Julien DUMAS et Enzo DURAND - 4A AE-SE TP4

Compte rendu du projet de programmation orientée objet : Écran connecté et ++

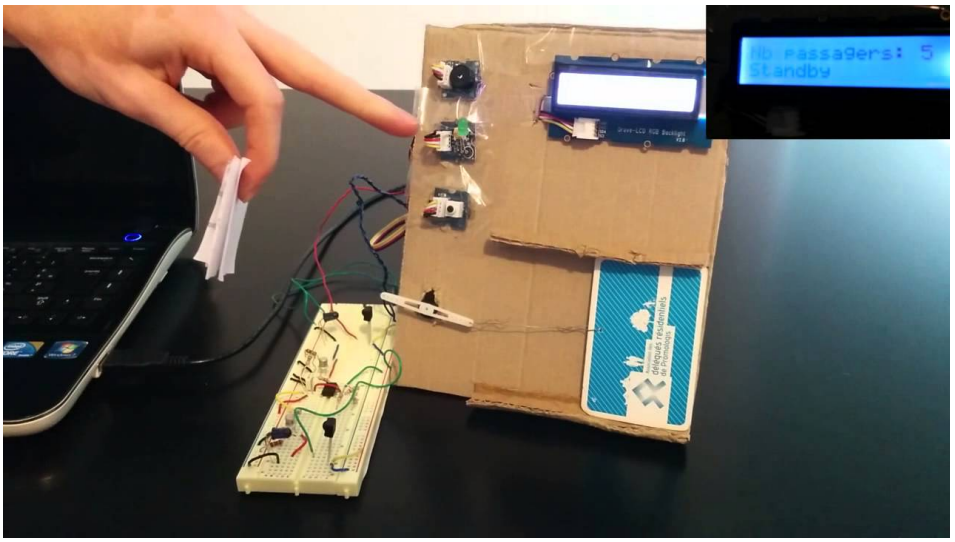


photo non contractuelle

Introduction :

Suite à nos enseignements théoriques sur la programmation orientée objet, nous sommes passés à la pratique avec un projet liant langage C++ et maquette physique.

Le but initial était de concevoir une matrice led connectée mais, n'ayant pas reçu les composants, nous nous sommes rabattus sur un écran lcd connecté. Grâce à notre avancement en dehors des séances, nous avons été en mesure d'y ajouter des fonctionnalités qui seront décrites ci après.

I.Description du projet

Dans ce projet, notre objectif était d'exploiter le langage C++ dans le domaine de l'IoT automobile, en mettant en valeur le plein potentiel de la carte ESP8266 dont nous disposions. Cette carte présente des capacités telles que la communication en Wifi, la possibilité de piloter de nombreux actionneurs grâce à ses 16 ports numériques, et la capacité de recevoir une entrée analogique.

Initialement, nous avions envisagé d'implémenter des fonctionnalités innovantes visant à améliorer l'expérience du conducteur. Cela incluait la conception d'une matrice de LEDs rabattable à l'aide d'un servomoteur, permettant d'afficher du texte ou des images à l'attention du conducteur suivant, ainsi qu'un capteur de pollution de l'air pour informer le conducteur sur la qualité de l'air. Cependant, en raison de contraintes de temps et du non-acheminement de certains composants, nous avons dû nous rabattre sur des actionneurs de remplacement.

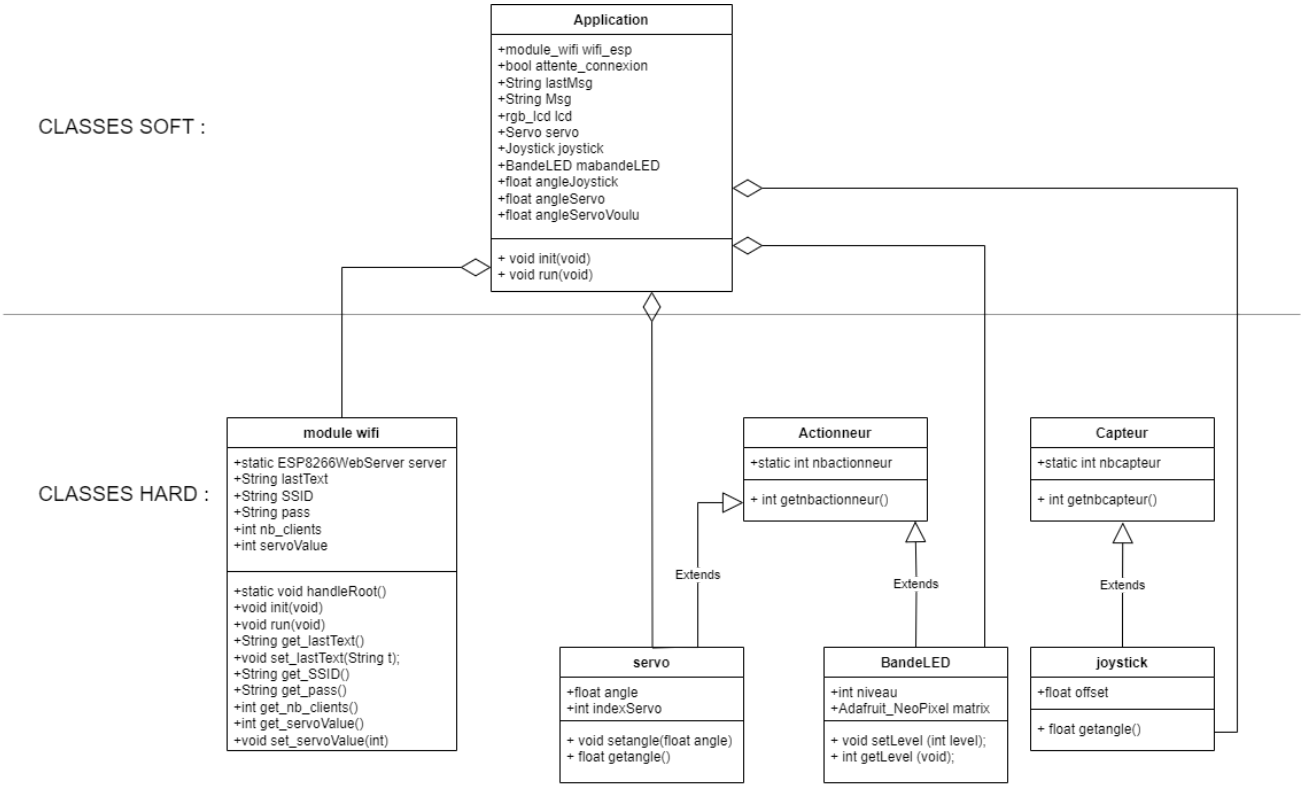
Finalement, notre système comprend un afficheur LCD, un joystick, un servomoteur, une barette de LEDs, et l'ESP8266. Bien que nous n'ayons pas pu intégrer le capteur de pollution de l'air comme initialement prévu, notre solution reste orientée vers une expérience utilisateur améliorée et des fonctionnalités pratiques pour le conducteur.

Une interface web en HTML permet d'interagir avec le projet. On peut l'utiliser pour afficher des messages sur l'écran, ou régler la position du servomoteur lorsque le joystick n'est pas actionné manuellement. (fonctions web dédiées aux passagers bien évidemment…).

II. Conception

Le projet ayant été décrit ci-dessus, nous allons ici nous intéresser à sa conception d'un point de vue classes et programmation. Nous avons décidé pour ce projet de regrouper les éléments externes à l'ESP8266 sous les classes "Actionneur" et "Capteur", qui permettent de compter le nombre de périphériques instanciés. La carte WiFi étant intégrée à la carte et ayant une gestion particulière, elle dispose de sa propre classe "module WiFi". La figure ci-dessous présente les aspects de notre conception.

Diagramme des classes du projet :



Librairies utilisées :

-Arduino.h
-FS.h
-Wire.h
-rgb_lcd.h
-ESP8266WiFi.h
-WiFiClient.h
-ESP8266WebServer.h
-Adafruit_NeoPixel.h

Enfin, nous avons pris les ports suivants :

- afficheur LCD I2C : SDA et SCL
- joystick : A0 (seule entrée analogique disponible)
- servomoteur : D5
- bande LED : D7

Les broches D5 et D7 sont choisies lors de l'appel des constructeurs dans la classe Application. A0 est définit dans la classe joystick.

III. Fonctionnement du système

Au démarrage, l'écran affiche les informations WiFi permettant de se connecter directement à l'ESP (solution retenue car il n'y a pas forcément de borne WiFi dans une voiture ordinaire).

Une fois connecté avec un smartphone par exemple, les informations WiFi s'effacent de l'écran, et la maquette devient opérationnelle d'un point de vue gestion sans fil.

Sur la page HTML, on retrouve un bouton pour basculer l'état de la led de l'ESP. Cela peut servir pour tester la bonne communication. Un champ de saisie est proposé pour envoyer un message à l'écran.

Le joystick à la priorité sur la page HTML. Tous deux permettent de fixer le servomoteur à un angle compris entre 0 et 180°. De plus, le joystick permet aussi une utilisation en tant que commodo, pour activer les clignotants (sur la bande led).

Le readme ci-joint explique comment utiliser l'application.

Nous avons cependant rencontré deux problèmes : le servomoteur appelle trop de courant, ce qui rend le système instable, et la lecture de la valeur du joystick n'est pas compatible avec le WiFi : le SSID n'est plus diffusé lorsque la conversion ADC du joystick est présente dans le code.

En revanche, lorsque les composants ne sont pas utilisés tous en même temps, ces soucis disparaissent et l'application fonctionne correctement.

IV. Analyse et Conclusion

Bien que les composants commandés n'ont pas pu nous être acheminés, nous avons su rebondir et produire une maquette fonctionnelle. Comme nous avons bien avancé pendant et entre les séances, plusieurs actionneurs ont pu être utilisés. Avec un peu plus de temps, nous aurions pu proposer un boîtier pour nos composants, afin de les fixer sur une grille de climatisation d'un véhicule par exemple, et ajouter un écran sur la lunette arrière.

Toutefois, ce projet intègre différentes notions du C++, comme l'instanciation des classes, la hiérarchisation, la redéfinition d'opérateur, les exceptions et une proposition d'utilisation de la STL (cf. code Application.cpp).

Ainsi, ce projet est à nos yeux une réussite au vu de l'apport pédagogique à travers la mise en pratique, de son fonctionnement plutôt correct, mais aussi au vu du temps imparti et de notre rebond avec les composants finalement obtenus.