الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Centre Universitaire Abdelhafid Boussouf - Mila

Institut des Sciences et de Technologie

Département des Sciences et Techniques

Nº Réf :……

Projet de Fin d'Etude préparé En vue de l'obtention du diplôme de MASTER

Spécialité: Electromécanique

Conception et réalisation d'un système de capteurs sans fil pour la maintenance prédictive des machines

Réalisé par :
- BOUANIK Samah
- BOUDJATAT Sara

Soutenu devant le jury :

M.	**Rafik BOUHANECHE**	**Président**
M.	**Bessem KAGHOUCHE**	**Examinateur**
Mme	**Farida MEDJANI**	**Promoteur**

Année universitaire : 2019/2020

Résumé

L'utilisation des réseaux des capteurs sans fil (RCSF) dans la plupart des domaines est en plein essor. Ces RCSF sont utilisés pour la surveillance dans les environnements considérés à risques ou non accessibles.

Le but de ce travail de mémoire est la conception d'un réseau sans fil pour la maintenance prédictive des machines. Notre RCSF est constitué des nœuds ESP32.

L'analyse vibratoire est une technique fondamentale de détection de défauts pour la maintenance prédictive des machines. L'algorithme de transformée de Fourier rapide (FFT) est programmé pour calculer la transformée de Fourier discrète. Les résultats obtenus seront utilisés pour le suivi et le diagnostic des défauts dans les machines. La transmission des données dans le réseau, en notation JSON, se fait en temps réel à l'aide de la gestion des taches.

ملخص

يشهد استخدام شبكات الاستشعار اللاسلكية (RCSF) ا ازدهارا في مختلف المجالات يتم استخدام RCSFs للمراقبة في البيئات التي تعتبر معرضة للخطر أو لا يمكن الوصول إليها.

الهدف من عمل هذه المذكرة هذا هو تصميم شبكة لاسلكية للصيانة التنبؤية للآلات. يتكون RCSF الخاص بنا من عقد ESP32.

تحليل الاهتزاز هو تقنية أساسية لاكتشاف الأخطاء من أجل الصيانة التنبؤية للآلة. تمت برمجة خوارزمية تحويل فورييه السريع (FFT) لحساب تحويل فورييه المنفصل. تستخدم النتائج التي تم الحصول عليها لرصد وتشخيص الأعطال في الآلات. يتم نقل البيانات في الشبكة، بترميز JSON، في الوقت الفعلي باستخدام إدارة المهام.

Abstract

The use of Wireless Sensor Networks (RCSF) in most domains is growing. These RCSF are used for surveillance in environments considered at risk or not accessible.

The goal of this memory work is the design of a wireless network for the predictive maintenance of machines. Our RCSF consists of ESP32 nodes.

Vibration analysis is a fundamental defects detection technique for predictive machine maintenance. The Fast Fourier Transform (FFT) algorithm is programmed to calculate the discrete Fourier transform. The results obtained will be used for monitoring and diagnosing defects in the machines. The transmission of data in the network, in JSON notation, is done in real time using task management.

Remerciements

Nous remercions Dieu pour la volonté, la force, la santé et la patience qu'il nous a données pour faire ce travail.

Nous tenons à exprimer nos sincères remerciements à notre promoteur Madame ***Farida MEDJANI*** *pour avoir d'abord proposé ce thème, pour son suivi continuel tout le long de la réalisation de ce mémoire et à lui exprimer notre gratitude pour ses précieux conseils son soutien continu, sa confiance et sa patience.*

Ce fut un honneur et un plaisir pour nous de travailler sous la supervision de Mr le professeur ***KAZAI Tahar*** *président de Innovation Academy Mila pendant le projet de maîtrise, nous le remercions pour ses observations perspicaces et ses grandes contributions tout au long de l'achèvement de ce travail, et des remerciements particuliers à* ***Ghoulam CHELAT*** *et* ***Ahmed BENMASAOUD****.*

Nous remercions également président ***Rafik BOUHANECHE*** *et examinateur* ***Bessem KAGHOUCHE*** *qui nous ont honorés en acceptant de juger ce modeste horaire de travail.*

Nous remercions également nos professeurs et le chef du département des sciences et technologies d'Abdelhafid Bousouf Mila.

Nous remercions également nos collègues de la promotion Master 2019-2020.

Merci à tous ceux qui nous ont aidés pendant nos études universitaires.

Je vous remercie du fond du cœur.

Samah et Sara

Sommaire

Chapitre 3 : Traitement du signal et techniques d'analyse

Chapitre 4 : Partie réalisation

Liste des abréviations

AFNOR	Association française de normalisation
BSS	Business Support System
COMAU	COnsorzio Machine Utensili
CEI	Commission Electrotechnique Internationale
CPU	Central Processing Unit
CAN	Convertisseur Analogique Numérique
CNA	Convertisseur Numérique Analogique
DSSS	Direct-Sequence Spread Spectrum
DSP	Digital Signal Processor
FHSS	Frequency-Hopping Spread Spectrum
FFT	Fast Fourier Transform
JSON	JavaScript Object Notation
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IR	InfraRed
IP	Internet Protocol
ISR	Interrupt Service Routine
ISO	Organisation internationale de normalisation
I2C	Integrated Circuit
IrDA	Infrared Data Association
LLC	Logical Link Control
LAN	Local Area Network
MPU	Memory Protection Unit
PDA	Personal Digital Assistant
PCI	Process Conception Ingénierie
RTOS	Real-Time Operating System

SDK	Software Development kit
SSID	Service Set Identifier
STA	Single Thread Apartment
SPI	Serial Peripheral Interface
TCP	Transmission Control Protocol
TF	Transformée de Fourier
TFR	Transformation de Fourier Rapide
TFTD	Transformée de Fourier à temps discret
TOR	Tout Ou Rien
T.S	Traitement du Signal
UART	Universal Asynchronous Receiver/Transmitter
USCI	Universal Serial COMMUNICATIONS Interface
UGV	Usinage à grande vitesse
VC	La valeur crête
Wifi	Wireless Fidelity
Wimax	Worldwide Interoperability for Microwave Access
WDS	Wireless Distribution System
XML	Extensible Markup Language

Liste des tableaux

Liste des figures

Introduction générale

Introduction générale

Ces dernières années, de nouveaux types des capteurs dotés de moyens de communication sans fil ont connu une grande évolution et une large utilisation, grâce à leur faible coût et la possibilité de les configurés pour former des réseaux autonomes : réseaux des capteurs sans fil (RCSF). Ce type de capteur dite embarqué récolte les grandeurs physiques (température, lumière, vibration etc.) les traites et les transmettes à travers le réseau.

L'intervention de l'être humaine pour la surveillance, la protection et l'observation de son environnement n'est pas possible dans tous les lieux d'investigation. Le RCSF semble être la solution la plus adéquate pour relever les informations dans les environnements hostiles, auxquels l'homme n'a pas toujours accès. De ce fait les RCSF sont utilisés dans différents domaines : domotique, transport, santé, industriel, surveillance de phénomènes environnementaux…

Notre travail consiste à analyser des signaux vibratoires d'une machine tournante afin de prédire des éventuels défauts ou des pannes. Pour cela nous procédons par la conception d'un RCSF qui a pour but la récolte, le traitement et la transmission de ces signaux vibratoires. Le traitement de signal s'appuyé plus particulièrement sur l'algorithme de transformée de Fourier rapide.

Ce mémoire est organisé en quatre chapitres. Le premier chapitre traite les différents types de la maintenance et les techniques de surveillance. Après, une description des machines tournantes est abordée. Par la suite l'analyse vibratoire des machines tournantes qui constitue un outil puissant de détection des défauts dans la maintenance prédictive, a été décrite.

Dans le deuxième chapitre, nous exposons les réseaux sans fil en général et nous mettons l'accent sur les réseaux des capteurs sans fil (RCSF). Une étude descriptive de RCSF utilisé dans notre travail est détaillée. Dans cette étude nous discutons les composants du réseau à savoir les cartes de développement ESP32 et la conception du réseau via l'environnement de développement Arduino. La transmission de messages dans le réseau en temps réel fait appel à la notation JSON et le système d'exploitation FreeRTOS qui seront développés.

Le troisième chapitre est dédié à l'acquisition et le traitement de signal. Quelques rappels généraux sur la chaine d'acquisition, les capteurs et le traitement de signal sont exposés. L'élaboration d'un programme pour l'algorithme de la transformé de Fourier rapide afin d'extraire les composantes spectrales du signal est illustré.

Nous décrivons notre scénario d'acquisition, de traitement et transmission des données des capteurs sur notre réseau sans fil dans le quatrième chapitre.

Nous achevons notre étude par une conclusion dressant le bilan du travail réalisé.

Chapitre 1

La maintenance

1.1. Introduction

La maintenance occupe une position stratégique dans le milieu industriel, aussi bien sur le plan technologique que sur le plan économique. Elle permet d'augmenter la disponibilité de machines industrielles et d'allonger leur cycle de vie. Principalement elle permet de prévenir les incidents et accidents et surtout d'assurer la sécurité des personnes et des biens.

Initialement, la maintenance industrielle consistait à effectuer des opérations tel que le dépannage, la réparation, le graissage, le contrôle… qui permettent de conserver le potentiel du matériel pour assurer la production avec efficacité et qualité. Ensuite la maintenance a beaucoup évolué à l'aide des outils technologique (les systèmes d'acquisition des données, les logiciels de gestion de bases de données et la technologie sans fil en particulier les réseaux des capteurs intelligents) qui aident à prévenir les pannes et prédire le temps restant de bon fonctionnement de l'équipement.

1.2. Maintenance et techniques de Surveillance

1.2.1. La maintenance

La norme AFNOR indique que la maintenance est « l'ensemble de tous les actions techniques, administratives et de gestions durant le cycle de vie d'un bien, destinées à le maintenir ou à le rétablir dans un état dans lequel il peut accomplir la fonction requise ». Selon cette vision dans une entreprise le rôle de la maintenance à deux objectifs :

Objectifs financiers : [1]

- Réduire au minimum les dépenses de maintenance.
- Assurer le service de maintenance dans les limites d'un budget.

Objectifs opérationnels :

- Maintenir l'équipement dans les meilleures conditions possibles.
- Assurer la disponibilité maximale de l'équipement à un prix minimum.
- Augmenter la durée de vie des équipements.
- Entretenir les installations avec le minimum d'économie et les remplacer à des périodes prédéterminées.
- Assurer un fonctionnement sûr et efficace à tout moment

Figure 1.1 : Objectifs de la maintenance dans une entreprise.

1.2.2. Différents types de maintenance

La définition précédente de la maintenance met l'accent sur deux mots-clés : maintenir et rétablir, qui font référence aux différents types de maintenance. Le premier mot est lié à la notion de surveillance et de prévention d'où le type de maintenance préventive tandis que le deuxième contient la notion de correction : la remise à niveau après perte de fonction, et est lié à l'aspect correctif comme illustré sur l'organigramme ci-dessus.

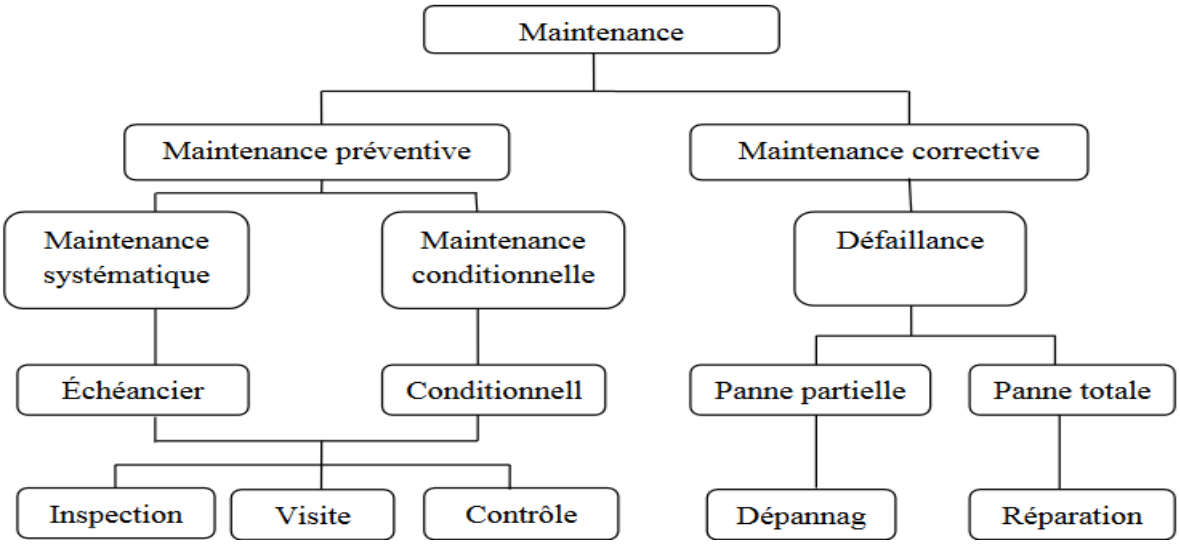

Figure 1.2 : Organigramme de la maintenance.

Nous exposons dans les paragraphes qui suivent les définitions de chaque type de maintenance :

a- Maintenance corrective : est définie comme une maintenance effectuée après défaillance (AFNOR X 60-010). Elle est caractérisée par son caractère aléatoire et requiert des ressources

humaines compétentes et des ressources matérielles (pièces de rechange et outillage) disponibles sur place. La maintenance corrective se débouche sur deux types d'intervention. :

- Le premier type est à caractère provisoire, ce qui caractérise la maintenance palliative.
- Le deuxième type est à caractère définitif, ce qui caractérise la maintenance curative

b- Maintenance préventive : est définie comme une maintenance effectuée dans l'intention de réduire la probabilité de défaillance d'un bien ou d'un service rendu. Les activités correspondantes sont déclenchées selon un échéancier établi à partir d'un nombre prédéterminé d'unités d'usage (maintenance systématique) ou de critères prédéterminés significatifs de l'état de dégradation du bien ou du service (maintenance conditionnelle).

- **Maintenance préventive systématique :** d'après la Norme NF X 60010 «Maintenance préventive effectuée selon un échéancier établi selon le temps ou le nombre d'unités d'usage». Généralement, la maintenance préventive s'adresse aux éléments dont le coût des pannes est élevé, mais ne revenant pas trop cher en changement (les meilleurs exemples sont le changement systématique de l'huile, changement de la courroie de synchronisation,...).

Ces avantages :

- Planification des arrêts d'entretien.
- Optimisation de l'intervention (préparation)
- Limite les risques de panne.

Ces inconvénients :

- Coût de maintenance élevé.
- Approche statistique.
- Risque induit par une intervention parfois non nécessaire.

- **La maintenance préventive conditionnelle :** d'après la Norme NF X 60010, la maintenance préventive conditionnelle définit comme « une maintenance préventive subordonnée à un type d'événement prédéterminé (auto diagnostic, information d'un capteur, Mesure d'une usure, révélateur de l'état de dégradation du bien) »

Ces avantages :

- Optimisation de la durée de fonctionnement.
- Optimisation de l'intervention (préparation).
- Evaluation réelle de l'état de la machine.
- Evite les pertes de production.

Ces inconvénients :

- Coût de l'investissement (homme / matériel).
- Limiter à un programme suivi.

Les différentes étapes de l'organisation de la maintenance conditionnelle sont définies par l'organigramme suivant :

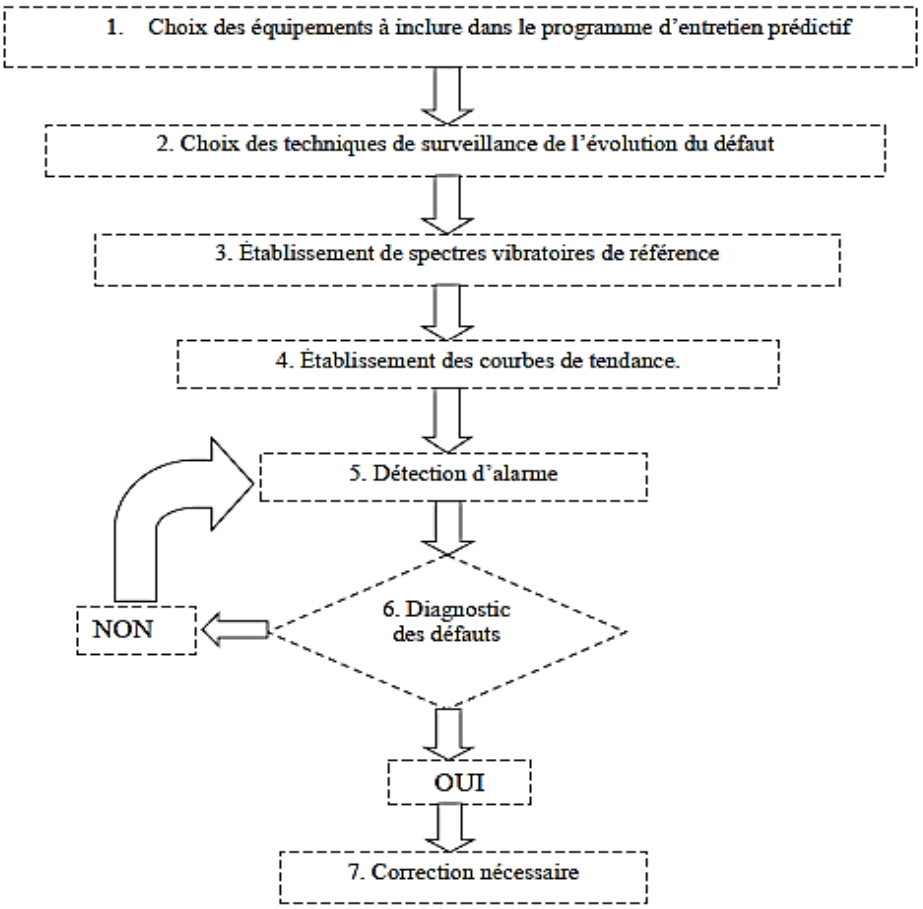

Figure 1.3 : L'organisation de la maintenance conditionnelle [2].

- **Etape 1 :** comprend la codification des équipements, l'inventaire et la sélection du type d'entretien appliqué à chaque équipement,
- **Etape 2 :** comprend le choix de la technique de mesure, leur périodicité, leur endroit et repérage, la création d'une route et des dossiers de suivi,
- **Etape 3 :** s'occupe de la détermination des alarmes,
- **Etape 4 :** comprend la prise de mesure, l'enregistrement et la gestion des données vibratoires,
- **Etape 5 :** vérification du seuil d'alarme,
- **Etape 6 :** déterminer l'élément défaillant.
- **Etape 7 :** Changer ou corriger l'élément défaillant

1.2.3. Niveaux de maintenance

Les interventions de maintenance peuvent être classées par ordre croissant de degrés de la défaillance de la machine et la complexité, la norme NF X 60-010 définit, à titre indicatif, 5 niveaux de maintenance des opérations. Ces niveaux sont regroupés dans le tableau suivant

Tableau 1.1 : Niveaux de la maintenance.

Niveau	1	2	3	4	5
compétence	Technicien quelconque	Technicien habilité	Technicien Spécialisé et habilité	Technicien très spécialisé	constructeur
lieu	Sur place	Sur place	Sur place ou dans locale spécifique	Atelier spécialisé	Atelier centralisé ou extérieur
outillage	Basique (pinceau, burette)	Portatif standard (clefs, pinces)	Spécifique (poste de soudage,..)	Collectif et spécifique	Défini et spécifique au constructeur
instructions	Consignes d'utilisation et d'entretien	Consignes de réglage, Gammes de maintenance	Consignes, Gammes, plans & schéma constructeur	Consignes, Gammes, plans & schéma constructeur	Documentation complète constructeur
Pièce de rechange	Consommable (fluides)	Disponible à proximité	Disponibilité Approvisionnées par le magasin	Disponibilité Approvisionnées par le magasin	Approvisionnées en extérieur
Essais et /ou contrôles	Visuel	Contrôle conventionnelle	Sure bancs équipes	Sure bancs équipes	Protocole à établir entre constructeur & utilisateur
Exemple	Vérification de Nivaux, changement de filtres, Graissage…	Réglage après changement de fab. Echange de modules	Intervention sur circuit électronique	Reprogrammation ou réparation complète d'un système	Reconstruction ou rénovation d'une machine

1.2.4. Politique de maintenance

La politique de maintenance à choisir est basée essentiellement sur de paramètres économique et humaine. Dans le but de créer une politique qui peut réduire les coûts de

maintenance, un procédé de suivi de l'état de fonctionnement d'une pièce ou d'une machine doit être trouvé afin d'améliorer la disponibilité et pour évaluer de façon plus précise le risque de panne. L'enjeu consiste réaliser un système de mesures qui consiste à limiter le nombre de paramètres et de mesures à réaliser et de trouver des indicateurs externes, facilement accessibles. Pour cela, il est nécessaire d'effectuer une analyse statistique pour déterminer les pannes fréquentes et une analyse technologique des composants les plus problématiques (étude faite par deux fabricants français des machines UGV, PCI et COMAU et deux laboratoires de recherche, le LSIS et le LARAMA [3].

Toutes les machines que l'on dit tournantes (Turbines, pompes, moteurs, compresseurs, alternateurs, centrifugeuses, ventilateurs...) ont des points communs ; elles comprennent des organes en rotation et sont composés des pièces fragiles (roulements, engrenages etc...) soumis à des contraintes mécaniques importantes et à des environnements industriels difficiles. Les sources de défaillances sont donc multiples :

- écaillage d'un roulement
- rupture d'une dent d'un engrenage
- désalignement d'un des axes, etc....

Ces défauts peuvent s'avérer lourds de conséquences pour certaines machines vitales d'un processus de production. Pour anticiper des arrêts de production imprévus et les pertes économiques qui en découlent, il faut surveiller en permanence ces équipements et poursuivre tous les signes précurseurs de défauts avant qu'il ne soit trop tard. Pour cela il existe une variété de techniques [3].

1.2.5. Techniques de surveillance

L'indicateur d'état de dégradation (ou de performance) relevé périodiquement garanti la surveillance d'un composant de machine. Différentes techniques d'analyse existent parmi elles :

- L'analyse vibratoire
- l'analyse des huiles et des lubrifiants
- la thermographie infrarouge
- l'émission acoustique
- la variation de résistance dans un circuit électrique….

Le choix de l'indicateur dépend du type de machine à étudier et du type de défaillance que l'on souhaite détecter. Un indicateur de type vibratoire, pour les machines tournantes, permet de détecter la plupart des défauts. On établit une courbe d'évolution de l'indicateur au cours du temps.

Sur cette courbe, on définit différents seuils correspondant à un niveau d'alerte, à une alarme, à un niveau de défaillance. Ces niveaux sont établis soit par expérience soit en appliquant une norme (pour les roulements, on utilise des abaques de sévérité vibratoire pour définir les différents seuils).

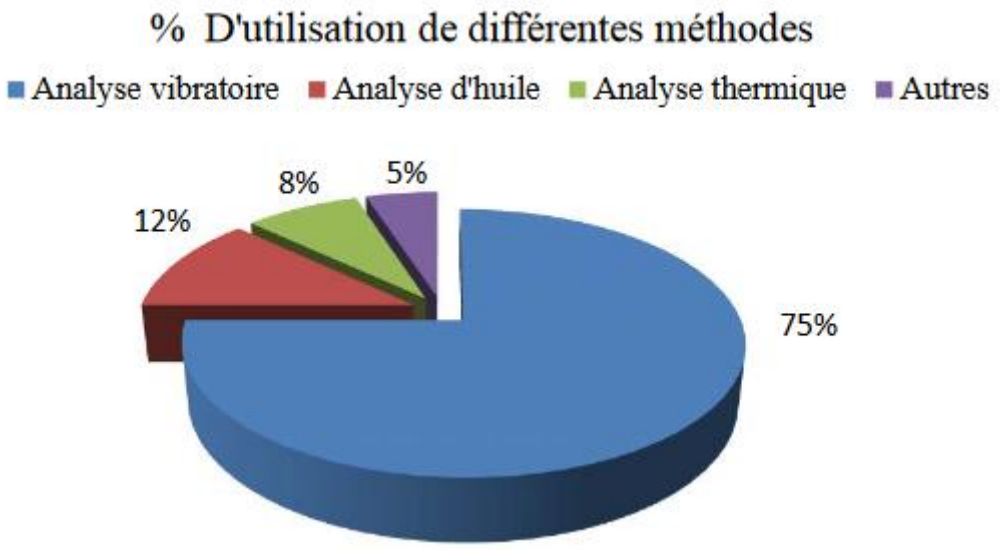

Figure 1.4 : Différentes méthodes d'analyse.

- **L'analyse vibratoire :** Toutes les machines, et particulièrement les machines tournantes, vibrent et l'image vibratoire de leurs vibrations a un profil très particulier lorsqu'elles sont en état de bon fonctionnement. Dès qu'un ou plusieurs défauts apparaissent, l'allure de cette image change, ce qui permet, de quantifier l'intervention. La plupart des défauts mécaniques peuvent être détectés par cette technique [4].

L'analyse vibratoire permettra donc de mettre en évidence un grand nombre de problèmes entraînant une diminution de la durée de vie des éléments de la machine. Les défauts détectés par analyse vibratoire sont :

- résonance de structure
- jeu de palier
- balourd ou déséquilibre de masse
- mauvaise fixation
- défauts de roulements
- distorsion ou tension trop élevée des courroies
- mauvais alignement d'une ligne d'arbre
- défauts d'engrenage...,

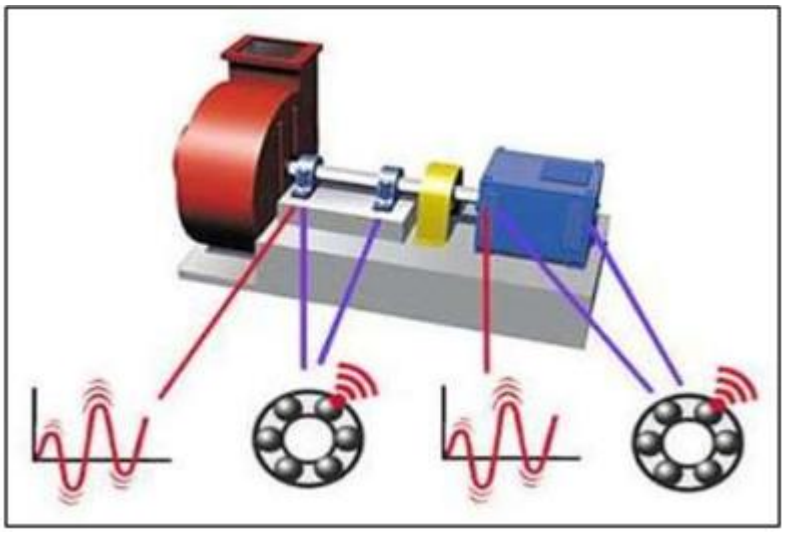

Figure 1.5 : Sources de vibration.

- **l'analyse des huiles et des lubrifiants** : est appliquée à toutes les machines contenant des fluides de lubrification (réducteurs, motoréducteurs, moteurs thermiques...). Elle consiste à prélever un échantillon d'huile et de l'analyser (particules d'usure) pour déduire l'état de l'équipement [5].
- **la thermographie infrarouge :** Un défaut sur un équipement se traduit toujours par une élévation de température, cette élévation, si elle ne se voit pas dans la visible, apparaît instantanément dans l'infrarouge, ainsi, la thermographie infrarouge est une technologie efficace de maintenance prédictive pour localiser les problèmes rapidement, en toute sécurité. Avec la thermographie infrarouge on peut visualiser les défauts avant qu'une panne sur les installations ne se produit et qu'elle ait des conséquences fâcheuses : perturbation ou arrêt de la production jusqu'à un début d'un incendie dans le pire de cas**.** [5]

La totalité de l'énergie émise par une surface et la distribution spectrale de cette énergie varie avec la température, en général aux basses températures le maximum de radiations est dans le moyen infrarouge (2 à 20μm). Au fur et à mesure que la température s'élève, l'énergie émise s'élève et le gros de l'énergie va vers des longueurs d'ondes de plus en plus courtes.

On parle d'ultraviolet, fréquence au-delà du violet, alors que la longueur d'onde du violet est plus courte. On parle aussi d'infrarouge, fréquence en delà du rouge, alors que la longueur d'onde du rouge est plus longue. C'est à dire que l'infrarouge est moins puissant, moins énergétique que l'ultraviolet. En parlant en longueur d'onde, l'infrarouge se situe donc au-delà de 0,8 μm, et il regroupe les longueurs d'onde des rayonnements de la matière plus froide que celle chaude comme la lampe à incandescence, est vue directement avec nos yeux. Nous ne voyons donc pas le rayonnement infrarouge avec nos yeux. La lampe dite "infrarouge" est moins alimentée que la

lampe à incandescence normale : elle consomme moins d'énergie et est donc plus froide. Elle est conçue pour émettre principalement dans l'infrarouge, pour chauffer moins qu'une lampe normale. Ainsi, la thermographie est rapide, tout autant que sa mise en œuvre, elle procure une vision immédiate, globale et discriminative. C'est une méthode puissante de vision du froid, du plus chaud ou du plus froid**.**

Différents instruments de mesure existe :

- **Le thermomètre infrarouge :** Les thermomètres infrarouges sont particulièrement recommandés dans des domaines où la mesure de température par contact est impossible :
  - nécessite une réponse rapide.
  - sur des objets en mouvement.
  - Derrière une fenêtre.
  - si le thermomètre peut être détruit par le contact.
  - si un profil de température existe sur la surface- si la température peut être affectée par le contact.
- **La caméra infrarouge :** La caméra infrarouge ne mesure pas les températures, mais les rayonnements, alors que, visualisée par le thermographe, l'image thermique que la caméra fournit pourra être transformée en thermo gramme, en images des températures. C'est bien ce que l'on cherche pour déterminer l'état de santé des matériels et surtout prévoir ce qui se passera dans l'avenir, en maintenance prédictive. Le thermographe, aidé de sa caméra, va voir dans l'infrarouge les objets froids et, parmi ces objets, ceux qui sont anormalement chauds ou anormalement froids. Ayant vu, le thermographe pourra quantifier et fournir une cartographie des températures.
- **L‘analyse acoustique :** permet de détecter tout bruit anormal à l‘aide de microphones placés à distance de l‘équipement.
- **Le contrôle par ultrasons :** permet de détecter des défauts de faibles amplitudes à hautes fréquences tels que l‘initiation de la dégradation d‘un roulement

1.2.6. Choix d’une technique d’analyse

Chaque méthode est appliquée dans un domaine déterminé. Dans une plage de fréquences situées généralement entre quelques Hertz et plusieurs dizaines de KHz, l‘analyse vibratoire est la mieux adaptés pour détecter les défauts liés à la cinématique et la structure de la machine. Au-delà là de 20 KHz, il est souvent préférable d‘utiliser un contrôle par ultrasons ou par émission acoustique. L‘analyse acoustique se limite à la détection de bruits dans les fréquences audibles,

mais lorsque la dégradation d'un roulement se manifeste en une fréquence audible, il est souvent trop tard pour intervenir. L'analyse d'huile consiste principalement à analyser les particules présentes dans l'huile, ce qui va révéler une usure anormale d'un ou plusieurs organes. Elle doit être appliquée dans le cas de machines où l'huile joue un rôle primordial et lorsque l'analyse des débris d'usure est significative**.**

Tableau 1.2 : Techniques d'analyse de l'état d'une machine tournante**.**

	Principaux avantages	**Principales limitations**	**Champs d'applications privilégié**
Analyse vibratoire	- détection de défauts à un stade précoce - possibilités de réaliser un diagnostic approfondi - autorise une surveillance relativement continue -permet de surveiller -l'équipement à distance (télémaintenance)	- spectres parfois difficile à interpréter - dans le cas de la surveillance continue, installations relativement Coûteuses	- détection des défauts de tous les organes cinématiques de la machine (balourd, défauts coûteuses d'alignement, jeux, etc..) et de sa structure.
Analyse d'huile	- détection d'une pollution anormale du lubrifiant avant que celle-ci n'entraîne une usure ou un échauffement - possibilités de connaître. l'origine de l'anomalie par analyse des particules	- ne permet pas de localiser précisément le défaut - nécessite de prendre nombreuses précautions dans le prélèvement de l'échantillon	- contrôle des propriétés physico-chimiques du lubrifiant, détection d'un manque de lubrifiant analyse des éléments d'usure

Les Thermographie	- permet de réaliser un contrôle rapide de l'installation - interprétation souvent immédiate des résultats	- détection de défauts à un stade moins précoce que l'analyse vibratoire - contrôle limité à ce que « voit »la caméra (échauffements de surface) - ne permet pas de réaliser un diagnostic approfondi	- détection de tous les défauts engendrant un échauffement (manque de lubrification en Particulier)
Analyse acoustique	- permet de détecter l'apparition des défauts audibles - autorise une surveillance continue	- sensibilité au bruit ambiant - diagnostic souvent difficile à réaliser - problèmes de répétabilité des mesures	- détection d'un bruit inhabituel pouvant ensuite être analysé par analyse vibratoire

La détection précoce de tous les défauts susceptible d'être rencontrés sur une machine tournante n'est pas possible par une seule technique commune mais la combinaison de plusieurs techniques permet un diagnostic plus fiable et plus rapide.

La surveillance des machines ne se limite pas juste à détecter la présence d'un défaut, mais il est aussi nécessaire de pouvoir réaliser un diagnostic approfondi pour le localiser précisément et quantifier sa sévérité .L'analyse vibratoire est une technique qui permet de réaliser ce diagnostic [7].

1.3. Machine tournante et l'analyse vibratoire

1.3.1. Machine tournante

Une machine est un ensemble des pièces mécaniques, hydrauliques ou électriques concourant à exercer une ou plusieurs fonctions données et en particulier, l'application d'une force modulée ou non, destinée à vaincre une résistance ou à assurer un mouvement avec ou sans transmission de force. Les machines tournantes sont des systèmes dans lesquels on peut distinguer : un rotor, une structure, des liaisons.

- **Le rotor :** est une structure dont les éléments tournent autour d'une ligne de rotation définie en fonction de l'état mécanique de ce dernier, en général distincte d'un axe (ligne droite). Le rotor fabriqué avec plusieurs matériaux (acier, cuivre, bois, plastique...) réalise une fonction bien définie : (manipulation fluide, de solide, parcours dans un champ électromagnétique,...etc.).
- **La structure :** la structure non rotative comprend les éléments essentiels suivants :
  - Les coussinets : ils sont de faible dimension au droit des tourillons des rotors. Des bagues peuvent être substituées aux coussinets (roulements).
  - Les paliers : relient les coussinets (bagues) au stator
  - Le stator :(enveloppe de la machine) : il contient les éléments essentiels, à savoir les circuits magnétiques dans les machines électriques, les ailettes pour les turbomachines.
  - Le massif : il peut prendre des formes beaucoup plus variées que celles des systèmes terrestres dont les massifs sont liés au radier.
  - Le radier : c'est un élément spécifique au système. Il assure la liaison entre le massif et le sol et a pour mission de diminuer les pressions exercées au sol dans des limites acceptables. C'est par lui que les séismes perturbent les machines tournantes
- **Les liaisons :** Le rotor est lié à la structure non rotative par des liaisons qui assurent le guidage des rotors. Les liaisons sont classées dans trois ensembles : à fluide, à roulements, magnétique.

1.3.2. Classification des machines tournantes

Les machines tournantes peuvent être classées selon plusieurs critères à savoir :

- En fonction du nombre (n) de liaisons.
- En fonction de leur état rigide ou flexible.

- **En fonction du nombre (n) de liaisons :** Les liaisons disposées dans les paliers guident le rotor par rapport au stator avec deux liaisons, les forces moyennes (statiques) applique sur chacune d'elles sont déterminées à partir des forces appliquées sur le rotor dans une direction donnée par

rapport à l'axe qui joint le centre des liaisons, cette configuration est dite isostatique. Si le nombre de liaisons est supérieur à deux, les forces appliquées sur les liaisons dépendent des impédances au droit des liaisons, cette configuration est dite hyperstatique.

- **Classification selon l'état (rigide ou flexible) :** Les matériaux qui constituent les rotors étant doués de masse et élasticité, les rotors se déforment sous l'action des forces centrifuges dues aux balourds et à la vitesse de rotation. Cependant lorsque les déformations sont faibles, il est possible de considérer le rotor en état rigide, tout état non-rigide est dite flexible.

1.3.3. Les vibrations

Une vibration est la variation avec le temps d'une grandeur caractéristique du mouvement ou de la position d'un système mécanique, lorsque la grandeur est alternativement plus grande et plus petite, qu'une certaine valeur moyenne ou de référence (AFNOR 90.001).

Une vibration est caractérisée par trois paramètres :

- **La fréquence :** C'est le paramètre caractérisant la rapidité d'oscillation (c'est le nombre d'oscillations par seconde), elle s'exprime en Hz (Hertz).
- **La phase :** Elle permet de déterminer la position de la particule d'un corps oscillant à instant donnée.
- **L'amplitude :** Elle s'exprime soit en déplacement, soit en vitesse, soit en accélération.

1.3.4. Différentes formes de vibration

On classe généralement les vibrations d'après l'évolution de la variable considérée dans le temps (périodicité). On distingue ainsi les vibrations :

- Harmoniques
- Périodiques
- Apériodiques

1.3.4.1. Vibration harmonique

Dont le diagramme amplitude-temps, est représenté par une sinusoïde (figure 3.1)

$$S(t) = s_0 \sin(\omega t + j) \qquad 1.1$$

- ω : Vitesse ou pulsation du mouvement en [rd/s].
- f : Fréquence du mouvement en [Hz].
- j: Phase du mouvement par rapport à un repère dans le temps en [rd].

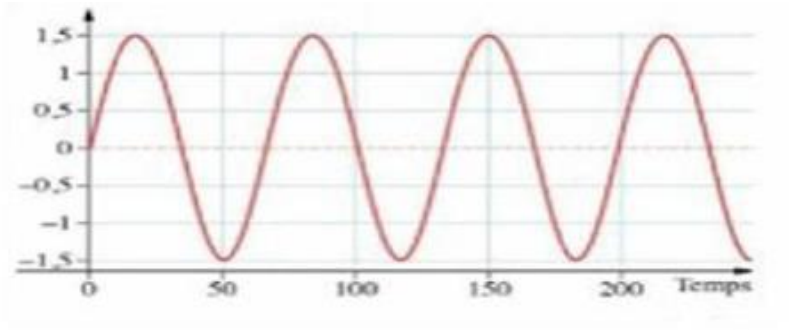

Figure 1.6 : Vibration harmonique (une sinusoïde).

Le meilleur exemple d'une vibration harmonique est celle généré par le balourd d'un rotor en mouvement.

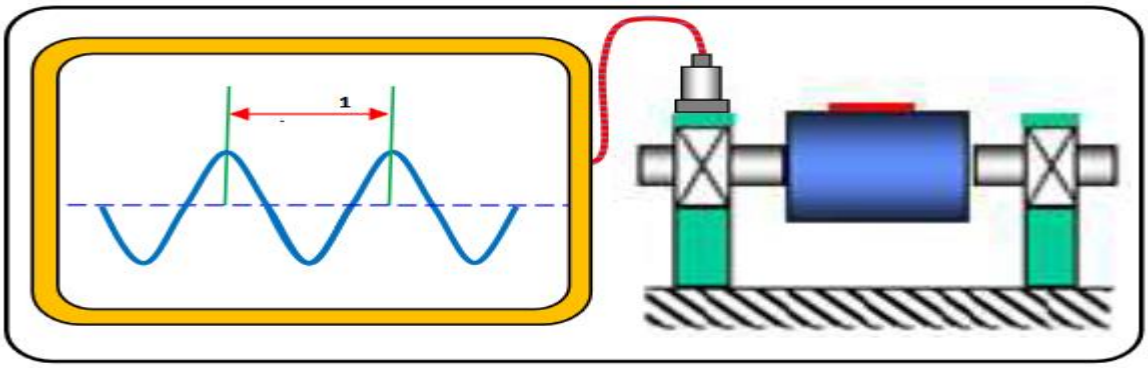

Figure 1.7 : Signal vibratoire généré par un balourd**.**

Si on mesure l'amplitude du signal vibratoire délivré par un capteur placé sur le palier supportant un rotor déséquilibré **(figure 3.2),** on remarque que l'amplitude sera maximale lorsque la masse sera en haut du rotor et elle sera minimale lorsqu'elle sera en bas, et ainsi de suite, à chaque tour du rotor. Le signal vibratoire est périodique de période 1 tour, donc de fréquence égale à la fréquence de rotation du rotor.

1.3.4.2. Vibration périodique

Est telle qu'elle se produit exactement après un certain temps appelé période. Une telle vibration est créé par une excitation elle-même périodique c'est le cas le plus fréquent rencontré dans les machines. La vibration périodique est composée de plusieurs vibrations harmoniques (**Figure 3.3).** Elle est décrite par l'équation suivant :

$$S(t) = \sum_{i=0}^{n} S_i \sin(\omega_i t + \varphi) \qquad \textbf{1.2}$$

$\omega_1, \omega_2 \ldots\ldots \quad \omega_n$ **:** sont des multiples de ω_0 (pulsation fondamentale)

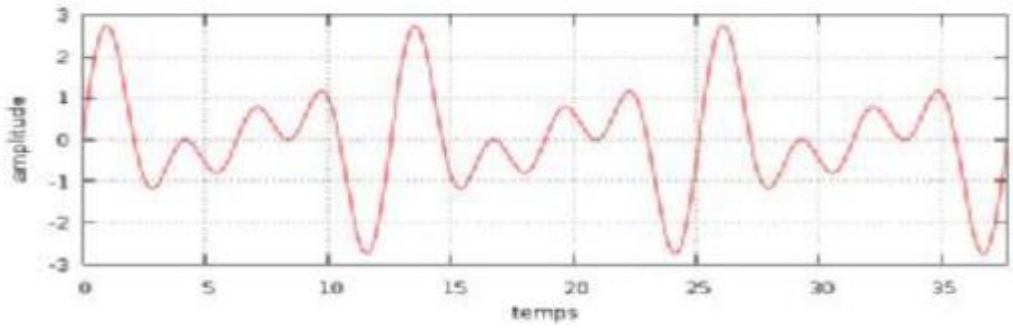

Figure 1.8 : Vibration périodique.

1.3.4.3. Vibration apériodique

Son comportement temporel est quelconque, c'est-à-dire que l'on n'observe jamais de reproductibilité dans le temps, c'est le cas des chocs qu'enregistre sur quelques systèmes (Figure 1.9). Elle est décrite par l'équation suivant :

$$S(t) = \sum_{i=0}^{n} S_i \sin(\omega_i t + \varphi) \qquad \textbf{1.3}$$

$\omega_1, \omega_2 \ldots\ldots \quad \omega_n$ **:** sont des multiples de ω_0 (pulsation fondamentale)

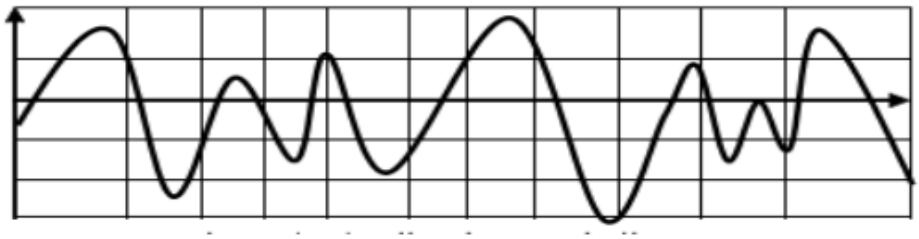

Figure 1.9 : Vibration apériodique**.**

Une machine tournante quelconque en fonctionnement génère des vibrations que l'on peut classer de la façon suivante :

1. Les vibrations périodiques de type sinusoïdal simple (**Figure 1.6**) ou sinusoïdal complexe **(Figure 1.8)** représentatives du fonctionnement normal ou anormal d'un certain nombre d'organes mécaniques (rotation de lignes d'arbres, engrènements,...) ou d'un certain nombre d'anomalies (déséquilibre, désalignement, déformations, instabilité de paliers fluides, déversement de bagues sur roulements, ...)
2. Les vibrations périodiques de type impulsionnel **(Figure 1.10)** sont appelées ainsi par référence aux forces qui les génèrent et à leur caractère brutal, bref et périodique. Ces chocs peuvent être produits par des événements normaux (presses automatiques,

broyeurs à marteaux, compresseurs à pistons, ...) ou par des événements anormaux comme l'écaillage de roulements ou un défaut sur des engrenages, un jeu excessif, ...

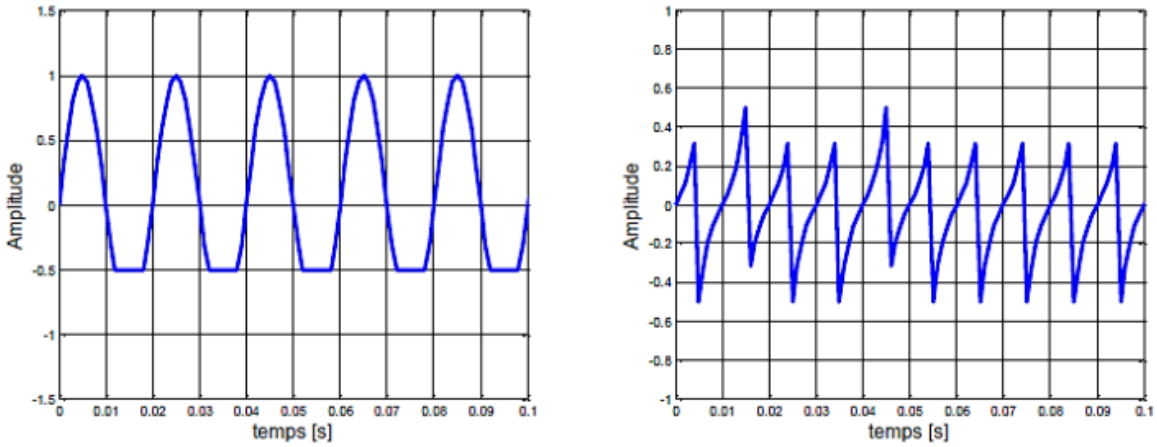

Figure 1.10 : Signaux impulsionnels [4].

3. Les vibrations aléatoires de type impulsionnel **(Figure 3.11)** peuvent, par exemple, être générées par un défaut de lubrification sur un roulement, la cavitation d'une pompe, ...

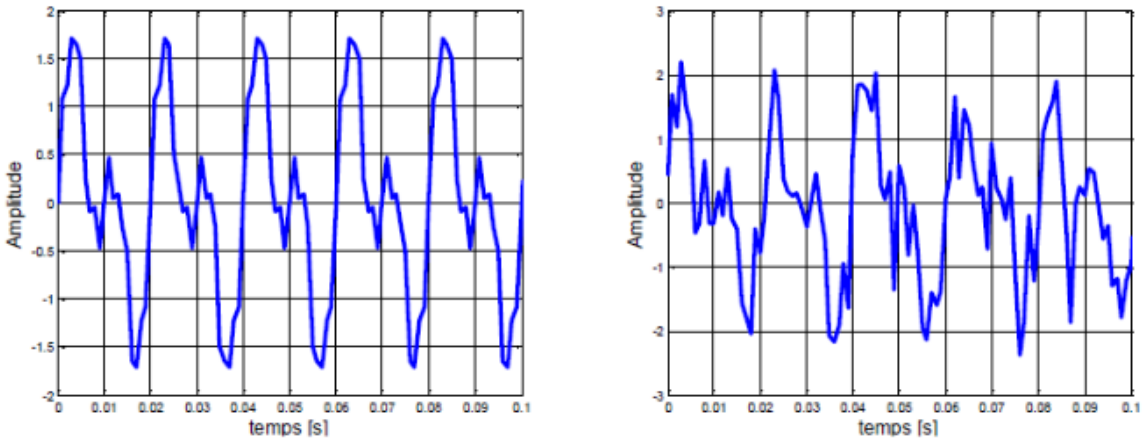

Figure 1.11 : Signaux impulsionnels complexes [4].

1.3.5. Les grandeurs de mesure

Les quantités physiques de base qui définissent les vibrations sont le déplacement, la vitesse et l'accélération

- **Déplacement :** Grandeur vectorielle qui définit le changement de position d'un corps ou du point matériel par rapport à un système référence

- **Vitesse :** Cette quantité exprime le déplacement par unité de temps. Elle est liée à l'énergie vibratoire. Lors de mesures vibratoires, l'amplitude du déplacement et sa direction peuvent changer au cours du temps et la vitesse n'est généralement pas constante. La relation suivante existe :

Vitesse = Déplacement x 2 π x fréquences de vibrations **1.4**

- **L'accélération :** est la variation de vitesse par unité de temps. Elle est proportionnelle à la force générant l'impact ou à toute autre force extérieure. Lors de mesures vibratoires, l'amplitude de la vitesse et sa direction peuvent changer au cours du temps et l'accélération n'est généralement pas constante. La relation suivante existe [8].

Accélération = vitesse x 2 π x fréquences de vibration **1.5**

1.3.6. Analyse vibratoire

En partant du principe que toute machine tournante vibre, l'analyse vibratoire se présente comme étant une solution efficace pour surveiller en temps réel l'état de ces machines, afin de détecter de manière précoce leurs défaillances pour anticiper l'apparition d'une panne et ainsi assurer la continuité de service de la chaine de production. Aussi pour qualifier l'état général d'une machine on procède par mesure des vibrations au niveau global et les comparer à des normes ou des mesures précédentes. Cette stratégie de surveillance consiste en un suivi de l'évolution dans le temps d'un ou de plusieurs indicateurs (déplacement, vitesse ou accélération). L'analyse vibratoire est considérée comme un type de maintenance préventive conditionnelle.

Un capteur de vibrations, détecte les vibrations mécaniques de la machine et converti le signal mécanique en signal électrique qui sera acheminé à un mesureur de vibrations pour l'analyser et afficher la valeur globale. Parmi les capteurs de vibrations les plus utilisés, on trouve le proximètre (mesure de déplacement), le vélocimètre (mesure de vitesse) et le l'accéléromètre (mesure de l'accélération).

La sensibilité des capteurs représente la proportionnalité de la sortie électrique exploitable (tension, charge, courant) par rapport au paramètre de vibration (accélération, vitesse, déplacement). Elle s'exprime en terme de : sortie électrique / paramètre de vibration.

Le suivi se fait de deux façons différentes : **continu** (en ligne) ou **périodique** sous forme de rondes plus ou moins espacées dans le temps. La périodicité des mesures est adaptée en fonction de l'évolution des indicateurs. Plus une augmentation est rapide, plus les contrôles doivent être rapprochés. Il est obligatoires que les conditions de fonctionnement de la machine ainsi que les

conditions de mesure (vitesse, charge, températures etc.) doivent être rigoureusement identiques d'une mesure à l'autre.

Le choix de la grandeur à mesurer (déplacement, vitesse ou accélération) pour la surveillance d'une machine tournante dépend essentiellement du défaut recherché et la plage de fréquences dans laquelle il est susceptible de s'exprimer. La grandeur retenue est appelée paramètre ou indicateur de surveillance. Le domaine de surveillance pour chaque indicateur est résumé sur le tableau 1.3.

Tableau 1.3 : Domaine de surveillance pour chaque indicateur**.**

Indicateur (Niveau global)	**Domaine de surveillance**
Déplacement (µm c/c)	Phénomènes lents basses fréquences [2–100 Hz] : balourd, désalignement, instabilités de paliers etc.
Vitesse (mm/s)	Moyennes fréquences [1000 Hz] : balourd, lignage, instabilités de paliers, cavitation, passage d'aubes, engrènement etc.
Accélération (mm/s^2)	Phénomènes très rapides Hautes fréquences [20000 Hz] : engrenages, roulements, passages d'ailettes, cavitation…)

L'analyse vibratoire poursuit deux objectifs :

- **la détection des défauts :** étude des valeurs du niveau global des vibrations
- **l'analyse détaillée des défauts :** étude des contenus fréquentiels des signaux à l'aide outils sophistiqués du traitement de signal (spectre, cepstre, analyse d'enveloppe etc...)

  On utilise à cet effet des paramètres calculés :

  - soit dans le domaine temporel
  - soit dans le domaine fréquentiel
  - soit dans les deux à la fois

1.4. Conclusion

Dans ce chapitre, premièrement les concepts nécessaires à la compréhension de la maintenance prédictive sont présentés. Par la suite l'analyse vibratoire des machines tournantes qui constitue un outil puissant de détection des défauts est détaillée. Après une définition de la vibration, de sa nature et de ses amplitudes, on détermine le choix des grandeurs physiques à mesurer. De plus l'évaluation de ces vibrations dans le cadre de la maintenance prédictive est décrite.

Chapitre 2

Le réseau sans fil

2.1. Introduction

L'observation et le contrôle des phénomènes physique tels que la température, la pression ou encore les vibrations est essentiel pour de nombreuse applications industrielles est scientifiques. Dans le domaine industriel la surveillance des machines pourrait considérablement augmenter la qualité des produits et prévenir les pannes, les incidents, les accidents et surtout d'assurer la sécurité des personnes et des biens. La solution pour acheminer les données du capteur jusqu'au contrôleur central, était le câblage qui avait comme principaux défauts d'être couteux et encombrant. Le développement technologique au cours de dernières décennies ont permis un développement fulgurant des technologies en matière de communication à travers les réseaux sans fil. Un Réseau de Capteurs Sans Fil (RCSF) est un ensemble de nœuds, communiquant sans fil et capable de récolter et transmettre des données et de réagir en cas de besoin.

Nous présentons dans ce chapitre les différents concepts liés aux réseaux sans fil et le WiFi, en mettant la lumière sur la création des réseaux maillés à base des nœuds ESP32. Une description détaillé de module ESP32 est présenté. Des exemples de programmation de réseau maillé dans Arduino IDE sont discutés

2.2 Définition des réseaux sans fil

Un réseau sans fil (en anglais : Wireless network) est un réseau informatique numérique qui connecte différents postes ou systèmes entre eux. Il peut être associé à un réseau de télécommunications pour réaliser des interconnexions à distance entre nœuds.

Les réseaux sans fil sont basés sur une liaison utilisant des ondes radioélectriques (radio et infrarouges) au lieu des câbles habituels. Il existe plusieurs technologies se distinguant d'une part par la fréquence d'émission utilisée ainsi que le débit et la portée des transmissions.

Grâce aux réseaux sans fil, un utilisateur a la possibilité de rester connecté dans un périmètre géographique plus ou moins étendu et permettent de relier très facilement des équipements distants d'une dizaine de mètres à quelques kilomètres [9].

2.2.1 Techniques de transmission dans les réseaux sans fil

Il existe principalement deux méthodes pour la transmission dans les réseaux sans fil :

2.2.1.1 Transmission par les ondes infrarouges

La transmission par les ondes infrarouges nécessite que les appareils soient en face l'un des autres et aucun obstacle ne sépare l'émetteur du récepteur (car la transmission est directionnelle) cette technique est utilisée pour créer des petits réseaux de quelques dizaines de mètres (télécommande de : télévision, les jouets, voitures...).

2.2.1.2 Transmission par les ondes radios

La transmission par les ondes radios est utilisée pour la création des réseaux sans fil sur plusieurs kilos mètres. Les ondes radios ont l'avantage de ne pas être arrêtés par les obstacles car elles sont émises d'une manière omnidirectionnelle. Le problème de cette technique est les perturbations extérieures qui peuvent affecter la communication à cause de l'utilisation de la même fréquence par exemple. [10]

2.2.2 Technologies sans fil

Les technologies dites « sans fil », la norme 802.11 en particulier, facilitent et réduisent le coût de connexion pour les réseaux de grande taille. Avec peu de matériel et un peu d'organisation, de grandes quantités d'informations peuvent maintenant circuler sur plusieurs centaines de mètres, sans avoir recours à une compagnie de téléphone ou de câblage. Ces technologies peuvent être classées en quatre parties [11] :

- WPAN (Wireless Personal Area Network)
- WLAN (Wireless Local Area Network)
- WWAN (Wireless Wide Area Network)
- WMAN (Wireless Metropolitan Area Network)

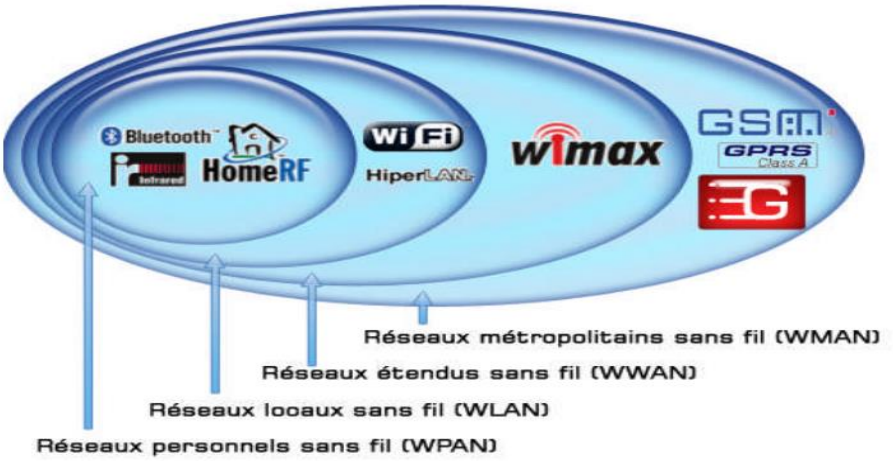

Figure 2.1 : Classification des réseaux sans fil.

2.2.3 Topologie de réseau

Une topologie de réseau, en informatique, est une définition de l'architecture (physique ou logique) d'un réseau. Elle définit les connexions entre ces postes et la hiérarchie éventuelle entre eux ; elle peut déterminer la façon dont les équipements sont interconnectés ou la représentation, spatial du réseau (topologie physique) ou la façon dont les données transitent dans les lignes de communication (topologie logique). Plusieurs topologies existent :

2.2.3.1 Le réseau en anneau

Un réseau a une topologie en anneau quand toutes ses stations sont connectées en chaine les unes aux autres par une liaison bipoint de la dernière à la première. Chaque station joue le rôle de station intermédiaire. Chaque station qui reçoit une trame, l'interprète et le réémet à la station suivante de la boucle si c'est nécessaire. La défaillance d'un hôte rompt la structure d'un réseau en anneau si la communication est unidirectionnelle.

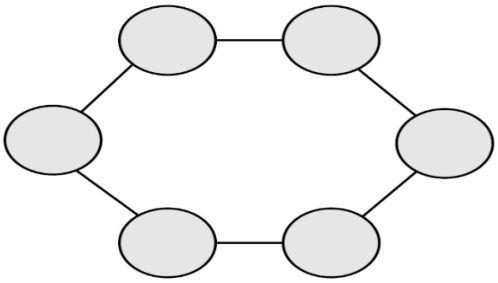

Figure 2.2 : Topologie en anneau.

2.2.3.2 Le réseau hiérarchique

Aussi connu sous le nom de Réseau en arbre, il est divisé en niveaux. Le sommet, de haut niveau, est connecté à plusieurs nœuds de niveau inférieur dans la hiérarchie. Ces nœuds peuvent être eux-mêmes connectés à plusieurs nœuds de niveau inférieur. Le tout dessine alors un arbre ou une arborescence. Le point faible de ce type de topologie réside dans l'ordinateur "père" de la hiérarchie qui s'il tombe en panne interdit alors toute communication entre les deux moitiés du réseau. [12]

2.2.3.3 Le réseau en bus

La topologie Réseau en bus (informatique) est représentée par un câblage unique des unités réseaux. La défaillance d'un nœud (ordinateur) ne scinde pas le réseau en deux sous-réseaux. Ces unités sont reliées de façon passive par dérivation électrique ou optique.

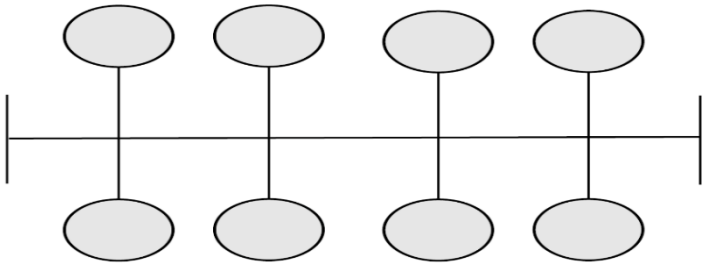

Figure 2.3 : Topologie en bus.

2.2.3.4 Le réseau en étoile

La topologie Réseau en étoile aussi appelée Hub and spoke (point à point) est la topologie la plus courante actuellement. Omniprésente, elle est aussi très souple en matière de gestion et de dépannage d'un réseau, la panne d'un nœud ne perturbe pas le fonctionnement global du réseau.

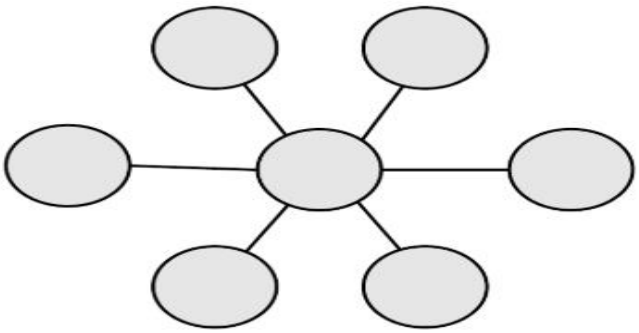

Figure 2.4 : Topologie en étoile.

2.2.3.5 Le réseau linéaire

Il a pour avantage son faible coût de déploiement, mais la défaillance d'un nœud (ordinateur) peut scinder le réseau en deux sous-réseaux.

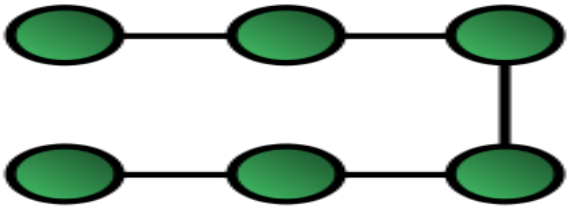

Figure 2.5 : Topologie linéaire.

2.2.3.6 Le réseau maillé

Un réseau maillé (ou Mesh) est un réseau local (LAN : Local Area Network), un réseau local sans fil (WLAN : Wireless Local Area Network) ou un réseau local virtuel (VLAN : LAN virtuel) qui utilise comme méthodes de connexion décentralisée une topologie de maillage complet ou partiel, c'est-à-dire où les nœuds du réseau sont reliés les uns aux autres de manière décentralisée et forment une « maille ». Dans la topologie du maillage complet, chaque nœud du réseau est directement relié à chacun des autres. Dans la topologie du maillage partiel, certains nœuds sont connectés à tous les autres, tandis que d'autres ne sont reliés qu'aux nœuds avec lesquels ils échangent le plus de données.

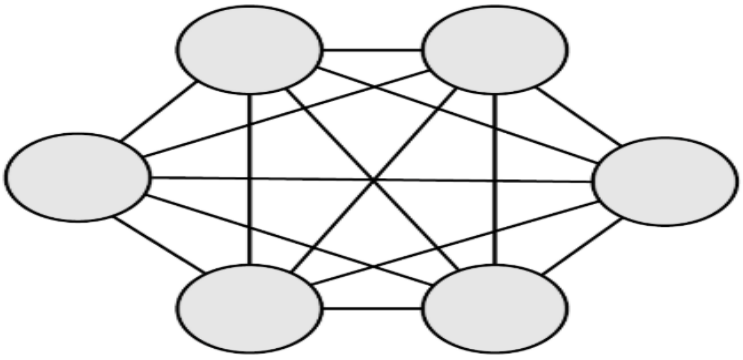

Figure 2.6: Topologie maillé.

2.3 LE WIFI

2.3.3 Définition du WIFI

Est un ensemble de protocoles de communication sans fil régis par les normes du groupe IEEE 802.11 (ISO/CEI 8802-11). Un réseau Wi-Fi permet de relier par ondes radio plusieurs appareils informatiques (ordinateur, routeur, smartphone, modem Internet, etc.) au sein d'un réseau informatique afin de permettre la transmission de données entre eux.

Les réseaux WIFI présentent une multitude de fonctionnalités qui viennent aussi bien du monde fixe que du monde mobile. Ces fonctionnalités les permettent d'être plus fiables et de faire bénéficier au maximum l'utilisateur de service. Les principales fonctionnalités d'un réseau WIFI sont :

- La fragmentation et le réassemblage qui permettent d'éviter le problème de transmission d'importants volumes de données donc de diminuer le taux d'erreur.
- La gestion de la mobilité.
- La variation du débit en fonction de l'environnement radio.

2.3.4 La norme IEE 802.11

Fait partie de l'ensemble de protocoles de réseau local (LAN) IEEE 802 et spécifie l'ensemble des protocoles de contrôle d'accès au support (MAC : Media Access Control) et de la couche physique (PHY) pour la mise en œuvre de la communication informatique Wi-Fi du réseau local sans fil (WLAN) dans diverses fréquences, y compris, mais sans s'y limiter, les bandes de fréquences de 2,4 GHz, 5 GHz, 6 GHz et 60 GHz. La norme IEEE 802.11 se décompose en éléments identifiés comme suit :

- 802 : standard général de base pour le déploiement de réseaux numériques locaux ou métropolitains à liaison filaire ou sans fil.
- 802.1 : gestion des réseaux.
- 802.10 : sécurisation des échanges pour les systèmes à liaison filaire ou sans fil (Token Ring, Ethernet, Wi-Fi, WiMAX).
- 802.11 : spécifications pour l'implémentation de réseaux numériques locaux à liaison sans fil.
- 802.2 : description générale de la sous-couche Logical Link Control.

2.4 Réseaux Maillés Sans Fil (RMSF)

2.4.3 Définition du réseau maillé sans fil

Un réseau maillé sans fil est un ensemble des équipements informatiques sans fil (possèdent un ou plusieurs interfaces radio) interconnectés de proche en proche sans hiérarchie centrale, formant une structure de filet. Les nœuds de réseau maillé sans fil sont capables de se configurer et s'organiser dynamiquement, la communication entre les nœuds est basée sur le principe de multi-sauts, d'où plusieurs nœuds intermédiaire participent intelligemment dans la retransmission des informations jusqu'à la destination. C'est un réseau qui s'étend en fonction du nombre de participants.

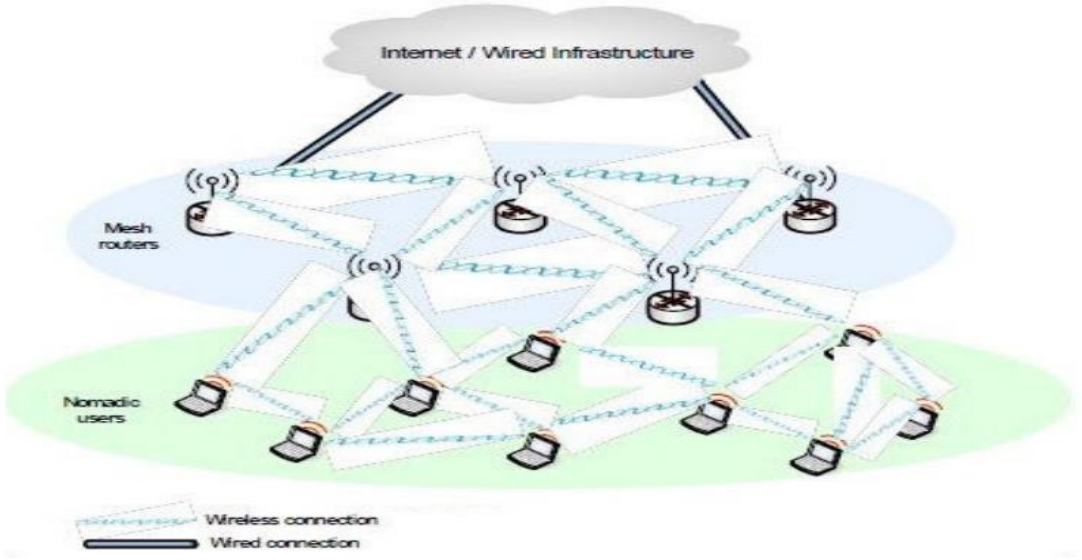

Figure 2.7 : Architecture des réseaux maillés sans fil.

2.4.4 Principe du mesh

Le principe du mesh est de remplacer tout câblage dans un réseau par un lien sans fil. Contrairement, à un réseau wifi composé de plusieurs points d'accès connectés en filaire, le mesh a pour ambition de supprimer le câblage entre les points d'accès. L'idée est que chaque nœud va servir d'un point d'accès pour ses voisins. C'est-à-dire que tout le monde peut communiquer avec tout le monde par l'intermédiaire de son voisin. Le réseau maillé est donc constitué de « mailles » permettant la communication d'un nœud vers un autre. Le mode mesh peut-être implémenté de plusieurs manières:

- **Réseau maillé dynamique :** Tous les nœuds sont en « ad hoc », ce qui veut dire que tout nœud peut faire office de routeur. Techniquement, c'est un réseau maillé dynamique ou les mailles peuvent se former ou disparaître selon les positions de chaque nœud.
- **Réseau maillé statique (mode WDS) :** Les points d'accès communiquent entre eux par l'intermédiaire du wifi, et sont les seuls à jouer un rôle de routeur.

2.4.5 Caractéristiques des réseaux maillés sans fil

Les caractéristiques des réseaux maillés sans-fil sont détailles ci-dessous :

- **Un réseau en multi sauts :** Deux nœuds qui ne sont pas liés directement peuvent communiquer par l'intermédiaire des nœuds qui assurent le routage saut par saut. Cette caractéristique permet à un nœud d'atteindre la totalité des nœuds du réseau.
- **La mobilité :** Les points d'accès possèdent une mobilité très réduite (dans la plupart des cas, ils sont fixes). Les nœuds clients peuvent être stationnaires ou mobiles (PC portables, PDA, Téléphones mobiles, etc.).
- **Compatibilité et Interopérabilité avec les réseaux sans fil existants :** Les réseaux maillés basés sur les technologies IEEE 802.11 doivent être compatibles avec les normes existantes IEEE 802.11 dont l'objectif est de supporter aussi bien les clients conventionnels Wifi que la technique de maillage. Tels réseaux doivent inter-opérer avec d'autres technologies sans-fil comme le WiMAX, le ZigBee et les réseaux cellulaires.
- **Propriété et responsabilité partagées :** L'appartenance du réseau maillé est partagée au sein des adhérents du réseau, donc la gestion du réseau est assurée par la totalité des nœuds [13].

Cette topologie décentralisée présente un gros avantage : si un nœud ne fonctionne plus, les autres ne continuent de communiquer entre eux, directement ou via un ou plusieurs nœuds intermédiaires. Cela assure que le réseau poursuis de fonctionner en continu [14].

2.4.6 Les Types de nœuds

- **Nœud racine :** le nœud racine est le nœud supérieur du réseau et sert de seule interface entre le réseau ESP-MESH et un réseau IP externe. Le nœud racine est connecté à un routeur Wi-Fi conventionnel et relaie les paquets vers / depuis le réseau IP externe vers les nœuds du réseau ESP-MESH. Il ne peut y avoir qu'un seul nœud racine dans un réseau ESP-MESH et la connexion en amont du nœud racine ne peut être qu'avec le routeur.
- **Nœuds feuilles** : un nœud feuille est un nœud qui ne peut pas avoir de nœuds enfants. Par conséquent, un nœud feuille peut uniquement transmettre ou recevoir ses propres paquets,

mais ne peut pas transmettre les paquets d'autres nœuds. Si un nœud est situé sur la couche maximale autorisée du réseau, il sera affecté en tant que nœud feuille.

- **Nœuds parents intermédiaires :** les nœuds connectés qui ne sont ni le nœud racine ni un nœud feuille sont des nœuds parents intermédiaires. Un nœud parent intermédiaire doit avoir une seule connexion avec un seul nœud parent, mais peut avoir de zéro à plusieurs connexions avec des nœuds enfants. Par conséquent, un nœud parent intermédiaire peut transmettre et recevoir des paquets, mais aussi retransmettre des paquets envoyés à partir de ses connexions.
- **Nœuds inactifs** : les nœuds qui n'ont pas encore rejoint le réseau sont affectés en tant que nœuds inactifs. Les nœuds inactifs tenteront de former une connexion avec un nœud parent intermédiaire ou tenteront de devenir le nœud racine dans les circonstances appropriées [15].

2.4.7 Réseau de capteur sans fil (RCSF)

Les réseaux de capteurs sans fil (RCSF) constituent une catégorie particulière de réseaux ad hoc. Ils sont souvent composés d'un nombre très important de nœuds. Ces nœuds sont des entités capables d'opérer en toute autonomie afin de collecter, traiter et envoyer les données relative à leur environnement. Déployés de façon à couvrir un territoire donné, les capteurs communiquent par radio afin de concentrer l'information sur une station collectrice située au cœur ou en bordure du territoire, appelée nœud racine

Un nœud est une architecture matérielle et logicielle qui peut effectuer plusieurs fonctionnalités [16]. Elle se compose généralement des éléments suivants :

- Un capteur qui réalise l'acquisition des données.
- Une unité de traitement pour l'analyse et le traitement des mesures acquises par les capteurs.
- Une unité de communication pour la réception et la transmission des données vers d'autres nœuds.
- Une mémoire pour le stockage temporel ou permanent.
- Un actionneur pour agir sur l'environnement.
- Une source d'énergie pour alimenter toutes ces unités.

L'architecture logicielle implémente des fonctions comme l'acquisition de données, la communication avec un autre nœud, la sécurité pour protéger les informations échangées sur le réseau, etc.

Dans notre travail qui consiste à la conception et la réalisation d'un réseau de capteurs sans fil pour la maintenance prédictive des machines, nous avons utilisé un réseau mesh constitue des cartes de développement à base de microcontrôleurs esp32 doté d'une connexion wifi, qui permet aux nœuds de «se parler» directement les uns avec les autres sans connexion Internet.

2.4.7.1 ESP32

ESP32 est une série de microcontrôleurs à faible coût et à faible consommation d'énergie sur puce avec Wi-Fi intégré et Bluetooth bimode. La série ESP32 utilise un microprocesseur Tensilica Xtensa LX6 dans des variantes à double cœur et à cœur unique et comprend des commutateurs d'antenne intégrés, un balun RF, un amplificateur de puissance, un amplificateur de réception à faible bruit, des filtres et des modules de gestion de l'alimentation. ESP32 est créé et développé par Espressif Systems, et est fabriqué par TSMC (Taiwan Semiconductor Manufacturing Company) en utilisant leur processus de 40 nm.

Figure 2.8 : Modèle ESP32.

- **Les caractéristiques techniques :**
  - **Processeurs :**
    - CPU : microprocesseur Xtensa dual-core (ou single-core) 32 bits LX6, fonctionnant à 80 ou 240 MHz et fonctionnant jusqu'à 600 DMIPS.
    - Co-processeur ultra basse consommation (ULP).
  - **Mémoire:** 520 KB SRAM
  - **Connectivité sans fil:**
    - Wi-Fi: 802.11 b / g / n
    - Bluetooth: v4.2 BR / EDR et BLE (partage la radio avec Wi-Fi).

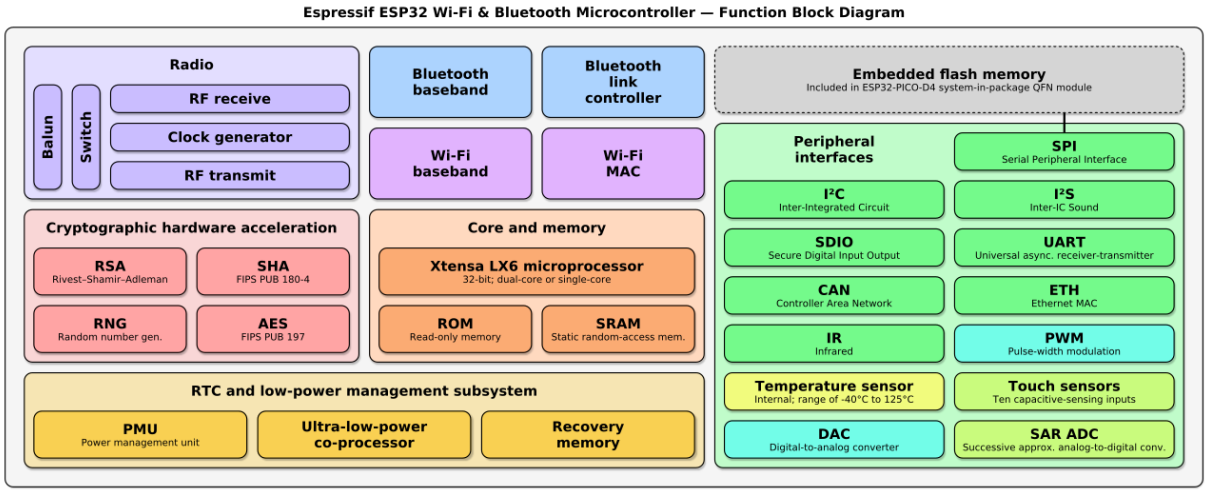

Figure 2.9 : Schéma fonctionnel ESP32.

➢ **Brochage de la carte de développement ESP32 :**

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

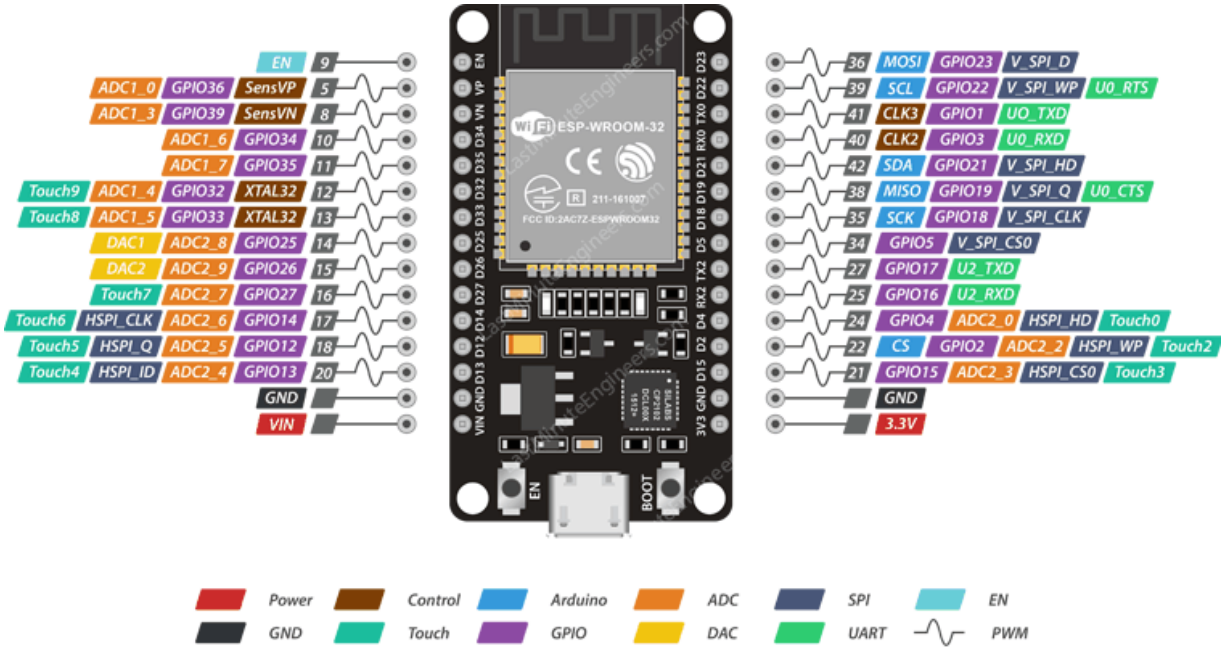

Figure 2.10 : Pinout d'ESP32.

- **Broches d'alimentation** : deux broches d'alimentation sont disponible à savoir : Broche VIN et broche 3,3 V. La broche VIN peut être utilisée pour alimenter directement l'ESP32 et ses périphériques, si vous avez une source de tension 5V régulée. La broche 3,3 V est la sortie d'un régulateur de tension intégré. Cette broche peut être utilisée pour alimenter des composants externes.

- **GND** : est une broche de masse de la carte de développement ESP32.
- **Broches Arduino :** ne sont que des broches matérielles I2C et SPI d'ESP32 pour brancher toutes sortes de capteurs et de périphériques dans votre projet.
- **Broches GPIO :** La carte de développement ESP32 possède 25 broches GPIO qui peuvent être affectées à diverses fonctions par programmation. Chaque GPIO activé numérique peut être configuré en pull-up ou pull-down interne, ou réglé en haute impédance. Lorsqu'il est configuré comme entrée, il peut également être défini sur front-trigger ou level-trigger pour générer des interruptions CPU.
- **Tablettes tactiles :** La carte propose 9 GPIO à détection capacitive qui détectent les variations capacitives introduites par le contact direct du GPIO ou la proximité avec un doigt ou d'autres objets.
- **Pins UART :** La carte de développement ESP32 possède 2 interfaces UART, à savoir UART0 et UART2, qui fournissent une communication asynchrone (RS232 et RS485) et un support IrDA, et communiquent jusqu'à 5 Mbps. UART fournit également la gestion matérielle des signaux CTS et RTS et le contrôle de flux logiciel (XON et XOFF).
- **Capteur à effet Hall :** L'ESP32 intègre un capteur Hall basé sur une résistance à porteuse N. Lorsque la puce est dans le champ magnétique, le capteur Hall développe une petite tension latéralement sur la résistance, qui peut être directement mesurée par l'ADC.
- **Capteur tactile :** L'ESP32 dispose de 10 GPIO à détection capacitive, qui détectent les variations induites par le contact ou l'approche des GPIO avec un doigt ou d'autres objets. La nature à faible bruit de la conception et la haute sensibilité du circuit permet d'utiliser des plots relativement petits. Des tableaux de tampons peuvent également être utilisés, de sorte qu'une plus grande zone ou plusieurs points peuvent être détectés [17].

➢ **Outils de développements :**

Il existe une variété de plates-formes de développement qui peuvent être équipées pour programmer l'ESP32 :

- Arduino IDE avec le module ESP32 Arduino Core
- Espruino
- FAUST, langage de programmation de traitement de données audio, utilisant son DSP
- Lua RTOS pour ESP32
- MicroPython, une variante pour l'embarqué du langage Python
- mruby une variante pour l'embarqué du langage Ruby

- NodeMCU ;

Dans notre travail nous avons optés pour l'environnement de développement intégré Arduino (**Arduino IDE**) qui est une application multiplateforme (pour Windows, macOS, Linux) écrite dans des fonctions de C et C ++. Il est utilisé pour écrire et télécharger des programmes sur des cartes compatibles Arduino et autres. Des modules supplémentaires sont ajoutés à l'IDE Arduino, afin de programmer l'ESP32. Pour vérifier l'installation correcte de l'esp32 dans Arduino IDE nous avons testé l'exemple simple de LED interne déjà disponible dans Arduino IDE pour la carte de développement ESP32 pour l'IoT. Ces exemples sont installés lors de l'installation de la bibliothèque ESP32 dans Arduino IDE.

Etapes à suivre

- Connexion de la carte à l'ordinateur
- Sélection de notre carte (fenêtre outil >> Cartes : clique sur le module Dev -type de carte)
- Sélection du port auquel notre esp32 est connecté (fenêtre outil >> port)
- Télécharger l'exemple et vérifier sur l'esp32 le clignotement de LED

```
#define LED 2
void setup() {
pinMode(LED,OUTPUT);
}
void loop() {
digitalWrite(LED,HIGH);
delay(1000);
digitalWrite(LED,LOW);
delay(1000);
}
```

Comme nous le remarquons la structure d'un programme Arduino doit impérativement être divisé en deux parties : une **fonction setup** et une **fonction loop**. Ces deux fonctions sont de type void, c'est-à-dire qu'elles ne peuvent pas prendre des valeurs.

La fonction setup est la fonction qui se lance au début du programme. Elle permet d'initialiser les variables et de définir les broches de la carte ESP32 qui seront utilisées.

La fonction loop se lance après la fonction setup et, comme son nom l'indique, elle fait une boucle jusqu'à ce que la carte soit débranchée.

Dans le cas de l'exemple de LED nous procédons comme suit ;

1. Définir l'emplacement de la lampe intérieure dans la broche 2 (Pin 2)
2. Appel de la fonction pinMode qui permet de configure les broches comme OUTPUT ou INPUT, et la fonction digitalWrite qui de régler la tension sur la broche a 5V ou 3,3V (HIGH), ou 0V (LOW).
3. Appel à la boucle pour faire clignoter la LED avec un délai de 1000.

Afin de programmer notre réseau maillé à partir de l'Arduino IDE nous installons la bibliothèque painlessMesh qui s'occupe des détails de la création d'un réseau maillé simple à l'aide de l'ESP32 ;

2.4.8 PainlessMesh

Le painlessMesh est une bibliothèque de réseau maillé WiFi spécialement écrite pour les plates-formes ESP8266 et ESP32. Il est appelé painlessMesh car il est conçu pour être auto-configuré et facile à installer. C'est un réseau ad-hoc qui ne nécessite ni plan de routage ni contrôleur central, et tous les nœuds sont égaux. Elle permet au périphérique ESP32 de former un réseau maillé, deux ou plusieurs modules, également appelés nœuds, avec le même SSID (identifiant d'ensemble de services) seront automatiquement connectés pour former un réseau maillé.

La topologie n'est en fait pas un réseau maillé complet : en raison de ressources limitées, les chemins cycliques sont activement évités afin de simplifier considérablement les tâches de routage. Le réseau formé ressemble plus à un réseau en étoile. Cependant, il diffère d'un réseau en étoile WiFi habituel qui consiste généralement en un point d'accès central auquel une ou plusieurs stations sont connectées.

Dans le réseau painlessMesh :

- Chaque nœud peut servir à la fois de point d'accès pour les autres nœuds auxquels se connecter et également de station se connectant à un autre nœud.
- Chaque nœud est conscient de la topologie du réseau dans son ensemble qui est mise à jour périodiquement.
- Chaque nœud informe ses voisins des autres nœuds auxquels il est directement ou indirectement connecté. Station d'un nœud auquel n'est pas connecté tout point d'accès recherchera activement un point d'accès qui a le signal le plus fort mais qui n'est pas encore répertorié dans sa topologie de réseau. Ce mécanisme empêche la

formation de chemins cycliques. Il n'y aura donc qu'un seul chemin entre une paire de nœuds. La figure 2.11 montre le schéma d'un réseau painlessMesh. [18].

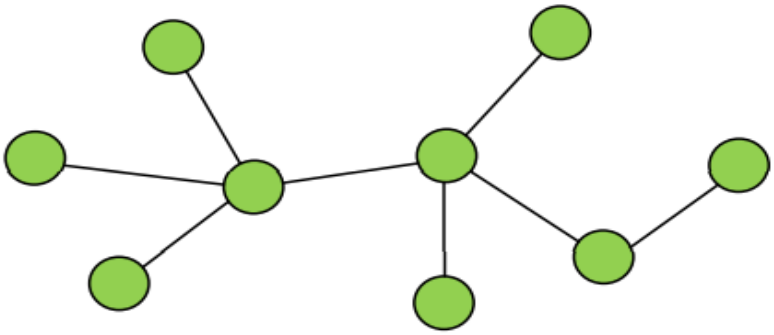

Figure 2.11 : Schéma du réseau painlessMesh.

Pour la création de notre réseau nous avons utilisé l'exemple basic de la bibliothèque painlessMesh:

```
#include "painlessMesh.h"
#define   MESH_PREFIX     "whateverYouLike"
#define   MESH_PASSWORD   "somethingSneaky"
#define   MESH_PORT       5555
Scheduler userScheduler;
painlessMesh  mesh;
void sendMessage() ;
Task taskSendMessage( TASK_SECOND * 1 , TASK_FOREVER, &sendMessage );
void sendMessage() {
String msg = "Hello from node ";
msg += mesh.getNodeId();
mesh.sendBroadcast( msg );
taskSendMessage.setInterval( random( TASK_SECOND * 1, TASK_SECOND * 5 ));
}
void receivedCallback( uint32_t from, String &msg ) {
Serial.printf("startHere: Received from %u msg=%s\n", from, msg.c_str());
}
void newConnectionCallback(uint32_t nodeId) {
Serial.printf("--> startHere: New Connection, nodeId = %u\n", nodeId);
}
void changedConnectionCallback() {
Serial.printf("Changed connections\n");
}
void nodeTimeAdjustedCallback(int32_t offset) {
Serial.printf("Adjusted time %u. Offset = %d\n", mesh.getNodeTime(),offset);
}
void setup() {
Serial.begin(115200);
```

```
mesh.setDebugMsgTypes( ERROR | STARTUP );
mesh.init( MESH_PREFIX, MESH_PASSWORD, &userScheduler, MESH_PORT );
mesh.onReceive(&receivedCallback);
mesh.onNewConnection(&newConnectionCallback);
mesh.onChangedConnections(&changedConnectionCallback);
mesh.onNodeTimeAdjusted(&nodeTimeAdjustedCallback);
userScheduler.addTask( taskSendMessage );
taskSendMessage.enable();
}
void loop() {
mesh.update();
}
```

Nous avons procédé comme suit :

Inclure le fichier d'entête painlessMesh

1. Définir les informations d'identification Wi-Fi pour le réseau maillé. Il comprend :
   - **SSID** : Le nom de votre maillage. Tous les nœuds partagent le même SSID AP
   - **Password** : mot de passe de notre maillage
   - **Port** : Le port TCP sur lequel nous souhaitons que le serveur s'exécute. La valeur par défaut est 5555 si non spécifié

   Ces trois paramètres doivent rester les mêmes pour chaque nœud du réseau.
2. Créez des instances pour ***PainlessMesh*** et Scheduler
3. définir deux fonctions. Une pour envoyer des messages et une autre pour les recevoir.
   - **bool painlessMesh :: sendBroadcast (String & msg, bool includeSelf = false):** envoie des **msg** à chaque nœud sur l'ensemble du réseau maillé. Par défaut, le nœud actuel est exclu de la réception du message **(includeSelf = false)**.
   - **includeSelf = true** remplace ce comportement, provoquant le **receivedCallback** appel de lors de l'envoi d'un message de diffusion. Renvoie **true** si tout fonctionne, false sinon. Imprime un message d'erreur dans **Serial.print,** en cas d'échec.
   - **uint32_t painlessMesh :: getNodeId (void)** Renvoie le **chipId** du nœud sur lequel nous exécutons.
   - **Void painlessMesh :: onReceive (& receivedCallback) :** Définit une routine de rappel pour tous les messages qui sont adressés à ce nœud. La routine de rappel à la structure suivante : ***void receivedCallback(uint32_t from, String &msg).*** Chaque fois que ce nœud reçoit un message, cette routine de rappel sera appelée. **"From"** est l'**ID** de l'expéditeur d'origine du message. **"msg"** est une chaîne qui contient le message. Le

message peut être n'importe quoi. Un JSON, une autre chaîne de texte ou des données binaires

4. Appel de toutes les fonctions de rappel une par une. Chaque fois qu'un nœud reçoit un message, la routine de rappel ci-dessous sera appelée :
   - **mesh.onReceive (& receiveCallback) -** Lorsqu'un nouveau nœud établit une connexion, ce rappel est appelé
   - **Void painlessMesh :: onNewConnection (& newConnectionCallback) :** se déclenche chaque fois que le nœud local établit une nouvelle connexion. Le rappel à la structure suivante ***: Void newConnectionCallback(uint32_t nodeId)***

     nodeId est le nouvel **ID** de nœud connecté dans le maillage.
   - **void painlessMesh :: onChangedConnections (& hangedConnectionsCallback) :** Cela se déclenche chaque fois qu'il y a un changement dans la topologie du maillage. Le rappel à la structure suivante : ***void onChangedConnections ()***
     Aucun paramètre n'est passé. Ceci n'est qu'un signal.
   - **mesh.onNodeTimeAdjusted (& nodeTimeAdjustedCallback)** Cela se déclenche chaque fois que l'heure locale est ajustée pour la synchroniser avec l'heure du maillage. Le rappel à la structure suivante : ***void onNodeTimeAdjusted(int32_t offset) ;*** le **offset** est le delta d'ajustement qui a été calculé et appliqué à l'horloge locale.
   - **void painlessMesh::update(void)** Ajoutez ceci à la fonction loop () Cette routine exécute diverses tâches de maintenance [19].

Afin de tester la création de notre réseau, nous avons testé l'exemple basic pour le capteur de Hall et de touche intégrés dans la carte ESP32. Le programme reste le même et les changement concerne que la variable **msg** dans la fonction **sendMessage()** :

Pour le capteur de hall :

```
void sendMessage() {

int val = 0;

val = hallRead();

bool includeSelf = true;

String msg = "Hello from node ";

msg += mesh.getNodeId();
msg +=    "hall value" + String (val);
```

```
mesh.sendBroadcast( msg,includeSelf );
taskSendMessage.setInterval( random( TASK_SECOND * 1, TASK_SECOND * 5 ));
}
```

Les résultats obtenus sont illustrés sur PuTTY :

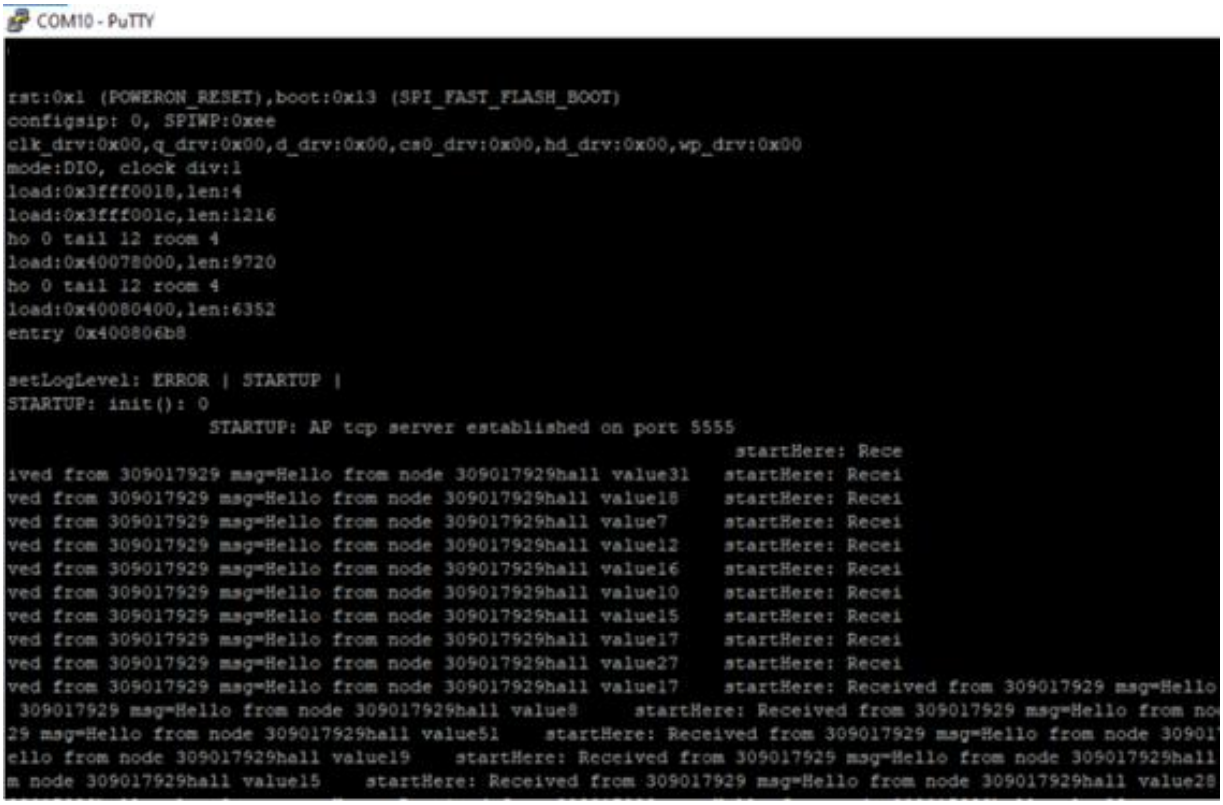

Figure 2.12 : Résultats du test de capteur de hall.

Pour le capteur Touch :

```
void sendMessage() {

bool includeSelf = true;

String msg = "Hello from node ";

msg +=  "touch value" + String(touchRead(T0)) ;
msg += mesh.getNodeId();
mesh.sendBroadcast( msg,includeSelf );
```

```
taskSendMessage.setInterval( random( TASK_SECOND * 1, TASK_SECOND * 5 ));
}
```

Les résultats obtenus sont illustrés sur PuTTY :

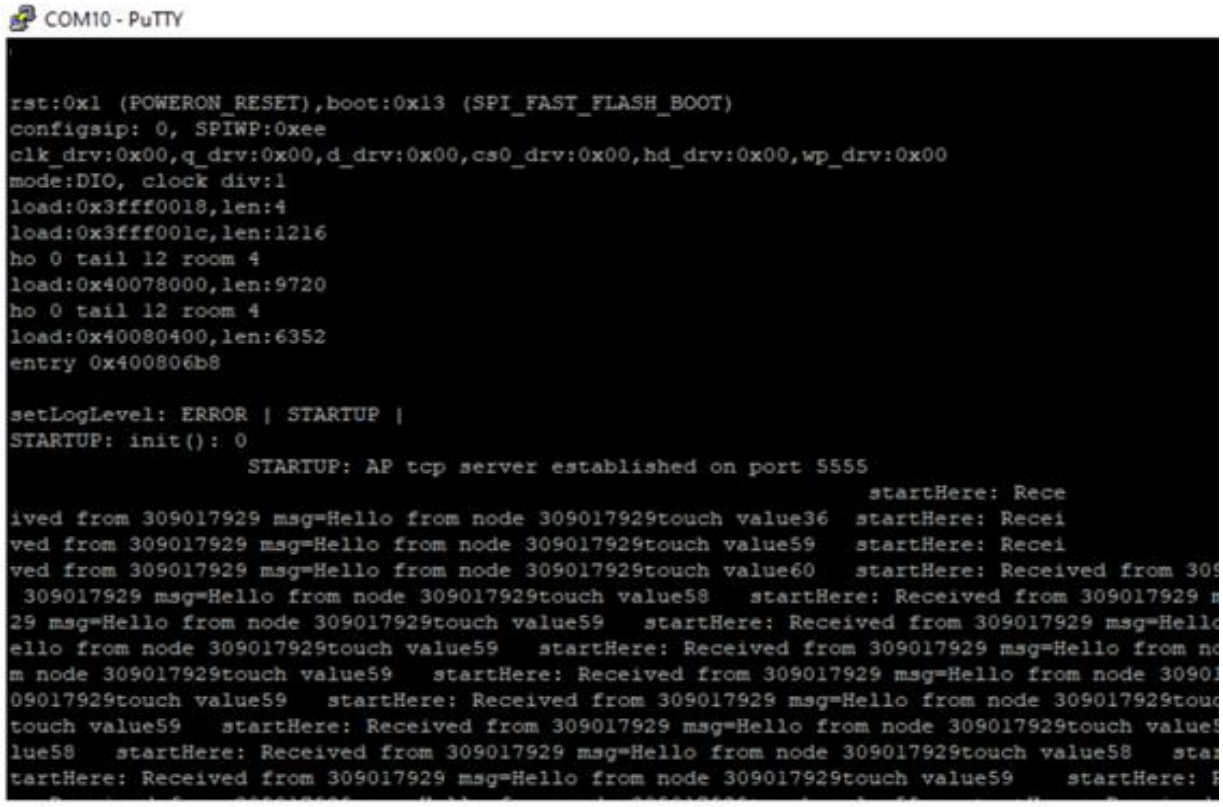

Figure 2.13 : Résultats du test de capteur de touche.

Les messages utilisés dans painlessMesh sont basés sur des objets JSON (JavaScript Object Notation). Le principal avantage de JSON est qu'il rend le code et les messages lisibles par l'homme et faciles à comprendre. De plus, il facilite l'intégration de painlessMesh avec des frontaux JavaScript, des applications Web et d'autres applications.

2.5 La notation JSON

2.5.3 Définition de JSON

JSON (JavaScript Object Notation) est un langage léger d'échange de données textuelles. Pour les ordinateurs, ce format se génère et s'analyse facilement. Pour les humains, il est pratique à écrire et à lire grâce à une syntaxe simple et à une structure en arborescence, ce format peut toutefois être généré et lu par la plupart des langages de programmation, Cette universalité lui a

permis de devenir une façon très populaire de stocker, organiser, lire et partager des données dans les applications et services web.

JSON permet de représenter des données structurées (comme XML par exemple), Les données sont présentées dans un format textuel constitué de paires ”key / value“.

2.5.4 Fonctionnement de JSON

Fondé sur un sous-ensemble de JavaScript, JSON est un format texte totalement indépendant de tout langage. Pourtant, les conventions utilisées ne surprendront pas les codeurs familiers aux langages descendant du C tels que JavaScript, Python, Pearl ou d'autres.

Autrement dit, il fonctionne un peu comme le XML (mais en moins développé) et facilite la structuration des informations présentes dans un document informatique. Comme il sert simplement à fluidifier l'échange de données, il n'est pas supposé contenir de commentaires, par exemple, ce qui le distingue d'un langage informatique à part entière. Toutefois, certaines bibliothèques en acceptent, s'ils sont écrits en JavaScript.

Le JSON fait partie des langages compréhensibles aussi bien par un esprit humain que par une machine. D'ailleurs, son apprentissage est facile et intuitif [20].

2.5.5 Stockage des objets JSON

Concrètement, les objets JSON ne sont que du texte. Il est donc possible de les stocker de multiples manières. On peut les stocker dans une base de données, dans un fichier texte séparé, dans un stockage client comme les cookies ou le local Storage, ou même en utilisant le format JSON.

Une fois le contenu stocké, il peut être récupéré et déchiffré de différentes façons et dans différents langages avec le langage JavaScript [21].

2.5.6 Les types de données

- **Nombre :** un nombre décimal signé qui peut contenir une partie fractionnaire.
- **Booléen :** l'une des valeurs true ou false.
- **Tableau :** une liste ordonnée de zéro ou plusieurs valeurs, chacune pouvant être de n'importe quel type. Les tableaux utilisent une notation entre crochets avec des éléments séparés par des virgules.
- **Chaîne de caractères :** est un type de donnée dans de nombreux langages informatiques. La traduction en anglais est string.

- **un objet :** est un conteneur symbolique et autonome qui contient des informations et des mécanismes concernant un sujet, manipulés dans un programme.
- **nul :** une valeur vide, en utilisant le mot nul.

2.5.7 Les structure de données JSON

JSON est construit par rapport à deux structures :

- Une collection de paires nom / valeur. Dans les différents langages, ce type de structure peut s'appeler objet, enregistrement, dictionnaire, table de hachage, liste à clé ou tableau associatif.
- Une liste ordonnée de valeurs. Dans la plupart des langages, c'est ce qu'on va appeler tableau, liste, vecteur ou séquence.

Ces deux structures sont des structures de données universelles. Pratiquement tous les langages de programmation modernes les prennent en charge sous une forme ou une autre. Il est logique qu'un format de données interchangeable avec les langages de programmation soit également basé sur ces structures [22].

2.5.8 Notation de base sur la syntaxe JSON

En JSON, les données sont structurées d'une manière spécifique. JSON utilise des symboles comme **{} , : ” ” [ ]** et il a la syntaxe suivante:

- Les données sont représentées par des paires clé / valeur.
- Le deux-points **(:)** attribue une valeur à la clé.
- les paires clé / valeur sont séparées par des virgules **(,).**
- Les accolades contiennent des objets **({}).**
- Les crochets contiennent des tableaux **([]).**

Par exemple, pour représenter des données en JSON, les paires clé / valeur se présentent comme suit : {**" key1 ": “value1 ",” key2 ": “value2 ", " key3 ": “value3 "}**

En JSON, les valeurs peuvent être un autre objet JSON (sports) ou un tableau (pets). Par exemple [23] :

```
{
 " Nom ": " Rui ",
 " sports ": {
 " outdoor ": " randonnée ",
 " indoor ": " natation "
},
```

```
" Animaux ": [
 " Max ",
" Dique "
 ]
}
```

2.5.9 Décodage JSON – Analyse de la chaîne JSON

2.5.9.1 Desérialisaton du document JSON

La fonction **deserializeJson()** analyse une entrée JSON et place le résultat dans un fichier **JsonDocument**. Dans cette partie nous expliquerons comment analyser les messages JSON avec l'ESP32 et la bibliothèque ArduinoJson . Cette dernière est installée via le gestionnaire de bibliothèque Arduino IDE. L'une des caractéristiques uniques d'ArduinoJson est sa stratégie d'allocation de mémoire. Cela fonctionne comme suit :

1. Tout d'abord, nous créons un JsonDocument pour réserver une quantité spécifiée de mémoire.
2. Ensuite, nous désérialisons le document JSON.
3. Enfin, nous détruisons le JsonDocument, qui libère la mémoire réservée.

La mémoire du JsonDocument peut être soit dans la pile, soit dans le tas. L'emplacement dépend de la classe dérivée que nous choisissons. Si nous utilisons un **StaticJsonDocument**, il sera dans la pile ; si nous utilisons un **DynamicJsonDocument**, il sera dans le tas. Un **JsonDocument** est responsable de la réservation et de la libération de la mémoire utilisée par ArduinoJson

Lorsque nous créons un JsonDocument, nous devons spécifier sa capacité en octets. Dans le cas de **DynamicJsonDocument**, nous définissons la capacité via un argument constructeur : **DynamicJsonDocument doc (capacity);**

Comme il s'agit d'un paramètre du constructeur, nous pouvons utiliser une variable régulière, dont la valeur peut changer au moment de l'exécution.

Dans le cas d'un **StaticJsonDocument,** nous définissons la capacité via un paramètre de modèle : **StaticJsonDocument <capacity>doc ;**

Comme il s'agit d'un paramètre de modèle, nous ne pouvons pas utiliser de variable. Au lieu de cela, nous devons utiliser une expression constante, ce qui signifie que la valeur doit être calculée au moment de la compilation.

Comme le compilateur gère la pile, il a donc besoin de savoir la taille de chaque variable lors de la compilation du programme. Pour déterminer la capacité de nous JsonDocument nous devons comprendre qu'ArduinoJson stocke dans le JsonDocument. Ce dernier doit stocker une structure de données qui reflète la hiérarchie des objets dans le document JSON. En d'autres termes, le JsonDocument contient des objets qui sont liés les uns aux autres de la même manière qu'ils le font dans le document JSON. Par conséquent, la capacité du JsonDocument dépend fortement de la complexité du document JSON. S'il ne s'agit que d'un seul objet avec peu de membres, quelques dizaines d'octets suffisent. S'il s'agit d'un document JSON volumineux, des centaines de kilo-octets sont nécessaires.

ArduinoJson fournit des macros pour calculer précisément la capacité du JsonDocument. La macro pour calculer la taille d'un objet est **JSON_OBJECT_SIZE ()**. Il prend un argument : le nombre de membres dans l'objet.

Donc, Arduino JSON utilise un pool de mémoire préalloué pour stocker l'arborescence JsonObject, Pour cela, nous allons créer une instance de JsonBuffer qui stockera la représentation en mémoire de notre entrée. Il existe deux implémentations de JsonBuffer, StaticJsonBuffer qui utilise la pile et DynamicJsonBuffer qui utilise le tas.

Ceci est effectué par StaticJsonBuffer. Nous pouvons utiliser ArduinoJson Assistant pour calculer la taille exacte du tampon comme suit [24] :

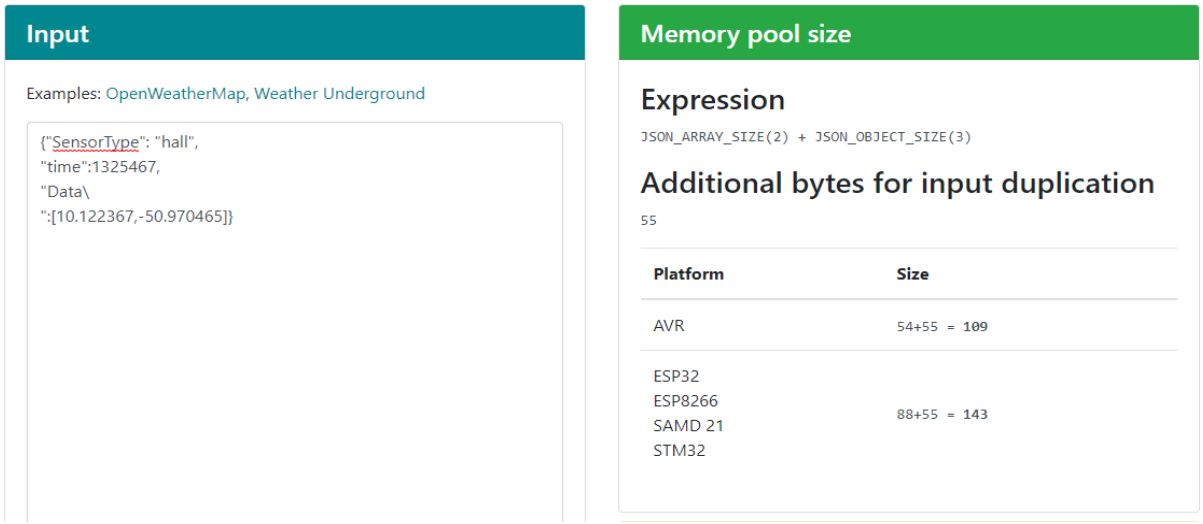

Figure 2.14 : Assistant d'ArduinoJson.

Après la définition de type de JsonDocument et la location de la mémoire nécessaire, le JsonDocument est prêt, nous pouvons analyser l'entrée avec **deserializeJson ():**

DeserializationError err = deserializeJson(doc, input);

deserializeJson () renvoie une **DeserializationError** qui indique si l'opération a réussi. Il peut avoir l'une des valeurs suivantes

- **DeserializationError :: Ok:** la désérialisation a réussi.
- **DeserializationError :: IncompleteInput:** L'entrée était valide mais s'est terminée prématurément, ou il était vide; ce code signifie souvent qu'un dépassement de délai s'est produit.
- **DeserializationError :: InvalidInput:** l'entrée n'était pas un document JSON valide.
- **DeserializationError :: NoMemory:** Le JsonDocument était trop petit
- **DeserializationError :: NotSupported:** Le document d'entrée était valide, mais il contient fonctionnalités non prises en charge par la bibliothèque.
- **DeserializationError :: TooDeep:** L'entrée était valide, mais elle contenait trop niveaux de nidification [25] ;

Dans l'exemple suivant nous avons décodé et analysé la chaîne JSON pour le capteur hall interne de notre ESP32 :

```
" {\"SensorType\": \"hall\", \"time\":1325467, \"Data\":[10.122367,-50.970465]}";
#include <ArduinoJson.h>
void setup() {
Serial.begin (115200);
const char* msg = " {\"SensorType\": \"hall\", \"time\":1325467, \"Data\":[10.122367,
-50.970465]}";
StaticJsonDocument <200> buffr;
DeserializationError err = deserializeJson (buffr,msg);
if (err) {  //Check for errors in parsing
Serial.println ("ERROR :");
Serial.println (err.c_str()); //for display the name of the error
return ;  }
const char* SensorType = buffr["SensorType"];
long time = buffr["time"];
float lat= buffr["Data"][0];
float lon= buffr["Data"][1];
Serial.println (SensorType);
Serial.println (time);
Serial.println (lat, 6);
Serial.println (lon, 6);
}
```

```
void loop() {
}
```

1- Nous incluons la bibliothèque ArduinoJson mentionnée précédemment, afin que nous puissions avoir accès à la fonctionnalité d'analyse JSON. Puisque nous allons faire l'analyse réelle dans la fonction de boucle principale, nous allons simplement ouvrir la connexion série sur la fonction de configuration, afin d'imprimer la sortie de notre programme.

2- Nous déclarerons une variable pour contenir le message JSON à analyser

3- Nous déclarerons un objet de classe **StaticJsonBuffer**, qui correspond à un pool de mémoire pré-alloué pour stocker l'arborescence d'objets JSON, nous avons passé une valeur de 200 octets, qui est suffisant pour que la chaine soit analysée.

4- Nous passons à l'analyse de notre entrée, nous appelons la fonction **deserializeJson ().**

Pour tester notre code nous, nous l'avons simplement compilez et télécharger sur l'ESP32 à l'aide de l'IDE Arduino. Une fois la procédure terminée nous ouvrons le PuTTy pour voir la chaine JSON en cours d'impression comme illustré sur la figure 2.15.

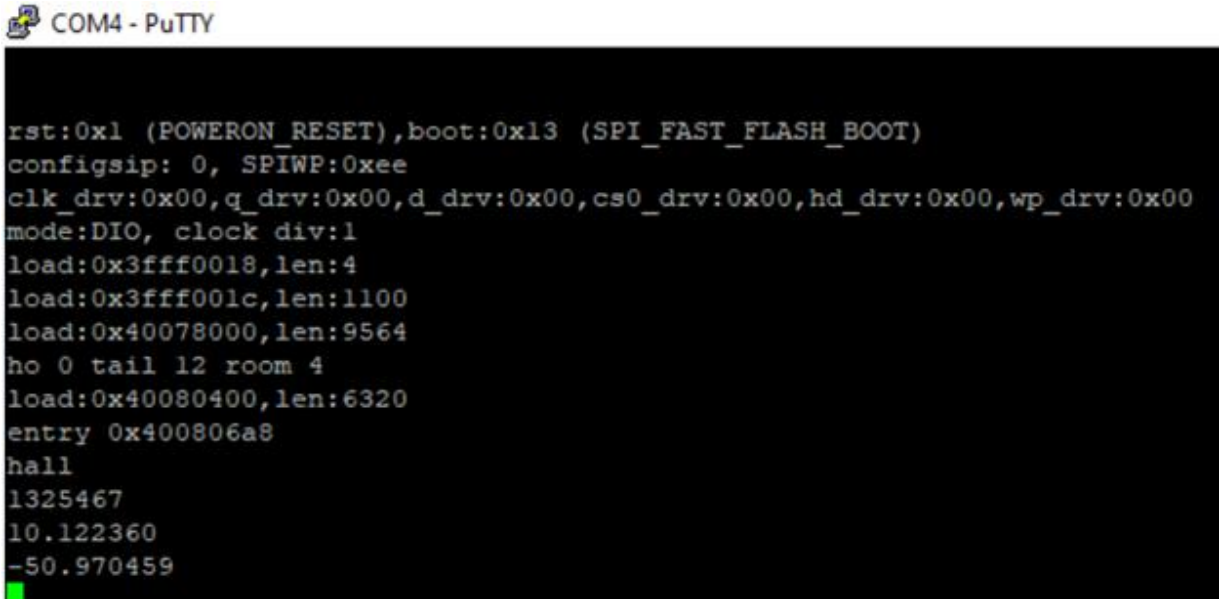

Figure 2.15 : Résultats du programme Deserializing.

2.5.9.2 La sérialisation d'objets

La sérialisation est utilisée pour exporter les données d'application dans un fichier. L'application de destination utilise ensuite la désérialisation pour extraire les données de

l'application pour une utilisation ultérieure. La sérialisation est un concept dans lequel les objets de classe C # sont écrits ou sérialisés dans des fichiers [26].

Dans l'exemple suivant nous avons décodé et analysé la chaîne JSON pour le capteur touche interne de notre ESP32 :

```
#include <ArduinoJson.h>
void setup(){
Serial.begin(115200);
StaticJsonDocument<300> test;
for (int i=0;i<6;i++){
int value=touchRead(T0);
test ["sensorType"] = "tauch";
delay (5000);
test ["value"] = value;
char memoire[300];
serializeJson(test, memoire);
Serial.println(memoire);
}
}
void loop() {
}
```

- Nous avons utilisé une capacité de 300 octets, ce qui est largement suffisant pour l'objet que nous allons représenter.

StaticJsonDocument<300> test;

- Ensuite pour ajouter un nouveau membre à notre document, nous devons simplement utiliser l'opérateur [ ] :

test ["sensorType"] = "tauch";

- pour sérialiser le document en une chaine JSON, nous aurons besoin d'un tampon de caractères pour le stocker :

char memoire[300];

- Pour obtenir la chaine JSON, nous devons simplement appeler la fonction serializeJson. Comme première entrée, nous devons passer notre objet StaticJsonDocument et comme deuxième entrée le tampon de caractères que nous avons précédemment déclaré :

serializeJson(test, memoire);

- Pour finaliser, nous imprimerons le contenu obtenu dans notre tampon de caractères. Il doit afficher une chaine contenant notre document JSON :

Serial.println(memoire);

Pour tester notre code, nous l'avons simplement compilez et télécharger sur l'ESP32 à l'aide de l'IDE Arduino. Une fois la procédure terminée nous ouvrons le PuTTy pour voir la chaine JSON en cours d'impression comme illustré sur la figure 2.16.

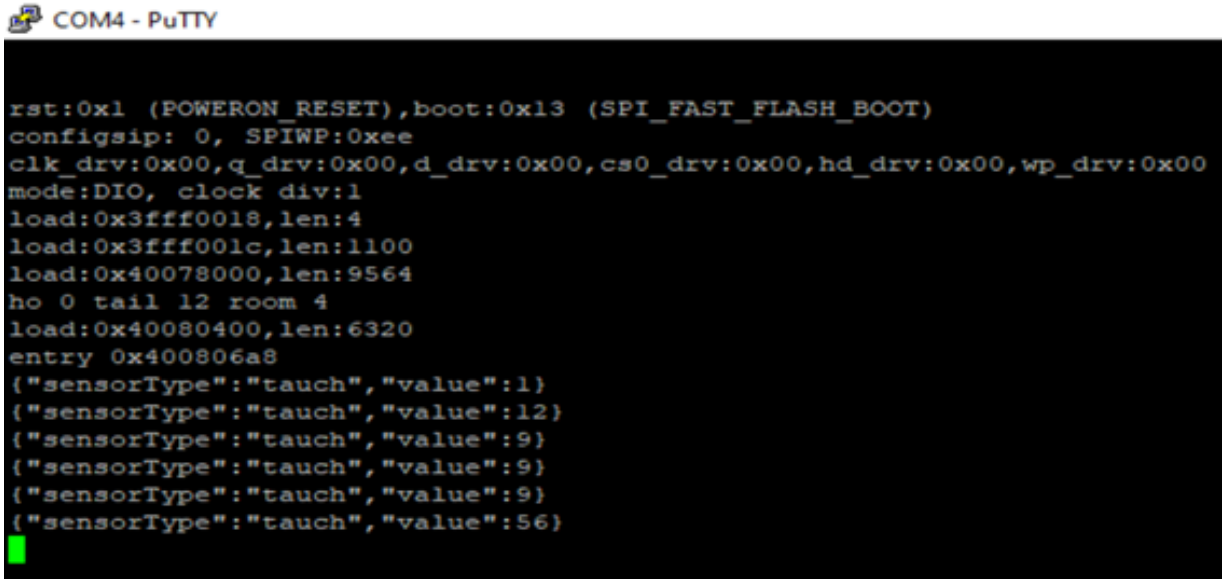

Figure 2.16 : Chaîne JSON en cours d'impression.

2.5.10Les avantages de JSON

- La vitesse de traitement.
- La simplicité de mise en œuvre.
- On n'a pas besoin de passer un fichier XML pour extraire des informations à travers le net, car JSON est reconnu nativement par JavaScript.
- Les contenus binaires peuvent être intégré et échangés sur le net avec une représentation textuelle spéciale avec une commande [27].

2.6 RTOS

RTOS (Real time Operating system) est un programme informatique qui permet de crée un environnement multitâche temps réel avec les services associés le tout avec des contraintes temps réel. En d'autre termes c'est un ensemble de fichiers que l'on va compiler avec notre projet et qui permet de faire comme si plusieurs programme s'exécutent en parallèle alors qu'un processeur monocœur ne peut exécuter qu'une instruction à la fois.

On va ainsi pouvoir programmer sous forme de tâches (task) qui s'exécutent en parallèles et qui peuvent communiquer. En effet, le RTOS est aussi ici pour mettre en place les mécanismes de communication entres les tâches. On va pouvoir envoyer des messages d'une tâche à une autre et faire pleins d'autres choses [28].

2.6.3 FreeRTOS

FreeRTOS est une classe de RTOS qui est conçue pour être suffisamment petite pour fonctionner sur un microcontrôleur - bien que son utilisation ne se limite pas aux applications de microcontrôleur-.

L'ordonnanceur désigne le composant du noyau du système d'exploitation choisissant l'ordre d'exécution des processus sur les processeurs d'un ordinateur. En anglais, l'ordonnanceur est appelé « scheduler ». L'ordonnancement des tâches a pour but principal de décider parmi les tâches qui sont dans l'état « prêt », laquelle exécuter. Pour faire ce choix, l'ordonnanceur de FreeRTOS se base uniquement sur la priorité des tâches [29].

2.6.4 Taches

2.6.4.1 Définition

Les tâches sont généralement les éléments constitutifs des systèmes d'exploitation en temps réel. Ils s'exécutent sur leur propre contexte et l'ordonnanceur est responsable de décider quelle tâche s'exécute à un moment donné dans un seul cœur. Néanmoins, comme indiqué précédemment, nous pouvons exécuter plusieurs tâches en parallèle (une seule s'exécute à chaque fois, mais il peut y en avoir plusieurs instanciées) afin que nos programmes soient plus faciles à coder.

Dans FreeRTOS, les tâches sont implémentées comme des fonctions C et suivent un prototype prédéfini, comme on peut le voir ci-dessous [30] :

```
void taskImplementingFunction( void * parameter )
```

2.6.4.2 Création d'une tâche FreeRTOS:

Nous allons démarrer notre fonction de configuration en ouvrant une connexion série, afin de pouvoir obtenir la sortie de notre programme de test : Serial.begin (115200) ;

Ensuite, nous créerons les tâches, avec un appel à la fonction xTaskCreate. Les arguments de cette fonction sont les suivants :

```
BaseType_t xTaskCreate (
            TaskFunction_t pvTaskCode,
            const char * const pcName,
            uint16_t usStackDepth,
            void * pvParameters,
            UBaseType_t uxPriority,
            TaskHandle_t * pvCreatedTask
          );
```

Les arguments de la fonction xTaskCreate sont les suivants ;

- **pvTaskCode :** Pointeur vers la fonction de saisie des tâches. Les tâches doivent être implémentées pour ne jamais revenir.
- **pcName :** Un nom descriptif pour la tâche. Ceci est principalement utilisé pour faciliter le débogage. Longueur maximale définie par config MAX_TASK_NAME_LEN - la valeur par défaut est 16.
- **usStackDepth :** La taille de la pile de tâches spécifiée comme le nombre de variables que la pile peut contenir - pas le nombre d'octets. Par exemple, si la pile a une largeur de 16 bits et que usStackDepth est défini sur 100, 200 octets seront alloués pour le stockage de la pile.
- **pvParameters :** Pointeur qui sera utilisé comme paramètre pour la tâche en cours de création.
- **uxPriority :** Priorité à laquelle la tâche doit s'exécuter. Les systèmes qui prennent en charge MPU peuvent éventuellement créer des tâches dans un mode privilégié (système) en définissant le bit portPRIVILEGE_BIT du paramètre de priorité. Par exemple, pour créer une tâche privilégiée à la priorité 2, le paramètre uxPriority doit être défini sur (2 | portPRIVILEGE_BIT).
- **pvCreatedTask :** Utilisé pour renvoyer une poignée par laquelle la tâche créée peut être référencée.
- **TaskHandle:** renvoie un descripteur qui peut être utilisé pour la dernière référence de la tâche lors d'appels à des fonctions (par exemple, pour supprimer une tâche ou modifier

sa priorité). De plus, pour cet exemple simple, nous n'allons pas l'utiliser, il sera donc NULL.

Cette fonction retourne **pdPASS** si la tâche a été créée avec succès et ajoutée à une liste prête, sinon un code d'erreur défini dans le fichier projdefs.h [31].

Nous allons créer, deux tâches, task1 et task2 qui nous nommons "Hello" et "Touch" successivement. Leur taille sur la pile est de 10000 octets et ont la même priorité 1.

```
void setup(){
xTaskCreate ( task1, " Hello ", 10000, NULL, 1, NULL);
xTaskCreate ( task2, " Touch ", 10000, NULL, 1, NULL);
}
```

Dans la boucle principale, nous ne ferons rien car nos tâches implémenteront toutes les fonctionnalités. Alors mettons juste un retard.

```
void loop() {
delay (1000);
}
```

Nous allons implémenter les deux fonctions de manière très simple. Nous allons faire une boucle simple où nous allons dans la première tâche imprimer un «Hello from task 1», la deuxième tâche mesure les valeurs de capteur de touche et imprimer ces valeurs et après toutes les itérations de la boucle, nous imprimerons un message indiquant que la tâche se terminera.

```
void task1( void * parameter ){
for( int i = 0;i<10;i++ ){
Serial.println ("Hello from task 1");
delay (1000);
   }
Serial.println("Ending task 1");
vTaskDelete( NULL);
 }
void task2( void * parameter ){
for (int i = 0; i<10; i++){
int value2=touchRead(T0);
Serial.print ("valeur touch ");
Serial.print (value2);
Serial.print("|");
delay (1000);
}
Serial.println ("Ending task 2");
vTaskDelete(NULL );
}
```

Pour supprimer une tâche depuis son propre code, il suffit d'appeler la fonction vTaskdelete. Cette fonction reçoit en entrée le handle de la tâche à supprimer. Néanmoins, si nous passons NULL en entrée, la tâche appelante sera supprimée, ce que nous voulons, puisque nous allons l'appeler à partir du propre code de la tâche.

Après téléchargement du code sur la carte ESP32, les résultats obtenus après exécution du code sont illustrés sur la figure 2.17.

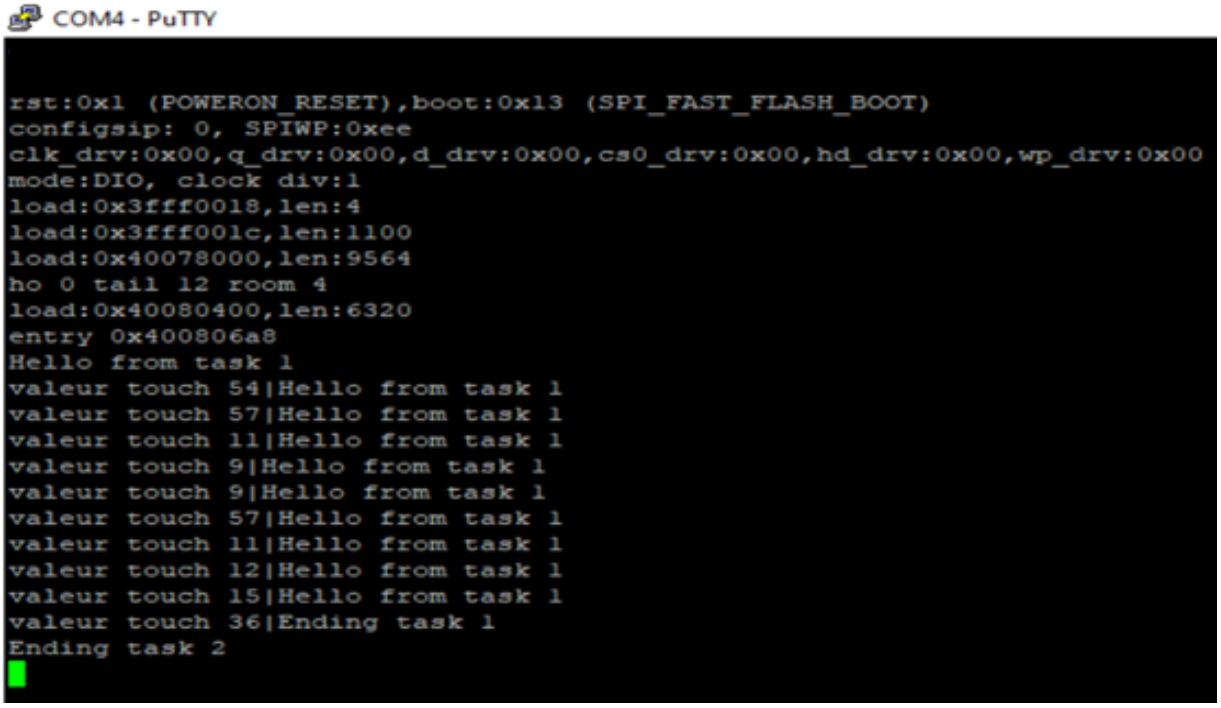

Figure 2.17 : Résultats du programme création d'une tâche FreeRTOS.

On a remarqué que les deux tâches s'exécutaient en parallèle, ce qui permet de mélanger les impressions de chaque tâche .l'heure d'exécution de chacune des tâches était décidée par le planificateur RTOS.

La façon la plus simple pour communiquer entre deux tâches c'est l'utilisation de Mailbox. Dans FreeRTOS cela s'appelle une Queue car cela gère des files d'attente de toutes sortes.

2.6.5 La file d'attente (Queue)

La file d'attente est une structure de données qui peut contenir un certain nombre d'éléments dont la taille est configurable. Semblable à une file d'attente dans un supermarché où les gens attendent de passer à la caisse. Celui qui vient en premier dans la file d'attente obtient la première position, puis une autre personne arrive et remplit la deuxième position, et ainsi de suite. Lorsque le caisserie commence à servir, la première personne sort de la file d'attente, puis la deuxième personne avance d'une étape et reste à la place de la première personne, tout comme chacun dans

la file d'attente. Une file d'attente peut être caractérisée par le nombre de personnes qu'elle peut contenir, si la file d'attente est pleine, personne ne peut y entrer [32].

La file d'attente (Queue) sont la principale forme de communication Inter-tâches. Ils peuvent être utilisés pour envoyer des messages entre les tâches ou entre les interruptions et les tâches. Dans la plupart des cas, Ils sont généralement utilisés comme FIFO (First In First Out), ce qui signifie que de nouvelles données sont insérées à l'arrière de la file d'attente et consommées par l'avant. Les données à transmettre peuvent être de type entier ou pointeur.

Un aspect très important à garder à l'esprit est que les données insérées dans la file d'attente sont copiées plutôt que seule une référence à leur stockage. Cela signifie que si nous envoyons un entier à la file d'attente, sa valeur sera réellement copiée et si nous modifions la valeur d'origine après cela, aucun problème ne devrait se produire [33].

2.6.5.1 Caractéristique d'une file d'attente

2.6.5.1.1 Stockage des données

Une file d'attente peut contenir un nombre limité d'éléments de données de taille fixe. Le nombre maximum d'éléments qu'une file d'attente peut contenir est appelé sa *longueur*. La longueur et la taille de chaque élément de données sont définies lorsque la file d'attente est créée.

Les files d'attente sont normalement utilisées en tant que tampons premier entré, premier sorti (FIFO), où les données sont écrites à la fin (queue) de la file d'attente et supprimées de l'avant (tête) de la file d'attente.

L'illustration suivante montre des données écrites dans et lues depuis une file d'attente qui est utilisée comme FIFO. Il est également possible d'écrire à l'avant d'une file d'attente et de remplacer les données qui sont déjà à l'avant d'une file d'attente.

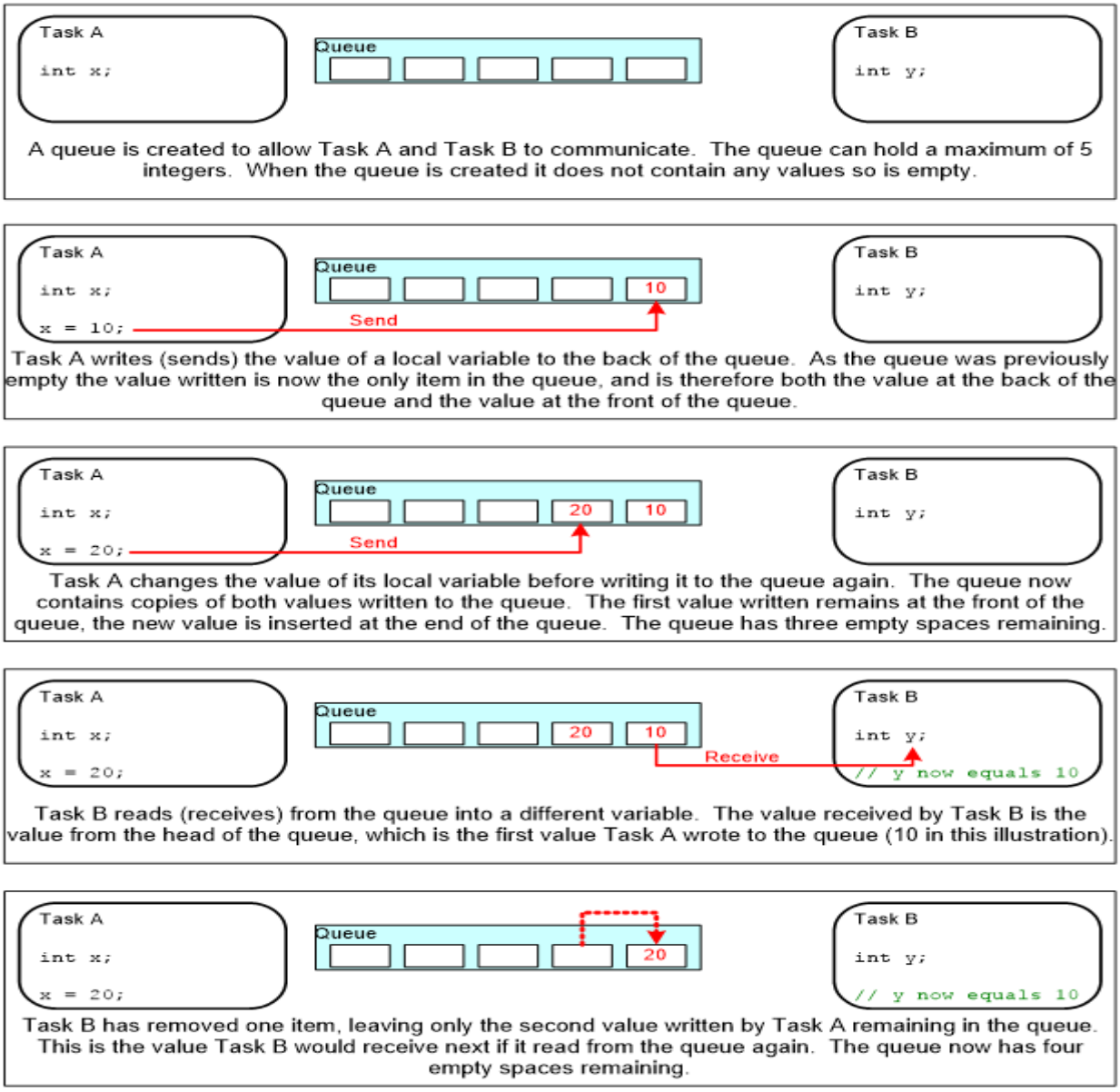

Figure 2.18 : Déroulement des données écrites et lues depuis une file d'attente.

Il existe deux manières d'implémenter un comportement de file d'attente par copie ou par référence. FreeRTOS utilise la file d'attente par copie. Cette méthode est considérée comme plus puissante et plus simple à utiliser que la mise en file d'attente par référence.

2.6.5.1.2 Accès par plusieurs tâches

Les files d'attente sont des objets qui sont accessibles par toute tâche ou Registre des services d'interruption (ISR) qui connaît leur existence. N'importe quel nombre de tâches peut écrire dans la même file d'attente et n'importe quel nombre de tâches peut lire à partir de la même file d'attente. Il est très courant pour une file d'attente d'avoir plusieurs rédacteurs, mais nettement moins fréquent pour une file d'attente d'avoir plusieurs lecteurs [34].

2.6.5.1.3 Blocage sur les files d'attente

Les fonctions de l'API de file d'attente permettent de spécifier un temps de blocage.

Lorsqu'une tâche tente de lire à partir d'une file d'attente vide, la tâche est placée dans l'état bloqué (elle ne consomme donc pas de temps CPU et d'autres tâches peuvent s'exécuter) jusqu'à ce que les données deviennent disponibles dans la file d'attente ou que le temps de blocage expire.

Lorsqu'une tâche tente d'écrire dans une file d'attente complète, la tâche est placée dans l'état bloqué jusqu'à ce que de l'espace soit disponible dans la file d'attente ou que le temps de blocage expire.

Si plusieurs blocs de tâches sur la même file d'attente, la tâche avec la priorité la plus élevée sera la tâche qui est débloquée en premier [35].

Comme nos tâches devront pouvoir accéder à la file d'attente, nous allons commencer notre code on déclarant une variable globale de **type QueueHandle_t.**

```
QueueHandle_t queue;
```

Dans la fonction de configuration, nous commencerons par ouvrir une connexion série Ensuite, nous allons créer notre file d'attente en appelant la fonction **xQueueCreate**.

```
QueueHandle_t xQueueCreate (UBaseType_t uxQueueLength,UBaseType_t uxItemSize);
```

La fonction **xQueueCreate** reçoit deux paramètres :

- **uxQueueLength :** Le nombre maximum d'éléments que la file d'attente peut contenir à tout moment.
- **uxItemSize :** La taille, en octets, requise pour contenir chaque élément de la file d'attente.

En retour si la file d'attente est créée avec succès, un descripteur de la file d'attente créée est renvoyé. Si la mémoire requise pour créer la file d'attente n'a pas pu être allouée, NULL est renvoyé [36].

```
Serial.begin (115200);
```

```
queue = xQueueCreate( 10, sizeof ( int ) );
if (queue == NULL){
Serial.println("Error creating the queue");
 }
```

Maintenant que nous avons la file d'attente, nous pouvons créer les tâches. L'un d'eux, appelé producteur, mettra des valeurs dans la file d'attente, et l'autre, appelé consommateur, consommera les valeurs de la file d'attente. Nous avons mentionné précédemment comment créer une tâche :

```
void setup(){
Serial.begin (115200);
queue = xQueueCreate( 10, sizeof (int ) );
if (queue == NULL){
Serial.println("Error creating the queue");
}
xTaskCreate( task1, " producteur ", 10000, NULL, 1, NULL);
xTaskCreate (task2, " consommateur ", 10000, NULL, 1, NULL);
}
```

La première tâche est la tâche du producteur, cette tâche mettra simplement des éléments mesurés à partir de capteur de touche dans la file d'attente. Nous allons donc l'implémenter dans une boucle allant de 0 à la taille de la file d'attente moins 1. Pour faire l'insertion réelle de l'élément, nous appelons la fonction **xQueueSend**

```
BaseType_t xQueueSend (
                    QueueHandle_t xQueue,
                    const void * pvItemToQueue,
                    TickType_t xTicksToWait
                  );
```

La fonction **xQueueSend** reçoit les paramètres suivants :

- **xQueue :** Le handle de la file d'attente sur lequel l'élément doit être publié.

- **pvItemToQueue :** Un pointeur vers l'élément à placer dans la file d'attente. La taille des éléments que la file d'attente contiendra a été définie lors de la création de la file d'attente, de sorte que ce nombre d'octets sera copié de **pvItemToQueue** dans la zone de stockage de la file d'attente.
- **xTicksToWait :** Durée maximale pendant laquelle la tâche doit bloquer en attendant que de l'espace devienne disponible dans la file d'attente, si elle est déjà pleine. L'appel sera renvoyé immédiatement si la file d'attente est pleine et que **xTicksToWait** est défini sur 0. L'heure est définie en périodes de graduation, donc la constante **portTICK_PERIOD_MS** doit être utilisée pour convertir en temps réel si cela est nécessaire [37].

```
void task1( void * parameter ){
for (int i = 0; i<10; i++){
int value1 =touchRead(T0);
xQueueSend(queue, &value1, portMAX_DELAY);
}
Serial.println ("Ending task 1");
vTaskDelete( NULL );
}
```

La deuxième tâche est la tâche du consommateur cette tâche s'exécutera également en boucle et consommera les éléments précédemment insérés, Pour obtenir l'élément réel de la file d'attente, nous devons appeler la fonction **xQueueReceive.**

```
BaseType_t xQueueReceive (
                    QueueHandle_t xQueue,
                    void *pvBuffer,
                    TickType_t xTicksToWait
                  );
```

La fonction **xQueueReceive** reçoit les paramètres suivants :

- **xQueue :** Le handle de la file d'attente à partir de laquelle l'élément doit être reçu.
- **pvBuffer :** Pointeur vers la mémoire tampon dans laquelle l'élément reçu sera copié.

- **xTicksToWait :** Durée maximale pendant laquelle la tâche doit bloquer l'attente d'un élément à recevoir si la file d'attente est vide au moment de l'appel. La définition de xTicksToWait sur 0 entraînera le retour immédiat de la fonction si la file d'attente est vide. L'heure est définie en périodes de graduation, donc la constante portTICK_PERIOD_MS doit être utilisée pour convertir en temps réel si cela est nécessaire.

Retour :

pdTRUE si un élément a été reçu avec succès de la file d'attente, sinon pdFALSE [38].

```
void task2( void * parameter ){
 int value2;
 for (int i = 0; i<10; i++){
 xQueueReceive(queue, &value2, portMAX_DELAY);
 Serial.print (value2);
 delay (2000);
 Serial.print ("|");
}
Serial.println("Ending task 2");
vTaskDelete( NULL );
 }
```

Les résultats du test de code sont exposés sur la figure 2.19. Les valeurs insérées dans la file d'attente par la tâche de producteur sont imprimées par le consommateur, dans le même ordre.

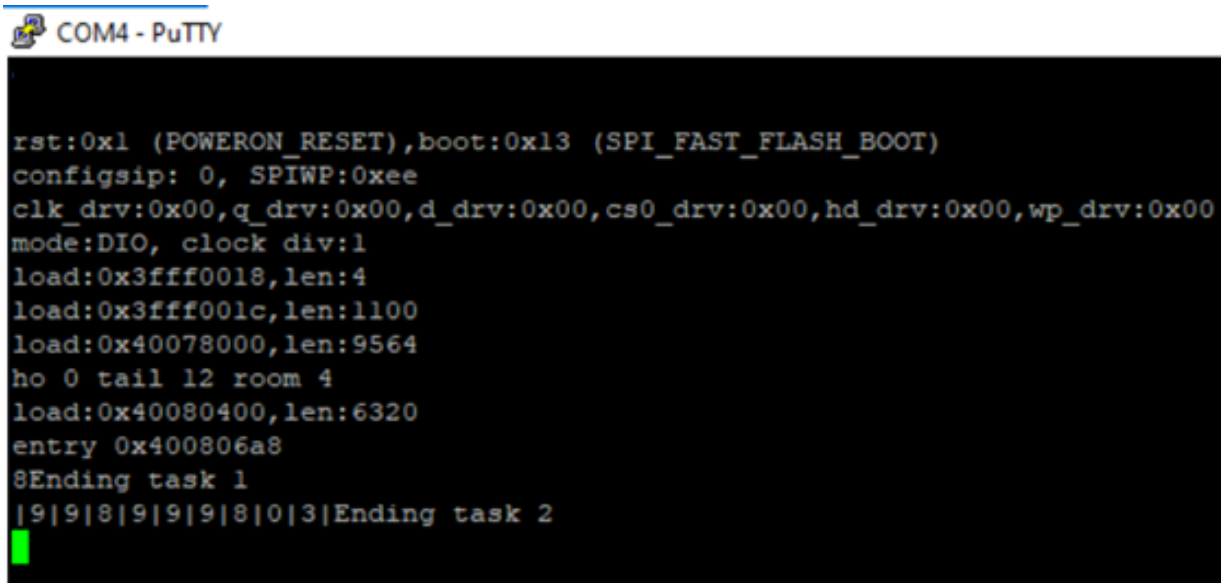

Figure 2.19 : Résultats du programme de communication inter-tâches.

2.7 Conclusion

Dans ce chapitre nous avons présenté la conception de notre réseau capteurs pour la maintenance prédictive de machines. Notre RCSF est constitué des nœuds ESP32. Plusieurs programmée sont développés sous IDE Arduino à l'aide de la fonction PainlessMesh. Les messages entre les nœuds sont basés sur la notation JSON. L'acquisition et la transmission des données dans le réseau se fait en temps réel à l'aide de la gestion des taches.

Chapitre 3

Traitement du signal et techniques d'analyse

2.8 Introduction

Le traitement du signal qui a pour objet l'élaboration ou l'interprétation des signaux porteurs d'informations, est considéré comme une discipline indispensable de nos jours. Son objectif est d'extraire un maximum d'information utile sur un signal perturbé par du bruit. Ce dernier correspond à tout phénomène perturbateur gênant la transmission ou l'interprétation d'un signal.

Le diagnostic industriel basé sur le traitement de signal est parmi les techniques les plus efficaces et les plus utilisées dans l'industrie, surtout ce qui concerne la détection des défauts affectant les machines tournantes (les défauts des roulements, les défauts des engrenages...etc.) La détection d'un défaut nécessite d'une part une prise de mesure vibratoire, via une chaîne d'acquisition, puis une exploitation du signal recueilli.

Dans ce chapitre nous présenterons quelques généralités sur le traitement du signal, la structure générale de la chaîne d'acquisition, les capteurs et le type de leurs signaux, Finalement nous présenterons quelques techniques d'analyse des signaux, dans ce sens-là l'analyse « en fréquence » est l'outil fondamental pour le traitement des signaux vibratoires. Elle s'appuie sur la transformée de Fourier.

Le spectre de fréquences est généré via l'algorithme de la transformée de Fourier rapide noté FFT (Fast Fourier Transform). Pour cela, nous simulons l'algorithme FFT dans l'environnement de développement IAR Embedded, pour illustrer la transformation du signal de domaine temporelle au domaine fréquentielle, ce qui permet de connaître la totalité des composantes spectrales du signale pour détecter la présence d'un défaut, qui est l'objectif principal de ce chapitre. Les données fréquentielle de type complexe donc, l'amplitude de ces valeurs doit être calculée et introduites dans un vecteur, afin de faciliter le processus de transfert d'informations dans le réseau discutés dans les sections précédentes du chapitre 2.

2.9 Traitement de signal

Le traitement du signal (T.S) est une discipline technique qui a pour objet l'élaboration, la détection et l'interprétation des signaux porteurs d'informations. Cette discipline s'appuie sur la théorie du signal qui donne une description mathématique des signaux [39].

Cette représentation mathématique du signal vise à établir des relations entre les différentes représentations et à créer des modifications distinctes de signaux lorsqu'ils sont appliqués à des systèmes physiques. Le but du traitement de signal est la manipulation des signaux, l'étude de leur transformation et les moyens qui les permettent, ainsi que la séparation des signaux utiles et des bruits ce qui comprends les techniques de détection, d'estimation et de classification.

2.9.3 Signal

Le signal est la représentation physique de l'information, qu'il convoie de sa source à son destinataire. La description mathématique des signaux est l'objectif de la théorie du signal. Cette représentation commode du signal permet de mettre en évidence ses principales caractéristiques (distribution fréquentiel, énergie, etc.) et d'analyser les modifications subies lors de la transmission ou du traitement de ces signaux.

L'ensemble des signaux peut être classé en deux grandes classes :

- **Signaux déterministes** : ou signaux certains, leur évolution en fonction du temps peut être modélisé par une fonction mathématique. Dans cette classe, on trouve les signaux périodiques, les signaux transitoires, les signaux pseudo-aléatoires, etc.
- **Signaux aléatoires** : Il s'agit des signaux dont le modèle mathématique n'est pas connu et leur évolution en fonction de temps est imprévisible [40].

Le signal, reproduction d'un phénomène physique qui évolue dans le temps ou dans l'espace est représenté par une fonction **s(t)** à valeurs réelles d'une variable réelle t (temps).

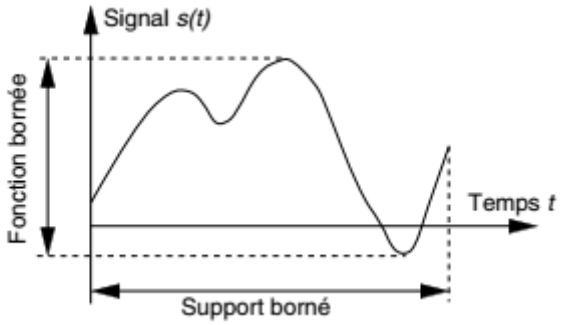

Figure 3.1 : Signal physique.

En pratique le signal physique n'est pas étudié, mais seulement la représentation numérique est utilisée pour réaliser les calculs voulus. La numérisation du signal est l'opération qui consiste à faire passer un signal de la présentation dans le domaine des temps et des amplitudes continus au domaine des temps et amplitudes discrets. Cette opération de numérisation d'un signal peut être décomposée en deux étapes principales :

- échantillonnage
- quantification

a) Echantillonnage :

L'échantillonnage consiste à prélever à des instants précis, le plus souvent équidistants, les valeurs instantanées d'un signal. Le signal analogique continue ***s(t)*** est présenté par un ensemble de valeurs discrètes ***s(n, Te)*** :

$$S_e(t) = S(n, T_e) \qquad \begin{cases} n & entier \\ T_e & période \quad d'échantiollannage \end{cases} \qquad \textbf{3.1}$$

La période d'échantillonnage $\boldsymbol{T_e}$ (en seconde) est constante et représente le temps entre deux mesures successives. Cette opération est réalisé par un circuit appelé préleveur ou échantillonneur symbolisé souvent par un interrupteur.

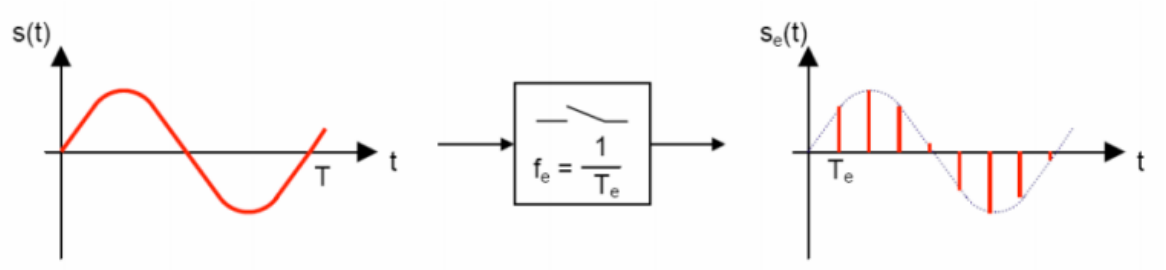

Figure 3.2 : Echantillonnage d'un signal **s(t)**.

La fréquence d'échantillonnage $\boldsymbol{F_e}$, correspond au nombre de mesures effectuées par seconde :

$$F_e = \frac{1}{T_e} \qquad \textbf{3.2}$$

Le choix de la fréquence d'échantillonnage est crucial afin de reproduire fidèlement le signal étudié. En effet si le signal analogique varie trop vite par rapport à la fréquence d'échantillonnage, la numérisation il faut et il suffit que la fréquence d'échantillonnage $\boldsymbol{F_e}$ égale ou supérieure deux fois la fréquence maximale $\boldsymbol{F_{max}}$ du signal initial. C'est le théorème d'échantillonnage appelle aussi théorème de Shannon :

$$F_e \geq 2 * F_{\max} \qquad \textbf{3.3}$$

En pratique, la valeur couramment choisie pour la fréquence d'échantillonnage est :

$$F_e = 2{,}56 * F_{\max} \qquad \textbf{3.4}$$

- ✓ Le théorème de SHANNON montre que la reconstitution correcte d'un signal nécessite que la fréquence d'échantillonnage F_e soit au moins deux fois plus grande que la plus grande des fréquences F_{max} du spectre du signal : $F_e > F_{max}$.

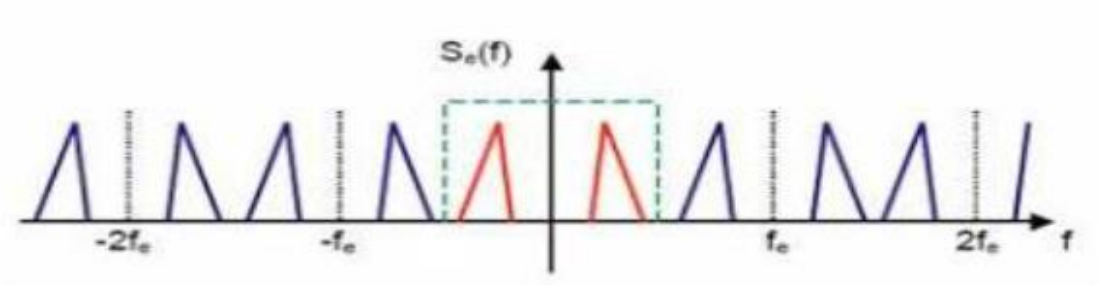

Figure 3.3 : Exemple d'un échantillonnage parfait.

- ✓ Si F_{max} la fréquence maximale du spectre du signal à échantillonner, est supérieure $F_e/2$ restitution du signal original sera impossible car il va apparaître un recouvrement spectral lors de l'échantillonnage. On dit qu'on est en sous-échantillonnage.

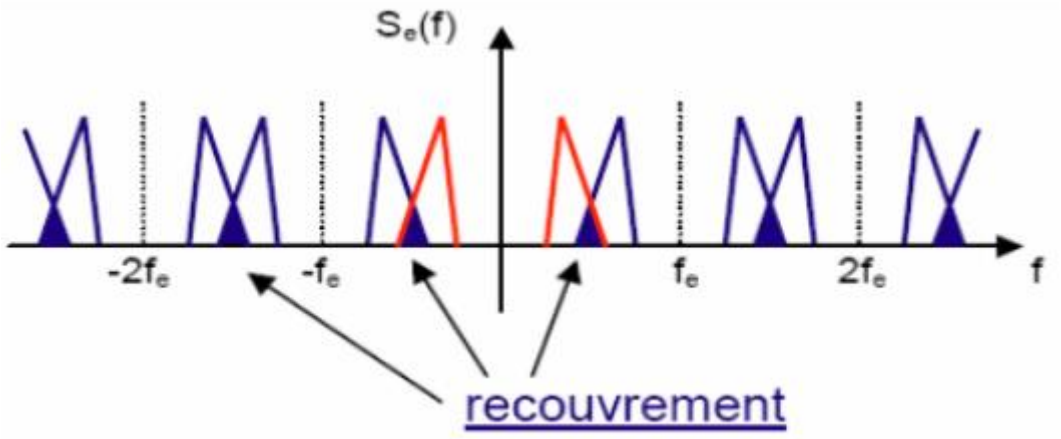

Figure 3.4 : Exemple d'un mauvais échantillonnage (recouvrement).

b) Quantification : permet d'associer une valeur numérique à l'échantillon prélevé. C'est une discrétisation en amplitude. Les valeurs discrètes obtenues sont codés sur un ou plusieurs bits. La qualité du signal numérique dépend également directement du nombre de bits utilisés pour coder le signal source.

Le nombre de valeurs différentes pouvant être discrétisées est égal à 2^n (n est le nombre de bits utilisés)

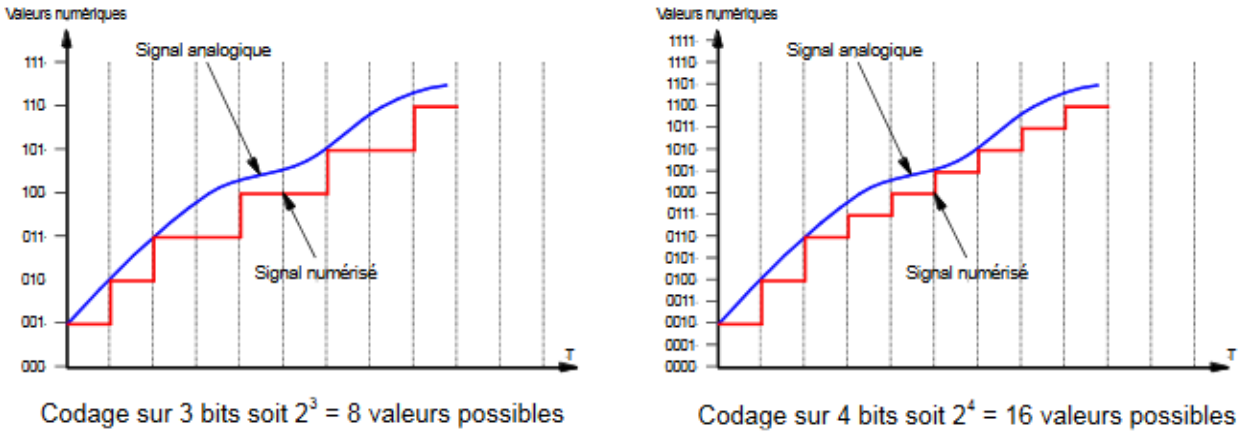

Figure 3.5 : Codage sur 3 bits et 4 bits.

2.9.4 Résolution

Comme mentionné précédemment le signal numérique ne peut prendre que certaines valeurs (quantification). Chaque valeur est arrondie à la valeur permise la plus proche par défaut (juste en dessous). On appel résolution l'écart constant entre deux valeurs permise successives. Pour réaliser de bonnes mesures, il est nécessaire de choisir une résolution appropriée dans la bande de fréquences de travail. Une résolution est satisfaisante lorsqu'elle permet de dissocier des fréquences voisines. Elle dépend de la largeur de la bande de fréquences, du nombre de points N_e et du nombre de lignes de l'analyseur [41].

Avec :

$$\Delta f = \frac{f_e}{N_e} = \frac{1}{N_e \cdot T_e} \qquad \textbf{3.5}$$

N_e : Nombre de points enregistrés.

T_e : Période d'échantillonnage.

F_e : Fréquence d'échantillonnage.

2.9.5 Représentation d'un signal vibratoire

- **Représentation temporelle :**

Le signal vibratoire délivré par un capteur peut être représenté de différentes façons.

La première est la représentation de chaque évènement en fonction de sa progression dans le temps (représentation temporelle). Cette représentation est utilisée pour suivre le comportement et l'évolution vibratoire d'une machine en fonction du temps. Ce type de représentation temporelle

est aisé à exploiter lorsque le signal délivré par le capteur est simple. Il est encore facile à exploiter lorsque ce signal a fait l'objet d'un traitement approprié (filtrage), mais il devient inextricable lorsque le signal a pour origine des sollicitations multiples.

- **Représentation fréquentielle :**

La représentation temporelle du signal vibratoire devient difficile à interpréter lorsque le signal est complexe, on a recherché à le représenter dans un diagramme amplitudes en fonction des fréquences appelé spectre. Avec ce type de représentation, chacune des composantes sinusoïdales élémentaires constituant le signal est parfaitement défini par son amplitude et sa fréquence.

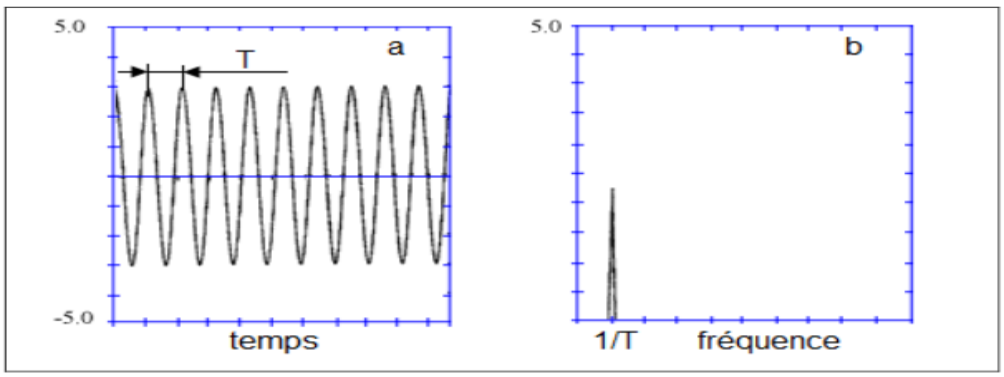

Figure 3.6 : Représentation temporelle (a), et Fréquentielle(b) d'un signal sinusoïdal.

2.10 Chaîne d'acquisition

2.10.3 Définition

La chaîne d'acquisition de données est l'ensemble des éléments nécessaires à la "capture" des données (analogiques ou numériques) à leur transmission jusqu'au récepteur et à l'utilisateur (homme ou machine) des données capturées. Ces dernières sont soit utiliser immédiatement ou stoker pour une utilisation ultérieur.

La chaîne d'acquisition peut assurer plusieurs rôles :

- recueillir les informations nécessaires à la connaissance de l'état d'un système.
- délivrer ces informations sous une forme appropriée à leur exploitation, sachant que l'état d'un système est caractérisé par des grandeurs physiques ou chimiques appelées mesurandes.
- Assigner une valeur (un nombre) à un mesurande.

2.10.4Structure d'une chaîne d'acquisition numérique

Une chaîne d'acquisition numérique peut se représenter selon la figure suivante :

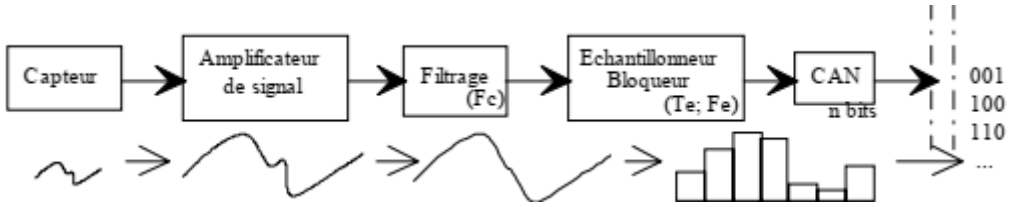

Figure 3.7 : Structure d'une chaine d'acquisition numérique.

Elle est souvent associée à une chaîne de restitution qui fonctionne dans le sens inverse de la précédente

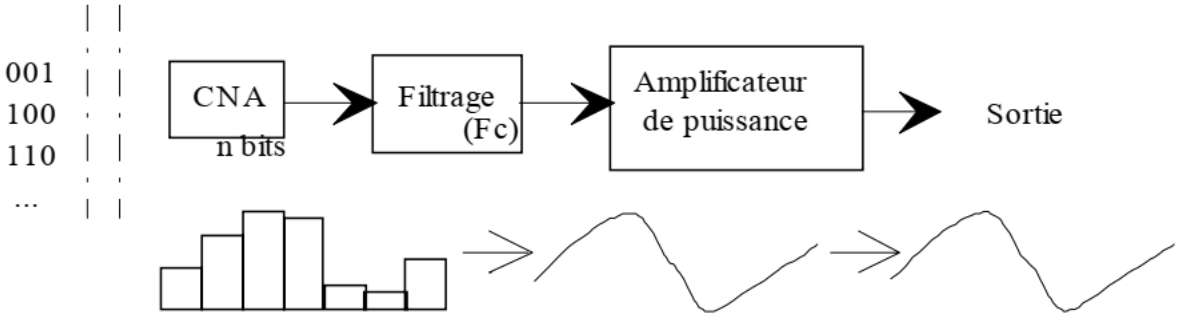

Figure 3.8 : Structure de la chaîne de restitution.

On peut définir très simplement le rôle de chacun des éléments [42].

- **Capteur** : Il est l'interface entre le monde physique et le monde électrique. C'est un transducteur qui va délivrer un signal électrique image du phénomène physique que l'on souhaite numériser. Il est toujours associé à un circuit de mise en forme.
- **Amplificateur de signal** : Cette étape permet d'adapter le niveau du signal issu du capteur à la chaîne globale d'acquisition.
- **Filtre d'entrée** : Ce filtre est communément appelé filtre anti-repliement. Son rôle est de limiter le contenu spectral du signal aux fréquences qui nous intéressent. Ainsi il élimine les parasites. C'est un filtre passe bas que l'on caractérise par sa fréquence de coupure et son ordre.
- **L'échantillonneur** : Son rôle est de prélever à chaque période d'échantillonnage (T_e) la valeur du signal. On l'associe de manière quasi-systématique à un bloqueur. Le bloqueur va figer l'échantillon pendant le temps nécessaire à la conversion. Ainsi durant la phase de numérisation, la valeur de la tension de l'échantillon reste constante assurant une conversion aussi juste que possible. On parle d'échantillonneur bloqueur

- **Le convertisseur analogique numérique (CAN)** : Il transforme la tension de l'échantillon (analogique) en un code binaire (numérique).
- **La zone de stockage** : Elle peut être un support de traitement (DSP, ordinateur), un élément de sauvegarde (RAM, Disque dur) ou encore une transmission vers un récepteur situé plus loin.
- **Le convertisseur numérique analogique (CNA) :** Il effectue l'opération inverse du CAN, il assure le passage du numérique vers l'analogique en restituant une tension proportionnelle au code numérique.
- **Le filtre de sortie :** Son rôle est de « lisser » le signal de sortie pour ne restituer que le signal utile. Il a les mêmes caractéristiques que le filtre d'entrée.
- **Amplificateur de puissance :** Il adapte la sortie du filtre à la charge.
- **Actionneur :** charge passive ou active agissant sur le système à monitorer.

2.11 Les capteurs

Le capteur est le premier élément de la chaîne d'acquisition des données. Il a pour fonction de délivrer un signal électrique de sortie ***S*** qui peut être soit **:**

- une charge
- une tension
- un courant
- une impédance (R, L, C)

Il transforme la mesure de la grandeur physique, qu'on appelle mesurande ***m***, en grandeur électrique ***S(m)*** **(Figure 3.9).**

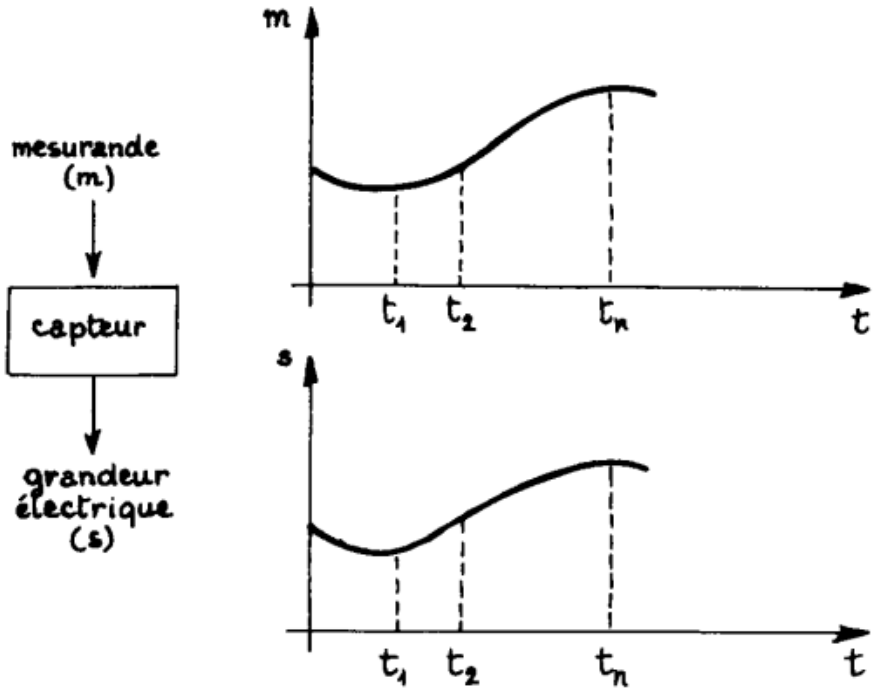

Figure 3.9 : Principe fonctionnement d'un capteur**.**

Les capteurs sont classés en fonction de la caratéristique électrique de la grandeur de sortie en deux classes : les capteurs passifs (fonctionnent sans alimentation électrique) et les capteurs actifs (nécessitent une alimentation électrique**)** [43].

- **Capteurs passifs :**

Il s'agit généralement d'impédance dont l'un des paramètres déterminants est sensible à la grandeur mesurée. La variation d'impédance résulte :

- Soit d'une variation de dimension du capteur, c'est le principe de fonctionnement d'un grand nombre de capteur de position, potentiomètre, inductance à noyaux mobile, condensateur à armature mobile.
- Soit d'une déformation résultant de force ou de grandeur s'y ramenant, pression accélération (Armature de condensateur soumise à une différence de pression, jauge d'extensomètre liée à une structure déformable).

L'impédance d'un capteur passifs et ses variations ne sont mesurables qu'en intégrant le capteur dans un circuit de conditionnement électronique qui permet son alimentation et l'adaptation du signal à la sortie (**Figure 3.10) [**44].

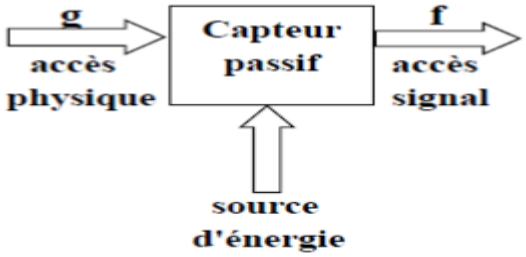

Figure 3.10 : Schéma bloc d'un capteur passif.

Tableau 3.1 : Type de matériau utilisé et caractéristique électrique des capteurs passifs.

MESURANDE	EFFET UTILISE (Grandeur de sortie)	MATERIAUX
Température	Résistivité	Platine, nickel, cuivre, semi-conducteurs
Très basse température	Constante diélectrique	Verre
Flux optique	Résistivité	Semi-conducteurs
Déformation	Résistivité	Alliages nickel Alliages ferromagnétiques
Position	Résistivité Perméabilité	Magnétorésistances : Bismuth, antimoine d'indium
Humidité	Résistivité	Chlorure de lithium

- **Capteurs actifs :**

Fonctionnant en générateur, un capteur actif est généralement fondé dans son principe sur un effet physique qui assure la conversion en énergie électrique de la forme d'énergie propre à la grandeur physique à prélever, énergie thermique, mécanique ou de rayonnement. Donc, un capteur actif produit lui-même un signal électrique de sortie par conversion de l'énergie fournie par le grandeur d'entrée ou de ces variations. On va schématiser dans la Figure () ce type de capteur par un bloc possédant un accès "physique" et un accès "signal" [44].

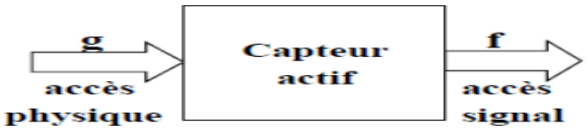

Figure 3.11 : Schéma bloc d'un capteur actif.

Tableau 3.2 : Grandeurs d'entrée et de sortie et effet utilisé pour les capteurs actifs**.**

MESURANDE	EFFET UTILISE	Grandeur de sortie
Température	Thermoélectricité (thermocouple)	Tension
Flux optique	Photoémission Pyroélectricité	Courant Charge
Force, pression, accélération	Piézoélectricité	Charge
Position	Effet Hall	Tension
Vitesse	Induction	Tension

2.11.3Constitution d'un capteur

Un capteur est constitué principalement de 3 parties : un corps d'épreuve, un élément de transduction et d'un boitier, Figure 3.12**.** Selon les cas, il peut être complété par un module électronique de conditionnement "transmetteur" [45].

- **Le corps d'épreuve** (tête ou dispositif de commande) est élément qui réagit sélectivement à la grandeur qu'il faut mesurer. Il transforme la grandeur à mesurer en une autre grandeur physique dite mesurable, cette grandeur constitue la réaction du corps d'épreuve.
- **L'élément de transduction** est une partie sensible lié au corps d'épreuve. Il traduit les réactions du corps d'épreuve en une grandeur électrique constituant le signal de sortie.
- **Le boîtier (ou corps)** est un élément mécanique de protection, de maintien et de fixation du capteur.
- **Module électronique de conditionnement :** il a, selon les cas, les fonctions suivantes :
    - alimentation électrique du capteur (si nécessaire)
    - mise en forme et amplification du signal de sortie
    - filtrage, amplification
    - conversion du signal (CAN,...)

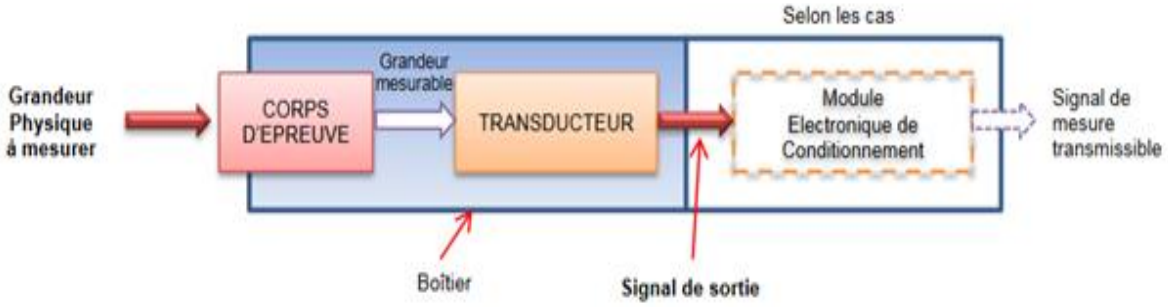

Figure 3.12 : Constituants d'un capteur.

2.11.4Grandeurs d'influence

Les grandeurs d'influence sont des grandeurs étrangères qui, selon leur nature et leur importance, peuvent provoquer des perturbations sur le capteur. C'est donc une cause d'erreurs agissant sur le signal de sortie. Les principales grandeurs d'influence sont [44] :

- **la température** qui modifie les caractéristiques électriques, mécaniques et dimensionnelles des composants du capteur ;
- **La pression, l'accélération et les vibrations** susceptibles de créer dans certains éléments constitutifs du capteur des déformations et des contraintes qui altèrent la réponse.
- **L'humidité** à laquelle certaines propriétés électriques comme la constante diélectrique ou la résistivité peuvent être sensibles et qui risque de dégrader l'isolation électrique entre composants du capteur ou entre le capteur et son environnement.
- **Les champs magnétiques variables ou statiques** ; les premiers créent des f.é.m. d'induction qui se superposent au signal utile, les seconds peuvent modifier une propriété électrique.
- **La tension d'alimentation**

3.4.3 Nature des signaux délivrés par le capteur

L'information transmise par un capteur peut être :

- Logique
- Analogique
- Numérique.

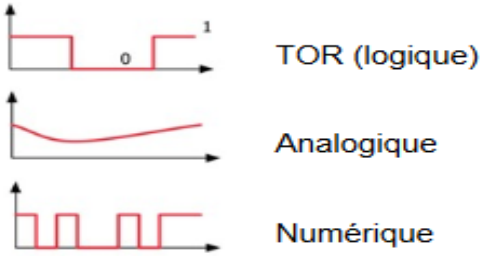

Figure 3.13 : Nature des signaux délivrés par le capteur.

2.11.4.1Signaux logiques

L'amplitude de la grandeur physique le représentant ne peut prendre que deux états généralement représenter par 0 et 1. Connues aussi par le nom signal TOR (**T**out **O**u **R**ien)

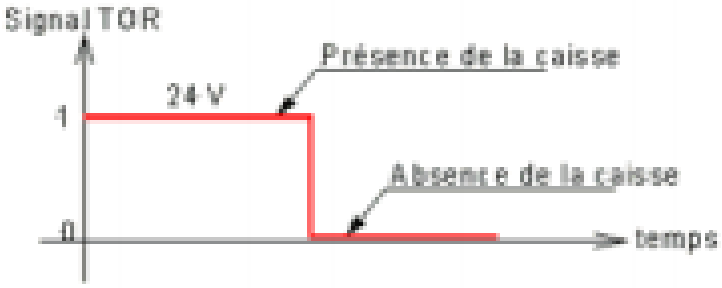

Figure 3.14 : Signal TOR.

2.11.4.2Signaux analogiques

L'amplitude de la grandeur physique le représentant peut prendre une infinité de valeurs dans un intervalle donné.

2.11.4.3Signaux numériques

Signal composé d'un nombre fini de valeurs numériques. Les informations délivrées par le capteur numérique peuvent être sous la forme d'un code binaire (avec un nombre de bits définis), d'un train d'impulsions (avec un nombre précis d'impulsions ou avec une fréquence précise)

- **Transmission du signal numérique :**

Les capteurs numériques sont capables de transmettre des valeurs déterminant des positions, des pressions, des températures, etc... Les informations qui sont des combinaisons de signaux 0-1, sont transmises à l'unité de traitement et peuvent être lues soit en parallèle, soit en série [45].

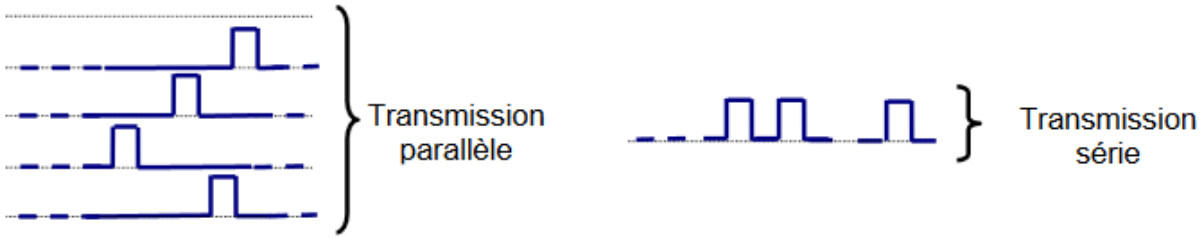

Figure 3.15 : Transmission du signal numérique.

2.11.5Capteur intelligent

Le capteur intelligent est un capteur intégrant une interface de communication bidirectionnelle et un microcontrôleur/DSP. L'interface de communication permet de commander à distance le capteur et d'en gérer plusieurs

Le microcontrôleur permet de gérer les différentes mesures et de corriger les erreurs dues à des variations de grandeurs physiques parasites (exemple : mesure simultanée de la température pour corriger la dérive thermique) [46].

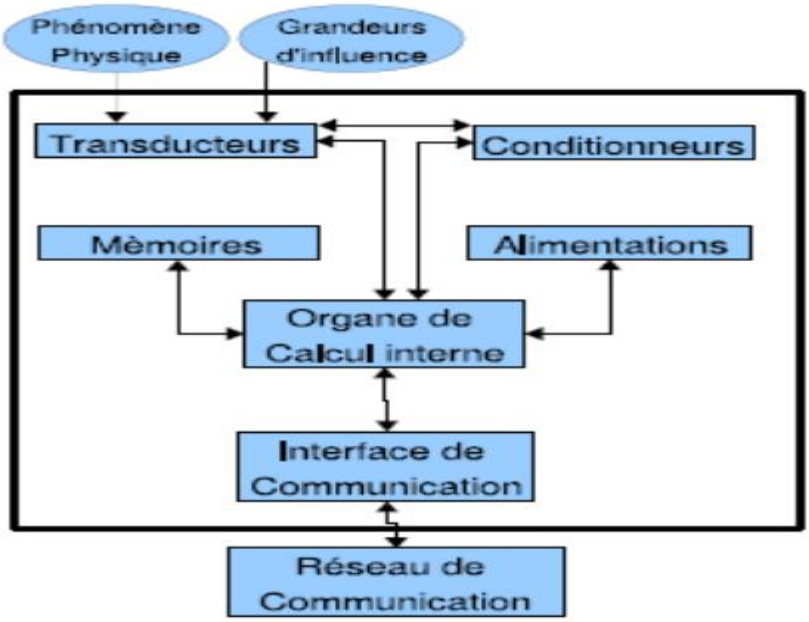

Figure 3.16 : Architecture générique de capteur intelligent.

Cette architecture générique regroupe les composants de base qui permettent d'assurer les fonctionnalités attendues d'un capteur intelligent (mesure, validation, configuration, communication).

Les capteurs utilisés dans notre travail sont des capteurs intelligents intégrés dans la carte de développement du microcontrôleur ESP32.

2.11.6Test de capteur tactile capacitif

Pour cela nous avons utilisé :

- ESP 32(DIVE KIT .C)
- Fils conducteurs (Jumper wires)
- câble USB
- afficheur (pc)

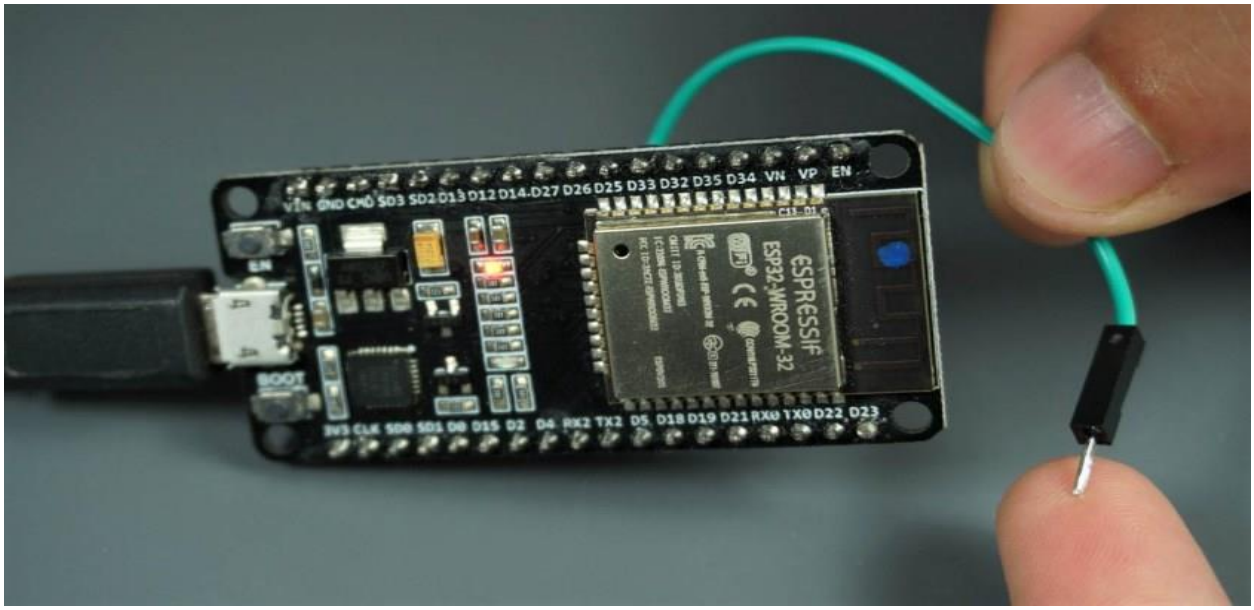

Figure 3.17 : Test de capteur tactile capacitif.

Le sketch suivant affiche dans le moniteur série la valeur retournée par touchRead()

```
int touchRead();
 int touchPin=4;//GPIO4 TOUCH0
 int touchValue;
 int  threshold=30;
void setup(){
  Serial.begin (115200);
 delay (1000); // give me time to bring up serial monitor
 Serial.println ("ESP32 Touch Test");
 }
void setup(){
   Serial.println ("threshold");
    Serial.println (threshold);
    touchValue = touchRead(touchPin);
```

```
Serial.println ("touchValue");// get value using T0
 Serial.println (touchValue);
 if (touchValue>threshold){
Serial.println ("touchValue>threshold");
}
 else{
   Serial.println ("threshold");
  Serial.println (threshold);
   Serial.println ("touchValue");
   Serial.println (touchValue);
   Serial.println ("touchValue<threshold");
   }
delay(1000);}
```

Cet exemple lit la broche tactile 0 et affiche les résultats dans le moniteur série.

- Nous définissons la broche T0 (touche broche 0), correspond à GPIO 4**,** pour le test **: int touchPin=4;//GPIO4 TOUCH0**
- Nous allons démarrer notre fonction de configuration en ouvrant une connexion série, afin de pouvoir obtenir la sortie de notre programme de test : **Serial.begin (115200) ;**
- Dans la boucle (), lire la broche GPIO4 du capteur via l'argument **touchRead(touchPin);** et faire la comparaison avec la valeur de threshold
- Après le téléchargement du code sur la carte ESP32 et on s'assure que la bonne carte et le bon port COM sont sélectionnés.

Pour le test, nous pourrions se contenter de toucher directement les broches de la carte ESP32 mais pour une meilleure ergonomie nous avons connecté un fil conducteur à la broche tactile GPIO4. Lorsqu'on ne touche pas le fil, touchRead() retourne une valeur de 60 à 70. Lorsqu'on touche du doigt le fil conducteur, la valeur retournée par touchRead() diminue pour atteindre une valeur inférieur à 20. Les résultats obtenus sont illustrés sur la figure 3.18**.**

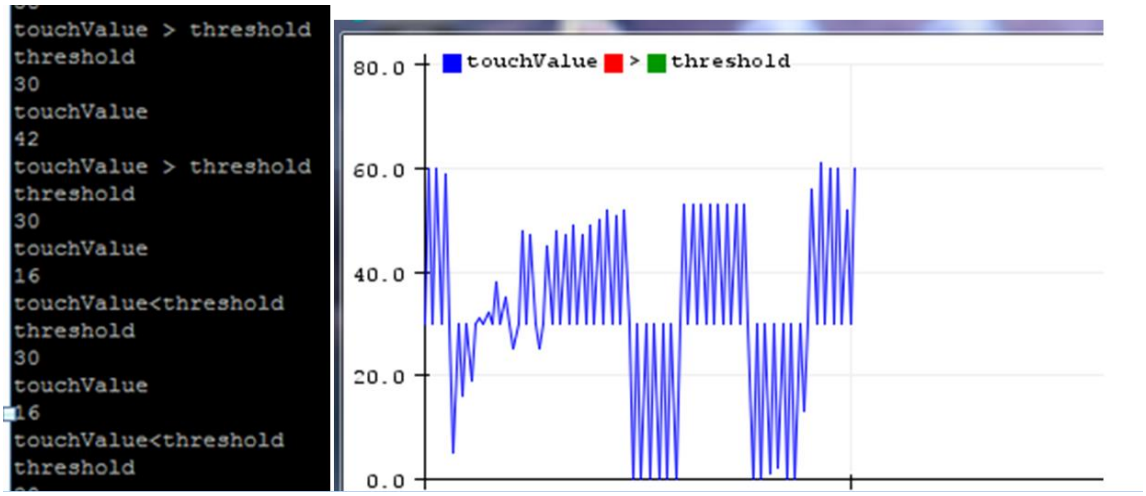

Figure 3.18 : Signal analogique d'un capteur tactile capacitif.

2.12 Techniques d'analyse du signal appliqué pour la détection des défauts dans les machines

2.12.3 Analyse temporelle

Les méthodes temporelles sont basées sur l'analyse statistique du signal recueilli, elles s'appliquent à des machines simples et consistent à effectuer des mesures de vitesse dans des gammes de fréquences faibles et des mesures d'accélération dans des gammes de fréquences élevées. Le but est de minimiser l'influence des vibrations induites par la rotation des arbres. Cette méthode utilise des indicateurs scalaires qui permettent de suivre l'évolution d'une grandeur décrivant de la puissance ou de l'amplitude crête du signal. Sa valeur peut ne pas avoir de signification intrinsèque, c'est son évolution dans le temps qui est significative du défaut [3].

Une multitude d'indicateurs existent, plus ou moins performants et adéquats pour le dépistage de certaines défaillances plus que d'autres. Les indicateurs les plus utilisés sont décrits comme suit.

- **Le kurtosis :**

Plus spécifique au dépistage des défauts de roulements, le kurtosis est une grandeur statistique permettant d'analyser le caractère « pointu » ou « plat » d'une distribution, et donc d'observer la forme du signal. Dérivé du moment statistique d'ordre quatre, il est définit comme le rapport de la valeur moyenne du signal élevée à la puissance 4 sur le carré de son énergie. Il est donné par la formule suivante [47] :

$$Kurtosis = \frac{\frac{1}{N}\sum_{n=1}^{N_e}\left(x(n)-\bar{s}\right)^4}{\left[\frac{1}{N}\sum_{n=1}^{N_e}\left(s(n)-\bar{s}\right)^2\right]^2} \qquad \textbf{3.6}$$

- $x(n)$ est le signal temporel
- $\bar{x}$ est la valeur moyenne
- N_e est le nombre d'échantillons prélevés dans le signal

Le Kurtosis approche la valeur de 3 pour un fonctionnement sans défauts de roulement, et augmente de façon remarquable dès l'apparition d'impulsions dues à la naissance d'un défaut. Le Kurtosis tend à revenir à 3 dès que la dégradation entre en phase terminale

- **La valeur crête : VC**

Représente la valeur maximale du signal. Pour un signal x (n) la valeur crête est donné par :

$$Valeur \quad crête \quad = Sup|x(n)| \qquad \textbf{3.7}$$

Est un indicateur qui caractérise l'amplitude maximale des chocs. Il se manifeste dès l'apparition de la première écaillure et donne une information très précoce de la prédiction. Malheureusement, c'est un mauvais indicateur une fois que la dégradation s'accentue.

- **La valeur efficace ou valeur RMS (Root Mean Square)** :

C'est une valeur très caractéristique du signal, vu qu'elle a une relation directe avec l'énergie contenue dans celui-ci:

$$RMS = \sqrt{\frac{1}{N}\sum_{n=1}^{N_e}\left[(x(t)\right]^2} \qquad \textbf{3.8}$$

Ce paramètre est le plus couramment utilisé en analyse vibratoire, mais il est beaucoup imprécis dans le cas des roulements, car il ne varie de façon significatif que lorsque le défaut devient très sévères mais il encore utilisable et très efficace mais pas tous seul.

- **Facteur de crête** :

C'est le rapport entre la valeur de crête et la valeur efficace (RMS).

$$F_c = \frac{valeur \quad crête}{RMS} = \frac{Sup|x(t)|}{\sqrt{\frac{1}{N}\sum_{n=1}^{N_e}\left[x(t)\right]^2}} \qquad \textbf{3.9}$$

Il faut rappeler qu'une vibration de type sinusoïdale aura un facteur de crête voisin de 2, alors qu'une vibration de type impulsionnel aura un facteur de crête plus important. Lorsqu'il n'y a pas de défaut le F_c reste proche de trois (3) et il faut rappeler que l'apparition d'un défaut entraine l'augmentation du facteur de crête.

2.12.4Méthode fréquentiel :

Le second type de représentation est la représentation du domaine de fréquence (fréquence, amplitude) appelé spectre ou représentation spectrale (Analyse fréquentiel). Le signal complexe ***F(t)*** qui est difficile à interpréter, est décomposé en une série de composants élémentaires définis par leurs fréquences et leurs amplitudes.

L'outil mathématique utilisé dans ce cas est la décomposition du signal à l'aide de la transformée de Fourier. Si cette décomposition est possible, sa représentation dans le domaine temporel est encore inutilisable. Il consiste à représenter dans un diagramme appelé spectre la fréquence et l'amplitude. Avec ce type de représentation, chaque composante sinusoïdale est définie par son amplitude et sa fréquence. La représentation spectrale devient plus nette et réalisable. Le spectre final contient toutes les fréquences sinusoïdales (lignes discrètes) formant le signal de vibration d'origine. A noter que le spectre d'un choc périodique comporte un peigne de lignes à la fréquence de choc [48].

2.13Transformation de Fourier

2.13.3La décomposition en série de Fourier de signaux périodiques

Tout signal de période $T_0 = 1/f_0$ peut se décomposer en une somme de fonctions sinusoïdales de fréquences $f_N = nf_0$ multiples de la fréquence fondamentale [49].

Soit :

$$x(t) = a_0 + \sum_{n=1}^{+\infty}(a_n \cos(2\pi n f_0 t) + b_n \sin(2\pi n f_0 t)) \qquad \textbf{3.10}$$

a_n et $b_n$sont les coefficients de la série de Fourier. a_0 est appelé valeur moyenne ou composante continue du signal. Ils sont déterminés à partir des relations suivantes :

$$a_0 = \frac{1}{T_0}\int_0^{T_0} x(t)\,dt$$

$$a_n = \frac{2}{T_0}\int_0^{T_0} x(t)\cos(2\pi f_0 nt)\,dt \qquad \textbf{3.11}$$

$$b_n = \frac{2}{T_0}\int_0^{T_0} x(t)\sin(2\pi f_0 nt)dt$$

L'expression précédente peut également s'écrire sous la forme d'un développement en harmoniques :

$$x(t) = a_0 \sum_{n=1}^{+\infty} c_n \cos(2\pi n f_0 t + \emptyset_n) \qquad \textbf{3.12}$$

$$c_n = \sqrt{a_n^2 + b_n^2}$$

$$\emptyset_n = \arctan\left(-\frac{b_n}{a_n}\right)$$

Le spectre en fréquence du signal représente l'amplitude du fondamental a_0 pour $f = f_0$ ainsi que les différentes harmoniques c_n pour$f = nf_0$.

Le spectre d'une fonction périodique est discontinu et composé de raies dont l'écart minimum sur l'axe des fréquences est f_0.

La décomposition en série de Fourier peut aussi s'écrire en utilisant la notation complexe. On introduit alors des valeurs de **n** négatives dans un but de simplification, étant donné que le signal **x(t)** est réel, nous avons $a_{-n} = a_n$ et $b_{-n} = b_n$

$$\hat{x} = \sum_{n=-\infty}^{n=+\infty} s_n e^{j2\pi n f_0 t} \qquad \textbf{3.13}$$

Avec $s_n = \frac{1}{2}(a_n - jb_n) = \frac{1}{T_0}\int_0^{T_0} x(t)e^{-j2\pi n f_0 t}$

Les coefficients S_n sont généralement complexes, on préfèrera représenter son module $|s_n| = \frac{c_n}{2}$ et sa phase $\emptyset_n = \arctan(-\frac{b_n}{a_n})$

Le spectre d'une fonction périodique est alors représenté par une suite de raies d'amplitude

$$s_n = |s_n| e^{-j\emptyset_n}$$

Pour $f = nf_0$ On peut donc l'écrire sous la forme :

$$s(f) = \sum_{-\infty}^{+\infty} s_n \delta\,(f - nf_0) \qquad \mathbf{3.14}$$

Le spectre est formé par une suite d'impulsions Dirac de poids s_n réparties sur l'axe des fréquences négatives et positives. Le poids étant a priori complexe, le spectre devrait être représenté par sa partie réelle et sa partie imaginaire ou par son module et sa phase.

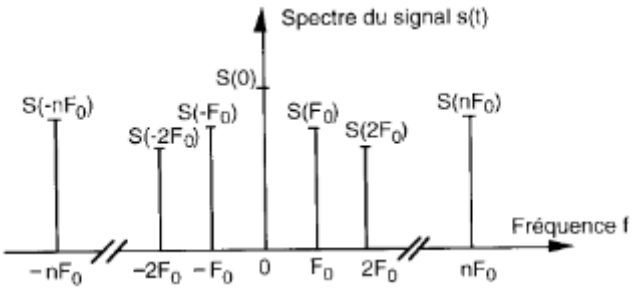

Figure 3.19 : Spectre en fréquence d'un sinal périodique avec un axe fréquences de $-\infty$ a$+\infty$.

2.13.4Transformée de Fourier des signaux non-périodiques

Soit *x(t)* un signal quelconque, on note *x(f)* ou TF(x(t)) sa transformée de Fourier telle que :

$$x(f) = TF\big(x(t)\big) = \int_{-\infty}^{+\infty} x(t) e^{-i2\pi ft} dt \qquad \mathbf{3.15}$$

Inversement, on peut définir une transformée de Fourier inverse $\boldsymbol{TF^{-1}}$ telle que :

$$\boldsymbol{x(t) = TF^{-1}\big(x(f)\big) = \int_{-\infty}^{+\infty} x(f) e^{-i2\pi ft} df} \qquad \mathbf{3.16}$$

x(f) est une fonction complexe même si $x(t)$ est réel. La transformée de Fourier contient donc une partie réelle et une partie imaginaire et est représentée facilement grâce à son module et à son argument : $|x(f)|$ est appelé spectre d'amplitude et $\arg(x(f))$ le spectre de phase du signal. La variable f s'appelle la fréquence dont l'unité est le Hertz (en abrégé : *Hz*).

- **Remarques importantes :**

La représentation complète d'une transformée de Fourier nécessite 2 graphiques : le module et la phase, ou bien la partie réelle et le partie imaginaire.

Pour représenter les transformées de Fourier de signaux, l'échelle logarithmique est généralement utilisé.

Pour un signal acoustique, par exemple, on calcule $20\log(\frac{|X(f)|}{2.10^{-5}})$ et arg (*x(f)*).

Ainsi, la transformée de Fourier est un opérateur mathématique qui permet d'analyser et de représenter un signal dans le domaine fréquentiel. La TF ne modifie pas le signal mais permet seulement de l'observer selon différents points de vue (temporel ou fréquentiel). Il est important de retenir que *x(t)* et *x(f)* sont deux descriptions équivalentes du même signal. Ces deux fonctions contiennent la même information il s'agit juste de deux descriptions dans des domaines différents.

x(f) apporte des informations sur le système physique à l'origine du signal. Elle permet par exemple de différentier un son de trompette d'un son trombone, ou bien encore différentes ondes cérébrales, plus facilement qu'en observant le signal dans le domaine temporel. Le contenu spectral d'un signal est en effet assimilable à sa « carte d'identité ».

2.13.5Transformée de Fourier d'un signal échantillonné

Nous avons vu que la plupart du temps, les signaux étaient échantillonnés à la fois en temps et en amplitude. Dans le cas des signaux échantillonnés où le temps est discrétisé, il n'est plus nécessaire d'utiliser des intégrales continues pour sommer les valeurs de *x(t)* sur tout l'axe des temps, puisqu'un signal échantillonné peut être assimilé à une suite contenant un nombre fini d'éléments.

2.13.6Transformée de Fourier à temps discret (TFTD)

La transformée de Fourier discrète d'un signal échantillonné $x_e(t)$ de période d'échantillonnage T_e est donnée par :

$$x(f) = \sum_{n=-\infty}^{+\infty} x(nT_e)e^{-2j\pi nT_e f} \qquad \textbf{3.17}$$

2.14Traitement des donné par analyse FFT

Les techniques classiques de traitement du signal ont été discutées dans les sections précédentes. Vu que la TF ne modifie pas le signal mais permet seulement de l'observer selon différents points de vue (temporel ou fréquentiel), une étude plus approfondie sur la façon dont les données critiques ont été obtenues à partir d'un signal est effectuée en utilisant la transformée de

Fourier rapide (FFT) qui est un algorithme qui calcule transformé » de Fourier discrète ou de son inverse. Pour cela, nous proposons dans cette section la programmation d'un algorithme de la FFT.

Le signal reçu de la machine sera traité par l'algorithme de la FFT afin de convertir les données du domaine temporel au domaine fréquentiel et extraire la totalité de composantes spectrales. Après, l'acheminement de données vers RCSF sera effectué via le microcontrôleur MSP430. Pour cela, le codage de l'algorithme de la FFT est effectué en langage C dans l'environnement de développement IAR. Les résultats obtenus de l'algorithme seront stockés dans un vecteur et seront communiqués au microcontrôleur ESP32 via le protocole de communication UART pour être transmis dans le réseau.

2.14.3IAR Embedded Workbench

IAR Embedded Workbench est un environnement de développement intégré (IDE) très puissant qui permet de développer et de gérer des projets d'application intégrés complets. Il favorise une méthodologie de travail utile, et donc une réduction significative du temps de développement [50].

Lors de l'installation du produit IAR, vous trouverez un ensemble d'outils, d'exemples de code et de documentation utilisateur, tous adaptés au développement de logiciels pour les applications embarquées basées sur MSP430. Les outils vous permettent de développer votre application en C, C ++ ou en langage assembleur.

Quelques étapes pour créer un programme en langage C pour le µ-contrôleur de la famille MSP430G sous l'environnement IAR Embedded Workbench :

1. Lancer le logiciel IAR Embedded Workbench
2. Créer un nouveau Projet Project > Create a new Project : une fenêtre apparait (Figure 3.20)
3. Choisir un template C / main puis appuyer sur OK 4

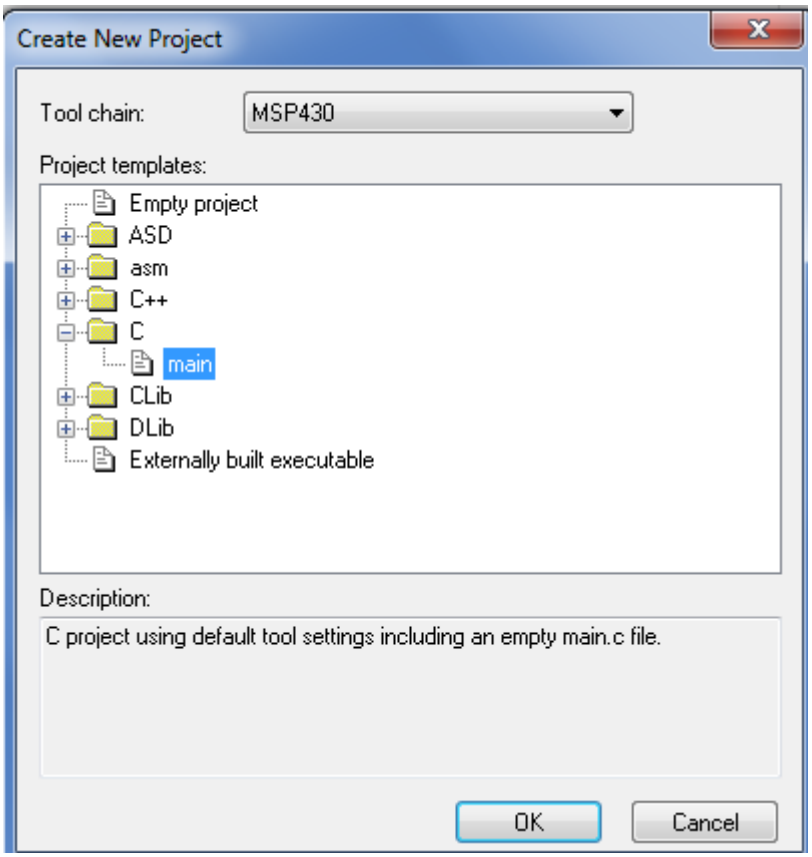

Figure 3.20 : Fenêtre IAR pour la création d'un nouveau projet

4. A ce niveau il convient de choisir (ou créer) un répertoire de travail.
5. L'environnement de travail apparaît alors dans lequel le programme main ne comporte que quelques lignes :

```
#include "io430.h"

int main( void )
{
  // Stop watchdog timer to prevent time out reset
  WDTCTL = WDTPW + WDTHOLD;

  return 0;
}
```

Figure 3.21 : Environnement de travail IAR.

6. Dans la barre de menu Project choisir Options. La fenêtre suivante apparaît :

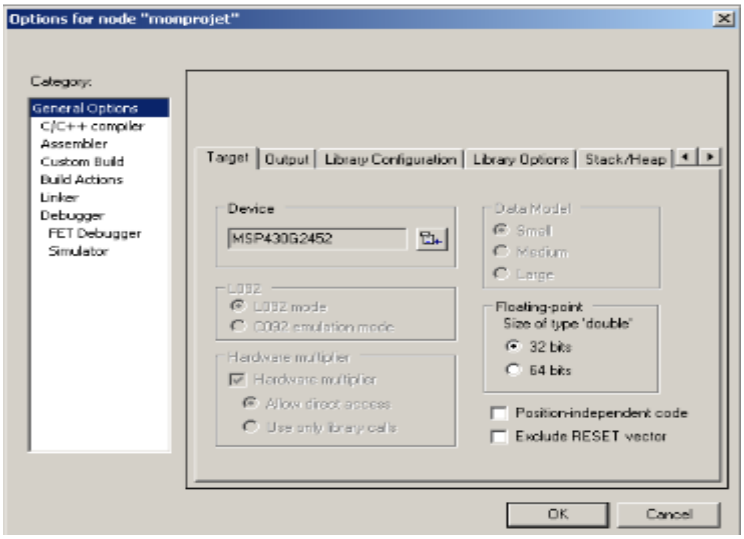

Figure 3.22 : Choix des options du projet.

7. Dans la partie General Conditions, choisir la référence de la cible : MSP430Gxxx sur la figure ci-contre.

8. Dans la partie Debugger, choisir le Driver : FET Debugger puis dans FET Debugger choisir Texas Instrument USB-IF pour les kits de développement EZ430 ou Launchpad.

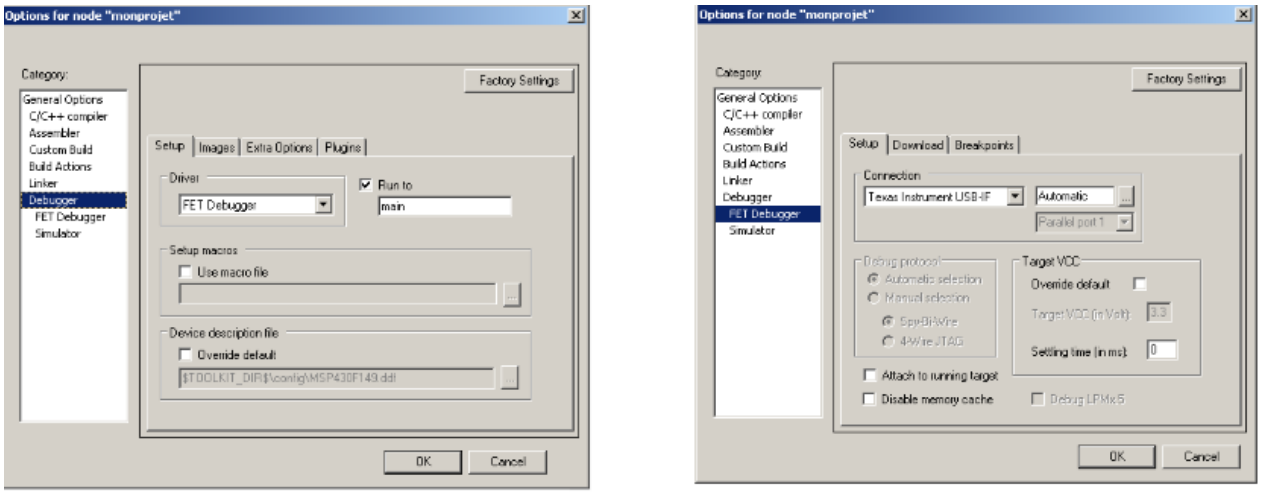

Figure 3.23 : Choix de la référence de la cible MSP430G2553 et le Driver.

2.14.4MSP430G2553

C'est un système informatique contenu dans un seul circuit intégré, il est disponible dans plusieurs boitiers, dont le DIL20 (dual in line 20 pins) facile à mettre en œuvre.

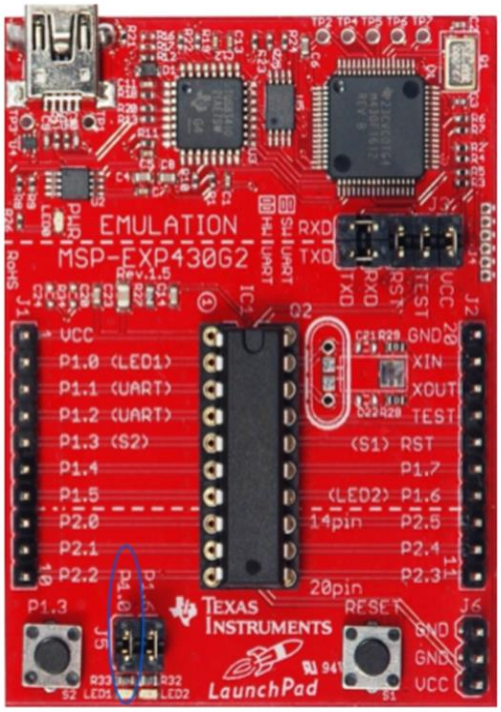

Figure 3.24 : Carte de développement MSP430**.**

Les principales caractéristiques de MSP430 sont :

- Gamme de basse-tension d'alimentation : 1,8 V à 3,6 V.
- Ultra-fable consommation d'énergie :

-Mode actif : 230 µA à 1 MHz et 2,2 V.

-Mode veille : 0,5 µA.

-Mode "arrêt" : 0,1 µA.

- Architecture 16 bits RISC.
- Fréquences internes jusqu'à 16 MHz avec quatre fréquences calibrées.
- Oscillateur interne en Crystal puissance très-faible et à fréquence basse 32 kHz.
- Source d'horloge numérique externe.

2.14.4.1Brochages de msp430G2553

Nous pouvons diviser les broches de msp430 en trois groupes : les alimentations, les broches de programmation et les broches d'entrée-sortie :

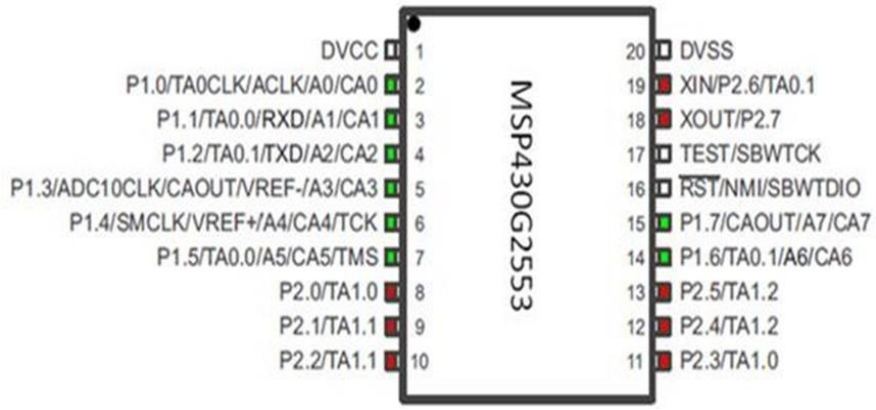

Figure 3.25 : Brochages de MSP430G2553.

A. Les alimentations :

Les MSP430 acceptent une tension comprise entre 1,8 et 3,6 V.

La borne négative est Gnd (Ground : masse) ⇒ broche 20

La borne positive est DVCC⇒ broche 1

B. Les broches de programmation :

La programmation des MSP430 peut se faire par l'intermédiaire de deux signaux : le **RST** est câblée sur la broche 16 et un signal nommé **TEST** sur la broche 17.

C. Les broches d'entrée-sortie :

C'est-à-dire toutes les autres broches sauf les broches d'alimentations et de programmation Elles sont regroupées en 2 ports : P1 et P2. Ces deux ports sont completes, ils comportment chacun 8 broches.

2.14.4.2Composants intégrés de msp430G2553

- un processeur (CPU) de16 bits, des instructions cadencées entre 16KHz et 32 KHz avec 16 registres de 16 bits.
- la mémoire vive (RAM) pour stocker les données et variables de 512 octet.
- Une mémoire morte de type flash de 16 Kbits. Cette mémoire peut être effacée et réécrite à nouveau plusieurs fois.
- capables d'effectuer des tâches spécifiques. On peut mentionner entre autres :

02 timers à 16 bits, d'un UART, d'un I2C, d'un SPI, d'un ADC (convertisseur analogique numérique) à 10 bits et 8 canaux, certaines pattes ont également d'autres fonctions spécifiques (oscillateur à quartz, etc…).

2.14.5Protocole de communication UART [51]

L'UART (Universal Asynchronous Receiver/Transmitter) est un protocole de communication série intégré dans la pluparts des microcontrôleurs modernes, il permet la communication entre deux microprocesseurs. Chaque UART dispose d'une entrée RX et d'une sortie TX, le transmetteur envoie une trame de données bit par bit au récepteur, cette trame est composée de :

- un bit de début toujours à 0 (Low level).
- un octet de données (08 bits).
- un bit de parité (paire ou impaire).
- un ou deux bits d'arrêt «stop» toujours à 1 (High level).

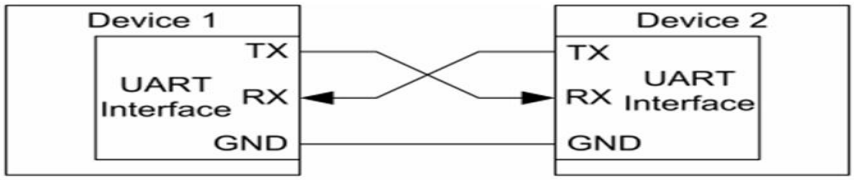

Figure 3.26 : La communication entre deux equipments.

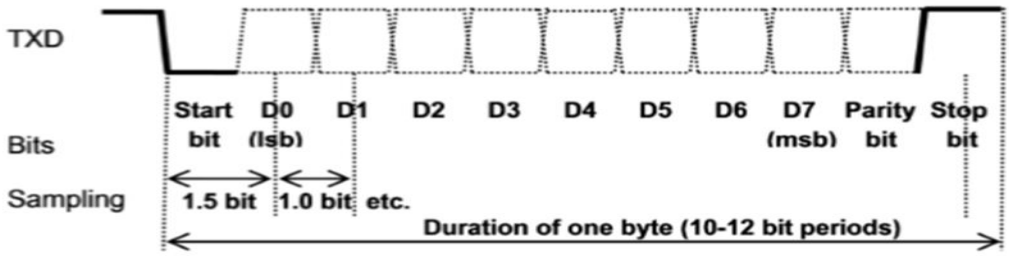

Figure 3.27 : Format de tram de données.

La transmission des données est faite d'une façon asynchrone, cela signifie qu'il n'y a pas de signal d'horloge (CLK) entre l'émetteur et le récepteur, d'où Le récepteur se base sur la durée de ses bits qui doit être fixé et connue de l'émetteur et du récepteur pour identifier les différents bits d'un octet. La durée d'un bit est définie par le BAUDRATE, nombre de bits par secondes.

Le BAUDRATE peut avoir les valeurs suivantes : **300 bps, 1200 bps, 2400 bps, 4800 bps, 9600 bps, 19200 bps, 38400 bps**, des vitesses rapides sont parfois aussi possibles.

- **L'UART de MSP430G2553 :**

Dans MSP430, la communication en série est assurée par une puce sur périphérique appelé USCI (Universal Serial COMMUNICATIONS Interface).

MSP430G2553 a deux noms de modules USCI comme :

- USCI_A0 peut être configuré pour gérer LIN, IrDA, SPI et la communication série asynchrone (UART)
- USCI_B0 peut gérer SPI et I2C.

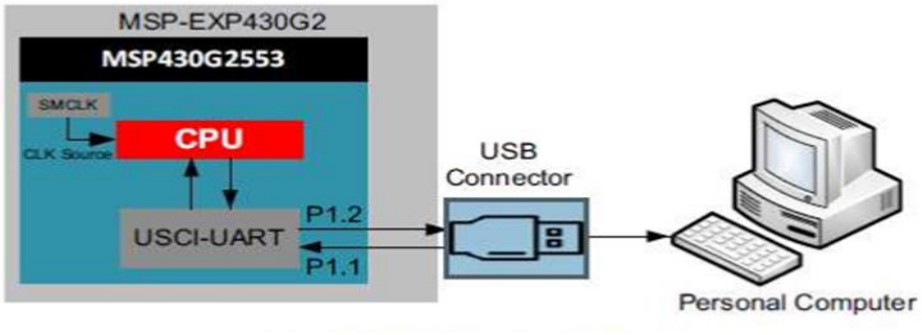

Figure 3.28 : Communication entre MSP430G2553 et PC.

2.14.6Programmation l'algorithme transformation de Fourier discrète

Certes il existe de nombreux algorithmes "rusés" permettant le calcul de la transformée de Fourier discrète d'un signal discret de longueur finie mais nous nous somme intéressé qu'un, appelé "Algorithme de la transformation de Fourier rapide" (TFR, ou FFT en anglais). Nous présentons ci-dessous notre programme de l'algorithme FFT en langage C dans l'environnement logiciel IAR Embedded.

- **Le passage domaine temporelle a domaine fréquentielle :**

```
# include <stdio.h>
# include <math.h>
# include <complex.h>

double PI;
double   module0 ;
double   module;
double module1;
double somme;
double c1real;
double sqrt(double X);
int i;
typedef double complex cplx;

void _fft(cplx buf[], cplx out[], int n, int step)
```

```
{
if (step < n) {
_fft(out, buf, n, step * 2);
_fft(out + step, buf + step, n, step * 2);

for (int i = 0; i < n; i += 2 * step) {
cplx t = cexp(-I * PI * i / n) * out[i + step];
buf[i / 2]     = out[i] + t;
buf[(i + n)/2] = out[i] - t;
}
}  }

void fft(cplx buf[], int n)
{
cplx out[8];
for (int i = 0; i < n; i++) out[i] = buf[i];
_fft(buf, out, n, 1);
}
void amplitude(const char * s, cplx buf[])
{
printf("%s", s);
for (int i = 0; i < 8; i++)
if (!cimag(buf[i]))
{
printf("%g ", creal(buf[i]));
}
else
{
module0 = creal(buf[i])*creal(buf[i]);
module1= cimag(buf[i])*cimag(buf[i]);
somme =( module0+module1 );
module = sqrt( somme);
printf("%g ", module );
}
}

void show(const char * s, cplx buf[]){
printf("%s", s);
for (int i = 0; i < 8; i++)
if (!cimag(buf[i]))// Boolean negation (true <--> false)
printf("%g ", creal(buf[i]));
else
printf("(%g, %g) ", creal(buf[i]), cimag(buf[i]));
}

int main()
{
PI = atan2(1, 1) * 4; // PI=3.14
```

```
cplx buf[] = {1,4,3,2,0,6,7,1}; show("Data: ", buf); //
fft(buf,8);
show("\nFFT : ", buf);
amplitude("\nAmplitude : ", buf );
return 0;
}
```

Discussion et résultats :

Le but de code FFT est de convertir un certain nombre d'échantillons d'un signal temporel en valeurs de fréquence puis de calculer des amplitudes fréquentielle selon les étapes suivantes :

1. Nous incluons Les bibliothèques nécessaires à l'écriture de programme
2. Nous allons définir la constante **PI** et déclarer les variables nécessaires
3. Ensuite nous déclarons les 4 fonctions principales qui permettent de calculer les valeurs fréquentielle et leur amplitude puis la fonction show qui permet afficher les résultats
    - **void fft(cplx buf[], int n)**
    - **void _fft(cplx buf[], cplx out[], int n, int step)**
    - **void amplitude(const char * s, cplx buf[])**
    - **void show(const char * s, cplx buf[])**
    - la fonction fft a besoins la fonction _fft pour faire les calculs requis, ce pour cette raison que les deux fonctions _fft et fft ont besoins deux tableaux d'entrée et de sortie de même taille
4. Dans le Programme principale **int main()** :
    - Nous calculons d'abord la constante de PI et remplissons la tableau d'entrée **cplx buf []** avec les 8 valeurs complexes échantillonnées d'un signal discret **(fn)[0,N-1].**
    - Nous appelons la fonction **fft** pour calculer les valeurs de fréquence et les stocker toujours dans le **buf [],**.
    - Nous affichons les résultats dans le terminal IO via la fonction **show** (les valeur réel et imaginaire qui sont stokés dans **buf[]**).

- Enfin, nous appelons la fonction **amplitude** pour calculer et afficher les amplitudes fréquentielles,
- Les résultats obtenus sont illustrés sur la Figure 3.29.

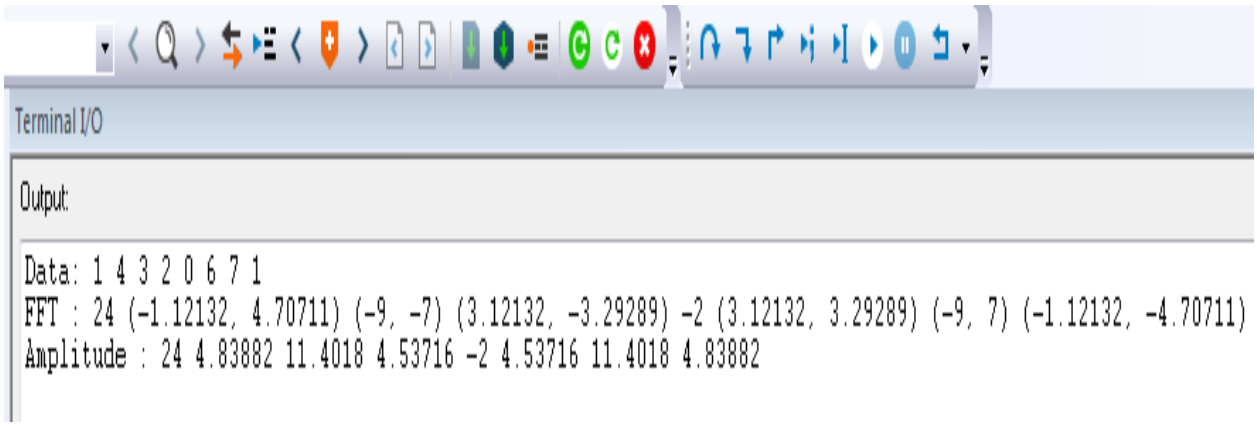

Figure 3.29 : Résultats d'affichage des amplitudes fréquentielles.

➢ **Le Stockage des amplitudes fréquentielle dans un nouveau vecteur :**

Afin de transmettre les résultats des amplitudes fréquentielle vers le RCSF, ceux-ci doivent être stoker dans un vecteur qui sera introduit dans le main précèdent. Pour cela nous créons un vecteur nommé **ACC** de taille 8 comme illustré sur le code suivant :

```
int main()
{
PI = atan2(1, 1) * 4; // PI=3.14
int n=8;
cplx buf[] = {1,4,3,2,0,6,7,1}; show("\nData: ", buf);
fft(buf,8);
 show("\nFFT : ", buf);
amplitude("\nAmplitude : ", buf );
cplx ACC[8];
for (int i = 0; i < n; i++) ACC[i]=buf[i];
amplitude("\nACC : ", ACC );
return 0;}
```

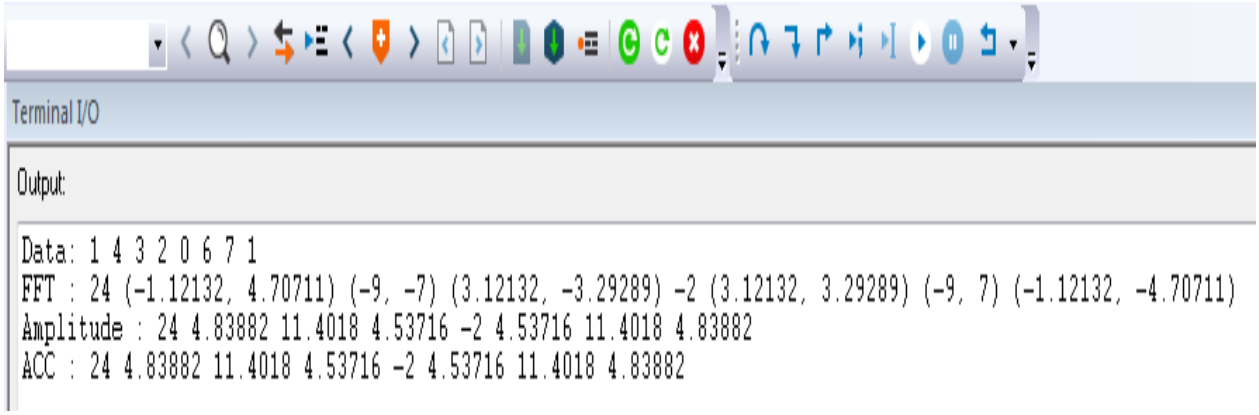

Figure 3.30 : Résultats d'affichage de vecteur **ACC**

2.15 Conclusion

Dans ce chapitre, nous avons discuté principalement les généralités de base et les définitions du traitement du signal, ainsi que les méthodes d'analyse du signal. Dans le but de faciliter l'extraction d'informations intéressantes d'un spectre fréquentielle, nous avons programmé l'algorithme de transformée de Fourier rapide (FFT) qui calcule la transformée de Fourier discrète. Les résultats obtenus seront utilisés pour le suivi et le diagnostic des défauts dans les machines tournantes comme sera discuté dans le chapitre suivant.

Chapitre 4

Partie simulation

4.1 Introduction

De nos jours, l'automatisation des lignes de production ou l'intervention de l'être humain est moins fréquente, ne cesse de croître. Cela requiers la mise en œuvre d'algorithmes de détection automatique des défauts et que les appareils soit capables de continuer à surveiller les états des processus pendant longtemps en mode autonome, en particulier quand les endroits où sont placés ces appareils sont difficilement accessibles. L'Iot (Internet des objets) semble offrir de grandes opportunités dans ce domaine. Grâce à la disponibilité croissante d'internet et l'augmentation de la vitesse de communication, la surveillance des différentes valeurs est devenue presque en temps réel.

Notre but dans ce chapitre, consiste à donner une solution pour identifier et prévoir les problèmes de performances d'une machine avant une panne. Pour cela, nous allons décrire comment acquérir, traiter et transmettre les données des capteurs sur un réseau sans fil

2.16 Description de la structure adoptée

La structure qui nous allons adopter dans notre travail est schématisé sur la figure 1. Elle comprend de deux blocs, le premier pour le traitement de signal et le deuxième pour la transmission de signal dans le réseau.

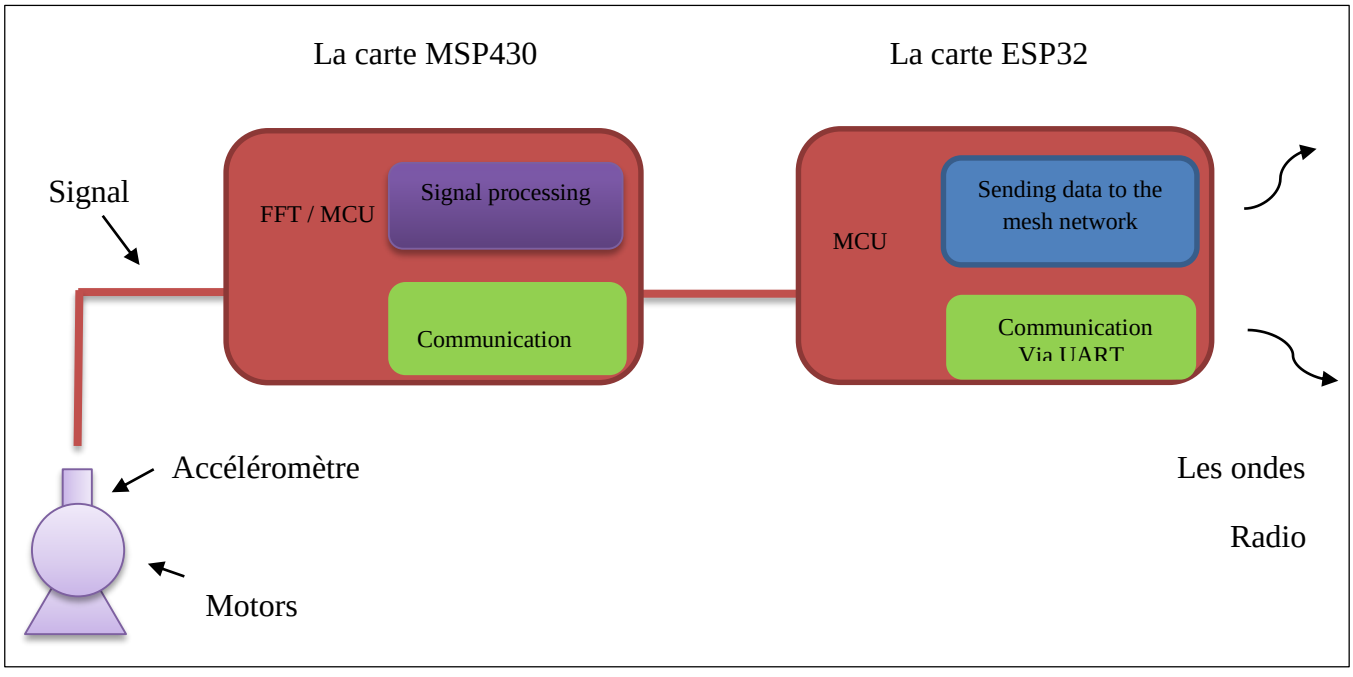

Figure 4.1 : Bloc de traitement du signal d'accéléromètre.

Le bloc du traitement du signal, représenté par la carte MSP430, reçoit les données d'un accéléromètre analogique via un convertisseur AD avec une certaine fréquence d'échantillonnage

et exécute l'algorithme de traitement de signal (FFT). La communication UART du MSP430 est responsable de l'envoi de données traitées à la module WiFi.

Dans le deuxième bloc constitué de la carte ESP32 (qui contient le module WiFi), le pavé communication via UART est chargé de recevoir tous les paquets de données traitées et de le transformer en un tampon (buffer), puis l'envoyer au module WiFi ESP32. Ce dernier est chargé de se connecter à point d'accès, pour transmettre des données et générer les différentes requêtes.

4.2.1 Traitement du signal

Pour surveiller l'état d'une machine, un accéléromètre est monté sur le moteur pour détecter les vibrations mécaniques. Les accéléromètres les plus courants utilisés pour ce type de diagnostic sont de type piézoélectrique, piézorésistif ou capacitif. Les données brutes peuvent être filtrées et jugées en fonction de l'interprétation du domaine fréquentiel au sein d'un microcontrôleur MSP430.

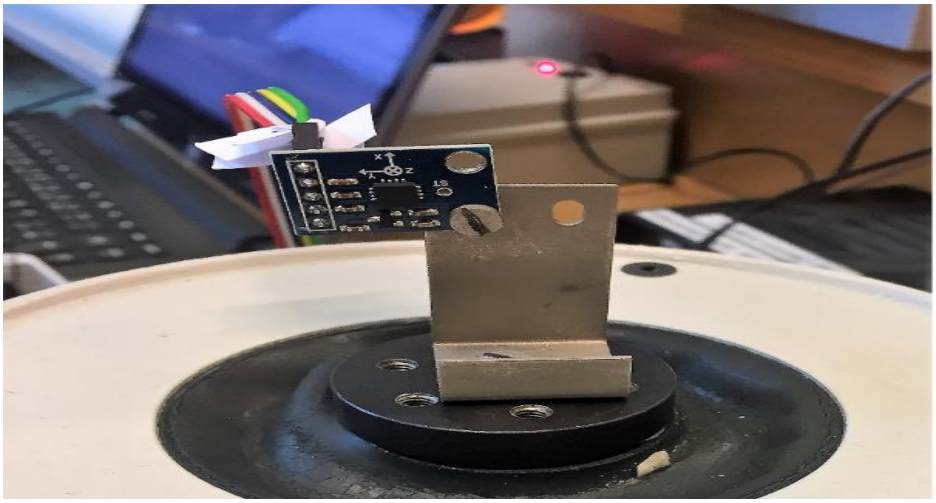

Figure 4.2 : placement de capteur sur le moteur.

Dans notre étude on s'intéresse seulement à l'analyse de mesures vibratoires enregistrées sur le moteur d'une machine tournantes. Dans la section d'outils de traitement du signal l'analyse « en fréquence » est l'outil fondamental pour le traitement des signaux vibratoire qui permet de détecté efficacement les défauts pouvant affecter le bon fonctionnement de ces machines. L'analyse fréquentiel s'appuie sur le transformé de Fourier. L'organigramme schématisant le processus de traitement de signal est présenté sur la figure suivante.

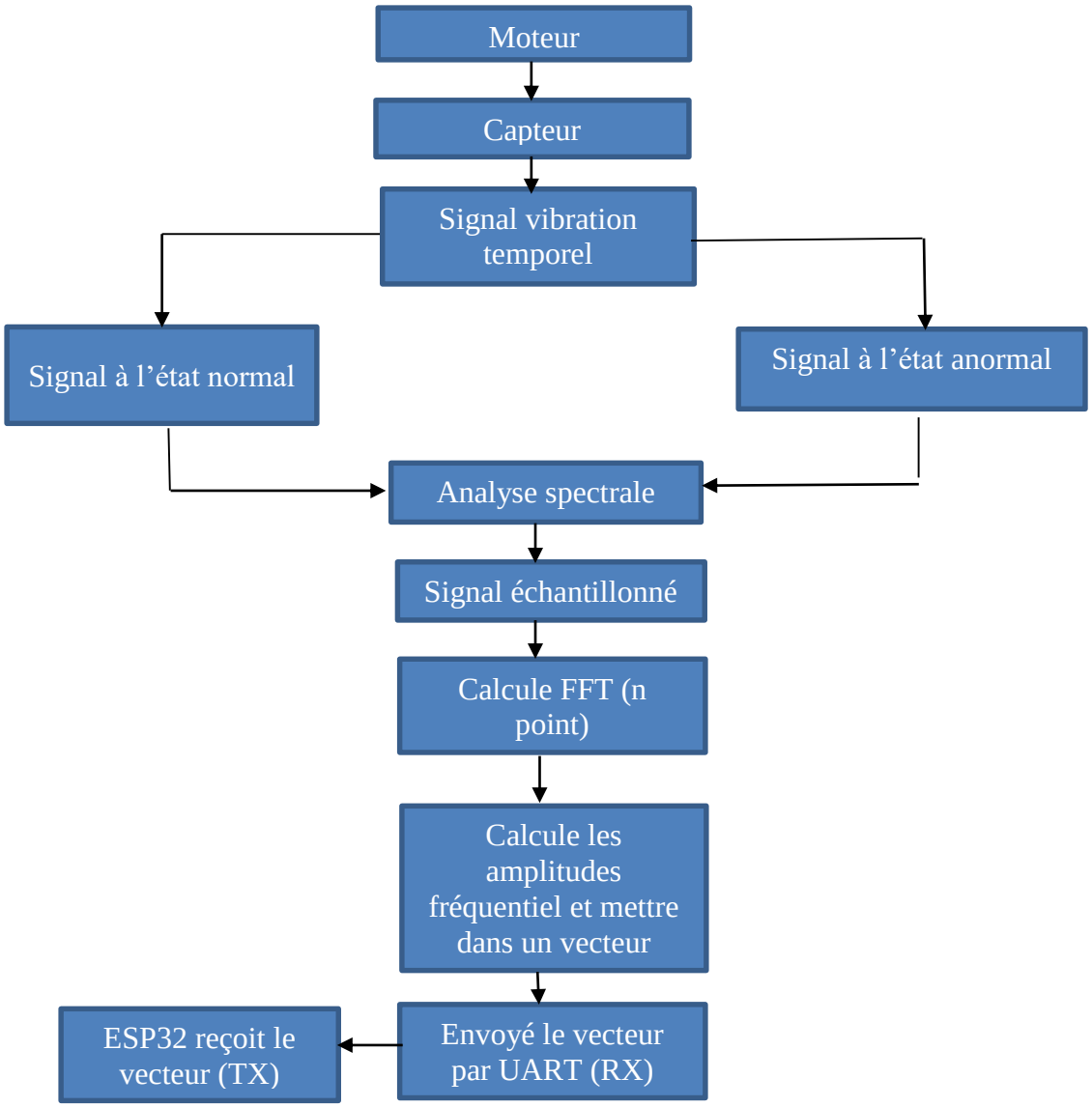

Figure 4.3 : Structure de traitement du signal vibratoire.

Pour une étude complète nous avons besoin de comparer les signaux enregistrés par l'accéléromètre à un signal de référence qui représente l'état sain (normale) de la machine afin de lancer les alertes d'avertissement en aval. Malheureusement, l'apparition de la pandémie du Covid 19 et la mise en place de confinement total à engendrer l'arrêt total de moyens de transport publique a constitué un obstacle pour l'obtention des données. Afin de surmonter ces difficultés nous avons utilisés des données bibliographiques pour notre travail. Pour tester notre programme de l'algorithme de traitement de signal (FFT), nous avons utilisé les données citées dans la référence [52]. Les figures suivantes représentent deux signaux temporaux d'une machine à l'état normal et à l'état défaillant

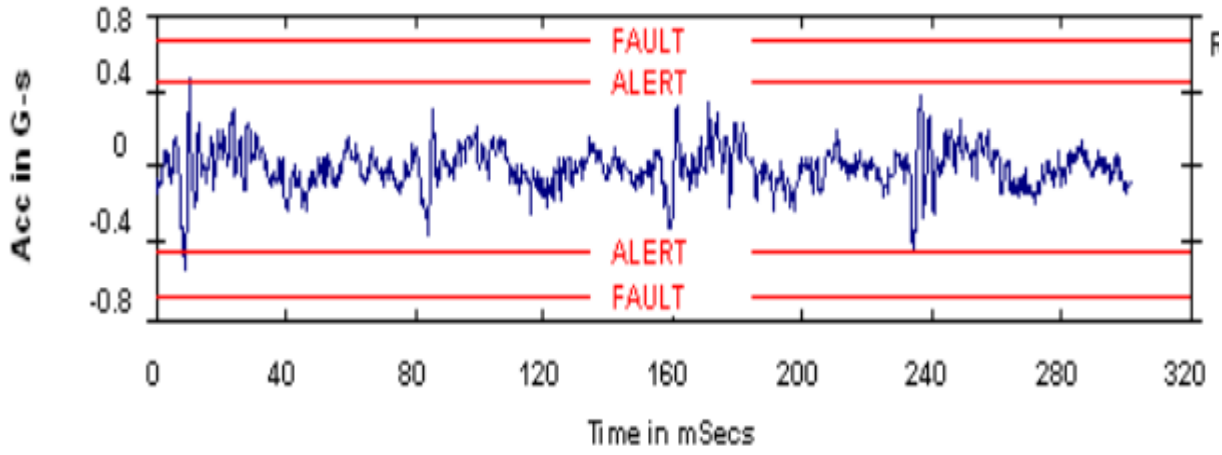

Figure 4.4 : Signal temporal à l'état normal [52].

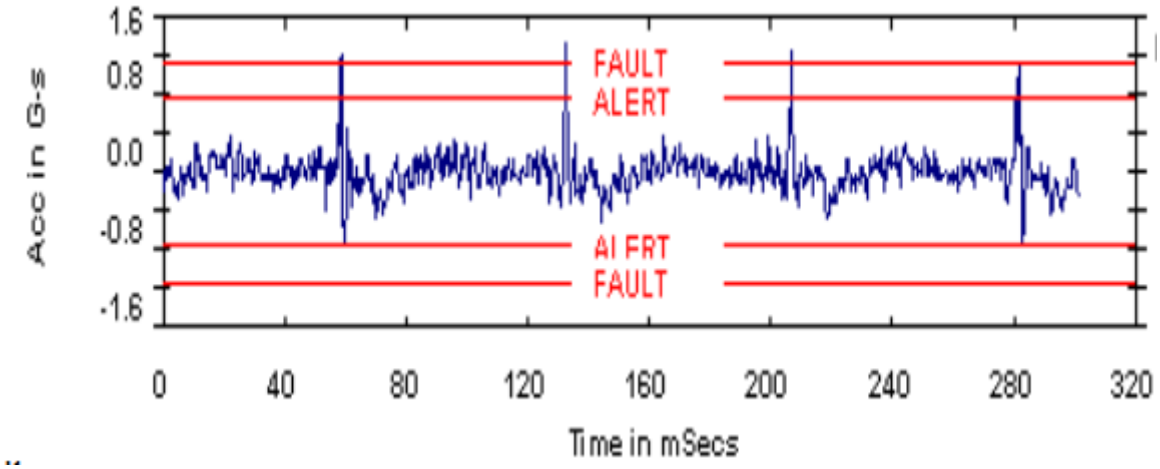

Figure 4.5 : Signal temporal à l'état défaillant [52].

L'échantillonnage des signaux est effectué sur un certaine nombre des échantillons dans les deux cas pendant la période d'échantillonnage Te (Te=0.02sec=20ms, fe=2fmax=50HZ) donne les résultats suivant :

```
cplx buf[] = {0. 3,0.1,0.2,0.3,0.2,0,-0.1,0.4};  // a  l'état sain
cplx buf[] = {0.5,0.1,1.2,-0.5,0.3,0.2,0.3,0.4}; // a l'état défaillant
```

4.2.2 Calcul FFT :

Dans le module MSP430, les échantillons sont traités à l'aide de l'algorithme FFT discuté dans le chapitre 3, le rôle de l'algorithme FFT est de :

- convertir le signal de l'accéléromètre du domaine temporel au domaine fréquentiel
- calculer leur amplitude fréquentielleles
- placer ces amplitudes dans le vecteur **ACC** pour la transmettre au module ESP32.

L'application de l'algorithme FFT pour les signaux de figures 4.4 et 4.5 donne les résultats illustrés sur les figures suivantes :

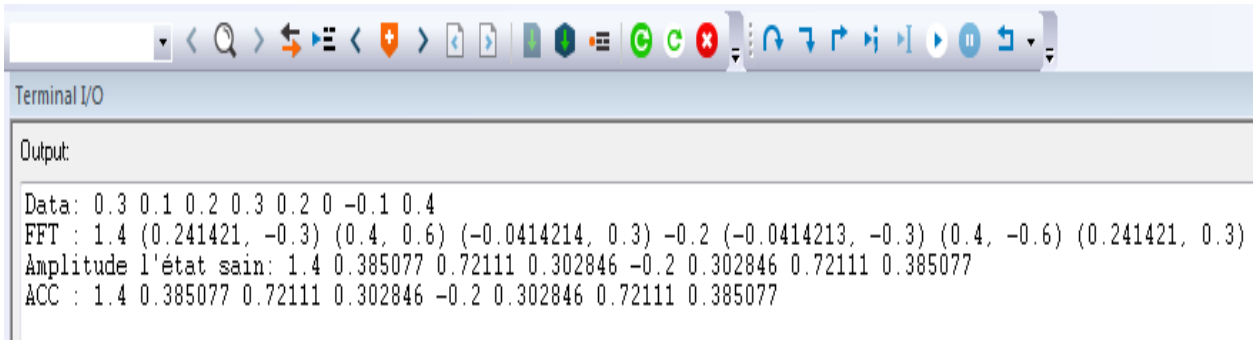

Figure 4.6 : le résultat de FFT à l'état sain.

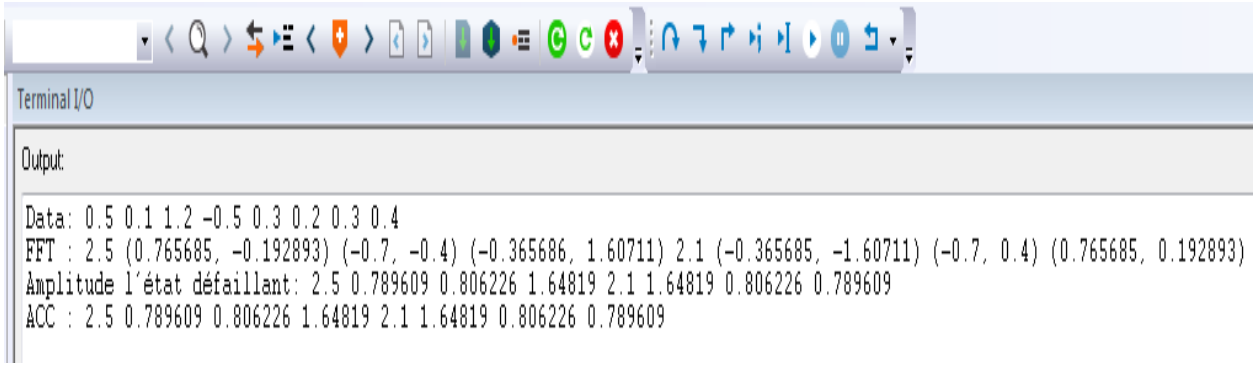

Figure 4.7 : le résultat de FFT à l'état défaillant.

4.2.3 Communication entre MSP430 et ESP32 :

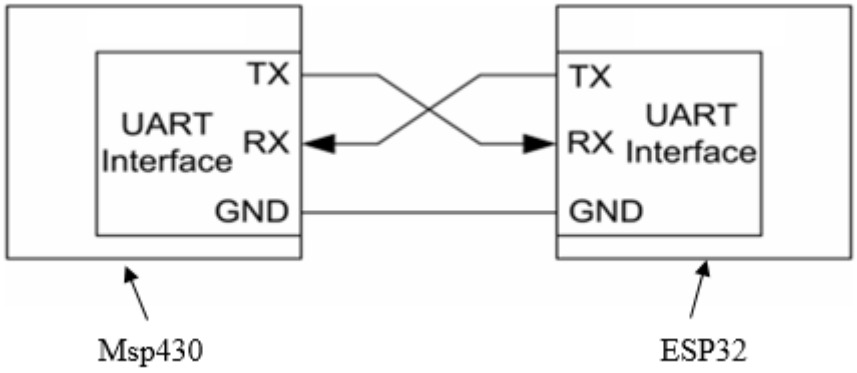

Figure 4.8 : communication entre MSP430 et ESP32.

L'envoi de vecteur **ACC** entre les unités MSP430 et ESP32, s'effectue via le protocole de communication UART à une vitesse de transmission UAR115200. Pour la quantité de données nécessaires au transfert, cette norme de communication série est suffisante.

4.2.4 La carte ESP32

Dans le module Wi-Fi ESP32, les données traitées sont reçues via le port UART puis sont transmises dans le réseau sans fil. Donc la carte esp32 est une interface entre le block de réseau maillé et le block traitement des données.

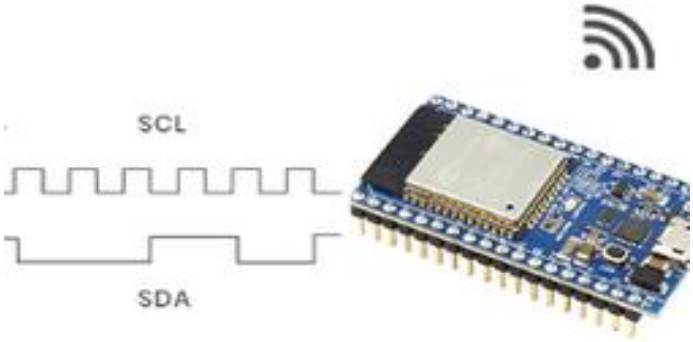

Figure 4.9 : interface entre le block de réseau maillé et le block traitement des données.

4.3 Transfert de données par le réseau maillé

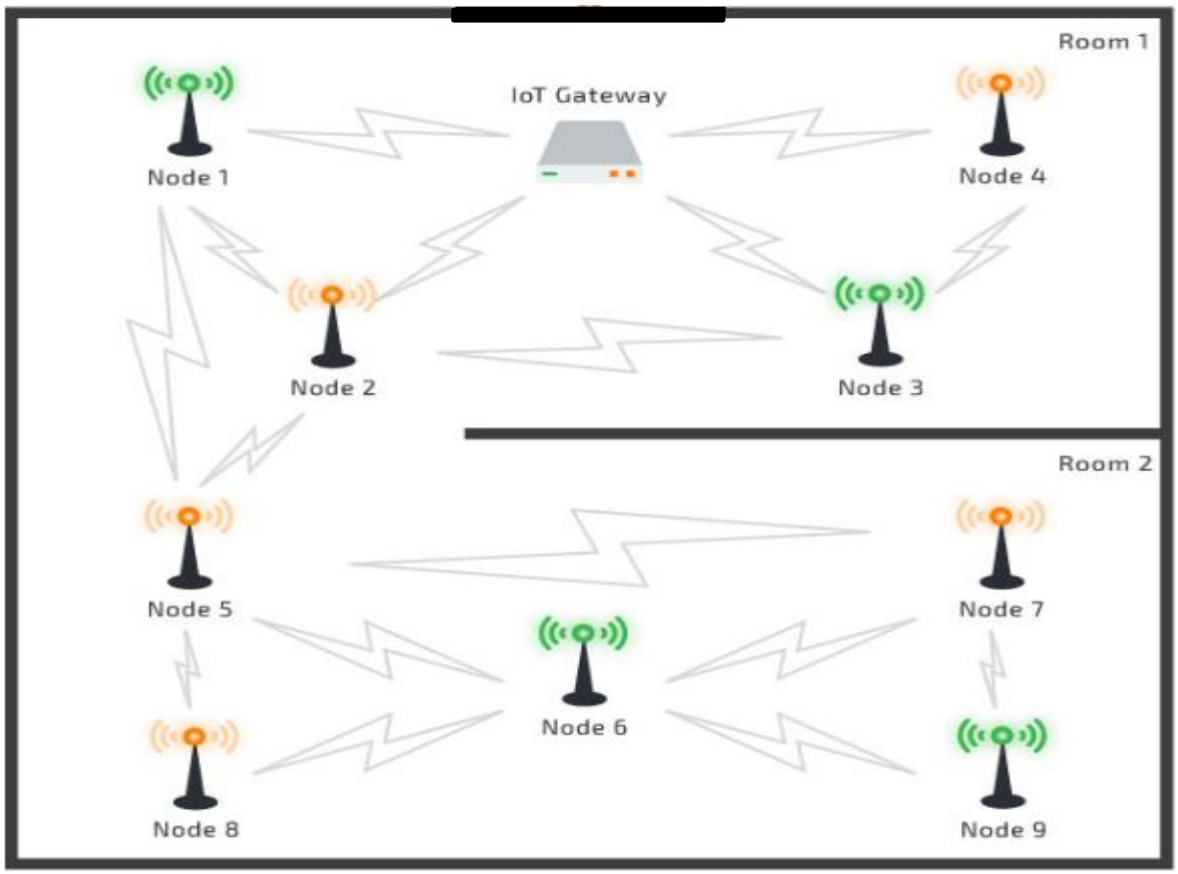

Figure 4.10 : le réseau maillé avec clusters.

Notre réseau maillé se compose de plusieurs nœuds comme indiqué sur la Figure 4.10, le nombre de nœuds dépend de besoin. Dans notre étude les nœuds sont divisés en trois catégories :

1. Le nœud feuille dans lequel se trouve l'accéléromètre est un nœud qui ne peut pas avoir de nœuds enfants. Par conséquent, il ne peut envoyer ou recevoir que ses propres paquets, mais il ne peut pas envoyer de paquets à partir d'autres nœuds. Comme il est situé sur la couche réseau maximale autorisée, il sera désigné comme nœud terminal.

Dans ces nœuds les données sont stockées dans une file d'attente, puis placées dans des objets JSON et transmises sur le réseau maillé (Fig 4.11).

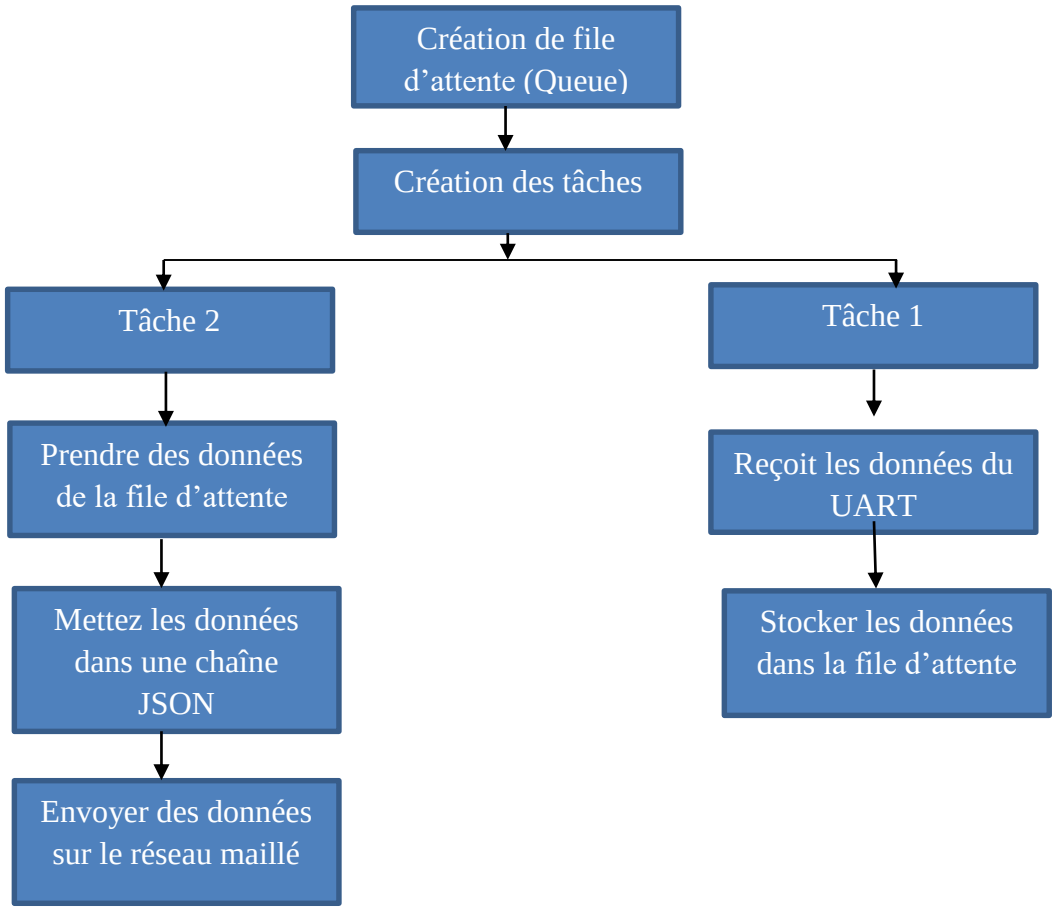

Figure 4.11 : Partage de données sur le réseau maillé.

2. Les nœuds parents intermédiaires reçoivent les informations des nœuds de traitement (les nœuds feuilles) et les envoient vers le Gatwey. Ce type de nœuds est la solution au problème des nœuds de Room 2 qui ne peuvent pas accéder directement aux nœuds racine (Gatwey) de Room 1, cela peut être dû probablement au problème de distance entre les nœuds ou aux obstacles qui limitent la propagation des ondes radio. La configuration de ces nœuds nécessite la définition des fonctions telles que **sendBroadcas , receivedCallback…** pour envoyer et recevoir des données. Le rôle de chaque fonction est bien détaillé dans le chapitre 2.

3. Les nœuds racine, qui fonctionnent comme une passerelle (Gatwey) qui permettant de relier deux réseaux informatique de types différents et le nœud supérieur du réseau, sert de seule interface entre le réseau ESP-MESH et un réseau IP externe. Il ne peut y avoir qu'un seul nœud racine dans un réseau ESP-MESH.

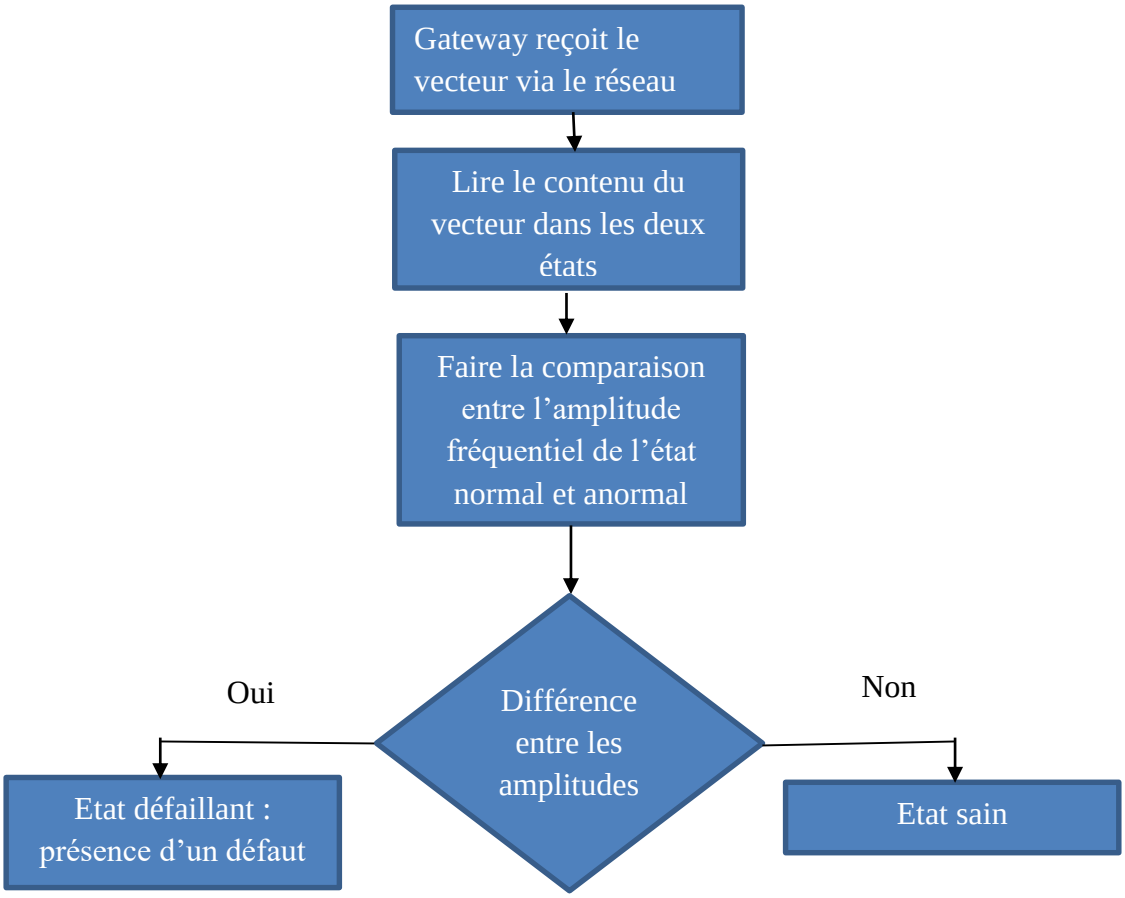

Figure 4.12 : Analyse de donné.

L'organigramme fonctionne comme suit :

- Le Gateway reçoit les données traitées par l'algorithme de la FFT : vecteur ACC, via le réseau maillé.
- Lit le contenue du vecteur, compare les amplitudes avec les amplitudes de référence : état sain.
- Le Gaetway envoie un message de l'état de la machine.

Dans le cas des exemples précédents, une différence considérable est constatée entre les amplitudes de deux états.

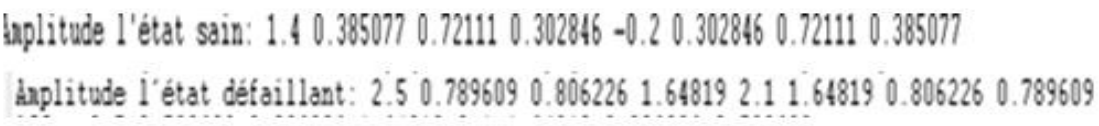

Figure 4.13 : le contenu du vecteur dans les deux états.

La transmission des données entre le nœud feuil et le Gatway nécessite d'interpréter le chemin le plus court via les nœuds parents afin d'économiser l'énergie. Pour résoudre ce problème nous utilisons l'algorithme de Dijkstra, [53] qui calcule le plus court chemin à partir d'une source vers tous les autres sommets et prend le chemin le plus court d'entre eux. La distance réelle entre les nœuds peut être présentée par RSSI indicateur d'intensité du signal reçu qui est une mesure de la puissance présente dans un signal radio reçu.

4.4 Algorithme de Dijkstra

L'algorithme de Dijkstra est également appelé algorithme de chemin le plus court à source unique pour déterminer le chemin le plus court est l'un des algorithmes les plus populaires qui sont utilisés à cette fin et il a des applications importantes que nous utilisons quotidiennement.

Les données qui voyagent sur Internet passent par différents chemins à travers le monde, il est donc important qu'elles soient dirigées par des chemins courts et moins encombrés, et cela est possible grâce à l'algorithme Dijkstra.

De plus, en utilisant cet algorithme, Google Maps peut déterminer le chemin le plus court que nous pouvons emprunter pour aller d'un point spécifique à un restaurant ou à un aéroport spécifique.

4.4.1 Implémentation de l'algorithme

❖ **Initialisation de l'algorithme :**

1. Créez la matrice de coûts (cost matrix) cost [] [] à partir de la matrice d'adjacence adj [] []. cost [i] [j] est le coût du passage du sommet i au sommet j. S'il n'y a pas d'arête entre les sommets i et j, alors cost [i] [j] est infini.
   Si G [i][j]= 0 →cost[i][j]=INFINITY
   Alors → cost[i][j]=G[i][j];
2. Le tableau visité [] est initialisé à zéro.

   pour (i = 0; i <n; i ++)

visité [i] = 0;

3. Si le sommet 0 est le sommet source, alors visité [0] est marqué comme 1.

❖ **Recherche d'un nœud de distance minimale :**

1. Créez la matrice de distance, en stockant le coût des sommets à partir du sommet no. 0 à n-1 à partir du sommet source 0.

pour (i = 1; i <n; i ++)

distance [i] = coût [0] [i];

2. Initialement, la distance du sommet source est prise à 0. c'est- à- dire la distance [0] = 0;
3. pour (i = 1; i <n; i ++)

- Choisissez un sommet w, tel que la distance [w] soit minimale et visitée[w] soit 0. Marquez visité [w] comme 1.
- Recalculez la distance la plus courte des sommets restants de la source.
- Seuls les sommets non marqués comme 1 dans le tableau visité [] doivent être pris en compte pour le recalcule de la distance. C'est-à-dire pour chaque sommet v [40] :

if (visité [v] == 0)

distance [v] = min (distance [v],

distance [w] + coût [w] [v]),

❖ **Le code :**

```
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
        int G[MAX][MAX],i,j,n,u;
        printf("Enter no. of vertices:");
        scanf("%d",&n);
        printf("\nEnter the adjacency matrix:\n");

        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                        scanf("%d",&G[i][j]);

        printf("\nEnter the starting node:");
```

```
	scanf("%d",&u);
	dijkstra(G,n,u);

	return 0;
}

void dijkstra(int G[MAX][MAX],int n,int startnode)
{

	int cost[MAX][MAX],distance[MAX],pred[MAX];
	int visited[MAX],count,mindistance,nextnode,i,j;

	//pred[] stores the predecessor of each node
	//count gives the number of nodes seen so far
	//create the cost matrix
	for(i=0;i<n;i++)
		for(j=0;j<n;j++)
			if(G[i][j]==0)
				cost[i][j]=INFINITY;
			else
				cost[i][j]=G[i][j];

	//initialize pred[],distance[] and visited[]
	for(i=0;i<n;i++)
	{
		distance[i]=cost[startnode][i];
		pred[i]=startnode;
		visited[i]=0;
	}

	distance[startnode]=0;
	visited[startnode]=1;
	count=1;

	while(count<n-1)
	{
		mindistance=INFINITY;

		//nextnode gives the node at minimum distance
		for(i=0;i<n;i++)
			if(distance[i]<mindistance&&!visited[i])
			{
				mindistance=distance[i];
				nextnode=i;
			}

			//check if a better path exists through nextnode
			visited[nextnode]=1;
```

```
                    for(i=0;i<n;i++)
                        if(!visited[i])
                            if(mindistance+cost[nextnode][i]<distance[i])
                            {
                                distance[i]=mindistance+cost[nextnode][i];
                                pred[i]=nextnode;
                            }
            count++;
        }

        //print the path and distance of each node
        for(i=0;i<n;i++)
            if(i!=startnode)
            {
                printf("\nDistance of node%d=%d",i,distance[i]);
                printf("\nPath=%d",i);

                j=i;
                do
                {
                    j=pred[j];
                    printf("<-%d",j);
                }while(j!=startnode);
        }
}
```

Dans notre travail on a besoin aussi à RSSI pour obtention de la distance par ondes radio, à utiliser dans Dijkstra algorithme :

4.4.2 RSSI

L'indicateur d'intensité du signal reçu (RSSI) est une mesure de la puissance présente dans un signal radio reçu.

RSSI est généralement invisible pour un utilisateur d'un appareil récepteur. Cependant, comme la force du signal peut varier considérablement et affecter la fonctionnalité des réseaux sans fil, les périphériques IEEE 802.11 mettent souvent la mesure à la disposition des utilisateurs.

Dans l'exemple suivant nous avons expliqué comment on a utilisé RSSI :

```
#include "WiFi.h"

const char* ssid = "yourNetworkName";
```

```
const char* password =  "yourNetworkPass";
long rssi;
void setup() {
Serial.begin(115200);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.println("Connecting to WiFi..");
  }
Serial.println("Connected to the WiFi network");
}
void loop() {
Serial.print(“rssi=“);
rssi = WIFI.RSSI();
Serial.println(rssi);
}
```

❖ **WiFi.begin:** Initialise les paramètres réseau de la bibliothèque WiFi et fournit l'état actuel.

WiFi.begin(ssid, pass);

- ssid: le SSID (Service Set Identifier) est le nom du réseau WiFi auquel vous souhaitez vous connecter.
- pass: les réseaux cryptés WPA utilisent un mot de passe sous la forme d'une chaîne pour la sécurité.

Retour :

- WL_CONNECTED lorsqu'il est connecté à un réseau ;
- WL_IDLE_STATUS lorsqu'il n'est pas connecté à un réseau, mais sous tension ;

❖ **WIFI.RSSI():**Obtient la force du signal de la connexion au routeur . [54]

4.5 Conclusion

Dans ce chapitre un scénario de surveillance des vibrations des machines par RCSF a été conçu. Il comporte deux structures, une mesure les vibrations et traite les données et l'autre envoie les données reçus dans le réseau maillé. La transformé rapide de Fourier est l'outil de diagnostic utilisé pour la surveillance des vibrations dans ce travail.

Conclusion générale

Conclusion générale

La recherche sur les réseaux des capteurs est actuellement en pleine croissance. Dans le domaine industriel l'intérêt pour RCSF s'est amplifié par la précision, flexibilité, faible coût ainsi que la facilité de déploiement de ces systèmes. Ces réseaux basés sur des capteurs sans fil déployés sur une grande échelle afin de collecter des données sur l'état des machines, une opération qui auparavant était effectué manuellement par les opérateurs.

Les machines industrielles sont soumises à des vibrations croissantes qui affectent leur bon fonctionnement. L'analyse vibratoire introduit dans RSCF dans se présente comme étant une solution efficace pour surveiller en temps réel de l'état de ces machines, afin de détecter de manière précoce leurs défaillances pour anticiper l'apparition d'une panne et ainsi assurer la continuité de service de la chaine de production.

Au court de ce projet, il nous a été demandé de mettre en place un réseau de capteur sans fil pour la maintenance prédictive de machine. Pour cela notre travail est reparti sur deux volets. Le premier volet consiste à la conception du réseau et le deuxième au traitement de signal vibratoire et son transmission dans le réseau.

La mise en place du réseau est une tâche difficile et demande de compétence en informatique, le traitement de signal demande des connaissances en électronique et l'étude des vibrations de compétence en mécanique. Il a été pour nous un grand plaisir de travailler sur une activité possédant des enjeux aussi enthousiasmants et pluridisciplinaire, et nous à donner la motivation pour mener à terme ce projet.

La conception de notre RCSF est effectuée à l'aide la bibliothèque PainlessMesh dans l'environnement de développement Arduino. Les messages entre les nœuds sont basés sur la notation JSON. La transmission des données dans le réseau se fait en temps réel à l'aide de la gestion des taches.

Le signal reçu de la machine sera traité par l'algorithme de la FFT afin de convertir les données du domaine temporel au domaine fréquentiel et extraire la totalité de composantes spectrales pour les transmettre dans le réseau.

Références bibliographiques

Références

[1] MOKHTARI Yaakoub « Diagnostic des défauts mécaniques du moteur asynchrone par l'analyse vibratoire » MÉMOIRE DE MASTER Université Mohamed Khider de Biskra 2018_2019

[2] François Lafleur «L'organisation d'un programme de maintenance prédictive»2003

[3] Omar DJEBILI « Contribution à la maintenance prédictive par analyse vibratoire des composants mécaniques tournants. Application aux butées à billes soumises à la fatigue de contact de roulement » Thèse de DOCTORAT Université Mhamed Bougara de Boumerdés 2012_2013

[4] Diagnostic des défauts Cours et Travaux dirigés, Ahmed GHORBEL, Institut supérieur des sciences appliquées et de technologie de Kairouan. 2018-2019.

[5] Abdallah KABOUCHE « Techniques de Maintenance Prédictive pour l'Amélioration de la disponibilité des Installations » THESEDOCTORAT D'ETATUNIVERSITE BADJI MOKHTAR – ANNABA

[6] Chapelot M. (EMS) et RichardA., consultants au CETIM «surveillance des machines tournantes», guide d'achat Mesures N° 757,septembre 2003.

[7] Chevalier R., «Etat de l'art de la surveillance et du diagnostic des machines tournantes à EDF», RFM, 200

[8] https://www.viaxys.com/app/download/13374497/Brochure_ViaXys_Vibrometre_Analyseur_VA-12.pdf

[9] https://www.commentcamarche.net/contents/1309-reseaux-sans-fil-wireless-networks (Date de consultation : 14/10/ 2008)

[10] Les réseaux sans fil, HADDACHE, 2010/2011 http://www.univ-bouira.dz/fr/wp-content/plugins/download-attachments/includes/download.php?id=210918

[11] BELABDELLI Abdelheq, OUKAZ Mokhtar, « Dimensionnement D'un Réseau Sans Fil Wifi », Mémoire de Fin d'études, Département de génie électrique et électronique, 2011-2012.

[12] https://sites.google.com/site/topologiesdereseau/ (Date de consultation : 6/03/ 2015)

[13] Zouatine Djamel Eddine, «Routage Multicast à Travers un Backbone Maillé Sans Fil», mémoire pour l'obtention du diplôme de Master en informatique, Département des Mathématiques et d'Informatique, Université Larbi Ben M'hidi Oum El Bouaghi, 2017, P 23, 28-29.

[14] https://www.lemagit.fr/definition/Mesh-Reseau (Date de consultation : en août 2016)

[15] https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/mesh.html

[16] Ian F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. Computer Networks, 38(4): 393-422, 2002. ISSN 1389-1286. URL http://www. sciencedirect.com/science/article/pii/S1389128601003024

[17] https://lastminuteengineers.com/esp32-arduino-ide-tutorial/

[18] Yoppy, R Harry Arjadi, Endah Setyaningsih, Priyo Wibowo, M I Sudrajat "Performance Evaluation of ESP8266 Mesh Networks ", Research Center for Quality System and Testing Technology (P2SMTP-LIPI), Indonesia Universitas Tarumanagara, 2019 , P 2.

[19] https://github.com/gmag11/painlessMesh/blob/master/README.md (Date de consultation : 30/06/ 2019)

[20] https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445308-json-definition-et-presentation-de-ce-format-de-donnees/ (Date de consultation : 11/10/2019)

[21] https://www.lebigdata.fr/json-definition(Date de consultation : 5 /12/ 2018)

[22] https://www.pierre-giraud.com/javascript-apprendre-coder-cours/json/

[23] https://randomnerdtutorials.com/decoding-and-encoding-json-with-arduino-or-esp8266/ (Date de consultation : 30/08/ 2017)

[24] https://arduinojson.org/v5/assistant/

[25] (Ref BENOIT BLANCHON CREATOR OF ARDUINOJSON, Mastering ArduinoJson 6 Efficient JSON serialization for embedded C++, P 68)

[26] https://www.guru99.com/c-sharp-serialization.html

[27] https://www.xul.fr/ajax-format-json.php

[28] https://wiki.clubelek.fr/logiciels:freertos (Date de consultation : 26 /10/2019)

[29] (Ref Richard Barry, Using the FreeRTOS real time kernel - A Practical Guide, 2009, 163 p.)

[30] https://techtutorialsx.com/2017/05/06/esp32-arduino-using-freertos-functions/ (Date de consultation : 6/05/ 2017)

[31] http://esp32.info/docs/esp_idf/html/dd/d3c/group__xTaskCreate.html (Date de consultation: 26/09/ 2016)

[32] https://www.embeddedclass.com/2019/01/11/learn-freertos-with-arduino-2-queue-and-seven-segment/(Date de consultation : 09/02/ 2020)

[33] https://techtutorialsx.com/2017/08/20/esp32-arduino-freertos-queues/(Date de consultation : 20/08/ 2017)

[34] https://docs.aws.amazon.com/fr_fr/freertos-kernel/latest/dg/queue-management.html

[35] https://www.freertos.org/Embedded-RTOS-Queues.html

[36] https://www.freertos.org/a00116.html

[37] https://www.freertos.org/a00117.html

[38] https://www.freertos.org/a00118.html

[39] ZARADER J.L« Cours de traitement du signal » Ecole Polytechnique Universitaire de Paris Spécialité Electronique Informatique ELI, 3ème Année, 2008/2009

[**40]** TEDJINI Mohsein «Sélection d'un outil du traitement du signal pour le diagnostic d'une machine tournante.», Mémoire de Magister, Université Mohamed Chérif Messaâdia de Souk-Ahras, 2015

[41] Philippe ESTOCQ « Une approche méthodologique numérique et expérimentale d'aide à la détection et au suivi vibratoire de défauts d'écaillage de roulements à billes ».THESE de doctorat. Université de REIMS CHAMPAGNE ARDENNE, 2004

[42] Carolin , Petit jean,2000,chaine acquisition restitution , Olivier Français(5,11)

[43] https://www.les-electroniciens.com/sites/default/files/cours/capteurs.pdf

[44] « Capteurs et Chaines de mesure » ministère de l'Enseignement Supérieur et de La Recherche Scientifique Université Labri Ben M'hindi, Oum El Zouaghi Faculté des Sciences et des Sciences Appliquées Département de Génie Electrique Année ; 3eme /ELN

[45] http://sitelyceejdarc.org/autodoc/cours/Ressources/Sciences%20et%20technique/060%20-Systeme/Capteurs.pdf (date de consultation : de 29/08/2011)

[46] mémoire Master2 «CHAINE D'ACQUISITION AVEC COMMUNICATION SANS FILS », 17juin2015, Khalfaoui, Nouredine ,Brahim,Mourad ,Saida(14,20)p

[47] KADDOURI Ibrahim SERAH Youcef «Détection des défauts d'une machine tournantes par l'analyse FFT » Mémoire de Master, UNIVERSITE MOHAMED BOUDIAF -M'SILA, 2018/2019

[48] MOKHTARI Yaakoub « Diagnostic des défauts mécaniques du moteur asynchrone par l'analyse vibratoire », Mémoire de Master, Université Mohamed Khider de Biskra, 2018–2019

[49] http://perso.univlemans.fr/~mtahon/doc/cpda_signal.pdf, Traitement de Signal, Marie Tahon, 2014_2015

[50] http://ftp.iar.se/WWWfiles/msp430/webic/doc/EW430_CompilerReference.pdf

[51] KARA ALI REDHA Etude, Conception et Réalisation d'une Imprimante 3D à dépôt de la matière

[52] EL HANNANI Abdelhak, REFASSI Kaddour, EL MAICHE Abbes, BOUAMAMA Mohamed "MAINTENANCE PREDICTIVE ET PREVENTIVE BASEE SUR L'ANALYSE VIBRATOIRE DES ROTORS", Laboratoire de Mécanique des Solides et des Structures Faculté des Sciences de l'Ingénieur, Université de Sidi Bel Abbes(10,12)

[53] https://www.thecrazyprogrammer.com/2014/03/dijkstra-algorithm-for-finding-shortest-path-of-a-graph.html

[54] https://www.arduino.cc/en/Reference/WiFiBegin