

# IoT

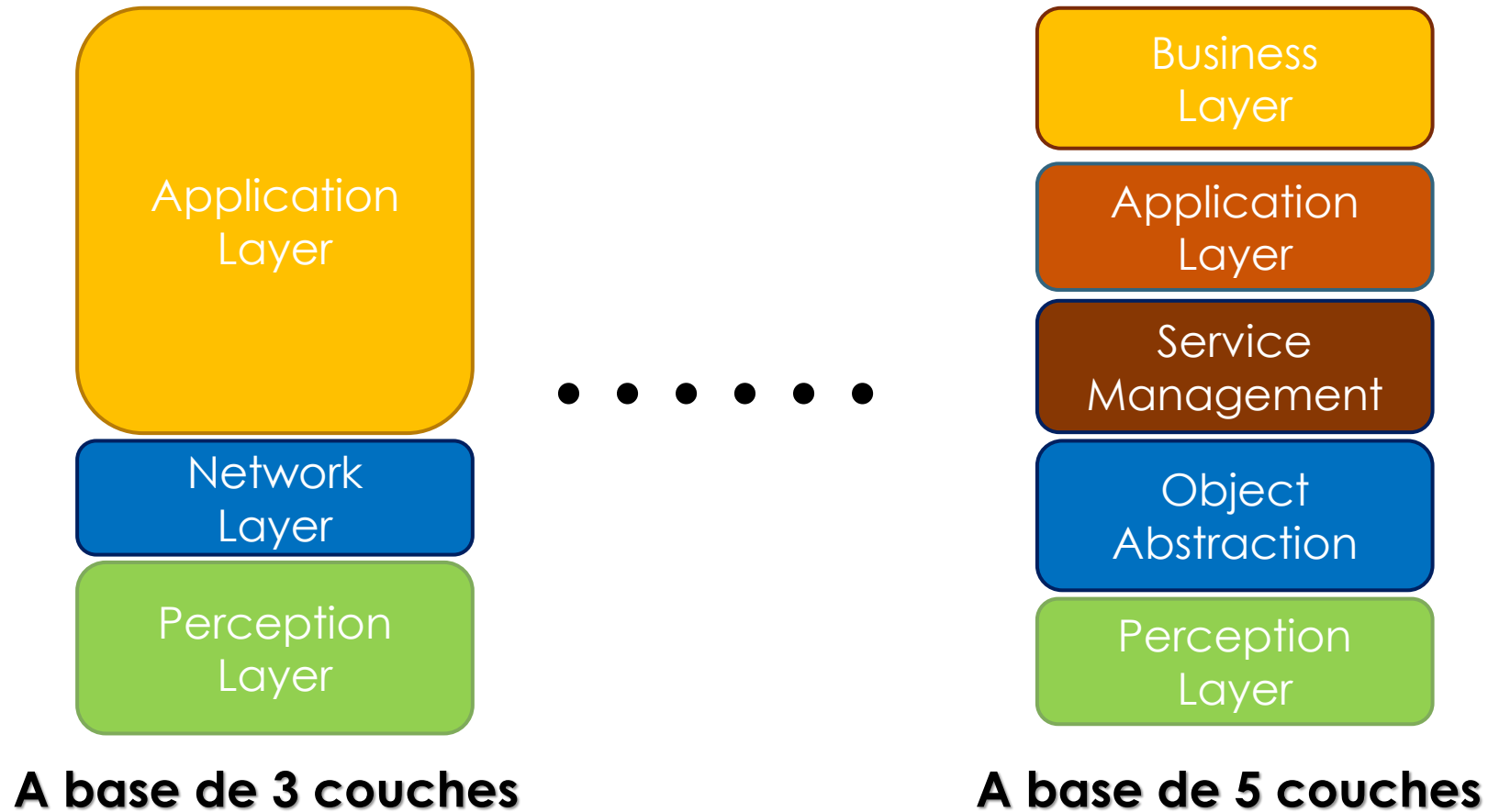
## Concepts de Base

# Cours 2

# IoT Architecture

- ▶ Devrait être capable d'interconnecter des milliards voire des trillions d'objets hétérogènes via Internet.
- ▶ Nécessite une architecture en couches flexible.
- ▶ Le nombre croissant d'architectures proposées n'a pas encore convergé vers un modèle de référence.
- ▶ En attendant, il existe des projets tels que **IoT-A** qui tentent de concevoir une architecture commune basée sur l'analyse des besoins des chercheurs et de l'industrie.

# IoT Architecture



# IoT Architecture

- ▶ Parmi les modèles proposés, le modèle de base est une architecture à 3 couches.
- ▶ Composée des couches Application, Réseau et Perception.
- ▶ Cependant, dans la littérature récente, d'autres modèles ont été proposés, ajoutant davantage d'abstraction à l'architecture de l'IoT.

Application  
Layer

Network  
Layer

Perception  
Layer

# IoT Architecture

- ▶ Certaines architectures courantes, dont le modèle à 5 couches, qui a été utilisé.

- ▶ Object Abstraction prend certaines action de perception layer
- ▶ Ainsi que celle du Network Layer

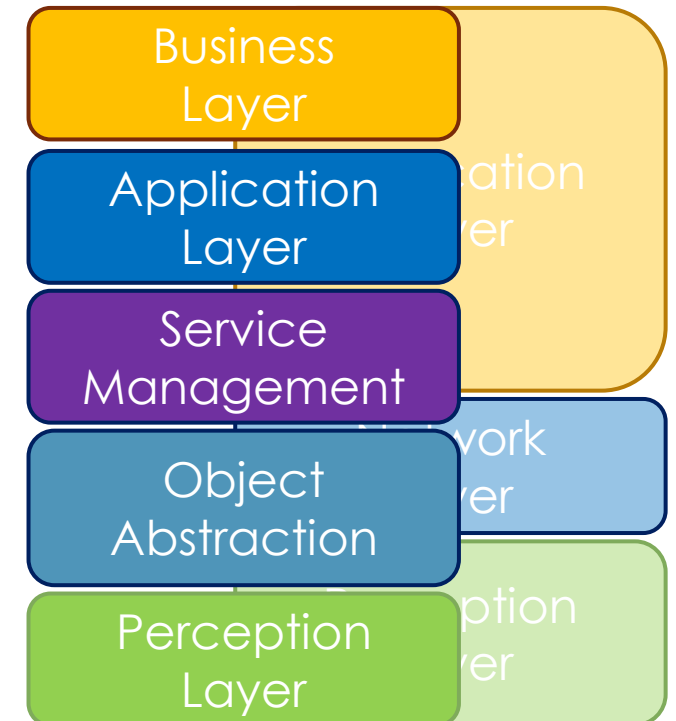
Object  
Abstraction

Service  
Management

Business  
Layer

- ▶ Application layer est éclatée Business, Application, et Service Management en raison de

- ▶ Remarque : Plusieurs modèles à 5 couches peuvent être trouvés dans la littérature.



# IoT Architecture, les couches en détails.

## Objects Layer

- ▶ Les données volumineuses créées par l'IoT sont initiées à cette couche.
- ▶ Représente les capteurs physiques de l'IoT qui ont pour but de collecter et de traiter des informations.
- ▶ Comprend des capteurs et des actionneurs pour effectuer différentes fonctionnalités telles que la localisation, les mouvements mécaniques ou les mesures des grandeurs physiques.
- ▶ Des mécanismes standardisés de branchement et de lecture doivent être utilisés par la couche de perception pour configurer des objets hétérogènes.
- ▶ Numérise et transfère les données vers la couche d'abstraction des objets via des canaux sécurisés.

Perception  
Layer

# IoT Architecture, les couches en détails.

## Objects Abstraction Layer

8

- ▶ La couche d'abstraction des objets transfère les données produites par la couche des objets vers la couche de gestion des services.
- ▶ Le transfert est via des canaux sécurisés.
- ▶ Les données peuvent être transférées via différentes technologies telles que RFID, 3G, GSM, UMTS, WiFi, Bluetooth Low Energy, infrarouge, ZigBee, CanBus, I2C etc.
- ▶ De plus, d'autres fonctions telles que le cloud computing et les processus de gestion des données sont gérées à cette couche.
- ▶ Inclusion les communications de base (I2C, RFID, CanBus ...) qui étaient dans la couche perception du modèle à 3 couches.
- ▶ Ainsi, les service de networking IP, ... .

Object  
Abstraction



# IoT Architecture, les couches en détails. Service Management Layer

- ▶ La couche middleware
- ▶ Associe un service à son demandeur en fonction des adresses et des noms.
- ▶ Cette couche permet aux programmeurs d'applications IoT de travailler avec des objets hétérogènes sans se soucier d'une plateforme matérielle spécifique.
- ▶ De plus, cette couche traite les données reçues, prend des décisions et fournit les services requis via les protocoles de communication réseau.

Service  
Management

# IoT Architecture, les couches en détails.

## Application Layer

- ▶ Fournit les services demandés par les clients.
  - ▶ Par exemple, la couche d'application peut fournir des mesures de température et d'humidité de l'air au client qui demande ces données.
- ▶ Son importance réside dans sa capacité à fournir des services intelligents de haute qualité pour répondre aux besoins des clients.
- ▶ La couche d'application couvre de nombreux marchés verticaux tels que la domotique, les bâtiments intelligents, les transports, l'automatisation industrielle et les soins de santé intelligents.

Application  
Layer

# IoT Architecture, les couches en détails.

## Business Layer – Couche Métier

- ▶ Gère l'ensemble des activités et des services du système IoT.
- ▶ Elaborer un modèle commercial
  - ▶ des graphiques, des organigrammes, etc.
  - ▶ basés sur les données reçues de la couche d'application.
- ▶ Chargée de concevoir, d'analyser, de mettre en œuvre, d'évaluer, de surveiller et de développer des éléments liés au système IoT.
- ▶ Assure les processus de prise de décision basés sur l'analyse de Big Data.
- ▶ Assure la surveillance et la gestion des quatre autres couches sous-jacentes.
- ▶ Préserver la confidentialité des utilisateurs.

Business  
Layer

# Cours 3

# La pile IoT ou le design physique de l'IoT



Objet



Traitement de  
la donnée

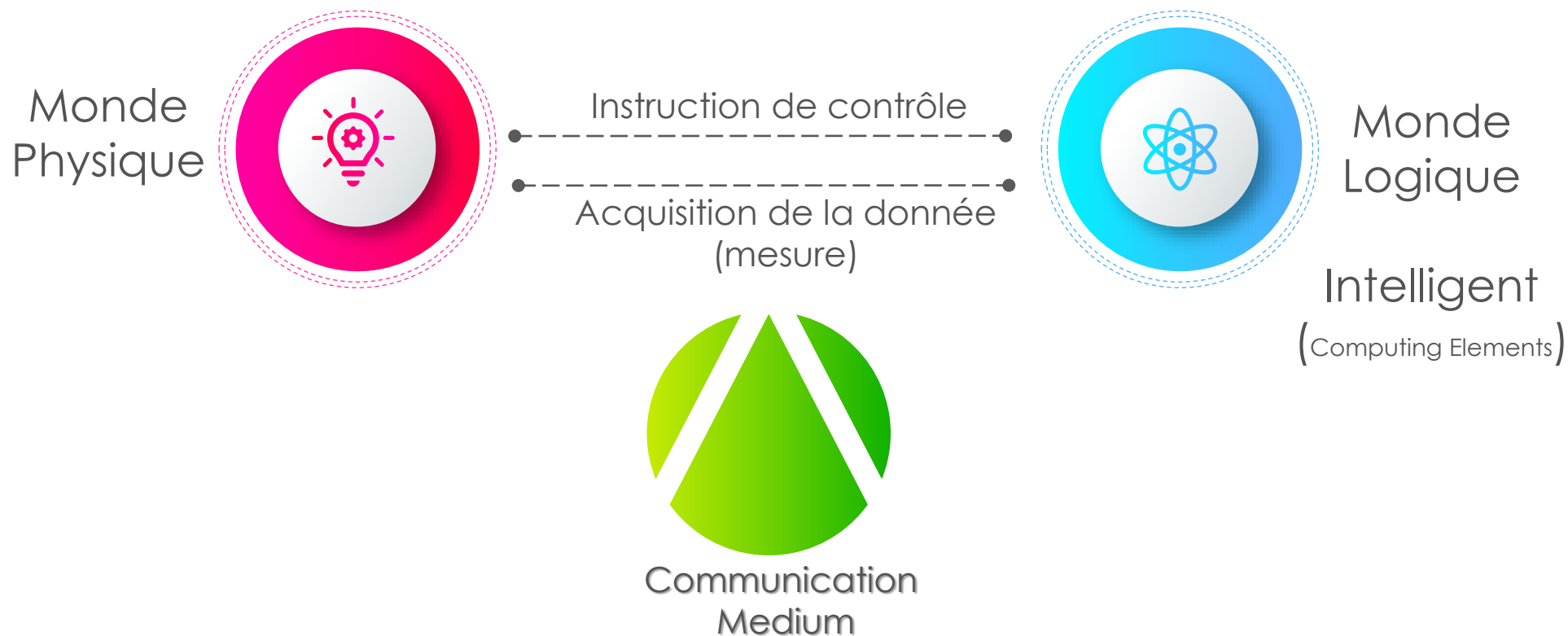


Acquisition de  
la donnée



Module de  
communication

# Cyber-Physical System

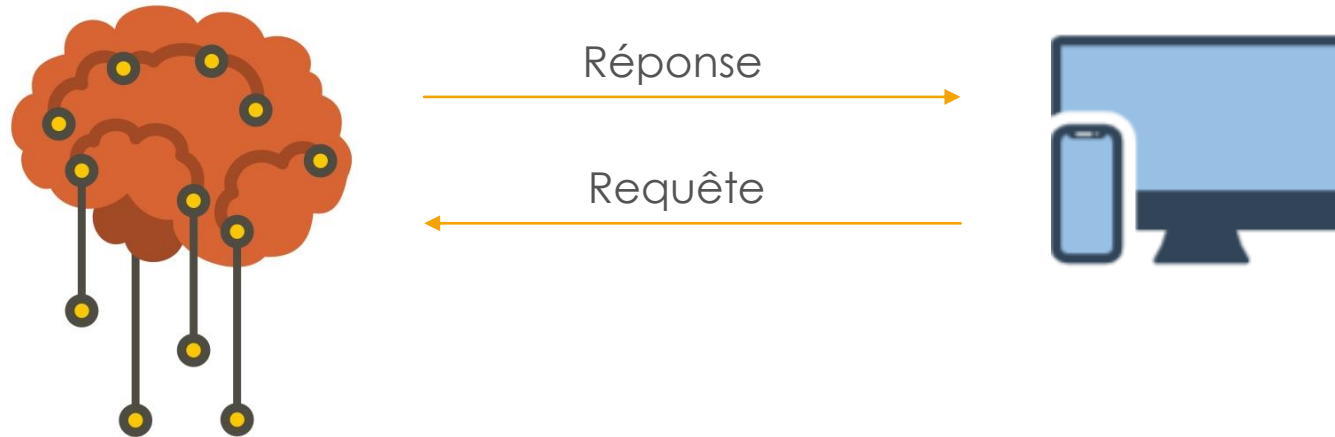


# Les modèles de Communications

- ▶ Les protocoles de la couche Application sont principalement catégorisés :
  - ▶ Request-Response
  - ▶ Publish-Subscribe
  - ▶ Push-Pull
  - ▶ Paire Exclusive
- ▶ Les protocoles à base de : Request-Response et Publish-Subscribe sont les mieux accueillis par les communautés scientifiques et industrielles.

# Les modèles de Communications

## ► Request-Response





# Les modèles de Communications

## ▶ Request-Response

- ▶ Basé sur le principe d'une interaction directe entre un appareil émetteur (client) et un appareil récepteur (serveur).
- ▶ Le client envoie une demande au serveur, qui répond ensuite avec une réponse.
- ▶ C'est le modèle le plus simple et largement utilisé dans les applications IoT.

## ▶ Exemples

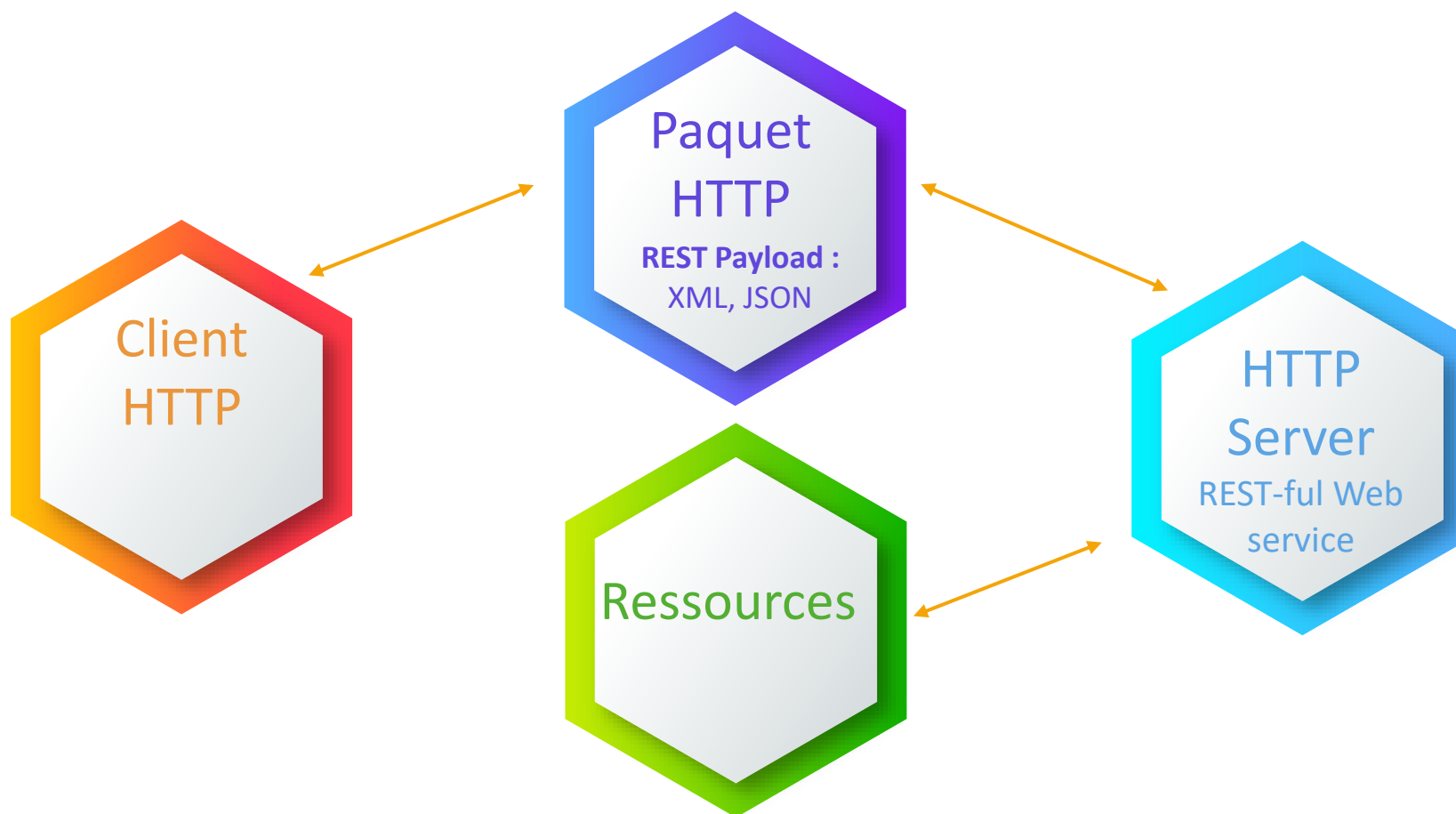
- ▶ RESTful HTTP
- ▶ XMPP Extensible messaging and presence protocol
- ▶ CoAP
- ▶ WebSocket

# Les modèles de Communications

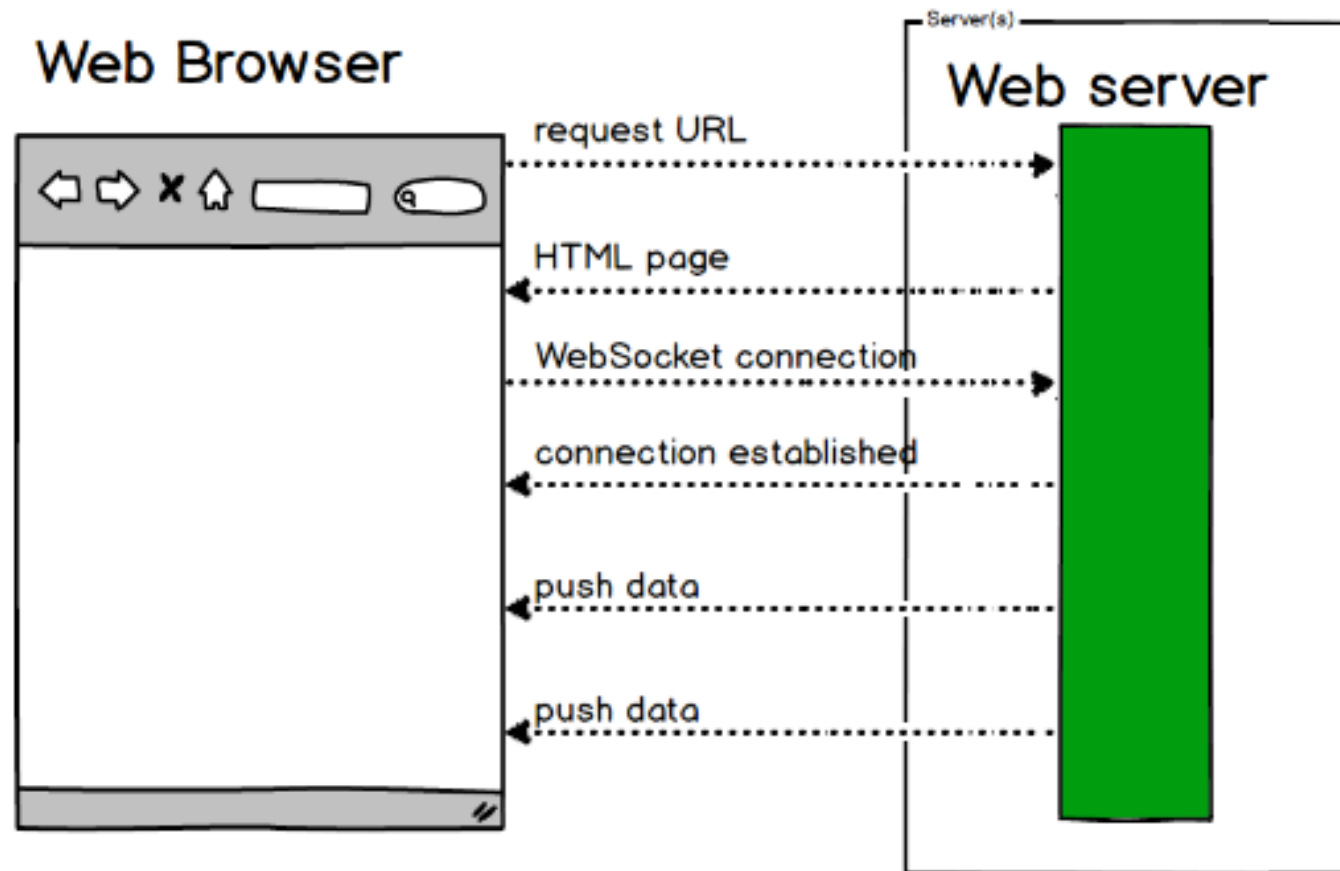
## CoAP

- ▶ Un protocole de couche de session qui offre une interface RESTful (HTTP) entre un client et un serveur HTTP.
- ▶ Conçu par le groupe de travail Constrained RESTful Environment (CoRE) de l'IETF.
- ▶ Son architecture complète se compose de:
  - ▶ Un client CoAP.
  - ▶ Un serveur CoAP.
  - ▶ Un proxy REST CoAP.

# Communication APIs : REST-based.

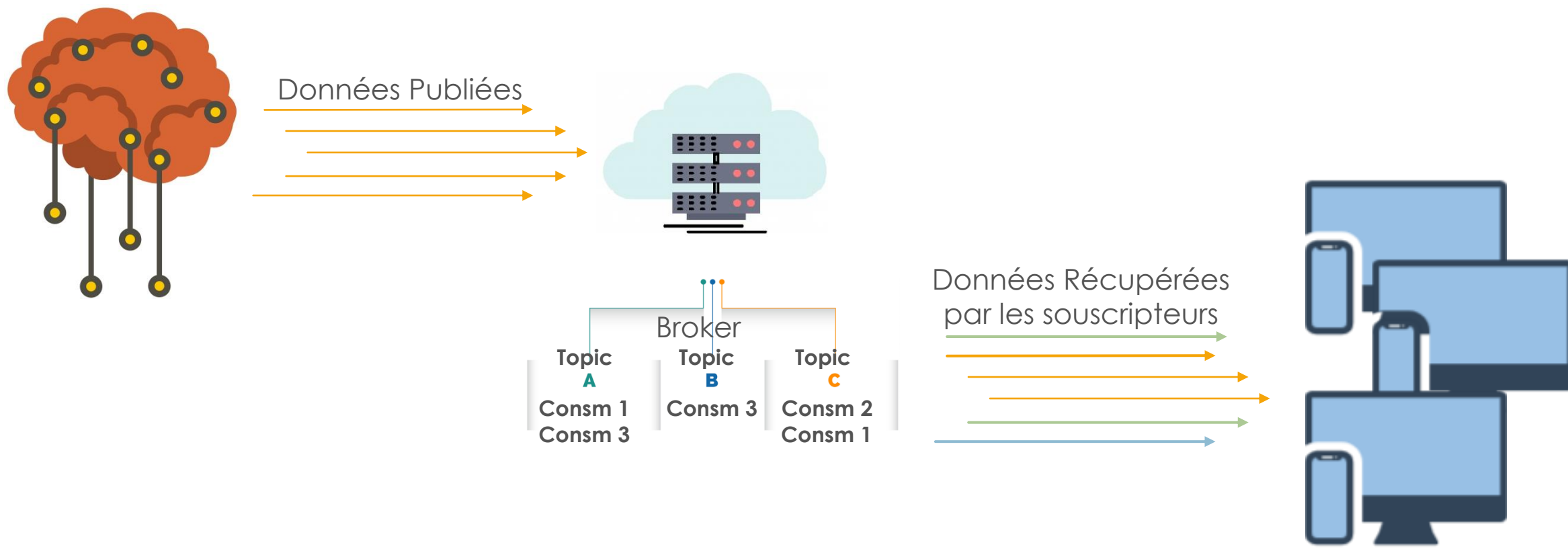


# Communication APIs : WebSocket-based.



# Les modèles de Communications

## ► Publish-Subscribe



# Les modèles de Communications

## ► Publish-Subscribe

- Les devices sont divisés en deux rôles : les éditeurs (publishers) et les abonnés (subscribers).
- Les éditeurs envoient des messages au broker (Ex : serveur MQTT)
- Ce Broker peut héberger des canaux spécifiques appelés Topics.
- Les abonnés s'abonnent à ces canaux pour recevoir les messages.

- La communication est asynchrone entre les deux entités communicantes.

## ► Exemples

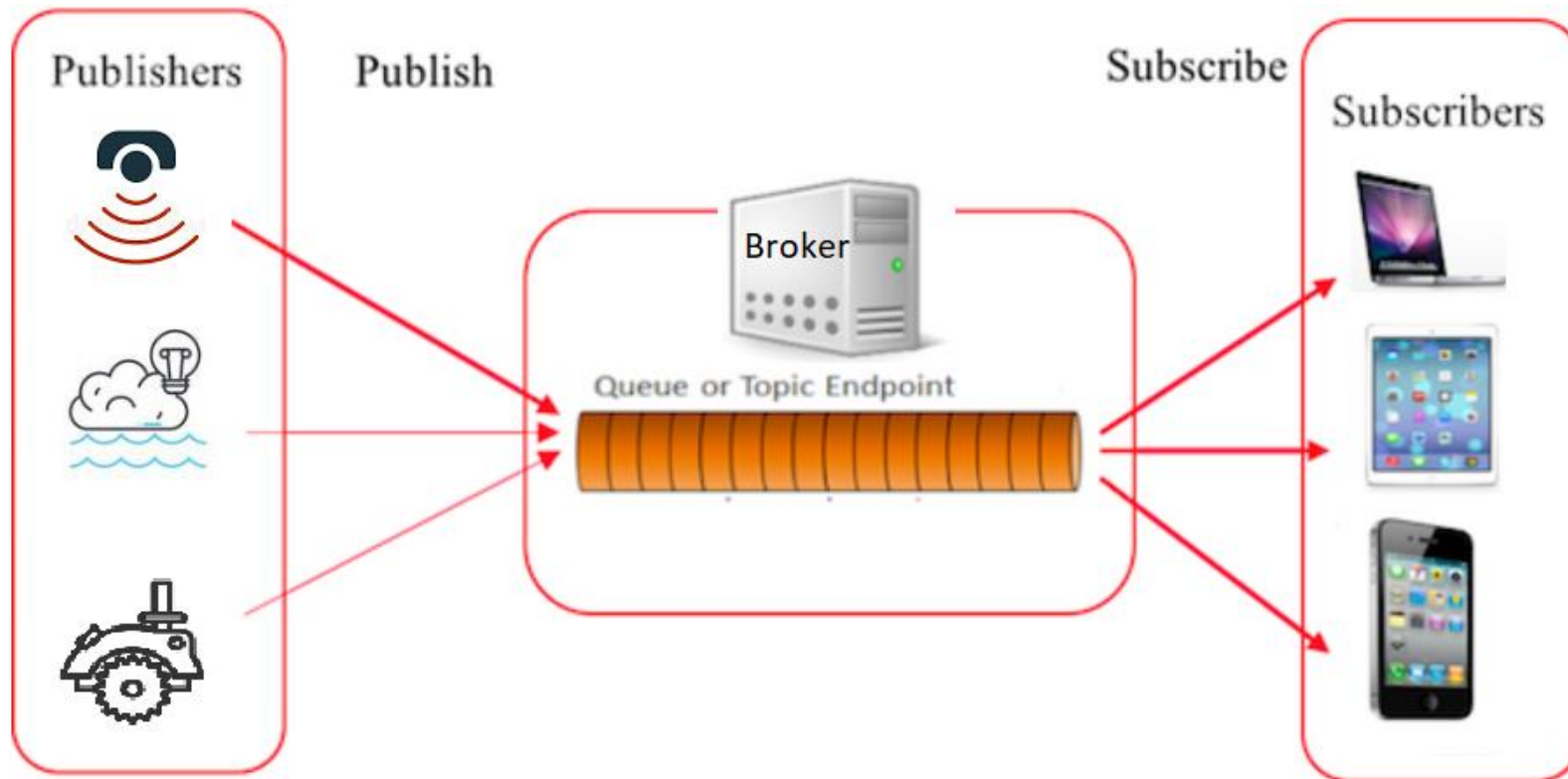
- MQTT (Message Queuing Telemetry Transport).
- AMQP (Advanced Message Queuing Protocol).
- Web Services-Notification
- STOMP (Simple/Streaming TextOriented Messaging Protocol)
- DDS (Data Distribution Service)

# Les modèles de Communications

- ▶ Exemple d'application de MQTT le plus répandu a nos jours.
- ▶ Introduit par IBM en 1999, conçu pour surveiller les nœuds capteurs et le suivi à distance dans l'IoT.
- ▶ Les publishers sont des capteurs, et Les subscribers sont des applications qui s'intéressent à des données sensorielles spécifiques.
- ▶ Les subscribers se connectent aux brokers pour être informées dès que de nouvelles données sont reçues.
- ▶ Un Broker reçoit les données sensorielles, les filtre en fonction des différents sujets et les envoie aux subscribers intéressés par ces sujets.
- ▶ Dans notre exemple nous adoptons le serveur MQLATTO qui offre un service de broker sur :  
<https://maqiatto.com/>
  - ▶ Il suffit de créer un compte qui peut représenter l'identifiant de l'application.
  - ▶ Créer les Topics suivant les besoins.

# Les modèles de Communications


## MQTT





# Configuration – Broker MAQIATTO

[←](#) [↻](#) <https://maqiatto.com/configure>

 [ABOUT](#) [HOW TO ▾](#) [MONITOR](#) [CONTACT](#) [PRICING & SUPPORT](#)

---

## Broker Configuration


### Topic Management

- Notice your MQTT user name is a prefix for all topics. To get rid of prefixes and be able to use wildcards:  
[Become a Patron!](#)

New Topic :

Identifiant/

+ Add Topic


- Available Topics :
  - | Identifiant/test 

---

### User Management

- Username for your MQTT connection is :  . To change it:  
[Become a Patron!](#)

MQTT Username :

 Change Username

# Configuration – Broker MAQIATTO

- ▶ En effet, MAQIATTO, offre également la possibilité d'utiliser le mode Web-Socket.

## Port Management

**MQTT TCP Port :**

**Secure MQTT over TLS/SSL:**

[Use Certificate](#)

Become a Patron!

**MQTT over Websocket Port :**

# Codage – Publisher MQTT MAQIATTO

- ▶ Il est impératif d'installer la librairie du client MQTT : paho-mqtt.
- ▶ Définir les paramètres du compte Broker créé préalablement, ainsi que Les topics à utiliser.
- ▶ Connection et publication a travers la classe mqtt.Client.

```
import paho.mqtt.client as mqtt

# Set up MQTT broker connection
broker_address = "maqiatto.com"
broker_port = 1883
broker_username = "Identifiant"
broker_password = "IoTT"

# Broker topics
Topic = broker_username+"/test"

# MQTT Client
client = mqtt.Client()
client.username_pw_set(username=broker_username,
                        password=broker_password)
client.connect(broker_address)
client.publish(Topic, number)
```

# Codage – Subscriber MQTT MAQIATTO

- ▶ En suivant la même logique de connexion du publisher est appliquée pour le subscriber.
- ▶ Cependant, a chaque fois cette entité Souhaite récupérer une donnée, la méthode Subscribe(topic) de la classe client est invoquée

```
import paho.mqtt.client as mqtt
# Set up MQTT broker connection
broker_address = "maqiatto.com"
broker_port = 1883
broker_username = "Identifiant"
broker_password = "IoTT"
# MQTT Client
client = mqtt.Client()
client.username_pw_set(username=broker_username,
                        password=broker_password)
client.connect(broker_address)
# Topic
Topic = broker_username+"/test"

def on_message(client, userdata, message):
    access_message = str(message.payload.decode("utf-8"))
    print(access_message)

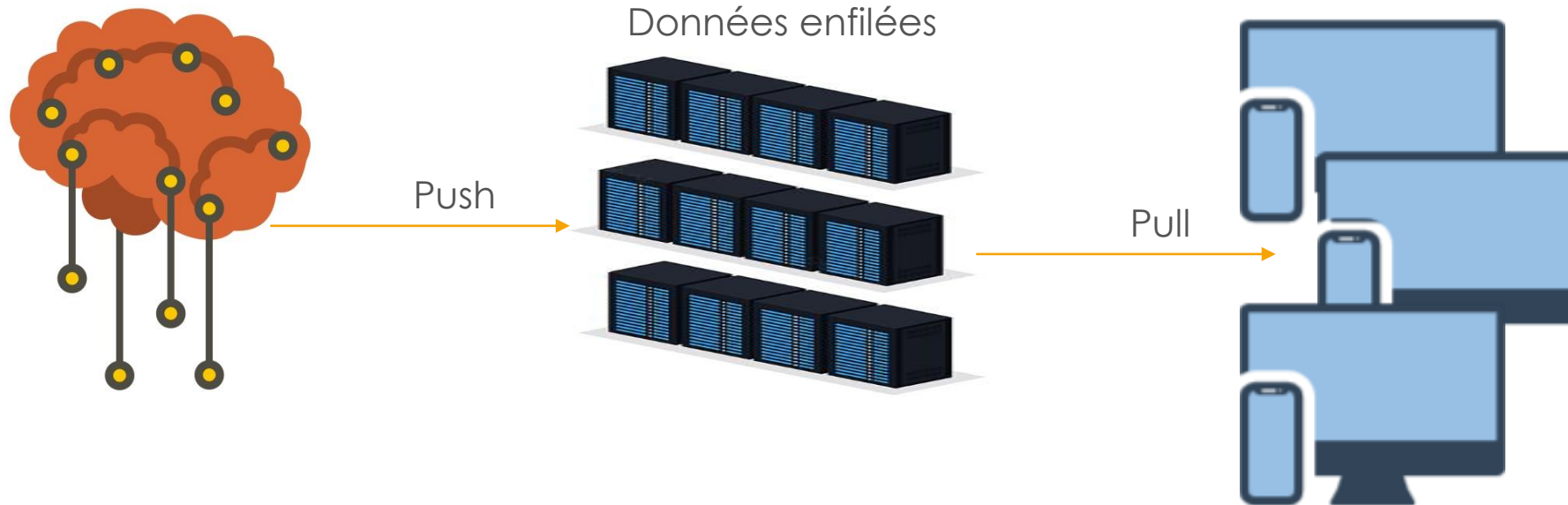
client.loop_start()

client.subscribe(door)
client.on_message = on_message
time.sleep(300)

client.loop_stop()
```

# Les modèles de Communications

## ► Push-Pull



# Les modèles de Communications

## ▶ Push-Pull

- ▶ Combine les caractéristiques des modèles Request-Response et Publish-Subscribe.
- ▶ Les devices peuvent envoyer des messages de manière proactive (push) à d'autres devices spécifiques.
- ▶ Les appareils destinataires peuvent récupérer (pull) les messages lorsqu'ils en ont besoin.
- ▶ Les files d'attente agissent également comme un tampon qui aide dans les situations où il y a un décalage entre la vitesse à laquelle les producteurs poussent les données et la vitesse à laquelle les consommateurs tirent les données.

## ▶ Exemple

- ▶ MQTT based-on WebSocket

# Cours 4

# Déploiement des Systèmes IoT

- ▶ Un système IoT est constitué de la composante suivante, pour un éventuel déploiement :
  - ▶ Dispositif (Device)
  - ▶ Ressource (Resource)
  - ▶ Service de contrôleur (Controller Service)
  - ▶ Base de données (Data Base, Data Wharehouse, Data Lake)
  - ▶ Service Web (Web Service)
  - ▶ Composant d'analyse (Analyse Component)
  - ▶ Application.



# Déploiement des Systèmes IoT

- ▶ Dispositif (Device)
  - ▶ Possède des capacités telles que la détection, l'actionnement et la surveillance.
  - ▶ Un dispositif IoT aura toujours une identité unique.
  - ▶ Les capteurs sans fil, les applications, les actionneurs et les dispositifs informatiques sont des exemples de dispositifs IoT.
- ▶ Ressource (Resource)
  - ▶ Chaque dispositif IoT dispose d'un module logiciel pour l'entrée, le traitement et le stockage des données des capteurs.
  - ▶ Ils sont donc utilisés pour contrôler les actionneurs connectés aux dispositifs.

# Déploiement des Systèmes IoT

- ▶ Service de contrôleur (Controller Service)
  - ▶ Agit comme un connecteur entre le service web et le dispositif.
  - ▶ Envoie des données du système vers le service web
  - ▶ Reçoit des commandes pour contrôler le dispositif à partir de l'application (via les services web).
- ▶ Base de données
  - ▶ Le nom du composant est uniquement représentatif, en effet, C'est référentiel de toutes les données fournies par les dispositifs IoT.
  - ▶ Il est soit conservé localement, soit sur le cloud sous la forme d'une base de données ou autres.

# Déploiement des Systèmes IoT

- ▶ Service Web
  - ▶ Les services web connectent les dispositifs IoT, l'application, la base de données et les composants d'analyse.
  - ▶ Les services web peuvent être mis en œuvre soit en utilisant différents concepts en occurrence HTTP, REST, WebSocket.
- ▶ Composant d'analyse (Analyse Component)
  - ▶ Récupère les données de la base de données des dispositifs IoT.
  - ▶ Les transforme en informations utiles afin d'extraire une valeur ajoutée.
  - ▶ C'est une composante qui entre dans le cadre de traitement des données BigData.
  - ▶ Ce module analyse les données, produit des résultats et les présente sous une forme conviviale à l'aide de divers algorithmes.
  - ▶ Cette analyse peut être effectuée localement ou dans le cloud, et les données résultantes peuvent être stockées localement ou dans le cloud.

# Déploiement des Systèmes IoT

- ▶ Application.
  - ▶ Actuellement, Les applications IoT fournissent une interface que les utilisateurs peuvent utiliser pour contrôler et surveiller différents aspects du système IoT en occurrence les dispositifs.
  - ▶ Les applications permettent également aux utilisateurs de visualiser l'état du système et les données traitées.
- ▶ Ces différents composants sont déployés suivant les différents objectifs des solutions IoT. Dans ce qui suit une nombre de niveaux de déploiement sont discutés.

# Niveaux de déploiement IoT

01

LEVEL

02

LEVEL

03

LEVEL

04

LEVEL

05

LEVEL

06

LEVEL

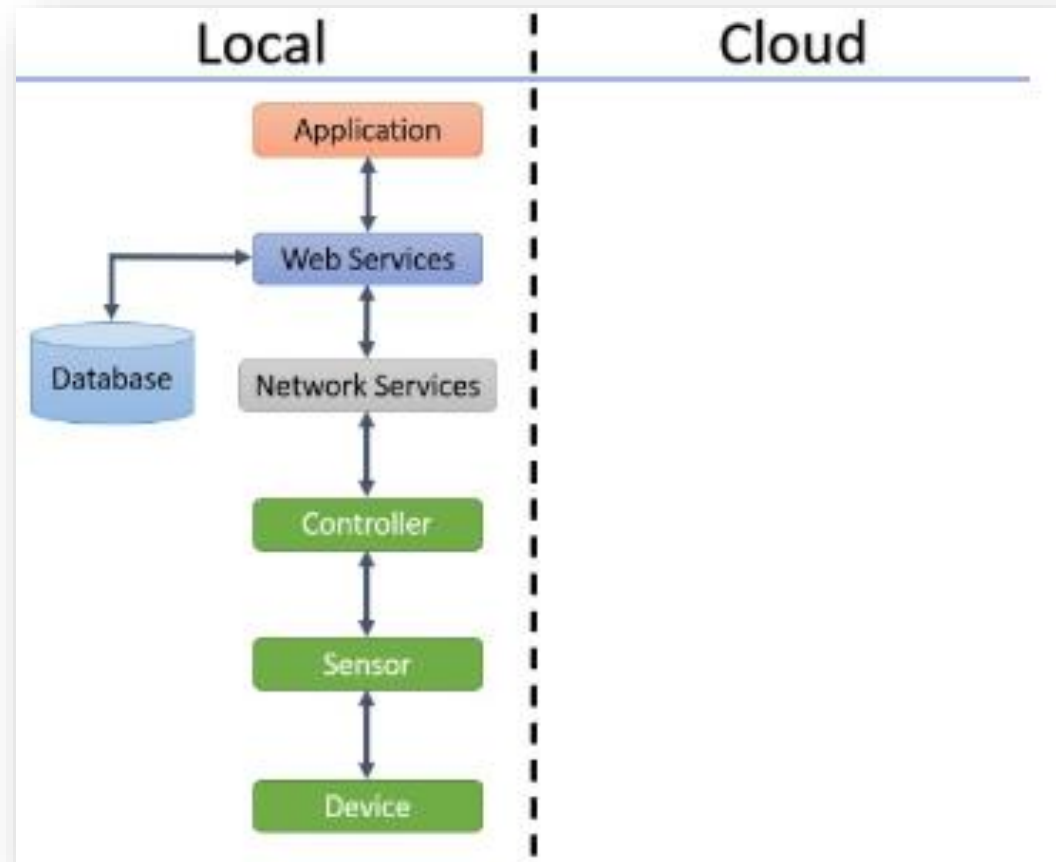
# Niveaux de déploiement IoT

- ▶ Un écosystème IoT peut être conçu à différents niveaux.
- ▶ Chaque niveau est fondamentalement une feuille de route sur la manière de déployer les composants de l'écosystème IoT.
- ▶ Ainsi, le même écosystème IoT peut être conçu en utilisant à la fois les normes de niveau 1 et de niveau 2.
- ▶ La seule chose qui différera est la façon dont les composants sont disposés ensemble.

# Déploiement de niveau 1

**01**

LEVEL



# Niveaux de déploiement IoT

## LEVEL 01

- ▶ A ce niveau, tous les composants sont déployés localement.
- ▶ Il n'y a aucun nuage ou réseau externe impliqué dans le processus.
- ▶ Cette norme est dédiée pour les écosystèmes où les données ne sont ni volumineuses ni variables.
- ▶ Un flux uniforme de données provient d'un groupe prédéfini de capteurs et tout se passe de manière simple.



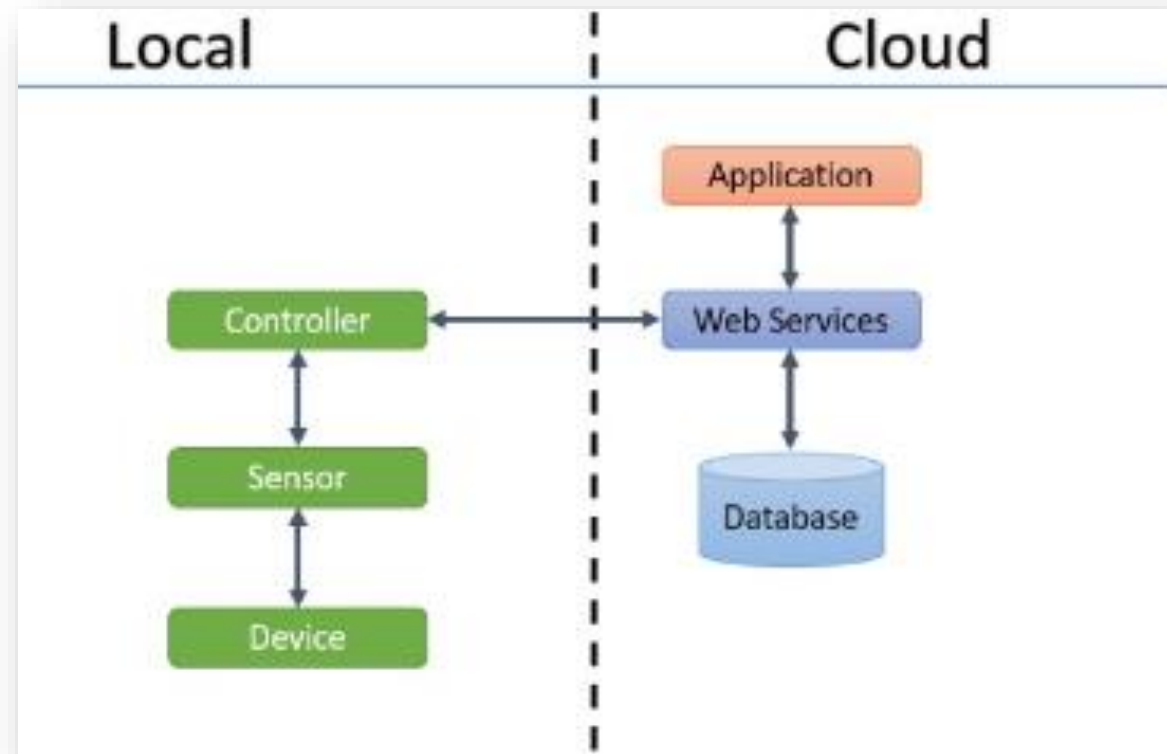
# Déploiement de niveau 1

- ▶ Un exemple d'IoT de niveau 1 est une maison intelligente.
  - ▶ Un seul nœud surveille l'humidité du sol dans le champ qui est envoyée à la base de données sur le cloud à l'aide de REST APIs.



# Déploiement de niveau 2

## 02 LEVEL



# Déploiement de niveau 2

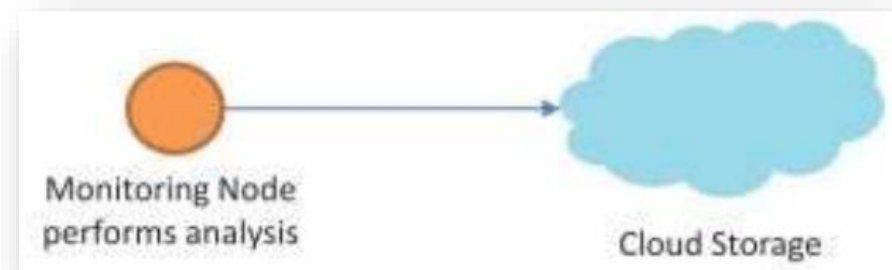
## LEVEL 02

- ▶ A ce niveau, tous les composants sont déployés localement à l'exception des serveurs.
- ▶ Un cloud ou un réseau externe est impliqué dans le processus, à des fins de stockage et d'analyse.
- ▶ Du côté de l'utilisateur, il n'y a que des capteurs, des routeurs et des applications.
- ▶ Cette norme est dédiée pour les écosystèmes où les données sont volumineuses qui proviennent de nombreux composants et à une grande vitesse.

# Déploiement de niveau 2

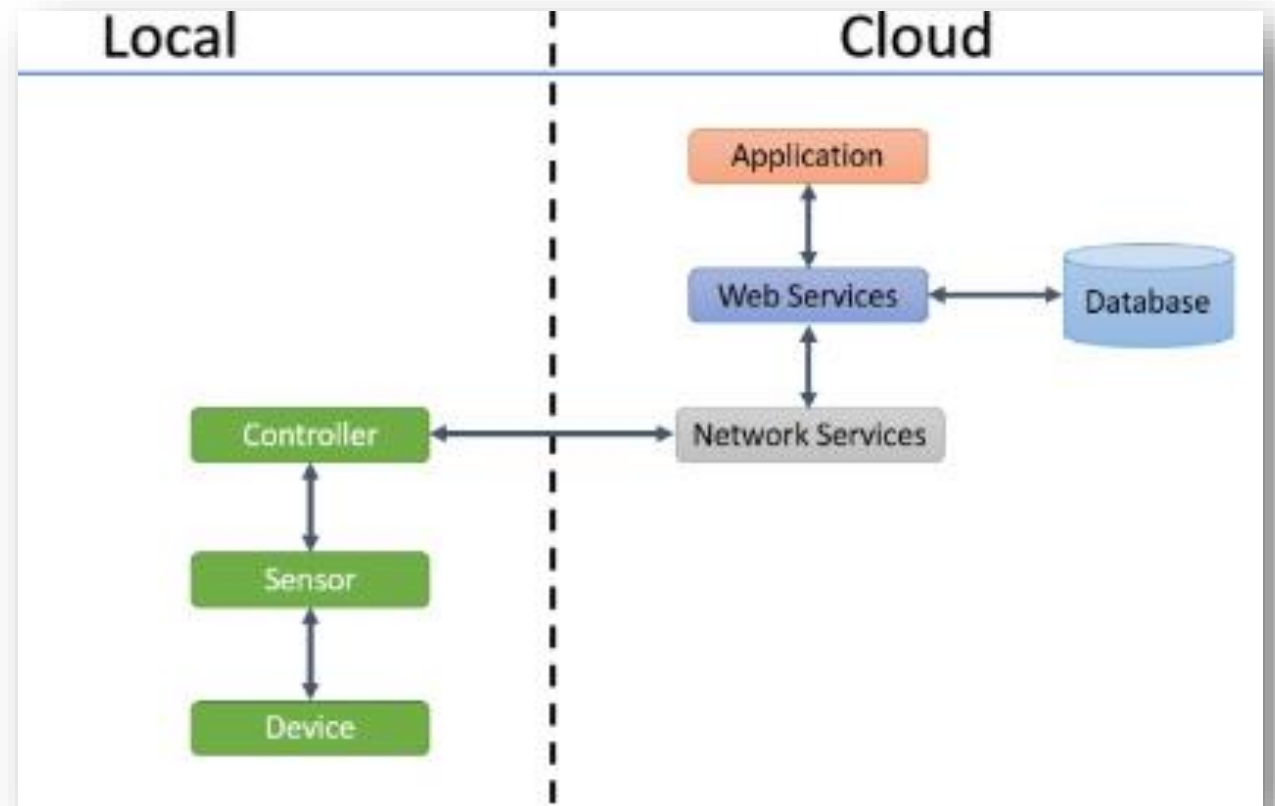
## LEVEL 02

- ▶ Un exemple d'IoT de niveau 2 est d'irrigation intelligente.
  - ▶ Une application basée sur le cloud est utilisée pour surveiller et contrôler le système IoT.
  - ▶ Un nœud surveille l'humidité du sol sur le terrain qui est envoyée à la base de données sur le cloud à l'aide de REST APIs.
  - ▶ Le service de contrôle surveille en permanence les niveaux d'humidité.
  - ▶ Un seul nœud surveille l'humidité du sol et contrôle le système d'irrigation.



# Déploiement de niveau 3

## 03 LEVEL



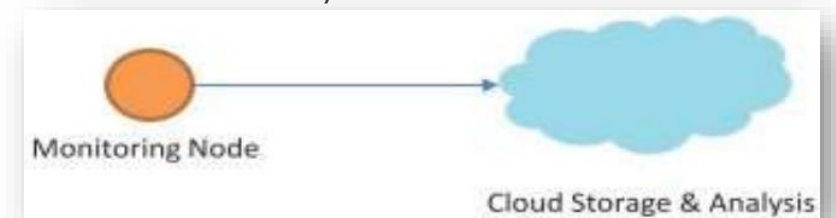
# Déploiement de niveau 3

- ▶ A ce niveau tous les composants sont déployés localement sauf les serveurs et la partie connectivité réseau.
- ▶ Un cloud ou un réseau externe est impliqué dans le processus.
- ▶ Les données sont stockées et analysées dans le cloud et l'application est basée sur le cloud.
- ▶ Convient aux solutions impliquant de grandes quantités de données et les exigences d'analyse sont intensives en calcul.
- ▶ Un exemple de niveau 3 IoT est une industrie intelligente.

LEVEL 03

# Déploiement de niveau 3

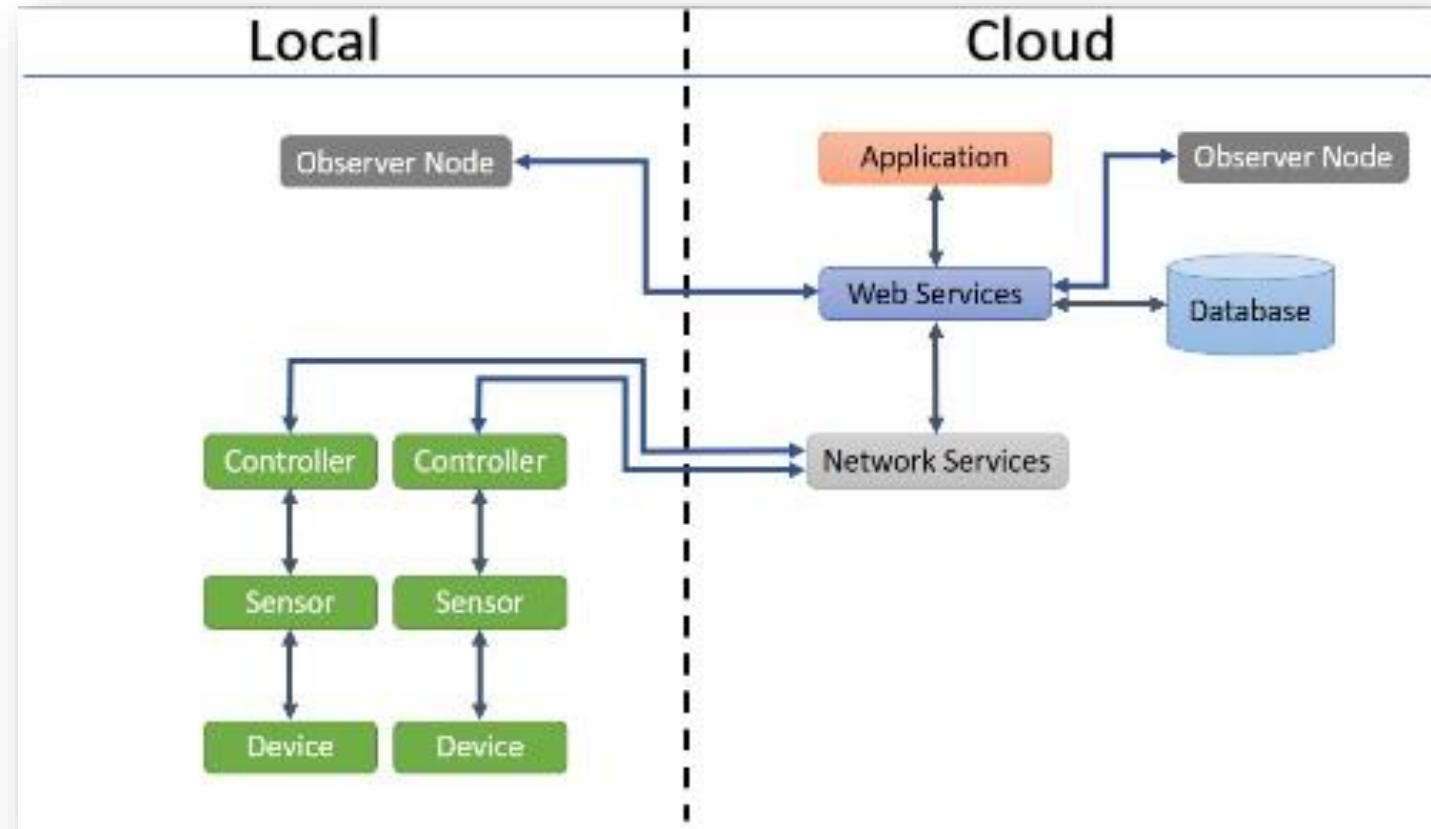
- ▶ La surveillance des colis dans un système de distribution et de livraison de colis (parcels).
  - ▶ Les mouvements qui se produisent sur le colis sont passés en analyse.
  - ▶ S'ils dépassent le seuil, une alarme est déclenchée.
- ▶ Pour détecter ces mouvements, le système IoT dispose de capteurs gyroscope et accéléromètre.
- ▶ Le contrôleur de service utilise l'API WebSocket pour envoyer des données en temps réel vers le cloud, ce qui est utile dans les applications en temps réel en raison de sa faible surcharge.
- ▶ Un service WebSocket basé sur le cloud récupère les données en temps réel des dispositifs IoT et les stocke dans une base de données.
- ▶ La fréquence de détection des données est élevée et les données détectées collectées sont stockées sur le cloud car elles sont volumineuses.
- ▶ Les données sont analysées dans le cloud et les actions de contrôle sont activées à l'aide d'une application mobile ou d'une application Web en fonction des résultats de l'analyse.



# Déploiement de niveau 4

04

LEVEL





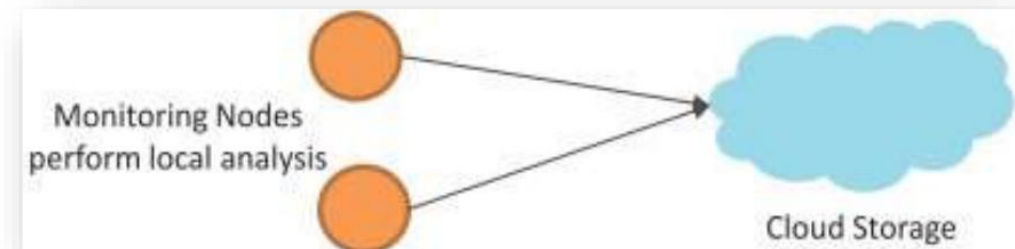
# Déploiement de niveau 4

## LEVEL 04

- ▶ Possède plusieurs nœuds qui effectuent une analyse locale.
- ▶ Les données sont stockées dans le cloud et l'application est basée sur le cloud.
- ▶ Contient des nœuds observateurs locaux et basés sur le cloud qui peuvent s'abonner et recevoir des informations collectées dans le cloud à partir des dispositifs IoT.
- ▶ Dédié pour les solutions nécessitant plusieurs nœuds.
- ▶ Les données impliquées sont volumineuses et les exigences d'analyse sont intensives en calcul.

# Déploiement de niveau 4

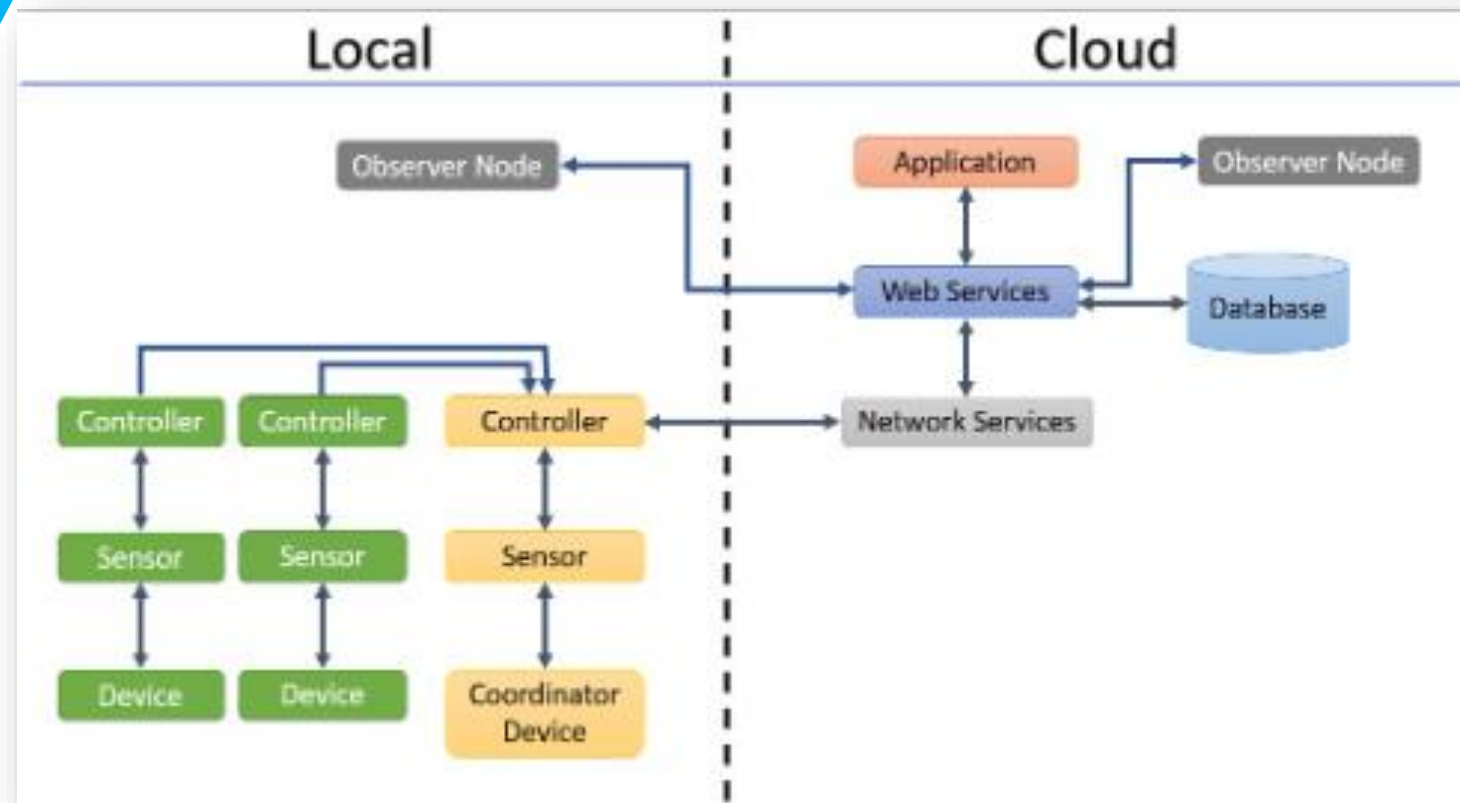
- ▶ Le système de surveillance du bruit basé sur l'IoT.
- ▶ Plusieurs nœuds/dispositifs sont répartis sur divers endroits pour détecter le bruit dans une région spécifique.
- ▶ Les capteurs sonores sont des exemples de nœuds/dispositifs dans ce contexte.
- ▶ Chaque nœud/appareil est autonome, avec son propre contrôleur de service de qui fournit des données au cloud pour le stockage et le traitement.
- ▶ Ce niveau comprend de nombreux capteurs, la collecte et l'analyse de données, ainsi qu'une application de contrôle et de surveillance.



# Déploiement de niveau 5

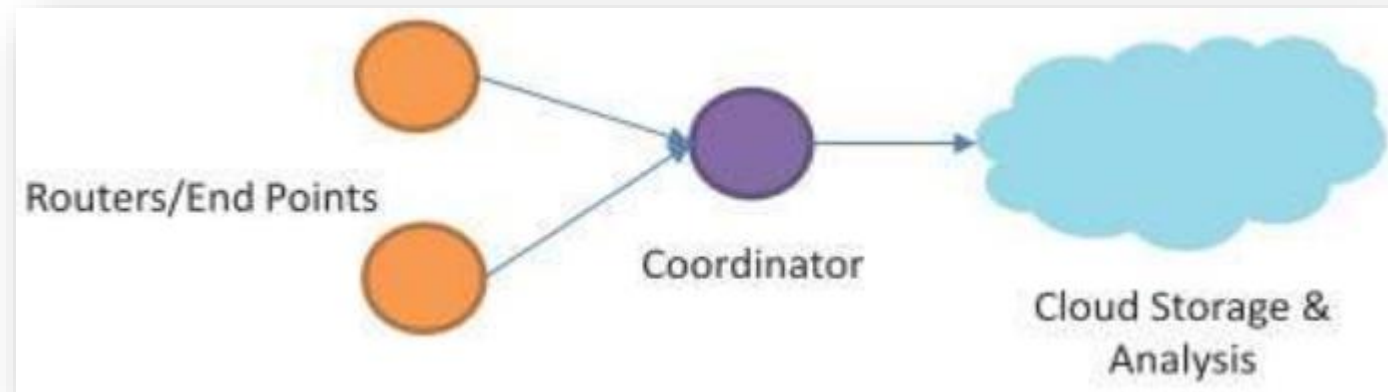
05

LEVEL



# Déploiement de niveau 5

- ▶ Un système IoT de niveau 5 comprend plusieurs nœuds finaux et un nœud coordinateur.
- ▶ Les nœuds finaux effectuent la détection et/ou l'actionnement.
- ▶ Le nœud coordinateur collecte les données des nœuds finaux et les envoie vers le cloud.
- ▶ Les données sont stockées et analysées dans le cloud, et l'application est basée sur le cloud.



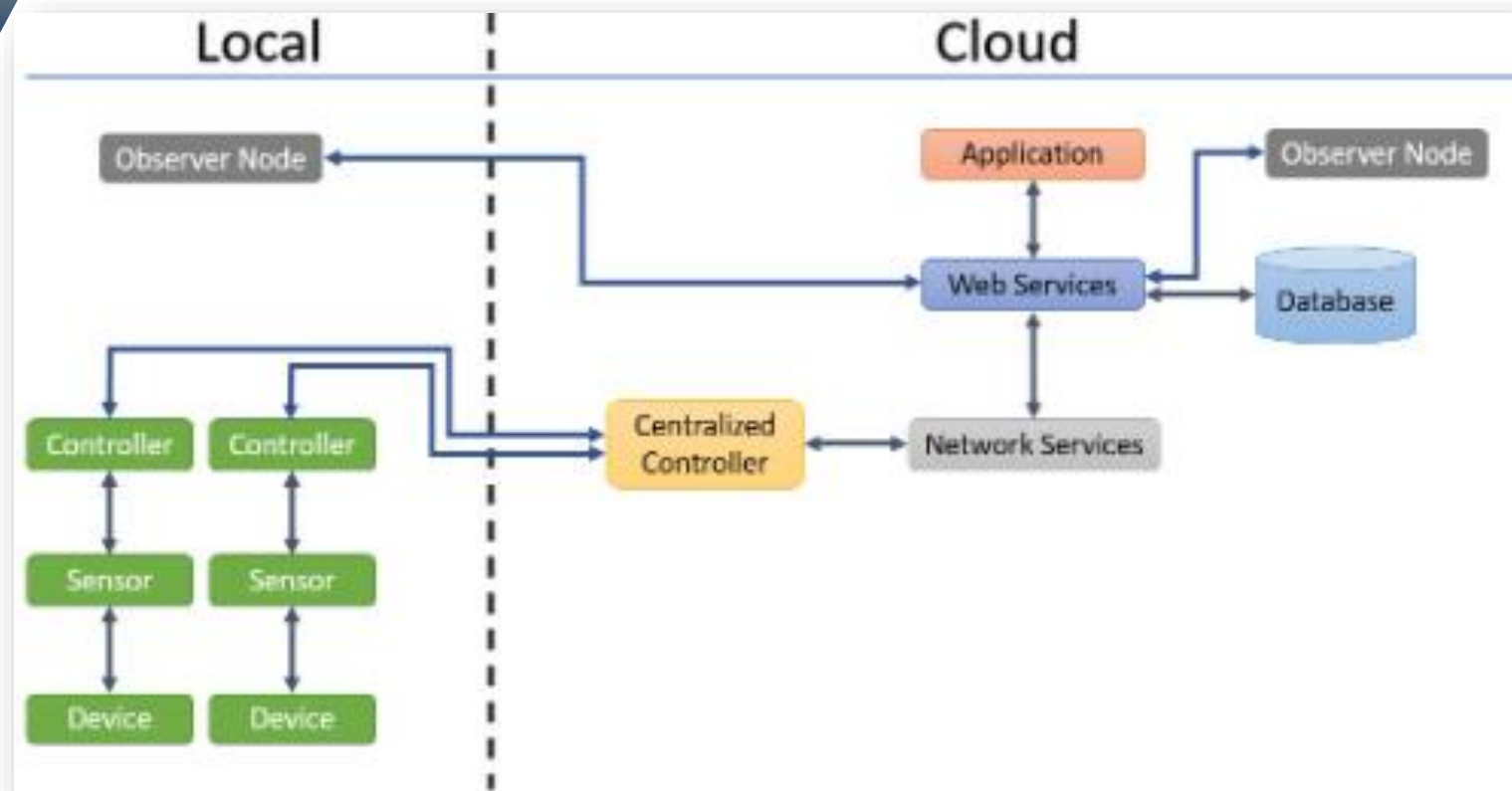
# Déploiement de niveau 5

- ▶ Idéal pour les solutions axées sur les réseaux sans fil avec des besoins importants en termes de données et d'analyses intensives.
- ▶ Un système de détection d'incendie en forêt.
  - ▶ Les dispositifs sont principalement utilisés pour détecter la température, l'humidité et les niveaux de CO<sub>2</sub>.
  - ▶ Ces nœuds détectent les données, le nœud coordinateur collecte ces données et le service de contrôleur sur le nœud coordinateur les migre vers le cloud.
  - ▶ Le nœud coordinateur sert de portail vers le système IoT et fournit un accès à Internet.
  - ▶ Le module d'analyse peut être utilisé pour prédire/générer des résultats à partir des données stockées sur le cloud.
  - ▶ La collecte et l'analyse des données sont effectuées au niveau du cloud.

# Déploiement de niveau 6

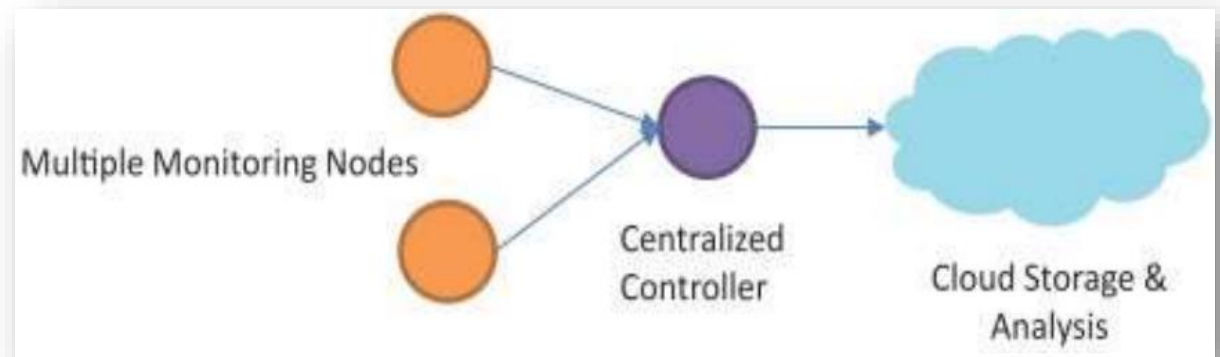
06

LEVEL



# Déploiement de niveau 6

- ▶ Il comporte plusieurs nœuds terminaux distincts
- ▶ Utilisés pour la surveillance et/ou le transfert de données vers le cloud.
- ▶ La base de données est construite sur le Cloud.
- ▶ La partie analytique effectue son activité et stocke les données dans le cloud et affiche, également, les résultats basés sur le cloud.
- ▶ Le contrôleur centralisé connaît l'état de tous les nœuds et délivre des signaux de commande aux nœuds.



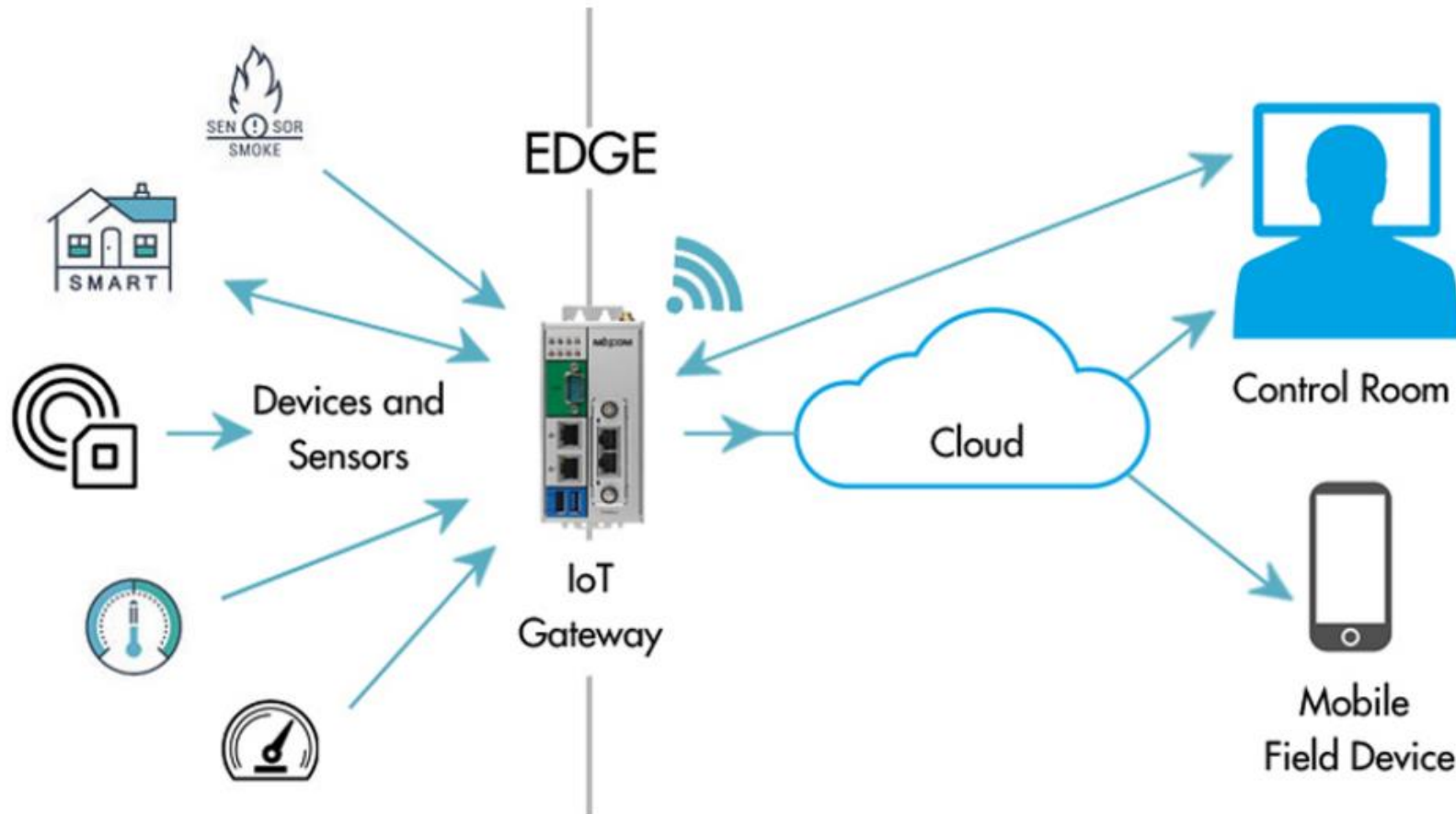
# Déploiement de niveau 6

- ▶ Système de surveillance météorologique.
- ▶ De nombreux capteurs de température, d'humidité, etc. que le système dispose.
- ▶ Ces nœuds sont installés à divers endroits et sont envoyés via l'API basée sur WebSocket (exemple) vers un stockage basé sur le cloud en temps réel.
- ▶ Toute mise à jour ou modification de nœud est effectuée via le contrôleur centralisé.
- ▶ Le module d'analyse est utilisé pour prévoir/générer des rapports en utilisant les données stockées sur le cloud.

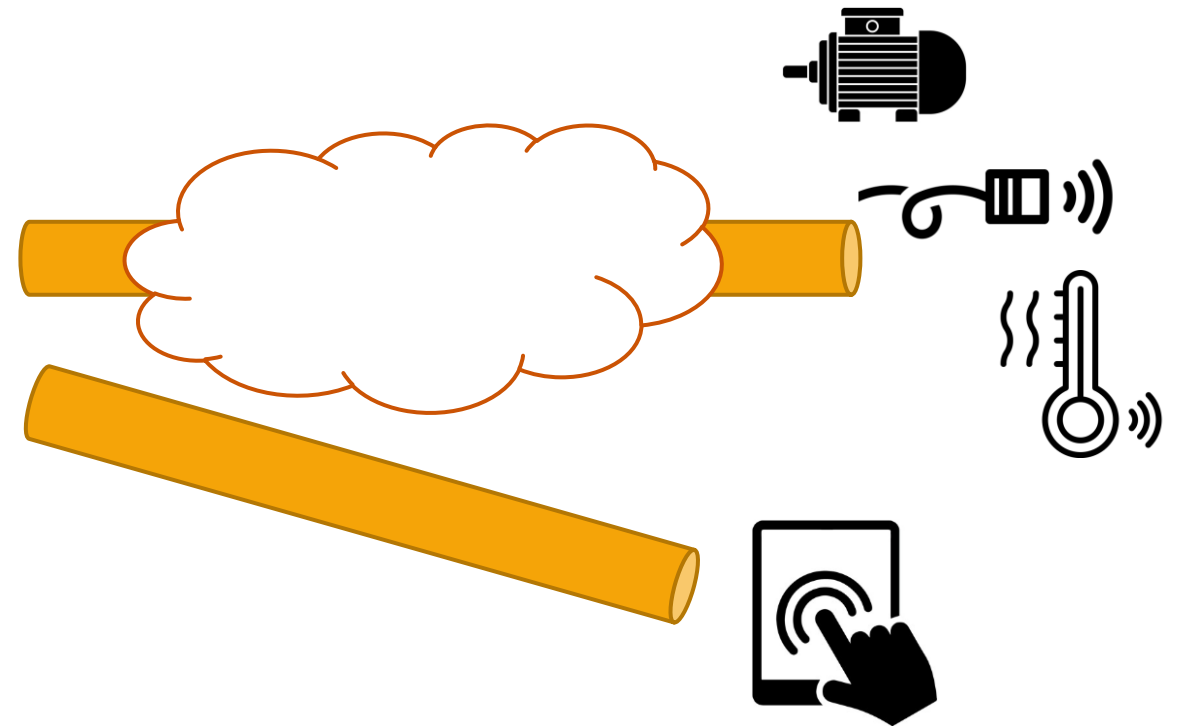


# Cours 5

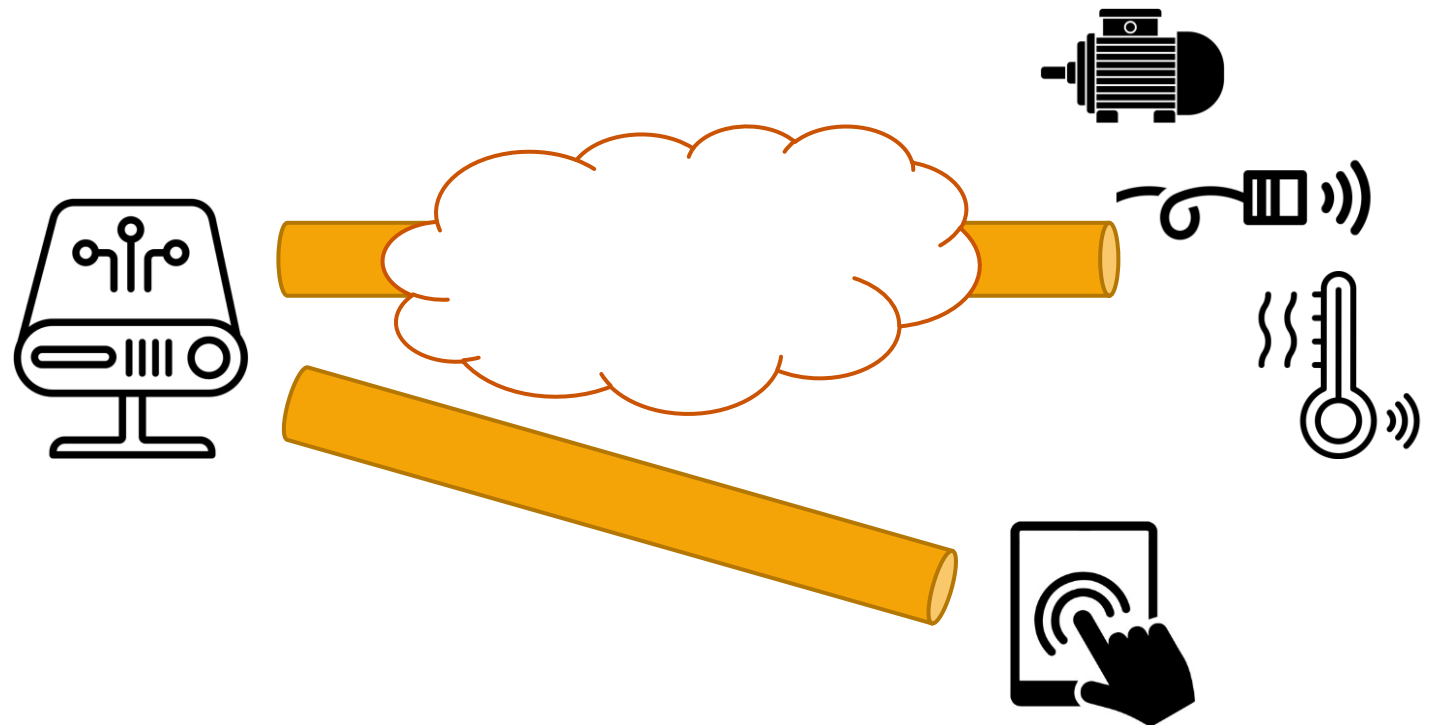
# Arduino, capteurs et Actionneurs, Éléments d'IoT



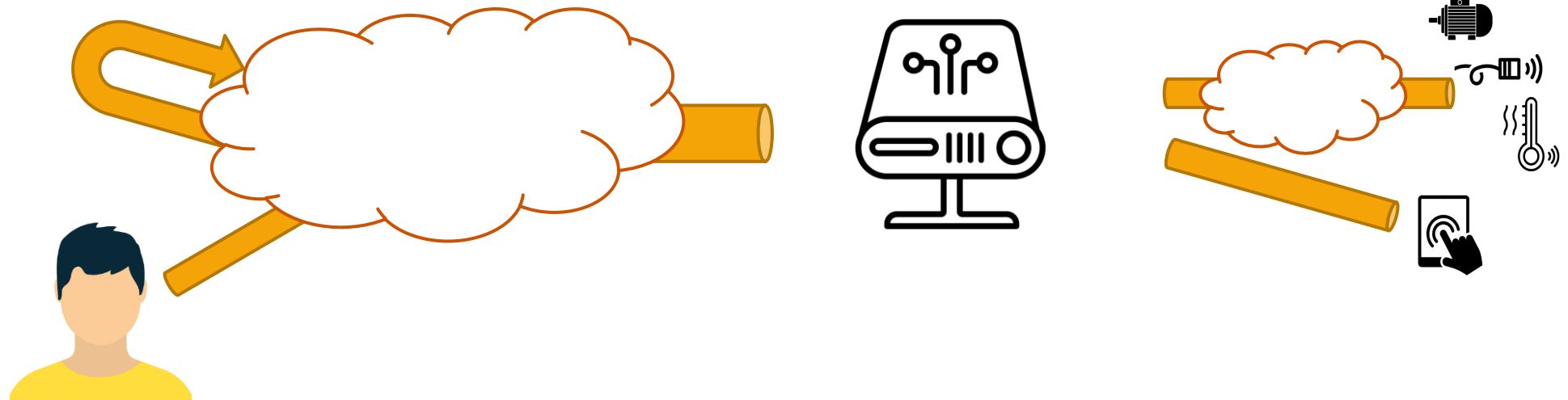
# Arduino, capteurs et Actionneurs, Éléments d'IoT



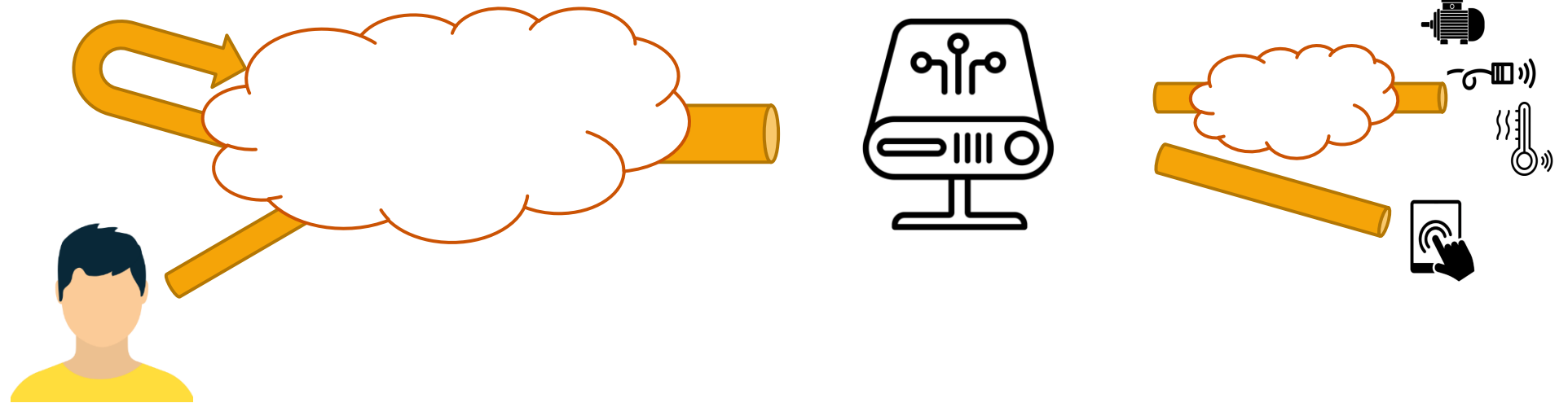
# Arduino, capteurs et Actionneurs, Éléments d'IoT



# Arduino, capteurs et Actionneurs, Éléments d'IoT

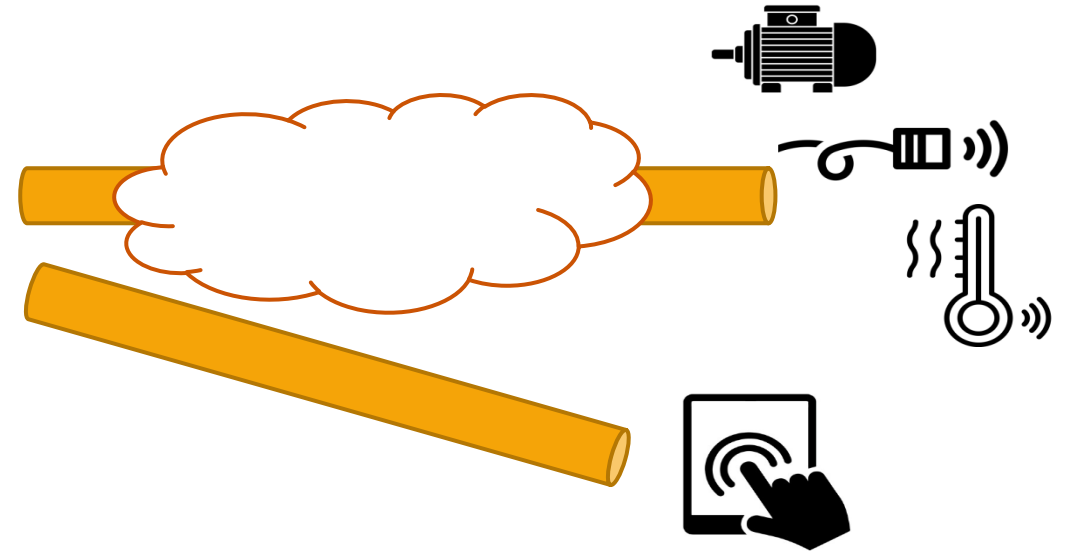


# Architecture IoT



# De l'objet vers la passerelle

- ▶ Capteur
  - ▶ Grandeur Physique
  - ▶ ....
- ▶ Actionneur
  - ▶ Mouvement mécanique
  - ▶ ....

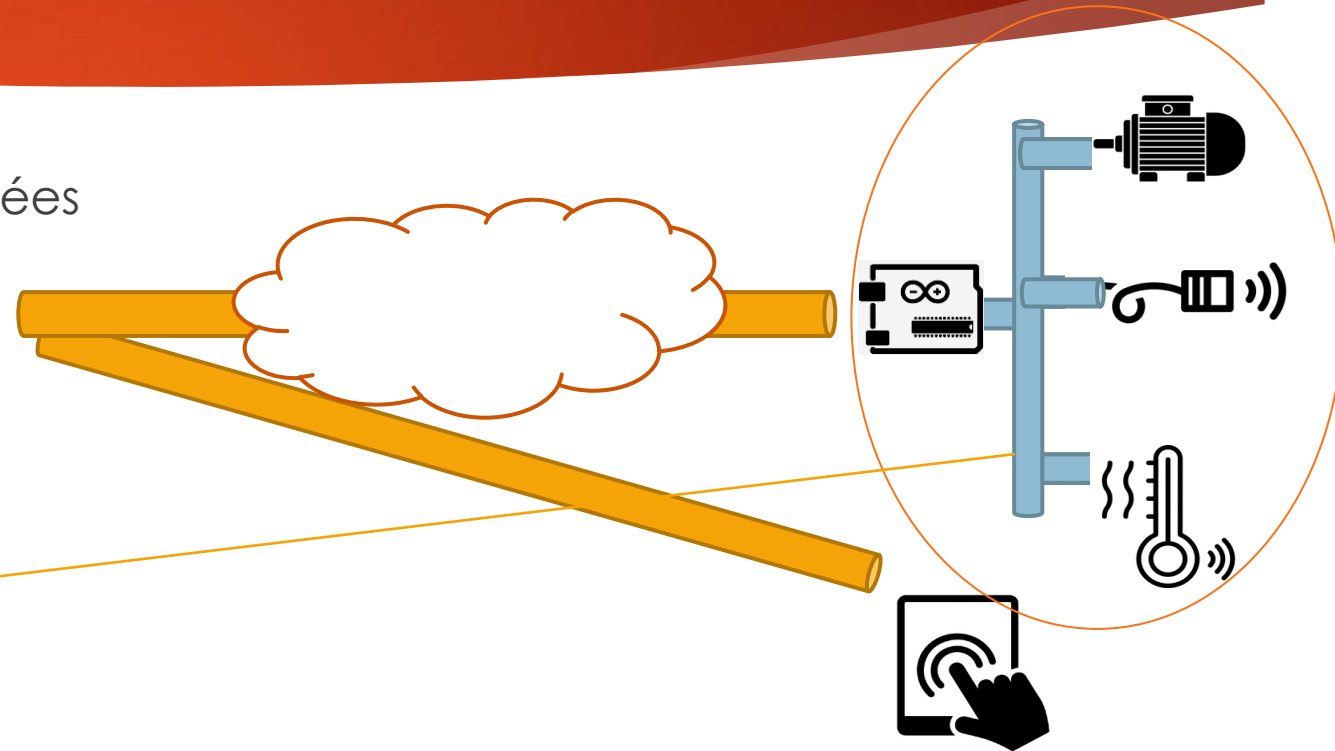


# De l'objet vers la passerelle

- Premières communications des données pour la constitution d'un nœud:

- Protocol de couche 2

- Serial
- I2C
- UART
- CAN
- SPI
- RFID
- RS-232





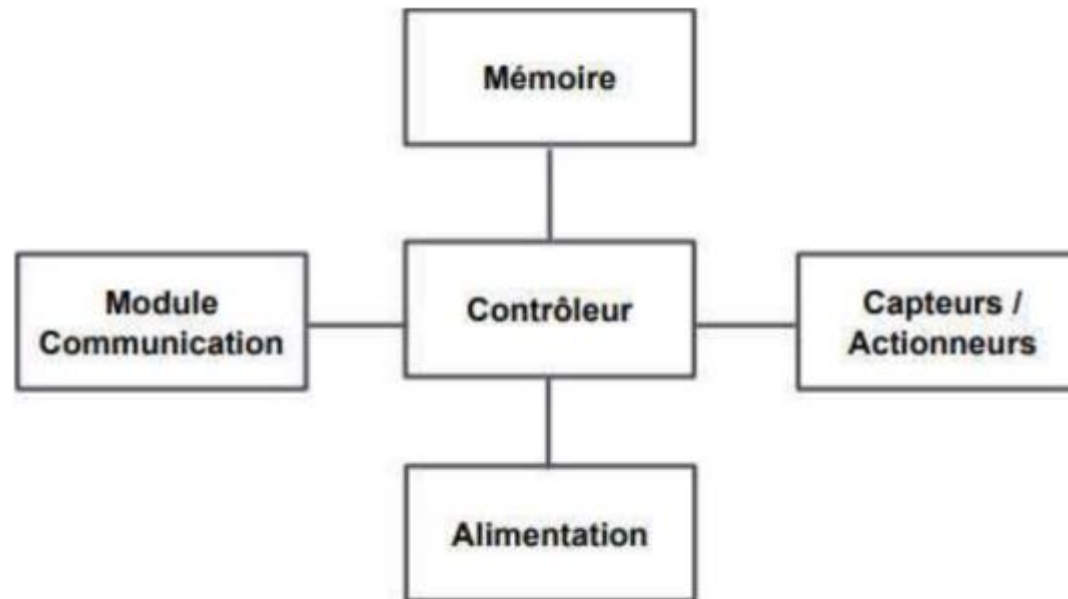
# Microcontrôleurs

- ▶ Un microcontrôleur est un circuit intégré programmé pour avoir le contrôle prévu sur une tâche arithmétique ou logique.
- ▶ Les microcontrôleurs sont moins chers que d'avoir un microprocesseur à part entière, et consomment moins d'énergie.
- ▶ Les microcontrôleurs sont de petits ordinateurs indépendants, donc assez rapides.
- ▶ En effet, les microcontrôleurs ne sont pas des systèmes de multiprogrammation.
- ▶ Néanmoins, un microcontrôleur comprend presque tous les principaux composants d'une architecture informatique. En occurrence :
  - ▶ Mémoire
  - ▶ Ports d'E/S
  - ▶ Ports sérieMinuteries
  - ▶ ADC (convertisseurs analogique-numérique)
  - ▶ DAC (convertisseurs numérique-analogique)
  - ▶ Contrôleur d'interruption



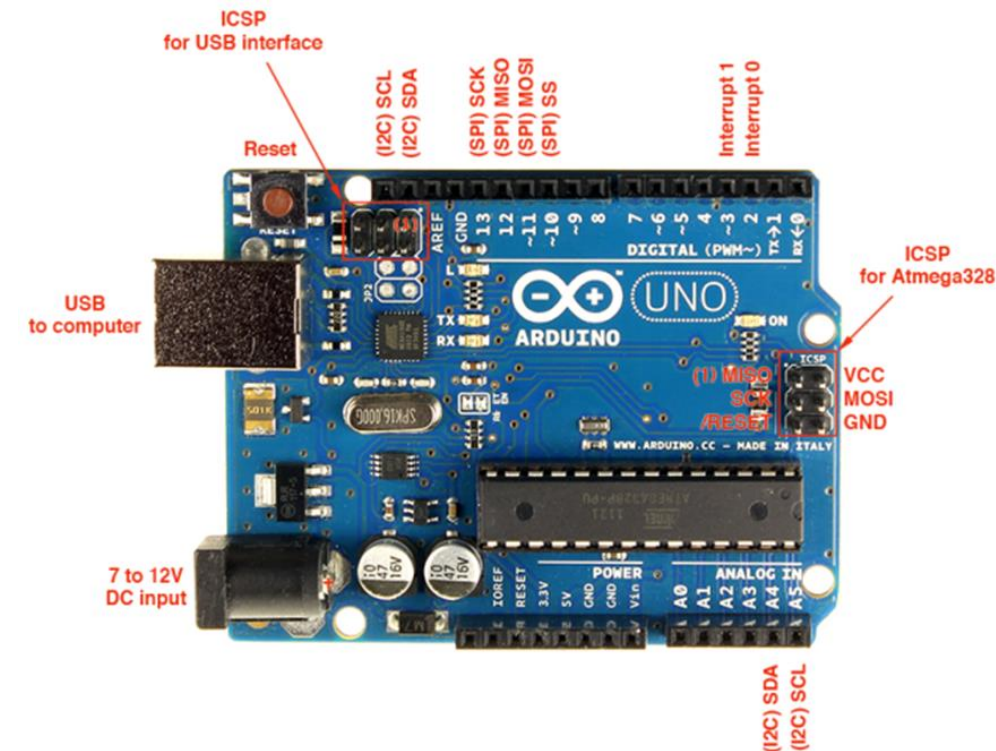
# Microcontrôleurs

- ▶ Avec les actionneurs et les capteurs, on associe des cartes à microcontrôleurs pour extraire les informations utiles ou commander les actionneurs

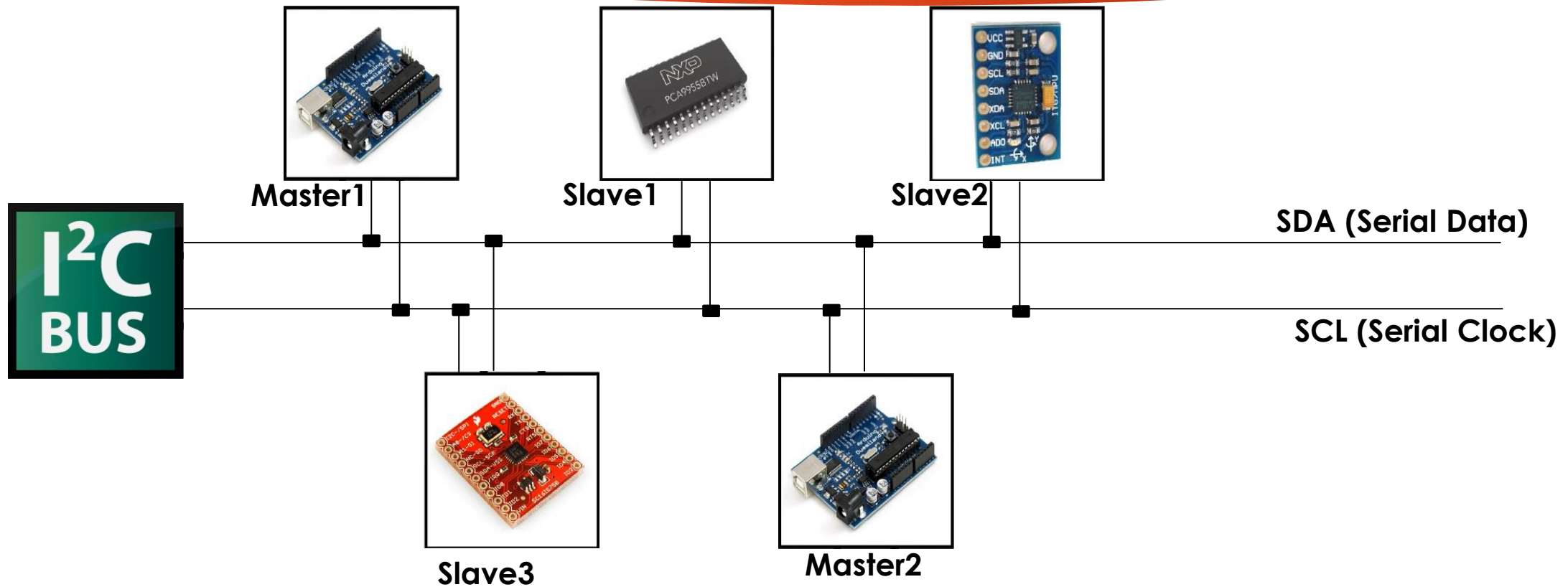


# Arduino

- ▶ Arduino représente une solution complète avec :
- ▶ Basé sur le micro-contrôleur ATmega + 4K à 256 de RAM
- ▶ SPI, I<sup>2</sup>C, UART et un port ADC de 8 à 10 canaux
- ▶ De 23 à 53 lignes d'entrée-sortie universelles
  - ▶ Dans nos activités de travaux pratique, la communication I2C est appliquée a travers les 2 pins qu'arduino offre A4 et A5 pour SDA et SDA respectivement.
  - ▶ Arduino peut représenté la passerelle vers le réseau Internet en lui ajoutant un shield wifi ou ethernet.



# Le protocole I2C



# I2C on Arduino

i2c-master-demo

## Master

```
// Include Arduino Wire library for I2C
#include <Wire.h>

// Define Slave I2C Address
#define SLAVE_ADDR 9

// Define Slave answer size
#define ANSWERSIZE 5

void setup() {

    // Initialize I2C communications as Master
    Wire.begin();

    // Setup serial monitor
    Serial.begin(9600);
    Serial.println("I2C Master Demonstration");
}

void loop() {
```

i2c-slave-demo

## Slave

```
// Include Arduino Wire library for I2C
#include <Wire.h>

// Define Slave I2C Address
#define SLAVE_ADDR 9

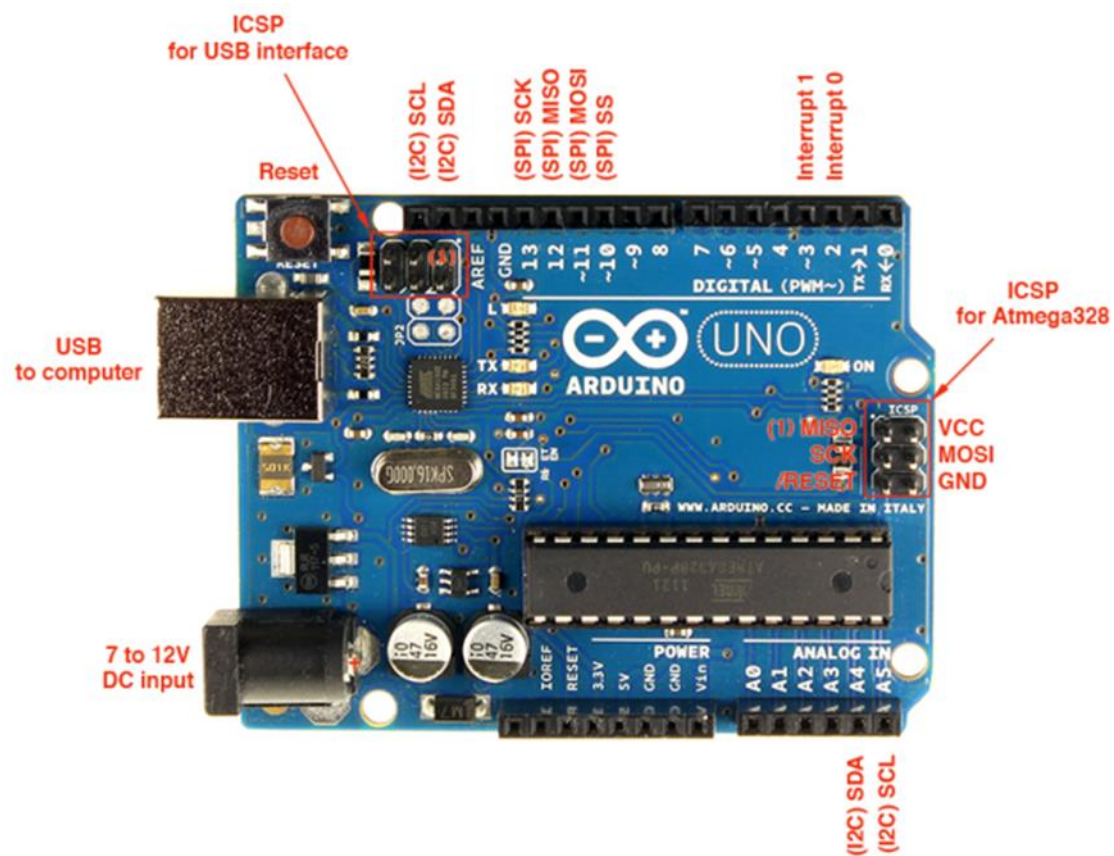
// Define Slave answer size
#define ANSWERSIZE 5

// Define string with response to Master
String answer = "Hello";

void setup() {

    // Initialize I2C communications as Slave
    Wire.begin(SLAVE_ADDR);

    // Function to run when data requested from master
    Wire.onRequest(requestEvent);
```





# RFID

- ▶ RFID ( Radio Frequency Identification ) : procédé permettant de récupérer des données à distances grâce à un système composé :
  - ▶ Une étiquette radio contenant des informations
  - ▶ Un lecteur permettant de récupérer les informations d'une étiquette radio à distance
- ▶ Fréquences utilisées
  - ▶ Basses fréquences (LF) : 125 à 134 Khz
  - ▶ Hautes fréquences (HF) : 13,56 Mhz
  - ▶ Ultra-Hautes fréquences (UHF) : 850 à 950 Mhz
- ▶ [\(101\) Arduino RFID sensor - read and write RFID RC522 - LCD16\\*2 character - rfid-rc522 - YouTube](#)



# RFID

Reader



Tag

Active

Passive

Semi Passive





# RFID, use case



# RFID, use case



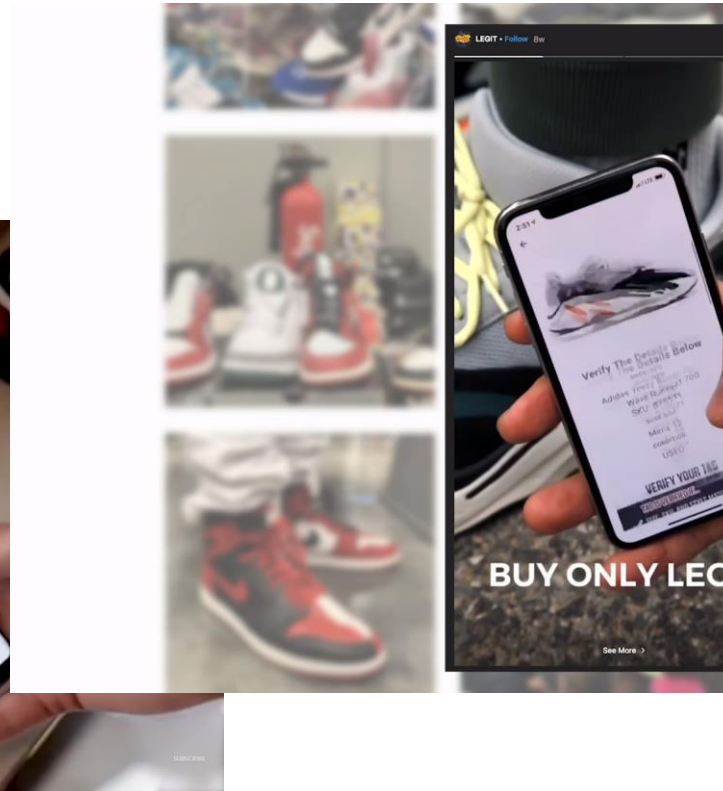
# RFID, use case



# NFC

- ▶ L'interface NFC (Near Field Communication) : une technologie dérivée de RFID
- ▶ Permettant une communication à très faible distance entre deux objets
- ▶ Bande : 13,56 Mhz
- ▶ Distance : 0 et 20 cm
- ▶ Débit : 106 Kb/s et 424 Kb/s
- ▶ Fréquence utilisée (HF) : 13,56 Mhz

# NFC, use case





# NFC, use case

