INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

RAPPORT BE C++

Questions pour un champion

Binôme 2

MONTAIGU Emilie

OUVRARD Marine

18 janvier 2026

Introduction

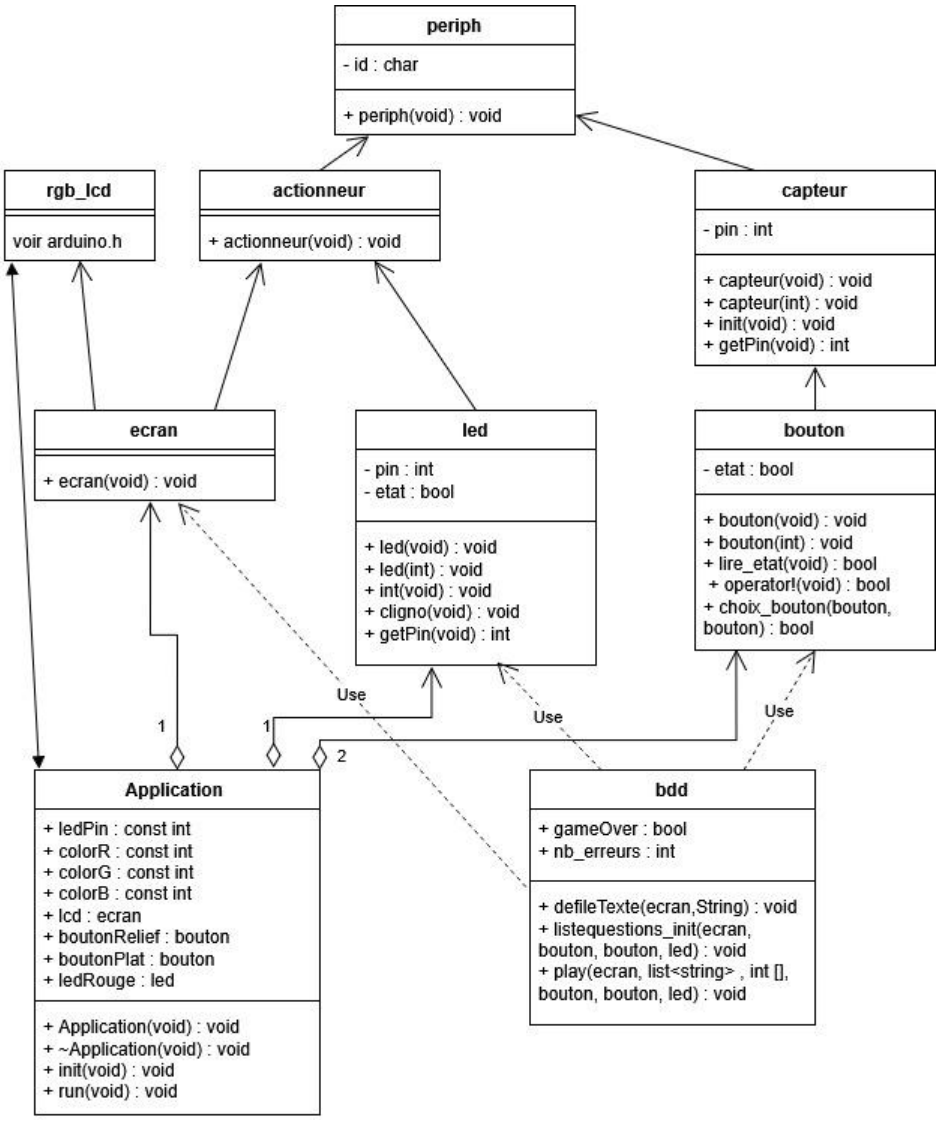
Dans le cadre du bureau d'étude sur le langage C++, nous avons réalisé un quiz nommé « Questions pour un champion ». Le but était de concevoir une bibliothèque modulaire qui permet d'interfacer des périphériques et de créer une application utilisant cette bibliothèque.

Nous voulions créer un projet éducatif et amusant c'est pour cela que nous nous sommes tournées vers l'idée d'un quiz. L'application propose cinq thématiques différentes de questions avec 10 questions par thème. Nous avons eu besoin, en plus de la carte ESP8266 NodeMCU, d'un écran LCD RGB (protocole I2C) pour afficher ce qu'on voulait, de deux boutons (un en relief et un plat), d'une LED rouge pour signaler les erreurs, de câbles pour les connexions et d'un câble USB pour relier la carte à l'ordinateur.

L'environnement de développement requis comprend l'Arduino IDE, le support ESP8266 pour Arduino et la bibliothèque rgb_lcd nécessaire pour piloter l'écran LCD.

Architecture logicielle

Diagramme de classe



Justification des choix de classe

Classe periph : Classe de base représentant tout périphérique connecté. Elle possède l'attribut id commun à tous les composants.

Classe capteur : Hérite de periph. Représente les périphériques d'entrée (lecture de données).

Classe actionneur : Hérite de periph. Représente les périphériques de sortie (affichage, lumière).

Classe bouton : Hérite de capteur. Capteur avec état booléen, méthode lire_etat() et surcharge de l'opérateur !.

Classe led : Hérite de actionneur. Actionneur avec pin et etat, méthodes init() et cligno().

Classe ecran : Héritage multiple de actionneur et rgb_lcd pour combiner fonctionnalités d'affichage et contrôle de la couleur de l'écran.

Éléments C++ utilisés

Héritage : périphériques → capteurs/actionneurs → composants

Héritage multiple : ecran hérite de actionneur et rgb_lcd

Surcharge d'opérateur : L'opérateur operator!() est redéfini dans la classe bouton pour faciliter la lecture de l'état du bouton.

STL : utilisation de la Standard Template Library avec les conteneurs list et string (via #include <list> et #include <string>) pour stocker les questions de manière dynamique.

Exceptions : principe de gameOver avec un booléen. Si l'utilisateur fait plus de 3 fautes, sa partie s'arrête.

Architecture matérielle

Schéma de connexion

ESP8266 NodeMCU

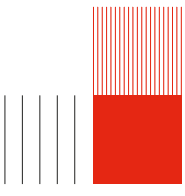
- D8 → Bouton Relief (capteur)
- D6 → Bouton Plat (capteur)
- D7 → LED Rouge (actionneur)
- I2C → Écran LCD RGB (actionneur)

Schéma de fonctionnement

Initialisation : Configuration des pins, initialisation de l'écran LCD

Sélection du thème : Affichage des thèmes disponibles, sélection par boutons

Déroulement du quiz : Affichage de la question avec défilement du texte à l'écran, lecture des boutons pour la réponse (relief = vrai, plat = faux), validation et feedback optionnel (si réponse incorrecte, affichage du nombre d'erreurs et LED rouge) puis passage à la question suivante.



Fin de partie : Si pas de game over (moins de 3 fautes), affichage du score et message de réussite. Sinon, affichage de « GAME OVER ».

Relance d'une nouvelle partie.

Conclusion

But atteint

Le projet a permis de créer une bibliothèque orientée objet modulaire et extensible qui respecte les principes de la conception orientée objet et de la programmation orientée objet. L'application est fonctionnelle avec une interface utilisateur intuitive construite à partir des éléments Grove.

Problèmes rencontrés

La gestion du défilement de texte sur l'écran LCD 16x2 a représenté problème. L'affichage étant limité à deux lignes de 16 caractères, il a fallu implémenter une fonction de défilement progressif pour permettre la lecture complète des questions longues.

La synchronisation des boutons a également posé problème en raison du rebond. Lors d'un appui sur un bouton, plusieurs impulsions parasites peuvent être détectées nécessitant la mise en place d'un mécanisme de filtrage temporel pour éviter les lectures multiples non désirées et l'utilisation de délais pour laisser le temps au système.

Perspectives d'évolution

Le projet offre de nombreuses possibilités d'amélioration et d'extension. L'ajout d'une connexion WiFi permettrait de télécharger des quiz depuis internet ce qui permettrait d'avoir des nouveaux contenus régulièrement mis à jour sans nécessiter de reprogrammer la carte à chaque fois. L'ajout d'un buzzer pourrait fournir un feedback sonore pour améliorer l'expérience utilisateur lors des bonnes ou mauvaises réponses.