Фонарь (тестовое задание)

Требуется написать управляемый по сети фонарь. Команды управления фонарь принимает от сервера фонаря. Предполагается, что реализация сервера уже существует (однако недоступен вам в процессе разработки клиента фонаря). Фонарь и сервер общаются по Протоколу Управления Фонарем, работающему поверх соединения TCP.

Протокол Управления Фонарем (ПУФ) устроен следующим образом. Для изменения состояния фонаря сервер передает ему команду управления. Все команды кодируются в виде TLV (http://en.wikipedia.org/wiki/Type-length-value), при этом поле **`type`** имеет размер 1 байт, поле **`length`** — 2 байта и поле **`value`** — length байт. Все данные передаются по сети в Big Endian.

ПУФ версии 1 описывает три команды:

- ON (включить фонарь): type = 0x12, length = 0
- OFF (выключить фонарь): type = 0x13, length = 0
- COLOR (сменить цвет): type = 0x20, length = 3, value интерпретируется как новый цвет фонаря в RGB.

Предполагается, что в будущих версиях ПУФ могут появляться новые команды или расширяться набор данных, передаваемых с существующими командами, однако структура TLV останется неизменной.

Реализация фонаря должна удовлетворять следующим **требованиям**:

1. При запуске фонарь должен запрашивать хост:порт (по умолчанию 127.0.0.1:9999), подсоединяться по TCP и после этого начать отрабатывать протокол управления.

2. При получении данных от сервера фонарь собирает целые команды (type + length + value) и, если type известен, обрабатывает команду, иначе молча ее игнорирует.

3. При получении команды ON фонарь включается (отрисовку фонаря оставляем на
ваше усмотрение).

4. При получении команды OFF фонарь выключается.

5. При получении команды COLOR фонарь меняет цвет.

6. При завершении работы фонарь корректно закрывает соединение с сервером.

7. Реализация фонаря позволяет легко добавлять любые новые команды.

8. При неожиданных разрывах связи фонарь должен пытаться восстановить соединение.

Технологические требования:

1. Задание принимается в виде готового к выполнению Python-пакета.
2. Версия Python — 2.7 или 3.6 (на выбор).
3. Для реализации сетевого протокола использовать фреймворк Asyncio или Tornado.
4. Репозиторий с исходниками должен быть доступен на GitHub или Bitbucket.