

CONCEPTION D'UN SYSTÈME COMBINÉ RÉALITÉ AUGMENTÉE ET IMAGERIE CÉRÉBRALE NIRS

DOCUMENT DE DESIGN
(DESIGN DOCUMENT)

GBM8970-ARNIRS.003 v 4.0
2022-04-18










Nom	Matricule	Signature
Juliette Pelletier	1953775	
Achille Vigneault	1962145	
Justine Loignon-Lapointe	1955621	
Gabriel Potvin	1957331	
Kayla Rezendes	1938396	
Khadija Yakoubi	1958714	
Sandrine Bédard	1954359	
Chiara Roche	1888481	
Remidler Damour	1963236	

TABLE DES MATIÈRES

1	Design retenu.....	3
1.1	Structure globale	3
1.1.1	Paradigmes en réalité augmentée	3
1.1.2	Synchronisation des signaux	4
1.1.3	Intégration NIRS-casque AR et boîtier Arduino	4
1.2	Paradigmes en réalité augmentée	7
1.2.1	Général.....	7
1.2.2	Stroop.....	7
1.2.3	N-back.....	21
1.3	Synchronisation des signaux	29
1.3.1	Équipement et matériel utilisés	29
1.3.2	Mise en application	29
1.4	Intégration physique.....	33
1.4.1	Intégration des casques NIRS et AR	33
1.4.2	Design de la boîte Arduino	37
1.5	Guide de l'utilisateur.....	38
2	Designs alternatifs non retenus.....	38
2.1	Paradigme en réalité augmentée.....	38
2.1.1	Stroop.....	39
2.1.2	N-back.....	39
2.2	Synchronisation des signaux	39
2.2.3	Mise en application	39
2.3	Intégration des casques NIRS et AR	40
5.	Sommaire des designs.....	41

INTRODUCTION

Le mandat du projet, relatif à la conception d'un système combiné de réalité augmentée (AR) et d'imagerie cérébrale NIRS, nous a été fourni en septembre 2021 et nous avons élaboré des solutions pour y répondre. Ce document présente le design de la solution élaborée afin de répondre aux exigences fonctionnelles et aux contraintes établies dans le document de spécifications, rédigé au début du projet. La solution retenue est présentée dans l'ensemble et est suivie par une explication plus complète et détaillée des designs de chacune des composantes de la solution, principalement le design des paradigmes en réalité augmentée, de la synchronisation des signaux et de l'intégration physique des systèmes du projet. Un autre document présente une matrice de traçabilité qui lie chaque design présenté dans ce document à une fonction du projet. Le design a également été amélioré selon les résultats des tests du prototype.

1 DESIGN RETENU

1.1 Structure globale

Le projet comprend trois composantes principales : l'élaboration des paradigmes en réalité augmentée, la synchronisation des signaux entre les différents systèmes ainsi que l'intégration physique des casques AR et NIRS.

La structure globale des paradigmes en réalité augmentée est présentée à la **section 1.1.1**, pour la synchronisation des signaux, à la **section 1.1.2** et pour l'intégration physique du casque AR et du casque NIRS et le boîtier de l'Arduino, à la **section 1.1.3**.

1.1.1 Paradigmes en réalité augmentée

La structure des paradigmes en réalité augmentée aura la forme présentée à la **Figure 1-1**.

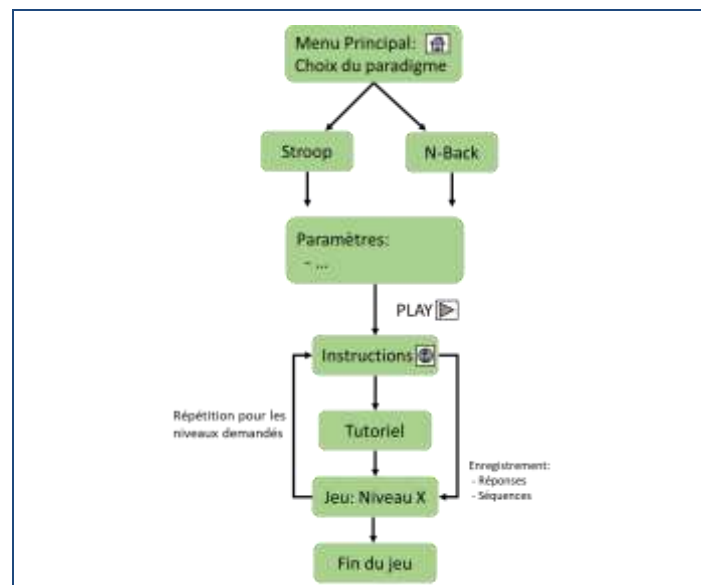


Figure 1-1 : Schéma bloc des paradigmes en réalité augmentée.

Tout d'abord, au menu principal, le chercheur choisit le paradigme psychologique désiré dans la liste déroulante, parmi les options **Stroop** et **N-back**. Par la suite, il peut déterminer la valeur des différents paramètres spécifiques au jeu choisi. Les instructions du jeu sont présentées au chercheur, qui les répète au participant. Ce dernier effectue un tutoriel du niveau du jeu et par la suite, commence l'essai. Les réponses sont enregistrées ainsi que les temps de

références et de réponses pour chaque séquence. Selon les différents niveaux demandés par le chercheur, les étapes d'instruction, du tutoriel et du jeu sont répétées. Lorsque tous les niveaux demandés sont effectués, le jeu est terminé.

1.1.2 Synchronisation des signaux

La synchronisation des signaux se fera à l'aide d'*Unity version 2018.2.5*, une plateforme permettant la conception d'expérience 3D, et *Arduino IDE version 1.8*, un logiciel permettant la programmation du microcontrôleur Arduino. Ces logiciels seront utilisés pour synchroniser plusieurs signaux provenant de divers équipements, incluant un casque de réalité augmentée *Meta 2* (Brown, s. d.), un casque NIRS *Wireless Brite MKII* (New Biotechnology Ltd, 2021), une télécommande *PortaSync* (Artinis Medical Systems, 2021), un tapis roulant *Trackmaster TMX425CP* (Full Vision inc., 2004), des boutons de réponse (The Black Box ToolKit, 2021), et des capteurs EMG *DTS Footswitches* (Noraxon U.S.A inc., 2013). La séquence de communication entre les équipements est démontrée à la **Figure 1-2**.

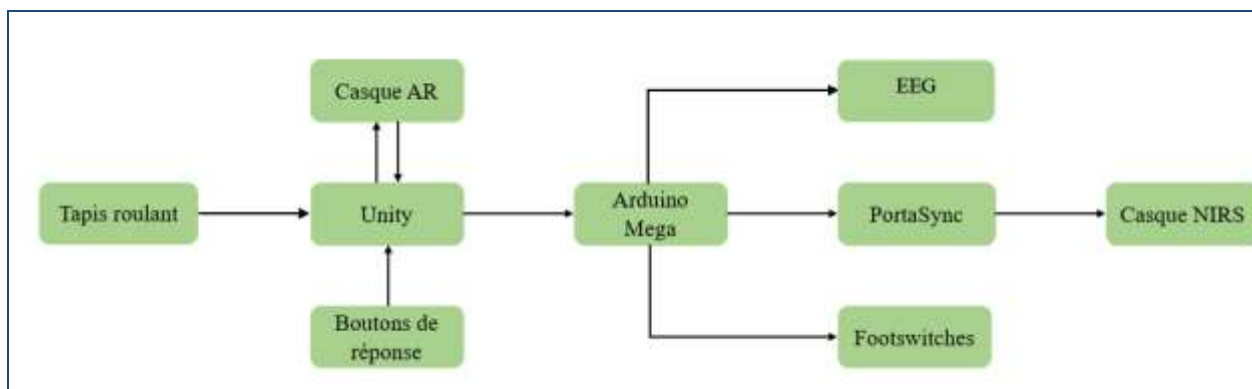


Figure 1-2 : Séquence de communication de la synchronisation

Le tapis roulant communique avec *Unity*. *Unity* communique avec le microcontrôleur *Arduino Mega 2560 R3* (Arduino, s. d.) et le casque AR. Le casque AR et les boutons de réponse communiquent avec *Unity*. Ensuite, le microcontrôleur communique avec les *Footswitches*, l'EEG et la télécommande *PortaSync*, qui peut communiquer avec le casque NIRS. Toute la programmation est en langage C#.

1.1.3 Intégration NIRS-casque AR et boîtier Arduino

Le système d'intégration physique entre le casque NIRS et le casque AR est composé d'un large appui-tête de mousse semi-rigide présenté à la **Figure 1-3**. Ce dispositif en mousse permet l'utilisation des deux appareils simultanément par sa position entre les casques NIRS et AR. Il est composé de trous correspondants à l'emplacement des capteurs du NIRS et son épaisseur est similaire à la hauteur de ceux-ci lorsqu'ils sont insérés.

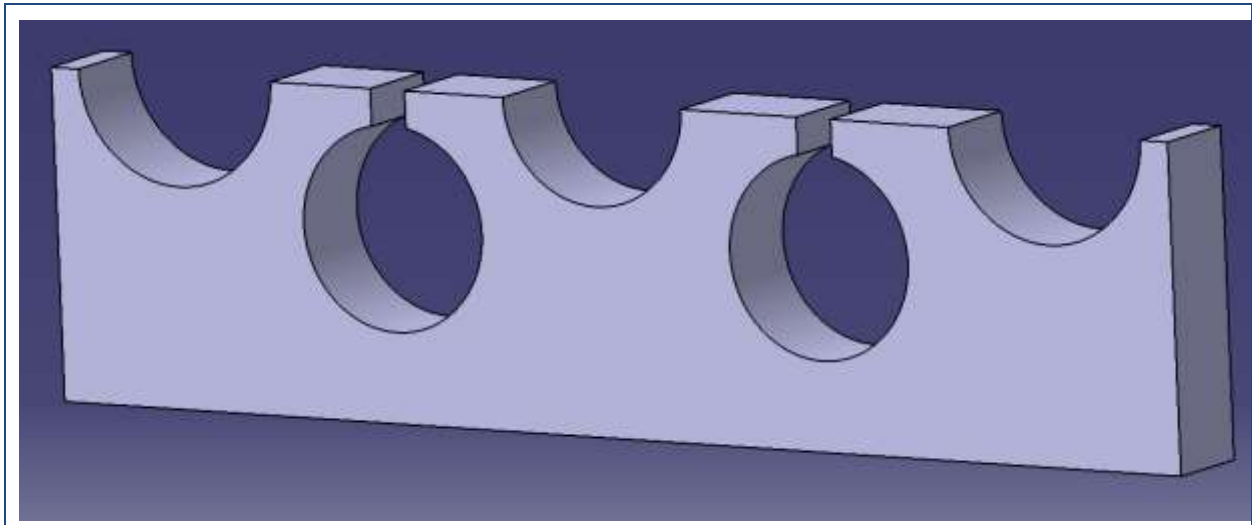


Figure 1-3 : Large appui-tête.

Le système d'intégration NIRS-casque AR comprend également deux types de petits appuis-tête en mousse semi-rigide comme présentés à la [Figure 1-4](#). Ceux-ci permettent un meilleur confort du participant lors de l'utilisation des deux casques puisqu'ils se positionnent simplement entre les deux appareils ou au niveau de n'importe quel capteur du NIRS sur les côtés de la tête de l'utilisateur.

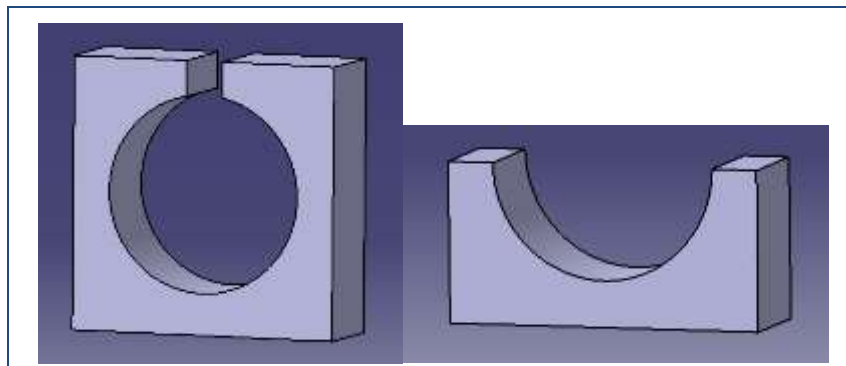


Figure 1-4 : Modèle de petit appui-tête.

Le système d'intégration est finalement composé d'une pochette destinée à accueillir le boîtier de transmission de signal du NIRS et positionnée à l'arrière du casque AR comme le montre la [Figure 1-5](#). L'intégration NIRS-casque AR nécessite, cependant, des ajustements au niveau de la hauteur du casque AR et le serrage de celui-ci afin d'optimiser le confort et le champ de vision du participant.



Figure 1-5 : Pochette de boîtier de transmission NIRS

Le boîtier de l'Arduino prend la forme d'une boîte avec un couvercle étanche comme présenté à la [Figure 1-6](#). Cette boîte a quatre sorties, un à l'avant pour la connexion à l'ordinateur et deux prises BNC mâle sur le côté pour les connexions aux *Footswitches* et *PortaSync*. Il y a une prise BNC supplémentaire sur le côté pour permettre l'ajout éventuel de l'EEG. Le boîtier a une lumière sur le dessus pour permettre un signal visuel de la connexion entre l'Arduino et l'ordinateur. De plus, le boîtier contient aussi un PCB du circuit externe de l'Arduino. Des fentes de ventilation se trouvent sur le côté du boîtier pour éviter la surchauffe des composants internes.

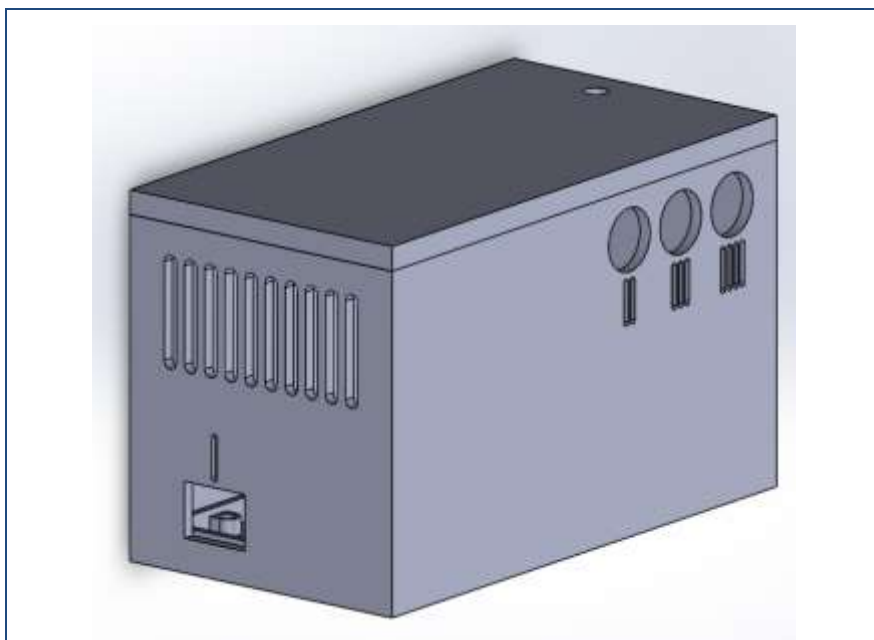


Figure 1-6 : Vue orthogonale du boîtier de l'Arduino.

1.2 Paradigmes en réalité augmentée

1.2.1 Général

Le design des paradigmes en réalité augmentée est réalisé sur *Unity* version 2018.2.5f1 (pour des raisons de compatibilité avec le casque AR Meta 2 par Meta) et le logiciel *Meta SDK 2 Beta*. Le fonctionnement de *Unity* permet la création de scènes, de pages et de différentes caméras. Une première caméra est alors assignée à l'écran d'ordinateur du chercheur, et une deuxième pour la réalité augmentée affichée au participant. Chaque scène peut contenir différentes pages. Des scripts en C# permettent de modifier et d'adapter différents éléments et variables du jeu.

Une scène est attribuée au menu principal et à chaque paradigme (**Stroop** et **N-Back**). Ces paradigmes constituent ainsi différentes scènes d'un même jeu de réalité augmentée. La première page de la scène du menu principal permet de choisir le paradigme (**Figure 1-7**). Ce modèle permet d'ajouter de futurs tests qui pourront avoir leur propre scène dans le même jeu. Il suffira donc d'ajouter les options dans la première page du menu.

La police et les couleurs de l'ensemble du jeu ont été choisies afin d'accommoder différentes déficiences visuelles et de façon à être accessibles à tous. La police choisie, *Atkinson Hyperlegible*, utilise des caractères conçus pour être différenciables des autres et offrir une plus grande lisibilité pour les lecteurs (Braille Institute, 2020). La couleur choisie pour l'interface est le bleu parce que le daltonisme bleu est le moins fréquent. En effet, il affecte seulement 0,05% de la population (We are Colorblind, 2012).

1.2.2 Stroop

Le test **Stroop** est un test d'inférence entre les couleurs et les mots. Ce test comporte quatre différents niveaux expliqués ci-dessous. Le participant donne le plus de réponses possible en 90 secondes (valeur par défaut, mais modifiable). Un élément apparaît à la fois, et lorsque le participant donne une réponse, le prochain élément s'affiche (Dr Karen Li, communication personnelle, 14 octobre). Les réponses sont enregistrées par la fonction *handtracking* du casque de réalité augmentée, qui est détaillée à la **section 1.2.2.5**. Le participant a alors dans son champ de vision trois boutons interactifs, un pour chaque couleur (*red, green, blue*). La **section 1.2.2.1** présente les différents paramètres que le chercheur peut choisir pour le paradigme. Le détail de chaque scène et page pour le design du jeu sur *Unity* est détaillé à la **section 1.2.2.2**. Puisque le jeu Stroop contient plusieurs niveaux, la séquence des scènes, des pages ainsi que les différents paramètres/variables à modifier pour ajuster à chaque niveau sont présentés à la **section 1.2.2.3**. Par la suite, les différents niveaux sont décrits à la **section 1.2.2.4**.

1.2.2.1 Paramètres :

- **Mode (*random/fixed*)** : la séquence de couleurs présentée au participant peut être aléatoire (*random*) ou définie (*fixed*) selon chaque difficulté. Si l'option *fixed* est choisie, un bouton est affiché et permet à l'utilisateur de charger un fichier .txt contenant une séquence personnalisée spécifique pour chaque difficulté. Si le fichier choisi n'est pas valide, un message d'erreur apparaît. Le format du fichier est décrit dans le manuel d'utilisation. Valeur par défaut = *Fixed*.
- **Durée de chaque niveau** : lors de chaque niveau, le participant répond au maximum de réponses possible dans la durée déterminée par le chercheur. Seuls les nombres entiers sont acceptés, une valeur inférieure à 1s est remplacée automatiquement par 1s. Valeur par défaut = 90s.
- **Nombre de niveaux** : le chercheur peut choisir le nombre de niveaux qu'il désire effectuer. Chaque niveau dans la séquence sera joué l'un après l'autre, aura la même durée (par défaut, 90s), une tâche particulière (*dual task, single cognitive task, single walking task*) et une difficulté particulière (1, 2, 3, 4). Le nombre de niveaux doit être entre 1 et 12 : une valeur inférieure à 1 sera automatiquement remplacée par 1, et une valeur supérieure à 12 par 12. Selon le nombre de niveaux choisi, le nombre adéquat de menus déroulants permettant de choisir la tâche et la difficulté pour chaque niveau s'affichent. Valeur par défaut = 1.

- **Choix de la tâche** (*dual task*, *single cognitive task*, *single walking task*) : pour chaque niveau de la séquence, le chercheur peut choisir la tâche à l'aide d'un menu déroulant. S'il choisit *dual task* ou *single cognitive task*, le même jeu de réalité augmentée sera affiché au participant lors de la tâche, mais il sera indiqué dans l'enregistrement de la séquence que le test a été jumelé à une tâche de marche pour *dual task*. Pour *single walking task*, rien ne sera affiché au participant lors de la tâche, mais un menu permettra au chercheur de démarrer le niveau et de l'arrêter pour permettre l'enregistrement de la séquence et des temps.
- **Choix des difficultés** (1, 2, 3, 4) : pour chaque niveau, le chercheur peut choisir la difficulté de la tâche à l'aide d'un menu déroulant. Les différentes difficultés sont détaillées dans la **section 1.2.2.4**. Si la tâche est de type *single walking task*, il n'est pas possible de choisir une difficulté, le menu déroulant disparaît.
- **Presets** : un bouton permet de sauvegarder les paramètres choisis dans un fichier .txt. Un autre bouton permet de charger les paramètres ayant précédemment été sauvegardés dans un fichier .txt. Si le fichier choisi n'est pas valide, un message d'erreur apparaît.

1.2.2.2 Descriptions des scènes et pages

La première scène correspond au menu principal et inclut le choix du paradigme et des différents paramètres de jeu. L'affichage se fait uniquement au niveau du chercheur, le participant n'a aucun affichage en réalité augmentée. La première page de la scène 1 est présentée à la **Figure 1-7** ; la vue du chercheur contient une liste déroulante permettant de choisir le paradigme désiré, parmi les options **Stroop** et **N-back**. Lors de la sélection du paradigme, on passe à la page 2 de la scène 1.

* Pour toutes les figures, un rectangle gris représente un bouton.

1. Scène 1 : menu principal

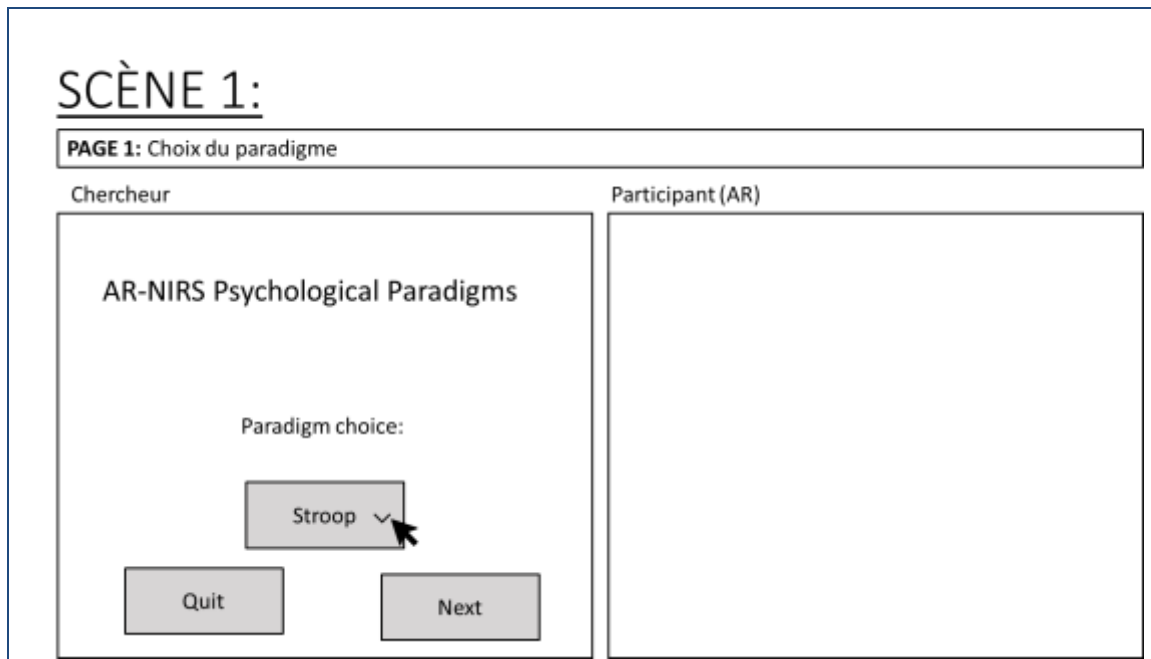


Figure 1-7 : Page 1 de la scène 1 - choix du paradigme.

La page 2 permet au chercheur de saisir l'emplacement et le nom des fichiers .txt où les résultats des tests seront inscrits (**Figure 1-8**). En appuyant sur *Choose a folder*, il est possible de sélectionner le dossier dans lequel le fichier sera sauvegardé, à l'aide de l'explorateur de fichiers. Un emplacement doit être choisi (sinon, un message d'erreur

apparaîtra), mais le nom est facultatif : un nom par défaut est donné, incluant la date et l'heure du test. Trois fichiers différents de synchronisation seront créés dans le dossier choisi : « *master* », « *test_synchro-AR* » et « *test_synchro-Arduino* ». Le chercheur indique également le numéro du port pour connecter l'Arduino comme présenté à la **section 1.3.2.2**. Le chercheur, en appuyant sur le bouton *Next*, passera à la page 3 si l'emplacement choisi est valide et que l'ordinateur réussit à se connecter à l'Arduino avec le port donné. Sinon, un message d'erreur s'affiche. S'il appuie sur le bouton *Back*, il retournera à la page 1, soit au choix des paradigmes.

SCÈNE 1:

PAGE 2: Nom du fichier (Stroop)

Chercheur

Participant (AR)

AR Stroop Study

Please input the file name and path:

ENTER FILENAME Choose a folder

Input the Arduino port:

ENTER PORT

Back Next

Figure 1-8 : Page 2 de la scène 1 - nom du fichier et port.

La page 3 (**Figure 1-9**) permet au chercheur de choisir les différents paramètres du paradigme *Stroop*. Ceux-ci sont présentés à la **section 1.2.2.1**. Lorsque l'utilisateur sélectionne *Start*, la scène *Stroop* est chargée et affichée.

SCÈNE 1:

PAGE 3: Paramètres

Chercheur

Participant (AR)

Mode Random Fixed

Time: 90 Second/level

Number of levels: Max 12

Sequence and level

1. Dual task 1

Save Parameters Load Parameters Load fixed sequence

Back Start

Figure 1-9 : Page 3 de la scène 1 - choix des paramètres.

2. Scène 2 : jeu Stroop

La scène 2 inclut le jeu *Stroop*. L’affichage s’effectue toujours sur l’écran pour le chercheur, mais désormais en réalité augmentée pour le participant. Si une erreur se produit durant le jeu, un bandeau contenant le message d’erreur est affiché sur l’écran du chercheur et peut être masqué grâce au bouton *Dismiss*.

Les pages 1 et 2 (**Figure 1-10** et **Figure 1-11**) sont affichées avant chaque niveau de la séquence. Si le niveau suivant est le premier de la séquence, soit le niveau contrôle, elle indique tout d’abord au chercheur de s’assurer que la calibration du casque de réalité virtuelle a été complétée par le participant. Si c’est le cas, le chercheur appuie sur le bouton *Display instructions*. Le niveau et les instructions du jeu sont alors affichés au chercheur, qui peut les communiquer au participant. Ces informations sont ainsi visibles avant chaque niveau. Le chercheur appuie sur le bouton *Play* afin de démarrer le niveau. La fonction `playLevel()` est donc appelée (**section 1.2.2.6.1**). Lorsque le niveau suivant n’est pas le niveau contrôle, un bouton *Tutorial* est affiché et permet de démarrer le tutoriel lié au niveau en question en appelant la fonction `playTuto()` (**section 1.2.2.6.1**). Lorsque le tutoriel est terminé, la page 1 est affichée de nouveau. Lors du démarrage d’un niveau ou d’un tutoriel, un signal sonore est émis.

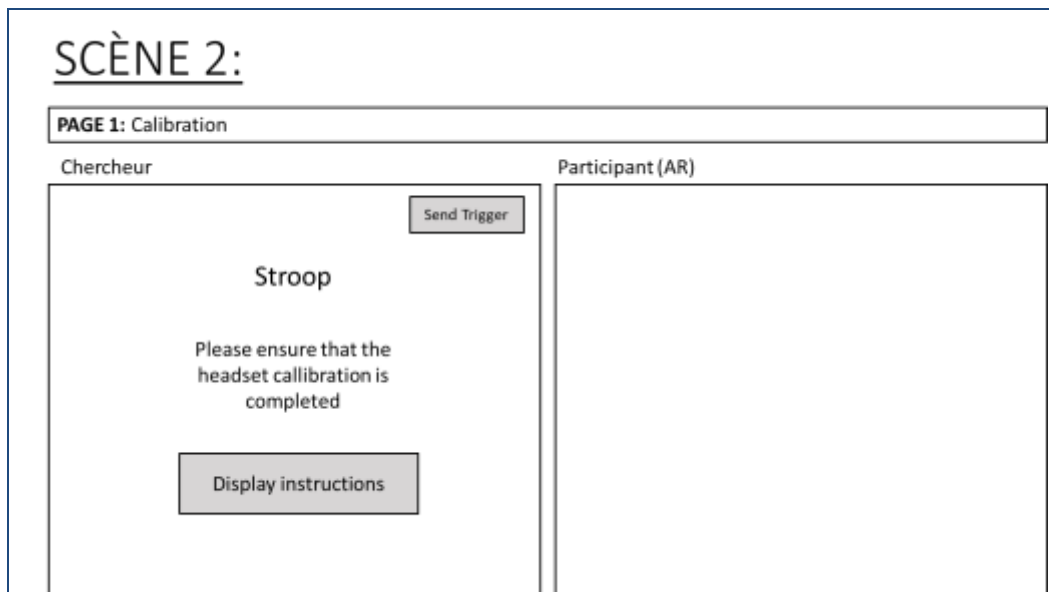


Figure 1-10 : Page 1 de la scène 2 – calibration.

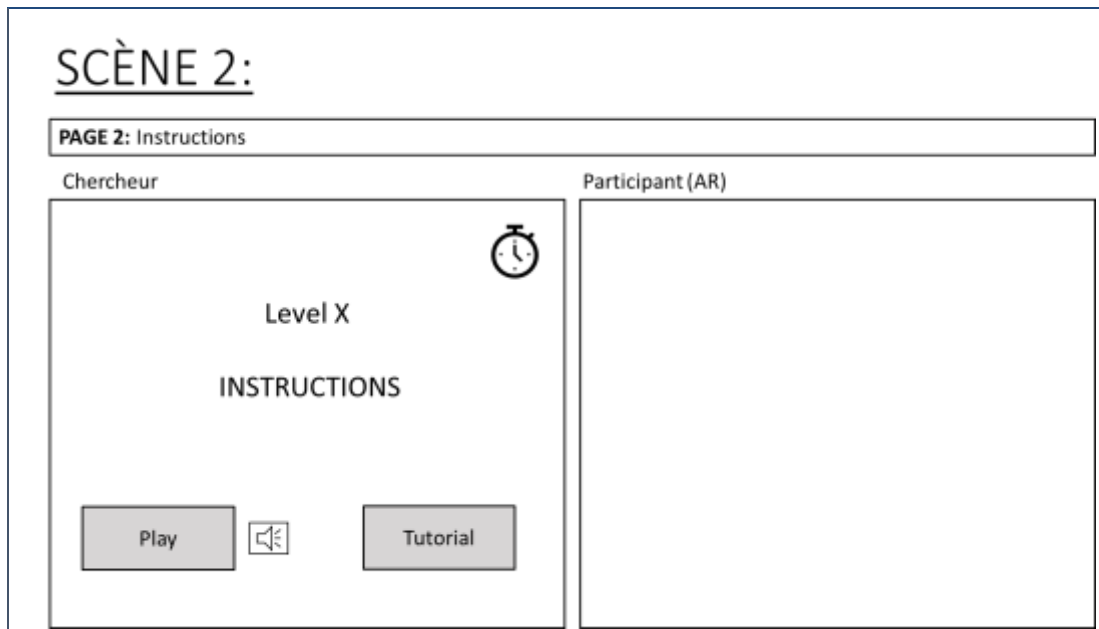


Figure 1-11 : Page 2 de la scène 2 – instructions.

La page 3

inclut le jeu *Stroop* (Figure 1-12). Selon la difficulté choisie par le chercheur, une couleur et trois boutons interactifs (*handtracking*) (*Red, Green, Blue*) sont affichés dans le champ de vue du participant. Les différents niveaux sont détaillés à la section 1.2.2.4. Le *handtracking* est effectué grâce au casque AR et permet au participant de choisir sa réponse en touchant à un bouton avec sa main (voir section 1.2.2.5). Les boutons sont à la hauteur des mains du participant et à une distance plus courte que la longueur de bras dans le champ de vue de la réalité augmentée. La question est affichée plus loin, à la hauteur des yeux. Chaque réponse est enregistrée dans un fichier et est affichée sur l'écran du chercheur au fur et à mesure que le test se déroule. La réponse attendue est également affichée. Lorsque le participant choisit une réponse, le temps de réponse est enregistré et une prochaine couleur s'affiche (déterminée au hasard ou fixée, selon le mode de jeu). Le participant effectue alors le plus de réponses possible dans le temps du niveau choisi par le chercheur. Le chercheur a également un minuteur affiché sur son écran, montrant le temps restant au niveau. Cette page est identique pour les tâches *dual task* et *single cognitive task*. Cependant, elle est différente pour la tâche *single walking task* (voir la page 5). Cette même page est aussi utilisée pour le niveau *contrôle* et pour les tutoriels, mais un nombre précis de réponses (6 pour *contrôle*, 10 pour *tutoriel*) est attendu par le participant au lieu du temps prescrit. Le niveau *contrôle* est toujours le premier niveau de la séquence. les tutoriels, mais un nombre

précis de réponses (6 pour *contrôle*, 10 pour tutoriel) est attendu par le participant au lieu du temps prescrit. Le niveau *contrôle* est toujours le premier niveau de la séquence.

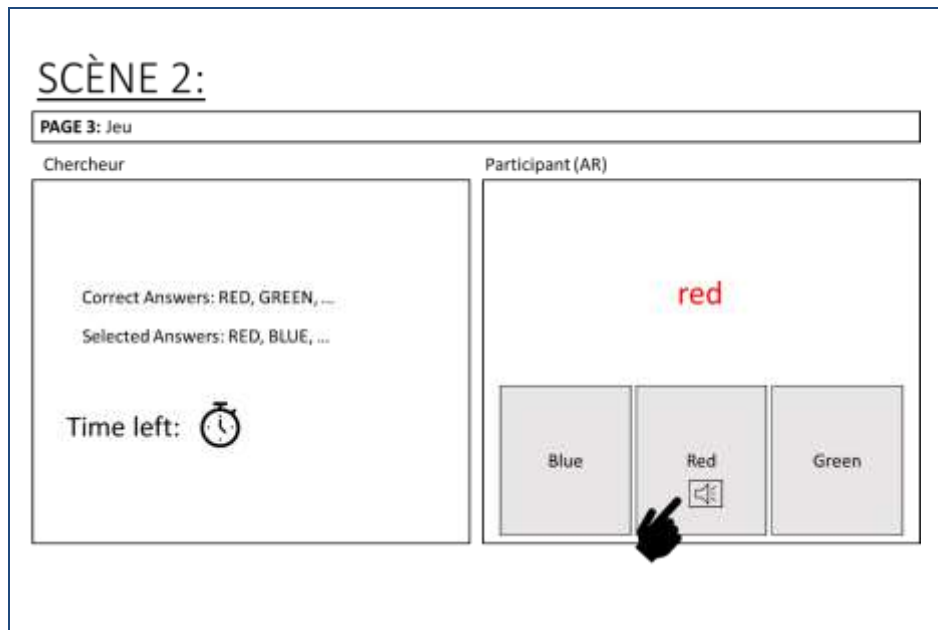


Figure 1-12 : Page 3 de la scène 2 – niveau ou tutoriel *Stroop*.

Lorsque le minuteur atteint 0s, un signal sonore est émis et la page 4 s’affiche. La page 4 présente ainsi la fin d’un niveau (Figure 1-13). Du côté du chercheur, les résultats du test s’affichent ainsi que le temps de réponse moyen. Le chercheur a l’option de passer aux instructions du prochain test de la séquence, soit les pages 1 et 2, en appuyant sur le bouton *Next level* ou bien de recommencer le niveau (*Replay level*). En cas de *Replay*, le niveau sera décrit une nouvelle fois dans le fichier .txt. Un chronomètre (*Resting Time*) affiché à l’écran du chercheur peut aussi lui permettre d’attendre une durée précise avant de débiter le prochain niveau. Un bouton (*Send Trigger*) lui offre la possibilité d’envoyer manuellement un événement à l’Arduino, par exemple avant et après le temps de repos. Rien n’est affiché sur l’écran du participant, pour lui permettre de se préparer au prochain niveau.

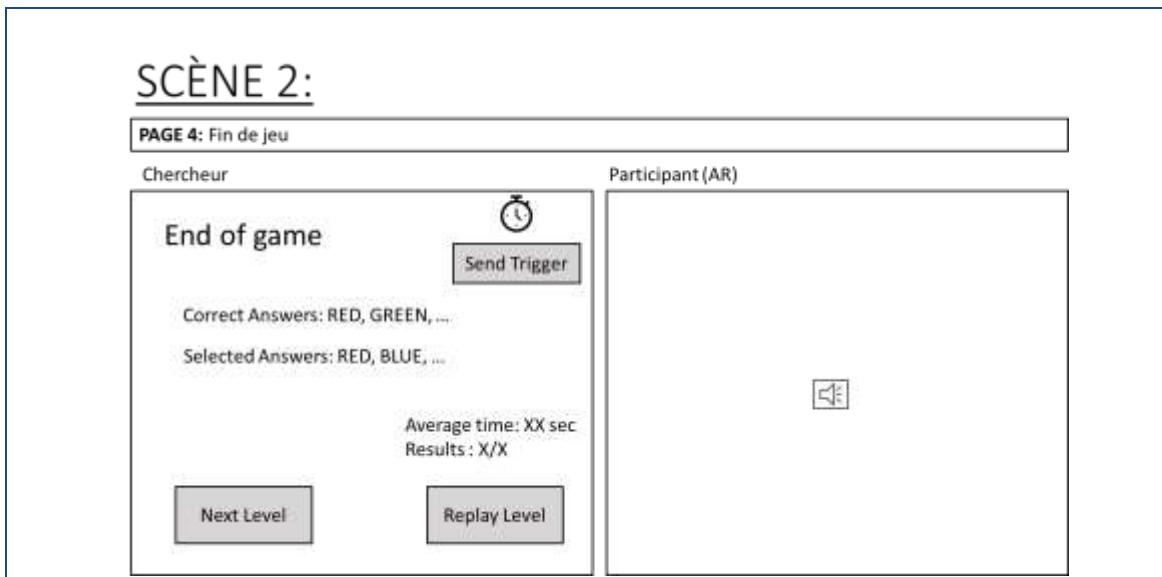


Figure 1-13 : Page 4 de la scène 2 – fin d'un niveau.

La page 5 (Figure 1-14) est affichée pour les niveaux de tâche *walking single task*. Rien n'est affiché en réalité augmentée pour le participant. Pour le chercheur, cette page est identique à la page 3, à l'exception qu'aucune réponse n'est enregistrée ou inscrite. Ainsi, seulement le temps restant au niveau s'affiche. Ce type de niveau incluant uniquement la marche, nécessite une page du jeu pour faciliter la synchronisation, afin que les temps de début et de fin de chaque niveau soient enregistrés de façon centralisée.

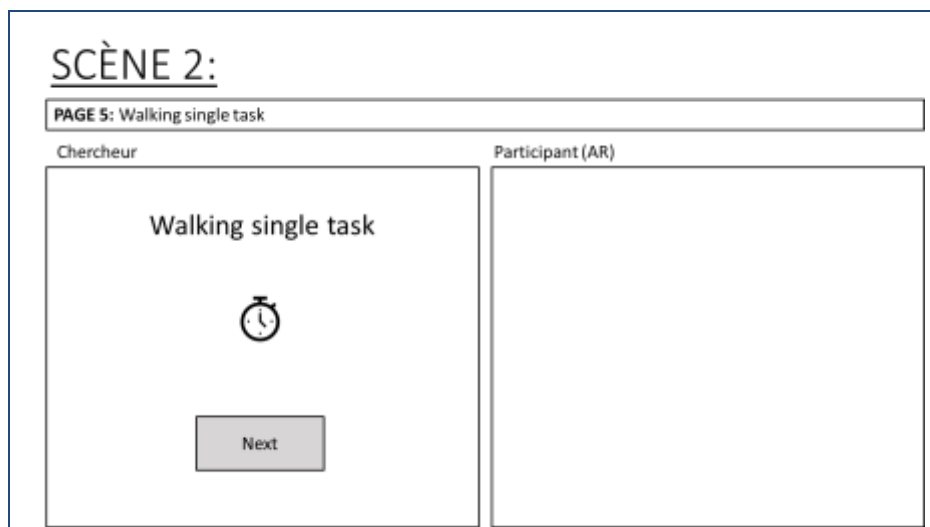


Figure 1-14 : Page 5 de la scène 2 – *walking single task*.

La page 6 (Figure 1-15) est affichée sur l'écran du chercheur lorsque tous les niveaux de la séquence choisie ont été joués. Elle présente dans un tableau résumé, chaque niveau joué (à l'exception des tutoriels), dans l'ordre, avec la tâche et la difficulté choisies, les réponses obtenues, les réponses attendues, les résultats et les temps moyens. Si un

niveau a été recommencé, seulement le dernier essai s’affiche. Les informations complètes sont disponibles dans le fichier .txt créé. Rien n’est affiché au participant.

#	Level	DWF	Correct Answers	Selected Answers	Results	Rx.time
0	Control	0	RED, GREEN, RED...	RED, RED, RED, ...	2/6	0.39 s
1	Dual Task	2
2	Dual Task	1
3	Dual Task	5
4	Dual Task	6
...

QUIT

Figure 1-15 : Page 6 de la scène 2 – fin de la séquence.

1.2.2.3 Séquence

Une séquence standard comportant seulement un niveau serait jouée dans l’ordre suivant.

1. Page 1 : Calibration
2. Page 2 : Instructions
3. Page 3 : Niveau (*contrôle*)
4. Page 4 : Fin du niveau
5. Page 2 : Instructions
6. Page 3 ou 5 : Niveau (tutoriel ou jeu, selon la tâche et la difficulté choisies)
7. Page 4 : Fin du niveau
8. Page 6 : Fin de la séquence

Afin d’accommoder aux différentes difficultés du test *Stroop*, les variables suivantes seront modifiées lorsque la difficulté change :

Tableau 1.1 : Variables des difficultés de *Stroop*.

Type	Variable	Options	Description
Liste	Couleur	[Red, Green, Blue]	Couleur de l’élément affiché
Bool	Écriture du texte	True/False	Écriture du texte ou affichage d’un carré
Bool	Tutoriel	True/False	Affichage d’un nombre prédéfini d’éléments (10)
Bool	Encadrement du texte	True/False	Encadrement en noir de l’élément

1.2.2.4 Description de chaque difficulté :

Cette section présente les niveaux 0 à 4 du test *Stroop*, soit la description du niveau et le design de l’affichage en réalité augmentée.

0. **Contrôle** : Cette difficulté ne peut pas être choisie dans le menu : un niveau de *contrôle* de difficulté 0 est ajouté automatiquement au début de chaque séquence jouée. Les couleurs sont affichées dans leur couleur respective (RED, BLUE, GREEN). L’objectif de ce niveau est de déterminer le temps moyen nécessaire au participant pour appuyer sur les boutons de réponse de chaque couleur, ceux-ci étant situés à des endroits différents. Ainsi, le biais associé à chaque localisation de bouton pourra être considéré par le chercheur. Le temps de réponse est enregistré pour chaque couleur. Chaque couleur est présentée deux fois (total de 6 éléments). La **Figure 1-15** présente ce que le participant observera en réalité augmentée.

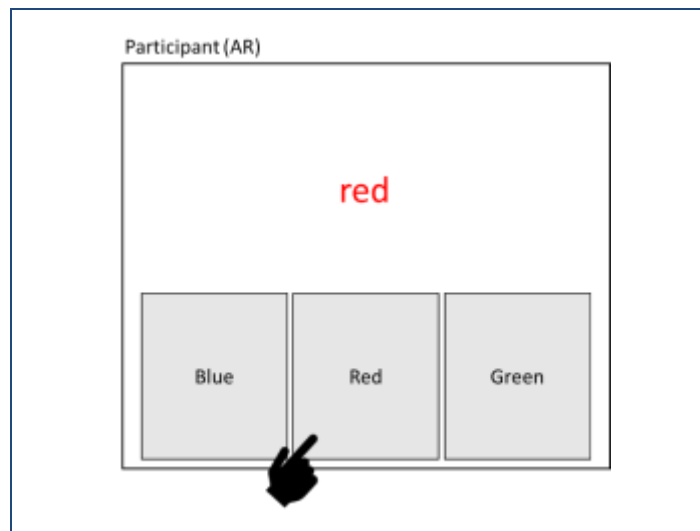


Figure 1-16 : Difficulté 0 - Contrôle négatif.

1. **Difficulté 1** : Des carrés de couleur sont affichés dans ce niveau. On a alors, par exemple :



La **Figure 1-16** présente ce que le participant observera en réalité augmentée. Le participant doit sélectionner le texte correspondant à la couleur affichée. L’instruction affichée sur l’écran du chercheur est : « Select the color of the rectangle ».

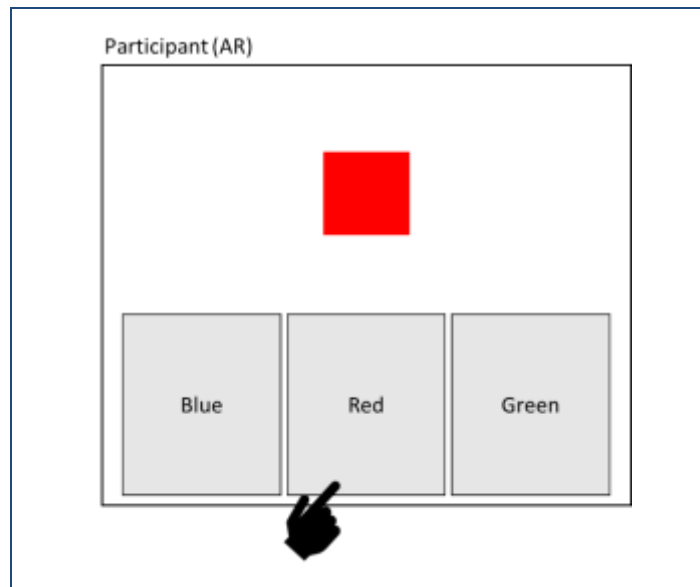


Figure 1-17 : Difficulté 1 du paradigme *Stroop*.

2. **Difficulté 2** : Les couleurs sont écrites en blanc et le participant doit sélectionner le texte correspondant au texte affiché, comme présenté à la Figure 1-17. L'instruction affichée sur l'écran du chercheur est : « Select the written color ».

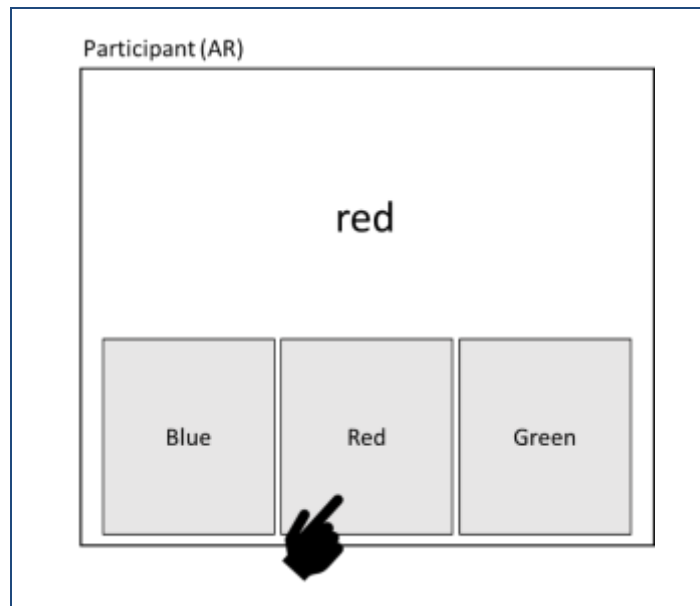


Figure 1-18 : Difficulté 2 du paradigme *Stroop*.

Difficulté

3. 3 : Dans ce niveau, la couleur de l’affichage du texte ne correspond pas au nom écrit de la couleur. Par exemple, rouge peut être écrit en vert : RED. Le participant doit sélectionner la couleur d’affichage du texte, comme présenté à la Figure 1-18. L’instruction affichée au chercheur est : « Select the color of the ink ».

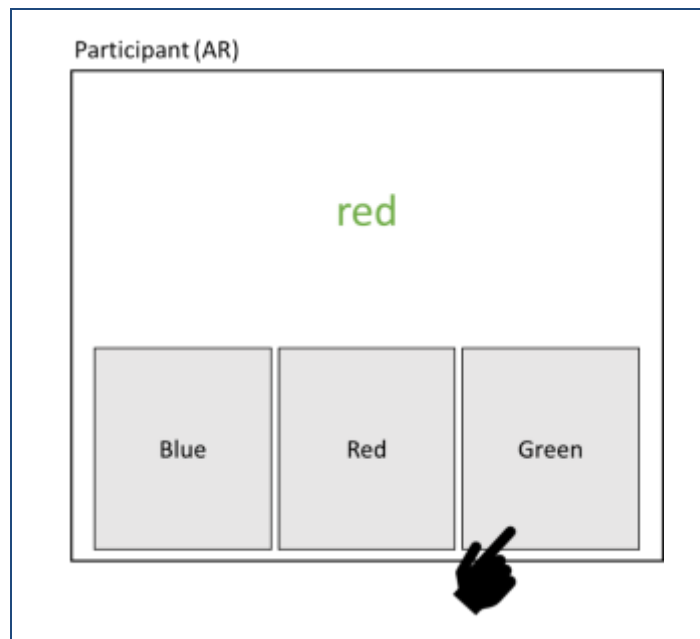


Figure 1-19 : Difficulté 3 du paradigme *Stroop*.

4. **Difficulté 4** : Comme pour la difficulté 3, la couleur de l’affichage du texte ne correspond pas au nom écrit de la couleur. De plus, la couleur est parfois également encadrée. Si la couleur est encadrée, le participant doit répondre la couleur écrite, sinon il doit répondre la couleur affichée, comme présenté à la [Figure 1-19](#). L’instruction affichée sur l’écran du chercheur est : « If the text is framed, select the written color. Otherwise, select the color of the word ».

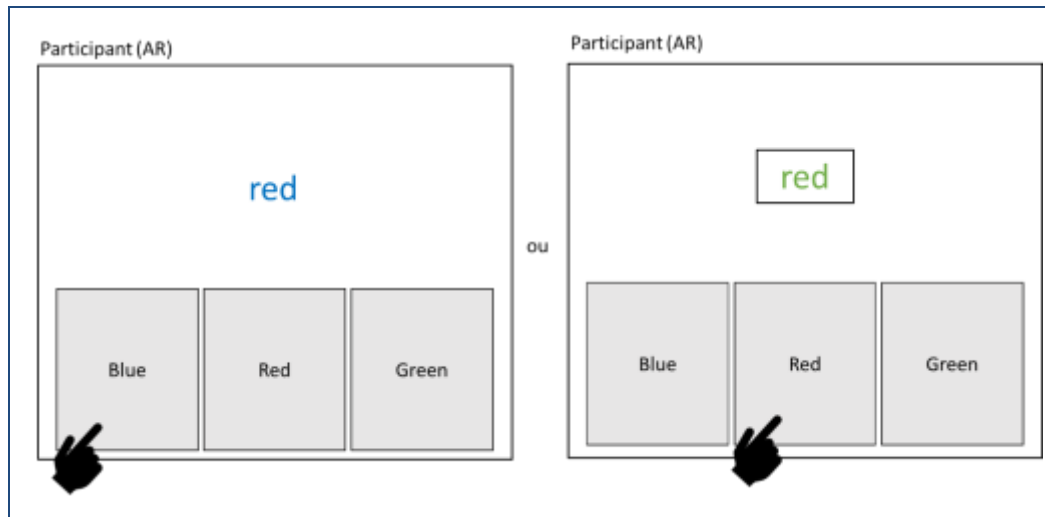


Figure 1-20 : Difficulté 4 du paradigme *Stroop*.

1.2.2.5 Enregistrement des réponses du participant :

Pour permettre au participant de répondre au test et d’enregistrer ses réponses, le suivi des mains avec les caméras du casque *Meta 2* a été choisi. En effet, cette fonction est disponible à partir du casque et des bibliothèques *Meta* dans *Unity*. En plus d’ajouter une caméra spécifique pour permettre l’utilisation du casque (*Meta Camera Rig*), on ajoute le module *Meta hands* dans *Unity* et l’on importe la bibliothèque *Meta.HandInput*. Les objets choisis pour agir comme boutons de réponse sont des cubes. La classe *Interaction* de *Meta* permet d’utiliser la fonction *Engage*, pour l’engagement avec un objet, *Disengage*, pour la fin d’une interaction et *Manipulate*, pour la modification d’objet. Dans notre cas, la fonction *Manipulate* n’est pas utilisée, puisque les objets ne doivent pas être modifiés. La fonction *Engage* reconnaît qu’une main s’est fermée sur un objet et appelle la fonction *changeText* ([section 1.2.2.6](#)) qui modifie l’affichage du jeu, notamment en changeant la couleur qui est affichée. La fonction *Disengage* ne fait que détecter que l’objet qui avait été saisi ne l’est plus.

Les caméras et la bibliothèque *Meta* permettent de repérer lorsqu’une main se situe sur un objet. Pour sélectionner la réponse souhaitée, la main sur l’objet doit se fermer, le participant doit donc effectuer un mouvement pour saisir l’objet. Lorsqu’un bouton est saisi, la réponse sélectionnée est enregistrée dans un document texte ([voir section 1.3.2.1](#)) et affichée sur l’écran du chercheur, selon la [section 1.2.2.2](#) ([Figure 1-11](#)). Dès qu’un objet est saisi, la prochaine question s’affiche, avec un délai inférieur à une seconde (voir fonction *changeText*, [section 1.2.2.6](#)).

1.2.2.6 Plan de la structure du code et principales fonctions

Un exemple du niveau 3 (voir [section 1.2.2.4](#), [Figure 1-18](#)) est illustré ci-dessous à la [Figure 1-21](#).

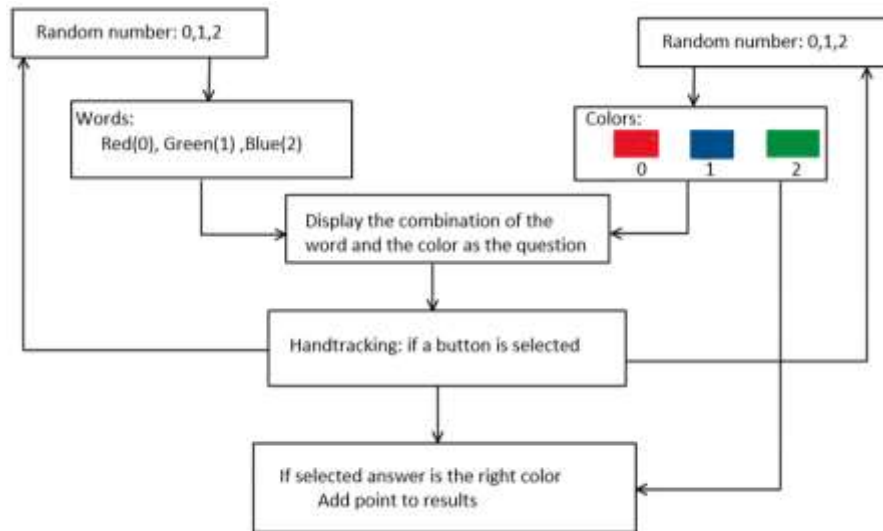


Figure 1-21 : Plan de la structure du code du niveau 3.

Les principales fonctions utilisées par chaque classe pour le paradigme *Stroop* sont décrites ci-dessous.

1.2.2.6.1 Question

La classe *Question* contient 5 fonctions principales pour le fonctionnement du paradigme *Stroop*.

```
public void playInstruction()
```

Cette fonction est appelée lorsque le chercheur appuie sur *Display Instructions* (**page 1 de la scène 2**) avant le premier niveau de la séquence ou *Next Level* (**page 4 de la scène 2**) avant chaque autre niveau. Elle affiche les instructions que doit lire le chercheur pour le prochain niveau. Un exemple de code pour le niveau 2 est comme suit :

```
textInstruction.GetComponent<TMPPro.TextMeshProUGUI>().text = "Select the written color.";
```

Si la séquence est terminée, la fonction affiche la page 6 de la scène 2, qui résume les résultats du participant pour toute la séquence.

```
public void playLevel()
```

Cette fonction est appelée lorsque le chercheur sélectionne *Play* (**page 2 de la scène 2**). Elle active les éléments visuels dans l'écran du participant, comme les boutons et l'endroit pour afficher la question. Si le mode de jeu est *fixed*, la fonction lit le fichier et stocke la séquence dans la variable *question*. Cette fonction affiche la première question aux participants en appelant la fonction de la classe *Difficulty* (voir [1.2.2.6.3](#)) qui correspond au niveau de difficulté souhaité. Cette fonction rend la variable booléenne *flagBeginTimer* = True, ce qui permet à la fonction *Update()* de démarrer le chronomètre.

```
public void playTuto()
```

Cette fonction est appelée lorsque le chercheur sélectionne *Tutorial* (**page 2 de la scène 2**). Elle est aussi appelée lors du niveau *contrôle*, soit le premier niveau de la séquence. Elle effectue exactement les mêmes étapes que *playLevel()*, mais ne démarre pas le chronomètre.

```
void Update()
```

Par définition, cette fonction est appelée à chaque seconde. Lorsque `flagBeginTimer = True`, la fonction commence à décrémenter le temps du chronomètre de 1 unité chaque seconde jusqu'à ce qu'il soit nul.

Lorsque le chronomètre est terminé, la fonction désactive le visuel du participant et affiche les résultats au chercheur.

```
public void Restart()
```

Cette fonction est appelée lorsque le chercheur appuie sur le bouton *Replay Level* (**page 4 de la scène 2**) après un niveau. La fonction réinitialise les variables nécessaires et appelle la fonction `playInstruction()`.

1.2.2.6.2 Response

La classe `Response` compte trois fonctions qui permettent au participant de jouer. Cette classe est une classe qui hérite de la classe `Interaction` de la bibliothèque `Meta`.

```
public void changeText()
```

Lorsque le participant saisit un bouton avec sa main, à l'aide du *handtracking* (**section 1.2.2.5**), on appelle la fonction `changeText`. Cette fonction enregistre la réponse du participant et compare sa réponse à celle attendue. On augmente la valeur du résultat si la réponse est adéquate. Elle appelle ensuite la bonne fonction de la classe `Difficulty` (voir **1.2.2.6.3**) pour continuer le jeu et afficher une nouvelle question au participant.

```
protected override void Engage() et protected override void Disengage()
```

Les fonctions permettent à l'utilisateur de choisir un objet et de terminer une interaction avec un objet.

1.2.2.6.3 Difficulty

Cette classe compte sept classes. Les cinq premières sont celles qui correspondent aux différentes difficultés du jeu (voir **1.2.2.4**). En effet, elles sont appelées par les fonctions `playLevel`, `playTutorial` et `changeText`.

```
public void BaseLine()
```

La fonction `Baseline` correspond à la difficulté 0 (voir **1.2.2.4**, 0). Elle permet d'afficher des noms de couleurs écrits avec de l'encre de la même couleur.

```
public void backgroundColor()
```

La fonction `backgroundColor` correspond à la difficulté 1 (voir **1.2.2.4**, 1). Elle permet d'afficher un rectangle de couleur.

```
public void blackText()
```

La fonction `blackText` affiche des noms de couleur écrits en blanc. Elle correspond à la difficulté 2 (voir **1.2.2.4**, 2).

```
public void inkColor()
```

La fonction `inkColor` affiche des noms de couleur écrits avec de l'encre d'une autre couleur. Elle correspond à la difficulté 3 (voir **1.2.2.4**, 3).

```
public void randomRectangle()
```

La fonction `randomRectangle` affiche des noms de couleur écrits avec de l'encre d'une autre couleur. De plus, elle affiche des encadrés autour de certaines questions. Elle correspond à la difficulté 4 (voir 1.2.2.4, 4).

```
int file_convert(string color)
```

Cette fonction convertit le fichier d'une séquence fixe en indice pour les quatre fonctions des différents niveaux. Elle prend en entrée le contenu du fichier *fixed* et sort un indice qui correspond à la couleur. G devient 0, R devient 1, B devient 2.

1.2.3 N-back

Un test *N-back* consiste à présenter une série d'items différents à un certain intervalle de temps. Le participant indique si l'item devant lui est le même que l'item présenté *N* items auparavant. Les réponses et temps de réponse sont enregistrés (Dr Karen Li & Rachel Downey, 2019).

Le test *N-back* en réalité augmentée avait déjà été effectué par un étudiant du client. Ainsi, pour ce paradigme, la tâche consiste à l'intégrer dans le même environnement que le paradigme *Stroop* afin que le chercheur ait plusieurs options de paradigmes, d'adapter les paramètres selon les besoins du client et d'effectuer quelques modifications visuelles afin d'uniformiser le jeu. Un menu d'accueil, le choix des paramètres, les tutoriels ainsi que différents niveaux ont déjà été réalisés en réalité augmentée. Les réponses sont enregistrées à l'aide des boutons (seront décrits dans la section 1.3.2.2), soit le bouton dans la main gauche pour *same* ou celui de la main droite pour *different*.

Principalement, l'ajout de deux options sera effectué, pour un total de trois en incluant l'option de *N-back* visuel, le *N-back* auditif, le *N-back* combiné auditif et visuel. La section 1.2.3.1 présente les différents paramètres que le chercheur peut choisir. Le menu paramètres aura la même allure que pour le *Stroop* (voir Figure 1-9). La section 1.2.3.2 présente les différentes modifications à apporter au jeu.

1.2.3.1 Paramètres

- **Mode** (*random/fixed*) : la séquence d'objets présentée au participant peut être aléatoire (*random*) ou définie (*fixed*) pour chaque niveau. Si l'option *fixed* est choisie, un bouton est affiché et permet à l'utilisateur de charger un fichier .txt contenant une séquence personnalisée spécifique pour chaque niveau. De plus, le nombre d'objets par niveau ne pourra pas être modifié, car il est déterminé par le fichier. Si le fichier choisi n'est pas valide, un message d'erreur apparaît. Le format du fichier est décrit dans le manuel d'utilisation. Valeur par défaut = *Fixed*.
- **Type** (*visuel* et/ou *audio*) : des cases à cocher permettent au chercheur de choisir si le visuel et/ou l'audio du paradigme sera présenté au participant. Si la case relative au visuel est décochée, rien n'est affiché dans le casque de réalité virtuelle. Si la case relative à l'audio est décochée, aucun son n'est produit lors du paradigme et il n'est pas possible de choisir un paramètre de volume. Ainsi, le chercheur peut choisir d'effectuer le *N-back* visuel, auditif ou combiné. Valeur par défaut = deux cases cochées.
- **Volume** : le niveau sonore diffusé au participant à l'aide du casque AR, si la case relative à l'audio est cochée. La valeur du volume doit se trouver entre 0.1 et 100.0 : une valeur inférieure à 0.1 sera automatiquement remplacée par 0.1, et une valeur supérieure à 100.0 par 100.0. Le volume peut être saisi dans une case ou à l'aide d'un curseur (*slider*). Cette valeur est modifiable tout au long du jeu. Valeur par défaut = 50.
- **Nombre d'objets par niveau** : le nombre d'objets qui seront affichés au participant dans un même niveau. Si le mode *fixed* est choisi, le nombre d'objets ne peut pas être modifié et est contrôlé par le fichier .txt

personnalisé. La valeur du nombre d'objets doit être supérieure ou égale à 2 : une valeur inférieure sera automatiquement remplacée par 2. Valeur par défaut = 15.

- **Sélection des objets** : permet de choisir quels objets peuvent être affichés au participant. Si un objet est décoché, il ne sera jamais montré au participant. Valeur par défaut = femme, drapeau, banc, bicyclette, voiture, chat, arbre, panneau *Stop* et maison sont cochés.
- **Vitesse** : la vitesse d'apparition des objets le long du plan de la marche. La valeur du nombre d'objets doit être supérieure ou égale à 0.01 : une valeur inférieure sera automatiquement remplacée par 0.01. Valeur par défaut = 1.00.
- **Nombre de niveaux** : le chercheur peut choisir le nombre de niveaux qu'il désire effectuer. Chaque niveau dans la séquence sera joué l'un après l'autre, aura la même durée (par défaut, 90s), une tâche particulière (*dual task*, *single cognitive task*, *single walking task*) et une difficulté particulière (1, 2, 3, 4). Le nombre de niveaux doit être entre 1 et 12 : une valeur inférieure à 1 sera automatiquement remplacée par 1, et une valeur supérieure à 12 par 12. Selon le nombre de niveaux choisi, le nombre adéquat de menus déroulants permettant de choisir la tâche et la difficulté pour chaque niveau s'affichent. Valeur par défaut = 1.
- **Choix de la tâche** (*dual task*, *single cognitive task*, *single walking task*) : pour chaque niveau de la séquence, le chercheur peut choisir la tâche à l'aide d'un menu déroulant. S'il choisit *dual task* ou *single cognitive task*, le même jeu de réalité augmentée sera affiché au participant lors de la tâche, mais il sera indiqué dans l'enregistrement de la séquence que le test a été jumelé à une tâche de marche pour *dual task*. Pour *single walking task*, rien ne sera affiché au participant lors de la tâche, mais un menu permettra au chercheur de démarrer le niveau et de l'arrêter pour permettre l'enregistrement de la séquence et des temps.
- **Choix du numéro de N-Back** (1, 2) : pour chaque niveau, il est possible de choisir à l'aide d'un menu déroulant le rang de l'objet précédent avec lequel le participant doit comparer l'objet actuel. Par exemple, en choisissant 1, le participant devra comparer chaque objet avec le précédent. Si la tâche est de type *single walking task*, il n'est pas possible de choisir un numéro de *N-Back* (le menu déroulant disparaît).
- **Presets** : un bouton permet de sauvegarder les paramètres choisis dans un fichier .txt. Un autre bouton permet de charger les paramètres ayant précédemment été sauvegardés dans un fichier .txt. Si le fichier choisi n'est pas valide, un message d'erreur apparaît.

1.2.3.2 Modifications à effectuer

La majorité du jeu a déjà été effectué. Cette section décrit les différentes modifications à apporter afin d'accommoder le jeu aux besoins du client et d'uniformiser l'interface et le visuel du jeu.

1. **Ajout de l'option auditive du N-back** : si seulement l'option auditive est activée, le participant n'observera pas les objets. Rien ne sera affiché en réalité augmentée, uniquement l'audio du test *N-back* sera présenté par l'intermédiaire des haut-parleurs du casque. Les réponses seront enregistrées de la même façon que pour le *N-back* visuel, à l'aide de boutons. L'intégration du *N-back* auditif va permettre un enregistrement des réponses et une synchronisation avec les autres dispositifs uniformes entre les divers paradigmes.
2. **Intégrer l'option auditive et visuelle** : ajouter l'audio des items apparaissant lors du jeu *N-back*. Le participant observera, par exemple un vélo, et il entendra le nom de l'objet simultanément. Ainsi, au total, trois types seront possibles : le *N-back* auditif, le *N-back* visuel et le *N-back* combiné auditif et visuel.
3. **Ajustement du visuel** : lors du jeu *N-back* visuel, en réalité augmentée, un plan rose est présent, les items défilent de chaque côté de celui-ci. Le plan rose sera remplacé par un plan sobre de taille adéquate rappelant une rue afin d'augmenter le réalisme du jeu. Les mots *Start* et *End* seront désormais immobiles.

4. **Ajout du bouton *Restart*** : pour chaque niveau et tutoriel, un bouton *Restart* sera ajouté (comme pour le paradigme *Stroop*) pour donner l'option de recommencer une séquence en cas de problème.
5. **Ajout de nouveaux paramètres** :
 - a. **Choix de la tâche et du N** : ajout de l'option de la tâche (*dual task*, *single cognitive task*, *single walking task*) et du numéro de *N-back* (1, 2) pour chaque niveau de la séquence
 - b. **Nombre de niveaux** (entre 1 et 12)
 - c. **Type de *N-back*** (auditif, visuel ou combiné)
 - d. **Volume** (si auditif ou combiné)
 - e. **Presets** (option de sauvegarder et charger des paramètres prédéfinis)
 - f. **Fixed Sequence** (option de charger une séquence d'objets fixe prédéfinie pour chaque niveau)
6. **Ajout d'un résumé des résultats fonctionnels** : celui qui est présent compte plusieurs lacunes. Il sera corrigé en améliorant le visuel et en permettant à l'utilisateur de retourner au menu principal.
7. **Ajout du Resting Time et de Send Trigger** : un chronomètre (*Resting Time*) sera affiché à l'écran du chercheur pouvant aussi lui permettre d'attendre une durée précise avant de débiter chaque niveau. Un bouton (*Send Trigger*) lui offrira la possibilité d'envoyer manuellement un événement à l'Arduino.
8. **Ajouter les instructions dans la vue du chercheur** : pour chaque niveau de la séquence

1.2.3.3 Descriptions des scènes et pages

1. Scène 1 : Menu principal

La première scène correspond au menu principal et inclut le choix du paradigme et des différents paramètres de jeu. L'affichage se fait uniquement au niveau du chercheur, le participant n'a aucun affichage en réalité augmentée. La première page de la scène 1 est présentée à la [Figure 1-22](#) : la vue du chercheur contient une liste déroulante permettant de choisir le paradigme désiré, parmi les options *Stroop* et *N-back*. Lors de la sélection du paradigme (dans ce cas-ci, *N-back*), on passe à la page 2 de la scène 1.

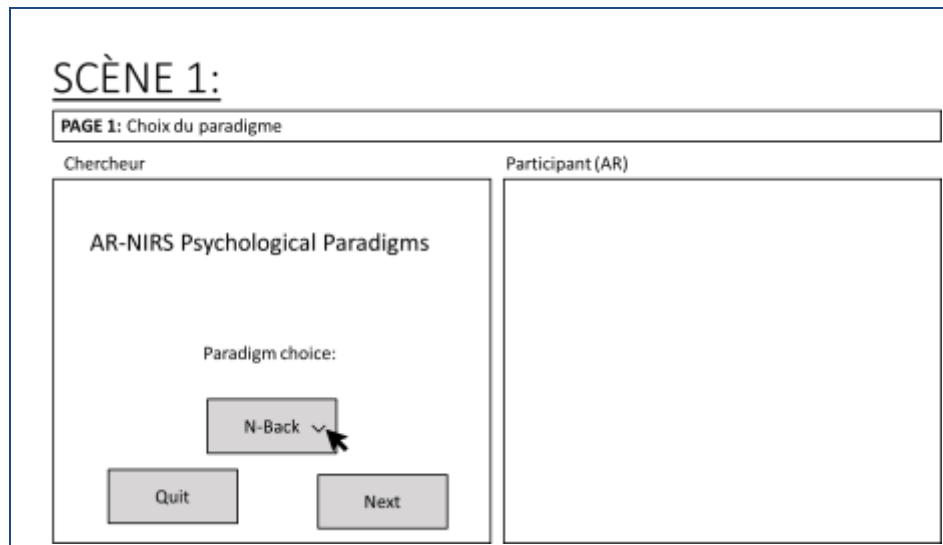


Figure 1-22 : Page 1 de la scène 1 – choix du paradigme.

La page 2 (Figure 1-23) permet de sélectionner le dossier où enregistrer les résultats ainsi qu'entrer le numéro du port pour se connecter à l'Aduino, de la même façon que pour le *Stroop* (voir section 1.2.2.2). Lorsque l'utilisateur sélectionne *Next*, la page 3 s'affiche.

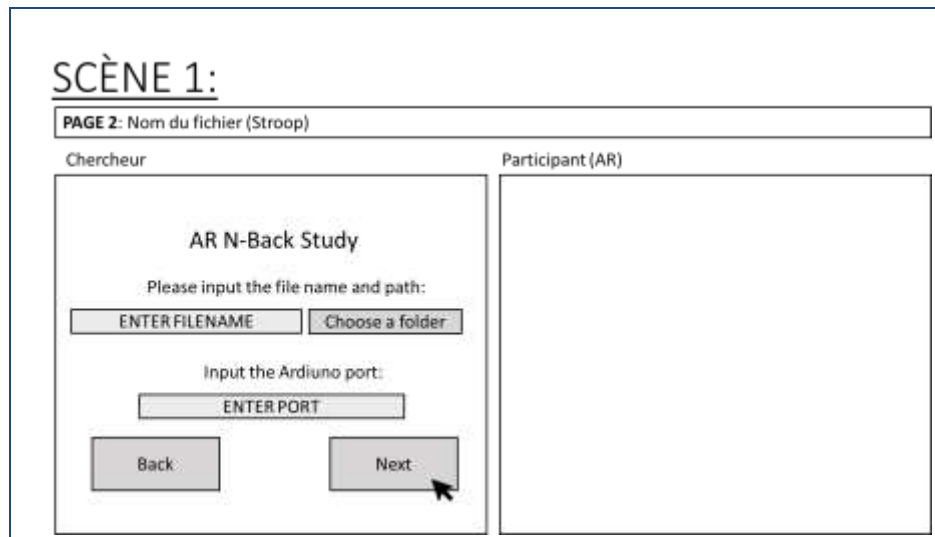


Figure 1-23 : Page 2 de la scène 1 – nom du fichier et port.

La page 3 (Figure 1-24) permet la sélection des paramètres tels que décrits à la section 1.2.3.1. L'utilisateur peut retourner à la page précédente à l'aide du bouton *Back* ou continuer vers la scène 2 du jeu en appuyant sur *Start*.

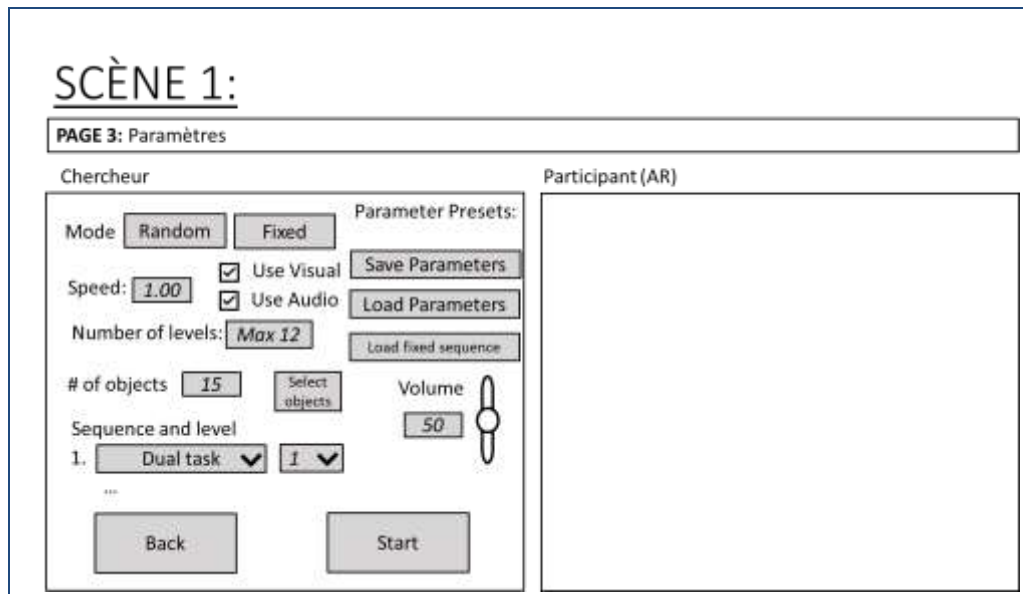


Figure 1-24 : Page 3 de la scène 1 – paramètres.

2. Scène 2 : Jeu N-Back

La scène 2 comprend les tutoriels et le jeu *N-back*. La première page (Figure 1-25) affiche un message d'avertissement au chercheur pour qu'il s'assure que la calibration du casque est complétée avant de débiter le jeu. Il peut également ajuster le volume du casque ainsi qu'envoyer un *trigger* vers l'Arduino. Lorsqu'il sélectionne *Display instructions*, il passe à la page 2 (Figure 1-26) où les instructions sont affichées afin qu'il puisse les lire au participant. Lors des pages 1 et 2, le plan (rue) est affiché sur l'écran du participant.

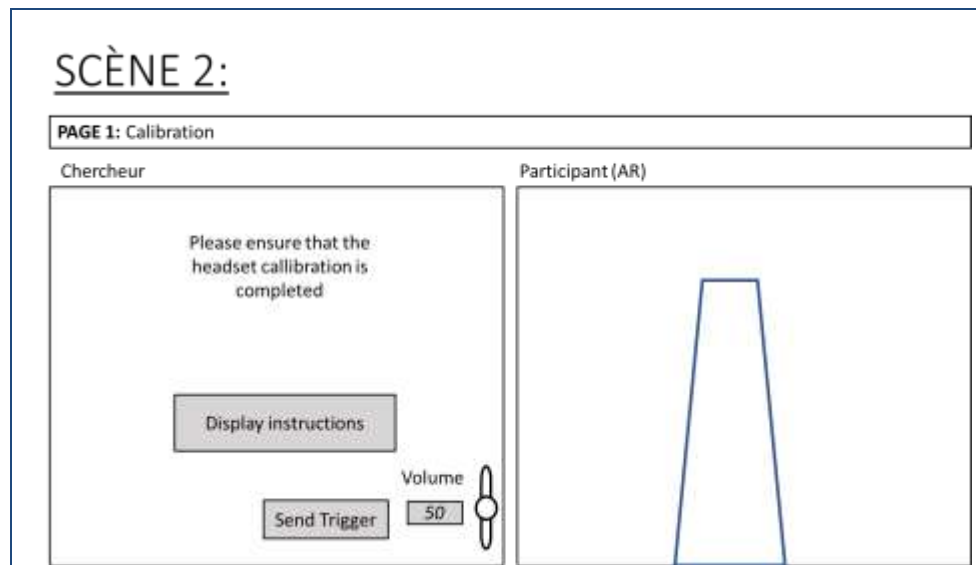


Figure 1-25 : Page 1 de la scène 2 – calibration.

À la page 2 (Figure 1-26), un minuteur démarre afin que le chercheur puisse compter un certain temps de repos (*Resting Time*). Il a aussi la possibilité d'envoyer un *trigger* à l'Arduino avant et après ce temps de repos. Le niveau et le

nombre N sont affichés sur l'écran du chercheur. Les pages 1 et 2 sont affichées avant chaque tutoriel et chaque niveau. Lors du démarrage d'un niveau ou d'un tutoriel, un signal sonore est émis.

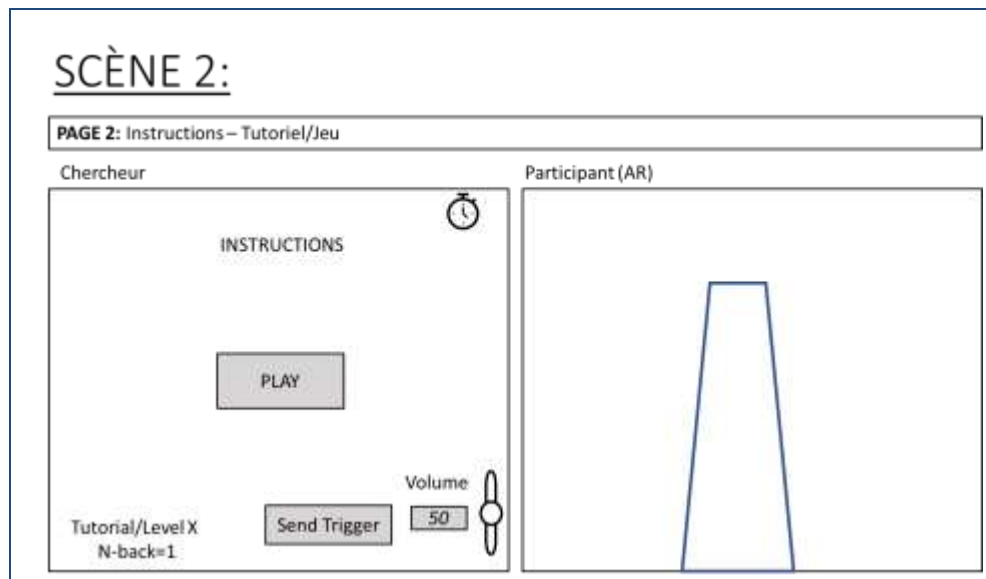


Figure 1-26 : Page 2 de la scène 2 – instructions.

La page 3 correspond au jeu *N-back* (Figure 1-27). Lors du début du jeu, *START* s'affiche à l'écran du participant. La séquence des items est inscrite sur l'écran du chercheur et les réponses du participant s'affichent au fur et à mesure qu'il répond. Sur l'écran du participant, les items défilent à la vitesse sélectionnée et restent affichés pendant une durée de 5s si l'option visuelle a été sélectionnée. Peu importe la vitesse de défilement des objets, il y a toujours la même période de temps d'attente entre deux objets. Si l'option audio est sélectionnée, le nom de chaque objet, lors de son apparition, sera énoncé dans les haut-parleurs du participant. Lorsque tous les objets ont été affichés, *END* s'affiche à l'écran du participant et la page 4 ou 5 (tutoriel ou jeu) s'affiche par la suite (Figure 1-28 et Figure 1-29).

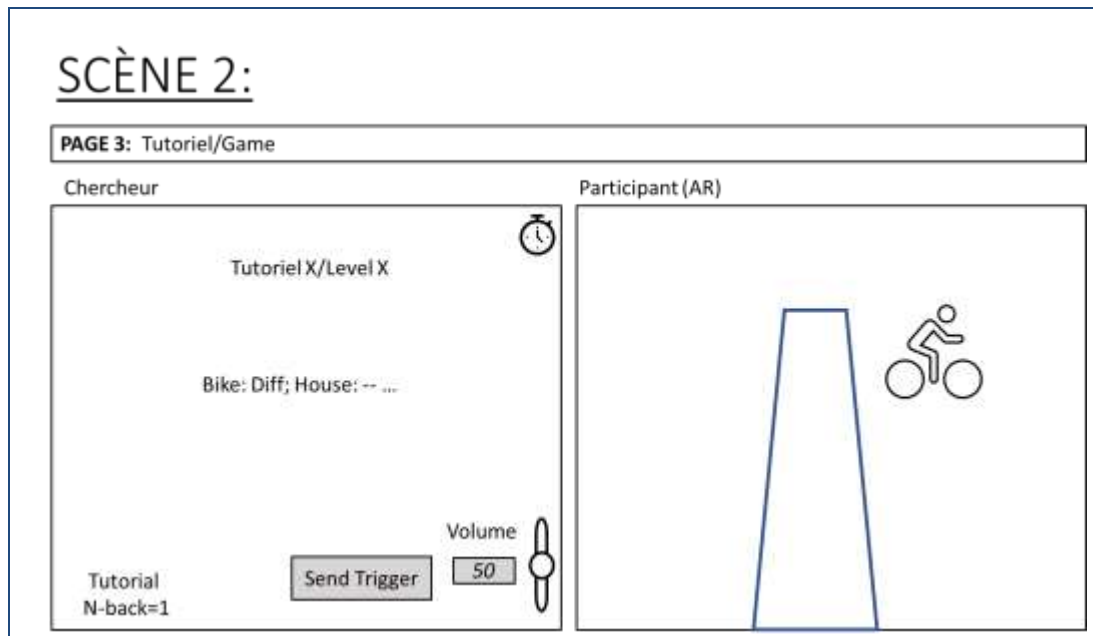


Figure 1-27 : Page 3 de la scène 1 – jeu.

La page 4 (Figure 1-28) présente le menu de fin de tutoriel tandis que la page 5 (Figure 1-29) présente le menu de fin de jeu. Le chercheur peut alors afficher les résultats en sélectionnant *RESULTS* : dans ce cas, la page 6 s’affichera (Figure 1-30). Si le dernier niveau était un tutoriel, il peut passer au prochain tutoriel en sélectionnant *NEXT TUTORIALS* ou passer directement au jeu avec *SKIP TUTORIALS*. Il peut également quitter le jeu à l’aide du bouton *QUIT* et la fenêtre du jeu se fermera.

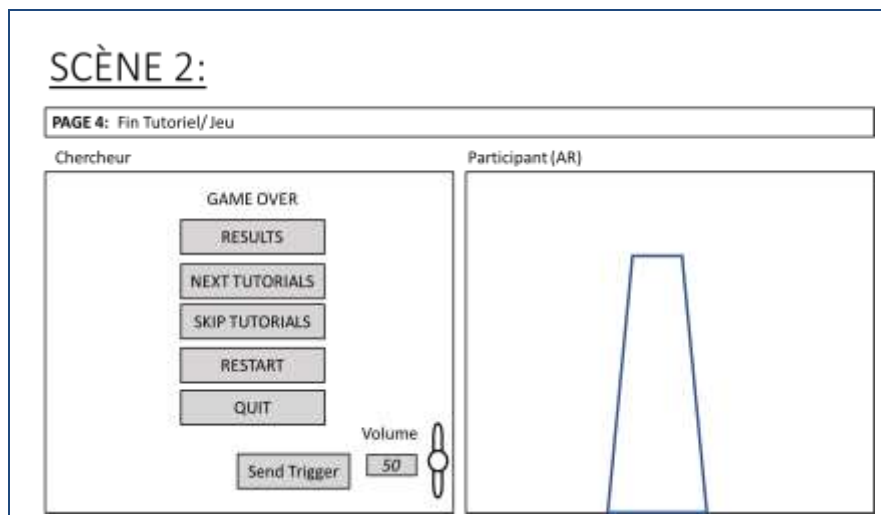


Figure 1-28 : Page 4 de la scène 1 – fin du Tutoriel.

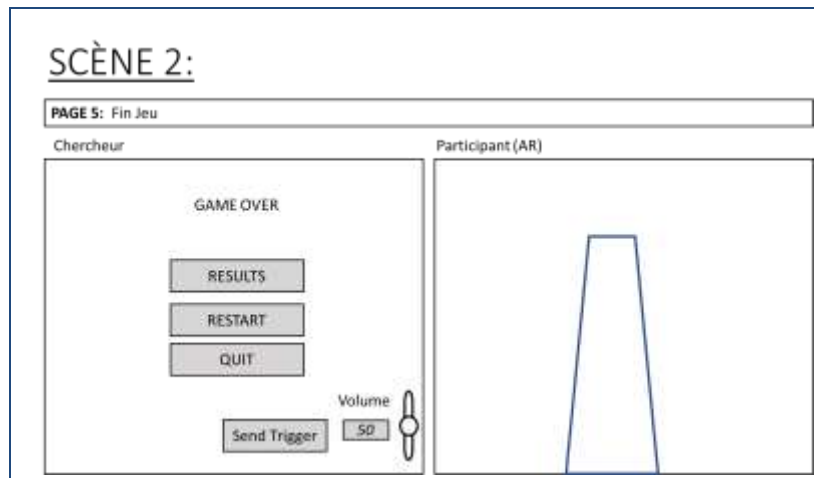


Figure 1-29 : Page 5 de la scène 2 – fin du jeu.

La page 6 (Figure 1-30) présente un résumé des résultats, soit le pourcentage de questions réussies. En sélectionnant un niveau, les réponses du participant s’afficheront. Les informations complètes sont disponibles dans le fichier .txt créé. Le chercheur peut revenir au menu de fin de jeu à l’aide du bouton *Back*.

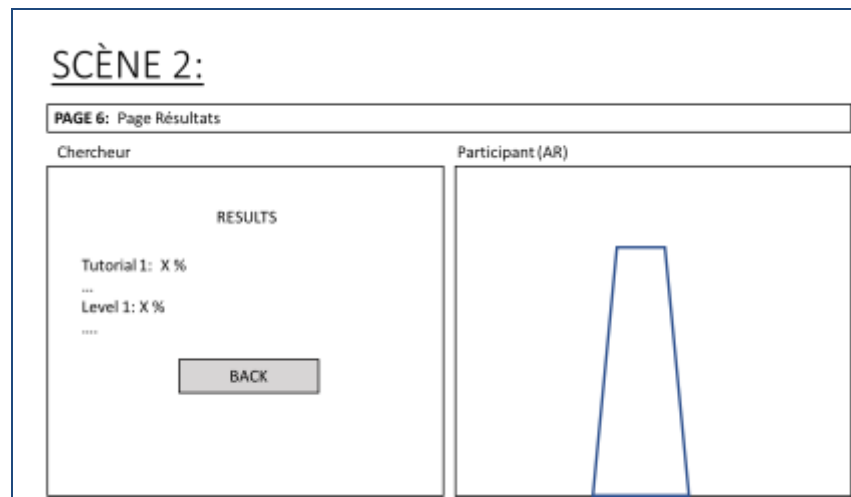


Figure 1-30 : Page 6 de la scène 2 – résultats.

1.3 Synchronisation des signaux

Cette section présente la synchronisation des signaux. On y retrouve deux sections, soit sur le matériel utilisé pour effectuer la synchronisation ainsi que sur sa mise en application.

1.3.1 Équipement et matériel utilisés

Le **Tableau 1.2** contient le nom de référence, le modèle et le fournisseur de l'équipement et le matériel utilisé dans le design de la synchronisation.

Tableau 1.2 : Équipement et matériel pour la synchronisation.

No.	Nom de référence	Modèle	Fournisseur
Équipements			
1	Casque NIRS	Dual Brite MKII	Artinis Medical Systems
2	Casque AR	Meta 2	Meta
3	Tapis roulant	Trackmaster TMX425CP	Full Vision Inc.
4	Capteur EMG	DTS Footswitch	Noraxon U.S.A. Inc.
5	Télécommande	PortaSync MKI	Artinis Medical Systems
6	Boutons de réponse	Millisecond accurate hand-held USB response button key features	The Black Box ToolKit
7	Boîte de réponse	Millisecond accurate 4 port USB response box key features	The Black Box ToolKit
8	Microcontrôleur	Arduino Mega 2560 R3	Arduino
9	Système trigger	Vicon Sync Box	Vicon
Matériel			
10	Câble PortaSync	Câble BNC Femelle à jack 3.5 mm	-
11	Câble Footswitch	Câble BNC Femelle à BNC Femelle	-
12	Câble Arduino	USB 2.0 Cable Type A/B	-

1.3.2 Mise en application

Le design retenu pour faire la synchronisation des différents systèmes est basé sur un principe d'enregistrement du temps et de communication au NIRS par le biais du microcontrôleur et de la télécommande *PortaSync* du NIRS. Un schéma du design est présenté à la **Figure 1-31** suivante :

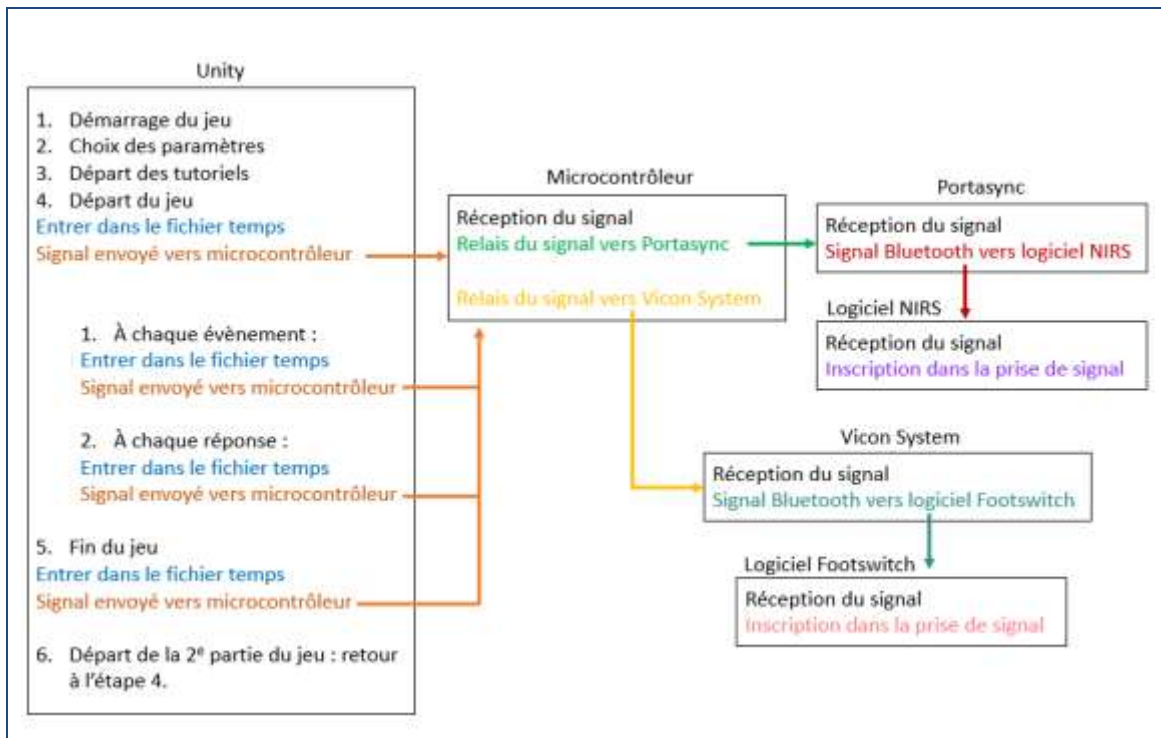


Figure 1-31 : Design de la synchronisation.

1.3.2.1 Synchronisation pour le calcul du temps de réponse du participant

Le premier aspect important de la synchronisation est de permettre le calcul des temps de réponse du participant, dont le temps entre l'évènement et la réponse du participant. Pour ce faire, on crée un fichier nommé *date-heure_MASTER* qui permet de faire l'enregistrement et le suivi des évènements du jeu. À chaque début de test, évènements, réponses et fin de test, l'heure est inscrite dans le fichier. Toutes les situations nécessitant une réponse du participant sont considérées comme étant des évènements. Par exemple, dans le test *N-back*, chaque fois qu'un objet apparaît, cela est considéré comme un évènement et l'heure de l'apparition de l'évènement est inscrite dans le fichier temps. Pour la réponse, les boutons *Millisecond accurate hand-held USB response button* sont connectés à l'ordinateur et sont intégrés au jeu, dans *Unity*, sous la forme de clavier grâce à la boîte de réponse pour le paradigme *N-back*. Pour le paradigme *Stroop*, les réponses *handtracking* proviennent directement du casque AR et peuvent être enregistrées, comme mentionné à la section 1.2.2.5. Cela signifie que l'acquisition d'une réponse pourra se faire directement dans *Unity* et permettre l'enregistrement de la réponse, du temps de réponse et de l'heure dans le fichier temps lorsque l'utilisateur sélectionne un bouton (en réalité augmentée ou à l'aide des boutons physiques) tous réunis dans un même logiciel. Afin d'écrire les évènements dans le fichier *MASTER*, à chaque évènement que l'on veut avoir dans notre liste de checkpoint, on ajoute une ligne référençant une fonction *checkpoint*. La fonction *checkpoint* écrit alors dans le fichier .txt le type de *checkpoint* (début, fin, réponses...) et l'heure à laquelle l'évènement s'est produit.

La Figure 1-32 présente le design du fichier *MASTER*. On peut voir un en-tête présentant les paramètres, puis une liste de *checkpoint* de fonctions diverses : la fin du menu, le début d'un niveau, le début d'un tutoriel, un évènement *Resting Time* (déclencher par le bouton *Resting Time*, une question posée, une réponse donnée, et la fin d'un niveau.

A :

```
Parameter; Filename; C:\Users\sandrine\Desktop\test\2022_03_30_15_11.txt
Parameter; Arduino Port; COM3
Parameter; Trial Time; 90
Parameter; Number Trials; 1
Parameter; Sequence; Control,Dual Task
Parameter; Sequence Levels; 0, 1
Parameter; Game Mode; Fixed
Checkpoint; End of Menu; 15:11:13.425
Checkpoint; Difficulty: Control 0; 15:11:21.428
Checkpoint; Participant's response: RED; Response time: 0.4242221sec; 15:11:29.863
Checkpoint; Participant's response: RED; Response time: 0.1888924sec; 15:11:30.055
Checkpoint; Participant's response: RED; Response time: 0.1619302sec; 15:11:30.220
Checkpoint; Participant's response: RED; Response time: 0.1560876sec; 15:11:30.382
Checkpoint; Participant's response: RED; Response time: 0.1630771sec; 15:11:30.553
Checkpoint; Participant's response: RED; Response time: 0.3389256sec; 15:11:30.896
Checkpoint; Result: 2/6; 15:11:30.901
Checkpoint; Average Response Time: 1.572355813333333; 15:11:30.925
Checkpoint; Difficulty: Dual Task 1; 15:11:49.000
Checkpoint; Question shown; True response: RED; 15:11:49.812
Checkpoint; Participant's response: RED; Response time: 3.4104248sec; 15:11:53.223
Checkpoint; Question shown; True response: BLUE; 15:11:53.228
Checkpoint; Participant's response: RED; Response time: 1.5052575sec; 15:11:54.732
Checkpoint; Question shown; True response: RED; 15:11:54.738
Checkpoint; Participant's response: RED; Response time: 0.8119638sec; 15:11:55.550
Checkpoint; Question shown; True response: GREEN; 15:11:55.556
Checkpoint; Participant's response: RED; Response time: 0.0498425sec; 15:11:56.005
Checkpoint; Question shown; True response: RED; 15:11:56.412
```

B :

```
Parameter; Filename; C:\Users\sandrine\Desktop\test\2022_03_30_15_11_Master.txt
Parameter; Arduino Port; COM3
Parameter; Mode; Random
Parameter; Number Trials; 1
Parameter; Sequence; Tutorial,Tutorial,Tutorial,Tutorial,Tutorial,Tutorial,Tutorial,Dual Task
Parameter; Sequence HBack; 1, 1, 1, 1, 1, 1, 1, 1
Parameter; Number Objects; 15
Parameter; Speed; 1
Checkpoint; End of Menu; 15:13:25.612
Checkpoint; Trigger Resting Time; 15:13:29.329
Checkpoint; Trigger Resting Time; 15:13:30.062
Checkpoint; Trigger Resting Time; 15:13:30.311
Checkpoint; Trigger Resting Time; 15:13:30.354
Checkpoint; Trigger Resting Time; 15:13:30.884
Checkpoint; Trigger Resting Time; 15:13:31.050
Checkpoint; Trigger Resting Time; 15:13:31.291
Checkpoint; Trigger Resting Time; 15:13:31.542
Checkpoint; Trigger Resting Time; 15:13:35.263
Checkpoint; Trigger Resting Time; 15:13:35.513
Checkpoint; Trigger Resting Time; 15:13:35.753
Checkpoint; Start; 15:13:44.824
Checkpoint; Spawn Bike(Clone) (UnityEngine.GameObject); 15:13:50.003
Checkpoint; Spawn House(Clone) (UnityEngine.GameObject); 15:13:55.154
Checkpoint; Spawn House(Clone) (UnityEngine.GameObject); 15:14:00.316
Checkpoint; Same; 15:14:04.188
Checkpoint; Spawn Girl(Clone) (UnityEngine.GameObject); 15:14:05.618
Checkpoint; Diff; 15:14:07.858
Checkpoint; Spawn Girl(Clone) (UnityEngine.GameObject); 15:14:10.922
Checkpoint; Spawn Stop sign(Clone) (UnityEngine.GameObject); 15:14:16.077
Checkpoint; Spawn Girl(Clone) (UnityEngine.GameObject); 15:14:21.226
Checkpoint; Spawn Flag(Clone) (UnityEngine.GameObject); 15:14:26.375
Checkpoint; Spawn Flag(Clone) (UnityEngine.GameObject); 15:14:31.523
Checkpoint; Diff; 15:14:36.002
```

Figure 1-32 : Design du fichier MASTER. A : Stroop. B : N-Back

Le fichier temps est écrit sous forme .txt pour permettre une écriture facile et simple des checkpoints ou point de contrôle et permettre l'exportation des données dans d'autres formats. En effet, le fichier .txt est facilement convertible dans la suite *Microsoft*. Pour calculer le temps de réaction, il suffit de soustraire le temps de l'évènement au temps de la réponse. Un code *Matlab* construit à cet effet sera fourni pour faire le calcul et la présentation des temps de réponse pour les 2 jeux. Les temps de réaction pour le jeu *Stroop* seront également affichés sur l'écran du chercheur à la fin du test et dans le fichier *MASTER*. Dans le code *Matlab*, 2 informations sont à remplir pour son utilisation, soit le nom du fichier *MASTER* à utiliser et le type de jeu.

Afin de s'assurer qu'il n'y a pas eu de problème dans la synchronisation entre les systèmes AR et Arduino, deux autres fichiers ont été créés, le fichier *date-heure_AR* affiche pour chaque évènement à dater le temps d'envoi effectué. Cela signifie qu'il est possible de vérifier que l'ordinateur de l'AR n'a pas lagué durant le test puisque le temps d'envoi correspond au moment lorsque l'ordinateur reçoit un évènement et lorsque l'ordinateur le transfère à l'Arduino. De la même façon le fichier *date-heure_Arduino* retrace les temps d'envoi entre la réception du signal de l'Arduino jusqu'à la transmission du signal vers les télécommandes pour tous les évènements.

1.3.2.2 Synchronisation avec Arduino

Le deuxième aspect de la partie synchronisation porte sur la synchronisation du jeu avec l'imagerie NIRS et les capteurs EMG. Pour ce faire, nous avons décidé d'utiliser un microcontrôleur qui sert de relais vers la télécommande accessoire du NIRS, le *PortaSync* (Artinis Medical Systems, 2021) et les systèmes *trigger* des *Footswitches*. Le microcontrôleur permet de détecter le signal provenant d'*Unity* et d'envoyer le signal analogiquement vers le *PortaSync* et le *Vicon Sync Box*. Le *PortaSync* permet d'envoyer un signal Bluetooth vers le NIRS. Lorsque le logiciel *Oxysoft* du NIRS le reçoit, des *triggers* sont enregistrés directement lors de l'acquisition des données. Le processus se fait en temps réel. Le système *trigger* des *Footswitches* permet d'envoyer un signal vers le logiciel *Vicon Nexus* en temps réel.

Le démarrage du programme *Oxysoft*, soit le début de l'acquisition du NIRS, se fait avant les tests. Le microcontrôleur doit être connecté à l'ordinateur avec le jeu pour recevoir les signaux. Pour ce faire, le chercheur doit trouver le port dans lequel le microcontrôleur est connecté : un guide sera créé pour faciliter l'acquisition du numéro du port. Le chercheur doit entrer le numéro dans une boîte texte présentée au démarrage, tout comme le nom du fichier utilisé pour enregistrer les temps (voir [Figure 1-8](#)). Lorsque l'Arduino reçoit un signal de connexion de l'ordinateur, une lumière verte s'allume. Pour démarrer la synchronisation, le chercheur doit démarrer le jeu à partir de l'écran du chercheur. Les boutons sont activés dès le lancement du jeu, mais l'acquisition de réponse n'est possible que dans le jeu lorsque le programme attend une réponse du participant. La [Figure 1-33](#) présente le pseudo-code de l'envoi des signaux vers le microcontrôleur et le pseudo-code de l'envoi des signaux du microcontrôleur vers le *PortaSync* et le système *trigger*.

```
Envoi du signal du jeu vers l'Arduino

public static void TriggerArduino(string line)
{
    if (the port n'est pas ouvert)
        Ouvre le port
    Écrire le string au port
}

Envoi du signal de l'Arduino vers le PortaSync et le Vicon Sync Box

Définir une variable inputTrigger

if (Donnée) {
    inputTrigger = Lire du port;
    if (inputTrigger == 'C') {
        Allumer la lumière;
    }
    if (inputTrigger == '0') {
        analogWrite(PortaSync);
        analogWrite(Footswitch);
    }
}
```

Figure 1-33 : Pseudo-code pour la communication avec l'Arduino et le *PortaSync*.

La fonction *TriggerArduino* permet de prendre en paramètre un *string*. Ensuite, le port est ouvert, s'il ne l'est pas le *string* est envoyé vers le port de l'Arduino. Cela permet de différencier les *triggers* envoyer vers l'Arduino. Cette

fonction est appelée lors de la connexion, lors du début et la fin d'un jeu, lors d'une réponse et lorsqu'il y a un événement dans le jeu. Le code de l'Arduino permet de lire du port si une donnée est présente. Il différencie entre le *trigger* pour la lumière et pour le relais de signal. Si le *trigger* reçu est 'c', la lumière est allumée. Si le *trigger* reçu est '0', un signal analogique est envoyé vers le *PortaSync* et le système *trigger* des *Footswitches*.

La connexion entre l'ordinateur du jeu et l'Arduino se fait à l'aide d'un câble *USB 2.0 Cable Type A/B*. De plus, un PCB permet de faciliter les connexions *Arduino-PortaSync* avec le câble *PortaSync* et *Arduino-Footswitches* avec le câble *Footswitch*. Le PCB est composé d'une lumière et de trois sorties BNC mâles. Le PCB se connecte à l'Arduino avec des pins de type *header*. La vue 3D du PCB se trouve à la **Figure 1-34**.



Figure 1-34 : Vue 3D du PCB.

1.3.2.3 Synchronisation avec le tapis roulant

Le dernier aspect de la synchronisation est la synchronisation avec le tapis roulant. Quand le jeu *N-back* est choisi dans *Unity*, un champ permettant d'indiquer la vitesse du tapis roulant est présent. Cette vitesse est prise en considération pour la vitesse de déplacement des objets. Avant de débiter le niveau, le chercheur peut faire marcher le participant sur le tapis roulant en augmentant la vitesse graduellement pour trouver la vitesse la plus confortable pour le participant. La vitesse n'est pas prise en considération pour le paradigme *Stroop* puisqu'il n'y a aucun objet qui bouge.

1.4 Intégration physique

Ce module est divisé en deux sections qui représentent l'intégration physique des différents dispositifs utilisés. La **section 1.4.1** traite de l'assemblage des casques NIRS et AR sur le participant et la **section 1.4.2** présente la solution de stockage du microcontrôleur et des fils qui lui sont reliés.

1.4.1 Intégration des casques NIRS et AR

La solution doit permettre l'utilisation du casque AR et du casque NIRS simultanément par le participant. Il doit également être confortable pour l'utilisateur et le poids doit être supportable par la tête du participant. De plus, il est nécessaire que le système développé n'empêche pas la calibration des yeux par le casque AR et n'encombre pas le

champ de vision du casque AR lors des paradigmes. Enfin, le dispositif doit permettre l'enregistrement de signaux par le casque NIRS sur le participant en limitant les interférences.

Afin de respecter les exigences fonctionnelles et les contraintes du produit, le système d'intégration physique NIRS-AR comprend 3 sous-systèmes. Le sous-système 1 est le large appui-tête, la mousse de confort, qui est illustré à la **Figure 1-35**.



Figure 1-35 : Vue de face et de haut de la mousse de confort.

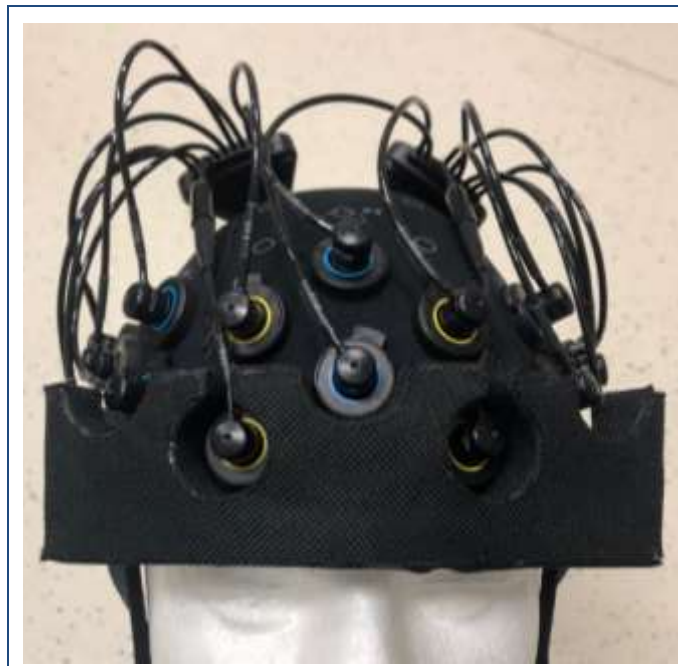


Figure 1-36 : Assemblage du NIRS et de la mousse de confort.

Ce sous-système est réalisé à partir d'une mousse épaisse semi-rigide EVA (éthylène vinyle acétate) afin d'être suffisamment flexible pour épouser la forme de la tête du participant et suffisamment ductile pour amortir l'effet de compression du casque AR sur le casque NIRS. Les trous et arrondis présents permettent une correspondance avec les capteurs NIRS pour une meilleure insertion du dispositif entre les appareils. L'épaisseur de la mousse de confort est semblable à celle des capteurs insérés afin d'éviter une compression supplémentaire sur la tête du participant et de ne

pas altérer leur fonctionnement. Les dimensions du dispositif respectent celles du bloc au-dessus des lunettes du casque AR pour ne pas encombrer le champ de vision et sont présentées en annexe avec l'épaisseur. De plus, le dispositif est recouvert d'un tissu imperméable afin de le nettoyer à l'aide d'un chiffon humide si nécessaire. La mousse de confort comprend également des bandes adhésives de types *Velcro* qui coïncident avec des bandes de *Velcro* ajoutées sur le casque NIRS pour limiter le mouvement du dispositif lors de l'installation des deux casques. Le sous-système inclut aussi une tige en fer torsadée insérée au milieu de la mousse dans le sens de la longueur afin d'épouser la forme de la tête de l'utilisateur. Enfin, la mousse de confort se positionne sur le casque NIRS comme présenté à la **Figure 1-36** avant la mise en place du casque AR.

Le sous-système 2 comprend deux petits appuis-tête, les *Adjust Foams*, qui sont illustrés à la **Figure 1-37**.

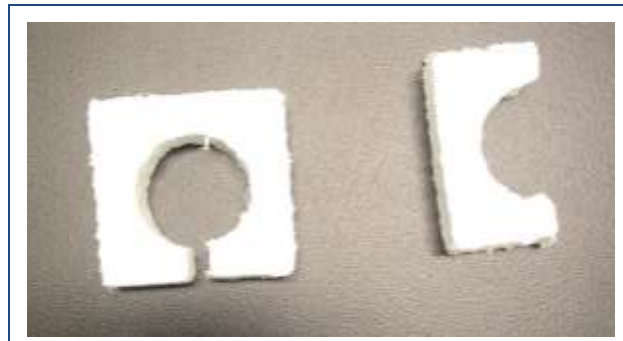


Figure 1-37 : Vue de haut et de face des *Adjust Foams*.

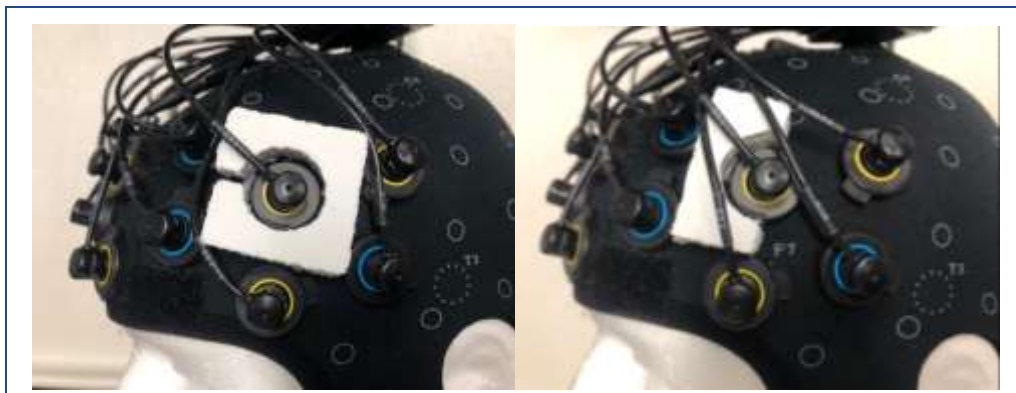


Figure 1-38 : Exemple de positionnement des *Adjust Foams*.

Ce sous-système est également réalisé en mousse EVA pour respecter les mêmes contraintes et fonctionnalités que le sous-système 1. Le recouvrement est également constitué de tissu imperméable pour permettre un entretien du sous-système. Il inclut un dispositif avec un trou coïncidant avec le diamètre d'un capteur du NIRS afin d'être inséré autour d'un capteur. Le second dispositif est composé d'un arrondi pour le rendre plus flexible, adaptable et lui permettre une insertion plus facile n'importe où entre les casques. Les dimensions des deux dispositifs respectent les espaces entre les capteurs afin de permettre une grande liberté d'insertion et sont présentées en annexe. Ce sous-système est conçu pour se positionner après l'installation des casques NIRS et AR, sur le casque NIRS comme l'illustre la **Figure 1-38** et ainsi soulager les zones de pression exercée sur les côtés de la tête du participant.

Le sous-système 3 correspond à la poche présentée à la **Figure 1-39**. Ce sous-système est réalisé en tissu et comprend des sangles pour le positionner à l'arrière du casque AR après son installation comme illustré à la **Figure 1-40**. Les sangles incluent des bandes adhésives en *Velcro* pour conserver leur fixation au casque AR. La poche permet alors le maintien de la boîte de transmission de signal du NIRS afin de limiter sa mobilité et ainsi des interférences lors de mouvement du participant. Les dimensions de la poche correspondent à celles du boîtier de transmission et sont présentées en **Figure 1-41**.



Figure 1-39 : Poche pour le boîtier de transmission NIRS.



Figure 1-40 : Positionnement de la pochette du boîtier.

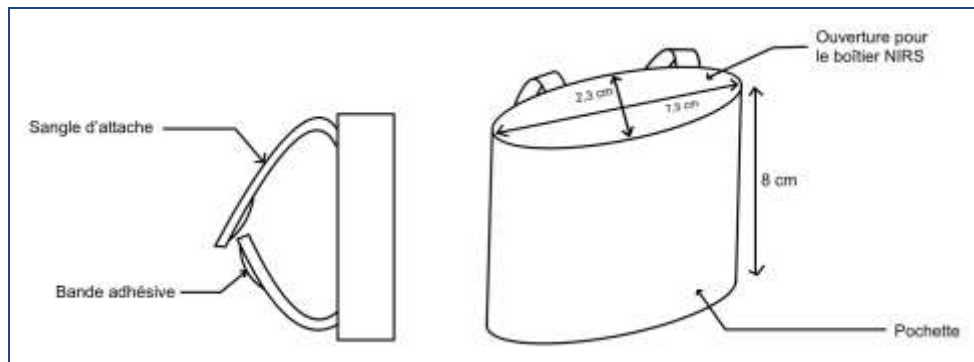


Figure 1-41 : Illustration de la pochette pour le boîtier NIRS.

1.4.2 Design de la boîte Arduino

La fonction principale du boîtier est de sécuriser la synchronisation entre les différents systèmes qui font partie du projet. Le boîtier doit limiter le mouvement des fils et des branchements et être électriquement sécurisé. Il doit aussi avoir l'espace nécessaire pour accueillir confortablement le contrôleur Arduino et le PCB. De plus, la structure doit pouvoir résister au choc (par exemple, un objet qui tombe dessus soudainement) et conserver son intégrité physique pendant et après chaque utilisation du système. Pour finir, il doit pouvoir accommoder les sorties BNC qui s'attachent au PCB ainsi que la sortie USB de l'Arduino vers l'ordinateur.

La boîte et le couvercle (Figure 1-42) sont modélisés avec *CATIA* et *SOLIDWORKS* dans le but de les générer avec une imprimante 3D. Les dessins techniques comprenant les dimensions spécifiques pour le boîtier utilisé dans le cadre de ce projet se trouvent dans l'annexe.

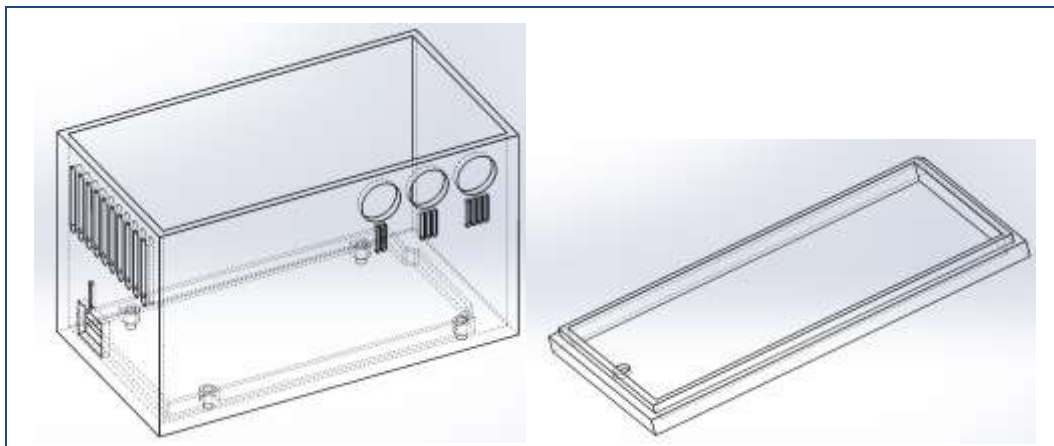


Figure 1-42 : Boîte de stockage (à gauche) et son couvercle (à droite).

La boîte (Figure 1-42) a un contour de la taille de l'Arduino en bas pour permettre la fixation à l'aide de vis et écrous. Le boîtier est percé d'un trou à l'avant au niveau de la sortie USB de l'Arduino. Il y a aussi trois trous situés vers le haut du boîtier pour permettre de rendre visibles les sorties BNC selon le design du PCB. Le couvercle (Figure 1-42) vient fermer la boîte hermétiquement et est conçu de manière à ne pas augmenter le volume occupé par la boîte tout en fournissant un accès facile à l'Arduino. Un trou se trouve dans le couvercle pour permettre de visualiser la lumière verte. Des fentes de ventilation se trouvent au-dessus de la sortie USB pour éviter la surchauffe de l'Arduino et les composants du PCB. La boîte et le couvercle sont imprimés à partir de plastique PLA qui possède une bonne résistance au choc.

Le boîtier est muni d'un support aussi imprimé à partir de plastique PLA (**Figure 1-43**). Ce support vient s'ajouter en dessous de la boîte, vers le milieu. Le côté triangulaire se positionne du côté des sorties BNC. Le support a pour but d'empêcher tout renversement du boîtier Arduino qui pourrait survenir à la suite d'un moment créé par les câbles BNC. Le support stabilise aussi le boîtier sur des surfaces plus irrégulières. Les spécifications pour les dimensions se trouvent en annexe.

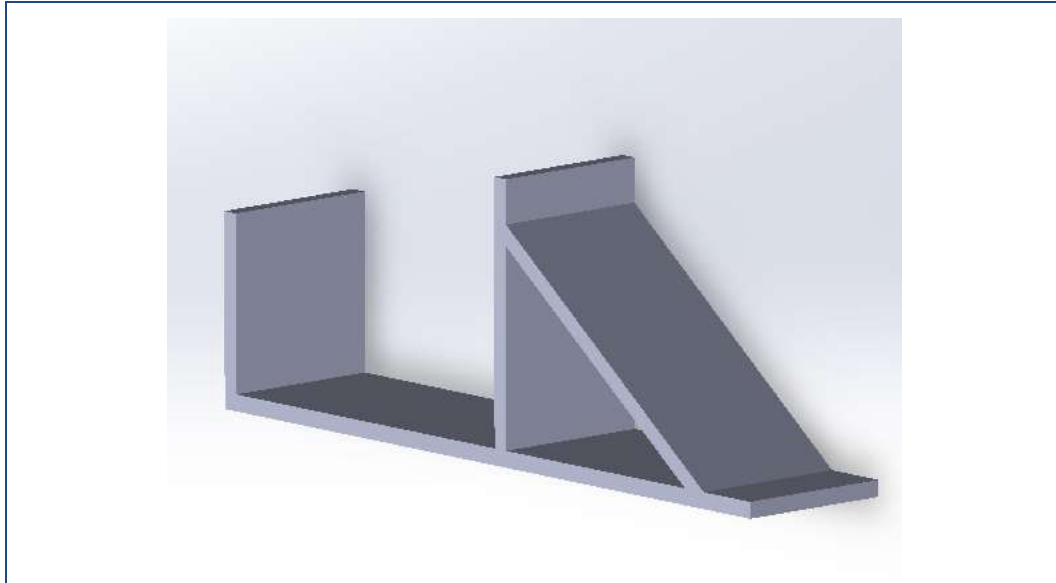


Figure 1-43 : Support du boîtier *Arduino*.

1.5 Guide de l'utilisateur

Un guide de l'utilisateur sera inclus dans le produit final. Les éléments inclus sont les suivants :

- Une description du produit
- Les spécifications du produit
- La procédure d'installation
- L'opération du produit
- La maintenance du produit
- Le débogage des programmes

2 DESIGNS ALTERNATIFS NON RETENUS

Dans cette section, les designs alternatifs au niveau des paradigmes en réalité augmentée ainsi que pour la synchronisation sont présentés.

2.1 Paradigme en réalité augmentée

Dans le design choisi, les deux jeux *N-back* et *Stroop* sont intégrés dans la même interface : le chercheur choisit le jeu dans le menu principal. Il aurait été possible de créer deux jeux complètement séparés (deux projets différents sur *Unity*). Toutefois, cette option réduit la facilité d'utilisation du jeu par le chercheur. Aussi, au niveau de la

synchronisation, en intégrant dans le même projet, les mêmes scripts peuvent être utilisés et ainsi éviter du dédoublement qui risqueraient davantage d'erreurs lors de modification d'un script dans un des deux projets.

2.1.1 Stroop

Pour le jeu *Stroop*, l'enregistrement des données pourrait être effectué à l'aide des boutons physiques, tel qu'effectué pour le jeu *N-back*. Cela simplifierait l'enregistrement des données puisqu'il serait le même pour les deux jeux. Pour le test *Stroop*, il faut être capable d'enregistrer trois réponses (*Red*, *Green*, *Blue*). Ainsi, appuyer une fois sur le bouton représenterait la réponse *Red*, deux clics : *Green*, trois clics : *Blue*. Toutefois, le temps entre chaque clic sur le bouton est de 50 ms, cela limite alors la possibilité de cliquer rapidement à plusieurs reprises sur le bouton. Il y a un risque qu'un clic ne soit pas détecté et qu'une mauvaise réponse soit enregistrée.

Pour ce qui est des boutons de réponse virtuels, il a été considéré d'utiliser des boutons de la bibliothèque UI dans *Unity*. Toutefois, l'utilisation du *handtracking* avec ce type d'objets était plus complexe qu'avec des formes 3D. De plus, ce type d'objet est seulement disponible en 2D, ce qui réduit l'effet d'immersion de l'utilisateur.

2.1.2 N-back

Puisque le test *N-back* avait déjà été développé préalablement, et que le client était satisfait du design, aucune option alternative n'a alors été considérée.

2.2 Synchronisation des signaux

2.2.3 Mise en application

Le design actuel est principalement concentré dans *Unity*, le microcontrôleur n'ayant qu'un rôle de relais. Une autre solution aurait été d'effectuer la synchronisation à partir du microcontrôleur. Pour la partie vitesse de réponse, soit la synchronisation AR et les boutons, cette solution demande qu'*Unity* envoie des signaux au microcontrôleur pour signaler les événements et les boutons auraient été connectés directement au microcontrôleur. Il y aurait alors eu le même principe de fichier temps, mais c'est le microcontrôleur qui l'aurait écrit. De plus, puisque les événements et les réponses ne seraient plus gérés dans le même programme, l'un avec *Unity* et l'autre avec le microcontrôleur, il aurait fallu s'assurer que les 2 systèmes aient la même horloge pour que les temps soient comparables. Aussi, cela aurait demandé que le microcontrôleur demande l'heure à *Unity* à plusieurs moments du protocole pour vérifier que les heures concordent. Une autre façon aurait été que le microcontrôleur impose une horloge au programme *Unity*. Aucune de ces solutions n'a été choisie puisque gérer l'événement et la réponse dans le même programme permet de réduire les délais liés à l'envoi et la réception du signal vers un 2^e système et d'enlever les risques que les horloges soient asynchrones. Pour la partie synchronisation de la réalité augmentée et du NIRS, le système aurait été le même : le logiciel *Unity* signale au microcontrôleur un événement, qui relaie l'information au *PortaSync*, qui communique avec *Brainstorm*. La réponse des boutons peut être relayée directement au *PortaSync* par le microcontrôleur.

Le démarrage des systèmes aurait pu être fait avec un microcontrôleur de type *Raspberry Pi* et un programme qui permet de contrôler à distance les différents systèmes, *Unity* et *Brainstorm*. Les boutons sont directement connectés au microcontrôleur pour cette solution. Toutefois, comme le casque AR est connecté obligatoirement à l'ordinateur avec le jeu, il a été jugé inutile et compliqué de s'imposer une contrainte de connexion sans fil. La connexion avec fil est plus fiable et rapide que le Bluetooth et les systèmes réseau tout en diminuant les risques de latence possibles.

Pour la synchronisation du tapis roulant, nous pourrions développer une solution afin que le microcontrôleur le contrôle directement, et non le jeu. Le tapis roulant serait contrôlé par le microcontrôleur avec un câble VGA. Un écran et une souris seraient ajoutés au microcontrôleur. Il y aurait une application permettant d'arrêter, de démarrer et de choisir des vitesses pour le tapis roulant. Le chercheur choisirait les différentes actions avec la souris.

2.3 Intégration des casques NIRS et AR

Le design retenu permet de respecter au mieux le champ visuel du casque AR lors de l'utilisation par un participant. De plus, il permet au casque AR d'être au plus proche des yeux de l'utilisateur lorsque celui-ci porte le casque NIRS, ce qui facilite la calibration des yeux par le casque AR. Avant de choisir ce design, un autre fut éliminé, car il ne respectait pas totalement les contraintes posées. En effet, le design précédent était fait de deux morceaux de mousse semi-rigide épais de 12,4 mm empilés. Cela permettait d'augmenter le confort du participant, mais cette solution limitait la calibration des yeux puisque le casque AR était trop éloigné du visage de l'utilisateur. Ainsi, il a été choisi de supprimer un morceau de mousse semi-rigide pour le design retenu et d'ajouter d'autres petits morceaux de mousse pour augmenter le confort du participant. Le design de l'appui-tête central fut également amélioré puisque celui-ci devait se positionner grâce à du *Velcro* sur le casque AR, mais il a été choisi de l'installer directement sur le casque NIRS pour enlever une pièce amovible du haut du casque AR et réduire encore l'espace entre le champ de vision du casque AR et les yeux du participant et rendre ainsi le système plus immersif. Pour finir, lors de l'installation du casque AR, il était difficile de conserver l'appui-tête à l'avant immobile, même avec les bandes adhésives de *Velcro*. Ainsi, il a été choisi d'ajouter une tige en fer dans la mousse de l'appui-tête pour qu'il épouse mieux la tête du participant et que l'installation soit facilitée.

Ensuite, lors des tests des prototypes, le casque AR avait tendance à être déséquilibré et pas maintenu sur la tête du participant. Ainsi, initialement un design incluant un harnais porté par le participant auquel le casque AR pourrait être attaché par une sangle à l'arrière de la tête était prévu. Ce design fut rapidement abandonné, car il représentait un encombrement chez l'utilisateur et était assez invasif. Le design suivant fut des sangles qui permettraient d'attacher de chaque côté le casque AR au casque NIRS directement. L'attachement était prévu par les trous du casque NIRS situés au niveau des oreilles de l'utilisateur. Les sangles devaient être suffisamment longues et constituées de bandes de *Velcro* sur une grande partie de leur longueur pour s'adapter aux différents participants. Finalement, ce design fut abandonné puisqu'avec les solutions d'appui-tête finales, le casque AR était suffisamment stable sur la tête du participant et permettait également à celui-ci l'exécution de mouvement.

Enfin, les morceaux de mousse devaient être recouverts de tissu puis de plastique pour permettre l'étanchéité et la désinfection de ceux-ci. Or, le design final comprend seulement un recouvrement composé de tissu imperméable pour faciliter la réalisation du produit final et cela n'affecte aucunement le respect des contraintes fixées.

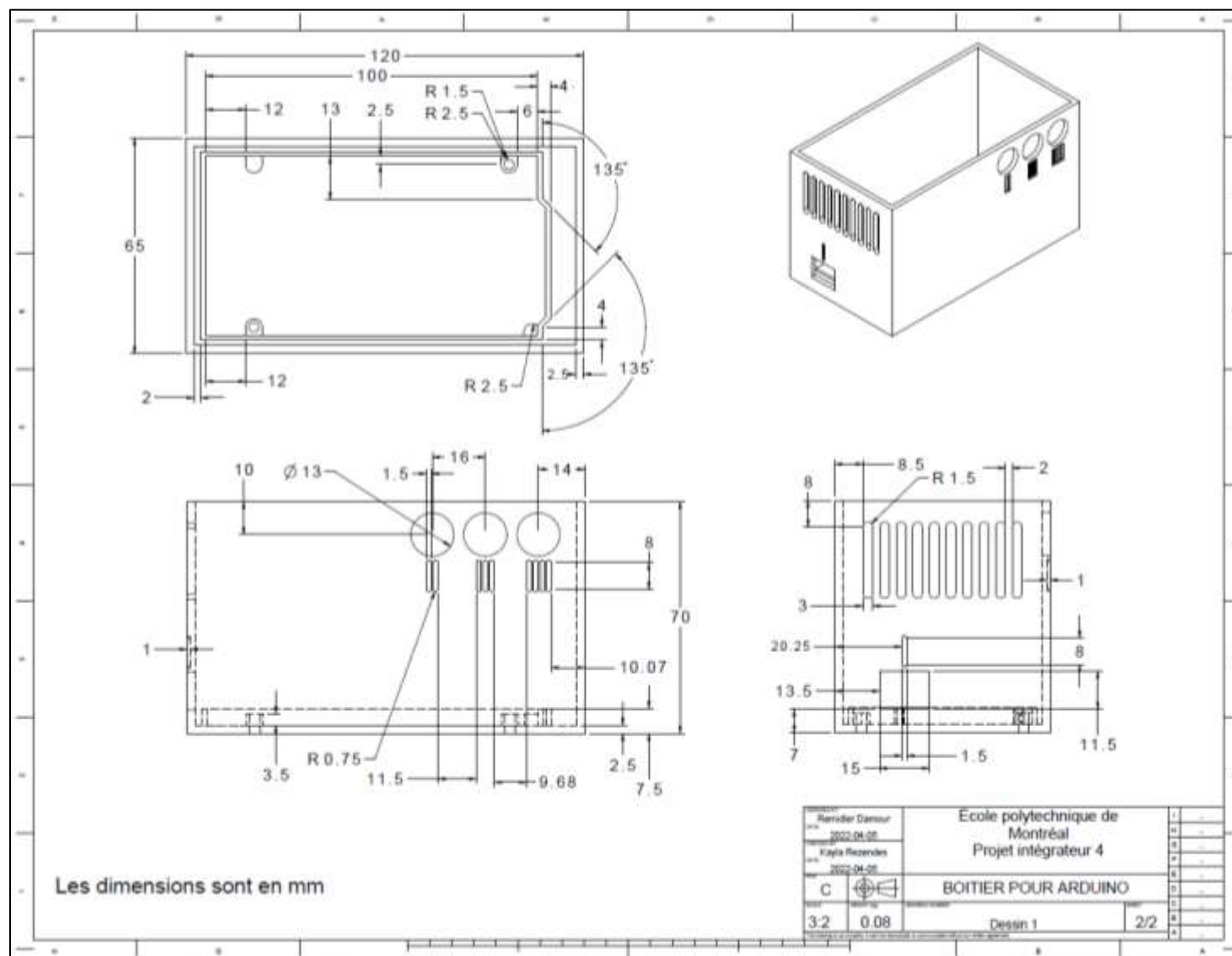
5. SOMMAIRE DES DESIGNS

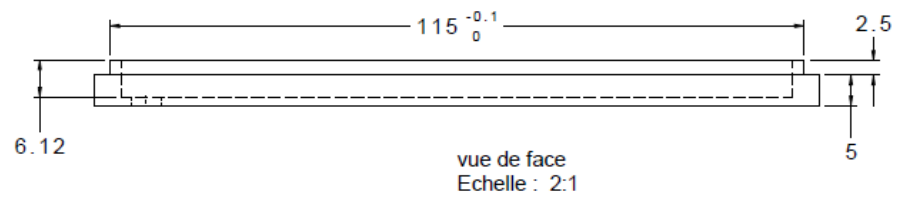
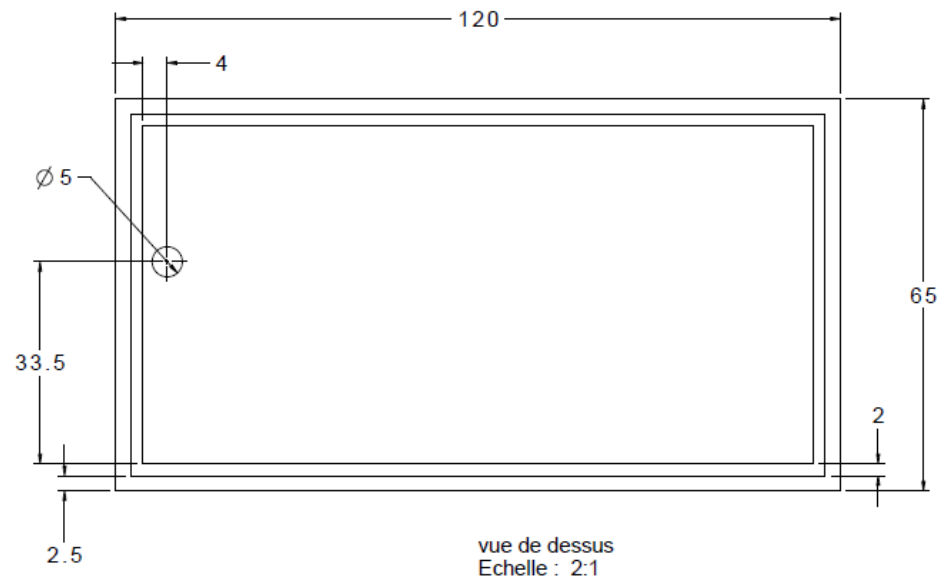
ID	Description	Réf. vers une section de ce document	Notes
D1	Structure globale	1.1	
D3	Paradigmes en réalité augmentée	1.2	
D4	Paradigme Stroop en réalité augmentée	1.2.2	
D5	Paramètres du jeu Stroop	1.2.2.1	
D6	Descriptions des scènes et pages du jeu Stroop	1.2.2.2	
D7	Séquence et design des variables du Stroop	1.2.2.3	
D22	Description des difficultés du Stroop	1.2.2.4	
D8	Enregistrement des réponses du participant pour le test Stroop	1.2.2.5	
D23	Plan de la structure du code et fonctions Stroop	1.2.2.6	
D9	Test N-back en réalité augmentée	1.2.3	
D10	Paramètres du jeu N-back	1.2.3.1	
D11	Modifications à effectuer du N-back	1.2.3.2	
D12	Descriptions des scènes et pages du jeu Stroop	1.2.3.3	
D13	Liste de matériel de synchronisation à utiliser	1.3.1	
D14	Mise en application de la synchronisation	1.3.2	
D15	Synchronisation des réponses	1.3.2.1	
D16	Synchronisation avec le NIRS et Arduino	1.3.2.2	
D17	Synchronisation du tapis roulant	1.3.2.3	
D18	Intégration physique	1.4	
D19	Intégration des casques NIRS et AR	1.4.1	
D20	Boîtier pour le Arduino	1.4.2	
D21	Guide de l'utilisateur	1.5	
D22	Paradigmes en réalité augmentée non retenus	2.1	
D23	Alternatif de synchronisation des signaux non retenus	2.2	
D24	Alternatif d'intégration des casques NIRS et AR non retenus	2.3	

6. RÉFÉRENCES

- Arduino. (s. d.). *Arduino Mega 2560 Rev3*. Arduino. Consulté 23 novembre 2021, à l'adresse <http://store-usa.arduino.cc/products/arduino-mega-2560-rev3>
- Artinis Medical Systems. (2021). *PortaSync MKII*. Artinis Medical Systems | FNIRS and NIRS Devices. <https://www.artinis.com/PortaSync>
- Braille Institute, (2019), Atkinson Hyperlegible. https://brailleinstitute.org/wp-content/uploads/2020/02/BIA_AtkinsonHyperlegible-Specimen_200210.pdf
- Brown, R. (s. d.). *Meta 2: Full Specification*. VRcompare. Consulté 4 octobre 2021, à l'adresse <https://vrcompare.com/headset/meta2>
- Dr Karen Li & Rachel Downey (2019). *Hearing and Walking study*.
- Full Vision Inc. (2004). *TRACKMASTER Treadmill Owner's Manual*. <https://www.manualslib.com/manual/624450/Trackmaster-Tmx425.html>
- New Biotechnology Ltd. (2021). *The Brite system—Completely wearable 24 channel fNIRS*. NBT. <https://nbt ltd.com/products/brite-completely-wearable-24-channel-fnirs/>
- Noraxon U.S.A Inc. (2013). *DTS Footswitch User Manual*. <http://www.noraxon.com/wp-content/uploads/2015/05/P-5008-Rev-G-DTS-Footswitch-Manual.pdf>
- The Black Box ToolKit. (2021). *Millisecond accurate hand-held USB response button key features*. <https://www.blackboxtoolkit.com/hhusbbutton.html>
- We are Colorblind. (2012, janvier 10). A Quick Introduction to Color Blindness. *We Are Colorblind*. <https://wearecolorblind.com/articles/a-quick-introduction-to-color-blindness/>

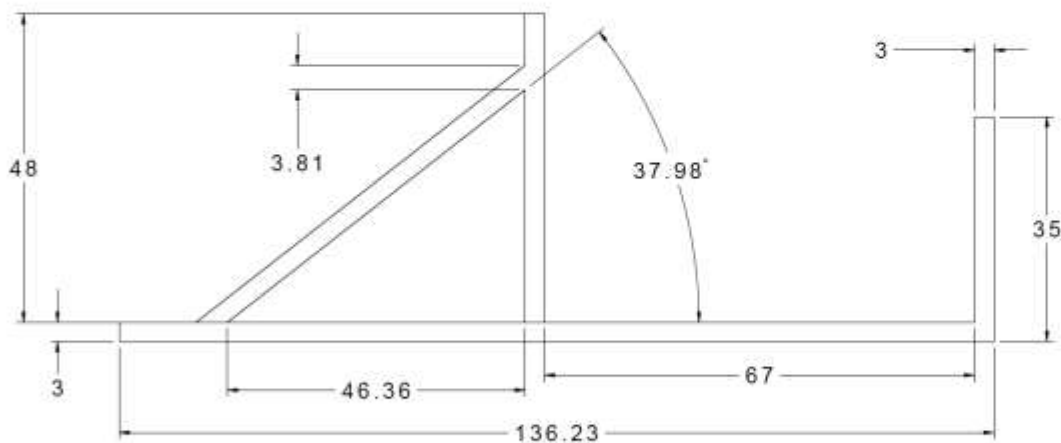
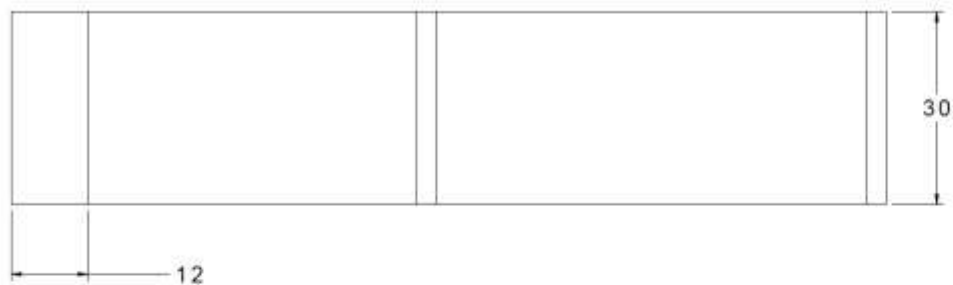
7. ANNEXE





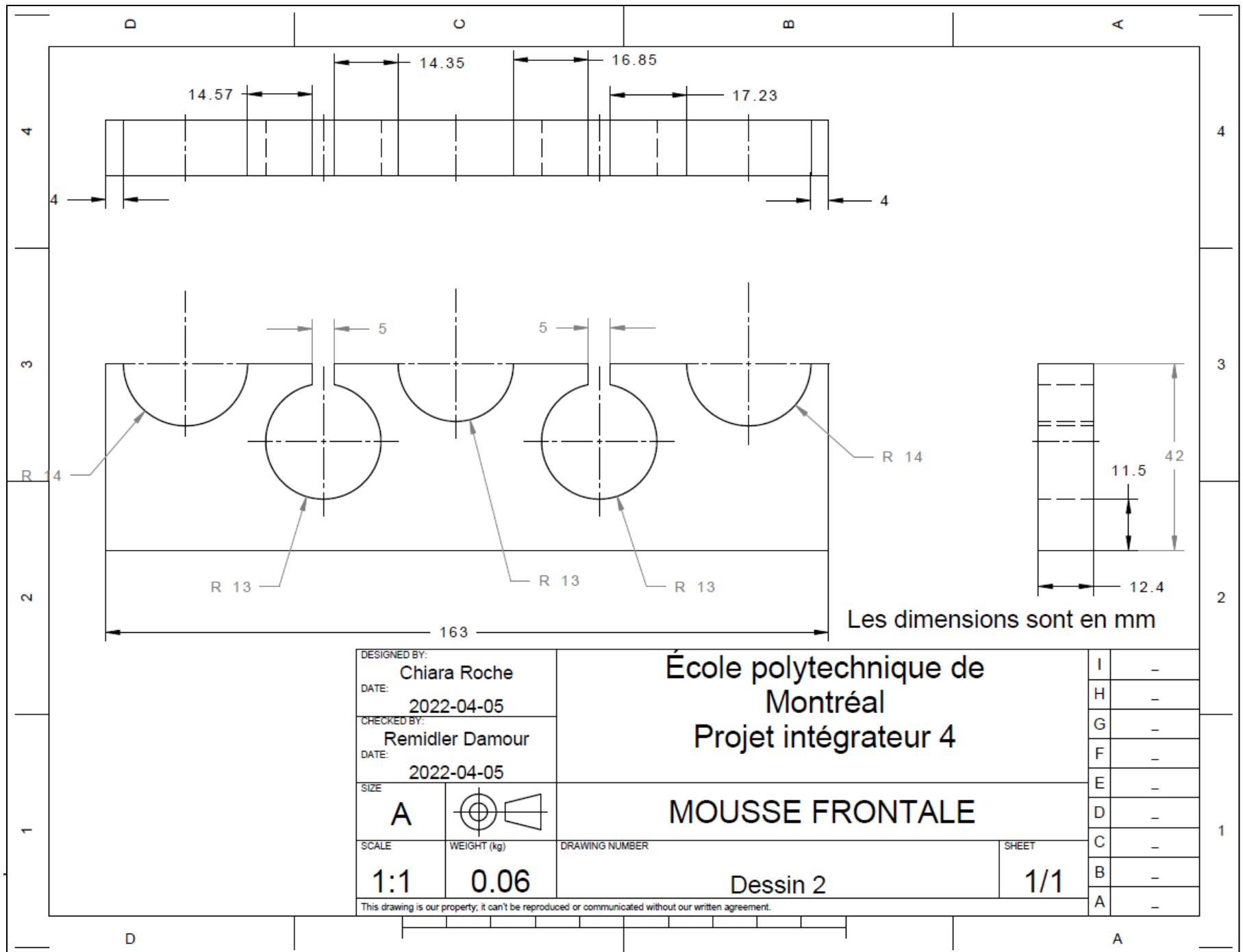
Les dimensions sont en mm

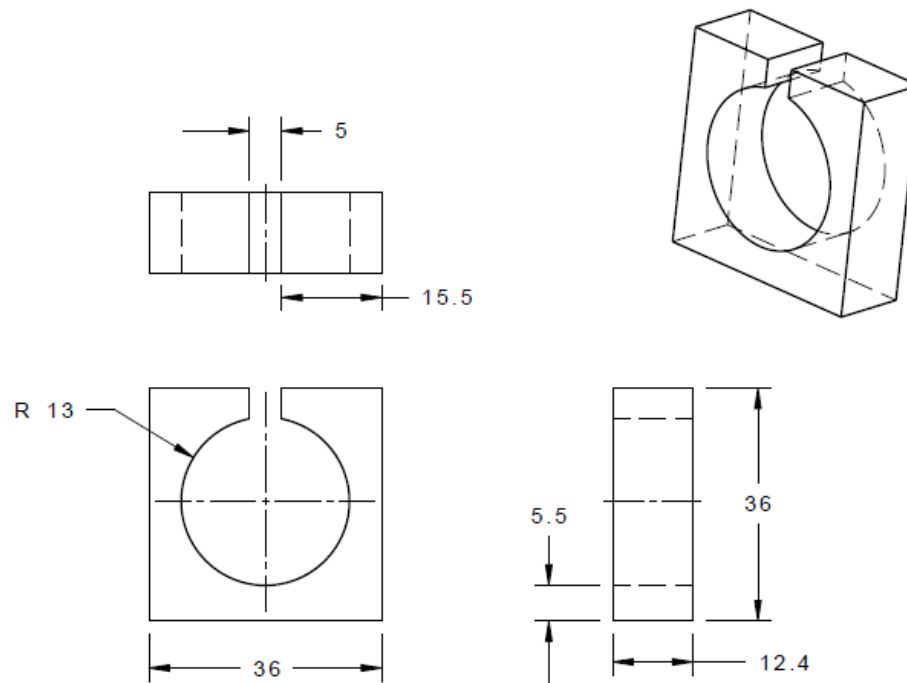
Dessiné par Remidier Damour		École polytechnique de Montréal Projet intégrateur 4		I	...
DATE 2022-04-05				H	...
Dessiné par Kayla Rezendes		COUVERCLE DE BOITIER		G	...
DATE 2022-04-05				F	...
C			Dessin 1	E	...
				D	...
				C	...
				B	...
2:1	0.02			A	...



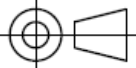
Les dimensions sont en mm

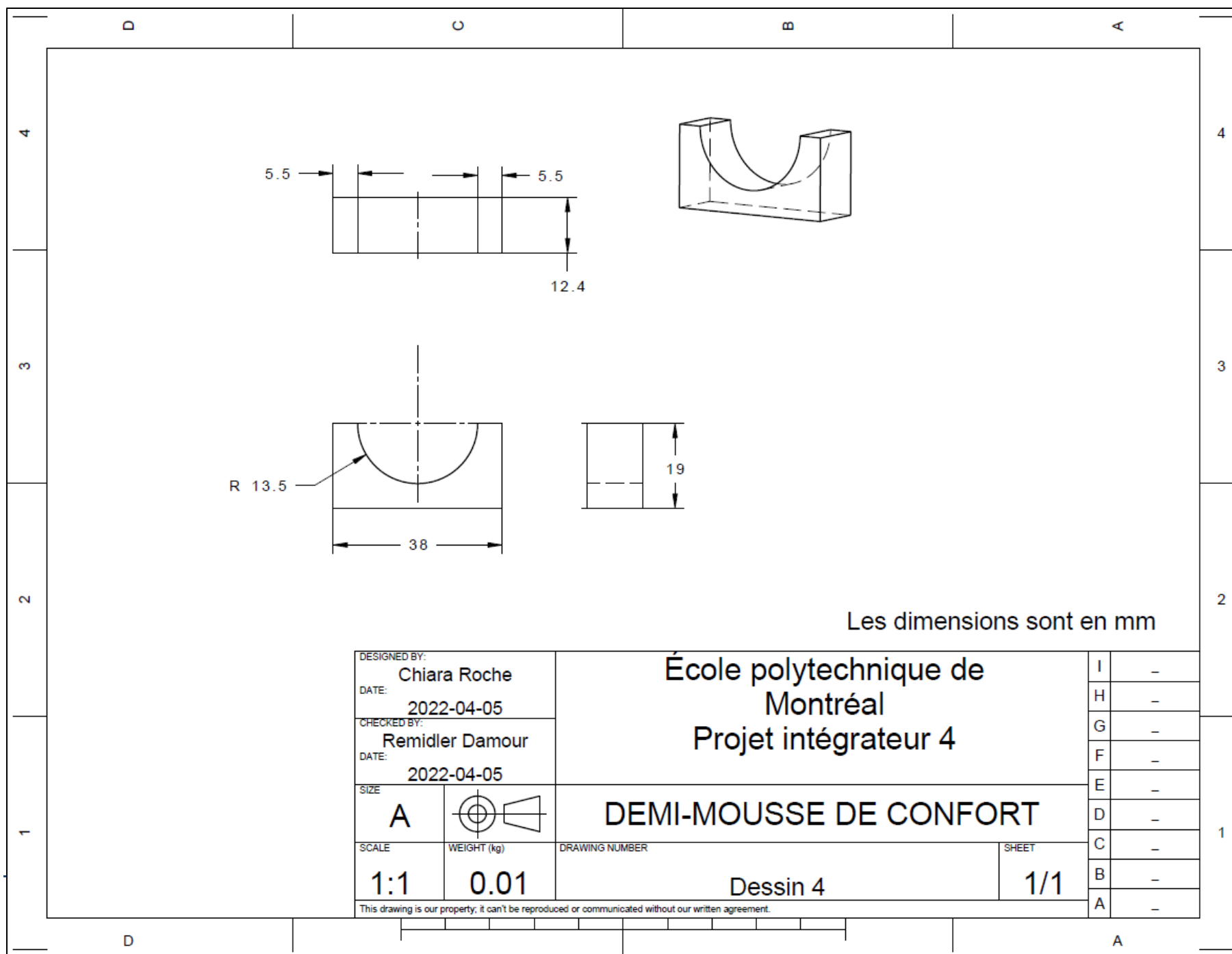
DESIGNÉ PAR Remidier Damour DATE 4/5/2022		École polytechnique de Montréal Projet intégrateur 4		I	-
CHARGÉ DE PROJET Kyla Rezendes DATE 2022-04-05				H	-
SCALE C		SUPPORT DU BOITIER ARDUINO	G	-	
			F	-	
E	-				
D	-				
C	-				
B	-				
A	-				
2:1 0.03			Dessin 5	1/1	





Les dimensions sont en mm

DESIGNED BY: Chiara Roche		École polytechnique de Montréal Projet intégrateur 4		I	–
DATE: 2022-04-05				H	–
CHECKED BY: Remidler Damour				G	–
DATE: 2022-04-05				F	–
SIZE A		MOUSSE DE CONFORT		E	–
SCALE 1:1	WEIGHT (kg) 0.01			D	–
		DRAWING NUMBER Dessin 3	SHEET 1/1	C	–
				B	–
				A	–
This drawing is our property; it can't be reproduced or communicated without our written agreement.					



HISTORIQUE DES VERSIONS

Version	Date	Détails	Auteur(s)
1.0	2021-24-11	Version initiale	Tous
2.0	2021-30-11	Version modifiée	Tous
3.0	2021-02-12	Version modifiée	Tous