Plan de Cours

1. Projets réalisés
2. Introduction
3. Le marché de l'IOT
4. Interactions entre le « monde numérique » et le « monde physique »
5. Principes fondamentaux de l'IoT
6. Composants de base de l'IoT
7. **Connectivité**

➢ **Réseaux cellulaires**
- 3G, 4G, 5G
- NB-IoT (Narrowband IoT)
- **LTE-M**
- SIM900 GSM/GPRS Shield
- SIM800L GSM/GPRS Shield

➢ **Réseaux sans fil**
- Wi-Fi
- Bluetooth
- Zigbee

➢ **Détection de proximité**
- NFC (Near Field Communication)
- RFID (Radio-Frequency Identification)
- QR Codes

Connectivité

I. Réseaux cellulaires
- A. Pourquoi les Réseaux Cellulaires ?
- B. Réseaux : 2G, 3G, 4G, 5G
- C. Comparaison entre les réseaux : 2G, 3G, 4G, 5G
- D. LTE-M and NB-IOT
  - D.1. LTE-M
  - D.2. NB-IoT (Narrowband IoT)
  - D.3. M2M-GSM (2G)
  - D.4. Comparaison entre les réseaux : LTE, LTE-M, GSM-M2M et NB-IOT
- A. Shields IoT pour une Connectivité de Réseau Cellulaire
- B. SIM900 GSM/GPRS Shield
- C. Envoi d'un SMS via le shield SIM900 GSM/GPRS
- D. SIM800L GSM/GPRS Shield
- E. Appel téléphonique en utilisant le shield SIM800L GSM/GPRS

II. Réseaux sans fil

II.1 Importance de la Connectivité sans fil

II.2Wi-Fi dans l'IoT
- A. Wi-Fi dans l'IoT – Technologie
- B. Shields Wi-Fi pour Arduino
- C. Exemple de code d'Utilisation d'ESP32 qui se connecte à un réseau Wi-Fi

II.3 Bluetooth dans l'IoT
- A. Bluetooth dans l'IoT - Technologie
- B. Shields Bluetooth pour Arduino
- C. Exemple d'Utilisation du Module Bluetooth HC-06
- D. Code pour envoyer un message à un appareil Bluetooth connec

II.4 Zigbee dans l'IoT
- A. Zigbee dans l'IoT - Technologie
- B. Shields Zigbee pour Arduino
- C. Exemple d'Utilisation du Module Zigbee
- D. Code d'Exemple Zigbee

II.5 Récapitulatif et Synthèse

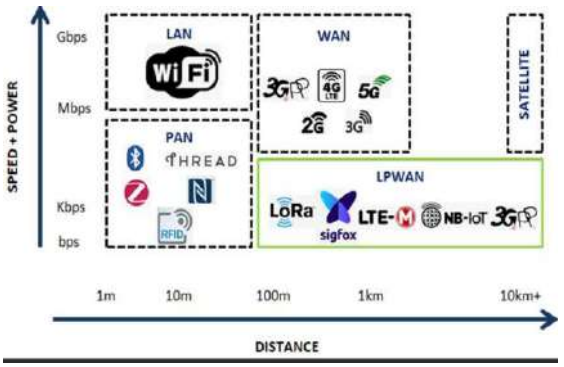
III. Détection de proximité
- A. NFC (Near Field Communication)
  - Fonctionnement de la technologie NFC
  - Applications de détection de proximité basées sur le NFC
- B. RFID (Radio-Frequency Identification)
  - Principes de fonctionnement des systèmes RFID
  - Exemples d'applications de détection de proximité avec RFID
- C. QR Codes
  - Comment les QR codes peuvent être utilisés pour la détection de proximité
  - Exemples d'applications basées sur les QR codes

I. Réseaux cellulaires (1/16)

A. Pourquoi les Réseaux Cellulaires ?

- **Fiabilité pour la transmission de données IoT** :
Les réseaux cellulaires offrent une connectivité stable et fiable, garantissant que les données IoT sont transmises en temps opportun et de manière cohérente.

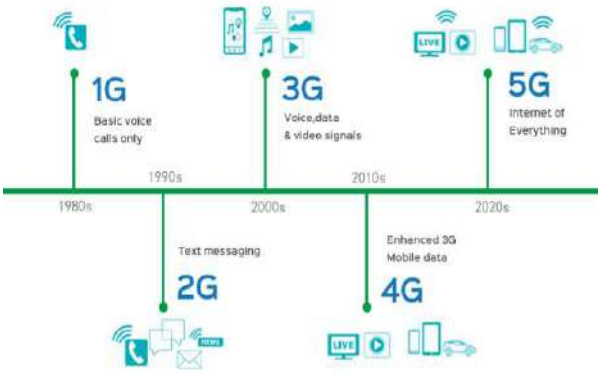
- **Portée étendue couvrant de vastes zones géographiques** :
Les réseaux cellulaires ont une portée qui peut couvrir de vastes territoires, ce qui les rend adaptés aux applications IoT étendues, telles que la surveillance agricole ou la gestion de flottes.

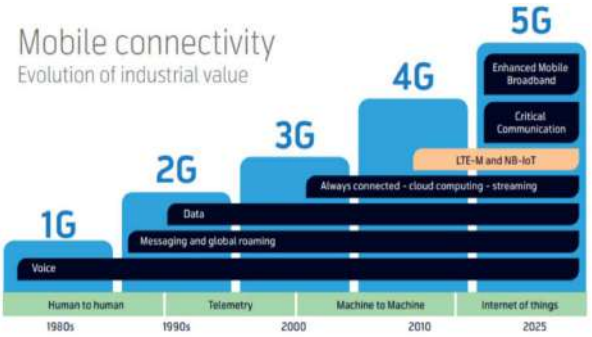


- **Disponibilité d'une connexion permanente** :
Les dispositifs IoT peuvent rester constamment connectés grâce aux réseaux cellulaires, assurant ainsi une communication ininterrompue.

I. Réseaux cellulaires (2/16)

B. Réseaux : 2G, 3G, 4G, 5G





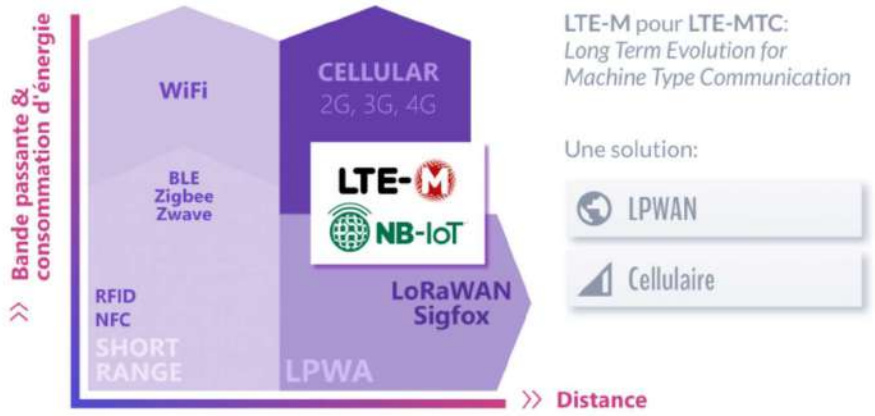
I. Réseaux cellulaires (3/16)

C. Comparaison entre les réseaux : 2G, 3G, 4G, 5G

Caractéristique	Réseau 2G	Réseau 3G	Réseau 4G	Réseau 5G
Débit de données	Faible (jusqu'à 236 Kbps)	Moyen (jusqu'à 3.1 Mbps)	Élevé (jusqu'à 100 Mbps)	Très élevé (jusqu'à plusieurs Gbps)
Latence	Modérée	Modérée	Faible	Très faible
Couverture	Bonne	Bonne	Excellente	En cours de déploiement
Consommation d'énergie	Faible	Modérée	Variable (optimisation possible)	Variable (optimisation possible)
Capacité d'appareils pris en charge	Faible	Modérée	Élevée	Très élevée
Utilisation principale	M2M (Machine-to-Machine)	Voix et données	Données et multimédia	Diverses applications IoT et multimédia

I. Réseaux cellulaires (4/16)

D. LTE-M and NB-IOT



I. Réseaux cellulaires (5/16)

D.1. LTE-M

LTE-M est une technologie de communication sans fil conçue pour répondre aux besoins de l'Internet des objets (IoT) en utilisant les réseaux 4G LTE.

Caractéristiques clés :

- **Faible consommation d'énergie** : LTE-M permet une utilisation efficace de l'énergie, prolongeant ainsi la durée de vie de la batterie des appareils IoT.
- **Pénétration accrue** : Cette technologie offre une meilleure pénétration à travers les obstacles physiques, améliorant la connectivité en intérieur et en souterrain.
- **Vitesse de données modérée** : LTE-M offre des débits de données modérés adaptés aux applications IoT telles que la surveillance et le suivi.
- **Prise en charge de nombreux appareils** : Il peut gérer un grand nombre d'appareils connectés simultanément.

Applications :

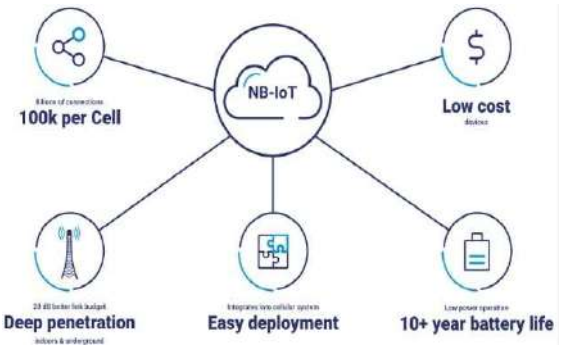
- Suivi d'actifs : Pour suivre la localisation des biens et des équipements.
- Télémétrie : Utilisée pour la surveillance à distance, notamment dans l'industrie.
- Gestion de flottes : Pour la surveillance et la gestion des véhicules commerciaux.

I. Réseaux cellulaires (6/16)

D.2. NB-IoT (Narrowband IoT)

NB-IoT est une technologie de communication sans fil conçue spécifiquement pour l'Internet des objets (IoT).

- Avantages :
  - Faible consommation d'énergie
  - Longue portée de transmission
  - Connectivité robuste
- Applications :
  - Suivi d'actifs
  - Compteurs intelligents
  - Agriculture intelligente
- Évolutivité :
  - NB-IoT est une solution évolutive pour les besoins croissants de l'IoT.



I. Réseaux cellulaires (7/16)

D.3. M2M-GSM (2G)

M2M-GSM fait référence à la communication machine à machine utilisant la technologie GSM. Le GSM est utilisé où les appareils et les capteurs sont connectés pour échanger des données sans intervention humaine directe.

- **Ancienne technologie** : Le GSM est une technologie de deuxième génération (2G) qui a été largement utilisée dans le passé, mais qui est devenue obsolète dans de nombreuses régions en raison du déploiement de technologies plus avancées.
- **Débit de données limité** : Le GSM offre des débits de données relativement faibles, généralement adaptés aux applications M2M nécessitant un faible volume de données.
- **Faible consommation d'énergie** : Le GSM est relativement efficace en termes de consommation d'énergie, mais il n'est pas aussi optimisé que le NB-IoT en matière de faible consommation d'énergie.

I. Réseaux cellulaires (8/16)

D.4. Comparaison entre les réseaux : LTE, LTE-M, GSM-M2M et NB-IOT

Caractéristique	GSM-M2M	LTE	LTE-M	NB-IoT
Génération	2G	4G	4G	LPWA
Débit de données	Faible (jusqu'à 236 Kbps)	Élevé (jusqu'à plusieurs Mbps)	**Modéré (jusqu'à 1 Mbps)**	Faible (jusqu'à 250 Kbps)
Consommation d'énergie	Modérée à élevée	Modérée à élevée	Faible	**Très faible**
Pénétration des obstacles	Modérée à bonne	Modérée à bonne	Bonne	**Excellente**
Portée de transmission	**Bonne**	Moyenne à bonne	Moyenne à bonne	**Bonne à excellente**
Prise en charge d'appareils	Modéré	Élevée	Modérée à élevée	**Élevée**
Coût de la connectivité	Modéré	Modéré	Modéré à faible	**Faible**
Vitesse de données adaptée à	Applications M2M	Applications gourmandes en données	Applications de surveillance et de suivi	Applications de faible bande passante
Applications typiques	Surveillance, télémétrie	Voix, vidéo, données riches	Surveillance, suivi, capteurs	Capteurs, compteurs intelligents, suivi d'actifs

I. Réseaux cellulaires (9/16)

E. Shields IoT pour une Connectivité de Réseau Cellulaire

SIM900 GSM/GPRS Shield
Il offre une connectivité GSM/GPRS pour les projets Arduino et est compatible avec de nombreux modèles de cartes Arduino

SIM800L GSM/GPRS Shield
Ce shield est conçu pour les communications GSM/GPRS et est largement utilisé pour les projets IoT nécessitant une connectivité cellulaire.

Quectel EC25 Mini PCIe LTE Module
Bien qu'il s'agisse d'un module PCIe, il peut être utilisé avec un shield PCIe pour ajouter une connectivité LTE à un projet IoT.

I. Réseaux cellulaires (10/16)

E. Shields IoT pour une Connectivité de Réseau Cellulaire

Quectel BC95-G NB-IoT Shield
conçu pour les applications NB-IoT, offrant une connectivité adaptée aux besoins de l'IoT à faible consommation d'énergie.

Particle Boron LTE
une carte de développement IoT intégrant un modem LTE pour des projets de bout en bout..

Pycom FiPy
Carte de développement IoT polyvalente qui prend en charge la connectivité cellulaire, y compris LTE-M et NB-IoT, en utilisant des modules Pycom

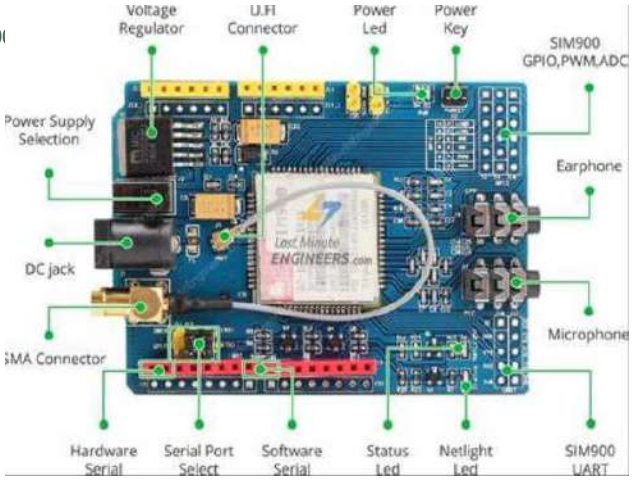
I. Réseaux cellulaires (11/16)

F. SIM900 GSM/GPRS Shield

Shield possède 12 GPIOs, 2 PWMs et un ADC du module SIM900 (ce sont tous des logiques 2V8) présents à bord.

Caractéristiques:

- Quadri-bande, compatible avec les réseaux GSM.
- Transmission de données efficace grâce au GPRS multi-slot.
- Puissance de transmission de Classe 4 (2 W @ 850/900 MHz) et Classe 1 (1 W @ 1800/1900 MHz).
- Envoi de SMS pour la transmission de données.
- Pile TCP/UDP intégrée pour le téléchargement de données sur un serveur web.
- Intégration d'une horloge temps réel (RTC).
- Faible consommation d'énergie en mode veille (1.5 mA).
- Compact et facile à intégrer dans des projets Arduino et IoT.



I. Réseaux cellulaires (12/16)

G. Envoi d'un SMS via le shield SIM900 GSM/GPRS

Explication du code :

1.Utilisation la bibliothèque SoftwareSerial pour établir une communication série avec le SIM900 via les broches 7 (TX) et 8 (RX) de l'Arduino.

2.Dans la fonction setup(), nous initialisons la communication série avec le moniteur série et avec le SIM900 via mySerial. Nous attendons un certain temps pour que le module SIM900 s'initialise.

3.La boucle loop() est utilisée pour envoyer un SMS. Nous utilisons des commandes AT pour interagir avec le SIM900.
4.Nous envoyons la commande AT pour vérifier si le module est prêt.
5.Ensuite, nous configurons le mode SMS en mode texte avec la commande "AT+CMGF=1".
6.Nous envoyons la commande "AT+CMGS" pour entrer le numéro de téléphone de destination.
7.Ensuite, nous envoyons le contenu du message SMS.
8.Pour terminer le message SMS, nous envoyons le caractère Ctrl+Z (0x1A).
9.Nous attendons quelques secondes avant de répéter le processus.

I. Réseaux cellulaires (13/16)

G. Envoi d'un SMS via le shield SIM900 GSM/GPRS

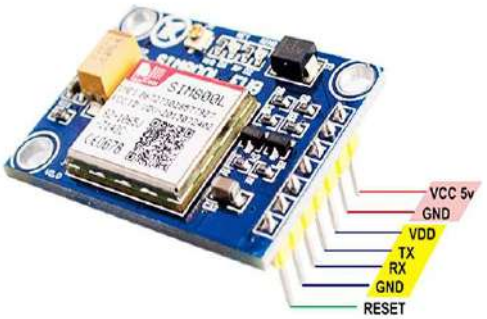
```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(7, 8); // Crée un objet SoftwareSerial pour la communication avec le SIM900
void setup() {
  // Initialise la communication série avec le moniteur série
  Serial.begin(9600);
  Serial.println("Initialisation...");
  // Initialise la communication série avec le SIM900
  mySerial.begin(9600);
  // Attendre que le SIM900 s'initialise (cela peut prendre quelques secondes)
  delay(20000);
  Serial.println("Le SIM900 est prêt.");
}
void loop() {
  // Envoie la commande AT pour s'assurer que le module est prêt
  mySerial.println("AT");
  delay(1000);

  // Attendez la réponse du module
  while (mySerial.available()) { Serial.write(mySerial.read());}

  delay(1000); // Attendez une seconde

  // Envoie la commande pour configurer le mode SMS
  mySerial.println("AT+CMGF=1"); // Mode texte
  delay(1000);

  // Attendez la réponse
  while (mySerial.available()) { Serial.write(mySerial.read());}

  delay(1000);// Attendez une seconde

  // Envoie la commande pour entrer le numéro de téléphone de destination
  mySerial.println("AT+CMGS=\"+1234567890\""); // Remplacez +1234567890 par le numéro de téléphone de destination
  delay(1000);
  // Attendez la réponse
  while (mySerial.available()) { Serial.write(mySerial.read());}

  delay(1000);// Attendez une seconde

  // Envoyer le message SMS
  mySerial.println("Ceci est un message SMS envoyé depuis Arduino !");
  delay(1000);

  // Attendez la réponse
  while (mySerial.available()) { Serial.write(mySerial.read());}

  // Envoyer Ctrl+Z pour terminer le message SMS
  mySerial.write(0x1A);
  delay(1000);

  // Attendez la réponse
  while (mySerial.available()) { Serial.write(mySerial.read());}

  // Attendez quelques secondes avant de répéter le processus
  delay(5000);
}
```


I. Réseaux cellulaires (14/16)

H. SIM800L GSM/GPRS Shield

Caractéristiques:

- Support de données pour 4 fréquences de communication, utilisable à l'échelle mondiale.
- Port série TTL compatible avec les MCU de 3,3V et 5V, permettant une connexion directe.
- Alimentation de 5V, module USB pour le débogage, et port série TTL.
- Courant de sortie de l'alimentation de 800 mA.
- Fonctionnalités supplémentaires :
  - Ultra-petite taille adaptée à diverses applications, y compris les brassards et les systèmes embarqués.
  - Conception de fente latérale pour faciliter le remplacement de la carte SIM lors de la conception dans un boîtier.
  - Interface IPX offrant la possibilité de remplacer l'antenne (antenne PCB par défaut).



I. Réseaux cellulaires (15/16)

I. Appel téléphonique en utilisant le shield SIM800L GSM/GPRS

Explication du code :

1.Nous utilisons la bibliothèque SoftwareSerial pour établir une communication série avec le SIM800L via les broches 7 (TX) et 8 (RX) de l'Arduino.

2.Dans la fonction setup(), nous initialisons la communication série avec le moniteur série et avec le SIM800L via sim800l. Nous attendons un certain temps pour que le module SIM800L s'initialise.

1.La boucle loop() est utilisée pour effectuer un appel téléphonique. Nous utilisons des commandes AT pour interagir avec le SIM800L.
2.Ensuite, nous utilisons la commande "ATD+1234567890;" pour appeler le numéro de téléphone spécifié (remplacez "+1234567890" par le numéro que vous souhaitez appeler).
3.Nous attendons un certain temps (par exemple, 30 secondes) pour l'appel.
4.Ensuite, nous raccrochons l'appel en envoyant la commande "ATH".
5.Nous attendons quelques secondes avant de répéter le processus.

I. Réseaux cellulaires (16/16)

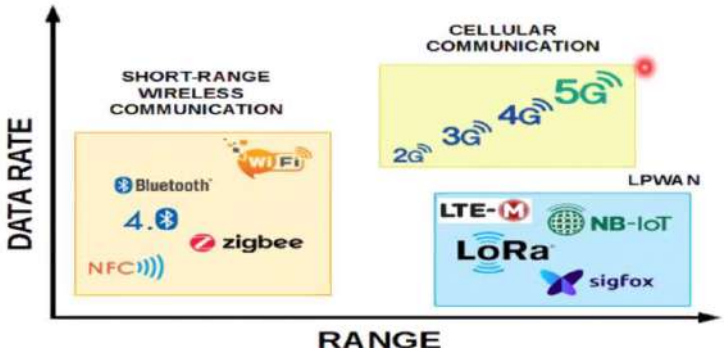
I. Appel téléphonique en utilisant le shield SIM800L GSM/GPRS

```
#include <SoftwareSerial.h>

SoftwareSerial sim8001(7, 8); // Crée un objet SoftwareSerial pour communiquer avec le SIM800L (TX sur 7, RX sur 8)

void setup() {
  // Initialise la communication série avec le moniteur série
  Serial.begin(9600);
  Serial.println("Initialisation...");

  // Initialise la communication série avec le SIM800L
  sim8001.begin(9600);

  // Attendre que le SIM800L s'initialise (cela peut prendre quelques secondes)
  delay(20000);

  Serial.println("Le SIM800L est prêt.");
}

void loop() {
  // Envoie la commande AT pour s'assurer que le module est prêt
  sim8001.println("AT");
  delay(1000);

  // Attendez la réponse du module
  while (sim8001.available()) { Serial.write(sim8001.read());}

  // Attendez une seconde
  delay(1000);

  // Envoie la commande pour appeler un numéro de téléphone
  sim8001.println("ATD+1234567890;"); // Remplacez +1234567890 par le numéro de téléphone à appeler
  delay(1000);

  // Attendez la réponse
  while (sim8001.available()) { Serial.write(sim8001.read());}

  // Attendez un moment (par exemple, 30 secondes) pour l'appel
  delay(30000);

  // Raccrochez l'appel en envoyant la commande ATH
  sim8001.println("ATH");
  delay(1000);

  // Attendez la réponse
  while (sim8001.available()) { Serial.write(sim8001.read());}

  // Attendez quelques secondes avant de répéter le processus
  delay(5000);
}
```

II. Réseaux sans fil (1/15)

A. Importance de la Connectivité sans fil

La connectivité sans fil est incontournable dans l'IoT en permettant la communication et le contrôle à distance, assurant une flexibilité, une économie d'énergie et une évolutivité essentielles pour le déploiement massif d'appareils connectés.



II. Réseaux sans fil (2/15)

II.I Wi-Fi dans l'IoT

A. Wi-Fi dans l'IoT - Technologie

Wi-Fi est une technologie de communication sans fil qui permet la connexion d'appareils à un réseau local sans utiliser de câbles physiques.

Caractéristique :

- **Sans fil :** Élimine le besoin de câbles.
- **Large portée :** Permet la connectivité à distance.
- **Haute vitesse :** Prend en charge des débits élevés.
- **Normes multiples :** 802.11n, 802.11ac, etc.

802.11b	Première version grand public, débit plus lent.
802.11g	Amélioration de la vitesse.
802.11n	Haute vitesse et portée améliorée.
802.11ac	Haute performance et largeur de bande.
802.11ax (Wi-Fi 6)	Haute efficacité et capacité.

II. Réseaux sans fil (3/15)

II.I Wi-Fi dans l'IoT

B. Shields Wi-Fi pour Arduino

- **ESP8266** : Le module ESP8266 est largement utilisé pour ajouter une connectivité Wi-Fi à des projets Arduino ou autonomes. Il est doté d'un microcontrôleur intégré et offre une connectivité Wi-Fi abordable.

Module sans fil
WiFi ESP ESP8266

Module sans fil
WiFi ESP-12N ESP8266

Module sans fil
WiFi ESP8266-07

II. Réseaux sans fil (4/15)

II.I Wi-Fi dans l'IoT

B. Shields Wi-Fi pour Arduino

- **NodeMCU** : Il s'agit d'une carte de développement basée sur l'ESP8266. Elle offre une connectivité Wi-Fi et un grand nombre de broches GPIO pour connecter des capteurs et des actionneurs.

NODEMCU LUA - WIFI MODULE DE CONNEXION WI-FI SÉRIE MODULE W / INTÉGRÉ CP2102 DRIVER IC

CARTE DE DÉVELOPPEMENT D1 MINI NODEMCU LUA

NODEMCU ESP8266 + 0,96 POUCES OLED CARTE DE DÉVELOPPEMENT

II. Réseaux sans fil (5/15)

II.I Wi-Fi dans l'IoT

B. Shields Wi-Fi pour Arduino

- **ESP32** : L'ESP32 est une évolution de l'ESP8266 avec une plus grande puissance de traitement, une connectivité Wi-Fi et Bluetooth intégrée, ainsi que de nombreuses interfaces matérielles. Il est adapté aux projets IoT plus avancés.

ESP32 WROOM32
MODULE 30 BROCHES
DOIT DEVKIT V1

ESP32 WIFI + CARTE DE
DÉVELOPPEMENT
BLUETOOTH 38 BROCHES

MODULE SANS FIL TTGO T-CALL V1.3
ESP32 ANTENNE GPRS MODULE
SIM800L DE LA CARTE SIM

II. Réseaux sans fil (6/15)

II.I Wi-Fi dans l'IoT

C. Exemple de code d'Utilisation de ESP32 qui se connecte à un réseau Wi-Fi

```
#include <WiFi.h>

const char* ssid = "Nom_du_Réseau_Wi-Fi";
const char* password = "Mot_de_passe_du_Réseau_Wi-Fi";
const char* targetIP = "192.168.1.2";  // Remplacez par l'adresse IP de la machine B

void setup() {
  Serial.begin(115200);
  delay(1000);

  // Connexion au réseau Wi-Fi
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connexion en cours au réseau Wi-Fi...");
  }

  Serial.println("Connecté au réseau Wi-Fi");
  Serial.print("Adresse IP de l'ESP32: ");
  Serial.println(WiFi.localIP());

  // Envoie d'un message à la machine B
  sendMessage("Salut, je suis connecté à la machine B!");
}
```

```
void loop() {
  // Votre code ici
}

void sendMessage(const char* message) {
  WiFiClient client;

  if (client.connect(targetIP, 80)) {
    Serial.println("Connexion à la machine B réussie.");
    client.print("GET /send_message?message=");
    client.print(message);
    client.println(" HTTP/1.1");
    client.println("Host: example.com");
    // Remplacez par le nom de domaine ou l'adresse IP de la machine B
    client.println("Connection: close");
    client.println();
    client.stop();
    Serial.println("Message envoyé avec succès à la machine B.");
  } else {
    Serial.println("Échec de la connexion à la machine B.");
  }
}
```

II. Réseaux sans fil (7/15)

II.2 Bluetooth dans l'IoT

A. Bluetooth dans l'IoT - Technologie

Bluetooth est une technologie de communication sans fil à courte portée qui permet la connexion d'appareils électroniques entre eux.

Caractéristique :

- **Courte portée :** Généralement jusqu'à 10 mètres.
- **Faible consommation d'énergie** : Idéal pour les appareils alimentés par batterie.
- **Connexions simultanées :** Plusieurs appareils peuvent être connectés en même temps.
- **Interopérabilité :** Prise en charge par une grande variété d'appareils.

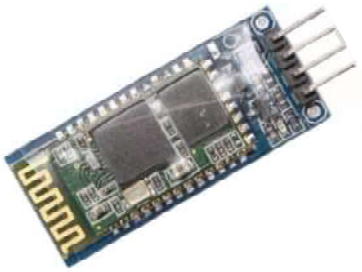
Bluetooth 1.x	Première version, faible débit binaire.
Bluetooth 2.0	Introduit l'EDR (Enhanced Data Rate).
Bluetooth 3.0	Technologie HS (High Speed) pour le transfert de données.
Bluetooth 4.0	Bluetooth Low Energy (BLE) pour les appareils IoT.
Bluetooth 5.0	Améliorations de la portée et de la vitesse.

II. Réseaux sans fil (8/15)

II.2 Bluetooth dans l'IoT

B. Shields Bluetooth pour Arduino

le contrôle précis de l'angle dans les projets de robotique et de mécanique.



Hc-06 du module sans fil bluetooth

ESP32-CAM WIFI + BLUETOOTH CAMÉRA MODULE

JDY-31 SPP-C MODULE BLUETOOTH

II. Réseaux sans fil (9/15)

II.2 Bluetooth dans l'IoT

C. Exemple d'Utilisation du Module Bluetooth HC-06

HC-06 fonctionne en mode esclave (Slave) et est principalement utilisé pour établir une communication série sans fil avec d'autres appareils Bluetooth, comme un smartphone ou un ordinateur.

- HC-06 RXD -> Arduino TX (Pin 1);
- HC-06 TXD -> Arduino RX (Pin 0);
- VCC (HC-06) -> 5V (Arduino);
- GND (HC-06) -> GND (Arduino);

II. Réseaux sans fil (10/15)

II.2 Bluetooth dans l'IoT

D. Code pour envoyer un message à un appareil Bluetooth connecté

```
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(0, 1);  // RX, TX

void setup() {
  Serial.begin(9600);
  BTSerial.begin(9600);  // Définit la vitesse de communication Bluetooth

  Serial.println("En attente de la connexion Bluetooth...");
}

void loop() {
  if (BTSerial.available()) {
    char data = BTSerial.read();
    Serial.print(data);

    // Si vous recevez un caractère spécifique, envoyez un message
    if (data == 'A') {
      sendMessageToBluetooth("Message depuis l'Arduino vers la machine B.");
    }
  }
}
```

```
void sendMessageToBluetooth(const char* message) {
  BTSerial.println(message);
  Serial.println("Message envoyé via Bluetooth :");
  Serial.println(message);
}
```

II. Réseaux sans fil (11/15)

II.3 Zigbee dans l'IoT

A. Zigbee dans l'IoT - Technologie

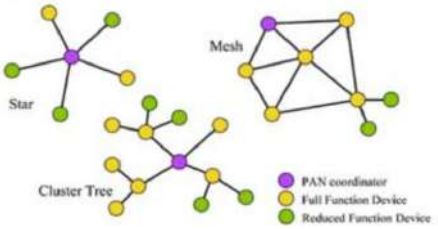
ZigBee est un protocole de communication sans fil conçu pour les réseaux de capteurs et les appareils à faible consommation d'énergie.

Caractéristique :

- **Faible consommation d'énergie**;
- **Réseaux maillés** : ZigBee permet la création de réseaux maillés auto-organisés;
- **Fiabilité** : ZigBee est robuste et peut fonctionner dans des environnements perturbés;
- **Faible débit binaire** : Convient pour la transmission de petites quantités de données.

	Zigbee IEEE 802.15.4	Zigbee IEEE 802.15.4	Zigbee IEEE 802.15.4a
Portée théorique	100m	1km	100m
Débit	250kbps	20kbps	250kbps
Bande de fréquence	2.4GHz	868MHz	2.4GHz
Localisation			X

Topologies :



II. Réseaux sans fil (12/15)

II.3 Zigbee dans l'IoT

B. Shields Zigbee pour Arduino

XBee S2 2mW Zigbee Wireless Data Transmission Module 120 Meters For Arduino

ZigBee / CC2530 Development Module

Arduino ZigBee Shield

II. Réseaux sans fil (13/15)

II.3 Zigbee dans l'IoT

C. Exemple d'Utilisation du Module Zigbee

Le "Arduino ZigBee Shield" est conçu pour permettre la communication sans fil ZigBee avec des modules compatibles. ZigBee est un protocole de communication sans fil populaire utilisé pour la communication entre des appareils à faible consommation d'énergie.

Matériel requis :

- ❑ Deux modules ZigBee compatibles (par exemple, les modules XBee de Digi);
- ❑ Un Arduino avec le ZigBee Shield;
- ❑ Câbles de raccordement.

Vous aurez besoin de connecter le module ZigBee sur le bouclier avec les broches TX et RX (pour la communication série).

II. Réseaux sans fil (14/15)

II.3 Zigbee dans l'IoT

D. Code d'Exemple Zigbee

```
#include <SoftwareSerial.h>

SoftwareSerial ZigBeeSerial(2, 3);  // RX, TX (vous pouvez utiliser d'autres broches)

void setup() {
  Serial.begin(9600);
  ZigBeeSerial.begin(9600);  // Configurez la vitesse de communication de votre module ZigBee

  Serial.println("En attente de la connexion ZigBee...");
}

void loop() {
  if (ZigBeeSerial.available()) {
    char data = ZigBeeSerial.read();
    Serial.print(data);

    // Si vous recevez un caractère spécifique, envoyez un message
    if (data == 'A') {
      sendMessageToZigBee("Message depuis l'Arduino vers la machine B.");
    }
  }
}
```

```
void sendMessageToZigBee(const char* message) {
  ZigBeeSerial.println(message);
  Serial.println("Message envoyé via ZigBee :");
  Serial.println(message);
}
```

II. Réseaux sans fil (15/15)

II.4 Récapitulatif et Synthèse

Caractéristiques	Wi-Fi	Bluetooth	Zigbee
Portée	30 mètres à plusieurs centaines de mètres avec des routeurs Wi-Fi	jusqu'à 100 mètres	30 mètres à environ 100 mètres
Débit de données	Haut débit (jusqu'à plusieurs Gbit/s, dépend du standard)	Faible à moyen débit (jusqu'à 3 Mbit/s)	Faible à moyen débit (250 kbit/s à 1 Mbit/s)
Consommation d'énergie	Modérée à élevée (plus élevée pour les applications de diffusion en continu, mais optimisée pour les transferts de données importants)	Faible (optimisée pour les applications à faible consommation d'énergie)	Faible (optimisée pour les applications à faible consommation d'énergie)
Utilisation principale	Connexions haut débit dans des environnements avec une alimentation électrique stable (domotique, vidéosurveillance, appareils multimédias)	Connexions à faible consommation d'énergie pour les appareils mobiles (casques, écouteurs, capteurs)	Applications IoT à faible consommation d'énergie (domotique, capteurs, éclairage intelligent)
Protocole de communication	TCP/IP	Protocole propriétaire pour les applications de profil générique (ex : Bluetooth Low Energy)	Protocole Zigbee (IEEE 802.15.4)
Sécurité	Forte, avec des protocoles de sécurité avancés (WPA3, etc.)	Bonne, avec des protocoles de sécurité pour les connexions sécurisées	Bonne, avec des protocoles de sécurité Zigbee
Scalabilité	Réseaux étendus, mais avec des exigences en alimentation électrique	Connexions point à point ou point à plusieurs points	Réseaux maillés et la mise en réseau de nombreux appareils

Fin