REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté *des Sciences et de la technologie*

Département ELECTROMECANIQUE

Mémoire

Présenté pour obtenir

LE DIPLOME DE MASTER

FILIERE : **automatique**

Spécialité : AUTOMATIQUE ET INFORMATIQUE INDUSTRIEL

Par

- **GOUADER TAHANI**
- **NAGHAR HANANE**

Intitulé

CONCEPTION D'UNE CARTE DE COMMANDE A DISTANCE POUR LES PIVOTS D'IRRIGATION

Soutenu le : 25/06/2022

Nom & Prénom	*Qualité*	*Etablissement*
M. RIAD KHENFER	*Encadreur*	*Univ-BBA*

Année Universitaire 2021/2022

Remerciement

Nous tenons à exprimer notre reconnaissance à Monsieur **(KHENFER RIAD)** qui a bien voulu diriger ce travail de recherche.

Nous lui présentons nos vifs remerciements pour sa disponibilité et ses conseils pertinents qui ont aidé de façon très significative à l'amélioration de ce mémoire.

Nous remercions l'ensemble des professeurs et animateurs du Département électromécanique.

Nous présentons aussi nos remerciements à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce travail.

Remerciement

Nous exprimons nos sincères remerciements et notre gratitude au vertueux professeur « **Belazoug Khaled »** qui nous a aidé Il a fait un excellent travail en complétant cette étude et il ne nous a pas épargné ses précieux conseils et orientations, alors que Dieu le récompense par le bien.

Nous n'oublions pas non plus d'exprimer nos plus profonds remerciements et notre gratitude à monsieur le **PDG** et tous les employés de la Fondation « **ANABIB IRRAGRIS** » Bordj Bou Arreridj, pour le bon accueil que nous avons reçu et les précieuses informations qu'ils nous ont fournies, pendant la période de stage de 45 jours.

Sommaire

Table des figures

Liste des tableaux

الملخص:

محتوى هذا العمل هو تصميم بطاقة تحكم عن بعد لمحاور الري، وتتكون هذه البطاقة من اردوينو نانو مبرمج في IDE وتردد راديو من نوع nRF24L01 ، من أجل حل مشكلة شركة ANABIB IRRAGRIS برج بوعريريج.

الكلمات المفتاحية: Arduino (uno-nano) - nRF24L01 - LoRa Ra - 02 SX 1278 - IDE- الري المحوري.

Résumé :

Le contenu de ce travail c'est la conception d'une carte de commande à distance pour les pivots d'irrigation, cette carte consiste d'une arduino nano programmé en IDE et un radon fréquence de type nRF24L01, afin de résoudre de problème de la société ANABIB IRRAGRIS Bordj Bou Arreridj.

Mots clé : Arduino (uno-nano) - nRF24L01 – LoRa Ra – 02 SX 1278 – IDE- pivot irrigation.

Abstract :

The content of this work is the design of a remote control card for irrigation pivots, this card consists of an arduino nano programmed in IDE and a radio frequency type nRF24L01, in order to solve the problem of the company ANABIB IRRAGRIS Bordj Bou Arreridj.

Keywords: Arduino (uno-nano) - nRF24L01 – LoRa Ra – 02 SX 1278 – IDE- pivot irrigation.

Introduction Générale

L'agriculture est un mot qui comprend toutes les activités de base pour la production d'aliments de fibres qui nécessitent les techniques nécessaires à la préservation des plantes et des animaux.

C'est un vaste domaine qui s'est longtemps étendu bien au-delà des méthodes traditionnelles. On sait à notre époque que l'agriculture est le pilier principal et le premier pilier sur lequel repose la sécurité alimentaire et nationale, et le plus grand soutien de l'économie de tout pays, que ce soit dans le produit intérieur brut pour atteindre l'autosuffisance, en particulier aussi l'exportation externe.

Et après que la technologie est intégrée dans le domaine de l'agriculture, les rapports ont changé entre le passé et le présent, après que les régions du Moyen-Orient, l'Égypte, le sud de la Chine, la côte africaine, l'Inde et les régions américaines aient été parmi les régions les plus importantes dans lesquelles l'agriculture a été active pendant des milliers d'années, et avec le début du XVIIIe siècle, le monde a atteint le stade de la révolution agricole qui visait à :

- La recherche des moyens réalisables pour augmenter la production.
- Exploitation des terres agricoles pleinement et aux moindres coûts.
- Développement du concept de rotation des cultures.
- La capacité de découvrir de nouvelles méthodes, moyens et compétences agricoles, comme l'agriculture sous serre et les maisons avec et sans sol.
- Hybridation et manipulation génétique.
- Meilleure gestion des éléments nutritifs du sol et contrôle des mauvaises herbes.
- Le plus important est la culture de surface, en particulier dans les zones sèches et semi-arides

L'impact de la technologie sur la modernisation de l'agriculture n'a été que des compétences qui se sont développées au fil du temps jusqu'à ce qu'elle prenne la forme que nous voyons aujourd'hui, et puisque la terre contient 41 % de la zone sèche, elle est connue pour sa faible productivité agricole d'une saison à l'autre, car la quantité de pluie

est le principal déterminant du succès de cette zone et la stabilité de la production dans ces zones doit être la suivante :

- Gérer les ressources naturelles, en particulier l'eau et la terre, en suivant les systèmes d'irrigation.
- Mise en œuvre du système d'agriculture de conservation.
- Développer des systèmes agricoles irrigués et améliorer l'efficacité de l'utilisation de l'eau.
- L'application de cultures alternatives qui nécessitent moins d'eau, supportent l'effort et sont rentables.

La technologie était à son apogée lorsque l'agriculteur était capable d'atteindre les rendements qu'il souhaitait pour pouvoir faire revivre les zones sèches et semi-arides, mais les obstacles surgissent toujours. Les agriculteurs qui manquent de ressources adoptent rapidement des innovations à faible coût et adaptées à la situation. Et quand on parle d'irrigation, on dit toujours que c'est un dur labeur manuel qui prend beaucoup de temps et qui consomme de grandes quantités d'eau, et pour s'assurer que nos plantes restent vertes et saines, et pour gagner du temps, introduire l'automatisation et les systèmes embarqués sont indispensables, et pour renforcer la relation entre l'université et le secteur socio-économique, le sujet de ce mémoire est un projet de collaboration entre l'université de bordj Bou Arreridj et la société ANABIB.

Le bute de ce travail et de concevoir une solution sans fil pour la surveillance des défauts dans les pivots d'irrigation, pour ce la on a choisi la carte ARDUINO UNO pour faire le contrôle et la commande a distance par une communication sans fil radio fréquence.

En fin notre travaille facilite la vie humaine et soulève le problème d'irrigation dans ces pays, donc l'Automatique à prouver sa présence dans tous les domaines.

On va résumer notre projet en 3 chapitres principaux :

- ✓ Le 1er chapitre généralité sur les systèmes irrigations et leur étude.

Introduction Générale

- ✓ Le 2éme chapitre présentation de la carte ARDUINO et les différents modules de communication RF.
- ✓ Le 3eme chapitre la partie de traitement et programmation du système.
- ✓ Et enfin, une conclusion générale sanctionnera l'ensemble de ce travail.

ChapitreI Généralité sur Les systèmes d'irrigation

ChapitreI : Généralité sur Les systèmes d'irrigation

I.1 Introduction

Pendant longtemps, l'homme a fait face aux obstacles de l'agriculture pour augmenter sa production, en particulier dans les zones sèches, où il s'est installé dans des zones proches des sources d'eau pour faciliter le processus d'approvisionnement en eau des terres agricoles par des méthodes traditionnelles.

Au fil du temps, de nombreuses techniques et méthodes sont apparues qui ont éliminé ces problèmes et ont permis aux agriculteurs et aux paysans d'investir des terres et de construire de grandes surfaces et de les exploiter en plantant divers types de plantes et en atteignant une autosuffisance à un niveau acceptable, et parmi ces récente méthodes est l'aspersion.

Dans notre pays l'Algérie, la société ANABIB a eu un impact en facilitant le processus d'irrigation à travers le pivot d'irrigation, qui est un système moderne, performant et efficace et outil, donc c'est quoi l'irrigation ?et le pivot d'irrigation ?

On va les apprendre dans ce chapitre

I.2 Les systèmes d'irrigation

Les systèmes d'irrigations peuvent être classés en deux grandes catégories:

L'irrigation gravitaire et l'irrigation sous pression. Dans la pratique, on distingue l'irrigation gravitaire, l'irrigation goutte à goutte et l'irrigation par aspersion. [1]

Figure I.1 : l'irrigation

Il est toujours nécessaire de mener des études sur l'état du sol et de la culture agricole à irriguer en termes de ses besoins en eau pour assurer une répartition adéquate de l'eau aussi estimation des coûts, bénéfices et avantages de chaque méthode. A partir de ce schéma, nous apprendrons les différentes méthodes d'irrigation et la différence entre eux.

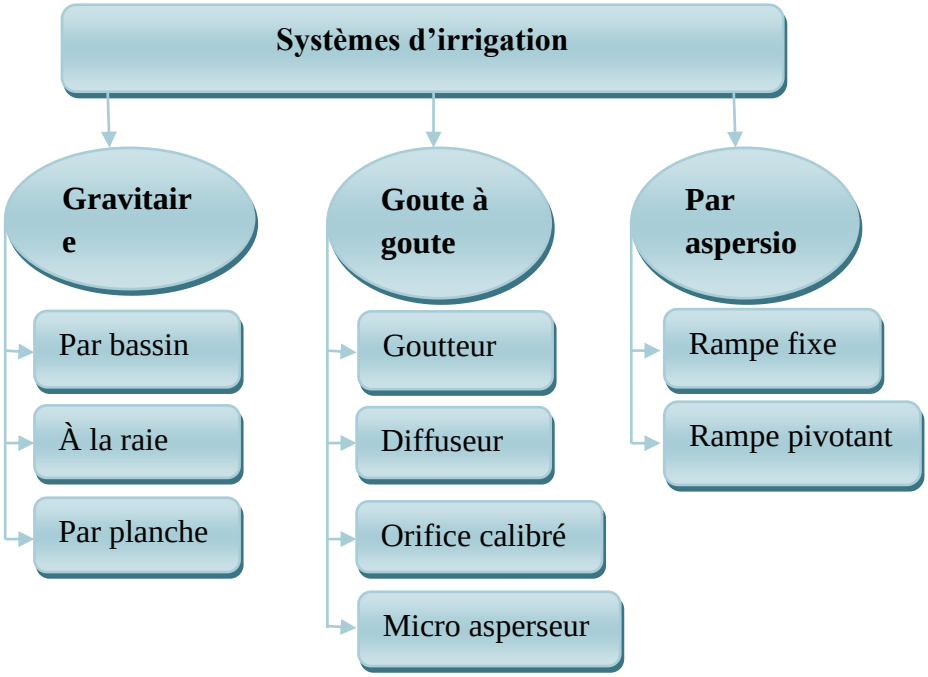

Figure II.2 : les grandes catégories de système d'irrigation

Il est toujours nécessaire de mener des études sur l'état du sol et de la culture agricole à irriguer en termes de ses besoins en eau pour assurer une répartition adéquate de l'eau aussi estimation des coûts, bénéfices et avantages de chaque méthode. A partir de ce schéma, nous apprendrons les différentes méthodes d'irrigation et la différence entre eux.

I.2.1 Irrigation de surface

Figure III.3 : Irrigation gravitaire

C'est la méthode la plus ancienne utilisée par nos ancêtres en raison du manque de développement intellectuel que l'homme a à notre époque, On peut dit irrigation gravitaire, il s'agit de recueillir l'eau en hauteur puis de la laisser s'écouler par gravité pour la répartir sur les champs ,Cette méthode se caractérise par le faible coût de sa préparation, et nous pouvons reverser l'eau de pluie dans les canaux et l'utiliser pour l'irrigation.

Quant à ses inconvénients, c'est en l'absence d'un système intelligent qui le contrôle, que les inondations peuvent complètement gâcher ce système, obligeant l'agriculteur à rester des heures supplémentaires dans son travail pour surveiller le processus d'irrigation.

Donc cette catégories peut faire par :

- Submersion (irrigation par bassins).
- Dans des sillons en terre (irrigation a la raie).
- Ruissellement à la surface d'une planche d'arrosage (irrigation par planches).

I.2.2 L'irrigation goutte à goutte

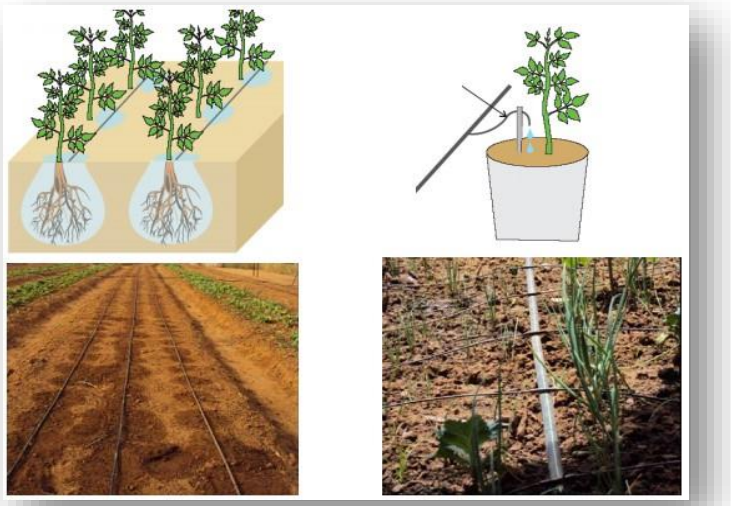

Figure IV.4 : Irrigation goute a goute

Cette méthode utilisé depuis 1970 et prévalent dans nombreux pays de premier monde, notamment l'Espagne et l'Italie puis dans les autres pays, est développé en 1995 par les indiens parce que ils sont parmi les pauvres peuple au monde donc ils ont préféré les pauvres moyens de technologie à faible cout et haut production, est une méthode dans laquelle l'eau est fournie à la plante en quantités égales et lentement sous forme de points, elle est utilisée dans l'irrigation des cultures maraîchères, des forêts et des arbres d'ornement.

L'un de ses avantages est la croissance de plantes saines sans plantes nuisibles, il convient aux zones inclinées afin que l'eau ne s'écoule pas à un niveau bas. Il peut également fournir un arrosage aux plantes fertilisées en même temps.

Et parmi ses inconvénients, coût de construction élevé pour l'utilisation d'un grand réseau de tuyaux qui sont plus susceptibles d'être endommagés en cas de présence de rongeurs, en plus de boucher les trous avec des sédiments et du sel.

Donc cette catégories peut faire par :

- Goutteur ou gain.
- Diffuseur.

- Orifice calibré ou ajutage.
- Micro asperseur.

I.2.3 Irrigation par aspersion

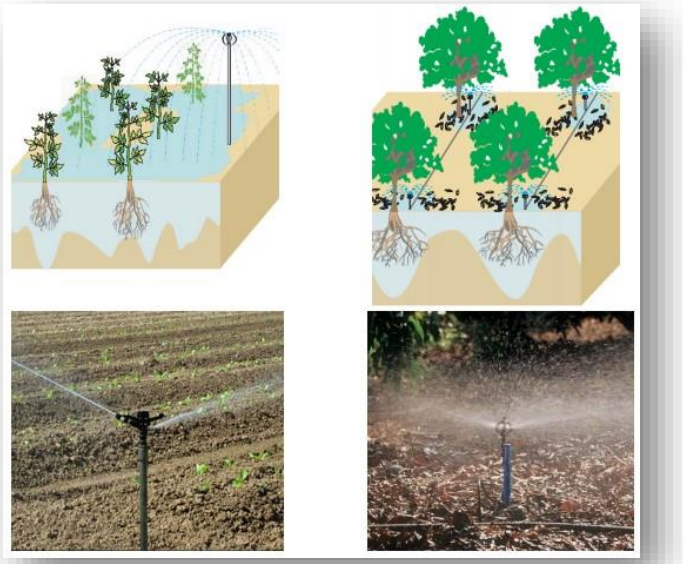

Figure V.5 : Irrigation par aspersion

La technique d'irrigation par aspersion est conçue sur le modèle de la pluie naturelle. L'eau est refoulée sous pression dans un réseau de conduites, ensuite elle est diffusée par des asperseurs rotatifs sous la forme d'une pluie artificielle [2], elle est appliquée dans les grandes terres agricoles de forme carrée ou rectangulaire régulière, et la forme ronde est la plus couramment utilisée.

Ses avantages sont qu'il est plus adapté aux zones désertiques en raison de la perméabilité du sol, qui peut retenir l'eau plus longtemps sans érosion du sol. Il n'a pas non plus besoin de consommer de grandes quantités d'eau avec la possibilité de fournir des engrais aux plantes, et sels avec irrigation en même temps.

Quant à ses inconvénients, il réside dans le coût élevé de la construction. Et le besoin de réservoirs d'eau, car les zones désertiques manquent de pluie, et l'entretien doit également être assuré et la main-d'œuvre doit avoir de l'expérience, avec une surveillance permanente.

Donc cette catégories peut faire par :

- Rampes fixe.
- Rampe mobile (pivotantes).

Aujourd'hui dans l'agriculture tous est moderne et la rampe mobile est la méthode la plus utilisée dans notre pays, parce que l'agriculture est nos or vert, nos terres arables constituent 5,5% de la superficie totale, dont 3,1% nos arables et 2,4% irriguées, dons nous cherchons à augmenter ce pourcentage en soutenant l'agriculture intelligente afin d'étendre les zones irriguées et ainsi augmenter la production nationale locale.

Parmi les dernières expérimentations algériennes dans ce domaine est l'expérience de l'agriculteur Louay Assem dans la région de Hassi Ramel dans la wilaya d'El Manea, qui vise à utiliser l'irrigation par pivot dans de grandes zones désertiques pour cultiver du blé et du maïs en s'appuyant sur une irrigation intensive et récolter les récoltes plantée en trois mois au lieu de six mois et sa première application était le 15 mars 2022 et la saison des récoltes est prévue à la mi-juin, tout en maintenant et en améliorant les spécifications du produit.

I.3 Quelques exemples d'étude d'irrigation

Au sein du périmètre du projet Allpamanta on compte 27 communautés, parmi lesquelles 6 communautés irriguent en gravitaire, une communauté en gravitaire et en aspersion, et 20 communautés en aspersion comme on peut le voir dans le Tableau 1 [3]

Méthodes	**Surfaces (ha)**	**Nombre de communautés**
Aspersion	4481,2	20
Gravitaire	1135,9	6
Aspersion + gravitaire	276,6	1
TOTAL	5893,7	27

Tableau 1:Techniques d'irrigation dans le périmètre étudié réalisé par Tamara Communal [7]

Les communautés irriguant en aspersion représentent donc les 3/4 des communautés de la paroisse de Cangahua, membres du système d'irrigation Guanguilquí-Porotog [3].

L’irrigation automatique gérée par WEM (Water mark Electronic Module) a permis de diminuer nettement la consommation d’eau par rapport à l’irrigation manuelle traditionnelle: 41 à 58 % d’économie selon l’année [4].

L’irrigation automatique a entraîné une augmentation moyenne du rendement de 13 % (l’étude dans Maroc « 2012-1016 ») sur les trois années d’essai, sans incidence significative sur le calibre des fruits [5].

I.4 Les pivots d’irrigation :

Nous continuons par la suite notre étude aux rampes pivotantes. Le pivot irrigation est un appareil d’irrigation automoteur qui arrose les prairies et autres cultures. Il se distingue des autres dispositifs par son fonctionnement circulaire ou sectoriel. Fixés à une extrémité, les pivots se déplacent en cercle autour de ce point central. Ils incluent des travées formées de tubes asperseurs qui sont montés sur une charpente et un jambage.

Ce dernier est doté d’un système de motorisation. Ils tournent autour de l’unité centrale pour assurer la gestion de l’eau dans une zone aride, que la parcelle soit petite ou grande [6].

Figure VI.6 : pivot d’irrigation pivot centrale

Figure VII.7 : champs circulaires irriguées par

C’est pivots inventée en Amérique la fin de 1940, et débuté en France en 1960, installé dans les zones plates et larges car ils sont très convenables pour arroser jusqu’à 200 ha, elle variait selon la forme du sol.

I.1.1 Types de machine d’irrigation :

- Arroseur géant pivotant : arrosant en cercle.
- Arroseur canons automoteurs : arrosant des bandes juxtaposées.

- Arroseur en rampe frontale : arrosant en rectangle

I.4.1 Choix d'un pivot

Pour choisir un pivot il faut mis en considération :

- Leur simple utilisation : réglage facile des pluviométries selon les cultures.
- Leur écologie : utiliser l'eau dont la plante a besoin sans gaspillage.
- Contrôlable à distance : la plupart des pivots moderne sont équipé par des systèmes de contrôle programmé, cela permet a l'agriculteur de jouer avec l'automatisme embarqué.
- Qualité de l'acier et l'épaisseur : des aciers de catégories S355 et S460 seront plus résistants face aux risques de déformation que peuvent entraîner les vents violents, tout comme des pivots équipés de **tube** set **cornières épaisses** seront également plus **stables** (un bon pivot est un pivot lourd !)[6].

I.4.2 Composants de pivots d'irrigation d'IRRAGRIS

Avant tout On va faire connaitre la société ANABIB et leur branche autour de l'Algérie.

D'abord, ANABIB est l'entreprise nationale de tubes et transformation de produit plats, elle travaille à rencontré tous les besoins nationaux en produits sidérurgique de 2 éme transformation depuis 2001, elle propose plusieurs produit dans le marché national algérienne, tels que les tube hydraulique, les tôles noir et galvanisée, les pivots d'arrosage et rampe d'aspersion, le rayonnage métallique et d'autre produits.

Contient 5 unités :

- IRRAGRIS : l'irrigation maitrisée par aspersion et pivot de 1 à 50 ha (Z.I.Route de M'sila-Bp 501-Bordj Bou Arreridj).
- ALTULET : Tube métallique, Energie et Hydraulique (Route national N°5 Z.I B.P 11 Réghaia ALGER).
- PTS : Unité Petits Tube Soudé (Route nationale N°5 A.I B.P 13 Réghaia-ALGER).

- PTTP: Unité Petits Tube Soudé et Tôles profilées (Z.I de Hassi Ameur B.P 23 ORAN)
- PAF : Unité Profilées A Froid (Z.I de Réghaia B.P 16 Réghaia-ALGER)

D'après cette schéma nous aprons ces composants :

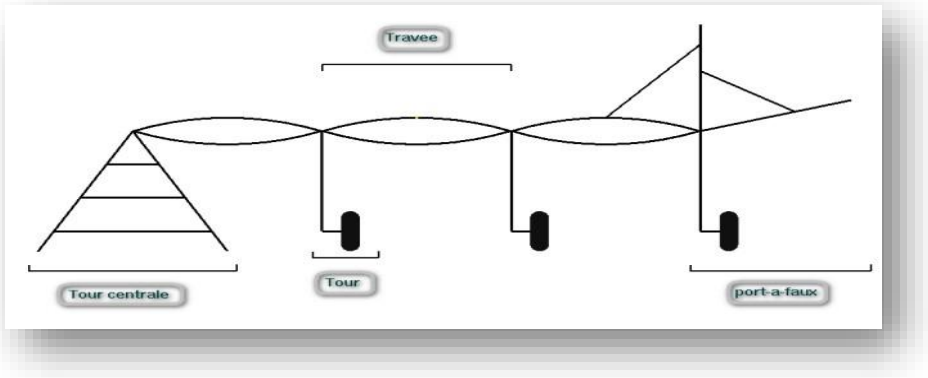

Figure VIII.8 : Schématisation d'une rampe pivotante

- La centrale de commande

Est une armoire contient des composants électrique permis de contrôlé et préservé les pivots par coupé l'arrosage dans le cas ou déformation anormale sur a une travée.

- La tour centrale

Est l'axe vertical autour du quel le pivot tourne , il est équipée par un tube d'alimentation en eau fixe accordé par la canal servant de palier pour le mouvement rotatif, et d'une base de 4 support bétonné , aussi d'un anneau collecteur qui collecte et protégé les câbles électrique de ne se rompt pas quand le pivot est en marche.

Figure IX.9 : la commande et la tour centrale

- La travée charpente

Est un axes sous forme d'un pont qui offre une grande stabilité contre les vont violentes, il se compose de réseau d'entré et d'équerres de traverses supportant les tubes.

Figure X.10 : la travée charpente d'ANABIB

- **La tour mobile**

Equipé par des roue de type **«** High flottaison **»,** anti-dérapent dans le sol humide.

Figure XI.11 : La tour mobile d'ANABIB

- Accouplement

Est le système mobile entre 2 travées reliées par rotule mobile permet de formé un angle de 30°.

Figure XII.12 : rotule mobile de pivot d'une rampe

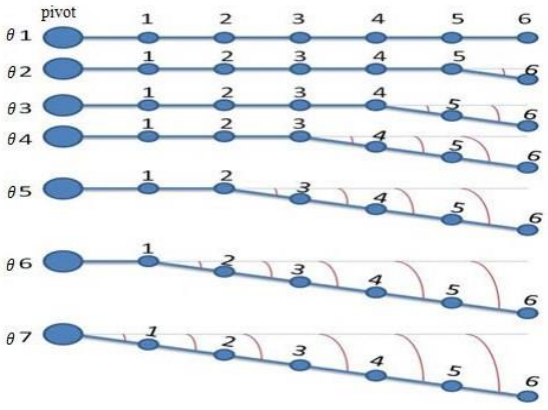

Figure XIII.13 : Principe d'avancement

- Porte a faux

Par conséquence, l'irrigation par aspersion continue facilement leur dernière tour, équipé par un dessableur pour purge, et un canon d'arrosage à ses extrémités.

Figure XIV.14 : Porte a faux de pivot

I.5 Caractéristique technique du système pivot ANABIB 160El

Parmi les différents produit d'ANABIB IRRAGRIS le modèle de la rampe pivotante 160El, il se trouve en 3 modèles :

- 160El (20 ha)
- 160El (30 ha)
- 160El (50 ha)

On va prend le El160 (30 ha) comme un exemple pour découvrir leurs caractéristique, illustré dans le tableau si dessous :

Paramètres	Caractéristique
Longueur du système (6 tours mobiles)	324m
Surface arrosée avec arrosage au-delà de bords	32.96 ha
Type de travée	52,0 m
Nombre des tours mobiles	0 6
Diamètre de tube	160,0 mm
Hauteur de construction	3,7m
Hauteur libre sous charpente	2,9 m
Hauteur moyenne de buses par rapport au sol	1,30m
	0,74 KW
	2954 KP

Propulsion par moteurs électriques	14.9-24
Poids avec de l'eau	NELSON
Type de pneus	11,7 m
Type de buse	2,6 m
Longueur du porte-à-faux	314,6 m
Arrosage au-delà des bords	24.0 h
Rayon arrosé avec arrosage au-delà des bords	83,2 m3/h 123 m/h.
Heures de service proposées par jour	12h 10min
Débit total	4-6 mm
Vitesse maximale	1,0 bar
Temps minimum de rotation à vitesse maxi	2,0 bar
Pluviométrie mini par rotation	0,93 a 1.39 L /s/ha (soit 8 a12 mm/ j)
Pression à la buse au dernier branchement	69,6 m3/h- 104,4 m3/h
Pression d'alimentation à l'entrée du pivot	380 v
Capacité d'arrosage	50 Hz
Consommation total le en eau	10,5 KW
Tension	7,2 KW
Fréquence	
Puissance absorbe du réseau public	
Puissance de groupe électrogène requise	

Tableau 2:caractéristique techniques système pivot ANABIB 160 EL(30 ha)[7]

I.6 Conclusion

Dans ce chapitre nous donnons un aperçu de touts les systèmes d'irrigation et les pivots qu'elle considère la société ANABIB IRRAGRIS parmi leur plus grande préoccupation, car lorsqu'une organisation comme elle met toute énergie, technologie et l'expérience qu'elle possède dans ce domaine afin de fournir des produits de qualité et efficacité et obtenir le rendement le plus élevé possible.

ChapitreII

Présentation de la carte ARDUINO et les différents modules de communication RF.

II.1 Introduction:

En raison du développement dans le domaine de la technologie et de la tentative de réduire tout composant puis de développer une carte électronique par l'équipe de développeurs {Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino, David Mellis et Nicholas Zambetti} cette dernière s'appelait arduino associée à une interface USB qui permet de la programmer, et tour à tour Dans ce chapitre, nous avons détaillé cette carte et ses types, caractéristiques et avantages, tout en abordant les moyens de communication RF.

II.2 La carte Arduino :

C'est un petit ordinateur qui peut mieux interagir et contrôler l'environnement qui l'entoure qu'un ordinateur de bureau. Techniquement, c'est une plate-forme open source qui se compose d'un contrôleur électronique (microcontrôleur) et d'un environnement de développement intégré pour l'écriture de logiciels IDE. La force d'arduino se manifeste par sa grande capacité à communiquer avec d'autres composants électroniques ou interrupteurs ou capteurs et à tirer parti notamment de l'obtention de diverses données. Les projets Arduino peuvent être exécutés en le connectant à un ordinateur et en le faisant traiter avec l'un des programmes de l'appareil, ou il peut s'exécuter indépendamment.

II.3 Types d'Arduino :

En fait, il existe plus de 40 types de types Arduino qui se différencient par leurs capacités, leur forme, leur taille et leur prix, Parmi eux : **Arduino Uno / Arduino Nano**

II.3.1. Arduino Uno :

Ce type est considéré comme le plus largement utilisé, comme le montre l'image ci-dessous, il se compose d'un groupe d'éléments électroniques que nous expliquerons

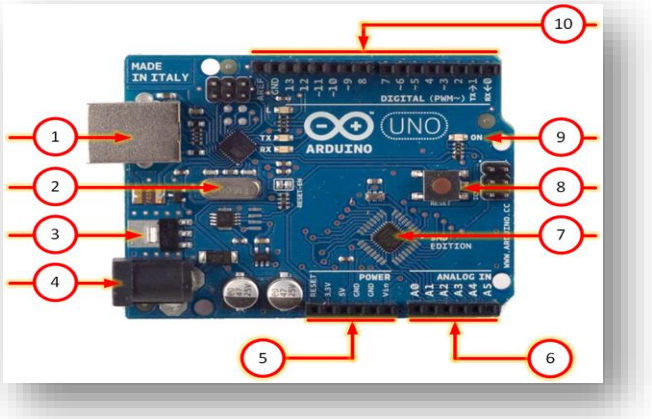

Figure XV.15 : carte arduino Uno et ces éléments

1. Le port USB de type B est connecté à l'ordinateur, et il alimente la carte mère en alimentation (énergie), ainsi qu'à travers lequel le microcontrôleur peut être programmé.
2. Un cristal Oscillateur est utilisé pour générer des impulsions à une fréquence de 16 mhz.
3. Il est utilisé pour s'assurer que la tension fonctionne dans une plage spécifiée.
4. Un port pris de courant de type Barrel jack, il est recommandé que l'entrée soit comprise entre 7 et 12 volts.
5. Un groupe composé de 6 prises reliées au secteur, qui sont les suivantes :
6. **.** Microcontrôleur de type Atmega 328.
7. Bouton permettant de relancer le programme.
8. LED indique que l'alimentation fonctionne correctement.
9. Un groupe de 16 ports, que nous expliquons dans les points suivants :

- Les ports 0 à 13 sont utilisés comme ports d'E/S numériques et ne peuvent pas être utilisés avec des signaux analogiques.
- Les ports avec le symbole z peuvent être utilisés comme sortie PWM et ce sont 6 ports.
- Les ports 0 et 1 en plus de servir d'entrée et de sortie du signal numérique peuvent être utilisés avec Serial interface

Vin : Un autre port utilisé pour alimenter la carte mère Arduino à partir d'une source externe.

GND : Un port qui peut être utilisé pour connecter la terre d'un appareil externe à la terre de l'Arduino, et cela se fait si cet appareil a été alimenté par la carte mère de l'Arduino ou par la même source à partir de laquelle l'Arduino a été pris.

5V : Une source de charge pouvant produire 5V peut être utilisée pour alimenter des appareils externes.

3,3 V : Une source de charge pouvant produire 3,3 V pouvant être utilisée pour alimenter des appareils externes.

Réinitialiser : utilisé pour redémarrer le programme.

AREF : utilisé pour déterminer la tension la plus élevée pouvant être reçue d'un signal analogique, et AREF est l'acronyme d'Analogue Reference.

- Caractéristiques de l'Arduino Uno :

Microcontrôleur :	**Microchip Atmega**
Tension de fonctionnement :	5v
Tension d'entrée :	7-20 V
Ports d'E/numériques :	14 (dont il peut fournir une sortie pwm).
Ports d'entrée analogique :	**6**
Courant continu par broche Ino :	20 ma
Courant continu pour 3.3V :	50ma
Mémoire Flash :	32 Ko dont 0,5 Ko est utilisé par le chargeur de démarrage
SRAM :	2 Ko
EEPROM :	1 ko
Vitesse de traitement :	16 mhz
Longueur	68,6 mm
Largeur :	53,4 mm
Poids :	25 grammes

Tableau 3:caractéristique d'un microcontrôleur Microchip Atmega d'arduino Uno[14]

Figure XVI.16 : carte arduino Uno

II.3.2. Arduino Nano :

C'est une petite carte intégrée qui contient un microcontrôleur qui a les mêmes fonctions et la différence réside dans le nombre d'entrées et de sorties et la taille.

- Caractéristiques d'Arduino Nano:

Microcontrôleur	**Atmega 328**
Tension de fonctionnement :	5 V
Mémoire flash :	32 ko dont 2 ko
Sram :	2 ko
Fréquence d'horloge :	16 MHz
EEP ROM :	1 ko
Courant continu par broches d'E/S :	40 mA
Tension d'entrée :	7/12 v
Broches I/Q numériques :	22/607 qui sont PWM
Sortie PWM :	6
Consommation électrique :	19 mA
Taille du circuit imprimé :	18x45 mm
Poids :	7grammes

Tableau 4:caractéristique d'un microcontrôleur Atmega 328 d'arduino Nano [15]

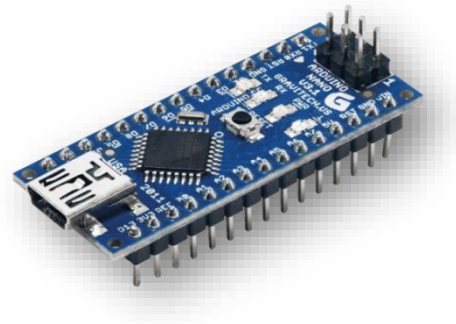

Figure XVII.17 : carte arduino nano

II.4 Bienfaits d'Arduino :

Il existe de nombreux microcontrôleurs électroniques disponibles sur le marché tels que Parallex, Basic Stamp, Netm Bx_24 Phidgets et Rasberry pi, qui ont tous de fortes capacités et ont la capacité de contrôler divers composants électroniques et logiciels, bien sûr, avec une préférence variable, mais ce qui distingue l'Arduino est un ensemble de choses qui font la différence dans sa structure Entre autres, les plus importantes d'entre elles sont :

- Simplicité

- Le prix
- Auto assemblage
- Multiplateformes.
- Environnement de programmation simple et facile.
- Logiciel open source :
- Matériel open source

II.5 Le Principe de fonctionnement :

1. On conçoit ou on ouvre un programme existant avec le logiciel arduino.

2. On vérifie ce programme avec le logiciel Arduino (compilation).

3. Si des erreurs sont signalées, on modifie le programme.

4. On charge le programme sur la carte.

5. On câble le montage électronique.

6. L'exécution de programme est automatique après quelques secondes.

7. On alimente la carte soit par le port USB, soit par une source d'alimentation.

8. autonome (pile 9 volts par exemple).

9. On vérifie que notre montage fonctionne. [8]

II.6 Logiciel Arduino :

Le logiciel qui permet de programmer votre carte Arduino porte le nom d'IDE, ce qui signifie "Integrated Développement Environnement" ou "Environnement de Développement Intégrale".

Le logiciel de programmation des modules Arduino, dont l'interface, appelée Arduino IDE, est une application Java, libre et multiplateforme dérivée de Processing servant d'éditeur de code et de compilateur, et qui peut transférer le firmware et le programme au travers de la liaison série (RS-232**,** Bluetooth ou USB selon le module). Il est également possible de se passer de l'interface Arduino, et de compiler et téléverser les programmes via l'interface en ligne de commande [9].

Le langage de programmation utilisé est le C++, compilé avec avr-g++ [11], et lié à la bibliothèque de développement Arduino, permettant d'utiliser la carte et ses entrées/sorties.

II.7 La radiofréquence (RF) :

La radiofréquence (RF) est le taux d'oscillation d'un courant électrique alternatif, d'une tension, d'un système magnétique, électrique, électromagnétique ou mécanique dans la gamme de fréquences d'environ 20 kHz à environ 300 GHz. Celle-ci se situe approximativement entre la limite supérieure des fréquences sonores et la limite inférieure des fréquences infrarouges ; Ce sont les fréquences auxquelles l'énergie d'un courant oscillant d'un conducteur peut être rayonnée dans l'espace sous forme d'ondes radio. Différentes sources définissent différentes limites supérieures et inférieures de la gamme de fréquences.[10]

II.7.1 Nrf24 :

Le module RF est l'un des composants les plus populaires et les plus polyvalents de l'Arduino. Le nRF24L01 est utilisé dans une grande variété d'applications nécessitant un contrôle radio, et c'est un émetteur et un récepteur, ce qui signifie que chaque module peut envoyer et recevoir des données, et ces unités sont peu coûteuses. Elles peuvent être utilisées avec n'importe quel MCU.

La puce nRF24 est un circuit intégré de chez Nordic Semi conducteur. En fait, il s'agit ni plus ni moins que d'un module d'émission / réception radio, opérant sur la bande des 2,4 GHz (comme le WiFi, ou le Bluetooth). Mais ce qui fait toute sa beauté, c'est qu'il se pilote le plus simplement du monde. En effet, un simple petit Arduino permet d'en prendre le contrôle, via le port SPI. [11]

Sachez que le nRF24 est décliné en 2 versions courantes, actuellement :

Le nRF24L01+, avec son antenne intégrée au PCB (reconnaissable avec ses pistes de cuivre en zigzag).

Le nRF24L01+ PA LNA, avec son antenne externe (raccordable avec un connecteur à visser, type SMA).

- Caractéristiques du NRF24L01 (avec ou sans antenne)

Ici, nous allons voir une à une les principales caractéristiques du NRF24L01+ (avec antenne intégrée) et du NRF24L01+ PA LNA (avec antenne externe). Nous verrons : la fréquence d'utilisation (canaux), la vitesse de pilotage via arduino (dataRate), le niveau d'émission, les

tensions d'alimentation et de communication, … et j'en passe. Comme vous l'aurez compris, il y aura pas mal de choses à voir ici [11]

Figure XVIII.18 : Le nRF24L01

- **Fréquence de transmission (2,4 GHz et plus)**

Le NRF24L01 est prévu pour fonctionner dans la plage de 2,4 GHz à 2,525 GHz, par « pas » de 1 MHz. Chaque pas de 1 MHz est ce qu'on appelle ici un canal. Du coup, on dispose ici de 125 canaux, permettant de communiquer sur la fréquence de notre choix, entre 2,4 à 2,525 MHz.[11]

N° DU CANAL	FRÉQUENCE DE TRANSMISSION
Canal 1	2,4 GHz
Canal 2	2,401 GHz
Canal 3	2,402 GHz
Canal 124	2,524 GHz
Canal 125	2,525 GHz

Tableau 5:fréquence de transmission des canneaux [11]

À noter qu'il y a d'autres appareils qui peuvent émettre ou recevoir sur ces fréquences. Je pense notamment au Bluetooth et au WiFi, qui eux aussi fonctionnent dans la bande des 2,4 GHz , Concernant le Bluetooth, il en va de même. En effet, les fréquences comprises entre 2400 MHz et 2483,5 MHz peuvent être utilisées, nous laissant donc « le champ libre » entre 2484 MHz et 2525 MHz.[11]

Même si vous êtes sur une fréquence « libre », il ne vaut mieux pas coller son émetteur récepteur NRF24L01 à proximité de sa box internet WiFi, ou de toute autre clé ou appareil Bluetooth ! Sinon, gare aux interférences et aux fonctionnements aléatoires [11]

- **Vitesse de transmission des données, par radio**

Le nRF24 permet la transmission de données à vitesse plus ou moins rapide, selon ses

besoins, et la portée que l'on souhaite atteindre. Par défaut, nous avons le choix entre trois valeurs possibles :

1. Vitesse de 250 kbps (vitesse la plus lente, qui « porte » le plus loin)
2. Vitesse de 1 Mbps
3. Vitesse de 2 Mbps (vitesse la plus rapide, qui « porte » le moins loin)

Ces valeurs sont exprimées en kbps (kilo bits par seconde), ou Mbps (méga bits par seconde). [11]

Et comme indiqué précédemment, vouloir émettre à grande vitesse a un inconvénient majeur : une plus faible portée de signal, et un plus grand risque d'erreurs, dues aux interférences. Il faut donc toujours adapter la vitesse de transmission de données à son besoin, et ne pas chercher à aller trop vite [11]

- **Niveau du signal**

Autre paramètre ajustable avec les puces nRF24L01 : le niveau de signal. En fait, nous aurons le choix entre plusieurs puissances d'émission possibles, allant de très faible, à particulièrement fort. Et le fait d'avoir une antenne intégrée ou une antenne externe fera toute la différence, si nous cherchons à porter loin. [11]

À noter également que certains nRF24 sont dotés d'amplificateur, noté « PA ». Ainsi, si vous voyez marqué « NRF24L01 PA LNA » sur votre module radio, alors vous saurez qu'il s'agit là d'un modèle avec amplificateur de puissance de signal (PA signifiant « Power Amplification »). Accompagné de la mention LNA, signifiant « Low Noise Amplifier », vous aurez là un des meilleurs émetteurs récepteurs qui soit, car hyper puissant, tout en étant le moins sensible possible aux perturbations environnantes. [11]

Concernant notre NRF24L01, voici des valeurs de rapport de puissance que j'ai pu trouver :

NIVEAU DE SIGNAL	NRF24L01+	NRF24L01+ PA	NRF24L01+ PA LNA
MINIMAL (min)	-18 dBm	-12 dBm	-6 dBm
BAS (low)	-12 dBm	-4 dBm	0 dBm
HAUT (high)	-6 dBm	1 dBm	3 dBm
MAXIMAL (max)	0 dBm	4 dBm	7 dBm

Tableau 6:les niveaux de signal [11]

Pour rappel, le « dBm » est en quelque sorte un comparateur de puissance, faisant le rapport entre une puissance mesurée et 1 milliwatt. Ainsi :

1. 0 dBm correspond à une puissance de 1 milliwatt (soit 1 mW, ou encore, 0,001 W)
2. Une augmentation de 3 dB correspond à un doublement de la puissance (soit 2 mW)
3. Une diminution de 3 dB correspond à une division de la puissance par deux (soit 0,5 mW)

Bien entendu, si vous souhaitez monter en puissance, faites très attention à ce que votre alimentation soit suffisamment puissante, et bien stabilisée. Sinon, vous risquez de voir apparaître des bugs ou problèmes de communication, comme c'est le cas lorsqu'on essaye d'alimenter ses montages directement sur 3,3 volts des Arduino [11].

Enfin, dites vous que côté consommation, le niveau de puissance va forcément influer sur le courant consommé.

Bien sûr, ce ne sont ici que quelques chiffres indicatifs, qui en réalité seront à pondérer, en pratique (car également fonctions de la quantité de données échangées et de la vitesse d'échange, en plus de l'amplification de puissance). [11]

- **Mémoire du NRF24L01+ :** limite à 32 octets par message

Petite parenthèse au sujet de la mémoire intégrée au nRF24L01 : cette puce n'intègre que 32 octets de mémoire, en émission comme en réception, et par canaux (sachant qu'un nRF24 peut gérer jusqu'à 6 canaux de communication simultanément). C'est pourquoi nous verrons souvent des variables limitées à 32 caractères dans les codes de programmes Arduino.]

Tunnels de communication (« Pipe ») : sens de communication, et adressage Comme évoqué précédemment, les nRF24 sont limités à 6 canaux de communication simultanés. Ceux-ci

s’appellent des « pipe », en anglais (pour ma part, je les appelle des « tunnel » de communication). [11]

Basiquement, un nRF24 ne peut communiquer qu’avec 6 autres nRF24 au maximum (mais nous verrons comment dépasser cette limite en fin d’article, en créant un réseau de NRF24L01). Mais attention, car sur ces 6 pipes, un seul est utilisable en émission/réception (les autres ne pouvant qu’écouter, et non émettre). [11]

Par ailleurs, il faut savoir que chaque pipe aura une adresse qui lui est propre, codée sur 5 octets (40 bits). Mais que seuls 2 pipes sur 6 sont encodées sur 5 octets. Car les 4 autres pipes n’auront qu’une « adresse » à un octet, qui sera en fait complétés avec quatre premiers octets pris sur la seconde pipe. Tout se résume avec le tableau suivant :

N° DU PIPE	NOM DU PIPE	PEUT ÉMETTRE / PEUT RECEVOIR	TAILLE DE L’ADRESSE
1	PIPE0X	X	5 octets (40 bits)
2	PIPE1	X	5 octets (40 bits)
3	PIPE2	X	1octets*
4	PIPE3	X	1octets*
5	PIPE4	X	1octets*
6	PIPE5	X	1octets*

Tableau 7:les tunnels de communication [11]

Chaque tunnel peut prendre le nom qu’il veut (ou presque !), dans la limite de 5 caractères alphanumérique (d’un octet chacun).). [11]

II.7.2 Module LoRa RA-02 SX1278

Le Ra-02 est un module de communication sans fil qui utilise la technologie LoRa

- **Description :**

Basé sur l'émetteur-récepteur Semtex SX1278, il adopte la technologie LoRa (long range radio) d'étalement du spectre qui permet des communications jusqu'à 10000m de distance. Ce module offre une bonne capacité anti-interférences et des fonctions d'économie d'énergie (air Wake up). La technologie LoRa permet de couvrir des milliers de personnes au sein des zones urbaines, quartiers, etc. cette technologie est fréquemment utilisée pour la gestion des systèmes

publics tels que l'éclairage, la surveillance de paramètres au sein de multiples zones, la mesure environnementale, la lecture des compteurs, maison intelligente, l'équipement d'alarme etc.

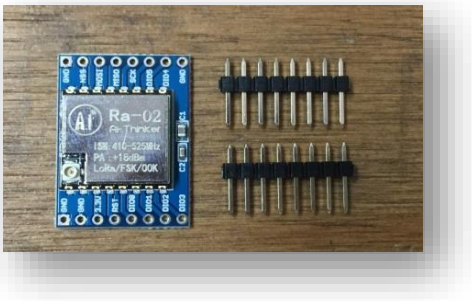

Figure XIX.19 : Module LoRa RA-02 SX1278

- **Caractéristiques de Module LoRa RA-02 SX1278**

Fréquence de fonctionnement:	**433MHz (420-450MHz)**
Sensibilité en réception:	-148dBm
Puissance maximale en sortie:	20dBm-10mW
Tension de fonctionnement:	1.8-3.7V (3.3V par défaut)
Consommation :	réception 10.8mA max/émission 120mA/Sleep mode 0.2µA

Tableau 8:Module LoRa RA-02 SX1278 [12]

La technologie LoRa est largement répandue dans le monde de l'internet des objets (IoT). Elle permet notamment la transmission de données de faible volume par des capteurs fixes. Le principe des réseaux LoRa est de transmettre des données par liaison hertzienne depuis des capteurs à faible puissance d'émission, potentiellement isolés ou difficile d'accès, fonctionnant sur batterie pour 5 à 10 ans. [12]

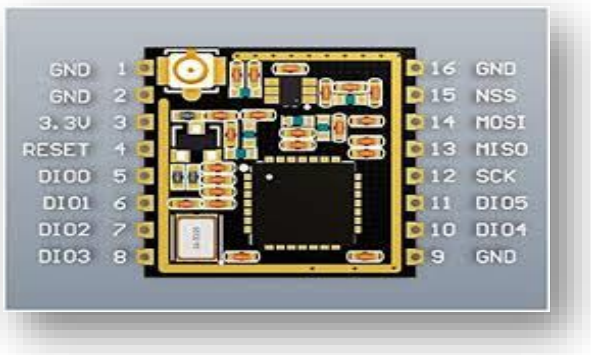

Figure XX.20 : Broches de module LoRa RA-02 SX1278

Fonctionnellement, les capteurs mesurent et stockent leurs données avant de les transmettre sur le réseau LoRa. Ils envoient leurs trames à une (ou plusieurs) Gateway(s) (ou concentrateur(s)) qui fait office de relais entre un grand nombre de capteurs dans son rayon de couverture et les serveurs de stockage et de traitement de données ad hoc. Gateway et serveurs sont interconnectés sur réseau cellulaire, en fibre optique, ou en liaison cuivre selon les cas de figure. [12]

Les liaisons bidirectionnelles sont possibles, c'est-à-dire capteurs à SI et SI à capteurs. Cette fonctionnalité peut avoir son importance s'il y a des mises à jour du firmware à effectuer par exemple. [12].

II.8 Conclusion :

On peut résumer ce que nous avons expliqué dans ce chapitre qu'avec une petite carte électronique appelée arduino et 2RF, on peut contrôler à distance n'importe quelle machine, qu'elle soit simple ou complexe, et corriger ses erreurs avec un programme facile et simple basé sur le langage c++ que tout débutant peut apprendre et développer sa programmation.

Arduino est vraiment une découverte créative qui a contribué au développement de la technologie et à faciliter les choses pour les programmeurs, surtout au prix.

ChapitreIII

La partie de traitement et programmation du système.

III.1 Introduction

Dans ce chapitre, un cahier de charge définit a partir d'un entretien entre l'université de MOHAMED EL BACHIR EL IBRAHIMI de Bordj Bou Arreridj et les ingénieurs de la société ANABIB IRRAGRIS, ils sont entravé face à un problème qui nécessite la réalisation d'une carte de communication radio fréquence pour détecter un décalage d'angle d'accouplement des rampes, donc cette carte de communication sans fils va nous informer dans le cas d'un décalage de 30°.

Après l'étude des régions équipées par les pivots d'irrigation, on a constaté que la plupart de ces systèmes sont installés dans des régions vastes et isolées (aucun mode de communication GSM ou WIFI …), pour cela on a opté pour une communication radio fréquence.

Pour atteindre notre but, on effectue plusieurs testes, le premier test consiste à établir une communication radio fréquence entre les nRF, puis on essaye de transmîtes une action TOR, par la suit un autre test d'envoie d'un paquet de données analogique. À la fin on va essaye de surmonté le problème de gourmandé de la consommation d'énergie électrique par l'implémentation d'un algorithme qui déclenche le mode sommeil de notre microcontrôleur, pour minimiser au maximum la consommation d'énergie de batterie. On va le résumé dans ce schéma qui présente touts les programmes que nous avons faire pour établir cette communication.

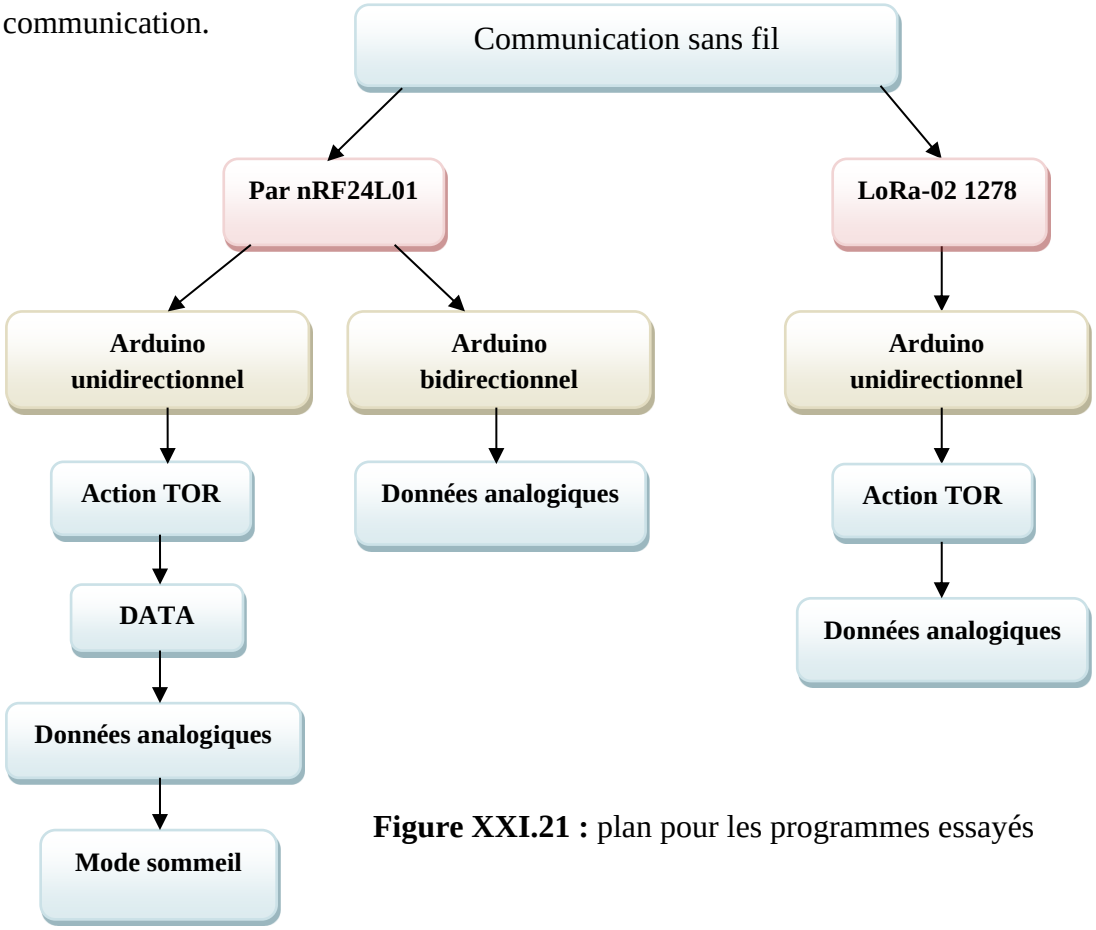

Figure XXI.21 : plan pour les programmes essayés

III.2 Communication sans fils par NRF24L01 (1 kilomètre)

On prend deux arduino de type Uno ou Nano et deux nRF24L01, on branche chacun des deux (chaque broche dans l'arduino a une broche spécifié dans le nRF24L01), donc on obtient deux cartes un émetteur et un récepteur, ces deux dernier va réalisons un bon réseau de communication sans fil dans l'absence des différentes obstacles.

D'abord, le premier pas était avec l'arduino et le NRF24L01, leurs caractéristiques, leurs datasheets et leurs programmes, ce pas nous permettons de nous perfectionner notre niveau et acquérir de l'expérience. L'étude sur logiciel arduino était très facile à cause de la simplicité de ses instructions.

III.2.1 visé sur le programme

Pour instancier la bibliothèque nRF dans l'arduino, il faut inclure quatre librairies, une de la carte arduino et les trois autres pour le NRF24L01.

```
#include <SPI.h>          // appel des bibliothèques pour le microcontroleur
#include <RF24.h>         // appel des bibliothèques pour le nRF24L01
#include <nRF24L01.h>     // appel des bibliothèques pour le nRF24L01
#include <RF24Network.h>  // appel des bibliothèques pour le nRF24L01
```

Figure XXII.22 : instanciation des bibliothèques

- on définie les pins CE et CSN du NRF24L01 associées au sorties digital d'arduino

```
#define pinCE   7        // On associe la broche "CE" du NRF24L01 à la sortie digitale D7 de l'arduino
#define pinCSN  8        // On associe la broche "CSN" du NRF24L01 à la sortie digitale D8 de l'arduino
RF24 radio(pinCE,pinCSN);// création de l'objet radio
```

Figure XXIII.23 : Définition des pins CE et CSN

- Ensuite, on fait l'appel de la fonction d'initialisation de NRF24L01 dans la void setup d'arduino qui n'est appelée qu'une seule fois et sert à affecter le mode aux broches ou aux commandes qui doivent être exécutées uniquement au moment du chargement du programme.

```
void setup() {
  radio.begin(); // Initialisation du module NRF24
  }
```

Figure XXIV.24 : Initialisation du NRF24

- On fait la sélection du canal de communication et le niveau de transmission du signal, est aussi la vitesse de communication

```
void setup() {
  radio.begin();                       // Initialisation du module NRF24
  radio.openWritingPipe(adresses[0]);     // Ouverture du "tunnel1" en ÉCRITURE
  radio.openReadingPipe(1, adresses[1]); // Ouverture du "tunnel2" en LECTURE
  radio.setPALevel(RF24_PA_MIN);          // Sélection d'un niveau "MINIMAL" pour communiquer
  Serial.begin(9600);                     // initialize serial communications at 9600 bps
 }
```

Figure XXV.25 : Fonctions de void setup nRF

```
#define tunnel1 "PIPE1"     // On définit un premier "nom de tunnel" (5 caractères), pour pouvoir envoyer des données à l'autre NRF24
#define tunnel2 "PIPE2"     // On définit un second "nom de tunnel" (5 caractères), pour pouvoir recevoir des données de l'autre NRF24
const byte adresses[][6] = {tunnel1, tunnel2};    // Tableau des adresses de tunnel, au format "byte array"
```

Figure XXVI.26 : Définition des tunnels de communication

- On définit le sens de communication entre l'arduino et le radio

```
radio.startListening();    // permet de pouvoir utiliser la fonction « read »
radio.stopListening();     // permet de pouvoir utiliser la fonction « write »
```

Figure XXVII.27 : Définition de sens de communication

Enfin, les ordres d'émission /réception des données sera exécuté si le radio est ouvert à l'aide de plusieurs fonctions dans la void loop.

```
if(radio.available()) {                 // On regarde si une donnée a été reçue
radio.read(&donnees, sizeof(donnees)); // pour charger la totalité des données reçues en une seule fois
radio.write(&donnees, sizeof(donnees)); //pour envoyer toutes les données de la « struct », à la fois
}
```

Figure XXVIII.28 : les ordres d'émission /réception

Pour afficher de résultat dans le moniteur on utilise ces fonctions

```
Serial.print( ); Serial.println( );//affiche les données dans le moniteur série
```

Figure XXIX.29 : instructions d'affichage de données

- On peut met le programme en pause pendant la durée de quelque millisecondes par l'instruction Delay

```
delay(ms); // attente en ms
```

Figure XXX.30: instructions de pause

III.2.2 Programmation de NRF24L0 unidirectionnel

Nous avons fait quelque expérience simple au début pour faire communiquer à une seule direction avec le nRF24L01, ca veut dire que la carte arduino va faire la gestion d'émission ou la gestion de la réception.

- Transmission d'une action TOR

Dans cette partie on établie l'envoie d'un message pour assurer que nous avons bien définit notre programme, on essaye d'envoyer une donnée de type caractère «HELLO HANANE », le schéma de câblages entre les deux carte ce faire avec le logiciel Fitzing.

Fitzing est un outil complet pour l'automatisation des processus de design électronique, avec lequel vous pouvez créer des schémas électriques, monter les prototypes sur une plaque d'essais virtuelle ou choisir le meilleur routage du circuit p pour construire le circuit imprimé, Il s'agit d'un outil parfait pour le domaine éducatif et professionnel, en étant très utile pour les professeurs en électronique, ingénieurs ou amateurs de l'électronique[13].

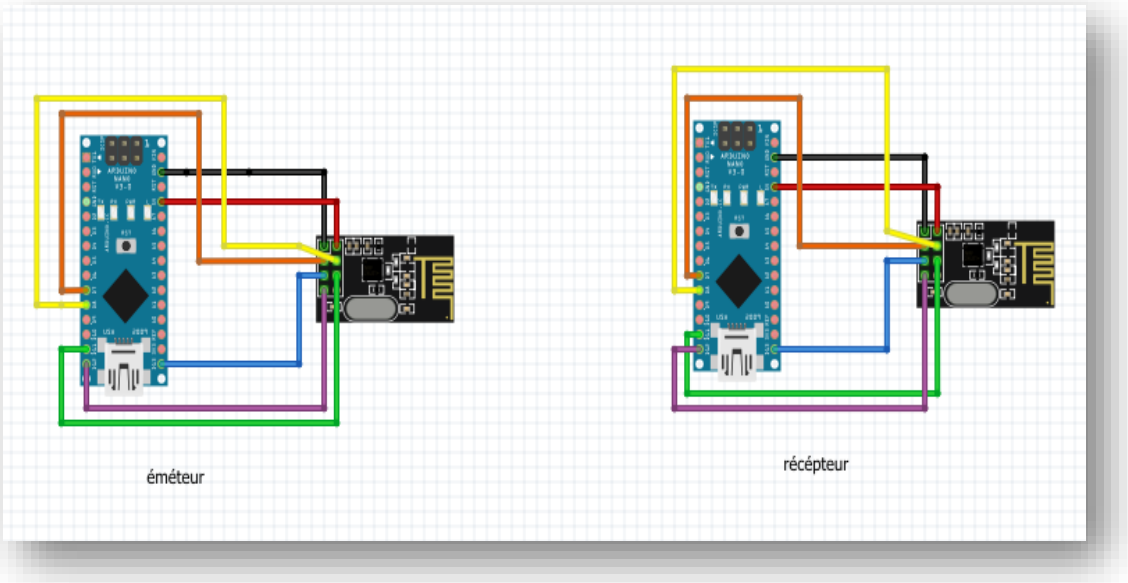

Figure XXXI.31 : schéma de câblage pour l'envoie des données TOR

Les instructions dans la void loop étaient comme suite :

```
const char message[] = "Hello HANANE !!!";     // Message à transmettre à l'autre NRF24 (32 caractères maxi, avec cette librairie)
```

Figure XXXII.32 : déclaration de type de message

- émission :

```
void loop() {
  radio.write(&message, sizeof(message));      // Envoi de notre message
  delay(1000);                                 // ... toutes les secondes !
}
```

Figure XXXIII.33 : void loop d'émetteur d'action TOR

- **réception :**

```
void loop() {
  // On vérifie à chaque boucle si un message est arrivé
  if (radio.available()) {
    radio.read(&message, sizeof(message));                  // Si un message vient d'arriver, on le charge dans la variable "message"
    Serial.print("Message reçu : "); Serial.println(message);     // ... et on l'affiche sur le port série !
  }
}
```

Figure XXXIV.34 : void loop de récepteur d'action TOR

Le moniteur série affiche les résultats d'exécution des données comme suite :

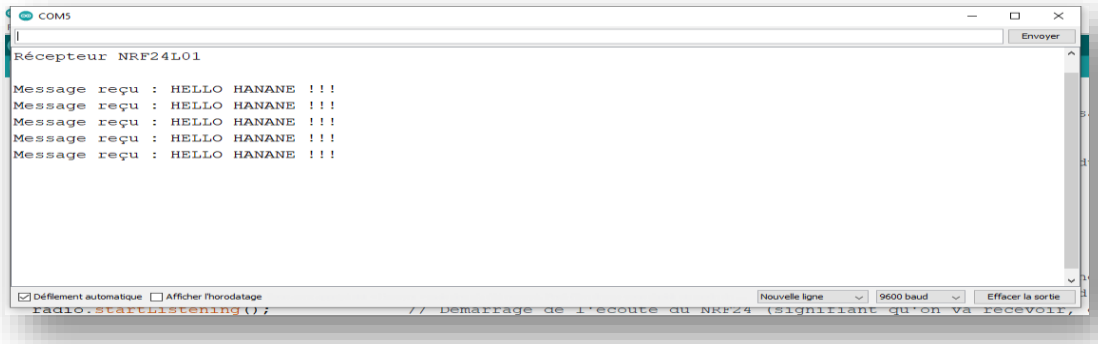

Figure XXXV.35 : moniteur série d'action TOR

Le câblage en réalité

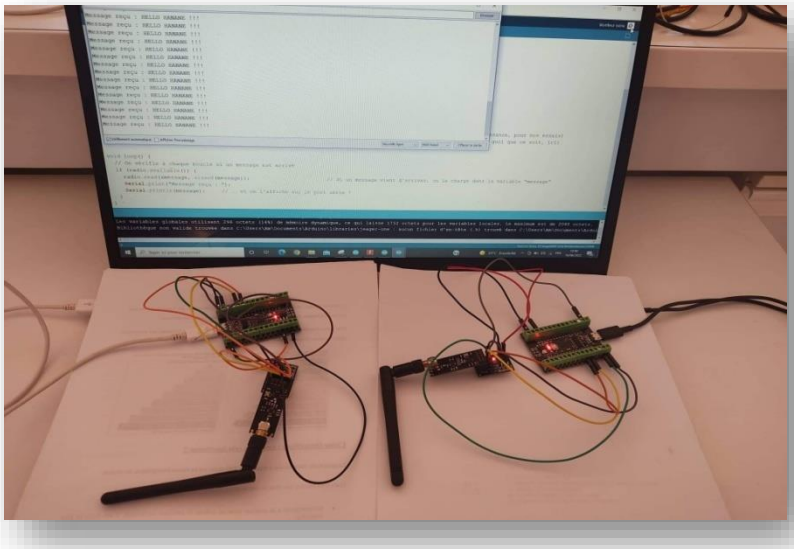

Figure XXXVI.36 : câblage réelle pour l'envoie des données TOR

L'envoie de paquet de données numérique (Data) :

Dans ce programme nous besoin d'envoyé un tableau de valeurs ou un paquet de données numérique, et le récepteur doit nous affiché le contenue de ce paquet de data dans le moniteur série, et allumer la Led si le signal est reçue sinon la LED va rester atteint, à partir de montage suivant :

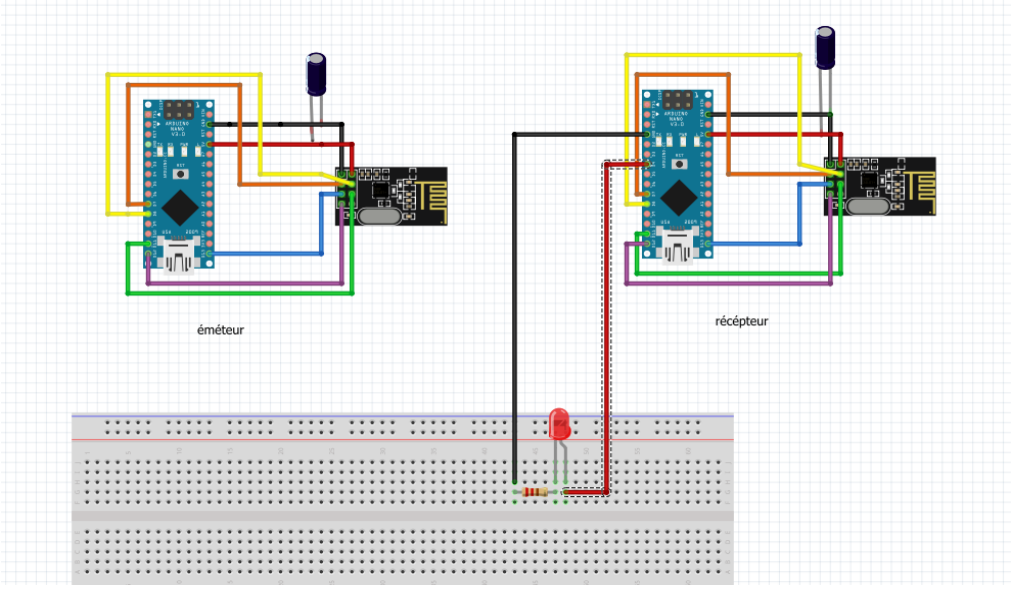

Figure XXXVII.37 : câblage des nRF pour l'envoie des DATA

- Emission : on définit le type de data

```
int data[5];// définit à 5 données
```

Figure XXXVIII.38 : Définition de type de Data

```
void loop()
{
  data[0] = 11;
  data[1] = 12;
  data[2] = 13;
  data[3] = 14;
  data[4] = 15;
  radio.write( data, sizeof(data) );   // envoi des données
  delay(4000);
}
```

Figure XXXIX.39 : void loup d'émetteur de Data

- Réception : On déclare le type et la broche associer à LED

```
const int ledPin =  4 ;      //brancher la led dan la pin 4
```

Figure XL.40 : déclaration du LED

```
void loop() {
  if ( radio.available() )            // si des données sont présentes
  {
    radio.read( data, sizeof(data) ); // lecture des données
    Serial.print("data[0]=");         // affichage dans le moniteur série
    Serial.println(data[0]);
    Serial.print("data[1]=");
    Serial.println(data[1]);
    Serial.print("data[2]=");
    Serial.println(data[2]);
    Serial.print("data[3]=");
    Serial.println(data[3]);
    Serial.print("data[4]=");
    Serial.println(data[4]);
        }
  else Serial.print("no data");       //si des données ne sont pas présentes affiche "no data"
  if ( data[0] == 11)                 //si cette condition est vérifiée
  {  digitalWrite(ledPin, HIGH);      //allumez la LED
   }
  else digitalWrite(ledPin, LOW);     //atteindre la LED
  delay(2000);                        //avec une pause
```

Figure XLI.41 : void loop de récepteur de Data

Le moniteur série affiche les résultats d'exécution des données comme suite :

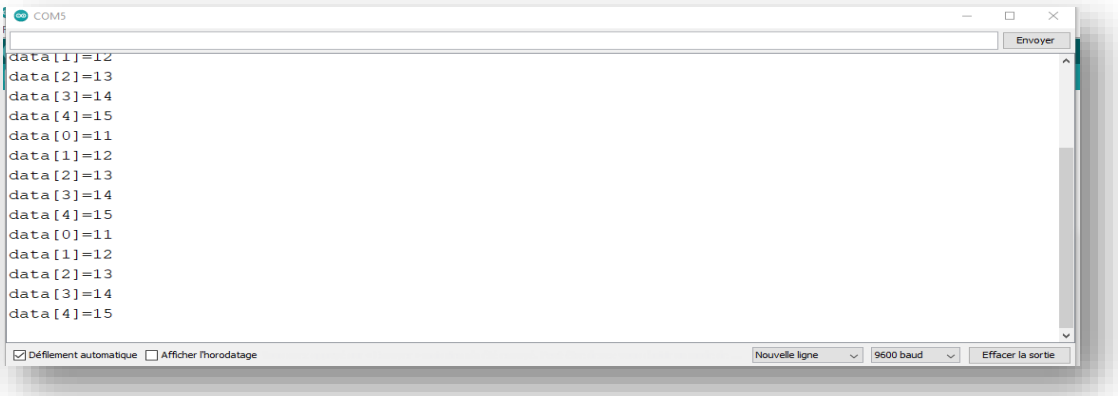

Figure XLII.42 : moniteur série de Data

Et quand la condition : [**if (data[0] == 11)]** est vérifié, la LED doit etre allumer

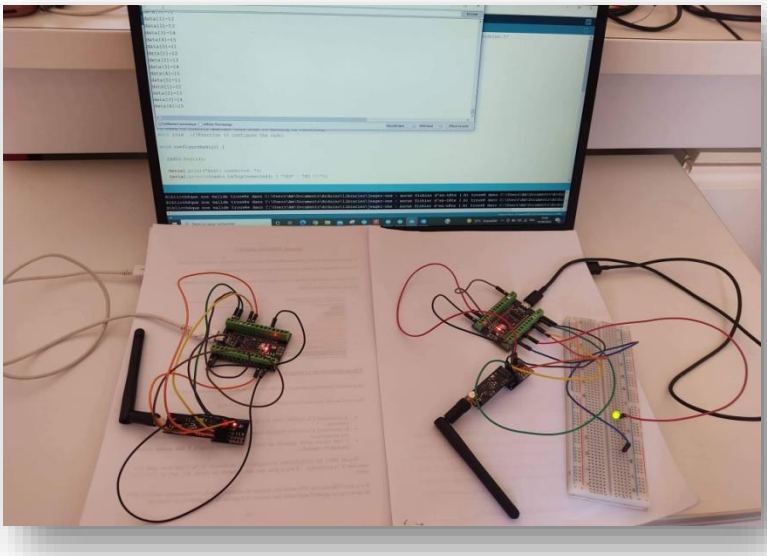

Figure XLIII.43 : câblage réelle des data pour allumer la LED

- L'envoie de paquet de données analogique **:**

Cette essaye est réalisé à cause de notre volonté d'utiliser un potentiomètre pour ajuster, régler divers paramètres dans un circuit électrique-puissance, tension, volume sonore, luminosité du led ou angle de servomoteur, Donc, on connecte correctement la résistance variable (potentiomètre) à l'arduino pour envoyer les données analogique. C'est quand on fait varie la valeur de résistance variable, l'émetteur va transmîtes les données vers le récepteur, et ce dernier va l'afficher dans le moniteur.Ce potentiomètre est alimenté en 5 volts et GND, la broche de centre est un contact mobile, quand on tourne le potentiomètre la tension du signal est varie entre le max et le 0.

Le schéma suivant va présente le montage pour réaliser cette essaye :

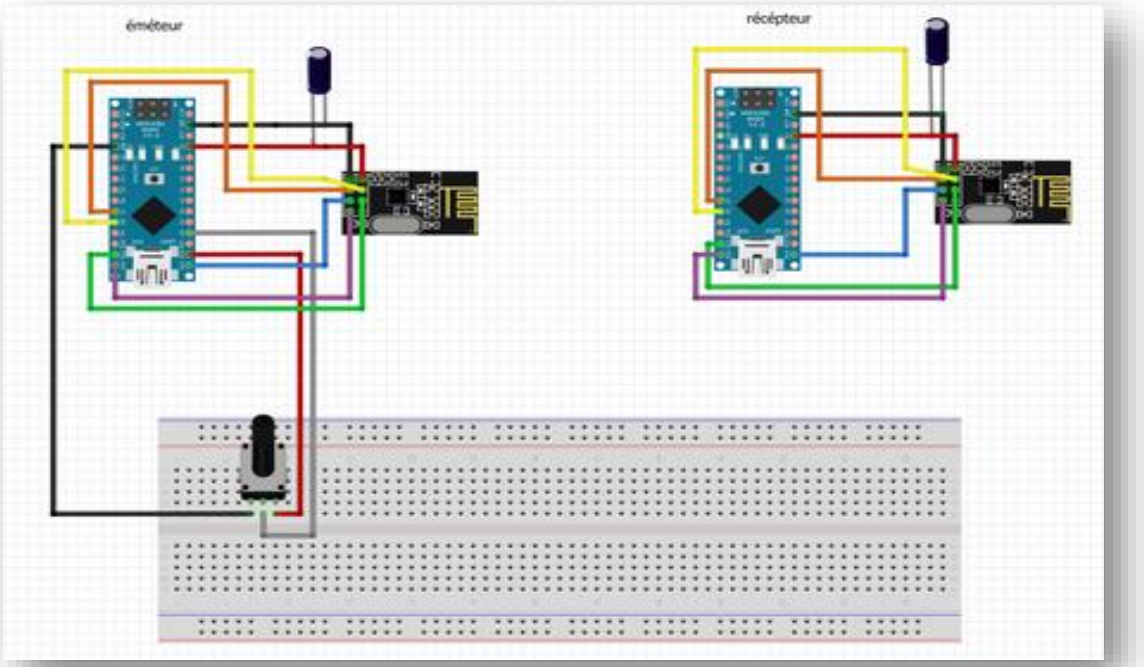

Figure XLIV.44 câblage pour l'envoie des données analogique unidirectionnelle

- **Emetteur : On déclare les différents variables qu'on besoin**

```
//déclaration des type des variables
#define pinPOT2    A0        // On associe le point milieu du potentiomètre à l'entrée analogique A0 de l'arduino
pinMode(pinPOT2, INPUT); // Déclaration de la pin "pinPOT2" en entréeint valeurPotLocal;
int valeurPotLocal;      // Variable contenant la valeur du potentiomètre
int valeurAnaloge;       //valeur de potentiométre envoyé
```

Figure XLV.45: déclaration des variables analogique d'émetteur

```
void loop() {
valeurPotLocal = analogRead(pinPOT2);                //lire la valeur de pinPOT2
valeurAnaloge = map(valeurPotLocal, 0, 1023, 0, 255); // On converti la valeur [0..1023] en [0..255]
radio.write(&valeurAnaloge, sizeof(valeurAnaloge));   // ... et on envoi cette valeur à l'autre arduino, via le NRF24
Serial.print("pot=");
Serial.println(valeurAnaloge);                       // affichage dans le moniteur série
delay(20);                                           // avec une petite pause, avant de reboucler
delay(1000);
}
```

Figure XLVI.46 : void loop d'émetteur des données analogique

- Récepteur

```
#define pinoutpot  9  // On associe la broche "SIGNAL" du SERVO à la sortie digitale D9 de l'arduino
int valeurpotLocal;  // définir le type de donné
```

Figure XLVII.47 : déclaration des variables analogique d'émetteur

```
void loop() {
  if(radio.available()) {                                    // On regarde si une donnée a été reçue
  while (radio.available()) {                                // Si une donné est en attente de lecture, on va la lire
  radio.read(&valeurAngleServoLocal, sizeof(valeurAngleServoLocal)); // Lecture des données reçues, une par une
  Serial.print(" pot2=");                                  // affichage dans le moniteur série
  Serial.println(valeurAngleServoLocal);
  }
    delay(20);                                               // avec une petite pause, avant de reboucler
  }
  delay(1000);
}
```

Figure XLVIII.48 : void loop du récepteur des données analogique

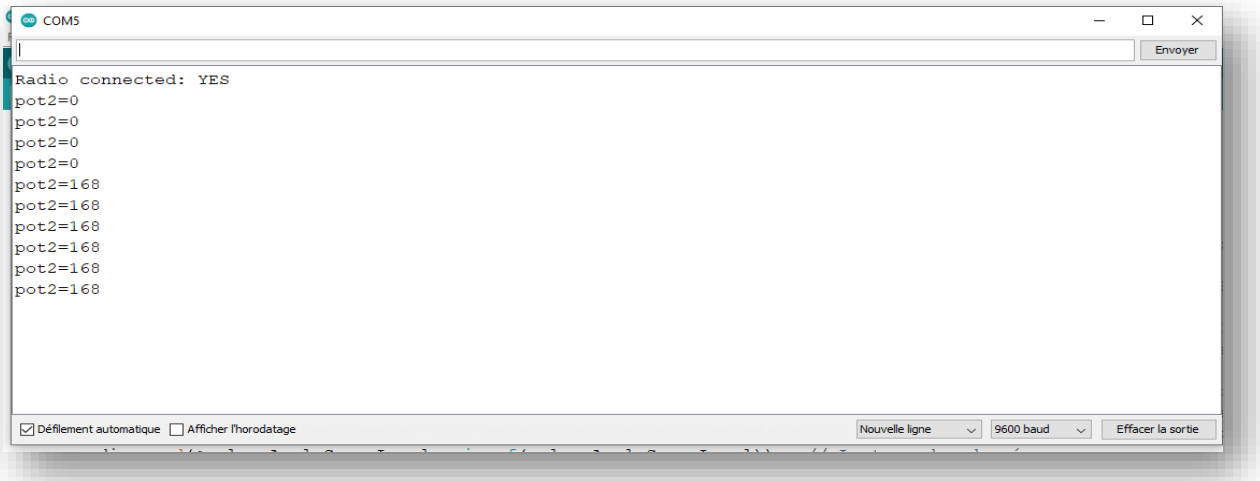

Figure XLIX.49 : moniteur série des données analogiques

III.2.3 Mode sommeille

Le programme si dessous s'agit de montrer comment mettre notre arduino en sommeil profond et comment le réveiller.

```
#include <avr/sleep.h>  //cette bibliothèque AVR contient les méthodes qui contrôlent les modes de veille
#define interruptPin 2  //Pin que nous allons utiliser pour réveiller l'Arduino
```

Figure L.50 : instanciation de bibliothèques on mode sommeil et pin d'interruption

```
void setup() {
Serial.begin(115200);              //Démarrer la communication série
pinMode(LED_BUILTIN,OUTPUT);       // Nous utilisons la led sur la broche 13 pour indiquer quand Arduino est en veille
pinMode(interruptPin,INPUT_PULLUP); // Définissez la broche d2 sur l'entrée à l'aide de la résistance de pullup intégrée
digitalWrite(LED_BUILTIN,HIGH);    //allumer la LED
}
```

Figure LI.51 : void setup du mode sommeil

```
void loop() {
delay(5000);                           // attend 5 secondes avant d'aller dormir
Going_To_Sleep();
}
void Going_To_Sleep(){
sleep_enable();                        //Activation du mode veille
attachInterrupt(0, wakeUp, LOW);       // attacher une interruption à la broche d2
set_sleep_mode(SLEEP_MODE_PWR_DOWN);   //Définition du mode veille, dans notre cas la veille complète
digitalWrite(LED_BUILTIN,LOW);         // éteindre la LED
delay(1000);                           // attendez une seconde pour permettre à la led de s'éteindre avant d'aller dormir
sleep_cpu();                           //activation du mode veille
Serial.println("just woke up!");       //prochaine ligne de code exécutée après l'interruption
digitalWrite(LED_BUILTIN,HIGH);        //allumer la LED
}
void wakeUp(){
Serial.println("Interrrupt Fired");    //Imprime le message sur le moniteur série
sleep_disable();                       //Désactiver le mode veille
detachInterrupt(0);                    //Supprime l'interruption de la broche 2 ;
}
```

Figure LII.52 : void loop du mode sommeil

Pour réaliser ce montage il suffit de faire le Montage suivant :

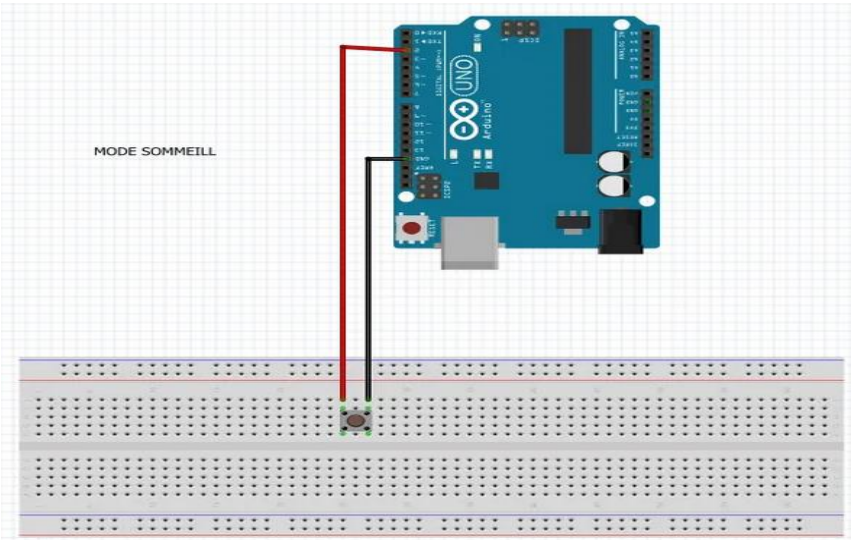

Figure LIII.53 câblage pour le mode sommeil

III.3 Programmation de NRF24L0 bidirectionnel

Nous devons créer une communication série entre deux carte arduino, chaque carte doit fonctionne comme un émetteur et récepteur à la fois, donc chaque arduino va lit la valeur de chaque potentiomètre et l'envoyée à l'autre, et chacun des deux va recevoir au même temps.

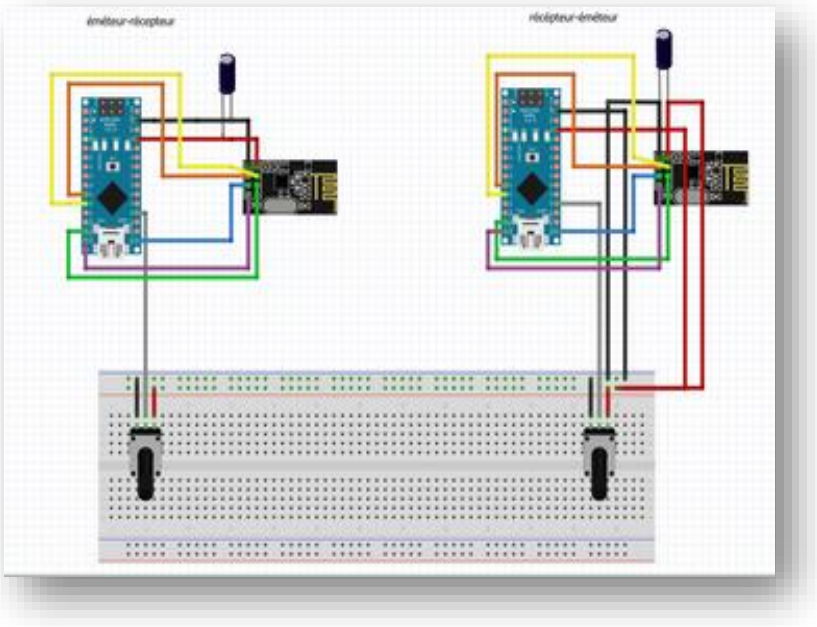

Figure LIV.54 câblage pour l'envoie des données analogique unidirectionnelle

- Emetteur-Récepteur

```
#define pinPOT2    A0       // On associe le point milieu du potentiomètre à l'entrée analogique A0 de l'arduino
int valeurPotLocal;         // Variable contenant la valeur du potentiomètre1
int valeurPotLocal2;        // Variable contenant la valeur du potentiomètre2
int valeurAngleServoLocal;  //valer lit par le radio
int valeurAnaloge;          //valeur de potensiometre1
int data[2];                //défini à 2 données
```

Figure LV.55 : déclaration des variables analogique Emetteur-Récepteur

- Dans la void loop on se trouve :

```
void loop() {
  // ******** ENVOI ********
radio.stopListening();                                  //définir le sens de communication
valeurPotLocal = analogRead(pinPOT2);                   //lire la valeur de pinPOT2
valeurAnaloge = map(valeurPotLocal, 0, 1023, 0, 255);   // On converti la valeur [0..1023] en [0..255]
radio.write(&valeurAnaloge, sizeof(valeurAnaloge));     // ... et on envoi cette valeur à l'autre arduino, via le NRF24
delay(5);                                               // avec une petite pause, avant de passer à la suite
  // ******** RÉCEPTION ********
radio.startListening();                                  // On commence par arrêter le mode envoi, pour pouvoir réceptionner des données
if(radio.available()) {                                  // On regarde si une donnée a été reçue
while (radio.available()) {                              // Si une donné est en attente de lecture, on va la lire
radio.read(&valeurAngleServoLocal, sizeof(valeurAngleServoLocal)); // Lecture des données reçues, une par une
Serial.print("pot2=");                                   //met la valeur de pot2 dans le moniteur série
Serial.println(valeurAngleServoLocal);                   //afficher ma valeur dans le moniteur
}
delay(20);                                               // avec une petite pause, avant de reboucler
}
delay(1000);
}
```

Figure LVI.56 : void loop Emetteur-Récepteur des données analogiques

- Récepteur-Emetteur

```
#define pinPOT    A1        // On associe le point milieu du potentiomètre à l'entrée analogique A0 de l'arduino
#define pinoutpot  9        // On associe la broche "SIGNAL" du SERVO à la sortie digitale D9 de l'arduino
int valeurPotLocal;         // Variable contenant la valeur du potentiomètre1
int valeurPotLocal2;        // Variable contenant la valeur du potentiomètre2
int valeurAnaloge2;         //valeur de potensiometre 2
int valeurAngleServoLocal;  //valeur lit par le radio
 pinMode(pinPOT, INPUT);    // Déclaration de la pin "pinPOT" en entrée
```

Figure LVII.57 : déclaration des variables analogique Récepteur-Emetteur

- Dans la void loop on se trouve :

```
void loop() {
  // ******** ENVOI ********
radio.stopListening();                                  // On commence par arrêter le mode écoute, pour pouvoir émettre les données
valeurPotLocal2 = analogRead(pinPOT);                   //lit la valeur de potentiometre 1
valeurAnaloge2 = map(valeurPotLocal2, 0, 1023, 0, 255); // On converti la valeur [0..1023] en [0..255]
radio.write(&valeurAnaloge2, sizeof(valeurAnaloge2));   // ... et on envoi cette valeur à l'autre arduino, via le NRF24
delay(5);                                               // avec une petite pause, avant de passer à la suite
 // ******** RÉCEPTION ********
radio.startListening();                                 // On commence par arrêter le mode envoi, pour pouvoir réceptionner des données
if(radio.available()) {                                 // On regarde si une donnée a été reçue
while (radio.available()) {                             // Si une donné est en attente de lecture, on va la lire
radio.read(&valeurAngleServoLocal, sizeof(valeurAngleServoLocal));  // Lecture des données reçues, une par une
Serial.print("pot1=");                                  // affichage dans le moniteur série
Serial.println(valeurAngleServoLocal);
}
delay(20);                                              // avec une petite pause, avant de reboucler
}
delay(1000);
```

Figure LVIII.58 : void loop Récepteur-Emetteur des données analogiques

- Emetteur-Récepteur

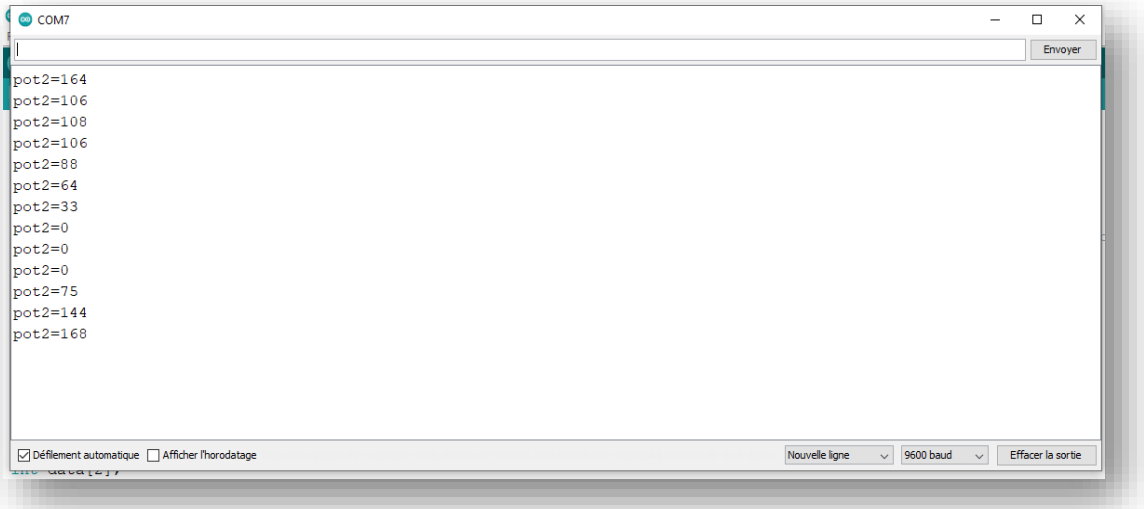

Figure LIX.59 : moniteur série d'Emetteur-Récepteur des données analogiques

- Récepteur-Emetteur

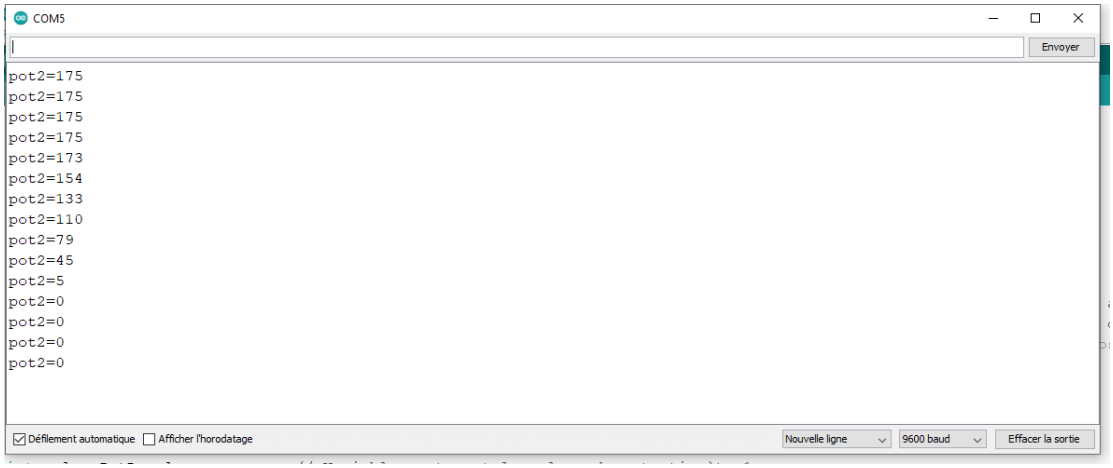

Figure LX.60 : moniteur série du Récepteur-Emetteur des données analogiques

III.4 Communication sans fils par LoRa02-1278 (20 kilomètres)

Le deuxième pas était avec l'arduino et LoRa, on va faire la même étape pour l'étude de programme, on va commencer par :

- instancier la bibliothèque LoRa dans l'arduino

```
#include <SPI.h>    //appel de bibliothèque pour le microcontroleur
#include <LoRa.h>   //appel de bibliothèque pour le module LoRa
```

Figure LXI.61 : instanciation des bibliothèques de LoRa

- dans la void setup, on va configurer directement les caractéristiques de LoRa, et touts les fonctions commun on les expliquées avec le nrf24L01

```
void setup() {
Serial.begin(115200);                    //initialiser la communication à partir de 115200 mbs
while (!Serial);                         //Attend que le port série s'ouvre
Serial.println("LoRa Sender");           //afficher LoRa Sender dans le moniteur série
if (!LoRa.begin(433E6)) {                //configurer le module LoRa pour qu'il fonctionne à 433 MHz
Serial.println("Starting LoRa failed!"); //Starting LoRa failed!
}
}
```

Figure LXII.62 : Fonctions de void setup LoRa

- dans la void loop on peut ajouter quelconque fonction ou instruction qui va nous aider pour exécuter le programme que nous besoin.

III.5 Programmation de LoRa unidirectionnel

III.5.1 Transmission d'une action TOR

Comme la fois passé avec le nRF24L01 on va envoyer au récepteur une donné de type caractère, cette donné est l'expression « ''hello'' », Le schéma suivant va présente le montage pour réaliser cette essaye.

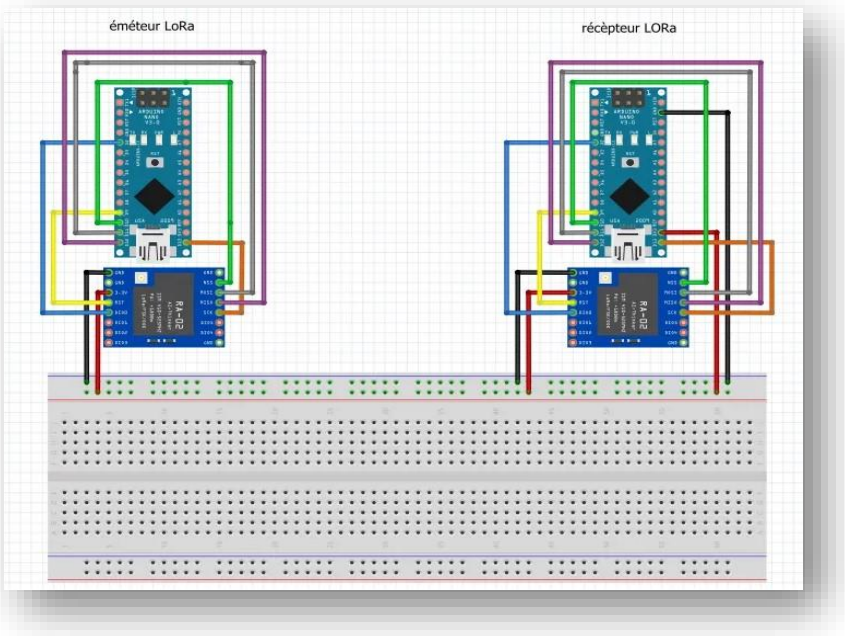

Figure LXIII.63 : schéma de câblage LoRa pour l'envoie d'une action TOR

- On défini un compteur pour l'incrémentation, c'est à dire à chaque fois le compteur incrémenter et terminer la boucle

```
int counter = 0;//définit le compteur de type integer
```

Figure LXIV.64 : définir le type de compteur

- Emetteur

```
void loop() {
Serial.print("Sending packet: ");// met cette experssion dans le moniteur
Serial.println(counter);          //démmarage du compteur
// envoie de paquet
LoRa.beginPacket();               //début d'onvoie de paquet
LoRa.print("hello");              //envoyer hello dans le moniteur
LoRa.print(counter);              //afficher le compteur dans le moniteur
LoRa.endPacket();                 //fin d'envoie de paquet
counter++;                        //incrémentation de compteur
delay(1000);                      //pause avant reboucler
}
```

Figure LXV.65 : void loop d'émetteur LoRa pour action TOR

- Récepteur

```
void loop() {
int packetSize = LoRa.parsePacket(); //Vérifiez si un paquet a été reçu.
if (packetSize) {                    //reçu un paquet
Serial.print("Received packet '");   // met dans le moniteur série
while (LoRa.available()) {           //vérifier si un paquet a été reçu
Serial.print((char)LoRa.read());     //met le paquet reçu dans le moniteur
}
}
}
```

Figure LXVI.66 : void loop de récepteur LoRa pour action TOR

L'affichage dans le moniteur pour l'émetteur étaits comme suite :

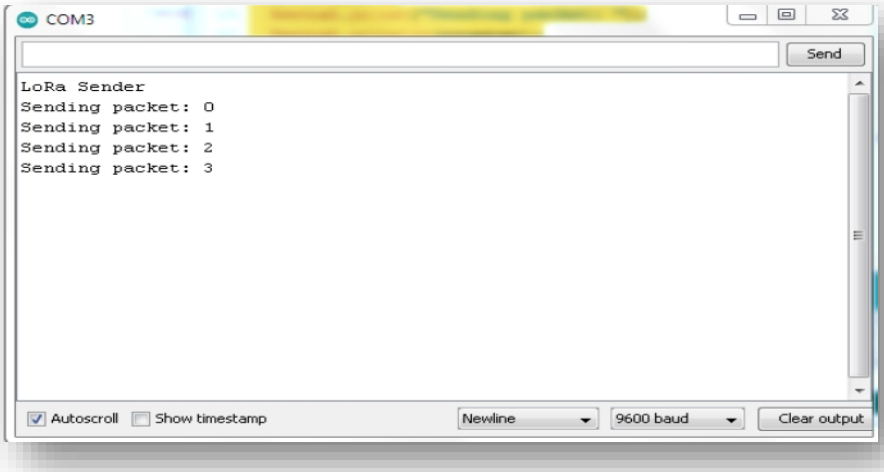

Figure LXVII.67: moniteur série d'émetteur LoRa d'action TOR

L'affichage dans le moniteur pour le récepteur étaits comme suite

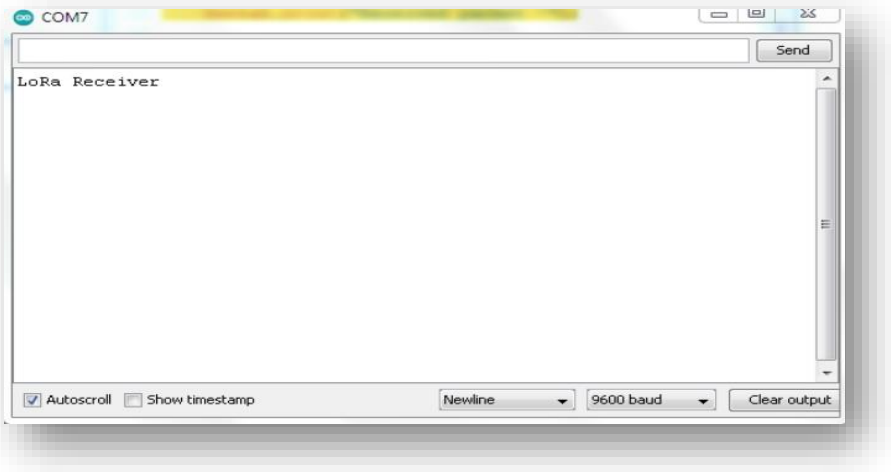

Figure LXVIII.68 : moniteur série de récepteur LoRa d'action TOR

III.5.2 L'envoie de paquet de données analogique

Le même travaille que l'on faire avec le nrf24l0, ces d'essayé d'envoyé des donnés analogique à l'aide de résistance variable.

- On suivi cette schéma de câblage

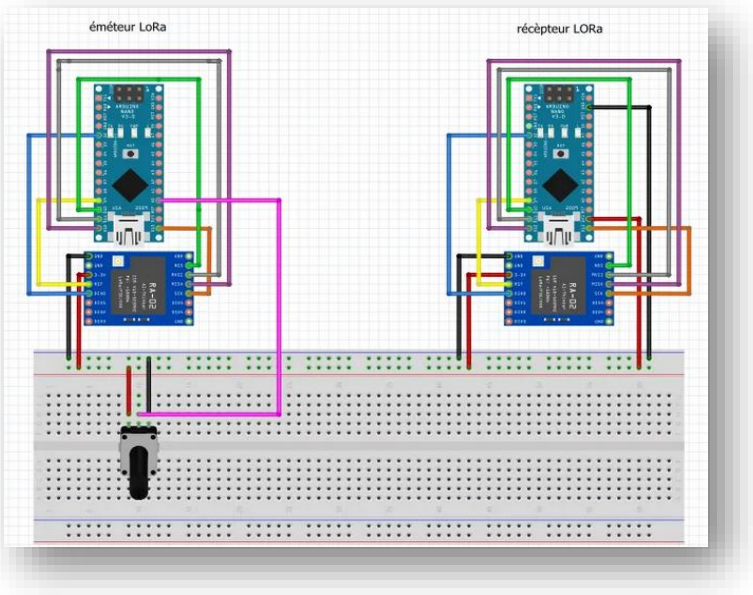

Figure LXIX.69 : schéma de câblage de LoRa avec une résistance variable

- Donc, on va déclarer les variables :

```
int pot = A0;        //on associe le point milieu du potenstiometre à l'entré analogique A0
pinMode(pot,INPUT); //déclaration du potenstion métre comme un entré
```

Figure LXX.70 : déclaration des variables analogique avec LoRa

```
void loop() {
int val = map(analogRead(pot),0,1024,0,255);//on conveti la valeur [0..1023]en [0..255]
LoRa.beginPacket();  //démmarage d'envoie des données
LoRa.print(val);     // met la valeur de potentionmetre dans le moniteur
LoRa.endPacket();    //fin d'envoie des données
delay(50);           //pause avant reboucler
}
```

Figure LXXI.71 : Void loop d'émetteur des données analogique avec LoRa

Le résultat dans le moniteur est comme suite :

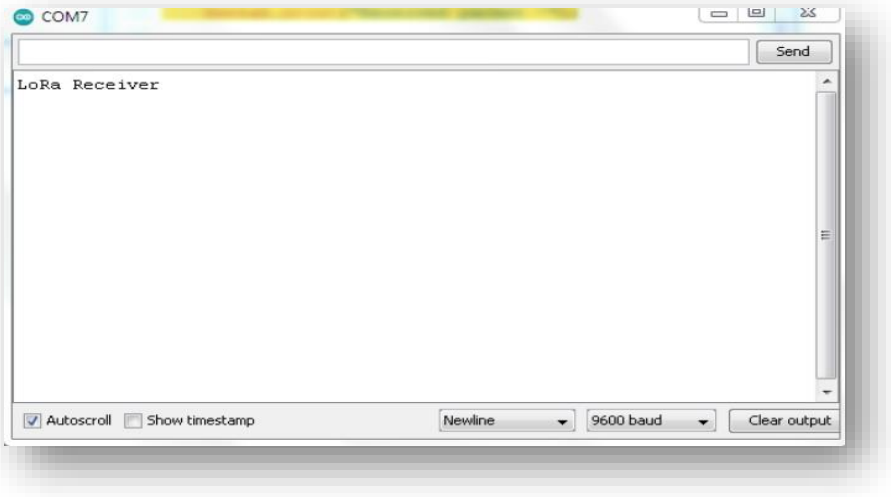

Figure LXXII.72 : moniteur série de récepteur LoRa pour l'envoie des données analogique

Donc, on remarque que l'émetteur est dans l'état de bon fonctionnement, par contre le récepteur qui n'affiche rien, ou afficher des informations perturbé ou un message incomplet.

On essayé d'ajouter une alimentation externe, installé des bibliothèques concerné au LoRa 1278, changé la vitesse de transmission, mais pas de changement à l'état de récepteur.

On trouve que nos premières tentatives ont été infructueuses, mais nous essaierons encore et encore jusqu'à ce que nous trouvions le problème et leur solution.

III.6 Le cahier de charge

Dans cette partie on va faire un programme qui résout le problème de la société d'ANABIB IRRAGRIS que nous mentionnons à l'introduction.

Donc, leur système d'irrigation approuvé est un système d'arrosage comme nous avons parlé dans le premier chapitre, et pour détecter l'angle d'accouplement dans la rampe, on branche un capteur de fin de course à cette point la, et quand la rampe faite cette déplacement d'angle, l'émetteur branché avec le capteur de fin course va envoyer un signal au récepteur qui est branché aussi avec un relie de l'autre coté, Le récepteur va joue sont rôle et interrompt le processus par un relie pour éviter la casse de rampe d'arrosage et préserver ce système.

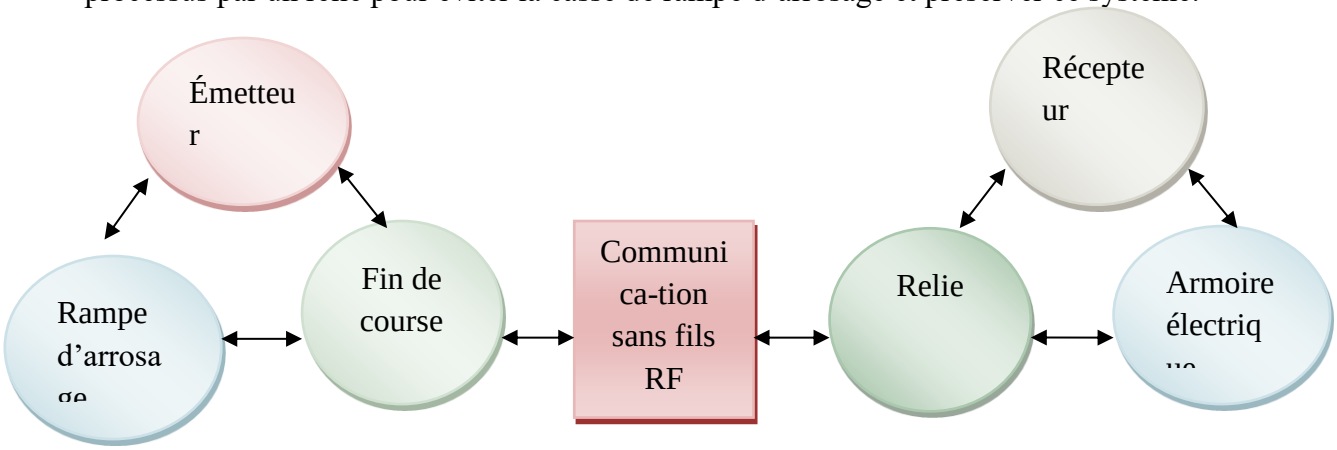

Figure LXXIII.73 : schéma de solution de problème

III.7 Programme de ce problème

Premièrement, on définit un schéma de câblage par fritzing pour ce cahier de charge, on remplace le fin de course par un Botton et le relie par une LED elle est définit comme suite :

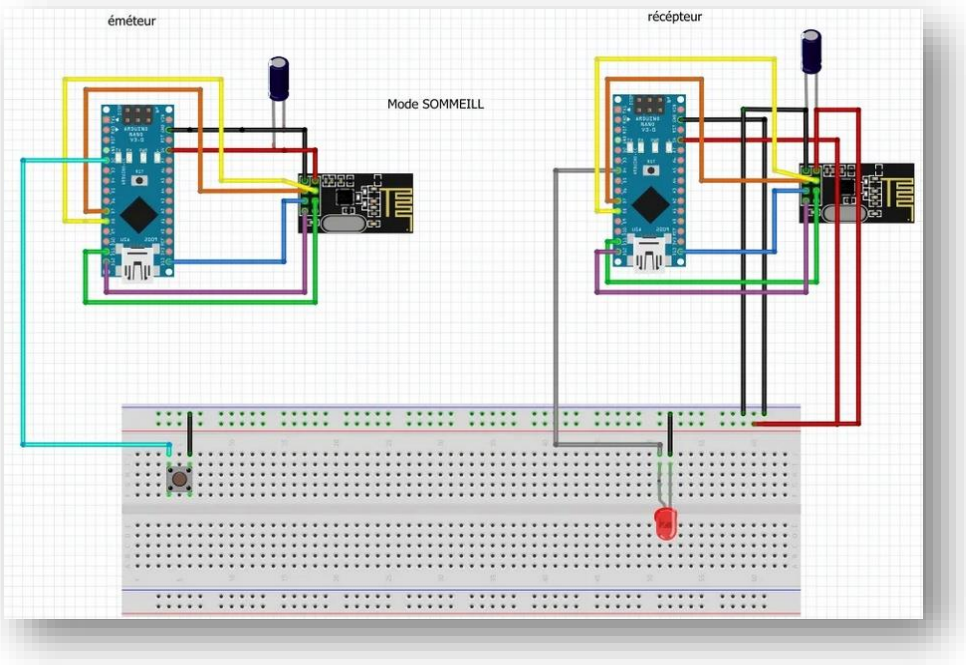

Figure LXXIV.74 : schéma de câblage de système réalisé

Deuxièmement, le mode sommeil sera fait dans l'émetteur pour économiser au maximum l'énergie de batterie, c'est quand on fait l'interruption, l'émetteur passe de l'état de sommeil à l'état de veille et envoyé le signal puis revient à l'état de sommeil.

- On définie les variable

```
const char message[] = "PROBLEM !!!";// Message à transmettre à l'autre NRF24 (32 caractères maxi, avec cette librairie)
pinMode(LED_BUILTIN,OUTPUT);        //Nous utilisons la led pour indiquer quand Arduino est en veille
pinMode(interruptPin,INPUT_PULLUP); //broche d2 pour entrer en utilisant la résistance pullup intégrée
```

Figure LXXV.75 : définition des variables de cahier de charge d'émetteur

- On fait l'appel de plusieurs loop ici :

```
void loop() {
delay(5000);                              //attendre 5 secondes avant d'aller dormir
 Going_To_Sleep();
}
```

Figure LXXVI.76 : void loop de cahier de charge d'émetteur

```
void Going_To_Sleep(){
sleep_enable();                         //Activation du mode veille
attachInterrupt(0, wakeUp, LOW);        //attacher une interruption à la broche d2
set_sleep_mode(SLEEP_MODE_PWR_DOWN); //Réglage du mode veille, dans notre cas le sommeil complet
digitalWrite(LED_BUILTIN,LOW);          //éteindre la LED
delay(1000);                            //attendre une seconde pour permettre à la led de s'éteindre avant d'aller dormir
sleep_cpu();                            //activer le mode veille
Serial.println("just woke up!");        //prochaine ligne de code exécutée après l'interruption
digitalWrite(LED_BUILTIN,HIGH);         //allumer la LED
}
```

Figure LXXVII.77 : Void d'état de sommeille

```
void wakeUp(){
Serial.println("Interrrupt Fired");      //Imprimer le message sur le moniteur série
sleep_disable();                         //Désactiver le mode veille
detachInterrupt(0);                      //Supprime l'interruption de la broche 2
radio.write(&message, sizeof(message)); // Envoi de notre message
delay(1000);                             // ... toutes les secondes !
}
```

Figure LXXVIII.78 : Void d'état de veille

Troisièmement, on défini le cahier de charge dans le récepteur en langage C par ce programme

- On définie les variable

```
const int LED4=4;//on associe la broche de la LED à la sortie digital 4
pinMode(LED4, OUTPUT);//on définit le type de LED comme sortie
```

Figure LXXIX.79 : définition des variables

- Void loop de récepteur : nous demandons d'afficher l'expression "PROBLEM : " si le signal est reçue et allumez la LED, c'est à dire coupé le relie

```
void loop() {
  // On vérifie à chaque boucle si un message est arrivé
if (radio.available()) {
radio.read(&message, sizeof(message));// Si un message vient d'arriver, on le charge dans la variable "message"
Serial.print("PROBLEM : ");
Serial.println(message);              // ... et on l'affiche sur le port série !
digitalWrite(LED4, HIGH);             // turn the LED on (HIGH is the voltage level)
delay(5000);
digitalWrite(LED4, LOW);
}
else {digitalWrite(LED4, LOW);        // turn the LED off by making the voltage LOW
}
}
```

Figure LXXX.80 : Void loop de récepteur

Le moniteur série nous afficher l'expression "PROBLEM : " et nous allumons la LED quand le message est reçue.

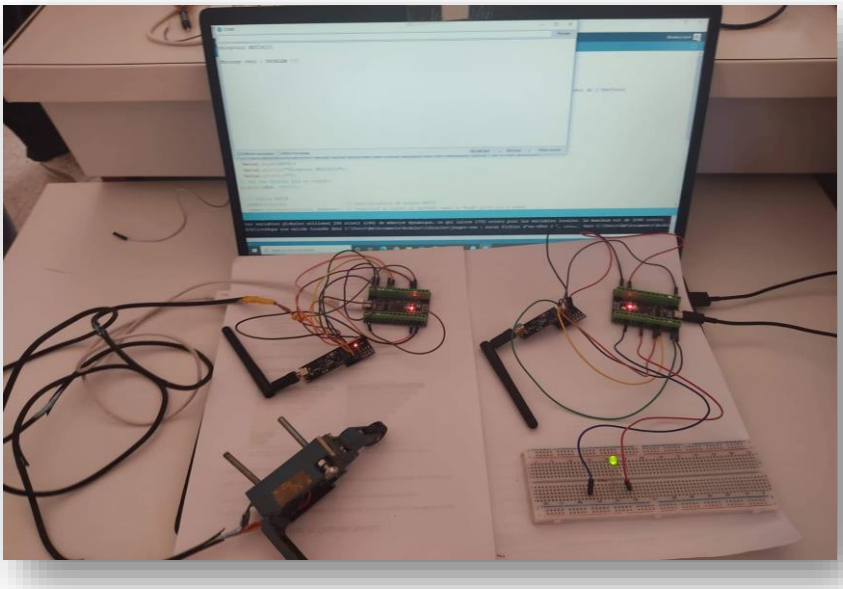

Figure LXXXI.81 : résultat de teste d'interruption

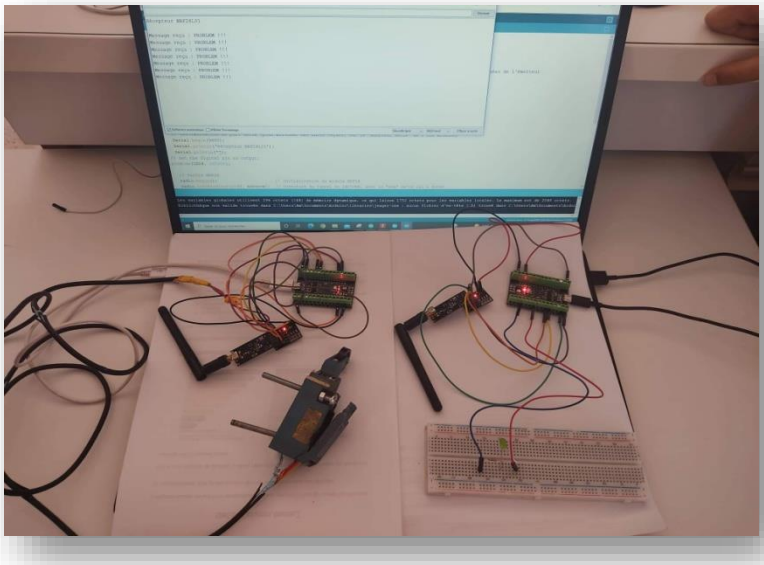

Figure LXXXII.82 : résultat de fin d'interruption

III.8 Conclusion

Dans ce chapitre, on a réalisons une communication sans fils entre les arduino à base de radio fréquence, on essaye d'envoyer des différents type de données par utilisé deux types de radio fréquence : nRF24L01et LoRa 1278, à partir les essayes que nous faisons, nous choisissons le nRF24l01 pour son bon fonctionnement et son distance de communication (1Km), elle est la distance que nous besoins pour résoudre notre problème dans le système d'arrosage, dans nos recherches nous avons atteint une distance d'environ 300 mètres .

Par la suite, nous poursuivrons nos recherche sur le radio fréquence LoRa 1278, parce qu'il utilise une technologie avancé pour faire connecter à spectre étalé longue porté, on motionne aussi leur distance e communication qui est estimé à environ 20Km.

Conclusion générale :

Notre travaille présente l'étude et l'analyse et la Conception d'une solution de commande a distances pour les pivots d'irrigation que nous considérons comme une solution au problème de la société ANABIB IRRAGRIS qui entrave le fonctionnement de la rampe d'arrosage existé dans le système d'irrigation en cas de dépassement de l'angle formé entre deux travées, ce projet est réalisé à base de la carte arduino et le nrf24l01.

Dans un premier temps, nous avons également appris la recherche par les sites web, les livres, mémoires …etc, sur les systèmes agricoles traditionnelles et modernes, dont le développement reste continu au maximum afin de couvrir les zones stériles, intensifier la production et de stimuler le couvert végétal dans le monde entier.

Nous avons passé en revue quelques-unes des expériences menées par les agriculteurs pour trouver la méthode la plus efficace, nous nous sommes approfondis dans la recherche des caractéristiques du système d'irrigation par aspersion et ses composant, donc on appuyées sur le modèle EL1160 (30ha) existant dans la société IRRAGRISS.

On a aussi eu un bref aperçu de l'entreprise nationale de tube et transformation de produit plats, qui contient 5 unités, parmi eux : ANABIB IRRAGRIS existante au notre wilaya de Bordj Bou Arreridj.

Dans un deuxième temps, nous avons appris à connaitre les différents composants que nous avons utilisées dans cette recherche, tels que le type de la carte arduino chacun de sa caractéristique, les bienfait, et connaitre le rôle important qu'elle joue dans le domaine électronique et le développement technologique, aussi le simple langage C dont il dépend pour faire ses commandes.

On mentionne également les module de radio fréquence : nrf24l01 et Lora Ra-02 sx1278 qui joue le rôle d'émetteur ou récepteur et peut être les deux au même temps, appris comment obtenir ce modèle de communication par brancher l'arduino avec un module de radio fréquence.

Finalement, nous avons faire une programmation sur logiciel arduino après avoir fait identifier les instructions que nous utilise pour rédiger un programme simple avec un mode sommeil qui définit notre cahier de charge et résoudre notre problème à base de carte arduino de type Nano et radio fréquence de type nrf24l01.

Et comme point de vue, nous proposons de continuer et réessayer avec le module LoRa malgré certains des obstacles que nous avons rencontrés avec cette dernière et développer notre réalisation par des autres radios qui communique à longue distance pour faciliter l'irrigation et gagner du temps.

Références bibliographiques :

[1] : le différent système d'irrigation (Par Prof. Mohammed AZOUGGAGH)

[2] : GESTION DES EAUX EN IRRIGATION Manuel de formation n° 5 Méthodes d'irrigation Manuel (par C. Brouwer) Institut international pour l'amélioration et la mise en valeur des terres

[3] : Tamara Communal. Directorio de Aguas de Cangahua, 2014

[4] : agroscope, 1964conthey (André ANçAY.catherine A. baroffio et vincent michel,)

[5] : Lahlouh Mohamed. Ridha Azizou Abderrahmane, (Etude et réalisation d'un Système d'irrigation automatique), Université Djilali Bounaâma de Khemis Miliana, 2017 / 2018.page20

[6] : https://france-pivots.com/pivots-irrigation/systeme-pivot-irrigation/

[7] : SOUALAH MOHAMMED Yazid. KOUKA Abed El Basset, (Etude hydraulique du système de pivot d'irrigation Type ANABIB), Université Echahid Hamma Lakhdar d'El-Oued, 2017-2018, page 19

[8] : livret arduino en français par (Jean-Noël montagné centre de ressource Art sensitif)

[9] : programmation arduino en ligne de commande » [archive], sur blog de François Tessier

[10] : www.encyclopedie-environnement.org

[11] :passionelectronique.fr/tutorial-nrf24l01/

[12] :www.tactis.fr/iot-lora/

[13] : https://fritzing.fr.malavida.com

[14] : https://ar.wikipedia.org/wiki/

[15] : www.geeksvalley.com/product/arduino-nano/

Références bibliographiques du Figures :

[Figure I.1] : www.ksb.com

[Figure I.2] : www.agronomie.info/fr/wp-content/uploads/2016/02/DSC_0841-1.jpg

[Figure I.3] : Utilisation des systèmes d'irrigation, Chapeaux & Enomoto/PCDA, 2009 - Practica Foundation 2009

[Figure I.4] : Utilisation des systèmes d'irrigation, Chapeaux & Enomoto/PCDA, 2009 – Practica Foundation 2006 – Netafim 2009

[Figure I.5]: www.agrocomposites.fr

[Figure I.6]: www.agrocomposites.fr

[Figure I.7]: fr.wikipedia.org/

[Figure I.8]: www. irrigazette.com

[Figure I.9]: www.agriexpo.online

[Figure I.10] : www.dspace.univeloued.dz/

[FigureI.11]: www.docplayer.fr/90024974-Calcul-et-optimisation-d-un-mini-pivot-dirrigation.html

[Figure I.12] : www.serie04.com

[Figure I.13]: Mghezzi Chaa Khaled, (Calcul et Optimisation D'un Mini Pivot D'irrigation), UNIVERSITE MOHAMED KHIDER BISKRA, 2009

[Figure I.14]: www.2ie.com

[Figure I.15]: www.electronics212.com/2018/02/arduino-types.html

[Figure I.16] : www.electronics212.com/2018/02/arduino-types.html

[Figure I.17] : www.electronics212.com/2018/02/arduino-types.html

[Figure I.18] : www.amazon.fr

[FIGURE I.19] : Www.letmeknow.fr/fr/communications/

[Figure I.20] : www.letmeknow.fr/fr/communications/

Annexes:

Emetteur « Hello HANANE »

```
 #include <SPI.h>
#include <RF24.h>
#define pinCE   7
#define pinCSN  8
#define tunnel  "PIPE1"
RF24 radio (pinCE, pinCSN);
const byte adresse [6] = tunnel;
const char message[] = "HELLO HANANE!!!»
void setup() {
 radio.begin();
 radio.openWritingPipe (adresse);
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening ();
}
Void loop () {
 radio.write (&message, sizeof (message));
 Delay(1000);
}
```

Récepteur « Hello HANANE »

```
#include <SPI.h>
#include <RF24.h>
#define pinCE   7
#define pinCSN  8
#define tunnel  "PIPE1"
RF24 radio(pinCE, pinCSN);
```

```
const byte adresse[6] = tunnel;
char message[32];
void setup() {
  Serial.begin(9600);
  Serial.println("Récepteur NRF24L01");
  Serial.println("");
  radio.begin();
  radio.openReadingPipe(0, adresse);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
}
Void loop () {
If (radio.available()) {
radio.read (&message, sizeof (message));
Serial.print("Message reçu : ");
Serial.println(message);
  }}
```

Émetteur DATA

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
bool radioNumber = 1;
#define CE_PIN   7
```

```
#define CSN_PIN  8

const uint64_t adresse = 0xE8E8F0F0E1LL;

int data[5];

const int bp = 2;

int buttonState = 0;

RF24 radio(CE_PIN, CSN_PIN);

byte addresses[][6] = {"1Node", "2Node"};

bool role  ;

void configureRadio() {

 radio.begin();

Serial.print("Radio connected: ");

Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");

 radio.setPALevel(RF24_PA_LOW);

  if (radioNumber) {

    radio.openWritingPipe(addresses[1]);

    radio.openReadingPipe(1, addresses[0]);

  } else {

    radio.openWritingPipe(addresses[0]);

    radio.openReadingPipe(1, addresses[1]);

  }

  radio.startListening();

  radio.printDetails();
```

```
}
void setup()
{
  radio.begin();
  radio.openWritingPipe(adresse);
   pinMode(bp, INPUT);
}
void loop()
{
  buttonState = digitalRead(bp);
  if (buttonState==HIGH){
  data[0] = 11;
  data[1] = 12;
  data[2] = 13;
  data[3] = 14;
  data[4] = 15;
  }
  else {
  data[0] = 00;
  data[1] = 12;
  data[2] = 13;
  data[3] = 14;
  data[4] = 15;
```

```
 }
  radio.write( data, sizeof(data) );
  delay(2000);
}
```

Récepteur DATA

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
bool radioNumber = 0;
#define CE_PIN  7
#define CSN_PIN 8
const uint64_t adresse = 0xE8E8F0F0E1LL;
const int ledPin =  4 ;
int data[5];
RF24 radio(CE_PIN, CSN_PIN);
byte addresses[][6] = {"1Node", "2Node"};
bool role = 0;
void configureRadio() {
  radio.begin();
  Serial.print("Radio connected: ");
  Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");
  radio.setPALevel(RF24_PA_LOW);
  if (radioNumber) {
```

```
    radio.openWritingPipe(addresses[1]);
    radio.openReadingPipe(1, addresses[0]);
  } else {
    radio.openWritingPipe(addresses[0]);
    radio.openReadingPipe(1, addresses[1]);
  }
  radio.startListening();
  radio.printDetails();
}
int data[5];
void setup()
{
  Serial.begin(9600);
  radio.begin();
    Serial.print("Radio connected: ");
    Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");
    radio.openReadingPipe(1,adresse);
  radio.startListening();
  pinMode(ledPin, OUTPUT);
  }
void loop() {
  if ( radio.available() )
  {
```

```
    radio.read( data, sizeof(data) );
    Serial.print("data[0]=");
    Serial.println(data[0]);
    Serial.print("data[1]=");
    Serial.println(data[1]);
    Serial.print("data[2]=");
    Serial.println(data[2]);
    Serial.print("data[3]=");
    Serial.println(data[3]);
    Serial.print("data[4]=");
    Serial.println(data[4]);
        }
  else Serial.print("no data");
  if (data[0] == 11)
  {  digitalWrite(ledPin, HIGH);
   }
  else digitalWrite(ledPin, LOW);
  delay(2000);
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Emetteur données analogique unidirectioonnelle

```
#include <SPI.h>
```

```
#include <RF24.h>

#include <nRF24L01.h>

bool radioNumber = 0;

#define pinCE     7

#define pinCSN    8

#define pinPOT2    A0

#define tunnel1 "PIPE1"

RF24 radio(pinCE, pinCSN);

const byte adresses[][6] = {tunnel1};    "

int valeurPotLocal;

int valeurAngleServoLocal;

int valeurAnaloge;

void configureRadio() {

bool role = 0;

  radio.begin();

  Serial.print("Radio connected: ");

  Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");

. RF24_PA_MAX is default.

  radio.setPALevel(RF24_PA_LOW);

}

void setup() {

  pinMode(pinPOT2, INPUT);

  radio.begin();
```

```
 radio.openWritingPipe(adresses[0]);
 radio.setPALevel(RF24_PA_MIN);
 delay(2000);
Serial.begin(9600);
radio.begin();
Serial.print("Radio connected: ");
Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");
}
void loop() {
 valeurPotLocal = analogRead(pinPOT2);
 valeurAnaloge = map(valeurPotLocal, 0, 1023, 0, 255);
 radio.write(&valeurAnaloge, sizeof(valeurAnaloge));
 Serial.print("pot=");
 Serial.println(valeurAnaloge);
 delay(20);
 delay(1000);
}
```

Récepteur données analogique unidirectionnelle

```
#include <SPI.h>
#include <RF24.h>
#include <nRF24L01.h>
bool radioNumber = 0;
#define pinCE     7
```

```
#define pinCSN    8
#define pinoutpot  9
#define tunnel2 "PIPE1"
RF24 radio(pinCE, pinCSN);
const byte adresses[][6] = {tunnel2};    "
int valeurAngleServoLocal;
bool role = 0;
void configureRadio() {
  radio.begin();
  Serial.print("Radio connected: ");
  Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");
}
void setup() {
  pinMode(pinoutpot, OUTPUT);
  radio.begin();
  radio.openReadingPipe(1, adresses[1]);
  radio.setPALevel(RF24_PA_MIN);
  delay(2000);
  Serial.begin(9600);
  radio.begin();
    Serial.print("Radio connected: ");
    Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");
}
```

```
void loop() {
  if(radio.available()) {
    radio.read(&valeurAngleServoLocal, sizeof(valeurAngleServoLocal));
    Serial.print("potrec=");
    Serial.println(valeurAngleServoLocal);
       }
    delay(20);
  delay(1000);
}
```

Emetteur –récepteur analogique

```
#include <SPI.h>
#include <RF24.h>
#include <nRF24L01.h>
#include <RF24Network.h>
bool radioNumber = 0;
#define pinCE     7
#define pinCSN    8
#define pinPOT2    A0
#define tunnel1 "PIPE1"
#define tunnel2 "PIPE2"
RF24 radio(pinCE, pinCSN);
RF24Network network(radio);
const byte adresses[][6] = {tunnel1, tunnel2};    "byte array"
```

```
int valeurPotLocal;
int valeurPotLocal2;              int valeurAngleServoLocal;
int valeurAnaloge;
int data[2];
void configureRadio() {
bool role = 0;
  radio.begin();
  Serial.print("Radio connected: ");
  Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");
  radio.setPALevel(RF24_PA_LOW);
}
void setup() {
  pinMode(pinPOT2, INPUT);
  radio.begin();
  radio.openWritingPipe(adresses[0]);
  radio.openReadingPipe(1, adresses[1]);
  radio.setPALevel(RF24_PA_MIN);
  delay(2000);
  Serial.begin(9600);
  radio.begin();
    Serial.print("Radio connected: ");
    Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");
}
```

```
void loop() {
  // ******** ENVOI ********
  radio.stopListening();
  valeurPotLocal = analogRead(pinPOT2);
  valeurAnaloge = map(valeurPotLocal, 0, 1023, 0, 255);


  radio.write(&valeurAnaloge, sizeof(valeurAnaloge));
  delay(5);
// ******** RÉCEPTION ********
  radio.startListening();
  if(radio.available()) {
    while (radio.available()) {
      radio.read(&valeurAngleServoLocal, sizeof(valeurAngleServoLocal));
    Serial.print("pot2=");
    Serial.println(valeurAngleServoLocal);
    }
    delay(20);
  }
  delay(1000);
}
```

Récepteur /émetteurs analogique

```
#include <SPI.h>
#include <RF24.h>
```

```
#include <nRF24L01.h>
#include <RF24Network.h>
bool radioNumber = 0;
#define pinCE     7
#define pinCSN    8
#define pinPOT    A1
#define pinoutpot  9
#define tunnel1 "PIPE2"
#define tunnel2 "PIPE1"
RF24 radio(pinCE, pinCSN);  RF24Network network(radio);
const byte adresses[][6] = {tunnel1, tunnel2};
int valeurPotLocal;
int valeurPotLocal2;
int valeurAnaloge2;
int valeurAngleServoLocal;
bool role = 0;
int data[2];
void configureRadio() {
  radio.begin();
  Serial.print("Radio connected: ");
  Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");
  radio.setPALevel(RF24_PA_LOW);
}
```

```
void setup() {
  pinMode(pinPOT, INPUT);
  radio.begin();
  radio.openWritingPipe(adresses[0]);
  radio.openReadingPipe(1, adresses[1]);
  radio.setPALevel(RF24_PA_MIN);
  delay(2000);
  Serial.begin(9600);
  radio.begin();
    Serial.print("Radio connected: ");
    Serial.println(radio.isChipConnected() ? "YES" : "NO !!!");
}
void loop() {
  // ******** ENVOI ********
  radio.stopListening();
  valeurPotLocal2 = analogRead(pinPOT);
    valeurAnaloge2 = map(valeurPotLocal2, 0, 1023, 0, 255);
  radio.write(&valeurAnaloge2, sizeof(valeurAnaloge2));
  delay(5);
 // ******** RÉCEPTION ********
  radio.startListening();
  if(radio.available()) {
    while (radio.available()) {
```

```
      radio.read(&valeurAngleServoLocal, sizeof(valeurAngleServoLocal));
    Serial.print("pot2=");
    Serial.println(valeurAngleServoLocal);
   }
    delay(20
  }
  delay(1000);
}
```

Emetteur de cahier de charge

```
#include <SPI.h>
#include <RF24.h>
#include <avr/sleep.h>
#define interruptPin 2
#define pinCE   7
#define pinCSN  8
#define tunnel  "PIPE1"
RF24 radio(pinCE, pinCSN);
const byte adresse[6] = tunnel;
const char message[] = "PROBLEM !!!";
void setup() {
  Serial.begin(9600);
  radio.begin();
  radio.openWritingPipe(adresse);
```

```
  radio.setPALevel(RF24_PA_MIN);
  radio.stopListening();
  pinMode(LED_BUILTIN,OUTPUT);
  pinMode(interruptPin,INPUT_PULLUP);/
  digitalWrite(LED_BUILTIN,HIGH);
}
void loop() {
  delay(5000);
 Going_To_Sleep();
}
void Going_To_Sleep(){
    sleep_enable();
    attachInterrupt(0, wakeUp, LOW);
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    digitalWrite(LED_BUILTIN,LOW);
    delay(1000);
    sleep_cpu();
    Serial.println("just woke up!");
    digitalWrite(LED_BUILTIN,HIGH);
  }
void wakeUp(){
  Serial.println("Interrrupt Fired");
   sleep_disable();
```

```
 detachInterrupt(0);
 radio.write(&message, sizeof(message));
 delay(1000);
}
```

récepteur de cahier de charge

```
#include <SPI.h>
#include <RF24.h>
#define pinCE   7
#define pinCSN  8
#define tunnel  "PIPE1"
RF24 radio(pinCE, pinCSN);
const byte adresse[6] = tunnel;
char message[32];
const int LED4=4;
void setup() {
 Serial.begin(9600);
  Serial.println("Récepteur NRF24L01");
  Serial.println("");
pinMode(LED4, OUTPUT);
  radio.begin();
 radio.openReadingPipe(0, adresse);
  radio.setPALevel(RF24_PA_MIN);
  radio.startListening();
```

```
}
void loop() {
    if (radio.available()) {
    radio.read(&message, sizeof(message));
    Serial.print("Message reçu : ");
    Serial.println(message);
   digitalWrite(LED4, HIGH);
   delay(5000);
     digitalWrite(LED4, LOW);
     }
  else {digitalWrite(LED4, LOW);
}
 }
```