REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

الجمهورية الجزائرية الديموقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE EN SCIENCES APPLIQUEES
TLEMCEN

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
تلمسان

Mémoire de fin d'étude

Pour l'obtention du diplôme de Ingéniorat

Filière : Automatique

Spécialité : Automatique

Présenté par : Ahmed Ghezal

Mohammed Nehila

Thème

Conception d'un système IoT de contrôle et de supervision d'une chaîne de production pilotée par un automate programmable industriel

Soutenu publiquement, le 21 juin 2026, devant le jury composé de :

Dr. Megnafi Hicham	MCA	ESSA-Tlemcen	Encadrant
Dr. Karaouzene Zoheir	MCB	ESSA-Tlemcen	Co-Encadrant
Dr. Abdi Sidi Mohamed	MCA	ESSA-Tlemcen	Président du jury
Dr. ARICHI Fayçal	MCA	ESSA-Tlemcen	Examinateur 1
Dr. Alouane Mohammed Amine		EMP-Alger	Examinateur 2
Pr. Zidi Salah	Professeur	ISSIG Gabes-Tunisie	Invité

Année Universitaire : **2025 / 2026**

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Dédicaces

"To my dear parents "

Ghezal Ahmed
12/06/2026

"In the name of Allah, the Most Gracious, the Most Merciful.
First and foremost, I dedicate this work to Allah Almighty, whose blessings, guidance, and mercy have given me the strength, patience, and perseverance to complete this Engineering project.
I dedicate this work to the memory of my beloved mother. May Allah grant her His infinite mercy and the highest place in Jannah. Her love, sacrifices, and prayers continue to inspire me every day, and her memory remains forever in my heart.
To my father, for his support, guidance, and the values he has instilled in me throughout my life.
To my dear brothers, Farouk and Ayoub, whose encouragement and support have always motivated me to move forward.
To my project partner, GHEZAL Ahmed, for his cooperation, dedication, and the efforts we shared throughout the completion of this project.
To my friends Nadir, Sakina, Aziz, Ali and Bilel, thank you for your friendship, encouragement, and the memorable moments we shared during this academic journey.
To all my classmates in the Automation Engineering program, with whom I shared years of learning, challenges, teamwork, and unforgettable experiences.
I would also like to dedicate this work to myself, in recognition of the effort, determination, resilience, and commitment invested in achieving this important milestone.
Finally, I dedicate this work to everyone who supported me, directly or indirectly, throughout my studies and the realization of this project.
With sincere gratitude and respect ".

NEHILA Mohammed
12/06/2026

Remerciements :

Avant toute chose, je tiens à exprimer ma profonde gratitude à Dieu Tout-Puissant, pour m'avoir accordé la force, la patience et la persévérance nécessaires tout au long de ce travail. Sans Sa miséricorde et Ses bénédictions, l'accomplissement de ce mémoire n'aurait jamais été possible.

Je tiens à adresser mes plus sincères remerciements à Monsieur le Docteur Hicham MEGNAFI, mon encadreur, pour sa disponibilité, ses conseils éclairés et son accompagnement rigoureux tout au long de ce projet. Sa pédagogie, son exigence bienveillante et son suivi constant ont été déterminants dans la concrétisation de ce travail.

Je remercie également Monsieur le Docteur Zoheir KARAOUZENE, second encadreur, pour son implication, ses orientations judicieuses et l'intérêt qu'il a porté à notre travail. Ses remarques pertinentes et son soutien ont contribué à enrichir ce mémoire.

J'adresse mes sincères remerciements à Monsieur le Professeur Salah ZIDI, co-encadreur, pour sa précieuse collaboration, son expertise reconnue et son accompagnement tout au long de ce travail. Sa contribution scientifique et son regard extérieur ont apporté une dimension supplémentaire à ce mémoire, et nous lui sommes profondément reconnaissants pour le temps et l'énergie qu'il y a consacrés.

Je tiens également à exprimer ma profonde gratitude au **président du jury, Dr. Abdi Sidi Mohamed**, pour l'honneur qu'il nous a fait en acceptant de présider cette soutenance. Mes sincères remerciements s'adressent également aux **examinateurs, Dr. ARICHI Fayçal et Dr. ALOUANE Mohammed Amine**, pour le temps qu'ils ont consacré à la lecture et à l'évaluation de ce travail. Leurs remarques pertinentes, leurs observations constructives et leurs questions enrichissantes ont contribué à améliorer la qualité de ce mémoire et à approfondir notre réflexion.

Je tiens également à remercier chaleureusement Monsieur Sidi Mohammed ABDI et Monsieur Mohammed MESSIRDI, pour leur aide précieuse et leur soutien concret tout au long de la réalisation de ce projet. Leur disponibilité, leurs conseils avisés et leur implication nous ont été d'une grande valeur, tant sur le plan technique que humain.

Mes remerciements s'adressent enfin à l'ensemble des enseignants de notre formation, qui ont contribué à notre apprentissage durant ces années d'études, ainsi qu'à toutes les personnes qui, de près ou de loin, ont participé à l'aboutissement de ce travail.

Résumé :

Ce mémoire présente la conception et la réalisation d'un système IoT industriel permettant la supervision à distance de processus automatisés pilotés par des automates programmables Siemens S7-1200. Le système repose sur une passerelle Raspberry Pi 5 qui lit les données du PLC via la bibliothèque Snap7, les transmet par protocole MQTT à un serveur Flask, lequel les redistribue en temps réel vers une interface web de supervision. L'architecture a été validée sur deux processus industriels : un système virtuel de commande de pompes et un système réel de régulation de niveau sur maquette FLTP/EV. Les résultats confirment la fiabilité et l'universalité de la solution, avec une latence inférieure à une seconde, ouvrant la voie vers une supervision industrielle connectée et économique dans le cadre de l'Industrie 4.0.

Mots-clés : IoT Industriel, Supervision à Distance, Siemens S7-1200, Snap7, MQTT, Raspberry Pi, Flask, Industrie 4.0, Régulation de Niveau.

Abstract :

This thesis presents the design and implementation of an industrial IoT system enabling the remote supervision of automated processes controlled by Siemens S7-1200 programmable logic controllers. The system relies on a Raspberry Pi 5 gateway that reads PLC data via the Snap7 library, transmits it through the MQTT protocol to a Flask server, which redistributes it in real time to a web-based supervision interface. The architecture was validated on two industrial processes: a virtual pump control system and a real liquid level regulation system on an FLTP/EV test bench. The results confirm the reliability and universality of the solution, with a total latency of less than one second, paving the way for connected and cost-effective industrial supervision within the framework of Industry 4.0.

Keywords: Industrial IoT, Remote Supervision, Siemens S7-1200, Snap7, MQTT, Raspberry Pi, Flask, Industry 4.0, Level Regulation.

ملخص :

يقدم هذا المذكرة تصميم وتنفيذ نظام إنترنت الأشياء الصناعي يتيح الإشراف عن بُعد على العمليات الآلية التي تتحكم فيها وحدات التحكم المنطقية القابلة **للبرمجة** Siemens S7-1200. يعتمد النظام على **بوابة** Raspberry Pi 5 تقرأ بيانات وحدة التحكم عبر مكتبةSnap7 ، وتنقلها عبر بروتوكول MQTT إلى خادمFlask ، الذي يعيد توزيعها في الوقت الفعلي على واجهة إشراف عبر الويب. تم التحقق من صحة البنية على عمليتين صناعيتين: نظام افتراضي للتحكم في المضخات ونظام حقيقي لتنظيم مستوى السائل على منصة FLTP/EV. تؤكد النتائج موثوقية الحل وشموليته، مع زمن استجابة إجمالي أقل من ثانية واحدة، مما يفتح الطريق نحو إشراف صناعي متصل واقتصادي في إطار الصناعة 4.0.

الكلمات المفتاحية: إنترنت الأشياء الصناعي، الإشراف عن بُعد، Siemens S7-1200، Snap7، MQTT، Raspberry Pi، Flask، الصناعة 4.0، تنظيم المستوى.

TABLE DES MATIÈRES

Introduction Générale **1**

Chapitre 1 : Internet des Objets (IoT) **3**

1. Introduction et contexte général 4
2. Historique et évolution de l'IoT 4
3. Définitions et concepts fondamentaux 5
   - 3.1 Définition de l'IoT 5
   - 3.2 Les six concepts fondamentaux 5
   - 3.3 Caractéristiques fondamentales d'un système IoT 6
   - 3.4 Distinction entre IoT, M2M et systèmes cyber-physiques 6
4. Architecture générale d'un système IoT 7
   - 4.1 Le modèle en trois couches 7
   - 4.2 L'architecture en cinq couches 7
   - 4.3 L'architecture Edge Computing et Fog Computing 7
5. Les composants d'un système IoT 8
   - 5.1 Les capteurs (Sensors) 8
   - 5.2 Les actionneurs (Actuators) 10
   - 5.3 Les passerelles IoT (IoT Gateways) 10
   - 5.4 Les réseaux de communication IoT 10
6. Les protocoles de communication IoT 11
   - 6.1 Le protocole MQTT 11
   - 6.2 Le protocole HTTP/REST 12
   - 6.3 Le protocole CoAP 12
   - 6.4 Les protocoles industriels 13
7. Les plateformes IoT 14
   - 7.1 Introduction 14
   - 7.2 Node-RED 14
   - 7.3 ThingsBoard 14
   - 7.4 Grafana 15
8. L'IoT Industriel (IIoT) et l'Industrie 4.0 15
   - 8.1 L'Internet des Objets Industriel 15

8.2 L'Industrie 4.0 16
8.3 Les Cyber-Physical Systems (CPS) 16
8.4 Le Digital Twin 16
9. Sécurité dans les systèmes IoT 16
9.1 Les principales menaces 16
9.2 Les bonnes pratiques de sécurité IoT 17
10. Avantages, défis et applications de l'IoT 17
10.1 Avantages et défis 17
10.2 Applications sectorielles 18
10.3 L'IoT en Algérie et dans le monde arabe 18
11. Exemples des projets IoT dans ESSAT 19
11.1 Système de surveillance de la qualité de l'air par drone quadrotor 19
11.1.5 Résultats et apport IoT du projet 22
11.2 Système de feux de signalisation intelligents et automatisés 23
11.2.4 Réalisation matérielle et logicielle 25
11.2.5 Résultats et apport par rapport aux systèmes traditionnels 25
12. Exemples d'applications IoT 27
12.1 Surveillance médicale à distance — Philips Healthcare (2025) 27
12.2 Gestion intelligente de l'énergie urbaine — Projet NEOM Smart City (2024–2025) 27
Conclusion 28
Chapitre 2 : Automatisation des Systèmes de Production 29
1. Introduction 30
2. Le Système de Production Automatisé 30
2.1 Définition et Structure Générale 30
2.2 La Partie Opérative (PO) 31
2.3 La Partie Commande (PC) 32
2.4 La Partie Interface Homme-Machine (IHM) 33
2.5 Interactions entre les Trois Parties 34
3. Les Technologies de Commande 35
3.1 La Logique Câblée 35
3.2 La Logique Programmée 36
3.3 L'Automate Programmable Industriel (API) 36
4. La Communication dans les Systèmes Industriels 40

4.1 La Pyramide CIM 40
4.2 Les Niveaux de Communication 41
4.3 Les Principaux Protocoles Industriels 41
5. La Supervision Industrielle 43
5.1 Définition et Rôle 43
5.2 Les Fonctions d'un Système de Supervision 43
5.3 Du SCADA Classique vers la Supervision Connectée 44
6. Conclusion 46
Chapitre 3 : Conception d'un Système IoT de Supervision Industrielle Universelle 47
1. Introduction 48
2. Problématique et Objectifs 48
2.1 Problématique Générale 48
2.2 Objectifs du Système 49
2.3 Cahier des Charges Fonctionnel 50
3. Principe Général du Système 50
3.1 Idée Fondamentale du Projet 50
3.2 Principe de Fonctionnement 51
3.3 Le Flux de Données — Du Terrain vers la Supervision 51
4. Architecture Globale du Système 52
4.1 Vue d'Ensemble de l'Architecture 52
4.2 Couche Terrain — L'Automate Programmable Industriel 53
4.3 Couche Passerelle — Le Système Embarqué 55
4.4 Couche Transport — Le Protocole MQTT et le Broker 56
4.5 Couche Serveur — Traitement et Stockage des Données 58
4.6 Couche Application — L'Interface de Supervision 59
5. Les Composants Logiciels Clés 60
5.1 La Bibliothèque Snap7 60
5.2 Le Protocole MQTT dans le Contexte Industriel 62
6. Universalité et Adaptabilité du Système 65
6.1 Indépendance vis-à-vis du Processus Industriel 65
6.2 Extensibilité et Scalabilité 66
6.3 Perspectives d'Évolution 66
7. Conclusion 67

Chapitre 4 : Réalisation et Expérimentation 68
1. Introduction 69
2. Présentation de l'Environnement de Travail 69
2.1 Le Banc Didactique API-24/EV 69
2.2 La Maquette FLTP/EV 71
2.3 La Passerelle IoT — Raspberry Pi 5 74
3. Processus 1 — Commande de Pompes (Virtuel) 76
3.1 Description du Processus 76
3.2 Programme Ladder — Processus 1 (TIA Portal) 77
3.3 Code Python — Raspberry Pi 5 (Processus 1) 80
3.4 Tests et Résultats — Raspberry Pi 5 (Processus 1) 82
4. Processus 2 — Régulation de Niveau (Réel) 83
4.1 Description du Processus 83
4.2 Programme Ladder — Processus 2 (TIA Portal) 84
4.3 Code Python — Raspberry Pi 5 (Processus 2) 86
4.4 Tests et Résultats — Raspberry Pi 5 (Processus 2) 88
5. Le Serveur Applicatif 89
5.1 Description et Rôle 89
5.2 Structure du Code 89
5.3 Réception des Données MQTT 90
5.4 Transmission vers l'Interface via WebSocket 90
5.5 Tests et Résultats du Serveur 91
6. L'Interface de Supervision 91
6.1 Description Générale 91
6.2 Structure de l'Interface 92
6.3 Fonctionnalités de l'Interface 93
6.4 Résultats — Interface en Fonctionnement 93
7. Discussion et Analyse des Résultats 94
7.1 Validation de l'Architecture Globale 94
7.2 Performances du Système 94
7.3 Perspectives d'Amélioration 94
8. Conclusion 95
Conclusion Générale 96

BIBLIOGRAPHIE **98**

TABLE DES FIGURES

Figure 1.1 : Architecture en trois couches d'un système IoT (Perception, Réseau, Application) 06

Figure 1.2 : Diagramme de Venn : zones communes entre Edge Computing, Cloud Computing et Fog Computing 07

Figure 1.3 : Exemples de capteurs utilisés dans les systèmes IoT 08

Figure 1.4 : Modèle Publish/Subscribe du protocole MQTT avec Broker central 11

Figure 1.5 : Architecture IoT du système de surveillance de la qualité de l'air par drone (Perception → Réseau → Application Cloud) 20

Figure 1.6 : Chaîne de traitement algorithmique du système de feux intelligents (de la caméra à la commande des feux) 23

Figure 2.1 : Structure générale d'un système de production automatisé : la Partie Opérative (PO), la Partie Commande (PC) et la Partie Interface Homme-Machine (IHM) 33

Figure 2.2 : Architecture interne d'un automate programmable industriel (CPU, modules E/S, alimentation, communication) 36

Figure 2.3 : Cycle de fonctionnement d'un API : lecture des entrées, exécution du programme, écriture des sorties 37

Figure 2.4 : Exemple de programme en langage Ladder (LAD) 38

Figure 2.5 : Exemple de programme en langage Structured Text (ST) 38

Figure 2.6 : La pyramide CIM (Computer Integrated Manufacturing) et ses cinq niveaux hiérarchiques 39

Figure 2.7 : Évolution de la supervision industrielle : du SCADA classique vers l'architecture IoT connectée 44

Figure 3.1 : Schéma du flux de données complet — du capteur physique à l'interface de supervision 51

Figure 3.2 : Architecture complète du système IoT de supervision industrielle universelle .. 52

Figure 3.3 : Schéma des zones mémoire accessibles dans l'automate programmable S7-120053

Figure 3.4 : Rôle de la passerelle — pont entre le monde industriel et le monde IoT 55

Figure 3.5 : Organisation hiérarchique des topics MQTT 56

Figure 3.6 : Organisation des topics MQTT et rôle du broker dans l'architecture 56

Figure 3.7 : Fonctionnement interne du serveur — réception, traitement, stockage et exposition des données 57

Figure 3.8 : Interface de supervision — tableau de bord multi-processus accessible sur PC, tablette et mobile 58

Figure 3.9 : Schéma de la communication Snap7 entre la passerelle et l'automate — Port TCP 102 60
Figure 3.10 : Structure hiérarchique des topics MQTT et exemple de message JSON 62
Figure 3.11 : Rôle central du broker Mosquitto dans l'architecture de supervision 63
Figure 3.12 : Universalité du système — la même architecture appliquée à des processus industriels différents 64
Figure 4.1 : Banc didactique API-24/EV avec automate Siemens S7-1200 intégré 69
Figure 4.2 : Schéma de câblage des entrées analogiques pour la simulation du niveau — Processus 1 70
Figure 4.3 : Maquette FLTP/EV — vue générale avec ses composants 71
Figure 4.4 : Gros plan de la maquette FLTP/EV — réservoir, pompe P1, électrovanne et pressostat 72
Figure 4.5 : Raspberry Pi 5 utilisé comme passerelle IoT dans le projet 73
Figure 4.6 : Schema represente principe de fonctionnement du Processus 1 75
Figure 4.7 : Network 1 du Ladder Processus 1 — Démarrage/Arrêt et commande des pompes 77
Figure 4.8 : Network 2 du Ladder Processus 1 — Normalisation et mise à l'échelle du niveau analogique 77
Figure 4.9 : Network 3 du Ladder Processus 1 — Logique de détection des seuils de niveau 78
Figure 4.10 : Code Python du Raspberry Pi 5 — Processus 1 (partie 1 : connexion et lecture) 79
Figure 4.11 : Code Python du Raspberry Pi 5 — Processus 1 (partie 2 : publication MQTT) . 80
Figure 4.12 : Résultats des tests du Raspberry Pi 5 — Processus 1 (terminal) 81
Figure 4.13 : Network 1 et 2 du Ladder Processus 2 — Acquisition et mise à l'échelle du niveau Calcul de la mesure 84
Figure 4.14 : Network 3 du Ladder Processus 2 — Logique de commande de l'électrovanne 84
Figure 4.15 : Code Python du Raspberry Pi 5 — Processus 2 (partie 1 : connexion et lecture) 85
Figure 4.16 : Code Python du Raspberry Pi 5 — Processus 2 (partie 2 : publication MQTT) . 86
Figure 4.17 : Résultats des tests du Raspberry Pi 5 — Processus 2 (terminal) 87
Figure 4.18 : Code complet du serveur applicatif (Server.py) 89
Figure 4.19 : Interface de supervision complète — vue générale avec les deux processus et les graphiques 90

TABLE DES TABLEAUX

Tableau 1.1 : Chronologie des grandes étapes de l'évolution de l'IoT 03
Tableau 1.2 : Distinction entre IoT, M2M, CPS, IIoT et WoT 05
Tableau 1.3 : Comparaison entre Cloud Computing, Edge Computing et Fog Computing ... 07
Tableau 1.4 : Classification des capteurs IoT : grandeurs mesurées et domaines d'application 08
Tableau 1.5 : Types d'actionneurs IoT : actions produites et exemples d'applications 09
Tableau 1.6 : Comparaison des technologies de communication IoT 10
Tableau 1.7 : Niveaux de Qualité de Service (QoS) du protocole MQTT 10
Tableau 1.8 : Caractéristiques techniques du protocole MQTT 11
Tableau 1.9 : Méthodes HTTP/REST et leur usage dans les systèmes IoT 11
Tableau 1.10 : Comparaison des principaux protocoles de communication IoT 12
Tableau 1.11 : Comparaison entre les plateformes IoT : Node-RED, ThingsBoard et Grafana 14
Tableau 1.12 : Les quatre révolutions industrielles et leurs innovations clés 15
Tableau 1.13 : Avantages et défis des systèmes IoT 16
Tableau 1.14 : Capteurs embarqués sur le drone quadrotor et paramètres mesurés 19
Tableau 1.15 : Échelle de l'Indice de Qualité de l'Air (IQA) selon la méthode EPA 20
Tableau 1.16 : Composants matériels du système de feux de signalisation intelligents 24
Tableau 1.17 : Comparaison entre système de feux traditionnel et système de feux intelligent 25
Tableau 2.1 : Comparaison entre Modbus TCP, Profinet et OPC-UA 41
Tableau 3.1 : Cahier des charges fonctionnel du système de supervision 49
Tableau 3.2 : Zones mémoire de l'automate accessibles par le système de supervision 53
Tableau 3.3 : Caractéristiques techniques de la bibliothèque Snap7 59
Tableau 3.4 : Accès aux zones mémoire via Snap7 61
Tableau 3.5 : Comparaison MQTT vs HTTP dans le contexte du projet 62
Tableau 4.1 : Ressources d'entrées/sorties du banc API-24/EV 70
Tableau 4.2 : Composants de la maquette FLTP/EV utilisés dans le Processus 2 71
Tableau 4.3 : Comparaison Raspberry Pi 5 vs ESP32 73
Tableau 4.4 : Configuration logicielle du Raspberry Pi 5 74
Tableau 4.5 : Variables du Processus 1 76
Tableau 4.6 : Synthèse des résultats de test — Processus 1 82
Tableau 4.7 : Variables du Processus 2 82
Tableau 4.8 : Synthèse des résultats de test — Processus 2 88

Tableau 4.9 : Synthèse des performances du système 93

Introduction Générale

L'industrie mondiale traverse actuellement une transformation numérique profonde, portée par l'émergence de l'Internet des Objets industriel et les principes de l'Industrie 4.0. Dans ce contexte, les systèmes de production automatisés, aussi performants soient-ils, se heurtent à une limitation fondamentale : leur isolement informationnel. Les automates programmables industriels qui pilotent ces systèmes génèrent en permanence des données précieuses — états des entrées et sorties, mesures analogiques, variables de processus — mais ces données demeurent confinées dans leurs mémoires internes, inaccessibles à distance et inexploitées au-delà du réseau industriel local.

Cette réalité engendre des contraintes opérationnelles considérables : l'opérateur est contraint d'être physiquement présent pour connaître l'état du système, les interventions de maintenance restent réactives plutôt que préventives, et aucune historisation des données de production n'est disponible pour l'analyse et l'optimisation des performances. Les solutions de supervision industrielle traditionnelles, bien que fonctionnelles, sont souvent propriétaires, coûteuses et difficiles à déployer dans des installations existantes sans modifications profondes.

C'est dans ce contexte que s'inscrit le présent projet, qui vise à concevoir et réaliser une solution IoT industrielle open-source permettant de connecter des automates programmables Siemens S7-1200 existants à une infrastructure de supervision moderne, accessible à distance, sans modifier leur programme ni interrompre leur fonctionnement. L'approche adoptée repose sur le principe du retrofitting non-intrusif : une passerelle intelligente, basée sur un Raspberry Pi 5, se connecte discrètement à l'automate via le réseau Ethernet et la bibliothèque de communication Snap7, lit périodiquement ses données, et les transmet via le protocole MQTT vers un serveur applicatif qui les met à disposition d'une interface de supervision web accessible depuis n'importe quel appareil connecté.

L'originalité de cette solution réside dans son caractère universel et scalable : conçue pour être indépendante de la nature du processus industriel supervisé, elle constitue une architecture générique applicable à toute installation pilotée par un automate programmable, qu'il s'agisse d'un système de pompage, de régulation de niveau, de contrôle de température ou de tout autre processus industriel.

Pour atteindre ces objectifs, le présent mémoire est structuré en quatre chapitres :

- Le **premier chapitre** présente les fondements théoriques de l'Internet des Objets, son évolution historique, ses architectures, ses protocoles de communication et ses applications dans le contexte de l'IoT industriel et de l'Industrie 4.0.
- Le **deuxième chapitre** établit les bases théoriques de l'automatisation industrielle, en présentant la structure des systèmes de production automatisés, les technologies de commande, les réseaux de communication industriels et les systèmes de supervision.

- Le **troisième chapitre** présente la conception de l'architecture IoT de supervision industrielle développée dans ce projet, en détaillant son principe général, ses couches fonctionnelles, ses composants logiciels clés et son caractère universel.
- Le **quatrième chapitre** décrit la réalisation concrète du système et présente les résultats expérimentaux obtenus lors de sa mise en œuvre sur deux processus industriels distincts : un système virtuel de commande de pompes et un système réel de régulation de niveau de liquide.

Chapitre 1 : Internet des Objets (IoT)

1. Introduction et contexte général

À l'aube du XXIe siècle, le monde a connu une transformation numérique sans précédent, au cœur de laquelle se trouve une technologie désormais incontournable : l'Internet des Objets (Internet of Things — IoT). Ce concept désigne l'ensemble des infrastructures permettant à des objets physiques, équipés de capteurs et de moyens de communication, de collecter, échanger et exploiter des données via Internet, de manière automatique et sans intervention humaine directe.[1]

L'IoT représente un changement de paradigme fondamental : là où Internet reliait essentiellement des êtres humains entre eux, il relie désormais les objets eux-mêmes, introduisant la communication Machine to Machine (M2M). Cette évolution constitue l'un des piliers fondamentaux de la transformation digitale mondiale, touchant l'industrie, la santé, l'agriculture, l'énergie et les villes intelligentes.[2]

L'ampleur du phénomène se reflète dans les statistiques mondiales : plus de 15 milliards d'appareils connectés en 2026, avec des projections dépassant 30 milliards d'ici 2030, et un marché mondial estimé à plus de 500 milliards de dollars, en croissance annuelle de plus de 20%. Ces chiffres traduisent une réalité incontournable : l'IoT n'est plus une technologie expérimentale émergente, mais une infrastructure technologique mature en cours de déploiement massif à l'échelle mondiale.[3]

2. Historique et évolution de l'IoT

L'histoire de l'IoT est celle d'une évolution progressive portée par les avancées conjointes de la microélectronique, des réseaux de communication et des logiciels embarqués. Les grandes étapes de cette évolution sont résumées dans le tableau suivant :[5]

Période	Événements clés
1982	Première machine connectée (distributeur automatique, Carnegie Mellon University)
1999	Naissance du terme « IoT » par Kevin Ashton — Création du protocole MQTT
2003–2004	Généralisation de la technologie RFID — Publication IEEE 802.15.4 (Zigbee)
2008–2009	Le nombre d'objets connectés dépasse le nombre d'humains sur Terre
2011	Introduction du concept Industrie 4.0 en Allemagne
2012	Démocratisation des plateformes embarquées open source
2014–2015	Émergence des grandes plateformes IoT (AWS IoT, Azure IoT Hub)
2020	Déploiement commercial de la 5G
2026	Plus de 15 milliards d'appareils IoT connectés dans le monde
2030	Projection : plus de 30 milliards d'appareils connectés

Tableau 1.1 : Chronologie des grandes étapes de l'évolution de l'IoT

Le terme « Internet of Things » a été officiellement introduit en 1999 par Kevin Ashton, ingénieur britannique au MIT, pour décrire un système dans lequel des objets physiques seraient équipés de puces RFID permettant leur identification et leur suivi automatique via Internet. Son idée initiale était appliquée à la gestion de chaînes logistiques, mais le concept s'est rapidement élargi bien au-delà de cette application initiale.[1]

La décennie 2010-2020 a marqué le véritable essor de l'IoT, avec l'émergence de l'Industrie 4.0, des plateformes Cloud dédiées et des cartes embarquées accessibles, démocratisant le développement de solutions IoT à grande échelle. La décennie actuelle est celle de la maturité et de la consolidation, portée par le déploiement de la 5G, l'Edge Computing et l'intégration croissante de l'intelligence artificielle dans les systèmes connectés.[5]

3. Définitions et concepts fondamentaux

3.1 Définition de l'IoT

Plusieurs organismes internationaux ont proposé des définitions officielles de l'IoT :

Selon l'Union Internationale des Télécommunications (UIT-T) :

> *« L'IoT est une infrastructure mondiale pour la société de l'information, qui permet de disposer de services évolués en interconnectant des objets grâce aux technologies de l'information et de la communication interopérables existantes ou en évolution. »*

Selon l'Institut européen des normes de télécommunications (ETSI) :

> *« L'IoT désigne un écosystème mondial d'objets interconnectés, identifiables de manière unique, basé sur des protocoles de communication standardisés. »*

De manière plus synthétique, on peut définir l'IoT comme tout système dans lequel un objet physique est capable de percevoir son environnement, de traiter des informations et de communiquer avec d'autres systèmes via un réseau, afin d'automatiser des actions ou d'améliorer la prise de décision.[7]

3.2 Les six concepts fondamentaux

Tout système IoT repose sur six concepts fondamentaux interdépendants :

L'objet connecté (Smart Object) est un dispositif physique doté de capacités de perception, de traitement et de communication, capable d'interagir avec son environnement numérique de manière autonome. Il constitue l'élément de base de tout écosystème IoT.

L'identifiant unique garantit que chaque objet est identifiable de manière univoque dans le réseau, via une adresse IP, une adresse MAC, un identifiant RFID ou un URI. Cette unicité est une condition indispensable à la cohérence et à la fiabilité du système.

La connectivité désigne l'ensemble des mécanismes assurant la communication entre les objets, les passerelles et les systèmes d'information, via des réseaux filaires ou sans fil. Elle constitue la colonne vertébrale de tout système IoT.

La donnée est la matière première du système IoT, caractérisée par son volume, sa vélocité, sa variété et sa valeur potentielle une fois traitée et contextualisée. La donnée brute ne prend de sens que lorsqu'elle est analysée et interprétée.

L'intelligence désigne la capacité de traitement et d'analyse des données, exercée à différents niveaux (objet, passerelle, Cloud) pour transformer les données brutes en informations utiles

à la prise de décision. Elle peut être locale, distribuée ou centralisée selon l'architecture retenue.

L'action est la réponse physique déclenchée par les actionneurs en réaction aux décisions prises par le système, complétant la boucle fondamentale percevoir → traiter → agir. Cette boucle constitue le cycle de fonctionnement de tout système IoT autonome.

3.3 Caractéristiques fondamentales d'un système IoT

Un système IoT performant doit satisfaire un ensemble de caractéristiques essentielles :

Interopérabilité : capacité à communiquer et à fonctionner avec des équipements de différents fabricants et de différentes technologies,

Scalabilité : capacité à intégrer de nouveaux objets et à gérer un volume croissant de données sans dégradation des performances,

Fiabilité : garantie de continuité de service et d'absence de perte de données critiques,

Sécurité : protection des données et des équipements contre les accès non autorisés et les cyberattaques,

Faible latence : transmission et traitement des données dans des délais compatibles avec les exigences temps réel des applications,

Autonomie énergétique : optimisation de la consommation d'énergie pour les objets alimentés sur batterie.

3.4 Distinction entre IoT, M2M et systèmes cyber-physiques

Il est important de distinguer l'IoT de concepts voisins avec lesquels il est parfois confondu :

CONCEPT	DEFINITION	RELATION AVEC L'IOT
M2M	Communication directe entre machines via un réseau	Sous-ensemble de l'IoT, sans nécessairement passer par Internet
CPS	Systèmes intégrant calcul informatique et processus physiques	Concept plus large dont l'IoT est une implémentation pratique
IIOT	Application de l'IoT en milieu industriel	Spécialisation de l'IoT avec des exigences de fiabilité plus strictes
WOT	Extension de l'IoT utilisant les standards du Web	Approche spécifique basée sur HTTP et les API REST

Tableau 1.2 : Distinction entre IoT, M2M, CPS, IIoT et WoT

4. Architecture générale d'un système IoT

4.1 Le modèle en trois couches

L'architecture d'un système IoT structure l'ensemble de ses composants en niveaux fonctionnels distincts. Le modèle le plus fondamental et le plus largement référencé dans la littérature académique est l'architecture en trois couches :

La couche de perception (Perception Layer) est la couche physique du système. Elle regroupe l'ensemble des capteurs, actionneurs et objets connectés qui interagissent directement avec le monde physique. Son rôle est de collecter les données brutes de l'environnement et d'exécuter les commandes reçues des couches supérieures.**La couche réseau (Network Layer)** assure le transport des données collectées vers les systèmes de traitement. Elle englobe les passerelles IoT, les protocoles de communication (MQTT, HTTP, CoAP, Modbus) et les infrastructures réseau (Wi-Fi, Ethernet, LoRaWAN, 4G/5G). Elle constitue la colonne vertébrale du système IoT.

La couche application (Application Layer) est le niveau de traitement, d'analyse et de visualisation des données. Elle regroupe les plateformes IoT, les bases de données, les tableaux de bord et les interfaces utilisateur permettant l'exploitation des données collectées.

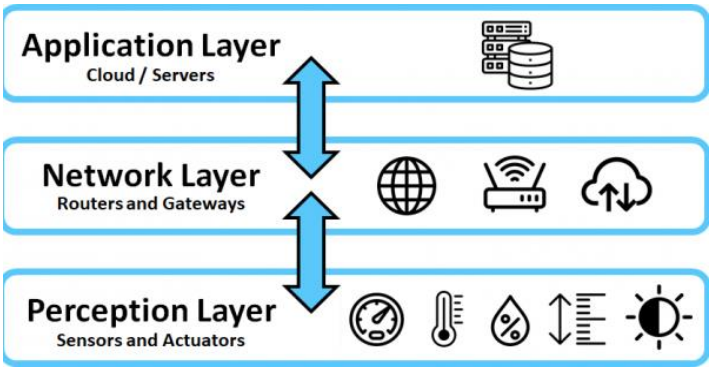


Figure 1.1 : Architecture en trois couches d'un système IoT (Perception, Réseau, Application)

4.2 L'architecture en cinq couches

Pour les systèmes IoT plus complexes, une architecture enrichie en cinq couches**[5]**est proposée, ajoutant une couche de traitement intermédiaire (Processing Layer) et une couche métier (Business Layer) permettant de mieux distinguer les fonctions de traitement, d'application et de gestion organisationnelle.

4.3 L'architecture Edge Computing et Fog Computing

Face à l'explosion des volumes de données générées par les objets connectés, les approches *Edge Computing* et *Fog Computing* ont émergé pour répondre aux limitations des architectures Cloud centralisées.

Dans une architecture **Edge Computing**, le traitement des données est effectué **localement**, au plus près de leur source, sur des passerelles ou des dispositifs embarqués, avant que les données consolidées ne soient transmises vers le Cloud. Cette approche offre plusieurs

avantages : réduction significative de la latence, diminution du trafic réseau, fonctionnement possible en mode déconnecté et meilleure confidentialité des données sensibles.

Le **Fog Computing**, concept introduit par Cisco, étend le Edge Computing en créant un réseau distribué de nœuds de traitement intermédiaires répartis géographiquement entre les objets et le Cloud, offrant ainsi une plus grande flexibilité et scalabilité.

Critère	Cloud Computing	Edge Computing	Fog Computing
Localisation du traitement	Centralisé (distant)	Local (périphérique)	Distribué (intermédiaire)
Latence	Élevée	Très faible	Faible
Dépendance réseau	Forte	Faible	Moyenne
Scalabilité	Excellente	Limitée	Bonne
Coût infrastructure	Élevé	Moyen	Moyen

Tableau 1.3 : Comparaison entre Cloud Computing, Edge Computing et Fog Computing

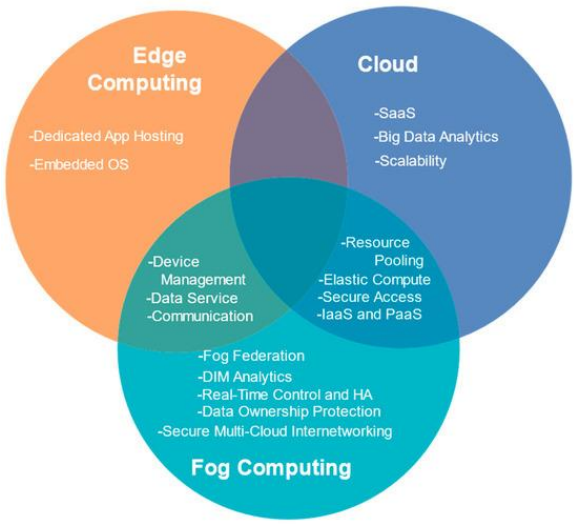


Figure 1.2 : Diagramme de Venn : zones communes entre Edge Computing, Cloud Computing et Fog Computing

5. Les composants d'un système IoT

5.1 Les capteurs (Sensors)

Un **capteur** est un dispositif électronique capable de détecter et mesurer une grandeur physique de son environnement, et de la convertir en signal électrique exploitable par un système numérique. Cette opération de conversion est appelée **transduction**.

Les capteurs peuvent être classifiés selon plusieurs critères :

Selon le type de signal de sortie :

Capteurs TOR *(Tout Ou Rien)* : fournissent un signal binaire (0 ou 1) indiquant la présence ou l'absence d'une condition,

Capteurs analogiques : fournissent un signal continu proportionnel à la grandeur mesurée (0-10V, 4-20mA),

Capteurs numériques : communiquent directement en numérique via des protocoles série (I^2C, SPI, 1-Wire).

Selon la grandeur mesurée :

Type de capteur	*Grandeur mesurée*	*Domaine d'application*
Capteur de présence/mouvement	Présence d'un objet ou d'une personne	Sécurité, automatisation du bâtiment
Capteur de température	Température (°C)	Industrie, santé, agriculture
Capteur de pression	Pression (Pa, bar)	Process industriel, météorologie
Capteur de luminosité	Intensité lumineuse (lux)	Gestion de l'éclairage, domotique
Capteur de CO_2	Concentration en CO_2 (ppm)	Qualité de l'air, ventilation
Capteur ultrasonique	Distance, présence	Détection d'obstacles, niveau de liquide
Capteur d'humidité	Humidité relative (%)	Agriculture, climatisation
Capteur de courant	Intensité électrique (A)	Gestion de l'énergie
Capteur RFID	Identification par radiofréquence	Logistique, contrôle d'accès

Tableau 1.4 : Classification des capteurs IoT : grandeurs mesurées et domaines d'application

Caractéristiques techniques importantes :

Sensibilité : capacité à détecter de faibles variations de la grandeur mesurée,
Précision : écart entre la valeur mesurée et la valeur réelle,
Résolution : plus petite variation détectable,
Plage de mesure : étendue des valeurs mesurables,
Temps de réponse : délai entre la variation physique et la réponse du capteur.



(a) ***(b)*** ***(c)***

Figure 1.3 : Exemples de capteurs utilisés dans les systèmes IoT

(a) : Capteur de température et d'humidité (DHT11)

(b) : Capteur ultrasonique de distance (HC-SR04)

(c) : Capteur de luminosité (LDR)

5.2 Les actionneurs (Actuators)

Un **actionneur** est un dispositif qui convertit un signal électrique de commande en une action physique sur l'environnement. Si le capteur est l'organe de perception du système IoT, l'actionneur en est l'organe d'action.

Type d'actionneur	Action produite	Exemples d'applications
Moteur électrique	Rotation mécanique	Convoyeurs, pompes, ventilateurs
Vérin pneumatique	Translation linéaire	Positionnement, serrage industriel
Électrovanne	Ouverture/fermeture de circuits fluides	Irrigation, process industriel
Relais électromagnétique	Commutation électrique	Commande de circuits de puissance
Luminaire LED connecté	Éclairage variable	Gestion intelligente de l'éclairage
Afficheur numérique	Signal visuel	Signalisation d'information
Sirène / Buzzer	Signal sonore	Alarmes et alertes

Tableau 1.5 : Types d'actionneurs IoT : actions produites et exemples d'applications

5.3 Les passerelles IoT (IoT Gateways)

La passerelle IoT est un composant stratégique de l'architecture IoT. Elle assure la jonction entre le monde des objets connectés et les systèmes d'information, en remplissant cinq fonctions essentielles :

Agrégation des données : collecte et consolidation des données provenant de multiples capteurs hétérogènes,

Conversion de protocoles (Protocol Translation) : traduction des protocoles des capteurs vers les protocoles Internet standards (MQTT, HTTP),

Prétraitement local (Edge Processing) : filtrage des données, calcul de statistiques, détection d'anomalies,

Stockage temporaire (Local Buffering) : conservation des données en cas d'interruption de la connexion réseau,

Sécurité : chiffrement des communications, authentification des dispositifs et contrôle d'accès.

5.4 Les réseaux de communication IoT

Le choix de la technologie réseau est une décision critique dans la conception d'un système IoT, conditionnant ses performances, sa fiabilité et son coût. Les critères de sélection incluent la portée, le débit, la latence, la consommation énergétique et le coût.[2]

Technologie	*Portée*	*Débit*	*Consommation*	*Usage typique*
Wi-Fi (IEEE 802.11)	50-100 m	100 Mbps+	Moyenne	Environnements intérieurs
Bluetooth BLE	10-100 m	2 Mbps	Très faible	Objets portables, capteurs
Zigbee (IEEE 802.15.4)	10-100 m	250 Kbps	Très faible	Réseaux de capteurs domotiques
LoRaWAN	1-15 km	50 Kbps	Ultra faible	Smart city, agriculture connectée
Sigfox	10-50 km	100 bps	Ultra faible	Compteurs intelligents
NB-IoT	National	200 Kbps	Très faible	IoT cellulaire basse consommation
Ethernet	Bâtiment	1 Gbps	Moyenne	Infrastructures industrielles
4G/5G	National	100 Mbps+	Élevée	Applications mobiles

Tableau 1.6 : Comparaison des technologies de communication IoT

6. Les protocoles de communication IoT

6.1 Le protocole MQTT

MQTT (Message Queuing Telemetry Transport) est un protocole de messagerie léger, standardisé ISO/IEC 20922, spécialement conçu pour les environnements IoT contraints en bande passante et en ressources. Développé initialement en 1999 par IBM pour la surveillance d'oléoducs via satellite, il est aujourd'hui le protocole de référence pour la télémétrie IoT en temps réel.**[9]**

MQTT repose sur le modèle Publish/Subscribe, organisé autour de trois acteurs :

Le publisher publie des messages sur des topics (sujets hiérarchiques),
Le broker (serveur central, ex. : Mosquitto) reçoit et redistribue les messages,
Le subscriber s'abonne aux topics qui l'intéressent et reçoit les messages correspondants.

Les topics sont structurés de manière hiérarchique en utilisant le caractère « / » comme séparateur. Par exemple : batiment/etage2/salle201/temperature.

MQTT définit trois niveaux de Qualité de Service (QoS) :

Niveau	*Nom*	*Garantie*	*Usage recommandé*
QoS 0	At most once	Message envoyé une fois, sans confirmation	Données non critiques
QoS 1	At least once	Message envoyé jusqu'à confirmation	Données importantes
QoS 2	Exactly once	Livraison garantie exactement une fois	Commandes critiques, alarmes

Tableau 1.7 : Niveaux de Qualité de Service (QoS) du protocole MQTT

Caractéristiques techniques :

Paramètre	Valeur
Port TCP standard	1883 (non chiffré), 8883 (TLS/SSL)
Taille minimale d'un paquet	2 octets
Protocole de transport	TCP/IP
Modèle	Publish/Subscribe asynchrone

Tableau 1.8 : Caractéristiques techniques du protocole MQTT

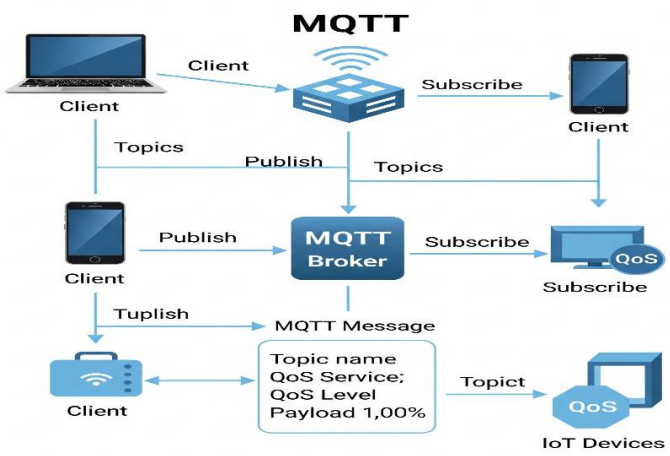


Figure 1.4 : Modèle Publish/Subscribe du protocole MQTT avec Broker central

6.2 Le protocole HTTP/REST

HTTP (HyperText Transfer Protocol) est le protocole fondateur du Web. Dans les architectures IoT, il est utilisé principalement via le style architectural REST (Representational State Transfer), qui expose des ressources accessibles par des requêtes standardisées :

Méthode HTTP	*Action*	*Usage IoT*
GET	Lire une ressource	Récupérer l'état d'un capteur
POST	Créer une ressource	Envoyer une nouvelle mesure
PUT	Mettre à jour une ressource	Modifier un paramètre de configuration
DELETE	Supprimer une ressource	Effacer un historique de données

Tableau 1.9 : Méthodes HTTP/REST et leur usage dans les systèmes IoT

HTTP/REST est particulièrement adapté aux interfaces de configuration, aux exports de données historiques et aux intégrations avec des services web tiers.

6.3 Le protocole CoAP

CoAP (Constrained Application Protocol), standardisé par l'IETF dans la RFC 7252 en 2014, est un protocole léger conçu pour les objets très contraints en ressources. Basé sur UDP, il offre un en-tête de seulement 4 octets et supporte le multicast, le rendant particulièrement adapté aux réseaux de capteurs à faible débit et faible consommation.[11]

6.4 Les protocoles industriels

Dans les environnements industriels, des protocoles spécifiques assurent la communication entre les équipements de terrain et les systèmes de supervision :

Modbus TCP : protocole industriel universel, libre de droits, supporté par la quasi-totalité des fabricants d'automates. Il organise les données en quatre zones mémoire (Coils, Discrete Inputs, Input Registers, Holding Registers) accessibles via des codes fonction standardisés. Utilise le port TCP 502.

OPC-UA (OPC Unified Architecture) : standard de communication de l'Industrie 4.0, offrant une interopérabilité totale, une sécurité avancée et un modèle d'information riche. Considéré comme le protocole de référence pour les nouvelles installations industrielles.

Profinet : protocole industriel Ethernet développé par Siemens, offrant des performances temps réel très strictes (latence < 1 ms pour Profinet IRT).

Comparaison des protocoles IoT :

Protocole	*Modèle*	*Transport*	*Usage principal*
MQTT	Publish/Subscribe	TCP	Télémétrie IoT temps réel
HTTP/REST	Requête/Réponse	TCP	API web, configuration
CoAP	Requête/Réponse	UDP	Objets très contraints
Modbus TCP	Maître/Esclave	TCP	Automatisme industriel
OPC-UA	Client/Serveur + Pub/Sub	TCP	Industrie 4.0
AMQP	Publish/Subscribe	TCP	Messagerie entreprise

Tableau 1.10 : Comparaison des principaux protocoles de communication IoT

L'analyse comparative des protocoles présentés dans le tableau ci-dessus appelle plusieurs observations critiques.

MQTT s'impose comme le protocole le plus adapté aux applications IoT de supervision en temps réel, grâce à sa légèreté (paquet minimal de 2 octets), son modèle asynchrone Publish/Subscribe et son support natif de la qualité de service (QoS 0, 1, 2). Il est particulièrement pertinent lorsque la bande passante est limitée ou lorsque le nombre de dispositifs connectés est élevé. Son principal inconvénient est la dépendance à un broker central, qui constitue un point de défaillance unique si aucune redondance n'est prévue.

HTTP/REST, bien que plus lourd et plus lent que MQTT, reste indispensable pour les interfaces de configuration, les exports de données historiques et les intégrations avec des services web tiers. Son modèle synchrone requête/réponse le rend inadapté à la télémétrie temps réel, mais son universalité et sa facilité d'intégration en font un complément naturel à MQTT dans les architectures IoT hybrides.

CoAP offre des performances similaires à MQTT sur UDP pour les objets très contraints, mais sa faible adoption industrielle et l'absence de broker standardisé limitent son usage aux réseaux de capteurs spécialisés (6LoWPAN, Zigbee).

Modbus TCP se distingue fondamentalement des protocoles précédents par sa destination industrielle : il n'est pas conçu pour l'Internet, mais pour les réseaux Ethernet industriels locaux. Sa fiabilité éprouvée sur plus de 40 ans et son support universel par les fabricants d'automates en font le protocole de référence pour la collecte de données depuis les PLC, mais son absence de mécanismes de sécurité natifs impose une segmentation réseau stricte dans tout déploiement IoT.

7. Les plateformes IoT

7.1 Introduction

Une plateforme IoT est un ensemble logiciel intégré fournissant les services nécessaires à la gestion, au traitement, à la visualisation et à l'exploitation des données collectées par les objets connectés. Elle constitue le cerveau applicatif du système IoT.

7.2 Node-RED

Node-RED est un outil de programmation visuelle par flux (flow-based programming), développé par IBM en 2013 et aujourd'hui géré par la OpenJS Foundation. Il est construit sur Node.js et permet de créer graphiquement des flux de traitement de données IoT par simple glisser-déposer de nœuds interconnectés, sans nécessiter d'expertise approfondie en programmation.[15]

Chaque flux est constitué de nœuds d'entrée (réception de données), de nœuds de traitement (transformation, filtrage, routage) et de nœuds de sortie (envoi vers des destinations). Le module complémentaire node-red-dashboard permet de créer des interfaces de supervision interactives avec des jauges, des graphiques, des voyants et des boutons de commande.

7.3 ThingsBoard

ThingsBoard est une plateforme IoT open source complète, disponible en version Community (gratuite) et Professional (commerciale). Elle intègre dans une seule solution:[16]

La gestion des appareils : enregistrement, provisionnement et monitoring des dispositifs IoT,

La collecte multi-protocoles : MQTT, HTTP et CoAP,

Un moteur de règles avancé : définition de chaînes de traitement automatisées (filtrage → transformation → action) pour la génération d'alarmes et l'envoi de notifications,

Des tableaux de bord configurables : visualisation en temps réel avec une large gamme de widgets (courbes, jauges, cartes, tableaux).

7.4 Grafana

Grafana est la référence mondiale en matière de visualisation de données, utilisée aussi bien dans les infrastructures IT que dans les systèmes IoT industriels. Associé à la base de données temporelle InfluxDB, il forme une stack de supervision particulièrement efficace pour la visualisation de séries temporelles.[17]

Grafana se distingue par la qualité graphique de ses tableaux de bord, son système d'alertes avancé (définition de seuils, notifications multicanaux) et sa capacité à se connecter à de nombreuses sources de données (InfluxDB, MySQL, PostgreSQL, Prometheus, etc.).

Comparaison des trois plateformes :

Critère	Node-RED	ThingsBoard	Grafana
Type	Outil de flux	Plateforme complète	Visualisation
Facilité d'utilisation	Très facile	Moyenne	Facile
Qualité de visualisation	Basique	Bonne	Excellente
Gestion des alarmes	Partielle	Complète	Partielle
Ressources requises	Faibles	Élevées	Moyennes
Idéal pour	Prototypage rapide	Supervision complète	Dashboards avancés

Tableau 1.11 : Comparaison entre les plateformes IoT : Node-RED, ThingsBoard et Grafana

8. L'IoT Industriel (IIoT) et l'Industrie 4.0

8.1 L'Internet des Objets Industriel

L'Internet des Objets Industriel (Industrial Internet of Things — IIoT) désigne l'application des technologies IoT aux environnements de production industrielle, avec des exigences beaucoup plus strictes de fiabilité (99,99%), de latence (millisecondes) et de sécurité que l'IoT grand public.**[13]**

Ses principales applications sont :

La maintenance prédictive : analyse en continu des données de vibration, de température et de courant des équipements pour anticiper les pannes avant qu'elles ne surviennent, permettant une réduction des coûts de maintenance de 10 à 25%,

Le suivi de production en temps réel : monitoring des indicateurs de performance (TRS, cadence de production, taux de rebuts),

L'optimisation énergétique : réduction de la consommation en adaptant le fonctionnement des machines aux besoins réels,

La traçabilité : enregistrement de tous les paramètres de production pour chaque pièce fabriquée.

8.2 L'Industrie 4.0

L'Industrie 4.0, concept introduit en Allemagne lors du salon de Hanovre en 2011, désigne la quatrième révolution industrielle caractérisée par l'intégration profonde des technologies numériques dans les processus de production.**[5]**

Révolution	*Période*	*Innovation clé*
Industrie 1.0	~1760	Machine à vapeur, mécanisation
Industrie 2.0	~1870	Électricité, chaîne de montage
Industrie 3.0	~1970	Informatique, automates programmables
Industrie 4.0	~2011	IoT, Intelligence Artificielle, Big Data, Cloud

Tableau 1.12 : Les quatre révolutions industrielles et leurs innovations clés

L'Industrie 4.0 repose sur neuf technologies piliers : l'IIoT, le Big Data, l'Intelligence Artificielle, le Cloud Computing, la Cybersécurité, la Réalité Augmentée, la Fabrication Additive, la Simulation numérique et les Robots Collaboratifs. L'IIoT en est la technologie centrale, sans laquelle aucune des autres ne peut fonctionner pleinement à l'échelle industrielle.

8.3 Les Cyber-Physical Systems (CPS)

Le concept de Systèmes Cyber-Physiques (Cyber-Physical Systems — CPS) est au cœur de l'Industrie 4.0. Un CPS est un système dans lequel des composants informatiques et des processus physiques sont profondément intégrés et interdépendants : les processus physiques influencent les calculs informatiques, et réciproquement. L'IoT constitue l'infrastructure de communication qui rend possible la mise en œuvre des CPS à grande échelle.

8.4 Le Digital Twin

Le jumeau numérique (Digital Twin) est une réplique virtuelle d'un objet, d'un processus ou d'un système physique, mise à jour en temps réel grâce aux données collectées par des capteurs IoT. Il permet de simuler des scénarios, d'anticiper des pannes, d'optimiser des processus et de former des opérateurs, sans risquer d'endommager les équipements réels. Le Digital Twin est l'une des applications les plus prometteuses de la convergence IoT-Intelligence Artificielle.

9. Sécurité dans les systèmes IoT

9.1 Les principales menaces

La sécurité est une dimension critique des systèmes IoT. La multiplication des objets connectés crée une surface d'attaque considérablement élargie, exposant les systèmes à de nouvelles catégories de menaces :

Attaques DDoS (Distributed Denial of Service) : des objets IoT mal sécurisés sont compromis et intégrés dans des botnets pour mener des attaques massives. Le botnet Mirai (2016) avait infecté des centaines de milliers de caméras IP et routeurs connectés,

Attaques Man-in-the-Middle (MitM) : interception et modification des communications entre objets et serveurs,

Vulnérabilités des firmwares : exploitation de failles dans les logiciels embarqués non mis à jour,

Usurpation d'identité (Spoofing) : un attaquant se fait passer pour un objet légitime afin d'injecter de fausses données dans le système,

Attaques physiques : accès direct aux équipements pour extraire des données ou modifier le firmware.

9.2 Les bonnes pratiques de sécurité IoT

Chiffrement des communications via TLS/SSL pour tous les échanges de données sensibles,

Authentification forte des dispositifs par certificats numériques X.509 ou tokens JWT,

Segmentation réseau (VLAN, DMZ) séparant les objets IoT des systèmes d'information critiques,

Mises à jour régulières des firmwares et systèmes d'exploitation de tous les dispositifs,

Surveillance continue du trafic réseau pour détecter les comportements anormaux (IDS),

Principe du moindre privilège : chaque dispositif n'a accès qu'aux ressources strictement nécessaires à son fonctionnement.

10. Avantages, défis et applications de l'IoT

10.1 Avantages et défis

Avantages	Défis et limites
Amélioration de la productivité (10-25%)	Complexité d'interopérabilité entre standards
Réduction des coûts de maintenance	Gestion des volumes massifs de données
Amélioration de la prise de décision	Dépendance à la connectivité réseau
Renforcement de la sécurité des personnes	Résistance au changement organisationnel
Traçabilité et conformité réglementaire	Obsolescence technologique rapide
Émergence de nouveaux modèles économiques	Investissement initial significatif

Tableau 1.13 : Avantages et défis des systèmes IoT

10.2 Applications sectorielles

a) IoT dans l'industrie manufacturière Supervision de production en temps réel, maintenance prédictive des équipements, contrôle qualité automatisé, optimisation énergétique et traçabilité complète des produits tout au long du cycle de fabrication.

b) IoT dans la santé (IoMT — Internet of Medical Things) Monitoring à distance des patients chroniques via des capteurs portables, gestion intelligente des stocks de médicaments, maintenance prédictive des équipements médicaux coûteux (IRM, scanners), et piluliers connectés rappelant les prises médicamenteuses.

c) IoT dans l'agriculture (Smart Farming) Capteurs de sol mesurant l'humidité, le pH et les nutriments pour l'optimisation de l'irrigation et de la fertilisation, surveillance des cultures par drones, serres intelligentes régulant automatiquement les conditions de croissance.

d) Smart Cities Gestion intelligente de l'éclairage public adaptatif, optimisation des tournées de collecte des déchets, surveillance de la qualité de l'air, gestion dynamique du trafic et des parkings en temps réel.

e) IoT dans le secteur de l'énergie Compteurs intelligents (Smart Meters) permettant la facturation à la consommation réelle, réseaux électriques intelligents (Smart Grids) optimisant la distribution selon la demande, monitoring des installations de production d'énergie renouvelable.

10.3 L'IoT en Algérie et dans le monde arabe

En Algérie, le déploiement de l'IoT est en phase d'émergence, soutenu par le déploiement progressif de la 4G, les initiatives de grandes entreprises publiques comme Sonatrach et Sonelgaz, et la dynamique croissante des universités et des startups technologiques dans le domaine du numérique. Les défis restent cependant importants : infrastructures de connectivité insuffisantes dans les zones rurales, manque de compétences locales spécialisées et cadre réglementaire à adapter aux spécificités de l'IoT.

Dans le monde arabe, plusieurs pays se distinguent par leur engagement dans la transformation numérique : les Émirats Arabes Unis avec le projet Smart Dubai, l'Arabie Saoudite avec la Saudi Vision 2030 et le projet NEOM, et l'Égypte avec sa Nouvelle Capitale Administrative, investissent massivement dans les technologies IoT pour moderniser leurs infrastructures urbaines et industrielles.

11. Exemples des projets IoT dans ESSAT

11.1 Système de surveillance de la qualité de l'air par drone quadrotor

11.1.1 Présentation générale du projet

L'un des exemples les plus illustratifs des applications concrètes de l'IoT dans le domaine environnemental est le projet de conception d'un système de surveillance de la qualité de l'air embarqué sur un drone quadrotor, réalisé par Laribi Nesserine et Bouri Abdennour à l'École Supérieure en Sciences Appliquées de Tlemcen (ESSA-Tlemcen), en 2022.[18]

Ce projet démontre comment les technologies IoT peuvent être intégrées dans un système mobile autonome pour collecter, transmettre et analyser des données environnementales en temps réel, illustrant ainsi de manière concrète plusieurs des concepts fondamentaux de l'IoT présentés dans ce chapitre : la couche de perception, la couche réseau, la passerelle IoT et la couche application Cloud.

11.1.2 Problématique et objectifs

La dégradation croissante de la qualité de l'air représente un risque sanitaire majeur à l'échelle mondiale. Les systèmes de surveillance traditionnels, fixes et stationnaires, présentent des limitations importantes en termes de couverture géographique et de flexibilité de déploiement. Ce projet propose une solution innovante en combinant deux technologies complémentaires : les drones UAV (Unmanned Aerial Vehicle) et l'Internet des Objets, pour créer un système de surveillance mobile, flexible et connecté.

Les objectifs principaux du projet sont les suivants :

Concevoir et assembler un drone quadrotor capable de transporter une charge utile de mesure,
Développer un système embarqué de collecte de données environnementales multi-capteurs,
Transmettre les données en temps réel vers une plateforme IoT Cloud,
Estimer l'indice de qualité de l'air (IQA) par deux approches : conventionnelle et logique floue.

11.1.3 Architecture IoT du système

L'architecture du système s'articule autour de trois couches fonctionnelles, directement alignées avec le modèle IoT en trois couches décrit précédemment dans ce chapitre :

Couche de perception — Le nœud de transmission embarqué

Le nœud de mesure, embarqué sur le drone, intègre plusieurs capteurs spécialisés dans la mesure des polluants atmosphériques :

Capteur	*Paramètre mesuré*	*Polluants détectés*
MICS-6814	Gaz polluants	CO, NH_3, NO_2
PMS5003	Particules en suspension	PM2.5, PM10
MQ-131	Ozone	O_3 (10-1000 ppb)
DHT11	Conditions atmosphériques	Température, humidité

Tableau 1.14 : Capteurs embarqués sur le drone quadrotor et paramètres mesurés

Un microcontrôleur Arduino Nano (ATmega328, 8 bits, 32 Ko Flash) assure l'acquisition et le traitement local des données issues de ces capteurs. Les mesures sont ensuite transmises sans fil à la station au sol via un module radio NRF24L01+PA+LNA, opérant sur la bande de fréquence ISM 2,4 GHz, avec une portée de transmission atteignant 720 mètres à un débit de 1 Mbps.

Couche réseau — La passerelle IoT

La passerelle de réception est constituée d'un module NodeMCU ESP8266, combinant un microcontrôleur et un module Wi-Fi intégré. Ce composant reçoit les données transmises par le module radio embarqué sur le drone et les transfère vers la plateforme Cloud via le protocole HTTP, illustrant parfaitement le rôle de conversion de protocoles et d'agrégation de données propre aux passerelles IoT.

Couche application — La plateforme Cloud ThingSpeak

Les données sont visualisées et analysées sur la plateforme IoT Cloud ThingSpeak, développée par MathWorks. Cette plateforme permet :

La visualisation en temps réel des courbes de concentration des polluants,

L'exécution de scripts MATLAB directement sur le serveur pour le traitement des données,

Le calcul et l'affichage de l'Indice de Qualité de l'Air (IQA) selon deux méthodes d'estimation.

L'architecture globale du système peut être représentée comme suit :

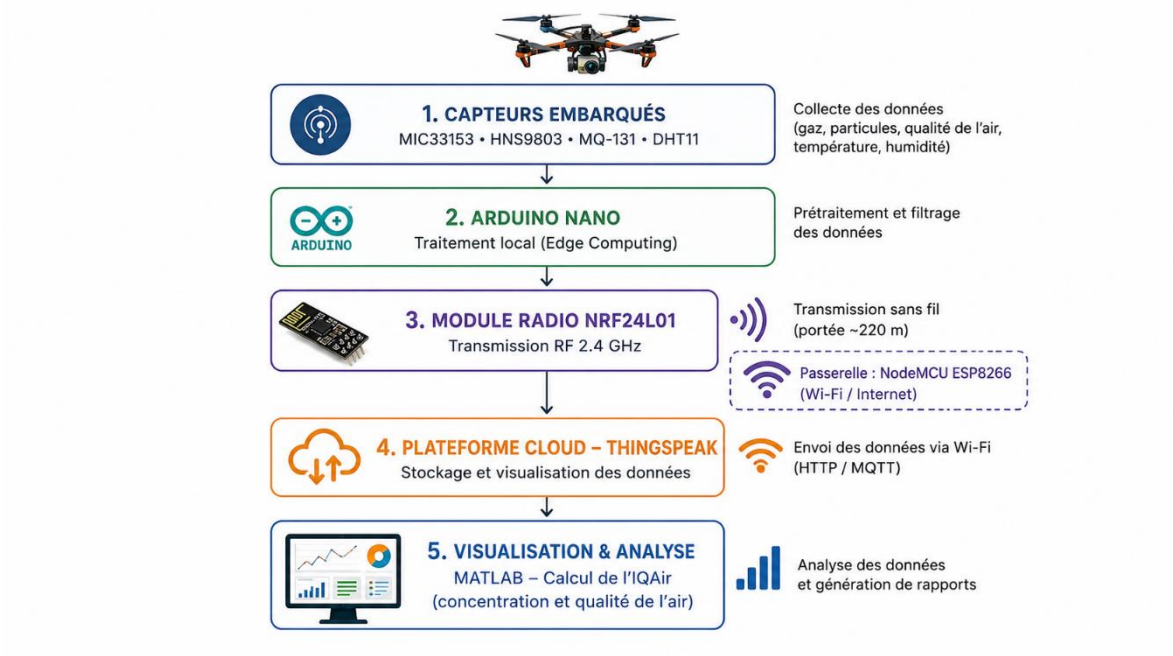


Figure 1.5 : Architecture IoT du système de surveillance de la qualité de l'air par drone (Perception → Réseau → Application Cloud)

11.1.4 Estimation de l'indice de qualité de l'air

L'une des contributions techniques majeures de ce projet réside dans la comparaison de deux approches pour l'estimation de l'Indice de Qualité de l'Air (IQA) :

Approche conventionnelle Cette méthode utilise l'interpolation linéaire définie par l'agence américaine de protection de l'environnement (EPA). Elle calcule un sous-indice pour chaque polluant mesuré, puis retient la valeur maximale comme IQA global. Les catégories de qualité de l'air sont définies sur une échelle de 0 à 500 :

Indice IQA	Catégorie	Signification
0 – 50	Bon	Qualité satisfaisante
51 – 100	Modéré	Risque modéré pour les personnes sensibles
101 – 150	Mauvais pour groupes sensibles	Risque pour les personnes vulnérables
151 – 200	Mauvais	Risque pour toute la population
201 – 300	Très mauvais	Alerte sanitaire
301 – 500	Dangereux	Urgence sanitaire

Tableau 1.15 : Échelle de l'Indice de Qualité de l'Air (IQA) selon la méthode EPA

Approche par logique floue La logique floue, concept introduit par Lotfi Zadeh, permet de traiter des variables linguistiques imprécises plutôt que des valeurs binaires strictes. Le système de logique floue développé dans ce projet prend en entrée les concentrations des cinq polluants mesurés, et produit en sortie une estimation de l'IQA global. Le processus comprend trois étapes : la fuzzification (conversion des valeurs mesurées en degrés d'appartenance à des ensembles flous), l'inférence (application de règles IF-THEN définissant les relations entre polluants et qualité de l'air) et la défuzzification (conversion du résultat flou en valeur numérique précise, par la méthode du centre de gravité).

Les résultats expérimentaux ont démontré que la méthode par logique floue fournit des estimations plus précises et plus représentatives de la qualité réelle de l'air que la méthode conventionnelle, en particulier dans les situations où plusieurs polluants sont présents simultanément à des concentrations modérées.

11.1.5 Résultats et apport IoT du projet

Les tests de vol ont été conduits en trois phases progressives : tests assistés sur banc d'essai, tests en environnement intérieur contrôlé, puis tests en extérieur dans différentes zones de la ville de Tlemcen. Les données de qualité de l'air collectées lors des vols extérieurs ont été transmises en temps réel à la plateforme ThingSpeak, où elles ont été visualisées et analysées.

Ce projet illustre plusieurs dimensions fondamentales de l'IoT abordées dans ce chapitre :

L'hétérogénéité des capteurs : quatre types de capteurs mesurant des grandeurs physiques différentes coexistent dans un même nœud de perception,

La communication sans fil contrainte : le protocole RF NRF24L01 assure la transmission de données entre un objet mobile (le drone) et une station fixe, avec des contraintes de poids et de consommation énergétique,

Le rôle de la passerelle IoT : le module ESP8266 assure la conversion du protocole RF vers le protocole HTTP, permettant l'intégration des données dans l'écosystème Internet,

L'exploitation Cloud des données : la plateforme ThingSpeak centralise, visualise et analyse les données en temps réel, conformément au modèle de la couche application IoT,

L'intelligence embarquée : l'algorithme de logique floue ajoute une couche d'intelligence dans le traitement des données, transformant des mesures brutes en une évaluation qualitative exploitable directement par les décideurs et les services de santé publique.

Ce projet démontre concrètement que l'architecture IoT en trois couches peut être implémentée avec des composants embarqués économiques (Arduino Nano, ESP8266) et une plateforme Cloud open source (ThingSpeak). Les résultats obtenus confirment que la combinaison d'une transmission radio NRF24L01 et d'une passerelle Wi-Fi ESP8266 assure une collecte de données fiable sur une portée de 720 mètres, ce qui est suffisant pour les environnements urbains ciblés. La comparaison entre la méthode conventionnelle EPA et la logique floue a montré que cette dernière fournit une estimation plus représentative de la qualité réelle de l'air dans les situations de pollution multi-polluants, ce qui constitue un apport scientifique direct du projet.**[18]**

11.2 Système de feux de signalisation intelligents et automatisés

11.2.1 Présentation générale du projet

Un second exemple concret illustrant l'application des technologies IoT dans le domaine des villes intelligentes est le projet de conception et d'implémentation d'un système de feux de signalisation intelligent et automatisé, réalisé par Loulhaci Romaissa et Bensouna Bouchra à l'École Supérieure en Sciences Appliquées de Tlemcen (ESSA-Tlemcen), en 2024.[19]

Ce projet s'inscrit pleinement dans la vision des Smart Cities (villes intelligentes), en proposant une solution technologique avancée pour la gestion dynamique du trafic urbain. Il mobilise plusieurs composantes fondamentales de l'IoT — capteurs, traitement de données en temps réel, communication entre dispositifs et prise de décision automatisée — pour répondre à une problématique urbaine majeure : la congestion routière aux intersections.

11.2.2 Problématique et objectifs

La congestion routière est un défi universel affectant la qualité de vie urbaine, la consommation énergétique et l'environnement. Les systèmes de feux de signalisation traditionnels, fonctionnant sur des cycles temporels fixes, sont structurellement incapables de s'adapter aux variations en temps réel de la densité du trafic, générant des attentes inutiles et des embouteillages aux heures de pointe.

Ce projet propose une réponse à cette limitation en développant un système de feux de signalisation adaptatif et intelligent, capable de :

- Détecter, suivre et compter les véhicules en temps réel par traitement vidéo,
- Ajuster dynamiquement la durée des phases de signalisation en fonction de la densité du trafic mesurée,
- Assurer une communication entre les feux de différentes intersections pour une coordination optimale du trafic,
- Réduire les temps d'attente, optimiser la consommation énergétique et améliorer la fluidité des déplacements urbains.

11.2.3 Architecture du système et composants IoT

Le système développé s'articule autour de trois fonctions principales : la perception de l'environnement routier, le traitement intelligent des données collectées, et la communication entre les nœuds du réseau de feux.

Couche de perception — Acquisition vidéo en temps réel

La collecte de données repose sur des caméras positionnées au niveau de chaque voie de l'intersection. Ces caméras constituent les capteurs principaux du système IoT, fournissant un flux vidéo continu qui sert de source de données brutes pour le traitement algorithmique. Contrairement aux capteurs traditionnels encastrés dans la chaussée (boucles inductives, tubes pneumatiques, capteurs magnétiques), l'approche caméra présente l'avantage d'être non

intrusive, facilement déployable et capable de fournir des informations riches sur la nature, la position et le mouvement des véhicules.

Couche de traitement — Intelligence embarquée sur Raspberry Pi

Le traitement des données vidéo est assuré par des cartes Raspberry Pi 4 Model B, qui jouent ici le double rôle de passerelles IoT et d'unités de traitement embarqué (Edge Computing). Chaque Raspberry Pi exécute localement une chaîne de traitement algorithmique complète :

Détection des véhicules par le modèle de deep learning TensorFlow Lite, optimisé pour les plateformes embarquées à ressources limitées. Ce modèle, basé sur une architecture de réseau de neurones convolutifs, identifie et localise les véhicules dans chaque image vidéo avec une précision mesurée en conditions réelles,

Suivi des véhicules par l'algorithme de tracking par centroïde (Centroid Tracking Algorithm), qui assigne un identifiant unique à chaque véhicule détecté et suit sa trajectoire d'une image à l'autre,

Comptage en temps réel des véhicules franchissant une ligne de comptage virtuelle définie dans le champ de la caméra, permettant d'évaluer en continu la densité du flux de trafic sur chaque voie.

La chaîne de traitement peut être représentée comme suit :

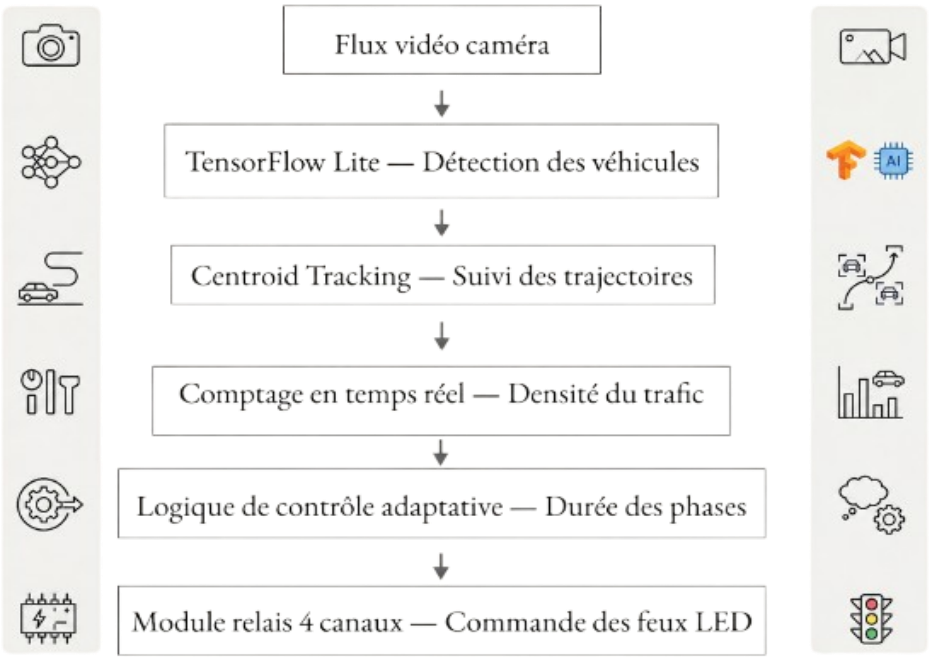


Figure 1.6 : Chaîne de traitement algorithmique du système de feux intelligents (de la caméra à la commande des feux)

Couche de communication — Coordination inter-intersections

L'un des aspects les plus innovants du projet est la mise en œuvre d'une communication entre les Raspberry Pi des différentes intersections via la bibliothèque pigpio, utilisant le réseau local comme infrastructure de communication. Cette architecture permet aux feux de signalisation de partager leurs données de trafic en temps réel et de coordonner leurs cycles de signalisation de manière cohérente, évitant ainsi les situations où plusieurs feux adjacents créent des blocages en cascade.

Cette communication inter-dispositifs illustre parfaitement le paradigme M2M (Machine to Machine) au cœur de l'IoT : les feux ne sont plus des équipements isolés fonctionnant de manière indépendante, mais des nœuds d'un réseau intelligent collaborant pour optimiser le flux global du trafic.

11.2.4 Réalisation matérielle et logicielle

Composants matériels :

COMPOSANT	ROLE DANS LE SYSTEME
RASPBERRY PI 4 MODEL B	Unité centrale de traitement et passerelle IoT
CAMERA	Acquisition du flux vidéo en temps réel
MODULE RELAIS 4 CANAUX	Interface de commande des feux de signalisation
LEDS TRICOLORES	Signalisation physique (rouge, orange, vert)
BOITIER IMPRIME EN 3D (ABS)	Structure physique du feu de signalisation

Tableau 1.16 : Composants matériels du système de feux de signalisation intelligents

Le boîtier du feu a été entièrement conçu et fabriqué par impression 3D en matériau ABS, illustrant l'intégration des technologies de fabrication additive dans la réalisation de prototypes IoT embarqués.

Algorithme de contrôle adaptatif

La logique de contrôle adaptative du système repose sur le principe suivant : la durée de la phase verte est calculée dynamiquement en fonction du nombre de véhicules comptés sur chaque voie. Une voie présentant une densité de trafic élevée bénéficie d'une phase verte prolongée, tandis qu'une voie faiblement chargée reçoit une phase verte réduite, évitant les temps d'attente inutiles. Ce mécanisme d'adaptation en temps réel constitue une rupture fondamentale avec les systèmes traditionnels à cycles fixes.

11.2.5 Résultats et apport par rapport aux systèmes traditionnels

Les tests réalisés en conditions réelles ont permis de valider les performances du système et de les comparer aux systèmes de feux traditionnels à cycles fixes. Les résultats démontrent que le système intelligent permet une réduction significative des temps d'attente aux intersections, une meilleure fluidité du trafic dans les situations de densité variable, et une optimisation de la consommation énergétique en évitant les phases vertes inutilement longues en l'absence de véhicules.

Le tableau suivant résume les avantages comparatifs identifiés :

Critère	Système traditionnel	Système intelligent
Adaptation au trafic	Cycles fixes, non adaptatifs	Adaptation en temps réel
Gestion des embouteillages	Insuffisante	Optimisée par comptage vidéo
Coordination inter-intersections	Inexistante	Communication M2M via réseau
Consommation énergétique	Non optimisée	Réduite par cycles adaptatifs
Technologie de détection	Capteurs encastrés coûteux	Caméra + vision par ordinateur

Tableau 1.17 : Comparaison entre système de feux traditionnel et système de feux intelligent

11.2.6 Dimension Smart City et valorisation économique

Ce projet va au-delà de la dimension purement technique en proposant un modèle économique complet sous la forme d'une startup baptisée AlgiTraffic, dont le Business Model Canvas a été développé dans le cadre du mémoire. Cette initiative illustre comment une solution IoT académique peut être transformée en un produit commercialisable et scalable, contribuant directement aux objectifs de développement des villes intelligentes en Algérie.

AlgiTraffic se positionne comme une solution moderne et durable pour la gestion intelligente du trafic urbain, adressant aussi bien les collectivités territoriales que les gestionnaires d'infrastructures routières, dans un contexte où l'Algérie amorce progressivement sa transition vers des systèmes de transport intelligents.

Ce projet démontre que l'intégration de TensorFlow Lite sur Raspberry Pi permet une détection de véhicules en temps réel avec une précision suffisante pour adapter dynamiquement les cycles de feux. La communication inter-intersections via la bibliothèque pigpio sur réseau local constitue une implémentation concrète du paradigme M2M, validant le principe de coordination distribuée entre nœuds IoT. Les tests comparatifs ont établi une réduction mesurable des temps d'attente par rapport aux systèmes à cycles fixes, ce qui confirme l'intérêt pratique de l'approche adaptative dans des conditions de trafic variable.**[19]**

12. Exemples d'applications IoT

12.1 Surveillance médicale à distance — Philips Healthcare (2025)

En 2025, Philips Healthcare a déployé à grande échelle un système IoT de surveillance à distance des patients chroniques dans plusieurs établissements hospitaliers européens, notamment aux Pays-Bas et en Allemagne. Ce système repose sur des capteurs portables (*wearables*) mesurant en continu les paramètres vitaux des patients — fréquence cardiaque, saturation en oxygène, pression artérielle et température corporelle — et transmettant ces données en temps réel vers une plateforme Cloud Microsoft Azure IoT Hub via le protocole MQTT.

L'architecture retenue intègre une couche Edge Computing assurée par des passerelles locales installées dans chaque service hospitalier, permettant un prétraitement des données et une détection immédiate des anomalies critiques sans dépendance à la connectivité Internet. Un algorithme d'intelligence artificielle embarqué analyse les tendances des paramètres vitaux et déclenche automatiquement des alertes vers le personnel soignant en cas de dégradation de l'état du patient.

Les résultats cliniques publiés en 2025 font état d'une réduction de 28% des hospitalisations d'urgence non planifiées et d'une amélioration significative de la qualité de suivi des patients chroniques, démontrant l'impact direct des architectures IoT sur les pratiques médicales modernes [20].

12.2 Gestion intelligente de l'énergie urbaine — Projet NEOM Smart City (2024–2025)

NEOM, la mégapole intelligente en cours de construction en Arabie Saoudite dans le cadre de la Saudi Vision 2030, constitue l'un des projets de Smart City les plus ambitieux et les plus avancés technologiquement au monde. Depuis 2024, la phase de déploiement des infrastructures IoT de la ville est en cours, intégrant un réseau dense de plusieurs millions de capteurs connectés pour la gestion en temps réel de l'ensemble des systèmes urbains.

Sur le plan énergétique, des capteurs de consommation intelligents (*Smart Meters*) sont déployés dans chaque bâtiment et infrastructure, transmettant leurs données via un réseau LoRaWAN à une plateforme centrale de gestion énergétique basée sur ThingsBoard Enterprise. Le système ajuste automatiquement la production et la distribution d'énergie renouvelable — solaire et éolienne — en fonction de la demande en temps réel, visant un objectif de 100% d'énergie propre pour l'ensemble de la ville.

Sur le plan de la mobilité urbaine, un réseau de capteurs de trafic et de qualité de l'air, couplé à des systèmes de feux intelligents similaires à ceux développés dans les projets présentés aux sections 11 et 12 de ce chapitre, assure une gestion dynamique et optimisée des flux de circulation. Les données collectées sont analysées en temps réel par des algorithmes de machine learning déployés en Edge Computing, permettant une réduction estimée de 40% de la congestion routière par rapport aux systèmes traditionnels [21].

Ce projet illustre parfaitement la convergence de l'ensemble des technologies IoT présentées dans ce chapitre — capteurs hétérogènes, protocoles de communication (MQTT, LoRaWAN, HTTP), Edge Computing, plateformes de supervision et intelligence artificielle — au service d'une vision urbaine intégrée et durable.

Conclusion

L'étude menée dans ce chapitre a permis d'établir les fondements théoriques nécessaires à la compréhension et à la mise en œuvre d'un système de supervision industrielle connecté. L'analyse des architectures IoT a mis en évidence que le modèle en trois couches — perception, réseau, application — constitue le cadre le plus adapté à un déploiement industriel réaliste, notamment lorsqu'il est enrichi par une approche Edge Computing permettant de réduire la latence et la dépendance au Cloud.

L'étude comparative des protocoles de communication a montré que MQTT s'impose comme le choix le plus pertinent pour la transmission de données de supervision en temps réel, en raison de sa légèreté et de son modèle asynchrone, tandis que Modbus TCP reste la solution de référence pour la collecte de données directement depuis les automates industriels.

L'analyse des deux projets réalisés à l'ESSA-Tlemcen a confirmé la faisabilité technique d'une intégration IoT avec des composants embarqués accessibles (Raspberry Pi, ESP8266, Arduino), et a démontré que des résultats concrets et mesurables peuvent être obtenus dans des délais raisonnables. Ces observations orientent directement les choix architecturaux et technologiques retenus dans le présent projet, présentés dans les chapitres suivants.

Chapitre 2 : Automatisation des Systèmes de Production

1. Introduction

L'industrie moderne repose sur des systèmes de production de plus en plus complexes, où la précision, la rapidité et la fiabilité sont des exigences fondamentales. Pour répondre à ces défis, l'automatisation industrielle s'est imposée comme une nécessité incontournable, transformant profondément la manière dont les processus de fabrication sont conçus, pilotés et supervisés.

Un système de production automatisé ne se résume pas à une simple machine : il constitue un ensemble cohérent d'éléments mécaniques, électriques, électroniques et informatiques, organisés selon une architecture précise et pilotés par une logique de commande rigoureuse. Comprendre cette architecture, ses composantes et ses interactions est une étape essentielle avant toute démarche d'intégration de technologies avancées telles que l'IoT industriel.

Ce chapitre présente les fondements théoriques des systèmes de production automatisés. Nous examinerons d'abord la structure générale d'un système automatisé et ses trois parties constitutives. Nous étudierons ensuite les technologies de commande, depuis la logique câblée jusqu'à l'automate programmable industriel. Nous aborderons également les réseaux de communication industriels et leur rôle dans l'intégration des équipements. Enfin, nous présenterons les systèmes de supervision industrielle et leur évolution vers des architectures connectées.

2. Le Système de Production Automatisé

2.1 Définition et Structure Générale

Un système de production automatisé est un ensemble organisé de moyens techniques, matériels et logiciels, capable d'exécuter des tâches de transformation, de traitement ou de transport de matière, d'énergie ou d'information, de manière autonome ou semi-autonome, avec une intervention humaine réduite ou nulle lors du fonctionnement normal. [22]

L'automatisation d'un système de production répond à plusieurs objectifs fondamentaux :

Amélioration de la productivité : augmentation des cadences de production et réduction des temps de cycle,

Amélioration de la qualité : réduction des erreurs humaines et garantie de la répétabilité des opérations,

Amélioration des conditions de travail : suppression des tâches pénibles, dangereuses ou répétitives,

Réduction des coûts : optimisation de la consommation d'énergie et des matières premières,

Traçabilité et conformité : enregistrement systématique des paramètres de production.[22]

Tout système de production automatisé s'articule autour de trois parties fondamentales et interdépendantes, dont les interactions définissent le comportement global du système :

La Partie Opérative (PO) : elle agit sur la matière,

La Partie Commande (PC) : elle pilote la partie opérative,

La Partie Interface Homme-Machine (IHM) : elle assure la communication entre l'opérateur et le système.

2.2 La Partie Opérative (PO)

La partie opérative est l'ensemble des éléments physiques qui agissent directement sur la matière d'œuvre — c'est-à-dire le produit à transformer, à déplacer ou à traiter. Elle constitue la dimension physique et mécanique du système automatisé, visible et tangible dans l'environnement industriel.

2.2.1 Les Actionneurs et Préactionneurs

Un **actionneur** est un dispositif qui convertit une énergie (électrique, pneumatique, hydraulique) en une action mécanique sur la matière d'œuvre. Les actionneurs les plus couramment utilisés dans les systèmes industriels sont :

Les moteurs électriques : assurent les mouvements de rotation (convoyeurs, pompes, ventilateurs, broches d'usinage),

Les vérins pneumatiques : assurent les mouvements de translation linéaire (serrage, positionnement, éjection de pièces),

Les vérins hydrauliques : utilisés pour les efforts importants (presses, machines de découpe),

Les électrovannes : contrôlent les circuits de fluides (eau, air comprimé, huile),

Les résistances chauffantes : assurent le traitement thermique des produits.

Un **préactionneur** est l'interface entre la partie commande et l'actionneur. Il reçoit les ordres logiques de faible puissance provenant de la partie commande et les amplifie pour alimenter les actionneurs en puissance. Les préactionneurs typiques sont les contacteurs, les distributeurs pneumatiques et les variateurs de fréquence.

2.2.2 Les Capteurs et Détecteurs

Les **capteurs** sont les organes sensoriels de la partie opérative. Ils mesurent en permanence l'état du système et de la matière d'œuvre, et transmettent ces informations à la partie commande pour lui permettre de prendre les décisions appropriées.

Les capteurs industriels peuvent être classifiés selon le type de signal qu'ils produisent :

Capteurs TOR (Tout Ou Rien) : fournissent un signal binaire (0 ou 1). Exemples : détecteurs de présence inductifs, capteurs de fin de course, cellules photoélectriques,

Capteurs analogiques : fournissent un signal continu proportionnel à la grandeur mesurée, généralement 0-10V ou 4-20mA. Exemples : capteurs de température, de pression, de débit,

Capteurs numériques : communiquent directement en numérique via des protocoles série (I^2C, SPI, RS-485). Exemples : encodeurs incrémentaux, capteurs de vision industrielle.

Les grandeurs physiques les plus fréquemment mesurées dans un contexte industriel incluent la position, la vitesse, la température, la pression, le débit, le niveau et l'intensité électrique.

2.3 La Partie Commande (PC)

La partie commande est le cerveau du système automatisé. Elle reçoit les informations provenant des capteurs de la partie opérative et de l'interface homme-machine, les traite selon un programme prédéfini, et envoie les ordres aux préactionneurs pour piloter les actionneurs.

2.3.1 Rôle et Fonctions

La partie commande remplit trois fonctions essentielles :

Acquisition : lecture des états des capteurs et des consignes de l'opérateur,

Traitement : exécution du programme de commande pour déterminer les actions à effectuer,

Restitution : envoi des ordres aux préactionneurs et transmission des informations à l'IHM.

2.3.2 Les Informations Échangées

La partie commande échange deux types d'informations avec son environnement :

Avec la partie opérative :

En entrée (PO → PC) : comptes rendus des capteurs (présence de pièce, fin de course atteint, température mesurée, pression actuelle...),

En sortie (PC → PO) : ordres aux préactionneurs (démarrer moteur, ouvrir vanne, activer vérin...).

Avec la partie interface :

En entrée (IHM → PC) : consignes de l'opérateur (démarrage, arrêt, paramètres de production),

En sortie (PC → IHM) : informations sur l'état du système (voyants, alarmes, valeurs mesurées).

2.4 La Partie Interface Homme-Machine (IHM)

L'interface homme-machine est le point de contact entre l'opérateur humain et le système automatisé. Elle permet à l'opérateur de surveiller l'état du système, d'intervenir en cas de besoin et de modifier les paramètres de fonctionnement.

2.4.1 Types d'IHM

Les interfaces homme-machine se déclinent en plusieurs niveaux de complexité :

Le pupitre de commande : solution la plus simple, composée de boutons-poussoirs, de voyants lumineux et de sélecteurs. Il permet uniquement des interactions basiques (démarrage, arrêt, acquittement d'alarme),

Le panneau de commande avec afficheur : intègre un écran alphanumérique ou graphique permettant d'afficher des valeurs numériques et des messages d'état,

L'écran tactile (Panel PC) : solution moderne offrant une interface graphique riche, avec des synoptiques animés, des courbes de tendance et des tableaux de bord interactifs,

Le système SCADA : logiciel de supervision s'exécutant sur un ordinateur industriel, offrant les fonctionnalités les plus avancées (historisation, rapports, gestion des alarmes, accès distant).

2.4.2 Évolution vers les IHM Connectées

Avec l'émergence de l'Industrie 4.0, les interfaces homme-machine évoluent vers des solutions accessibles à distance via des navigateurs web ou des applications mobiles, permettant à l'opérateur de surveiller et de contrôler le système depuis n'importe quel endroit disposant d'une connexion réseau. Cette évolution constitue l'un des piliers de la supervision industrielle connectée.

2.5 Interactions entre les Trois Parties

Les trois parties du système automatisé forment une boucle de contrôle fermée et permanente. La figure suivante illustre les flux d'information entre ces parties :

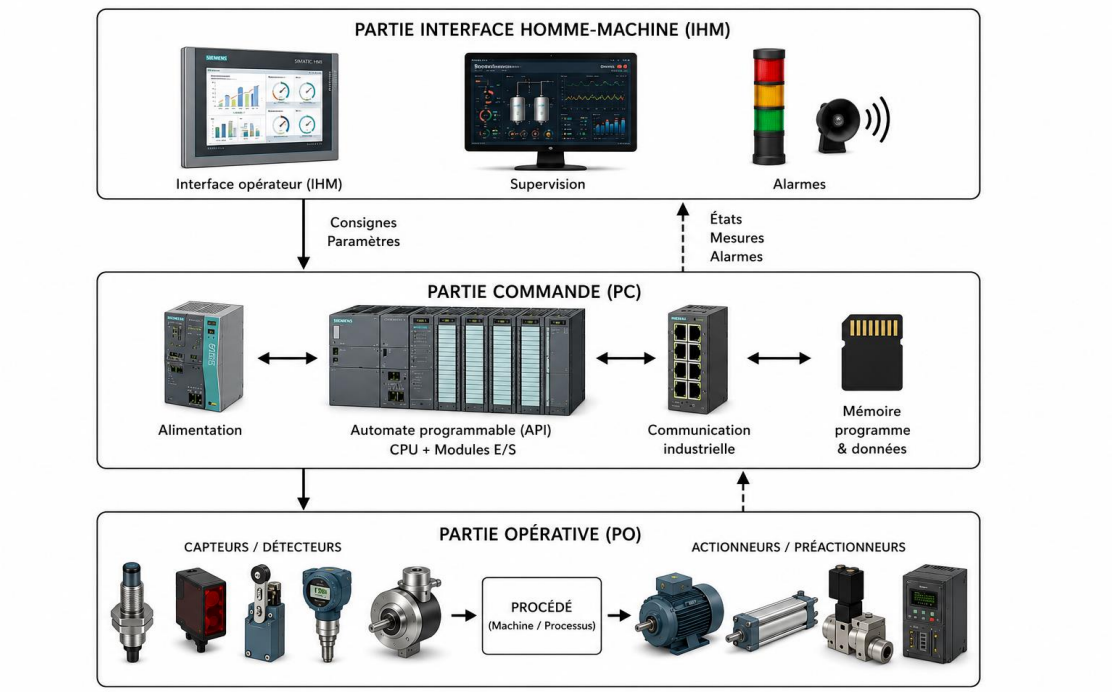


Figure 2.1 : Structure générale d'un système de production automatisé : la Partie Opérative (PO), la Partie Commande (PC) et la Partie Interface Homme-Machine (IHM)

Cette boucle fonctionne en temps réel et en continu : la partie commande lit en permanence l'état de la partie opérative, prend ses décisions et envoie les ordres appropriés, tout en informant l'opérateur via l'IHM. C'est ce cycle ininterrompu qui garantit le bon déroulement du processus automatisé.

3. Les Technologies de Commande

3.1 La Logique Câblée

3.1.1 Principe de Fonctionnement

La logique câblée, également appelée automatisme à relais, est la première technologie utilisée pour automatiser les systèmes industriels. Elle repose sur l'interconnexion physique de composants électromécaniques — principalement des relais, des contacteurs, des temporisateurs et des compteurs — dont les associations réalisent des fonctions logiques (ET, OU, NON, mémoire).

Dans un schéma à logique câblée, chaque fonction logique correspond à un câblage physique spécifique. La modification du comportement du système nécessite obligatoirement une intervention physique sur le câblage, c'est-à-dire le débranchement et le reconnectage de conducteurs électriques.

3.1.2 Avantages et Limites

La logique câblée présente certains avantages dans des contextes spécifiques :

Robustesse : les composants électromécaniques sont résistants aux environnements sévères (vibrations, températures extrêmes),

Simplicité de diagnostic : l'état de chaque relais est directement visible et mesurable,

Coût faible pour les automatismes très simples.

Cependant, ses limites sont considérables pour les applications industrielles modernes :

Rigidité : toute modification du comportement implique un recâblage complet,

Volumétrie : les armoires électriques à relais sont encombrantes et lourdes,

Fiabilité limitée : les contacts mécaniques s'usent et sont sujets à des défaillances,

Impossibilité de traiter des variables analogiques ou des algorithmes complexes,

Absence de communication avec les systèmes informatiques.

Ces limitations ont conduit au développement de la logique programmée et, en particulier, des automates programmables industriels.

3.2 La Logique Programmée

3.2.1 Principe et Avantages

La logique programmée repose sur l'exécution d'un programme stocké dans une mémoire par un processeur. Au lieu d'être figée dans le câblage, la logique de commande est définie dans un programme modifiable, ce qui confère au système une flexibilité considérable.

Les avantages de la logique programmée par rapport à la logique câblée sont déterminants :

Flexibilité : la modification du comportement du système nécessite uniquement une modification du programme, sans toucher au câblage,

Compacité : un seul automate programmable peut remplacer des centaines de relais,

Capacités étendues : traitement de variables analogiques, calculs arithmétiques, gestion de communications réseau,

Diagnostic avancé : autodiagnostic permanent, historisation des défauts, visualisation en ligne de l'état du programme,

Communicabilité : interfaces réseau intégrées permettant la connexion aux systèmes de supervision et aux réseaux d'entreprise.

3.3 L'Automate Programmable Industriel (API)

L'automate programmable industriel (API), désigné en anglais par le terme PLC (Programmable Logic Controller), est le composant central de la logique programmée dans les systèmes industriels. Il s'agit d'un calculateur numérique spécialement conçu pour le contrôle des processus industriels en temps réel, dans des environnements contraignants (températures, vibrations, perturbations électromagnétiques). [27]

3.3.1 Architecture Interne

Un API est constitué de plusieurs modules fonctionnels interconnectés :

L'unité centrale (CPU — Central Processing Unit) est le cœur de l'automate. Elle contient le processeur qui exécute le programme de commande, ainsi que les mémoires nécessaires à son fonctionnement :

Mémoire programme : stocke le programme applicatif développé par l'ingénieur automaticien,

Mémoire données : stocke les variables utilisées par le programme (états des entrées/sorties, valeurs des temporisateurs, compteurs, variables internes),

Mémoire système : stocke le système d'exploitation de l'automate.

Les modules d'entrées/sorties (E/S) constituent l'interface entre l'automate et le monde physique :

Entrées TOR : reçoivent les signaux binaires des capteurs et détecteurs (0V ou 24V),

Entrées analogiques : convertissent les signaux analogiques des capteurs (0-10V, 4-20mA) en valeurs numériques exploitables par le CPU,

Sorties TOR : envoient des signaux binaires vers les préactionneurs,

Sorties analogiques : génèrent des signaux analogiques pour le contrôle des actionneurs proportionnels (variateurs de fréquence, vannes proportionnelles).

L'alimentation convertit la tension du réseau électrique (230V AC) en tensions continues stabilisées (24V DC, 5V DC) nécessaires aux circuits électroniques de l'automate.

Les modules de communication permettent à l'automate de s'connecter aux réseaux industriels (Profinet, Modbus TCP, Ethernet/IP) pour échanger des données avec d'autres automates, des systèmes de supervision ou des passerelles IoT.

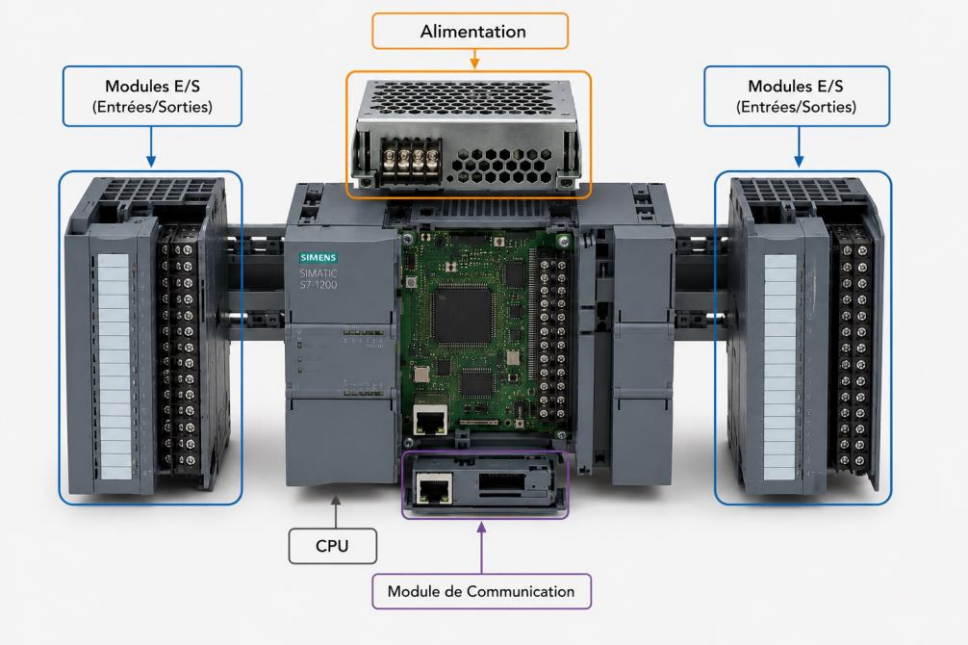


Figure 2.2 : Architecture interne d'un automate programmable industriel (CPU, modules E/S, alimentation, communication)

3.3.2 Cycle de Fonctionnement

Le fonctionnement d'un API est cyclique et déterministe. À chaque cycle — dont la durée est appelée **temps de cycle** et varie typiquement entre 1 et 100 ms selon la complexité du programme — l'automate exécute séquentiellement trois phases :

Phase 1 — Lecture des entrées : l'automate lit l'état de toutes ses entrées physiques (capteurs, boutons, signaux analogiques) et copie leurs valeurs dans une zone mémoire appelée **image des entrées**. Cette image reste figée pendant toute la durée du cycle.

Phase 2 — Exécution du programme : l'automate exécute le programme de commande instruction par instruction, en utilisant l'image des entrées pour ses calculs et en écrivant les résultats dans une zone mémoire appelée **image des sorties**.

Phase 3 — Écriture des sorties : l'automate transfère le contenu de l'image des sorties vers les sorties physiques, activant ou désactivant les préactionneurs correspondants.

Ce fonctionnement cyclique garantit le **déterminisme** du système : pour un même état des entrées, le comportement du programme est toujours identique et prévisible, ce qui est une exigence fondamentale en automatisme industriel.

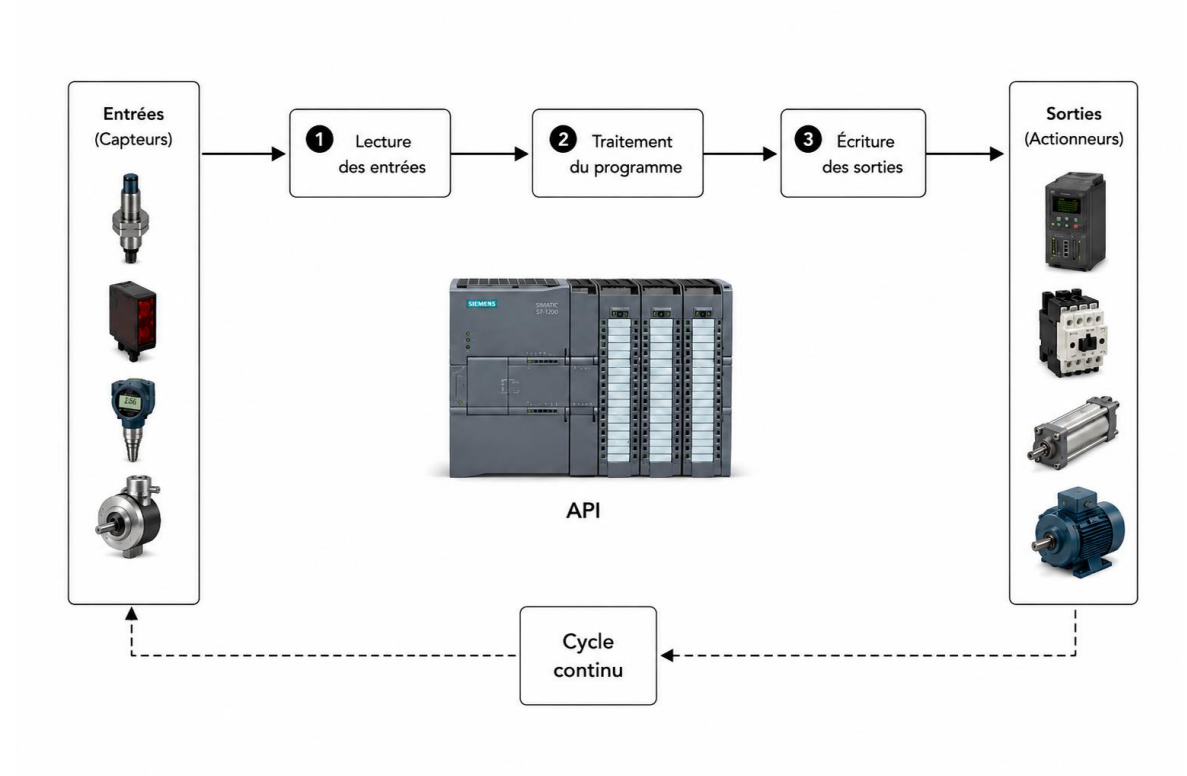


Figure 2.3 : Cycle de fonctionnement d'un API : lecture des entrées, exécution du programme, écriture des sorties

3.3.3 Langages de Programmation — Norme IEC 61131-3

La norme internationale **IEC 61131-3** définit cinq langages standardisés pour la programmation des automates programmables industriels. Cette standardisation garantit la portabilité des programmes entre automates de différents fabricants. [27][51]

Le Ladder Diagram (LD) est le langage graphique le plus répandu. Il représente le programme sous forme de schémas à contacts, directement inspirés des schémas électriques à relais. Il est particulièrement adapté aux automatismes séquentiels et aux logiques combinatoires simples, et est facilement compréhensible par les électriciens et automaticiens.

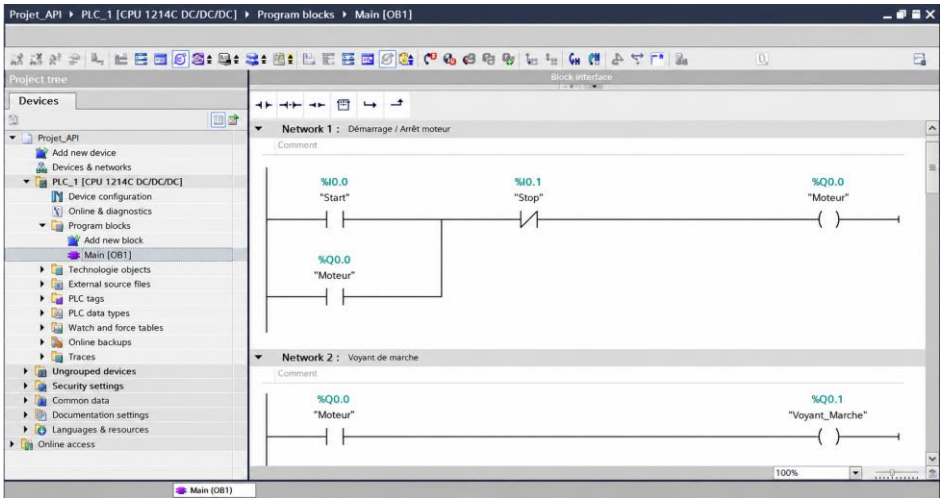


Figure 2.4 : Exemple de programme en langage Ladder (LAD)

Le Sequential Function Chart (SFC) est un langage graphique de description des comportements séquentiels, basé sur les réseaux de Petri. Il décompose le programme en étapes et en transitions, offrant une représentation claire et lisible des séquences d'opérations complexes.

Le Function Block Diagram (FBD) est un langage graphique représentant le programme sous forme de blocs fonctionnels interconnectés, similaire aux schémas de l'électronique analogique. Il est particulièrement adapté aux traitements de signaux analogiques et aux régulations.

Le Structured Text (ST) est un langage textuel de haut niveau, syntaxiquement proche du Pascal ou du C. Il permet d'écrire des algorithmes complexes (calculs mathématiques, traitement de données, communications) de manière concise et structurée. C'est le langage le plus utilisé pour les applications avancées impliquant des calculs ou des communications réseau.

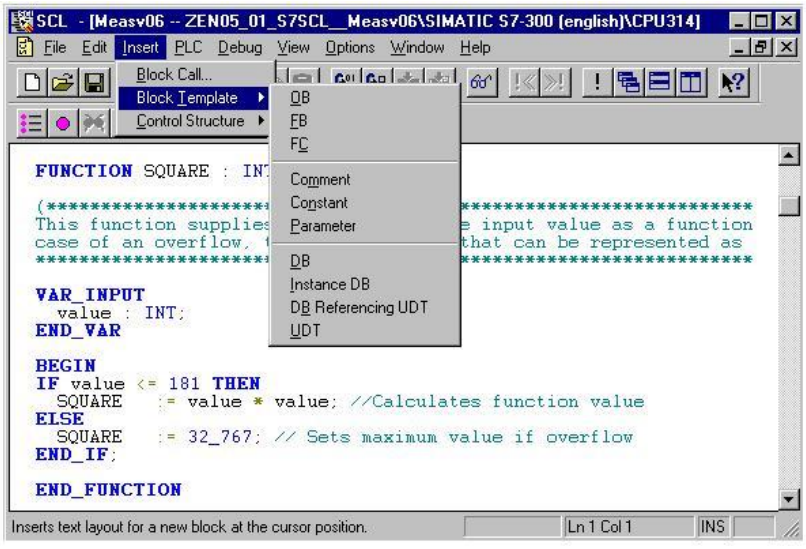


Figure 2.5 : Exemple de programme en langage Structured Text (ST)

L'Instruction List (IL) est un langage textuel bas niveau, proche de l'assembleur. Bien que toujours présent dans la norme, il est de moins en moins utilisé dans les nouvelles applications au profit du Structured Text.

4. La Communication dans les Systèmes Industriels

4.1 La Pyramide CIM

La pyramide **CIM (Computer Integrated Manufacturing)** est un modèle conceptuel qui représente l'organisation hiérarchique des systèmes d'information et de communication dans une entreprise industrielle. Elle définit cinq niveaux distincts, chacun ayant ses propres exigences en termes de vitesse de communication, de volume de données et de criticité temps réel.

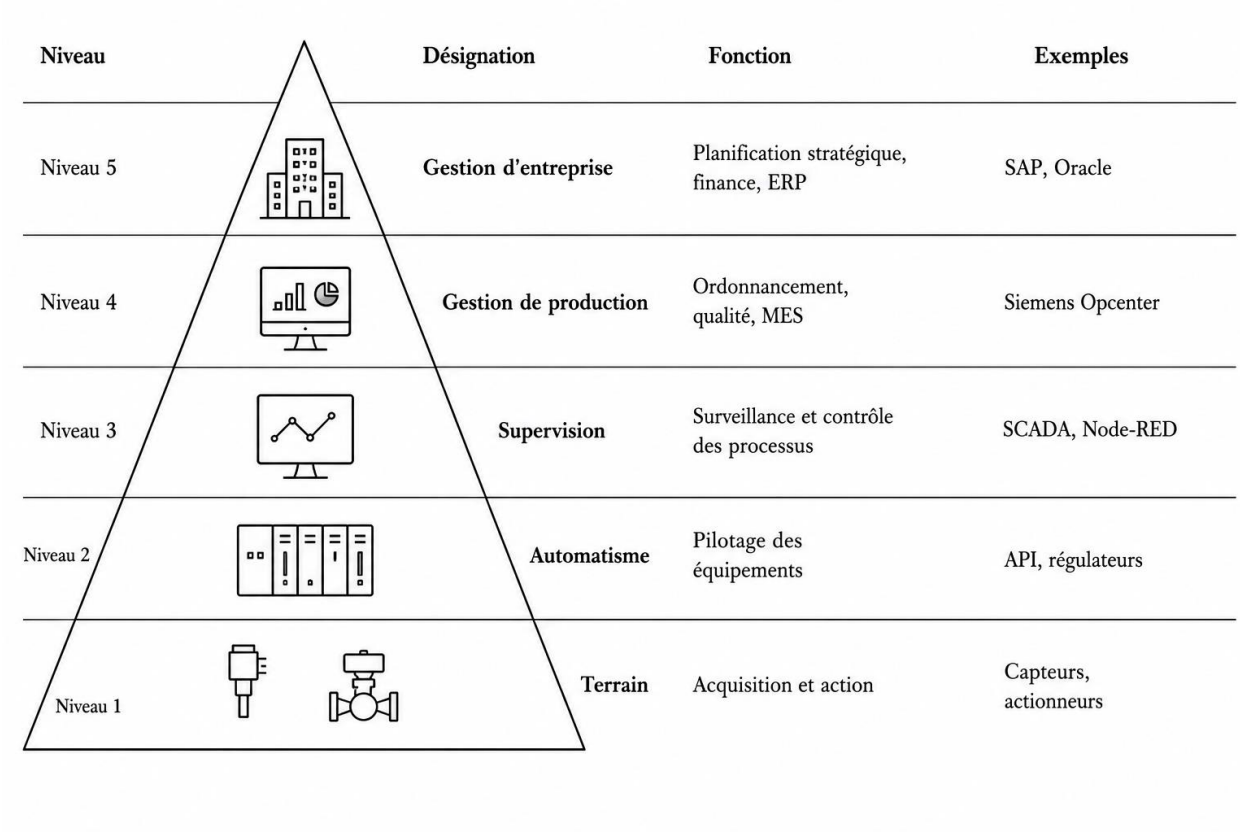


Figure 2.6 : La pyramide CIM (Computer Integrated Manufacturing) et ses cinq niveaux hiérarchiques

La communication entre ces niveaux est assurée par différents types de réseaux industriels, dont les caractéristiques varient en fonction des exigences de chaque niveau. Les niveaux inférieurs (terrain et automatisme) requièrent des communications rapides et déterministes, tandis que les niveaux supérieurs (supervision et gestion) tolèrent des latences plus importantes mais nécessitent des débits plus élevés.

4.2 Les Niveaux de Communication

Le niveau terrain regroupe les communications entre les capteurs, actionneurs et préactionneurs d'une part, et les automates programmables d'autre part. Les technologies utilisées à ce niveau sont des **bus de capteurs** à faible débit mais à temps de réponse très court (AS-Interface, IO-Link).

Le niveau automatisme assure la communication entre les automates programmables, entre les automates et les variateurs de fréquence, et entre les automates et les interfaces opérateur. Les **bus de terrain** (Profibus, CANopen, DeviceNet) et les réseaux **Ethernet industriels** (Profinet, Ethernet/IP, Modbus TCP) sont utilisés à ce niveau.[10]

Le niveau supervision connecte les automates aux systèmes de supervision (SCADA), aux bases de données historiques et aux passerelles vers les niveaux supérieurs. Les réseaux **Ethernet standard** et les protocoles **OPC-UA** sont privilégiés à ce niveau, offrant une interopérabilité maximale avec les systèmes informatiques d'entreprise.

4.3 Les Principaux Protocoles Industriels

4.3.1 Modbus TCP

Modbus est le protocole industriel le plus ancien et le plus répandu dans le monde. Initialement développé par Modicon en 1979 pour les communications série (RS-232, RS-485), il a été adapté pour les réseaux Ethernet sous la forme de **Modbus TCP**, utilisant le port TCP 502.[29]

Modbus TCP repose sur un modèle **maître/esclave** (ou client/serveur dans sa terminologie moderne) : un seul maître initie toutes les transactions de communication, et les esclaves répondent aux requêtes. Les données sont organisées en quatre zones mémoire standardisées :

Coils (bobines) : variables binaires en lecture/écriture (sorties TOR de l'automate),

Discrete Inputs : variables binaires en lecture seule (entrées TOR de l'automate),

Holding Registers : registres 16 bits en lecture/écriture (paramètres, consignes),

Input Registers : registres 16 bits en lecture seule (mesures analogiques).

Sa simplicité, son universalité et son caractère libre de droits en font le protocole de référence pour la collecte de données depuis les équipements industriels, y compris dans les architectures IoT industrielles. Sa principale limitation réside dans l'absence de mécanismes de sécurité natifs, imposant une segmentation réseau stricte dans les déploiements exposés.

4.3.2 Profinet

Profinet (Process Field Network) est le protocole Ethernet industriel développé par Siemens et standardisé par l'organisation Profibus & Profinet International (PI).[33] Il constitue l'évolution naturelle du bus de terrain Profibus vers les réseaux Ethernet et est largement déployé dans les installations utilisant des équipements Siemens.

Profinet se distingue par ses performances temps réel exceptionnelles définies selon trois classes:

Profinet RT (Real Time) : latence inférieure à 10 ms, pour la majorité des applications d'automatisme,

Profinet IRT (Isochronous Real Time) : latence inférieure à 1 ms, pour les applications à synchronisation très stricte (commandes d'axes, robotique).[25]

4.3.3 OPC-UA

OPC-UA (OPC Unified Architecture) est le standard de communication de l'Industrie 4.0, développé par la OPC Foundation et standardisé par la norme IEC 62541. Il se distingue fondamentalement des protocoles précédents par son approche orientée **modèle d'information** plutôt que simple échange de données brutes. .[28][34]

OPC-UA offre plusieurs caractéristiques qui en font le protocole de référence pour les nouvelles installations industrielles :

Indépendance de la plateforme : fonctionne sur Windows, Linux, systèmes embarqués,

Sécurité intégrée : authentification, chiffrement et signature des messages sont natifs,

Modèle d'information riche : les données sont accompagnées de métadonnées décrivant leur signification, leurs unités et leurs relations,

Double mode : client/serveur pour les accès à la demande, Publish/Subscribe pour la télémétrie en temps réel.

PROTOCOLE	MODELE	TRANSPORT	SECURITE NATIVE	USAGE PRINCIPAL
MODBUS TCP	Maître/Esclave	TCP/502	Non	Équipements existants
PROFINET	Maître/Esclave	Ethernet	Partielle	Équipements Siemens
OPC-UA	Client/Serveur + Pub/Sub	TCP	Oui	Industrie 4.0

Tableau 2.1: Comparaison entre Modbus TCP, Profinet et OPC-UA

5. La Supervision Industrielle

5.1 Définition et Rôle

La **supervision industrielle** désigne l'ensemble des moyens techniques et logiciels permettant à un opérateur de surveiller, contrôler et optimiser un processus industriel depuis un poste centralisé. Elle constitue le niveau de commande le plus élevé dans la pyramide CIM, au-dessus de l'automatisme.

Un système de supervision remplit une mission fondamentale : **transformer les données brutes du terrain en informations exploitables** pour les opérateurs et les responsables de production. Il établit ainsi le lien entre le monde physique des machines et le monde décisionnel de l'entreprise.[32]

5.2 Les Fonctions d'un Système de Supervision

Un système de supervision industrielle complet remplit plusieurs fonctions essentielles :

5.2.1 Acquisition et Visualisation en Temps Réel

La fonction principale d'un système de supervision est de collecter en permanence les données provenant des automates et des instruments de terrain, et de les présenter à l'opérateur sous forme de **synoptiques** — représentations graphiques animées du processus industriel. Les valeurs numériques (températures, pressions, débits) sont affichées en temps réel, et l'état des équipements (marche/arrêt, ouvert/fermé) est représenté visuellement par des animations et des codes couleur.

5.2.2 Historisation et Archivage

Toutes les valeurs mesurées et tous les événements du système sont horodatés et archivés dans une **base de données historique**. Ces données constituent une mémoire du processus indispensable pour l'analyse des performances, la détection des dérives, la traçabilité de la production et la conformité réglementaire.

5.2.3 Gestion des Alarmes et des Événements

Le système de supervision surveille en permanence les valeurs mesurées par rapport à des **seuils prédéfinis**. Lorsqu'une valeur dépasse un seuil critique, une alarme est générée, notifiée à l'opérateur (visuellement et soniquement) et enregistrée dans un journal des alarmes. La gestion des alarmes est une fonction critique pour la sûreté des installations industrielles.

5.2.4 Génération de Rapports

Les systèmes de supervision modernes génèrent automatiquement des **rapports de production** périodiques (horaires, journaliers, hebdomadaires) compilant les indicateurs de performance clés : taux de disponibilité, cadences de production, consommations d'énergie, nombre et durée des arrêts.

5.3 Du SCADA Classique vers la Supervision Connectée

5.3.1 Les Systèmes SCADA Traditionnels

Les systèmes **SCADA (Supervisory Control and Data Acquisition)** traditionnels sont des logiciels de supervision s'exécutant sur des postes de travail industriels dédiés, connectés aux automates via des réseaux industriels propriétaires. Ils offrent des fonctionnalités avancées mais présentent des limitations significatives dans le contexte actuel :

Architecture fermée : les données restent confinées au réseau industriel local,

Accès limité : l'opérateur doit être physiquement présent dans la salle de contrôle,

Interopérabilité réduite : les systèmes de différents fabricants communiquent difficilement entre eux,

Coûts élevés : les licences logicielles et le matériel dédié représentent des investissements importants.

5.3.2 Les Limites des Systèmes Classiques face aux Exigences Modernes

L'industrie contemporaine impose de nouvelles exigences auxquelles les SCADA traditionnels répondent difficilement :

Accès distant : la nécessité de surveiller les installations depuis n'importe quel endroit, notamment pour la maintenance à distance,

Intégration multi-sites : la supervision centralisée de plusieurs usines géographiquement dispersées,

Analyse avancée : l'exploitation des données historiques par des algorithmes d'intelligence artificielle pour la maintenance prédictive,

Flexibilité et évolutivité : l'intégration rapide de nouveaux équipements sans refonte complète de l'architecture.

5.3.3 L'Évolution vers la Supervision Distante et Connectée

Face à ces limitations, une nouvelle génération de systèmes de supervision a émergé, s'appuyant sur les technologies Internet et IoT pour offrir une supervision accessible, ouverte et intelligente. Cette évolution se caractérise par :

L'architecture web : les interfaces de supervision sont accessibles via un navigateur standard, sans installation de logiciel dédié.[23]

L'ouverture des données : les données industrielles sont exposées via des API standardisées (REST, MQTT), permettant leur exploitation par des systèmes tiers.[24]

L'intégration Cloud : le stockage et le traitement des données dans des infrastructures Cloud permettent une scalabilité et une disponibilité maximale.

L'Edge Computing : le traitement local des données au niveau des passerelles IoT réduit la latence et la dépendance à la connectivité réseau.

5.3.4 Le Rôle de l'IoT dans la Supervision Moderne

L'Internet des Objets industriel (IIoT) joue un rôle central dans cette transformation. En équipant les systèmes automatisés existants de **passerelles IoT** capables de lire les données des automates via les protocoles industriels standards et de les transmettre vers des plateformes de supervision modernes, il devient possible de **connecter l'héritage industriel** aux technologies numériques actuelles sans remplacer les équipements existants.[30]

Cette approche — souvent désignée par le terme **retrofitting** ou modernisation non-intrusive — offre un rapport coût/bénéfice particulièrement attractif pour les installations industrielles disposant d'automates programmables récents mais dépourvus de connectivité avancée. Elle constitue l'un des axes majeurs du déploiement de l'Industrie 4.0 dans le tissu industriel existant.[26][31]

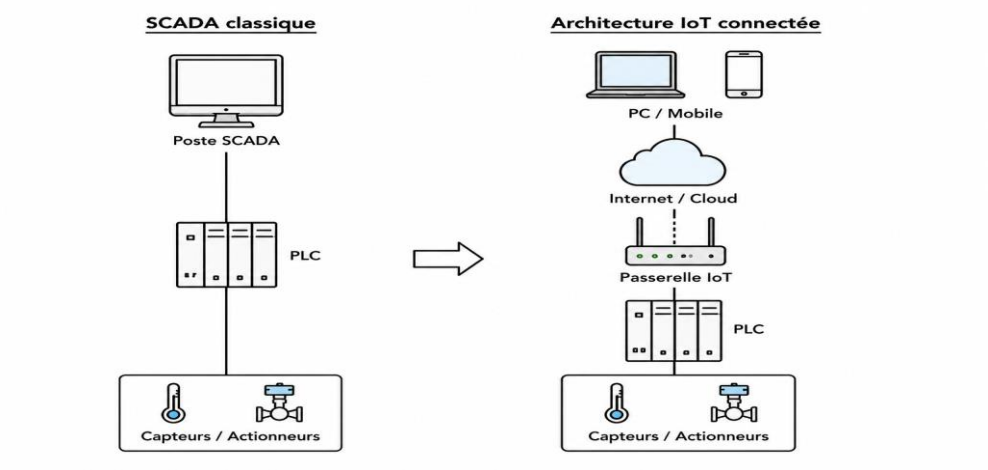


Figure 2.7 : Évolution de la supervision industrielle : du SCADA classique vers l'architecture IoT connectée

6. Conclusion

Ce chapitre a établi les fondements théoriques nécessaires à la compréhension des systèmes de production automatisés et de leur supervision. L'analyse de la structure en trois parties — partie opérative, partie commande et interface homme-machine — a mis en évidence l'organisation cohérente et hiérarchique qui gouverne tout système industriel automatisé.

L'étude comparative entre la logique câblée et la logique programmée a démontré la supériorité de l'automate programmable industriel pour les applications modernes, grâce à sa flexibilité, ses capacités de communication et son aptitude à s'intégrer dans des architectures numériques avancées. La maîtrise de son architecture interne et de son cycle de fonctionnement est indispensable pour concevoir des solutions d'interfaçage fiables avec des systèmes externes.

L'analyse des protocoles de communication industriels a révélé la complémentarité entre Modbus TCP — protocole universel pour la collecte de données depuis les équipements existants — et OPC-UA — standard d'interopérabilité de l'Industrie 4.0. Ces protocoles constituent les briques de communication sur lesquelles reposent les architectures IoT industrielles modernes.

Enfin, l'étude de l'évolution de la supervision industrielle, du SCADA classique vers les architectures connectées et ouvertes, a mis en lumière le rôle déterminant de l'IoT comme catalyseur de cette transformation. C'est précisément cette convergence entre automatisme industriel et technologies IoT qui constitue le cœur du projet développé dans les chapitres suivants.

Chapitre 3 : Conception d'un Système IoT de Supervision Industrielle Universelle

1. Introduction

L'industrie moderne est confrontée à un défi fondamental : les systèmes de production automatisés, aussi performants soient-ils, demeurent pour la plupart isolés du monde numérique. Les automates programmables industriels qui pilotent ces systèmes exécutent leurs programmes avec précision et fiabilité, mais les données qu'ils génèrent en permanence — états des entrées et sorties, valeurs analogiques, variables internes — restent confinées dans leur mémoire interne, inaccessibles à distance et inexploitées au-delà du réseau industriel local.

Cette réalité engendre des contraintes opérationnelles considérables : l'opérateur doit être physiquement présent pour connaître l'état du système, les interventions de maintenance sont réactives plutôt que préventives, et aucune historisation des données de production n'est disponible pour l'analyse et l'optimisation des performances.

C'est précisément pour répondre à cette problématique que s'inscrit le présent projet. L'objectif est de concevoir une solution IoT industrielle permettant de connecter des automates programmables existants à une infrastructure de supervision moderne, accessible à distance, sans modifier ni interrompre leur fonctionnement. Ce chapitre présente le principe général, l'architecture globale et les composants logiciels clés de cette solution, conçue pour être universelle et applicable à tout processus industriel piloté par un automate programmable.

2. Problématique et Objectifs

2.1 Problématique Générale

Les systèmes de production industriels actuels reposent massivement sur des automates programmables industriels fiables et performants. Cependant, ces équipements ont été conçus et déployés dans une logique d'automatisme local : ils pilotent efficacement leurs processus, mais ne disposent d'aucune connectivité native vers les systèmes d'information modernes ou vers Internet.

Cette absence de connectivité se traduit par plusieurs limitations opérationnelles majeures qui pèsent sur la performance et la compétitivité des installations industrielles :

L'isolement informationnel constitue la première limitation. Les données générées par le processus industriel — températures, pressions, états des actionneurs, cadences de production — restent enfermées dans la mémoire de l'automate. Elles ne sont ni transmises, ni stockées, ni exploitées au-delà du cycle de scrutation de l'automate lui-même. Une donnée qui n'est pas transmise est une donnée perdue.

La contrainte de présence physique représente la deuxième limitation. Pour connaître l'état d'un système automatisé, l'opérateur doit se rendre physiquement sur site, devant le pupitre de commande ou l'interface locale. Cette contrainte est incompatible avec les exigences de réactivité et de flexibilité de l'industrie moderne, où la supervision à distance est devenue une nécessité opérationnelle.

L'absence d'historisation constitue la troisième limitation. Sans système de collecte et d'archivage des données, il est impossible d'analyser les tendances, de détecter les dérives de performance, d'anticiper les pannes ou de générer des rapports de production. L'industrie fonctionne alors en mode réactif plutôt qu'en mode prédictif.

Le coût prohibitif des solutions SCADA propriétaires représente enfin un obstacle majeur pour de nombreuses installations. Les systèmes de supervision traditionnels nécessitent des licences logicielles coûteuses, du matériel dédié et des compétences spécialisées, rendant leur déploiement inaccessible pour une grande partie du tissu industriel.

Face à ces limitations, une question fondamentale se pose : **comment connecter des automates programmables industriels existants à une infrastructure de supervision moderne, économique et accessible, sans modifier leur programme ni interrompre leur fonctionnement ?**

2.2 Objectifs du Système

Le système développé dans ce projet répond à cette question par une approche non-intrusive, économique et universelle. Les objectifs fixés sont les suivants :

Objectif principal : Concevoir et réaliser une plateforme IoT de supervision industrielle permettant de collecter en temps réel les données de n'importe quel automate programmable Siemens S7-1200, de les transmettre via un réseau de communication standard, de les stocker dans une base de données et de les visualiser sur une interface accessible depuis n'importe quel appareil connecté.

Objectifs secondaires :

Assurer la **non-intrusivité** du système : le PLC continue à fonctionner normalement, sans modification de son programme ni interruption de sa production,

Garantir l'**universalité** de la solution : le système doit être applicable à tout processus industriel piloté par un automate programmable, indépendamment de la nature du processus,

Assurer la **scalabilité** de l'architecture : le système doit pouvoir superviser simultanément plusieurs automates et plusieurs processus industriels depuis une interface unifiée,

Utiliser exclusivement des **technologies open-source** pour garantir l'accessibilité économique de la solution et son indépendance vis-à-vis des éditeurs propriétaires,

Offrir une **accessibilité multi-appareils** : l'interface de supervision doit être consultable depuis un ordinateur, une tablette ou un téléphone mobile, depuis n'importe quel endroit disposant d'une connexion réseau.

2.3 Cahier des Charges Fonctionnel

Le tableau suivant synthétise les exigences fonctionnelles et non fonctionnelles auxquelles le système doit satisfaire :

EXIGENCE	DESCRIPTION	PRIORITE
LECTURE DES DONNEES PLC	Lire en temps réel les entrées TOR, sorties TOR, entrées analogiques et mémoires internes du PLC	Essentielle
TRANSMISSION DES DONNEES	Transmettre les données collectées vers un serveur via un protocole de communication standard	Essentielle
STOCKAGE ET HISTORISATION	Archiver toutes les données horodatées dans une base de données persistante	Essentielle
VISUALISATION EN TEMPS REEL	Afficher l'état du système sur une interface graphique mise à jour en continu	Essentielle
NON-INTRUSIVITE	Fonctionner sans modifier le programme de l'automate ni interrompre la production	Essentielle
UNIVERSALITE	Être applicable à tout processus industriel piloté par un API S7-1200	Essentielle
SCALABILITE	Superviser plusieurs PLC et plusieurs processus simultanément	Importante
ACCESSIBILITE DISTANTE	Interface consultable depuis tout appareil connecté	Importante
TECHNOLOGIES OPEN-SOURCE	Utiliser exclusivement des outils libres et gratuits	Importante
GESTION DES ALARMES	Détecter et signaler les dépassements de seuils	Secondaire
GENERATION DE RAPPORTS	Produire des rapports de production périodiques	Secondaire

Tableau 3.1 : Cahier des charges fonctionnel du système de supervision

3. Principe Général du Système

3.1 Idée Fondamentale du Projet

L'idée centrale du projet repose sur un concept simple mais puissant : **placer une passerelle intelligente entre l'automate programmable et Internet**.

Cette passerelle — matérialisée par un micro-ordinateur embarqué — se connecte discrètement à l'automate via son interface Ethernet, lit périodiquement les données de sa mémoire, et les transmet vers une infrastructure de supervision distante. Du point de vue de l'automate, rien ne change : il continue à exécuter son programme de commande, à lire ses capteurs et à piloter ses actionneurs exactement comme avant. La passerelle agit comme un observateur silencieux, invisible pour le processus industriel.

Cette approche est fondamentalement différente des solutions SCADA traditionnelles qui nécessitent une intégration profonde avec l'automate et une modification de sa configuration.

Ici, le principe est celui du **retrofitting non-intrusif** : on ajoute de la connectivité à un système existant sans le modifier, exactement comme on branche un enregistreur de données sur un équipement de mesure sans modifier cet équipement.

3.2 Principe de Fonctionnement

Le fonctionnement du système suit un cycle continu et automatique, composé de cinq étapes séquentielles qui se répètent indéfiniment tant que le système est en marche :

Étape 1 — Pilotage du processus industriel : L'automate programmable exécute son programme de commande de manière autonome et continue. Il lit l'état de ses capteurs, prend ses décisions logiques et pilote ses actionneurs. Les résultats de ce traitement sont stockés dans sa mémoire interne sous forme d'entrées, de sorties et de variables internes.

Étape 2 — Lecture des données par la passerelle : La passerelle IoT, connectée au PLC via le réseau Ethernet local, interroge périodiquement la mémoire de l'automate grâce à la bibliothèque de communication Snap7. Elle lit l'état de toutes les zones mémoire configurées : entrées physiques, sorties physiques, variables internes et blocs de données. Cette lecture est non-intrusive et n'interfère en aucun cas avec l'exécution du programme de l'automate.

Étape 3 — Structuration et publication des données : Les données brutes lues depuis le PLC sont converties par la passerelle en messages structurés au format JSON, horodatés avec précision. Ces messages sont ensuite publiés vers un broker de messagerie via le protocole MQTT, chaque type de donnée étant associé à un topic hiérarchique spécifique.

Étape 4 — Réception et stockage par le serveur : Le serveur applicatif, abonné aux topics MQTT correspondants, reçoit les messages publiés par la passerelle. Il les traite, les valide et les stocke dans une base de données temporelle, constituant ainsi un historique complet et horodaté de l'état du processus industriel.

Étape 5 — Visualisation sur l'interface de supervision : L'interface de supervision interroge le serveur et affiche en temps réel l'état du système sous forme de tableaux de bord graphiques. L'opérateur peut consulter cette interface depuis n'importe quel appareil connecté — ordinateur, tablette ou smartphone — depuis n'importe quel endroit disposant d'un accès réseau.

3.3 Le Flux de Données — Du Terrain vers la Supervision

Le chemin parcouru par une donnée depuis sa naissance dans le monde physique jusqu'à son affichage sur l'interface de supervision peut être décrit comme suit :

Un capteur physique connecté à l'automate — un détecteur de présence, un capteur de température, un fin de course — détecte une grandeur physique et la convertit en signal électrique. L'automate lit ce signal à chaque cycle de scrutation et stocke sa valeur dans sa mémoire interne. La passerelle lit cette valeur dans la mémoire de l'automate via Snap7, la structure en message JSON et la publie via MQTT. Le broker reçoit et redistribue le message au serveur. Le serveur le stocke dans la base de données et le met à disposition de l'interface. L'interface affiche la valeur à l'opérateur en temps réel.

Ce flux, représenté schématiquement ci-dessous, est **unidirectionnel et continu** :

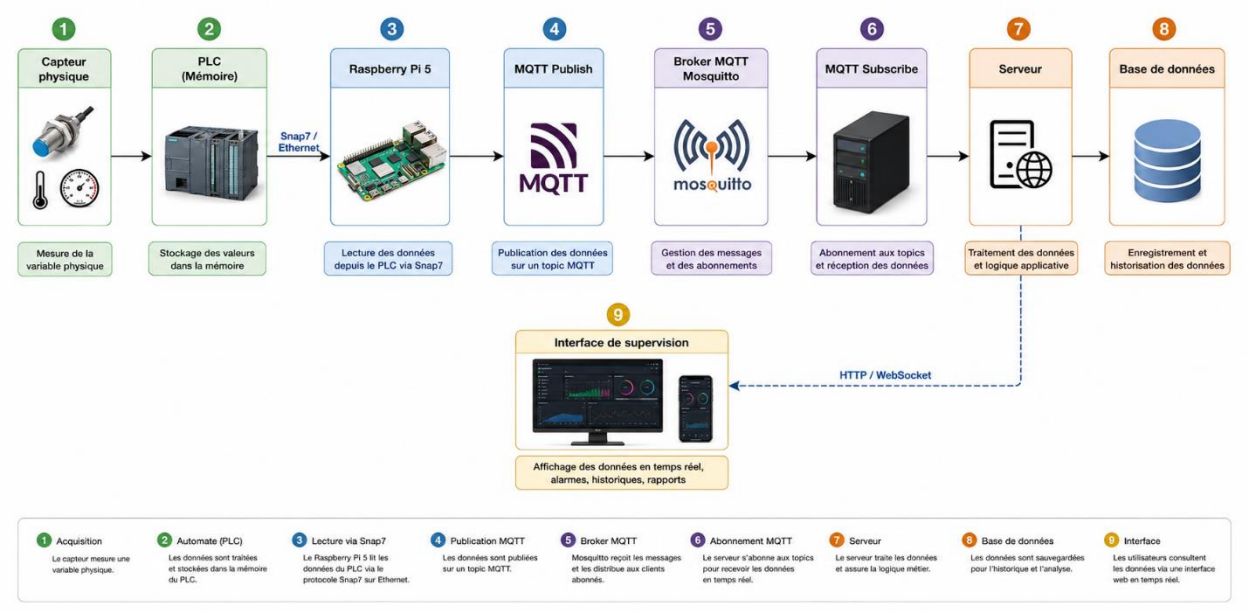


Figure 3.1 : Schéma du flux de données complet — du capteur physique à l'interface de supervision

4. Architecture Globale du Système

4.1 Vue d'Ensemble de l'Architecture

L'architecture du système est organisée en **cinq couches fonctionnelles** distinctes et hiérarchisées, chacune ayant un rôle précis et des responsabilités clairement définies. Cette organisation en couches garantit la modularité du système : chaque couche peut être modifiée, remplacée ou améliorée indépendamment des autres, sans remettre en cause l'ensemble de l'architecture.

La ***Figure 3.2*** — reproduite ci-après — présente l'architecture complète du système. Elle constitue la représentation centrale du projet et synthétise en une seule image l'ensemble des composants, leurs rôles et leurs interactions.

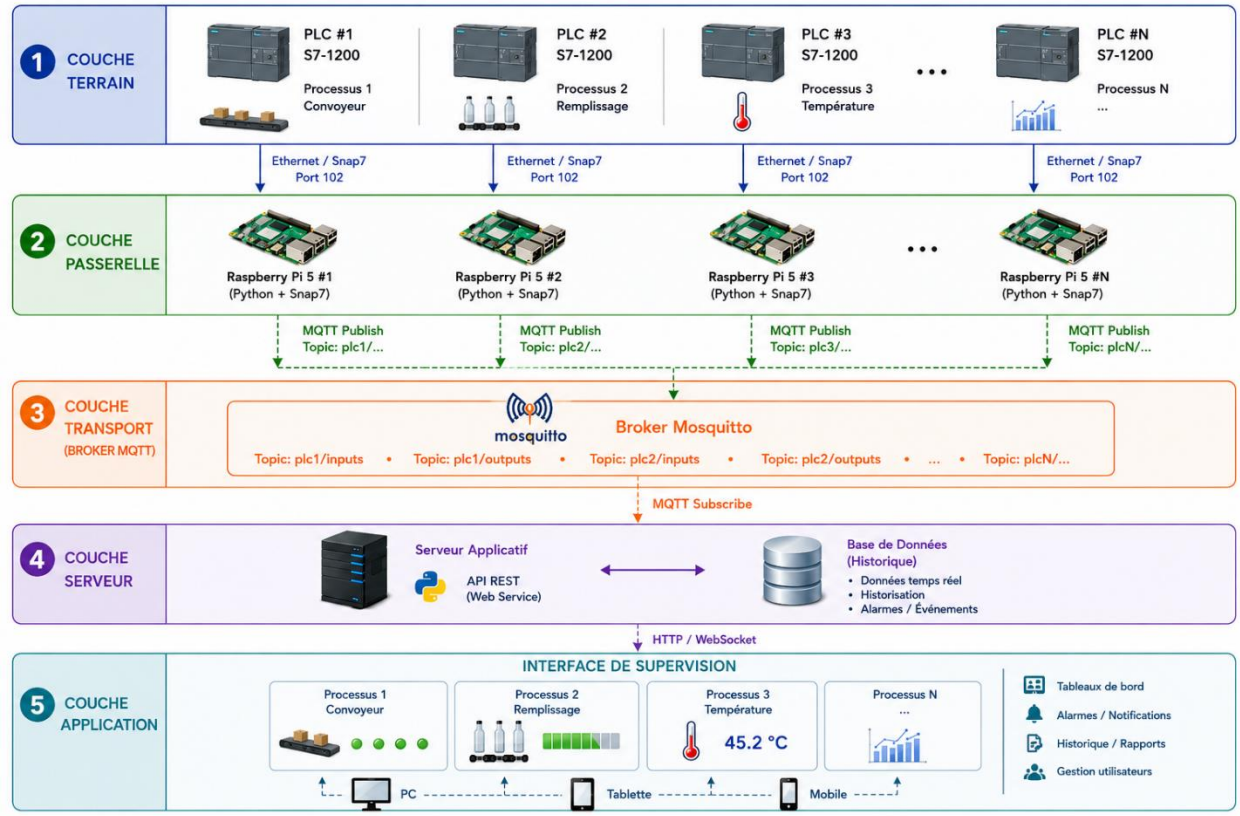


Figure 3.2 : Architecture complète du système IoT de supervision industrielle universelle

Cette architecture se distingue par deux caractéristiques essentielles. D'une part, elle est **universelle** : la même architecture s'applique quel que soit le processus industriel supervisé, qu'il s'agisse d'un convoyeur, d'un système de remplissage, d'un processus thermique ou de tout autre application. D'autre part, elle est **scalable** : le même serveur, le même broker et la même interface peuvent gérer simultanément N automates et N processus industriels, simplement en ajoutant une passerelle par automate supplémentaire.

4.2 Couche Terrain — L'Automate Programmable Industriel

La couche terrain constitue le point de départ du système. Elle regroupe l'ensemble des automates programmables industriels qui pilotent les processus de production.

Chaque automate programmable fonctionne de manière totalement autonome et indépendante. Il exécute son programme de commande, gère ses entrées et sorties physiques, et pilote le processus industriel qui lui est confié. Son fonctionnement n'est en aucun cas affecté par la présence du système de supervision : il ignore complètement que ses données sont lues et transmises vers l'extérieur.

Les données disponibles dans la mémoire de l'automate et accessibles par le système de supervision sont organisées en zones mémoire standardisées, décrites dans le tableau suivant :

ZONE MEMOIRE	DESIGNATION	CONTENU	TYPE DE DONNEE
I (INPUTS)	Entrées physiques	État des capteurs et détecteurs connectés	Binaire (TOR)
Q (OUTPUTS)	Sorties physiques	État des actionneurs et préactionneurs commandés	Binaire (TOR)
IW / QW	Entrées/Sorties analogiques	Valeurs mesurées ou générées en analogique	Entier 16 bits
M (MERKERS)	Mémoires internes	Variables internes du programme	Binaire / Entier
DB (DATA BLOCKS)	Blocs de données	Données structurées définies par le programmeur	Tous types

Tableau3.2 : Zones mémoire de l'automate accessibles par le système de supervision

L'automate expose ces données via son interface Ethernet intégrée, en utilisant le protocole de communication natif Siemens S7. C'est ce protocole que la passerelle exploite pour accéder aux données, via la bibliothèque Snap7.[44]

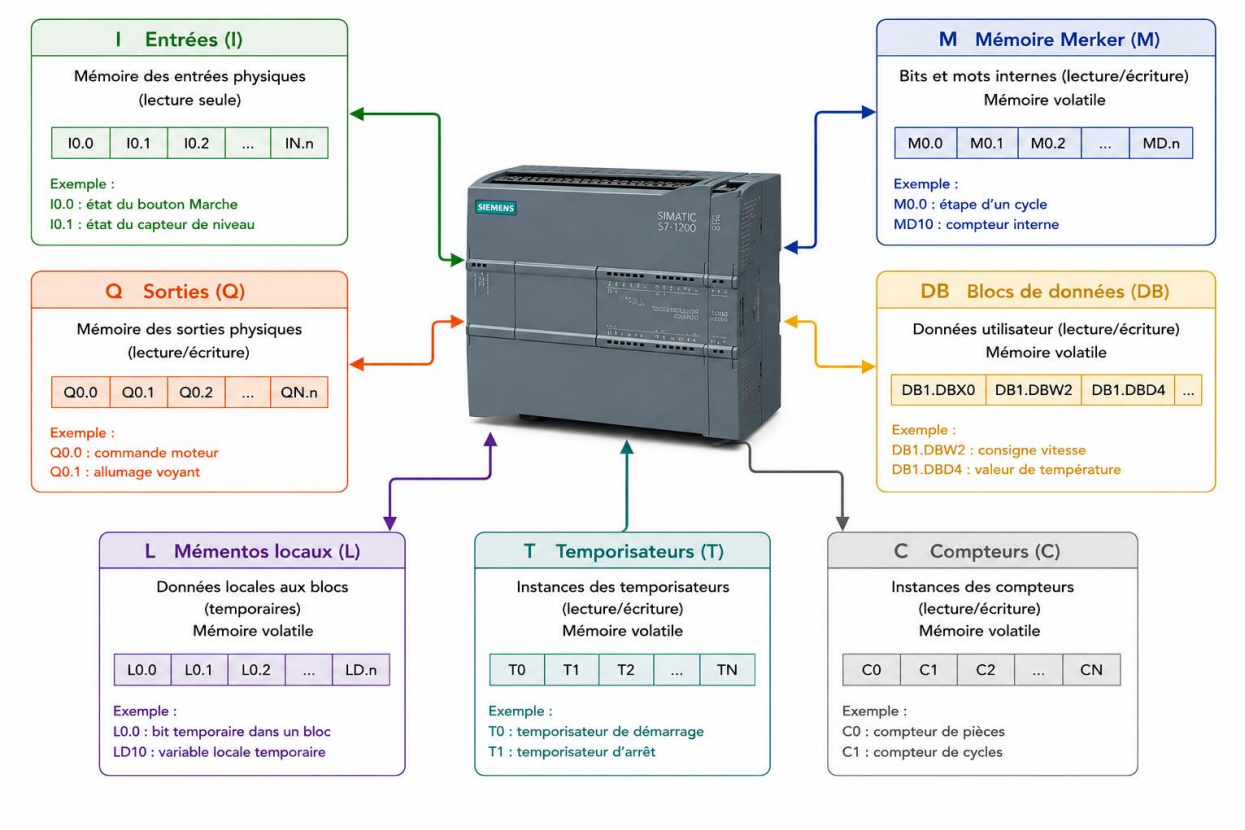


Figure 3.3: Schéma des zones mémoire accessibles dans l'automate programmable S7-1200

4.3 Couche Passerelle — Le Système Embarqué

La couche passerelle est le composant central et le plus stratégique de l'architecture. Elle assure le rôle de pont entre le monde industriel — régi par des protocoles propriétaires et des réseaux locaux fermés — et le monde IoT — caractérisé par des protocoles ouverts et une connectivité étendue.

Cette passerelle est matérialisée par un micro-ordinateur embarqué, disposant d'une connectivité Ethernet native, d'une puissance de calcul suffisante pour exécuter des **scripts Python** en continu, et d'une compatibilité totale avec les bibliothèques de communication industrielles nécessaires au projet.[40][41]

Chaque passerelle est associée à un automate programmable. Elle exécute en continu un programme Python qui accomplit trois missions fondamentales :

Mission 1 — Acquisition des données : La passerelle se connecte à l'automate via le réseau Ethernet local et interroge périodiquement sa mémoire grâce à la bibliothèque Snap7. Elle lit l'ensemble des zones mémoire configurées et récupère les valeurs courantes de toutes les variables surveillées.

Mission 2 — Traitement et structuration : Les données brutes récupérées sous forme d'octets sont décodées, converties en valeurs exploitables et structurées en messages au format JSON. Chaque message est horodaté avec précision pour garantir la traçabilité temporelle des données.

Mission 3 — Publication via MQTT : Les messages structurés sont publiés vers le broker MQTT via le réseau. Chaque type de donnée est publié sur un topic distinct, permettant au serveur de les distinguer et de les traiter de manière appropriée.

La passerelle fonctionne de manière **autonome et continue**, sans intervention humaine. Elle gère automatiquement les reconnexions en cas de coupure réseau, garantissant ainsi la continuité de la supervision.

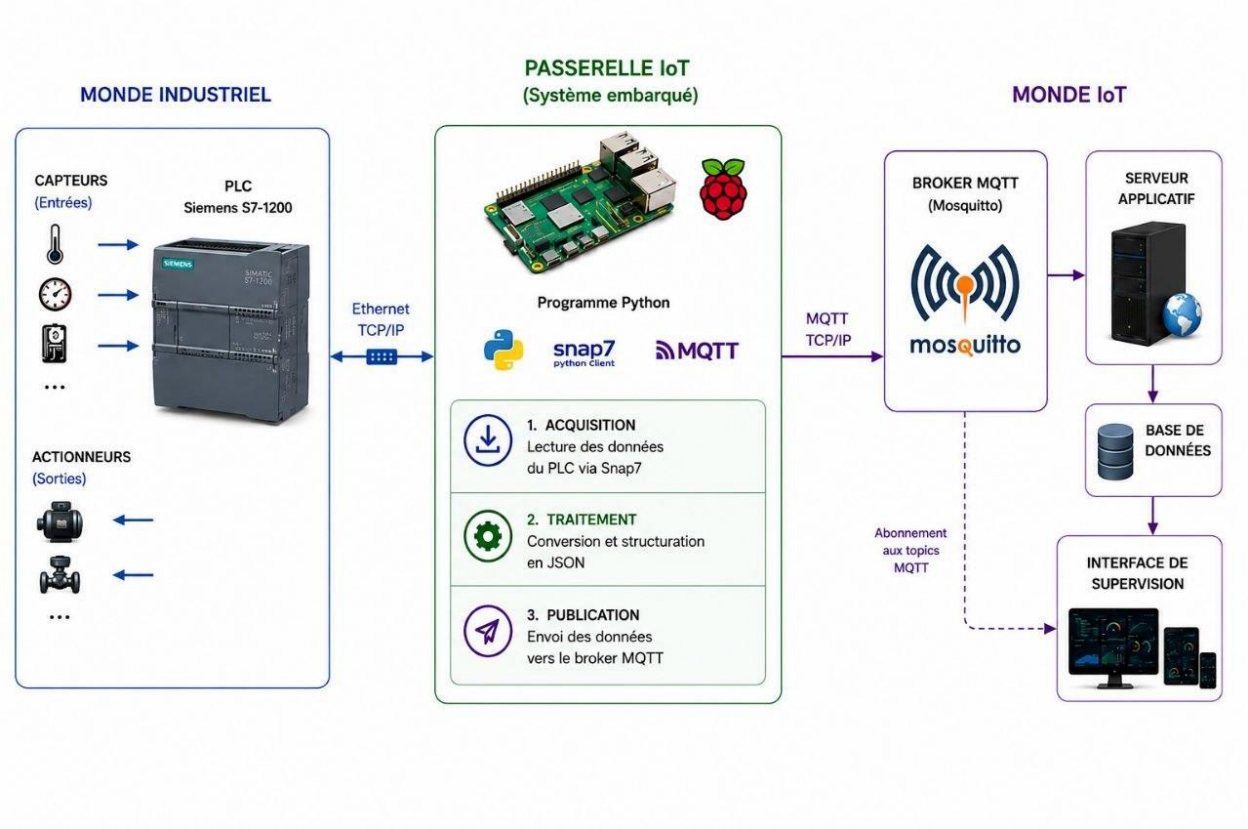


Figure 3.4: Rôle de la passerelle — pont entre le monde industriel et le monde IoT

4.4 Couche Transport — Le Protocole MQTT et le Broker

La couche transport assure le transit des données entre les passerelles et le serveur. Elle repose sur le protocole MQTT et son composant central : le **broker**.

Le broker MQTT est le point de convergence de toutes les données publiées par l'ensemble des passerelles. Il reçoit les messages publiés par chaque passerelle sur ses topics respectifs, les conserve temporairement et les redistribue à tous les abonnés concernés — en l'occurrence le serveur applicatif.

L'organisation des topics MQTT suit une structure hiérarchique claire et cohérente, permettant d'identifier sans ambiguïté l'origine et la nature de chaque donnée :

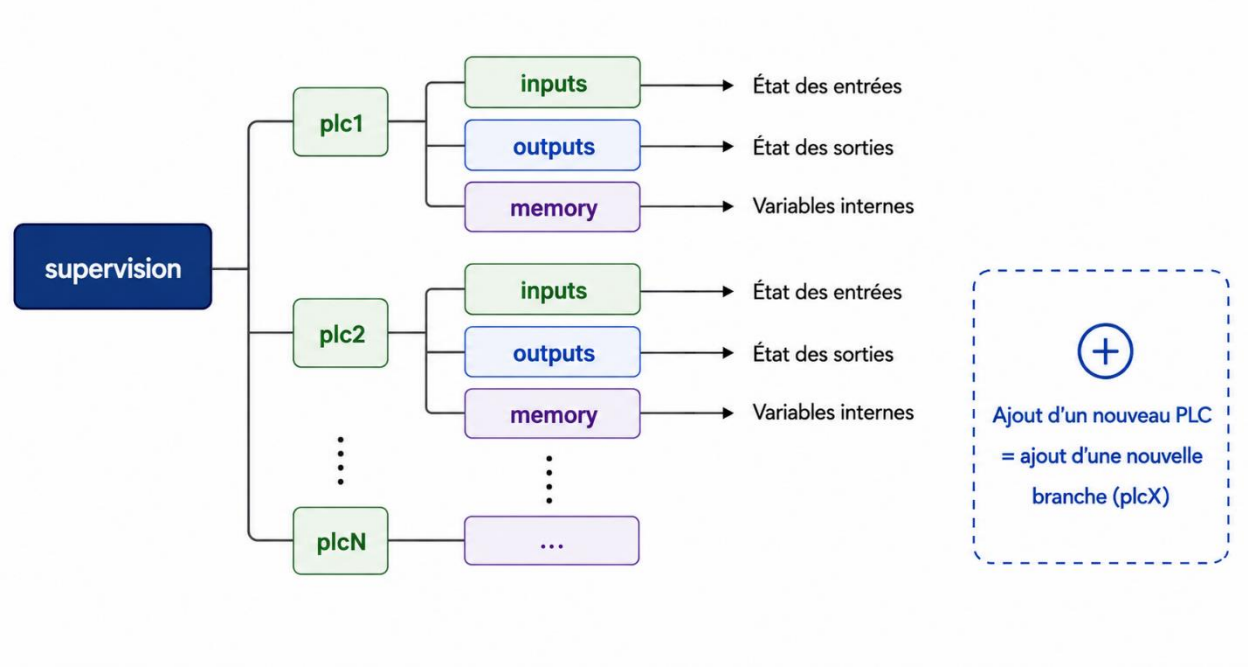


Figure 3.5 : organisation hiérarchique des topics MQTT

Cette organisation hiérarchique des topics présente un avantage majeur : l'ajout d'un nouveau PLC dans le système se résume à l'ajout d'une nouvelle branche dans l'arborescence des topics, sans modifier quoi que ce soit dans la configuration existante.

Le broker gère également les niveaux de **Qualité de Service (QoS)** des messages, garantissant que les données critiques sont bien reçues par le serveur, même en cas d'instabilité réseau momentanée.

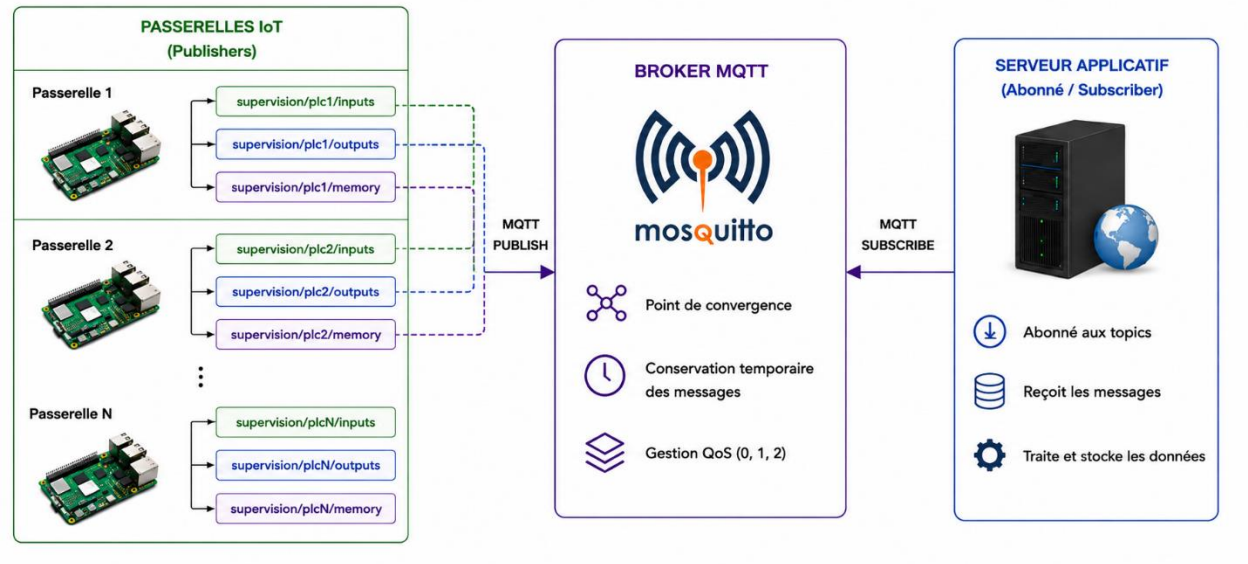


Figure 3.6 : Organisation des topics MQTT et rôle du broker dans l'architecture

4.5 Couche Serveur — Traitement et Stockage des Données

La couche serveur est le cerveau applicatif du système de supervision. Elle reçoit l'ensemble des données transmises par les passerelles via le broker MQTT, les traite, les stocke et les met à disposition de l'interface de supervision.

Le serveur remplit trois fonctions essentielles et complémentaires :

Fonction 1 — Réception et traitement des données : Le serveur est abonné à l'ensemble des topics MQTT publiés par les passerelles. À chaque réception d'un message, il le décode, vérifie son intégrité, extrait les valeurs et les prépare pour le stockage. Il peut également appliquer des règles de traitement : détection de dépassements de seuils, calcul de moyennes, génération d'alarmes.

Fonction 2 — Stockage et historisation : Toutes les données reçues sont stockées dans une base de données avec leur horodatage précis. Cette base de données constitue la mémoire du système : elle conserve l'historique complet de l'état de chaque processus industriel supervisé, permettant des analyses rétrospectives, la détection de tendances et la génération de rapports de production.

Fonction 3 — Exposition des données à l'interface : Le serveur expose les données stockées à l'interface de supervision via une API web standardisée. L'interface peut ainsi interroger le serveur pour obtenir les valeurs en temps réel, consulter l'historique ou récupérer les alarmes actives.

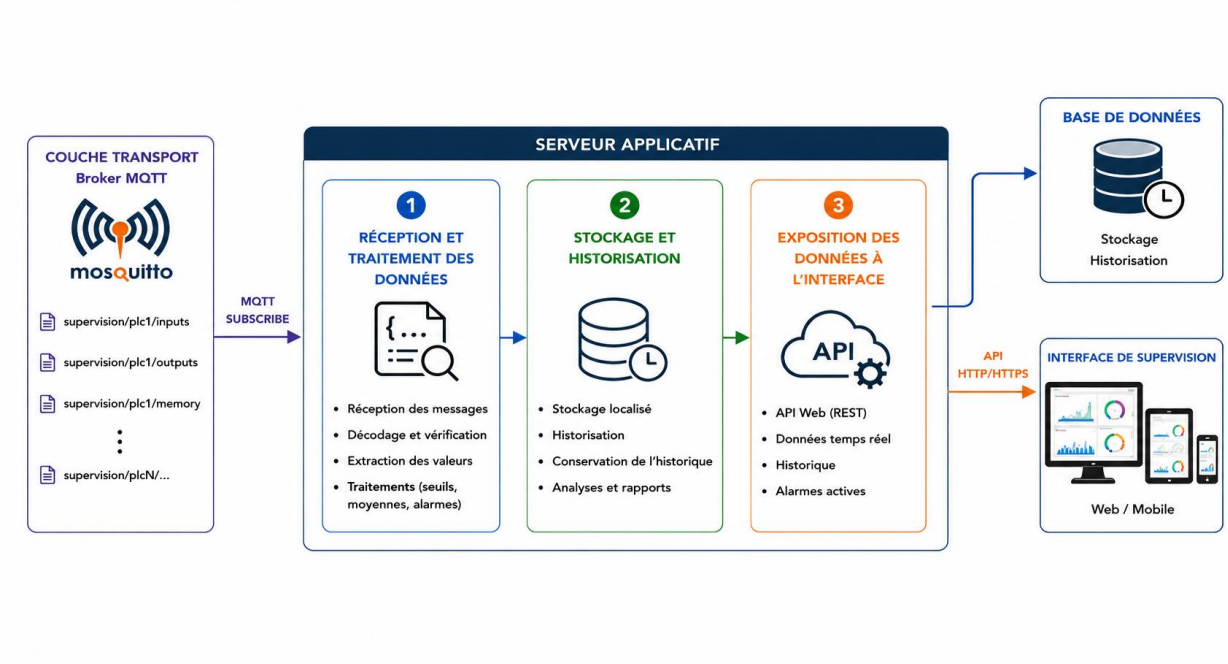


Figure 3.7 : Fonctionnement interne du serveur — réception, traitement, stockage et exposition des données

4.6 Couche Application — L'Interface de Supervision

La couche application est le point de contact entre le système et l'opérateur humain. Elle matérialise la valeur ajoutée de l'ensemble du système en transformant les données brutes stockées dans la base de données en informations visuelles claires, compréhensibles et exploitables.

L'interface de supervision offre plusieurs fonctionnalités essentielles pour l'opérateur :

Visualisation en temps réel : L'interface affiche en permanence l'état actuel de chaque processus supervisé. Les valeurs numériques sont présentées sous forme de jauges, de courbes d'évolution et de voyants d'état. L'opérateur perçoit instantanément la situation de chaque installation, qu'il s'agisse d'un seul processus ou de plusieurs dizaines d'installations simultanées.

Tableaux de bord configurables : Chaque processus industriel dispose de son propre tableau de bord, présentant les informations pertinentes selon la nature du processus. Ces tableaux de bord sont configurables pour s'adapter aux besoins spécifiques de chaque installation.

Gestion des alarmes : Lorsqu'une valeur dépasse un seuil critique prédéfini, l'interface affiche une alarme visuelle et enregistre l'événement dans le journal des alarmes. L'opérateur peut ainsi réagir rapidement aux situations anormales, depuis n'importe quel endroit.

Historique et rapports : L'opérateur peut consulter l'historique des données sur n'importe quelle période, visualiser l'évolution temporelle des paramètres et générer des rapports de production.

Accessibilité multi-appareils : L'interface est accessible via un navigateur web standard, depuis un ordinateur de bureau, une tablette ou un téléphone mobile, sans installation de logiciel dédié. Cette accessibilité universelle constitue l'un des apports fondamentaux du système par rapport aux solutions SCADA traditionnelles.

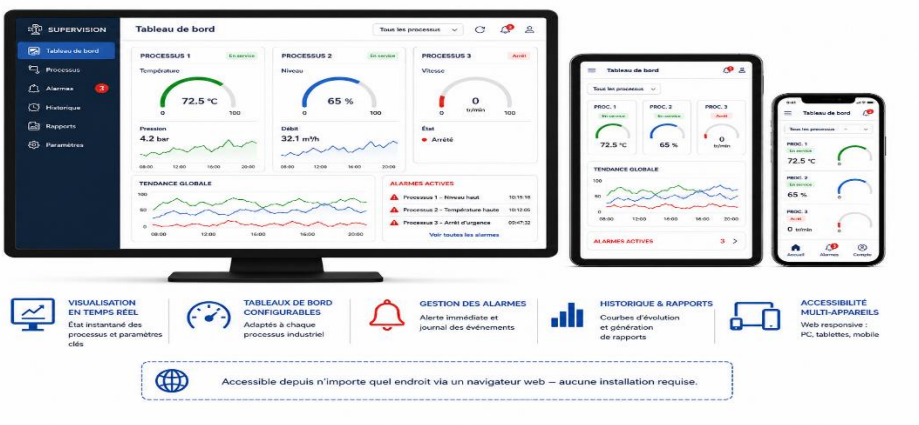


Figure 3.8 : Interface de supervision — tableau de bord multi-processus accessible sur PC, tablette et mobile

5. Les Composants Logiciels Clés

5.1 La Bibliothèque Snap7

5.1.1 Présentation Générale

Snap7 est une bibliothèque de communication open-source, multi-plateforme et multi-langage, spécialement conçue pour la communication avec les automates programmables Siemens de la famille S7. Elle constitue le composant logiciel fondamental qui rend possible l'accès aux données des automates depuis la passerelle.

Développée initialement par Davide Nardella, Snap7 implémente le protocole de communication natif Siemens S7 — connu sous le nom de **S7Comm** — permettant ainsi d'établir une communication directe avec les automates S7-300, S7-400, S7-1200 et S7-1500 via le réseau Ethernet, sans nécessiter l'installation d'aucun logiciel ni driver supplémentaire côté automate.[35]

Les caractéristiques techniques de Snap7 qui justifient son choix dans ce projet sont les suivante:

CARACTERISTIQUE	VALEUR
LICENCE	Open-source (LGPL)
LANGAGES SUPPORTES	Python, C++, Java, C#, Delphi
SYSTEMES SUPPORTES	Windows, Linux, macOS, Raspberry Pi
PROTOCOLE IMPLEMENTE	S7Comm (protocole natif Siemens)
PORT DE COMMUNICATION	TCP 102
AUTOMATES COMPATIBLES	S7-300, S7-400, S7-1200, S7-1500
MODE DE COMMUNICATION	Client/Serveur

Tableau 3.3 : Caractéristiques techniques de la bibliothèque Snap7 [36]

5.1.2 Principe de Communication avec le PLC

Snap7 fonctionne selon un modèle **client/serveur** : la passerelle joue le rôle de client et l'automate joue le rôle de serveur. C'est toujours la passerelle qui initie les échanges en envoyant des requêtes de lecture, et l'automate qui répond passivement en fournissant les données demandées.

La communication s'établit selon le processus suivant :

1. La passerelle établit une connexion TCP avec l'automate sur le **port 102**, en spécifiant l'adresse IP de l'automate ainsi que son rack et son slot,
2. Une fois la connexion établie, la passerelle envoie des requêtes de lecture spécifiant la zone mémoire à lire, l'offset de départ et la longueur des données à récupérer,
3. L'automate répond en fournissant les données demandées sous forme de tableau d'octets,

4. La passerelle décode ce tableau d'octets pour en extraire les valeurs exploitables,
5. La connexion est maintenue ouverte en permanence pour permettre des lectures répétées et continues.

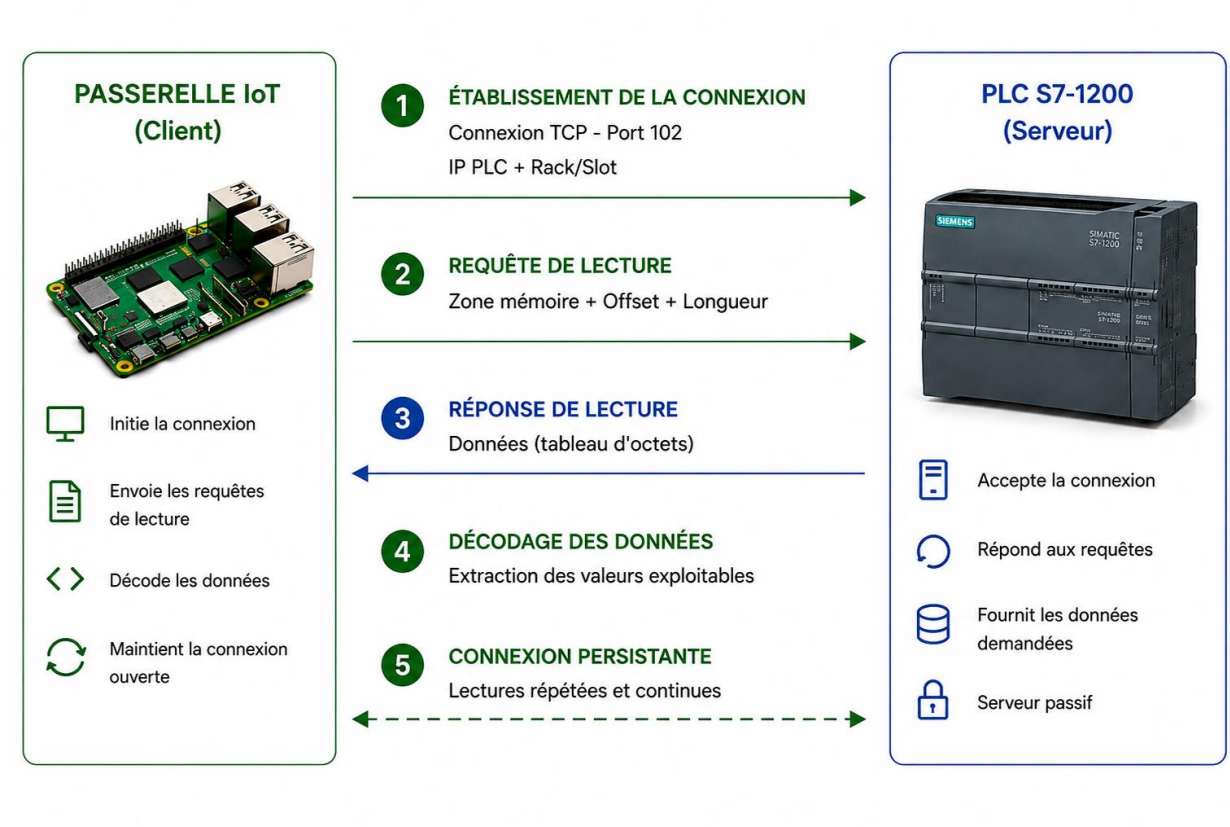


Figure 3.9 : Schéma de la communication Snap7 entre la passerelle et l'automate — Port TCP 102

5.1.3 Les Zones Mémoire Accessibles

Snap7 offre un accès complet à l'ensemble des zones mémoire de l'automate. Chaque zone est identifiée par un type, et chaque variable est adressée par son offset (position en octets depuis le début de la zone) et sa longueur.

ZONE	TYPE SNAP7	ACCES	DESCRIPTION
ENTREES PHYSIQUES	S7AreaPE	Lecture	État binaire des capteurs
SORTIES PHYSIQUES	S7AreaPA	Lecture	État binaire des actionneurs
MEMOIRES INTERNES	S7AreaMK	Lecture	Variables internes du programme
BLOCS DE DONNEES	S7AreaDB	Lecture	Données structurées utilisateur

Tableau 3.4 : Accès aux zones mémoire via Snap7

5.2 Le Protocole MQTT dans le Contexte Industriel

5.2.1 Pourquoi MQTT pour ce Projet

Le choix du protocole MQTT comme protocole de transport des données entre la passerelle et le serveur résulte d'une analyse comparative rigoureuse des protocoles disponibles, en regard des exigences spécifiques du projet.

Comme présenté au Chapitre 1, MQTT est un protocole de messagerie léger, standardisé ISO/IEC 20922, reposant sur un modèle Publish/Subscribe asynchrone. [9]

Dans le contexte de ce projet, plusieurs caractéristiques de MQTT le rendent particulièrement adapté :

La légèreté du protocole est déterminante pour une passerelle embarquée disposant de ressources limitées. Avec un en-tête minimal de seulement 2 octets, MQTT génère un overhead réseau négligeable, même lors de publications fréquentes de données.

Le modèle Publish/Subscribe correspond parfaitement à la logique de supervision unidirectionnelle du projet. La passerelle publie les données sans se soucier de qui les reçoit, et le serveur s'abonne aux topics qui l'intéressent sans se soucier de qui les publie. Ce découplage entre émetteur et récepteur simplifie considérablement l'architecture et facilite l'ajout de nouveaux composants.

La gestion native des déconnexions est essentielle dans un environnement industriel où la fiabilité du réseau ne peut être garantie à tout moment. MQTT intègre nativement des mécanismes de détection de déconnexion (Last Will and Testament) et de persistance des messages (Retain), garantissant qu'aucune donnée critique n'est perdue en cas d'interruption réseau momentanée. [37]

Le tableau suivant illustre pourquoi MQTT a été préféré à HTTP dans ce contexte

CRITERE	MQTT	HTTP
MODELE DE COMMUNICATION	Publish/Subscribe asynchrone	Requête/Réponse synchrone
OVERHEAD RESEAU	Très faible (2 octets min.)	Élevé (headers HTTP)
ADAPTE A LA TELEMETRIE TEMPS REEL	Oui	Non
GESTION DES DECONNEXIONS	Native	Non native
CONSOMMATION RESSOURCES	Très faible	Moyenne
ADAPTE AUX PASSERELLES EMBARQUEES	Oui	Partiel

Tableau 3.5 : Comparaison MQTT vs HTTP dans le contexte du projet [42]

5.2.2 Structure des Topics et des Messages

L'organisation des topics MQTT constitue un élément architectural important du système. Une structure de topics bien conçue garantit la lisibilité, la maintenabilité et l'extensibilité du système.

Dans ce projet, les topics sont organisés selon une hiérarchie à trois niveaux :

Niveau 1 : identifiant du domaine (supervision)

Niveau 2 : identifiant du PLC (plc1, plc2, ..., plcN)

Niveau 3 : type de données (inputs, outputs, memory)

Exemple de topic complet : supervision/plc1/inputs

Les messages publiés sur ces topics sont formatés en **JSON** (JavaScript Object Notation), un format léger, lisible et universellement supporté. Chaque message contient l'horodatage de la mesure et les valeurs de toutes les variables de la zone mémoire correspondante.[43]

Exemple de message JSON publié sur le topic supervision/plc1/inputs :

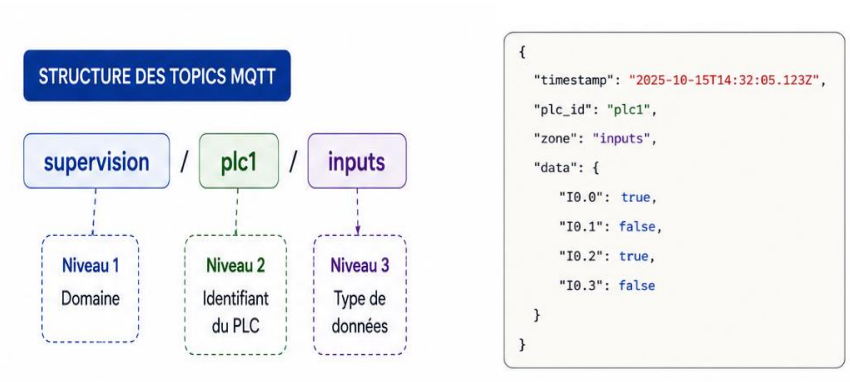


Figure 3.10 : Structure hiérarchique des topics MQTT et exemple de message JSON

5.2.3 Le Broker Mosquitto

Le broker MQTT utilisé dans ce projet est **Eclipse Mosquitto**, le broker open-source de référence dans l'écosystème MQTT. Mosquitto est développé et maintenu par la Eclipse Foundation, garantissant sa pérennité et la qualité de son code.[38]

Mosquitto remplit plusieurs rôles essentiels dans l'architecture :

Centralisation : il reçoit tous les messages publiés par l'ensemble des passerelles et les redistribue aux abonnés concernés,

Persistance : grâce au mécanisme Retain, il conserve le dernier message reçu sur chaque topic et le transmet immédiatement à tout nouvel abonné,

Fiabilité : il gère les niveaux de QoS et garantit la livraison des messages selon le niveau de service configuré,

Légèreté : Mosquitto consomme très peu de ressources système et peut fonctionner sur des machines modestes comme sur des serveurs puissants.

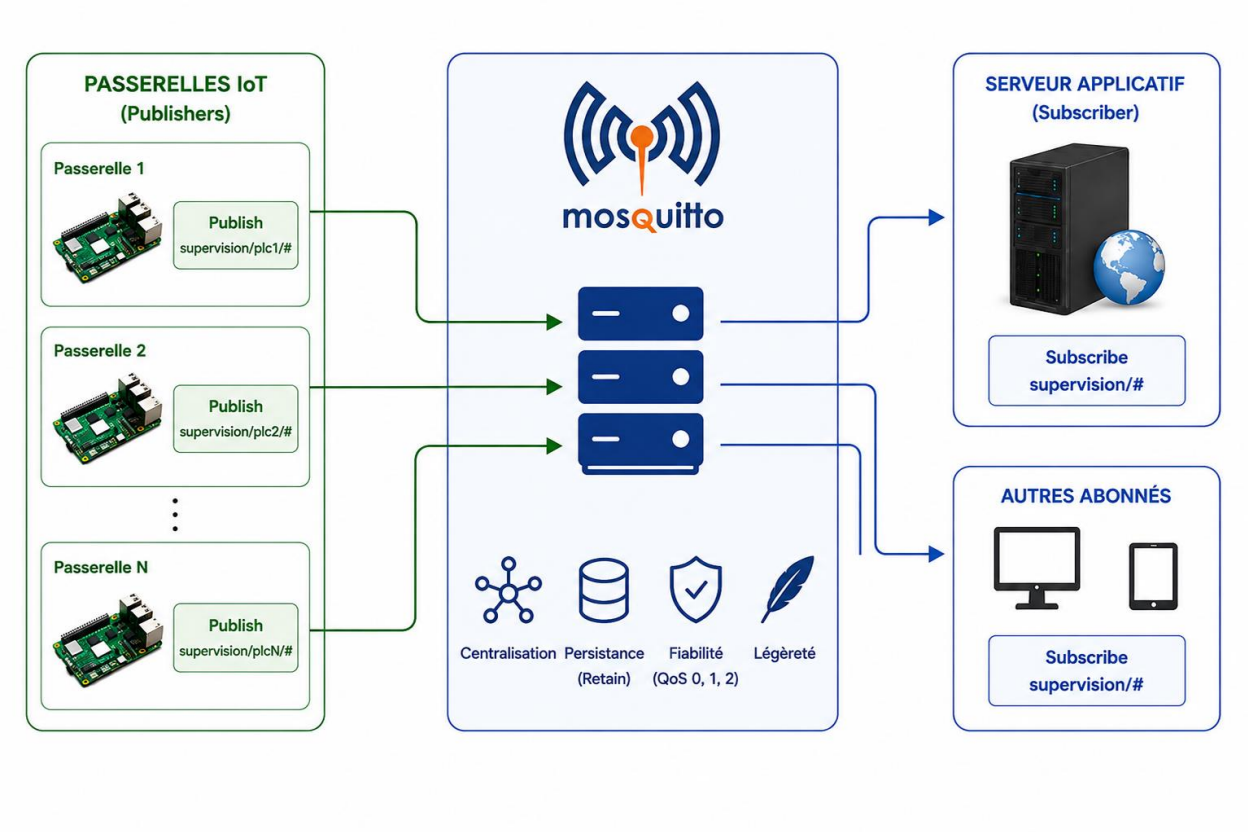


Figure 3.11 : Rôle central du broker Mosquitto dans l'architecture de supervision

6. Universalité et Adaptabilité du Système

6.1 Indépendance vis-à-vis du Processus Industriel

L'une des caractéristiques les plus importantes du système conçu dans ce projet est son **universalité fondamentale**. Contrairement aux solutions de supervision traditionnelles développées spécifiquement pour un processus industriel donné, ce système est conçu pour fonctionner de manière identique quel que soit le processus supervisé.

Cette universalité repose sur un principe architectural simple : **le système ne connaît pas la sémantique des données qu'il transporte**. Il lit des octets dans la mémoire de l'automate, les structure en messages JSON et les achemine vers le serveur, sans jamais interpréter leur signification physique. C'est l'interface de supervision qui attribue une signification aux données en les associant à des étiquettes, des unités et des représentations graphiques appropriées.

Concrètement, pour appliquer le système à un nouveau processus industriel, il suffit de configurer trois paramètres :

L'adresse IP de l'automate à superviser,

Les zones mémoire à lire et leurs offsets,

Les topics MQTT sur lesquels publier les données.

Tout le reste de l'architecture — broker, serveur, base de données, interface — reste identique et inchangé. Cette modularité est illustrée par la **Figure 3.12**.

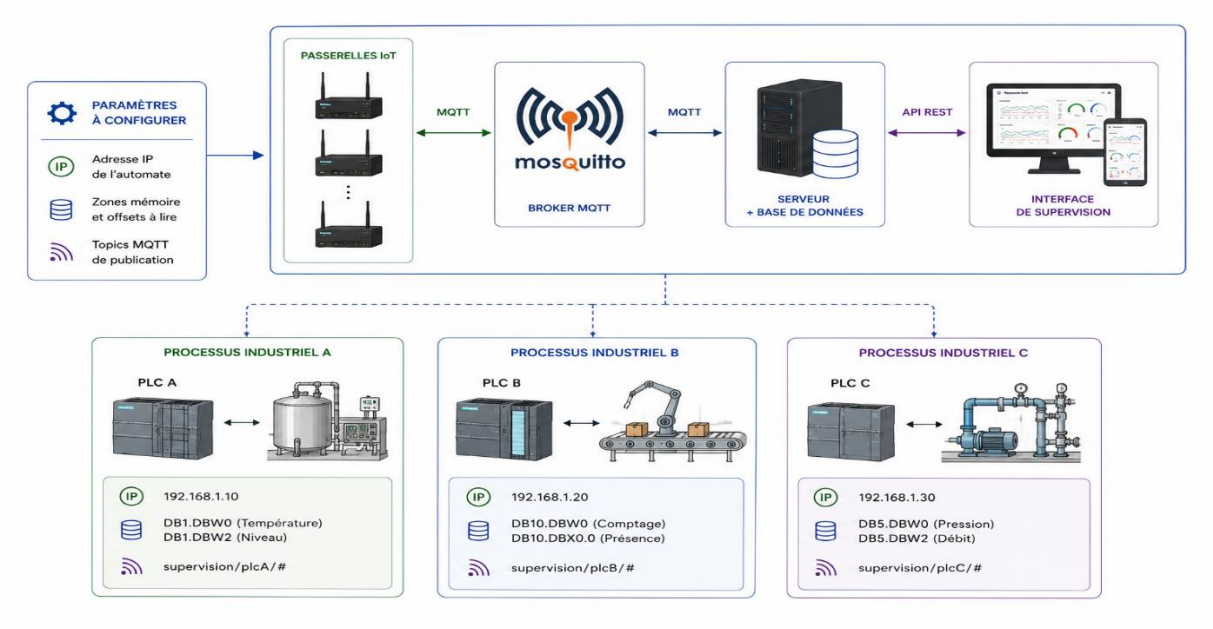


Figure 3.12 : Universalité du système — la même architecture appliquée à des processus industriels différents

6.2 Extensibilité et Scalabilité

L'architecture en couches du système lui confère une extensibilité naturelle. L'ajout d'un nouveau PLC dans le système de supervision ne nécessite aucune modification de l'infrastructure existante. Il suffit de :

1. Connecter une nouvelle passerelle au nouveau PLC via Ethernet,
2. Configurer la passerelle avec l'adresse IP du PLC et les zones mémoire à lire,
3. Définir les topics MQTT sur lesquels publier les données du nouveau PLC,
4. Ajouter un tableau de bord correspondant dans l'interface de supervision.

Le broker, le serveur et la base de données absorbent naturellement les nouvelles données sans reconfiguration. Cette scalabilité horizontale permet de faire évoluer progressivement le système, depuis la supervision d'un seul automate jusqu'à la supervision d'une installation industrielle complète comportant de nombreux automates et processus.

6.3 Perspectives d'Évolution

Le système conçu dans ce projet constitue une base solide et évolutive sur laquelle de nombreuses fonctionnalités peuvent être ajoutées dans le cadre de travaux futurs :

Vers le contrôle bidirectionnel : L'architecture actuelle est unidirectionnelle (supervision uniquement). Une évolution naturelle consisterait à ajouter un flux descendant permettant à l'opérateur d'envoyer des commandes vers l'automate depuis l'interface de supervision, transformant ainsi le système de supervision en système de télé-opération.

Vers la maintenance prédictive : L'historique des données accumulé dans la base de données constitue une ressource précieuse pour l'application d'algorithmes d'intelligence artificielle. Des modèles de machine learning pourraient analyser les tendances des données pour anticiper les défaillances avant qu'elles ne surviennent, réduisant ainsi les arrêts non planifiés.

Vers le Cloud : Le serveur applicatif peut être migré vers une infrastructure Cloud, offrant une disponibilité maximale, une scalabilité illimitée et une accessibilité mondiale, sans investissement en infrastructure matérielle locale.

Vers les alertes proactives : Le système peut être enrichi d'un module de notification automatique envoyant des alertes par email ou SMS lorsque des anomalies sont détectées, permettant une réactivité maximale des équipes de maintenance.

7. Conclusion

Ce chapitre a présenté en détail la conception du système IoT de supervision industrielle développé dans ce projet. Partant d'une problématique industrielle concrète et universelle — l'isolement informationnel des automates programmables et l'impossibilité de superviser les processus à distance — il a proposé une réponse architecturale cohérente, modulaire et économique.

L'architecture en cinq couches fonctionnelles — terrain, passerelle, transport, serveur et application — organise clairement les responsabilités de chaque composant et garantit la modularité et l'extensibilité du système. La bibliothèque Snap7 assure la communication non-intrusive avec les automates programmables, le protocole MQTT garantit un transport léger et fiable des données, et l'interface de supervision offre une visualisation accessible depuis n'importe quel appareil connecté.

La caractéristique fondamentale de ce système est son **universalité** : conçu pour être indépendant de la nature du processus industriel supervisé, il constitue une solution générique applicable à toute installation pilotée par un automate programmable Siemens S7-1200, qu'il s'agisse d'un convoyeur, d'un système de remplissage, d'un processus thermique ou de toute autre application industrielle.

Le chapitre suivant mettra en œuvre cette architecture sur deux processus industriels concrets, permettant de valider expérimentalement les principes exposés dans ce chapitre et de démontrer la faisabilité et les performances de la solution proposée.

Chapitre 4 : Réalisation et Expérimentation

1. Introduction

Les chapitres précédents ont établi les fondements théoriques nécessaires à la compréhension des systèmes IoT industriels et des automates programmables, puis ont présenté l'architecture générale et universelle du système de supervision conçu dans ce projet. Le présent chapitre constitue la concrétisation pratique de cette architecture : il décrit en détail la réalisation matérielle et logicielle du système, et présente les résultats expérimentaux obtenus lors de sa mise en œuvre sur deux processus industriels distincts.

Pour valider l'architecture proposée et démontrer son universalité, deux processus industriels ont été choisis, délibérément différents dans leur nature et leur complexité. Le premier est un processus virtuel de commande de pompes, simulé sur un banc didactique, permettant de tester la logique de supervision dans un environnement contrôlé et reproductible. Le second est un processus réel de régulation de niveau de liquide, mis en œuvre sur une maquette industrielle physique, permettant de valider le système dans des conditions proches de la réalité industrielle.

Ce chapitre présente successivement l'environnement de travail et le matériel utilisé, puis détaille pour chaque processus le programme de commande, le code de la passerelle, et les résultats obtenus. Il décrit ensuite le serveur applicatif et l'interface de supervision, avant de conclure par une analyse critique des résultats et des perspectives d'amélioration.

2. Présentation de l'Environnement de Travail

2.1 Le Banc Didactique API-24/EV

2.1.1 Description Générale

Le banc didactique **API-24/EV**, conçu et fabriqué par la société italienne **Elettronica Veneta**, est un équipement pédagogique complet destiné à l'étude et à la programmation des automates programmables industriels. Il constitue la plateforme matérielle utilisée pour le Processus 1 de ce projet. [48]

Ce banc intègre dans un même ensemble compact tous les éléments nécessaires à la mise en œuvre d'un système automatisé complet : l'automate programmable, les entrées et sorties numériques simulées, les entrées et sorties analogiques avec leurs simulateurs, et les interfaces de connexion permettant de câbler les différents éléments entre eux.



Figure 4.1 : Banc didactique API-24/EV avec automate Siemens S7-1200 intégré

2.1.2 L'Automate Siemens S7-1200

L'automate programmable intégré dans le banc API-24/EV utilisé dans ce projet est un **Siemens S7-1200**. Cet automate appartient à la famille des API compacts de Siemens, conçus pour les applications d'automatisme de petite et moyenne complexité. Il dispose nativement d'une interface **Ethernet RJ45** intégrée, permettant la communication avec des systèmes externes — dont la passerelle Raspberry Pi — via le protocole S7.

L'adresse IP de l'automate est configurée statiquement dans TIA Portal, permettant à la passerelle de s'y connecter de manière fiable et reproductible à chaque démarrage du système.

2.1.3 Les Entrées/Sorties Numériques et Analogiques

Le banc API-24/EV dispose des ressources d'entrées/sorties suivantes, directement accessibles via des bornes de connexion :

Type	Adresse PLC	Description
Entrées numériques	I0.0 → I0.13	Simulateurs à boutons-poussoirs et interrupteurs
Sorties transistors	Q0.0 → Q0.9	Sorties 24V DC — 0.5A
Sorties relais	Q0.0 → Q0.1	Sorties relais 250V — 2A
Entrée analogique 0	IW0.0 (AI0)	Potentiomètre 0-10V
Entrée analogique 1	IW0.1 (AI1)	Potentiomètre 0-10V
Sortie analogique 0	QW0.100 (AQ0)	Sortie 0-10V
Sortie analogique 1	QW0.101 (AQ1)	Sortie 0-10V

Tableau 4.1 : Ressources d'entrées/sorties du banc API-24/EV

Pour la simulation du niveau analogique dans le Processus 1, les entrées analogiques AI0 et AI1 sont câblées en boucle sur elles-mêmes (court-circuit entre la borne de sortie et la borne d'entrée du potentiomètre), permettant de faire varier manuellement la valeur analogique lue par l'automate en tournant le potentiomètre correspondant.

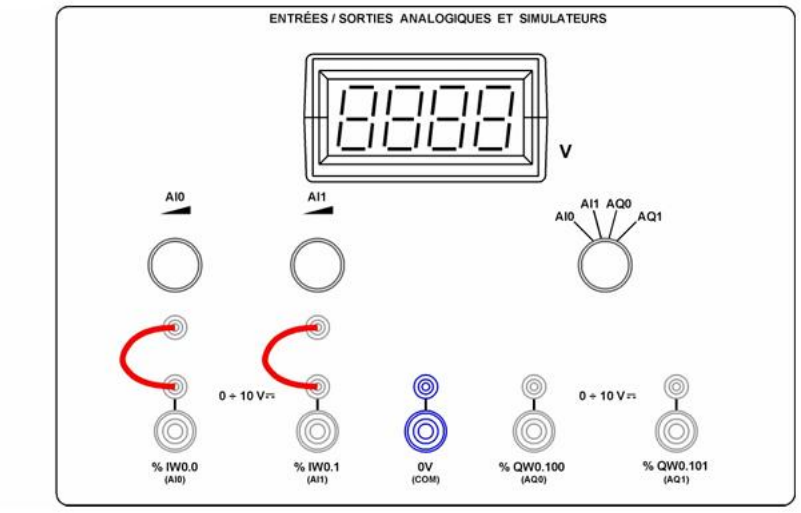


Figure 4.2 : Schéma de câblage des entrées analogiques pour la simulation du niveau — Processus 1

2.2 La Maquette FLTP/EV

2.2.1 Description Générale

La maquette **FLTP/EV** (Fluid Level Training Process), également fabriquée par **Elettronica Veneta**, est un système d'entraînement multiprocessus destiné à l'étude des systèmes de régulation industriels. Elle constitue la plateforme physique utilisée pour le Processus 2 de ce projet.[49]

Cette maquette permet d'étudier quatre types de régulation industrielle : le contrôle de niveau, le contrôle de débit, le contrôle de température et le contrôle de pression. Dans le cadre de ce projet, seul le **contrôle de niveau** est mis en œuvre.



Figure 4.3 : Maquette FLTP/EV — vue générale avec ses composants

2.2.2 Les Composants de la Maquette

La maquette FLTP/EV est constituée des éléments suivants, tous visibles et identifiables sur les photos du laboratoire :

Composant	*Rôle dans le processus*
Réservoir en plexiglas transparent	Contient le liquide, gradué de 0 à 500mm
Pompe de recirculation P1	Alimente le réservoir en liquide à débit constant
Capteur de niveau piézométrique	Mesure la hauteur du liquide (signal 0-10V)
Électrovanne ON/OFF (Solenoid Valve)	Commande la vidange du réservoir
Pressostat de sécurité	Protection contre les surpressions
Vannes manuelles	Réglage manuel du débit
Unité de conditionnement du signal	Adapte le signal du capteur pour le PLC

Tableau 4.2 : Composants de la maquette FLTP/EV utilisés dans le Processus 2

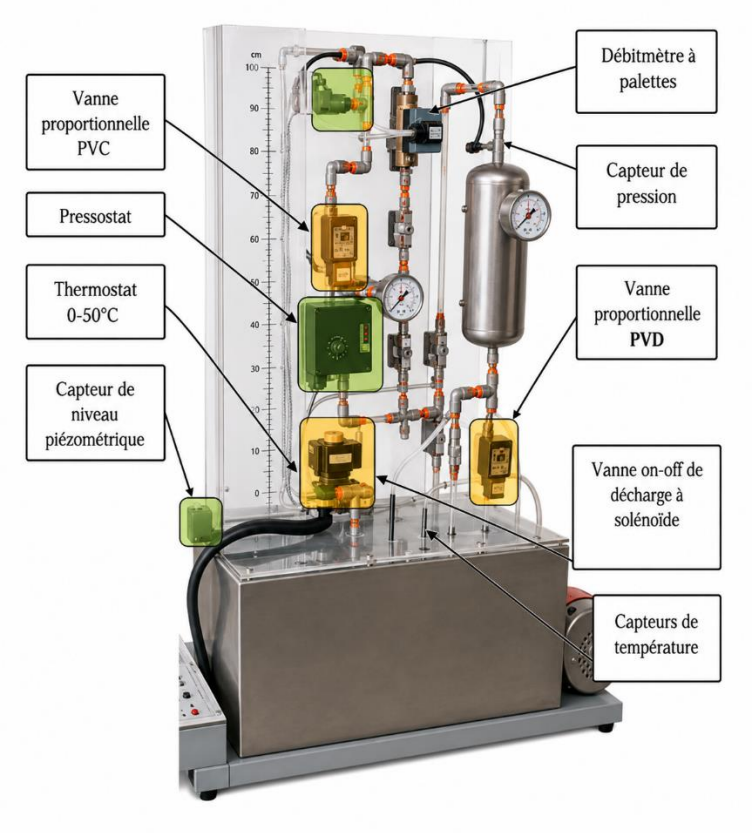


Figure 4.4 : Gros plan de la maquette FLTP/EV — réservoir, pompe P1, électrovanne et pressostat

2.2.3 Le Capteur Piézométrique

Le capteur de niveau utilisé dans la maquette est un **capteur piézométrique**. Son principe de fonctionnement repose sur la mesure de la pression hydrostatique exercée par la colonne de liquide au-dessus du point de mesure. Cette pression est directement proportionnelle à la hauteur du liquide, permettant ainsi de déduire le niveau du réservoir.

Le capteur produit un signal de sortie **0-10V**, correspondant à une plage de mesure de **0 à 500mm** de hauteur de liquide. Ce signal est transmis à l'unité de conditionnement, qui l'adapte et le transmet à l'entrée analogique de l'automate. L'automate reçoit ce signal sous forme de valeur numérique entière comprise entre **0 et 27648**, conformément à la normalisation standard des entrées analogiques Siemens S7.

2.3 La Passerelle IoT — Raspberry Pi 5

2.3.1 Présentation Générale

La passerelle IoT utilisée dans ce projet est un **Raspberry Pi 5**, le micro-ordinateur monocarte le plus récent et le plus performant de la Fondation Raspberry Pi. Contrairement aux microcontrôleurs simples comme l'Arduino, le Raspberry Pi est un véritable ordinateur fonctionnant sous un système d'exploitation Linux complet, capable d'exécuter des applications Python complexes, de gérer des connexions réseau multiples et de communiquer simultanément avec plusieurs services. [40]

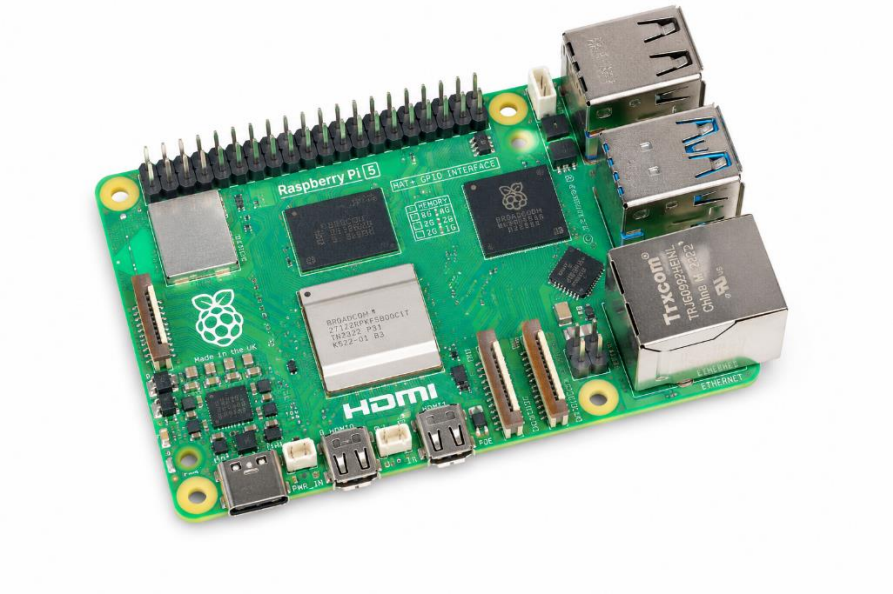


Figure 4.5 : Raspberry Pi 5 utilisé comme passerelle IoT dans le projet

2.3.2 Pourquoi le Raspberry Pi 5 et Non l'ESP32

Le choix du Raspberry Pi 5 comme passerelle IoT résulte d'une analyse comparative rigoureuse avec l'ESP32, le microcontrôleur IoT le plus couramment utilisé dans les projets similaires. Le tableau suivant présente les critères de comparaison qui ont guidé ce choix :

CRITERE	RASPBERRY PI 5	ESP32
TYPE	Micro-ordinateur	Microcontrôleur
PROCESSEUR	ARM Cortex-A76 — 2.4 GHz quad-core	Xtensa LX7 — 240 MHz dual-core
RAM	4 GB LPDDR4X	512 KB SRAM
SYSTEME D'EXPLOITATION	Linux (Raspberry Pi OS)	Pas de système d'exploitation complet

LANGAGE PYTHON	Python complet avec toutes les bibliothèques	MicroPython (bibliothèques limitées)
BIBLIOTHEQUE SNAP7	Compatible (python-snap7 officiel)	Non compatible
CONNECTIVITE ETHERNET	RJ45 intégré (1 Gbps)	Non (uniquement Wi-Fi)
FACILITE DE DEVELOPPEMENT	Très élevée	Moyenne
COUT	Plus élevé	Faible

Tableau 4.3 : Comparaison Raspberry Pi 5 vs ESP32

Le critère déterminant est la **compatibilité avec la bibliothèque Snap7**. Cette bibliothèque, indispensable pour la communication avec les automates Siemens S7-1200, requiert un système d'exploitation complet et une implémentation Python standard — conditions que l'ESP32 avec MicroPython ne peut pas satisfaire. De plus, la connexion Ethernet filaire native du Raspberry Pi 5 garantit une communication fiable et stable avec le PLC, sans les aléas d'une connexion Wi-Fi.

2.3.3 Configuration et Environnement Logiciel

Chaque Raspberry Pi 5 est configuré avec les éléments logiciels suivants :

ÉLEMENT	VERSION/DESCRIPTION
SYSTEME D'EXPLOITATION	Raspberry Pi OS (64-bit, basé sur Debian)
PYTHON	Version 3.11
BIBLIOTHEQUE PYTHON-SNAP7	Version 1.3
BIBLIOTHEQUE PAHO-MQTT	Version 1.6.1
BROKER MQTT MOSQUITTO	Version 2.0 (installé sur Pi 2)
ÉDITEUR DE CODE	Visual Studio Code (SSH distant)

Tableau 4.4 : Configuration logicielle du Raspberry Pi 5

Le projet utilise **deux Raspberry Pi 5** distincts : le premier (Pi 1) est dédié à la supervision du Processus 1, et le second (Pi 2) est dédié à la supervision du Processus 2. Les deux passerelles publient leurs données vers un **laptop externe** dont l'adresse IP est **10.186.148.13**, sur lequel sont installés et exécutés simultanément le **broker MQTT Mosquitto** [38] et le **serveur Flask**. Le serveur se connecte au broker en local via **localhost**, reçoit les données des deux processus et les redistribue vers l'interface de supervision via WebSocket. L'interface est ainsi accessible depuis n'importe quel navigateur web connecté au même réseau, sur le port **5000**.

3. Processus 1 — Commande de Pompes (Virtuel)

3.1 Description du Processus

3.1.1 Principe de Fonctionnement

Le premier processus mis en œuvre est un système virtuel de commande de pompes pour le maintien du niveau d'un réservoir. Ce processus, entièrement simulé sur le banc API-24/EV, reproduit le comportement d'un système industriel réel de régulation de niveau par pompage.

Le principe de fonctionnement est le suivant : un réservoir est équipé de trois détecteurs de niveau N1, N2 et N3, disposés à des hauteurs croissantes. Trois pompes P1, P2 et P3 alimentent ce réservoir. La logique de commande est simple et efficace : le nombre de pompes mises en service est égal au nombre de détecteurs de niveau dont le seuil n'est pas atteint. Ainsi, lorsque le niveau est très bas (les trois détecteurs non atteints), les trois pompes fonctionnent simultanément pour remplir rapidement le réservoir. Au fur et à mesure que le niveau monte et atteint les seuils, les pompes s'arrêtent progressivement.

Dans ce projet, les détecteurs de niveau sont simulés par des seuils définis sur la valeur analogique du potentiomètre du banc, et les pompes sont simulées par les sorties numériques de l'automate.

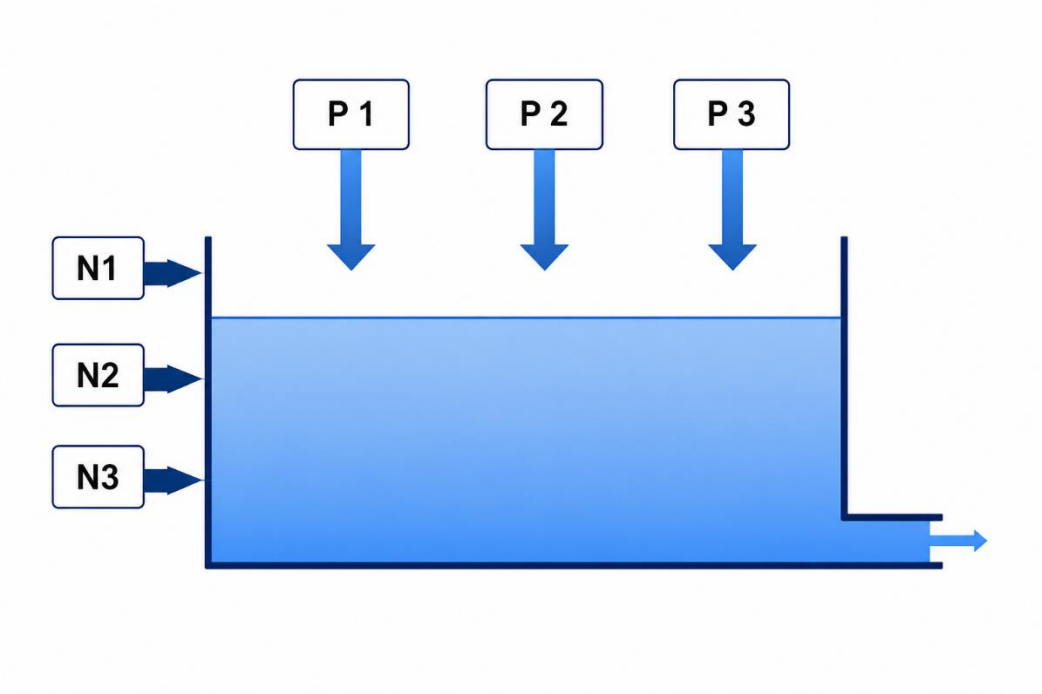


Figure 4.6 : schema represente principe de fonctionnement du Processus1

3.1.2 Les Variables du Système

Variable	Adresse PLC	Type	Description
Start	I0.0	Entrée TOR	Bouton de démarrage
Stop	I0.1	Entrée TOR	Bouton d'arrêt
Niveau (analogique)	IW66	Entrée analogique	Valeur brute 0-27648
LEV	MD2	Mémoire réelle	Niveau mis à l'échelle (0-100mm)
Sys_enable	M0.0	Bit mémoire	Activation du système
Niveau 1	M0.1	Bit mémoire	Seuil N1 atteint (≥ 85mm)
Niveau 2	M0.2	Bit mémoire	Seuil N2 atteint (≥ 50mm)
Niveau 3	M0.3	Bit mémoire	Seuil N3 atteint (≥ 15mm)
Pompe 1	Q0.0	Sortie TOR	Commande pompe P1
Pompe 2	Q0.1	Sortie TOR	Commande pompe P2
Pompe 3	Q0.2	Sortie TOR	Commande pompe P3

Tableau 4.5 : Variables du Processus 1

3.2 Programme Ladder — Processus 1 (TIA Portal)

Le programme de commande du Processus 1 est développé en langage **Ladder Diagram (LD)** dans l'environnement **TIA Portal** de Siemens. Il est structuré en trois réseaux distincts, chacun remplissant une fonction précise et indépendante.

3.2.1 Network 1 — Démarrage, Arrêt et Commande des Pompes

Le premier réseau gère deux fonctions complémentaires : la gestion du démarrage et de l'arrêt du système, et la logique de commande des trois pompes en fonction des niveaux détectés.

La gestion du démarrage/arrêt utilise un bloc fonctionnel **SR (Set/Reset)** : le contact normalement ouvert du bouton Start (I0.0) active le bit mémoire Sys_enable (M0.0) via l'entrée S du bloc SR, tandis que le contact normalement ouvert du bouton Stop (I0.1) le réinitialise via l'entrée R1. Ce mécanisme garantit que le système reste actif après relâchement du bouton Start, et s'arrête immédiatement à l'appui sur Stop.

La logique de commande des pompes repose sur des conditions en série :

Pompe 1 (Q0.0) s'active si Sys_enable est actif ET si le Niveau 1 n'est pas atteint (/M0.1)

Pompe 2 (Q0.1) s'active si Sys_enable est actif ET si le Niveau 1 ET le Niveau 2 ne sont pas atteints

Pompe 3 (Q0.2) s'active si Sys_enable est actif ET si les trois niveaux ne sont pas atteints

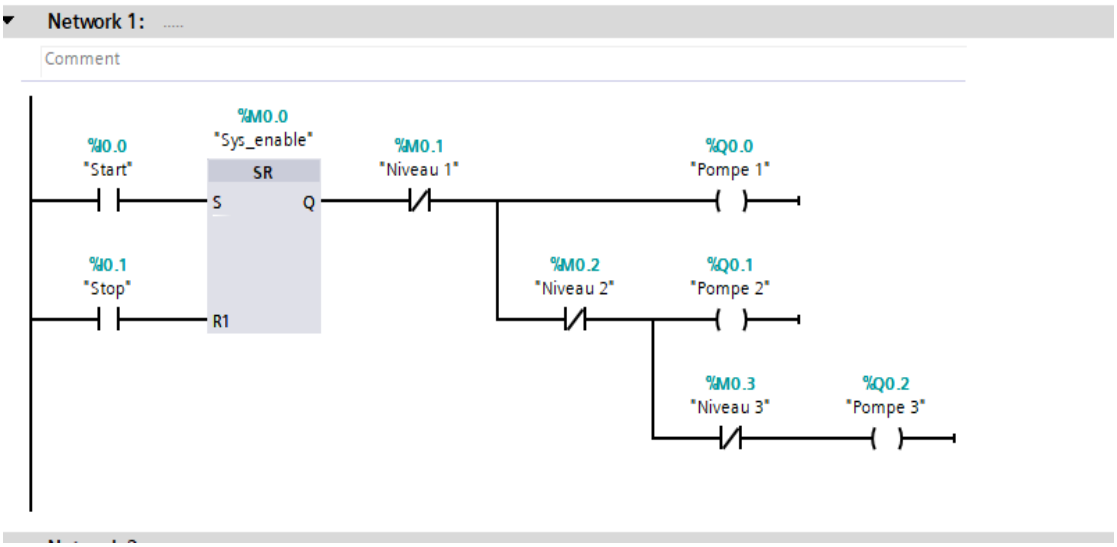


Figure 4.7 : Network 1 du Ladder Processus 1 — Démarrage/Arrêt et commande des pompes

3.2.2 Network 2 — Acquisition et Mise à l'Échelle du Niveau Analogique

Le deuxième réseau assure la conversion de la valeur analogique brute lue sur l'entrée IW66 en une valeur de niveau exploitable en millimètres, grâce à deux blocs fonctionnels en cascade :

Le bloc **NORM_X (Int to Real)** normalise la valeur entière brute de l'entrée analogique IW66 (comprise entre 0 et 27648) en une valeur réelle normalisée comprise entre 0.0 et 1.0, stockée dans une variable intermédiaire #LEVEL.

Le bloc **SCALE_X (Real to Real)** met ensuite à l'échelle cette valeur normalisée sur la plage physique souhaitée : de 0.0 à 100.0 mm, produisant la valeur finale stockée dans MD2 "LEV", exprimée en millimètres.

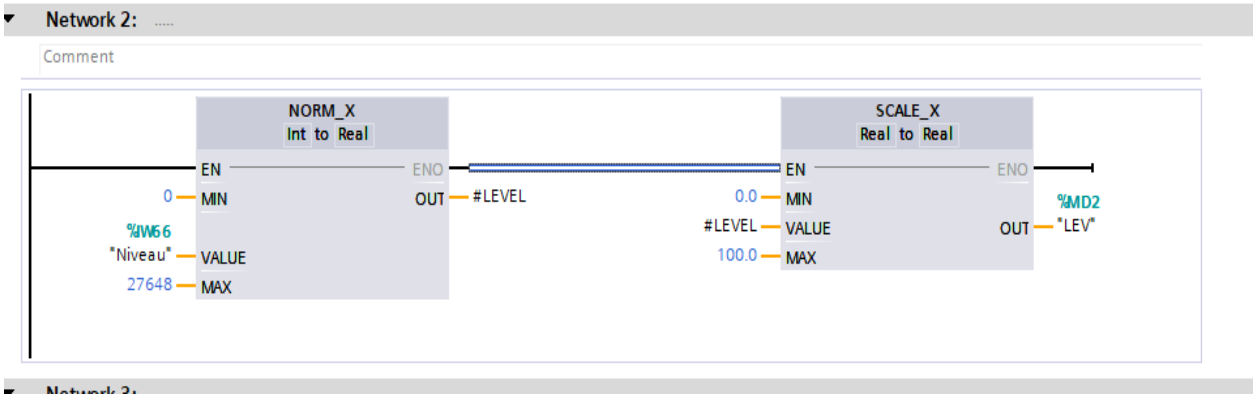


Figure 4.8 : Network 2 du Ladder Processus 1 — Normalisation et mise à l'échelle du niveau analogique

3.2.3 Network 3 — Logique de Détection des Niveaux

Le troisième réseau compare la valeur de niveau mise à l'échelle (MD2 "LEV") à trois seuils prédéfinis, et active les bits mémoire de niveau correspondants :

Si **LEV ≥ 85.0 mm** → M0.1 "Niveau 1" = 1 (réservoir bien rempli)

Si **LEV ≥ 50.0 mm** → M0.2 "Niveau 2" = 1 (niveau intermédiaire)

Si **LEV ≥ 15.0 mm** → M0.3 "Niveau 3" = 1 (niveau bas)

Ces bits mémoire sont ensuite utilisés par le Network 1 pour déterminer le nombre de pompes à mettre en service. Lorsque le niveau est inférieur à 15mm, les trois bits sont à 0 et les trois pompes fonctionnent. Lorsque le niveau dépasse 85mm, les trois bits sont à 1 et les trois pompes s'arrêtent.

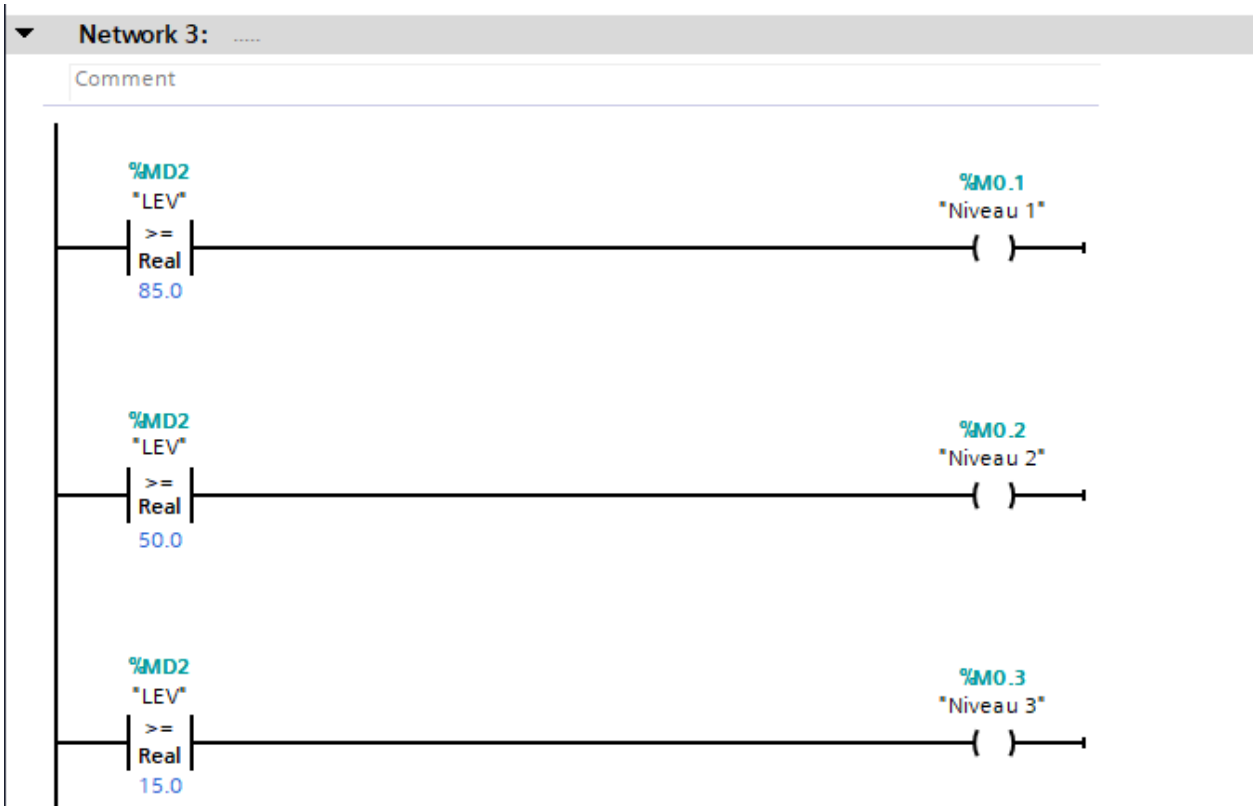


Figure 4.9: Network 3 du Ladder Processus 1 — Logique de détection des seuils de niveau

3.3 Code Python — Raspberry Pi 5 (Processus 1)

3.3.1 Structure du Code

Le code Python exécuté sur le premier Raspberry Pi 5 assure trois fonctions séquentielles et continues : la connexion et la lecture des données du PLC via Snap7, le traitement et la structuration des données en messages JSON, et la publication de ces messages vers le broker MQTT. Le code s'exécute en boucle infinie avec un intervalle de **10 secondes** entre chaque cycle de lecture.

3.3.2 Connexion au PLC via Snap7

La connexion au PLC est établie en deux étapes. La première étape initialise le client Snap7 et établit la connexion TCP avec l'automate en spécifiant son adresse IP (saisie manuellement au démarrage), son rack (0) et son slot (1), conformément à la configuration matérielle du S7-1200. La deuxième étape initialise le client MQTT et établit la connexion avec le broker Mosquitto, dont l'adresse IP est fixée à 10.186.148.13 sur le port standard 1883.[35]

```python
plc1 copy.py > ...
import snap7
from snap7.type import Area
from snap7.util import get_bool, get_int
import paho.mqtt.client as mqtt
import json
import time

# MQTT Setup - sends to Pi 2
mqtt_client = mqtt.Client()
mqtt_client.connect("10.186.148.13", 1883)

plc1 = snap7.client.Client()
ip_address1 = input("Enter PLC 1 IP address: ")
plc1.connect(ip_address1, 0, 1)

while True:
    print("-----------------------------------------")
    print("\n--------  New status Process(1):  -------------")

    data_inputs = plc1.read_area(Area.PE, 0, 0, 1)
    inputs = []
    for i in range(6):
        inputs.append(get_bool(data_inputs, 0, i))
    print("Entrées I0.0 → I0.5 :", inputs)

    data_outputs = plc1.read_area(Area.PA, 0, 0, 1)
    outputs = []
    for i in range(6):
        outputs.append(get_bool(data_outputs, 0, i))
    print("Sorties Q0.0 → Q0.5 :", outputs)

    data_analog = plc1.read_area(Area.PE, 0, 66, 2)
```

Figure 4.10 : Code Python du Raspberry Pi 5 — Processus 1 (partie 1 : connexion et lecture)

3.3.3 Lecture des Données et Publication MQTT

À chaque cycle de 10 secondes, le code lit trois zones mémoire distinctes de l'automate :

Lecture des entrées TOR : La zone PE (Process Inputs), à l'offset 0, sur 1 octet, fournit l'état des 6 premières entrées numériques (I0.0 à I0.5). Chaque bit est extrait individuellement grâce à la fonction get_bool() de Snap7.

Lecture des sorties TOR : La zone PA (Process Outputs), à l'offset 0, sur 1 octet, fournit l'état des 6 premières sorties numériques (Q0.0 à Q0.5), dont les trois bits correspondant aux pompes P1, P2 et P3.

Lecture de l'entrée analogique : La zone PE, à l'offset 66, sur 2 octets, fournit la valeur brute de l'entrée analogique IW66 sous forme d'entier signé 16 bits. Cette valeur est normalisée par le code Python selon la formule : niveau_mm = (valeur_brute / 27648) × 100, produisant le niveau en millimètres sur une plage de 0 à 100mm.

Les données collectées sont ensuite structurées en un dictionnaire Python et sérialisées en format JSON avant publication sur le topic MQTT plc/1 :

```
33      analog_value = get_int(data_analog, 0)
34      print("IW64 =", analog_value)
35
36      normalized = analog_value / 27648.0
37      LEVEL_MIN = 0.0
38      LEVEL_MAX = 100.0
39      level_mm = normalized * (LEVEL_MAX - LEVEL_MIN) + LEVEL_MIN
40      print("Level (mm) =", round(level_mm, 2))
41
42      pump1 = get_bool(data_outputs, 0, 0)
43      pump2 = get_bool(data_outputs, 0, 1)
44      pump3 = get_bool(data_outputs, 0, 2)
45      print("\n--- Pump Status ---")
46      print(f"Pump 1: {'ON' if pump1 else 'OFF'}")
47      print(f"Pump 2: {'ON' if pump2 else 'OFF'}")
48      print(f"Pump 3: {'ON' if pump3 else 'OFF'}")
49
50      plc1_data = {
51          "plc": 1,
52          "inputs": inputs,
53          "outputs": outputs,
54          "level_mm": round(level_mm, 2),
55          "pump1": pump1,
56          "pump2": pump2,
57          "pump3": pump3
58      }
59      mqtt_client.publish("plc/1", json.dumps(plc1_data))
60
61      time.sleep(10)
62
63  plc1.disconnect()
```

Figure 4.11: Code Python du Raspberry Pi 5 — Processus 1 (partie 2 : publication MQTT)

3.4 Tests et Résultats — Raspberry Pi 5 (Processus 1)

3.4.1 Résultats des Lectures

Les tests du Raspberry Pi 5 sur le Processus 1 ont été réalisés en faisant varier manuellement le niveau du potentiomètre sur le banc API-24/EV, simulant ainsi l'évolution du niveau dans le réservoir. Les résultats affichés dans le terminal du Raspberry Pi confirment le bon fonctionnement de l'ensemble du système de lecture et de publication.

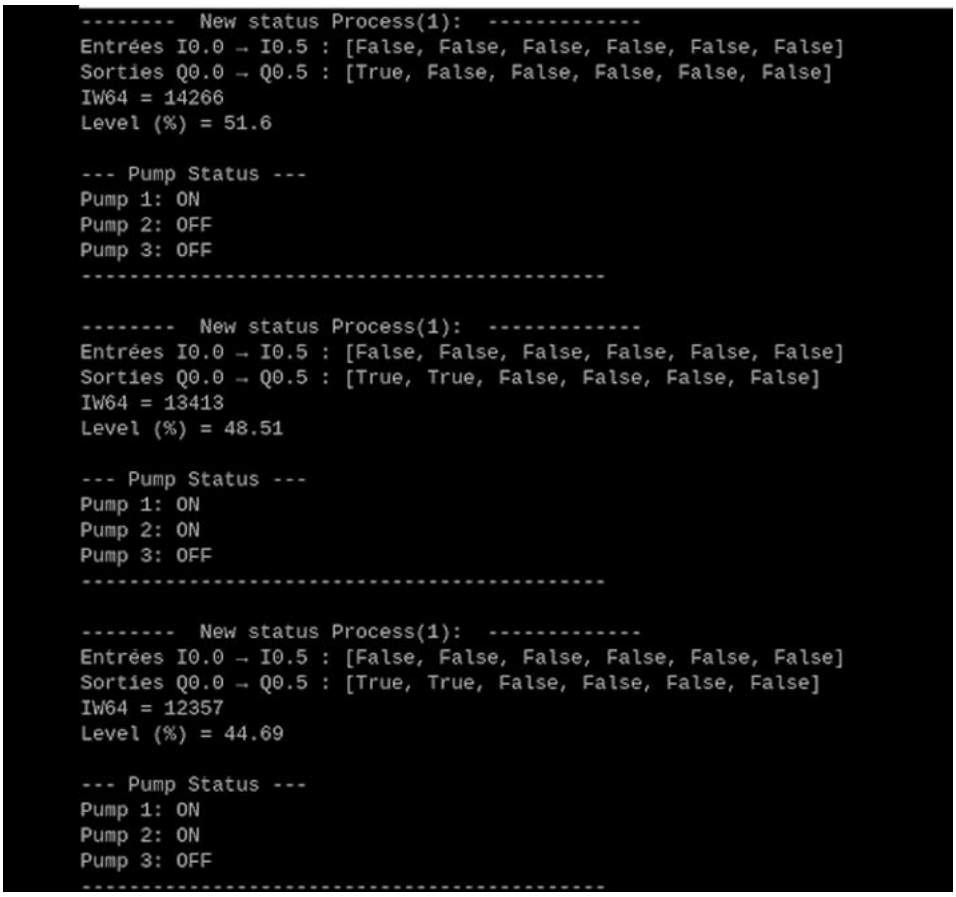


Figure 4.12 : Résultats des tests du Raspberry Pi 5 — Processus 1 (terminal)

3.4.2 Analyse des Données Collectées

L'analyse des résultats obtenus permet de tirer les conclusions suivantes :

Lecture 1 : Niveau = 51.6mm — La valeur brute IW64 = 14266 est correctement normalisée (14266 / 27648 × 100 = 51.6mm). Le niveau se situant entre 50mm et 85mm, seul le seuil N2 est dépassé (M0.2 = 1), ce qui entraîne la mise en service de la Pompe 1 uniquement (Q0.0 = 1).

Lecture 2 : Niveau = 48.51mm — Le niveau est passé sous le seuil N2 (50mm), ce qui active le seuil N3 uniquement. Les Pompes 1 et 2 sont en service (Q0.0 = 1, Q0.1 = 1).

Lecture 3 : Niveau = 44.69mm — Le niveau continue de baisser, restant sous 50mm. Les Pompes 1 et 2 restent en service, confirmant la cohérence de la logique de commande.

LECTURE	IW64 BRUT	NIVEAU (MM)	PUMP1	PUMP2	PUMP3
1	14266	51.6	ON	OFF	OFF
2	13413	48.51	ON	ON	OFF
3	12357	44.69	ON	ON	OFF

Tableau 4.6 : Synthèse des résultats de test — Processus 1

Ces résultats confirment que la logique de commande des pompes fonctionne correctement et que les données sont lues, traitées et publiées avec fidélité par le Raspberry Pi 5.

4. Processus 2 — Régulation de Niveau (Réel)

4.1 Description du Processus

4.1.1 Principe de Fonctionnement

Le second processus mis en œuvre est un système réel de régulation de niveau de liquide sur la maquette FLTP/EV. Contrairement au Processus 1, ce processus implique un système physique réel avec un liquide, un capteur analogique et un actionneur physique, offrant ainsi une validation expérimentale dans des conditions proches de la réalité industrielle.

Le principe de fonctionnement repose sur une régulation à seuils (régulation tout ou rien) : la pompe P1 alimente en continu le réservoir à débit constant, faisant monter progressivement le niveau du liquide. Lorsque ce niveau atteint le seuil supérieur de **300mm**, l'électrovanne de vidange s'ouvre automatiquement, permettant au liquide de s'écouler plus vite qu'il n'entre, ce qui fait baisser le niveau. Lorsque le niveau redescend au seuil inférieur de **250mm**, l'électrovanne se ferme, et le cycle recommence. Le niveau du liquide oscille ainsi de manière permanente et contrôlée entre 250mm et 300mm.

4.1.2 Les Variables du Système

Variable	*Adresse PLC*	*Type*	*Description*
level_input	IW64	Entrée analogique	Valeur brute capteur 0-27648
Tag_1	MD0	Mémoire réelle	Valeur normalisée intermédiaire
Tag_2	MD2	Mémoire réelle	Niveau mis à l'échelle (0-500mm)
mesure_en_litre	DB.mesure	Mémoire réelle	Mesure finale du niveau
Solenoid_valve	Q0.0	Sortie TOR	Commande de l'électrovanne

Tableau 4.7 : Variables du Processus 2

4.1.3 La Logique de Régulation

La logique de régulation est délibérément simple pour garantir la robustesse du système. Elle repose sur deux conditions exclusives :

Si le niveau est **inférieur à 250mm** → l'électrovanne est **fermée** (Reset de Q0.0) : le réservoir se remplit

Si le niveau est **supérieur à 300mm** → l'électrovanne est **ouverte** (Set de Q0.0) : le réservoir se vide

La plage de **50mm** entre les deux seuils (250mm et 300mm) constitue une zone morte (hystérésis) qui évite les oscillations rapides de l'électrovanne et prolonge sa durée de vie.

4.2 Programme Ladder — Processus 2 (TIA Portal)

Le programme Ladder du Processus 2 est structuré en trois réseaux distincts, chacun correspondant à une étape du traitement du signal analogique et de la logique de régulation.

4.2.1 Network 1 — Acquisition et Mise à l'Échelle du Niveau

Le premier réseau assure la conversion de la valeur brute du capteur piézométrique en une valeur de niveau physique en millimètres.

Le bloc **NORM_X** normalise la valeur entière de l'entrée analogique IW64 "level_input" (0 → 27648) en une valeur réelle normalisée (0.0 → 1.0), stockée dans MD0 "Tag_1". Le bloc **SCALE_X** met ensuite à l'échelle cette valeur sur la plage physique du capteur : de 0.0 à **500.0mm**, produisant la valeur MD2 "Tag_2".

4.2.2 Network 2 — Calcul de la Mesure

Le second réseau effectue une opération d'addition (bloc ADD) sur la valeur Tag_2, en y ajoutant 0.0, pour stocker le résultat dans la variable structurée "Donnee_process".mesure_en_litre. Cette étape permet d'intégrer la mesure dans une structure de données organisée, facilitant son exploitation par la logique de régulation du réseau suivant.

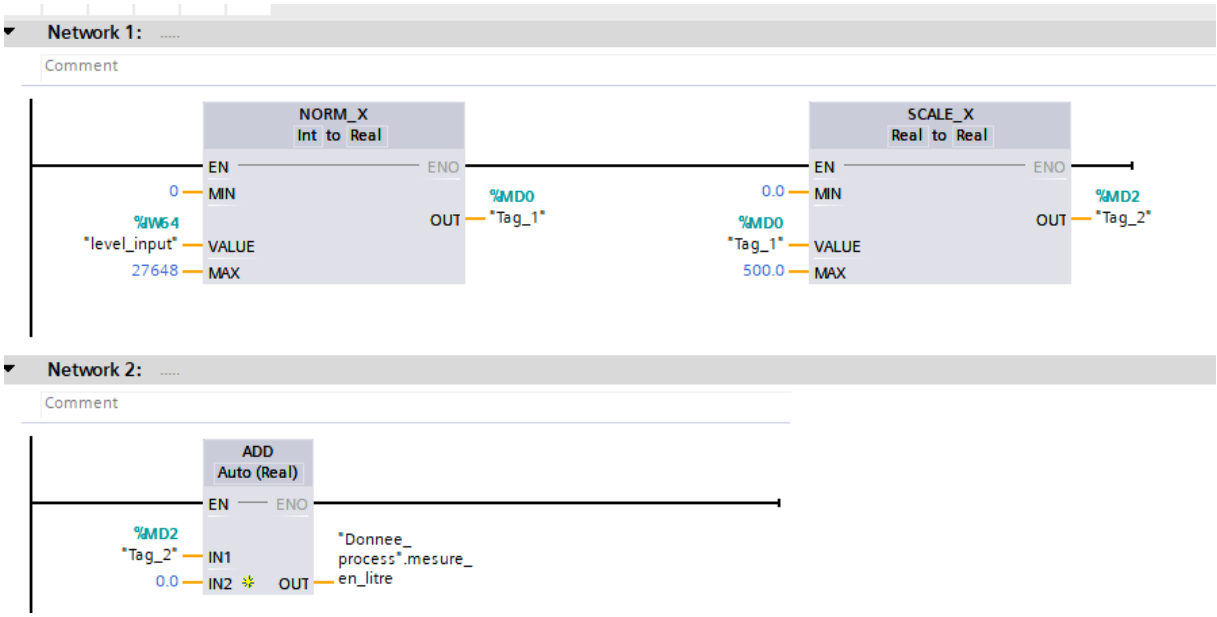


Figure 4.13 : Network 1 et 2 du Ladder Processus 2 — Acquisition et mise à l'échelle du niveau Calcul de la mesure

4.2.3 Network 3 — Logique de Commande de l'Électrovanne

Le troisième réseau constitue le cœur de la logique de régulation. Il compare en permanence la valeur de mesure aux deux seuils de régulation et commande l'électrovanne en conséquence :

Condition de fermeture : Si "Donnee_process".mesure_en_litre **< 250.0** → Reset de Q0.0 "Solenoid_valve" → l'électrovanne se ferme

Condition d'ouverture : Si "Donnee_process".mesure_en_litre **> 300.0** → Set de Q0.0 "Solenoid_valve" → l'électrovanne s'ouvre

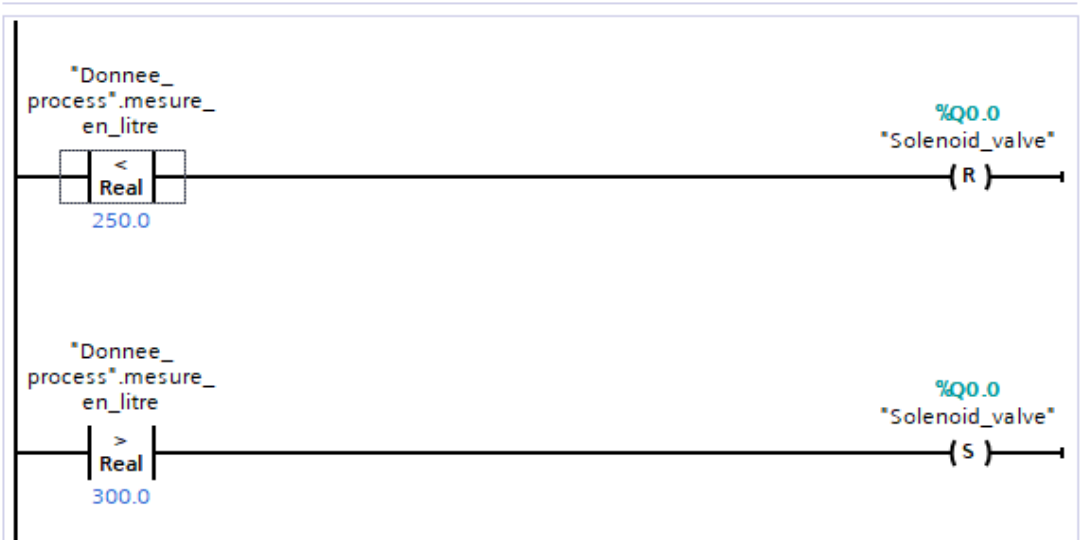


Figure 4.14 : Network 3 du Ladder Processus 2 — Logique de commande de l'électrovanne

4.3 Code Python — Raspberry Pi 5 (Processus 2)

4.3.1 Structure du Code

Le code Python du second Raspberry Pi 5 suit la même structure que celui du Processus 1, avec les adaptations nécessaires aux spécificités du Processus 2 : adresse mémoire analogique différente (IW64 au lieu de IW66), plage de normalisation différente (500mm au lieu de 100mm), et supervision de l'électrovanne au lieu des pompes.

```python
import snap7
from snap7.type import Area
from snap7.util import get_bool, get_int
import paho.mqtt.client as mqtt
import json
import time

# MQTT Setup - sends to itself (Pi 2)
mqtt_client = mqtt.Client()
mqtt_client.connect("10.186.148.13", 1883)

plc2 = snap7.client.Client()
ip_address2 = input("Enter PLC 2 IP address: ")
plc2.connect(ip_address2, 0, 1)

while True:
    print("-----------------------------------------")
    print("\n--------  New status Process(2):  -------------")

    data_inputs1 = plc2.read_area(Area.PE, 0, 0, 1)
    inputs1 = []
    for i in range(6):
        inputs1.append(get_bool(data_inputs1, 0, i))
    print("Entrées I0.0 → I0.5 :", inputs1)

    data_outputs1 = plc2.read_area(Area.PA, 0, 0, 1)
    outputs1 = []
    for i in range(6):
        outputs1.append(get_bool(data_outputs1, 0, i))
    print("Sorties Q0.0 → Q0.5 :", outputs1)

    data_analog1 = plc2.read_area(Area.PE, 0, 64, 2)
```

Figure 4.15 : Code Python du Raspberry Pi 5 — Processus 2 (partie 1 : connexion et lecture)

4.3.2 Lecture des Données et Publication MQTT

La lecture de l'entrée analogique s'effectue sur la zone PE, à l'**offset 64**, sur 2 octets, fournissant la valeur brute de IW64. La normalisation applique la formule : niveau_mm = (valeur_brute / 27648) × 500, produisant le niveau en millimètres sur une plage de 0 à 500mm.

L'état de l'électrovanne est extrait du premier bit de la zone des sorties PA (Q0.0), directement via la fonction get_bool(). Les données sont publiées sur le topic MQTT plc/2 sous la forme suivante :

```
    analog_value1 = get_int(data_analog1, 0)
    print("IW64 =", analog_value1)

    normalized1 = analog_value1 / 27648.0
    LEVEL_MIN = 0.0
    LEVEL_MAX = 500.0
    level_mm1 = normalized1 * (LEVEL_MAX - LEVEL_MIN) + LEVEL_MIN
    print("Level (mm) =", round(level_mm1, 2))

    valve = get_bool(data_outputs1, 0, 0)
    print("\n--- Solenoid valve status ---")
    print(f"Solenoid valve: {'ON' if valve else 'OFF'}")

    plc2_data = {
        "plc": 2,
        "inputs": inputs1,
        "outputs": outputs1,
        "level_mm": round(level_mm1, 2),
        "valve": valve
    }
    mqtt_client.publish("plc/2", json.dumps(plc2_data))

    time.sleep(10)

plc2.disconnect()
```

Figure 4.16 : Code Python du Raspberry Pi 5 — Processus 2 (partie 2 : publication MQTT)

4.4 Tests et Résultats — Raspberry Pi 5 (Processus 2)

4.4.1 Résultats des Lectures

Les tests du Raspberry Pi 5 sur le Processus 2 ont été réalisés en laissant le système de régulation fonctionner de manière autonome sur la maquette FLTP/EV. Les résultats du terminal montrent l'évolution réelle du niveau du liquide et l'état de l'électrovanne au cours du temps.

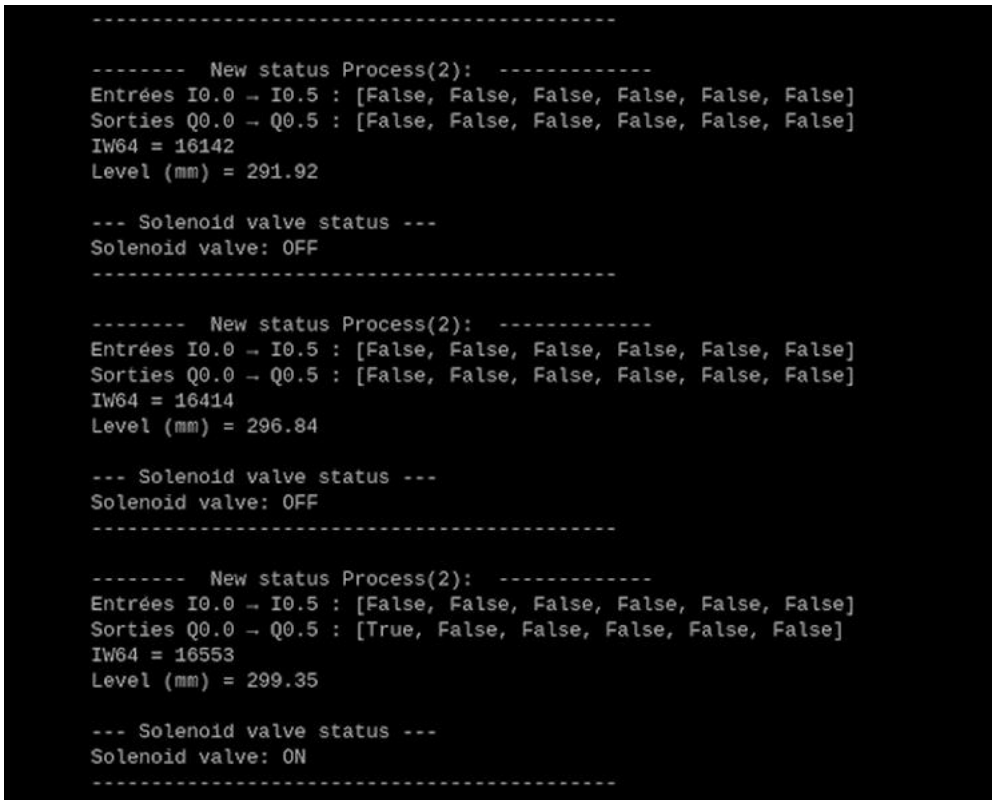


Figure 4.17 : Résultats des tests du Raspberry Pi 5 — Processus 2 (terminal)

4.4.2 Validation de la Régulation

L'analyse des résultats de test démontre le bon fonctionnement de la régulation de niveau :

Phase de vidange (électrovanne ouverte) :

Lecture 1 : Niveau = 284.69mm — Valve ON (IW64 = 15742)

Lecture 2 : Niveau = 277.36mm — Valve ON (IW64 = 15337)

Lecture 3 : Niveau = 269.71mm — Valve ON (IW64 = 14914)

Le niveau diminue progressivement, confirmant que la vidange est en cours. La vitesse de descente est d'environ 7-8mm par cycle de 10 secondes.

Phase de remplissage (électrovanne fermée) :

Lecture 4 : Niveau = 291.92mm — Valve OFF (IW64 = 16142)

Lecture 5 : Niveau = 296.84mm — Valve OFF (IW64 = 16414)

Lecture 6 : Niveau = 299.35mm — Valve ON (IW64 = 16553)

Le niveau remonte progressivement jusqu'à atteindre le seuil de 300mm, déclenchant l'ouverture de l'électrovanne. La vitesse de montée est d'environ 4-5mm par cycle de 10 secondes.

Lecture	IW64 brut	Niveau (mm)	Électrovanne	Phase
1	15742	284.69	ON	Vidange
2	15337	277.36	ON	Vidange
3	14914	269.71	ON	Vidange
4	16142	291.92	OFF	Remplissage
5	16414	296.84	OFF	Remplissage
6	16553	299.35	ON	Seuil atteint

Tableau 4.8 : Synthèse des résultats de test — Processus 2

Ces résultats confirment que la régulation fonctionne correctement dans la plage définie (250-300mm) et que les données sont lues et transmises avec précision par le Raspberry Pi 5.

5. Le Serveur Applicatif

5.1 Description et Rôle

Le serveur applicatif constitue le composant central de traitement et de redistribution des données dans l'architecture du système. Il joue le rôle d'intermédiaire entre la couche de transport MQTT et la couche application (interface de supervision), assurant la réception des données publiées par les deux passerelles et leur transmission en temps réel vers l'interface web.

5.2 Structure du Code

Le serveur est développé en **Python**, en utilisant deux bibliothèques complémentaires :

Flask est un framework web léger et flexible qui permet de créer rapidement des applications web en Python. Dans ce projet, Flask remplit deux rôles : servir les fichiers statiques de l'interface (fichier HTML/CSS/JS) et fournir le point d'entrée de l'application web accessible sur le port 5000.

Flask-SocketIO est une extension de Flask qui intègre le protocole **WebSocket** via la bibliothèque Socket.IO . Ce protocole permet une communication bidirectionnelle et persistante entre le serveur et le navigateur web, garantissant la mise à jour de l'interface en temps réel dès réception de nouvelles données, sans nécessiter de rechargement de la page.

Server.py

```python
from flask import Flask, render_template
from flask_socketio import SocketIO
import paho.mqtt.client as mqtt
import json

app = Flask(__name__)
socketio = SocketIO(app)

def on_message(client, userdata, message):
    data = json.loads(message.payload.decode())
    socketio.emit("plc_update", data)

mqtt_client = mqtt.Client()
mqtt_client.on_message = on_message
mqtt_client.connect("localhost", 1883)
mqtt_client.subscribe("plc/#")
mqtt_client.loop_start()

@app.route("/")
def index():
    return render_template("index.html")

if __name__ == "__main__":
    socketio.run(app, host="0.0.0.0", port=5000)
```

Figure 4.18 : Code complet du serveur applicatif (Server.py)

5.3 Réception des Données MQTT

Le serveur s'abonne à tous les topics MQTT publiés par les passerelles en utilisant le caractère générique plc/#, qui correspond à tous les topics commençant par plc/. Cela inclut plc/1 (Processus 1) et plc/2 (Processus 2).

La fonction de rappel on_message() est déclenchée automatiquement à chaque réception d'un message MQTT. Elle décode le payload JSON du message et émet immédiatement un événement WebSocket plc_update vers tous les clients connectés à l'interface, en transmettant les données décodées sans modification.

5.4 Transmission vers l'Interface via WebSocket

Le mécanisme de transmission en temps réel repose sur le couple **MQTT Subscribe → WebSocket Emit**. Dès qu'un message arrive sur le broker MQTT (toutes les 10 secondes, lors de chaque cycle de lecture des passerelles), le serveur le reçoit instantanément grâce à la boucle mqtt_client.loop_start() qui s'exécute en arrière-plan, puis le redistribue immédiatement vers l'interface via socketio.emit().

5.5 Tests et Résultats du Serveur

Le serveur fonctionne de manière stable et continue, sans interruption ni perte de données observée durant les sessions de test. La latence entre la publication MQTT par la passerelle et l'affichage des données sur l'interface est inférieure à **une seconde**, garantissant une supervision véritablement en temps réel.

6. L'Interface de Supervision

6.1 Description Générale

L'interface de supervision est une application web développée en **HTML5, CSS3 et JavaScript**, servie par le serveur Flask et accessible depuis n'importe quel navigateur web sur le réseau local. Elle constitue le point de contact entre le système de supervision et l'opérateur humain, transformant les données brutes transmises par les passerelles en informations visuelles claires et exploitables.

L'interface adopte un **thème sombre professionnel** (dark theme), avec une couleur principale verte (#00ff88) évoquant les interfaces de supervision industrielles modernes. Elle est structurée en deux parties principales correspondant aux deux processus supervisés, complétées par des graphiques d'historique.

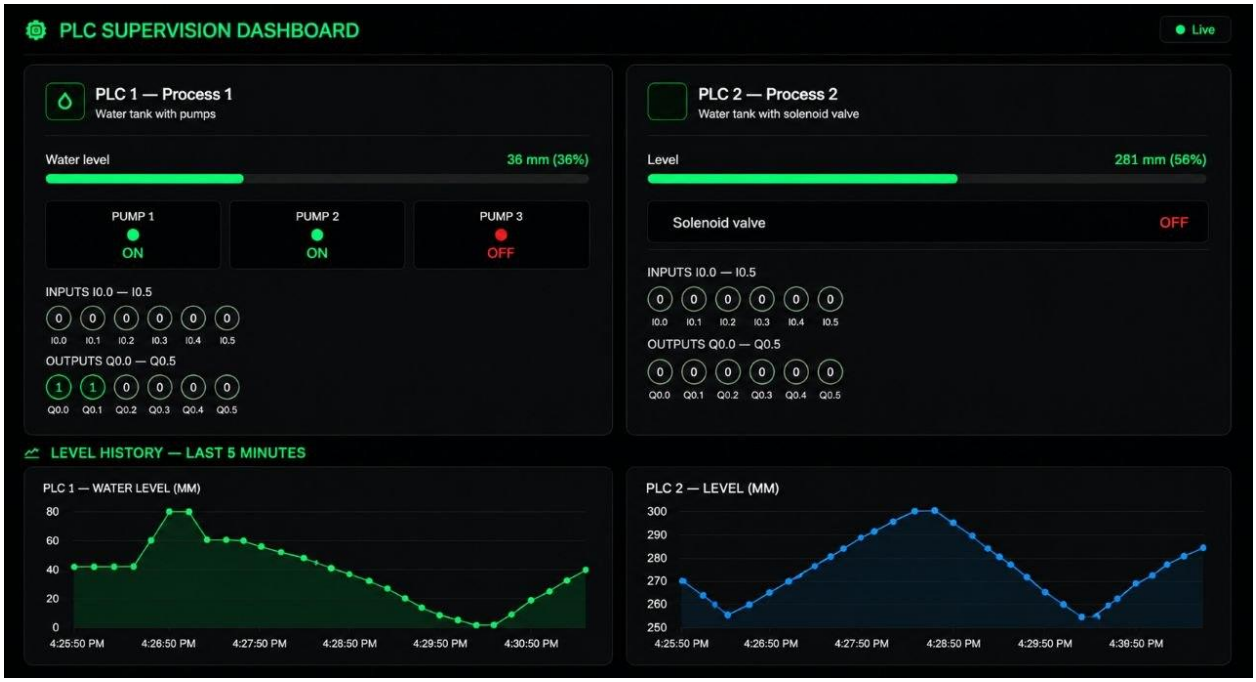


Figure 4.19 : Interface de supervision complète — vue générale avec les deux processus et les graphiques

6.2 Structure de l'Interface

6.2.1 En-tête et Indicateur de Connexion

L'en-tête de l'interface affiche le titre "PLC Supervision Dashboard" accompagné d'un indicateur de connexion dynamique. Cet indicateur affiche "Live" en vert lorsque la connexion WebSocket avec le serveur est active, et "Disconnected" en rouge en cas de perte de connexion. Il fournit ainsi à l'opérateur une confirmation visuelle permanente du bon fonctionnement du système de supervision.

6.2.2 Carte PLC 1 — Processus 1 (Water tank with pumps)

La carte du Processus 1 présente les informations suivantes :

Niveau d'eau : Une barre de progression horizontale affiche le niveau actuel du réservoir en millimètres et en pourcentage. La couleur de la barre évolue selon le niveau : verte pour un niveau normal (< 60%), orange pour un niveau élevé (60-80%), et rouge pour un niveau critique (> 80%).

État des pompes : Trois blocs distincts affichent l'état de chaque pompe (Pump 1, Pump 2, Pump 3) avec un indicateur LED circulaire — vert pour ON, rouge pour OFF — et la valeur textuelle correspondante.

Entrées/Sorties : Les états binaires des entrées (I0.0 → I0.5) et des sorties (Q0.0 → Q0.5) sont affichés sous forme de points colorés — vert allumé pour l'état 1, gris pour l'état 0 — avec le nom de chaque adresse affiché en dessous.

6.2.3 Carte PLC 2 — Processus 2 (Water tank with solenoid valve)

La carte du Processus2 présente une structure similaire, adaptée aux spécificités de ce processus :

Niveau : Barre de progression sur une plage de 0 à 500mm, avec affichage de la valeur en millimètres et en pourcentage.

État de l'électrovanne : Un indicateur unique affiche l'état de la Solenoid Valve — "ON" en rouge (électrovanne ouverte, vidange en cours) ou "OFF" en vert (électrovanne fermée, remplissage en cours).

Entrées/Sorties : Affichage identique au Processus 1.

6.2.4 Graphiques d'Historique

La partie inférieure de l'interface présente trois graphiques générés par la bibliothèque **Chart.js** :

Graphique PLC 1 — Water Level (mm) : Courbe verte affichant l'évolution du niveau du Processus 1 sur les 5 dernières minutes (30 points maximum, un point toutes les 10 secondes).

Graphique PLC 2 — Level (mm) : Courbe bleue affichant l'évolution du niveau du Processus 2 sur la même période.

Graphique Pump Status History : Trois courbes superposées (verte pour Pump 1, orange pour Pump 2, bleue pour Pump 3) affichant l'historique de l'état ON/OFF de chaque pompe du Processus 1.

6.3 Fonctionnalités de l'Interface

Le code JavaScript de l'interface gère la réception des événements WebSocket et la mise à jour dynamique de tous les éléments graphiques en temps réel. À chaque réception d'un événement plc_update, le code identifie le processus concerné (PLC 1 ou PLC 2 selon le champ data.plc), met à jour tous les éléments correspondants et ajoute le nouveau point aux graphiques d'historique.

La gestion des graphiques limite automatiquement le nombre de points affichés à 30 (paramètre MAX_POINTS), assurant une lisibilité optimale et évitant la surcharge mémoire du navigateur lors de sessions de supervision prolongées.

6.4 Résultats — Interface en Fonctionnement

Les captures d'écran de l'interface en fonctionnement, obtenues lors des sessions de test simultané des deux processus, démontrent le bon fonctionnement de l'ensemble du système de supervision.

Sur la capture présentée en Figure 19, on observe :

PLC 1 — Processus 1 : Niveau à 36mm (36%), Pump 1 ON, Pump 2 ON, Pump 3 OFF — cohérent avec un niveau inférieur à 50mm

PLC 2 — Processus 2 : Niveau à 281mm (56%), Solenoid valve OFF — le réservoir est en phase de remplissage

Graphiques : L'historique du PLC 1 montre une variation du niveau entre 20mm et 80mm ; l'historique du PLC 2 montre clairement les oscillations de niveau entre 250mm et 300mm, confirmant le bon fonctionnement de la régulation

7. Discussion et Analyse des Résultats

7.1 Validation de l'Architecture Globale

Les résultats expérimentaux obtenus sur les deux processus valident l'architecture IoT de supervision industrielle proposée dans ce projet . L'ensemble de la chaîne de données — depuis la lecture des variables dans la mémoire de l'automate jusqu'à leur affichage sur l'interface web — fonctionne de manière fiable, continue et cohérente.

La supervision simultanée des deux processus depuis une interface unifiée démontre concrètement la **scalabilité** de l'architecture : l'ajout du second processus n'a nécessité aucune modification du broker, du serveur ou de l'interface existants. Il a suffi de connecter un second Raspberry Pi au second automate et de définir un nouveau topic MQTT.

La **non-intrusivité** du système est également confirmée : les deux automates ont fonctionné normalement tout au long des sessions de test, sans aucune perturbation liée à la présence des passerelles de supervision . Le temps de cycle des automates n'a pas été affecté, et les connexions Snap7 n'ont causé aucune interruption ou dysfonctionnement du programme de commande.

7.2 Performances du Système

Indicateur	*Valeur mesurée*
Période de lecture	10 secondes
Latence MQTT (passerelle → serveur)	< 100 ms
Latence WebSocket (serveur → interface)	< 500 ms
Latence totale (PLC → interface)	< 1 seconde
Stabilité de la connexion Snap7	Aucune déconnexion observée
Précision de la lecture analogique	± 0.5mm
Validation de la régulation (Processus 2)	Niveau maintenu entre 250mm et 300mm

Tableau 4.9 : Synthèse des performances du système

7.3 Perspectives d'Amélioration

Les limites identifiées ouvrent des perspectives d'amélioration claires pour les travaux futurs :

Intégration d'une base de données temporelle (InfluxDB, TimescaleDB) pour la persistance et l'historisation à long terme des données

Réduction de la période de rafraîchissement à 1-2 secondes pour une supervision plus réactive

Implémentation de la sécurité : chiffrement TLS/SSL pour MQTT et WebSocket, authentification des utilisateurs sur l'interface

Ajout d'un système d'alarmes : détection des dépassements de seuils et notifications par email ou SMS.

Migration vers le Cloud : déploiement du serveur sur une infrastructure Cloud pour un accès mondial et une disponibilité maximale [41].

Intégration de l'intelligence artificielle : analyse des données historiques pour la maintenance prédictive et la détection d'anomalies.

8. Conclusion

Ce chapitre a présenté en détail la réalisation complète du système IoT de supervision industrielle développé dans ce projet, depuis la programmation des automates jusqu'à l'affichage des données sur l'interface web, en passant par la configuration des passerelles, du broker MQTT et du serveur applicatif.

La mise en œuvre du système sur deux processus industriels distincts — un processus virtuel de commande de pompes et un processus réel de régulation de niveau — a permis de valider expérimentalement l'architecture proposée au Chapitre 3 et de démontrer son universalité. Dans les deux cas, les données ont été lues avec précision depuis les automates Siemens S7-1200 via la bibliothèque Snap7, transmises de manière fiable via le protocole MQTT, et affichées en temps réel sur l'interface de supervision.

Les résultats obtenus confirment la faisabilité technique et la pertinence de l'approche adoptée : une solution IoT industrielle open-source, non-intrusive et économique, capable de connecter des automates programmables existants à une infrastructure de supervision moderne et accessible. Ce projet démontre ainsi que la transition vers l'Industrie 4.0 est accessible à toute installation industrielle disposant d'un automate programmable Siemens, sans investissement majeur en équipements propriétaires.

Conclusion Générale

Le présent travail avait pour objectif principal de concevoir et réaliser une solution IoT industrielle permettant la supervision à distance de systèmes automatisés pilotés par des automates programmables Siemens S7-1200, en s'appuyant exclusivement sur des technologies open-source et une approche non-intrusive ne nécessitant aucune modification des équipements existants.

Dans un contexte industriel marqué par la transition vers l'Industrie 4.0 et la nécessité croissante de connectivité et de traçabilité des données de production, ce projet apporte une réponse concrète, économique et universelle à la problématique de l'isolement informationnel des automates programmables industriels.

Les principales étapes de notre démarche ont été :

- L'étude approfondie des technologies IoT industrielles, des protocoles de communication et des architectures de supervision, constituant le socle théorique nécessaire à la conception du système,
- La conception d'une architecture IoT en cinq couches fonctionnelles, organisée autour de la bibliothèque Snap7 pour la communication avec les automates, du protocole MQTT pour le transport des données, d'un serveur Flask pour le traitement et la redistribution, et d'une interface web pour la visualisation en temps réel,
- La réalisation concrète du système sur deux processus industriels distincts, validant l'universalité et la scalabilité de l'architecture proposée,
- L'évaluation expérimentale des performances du système, démontrant une latence totale inférieure à une seconde entre la lecture des données dans l'automate et leur affichage sur l'interface de supervision.

Les résultats obtenus confirment la faisabilité technique et la pertinence de l'approche adoptée. La supervision simultanée des deux processus depuis une interface web unifiée, accessible depuis un ordinateur, une tablette ou un smartphone, démontre concrètement que la transition vers l'Industrie 4.0 est accessible à toute installation industrielle disposant d'un automate programmable Siemens, sans investissement majeur en équipements propriétaires.

Perspectives :

Ce travail ouvre la voie à plusieurs perspectives d'évolution, parmi lesquelles :

- L'intégration d'une base de données temporelle pour la persistance et l'historisation à long terme des données de production,
- L'ajout d'un flux de contrôle bidirectionnel permettant l'envoi de commandes vers l'automate depuis l'interface de supervision,
- Le déploiement du serveur sur une infrastructure Cloud pour un accès mondial et une disponibilité maximale,

- L'intégration d'algorithmes d'intelligence artificielle pour la maintenance prédictive et la détection automatique d'anomalies,
- L'extension du système à d'autres familles d'automates programmables, au-delà de la gamme Siemens S7.

En somme, ce mémoire confirme le potentiel des technologies IoT open-source pour moderniser les installations industrielles existantes et contribuer à leur transition vers l'Industrie 4.0. Il constitue une première étape vers des systèmes de supervision industrielle connectés, accessibles et intelligents, au service de la performance et de la compétitivité du tissu industriel national.

BIBLIOGRAPHIE

[1] Ashton, K. (2009). That 'Internet of Things' Thing. *RFID Journal*.

[2] Rose, K., Eldridge, S., & Chapin, L. (2015). *The Internet of Things: An Overview*. Internet Society.

[3] IoT Analytics (2023). *State of IoT Report*. IoT Analytics GmbH.

[4] Gartner (2023). *Hype Cycle for the Internet of Things*.

[5] Uckelmann, D., Harrison, M., & Michahelles, F. (2011). *Architecting the Internet of Things*. Springer.

[6] ITU-T Y.2060 (2012). *Overview of the Internet of Things*. International Telecommunication Union.

[7] Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15), 2787–2805.

[8] Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), 1645–1660.

[9] ISO/IEC 20922:2016. *MQTT Protocol Standard — Message Queuing Telemetry Transport*. International Organization for Standardization.

[10] IEC 61158. *Industrial Communication Networks — Fieldbus Specifications*. International Electrotechnical Commission.

[11] RFC 7252 (2014). *The Constrained Application Protocol (CoAP)*. IETF.

[12] Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of Things for Smart Cities. *IEEE Internet of Things Journal*, 1(1), 22–32.

[13] McKinsey Global Institute (2015). *The Internet of Things: Mapping the Value Beyond the Hype*.

[14] Site officiel MQTT : https://mqtt.org

[15] Documentation Node-RED : https://nodered.org/docs

[16] Documentation ThingsBoard : https://thingsboard.io/docs

[17] Documentation Grafana : https://grafana.com/docs

[18] Laribi N., Bouri A. Mr MEGNAFI Hicham, (2022). *Design of Air Quality Monitoring Quadrotor*. Mémoire d'Ingénieur, filière Automatique, École Supérieure en Sciences Appliquées de Tlemcen (ESSA-Tlemcen).

[19] Loulhaci R., Bensouna B. Mr ABDELLAOUI Ghouti, Mrs. SEBBAGH Hafidha (2024). *Implementation of an Intelligent Automated Traffic Light System*. Mémoire d'Ingénieur, filière Automatique, École Supérieure en Sciences Appliquées de Tlemcen (ESSA-Tlemcen).

[20] Philips Healthcare. (2025). https://www.philips.com/healthcare/patient-monitoring

[21] NEOM Company (2024–2025). https://www.neom.com/en-us/regions/neom/technology

[22] Brandl, D., & Paprocki, M. (2021). *Industrial Automation and Control Systems*. ISA — International Society of Automation. ISBN: 978-1-945541-78-4.

[23] Thramboulidis, K., & Christoulakis, F. (2022). UML4IoT: A UML-based approach for the development of IoT applications in industrial environments. *Computers in Industry*, 136, 103580. https://doi.org/10.1016/j.compind.2021.103580

[24] Wollschlaeger, M., Sauter, T., & Jasperneite, J. (2017). The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Industrial Electronics Magazine*, 11(1), 17–27. https://doi.org/10.1109/MIE.2017.2649104

[25] Felser, M. (2021). Real-Time Ethernet — Industry prospective. *Proceedings of the IEEE*, 109(4), 544–557. https://doi.org/10.1109/JPROC.2021.3060450

[26] Leitão, P., Colombo, A. W., & Karnouskos, S. (2023). Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*, 81, 11–25. https://doi.org/10.1016/j.compind.2015.08.004

[27] IEC 61131-3:2013. *Programmable controllers — Part 3: Programming languages*. International Electrotechnical Commission.

[28] IEC 62541. *OPC Unified Architecture (OPC UA)*. International Electrotechnical Commission.

[29] Modbus Organization. (2021). *Modbus Application Protocol Specification V1.1b3*. https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf

[30] Mourtzis, D., Angelopoulos, J., & Panopoulos, N. (2022). A literature review of the challenges and opportunities of the transition from industry 4.0 to society 5.0. *Energies*, 15(17), 6276. https://doi.org/10.3390/en15176276

[31] Sisinni, E., Saifullah, A., Han, S., Jennehag, U., & Gidlund, M. (2022). Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11), 4724–4734. https://doi.org/10.1109/TII.2018.2852491

[32] Yoong, C. K., et al. (2023). A systematic review on SCADA systems in industrial environments. *Journal of Industrial Information Integration*, 31, 100418. https://doi.org/10.1016/j.jii.2022.100418

[33] Profibus & Profinet International. (2023). *PROFINET — The Industrial Ethernet Standard*. https://www.profibus.com/technology/profinet

[34] OPC Foundation. (2023). *OPC UA — Unified Architecture*. https://opcfoundation.org/about/opc-technologies/opc-ua

[35] Nardella, D. (2023). *Snap7 — A multi-platform Siemens S7 Communication Suite*. Documentation officielle. http://snap7.sourceforge.net

[36] Blocher, A., & Schmitt, J. (2022). Industrial IoT integration using open-source S7 communication libraries. *Journal of Industrial Information Integration*, 28, 100350. https://doi.org/10.1016/j.jii.2022.100350

[37] Mishra, B., & Kertesz, A. (2021). The use of MQTT in M2M and IoT systems: A survey. *IEEE Access*, 8, 201071–201086. https://doi.org/10.1109/ACCESS.2020.3035849

[38] Eclipse Foundation. (2023). *Eclipse Mosquitto — An Open Source MQTT Broker*. Documentation officielle. https://mosquitto.org/documentation

[39] Boyes, H., Hallaq, B., Cunningham, J., & Watson, T. (2021). The industrial internet of things (IIoT): An analysis framework. *Computers in Industry*, 101, 1–12. https://doi.org/10.1016/j.compind.2018.04.015

[40] Raspberry Pi Foundation. (2023). *Raspberry Pi 5 — Official Documentation*. https://www.raspberrypi.com/documentation

[41] Khodaei, M., & Kiavi, H. (2022). Edge computing platforms for industrial IoT applications: A comparative study. *Internet of Things*, 18, 100505. https://doi.org/10.1016/j.iot.2022.100505

[42] Naik, N. (2021). Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. *IEEE Systems, Man, and Cybernetics*, 1–7. https://doi.org/10.1109/SMC.2017.8122697

[43] ECMA International. (2022). *ECMA-404 — The JSON Data Interchange Standard* (2nd ed.). https://www.ecma-international.org/publications-and-standards/standards/ecma-404

[44] Siemens AG. (2023). *SIMATIC S7-1200 Programmable Controller System Manual*. Siemens Industry Online Support. https://support.industry.siemens.com

[45] Grinberg, M. (2021). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media.

[46] Ronacher, A. (2023). *Flask Documentation* (version 3.0). https://flask.palletsprojects.com

[47] Grinberg, M. (2022). *Flask-SocketIO Documentation*. https://flask-socketio.readthedocs.io

[48] Elettronica Veneta SPA. (2022). *Manuel API-24/EV — Panneau Didactique pour l'Étude de l'API*. Elettronica Veneta.

[49] Elettronica Veneta SPA. (2022). *Manuel FLTP/EV — Fluid Level Training Process*. Elettronica Veneta.

[50] Siemens AG. (2023). *TIA Portal — STEP 7 Professional Programming Manual*. Siemens Industry Online Support.

[51] John, K. H., & Tiegelkamp, M. (2022). *IEC 61131-3: Programming Industrial Automation Systems* (3rd ed.). Springer. https://doi.org/10.1007/978-3-642-12015-2

[52] Chart.js Contributors. (2023). *Chart.js — Simple yet flexible JavaScript charting library*. https://www.chartjs.org/docs