MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE L'ENSEIGNEMENT
SUPÉRIEUR
ET DE LA RECHERCHE

Liberté
Égalité
Fraternité

Concours externe BAC + 3 du CAPET

Cafep-Capet

Section : Sciences industrielles de l'ingénieur
Option : ingénierie informatique

1) Exemple de sujet pour la première épreuve d'admissibilité
2) Attendus de l'épreuve
3) Extrait de l'arrêté du 17 avril 2025

Les épreuves du concours externe du Capet et Cafep-Capet BAC+ 3 sont déterminées dans l'arrêté du 17 avril 2025 fixant les modalités d'organisation du concours externe du certificat d'aptitude au professorat de l'enseignement technique, publié au Journal Officiel du 19 avril 2025, qui fixe les modalités d'organisation du concours et décrit le schéma des épreuves.

1) Exemple de sujet pour la première épreuve d'admissibilité

CAPET BAC + 3
Sujet 0

Sommaire

Les parties A, B et C sont à traiter obligatoirement par tous les candidats.
Les candidats devront choisir de traiter la partie D (choix 1) ou la partie E (choix 2).

Présentation du support

Escape Game

Un Escape Game est une expérience ludique où un groupe de joueurs (2 à 6) doit résoudre une suite d'énigmes pour s'échapper d'une salle dans un temps limité. Les premières versions reposaient sur des mécanismes simples (cadenas, clés, intervention manuelle d'un opérateur). La société Kairos Escape Game a enrichi ce concept grâce à un système connecté basé sur des modules d'acquisition originaux, centralisés par une supervision pilotée à distance par le Game Master.

Figure 1 : exemple de décor d'une salle à l'ambiance futuriste

Partie A (à traiter obligatoirement)

Implémentation d'une nouvelle énigme dans la salle Athena Station

Objectif : il s'agit d'implémenter une nouvelle énigme dans la salle Athena Station en s'inspirant d'une énigme existante dans une autre salle.
La salle Athena Station propose une succession d'épreuves à résoudre en une heure. La direction artistique souhaite y intégrer une nouvelle énigme où les joueurs doivent retrouver les badges de l'équipage et les placer sur les modules correspondants du poste de commandement. Cette partie consiste donc à analyser une énigme similaire utilisée dans une autre salle, afin d'adapter son fonctionnement et de l'intégrer au code de la carte de commande de la salle Athena Station.

Un exemple de système de déclenchement de portes : runes magiques et tags RFID.
Dans un souci de cohérence, les technologies modernes doivent être dissimulées dans des salles à thématique historique ou magique, comme la salle Pirates des Antilles. Les systèmes de validation d'énigmes sont donc « habillés » pour rester cohérents. On s'intéresse ici à la reconstitution de pierres runiques basées sur l'utilisation de tags RFID contenus dans des pierres en plâtre sur lesquelles sont tracés des symboles (Figure A1, photo de gauche). La lecture de ces « tags » est effectuée par des « tags readers » dissimulés dans des socles visibles aux symboles correspondants (Figure A1 photo de droite : les 3 supérieurs sont noirs et celui du bas est clair).

Figure A1 : détail des pierres et des socles

Question A1 | **Expliciter** brièvement le fonctionnement d'un « tag » RFID en donnant un exemple usuel d'utilisation de cette technologie.

L'énigme des pierres runiques est la suivante : quatre pierres sur lesquelles un symbole magique différent a été dessiné sont cachées dans une salle. Quatre socles représentant chacun le symbole d'une des pierres sont également disposés dans la salle. Les joueurs doivent retrouver les quatre pierres et les déposer sur le bon socle pour déclencher l'ouverture d'une porte.

Question A2 | **Préciser** la nature du système contenu dans chacun des socles. On pourra notamment expliciter celui-ci grâce à un schéma.

Question A3 | **Indiquer** s'il est possible d'implémenter cette énigme en plaçant un aimant dans chacune des pierres et en cachant une sonde à effet Hall dans chacun des socles ? **Justifier** votre réponse.

La partie logicielle repose sur l'implémentation d'une fonction `pierres_placees(id_sp_1,id_sp_2, id_sp_3, id_sp_4)` qui prend en argument 4 entiers indiquant pour chaque socle, l'identifiant de la pierre placée dessus : si la variable `id_sp_1` prend la valeur 2, cela signifie que la pierre numéro 2 est placée sur le socle numéro 1. Cette fonction renvoie le booléen `True` si toutes les pierres (d'identifiants respectifs 1,2,3 et 4) sont placées sur le bon socle.

Question A4 | **Proposer** une implémentation en langage Python de la fonction `pierres_placees(id_sp_1,id_sp_2, id_sp_3, id_sp_4)`.

Certains joueurs ne remarquent pas au cours de la partie que chaque pierre est munie d'un symbole différent et ceux-ci ne comprennent pas pourquoi rien ne se passe lorsque les 4 pierres sont placées sur les socles. Par ailleurs, le socle du bas qui est clair n'est pas toujours repéré par les joueurs. Il a été décidé d'utiliser des leds de couleur rouge pour éclairer les socles lorsque les 4 pierres ont été placées mais sur les mauvais socles. Les leds s'allument également lorsque les 3 premières pierres ont bien été placées, mais pas la 4ième.
Une fonction `pierres_incompletes(id_sp_1, id_sp_2, id_sp_3, id_sp_4)` renvoie le booléen `True` si toutes les pierres sont bien placées sur un socle mais à des emplacements incorrects ou si les pierres 1,2 et 3 sont bien placées mais pas la 4ième. Les variables `id_sp_i` pour i = 1,2,3 ou 4 sont définies de la même manière que dans la question précédente et on supposera par ailleurs que celles-ci peuvent prendre la valeur 0 si aucune pierre n'est placée sur le socle i.

Question A5 | **Proposer** une implémentation en langage Python de la fonction `pierres_incompletes(id_sp_1,id_sp_2,id_sp_3, id_sp_4)`

Adaptation à la salle Athena Station :
La nouvelle énigme implantée dans la salle Athena Station repose sur 6 badges munis de tags rfid qui doivent être déposés sur leurs socles correspondants. Contrairement à l'énigme précédente, les joueurs ne disposent pas d'indication visuelle pour faire correspondre les badges et leurs socles. En revanche lorsque tous les badges sont placés sur un socle, un écran affiche le nombre de tags correctement placés lors de l'appui sur un bouton. L'objectif des joueurs est donc de trouver le plus rapidement possible l'emplacement correct de chacun des badges. La carte de commande collecte en commençant par 0 les identifiants (0 ,1,2,3,4,5) des tags rfid des badges placés sur les socles dans une liste Python `L[i]` dont l'élément d'indice `i` contient le tag rfid du badge placé sur le socle numéro i. On considère que le badge est bien placé si son identifiant numéro `i` est placé à l'indice `i` de la liste.

Question A6 | **Écrire** en langage python une fonction `nombre_places(L)` qui reçoit en entrée une liste d'identifiants suivant la définition établie ci-dessus et qui renvoie le nombre d'éléments bien placés.

On se place du point de vue des joueurs et on souhaite construire un algorithme permettant d'identifier rapidement la bonne combinaison de tags rfid. On note n le nombre de tags rfid à placer sur les socles.

Question A7 | **Exprimer** le nombre de permutations possibles des tags rfid en fonction de n. Pour n = 6 et en supposant que les joueurs prennent en moyenne 10 secondes pour évaluer une permutation, calculer la durée maximale mise par les joueurs pour résoudre l'énigme. Conclure sur la pertinence de cette énigme.

En vue de simplifier l'énigme, les concepteurs ont choisi d'éclairer en vert les badges correctement placés sur un socle et en rouge ceux qui sont mal placés. L'éclairage ne se fait qu'une fois tous les badges placés sur un socle. On dispose pour cela du programme fourni à la figure A2.

Question A8 | **Analyser** ce programme et **préciser** les états successifs de la liste L affichés par la fonction lors d'un appel à `cherche_permutation([4,0,1,3,2])`.

Question A9 | En suivant la stratégie précédente et en se plaçant dans le pire des cas avec un ensemble de 6 tags à placer correctement, **calculer** le nombre de permutations que les joueurs devront tester avant de trouver la bonne ? En **déduire** le temps nécessaire pour résoudre l'énigme en supposant que les joueurs prennent 10 secondes pour tester chaque permutation et **conclure** sur ce choix d'indiquer aux joueurs les tags correctement placés.

```
def liste_places(L):
  places = []
  for i in range(len(L)):
     if L[i]==i:
        places.append(True)
     else:
        places.append(False)
  return places
def cherche_permutation (L) :
‘’’ La fonction reproduit une stratégie de recherche de la permutation des tags rfid par les joueurs. ‘’’
  while nombre_places(L) != len(L) :
    print(L)
    places = liste_places(L)
    dernier = None
    premier = None
     for i in range(len(L)):
       if places[i] == False:
         if dernier == None:
           premier = i
           dernier = L[i]
         else:
           dernier,L[i] = L[i],dernier # permutation du contenu de la variable dernier et du i-ème élément de la liste L :  L[i]
    L[premier] = dernier
    return L
```

Figure A2 : algorithme de recherche de la permutation encodée dans la liste L

Partie B (à traiter obligatoirement)

Intégration matérielle d'une nouvelle énigme dans la salle Athena Station

Objectif : Afin de pouvoir mettre en œuvre une nouvelle énigme dans la salle Athena Station, il est nécessaire de mettre en place de nouveaux composants et notamment un dispositif de lecture des tags RFID.

La lecture d'un tag RFID s'effectue au moyen du circuit dédié RFID-RC522 (tag reader). Il a été décidé de ne pas relier directement ce circuit à la carte de commande Raspberry Pi orchestrant les programmes de la salle de jeu mais d'effectuer la lecture des identifiants RFID via une carte Arduino qui reçoit les informations des identifiants via un bus SPI. Ces informations seront dans un second temps transmises par la carte Arduino à la carte Raspberry Pi. On rappelle que le bus SPI est un bus de type maître-esclave où la carte Arduino jouera le rôle de maître.

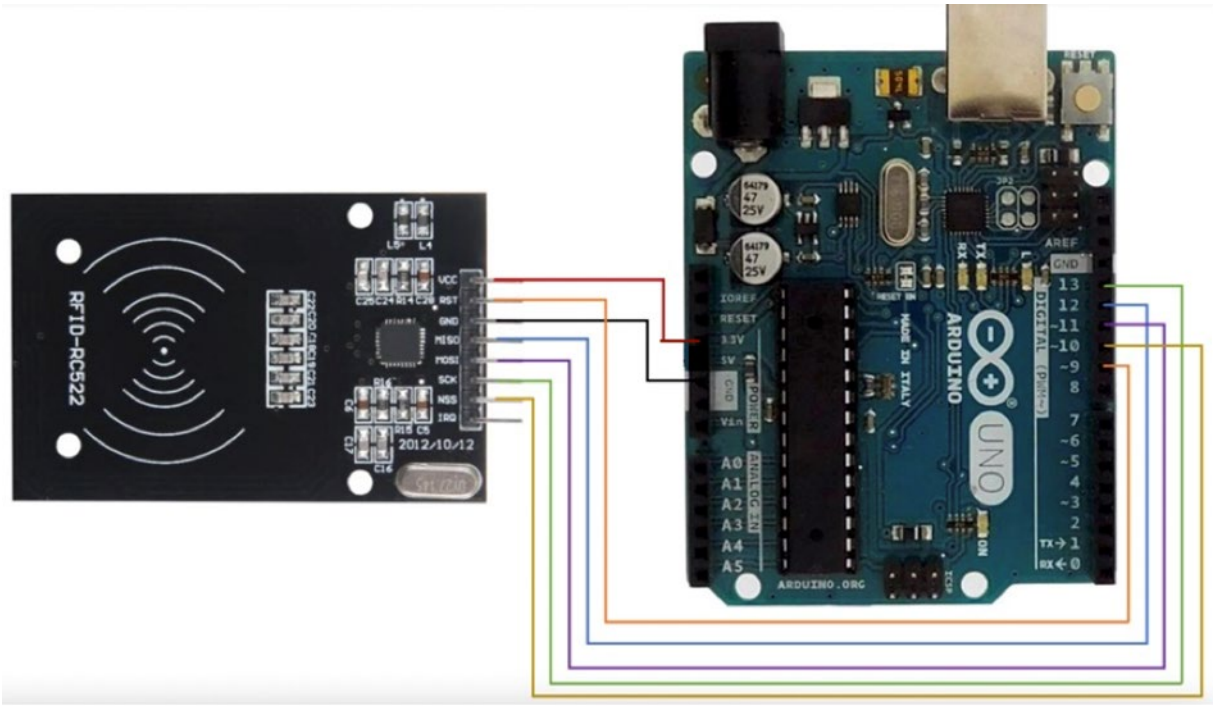


Figure B1 : Schéma de câblage du Tag Reader et de la carte Arduino

- L'alimentation de la carte RC522 s'effectue via les bornes VCC et GND.
- Les bornes MISO (Master Input Slave Output) et MOSI (Master Output Slave Input) permettent d'effectuer les transferts des données.
- La borne SCK reçoit le signal d'horloge transmise par le maître.
- La borne NSS permet de choisir l'esclave vers lequel la communication s'effectue : le maître met la tension au niveau bas sur la borne de l'esclave avec lequel il communique.

On s'intéresse dans un premier temps à la communication avec un seul esclave.

Question B1 | En s'appuyant sur la description précédente, **indiquer** quelle(s) borne(s) du tag reader il faut relier à l'oscilloscope pour visualiser les trames contenant les identifiants du tag RFID posé sur le tag reader ?

Chaque identifiant de tag RFID est codé sur 4 octets. Le débit maximal de transfert sur le bus SPI est de 10 Mbits·s^{-1}.

Question B2 | **Calculer** la durée minimale de récupération de l'identifiant d'un tag RFID ?

Il est courant que des données transmises entre deux cartes électroniques soient accompagnées d'un bit de parité.

Question B3 | **Expliquer** ce qu'est un bit de parité et **préciser** l'intérêt d'ajouter ce bit à une transmission de données.

La structure en bus permet de réduire le nombre de bornes que la carte électronique maître doit occuper pour la communication.

Question B4 | **Expliquer** pourquoi il sera nécessaire que la borne NSS de chaque esclave soit connectée à une borne différente du maître.

On s'intéresse maintenant à la lecture des identifiants de plusieurs modules RC522 connectés sur le même bus SPI.

Question B5 | **Compléter** sur le document réponse DRB1 le schéma de câblage de 4 modules RFID-RC522 sur le même bus SPI dont la carte Arduino joue le rôle de maître.

Le programme implémenté sur la carte Arduino est représenté par l'algorigramme fourni figure B2 . On rappelle que dans cet algorigramme a : = b représente l'opération d'affectation de la valeur b dans la variable a et que a == b est un test d'égalité des variable a et b.

Question B6 | **Expliquer** la fonction de la variable flag et **préciser** dans quel cas celle-ci vaut `True` ou `False`.

Question B7 | **Expliquer** quel est l'avantage d'utiliser une carte Arduino dédiée à la lecture des tags reader plutôt qu'effectuer une connexion directe des tags reader à la carte Raspberry Pi.

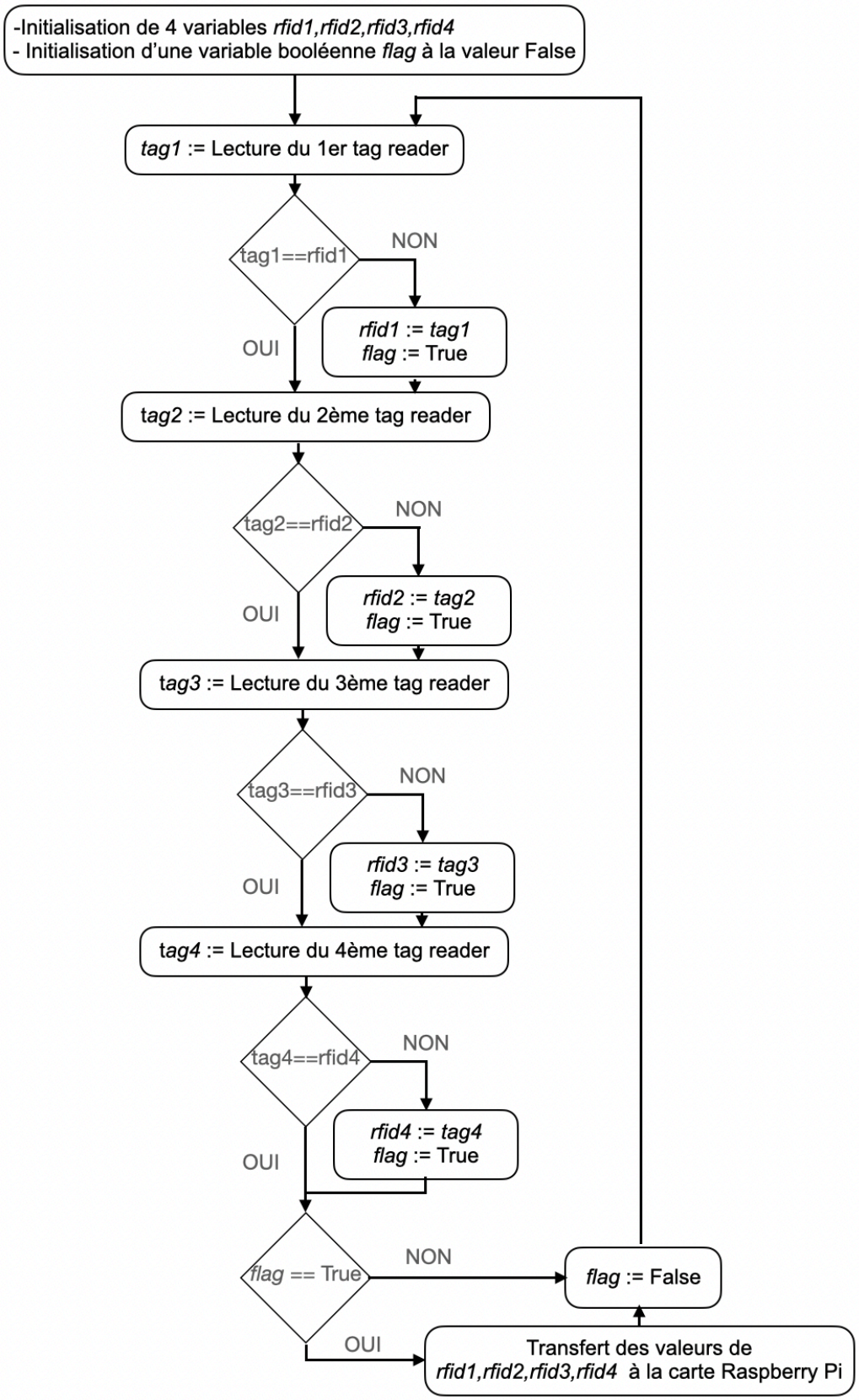


Figure B2 : algorigramme représentant l'exécution du code implémenté sur la carte Arduino

Dans la suite on s'intéresse à la liaison série (UART) établie entre la carte Arduino et la carte Raspberry Pi. La liaison est paramétrée à 9600 bauds sans bit de parité.

Question B8 | **Préciser** ce que représente la quantité 9600 bauds. Donner un exemple de cas où le débit de la transmission en baud est différent du débit de transmission en bits·s^{-1}.

Question B9 | **Expliquer** la différence entre une liaison synchrone et asynchrone. **Préciser** si le bus SPI étudié précédemment et/ou la liaison série sont synchrones ou asynchrones ?

Question B10 | En reprenant les résultats établis précédemment, **calculer** un ordre de grandeur du temps nécessaire pour que la carte Raspberry Pi reçoivent l'information des identifiants des tags RFID posés sur les tags readers. **Conclure** sur la possibilité d'utiliser le dispositif étudié dans le cadre d'une énigme où l'on souhaite que le système prenne en compte les actions des joueurs en moins de 200 ms.

Partie C (à traiter obligatoirement)

Évolution de l'architecture réseau

Objectif : On souhaite faire évoluer l'architecture réseau du site afin de pouvoir accueillir de nouvelles salles éventuellement déportées dans un autre bâtiment.

Actuellement, toutes les salles sont sur un même réseau dont l'adresse est la suivante : 192.168.17.0/24.Dans l'optique d'accueillir de nouvelles salles qui pourront être sur le même site ou sur un site déporté, il est proposé de faire évoluer le réseau suivant le schéma de la Figure C1.

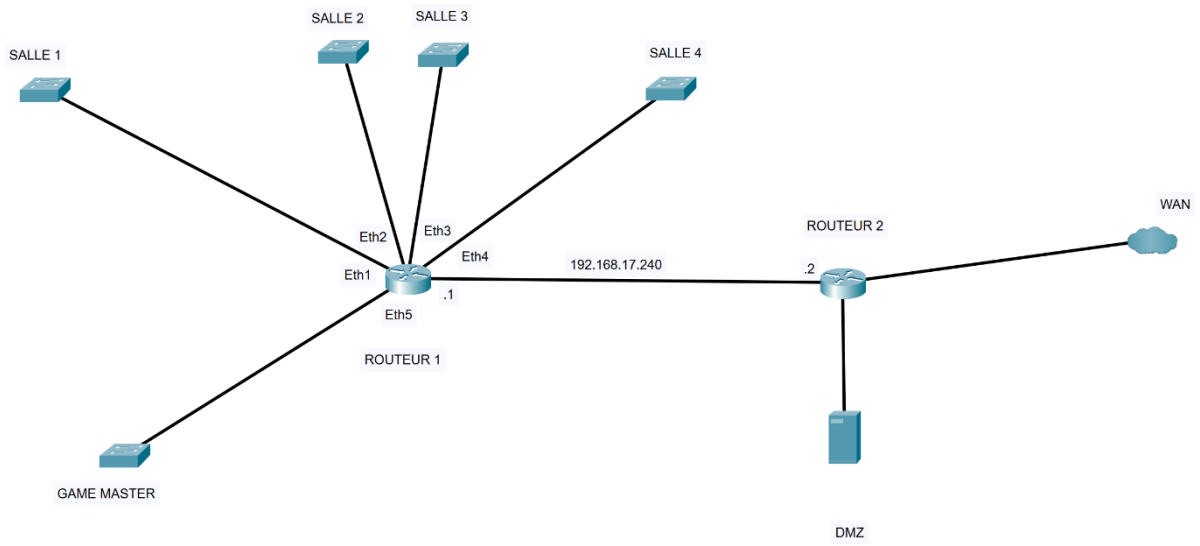


Figure C1 : nouvelle topologie envisagée du réseau

Des sous-réseaux vont être créés pour chaque salle de jeu ainsi que pour la salle de supervision du Game Master en appliquant un nouveau masque au réseau actuel : 192.168.17.0/28.

Dans un souci de protection des échanges avec l'extérieur, une zone DMZ est déployée afin d'accueillir un serveur web (192.168.17.225/28) et un serveur de données (192.168.17.226/28). Un réseau est également mis en place entre les deux routeurs présents sur cette architecture.

Question C1 — **Préciser** l'adresse du réseau actuel ainsi que le nombre de machines qui peuvent être interconnectés à celui-ci.

Question C2 — **Indiquer** quelle méthode est utilisée pour créer de nouveaux sous-réseaux.

Question C3 — **Donner** le nombre total de sous réseaux supplémentaires réalisables avec cette nouvelle configuration.
Préciser combien de ces sous-réseaux vont être utilisés avec la configuration actuelle ainsi que le nombre d'équipements qui pourront être associés à chaque sous réseau.

Question C4 | **Compléter** la colonne « Adresse du réseau » du document réponse DRC1 en indiquant l'adresse du sous réseau de chaque salle. Le plan d'adressage sera fait dans l'ordre croissant :
1ère adresse de réseau disponible = salle 1, 2ème adresse de réseau disponible = salle 2,…, 5ème adresse de réseau disponible = salle Game Master

Question C5 | **Compléter** le reste du tableau du document réponse DRC1 en indiquant les plages d'adresses qui seront disponibles pour chacune des salles.

Par souci d'uniformisation, on donne la première adresse disponible du réseau aux interfaces du routeur 1.

Question C6 | **Compléter** la table de routage du routeur 1 donnée sur le document réponse DRC2.

Question C7 | **Donner** le rôle de la DMZ et indiquer quelles sont les architectures de protection possibles.

Question C8 | Le ou les firewall mis en place assureront certaines protections. **Indiquer** les équipements qui seront accessibles depuis l'extérieur ?

Question C9 | Avec la configuration mise en place, **indiquer** le nombre de salles qui pourront être ajoutées sur le site.

Partie D (choix 1)

Inspection de la base de données et gestion du leaderboard

L'analyse de la base de données de l'Escape Game et l'écriture de requêtes doit permettre d'obtenir des statistiques sur les parties.

Objectif : Les cartes de commande enregistrent les instants auxquels les joueurs terminent les énigmes de chaque salle. Cette collecte de données permet de produire un classement des équipes les plus efficaces à destination des joueurs. Ces durées de résolution des différentes énigmes sont également examinées par les concepteurs de salle de la société afin d'ajuster la difficulté des différentes énigmes d'une même salle et d'améliorer l'expérience utilisateur.

La base de données est enregistrée au format MySQL et est organisée selon le schéma relationnel donné à la figure D1 :

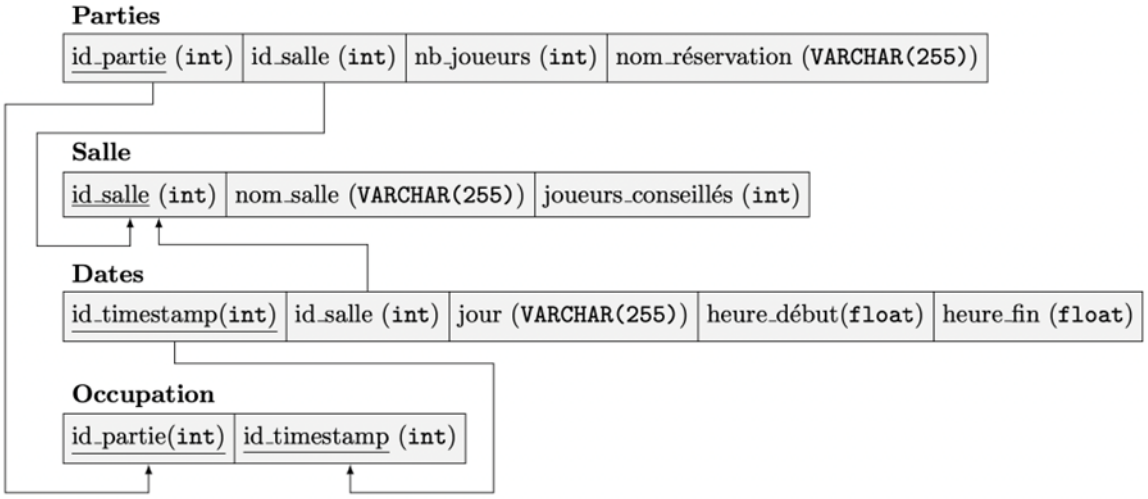


Figure D1 : schéma relationnel de la base de données étudiée.

Les types des différents attributs des tables Parties, Salle, Dates et Occupation sont indiqués entre parenthèses. Les clés primaires de chacune des tables sont soulignées. On note les contraintes suivantes dans les données de la table :

- Une seule réservation est possible par salle pour une heure de démarrage donnée.
- L'heure de fin d'une session correspond à l'heure à laquelle les joueurs terminent réellement la partie. Lors de la réservation celle-ci est fixée à la valeur NULL puis remplacée par la véritable heure une fois la session terminée.
- La table Parties contient le nom sous lequel les clients ont déposé une réservation, le nombre de joueurs qui seront présents ainsi que l'identifiant de la salle réservée et un identifiant unique pour chaque réservation. À titre d'exemple la salle Athena Station a pour identifiant 1.
- La table Salle décrit chaque salle de l'Escape Game en spécifiant son nom, le nombre de joueurs conseillés et un identifiant unique.
- La table Dates répertorie les créneaux horaires correspondant aux différentes réservations par salle.
- La table Occupation répertorie les parties et les créneaux horaires associés.
- Les heures sont enregistrées au format float : 17.5 code 17 heures et 30 minutes.

Un rappel de la syntaxe des requêtes SQL usuelles est fourni figure D2.

Utiliser (rendre active) une base de données existante	**USE** nom_de_la_base;
Créer une base de données	**CREATE DATABASE** nom_de_la_base;
Créer une table dans une base de données	**CREATE TABLE** nomTable (id **INT NOT NULL AUTO_INCREMENT**, champ1 **DOUBLE**, champ2 **VARCHAR**, champ3 **TIMESTAMP NOT NULL**, …, **PRIMARY KEY**(id)) ;
Écrire une nouvelle entrée dans une table de BDD	**INSERT INTO** nomTable(champ1, champ2) **VALUES**('valeur1', 'valeur2') ;
Modifier les informations de l'entrée dont le champ id = 51	**UPDATE** nomTable **SET** nomChamp1=10, valeur2=32 **WHERE** id=51 ;
Ajouter des nouveaux champs (colonnes) dans une table	**ALTER TABLE** nomTable **ADD** champ1 **DOUBLE**, **ADD** champ2 **BOOLEAN DEFAULT FALSE** ;
Sélectionner toutes les informations de la table	**SELECT** * **FROM** nomTable ;
Sélectionner seulement les informations d'un champ	**SELECT** nomChamp **FROM** nomTable ;
Sélectionner tous les champs de la table nomTable correspondant à deux critères	**SELECT** * **FROM** nomTable **WHERE** nomChamp1 = 'poste' **AND** nomChamp3 < 12 ;
Sélectionner sur plusieurs tables (jointure) nomTable1.nomChamp1 est clé primaire. nomTable2.nomChamp4 est une clé étrangère vers nomTable1	**SELECT** * **FROM** nomTable1 **JOIN** nomTable2 **ON** nom_table1.nomChamp1 = nom_table2.nomChamp4 ;
Appliquer une fonction d'agrégation (parmi **MIN**,**MAX**,**COUNT**,**AVG**) sur des valeurs d'un champs nomChamp1 regroupées par ensembles partageant une même valeur dans le champs nomChamp2	**SELECT MAX** (nomChamp1) **FROM** nomTable **GROUP BY** nomChamp2
Ordonner les résultats d'une requête par ordre croissant (**ASC**) ou décroissant (**DESC**) des valeurs d'un champ NomChamp1.	**ORDER BY** nomChamp1 **ASC** (instructions à ajouter en fin de requête)
Limiter le nombre d'entrées renvoyées par une requête en se retreignant aux *k* premières lignes où *k* est un entier.	**LIMIT** k (instruction à ajouter en fin de requête)

Figure D2 : rappel des requêtes SQL usuelles

Question D1 | **Identifier** les clés primaires et étrangères présentes dans le schéma de la figure D1

Question D2 | **Indiquer** quelle est la relation entre les tables Parties et Dates

Question D3 | **Ecrire** la requête SQL listant toutes les sessions de jeu réservées par une personne donnée (ex : 'Martin').

Question D4 | **Écrire** la ou les requêtes SQL permettant d'insérer dans la table Parties une partie à 4 joueurs pour 1 heure réservée par Mr Durand dans la salle Pirates des Antilles ayant pour identifiant 3. On suppose que ce créneau a déjà été ajouté dans les tables Dates et Occupation et que l'identifiant de partie id_partie vaut 2030**.**

Certaines salles sont plus difficiles que d'autres et nécessitent une attention particulière du maître de jeu pour aider les participants notamment lorsqu'ils sont peu nombreux.

Question D5 | **Écrire** une requête permettant d'extraire les réservations dont le nombre de joueurs est inférieur ou égal à 4.

Le planning de travail des *Game Masters* est constitué à partir de la table Dates qui permet de calculer le nombre de créneaux réservés sur une journée et d'en déduire le nombre de maîtres du jeu à convoquer sur cette journée.

Question D6 | **Écrire** une requête qui renvoie le nombre de créneaux réservés le 2 mars 2025. Les dates sont encodées dans la base de données sous la forme de chaînes de caractères au format JJ/MM/AAAA.

Question D7 | **Écrire** une requête permettant d'afficher les horaires (jour, heure de début, heure de fin) pour toutes les parties jouées dans la salle ‘’Athena Station’’.

Question D8 | **Écrire** une requête qui renvoie le meilleur score de la base de données (c'est-à-dire la durée la plus courte pour terminer la salle). La requête doit afficher le nom de la personne qui a réservé la salle, le nom de la salle terminée ainsi que l'heure de démarrage de la session.

Afin d'évaluer plus précisément la difficulté des énigmes et d'établir les records de vitesse par énigme et non par salle, la société souhaite modifier la base de données afin de stocker les instants auxquels chacune des énigmes d'une salle sont terminées. On considérera qu'une salle contient au plus 5 énigmes.

Question D9 | **Proposer** des modifications et/ou ajouts au schéma relationnel précédent afin de stocker les heures de début et de fin de chaque énigme à l'issue d'une session. Cette proposition devra contenir une table dont les attributs seront les heures de début et de fin de chacune des énigmes ainsi que d'autres attributs qui seront précisés.

Partie E (choix 2)

Conception d'une chaîne d'acquisition pour un jeu d'adresse

Il s'agit de dimensionner la chaîne d'acquisition associée à un capteur permettant d'acquérir les actions d'un joueur dans un contexte de suivi en temps réel d'un signal affiché sur un écran.

Objectif : Dans le cadre de l'une des énigmes, les joueurs sont conduits à manipuler des potentiomètres à glissière en vue de déplacer un curseur sur un écran et suivre un signal évoluant au cours du temps. On s'intéresse dans un premier temps à la modélisation de ce capteur puis on dimensionnera la chaîne d'acquisition associée en vue de l'application ciblée.

Figure E1 : potentiomètre glissière de 10 kΩ avec lequel les joueurs interagissent

Le potentiomètre glissière sélectionné a une résistance minimale de 0 Ω et une résistance maximale de R = 10 kΩ à chacune de ses 2 positions extrêmes. La longueur de la piste sur laquelle le bouton peut se déplacer est de 10 cm. On alimente le potentiomètre sur la borne VCCA reliée au potentiel 5 V et la borne GND reliée au potentiel 0 V.

Question E1 | **Rappeler** la grandeur physique mesurée par un potentiomètre en tant que capteur. **Préciser** en justifiant quelle est la nature du signal produit par ce capteur (nature de la grandeur physique et unité associée) et si ce signal est analogique ou numérique.

On note x la position en cm du bouton manipulé par les joueurs et L la longueur totale de la glissière. On propose le schéma électrique équivalent du potentiomètre :

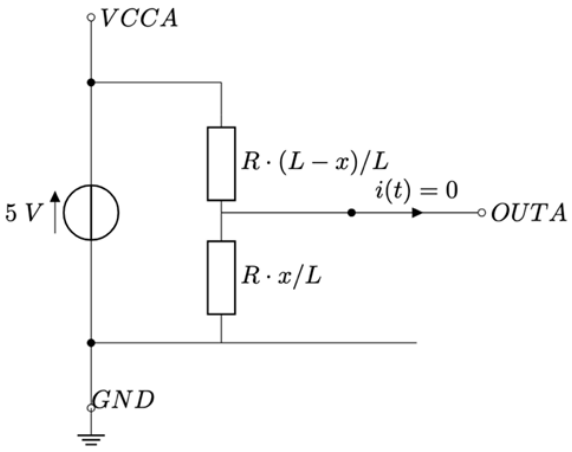


Figure E2 : schéma électrique équivalent du potentiomètre alimenté par une source de tension et reliée à un convertisseur analogique numérique

On suppose que le courant circulant dans la branche OUTA peut être négligé.

Question E2 | **Établir** l'équation liant la tension OUTA à la position *x* du bouton.

Dans la suite on utilisera la relation OUTA = $K \cdot x$ avec K = 0,5 $V \cdot cm^{-1}$.

La tension OUTA est numérisée au moyen d'un Convertisseur Analogique Numérique (CAN) unipolaire associant aux tensions mesurées un nombre codé sur un octet. On suppose donc que ce CAN code les tensions mesurées dans un registre de type unsigned integer sur 8 bits : uint8

Question E3 | **Rappeler** la plus grande et la plus petite valeur codables sur un uint8. **Préciser** le nombre de valeurs différentes qui pourront donc être codées. On notera N cette valeur dans la suite.

Le CAN est alimenté avec une tension Vcc_{CAN} = 5,5 V qui correspond à la plus grande tension qu'il pourra mesurer.

Question E4 | **Déterminer** la plus petite variation de tension (en V) que ce CAN peut distinguer. Cette quantité est appelée résolution du CAN. **Exprimer** cette résolution en fonction de Vcc_{CAN} et N puis procéder à l'application numérique.

En étendant la définition précédente, on appelle résolution en position la différence entre deux positions sur la glissière conduisant à une différence de tension en entrée du CAN égale à sa résolution. Le système d'acquisition est suffisamment précis si l'erreur d'estimation de la position de la glissière est inférieure à 1 mm.

Question E5 | En **déduire** l'expression de la résolution en position Δx (en cm) obtenue sur la glissière en fonction de la constante K établie Question E2, de Vcc_{CAN} et N puis **procéder** à l'application numérique et **conclure** sur la précision de la chaîne d'acquisition.

Dans la suite on affiche sur un écran un signal dont le joueur doit reproduire l'évolution de manière la plus précise possible en déplaçant la glissière du potentiomètre. On affiche sur la courbe ci-dessous le signal à suivre, c'est à dire l'évolution de position de la glissière souhaitée.

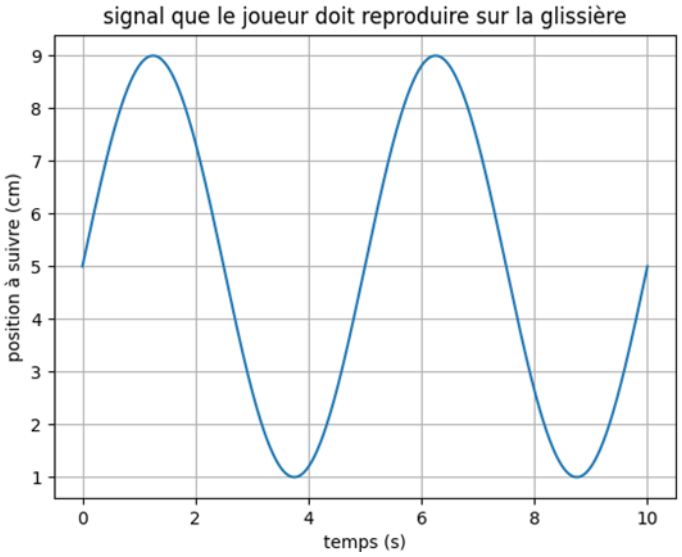


Figure E3 : signal à reproduire sur la glissière

Question E6 | **Donner** une expression du signal affiché sur la courbe en supposant que celle-ci est purement sinusoïdale. Exprimer cette fonction sous la forme $s(t) = A + B\,sin(\,\omega \cdot t\,)$ où les valeurs de A, B et ω sont à relever sur la courbe. Montrer que la fréquence de ce signal vaut alors environ 0,2 Hz.

Question E7 | **Déterminer** quelles valeurs de fréquences d'échantillonnage seraient correctes pour permettre une numérisation du signal respectant l'application du critère de Shannon.

On considère que l'énigme est résolue lorsque le joueur parvient à suivre la consigne avec un écart inférieur à 0,5 cm pendant au moins 5 secondes.
On s'intéresse au programme permettant d'implémenter cette énigme. On s'intéresse dans un premier temps à la fonction d'acquisition de la position du potentiomètre. Cette fonction effectue une lecture de la valeur du potentiometer.

Dans la question suivante, on suppose que l'on peut librement appeler les fonctions définies ci-après :

- get_time() : renvoie la valeur du temps à l'instant de l'appel en ms
- get_consigne(t) : renvoie la valeur de la courbe indiquant la position à suivre par le joueur s(t) (en cm) à l'instant t de l'affichage (exprimé en ms).
- get_position() : renvoie la valeur de la position du potentiomètre manipulé par le joueur (en cm).
- affiche_courbes(consigne,mesure) : affiche à l'écran la consigne de position demandée au joueur ainsi que la mesure de la position réelle du potentiomètre

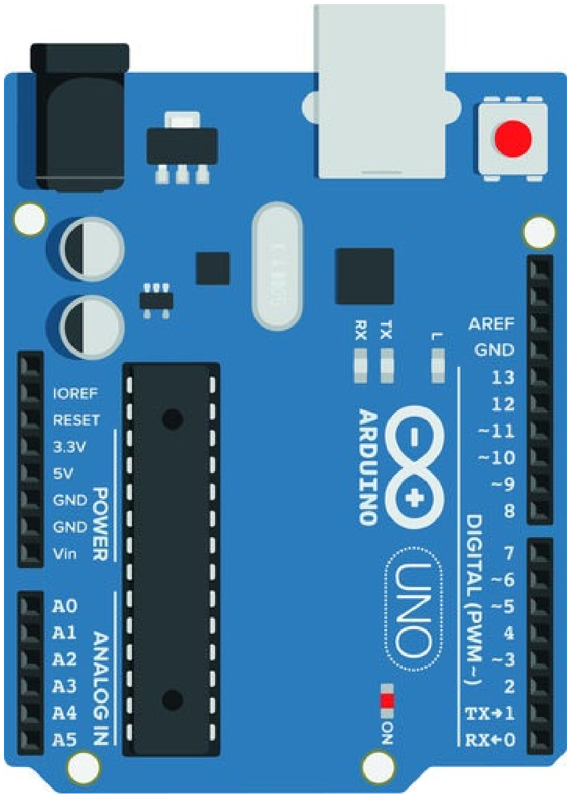
Question E8 | **Compléter** l'algorithme en pseudo-code du document réponse DRE1 permettant de vérifier cette condition à partir des données acquises et de déclencher l'action.

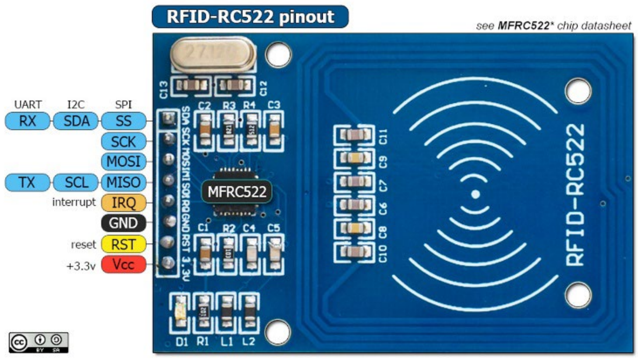
Les instructions de la boucle « tant que » du code précédemment doivent être exécutées suffisamment vite pour garantir un affichage fluide des courbes de la position de consigne et

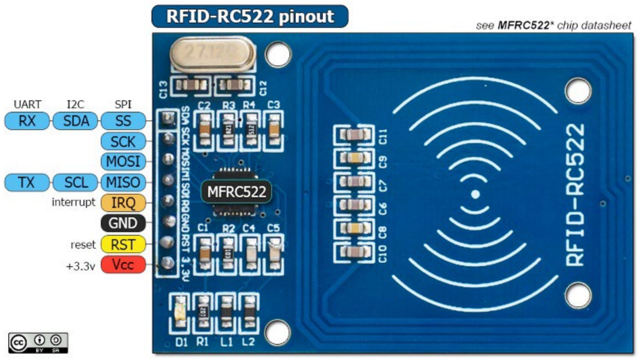
de la position de la glissière.

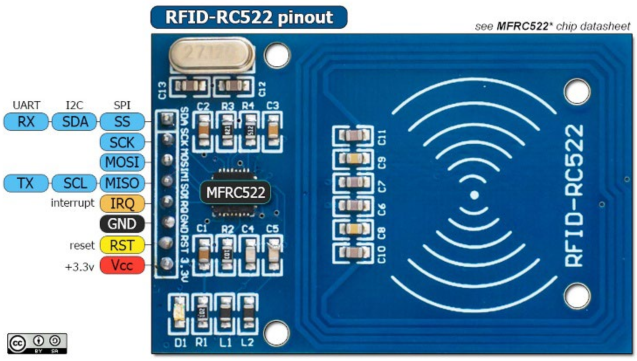
Question E9	En exploitant le résultat de la Question E7, **définir** la durée maximale d'exécution de la sequence des operations effectuées lors d'un passage dans la boucle « tant que ».
Question E10	En supposant que chaque instruction du code précédent s'exécute en un temps inférieur à 50 ms sur la carte électronique cible, **conclure** sur la faisabilité d'implémenter cette énigme avec le dispositif électronique étudié.

DOCUMENT RÉPONSE DRB1 (Question B5)









DOCUMENT RÉPONSE DRC1 (Question C4 et C5)

	Adresse du réseau	Adresse mini	Adresse maxi
Salle 1			
Salle 2			
Salle 3			
Salle 4			
Salle Game Master			
Zone DMZ			

DOCUMENT RÉPONSE DRC2 (Question C6)

Destination	Adresse réseau	Masque	Adresse interface	Passerelle
Salle 1				
Salle 2				
Salle 3				
Salle 4				
Game Master				
DMZ				

DOCUMENT RÉPONSE DRE1 (Question E8)

```
t_debut := get_time()
duree_suivi := 0 ## temps pendant lequel le joueur a correctement suivi la consigne
tant que duree_suivi < ……..….
        t_actuel := get_time()
        valeur_consigne := ……..….
        valeur_mesure := ……..….
        duree_suivi := ……..…..
        affiche_courbes(valeur_consigne,valeur_mesure)
        si ……..……
                t_debut := ……..……..
                duree_suivi := ……..…..
```

2) Attendus de l'épreuve (éléments de corrigé)

Conseils aux candidats
Il est demandé aux candidats :

- de rédiger les réponses aux différents exercices sur des feuilles de copie séparées et clairement repérées ;
- de numéroter chaque feuille de copie et indiquer le numéro de la question traitée ;
- de rendre tous les documents réponses, même non complétés ;
- d'utiliser exclusivement les notations indiquées dans le sujet lors de la rédaction des réponses ;
- de justifier clairement les réponses ;
- d'encadrer ou souligner les résultats ;
- de présenter lisiblement les applications numériques, sans omettre les unités, après avoir explicité les expressions littérales des calculs ;
- de formuler les hypothèses nécessaires à la résolution des problèmes posés si celles-ci ne sont pas indiquées dans le sujet.

Partie A
Implémentation d'une nouvelle énigme dans la salle Athena Station

Question A1 :
Un système RFID est constitué d'un émetteur actif et d'un récepteur passif (le tag). Lorsque le récepteur est placé près de l'émetteur, celui-ci est alimenté par les ondes électromagnétiques émises par l'émetteur et indique son identifiant par émission d'une onde électromagnétique. L'identifiant est alors reconnu par l'émetteur qui peut transmettre sa valeur au reste d'un circuit électronique. Cette technologie peut être utilisée pour identifier des badges dans une entreprise, ou des produits dans un magasin.

Question A2 :
Chaque pierre est une boule de plâtre contenant un récepteur rfid passif. Les socles contiennent des émetteurs rfid qui pourront identifier les récepteurs contenus dans les pierres.

Question A3 :
Un aimant pourrait effectivement être détecté par une sonde à effet Hall mais il ne serait pas possible d'identifier et reconnaître les différents aimants (même en mettant des aimants avec des aimantations différentes pour essayer de les reconnaître, le socle ne pourrait pas distinguer un aimant faiblement aimanté placé au milieu du socle d'un aimant fortement aimanté placé en bord de socle).

Question A4 :
```
def pierres_placees(id_sp_1, id_sp_2, id_sp_3, id_sp_4):
    if id_sp_1 == 1 and id_sp_2 == 2 and id_sp_3 == 3 and id_sp_4 == 4:
        return True
    else:
        return False
```

Question A5 :

```
def pierres_incompletes(id_sp_1, id_sp_2, id_sp_3, id_sp_4):
    if id_sp_1 != 0 and id_sp_2 != 0 and id_sp_3 != 0 and id_sp_4 != 0 and
pierres_placees(id_sp_1, id_sp_2, id_sp_3, id_sp_4)==False:
        return True
    elif id_sp_1==1 and id_sp_2==2 and id_sp_3==3 and id_sp_4 == 0 :
        return True
    else:
        return False
```

Question A6 :

```
def nombre_places(L):
    cpt = 0
    for i in range(len(L)):
        if L[i] == i:
            cpt+=1
    return cpt
```

Question A7 :
Il y a n ! combinaisons soit une durée de 10*(6 !) = 7200s = 120 minutes (c'est trop long pour que les joueurs puissent s'amuser à le faire sachant que les énigmes de la Station Athena doivent être terminées en moins d'une heure).

Question A8 :
A chaque passage dans le while, on décale d'un cran à droite tous les éléments qui sont mal placés :
[4, 0, 1, 3, 2]
[2, 4, 0, 3, 1]
[1, 2, 4, 3, 0]
[0, 1, 2, 3, 4]

Question A9 :
Chaque passage dans le while correspond à un test de la permutation par les joueurs. La stratégie consiste à décaler d'un cran à droite tous les éléments qui ne sont pas correctement placés. Dans le pire des cas, on doit décaler tous les éléments de n-1 positions à droite avant de trouver leur bon emplacement (c'est donc le cas où la liste initialement passée dans la fonction vaut [n,0,1,2,3 ,4,…,n-1]). On résout donc dans le pire des cas l'énigme en n-1 tests de permutations. Pour 6 tags, cela prendra donc (6-1)*10 = 50 secondes ce qui est nettement plus raisonnable que le cas où seul le nombre de badges bien placés était indiqué dans l'optique de faire une énigme (sachant que les joueurs n'utiliseront certainement pas cette stratégie efficace).

Partie B

Intégration matérielle d'une nouvelle énigme dans la salle Athena Station

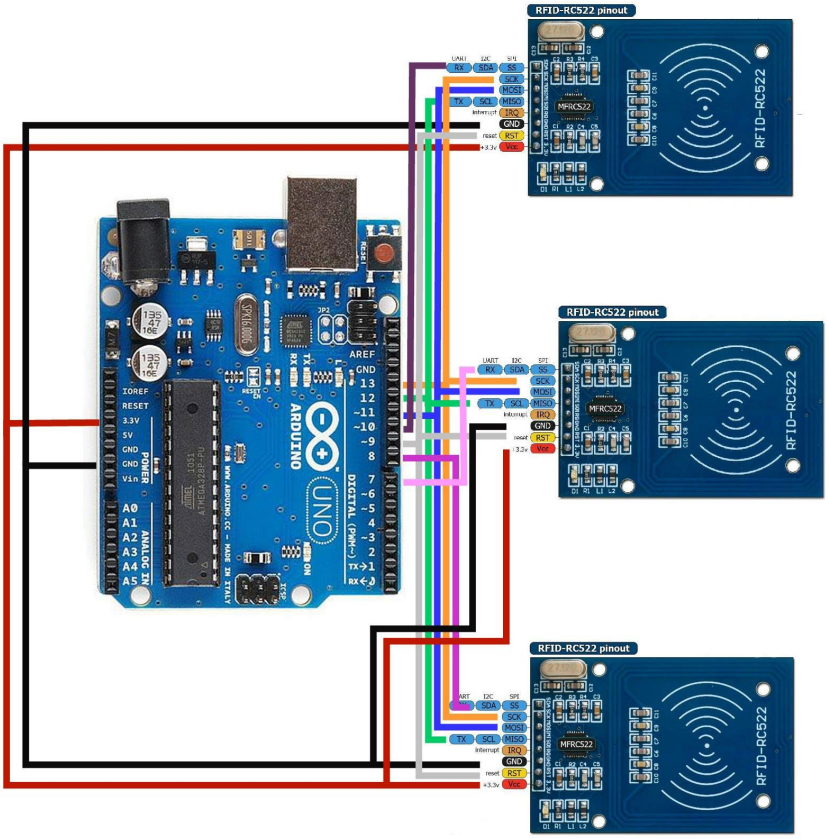
Question B1 :
Les trames sont émises par le tag reader et reçues par la carte Arduino. Il s'agit donc de trames envoyées sur l'entrée du maître par la sortie de l'esclave donc visibles sur la borne MISO. Il sera également nécessaire de relier la borne GND à l'oscilloscope pour disposer d'une masse commune.
On peut aussi relier les autres bornes mais elles ne serviront à rien pour la visualisation demandée.

Question B2 :
La transmission de 4 octets correspond à $4 \times 8 = 32\ bits$ par trame. Avec un débit maximal de 10 Mbits /s, cela correspond à un temps de transmission minimal de $\frac{32}{10 \cdot 10^6} = 3.2 \cdot 10^{-6}\ s = 3.2\ \mu s$.

Question B3 :
Un bit de parité est un bit qui est ajouté à la fin d'une trame de manière à fixer la parité du nombre de bits à 0 ou à 1 transmis. Par exemple on peut choisir de mettre ce bit à 1 si le nombre de 1 dans la trame est impair et de le mettre à 0 sinon. Ainsi le message complet (trame + bit de parité aura toujours un nombre de 1 pairs). Ce bit de parité permet de renforcer la fiabilité de la communication : Si un bit a été mal reçu (un 1 transformé en 0 ou inversement) alors le nombre de 1 ne sera pas cohérent avec la valeur du bit de parité et le récepteur pourra déterminer que la trame reçue est corrompue (et éventuellement ne pas la prendre en compte).

Question B4 :
Un seul esclave à la fois ne peut transmettre des données sur le bus, il n'y aura donc qu'un seul esclave avec la borne NSS à l'état bas. Il faut donc avoir un fil différent pour chaque esclave (sinon 2 esclaves pourraient avoir cette borne à l'état bas en même temps s'ils partagent un fil).

Question B5 :



Question B6 :
La variable *flag* change d'état et passe à True lorsque le tag rfid posé (ou non) sur l'un des tag reader change. A la fin d'un tour de lecture des 4 tags readers, si la variable flag est à True, la carte Arduino transmet l'identifiant des tags posés (ou non) sur chaque tag reader à la carte Raspberry Pi. Si en revanche aucun des tags posés sur les tags reader n'est changé, la variable flag reste à False et il n'y a pas de transmission effectuée vers la carte Raspberry Pi.

Question B7 : La seule tâche de la carte Arduino est d'inspecter régulièrement les différents tags reader et ne transmettre les identifiants des tags lus à la carte Raspberry Pi que lorsqu'il y a au moins un changement. Ceci permet de délester la carte Raspbery Pi de cette tâche en consacrant son temps d'exécution à la gestion des autres périphériques de la salle et de l'interface avec le maître du jeu.

Question B8 : Le débit en bauds détermine le nombre de symboles transmis par seconde. Lorsque la transmission s'effectue sur un canal physique avec 2 états (haut et bas) alors chaque symbole

correspond à un bit et le débit en baud est égal au débit en bits/s. En revanche si une modulation permet de transmettre plusieurs bits par symbole transmis (par exemple en mettant un fil à 8 niveau de tensions différents, la lecture du niveau de tension côté récepteur permet de récupérer une valeur parmi 8 donc 4 bits) alors le débit en bauds sera différent du nombre de bits transmis par seconde.

Question B9 : Une liaison synchrone est une liaison pour laquelle un signal d'horloge est partagé entre l'émetteur et le récepteur. Ainsi la liaison SPI étudiée précédemment est synchrone. En revanche la liaison UART (Universal Asynchronous Receiver Transmitter) repose sur une transmission via seulement deux fils (TX et RX) et il n'y a pas de signal d'horloge partagé elle est donc asynchrone.

Question B10 :
A la question B2, on a calculé un ordre de grandeur du temps de récupération d'un identifiant sur un tag rfid **:** $3.2\ \mu s$. Il faut donc prendre 4 fois ce temps pour collecter les 4 identifiants soit $12.8\ \mu s$. Ces 4 identifiants (correspondant à $4\ \times 4 = 16\ octets$) doivent être transmis à la carte Raspberry Pi au débit de 9600 bauds (soit 9600 bits / s sur une transmission UART à 1 bit / symbole). Ceci prendra un temps de transmission égal à $\frac{16\times 8}{9600} \simeq 0.013\ s = 13\ ms$. Le temps de collecte des identifiants sur le bus SPI calculé avec le débit maximal est négligeable devant ce temps de transmission, on peut donc supposer que dans un fonctionnement normal (et non optimal), le temps de récupération par la carte Raspberry Pi des identifiants des 4 tags est approximativement égal au temps de transmission sur la liaison UART entre la carte Arduino et la carte Raspberry Pi. Cette durée de 13 ms est largement inférieure à la limite de 200 ms tolérée, donc le système proposé sera valide pour l'application étudiée.

Partie C
Évolution de l'architecture réseau

Question C1 :
L'adresse est 192.168.17.0 et le nombre de machines connectables est 2^8 – 2 = 254 hôtes

Question C2 :
Modifier le masque de sous réseau en augmentant le nombre de bits associées au réseau (+4)

Question C3 :
Réponse : 2^4 = 16 sous réseaux supplémentaires – 7 sont actuellement utilisés et il est possible de connecter 2^4 – 2 = 14 hôtes

Question C4 :
Le plan d'adressage sera fait dans l'ordre croissant :
1er adresse de réseau disponible = salle 1, 2ème adresse de réseau disponible = salle 2,…, 5ème adresse de réseau disponible = salle Game Master

Question C5 :

	Adresse du réseau	Adresse mini	Adresse maxi
Salle 1	192.168.17.0	192.168.17.1	192.168.17.14
Salle 2	192.168.17.16	192.168.17.17	192.168.17.30
Salle 3	192.168.17.32	192.168.17.33	192.168.17.46
Salle 4	192.168.17.48	192.168.17.49	192.168.17.62
Salle Game Master	192.168.17.64	192.168.17.65	192.168.17.78

Question C6 :

Destination	Adresse réseau	Masque	Adresse interface	Passerelle
Salle 1	192.168.17.0	255.255.255.240	192.168.17.1	Connecté ou direct
Salle 2	192.168.17.16	255.255.255.240	192.168.17.17	Connecté ou direct
Salle 3	192.168.17.32	255.255.255.240	192.168.17.33	Connecté ou direct
Salle 4	192.168.17.48	255.255.255.240	192.168.17.49	Connecté ou direct
Game Master	192.168.17.64	255.255.255.240	192.168.17.65	Connecté ou direct
DMZ	192.168.17.224	255.255.255.240	192.168.17.241	192.168.17.242

Question C7 :
DMZ : Zone démilitarisée qui permet de rendre accessible certaines parties de votre réseau (ici le serveur web et le serveur de données) tout en protégeant votre réseau local (LAN)
2 architectures possibles : Avec un ou deux firewalls.

Question C8 :
Les équipements présents dans la DMZ (serveur web et serveur de données actuellement).

Question C9 :
Actuellement 7 sous réseaux sont utilisés, il reste donc 9 Sous réseaux possibles donc 9 salles.

Partie D
Inspection de la base de données et gestion du leaderboard

Question D1 :
Clés primaires :
- Parties(id_partie)
- Salle(id_salle)
- Dates(id_timestamp)
- Occupation(id_partie, id_timestamp) (composée)

- Clés étrangères :
  - Parties.id_salle → Salle.id_salle
  - Dates.id_salle → Salle.id_salle
  - Occupation.id_partie → Parties.id_partie
  - Occupation.id_timestamp → Dates.id_timestamp

Question D2 :
C'est une relation indirecte via la table Occupation. Une partie est associée à un créneau horaire via id_partie et id_timestamp.

Question D3 :
```
SELECT *
FROM Parties
WHERE nom_réservation = 'Martin'
```

Question D4
```
INSERT INTO Parties (id_partie, id_salle, nb_joueurs, nom_réservation)
VALUES (2030,3,4,'Mr Durand')
```

Question D5 :

```
SELECT * FROM Parties WHERE nb_joueurs <= 4
```

Question D6 :

```
SELECT COUNT(*) FROM Dates WHERE jour = '02/03/2025'
```

Question D7 :

```
SELECT jour, heure_début, heure_fin
FROM Parties as P JOIN Salle as S JOIN Occupation as O JOIN Dates as D
ON P.id_salle = S.id_salle AND O.id_partie = P.id_partie AND D.id_timestamp
= O.id_timestamp
WHERE S.nom_salle = 'Athena Station'
```

Question D8 :

```
SELECT nom_reservation,nom_salle,heure_début
FROM Parties as P JOIN Dates as D JOIN Salle as S JOIN Occupation as O
ON P.id_partie = O.id_partie AND P.id_salle = S.id_salle AND S.id_salle =
D.id_salle AND D.id_timestamp = O.id_timestamp
WHERE heure_fin -heure_début = (SELECT MIN (heure_fin – heure_début) FROM
Dates)
```

Question D9 :

On introduit une table dont les champs sont les instants debut_enigme_1,fin_enigme_1,debut_enigme_2,fin_enigme_2,… fin_enigme_5 (ou plus généralement fin_enigme_k où k le nombre d'énigme maximal pris dans la salle de l'escape Game qui contient le plus d'énigmes). Par ailleurs cette table doit contenir au moins un champs permettant d'effectuer des jointures. On peut par exemple ajouter le champs id_partie qui sera une clé primaire de cette table et qui permettra d'effectuer des jointures avec la table Parties.

Partie E

Conception d'une chaîne d'acquisition pour un jeu d'adresse

Question E1 : Un potentiomètre convertit une position (angulaire en rad pour un potentiomètre rotatif ou linéaire en m pour un potentiomètre linéaire) en tension en V. Ce composant produit un signal analogique (les tensions produites prennent des valeurs continues).

Question E2 : En appliquant la formule du pont diviseur de tension (on peut aussi trouver cette équation en combinant une loi des mailles et les équations liant la tension aux bornes des résistances au courant identique qui les traverse).

OUTA = VCCA (Rx/L)/(Rx/L + R(1-x)/L)) = VCCA x/L

$$OUTA = VCCA \cdot \frac{R \cdot \frac{x}{L}}{\left(R \cdot \frac{x}{L} + R \cdot \frac{1-x}{L}\right)} = \frac{VCCA}{L} \cdot x$$

Question E3 : La plus petite valeur codable (car codée sur un entier non signé) est 0 et la plus grande est codée par le nombre binaire rempli de 1 qui vaut $2^7 - 1 = 255$. On peut donc coder N = 256 valeurs différentes.

Question E4 : La plus grande valeur 255 sera associé aux tensions supérieures à 5,5V et la plus petite valeur 0V aux tensions négatives. 2 valeurs successives entre 0 et 255 correspondent à un espacement de tension de VCCCAN / (N-1) = 5,5/255 ≃ 21,6 mV.

Question E5 : La résolution en tension vaut $\frac{VCCCAN}{N-1}$. La résolution en position correspond à la différence de position qui permet de faire varier OUTA d'une tension inférieure ou égale à la résolution en tension. On utilise donc la relation impliquant K en remplaçant OUTA par la résolution en tension pour obtenir $\Delta x = \frac{VCCCAN}{K\cdot(N-1)} = \frac{5,5}{0.5\cdot 255} = 0{,}043\ cm = 0{,}43\ mm$. Cette valeur est inférieure à l'exigence d'1 mm donc la chaîne d'acquisition est suffisamment précise.

Question E6 : A t=0 , on mesure A = 5 cm
A $t = \frac{2\cdot\pi}{\omega}$ on a atteint la fin de la première période (donc au bout de 5 secondes) ; d'où $\omega = \frac{5}{2\cdot\pi} \simeq 0.8\ rad/s$. On voit que l'amplitude du signal vaut B = 4 cm. Par ailleurs la fréquence du signal vaut $f = \frac{\omega}{2\cdot\pi} \simeq 0.2\ Hz$.

Question E7 : D'après le critère de Shannon, la fréquence d'échantillonnage doit être au moins égale au double de la plus haute fréquence constituant le spectre du signal. Ici le signal n'est constitué que d'un seul signal sinus donc la seule fréquence f déterminée à la question précédente. On aura donc besoin d'une fréquence d'échantillonnage au moins égale à $2 \cdot 0.2 = 0.4\ Hz$.

Question E8 :

```
t_debut := get_time()
duree_suivi := 0 ## temps pendant lequel le joueur a correctement suivi la
consigne
tant que duree_suivi < 5000 duree en ms
      t_actuel := get_time()
      valeur_consigne := get_consigne(t_actuel)
      valeur_mesure := get_position()
      duree_suivi := t_actuel - t_debut
      affiche_courbes(valeur_consigne,valeur_mesure)
      si valeur_mesure<valeur_consigne-5 or valeur_mesure>valeur_consigne+5
            t_debut := t_actuel
            duree_suivi := 0
```

Question E9 : La fréquence d'échantillonage minimale étant de 0,4Hz, la période d'échantillonage (qui sépare ici deux mesures effectuées par l'instruction get_mesure() dans la boucle while) doit être d'au plus 1/0,4Hz = 2,5 s.

Question E10 : La boucle while comporte 9 lignes qui seront exécutées dans le pire des cas (si la condition du if est vraie) donc un temps d'exécution de $9 \times 20 = 180\ ms$ ce qui est largement inférieur aux 2,5 s maximum entre 2 mesures de la position du potentiomètre. On pourra donc utiliser le dispositif étudié pour implémenter l'énigme.

3) Extrait de l’arrêté du 17 avril 2025

Extrait de l’annexe de l’arrêté du 17 avril 2025 fixant les modalités d'organisation du concours externe du certificat d'aptitude au professorat de l’enseignement du segond degré, publié au Journal Officiel du 19 avril 2025

A. - Epreuves d'admissibilité

1° Première épreuve d'admissibilité.

L'épreuve consiste en l'étude de la modélisation d'un produit.
Elle a pour objectif de vérifier la capacité du candidat à conduire une étude d'un produit pour en caractériser les performances en mobilisant ses connaissances scientifiques et technologiques relevant de l'option du concours.
L'épreuve est constituée de plusieurs parties indépendantes, certaines d'entre elles sont à traiter obligatoirement et d'autres sont au choix du candidat.
Durée : quatre heures.
Coefficient 3.
L'épreuve est notée sur 20. Une note globale égale ou inférieure à 5 est éliminatoire ;