
DÉDICACE

C'est grâce à Dieu que tout a commencé, et c'est à Lui que je rends grâce.

Le reste n'est que dédicace.

À mon père

L'homme dont la force tranquille m'a appris à ne jamais renoncer. Derrière chaque étape de ce parcours se cachent tes sacrifices, souvent silencieux, toujours sincères. Ta confiance en moi a précédé la mienne, et c'est elle qui m'a portée jusqu'ici. Que Dieu te garde et te rende, au centuple, tout ce que tu as donné sans jamais rien demander en retour.

À ma mère

Mon port d'attache. Tu as su être ma confidente dans les silences et mon refuge dans les tempêtes. Ta patience n'a jamais eu de limites, ta tendresse n'a jamais faibli. Ce travail porte, quelque part, la trace de ton amour inconditionnel.

À Rahma

Mon amie d'enfance, celle que le temps n'a jamais éloignée. Tu as traversé mes périodes les plus sombres à mes côtés, sans jamais me laisser porter ce poids seule. Ta loyauté est un cadeau rare, et je mesure chaque jour sa valeur.

À une personne qui a compté

Le temps d'un chapitre de ma vie. Nos chemins se sont croisés, puis séparés, mais ce qu'ils ont laissé derrière eux — les souvenirs, les instants suspendus, la douceur de certains moments — je les garde intacts, sans amertume. Je te souhaite, sincèrement, que la suite te soit belle.

À mes amies

Qui ont su transformer le quotidien en quelque chose de plus léger. Vos rires, vos mots justes au bon moment et votre présence discrète mais constante ont adouci les moments de doute. Merci d'avoir fait partie de cette aventure, même sans le savoir toujours.

Remerciements

Je tiens à exprimer ma profonde gratitude à toutes les personnes qui ont contribué à la réalisation de ce projet et qui m'ont accompagnée tout au long de cette expérience enrichissante.

Je tiens tout particulièrement à remercier la société **Mentor Tunisie SCS** pour m'avoir accueillie au sein de son équipe et pour m'avoir offert l'opportunité de réaliser ce projet dans un environnement professionnel stimulant.

Mes sincères remerciements s'adressent à Monsieur **Mohamed ALAYA** et Monsieur **Aymen BEN NHILA**, responsables des ressources humaines de Mentor Tunisie SCS, pour leur confiance, leur accueil chaleureux et les conditions favorables qu'ils ont mises à ma disposition tout au long de mon stage.

J'adresse également ma profonde reconnaissance à mon encadrant industriel, Monsieur **Hosni FATHALLAH**, pour sa disponibilité, ses précieux conseils, son accompagnement constant et les connaissances qu'il m'a transmises durant toute la période de réalisation de ce projet. Son expertise et son soutien ont été d'une grande valeur pour l'aboutissement de ce travail.

Je tiens aussi à exprimer mes sincères remerciements à mes encadrants académiques, Monsieur **Mohamed Habib KAMMOUN** et Madame **Mariem ZOUARI**, pour leur suivi rigoureux, leurs orientations pertinentes, leurs remarques constructives et leur disponibilité tout au long de ce projet.

Mes remerciements s'adressent également à l'ensemble du corps enseignant de la **Faculté des Sciences**, pour la qualité de la formation dispensée et les connaissances acquises tout au long de mon parcours universitaire.

Enfin, à toutes les personnes qui ont participé, directement ou indirectement, à la réalisation de ce travail, j'exprime ma profonde gratitude et mes sincères remerciements.

Table des matières

Introduction générale.....	13
Chapitre 1 : Présentation du projet.....	14
1.1 Introduction.....	14
1.2 Cadre du projet	14
1.3 Présentation de l'organisation d'accueil.....	14
1.4 Problématique de la traçabilité produit	15
1.4.1 Identification du produit critique	15
1.4.2 Analyse des faiblesses actuelles.....	15
1.5 Étude de l'existant	17
1.5.1 La plateforme COOX.....	17
1.5.2 Le module MapexTrace	18
1.5.3 Le logiciel Cimag Production.....	18
1.5.4 Tableau comparatif des solutions de traçabilité	18
1.6 Solution proposée et objectifs opérationnels	19
1.7 Spécification des besoins	20
1.7.1 Besoins fonctionnels	20
1.7.2 Besoins non fonctionnels	20
1.8 Bases du projet	21
1.8.1 Internet des objets.....	21
1.8.2 Communication légère	21
1.8.3 Plateforme IoT.....	22
1.8.4 Gestion des données.....	22
1.9 Architecture globale du système.....	22
1.9.1 Architecture Client-Serveur.....	22
1.9.2 Architecture Publish/Subscribe.....	23
1.9.3 Architecture d'application	23
1.10 Conclusion	24
Chapitre 2 : Étude et conception	25

TABLE DES MATIÈRES

2.1	Introduction.....	25
2.2	Choix de la modélisation de la méthodologie	25
2.2.1	UML	25
2.2.2	Modèle de cycle de vie : Cascade	25
2.3	Analyse spécification de besoin.....	26
2.3.1	Analyse des acteurs.....	26
2.3.1.1	Acteurs.....	26
2.3.2	Spécification de besoin "Diagramme de cas d'utilisation global"	27
2.4	Conception	28
2.4.1	Cas d'utilisation "S'authentifier à l'interface web ".....	28
2.4.2	Cas d'utilisation "Gérer les étapes de traçabilité de production".....	31
2.4.3	Cas d'utilisation "Déclarer un défaut"	32
2.4.4	Diagramme d'Activités.....	34
2.4.5	Diagramme de classes.....	36
2.4.6	Structure de la base de données	37
2.5	Conclusion	37
Chapitre 3 : Configuration et installation du système.....		38
3.1	Introduction.....	38
3.2	Description de l'architecture matérielle du système :.....	38
3.3	Description des composants du système de traçabilité	39
3.4	Composants matériels	39
3.4.1	ESP32	39
3.4.1.1	Caractéristiques	40
3.4.2	Scanner GM65.....	40
3.4.2.1	Caractéristiques	40
3.4.3	MB102 Breadboard Power Supply Module.....	40
3.4.3.1	Caractéristiques	41
3.4.4	Écran OLED.....	41
3.4.4.1	Caractéristiques	41
3.4.5	Les boutons	42
3.5	Environnement de développement.....	42
3.5.1	Développement du système embarqué.....	42
3.5.1.1	Visual Studio Code	42

3.5.1.2	PlatformIO	42
3.5.1.3	C++	43
3.5.1.4	JSON	43
3.5.2	Développement de la plateforme IoT	43
3.5.2.1	Node-RED	43
3.5.2.2	Mosquitto	43
3.5.2.3	Wokwi	44
3.5.2.4	MySQL	44
3.5.2.5	Node.js	44
3.5.3	Développement de l'interface Web	45
3.5.3.1	React.js	45
3.5.3.2	JavaScript	45
3.5.3.3	CSS	45
3.5.3.4	HTML	45
3.6	Configuration de l'environnement de travail	46
3.6.1	Configuration des logiciels utilisés	46
3.6.1.1	Configuration du Node-RED	46
3.6.1.2	Configuration du Mosquitto	48
3.6.2	Configuration les composants de travail	48
3.6.2.1	Configuration scanner GM65 avec l'ESP32-S3 en mode UART	48
3.7	Réalisation matérielle	49
3.7.1	Simulation de câblage d'ESP32 avec Wokwi	49
3.7.2	Montage de câblage d'ESP32 du système sur prototype réel	49
3.8	Test pratique :	50
3.8.1	Lecture et afficher les données QR code depuis le scanner	50
3.8.2	Importation des données d'après la base de données	50
3.8.3	Insérer les données dans la base de données	51
3.9	Conclusion	52
Chapitre 4 : Réalisation		53
4.1	Introduction.....	53
4.2	Réalisation les étapes de production.....	53
4.2.1	Établissement de la communication.....	53
4.2.1.1	Connexion Wi-Fi	53

4.2.1.2	Connexion MQTT	54
4.2.2	Étape de scannage de la machine.....	55
4.2.2.1	Cas d'erreur.....	55
4.2.2.2	Cas de validation	55
4.2.3	Étape de scannage du processus	56
4.2.3.1	Cas d'erreur.....	56
4.2.3.2	Cas de validation	56
4.2.4	Étape de scannage du matricule.....	57
4.2.4.1	Cas d'erreur.....	57
4.2.4.2	Cas de validation	58
4.2.5	Étape de scannage du code produit	58
4.2.5.1	Identifier l'état du produit	58
4.3	Réalisation de l'interface web	61
4.3.1	Interface Web d'opérateur	61
4.3.1.1	Onglet Accueil.....	61
4.3.1.2	Onglet historique	63
4.3.2	Interface Web d'administrateur.....	63
4.3.2.1	Onglet accueil.....	63
4.3.2.2	Onglet Employés	65
4.3.2.3	Onglet PPA (Plan de Production et d'Articles).....	66
4.3.2.4	Onglet Processus et Machine	67
4.3.2.5	Onglet Défaut	67
4.3.2.6	Onglet Historique	68
4.4	Conclusion	69
Conclusion générale		70
Bibliographie.....		71

Liste des tableaux

1.1	Étude comparative des systèmes de traçabilité industrielle, incluant la solution IoT proposée.....	18
2.1	Phases de la méthodologie adoptée	26
2.2	Table explicative des cas d'utilisation	28
2.3	Description du cas d'utilisation « Accès opérateur ».....	29
2.4	Description du cas d'utilisation « Accès administrateur »	30
2.5	Description du cas d'utilisation « Gérer les étapes de traçabilité de production »	31
2.6	Description textuelle du cas d'utilisation «Identifier l'état du produit»	33
3.1	Composants du système de traçabilité industrielle basé sur IoT.....	39

Liste des abréviations

API	Application Programming Interface
CMOS	Complementary Metal-Oxide-Semiconductor
CSS	Cascading Style Sheets
DC	Direct Current (courant continu)
ESP32	Espressif Systems Processor
FSS	Faculty of Sciences of Sfax
GND	Ground (masse électrique)
GPIO	General Purpose Input/Output
HTML	HyperText Markup Language
ID	Identifiant
IO	État conforme du produit
IP	Internet Protocol
I2C	Inter-Integrated Circuit
IoT	Internet of Things (Internet des Objets)
ISO	International Organization for Standardization
JSON	JavaScript Object Notation
KO	Kilo-octet
MDP	Mot de passe
MES	Manufacturing Execution System
MISO	Master In Slave Out
MOSI	Master Out Slave In
MQTT	Message Queuing Telemetry Transport
NIO	État non conforme du produit
npm	Node Package Manager
OLED	Organic Light-Emitting Diode
OMG	Object Management Group
PPA	Plan de Production et d'Articles
QR Code	Quick Response (Code)
RFID	Radio Frequency Identification

RX	Receiver (réception série)
S.C.S	Société en Commandite Simple
SCL	Serial Clock Line (I2C)
SDA	Serial Data Line (I2C)
SGBD	Système de Gestion de Bases de Données
SoC	System on Chip
SRAM	Static Random Access Memory
SSD	Solomon Systech Driver
SSID	Service Set Identifier
TTL	Transistor-Transistor Logic
TX	Transmitter (émission série)
UART	Universal Asynchronous Receiver-Transmitter
UML	Unified Modeling Language
USB	Universal Serial Bus
VS Code	Visual Studio Code
VSS	Voltage Source Supply
Wi-Fi	Wireless Fidelity
XML	Extensible Markup Language

Table des figures

1.1	Logo Mentor Tunisie	15
1.2	Produit fini de Fasermatten	15
1.3	Pièces non conformes identifiées.....	16
1.4	Carte de suivi de production.....	17
1.5	Carte de collecte d'erreurs	17
1.6	Architecture Client-Server	23
1.7	Architecture Publish/Subscribe.....	23
1.8	Architecture d'application.....	24
2.1	Diagramme de cas d'utilisation	27
2.2	Diagramme de la séquence "Consultation de l'interface pour l'opérateur".....	29
2.3	Diagramme de la séquence "Consultation de l'interface pour l'administrateur" .	30
2.4	Diagramme de séquences "Scanner machine".....	31
2.5	Diagramme de séquences "Associer un processus"	32
2.6	Diagramme de séquences "Associer un opérateur"	32
2.7	Diagramme de séquences "Scanner et identifier l'état du produit".....	34
2.8	Diagramme d'activités "Les étapes de la traçabilité".....	35
2.9	Diagramme d'activités "Déclarer l'état de production"	35
2.10	Diagramme de classes.....	36
3.1	Montage de système	38
3.2	Carte ESP32-S3-DevKitC-1.....	39
3.3	Modèle du scanner GM65.....	40
3.4	Alimentation modèle MB102.....	41
3.5	OLED SSD1306.....	41
3.6	Bouton poussoir.....	42
3.7	Logo Visual Studio Code [27].....	42
3.8	Logo de PlatformIO.....	43

3.9	Logo Node-RED.....	43
3.10	Logo Mosquitto	43
3.11	Logo WOKWI.....	44
3.12	Logo MySQL	44
3.13	Logo Node.js.....	44
3.14	Logo React.js.....	45
3.15	Logo JavaScript.....	45
3.16	Logo CSS.....	45
3.17	Logo HTML.....	46
3.18	Validation de l'installation de Node-RED.....	46
3.19	Lancement du Node-RED	47
3.20	Relation de Node-RED avec ESP32	47
3.21	Des commandes pour configurer le fichier "mosquitto.conf"	48
3.22	Montage expérimental du prototype	49
3.23	Test affichage OLED	50
3.24	Importation de la liste des défauts	51
3.25	Message OLED " insertion réussie"	51
3.26	Test l'insertion des données.....	52
4.1	Message OLED « connexion Wi-Fi ».....	53
4.2	Code de configuration Wi-Fi de l'ESP32.....	53
4.3	Message OLED « Connexion MQTT »	54
4.4	Configuration du broker MQTT dans le code source.....	54
4.5	Système prêt et démarrage effectif	54
4.6	Affichage de l'erreur « Machine introuvable »	55
4.7	Vérification du numéro de machine	55
4.8	Traitement Node-RED de la machine	56
4.9	Affichage de l'erreur « Processus introuvable »	56
4.10	Vérification du processus scanné.....	57
4.11	Traitement Node-RED du processus.....	57
4.12	Affichage de l'erreur « Opérateur introuvable »	57
4.13	Vérification du matricule de l'opérateur	58
4.14	Traitement Node-RED du matricule	58
4.15	Confirmation de l'enregistrement des données	59

4.16	Traitement Node-RED de l'insertion de l'état IO.....	59
4.17	Affichage la liste des défaut.....	59
4.18	Traitement Node-RED de l'importation de la liste des défauts	60
4.19	Affichage de la confirmation d'insertion de l'état NIO	60
4.20	Traitement Node-RED de l'insertion des défauts sélectionnés.....	60
4.21	Traitement Node-RED de la gestion du bouton bleu.....	61
4.22	Page d'accueil de l'espace opérateur	62
4.23	Consultation des données de traçabilité dans l'espace opérateur	62
4.24	Consultation de l'historique des produits dans l'espace opérateur	63
4.25	Page d'accueil de l'espace administrateur.....	64
4.26	Consultation et analyse des défauts dans l'espace administrateur	64
4.27	Gestion des réclamations dans l'espace administrateur	65
4.28	Gestion des employés	65
4.29	Gestion des codes PPA	66
4.30	Modèle des tickets imprimés	66
4.31	Gestion des processus et des machines	67
4.32	Gestion des Défauts.....	68
4.33	Consultation de l'historique des produits.....	68

Introduction générale

Dans un environnement industriel caractérisé par des exigences accrues en qualité, productivité et compétitivité, la traçabilité des produits est devenue un élément essentiel pour assurer le suivi des opérations, maîtriser les processus de fabrication et garantir la conformité des produits. L'émergence des technologies de l'Internet des objets (IoT) offre aujourd'hui de nouvelles perspectives pour mettre en place des systèmes de traçabilité intelligents, capables de collecter et de transmettre les données en temps réel.

C'est dans ce cadre que s'inscrit ce projet de fin d'études, réalisé au sein de Mentor Tunisie S.C.S., en partenariat avec la Faculté des Sciences de Sfax. L'objectif est de concevoir et de développer un système de traçabilité intelligent destiné au suivi des produits Fasermatten, utilisés dans le secteur du textile automobile. L'absence d'un système de suivi en temps réel entraînait des difficultés dans le contrôle de la production, une augmentation des déchets ainsi que des pertes de temps liées à l'identification des anomalies.

Afin de répondre à cette problématique, une solution basée sur les technologies IoT a été développée. Elle repose sur un microcontrôleur ESP32, un lecteur de codes QR, une base de données centralisée et une interface web permettant d'assurer le suivi des produits tout au long du processus de fabrication. Cette solution améliore la fiabilité des données collectées, facilite la détection des anomalies et contribue à l'optimisation de la qualité et des performances du processus de production.

Le présent rapport est structuré en quatre chapitres. Le premier présente le contexte général du projet, l'entreprise d'accueil ainsi que l'étude de l'existant et la problématique. Le deuxième chapitre est consacré à l'analyse des besoins et à la conception de la solution proposée. Le troisième décrit l'architecture technique ainsi que les technologies mises en œuvre pour le développement du système. Enfin, le quatrième chapitre présente la réalisation de la solution, les tests effectués et les résultats obtenus.

Présentation du projet

1.1 Introduction

Ce chapitre présente le contexte du projet, l'entreprise d'accueil, la problématique de la traçabilité en production, la solution proposée ainsi que les besoins fonctionnels et non fonctionnels. Il introduit également les principales technologies et concepts utilisés pour la conception du système.

1.2 Cadre du projet

Dans le cadre de mon stage de fin d'études chez Mentor Tunisie S.C.S en partenariat avec la **Faculté des Sciences de Sfax**, j'ai mené à bien ce projet. L'objectif était d'améliorer le suivi de la production et de limiter les erreurs humaines grâce à une approche d'amélioration continue.

1.3 Présentation de l'organisation d'accueil

Mentor Tunisie S.C.S [1] fait partie du groupe Mentor, une entreprise allemande fondée en 1920. La société conçoit et assemble des composants électromécaniques et électroniques pour des clients internationaux tels qu'Audi, BMW, Volkswagen et Siemens. Implantée à **Zeramidine (Monastir)** depuis 1980, la seule branche située en Tunisie, la société a évolué d'une petite unité de production à un acteur industriel majeur, intégrant des technologies avancées et respectant les normes internationales de qualité et de sécurité, notamment les normes suivantes :

- **ISO 9001 :2015** - Qualité management
- **ISO 14001 :2015** - Management environnemental

L'entreprise fonctionne en s'appuyant sur plusieurs services clés : la production, la qualité, la logistique, l'informatique, les ressources humaines et la sécurité. Cette organisation permet d'assurer une gestion industrielle efficace et bien structurée.



FIGURE 1.1. Logo Mentor Tunisie

1.4 Problématique de la traçabilité produit

Dans le processus de fabrication, chaque incident est associé à un produit spécifique, ce qui permet une traçabilité exacte, facilite la détermination de la cause sous-jacente et permet la mise en œuvre d'une solution appropriée et durable.

1.4.1 Identification du produit critique

Le **Fasermatten** [2] est un tapis textile en fibres naturelles ou synthétiques, spécialement conçu pour la décoration et la finition intérieure des automobiles. C'est le premier produit de ce type au monde et en Tunisie développé par la seule entreprise Mentor Tunisie S.C.S, ce qui en fait un produit unique sur le marché local. La figure 1.2 présente le produit fini de Fasermatten dans un modèle automobile.



FIGURE 1.2. Produit fini de Fasermatten

1.4.2 Analyse des faiblesses actuelles

Les déchets du produit Fasermatten chez Mentor sont trop élevés, et cela était dû à de nombreux facteurs, comme :

- **L'absence de l'industrie 4.0** : L'industrie 4.0 présente l'efficacité et la traçabilité des processus industriels. Son absence accroît le risque d'erreurs humaines et limite la réactivité face aux besoins de production.
- **Manque de suivi de la traçabilité en temps réel** : L'entreprise ne possède pas d'outil pour suivre en direct la production et les déchets générés. Cette absence de visibilité en temps réel complique le contrôle des opérations, réduit sa capacité à réagir rapidement et compromet la fiabilité des données relatives au rendement des pièces. La figure 1.3 illustre un exemple de pièces rebutées identifiées manuellement sans système de traçabilité automatisé.

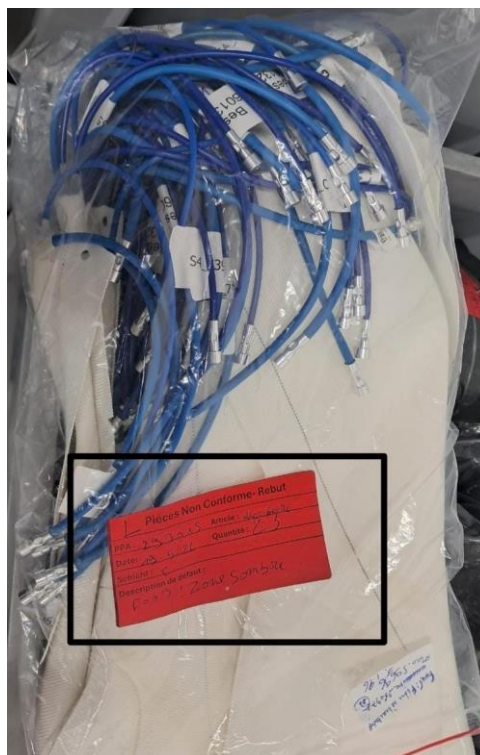


FIGURE 1.3. Pièces non conformes identifiées

- **Déclaration manuelle** : Le système actuel de déclaration manuelle, qu'il s'agisse de remplir des formulaires papier ou de saisir les données au clavier, est source d'inefficacités. Il fait perdre un temps précieux, complique le suivi en temps réel et expose inutilement les chiffres de production et de rendement à des erreurs. Les figures 1.4 et 1.5 montrent les cartes papier actuellement utilisées par les opérateurs pour la collecte des erreurs et le suivi de production, remplies manuellement

Carte de suivi(a conserver impérativement)	
Article	
PPA	
Process	
Date	
Matricule opérateur	
Visa Chef d'équipe	
Visa controle Qualité	

FIGURE 1.4. Carte de suivi de production

Fehlersammelkarte - TN / Carte de collecte d'erreurs						
Date				Article		
PPA				PPA Qty		
Process				Machine		
Schicht						
Matricule						
	S-1	S	S-1	S	S-1	S
Qty produite						
Défautes	Produkte					
Remarques						

FIGURE 1.5. Carte de collecte d'erreurs

1.5 Étude de l'existant

Dans le domaine industriel, plusieurs solutions de traçabilité existent, nous présentons ci-dessous les principales afin de les comparer aux besoins de l'entreprise.

1.5.1 La plateforme COOX

COOX [3] est une plateforme de traçabilité industrielle qui avance et permet de suivre chaque produit en temps réel. Un module de généalogie ultra-performant. Il retrace la vie complète d'un produit, depuis la matière première jusqu'à l'expédition, en enregistrant chaque opération, chaque contrôle, chaque main qui l'a transformé. COOX est utilisé dans plusieurs secteurs nécessitant une traçabilité fiable. Il permet de suivre les produits alimentaires, de garantir l'authenticité des médicaments et d'assurer le suivi des pièces dans les industries automobiles et aéronautiques. COOX [4] offre une traçabilité totale, sans faille, et la garantie d'être en conformité avec les normes les plus strictes. C'est un partenaire stratégique pour sécuriser votre production et la réputation de votre marque.

1.5.2 Le module MapexTrace

MapexTrace [6] est un module de traçabilité industrielle conçu par Mapex. Il garantit une traçabilité totale des produits pendant tout le processus de fabrication, des matières premières aux produits finis. En collectant automatiquement les données grâce à des technologies telles que les codes-barres et la RFID, il permet de suivre les lots, d'identifier les composants utilisés et de générer un historique de fabrication détaillé, ce qui assure la qualité, la sécurité et la conformité des produits.

1.5.3 Le logiciel Cimag Production

Cimag Production [5] est une solution MES (Manufacturing Execution System) développée par Alpha-3i, destinée à la supervision et au pilotage en temps réel des ateliers de production. Elle permet de recueillir les données de fabrication, d'assurer la traçabilité des produits et des processus, de gérer la qualité, de suivre les performances industrielles et d'optimiser les opérations de production afin d'améliorer la productivité et la réactivité des entreprises.

1.5.4 Tableau comparatif des solutions de traçabilité

Afin de mieux positionner notre projet par rapport aux solutions existantes, nous avons élaboré un tableau comparatif présentant les principales caractéristiques. Le tableau 1.1 présente une comparaison des solutions existantes selon leurs critères.

Critère	COOX	Cimag Production	MapexTrace	Système proposé
Traçabilité temps réel	Faible	Moyenne	Faible	Élevée
IoT	Non	Non	Non	Oui
Coût	Élevé	Moyen	Élevé	Moyen
Maintenance	Complexe	Complexe	Moyenne	Simplifiée
Personnalisation	Faible	Moyenne	Moyenne	Élevée
Alertes	Non	Oui	Non	Oui
Dashboard	Basique	Moyen	Basique	Avancé
Détection défauts	Non	Oui	Non	Oui

TABLE 1.1. Étude comparative des systèmes de traçabilité industrielle, incluant la solution IoT proposée

Le tableau 1.1 montre que le système proposé se distingue par une traçabilité élevée en temps réel, l'intégration de l'IoT, une maintenance simplifiée et un tableau de bord avancé. Il offre également des fonctionnalités de détection des défauts et d'alertes, absentes ou limitées dans plusieurs solutions existantes. Avec un coût moyen et une forte capacité de personnalisation, il constitue une solution plus flexible et mieux adaptée aux besoins de l'industrie moderne.

1.6 Solution proposée et objectifs opérationnels

Pour pallier les faiblesses identifiées, nous proposons la mise en place d'un système de traçabilité IoT en temps réel pour le suivi des produits Fasermatten sur une ligne de production composée de dix postes de travail. Cette solution est basée sur une architecture qui inclut un microcontrôleur ESP32, un scanner de codes QR, un broker MQTT, Node-RED et une base de données MySQL. Elle permet d'acquérir, de transmettre et d'enregistrer automatiquement les informations concernant les machines, les processus, les opérateurs, les produits et les défauts détectés. Les principales missions de ce projet sont les suivantes :

1. Automatisation du suivi de la production : remplacer la saisie manuelle par un scannage séquentiel des machines, des processus, des opérateurs et des produits pour garantir une collecte automatique et fiable des données.
2. Contrôle de conformité : prévoir des boutons physiques permettant de valider la conformité des produits ou de signaler les défauts détectés pendant la production.
3. Centralisation des données : transférer les informations en temps réel via le protocole MQTT et les enregistrer automatiquement avec horodatage dans une base de données, à l'aide de Node-RED.
4. Supervision et alertes : concevoir une interface web pour suivre en temps réel les performances de la ligne de production et alerter immédiatement l'administrateur en cas d'anomalie ou d'opérateur introuvable.

Cette solution permet d'améliorer la traçabilité des produits, de réduire les erreurs humaines, d'optimiser le suivi de la production et de répondre aux exigences de l'Industrie 4.0 par la digitalisation des processus industriels.

1.7 Spécification des besoins

L'étude de l'existant nous a permis d'identifier les besoins fonctionnels et non fonctionnels auxquels notre solution doit répondre.

1.7.1 Besoins fonctionnels

1. Administrateur :
 - Ajouter, supprimer et modifier les données des machines.
 - Ajouter, supprimer et modifier les données des processus de production.
 - Ajouter, supprimer et modifier les données des opérateurs.
 - Gérer les listes des défauts associés à chaque processus.
 - Gérer les listes des machines associées à chaque processus.
 - Consulter l'historique de production.
 - Recevoir les notifications en cas d'anomalie.
2. Opérateur :
 - Scanner la machine (une seule fois).
 - Scanner le processus (une seule fois).
 - Scanner leur matricule pour l'identification (une seule fois).
 - Scanner le code du produit.
 - Déclarer l'état du produit via des boutons de validation et d'erreur.
 - Sélectionner un ou plusieurs défauts en cas d'erreur.
 - Naviguer entre les étapes via des boutons (validation, retour).
 - Terminer la session (déconnexion) via un bouton dédié.

1.7.2 Besoins non fonctionnels

1. **Sécurité** : Le système doit protéger les données et empêcher les accès ou les modifications non autorisés.
2. **Fiabilité** : Le système doit fonctionner correctement et assurer le bon déroulement des différentes étapes du processus.
3. **Performance** : Le système doit traiter les informations rapidement afin de ne pas perturber la production.
4. **Maintenabilité** : L'architecture doit être organisée de manière modulaire pour faciliter la

maintenance et les futures améliorations.

5. **Traçabilité** : Le système doit enregistrer les différentes étapes de production afin d'assurer un suivi complet des opérations.
6. **Scalabilité** : Le système doit permettre l'ajout de nouvelles stations de production sans modification importante de l'architecture existante.
7. **Facilité d'utilisation** : L'interface opérateur doit être simple et intuitive afin de faciliter l'utilisation du système.
8. **Consommation énergétique** : Le système doit fonctionner avec une consommation énergétique réduite et une alimentation stable pour garantir son bon fonctionnement.

En respectant ces exigences non fonctionnelles, le système de traçabilité développé garantit une gestion efficace des données, un suivi en temps réel de la production et une architecture capable d'évoluer selon les besoins futurs.

1.8 Bases du projet

Dans ce cadre, nous fournissons les éléments techniques et fonctionnels permettant de structurer l'architecture de notre système.

1.8.1 Internet des objets

L'Internet des objets (IoT) [9] permet de connecter des objets physiques à des systèmes informatiques afin de collecter et échanger des données. Les objets connectés peuvent communiquer entre eux ou avec une plateforme de supervision. L'IoT est utilisé pour connecter les équipements de production au système de traçabilité. Il permet de récupérer les informations des produits, de les transmettre en temps réel et d'assurer un meilleur suivi de la production.

1.8.2 Communication légère

MQTT [10] est un protocole de communication utilisé dans les systèmes IoT pour permettre l'échange de données entre les différents éléments d'un système. Il fonctionne selon un modèle de publication/abonnement, où les données sont envoyées sur des canaux appelés « topics » et reçues par les éléments intéressés. MQTT permet d'assurer une communication rapide et fiable entre les différents composants du système, tout en réduisant la quantité de données échangées.

1.8.3 Plateforme IoT

Node-RED [19] est un outil qui permet de créer des applications en reliant des blocs appelés « Nœuds ». Chaque nœud réalise une tâche spécifique comme la collecte, le traitement ou la transmission des données. Node-RED est utilisé pour assurer la communication entre les différents éléments du système. Il permet de traiter les données en temps réel, de les organiser et de les transmettre vers les services nécessaires afin d'améliorer le suivi de la production.

1.8.4 Gestion des données

Une base de données est un système qui permet de stocker, organiser et gérer des informations de manière efficace. Elle permet d'ajouter, consulter, modifier et supprimer des données selon les besoins. La base de données joue un rôle important dans le système de traçabilité. Elle permet de centraliser les informations de production, de conserver l'historique des opérations et de faciliter le suivi et l'analyse des données.

1.9 Architecture globale du système

Cette section décrit l'architecture de notre système. Elle présente les modèles Client-Serveur, Publish/Subscribe ainsi que l'architecture de l'application.

1.9.1 Architecture Client-Serveur

L'architecture client-serveur [24] est un modèle de communication reposant sur deux entités principales : le client, qui émet des requêtes, et le serveur, qui les traite et fournit les réponses appropriées. Chaque composant possède un rôle bien défini. Le client est chargé de collecter ou de demander des informations, tandis que le serveur assure leur traitement, leur stockage et leur gestion. Dans le cadre de notre projet de traçabilité basé sur l'ESP32, cette architecture a été organisée en trois couches principales :

- Client : inclut l'ESP32 lui-même et une interface web.
- Middleware : Mosquitto qui fait office de broker MQTT et Node-RED pour coordonner tout ça.
- Serveur : on retrouve une base de données MySQL qui conserve l'historique des données.

La figure 1.6 illustre l'architecture client-serveur mise en œuvre dans ce projet.

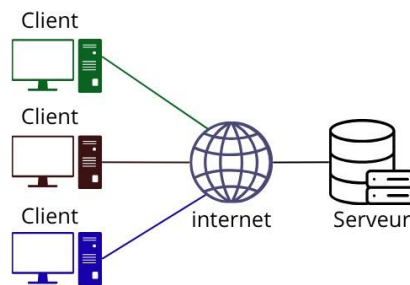


FIGURE 1.6. Architecture Client-Server

1.9.2 Architecture Publish/Subscribe

L'architecture Publish/Subscribe [25] est un modèle de communication dans lequel les producteurs de données publient leurs messages vers un broker, tandis que les consommateurs reçoivent uniquement les messages correspondant aux sujets auxquels ils sont abonnés. Contrairement au modèle client-serveur, les échanges ne s'effectuent pas directement entre l'émetteur et le récepteur ; le broker assure la réception et la distribution des messages.

Dans notre projet, cette architecture comprend trois composants principaux :

- Publisher : représente l'ESP32 et le scanner qui publient les données.
- Broker : Mosquitto qui centralise et distribue les messages.
- Subscriber : Node-RED, interface web et base de données qui consomment et exploitent les données.

La figure 1.7 montre la structure de l'architecture Publish/Subscribe.

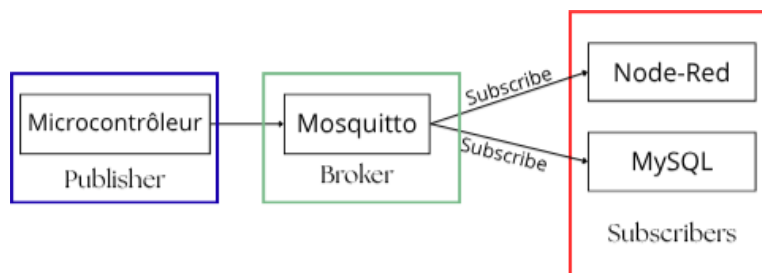


FIGURE 1.7. Architecture Publish/Subscribe

1.9.3 Architecture d'application

Cette architecture montre comment est organisée l'application de traçabilité de production dans son ensemble, ainsi que les échanges entre ses différents éléments. L'opérateur réalise les

Opérations de traçabilité sur la ligne de production grâce à l'ESP32-S3 qui dispose d'un lecteur de codes QR, d'un afficheur OLED et de boutons de commande. Les données collectées sont envoyées sur le réseau Wi-Fi suivant le protocole MQTT vers le broker Mosquitto, qui gère et transmet les messages. Node-RED reçoit ces messages, les traite comme il se doit, vérifie ce qui doit être vérifié et enregistre les informations dans la base de données.

L'interface Web dédiée à l'administrateur permet de consulter, suivre et gérer les données de production en temps réel. Cette architecture intègre les communications filaires pour les équipements locaux et les communications sans fil pour les échanges entre les différents services, assurant ainsi une solution fiable, flexible et adaptée aux exigences de la traçabilité industrielle.

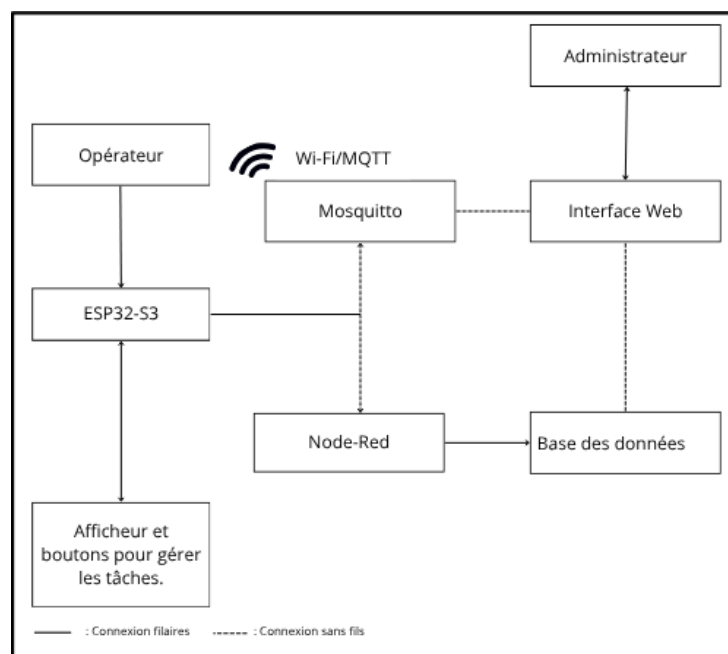


FIGURE 1.8. Architecture d'application

1.10 Conclusion

Ce chapitre a posé les bases du projet en présentant son contexte, la problématique abordée et la solution envisagée. Les besoins fonctionnels et non fonctionnels du système ont été définis, de même que les principales technologies retenues pour sa mise en œuvre. Cette analyse servira de fondement au chapitre suivant, qui sera consacré à l'analyse, à la conception et à l'architecture de la solution.

Étude et conception

2.1 Introduction

Ce chapitre présente la conception du système de traçabilité des produits Fasermatten à l'aide des diagrammes UML et justifie le choix de la méthode adoptée

2.2 Choix de la modélisation de la méthodologie

Pour structurer le développement du projet, nous avons utilisé le langage UML pour modéliser le système et la méthodologie en cascade pour organiser les différentes phases de réalisation.

2.2.1 UML

UML (Unified Modeling Language) [7] est un langage de modélisation qui sert à décrire le fonctionnement et la structure d'un système. Il permet d'analyser et de concevoir avant de développer l'application. Nous avons choisi d'utiliser PlantUML [11] pour la réalisation de nos différents diagrammes UML car son ergonomie est simple et ses fonctionnalités répondent à nos besoins.

2.2.2 Modèle de cycle de vie : Cascade

Nous avons choisi d'organiser le développement de notre projet selon la méthodologie en cascade (Waterfall) [8]. Ce choix s'explique par le fait que les besoins et les objectifs sont clairement définis dès le début du stage. Cette méthode comprend des étapes effectuées de façon séquentielle et chaque phase doit être validée avant de passer à la suivante afin de garantir le bon déroulement du projet. Le tableau ci-dessous présente les principales phases suivies :

Phase	Activités réalisées
Analyse	Étude du système existant, identification des besoins fonctionnels et non fonctionnels, analyse des solutions de traçabilité existantes.
Conception	Élaboration des diagrammes UML, conception de l'architecture matérielle, logicielle et modélisation de la base de données.
Développement	Programmation de l'ESP32, mise en place des flux Node-RED, développement de l'interface web et intégration de la base de données.
Tests et validation	Vérification du fonctionnement global du système, validation des communications MQTT et tests en environnement réel.

TABLE 2.1. Phases de la méthodologie adoptée

2.3 Analyse spécification de besoin

Nous présentons les différents cas d'utilisation du projet et les expliquons en détail.

2.3.1 Analyse des acteurs

2.3.1.1 Acteurs

Administrateur : Il s'occupe des données lors de la fabrication et du système de gestion de base de données. Il dispose des modules suivants :

- Gérer les données.
- Consulter les alertes d'anomalies.
- Analyser les graphiques de défauts.

Opérateur : Il exécute les opérations de traçabilité.

- Gérer les étapes de la traçabilité.
- Déclarer les défauts (conforme / non conforme).
- Retourner à l'étape précédente.
- Se déconnecter du système de traçabilité.

Utilisateur : Il accède à l'interface web.

- Consulter l'interface web.
- Se déconnecter de l'interface web.

La consultation de l'interface dépend du rôle de l'utilisateur, et c'est pour cela que l'on applique

L'héritage des cas d'utilisation afin de partager les fonctionnalités communes tout en distinguant les responsabilités propres à chaque acteur.

2.3.2 Spécification de besoin "Diagramme de cas d'utilisation global"

La figure 2.1 représente le diagramme de cas d'utilisation qui regroupe les principales fonctionnalités du système. Une description détaillée de chaque cas d'utilisation est présentée dans le tableau 2.2.

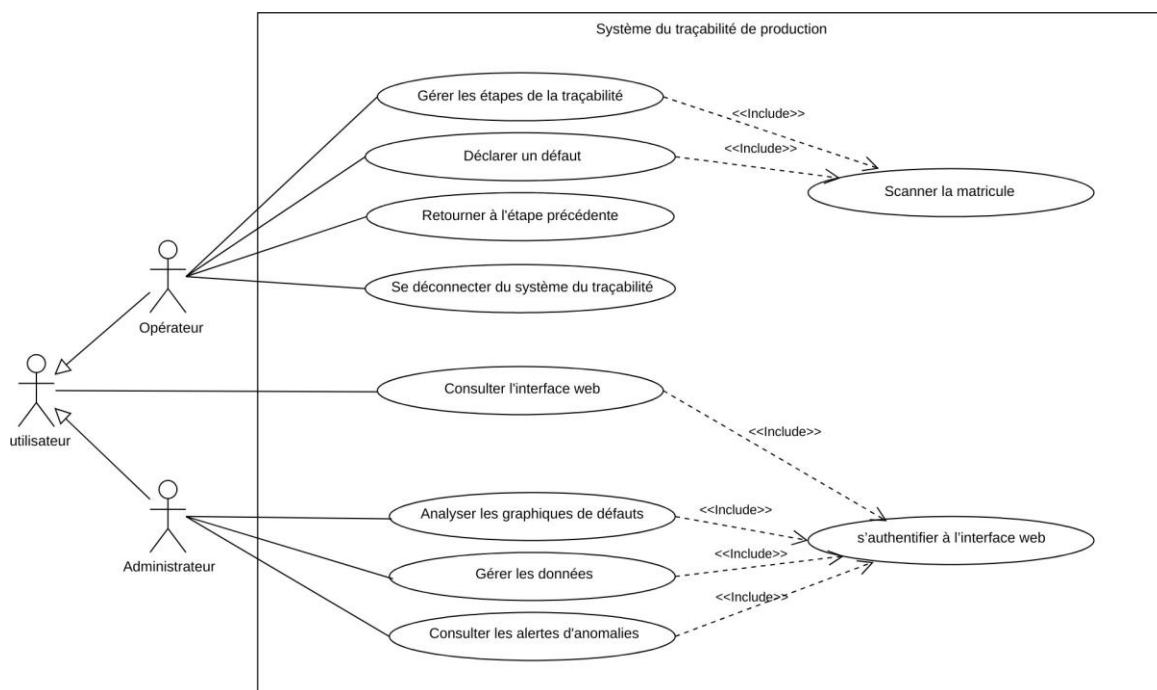


FIGURE 2.1. Diagramme de cas d'utilisation

Ce tableau 2.2 représente les différents cas d'utilisation étudiés et les différents accès du système de traçabilité industrielle pour chaque acteur.

Cas d'utilisation	Détails	Accessible via
Gérer les étapes de la traçabilité	Machine : identification avant démarrage de la production	ESP32 + Scanner
	Processus : identification du processus associé à la machine	ESP32 + Scanner
	Matricule : identification de l'opérateur	ESP32 + Scanner
	Code produit : identification du produit et association des données de traçabilité	ESP32 + Scanner
Déclarer un défaut	Sélection d'un ou plusieurs codes d'erreur, avec import automatique des données associées	ESP32 + Bouton
Retourner à l'étape précédente	Utilisation du bouton bleu pour revenir à l'étape précédente	ESP32 + Bouton
S'authentifier sur l'interface web	Vérification de l'identité par identifiant et mot de passe	WEB
Consulter l'interface web	Suivi en direct des principaux indicateurs de production	WEB
Analyser les graphiques de défauts	Analyse statistique des défauts par produit, machine ou période	WEB
Consulter les alertes d'anomalies	Déclenchement automatique d'une alerte en cas d'anomalie	WEB
Gérer les données	Mise à jour des données existantes et ajout des informations manquantes	WEB

TABLE 2.2. Table explicative des cas d'utilisation

2.4 Conception

Cette partie détaille quelques cas d'utilisation représentatifs du système, afin d'illustrer les principales interactions entre les acteurs et l'application.

2.4.1 Cas d'utilisation "S'authentifier à l'interface web "

Accès opérateur :

A. Description textuelle :

Élément	Description
Cas d'utilisation	Accès opérateur
Acteur	Opérateur
Précondition	L'opérateur est enregistré dans la base de données.
Scénario nominal	1. L'opérateur saisit son matricule. 2. L'interface transmet les données au serveur. 3. Le serveur vérifie l'existence du compte ainsi que son rôle. 4. Le système confirme que le rôle est « opérateur ». 5. L'accès à l'interface opérateur est autorisé.
Scénario alternatif	1. L'opérateur n'existe pas ou son rôle est invalide. 2. Le système affiche le message : « Matricule inconnu ». 3. L'accès est refusé.

TABLE 2.3. Description du cas d'utilisation « Accès opérateur »

A. Diagramme de séquence :

La figure 2.2 illustre le diagramme de séquence de l'authentification de l'opérateur.

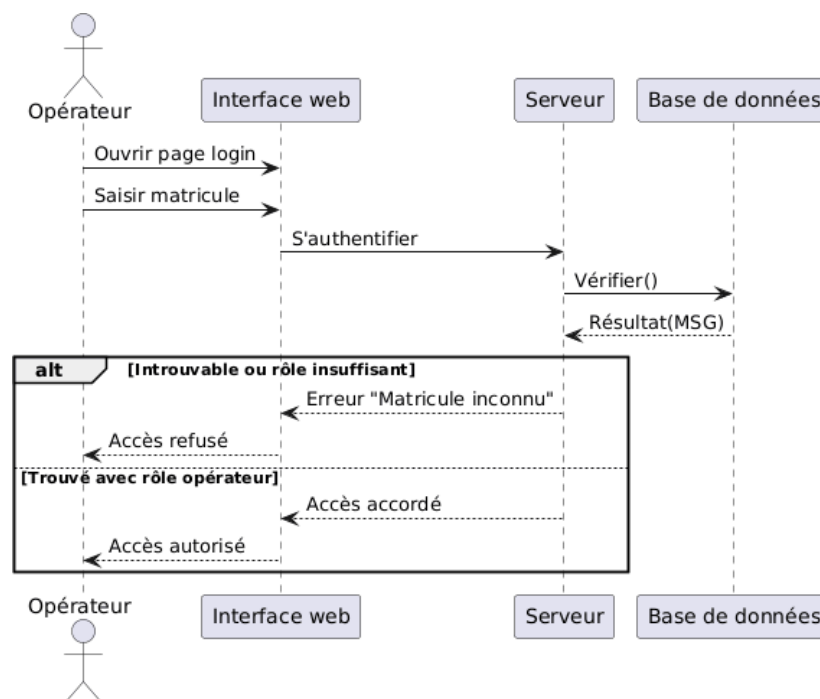


FIGURE 2.2. Diagramme de la séquence "Consultation de l'interface pour l'opérateur"

Accès administrateur :

A. Description textuelle

Élément	Description
Cas d'utilisation	Accès administrateur
Acteur	Administrateur
Précondition	l'administrateur est enregistré dans la base de données.
Scénario nominal	1. L'administrateur saisit son matricule. 2. L'interface transmet les données au serveur. 3. Le serveur vérifie l'existence du compte et son rôle. 4. Le système confirme que le rôle est « administrateur ». 5. L'administrateur saisit son mot de passe. 6. L'interface transmet les données au serveur. 7. Le serveur vérifie la correspondance entre le matricule et le mot de passe. 8. Le système valide les informations saisies. 9. L'accès à l'interface administrateur est autorisé.
Scénario alternatif	1. L'administrateur n'existe pas dans la base de données. 2. Le système affiche le message : « Matricule inconnu ». 3. Le mot de passe est incorrect. 4. Le système affiche le message : « Mot de passe incorrect ». 5. L'accès est refusé.

TABLE 2.4. Description du cas d'utilisation « Accès administrateur »

A. Diagramme de séquences

La figure 2.3 présente le diagramme de séquence de l'authentification à l'interface web.

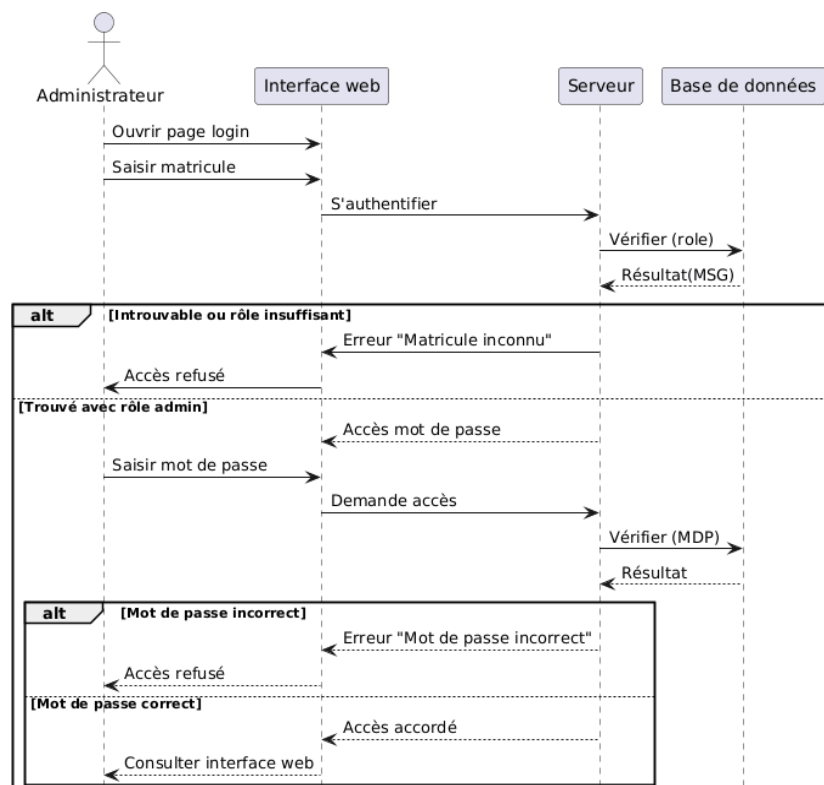


FIGURE 2.3. Diagramme de la séquence "Consultation de l'interface pour l'administrateur"

2.4.2 Cas d'utilisation "Gérer les étapes de traçabilité de production"

B. Description textuelle :

Élément	Description
Cas d'utilisation	Gérer les étapes de traçabilité de production
Acteur	Opérateur
Précondition	L'opérateur est au poste de production, l'ESP32 est connecté à la plateforme IoT et les données des machines, des processus et des opérateurs sont enregistrées dans la base.
Scénario nominal	1. L'opérateur scanne le code de la machine. 2. Le système vérifie l'existence de la machine dans la base de données. 3. L'opérateur scanne le code du processus. 4. Le système vérifie la correspondance entre la machine et le processus. 5. Le système valide le processus et autorise le passage à l'étape suivante. 6. L'opérateur scanne son matricule. 7. Le système vérifie l'existence de l'opérateur. 8. Le système associe l'opérateur à l'opération de production. 9. Le système enregistre les informations de traçabilité. 10. Le système confirme la validation de l'opération.
Scénario alternatif	1. Machine inexistante : la machine n'est pas trouvée, message d'erreur OLED, rescanner une machine valide 2. Processus non compatible : incompatibilité détectée, message « Processus non compatible », rescanner un processus valide. 3. Opérateur inexistant : opérateur introuvable, message d'erreur, rescanner le matricule.

TABLE 2.5. Description du cas d'utilisation « Gérer les étapes de traçabilité de production »

B. Diagramme de séquences :

Les figures 2.4, 2.5 et 2.6 illustrent les diagrammes de séquence de la traçabilité de production.

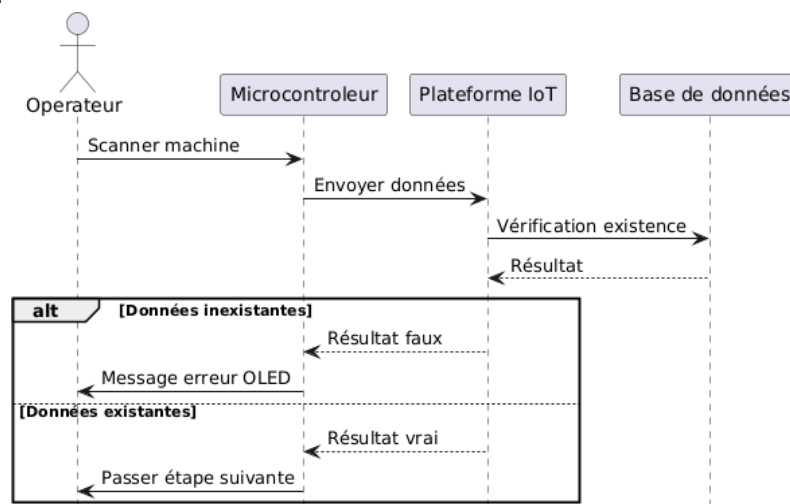


FIGURE 2.4. Diagramme de séquences "Scanner machine"

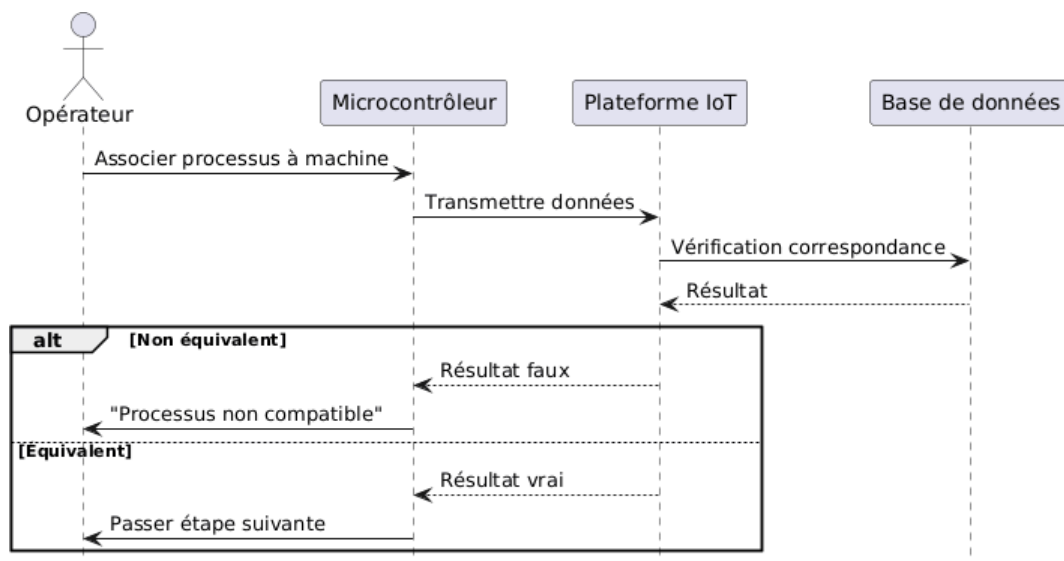


FIGURE 2.5. Diagramme de séquences "Associer un processus"

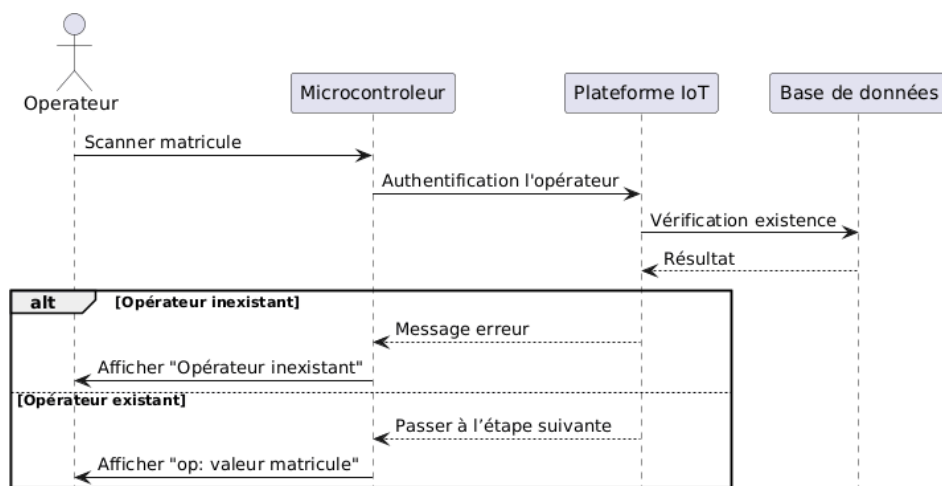


FIGURE 2.6. Diagramme de séquences "Associer un opérateur"

2.4.3 Cas d'utilisation "Déclarer un défaut"

C. Description textuelle

Élément	Description
Cas d'utilisation	Identifier l'état du produit
Acteur	Opérateur
Précondition	— L'opérateur a déjà validé la machine, le processus et son matricule. — Le microcontrôleur est connecté à la plateforme IoT. — La base de données est accessible.
Scénario nominal	1. L'opérateur scanne le code du produit. 2. Le microcontrôleur affiche le code produit scanné. 3. Le système vérifie la structure du code produit. 4. Le système valide le format du code produit. 5. Le système identifie l'état du produit. 6. L'opérateur confirme l'opération à l'aide du bouton approprié. 7. Le microcontrôleur transmet les données à la plateforme IoT. 8. La plateforme IoT enregistre les données dans la base de données. 9. La base de données confirme l'insertion. 10. La plateforme IoT retourne une confirmation au microcontrôleur. 11. Le microcontrôleur affiche un message de validation. 12. Les informations de traçabilité sont enregistrées avec succès.
Scénario alternatif	Cas 1 : Produit conforme (IO) 1. Le système reconnaît le produit comme conforme. 2. Les données sont transmises à la plateforme IoT. 3. Les informations sont enregistrées dans la base de données. 4. Un message de validation est affiché. 5. Le scénario se termine avec succès. Cas 2 : Produit non conforme (NIO) 1. Le système identifie le produit comme non conforme. 2. L'opérateur appuie sur le bouton de validation. 3. Le microcontrôleur demande la liste des défauts à la plateforme IoT. 4. La plateforme IoT récupère la liste depuis la base de données et l'affiche sur l'écran OLED. 5. L'opérateur utilise le bouton bleu pour parcourir les défauts disponibles. 6. Le système affiche successivement les défauts. 7. L'opérateur utilise le bouton vert pour sélectionner un défaut. 8. Le défaut est ajouté à la liste des défauts sélectionnés. 9. Les étapes précédentes sont répétées jusqu'à la fin de la sélection. 10. L'opérateur valide la liste des défauts. 11. Le microcontrôleur transmet la liste des défauts à la plateforme IoT. 12. La plateforme IoT enregistre les informations dans la base de données. 13. La base de données confirme l'insertion. 14. Le microcontrôleur affiche un message de confirmation. 15. Le scénario se termine avec succès.

TABLE 2.6. Description textuelle du cas d'utilisation « Identifier l'état du produit »

C. Diagramme de séquences :

La figure 2.7 présente le diagramme de séquence de la déclaration d'un défaut du produit.

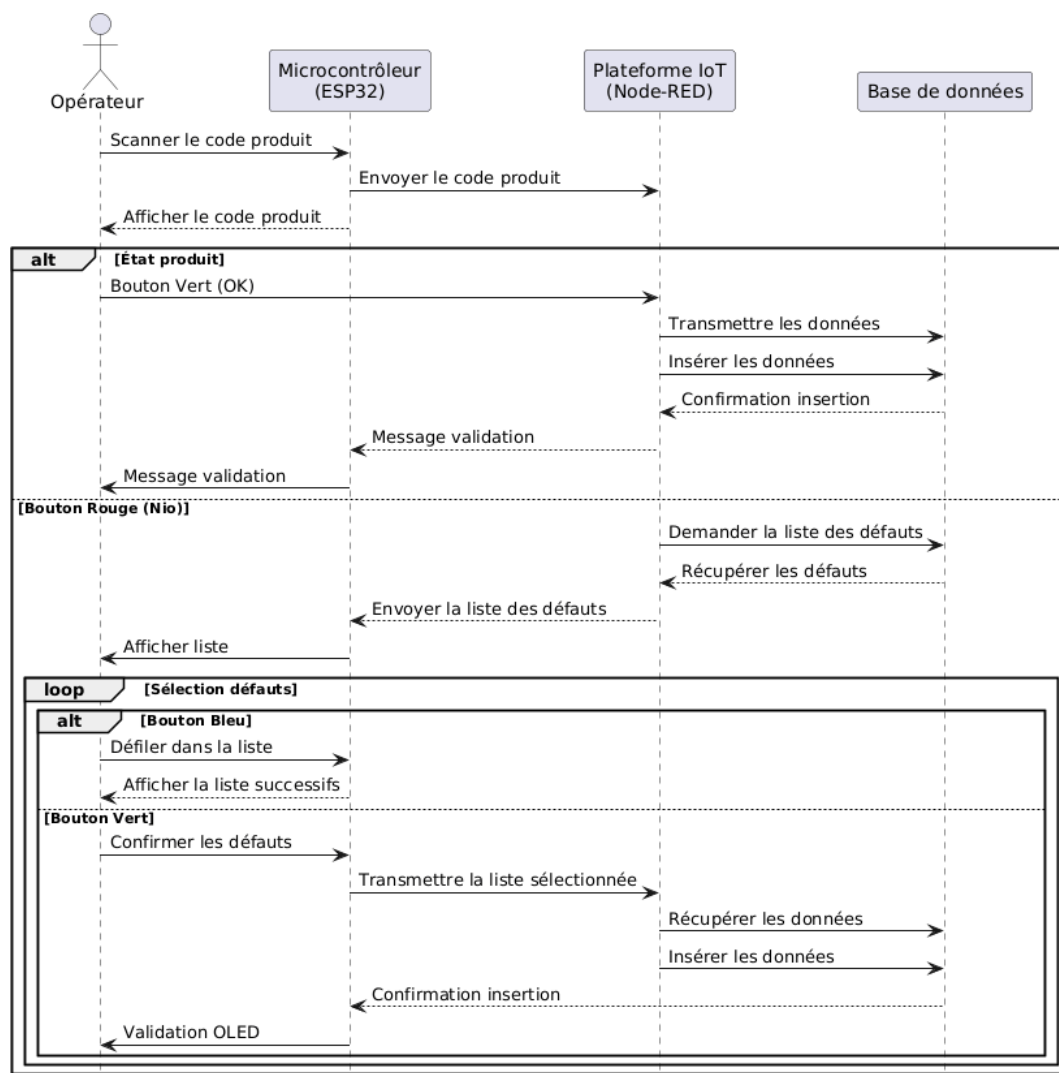


FIGURE 2.7. Diagramme de séquences "Scanner et identifier l'état du produit"

2.4.4 Diagramme d'Activités

Les deux figures illustrent le processus de traçabilité, organisé en deux volets complémentaires. La figure 2.8 présente les étapes successives d'identification et de validation des ressources de production, garantissant la cohérence entre la machine, le processus et l'opérateur.

La figure 2.9 présente l'état décrit quant à elle la gestion du produit scanné, l'évaluation de son état (conforme, défaut ou autre), ainsi que l'enregistrement des informations correspondantes dans la base de données. Cette structuration en deux parties permet de distinguer clairement la phase d'association des entités de production et la phase de suivi du produit et de ses états.

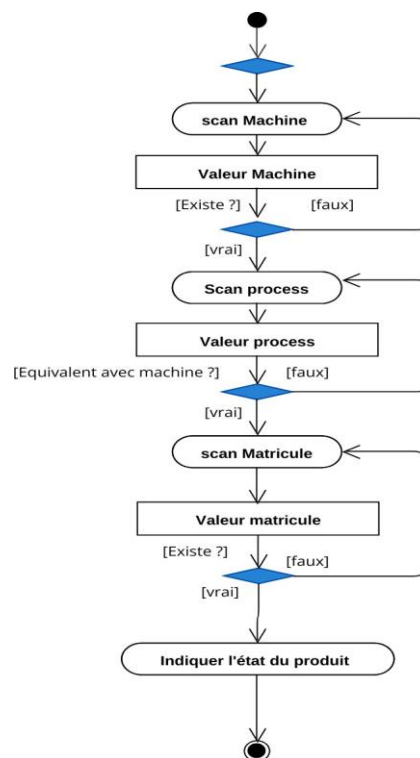


FIGURE 2.8. Diagramme d'activités "Les étapes de la traçabilité"

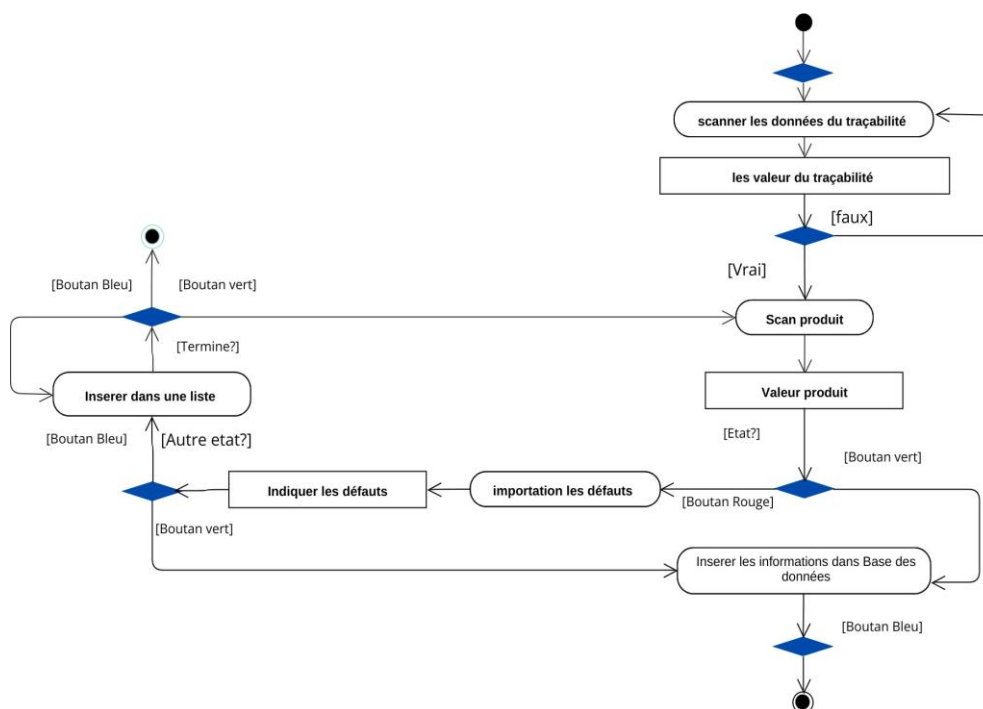


FIGURE 2.9. Diagramme d'activités "Déclarer l'état de production"

2.4.5 Diagramme de classes

La figure 2.10 présente le diagramme de classes global du système, mettant en évidence les différentes classes, leurs attributs, leurs méthodes et leurs associations.

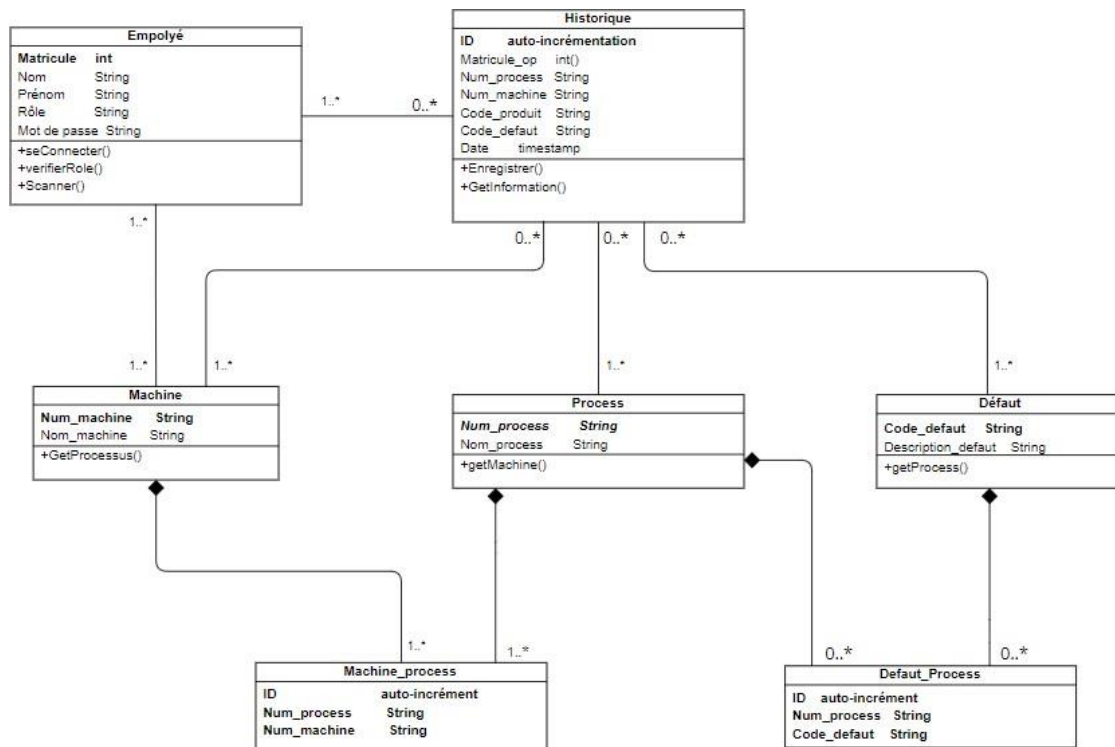


FIGURE 2.10. Diagramme de classes

Le système est constitué des classes Employé, Machine, Process, Défaut et Historique. La classe Historique conserve les informations concernant les opérations de traçabilité. Les relations plusieurs-à-plusieurs entre les classes Machine et Process et entre Process et Défaut sont matérialisées par les classes d'association Machine-process et Default-Process. Cette modélisation permet d'associer plusieurs machines à un même processus et plusieurs processus à une même machine, et assure une gestion flexible et cohérente des données.

2.4.6 Structure de la base de données

Dans cette partie, nous décrivons la conception de la base de données du système de traçabilité et de suivi de production. Le modèle relationnel de la base de données se définit ainsi :

- Employé (**Matricule**, nom, prénom, rôle, MDP)
- Défaut (**code-defaut**, description-defaut)
- Machine (**num-machine**, nom-machine)
- Process (**num-process**, nom-process)
- Machine-Process (**ID**, #num-process, #num-machine)
- Defaut-Process (**ID**, #num-process, #code-defaut)
- Historique (**ID**, #Matricule, code-produit, #num-process, #num-machine, #code-defaut, date)

2.5 Conclusion

Dans ce chapitre, nous avons vu la conception de notre système grâce aux diagrammes de cas d'utilisation, de séquences, d'activités et de classes. Le but du chapitre est de présenter la réalisation de notre projet.

Configuration et installation du système

3.1 Introduction

Dans ce chapitre, nous allons voir concrètement comment mettre en place notre système de traçabilité de production.

3.2 Description de l'architecture matérielle du système :

Pour fonctionner de manière fluide et efficace, le système de traçabilité repose sur une architecture matérielle bien pensée. Plusieurs composants essentiels, soigneusement sélectionnés et assemblés, assurent son bon fonctionnement.

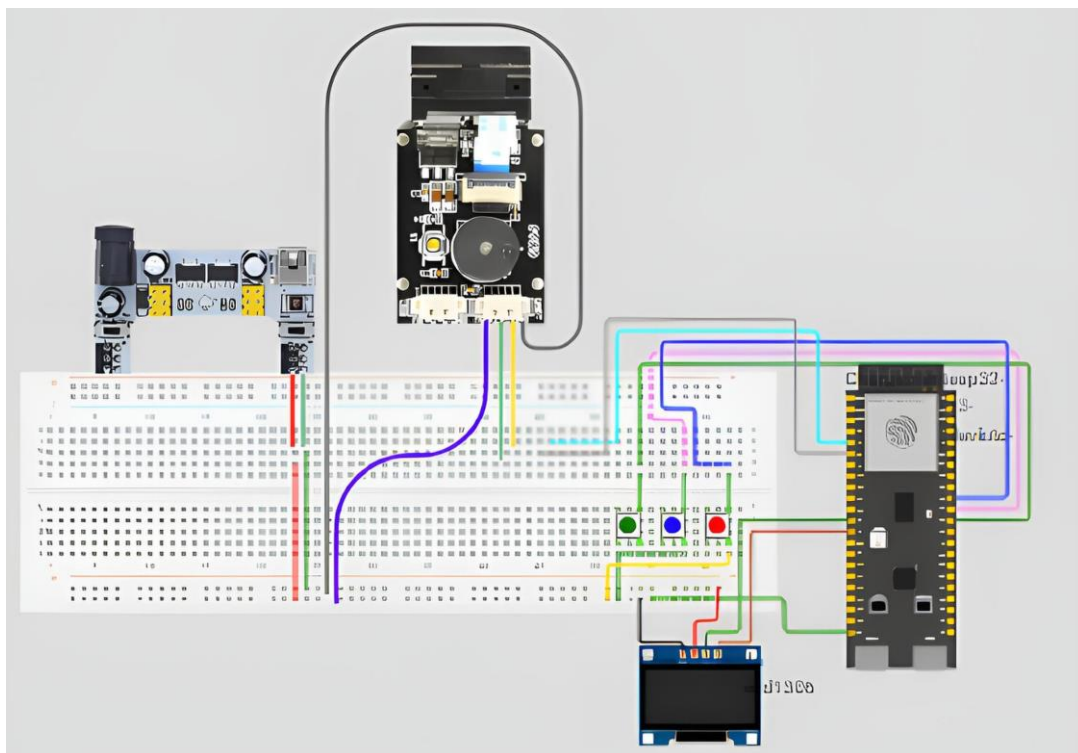


FIGURE 3.1. Montage de système

3.3 Description des composants du système de traçabilité

Le système de traçabilité repose sur une carte ESP32 et un lecteur de codes QR. Les principaux composants utilisés sont présentés dans le tableau 3.1 suivant :

Composant	Rôle principal	Exemple d'utilisation
ESP32-S3	Microcontrôleur central	Gère les opérations de scan et transmet les données
Scanner GM65	Identification	Associe machine, processus, opérateur et produit
Boutons physiques (rouge, vert, bleu)	Interaction opérateur	Déclarer un défaut, valider un produit, revenir en arrière
Écran OLED	Affichage	Présente les messages d'erreur et l'état du système
Protocole MQTT	Communication légère	Assure une transmission rapide et fiable des données
Serveur Node-RED	Traitement des flux	Automatise et achemine les informations collectées
Base MySQL	Stockage centralisé	Conserve l'historique complet de production et de traçabilité

TABLE 3.1. Composants du système de traçabilité industrielle basé sur IoT

3.4 Composants matériels

3.4.1 ESP32

L'ESP32 [12] est un microcontrôleur SoC (System on a Chip), développé par Espressif Systems. Il inclut le Wi-Fi et le Bluetooth, ce qui le rend adapté aux applications IoT. Il a été choisi pour notre projet en raison de sa facilité d'utilisation, de sa faible consommation énergétique et de sa capacité à communiquer en temps réel avec le serveur grâce au protocole MQTT.



FIGURE 3.2. Carte ESP32-S3-DevKitC-1

3.4.1.1 Caractéristiques

- Processeur à double cœur : Le microprocesseur à double cœur est accompagné d'un coprocesseur à ultra-basse consommation qui prend le relais.
- Connectivité sans fil : Wi-Fi 802.11 b/g/n et Bluetooth 4.2 et 5.0, assurant une communication sans fil fiable pour les projets IoT.
- Mémoire intégrée : Intègre 520 KO de SRAM qui peut être utilisée pour le stockage du programme et des données.

3.4.2 Scanner GM65

Le module GM65 [13] est un lecteur de codes-barres et de QR-codes rapide et fiable. Il est capable de lire les codes imprimés sur papier ou affichés à l'écran, même dans des conditions de faible luminosité. Nous l'avons choisi pour notre projet en raison de sa rapidité de lecture et de sa bonne précision, ce qui permet une identification fiable des produits.



FIGURE 3.3. Modèle du scanner GM65

3.4.2.1 Caractéristiques

- Lecture 1D et 2D : Code 128, Code 39, QR-Code, Data Matrix.
- Capteur : CMOS 640 × 480 px
- Temps de décodage : < 100 ms
- Interface : UART TTL (3,3 V / 5 V)
- Tension d'alimentation : de 3,3 V à 5 V

3.4.3 MB102 Breadboard Power Supply Module

Le MB102 [15] est un petit module d'alimentation destiné aux breadboards. Il permet d'alimenter facilement un montage électronique. Il se connecte directement sur la breadboard et peut être alimenté soit par un adaptateur secteur à l'aide d'un connecteur Jack, soit par un port USB.



FIGURE 3.4. Alimentation modèle MB102

3.4.3.1 Caractéristiques

- Entrée : Jack DC (7–12 V) ou USB (5 V).
- Sorties régulées : 3.3 V (AMS1117-3.3) et 5 V (AMS1117-5.0).
- Compatible : 3,3 V / 5 V selon les besoins du montage.

3.4.4 Écran OLED

L'écran OLED [16] est un écran compact offrant un affichage clair et une faible consommation d'énergie. Dans notre projet, il est utilisé pour afficher les informations et les messages liés au fonctionnement du système, afin de faciliter l'interaction avec l'utilisateur.



FIGURE 3.5. OLED SSD1306

3.4.4.1 Caractéristiques

- Le contrôleur SSD1306 conçu par Adafruit possède une résolution d'affichage de 128×64 pixels.
- La tension d'alimentation varie entre 3,3 V et 5 V en courant continu.

3.4.5 Les boutons

Les boutons [17] sont des interrupteurs qui permettent à l'utilisateur d'exécuter différentes commandes afin d'interagir avec le système. Dans notre projet, chaque bouton possède une fonction précise pour faciliter l'utilisation du système.



FIGURE 3.6. Bouton poussoir

3.5 Environnement de développement

Le système de traçabilité repose sur plusieurs outils logiciels utilisés pour son développement, son déploiement et sa supervision.

3.5.1 Développement du système embarqué

3.5.1.1 Visual Studio Code

Visual Studio Code (VS Code) [26] est un éditeur de code léger et performant. Il a été utilisé pour développer et gérer l'ensemble du code grâce à ses nombreuses extensions et à sa prise en charge de plusieurs langages.



FIGURE 3.7. Logo Visual Studio Code [27]

3.5.1.2 PlatformIO

PlatformIO [18] est un environnement de développement destiné aux systèmes embarqués. Dans notre projet, il sert à programmer la carte ESP32, à compiler le code, et à simplifier le développement et le téléversement des applications.



FIGURE 3.8. Logo de PlatformIO

3.5.1.3 C++

C++ [31] est un langage de programmation rapide et performant. Il est utilisé pour programmer l'ESP32 et gérer la lecture des données du scanner ainsi que leur transmission.

3.5.1.4 JSON

JSON [35] est un format léger d'échange de données. Dans notre projet, il est utilisé pour transmettre les données entre l'ESP32, le backend et l'interface web.

3.5.2 Développement de la plateforme IoT

3.5.2.1 Node-RED

Node-RED [19] est un outil de programmation visuelle fondé sur les flux, utilisé dans les applications IoT. Dans notre projet, il sert à traiter les données reçues de l'ESP32, à les envoyer à la base de données et à assurer la communication entre les différents éléments du système.



FIGURE 3.9. Logo Node-RED

3.5.2.2 Mosquitto

Mosquitto [21] est un courtier MQTT qui permet la communication entre les divers équipements du système. Il reçoit les messages des ESP32 et les transmet aux applications abonnées, assurant un transfert de données fiable et en temps réel.



FIGURE 3.10. Logo Mosquitto

3.5.2.3 Wokwi

Wokwi [23] est une plateforme en ligne permettant de concevoir et de simuler des projets électroniques. Elle a été utilisée dans le projet pour tester le montage avant de passer à la version réelle. Il manque quelques composants, mais la simulation reste assez proche du système que l'on voulait.



FIGURE 3.11. Logo WOKWI

3.5.2.4 MySQL

MySQL [28] est un système de gestion de bases de données relationnelles. Dans notre projet, il est utilisé pour stocker les données de traçabilité. Nous avons utilisé MySQL Server pour gérer la base de données et MySQL Workbench [29] pour sa conception et son administration.



FIGURE 3.12. Logo MySQL

3.5.2.5 Node.js

Node.js [20] est un environnement d'exécution JavaScript côté serveur. Dans notre projet, il est utilisé pour développer le backend, gérer les API et assurer la communication entre l'interface web, Node-RED et la base de données MySQL, afin de garantir un traitement rapide et fiable des données.



FIGURE 3.13. Logo Node.js

3.5.3 Développement de l'interface Web

3.5.3.1 React.js

React.js [30] est une bibliothèque JavaScript utilisée pour créer des interfaces web. Dans notre projet, elle a servi à développer l'interface de supervision pour afficher les données de traçabilité, les rapports et les alertes.

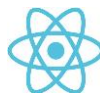


FIGURE 3.14. Logo React.js

3.5.3.2 JavaScript

JavaScript [32] est un langage de programmation utilisé principalement pour rendre les pages web interactives. Dans notre projet, il est utilisé pour l'interface web et pour créer des flux dans Node-RED afin de traiter les données et gérer les échanges entre les composants du système.



FIGURE 3.15. Logo JavaScript

3.5.3.3 CSS

CSS [33] est un langage informatique utilisé pour mettre en forme des documents XML et HTML. Nous utilisons ce langage pour concevoir une interface moderne et compréhensive.



FIGURE 3.16. Logo CSS

3.5.3.4 HTML

HTML (HyperText Markup Language) [34] est un langage de balisage à l'intention de structurer le contenu des pages web. Il permet de créer les éléments de l'interface utilisée.

L'HTML permet de créer la structure de notre interface web.



FIGURE 3.17. Logo HTML

3.6 Configuration de l'environnement de travail

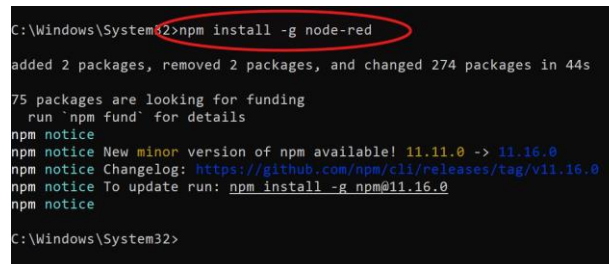
3.6.1 Configuration des logiciels utilisés

Configurer les logiciels permet de les adapter aux besoins réels du projet et de les intégrer sans accroc au système.

3.6.1.1 Configuration du Node-RED

L'installation des palettes dans Node-RED [19] permet d'ajouter facilement les modules nécessaires, tels que MQTT, MySQL ou Dashboard, afin d'élargir les fonctionnalités de l'environnement.

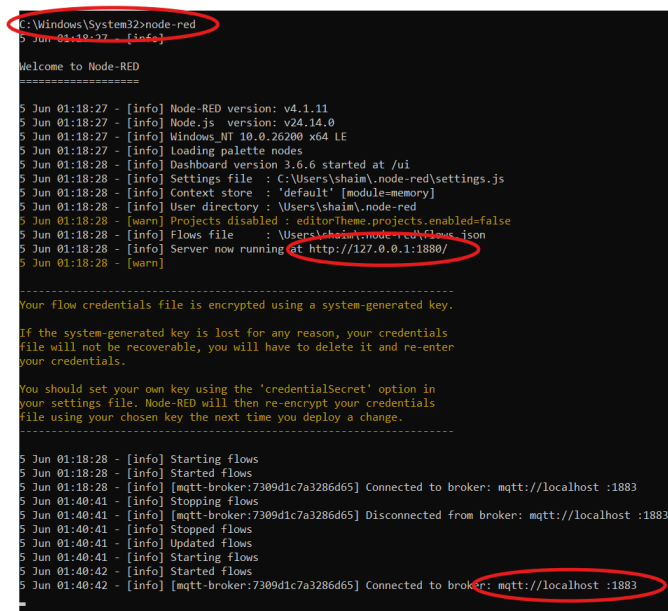
1. Installer Node-RED à partir de Node.js afin d'avoir l'interface de développement de flux à partir des cmds montrés dans les figures suivantes :



```
C:\Windows\System32>npm install -g node-red
added 2 packages, removed 2 packages, and changed 274 packages in 44s
75 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New minor version of npm available! 11.11.0 -> 11.16.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.16.0
npm notice To update run: npm install -g npm@11.16.0
npm notice
C:\Windows\System32>
```

FIGURE 3.18. Validation de l'installation de Node-RED

2. Lancer le serveur, démarrer Node-RED et accéder à l'éditeur dans le navigateur à l'adresse " `http://127.0.0.1:1880` ".



```
C:\Windows\System32>node-red
5 Jun 01:18:27 - [info]
Welcome to Node-RED
=====
5 Jun 01:18:27 - [info] Node-RED version: v4.1.11
5 Jun 01:18:27 - [info] Node.js version: v24.14.0
5 Jun 01:18:27 - [info] Windows_NT 10.0.26280 x64 LE
5 Jun 01:18:27 - [info] Loading palette nodes
5 Jun 01:18:28 - [info] Dashboard version 3.6.6 started at /ui
5 Jun 01:18:28 - [info] Settings file : C:\Users\shaim\.node-red\settings.js
5 Jun 01:18:28 - [info] Context store : 'default' [module=memory]
5 Jun 01:18:28 - [info] User directory : \Users\shaim\.node-red
5 Jun 01:18:28 - [warn] Projects disabled : editorTheme.projects.enabled=false
5 Jun 01:18:28 - [info] Flows file : \Users\shaim\.node-red\flows.json
5 Jun 01:18:28 - [info] Server now running at http://127.0.0.1:1880/
5 Jun 01:18:28 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----

5 Jun 01:18:28 - [info] Starting flows
5 Jun 01:18:28 - [info] Started flows
5 Jun 01:18:28 - [info] [mqtt-broker:7309d1c7a3286d65] Connected to broker: mqtt://localhost:1883
5 Jun 01:40:41 - [info] Stopping flows
5 Jun 01:40:41 - [info] [mqtt-broker:7309d1c7a3286d65] Disconnected from broker: mqtt://localhost:1883
5 Jun 01:40:41 - [info] Stopped flows
5 Jun 01:40:41 - [info] Updated flows
5 Jun 01:40:41 - [info] Starting flows
5 Jun 01:40:42 - [info] Started flows
5 Jun 01:40:42 - [info] [mqtt-broker:7309d1c7a3286d65] Connected to broker: mqtt://localhost:1883
```

FIGURE 3.19. Lancement du Node-RED

3. Installation des palettes nécessaires pour ajouter les modules requis comme MQTT, MySQL et Dashboard depuis Manage Palette.
4. Configuration du broker MQTT par l'indication de l'adresse du broker Mosquitto dans les nœuds MQTT IN/OUT.
5. Construire les flux en y ajoutant les nœuds MQTT, JSON, Function, Switch et MySQL selon la logique du projet.
6. Déployer les flux afin d'appliquer la configuration et d'activer la communication en temps réel.
7. Mettre en relation les flux de Node-RED avec votre code développé pour l'ESP32 selon la définition du topic du MQTT IN/OUT.

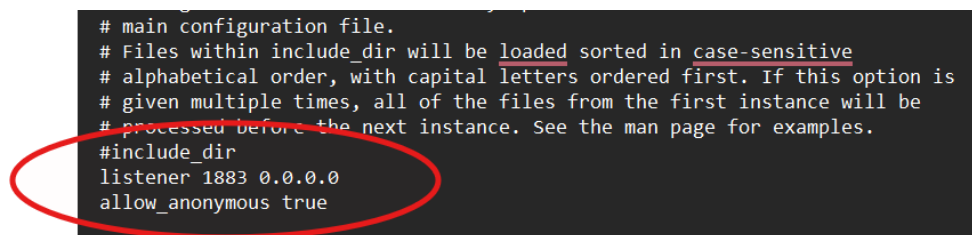
```
12 // --- MQTT Broker ---
13 #define MQTT_BROKER      "10.40.16.40" // 192.168.1.3
14 #define MQTT_PORT        1883
15 #define MQTT_CLIENT_ID   "ESP32_TRACABILITE_01"
16
17 // --- Topics ESP32 → Node-RED ---
18 #define TOPIC_MACHINE_SCAN "machine/scan"
19 #define TOPIC_PROCESS_SCAN "process/scan"
20 #define TOPIC_OPERATEUR_SCAN "operateur/scan"
21 #define TOPIC_PRODUIT_SCAN "code_produit/ok"
22 #define TOPIC_DEFAULT_REQUEST "default/request"
23 #define TOPIC_NOK_INSERT "code_produit/nok"
24
25 // --- Topics Node-RED → ESP32 ---
26 #define TOPIC_MACHINE_OK "esp32/machine/response"
27 #define TOPIC_PROCESS_OK "esp32/process/ok"
28 #define TOPIC_MATRICULE_OK "esp32/matricule/ok"
29 #define TOPIC_PRODUIT_OK "esp32/produitok/insert"
30 #define TOPIC_DEFAULT_RESP "default/response"
31 #define TOPIC_INSERT_CONFIRM "esp32/insert/confirm"
32
```

FIGURE 3.20. Relation de Node-RED avec ESP32

3.6.1.2 Configuration du Mosquitto

La configuration de Mosquitto [22] permet d'installer le broker MQTT qui deviendra le point central d'échange de messages entre les différents équipements.

1. Installation du fichier d'installation correspondant au PC utilisé.
2. Création du fichier de configuration "mosquitto.conf" et ajout des lignes affichées dans la figure 3.21.



```
# main configuration file.
# Files within include_dir will be loaded sorted in case-sensitive
# alphabetical order, with capital letters ordered first. If this option is
# given multiple times, all of the files from the first instance will be
# processed before the next instance. See the man page for examples.
#include_dir
listener 1883 0.0.0.0
allow_anonymous true
```

FIGURE 3.21. Des commandes pour configurer le fichier "mosquitto.conf"

3. Lancer Mosquitto avec terminale
 - On accède au dossier contenant les fichiers de Mosquitto.
 - On utilise la commande cmd "mosquitto.exe -c mosquitto.conf -v " pour Windows pour le lancer.

Lorsque Mosquitto est en marche, on peut commencer à configurer Node-RED pour gérer et visualiser les données MQTT.

3.6.2 Configuration les composants de travail

Le scanner GM65 est configuré en mode UART TTL pour communiquer avec l'ESP32 [14].

3.6.2.1 Configuration scanner GM65 avec l'ESP32-S3 en mode UART

On a d'abord tous besoin du câble USB du scanner pour valider les étapes.

1. Activer le mode configuration [13] : cette étape donne accès au GM65 pour ouvrir le port pour réaliser les configurations nécessaires.
2. Réinitialisation de l'usine [13] : remettre le scanner à ses paramètres d'usine.
3. Configurer la communication d'interface UART [13] : Faire l'activation du port UART pour envoyer les données vers TX/RX.
4. Configuration du baudrate UART [13] : faire le réglage de la vitesse de transmission série aux valeurs équivalentes à celles de l'ESP32.

5. Enregistre les paramètres modifiés [13] : faire l'enregistrement des configurations obtenues dans la mémoire interne du scanner .

3.7 Réalisation matérielle

3.7.1 Simulation de câblage d'ESP32 avec Wokwi

On a appliqué la simulation suivante :

- Scanner GM65 relié à ESP32 par GPIO5 et GPIO6, 5 V et GND d'alimentation.
- Écran OLED, les pins SDA et SCL reliés à l'ESP32 par GPIO8, GPIO9, 3.3V et GND.
- Bouton rouge relié à l'ESP32 par GPIO12 et GND d'alimentation.
- Bouton vert relié à ESP32 par GPIO13 et GND d'alimentation.
- Bouton bleu relié à l'ESP32 par GPIO11 et GND d'alimentation.
- Les pins de VSS et GND reliés aux bornes d'alimentation.

3.7.2 Montage de câblage d'ESP32 du système sur prototype réel

Dans la section précédente, après avoir décrit les outils du montage et le câblage du prototype, chaque composant a été monté et testé individuellement :

- Le scanner GM65 a été validé à l'aide du moniteur série.
- L'afficheur OLED a été testé en affichant des messages de contrôle.
- Les boutons ont été vérifiés sur leurs broches GPIO.

Tous les composants ont été rassemblés autour de l'ESP32-S3, conformément au schéma de câblage présenté précédemment, pour constituer le prototype complet.

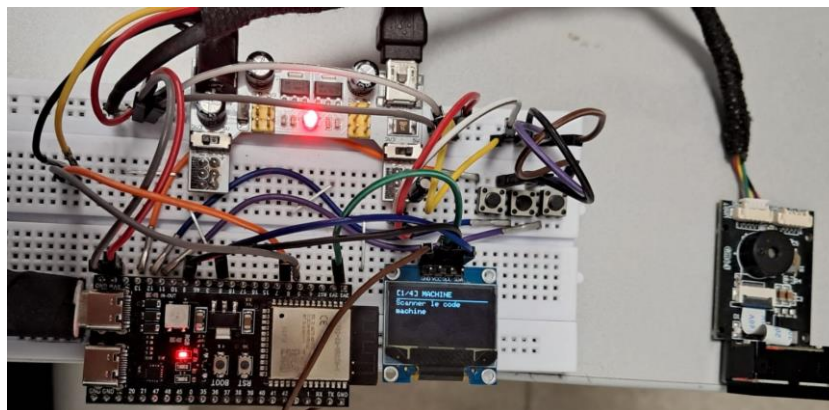


FIGURE 3.22. Montage expérimental du prototype

3.8 Test pratique :

Les tests d'intégration ont permis de valider le bon fonctionnement de la chaîne complète :

3.8.1 Lecture et afficher les données QR code depuis le scanner

L'écran OLED valide la lecture des données du QR code et du code-barres aussi en affichant la valeur au niveau de l'OLED. Comme le présente la figure 3.23, les données scanner et leur affichage.



Figure 3.23. Test affichage OLED

3.8.2 Importation des données d'après la base de données

Le bouton rouge, est le principal composant qui identifie l'état du produit scanné. Si un défaut est détecté, on appuie sur le bouton, Node-RED reçoit l'ordre d'importer les défauts équivalents. Comme l'indique la figure 3.24, l'affichage de la liste des défauts sur l'OLED.



FIGURE 3.24. Importation de la liste des défauts

3.8.3 Insérer les données dans la base de données

Après avoir terminé de recueillir les informations requises, quand on appuie sur le bouton vert, on valide l'insertion des données dans notre base de données. La figure 3.25 représente le message de la validation de l'insertion des données au niveau de la base de données.

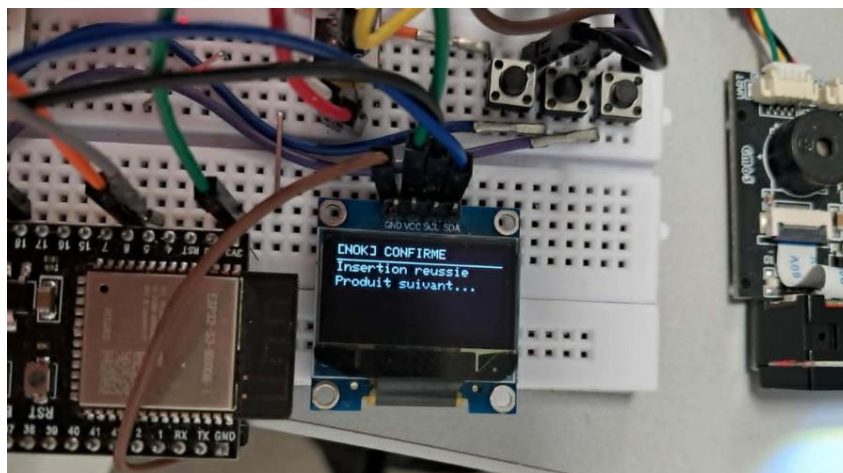
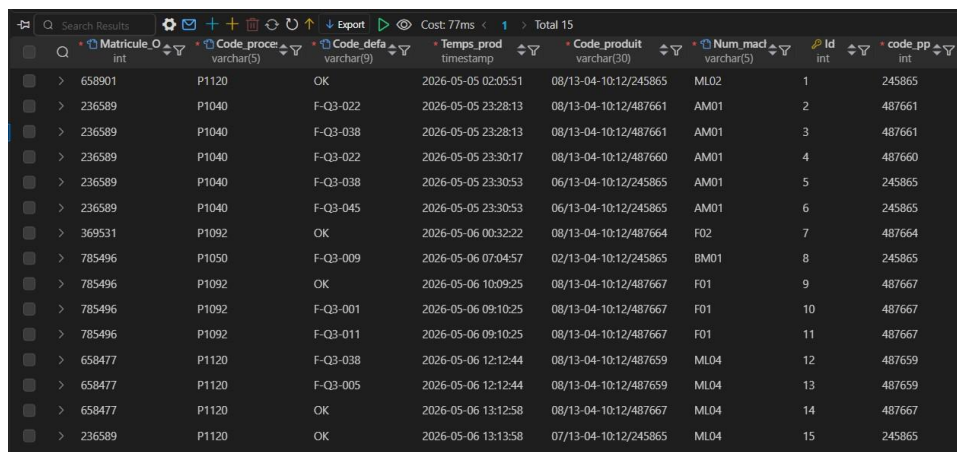


FIGURE 3.25. Message OLED "insertion réussie".

La figure 3.26 montre l'insertion des données dans la base de données.



Matricule_O int	Code_proce varchar(5)	Code_defa varchar(9)	Temps_prod timestamp	Code_produit varchar(30)	Num_mad varchar(5)	Id int	code_pp int
> 658901	P1120	OK	2026-05-05 02:05:51	08/13-04-10:12/245865	ML02	1	245865
> 236589	P1040	F-Q3-022	2026-05-05 23:28:13	08/13-04-10:12/487661	AM01	2	487661
> 236589	P1040	F-Q3-038	2026-05-05 23:28:13	08/13-04-10:12/487661	AM01	3	487661
> 236589	P1040	F-Q3-022	2026-05-05 23:30:17	08/13-04-10:12/487660	AM01	4	487660
> 236589	P1040	F-Q3-038	2026-05-05 23:30:53	06/13-04-10:12/245865	AM01	5	245865
> 236589	P1040	F-Q3-045	2026-05-05 23:30:53	06/13-04-10:12/245865	AM01	6	245865
> 369531	P1092	OK	2026-05-06 00:32:22	08/13-04-10:12/487664	F02	7	487664
> 785496	P1050	F-Q3-009	2026-05-06 07:04:57	02/13-04-10:12/245865	BM01	8	245865
> 785496	P1092	OK	2026-05-06 10:09:25	08/13-04-10:12/487667	F01	9	487667
> 785496	P1092	F-Q3-001	2026-05-06 09:10:25	08/13-04-10:12/487667	F01	10	487667
> 785496	P1092	F-Q3-011	2026-05-06 09:10:25	08/13-04-10:12/487667	F01	11	487667
> 658477	P1120	F-Q3-038	2026-05-06 12:12:44	08/13-04-10:12/487659	ML04	12	487659
> 658477	P1120	F-Q3-005	2026-05-06 12:12:44	08/13-04-10:12/487659	ML04	13	487659
> 658477	P1120	OK	2026-05-06 13:12:58	08/13-04-10:12/487667	ML04	14	487667
> 236589	P1120	OK	2026-05-06 13:13:58	07/13-04-10:12/245865	ML04	15	245865

FIGURE 3.26. Test l’insertion des données

3.9 Conclusion

Ce chapitre présente toutes les étapes nécessaires à la configuration et à l’installation des composants et des logiciels du système de traçabilité. Pour cela, on va voir le résultat de cette configuration, on applique les procédures nécessaires au travail.

Réalisation

4.1 Introduction

Dans ce chapitre, nous verrons le fonctionnement des étapes mentionnées dans les chapitres premier et deuxième, liées à une interface web dont l'accès est basé sur le rôle de l'opérateur.

4.2 Réalisation les étapes de production

4.2.1 Établissement de la communication

4.2.1.1 Connexion Wi-Fi

Au démarrage, l'ESP32 se connecte au réseau Wi-Fi à l'aide du SSID et du mot de passe configurés. Une fois la connexion établie, une adresse IP lui est automatiquement attribuée, confirmant sa connexion au réseau. La figure 4.1 présente le message de connexion au Wi-Fi, tandis que la figure 4.2 montre le code utilisé pour établir cette connexion.



FIGURE 4.1. Message OLED « connexion Wi-Fi »

```
// — WiFi —  
#define WIFI_SSID      "chhaima_"  
#define WIFI_PASSWORD  "20031925"
```

FIGURE 4.2. Code de configuration Wi-Fi de l'ESP32.

4.2.1.2 Connexion MQTT

Une fois connecté au réseau Wi-Fi, l'ESP32 se connecte au broker MQTT grâce à son adresse IP et au port de communication (1883). Une fois cette connexion réussie, le système est prêt à échanger les messages MQTT avec les autres composants. La figure 4.3 présente le message OLED affiché pour la connexion MQTT. La figure 4.4 montre le code utilisé pour la configuration de la connexion MQTT.



FIGURE 4.3. Message OLED « Connexion MQTT »

```
// — MQTT Broker —  
#define MQTT_BROKER    "10.40.X.X" // 192.168.X.X  
#define MQTT_PORT      1883  
#define MQTT_CLIENT_ID "ESP32_TRACABILITE_01"
```

FIGURE 4.4. Configuration du broker MQTT dans le code source

Un message de confirmation, la figure 4.5, est alors affiché sur l'écran OLED pour informer l'opérateur que le système est prêt à effectuer la lecture des codes QR.



FIGURE 4.5. Système prêt et démarrage effectif

4.2.2 Étape de scannage de la machine

À cette étape, l'opérateur scanne le code QR de la machine à l'aide du scanner connecté à l'ESP32. Les données sont ensuite envoyées à Node-RED via le protocole MQTT pour vérification, puis un message de validation ou d'erreur est renvoyé.

4.2.2.1 Cas d'erreur

Ce cas présente la condition, si le numéro de la machine n'existe pas dans la base de données, le système affiche sur l'écran OLED le message d'erreur « Machine introuvable ». La figure 4.6 suivante présente ce message.



FIGURE 4.6. Affichage de l'erreur « Machine introuvable »

4.2.2.2 Cas de validation

Si le numéro de la machine existe dans la base de données, le système le valide automatiquement puis demande à l'opérateur de scanner le code QR du processus



FIGURE 4.7. Vérification du numéro de machine

Nous avons développé ce flux Node-RED pour gérer cette étape de scannage, la vérification de l'existence dans la base de données ainsi que le traitement des cas d'erreur et la validation.

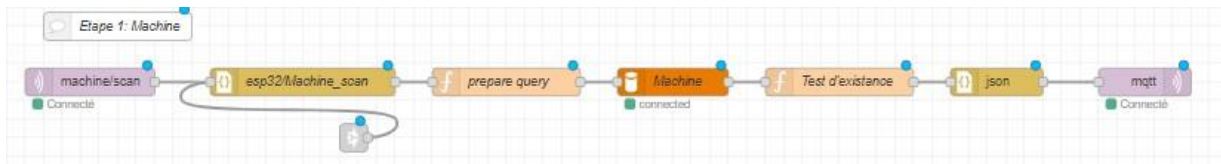


FIGURE 4.8. Traitement Node-RED de la machine

4.2.3 Étape de scannage du processus

4.2.3.1 Cas d'erreur

Le système va vérifier la correspondance de la valeur du processus avec le code de la machine scanner. On a un message d'erreur affiché au niveau de l'OLED pour indiquer à l'opérateur qu'il y a une faute de scannage.



FIGURE 4.9. Affichage de l'erreur « Processus introuvable »

4.2.3.2 Cas de validation

C'est le contraire du cas précédent, si la valeur de scannage correspond à la valeur de code machine au niveau de la base de données, il passe automatiquement à l'étape suivante.



FIGURE 4.10. Vérification du processus scanné

Le flux Node-RED de la figure 4.11 présentée permet d'assurer la vérification des données du processus et le traitement automatique des cas d'erreur et de validation.

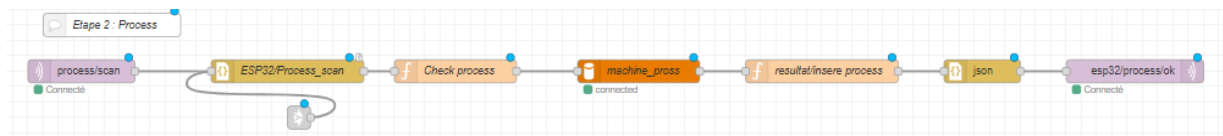


FIGURE 4.11. Traitement Node-RED du processus

4.2.4 Étape de scannage du matricule

4.2.4.1 Cas d'erreur

Dans ce cas, c'est presque comme la première étape, sauf qu'on fait le test avec le matricule de l'opérateur. La figure 4.12 affiche le message affiché au niveau de l'OLED.



FIGURE 4.12. Affichage de l'erreur « Opérateur introuvable »

4.2.4.2 Cas de validation

On vérifie l'existence du matricule et son type d'emploi actif, qui indique le rôle d'employé (responsable ou opérateur de production).



FIGURE 4.13. Vérification du matricule de l'opérateur

On présente la figure 4.14, le flux de Node-RED pour vérifier des données de l'opérateur, ainsi que le traitement des cas d'erreur, même pour la validation.

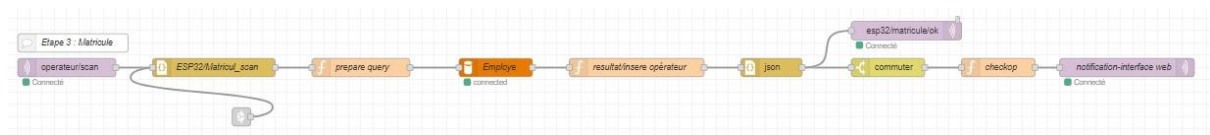


FIGURE 4.14. Traitement Node-RED du matricule

4.2.5 Étape de scannage du code produit

Dans cette étape, l'opérateur scanne le code de produit et l'écran OLED affiche le résultat, il choisit ensuite l'état du produit à l'aide des boutons disponibles.

4.2.5.1 Identifier l'état du produit

Cette étape consiste à scanner le code du produit. Les données sont ensuite envoyées au système afin de vérifier la validité du code et d'assurer le suivi du produit tout au long de la production.

1. Cas OK (Produit conforme) : si l'opérateur appuie sur le bouton vert, le produit est enregistré dans la base de données avec l'état « OK ».

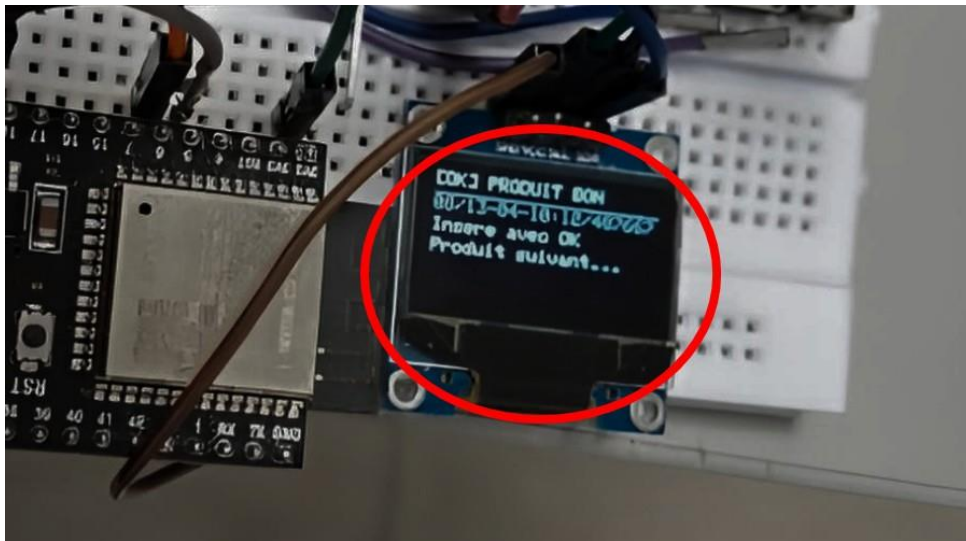


FIGURE 4.15. Confirmation de l'enregistrement des données

Comme présenté dans la figure 4.16, le flux de Node-RED et la validation.

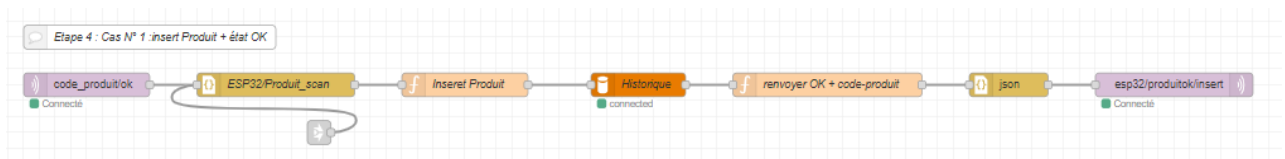


FIGURE 4.16. Traitement Node-RED de l'insertion de l'état IO

2. Cas NIO (Produit non conforme) : Le deuxième choix, bouton rouge équivalent au choix de NIO, donne la demande de liste des codes de défaut équivalents au numéro de process.



FIGURE 4.17. Affichage la liste des défauts

La figure 4.18 de flux Node-RED présente l'exécution d'une étape.

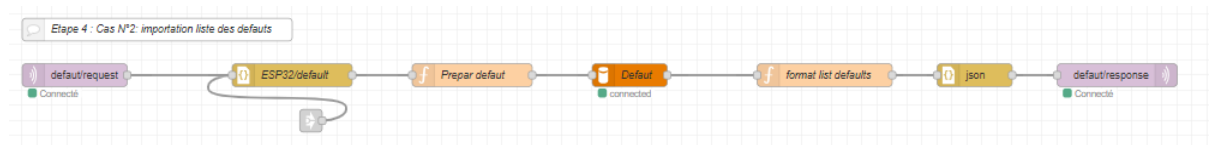


FIGURE 4.18. Traitement Node-RED de l'importation de la liste des défauts

Afin de choisir les défauts, le Node-RED permet d'insérer et de valider l'insertion avec un message envoyé à l'ESP32, comme elle l'a indiqué à la figure 4.19.



FIGURE 4.19. Affichage de la confirmation d'insertion de l'état NIO

La figure 4.20 suivante présente le flux Node-RED utilisé pour valider et insérer les informations dans la base de données.

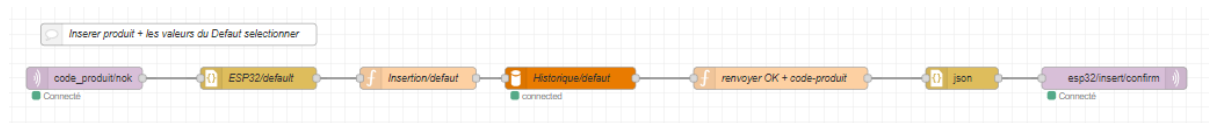


FIGURE 4.20. Traitement Node-RED de l'insertion des défauts sélectionnés

Après la validation de toutes les étapes, le système revient automatiquement à la lecture du code produit. L'opérateur peut également revenir à l'étape précédente à l'aide du bouton bleu. La figure 4.21 présente le flux Node-RED qui gère cette fonctionnalité

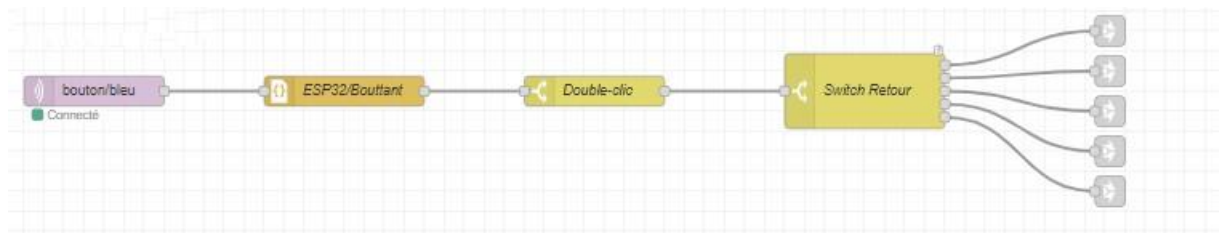


FIGURE 4.21. Traitement Node-RED de la gestion du bouton bleu

4.3 Réalisation de l'interface web

Une interface web a été développée pour permettre le suivi des données de traçabilité, la visualisation des résultats et la supervision du système en temps réel. Elle est organisée en deux niveaux d'accès.

4.3.1 Interface Web d'opérateur

L'interface dédiée à l'opérateur a été conçue pour répondre aux besoins du terrain, en lui offrant un accès rapide et structuré aux informations essentielles à son activité quotidienne.

4.3.1.1 Onglet Accueil

Dès la connexion, l'opérateur est accueilli par un tableau de bord (onglet Accueil) qui affiche en temps réel les indicateurs clés de sa journée :

- Le nombre de produits scannés.
- Le nombre de défauts détectés.
- La machine la plus active.
- Un diagramme circulaire représente le taux de défauts en pourcentage.

Le tableau de bord est complété par plusieurs sections analytiques :

- Défauts les plus fréquents : montre les erreurs rencontrées dans le déroulement actuel de la journée.
- Défauts par processus et défauts par machine : permettent à l'opérateur de repérer plus vite la source de ces anomalies détectées.
- Défauts par heure : Donne une vue horaire de la distribution des erreurs durant chaque poste.

Deux encarts complémentaires sont également disponibles en permanence :

- L'utilisateur permet d'afficher les informations nécessaires équivalentes au matricule saisi.
- Les notifications : des alertes envoyées par l'ESP32 vers l'interface.
- Les annonces : des informations envoyées par l'administrateur.

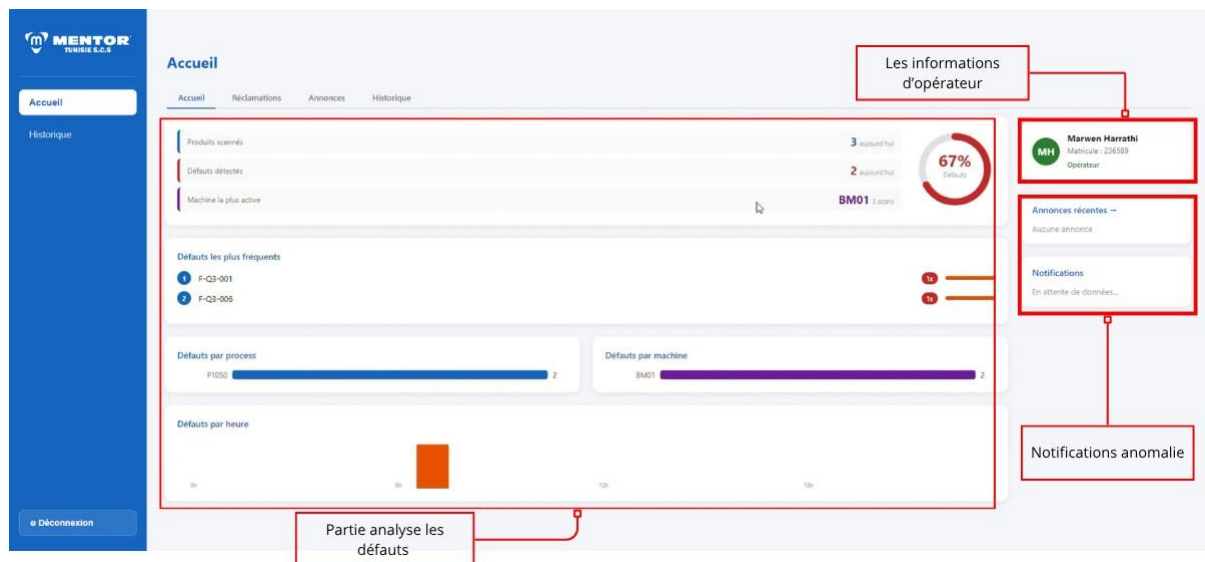


FIGURE 4.22. Page d'accueil de l'espace opérateur

L'opérateur accède à plusieurs sous-onglets qui lui permettent d'envoyer des réclamations à l'administrateur et de recevoir des annonces. La figure 4.23 ci-dessus présente les structures d'onglet d'accueil.

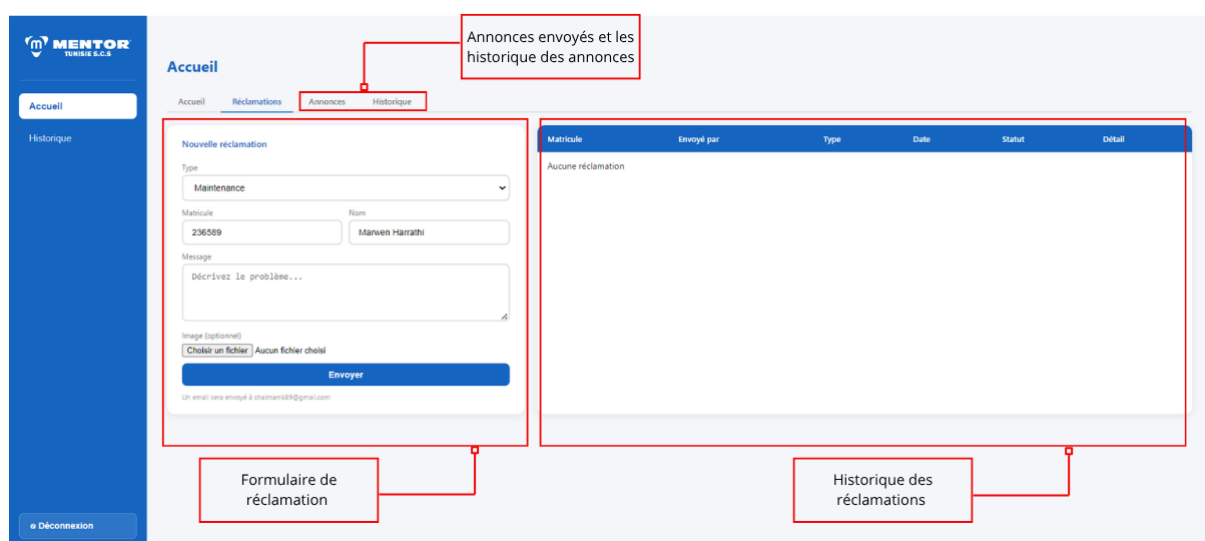
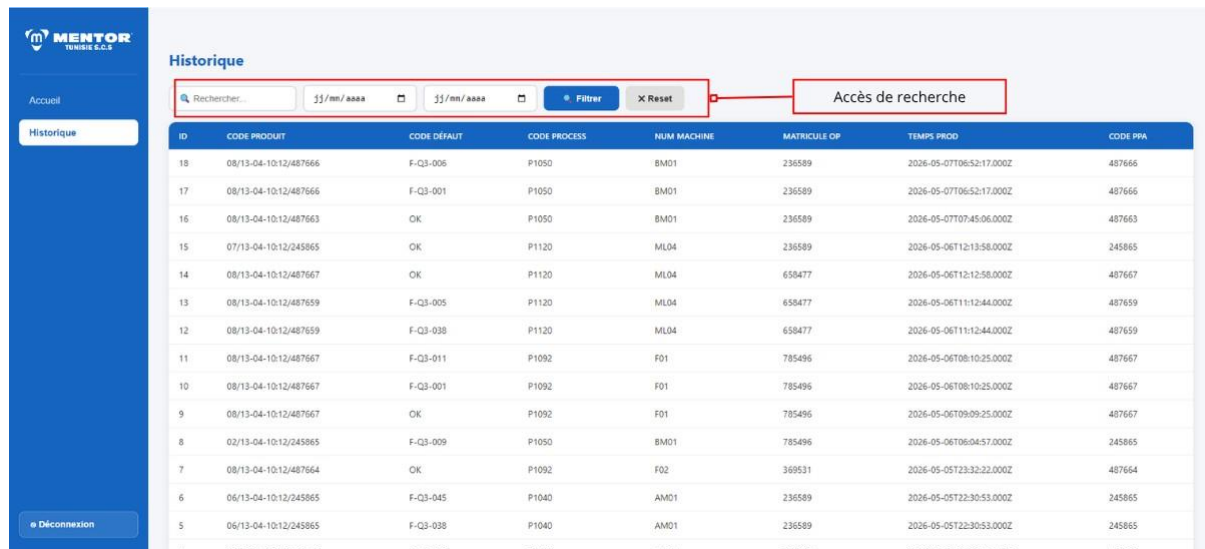


FIGURE 4.23. Consultation des données de traçabilité dans l'espace opérateur

4.3.1.2 Onglet historique

Cet onglet Historique lui offre la possibilité de consulter les enregistrements passés relatifs à ses activités et aux défauts précédemment signalés. Comme présenté dans la figure 4.24.



Historique

Rechercher...

Accès de recherche

ID	CODE PRODUIT	CODE DÉFAUT	CODE PROCESS	NUM MACHINE	MATRICULE OP	TEMPS PROD	CODE PPA
18	08/13-04-10:12/487666	F-Q3-006	P1050	BM01	236589	2026-05-07T06:52:17.000Z	487666
17	08/13-04-10:12/487666	F-Q3-001	P1050	BM01	236589	2026-05-07T06:52:17.000Z	487666
16	08/13-04-10:12/487663	OK	P1050	BM01	236589	2026-05-07T07:45:06.000Z	487663
15	07/13-04-10:12/245865	OK	P1120	ML04	236589	2026-05-06T12:13:58.000Z	245865
14	08/13-04-10:12/487667	OK	P1120	ML04	658477	2026-05-06T12:12:58.000Z	487667
13	08/13-04-10:12/487659	F-Q3-005	P1120	ML04	658477	2026-05-06T11:12:44.000Z	487659
12	08/13-04-10:12/487659	F-Q3-038	P1120	ML04	658477	2026-05-06T11:12:44.000Z	487659
11	08/13-04-10:12/487667	F-Q3-011	P1092	F01	785496	2026-05-06T08:10:25.000Z	487667
10	08/13-04-10:12/487667	F-Q3-001	P1092	F01	785496	2026-05-06T08:10:25.000Z	487667
9	08/13-04-10:12/487667	OK	P1092	F01	785496	2026-05-06T09:09:25.000Z	487667
8	02/13-04-10:12/245865	F-Q3-009	P1050	BM01	785496	2026-05-06T06:04:57.000Z	245865
7	08/13-04-10:12/487664	OK	P1092	F02	369531	2026-05-05T23:32:22.000Z	487664
6	06/13-04-10:12/245865	F-Q3-045	P1040	AM01	236589	2026-05-05T22:30:53.000Z	245865
5	06/13-04-10:12/245865	F-Q3-038	P1040	AM01	236589	2026-05-05T22:30:53.000Z	245865

FIGURE 4.24. Consultation de l'historique des produits dans l'espace opérateur

4.3.2 Interface Web d'administrateur

L'interface administrateur permet de superviser et de gérer l'ensemble du système. La barre de navigation latérale donne accès à tous les modules de l'application.

4.3.2.1 Onglet accueil

Cet onglet permet de gérer les éléments du système en temps réel. Cela indique différentes sous-onglets pour bien structurer :

- Accueil : affiche une vue d'ensemble de l'état du système (Figure 4.25).
- Analyse des défauts : présente les défauts enregistrés sous forme de tableau de bord, la figure 4.26.
- Envoyer : permet à l'administrateur d'envoyer des annonces ou des réclamations aux utilisateurs.
- Réclamations reçues : regroupe les réclamations envoyées par les opérateurs et les administrateurs, voir la figure 4.27.

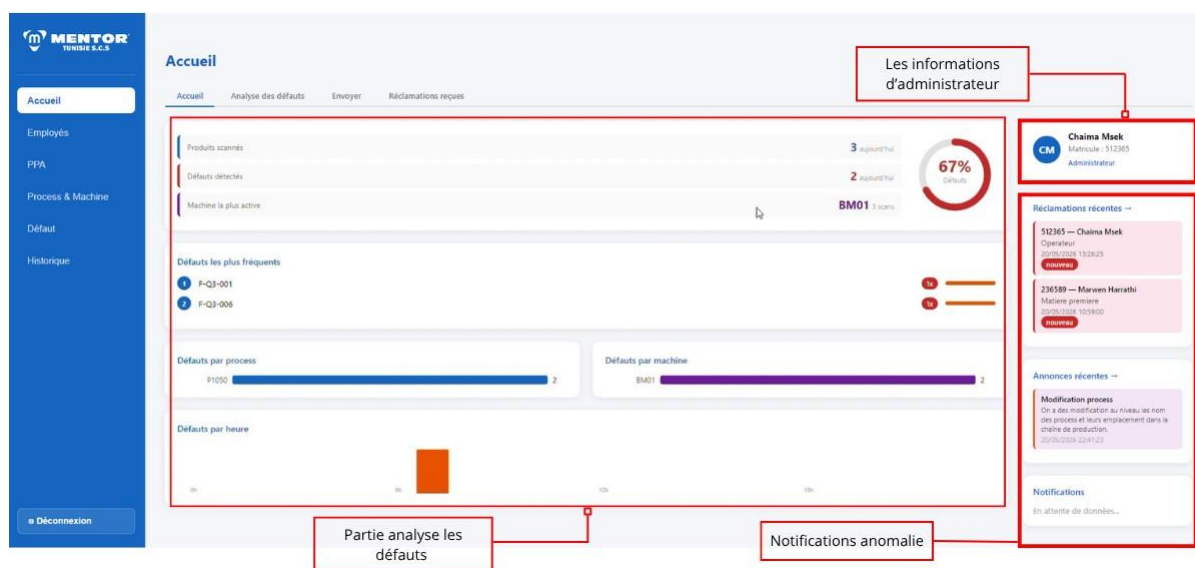


FIGURE 4.25. Page d'accueil de l'espace administrateur

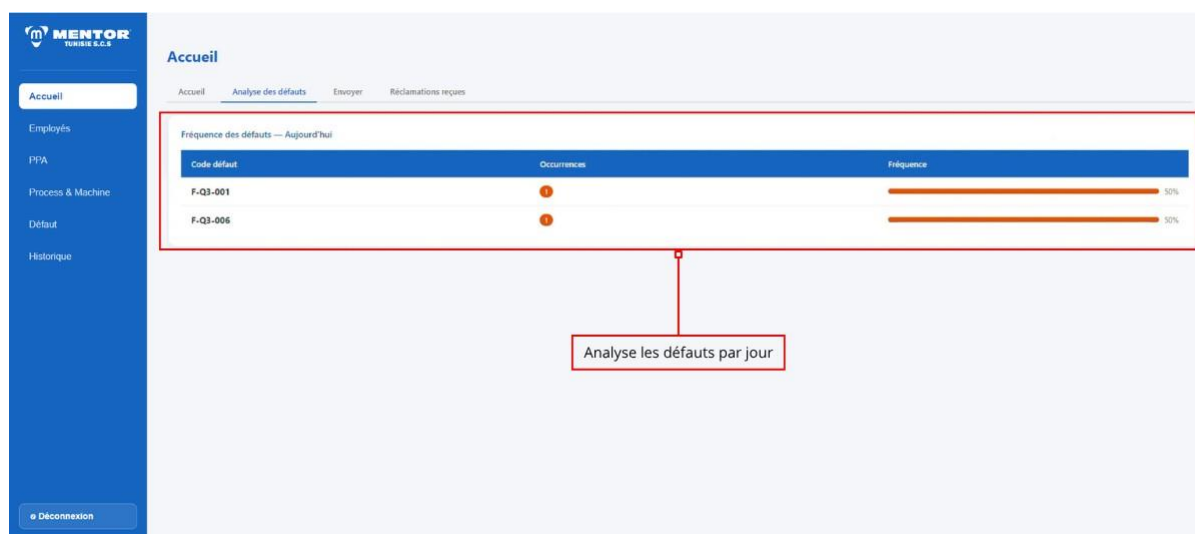


FIGURE 4.26. Consultation et analyse des défauts dans l'espace administrateur

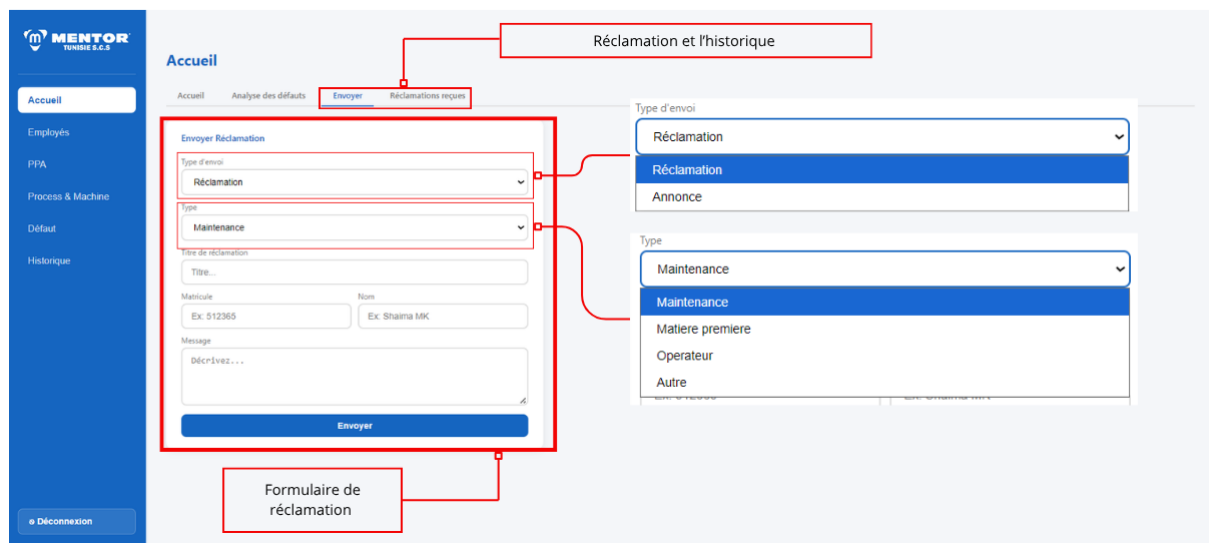


FIGURE 4.27. Gestion des réclamations dans l'espace administrateur

4.3.2.2 Onglet Employés

Cet onglet permet à l'administrateur de gérer les comptes des employés et leurs droits d'accès au système. Il offre les fonctionnalités suivantes :

- Ajouter un nouvel employé en renseignant son matricule, son nom, son prénom et son rôle.
- Modifier les informations d'un employé.
- Supprimer un employé ou retirer son accès au système.

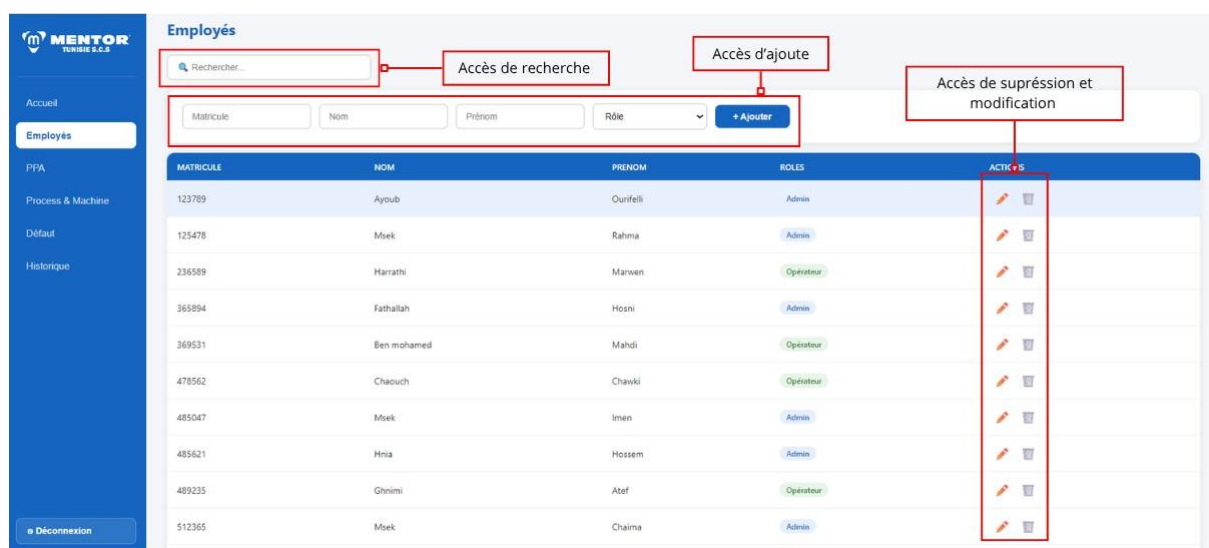


FIGURE 4.28. Gestion des employés

4.3.2.3 Onglet PPA (Plan de Production et d'Articles)

Cet onglet permet à l'administrateur de lancer une production en saisissant le numéro de contrat PPA et la quantité à fabriquer. Le système génère automatiquement un ticket contenant un code produit unique et un code QR pour assurer la traçabilité des produits. L'administrateur peut également imprimer les tickets ou les réimprimer si nécessaire (la figure 4.29).

Un tableau « Valeurs PPA » affiche l'historique des lancements avec le numéro de contrat PPA, la quantité et la date (Figure 4.30).

The screenshot shows the PPA management interface. On the left is a sidebar with the MENTOR TUNISIE S.C.S. logo and navigation links. The main area is titled 'PPA'. It features a 'déclarer un PPA' section with input fields for 'Code PPA (6 chiffres)' and 'Quantité', and a 'Générer' button. Below this is a table 'Codes Produits — PPA 789156' with columns 'CODE PPA', 'CODE PRODUIT', and 'QR CODE'. To the right is a 'Valeurs PPA' section with a search bar 'Recherche selon la date' and a table showing PPA values. Red boxes and arrows highlight key elements: 'déclarer un PPA', 'Recherche selon la date', 'Historique de PPA déclaré', and 'Liste PPA'.

CODE PPA	QUANTITE	DATE
458962	40	07/05/2026 11:54
789156	10	07/05/2026 08:58
458216	11	06/05/2026 09:43

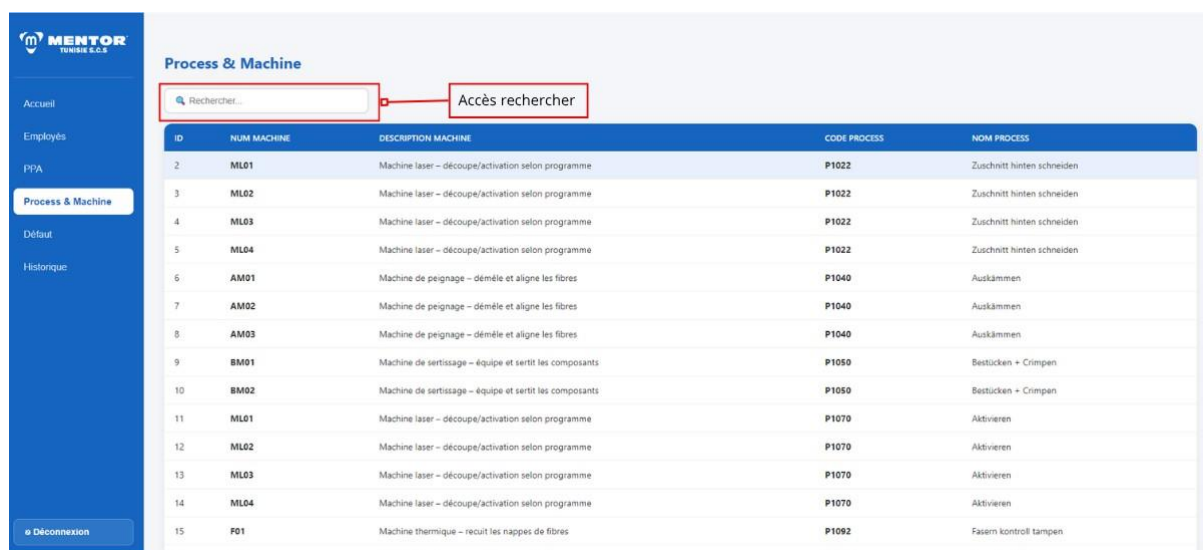
FIGURE 4.29. Gestion des codes PPA



FIGURE 4.30. Modèle des tickets imprimés

4.3.2.4 Onglet Processus et Machine

L'onglet Processus et Machine permet à l'administrateur de consulter les associations entre les machines et les processus de production. Un même processus peut être réalisé sur plusieurs machines, et une même machine peut être utilisée pour plusieurs processus. L'onglet affiche les informations de chaque machine et du processus correspondant, avec des fonctions de recherche et de filtrage pour faciliter la consultation des données.



ID	NUM MACHINE	DESCRIPTION MACHINE	CODE PROCESS	NOM PROCESS
2	ML01	Machine laser – découpe/activation selon programme	P1022	Zuschnitt hinten schneiden
3	ML02	Machine laser – découpe/activation selon programme	P1022	Zuschnitt hinten schneiden
4	ML03	Machine laser – découpe/activation selon programme	P1022	Zuschnitt hinten schneiden
5	ML04	Machine laser – découpe/activation selon programme	P1022	Zuschnitt hinten schneiden
6	AM01	Machine de peignage – démêle et aligne les fibres	P1040	Auskämmen
7	AM02	Machine de peignage – démêle et aligne les fibres	P1040	Auskämmen
8	AM03	Machine de peignage – démêle et aligne les fibres	P1040	Auskämmen
9	BM01	Machine de sertissage – équipe et sertit les composants	P1050	Bestücken + Crimpen
10	BM02	Machine de sertissage – équipe et sertit les composants	P1050	Bestücken + Crimpen
11	ML01	Machine laser – découpe/activation selon programme	P1070	Aktivieren
12	ML02	Machine laser – découpe/activation selon programme	P1070	Aktivieren
13	ML03	Machine laser – découpe/activation selon programme	P1070	Aktivieren
14	ML04	Machine laser – découpe/activation selon programme	P1070	Aktivieren
15	F01	Machine thermique – recuit les nappes de fibres	P1092	Fasern kontrolltampfen

FIGURE 4.31. Gestion des processus et des machines

4.3.2.5 Onglet Défaut

L'onglet Défaut permet à l'administrateur de gérer les défauts détectés pendant la production. Un même défaut peut être associé à plusieurs processus, et un même processus peut contenir plusieurs défauts. L'administrateur peut :

- Ajouter un nouveau défaut en indiquant le code du processus, le code du défaut et sa description.
- Supprimer un défaut.
- Rechercher un défaut à l'aide de la barre de recherche.

La figure 4.32 représente la structure de cet onglet.

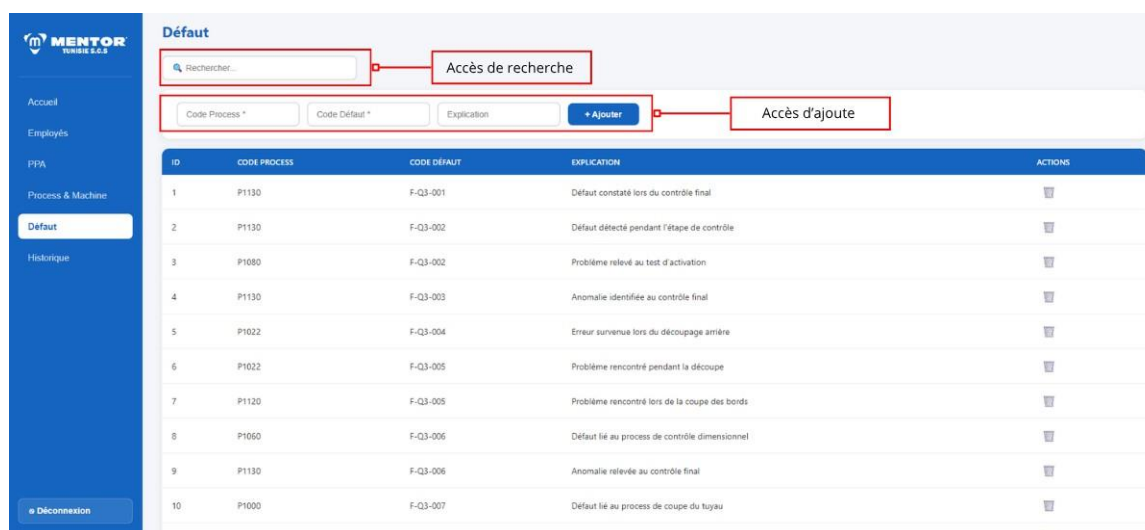


FIGURE 4.32. Gestion des Défauts

4.3.2.6 Onglet Historique

L'onglet Historique permet à l'administrateur de consulter les données de traçabilité des produits. Il affiche les informations de chaque scan, telles que le code produit, le statut, le processus, la machine, l'opérateur, la date, l'heure et le code PPA. Des fonctions de recherche et de filtrage facilitent la consultation des données. La figure 4.33 présente cette interface.

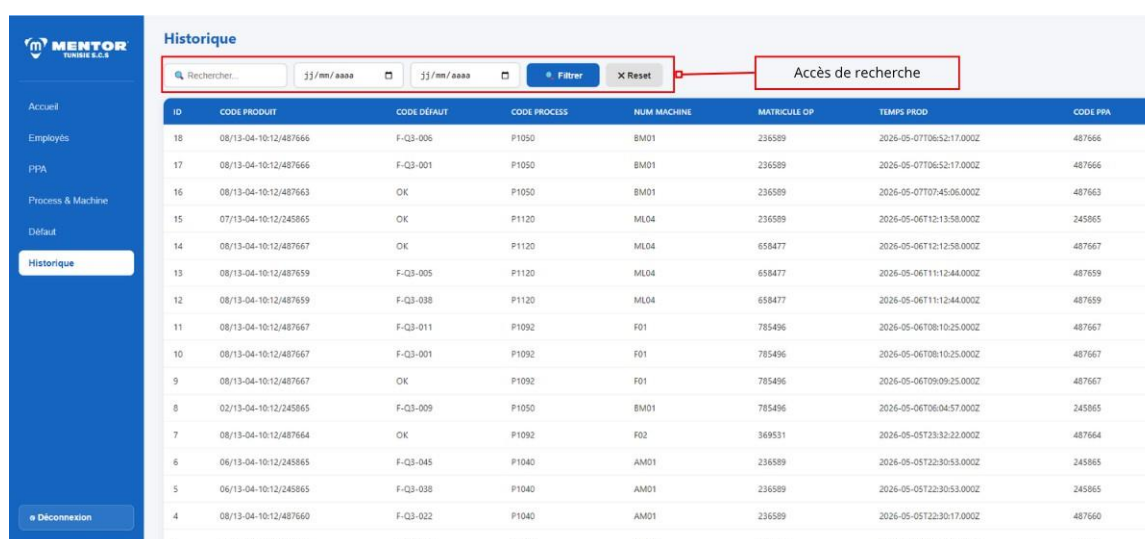


FIGURE 4.33. Consultation de l'historique des produits

4.4 Conclusion

Dans ce chapitre, nous avons présenté la réalisation du système de traçabilité, les outils utilisés, les flux Node-RED développés ainsi que les interfaces web mises en place. Ces éléments ont permis de concrétiser la solution proposée et d'assurer le suivi des données de production.

Conclusion générale

Ce projet intervient dans le cadre de la digitalisation industrielle et de la mise en place d'un système de traçabilité de production en temps réel. Nous avons conçu une solution reposant sur les technologies IoT, avec une carte ESP32, le protocole MQTT et une interface web de supervision, pour garantir un suivi fiable et en temps réel des produits durant tout le processus de production.

Une analyse des besoins a d'abord été menée, puis le système a été conçu à l'aide de diagrammes UML. La phase de réalisation a ensuite permis d'intégrer les différents composants matériels et logiciels, notamment la base de données MySQL, le scanner GM65, l'écran OLED et l'interface de supervision web. Enfin, des tests ont confirmé le bon fonctionnement du système et ont vérifié que les objectifs fixés ont été atteints. Afin d'améliorer davantage cette solution, plusieurs perspectives peuvent être envisagées :

- Améliorer l'architecture du système afin d'assurer une communication encore plus fiable ;
- Intégrer de nouveaux capteurs pour enrichir les données collectées ;
- Renforcer la sécurité des échanges et la gestion des accès utilisateurs ;
- Héberger les données sur une plateforme Cloud pour améliorer leur stockage et leur disponibilité ;
- Interconnecter le système avec les solutions MES et Proalpha afin d'assurer une digitalisation plus complète du processus de production.

En conclusion, ce projet est une étape décisive dans la transformation numérique de l'entreprise. Il permet d'améliorer la traçabilité des produits, la qualité du suivi de la production, la réactivité aux anomalies, dans le respect des principes de l'Industrie 4.0

Bibliographie

[1] Mentor Tunisie SCS. Disponible sur :

<https://www.mentor.de.com/fr/>

Consulté le 23 février 2026.

[2] Fasermatten Automobile. Disponible sur :

<https://www.mentor.de.com/fr/portfolio/automobile/>

Consulté le 23 février 2026.

[3] Système COOX (vidéo explicative). Disponible sur :

<https://industry.codra.net/fr/solutions/plateforme-coox-origin-mes-caracteristi>

Consulté le 28 février 2026.

[4] COOX image système. Disponible sur :

<https://codra.net/fr/service/formation/coox-mes/>

Consulté le 28 février 2026.

[5] Cimag Production. Disponible sur :

<https://www.alpha3i.com/cimag-production/>

Consulté le 1er mars 2026.

[6] Mapex Trace - logiciel de traçabilité industrielle. Disponible sur :

<https://mapex.io/fr/logiciel-tracabilite-industrielle/>

Consulté le 2 mars 2026.

[7] UML (Unified Modeling Language). Disponible sur :

<https://www.omg.org/uml/>

Consulté le 20 mars 2026.

[8] Méthodologie en cascade. Disponible sur :

<https://blog-gestion-de-projet.com/modele-en-cascade/>

Consulté le 12 mars 2026.

[9] Internet of Things (IoT). Disponible sur :

https://en.wikipedia.org/wiki/Internet_of_things

Consulté le 21 mars 2026.

[10] MQTT Protocol. Disponible sur :

<https://mqtt.org/>

Consulté le 21 mars 2026.

[11] PlantUML. Disponible sur :

<https://plantuml.com/>

Consulté le 23 mars 2026.

[12] ESP32-S3 DevKitC-1. Disponible sur :

<https://docs.espressif.com/projects/esp-dev-kits/en/latest/esp32s3/esp32-s3-devkitc-1/index.html>

Consulté le 1er avril 2026.

[13] Scanner GM65 - User Manual. Disponible sur :

<https://www.electroniccomp.com/image/catalog/>

GM65-Barcode-reader-module-User-Manual.pdf Consulté le 1er avril 2026.

[14] Scanner GM65 - User Manual. Disponible sur :

<https://youtu.be/RvjQS2qyEw?si=29GY6x7o-QCIEMQl>

Consulté le 1er avril 2026.

[15] MB102 Module d'alimentation. Disponible sur :

<https://v2.ttrinfo.be/articles/robotique/electronique/breadboards/mb-102/mb-102.pdf>

Consulté le 1er avril 2026.

- [16] Écran OLED SSD1306. Disponible sur :
<https://makerspace-amiens.fr/electronic-design/docs/tutorial/esp32-sd1306-oled/>
Consulté le 1er avril 2026.
- [17] Bouton électrique. Disponible sur :
[https://fr.wikipedia.org/wiki/Bouton_\(%C3%A9lectricit%C3%A9\)](https://fr.wikipedia.org/wiki/Bouton_(%C3%A9lectricit%C3%A9))
Consulté le 1er avril 2026.
- [18] PlatformIO. Disponible sur :
<https://platformio.org/>
Consulté le 7 avril 2026.
- [19] Node-RED. Disponible sur :
<https://fr.wikipedia.org/wiki/Node-RED>
Consulté le 7 avril 2026.
- [20] Node.js installation. Disponible sur :
<https://nodejs.org/en/download>
Consulté le 7 avril 2026.
- [21] Eclipse Mosquitto. Disponible sur :
<https://mosquitto.org/>
Consulté le 7 avril 2026.
- [22] Installation Mosquitto Windows. Disponible sur :
<https://mosquitto.org/files/binary/win64/mosquitto-2.1.2-install-windows-x64.exe>
Consulté le 7 avril 2026.
- [23] Wokwi Simulator. Disponible sur :
<https://wokwi.com/>
Consulté le 10 avril 2026.

- [24] Architecture client-serveur. Disponible sur :
<https://www.geeksforgeeks.org/system-design/client-server-model/>
Consulté le 20 avril 2026.
- [25] Architecture publish/subscribe. Disponible sur :
<https://learn.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber>
Consulté le 20 avril 2026.
- [26] Visual Studio Code. Disponible sur :
https://code.visualstudio.com/download?_exp_download=fb315fc982
Consulté le 12 avril 2026.
- [27] Installation de VS Code. Disponible sur :
<https://code.visualstudio.com/>
Consulté le 12 avril 2026.
- [28] MySQL. Disponible sur :
<https://www.oracle.com/fr/mysql/what-is-mysql/>
Consulté le 13 avril 2026.
- [29] Téléchargement de MySQL Workbench. Disponible sur :
<https://dev.mysql.com/downloads/workbench/>
Consulté le 13 avril 2026.
- [30] React.js. Disponible sur :
<https://fr.legacy.reactjs.org/>
Consulté le 13 avril 2026.
- [31] C/C++. Disponible sur :
<https://fr.wikipedia.org/wiki/C%2B%2B>
Consulté le 14 avril 2026.

- [32] JavaScript. Disponible sur :
<https://fr.wikipedia.org/wiki/JavaScript>
Consulté le 14 avril 2026.
- [33] CSS. Disponible sur :
https://fr.wikipedia.org/wiki/Feuilles_de_style_en_cascade
Consulté le 14 avril 2026.
- [34] HTML. Disponible sur :
https://fr.wikipedia.org/wiki/Hypertext_Markup_Language
Consulté le 14 avril 2026.
- [35] JSON format. Disponible sur :
<https://www.json.org/json-en.html>
Consulté le 14 avril 2026.