федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

Дисциплина:
«Технологии проектирование программного обеспечения»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Выполнил:
Магистрант гр. N42605
Смирнов Кирилл Дмитриевич

Проверил:
Гирик Алексей Валерьевич

Санкт-Петербург,
2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Умный вентилятор (далее - устройство) является системой, которая регулирует потоки воздуха.

Целью создания программной системы - обеспечение управления устройством по сети с помощью современного стека технологий и средств проектирования и разработки ПО.

Было получено задание: спроектировать и разработать две программы для для ОС Linux на языке программирования Python: сервера и клиента, где сервер - управляет состоянием устройства, клиент - отправляет запросы серверу на установку параметров устройства и получения информации.

АНАЛИЗ ТРЕБОВАНИЙ

1 АНАЛИЗ ОСУЩЕСТВИМОСТИ ТРЕБОВАНИЙ

В ходе анализа требований стало известно, что продукт должен состоять из двух подсистем:

- сервисная подсистема, отслеживающая изменения в состоянии устройства, получающий/передающий запросы и ответы по сетевому протоколу управления и создающую/записывающую/читающую базу данных.
- клиентская подсистема, программу для управления устройством через сервис: выставление значений скорости вращения, температуры, получения данных о состоянии умной лампы.

Следовательно к системе предъявляются следующие требования:

- система должна обладать клиент-серверной архитектурой;
- система должна создавать базу данных в формате .bin;
- система должна быть консольным приложением, настройка работы которого выполняется путем передачи необходимых данных в аргументах командной строки и/или переменных окружения;
- система должна работать под ОС Linux;
- система должна по полученной от пользователя команды выставлять настройки скорости и температуры;
- система должна вывести пользователю информацию о состоянии параметров устройства;
- система должна отправлять оповещение пользователю об изменении параметров;
- система должна обрабатывать ошибки в процессе работы и при необходимости корректно завершаться.

При использовании возможностей языка Python 3.7 осуществимо выполнение требований, возможные средства реализации требований представлены в Таблице 1.

Требование	Средство реализации
Система должна работать под ОС Linux	Python
Система должна обладать клиент-серверной архитектурой	Библиотека socket

Система должна создавать базу данных в формате JSON	Библиотека json Библиотека os
Система должна быть консольным приложением, настройка работы которого выполняется путем передачи необходимых данных в аргументах командной строки и/или переменных окружения	Библиотека sys
Система должна по полученной от пользователя команды выставлять настройки параметров	Библиотека socket Библиотека sys
Система должна вывести пользователю информацию о состоянии устройства	Встроенные возможности языка Python Библиотека socket
Система должна отправлять оповещение пользователю о смене скорости вращения и температуре	Библиотека socket
Система должна обрабатывать ошибки в процессе работы и при необходимости корректно завершаться.	Встроенные возможности языка Python

Из анализа Таблицы 1 можно сделать вывод, что выставленные к системе требования осуществимости.

2 ОПРЕДЕЛЕНИЕ ПРИОРИТЕТНОСТИ ТРЕБОВАНИЙ

Исходя из назначения продукта можно расписать требования в следующем порядке по убыванию:

1. система должна по полученной от пользователя команды выставлять настройки скорости вращения и температуры;
2. система должна вывести пользователю информацию о состоянии параметров скорости вращения и температуры;

3. система должна отправлять оповещение пользователю о смене параметров скорости вращения и температуры.

4. система должна работать под ОС Linux;

5. система должна обладать клиент-серверной архитектурой;

6. система должна быть консольным приложением, настройка работы которого выполняется путем передачи необходимых данных в аргументах командной строки и/или переменных окружения;

7. система должна обрабатывать ошибки в процессе работы и при необходимости корректно завершаться;

8. система должна создавать базу данных в формате .bin.

3 ВЫВОД

Исходя из анализа требований можно установить, что предъявляемые системе требования являются:

- независимыми - требования не пересекаются между собой;
- однозначными - требования трактуются в единственном значении;
- полными - требования полностью определяются в одном месте и все необходимая информация описана;
- атомарны - требование нельзя разделить на более мелкие;
- достижимыми - требования достижимы в рамках проекта;
- проверяемыми - реализация требований может быть проверена в ходе тестирования программного обеспечения;
- отслеживаемыми - требования четко задокументированы в задании к Лабораторной работе №1;
- необходимыми - требования определены и имеют заинтересованное лицо.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1 ПРОГРАММНАЯ АРХИТЕКТУРА

Программная архитектура продукта состоит из сервиса и клиентской программы.

1.1 Сервисная программа

1.1.1 Сервис выполнен в виде консольного приложения, которое можно запустить либо интерактивно в терминале, либо с помощью подсистемы инициализации и управления службами systemd.

1.1.2 Функциональность сервиса делиться на две части: имитация взаимодействия (опроса или получения уведомлений) с устройством и взаимодействие с клиентами по сети.

1.1.3 Сервис обнаруживает внешние изменения в состоянии устройства и отправлять уведомления о таких изменениях заинтересованным клиентам.

1.2 Представление устройства

1.2.1 Устройство, управление которым обеспечивает сервис, представляется в виде файла с заданным форматом.

1.2.2 Файл содержит данные о состоянии устройства.

1.2.3 Внешнее изменение файла соответствует внешним изменениям, которые могут произойти с устройством.

1.2.4 Изменение состояния устройства выполняется с помощью клиентской программы через сеть.

1.3 Клиентская часть системы

1.3.1 Клиентская программа может выполнена в виде консольного приложения, которое можно запустить интерактивно в терминал.

1.3.2 Любое количество клиентов может подключиться к серверной части (сервису) и согласованно управлять «устройством».

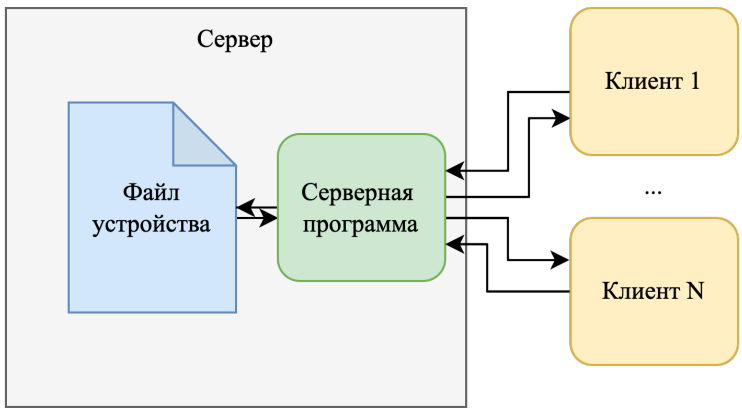

Рисунок 1 - Схема взаимодействия компонентов продукта.

2 ТЕХНИЧЕСКИЕ РЕШЕНИЯ

2.1 Реализация серверной части

2.1.1 Серверная программа (СП) реализована на языке программирования Python 3.7.

2.1.2 Выполняемые функции:

- установка скорости вращения
- установка температуры;
- отправка клиентской части значения параметров скорости и температуры;
- обработка ошибок.

2.1.3 Форматом передачи данных выбран JavaScript Object Notation (JCON) - текстовый формат обмена данными, основанный на JavaScript.

2.1.4 Форматом хранения данных выбран формат Binary - это формат, при котором информация записана при помощи последовательности байт.

2.1.5 Протоколом взаимодействия с клиентской частью выбран протокол UDP.

2.1.6 Для функционирования СП необходимы следующие программные пакеты:

- socket - интерфейс Python представляет собой прямую транслитерацию системного вызова Unix и интерфейса библиотеки для сокетов в объектно-ориентированный стиль Python: функция socket() возвращает объект сокета, методы которого реализуют различные системные вызовы сокетов;
- sys - модуль обеспечивает доступ к некоторым переменным, используемым или поддерживаемым интерпретатором, а также к функциям, тесно взаимодействующим с интерпретатором;
- os - модуль предоставляет портативный способ использования функций, зависящих от операционной системы;
- json - модуль позволяет кодировать и декодировать данные в формате JSON;
- datetime - модуль предоставляет классы для управления датами и временем.

2.2 Реализация клиентской части

2.2.1 Клиентская программа (КП) реализована на языке программирования Python 3.7.

2.2.2 Выполняемые функции:

- отправка сообщений на установку параметров устройства;

- получение сообщений от серверной части, а именно значений параметров устройства
- получение сообщений от серверной части об изменении параметров устройства;
- обработка ошибок;

2.1.3 Форматом передачи данных выбран формат JSON.

2.1.4 Протоколом взаимодействия с клиентской частью выбран протокол UDP, т.е. для передачи данных ему не обязательно устанавливать соединение между отправителем и получателем. Информация передается без предварительной проверки готовности принимающей стороны.

2.1.5 Для функционирования КП необходимы следующие программные пакеты:

- socket - интерфейс Python представляет собой прямую транслитерацию системного вызова Unix и интерфейса библиотеки для сокетов в объектно-ориентированный стиль Python: функция socket() возвращает объект сокета, методы которого реализуют различные системные вызовы сокетов;
- sys - модуль обеспечивает доступ к некоторым переменным, используемым или поддерживаемым интерпретатором, а также к функциям, тесно взаимодействующим с интерпретатором.

2.3 Установка и запуск серверной и клиентской программы

2.3.1 Для установки системы программ требуется клонировать git “https://github.com/ruskape/tppo_3232.git ” или скачать ZIP-файл.

2.3.2. Для запуска приложения требуется

2.3.2.1 Сделать файлы программ исполняемыми:

```
chmod +x tppo_client_3232.py
chmod +x tppo_server_3232.py
```

2.3.2.2 Запустить СП:

```
./tppo_server_3232.py
```

2.3.2.3 Запусть КП с одним из следующих ключей:

2.3.2.3.1 `-info` ключ для получения информации о состоянии устройства;

2.3.2.3.2 `-edit, -c` для настройки параметров устройства.

2.3.2.3.3 Пример:

```
./tppo_client_3232.py -edit 10 15
./tppo_client_3232.py -info
```

2.3.2 Требования к окружению

2.3.2.1 Целефой платформой является ОС Linux, также программа работает на Mac OS.

2.3.2.2. Архитектура системы должна соответствовать x86-64.

РАБОЧИЙ ПРОЕКТ

1 КОМПОНЕНТНЫЙ СОСТАВ

Предлагаемое приложение состоит из двух модулей - серверного модуля (tppo_server_3232.py) и клиентского (tppo_client_3232.py).

1.1 tppo_server_3232.py отвечате за:

- изменение параметров устройства;
- отправку клиентской части значения параметров устройства;
- отправку уведомления клиентской части об изменении параметров устройства;
- обработку ошибок.

Модуль tppo_server_3232.py состоит отвечает за:

- подключение по протоколу UDP;
- чтенияе сообщений от клиентской части;
- отправки сообщений клиентской части.

1.2 tppo_client_3232.py отвечате за:

- отправку сообщений на установку интенсивности свечения устройства;
- отправку сообщений на установку значений параметров;
- получение сообщений от серверной части значений параметров;
- получение сообщений от серверной части об изменении значений параметров;
- отправку серверной части запроса на получение актуальной информации об значениях параметров.

2 СКРИНШОТЫ РАБОТЫ ПРОГРАММ

Ниже на Рисунке 2 представлен пример работы серверной части:

1. запуск СП;
2. получение актуальной информации о параметрах устройства из файла формата .bin;
3. отправка информации о параметрах устройства;
4. изменения состояния устройства;
5. сохранение данных в файл .bin;

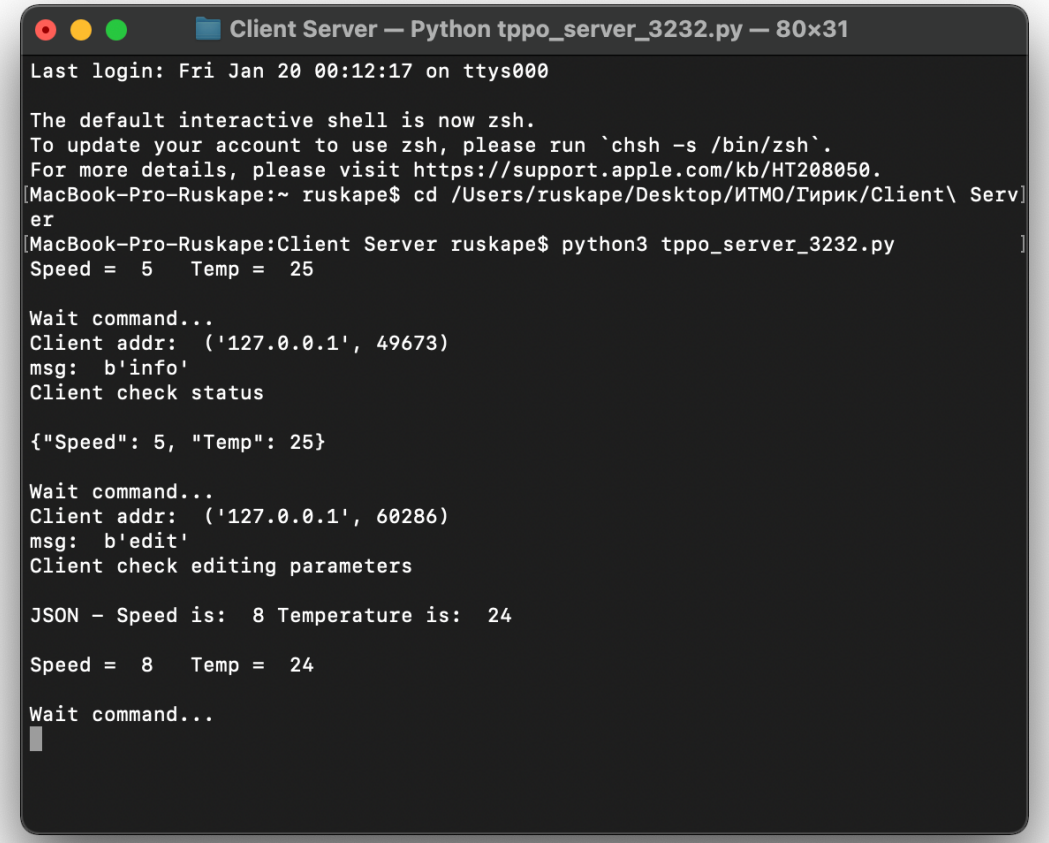

Рисунок 2 - Пример работы серверной части.

Ниже на Рисунке 3 представлен пример работы клиентской части:

1. получение актуальной информации о параметрах устройства;
2. запрос на смену параметров;

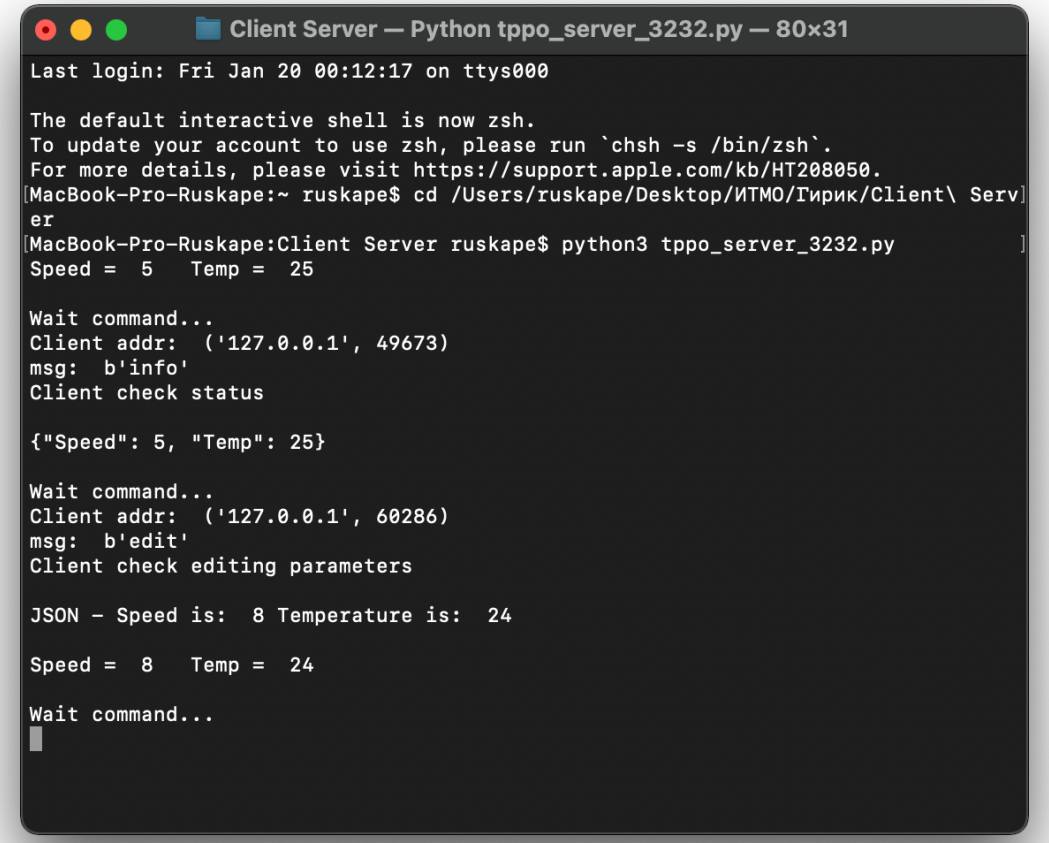

Рисунок 3 - Пример работы клиентской части.