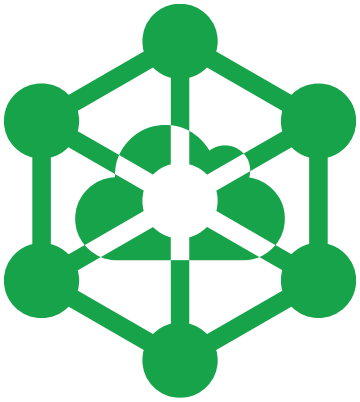


C216

Intégrer les terminaux IoE dans une plateforme existante

Module ICT-216

Introduction à l'IoE

L'Internet of Everything (IoE) étend le concept de l'Internet des objets en intégrant non seulement les objets connectés, mais aussi les personnes, les données et les processus dans un écosystème intelligent.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Définir les concepts d'**IoT** et d'**IoE** et distinguer leurs différences.
- Identifier les **4 piliers** de l'IoE (objets, personnes, données, processus).
- Reconnaître les principaux **domaines d'application** de l'IoE.
- Comprendre l'**impact** des terminaux IoE sur les utilisateurs et les processus métier.

IoT « Internet of Things »

L'Internet des objets (IoT) désigne un ensemble d'**objets physiques connectés** capables de :

- 📡 **Capter** des informations (température, mouvement, luminosité...)
- 📤 **Transmettre** des données via un réseau
- ⚙️ **Agir** automatiquement dans certains cas

Exemples d'objets IoT

Objet	Fonction
Capteur de température	Mesure et transmet la température ambiante
Montre connectée	Suit l'activité physique et les notifications
Ampoule connectée	S'allume/éteint à distance ou automatiquement
Frigo intelligent	Surveille les stocks alimentaires

Origine de l'IoT

Le terme **Internet of Things** a été inventé par **Kevin Ashton** en **1999**, dans le contexte de la technologie **RFID** appliquée à la gestion de produits en rayon.

L'IoT est né de la convergence de plusieurs facteurs :

- Développement d'**Internet**
- **Miniaturisation** des composants électroniques
- **Baisse du coût** des capteurs
- Généralisation du **Wi-Fi**, du **Bluetooth** et des **réseaux mobiles**
- Besoin de **collecter et exploiter** des données en temps réel

---

IoE « Internet of Everything »

L'IoE va plus loin que l'IoT. Il ne s'agit pas seulement de connecter des objets, mais de créer un **écosystème complet** reliant :

Les 4 piliers de l'IoE

Pilier	Description
Objets	Appareils physiques et objets connectés à Internet et entre eux (IoT)
Personnes	Utilisateurs, techniciens, clients — connectés de manière pertinente
Données	Informations récoltées, analysées, stockées et exploitées pour la prise de décision
Processus	Règles, automatisations et workflows qui fournissent la bonne information au bon moment

Exemple de flux IoE

📡 Capteur détecte un événement → 📤 Donnée envoyée →
⚙️ Règle s'applique → 🔔 On reçoit une alerte

Différence entre IoT et IoE

Critère	IoT	IoE
Focus	Les **objets connectés**	L'**écosystème** complet
Composantes	Objets + réseau + données	Objets + personnes + données + processus
Exemple	Une montre mesure le rythme cardiaque	La montre transmet à une app, alerte un médecin, adapte un programme

En résumé

🔌 **IoT**
les objets connectés

⊂

🌐 **IoE**
les objets + les personnes + l'analyse + les décisions

Domaines d'application

L'IoE s'applique à de nombreux secteurs :

Domaine	Exemples
🏠 **Maison connectée**	Thermostat, éclairage, sécurité, assistants vocaux
🏭 **Industrie**	Maintenance prédictive, robots, contrôle qualité
🏥 **Santé**	Suivi patient, dispositifs médicaux, télémédecine
🌾 **Agriculture**	Irrigation intelligente, drones, surveillance des sols
📦 **Logistique**	Suivi de flotte, gestion des stocks, traçabilité
🏙️ **Smart City**	Gestion du trafic, éclairage public, gestion des déchets

Exemple concret : la maison intelligente

Composante IoE	Éléments
Objets	Thermostat, capteurs de présence, lumières
Personnes	Habitants
Données	Température, consommation, horaires
Processus	Chauffage automatique selon présence et heure

Les bénéfices : **confort** au quotidien, **économie** d'énergie, **automatisation** des tâches et **meilleure gestion** des ressources.

Impact sur les utilisateurs

Les terminaux IoE influencent les processus métier et le quotidien :

Impact positif	Impact négatif potentiel
Automatisation des tâches répétitives	Dépendance technologique
Collecte de données en temps réel	Risques liés à la vie privée
Confort et surveillance améliorés	Complexité de maintenance

Impact positif	Impact négatif potentiel
Prise de décision basée sur les données	Vulnérabilités de sécurité

Résumé

L'IoE étend l'IoT en intégrant personnes, données et processus autour des objets connectés. Cette vision globale permet de créer des écosystèmes intelligents dans des domaines variés comme la domotique, l'industrie ou la santé. Comprendre la différence entre IoT et IoE est fondamental pour appréhender l'intégration de terminaux dans une plateforme existante. L'impact de ces technologies est à la fois une source d'opportunités et de responsabilités.

Terminaux IoE

Les terminaux IoE sont les éléments physiques qui interagissent avec le monde réel : ils captent, agissent, relaient ou combinent ces fonctions. Comprendre leur classification et leur structure interne est essentiel pour les intégrer correctement.

Objectifs

À la fin de ce chapitre, vous serez capable de :

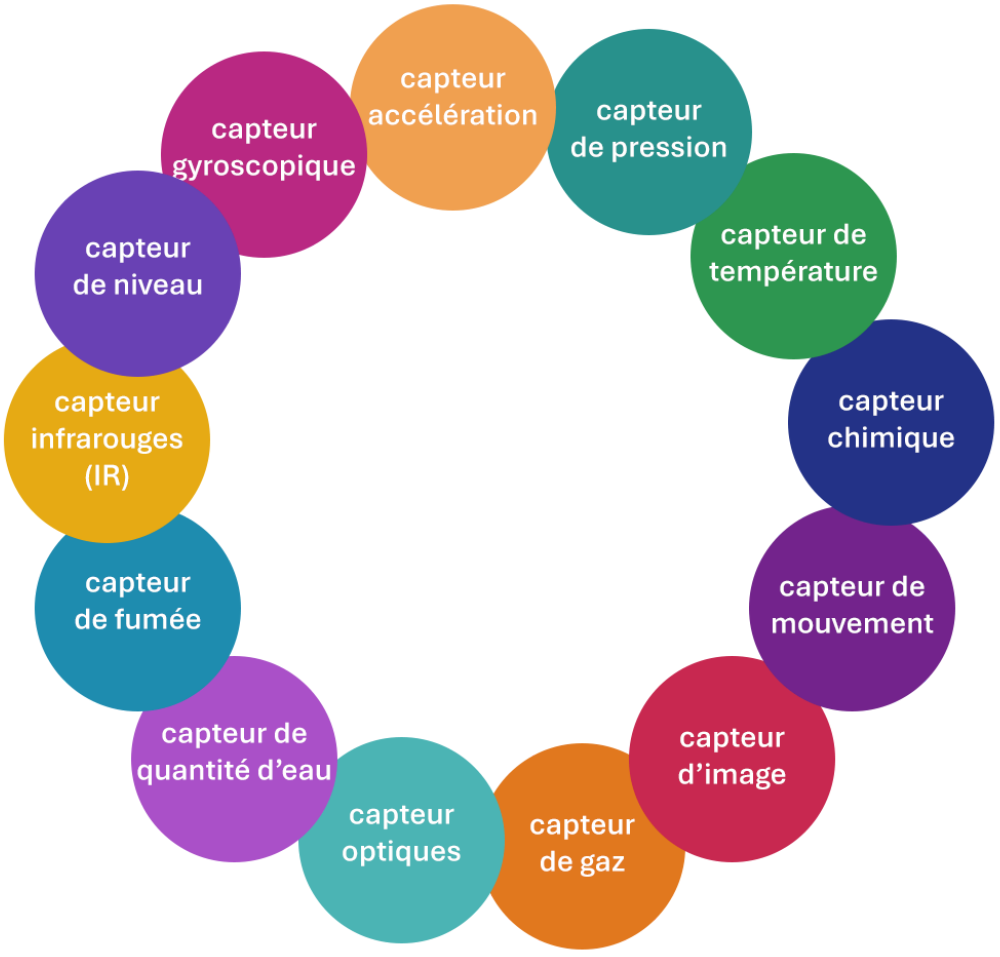
- Distinguer les différents **types de terminaux** IoE (capteur, actionneur, passerelle, hybride).
- Décrire la **structure interne** d'un terminal IoE en 4 couches.
- Identifier le **rôle** d'un terminal dans un écosystème connecté.
- Analyser un terminal réel et en déduire ses caractéristiques.

Types de terminaux

Capteurs

Les capteurs mesurent une **grandeur physique** et transmettent l'information sous forme de données numériques.

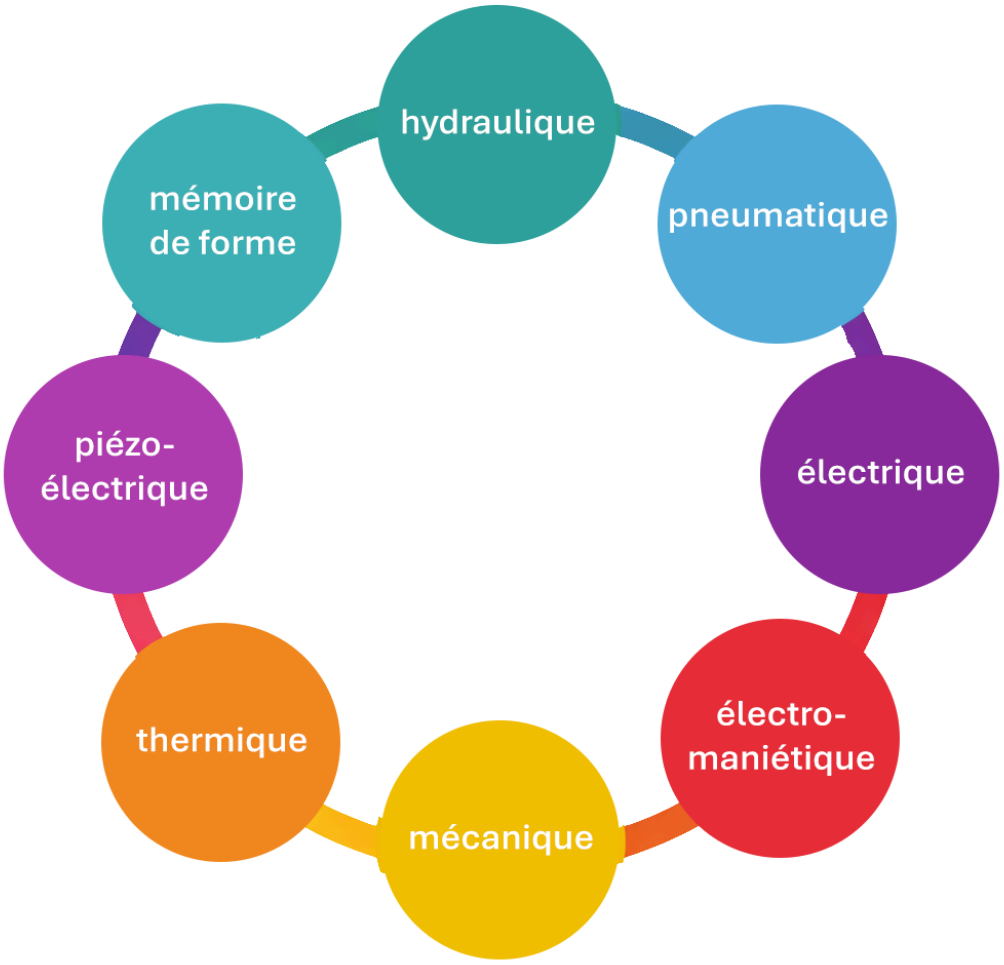
Grandeur mesurée	Exemples de capteurs
Température	Sonde thermique, thermomètre connecté
Humidité	Capteur hygrométrique
Mouvement	Détecteur PIR, accéléromètre
Luminosité	Photorésistance, capteur LUX
Pression	Baromètre, capteur de pression



Actionneurs

Les actionneurs **agissent sur l'environnement** en réponse à une commande reçue.

Type d'action	Exemples
Mouvement mécanique	Moteur, servo-moteur
Ouverture/fermeture	Vanne connectée, serrure
Signal lumineux	LED, bandeau LED
Signal sonore	Buzzer, sirène
Commutation	Relais, prise connectée



Passerelles (Gateways)

Les passerelles font le **lien entre les terminaux** et le réseau ou le cloud. Elles agrègent, convertissent et transmettent les données.

Exemples : box domotique, hub Zigbee, gateway LoRaWAN

Une passerelle n'est généralement pas un périphérique final comme un capteur de température ou une lampe connectée. Son rôle principal est de relayer, traduire, filtrer, sécuriser ou agréger des données.

Fonctions typiques d'une passerelle :

- Conversion de protocole, par exemple Zigbee/Bluetooth/LoRa → IP
- Envoi de données vers un serveur, le cloud ou une plateforme locale
- Traitement en périphérie
- Gestion des périphériques
- Contrôle de sécurité

Objets hybrides

Les objets hybrides **combinent** capteur, actionneur et connectivité dans un seul dispositif.

Exemples : montre connectée, thermostat intelligent, robot aspirateur

---

Structure d'un terminal IoE

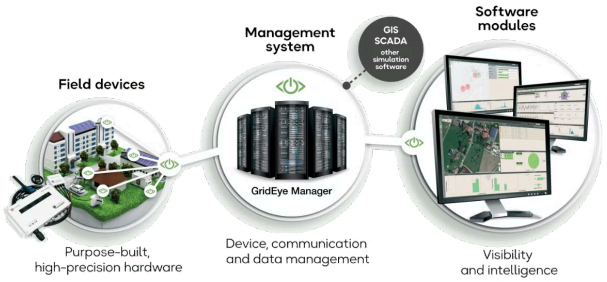
Chaque terminal IoE est composé de **4 couches** fondamentales :

Couche	Rôle	Exemples
🔩 **Matériel**	Microcontrôleur, carte électronique, boîtier, alimentation	ESP32, Arduino, Raspberry Pi
📡 **Capteur / Actionneur**	Composant qui interagit avec le monde physique	Sonde de température, relais
📶 **Connectivité**	Technologie de communication réseau	Wi-Fi, Bluetooth, Zigbee, LoRa
💾 **Firmware / Logiciel**	Programme embarqué qui gère la logique et la communication	Code C/Python, RTOS

INFO

Ces 4 couches sont présentes dans **tout** terminal IoE, du plus simple capteur au dispositif hybride le plus complexe.

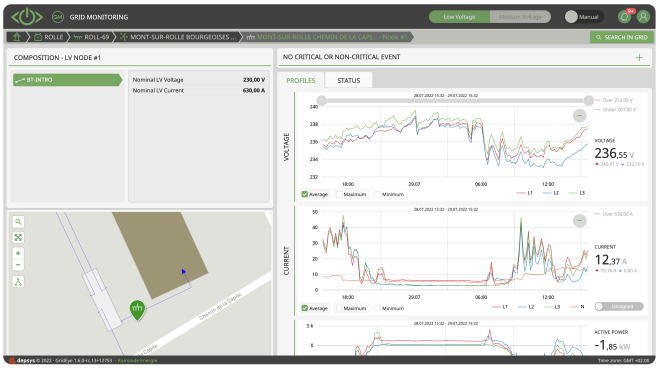
Exemple concret

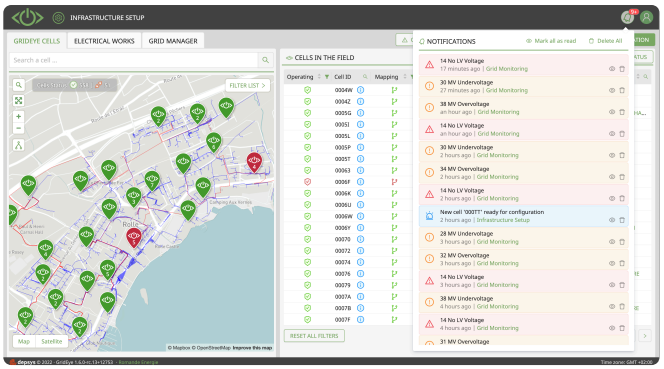


Ceci est un exemple de terminal IoE **hybride** utilisé dans la surveillance intelligente du réseau électrique :

- **Matériel** : boîtier industriel avec microprocesseur
- **Capteur** : mesure de tension, courant et qualité du réseau
- **Connectivité** : réseau Ethernet / 4G vers la plateforme cloud
- **Firmware** : logiciel embarqué de mesure et de transmission

Cet écosystème illustre parfaitement les 4 piliers de l'IoE : le terminal (objet) mesure le réseau (données), transmet à une plateforme (processus) consultée par des techniciens (personnes).





Classification d'un terminal

Pour identifier et classifier un terminal IoE, posez-vous ces questions :

1. **Que fait-il ?** → Capteur, actionneur, passerelle ou hybride ?
2. **Quelles données** produit-il ou reçoit-il ?
3. **Comment communique-t-il ?** → Quelle technologie de connexion ?
4. **Qui l'utilise ?** → Quel utilisateur ou quel processus ?
5. **Quels risques** présente-t-il ? → Données sensibles, accès physique ?

Résumé

Les terminaux IoE se classent en quatre catégories : capteurs, actionneurs, passerelles et objets hybrides. Chaque terminal repose sur une architecture en 4 couches (matériel, capteur/actionneur, connectivité, firmware). La compréhension de cette structure est indispensable pour configurer, intégrer et maintenir un terminal dans un écosystème IoE. L'analyse systématique d'un terminal permet de déterminer son rôle et ses contraintes.

Technologies de communication

Les terminaux IoE utilisent des technologies de connexion variées pour communiquer. Chaque technologie possède ses propres caractéristiques en termes de portée, débit, consommation et coût, ce qui détermine son usage optimal.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Identifier les principales **technologies réseau** utilisées en IoE.
- Comparer les technologies selon leur **portée**, **consommation**, **débit** et **coût**.
- Distinguer une **technologie de connexion** d'un **protocole de communication**.
- Choisir la **technologie adaptée** à un scénario IoE donné.

Technologie vs Protocole

Ces deux notions sont souvent confondues, mais elles sont bien distinctes :

Concept	Définition	Analogie	Exemples
Technologie de connexion	Le support physique ou radio qui transporte les données	Le « tuyau »	Wi-Fi, Ethernet, Bluetooth, LoRa
Protocole de communication	Les règles qui définissent comment les données sont formatées et échangées	Le « langage »	HTTP, MQTT, CoAP

Analogie

Le **Wi-Fi** c'est l'autoroute. **HTTP** ou **MQTT**, c'est le code de la route qui régit comment y circuler.

Vue d'ensemble des technologies

Catégorie	Technologie	Portée	Débit	Consommation
Très courte portée	**NFC**	< 10 cm	Faible	Très faible (passif)
Courte portée	**Bluetooth LE**	10 – 100 m	Faible	Ultra-faible
Courte portée	**Zigbee**	10 – 100 m (mesh)	Faible	Très faible
Portée moyenne	**Wi-Fi**	30 – 300 m	Élevé	Moyenne à élevée
Portée moyenne	**Ethernet**	~100 m (câble)	Très élevé	Moyenne
Longue portée	**LoRa / LoRaWAN**	2 – 15 km	Très faible	Extrêmement faible
Longue portée	**Réseaux mobiles**	National	Variable	Variable

Ethernet

- Connexion **filaire** via câble RJ45
- **Fiable, rapide et stable** (jusqu'à 10 Gbps)
- Pas d'interférences radio → latence très faible
- Consommation plus élevée qu'un capteur sans fil
- Pas de mobilité → **usage fixe uniquement**

Usage typique : caméras IP fixes, passerelles industrielles, NAS, serveurs

Wi-Fi

- Connexion **sans fil**, norme IEEE 802.11
- Portée de ~30 à 300 m selon l'environnement

- **Débit élevé** (plusieurs Gbps avec Wi-Fi 6)
- Consommation plus élevée que BLE ou Zigbee
- Nécessite un **routeur / point d'accès**

Usage typique : TV connectée, aspirateur robot, caméra sans fil, tablette domotique

---

Bluetooth Low Energy (BLE)

- Évolution du Bluetooth, optimisé pour l'**autonomie**
- Portée de ~10 à 100 m
- Débit faible → suffisant pour des mesures ponctuelles
- **Consommation ultra-faible** : piles durables plusieurs années
- Connexion directe à un smartphone **sans infrastructure**

Usage typique : montres connectées, bracelets fitness, capteurs médicaux, balises de localisation

---

Zigbee

- Réseau **maillé (mesh)** : chaque nœud relaie les messages
- Portée de ~10 à 100 m par nœud, extensible par maillage
- **Très faible consommation**
- Jusqu'à **65 000 nœuds** sur un même réseau
- Très populaire en **domotique**

Usage typique : ampoules Philips Hue, capteurs de porte, stores automatiques, prises connectées

> **Réseau mesh**
>
> Si un nœud Zigbee est hors de portée de la passerelle, il peut relayer le message via un nœud intermédiaire — contrairement au BLE.

LoRa / LoRaWAN

- **Long Range** : jusqu'à **15 km** en milieu ouvert
- Consommation **extrêmement faible** (batteries > 10 ans)
- Débit très faible (~300 bps – 50 kbps) → adapté aux petits messages
- Réseau public ou privé (The Things Network, Swisscom...)
- Idéal pour les **zones rurales** ou les bâtiments difficiles d'accès

Usage typique : capteurs agricoles, compteurs d'eau et de gaz, suivi de cheptel, détection d'incendie

---

Réseaux mobiles (4G / 5G / NB-IoT)

- Couverture **nationale et internationale**
- Débit élevé (4G/5G) ou ultra-faible pour IoE (NB-IoT)
- **Infrastructure déjà en place** → aucune installation nécessaire
- Coût d'une carte SIM et d'un abonnement opérateur
- Consommation variable selon la technologie

Usage typique : traceurs GPS, véhicules connectés, terminaux mobiles, distributeurs automatiques

---

NFC (Near Field Communication)

- Portée **extrêmement courte** : moins de 10 cm
- Communication par **champ électromagnétique**
- Les tags NFC sont **passifs** (sans batterie)
- Activation **quasi-instantanée** → pas de jumelage
- Modes : lecture, écriture, paiement, peering

Usage typique : badge d'accès, paiement sans contact, étiquettes de traçabilité, configuration rapide

Limitation

La portée très courte du NFC empêche toute communication continue. Il est optimal pour l'**identification** et les **interactions ponctuelles**.

Choisir la bonne technologie

Le choix dépend du contexte d'utilisation :

Scénario	Technologie adaptée	Raison
Capteur météo à 5 km	**LoRaWAN**	Longue portée, faible consommation
Montre connectée	**Bluetooth LE**	Courte portée, faible énergie, lien smartphone
Caméra IP dans un couloir	**Ethernet / Wi-Fi**	Débit élevé nécessaire pour la vidéo
Capteur de porte domotique	**Zigbee**	Réseau mesh, très faible consommation
Traceur GPS véhicule	**4G / NB-IoT**	Couverture nationale, mobilité
Badge d'accès bâtiment	**NFC**	Interaction ponctuelle, sans batterie

Résumé

Les technologies de communication IoE couvrent un large spectre, de la très courte portée (NFC) à la couverture nationale (4G/5G). Chacune présente des compromis entre portée, débit, consommation et coût. Il n'existe pas de technologie universelle : le choix dépend toujours du contexte (distance, énergie, volume de données, mobilité). Cette distinction entre technologie de connexion et protocole de communication est fondamentale pour concevoir une solution IoE adaptée.

Protocoles IoE

Les protocoles définissent les règles d'échange de données entre les terminaux et les plateformes. En IoE, trois protocoles dominent : HTTP/REST pour le web, MQTT pour le temps réel léger, et CoAP pour les microcontrôleurs contraints.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Expliquer le fonctionnement de **HTTP/REST** (requête/réponse).
- Décrire le modèle **publication/souscription** de MQTT.
- Comprendre le rôle de **CoAP** pour les réseaux contraints.
- Choisir le **protocole adapté** selon le contexte IoE.

HTTP / REST → Requête / Réponse

HTTP est le protocole standard du **Web**. En IoE, il permet à un terminal d'interagir avec une **API REST**.

Fonctionnement

Le terminal (client) envoie une **requête** au serveur, qui retourne une **réponse** :

Méthode	Action	Exemple
`GET`	Lire des données	`GET /capteurs/temperature`
`POST`	Envoyer des données	`POST /donnees`
`PUT`	Modifier une configuration	`PUT /config`
`DELETE`	Supprimer une ressource	`DELETE /capteurs/42`

Les données sont généralement échangées au format **JSON**.

Avantages

- Standard et **universel**
- Facile à **tester et déboguer**
- Compatible avec tous les langages et services

Limites en IoE

- Plus **lourd** que MQTT (en-têtes HTTP importants)
- Le terminal doit **toujours initier** la connexion
- Consomme plus d'**énergie** et de **mémoire**

Contexte d'usage : terminaux puissants, APIs web, services cloud

---

MQTT → Publication / Souscription

MQTT (Message Queuing Telemetry Transport) est un protocole **léger** conçu spécialement pour les objets connectés à faibles ressources.

Fonctionnement

MQTT repose sur un modèle **publish/subscribe** avec un intermédiaire central appelé **broker** :

1. Un **publisher** (ex. capteur) publie un message sur un **topic**
2. Le **broker** reçoit le message et le redistribue
3. Les **subscribers** (ex. tableau de bord) abonnés à ce topic reçoivent le message

Exemple de topics MQTT

```
maison/salon/temperature     → 22.5
maison/cuisine/humidite      → 65
batiment/salle-serveur/alerte → true
```

Avantages

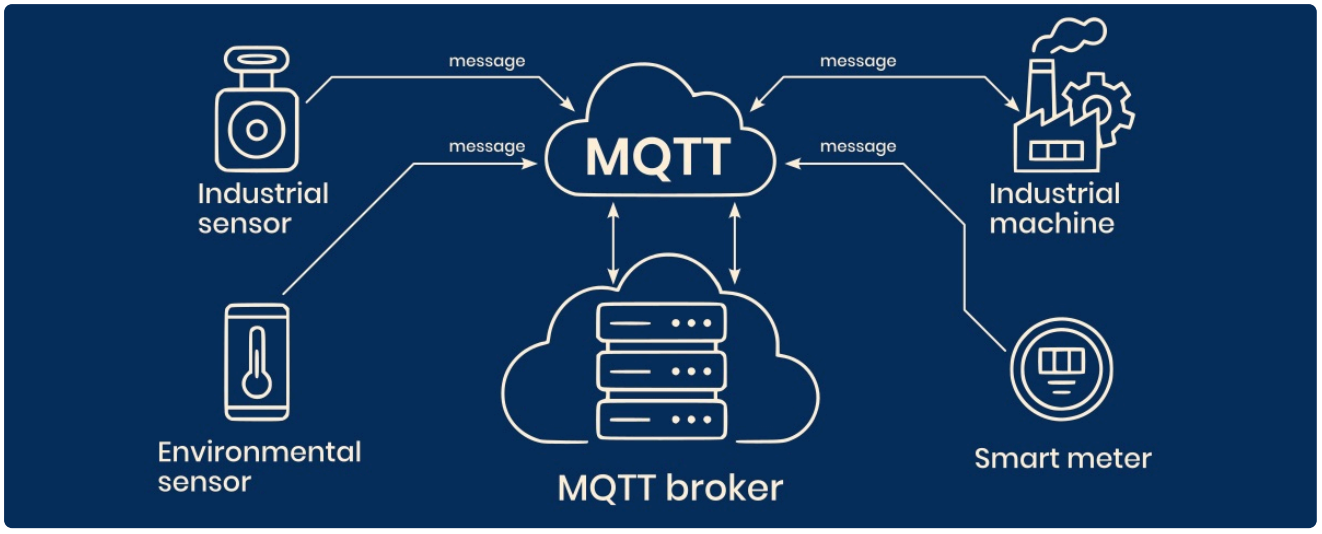
- **Ultra-léger** : fonctionne sur des terminaux avec très peu de ressources

- Communication en **temps réel**
- Le terminal n'a pas besoin d'être interrogé → le message est **poussé**
- Gestion de la **qualité de service** (QoS 0, 1, 2)
- Supporte des **milliers d'appareils** simultanément

Limites

- Nécessite un **broker** central (ex. Mosquitto, HiveMQ)
- Moins adapté aux interactions de type requête/réponse classiques

Contexte d'usage : capteurs IoE, domotique, temps réel, milliers d'appareils



CoAP → HTTP ultra-léger

CoAP (Constrained Application Protocol) est un protocole conçu pour les **microcontrôleurs** et les réseaux très contraints.

Fonctionnement

- Modèle **requête/réponse** similaire à HTTP
- Fonctionne sur **UDP** (plus léger que TCP)
- Supporte le mode **observe** (notification automatique en cas de changement)

Comparaison avec HTTP

Critère	HTTP	CoAP
Transport	TCP	UDP
En-têtes	Lourds	Très compacts
Ressources requises	Élevées	Très faibles
Mode push	Non natif	**Observe** intégré

Contexte d'usage : capteurs embarqués, réseaux LoRa, Zigbee, NB-IoT

Comparaison des protocoles

Critère	HTTP/REST	MQTT	CoAP
Modèle	Requête/réponse	Publish/subscribe	Requête/réponse
Transport	TCP	TCP	UDP
Poids	Lourd	Léger	Ultra-léger
Temps réel	Non	Oui	Partiel (observe)
Usage principal	APIs web, cloud	IoE temps réel	Microcontrôleurs

En résumé

HTTP = web standard · **MQTT** = publish/subscribe temps réel · **CoAP** = HTTP ultra-léger pour microcontrôleurs

---

Résumé

Les protocoles IoE définissent comment les données circulent entre terminaux et plateformes. HTTP/REST convient aux interactions web classiques, MQTT excelle dans les scénarios temps réel avec de nombreux appareils grâce à son modèle publish/subscribe, et CoAP offre une alternative ultra-légère pour les terminaux très

contraints. Le choix du protocole dépend des ressources du terminal, du besoin en temps réel et de l'infrastructure disponible.

Adressage et réseau

Pour fonctionner dans un écosystème connecté, chaque terminal IoE doit être identifiable sur le réseau. La compréhension de l'adressage IP, MAC et des mécanismes comme le DHCP est indispensable pour intégrer correctement un terminal — mais tous les réseaux IoT n'utilisent pas TCP/IP directement au niveau du terminal. Certains, comme LoRaWAN ou Zigbee, disposent de leur propre pile protocolaire et rejoignent le réseau IP uniquement via une passerelle.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Expliquer les notions d'**adresse IP**, d'**adresse MAC** et de **sous-réseau**.
- Distinguer l'adressage **DHCP** de l'adressage **statique** et justifier le choix.
- Situer un terminal IoE dans un **schéma réseau** existant.
- Préparer l'**intégration réseau** d'un terminal IoE.
- Reconnaître les **piles protocolaires non-IP** (Zigbee, LoRaWAN) et comprendre leur mode de connexion au réseau IP.

Rappels réseau essentiels

Adresse MAC

L'adresse MAC (Media Access Control) est un identifiant **unique et physique** attribué à chaque interface réseau lors de sa fabrication.

- Format : `AA:BB:CC:DD:EE:FF` (6 octets en hexadécimal)
- **Ne change pas** (sauf cas de spoofing)
- Fonctionne au niveau de la **couche liaison** (couche 2)

Adresse IP

L'adresse IP est un identifiant **logique** attribué à un appareil sur un réseau.

- **IPv4** : `192.168.1.100` (4 octets, le plus courant en réseau local)
- **IPv6** : `fe80::1` (16 octets, adressage étendu)
- Fonctionne au niveau de la **couche réseau** (couche 3)
- Peut être attribuée dynamiquement (DHCP) ou manuellement (statique)

Différence MAC / IP

Critère	Adresse MAC	Adresse IP
Type	Physique (matériel)	Logique (réseau)
Attribution	Usine (fixe)	Réseau (configurable)
Portée	Réseau local uniquement	Réseau local ou étendu
Couche OSI	Couche 2 (liaison)	Couche 3 (réseau)

Sous-réseau

Le sous-réseau permet de **segmenter** un réseau en groupes logiques. Le masque de sous-réseau (ex. `255.255.255.0` ou `/24`) détermine quelles adresses font partie du même réseau local.

Nom d'hôte

Chaque terminal peut recevoir un **nom d'hôte** (hostname) pour faciliter son identification. Par exemple : `capteur-salle-201`, `gateway-etage-2`.

DHCP vs IP statique

DHCP (Dynamic Host Configuration Protocol)

Le serveur DHCP attribue **automatiquement** une adresse IP aux appareils qui se connectent au réseau.

Avantages	Inconvénients
Configuration automatique	L'adresse peut changer au renouvellement

Avantages	Inconvénients
Pas de gestion manuelle	Moins prévisible pour les accès distants
Adapté aux appareils « de passage »	Nécessite un serveur DHCP fonctionnel

IP statique

L'administrateur attribue **manuellement** une adresse fixe au terminal.

Avantages	Inconvénients
Adresse toujours identique	Configuration manuelle nécessaire
Idéal pour les accès distants et les règles pare-feu	Risque de conflit si mal gérée
Plus facile à identifier et surveiller	Ne s'adapte pas automatiquement

Réservation DHCP

Compromis entre les deux : le serveur DHCP **réserve** toujours la même adresse IP pour une adresse MAC donnée. Le terminal utilise le DHCP normalement, mais reçoit systématiquement la même IP.

> **Recommandation en IoE**
>
> Les terminaux IoE critiques (passerelles, caméras, serveurs) bénéficient d'une **IP statique ou d'une réservation DHCP**. Les capteurs temporaires peuvent utiliser le DHCP standard.

Intégration réseau d'un terminal

Éléments nécessaires

Pour intégrer un terminal IoE dans un réseau existant, il faut déterminer :

1. **Adresse IP** (statique ou DHCP)
2. **Masque de sous-réseau**
3. **Passerelle par défaut** (gateway réseau)
4. **Serveur DNS** (si résolution de noms nécessaire)

5. **Identifiants Wi-Fi** (SSID et mot de passe, si sans fil)

Place du terminal dans le réseau

Un schéma réseau typique en IoE inclut :

Élément	Rôle
Routeur / Box	Accès Internet et passerelle par défaut
Switch	Connexion filaire des terminaux
Point d'accès Wi-Fi	Connexion sans fil
Gateway IoE	Pont entre les terminaux (Zigbee, LoRa...) et le réseau IP
Terminaux IoE	Capteurs, actionneurs, objets hybrides

Découverte d'équipements

Sur un réseau local, plusieurs méthodes permettent de **découvrir** les terminaux connectés :

- Scan réseau (ex. `nmap`, `arp -a`)
- Consultation de la table DHCP du routeur
- Protocoles de découverte (mDNS, UPnP)

Piles protocolaires alternatives (non-IP)

Tous les terminaux IoE ne parlent pas IP

Les sections précédentes décrivent l'adressage TCP/IP classique. Cependant, de nombreux terminaux IoE — notamment ceux à très faible consommation ou longue portée — ne disposent pas d'une pile IP complète. Ils utilisent des **piles protocolaires propres** et rejoignent le réseau IP uniquement via une **passerelle**.

Pourquoi certains terminaux évitent TCP/IP ?

Contrainte	Problème avec TCP/IP
Très faible mémoire (quelques kB)	Une pile IP complète est trop lourde
Batterie longue durée (années)	L'overhead TCP/IP consomme trop d'énergie
Longue portée, très faible débit	TCP/IP est surdimensionné pour quelques octets
Réseau maillé à grande échelle	IP nécessite une infrastructure plus complexe

Exemples de piles non-IP

Zigbee (IEEE 802.15.4)

Zigbee repose sur la norme radio **IEEE 802.15.4** et définit ses propres couches réseau et application.

Couche	Protocole Zigbee
Application	Zigbee Cluster Library (ZCL)
Réseau	Zigbee Network Layer
MAC / PHY	IEEE 802.15.4

- **Adressage** : adresse EUI-64 (unique, similaire au MAC) + adresse 16 bits dans le réseau
- **Topologie** : maillage (mesh) — les terminaux relaient les messages entre eux
- **Passerelle** : le **coordinateur Zigbee** fait le pont vers le réseau IP

LoRaWAN

LoRaWAN utilise la modulation radio **LoRa** et définit sa propre couche MAC et réseau.

Couche	Protocole LoRaWAN
Application	Application Server (cloud)
Réseau / MAC	LoRaWAN
PHY	LoRa (modulation radio)

- **Adressage** : chaque terminal a un **DevEUI** (64 bits, unique, attribué en usine) et un **DevAddr** (32 bits, attribué lors de la jonction au réseau par OTAA)
- **Topologie** : étoile — les terminaux transmettent à des **gateways LoRaWAN**, qui relaient vers un **Network Server** via IP
- **Portée** : jusqu'à plusieurs kilomètres en zone dégagée

La passerelle comme traducteur de pile

Dans ces architectures, la passerelle ne se contente pas de router des paquets IP : elle effectue une **traduction complète de pile protocolaire**.

```
Capteur Zigbee  ──Zigbee──▶ Coordinateur/Gateway ──MQTT/IP──▶ Broker
Capteur LoRaWAN ──LoRa──▶ Gateway LoRaWAN       ──UDP/IP──▶ Network Serve
Capteur Wi-Fi   ──MQTT──▶ Switch / Routeur       ──IP──────▶ Broker
```

Le terminal **n'a pas d'adresse IP**, mais est identifié par son adresse propre à sa pile. C'est la passerelle qui l'expose comme ressource IP vers le reste du système.

Comparaison de l'adressage selon la pile

Technologie	Identifiant unique (≈ MAC)	Identifiant réseau (≈ IP)	Attribué par
IPv4	Adresse MAC	Adresse IP	Usine / DHCP ou statique
Zigbee	EUI-64	Adresse réseau 16 bits	Coordinateur Zigbee
LoRaWAN	DevEUI (64 bits)	DevAddr (32 bits)	Network Server (OTAA)

Logique universelle d'adressage

Dans toutes ces piles, on retrouve la même logique : un **identifiant unique physique** (similaire au MAC) et un **identifiant de réseau** attribué dynamiquement (similaire à l'IP via DHCP). La logique est universelle — seule l'implémentation change.

Résumé

L'adressage réseau est un prérequis fondamental pour l'intégration des terminaux IoE. Dans les réseaux TCP/IP, chaque terminal est identifié par une adresse MAC (physique) et une adresse IP (logique), attribuée par DHCP ou manuellement. Cependant, de nombreux protocoles IoE « comme Zigbee ou LoRaWAN » disposent de leur propre pile protocolaire et de leur propre mécanisme d'adressage : ils rejoignent le réseau IP uniquement via une passerelle qui effectue la traduction de pile. Comprendre ces deux modes d'adressage et la topologie réseau associée est une compétence essentielle pour tout technicien IoE.

Les bases de l'électricité

Avant de connecter un capteur ou un microcontrôleur, il faut comprendre les bases de l'électricité. Ces notions sont indispensables pour réaliser des montages corrects, éviter de détruire du matériel et comprendre les schémas de connexion.

Objectifs

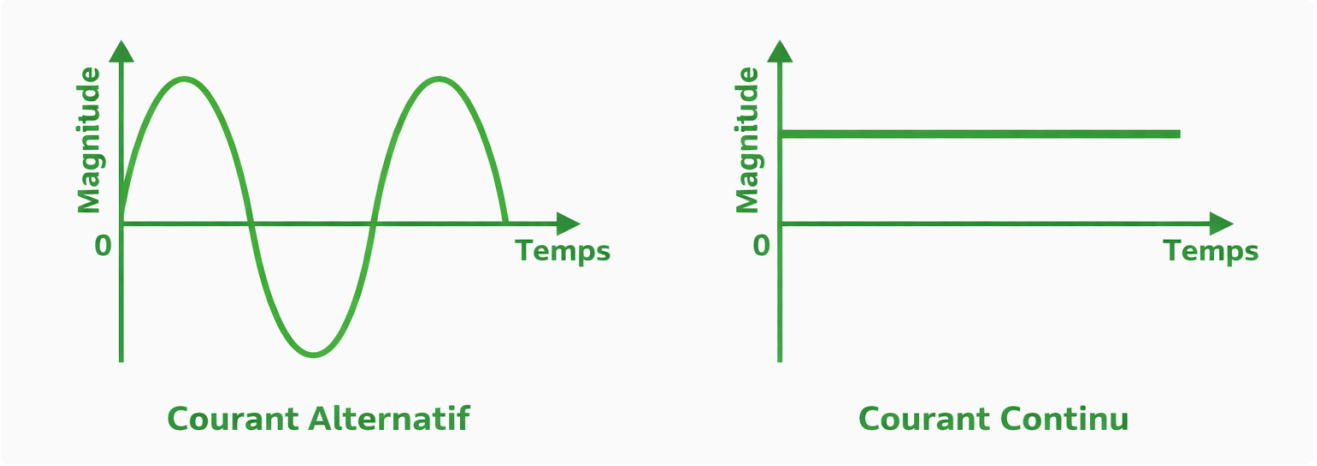
À la fin de ce chapitre, vous serez capable de :

- Distinguer le **courant continu (DC)** du **courant alternatif (AC)**.
- Expliquer les notions de **tension**, **intensité**, **résistance** et **puissance**.
- Identifier le rôle du **GND** (masse) dans un circuit.
- Lire et réaliser un **schéma de connexion** entre composants (Arduino, xBee, LCD).
- Reconnaître les interfaces **UART** (TX/RX) et **I2C** (SDA/SCL).

Courant continu vs alternatif

Il existe deux façons de faire circuler l'électricité :

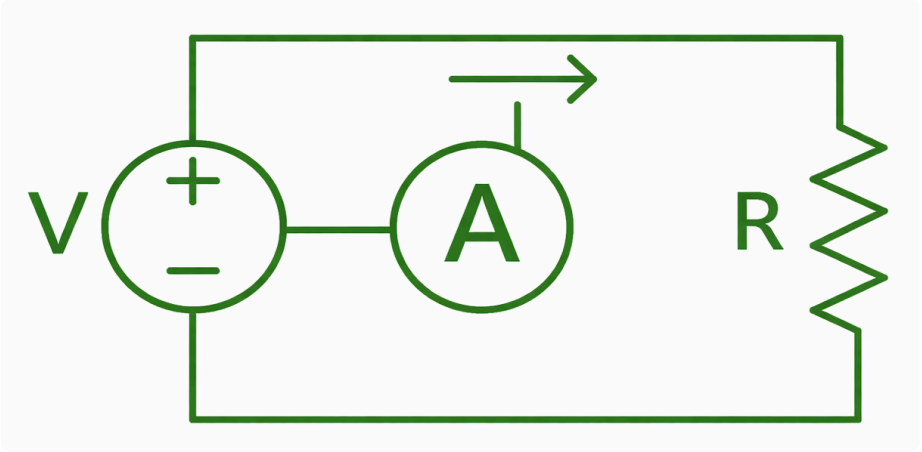
Type	Abréviation	Caractéristique	Exemples
Courant continu	DC (*Direct Current*)	Le courant circule toujours dans le même sens	Piles, batteries, USB, Arduino
Courant alternatif	AC (*Alternating Current*)	Le courant change de sens ~50 fois par seconde	Prises murales (230 V en Europe)



Le courant alternatif possède une **fréquence**, exprimée en **hertz (Hz)**.
En Europe, cette fréquence est généralement de **50 Hz**, ce qui signifie que le courant change de sens **50 fois par seconde**.

> ⚠️ **Attention**
>
> En électronique embarquée (Arduino, capteurs, modules IoT), on travaille **exclusivement en courant continu** à basse tension (3.3 V ou 5 V). Le 230 V des prises murales est réservé à des professionnels habilités.

Les grandeurs électriques fondamentales



La tension « Volt (V) »

La **tension** (aussi appelée *voltage* ou *différence de potentiel*) est la « force » qui pousse les électrons à circuler dans un circuit.

> **Analogie**

Imaginez un tuyau d'eau : la tension, c'est la **pression** de l'eau. Plus la pression est élevée, plus l'eau a de force pour avancer.

Tension	Utilisation typique
1.5 V	Pile AA/AAA
3.3 V	Logique des modules ESP, xBee
5 V	Arduino Uno, USB
9 – 12 V	Alimentation externe Arduino

Le courant « Ampère (A) »

L'**intensité du courant** représente la quantité d'électricité qui circule par seconde dans un fil.

Analogie

Avec notre tuyau d'eau, le courant, c'est le **débit** → combien de litres passent par seconde.

En électronique embarquée, les courants sont souvent faibles. On utilise le **milliampère (mA)** :

Composant	Consommation typique
LED	~20 mA
Capteur de température	~1 – 5 mA
Module xBee (émission)	~250 mA
Arduino Uno	~50 mA
Écran LCD I2C	~20 – 60 mA

La résistance « Ohm (Ω) »

La **résistance** s'oppose au passage du courant. Elle permet de limiter l'intensité afin de protéger les composants.

La loi d'Ohm relie les trois grandeurs :

$$U = R \times I$$

Symbole	Grandeur	Unité
U	Tension	Volt (V)
R	Résistance	Ohm (Ω)
I	Intensité du courant	Ampère (A)

Exemple

Une LED a besoin de ~2 V et supporte ~20 mA. Alimentée en 5 V :

$$R = \frac{U}{I} = \frac{5 - 2}{0.02} = 150\ \Omega$$

Il faut placer une résistance de **150 Ω** en série avec la LED.

La puissance « Watt (W) »

La **puissance** indique la quantité d'énergie consommée par seconde.

La puissance électrique se calcule à partir de la tension et du courant :

$$P = U \times I$$

Symbole	Grandeur	Unité
P	Puissance	Watt (W)
U	Tension	Volt (V)
I	Courant	Ampère (A)

Exemples :

	Calcul	Puissance
Arduino Uno en veille	5 V × 0.05 A	**0.25 W**
xBee en émission	3.3 V × 0.25 A	**0.83 W**

	Calcul	Puissance
Chargeur de téléphone	5 V × 2 A	**10 W**

La masse « GND »

Le **GND** (*Ground*, en français : *masse* ou *terre*) est le **point de référence 0 V** de tout circuit électronique.

> C'est le fil de retour commun à tous les composants. Sans GND partagé, aucun courant ne peut circuler.

Règle fondamentale

Tous les composants d'un même montage **doivent partager le même GND**. Si deux modules ont des GND non reliés, ils ne peuvent pas communiquer correctement, même si leurs tensions sont identiques.

```
Module A ——— GND ———┐
Module B ——— GND ———┼—— GND commun (0 V)
Module C ——— GND ———┘
```

Niveaux logiques : 5 V vs 3.3 V

Les microcontrôleurs utilisent des niveaux de tension pour représenter les valeurs binaires :

Niveau	Signification
0 V → GND	Bit `0` (LOW)
3.3 V ou 5 V	Bit `1` (HIGH)

Incompatibilité 5 V ↔ 3.3 V

L'**Arduino Uno** fonctionne en **5 V**. Le module **xBee** fonctionne en **3.3 V logique**. Envoyer 5 V sur une broche 3.3 V peut **endommager définitivement le module**.

ℹ️ Dans notre montage, le shield SparkFun XBee Explorer Regulated gère l'alimentation 3.3 V et la conversion de niveaux logiques. Sans cette carte d'interface, il est important de passer l'Arduino Uno en mode 3.3V.

Interfaces de communication

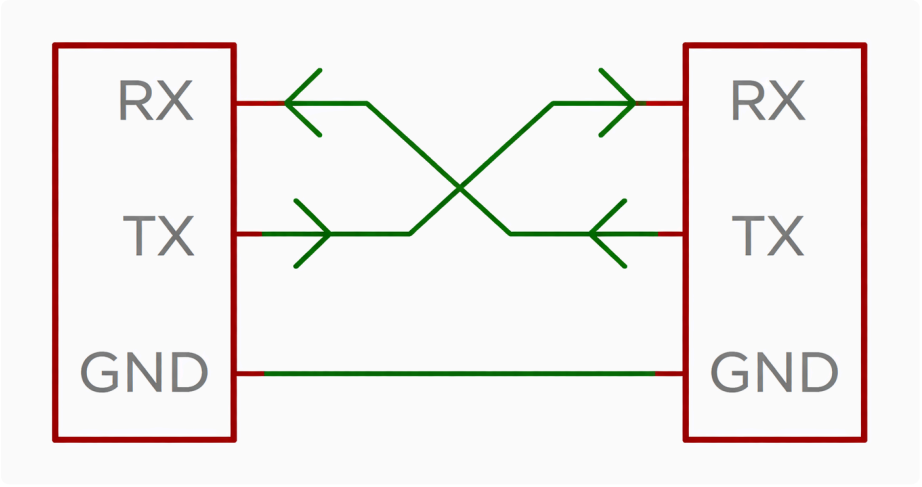
UART « Communication série (TX / RX) »

UART (*Universal Asynchronous Receiver-Transmitter*) est une interface de communication série point à point.

- **TX** (*Transmit*) → broche d'**envoi** de données
- **RX** (*Receive*) → broche de **réception** de données

Règle de câblage croisé

Le TX d'un module se connecte au **RX** de l'autre, et vice versa :



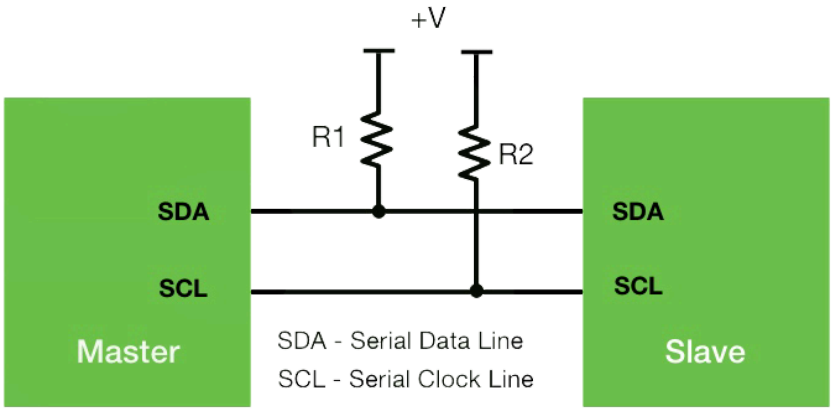
I2C « Bus à deux fils (SDA / SCL) »

I²C (*Inter-Integrated Circuit*) est un protocole permettant de connecter **plusieurs composants** sur seulement **2 fils** partagés.

- **SDA** (*Serial Data*) → fil de **données**
- **SCL** (*Serial Clock*) → fil d'**horloge** (synchronisation)

Un circuit I2C fonctionne en mode **maître/esclave** :

```
Arduino (maître)
  SDA ──────────── SDA ── Écran LCD
  SCL ──────────── SCL ── Écran LCD
  GND ──────────── GND ── Écran LCD
  +5V ──────────── VCC ── Écran LCD
```



Avantage I2C

On peut brancher jusqu'à **127 esclaves** différents (écran, capteur, EEPROM...) sur les mêmes 2 fils, en les distinguant par leur **adresse unique (0x27, 0x3C...)**.

Résumé

Notion	Symbole	Unité	Rôle
Tension	U	Volt (V)	Force qui pousse le courant
Intensité	I	Ampère (A)	Quantité de courant qui circule
Résistance	R	Ohm (Ω)	S'oppose au passage du courant
Puissance	P	Watt (W)	Énergie consommée par seconde
Masse	GND	—	Référence 0 V commune

Les interfaces clés du projet :

- **UART (TX/RX)** → communication série Arduino ↔ xBee (câblage croisé)
- **I2C (SDA/SCL)** → communication Arduino ↔ LCD (2 fils partagés)

- **GND commun** → indispensable entre tous les composants

Configuration des terminaux

La configuration d'un terminal IoE doit suivre une procédure structurée et documentée. Une approche méthodique réduit les erreurs, facilite la maintenance et garantit la cohérence de l'écosystème connecté. Selon la technologie utilisée, les paramètres à configurer diffèrent sensiblement : un terminal Wi-Fi/Ethernet nécessite une configuration TCP/IP classique, tandis qu'un terminal Zigbee ou LoRaWAN exige la saisie de clés cryptographiques et un processus d'association à son réseau propre.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Identifier les principaux **paramètres de configuration** d'un terminal IoE selon sa technologie.
- Distinguer la configuration d'un terminal **TCP/IP** de celle d'un terminal **non-IP** (Zigbee, LoRaWAN).
- Appliquer une **procédure structurée** de mise en service.
- Utiliser une **checklist** de configuration avant la mise en production.
- Reconnaître les **erreurs fréquentes** de configuration.

Paramètres de configuration

Deux grandes familles de terminaux

Les paramètres à configurer dépendent de la pile protocolaire du terminal. Un terminal **TCP/IP** (Wi-Fi, Ethernet) se configure comme un équipement réseau classique. Un terminal **non-IP** (Zigbee, LoRaWAN) utilise ses propres identifiants et mécanismes d'association.

Paramètres réseau (terminaux TCP/IP)

Paramètre	Description	Exemple
Réseau Wi-Fi / Ethernet	Connexion au réseau local	SSID: `IoE-Lab`, câble RJ45
Adresse IP	Identification sur le réseau	`192.168.1.50` (statique) ou DHCP
Masque de sous-réseau	Délimitation du réseau local	`255.255.255.0`
Passerelle par défaut	Accès au réseau étendu / Internet	`192.168.1.1`
DNS	Résolution de noms	`8.8.8.8`

Paramètres d'association (terminaux non-IP)

Ces terminaux ne configurent pas d'adresse IP : ils s'associent à leur réseau via des **identifiants cryptographiques** ou un **processus d'inclusion**.

Zigbee

Paramètre	Description	Exemple
Canal radio	Fréquence utilisée (2.4 GHz)	Canal 11–26
PAN ID	Identifiant du réseau Zigbee	`0x1A2B`
Processus d'appairage	Activer le mode « permit join » sur le coordinateur	Bouton ou commande logicielle

> Le terminal obtient alors automatiquement son adresse réseau 16 bits auprès du coordinateur.

LoRaWAN

Paramètre	Description
DevEUI	Identifiant unique du terminal (gravé en usine, lecture seule)

Paramètre	Description
AppEUI / JoinEUI	Identifiant de l'application sur le Network Server
AppKey	Clé AES-128 partagée — permet la jonction sécurisée (OTAA)
Plan de fréquences	Région radio (ex. `EU868` pour l'Europe)

> En mode OTAA (recommandé), le terminal négocie ses clés de session à chaque jonction. En mode ABP (statique), `DevAddr`, `NwkSKey` et `AppSKey` sont saisis directement.

Sécurité LoRaWAN

L'`AppKey` est une clé cryptographique sensible. Elle ne doit jamais être partagée ou stockée en clair dans une documentation publique.

Paramètres d'identification

Paramètre	Description
Nom du terminal	Identifiant lisible (ex. `capteur-temp-salle-201`)
Identifiant / login	Compte d'administration du terminal
Mot de passe	Accès sécurisé au terminal

Paramètres fonctionnels

Paramètre	Description	Exemple
Fréquence d'envoi	Intervalle entre chaque transmission	Toutes les 60 secondes
Serveur / broker cible	Plateforme de réception des données	`mqtt.exemple.ch:1883`
Fuseau horaire	Horodatage correct des données	`Europe/Zurich`
Seuils d'alerte	Valeurs déclenchant une action	Température > 30°C

Procédure structurée de configuration

Étapes recommandées

1. **Identifier le terminal** : type, modèle, version firmware, pile protocolaire (TCP/IP ou non-IP)
2. **Préparer l'infrastructure** :
   - TCP/IP : vérifier la disponibilité d'une adresse IP, tester la connectivité réseau
   - Non-IP : s'assurer que le coordinateur / gateway est opérationnel et accessible
3. **Configurer la connexion** :
   - TCP/IP : Wi-Fi ou Ethernet, IP, masque, passerelle, DNS
   - Zigbee : saisir le canal / PAN ID et déclencher le mode « permit join »
   - LoRaWAN : saisir DevEUI, AppEUI, AppKey et le plan de fréquences
4. **Changer les identifiants par défaut** (si interface web ou SSH disponible) : nouveau mot de passe fort
5. **Paramétrer la destination** : serveur, broker MQTT, API cible
6. **Régler les paramètres fonctionnels** : fréquence, seuils, fuseau horaire
7. **Nommer le terminal** : nom logique et cohérent
8. **Tester la communication** : vérifier que les données arrivent à destination
9. **Documenter** : noter la configuration dans une fiche technique

Importance de la standardisation

Lorsque plusieurs terminaux sont déployés, il est essentiel de :

- Utiliser une **convention de nommage** cohérente
- Appliquer les **mêmes paramètres réseau** de base
- Maintenir un **inventaire** à jour des terminaux et de leur configuration

---

Checklist de mise en service

Avant de déclarer un terminal opérationnel, vérifier :

Général (tous les terminaux)

- [ ] Type, modèle et firmware identifiés et documentés
- [ ] Nom du terminal défini et cohérent avec la convention de nommage
- [ ] Données reçues et cohérentes côté plateforme
- [ ] Fuseau horaire correct
- [ ] Firmware à jour
- [ ] Fiche de configuration complétée

Terminaux TCP/IP (Wi-Fi / Ethernet)

- [ ] Connexion réseau fonctionnelle (ping, accès web)
- [ ] Identifiants par défaut modifiés — mot de passe fort configuré
- [ ] Adresse IP, masque, passerelle, DNS vérifiés
- [ ] Serveur / broker cible correctement renseigné

Terminaux Zigbee

- [ ] Coordinateur opérationnel et joignable
- [ ] Canal et PAN ID vérifiés
- [ ] Terminal bien appairé (visible dans la liste des nœuds du coordinateur)

Terminaux LoRaWAN

- [ ] DevEUI, AppEUI et AppKey saisis correctement
- [ ] Plan de fréquences conforme à la région (ex. EU868)
- [ ] Jonction OTAA réussie (DevAddr attribué par le Network Server)
- [ ] AppKey non exposée publiquement

Erreurs fréquentes

Erreur	Conséquence
Mauvais réseau Wi-Fi	Terminal isolé, aucune donnée transmise
Mot de passe par défaut conservé	Faille de sécurité critique
Mauvais serveur / broker cible	Données perdues ou envoyées au mauvais endroit

Erreur	Conséquence
Fuseau horaire incorrect	Horodatage erroné, analyse faussée
Documentation absente	Maintenance impossible, configuration perdue
Conflit d'adresse IP	Dysfonctionnement réseau pour plusieurs appareils
AppKey LoRaWAN incorrecte	Jonction OTAA échoue, terminal muet sur le réseau
Plan de fréquences LoRaWAN inadapté	Terminal et gateway sur des canaux différents, pas de réception
Appairage Zigbee sans mode « permit join » actif	Terminal non reconnu, association impossible

---

Résumé

La configuration d'un terminal IoE couvre les paramètres réseau, d'identification et fonctionnels. Les paramètres varient selon la pile protocolaire : un terminal TCP/IP (Wi-Fi, Ethernet) requiert adresse IP, masque et passerelle, tandis qu'un terminal non-IP (Zigbee, LoRaWAN) nécessite des clés cryptographiques ou un processus d'association propre à sa technologie. Dans tous les cas, une procédure structurée, une checklist et une documentation systématique sont essentielles pour garantir la cohérence du déploiement et faciliter la maintenance.

Sécurité et maintenance

La sécurité des terminaux IoE est un enjeu critique : objets souvent exposés, données sensibles collectées et mises à jour parfois négligées. Ce chapitre couvre les mesures de sécurité essentielles et les bonnes pratiques de maintenance.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Comprendre l'importance des **mises à jour** (firmware, logiciel).
- Identifier les principales **mesures de sécurité** applicables aux terminaux IoE.
- Reconnaître les **données sensibles** collectées par les terminaux.
- Appliquer les **bonnes pratiques** de sécurité et de maintenance.

Firmware, système et application

Définitions

Composant	Description	Exemple
Firmware	Programme embarqué bas niveau, contrôle le matériel	Code sur un ESP32, BIOS d'une gateway
Système d'exploitation	Logiciel de base gérant les ressources matérielles	Linux embarqué, RTOS, Raspberry Pi OS
Application embarquée	Programme applicatif fonctionnant sur le terminal	Script Python de mesure, agent MQTT

Pourquoi mettre à jour ?

Raison	Explication
Sécurité	Correction de vulnérabilités connues

Raison	Explication
Stabilité	Résolution de bugs et amélioration des performances
Compatibilité	Maintien de la compatibilité avec les plateformes et protocoles

WARNING

Un firmware obsolète est l'une des failles de sécurité les plus fréquentes en IoE. Les mises à jour doivent être planifiées et documentées.

Risques de sécurité

Faiblesses courantes

Risque	Description
Firmware obsolète	Vulnérabilités non corrigées, exploitables à distance
Mots de passe par défaut	Accès non autorisé trivial (ex. admin/admin)
Absence de chiffrement	Données transmises en clair, interceptables
Terminal exposé inutilement	Ports ouverts, accès depuis Internet sans nécessité
Absence de segmentation	Un terminal compromis peut accéder à tout le réseau

Données sensibles collectées

Les terminaux IoE collectent souvent des données à caractère personnel ou sensible :

Type de données	Exemples	Sensibilité
Localisation	GPS, triangulation Wi-Fi	Élevée
Image / vidéo	Caméras de surveillance	Très élevée
Audio / voix	Assistants vocaux, interphones	Très élevée
Présence	Détecteurs de mouvement	Moyenne à élevée

Type de données	Exemples	Sensibilité
Habitudes	Horaires, fréquences d'utilisation	Moyenne
Données médicales	Rythme cardiaque, activité physique	Très élevée

---

Mesures de sécurité

Bonnes pratiques essentielles

1. **Changer les identifiants par défaut** dès la première configuration
2. **Utiliser des mots de passe forts** (longueur, complexité, unicité)
3. **Mettre à jour le firmware** régulièrement
4. **Chiffrer les communications** (TLS/SSL, MQTT over TLS)
5. **Segmenter le réseau** (VLAN dédié aux terminaux IoE)
6. **Limiter les accès** (pare-feu, listes blanches, ACL)
7. **Désactiver les services inutiles** (ports, protocoles non utilisés)
8. **Surveiller** les terminaux (logs, alertes en cas d'anomalie)

Segmentation réseau

Placer les terminaux IoE dans un **VLAN dédié** permet de :

- Isoler le trafic IoE du réseau bureautique
- Limiter la propagation en cas de compromission
- Appliquer des règles de pare-feu spécifiques

Communications chiffrées

Protocole	Version sécurisée
HTTP	**HTTPS** (HTTP over TLS)
MQTT	**MQTTS** (MQTT over TLS, port 8883)
CoAP	**CoAPS** (CoAP over DTLS)

Maintenance

Plan de maintenance

Un terminal IoE en production nécessite un suivi régulier :

Action	Fréquence recommandée
Vérification de la connectivité	Quotidienne (automatisée)
Contrôle de la cohérence des données	Hebdomadaire
Mise à jour firmware	Selon disponibilité (planifiée)
Revue des accès et mots de passe	Trimestrielle
Remplacement des batteries	Selon durée de vie estimée
Documentation à jour	À chaque modification

Résumé

La sécurité IoE repose sur des mesures simples mais indispensables : mots de passe forts, mises à jour régulières, chiffrement des communications et segmentation réseau. Les terminaux collectent souvent des données sensibles qui exigent une attention particulière à la protection de la vie privée. La maintenance régulière et documentée garantit la fiabilité et la sécurité du système dans le temps. Ignorer ces pratiques expose l'infrastructure à des risques importants.

Cloud & Edge Computing

Les terminaux IoT génèrent en permanence des quantités colossales de données. Où et comment les traiter est un enjeu clé. Le cloud et l'edge computing offrent deux approches complémentaires pour y répondre.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Expliquer la différence entre **cloud computing** et **edge computing**.
- Identifier les **avantages et limites** de chaque approche pour l'IoT.
- Décrire l'**architecture en 3 couches** d'un système IoT moderne.
- Associer une approche (cloud, edge ou hybride) à un **cas d'usage** concret.

Le défi : des données en permanence

Un système IoT peut générer des **millions de messages par seconde**. Imaginez :

- Des milliers de capteurs de température dans une usine
- Des caméras de surveillance dans une ville entière
- Des capteurs embarqués sur une flotte de véhicules

Problème : Faut-il vraiment tout envoyer vers un serveur distant avant d'agir ?

Exemple concret

Une voiture autonome génère jusqu'à **40 Go de données par heure**. Il est impossible d'envoyer tout cela vers un cloud distant, car la décision de freiner doit être prise en **quelques millisecondes**, directement à bord.

☁️ Cloud Computing

Principe

Le cloud computing consiste à **stocker et traiter les données sur des serveurs distants**, accessibles via Internet. Les données « montent » vers des datacentres, y sont traitées, et les résultats « redescendent » vers les utilisateurs ou les terminaux.

📡 Capteurs → 🌐 Internet → ☁️ Serveurs cloud → ✅ Résultats

Caractéristiques

Aspect	Description
Localisation	Serveurs dans des datacentres distants
Puissance	Très haute capacité de calcul et de stockage
Accès	Via Internet, depuis n'importe où dans le monde
Latence	Élevée (50 à 300 ms en général)

Aspect	Description
Scalabilité	Quasi illimitée, s'adapte automatiquement

Avantages

- **Puissance quasi illimitée** : des milliers de serveurs à disposition
- **Stockage massif** : des téraoctets de données historiques
- **Accessibilité** : disponible depuis n'importe où
- **Scalabilité** : la charge s'adapte automatiquement
- **Maintenance centralisée** : pas de matériel à gérer localement

Limites pour l'IoT

- **Latence** : trop lent pour des décisions en temps réel
- **Dépendance réseau** : sans connexion, pas de traitement
- **Coût de bande passante** : envoyer toutes les données peut être coûteux
- **Confidentialité** : les données sensibles quittent le site

Exemples de services cloud IoT

AWS IoT Core

Amazon

Azure IoT Hub

Microsoft

Cloud IoT Core

Google

Watson IoT

IBM

IoT Platform

Alibaba Cloud

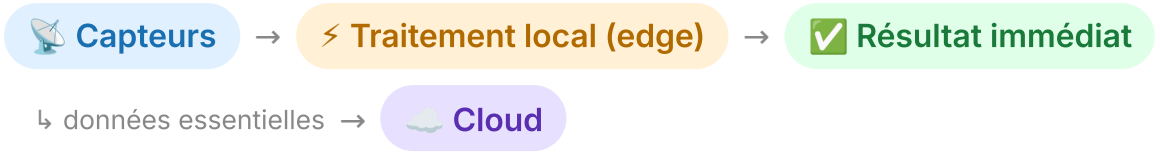
App Platform

DigitalOcean

⚡ Edge Computing

Principe

« Edge » signifie **« bord »** en anglais. L'edge computing consiste à **traiter les données au plus près de leur source** - directement sur le terminal, une passerelle locale ou un mini-serveur de proximité - avant (ou plutôt que) de les envoyer vers le cloud.



> **À retenir**
>
> Avec l'edge, on ne supprime pas le cloud : on **filtre et pré-traite** localement les données pour n'envoyer que l'essentiel.

Caractéristiques

Aspect	Description
Localisation	Sur le terminal, une passerelle ou un serveur local
Puissance	Limitée, mais suffisante pour le traitement local
Accès	Local, ne nécessite pas d'Internet
Latence	Très faible (moins de 5 ms)
Indépendance	Fonctionne même sans connexion Internet

Avantages

- **Réponse en temps réel** : décision prise immédiatement sur place
- **Fonctionnement hors-ligne** : indépendant d'Internet
- **Moins de trafic réseau** : seules les données utiles partent vers le cloud
- **Confidentialité renforcée** : les données sensibles restent localement
- **Usage industriel** : adapté aux environnements isolés ou critiques

Limites

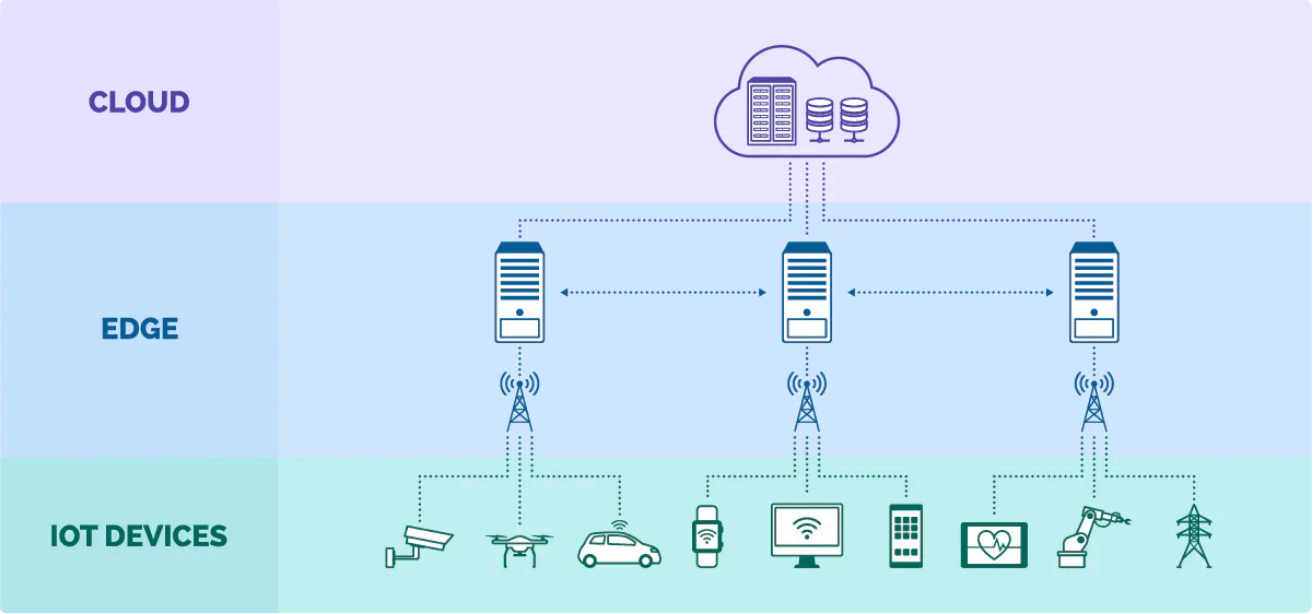
- **Puissance limitée** : pas adapté aux calculs très complexes
- **Maintenance locale** : le matériel est sur site
- **Stockage restreint** : pas d'historique long terme
- **Mises à jour** : plus complexes à déployer à distance

Exemples d'edge computing en IoT

Situation	Traitement à l'edge
Caméra de surveillance	Analyse d'image directement sur la caméra
Machine industrielle	Détection de panne sur la passerelle locale
Voiture autonome	Calcul des décisions de freinage embarqué
Bâtiment intelligent	Régulation automatique du chauffage via hub local
Capteur médical	Détection d'arythmie sur la montre connectée

Architecture en 3 couches

Un système IoT moderne combine généralement les deux approches dans une **architecture à 3 niveaux** :



Couche	Rôle principal	Exemples
Cloud	Analyser, stocker à long terme, IA	AWS, Azure, Google Cloud
Edge	Filtrer, pré-traiter, réagir localement	Raspberry Pi, gateway LoRaWAN, automate
Terminaux IoT	Collecter et transmettre les données brutes	Capteur de temp., caméra IP, actionneur

---

Cloud vs Edge : comparaison directe

Critère	☁️ Cloud	⚡ Edge
Latence	Élevée (50–300 ms)	Très faible (< 5 ms)
Connexion Internet requise	Oui	Non
Puissance de calcul	Très haute	Limitée
Stockage	Pratiquement illimité	Limité
Scalabilité	Excellente	Faible
Coût bande passante	Élevé	Faible
Confidentialité	Risque (données distantes)	Forte (données locales)
Maintenance	Centralisée (à distance)	Locale (sur site)

Approche hybride

La plupart des systèmes IoT professionnels combinent les deux :

- **Edge** pour les décisions rapides et critiques
- **Cloud** pour l'analyse approfondie et l'historique

Cas pratiques

🏠 Maison connectée

Fonction	Approche	Pourquoi
Détection de fumée → alarme locale	**Edge**	Immédiat, ne dépend pas d'Internet
Historique de température → graphiques	**Cloud**	Stockage et visualisation long terme
Contrôle à distance depuis l'étranger	**Cloud**	Accès depuis n'importe où

🏭 Usine connectée

Fonction	Approche	Pourquoi
Arrêt d'urgence d'une machine	**Edge**	Temps réel critique (ms)
Analyse prédictive de pannes	**Cloud**	IA sur données historiques massives
Rapport hebdomadaire de production	**Cloud**	Tableaux de bord centralisés

🚑 Santé connectée

Fonction	Approche	Pourquoi
Détection d'arythmie cardiaque	**Edge**	Réaction immédiate sur la montre
Partage avec le médecin	**Cloud**	Accès sécurisé à distance
Analyse de tendances sur 6 mois	**Cloud**	Historique et IA médicale

---

Technologies habilitantes

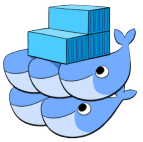
Trois technologies clés rendent le cloud et l'edge computing accessibles et efficaces pour l'IoT :

5G

Le réseau **5G** réduit la latence de communication entre les terminaux et le cloud à moins d'**1 ms**, rendant certains cas d'usage temps-réel envisageables directement via le cloud, sans edge intermédiaire.

Conteneurisation

Les **conteneurs** (Docker, Kubernetes) permettent d'empaqueter une application et de la déployer facilement sur des milliers d'appareils edge à distance, avec des mises à jour simplifiées.



API REST

Les **interfaces de programmation (API)** standardisent la communication entre les couches edge et cloud. Elles permettent d'intégrer des services tiers facilement et de faire communiquer des systèmes hétérogènes.

Résumé

Le **cloud computing** offre une puissance de traitement et de stockage quasi illimitées, accessible partout, mais avec une latence et une dépendance au réseau. L'**edge computing** rapproche le traitement des données de leur source pour des décisions en temps réel, même hors ligne. Dans la pratique, les systèmes IoT combinent les deux dans une **architecture à 3 couches** : terminaux → edge → cloud. On traite localement ce qui est urgent et critique, et on centralise ce qui nécessite une analyse approfondie ou un historique.

Approche	Cas d'usage idéal
Cloud uniquement	Analyse de données historiques, tableaux de bord, IA
Edge uniquement	Alarme locale, action réflexe, site isolé
Hybride (Edge + Cloud)	La majorité des systèmes IoT professionnels

Plateformes et intégration

Une plateforme IoE centralise la collecte, le traitement et la visualisation des données issues des terminaux. Comprendre le processus d'intégration et la chaîne de traitement est essentiel pour déployer une solution fonctionnelle.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Expliquer le rôle d'une **plateforme IoE** et ses composants.
- Décrire les **étapes d'intégration** d'un terminal dans une plateforme.
- Identifier les **services** nécessaires (broker, API, base de données, tableau de bord).
- Tracer le **flux de données** complet, du capteur à l'action.

Qu'est-ce qu'une plateforme IoE ?

Une plateforme IoE est un ensemble de **services logiciels** qui permettent de :

- **Collecter** les données des terminaux
- **Stocker** ces données de manière structurée
- **Traiter** et analyser les informations
- **Visualiser** les résultats (tableaux de bord, graphiques)
- **Déclencher** des actions automatiques (alertes, commandes)

Exemples de plateformes

Plateforme	Type	Description
Home Assistant	Open source	Domotique, intégration multi-protocoles
ThingsBoard	Open source	Tableau de bord IoT, gestion de terminaux
Node-RED	Open source	Automatisation par flux visuels

Plateforme	Type	Description
AWS IoT Core	Cloud	Plateforme IoT d'Amazon
Azure IoT Hub	Cloud	Plateforme IoT de Microsoft
The Things Network	Communautaire	Réseau LoRaWAN ouvert

Flux de données type

Le parcours des données dans un écosystème IoE suit généralement ce chemin :

📡 Capteur → 🌐 Réseau → 🔌 Plateforme → 🗄️ Stockage → 📊 Tableau de bord → ⚡ Action

Détail de chaque étape

Étape	Rôle	Exemple
Capteur	Mesure une grandeur physique	Température = 28.5°C
Réseau	Transporte les données	Wi-Fi, LoRa, Zigbee
Plateforme	Reçoit et route les données	Broker MQTT, API REST
Stockage	Conserve les données	Base de données, fichier
Tableau de bord	Affiche les données	Graphiques, jauges, cartes
Action	Réagit selon des règles	Alerte email, commande d'actionneur

Services d'une plateforme IoE

Broker MQTT

Le broker est le **point central** de communication en MQTT. Il reçoit les messages des publishers et les redistribue aux subscribers.

Exemples : Mosquitto, HiveMQ, EMQX

API REST

L'API REST expose des **endpoints** permettant aux terminaux et aux applications d'échanger des données via HTTP.

Exemples : API personnalisée (Node.js, Python Flask), API cloud

Base de données

Stocke les données historiques des terminaux pour l'analyse et la consultation.

Type	Exemples	Usage
Relationnelle	PostgreSQL, MySQL	Données structurées, configurations
Séries temporelles	InfluxDB, TimescaleDB	Mesures horodatées (capteurs)
NoSQL	MongoDB	Données hétérogènes, flexibles

Tableau de bord

Interface visuelle qui affiche les données en temps réel ou historiques.

Exemples : Grafana, ThingsBoard, Node-RED Dashboard

Moteur d'automatisation

Exécute des **règles** conditionnelles pour déclencher des actions automatiques.

Exemples : Node-RED, IFTTT, règles ThingsBoard

Étapes d'intégration

Pour intégrer un terminal dans une plateforme IoE :

1. **Préparer le terminal** : configuration réseau, identifiants
2. **Connecter au réseau** : Wi-Fi, Ethernet, LoRa...
3. **Authentifier le terminal** : clé API, certificat, identifiant MQTT
4. **Configurer l'envoi de données** : topic MQTT, endpoint API, format
5. **Vérifier la réception** : contrôler que les données arrivent sur la plateforme
6. **Configurer le stockage** : base de données, rétention
7. **Créer la visualisation** : tableau de bord, widgets
8. **Définir les automatisations** : alertes, actions conditionnelles
9. **Documenter** : schéma d'architecture, paramètres, procédures

---

Différencier les traitements

Traitement	Description	Exemple
Collecte	Recevoir les données brutes	Broker MQTT reçoit `temperature: 28.5`
Traitement	Transformer ou enrichir les données	Calculer une moyenne sur 10 minutes
Visualisation	Afficher les données	Graphique en temps réel dans Grafana
Déclenchement	Exécuter une action conditionnelle	Envoyer une alerte si température > 30°C

Résumé

Une plateforme IoE orchestre la collecte, le stockage, la visualisation et l'automatisation des données issues des terminaux. Le flux de données suit un chemin logique du capteur à l'action, en passant par le réseau, la plateforme et le stockage. L'intégration

d'un terminal nécessite une procédure en plusieurs étapes, de la configuration réseau à la documentation. Comprendre le rôle de chaque service (broker, API, base de données, tableau de bord) permet de concevoir et maintenir une solution IoE complète.

Paramétrage et automatisation

Au-delà de la configuration initiale, les terminaux IoE peuvent être adaptés à des besoins métier spécifiques par le paramétrage avancé ou le scriptage. Ce chapitre distingue ces deux approches et explore leurs possibilités.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Distinguer **paramétrage** et **scriptage** d'un terminal IoE.
- Identifier les possibilités d'**adaptation** d'un terminal à un besoin métier.
- Comprendre des exemples de **logique conditionnelle** simple.
- Évaluer les **limites** et les risques liés à l'adaptation des terminaux.

Paramétrage

Le paramétrage consiste à **ajuster les réglages** d'un terminal dans les limites prévues par le fabricant, sans modifier le code.

Exemples de paramétrage

Paramètre	Valeur par défaut	Adaptation
Fréquence de mesure	Toutes les 60 s	Toutes les 10 s (suivi intensif)
Seuil d'alerte	Aucun	Alerte si température > 30°C
Nommage	`device-001`	`capteur-temp-salle-201`
Mode de fonctionnement	Continu	Actif uniquement 8h–18h
Format de données	Brut	JSON structuré

Caractéristiques

- Ne nécessite **pas de programmation**
- Utilise l'interface du terminal ou de la plateforme
- Réversible et simple à modifier
- Limité aux **options prévues** par le fabricant

Scriptage

Le scriptage consiste à écrire ou modifier du **code** pour créer un comportement personnalisé qui n'est pas prévu nativement.

Exemples de logique conditionnelle

Scénario 1 : Alerte température

```
SI température > 30°C
  ALORS envoyer alerte "Surchauffe détectée"
FIN SI
```

Scénario 2 : Porte ouverte trop longtemps

```
SI porte ouverte DEPUIS plus de 2 minutes
  ALORS activer buzzer
  ET envoyer notification
FIN SI
```

Scénario 3 : Fréquence adaptative

```
SI heure ENTRE 22h ET 6h
  ALORS fréquence de mesure = 5 minutes
SINON
  fréquence de mesure = 30 secondes
FIN SI
```

Caractéristiques

- Nécessite des **compétences en programmation**
- Offre une flexibilité **bien supérieure** au paramétrage
- Peut introduire des **bugs** ou des comportements imprévus
- Doit être **documenté** et **testé**

Comparaison paramétrage vs scriptage

Critère	Paramétrage	Scriptage
Complexité	Faible	Moyenne à élevée
Compétences requises	Utilisation d'interface	Programmation
Flexibilité	Limitée aux options	Quasi-illimitée
Risque d'erreur	Faible	Moyen
Maintenance	Simple	Nécessite suivi
Documentation	Optionnelle	Indispensable

Règle pratique

Si le besoin peut être couvert par un **paramètre existant**, préférez le paramétrage. Le scriptage n'est justifié que lorsque le paramétrage ne suffit pas.

Adapter un terminal à un besoin métier

Démarche

1. **Analyser le besoin** : que doit faire le terminal exactement ?
2. **Vérifier les paramètres disponibles** : le besoin est-il couvert nativement ?
3. **Si non, évaluer le scriptage** : est-ce faisable ? quels risques ?
4. **Implémenter** : configurer ou coder l'adaptation
5. **Tester** : vérifier le comportement dans des conditions réelles

6. **Documenter** : noter les modifications et leur justification

Exemples concrets

Besoin métier	Solution
Alerte si température dépasse 30°C	Paramétrage (seuil d'alerte)
Allumer lumière si mouvement détecté	Scriptage (règle d'automatisation)
Réduire la fréquence d'envoi la nuit	Scriptage (condition horaire)
Changer le nom du terminal	Paramétrage (interface admin)
Envoyer un résumé quotidien par email	Scriptage (agrégation + envoi)

Limites et risques

Risque	Description
Complexité	Un script mal conçu peut rendre le terminal instable
Maintenance	Le code doit être maintenu lors des mises à jour firmware
Sécurité	Un script mal sécurisé peut créer des vulnérabilités
Documentation	Une adaptation non documentée est un risque pour l'équipe

---

Résumé

Le paramétrage permet d'ajuster un terminal IoE dans les limites prévues par le fabricant, tandis que le scriptage offre une flexibilité supérieure en ajoutant de la logique conditionnelle personnalisée. Le choix entre les deux dépend de la complexité du besoin et des compétences disponibles. Toute adaptation, qu'elle soit par paramétrage ou par scriptage, doit être testée et documentée pour garantir la fiabilité et la maintenabilité du système.

Tests et documentation

Le test et la documentation sont les étapes finales mais essentielles de l'intégration d'un terminal IoE. Ils garantissent le bon fonctionnement de la solution et permettent sa maintenance à long terme.

Objectifs

À la fin de ce chapitre, vous serez capable de :

- Rédiger des **cas de test** pertinents pour un terminal IoE.
- Distinguer **cas de test**, **protocole de test** et **rapport de test**.
- Produire une **documentation technique** complète d'un terminal.
- Comprendre l'importance de la **documentation utilisateur**.

Pourquoi tester ?

Tester un terminal et son intégration permet de :

- **Vérifier** que le terminal fonctionne comme prévu
- **Détecter** les problèmes avant la mise en production
- **Valider** la conformité avec les exigences
- **Garantir** la fiabilité dans des conditions réelles

Types de vérifications

Vérification	Description	Exemple
Connectivité	Le terminal est-il joignable sur le réseau ?	Ping, accès interface web
Envoi de données	Les données arrivent-elles à la plateforme ?	Vérification dans le broker/API

Vérification	Description	Exemple
Cohérence des données	Les valeurs sont-elles plausibles et correctes ?	Température = 22°C (pas 220°C)
Sécurité de base	Les accès sont-ils protégés ?	Mot de passe changé, HTTPS actif
Réaction attendue	Le système réagit-il correctement aux événements ?	Alerte déclenchée si seuil dépassé

Cas de test

Un cas de test décrit une **situation précise** à vérifier, avec le résultat attendu.

Structure d'un cas de test

Champ	Description
ID	Identifiant unique du test (ex. TC-01)
Description	Ce qui est testé
Prérequis	Conditions nécessaires avant le test
Étapes	Actions à réaliser
Résultat attendu	Ce qui doit se passer
Résultat obtenu	Ce qui s'est réellement passé
Statut	Réussi / Échoué

Exemple

Champ	Contenu
ID	TC-01
Description	Vérifier que le capteur de température envoie des données au broker MQTT

Champ	Contenu
Prérequis	Capteur configuré, broker Mosquitto actif
Étapes	1. Allumer le capteur — 2. Attendre 60 s — 3. Consulter le topic MQTT
Résultat attendu	Un message JSON avec la température apparaît sur le topic `salle/temp`
Résultat obtenu	*(à compléter lors du test)*
Statut	*(à compléter)*

Protocole de test

Le protocole de test est un **document structuré** qui regroupe l'ensemble des cas de test à exécuter pour valider une solution.

Contenu d'un protocole

- **Objectif** du test
- **Périmètre** : quels terminaux, quels services
- **Environnement** de test : réseau, plateforme, versions
- **Liste des cas de test** ordonnés
- **Critères de réussite** : seuil d'acceptation
- **Responsable** du test

Rapport de test

Le rapport de test est le **compte-rendu** de l'exécution du protocole.

Contenu d'un rapport

- **Date et heure** d'exécution
- **Résultats** de chaque cas de test
- **Anomalies** détectées et leur gravité

- **Conclusion** : solution validée ou non
- **Actions correctives** si nécessaire

Documentation technique

La documentation technique décrit de manière complète le terminal et son intégration.

Éléments à documenter

Élément	Description
Schéma d'architecture	Positionnement du terminal dans le réseau et la plateforme
Adresse IP / MAC	Identifiants réseau du terminal
Rôle	Fonction du terminal dans l'écosystème
Configuration	Paramètres réseau, fonctionnels, identifiants
Version firmware	Version installée et date de mise à jour
Procédure de maintenance	Étapes pour mettre à jour, redémarrer, remplacer
Historique des modifications	Changements apportés et dates

Documentation utilisateur

La documentation utilisateur est destinée aux **personnes qui utilisent** le système au quotidien.

Contenu

Élément	Description
Usage	Comment utiliser le terminal ou consulter les données
Limites	Ce que le terminal ne peut pas faire
Précautions	Consignes de manipulation, environnement requis

Élément	Description
Contact support	Qui contacter en cas de problème

Distinction cas / protocole / rapport

Document	Quand ?	Contenu
Cas de test	Avant le test	Décrit UNE vérification et son résultat attendu
Protocole de test	Avant le test	Regroupe TOUS les cas de test et les conditions
Rapport de test	Après le test	Résultats réels, anomalies, conclusion

Résumé

Le test et la documentation sont indissociables de l'intégration IoE. Les cas de test vérifient le fonctionnement individuel de chaque composant, le protocole organise l'ensemble des vérifications, et le rapport consigne les résultats. La documentation technique assure la traçabilité de la configuration et facilite la maintenance, tandis que la documentation utilisateur guide l'exploitation au quotidien. Ces livrables constituent la preuve formelle qu'une solution IoE est opérationnelle et maintenable.