# POA Projet C++ : Doc CPC

*Informatique de Confiance : Cybersécurité du logiciel*

## Introduction

Ce document présente et documente les classes et les fonctionnalités du projet CPC, CoffeePotController par Félix Bezançon et Matthieu Jan. Pour une information sur la compilation et l'usage, voir le README.md du projet.

## Documentation Client

### Librairies :

Le client utilise la librairie standard c++ (lstdc++) et la librairie boost (-lboost_xxx)

### ClientCtl :

La classe ClientCtl décrit l'objet utilisé pour instancier un client. Il prend en paramètre une vue et un controleur réseau dédié au client. L'execution des commandes. (start, stop, get) se fait en utilisant la fonction void run(string commande), qui va demander au controleur réseau de la traiter, et va l'afficher via la vue.

### ClientNetCtl :

La classe ClientNetCtl décrit le controleur de réseau abstrait dédidé au client, possédant une méthode run permettant d'executer des actions. Son constructeur prend en paramètre l'hôte et le port à contacter, qui seront fixés pour le reste de l'éxécution.

### BasicClientNetCtl :

La classe BasicClientNetCtl est une implémentation de ClientNetCtl. Celle ci génére des messages formatés pour pouvoir marcher avec l'implémentation BasicServerNetCtl, mais sans respecter les standards HTTP. La méthode emitCmd envoie une commande au serveur, et la méthode handleReply récupère la réponse.

### View et ConsoleView :

Ces deux classes étaient initialement prévues pour gérer une interface avec l'utilisateur. Le client loggant finalement ces messages dans la sortie standard elles n'ont pas été implémentées.

**Documentation Server**

## Support :

Le serveur est prévu pour fonctionner sur un RaspberryPi, et utilise les librairies et fonctions associés.

## Librairies :

Le serveur utilise la librairie standard c++ (lstdc++), la librairie boost (-lboost_xxx) et les librairies wiringPi (lwiringPi) et python2.7 (lpython2.7)

## ServerCtl :

La classe ServerCtl décrit l'objet utilisé pour instancier un serveur. Il prend en paramètre un driver et un controleur réseau dédié au server. La fonction run demande au serveur de se lancer.

## ServerNetCtl :

La classe ServerNet décrit le controleur de réseau abstrait dédidé au serveur, possédant une méthode listen permettant d'executer d'écouter les connexions entrantes.

## BasicServerNetCtl :

La classe BasicServerNetCtl est une implémentation simple de ServerNetCtl, dans le sens où il réagit comme une serveur web très simple. Bien que ses réponses soient conformes aux standards HTTP, une connexion entrante recevra, peu importe l'url, une réponse HTTP 200 OK avec une page web unique contenant des informations simples et deux boutons. C'est aussi lui qui appelle PotState pour changer l'état de la cafetière.

## PotState :

La classe PotState conserve l'état de la cafetière. Elle est modifié via la méthode setState(bool s) permettant de passer l'état à On ou OFF. Cette méthode va appeler le driver qui lui a été associé.

PotState est un singleton, afin de n'avoir qu'une seule instance de l'état et un seul Driver associé. Il n'est à priori pas thread safe, ce qui peut générer des erreurs. Son driver doit être set (via setDriver) avant l'utilisation, typiquement lors de la création du controleur de serveur.

## Driver, LedDriver et CoffeeDriver :

La classe Driver propose une interface simple, avec un void set (bool) et un bool get(). Un set(true) « allume » le périphérique associé, et un set(false) l'éteint.

LedDriver permet de controler de cette façon une Led connecté en entrée sur la pin 7 du RaspberryPi.

CoffeeDriver permet de controler de cette façon deux servo moteur connecté en entrée sur les pin 11 (pour allumer) et 7 (pour éteindre) du RaspberryPi. Le montage sur la machine a café devra se faire en fonction. Le calibrage des servomoteurs se fait « en dur » dans la classe CoffeeDriver.cpp.