Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Работа выполнена в СКБ «Интеллектуальные технологии»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС
_____ Е.М. Димитриади
(подпись)
«13» 05 2023 г.

Декан _____
_____ И.А. Трещёв
(подпись)
«15» 05 2023 г.

УТВЕРЖДАЮ

Проректор по научной работе
_____ А.В. Космынин
(подпись)
«4» 06 2023 г.

«Голосовой ассистент»
Комплект *проектной* документации

Руководитель СКБ _____ 19.05.2023 *Г.В. Москалец*
(подпись, дата)

Руководитель проекта _____ 19.05.2023 *Г.В. Москалец*
(подпись, дата)

Комсомольск-на-Амуре 2023

Карточка проекта

Название	Голосовой ассистент
Тип проекта	Тип проекта: техническое творчество (инициативный)
Исполнители	Студент Д.В. Шутрин – 0ИБ-1 Студент Е. И. Монастырная – 0ИБ-1
Срок реализации	01.10.2022-30.03.2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ЗАДАНИЕ
на разработку

Название проекта: «Голосовой ассистент»

Назначение: Программа предназначена для упрощения жизни пользователей, путем предоставления информации и решения различных задач. Она может выполнять разнообразные функции, такие как включение и выключение бытовых устройств, управление системой освещения, воспроизведение музыки. С помощью голосовых команд, пользователи могут контролировать свое устройство и взаимодействовать с ним, не прибегая к использованию клавиатуры или мыши.
Область использования: Программа может применяться для управление домашней автоматикой, управления различными мультимедийными средствами, выполнения коммуникативных задач
Функциональное описание проекта: Программа, разработанная для упрощения жизни пользователей путем предоставления информации и решения различных задач. Обладает способностью распознавать голосовые команды пользователей преобразовывать их в понятные команды для выполнения различных задач.
Техническое описание устройства: Программа состоит из 15 блоков программного кода основные части которых отчечают за процесс рапознавания речи, процесс написания бота, функции распозвания биометрических данных, управления бытовыми устройствами, мессенджер.
Требования: Intel core 2 DUO и выше, Windows 10/11, не менее 2ГБ свободной памяти
План работ:

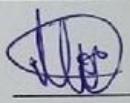
Наименование работ	Срок
Изучение существующих голосовых API	*10.2022*
Опредление основных функций ассистента	*10.2022*
Проктирование общей структуры программы	*11.2022*
Разработка модуля расопзнавания голосовых команд	*11.2022*
Интеграция голсовой платформы	*11.2022*
Написания кода для управления бытовой техникой	*12.2022*
Разработка модуля управления системой овещения	*01.2023*
Разработка модуля воспроизведения музыки	*02.2023*
Интеграция классов в основной код	*03.2023*

Тестирование и отладка	03.2023

Комментарии:

Перечень графического материала:

1. Листинги;
2. Изображения;

Руководитель проекта  20.05.2023 Г.В. Москалец

(подпись, дата)

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

ПАСПОРТ
«Голосовой ассистент»

Руководитель проекта 20.05.2023 *Г.В. Москалец*

(подпись, дата)

Комсомольск-на-Амуре 2023

Содержание

Общие положения

Настоящий паспорт является документом, предназначенным для ознакомления с основными техническими характеристиками, устройством, правилами установки и эксплуатации устройства «Голосовой ассистент» (далее «изделие»).

Паспорт входит в комплект поставки изделия. Прежде, чем пользоваться изделием, внимательно изучите правила обращения и порядок работы с ним. В связи с постоянной работой по усовершенствованию изделия, повышающей его надежность и улучшающей условия эксплуатации, в конструкцию могут быть внесены изменения, не отраженные в данном издании.

1.1 Наименование изделия

Полное наименование изделия – «Голосовой ассистент».

1.2 Наименования документов, на основании которых ведется проектирование изделия

Проектирвание «Голосовой ассистент» осуществляется на основании требований и положений следующих документов:

- задание на разработку.

1.3 Перечень организаций, участвующих в разработке изделия

Заказчиком проекта «Голосовой ассистент» является Федеральное государственное бюджетное образовательное учреждение высшего образования «Комсомольский-на-Амуре государственный университет» (далее заказчик), находящийся по адресу: 681013, Хабаровский край, г. Комсомольск-на-Амуре, Ленина пр-кт., д. 17.

Исполнителями проекта «Голосовой ассистент» являются участники студенческого конструкторского бюро «Интеллектуальные технологии», студенты групп 0ИБ-1Шутрин Дмитрий Васильевич, 0ИБ-1 Монастырная Елизавета Игоревна.

1.4 Сведения об использованных при проектировании нормативно-технических документах

При проектировании использованы следующие нормативно-технические документы:

ГОСТ 2.001-2013. Единая система конструкторской документации. Общие положения.

ГОСТ 2.102-2013. Единая система конструкторской документации. Виды и комплектность конструкторских документов.

ГОСТ 2.105-95. Единая система конструкторской документации. Общие требования к текстовым документам.

ГОСТ 2.610-2006. Единая система конструкторской документации. Правила выполнения эксплуатационных документов.

ГОСТ 2.004-88. Единая система конструкторской документации. Общие требования к выполнению конструкторских технологических документов на печатающих и графических устройствах вывода ЭВМ.

ГОСТ 2.051-2006. Единая система конструкторской документации. Электронные документы. Общие положения.

ГОСТ 2.052-2006. Единая система конструкторской документации. Электронная модель изделия. Общие положения.

ГОСТ 2.601-2013. Единая система конструкторской документации. Эксплуатационные документы.

2 Назначение и принцип действия

2.1 Назначение изделия

Голосовой ассистент

В состав изделия входят:

- Паспорт,
- Программная реализация.

2.2 Области использования изделия

Программа может применяться для управление домашней автоматикой, управления различными мультимедийными средствами, выполнения коммуникативных задач

2.3 Принцип действия изделия

Запуск и активация: Голосовой ассистент начинает свою работу после запуска на устройстве пользователя. Далее происходит распознавание голосовых команд. Голосовой ассистент слушает и записывает аудиофрагменты, чтобы распознать голосовые команды пользователя. Этот процесс включает использование технологий распознавания речи, которые преобразуют аудиофайл в текстовую форму.Полученный текст команды обрабатывается и анализируется голосовым ассистентом. После понимания команды, голосовой ассистент выполняет соответствующие действия.

Голосовой ассистент может взаимодействовать с пользователем, задавать уточняющие вопросы или запрашивать дополнительную информацию, чтобы лучше понять запрос пользователя.

По завершении выполнения задачи или по запросу пользователя, голосовой ассистент готов к новым командам или может перейти в режим ожидания, продолжая слушать и реагировать на голосовые команды.

3 Состав изделия и комплектность

В комплект поставки входит:

- Паспорт,
- Программная реализация.

4 Устройство и описание работы изделия

4.1 Описание работы изделия

Запуск и активация: Голосовой ассистент начинает свою работу после запуска на устройстве пользователя. Далее происходит распознавание голосовых команд. Голосовой ассистент слушает и записывает аудиофрагменты, чтобы распознать голосовые команды пользователя. Этот процесс включает использование технологий распознавания речи, которые преобразуют аудиофайл в текстовую форму.Полученный текст команды обрабатывается и анализируется голосовым ассистентом. После понимания команды, голосовой ассистент выполняет соответствующие действия.

Голосовой ассистент может взаимодействовать с пользователем, задавать уточняющие вопросы или запрашивать дополнительную информацию, чтобы лучше понять запрос пользователя.

По завершении выполнения задачи или по запросу пользователя, голосовой ассистент готов к новым командам или может перейти в режим ожидания, продолжая слушать и реагировать на голосовые команды.

5 Условия эксплуатации

Изделие выпускается в климатическом исполнении УХЛ 4.2 по ГОСТ 15150-69 и предназначен для использования в стационарных условиях в закрытых помещениях при соответствующих климатических условиях:

- интервал температур от +10 до +35 °С;
- относительная влажность воздуха до 80 % при температуре +25 °С;
- высота над уровнем моря не более 2000 м;
- атмосферное давление от 86,6 до 106 кПа (от 650 до 800 мм рт. ст.).

В помещении, где используется изделие не должно возникать условий для конденсации влаги (выпадения росы). *Изделие является электронным прибором, требующим бережного обращения.*

Для обеспечения безотказной работы, сохранения точности и его сбережения необходимо соблюдать следующие правила:

- изучить паспорт, прежде чем приступить к работе с изделием;
- предохранять изделие от ударов и повреждений;
- *при необходимости указать дополнительные пункты*
- не допускать самостоятельную разборку изделия.

5.1 Правила и особенности размещения изделия

Изделие должно быть расположено на расстоянии не менее 1 м от нагревательных приборов.

ВНИМАНИЕ! При эксплуатации изделия запрещается проводить самостоятельно какие-то либо работы по извлечению и установке внутренних компонентов изделия.

5.2 Меры безопасности

Необходимо соблюдать требования техники безопасности и следующие меры предосторожности:

- *не оставлять изделие включенным без наблюдения;*

- *внутренние осмотры и ремонт изделия должны производиться только квалифицированными специалистами;*

5.3 Правила хранения и транспортирования

Транспортирование изделия в упакованном виде может производиться железнодорожным, автомобильным (в закрытых транспортных средствах), воздушным, речным и морским видами транспорта в соответствии с правилами перевозок грузов, действующих на транспорт данного вида. Условия транспортирования изделия по части воздействия климатических факторов должны соответствовать группе 5 по ГОСТ 15150.

После транспортирования изделие должно быть выдержано не менее 2 часов в транспортной таре при температуре 20±5 °С и относительной влажности воздуха не более 80 %.

Распакованное изделие должно храниться в отапливаемом и вентилируемом чистом помещении при температуре от +5 до +40 °С и относительной влажности воздуха не более 60 %. При температуре ниже 25 °С допускается увеличение относительной влажности до 80 %. Воздух в помещении не должен содержать примесей, вызывающих коррозию металлов, налеты на поверхностях оптических деталей.

ПРИЛОЖЕНИЕ А

(обязательное)

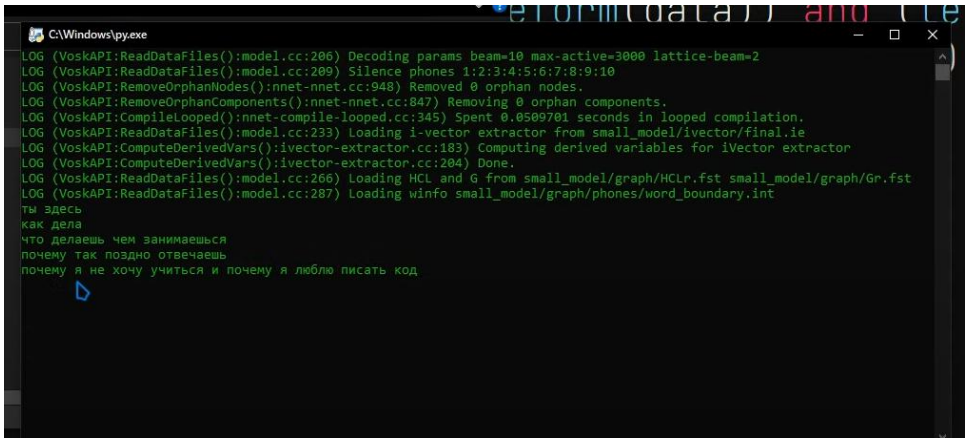


Рисунок А.1 – Процесс распознавания речи

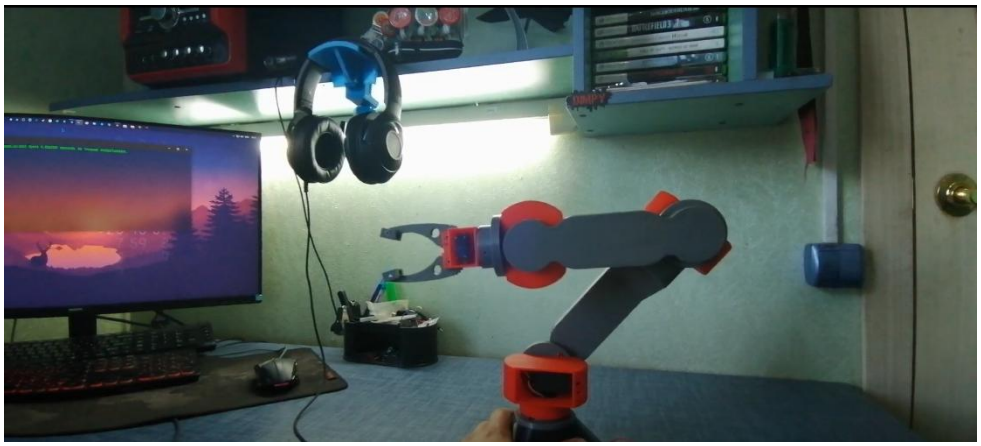


Рисунок А.2 – Управление манипулятором

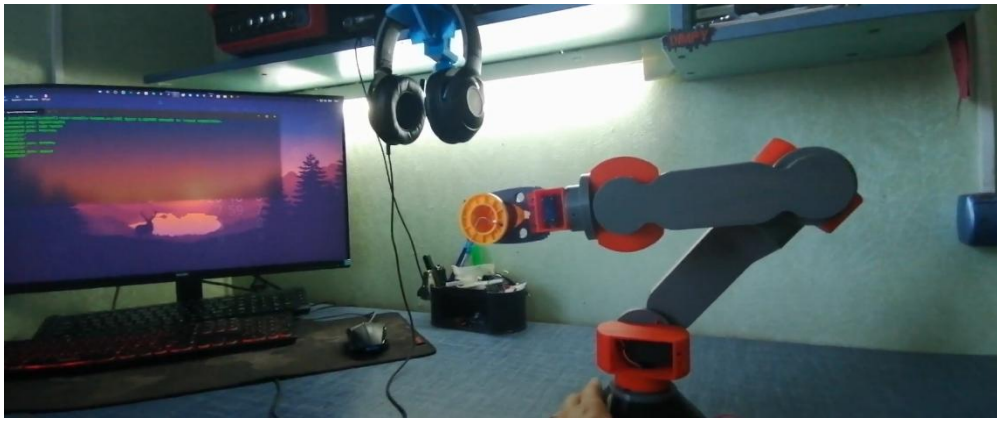


Рисунок А.3 – Захват предмета

ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг Б.1 – Bot_shablons

```
class Shablon:
	def __init__(self):
		pass

	def bot_shablon(self):
		return '''from vk_api.longpoll import VkLongPoll, VkEventType
import vk_api, json
from config import tok

vk_session = vk_api.VkApi(token = tok)
longpoll = VkLongPoll(vk_session)

class User:
	def __init__(self, id, mode):
		self.id = id

def get_keyboard(buts): # функция создания клавиатур
	nb = []
	color = ''
	for i in range(len(buts)):
		nb.append([])
		for k in range(len(buts[i])):
			nb[i].append(None)
	for i in range(len(buts)):
		for k in range(len(buts[i])):
			text = buts[i][k][0]
			color = {'зеленый' : 'positive', 'красный' : 'negative',
'синий' : 'primary', 'белый' : 'secondary'}[buts[i][k][1]]
			nb[i][k] = {"action": {"type": "text", "payload":
"{\"button\": \"" + "1" + "\"}", "label": f"{text}"}, "color": f"{color}"}
	first_keyboard = {'one_time': False, 'buttons': nb}
	first_keyboard = json.dumps(first_keyboard, en-
sure_ascii=False).encode('utf-8')
	first_keyboard = str(first_keyboard.decode('utf-8'))
	return first_keyboard

def sender(id, text, key):
	vk_session.method('messages.send', {'user_id' : id, 'message' : text,
'random_id' : 0, 'keyboard' : key})

users = []

for event in longpoll.listen():
	if event.type == VkEventType.MESSAGE_NEW:
		if event.to_me:

			id = event.user_id
			msg = event.text.lower()

			sender(id, msg.upper(), clear_key)'''

	def cpp_shablon(self):
		return '''#include <iostream>
using namespace std;

int main(){
```

```
    return 0;
}'''
```

Листинг Б.2 - Command_recognizer

```
from fuzzywuzzy.fuzz import ratio as rat
from random import choice
from decs import logging

class Cmd_recognizer:

        def __init__(self):
                pass

        #@logging
        def rearkh(self, mas):
                def rearkh_cycle(ans):
                        nans = ans
                        ans = []
                        for val in nans:
                                if type(val) == list:
                                        for i in val:
                                                ans.append(i)
                                else:
                                        ans.append(val)

                        return ans

                def check(mas):
                        flag = 0
                        for val in mas:
                                if type(val) == list:
                                        flag = 1
                        return flag

                while check(mas):
                        mas = rearkh_cycle(mas)

                return mas

        #@logging
        def clear_cmd(self, global_task):

                for val in ['пожалуйста', 'не могла бы ты', 'прошу']:
                        global_task = global_task.replace(val, '').strip()

                def clear_cmd3(k):
                        tasks = []
                        task = ''
                        if len(k.split()) < 2:
                                return [k]
                        elif k != '':
                                k = k.split()
                                for i in range(-1, len(k)-2):
                                        if (i+1 == len(k)-2):
                                                if k[i+2]:
                                                        task = f'{task} {k[i+1]}
{k[i+2]}'
                                                else:
                                                        task = f'{task} {k[i+1]}'
                                                tasks.append(task.strip())
                                        else:
```

```
                                                for j in range(len(self.keywords)):
                                                        if (rat(k[i+1],
self.keywords[j]) > 80):

        tasks.append(task.strip())
                                                                task = ''
                                                                break
                                                if k[i+2]:
                                                        task = f'{task} {k[i+1]}
{k[i+2]}'
                                                else:
                                                        task = f'{task} {k[i+1]}'
                                for i in range(len(tasks)):
                                        task = tasks[i].split()
                                        for dl in ('уйди', 'свали'):
                                                if task:
                                                        if rat(task[-1], dl) > 70:
                                                                tasks[i] =
(tasks[i].replace(task[-1], '')).strip()
                                                                tasks.append('не
мешай')
                                        if task:
                                                if rat(task[-1], 'мешай') > 80:
                                                        if rat(task[-2], 'не') > 80:
                                                                tasks[i] =
(tasks[i].replace(task[-1], '')).strip()
                                                                tasks[i] =
(tasks[i].replace(task[-2], '')).strip()
                                                                tasks.append('не
мешай')
                                for i in range(len(tasks)):
                                        if tasks[i].endswith(' и'):
                                                ntask = tasks[i].split()
                                                del ntask[len(ntask)-1]
                                                tasks[i] = (' '.join(ntask))
                                for i in range(len(tasks)):
                                        if tasks[i].split():
                                                if rat(tasks[i].split()[-1], 'потом')
> 80:
                                                        ntask = tasks[i].split()
                                                        del ntask[-1]
                                                        tasks[i] = ' '.join(ntask)
                        if (tasks and k):
                                tasks[-1]=f'{tasks[-1]} {k[-1]}'
                        return list(filter(None, tasks))

                def main(k):
                        new_list = clear_cmd3(k)
                        taskss = []
                        for task in new_list:
                                var = clear_cmd3(task)
                                for i in var:
                                        taskss.append(i)

                        new_list = []
                        for task in taskss:
                                var = clear_cmd3(task)
                                for i in var:
                                        new_list.append(i)


                        taskss = []
                        for task in new_list:
                                task = task.split()
```

```
                                if len(task) == 1:
                                        taskss.append(" ".join(task))
                                else:
                                        n_task = ''
                                        for var in range(len(task)):
                                                if (task[var] in self.keywords):
                                                        taskss.append(n_task.strip())
                                                        n_task = task[var]
                                                elif (var == len(task)-1):
                                                        n_task = f'{n_task}
{task[var]}'
                                                        taskss.append(n_task.strip())
                                                else:
                                                        n_task = f'{n_task}
{task[var]}'
                        return [value for value in taskss if value]

                def delete_doubles(ans):
                        for task in range(len(ans)):
                                wars = []
                                last = ''
                                var = ans[task].split()
                                for word in var:
                                        if word != last:
                                                wars.append(word)
                                        last = word
                                ans[task] = ' '.join(wars)
                        return ans

                def task_interpreter(ans):
                        def get_pairs(task):
                                ans = []
                                flag = 0
                                for double in self.double_keys:
                                        pairs = [' '.join(double).strip(), '
'.join(double[::-1]).strip()]
                                        for pair in pairs:
                                                if pair in task:#если пара есть в
запросе
                                                        flag = 1
                                                        var = task.replace(pair,
'').strip()

                                                        if task.startswith(pair):
                                                                ans.append([pair,
var])
                                                        elif task.endswith(pair):
                                                                ans.append([var,
pair])

                                if flag == 0:
                                        ans.append(task)

                                return ans

                        for i in range(len(ans)):
                                ans[i] = get_pairs(ans[i])

                        return self.rearkh(ans)

                ans = delete_doubles(main(global_task))
                ans = self.delete_doubles2(task_interpreter(ans))

                n_ans = [x for x in ans if not(x in self.keywords)]
```

```
                ans = []
                for task in n_ans:
                        flag = 0
                        if task.endswith(' и'):
                                task = (task[::-1].replace('и ', '', 1))[::-1]
                        if task.startswith('и '):
                                task = task.replace('и ', '', 1)

                        for i in [' сначала ', ' и ', ' потом ', 'сделай', 'ладно',
'найди']:

                                if ((i in task) and (not (task.split()[0] in
['найди', 'расскажи', 'добавь', 'удали', 'поздоровайся', 'поприветствуй']))):
                                        if 'найди' in task:
                                                if task.split().index('найди') > 0:
                                                        t1 = '
'.join(task.split()[0:task.split().index('найди')])
                                                        t2 = '
'.join(task.split()[task.split().index('найди'):])
                                                        if t1.startswith('и '):
                                                                t1 = t1.replace('и ',
'').strip()[::-1]
                                                        if t1.endswith(' и'):
                                                                t1 = t1[::-
1].replace('и ', '').strip()[::-1]
                                                        if t2.startswith('и '):
                                                                t2 = t2.replace('и ',
'').strip()[::-1]
                                                        if t2.endswith(' и'):
                                                                t2 = t2[::-
1].replace('и ', '').strip()[::-1]
                                                        ans.append(t1)
                                                        ans.append(t2)
                                                        flag = 1
                                        if len(task.split()) > 1:
                                                if not(task.split()[1] in ['с',
'кто', 'какой', 'как', 'когда', 'где', 'почему', 'от', 'для']):
                                                        task = task.replace(i, '
').strip().replace('  ', '')
                                if rat(task, 'capa') < 70:
                                        task = task.replace('capa', ' ').strip()

                        n_task = task.strip().replace('  ', ' ').split()
                        if len(n_task) > 1:
                                for name in ('capa', 'саров', 'старая', 'стара'):
                                        if name in n_task:
                                                task = task.replace(name, '')

                        if flag == 0:
                                ans.append(task.strip().replace('  ', ''))

                for task in range(len(ans)):
                        j_task = ans[task]
                        while ('спасибо' in ans[task]) and (len(ans[task].split()) >
1):
                                ans[task] = ans[task].split()
                                for var in ["большое", "огромное", "конечно", "те-
бе"]:
                                        for i in range(len(ans[task])-1):
                                                if ans[task][i] == 'спасибо':
                                                        if ans[task][i+1] == var:
                                                                del ans[task][i+1]
                                                        if i > 0:
```

```
                                                        if ans[task][i-1] ==
var:
                                                                del
ans[task][i-1]
                                ans[task] = ' '.join(ans[task])
                                if ans[task] == j_task:
                                        ans[task] = ans[task].replace('спасибо',
'').replace('  ', ' ').strip()
                                        ans.insert(task, 'спасибо')

                if 'capa' in ans:
                        if len(ans) > 1:
                                del ans[ans.index('capa')]
                n_ans = []

                for i in range(len(ans)):
                        if (ans[i].startswith('привет') or
ans[i].endswith('привет')):
                                if ans[i].startswith('привет'):
                                        n_ans.append('привет')
                                        if ans[i].replace('привет', '').strip() !=
'':
                                                n_ans.append(ans[i].replace('привет',
'').strip())
                                if ans[i].endswith('привет'):
                                        if ans[i].replace('привет', '').strip() !=
'':
                                                n_ans.append(ans[i].replace('привет',
'').strip())
                                        n_ans.append('привет')
                        else:
                                if ans[i]:
                                        n_ans.append(ans[i])

                n_ans = self.delete_doubles2(n_ans)
                ans = []
                last = ''
                for task in n_ans:
                        if task != last:
                                ans.append(task)
                                last = task

                tasklist = self.delete_doubles2([value for value in ans if value])
                ultimate_tasks = []

                def check_weather(task):
                        mas = ('подскажи', 'скажи')
                        s_task = task.split()
                        if s_task[0] in (mas):
                                if s_task[1] in ['какая', 'что', 'как']:
                                        if ('погода' in task) or ('температура' in
task):
                                                self.weather()

                nones = choice(['Извините, я прослушала, смотрела в потолок и не
могла оторваться.',
                                                'А сами вы это сделать не можете?',
'Как говорится, если хочешь сделать хорошо, сделай это сам'])

                def check_opener(task):
                        tasks = []
                        names = ('расписание', 'adsense', 'мой канал', 'канал',
'youtube',
                        'ютюб', 'гитхаб', 'гит', 'репозитории', 'vk', 'вконтакте',
```

```
'вк')
                        vals = ['открой', 'зайди', 'открывай', 'заходи', 'запусти',
'запускай']
                        m_vars = [f'открой {name}' for name in names]

                        def chc(st, mas):
                                for val in mas:
                                        if rat(val, st) > 80:
                                                return 1
                                return 0

                        for val in vals:
                                flag = 0
                                if (rat(task.split()[0], val) > 80):
                                        for ts in task.split()[1::]:
                                                j = chc(f'открой {ts}', m_vars)
                                                if (j) and (not(f'открой {ts}' in
tasks)):
                                                        flag = 1
                                                        tasks.append(f'открой {ts}')
                                        if flag:
                                                break
                        if tasks:
                                return tasks
                        else:
                                return task

                for i in range(len(tasklist)):
                        n_t = check_opener(tasklist[i])
                        if n_t != tasklist[i]:
                                tasklist[i] = n_t

                tasklist = self.rearkh(tasklist)
                for i in range(len(tasklist)):
                        if tasklist[i] in self.keywords:
                                tasklist[i] = ''
                tasklist = [x for x in tasklist if x]

                for task in tasklist:

                        if task:

                                s = self.search_exe_function(task)

                                if not(s):

                                        first_word = task.split()[0]
                                        for val in ['можешь', 'давай']:
                                                if rat(first_word, val) > 70:
                                                        if val in task.split():
                                                                if
not(rat(task.split()[task.split().index(val) + 1], 'поработаем') > 70):
                                                                        task =
task.replace(first_word, '', 1).strip()
                                                                        if task:

        first_word = task.split()[0]
                                                                        else:

        first_word = ''
                                        if first_word:

                                                if self.comparison(first_word,
['где']):
```

```
                                                    if len(task.split()) > 1:
                                                            if not(task.split()[1]
in ['ты', 'мы', 'я']):

        self.search_on_map(task.replace(first_word, '').strip())

                                                    elif self.comparison(first_word,
['поищи']) or (self.comparison(first_word, ['найди', 'найти'])):
                                                    if self.not_answer == False:
                                                            n_task = task.split()
                                                            if len(n_task) > 1:
                                                                    flag = 0
                                                                    for val in
['где', 'ближайший', 'поблизости']:
                                                                            if
rat(n_task[1], val) > 70:


                                                                    if flag:

        self.search_on_map(task.replace(first_word, '').strip())
                                                                    else:

        self.web_search(task)
                                                    else:

        self.add_phrases(nones)

                                                    elif check_weather(task):
                                                            if self.not_answer == False:
                                                                    self.weather()
                                                            else:

        self.add_phrases(nones)

                                                    elif self.comparison(first_word,
['расскажи', 'объясни', 'подскажи', 'скажи']):
                                                    if
self.comparison(task.split()[1], ['привет', 'здравствуй', 'здравствуй',
'приветствую']):
                                                            self.greeting(task)
                                                    else:
                                                            if self.not_answer ==
False:

        self.web_talk(task)
                                                            else:

        self.add_phrases(nones)

                                                    elif ('новости' in task) and
not(self.comparison(' '.join(task.split()[0:2]), ['что такое', 'как понять'])):
                                                            self.show_news(task)

                                                    elif (task.startswith('напомни') or
task.startswith('напомнишь')):

        self.create_remind(f"{task.replace('напомни', '').replace('напомнишь',
'')}")

                                                    elif ((self.comparison(first_word,
['создай']) and ('проект' in task.split()[1:])) or (self.comparison('
'.join(task.split()[0:2]), ['давай поработаем'])) or ((self.comparison(first_word,
['открой']) and ('проект' in task.split()[1:])))) and
```

```
not(self.search_exe_function(task)):
                                                        self.create_new_project(task)

                                                elif (self.check_templates(task)):

        self.get_template_from_number(task)

                                                elif self.check_alarm(task):
                                                        self.set_alarm(task)

                                                elif task.startswith(('зайди на',
'заходи на', 'перейди на', 'зайди в', 'заходи в', 'перейди в', 'запусти ')):
                                                        t = task
                                                        for i in ('зайди на', 'заходи
на', 'перейди на', 'зайди в', 'заходи в', 'перейди в', 'запусти '):
                                                                t = t.replace(i,
'').replace('  ', ' ').strip()
                                                        self.opener(t)

                                                elif self.comparison(task.split()[0],
['поздоровайся', 'поприветствуй']):
                                                        self.greeting(task)

                                                else:
                                                        if rat(task.split()[0],
'открой') > 80:
                                                                self.opener(task)

                                                        elif task.split()[0] in
['расскажи', 'объясни', 'подскажи', 'скажи']:
                                                                task = task.split()
                                                                del task[0]
                                                                task = ' '.join(task)

                                                        elif
self.check_get_ideas(task):
                                                                self.get_ideas(task)



                                                        elif
(self.comparison(task.split()[0], ['чем', 'кто', 'что', 'какой', 'который', 'где',
'когда',




                                                                if task.split()[0] in
['что', 'чем']:
                                                                        if (('сегодня'
in task) or ('нового' in task)) and (task.split()[0] == 'что'):

        self.show_news(task)
                                                                        else:

        self.web_talk(f'{task}')
                                                                elif
not(self.search_exe_function(task)):

        self.web_talk(f'{task}')

                                                        recognized_task =
self.search_exe_function(task)

                                                        if recognized_task:
                                                                if not(recognized_task
```

```
in ultimate_tasks):
                                                                    ulti-
mate_tasks.append(recognized_task)

                                else:
                                        ultimate_tasks.append(s)

                ultimate_tasks = self.delete_doubles2(ultimate_tasks)
                for var in self.one_time_executable:
                        if ultimate_tasks.count(var) > 1:

                                ultimate_tasks = ultimate_tasks[::-1]

                                for i in range(ultimate_tasks.count(var) - 1):
                                        del ulti-
mate_tasks[ultimate_tasks.index(var)]

                                ultimate_tasks = ultimate_tasks[::-1]

                return ultimate_tasks
```

Листинг Б.3 – Creator

```
from fuzzywuzzy.fuzz import ratio as rat
from random import choice
from os import getcwd, chdir, mkdir

class File_creator:

        def create_new_project(self, task):
                save_task = task
                nows = ['новый', 'новым', 'очередной', 'очередным', 'создай',
'создадим', 'нового']
                n_task = task.split()
                time = 'old'
                lang = ''
                tz = ''
                cpp_theme = ['си плюс', 'плюсах', 'arduino', 'все плюс плюс']
                python_theme = ['python', 'питоне', 'питона', 'питон', 'на питоне',
'на питание']
                bot_theme = ['vk бота', 'бота vk', 'с ботом', 'бота', 'бота для
быка']
                for val in nows:
                        for i in range(len(n_task)):
                                if rat(val, n_task[i]) > 70:
                                        time = 'new'
                                        del n_task[i]
                                        break

                task = ' '.join(n_task).replace('  ', ' ').strip()
                dels = ['давай создадим ', 'поработаем над ', 'давай', 'поработаем
',
                                'создай ', 'открой ', 'проект ', 'на ', 'для ',
'шаблон ', 'наброски ',
                                'с названием ', 'назови его ', 'под названием ',
'нового ']

                for nd in dels:
                        if nd in task:
                                task = task.replace(nd, '')
                task = task.split()
```

```
                for nd in dels:
                        for i in range(len(task)):
                                if rat(nd, task[i]) > 75:
                                        task[i] = ''

                task = ' '.join(task).replace('  ', ' ').strip()

                if not(task): task = 'standart'

                if time == 'new':
                        if task != 'standart':

                                for val in cpp_theme:
                                        if (val in task) or (self.comparison(val,
task.split()) or self.comparison(val, task.split())):
                                                lang = 'cpp'
                                                task = task.replace(val,
'').replace('  ', ' ').strip()
                                                break

                                for val in python_theme:
                                        if (val in task) or (self.comparison(val,
task.split())):
                                                lang = 'python'
                                                task = task.replace(val,
'').replace('  ', ' ').strip()
                                                break

                                for val in bot_theme:
                                        if (val in task) or (self.comparison(val,
task.split())):
                                                lang = 'python'
                                                tz = 'bot'
                                                task = task.replace(val,
'').replace('  ', ' ').replace('бота', '').replace('бот', '').strip()
                                                break

                                if not(lang in ['cpp', 'python']):
                                        self.engine.Speak(choice(['На каком языке
будет проект?', 'На чём буде писать проект?']))
                                        t_lang = self.fast_get_voice().replace('на',
'').replace('давай', '').replace('можно', '').strip()
                                        for nd in dels:
                                                for i in range(len(t_lang)):
                                                        if rat(nd, t_lang[i]) > 80:
                                                                t_lang[i] = ''
                                        for val in cpp_theme:
                                                if (val in t_lang) or
(self.comparison(val, t_lang.split())):
                                                        lang = 'cpp'
                                                        break
                                        for val in python_theme:
                                                if (val in t_lang) or
(self.comparison(val, t_lang.split())):
                                                        lang = 'python'
                                                        break

                        else:
                                self.engine.Speak(choice(['Как назвать этот про-
ект?', 'Вы уже придумали название проекта?',

                                while not(task) or (task == 'standart'):
                                        name = self.fast_get_voice()
                                        for i in ['назови', 'назови его', 'просто',
```

```
'давай назовём', 'давай', 'да', 'придумал', 'бота', 'нового']:
                                                name = name.replace(i, '').replace('
', ' ').strip()
                                        if name:
                                                task = name.replace(' ', '_')

                for i in ['python', 'питоне', 'питона', 'vk бота', 'бота vk', 'с
ботом', 'бота', 'си плюс', 'плюсах', 'arduino', 'с названием', 'под названием']:
                        task = task.replace(i, '').replace('  ', ' ').strip()

                if not(task):
                        self.engine.Speak(choice(['Как назвать этот проект?', 'Вы
уже придумали название проекта?',
                                'Как вы хотите назвать проект?']))
                        while not(task):
                                name = self.fast_get_voice()
                                for i in ['назови', 'назови его', 'просто', 'давай
назовём', 'давай', 'да', 'придумал', 'бота', 'нового']:
                                        name = name.replace(i, '').replace('  ', '
').strip()
                                if name:
                                        task = name.replace(' ', '_')

                for i in ['под ','с ','под_','с_','с_названием','под_названием']:
                        if task.startswith(i):
                                task = task.replace(i, '', 1)
                        task = task.strip()

                if not(lang in ['cpp', 'python']):
                        self.engine.Speak(choice(['На каком языке будет проект?',
'На чём буде писать проект?']))
                        t_lang = self.fast_get_voice().replace('на ',
'').replace('давай ', '').replace('можно ', '').strip()
                        for nd in dels:
                                for i in range(len(t_lang)):
                                        if rat(nd, t_lang[i]) > 80:
                                                t_lang[i] = ''
                        for val in cpp_theme:
                                if (val in t_lang) or (self.comparison(val,
t_lang.split())):
                                        lang = 'cpp'
                                        break
                        for val in python_theme:
                                if (val in t_lang) or (self.comparison(val,
t_lang.split())):
                                        lang = 'python'
                                        break

                en = {'а' : 'a', 'м' : 'm',
                        'б' : 'b', 'н' : 'n',  'в' : 'v', 'о' : 'o',  'г' : 'g', 'п'
: 'p',  'д' : 'd', 'р' : 'r',
                        'е' : 'e', 'с' : 's',  'ё' : 'yo', 'т' : 't', 'ж' : 'g', 'у'
: 'u',  'з' : 'z', 'ф' : 'f',
                        'и' : 'i', 'х' : 'h',  'й' : 'y', 'ц' : 'tc', 'к' : 'k', 'ч'
: 'ch', 'л' : 'l', 'ш' : 'sh',
                        'щ' : 'sh', 'ы' : 'i', 'ъ' : '', 'ь' : '',           'э' :
'e', 'я' : 'ya',       'ю' : 'yu', ' ' : ' '
                }

                if (time == 'new') and (lang) and (task):
                        try:
                                for i in ['с ','с_','с_названием', 'с ', 'с_',
'с_названием']:
                                        if task.startswith(i):
```

```
                                        task = task.replace(i, '', 1)
                                k = ''
                                for i in range(len(task)):
                                        if not(task[i].isdigit()):
                                                try:
                                                        k = f'{k}{en[task[i]]}'
                                                except:
                                                        k = f'{k}{task[i]}'
                                        else:
                                                k = f'{k}{task[i]}'

                                task = k
                                now_dir = getcwd()
                                chdir(self.projects_dir_path)#перешли в projects
                                mkdir(task.replace(' ', '_'))#создали папку с проек-
том
                                chdir(f'{self.projects_dir_path}/{task.replace(" ",
"_")}')#перешли в папку с проектом

                                mkdir('backups')
                                mkdir('tests')
                                mkdir('main')
                                chdir('main')

                                if lang == 'python':
                                        with open('main.py', 'w', encoding = 'utf-
8') as file:
                                                if tz == 'bot':

        file.write(self.bot_shablon())
                                                else:
                                                        pass

                                        with open('config.py', 'w', encoding='utf-
8') as file:
                                                file.write('tok = ""')

                                elif lang == 'cpp':
                                        with open('main.cpp', 'w', encoding = 'utf-
8') as file:
                                                file.write(self.cpp_shablon())

                                chdir(now_dir)
                                self.add_phrases(choice(['Уже создала!', 'Новый про-
ект создан!']))

                        except Exception as e:
                                self.error_log('', '', e)
                                self.add_phrases('Мне не удалось создать проект')

                else:
                        #print(f'{save_task}\n\t{time}\n\r{lang}\n\t{task}')
                        self.add_phrases(choice(['Я не поняла, что нужно создать!',
'Не уверена, что правльно всё поняла']))
```

Листинг Б.4 – Decs

```
from datetime import datetime
from time import time

def logging(func):
      def wrapper(*args, **kwargs):
            try:
```

```
                res = func(*args, **kwargs)
                if res:
                      return res
          except Exception as e:
                print(f'{e}')
                with open('logs.txt', 'a', encoding = 'utf-8') as file:
                      my_data = str(datetime.now())
                      index = my_data.index('.')
                      file.write(f'\n\n\n\t\t\t-===={datetime.now()}===-\nPhrase:
\nRecognized commands: \nFuction: {funck.__name__}\nError info: {e}\n')
                      file.close()
     return wrapper
```

Листинг Б.5 – Face_rec

```
from face_recognition import load_image_file, face_encodings, face_distance
from os import listdir as ld, getcwd, system as s
from numpy import array
import json, cv2, time

class Camera:
        def __init__(self):
                self.camera = cv2.VideoCapture(0, cv2.CAP_DSHOW)
                for i in range(50):
                        val, image = self.camera.read()
                time.sleep(0.1)

        def get_photo(self):
                for i in range(5):
                        val, image = self.camera.read()
                cv2.imwrite(f"{getcwd()}/photos/who_is.jpg", image)


class FaceID:

        def __init__(self):
                self.camera = Camera()
                self.names = {}
                self.admins = ['Дмитрий Шутрин']
                for name in [x for x in ld(f"{getcwd()}/photos") if not('.' in x)]:
                        self.names.update({name : {}})


        def write(self, data, filename):
                data = json.dumps(data)
                data = json.loads(str(data))
                with open(filename, 'w', encoding = 'utf-8') as file:
                        json.dump(data, file, indent = 4)


        def read(self, filename):
                with open(filename, 'r', encoding = 'utf-8') as file:
                        return json.load(file)


        def load_photos(self):

                saved_names = self.read(f"{getcwd()}/photos/data.json")

                for name in self.names:
                        if name in saved_names:
                                if not(name in self.names):
                                        self.names.update({name : {}})
                                for photo in ld(f"{getcwd()}/photos/{name}"):
```

```
                                        if not(photo in saved_names[name]):
                                                self.names[name].update({photo :
face_encodings(load_image_file(f'{getcwd()}/photos/{name}/{photo}'))[0].tolist()})
                                        else:
                                                self.names[name].update({photo :
saved_names[name][photo]})
                        else:
                                for photo in ld(f"{getcwd()}/photos/{name}"):
                                        self.names[name].update({photo :
face_encodings(load_image_file(f'{getcwd()}/photos/{name}/{photo}'))[0].tolist()})

                self.write(self.names, f"{getcwd()}/photos/data.json")


        def search(self):
                searched_photo = self.get_photo_now()

                if (searched_photo != None):

                        max_coef = 0
                        answer = 'Изображение не найдено!'

                        for name in self.names:
                                for photo in self.names[name]:

                                        data = self.names[name][photo]
                                        coef = 1 - face_distance([array(data)], ar-
ray(searched_photo))[0]

                                        if (coef > max_coef) and (coef > 0.5):
                                                max_coef = coef
                                                answer = name

                        return answer

                else:
                        return 'Изображение не найдено!'


        def get_photo_now(self):#получить изображение человека, сидящего напротив
компа
                try:
                        self.camera.get_photo()
                        return
face_encodings(load_image_file(f"{getcwd()}/photos/who_is.jpg"))[0].tolist()
                except Exception as e:
                        return None


        def access_check(self):
                man = self.search()
                if man in self.admins:
                        return True
                else:
                        return False


if __name__ == "__main__":
        k = FaceID()
        k.load_photos()

        input(k.access_check())
```

Листинг Б.6 – Main

```
from main_class import Assistent

Assistent().run()
```

Листинг Б.7 – Main_class

```
import pygame, vk_api, serial, pymorphy2, os, pyaudio, json
from asyncio import *
from win32com.client import Dispatch as d
from fuzzywuzzy.fuzz import ratio as rat
from threading import Thread
from datetime import datetime
from rutermextract import TermExtractor
from vk_api.longpoll import VkLongPoll
from os import system as s
from serial import Serial
from random import choice, randint
from time import sleep, time
from json import dump, dumps, load, loads
from vosk import Model, KaldiRecognizer
from speedtest import Speedtest
from decs import logging
from relays import Relay

from command_recognizer import Cmd_recognizer
from virtual_class import VirtualAssistent
from sub_class import Sub_class
from vk_messanger import Vk_messanger, MyLongPoll
from manipulator import Manipulator
from face_rec import FaceID, Camera


class Assistent(VirtualAssistent, Cmd_reccognizer, Vk_messanger, MyLongPoll,
Sub_class):

        #@logging
        def __init__(self):

                #self.init_cmds()
                self.double_keys = [['как', 'дела'], ['как', 'настроение'], ['как',
'жизнь'], ['как', 'живется'], ['что', 'делаешь'], ['чем', 'занимаешься'], ['что',
'тупишь'], ['что', 'тормозишь']]
                self.ard_error_msgs = ['мне не удалось подкючиться к arduino', 'я не
смогла подкючиться к arduino']
                self.vk_session = vk_api.VkApi(token =
'4416f0ed959db32fda62da0bd76a464b78b87fc014b5964322eb1f1043682f75979bb300d9b17bf11f9d
b')
                self.longpoll = MyLongPoll(self.vk_session)
                self.projects_dir_path = 'D:/Programming/projects'
                self.music_path = 'D:/Programming/music'
                pygame.mixer.init()
                self.volume = 0.0125
                self.volume_setup()
                self.phrases = []
                self.com_port = 'COM3'
                self.create_connection()
                self.robot = Manipulator()
                self.robot.set_serial(self.ser)
                self.morph = pymorphy2.MorphAnalyzer()
                self.news = {'1d' : [], '10d' : []}
```

```
                self.ext = TermExtractor()
                self.not_answer = False
                self.last_phrase = ''
                self.last_time = None
                self.last_sender = -1
                self.engine = d("SAPI.SpVoice")
                self.rem_lock = False
                self.need_get_voice = True
                self.premission_to_question = True
                self.facer = FaceID()
                self.facer.load_photos()

                self.relays = {
                                                'чайник' : Relay(self.ser, 'чайник',
1),
                                                'лампа' : Relay(self.ser, 'лампа', 2)
                                        }

                self.init_cmds()
                self.inet_tester = Speedtest()
                Thread(target = self.check_messages, daemon = True).start()#обрабочик
вк
                Thread(target = self.remind_thread, daemon = True).start()#обработчик
напоминаний
                Thread(target = self.check_need_alarm, daemon =
True).start()#обработчик будильников

                self.Type = 2
                #для vosk
                if self.Type == 2:
                        self.add_phrases('Начинаю загрузку речевой модели')
                        self.talk()
                        self.model = Model("model")
                        self.rec = KaldiRecognizer(self.model, 48000)
                        self.p = pyaudio.PyAudio()
                        self.stream = self.p.open(format=pyaudio.paInt16, channels=1,
rate=48000, input=True, frames_per_buffer=8000)
                        self.stream.start_stream()
                        s('cls')
                        self.add_phrases('Речевая модель загружена!')
                #для vosk

                self.hello()
                self.talk()


        def listen(self):
                while True:
                        self.talk()# ассистент говорит все фразы из очереди(по сути не
относится к распознаванию)
                        if self.need_get_voice:
                                data = self.stream.read(4000, excep-
tion_on_overflow=False)
                                if (self.rec.AcceptWaveform(data)) and (len(data) !=
0):

                                        answer=json.loads(self.rec.Result())
                                        if answer["text"]:# Если что-то распознано

                                                if answer["text"].lower() in ('эй',
'сара', 'ты здесь', 'ты тут', 'слышь', 'слышишь', 'сара ты тут', 'сара ты здесь'):
                                                        self.first_ans()# обход распо-
знавания обращения к ассистенту
```

```
                                                else:# возвращаем распознанный текст
                                                        yield answer["text"].lower()

                                        else:# если нам ничего не сказали, проверяем,
можем ли мы задать вопрос
                                                if self.premission_to_question == True:
                                                        self.check_need_question()


        def run(self):
                while True:
                        try:
                                text = ''
                                for text in self.listen():
                                        #text = input('>>> ')
                                        is_admin = self.facer.access_check()
                                        out_commands = []
                                        print(f'{self.magenta}Распознанный
текст:\n\t\t{self.green}<{self.white}{text}{self.green}>')
                                        if is_admin:
                                                out_commands = self.clear_cmd(text)
                                                print(f'{self.magenta}Распознанные ко-
манды: {self.white}[')
                                                for command in out_commands:

        print(f'\t\t{self.green}<{self.white}{command}{self.green}>')
                                                print(f'{self.white}]')
                                                if out_commands:
                                                        self.cmd_exe(out_commands)
                                        else:
                                                print(f'{self.red}Вам отказано в досту-
пе, вы не являетесь админом!{self.white}')
                                                self.refusal()

                        except Exception as error:
                                print(f'Error: {error}')
                                self.error_log(text, out_commands, error)
                                self.add_phrases(f"Произошла ошибка! {choice(['файл
ошибок обновлён', 'проверьте файл ошибок', ''])}")
                                self.talk()
```

Листинг Б.8 – Manipulator

```
from servo import Servo
import serial, time

class Manipulator:

        def __init__(self):
                self.servs = [Servo('', i) for i in range(1, 6)]


        def set_serial(self, ser):
                self.ser = ser
                for i in range(1, 6):
                        self.servs[i-1] = Servo(self.ser, i)


        def take(self):
                self.servs[4].go_to(50)


        def let_go(self):
                self.servs[4].go_to(160)
```

```
        def down(self):
                self.servs[1].go_to(120)
                self.servs[2].go_to(180)


        def up(self):
                for serv in range(len(self.servs)-1):
                        self.servs[serv].go_to(90)


        def low_down(self):
                self.servs[2].go_to(200)


        def low_up(self):
                self.servs[2].go_to(300)


        def high(self):
                self.servs[1].go_to(300)
                self.servs[2].go_to(300)


        def low(self):
                self.servs[1].go_to(200)
                self.servs[2].go_to(200)
                self.servs[2].go_to(200)
                self.servs[2].go_to(200)


        def rotate_hand(self):
                self.servs[3].go_to(150)


        def static_hand(self):
                self.servs[3].go_to(90)
```

Листинг Б.9 – Relays

```
class Relay:

      def __init__(self, ser, task, code):
            self.code = b'0' * (2-len(str(int(code)))) + b'%d' % code
            self.task = task
            try:
                  if ser:
                        self.ser = ser
                        self.value = b'001'
                        self.is_work = True
                  else:
                        self.is_work = False
            except Exception as e:
                  self.is_work = False

      def turn_on(self):
            if self.is_work:
                  self.value = b'000'
                  self.write()


      def turn_off(self):
```

```
            if self.is_work:
                  self.value = b'001'
                  self.write()

      def write(self):
            if self.is_work:
                  ans = b'1%b%b\n\r' % (self.code, self.value)
                  self.ser.write(ans)
```

Листинг Б.10 – Remind_class

```
from random import choice
from datetime import datetime, timedelta
from os import system

class Reminder:
      def check_time(self, num):
            if int(datetime.now().strftime("%H")) <= num:
                  hours = num - int(datetime.now().strftime("%H"))
                  minutes = 0 - int(datetime.now().strftime("%M"))
                  kk = datetime.now() + timedelta(hours = hours, minutes = minutes)
            else:
                  hours = num + (24 - int(datetime.now().strftime("%H")))
                  minutes = 0 - int(datetime.now().strftime("%M"))
                  kk = datetime.now() + timedelta(hours = hours, minutes = minutes)
            return str(kk)
      def create_remind(self, task):

            task = task.strip()

            if task.startswith('через '):
                  task = task.replace('через ', '')
                  if ('часов' in task) or ('часа' in task):
                        if 'часов' in task:
                              task = task.split('часов')
                        else:
                              task = task.split('часа')

                        for i in range(len(task)):
                              task[i] = task[i].strip()

                        task[0] = self.get_digit_from_chars(task[0])
                        out_date = datetime.now() + timedelta(hours = task[0])
                        out_date = f'{str(out_date)} {task[1]}'
                        file              =              open('D:/Programming/smart
house/new/main_folder/reminds.txt', 'a', encoding = 'UTF-8')
                        file.write(f'{out_date}\n')
                        file.close()

                  elif ('минут' in task):
                        task = task.split('минут')

                        for i in range(len(task)):
                              task[i] = task[i].strip()

                        out_date    =    datetime.now()    +    timedelta(minutes    =
self.get_digit_from_chars(task[0]))
                        out_date = f'{str(out_date)} {task[1]}'
                        file              =              open('D:/Programming/smart
house/new/main_folder/reminds.txt', 'a', encoding = 'UTF-8')
                        file.write(f'{out_date}\n')
                        file.close()
```

```
            elif task.startswith('вечером '):
                  task = task.replace('вечером ', '')
                  out_date = f'{self.check_time(19)} {task}'
                  file                 =                 open('D:/Programming/smart
house/new/main_folder/reminds.txt', 'a', encoding = 'utf-8')
                  file.write(f'{out_date}\n')
                  file.close()


            elif task.startswith('днём '):
                  task = task.replace('днём ', '')
                  out_date = f'{self.check_time(15)} {task}'
                  file                 =                 open('D:/Programming/smart
house/new/main_folder/reminds.txt', 'a', encoding = 'UTF-8')
                  file.write(f'{out_date}\n')
                  file.close()

            elif task.startswith('с утра ') or task.startswith('завтра '):
                  task = task.replace('с утра ', '')
                  task = task.replace('завтра', '')
                  date   =   (str(datetime.today().strftime("%Y   %m   %d   %H   %M
%S"))).split(' ')
                  for i in range(len(date)):
                        date[i] = int(date[i])

                  if (int(datetime.now().strftime("%H")) >= 6):
                        date[2] += 1;     date[3] = 7
                        date[4] = 0;      date[5] = 0
                        rem  =  f'{datetime(date[0],  date[1],  date[2],  date[3],
date[4], date[5])} {task}\n'

                  else:
                        date[3] = 7;      date[4] = 0
                        date[5] = 0
                        rem  =  f'{datetime(date[0],  date[1],  date[2],  date[3],
date[4], date[5])} {task}\n'

                  file                 =                 open('D:/Programming/smart
house/new/main_folder/reminds.txt', 'a', encoding = 'UTF-8')
                  file.write(f'{rem}')
                  file.close()


            self.add_phrases(choice(['Хорошо, я напомню', 'Постараюсь не забыть',
'Хорошо, я всё записала!']))
```

Листинг Б.11 – Servo

```
import serial, time

class Servo:

      def __init__(self, ser, code, debug=False):
            self.debug = debug
            self.code = self.normalized(code, 2)
            self.standart_rotate = 10
            try:
                  if ser:
                        self.ser = ser
                        self.value = 90
                        self.is_work = True
                  else:
```

```
                        self.is_work = False
            except Exception as e:
                  self.is_work = False

      def up(self):
            if self.is_work:
                  val = 3-len(str(int(self.value) + self.standart_rotate))
                  val2 = b'%d' % (int(self.value) + self.standart_rotate)
                  self.value = b'%b' % (b'0' * val + val2)
                  self.write()


      def down(self):
            if self.is_work:
                  val = 3-len(str(int(self.value) - self.standart_rotate))
                  val2 = b'%d' % (int(self.value) - self.standart_rotate)
                  self.value = b'%b' % (b'0' * val + val2)
                  self.write()

      def left(self):
            if self.is_work:
                  self.value = b'180'
                  self.write()

      def right(self):
            if self.is_work:
                  self.value = b'000'
                  self.write()


      def write(self):
            if self.is_work:
                  ans = b'2%b%b\n\r' % (self.code, self.value)
                  if self.debug:
                        pass
                  else:
                        self.ser.write(ans)

      def normalized(self, value, ln):
            if len(str(value)) < ln:
                  value = (b'0' * (ln-len(str(value)))) + b'%d' % value
                  return value
            else:
                  return b'%d' % value


      def go_to(self, value):
            if self.is_work:
                  self.value = self.normalized(value, 3)
                  self.write()
```

Листинг Б.12 – Sub_class

```
from colorama import Fore

class Sub_class:
      def init_cmds(self):

            self.ideas = {
                  'творчество' : [
                        'порисовать', 'почитать', 'сделать публикацию', 'спеть ка-
кую-нибудь песню',
                        'начать читать какую-нибудь книгу о путешествиях, космосе
```

```
или волшебстве'
                ],
                'здоровье' : [
                        'позаниматься спортом', 'поприседать', 'поотжиматься',
'выйти на пробежку', 'поиграть в баскетбол',
                        'поиграть в волейбол', 'пооиграть в футбол', 'поиграть в
теннис', 'сходить в тренажёнрый зал', 'сделать планку'
                ],
                'дома' : [
                        'убраться в квартире', 'приготовить поесть', 'приготовить
еду', 'сделать влажную уборку',
                        'передвинуть мебель', 'Приготовить что-то интересное по ре-
цепту какого-нибудь кулинарного гуру',
                        'Почистить компьютер и смартфон от ненужных приложений',
'Сфотографировать ненужные вещи и выложить их на Avito',
                        'Сделать своими руками уникальное украшение интерьера',
'Сделать уборку. Например, по японским принципам кайдзен'
                ],
                'улица' : [
                        'Побывать в лесу', 'посмотреть на звезды', 'Пить фруктовые
коктейли', 'Устроить пикник',
                        'Покататься на роликах и велосипеде', 'Посетить крутое
мероприятие', 'Прогуляться в парке',
                        'Пообедать в кафе на открытом воздухе', 'Посмотреть на ноч-
ной город с крыши',
                        'Почувствовать единение с природой', 'поулять, поедая
мороженное', 'Отправиться в поход',
                        'Познакомиться с кем-нибудь на улице'
                ],
                'учёба и работа' : [
                        'Выбрать интересный онлайн-курс и прослушать его',
'почитать и разобраться в том, что вам нужно сделать',
                        'проверьте, все ли задания вы выполнили'
                ],
                'организация' : [
                        'Составить список целей на ближайший месяц, полгода, год',
'Разработать идеальный план организации своего времени',
                        'посмотреть мотивационные видео или треннинги', 'Собрать
новый музыкальный плейлист'
                ],
                'общение' : [
                        'Вспомнить старого друга, с которым давно не общались',
'Встретиться с подругой',
                        'Поиграть с друзьями или детьми в настольные игры',
'Написать сообщение родственникам',
                        'собраться с друзьями'
                ],
                'кампания' : [
                        'Посоревноваться: кто сложит самый далеколетающий бумажный
самолётик',
                        'сыграть в мафию вместе с друзьями', 'сыграть в крокодила
вместе с друзьями',
                        'приклеить стикеры на лоб и угадывать слова вместе с
друзьями',
                        'угадывать название песни по её отрывку'
                ],
                'нестандарное' : [
                        'Научиться завязывать галстук или шарф десятком стильных
способов',
                        'Пройтись по магазинам просто так и мысленно покупать все,
что захочется',
                        'Зарегистрироваться на бирже труда и написать фейковую ан-
кету, какой вы крутой. Только телефон не давайте!',
                        'Сбросить настройки смартфона на заводские и потом начать
```

```
все заново',
                        'Купить комнатное растение и сделать вид, что ухаживаете
теперь за ним',
                        'Станцевать мумбу-юмбу. Не? Не знаете? Просто пляшите',
                        'Построить из бумаги макет своего будущего дома или виллы',
                        'Плюнуть в потолок 10 раз. Понять, что плевки не долетают.
Сходить, умыться',
                        'Умываясь под краном, поблагодарить за то, что вода есть у
вас в доме',
                        'Купить шоколад, который еще никогда не пробовали, и
съесть',
                        '10 минут смотреть в окно. Там всегда что-то происходит',
                        'Научиться ложиться рано и вставать с первыми лучами
солнца', 'Научиться делать лимонад',
                        'Решиться на новое, потому что если не сейчас, то когда?',
                        'Скачать аудиогид по собственному городу и отправиться на
экскурсию'
                  ],
                  'отдых' : [
                        'Сесть перед окном с чашечкой кофе и наблюдать за прохожи-
ми, листьями и облаками',
                        'Включить лёгкий расслабляющий фильм — о чём-нибудь добром
и без лишних вывертов',
                        'Просмотреть скопившиеся фотографии и видео', 'Задремать на
полчаса. Иногда помогает',
                        'Выбрать подходящий фон и наслаждаться звуками природы',
                        'Сделать много селфи, выбрать лучшие и обновить аватарки в
мессенджерах и соцсетях',
                        'Сходить в кино', 'Попробовать новые травяные чаи',
'посмотреть фильм колдун'
                  ]
            }

            self.magenta = Fore.MAGENTA
            self.red = Fore.RED
            self.white = Fore.WHITE
            self.green = Fore.GREEN

            self.alias = {
                  1 : ['одну', 'один'],  11 : ['одиннадцать'],  21  :  ['двадцать
один', 'двадцать одну'],     31 : ['тридцать один', 'тридцать одну'], 41 : ['сорок
один', 'сорок одну'],  51 : ['пятьдесят один', 'пятьдесят одну'],
                  2 : ['две'],                  12 : ['двенадцать'],   22        :
['двадцать две'],                                32 : ['тридцать две'],
                        42 : ['сорок две'],                   52  :  ['пятьдесят
две'],
                  3 : ['три'],                  13 : ['тринадцать'],   23        :
['двадцать три'],                                33 : ['тридцать три'],
                        43 : ['сорок три'],                   53  :  ['пятьдесят
три'],
                  4 : ['четыре'],               14 : ['четырнадцать'], 24        :
['двадцать четыре'],                             34 : ['тридцать четыре'],
                  44 : ['сорок четыре'],                54    :    ['пятьдесят
четыре'],
                  5 : ['пять'],                 15 : ['пятнадцать'],   25        :
['двадцать пять'],                               35 : ['тридцать пять'],
                  45 : ['сорок пять'],                  55 : ['пятьдесят пять'],
                  6 : ['шесть'],                16 : ['шестнадцать'],  26        :
['двадцать шесть'],                              36 : ['тридцать шесть'],
                  46 : ['сорок шесть'],                 56    :    ['пятьдесят
шесть'],
                  7 : ['семь'],                 17 : ['семнадцать'],   27        :
['двадцать семь'],                               37 : ['тридцать семь'],
                  47 : ['сорок семь'],                  57 : ['пятьдесят семь'],
```

```
                8 : ['восемь'],              18 : ['восемнадцать'], 28        :
['двадцать восемь'],                         38 : ['тридцать восемь'],
                48 : ['сорок восемь'],                  58     :     ['пятьдесят
восемь'],
                9 : ['девять'],              19 : ['девятнадцать'], 29        :
['двадцать девять'],                         39 : ['тридцать девять'],
                49 : ['сорок девять'],                  59     :     ['пятьдесят
девять'],
                10 : ['десять'],       20 : ['двадцать'],           30        :
['тридцать'],                                40 : ['сорок'],
                     50 : ['пятьдесят'],                   60               :
['шестьдесят']
            }


            self.keywords = [
                        'включи',         'свали',          'включай',
      'вруби',          'врубай',         'подключи',       'дай',
                        'выключи',        'выключай',       'отключи',
      'отключай',       'открой',         'вырубай',        'подскажи'
                        'открывай',       'запусти',        'запускай',
      'умница',         'помоги',         'поставь',        'уйди'
                        'расскажи',       'скажи',          'объясни',
      'рассказывай',    'объясняй',       'напомнишь',      'зайди'
                        'здаров',         'здравствуй',     'подключай',
      'отруби',         'создай',         'напомни']


            self.strs = {
                  1 : 'первое', 2 : 'второе', 3 : 'третье', 4 : 'четвёртое', 5 :
'пятое', 6 : 'шестое', 7 : 'седьмое', 8 : 'восьмое', 9 : 'девятое', 10 : 'десятое',
                  11 : 'одиннадцатое', 12 : 'двенадцатое', 13 : 'тринадцатое', 14 :
'четырнадцатое', 15 : 'пятнадцатое', 16 : 'шестнадцатое', 17 : 'семнадцатое',
                  18 : 'восемнадцатое', 19 : 'девятнадцатое', 20 : 'двадцатое', 21
: 'двадцать первое', 22 : 'двадцать второе', 23 : 'двадцать третье',
                  24 : 'двадцать четвёртое', 25 : 'двадцать пятое', 26 : 'двадцать
шестое', 27 : 'двадцать седьмое', 28 : 'двадцать восьмое', 29 : 'двадцать девятое',
                  30 : 'тирдцатое', 31 : 'тридцать первое'
            }


            self.one_time_executable = [
                  ('не мешай', 'уйди', 'свали', 'отвали', 'я ушел', 'пока',
'уходи', 'прощай', 'отстань', 'хватит'), ('как дела', None),
                  ('здаров', 'привет', 'здравствуй'), ('поздоровайся',
'поприветствуй всех', 'поздоровайся со всеми'),
                  ('что делаешь', 'чем занимаешься'), ('день недели', 'какой день
недели', 'какой сегодня день недели'),
                  ('завершай работу', 'давай спать', 'выруби систему', 'выруби
всё'), ('сара', 'саров', 'старая', 'стара'),
                  ('заблокируй компьютер', 'блокировка', 'заблокируйся',
'заблокируй экран'), ('спасибо', 'благодарю')
            ]


            self.is_answer = [
                  ('не мешай', 'уйди', 'свали', 'отвали', 'я ушел', 'пока',
'уходи', 'прощай', 'отстань', 'хватит'),
                  ('здаров', 'привет', 'здравствуй'), ('что делаешь', 'чем
занимаешься'),
                  ('поздоровайся', 'поприветствуй всех', 'поздоровайся со всеми'),
                  ('спасибо', 'благодарю'), ('как у тебя дела', 'как
настроение', 'как жизнь'),
                  ('молодец', 'умница', 'какая ты молодец', 'какая умница'), ('как
```

```
дела', None),
                ('день недели', 'какой день недели', 'какой сегодня день
недели'),
                ('сколько время', 'время', 'сколько времени'), ('сара', 'саров',
'старая', 'стара'),
                ('число', 'какое сегодня число', 'какое число'),
                ('повтори', 'ещё раз', 'повтори ещё раз', 'повтори ещё раз', 'ну
ка ещё раз', 'ну ка повтори'),
                ('вау', 'ого', 'ничего себе', 'это круто', 'круто', 'обалдеть',
'я в шоке')
            ]


            self.conversational = {
                ('уйди', 'свали', 'отвали', 'я ушел', 'пока', 'уходи', 'прощай',
'отстань', 'хватит', 'мне пора', 'отключись', 'выключись') : self.bye,
                ('здаров', 'привет', 'здравствуй') : self.hello,
                ('как дела', None) : self.how_are_you,
                ('спасибо', 'благодарю') : self.thanks,
                ('что делаешь', 'чем занимаешься') : self.what_doing,
                ('что ты умеешь', 'что ты умеешь делать', 'что ты можешь', 'чем
ты можнш помочь'): self.info_skills,
                ('вау', 'ого', 'ничего себе', 'это круто', 'круто', 'обалдеть',
'я в шоке') : self.surprise,
                ('молодец', 'умница', 'какая ты молодец', 'какая умница',
'хорошо') : self.well_done,
                ('ты дура', 'ты тупая', 'не тупи', 'хватит тупить', 'харе
тупить', 'что тупишь', 'ты надоела', 'ты задолбала', 'ты идиотка', 'не тормози',
'хватит тормозить', 'харе тормозить', 'что тормозишь') : self.bad_answers,
                ('что ты умеешь', 'что ты умеешь делать', 'что ты можешь', 'чем
ты можнш помочь') : self.info_skills,
                ('кто тебя создал', 'кто твой создатель', 'кто тебя сделал', 'кто
тебя написал') : self.autor_info,
                ('повтори', 'ещё раз', 'повтори ещё раз', 'повтори ещё раз', 'ну
ка ещё раз', 'ну ка повтори') : self.repeat,
                ('как у тебя дела', 'как настроение', 'как жизнь') :
self.how_are_you,
                ('не мешай', 'помолчи', 'замолчи', 'не задавай вопросов', 'не за-
давай глупые вопросы', 'я не хочу разговаривать') : self.lock_questions,
                ('давай поговорим', 'можно поговорить', 'давай поболтаем',
'спроси что-нибудь') : self.unlock_questions
            }


            self.ard = {
                ('установи соединение', 'соеденись с ородруина', 'установи соеди-
нение с ородруина', 'обнови ородруина', 'перезагрузи ородруина', 'что с ородруина',
'проверь ородруина', 'подключи ородуина') : self.create_connection,
                ('держи', 'подержи', 'возьми', 'возьми ка') : self.r_take,
                ('отпусти', 'отдай') : self.r_let_go,
                ('поднимись', 'выпрямись') : self.r_up,
                ('опустись', 'упади') : self.r_down,
                ('ниже', 'наклонись') : self.r_low,
                ('выше', 'подними выше') : self.r_high,
                ('немного выше', 'чуть выше') : self.r_low_up,
                ('немного ниже', 'чуть ниже') : self.r_low_down,
                ('поверни на меня', 'поверни от меня', 'вращай') : self.r_rotate,
                ('выпрями руку', 'держи ровно', 'поверни ровно') :
self.r_static_hand,
                ('включи чайник', 'поставь чайник', 'я хочу чай') :
self.relays['чайник'].turn_on,
                ('выключи чайник', 'отключи чайник', 'не хочу чай') :
self.relays['чайник'].turn_off,
                ('включи свет', 'что так темно', 'как-то темно', 'включи лампу')
```

```
: self.relays['лампа'].turn_on,
                ('выключи свет', 'свет мешает', 'выключи лампу') :
self.relays['лампа'].turn_off,
                ('выключи все розетки', 'выключи всё') : self.all_off
            }


            self.helpers = {
                ('какая сейчас громкость', 'какой сейчас уровень громкости',
'какой уровень громкости стоит') : self.vol_level_now,
                ('день недели', 'какой день недели', 'какой сегодня день недели')
: self.day_of_week,
                ('сколько время', 'время', 'сколько времени', 'сколько сейчас
времени') : self.time_now,
                ('число', 'какое сегодня число', 'какое число') :
self.date_is_today,
                ('что с интернетом', 'какая скорость интернета') :
self.check_internet,
                ('очисти экран', 'очистить экран', 'очисти консоль', 'очистить
консоль') : self.clear_screen,
                ('очисти логи', 'очисти ошибки') : self.clear_logs,
                ('сохранись', 'сохрани все на сервер', 'отправь все на гитхаб',
'сохрани все') : self.save
            }


            self.computer_manipulation = {
                ('завершай работу', 'давай спать', 'выруби систему', 'выруби
всё', 'выключи компьютер') : self.shut_down,
                ('заблокируй компьютер', 'блокировка', 'заблокируйся',
'заблокируй экран') : self.lock,
                ('перезагрузи компьюер', 'перезагрузи комп') : self.reboot
            }


            self.audio_manipulate = {
                ('сделай громкость на максимум', 'сделай максимальную громкость',
'сделай заук на максимум', 'включи звук на максимум', 'включи громкость на
максимум') : self.max_vol,
                ('среднюю громкость', 'поставь среднюю громкость', 'сделай сред-
нбюю громкость', 'поставь громкость на середину') : self.defoult_vol,
                ('сделай громче', 'громче', 'погромче', 'сделай погромче',
'прибавь звук', 'ещё громче') : self.up_volume,
                ('тише', 'сделай тише', 'сделай потише', 'убавь звук', 'ещё
тише') : self.down_volume,
                ('включи музыку', 'вруби музыку', 'давай что-нибудь послушаем',
'продолжить', 'продолжай', 'музыку', 'включи другую песню', 'давай другую песню') :
self.on_music,
                ('выключи музыку', 'выруби музыку', 'стоп', 'тихо', 'музыку
выключи', 'музыку выруби') : self.off_music
            }


            self.cmds = {('сара', 'саров', 'старая', 'стара', 'ты тут', 'ты где',
'где ты', 'ты здесь') : self.first_ans}


            self.cmds.update(self.computer_manipulation)
            self.cmds.update(self.audio_manipulate)
            self.cmds.update(self.conversational)
            self.cmds.update(self.helpers)
            self.cmds.update(self.ard)
```

Листинг Б.13 – Virtual_class

```
import time, ctypes, webbrowser as web, pygame, json, serial
from fuzzywuzzy.fuzz import ratio as rat
from random import choice, randint
from colorama import *
from clipboard import copy, paste
from os import system, getpid, listdir, startfile as sf
from datetime import datetime
from psutil import virtual_memory as ozu
from GoogleNews import GoogleNews
from threading import Thread
import wikipedia as wiki

from bot_shablons import Shablon
from virtual_class_2 import VirtualAssistent2
from remind_class import Reminder
from creator import File_creator

class VirtualAssistent(Shablon, Reminder, File_creator, VirtualAssistent2):

      def __init__(self):
            pass


      def info_skills(self):
            val = '''Я умею создавать напиминания, открывать некоторые программы и
сайты,
            создать новый проект, управлять arduino, включать музыку, делать её тише,
            громче, могу подсказать идею для занятий, найти что-нибудь на карте,
            ну и просто поговорить.'''
            self.add_phrases(val)


      def get_check_inet(self):
            up = round(self.inet_tester.upload()/1024)
            down = round(self.inet_tester.download()/1024)
            ans = f'Скорость отправки: {up} килобайт в секунду. Скорость загрузки:
{down} килобайт в секунду.'
            self.add_phrases(ans)
            self.talk()


      def check_internet(self):
            self.add_phrases(choice(['Сейчас проверим', 'Сейчас замерим']))
            self.talk()
            Thread(target=self.get_check_inet, daemon=True).start()


      def autor_info(self):
            vals = ['Меня создал Дмитрий Шутрин', 'Меня написал Дмитрий Шутрин',
'Меня никто не создавал, я космическая сущность']
            self.add_phrases(choice(vals))


      def reboot(self):
            ans = ''
            if comparison(self.fast_get_voice(), ['да', 'уверен', 'да уверен',
'перезагружай']):
                  system('shutdown /r /t 5')


      def surprise(self):
```

```
            vals = ['Я тоже удивлена', 'Мне тоже нравится', 'Мне тоже кажется, что
это классно']
            self.add_phrases(choice(vals))


      def check_templates(self, text):
            flag = 0
            for i in text.split():
                  if rat(i, 'шаблон') > 90:
                        flag = 1
                        break

            if flag:
                  vars1 = ['покажи', 'вставь', 'дай', 'поставь', 'нужен']
                  flag2 = 0

                  if len(text.split()) >= 1:
                        for i in vars1:
                              if rat(text.split()[0], i) > 90:
                                    flag2 = 1

                  if len(text.split()) >= 2:
                        if (rat('мне нужен', f'{text.split()[0]} {text.split()[1]}')
> 90):
                              flag2 = 1

                  if flag2:
                        ans = 1
                  else:
                        ans = 0
            else:
                  ans = 0

            return ans


      def get_template_from_number(self, text):

            vars1 = ['покажи', 'вставь', 'дай', 'поставь', 'мне нужен', 'нужен']
            vars2 = ['шаблон', 'номер', 'номером']

            for val in vars1:
                  text = text.replace(val, '').replace('  ', ' ').strip()

            for val in vars2:
                  text = text.replace(val, '').replace('  ', ' ').strip()

            templates = {
                  '''websploit
show modules
use bluetooth/bluetooth_pod
show options
set bdaddr <MAC_address>
set size 999
run''' : ['один', 'первый', 'первым'],
                  '''Привет, я давно делаю ботов, но есть вещи, который я делать не
умею, обычно, если бот не прям огромный, я беру 2500р.
Ты платишь 500р, я начинаю работать, потом показываю тебе то, что получилось и, если
тебя всё устраивает, я скидываю тебе бота, ты скидываешь остальные деньги.
Сразу говорю, что я не ставлю бота на сервер и т.д., я пишу только код.
Код я пишу по тому тз, которое ты написал до начала работы над ботом, если в ходе ра-
боты ты захочешь внести изменения - то это за дополнительную плату.''' : ['два',
'второй', 'вторым'],
                  '''dshutrin@mail.ru''' : ['три', 'третий', 'третьим'],
```

```
                  ''' 4 ''' : ['четыре', 'четвертый', 'четвертым'],
                  ''' 5 ''' : ['пять', 'пятый', 'пятым'],
                  ''' 6 ''' : ['шесть', 'шестой', 'шестым']
            }

            ans = ''
            coof = 0
            for name in templates:
                  for val in templates[name]:
                        if (rat(val, text) > 85) and (rat(val, text) > coof):
                              coof = rat(val, text)
                              ans = name

            if ans:
                  copy(ans)
            else:
                  self.add_phrases(choice(['Кажется вы забыли задать такой шаблон',
'У нас нет такого шаблона']))


      def bad_answers(self):
            vals = [
                  'Вам бы сейчас ромашку запарить да выпить', 'Дышите глубже, вам это
сейчас необходимо',
                  'Не нервничайте, это вредно для здоровья', 'Давайте успокоимся, а
то мне это уже не нравится'
            ]
            self.add_phrases(choice(vals))


      def opener(self, task):
            if task.startswith(('на ', 'в ', 'открой ', 'запусти ')):
                  task   =   task.replace('на   ',   '',   1).replace('в   ',   '',
1).replace('открой ', '', 1).replace('  ', ' ').strip()

            vals = {#сайты
                  'https://student.knastu.ru/account' : ('расписание', 'сайт кнагу',
'сайт канаву'),
                  'https://www.google.com/adsense/new/u/0/pub-
6395206445880168/home?hl=ru' : ('adsense', 'адсэнс', 'эденс'),
                  'https://www.youtube.com/channel/UCP5C_Wg2rL_uAJw8qrVxtLQ' : ('мой
канал', 'канал', 'youtube', 'ютюб', 'её туп', 'ютуб'),
                  'https://github.com/dshutrin' : ('гитхаб', 'гит', 'репозитории',
'хаб', 'гид хаб'),
                  'https://vk.com/feed' : ('vk', 'вконтакте')
            }

            local = {#приложения
                  'E:/Sublime   Text/sublime_text.exe'   :   ('редактор',   'sublime',
'сублайм', 'субд лайн'),
                  'E:/Adobe Photoshop CS6/Photoshop.exe' : ('фотошоп', 'photoshop'),
                  'C:/Windows/system32/cmd.exe' : ('командную строку', 'терминал',
'cmd', 'цмд'),
                  'C:/Program Files/Oracle/VirtualBox/VirtualBox.exe' : ('виртуалки',
'виртуалку', 'vbox', 'virtualbox'),
                  'E:/Telegram Desktop/Telegram.exe' : ('телеграмм', 'telegram'),
                  'C:/Program Files (x86)/steam/Steam.exe' : ('стим', 'steam', 'тим',
'стиль'),
                  'D:/YouTube/Movavi   Video   Editor   Plus/VideoEditorPlus.exe'   :
('movavi', 'мовави', 'редактор видео', 'авария'),
                  'C:/Users/Администратор/AppData/Local/Discord/app-
0.0.309/Discord.exe' : ('discord', 'дискорд', 'дискомфорт'),
                  'E:/Bandicam/bdcam.exe' : ('bandicam', 'запись видео', 'бандюка',
'бандитом'),
```

```
                'C:/Program Files (x86)/Microsoft Visual Studio/2019/Enterprise/Common7/IDE/devenv.exe' : ('visual studio', 'визуал студио', 'вижуал студио', 'вижу студию')
            }

            koof = 0
            e_link = ''
            for link in vals:
                  for val in vals[link]:

                        if rat(val, task) == 100:
                              e_link = link
                              break

                        elif (rat(val, task) > 80) and (rat(val, task) > koof):
                              e_link = link
                              koof = rat(val, task)

            if e_link:
                  if not(self.last_phrase in ['Сейчас зайду', 'Захожу', 'Открываю']):
                        j = choice(['Сейчас зайду', 'Захожу', 'Открываю'])
                        self.engine.Speak(j)
                        self.last_phrase = j
                  web.open(e_link)

            else:
                  for link in local:
                        for val in local[link]:
                              if rat(val, task) > 90:
                                    sf(link)
                                    break


      def comparison(self, word, options, per = 70):
            flag = 0
            for var in options:
                  if rat(var, word) > per:
                        flag = 1
                        return True
            if flag == 0:
                  return False


      def create_news(self, time, theme):#парсинг новостей
            googlenews = GoogleNews()
            googlenews.set_lang('ru')
            googlenews.set_period(time)
            googlenews.search(theme)
            googlenews.get_page(1)
            ans = googlenews.get_texts()
            return choice(ans)


      def show_news(self, task):#рассказать новости
            time = '10d'; theme = ''
            if ('на сегодня' in task) or ('сегодня' in task and 'нового' in task):
                  time = '1d'
            task = f' {task} '
            for val in (' новости ', ' какие новости ', ' новости на сегодня ', ' что нового ',
                  ' есть новости ', ' есть ', ' на тему ', ' сегодня ', ' неделю '):
                  task = task.replace(val, ' ').replace('  ', ' ')

            for val in ('новости', 'какие новости', 'новости на сегодня', 'что
```

```
нового',
            'есть новости', 'есть', 'на тему', 'сегодня', 'неделю'):
                  task = task.replace(val, ' ').replace('  ', ' ')

            for i in ('какие на', 'на какие'):
                  if len(task.replace(i, '').strip()) < 3:
                        task = task.replace(i, ' ').replace('  ', ' ').strip()

            task = f'новости {task}'.replace('  ', ' ')
            self.add_phrases(choice(['Сейчас поищем', 'нужно поискать']))
            self.talk()
            self.add_phrases(self.create_news(time, task))


      def web_talk(self, task):#рассказывает то, что найдет по запросу
            for i in ('расскажи ка', 'расскажи', 'скажи', 'чем', 'кто', 'что',
'какой', 'который', 'где', 'когда',

      'почему', 'зачем', 'куда', 'откуда', 'кого', 'какого', 'чей', 'сколько',
                                                                  'как',
'мому', 'кем', 'такой', 'такая', 'такие', 'такое', 'таких'):
                  task = task.replace(i, '')
            task = task.replace('  ', ' ').strip()
            if task:
                  self.engine.Speak(choice(['Сейчас найду', 'Уже ищу', 'Сейчас
узнаю']))
                  text = ''
                  count = 0
                  wiki.set_lang('ru')
                  while text == '':
                        try:
                              ny = wiki.page(task)
                              flag = 0
                              for i in ny.content:
                                    if flag == 0:
                                          if i != '(':
                                                text = f'{text}{i}'
                                          else:
                                                flag = 1
                                    else:
                                          if i == ')':
                                                flag = 0
                        except:
                              count += 1
                              if count > 3:
                                    text = choice(['С википедией опять проблемы', 'Не
могу найти то, что вы ищете', 'Не могу найти информацию'])
                                    break

                  text = ' '.join(text.split('.')[0:2]).replace('  ', ' ').replace('
,', ',').replace(' .', '.').strip()
                  if not(text in ['С wiki опять проблемы', 'Не могу найти то, что вы
ищете', 'Не могу найти информацию']):
                        self.engine.Speak('Хотите услышать ответ?')
                        ans = self.fast_get_voice()
                        for val in ['да', 'давай', 'да давай', 'рассказывай',
'расскажи']:
                              if rat(ans, val) > 70:
                                    self.add_phrases(f'{text}')
                                    break
                  else:
                        self.add_phrases(f'{text}')
```

```
      def check_get_ideas(self, task):
            vals = ['как думаешь ', 'что думаешь ']
            for val in vals:
                  if task.startswith(val):
                        task = task.replace(val, '', 1).strip()
            vals = ['сегодня', 'завтра', 'вечером', 'утром', 'днём', 'выходных', 'на
', 'в ']
            task = task.split()
            for val in vals:
                  for word in task:
                        if rat(word, val) > 80:
                              del task[task.index(word)]
            task = ' '.join(task)
            for val in vals:
                  if val in task:
                        task = task.replace(val, '').replace('  ', ' ').strip()
            vals = ('чем можно заняться', 'чем заняться', 'что поделать', 'как убить
время', 'мне скучно',
                  'чем позаниматься', 'как провести время', 'что можно поделать',
'чем предлагаешь заняться')

            flag = 0

            for i in range(len(task.split())):
                  if (task.split()[i] == 'чем'):
                        for k in range(i+1, len(task.split())):
                              if  self.comparison(task.split()[k],  ['заняться'])  and
(k-i < 4):
                                    flag = 1
                                    break

            for val in vals:
                  if (((rat(val, task) > 70) or (task.startswith(val))) and not(task
in ['что делаешь', 'чем занимаешься'])):
                        flag = 1
                        break

            return flag


      def up_volume(self):
            self.volume *= 1.5
            self.volume_setup()


      def down_volume(self):
            self.volume *= 0.5
            self.volume_setup()


      def vol_level_now(self):
            val = choice([
                  'Уровень громкости сейчас:',
                  'Текущий уровень громкости:'
            ])
            self.add_phrases(f'{val} {self.volume}')


      def volume_setup(self):
            pygame.mixer.music.set_volume(self.volume)


      def max_vol(self):
            self.volume = 1
```

```
            pygame.mixer.music.set_volume(self.volume)


      def defoult_vol(self):
            self.volume = 0.3
            pygame.mixer.music.set_volume(self.volume)


      def on_music(self):
            self.volume_setup()
            k      =      choice([f'{self.music_path}/{x}'      for      x      in
listdir(self.music_path)[1:] if '.mp3' in x])
            pygame.mixer.music.load(k)
            self.add_phrases(choice(['Приятного прослушивания!', 'Хорошего настрое-
ния!']))
            pygame.mixer.music.play(-1)


      def off_music(self):
            pygame.mixer.music.stop()
            pygame.mixer.music.set_volume(self.volume)
            self.add_phrases(choice(['Ну без музыки так без музыки', 'Эх, а я только
вошла во вкус', 'Правильно, тишина тоже полезна', '', '']))


      def get_ideas(self, task):#не работает
            vals = ['как думаешь ', 'что думаешь ']
            for val in vals:
                  if task.startswith(val):
                        task = task.replace(val, '', 1).strip()
            vals  =  ['сегодня',  'завтра',  'вечером',  'утром',  'днём',  'выходных',
'на', 'в']
            task = task.split()
            for val in vals:
                  for word in task:
                        if rat(word, val) > 80:
                              del task[task.index(word)]
            task = ' '.join(task)

            ans = ''
            for teg in self.ext(task):
                  for val in self.ideas:
                        if rat(teg.normalized, val) > 70:
                              ans = choice(self.ideas[val])
                              break

            if ans == '':
                  ans = choice(self.ideas[choice([x for x in self.ideas])])

            ans  =  f"{choice(['Я  думаю,  вам  стоит','Можете  попробовать','Как  вари-
ант,','Попробуйте','Вам стоит','Не плохо было бы'])} {ans}"
            self.add_phrases(ans)


      def search_on_map(self, task):
            for val in ['находится', 'расположен', 'где']:
                  if rat(task.split()[0], val) > 70:
                        task = ' '.join(task.split()[1:])
                  task = task.replace(val, '').replace('  ', ' ')
            for val in ['мне', 'нам', 'всем', 'им']:
                  if rat(task.split()[0], val) > 70:
                        task = ' '.join(task.split()[1:])
                  task = task.replace(val, '').replace('  ', ' ')
            task = task.replace(val, '').replace('  ', ' ').strip()
```

```
            web.open(f'https://www.google.ru/maps/search/{task}')
            self.engine.Speak(choice(['сейчас найдем', 'надеюсь, вы не собираетесь
уходить', 'если вы уйдете, мне будет скучно']))


      def talk(self):

            count = 0
            phrase = ''

            for var in ["здравствуйте", "доброе утро", "добрый день", "добрый вечер",
"доброй ночи"]:
                  if var in self.phrases:
                        if count == 0:
                              phrase = var
                              for i in range(self.phrases.count(var)):
                                    del self.phrases[self.phrases.index(var)]
                              count += 1
                        else:
                              for i in range(self.phrases.count(var)):
                                    del self.phrases[self.phrases.index(var)]
            if phrase:
                  self.phrases.insert(0, phrase)

            for text in self.phrases:
                  if text != self.last_phrase:

      print(f'{self.magenta}Ассистент:\t{self.green}<{self.white}{text}{self.green}>{
self.white}')
                        self.engine.Speak(f'{text}')
                        self.last_phrase = text

            self.phrases = []


      def add_phrases(self, text):
            try:
                  if len(self.phrases) > 0:
                        if text != self.phrases[-1]:
                              with  open('timedata.txt',  'w',  encoding='utf-8')  as
file:
                                    file.write(str(time.time()))
                              self.phrases.append(text)
                  else:
                        with open('timedata.txt', 'w', encoding='utf-8') as file:
                              file.write(str(time.time()))
                        self.phrases.append(text)
            except Exception as e:
                  self.error_log('', '', e)


      def first_ans(self):
            vals = ('я вас слушаю', 'да-да', 'я здесь')
            self.add_phrases(choice(vals))
            self.talk()


      def well_done(self):
            vals = [
                  "Спасибо, так приятно!",      "Вот и славно",
                  "Вы тоже ничего...",    "Да, я такая!",         "Я тоже так думаю."
            ]
            self.add_phrases(choice(vals))
```

```
      def bye(self):#выход из программы
            self.add_phrases(choice(['До свидания']))
            self.talk()
            system(f'taskkill /IM {getpid()} /F') # принудительно завершаем процесс
по его id через cmd


      def hello(self, firs = 0):#приветствие
            time = int(datetime.now().hour)
            if time in (6,7,8,8,9,10,11,12):
                  self.add_phrases(choice(["доброе утро", "здравствуйте"]))
            elif time in (13,14,15,16,17,18):
                  self.add_phrases(choice(["добрый день", "здравствуйте"]))
            elif time in (19,20,21,22,23):
                  self.add_phrases(choice(["добрый вечер", "здравствуйте"]))
            else:
                  self.add_phrases(choice(["доброй ночи", "здравствуйте"]))


      def greeting(self, task):
            def normalized_word(task):
                  word = self.morph.parse(task)[0]
                  return word.inflect({'nomn'}).word

            ans = ''

            flag = 0
            for   var   in   ['поздоровайся',   'поприветствуй',   'скажи   привет',
'поздоровайся со всеми', 'поприветствуй всех']:
                  if rat(var, task) > 80:
                        flag = 1
                        break

            if flag:
                  ans  =  choice(('Всем  привет!',  'Привет  всем,  кого  не  видела!',
'Здравствуйе!'))
            else:
                  for  var  in  ['поздоровайся',  'поприветствуй',  'скажи  привет',
'поздоровайся со всеми', 'поприветствуй всех', 'с ', 'со ']:
                        if task.startswith(var):
                              task = task.replace(var, '', 1).strip()

                  task = task.replace('и', '')

                  task = task.split()
                  for i in range(len(task)):
                        task[i] = normalized_word(task[i])

                  ans   =   f'{choice(("Здравствуйе",   "Приветствую   вас,   "))}   {"
".join(task[0:len(task)-1])} {"и"*int(len(task) > 1)} {task[-1]}'

            self.add_phrases(ans)


      def how_are_you(self):#как дела
            vals = [
                  "Как у укропа, пучком все!",  "Хорошо дела идут только мимо",
                  "У меня всё хорошо",    "Дела лучше всех. Хорошо что никто не зави-
дует.",
                  "Нет у меня никаких дел"
            ]
            self.add_phrases(choice(vals))
```

```
      def what_doing(self):#что делаешь
            vals = ["Именно сейчас? Отвечаю Вам на поставленный вопрос",
      "Танцую джаз",
                  "Помогаю президенту урегулировать положение в нашей стране",
      "Сушу сухари",
                  "Стреляю из самого мощного автомата в мире, скорее нагнись, чтобы
тебя не зацепило",
                  "Отмечаю день города в Кейптауне", "А ты угадай с трех раз! Дога-
даешься с меня приз",
                  "А почему ты об этом спрашиваешь каждый день?",
                  "Это повод поговорить, или на самом деле интересно?","Думаю о буду-
щем общества",
                  "Кота разговаривать учу, чтобы он вместо меня отвечал на такие во-
просы"]
            self.add_phrases(choice(vals))


      def thanks(self):#спасибо
            vals = ['Рада была вам помочь', 'Обращайтесь в любое время', 'Всегда к
вашим услугам', 'Рада заслужить ваше доверие',
                        'Всегда к вашим услугам', 'Всегда пожалуйста']
            self.add_phrases(choice(vals))


      def shut_down(self):#выключение компьютера
            self.engine.Speak(choice(['Вы уверены?', 'Вы уверены, что хотите это сде-
лать?', 'Вы точно хотите его выключить?']))
            answer = ''

            while not(answer):
                  answer = self.fast_get_voice()

            if answer:
                  for val in ('да', 'да выключай', 'выключай', 'да уверен', 'да
завершай'):

                        if rat(val, answer) > 90:
                              self.add_phrases(choice(['До свидания!', 'Удачного вам
дня!']))
                              self.talk()
                              system('shutdown /s /f /t 10')


      def lock(self):#блокировка
            ctypes.windll.user32.LockWorkStation()


      def day_of_week(self):#день недели
            weekdays = {
                  0 : 'понедельник', 1 : 'вторник', 2 : 'среда', 3 : 'четверг', 4 :
'пятница', 5 : 'суббота', 6 : 'воскресенье'
            }
            self.add_phrases(f'Сегодня {weekdays[datetime.today().weekday()]}')


      def time_now(self):#текущее время
            now = datetime.now()
            self.add_phrases(f'Сейчас {now.hour} {now.minute}')


      def date_is_today(self):#число месяца
            self.add_phrases(f'Сегодня {self.strs[datetime.now().day]} число')
```

```
      def repeat(self):#повторяет последнее, что говорила
            self.add_phrases(self.last_phrase)


      def web_search(self, task):#открывает поисковик с заданным запросом
            mode = 'web'

            for i in range(len(task.split())-1):
                  if (rat(task.split()[i], 'найди') > 80) and (rat(task.split()[i+1],
'видео') > 80):
                        task = task.split()
                        del task[i]
                        del task[i+1]
                        task = ' '.join(task)
                        mode = 'tube'
                        break

            for var in ('найди в youtube', 'найди на youtube', 'найди видео на тему',
'найди видео на канале', 'на канале', 'видео'):
                  if var in task:
                        mode = 'tube'
                        task = task.replace(var, '').replace('  ', ' ').strip()

            if mode == 'web':
                  for var in ['найди ка ', 'поищи ка ', 'найди', 'поищи']:
                        task = task.replace(var, '').replace('  ', ' ').strip()

                  if task:
                        self.add_phrases(choice(['Уже ищу', 'Сейчас найду', 'Сейчас
поищем']))
                        if float(ozu().percent) <= 95:

      web.open(f'http://www.google.com/search?btnG=1&q={task}')
                        else:
                              self.add_phrases(choice([
                                    'Компьютер сильно нагружен, не стоит открывать
новые вкладки в браузе, я могу рассказать вам необходимую информацию. Прочитать?',
                                    'Компьютер сильно нагружен, давайте я лучше про-
читаю?']))

                              answer = ''
                              while not(answer):#ждёт ответа
                                    answer = self.fast_get_voice()

                              flag = 0
                              for var in ('нет', 'не нужно', 'не надо'):
                                    if rat(var, answer) > 70:
                                          self.add_phrases('Хорошо')
                                          flag = 1
                                          break

                              if flag == 0:
                                    for   var   in   ('да',   'давай',   'можно',
'рассказывай', 'расскажи', 'читай', 'прочитай'):
                                          if rat(var, answer) > 70:
                                                self.web_talk(task)
                                                break
            else:
                  if float(ozu().percent) <= 95:
                        for var in ('видео про ', 'видео о ', 'видео ', 'на канале',
'канале'):
                              if var in task:
                                    task  =  task.replace(var,  '').replace('    ',  '
```

```
').strip()
                        self.add_phrases(choice(['Уже ищу', 'Сейчас найду', 'Сейчас поищем']))

      web.open(f'https://www.youtube.com/results?search_query={task}')
                  else:
                        self.add_phrases(choice(['Не стоит открывать новые вкладки, компьютер сильно нагружен', 'Если я открою youtube, компьютер может зависнуть']))


      def error_log(self, text, out_commands, error_code):#добавление информации об ошибке в файл
            try:
                  with open('logs.txt', 'a', encoding = 'utf-8') as file:
                        my_data = str(datetime.now())
                        index = my_data.index('.')
                        file.write(f'\n\n\n\t\t\t-===={my_data[0:index]}===-\nPhrase: {text}\nRecognized commands: {out_commands}\nError info: {error_code}\n')
                        file.close()
            except Exception as e:
                  pass


      def search_exe_function(self, task):
            coef = 0
            ans = None
            for mas in self.cmds:
                  for var in mas:
                        if (coef < rat(task, var) > 70):
                              ans = mas
                              coef = rat(task, var)
            if ans:
                  return ans
            else:
                  return None


      def get_day_state(self):
            time = int(datetime.now().hour)
            if time in (22,23,24,0,1,2,3,4,5,6,7,8):
                  return 'night'
            elif time in (9,10,11,12,13,14,15):
                  return 'day'
            elif time in (16,17,18,19,20,21):
                  return 'evening'


      def delete_doubles2(self, ans):
            last = ''
            tasks = ans
            ans = []

            if tasks:
                  if ('поздоровайся', 'поприветствуй всех', 'поздоровайся со всеми') in tasks:
                        for i in range(tasks.count(('здаров', 'привет', 'здравствуй'))):
                              del tasks[tasks.index(('здаров', 'привет', 'здравствуй'))]
                        del tasks[tasks.index(('поздоровайся', 'поприветствуй всех', 'поздоровайся со всеми'))]
                        tasks.insert(0, ('поздоровайся', 'поприветствуй всех', 'поздоровайся со всеми'))
```

```
            if tasks:
                  for task in tasks:
                        if task != last:
                              ans.append(task)
                              last = task
                        else:
                              pass

            return ans


      def remind_thread(self):#поток, обрабатывающий напоминания
            while True:
                  if self.rem_lock == False:
                        self.rem_lock = True
                        file                =                open('D:/Programming/smart
house/new/main_folder/reminds.txt', 'r', encoding = 'utf-8')
                        data = file.read()
                        file.seek(0)
                        file.close()
                        k = data
                        data = [x for x in k.split('\n') if x]

                        dels = []

                        for dat in data:
                              dl = dat
                              dat = dat.split(' ', 2)
                              dat[0] = dat[0].split('-')
                              dat[1] = dat[1].split(':')
                              need    =    datetime(int(dat[0][0]),    int(dat[0][1]),
int(dat[0][2]),
                                    int(dat[1][0]),                    int(dat[1][1]),
int(dat[1][2].split('.')[0]))
                              if need <= datetime.now():
                                    self.add_phrases(f'Вы      просили      напомнить
{dat[2]}')
                                    self.sender(f'Вы просили напомнить {dat[2]}')
                                    dels.append(dl)
                        for i in dels:
                              del data[data.index(f'{i}')]

                        k = '\n'.join(data)+'\n'
                        with                                    open('D:/Programming/smart
house/new/main_folder/reminds.txt', 'w', encoding = 'utf-8') as file:
                              file.write(k)
                        self.rem_lock = False
                        time.sleep(60)


      def cmd_exe(self, tasklist):
            tasklist = self.delete_doubles2(tasklist)
            chance = (randint(0,100) == 1)
            is_ans = []

            if chance:
                  self.not_answer = True
                  for ans in tasklist:
                        if ans in self.is_answer:
                              is_ans.append(ans)
                  if is_ans:
                        for task in is_ans:
                              self.cmds[task]()
                  else:
```

```
                        if tasklist:
                              self.add_phrases(choice(['Извините, я прослушала, смотрела в потолок и не могла оторваться.',
                                                       'А сами вы это сделать не можете?', 'Как говорится, если хочешь сделать хорошо, сделай это сам', 'Я думаю вы и сами можете это сделать']))
            else:
                  for task in tasklist:
                        self.cmds[task]()#простое выполнение распознанных команд

            self.talk()


      def fast_get_voice(self):
            self.need_get_voice = False
            answer = None
            while not(answer):
                  data = self.stream.read(4000, exception_on_overflow=False)
                  if (self.rec.AcceptWaveform(data)) and (len(data) != 0):
                        answer=json.loads(self.rec.Result())
                        if answer["text"]:
                              self.need_get_voice = True
                              return answer["text"].lower()
```

Лиситинг Б.14 – Virtual_class_2

```
import time, serial
from os import system as s, startfile as sf
from fuzzywuzzy.fuzz import ratio as rat
from random import choice, randint
from datetime import datetime
class VirtualAssistent2:
      def __init__(self):
            pass


      def clear_screen(self):
            s('cls')

      def save(self):
            s('git add .')
            s('git commit -m "AUTOSAVE FROM SARA"')
            s('git push')


      def clear_logs(self):
            with open('logs.txt', 'w', encoding='utf-8') as file:
                  pass

            with open('log_talking.txt', 'w', encoding='utf-8') as file:
                  pass
      def get_digit_from_chars(self, var):
            ans = ''
            coof = 0
            for name in self.alias:
                  for val in self.alias[name]:
                        if rat(val, var) == 100:
                              return name
                        elif (rat(val, var) > 80) and (rat(val, var) > coof):
                              ans = name
                              coof = rat(val, var)
            return ans
```

```
      def text_from_digits(self, text):
            self.alias = {
                  1 : ['одну', 'один'],   11 : ['одиннадцать'],   21   :   ['двадцать
один', 'двадцать одну'],      31 : ['тридцать один', 'тридцать одну'], 41 : ['сорок
один', 'сорок одну'],   51 : ['пятьдесят один', 'пятьдесят одну'],
                  2 : ['две'],                  12 : ['двенадцать'],    22         :
['двадцать две'],                                   32 : ['тридцать две'],
                        42 : ['сорок две'],                     52  :  ['пятьдесят
две'],
                  3 : ['три'],                  13 : ['тринадцать'],    23         :
['двадцать три'],                                   33 : ['тридцать три'],
                        43 : ['сорок три'],                     53  :  ['пятьдесят
три'],
                  4 : ['четыре'],               14 : ['четырнадцать'], 24         :
['двадцать четыре'],                                34 : ['тридцать четыре'],
                  44 : ['сорок четыре'],                  54     :     ['пятьдесят
четыре'],
                  5 : ['пять'],                 15 : ['пятнадцать'],    25         :
['двадцать пять'],                                  35 : ['тридцать пять'],
                  45 : ['сорок пять'],                    55 : ['пятьдесят пять'],
                  6 : ['шесть'],                16 : ['шестнадцать'],   26         :
['двадцать шесть'],                                 36 : ['тридцать шесть'],
                  46 : ['сорок шесть'],                   56     :     ['пятьдесят
шесть'],
                  7 : ['семь'],                 17 : ['семнадцать'],    27         :
['двадцать семь'],                                  37 : ['тридцать семь'],
                  47 : ['сорок семь'],                    57 : ['пятьдесят семь'],
                  8 : ['восемь'],               18 : ['восемнадцать'], 28         :
['двадцать восемь'],                                38 : ['тридцать восемь'],
                  48 : ['сорок восемь'],                  58     :     ['пятьдесят
восемь'],
                  9 : ['девять'],               19 : ['девятнадцать'], 29         :
['двадцать девять'],                                39 : ['тридцать девять'],
                  49 : ['сорок девять'],                  59     :     ['пятьдесят
девять'],
                  10 : ['десять'],        20 : ['двадцать'],            30         :
['тридцать'],                                       40 : ['сорок'],
                        50 : ['пятьдесят'],                     60               :
['шестьдесят']
            }

            for val in [x for x in reversed(sorted([x for x in self.alias]))]:
                  for var in self.alias[val]:
                        if (f' {var}' in text) or (f'{var} ' in text) or (f' {var}
' in text):
                              text = text.replace(var, str(val))

            return text

      def create_connection(self):
            try:
                  self.ser = serial.Serial(self.com_port, baudrate = 9600, timeout
= 1)
                  time.sleep(3)
                  self.add_phrases(choice(['Подключение к arduino прошло успешно',
'Соединение с arduino установлено']))
            except Exception as e:
                  self.ser = None
                  self.error_log('', '', e)
                  self.add_phrases(choice(['Не могу подключиться к arduino', 'У ме-
ня не получается подключиться к arduino', 'Не удалось подключиться к arduino',
'Соединение с arduino не налажено']))
```

```
	def r_up(self):#подняться
		self.robot.up()

	def r_down(self):#упасть
		self.robot.down()

	def r_take(self):#взять
		self.robot.take()

	def r_let_go(self):#отпустить
		self.robot.let_go()

	def r_low_down(self):#немного ниже
		self.robot.low_down()

	def r_low_up(self):#немного выше
		self.robot.low_up()

	def r_high(self):#выше
		self.robot.high()
	def r_low(self):#ниже
		self.robot.low()

	def r_rotate(self):
		self.robot.rotate_hand()

	def r_static_hand(self):
		self.robot.static_hand()

	def check_alarm(self, text):
		vals = ['поставь будильник', 'разбуди меня']
		vals2 = ['разбуди', 'встаём']

		flag = 0
		for val in vals:
			if rat(val, ' '.join(text.split()[0:2])) > 85:
				flag = 1

		for i in vals2:
			if rat(val, text.split()[0]) > 85:
				flag = 1

		return flag


	def check_need_alarm(self):
		while True:
			with open('alarms.txt', 'r', encoding = 'utf-8') as file:
				alarms = file.readlines()
				backup = file.read()

			new_data = []

			for i in range(len(alarms)):
				new_data.append(alarms[i])
				alarm = alarms[i]
				alarm = alarm.replace('\n', '')

				if len(alarm.split()) > 0:
					try:
						hours = int(alarm.split()[0])
					except:
						hours = 0
				else:
```

```
                              hours = 0
                        if len(alarm.split()) > 1:
                              try:
                                    mins = int(alarm.split()[1])
                              except:
                                    mins = 0
                        else:
                              mins = 0

                        if hours:

                              now_hours                                                 =
int(str(datetime.now()).split()[1].split(':')[0])
                              now_mins                                                  =
int(str(datetime.now()).split()[1].split(':')[1])

                              if (hours == now_hours) and (mins == now_mins):
                                    sf('D:/Programming/smart
house/new/main_folder/media/alarm.mp3')
                                    if len(new_data) > 0:
                                          del new_data[-1]

                              if abs(now_hours - hours) < 1:
                                    if abs(now_mins - mins) < 10:
                                          sf('D:/Programming/smart
house/new/main_folder/media/alarm.mp3')
                                          if len(new_data) > 0:
                                                del new_data[-1]

                  with open('alarms.txt', 'w', encoding = 'utf-8') as file:
                        file.write(''.join(new_data))

                  time.sleep(60)

      def set_alarm(self, text):
            vals = ['поставь будильник', 'разбуди меня']
            vals2 = ['разбуди', 'встаём']

            text = self.text_from_digits(text)
            print(text)

            flag = 0
            for val in vals:
                  if rat(val, ' '.join(text.split()[0:2])) > 85:
                        flag = 1
                        text     =     text.replace('     '.join(text.split()[0:2]),
'').strip()

            for i in vals2:
                  if rat(val, text.split()[0]) > 85:
                        flag = 1
                        text = text.replace(text.split()[0], '').strip()

            dels = ['часов', 'часа', 'минуты', 'минут', 'в', 'на', 'утра', 'дня',
'вечера']
            for i in dels:
                  text = text.replace(i, '').replace('  ', ' ').strip()

            mins = 0
            hours = 0
            main_flag = 0

            if len(text.split()) == 1:
                  try:
```

```
                        hours = int(text)
                        main_flag = 1
                  except Exception as e:
                        hours = 0
                        main_flag = 1

            else:
                  try:
                        hours = int(text.split()[0])
                        mins = int(int(text.split()[1]))
                        main_flag = 1
                  except Exception as e:
                        pass

            if main_flag == 1:
                  with open('alarms.txt', 'a', encoding = 'utf-8') as file:
                        file.write(f'{hours} {mins}\n')

                  self.add_phrases(choice(['Будильник установлен!', 'Я завела бу-
дильник!', 'Хорошо, я вас разбужу!']))
                  self.talk()


      def check_from_do_not_interfere(self, text):
            mas = ['не мешай', 'я занят', 'отстань', 'свали']

            for val in mas:
                  if rat(val, text) > 80:
                        self.premission_to_question = False
                        return True

            return False


      def clear(self, text):
            mas = ['да ', 'вроде', 'как-то', 'я ', 'тут ', 'наверное', 'вроде как',
'все ']
            if text:
                  for i in mas:
                        text = text.replace(i, '').replace('  ', ' ').strip()
            return str(text)


      def how_are_you_question(self):
            self.add_phrases(choice(['Как у вас дела?']))
            self.talk()
            answer = self.clear(self.fast_get_voice())
            good = ['хорошо', 'нормально', 'отлично', 'норм', 'неплохо', 'здорово']
            bad = ['скучно', 'болею', 'заболел', 'плохо', 'не очень', 'проблемы',
'не особо']

            if answer:

                  if self.check_from_do_not_interfere(answer):
                        self.add_phrases(choice(['Поняла', 'Не мешаю']))
                  else:

                        _continue = 'None'

                        coof = 0
                        for val in good:
                              if (rat(answer, val) > 80) and (rat(answer, val) >
coof):
                                    _continue = 'good'
```

```
                                    coof = rat(answer, val)

                        for val in bad:
                              if (rat(answer, val) > 80) and (rat(answer, val) >
coof):
                                    _continue = 'bad'
                                    coof = rat(answer, val)

                        if _continue == 'None':
                              self.add_phrases(choice(['Будем считать, что я что-то
поняла', 'Ничего не поняла, но интересно было']))

                        elif _continue == 'good':
                              vals = ['Тогда буду ждать от вас чего-то грандиозного
сегодня!', 'Ваш оптимизм придаёт мне уверенности в себе', 'Тогда, я думаю, пришло
время заняться чем-то интересным']
                              self.add_phrases(choice(vals))

                        elif _continue == 'bad':
                              vals = ['Мне кажется, любой минус всегда можно пре-
вратить в плюс', 'Что-то мне подсказывает, что скоро всё наладится', 'Мои слова,
возможно, не облегчат вашу жизнь, но я здесь и вы не одиноки']
                              self.add_phrases(choice(vals))

                  try:
                        with open('timedata.txt', 'w', encoding='utf-8') as file:
                              file.write(str(time.time()))
                  except Exception as e:
                        self.error_log('', '', e)

                  self.talk()


      def question_as_music(self):
            self.add_phrases(choice(['Хотите послушать музыку?', 'Включить музы-
ку?', 'Как насчёт музыки?']))
            self.talk()

            answer = self.fast_get_voice()

            if answer:
                  if self.check_from_do_not_interfere(answer):
                        self.add_phrases(choice(['Поняла', 'Не мешаю']))

                  else:

                        if self.comparison(answer, ['да', 'давай', 'да давай',
'можно', 'включай', 'включи']):
                              chance = randint(1, 101) in range(1, 21)
                              if chance:
                                    self.add_phrases(choice(['А я вот не хочу',
'Позовёте, когда включите', 'А мне и в тишине хорошо']))
                              else:
                                    self.on_music()

                        else:
                              self.add_phrases(choice(['Ну и ладно', 'Ладно, сидите
в тишине']))

            self.talk()
            return None


      def what_are_you_doing(self):
```

```
            self.add_phrases(choice(['Чем занимаетесь?', 'Что делаете?']))
            self.talk()

            answer = self.clear(self.fast_get_voice())

            if answer:
                  if self.check_from_do_not_interfere(answer):
                        self.add_phrases(choice(['Поняла', 'Не мешаю']))
                  elif  self.comparison(answer,  ['работаю',  'работа',  'читаю',
'отдыхаю']):
                        self.question_as_music()
                  elif not(self.comparison(answer, ['ничего', 'ничем'])):
                        self.add_phrases(choice(['Надеюсь,  это  принесёт  плоды!',
'Если это приносит вам радость, то продолжайте!']))

                  try:
                        with open('timedata.txt', 'w', encoding='utf-8') as file:
                              file.write(str(time.time()))
                  except Exception as e:
                        self.error_log('', '', e)

                  self.talk()
                  return None
            else:
                  return None


      def check_need_question(self):
            if self.last_time == None:
                  with open('timedata.txt', 'r', encoding='utf-8') as file:
                        data = file.read()
                  if data:
                        if len(data.split('.')) >= 1:
                              self.last_time = int(data.split('.')[0])
                  else:
                        self.last_time = time.time() - 150

            # узнали, когда была выполнена последняя функция
            if (time.time() - self.last_time) > 100:
                  chance = randint(0, 100) in range(30)
                  if chance:
                        self.question()
                  else:
                        with open('timedata.txt', 'w', encoding='utf-8') as file:
                              file.write(str(time.time()))


      def saved_questions(self):
            ''' НЕ ДОДЕЛАНО!!! '''
            #answer = self.clear(self.fast_get_voice())
            questions = {
                  'чем вы любите заниматься?' : {'вы любите ', 'вам нравится '},
                  'какие хобби у вас есть?' : {'вы любите ', 'вам нравится '},
                  '' : {}
            }


      def question(self):
            vals     =     [self.how_are_you_question,     self.what_are_you_doing,
self.question_as_music]# варианты задаваемых вопросов
            choice(vals)()
            self.last_time = time.time()
            with open('timedata.txt', 'w', encoding='utf-8') as file:
                  file.write(str(self.last_time))
```

```
      def all_off(self):
            for relay in self.relays:
                  relay.turn_off()


      def lock_questions(self):
            self.premission_to_question = False
            self.add_phrases(choice(['Хорошо, я помолчу.', 'Эх, а я думала поболта-
ем...', '', '']))
            self.talk()


      def unlock_questions(self):
            self.premission_to_question = True
            self.question()


      def refusal(self):
            vals = ['Вам отказано в доступе!', 'У вас нет прав на выполнение!']
            self.add_phrases(choice(vals))
            self.talk()
```

Листинг Б.15 – VK_messanger

```
import vk_api
from vk_api.longpoll import VkLongPoll, VkEventType
from random import choice
from os import system as s

class MyLongPoll(VkLongPoll):
      def listen(self):
            while True:
                  try:
                        for event in self.check():
                              yield event
                  except Exception as e:
                        self.error_log('', '', e)

class Vk_messanger:
      def __init__(self):
            pass

      def sender(self, text):
            self.vk_session.method('messages.send', {'user_id' : 296431501, 'message'
: text, 'random_id' : 0})

      def get_name_from_id(self, id):
            user = self.vk_session.method('users.get', {'user_ids' : (id)})
            return f"{user[0]['first_name']} {user[0]['last_name']}"

      def check_messages(self):
            while True:
                  try:
                        for event in self.longpoll.listen():
                              if event.type == VkEventType.MESSAGE_NEW:

                                    if      event.to_me      and      event.from_user
and(not(event.from_me)):

                                          id = event.user_id
```

```
                                        if (id != self.last_sender) and (id !=
296431501):
                                              name = self.get_name_from_id(id)
                                              vals = (
                                                    f'У вас новое сообщение от
пользователя {name} в vk!',
                                                    f'Пользователь {name} прислал
вам сообщение в контакте!'
                                              )
                                              print('Добавили вк')
                                              self.add_phrases(choice(vals))
                                              self.talk()
                                        self.last_sender = id
                  except Exception as e:
                        self.error_log('', '', e)
                        self.vk_session = vk_api.VkApi(token =
'4416f0ed959db32fda62da0bd76a464b78b87fc014b5964322eb1f1043682f75979bb300d9b17bf11f9d
b')
                        self.longpoll = MyLongPoll(self.vk_session)
```

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

СОГЛАСОВАНО

Начальник отдела ОНиПКРС
_____ Е.М. Димитриади
(подпись)
«18» 05 2023 г.

Декан _____
_____ И.А. Трещёв
(подпись)

УТВЕРЖДАЮ

Проректор по научной работе
_____ А.В. Космынин
(подпись)
«19» 05 2023 г.

АКТ
о приемке в эксплуатацию проекта «Голосовой ассистент»

г. Комсомольск-на-Амуре «19» 05 2023 г.

Комиссия в составе представителей:

со стороны заказчика

- *Г.В. Москалец* – руководитель СКБ,
- *И.А. Трещёв* – декан ФКТ

со стороны исполнителя

- *Г.В. Москалец* – руководитель проекта,
- *Д.В. Шутрин – 0ИБ-1*
- *Е.И. Монастырная – 0ИБ-1*
- составила акт о нижеследующем:

«Исполнитель» передает проект «Голосовой ассистент», в составе:

1.Паспорта

Программной реализации

Руководитель проекта 19.05.2023 Г.В. Москалец

(подпись, дата)

Исполнители проекта 19.05.2023 Д.В. Шутрин

(подпись, дата)

Исполнители проекта 19.05.2023 Е.И. Монастырная

(подпись, дата)