La maison connectée(Projet C++)

Compte rendu

Objectif du Projet

L’objectif principal de notre projet est de transformer une maison traditionnelle en une maison intelligente et connectée. Grâce à notre solution, l’utilisateur pourra contrôler facilement et efficacement les différentes fonctionnalités de son domicile directement depuis une application mobile simple et intuitive. Cela inclut la gestion des lumières, des fenêtres, des dispositifs de sécurité, et bien plus encore, le tout en quelques clics. Notre ambition est d’améliorer le confort et la praticité de la vie quotidienne en mettant la technologie au service de l’utilisateur.

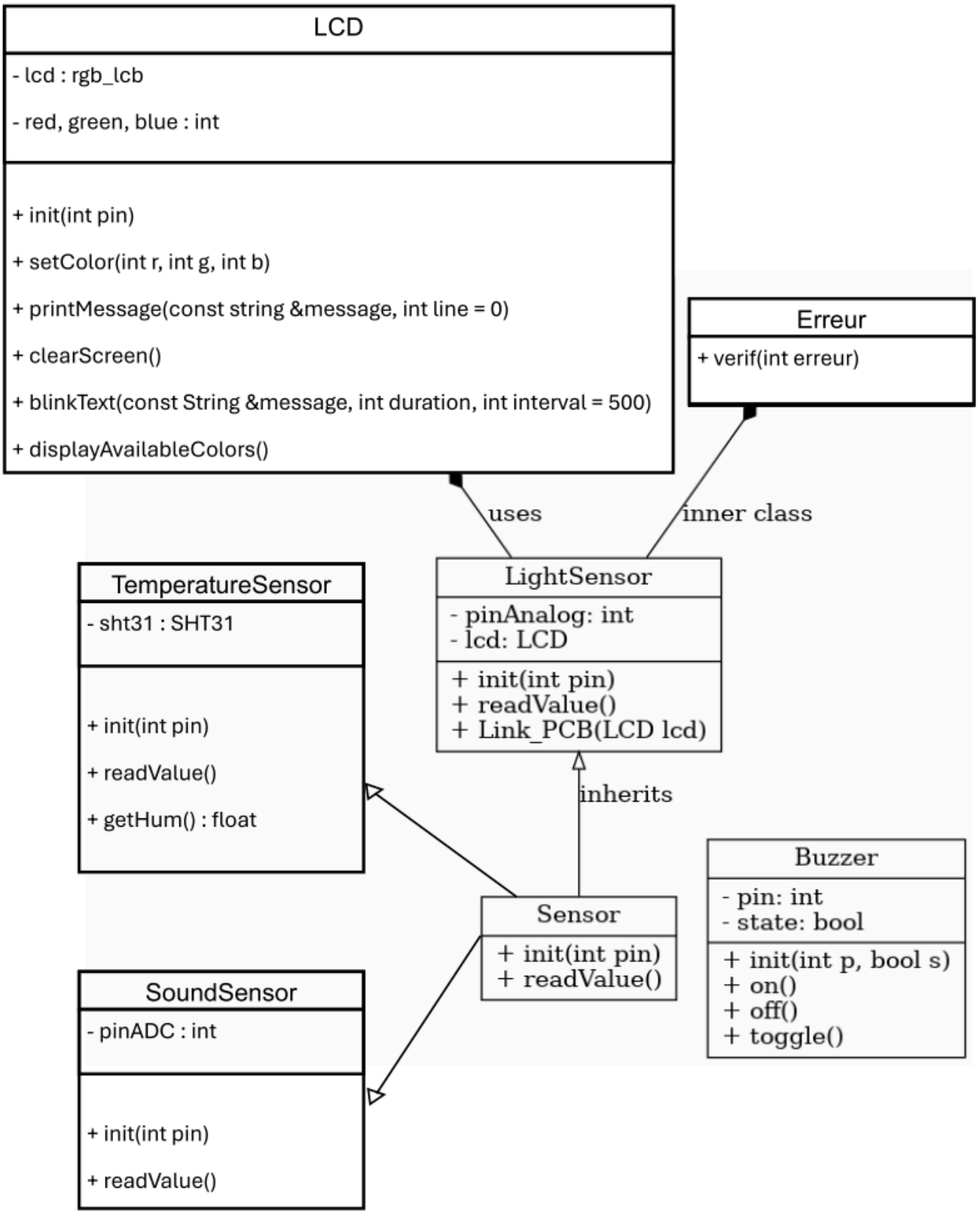
Hamrouni iyed
Ishq yonan Marvin
Ducasse Thomas

Démarche de Réalisation

Pour concrétiser cet objectif, nous avons suivi une approche méthodique en plusieurs étapes, tout en surmontant les défis rencontrés :

1. **Mise en Fonctionnement des Actionneurs, des Capteurs et du Module Wi-Fi**
    a. Initialement, nous avons rencontré des problèmes de connexion, même en utilisant les exemples fournis avec les modules.
    b. Nous avons constaté que lorsque plusieurs composants, notamment les actionneurs et les capteurs (excepté l'écran LCD), étaient connectés simultanément, des dysfonctionnements surviennent.
    c. Le contrôle du moteur s'est révélé être le plus complexe, entraînant des comportements inattendus tels que des déconnexions aléatoires et des vibrations inhabituelles.
2. **Création de Classes pour les Composants**
    a. Chaque capteur et actionneur a fait l'objet d'une encapsulation dans une classe dédiée afin de structurer le code et de faciliter la maintenance.
    b. Ces classes ont été testées individuellement pour garantir leur bon fonctionnement avant intégration.
3. **Implémentation et Tests Progressifs**
    a. Nous avons implémenté les fonctionnalités une à une, en nous assurant que chaque composant fonctionnait de manière isolée.
    b. Cela nous a permis d'identifier et de résoudre les problèmes spécifiques à chaque composant avant de les intégrer dans le système global.
4. **Intégration du Système et Tests Finaux**
    a. Une fois les fonctionnalités individuelles validées, nous avons intégré les différentes parties du projet pour former un système cohérent.
    b. Des tests finaux ont été réalisés, révélant quelques problèmes mineurs, tels que des interférences entre certains composants, que nous avons corrigés.

Diagramme de classes :



Scénario d’utilisation

Fonctionnalités principales

1. **Contrôle via serveur Web :**
    - Le programme héberge une interface web sur le microcontrôleur ESP8266. Les utilisateurs peuvent accéder à cette interface pour interagir avec les composants connectés.
    - Contrôle du moteur Servo : Une commande permet de régler l'angle du moteur via un curseur.
    - Contrôle de la LED : Les utilisateurs peuvent allumer ou éteindre une LED.
    - Contrôle du Buzzer : Les utilisateurs peuvent activer ou désactiver un buzzer.
    - Retour d'état : L'état actuel des composants (LED et Buzzer) est visible en temps réel.
2. **Capteurs surveillés :**
    - **Capteur de température et d'humidité :**
        - Mesure la température et l'humidité ambiantes.
        - Déclenche des alertes si les seuils définis (28°C pour la température et 70% pour l'humidité) sont dépassés.
    - **Capteur de lumière :**
        - Mesure la luminosité ambiante en utilisant un capteur analogique.
        - Allume une LED lorsque la lumière est insuffisante, et l'éteint lorsque la luminosité est suffisante.
    - **Capteur de son (non utilisé) :** Bien que le capteur soit initialisé, il n'est pas exploité dans ce programme.
3. **Actionneurs connectés :**
    - **Buzzer :**
        - Émet un signal sonore pour avertir en cas de température ou d'humidité élevée.
    - **Moteur Servo :**
        - Utilisé pour simuler une action d'ouverture/fermeture (par exemple, rideaux), en fonction de certaines conditions.
    - **LED :**
        - Allume ou éteint une lumière selon la luminosité mesurée.
4. **Interface utilisateur locale :**
    - Affichage LCD :
        - Utilise un écran RGB LCD pour afficher des messages informatifs sur les conditions ambiantes.
        - Change de couleur selon l'état (rouge pour des conditions critiques, jaune pour des alertes mineures, blanc pour des conditions optimales).

Détails techniques des fonctionnalités

1. Interface Web :

- **Les routes web sont configurées pour :**
    - "/" : Charger la page HTML de l'interface utilisateur.
    - "/toggleLED" : Basculer l'état de la LED.
    - "/toggleBuzzer" : Basculer l'état du Buzzer.
    - "/getState" : Retourner un JSON contenant les états actuels des composants.
- **Connexion WiFi :**
    - Les identifiants SSID et mot de passe permettent la connexion au réseau local.
    - Le serveur est accessible via l'adresse IP affichée dans le moniteur série.

2. Capteurs :

- **Les seuils définis dans le code sont :**
    - Température : 28°C.
    - Humidité : 70%.
    - Luminosité : Calculée via une résistance Rsensor (>100 pour déclencher l'action).

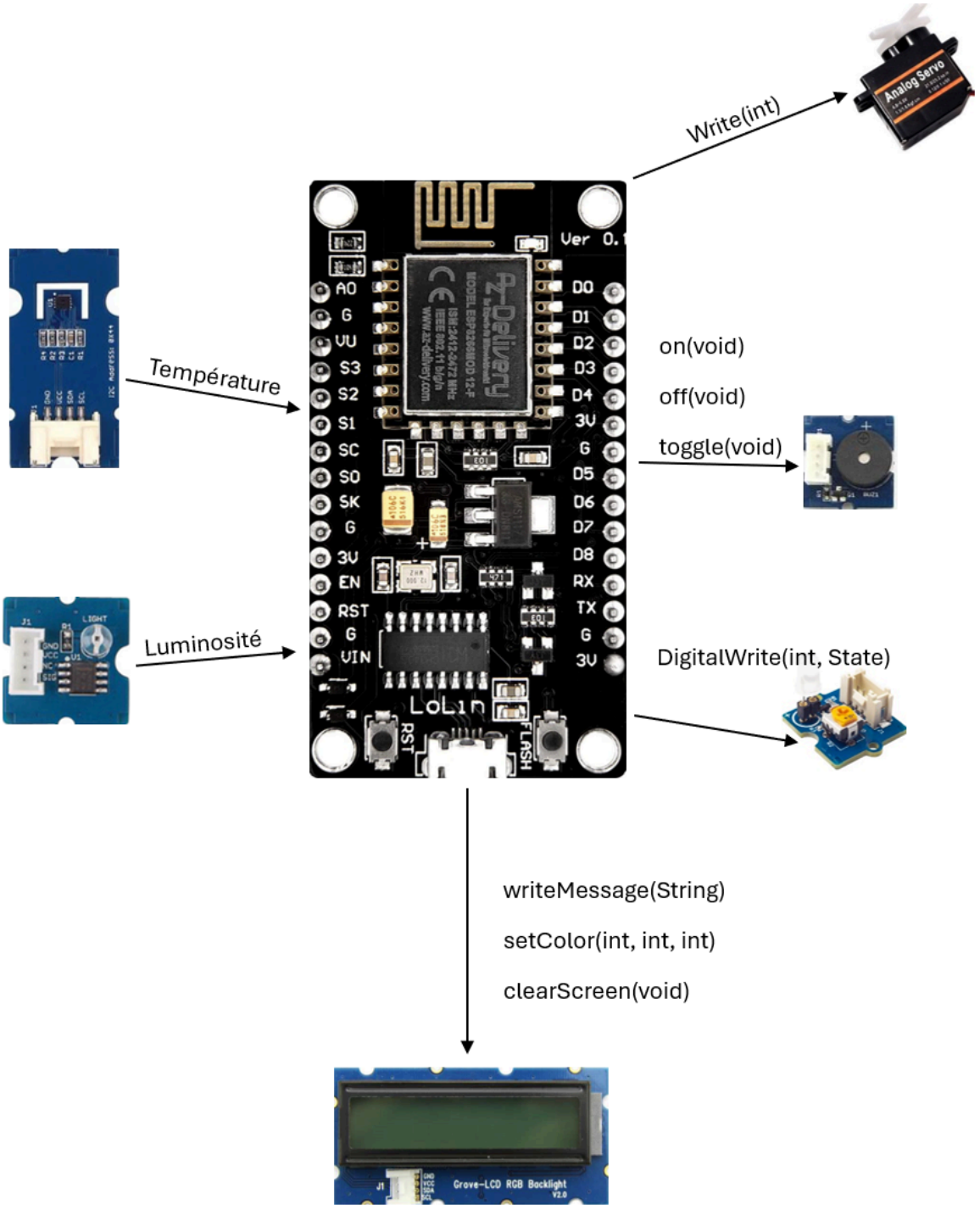
3. Actions automatiques :

- **En cas de température ou d'humidité élevée :**
    - Le buzzer s'active.
    - Le moteur servo s'ouvre (position 180°).
    - Un message d'avertissement est affiché sur l'écran LCD.
- **Si la luminosité est faible :**
    - La LED s'allume, et un message est affiché.
- **Si les conditions redeviennent normales :**
    - Le buzzer s'éteint.
    - Le moteur servo revient à 0°.
    - L'écran LCD affiche un message de conditions optimales.

Matériel utilisé :

- microcontrôleur : ESP8266
- capteurs : luminosité (Grove Light Sensor) , température et humidité (SHT31),
- actionneurs : servomoteur (Grove - Servo), LED (Grove - LED), buzzer (Grove - Buzzer)
- écran : écran LCD (Grove - LCD RGB Backlight)

Diagramme de système :



Tous les capteurs/actionneurs possèdent chacun une fonction init() (ou équivalent par exemple pinADC(int), attach(int),...) pour initialiser chaque composant sur la broche où on l'a attaché (GPIO, I2C, ADC).