Chapitre 7

Informatique embarquée et objets connectés

Objectifs du chapitre

Thème : Informatique embarquée et objets connectés.
Identifier les capteurs, les actionneurs et l'interface homme-machine d'un système embarqué, comprendre les algorithmes qui contrôlent son comportement, et écrire des programmes simples d'acquisition de données et de commande d'un actionneur.
Organisation : programme officiel, cours, méthodes, ateliers Python, exercices corrigés, QCM et fiche de synthèse.

Programme officiel (BO)

Thème « Informatique embarquée et objets connectés » – Contenus et capacités attendues

— **Systèmes informatiques embarqués** : Identifier des algorithmes de contrôle des comportements physiques à travers les données des capteurs, l'IHM et les actions des actionneurs dans des systèmes courants.

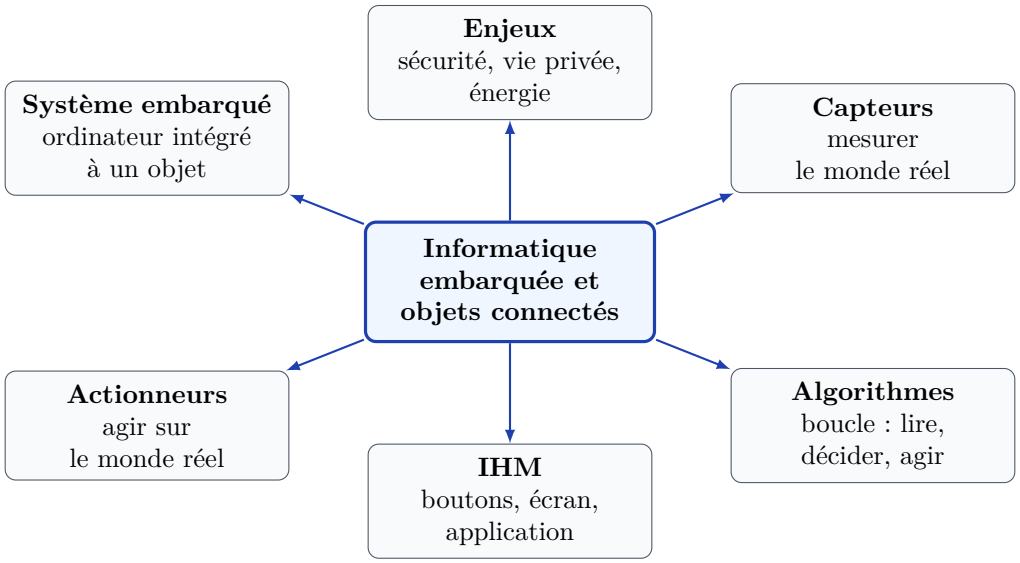
— **Interface homme-machine (IHM)** : Réaliser une IHM simple d'un objet connecté.

— **Commande d'un actionneur, acquisition des données d'un capteur** : Écrire des programmes simples d'acquisition de données ou de commande d'un actionneur.

Source : BO spécial n°1 du 22 janvier 2019, programme de Sciences numériques et technologie de Seconde générale et technologique.

7.1 Carte mentale du chapitre

Carte mentale



7.2 Les systèmes informatiques embarqués

Définition – Système embarqué

Un **système informatique embarqué** est un ordinateur, souvent minuscule (un **microcontrôleur**), intégré à un objet pour en contrôler le fonctionnement : voiture, lave-linge, montre, ascenseur, pacemaker, avion, robot aspirateur… Il fonctionne en autonomie, en temps réel, avec peu d'énergie et de mémoire, et l'utilisateur n'a souvent pas conscience de sa présence. Un **objet connecté** est un système embarqué qui communique par un réseau (Wi-Fi, Bluetooth, 4G/5G, LoRa) avec d'autres objets ou avec Internet.

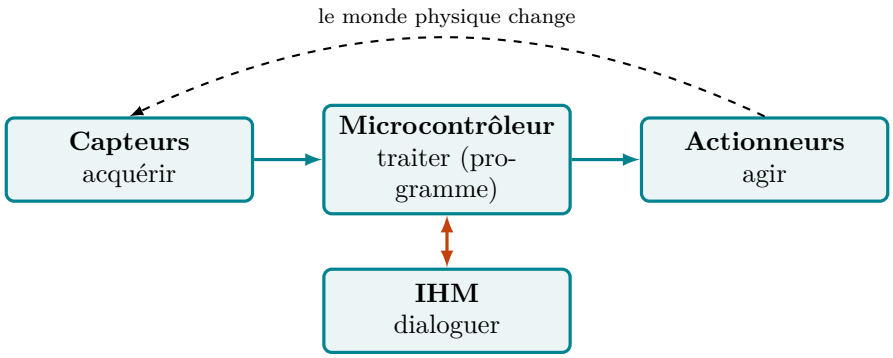
Date	Repère
1967	L'**Apollo Guidance Computer** guide les vaisseaux Apollo : l'un des premiers systèmes embarqués.
1971	Premier microprocesseur (Intel 4004) : un ordinateur tient sur une puce.
1978	L'**ABS** (antiblocage des roues) équipe des voitures de série : un calculateur pilote les freins.
2007	Les **smartphones** réunissent de nombreux capteurs (GPS, accéléromètre, caméra) et un ordinateur puissant.
Aujourd'hui	Une voiture récente contient plus de 50 calculateurs ; on compte environ **15 milliards** d'objets connectés dans le monde.

7.3 Capteurs, actionneurs et chaîne d'information

Définition – Capteur et actionneur

— Un **capteur** mesure une grandeur physique (température, lumière, distance, vitesse, pression, présence) et la transforme en une donnée numérique : c'est l'**acquisition**.
— Un **actionneur** transforme une commande du programme en action physique : moteur, résistance chauffante, lampe (LED), haut-parleur, vanne, électroaimant.

— Le **microcontrôleur** exécute un programme qui lit les capteurs, décide et commande les actionneurs.



Système	Capteurs	Actionneurs	IHM
Thermostat connecté	température, présence	chaudière (marche/arrêt)	écran, molette, application
Éclairage public	luminosité, détecteur de mouvement	lampes	aucune pour le passant ; réglages à distance
Voiture (ABS)	vitesse de chaque roue	freins (valves hydrauliques)	voyant sur le tableau de bord
Vélo à assistance électrique	force sur les pédales, vitesse	moteur électrique	boutons et écran du guidon
Robot aspirateur	distance (obstacles), vide (escaliers), choc	roues, brosses, aspiration	bouton, application, voix

7.4 Les algorithmes de contrôle

Propriété – La boucle de contrôle

Le programme d'un système embarqué tourne en permanence dans une **boucle infinie** :
1. **lire** les capteurs (acquisition) ;
2. **décider** à l'aide de conditions (algorithme de contrôle) ;
3. **commander** les actionneurs ;
4. attendre un court instant, puis recommencer.

```
# Schema du programme d'un thermostat (simulation)
consigne = 19
mesures = [17.5, 18.2, 18.9, 19.4, 20.1, 19.6, 18.8]   # valeurs lues par le capteur

for temperature in mesures:          # sur l'objet : while True
    if temperature < consigne:
        chauffage = "ALLUME"
    else:
        chauffage = "eteint"
    print(f"{temperature} degres -> chauffage {chauffage}")
```

Définition – Régulation avec hystérésis

Avec une seule consigne, le chauffage s'allumerait et s'éteindrait sans arrêt autour de 19 °C, ce qui use la chaudière. On utilise une **hystérésis** : on allume en dessous de $consigne - h$, on éteint au-dessus de $consigne + h$, et entre les deux on **ne change rien**.

Animation interactive « Simulation d'un thermostat » : à manipuler sur https://utspe.com/snt/chapitre-7.

Exemple – Des algorithmes partout

— **ABS** : si une roue ralentit beaucoup plus vite que les autres (elle va se bloquer), le calculateur relâche brièvement son frein, des dizaines de fois par seconde ; le conducteur garde le contrôle de la direction.
— **Vélo à assistance électrique** : le moteur aide proportionnellement à la force exercée sur les pédales, et la loi impose de couper l'assistance au-delà de **25 km/h**.
— **Éclairage automatique** : la lampe s'allume s'il fait sombre **et** si un mouvement est détecté, puis s'éteint après une minute sans mouvement.

7.5 Interface homme-machine et objets connectés

Définition – IHM

L'**interface homme-machine** (IHM) est l'ensemble des moyens qui permettent à un humain de dialoguer avec un système : boutons, écran tactile, voyants, molette, commande vocale, et pour un objet connecté, une **application** sur smartphone ou une page web. Une bonne IHM est simple, donne un retour à l'utilisateur (un voyant s'allume, un message s'affiche) et empêche les erreurs dangereuses (confirmation avant une action importante).

Animation interactive « Commander une lampe connectée depuis son téléphone » : à manipuler sur https://utspe.com/snt/chapitre-7.

Propriété – Enjeux des objets connectés

— **Sécurité** : un objet mal protégé (mot de passe par défaut jamais changé, logiciel pas mis à jour) peut être piraté. En 2016, le réseau de machines piratées **Mirai**, formé de centaines de milliers de caméras et de box, a rendu inaccessibles de grands sites web.
— **Vie privée** : montres, enceintes et caméras collectent des données intimes (santé, voix, présence au domicile).
— **Environnement** : fabrication, consommation électrique permanente, déchets électroniques difficiles à recycler.
— Mais aussi des **bénéfices** : sécurité routière, économies d'énergie, médecine, aide aux personnes dépendantes.

7.6 Ateliers Python

Les ateliers se font sur ordinateur, dans un éditeur Python ou directement sur `https://utspe.com/snt/chapitre-7`. Pour chacun : recopier le programme de départ, suivre les étapes dans l'ordre et exécuter le programme après chaque modification.

Atelier 1 – L'éclairage automatique conditions, and, fonctions

Objectif : Écrire l'algorithme de commande d'une lampe d'extérieur à partir de deux capteurs : luminosité et détecteur de mouvement.
1. Exécuter : la fonction `commande` renvoie toujours `"eteinte"`. Le programme teste quatre situations.
2. Compléter : la lampe doit s'allumer si la luminosité est inférieure à 50 lux **et** qu'un mouvement

est détecté.

3. Ajouter un mode « forcé » : un interrupteur de l'IHM (`force = True`) allume la lampe quoi que disent les capteurs.

Programme de départ :

```
def commande(lux, mouvement):
    # a completer
    return "eteinte"

situations = [(500, True), (20, False), (20, True), (49, True)]
for lux, mouvement in situations:
    print(f"lux={lux}, mouvement={mouvement} -> lampe {commande(lux, mouvement)}")
```

Coup de pouce : `if lux < 50 and mouvement: return "allumee"`. Pour le mode forcé, ajouter un paramètre `force` et tester `if force or (lux < 50 and mouvement):`.

Corrigé rédigé

```
def commande(lux, mouvement, force=False):
    if force or (lux < 50 and mouvement):
        return "allumee"
    return "eteinte"

situations = [(500, True), (20, False), (20, True), (49, True)]
for lux, mouvement in situations:
    print(f"lux={lux}, mouvement={mouvement} -> lampe {commande(lux, mouvement)}")
print("Mode force :", commande(500, False, True))
```

Atelier 2 – Acquisition d'une station météo

listes, min, max, moyenne

Objectif : Traiter les mesures d'un capteur de température : minimum, maximum, moyenne, et alerte de gel.

1. La liste `mesures` contient une température par heure, de 0 h à 23 h. Exécuter le programme.
2. Afficher la température minimale, maximale et la moyenne arrondie à 0,1 °C.
3. Afficher les heures où la température est négative ou nulle (alerte gel), avec la condition `t <= 0`.
4. Défi : un capteur peut renvoyer une valeur aberrante (ici 85). Ignorer les valeurs supérieures à 50 dans les calculs.

Programme de départ :

```
mesures = [1.2, 0.8, 0.1, -0.4, -1.1, -1.5, -0.9, 0.3, 2.4, 4.1, 5.8, 7.2,
           8.5, 9.1, 85, 8.6, 7.4, 5.9, 4.6, 3.8, 3.1, 2.5, 2.0, 1.6]

print(len(mesures), "mesures")
for heure in range(len(mesures)):
    print(f"{heure} h : {mesures[heure]}")
```

Coup de pouce : Construire d'abord une liste `valides` contenant les mesures inférieures ou égales à 50 (`t <= 50`), puis utiliser `min(valides)`, `max(valides)` et `sum(valides) / len(valides)`.

Corrigé rédigé

```
mesures = [1.2, 0.8, 0.1, -0.4, -1.1, -1.5, -0.9, 0.3, 2.4, 4.1, 5.8, 7.2,
           8.5, 9.1, 85, 8.6, 7.4, 5.9, 4.6, 3.8, 3.1, 2.5, 2.0, 1.6]

valides = []
```

```python
for t in mesures:
    if t <= 50:
        valides.append(t)
print(f"{len(mesures) - len(valides)} valeur(s) aberrante(s) ignoree(s)")
print("Minimum :", min(valides))
print("Maximum :", max(valides))
print("Moyenne :", round(sum(valides) / len(valides), 1))
for heure in range(len(mesures)):
    if mesures[heure] <= 0:
        print(f"Alerte gel a {heure} h : {mesures[heure]}")
```

Atelier 3 – Simuler un thermostat avec hystérésis

boucle for, variables d'état

Objectif : Simuler minute par minute la température d'une pièce chauffée par un thermostat, et compter les mises en marche du chauffage.

1. Exécuter : chaque minute, la pièce perd un peu de chaleur, et en gagne 0,3 °C si le chauffage est allumé. Le thermostat compare à la consigne.
2. Compter le nombre de fois où le chauffage s'allume (passage de `False` à `True`).
3. Remplacer la règle par une hystérésis de 0,5 °C : allumer si `t < consigne - 0.5`, éteindre si `t > consigne + 0.5`, sinon ne rien changer.
4. Comparer le nombre de mises en marche avec et sans hystérésis.

Programme de départ :

```python
consigne = 19
t = 16.0
chauffage = False

for minute in range(60):
    if t < consigne:
        chauffage = True
    else:
        chauffage = False
    if chauffage:
        t = t + 0.3
    t = t - 0.02 * (t - 8)      # pertes vers l'exterieur a 8 degres
    if minute % 5 == 0:
        print(f"{minute} min : {t:.1f} degres, chauffage {chauffage}")
```

Coup de pouce : Garder l'ancien état dans `avant = chauffage` avant la décision, puis `if chauffage and not avant: allumages = allumages + 1`. Pour l'hystérésis, utiliser `if ... elif ...` sans `else`.

Corrigé rédigé

```python
def simuler(h):
    consigne = 19
    t = 16.0
    chauffage = False
    allumages = 0
    for minute in range(60):
        avant = chauffage
        if t < consigne - h:
            chauffage = True
        elif t > consigne + h:
            chauffage = False
```

```
        if chauffage and not avant:
            allumages = allumages + 1
        if chauffage:
            t = t + 0.3
        t = t - 0.02 * (t - 8)
    return allumages, round(t, 1)

print("Sans hysteresis (allumages, temperature finale) :", simuler(0))
print("Hysteresis 0,5 degre :", simuler(0.5))
```

Atelier 4 – L'assistance d'un vélo électrique

fonctions, conditions, calculs

Objectif : Programmer la commande du moteur d'un vélo à assistance électrique en respectant la réglementation.

1. La fonction `puissance_moteur(force, vitesse, niveau)` doit renvoyer la puissance du moteur en watts. Exécuter le programme.
2. Règle 1 : pas d'assistance si le cycliste ne pédale pas (force nulle).
3. Règle 2 : la puissance vaut `force` × `coefficient`, avec un coefficient de 1, 2 ou 3 selon le niveau choisi sur l'IHM (éco, normal, sport).
4. Règle 3 : la puissance est limitée à 250 W, et l'assistance est coupée au-delà de 25 km/h (réglementation européenne).

Programme de départ :

```
def puissance_moteur(force, vitesse, niveau):
    # a completer
    return 0

essais = [(0, 15, 2), (60, 12, 1), (60, 12, 3), (120, 20, 3), (80, 26, 3)]
for force, vitesse, niveau in essais:
    p = puissance_moteur(force, vitesse, niveau)
    print(f"force {force} N, {vitesse} km/h, niveau {niveau} -> {p} W")
```

Coup de pouce : Tester d'abord les cas où le moteur est coupé : `if force == 0 or vitesse > 25: return 0`. Ensuite `p = force * niveau`, puis `if p > 250: p = 250` (ou `min(p, 250)`).

Corrigé rédigé

```
def puissance_moteur(force, vitesse, niveau):
    if force == 0 or vitesse > 25:
        return 0
    p = force * niveau
    if p > 250:
        p = 250
    return p

essais = [(0, 15, 2), (60, 12, 1), (60, 12, 3), (120, 20, 3), (80, 26, 3)]
for force, vitesse, niveau in essais:
    p = puissance_moteur(force, vitesse, niveau)
    print(f"force {force} N, {vitesse} km/h, niveau {niveau} -> {p} W")
```

Atelier 5 – L'IHM d'une lampe connectée

input(), while, menus

Objectif : Réaliser une IHM textuelle simple pour piloter deux actionneurs (une lampe et un ventilateur) et lire un capteur.

1. Exécuter : un menu s'affiche et `input()` attend le choix de l'utilisateur. Taper 1, puis 4.
2. Compléter les choix 2 (allumer ou éteindre le ventilateur) et 3 (afficher la température du capteur).
3. Afficher l'état des deux actionneurs après chaque commande : c'est le retour d'information de l'IHM.
4. Ajouter un message d'erreur si le choix n'existe pas.

Programme de départ :

```
lampe = False
ventilateur = False
temperature = 23.5          # valeur lue par le capteur (simulee)

choix = ""
while choix != "4":
    print("1. Lampe  2. Ventilateur  3. Temperature  4. Quitter")
    choix = input("Votre choix : ")
    if choix == "1":
        lampe = not lampe
        print("Lampe allumee" if lampe else "Lampe eteinte")
    # a completer
print("Au revoir")
```

Coup de pouce : `not ventilateur` inverse l'état. Pour l'affichage d'état, écrire après les `if` : `print(f"Etat : lampe={lampe}, ventilateur={ventilateur}")`. L'erreur se traite dans un `else` final (attention au choix 4).

Corrigé rédigé

```
lampe = False
ventilateur = False
temperature = 23.5          # valeur lue par le capteur (simulee)

choix = ""
while choix != "4":
    print("1. Lampe  2. Ventilateur  3. Temperature  4. Quitter")
    choix = input("Votre choix : ")
    if choix == "1":
        lampe = not lampe
    elif choix == "2":
        ventilateur = not ventilateur
    elif choix == "3":
        print(f"Temperature : {temperature} degres")
    elif choix != "4":
        print("Choix inconnu")
    print(f"Etat : lampe={lampe}, ventilateur={ventilateur}")
print("Au revoir")
```

7.7 Exercices corrigés

Exercice

Exercice 1 – Capteur, actionneur ou IHM ? Classer chaque élément d'un robot aspirateur : moteur des roues ; capteur de choc ; bouton « Start » ; capteur de vide (escaliers) ; brosse rotative ; application de programmation ; capteur de niveau de batterie ; haut-parleur qui annonce « Batterie faible ».

Corrigé rédigé

Capteurs : capteur de choc, capteur de vide, capteur de niveau de batterie. Actionneurs : moteur des roues, brosse rotative, haut-parleur. IHM : bouton « Start », application ; le haut-parleur sert aussi d'IHM puisqu'il informe l'utilisateur.

Exercice

Exercice 2 – L'algorithme d'un portail automatique. Un portail s'ouvre quand on appuie sur la télécommande. Il se referme 30 secondes après son ouverture complète, sauf si une cellule détecte un obstacle : la fermeture s'arrête alors et le portail se rouvre.
1. Identifier les capteurs, les actionneurs et l'IHM.
2. Écrire l'algorithme en langage naturel.

Corrigé rédigé

1. Capteurs : récepteur de la télécommande, cellule photoélectrique, fins de course (portail ouvert ou fermé). Actionneur : moteur (et souvent un feu clignotant). IHM : télécommande, feu clignotant.
2. Répéter indéfiniment : si la télécommande est appuyée, ouvrir jusqu'à la fin de course « ouvert » ; attendre 30 s ; fermer ; pendant la fermeture, si la cellule détecte un obstacle, arrêter et rouvrir ; s'arrêter à la fin de course « fermé ».

Exercice

Exercice 3 – Lire un programme de commande. Le programme d'un arrosage automatique contient la condition `if humidite < 30 and (heure <= 8 or heure >= 20):` suivie de la commande de la vanne d'arrosage.
1. Expliquer en français le rôle de chaque partie de la condition.
2. L'arrosage se déclenche-t-il dans les cas suivants : humidité 25 % à 7 h ; 25 % à 14 h ; 60 % à 21 h ; 10 % à 22 h ?

Corrigé rédigé

1. L'arrosage ne se déclenche que si le sol est sec (humidité inférieure à 30 %) **et** qu'il est tôt le matin ou le soir (pour limiter l'évaporation).
2. 7 h, 25 % : arrosage ; 14 h : pas d'arrosage (heure chaude) ; 21 h, 60 % : pas d'arrosage (sol humide) ; 22 h, 10 % : arrosage.

```
def arrosage(humidite, heure):
    if humidite < 30 and (heure <= 8 or heure >= 20):
        return "arrosage"
    return "pas d'arrosage"

for h, t in [(25, 7), (25, 14), (60, 21), (10, 22)]:
    print(f"{h} % a {t} h : {arrosage(h, t)}")
```

Exercice

Exercice 4 – Moyenne glissante. Un capteur de distance d'un robot renvoie des mesures bruitées : `[52, 49, 51, 80, 50, 48, 47, 45, 44, 42]` (en cm). Pour lisser, le robot utilise la moyenne des 3 dernières mesures.
Écrire un programme qui affiche la moyenne glissante à partir de la 3e mesure. Le robot doit freiner dès que la distance lissée passe sous 46 cm : à quelle mesure freine-t-il ?

Corrigé rédigé

Moyennes : 50,7 ; 60,0 ; 60,3 ; 59,3 ; 48,3 ; 46,7 ; 45,3. Le robot freine à la **9e mesure** (moyenne 45,3 cm). La valeur aberrante 80 a été atténuée par la moyenne mais retarde un peu la détection.

```
mesures = [52, 49, 51, 80, 50, 48, 47, 45, 44, 42]
for i in range(2, len(mesures)):
    moyenne = (mesures[i - 2] + mesures[i - 1] + mesures[i]) / 3
    print(f"mesure {i + 1} : {moyenne:.1f} cm")
    if moyenne < 46:
        print("Freinage !")
        break
```

Exercice

Exercice 5 – Sécurité des objets connectés. Une famille installe une caméra connectée pour surveiller son salon. Elle garde le mot de passe d'origine `admin` et ne fait jamais les mises à jour.
1. Quels risques cela fait-il courir ?
2. Proposer quatre bonnes pratiques.

Corrigé rédigé

1. N'importe qui connaissant le mot de passe par défaut peut regarder les images du salon (vie privée, cambriolage) ; la caméra peut être intégrée à un réseau de machines piratées comme Mirai pour attaquer d'autres sites.
2. Changer le mot de passe par un mot de passe long et unique ; faire les mises à jour ; désactiver l'accès à distance s'il est inutile ; placer les objets sur un réseau Wi-Fi séparé ; couvrir ou couper la caméra quand on est à la maison ; choisir un fabricant qui publie des mises à jour de sécurité.

Exercice

Exercice 6 – Consommation en veille. Un foyer possède 12 objets connectés qui consomment chacun 2 W en permanence, même quand on ne les utilise pas.
1. Calculer l'énergie consommée en un an (en kWh).
2. Avec un prix de 0,25 € par kWh, quel est le coût annuel ?
3. Pour 30 millions de foyers, comparer à la production annuelle d'un réacteur nucléaire (environ 7 TWh).

Corrigé rédigé

1. $12 \times 2 = 24$ W ; $24 \times 24 \times 365 = 210\,240$ Wh $\approx$ **210 kWh**.
2. $210 \times 0{,}25 \approx$ **53 €** par an.
3. $30\,000\,000 \times 210$ kWh $\approx 6{,}3 \times 10^9$ kWh = **6,3 TWh** : presque la production annuelle d'un réacteur nucléaire.

```
kwh = 12 * 2 * 24 * 365 / 1000
```

```
print(kwh, "kWh par an")
print(round(kwh * 0.25), "euros par an")
print(kwh * 30000000 / 1e9, "TWh pour 30 millions de foyers")
```

7.8 QCM – Teste tes connaissances

Pour chaque question, une seule réponse est correcte.

Q1. Un système embarqué est…

a) un ordinateur portable
b) un ordinateur intégré à un objet pour le contrôler
c) un logiciel de jeux
d) un serveur web

Q2. Lequel de ces éléments est un capteur ?

a) un moteur
b) une LED
c) un thermomètre électronique
d) un haut-parleur

Q3. Lequel de ces éléments est un actionneur ?

a) un détecteur de mouvement
b) une résistance chauffante
c) un capteur de luminosité
d) un microphone

Q4. L'IHM d'un système, c'est…

a) son programme
b) ce qui permet à l'humain de dialoguer avec lui
c) sa batterie
d) son réseau Wi-Fi

Q5. La boucle de contrôle d'un système embarqué consiste à…

a) lire les capteurs, décider, commander les actionneurs, recommencer
b) allumer puis éteindre l'objet
c) télécharger des mises à jour
d) envoyer les données sur Internet

Q6. Que produit ce programme ?

```
t = 18.5
consigne = 19
if t < consigne:
    print("chauffage allume")
else:
    print("chauffage eteint")
```

a) chauffage allume
b) chauffage eteint
c) rien
d) une erreur

Q7. Que produit ce programme ?

```
lux = 50
mouvement = True
print(lux < 50 and mouvement)
```

a) True
b) False
c) 50
d) une erreur

Q8. Une hystérésis dans un thermostat sert à…

a) chauffer plus vite
b) éviter que le chauffage s'allume et s'éteigne sans arrêt
c) mesurer la température
d) économiser la batterie du téléphone

Q9. Le système ABS d'une voiture…

a) relâche brièvement le frein d'une roue qui se bloque
b) accélère automatiquement
c) règle la radio
d) allume les phares

Q10. L'assistance d'un vélo électrique doit être coupée au-delà de…

a) 15 km/h
b) 25 km/h
c) 45 km/h
d) 50 km/h

Q11. Pour sécuriser un objet connecté, il faut d'abord…

a) le laisser toujours allumé
b) changer le mot de passe par défaut
c) le connecter au Wi-Fi public
d) désactiver les mises à jour

Q12. Ordre de grandeur du nombre d'objets connectés dans le monde :

a) 15 millions
b) 150 millions
c) 15 milliards
d) 15 000 milliards

Corrigé rédigé

Réponses du QCM.
Q1. b) Microcontrôleur intégré à une voiture, un lave-linge, une montre…
Q2. c) Un capteur mesure une grandeur physique ; les trois autres sont des actionneurs.
Q3. b) Un actionneur agit sur le monde physique : ici, il chauffe.
Q4. b) Boutons, écran, voyants, application, commande vocale…
Q5. a) C'est la boucle acquérir, traiter, agir.
Q6. a) 18,5 est inférieur à 19.
Q7. b) lux < 50 est faux (50 n'est pas strictement inférieur à 50), donc la conjonction est fausse.
Q8. b) On allume sous consigne $-$ h, on éteint au-dessus de consigne $+$ h.
Q9. a) Il garde la roue en rotation pour que le conducteur puisse diriger la voiture.
Q10. b) C'est la réglementation européenne des vélos à assistance électrique.
Q11. b) Les mots de passe par défaut sont connus de tous les pirates.

Q12. c) Environ 15 milliards, soit à peu près deux par habitant.

7.9 Fiche de synthèse

Fiche de synthèse

- **Système embarqué :** Ordinateur (microcontrôleur) intégré à un objet, autonome et en temps réel. Un objet connecté communique en plus par un réseau.
- **Capteurs et actionneurs :** Les capteurs mesurent (acquisition), le programme décide, les actionneurs agissent (moteur, LED, résistance).
- **Algorithme de contrôle :** Boucle infinie : lire, décider avec des conditions, commander, recommencer. L'hystérésis évite les marches/arrêts incessants.
- **Exemples :** Thermostat, ABS, vélo électrique (coupure à 25 km/h), éclairage automatique, robot aspirateur.
- **IHM :** Boutons, écrans, voyants, application, voix : simple, avec retour d'information et protection contre les erreurs.
- **Enjeux :** Sécurité (mot de passe par défaut, mises à jour, Mirai), vie privée, consommation et déchets, mais aussi sécurité routière et santé.