



**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 seconde 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 seconde épreuve d'admissibilité

CAPET BAC + 3 Sujet 0

Sommaire

	Page
Sommaire.....	2
Présentation du support	2
ÉTUDE A (à traiter obligatoirement)	3
ÉTUDE B (à traiter obligatoirement)	6
ÉTUDE C (à traiter obligatoirement)	10
DOCUMENT TECHNIQUE DTC1	15
DOCUMENT TECHNIQUE DTC2	16
DOCUMENT TECHNIQUE DTC3	16

Les études A,B et C sont à traiter obligatoirement par tous les candidats.

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 GameMaster.



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

ÉTUDE A (à traiter obligatoirement)

Analyse structurelle et fonctionnelle d'une salle de l'Escape Game Kairos.

L'objectif de cette partie est d'analyser les composants d'une salle de jeu et de concevoir l'architecture du réseau liant les éléments d'une salle de jeu au poste de supervision du Game Master.

La société Kairos escape Game héberge des salles de jeu dont les décors et les énigmes traduisent des ambiances et époques variées : Une salle à l'ambiance Pirates des Antilles, une salle à l'ambiance futuriste nommée Athena Station (Figure 1), et 2 salles traduisant des ambiances de bandes dessinées. Chaque salle comporte des systèmes permettant d'acquérir la solution d'une énigme résolue par un utilisateur. La nature de ces systèmes d'acquisition dépend de la thématique de la salle et repose sur des contraintes artistiques : la salle Pirates des Antilles contient des systèmes de déclenchement invisibles pour l'utilisateur, la salle Athena Station contient des claviers et des boutons poussoirs permettant de traduire une ambiance futuriste et l'une des salles à la thématique bande dessinée contient un téléphone à cadran analogique apparaissant dans l'œuvre littéraire originale. Chaque salle de jeu permet d'accueillir jusqu'à 6 joueurs simultanément. Ils sont sous la surveillance d'un Game Master qui supervise le bon déroulement de la partie depuis une salle distante via une interface connectée à la salle de jeu.

Question A1 | **Compléter** le diagramme des cas d'utilisation (document réponse DRA1) en ajoutant l'acteur manquant ainsi que les relations avec les différents cas d'utilisation sur lesquels il interagit.

Question A2 | **Ajouter** au diagramme du document réponse DRA1 le cas d'utilisation « Changer le mot de passe d'un terminal ». Ce cas d'utilisation constitue l'une des manières de reconfigurer une énigme.

Dans la suite on se focalisera sur la description structurelle de la salle futuriste Athena Station dont certains éléments sont représentés figure A1 et A2.

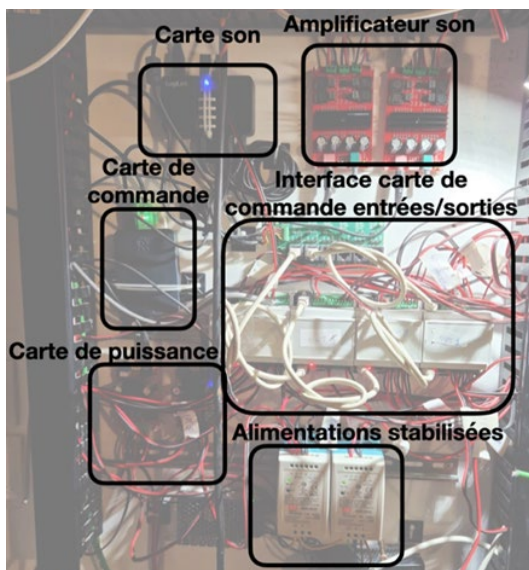


Figure A1 : contenu de l'armoire de commande d'une salle



Figure A2 : ventouse permettant l'ouverture d'une porte (à gauche) et caméra IP permettant le suivi de la partie (à droite)

Description structurelle d'une salle de jeu :

Chaque salle est équipée d'une carte de commande Raspberry-Pi effectuant la centralisation des données saisies par les joueurs sur des systèmes d'acquisition dédiés. La salle futuriste Athena Station est munie de 3 **claviers** et de 2 **digicodes** permettant de saisir des mots de passe et des codes. Des informations sont affichées en retour à ces joueurs sur 4 **écrans** situés dans la salle. Les données saisies sur les systèmes d'acquisition sont transmises à la **carte de commande** Raspberry-Pi située dans une **armoire de commande** représentée figure A1.

Cette carte de commande est chargée de :

- Recevoir les données saisies par les joueurs. Lorsqu'un code correct est saisi, la carte de commande doit modifier les images affichées sur les écrans et envoyer un ordre à une **carte de puissance** chargée de déclencher l'ouverture de portes dans la salle via des dispositifs appelés **ventouses électromagnétiques** (constituées d'électro-aimants). Une telle ventouse est représentée figure A2. Les différents systèmes d'acquisitions ainsi que la carte de puissance sont connectés à la carte de commande via une **interface carte de commande entrées/sorties**.
- Transmettre des sons aux **haut-parleurs** situés dans la salle. Ces sons sont constitués des indications données par le maître du jeu ainsi que des éventuels bruitages liés à l'ambiance de la salle tels que des bruits d'explosion ou d'ouverture de serrure. Ces sons sont combinés et mixés par une **carte son** dont le signal est amplifié par un **amplificateur son** associé à chaque haut-parleur.

Par ailleurs le suivi des activités des joueurs est effectué par un *Game Master* à distance depuis une pièce de supervision. Les images et les sons de l'activité des joueurs dans la salle lui sont transmis directement grâce à une **caméra IP** telle que celle représentée figure A2.

Question A3

Compléter le tableau du document réponse DRA2 qui indique les relations de contenance entre les éléments de l'Escape Game en utilisant les noms des sous-systèmes indiqués en gras dans la description structurelle d'une salle de jeu.

Les caméras IP d'une même salle ainsi que la carte de commande sont connectées à un switch lui-même connecté à un réseau interne.

Question A4

Rappeler la fonction d'un switch.

L'interface Game Master est une page web générée par la carte de commande de la salle lorsque celle-ci reçoit une requête http sur le port 8000. L'adresse IP de la carte de commande a été fixée en mode static à la valeur 192.168.17.1 par le concepteur de la salle.

Question A5

Donner l'url que le Game Master doit saisir dans un navigateur web pour accéder à la page web générée par la carte de commande.

Le serveur web récupère les images issues des caméras IP afin de les insérer dans la page visualisée par le Game Master. Le paramétrage des caméras IP est choisi en mode IP fixe.

Question A6

Expliquer ce que cela signifie et **préciser** l'intérêt de ce paramétrage.

Les Escapes Games ont récemment gagné en popularité du fait des avancées technologiques. Les premières entreprises proposaient une expérience ludique constituée d'une série d'énigmes dont la résolution de chaque problème permettait de trouver un code débloquent un cadenas et permettant d'accéder à l'énigme suivante. La solution développée par la société Kairos nécessite la mise en œuvre d'un système technique évolué étudié dans cette partie.

On s'intéresse désormais à la mise en œuvre des différents composants de système. Afin de simplifier l'étude, on se restreint à une description n'incluant qu'une seule salle de jeu et non pas quatre comme c'est le cas dans les locaux réels de la société.

- | | |
|--------------------|---|
| Question A7 | Compléter le diagramme de blocs internes sur le document réponse DRA3. Ce diagramme devra faire apparaître les différents flux entre les éléments de l'armoire de commande de l'Escape Game. Les flux d'énergie seront indiqués avec une flèche de couleur verte et les flux de données avec une flèche de couleur noir. |
| Question A8 | Identifier les éléments ayant trait à l'immersion des joueurs dans le jeu en les entourant sur le diagramme précédent (DRA3). Ces éléments ne sont pas nécessairement contenus dans l'armoire de commande. |
| Question A9 | Rédiger une synthèse récapitulant les différentes solutions technologiques mises en place par l'entreprise Kairos pour améliorer l'immersion du joueur et fluidifier son expérience ludique. |

ÉTUDE B (à traiter obligatoirement)

Architecture logicielle de la gestion des énigmes d'une salle.

L'objectif de cette partie est de s'approprier l'architecture du programme exécuté sur la carte de commande en vue d'y apporter des modifications telles que l'ajout d'une énigme ou la modification d'une énigme existante.

Les différentes salles contiennent des énigmes dont les solutions doivent être transmises à la carte de commande de la salle afin de débloquent l'énigme suivante. La thématique de la salle a une incidence sur la nature des systèmes qui s'y trouvent et donc sur la nature des données transmises à la carte de commande. Malgré la diversité des dispositifs d'acquisition, l'organisation des énigmes de toutes les salles est la même : la résolution d'une énigme permet de déclencher une transition vers la suivante et celles-ci se suivent les unes après les autres. Cependant, il peut arriver que la carte de commande ne parvienne pas à prendre en compte la fin d'une énigme (à cause d'un terminal utilisateur cassé par exemple), c'est pourquoi la transition d'une énigme à la suivante peut être forcée par un maître de jeu communiquant avec la carte de commande via une interface web.

Architecture logicielle de la gestion des énigmes d'une salle :

L'interface affichée sur l'écran du Game Master lors du lancement du mode préparation (pendant lequel un opérateur va cacher des indices dans la salle et y placer des objets) est représentée figure B1.

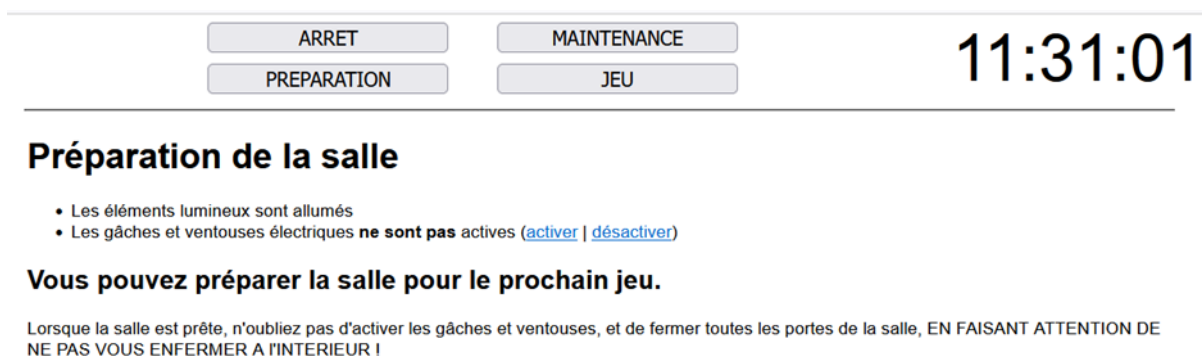


Figure B1 : interface Game Master pendant la phase de préparation

L'organisation des tâches exécutées par la carte de commande est décrite dans le diagramme d'état figure B2.

Les différentes transitions Début maintenance, Fin maintenance, Début préparation, Fin préparation, Lancement partie et Arrêt carte commande sont activées par le Game Master en cliquant sur des boutons associés dans son interface de commande.

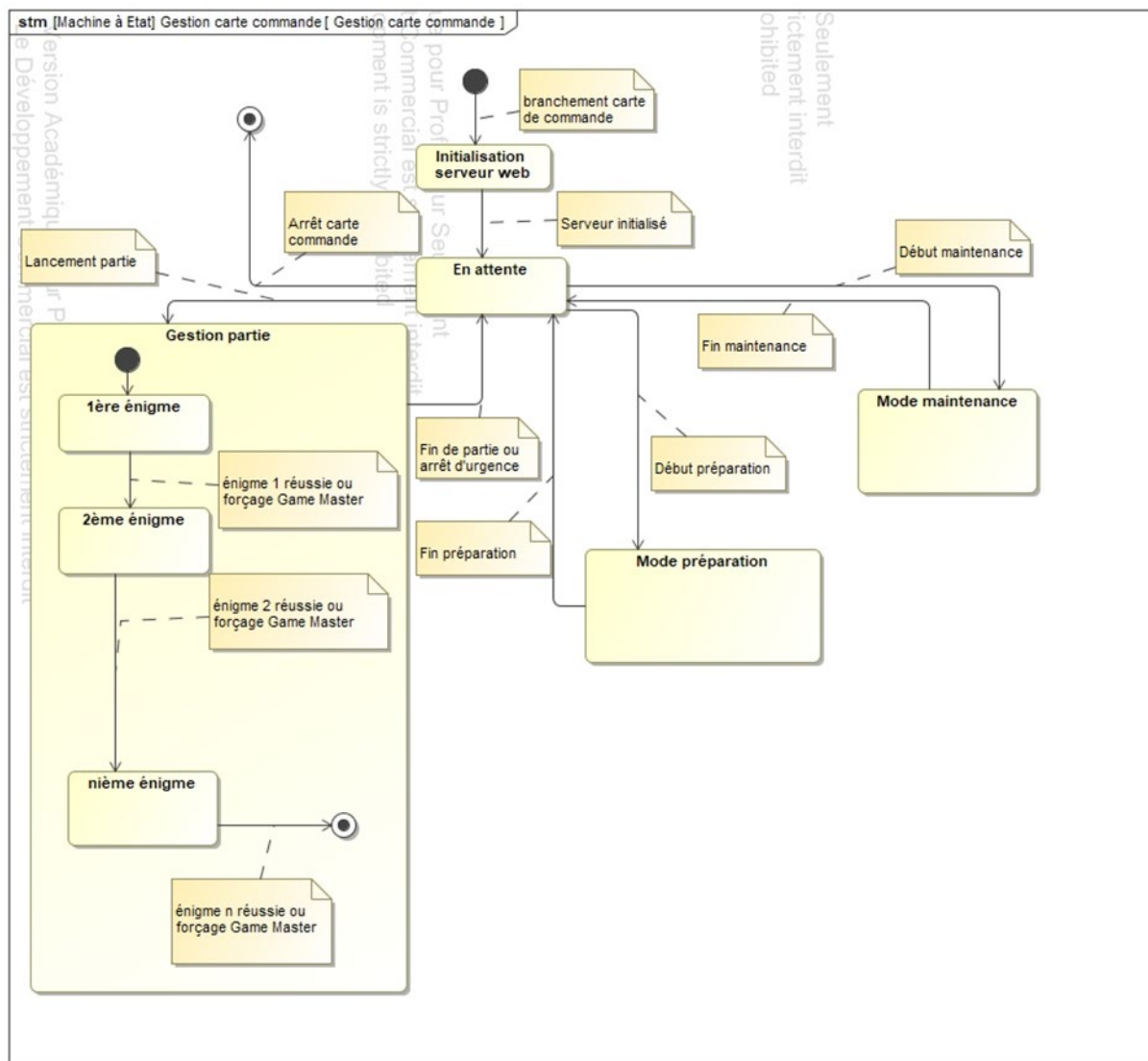


Figure B2 : diagramme d'état du système

Question B1

Compléter le diagramme d'état DRB1 associé au mode préparation en faisant apparaître les états :

- Gâches et ventouses activées
- Gâches et ventouses inactives
- Éléments lumineux allumés (il s'agit des éclairages et éléments de signalisation)

Ainsi que les transitions activer et désactiver déclenchées par l'appui sur les liens soulignés dans l'interface affichée figure B1.

La machine à état est implémentée dans une fonction en langage C. Celle-ci est codée par une procédure nommée `salle_machine_etat()` qui est exécutée de manière répétée dans un processus instancié lors du passage dans l'état « Gestion partie » représenté figure B2. Cette procédure est exécutée en parallèle d'une autre procédure `gestion_serveur()` permettant les interactions entre la carte de commande et le Game Master via le serveur web.

Question B2

Rappeler brièvement le principe du multithreading. **Proposer** une autre stratégie permettant d'exécuter ces deux tâches simultanément sur la carte Raspberry Pi.

On donne dans la figure B3 le code de la procédure `salle_machine_etat()` d'une machine qui n'est constituée pour des raisons de simplification que de 4 énigmes. Les fonctions `_enigme_i()` sont chargées de l'activation des différents périphériques liés à l'énigme en cours et renvoient 1 si l'énigme a été réussie. Les fonctions `_deblocage_GM_i()` consultent les données envoyées par le Game Master au serveur web et renvoient 1 si un ordre de déblocage de l'énigme en cours a été reçu.

```
static u16 cyc ;
static u16 verrou ;
static u32 save_time;
static u16 success_enigme;

// u16 : unsigned integer sur 16 bits, u32 : unsigned integer sur 32 bits

static void salle_machine_etat ( void ) {
cyc = 0;
verrou = 0;{
switch( cyc )
{ case 0 :
    if ( _enigme_0() == 1 || ( _deblocage_GM_0() == 1)
        {cyc++;}
    break;
case 1 :
    if ( _enigme_1() == 1 || _deblocage_GM_1() == 1)
        {save_time = _GameElapsedTime(); // sauvegarde
de l'instant (en secondes) où l'énigme 1 a été passée
        cyc++;}
    break;
case 2 : // Au bout de 3 minutes si l'énigme 2 n'est pas résolue, on verrouille la porte d'entrée
et on active la lumière d'urgence. Attention cet évènement ne doit être déclenché qu'une fois.

    if ( ..... )
    { verrou = 1 ;
      _roomEventAdd( "Verrouillage après 3 minutes");
      _outputLevelSet( ESPACE_GACHE_PORTE_ENTREE, OUTPUT_LEVEL_ON );
      _roomEventAdd( "Changement du mode d'éclairage" );
      _outputLevelSet( LUMIERE_STANDARD,LEVEL_OFF );
      _outputLevelSet( LUMIERE_URGENCE,LEVEL_ON );
      break;}
    if ( ..... )
    {cyc++;}
    break;
case 3 :
    if ( _enigme_3() == 1 || _deblocage_GM_3() == 1)
        {cyc++;}
    break;}}
```

Figure B3 : procédure `salle_machine_etat()`

Question B3

Proposer sur votre copie des instructions permettant de compléter le code associé à l'énigme activée par l'appel à la fonction `_enigme_2()`. Cette énigme doit être résolue par les joueurs en un temps suffisamment

court conformément aux indications dans les commentaires du code.
Dans le cas contraire, l'état des lumières de la salle est modifié.

Afin d'accorder plus de liberté aux joueurs, les scénaristes souhaitent que les joueurs puissent traiter dans l'ordre ou dans le désordre les énigmes 4 et 5 qui sont implémentées par les fonctions `_enigme_4()`, `_enigme_5()`, `_deblocage_GM_4()` et `_deblocage_GM_5()`.

Question B4 | **Proposer** sur votre copie le code à ajouter à la fin de la procédure `salle_machine_etat()` pour implémenter ce choix des scénaristes.

On s'intéresse désormais à l'implémentation d'une nouvelle énigme dans le scénario de l'Escape Game en apportant des modifications au code existant.

L'objectif est d'implémenter l'une des énigmes qui correspond à la recherche d'un code à saisir en un temps limité sur un digicode rétro-éclairé à 10 chiffres initialement éclairés en rouge. Le code à trouver est constitué de 4 chiffres.

- Lorsqu'un chiffre est saisi dans le bon ordre, son rétro-éclairage passe en vert.
- Si un des chiffres est faux dans la composition du code, tous repassent au rouge et il faut recommencer la saisie depuis le début.
- Par ailleurs, si les joueurs mettent plus de 2 secondes à appuyer sur un chiffre, là encore tous repassent au rouge et il faut recommencer la saisie depuis le début.
- Si les joueurs ne parviennent pas à résoudre l'énigme, le Game Master dispose d'une commande permettant de valider automatiquement l'énigme.

Question B5 | **Proposer** un diagramme d'état décrivant le déroulement de cette énigme. On pourra notamment faire figurer un état différent pour chaque numéro correctement identifié. Aucun code n'est attendu pour la mise en œuvre de cette énigme

Question B6 | **Spécifier** la ou les nouvelles fonctions à écrire pour implémenter cette énigme en détaillant la nature de leurs entrées et sorties ainsi que les opérations effectuées par ces fonctions. Il n'est pas demandé de rédiger le code de ces fonctions.

Question B7 | **Lister** les différentes modifications à opérer dans le code existant et exécuté par la carte de commande dans le cas où les concepteurs souhaitent introduire cette nouvelle énigme. Les modifications proposées devront être clairement identifiées et le code à ajouter sera fourni en explicitant où il s'insère.

ÉTUDE C (à traiter obligatoirement)

Adaptation de l'interface de supervision du Game Master à une nouvelle énigme.

L'objectif de cette partie est l'étude de l'interface entre le Game Master et la carte de commande d'une salle en vue d'intégrer dans cette salle un nouvel élément de décor qui sera pris en compte dans l'interface Game Master.

Les échanges entre la carte de commande et le Game Master se font au moyen d'une interface web, vue lors de l'étude B, dont la page d'accueil est affichée figure B1. Lors du démarrage de la carte de commande d'une salle, celle-ci instancie un serveur Mongoose qui génère la page affichée au Game Master et collecte les requêtes effectuées par celui-ci pour les exécuter. L'architecture générale est représentée figure C1.

Aucune connaissance préalable relative à la technologie Mongoose n'est requise pour aborder cette partie.



Figure C1 : communication entre le Game Master et la carte de commande.

Un extrait du code du fichier `index.html` encodant la page web vue par le Game Master est fourni dans le DTC1.

Question C1 | **Indiquer** sur quelle machine sont stockés les fichiers `index.html` ainsi que les différents fichiers `.js` qui lui sont liés. Indiquer également sur quelle machine ils sont exécutés. **Justifier** votre réponse.

La stratégie de communication entre l'interface Game Master et la carte de commande repose sur la technologie Ajax dont la description Wikipedia a été reportée sur le DTC2.

Question C2 | **Préciser** l'intérêt d'intégrer la technologie Ajax dans la page web de l'interface Game Master. La réponse devra mentionner l'affichage des images issues des caméras IP ainsi que l'interface avec les différents mécanismes de la salle de jeu.

Le code de la fonction `ajaxCallUrl()` contenue dans le fichier `ajax.js` est fournie dans le document DTC3.

Question C3 | **Expliciter** la différence entre une requête http reposant sur la méthode GET et une reposant sur la méthode POST. **Indiquer** et **justifier** le choix qui a été effectué ici.

L'interface Game Master permet d'afficher en temps réel des images issues des caméras de la salle afin de pouvoir guider les joueurs. Les trames transmises sur le réseau ont été relevées figure C2.

Time	Source	Destination	Info
16.278246	192.168.1.163	192.168.14.5	GET /image.jpg?t=1713530511898 HTTP/1.1
16.281896	192.168.1.163	192.168.14.6	GET /image.jpg?t=1713530511899 HTTP/1.1
16.288501	192.168.1.163	192.168.14.7	GET /image.jpg?t=1713530511899 HTTP/1.1
16.290334	192.168.1.163	192.168.14.8	GET /image.jpg?t=1713530511900 HTTP/1.1
16.295409	192.168.14.3	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
16.319844	192.168.14.4	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
16.325537	192.168.14.5	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
16.329584	192.168.14.7	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
16.339806	192.168.14.8	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
16.348995	192.168.1.163	192.168.14.1	POST /command HTTP/1.1 (text/plain)
16.351893	192.168.14.6	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
16.369387	192.168.14.1	192.168.1.163	HTTP/1.1 200 OK , JSON (application/json)
17.469436	192.168.1.163	192.168.14.3	GET /image.jpg?t=1713530513003 HTTP/1.1
17.478177	192.168.1.163	192.168.14.4	GET /image.jpg?t=1713530513004 HTTP/1.1
17.480496	192.168.1.163	192.168.14.5	GET /image.jpg?t=1713530513005 HTTP/1.1
17.482976	192.168.1.163	192.168.14.6	GET /image.jpg?t=1713530513005 HTTP/1.1
17.487093	192.168.1.163	192.168.14.7	GET /image.jpg?t=1713530513006 HTTP/1.1
17.496784	192.168.1.163	192.168.14.8	GET /image.jpg?t=1713530513007 HTTP/1.1
17.498455	192.168.1.163	192.168.14.1	POST /command HTTP/1.1 (text/plain)
17.509031	192.168.14.3	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
17.523542	192.168.14.4	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
17.527341	192.168.14.5	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
17.534421	192.168.14.7	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
17.542915	192.168.14.8	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
17.553741	192.168.14.6	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
17.596674	192.168.14.1	192.168.1.163	HTTP/1.1 200 OK , JSON (application/json)
17.999169	192.168.1.163	192.168.11.15	GET /image.jpg?t=1713530513645 HTTP/1.1
18.054771	192.168.11.15	192.168.1.163	HTTP/1.0 200 OK (image/jpeg)
18.462537	192.168.1.163	192.168.14.3	GET /image.jpg?t=1713530514100 HTTP/1.1
18.473009	192.168.1.163	192.168.14.4	GET /image.jpg?t=1713530514101 HTTP/1.1
18.480516	192.168.1.163	192.168.14.5	GET /image.jpg?t=1713530514102 HTTP/1.1
18.484124	192.168.1.163	192.168.14.6	GET /image.jpg?t=1713530514102 HTTP/1.1
18.485890	192.168.1.163	192.168.14.7	GET /image.jpg?t=1713530514103 HTTP/1.1
18.490700	192.168.1.163	192.168.14.8	GET /image.jpg?t=1713530514103 HTTP/1.1

Figure C2 : trames circulant sur le réseau, relevées grâce à l'analyseur de paquets Wireshark.

Question C4 | **Préciser** le nombre de caméras installées dans la salle et la fréquence de rafraîchissement des images configurées. **Commenter** cette valeur au regard de la fonction de ces caméras et de l'exigence de ne pas surcharger le réseau local.

L'activation des gâches (fermeture de toutes les portes) se fait lors d'un clic dans l'interface Game Master sur le bouton mis en surbrillance. Ce clic déclenche l'exécution de l'instruction : `ajaxCallUrl («/command,«cmd=room_set_gates&p1=255»,null);`

Cette commande permet de transmettre l'ordre à la carte de commande d'exécuter une fonction. On a relevé les trames transmises par le terminal du Game Master en utilisant l'analyseur de paquets Wireshark dans la figure C3.

No.	Time	Source	Destination	Protocol	Length	Info
246	1.942520	192.168.1.163	192.168.11.1	HTTP	670	POST /command HTTP/1.1
286	2.508150	192.168.1.163	192.168.14.1	HTTP	676	POST /command HTTP/1.1
371	2.718277	192.168.1.163	192.168.14.1	HTTP	670	POST /command HTTP/1.1
514	3.813024	192.168.1.163	192.168.14.1	HTTP	670	POST /command HTTP/1.1
599	4.502856	192.168.1.163	192.168.14.1	HTTP	674	POST /command HTTP/1.1
▶ Frame 286: 676 bytes on wire (5408 bits), 676 bytes captured (5408 bits) on interface \Device\NPF_{1588F319-F8EF-4854-8E89-8C8F14169D58} (0:8C:8F:14:16:9D:58) on interface \Device\NPF_{1588F319-F8EF-4854-8E89-8C8F14169D58} (0:8C:8F:14:16:9D:58) ▶ Ethernet II, Src: Intel_97:9e:6f (3c:21:9c:97:9e:6f), Dst: RaspberryPiF_ae:64:6d (b8:27:eb:ae:64:6d) ▶ Internet Protocol Version 4, Src: 192.168.1.163, Dst: 192.168.14.1 ▶ Transmission Control Protocol, Src Port: 51595, Dst Port: 8000, Seq: 1233, Ack: 15041, Len: 622 ▶ Hypertext Transfer Protocol Line-based text data: text/plain (1 lines) cmd=room set gates&p1=255						

Cette trame est suivie d'une réponse de la carte de commande :

No.	Time	Source	Destination	Protocol	Length	Info
288	2.559316	192.168.14.1	192.168.1.163	TCP	60	8000 → 51595 [ACK] Seq=15041 Ack=1855
▶ Frame 288: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{1588F319-F8EF-4854-8E89-8C8F14169D58} (0:8C:8F:14:16:9D:58) on interface \Device\NPF_{1588F319-F8EF-4854-8E89-8C8F14169D58} (0:8C:8F:14:16:9D:58) ▶ Ethernet II, Src: RaspberryPiF_ae:64:6d (b8:27:eb:ae:64:6d), Dst: Intel_97:9e:6f (3c:21:9c:97:9e:6f) ▶ Internet Protocol Version 4, Src: 192.168.14.1, Dst: 192.168.1.163 ▶ Transmission Control Protocol, Src Port: 8000, Dst Port: 51595, Seq: 15041, Ack: 1855, Len: 0						

Figure C3 : lecture de trames transmises lors d'un clic du Game Master sur le bouton activer de l'interface.

Question C5 | Indiquer les adresse MAC et IP du PC utilisé par le Game Master et de la carte de commande de la salle.

Les requêtes sont reçues par le serveur Mongoose et collectées dans une structure CGI (Common Gateway Interface) qui permet d'interfacer les données reçues par le serveur web avec les fonctions permettant d'actionner les systèmes contenus dans la pièce où se trouvent les joueurs. Le code C permettant de définir cette structure est donné figure C4.

```
typedef struct
{
    char name[256]; // Nom de la commande CGI, descripteur libre si le nom est vide
    int (*func)(int nbArg, char* args[], char* buffer, u32 clientIp); // Pointeur vers la fonction qui gère cette commande CGI
}
CgiCommand;
```

Figure C4 : structure permettant de stocker les différentes commandes cgi

Le nom d'une commande cgi est stocké dans l'attribut `name` et la fonction correspondante est stockée dans l'attribut `func`.

Le code permettant d'initialiser cette structure est le suivant :

```
#define CGI_MAX_COMMAND 64
static CgiCommand cgiCommand[CGI_MAX_COMMAND];
```

Question C6 | Rappeler la signification de l'instruction `#define` ainsi que la dénomination de ce type d'instruction en langage C.

Question C7 | Indiquer le nombre de commandes CGI différentes qui peuvent être enregistrées dans la variable `cgiCommand`. Préciser quelles sont les restrictions portant sur les noms de ces commandes.

On fournit la fonction `_cgiCommandRegister()` figure C5 qui permet d'actualiser la variable `cgiCommand` en associant à un nom de commande (`commandName`) une fonction (`cgiFunc()`) qui exécutera l'action commandée par le Game Master. Dans la suite on n'analysera pas en détail les arguments et le contenu des fonctions `cgiFunc`.

```
U8 _cgiCommandRegister ( char* commandName, int (*cgiFunc)(int nbArg, char* args[],
char* buffer, u32 clientIp) )
{
    int i ;
    // On scanne pour un emplacement de libre et on enregistre la commande CGI et son
    // pointeur de fonction
    // La fonction strcpy(s1,s2) permet de recopier la chaîne de caractères s1 dans s2.
    for ( i=0 ; i<CGI_MAX_COMMAND ; i++ )
    { if ( cgiCommand[i].name[0] == '\0' )
        { strcpy( cgiCommand[i].name, commandName );
          cgiCommand[i].func = cgiFunc ;
          debug( « Enregistrement de la commande CGI '%s'\n », commandName) ;
          return TRUE;
        }
    }
    return FALSE ;
}
```

Figure C5 : fonction effectuant l'enregistrement d'une nouvelle commande cgi dans la structure de stockage.

Un exemple de fonction stockée dans la variable `cgiCommand` est la fonction `roomCgiSetGates` (non détaillée dans le sujet). Une telle fonction peut être stockée grâce à l'instruction :

```
_cgiCommandRegister( "room_set_gates", roomCgiSetGates );
```

La fonction `roomCgiSetGates` sera exécutée lors de la réception de la requête ajax de la figure C3.

L'objectif est de compléter le code permettant d'instancier le serveur Mongoose et ainsi établir la communication entre le Game Master et la carte de commande Raspberry Pi d'une salle de l'Escape Game.

Question C8 | Écrire le code d'une procédure `void _cgiInit(void)` qui initialise le champ `name` de tous les éléments de la variable `cgiCommand` à la valeur par défaut (`'\0'`).

Question C9 | Compléter le code de la fonction `_cgiExecute()` donné dans le document réponse DRC1 permettant d'exécuter un ordre reçu du Game Master. Le nom de la fonction cgi à utiliser est stocké dans `args[0]`.

Par ailleurs, on souhaite faire évoluer l'interface Game Master lors de l'insertion d'un nouvel élément de décor : on s'intéresse au cas d'un écran qui s'allume lorsqu'une énigme est validée ou lorsque le Game Master force sa mise en fonctionnement.

Question C10 | **Proposer** une synthèse des différentes actions à mener pour que le Game Master dispose d'un bouton permettant d'allumer cet écran.
Préciser la nature des modifications à envisager en se référant aux différents codes vus dans cette partie et **préciser** quels sont ceux qu'il est nécessaire de modifier.

```
<!doctype html>
<html>
<head>
<title></title>
<meta charset="utf-8" />
<link rel="stylesheet" type="text/css" href="style.css"/>
<link rel="stylesheet" type="text/css" href="sweetalert.css"/>
<script type="text/javascript" src="room_description.js"></script>
<script type="text/javascript" src="ajax.js"></script>
<script type="text/javascript" src="camera.js"></script>
<script type="text/javascript" src="page.js"></script>
<script type="text/javascript" src="page_maintenance.js"></script>
<script type="text/javascript" src="page_preparation.js"></script>
<script type="text/javascript" src="page_game.js"></script>
</head>
<body onload="pageInit();">
<table width=100%>
<tr>
<td style="border:0px solid blue;">
<div>
<table width=100% border=0px>
<tr align=center>
<td>
<button class="menu_button"
onclick="pageChangeRoomMode('standby');">ARRET</button>
<button class="menu_button"
onclick="pageChangeRoomMode('maintenance');">MAINTENANCE</button>
<br/>
<button class="menu_button"
onclick="pageChangeRoomMode('preparation');">PREPARATION</button>
<button class="menu_button" onclick="pageChangeRoomMode('game');">JEU</button>
</tr>
</table>
</div>
<hr width="95%" align="center">
<div id="pageContent">
</div>
</td>
</tr>
</table>
</body>
</html>
```

DOCUMENT TECHNIQUE DTC2

Technologie AJAX

AJAX est une méthode utilisant différentes technologies ajoutées aux navigateurs web entre 1995 et 2005, et dont la particularité est de permettre d'effectuer des requêtes au serveur web et, en conséquence, de modifier partiellement la page web affichée sur le poste client sans avoir à afficher une nouvelle page complète. Cette architecture informatique permet de construire des applications web et des sites web dynamiques interactifs. AJAX est l'acronyme d'asynchronous JavaScript and XML : JavaScript et XML asynchrones.

La méthode AJAX consiste à utiliser de manière conjointe diverses technologies normalisées ouvertes et disponibles sur la plupart des navigateurs du marché :

- JavaScript, un langage de programmation incorporé dans les navigateurs. Les programmes écrits dans ce langage sont exécutés par le navigateur ; il est utilisé en particulier pour exploiter le XMLHttpRequest et le DOM. C'est la clé de voûte de l'AJAX.
- Le DOM (sigle de « Document Object Model ») c'est une collection d'objets, utilisée pour l'affichage dynamique et l'interaction avec les données, où chaque objet représente un élément structurel ou visuel d'une page web ou d'un document XML. Il est utilisé à partir d'un langage de programmation orientée objet tel que JavaScript pour inspecter et modifier le contenu des pages web.
- XMLHttpRequest est un objet de programmation, utilisé dans les programmes en langage JavaScript pour assurer la communication entre le navigateur et un serveur web. Il est utilisé pour la communication asynchrone : envoyer les requêtes vers le serveur et déclencher des opérations lors de la réception de réponses de celui-ci.

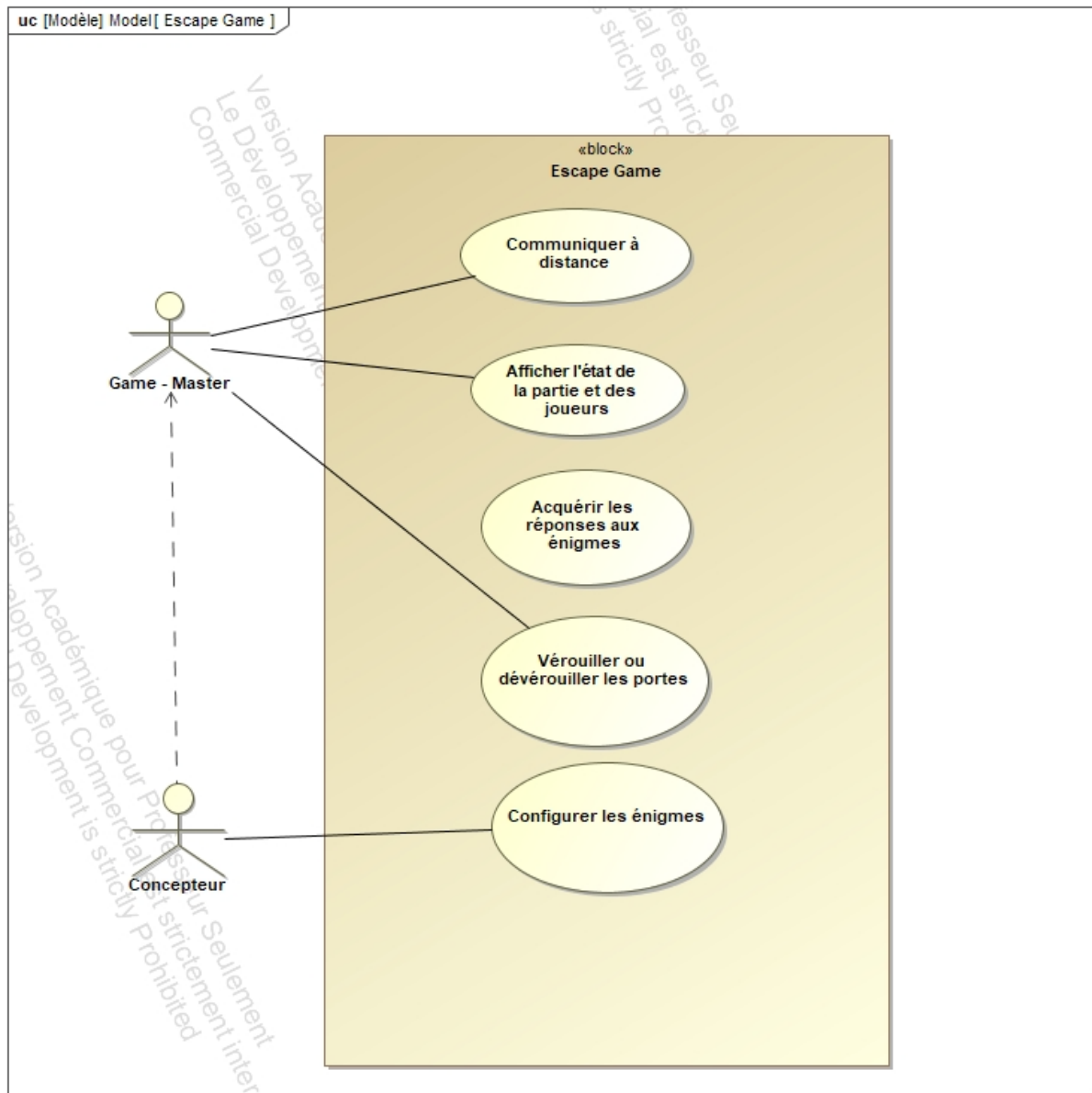
XML (sigle de anglais : Extensible Markup Language) est un langage de balisage et JSON (sigle de JavaScript Object Notation) est un format de données inspirée de la syntaxe du langage JavaScript. Ils sont utilisés pour structurer les informations envoyées par le serveur web. Le format XML est accompagné de XSLT pour sa manipulation.

DOCUMENT TECHNIQUE DTC3

Fonction transmettant la requête du Game Master au serveur

```
function ajaxCallUrl ( url, params, callback /*, ... */ )
{
    // url    : url à appeler pour le chargement des données
    // params  : paramètres http éventuels à passer au serveur
    // callback : fonction à appeler quand la réponse du serveur est arrivée
    // ... : paramètres éventuels à faire passer à la callback

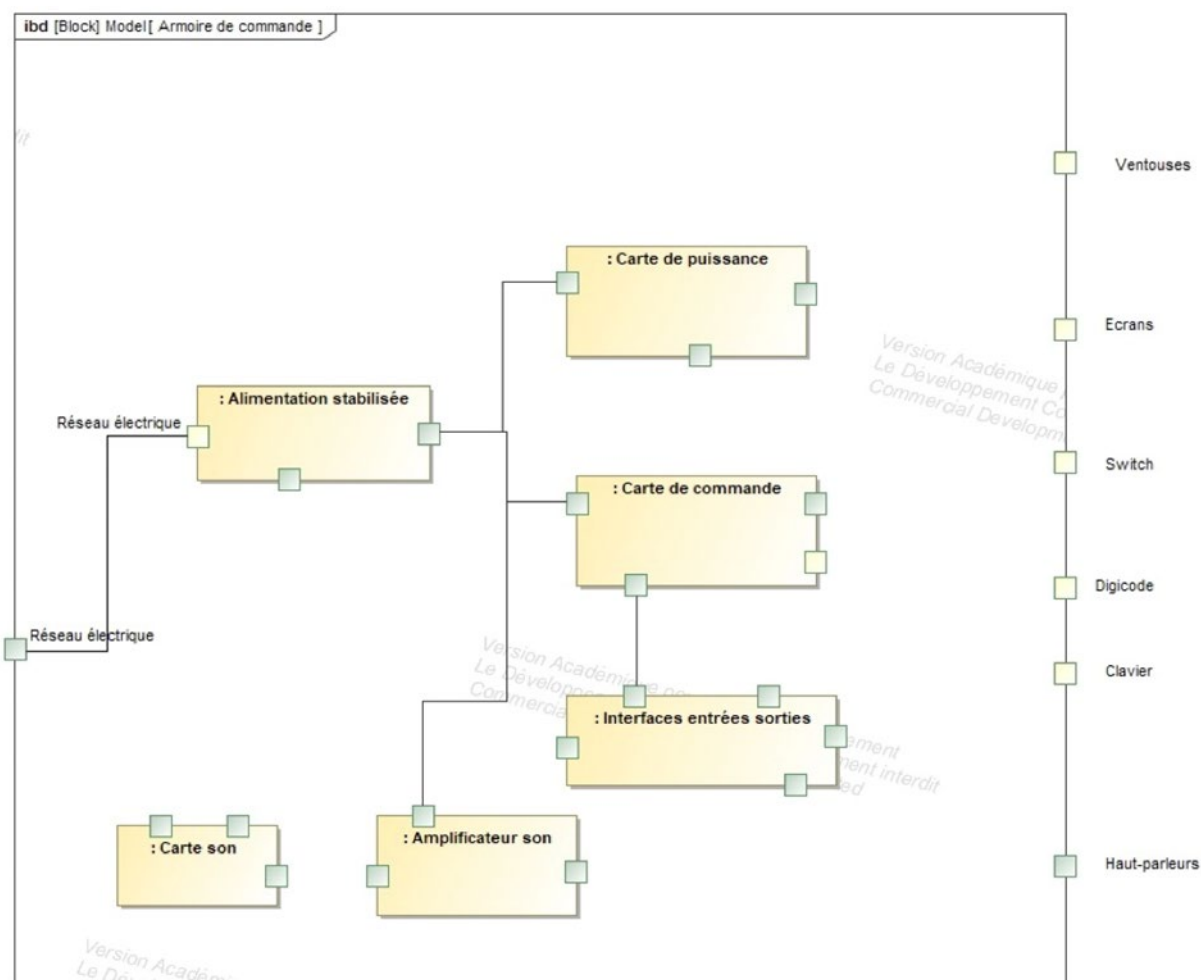
    var xhr = new XMLHttpRequest();
    xhr.callback = callback;
    xhr.arguments = Array.prototype.slice.call( arguments, 3);
    xhr.onload = ajaxSuccess;
    xhr.onerror = ajaxError;
    xhr.open( "POST", url, true );
    xhr.send( params ); }
```



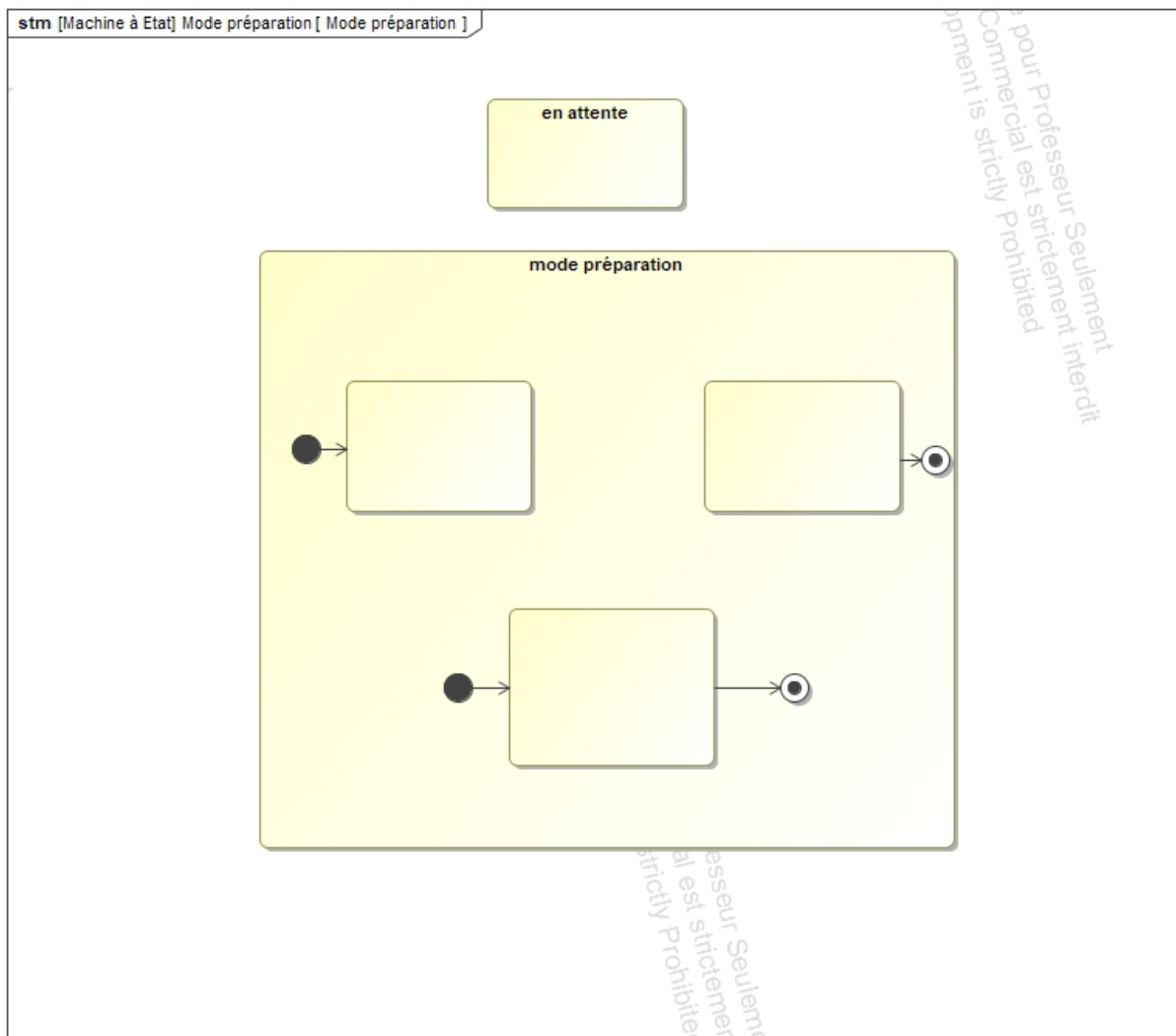
DOCUMENT RÉPONSE DRA2 (Question A3)

salle de l'Escape Game	dispositifs d'acquisition des		
	Saisies des joueurs		
	dispositifs recevant des informations de l'armoire de commande		
			
			
		Acquisition de la vidéo des joueurs	
	armoire de commande	gestion	
		du son	
		Alimentation stabilisée	
			
		Gestion des interfaces avec les joueurs	
			

DOCUMENT RÉPONSE DRA3 (Question A7)



DOCUMENT RÉPONSE DRB1 (Question B1)



DOCUMENT RÉPONSE DRC1 (Question C9)

```
int _cgiExecute ( int nbArg, char* args[], char* buffer, u32 clientIp
)
{
    int resultat; // permet de stocker le résultat renvoyé par les
fonctions cgi
    int i;
    // Les arguments en entrée de _cgiExecute sont à passer directement
en argument de la fonction cgi Souhaitée.
    // Le nom (name) de la commande cgi à exécuter est situé dans args[0]
    // On rappelle que la fonction int strcmp(s1,s2) renvoie 0 si deux
chaînes de caractère sont égales et une valeur non nulle sinon.

    // Votre code :
```

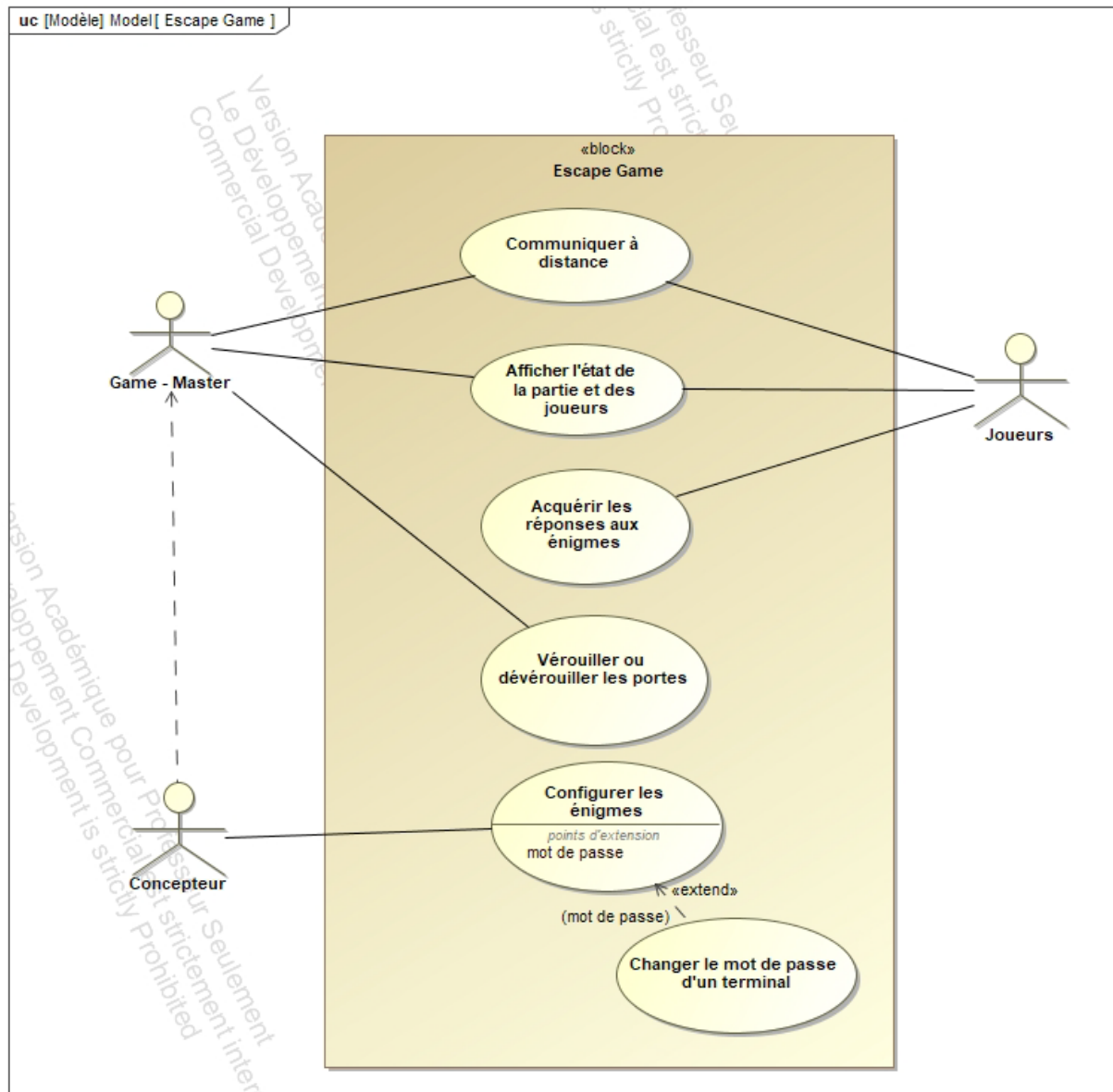
2) Attendus de l'épreuve (éléments de corrigé)

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 numérotter 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.

Question A1 et A2 (DRA1)



Question A3 : (DRA2)

salle de l'Escape Game	dispositifs d'acquisition des Saisies des joueurs	Clavier	
		Digicode	
	dispositifs recevant des informations de l'armoire de commande	Haut-parleur	
		Ventouse	
		Écran	
		Acquisition de la vidéo des joueurs	Caméra IP
	armoire de commande	gestion du son	Carte son Amplificateur son
		Alimentation stabilisée	
		Carte de commande	
		Gestion des interfaces	Interface entrée- sortie
		avec les joueurs	Carte de puissance

Question A4 :

Réponse : Un switch est un système qui permet de connecter différents systèmes sur un même réseau ethernet.

Question A5 :

Réponse : <http://192.168.17.1:8000>

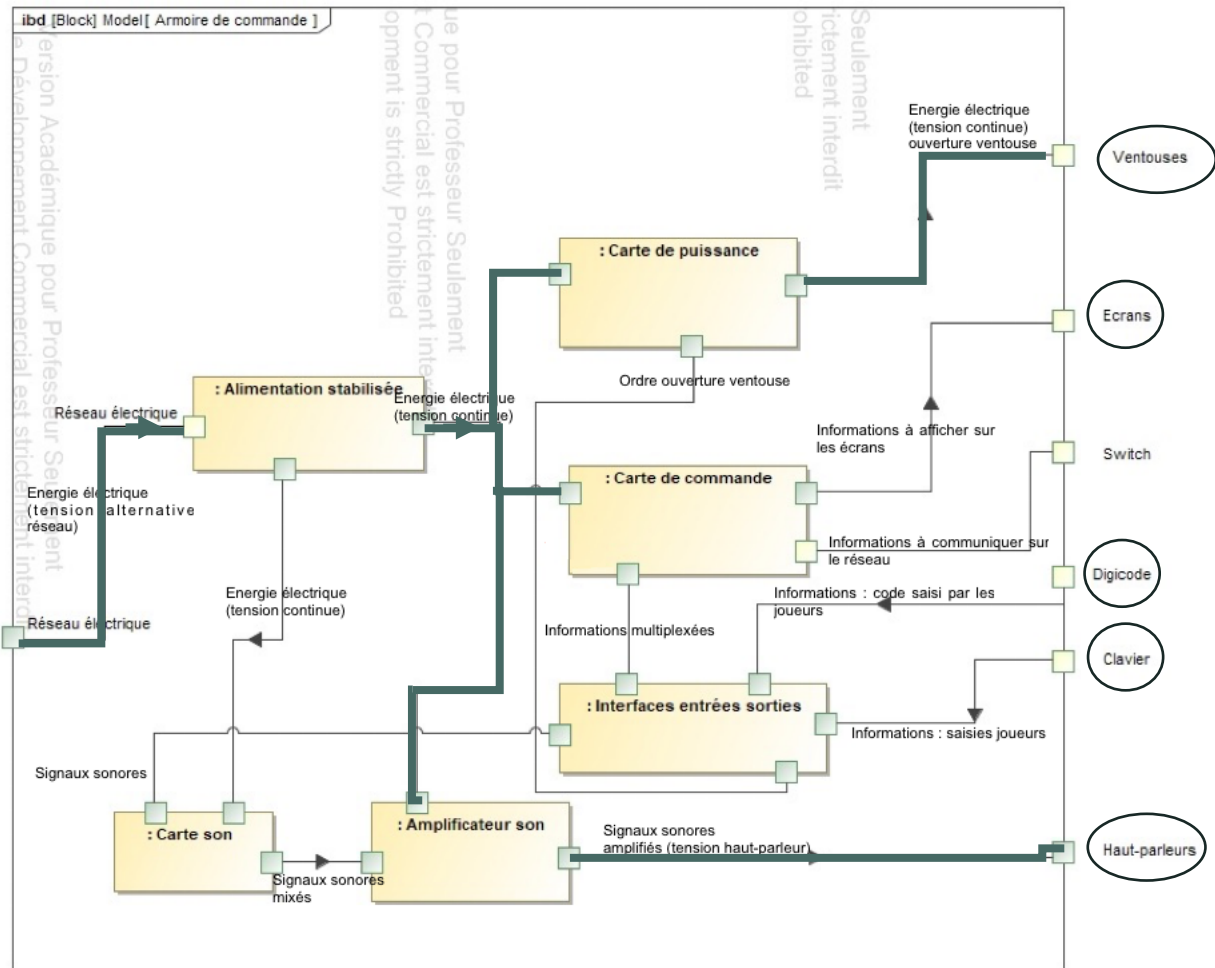
Question A6 :

Réponse : L'IP est fixée par la configuration de la caméra et n'est pas attribuée par un serveur DHCP. Ce paramétrage permet de configurer directement les IPs des caméras de chaque salle dans la carte de commande qui lui est associée afin que celle-ci ne collecte que les images provenant des caméras de sa salle.

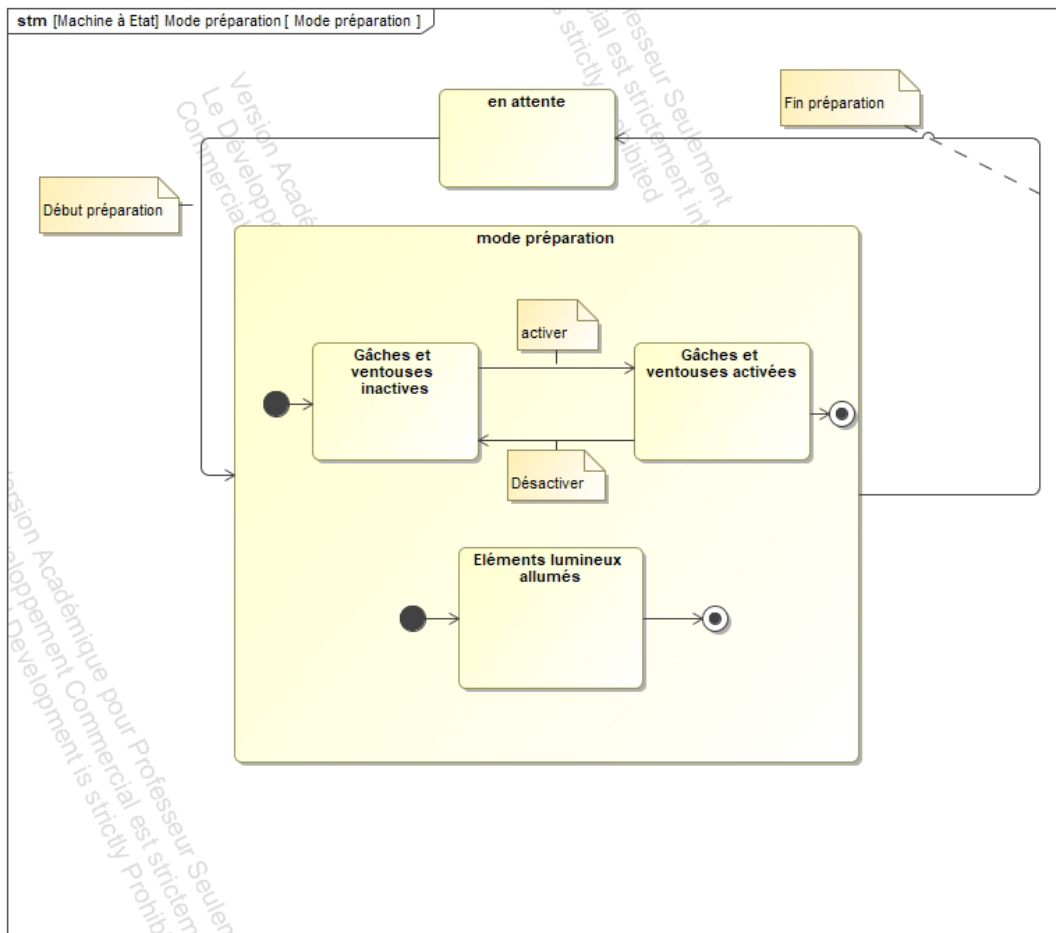
Question de conception préliminaire (A7, A8, A9) :

L'expérience ludique est enrichie en incorporant une carte de commande qui va permettre de gérer différentes tâches :

- L'animation de décors riches et immersifs (écrans / effets sonores)
- L'acquisition de données des joueurs permettant de diversifier les supports de jeu et renouveler l'expérience ludique (digicodes, écrans tactiles, capteurs)
- Créer un lien avec un Game Master afin d'éviter aux joueurs de rester bloqués et leur permettre d'interagir avec une aide extérieure. (caméras IP + hauts-parleurs)



Question B1 :



Question B2 :

Réponse : Le multithreading consiste à générer plusieurs processus qui évoluent en même temps du point de vue de l'utilisateur. C'est le système d'exploitation qui alloue du temps d'exécution aux différents threads en alternant leur activation. Dans certains cas, on peut également faire du parallélisme pur (en divisant une tâche à faire sur différents cœur du processeur) ou gérer le parallélisme apparent 'à la main', en écrivant un code qui appelle alternativement les fonctions gérant les différentes tâches du moment que ces fonctions terminent en un temps relativement court pour que l'utilisateur ne perçoivent pas cette alternance.

Question B3 :

Réponse : Premier cas : `if ( _GameElapsedTime()-save_time>180 && verrou = 0 );`

Deuxième cas : `if(_enigme_2() == 1 || _deblocage_GM_2() == 1)`

Question B4 :

Réponse :

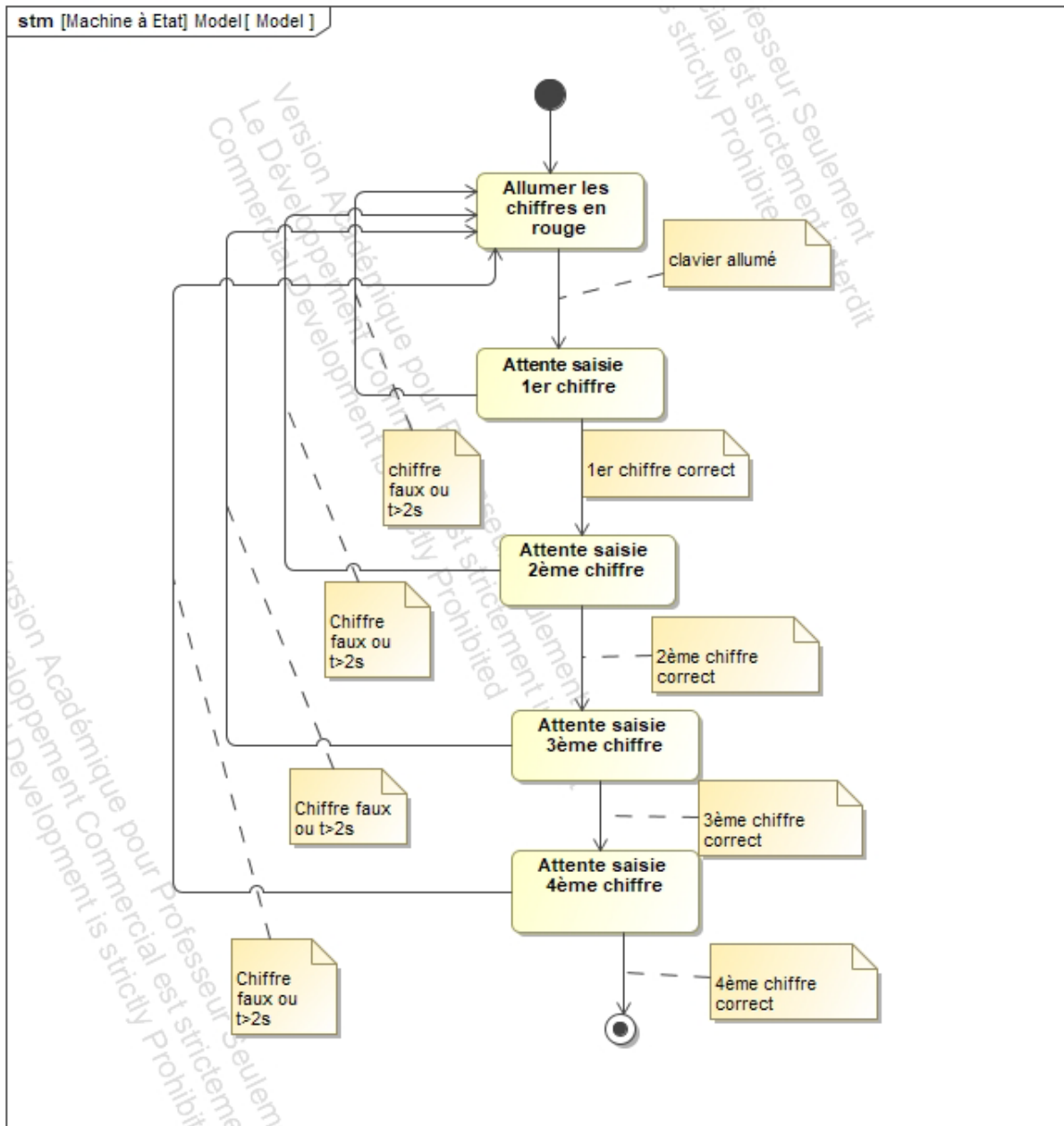
case 4 :

```
if( (_enigme_4() == 1 || _deblocage_GM_4() == 1) || (_enigme_5() == 1 || _deblocage_GM_5() == 1) )  
{ cyc ++ }
```

case 5 :

```
if( (_enigme_4() == 1 || _deblocage_GM_4() == 1) && (_enigme_5() == 1 || _deblocage_GM_5() == 1) )  
{ cyc ++ }
```

Questions de conception préliminaire(B5, B6, B7) :



L'introduction de cette nouvelle énigme se fait en

- Programmant les fonctions `_enigme_i` et `_deblocage_GM_i` associées à cette énigme. La fonction `_enigme_i` est le code vérifiant que l'énigme a bien été résolue et revient à implémenter le diagramme d'état ci-dessus. La fonction `_deblocage_GM_i` vérifiera les instructions fournies par le GM et transmises depuis son interface de commande. Conformément au code étudié aux questions 2.3 et 2.4, ces fonctions ne prennent pas de variable en entrée (la lecture des périphériques utilisés par les joueurs se fait donc par une procédure appelée depuis la fonction) et renvoient un entier qui vaut 1 si l'énigme est validée.
- Ajoutant un état dans la fonction `salle_machine_etat`. Ainsi si on insère une énigme numéro I à la fin de celles présentes, le code ajouté sera :

case i :

```

if (_enigme_i() == 1 || _deblocage_GM_i() == 1)
    {cyc++;}
break;

```

Question C1 :

Réponse : les fichiers sont sur la carte RaspBerry et sont exécutés depuis un navigateur sur le poste du Game Master.

Question C2 :

Réponse : Permet de mettre à jour uniquement les éléments modifiés ce qui permet de « gagner du temps » dans le rafraîchissement de la page. Ce sera utile pour les images et la mise à jour de celles-ci.

Question C3 :

Réponse : les données envoyées sont visibles sur le réseau avec la méthode Get et pas avec la méthode Post. La méthode utilisée ici est la méthode post mais il n'y a pas de données sensibles.

Question C4 :

Réponse : On voit que les ip sources de 192.168.14.3 à 192.168.14.8 remontent des images à la carte de commande il y a donc 6 caméras. La durée entre 2 images transmises vaut $17,6 - 16,4 = 1,2$ secondes soit une fréquence de rafraîchissement d'un peu moins d'1 Hz (0,8Hz) (ce qui est suffisant pour suivre les actions des joueurs sans saturer le réseau avec des flux vidéos potentiellement lourds).

Question C5 :

Réponse : La commande cmd est envoyée de la source (PC game master) vers la carte de commande de la salle, on voit donc que la carte de commande de la salle est une RaspberryPi d'ip 192.168.14.1 et d'adresse MAC b8:27:eb:ae:64:6d. Le PC du Game Master a donc pour ip 192.168.1.163 et pour adresse MAC 3c:21:9c:97:9e:6f.

Question C6 :

Réponse : #define est une directive de préprocesseur qui permet de construire un alias entre le nom de macro CGI_MAX_COMMAND et la valeur 64. Ainsi CGI_MAX_COMMAND ne sera pas une variable avec une occupation mémoire et une portée qu'il faut contrôler mais un raccourci sur une constante qu'on souhaitera réutiliser éventuellement.

Question C7 :

Réponse : Chaque commande cgi est caractérisé par une structure ayant pour attribut un nom codé sur un tableau de 256 octets (donc 256 lettres codées en char sur un octet) et un pointeur sur une fonction. Les différentes commandes cgi sont stockées dans un tableau de taille 64 qui permet de stocker 64 commandes différentes.

Questions de conception préliminaire (C8, C9, C10) :

```
void _cgiInit ( void )
{
    int i;
    for ( i=0 ; i<CGI_MAX_COMMAND ; i++ )
    { cgiCommand[i].name[0] = '\0';
    }
}

// Correction du code attendu :
for ( i=0 ; i<CGI_MAX_COMMAND; i++ )
{ if ( strcmp( cgiCommand[i].name, args[0] )==0 && cgiCommand[i].func )
    { resultat = cgiCommand[i].func( nbArg, args, buffer, clientIp );
      break;
    }
}

return( resultat );
}
```

Les différentes modifications sont les suivantes :

- 1) Le code html de la page affichée doit contenir un nouveau bouton. L'appui sur ce bouton transfère au serveur une nouvelle commande cgi.
- 2) Cette commande cgi doit être associée à une fonction qu'il est nécessaire de coder et qui sera stockée ainsi que le nom de la commande cgi transmis par la page web dans la structure `cgi_command`. Cette fonction déterminera les actions que doit effectuer la carte de commande lors de l'appui sur le bouton. Ici, il s'agirait de mettre sous tension / hors tension un écran.

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 second degré, publié au Journal Officiel du 19 avril 2025

A. - Epreuves d'admissibilité

2° Seconde épreuve d'admissibilité.

L'épreuve consiste en une résolution de problème.

L'épreuve vise à apprécier la capacité du candidat, à partir de l'exploitation de ressources et documents techniques, à résoudre un problème technique posé et à établir des éléments de conception préliminaire.

Durée : quatre heures.

Coefficient 2.

L'épreuve est notée sur 20. Une note globale égale ou inférieure à 5 est éliminatoire.