Développement d'un objet connecté
Immersion 3.5D

Réalisé par

KADDOUH Heba – BENSAAD Mohamed – COLOMBO Jessy – BARAKAT Khalid

Encadré par

M. Aomar OSMANI & M. Hamidi MASSINISSA

Remerciements

Avant tout développement, nous profitons de ce rapport pour remercier ceux qui nous ont beaucoup appris et ceux qui ont eu la gentillesse de faire de cette expérience un moment profitable.

Nous tenons tout d'abord à remercier Monsieur Aomar OSMANI, qui nous a aidé et encadré tout au long du projet avec beaucoup de patience et de pédagogie. Ainsi que Monsieur Hamidi MASSINISSA, qui est resté disponible pour toutes nos questions et qui a su nous conseiller sur nos différents choix ainsi que sur la méthodologie à prendre.

Table des matières

Présentation générale

Présentation de notre application

Le but de notre produit est une immersion dans une expérience multimédia s'approchant de la technologie de la 4DX à moindre coût. Pour cela nous devons créer un set-up adaptable sur toutes les structures (télévisions, ordinateurs...). Le set-up est composé de la façon suivante :

- **LED** : Il s'agit d'une bande LED connectée à votre smartphone en Bluetooth que vous pourrez contrôler à votre guise. Mais la particularité de cette bande est son mode immersif : Elle s'adaptera automatiquement aux couleurs prédominantes, que ce soit pour un jeu vidéo, un film ou juste une image, l'expérience est garantie.
- **Application** : L'application permettra de contrôler la bande LED. Elle pourra allumer ou éteindre la bande, activer les différents modes de celle-ci, augmenter ou diminuer la lumière ou bien changer la couleur de la bande.
- **Fauteuil** : Pour le moment, il s'agit d'une évolution du set-up qui permettrait une meilleure immersion quel que soit l'action. Pour cela deux enceintes connectées à votre périphérique en Bluetooth ou en filaire amèneront le son au plus proche de soi. Aux cotées de ces enceintes, nous allons intégrer des capteurs sonores liés à notre ESP32 qui permettront d'analyser le son. Après analyse, des vibrations seront activé en fonction de l'action. Par exemple, si le bruit d'une balle de pistolet est entendu, les moteurs vibrants situés sur le dos de la chaise s'activeront.

La finalité du projet est la création d'un prototype fonctionnel pouvant par la suite s'améliorer avec de plus grands moyens financier et une focalisation exclusif sur celui-ci.

Les enjeux

L'enjeu du projet qui nous a donné la motivation de le réaliser est d'ajouter une touche d'immersion dans le visionnage de diverses multimédias.

Besoin de notre projet

Dans notre cahier de charges on a défini les besoins comme suite :

LEDs connectées

Fonction	S'accorder en fonction de la couleur de l'écran TV
Objectif	S'éclairer de la couleur dominante de l'écran
Description	La bande LED sera munie d'un capteur de lumière qui lui transmettra les couleurs de l'écran afin qu'elle change de couleur
Contraintes	Le capteur de lumière doit être placé correctement par rapport à l'écran
Niveau de priorité	Haut

Fonction	S'accorder en fonction des couleurs définies par l'utilisateur
Objectif	S'éclairer de la couleur choisie par l'utilisateur
Description	La bande LED sera connectée à l'application et devra changer de couleur en fonction de ce qui est sélectionné dans l'application.
Contraintes	
Niveau de priorité	Haut

Fonction	Se connecter au périphérique
Objectif	Pouvoir contrôler la bande LED
Description	Le téléphone sera connecté en Bluetooth à la bande LED
Contraintes	La distance entre les deux objets connectés doit être raisonnable
Niveau de priorité	Haut

Fonction	Contrôler la bande LED à distance
Objectif	Permettre à l'utilisateur de changer la couleur de la bande
Description	L'application permettra de changer de couleur, allumer et éteindre les LEDs
Contraintes	L'application doit avoir la priorité sur le capteur de lumière
Niveau de priorité	Haut

Fonction	Capter le son de la TV
Objectif	Récupérer le signal sonore délivré par le son de la TV
Description	Le fauteuil récupérera le son
Contraintes	Nuisances sonores
Niveau de priorité	Moyen

Fonction	Réagir en fonction du son
Objectif	La housse devra vibrer en fonction de l'intensité du son
Description	Des moteurs vibrants intégrés à la housse réagiront en fonction du son
Contraintes	
Niveau de priorité	Haut

Conception architecturale

Choix des langages de programmations

Notre application mobile est codée en JAVA pour les activités et classes, et en XML pour le design. Pour cela nous utilisons Android Studio.

Pour l'Arduino et le capteur, nous utilisons du C++ (.ino) à l'aide du logiciel Arduino IDE.

Maquette de notre application

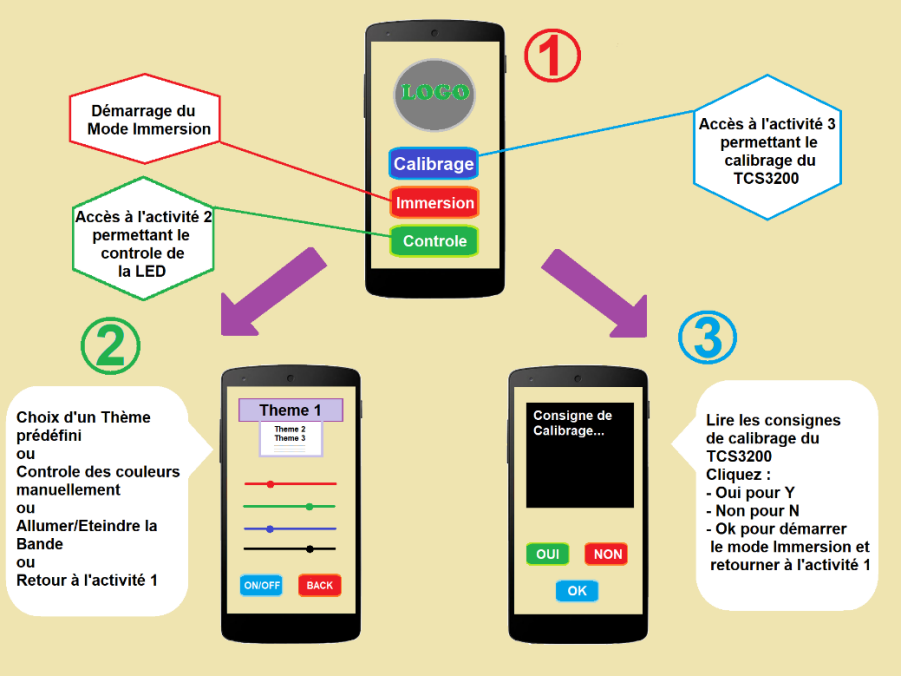
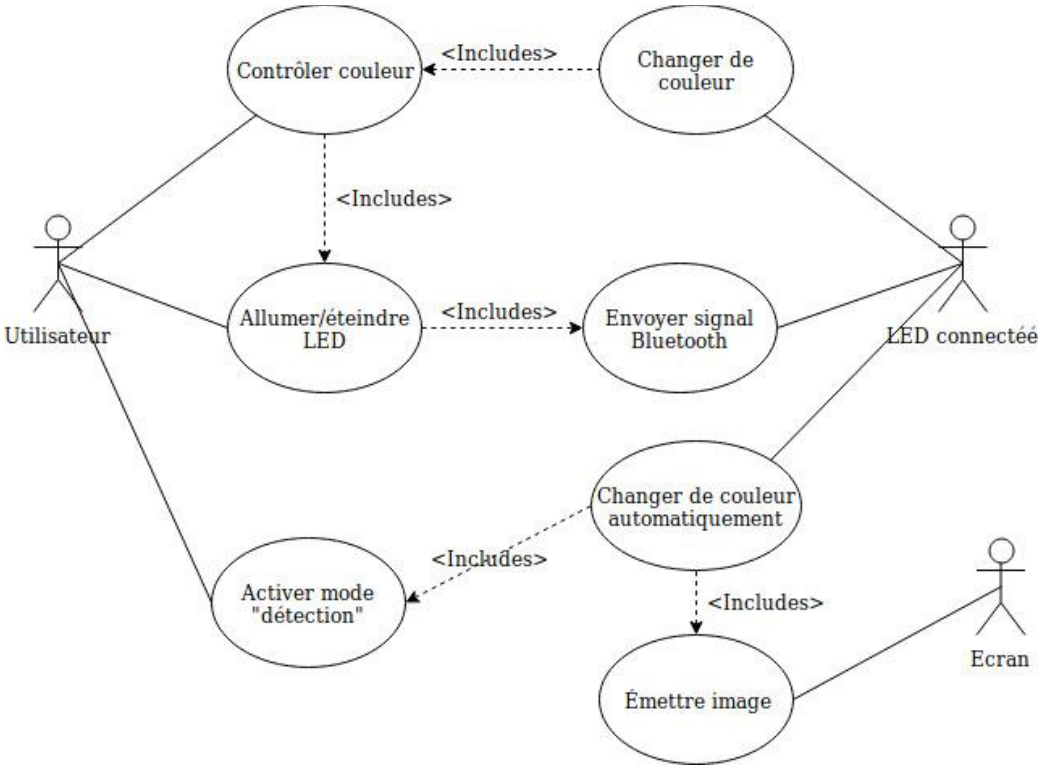


Diagramme des cas d'utilisation



Un utilisateur pourra donc allumer et éteindre la LED, contrôler les couleurs seulement si la LED est allumée et activer le mode « détection » comportant son calibrage.

Gestion de projet

Planification et environnement de travail

Afin de réussir le projet et respecter la date du rendu, il nous a fallu une très bonne gestion et pour cela nous avons opté pour deux méthodes :

La première est l'organisation des membres de l'équipe, d'où le choix d'un chef de projet (Mohamed BEN SAAD), d'une équipe pour l'application mobile (Heba KADDOUH & Jessy COLOMBO) et d'une autre pour l'Arduino (Mohamed BEN SAAD & Khalid BARAKAT). Pour faciliter la communication nous avons créé un groupe WhatsApp qui nous permettait d'organiser des réunions (le jeudi matin ainsi que les mardis à 17h00 après le cours d'IOT, sans oublier, les créneaux du module), de s'entraider, de faire un échange d'informations et de faire une mise au point.

De plus, notre chargé de TD nous a conseillé de créer un compte GitHub afin de pouvoir suivre toutes les modifications du code et y avoir accès en cas de problème technique, que nous avons utilisé sans modération.

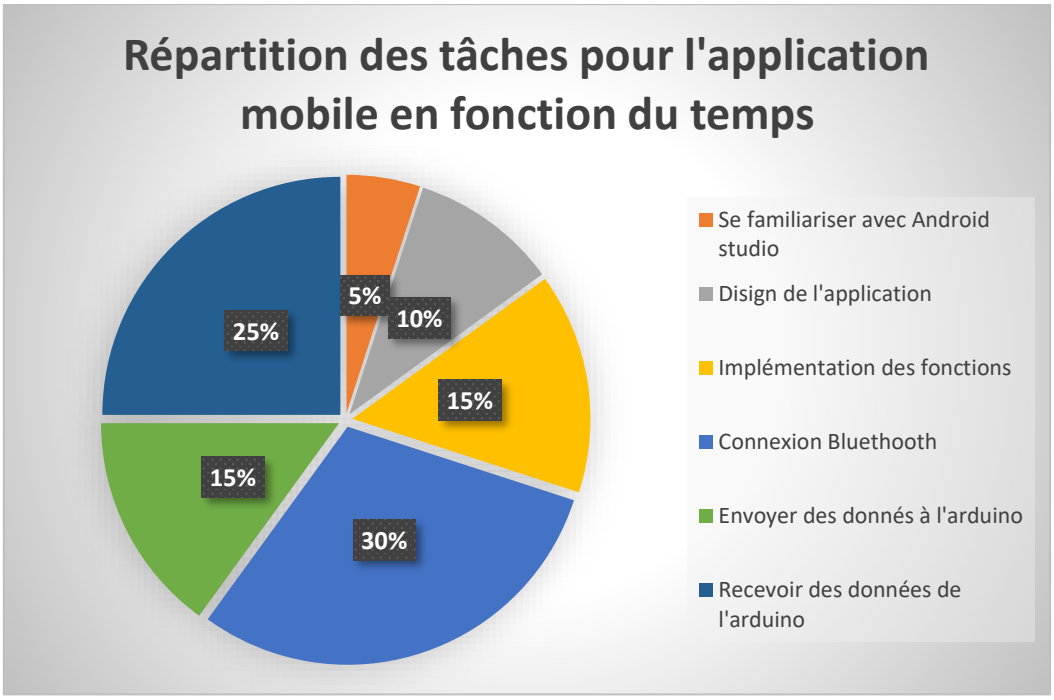
Nous avons mis en place sur GitHub dans l'onglet Projects, un planificateur de toutes les tâches que nous allons accomplir, que nous avons accomplies et que nous sommes en train d'accomplir. Ce planificateur nous a permis de se retrouver et de voir ce qu'il restait à faire. On peut aussi y retrouver les liens utiles, les composants et les logiciels utilisés.

Répartition des tâches

Comme nous l'avons dans la partie précédente, nous avons réparti le travail en deux équipes.

Nous avons tous contribué à la conception architecturale en réalisant le diagramme de Gantt et du cas d'utilisation qu'on peut retrouver sur le GitHub, ainsi qu'à la rédaction du cahier des charges, de l'étude de l'existant, du rapport et de la conception du diaporama de présentation.

Pour avoir un meilleur aperçu du temps accordé pour chaque tâche, nous avons réalisé deux diagrammes : pour l'application mobile et pour l'Arduino.


Répartition des tâches pour l'application mobile en fonction du temps
5%
10%
15%
30%
15%
25%
Se familiariser avec Android studio
Disign de l'application
Implémentation des fonctions
Connexion Bluethooth
Envoyer des donnés à l'arduino
Recevoir des données de l'arduino

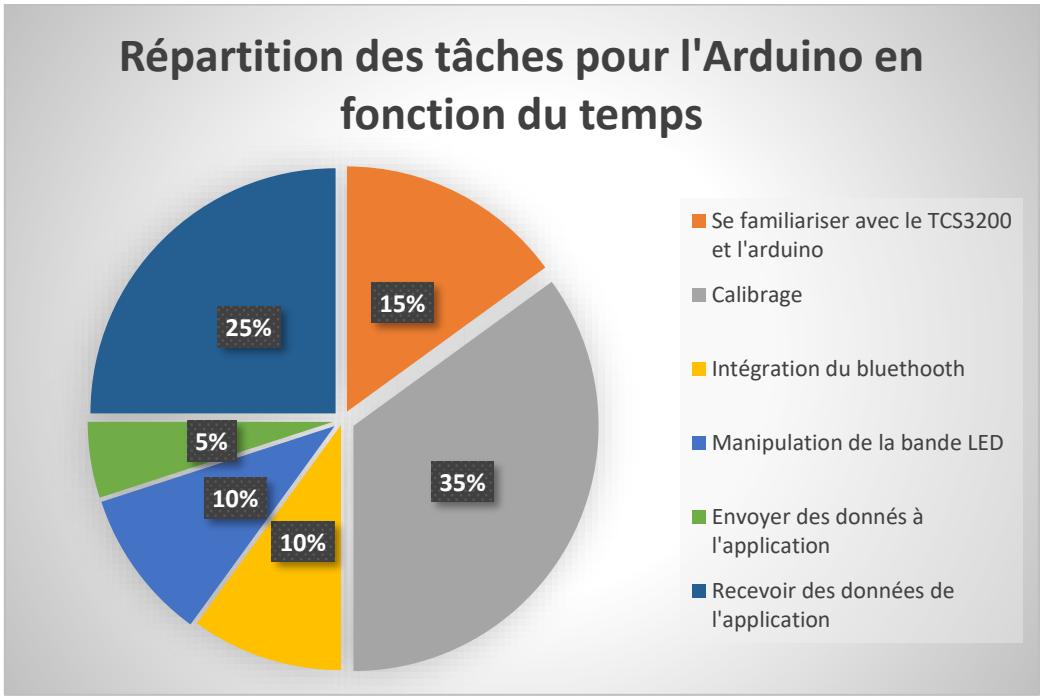
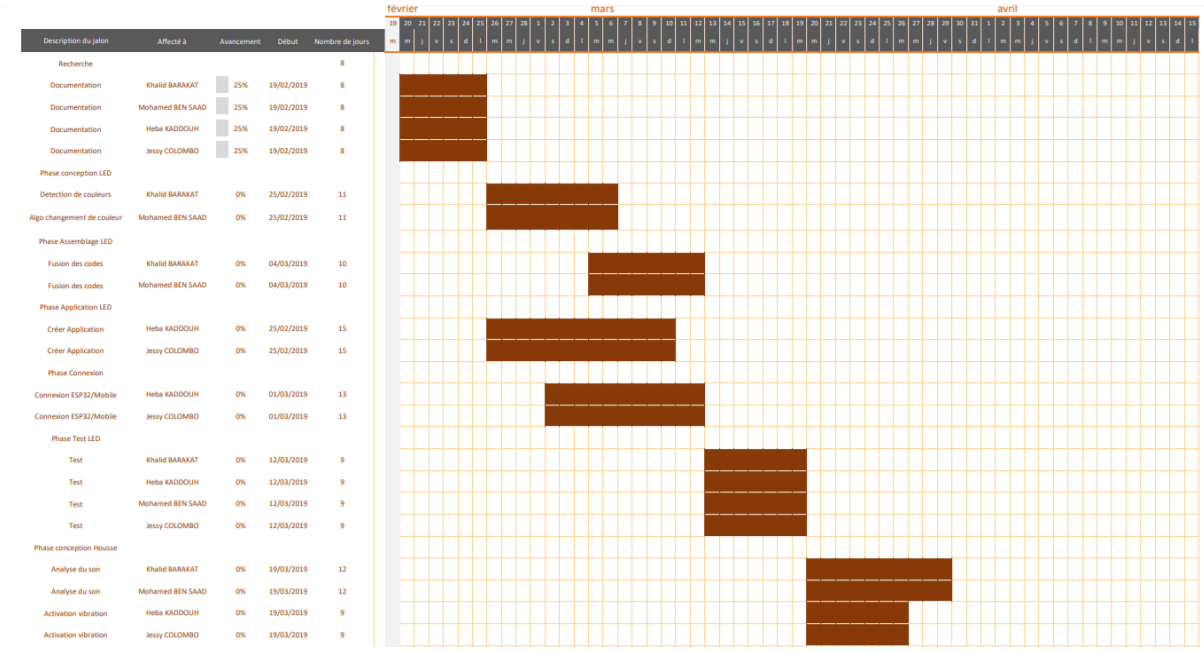
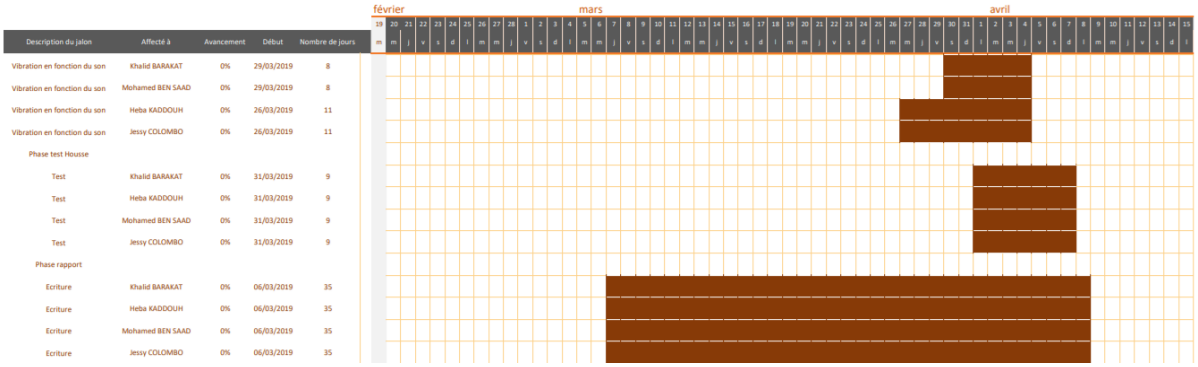

Répartition des tâches pour l'Arduino en fonction du temps
15%
35%
10%
10%
5%
25%
Se familiariser avec le TCS3200 et l'arduino
Calibrage
Intégration du bluethooth
Manipulation de la bande LED
Envoyer des donnés à l'application
Recevoir des données de l'application

Diagramme de Gantt

Le diagramme de GANTT est un outil efficace permettant de générer une visualisation de l'avancement du projet. Il permet de donner une vue globale des tâches à réaliser et le du temps associé.





Nous avons réalisé le diagramme de Gant dans l'optique de l'accomplissement total du projet. Sauf que nous avions sous-estimé la complexité de celui-ci. Nous détaillerons tous nos changements par la suite.

Développement du Set-Up

Arduino

Au départ nous avons décidé d'utiliser une ESP32 pour notre projet. Malheureusement, après plusieurs essais de couplage de celle-ci avec notre capteur TCS3200, nous avons remarqué que l'architecture de l'ESP32 n'était pas compatible. Il aura fallu tout implémenter nous même pour l'adapter avec le capteur. Nous avons donc jugé plus judicieux de passer à une Arduino vu le temps qui nous était imparti. Ce choix n'a pas été anodin car nous avons trouvé plusieurs codes fonctionnels sur Arduino. Cela nous a permis de comprendre et de nous familiariser avec le TCS3200.

Capteur TCS3200

Ces capteurs sont conçus pour détecter les couleurs d'objets et non celles d'un écran. Nous avons fait un pari risqué en le choisissant car nous ne savions pas s'il répondrait à nos attentes. Son faible prix nous a aidé à sauter le pas, nous réconfortant sur la faible perte qu'il aurait pu présenter.

Face droite

Face Avant

Face Gauche

Notre premier objectif a été de faire fonctionner notre capteur avec l'Arduino. Le branchement a été rapidement maîtrisé grâce à quelques exemples trouvés au préalable.

Un fois le montage opérationnel, nous avons testé notre capteur avec plusieurs codes comme celui fourni avec le manuel électronique du TCS3200 et d'autres trouvés sur le Web. Cependant, comme le capteur n'est pas destiné aux écrans, la plupart de nos tests ont malheureusement échoué. À ce stade, nous avons réussi à reconnaitre les couleurs primaires (rouge, vert et bleu) avec des valeurs non conformes à nos attentes allant de -10 000 à 10 000. Nous devions remédier à ce problème et trouver une fonction qui nous permettrait de calibrer le capteur sur tous types d'écrans ainsi que de ramener les valeurs reçues entre 0 et 255. Nous avons essayé plusieurs modèles mathématiques pour moduler les valeurs captées, mais les codes utilisés n'étaient toujours pas dans l'intervalle voulu pour notre projet.

Après de nouvelles recherches, nous avons découvert une bibliothèque sur GitHub : https://github.com/blascarr/TCS3200-ColorSensor avec « TSC3200.h », implémenté spécialement pour le modèle de capteur que nous utilisons, incluant une fonction de calibrage optimal. Nous avons analysé cette bibliothèque pour pouvoir la modifier et l'adapter à nos nécessités. Une fois accordée à nos besoins, le résultat a répondu à toutes nos espérances. Pour calibrer le TCS3200, il suffit juste de poser le capteur sur l'écran avec la suite de couleur demandées lors de l'exécution de la fonction « calibration() ». Toutes les instructions sont affichées sur le Serial de l'Arduino (Terminal). Cependant, une source de lumière externe peut fausser celui-ci.

Par la suite, nous avons intégré la bande LED « Pixel Strip » à l'Arduino. A l'aide de sa bibliothèque originale « Adafruit_NeoPixel.h », nous avons pu lui envoyer les valeurs perçues par le capteur. Après quelques optimisations sur la synchronisation entre le TCS3200, l'Arduino et la bande LED, le Mode Immersion venait de voir le jour.

Bluetooth HC-05

Le choix du HC-05 a été fait en fonction de son prix, de sa popularité et de sa bonne notation. À priori l'achat était sûr.

Il existe deux types de branchements, l'un est pour faire communiquer l'Arduino avec un autre objet connecté, le deuxième sert à paramétrer le module Bluetooth, par exemple donner un nouveau nom au HC-05 pour mieux l'identifier.

Branchement avec KEY pour configurer le Bluetooth

Pour vérifier que tout fonctionne comme il faut, il envoie « AT » vers l'Arduino. Si on obtient la réponse « OK », alors tout fonctionne et maintenant on peut utiliser les commandes de l'AT-MODE. Pour renommer le Bluetooth on rentre maintenant la commande AT-NAME = <nouveau_nom>.

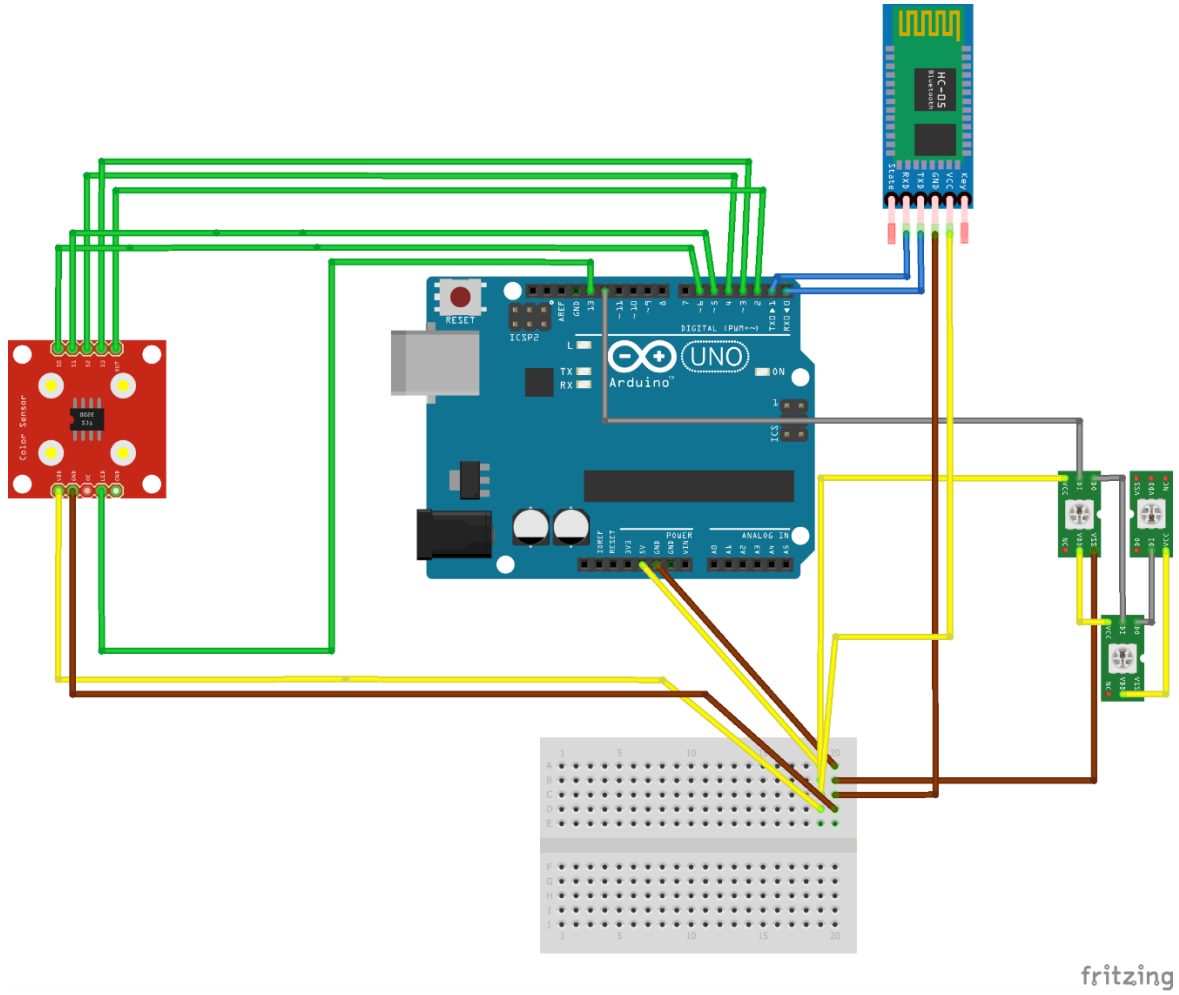
Branchement normal pour communication

On envoie les données sur les ports TX/RX de l'Arduino. Cela facilité l'implémentation du code car il n'est plus nécessaire d'ouvrir un nouveau Serial pour transférer des données. Il faut noter que le TX du HC-05 se branche sur le RX de l'Arduino et le RX sur le TX. La communication est très importante pour le calibrage du TCS3200, étant donné que chaque modèle d'écran a des caractéristiques de couleur et de luminosité différente. Il faut bien suivre les instructions lors du calibrage du capteur pour ne pas avoir des erreurs sur les couleurs affichées ou leur intensité.

Montage final

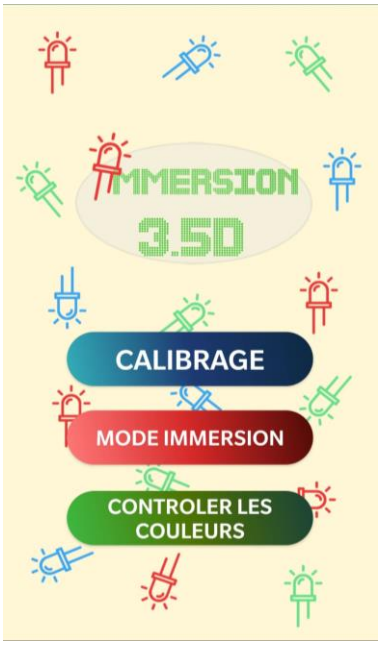
Une fois les deux parties précédentes accomplies, nous avons assemblé toutes les fonctionnalités dans un même code pouvant contrôler tous les composants de notre objet connecté ainsi que de toutes nouvelles fonctionnalités. C'est-à-dire, la communication Android/Arduino, le contrôle de la bande Led manuellement et les différents modes de celui-ci.

Voici le montage final de celui-ci :

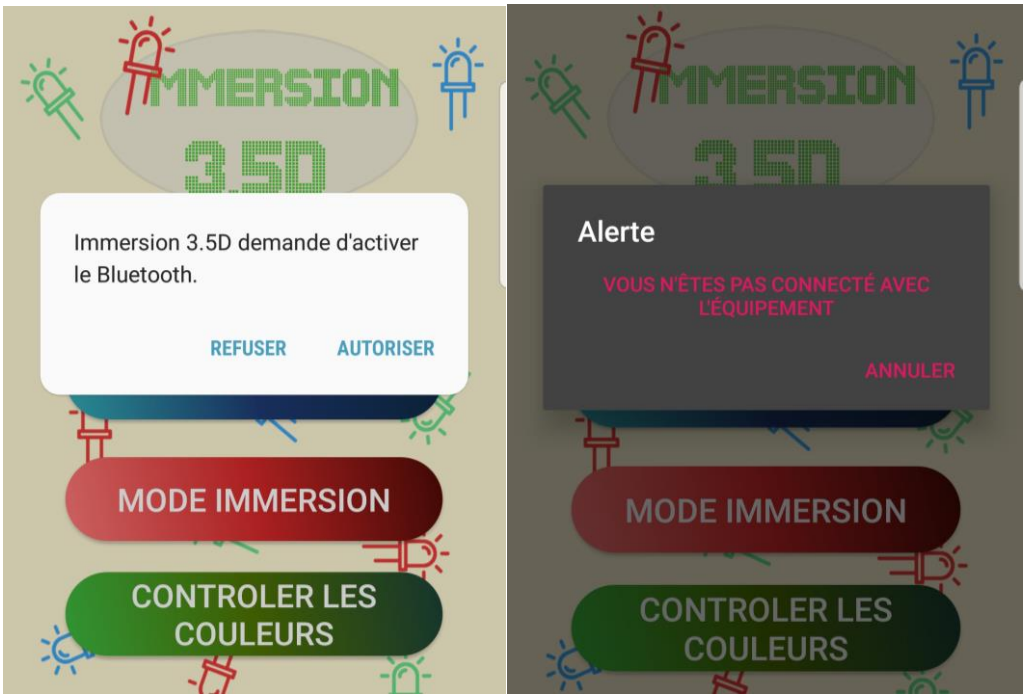


Composants	Port objet	Port Arduino
TCS3200	VCC	5V
	S3	3
	S2	4
	OUT	2
	GND	GND
	S1	5
	S0	6
	LED	13
HC-05	GND	GND
	VCC	5V
	RXD	TX 1
	TX	RX 0
LED	5V	5V
	GND	GND
	Din	12

Page d'accueil



Lorsque l'utilisateur lance l'application, il se retrouve sur cette page et une connexion BluetoothSocket (bSocket) est lancée. La connexion Bluetooth est établie grâce à notre fonction « bluetooth_connect_device()». Si la connexion n'est pas établie, une fenêtre pop-up s'affiche « Vous n'êtes pas connecté avec l'équipement » et l'interface ne peut pas être utilisée. Et si la connexion Bluetooth n'est pas activée alors une demande d'activation de Bluetooth est demandé.



Au lancement de l'application, on envoie la lettre « O » par Bluetooth à l'Arduino qui permet de la mettre en mode On/Off.

A partir de cette page l'utilisateur peut, au click :

- Calibrer le TCS3200, en envoyant la lettre « C » par Bluetooth à l'Arduino et passe à la page de calibrage.
- Entrer en mode immersion en envoyant la lettre « I » par Bluetooth à l'Arduino et passe en mode immersion tout en restant sur cette page.
- Contrôler les couleurs en envoyant la lettre « T » par Bluetooth à l'Arduino et passe à l'interface de contrôle de la bande de leds.

L'envoi de caractères à l'Arduino via Bluetooth se fait comme suit : « bSocket.getOutputStream().write("O".getBytes()); »

Page de calibrage

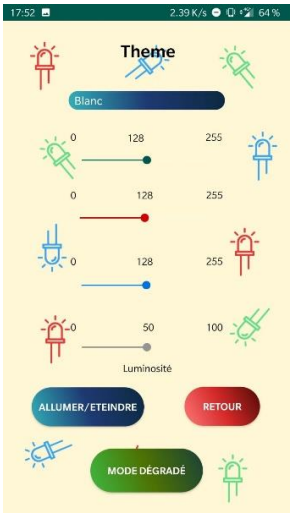
Lorsque l'on sélectionne le mode « calibrage », nous avons un champ de texte qui contient tout ce qui est renvoyé par l'Arduino et trois boutons :

- Oui, qui envoie la lettre « Y » à l'Arduino
- Non, qui envoie la lettre « N » à l'Arduino
- Mode immersion, qui envoie la lettre « O » à l'Arduino

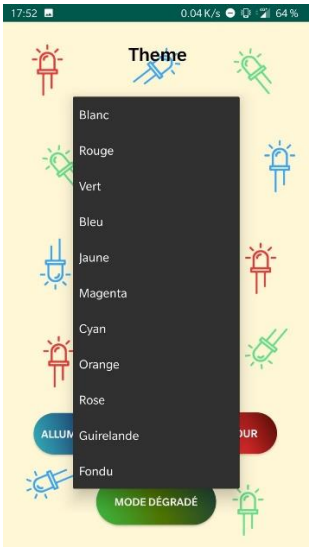
L'une des difficultés que nous avons eues avec cette page est de lire les données envoyées par l'Arduino. Pour pallier ce problème nous avons décidé de faire un thread qui reçoit, en continu et parallèlement du onCreate, les données grâce à la socket « bSocket » définie dans la page précédente. On dispose d'une boucle while permettant de lire les données envoyées par l'Arduino et de les afficher dans le champ de texte.

Page de contrôle

En appuyant sur le bouton « contrôler les couleurs » depuis la page d'accueil, la page se transforme en un panneau de contrôle (représenté ci-dessous) contenant trois jauges pour configurer les couleurs selon le rouge, le vert et le bleu, et une quatrième jauge pour régler la luminosité. Grâce à la fonction « onProgressChanged », à chaque fois que l'utilisateur déplace un curseur nous stockons la valeur de la jauge que nous envoyons à l'Arduino.

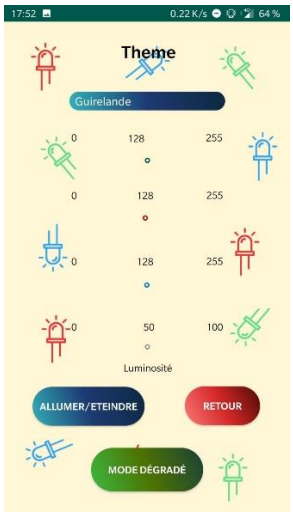


L'utilisateur a la possibilité de sélectionner des couleurs prédéfinies dans le menu déroulant se situant en haut de l'écran.



Lorsque l'utilisateur sélectionne un thème un caractère est envoyé à l'Arduino afin qu'il différencie les différents thèmes. Lorsque le thème « Guirlande » ou « Fondu » est activé, les jauges se désactivent. Pour chaque thème, on envoie le caractère « T » suivi des valeurs des couleurs sauf pour «Guirlande » et « fondu » :

- « G » pour guirlande
- « F » pour fondu



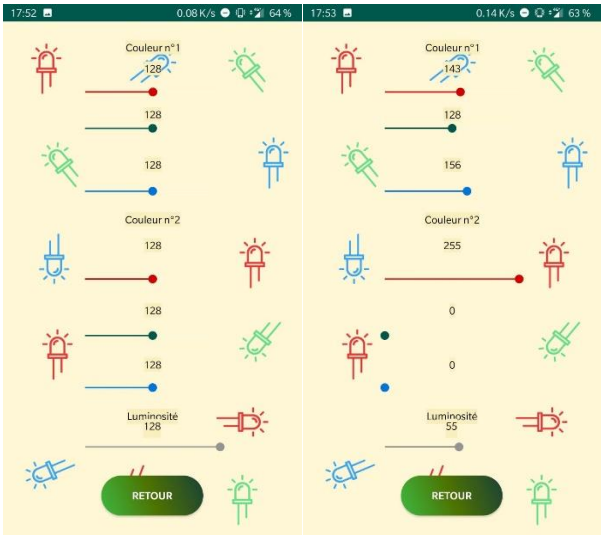
L'interface dispose également de trois boutons : « allumer/éteindre » pour allumer ou éteindre la bande de LEDS, « retour » pour revenir à la page d'accueil et « mode dégradé » menant à un nouveau panneau de contrôle servant cette fois-ci à configurer deux couleurs pour en faire le dégradé.

Là aussi des caractères spécifiques sont envoyés pour chaque bouton :

- « A » pour allumer
- « E » pour éteindre
- « D » pour le mode dégradé

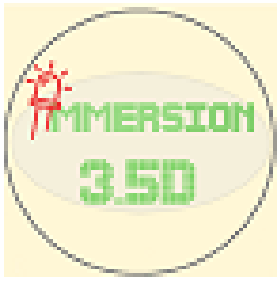
Mode dégradé

Le mode dégradé offre la possibilité de choisir deux couleurs, permettant ensuite à la bande de LEDS de s'illuminer du dégradé de la couleur n°1 à la couleur n°2. Comme pour la page précédente, on envoie les couleurs des différentes jauges ainsi que la luminosité à l'Arduino.



Logo

Pour réaliser ce logo nous avons utilisé Paint3D.



Contraintes/Difficultés

Choix de la carte électronique : Nous avons au début opté pour une ESP32 qui n'était pas compatible avec l'architecture de notre capteur.

Contraintes du capteur : environnement sombre et avoir juste l'écran comme source de lumières

Lecture et écriture de données : Nous avons eu du mal à :

- Lire les données envoyer par l'Arduino
- Envoyer des données à l'Arduino

Au vu de la complexité du projet et le manque de temps lié aux autres cours et projets, nous n'avons pas pu réaliser le fauteuil qui était une extension de notre projet.

Bilan du projet

Ce projet était d'une grande envergure sur l'aspect fonctionnel et structurel mais il a été formateur et enrichissant pour nous, en effet l'idée du projet était assez compliquée et pour la mettre en œuvre il nous a fallu beaucoup de recherche et de persévérance. On a pu s'auto former sur les différents langages utilisés et mettre en pratique nos précédentes connaissances comme UML pour réaliser les différents diagrammes de conception. Ce projet était aussi une occasion de découvrir le vrai travail d'équipe.

Annexe

Mode d'emploi

Pour utiliser l'application, l'utilisateur doit d'abord appairer en Bluetooth son smartphone Android à l'Arduino. Une fois connecté il pourra effectuer les actions suivantes :

- Calibrer
- Contrôler les couleurs
- Lancer le mode automatique

Lorsque l'utilisateur clique sur « Calibrage », il reçoit une demande de confirmation. Si l'utilisateur clique sur « non », l'Arduino va charger les réglages stockés dans la mémoire. En revanche si l'utilisateur appuie sur le bouton « oui », le calibrage rapide commence, il faut alors mettre le capteur sur du Blanc et du Gris. En effet les écrans émettant toujours de la lumière. Le noir ne sera jamais vraiment noir donc un gris sombre est alors le meilleur choix. A chaque étape, une confirmation de l'utilisateur est requise dans le cas où il n'était pas prêt à capturer la couleur demandée. Une fois cela fait, un message s'affiche demandant à l'utilisateur s'il veut remplacer les réglages par défaut par la capture qui vient d'être effectuée. Pour finir la première étape, un nouveau message s'affiche demandant si l'utilisateur veut aussi calibrer les couleurs. S'il appuie sur « non », le calibrage prend fin et affiche les valeurs enregistrées. A l'opposé si l'utilisateur lance la deuxième étape, des indications vont s'afficher pour calibrer sur les couleurs suivantes : bleu, jaune, marron, orange, rouge et vert, en suivant la même procédure que lors de la première étape. Pour terminer, un dernier message demande si l'utilisateur veut sauvegarder le calibrage dans la mémoire. Une fois la réponse reçue, sur le terminal sont affichées les valeurs capturées de chaque couleur lors du calibrage. Les valeurs affichés pour une couleur sont d'abord le rouge puis le vert puis le bleu et enfin la luminosité.

Pour un bon calibrage voici une table avec les intervalles de valeurs à obtenir pour chaque couleur :

	Rouge	**Vert**	**Bleu**
Blanc	> 250	> 250	> 250
Noir	< 15	< 15	< 15
Jaune	> 245	> 245	< 30
Orange	> 220	100 - 200	< 75
Rouge	> 220	< 40	< 40
Vert	< 40	> 220	< 40
Bleu	< 40	< 40	> 220
Marron	100 - 180	< 90	< 10

Lorsque l'utilisateur clique sur « Mode immersion », la bande LED change de couleur en fonction de celle perçue par le capteur. L'utilisateur aura juste à placer le capteur devant un écran.

Lorsque l'utilisateur clique sur « Contrôler les couleurs », il est amené sur une autre page qui lui permet de :

- Choisir des thèmes en cliquant sur le menu déroulant
- Changer les couleurs et la luminosité de la LED grâce aux jauges
- Eteindre ou allumer la LED en cliquant sur « Allumer/Eteindre »
- Retourner sur la première page en cliquant sur « Retour »
- Basculer en « Mode dégradé » qui nous amène sur une nouvelle page où il configurera alors deux couleurs.