INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

# Rapport du Projet Méthodologie pour le Numérique Durable

Projet Grand'EII

EII LIGHTING

Université de Rennes

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

# SOMMAIRE

# Présentation du projet - Objectifs

Ce projet est porté par BROGIALDI Maxime, DEVY Malo, EUDES Aloïs, HILKENS Boris et VU Giang.

Notre encadrant pour ce projet est M. Jacques MORIN.

Le projet que nous avons choisi consiste à la création d'un logiciel de gestion de lumière par commande infrarouge (IR). L'utilisateur peut ajouter des machines (lumières) en captant les codes IR de la télécommande d'origine, pour définir une action à chaque code.

Ce projet est venu d'une part d'une application domotique: il faut des télécommandes différentes pour chaque lumière ce qui peut poser des soucis de perte, de consommation de piles, ce n'est pas ergonomique et surtout rien n'est synchronisé entre différentes machines. Ce choix est aussi inspiré d'un logiciel que certains membres de l'équipe ont été amenés à utiliser dans le milieu associatif: GrandMA, qui est, en bref le logiciel leader du contrôle en direct d'éclairages pour le milieu de la scène et du spectacle.

Pour s'adapter au plus grand nombre de lumières possibles, le logiciel va gérer tout un système de couleurs et d'intensités pour s'approcher le plus possible du rendu désiré par l'utilisateur.

Le matériel nécessaire pour ce projet est:

- Le logiciel sur un ordinateur
- Une carte Arduino avec des émetteurs et récepteurs IR
- Des lumières pilotables par infrarouge (et leurs télécommandes)

Notre prototype a été réalisé en utilisant une Arduino MEGA 2560.

# Méthodologie

Nous avons fait le choix de travailler sous Linux (par une machine virtuelle ou Dual-Boot), afin de simplifier la gestion des exécutables et le lien avec la carte Arduino. Le projet a été versionné en utilisant un dépôt Git, sur le Gitlab fourni par l'INSA Rennes.

Nous avons fait le choix de structurer notre projet selon un modèle MVC (Modèle–Vue–Contrôleur), une architecture logicielle qui permet de séparer les responsabilités entre les différentes parties du code.

Dès le départ, trois pôles ont été définis, ou chaque pôle disposait d'une branche Git dédiée dans le dépôt partagé sur GitLab :

- Le pôle Vue : chargé de concevoir l'interface utilisateur graphique à l'aide de GTK et de la rendre interactive
- Le pôle Modèle : responsable de la définition des structures de données (lumières, couleurs, groupes, fonctions, etc.) et de l'implémentation de la logique métier associée.
- Le pôle Arduino : en charge de la communication matérielle, c'est-à-dire de l'envoi et la réception de signaux infrarouges entre la carte Arduino.

Le développement du projet s'est fait en 2 grandes parties:

La première, pendant laquelle une équipe s'occupait de créer la vue. Une autre équipe était chargée d'établir les premières structures de données, correspondant aux codes infrarouges, aux lumières, aux groupes, etc. Et une dernière équipe chargée d'implémenter l'envoi et la réception de codes IR par la carte Arduino et les capteurs.

La 2e étape, plus chronophage, était de réussir à lier tous les fichiers. Il a fallu d'abord lier la partie "vue" au contrôleur, pour faire en sorte d'appeler les fonctions lors de chaque appui - saisie sur la vue. Il fallait ensuite lier le contrôleur au modèle, pour exécuter les fonctions, et pouvoir émettre les codes infrarouges correspondant à l'action souhaitée.

# DÉVELOPPEMENT

| Dossier | View | Controller | Model |
|---|---|---|---|
| **Fichiers principaux:** | interface.ui<br>view.h<br>view_popup.h | controller<br>live_controller<br>popup_controller<br>light_wizard | list-commun<br>color<br>function<br>light<br>group<br>group_manager<br>json_parser- writer |
| **Résumé:** | Générer l'IHM | Récupérer les informations des boutons de l'IHM | Appliquer différentes fonctions sur les lumières / groupes |

Pour synthétiser les différentes structures établies et l'architecture de notre logiciel, voici un diagramme UML :

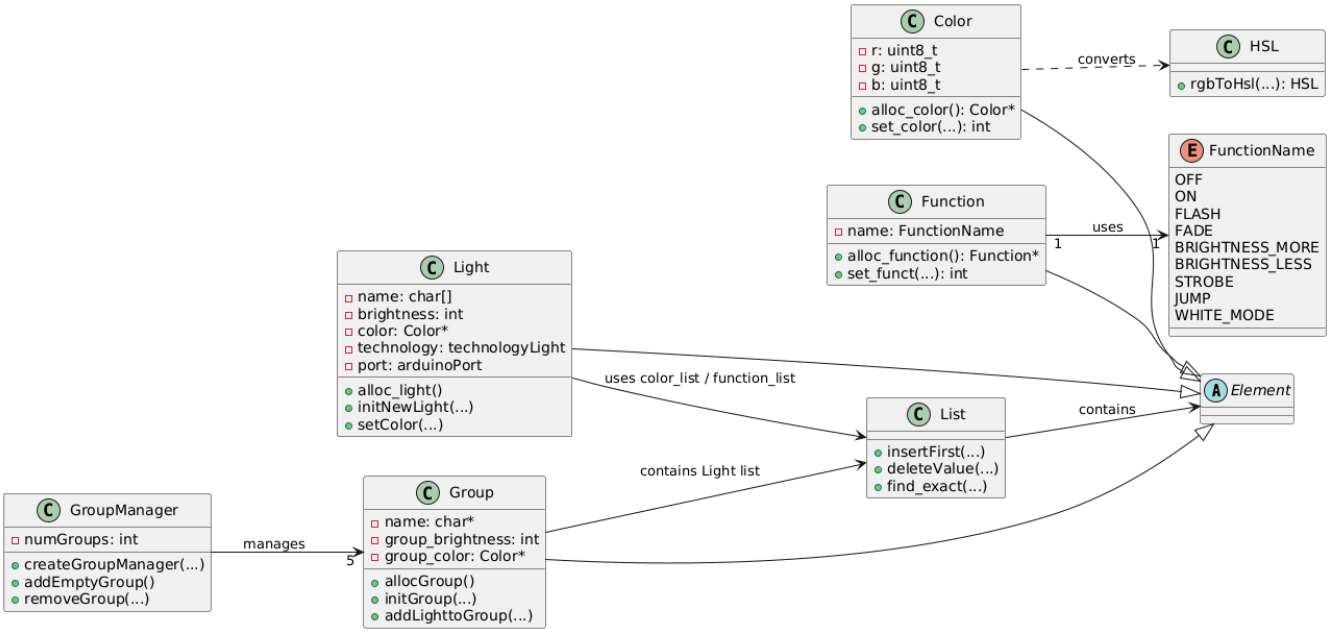


Figure 1 : Diagramme UML de nos structures de données

Nous avons utilisé différentes librairies pour ce projet.

Tout d'abord, pour la partie interface graphique, nous avons utilisé GTK 3. Nous nous sommes aidé de l'outil Glade pour créer un fichier interface.ui, simplifiant la gestion en "bloc" des différents éléments à l'écran. Cependant chaque élément a sa partie de code C, avec les fonctions de la librairie GTK, qui permet de rendre chaque élément utilisable.

Pour la partie Arduino, la librairie IR V2.6.0 Remote a été utilisée, pour la réception et l'émission des codes IR. Le fichier GRANDEII_ARDUINO.ino permet de gérer la communication du PC vers la carte Arduino.

Pour la partie modèle, nous avons développé un système de gestion des données. Celui-ci permet d'enregistrer tous les codes infrarouges saisis, ainsi que toutes les lumières associées à chaque groupe. (voir ci-dessus : Diagramme UML)

Chaque lumière est représentée par une structure de données contenant ses propriétés (nom, type, emplacement, ...) ainsi que ses fonctions spécifiques (allumage, extinction, changement de couleur, ...). L'ensemble des fonctions et des couleurs associées à une lumière est géré à l'aide d'une liste doublement chaînée, permettant une insertion et une suppression efficaces dans les deux sens.

Les lumières sont organisées en groupes, chacun pouvant contenir plusieurs lumières. Pour simplifier la gestion et le renommage rémunéré automatique, nous avons conçu une structure centrale appelée GroupManager. Cette dernière utilise un tableau (array) de groupes, afin de garantir un accès rapide et un renommage aisé des groupes.

Afin de rendre notre système le plus universel possible, nous avons également implémenté des fonctions permettant de trouver les couleurs les plus proches de celles souhaitées. Pour ce faire, les couleurs sont converties du format RGB vers le format HSL (Teinte, Saturation, Luminosité), ce qui simplifie les comparaisons entre couleurs, en particulier la recherche de teintes similaires.

Des tests unitaires couvrent les principales fonctions du modèle, ce qui garantit la stabilité du système lors de ses évolutions. Sur cette partie du code, nous avons atteint un taux de couverture de 90 %, ce qui reflète un niveau de test satisfaisant. Les tests de gestion mémoire (malloc) sont supposés réussir, et aucune fuite mémoire n'a été détectée lors des tests avec Valgrind.

Nous avons implémenté un système permettant d'enregistrer tous les codes infrarouges saisis, et toutes les lumières affiliées à chaque groupe. Pour ce faire, un bouton ouvrant une popup sur l'IHM permet d'enregistrer ces informations dans un fichiers en .json. Concernant la manipulation des fichiers JSON, nous avons utilisé la librairie cJSON, qui offre une API simple et efficace pour le parsing et l'écriture de données au format JSON en C.

Ces fichiers peuvent aussi être lus, pour transférer ces informations d'un ordinateur à un autre.

Pour faciliter la compréhension du code et l'intégration de nouveaux membres, nous avons mis en place une documentation complète générée avec Doxygen.

# RÉSULTATS

Initialement, le projet était prévu sous différentes versions:

V0: création des structures, création du menu à plusieurs fenêtres pour l'IHM, reconnaissance et duplication de codes IR par l'Arduino

V1: lien entre le PC et l'Arduino via le Serial port, création des onglets et widget pour l'IHM, développement des fonctions basiques (changements de couleurs, d'intensité, ..)

V2: création d'un système "drag & drop" pour assigner des lumières à un groupe, fiabiliser au maximum la liaison Arduino - PC

V3: permettre la modification des groupes et des lumières, création d'un système de sauvegardes pour enregistrer son "show" (avec tous les codes enregistrés)

Au vu de l'ampleur du projet, nous avons dû faire des choix dans l'implémentation des différentes features. La v0 et la v1 ont été réalisées, la liaison Arduino-PC a été fiabilisée et le système de sauvegarde a été implémenté.

Ce projet a un potentiel d'amélioration, en ajoutant une option pour pouvoir changer de technologie de contrôle: passer de l'infrarouge, au Wifi, Bluetooth ou bien DMX. Nous pouvons aussi imaginer l'implémentation d'un microphone pour se caler sur une musique, ou même un contrôleur de type MIDI pour faire varier les couleurs, intensités, effets via des boutons et des potentiomètres linéaires.

# ANNEXES

## Annexe 1 : Diagrammes de Gantt prévisionnel et réel


Diagramme de Gantt prévisionnel

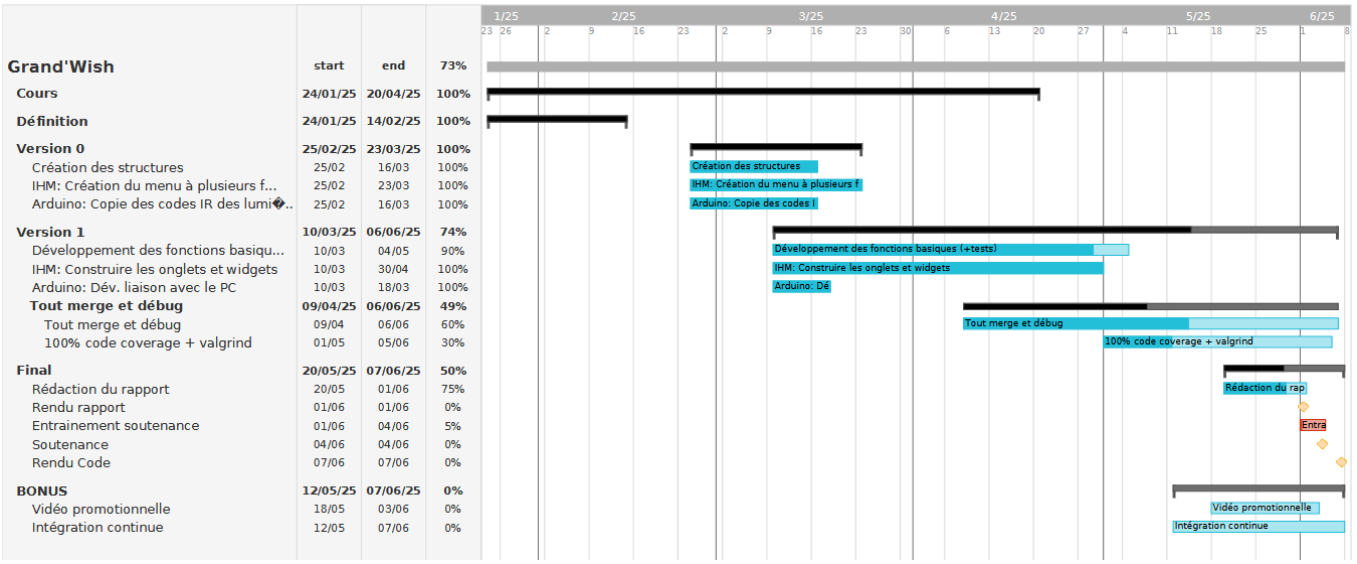


Diagramme de Gantt réel

**INSA Rennes**
20 avenue des Buttes de Coësmes
CS 70839
35708 Rennes cedex 7
Tél : + 33 (0)2 23 23 82 00
**www.insa-rennes.fr**

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE
Liberté
Égalité
Fraternité