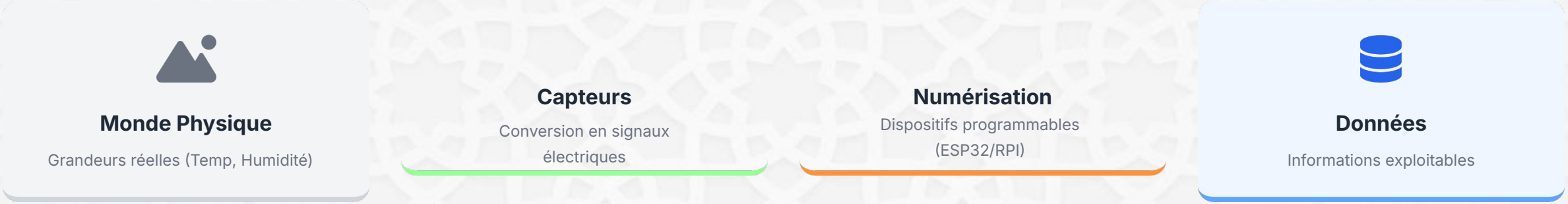
Chapitre 2 : Acquisition des Données IoT

De la grandeur physique à la donnée numérique exploitable.

1. Types de capteurs : température, humidité, lumière, mouvement, etc.
2. Principe de fonctionnement des capteurs
3. Critères de sélection des capteurs selon les applications
4. Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
5. Installation et configuration des environnements de développement
6. Lecture des données issues des capteurs et transmission des données

Introduction à l'Acquisition

L'acquisition de données constitue la **première étape fondamentale** de toute solution IoT. Elle agit comme le pont vital entre le monde physique et le domaine numérique.



Capter
Détecter les variations physiques avec précision.

Transformer
Convertir les mesures en signaux interprétables.

Communiquer
Transmettre les données pour analyse.

- **Typologie des capteurs**
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Les **capteurs** et les **actionneurs** sont des éléments fondamentaux des ***systèmes embarqués*, de *l'IoT* et de l'*automatisation industrielle***.

- **Les capteurs** permettent de **mesurer** une **grandeur physique** et de **la convertir en un signal exploitable par un système électronique.**
- **Les actionneurs** reçoivent un signal de commande et effectuent une action physique (mouvement, changement d'état, etc.).

Ces deux composants assurent **l'interaction entre le monde physique et les systèmes numériques**.

- **Typologie des capteurs**
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

1. Les Capteurs :

ENERGIE

Grandeur physique
- température
- Pression
- Force
- ...

→ Détecter un évènement →

Signal électrique
- signal logique (TOR)
- signal analogique
- signal numérique

1.1 Définition

Les **capteurs** sont des dispositifs permettant de mesurer une **grandeur physique** et de la convertir en un signal électrique sous form d'un signal **analogue** ou **numérique** utilisable par un système électronique.

Exemples de capteurs :

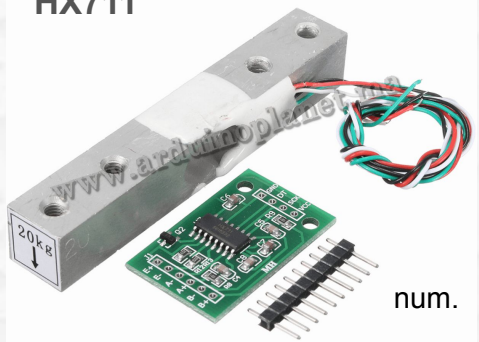
- **Capteur de température** : LM35, DS18B20
- **Capteur de poids** : **HX711,**
- **Capteur de gaz** : MQ-2 (détection de gaz inflammables)
- **Capteur de mouvement** : PIR (détection de présence)
- **Capteur de lumière** : LDR (détection de luminosité)
- **Capteur de son** : Microphones intégrés

LM35

num.

DS18B20

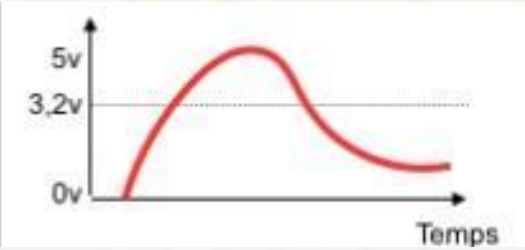
HX711



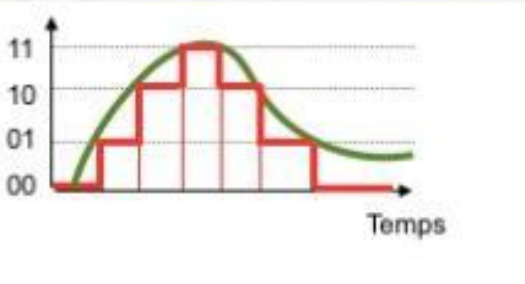
num.

1.2. Classification des capteurs:

1.2.1. Selon la nature du signal de sortie.



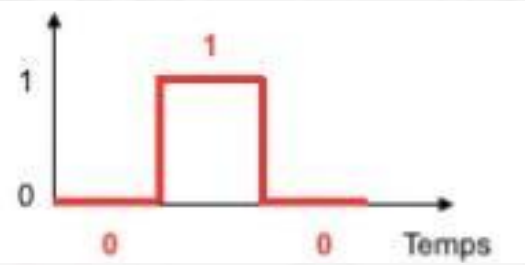
- **Capteurs analogiques** : génèrent un signal **continu** proportionnel à la grandeur mesurée (exemple : capteur de température **LM35**).



- **Capteurs numériques** : génèrent un signal **discret** (exemple : capteur de température **DHT11** qui transmet des données en numérique).

Cependant, certains capteurs peuvent effectivement être classés selon une troisième catégorie

Les capteurs logiques (TOR: Tout ou Rien **)** :Ceux-ci sont souvent une sous-catégorie des capteurs numériques, car ils fournissent une réponse de type binaire, mais se concentrent spécifiquement sur des événements ou des états binaires (par exemple, détecter un objet ou non, ou si un seuil est franchi).



- **Typologie des capteurs**
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Les capteurs logiques: L'information ne peut prendre que les valeurs 1 ou 0 ; on parle alors d'un capteur Tout ou Rien (TOR).

La figure montre la caractéristique d'un capteur de présence.

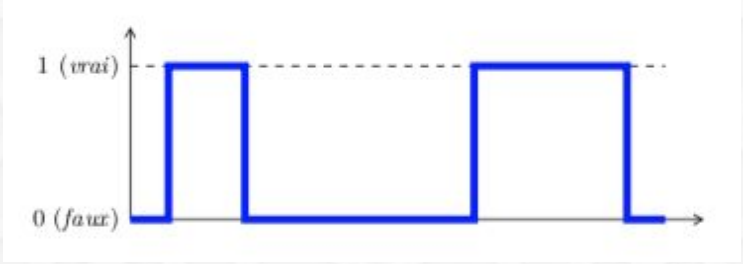


Fig. : Exemple d'un capteur TOR

Logique	Logique	Logique	Logique
			
Bouton poussoir	*Capteur fin de course*	*Barrière infrarouge*	*Détecteur de présence*

- **Capteurs numériques** : génèrent un signal **discret** (exemple : capteur de température **DHT11** qui transmet des données en numérique). L'information fournie par le capteur permet de déduire un nombre binaire sur n bits. c'est-à-dire un signal discret pouvant prendre un nombre fini de valeurs .

Le signal de sortie d'un capteur numérique est généralement une **séquence d'impulsions** ou de **mots binaires**. Par exemple, pour un codeur rotatif, le signal de sortie peut ressembler à ceci :

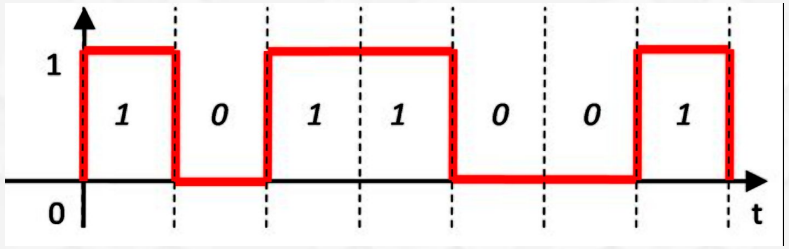
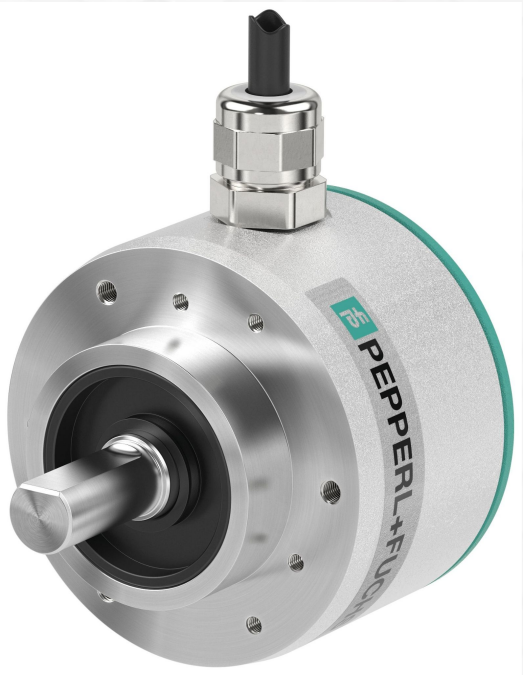


Diagramme typique du signal de sortie d'un **capteur numérique**



- **Typologie des capteurs**
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Capteurs analogiques : L'information peut prendre **toutes les valeurs possibles entre 2 certaines valeurs limites**, on parle alors d'un capteur analogique.

La figure montre la caractéristique d'un capteur de température.

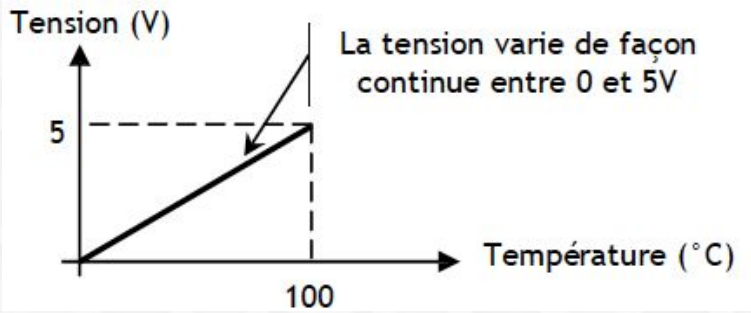


Fig. : Exemple d'un capteur analogique

Analogique	Analogique	Analogique	Analogique	Analogique	Analogique
					
Scanner	*Lecteur magnétique*	*Joystick*	*Capteur de luminosité*	*Capteur de T°C*	*Anémomètre*

- **Typologie des capteurs**
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

1.2.2. Selon la grandeur physique mesurée

Les capteurs sont classés selon la **grandeur physique** mesurée et leur **principe de transduction** pour convertir un signal physique en signal électrique.

Environnementaux

- Température (DHT11, DS18B20)
- Humidité (Capacitif, Résistif)

Optiques

- Luminosité (LDR, Photodiode)
- Infrarouge (PIR, Barrières)

Cinétiques

- Accéléromètres (MEMS)
- Gyroscopes (Orientation)

Mécaniques

- Pression (Barométrique)
- Force (Jauges de contrainte)

Chimiques

- Qualité de l'air (MQ-135)
- CO2, Gaz inflammables

Acoustiques

- Ultrasons (HC-SR04)
- Microphones (Niveau sonore)

Critères de performance : Précision, Résolution, Linéarité

- **Typologie des capteurs**
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

1.2.2. Selon la grandeur physique mesurée

Type de capteur	Grandeur mesurée	Exemple
Capteurs de température	Température	LM35, DS18B20, DHT11
Capteurs de pression	Pression atmosphérique, hydraulique	BMP280, MPX5700
Capteurs de lumière	Intensité lumineuse	LDR, BH1750
Capteurs de proximité	Présence d'objets	Ultrason HC-SR04, IR
Capteurs de mouvement	Mouvement d'un objet	Accéléromètre MPU6050, capteur PIR
Capteurs de gaz	Détection de gaz	MQ-2, MQ-135
Capteurs de courant et tension	Mesure de courant et tension	ACS712, ZMPT101B

Les capteurs permettent aux **systèmes IoT de capter en temps réel des données environnementales** et de les transmettre à un système de traitement.

1.3. Principes de fonctionnement des capteurs

La Transduction

Le capteur agit comme un **transducteur** : il convertit une grandeur physique (énergie thermique, mécanique, lumineuse) en un **signal électrique** (tension, courant, résistance) exploitable par un microcontrôleur.

Résistif
Variation de la résistance électrique suite à une contrainte physique.
EX: LDR, POTENTIOMÈTRE

Capacitif
Changement de la capacité via la modification du diélectrique ou des armatures.
EX: HUMIDITÉ, TACTILE

Inductif
Variation d'inductance provoquée par un mouvement ou un métal proche.
EX: DÉTECTEUR MÉTAUX

Piézo
Apparition de charges électriques sous l'effet d'une déformation mécanique.
EX: PRESSION, FORCE

Optique
Modulation d'un faisceau lumineux capté par un élément photosensible.
EX: INFRAROUGE, UV

Le choix du principe dépend de la précision, de la linéarité et du coût souhaité.

- Typologie des capteurs
- **Principes de fonctionnement**
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Capteurs de Température

La mesure de la température est l'un des besoins les plus courants en IoT. Le choix du capteur dépend de la **précision**, de la **plage** de mesure et de **l'interface**.

Thermistances

Composants passifs dont la **résistance électrique** varie avec la température.

- NTC (Coefficient négatif)
- PTC (Coefficient positif)
- Économique et sensible

Thermocouples

Basés sur l'effet **Seebeck** : génération d'une tension entre deux métaux différents.

- Plages de mesure extrêmes
- Robuste (Industriel)
- Nécessite une compensation

Numériques (DS18B20)

Circuits intégrés fournissant une valeur **numérique directe** via un bus de données.

- Protocole 1-Wire
- Haute précision
- Idéal pour Raspberry Pi/ESP32

Choix technologique selon l'application : Domotique : Thermistance / Numérique — Industrie / Fours : Thermocouple — Précision Labo : Numérique

Capteurs de Température: Thermistances

Les thermistances sont des capteurs de température **dont la résistance varie fortement avec la température**.

Composants passifs dont la résistance électrique varie avec la température :

- **NTC (Negative Temperature Coefficient)** : La résistance diminue quand la température augmente. Ils sont très sensibles aux variations thermiques et souvent utilisés comme **capteurs pour mesurer la température avec précision.**
- **PTC (Coefficient de température positif)** : La résistance augmente quand la température augmente. On les utilise notamment pour **la protection contre les surintensités**, la régulation thermique ou comme fusibles réarmables.

Caractéristiques clés :

- **Économiques et sensibles** : Ces composants offrent une **réponse rapide aux changements de température** tout en étant **peu coûteux** à produire et **faciles à intégrer dans des circuits électroniques**.

✔ **Thermistance NTC 10 kΩ** – très utilisée en électronique, mesure la température ambiante ou de surface dans les projets Arduino/électronique DIY(Do It Yourself).
✔ **Thermistance PTC 2 kΩ** – souvent utilisée pour la **protection contre les surintensités thermiques** et la **détection de surchauffe**.

Applications typiques : thermostats, électroménager, contrôles de charge, circuits de compensation thermique.

Choix technologique selon l'application : Domotique : Thermistance / Numérique — Industrie / Fours : Thermocouple — Précision Labo : Numérique

Thermistances

Composants passifs dont la **résistance électrique** varie avec la température.

- NTC (Coefficient négatif)
- PTC (Coefficient positif)
- Économique et sensible

- Typologie des capteurs
- **Principes de fonctionnement**
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Capteurs de Température: thermocouple

Un thermocouple se compose de **deux métaux différents soudés ensemble; il génère une tension proportionnelle à la différence de température** (effet Seebeck).

✔ **Thermocouple Type K (Chromel-Alumel)** – l'un des plus courants, **plage très large** (~ -200 °C à +1350 °C), bon compromis précision/coût.

✔ **Thermocouple Type J (Fer-Constantan)** – adapté à **températures moyennes** et sensible aux environnements corrosifs.

✔ **Thermocouple Type T (Cuivre-Constantan)** – **précis à basses températures**, utilisé pour la cryogénie et les applications médicales.

Thermocouples

Basés sur l'effet **Seebeck** : génération d'une tension entre deux métaux différents.

- Plages de mesure extrêmes
- Robuste (Industriel)
- Nécessite une compensation

Applications Clés

- Fours et étuves industrielles
- Turbines et moteurs thermiques
- Processus chimiques à haute T°
- Industrie agroalimentaire

PLAGE TYPIQUE

-200°C à +1800°C

(Selon le type de thermocouple choisi)

Auto-alimenté
Ne nécessite pas d'alimentation externe.

Robuste
Résiste aux chocs et vibrations.

Choix technologique selon l'application : Domotique : Thermistance / Numérique — Industrie / Fours : Thermocouple — Précision Labo : Numérique

Capteurs de Température: Thermocouples

Effet Seebeck: L'effet Seebeck est un phénomène thermoélectrique découvert par Thomas Seebeck en 1821.

Définition

Lorsqu'on relie **deux métaux différents** et que leurs jonctions sont à **des températures différentes**, une **tension électrique** apparaît dans le circuit.

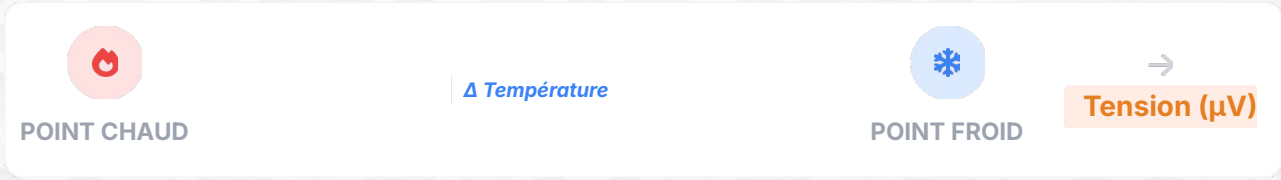
En résumé : **Différence de température → création d'une tension électrique.**

Principe de fonctionnement

- On assemble deux conducteurs de nature différente (ex : cuivre et constantan).
- On crée une différence de température entre les deux jonctions.
- Les électrons se déplacent différemment dans chaque matériau.
- Cela génère une **tension** ou **force électromotrice (f.e.m.)** mesurable.

La tension produite est proportionnelle à la différence de température : $V = S \times \Delta T$

- **V** : tension générée
- **S** : coefficient de Seebeck (dépend des matériaux)
- **ΔT** : différence de température



Choix technologique selon l'application : Domotique : Thermistance / Numérique — Industrie / Fours : Thermocouple — Précision Labo : Numérique

Thermocouple Type K (Chromel – Alumel)

Le standard industriel pour la mesure de température haute performance.

PLUS UTILISÉ

Composition

Chromel	**Alliage Ni-Cr**
Alumel	**Alliage Ni-Al**

Sensibilité

≈ 41 µV/°C

Bon compromis entre précision et coût.

Plage de Température

-200°C à +1350°C

INDUSTRIE

Fours, turbines, moteurs industriels.

LABORATOIRE

Mesures générales de haute précision.

- Résistant à l'oxydation
- Hautes températures

Thermocouple Type J

FER – CONSTANTAN

Composition Chimique

BRAS POSITIF	BRAS NÉGATIF
Fer (Fe)	**Constantan** (Cu-Ni)

Plage d'Utilisation

-40°C à **+750°C**

Idéal pour les températures moyennes.

Performance Technique

Sensibilité **≈ 55 µV/°C**

Sensibilité supérieure au Type K dans sa plage.

Points de Vigilance

- Le fer est extrêmement **sensible à l'oxydation**.
- Non recommandé en milieux humides ou corrosifs.

APPLICATIONS Industrie légère • Machines-outils • Plasturgie • Environnements secs

C'est le choix économique pour les mesures précises sous 750°C.

Thermocouple Type T (Cuivre – Constantan)

La référence pour les basses températures et la cryogénie.

TYPE T

COMPOSITION CHIMIQUE

Cu + Cn

Cuivre (+) **Constantan (-)**

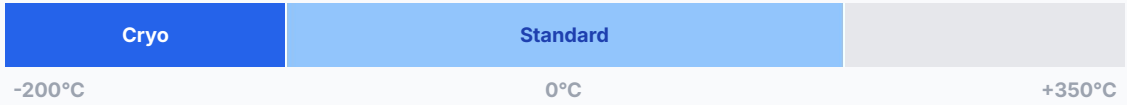
PLAGE

-200°C à +350°C

SENSIBILITÉ

~43 µV/°C

ÉTENDUE DE MESURE



Atouts Majeurs

- **Haute Précision :** Exceptionnelle dans les zones de froid intense.
- **Stabilité :** Très faible dérive temporelle des mesures.
- **Résistance :** Excellente tenue face à l'humidité et à la corrosion.

Domaines d'Application

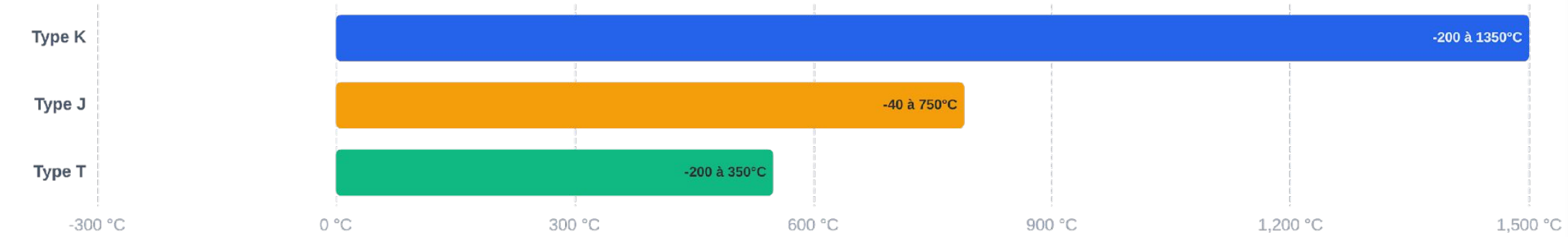
- Cryogénie
- Laboratoires
- Médical
- Agroalimentaire

Note Idéal pour les mesures de très basses températures.

Comparaison des Types de Thermocouples

Analyse comparative des performances et domaines d'application

PLAGES DE TEMPÉRATURE OPÉRATIONNELLES (°C)



TYPE	COMPOSITION	POINT FORT	LIMITE PRINCIPALE
K Type K	*Chromel – Alumel*	POLYVALENT	Précision moyenne
J Type J	*Fer – Constantan*	SENSIBILITÉ (55 MV/°C)	Sensible à la corrosion (Fer)
T Type T	*Cuivre – Constantan*	BASSE TEMPÉRATURE	Inadapté aux hautes T° (>350°C)

Données basées sur les spécifications standards des alliages métalliques.

Capteurs de Température: Numérique

DS18B20 – capteur de température **numérique très populaire** qui communique via un bus *1-Wire* (nécessite seulement un fil de données + alimentation).
Plage de mesure : −55 °C à +125 °C
Précision : ±0,5 °C (généralement entre −10 °C et +85 °C)
Résolution programmable : 9 à 12 bits
Chaque capteur possède **un identifiant unique**, permettant de brancher plusieurs capteurs sur une seule ligne de données.
Il existe aussi des **versions étanches avec sonde métallique** (boîtier acier inoxydable pour liquides/extérieur).

Applications typiques : systèmes embarqués (Arduino, Raspberry Pi), domotique, surveillance environnementale, contrôles IoT.

Numériques (DS18B20)

Circuits intégrés fournissant une valeur **numérique directe** via un bus de données.

- Protocole 1-Wire
- Haute précision
- Idéal pour Raspberry Pi/ESP32

Choix technologique selon l'application : Domotique : Thermistance / Numérique | Industrie / Fours : Thermocouple | Précision Labo : Numérique

- Typologie des capteurs
- **Principes de fonctionnement**
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Capteurs d'Humidité

Technologies de Mesure

CAPACITIVE

Mesure le changement de capacité d'un polymère ou d'un oxyde diélectrique.

Très stable et précise sur une large plage.

RÉSISTIVE

Mesure le changement de résistance électrique d'un milieu hygroscopique.

Souvent plus économique mais moins sensible aux extrêmes..

Standard IoT : Série DHT

Les modules DHT sont les capteurs de référence pour le prototypage rapide grâce à leur sortie numérique pré-calibrée.

DHT11
Usage basique, faible coût, boîtier bleu.

DHT22
Haute précision, plage étendue, boîtier blanc.

CARACTÉRISTIQUE	DHT11	DHT22
Plage Humidité	20 - 80%	0 - 100%
Précision	± 5%	± 2%
Échantillonnage	1 Hz (1s)	0.5 Hz (2s)
Coût	Très Faible	Modéré

Note : Ces capteurs intègrent une thermistance NTC pour la mesure simultanée de la température.

Application des capteurs d'humidité relative

Confort Humain

- Optimisation des systèmes CVC (chauffage, ventilation, climatisation)
- Régulation automatique dans les maisons intelligentes
- Prévention des allergies et problèmes respiratoires

Agriculture

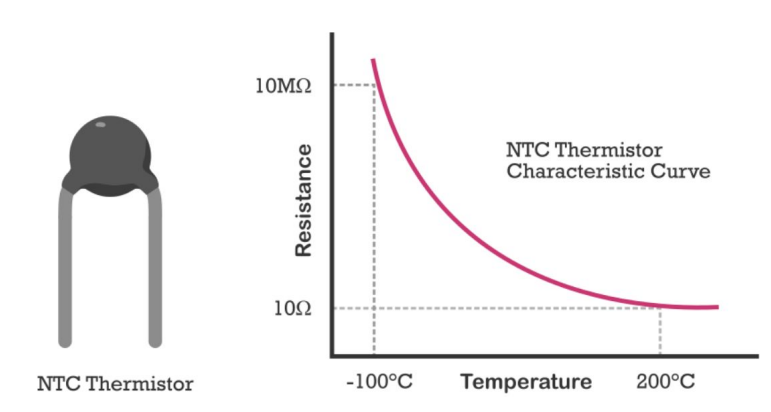
- Contrôle climatique des serres
- Irrigation intelligente
- Stockage des récoltes (prévention moisissures et fermentation)
- Bâtiments d'élevage (bien-être animal)

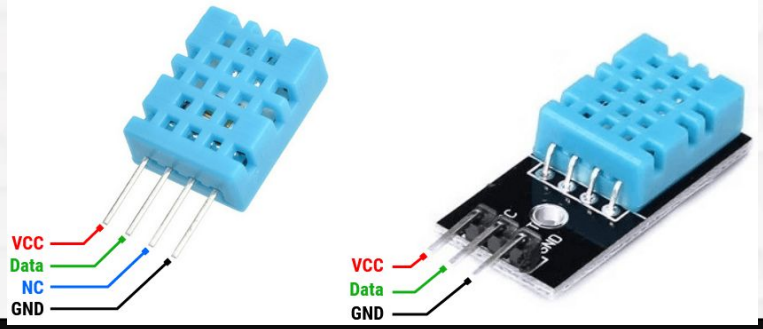
Industrie

- Salles blanches
- Industrie agroalimentaire
- Production pharmaceutique
- Stockage de matériaux sensibles

- Typologie des capteurs
- **Principes de fonctionnement**
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Standard IoT : Série DHT





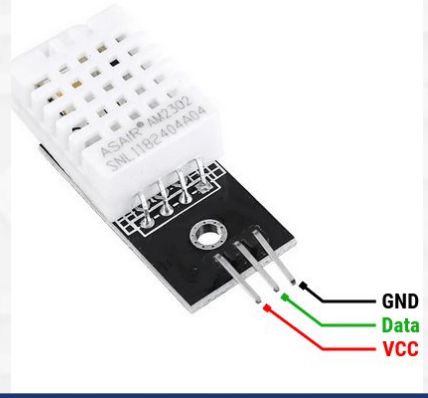
Note : Ces capteurs intègrent une thermistance NTC pour la mesure simultanée de la température.

Les modules DHT sont les capteurs de référence pour le prototypage rapide grâce à leur sortie numérique pré-calibrée.

DHT11
Usage basique, faible coût, boîtier bleu.

DHT22
Haute précision, plage étendue, boîtier blanc.

	DHT11	DHT22
Temperature Range	0 to 50 °C	-40 to 80 °C
Temperature Accuracy	+/-2 °C	+/-0.5°C
Humidity Range	20% to 80%	0% to 100%
Humidity Accuracy	+/-5%	+/-2%
Sampling Period	1 Second	1 Second



- Typologie des capteurs
- **Principes de fonctionnement**
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Capteurs de Lumière

Photorésistances (LDR)

Composants passifs dont la résistance varie inversement à l'intensité lumineuse.

- ✓ Faible coût
- ✓ Sortie Analogique

Photodiodes

Semi-conducteurs convertissant la lumière en courant électrique avec une grande rapidité.

- ✓ Haute précision
- ✓ Temps de réponse court

Capteurs Numériques

Circuits intégrés (ex: TSL2561) fournissant une mesure directe en Lux via I2C/SPI.

- ✓ Sortie Digitale
- ✓ Plage dynamique étendue

Applications IoT Courantes

- Contrôle d'éclairage intelligent
- Surveillance agricole (Ensoleillement)
- Détection d'intrusion & Sécurité
- Ajustement automatique d'affichage

NOTE TECHNIQUE

Le choix dépend de la **plage spectrale** (visible vs IR) et de la **sensibilité** requise pour l'environnement cible.

- Typologie des capteurs
- **Principes de fonctionnement**
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Capteurs de Mouvement et Proximité

Ces dispositifs permettent à l'objet connecté de percevoir son environnement spatial, de détecter des présences ou de mesurer des distances avec précision.

PIR (Infra-rouge Passif)

Détecte les variations de rayonnement infrarouge (chaleur) émises par les corps en mouvement.

USAGE PRINCIPAL

Sécurité & Domotique

Ultrasonique (HC-SR04)

Mesure la distance en calculant le temps de vol d'une onde sonore haute fréquence (Écho).

USAGE PRINCIPAL

Robotique & Niveaux

Infrarouge (Proximité)

Utilise un faisceau lumineux pour détecter la présence d'un obstacle à courte portée.

USAGE PRINCIPAL

Évitement d'obstacles

Sortie Numérique (PIR) | Signal PWM / Temps (Ultrason) | Analogique ou Numérique (IR)

- Typologie des capteurs
- **Principes de fonctionnement**
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Pression et Force

Les capteurs de pression et de force servent à mesurer les interactions mécaniques (force, poids, compression, traction) ainsi que les conditions atmosphériques. Ils sont utilisés dans des systèmes nécessitant précision et fiabilité.

Jauges de Contrainte

Utilisées pour mesurer la déformation d'un objet sous l'effet d'une force ou d'un **poids**.

APPLICATIONS
- **Balances électroniques**
- **Capteurs de poids**
- **Tests mécaniques des matériaux**
- **Surveillance des structures (ponts, bâtiments)**

Piézorésistifs

Détectent les changements de résistance électrique causés par une pression mécanique appliquée.

APPLICATIONS
- **Mesure de pression dans les pneus**
- **Dispositifs médicaux (pression sanguine)**
- **Systèmes automobiles**
- **Industrie**

Baromètres

Mesurent la pression atmosphérique pour le suivi météorologique ou l'estimation d'altitude.

APPLICATIONS
- **Prévisions météorologiques**
- **Estimation d'altitude (altimètres)**
- **Stations météo**
- **Drones et aviation**

- Typologie des capteurs
- **Principes de fonctionnement**
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Capteurs Chimiques & Qualité de l'Air

Ces capteurs détectent la présence et la concentration de substances chimiques spécifiques dans l'air ou les liquides, essentiels pour la **sécurité industrielle** et la **protection de l'environnement**.

Détection de Gaz

- **CO2 :**Qualité de l'air intérieur
- **CO :**Sécurité (Monoxyde)
- **NH3 :**Surveillance agricole

Qualité des Liquides

- **pH :**Acidité/Alcalinité
- **Conductivité :**Pureté de l'eau
- **O2 Dissous :**Aquaponie

Cas d'Usage Clés

INDUSTRIE
Détection de fuites toxiques ou inflammables.

ENVIRONNEMENT
Monitoring de la pollution urbaine (Smart City).

SANTÉ
Ventilation intelligente des bâtiments (HVAC).

** Les séries MQ (MQ-2, MQ-135) sont couramment utilisées avec ESP32 pour le prototypage.*

Sûreté *Environnement*

- Typologie des capteurs
- Principes de fonctionnement
- **Critères de sélection des capteurs**
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

1.4. Caractéristiques Techniques des Capteurs

Précision(Accuracy)

Capacité du capteur à donner une mesure proche de la valeur réelle de la grandeur physique.

PERFORMANCE CLÉ

Résolution

La plus petite variation de la grandeur physique que le capteur est capable de détecter.

SEUIL DE DÉTECTION

Linéarité

Constance du rapport entre la variation de la sortie et la variation de l'entrée sur toute la plage.

PROPORTIONNALITÉ

Temps de réponse

rapidité avec laquelle le capteur réagit à un changement de la grandeur mesurée.

RÉACTIVITÉ

Plage de mesure (Range)

Intervalle entre les valeurs minimale et maximale que le capteur peut mesurer avec précision.

LIMITES OPÉRATIONNELLES

Conditions Limites

Facteurs environnementaux (Température, humidité, chocs) influençant la fiabilité du capteur.

ENVIRONNEMENT

Sensibilité

Variation du signal de sortie par rapport à la variation de la grandeur mesurée S=ΔSortie/ΔEntreé.

Exemple : 41 µV/°C pour un thermocouple type K.

Consommation énergétique

Puissance électrique nécessaire au fonctionnement du capteur.

EFFICACITÉ ÉNERGÉTIQUE

Stabilité thermique

Influence de la température ambiante sur la précision et la stabilité de la mesure.

INFLUENCE ENVIRONNEMENTALE

- Typologie des capteurs
- Principes de fonctionnement
- **Critères de sélection des capteurs**
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

1.4. Critères de Sélection : Performance

Le choix d'un capteur dépend de sa capacité à répondre aux exigences techniques de l'application. La **performance** est définie par quatre piliers fondamentaux.

Précision

Écart maximal entre la valeur mesurée et la valeur réelle de la grandeur physique. Crucial pour le contrôle industriel.

Résolution

Plus petite variation de la grandeur physique que le capteur est capable de détecter et de traduire en signal.

Vitesse

Fréquence d'échantillonnage permettant de capturer les phénomènes transitoires rapides (ex: vibrations).

Stabilité

Capacité du capteur à maintenir ses caractéristiques métrologiques constantes dans le temps (dérive).

" *Le surdimensionnement de la performance augmente inutilement le coût et la complexité du traitement.* CONSEIL D'EXPERT

1.4. Critères de Sélection : Environnement

> Le choix d'un capteur dépend impérativement du milieu dans lequel il sera déployé. Un capteur inadapté peut entraîner des erreurs de mesure ou une défaillance prématurée.

Température
Plages de fonctionnement critiques (Indus: -40°C à +85°C).

Humidité
Résistance à la condensation et protection contre la corrosion interne.

Vibrations & Chocs
Stabilité mécanique pour les applications industrielles ou mobiles.

Étanchéité (IP)
Indices de protection (IP65/IP67) contre la poussière et l'eau.

Conditions Extrêmes
Exposition aux UV, aux agents chimiques ou aux bruits EM.

Durabilité
Vieillissement des matériaux face aux agressions environnementales.

Conseil : Toujours vérifier la fiche technique (Datasheet) pour les limites de fonctionnement non destructives.

- Typologie des capteurs
- Principes de fonctionnement
- **Critères de sélection des capteurs**
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

1.4. Critères de Sélection : Économiques

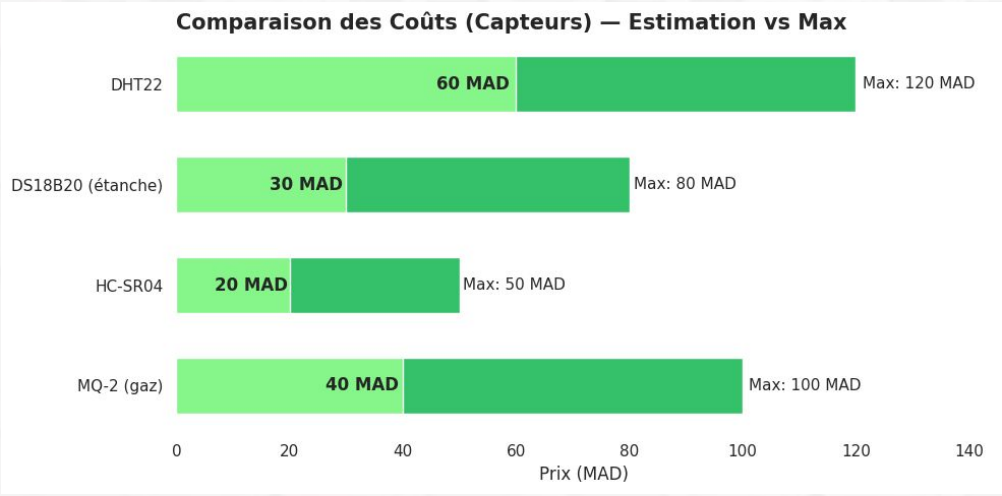
Prix d'Achat
Coût initial du matériel. Impact direct sur le budget du prototype ou de la série.

Maintenance
Coûts opérationnels, mises à jour et remplacement des unités défectueuses.

Durée de Vie
Rentabilité sur le long terme (ROI) basée sur la robustesse du composant.

Disponibilité
Stabilité de la chaîne d'approvisionnement pour éviter les ruptures de stock.

Comparaison des Coûts (Estimation)



Analyse Coût-Bénéfice
Le choix ne doit pas se limiter au prix d'achat. Un dispositif moins cher (ex: ESP32) peut nécessiter plus de temps de développement, tandis qu'un dispositif complet (ex: Raspberry Pi) réduit le temps de mise sur le marché mais augmente le coût unitaire.

- Optimisation des coûts pour les déploiements de masse.
- Évaluation du TCO (Total Cost of Ownership).

1.4. Critères de Sélection : Intégration du Système

> L'intégration réussie d'un capteur dépend de sa capacité à s'insérer harmonieusement dans l'écosystème matériel et logiciel de la solution IoT.

Compatibilité Électrique

- ✓ Niveaux de tension (3.3V vs 5V)
- ✓ Consommation en veille et active

Facteurs Mécaniques

- ✓ Dimensions et encombrement
- ✓ Type de montage (SMD, Through-hole)

Protocoles de Com.

- ✓ Interfaces (I2C, SPI, UART, One-Wire)
- ✓ Complexité de l'implémentation logicielle

Facilité d'Intégration

- ✓ Disponibilité des bibliothèques (drivers)
- ✓ Documentation et support communauté

- Typologie des capteurs
- Principes de fonctionnement
- **Critères de sélection des capteurs**
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

1.4. Sélection par Application : Domotique

En domotique, le choix des capteurs privilégie **l'intégration invisible**, la **faible consommation** et la **fiabilité** pour le confort et la sécurité.

Confort Thermique

- DHT22 / BME280
- Mesure Température & Humidité
- Pilotage Chauffage / Clim

Sécurité & Présence

- PIR (Mouvement)
- Contacteurs Magnétiques (Ouverture)
- Détection Intrusion / Éclairage Auto

Qualité de l'Air

- MQ-135 (CO2 / Gaz)
- PM2.5 (Particules)
- Ventilation Automatisée

Gestion Énergie

- SCT-013 (Courant Non-Invasif)
- Mesure Consommation Temps Réel
- Optimisation des Coûts

Éclairage

- LDR (Photorésistance)
- Gestion des Volets Roulants
- Adaptation Scénarios Jour/Nuit

CRITÈRES CLÉS

Wi-Fi / Zigbee **Alim. Batterie** **Design Compact**

- Typologie des capteurs
- Principes de fonctionnement
- **Critères de sélection des capteurs**
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

1.4. Sélection par Application : Industrie

Dans le contexte industriel (IIoT), les capteurs doivent répondre à des exigences de robustesse, précision et fiabilité extrêmes pour assurer la surveillance continue des machines et la maintenance prédictive.

Vibration
Détection précoce d'usure des roulements, déséquilibres et défauts mécaniques sur moteurs.

Température
Surveillance des points de chauffe critiques et contrôle des processus thermiques industriels.

Pression
Contrôle des systèmes hydrauliques, pneumatiques et étanchéité des réservoirs.

Débit
Mesure de consommation de fluides (air, gaz, eau) pour l'optimisation énergétique.

Défauts
Capteurs spécialisés pour la détection de fuites, arcs électriques ou anomalies de surface.

STANDARD INDUSTRIEL

IP67 / IP69K

Résistance aux chocs, poussières et produits chimiques.

- Typologie des capteurs
- Principes de fonctionnement
- **Critères de sélection des capteurs**
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

1.4. Sélection par Application : Agriculture

L'agriculture de précision s'appuie sur une acquisition de données multicritères pour optimiser les ressources, améliorer les rendements et assurer le bien-être animal.

Irrigation Intelligente

Capteurs d'humidité du sol (capacitifs/résistifs)

OPTIMISATION EAU

Météorologie Localisée

Température, hygrométrie, vitesse du vent et pluviométrie

PRÉVISIONS LOCALES

Qualité des Sols

Mesure du pH, salinité et nutriments (NPK)

SANTÉ DES CULTURES

Suivi du Bétail

Accéléromètres (mouvement), température corporelle et GPS

BIEN-ÊTRE ANIMAL

-30% CONSOMMATION D'EAU

+20% RENDEMENT AGRICOLE

24/7 SURVEILLANCE AUTOMATISÉE

Comment choisir un capteur ?

Critères de sélection :

Précision : écart entre valeur mesurée et valeur réelle

Résolution : plus petite variation détectable

Temps de réponse : délai avant stabilisation

Plage de mesure : min/max supportés

Dérive thermique : variation avec la température

Consommation : courant en actif et en veille

Coût : budget disponible

Interface : I2C, SPI, UART, analogique...

Comparaison de capteurs de température :

Critère	DHT22	DS18B20	LM35
Type	Numérique	Numérique	Analogique
Précision	±0.5°C	±0.5°C	±1°C
Interface	1-Wire	1-Wire	Analogique
Consommation	1.5 mA	1 mA	60 µA
Coût	~2€	~2€	~1€

2. Les Actionneurs:

2.1 Définition

Un **actionneur** est un dispositif qui reçoit un signal de commande et produit une action physique telle qu'un **mouvement, une vibration, un changement de position ou d'état**.

Exemples d'actionneurs :

- **LED** : Signalisation d'un événement (ex. : alarme)
- **Moteur électrique** : Contrôle du mouvement
- **Électrovanne** : Gestion des flux de liquides ou de gaz
- **Haut-parleur / BUZZER** : Émission de sons ou d'alertes
- **Affichage LCD/OLED** : Visualisation d'informations

Exemple d'application IoT :

- Un **capteur de température** détecte une **température élevée**.
- Un **système IoT envoie une commande** à un **actionneur**.
- L'actionneur active un **ventilateur** ou un **climatiseur** pour réduire la température.

Les capteurs **collectent** les informations, et les actionneurs **réagissent** en conséquence.

2.2 Caractéristiques des actionneurs

- **Type d'entrée** : électrique, pneumatique, hydraulique.
- **Type de sortie** : mouvement linéaire, rotation, émission de lumière, chaleur, etc.
- **Puissance** : quantité d'énergie consommée par l'actionneur.
- **Temps de réponse** : rapidité avec laquelle l'actionneur exécute une action.
- **Efficacité** : rapport entre la puissance de sortie et la puissance absorbée.

2.3 Classification des actionneurs

2.3.1 Selon le type d'énergie utilisée

Type d'actionneur	Énergie utilisée	Exemple
Actionneurs électriques	Électricité	Moteurs DC, moteurs pas à pas, relais
Actionneurs pneumatiques	Air comprimé	Vérins pneumatiques
Actionneurs hydrauliques	Fluide sous pression	Moteurs hydrauliques

2.3.2 Selon la nature de l'action

Type d'actionneur	Action effectuée	Exemple
Moteurs électriques	Mouvement rotatif	Moteur DC, servo-moteur, moteur pas à pas
Relais et contacteurs	Changement d'état électrique	Relais 5V, contacteur industriel
Électro-aimants	Génération d'un champ magnétique	Serrures magnétiques
LED et écrans	Emission de lumière	LED, écrans OLED
Résistances chauffantes	Production de chaleur	Résistances de chauffe, Peltier

3. Capteurs et Actionneurs dans un Système IoT

Dans un système **IoT (Internet des Objets)**, les capteurs et actionneurs sont **interconnectés** pour automatiser des processus et transmettre des données vers des serveurs ou applications.

3.1 Fonctionnement d'un Système IoT

1. **Un capteur collecte une donnée** (température, humidité, luminosité, etc.).
2. **Un microcontrôleur ou microprocesseur traite cette donnée** (exemple : Arduino, ESP32, Raspberry Pi).
3. **Un actionneur réagit en conséquence** (moteur activé, LED allumée, alarme déclenchée).
4. **Les données sont transmises** à un serveur ou une application (via Wi-Fi, LoRa, Bluetooth, 5G, etc.).

3.2 Exemples d'application

- **Domotique** : Détection de mouvement (capteur PIR) → Allumage automatique des lumières (LED).
- **Agriculture intelligente** : Capteur d'humidité du sol → Activation d'un arrosage automatique (électrovanne).
- **Surveillance industrielle** : Capteur de gaz (MQ-2) → Activation d'une alarme sonore.
- **Médecine connectée** : Capteur de fréquence cardiaque → Alerte envoyée en cas d'anomalie.

Introduction aux dispositifs programmables

Le dispositif programmable est l'intermédiaire entre monde physique et monde numérique.

Il existe plusieurs plateformes de développement IoT (Internet of Things) qui facilitent la création, le déploiement et la gestion d'applications IoT. Voici quelques-unes des plus populaires :

1. Plateformes Matériel (Hardware)

Ce sont des cartes de développement et des modules qui permettent de créer des **prototypes** et des produits IoT :

- **Arduino** : Facile à utiliser pour les débutants, avec un large écosystème de modules et de capteurs.
- **Raspberry Pi** : Mini-ordinateur polyvalent pour des applications IoT avancées avec Linux.
- **ESP32 / ESP8266** : Modules Wi-Fi/ Bluetooth économiques et puissants pour des applications IoT connectées.
- **STM32** : Microcontrôleurs robustes pour des applications industrielles et en temps réel.
- **BeagleBone** : Puissant comme le Raspberry Pi mais avec plus de ports d'E/S pour des applications industrielles.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

*"Les dispositifs programmables constituent le **cœur intelligent** des objets connectés, agissant comme le cerveau qui traite les données et gère les communications."*

Micro-ordinateurs (SBC)

- Puissance de calcul élevée (Multi-cœur)
- Applications multiples
- Système d'exploitation complet
- Mémoire importante (Go)
- Comparable à un petit PC
- Système d'exploitation complet (Linux)
- **Exemple : Raspberry Pi, BeagleBone.**

TRAITEMENT LOCAL & PASSERELLES

Microcontrôleurs (MCU)

- Très faible consommation d'énergie
- Tâche spécifique et répétitive
- Aucun système d'exploitation
- Exécution en temps réel
- Mémoire limitée (Ko à Mo)
- Optimisé pour fonctions dédiées

Exemple : Arduino,ESP8266,ESP32,STM32 .

NŒUDS DE CAPTEURS & AUTONOMIE

⚡ Énergie | Calcul | Connectivité

les bus de communication

- Un **microcontrôleur 8 bits** (ex. AVR ATmega328 dans Arduino Uno) a un bus de données de 8 bits.
- Un **microcontrôleur 16 bits** (ex. MSP430 de Texas Instruments) est souvent utilisé dans des applications basse consommation.
- Un **microcontrôleur 32 bits** (ex. STM32, ESP32) est plus performant et peut exécuter du code plus rapidement avec une mémoire plus grande.
- Un **microcontrôleur 64 bits** (ex. Raspberry Pi 4 avec Cortex-A72) est rare dans l'embarqué mais utilisé pour les applications avancées (intelligence artificielle, vision par ordinateur).

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Arduino
Description : plateforme de **prototypage open-source** avec un IDE facile à utiliser.

Utilisation : convient pour des **projets simples de capteurs et actionneurs**.

Exemple : un système de détection de mouvement avec un capteur PIR et un buzzer connecté à un Arduino Uno.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Présentation de la plateforme Arduino

Arduino est le nom d'une entreprise, une communauté et un projet libre **(open- source).** Ce projet avait été initié en **Italie**, dans le but de concevoir et fabriquer des **solutions matérielles** et **logiciel** à **faible cout** qui **faciliteront le développement d'objets interactifs qui peuvent interagir avec le monde physique**.

Les cartes Arduino peuvent être **achetées préassemblée**, ou réalisés par soi-même (RSM). **Les schémas de conception du matériel sont disponibles pour ceux qui voudraient assembler un Arduino à partir de zéro.**

Le microcontrôleur Arduino est un outil facile à utiliser. Ce puissant ordinateur à carte unique a réussi à se faire une bonne place dans le marché des amateurs ainsi que dans le marché des professionnels et académiques. Les cartes Arduino sont disponibles en plusieurs saveurs :

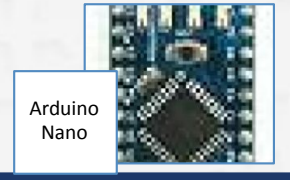

Arduino Uno


Arduino Mega


Arduino Leonardo


Arduino Due


Arduino YUN


Arduino Nano

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Présentation de la plateforme Arduino

Principales différences :
Taille et format :

- **Arduino Uno, Mega, Leonardo, Due, et YUN** sont des cartes de taille standard.
- **Arduino Nano** est une version compacte adaptée aux projets où l'espace est limité.

Puissance et performances :

- **Arduino Mega** est une version plus puissante de l'Uno avec plus d'entrées/sorties et de mémoire.
- **Arduino Due** utilise un microcontrôleur ARM 32 bits, plus puissant que les ATmega 8 bits.

Connectivité :

- **Arduino YUN** possède une connexion Wi-Fi et Ethernet pour les projets IoT.

Alimentation :

- **Arduino Due** fonctionne en **3.3V**, contrairement aux autres cartes qui sont en **5V**.

Quel Arduino choisir ?
Débutants : Arduino Uno (simple et bien documenté)
Projets nécessitant plus d'entrées/sorties et de mémoire : Arduino Mega
Projets USB (émulation de clavier/souris) : Arduino Leonardo
Projets IoT avec Wi-Fi et Linux : Arduino YUN
Projets compacts : Arduino Nano
Projets nécessitant plus de puissance de calcul : Arduino Due (ARM 32 bits)

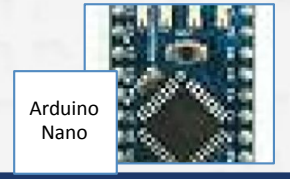

Arduino Uno


Arduino Mega


Arduino Leonardo


Arduino Due


Arduino YUN


Arduino Nano

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Présentation de la plateforme Arduino

Pour la programmation des microcontrôleurs, la plateforme Arduino fournit un environnement de développement intégré (IDE) qui prend en charge les langages de programmation C et C++.
La carte Arduino UNO, qui est une version populaire de la plateforme, est basée sur le microcontrôleur ATmega328. Elle fonctionne à 5V et possède les caractéristiques suivantes :

2 Ko de RAM,
32 Ko de mémoire flash pour le stockage des programmes,
1 Ko de mémoire EEPROM pour le stockage des paramètres.

La vitesse d'horloge de la carte est de 16 MHz, ce qui permet d'exécuter environ 300 000 lignes de code source C par seconde. Un connecteur USB permet de connecter la carte à un ordinateur hôte, tandis qu'une prise d'alimentation DCpermet de connecter une source d'alimentation externe de 7 à 12V (comme une batterie de 7.4V) pour faire fonctionner le programme sans être connecté à un ordinateur hôte.

Ainsi, la carte Arduino UNO est idéale pour des projets autonomes et des applications nécessitant un microcontrôleur avec des capacités de programmation simples mais puissantes.


Arduino Uno


Arduino Mega


Arduino Leonardo


Arduino Due


Arduino YUN

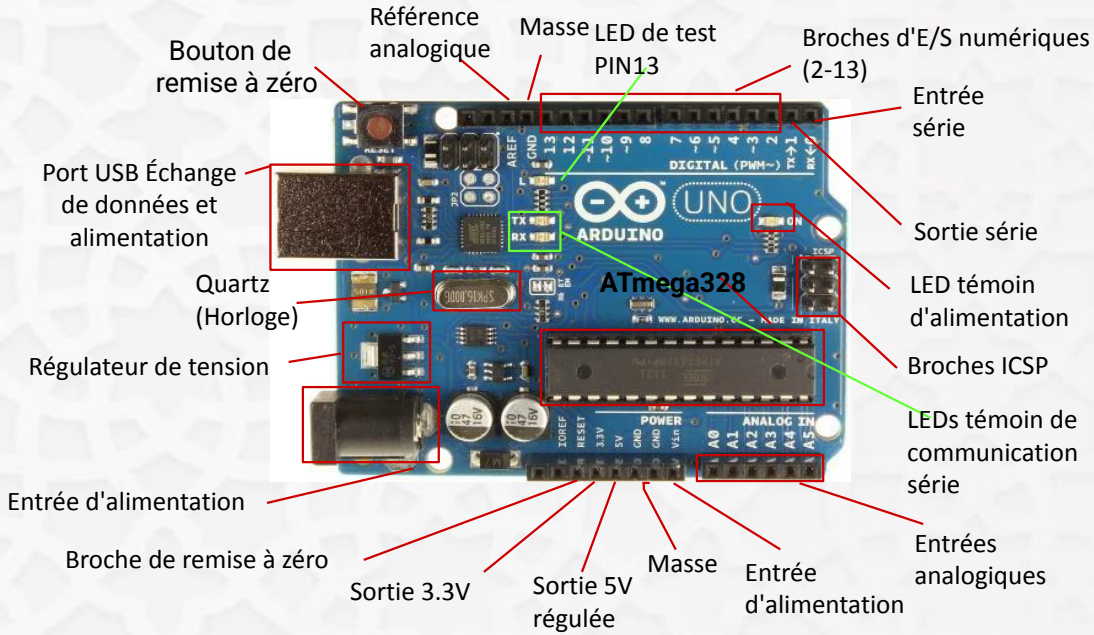

Arduino Nano

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Présentation de la plateforme Arduino

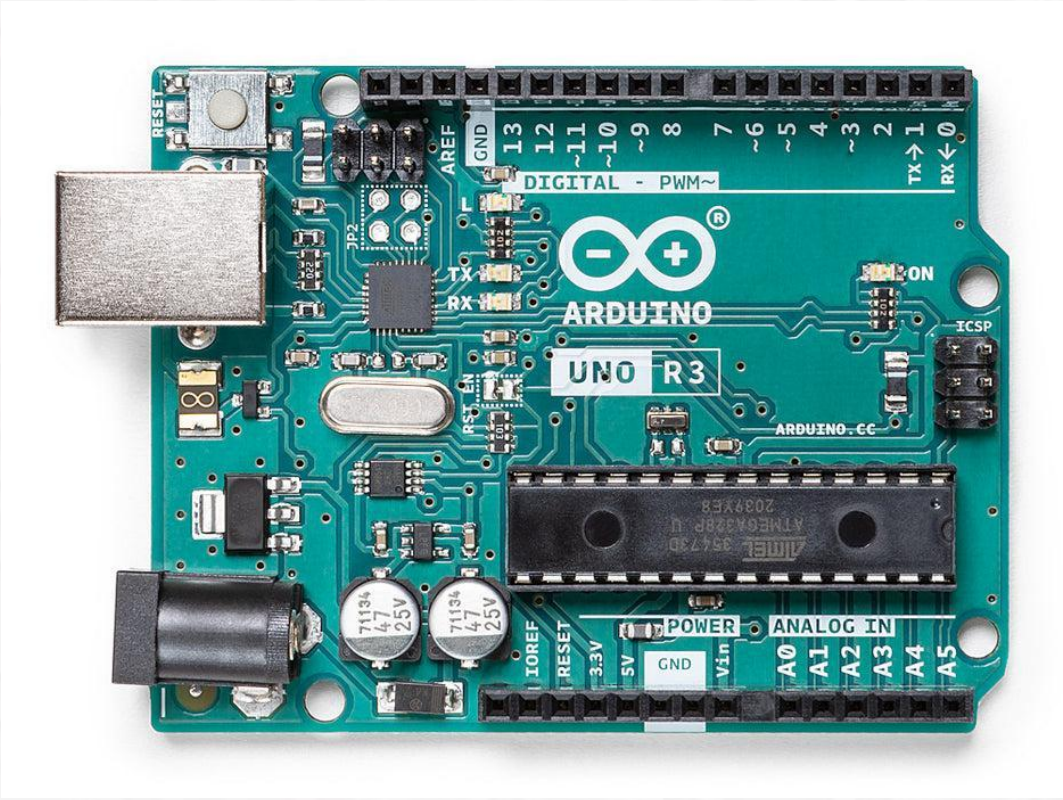
L'Arduino UNO en chiffres

- **Microcontrôleur** : ATmega328
- **Tension de fonctionnement** : 5V
- **Tension d'entrée (recommandée)** : 7-12V
- **Tension d'entrée (limites)** : 6-20V
- **Broches d'E/S numériques** : 14 (dont 6 peuvent fournir une sortie PWM)
- **Broches d'entrée analogiques** : 6
- **Courant DC par broche d'E/S** : 40 mA
- **Courant DC de la sortie 3,3V** : 50 mA
- **Mémoire Flash** : 32 Ko (dont 0,5 Ko utilisé par le bootloader)
- **SRAM** : 2 Ko
- **EEPROM** : 1 Ko
- **Vitesse d'horloge** : 16 MHz
- **Dimensions** : 68,6 mm x 53,4 mm
- **Poids** : 25 g



- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Arduino : Architecture et spécifications de l'UNO R3



Tech specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Arduino : Architecture et spécifications de l'UNO R4

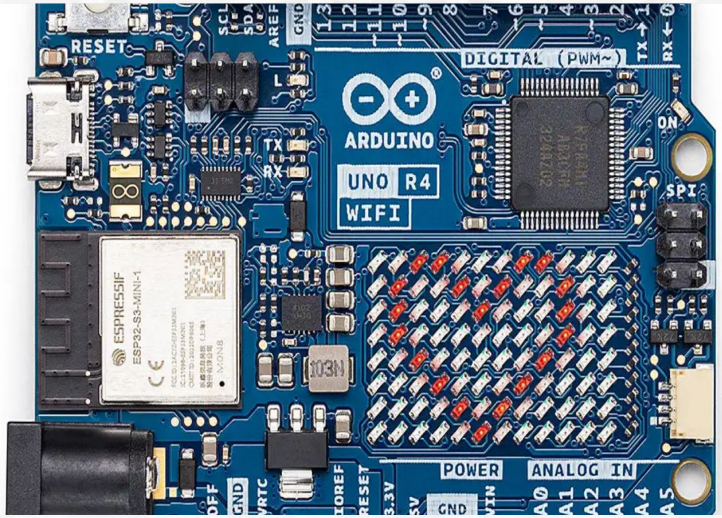
Plateforme MCU 32 bits

Architecture Renesas

Processeur	RA4M1 (Cortex-M4)
Vitesse	48 MHz (32 bits)
Flash / SRAM	256 KB / 32 KB
ADC	14 bits
Tension I/O	5V

Évolution Majeure

Le passage de l'AVR 8 bits au Renesas 32 bits décuple la puissance de calcul tout en maintenant la compatibilité matérielle avec les shields classiques.



Points Forts

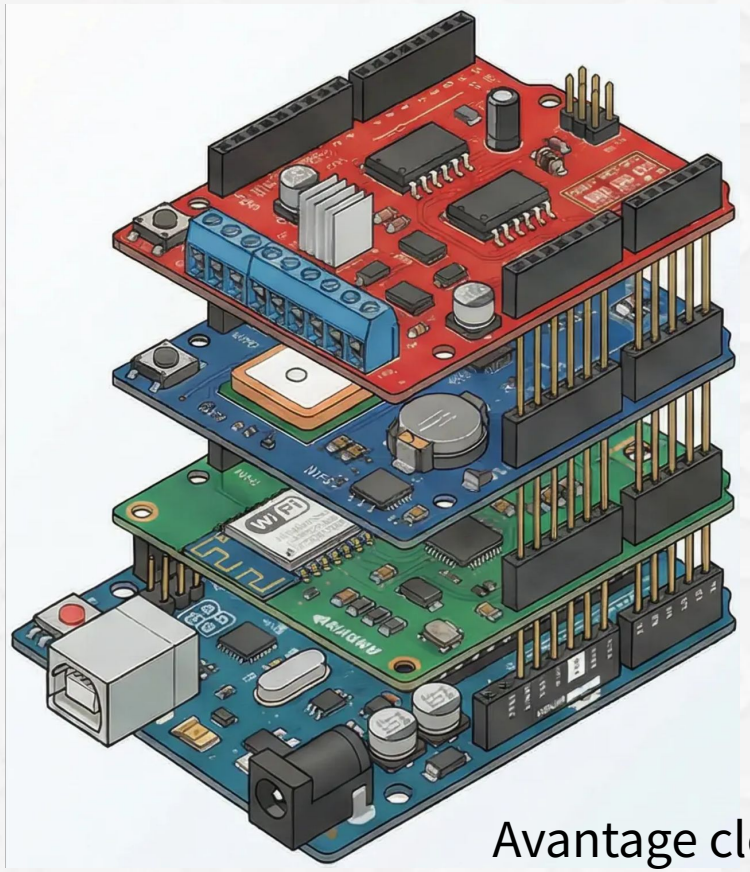
- ✓ Simplicité de l'IDE Arduino
- ✓ Version WiFi (ESP32-S3)
- ✓ Matrice LED 12x8 intégrée

Limitations

- ✗ Pas de multitâche complexe
- ✗ Mémoire SRAM limitée (32KB)
- ✗ Pas d'OS (temps réel pur)

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

L'écosystème des "Shields"



- **Qu'est-ce qu'un Shield ?**
  - Carte d'extension qui s'empile sur l'Arduino
  - Connexion directe via les broches
  - Ajout de fonctions sans câblage complexe
- **Types de Shields populaires :**
  - WiFi Shield - Connectivité sans fil
  - GPS Shield - Géolocalisation
  - Motor Shield - Contrôle de moteurs
  - LCD Shield - Affichage
  - Ethernet Shield - Connexion réseau

Avantage clé : Plug & Play - Aucune soudure requise !

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- **Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)**
- Installation et configuration des environnements de développement
- Lecture des données issues des capteurs et transmission des données

Avantages et limites d'Arduino

✔ AVANTAGES

- Facilité d'apprentissage
- Prix abordable
- Communauté active
- Écosystème riche (shields)
- Open-source complet
- Idéal pour débuter

⚠️ LIMITES

- Puissance de calcul limitée
- Mémoire restreinte
- Pas de connectivité native
- Pas de système d'exploitation
- Moins adapté aux projets complexes

Arduino excelle pour l'apprentissage et les projets simples à moyens

ESP32

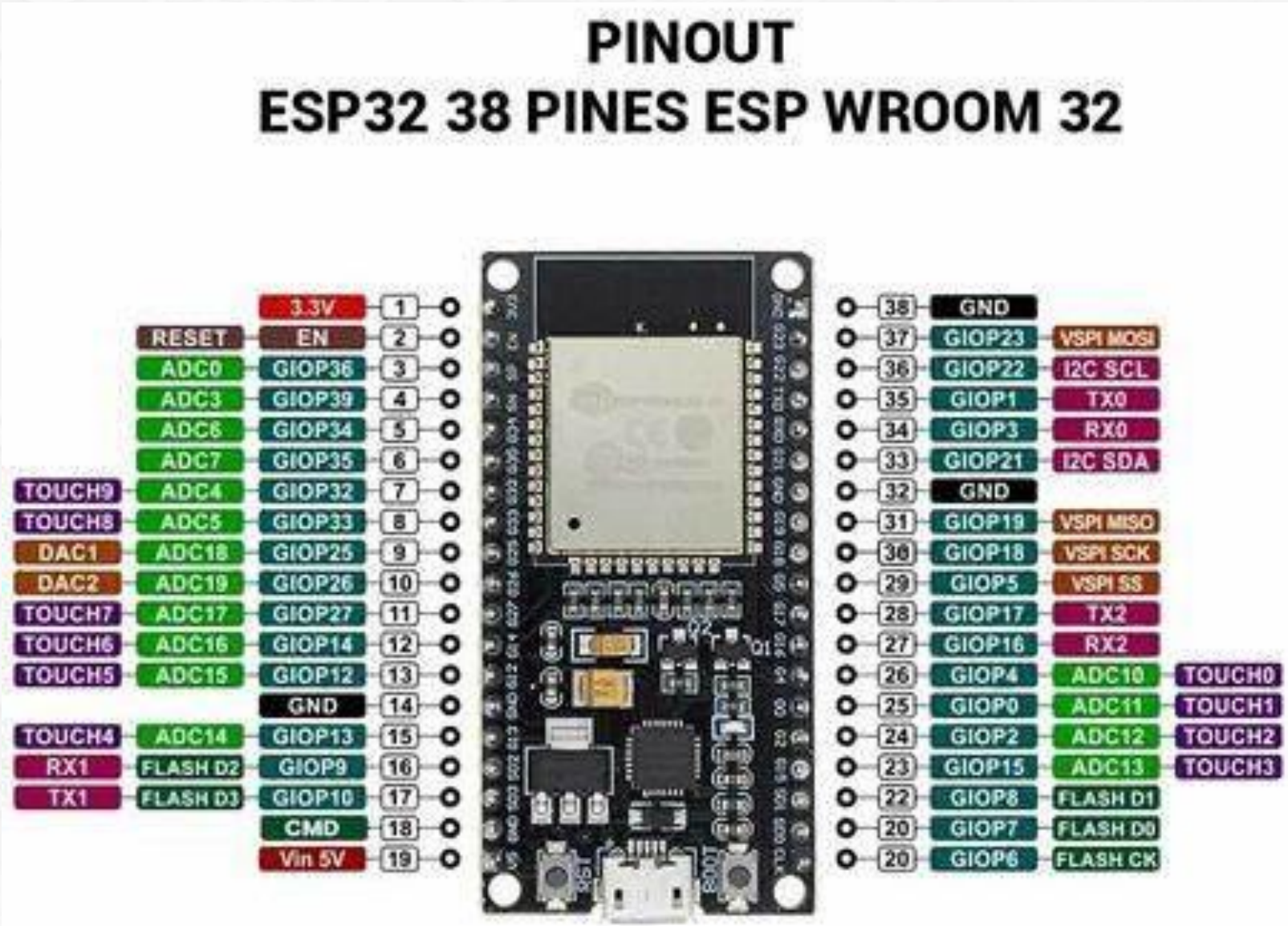
Description : microcontrôleur puissant avec Wi-Fi et Bluetooth intégrés.
Utilisation : parfait pour des projets IoT nécessitant une connectivité réseau et des performances élevées.
Exemple : Un système de surveillance de l'humidité du sol avec notifications sur application mobile.

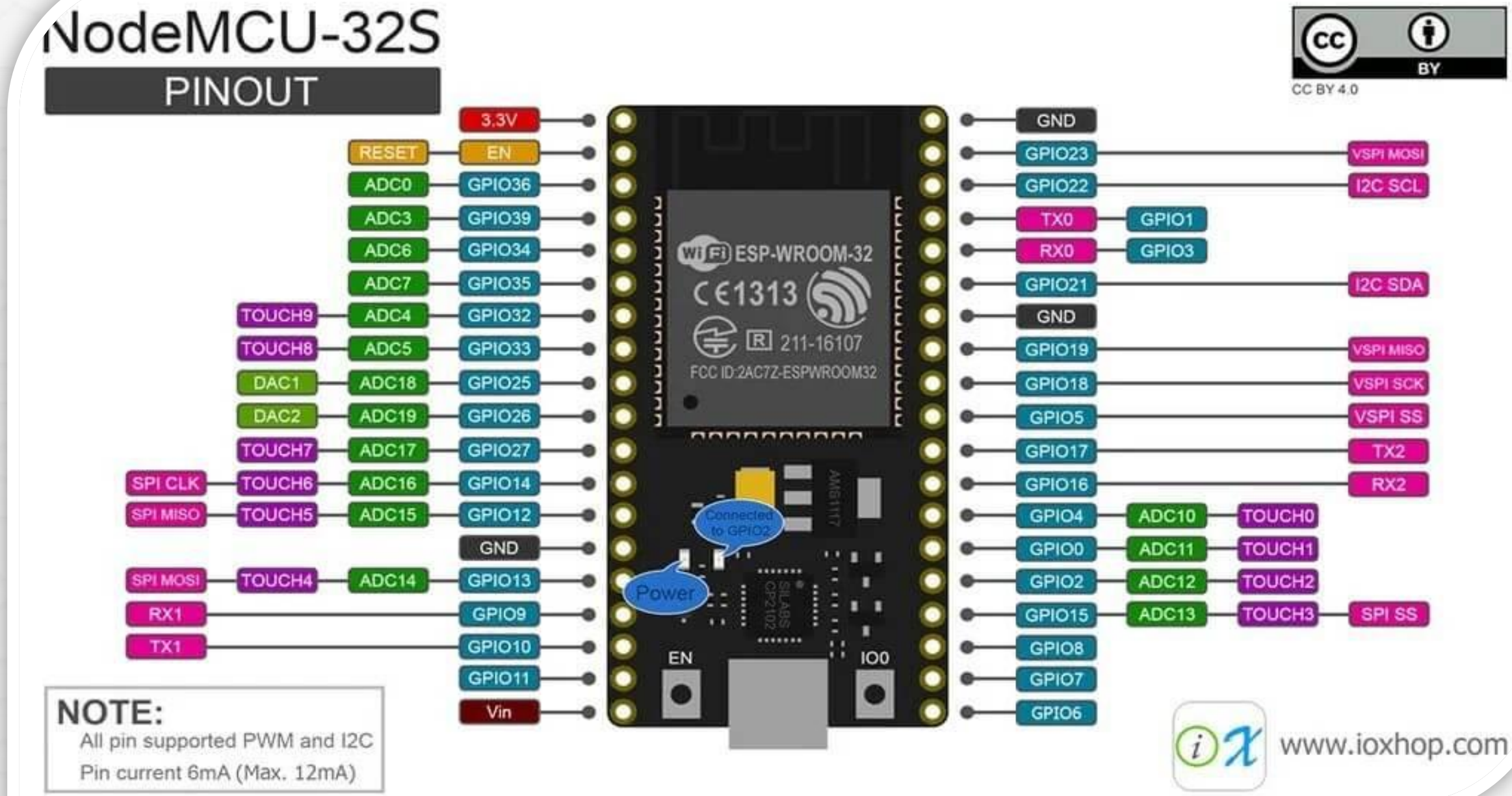
Fonctionnalités clés :

- Double cœur (Dual Core) pour le traitement parallèle.
- Support pour le chiffrement SSL/TLS pour des communications sécurisées.

ESP32


NODEMCU
ESP-32S
V1.1
101010


PINOUT
ESP32 38 PINES ESP WROOM 32
3.3V 1
RESET EN 2
ADC0 GIOP36 3
ADC3 GIOP39 4
ADC6 GIOP34 5
ADC7 GIOP35 6
TOUCH9 ADC4 GIOP32 7
TOUCH8 ADC5 GIOP33 8
DAC1 ADC18 GIOP25 9
DAC2 ADC19 GIOP26 10
TOUCH7 ADC17 GIOP27 11
TOUCH6 ADC16 GIOP14 12
TOUCH5 ADC15 GIOP12 13
GND 14
TOUCH4 ADC14 GIOP13 15
RX1 FLASH D2 GIOP9 16
TX1 FLASH D3 GIOP10 17
CMD 18
Vin 5V 19
38 GND
37 GIOP23 VSPI MOSI
36 GIOP22 I2C SCL TX0
35 GIOP1 TX0
34 GIOP3 RX0
33 GIOP21 I2C SDA
32 GND
31 GIOP19 VSPI MISO
30 GIOP18 VSPI SCK
29 GIOP5 VSPI SS
28 GIOP17 TX2
27 GIOP16 RX2
26 GIOP4 ADC10 TOUCH0
25 GIOP0 ADC11 TOUCH1
24 GIOP2 ADC12 TOUCH2
23 GIOP15 ADC13 TOUCH3
22 GIOP8 FLASH D1
20 GIOP7 FLASH D0
20 GIOP6 FLASH CK



Nodemcu esp-32s

ESP32 : Connectivité IoT et Puissance de Calcul

Focus sur le module ESP32-S3 : Performance, WiFi et Basse Consommation

Architecture Double Cœur

Processeur **Xtensa® 32-bit LX7** cadencé jusqu'à 240 MHz. Capacité de traitement parallèle pour gérer simultanément la pile réseau et l'application utilisateur.

Connectivité Sans Fil Native

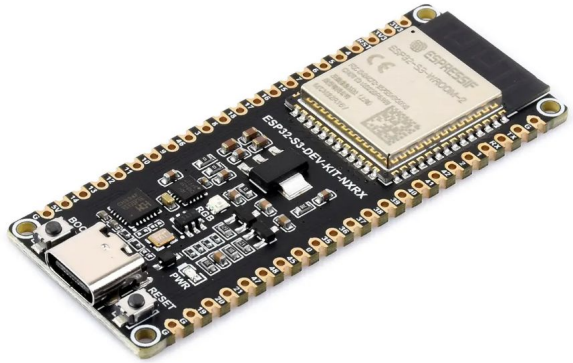
Intégration du **Wi-Fi 2.4 GHz (802.11 b/g/n)** et du **Bluetooth 5 (LE)**. Idéal pour les passerelles IoT et les capteurs connectés.

Gestion de l'Énergie

Conçu pour les applications sur batterie avec des modes **Deep Sleep** avancés. Consommation minimale tout en maintenant les fonctionnalités critiques.

Écosystème de Développement

Support complet de l'**ESP-IDF** (FreeRTOS) pour les professionnels et de l'**IDE Arduino** pour le prototypage rapide.

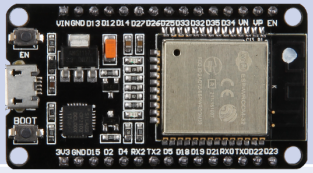
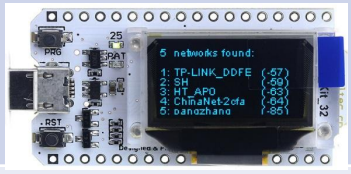
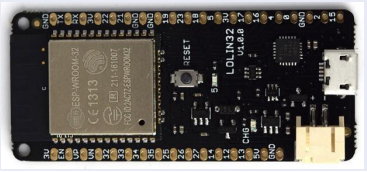


ESP32-S3-DEVKITC-1

Module WROOM avec interfaces USB-C et JTAG

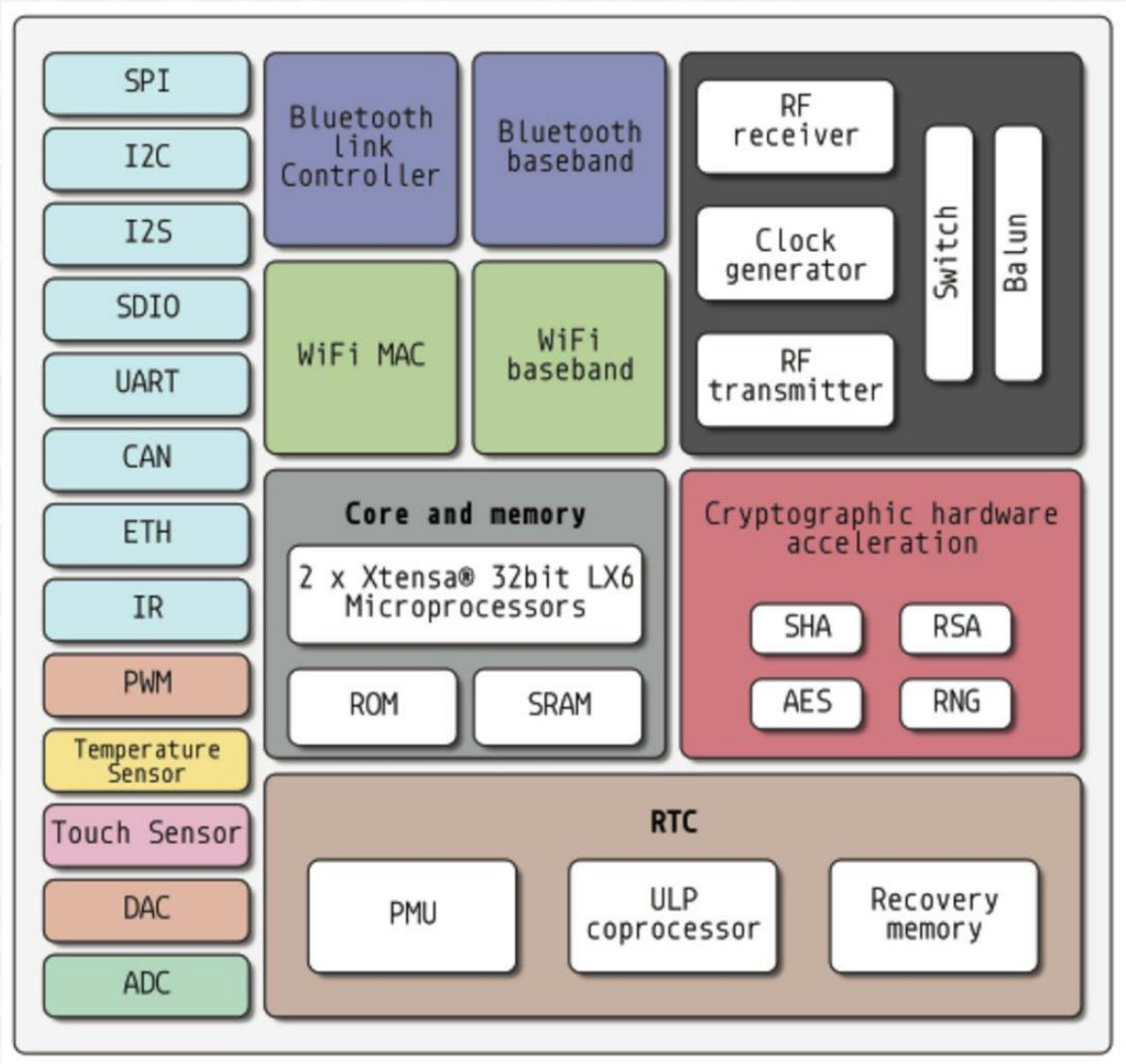
RAM: 512 KB SRAM / 8 MB PSRAM **I/O:** 45 GPIOs programmables **Sécurité:** Accélération matérielle AES/RSA

ESP32 boards

Figure	Name	GPIOS	ON-Board Buttons
	ESP 32 DEV KIT DOIT	30/36	EN and BOOT
	Adfruit HUZZAH32	28	RESET
	SX1278 ESP32 (LORA)	26	EN and BOOT
	LOLIN32 w/battery holder	40	EN and BOOT
	ESP32 LOLIN OLED	26	EN and BOOT

3. ESP32 CHIP SPECIFICATION

SPES/BOARD	ESP32	ESP8266	Arduino Uno
Number of cores	2	1	1
Architecture	32 Bit	32 Bit	8 Bit
CPU Frequency	160 or 240MHZ	80 MHZ	16 MHZ
WIFI	yes	yes	No
Bluetooth	yes	No	No
Ram	512 KB	160KB	2kB
Flash	16 MB	16 MB	32KB
GPIO	36 or 30	17	14
Busses	SPI, I2C, UART, I2S, CAN	SPI, I2C, UART, I2S	SPI, I2C, UART
ADC Pins	18	1	6
DAC Pins	2	0	0



ESP32 Architecture

FreeRTOS : Système d'exploitation pour l'IoT

FreeRTOS est un **système d'exploitation temps réel (RTOS) léger** conçu spécifiquement pour les microcontrôleurs (MCU). Il fournit les outils nécessaires pour gérer plusieurs tâches concurrentes sur des systèmes embarqués aux ressources limitées, rendant le développement d'applications IoT complexes plus structuré et efficace.

Gestion Multitâche

Permet l'exécution de plusieurs fonctions (tâches) **simultanément** (par exemple, lecture capteur, envoi réseau, clignotement LED).

Faible Empreinte

Optimisé pour les MCU avec des quantités limitées de RAM et de mémoire flash, assurant une **consommation énergétique minimale**.

Temps Réel Déterministe

Garantit que les opérations critiques sont exécutées dans des délais **prévisibles et reproductibles**, essentiel pour le contrôle et la réactivité.

Connectivité et Middleware

Facilite l'intégration de piles réseau (TCP/IP), de sécurité (TLS) et de protocoles IoT (MQTT) grâce à son écosystème.

Open Source & Large Support

Distribué sous licence MIT, supporté par une vaste communauté et de **nombreux microcontrôleurs**, notamment l'ESP32.

Raspberry Pi

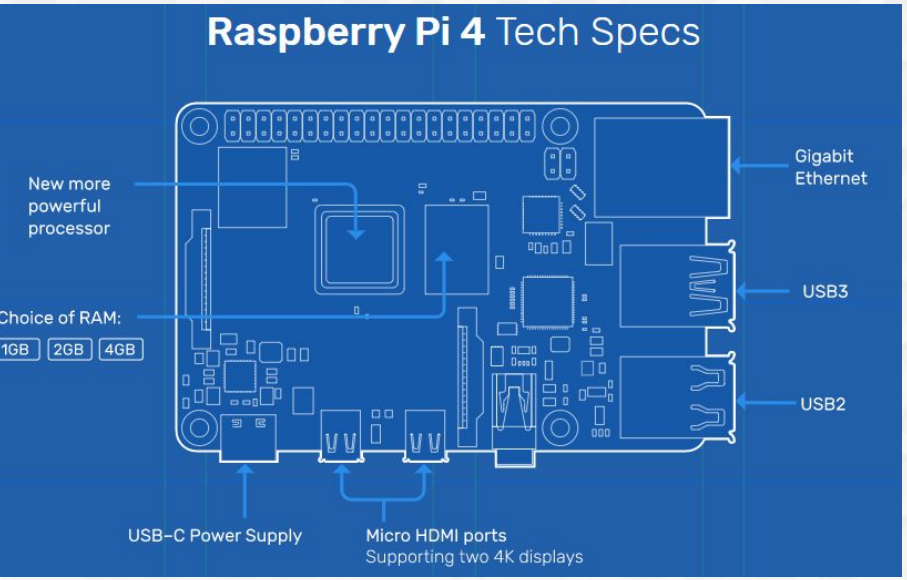
Description : Ordinateur monocarte puissant pour des applications complexes nécessitant un système d'exploitation.

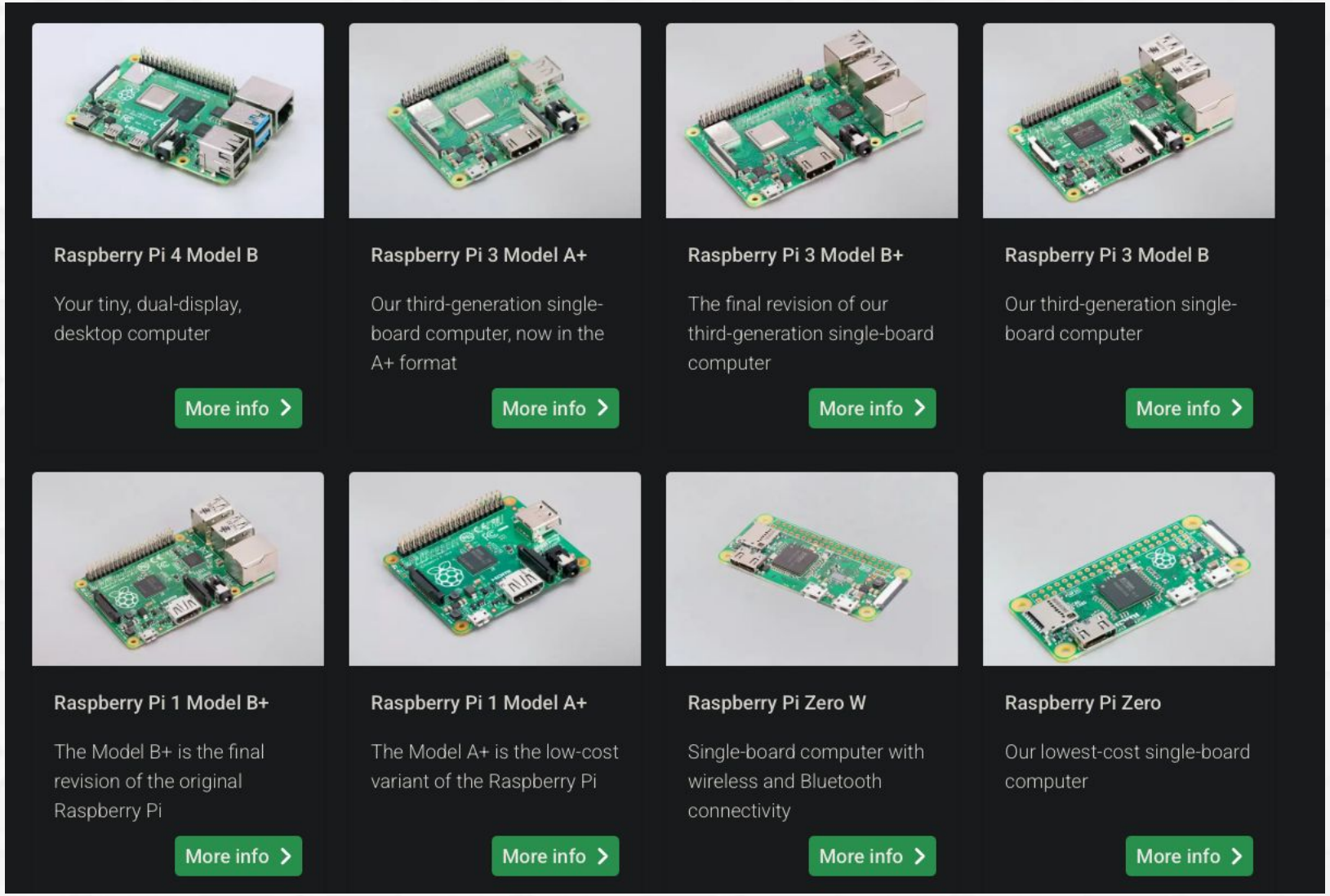
Utilisation : idéal pour les passerelles IoT et les applications nécessitant un traitement de données avancé.
Exemple : un serveur de vidéosurveillance intelligent avec traitement d'image en temps réel.

Modules populaires :

- **Raspberry Pi 4** : Processeur **Quad-Core**, **RAM jusqu'à 8 Go**.
- **Raspberry Pi Pico** : Basé sur un microcontrôleur RP2040 pour les applications embarquées.

Raspberry Pi





RASPBERRY PI 5 : L'ORDINATEUR MONOCARTE (SBC)

Architecture SBC

Une plateforme informatique complète sur une seule carte de circuit imprimé, intégrant CPU, RAM et connectique.

Système d'Exploitation

Exécute des distributions Linux complètes (Raspberry Pi OS) permettant le multitâche et la gestion de fichiers.

Performances Évoluées

Avancée significative en termes de puissance CPU et GPU, idéale pour l'IA, les serveurs web et les interfaces graphiques.

POLYVALENCE D'USAGE

- Domotique avancée
- Vision par ordinateur
- Serveurs Edge
- Développement IA

Raspberry Pi 5 : 16GB RAM avec processeur Broadcom 64-bit

Raspberry Pi

Types de Systèmes	Exemples	Avantages	Inconvenients
Avec OS embarqué	RASPBERRY PI, BEAGLEBONE, etc.	ouverts, puissants, langages de programmation multiples	parfois complexes à mettre en œuvre, prise en main longue, réactivité moyenne, coût relativement élevé, interfaçage plus difficile.
Logiciel propriétaire	ARDUINO, GENUINO, INTEL GALILEO, ESP32, ESP8266 etc.	Très réactifs, très faible coût, fonctionnement plus robuste (pas de couches logicielles), interfaçage aisé, prise en main très rapide.	moins puissants, langages de programmation plus limités, moins flexibles sur le plan logiciel.

Comparaison Arduino / ESP32 / Raspberry Pi

Critère	Arduino	ESP32	Raspberry Pi
Type	Microcontrôleur (MCU)	Microcontrôleur (MCU)	Ordinateur Monocarte (SBC)
OS	Non	Non (RTOS possible)	Linux
Connectivité	Non (module externe)	Wi-Fi / Bluetooth	Wi-Fi / BT 5.0 / Ethernet
Mémoire RAM	32 KB SRAM	512 KB SRAM	4 GB / 8 GB LPDDR4X
Consommation	Très faible	Faible	Élevée
Usage typique	Capteur simple	Objet connecté	Gateway(fog) / Edge
Tension (E/S)	5V(UNO)	3.3V	3.3V (E/S) / 5V (Alim)

Arduino

Idéal pour prototypage rapide, capteurs simples, pas de connectivité native

ESP32

Référence IoT, Wi-Fi + BLE intégrés, deep sleep, faible coût

Raspberry Pi

Puissance Linux, traitement local et intermédiaire (Edge,fog), interface riche

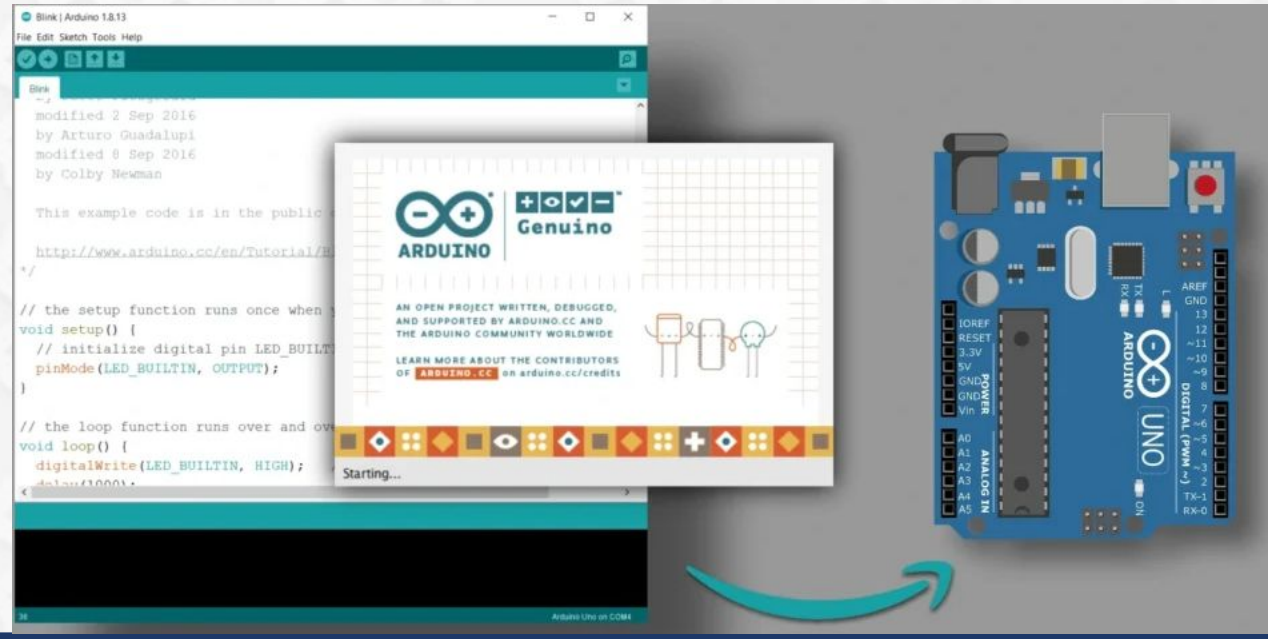
- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- **Installation et configuration des environnements de développement**
- Lecture des données issues des capteurs et transmission des données

Installation et configuration des environnements de développement: Arduino IDE

Description : environnement de développement simple pour coder les cartes Arduino et ESP.

Langage supporté : C/C++.

Exemple : Programmation d'un système de contrôle de température avec capteur DHT11 et écran OLED.



- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- **Installation et configuration des environnements de développement**
- Lecture des données issues des capteurs et transmission des données

Philosophie d'accessibilité et démocratisation de l'électronique.

Open-Source

Logiciel et matériel libres. Les schémas et le code source sont accessibles à tous pour encourager l'innovation et le partage.

Simplicité

L'IDE Arduino abstrait la complexité du bas-niveau, permettant aux débutants de créer sans connaissances approfondies en C++.

Communauté

Des millions d'utilisateurs partagent des bibliothèques et des tutoriels, rendant presque chaque projet réalisable rapidement.

LE CYCLE DE CRÉATION ARDUINO

Idée › Sketch (Code) › Téléversement › Interaction

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- **Installation et configuration des environnements de développement**
- Lecture des données issues des capteurs et transmission des données

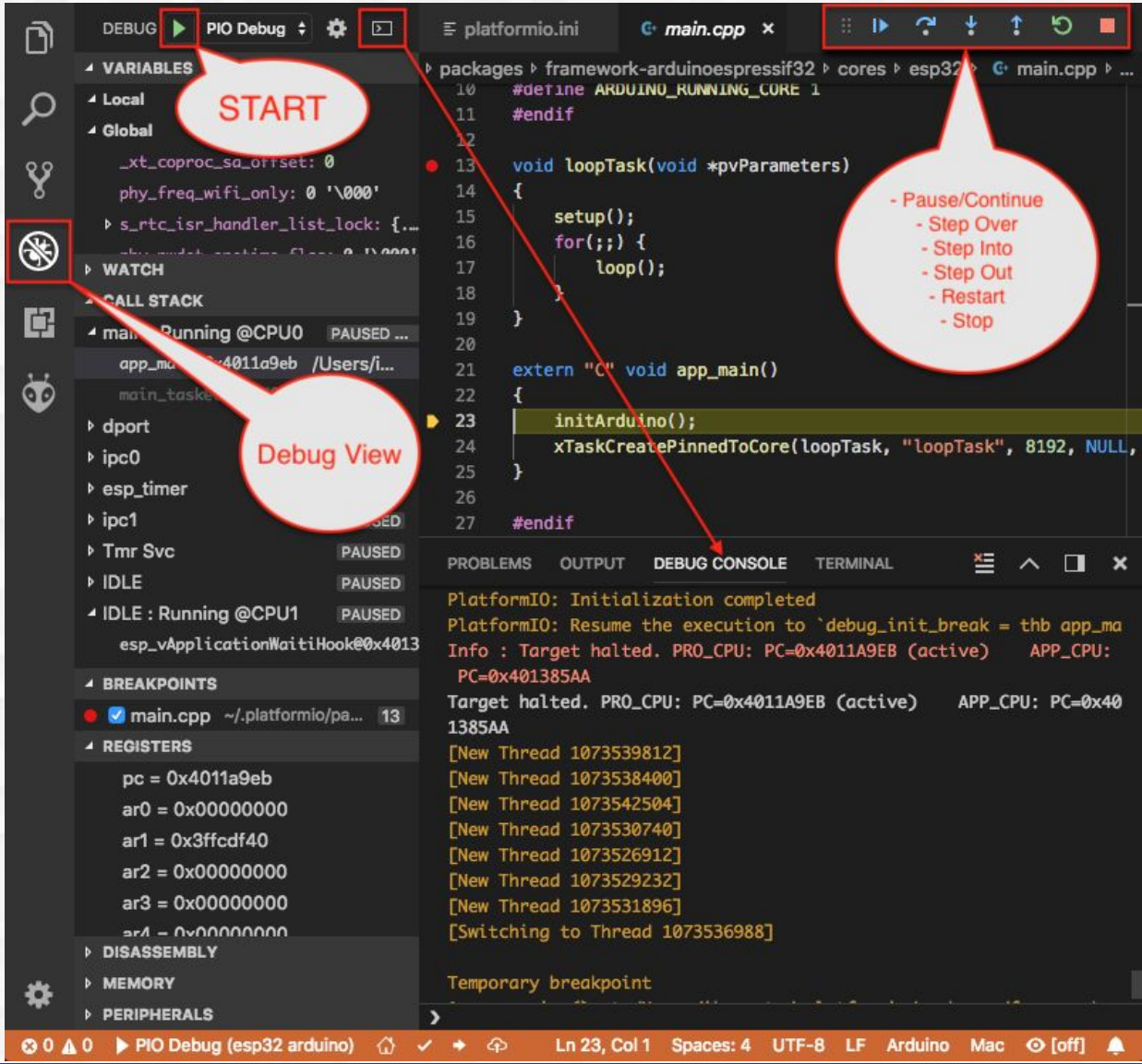
Installation et configuration des environnements de développement: PlatformIO pour VScode

Description : IDE multi-plateforme prenant en charge de nombreux microcontrôleurs (Arduino, ESP32, STM32).

Avantages :

- Compatible avec Visual Studio Code pour un développement avancé.
- Système de gestion de bibliothèques intégré.

Exemple : Développement d'une application de maison intelligente avec ESP32, intégrant plusieurs capteurs.



- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- **Installation et configuration des environnements de développement**
- Lecture des données issues des capteurs et transmission des données

Arduino IDE

SIMPLICITÉ & PROTOTYPAGE

- Interface intuitive pour l'écriture et le téléversement (Sketches).
- Langage basé sur **C/C++** avec bibliothèques simplifiées.
- Vaste écosystème de bibliothèques communautaires.
- support multi-cartes (Arduino, ESP32, ESP8266, STM32 via cores)
- Inclut: Autocomplétion, Debug basique, Serial Plotter amélioré

ESP-IDF

CONTRÔLE AVANCÉ & IoT PROFESSIONNEL

- Framework officiel Espressif **Basé sur FreeRTOS (multitâche temps réel)**.
- Gestion avancée de l'énergie et de la pile réseau.
- Compatible avec **VS Code** et l'IDE Arduino peut fonctionner au-dessus d'ESP-IDF
- Support OTA (Over-The-Air update) et Sécurité avancée (TLS, chiffrement),

RPi OS (Linux)

POLYVALENCE & EDGE / FOG COMPUTING

- Système d'exploitation complet (**Debian** optimisé).
- Multi-langages : Python, C/C++, Node.js
- Serveur local (Web, MQTT, DB)
- Support Docker & services réseau
- Environnement graphique ou headless

Lecture des données capteurs et transmission des données

Interfaces et protocoles de communication pour les capteurs/actionneurs

- **Capteurs analogiques** (utilisant des entrées ADC)
- **Capteurs numériques** (I^2C, SPI, UART, One-Wire)

- Protocoles courants en IoT :
  - **I^2C**(**I**nter-**I**ntegrated **C**ircuit): pour les capteurs nécessitant peu de fils et une communication multi-périphériques.
  - **SPI**(**S**erial **P**eripheral **I**nterface): pour les applications nécessitant des vitesses élevées et une communication rapide.
  - **UART**(**U**niversal **A**synchronous **R**eceiver/**T**ransmitter)**:** pour des échanges de données entre un microcontrôleur et un module externe sans nécessité d'horloge.
  - **One-Wire:** pour des capteurs simples nécessitant une connexion minimale.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**

a. I²C (Inter-Integrated Circuit)

L'I²C est un protocole de communication série synchrone qui utilise seulement deux fils :

- **SDA (Serial Data Line)** pour le transfert des données.
- **SCL (Serial Clock Line)** pour la synchronisation.

Il permet à plusieurs périphériques (capteurs, mémoires, écrans, etc.) d'être connectés au même bus, tout en utilisant une adresse unique pour chaque composant.

Caractéristiques :

- Protocole **série synchrone** utilisant **2 fils** : SDA (données) et SCL (horloge).
- Supporte plusieurs périphériques sur le même bus (adresse unique par périphérique).
- Communication **maître-esclave**.

I2C

Inter-Integrated Circuit

Fils: 2 (SDA – Données, SCL – Horloge)

Type: Série synchrone, Maître/Esclave

Usage: Bus court, capteurs multiples (ex: BME280)

ESP32-S3 : 2 interfaces I2C natives.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**

a. I²C (Inter-Integrated Circuit)

I2C

Inter-Integrated Circuit

Fils: 2 (SDA – Données, SCL – Horloge)

Type: Série synchrone, Maître/Esclave

Usage: Bus court, capteurs multiples (ex: BME280)

ESP32-S3 : 2 interfaces I2C natives.

Exemples en IoT :

- **Capteurs :** BMP280 (pression/température), MPU6050 (accéléromètre/gyroscope).
- **Actionneurs :** Écrans OLED, afficheurs LCD I²C.

Avantages :

- Peu de fils nécessaires.
- Bonne intégration de plusieurs capteurs.

Inconvénients :

- Vitesse limitée (~400 kHz standard).
- Distance de communication limitée.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**

SPI

Serial Peripheral Interface

Fils: 4 (SCLK, MOSI, MISO, CS)

Type: Synchrone Full-duplex

Performance: Très haute vitesse

Utilisé pour : Écrans, Cartes SD, ADCs externes.

b. SPI (Serial Peripheral Interface):

Le SPI est un protocole **synchrone haute vitesse** utilisé pour la communication entre un **microcontrôleur et un ou plusieurs périphériques**. Il fonctionne en mode **maître-esclave** et utilise généralement quatre fils :

- **MISO (Master In Slave Out)** : ligne de données envoyées de l'esclave vers le maître.
- **MOSI (Master Out Slave In)** : ligne de données envoyées du maître vers l'esclave.
- **SCK (Serial Clock)** : ligne d'horloge générée par le maître.
- **SS (Slave Select)** : permet de sélectionner un périphérique spécifique (aussi appelé CS:Chip Select).

Le SPI offre un débit de communication plus rapide que l'I^2C mais nécessite plus de connexions.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**

SPI

Serial Peripheral Interface

Fils: 4 (SCLK, MOSI, MISO, CS)

Type: Synchrone Full-duplex

Performance: Très haute vitesse

Utilisé pour : Écrans, Cartes SD, ADCs externes.

Exemples en IoT :

- **Capteurs (**haute vitesse**):** Modules RFID (**MFRC522**), capteurs de température **MCP3008.**
- **Actionneurs :** Écrans TFT, mémoires flash.

Avantages :

- Très rapide.
- Communication full-duplex.

Inconvénients :

- Nécessite plus de fils qu'I²C.
- Gestion compliquée avec plusieurs périphériques.

c. UART (Universal Asynchronous Receiver-Transmitter): L'UART est un protocole de communication série **asynchrone** couramment utilisé pour la transmission de données entre microcontrôleurs et périphériques. Contrairement à l'I²C et au SPI, il ne partage pas de signal d'horloge, ce qui signifie que les deux dispositifs doivent s'accorder sur une vitesse de transmission appelée **baud rate** (exemple : 9600, 115200 bauds).

L'UART utilise généralement deux fils :

- **TX (Transmission)** : envoi des données, **RX (Reception)** : réception des données.

C'est un protocole très utilisé dans les modules **Bluetooth, GPS, WiFi (ESP8266/ESP32), GSM (SIM800L)**. Il est simple à mettre en œuvre mais ne permet la communication qu'entre deux dispositifs directement connectés.

Caractéristiques :

- Protocole **série asynchrone** utilisant **2 fils** : TX (transmission) et RX (réception).
- Pas besoin d'horloge partagée.
- Utilisé pour la communication **point-à-point**.

Exemples en IoT :

Capteurs : GPS (NEO-6M), module RFID, lecteur de carte NFC.

Actionneurs : Modules GSM/GPRS (SIM800L), imprimantes thermiques.

Avantages : Facile à utiliser, Large compatibilité avec les modules IoT.

Inconvénients :

Supporte uniquement **2 périphériques** en connexion directe.

Nécessite une configuration précise des **baud rates** (vitesse de transmission).

UART

Universal Asynchronous Receiver/Transmitter

Fils: 2 (TX, RX)

Type: Asynchrone Point-à-Point

ESP32-S3: 3 contrôleurs UART

Usage: GPS, Bluetooth, Debug console

Communication série

Utilisé pour la communication entre la carte Arduino et un ordinateur ou d'autres appareils. Toutes les cartes Arduino ont au moins un port série (également connu comme un UART: Universal Asynchronous Receiver Transmitter).

Il communique sur les broches numériques 0 (Rx) et 1 (TX) et avec l'ordinateur via USB. Ainsi, si vous utilisez ces fonctions, vous ne pouvez pas utiliser également broches 0 et 1 comme entrée ou sortie numérique

La fonction begin()

Cette fonction une fois insérée dans la partie *setup* du programme, permet d'initialiser le port série de la carte Arduino. La vitesse de transmission des données est donnée en paramètre à la fonction Serial.begin().

Exemple :

```
void setup() { Serial.begin(9600);
}
void loop() {}
```

Communication série

UART (Universal Asynchronous Receiver Transmitter) est un protocole de communication série asynchrone utilisé pour transmettre des données entre deux appareils électroniques. C'est l'une des méthodes les plus courantes pour connecter des périphériques dans des systèmes embarqués.

- Les données sont envoyées **bit par bit** sur une **seule ligne**.Deux fils principaux sont nécessaires :**TX (Transmitter)** : Pour envoyer les données.
- **RX (Receiver)** : Pour recevoir les données.
- La synchronisation est assurée par des configurations partagées comme la vitesse de transmission (baud rate).
- La vitesse de transmission des données est définie en bits par seconde (bps), par exemple 9600, 115200, etc.

Avantages :

- Simple à configurer et à utiliser.
- Nécessite peu de fils (2 à 4 fils selon l'application).
- Utilisé largement dans les microcontrôleurs, les cartes Arduino, Raspberry Pi, etc.

Applications :

- Communication entre un microcontrôleur et un ordinateur.
- Transmission de données dans des modules Bluetooth ou GPS.
- Connexion entre différents périphériques comme les capteurs et les actionneurs.

Communication série

La fonction println()

Transmet les données au port série sous forme de texte lisible ASCII suivie d'un caractère de retour chariot (ASCII 13, ou '\ r') et un caractère de nouvelle ligne (ASCII 10, ou '\ n').

Exemple :

```
void setup() {
	Serial.begin(9600);
}
void loop() {
	Serial.println(“Hello world”);
	delay(500);
}
```

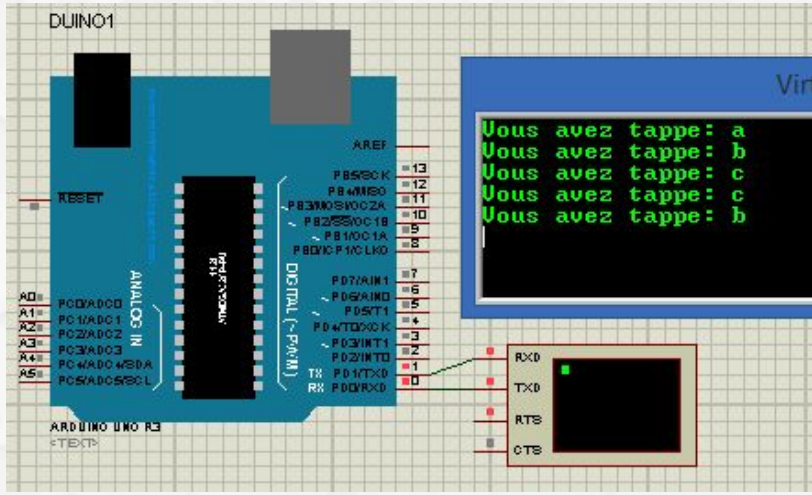
Communication série

La fonction available()

Lorsque la carte Arduino est configurée pour recevoir des données sur le port série, il est nécessaire de pouvoir vérifier à tout instant si une nouvelle trame de données a été reçue est prête à être traiter. La fonction Serial.available() permet d'obtenir le nombre d'octets (caractères) disponibles pour lecture sur port série. Ce sont des données qui sont déjà arrivé et ont été stockés dans la mémoire tampon de réception série (qui détient 64 octets).

La fonction read()

Lit les données série reçues.



d) One-Wire

Le protocole **One-Wire** est conçu pour fonctionner avec un seul fil de données (en plus du fil de masse), permettant ainsi une réduction du câblage. Il est utilisé pour connecter plusieurs périphériques sur une même ligne de données tout en permettant l'identification de chaque périphérique via une adresse unique.

Caractéristiques :

- Protocole **asynchrone** utilisant **1 seul fil** pour données + alimentation.
- Chaque périphérique a une **adresse unique**.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**

Exemples en IoT :

- Capteurs de température comme le **DS18B20**, qui est très populaire en raison de sa simplicité d'utilisation et de sa compatibilité avec les microcontrôleurs comme l'Arduino et l'ESP8266.

Avantages :

- Très simple à câbler.
- Peut fonctionner sur de longues distances.

Inconvénients :

- Vitesse faible.
- Moins utilisé que I^2C ou SPI.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**

Comparaison

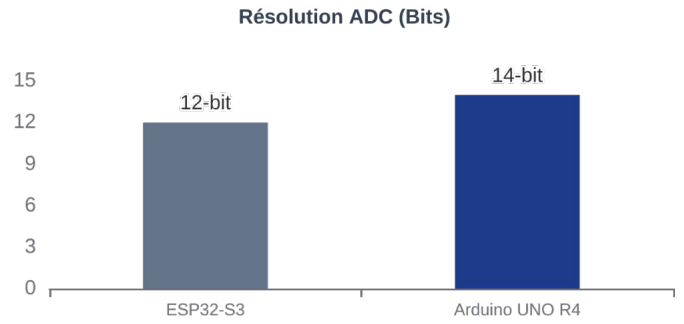
Protocole	Type	Horloge	Nb fils	Plage de vitesses	Bidirectionnel	Multi-capteurs sur le même bus	Nb possible sur Arduino Uno	Nb possible sur ESP32 standard
UART	Asynchrone	Non	2 (+ GND)	~300 bps à ~1 Mbps, parfois plus	Oui	Non, point à point	1 UART matériel	3 UART matériels
I²C	Synchrone	Oui	2 (+ GND)	100 kbps, 400 kbps, 1 Mbps, jusqu'à 3.4 Mbps	Oui	Oui, via adresses	1 bus I²C	1 à plusieurs bus (souvent 1 utilisé, très flexible)
SPI	Synchrone	Oui	4 minimum (+ GND)	~1 Mbps à dizaines de Mbps	Oui, full duplex	Oui, avec 1 CS par périphérique	1 bus SPI	2 bus SPI principaux
1-Wire	Asynchrone (timing)	Non	1 (+ GND)	~16 kbps standard, ~142 kbps overdrive	Oui	Oui, via adresse unique 64 bits	Autant que le bus supporte	Autant que le bus supporte

Lecture des données capteurs

Méthodes d'acquisition et communication synchrone/asynchrone

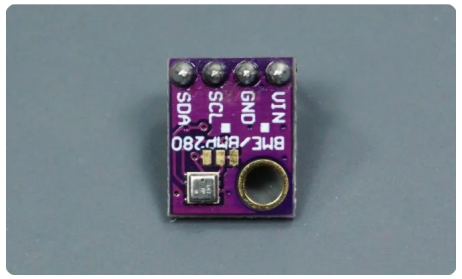
ADC (Analog-to-Digital)

Conversion des tensions analogiques en valeurs numériques.



- **Arduino UNO R4:** Résolution 14 bits
- **ESP32-S3:** Deux ADC SAR 12 bits

EXEMPLE DE CAPTEUR (I2C/SPI)



Module BME280 (Temp, Hum, Pression)

I2C

Inter-Integrated Circuit

Fils: 2 (SDA - Données, SCL - Horloge)

Type: Série synchrone, Maître/Esclave

Usage: Bus court, capteurs multiples (ex: BME280)

ESP32-S3 : 2 interfaces I2C natives.

SPI

Serial Peripheral Interface

Fils: 4 (SCLK, MOSI, MISO, CS)

Type: Synchrone Full-duplex

Performance: Très haute vitesse

Utilisé pour : Écrans, Cartes SD, ADCs externes.

UART

Universal Asynchronous Receiver/Transmitter

Fils: 2 (TX, RX)

Type: Asynchrone Point-à-Point

ESP32-S3: 3 contrôleurs UART

Usage: GPS, Bluetooth, Debug console

"Le choix du protocole dépend de la distance, de la vitesse requise et du nombre de périphériques."

Conversion Analogique/Numérique

Entrées analogiques

Les µcontrôleurs ATMega utilisés pour l'Arduino contiennent six canaux de conversion analogiques-numérique (A/D).

Le convertisseur a une résolution de **10 bits**, c'est-à-dire que chaque **valeur analogique réelle entre 0 et 5V** sera représentée par un nombre entier entre **0 et 1023**.

La fonction principale des broches analogiques pour la plupart des utilisateurs Arduino est de lire les capteurs analogiques.

Les broches analogiques ont aussi toutes les fonctionnalités d'entrée/sortie à usage général (GPIO) ; les mêmes que les broches numériques 0-13.

La fonction analogRead()

Lit la valeur de la broche analogique spécifiée.

Syntaxe : `analogRead(pin)`

pin : A0-A5

Conversion en tension : Convertir les valeurs analogiques en tension réelle avec la formule:

$$\text{Tension (V)} = \frac{\text{Valeur analogique} \times 5.0}{1023}$$

Méthodes de Transmission de Données

Du signal filaire local à l'intégration Cloud IoT

Liaison Filaire

Arduino UNO R4

Téléversement & Communication série (COM virtuel)

ESP32-S3

Programmation, JTAG & Interface USB OTG complète

Raspberry Pi 5

Ports USB 2.0/3.0 : Stockage & périphériques externes

Connectivité Sans Fil

Wi-Fi (802.11 b/g/n)

Intégré sur ESP32 et RPi 5
Liaison réseau LAN/Internet haut débit

Bluetooth 5.0 (LE)

Consommation ultra-faible (BLE)
Idéal pour capteurs de proximité et smartphones

Protocoles Cloud IoT

MQTT

LÉGER & EFFICACE

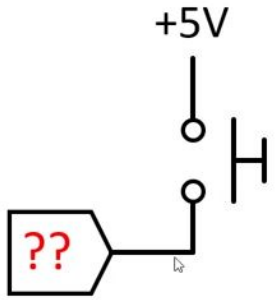
Modèle Publish/Subscribe. Optimisé pour la bande passante limitée et les connexions instables.

HTTP / REST

STANDARD WEB

Modèle Request/Response. Idéal pour l'intégration avec des API Web et services Cloud standard.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**



Propriétés des broches configurées en tant qu'ENTRÉES (INPUT).

Sensibilité au bruit :

Toutefois, les broches configurées avec **pinMode(pin, INPUT**) qui ne sont pas connectées à un circuit, ou qui ont des fils connectés mais non reliés à d'autres circuits, peuvent détecter des changements aléatoires d'état. Ces fluctuations sont dues :

- **Au bruit électrique** provenant de l'environnement.
- **Au couplage capacitif** avec l'état d'une broche voisine.

Ainsi, une broche non connectée ou flottante peut produire des lectures imprévisibles ou erronées.

Cause : Les broches sont en **haute impédance** et sensibles aux fluctuations lorsqu'elles ne sont pas reliées à un circuit.

Solution : Utilisez une **résistance pull-up** ou **pull-down** pour stabiliser l'état de la broche :

- **Pull-up** : Connectez une résistance entre la broche et le +5V.
- **Pull-down** : Connectez une résistance entre la broche et le GND.

Alternativement, activez la résistance **pull-up interne** via le code : **pinMode(pin, INPUT_PULLUP);** Cela évite les fluctuations sans ajouter de composant externe.

- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**

Les résistances de Pull-Up et Pull-Down

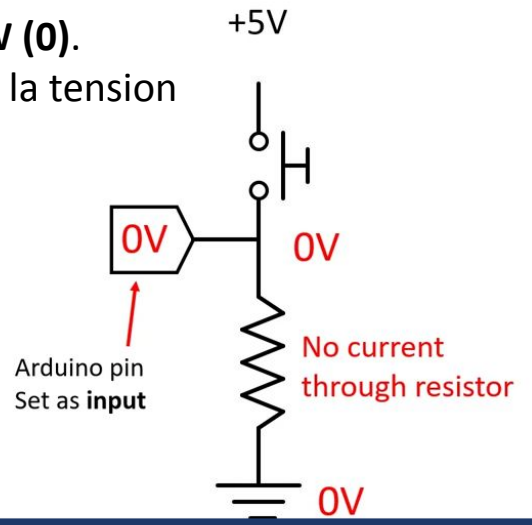
2. Résistance de Pull-Down

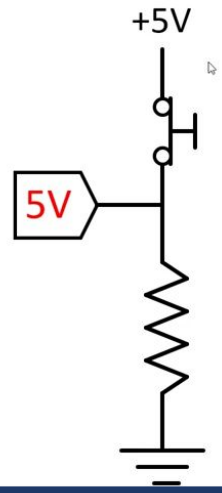
Définition : Une résistance de **pull-down** est une résistance connectée entre une **entrée** et la **masse (GND)**. Elle permet de maintenir l'entrée à un niveau **bas (LOW)** par défaut lorsque le bouton est ouvert.

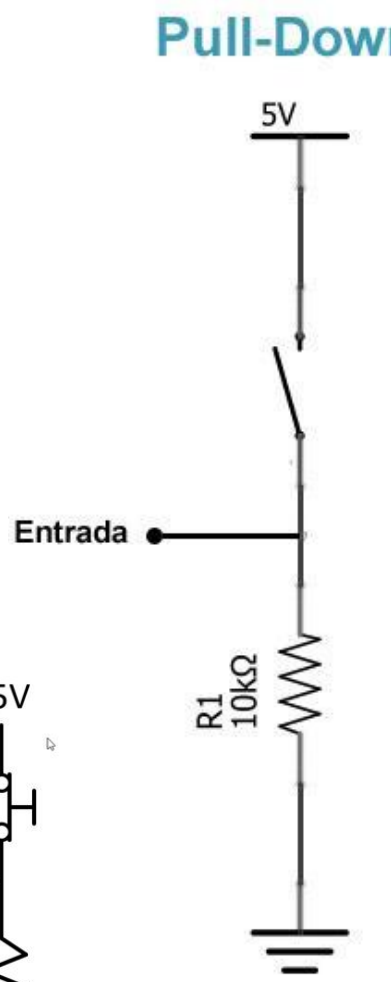
Fonctionnement :

- Lorsque le **bouton est relâché**, l'entrée est reliée à **GND (0V)** et reste à **LOW (0)**.
- Lorsque le **bouton est appuyé**, le courant passe directement par l'entrée et la tension devient **HIGH (1)**.

Utilisation courante : Capteurs logiques, circuits de détection de signaux.





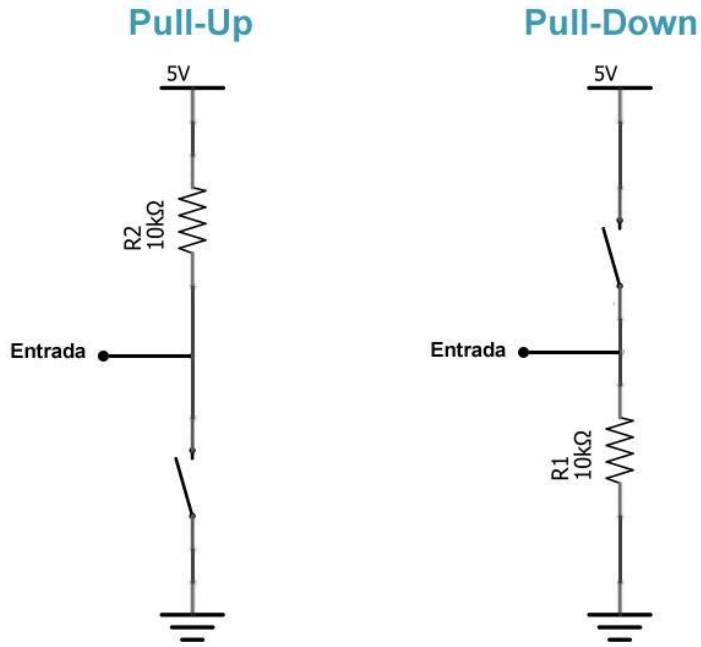


- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**

Les résistances de Pull-Up et Pull-Down

3. Différence entre Pull-Up et Pull-Down

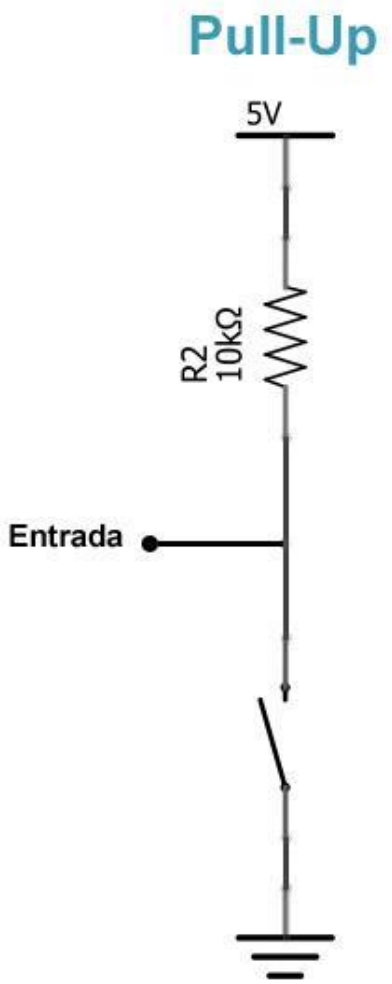
Critère	Pull-Up	Pull-Down
Connexion	Entre entrée et **Vcc (+5V / +3.3V)**	Entre entrée et **GND (0V)**
Valeur par défaut	**HIGH (1)**	**LOW (0)**
État actif	Quand on **appuie**, l'entrée passe à **LOW (0)**	Quand on **appuie**, l'entrée passe à **HIGH (1)**
Utilisation courante	Boutons, I²C, capteurs numériques	Capteurs, détection de signaux



- Typologie des capteurs
- Principes de fonctionnement
- Critères de sélection des capteurs
- Introduction aux dispositifs programmables (Raspberry Pi, ESP32, etc.)
- Installation et configuration des environnements de développement
- **Lecture des données issues des capteurs et transmission des données**

Les résistances de Pull-Up et Pull-Down

Dans un circuit électronique, les entrées des microcontrôleurs et des circuits logiques doivent être correctement définies pour éviter des lectures erronées dues aux fluctuations de tension. C'est là que les **résistances de pull-up et pull-down** interviennent.



1. Résistance de Pull-Up :

Définition : Une résistance de **pull-up** est une résistance connectée entre une **entrée** et une **alimentation positive** (+Vcc, généralement 5V ou 3.3V). Elle permet de maintenir l'entrée à un niveau **haut (HIGH)** par défaut lorsque le bouton ou l'interrupteur est ouvert.

Fonctionnement :

- Lorsque le **bouton est relâché**, le courant passe par la résistance et l'entrée du microcontrôleur est à **HIGH (1)**.
- Lorsque le **bouton est appuyé**, l'entrée est reliée à **GND (0V)** et passe à **LOW (0)**.

Utilisation courante : Boutons poussoirs, capteurs numériques, communication I^2C.

Chaîne typique d'acquisition des données

1

Grandeur physique

La propriété du monde réel à observer ou mesurer.

2

Capteur (transducteur)

Convertit la grandeur physique en un signal électrique.

3

Conditionnement du signal

Adapte le signal analogique pour une meilleure précision.

4

ADC

Convertit le signal analogique en données numériques.

5

Traitement microcontrôleur

Exécute les algorithmes sur les données numérisées.

6

Transmission réseau

Envoie les données vers le cloud ou un serveur.

Un capteur seul ne suffit pas — il faut une chaîne complète.

Types de signaux en IoT

Signal analogique

Tension continue proportionnelle à la grandeur mesurée.
Exemple : capteur de température LM35 → 0 à 1V.

Signal numérique

Valeur binaire 0 ou 1, ou protocole série (I2C, SPI, UART).
Exemple : capteur DHT22 → trame numérique.

Signal impulsionnel

Impulsions comptées pour mesurer un débit ou une vitesse.
Exemple : débitmètre à turbine.

Signal fréquence

La fréquence du signal est proportionnelle à la grandeur.
Exemple : capteur à effet Hall (vitesse de rotation).

Rôle et fonctionnement de l'ADC

- Résolution : nombre de bits (8, 10, 12 bits...) → précision de la conversion
- Plage de tension : Vref (ex: 0–3.3V ou 0–5V)
- Erreur de quantification : ±½ LSB inévitable

$$Résolution = \frac{V_{ref}}{2^n}$$

Exemple : ADC 10 bits

- Vref = 3.3V
- 2^{10} = 1024 niveaux
- Résolution = 3.3 / 1024 ≈ 3.22 mV/bit
- Si la tension lue = 512 → Valeur réelle ≈ 1.65V

Fréquence d'échantillonnage

C'est combien de fois par seconde on mesure un signal.
Si tu mesures la température 1 fois par seconde, Alors la fréquence d'échantillonnage = 1 Hz

Période d'échantillonnage Te : intervalle entre deux mesures.

Fréquence d'échantillonnage fe = 1/Te (nombre de mesures par seconde)

$$fe \geq 2 \times fmax$$

> Pour reconstruire un signal, il faut échantillonner au moins deux fois plus vite que sa fréquence maximale.

Cas	Signal	fe recommandée
IoT température	0.01 Hz	1 mesure / 10 s =0.1 Hz
Audio temps réel	20 kHz	44.1 kHz

Sur-échantillonnage : fe >> 2×fmax → bonne qualité, plus de données

Sous-échantillonnage : fe < 2×fmax → aliasing, signal déformé

Préparation du signal avant traitement

Techniques de conditionnement de signal :

Amplification : augmenter le niveau du signal (ex: ampli-op)

Filtrage : éliminer le bruit (filtre passe-bas RC)

Adaptation d'impédance : éviter les pertes de signal

Pull-up / Pull-down : forcer un état logique stable sur une ligne numérique

Anti-rebond (debounce) : éliminer les faux fronts d'un bouton

Chaîne de conditionnement typique :

1. **Capteur**
   Convertit grandeur physique.
2. **Filtre passe-bas**
   Élimine bruit haute fréquence.
3. **Amplificateur**
   Augmente niveau du signal.
4. **ADC**
   Convertit analogique en numérique.
5. **Microcontrôleur**
   Traite les données.

Bruit et erreurs de mesure

Sources d'erreurs :

- **Bruit électrique** : fluctuations aléatoires du signal (thermique, shot noise)
- **Interférences EMI** : perturbations électromagnétiques externes (moteurs, WiFi)
- **Mauvais câblage** : résistances parasites, boucles de masse
- **Mauvaise calibration** : décalage systématique entre valeur réelle et mesurée
- **Dérive capteur** : variation lente des caractéristiques dans le temps (vieillissement, température)

Comparaison

Signal bruité

Valeurs erratiques, pics parasites

Difficile à exploiter directement

Signal filtré (moyenne glissante)

Signal lissé, tendance claire

Prêt pour traitement ou transmission

Filtre numérique simple : moyenne sur N dernières valeurs

Importance du timestamp

Concepts clés :

millis() : compteur interne depuis le démarrage → pas fiable sur le long terme (dérive)

RTC (Real-Time Clock) : horloge matérielle autonome (ex: DS3231), précise même sans réseau

NTP (Network Time Protocol) : synchronisation via internet, précision ~ms

GPS time : synchronisation ultra-précise via satellite, utilisé en industrie

Cohérence multi-capteurs : si deux capteurs ne partagent pas la même horloge, les données ne peuvent pas être corrélées

Une donnée sans horodatage fiable perd de sa valeur.

Comparaison des sources d'horodatage :

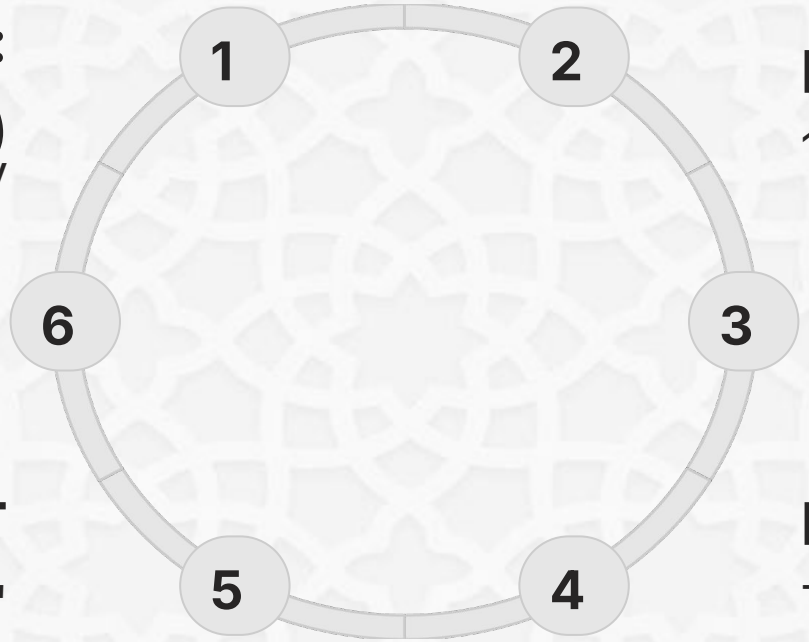
Source	Précision	Réseau requis
millis()	±secondes	Non
RTC	±secondes/jour	Non
NTP	±ms	Oui
GPS	±µs	Non

Cas pratique complet

Capteur humidité sol (ex: capacitif)
Signal analogique 0–3.3V

Cloud / Dashboard
Visualisation temps réel

Publication MQTT
Topic: "ferme/sol/humidite"



ESP32 — ADC 12 bits
1024 niveaux, Vref = 3.3V

Filtrage numérique
Moyenne glissante sur 10 valeurs

Horodatage NTP
Timestamp précis à la ms

Ce pipeline illustre tous les concepts du chapitre : acquisition, conversion, filtrage, horodatage et transmission.

Prochain chapitre : Protocoles de communication IoT

- **TP.2: Les Capteur et les Actionneurs**