues des usines de Mdhilla 3, Kef Eddour et Metlaoui qui ont démarré en exploitation respectivement en août 2005- septembre 2006 et novembre 2006. Il a permis jusqu'à présent, l'accumulation de 7 millions m³ par an d'effluents liquides correspondant à 63% du total quantité (11 millions de m³)

 Deuxième étape :

- Construire les digues à Redeyef, Moulares, Mdhilla-1 et Mdhilla-2 végétaux.
- 5 digues sont en cours d'exploitation accumulant près de 80 % des effluents liquides et environ 1,5 million de m³ de l'industrie l'eau est recyclée.

PLANTATION D'ARBRES

En ce qui concerne la protection de l'environnement dans la région minière, la CPG a poursuivi l'opération de plantation d'arbres qui a débuté en 1996 à la suite de l'accord de partenariat conclu avec les autorités régionales en charge de la protection de l'environnement.

16. Spécification des besoins

L'étude de l'existant a permis de mettre en évidence les points positifs et les points de dysfonctionnement du système étudié. Il s'agira dans les paragraphes suivants de livrer les spécifications des besoins afin de répertorier les contraintes à prendre en compte dans la conception de la solution.

16.1. Les besoins fonctionnels

Les besoins fonctionnels expriment une action que doit effectuer l'application en réponse à une demande (sorties qui sont produites pour un ensemble donné d'entrées).

Notre objectif est la réalisation d'un système d'irrigation intelligent contrôlé par des capteurs pour afficher les valeurs de la température, humidité et le taux de la lumière. Ce système permet de contrôler intelligemment un ensemble des plantes des zones vertes appartenant au nouveau projet de fertilisation de la CPG en vue de garantir une bonne performance. Pour des raisons pratiques nous allons faire une relation entre le contrôle automatique de ce système d'irrigation et le contrôle semi-automatique.

Le contrôle automatique permet au système de réagir d'une manière intelligente pour établir les conditions climatologiques idéales pour les plantes comme le contrôle de température humidité, ce type de contrôle est basé sur la programmation de la carte Raspberry pour interagir avec l'interface de Node Red.

Le contrôle semi-automatique est le mode de contrôle alternatif où l'utilisateur peut se renseigner sur les conditions climatologiques intérieures des serres via une interface graphique appropriée et peut intervenir sur le processus de contrôle.

16.2. Les besoins non fonctionnels

Les besoins non fonctionnels sont importants car ils agissent de façon indirecte sur le résultat et sur le rendement de l'utilisateur. Notre système doit répondre à ces besoins qui sont nécessaires pour atteindre la perfection et la bonne qualité du logiciel.

- ✓ **Fiabilité** : l'application doit fonctionner de façon cohérente sans erreurs,
- ✓ **Efficacité** : l'application doit permettre l'accomplissement des tâches avec le minimum de manipulations,
- ✓ **Sécurité** : l'application doit être sécurisée au niveau des données : authentification et contrôle d'accès,

- ✓ **Performance** : l'application doit être performante c'est-à-dire qu'elle doit répondre à travers ses fonctionnalités à toutes les exigences des utilisateurs d'une manière optimale.

17. Conclusion

L'internet des objets est un marché qui devient mature et qui offre de belles opportunités pour ses différents acteurs. L'internet des objets permet le développement d'un grand nombre d'application d'autant d'intelligence. Un certain nombre des domaines santé, maison, agriculture, ...etc.

Le chapitre suivant consiste à étudier l'approche de la serre intelligent via une carte Raspberry. Ce chapitre est composé essentiellement par deux parties principales, à savoir, la présentation du matériel utilisés dans ce projet ainsi que leur câblage entre et la partie programmation des cartes d'acquisition (Raspberry PI, Carte Arduino, Carte Wemos).

Chapitre 2 : Approche d'une serre intelligente via une carte RaspBerry

1. Introduction

La serre intelligent est une serre basée sur la technologie de l'internet des objets (IOT), qui permet de collecter les informations en temps réel. La serre intelligent permet aussi de faciliter la gestion des serres sur des grandes exploitations en capitalisant sur l'historique des données (météo, développement de culture...), afin d'anticiper les actions de gestion pour une meilleure qualité de production.

L'agriculture intelligente est une révolution de l'agriculture classique qui implique la réorientation des systèmes agricoles afin de soutenir efficacement le développement alimentaire. Le principal objectif de l'agriculture intelligente est d'accroître la productivité des revenus agricoles.

L'agriculture intelligente implique l'utilisation des technologies de la communication de l'information et en particulier de l'internet des objets.

2. Définition d'une serre intelligente

La serre intelligente est un abri exploitant le rayonnement solaire destiné à la culture et à la protection des plantes connecté aux réseaux d'internet des objets. La serre intelligente est une méthode de faciliter la production et le contrôle des plantes agricoles. En effet, la serre intelligente est une révolution technologique dans le monde moderne. Ainsi la serre intelligente est une solution pour augmenter la production dans les grandes cultures car elle favorise le contrôle de l'augmentation de température humidité à l'aide d'un capteur DHT11, et le contrôle de la luminosité ultra-violet par une photorésistance, ainsi que l'ouverture des fenêtres selon les conditions de taux de luminosité nécessaire pour la serre.

La serre intelligente favorise le contrôle d'irrigation, température, humidité et l'éclairage dans la serre.

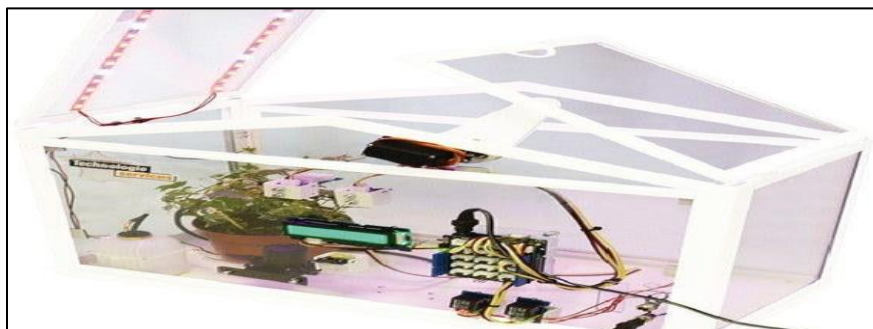


Figure 11 : Prototype d'une serre intelligente

3. Les types des serres

La classification des serres est compliquée et difficile. Il est généralement réalisé selon la forme donnée par le cadre de support du composant. Il existe deux types principaux. Elle appartient à deux grandes familles de serre : la serre tunnel et la serre multi chapelles.

3.1. La serre tunnel

Une serre tunnel est composée de plusieurs grandes arches métalliques, recouvertes d'un film plastique transparent flexible, qui lui donne la forme d'un tunnel.



Figure 12 : Serre tunnel

3.2. Serre chapelle

La serre Multi chapelle se compose d'allées interconnectées avec un toit elliptique et des parois latérales verticales. C'est une serre très polyvalente qui s'adapte parfaitement à tous types de climats et de cultures et permet d'adapter la largeur de chaque chapelle au cadre de plantation.



Figure 13 : Serre multi chapelle

3.3. Les besoins de la plante sous la serre

Les plantes ont besoin d'humidité, de chaleur et de lumière pour pousser. La serre stabilise l'environnement de culture en régulant la température dans la pièce et en protégeant les plantes du froid extrême.

4. Les besoins en lumière des plantes

Pour les plantes, la lumière est le moteur de la photosynthèse. Il est important de respecter leurs besoins d'éclairage. La photosynthèse est en fait le processus qui permet aux plantes de synthétiser leur matière organique à partir d'éléments minéraux. L'apport énergétique sous forme de lumière permet à la plante d'utiliser le dioxyde de carbone présent dans l'air pour récupérer le carbone et l'intégrer à des molécules organiques comme les sucres (fructose, amidon, cellulose) ou les protéines. Dans les réactions chimiques de la photosynthèse, de l'oxygène (O_2) se forme.). La figure. 14 montre le principe de la photosynthèse.



Figure 14 : Principe de la Photosynthèse

Globalement, par rapport aux besoins en lumière, il existe trois catégories de plantes : les plantes d'ombre, les plantes de lumière, et les plantes qui supportent un ensoleillement mitigé. Les plantes scaphites, sont adaptées à des conditions de luminosité faible (comme les fougères, les clivais, etc.), alors que d'autres ont au contraire besoin de beaucoup de lumière pour prospérer : il s'agit des plantes héliophiles.

4.1. Les besoins d'humidité, température

Les plantes tentent d'atteindre la température optimale et à cette température, l'équilibre entre la température de l'air, l'humidité relative et la lumière joue un rôle important. En présence de beaucoup de lumière, les plantes deviennent chaudes, ce qui crée une différence de température entre les plantes et l'air.

L'objectif, pour le jardinier, est de maintenir une température au-dessous de **35°C**. Le premier réflexe à avoir est d'ouvrir toutes les ouvertures (porte et fenêtres), voire de relever les côtés de votre serre tunnel, si ceux-ci ne sont pas enterrés, pour faire circuler l'air.

L'humidité de l'air est un des facteurs importants très difficile à contrôler. En général, les plants

cultivés en serre exigent une **humidité entre 40% et 75%** pendant leur phase de croissance.

4.2. Les besoins en eau des plantes

Le sol fournit à la plante l'eau et les éléments minéraux nécessaires à sa croissance et à son développement. L'eau puisée dans le sol, pénètre par les racines et transite dans les vaisseaux de la plante vers les feuilles. L'eau est le constituant majeur des plantes. Cependant la plus grande partie de cette eau est transpirée par les feuilles, sous forme de vapeur d'eau au moyen de multiples petits orifices (les stomates). C'est la "transpiration".

Dans le même temps, le sol, sous l'effet du rayonnement solaire et du vent laisse aussi échapper de l'eau vers l'atmosphère sous forme de vapeur d'eau ; ce phénomène est appelé "évaporation".

La transpiration de la plante et l'évaporation de l'eau du sol se déroulent en permanence simultanément. L'addition de ces deux phénomènes qui épuise progressivement la réserve d'eau du sol est dénommée "évapotranspiration".

Les arrosages doivent être suffisants pour permettre de refaire le plein du réservoir sol. Vérifiez qu'ils ont été quantitativement suffisants, en contrôlant l'humidité sous la serre.

5. Performance d'une serre intelligente

La serre intelligent favorise la possibilité de contrôler les données agriculture dans la serre, pour faciliter l'augmentation de production de culture, par la nouvelle technologie nous parlons les modernisations dans le domaine agriculture, comme le contrôle à distance avec un appareil téléphonique. La serre intelligent donne la possibilité de l'irrigation automatique, le contrôle d'éclairage, le contrôle de température et l'humidité dans la serre.

5.1. Contrôle de température humidité

Le contrôle de la température est effectué par des sondes de température à l'intérieure de la serre. À partir de la mesure de la température un certain nombre d'actionneur en fonction du programme installer dans la carte Arduino.

Le contrôle de température humidité se fait par le capteur DHT11, ce capteur est responsable des conditions météorologiques pour l'irrigation automatique.

5.2. Contrôle de l'éclairage

L'éclairage est un mécanisme qui est contrôler par la photorésistance installée dans le circuit intégré à l'intérieur de la serre pour réduire le rayonnement incident sur la culture quand la luminosité est trop élevée, les fenêtr se ferme pour l'ombrage de la serre.

5.3. Contrôle du taux de gaz

Le contrôle de taux de gaz se fait par le capteur MQ5 pour détecter les gaz dans la serre, pour

réduire les risques d'endommager la culture maraîchère en raison de certains gaz à effet serre.

6. Matériels utilisés

La liste de matériels utiliser dans ce projets (serre intelligent) est la suivante :

- ✓ Carte Arduino ;
- ✓ Carte Wemos ;
- ✓ Carte Raspberry PI ;
- ✓ Capteur température, humidité DHT11 ;
- ✓ Capteur de température LM35 ;
- ✓ Capteur de gaz MQ5 ;
- ✓ Capteur de pluie ;
- ✓ Photorésistance ;
- ✓ Moteur dc 5 v ;
- ✓ Module RTC (real time clock);
- ✓ Pompe d'eau ;
- ✓ Écran LCD 16*4 ;
- ✓ LEDs.

6.1. Carte Arduino

L'Arduino est un module est généralement construit autour d'un microcontrôleur atmel AVR et de composant complémentaire qui facilitent la programmation et l'interfaçage avec d'autre circuit.



Figure 15 : Catre Arduino.

6.2. Carte Wemos

Une carte Wemos D1 est une carte électronique proche de la carte UNO. Elle intègre un module WIFI ESP8266-12 en natif, une mémoire SPIFFS (comparable à une carte SD de 3MO et une puissance de calcul supérieure.

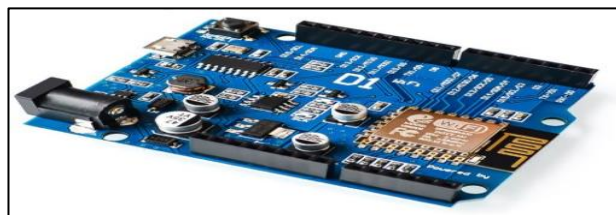


Figure 16 : Catre Wemos

6.3. Carte Raspberry PI

Une carte Raspberry pi 4+ est un nano ordinateur mono carte à processeur ARM de la taille d'une carte de crédit conçu par des professeurs du département de l'université de Cambridge dans le cadre de la fonction Raspberry pi 4+.



Figure 17 : Carte Raspberry pi

6.4. Capteur DHT11

Un capteur DHT11 permet de mesurer des températures de 0 à +50°C avec une précision de +/-2°C et des taux d'humidité relative de 20% à 80% avec une précision de +/- 5%. Ces caractéristiques sont données ci-dessous :

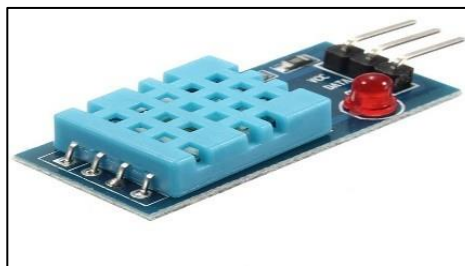


Figure 18 : Capteur DHT11

- Tension de fonctionnement : 3.5V à 5.5V ;
- Courant de fonctionnement : 0.3mA (mesure) 60uA (veille) ;
- Sortie : données en série ;
- Plage de température : 0 ° C à 50 ° C ;
- Gamme d'humidité : 20% à 90% ;
- Précision : ± 1 ° C et $\pm 1\%$.

Le capteur DHT11 est étalonné en usine et délivre des données série. Il est facile à configurer. Il y a aussi des équivalents au capteur DHT11 comme : DHT22, AM2302, SHT71.

6.5. Capteur de gaz

Le capteur de gaz est un appareil pour surveiller et mesurer le pourcentage atmosphérique piloté par un microcontrôleur. Les gaz détectés par ce capteur sont :

- ✓ Co ;
- ✓ H₂S ;
- ✓ O₂ ;
- ✓ SO₂ ;
- ✓ Gaz combustible.

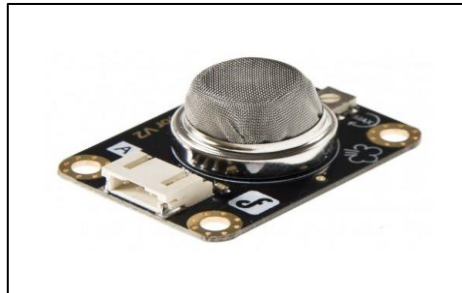


Figure 19 : Capteur MQ2

6.6. Système de ventilation

En plus de l'isolation, de l'arrosage ou encore du chauffage, la ventilation constitue une étape essentielle pour garantir le bon développement des plantes à l'intérieur de la serre.



Figure 20 : Ventilateur

6.7. Capteur LM35

Le capteur de température LM35 est un capteur analogique de température fabriqué par Texas instruments. La sortie analogique du capteur est proportionnelle à la température.

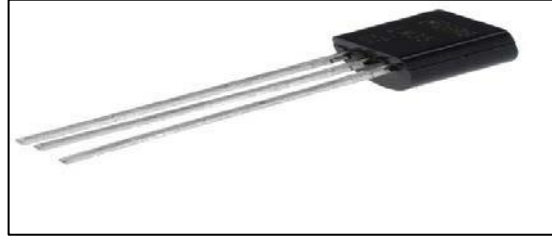


Figure 21 : Capteur LM35

6.8. Détecteur de lumière LDR

Une photorésistance (LDR= Light Dépendant Résistor) est composée d'un semi-conducteur à haut résistivité, deux électrodes sont séparées par ce matériau photoconducteur.

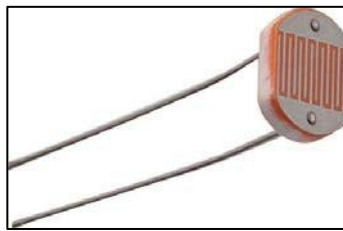


Figure 22 : Détecteur de lumière LDR

6.9. Pompe d'eau

Dans ce projet, le système d'irrigation est basé sur une pompe d'arrosage, c'est l'organe qui fournit au sol l'eau dont les plantes ont besoin pour garder l'humidité du sol au niveau de la consigne. Pour ce prototype, l'électrovanne utilisée sur la figure. 23 est alimentée par une source de tension 5V.



Figure 23 : Électrovanne d'eau

7. Contrôle du système à distance

La contrôle à distance de la serre intelligent se fait à travers une interface graphique développée avec le programme du Node-Red installé après la configuration de la carte Raspberry.

Node-Red est un programme qui favorise le contrôle à distance de l'irrigation automatique, l'éclairage ou la luminosité dans la serre et le contrôle de gaz.

Le contrôle à distance est une solution qui permet d'afficher les taux de la lumière, température, humidité et taux de gaz, pour contrôler le système de la serre selon les conditions programmer, ce qui permet aux agriculteurs d'augmenter la production culture.

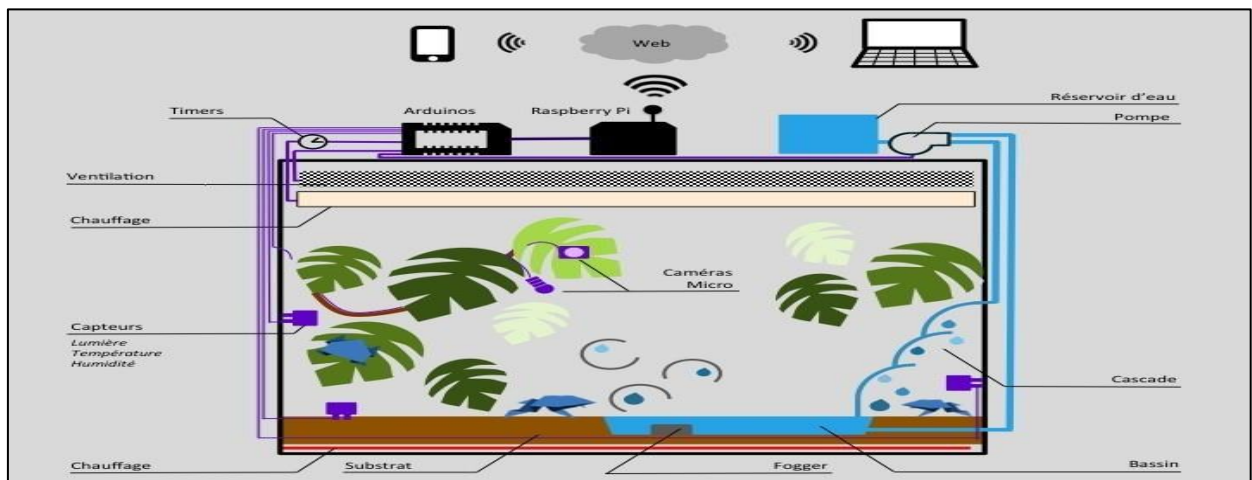


Figure 24 : Le contrôle à distance

8. Câblage de la carte Wemos

Le schéma de câblage de la carte Wemos avec le capteur de gaz et le DHT11 est illustré par la figure ci-dessous :

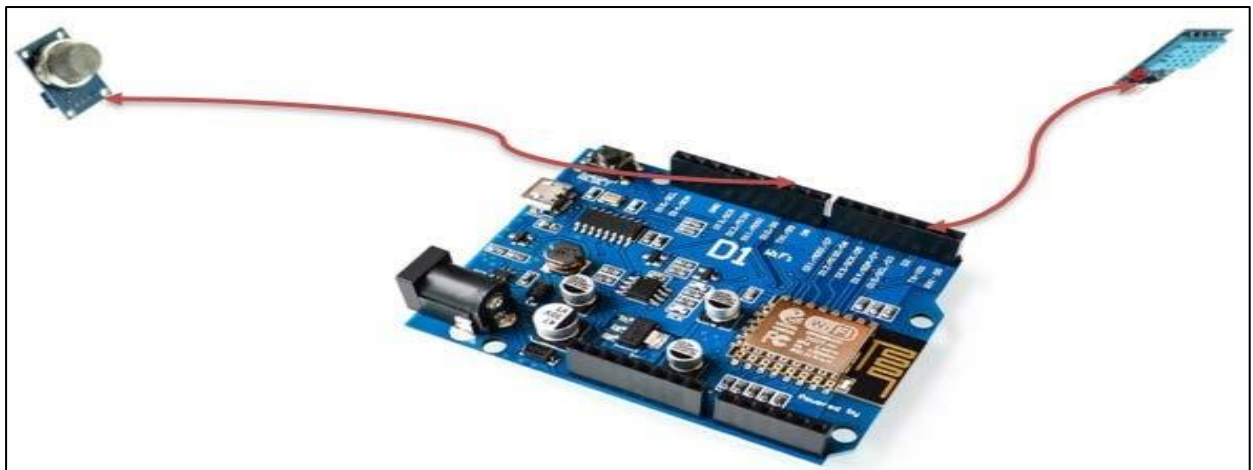


Figure 25 : Carte Wemos connecter avec les capteurs

9. Câblage de la carte Arduino

Le schéma de câblage de la carte Arduino avec le capteur de température, le capteur de pluie et la photorésistance et le DHT11 est présenté par la figure 26.

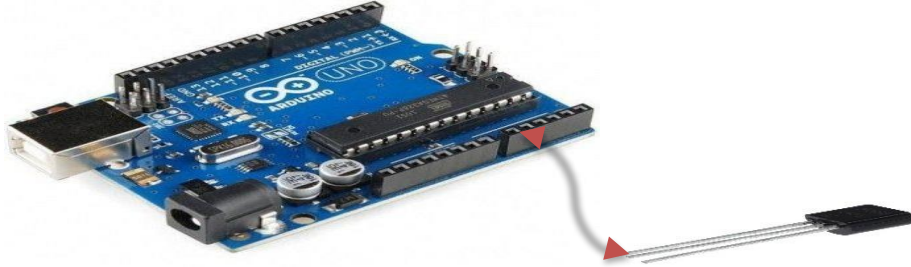


Figure 26 : Connexion du capteur de température LM35 avec la carte Arduino

10. Programmes utilisés

Pour les programmeurs confirmés, le langage C/C++ qui est traditionnellement utilisé pour programmer les microcontrôleurs peut être la solution la plus performante pour programmer une carte Arduino.

10.1. Programme pour la carte Wemos

Les bibliothèques utilisées pour la programmation de la carte Wemos sont :

- `#include "DHT.h"`

🔧 C'est la bibliothèque de la capteur DHT11.

- `#include <ESP8266WiFi.h>`

🔧 C'est la bibliothèque de la carte wifi (Wemos).

- `#include <Ticker.h>`

🔧 C'est une bibliothèque pour créer des trickers qui peuvent appeler des fonctions répétitives. Remplace Delay () par des fonctions non bloquantes.

- `#include <AsyncMqttClient.h>`

🔧 C'est une bibliothèque pour utiliser la MOSQUITTO (MQTT Broker)

Les images qui suivent présentent la partie du codage de la carte Wemos.


```

#include "DHT.h"
#include <ESP8266WiFi.h>
#include <Ticker.h>
#include <AsyncMqttClient.h>
#define WIFI_SSID "TOPNET_E788"
#define WIFI_PASSWORD "Emna2007"
// Raspberri Pi Mosquitto MQTT Broker
#define MQTT_HOST IPAddress(192, 168, 43, 119)
// For a cloud MQTT broker, type the domain name
// #define MQTT_HOST "example.com"
#define MQTT_PORT 1883
// Temperature MQTT Topics
#define MQTT_PUB_TEMP "esp/dht/temperature"
#define MQTT_PUB_HUM "esp/dht/humidity"
#define MQTT_PUB_GAZ "esp/dht/gaz"
// Digital pin connected to the DHT sensor
#define DHTPIN 14
#define MQSPIN A0
// Uncomment whatever DHT sensor type you're using
#define DHTTYPE DHT11 // DHT 11
// #define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
// #define DHTTYPE DHT21 // DHT 21 (AM2301)
// Initialize DHT sensor
DHT dht(DHTPIN, DHTTYPE);
// Variables to hold sensor readings
float temp;
float hum;
float gaz;
AsyncMqttClient mqttClient;
Ticker mqttReconnectTimer;

```

Figure 27 : Première partie du programme 'carte Wemos'

```

WiFiEventHandler wifiConnectHandler;
WiFiEventHandler wifiDisconnectHandler;
Ticker wifiReconnectTimer;
unsigned long previousMillis = 0; // Stores last time temperature was published
const long interval = 10000; // Interval at which to publish sensor readings
void connectToWifi() {
    Serial.println("Connecting to Wi-Fi...");
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
}
void onWifiConnect(const WiFiEventStationModeGotIP& event) {
    Serial.println("Connected to Wi-Fi.");
    connectToMqtt();
}
void onWifiDisconnect(const WiFiEventStationModeDisconnected& event) {
    Serial.println("Disconnected from Wi-Fi.");
    mqttReconnectTimer.detach(); // ensure we don't reconnect to MQTT while reconnecting to Wi-Fi
    wifiReconnectTimer.once(2, connectToWifi);
}
void connectToMqtt() {
    Serial.println("Connecting to MQTT...");
    mqttClient.connect();
}
void onMqttConnect(bool sessionPresent) {
    Serial.println("Connected to MQTT.");
    Serial.print("Session present: ");
    Serial.println(sessionPresent);
}
void onMqttDisconnect(AsyncMqttClientDisconnectReason reason) {
    Serial.println("Disconnected from MQTT.");
}

```

Figure 28 : Deuxième partie du programme 'carte Wemos'.


```

    if (WiFi.isConnected()) {
        mqttReconnectTimer.once(2, connectToMqtt);
    }
}

void onMqttPublish(uint16_t packetId) {
    Serial.print("Publish acknowledged.");
    Serial.print(" packetId: ");
    Serial.println(packetId);
}

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

    dht.begin();

    wifiConnectHandler = WiFi.onStationModeGotIP(onWifiConnect);
    wifiDisconnectHandler = WiFi.onStationModeDisconnected(onWifiDisconnect);

    mqttClient.onConnect(onMqttConnect);
    mqttClient.onDisconnect(onMqttDisconnect);
    //mqttClient.onSubscribe(onMqttSubscribe);
    //mqttClient.onUnsubscribe(onMqttUnsubscribe);
    mqttClient.onPublish(onMqttPublish);
    mqttClient.setServer(MQTT_HOST, MQTT_PORT);
    // If your broker requires authentication (username and password), set them below
    //mqttClient.setCredentials("REPLACE_WITH_YOUR_USER", "REPLACE_WITH_YOUR_PASSWORD");

    connectToWifi();
}

```

Figure 29 : Troisième partie du programme 'carte Wemos'.

```

void loop() {
    float sensorValue;
    float sensorVoltage;
    unsigned long currentMillis = millis();
    // Every X number of seconds (interval = 10 seconds)
    // it publishes a new MQTT message
    if (currentMillis - previousMillis >= interval) {
        // Save the last time a new reading was published
        previousMillis = currentMillis;
        // New DHT sensor readings
        hum = dht.readHumidity();
        // Read temperature as Celsius (the default)
        temp = dht.readTemperature();
        // Read temperature as Fahrenheit (isFahrenheit = true)
        //temp = dht.readTemperature(true);
        gaz = sensorValue = analogRead(A0);
        sensorVoltage = sensorValue;
        // Publish an MQTT message on topic esp/dht/temperature
        uint16_t packetIdPub1 = mqttClient.publish(MQTT_PUB_TEMP, 1, true, String(temp).c_str());
        Serial.printf("Publishing on topic %s at QoS 1, packetId: %i ", MQTT_PUB_TEMP, packetIdPub1);
        Serial.printf("Message: %.2f \n", temp);
        // Publish an MQTT message on topic esp/dht/humidity
        uint16_t packetIdPub2 = mqttClient.publish(MQTT_PUB_HUM, 1, true, String(hum).c_str());
        Serial.printf("Publishing on topic %s at QoS 1, packetId %i: ", MQTT_PUB_HUM, packetIdPub2);
        Serial.printf("Message: %.2f \n", hum);
        // Publish an MQTT message on topic esp/dht/gaz
        uint16_t packetIdPub3 = mqttClient.publish(MQTT_PUB_GAZ, 1, true, String(gaz).c_str());
        Serial.printf("Publishing on topic %s at QoS 1, packetId %i: ", MQTT_PUB_HUM, packetIdPub3);
        Serial.printf("Message: %.2f \n", gaz);
    }
}

```

Figure 30 : Quatrième partie du programme 'carte Wemos'

10.2. Moniteur série de la carte Wemos

Lors de la téléversement du programme, nous pouvons afficher les mesures des différents capteurs sur le moniteur série comme le montre la figure ci-dessous :

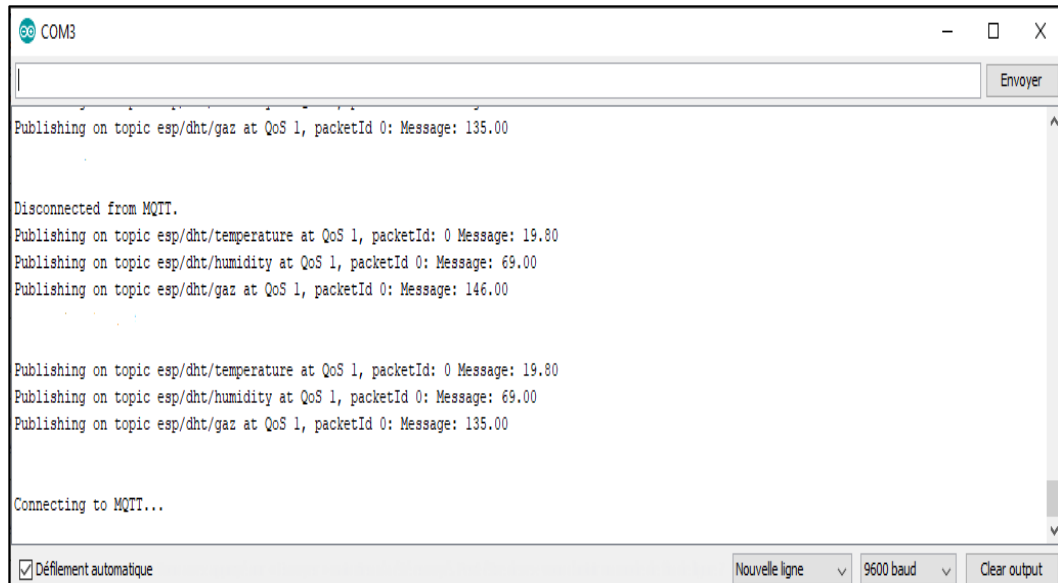


Figure 31 : La moniteur série du traitement du carte Wemos

10.3. Programmation de la carte Arduino

La carte Arduino est connectée avec plusieurs capteurs (LM35, LDR, module pluie). Le programme développé est le suivant :

```
int val;
int tempPin = A0;
const int LED_PIN = 9;
const int ON_COMMAND = 'C';
const int OFF_COMMAND = 'c';
void setup() {
    pinMode(LED_PIN, OUTPUT);
    Serial.begin(9600); // Match to "Serial out" node configuration
}

void loop() {
    int TV;          // incoming serial byte
    if (Serial.available() > 0) { // is a serial byte ready?
        // get incoming byte:
        TV = Serial.read();
        if (TV == ON_COMMAND) {
            digitalWrite(LED_PIN, HIGH);
        } else if (TV == OFF_COMMAND) {
            digitalWrite(LED_PIN, LOW);
        }
    }
}
```

Figure 32 : Première partie du programme de la carte Arduino

```

val = analogRead(tempPin);
Serial.print(val);
if (val > 50){
  digitalWrite(LED_PIN, HIGH); // send signal from pin 9 to circuit
  delay(10000);
  digitalWrite(LED_PIN, LOW); // send signal from pin 9 to circuit
  delay(10000);}
float mv = ( val/1024.0)*5000;
float cel = mv/10;
float farh = (cel*9)/5 + 32;
Serial.print("TEMPERATURE = ");
Serial.print(cel);
Serial.print("*C");
Serial.println();
delay(1000);
}

```

Figure 33 : Deuxième partie du programme de la carte Arduino

11. Conclusion

La serre intelligent est une solution pour plusieurs contraintes. Cette méthode nous donne la possibilité d'augmenter la production et est un moyen de surveiller les plantes agricoles à distance.

Le troisième chapitre est consacré pour la validation expérimentale via la carte Raspberry, dans lequel on détaillera la programmation de l'interface graphique avec Node red puis l'accès à distance du système conçue.

Chapitre 3 : Expérimentations

1. Introduction

La robotique est au service de l'homme dans de nombreux domaines tels que l'industrie, médecine, agriculture, militaire, ...etc. Dans ce chapitre, nous allons présenter les étapes de programmation de la carte Raspberry Pi 4 dans le cadre d'un environnement robotique. Dans un premier temps, nous allons présenter la carte Raspberry PI 4B+. Dans un second temps, nous allons voir en détails les étapes de création d'une interface de supervision et de contrôle d'une motopompe d'irrigation et de température / humidité sous le logiciel Node-red. Enfin, nous présenterons les différentes procédures, programmes et le principe de contrôle de notre système.

2. Carte Raspberry

2.1. Définition

De nos jours, le Raspberry pi est devenu un phénomène mondial. C'est un ordinateur qu'il fonctionne sous le système d'exploitation linux à partir d'une carte SD.

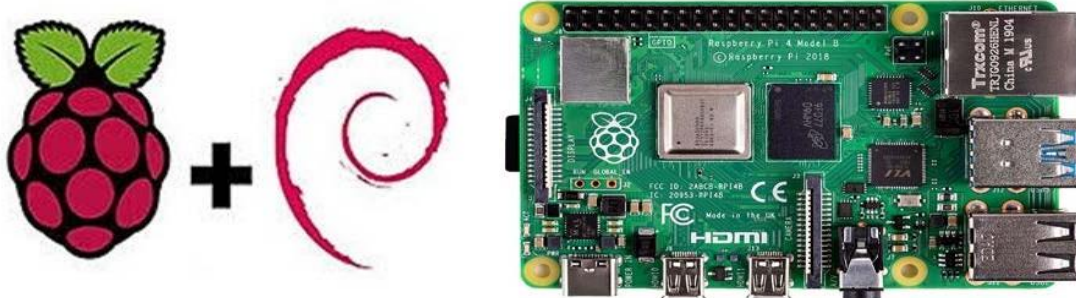


Figure 34 / Carte Raspberry Pi.

Le but de la fabrication de cette carte est pour faciliter l'étude des ordinateurs , aussi il est destiné à favoriser et encourager l'apprentissage de la programmation informatique en plusieurs langages (C, C++, python...) et d'être utilisé dans les systèmes embarqués ,mais le Raspberry pi est non seulement un excellent moyen d'acquérir des compétences en programmation dans le cadre de projet universitaire, il y'a aussi des douzaines d'applications Trans-curriculaires comme sciences, musique et même à l'installation des jeux vidéo.

2.2. Historique

En 2006, les premiers prototypes de Raspberry ont été élaborée. Dès le début, le projet a souligné sa volonté d'apporter des solutions pour inciter les jeunes à programmer et il a été inspiré par le BBC Micro (British Broadcasting Corporation Micro-computer System), qui a débuté en 1981 et a cessé sa production en 1986, et poursuit les mêmes objectifs.

La Raspberry Pi a été très réussi, et de nouvelles versions ont été publiées, celles-ci étaient plus puissantes ou possédant d'autre interfaces. Ce projet arrive à sa quatrième génération, et chaque génération apporte sa propre part de développement.



Figure 35 : Évolution des cartes Raspberry pi

2.3. Les caractéristiques des cartes Raspberry pi

La carte Raspberry est constituée essentiellement de :

- Processeur ARM : Les architectures ARM sont des architectures des processeurs, à faible consommation, proposée par "Acron Computers" en 1983 et développée par "ARM Ltd" en 1990 ;
- Mémoire vive RAM : C'est la mémoire dans laquelle le Raspberry place les données lors de son traitement ;
- Une connectivité réseaux ;

- HDMI : High Définition Multimédia Interface, permet de Connecter le Raspberry pi à un appareil compatible soit un écran LCD ou vidéoprojecteur ;
- Port USB 2.0 : Le port Universal Serial Bus est un port série qui sert à connecter le Raspberry aux autres périphériques ;
- Port Ethernet : C'est un port correspondant au protocole international ETHERNET utilisé pour le LAN à commutation des paquets ;
- Un slot micro SD : nécessite un stockage externe supplémentaire pour fonctionner. Ce slot permet de connecter la mémoire externe ;
- Prise RCA : Radio Corporation of American est un connecteur électrique utilisé dans le domaine audio/vidéo ;
- Une prise jack : C'est une connecteur audio-vidéo ;
- GPIO : General Purpose Input/Output, sont des ports d'Entrée/Sortie.

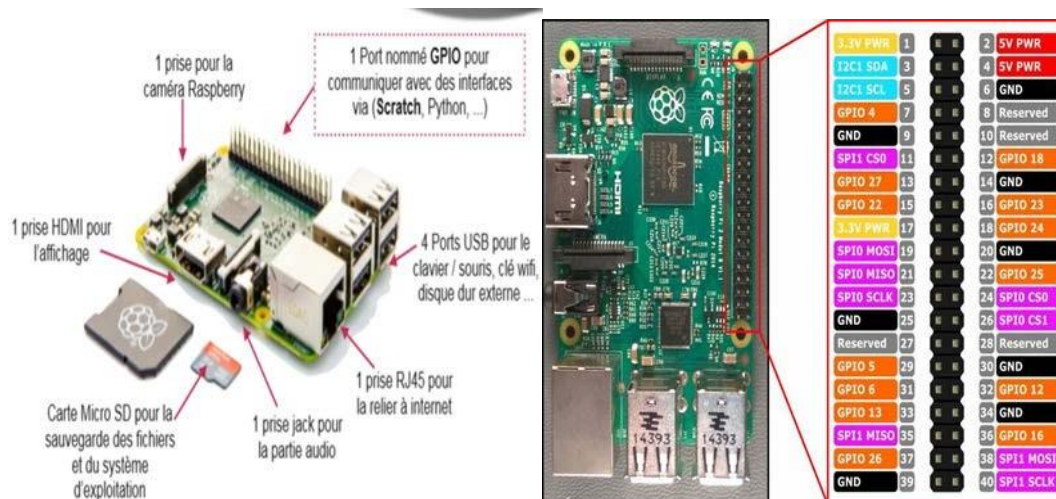


Figure 36 : Les composants standards de la carte Raspberry pi

❖ Exemple :

Les caractéristiques techniques de carte Raspberry pi 4 model B :

Tableau 3-1: Fiche technique de carte Raspberry Pi 4B.

Informations générales	Marque	Raspberry
	Modèle	Raspberry Pi 4 Model B
Processeurs	Processeur	ARM Cortex A53
	Type de processeur	Broadcom BCM 2837BO (ARM Cortex Quad-Core 1.4 GHz)
	Fréquence CPU	1.4GHz
RAM	Taille de la mémoire	4GO

Connectivités	Connecteurs disponibles	1 X Jack 3.5mm femelle stéréo
		1 X RJ45 femelle
		4 X USB 2.0
		1 X HDMI
	Type d'alimentation	Port USB
	Lecteur des cartes	Micro SD
		Micro SDHC
		Micro SDXC
Communication	Normes réseau	Bluetooth 4.2
		Wi-Fi AC

2.4. Matériels utilisés avec la carte Raspberry

❖ Une alimentation

Afin de se connecter à une prise de courant, tous les modèles de Raspberry Pi disposent d'un port USB (le même port sur de nombreux téléphones). Généralement, nous avons besoin d'une alimentation qui fournit au moins 2.0 ampères pour Raspberry Pi 3.



Figure 37 : Chargeur de carte Raspberry Pi

❖ Une carte micro SD

Tout le Raspberry Pi a besoin d'une carte SD pour mémoriser tous les fichiers et le système d'exploitation Raspbian. Dans notre cas, nous avons besoin d'une carte micro SD de capacité 16Go.



Figure 38 : Carte micro SD

❖ *Câble HDMI*

Le Raspberry Pi possède un port de sortie HDMI compatible avec le port HDMI de la plupart des téléviseurs et écrans d'ordinateur modernes.



Figure 39 : Câble HDMI

❖ *Un écran de télévision ou d'ordinateur*

Pour visualiser l'environnement de bureau Raspbian, nous avons besoin d'un écran et d'un câble pour relier l'écran et le Pi.

❖ *Un clavier ou une souris*

Pour commencer à utiliser votre Raspberry Pi, vous avez besoin d'un clavier USB et d'une souris USB.

❖ *Un boîtier*

Afin de gagner en esthétique et de le protéger contre la poussière, un boîtier dédié à l'ordinateur.



Figure 40 : Boîtier pour la Raspberry Pi

3. Les étapes de création d'une interface de supervision sous le logiciel Node-Red

3.1. Préparation du système d'exploitation

Pour installer le système Raspbian dans la Raspberry Pi :

D'abord il faut avoir une carte mémoire de taille 8g au minimum et le formater par le logiciel « SD Card Formatter »

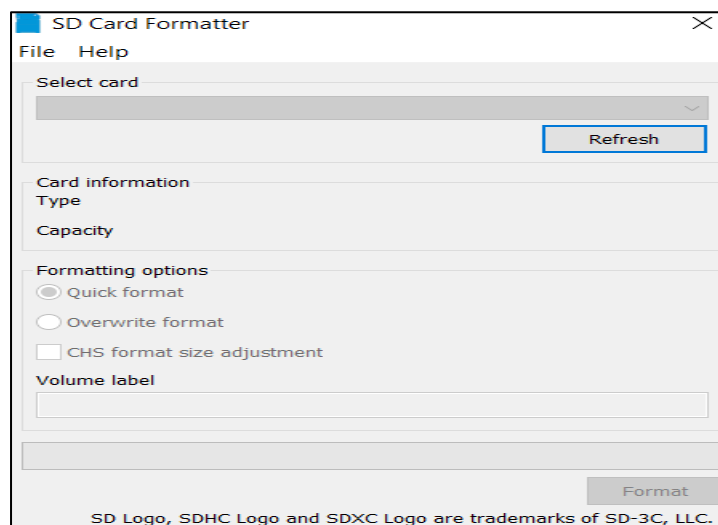


Figure 41 : SD Card Formatter

Après il faut télécharger le fichier Raspbian depuis le

lien : <https://www.raspberrypi.com/software/operating-systems/> et écrire ce fichier sur la carte mémoire par un logiciel tel que « Raspberry Pi Imager »



Figure 42 : Raspberry Pi Imager

A cette étape, il faut ajouter dans la carte mémoire un fichier ssh sans extension comme celui suivant pour booter par la méthode ssh :



Figure 43 : Fichier SSH

Ensuite la Raspberry Pi peut être bootable maintenant, il faut savoir l'adresse IP du Raspberry par le logiciel « Advanced IP Scanner ».

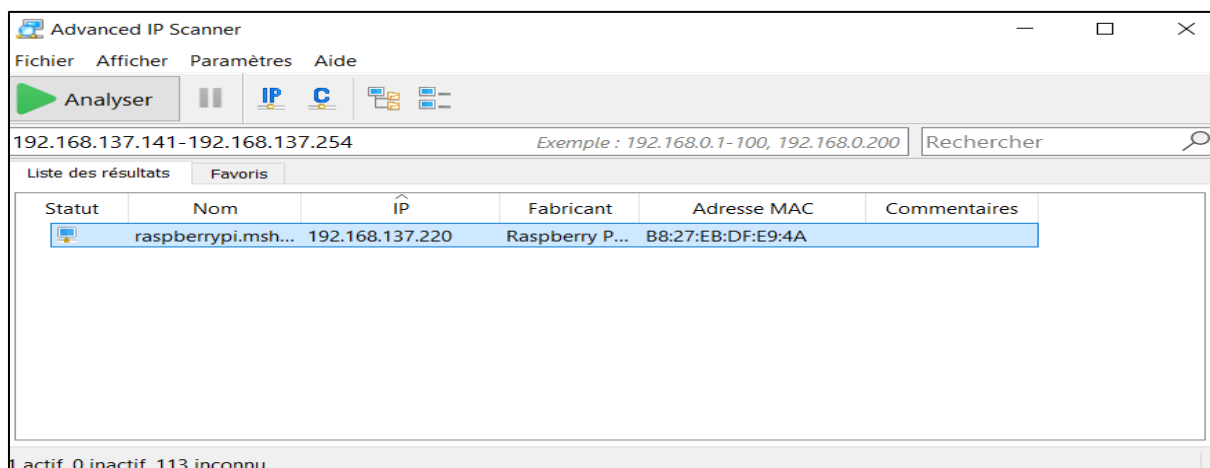


Figure 44 : Adresse IP du Raspberry

On peut accéder à la Raspberry par « PuTTY » et mettre l'adresse IP.

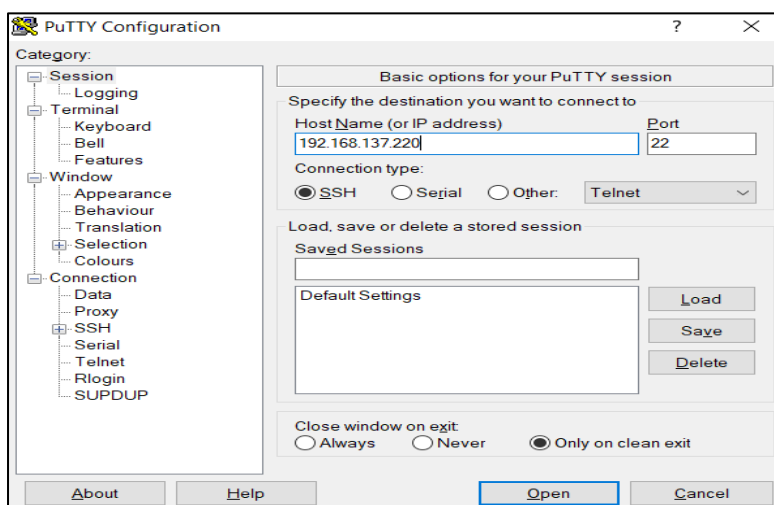


Figure 45 : Interface du PuTTY

Alors maintenant il faut entrer :

- Nom d'utilisateur : pi
- Mot de passe : raspberry



Figure 46 : La page de connexion avec Raspberry

3.2. Configuration d'une IP fixe

Pour faire la configuration d'une IP fixe de carte Raspberry pi on doit :

- 1) Ouvrir le fichier `/etc/dhcpd.conf` avec nano (ou un autre éditeur texte, peu importe).
- 2) Rendez-vous à la fin et ajoutez les lignes ci-dessous, en remplaçant : "wlan0 par eth0 si vous êtes connecté en Ethernet."

interface wlan0:

- static ip_address=192.168.43.119/24.
- static routers=192.168.1.1.

3.3. Les étapes pour mettre à jour le système de carte Raspberry Pi

Pour faire la mise à jour du système de carte Raspberry pi on exécute ces deux commandes dans la terminale de la Raspberry pi :

- Sudo apt-get update.
- Sudo apt-get upgrade.

3.4. Installation de Mosquitto sur Raspberry pi

3.4.1. Description

Mosquitto est un serveur MQTT, qui peut être installé sur le Raspberry Pi ou sur presque tous les systèmes d'exploitation (Windows, linux ...) et il est développé à l'origine pour simplifier la communication entre les machines. MQTT (Message Queue Telemetry Transmission) est un protocole de messagerie très rapide et léger. Il se divise en 3 types d'entités : les publishers qui envoient des messages, les subscribers qui écoutent pour recevoir des messages et un broker qui est le serveur qui fait le lien entre les deux. Les publishers et subscribers communiquent grâce à un topic auquel ils sont suscrits.

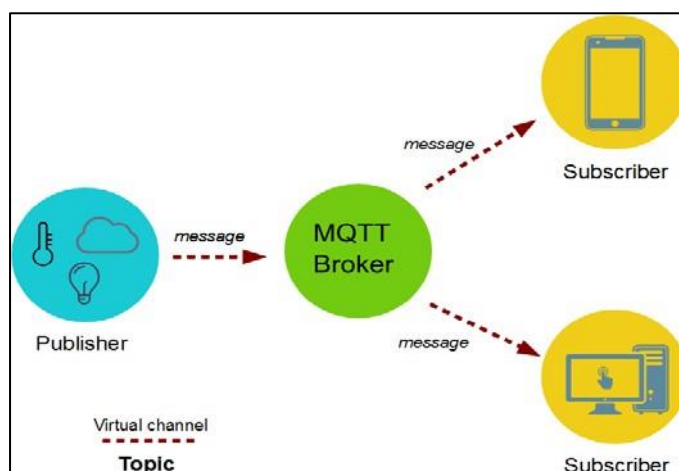


Figure 47 : Le serveur MQTT

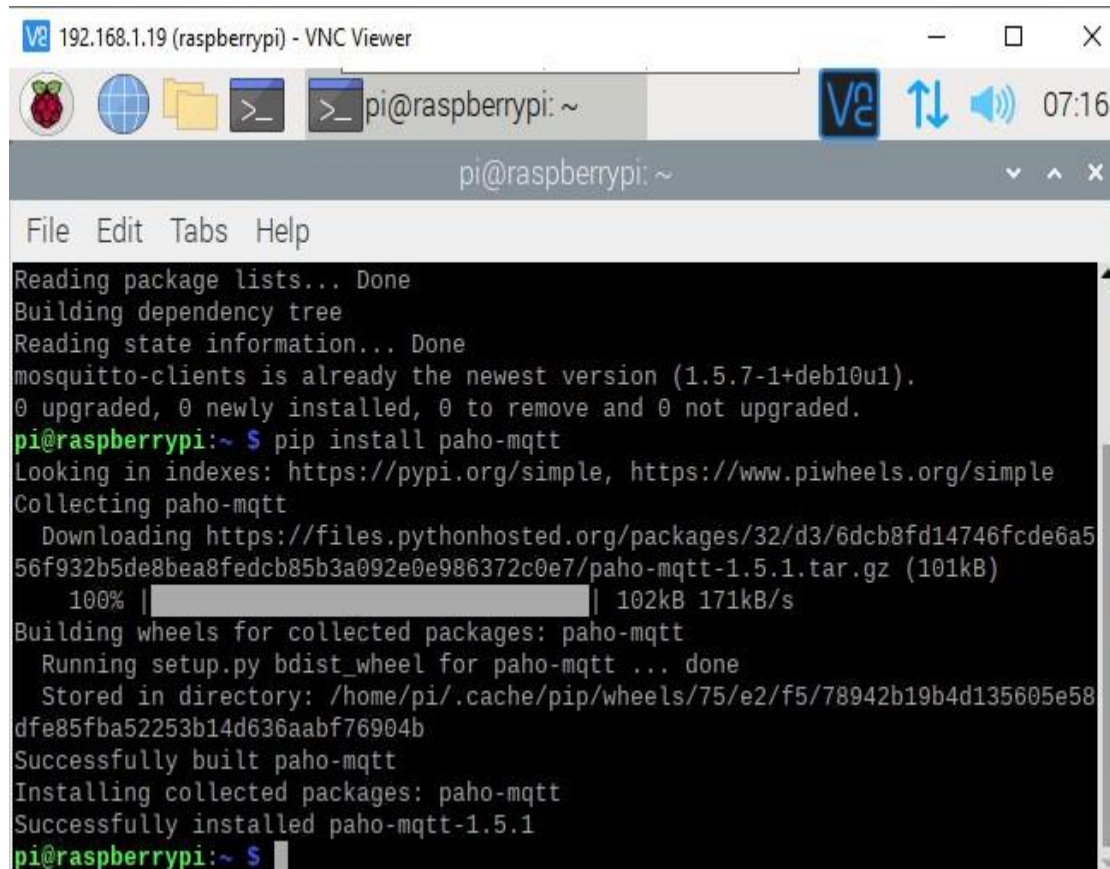
3.4.2. Étapes d'installation de Mosquitto sur Raspberry pi

L'installation du Broker Mosquitto est très simple :

```

pi@raspberrypi:~$ sudo apt-get install mosquitto
Reading package lists... Done
Building dependency tree
Reading state information... Done
mosquitto is already the newest version (1.5.7-1+deb10u1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~$ sudo apt-get install mosquitto-clients
Reading package lists... Done
Building dependency tree
Reading state information... Done
mosquitto-clients is already the newest version (1.5.7-1+deb10u1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~$
  
```

Figure 48 : Les codes d'installation



```

192.168.1.19 (raspberrypi) - VNC Viewer
pi@raspberrypi: ~
File Edit Tabs Help
Reading package lists... Done
Building dependency tree
Reading state information... Done
mosquitto-clients is already the newest version (1.5.7-1+deb10u1).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
pi@raspberrypi:~ $ pip install paho-mqtt
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting paho-mqtt
  Downloading https://files.pythonhosted.org/packages/32/d3/6dcb8fd14746fcde6a556f932b5de8bea8fedcb85b3a092e0e986372c0e7/paho-mqtt-1.5.1.tar.gz (101kB)
    100% |#####| 102kB 171kB/s
Building wheels for collected packages: paho-mqtt
  Running setup.py bdist_wheel for paho-mqtt ... done
  Stored in directory: /home/pi/.cache/pip/wheels/75/e2/f5/78942b19b4d135605e58dfe85fba52253b14d636aabf76904b
Successfully built paho-mqtt
Installing collected packages: paho-mqtt
Successfully installed paho-mqtt-1.5.1
pi@raspberrypi:~ $

```

Figure 49 : Installation de Mosquito

3.5. Node-red

3.5.1. Description

Node-Red est un logiciel permettant de gérer des flows d'événements, des suites des traitements qui seront exécuté après la réception d'un message ou d'un événement, et il est développé initialement par l'équipe de service des technologies émergentes d'IBM au début de 2013 en tant que projet parallèle de Nick O'Leary et Dave Conway-Jones.

C'est un outil qu'il rendre l'IoT plus accessible en aidant les développeurs à connecter rapidement à des périphériques matériels, des API et des services en ligne.

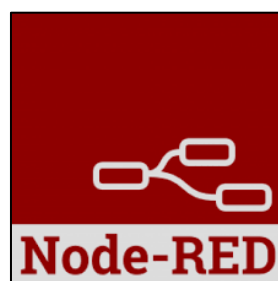


Figure 50 : Logiciel Node-red.

3.5.2. Installation de Node-red

La commande ci-dessous lance le script d'installation qui s'occupe de tout :

```
Bash < (curl -SL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered).
```

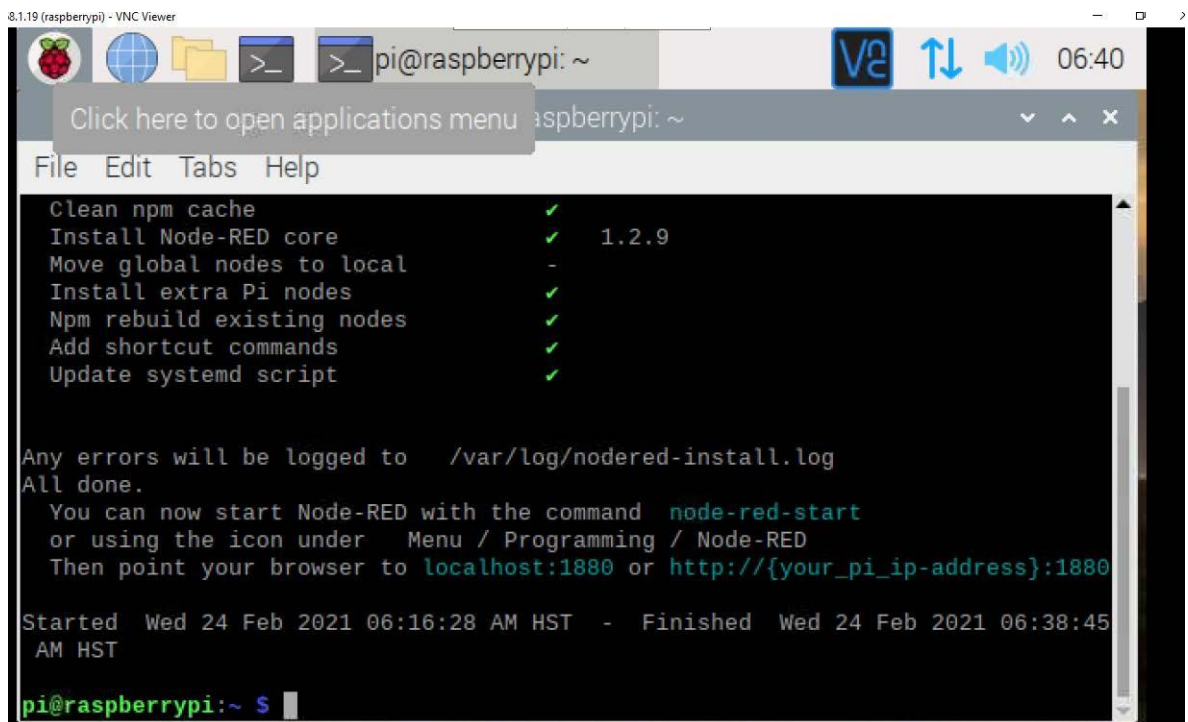


Figure 51 : Installation de Node-Red.

3.5.3. Démarrage de node-red

1) Après l'installation de Node-Red sur la distribution desktop, on peut l'accéder depuis :

Menu -> Programming -> Node-Red.

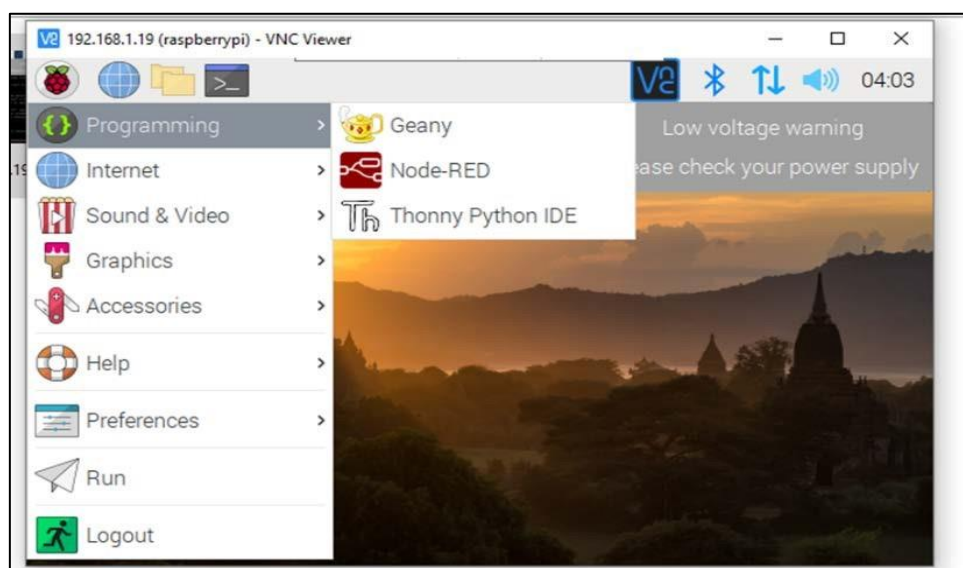


Figure 52 : Lancement de Node-Red.

- 2) Lorsque en cliquant sur l'icône de Node-Red, nous avons apparaître des lignes de texte dans la fenêtre du Terminal :

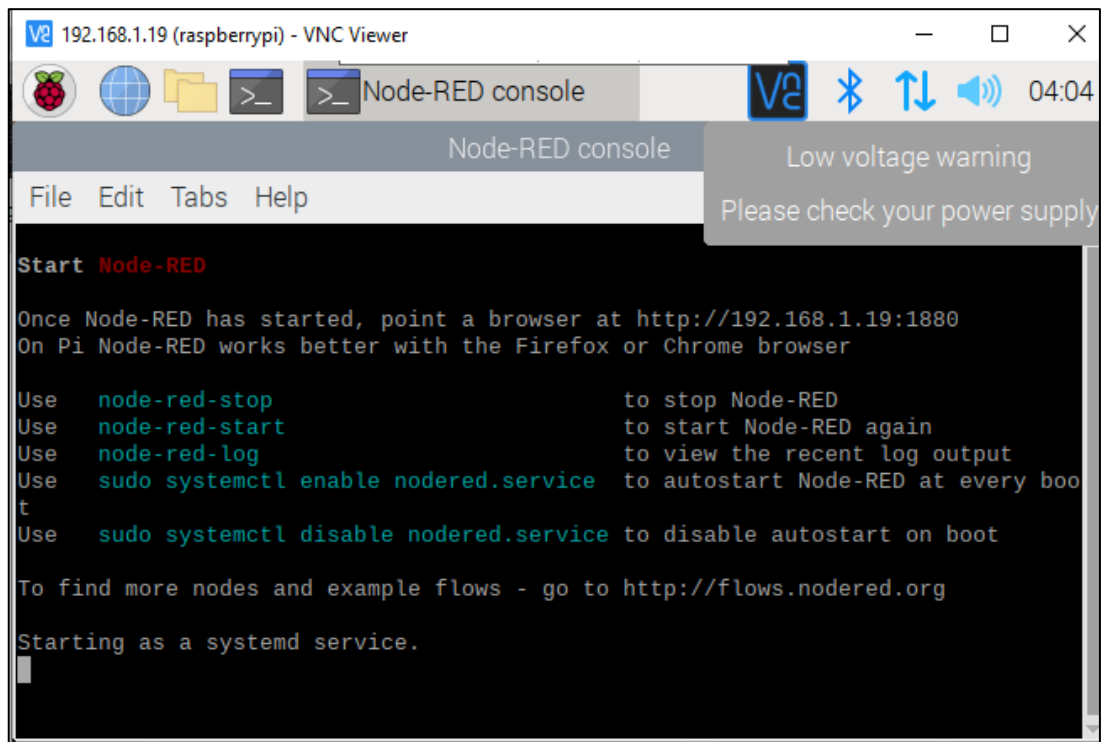


Figure 53 : Fenêtre terminale de Node-Red dans Raspberry.

Nous pourrions ensuite lancer Node-Red grâce à la commande : **Node-Red-Start**.

Pour arrêter Node-Red, nous utilisons la commande : **Node-Red-stop**.

Il est possible de lancer Node-Red automatiquement au démarrage de OS de Raspberry pi :

Sudo systemctl enable node red. Service.

Pour arrêter le démarrage automatique on utilise cette commande :

Sudo systemctl disable node red. Service.

Pour accéder à l'interface de Node -Red depuis un navigateur internet on saisit :

Http ://ip-de-votre-Raspberry :1880.

3.5.4. Interface de Node-Red

L'interface de Node-Red se compose de 4 parties, comme le montre la figure ci-dessous :

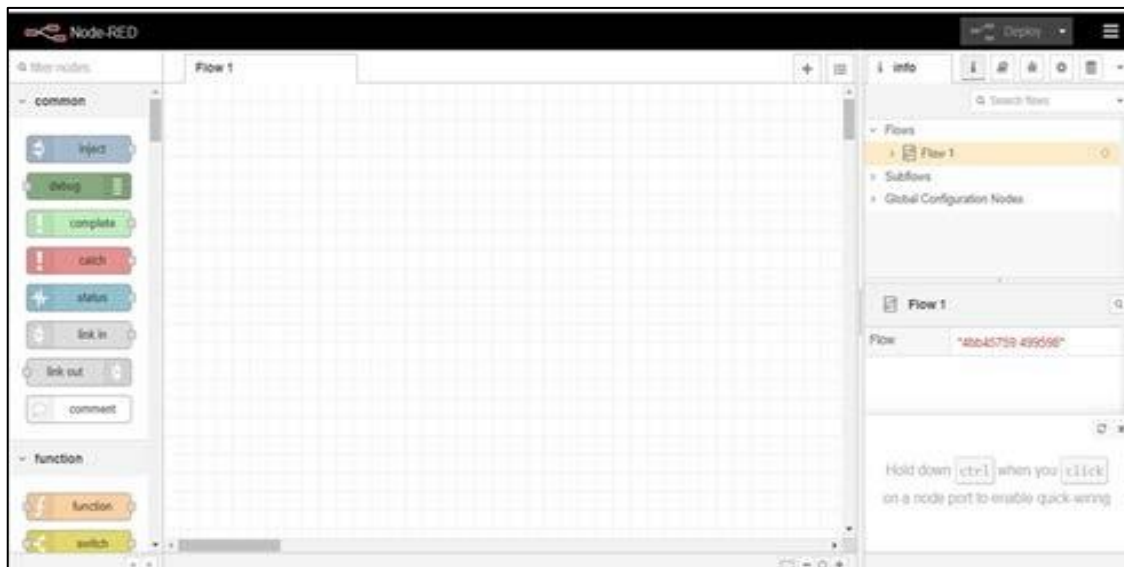


Figure 54 : Interface de Node-Red.

Les 4 parties de Node-Red sont :

- ✚ Au centre : **les flows**, chaque flow est indépendant et on ne peut pas agir sur d'autres;
- ✚ A gauche : **la liste des nodes**, pour le placer sur le flow on la sélectionne et on la glisse jusqu'à l'endroit voulu;
- ✚ A droite : Il y'a des onglets utiles :
 - L'onglet **i** permet d'avoir des informations détaillées sur toute node sélectionnée;
 - L'onglet **debug** il apparaît dès qu'une node debug est placée;
 - L'onglet **Dashboard**, une fois que le nœud du tableau de bord apparaît, il apparaîtra et vous permettra d'y accéder;
- ✚ En haut :

Le bouton **Deploy** qui permet de déployer le flow et le rendre actif;

Le bouton **Menu** est contenu de des plusieurs options :

View : permet de gérer la vue;

Import : permet de charger un flow sauvegardé;

Export : permet de sauvegarder les flows ouverts;


Manage Palette : permet de gérer les nodes installées et installer des nouveaux;

Flows / Subflows : permet de créer un nouveau flow ou subflow.

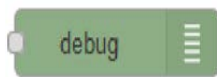
3.5.5. Les nodes de base

Node-Red contient plusieurs nodes de base qui sont très utiles ou pratiques. Ces nodes se retrouvent dans tout flow quel que soit le domaine et ils sont classés par fonctionnalité.

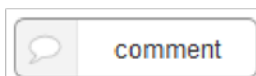
Les fonctionnalités de base dans Node Red sont :

 **Common** : Nodes communes, nous permette de faire des opérations simples sans traitements.


Exemples :



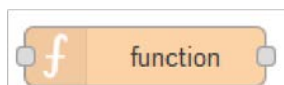
: Permet d'afficher un message de debug et pour tester toute fonction ou node ajoutée.



: Permet d'écrire un commentaire.

 **Function** : nous permette d'agir sur les messages, de modifier leur contenu et de leur soumettre des traitements.

Exemples :



: Permet de créer une fonction en JavaScript qui est utile pour traiter un message reçu pour le rendre utilisable par une node de sortie.



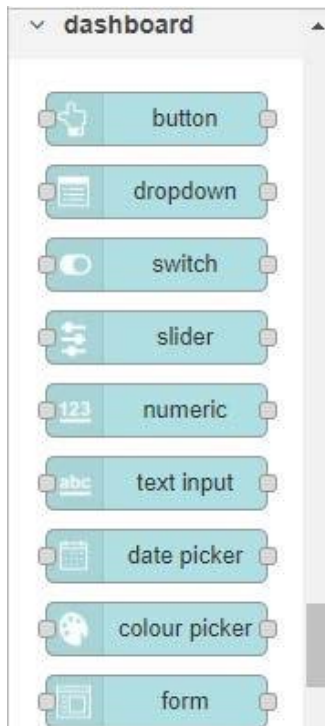
: Permet d'imposer un délai aux messages entrants.

🚦 **Dashboard** : Un Dashboard est une interface graphique composée de plusieurs éléments, chacun représenté par une node. Nous allons dans le menu **Manage Palette sur Node-Red** et entrez **dashboard** dans le volet **Install**.

🚦 Après sélectionnez le module **Node-Red-dashboard** et installez-le.

On peut les diviser en deux types :

➤ Éléments d'entrée :



: Un simple bouton.

: Une liste déroulante.

: Bouton on/off.

: Une barre de valeurs ajustable.

: Un champ pour choisir un nombre.

: Un champ pour entrer un texte.

: Un champ pour choisir une date.

: Un champ pour choisir une couleur.

: Un formulaire qui contient plusieurs champs des plusieurs types différents (texte, mot de passe, email, date...).

➤ Éléments des sorties :



: Un texte.

: Une jauge, qui peut être des plusieurs formes.

: Un graphique, qui peut être des plusieurs types.

: Permet de jouer un fichier audio.

: s'active lorsqu'il reçoit un message.

: c'est un contrôle dynamique de l'interface


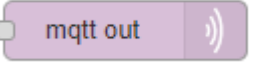
: Permet d'utiliser des directives HTML.

: permet d'intégrer une carte du monde.

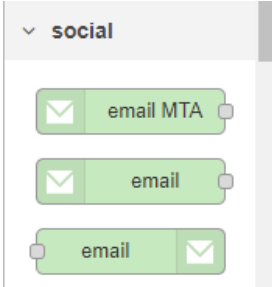
✚ **Network** : nous permette de gérer l'aspect réseau du flow, en paramétrant des requêtes http, des messages et aussi elle range les nodes mqtt (mosquitto).

Exemple :


Sur Node-red nous pouvons installer le module **Node-Red-contrib-mqtt-broker** pour que le broker soit directement sur Node-Red, et non Mosquitto.

	: La node mqtt in reçoit les messages de tous les topics.
	: La node mqtt out ou le Publisher, elle s'abonne à ces topics et y envoie les messages.

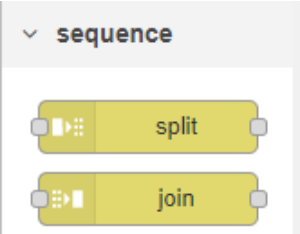
✚ **Social** : nous permette d'envoyer des e-mails depuis node-red.


	: Pour l'installer nous allons dans le menu Manage Palette , entrons email dans le volet Install et Sélectionné le module node-red-node-email .
--	--

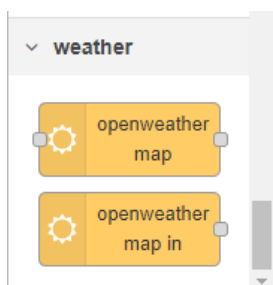
✚ **Storage** :

	: nous permette de sauvegarder des données de messages dans des fichiers, aussi de surveiller des fichiers pour y détecter tout changement.
---	---

✚ **Séquence** : nous permette d'agir sur la séquence de messages transmis et ainsi d'agir sur le déroulé du flow.

	: Permet de diviser un message entrant en plusieurs message sortants.
	: Permet de regrouper plusieurs messages entrants en un seul.

 **Weather :** nous permet de obtenir le rapport météo et les prévisions d'OpenWeatherMap.



: Pour l'installer on exécute la commande suivante dans le répertoire racine de notre installation Node-Red (**npm install node-red-node-OpenWeatherMap**).

3.6. Exécution de programme

3.6.1. Implémentation de programme de fonctionnement d'une motopompe d'irrigation et de contrôle taux de température / humidité/gaz sous forme deJSON en node-red

Lorsque les nœuds sont développés en JavaScript donc sont enregistrés sous format JSON, pour implanter notre programme et l'exécuter, nous avons suivi les étapes suivantes :

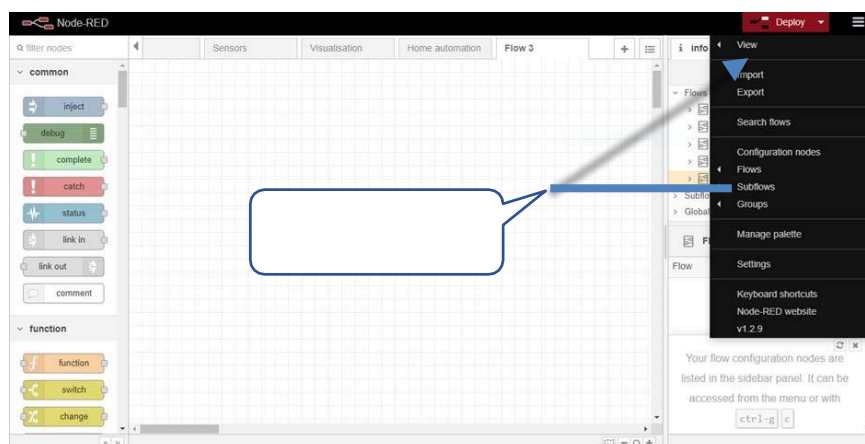


Figure 55 : 1ère étape d'implémenter une fichier JSON

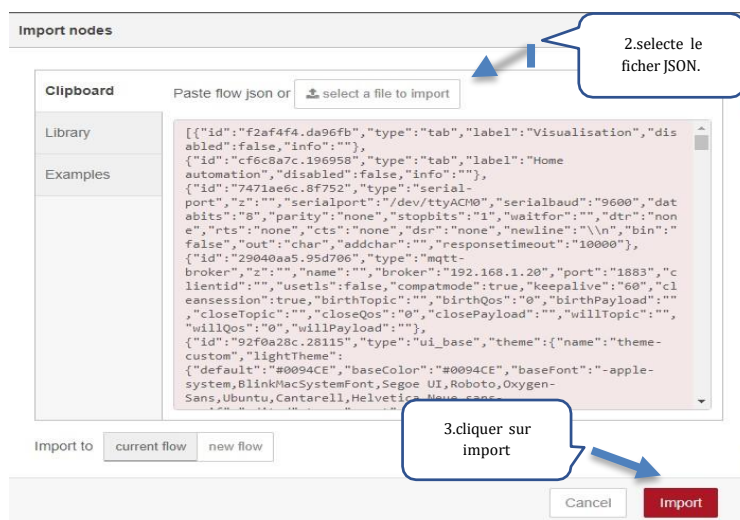


Figure 56 : 2ème étape d'implémentation.

Après l'implantation de fichier JSON, on aura le flow suivant qui représente notre programme de contrôle sur Node-Red.

Figure 57 : Le programme de fonctionnement d'une motopompe d'irrigation et de contrôle température/ humidité/gaz.

3.6.2. Implémentation du programme de contrôle météorologique

❖ DHT11 et MQ5 via Wemos

Nous avons utilisé la node **mqtt in** pour recevoir les messages de tous les topics par l'adressage de notre IP (192.168.43.119) et l'on a nommée wemos-DHT11, parce que la carte connectée à notre système est Wemos qui effectue trois lectures pour « température », « humidité » à l'aide de capteur DHT11 et pour « Gaz » par le MQ5.

Par la suite, il faut utiliser la node **Function** qui doit être écrite en JavaScript pour lire les topics envoyés à la carte (T, H et gaz) et les données sont ensuite récupérées par les nodes de **dashboard** qui permettent de les affichés soit graphiquement ou sous forme des gauges ou bien texte.

Au cours de cette opération et lorsqu'il y'a un dépassement au niveau du gaz c'est-à-dire que lorsque le taux de (gaz >200) , le ventilateur fonctionne.

Figure 58 : Programme du système de contrôle avec DHT11

❖ LM35 via Arduino

Tout d'abord, On va commencer par se connecter à l'Arduino à l'aide du Node Serial qui se trouve dans la bibliothèque de node input. On glisse la node sur le flow et on fait un double clic sur la Node Serial pour ouvrir le panneau de configuration, et après on choisit le port puisque l'Arduino est connecté sur un Raspberry Pi 4, donc Sur Windows, ce sera un port COMx.

Ensuite, à l'aide du capteur LM35 qui est connecté sur la carte Arduino, on peut contrôler l'irrigation automatique, c'est à dire le fonctionnement d'une motopompe. Puisque le LM35 est un capteur de mesure de température, donc dans notre système on a ajusté la température à une valeur supérieur ou égale à 50° et lorsque la température mesurée dépasse le seuil prédéfinie l'irrigation se déclenche d'une façon automatique.

Au cours de cette opération et lorsqu'il y'a un dépassement au niveau de la température une led rouge s'allume et un message d'alerte « high température » est envoyé à notre email.

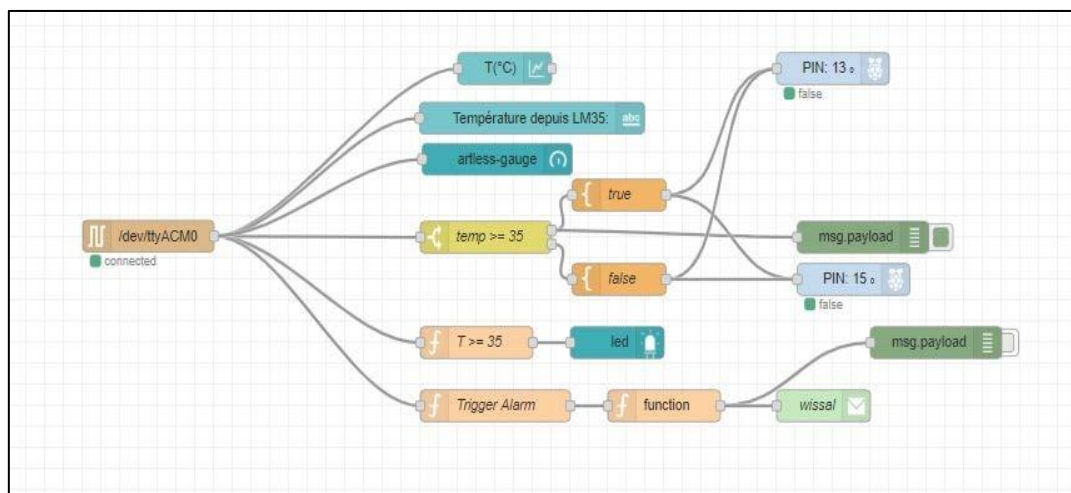


Figure 59 : Programme de système d'activation de motopompe

❖ Open Weather

Pour ce projet, nous allons créer un Widget qui affiche la météo du jour à l'aide de la node open Weather map.

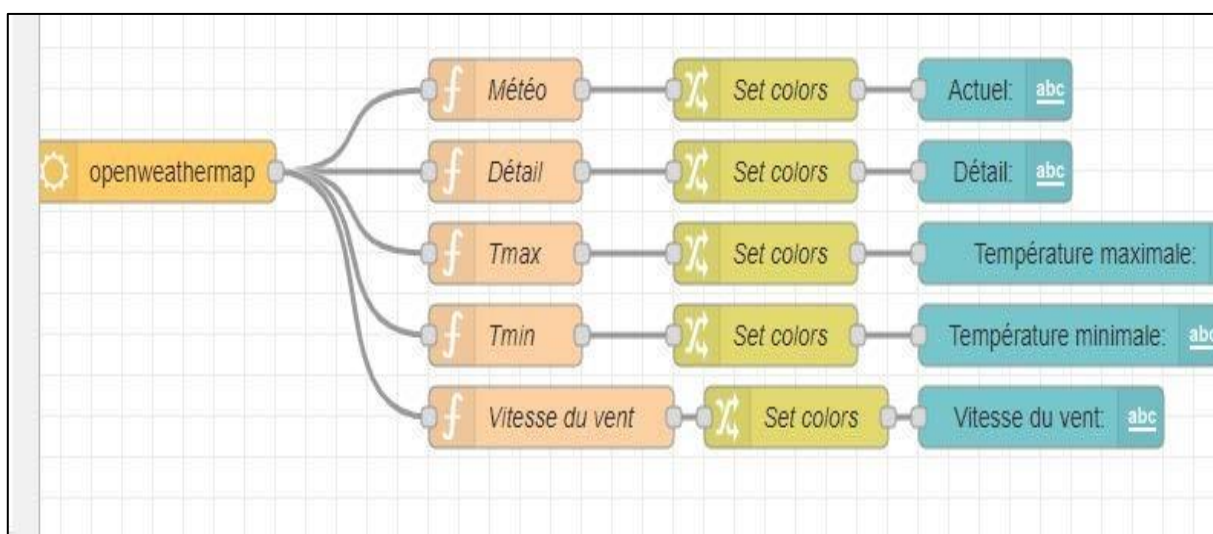


Figure 60 : Programme pour la récupération des données météo depuis le site OpenWeatherMap.

3.7. Access à distance

Avec Remote-RED, nous pouvons accéder à notre Node-Red dashboard avec téléphone portable de n'importe quel emplacement. Il est considéré comme un tunnel entre notre installation Node-Red locale dans notre réseau privé et notre smartphone.

Les étapes de configuration de Remote-RED dans node-red, sont les suivants :

- 1) Installation de la bibliothèque **node-red-contrib-Remote** par l'utilisation de la manage palette.

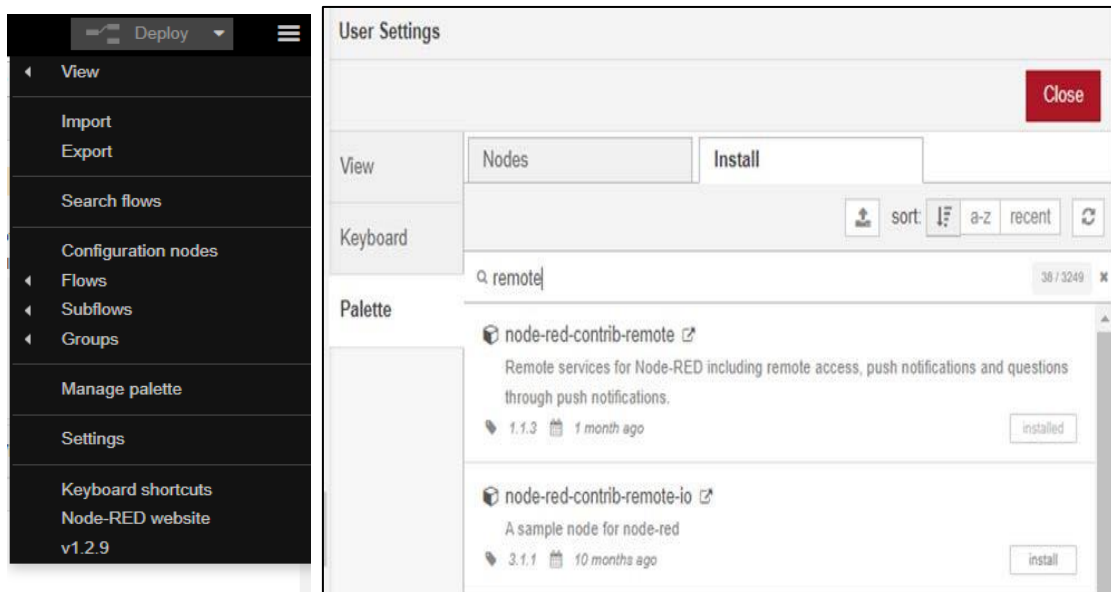


Figure 61 : Installation de bibliothèque Remote-RED

- 2) Le package Remote-Red est ajouté à notre liste des nodes.

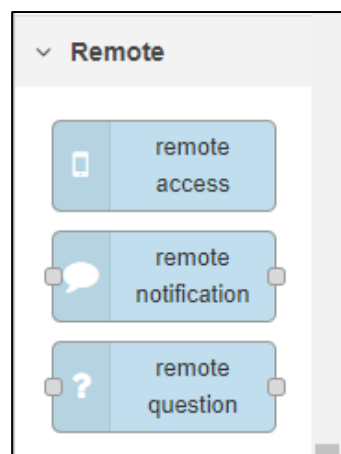


Figure 62 : La bibliothèque Remote-RED

- 3) On glisse la node **Remote Access** qui se trouve dans la bibliothèque **Remote** vers le flow de notre programme.

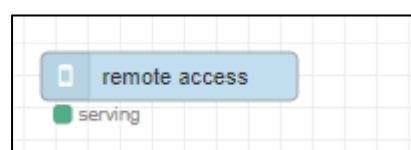


Figure 63 : Access node.

- 4) Pour chacun Access node, il faut créer a correspondant configuration de node. Dans cette configuration on doit ajouter quel host, port et URL où la connexion sera établie.

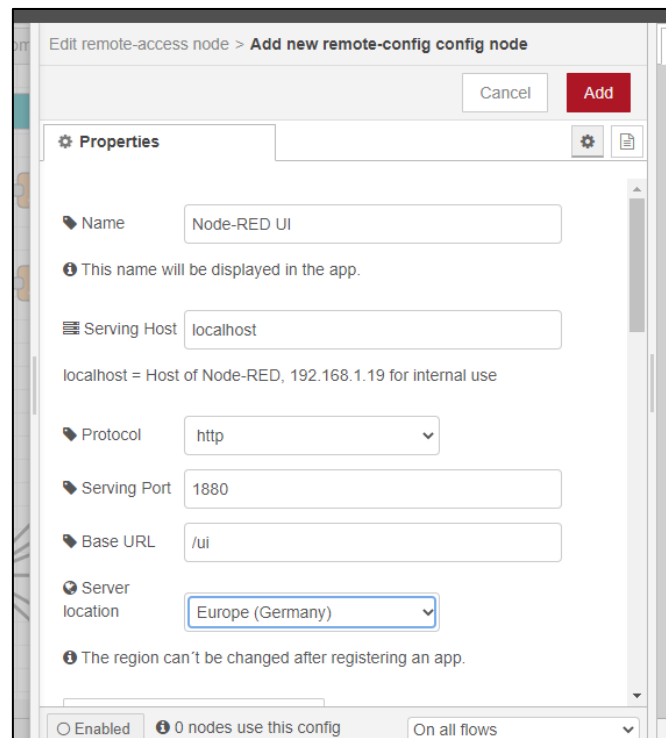


Figure 64 : Configuration de Remote-RED.

- 5) Installation de l'application **Remote red** de Google Play store ou App store.

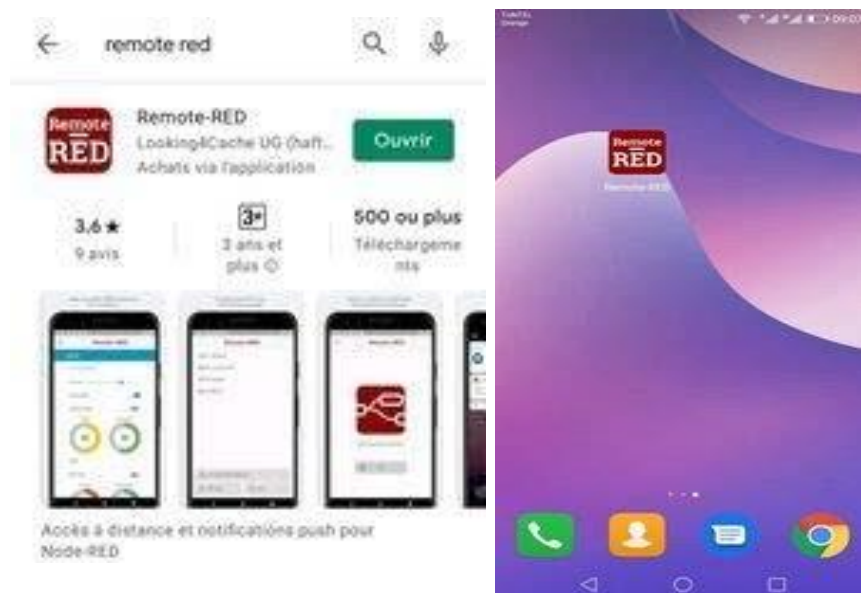


Figure 65 : Installation de l'application Remote-RED sur smartphone.

- 6) Après l'accès aux cette application on utilise le fonction **ADD NODE-RED INSTANCE** et on scan le **QR Code** qui se trouve au niveau de la configuration node de Access Remote dans Node-red.

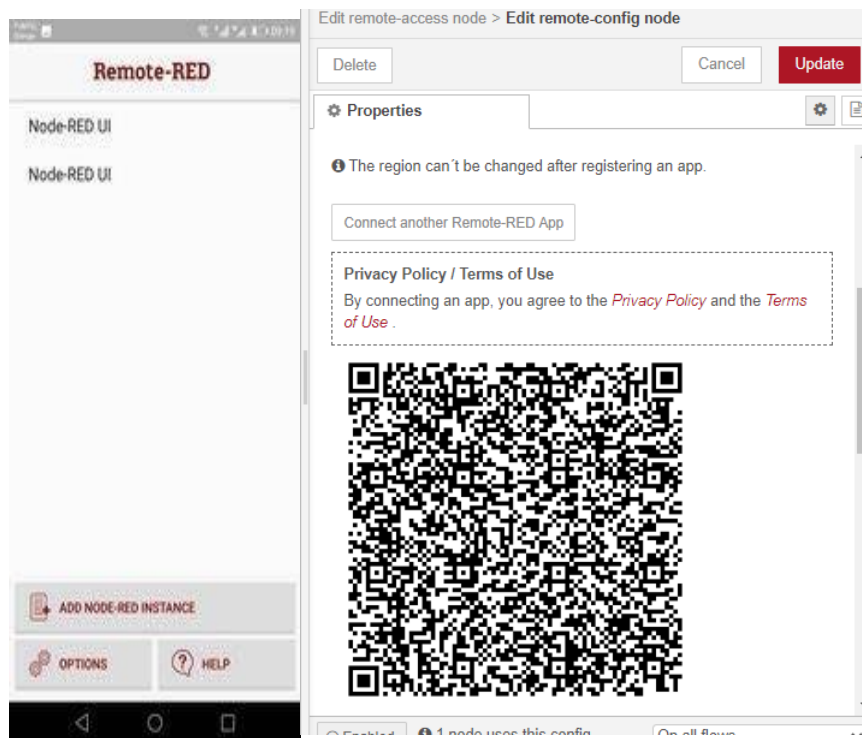


Figure 66 : Connexion de l'application Remote-RED avec programme

L'accès à distance devient plus facile grâce à Remote -RED. Il y'a aussi d'autres solutions pour l'accès à distance de nos programmes, parmi ces solutions, on peut citer le **NO-IP**. A chaque fois notre adresse IP dynamique change le NO-IP, nous pouvons en garder une trace avec notre projet et nous assurons que notre système disposant toujours des données à jour. Ensuite concerné la configuration de le NO-IP, dans node -red on commence :

- 1) Tout d'abord par la création de notre compte No-IP et d'Host Name.

Hostname ▲	Last Update	IP / Target	Type	
nodredwissal.ddns.net Expires in 29 days	Mar 15, 2021 02:45 PDT	102.159.76.40	A	⚙️ Modify ✕

Figure 67 : Création d'host Name

2) Les configurations au niveau de notre routeur wifi :

Tableau (Serveurs virtuels actifs):

Nom du serveur	Protocole	Adresse IP locale	Port local	Adresse IP du réseau étendu	Port Distant (Ouvert)	Etat	Action
Nodred-Pr	tcp+udp	192.168.1.19	1880-1880	pppoe1	1880-1880	Activer	Effacer Désactiver

Figure 68 : Configuration au niveau de service virtuel.

Table dynamique DDNS:

Sélectionner	Etat	Service	Nom d'hôte	Nom d'utilisateur	Interface
<input type="radio"/>	Activer	noip	nodredwissal	wissalachari11@gmail.com	pppoe1

Figure 69 : Configuration au niveau de DDNS.

Et voici donc programme no-ip dans node-red.

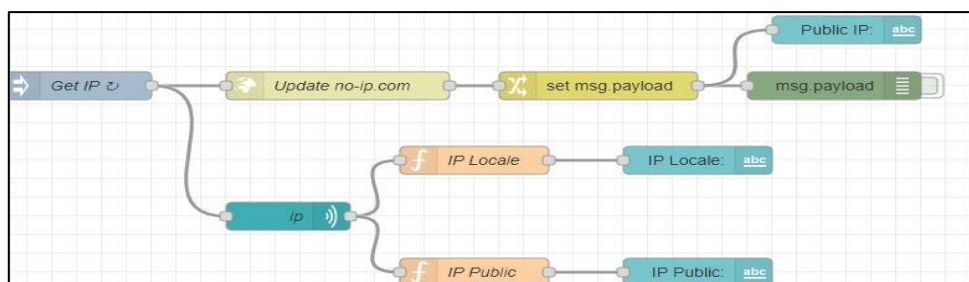


Figure 70 : Programme no-ip

4. Dashboard

Après l'exécution de notre programme, on aura cette dashboard qui permet d'ajouter une très belle interface graphique à un projet Node-Red qui résume tout le programme. Avec l'interface que nous avons développée, on peut surveiller et contrôler notre système d'irrigation.

Figure 71 : Dashboard du programme développé

5. Réalisation et conception

Afin de faciliter l'étude de notre projet, nous avons réalisé un prototype d'une mini serre d'agricole intelligent supervisé par un système de contrôle à base de la Raspberry pi et sous la direction de la logiciel Node-red.

Pour commencer la partie réalisation, tout d'abord nous avons préparé une maquette de dimensions 25cm x 25cm x 10cm, un squelette et une planche en bois pour faciliter de réalisation et de manipulation, aussi les parois sont en verre. Ensuite nous avons testé chaque composant à part et enfin nous avons tout regrouper dans une seule maquette.

Figure 72 : Prototype réel d'une serre intelligente.

6. Conclusion

Au cours de ce chapitre, nous avons présenté les différentes étapes de la réalisation du système d'irrigation avec une motopompe tout en contrôlant la « température/humidité » en utilisant la carte Raspberry, Node-Red. Nous avons bien détaillé les étapes de préparation de la configuration et les outils de communication avec la carte Raspberry. Enfin, nous avons présenté le résultat obtenu après l'exécution de notre programme ainsi que la méthode de connexion sur l'interface que nous avons développée.

Conclusion Générale

L'agriculteur moderne pourra contrôler des paramètres tels que la température, l'humidité et gaz en utilisant son Smartphone à l'aide de l'interface graphique développé sous Node-Red qui affichera les variables mesurées à partir de différents capteurs utilisés, à savoir le capteur de température et d'humidité DHT11, le capteur de température LM35, le capteur degaz MQ5, le capteur de pluie et la photorésistance, et lui permette grâce à sa plateforme programmer l'accès et l'interaction avec le système à distance.

Nous avons utilisé avec succès la technologie IoT dans l'automatisation des serres et la surveillance à distance avec la possibilité de détecter les mesures via message email et l'application Node Red pour ce projet, au sein de la CPG.

Il est important de mentionner que nous nous intéressons ici uniquement aux les facteurs climatiques et non à la croissance des plantes.

Dans le cadre ce notre projet, nous devons réaliser une serre automatique qui a pour fonctionla gestion des différentes tâches qui ont lieu dans le milieu agricole tout en offrant un climat favorable à la culture.

Résumé

Pour des raisons pratique nous allons faire une relation entre le contrôle automatique de la serre et le contrôle semi-automatique. Le contrôle automatique permet au système de réagir d'une manière intelligente pour établir les conditions climatologiques idéale pour les plantes comme le contrôle de température humidité, ce type de contrôle est basé sur la programmation de la carte Raspberry pour interagir avec l'interface de Node Red. Le contrôle semi- automatique est le mode de contrôle alternatif où l'utilisateur peut se renseigner sur les conditions climatologiques intérieure des serres via une interface graphique.

Mots clés : Serre intelligente, Réseau, Node RED, Raspberry PI.

Abstract

For practical reasons we will make a relationship between the automatic control of the greenhouse and the semi-automatic control. The automatic control allows the system to react in an intelligent way to establish the ideal climatological conditions for the plants such as the temperature humidity control, this type of control is based on the programming of the Raspberry card to interact with the Node interface Red. Semi-automatic control is the alternative control mode where the user can learn about the indoor climatological conditions of the greenhouses via a graphical interface.

Keywords: Smart greenhouse, Network, Node RED, Raspberry PI.