МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра РЭС

ОТЧЕТ

по учебной практике

Тема: Рефакторинг кода. Проектирование встроенных приложений умного дома

Студент гр. 1181 Яковлев В.А.

Руководитель Проценко И.М.

Санкт-Петербург

2023

ЗАДАНИЕ

НА УЧЕБНУЮ ПРАКТИКУ

Студент Яковлев В.А.
Группа 1181
Тема практики: Рефакторинг кода. Проектирование встроенных приложений умного дома

Задание на практику:

Произвести рефакторинг (упрощение, расширение и понимание без изменения поведения) кода, а именно:

а)Комментирование и тестирование программы

б)Построение функциональной схемы программы в составе лабораторного комплекса

в)Приведение кода к стандарту PEP8

г)Описание Qt Designer среды разработки применительно к техническим требованиям разрабатываемой программы

Сроки прохождения практики: 01.02.2023 – 04.06.2023

Дата сдачи отчета: 08.06.2023

Дата защиты отчета: 09.06.2023

Студент		Яковлев В.А.
Руководитель		Проценко И.М.

АННОТАЦИЯ

Целью данной работы является развитие профессиональных навыков в понимании и обработки чужого программного питоновского кода. В данной работе произведено:

- а)Комментирование и тестирование программы
- б)Построение функциональной схемы программы в составе лабораторного комплекса
- в)Приведение кода к стандарту PEP8
- г)Описание Qt Designer среды разработки применительно к техническим требованиям разрабатываемой программы

Благодаря этой колоссальной по сложности работе был заметно улучшен программный код учебного комплекса для проектирования устройств умного дома.

SUMMARY

The purpose of this work is to develop professional skills in understanding and processing someone else's programming Python code. This work produced:

a) Commenting and testing the program

b) Construction of a functional scheme of the program as part of the laboratory complex

c) Bringing the code to the PEP8 standard

d) Description of the Qt Designer development environment in relation to the technical requirements of the program being developed

Thanks to this colossal work, the program code of the educational complex for designing smart home devices has been noticeably improved.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Из-за коллективной разработки программ в современном мире всё большую важность приобретает качество оформления кода. Всё чаще появляется необходимость в проведении рефакторинга старого кода, чтобы упростить его понимание и, следовательно, улучшить его сопровождение. В данной работе были рассмотрены некоторые практики рефакторинга и их применение к реальному программному коду. В качестве реального программного кода был взят код лабораторного программного комплекса для проектирования встроенных приложений умного дома, этот комплекс был разработан Проценко И.М..

1. Комментирование и тестирование программы

1.1. Суть комментирования и тестирования

Комментирование – это одно из важных действий, которые нужно предпринимать над кодом. Без него почти невозможно сопровождать программный продукт. Оно объясняет непонятные в коде вещи.

Еще одним важным действием, которое нужно предпринимать над кодом, является тестирование. Тестирование – это испытания программы с целью проверки соответствия между реальным и ожидаемым поведениями. Обязательными результатами тестирования служат либо выявление полного соответствие между реальностью и ожидаемым результатом, либо своевременное исправление ошибок.

1.2. Примеры комментариев в коде

1)Прокомментировано тело handle_post_reply()

```
@with_err_handling('GET_reply')
def handle_get_reply(self):
    res = self.GET_reply.readAll().data()

    data = json.loads(res)  #    преобразование данных из джейсон

    self.ui.textEdit_message.append(json.dumps(data, separators=(',', ':'))) #  вывод значения в line_edit
```

2)Описана часть тела метода __init__

```
self.ui.pushButton_leds_on.clicked.connect(lambda: self.switch_all(True))#обработка кнопк включения индикаторов
self.ui.pushButton_leds_off.clicked.connect(lambda: self.switch_all(False))#обработка кнопки выключения индикаторов
self.ui.pushButton_leds_color.clicked.connect(self.set_color_all)#обработка кнопки изменения цвета индикаторов
```

1.3. Хронология тестирования программы

1)Запуск программы в Windows:
Программа запускается и корректно работает

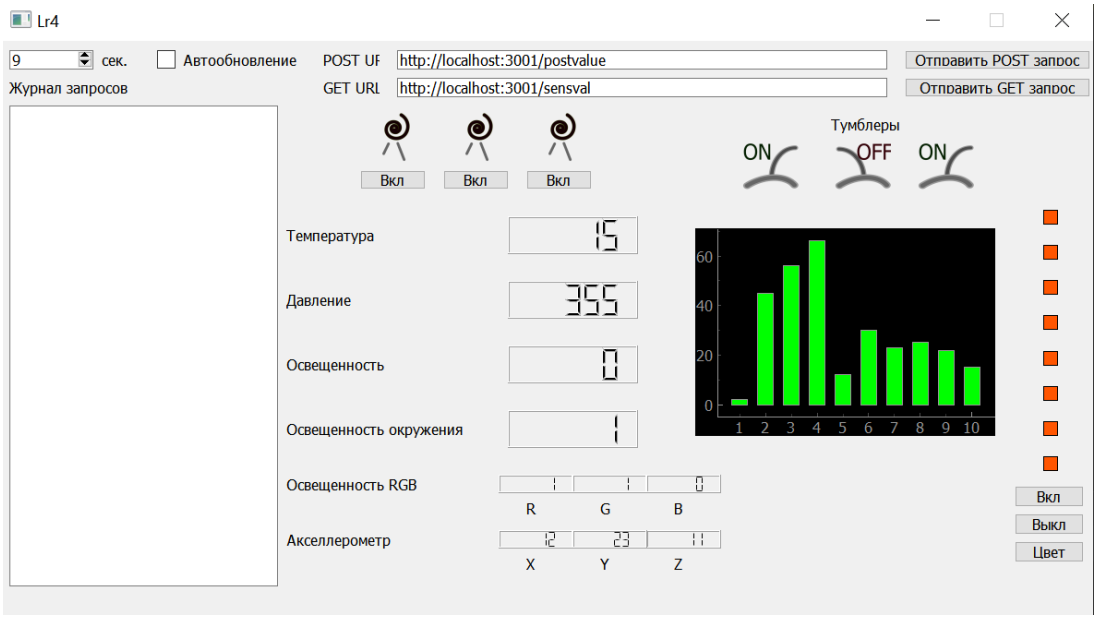

Рис. 1 «Работа программы»

Коллеги успешно протестировали работу программы в операционной системе Linux.

2) Тестирования переименования qt-элементов

Переименование элементов шло в несколько этапов: сначала работа велась в Qt Designer, где часть элементов получили своё второе название, а после этого был изменен код, обрабатывающий элементы из формы, порожденной Qt Designer. Позже, по смысловому пониманию кода, а также из эстетических побуждений, часть названий была вновь изменена в Qt Designer. Такой механизм действий был опасен: Qt Designer не сохраняет прошлые названия элементов и не всегда корректно сохраняет исправления. Поэтому, в случае несовпадения в названиях между визуальной средой и кодом, программа переставала запускаться. Переименование в коде было сделано автоматически с помощью функции Find-Replace

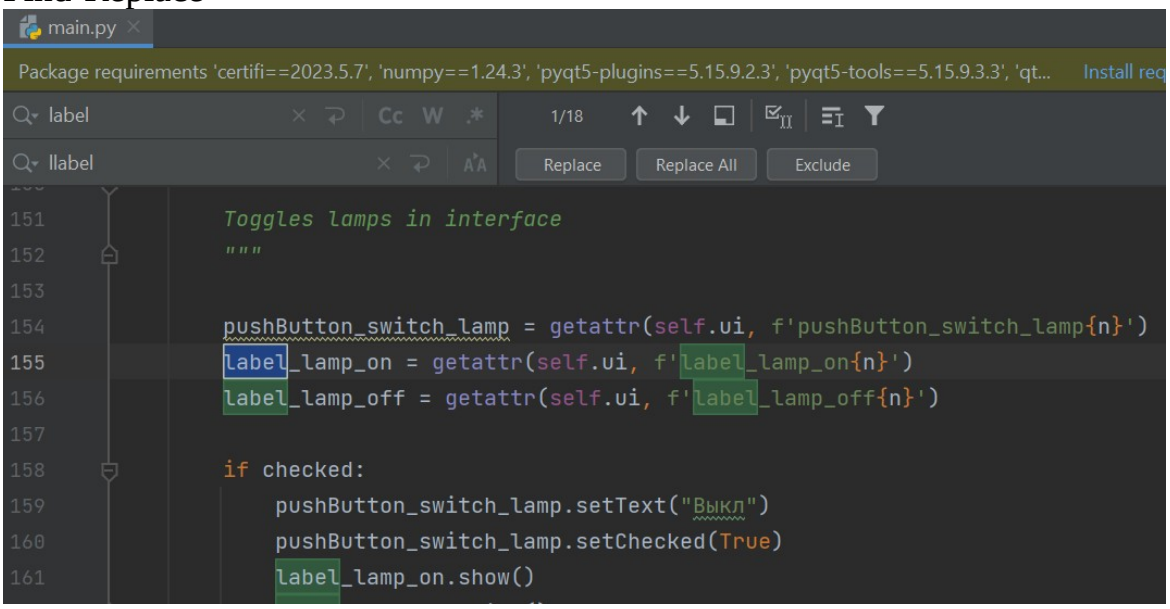

Рис. 2 «Работа Find-Replace»

3)Тестирования работы трёх ламп:
В процессе тестирования было обнаружено, что все три лампы не работают, то есть не включаются и не выключаются. Эта информация была донесена коллегам, которые исправили этот недочет.
4)Тестирование работы индикаторов led
Было выяснено, что кнопка цвет не работает: она не изменяет цвет индикаторов. Мною оперативно было внесено исправление в код

```
98          self.ui.pushButton_leds_color.clicked.connect(lambda: self.set_color_all()) # обработка кнопки изменения цвета индикаторов
```

5)Тестирование работы тумблеров
В ходе опытов было выяснено, что тумблеры отображаются одновременно как включенные и выключенные и не работают, эти сведения были донесены до коллег и тумблеры были починены.

2. Построение функциональной схемы лабораторного комплекса

1.1. Описание функциональной схемы

Функциональная схема лабораторной установки позволяет более быстро понять способ работы лабораторного комплекса. В данной работе за основу была взята функциональной схемы лабораторного комплекса без связи с ПК, в которую был добавлен недостающий элемент.

1.2. Функциональная схема

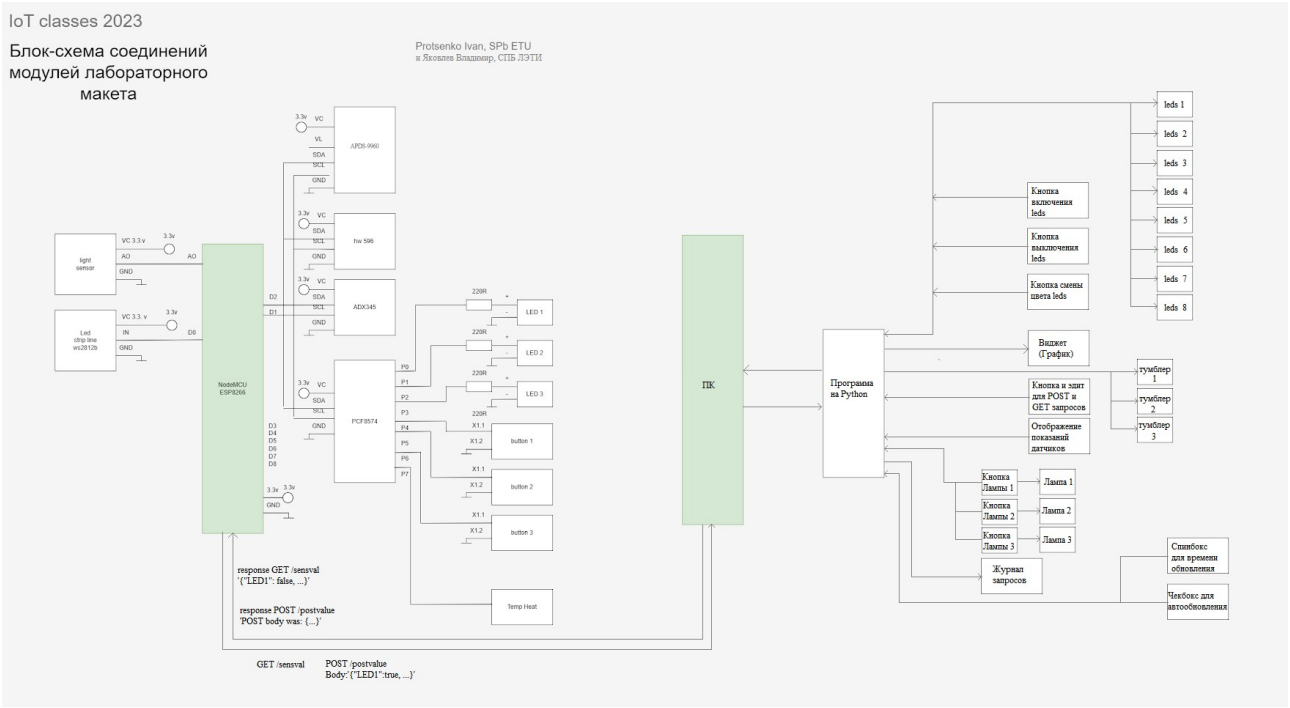

Рис. 3 «Функциональная схема лабораторной установки»

3. Приведение кода к стандарту PEP8

2.1. Основные принципы стандарта PEP8 [1, 2, 3]

1. Правило табуляции: необходимо используйте только четыре пробела на каждом уровне отступа
2. Максимальная длина строки должна ограничиваться максимум 79 символами
3. Комментарии должны быть короткими, содержательными, описательными до и после # должен стоять пробел
4. Функции верхнего уровня и определения классов должны быть отделены двумя пустыми строками.Определения методов внутри класса разделяются одной пустой строкой.Дополнительные пустые строки можно использовать для разделения различных групп похожих функций. Пустые строки могут быть опущены между несколькими связанными однострочными (например, набор фиктивных реализаций).Нужно использовать пустые строки в функциях, чтобы указать логические разделы.
5. Каждый импорт должен быть на отдельной строке и в алфавитном порядке помещен в начало файла, после комментариев к модулю и строк документации, и перед объявлением констант. Импорты должны быть сгруппированы в следующем порядке:
   а)импорты из стандартной библиотеки
   б)импорты сторонних библиотек
   в)импорты модулей текущего проекта
6. Нужно избегать пробелы в
   а)Непосредственно внутри круглых, квадратных или фигурных скобок
   б)Непосредственно перед запятой, точкой с запятой или двоеточием
   в)Сразу перед открывающей скобкой, после которой начинается список аргументов при вызове функции
   г)Сразу перед открывающей скобкой, после которой следует индекс или срез
   д)Использование более одного пробела вокруг оператора присваивания (или любого другого) для того, чтобы выровнять его с другим
7. Всегда окружайте эти бинарные операторы одним пробелом с каждой стороны: присваивания (=, +=, -= и другие), сравнения (==, <, >, !=, <>, <=, >=, in, not in, is, is not), логические (and, or, not).
8. При использовании операторов с разными приоритетами рекомендуется добавить пробелы вокруг операторов с самым низким приоритетом
9. Имена, которые видны пользователю должны следовать конвенциям, которые отражают использование, а не реализацию. Дополнительно стоит учесть, что согласно концепции ООП для названий методов нужно

использовать глаголы.В названиях свойств нужно использовать существительные

2.2. Примеры применения стандарта PEP8 к коду программы

1. Компьютерный код изначально писался в соответствие с правилом табуляции

```
def read_conf() -> dict:
    """
    Utility for reading, modifying and logging config

    открывает файл 'config.json',
    загружает его содержимое в переменную 'conf' в формате словаря (dictionary) при помощи функции 'json.load()',
    а затем выводит все ключи словаря 'conf' при помощи цикла 'for'.
    """

    # Opening JSON file
    f = open('config.json')
    conf = json.load(f)
    f.close()

    # Config logging
    print("Found arguments:")
    for i in conf:
        print(i)

    return conf
```

2. Комментарии писались в соответствие со стандартом

```
        # Setup network management
        self.nam = QNetworkAccessManager()

        # Init plotter
        self.plot = Plot(self.ui.plotwidget, initial_array=conf["temperature"])

        # Set window title
        self.setWindowTitle("Lr4")

        # Init request url editors
        self.ui.lineEdit_POST_URL.setText("http://" + conf['defaultMDNSname'] + conf['defaultPostRoute'])
        self.ui.lineEdit_GET_URL.setText("http://" + conf['defaultMDNSname'] + conf['defaultGetRoute'])
```

3. Импорты расположены в начало файла, на отдельных строках и в алфавитном порядке

```
import typing
from PyQt5 import QtCore, QtGui, QtWidgets #импорт нужных библиотек
from PyQt5 import uic
from PyQt5.QtGui import QColor, QPalette
from PyQt5.QtGui import QPixmap, QMouseEvent
from PyQt5.QtCore import QTimer, QJsonDocument, QUrl, QTime
from PyQt5.QtWidgets import *
from PyQt5.QtNetwork import QNetworkAccessManager, QNetworkRequest, QNetworkReply

import time
import json
import sys
import os

from plot import Plot
import Res_rc
```

4. Названия методов были взяты исходя из того, что они должны следовать конвенциям, которые отражают использование, а не реализацию

```
@with_autosend
def set_color_all(self):
    """
    Sets color using dialog for all RGB LEDs in strip
    """

    color = QColorDialog.getColor()

    if color.isValid():
        palette = QPalette()
        palette.setColor(QPalette.Button, color)
        self.ui.pushButton_leds_color.setPalette(palette)

        for led in self.ui.led_array:
            self.paint_led_color(led, color)
```

5. Так как PEP8 является рекомендацией, а не правилом, то иногда в коде происходили небольшие отступления от него

а. Для забавы был нарушен стандарт комментирования, поэтому пара комментариев не содержательные и длинные

```
            return wrapper  #   вообще-то, имя этого киберспортсмена из hearthstone – Viper

        return inner    #   это не игрок в hearthstone
```

б. Иногда строка кода превышала размер в 79 символов (бывает около 110 символов на строку), это нельзя было поправить, так как тогда нарушится легкость чтения кода

```
self.ui.pushButton_leds_on.clicked.connect(lambda: self.switch_all(True)) # обработка кнопк включения индикаторов
self.ui.pushButton_leds_off.clicked.connect(lambda: self.switch_all(False)) # обработка кнопки выключения индикаторов
self.ui.pushButton_leds_color.clicked.connect(lambda: self.set_color_all()) # обработка кнопки изменения цвета индикаторов
```

4. Описание Qt Designer среды разработки

3.1. Теоретическое описание

Qt Designer – это среда разработки графического интерфейса для компьютерной программы. Qt Designer позволяет создавать элементы управления,располагать их на форме и настраивать их свойства.

В отличие от аналогов(MFC) входит в состав фреймворка Qt, поэтому позволяет пользоваться всеми его компонентами. Ещё одним отличительным преимуществом является кроссплатформенность: может работать как на Windows, так и на Linux операционных системах. С MFC эту среду разработки роднит исповедание парадигмы ООП.

3.2. Практическая часть работы над Qt Designer средой разработки

Было произведено переименование названий элементов формы по их смысловому содержанию:

label_6 = label_temp
label_13 = label_pressure
label_14 = label_illumination
label_15 = label_abient_light
label_16 = label_abient_light
label_34 = label_accelorometer
label_4 = lubel_tumblers
label = label_url
label_2 = label_message
label_7 = label_r
label_8 = label_g
label_9 = label_b
label_10 = label_x
label_11 = label_y
label_12 = label_z

label_24 = label_lamp_off1
label_27 = label_lamp_off2
label_31 = label_lamp_off3

label_20 = label_lamp_on1

label_26 = label_lamp_on2
label_29 = label_lamp_on3

pushButton = pushButton_send_post
pushButton_5 = pushButton_send_get
pushButton_2 = pushButton_switch_lamp1
pushButton_3 = pushButton_switch_lamp2
pushButton_4 = pushButton_switch_lamp3

lcdNumber = lcd_temp
lcdNumber_7 = lcd_lightness
lcdNumber_8 = lcd_abient_light
lcdNumber_2 = lcd_red_light
lcdNumber_3 = lcd_green_light
lcdNumber_4 = lcd_blue_light
lcdNumber_5 = lcd_acceleration_x
lcdNumber_9 = lcd_acceleration_y
lcdNumber_6 = lcd_acceleration_z

vikl_b = pushButton_leds_off
vkl_b = pushButton_leds_on
color_b = pushButton_leds_color

lineEdit = lineEdit_URL
lineEdit_2 = lineEdit_request
textEdit = textEdit_message

on_1 = label_tumbler_on1
on_2 = label_tumbler_on2
on_3 = label_tumbler_on3
off_1 = label_tumbler_off1
off_2 = label_tumbler_off2
off_3 = label_tumbler_off3

Чтобы коллеги могли улучшить сетевой взаимодействие пришлось добавить несколько элементов:

spinBox = spinBox_autoupdate
checkBox = checkBox_autoupdate

Из-за добавленных элементов другие элементы пришлось изменить в размерах и переставить, чтобы добавить пространства.

ЗАКЛЮЧЕНИЕ

Благодаря индивидуальным и командным усилиям удалось справится с рефакторингом программы. Все задачи были успешно выполнены, результатом является более удобный для сопровождения код программы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. PEP 8 - руководство по написанию кода на Python // Python 3 для начинающих. URL: https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html?ysclid=li5r1h0b8m583090094 (дата обращения: 27.05.2023)
2. PEP8: руководство по написанию чистого и читаемого кода на Python // Егоров Егор. URL: PEP8: руководство по написанию чистого и читаемого кода на Python (egorovegor.ru) (дата обращения: 07.06.2023)
3. Методы классов. Параметр self | Объектно-ориентированное программирование Python// selfedu. URL: https://www.youtube.com/watch?v=Lw8TeLS4_IA&list=PLA0M1Bcd0w8zPwP7t-FgwONhZOHt9rz9E&index=3 (дата обращения: 08.06.2023)