Generic C++ Developer

Управление светодиодом

Тестовое задание

Необходимо реализовать систему управления светодиодом камеры. Светодиод камеры (LED) характеризуется следующим набором параметров.

1. Состояние (state): on/off.
2. Цвет (color): red, green, blue.
3. Частота мерцания (rate): 0..5 (в Герцах).

Архитектура

Система должна состоять из 2-х компонентов.

1. Приложение на C++ (сервер), имитирующее взаимодействие со светодиодом.
2. Приложение на C++ (клиент), которое обеспечивает интерфейс для управления светодиодом.

Сервер и клиент общаются между собой посредством TCP/IP с помощью простого текстового протокола запрос-ответ. Клиент устанавливает соединение с сервером и отправляет одну или несколько управляющих команд. Сервер обрабатывает команды и на каждую из них отправляет клиенту ответ.

Протокол

Каждый запрос и ответ представляют собой строку, заканчивающуюся символом '\n'.

Запрос: *COMMAND [ARGUMENT]\n*
Ответ: *STATUS [RESULT]\n*

Поле ответа *STATUS* может принимать следующие значения.

1. *OK* – команда выполнена успешно.
2. *FAILED* – в результате выполнения команды произошла ошибка.

Команды

Сервер должен поддерживать следующие команды.

Команда	Аргумент	Результат	Описание
set-led-state	on, off	OK, FAILED	включить/выключить LED
get-led-state		OK on|off, FAILED	запросить состояние LED
set-led-color	red, green, blue	OK, FAILED	изменить цвет LED
get-led-color		OK red|green|blue, FAILED	запросить цвет LED
set-led-rate	0..5	OK, FAILED	изменить частоту мерцания LED
get-led-rate		OK 0..5, FAILED	запросить частоту мерцания LED

Примеры

R – запрос, *A* – ответ.

R: "**get-led-state**\n"
A: "**OK on**\n"

R: "**set-led-color yellow**\n"
A: "**FAILED**\n"

Требование к серверу

1. Архитектура сервера должна позволять легко добавлять новые команды и изменять логику работы существующих.
2. Сервер должен уметь работать с несколькими клиентами, обслуживая запросы в реальном времени.
3. Сервер должен корректно обрабатывать любые нештатные ситуации и ошибки.
4. Визуализация работы сервера оставляется на усмотрение разработчика.

Требования к клиенту

1. Архитектура клиента должна позволять легко добавлять новые команды и изменять логику работы существующих.
2. Визуализация работы клиента оставляется на усмотрение разработчика.

Общие требования

1. Язык и библиотеки: ISO C++ (до 17-го включительно) и его стандартная библиотека, Boost и/или glibc (для Linux).

2. Система сборки: GNU Make или CMake.
3. Код должен быть рассчитан на сборку и выполнение на современном дистрибутиве ОС Linux.

Результат выполнения задания

Результат выполнения тестового задания должен быть доступен в виде репозитория Git (на Bitbucket, Github или Gitlab), содержащего все исходные коды программы, а также краткую инструкцию по сборке и использованию (если требуется).