Internet des objets

Introduction

- IoT = Internet of Things = Internet des objets
- **Internet des objets** est considéré comme un vaste réseau des objets.
- Objet: Informatique embarquée légère optimisée pour une grande autonomie généralement associée à un faible encombrement et de faibles coûts.
- Ces objets peuvent être: appareils électroménagers, instruments de mesure, robots, serrures, machines-outils, bennes à ordures, drones, jouets, montres, véhicules, etc.
- IoT est caractérisé par:

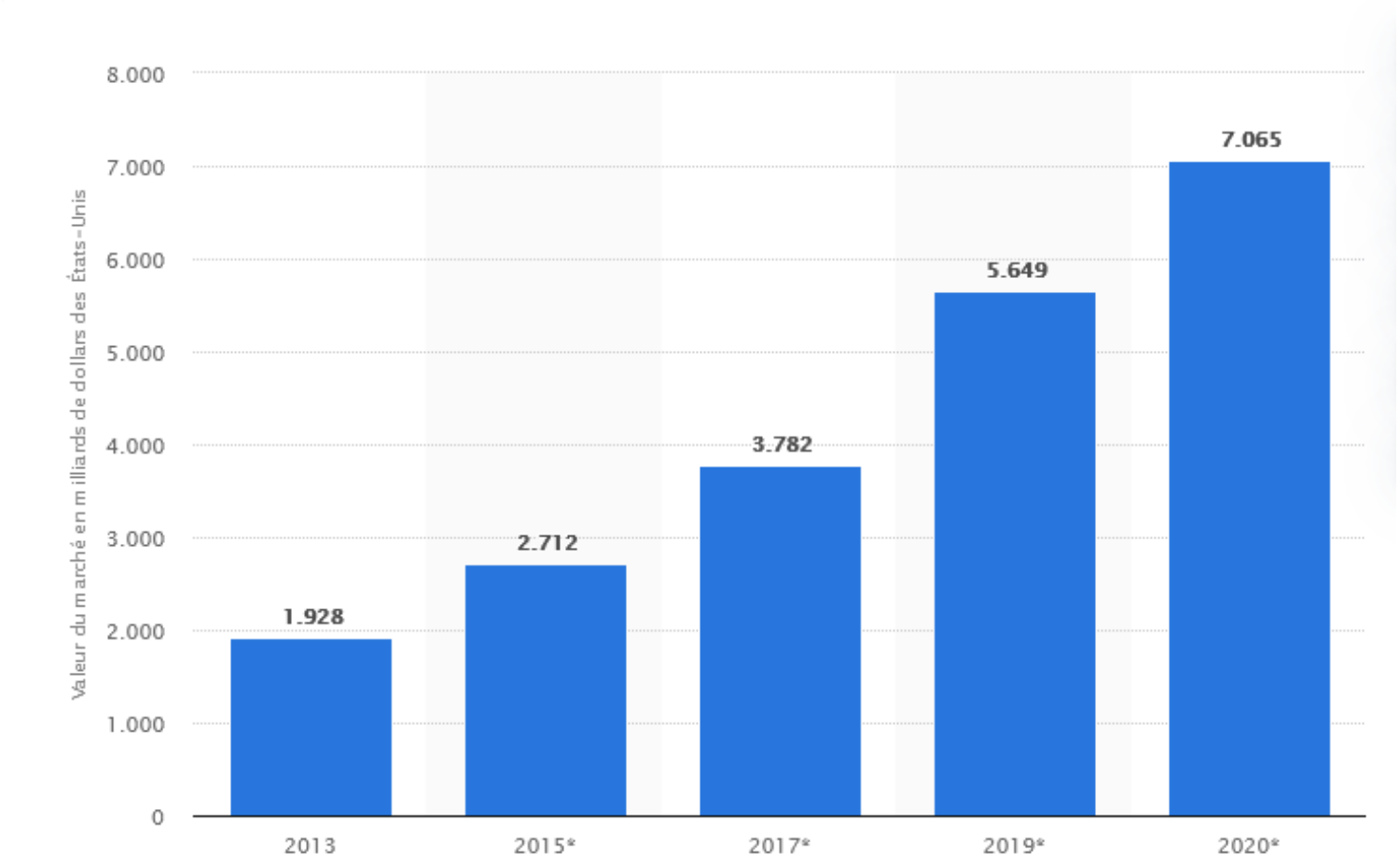
✓Mise en réseau de plusieurs objets connectés (capteurs)

✓Acquisition de signaux issus du monde physique

✓Action sur le monde physique (actuateurs)

✓Problématique de sécurité accrue

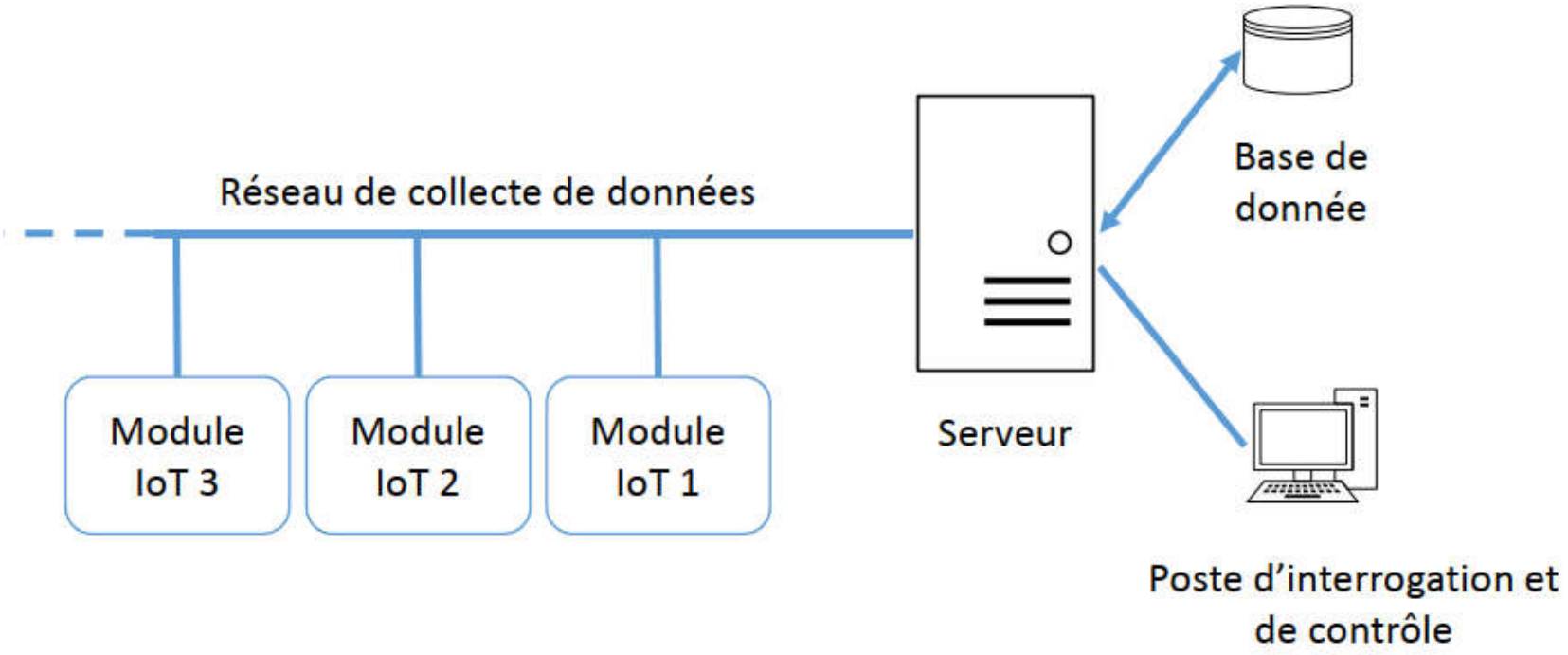
Marché de l’IoT(Modiale)



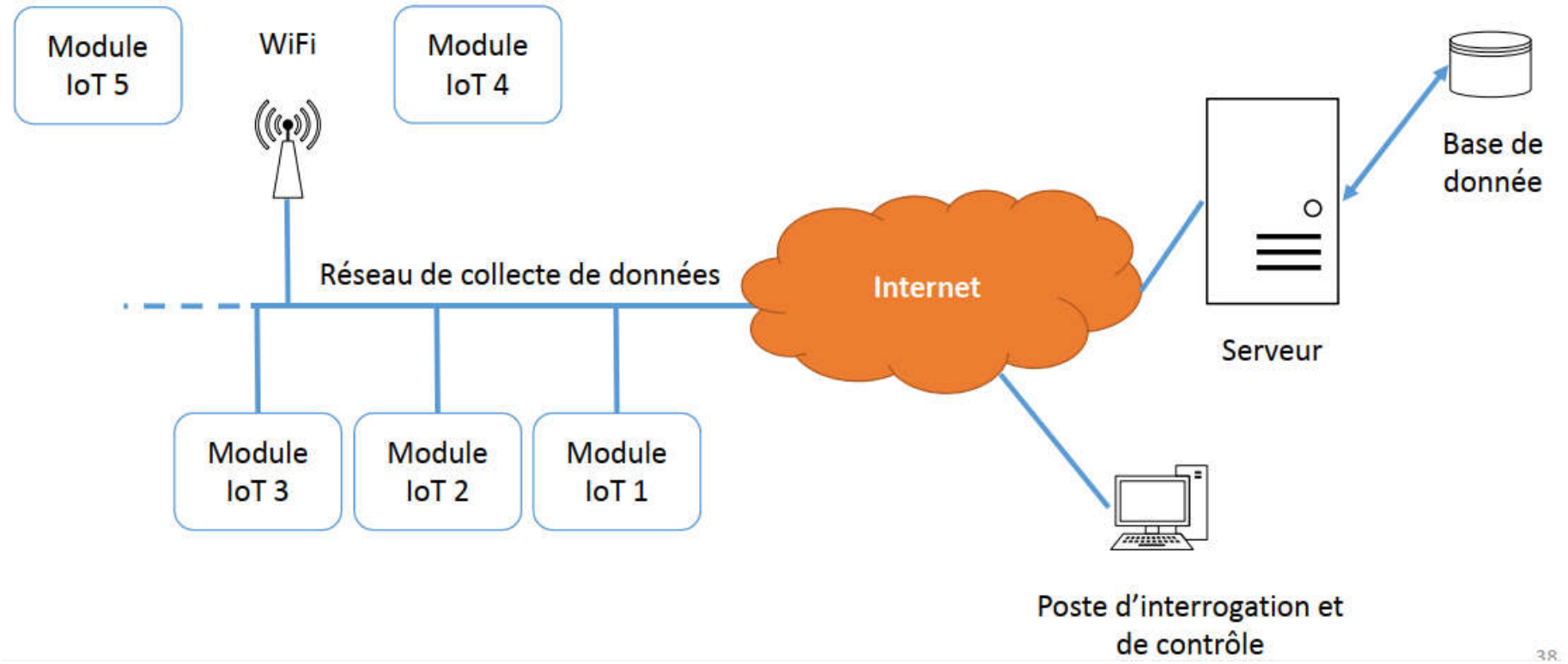
Domaines d’application

- **Ville intelligente :** circulation routière intelligente, transports intelligents, collecte des déchets, cartographies diverses (bruit, énergie, etc.)
- **Environnements intelligents :** prédiction des séismes, détection d’incendies, qualité de l’air, etc.
- **Sécurité et gestion des urgences :** radiations, attentats, explosions.
- **Logistique**
- **Contrôle industriel :** mesure, pronostic et prédiction des pannes, dépannage à distance.
- **Santé :** suivi des paramètres biologiques à distance.
- **Agriculture intelligente, domotique,** etc.

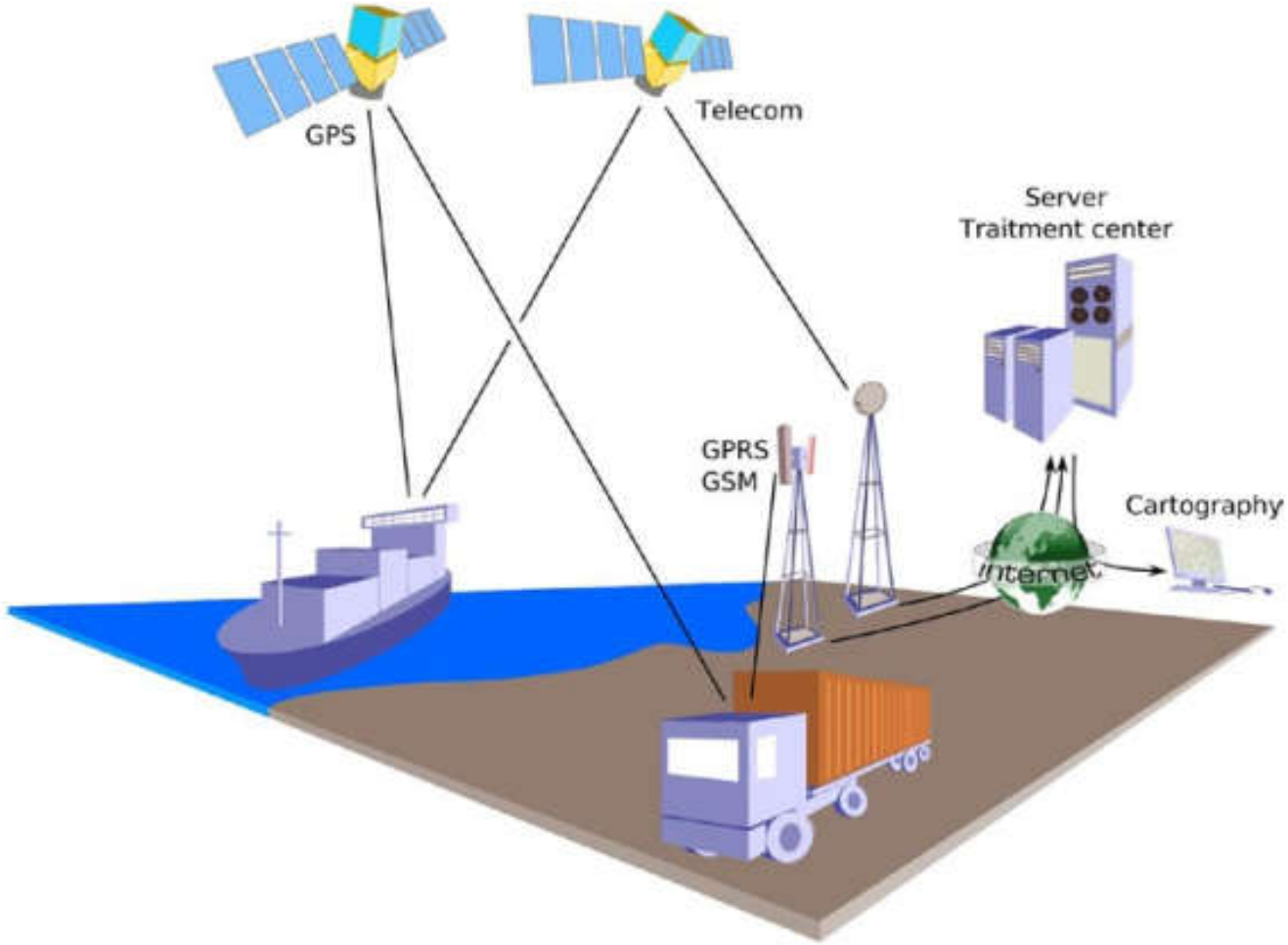
Infrastructure système pour IoT



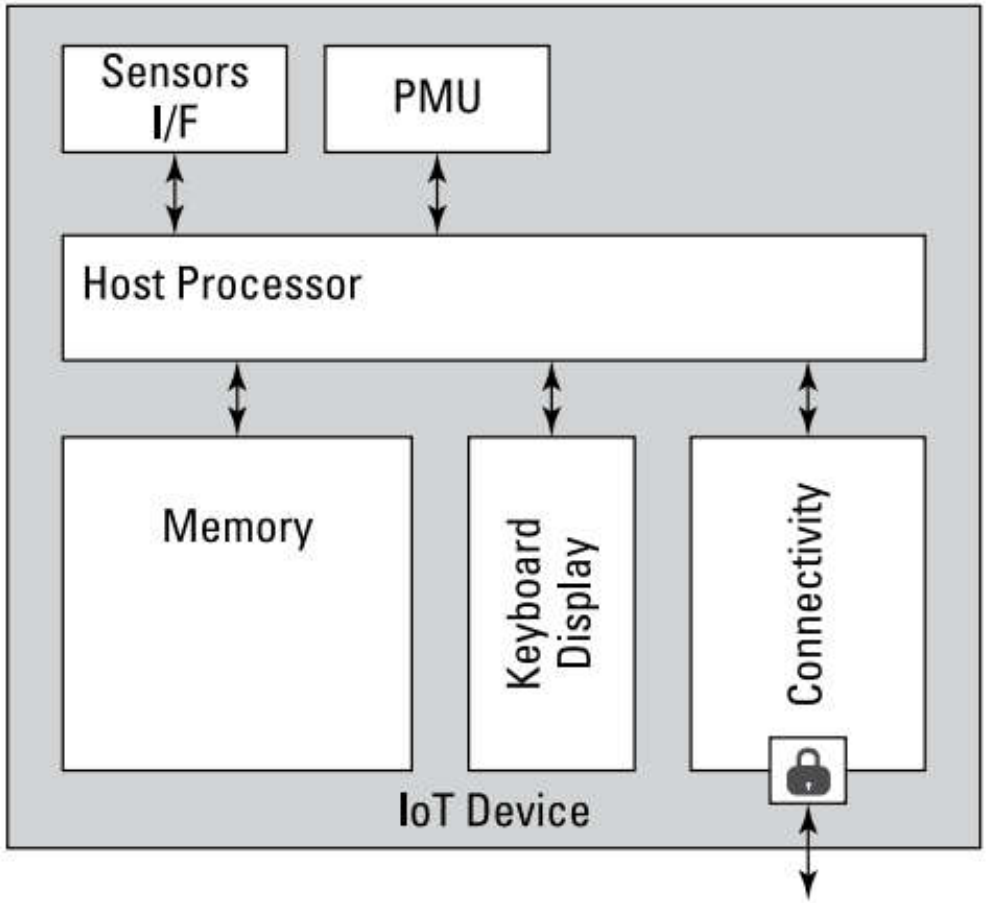
Infrastructure système pour IoT



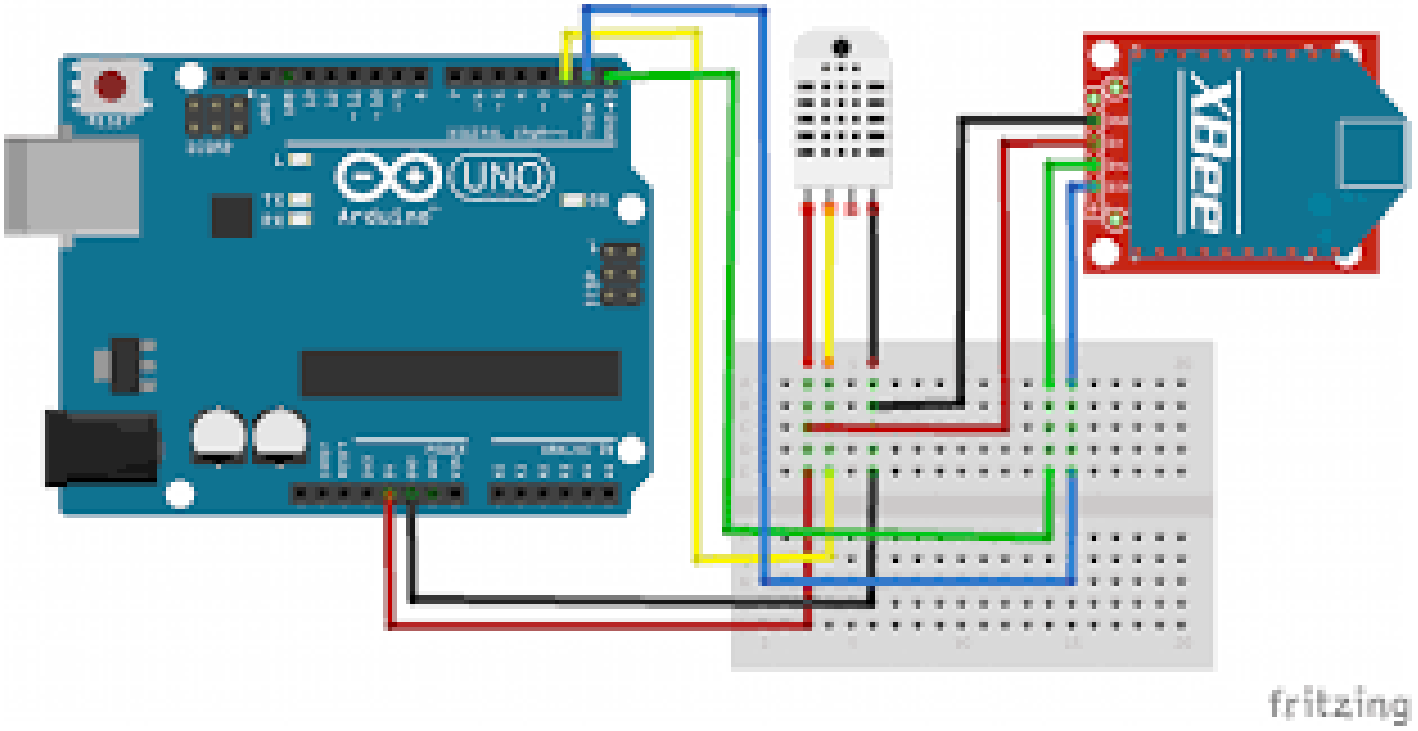
Infrastructure système pour IoT(Exemple)



Architecture d’un objet IoT



Architecture d’un objet IoT(Exemple)



Protocoles de communication

- Pour se communiquer, les objets utilisent généralement des protocoles:

✓HTTP **Hypertext Transfer Protocol)**

✓MQTT (**Message Queue Telemetry Transport)**

✓UPnP (**Universal Plug and Play)**

✓CoAP (**Constrained Application Protocol)**

✓XMPP(**Extensible Messaging and Presence Protocol)**

Protocole HTTP

- Protocole sans état utilisé sur l'architecture Client-Serveur:
- Requête: une méthode (GET, POST, …), une ressource, quelques en-têtes, contenus (optionnel).
- Réponse: état(200, 404,…), quelques en-têtes, contenus (optionnel).

Protocole HTTP: méthode GET

```
void doHttpGet()
{
  // Prepare data or parameters that need to be posted to server
  String requestData = "requestVar=test";
  // Check if a connection to server:port was made
  if (client.connect(server, port))
  {
    Serial.println("[INFO] Server Connected - HTTP GET Started");
    // Make HTTP GET request
    client.println("GET /get?" + requestData + " HTTP/1.1");
    client.println("Host: " + String(server));
    client.println("Connection: close");
    client.println();
    Serial.println("[INFO] HTTP GET Completed");
  }
}
```

Protocole HTTP: méthode POST

```
void doHttpPost()
{
  // Prepare data or parameters that need to be posted to server
  String requestData = "requestData={\"requestVar:test\"}";
  // Check if a connection to server:port was made
  if (client.connect(server, port))
  {
    Serial.println("[INFO] Server Connected - HTTP POST Started");
    // Make HTTP POST request
    client.println("POST /post HTTP/1.1");
    client.println("Host: " + String(server));
    client.println("User-Agent: Arduino/1.0");
    client.println("Connection: close");
    client.println("Content-Type: application/x-www-form-urlencoded;");
    client.print("Content-Length: ");
    client.println(requestData.length());
    client.println();
    client.println(requestData);
    Serial.println("[INFO] HTTP POST Completed");
  }
}
```

Protocole HTTP: méthode setup

```
void setup() {
  // Initialize serial port
  Serial.begin(9600);
  // Connect Arduino to internet
  connectToInternet();
  // Make HTTP GET request
  doHttpGet();
}
```

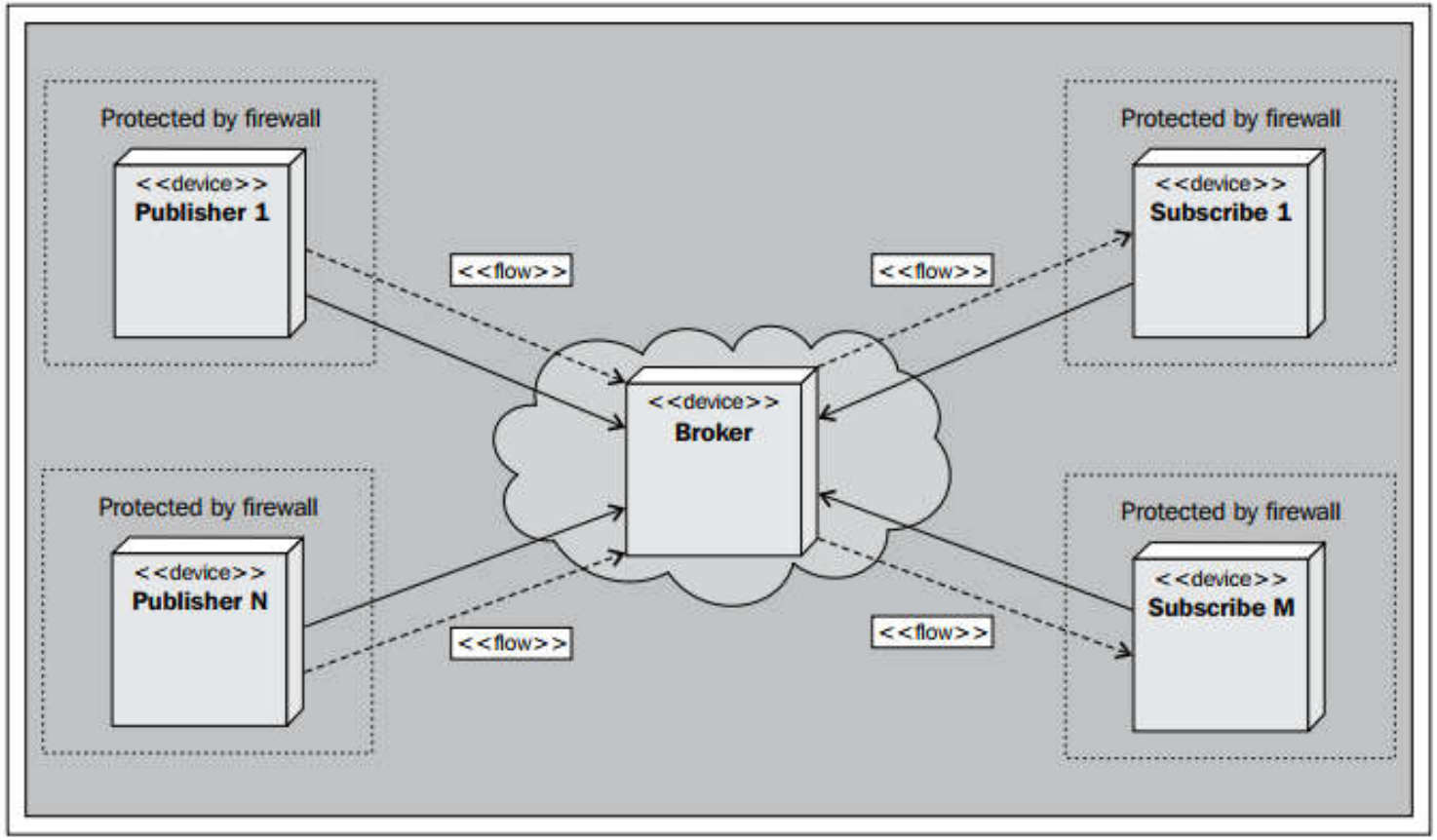
Protocole HTTP: méthode loop

```
void loop() {
  if (client.available())
  {
    Serial.println("[INFO] HTTP Response");
    // Read available incoming bytes from the server and print
    while (client.available())
    {
    char c = client.read();
    Serial.write(c);
    }
  }
  // If the server:port has disconnected, then stop the client
  if (!client.connected())
  {
    Serial.println();
    Serial.println("[INFO] Disconnecting From Server");
    client.stop();
  }
}
```

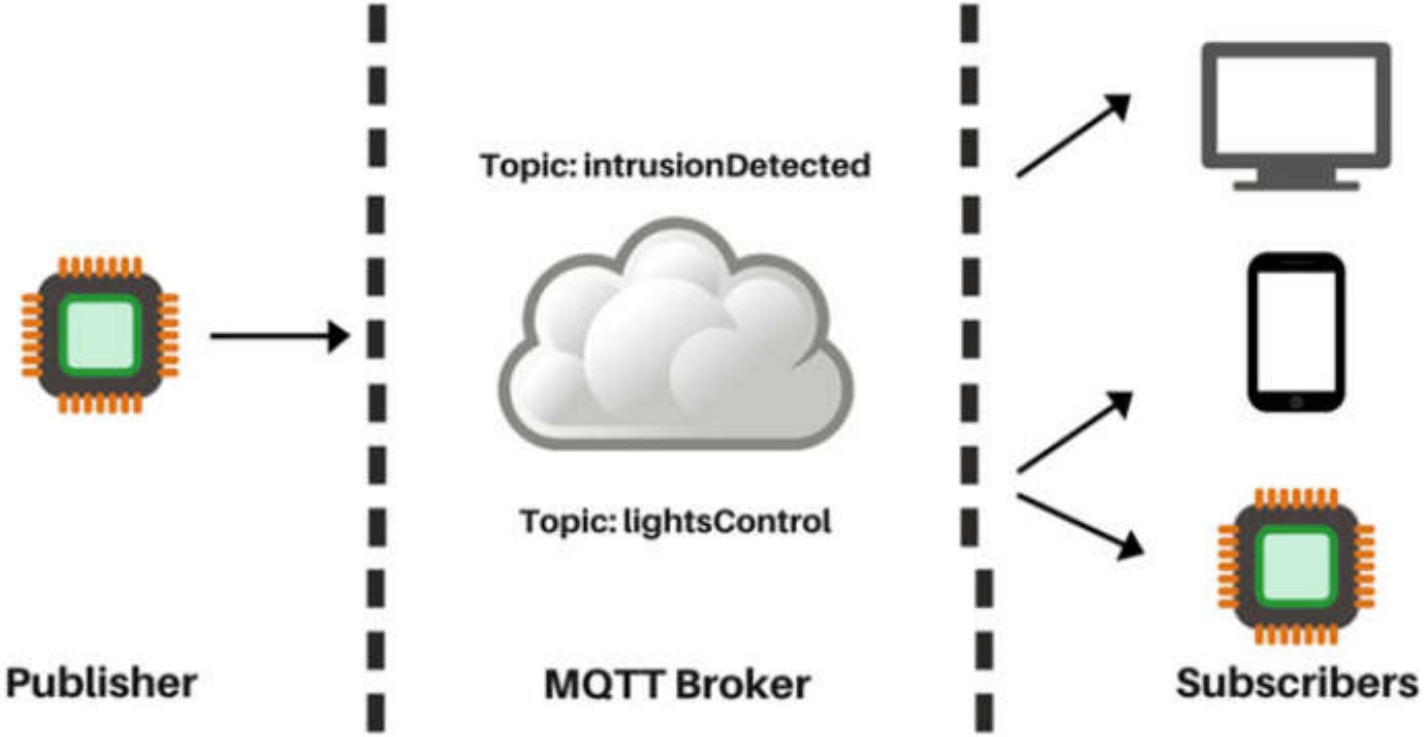
Protocole MQTT

- Protocole léger utilisé pour la communication machine-à-machine (M2M)
- Adopte le modèle Publisher - Subscriber
- Publisher: publie les données au serveur (Broker)
- Subscriber: reçoit les donnés auxquelles il souscrit.
- Publishers et Subscribers ne se connaissent, ils sont connectés au serveur. Ils peuvent être: capteurs, machines et applications mobile.
- Serveur(Broker) notifie les Subscribers sur les données qui viennent d'être publiées en utilisant le concept de TOPIC.

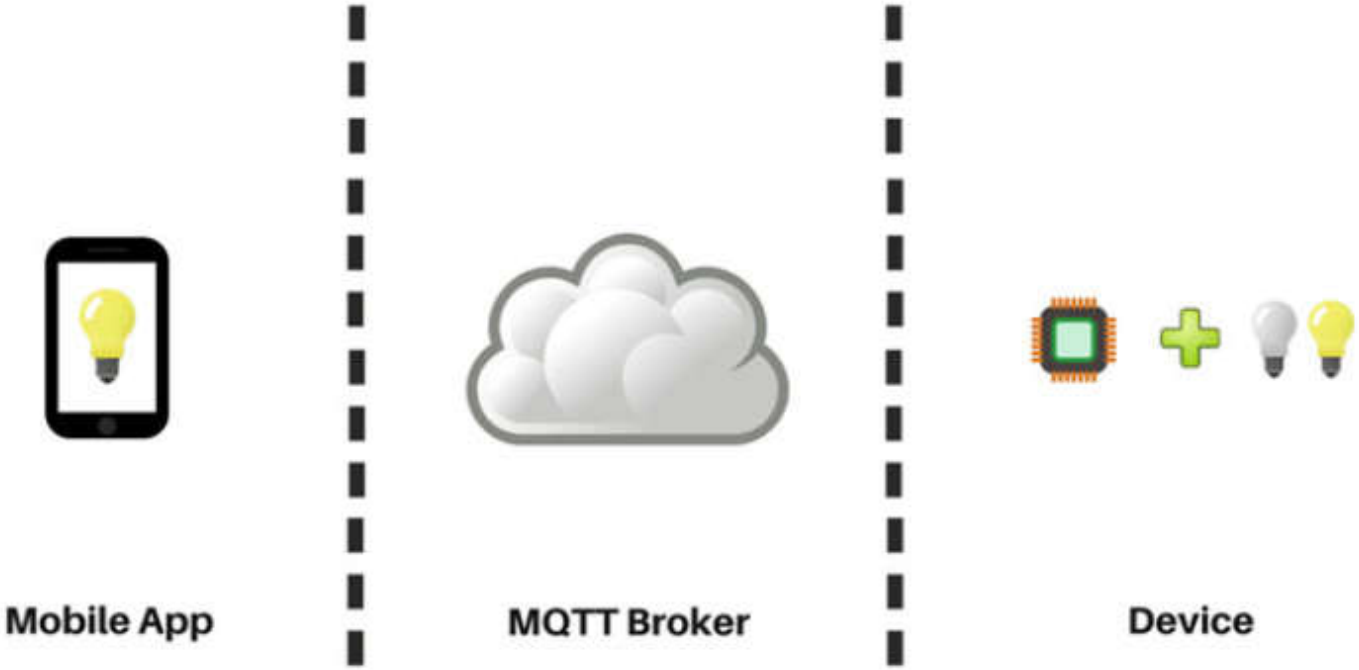
Protocole MQTT



Protocole MQTT: Système de détection d'intrusion



Protocole MQTT: Système de contrôle de lumières



Protocole MQTT: Code

```
#include <SPI.h>
#include <WiFi.h>
#include <PubSubClient.h>


// IP address of the MQTT broker
char server[] = {"x.y.z.t"};
int port = 1883
char topic[] = {"codifythings/testMessage"};
PubSubClient pubSubClient(server, 1883, callback, client);

void callback(char* topic, byte* payload, unsigned int length)
{
  // Print payload
  String payloadContent = String((char *)payload);
  Serial.println("[INFO] Payload: " + payloadContent);
}
```

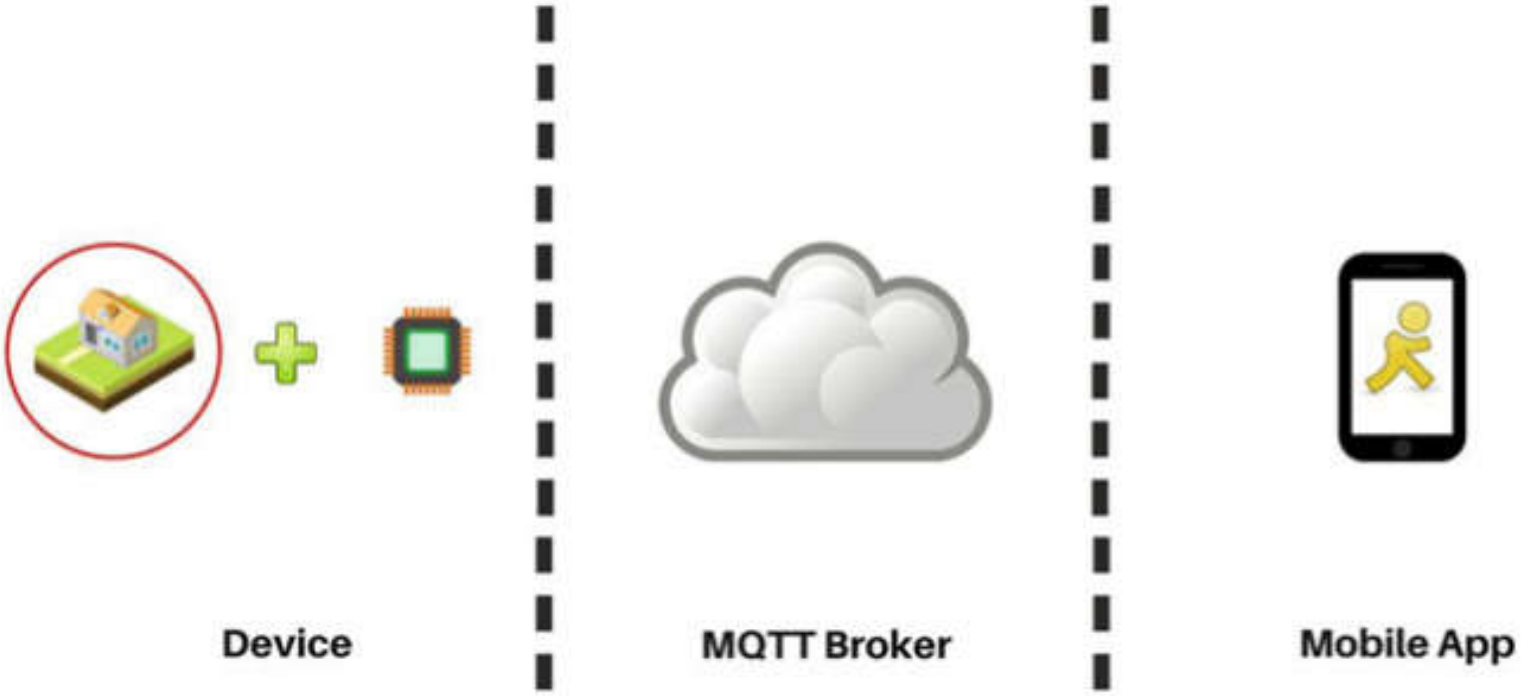
Protocole MQTT: Subscriber

```
void setup() {
  // Initialize serial port
  Serial.begin(9600);
  // Connect Arduino to internet
  connectToInternet();
  //Connect MQTT Broker
  Serial.println("[INFO] Connecting to MQTT Broker");
  if (pubSubClient.connect("arduinoClient"))
  {
    Serial.println("[INFO] Connection to MQTT Broker Successful");
    pubSubClient.subscribe(topic);
    Serial.println("[INFO] Successfully Subscribed to MQTT Topic ");
  }
}

void loop() {
  // Wait for messages from MQTT broker
  pubSubClient.loop();
}
```

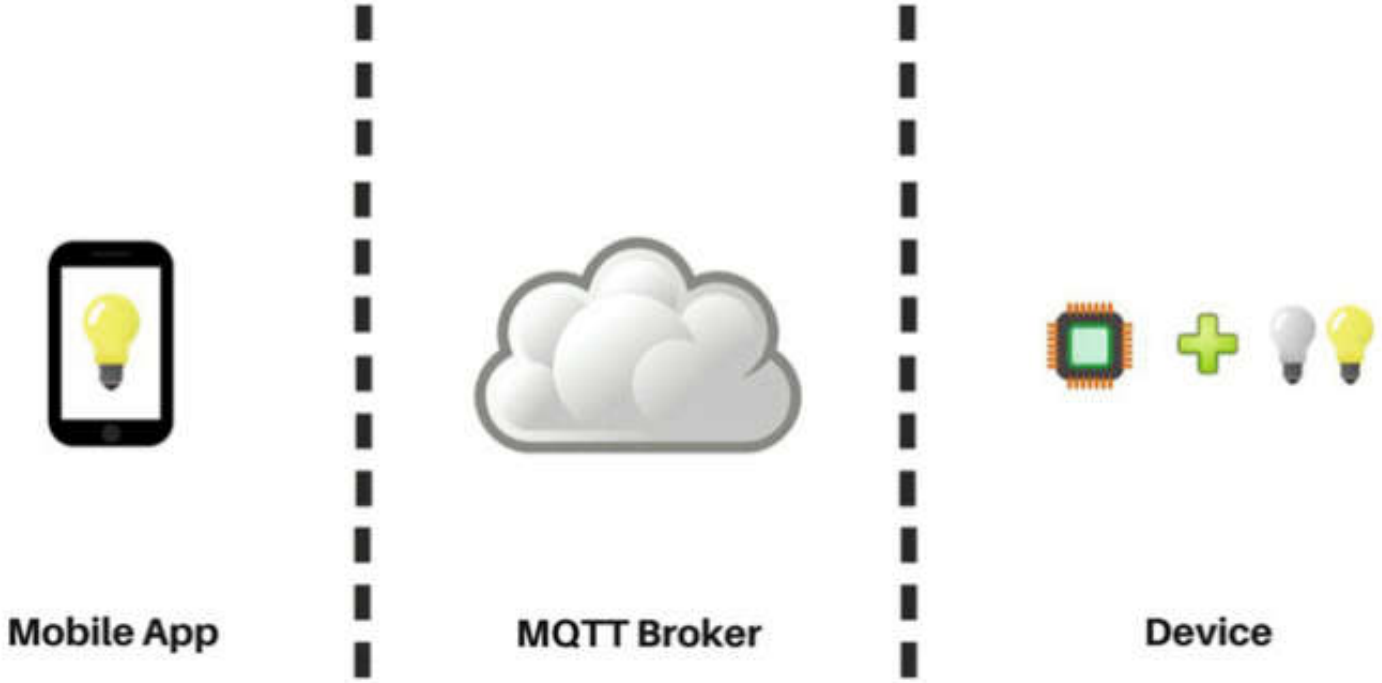
Protocole MQTT: Publisher

```
void setup() {
  // Initialize serial port
  Serial.begin(9600);
  // Connect Arduino to internet
  connectToInternet();
  //Connect MQTT Broker
  Serial.println("[INFO] Connecting to MQTT Broker");
  if (pubSubClient.connect("arduinoClient"))
  {
    Serial.println("[INFO] Publishing to MQTT Broker");
    pubSubClient.publish(topic, "Test Message");
  }
}

void loop() {
  // Wait for messages from MQTT broker
  pubSubClient.loop();
}
```

Pattern 1: Realtime Clients



Pattern 2: Remote control



Pattern 3: On-demand clients

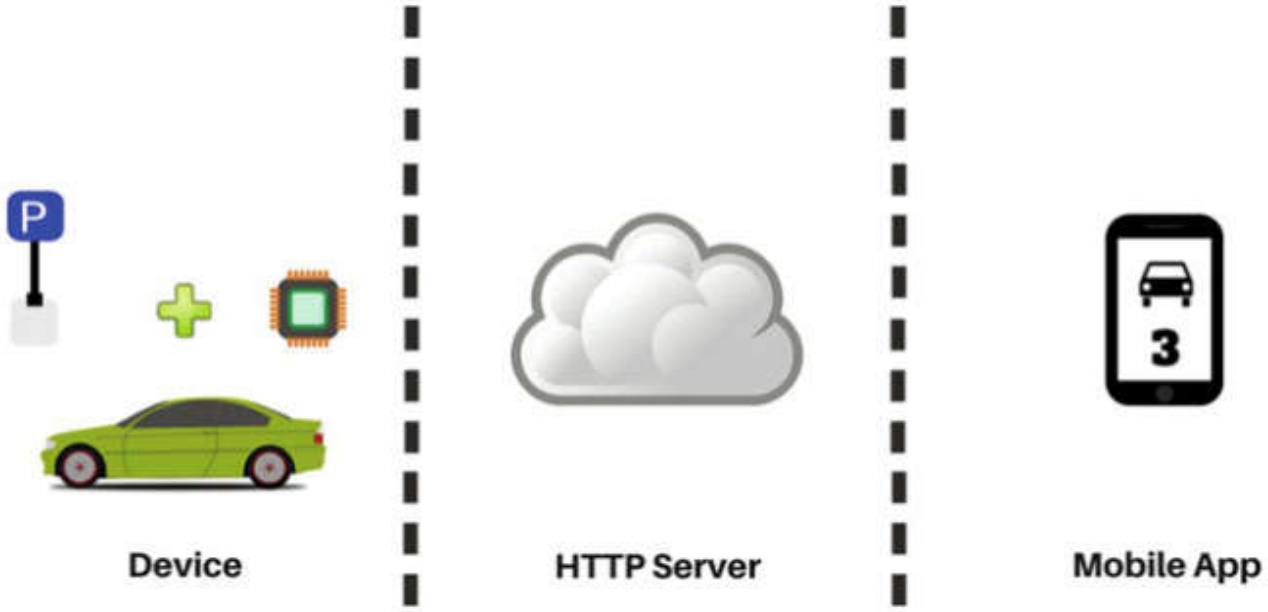
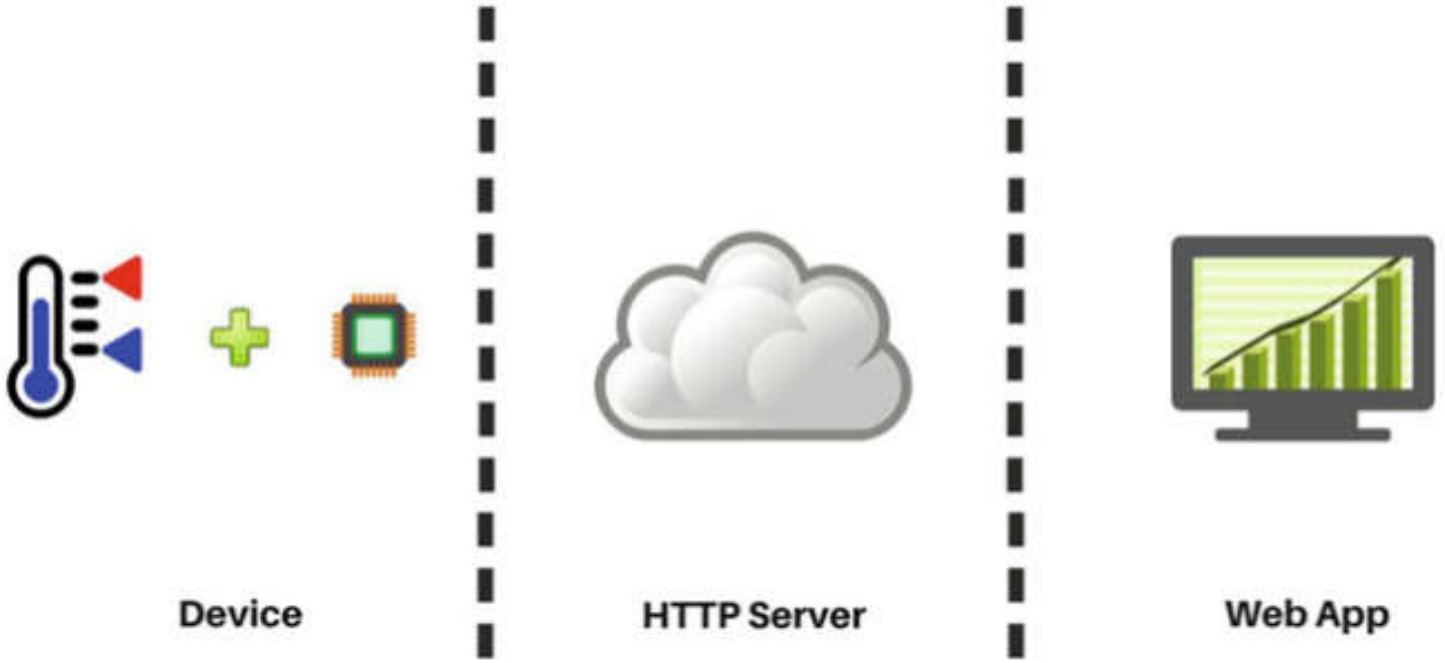
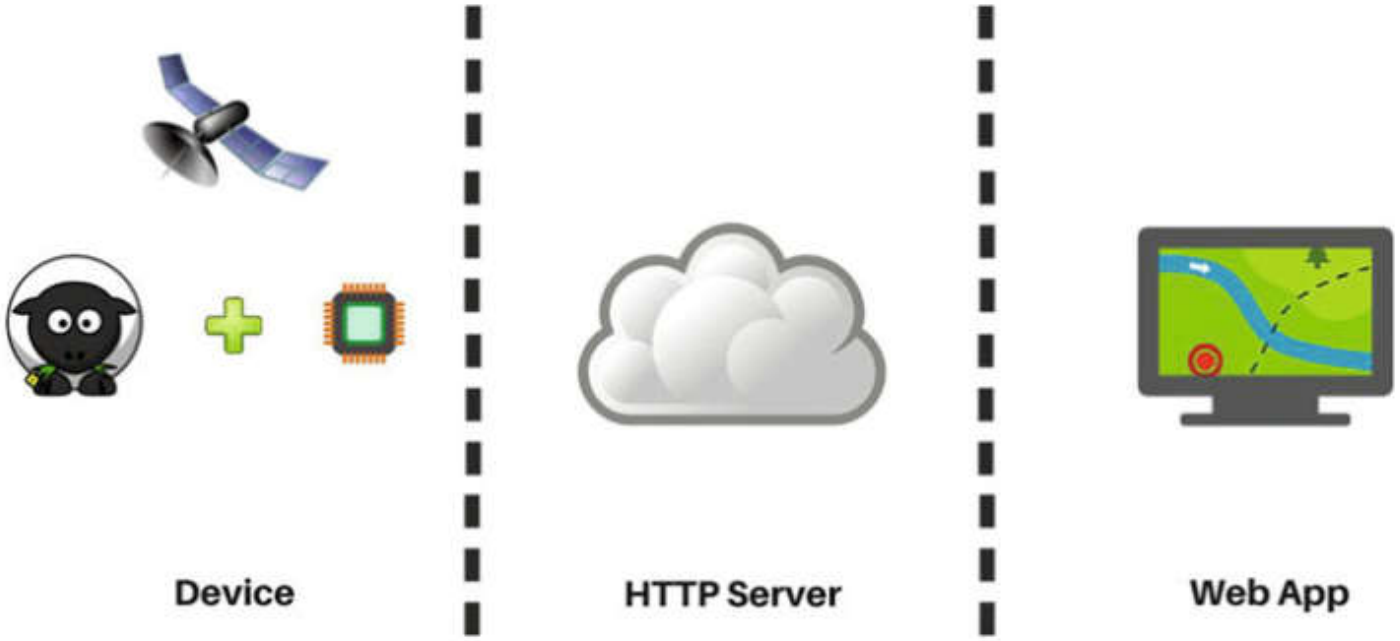


Figure 7-1. *Components of the smarter parking system*

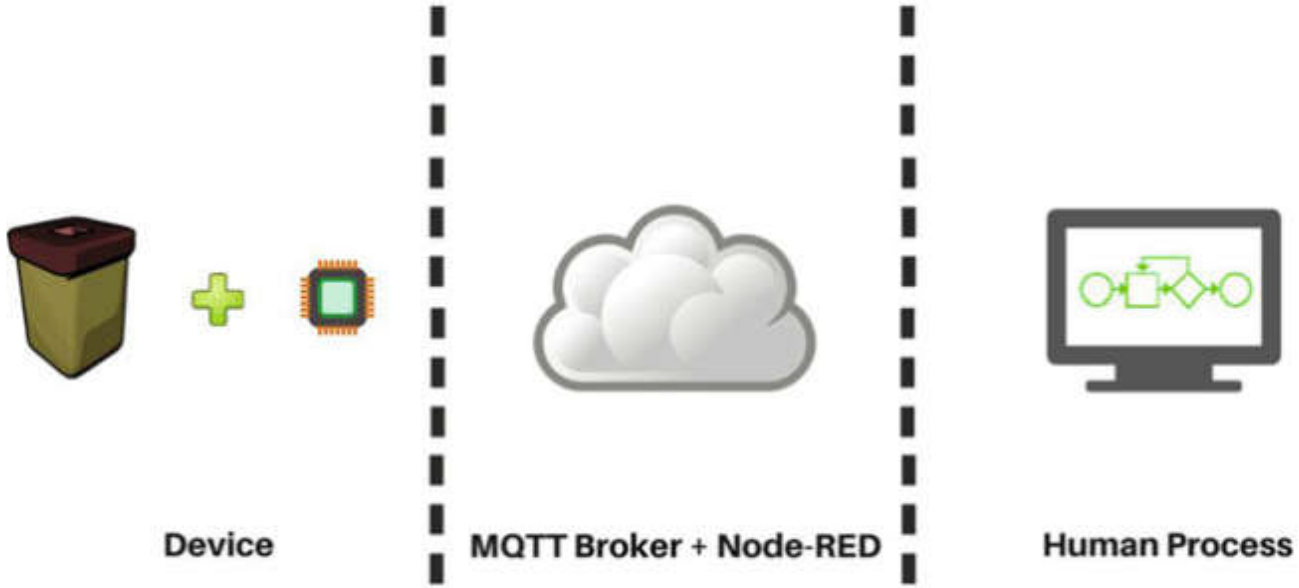
Pattern 4: Web-App



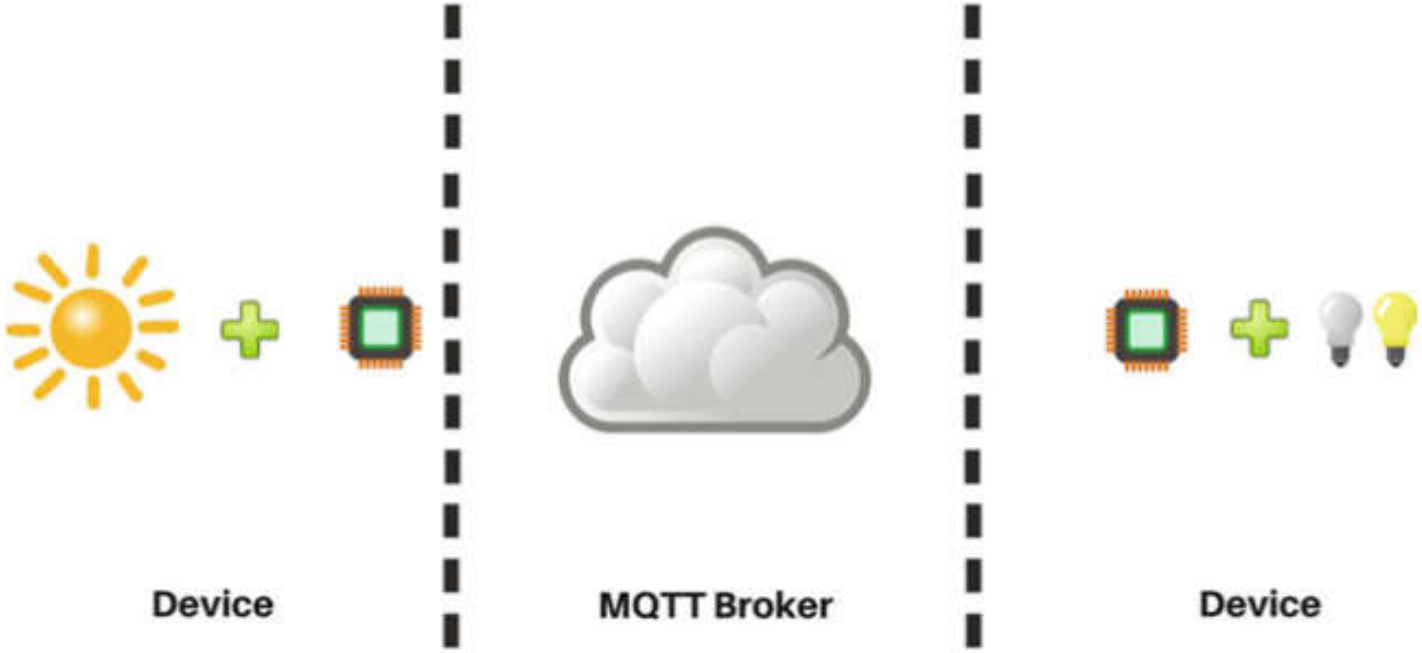
Pattern 4: Location Aware



Pattern 5: Machine-to-Human



Pattern 6: M2M



Références et Lectures

- Adeel Javel, 2016, Building Arduino projects for IoT
- Lawrence Miller, 2016, IoT security for Dummies
- Peter Waher, 2015, Learning IoT
- Charles Bell, 2017, MicroPython for IoT
- Charles Bell, 2020, Beginning Sensor Networks with Xbee, Raspberry Pi, and Arduino, 2nd Edition
- https://fr.statista.com/statistiques/561282/revenus-marche-objets-connectes-monde/