Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

(ФГБОУ ВО «АмГУ»)

Факультет энергетический

Кафедра автоматизации производственных процессов и электротехники

Направление подготовки 15.03.04 - Автоматизация технологических процессов и производств

Направленность (профиль) образовательной программы: Автоматизация технологических процессов и производств в энергетике

ДОПУСТИТЬ К ЗАЩИТЕ
И.о. заведующего кафедрой
_____А.А. Остапенко
«____»_____2017г.

БАКАЛАВРСКАЯ РАБОТА

на тему: Система диспетчеризации на основе GSM-технологий (комплексная выпускная квалификационная работа)

Исполнитель
студент группы 341об _____ (подпись, дата) ———И.А. Коняев——— (И.О.Ф)

Руководитель
доцент, канд. техн. наук. (должность, ученое звание) _____ (подпись, дата) ———А.Н. Рыбалев——— (И.О.Ф.)

Консультант
по безопасности и
экологичности
доцент, канд. техн. наук. (должность, ученое звание) _____ (подпись, дата) ———А.Б. Булгаков——— (И.О.Ф.)

Нормоконтроль
профессор, д-р техн. наук. (должность, ученое звание) _____ (подпись, дата) ———О.В. Скрипко——— (И.О.Ф.)

Благовещенск 2017

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет энергетический
Кафедра автоматизации производственных процессов и электротехники

УТВЕРЖДАЮ
И.о. заведующего кафедрой
_____А.А. Остапенко
«____»_____2017г.

ЗАДАНИЕ

К выпускной квалификационной работе студента *Коняева Игоря Александровича*
1. Тема выпускной квалификационной работы*Система диспетчеризации на основе GSM-технологий)*
(утверждена приказом от 07.12.2016 № 2673-уч)
2. Срок сдачи студентом законченной выпускной квалификационной работы: _____
3. Исходные данные к выпускной квалификационной работе: *данные преддипломной практики, техническая документация по RaspberryPi*
4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов): *Концептуальное проектирование устройства, разработка аппаратной части проектируемой системы, разработка программного обеспечения*
5. Перечень материалов приложения: (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.) Техническое задание на разработку, детальная электрическая схема со спецификацией, структура главного программного модуля, общая структура программного обеспечения, структура базы данных, структура системы и описание платформы, основные команды запросов к серверу, листинг кода ПО, интерфейс пользователя на ПК и на смартфоне
6. Консультанты по выпускной квалификационной работе (с указанием относящихся к ним разделов) *доценткафедры БЖД, канд. техн наук А.Б. Булгаков, Безопасность и экологичность*
7. Дата выдачи задания: 05.12.2016
Руководитель выпускной квалификационной работы: *Рыбалев Андрей Николаевич*
доцент кафедры АППиЭ, кандидат технических наук
(фамилия, имя, отчество, должность, ученая степень, ученое звание)

Задание принял к исполнению (дата): 05.12.2016

(подпись студента)

РЕФЕРАТ

Бакалаврская работа содержит 182с., 84 рисунка, 6 таблиц, 22источников, 9приложений.

ЛОГИЧЕСКИЙ КОНТРОЛЛЕР, ПОРТЫ ВВОДА И ВЫВОДА, ПРОГРАММЫ, СЕРВЕР, LINUX, TELEGRAM, ПРИЛОЖЕНИЕ, PYTHON, СУБД, СМАРТФОН, СЕНСОР ТЕМПЕРАТУРЫ, АТМОСФЕРНОЕ ДАВЛЕНИЕ, АППАРАТНО-СТРУКТУРНАЯ СХЕМА

Концептуальное проектирование устройства. Разработка аппаратной части проектируемой системы. Разработка программной части проектируемой системы.

СОДЕРЖАНИЕ

Введение 5
1 Концептуальное проектирование устройства 8
1.1 Объект управления 8
1.2 Постановка и описание задачи разработки 9
1.3 Разработка исходной структурной схемы устройства 9
1.4 Направление решения поставленной задачи и выбор основных элементов 10
2Разработка аппаратной части проектируемой системы 15
2.1 Детальная информация по аппаратной реализации системы 15
2.2 Разработка полной электрической схемы устройства 19
3 Разработка ПО 22
3.1 Установка и настройка программного обеспечения 22
3.2 Разработка общей архитектуры 29
3.3 Разработка модуля безопасности 49
4 Безопасность и экологичность 59
4.1 Безопасность и эргономика рабочего места 59
4.3 Чрезвычайные ситуации 64
4.4 Информационная безопасность 70
Заключение 72
Библиографический список 73
Приложение А – Техническое задание на разработку 75
Приложение Б – Детальная электрическая схема со спецификацией 84
Приложение В – Структура главного программного модуля 86
Приложение Г – Общая структура программного обеспечения 87
Приложение Д – Структура базы данных 88
Приложение Е – Структура системы и описание платформы 89
Приложение Ж – Основные команды запросов к серверу 90
Приложение И – Листинг кода ПО 91
Приложение К – Интерфейс пользователя на ПК и на смартфоне 182

ВВЕДЕНИЕ

Мобильный интернет – технология беспроводног подключения к сети интернет практически из любой точк земного шара. Сама технология мобильного интернет является постоянно развивающимся явлением в отрасли информационны и телекоммуникационных технологий. История прихода мобильной связ в массы началась в 1980-х, когда почти вся голосова связь была аналоговой, скорость передачи чере модемы была мала, а стоимость владения мобильны телефоном высока. Это было первое поколение, именуемо 1G.

Поколение 2G в начале 90-х принесло много новог . На смену аналоговым системам пришли цифровые, повысилас защищенность, производительность, качество передаваемого звука. В Европ поколение имело стандарт связи GSM, в США – CDMA. Так же появилась возможность передавать коротки SMS-сообщение и данные посредствомCSD.

С поколением 2.5G появился сервисGPRS (GeneralRadioService), который обеспечил технологию непрерывно передачи данных с большей скоростью, чем у CSD и без необходимости дозвона. В теории скорость должн была составлять до 100 кбит/с, что позволило еще и тарифицировать траффик Идейным продолжением была технология UMTS, которая относилась уже к поколению 3G. Но выжимкой из GPRS стала (неофициально названа 2.75G).

Дальнейшие усовершенствования мобильных сете (3G-3.75G) касались быстродействия, пропускной способности и стабильност сети. На смену с пропускной способностью в 21 Мбит/с пришла HSDPA. Постепенно развитие от HSDPA до DC-HSDPA-w/MIMO дало общи прирост в скорости на 60 Мбит/с. Технология LTE, пришедшая с поколением 4G, уже обеспечивает прием данных:326,4 Мбит/с; передач : 172,8 Мбит/с. Задержку уменьшили до 5 мс.

Telegram –бесплатный кроссплатформенный мессенджер для смартфоно и других устройств, позволяющий обмениваться текстовыми

сообщениями и медиафайлами различных формато . Используются проприетарная серверная часть с закрытым кодом работающая на мощностях нескольких компаний США и Германии, финансируемы Павлом Дуровым в объёме порядка 12 млн долларов США ежегодн , и несколько клиентов с открытым исходным кодом, в том числ под лицензией GNU GPL.

Количество активных пользователей сервис на февраль 2016 года составляло боле 100 млн человек, а количество ежедневно пересылаемых сообщений достигл 10 миллиардов на август 2015.

Для мессенджера был создан протоко MTProto, предполагающий использование нескольких протоколов шифрования При авторизации и аутентификации используются алгоритмы RSA-2048 DH-2048 для шифрования, при передаче сообщений протокола в сет они шифруются AES с ключом, известным клиенту и серверу. Такж применяются криптографические хеш-алгоритмы SHA- и MD5.

Безопасность от перехвата пересылаемых сообщений со сторон сервера Telegram обеспечивается лишь в режиме «секретных чатов (Secret Chats), доступном с 8 октября 2013 года. Этот режи реализует шифрование, при котором лишь отправитель и получател обладают общим ключом (end-to-end шифрование), с применение алгоритма AES-256 в режиме IGE (англ. Infinite Garbl Extension) для пересылаемых сообщений. В отличие от обычного режима сообщения в секретных чатах не расшифровываются сервером, истори переписки сохраняется лишь на тех двух устройствах, на которы был создан чат.

При обмене файлами можно как отправит файлы с устройства, так и искать медиаконтент в интернете, в том случае если используется мобильная версия для iOS или Android. Разме передаваемых файлов ограничен 1,5 Гб. Программа использу систему докачки файлов после обрыва связ .

Имеется возможность организовывать мультичаты до 200 участников, начина с ноября 2015 года, супергруппы до 1000 участнико , с 14 марта 2016 – супергруппы до 5000 участников.

При помощи специального API сторонние разработчик могут создавать «ботов», специальные аккаунты, управляемы программами. Типичные боты отвечают на специальные команд в персональных и групповых чатах, также они могут осуществлят поиск в интернете или выполнять иные задачи, применяютс в развлекательных целях или в бизнесе.

В сентябре 2015 года Павел Дуров заяви о скором появлении возможностей монетизации и размещения реклам в ботах.

Raspberry Pi –одноплатный компьютер размером с банковску карту, изначально разработанный как бюджетная система для обучени информатике, впоследствии получивший намного более широко применение и популярность, чем ожидали его авторы. Разрабатывается Raspberr Pi Foundation. Всего за три года было продано боле 4,5 миллионов устройств Raspberry Pi.

Компьютер распространяется полность собранным на четырёхслойной печатной плате размером с банковску карту. В стандартный комплект поставки входит тольк сама плата. Корпус, блок питания, флеш-карт необходимо заказывать отдельно.

1 КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ УСТРОЙСТВА

1.1 Объект управления

Объектом управления является лабораторный стенд, совмещающий в себе ручное и автоматическое управление лампами освещения как удаленно, так и на месте. Внешний вид стенда показан на рисунке 1.

Рисунок 1 – Лабораторный стенд

Удаленное управление осуществляется посредством передачи данных между двумя модемами ПМ01 по технологии CSD. Задействована SCADA-система TraceMode, через которую осуществляется управление нагрузкой и получение параметров тока с измерителя электрической сетиOMIX. Управление системой возложено на программируемый логический контроллер Овен ПЛК100.

В последующих пунктах будет представлены структурная и аппаратная схемы стенда до модификации.

1.2Постановка и описание задачи разработки

Необходимо разработать систему беспроводного мониторинга удаленного объектасредствами мобильного интернета. Вид объекта не имеет значения, это может быть объект малого (крупного) предприятия или объект частного характера. В предыдущем пункте был представлен объект управления, головную часть которого нужно заменить.

Главным требованием к системе является удаленный доступ из любого места, где есть доступ к мобильному или к проводному интернету.

Подсловом«мониторинг» нужно понимать следующее:

− Управление исполнительными механизмами через приложение на смартфоне или ПК;

−Сбор показателей с датчиков температуры, атм. давления,влажности воздуха и формирование статистики в виде графиков (сутки, неделя);

−Сбор токовых параметров с измерителя электрических параметров;

−Оповещение об ошибках в ПО (так же возможны оповещения по аппаратным неисправностям).

Режимы работы, при которых не требуется участие человека, согласовывается на этапе разработки программного решения.

1.3Разработка исходной структурной схемы устройства

Управляющее устройство совмещает в себе контроллер и сервер. Позже будет дано объяснение причин выбора такого решения, плюс альтернативы.

На рисунке 2 представлена структурная схема, которая показывает все элементы, подключаемые к серверу в дальнейшем.

−Питание – источник питания, осуществляющий питание элементов устройства и сервера;

−Измеритель токовых параметров «OMIX»

−Датчик влажности воздуха «HS»;

−Датчик атмосферного давления «PS»;

–Датчик температуры «TS»;

–4GUSB-модем или WI-FI-роутер «4G-модем»;

–Лампа освещения №1 «HL1»;

–Лампа освещения №2«HL2»;

–Лампа освещения №3 «HL3»;

–Пользователь, управляющий системой «Клиент»;

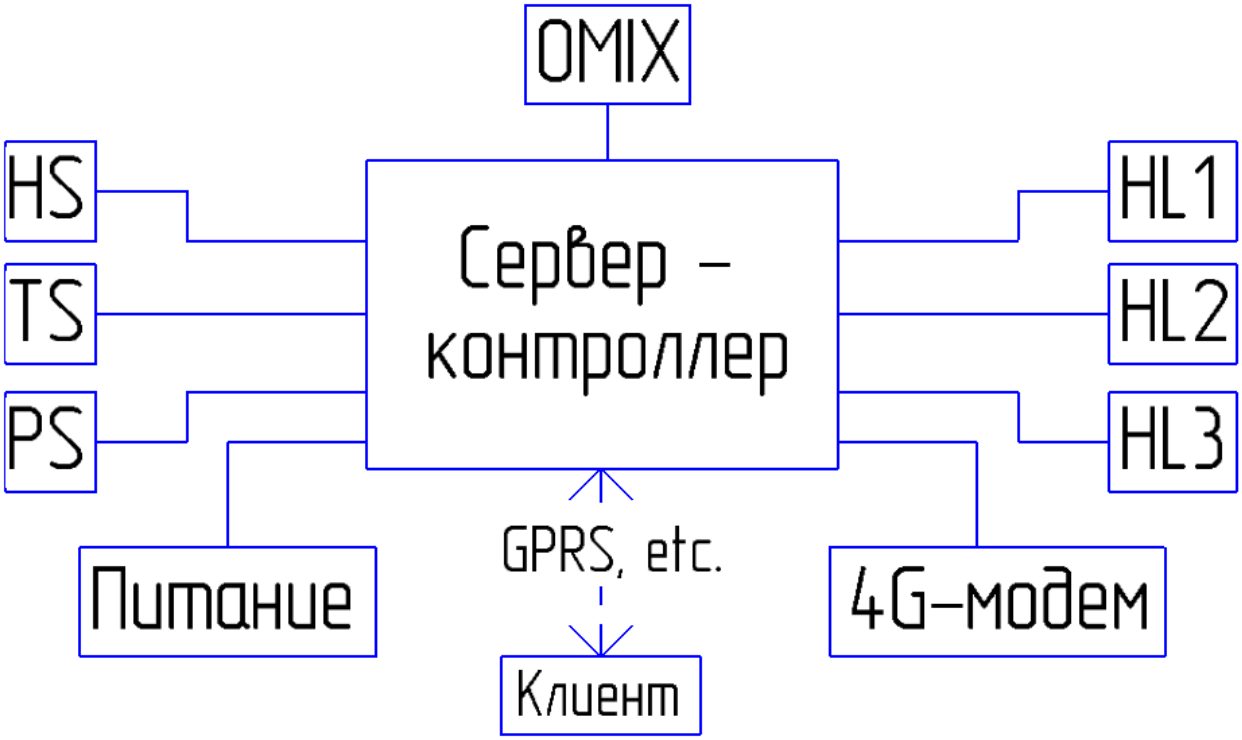


Рисунок 2 – Структурная схема устройства

1.4 Направление решения поставленной задачи и выбор основных элементов

В этом пункте будет произведен выбор основного оборудования для системы на уровне марок и моделей.

Датчик атмосферного давления выбирался сразу цифровой, работающий по цифровому интерфейсу I2C (расшифровка в следующих пунктах). Такие датчики могут измерять атмосферное давление для вычисления высоты над уровнем моря или для прогнозирования погоды. Так же у них есть сенсор измерения температуры. Моделей на рынке множество, при выборе

нужно смотреть на погрешность и диапазоны измеряемых величин. Из всего разнообразия был выбран BMP280 фирмы Bosch. Условно говоря, этот датчик будет измерять температуру снаружи и давление, эти данные потом можно использовать в разных целях. Внешний вид показан на рисунке 3.

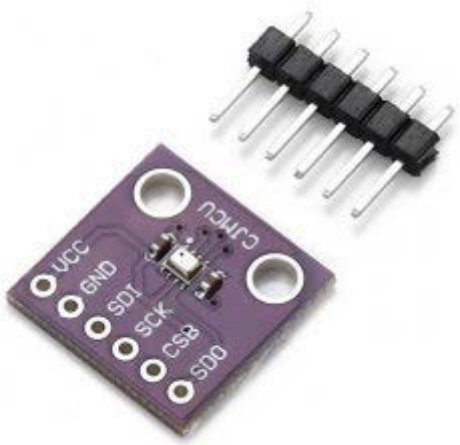


Рисунок 3 – Датчик давления и температурыBMP280

Датчик влажности – гигрометр был выбран тоже электронный. Среди подобных моделей по цене и функциональности был выбран DHT11. Помимо измеряемой влажности воздуха, он может измерять температуру. Диапазон измерения температур как раз подходит для использования внутри помещения. Внешний вид показан на рисунке 4.

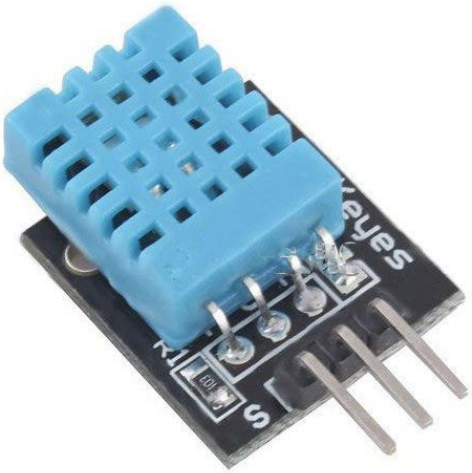


Рисунок 4 – Датчик влажности и температуры DHT11

Управлять токовой нагрузкой будем через токовые реле. Для трех ламп освещения подойдет трехканальный релейный модуль на базе реле SRD-05VDC-SL-C. Общий вид модуля на рисунке 5. В разделе по разработке аппаратной части будет раскрыты основные возможности устройства. Модуль был выбран из-за простоты в использовании и небольшой цены.

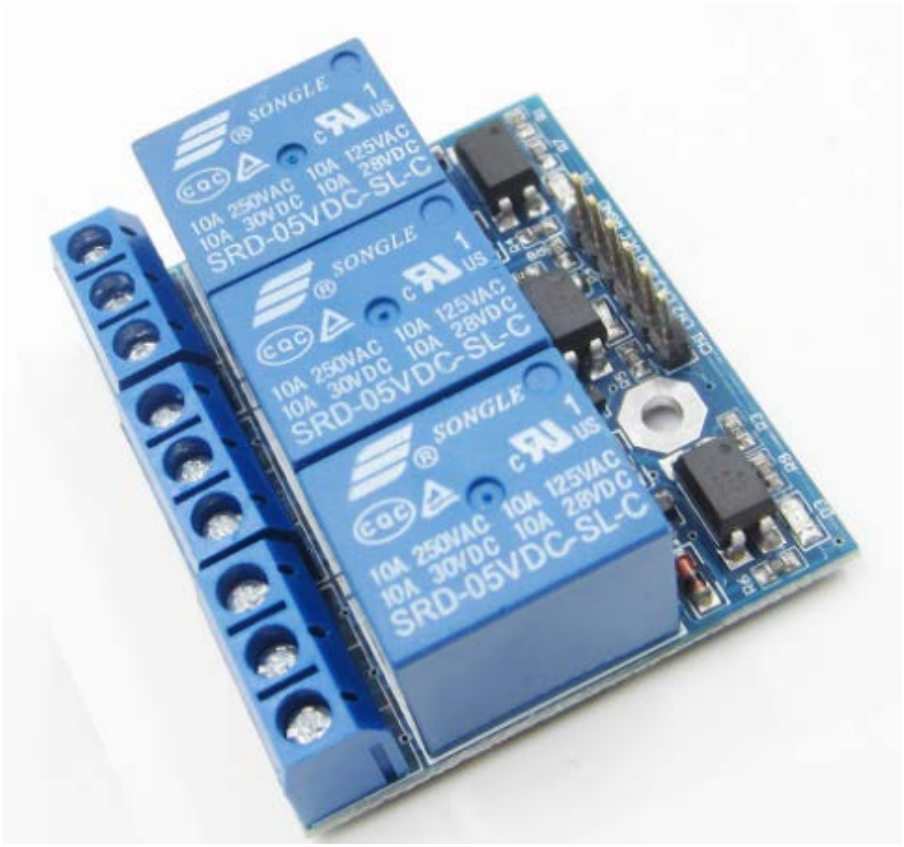


Рисунок 5 – Релейный модуль

Лабораторный стенд уже имеет в себе встроенный измеритель параметров электрической сети. Измеритель щитовой фирмы Omixмодели P94-MX-1-0.5. Имеет цифровой интерфейс RS485, внешний вид на рисунке 6.

Рисунок 6 – Измеритель OmixP94-MX-1-0.5

Преобразователь интерфейсов USB/RS-485 от фирмы Овен требуется для получения данных с измерителя по интерфейсу RS-485 и преобразования в USB. Внешний вид на рисунке 7.

Рисунок 7 – Преобразователь интерфейсов USB/RS-485

Далее нужно выбрать управляющее устройство.На самом деле, было рассмотрено очень много альтернатив.

Сначала была связкаОвен ПЛК100 и обычного ПК. ПК должен был выполнять роль веб-сервера, а модем Овен ПМ01 подсоединен к ПК. На модеме сим-карта со статическим IP, а передача данных по ModbusTCP/IP.

Позже была идея заменить ПК на удаленный сервер и преобразовывать данные с ПЛК библиотекой mod4j, написанной на Java.

Потом на глаза попалось семейство мини-компьютеров RaspberryPi. В этом пункте не стану описывать все характеристики, лишь назову преимущества и другие альтернативы.Среди альтернатив были и контроллерные платы Arduinoи megaAVR, но были отложены в сторону из-за недостаточной гибко-

сти в плане разработки программ, хранении данных и использовании сетевых преимуществ стационарных ПК.

Микрокомпьютеры такие как Banana Pi, RaspberryPi, Cubieboard4 имеют достаточно мощную начинку, чтобы быть веб-сервером и контроллером одновременно. В итоге был выбран RaspberryPi 2 ModelB [1]. Внешний вид на рисунке 8.

Рисунок 8 – RaspberryPi 2 ModelB

4Gмодем выбирается по усмотрению, в проекте используется WI-FI-роутер Yotaи USB-приемник для RaspberryPi. Полную структуру и подробности о платформе можно найти в приложении Е.

2РАЗРАБОТКА АППАРАТНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

2.1 Детальная информация по аппаратной реализации системы

Необходимо описать все доступные характеристики основных аппаратных решений, задействованных при построении системы [2].

2.1.1 RaspberryPi 2 ModelB

Втаблицу 1 занесли основные характеристики выбранной модели.

Таблица 1 – Основные характеристики RaspberryPi

Спецификация	Пояснение
Чип	Broadcom BCM2836 SoC
Архитектура ядра	Quad-core ARM Cortex-A7
CPU	900 MHz
GPU	МультиформатныйсопроцессорVideoCoreIV® Поддержка OpenGLES 2.0, аппаратноеускорениеOpenVG, и 1080p30; высококачественное декодирование H.264 Поддержка 1Gpixel/s, 1.5Gtexel/s или 24GFLOPsc фильтрацией текстур а так же инфраструктура DMA
ОЗУ	1GB LPDDR2
Операционные системы	Загрузка из MicroSDкарты, поддержка различных Linux-системи Windows 10
Габариты	85 x 56 x 17mm
Питание	Micro USB socket 5V, 2A
Ethernet	10/100 BaseT Ethernet socket
Видео выход	HDMI (rev 1.3 & 1.4)
Аудио выход	3.5mm jack, HDMI
USB	4 x USB 2.0 Connector
Интерфейс ввода/вывода GPIO	40-пин 2.54 mmрасширенный контактор: полоса 2x20 Поддержка 27 GPIO пинов,а также линий питания +3.3 V, +5 VиGND
Слот для карты памяти	Micro SDIO

Так же в таблицу 2 поместили данные о портах ввода/вывода.

Таблица 2 – Входы/выходы компьютера.

Пин	Имя	Пин	Имя
01	3.3v DC Power	02	5v DC Power
03	GPIO02 (SDA1, I2C)	04	5v DC Power
05	GPIO03 (SCL1, I2C)	06	Ground
07	GPIO04 (GPIO_GCLK)	08	GPIO14 (TXD0)
09	Ground	10	GPIO15 (RXD0)
11	GPIO17 (GPIO_GEN0)	12	GPIO18 (GPIO_GEN1)
13	GPIO27 (GPIO_GEN2)	14	Ground
15	GPIO22 (GPIO_GEN3)	16	GPIO23 (GPIO_GEN4)
17	3.3v DC Power	18	GPIO24 (GPIO_GEN5)
19	GPIO10 (SPI_MOSI)	20	Ground
21	GPIO09 (SPI_MISO)	22	GPIO25 (GPIO_GEN6)
23	GPIO11 (SPI_CLK)	24	GPIO08 (SPI_CE0_N)
25	Ground	26	GPIO07 (SPI_CE1_N)
27	ID_SD (I2C ID EEPROM)	28	ID_SC (I2C ID EEPROM)
29	GPIO05	30	Ground
31	GPIO06	32	GPIO12
33	GPIO13	34	Ground
35	GPIO19	36	GPIO16
37	GPIO26	38	GPIO20
39	Ground	40	GPIO21

Стоит рассказать об интерфейсе GPIO и о шине I^2C.

GPIO – интерфейс ввода/вывода общего назначения. Используется для связи микропроцессоров и периферийных устройств. Контакты могут быть как входами, так и выходами, зависит от конфигурации (программы). Питание у RaspberryPiобщее и питает все входы/выходы, это избавляет от подвода питание к каждому GPIOпо отдельности. Все GPIOцифровые.

I^2C – последовательная ассиметричная шина для связи между интегральными схемами внутри электронных приборов. Для передачи данных ис-

пользуются 2 двунаправленные линии связи (SDAи SCL). Классическая адресация включает 7-битное адресное пространство с 16 зарезервированными адресами. Это означает, что разработчикам доступно до 112 свободных адресов для подключения периферии на одну шину.Датчик BMP280 подключается именно к этой шине.

2.1.2 Датчик давления и температурыBMP280.

Характеристики датчика в полной мере отображает таблица 3.

Таблица 3 – Характеристики Bosch BMP280;

Параметр	Значение
Рабочее давление	300 – 1100 hPa
Точность по давлению	1 hPa
Тип выхода	Цифровой
Тип интерфейса	I2C, SPI
Рабочая температура	-45– 85°C
Рабочий ток источника питания	2.8 uA
Напряжение питания	1.71 –3.6 V

2.1.3Датчик температуры и влажностивоздухаDHT11.

Как и в остальных случаях, таблица – лучший формат отображения характеристик устройства, в таблице 4 все данные.

Таблица 4 – Характеристики датчика DHT11.

Параметр	Значение
Напряжение питания	3-5 V
Диапазон температур	0-50 °C
Погрешность температуры	±2 °C
Диапазон влажности	20–90%
Погрешность влажности	±5%

2.1.4 Релейный модуль на базе реле SRD-05VDC-SL-C

Модуль обеспечиваетвозможность одновременного коммутирования трех нагрузок переменного и постоянного тока. В таблице 5 перечислены основные характеристики.

Таблица 5 – Характеристики релейного модуля.

Uком AC	Uком DC	Iком AC	Iком DC	Uупр	Iупр	Каналы
До 250В	До 30В	До 10А	До 10А	3.3В - 5В	До 5мА	3

2.1.5 Измеритель токовых параметров OmixP94-MX-1-0.5

Измеритель электрических параметров Omix предназначен для измерения параметров однофазной электрической сети, таких как напряжение, ток, частота, полная, активная и реактивная мощности и cos φ, а также для сигнализации об изменении этих величин с помощью выходных коммутационных устройств. Прибор выполняет функции 7 обычных устройств, таких как амперметр, вольтметр, измеритель 3-х типов мощности (активной, реактивной, полной), измеритель коэффициента мощности cos φ, частотомер.

В таблице 6 перечислены основные характеристики.

Таблица 6 – Характеристики измерителя.

Параметры	Значение
Напряжение питания	~220В +10/-15% 50 Гц
Потребляемая мощность	< 5 Вт
Количество выходных реле	2
Нагрузочная способность реле	~220В 10А / =28В 8А
Переменное напряжение (измерение)	~5-500В через прямое подключение
Переменный ток	~0-5А через прямое подключение
Активная мощность	0-2500 Вт через прямое подключение
Реактивная мощность	0-2500 ВАР через прямое подключение
Полная мощность	0-2500 ВА через прямое подключение
Частота	15...120 Гц через прямое подключение
Коэффициент мощности cos φ	0...1 через прямое подключение

2.2 Разработка полной электрической схемы устройства

В этом пункте будет получена полная электрическая схема устройства и описаны принципы подключения основных компонентов. Электрическая схема выполняетсяв соответствии с правилами государственных стандартов ГОСТ 2.721-74 и ГОСТ 2.702-75.

2.2.1 Подключение датчика давления и температурыBMP280.

Как и говорилось выше, подключение датчика к RaspberryPiбудет осуществляться через шину I^2C. Питание подводится из компьютера. На рисунке 9 показана электрическая схема подключения датчика.

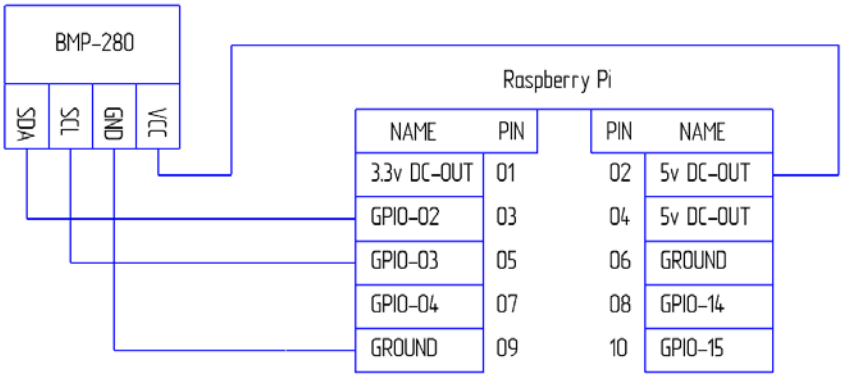


Рисунок 9 – Подключение BMP280

2.2.2 Подключение датчикатемпературы и влажности DHT11.

На рисунке 10 показана схема подключения DHT11. Для цифрового выхода задействован обычный GPIO-22, номер пина –15.

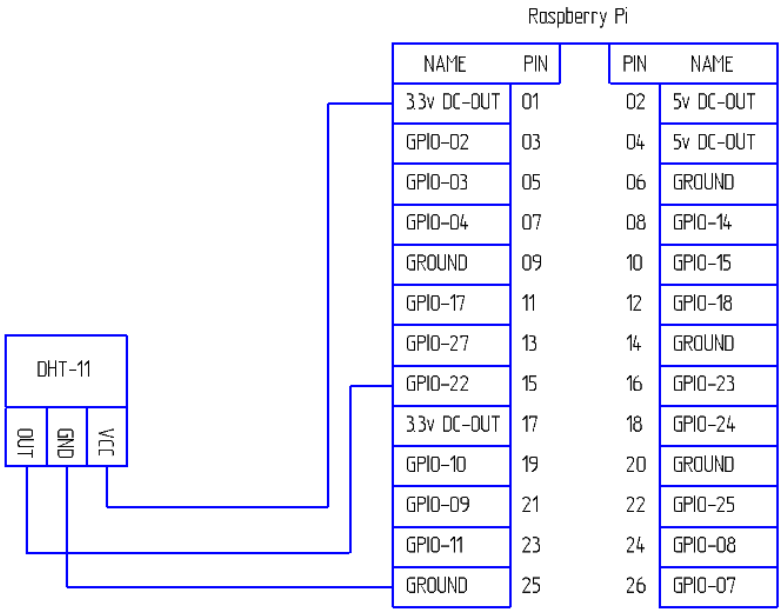


Рисунок 10 – Подключение DHT11

2.2.3 Подключение ламп освещения и реле.

Коммутационное реле, выбранное в прошлых пунктах, имеет 3 канала подключения нагрузки. На рисунке 11 показана схема входов/выходов реле.

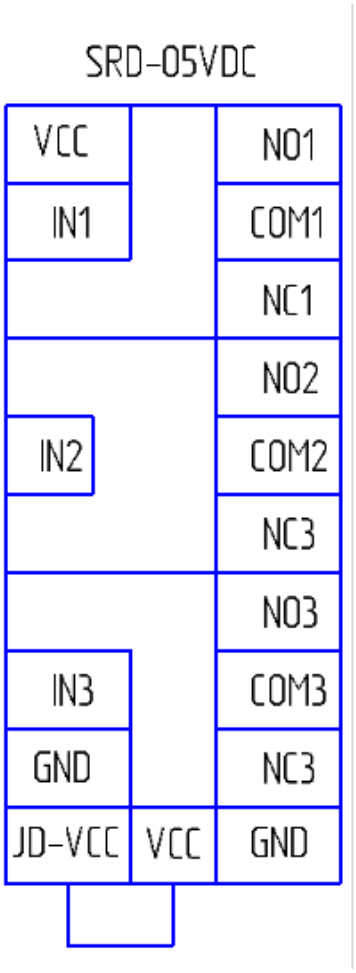


Рисунок 11 – Схема входов/выходов реле

Обозначения на схеме:

−VCC – питание (3.3В-5В);

−GND – заземление;

−IN1, IN2, IN3 – дискретные входы управления;

−JD-VCC, VCC, GND – перемычкой задается активный уровень, при котором срабатывает реле. JD-VCC+VCC – активный уровень 1. VCC+GND – активный уровень 0;

−COM1, COM2, COM3 – общий контакт для нагрузки;

−NO1, NO2, NO3 – контакт, подключаемый к сети при активном уровне равным единице;

–NC1, NC2, NC3 – контакт, подключаемый к сети при активном уровне равным нулю;

Схема подключения нагрузки к реле показана на рисунке 12. Полная схема с подключением реле к RaspberryPiв приложении Б. На схеме показан общий принцип подключения нагрузки через реле к компьютеру.

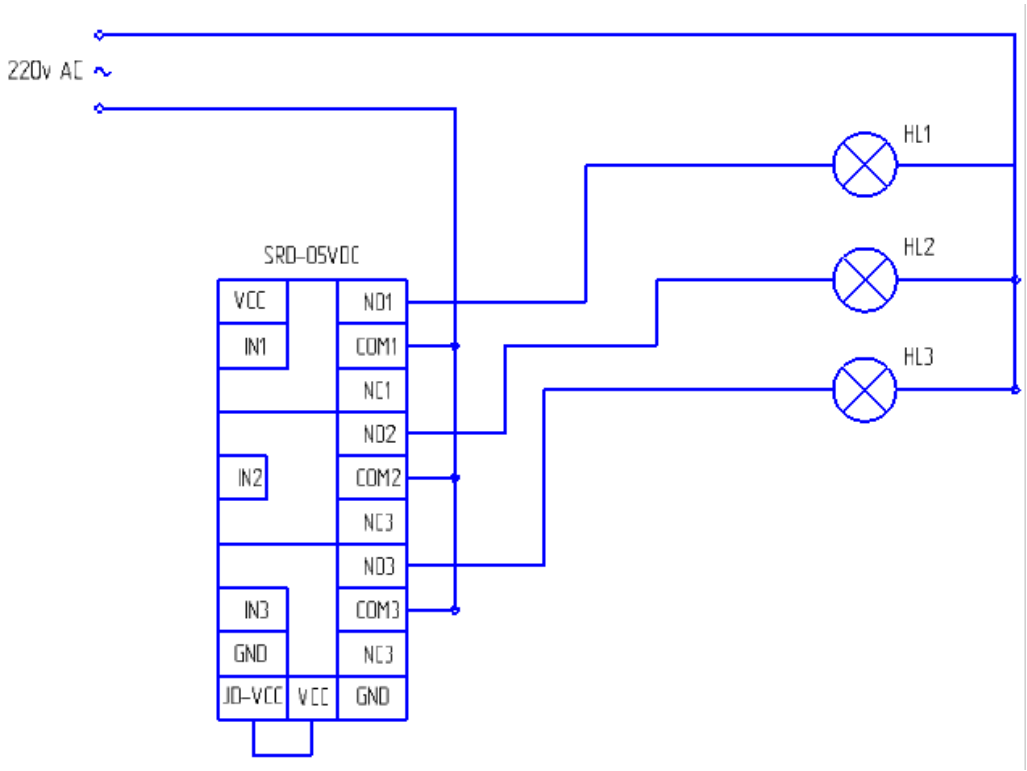


Рисунок 12 – Схема подключения нагрузки к реле

2.2.4Подключение измерителя параметров тока.

Схема подключения измерителя на рисунке 13. Полная схема в приложении Б, она имеет мало различий со схемой системы обмена по CSD, поэтому более подробное описание опускается.

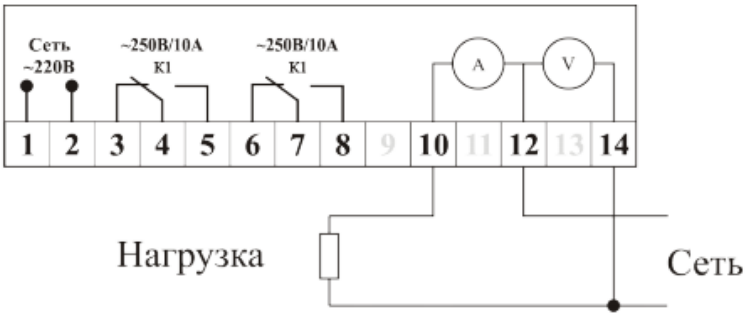


Рисунок 13 – Подключение измерителя токовых параметров

3 РАЗРАБОТКА ПРОГРАММНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

3.1Установка и настройка программного обеспечения

В этом пункте попытаемся установить и настроить операционную систему, опишем известные альтернативы программного обеспечения для разработки и работы системы.

Для RaspberryPiсуществует куча дистрибутивов (ОС) для разных задач. RetroPieделает из платы игровую приставку для ретро-игр, с помощью других можно сделать свой медиа-центр или станцию по слежению за спутниками. Elastix VOIP System вообще позволяет практически без предварительных настроек наладить IP-телефонию, запуститьпочтовый или факс-сервер, сервер конференций или сервер обмена сообщениями [4]. В общем, применений много, но главным преимуществом компьютера являются его порты ввода/вывода, которые позволяют управлять внешними устройствами и получать от них рабочие данные.Например, в зависимости от влажности воздухаможно запускать вентиляционное и сушильное оборудование или от соотношения показателей температуры снаружи и внутри включать отопление. Сфера применения этих технологий ограничивается лишь фантазией. Реле и магнитные пускатели подключаются аналогично тем же ПЛК150 или Simatic S7-300.

Raspbian – официальный дистрибутив для RaspberryPi, его мы и будем использовать. Согласно документации, следует форматировать micro-SDкарту и записать на нее содержимое образа дистрибутива. На рисунке 14 – скачиваемый образ на сайте.

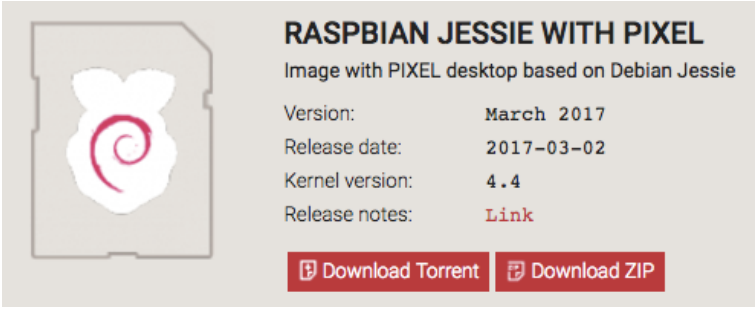


Рисунок 14 – Ссылки на образ

Затем вставляем карту в слот компьютера и запускаем систему. На рисунке 15 и 16показан процесс установки ОС.

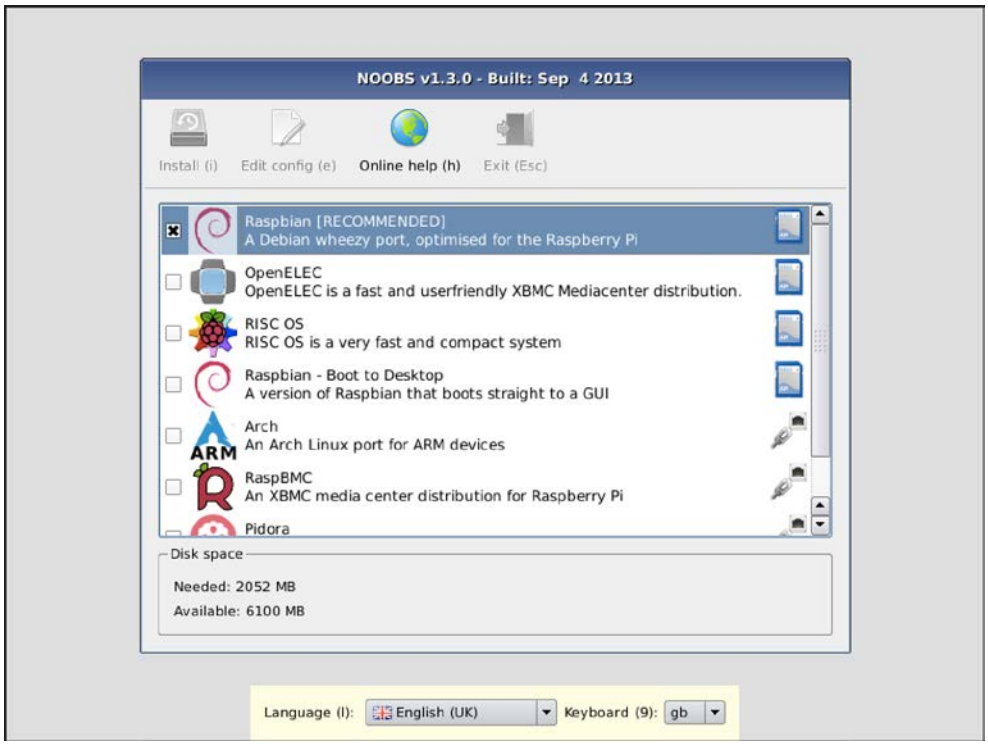


Рисунок 15 – Процесс установки системы

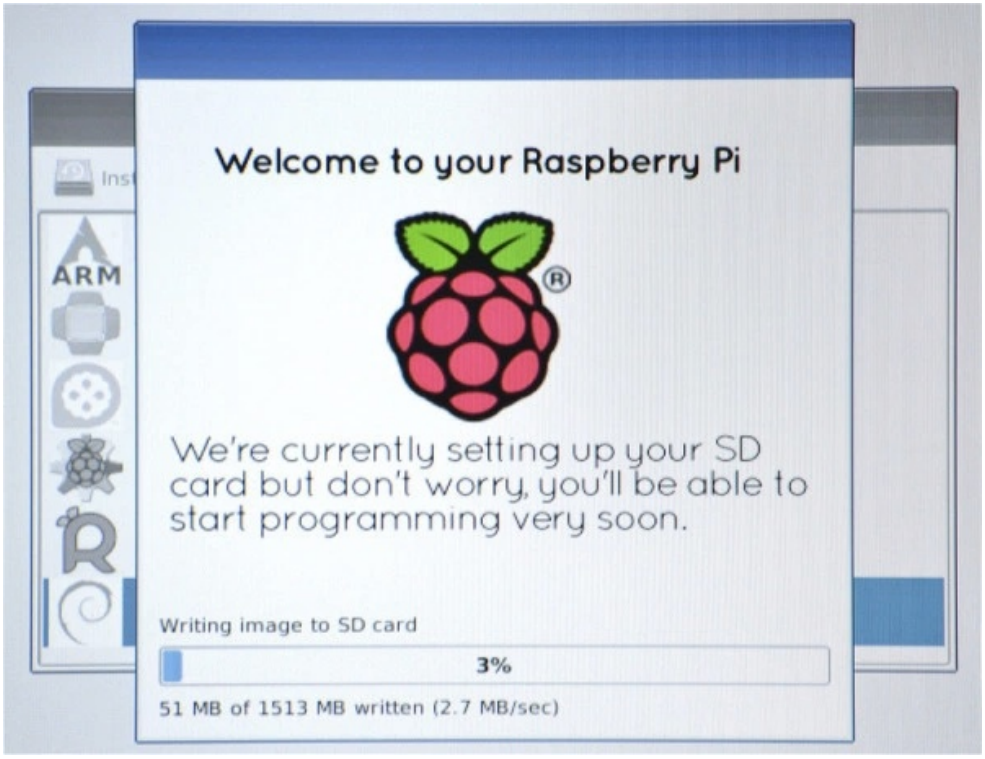


Рисунок 16 – Процесс установки системы

Установленная ОС предстает в виде, показанном на рисунке 17.

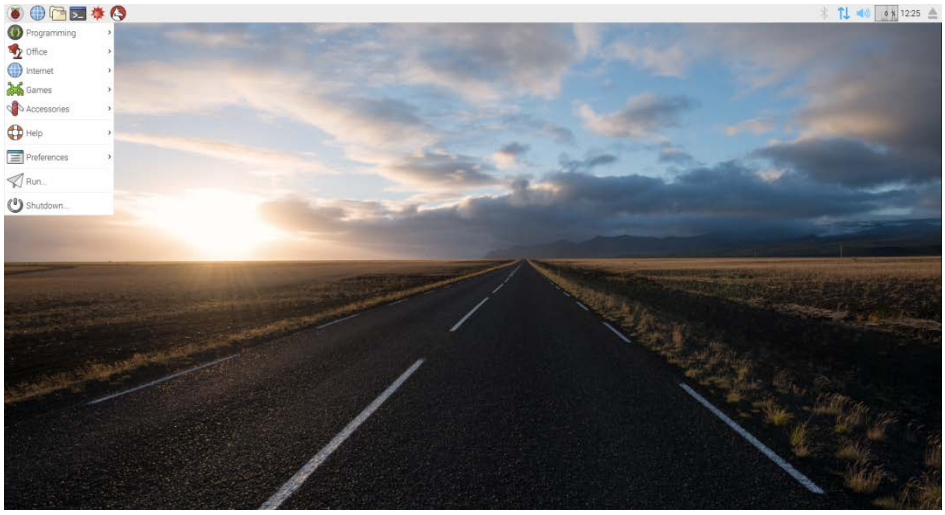


Рисунок 17 – Рабочий стол Raspbian

Далее нужно подключиться к локальному Wi-Fi, это позволит удобно работать с устройством через SSH. У RaspberryPiвторой модели нет встроенного модуля Wi-Fi, используем USBWi-Fi-адаптер, драйверов для негообычно не требуется.

Подключиться по SSHможно через командную строку или через специальные файловые менеджеры. Перед тем, как подключиться, в настройках Raspbianнужно включить все поддерживаемые интерфейсы, включая SSH. На рисунке 18 это демонстрируется.

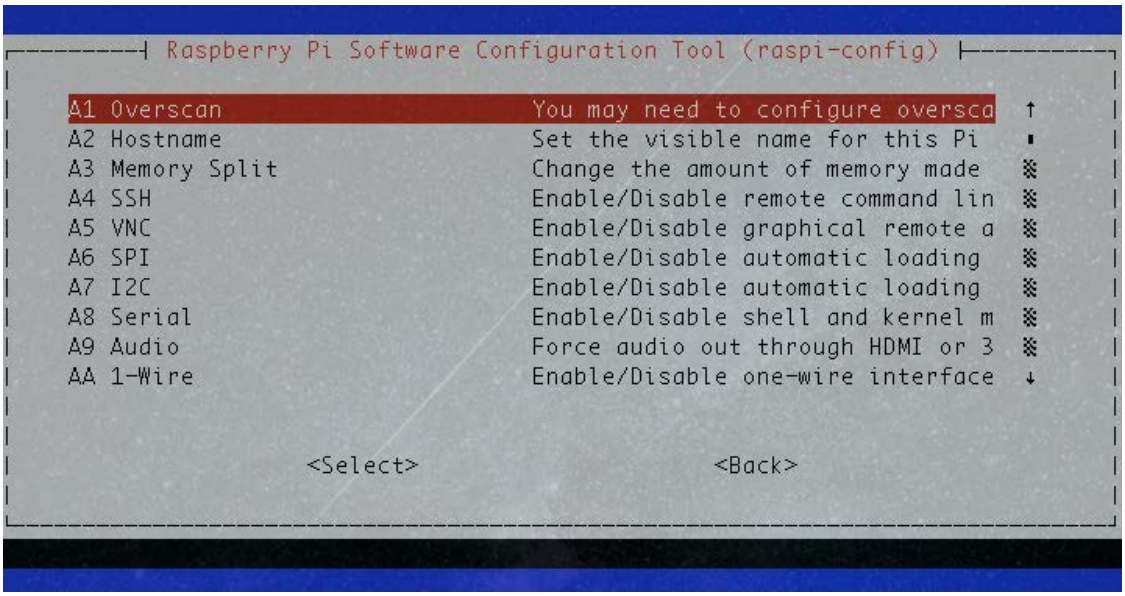


Рисунок 18 – Меню настроек

В командной строке вводим команду (без //):

// sshpi@ip

Где pi – имя пользователя в системе;

ip– локальный IPRaspberry.

После нужно ввести пароль, по умолчанию он «raspberry». После входа можно выполнять все доступные linux-команды, работать с каталогамии устанавливать программы.

Для работы с каталогами и файлами лучше всего использовать файловый менеджер, например, CyberDuck. Внешний вид интерфейса показан на рисунке 19.

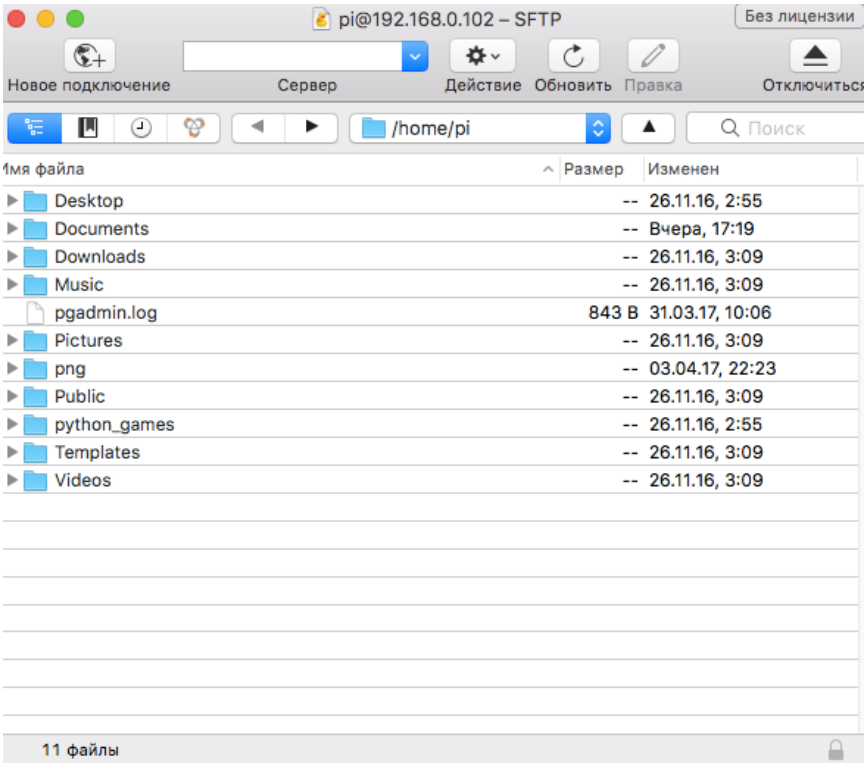


Рисунок 19 – CyberDuck

Писать ПО будем на языке Python, по умолчанию он уже установлен. Pythonбыл выбран из-за возможности малыми конструкциями делать серьезные вещи, плюс существует обильное количество библиотек для работы с сервисами баз данных, GPIOмалины (Raspberry), вебом и с др [14].

По плану готовая система должна уметь хранить данные с датчиков в базе данных и отдавать их пользователю в виде графиков по запросу. Систе-

мой управления базами данных (СУБД) была выбрана PostgreSQL. MySQL, MongoDB, OracleDBтоже рассматривались. Но для хранения малых объемов данных без отношений подошла бы любая БД. Была даже мысль хранить данные в XMLили CSV. В итоге было решено остановиться на Postgres.

Очевидные плюсы PostgreSQL:

- Поддержка пула соединений;
- Обширный набор типов данных;
- Поддержка индексов;
- Простота установки;
- Удобный синтаксис;

Установка СУБД производится следующими командами (нужно быть rootв системе):

```
// sudo apt-get install postgresql-9.4
```

Для запуска сервиса команда следующая:

```
// sudo service postgresql start
```

Для того, чтобы сервис СУБД стартовал каждый раз при запуске системы автоматически, в файле /etc/rc.local нужно прописать предыдущую команду с “&” в конце. Внутренний вид файла после изменений должен выглядеть как на рисунке 20.

Рисунок 20 – Файл rc.localпосле изменений

Еще нужно установить панель администрирования Postgres, через которую можно будет создавать БД и таблицы к ним в GUI:

```
// apt-get install pgadmin3
```

ДляPythonустанавливаемдрайверPostgres:

```
// pip install psycopg2
```

Теперь можно писать SQL-скрипты и инициализировать их из Python в созданную БД.

С библиотекой по созданию графиков познакомимся попозже[13].Сейчас только установим необходимые зависимости:

```
// pip3 install matplotlib
// sudo apt-get install python-numpy
```

После основных настроек и установок хотелось бы рассказать о разрабатываемом проекте и о ключевой технологииПО в целом.На рынке мобильных приложений на ряду с WhatsApp, Facebookmessengeri др. существует мессенджер Telegram [16]. Сервис для обмена сообщениями выделяется среди конкурентов следующим:

- Высокая стабильность работы;

- Защищенность общения (SecretChat), где весь траффик шифруется от третьих лиц;

- Возможность создавать ботов (искусственный собеседник), которых можно программировать на выполнение разного рода задач;

- Передача больших объемов данных между чатами;

- Большое количество тематических каналов со вместимостью до 5000 человек.

Боты – одна из самых любопытных вещей, придуманных в Telegram. Есть боты для рассылки новостей, переводчики, для подборки книг, музыки и кино, для отслеживания почты и др. На рисунках 21, 22, 23самые популярные из ботов [11].

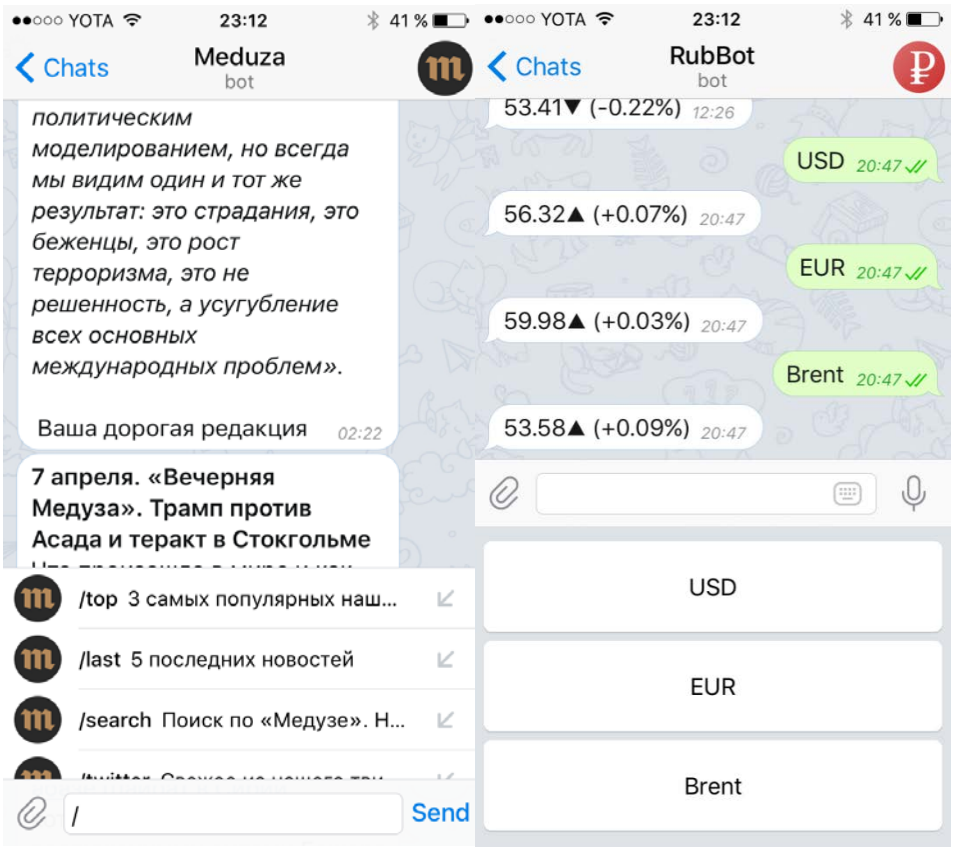


Рисунок 21, 22 – Новостной ботот Meduza и для проверки курса валют

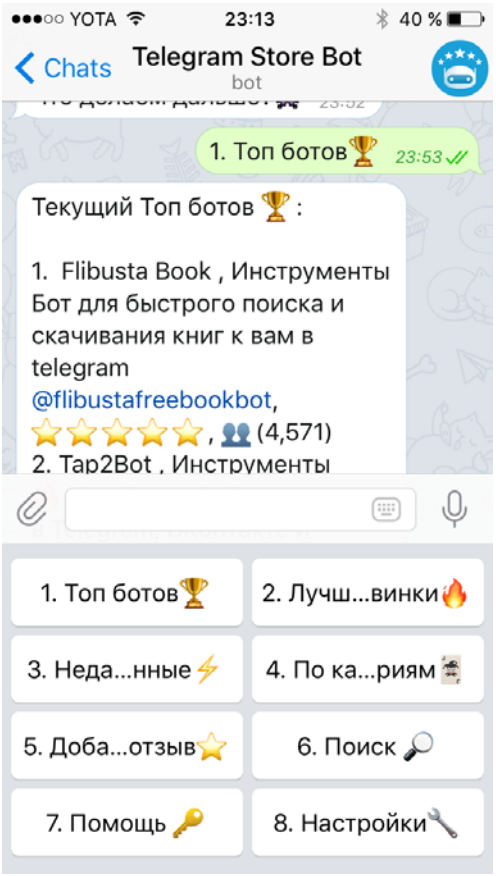


Рисунок 23 – Бот-магазин для поиска других ботов

С помощью ботов так же можно управлять электроникой, которая можетвыходить в интернет и взаимодействовать с APIот Telegram посредством библиотек на C++, Pythonи др.Этим и является RaspberryPi. В следующем пункте я опишу подробное создание такого бота. Перед этим нужно установить библиотеку по взаимодействию Pythonс серверами Telegram:

```
// pip install pytelegrambotapi
```

Так же установим библиотеку для работы с преобразователем интерфейсовRS-485 toUSB:

```
// pip install pymodbus
```

3.2Разработка общей архитектуры

В предыдущем пункте была даны общие представления оботахTelegram и установлены все необходимые зависимости. В этом будет раскрыт принцип взаимодействия между сервером малины, сервером Telegramи клиентским приложением Telegram, а так жебудут написаны скрипты.

Общая структурная схема системы расположена в приложении В и на рисунке 24.

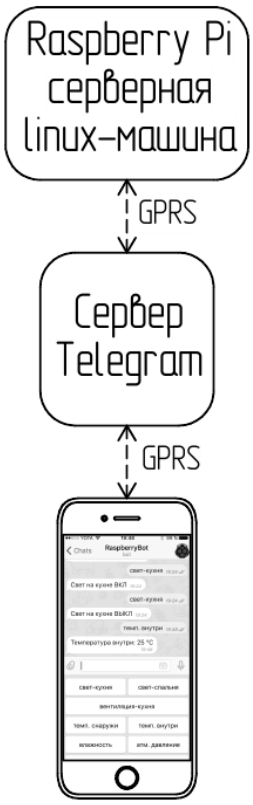


Рисунок 24 – Структурная схема системы

Общий принцип заключается в следующем. Программа в Raspberryциклично опрашивает сервер Telegram, опрашивает по токену (идентификационная строка, выдаваемая программисту после регистрации бота). Пользователь отправляет команды чат боту в виде HTTP-запросов. Запросы в виде очереди хранятся на сервере Telegram, а малина забирает и обрабатывает запросы по очереди при каждом обороте цикла.

Сейчас можно попробовать зарегистрировать своего бота. Нужно в глобальном поиске найти главного бота с именем @BotFather.

Команды:

- /start – начать работу с ботом;
- /newbot–создать нового бота, ввести имя и получить токен;
- /setname – изменить имя у уже существующего бота;
- /setdescrition – изменить описание бота;
- /setuserpic–изменить фото профилябота;
- /deletebot – удалить бота.

На рисунке 25 пример задания имени при создании бота.

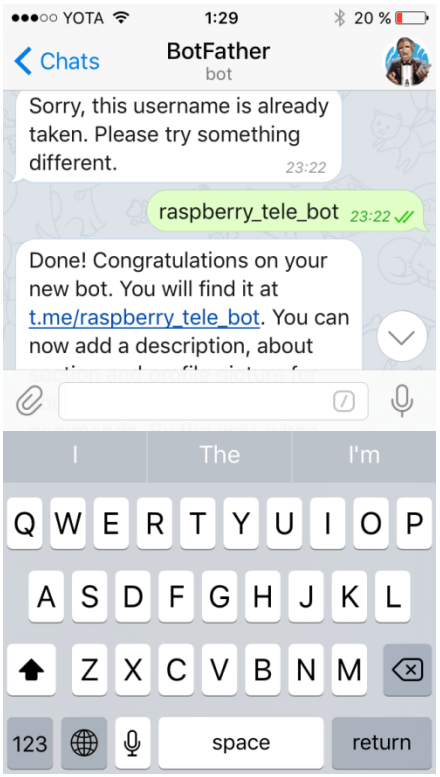


Рисунок 25 – Создание бота

Бот создан, теперь в любом удобном каталоге малины создадим 2 файла: config.pyи bot.py. Начнем с простого и попробуем сделать так, чтобы на каждое отправленное сообщение от пользователя приходил ответ от малины того же содержания.

Содержание config.py на рисунке 26:

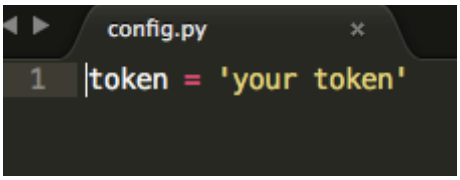


Рисунок 26 – config.py

Содержание bot.py на рисунке 27:

Рисунок 27 – bot.py

На рисунке 28 демонстрация работы.

Импорты вbot.byимпортируют токен и библиотеку telebot. Создается объект botот класса TeleBotс токеном. А метод «repeat_all_messages»с декоратором«@bot.message_handler»захватывает все сообщения от пользователей, если тип контента «text» и отправляет тот же текст обратно.

Вызовметода«polling(none_stop=True)»уобъектаbotцикличен и опрашивает сервер на запросы постоянно. Вот таким нетривиальным способом можно «общаться» со своей малиной. Дальше проект будет только расти и поэтому стоит начать с базы данных, создать структуру и двигаться дальше, тестировать БД будем после создания сервиса БД на Python.

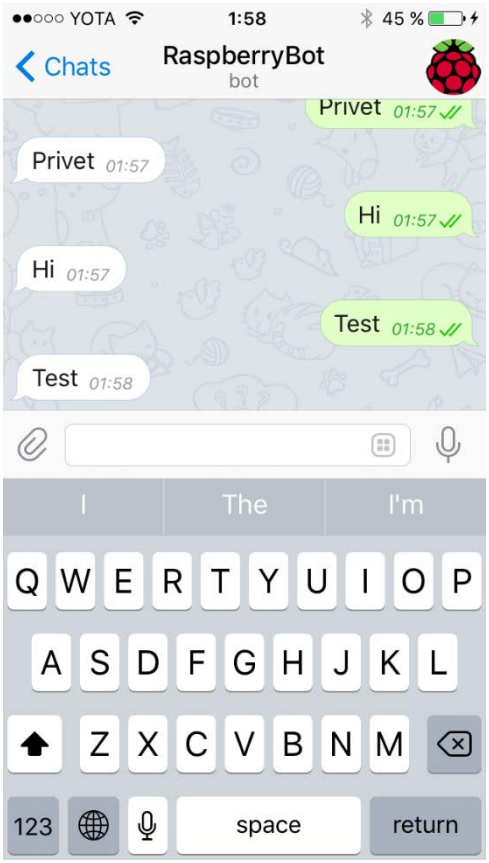


Рисунок 28 – Echo-бот в действии

Напомню, что в базу данных должны писаться следующие параметры:

- Температура снаружи помещения;

- Температура внутри помещения;

- Атмосферное давление;

- Влажность воздуха.

Черезpgadminpanelсоздаемновуюбазуданных «telebot»сталицами «humid_s», «press_s», «temps_in», «temps_out». Сама структура отображена на рисунках 29 и 30.

```
CREATE TABLE humid_s ( id SERIAL PRIMARY KEY,
                       humidity  INTEGER NOT NULL,
                       date_time TIMESTAMP DEFAULT now());

CREATE TABLE press_s ( id SERIAL PRIMARY KEY,
                       pressure  INTEGER NOT NULL,
                       date_time TIMESTAMP DEFAULT now());
```

Рисунок 29 – таблицы«humid_s»и«press_s»

```
CREATE TABLE temps_in ( id SERIAL PRIMARY KEY,
                        temp_in   FLOAT NOT NULL,
                        date_time TIMESTAMP DEFAULT now());

CREATE TABLE temps_out ( id SERIAL PRIMARY KEY,
                         temp_out  FLOAT NOT NULL,
                         date_time TIMESTAMP DEFAULT now());
```

Рисунок 30 - таблицы«temps_in» и «temps_out»

Idy каждой таблицы настроен на автоинкремент, дата и время, если не установлены,берутся актуальные. Ну и сами значения -Integeги Float.Файл инициализации БД называется initDB.sql, полный текст в приложении Г. Наполнением БД на данный момент занимается файл populateDB.sql, тоже в приложении Г, ничего сложного тамнет.

У малины нет встроенного источника питания, из-за такого неудобства возможны сбои во времени, необходима синхронизация по сети при запуске системы. Для этих целей существует утилита NTP-Client [10]. Команда для установки:

// sudo apt-get install ntp ntpdate

Настроим ntpдля нашего региона.

// sudonano /etc/ntp.conf

Ивместо:

server 0.debian.pool.ntp.org iburst dynamic

server 1.debian.pool.ntp.org iburst dynamic

server 2.debian.pool.ntp.org iburst dynamic

server 3.debian.pool.ntp.org iburst dynamic

Нужновставить:

server 0.ru.pool.ntp.org iburst dynamic

server 1.ru.pool.ntp.org iburst dynamic

server 2.ru.pool.ntp.org iburst dynamic

server 3.ru.pool.ntp.org iburst dynamic

ОстановимNTP:

//sudo service ntp stop

Теперьсинхронизируемчасыи запустим сервис:

```
//sudo ntpdate -bs 0.ru.pool.ntp.org
//sudoservicentpstart
```

Для бота в Telegramможно создавать кастомные клавиатуры из кнопок. Это упрощает отправку запроса клиентом. На рисунке 31 показан рабочий вид клавиатуры.

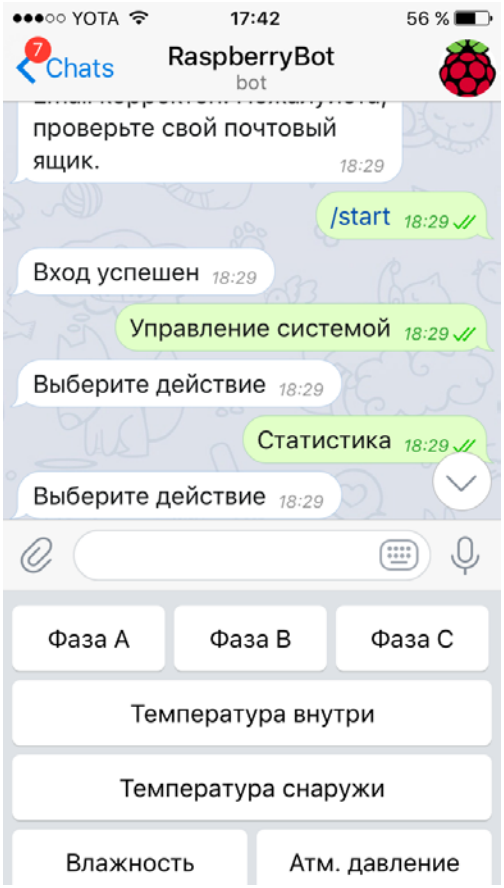


Рисунок 31 – Демонстрация клавиатуры в приложении

На рисунке 32 показан код инициализации простейшей клавиатуры.

Импортируются типы для клавиатури создается объект класса ReplyKeyboardMarkupс именем markup_main. Дальше просто добавляем строки с текстом. Строку можно разделить на 2 и более кнопок по горизонту. После приема какого-либо сообщения клавиатура в чате будет обновляться, это гарантирует аргумент «reply_markup=markup_main»ввызове метода отправки сообщения.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from telebot import types

markup_main = types.ReplyKeyboardMarkup(resize_keyboard=True)
markup_main.row('Фаза A', 'Фаза B', 'Фаза C')
markup_main.row('Температура внутри')
markup_main.row('Температура снаружи')
markup_main.row('Влажность', 'Атм. давление')
markup_main.row('Параметры электр. сети')
markup_main.row('Статистика')
markup_main.row('Закрыть меню системы')
```

Рисунок 32 – Код создания кастомных клавиатур

Попробуем включить нагрузку из приложения. Для этого потребуется импортировать GPIOбиблиотеку. Она позволяет работать с дискретными портами ввода/вывода. На рисунке 33 показан код для инициализации портов и метод, включающий фазу A.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import utils
import telebot
import time
import os

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)

# PHASE_A
@bot.message_handler(commands=['phase_a'])
def phase_a_command(message):
    chat_id = message.chat.id
    if not GPIO.input(27):
        GPIO.output(27,True)
        bot.send_message(chat_id, "Фаза A ВКЛ", reply_markup=utils.markup_main)
        return
    if GPIO.input(27):
        GPIO.output(27,False)
        bot.send_message(chat_id, "Фаза A ВЫКЛ", reply_markup=utils.markup_main)

@bot.message_handler(regexp=u"Фаза A")
def phase_a(message):
    phase_a_command(message)
```

Рисунок 33 – Код на включение фазы A

Сначала выставляется игнорирование предупреждениий режим GPIO.BCM. Устанавливаются GPIO 18, 17 и 27 как цифровые выходы. Хенд-

лер, обрабатывающий регулярное выражение «Фаза А»проверяет 27 порт на наличие сигнала. Если нет, то на выход посылается цифровой сигнал и светодиод горит. Если сигнал есть, то он сбрасывается и свет гаснет. После каждого события пользователю приходит отчет об успешном включении или отключении света. На рисунке 34 показано то, что видит пользователь.

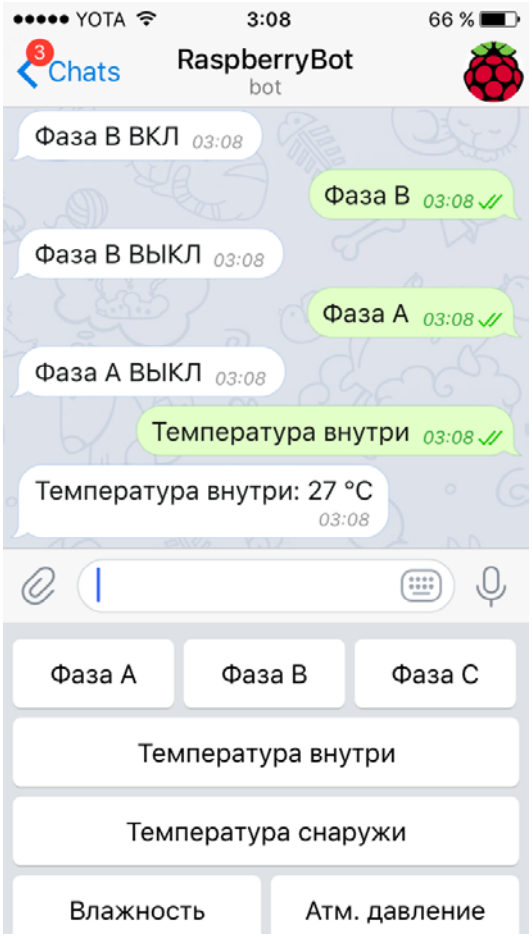


Рисунок 34 – Включение и отключение света

Сервера Telegramограничивают количество обращений от малины. Из-за этого программа может завершаться выбрасыванием исключения, разрыв соединения тоже возможен. Нужно обрабатывать эти ошибки. Для этого будем применять функциональные декораторы, по принципу работы схожие с хендлерами, обрабатывающими регулярные выражения или команды.

Принцип работы декораторов в следующем. Мы просто оборачиваем один метод другим и в случае выполнения определенного условия в обертке, выполняем обернутый метод. На рисунке 35 показан простой декоратор, ко-

торый позволяет обрабатывать ошибки в методах, помеченных конструкцией «@handleError». На рисунке 36 применение декоратора.

```
# EXCEPTIONS DECORATOR
def handleError(function):
    def wrapped(message):
        try:
            return function(message)
        except:
            send_traceback(traceback.format_exc(), message)
    return wrapped
```

Рисунок 35 – Декоратор исключений

```
# PHASE_A
@bot.message_handler(commands=['phase_a'])
@handleError
def phase_a_command(message):
    chat_id = message.chat.id
```

Рисунок 36 – Применение декоратора

Главная функция декоратора принимает помеченный метод, а дальше идет отлов исключений. При появлении исключения весь стек вызовов передается как аргумент методу «send_traceback()», который отправляет сообщение пользователю, который делал запрос к серверу.

На рисунке 37 показан код модифицированного цикла опроса сервера. Тут не столь необходимо обрабатывать ошибки, зависящие от серверов Telegram, их можно игнорировать и перезапускать опрос при каждом выброшенном исключении со стороны сервера. Чтобы их не было, рекомендуется приобрести статический IP-адрес для уменьшения кол-ва обращений к серверу.

```
while True:
    try:
        bot.polling(none_stop=True, interval=1)
    except Exception as e:
        time.sleep(4)
```

Рисунок 37 – Модифицированный цикл опроса сервера

Для считывания показателей с датчиков BMP280 иDHT11 используются готовые скрипты. Считываются данные с регистров и по специальным формулам высчитываются значения температур, давления и влажности.На рисунке 38 показан порядок работы со скриптом BMP280.рудля датчика BMP280.

```python
import utils
import telebot
from BME280 import *

bme_instance = BME280()
bot = telebot.TeleBot(utils.token)

# TEMP OUT
@bot.message_handler(commands=['tmp_out'])
@handleError
def tmp_out_command(message):
    chat_id = message.chat.id
    ps_data = bme_instance.get_data()
    temperature = "Температура снаружи: " + str(ps_data['t']) + " °C"
    bot.send_message(chat_id, temperature, reply_markup=utils.markup_main)

@bot.message_handler(regexp=u"Температура снаружи")
def get_temperature_outside(message):
    tmp_out_command(message)
```

Рисунок 38 – Код на получение температуры снаружи помещения

Импортируем BME280 и создаем экземпляр класса с именем «bme_instance». В каждом методе с хендлером получаем словарь с данными через метод «get_data». И отправляем пользователю давление, переведенное из Паскалей в мм рт. столба, или температуру вградусах по Цельсию. На рисунке 39 пользователь получает показатели.

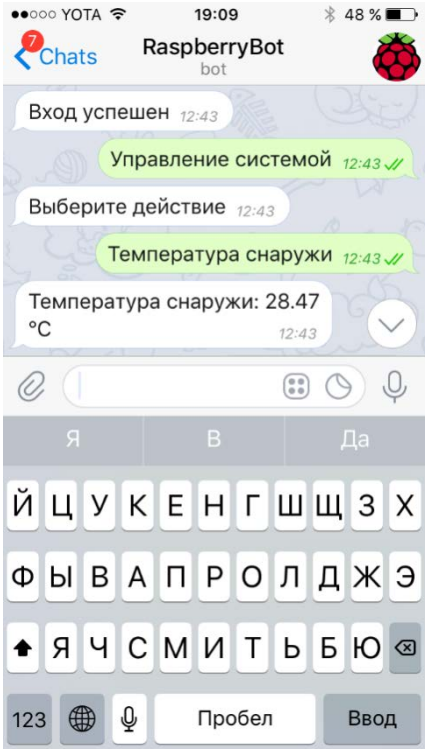


Рисунок 39 – Получение данных с BME280

Работа с датчиком DHT11 практически не отличается от вышеизложенного. На рисунке 40 показан код.

Импортируем dht11 и создаем объект «dht_instance»с переданным номером пина, к которому подсоединен цифровой выход датчика. В методах

«get_temperature_inside» и «get_humidity»производим проверку на валидность полученные данные и отправляем пользователю. На рисунке 41 получаемые пользователем влажность воздуха в процентах и температура внутри помещения в градусах по Цельсию. Погрешность по влажности составляет ±5%. Температурная погрешность ±2 градуса, давление ±10мм рт. столба.

```python
# TEMP IN
@bot.message_handler(commands=['tmp_in'])
@handleError
def tmp_in_command(message):
    chat_id = message.chat.id
    if not check_user_auth(chat_id):
        return

    result = dht_instance.read()
    while result.is_valid() != True :
        result = dht_instance.read()
    temperature = "Температура внутри: " + str(result.temperature) + " °C"
    bot.send_message(chat_id, temperature, reply_markup=utils.markup_main)

@bot.message_handler(regexp=u"Температура внутри")
def get_temperature_inside(message):
    tmp_in_command(message)

# HUMIDITY
@bot.message_handler(commands=['humid'])
@handleError
def humid_command(message):
    chat_id = message.chat.id
    if not check_user_auth(chat_id):
        return

    if message.text == u"Влажность воздуха за сутки":
        get_humidity_last_day(message)
        return
    if message.text == u"Влажность воздуха за неделю":
        get_humidity_last_week(message)
        return
    else:
        result = dht_instance.read()
        while result.is_valid() != True :
            result = dht_instance.read()
        humidity = "Влажность: " + str(result.humidity) + "%"
        bot.send_message(chat_id, humidity, reply_markup=utils.markup_main)

@bot.message_handler(regexp=u"Влажность")
def get_humidity(message):
    humid_command(message)
```

Рисунок 40 – Код для датчика DHT11

Рисунок 41 – Получение данных с DHT11

Перейдем к написанию сервиса работы с базой данных. Файл sensor-dao.русодержит класс «SensorDao». Класс содержит конструктор с настройками подключения к базе данных «telebot». На рисунке 42 показан наполнение конструктора. Конструктор вызывается автоматически при создании экземпляра класса. Импортируется драйвер БД «psycopg2».

```
import psycopg2
import datetime

class DBService(object):
    def __init__(self):
        self.connect = psycopg2.connect("dbname='telebot'" +
                                        "user='postgres'" +
                                        "host='localhost'" +
                                        "password='password'")
        self.cursor = self.connect.cursor()
```

Рисунок 42 – Соединение с базой данных

В этом классе есть 8 методов для каждого параметра (давление и др.). Методы, которые делают выборку за сутки, на данный момент берут данные за один определенный день (23.04.2017). SQL-запрос вытаскивает данные, сортирует по времени и упаковывает в двумерный массив. Ранее я упоминал о SQL-скрипте populateDB.sql, который наполняет каждую таблицу данными

на 1 неделю каждые 2 часа в сутки. На рисунке 43 показан метод, достающий влажность за сутки.

```
def get_humidity_last_day(self):
    self.cursor.execute("SELECT * FROM humid_s "+
                        "WHERE date_time>='2017-03-23 00:00:00' "+
                        "AND date_time<='2017-03-24 00:00:00' "+
                        "ORDER BY date_time ASC")
    rows = self.cursor.fetchall()
    return rows
```

Рисунок 43 – Метод для получения влажности за сутки

На рисунке 44 метод по давлению.

```
def get_pressures_last_day(self):
    self.cursor.execute("SELECT * FROM press_s "+
                        "WHERE date_time>='2017-03-23 00:00:00' "+
                        "AND date_time<='2017-03-24 00:00:00' "+
                        "ORDER BY date_time ASC")
    rows = self.cursor.fetchall()
    return rows
```

Рисунок 44 – Метод для получения давления за сутки

Методы, получающие выборку параметров за неделю представлены на рисунке 45.

```
def get_pressures_last_week(self):
    self.cursor.execute("SELECT * FROM press_s "+
                        "ORDER BY date_time ASC")
    rows = self.cursor.fetchall()
    return rows

def get_humidity_last_week(self):
    self.cursor.execute("SELECT * FROM humid_s "+
                        "ORDER BY date_time ASC")
    rows = self.cursor.fetchall()
    return rows

def get_temps_in_last_week(self):
    self.cursor.execute("SELECT * FROM temps_in "+
                        "ORDER BY date_time ASC")
    rows = self.cursor.fetchall()
    return rows

def get_temps_out_last_week(self):
    self.cursor.execute("SELECT * FROM temps_out "+
                        "ORDER BY date_time ASC")
    rows = self.cursor.fetchall()
    return rows
```

Рисунок 45 – Методы получения данных за неделю

Способ наполнения БД тестовыми данными показан на рисунке 46.

```
import psycopg2
import datetime

self.sqlfile = open('db/populateDB.sql', 'r')
self.cursor.execute(self.sqlfile.read())
self.connect.commit()

self.cursor.close()
self.connect.close()
```

Рисунок 46 – Наполнение БД

Метод для нахождения среднего значения в массиве параметров, например, температурына рисунке 47. Аргументом передается двумерный массив.

```
def average_value(self, rows):
    summ = 0
    for row in rows:
        summ += row[1]
    return float(summ) / len(rows)
```

Рисунок 47 – Метод для нахождения среднего значения

Класс «ModbusSerialService» для работы с измерителем параметров электрической сети находится в файле omix.py. На рисунке 48 показан конструктор класса с инициализацией служебных объектов и подключением к устройству преобразователю интерфейсов и переход в ожидание для чтения параметров.

```
def __init__(self):
    super(ModbusSerialService, self).__init__()
    self.client = ModbusClient(method='rtu', port='/dev/ttyUSB0',
                  stopbits = 1, bytesize = 8,
                  parity = 'N', baudrate=9600)
    if not self.client.connect():
        message = u"Device is not connected."
        log.error(message)
    else:
        log.info(u"Device is connected.")
```

Рисунок 48 – Конструктор класса «ModbusSerialService»

На рисунке 49 показан метод чтения данных с регистров, в которых хранятся параметры напряжения в типе данных Word. Рисунок 50 показывает процесс валидации полученных данных.

```python
def get_voltage(self):
    address = 0
    count = 2
    response = client.read_input_registers(address, count, unit=128)
    return check_valid_data(response, "Voltage")
```

Рисунок 49 – Получение напряжения

```python
def check_valid_data(self, response, data_type):
    if response != None:
        log.info(data_type + u" data is valid.")
        data = str(response[0]) + str(response[1])
        return data
    else:
        log.info(data_type + u" data is not valid.")
        response = "0"
        return response
```

Рисунок 50 – Валидация полученных данных

На рисунке 51 показан код обработки команды от клиента, на рисунке 52 вид глазами клиента.

```python
# ELECTRICITY
@bot.message_handler(commands=['electr'])
@handleError
def electricity_command(message):
    chat_id = message.chat.id
    if not check_user_auth(chat_id):
        return
    if not omix_instance.client.connect():
        bot.send_message(chat_id, u"Устройство не подключено", reply_markup=utils.markup_main)
        return

    data = ('Напряжение: ' + omix_instance.get_voltage() + 'V\n'
            'Сила тока:' + omix_instance.get_current() + 'A\n'
            'Частота:' + omix_instance.get_freq() + 'Гц\n'
            'Мощность:' + omix_instance.get_power() + 'Вт\n')
    bot.send_message(chat_id, data, reply_markup=utils.markup_main)

@bot.message_handler(regexp=u"Параметры электр. сети")
def get_electricity(message):
    electricity_command(message)
```

Рисунок 51 – Код в bot.py

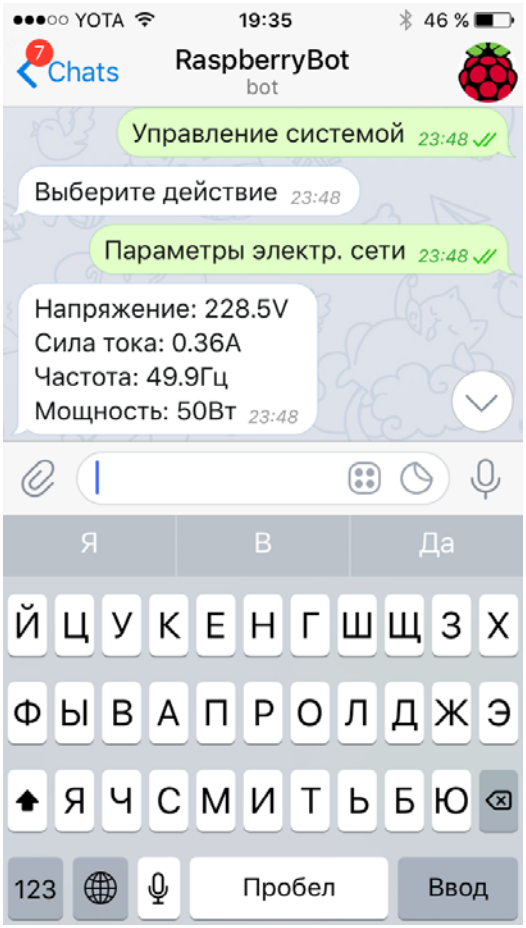


Рисунок 52 – Получение параметров электрической сети

Сервис для работы с графиками больше демонстрационный, чем применяемый на практике. Но все же интересен тем, что приходится делить архитектуру приложения на слои. В итоге скрипт bot.русоздает объект сервиса графиков, а в объекте графиковформируется экземпляр сервиса БД. Такая структура позволяет писать сервисы максимально не зависящие друг от друга. При внесении изменений в один, не придется переписывать другие.

Создаем файл plots.руи импортируем зависимости, показанные на рисунке 53.

```
import os
import matplotlib
matplotlib.use('Agg')
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np

from db import *
```

Рисунок 53 – Необходимые зависимости

В конструкторе класса PlotServiceпроизводится инициализация объекта класса DBService. Так же тут объявляются необходимые массивы для графиков. Конструктор показан на рисунке 54.

```
class PlotService(object):
    def __init__(self):
        super(PlotService, self).__init__()
        self.db_instance = DBService()
        self.x = []
        self.y = []
        self.xticks_day = [0.,2.,4.,6.,8.,10.,12.,14.,16.,18.,20.,22.,24.]
        self.xticks_week = [0., 1., 2., 3., 4., 5., 6., 7., 8.]
```

Рисунок 54 – Конструктор класса PlotService

Массив «xticks_day» содержит данные для оси Хна графике с шагом в 2 часа. Массив «xticks_week»данные по дням недели. «x[]»и «y[]»для параметров (темп. и др.).

Рассмотрим метод для рисования графика температуры за последние сутки на рисунке 55.

```
def get_temps_out_last_day(self):
    xlabel = u"Время, ч"
    ylabel = u"Температура, °C"
    title = u"Температура снаружи за последние сутки"
    temp_instance = self.db_instance.get_temps_out_last_day()

    for i in temp_instance:
        self.y.append(i[1])

    colors = ['red', 'blue']
    self.x = np.linspace(0.0, 24.0, num = 13)
    self.f_plot(self.xticks_day, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)
```

Рисунок 55 – Отрисовка графика

xlabel – метка на оси X;

ylabel–метка на оси Y;

title – заголовокграфика;

temp_instance – массив с данными по температуре.

В цикле записываем температуру в массив «y[]» и загружаем в массив «x[]» диапазон от 0 до 24 часов. Метод «f_plot»принимает метки, заголовок и данные по осям Хи Y. А так же цвета и толщину линии графика.

Метод «f_plot»показан на рисунке 56.

```
def f_plot(self, ticks, title, xlabel, ylabel, *args, **kwargs):
    xlist = []
    ylist = []
    for i, arg in enumerate(args):
        if(i % 2 == 0):
            xlist.append(arg)
        else:
            ylist.append(arg)

    colors = kwargs.pop('colors', 'k')
    linewidth = kwargs.pop('linewidth', 1.)

    fig = plt.figure()
    ax = fig.add_subplot(111)
    i = 0

    for x, y, color in zip(xlist, ylist, colors):
        i += 1
        ax.plot(x, y, color=color, linewidth=linewidth)

    ax.set_xlabel(xlabel, color='b')
    ax.set_ylabel(ylabel, color='b')
    ax.set_title(title, color='b')

    ax.xaxis.set_ticks(ticks)
    ax.grid(True)
    ax.legend()
    self.save('ex_1_6_1', fmt='png')
```

Рисунок 56 – Метод «f_plot»

Этот метод рисует график по переданным в аргументах данных и вызывает метод «save», показанный на рисунке 57.

```
def save(self, name='', fmt='png'):
    pwd = os.getcwd()
    iPath = './pictures/{}'.format(fmt)
    if not os.path.exists(iPath):
        os.mkdir(iPath)
    os.chdir(iPath)
    plt.savefig('{}.{}'.format(name, fmt), fmt='png')
    os.chdir(pwd)
    plt.close()
```

Рисунок 57 – Метод «save»

В аргументах метода «save»передается имя файла и формат. Можно так же сохранить в jpegили pdf.

Отрисовка суточных графиков для влажности и давления отличается лишь вызываемыми методами в сервисе БД.

Рисование семидневных графиков отличается от суточных тем, что за каждый день берется среднее значение и все 7 дней укладываются в ломаную

на осях. Метод «get_temps_out_last_week», вытаскивающий температуру снаружи за 7 дней показан на рисунке 58.

```
def get_temps_out_last_week(self):
    xlabel = u"Дни"
    ylabel = u"Температура, °C"
    title = u"Средняя температура снаружи за неделю"
    temp_instance = self.db_instance.get_temps_out_last_week()

    temps = []
    n = 0
    for i in temp_instance:
        if n == 11:
            val = "%.2f" % self.average_value(temps)
            self.y.append(val)
            temps = []
            n = 0
        temps.append(i[1])
        n+=1

    colors = ['red', 'blue']
    self.x = np.linspace(1.0, 7.0, num = 7)
    self.f_plot(self.xticks_week, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)
```

Рисунок 58 – Метод«get_temps_out_last_week»

Как и говорилось выше, основным отличием здесь является нахождение средних суточных значений и упаковка в массив длиной в 7 символов. Для нахождения среднего числа использовался метод “average_value”, показан на рисунке 59.

```
def average_value(self, rows):
    summ = 0
    for row in rows:
        summ += row
    return float(summ) / len(rows)
```

Рисунок 59 – Метод для нахождения среднего значения

Вот, почти все сервисы готовы, далее нужно доработать скрипт bot.py. Доработка заключается в следующем:

- Написать клавиатуру для статистики;

- Написать и протестировать методы отправки изображений графиков.

Инициализация клавиатуры почти аналогична основной, показана на рисунке 60.

```
markup_stat = types.ReplyKeyboardMarkup()
markup_stat.row('t°C снаружи за сутки')
markup_stat.row('t°C внутри за сутки')
markup_stat.row('t°C снаружи за неделю')
markup_stat.row('t°C внутри за неделю')
markup_stat.row('атм. давление за сутки')
markup_stat.row('влажность воздуха за сутки')
markup_stat.row('атм. давление за неделю')
markup_stat.row('влажность воздуха за неделю')
markup_stat.row('закрыть меню статистики')
```

Рисунок 60 – Клавиатура «markup_stat»

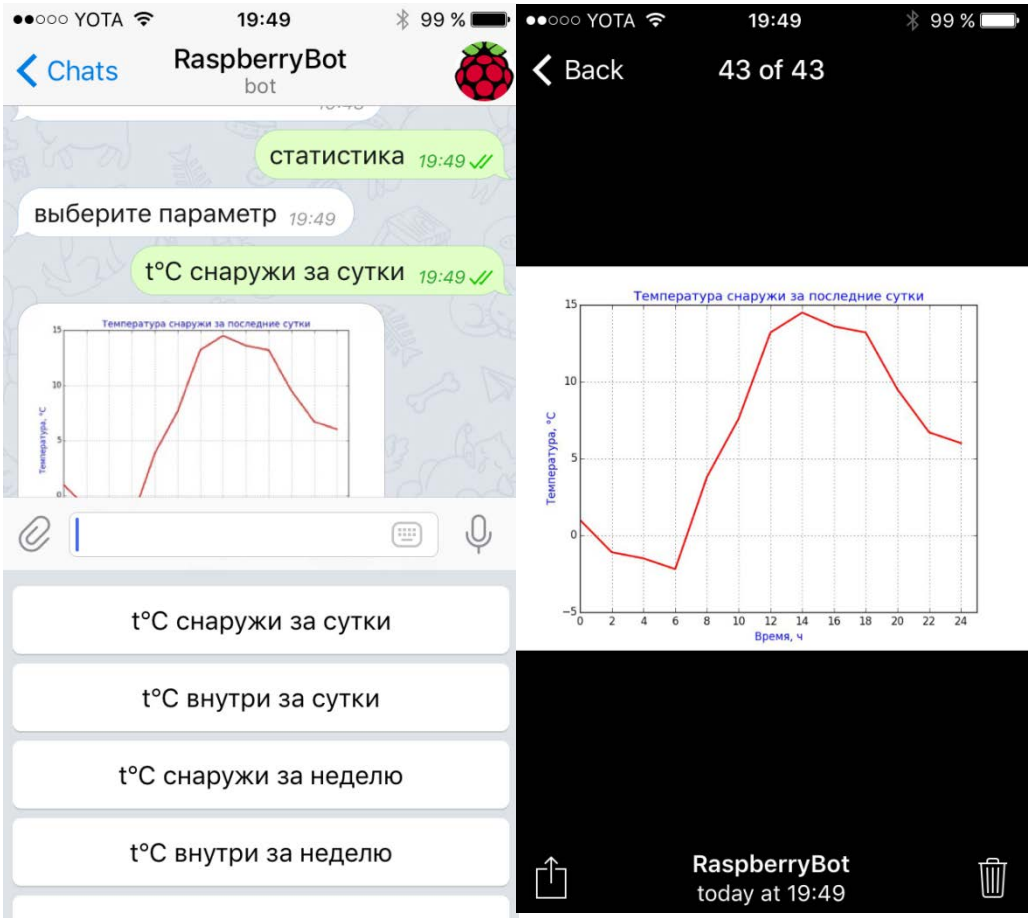
Допишем это: markup_main.row('статистика'). Кнопка понадобится для того, чтобы попасть в меню статистики из главного.

Метод, из которого будет вестись отправка изображения с графиком суточной температуры снаружи здания на рисунке 61.

```
@bot.message_handler(regexp=u"t°C снаружи за сутки")
def get_temps_out_last_day(message):
    plot_instance = PlotService()
    plot_instance.get_temps_out_last_day()
    bot.send_photo(message.chat.id, open('./pictures/png/ex_1_6_1.png', 'rb'))

@bot.message_handler(regexp=u"закрыть меню статистики")
def close_stat_menu(message):
    bot.send_message(message.chat.id, u"выберите параметр", reply_markup=markup_main)
```

Рисунок 61 – Методы отправки изображения и закрытия меню статистики

На рисунках 62 и 63 демонстрируется получение изображения. Остальные методы по своей сути почти не отличаются, меняется параметр только или диапазон выборки.



Рисунки62, 63 – Получение суточных графиков температуры

3.3 Разработка модуля безопасности

Боты всегда открыты для всех пользователей приложения Telegram, их можно найти, воспользовавшись глобальным поиском, это демонстрируется на рисунке 64. Можно не опасаться запросов от незваных гостей, если бот несет функцию переводчика или тому подобное. Но если бот управляет электроникой и имеет доступ к личной информации, то без элементарной защиты по типу пользовательского входа/выхода не обойтись.

Далее будет показана система в готовом виде с небольшими пояснениями к возможным действиям со стороны администратора и рядового пользователя. После, будет дано детальное объяснение того, как это все работает на уровне программ.

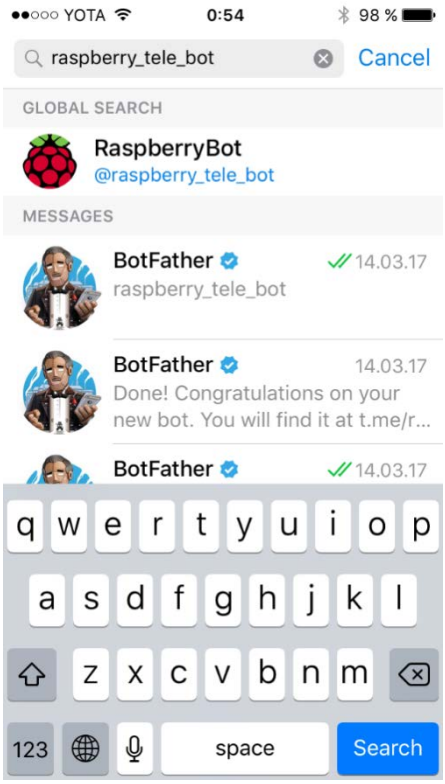


Рисунок 64 – Глобальный поиск контактов в Telegram

При первом «контакте» с ботом мы видим приветствие и возможность вызвать справку командой /help. Командой /loginможно начать процедуру входа в систему. Необязательно запоминать команды, можно производить все операции средствами панельной клавиатуры. В одних случаях удобнее использовать команды, в других клавиатуру. На рисунке 65 показан процесс.

Бот попросит ввести emailпосле команды /login. Нужно быть уверенным, что ваш email уже есть в системе, иначе бот ответит отказом. По поводу регистрации можно обратиться к администратору. Бот отправит emailна введенную почту в случае успеха. На рисунках 66 и 67 показаны ввод email-аи получение письма в случае успешной проверки. Для завершения процедуры входа нужно пройти по ссылке в письме. Рисунок 68 это демонстрирует.

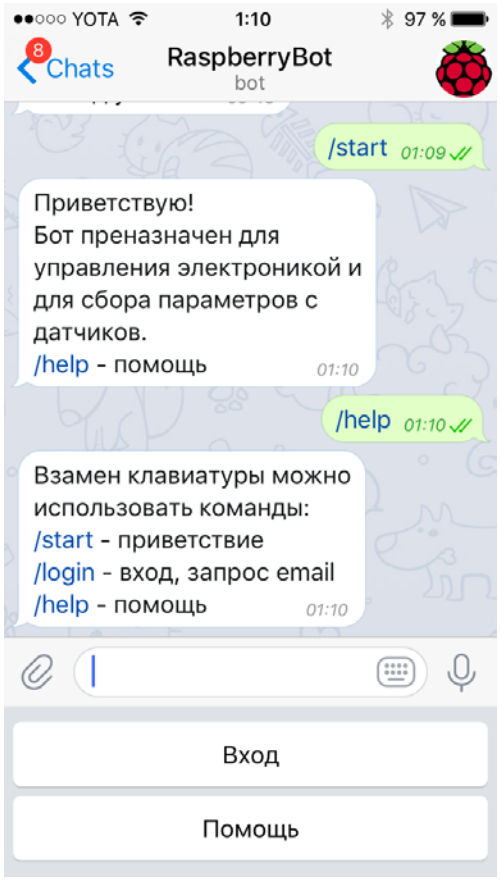
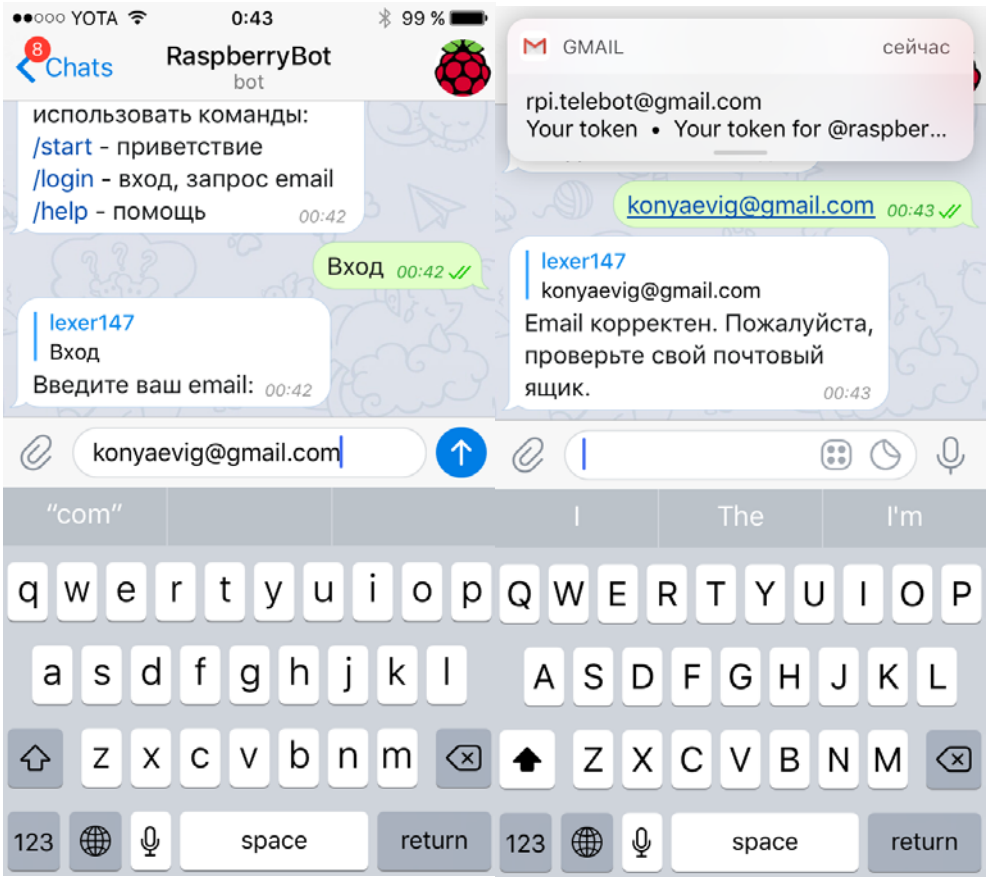


Рисунок 65 – Приветствие и справка



Рисунки 66, 67 – Ввод email-аи получение письма

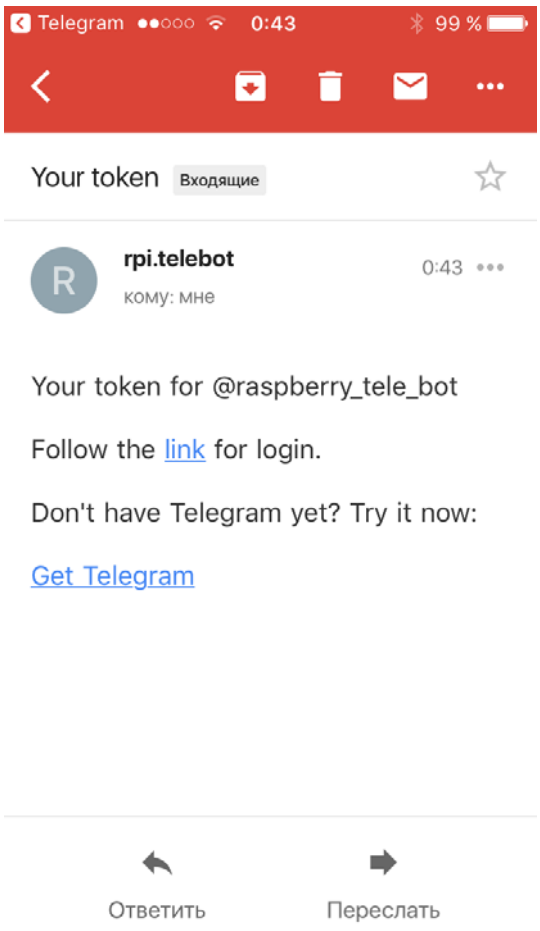
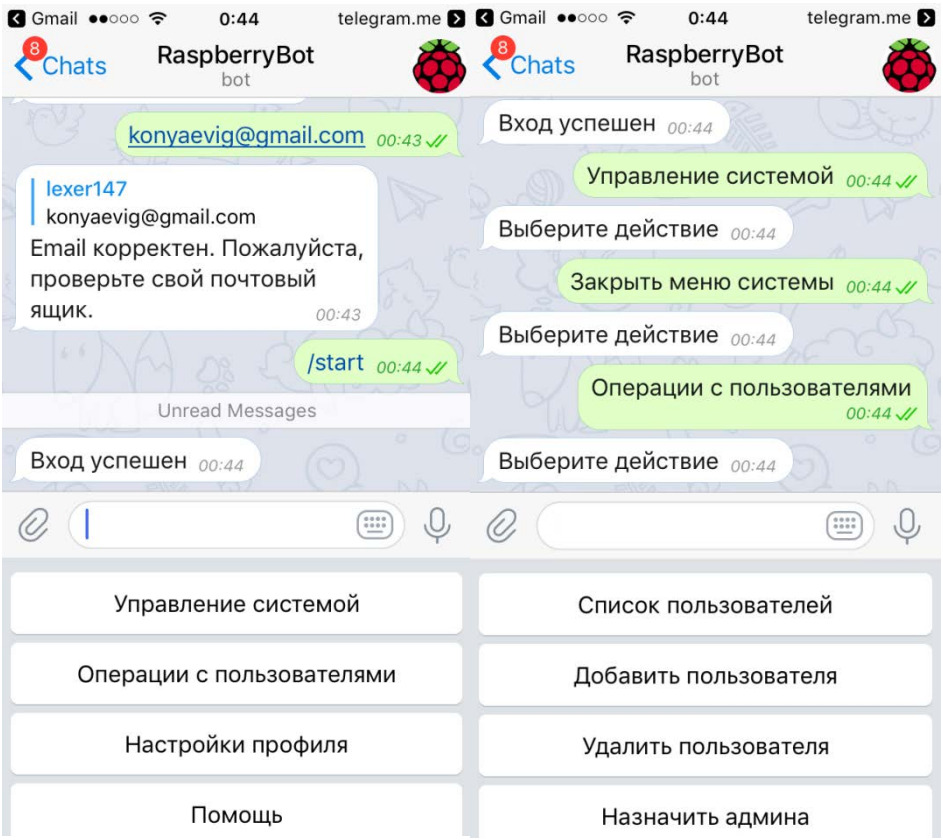


Рисунок 68 – Письмо с ссылкой

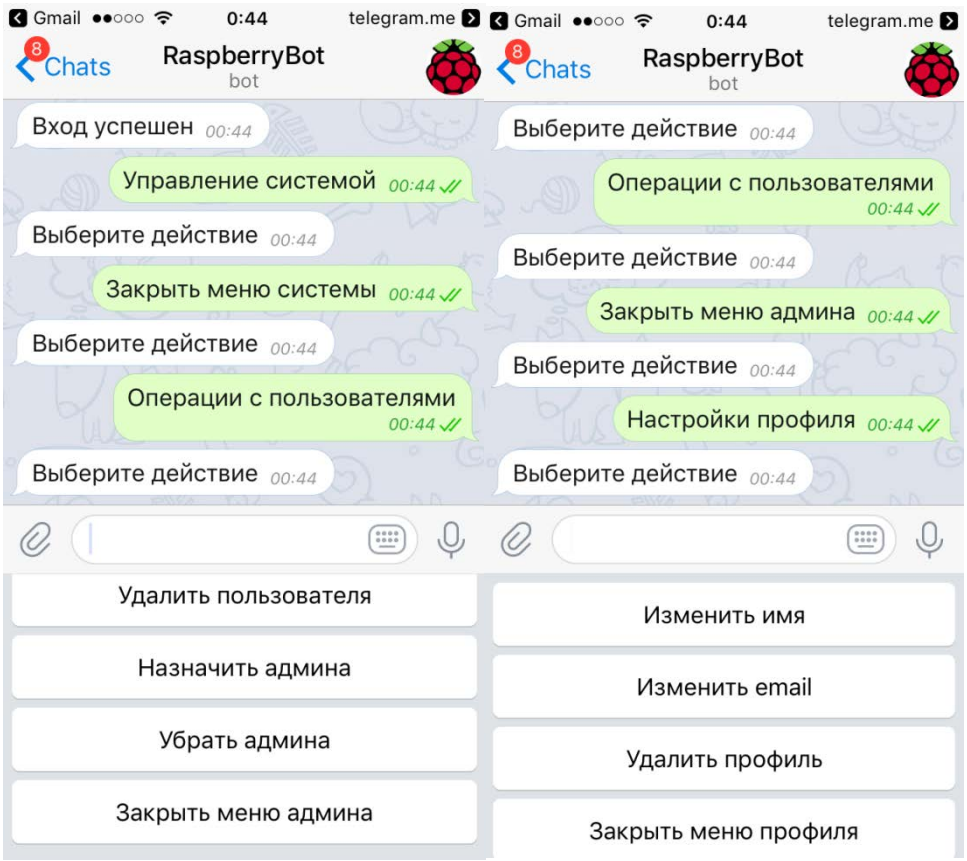
После входа, если вошедший – рядовой пользователь, то будут доступны меню управления системой с подменю статистики, настройки профиля и справка.

Меню управления системой содержит все функции, перечисленные в пункте разработки архитектуры, в настройках профиля можно сменить имя, email, удалить профиль. При изменении email-аили удалении профиля производится автоматический выход из системы. При каждом входе в систему необходимо получать письмо и переходить по ссылке, одна ссылка работает только один раз, это сделано для лучшей безопасности. Плюс всего этого в том, что помнить нужно лишь пароль к вашему ящику.

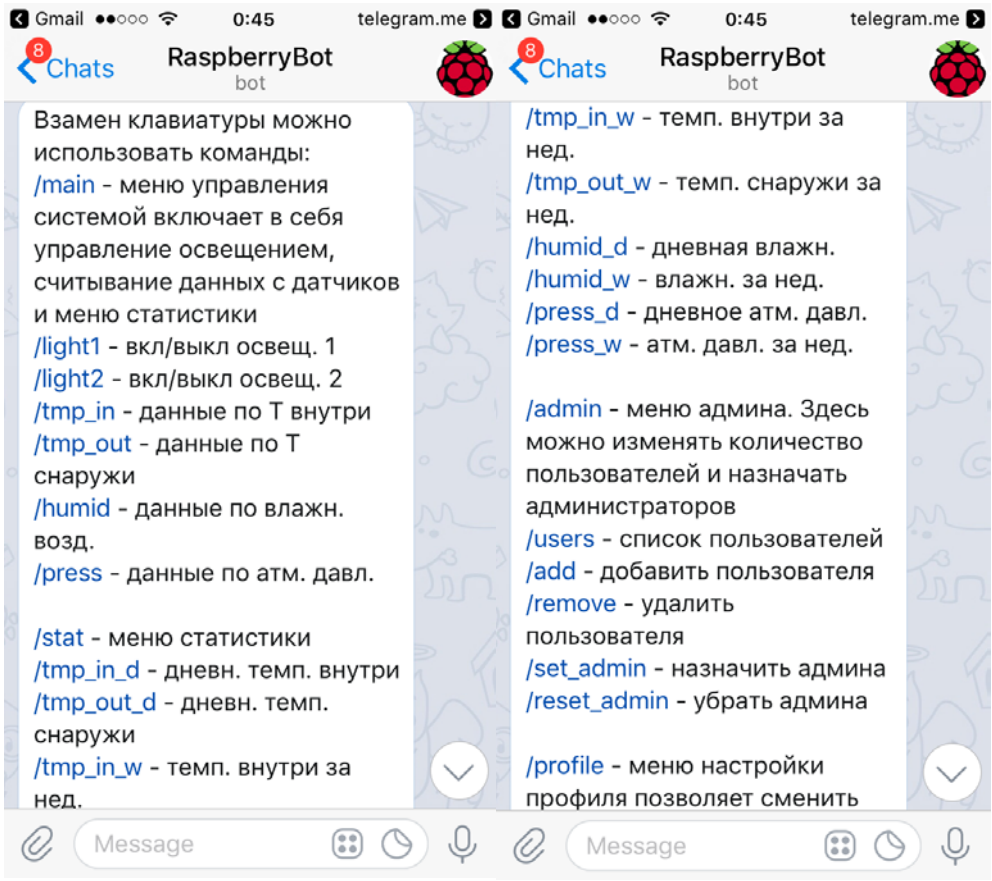
Возможности администратора расширяются посредством дополнительного меню, в котором можно просматривать весь список посетителей, добавлять или удалять пользователей, назначать или убирать администраторов. На рисунках с 69 по 75 показаны все возможности администрации.



Рисунки 69, 70 – Возможности администрации



Рисунки 71, 72 – Возможности администрации



Рисунки 73, 74 – Возможности администрации

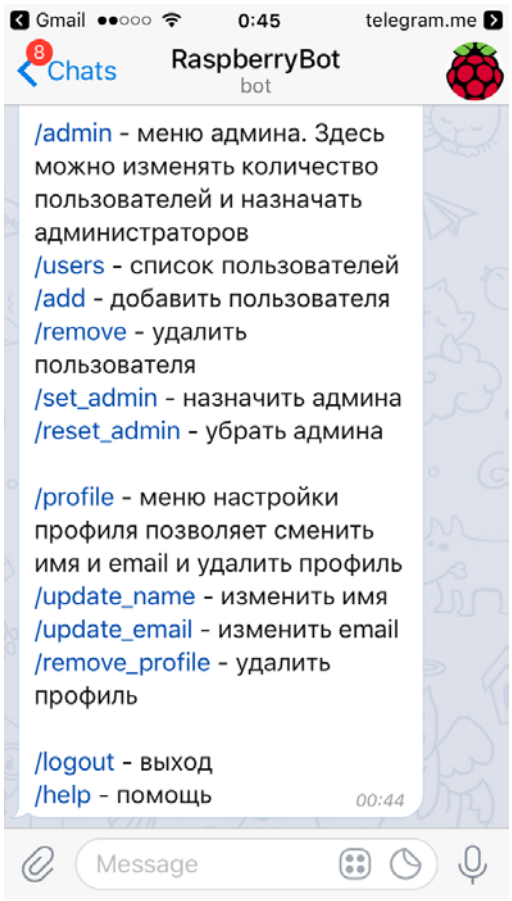


Рисунок 75 – Возможности администрации

Выход из системы показан на рисунке 76.

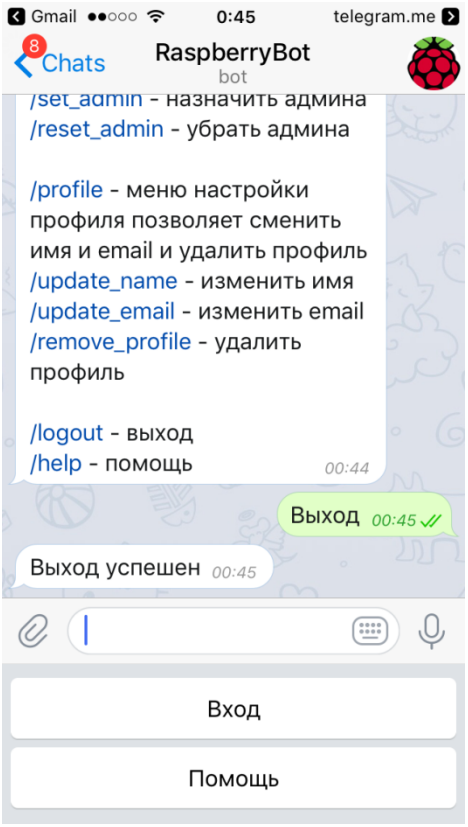


Рисунок 76 – Процесс выхода из системы

Далее рассмотрим все вышеуказанное в деталях реализации со стороны кода. Начнем с базы данных. Пользователь хранится в таблице users. Рисунок 77 это показывает. Пользовательские роли, необходимые для разделения прав, хранятся в таблице user_roles (рисунок 78).

```
CREATE TABLE users
(
    id  SERIAL PRIMARY KEY,
    name    VARCHAR NOT NULL DEFAULT 'NULL',
    email   VARCHAR NOT NULL,
    chat_id VARCHAR NOT NULL DEFAULT 'NULL',
    token   VARCHAR NOT NULL DEFAULT 'NULL'
);
CREATE UNIQUE INDEX users_unique_email_idx ON users (email);
```

Рисунок 77 – Таблица «users»

```
CREATE TABLE user_roles
(
  user_id    INTEGER NOT NULL,
  role       VARCHAR,
  CONSTRAINT user_roles_idx UNIQUE (user_id, role),
  FOREIGN KEY (user_id) REFERENCES users (id) ON DELETE CASCADE
);
```

Рисунок 78 – Таблица «user_roles»

Наполняем таблицы данными о пользователях. Для администратора хватит email-аи chat_id, который можно получить в скрипте bot.руиз беседы с ботом. Вид файла populateDB.sqlпосле изменений принимает вид как на рисунке 79.

```
DELETE FROM users;
DELETE FROM user_roles;
DELETE FROM humid_s;
DELETE FROM press_s;
DELETE FROM temps_in;
DELETE FROM temps_out;

ALTER SEQUENCE humid_s_id_seq RESTART WITH 1;
ALTER SEQUENCE press_s_id_seq RESTART WITH 1;
ALTER SEQUENCE temps_in_id_seq RESTART WITH 1;
ALTER SEQUENCE temps_out_id_seq RESTART WITH 1;
ALTER SEQUENCE users_id_seq RESTART WITH 1;

INSERT INTO users (email, chat_id) VALUES
    ('konyaevig@gmail.com', '170296575');

INSERT INTO users (email) VALUES
    ('konyaevig147@gmail.com');

INSERT INTO user_roles (user_id, role) VALUES
    (1, 'ROLE_ADMIN'),
    (2, 'ROLE_USER');
```

Рисунок 79 – Скрипт populateDB.sql

РассмотримDAO (Dataaccessobject) для пользователя. Сервис содержит методы для доставания пользователей из базы данных как общим списком, так и по отдельным параметрам (email, tokenи др.). На рисунках 80 и 81 показаны несколько основных методов, остальное можно найти в приложении И или в репозитории GitHub: https://github.com/ikonyaev/rpi_telebot

```
def get_all_with_roles(self):
    self.cursor.execute("SELECT * FROM users INNER JOIN user_roles"
        + " ON (users.id = user_roles.user_id)")
    rows = self.cursor.fetchall()

    log.info("Get all users with roles")
    return rows

def get_all(self):
    self.cursor.execute("SELECT * FROM users")
    rows = self.cursor.fetchall()

    log.info("Get all users")
    return rows
```

Рисунок 80 – Часть userdao.py

```
def update(self, name, email, chat_id, token):
    self.cursor.execute("UPDATE users SET name=(%s), chat_id=(%s), token=(%s) WHERE email LIKE (%s)",
                        (name,chat_id,token,email,))

    log.info("Update user with email:" + email)

def update_name(self, chat_id, name):
    self.cursor.execute("UPDATE users SET name=(%s) WHERE chat_id LIKE (%s)",
                        (name,chat_id,))

    log.info("Update user's name to " + name)
```

Рисунок 81 – Часть userdao.py

Сервис по отправке email-а содержит всего один метод, не считая конфигурации (рисунок 82).

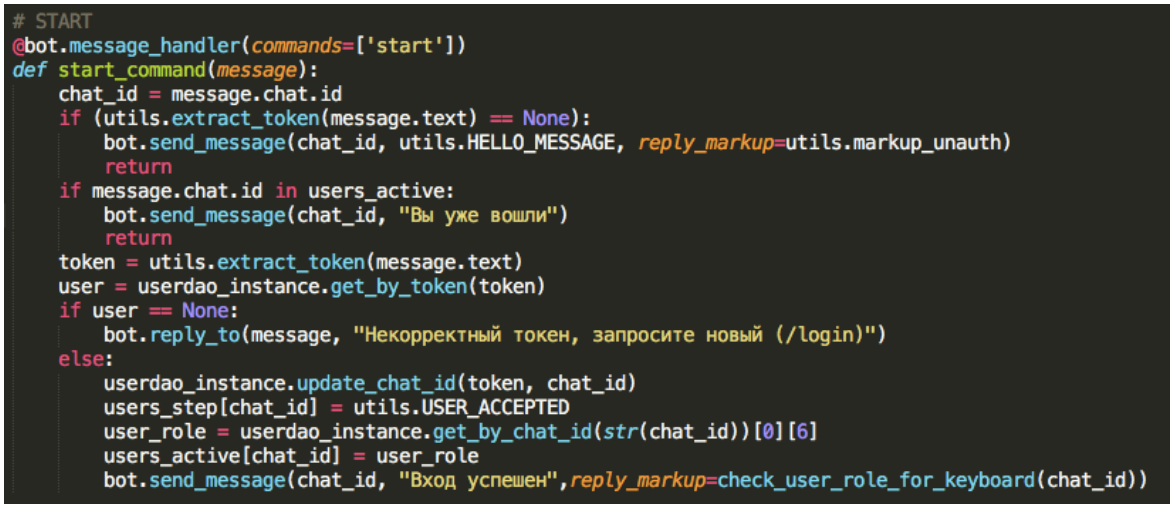
```
def send_mail(self, email, token):
    me = "rpi.telebot@gmail.com"
    you = email

    msg = MIMEMultipart('alternative')
    msg['Subject'] = "Your token"
    msg['From'] = me
    msg['To'] = you

    html = """\
    <html>
        <head>
            <meta charset="utf-8">
            <title>Telegram: Contact @raspberry_tele_bot</title>
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
        </head>

        <body>
            <div>
                <p>Your token for @raspberry_tele_bot</p>
                <p>Follow the <a href="https://telegram.me/raspberry_tele_bot?start="""+
                token +""""">link</a> for login.</p>
                <p>Don't have Telegram yet? Try it now:</p>
                <a href="telegram.org/dl">Get Telegram</a>
            </div>
        </body>
    </html>"""

    part = MIMEText(html, 'html')
    msg.attach(part)
    self.smtpObj.sendmail(me,email,msg.as_string())
    log.info("Sent successful.")
```

Рисунок 82 – Главный метод сервиса по отправке сообщений

В скрипт utils.рутеперь перенесены все инициализаторы кастомных клавиатур, добавлены сообщения для неавторизованного, обычного и привилегированного пользователей, так же теперь тут дополнительные методы и переменные состояния, они нам понадобятся немного позже.

Скрипт bot.рупретерпел значительные изменения. Для того, чтобы понимать, какой пользователь активен используется словарь users_active, хранящий данные chat_idи роли пользователя в структуре ключ-значение. Словарь, хранящий состояние пользователя users_step необходим для индикации того, что на данный момент делает пользователь (входит в систему или меняет свое имя), данные из словаря используются в нужных местах программы (хранятся chat_idи переменная состояния из utils.py).

Метод, который ловит команду /start, показан на рисунке 83.



```
# START
@bot.message_handler(commands=['start'])
def start_command(message):
    chat_id = message.chat.id
    if (utils.extract_token(message.text) == None):
        bot.send_message(chat_id, utils.HELLO_MESSAGE, reply_markup=utils.markup_unauth)
        return
    if message.chat.id in users_active:
        bot.send_message(chat_id, "Вы уже вошли")
        return
    token = utils.extract_token(message.text)
    user = userdao_instance.get_by_token(token)
    if user == None:
        bot.reply_to(message, "Некорректный токен, запросите новый (/login)")
    else:
        userdao_instance.update_chat_id(token, chat_id)
        users_step[chat_id] = utils.USER_ACCEPTED
        user_role = userdao_instance.get_by_chat_id(str(chat_id))[0][6]
        users_active[chat_id] = user_role
        bot.send_message(chat_id, "Вход успешен",reply_markup=check_user_role_for_keyboard(chat_id))
```

Рисунок 83 – Метод start_command

Метод в зависимости от того, пришла ли команда /startбез токена или с ним выводит приветствие или осуществляет вход пользователя в систему. Токен генерируется перед отправкой письма и отправляется как параметр HTTP-запроса. Тут же производятся проверки на переход по ссылке из письма уже вошедшего пользователя.

Структуру ПО можно наблюдать в приложениях В, Г, и Д. Таблица команд в приложении Ж.

К завершению раздела стоит сказать, что тестирование системы производилось на протяжении всего процесса разработки. И потому, нет необходимости создаватьраздел «Тестирование». Полные исходные коды в приложении И.

4 БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ

В данном разделе необходимо рассмотреть следующие аспекты безопасности: эргономические аспекты, чрезвычайные ситуации, информационная безопасность.

Лабораторный стенд по автоматическому управлению освещением находится в аудитории 308 шестого корпуса. Внешний вид стенда показан на рисунке 84.

Рисунок 84 – Внешний вид стенда

В состав лабораторного стенда входят: ПЛК100, 2 модема ПМ01, измеритель токовых параметров, лампы, автоматы, магнитные пускатели.

4.1 Безопасность и эргономика рабочего места

4.1.1 Эргономика

Эргономика – это научная дисциплина, комплексно изучающая человека (группу людей) в конкретных условиях его (их) трудовой деятельности, связанной с использованием машин или механизмов с целью повышения эф-

фективности функционирования таких систем путем оптимизации средств, условий и процесса труда [22].

Эргономика является одновременно и исследовательской и проектировочной дисциплиной.

Объектом исследования эргономики является система “человек – машина – среда” (СЧМ). Эргономика рассматривает СЧМ как сложное функционирующее целое, в котором ведущая роль принадлежит человеку.

Цель эргономики можно формулировать по-разному, но общий смысл останется один - согласование конструкции машин и условий их функционирования с психофизиологическими характеристиками работающего.

Чтобы добиться указанной цели, требуется решение довольно разноплановых задач эргономики:

а) нужно изучать особенности взаимодействия техники и человека в конкретных условиях;

б) необходимо совершенствовать технику и условия ее функционирования под возможности человека;

в) следует специально готовить людей для работы на машинах.

При организации рабочего места пользователя, следует обеспечить расположение всех элементов рабочего места согласно эргономическим требованиям с учетом характера выполняемой пользователем деятельности, комплексности технических средств, форм организации труда и основного рабочего положения пользователя.

Вероятность негативных последствий от использования персонального компьютера такая же, как и при эксплуатации другой бытовой техники. Пренебрежение элементарными рекомендациями имеет серьезные последствия для их владельцев. Компьютер – это такой же потенциальный источник угроз для здоровья, имущества и даже жизни пользователя.

4.1.2 Общие правила безопасности.

Техника безопасности при работе с компьютером на предприятии предусматривает наличие общедоступной инструкции, в которой указаны обяза-

тельные требования к обустройству рабочего места и процессу использования техники. Эти правила едины для всех организаций, их выполнение контролируется руководящими органами. Основные правила организации пространства вокруг рабочего места:

- при длительном и интенсивном использовании, на поверхности модулей ПК (системный блок, монитор, мышка и т.д.) возникают небольшие разряды тока. Эти частицы активизируются во время прикосновений к ним и приводят к выходу техники из строя. Нужно регулярно использовать нейтрализаторы, увлажнители воздуха, антистатики;

- вокруг стола не должно быть свисающих проводов, пользователь не должен контактировать с ними;

- важна целостность корпуса розетки и штепсельной вилки;

- помещение должно хорошо вентилироваться и охлаждаться в жаркую пору года. Важен своевременный отвод избыточного тепла от техники.

4.1.3 Требования безопасности перед началом работы.

Перед тем, как включить компьютер, необходимо уделить пару минут следующим действиям:

- Нужно убедиться в том, что в зоне досягаемости отсутствуют оголенные провода и различные шнуры. Они не только мешают работе, но и несут потенциальную опасность в случае короткого замыкания; нельзя начинать работу на технике с видимым повреждением. В случае обнаружения трещины на корпусе или повреждений другого рода, нужно обратиться за помощью в сервисный центр. Это же относится к ПК с неисправным индикатором включения/выключения. Предметы на столе не должны мешать обзору, пользованию мышкой и клавиатурой. Поверхность экрана должна быть абсолютно чистой; на системном блоке не должно находиться никаких предметов, так как в результате вибраций может нарушиться работа устройства. Нужно убедиться в том, что никакие посторонние предметы не мешают работе системе охлаждения. Недопустимо включать персональный компьютер в удлинители и розетки, в которых отсутствует заземляющая шина;

- Запрещается начинать работу в помещениях с повышенной влажностью, а также в случае, если рядом присутствуют открытые источники влажности (лужи, мокрый пол). Включить технику можно лишь после полного высыхания окружающих предметов;

- Недопустимо часто включать и выключать компьютер в течение рабочего дня без особой нужды.

4.1.4 При выполнении работы.

Поскольку персональный компьютер обладает всеми свойствами электрического прибора, то на него распространяются основные правила безопасности при взаимодействии с проводниками тока:

- Нельзя размещать какие-либо вещи на поводах, а также самостоятельно менять их расположение без особой нужды;

- Рекомендуется избегать расположения жидкостей рядом с модулями компьютера;

- Нельзя работать на ПК с мокрыми руками, нельзя очищать поверхность компьютера от загрязнений, когда он находится во включенном состоянии, недопустимо снимать корпус любой из составных частей ПК во время его работы;

- Разбор и ремонт техники имеют совершают только специализированные работники;

- Во время работы на компьютере нельзя одновременно прикасаться к другим металлическим конструкциям, которые стоят на той же поверхности;

- В помещении с компьютерами непозволительно курить или употреблять пищу непосредственно на рабочем месте;

- При ощущении даже незначительного запаха гари, нужно как можно быстрее выключить ПК из сети и обратиться к ответственному за обслуживание компьютерной техники.

Как было сказано выше, неправильная работа с персональным компьютером таит в себе множество угроз для здоровья человека. Что бы минимизировать это влияние даже при длительном нахождении за монитором, стоит

навсегда запомнить следующие постулаты: расстояние между глазами пользователя и экраном составляет не менее полуметра. Но пользователь должен быть в состоянии дотянуться кончиками пальцев до верхнего края монитора; Высота сидения позволяет держать ровную осанку; локти согнуты под прямым углом, а в кистях рук, лежащих на столе, не чувствуется напряжения; локти не висят в воздухе, а комфортно располагаются на подлокотниках кресла или столешнице. Их позиция существенно не меняется при передвижении мышки; ноги упираются в твердую поверхность, распрямлены вперед, а не подогнуты под себя; если пользователь носит очки, то нужно убедиться в том, что он может свободно регулировать угол наклона экрана. Чрезвычайно важна периодическая зарядка. Каждый час нужно вставать с кресла, разминать мышцы и суставы. Ведь, несмотря на неподвижность, они испытывают огромную нагрузку, пребывая в неестественном положении. Обязательно нужно делать разминку для глаз: круговые и линейные движения открытыми глазами, моргание и расфокусирование.

4.1.5 По окончании работы.

Перед завершением нужно правильно закрыть все программы и окна. Нельзя оставлять активные носители информации (диски и флэшки). Стоит отметить, что порядок выключения составляющих частей ПК отличается от порядка их включения ровно наоборот. Запуск компьютера происходит по цепочке: общее питание – периферия – системный блок. Выключение, соответственно, начинается с системного блока. Вытягивать штепсельную вилку необходимо крепко держась за её корпус. Нельзя совершать резких рывков и тем более тянуть за провод. После завершения работы, желательно устранять лишнее статическое напряжение с поверхности электроприборов и проводить влажную уборку рабочего места [21].

4.1.6 Эргономика программного обеспечения

Так же необходимо рассмотреть эргономику интерфейса программного обеспечения.

Программное обеспечение для система передачи данных по GPRS было разработано на языке программирования Python, TelegramBotAPI, PostgreSQL и др. Python позволяет строить гибкую архитектуру ПО простыми средствами и практически без низкоуровневых конструкций. Так же существует огромный перечень библиотек для работы с базами данных, с почтой и с внешними веб-сервисами. Интерфейс по взаимодействию человека и системы обеспечивает приложение Telegram.

4.1.7 Экологичность

Работа со стендом не сопряжена с образованием и выделением газообразных, жидких или твердых отходов. Так же он не требует использования ресурсов.

Утратившее работоспособность детали и компоненты передают специальным службам (предприятиям) для сортировки, вторичного использования или складирования на городских мусорных полигонах.

4.2 Чрезвычайные ситуации

Чрезвычайная ситуация (ЧС) – состояние, при котором в результате возникновения источника чрезвычайной ситуации на объекте, определенной территории или акватории нарушаются нормальные условия жизни и деятельности людей, возникает угроза их жизни и здоровью, наносится ущерб имуществу населения, народному хозяйству и окружающей природной среде.

В данном случае в качестве ЧС рассматривается возникновение пожара, который может возникнуть из-за превышения температурой максимального заданного значения. Под пожаром обычно понимают неконтролируемый процесс горения, сопровождающийся уничтожением материальных ценностей и создающий опасность для жизни людей. Пожар может принимать различные формы, однако все они, в конечном счете, сводятся к химической реакции между горючими веществами и кислородом воздуха (или иным видом окислительных сред), возникающей при наличии инициатора горения или в условиях самовоспламенения.

Анализ пожарной опасности заключается в определении наличия горючих веществ и возможных источников зажигания, вероятных путей распространения пожара, необходимых средств технической и конструктивной защиты, а также систем сигнализации и пожаротушения, имеющих параметры инерционности срабатывания соответствующие динамике развития пожара на предприятии [20].

Противопожарные мероприятия предотвращения пожара разрабатываются исходя из требований об исключении источника зажигания и (или) горючего вещества из системы, приводящей к пожару. Если источник зажигания и горючее вещество не могут быть изолированы по условиям технологического процесса производства, объект обеспечивается надежной системой противопожарной защиты.

Мероприятия по предотвращению пожара: предотвращение образования горючей среды, предотвращение образования в горючей среде источников зажигания**,** ограничение массы и объема горючих веществ, мероприятия противопожарной защиты.

Противопожарная защита на предприятии реализуется техническими (конструктивными) и пожарно-техническими мероприятиями. В зданиях и сооружениях необходимо предусмотреть технические средства (лестничные клетки, противопожарные стены, лифты, наружные пожарные лестницы, аварийные люки и т.п.), имеющие устойчивость при пожаре и огнестойкость конструкций не менее времени, необходимого для спасения людей при пожаре и расчетного времени тушения пожара.

Каждый объект должен иметь такое объемно-планировочное и техническое исполнение, чтобы эвакуация людей из него была завершена до наступления предельно допустимых значений опасных факторов пожара, а при нецелесообразности эвакуации была обеспечена защита людей в объекте.

На каждом объекте должно быть обеспечено своевременное оповещение людей и (или) сигнализация о пожаре в его начальной стадии техническими или организационными средствами.

4.2.1 Требования к путям эвакуации.

Эвакуация людей – вынужденный процесс движения людей из зоны, где имеется возможность воздействия на них опасных факторов пожара.

Эвакуационный выход –выход, ведущий в безопасную при пожаре зону.

Путь эвакуации – безопасный при эвакуации людей путь, ведущий к эвакуационному выходу.

Так же необходимо рассмотреть противопожарные нормы.

Эвакуационные пути должны обеспечить безопасную эвакуацию всех людей, находящихся в помещениях зданий, через эвакуационные выходы. При устройстве эвакуационных выходов из двух лестничных клеток через общий вестибюль одна из лестничных клеток кроме выхода в вестибюль должна иметь выход непосредственно наружу. Выходы наружу допускается предусматривать через тамбуры. Из зданий, с каждого этажа и из помещения следует предусматривать не менее двух эвакуационных выходов.

Ширина путей эвакуации в свету должна быть не менее 1 м**,** дверей не менее 0,8 м. Высота прохода на путях эвакуации должна быть не менее 2 м. В общих коридорах не допускается предусматривать устройство встроенных шкафов, за исключением шкафов для коммуникаций и пожарных кранов. Высота дверей в свету на путях эвакуации должна быть не менее 2 м. Высота дверей и проходов, ведущих в помещения без постоянного пребывания в них людей, а также в подвальные, цокольные и технические этажи, допускается уменьшать до 1,9 м, а дверей, являющимися выходом на чердак или бесчердачное покрытие - до 1,5 м.

Наружные эвакуационные двери зданий не должны иметь запоров, которые не могут быть открыты изнутри без ключа.

Двери лестничных клеток, ведущие в общие коридоры, двери лифтовых холлов и тамбуров-шлюзов должны иметь приспособления для самозакрывания и уплотнения в притворах и не должны иметь запоров, препятствующих их открыванию без ключа.

Ширина марша лестницы должна быть не менее ширины эвакуационного выхода (двери) в лестничную клетку.

Ширина лестничных площадок должна быть не менее ширины марша, а перед входами в лифты с распашными дверями - не менее суммы ширины марша и половины ширины двери лифта, но не менее 1,6 м.

В световых проемах лестничных клеток, заполненных стеклоблоками, следует предусматривать открывающиеся фрамуги площадью не менее 1,2 $м^2$ на каждом этаже.

4.2.2 Требования к пожарной безопасности зданий и сооружений.

Выходы являются эвакуационными, если они ведут:

а) из помещений первого этажа наружу: непосредственно; через коридор; через вестибюль (фойе); через лестничную клетку; через коридор и вестибюль (фойе); через коридор и лестничную клетку;

б) из помещений любого этажа, кроме первого: непосредственно в лестничную клетку или на лестницу 3-го типа; в коридор, ведущий непосредственно в лестничную клетку или на лестницу 3-го типа; в холл (фойе), имеющий выход непосредственно в лестничную клетку или на лестницу 3-го типа;

в) в соседнее помещение (кроме помещения класса Ф5 категории А и Б) на том же этаже, обеспеченное выходами, указанными в «а» и «б»; выход в помещение категории А или Б допускается считать эвакуационным, если он ведет из технического помещения без постоянных рабочих мест, предназначенного для обслуживания вышеуказанного помещения категории А или Б.

Количество и общая ширина эвакуационных выходов из помещений, с этажей и из зданий определяются в зависимости от максимально возможного числа эвакуирующихся через них людей и предельно допустимого расстояния от наиболее удаленного места возможного пребывания людей (рабочего места) до ближайшего эвакуационного выхода.

Не менее двух эвакуационных выходов должны иметь: помещения класса Ф1.1, предназначенные для одновременного пребывания более 10 чел.; помещения подвальных и цокольных этажей, предназначенные для од-

новременного пребывания более 15 чел., в помещениях подвальных и цокольных этажей, предназначенных для одновременного пребывания от 6 до 15 чел., один из двух выходов допускается предусматривать в соответствии с требованиями; помещения, предназначенные для одновременного пребывания более 50 чел., и т.д.

При устройстве двух эвакуационных выходов каждый из них должен обеспечивать безопасную эвакуацию всех людей, находящихся в помещении, на этаже или в здании. При наличии более двух эвакуационных выходов безопасная эвакуация всех людей, находящихся в помещении, на этаже или в здании, должна быть обеспечена всеми эвакуационными выходами, кроме каждого одного из них.

Высота эвакуационных выходов в свету должна быть не менее 1,9 м, ширина не менее: 1,2 м — из помещений класса Ф1.1 при числе эвакуирующихся более 15 чел., из помещений и зданий других классов функциональной пожарной опасности, за исключением класса Ф1.3, — более 50 чел., 0,8 м — во всех остальных случаях.

4.2.3 Эвакуационные пути.

Пути эвакуации должны быть освещены. Предельно допустимое расстояние от наиболее удаленной точки помещения, а для зданий класса Ф5 — от наиболее удаленного рабочего места до ближайшего эвакуационного выхода, измеряемое по оси эвакуационного пути, должно быть ограничено в зависимости от класса функциональной пожарной опасности и категории взрывопожароопасности помещения и здания, численности эвакуируемых, геометрических параметров помещений и эвакуационных путей, класса конструктивной пожарной опасности и степени огнестойкости здания.

Высота горизонтальных участков путей эвакуации в свету должна быть не менее 2 м**,** ширина горизонтальных участков путей эвакуации и пандусов должна быть не менее: 1,2 м — для общих коридоров, по которым могут эвакуироваться из помещений класса Ф1 более 15 чел., из помещений других

классов функциональной пожарной опасности — более 50 чел., 0,7 м — для проходов к одиночным рабочим местам; 1,0 м — во всех остальных случаях.

Эвакуация по лестницам и лестничным клеткам.

Ширина марша лестницы, предназначенной для эвакуации людей, в том числе расположенной в лестничной клетке, должна быть не менее, расчетной или не менее ширины любого эвакуационного выхода (двери) на нее, но, как правило, не менее: 1,35 м - для зданий класса Ф1.1; 1,2 м - для зданий с числом людей, находящихся на любом этаже, кроме первого, более 200 чел.; 0,7 м - для лестниц, ведущих к одиночным рабочим местам; 0,9 м - для всех остальных случаев.

Уклон лестниц на путях эвакуации должен быть, как правило, не более 1:1; ширина проступи - как правило, не менее 25 см, а высота ступени - не более 22 см.

Проанализировав шестой учебный корпус, а так же аудиторию 308 с точки зрения правил пожарной безопасности, можно сделать вывод, что данное здание полностью соответствует рассмотренным санитарным нормам и правилам.

В разработанном программном обеспечении предусмотрено аварийное отключение всей системы. Если значение температуры превышает один из установленных пределов, программа должна отправить сообщение с текстом ALARM, текущей датой и временем, текущим значением температуры и отключить систему.

Таким образом, в данном разделе были рассмотрены основные аспекты безопасности работы с проектируемой системой. Указаны необходимые условия обеспечения безопасности жизни и здоровья оператора. Составлены правила техники безопасности при работе с программным обеспечением. Также проанализировано и рассмотрено влияние эргономических показателей на работу человека.

4.2.4 Требования к безопасности при ЧС в отношении ЭВМ

Своевременная бдительность поможет избежать опасных ситуаций для жизни и сохранить целостность техники. Действия в аварийных ситуациях: при неполадках любого рода в электроснабжении устройства необходимо сразу отключить компьютер от сети; если обнаружен оголенный провод, то необходимо оперативно оповестить всех работников офиса, не допуская чьего-либо контакта с ним; в каждом учреждении должны находиться огнетушители ОУБ-3 или ОУ-2, а также ведра и полотна в необходимом количестве. Персонал обязан знать о том, где находятся средства для гашения пламени и куда нужно звонить в случае пожара; при поражении человека электрическим током, прежде всего, оказывается первая помощь: искусственное дыхание и внешний интенсивный массаж сердца. В первые же мгновения после удара током, вызывается скорая помощь.

Недопустимо размещать компьютерные провода рядом с отопительной системой, их изоляция должна быть целостной. Системный блок не должен стоять в нише стола или другом замкнутом пространстве, где нарушена нормальная вентиляция. Пользователь может и должен контролировать весь цикл взаимодействия с техникой. Процесс соблюдения всех этих несложных правил должен быть непрерывным и комплексным [21].

4.3 Информационная безопасность

В век информационных технологий очень важна защита личной и конфиденциальной информации от посторонних лиц. Предприятия и фирмы имеют права на защиту служебной информации в той же мере, как и обособленная личность имеет право на защиту личной информации (переписка, звонки и др.).

Предприятия, имеющие в своей технической структуре локальную вычислительную сеть или доступ к сети Интернет, должны обеспечивать защиту информации и доступ к важным органам предприятия от атак как извне, так и изнутри.

Система по передаче данных по GPRS обладает достаточными мерами по защите информации. Весь входящий и исходящий траффик шифруется по протоколу HTTPS. Только администратор может добавить нового пользователя в систему, добавленный пользователь взаимодействует с системой на ограниченных правах. Авторизация производится по Email, все пароли и ключи хранятся только в базе данных системы и пользователю нужно помнить лишь только свой Email.

ЗАКЛЮЧЕНИЕ

При выполнениивыпускной квалификационной работы были закреплены и углублены знания, полученные в течение обучения.

Даннаявыпускная квалификационная работа разработана в полном соответствие с методическими рекомендациями, выданным преподавателем, а такжедействующим стандартом АмГУ и ГОСТами.

При выполнении выпускной квалификационной работы была разработана полная электрическая схема, подобрано оборудование, разработано программное обеспечение.

Полученная система полностью соответствует поставленному заданию, выполняет все задуманные функции и допускает дальнейшую модификацию.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1Англоязычный сайт, посвященный разработке на RaspberryPi «Rhydo-LABZ» [Электронный ресурс]. –Режим доступа:http://www.rhydolabz.com/wiki/.– 28.02.2017.

2Англоязычный официальный сайт по RaspberryPi «RaspberryPi» [Электронный ресурс]. – Режим доступа:https://www.raspberrypi.org.– 28.02.2017.

3Сайтсо статьями по RaspberryPi «Espada» [Электронный ресурс]. – Режим доступа: http://raspberry.su.– 2.04.2017.

4Англоязычный форум по Linux&Unix «stackexchange» [Электронный ресурс]. – Режим доступа:http://unix.stackexchange.com.– 15.03.2017.

5«GitHub»для поиска библиотек и прочего[Электронный ресурс]. – Режим доступа: https://github.com.– 2.04.2017.

6Англоязычный сайт по RaspberryPi «RasPi» [Электронный ресурс]. – Режим доступа: http://raspi.tv.– 2.04.2017.

7Официальный форум RaspberryPi[Электронный ресурс]. –Режим доступа: https://www.raspberrypi.org/forums/.– 1.04.2017.

8Англоязычный блог Бена Уолтерса по разработке ПО [Электронный ресурс]. – Режим доступа: http://blog.bpwalters.com. – 1.04.2017.

9Англоязычный блог «SKPang» [Электронный ресурс]. –Режим доступа: http://skpang.co.uk/blog/.– 29.03.2017.

10Руководство по синхронизации времени на Linux-машинах[Электронный ресурс]. –Режим доступа: http://mycyberuniverse.com/ru/linux/sinhronizatsiya-vremeni-ntp-2.html. – 2.04.2017.

11Руководства по TelegramAPI[Электронный ресурс]. –Режим доступа: https://groosha.gitbooks.io/telegram-bot-lessons/. – 2.04.2017.

12Англоязычный форум по для программистов «stackoverflow» [Электронный ресурс]. –Режим доступа: http://stackoverflow.com.– 2.04.2017.

13Руководство по MatplotlibдляPython[Электронный ресурс]. –Режим доступа: https://pythonworld.ru/novosti-mira-python/scientific-graphics-in-python.html.– 2.04.2017.

14Сайт о Python [Электронный ресурс]. – Режим доступа: https://pythonworld.ru.– 2.04.2017.

15 Англоязычный официальный сайт по Python[Электронный ресурс]. – Режим доступа: https://www.python.org.– 2.04.2017.

16Руководство по TelegramBotAPI[Электронный ресурс]. –Режим доступа: https://core.telegram.org/bots/api.– 2.04.2017.

17 Сайт «Википедия»[Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/.– 1.04.2017.

18Сайт «Хабрахабр» [Электронный ресурс]. –Режим доступа: https://habrahabr.ru/interesting/.– 1.04.2017.

19 Государственный образовательный стандарт высшего профессионального образования. Направление подготовки дипломированного специалиста 220300 - Автоматизированные технологии и производства – М.: Высшая школа, 2005. – 35 с.

20 Собурь С.В. Пожарная безопасность предприятия. Курс пожарно-технического минимума: Справочник. – 4-е изд., доп. – М.: Спецтехника, 2000.– 448 с.

21 Сайт «ПроРемонтПК» [Электронный ресурс]. – Режим доступа: http://proremontpk.ru/ustanovka/tehnika-bezopasnosti-pri-rabote-s-personalnym-kompjuterom.html. - 3.06.2017.

22 Зинченко В. П. Основы эргономики /В. П. Зинченко, В. М. Мунипов. –М.: Изд-во Моск. ун-та, 1979. – 344с.

ПРИЛОЖЕНИЕ А

Техническое задание на разработку

Техническое задание разработано с требованиями ГОСТ 19.201–78.

1 ОБЩИЕ СВЕДЕНИЯ

1.) Настоящее ТЗ распространяется на разработку системы диспетчеризации на основе технологии GSM.

2.) Заказчик: ФГБОУ ВПО Амурский государственный университет (АмГУ)

Исполнитель: Коняев И.А.

3.) Система разрабатывается на основании следующих документов:

- ФГОС направления подготовки бакалавров 15.03.04 АТПиП

- Учебный план направления подготовки бакалавров 15.03.04 Автоматизации технологических процессов и производств

4.) Плановый срок начала работ по созданию системы диспетчеризации на основе GPRS- технологии. 14 октября 2016 года.

Плановый срок окончания работ по созданию системы диспетчеризации 4 апреля 2017 года.

2 НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

*2.1*Система диспетчеризации на основе GSM-технологии предназначена для мониторинга параметров автоматизированной системы.

2.2 Цели создания системы.

- упростить удаленную диспетчеризацию;

- удаленное управление с компьютера или смартфона;

- отображение показателей с датчиков на смартфоне или на ПК;

- пересылка данных на управляющую аппаратуру;

Продолжение приложения А

3 ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ

Необходимо разработать систему беспроводного мониторинга удаленного объектасредствами мобильного интернета. Вид объекта не имеет значения, это может быть объект малого (крупного) предприятия или объект частного характера. Главным требованием к системе является удаленный доступ из любого места, где есть мобильная связь.

4 ТРЕБОВАНИЯ К СИСТЕМЕ

4.1 Требования к системе в целом

Система управления должна включать следующие элементы:

- датчики дискретные
- сервер и ПЛК
- GSM-модем

Датчики должны обеспечивать возможность получения физических показателей.

Модем должны обеспечивать беспроводную связь для отправки или получения данных из сети интернет.

Веб-сервер необходим для обработки поступающей к нему информации с ПЛК, размещения ее в базе данных, хранения служебных компонентов (логика доступа к БД, классы доступа к датчикам и к удаленным серверам, содержащие Python-код формирования запросов и т.д.).

Блок контроллера, он же сервер, предназначен для сбора и обработки сигналов датчиков и выработки сигналов управления в соответствии с программой управления.

4.1.1. Требования к структуре и функционированию системы

1) требования к способам и средствам связи для информационного обмена между компонентами системы;

2) требования к режимам функционирования системы;

В данном курсовом проекте не предусмотрены режимы работы.

3) требования по диагностированию системы;

АС должна предоставлять инструменты диагностирования основных процессов системы мониторинга процесса выполнения программы. Компоненты должны предоставлять удобный интерфейс для возможности просмотра диагностических событий, мониторинга процесса выполнения программ.

При возникновении аварийных ситуаций, либо ошибок в программном обеспечении, диагностические инструменты должны позволять сохранять полный набор информации, необходимой разработчику для идентификации проблемы (снимки экранов, текущее состояние памяти, файловой системы).

4) перспективы развития, модернизации системы.

АС должна реализовывать возможность дальнейшей модернизации как программного обеспечения, так комплекса технических средств.

4.1.2 Требования к численности и квалификации персонала системы

Для плановой диагностики АС требуется один человек.

4.1.3 Требования кнадежности

Система должна сохранять работоспособность и обеспечивать восстановление своих функций при возникновении следующих внештатных ситуаций:

– при сбоях в системе электроснабжения аппаратной части;

– при ошибках в работе аппаратных средств;

– при ошибках, связанных с программным обеспечением.

4.1.4 Требования к безопасности

Система электропитания должна обеспечивать защитное отключение при перегрузках и коротких замыканиях в цепях нагрузки, а также аварийное ручное отключение.

Общие требования пожарной безопасности должны соответствовать нормам на бытовое электрооборудование. В случае возгорания не должно выделяться ядовитых газов и дымов. После снятия электропитания должно быть допустимо применение любых средств пожаротушения.

Факторы, оказывающие вредные воздействия на здоровье со стороны всех элементов системы (в том числе инфракрасное, ультрафиолетовое, рентгеновское и электромагнитное излучения, вибрация, шум, электростатические поля, ультразвук строчной частоты и т.д.), не должны превышать действующих норм (СанПиН 2.2.2./2.4.1340-03 от 03.06.2003 г.).

4.1.5 Требования к эргономике и технической эстетике

Взаимодействие пользователей с прикладным программным обеспечением, входящим в состав системы должно осуществляться посредством визуального графического интерфейса. Интерфейс системы должен быть понятными удобным, не должен быть перегружен графическими элементами и должен обеспечивать быстрое отображение экранных форм. Навигационные элементы должны быть выполнены в удобной для пользователя форме. Средства редактирования информации должны удовлетворять принятым соглашениям в части использования функциональных клавиш, режимов работы, поиска, использования оконной системы. Ввод-вывод данных системы, прием управляющих команд и отображение результатов их исполнения должны выполняться в интерактивном режиме. Интерфейс должен соответствовать современным эргономическим требованиям и обеспечивать удобный доступ к основным функциям и операциям системы.

Все надписи экранных форм, а также сообщения, выдаваемые пользователю должны быть на русском языке.

Система должна обеспечивать корректную обработку аварийных ситуаций, вызванных неверными действиями пользователей, неверным форматом или недопустимыми значениями входных данных. В указанных случаях система должна выдавать пользователю соответствующие сообщения.

4.1.6 Требования к транспортабельности для подвижных АС

АС в сложенном виде должна быть компактной, а также должна иметь возможность быстрой и простой сборки.

4.1.7 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы

Для нормальной эксплуатации разрабатываемой системы должно быть обеспечено бесперебойное питание. При эксплуатации система должна быть обеспечена соответствующая стандартам хранения и эксплуатации.

Периодическое техническое обслуживание используемых технических средств должно проводиться в соответствии с требованиями технической документации изготовителей, но не реже одного раза в год.

В процессе проведения периодического технического обслуживания должны проводиться внешний и внутренний осмотр и чистка технических средств, проверка контактных соединений, проверка параметров настроек работоспособности технических средств и тестирование их взаимодействия.

На основании результатов тестирования технических средств должны проводиться анализ причин возникновения обнаруженных дефектов и приниматься меры по их ликвидации.

Восстановление работоспособности технических средств должно проводиться в соответствии с инструкциями разработчика и поставщика технических средств и документами по восстановлению работоспособности технических средств и завершаться проведением их тестирования. Размещение оборудования, технических средств должно соответствовать требованиям техники безопасности, санитарным нормам и требованиям пожарной безопасности.

Все пользователи системы должны соблюдать правила эксплуатации электронной вычислительной техники.

4.1.8 Требования по сохранности информации при авариях

Программное обеспечение АС должно восстанавливать свое функционирование при корректном перезапуске аппаратных средств. Приведенные

выше требования не распространяются на компоненты системы, разработанные третьими сторонами и действительны только при соблюдении правил эксплуатации этих компонентов.

4.1.9 Требования к защите от влияния внешних воздействий

Защита от влияния внешних воздействий должна обеспечиваться средствами программно - технического комплекса.

4.1.10 Требования к патентной чистоте

Установка системы в целом, как и установка отдельных частей системы не должна предъявлять дополнительных требований к покупке лицензий на программное обеспечение сторонних производителей, кроме программного обеспечения, указанного в разделе 4.3.4.

4.1.11 Дополнительные требования

Дополнительные требования не предъявляются.

4.2 Требования к видам обеспечения

4.2.1 Требования к математическому обеспечению системы

Математические методы и алгоритмы, а также программное обеспечение, используемые при разработке АС должны быть максимально оптимизированными и понятными для разработчиков.

4.2.2 Требования к информационному обеспечению системы

Состав, структура и способы организации данных в системе должны быть определены на этапе технического проектирования.

Технические средства, обеспечивающие хранение информации, должны использовать современные технологии, позволяющие обеспечить повышенную надежность хранения данных и оперативную замену оборудования.

4.2.3 Требования к лингвистическому обеспечению системы

Все прикладное программное обеспечение системы для организации взаимодействия с пользователем должно использовать русский язык.

4.2.4 Требования к программному обеспечению системы

Продолжение приложения А

При проектировании и разработке системы необходимо максимально эффективным образом использовать программное обеспечение.

4.2.5 Требования к техническому обеспечению

Техническое обеспечение системы должно максимально и наиболее эффективным образом использовать существующие технические средства.

4.2.6.Требования к организационному обеспечению

Организационное обеспечение системы должно быть достаточным для эффективного выполнения возложенных на него обязанностей при осуществлении автоматизированных и связанных с ними неавтоматизированных функций системы.

5 СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ СИСТЕМЫ

Перечень документов, по ГОСТ 34.201-89, предъявляемых по окончании соответствующих стадий и этапов работ:

Этап	Содержание работ	Результаты работ
1	Разработка технического обеспечения	Создание чертежа принципиальной электрической схемы
2	Разработка ПО	Описание алгоритма, программного обеспечения, составление перечня входных сигналов и данных

6 ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ СИСТЕМЫ

6.1 Виды, состав, объем и методы испытаний системы

Виды, состав, объем, и методы испытаний системы должны быть изложены в программе и методике испытаний АС, разрабатываемой в составе рабочей документации.

6.2 Общие требования к приемке работ по стадиям

Все создаваемые в рамках настоящей работы программные изделия передаются заказчику, как в виде готовых модулей, так и в виде исходных кодов, представляемых в электронной форме на стандартном машинном носителе.

6.3 Статус приемочной комиссии

Статус приемочной комиссии определяется заказчиком до проведения испытаний.

7 ТРЕБОВАНИЯ К СОСТАВУ И СОДЕРЖАНИЮ РАБОТ ПО ПОДГОТОВКЕ ОБЪЕКТА АВТОМАТИЗАЦИИ К ВВОДУ СИСТЕМЫ В ДЕЙСТВИЕ

В ходе выполнения проекта на объекте автоматизации требуется выполнить работы по подготовке к вводу системы в действие. При подготовке к вводу в эксплуатацию АС заказчик должен обеспечить выполнение следующих работ:

- Обеспечить соответствие помещений и рабочих мест пользователей системы в соответствии с требованиями;

- Обеспечить выполнение требований, предъявляемых к программно-техническим средствам, на которых должно быть развернуто программное обеспечение АС;

- Совместно с исполнителем подготовить план развертывания системы на технических средствах заказчика;

- Провести опытную эксплуатацию АС.

Требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие, включая перечень основных мero-

приятий и их исполнителей должны быть уточнены на стадии подготовки рабочей документации и по результатам опытной эксплуатации.

8 ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ

Техническая часть:

Продолжение приложения А

3.) Схема электрическая принципиальная

Программная часть:

1.) Перечень входных сигналов и данных

3.) Описание программного обеспечения

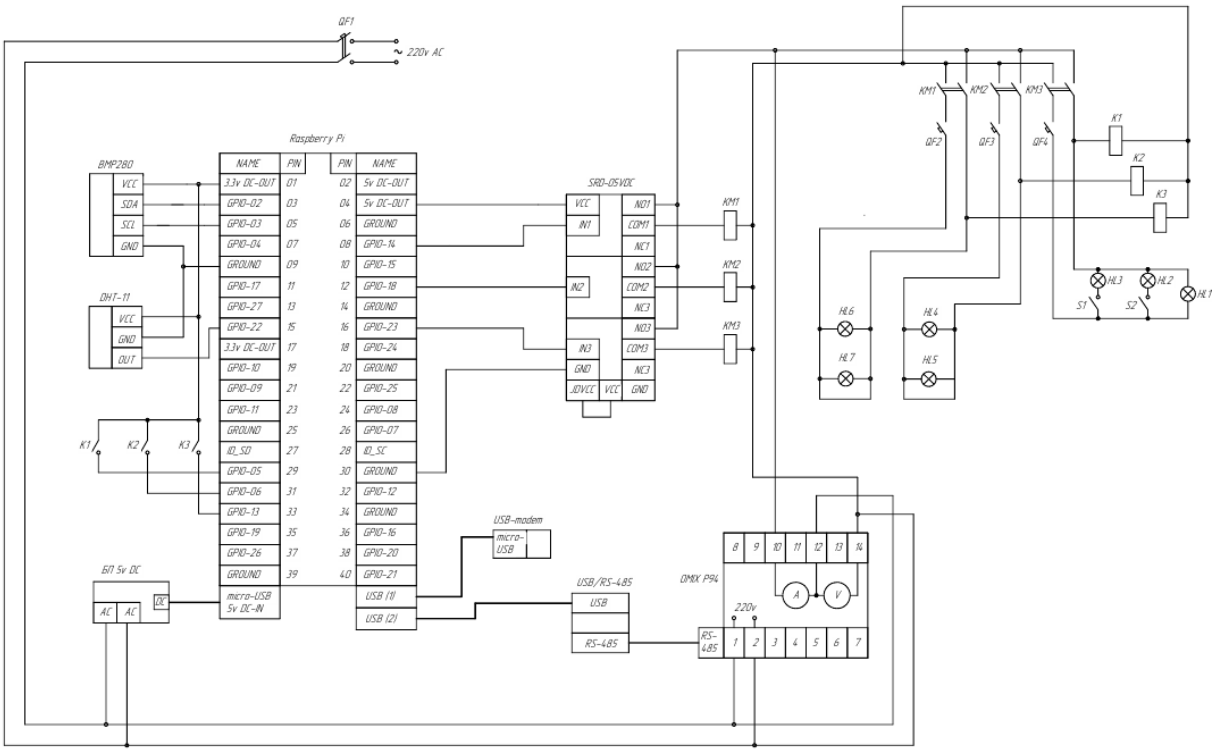
9 ИСТОЧНИКИ РАЗРАБОТКИ

Учебники, учебные пособия, и другие материалы:

- Официальная документация по RaspberryPi, форумы, учебники;

ПРИЛОЖЕНИЕ Б

Детальная электрическая схема устройства со спецификацией



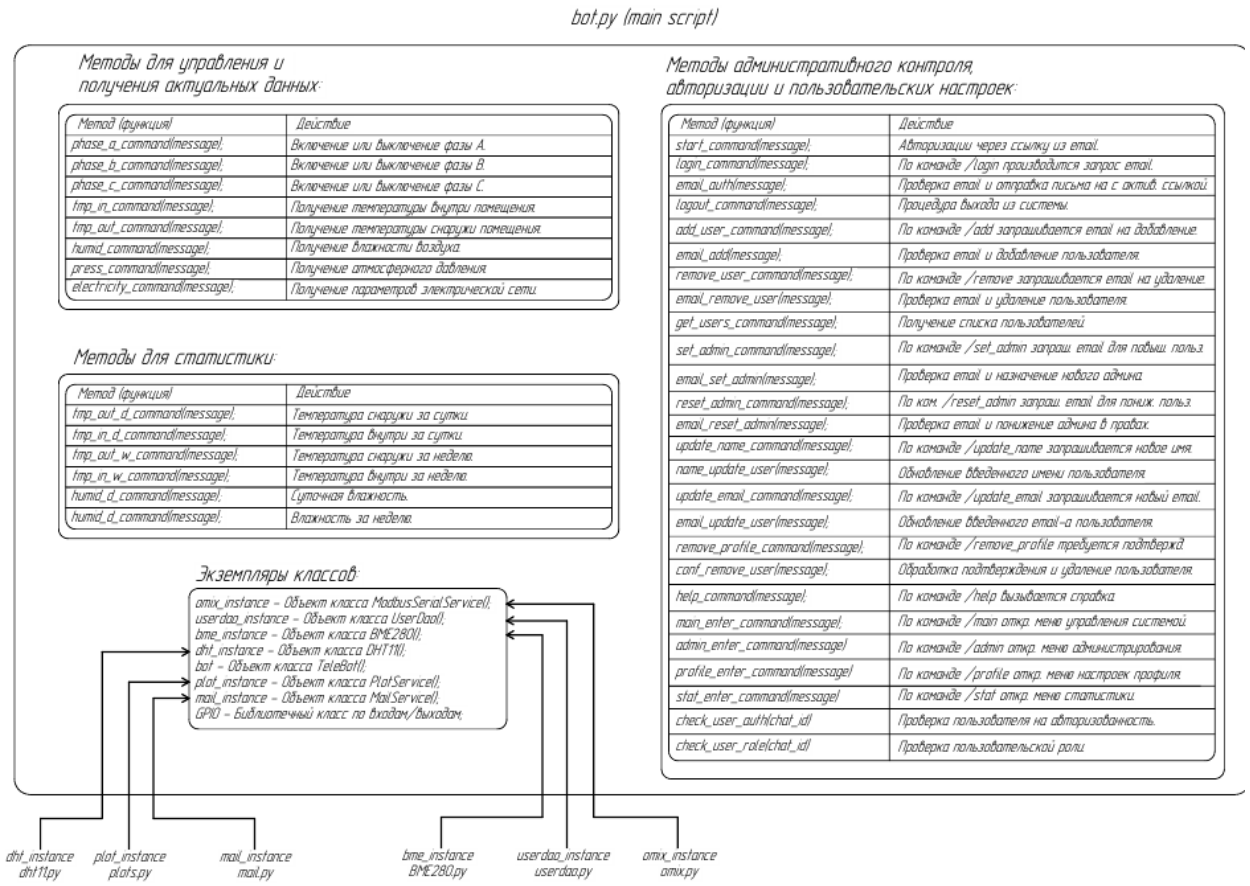
Продолжение приложения Б

Наименование		*Обозначение на сх.*
Ср-ва коммутации и защиты		
Модуль реле 3-канальный	*1*	*SRD-05VDC*
Выключатель автоматический	*4*	*QF1-QF4*
Выключатель кнопочный	*2*	*S1, S2*
Контактор магнитный	*3*	*КМ1-КМ3*
Реле сигнальное	*3*	*К1-К3*
Схемы интегральные цифровые		
Raspberry Pi 2 Model B	*1*	*Raspberry Pi*
Измеритель токовых параметров	*1*	*OMIX P94*
Сенсоры цифровые		
Сенсор давления и температуры BMP-280	*1*	*BMP-280*
Сенсор влажности и температуры DHT11	*1*	*DHT11*
Лампы сигнальные и осветительные	*7*	*HL1-HL7*
Преобразователь интерфейсов	*1*	*USB/RS-485*
Micro-USB блок питания - 5В	*1*	*БП 5v DC*
4G USB-модем	*1*	*USB-modem*

Изм.	*Лист*	*№ докум.*	*Подп.*	*Дата*	*ВКР.134008.15.03.04*		
Разраб.		*Коняев И.А.*					
Пров.		*Рыбалев А.Н.*			*Устройство цифровое*	*Лит*	*Лист* *1* *Листов*
Т.контр.		*Рыбалев А.Н.*				*У*	
Н.контр.		*Скрипко О.В.*				*АмГУ гр. 341-об*	
Утв.		*Остапенко А.А.*					

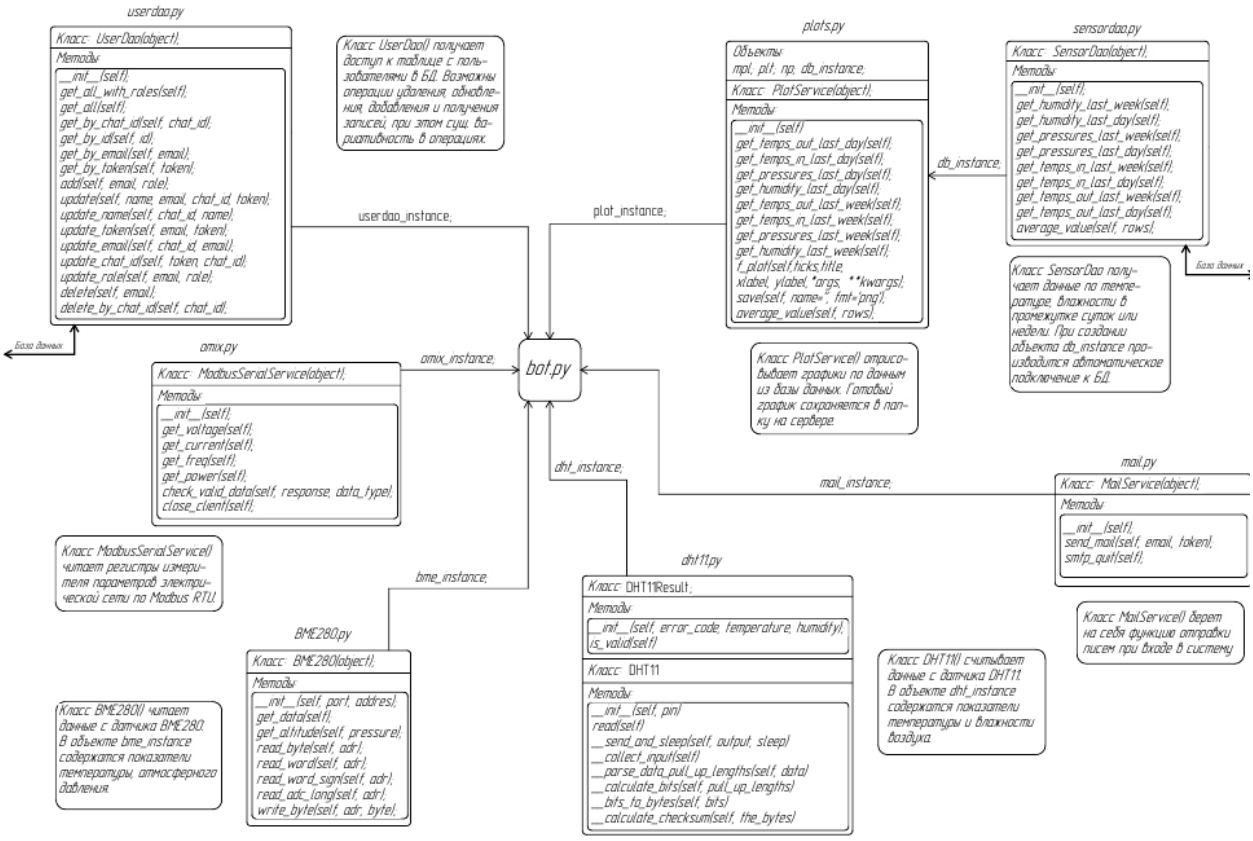
ПРИЛОЖЕНИЕ В

Структура главного программного модуля



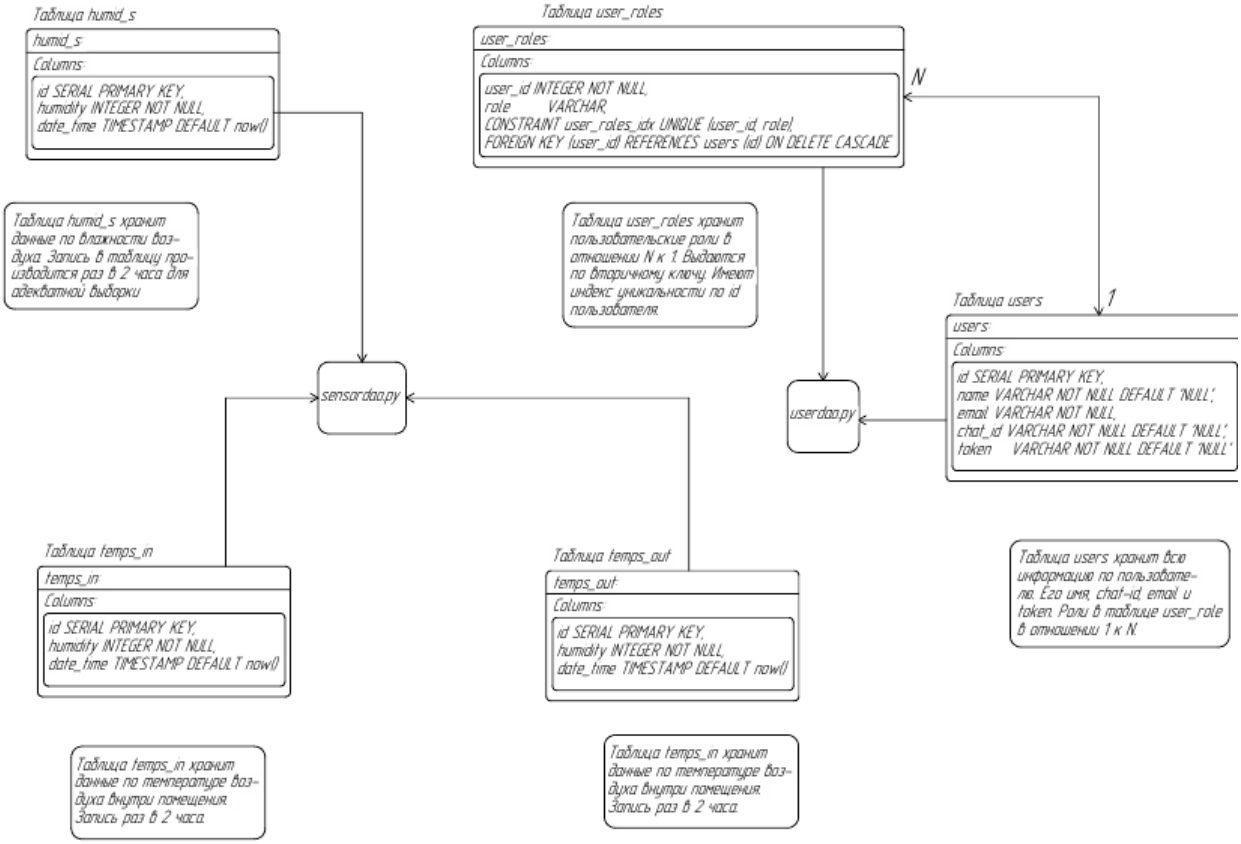
Приложение Г

Общая структура программного обеспечения



Приложение Д

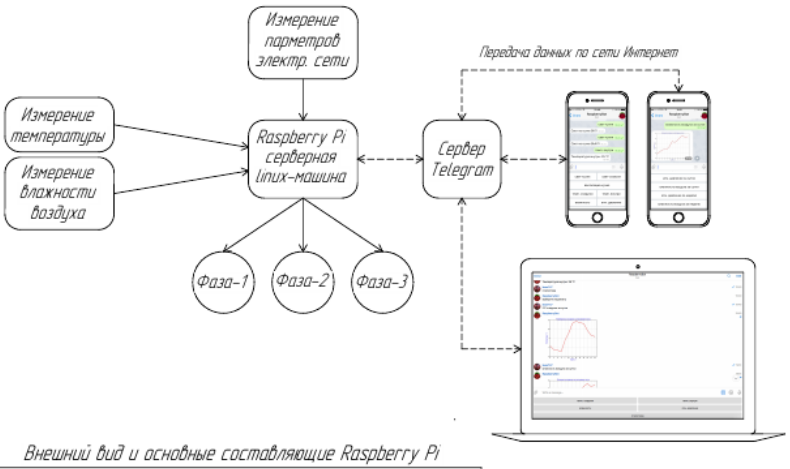
Структура базы данных



Приложение Е

Структура системы и описание платформы

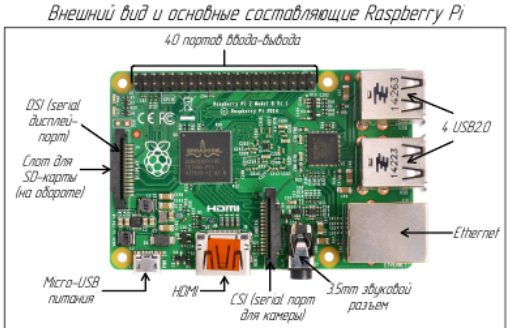
Общая структурная схема системы

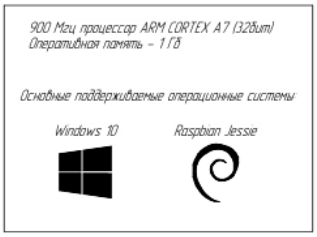


Наименование выводов Raspberry Pi

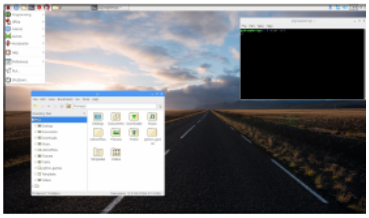
Пин №	Назначение		Назначение	Пин №
01	3.3v DC Power		DC Power 5v	02
03	GPIO-02 (SDA.1 I2C)		DC Power 5v	04
05	GPIO-03 (SCL.1 I2C)		GROUND	06
07	GPIO-04 (GPIO_GCLK)		(TXD0) GPIO-14	08
09	GROUND		(RXD0) GPIO-15	10
11	GPIO-17 (GPIO_GEN0)		(GPIO_GEN1) GPIO-18	12
13	GPIO-27 (GPIO_GEN2)		GROUND	14
15	GPIO-22 (GPIO_GEN3)		(GPIO_GEN4) GPIO-23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO-24	18
19	GPIO-10 (SPI_MOSI)		GROUND	20
21	GPIO-09 (SPI_MISO)		(GPIO_GEN6) GPIO-25	22
23	GPIO-11 (SPI_CLK)		(SPI_CE0_N) GPIO-08	24
25	GROUND		(SPI_CE1_N) GPIO-07	26
27	ID_SD (I2C ID EEPROM)		(I2C ID EEPROM) ID_SC	28
29	GPIO-05		GROUND	30
31	GPIO-06		GPIO-12	32
33	GPIO-13		GROUND	34
35	GPIO-19		GPIO-16	36
37	GPIO-26		GPIO-20	38
39	GROUND		GPIO-21	40

Внешний вид и основные составляющие Raspberry Pi



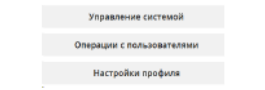
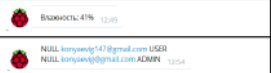
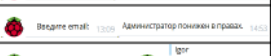


Демонстрация рабочего стола Raspbian



Приложение Ж

Основные команды запросов к серверу

Запрос клиента	Ответ сервера	Пояснение
"/start"		Команда приветствия и авторизации. Авторизация производится по ссылке из email: telegramme/raspberry_tele_bot?start=token
"/login" – "Вход"		По команде запрашивается email. После проверки email-а в БД на него отправляется письмо с ссылкой для входа в систему.
"/help" – "Помощь"		Команда вызывает справку. Ее содержание отличается в зависимости от пользовательских прав.
"/main" – "Управление системой" "/admin" – "Операции с пользователями" "/profile" – "Настройки профиля"		Команда "/main" открывает доступ к управлению системой (обновляется клавиатура); "/admin" – позволяет войти в меню администратора; "/profile" – раздел пользовательских настроек.
"/phase_a" – "Фаза A" "/phase_b" – "Фаза B" "/phase_c" – "Фаза C"		Включение и выключение фаз A, B, C.
"/electr" – "Параметры электр. сети"		Получение параметров электрической сети.
"/tmp_in" – "Температура внутри" "/tmp_out" – "Температура снаружи"		Получение условных показателей температуры снаружи и внутри помещения.
"/humid" – "Влажность"		Получение влажности воздуха.
"/users" – "Список пользователей"		Команда выводит список пользователей в виде "Имя" "email" "роль". Административский доступ.
"/add" – "Добавить пользователя"		Добавление нового пользователя в базу. Административский доступ.
"/remove" – "Удалить пользователя"		Удаление пользователя из базы. Административский доступ.
"/set_admin" – "Назначить админа"		Назначение нового администратора по email.
"/reset_admin" – "Убрать админа"		Понижение администратора по email.
"/update_name" – "Изменить имя"		Изменение имени пользователя.
"/update_email" – "Изменить email"		Изменение email-а пользователя.
"/remove_profile" – "Удалить профиль"		Полное удаление профиля с подтверждением.

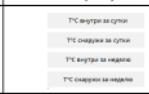
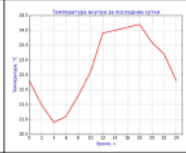
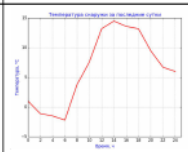
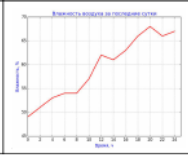
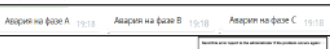
Запрос клиента	Ответ сервера	Пояснение
"/stat" – "Статистика"		Переход в подменю статистики.
"/tmp_in_d" – "Т°С внутри за сутки" "/tmp_in_w" – "Т°С внутри за неделю"		Команды предназначены для получения температурных графиков за неделю и за сутки снаружи помещения
"/tmp_out_d" – "Т°С внутри за сутки" "/tmp_out_w" – "Т°С внутри за неделю"		Команды предназначены для получения температурных графиков за неделю и за сутки снаружи помещения
"/humid_d" – "Влажность воздуха за сутки" "/humid_w" – "Влажность воздуха за неделю"		Команды предназначены для получения графиков влажности воздуха за неделю и за сутки

Табл. информирования при возникновении неисправностей

Тип неисправности	Информирование клиента
Авария на фазе A, B или C	
При программной ошибке, связанной с запросом пользователя, отправляется PDF-док. со стеком вызовов	

ПРИЛОЖЕНИЕ И

Листинг кода программ

___// Скрипт bot.py___

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO
import utils
import telebot
import dht11
import time
import os
import binascii
import re
import traceback

from telebot import types
from fpdf import FPDF
from plots import *
from userdao import *
from mail import *
from BME280 import *
from omix import *
```

```
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
GPIO.setup(17, GPIO.OUT)
GPIO.setup(27, GPIO.OUT)

omix_instance = ModbusSerialService()
bme_instance = BME280()
userdao_instance = UserDao()
dht_instance = dht11.DHT11(pin=22)
bot = telebot.TeleBot(utils.token)

GPIO.output(27,True)
time.sleep(1)
GPIO.output(27,False)

EMAIL_REGEX = re.compile(r"[^@]+@[^@]+\.[^@]+")

# FILL DICT FROM DAO {chat_id : role}
users_chatid_roles = utils.fill_users_with_roles()
users_step = {}
users_active = {}
```

```
# EXCEPTIONS DECORATOR
def handleError(function):
    def wrapped(message):
        try:
            return function(message)
        except:
            send_traceback(traceback.format_exc(), message)
    return wrapped

# START
@bot.message_handler(commands=['start'])
@handleError
def start_command(message):
            chat_id = message.chat.id
            if (utils.extract_token(message.text) == None):
                        bot.send_message(chat_id, utils.HELLO_MESSAGE, reply_markup=utils.markup_unauth)
                        return
            if message.chat.id in users_active:
                        bot.send_message(chat_id, "Выужевошли")
                        return
            token = utils.extract_token(message.text)
            user = userdao_instance.get_by_token(token)
```

```
        ifuser == None:
                bot.reply_to(message, "Некорректныйтокен, запроситеновый (/login)")
        else:
                userdao_instance.update_chat_id(token, chat_id)
                user_role = userdao_instance.get_by_chat_id(str(chat_id))[0][6]
                users_chatid_roles[str(chat_id)] = user_role
                users_active[chat_id] = user_role
                bot.send_message(chat_id, "Входуспешен",reply_markup=check_user_role_for_keyboard(chat_id))


# LOGIN
@bot.message_handler(regexp=u"Вход")
def login_keyboard_handler(message):
        login_command(message)


@bot.message_handler(commands=['login'])
@handleError
def login_command(message):
        chat_id = message.chat.id
        if chat_id in users_active:
                bot.send_message(chat_id, "Выужевошли")
        else:
                users_step[chat_id] = utils.USER_LOGGING
```

```
                bot.reply_to(message, "Введитевашemail:")


# EMAIL AUTH
@bot.message_handler(func=lambda message: users_step.get(message.chat.id) == utils.USER_LOGGING)
@handleError
def email_auth(message):
        user = userdao_instance.get_by_email(message.text)
        if user == None:
                users_step.pop(message.chat.id)
                bot.reply_to(message, "Email ненайден, повторитевход.")
        else:
                token = binascii.hexlify(os.urandom(16))
                userdao_instance.update_token(message.text, token)

                mail_instance = MailService()
                mail_instance.send_mail(message.text, token)
                mail_instance.smtp_quit()
                users_step[message.chat.id] = utils.USER_ACCEPTED
                bot.reply_to(message, "Email корректен. Пожалуйста, проверьте свой почтовый ящик.")


# LOGOUT
@bot.message_handler(regexp=u"Выход")
```

```
def logout_keyboard_handler(message):
	logout_command(message)

@bot.message_handler(commands=['logout'])
@handleError
def logout_command(message):
	chat_id = message.chat.id
	if chat_id in users_active:
		users_step.pop(chat_id)
		users_active.pop(chat_id)
		bot.send_message(chat_id, "Выходуспешен", reply_markup=utils.markup_unauth)
	else:
		bot.send_message(chat_id, "Вынеавторизованы")

# ADMIN OPS
# ADD USER
@bot.message_handler(regexp=u"Добавить пользователя")
def add_user_keyboard_handler(message):
	add_user_command(message)

@bot.message_handler(commands=['add'])
@handleError
```

```
def add_user_command(message):
        chat_id = message.chat.id
        if not resolve_admin_role(chat_id):
                return

        users_step[chat_id] = utils.ADMIN_ADD_USER
        bot.send_message(chat_id, "Введите email:", reply_markup=utils.markup_admin_ops)

@bot.message_handler(func=lambda message: users_step.get(message.chat.id) == utils.ADMIN_ADD_USER)
@handleError
def email_add(message):
        if check_email_match_regex(message):
                return
        if check_double_email(message):
                return

        userdao_instance.add(message.text, 'ROLE_USER')
        token = binascii.hexlify(os.urandom(16))
        userdao_instance.update_token(message.text, token)
        mail_instance = MailService()
        mail_instance.send_mail(message.text, token)
        mail_instance.smtp_quit()
```

```
        users_step[message.chat.id] = utils.USER_ACCEPTED

        user = userdao_instance.get_by_email(message.text)
        added_chat_id = user[0][3]
        users_chatid_roles[added_chat_id] = 'ROLE_USER'

        bot.reply_to(message, "Пользователь добавлен.")

# REMOVE USER
@bot.message_handler(regexp=u"Удалить пользователя")
def remove_user_keyboard_handler(message):
        remove_user_command(message)

@bot.message_handler(commands=['remove'])
@handleError
def remove_user_command(message):
        chat_id = message.chat.id
        if not resolve_admin_role(chat_id):
                return

        users_step[chat_id] = utils.ADMIN_REMOVE_USER
        bot.send_message(chat_id, "Введите email:", reply_markup=utils.markup_admin_ops)
```

```
@bot.message_handler(func=lambda message: users_step.get(message.chat.id) == utils.ADMIN_REMOVE_USER)
@handleError
def email_remove_user(message):
        user = userdao_instance.get_by_email(message.text)
        if check_email_match_regex(message):
                return

        userdao_instance.delete(message.text)
        users_step[message.chat.id] = utils.USER_ACCEPTED
        deleted_chat_id = user[0][3]

        if deleted_chat_id != 'NULL':
                if deleted_chat_id in users_chatid_roles:
                        users_chatid_roles.pop(deleted_chat_id)
                if int(deleted_chat_id) in users_active:
                        users_active.pop(int(deleted_chat_id))
                if int(deleted_chat_id) in users_step:
                        users_step.pop(int(deleted_chat_id))

        bot.reply_to(message, "Пользовательудален.")
        if deleted_chat_id != 'NULL':
                bot.send_message(deleted_chat_id, "Вашпрофильудален, вынеможетевойтивсистему", reply_markup=utils.markup_unauth)
```

```
# GET USERS LIST
@bot.message_handler(regexp=u"Список пользователей")
def get_users_keyboard_handler(message):
	get_users_command(message)

@bot.message_handler(commands=['users'])
@handleError
def get_users_command(message):
	chat_id = message.chat.id
	if not resolve_admin_role(chat_id):
		return

	users = userdao_instance.get_all_with_roles()
	list_users = ""
	for x in xrange(0,len(users)):
		name = users[x][1]
		email = users[x][2]
		role = users[x][6]
		list_users += name + " " + email + " " + role.split("_")[1] + "\n"

	bot.send_message(chat_id, list_users, reply_markup=utils.markup_admin_ops)
```

```
# UPDATE USER ROLE (ADMIN OR USER)
# SET
@bot.message_handler(regexp=u"Назначить админа")
def set_admin_keyboard_handler(message):
	set_admin_command(message)

@bot.message_handler(commands=['set_admin'])
@handleError
def set_admin_command(message):
	chat_id = message.chat.id
	if not resolve_admin_role(chat_id):
		return

	users_step[chat_id] = utils.ADMIN_SET_ADMIN
	bot.send_message(chat_id, "Введите email:", reply_markup=utils.markup_admin_ops)

@bot.message_handler(func=lambda message: users_step.get(message.chat.id) == utils.ADMIN_SET_ADMIN)
@handleError
def email_set_admin(message):
	user = userdao_instance.get_by_email(message.text)
	if check_email_match_regex(message):
		return
```

```
        if check_email_consistence(message, user):
                return

        userdao_instance.update_role(message.text, 'ROLE_ADMIN')
        users_step[message.chat.id] = utils.USER_ACCEPTED
        updated_chat_id = user[0][3]
        users_chatid_roles[updated_chat_id] = 'ROLE_ADMIN'

        if updated_chat_id != 'NULL':
                if int(updated_chat_id) in users_active:
                        users_active.pop(int(updated_chat_id))

        bot.reply_to(message, "В системе новый администратор.")
        if updated_chat_id != 'NULL':
                bot.send_message(updated_chat_id, "Васповысиливправах, перезайдитевсистему", reply_markup=utils.markup_unauth)

# RESET
@bot.message_handler(regexp=u"Убрать админа")
def reset_admin_keyboard_handler(message):
        reset_admin_command(message)

@bot.message_handler(commands=['reset_admin'])
```

```
@handleError
def reset_admin_command(message):
	chat_id = message.chat.id
	if not resolve_admin_role(chat_id):
		return

	users_step[chat_id] = utils.ADMIN_RESET_ADMIN
	bot.send_message(chat_id, "Введите email:", reply_markup=utils.markup_admin_ops)

@bot.message_handler(func=lambda message: users_step.get(message.chat.id) == utils.ADMIN_RESET_ADMIN)
@handleError
def email_reset_admin(message):
	user = userdao_instance.get_by_email(message.text)
	if check_email_match_regex(message):
		return
	if check_email_consistence(message, user):
		return

	userdao_instance.update_role(message.text, 'ROLE_USER')
	users_step[message.chat.id] = utils.USER_ACCEPTED
	updated_chat_id = user[0][3]
	users_chatid_roles[updated_chat_id] = 'ROLE_USER'
```

```
        ifupdated_chat_id != 'NULL':
                if int(updated_chat_id) in users_active:
                        users_active.pop(int(updated_chat_id))

        bot.reply_to(message, "Администратор понижен в правах.")
        if updated_chat_id != 'NULL':
                bot.send_message(updated_chat_id, "Васпонизиливправах, перезайдитевсистему", reply_markup=utils.markup_unauth)

# USER OPS
# UPDATE NAME
@bot.message_handler(regexp=u"Изменитьимя")
def update_name_keyboard_handler(message):
        update_name_command(message)

@bot.message_handler(commands=['update_name'])
@handleError
def update_name_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return
```

```
    users_step[chat_id] = utils.USER_UPDATE_NAME
    bot.send_message(chat_id, "Введитеновоеимя:", reply_markup=utils.markup_profile_ops)


@bot.message_handler(func=lambda message: users_step.get(message.chat.id) == utils.USER_UPDATE_NAME)
def name_update_user(message):
    userdao_instance.update_name(str(message.chat.id), message.text)
    users_step[message.chat.id] = utils.USER_ACCEPTED
    bot.reply_to(message, "Вашеимяизменено", reply_markup=utils.markup_profile_ops)



# UPDATE EMAIL
@bot.message_handler(regexp=u"Изменить email")
def update_email_keyboard_handler(message):
    update_email_command(message)

@bot.message_handler(commands=['update_email'])
@handleError
def update_email_command(message):
    chat_id = message.chat.id
    if not check_user_auth(chat_id):
        return
```

```
    users_step[chat_id] = utils.USER_UPDATE_EMAIL
    bot.send_message(chat_id, "Введитеновый email:", reply_markup=utils.markup_profile_ops)


@bot.message_handler(func=lambda message: users_step.get(message.chat.id) == utils.USER_UPDATE_EMAIL)
@handleError
def email_update_user(message):
    if check_email_match_regex(message):
        return
    if check_double_email(message):
        return

    userdao_instance.update_email(str(message.chat.id), message.text)
    users_step[message.chat.id] = utils.USER_ACCEPTED
    users_active.pop(message.chat.id)
    bot.reply_to(message, "Ваш email изменен. Перезайдитевсистему", reply_markup=utils.markup_unauth)

# REMOVE PROFILE
@bot.message_handler(regexp=u"Удалить профиль")
def remove_profile_keyboard_handler(message):
    remove_profile_command(message)

@bot.message_handler(commands=['remove_profile'])
```

```
@handleError
def remove_profile_command(message):
	chat_id = message.chat.id
	if not check_user_auth(chat_id):
		return

	users_step[chat_id] = utils.USER_REMOVE_CONF
	bot.send_message(chat_id, "Подтвердитеудаление (Да/Нет):", reply_markup=utils.markup_profile_ops)

@bot.message_handler(func=lambda message: users_step.get(message.chat.id) == utils.USER_REMOVE_CONF)
@handleError
def conf_remove_user(message):
	chat_id = message.chat.id
	if message.text.lower() != u'да':
		bot.reply_to(message, "Удалениеотменено", reply_markup=utils.markup_profile_ops)
		return

	userdao_instance.delete_by_chat_id(str(chat_id))
	users_step.pop(chat_id)
	users_active.pop(chat_id)
	users_chatid_roles.pop(str(chat_id))
	bot.reply_to(message, "Вашпрофильудален, вынеможетевойтивсистему", reply_markup=utils.markup_unauth)
```

```
# HELP
@bot.message_handler(regexp=u"Помощь")
def help_keyboard_handler(message):
	help_command(message)

@bot.message_handler(commands=['help'])
@handleError
def help_command(message):
	if message.chat.id in users_active:
		if users_chatid_roles[str(message.chat.id)] == 'ROLE_USER':
			bot.send_message(message.chat.id, utils.HELP_USER_MESSAGE, reply_markup=utils.markup_user)
			return
		else:
			bot.send_message(message.chat.id, utils.HELP_ADMIN_MESSAGE, reply_markup=utils.markup_admin)
	else:
		bot.send_message(message.chat.id, utils.HELP_UNAUTH_MESSAGE, reply_markup=utils.markup_unauth)

# KEYBOARD LEVELS
# MAIN
@bot.message_handler(commands=['main'])
@handleError
def main_enter_command(message):
```

```
    if not check_user_auth(message.chat.id):
        return
    bot.send_message(message.chat.id, "Выберитедействие", reply_markup=utils.markup_main)

@bot.message_handler(regexp=u"Управление системой")
def main_keyboard_enter(message):
    main_enter_command(message)

@bot.message_handler(regexp=u"Закрыть меню системы")
@handleError
def main_keyboard_close(message):
    if not check_user_auth(message.chat.id):
        return
    bot.send_message(message.chat.id, "Выберитедействие", reply_markup=check_user_role_for_keyboard(message.chat.id))

# ADMIN OPS
@bot.message_handler(commands=['admin'])
@handleError
def admin_enter_command(message):
    if not check_user_auth(message.chat.id):
        return
    if not resolve_admin_role(message.chat.id):
```

```
            return
        bot.send_message(message.chat.id, "Выберитедействие", reply_markup=utils.markup_admin_ops)


@bot.message_handler(regexp=u"Операции с пользователями")
def admin_keyboard_enter(message):
        admin_enter_command(message)


@bot.message_handler(regexp=u"Закрыть меню админа")
@handleError
def admin_keyboard_close(message):
        if not check_user_auth(message.chat.id):
            return
        if not resolve_admin_role(message.chat.id):
            return
        bot.send_message(message.chat.id, "Выберитедействие", reply_markup=utils.markup_admin)


# PROFILE OPS
@bot.message_handler(commands=['profile'])
@handleError
def profile_enter_command(message):
        if not check_user_auth(message.chat.id):
            return
```

```
        bot.send_message(message.chat.id, "Выберитедействие", reply_markup=utils.markup_profile_ops)


@bot.message_handler(regexp=u"Настройки профиля")
def profile_keyboard_enter(message):
        profile_enter_command(message)


@bot.message_handler(regexp=u"Закрыть меню профиля")
@handleError
def profile_keyboard_close(message):
        if not check_user_auth(message.chat.id):
                return
        bot.send_message(message.chat.id, "Выберитедействие", reply_markup=check_user_role_for_keyboard(message.chat.id))


# STATS OPS
@bot.message_handler(commands=['stat'])
@handleError
def stat_enter_command(message):
        if not check_user_auth(message.chat.id):
                return
        bot.send_message(message.chat.id, "Выберитедействие", reply_markup=utils.markup_stat)


@bot.message_handler(regexp=u"Статистика")
```

```
def stat_keyboard_enter(message):
        stat_enter_command(message)

@bot.message_handler(regexp=u"Закрыть меню статистики")
@handleError
def stat_keyboard_close(message):
        if not check_user_auth(message.chat.id):
                return
        bot.send_message(message.chat.id, u"Выберитедействие", reply_markup=utils.markup_main)

# UTIL METHODS
def check_email_match_regex(message):
        if not EMAIL_REGEX.match(message.text):
                bot.reply_to(message, "Email некорректен.")
                users_step[message.chat.id] = utils.USER_ACCEPTED
                return True
        return False

def check_email_consistence(message, user):
        if user == None:
                bot.reply_to(message, "Email ненайден.")
                users_step[message.chat.id] = utils.USER_ACCEPTED
```

```
            return True
        return False

def check_double_email(message):
        if userdao_instance.get_by_email(message.text) != None:
            bot.reply_to(message, "Email дублируется.")
            users_step[message.chat.id] = utils.USER_ACCEPTED
            return True
        return False

def check_user_role_for_keyboard(chat_id):
        if users_chatid_roles[str(chat_id)] == 'ROLE_ADMIN':
            return utils.markup_admin
        return utils.markup_user

def resolve_admin_role(chat_id):
        role_flag = check_user_role(chat_id)
        if not role_flag:
            bot.send_message(chat_id, "Доступккомандеограничен.")
            return False
        if role_flag == None:
            bot.send_message(chat_id, "Вы не авторизованы.")
```

```
                return False
        return True

def check_user_auth(chat_id):
        role_flag = check_user_role(chat_id)
        if role_flag == None:
                bot.send_message(chat_id, "Вынеавторизованы.", reply_markup=utils.markup_unauth)
                return False
        return True

def check_user_role(chat_id):
        if not check_user_role_activity(chat_id):
                return None
        if users_active[chat_id] == 'ROLE_ADMIN':
                return True
        else:
                return False

def check_user_role_activity(chat_id):
        if chat_id in users_active:
                returnTrue
        else:
```

```
                return False


# PHASE_A
@bot.message_handler(commands=['phase_a'])
@handleError
def phase_a_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return


        if not GPIO.input(27):
                GPIO.output(27,True)
                bot.send_message(chat_id, "Фаза А ВКЛ", reply_markup=utils.markup_main)
                return
        if GPIO.input(27):
                GPIO.output(27,False)
                bot.send_message(chat_id, "Фаза А ВЫКЛ", reply_markup=utils.markup_main)


@bot.message_handler(regexp=u"Фаза А")
def phase_a(message):
        phase_a_command(message)
```

```
# PHASE_B
@bot.message_handler(commands=['phase_b'])
@handleError
def phase_b_command(message):
	chat_id = message.chat.id
	if not check_user_auth(chat_id):
		return

	if not GPIO.input(17):
		GPIO.output(17,True)
		bot.send_message(chat_id, "Фаза В ВКЛ", reply_markup=utils.markup_main)
		return
	if GPIO.input(17):
		GPIO.output(17,False)
		bot.send_message(chat_id, "Фаза В ВЫКЛ", reply_markup=utils.markup_main)

@bot.message_handler(regexp=u"Фаза В")
def phase_b(message):
	phase_b_command(message)

# PHASE_C
@bot.message_handler(commands=['phase_c'])
```

```
@handleError
def phase_c_command(message):
	chat_id = message.chat.id
	if not check_user_auth(chat_id):
		return

	if not GPIO.input(18):
		GPIO.output(18,True)
		bot.send_message(chat_id, "Фаза С ВКЛ", reply_markup=utils.markup_main)
		return
	if GPIO.input(18):
		GPIO.output(18,False)
		bot.send_message(chat_id, "Фаза С ВЫКЛ", reply_markup=utils.markup_main)

@bot.message_handler(regexp=u"Фаза С")
def phase_c(message):
	phase_c_command(message)

# TEMP IN
@bot.message_handler(commands=['tmp_in'])
@handleError
def tmp_in_command(message):
```

```
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return

        result = dht_instance.read()
        while result.is_valid() != True :
                result = dht_instance.read()
        temperature = "Температуравнутри: " + str(result.temperature) + " °C"
        bot.send_message(chat_id, temperature, reply_markup=utils.markup_main)

@bot.message_handler(regexp=u"Температура внутри")
def get_temperature_inside(message):
        tmp_in_command(message)

# TEMP OUT
@bot.message_handler(commands=['tmp_out'])
@handleError
def tmp_out_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return
```

```
        ps_data = bme_instance.get_data()
        temperature = "Температураснаружи: " + str(ps_data['t']) + " °C"
        bot.send_message(chat_id, temperature, reply_markup=utils.markup_main)


@bot.message_handler(regexp=u"Температура снаружи")
def get_temperature_outside(message):
        tmp_out_command(message)

# HUMIDITY
@bot.message_handler(commands=['humid'])
@handleError
def humid_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return

        if message.text == u"Влажность воздуха за сутки":
                get_humidity_last_day(message)
                return
        if message.text == u"Влажность воздуха за неделю":
                get_humidity_last_week(message)
                return
```

```
        else:
                result = dht_instance.read()
                while result.is_valid() != True :
                        result = dht_instance.read()
                humidity = "Влажность: " + str(result.humidity) + "%"
                bot.send_message(chat_id, humidity, reply_markup=utils.markup_main)


@bot.message_handler(regexp=u"Влажность")
def get_humidity(message):
        humid_command(message)

# PRESSURE
@bot.message_handler(commands=['press'])
@handleError
def press_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return

        if message.text == u"Атм. давлениезасутки":
                get_pressures_last_day(message)
                return
```

```
        ifmessage.text == u"Атм. давлениезанеделю":
                get_pressures_last_week(message)
                return
        else:
                ps_data = bme_instance.get_data()
                pressure = "Давление: " + str(round(ps_data['p']/133.3224)) + " мм"
                bot.send_message(chat_id, pressure, reply_markup=utils.markup_main)


@bot.message_handler(regexp=u"Атм. давление")
def get_pressure(message):
        press_command(message)

# STATISTICS
# TMP_OUT_DAY
@bot.message_handler(commands=['tmp_out_d'])
@handleError
def tmp_out_d_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return

        plot_instance = PlotService()
```

```
        plot_instance.get_temps_out_last_day()
        bot.send_photo(chat_id, open('./png/ex_1_6_1.png', 'rb'))


@bot.message_handler(regexp=u"T°C снаружи за сутки")
def get_temps_out_last_day(message):
        tmp_out_d_command(message)


# TMP_IN_DAY
@bot.message_handler(commands=['tmp_in_d'])
@handleError
def tmp_in_d_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return

        plot_instance = PlotService()
        plot_instance.get_temps_in_last_day()
        bot.send_photo(chat_id, open('./png/ex_1_6_1.png', 'rb'))

@bot.message_handler(regexp=u"T°C внутри за сутки")
def get_temps_in_last_day(message):
        tmp_in_d_command(message)
```

```
# TMP_OUT_WEEK
@bot.message_handler(commands=['tmp_out_w'])
@handleError
def tmp_out_w_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return

        plot_instance = PlotService()
        plot_instance.get_temps_out_last_week()
        bot.send_photo(chat_id, open('./png/ex_1_6_1.png', 'rb'))

@bot.message_handler(regexp=u"T°C снаружи за неделю")
def get_temps_out_last_week(message):
        tmp_out_w_command(message)

# TMP_IN_WEEK
@bot.message_handler(commands=['tmp_in_w'])
@handleError
def tmp_in_w_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
```

```
        return

    plot_instance = PlotService()
    plot_instance.get_temps_in_last_week()
    bot.send_photo(chat_id, open('./png/ex_1_6_1.png', 'rb'))

@bot.message_handler(regexp=u"T°C внутри за неделю")
def get_temps_in_last_week(message):
    tmp_in_w_command(message)

# PRESS_DAY
@bot.message_handler(commands=['press_d'])
@handleError
def press_d_command(message):
    chat_id = message.chat.id
    if not check_user_auth(chat_id):
        return

    plot_instance = PlotService()
    plot_instance.get_pressures_last_day()
    bot.send_photo(chat_id, open('./png/ex_1_6_1.png', 'rb'))
```

```
def get_pressures_last_day(message):
        plot_instance = PlotService()
        plot_instance.get_pressures_last_day()
        bot.send_photo(chat_id, open('./png/ex_1_6_1.png', 'rb'))

# HUMID_DAY
@bot.message_handler(commands=['humid_d'])
@handleError
def humid_d_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return

        plot_instance = PlotService()
        plot_instance.get_humidity_last_day()
        bot.send_photo(chat_id, open('./png/ex_1_6_1.png', 'rb'))

def get_humidity_last_day(message):
        plot_instance = PlotService()
        plot_instance.get_humidity_last_day()
        bot.send_photo(message.chat.id, open('./png/ex_1_6_1.png', 'rb'))
```

```
# PRESS_WEEK
@bot.message_handler(commands=['press_w'])
@handleError
def press_w_command(message):
	chat_id = message.chat.id
	if not check_user_auth(chat_id):
		return

	plot_instance = PlotService()
	plot_instance.get_pressures_last_week()
	bot.send_photo(chat_id, open('./png/ex_1_6_1.png', 'rb'))

def get_pressures_last_week(message):
	plot_instance = PlotService()
	plot_instance.get_pressures_last_week()
	bot.send_photo(message.chat.id, open('./png/ex_1_6_1.png', 'rb'))

# HUMID_WEEK
@bot.message_handler(commands=['humid_w'])
@handleError
def humid_w_command(message):
	chat_id = message.chat.id
```

```
        if not check_user_auth(chat_id):
                return

        plot_instance = PlotService()
        plot_instance.get_humidity_last_week()
        bot.send_photo(chat_id, open('./png/ex_1_6_1.png', 'rb'))

def get_humidity_last_week(message):
        plot_instance = PlotService()
        plot_instance.get_humidity_last_week()
        bot.send_photo(message.chat.id, open('./png/ex_1_6_1.png', 'rb'))

# ELECTRICITY
@bot.message_handler(commands=['electr'])
@handleError
def electricity_command(message):
        chat_id = message.chat.id
        if not check_user_auth(chat_id):
                return
        if not omix_instance.client.connect():
                bot.send_message(chat_id, u"Устройстванеподключено", reply_markup=utils.markup_main)
                return
```

```
        data = ('Напряжение: ' + omix_instance.get_voltage() + 'V\n'
                               'Силатока:' + omix_instance.get_current() + 'A\n'
                               'Частота:' + omix_instance.get_freq() + 'Гц\n'
                               'Мощность:' + omix_instance.get_power() + 'Вт\n')
        bot.send_message(chat_id, data, reply_markup=utils.markup_main)


@bot.message_handler(regexp=u"Параметры электр. сети")
def get_electricity(message):
        electricity_command(message)


# EXCEPTIONS
def send_traceback(traceback, message):
        create_pdf(traceback)
        bot.send_document(message.chat.id, open('./pdf/stack.pdf', 'rb'))


def create_pdf(traceback):
        traceback = traceback.split("\n")
        pdf = FPDF()
        pdf.add_page()
        pdf.set_font('Arial', 'B', 12)
        pdf.cell(40, 10, "Send this error report to the administrator if the problem occurs again: \n", 0, 1)
```

```
            for x in xrange(0, len(traceback)):
                pdf.cell(40, 10, traceback[x], 0, 1)
                print traceback[x]
            pdf.output('pdf/stack.pdf', 'F')

    # CHOOSE_PARAM
    @bot.message_handler(content_types=["text"])
    def choose_param(message):
            if not check_user_auth(message.chat.id):
                return
            bot.send_message(message.chat.id, u"Выберитедействие", reply_markup=check_user_role_for_keyboard(message.chat.id))

    if __name__ == '__main__':
            while True:
                try:
                    bot.polling(none_stop=True, interval=1)
                except Exception as e:
                    time.sleep(4)

    GPIO.cleanup()
```

____//СкриптBME280.py____

```
# -*- coding: utf-8 -*-
# Copyright (c) 2015 Andrey Koryagin
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
# THE SOFTWARE.

import smbus
import time

# BME280 defaultaddress.
```

```
BME280_I2CADDR         = 0x76

# BME280 Registers
BME280_CONTROL_MEAS     = 0xF4
BME280_CONTROL_HUM      = 0xF2

BME280_CONFIG          = 0xF5
BME280_PRESSURE        = 0xF7
BME280_TEMP            = 0xFA
BME280_HUMIDITY        = 0xFD

BME280_DIG_T1          = 0x88
BME280_DIG_T2          = 0x8A
BME280_DIG_T3          = 0x8C
BME280_DIG_P1          = 0x8E
BME280_DIG_P2          = 0x90
BME280_DIG_P3          = 0x92
BME280_DIG_P4          = 0x94
BME280_DIG_P5          = 0x96
BME280_DIG_P6          = 0x98
BME280_DIG_P7          = 0x9A
BME280_DIG_P8          = 0x9C
```

```
BME280_DIG_P9          = 0x9E
BME280_DIG_H1          = 0xA1
BME280_DIG_H2          = 0xE1
BME280_DIG_H3          = 0xE3
BME280_DIG_H4          = 0xE4
BME280_DIG_H5          = 0xE5
BME280_DIG_H6          = 0xE7

# Oversampling Setting
BME280_OVERS_T1        = 0x20
BME280_OVERS_T2        = 0x40
BME280_OVERS_T4        = 0x60
BME280_OVERS_T8        = 0x80
BME280_OVERS_T16       = 0xA0

BME280_OVERS_P1        = 0x04
BME280_OVERS_P2        = 0x08
BME280_OVERS_P4        = 0x0C
BME280_OVERS_P8        = 0x10
BME280_OVERS_P16       = 0x14

BME280_OVERS_H1        = 0x01
```

```
BME280_OVERS_H2         = 0x02
BME280_OVERS_H4         = 0x03
BME280_OVERS_H8         = 0x04
BME280_OVERS_H16        = 0x05

# Power Modes.
# This lib uses NORMAL mode only!
#BME280_SLEEP_MODE       = 0x00
#BME280_FORCED_MODE      = 0x01
BME280_NORMAL_MODE      = 0x03

BME280_TSB_0_5          = 0x00
BME280_TSB_62_5         = 0x20
BME280_TSB_125          = 0x40
BME280_TSB_250          = 0x60
BME280_TSB_500          = 0x80
BME280_TSB_1000         = 0xA0
BME280_TSB_2000         = 0xC0
BME280_TSB_4000         = 0xE0

BME280_FILTER_OFF             = 0x00
BME280_FILTER_COEFFICIENT2    = 0x04
```

```
BME280_FILTER_COEFFICIENT4      = 0x08
BME280_FILTER_COEFFICIENT8      = 0x0C
BME280_FILTER_COEFFICIENT16     = 0x10

BME280_SPI_OFF          = 0x00
BME280_SPI_ON           = 0x01

BME280_CONTROL_MEAS_SET     = (BME280_OVERS_T16 | BME280_OVERS_P16 | BME280_NORMAL_MODE)
BME280_CONTROL_HUM_SET = BME280_OVERS_H2
BME280_CONFIG_SET       = (BME280_TSB_0_5 | BME280_FILTER_COEFFICIENT16 | BME280_SPI_OFF)

class BME280(object):
        def __init__(self, port=1, address=BME280_I2CADDR):
                self.bus = smbus.SMBus(port)
                self.address = address

                # Read calibration values
                self.dig_t1 = self.read_word(BME280_DIG_T1)     # Unsigned
                self.dig_t2 = self.read_word_sign(BME280_DIG_T2)
                self.dig_t3 = self.read_word_sign(BME280_DIG_T3)
                self.dig_p1 = self.read_word(BME280_DIG_P1)     # Unsigned
                self.dig_p2 = self.read_word_sign(BME280_DIG_P2)
```

```
self.dig_p3 = self.read_word_sign(BME280_DIG_P3)
self.dig_p4 = self.read_word_sign(BME280_DIG_P4)
self.dig_p5 = self.read_word_sign(BME280_DIG_P5)
self.dig_p6 = self.read_word_sign(BME280_DIG_P6)
self.dig_p7 = self.read_word_sign(BME280_DIG_P7)
self.dig_p8 = self.read_word_sign(BME280_DIG_P8)
self.dig_p9 = self.read_word_sign(BME280_DIG_P9)

self.dig_h1 = self.read_byte(BME280_DIG_H1)  # unsigned char
self.dig_h2 = self.read_word_sign(BME280_DIG_H2)
self.dig_h3 = self.read_byte(BME280_DIG_H3)  # unsigned char

self.dig_h4 = (self.read_byte(BME280_DIG_H4) << 24) >> 20
self.dig_h4 = self.dig_h4 | self.read_byte(BME280_DIG_H4+1) & 0x0F

self.dig_h5 = (self.read_byte(BME280_DIG_H5+1) << 24) >> 20
self.dig_h5 = self.dig_h5 | (self.read_byte(BME280_DIG_H5) >> 4) & 0x0F

self.dig_h6 = self.read_byte(BME280_DIG_H6)  # signed char
if self.dig_h6 > 127:
        self.dig_h6 = 127-self.dig_h6
```

```
        # Set Configuration
  self.write_byte(BME280_CONFIG, BME280_CONFIG_SET)
        self.write_byte(BME280_CONTROL_HUM, BME280_CONTROL_HUM_SET)
        self.write_byte(BME280_CONTROL_MEAS, BME280_CONTROL_MEAS_SET)

def get_data(self):
        adc_t = self.read_adc_long(BME280_TEMP)
        adc_p = self.read_adc_long(BME280_PRESSURE)
        adc_h = self.read_adc_word(BME280_HUMIDITY)

        var1 = (adc_t/16384.0 - self.dig_t1/1024.0) * self.dig_t2;
        var2 = ((adc_t/131072.0 - self.dig_t1/8192.0) * (adc_t/131072.0 - self.dig_t1/8192.0)) * self.dig_t3;
        t_fine = (var1 + var2);
        temperature = round((t_fine / 5120.0), 2);

        var1 = (t_fine/2.0) - 64000.0;
        var2 = var1 * var1 * self.dig_p6 / 32768.0;
        var2 = var2 + var1 * self.dig_p5 * 2.0;
        var2 = (var2/4.0)+(self.dig_p4 * 65536.0);
        var1 = (self.dig_p3 * var1 * var1 / 524288.0 + self.dig_p2 * var1) / 524288.0;
        var1 = (1.0 + var1 / 32768.0)*self.dig_p1;
```

```
        # Avoid exception caused by division by zero
        if (var1 == 0.0):
                return -1

        p = 1048576.0 - adc_p;
        p = (p - (var2 / 4096.0)) * 6250.0 / var1;
        var1 = self.dig_p9 * p * p / 2147483648.0;
        var2 = p * self.dig_p8 / 32768.0;
        pressure = round((p + (var1 + var2 + self.dig_p7) / 16.0), 2);

        var_H = t_fine - 76800.0
        var_H = (adc_h-(self.dig_h4*64.0+self.dig_h5 / 16384.0 * var_H))*(self.dig_h2/65536.0 * (1.0 + self.dig_h6/67108864.0 * var_H *
(1.0+self.dig_h3/67108864.0 * var_H)))
        humidity = round(var_H * (1.0 - self.dig_h1*var_H/524288.0), 2)
        if (humidity > 100.0):
                humidity = 100.0
        else:
                if(humidity < 0.0):
                        humidity = 0.0

        return {'t':temperature, 'p':pressure, 'h':humidity}
```

```
def get_altitude(self, pressure):
        temp = pressure/101325;
        temp = 1-pow(temp, 0.19029);
        altitude = round(44330*temp, 3);
        return altitude;

def read_byte(self, adr):
        return self.bus.read_byte_data(self.address, adr)

def read_word(self, adr):
        # ATANTION! Joke from Bosch! LBS before HBS. For calibration registers only!
        lbs = self.bus.read_byte_data(self.address, adr)
        hbs = self.bus.read_byte_data(self.address, adr+1)
        return (hbs << 8) + lbs

def read_word_sign(self, adr):
        val = self.read_word(adr)
        if (val >= 0x8000):
                return -((65535 - val) + 1)
        else:
                returnval
```

```
        def read_adc_long(self, adr):
                mbs = self.bus.read_byte_data(self.address, adr)
                lbs = self.bus.read_byte_data(self.address, adr+1)
                xbs = self.bus.read_byte_data(self.address, adr+2)
                val = (mbs << 16) + (lbs << 8) + xbs
                val = (val >> 4)
                return val

    def read_adc_word(self, adr):
        mbs = self.bus.read_byte_data(self.address, adr)
        lbs = self.bus.read_byte_data(self.address, adr+1)
        val = (mbs << 8) + lbs
        return val

        def write_byte(self, adr, byte):
                self.bus.write_byte_data(self.address, adr, byte)
```

___//Скриптdht11.py___

```
import time
import RPi
```

```
classDHT11Result:
'DHT11 sensor result returned by DHT11.read() method'

    ERR_NO_ERROR = 0
    ERR_MISSING_DATA = 1
    ERR_CRC = 2

    error_code = ERR_NO_ERROR
    temperature = -1
    humidity = -1

    def __init__(self, error_code, temperature, humidity):
        self.error_code = error_code
        self.temperature = temperature
        self.humidity = humidity

    def is_valid(self):
        return self.error_code == DHT11Result.ERR_NO_ERROR


class DHT11:
    'DHT11 sensor reader class for Raspberry'
```

```
    __pin = 0

def __init__(self, pin):
self.__pin = pin

    def read(self):
        RPi.GPIO.setup(self.__pin, RPi.GPIO.OUT)

        # send initial high
        self.__send_and_sleep(RPi.GPIO.HIGH, 0.05)

        # pull down to low
        self.__send_and_sleep(RPi.GPIO.LOW, 0.02)

        # change to input using pull up
        RPi.GPIO.setup(self.__pin, RPi.GPIO.IN, RPi.GPIO.PUD_UP)

        # collect data into an array
        data = self.__collect_input()

        # parse lengths of all data pull up periods
        pull_up_lengths = self.__parse_data_pull_up_lengths(data)
```

```
    # if bit count mismatch, return error (4 byte data + 1 byte checksum)
    if len(pull_up_lengths) != 40:
        return DHT11Result(DHT11Result.ERR_MISSING_DATA, 0, 0)

    # calculate bits from lengths of the pull up periods
    bits = self.__calculate_bits(pull_up_lengths)

    # we have the bits, calculate bytes
    the_bytes = self.__bits_to_bytes(bits)

    # calculate checksum and check
    checksum = self.__calculate_checksum(the_bytes)
    if the_bytes[4] != checksum:
        return DHT11Result(DHT11Result.ERR_CRC, 0, 0)

    # ok, we have valid data, return it
    return DHT11Result(DHT11Result.ERR_NO_ERROR, the_bytes[2], the_bytes[0])

def __send_and_sleep(self, output, sleep):
    RPi.GPIO.output(self.__pin, output)
    time.sleep(sleep)
```

```
def __collect_input(self):
    # collect the data while unchanged found
    unchanged_count = 0

    # this is used to determine where is the end of the data
    max_unchanged_count = 100

    last = -1
    data = []
    while True:
      current = RPi.GPIO.input(self.__pin)
      data.append(current)
      if last != current:
        unchanged_count = 0
        last = current
      else:
        unchanged_count += 1
        if unchanged_count > max_unchanged_count:
          break

returndata
```

```
def __parse_data_pull_up_lengths(self, data):
    STATE_INIT_PULL_DOWN = 1
    STATE_INIT_PULL_UP = 2
    STATE_DATA_FIRST_PULL_DOWN = 3
    STATE_DATA_PULL_UP = 4
    STATE_DATA_PULL_DOWN = 5

    state = STATE_INIT_PULL_DOWN

    lengths = [] # will contain the lengths of data pull up periods
    current_length = 0 # will contain the length of the previous period

    for i in range(len(data)):

      current = data[i]
      current_length += 1

      if state == STATE_INIT_PULL_DOWN:
        if current == RPi.GPIO.LOW:
          # ok, we got the initial pull down
          state = STATE_INIT_PULL_UP
          continue
```

```
    else:
        continue
if state == STATE_INIT_PULL_UP:
    if current == RPi.GPIO.HIGH:
        # ok, we got the initial pull up
        state = STATE_DATA_FIRST_PULL_DOWN
        continue
    else:
        continue
if state == STATE_DATA_FIRST_PULL_DOWN:
    if current == RPi.GPIO.LOW:
        # we have the initial pull down, the next will be the data pull up
        state = STATE_DATA_PULL_UP
        continue
    else:
        continue
if state == STATE_DATA_PULL_UP:
    if current == RPi.GPIO.HIGH:
        # data pulled up, the length of this pull up will determine whether it is 0 or 1
        current_length = 0
        state = STATE_DATA_PULL_DOWN
        continue
```

```
else:
continue
        if state == STATE_DATA_PULL_DOWN:
            if current == RPi.GPIO.LOW:
                # pulled down, we store the length of the previous pull up period
                lengths.append(current_length)
                state = STATE_DATA_PULL_UP
                continue
            else:
                continue

    return lengths

  def __calculate_bits(self, pull_up_lengths):
    # find shortest and longest period
    shortest_pull_up = 1000
    longest_pull_up = 0

    for i in range(0, len(pull_up_lengths)):
        length = pull_up_lengths[i]
        if length < shortest_pull_up:
            shortest_pull_up = length
```

```
if length > longest_pull_up:
            longest_pull_up = length

    # use the halfway to determine whether the period it is long or short
    halfway = shortest_pull_up + (longest_pull_up - shortest_pull_up) / 2
    bits = []

    for i in range(0, len(pull_up_lengths)):
      bit = False
      if pull_up_lengths[i] > halfway:
        bit = True
      bits.append(bit)

    return bits

  def __bits_to_bytes(self, bits):
    the_bytes = []
    byte = 0

    for i in range(0, len(bits)):
      byte = byte << 1
      if (bits[i]):
```

```
byte = byte | 1
else:
            byte = byte | 0
        if ((i + 1) % 8 == 0):
            the_bytes.append(byte)
            byte = 0

    return the_bytes

  def __calculate_checksum(self, the_bytes):
    return the_bytes[0] + the_bytes[1] + the_bytes[2] + the_bytes[3] & 255
```

___//Скриптmail.py___

```
import smtplib
import datetime
import logging

from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

logging.basicConfig(format = u'%(levelname)-8s [%(asctime)s] %(message)s',
                                        level = logging.DEBUG, filename = u'logging.log')
```

```
log = logging.getLogger()

class MailService(object):
	def __init__(self):
		self.smtpObj = smtplib.SMTP('smtp.gmail.com', 587)
		self.smtpObj.starttls()
		self.smtpObj.login('email','password for email')
		log.info("Login successful.")

	def send_mail(self, email, token):
		me = "rpi.telebot@gmail.com"
		you = email

		msg = MIMEMultipart('alternative')
		msg['Subject'] = "Your token"
		msg['From'] = me
		msg['To'] = you

		html = """\
		<html>
			<head>
				<metacharset="utf-8">
```

```
                <title>Telegram: Contact @raspberry_tele_bot</title>
                <meta name="viewport" content="width=device-width, initial-scale=1.0">
            </head>

            <body>
                <div>
                    <p>Your token for @raspberry_tele_bot</p>
                    <p>Follow the <a href="https://telegram.me/raspberry_tele_bot?start="""+ token +""""">link</a> for login.</p>
                    <p>Don't have Telegram yet? Try it now:</p>
                    <a href="telegram.org/dl">Get Telegram</a>
                </div>
            </body>
        </html>"""

        part = MIMEText(html, 'html')
        msg.attach(part)
        self.smtpObj.sendmail(me,email,msg.as_string())
        log.info("Sent successful.")

def smtp_quit(self):
        self.smtpObj.quit()
```

___//Скриптomix.py___

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import logging
import pymodbus
import serial
from pymodbus.pdu import ModbusRequest
from pymodbus.client.sync import ModbusSerialClient as ModbusClient
from pymodbus.transaction import ModbusRtuFramer

logging.basicConfig(format = u'%(levelname)-8s [%(asctime)s] %(message)s',
                                        level = logging.DEBUG, filename = u'logging.log')
log = logging.getLogger()

class ModbusSerialService(object):
        def __init__(self):
                super(ModbusSerialService, self).__init__()
                self.client = ModbusClient(method='rtu', port='/dev/ttyUSB0',
                                        stopbits = 1, bytesize = 8,
                                        parity = 'N', baudrate=9600)
                if not self.client.connect():
```

```
                message = u"Device is not connected."
                log.error(message)
        else:
                log.info(u"Device is connected.")

def get_voltage(self):
        address = 0
        count = 2
        response = client.read_input_registers(address, count, unit=128)
        return check_valid_data(response, "Voltage")

def get_current(self):
        address = 2
        count = 2
        response = client.read_input_registers(address, count, unit=128)
        return check_valid_data(response, "Current")

def get_freq(self):
        address = 4
        count = 2
        response = client.read_input_registers(address, count, unit=128)
        return check_valid_data(response, "Frequency")
```

```
defget_power(self):
        address = 10
        count = 2
        response = client.read_input_registers(address, count, unit=128)
        return check_valid_data(response, "Power")

def check_valid_data(self, response, data_type):
        if response != None:
                log.info(data_type + u" data is valid.")
                data = str(response[0]) + str(response[1])
                return data
        else:
                log.info(data_type + u" data is not valid.")
                response = "0"
                return response

def close_client(self):
        self.client.close()
```

___//Скриптplots.py___

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

```
importos
importlogging
import matplotlib
matplotlib.use('Agg')
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np

from sensordao import *

logging.basicConfig(format = u'%(levelname)-8s [%(asctime)s] %(message)s',
                                        level = logging.DEBUG, filename = u'logging.log')
log = logging.getLogger()

class PlotService(object):
        def __init__(self):
                super(PlotService, self).__init__()
                self.db_instance = SensorDao()
                self.x = []
                self.y = []
                self.xticks_day = [0.,2.,4.,6.,8.,10.,12.,14.,16.,18.,20.,22.,24.]
                self.xticks_week = [0., 1., 2., 3., 4., 5., 6., 7., 8.]
```

```
def get_temps_out_last_day(self):
        xlabel = u"Время, ч"
        ylabel = u"Температура, °C"
        title = u"Температура снаружи за последние сутки"
        temp_instance = self.db_instance.get_temps_out_last_day()

        for i in temp_instance:
                self.y.append(i[1])

        colors = ['red', 'blue']
        self.x = np.linspace(0.0, 24.0, num = 13)
        self.f_plot(self.xticks_day, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)

def get_temps_in_last_day(self):
        xlabel = u"Время, ч"
        ylabel = u"Температура, °C"
        title = u"Температура внутри за последние сутки"
        temp_instance = self.db_instance.get_temps_in_last_day()

        for i in temp_instance:
                self.y.append(i[1])
```

```
        colors = ['red', 'blue']
        self.x = np.linspace(0.0, 24.0, num = 13)
        self.f_plot(self.xticks_day, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)


def get_pressures_last_day(self):
        xlabel = u"Время, ч"
        ylabel = u"Атм. давление, мм рт. ст."
        title = u"Атмосферное давление за последние сутки"
        press_instance = self.db_instance.get_pressures_last_day()

        for i in press_instance:
                self.y.append(i[1])

        colors = ['red', 'blue']
        self.x = np.linspace(0.0, 24.0, num = 13)
        self.f_plot(self.xticks_day, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)

def get_humidity_last_day(self):
        xlabel = u"Время, ч"
        ylabel = u"Влажность, %"
        title = u"Влажность воздуха за последние сутки"
```

```
        humid_instance = self.db_instance.get_humidity_last_day()

        for i in humid_instance:
                self.y.append(i[1])

        colors = ['red', 'blue']
        self.x = np.linspace(0.0, 24.0, num = 13)
        self.f_plot(self.xticks_day, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)

def get_temps_out_last_week(self):
        xlabel = u"Дни"
        ylabel = u"Температура, °C"
        title = u"Средняя температура снаружи за неделю"
        temp_instance = self.db_instance.get_temps_out_last_week()

        temps = []
        n = 0
        for i in temp_instance:
                if n == 11:
                        val = "%.2f" % self.average_value(temps)
                        self.y.append(val)
                        temps = []
```

```
                    n = 0
                temps.append(i[1])
                n+=1

        colors = ['red', 'blue']
        self.x = np.linspace(1.0, 7.0, num = 7)
        self.f_plot(self.xticks_week, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)

def get_temps_in_last_week(self):
        xlabel = u"Дни"
        ylabel = u"Температура, °C"
        title = u"Средняя температура внутри за неделю"
        temp_instance = self.db_instance.get_temps_in_last_week()

        temps = []
        n = 0
        for i in temp_instance:
                if n == 11:
                        val = "%.2f" % self.average_value(temps)
                        self.y.append(val)
                        temps = []
                        n = 0
```

```
                temps.append(i[1])
                n+=1

        colors = ['red', 'blue']
        self.x = np.linspace(1.0, 7.0, num = 7)
        self.f_plot(self.xticks_week, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)

def get_pressures_last_week(self):
        xlabel = u"Дни"
        ylabel = u"Атм. давление, мм рт. ст."
        title = u"Атмосферное давление за неделю"
        press_instance = self.db_instance.get_pressures_last_week()

        press = []
        n = 0
        for i in press_instance:
                if n == 10:
                        val = "%.2f" % self.average_value(press)
                        self.y.append(val)
                        press = []
                        n = 0
                press.append(i[1])
```

```
                n+=1

        colors = ['red', 'blue']
        self.x = np.linspace(1.0, 7.0, num = 7)
        self.f_plot(self.xticks_week, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)

def get_humidity_last_week(self):
        xlabel = u"Дни"
        ylabel = u"Влажность воздуха, %"
        title = u"Влажность воздуха за неделю"
        humid_instance = self.db_instance.get_humidity_last_week()

        humid = []
        n = 0
        for i in humid_instance:
                if n == 10:
                        val = "%.2f" % self.average_value(humid)
                        self.y.append(val)
                        humid = []
                        n = 0
                humid.append(i[1])
                n+=1
```

```
        colors = ['red', 'blue']
        self.x = np.linspace(1.0, 7.0, num = 7)
        self.f_plot(self.xticks_week, title, xlabel, ylabel, self.x, self.y, colors=colors, linewidth=2.)

def f_plot(self, ticks, title, xlabel, ylabel, *args, **kwargs):
        xlist = []
        ylist = []
        for i, arg in enumerate(args):
                if(i % 2 == 0):
                        xlist.append(arg)
                else:
                        ylist.append(arg)

        colors = kwargs.pop('colors', 'k')
        linewidth = kwargs.pop('linewidth', 1.)

        fig = plt.figure()
        ax = fig.add_subplot(111)
        i = 0

        for x, y, color in zip(xlist, ylist, colors):
                i += 1
```

```
                ax.plot(x, y, color=color, linewidth=linewidth)

        ax.set_xlabel(xlabel, color='b')
        ax.set_ylabel(ylabel, color='b')
        ax.set_title(title, color='b')

        ax.xaxis.set_ticks(ticks)
        ax.grid(True)
        ax.legend()
        self.save('ex_1_6_1', fmt='png')

def save(self, name='', fmt='png'):
        pwd = os.getcwd()
        iPath = './{}'.format(fmt)
        if not os.path.exists(iPath):
                os.mkdir(iPath)
        os.chdir(iPath)
        plt.savefig('{}.{}'.format(name, fmt), fmt='png')
        os.chdir(pwd)
        plt.close()
        log.info(u"Save picture")
```

```
        defaverage_value(self, rows):
                summ = 0
                for row in rows:
                        summ += row
                return float(summ) / len(rows)
```

___//Скриптsensordao.py___

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import psycopg2
import datetime
import logging

logging.basicConfig(format = u'%(levelname)-8s [%(asctime)s] %(message)s',
                                        level = logging.DEBUG, filename = u'logging.log')
log = logging.getLogger()

# Init and populate DB (You can create your own methods in DBService)
# self.connect = psycopg2.connect("dbname='telebot'" +
#                                                                 "user='postgres'" +
#                                                                 "host='localhost'" +
#                                                                 "password='password'")
```

```
# self.cursor = self.connect.cursor()

# sqlfile = open('db/initDB.sql', 'r') # or populateDB.sql
# cursor.execute(self.sqlfile.read())
# connect.commit()

# cursor.close()
# connect.close()

class SensorDao(object):
        def __init__(self):
                self.connect = psycopg2.connect("dbname='telebot'" +
                                                "user='postgres'" +
                                                "host='localhost'" +
                                                "password='password'")
                self.cursor = self.connect.cursor()
                # sqlfile = open('db/populateDB.sql', 'r')
                # cursor.execute(self.sqlfile.read())
                # connect.commit()
                log.info("Connection successful.")

        def get_humidity_last_week(self):
```

```
        self.cursor.execute("SELECT * FROM humid_s "+
                                                "ORDER BY date_time ASC")
        rows = self.cursor.fetchall()

        log.info("Get humidity per last week")
        return rows

def get_humidity_last_day(self):
        self.cursor.execute("SELECT * FROM humid_s "+
                                                "WHERE date_time>='2017-03-23 00:00:00' "+
                                                "AND date_time<='2017-03-24 00:00:00' "+
                                                "ORDER BY date_time ASC")
        rows = self.cursor.fetchall()

        log.info("Get humidity per last day")
        return rows

def get_pressures_last_week(self):
        self.cursor.execute("SELECT * FROM press_s "+
                                                "ORDER BY date_time ASC")
        rows = self.cursor.fetchall()
```

```
        log.info("Get pressure per last week")
        return rows

def get_pressures_last_day(self):
        self.cursor.execute("SELECT * FROM press_s "+
                                                "WHERE date_time>='2017-03-23 00:00:00' "+
                                                "AND date_time<='2017-03-24 00:00:00' "+
                                                "ORDER BY date_time ASC")

        log.info("Get pressure per last day")
        rows = self.cursor.fetchall()
        return rows

def get_temps_in_last_week(self):
        self.cursor.execute("SELECT * FROM temps_in "+
                                                "ORDER BY date_time ASC")
        rows = self.cursor.fetchall()

        log.info("Get temperature in per last week")
        return rows

def get_temps_in_last_day(self):
```

```
        self.cursor.execute("SELECT * FROM temps_in "+
                                                "WHERE date_time>='2017-03-23 00:00:00' "+
                                                "AND date_time<='2017-03-24 00:00:00' "+
                                                "ORDER BY date_time ASC")
        rows = self.cursor.fetchall()

        log.info("Get temperature in per last day")
        return rows

def get_temps_out_last_week(self):
        self.cursor.execute("SELECT * FROM temps_out "+
                                                "ORDER BY date_time ASC")
        rows = self.cursor.fetchall()

        log.info("Get temperature out per last week")
        return rows

def get_temps_out_last_day(self):
        self.cursor.execute("SELECT * FROM temps_out "+
                                                "WHERE date_time>='2017-03-23 00:00:00' "+
                                                "AND date_time<='2017-03-24 00:00:00' "+
                                                "ORDER BY date_time ASC")
```

```
            rows = self.cursor.fetchall()

            log.info("Get temperature out per last day")
            return rows

    def get_user_by_token(self, token):
            self.cursor.execute("SELECT * FROM")

    def average_value(self, rows):
            summ = 0
            for row in rows:
                    summ += row[1]
            return float(summ) / len(rows)
```

___//Скриптuserdao.py___

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import psycopg2
import datetime
import logging

logging.basicConfig(format = u'%(levelname)-8s [%(asctime)s] %(message)s',
```

```
                                level = logging.DEBUG, filename = u'logging.log')
log = logging.getLogger()

class UserDao(object):
        def __init__(self):
                self.connect = psycopg2.connect("dbname='telebot'" +
                                                "user='postgres'" +
                                                "host='localhost'" +
                                                "password='password'")
                self.cursor = self.connect.cursor()
                self.connect.autocommit = True
                self.sqlfile = open('db/populateDB.sql', 'r')
                self.cursor.execute(self.sqlfile.read())
                self.connect.commit()
                log.info("Connection successful.")

        def get_all_with_roles(self):
                self.cursor.execute("SELECT * FROM users INNER JOIN user_roles"
                        + " ON (users.id = user_roles.user_id)")
                rows = self.cursor.fetchall()

                log.info("Get all users with roles")
```

```
        return rows

def get_all(self):
        self.cursor.execute("SELECT * FROM users")
        rows = self.cursor.fetchall()

        log.info("Get all users")
        return rows

def get_by_chat_id(self, chat_id):
        self.cursor.execute("SELECT * FROM users INNER JOIN user_roles"
                + " ON (users.id = user_roles.user_id) WHERE chat_id LIKE (%s)", (chat_id,))
        rows = self.cursor.fetchall()

        log.info("Get user by chat_id")
        return rows if len(rows) > 0 else None

def get_by_id(self, id):
        self.cursor.execute("SELECT * FROM users INNER JOIN user_roles"
                + " ON (users.id = user_roles.user_id) WHERE id=(%s)", (id,))
        rows = self.cursor.fetchall()
```

```
        log.info("Get user by id")
        return rows if len(rows) > 0 else None

def get_by_email(self, email):
        self.cursor.execute("SELECT * FROM users INNER JOIN user_roles"
                + " ON (users.id = user_roles.user_id) WHERE email LIKE (%s)", (email,))
        rows = self.cursor.fetchall()

        log.info("Get user by email")
        return rows if len(rows) > 0 else None

def get_by_token(self, token):
        self.cursor.execute("SELECT * FROM users INNER JOIN user_roles"
                + " ON (users.id = user_roles.user_id) WHERE token LIKE (%s)", (token,))
        rows = self.cursor.fetchall()

        log.info("Get user by token")
        return rows if len(rows) > 0 else None

def add(self, email, role):
        self.cursor.execute("INSERT INTO users (email) VALUES (%s)", (email,))
        self.cursor.execute("SELECT id FROM users WHERE email LIKE (%s)", (email,))
```

```
        user_id = self.cursor.fetchall()[0][0]
        self.cursor.execute("INSERT INTO user_roles (user_id, role) VALUES (%s,%s)", (user_id, role,))

        log.info("Create user with email:" + email + " and role " + role)

def update(self, name, email, chat_id, token):
        self.cursor.execute("UPDATE users SET name=(%s), chat_id=(%s), token=(%s) WHERE email LIKE (%s)",
                                                (name,chat_id,token,email,))

        log.info("Update user with email:" + email)

def update_name(self, chat_id, name):
        self.cursor.execute("UPDATE users SET name=(%s) WHERE chat_id LIKE (%s)",
                                                (name,chat_id,))

        log.info("Update user's name to " + name)

def update_token(self, email, token):
        self.cursor.execute("UPDATE users SET token=(%s) WHERE email LIKE (%s)",
                                                (token,email,))

        log.info("Update user's token with email:" + email)
```

```
def update_email(self, chat_id, email):
        self.cursor.execute("UPDATE users SET email=(%s) WHERE chat_id LIKE (%s)",
                                                (email,chat_id,))

        log.info("Update user's email to " + email)

def update_chat_id(self, token, chat_id):
        self.cursor.execute("UPDATE users SET chat_id=(%s) WHERE token LIKE (%s)",
                                                (chat_id,token,))

        log.info("Update user's chat id with token:" + token)

def update_role(self, email, role):
        self.cursor.execute("SELECT id FROM users WHERE email LIKE (%s)", (email,))
        user_id = self.cursor.fetchall()[0][0]
        self.cursor.execute("UPDATE user_roles SET role=(%s) WHERE user_id=(%s)", (role, user_id,))

        log.info("Update user's role with email:" + email)

def delete(self, email):
        self.cursor.execute("DELETE FROM users WHERE email LIKE (%s)", (email,))
```

```
            log.info("Delete user by email" + email)

    def delete_by_chat_id(self, chat_id):
            self.cursor.execute("DELETE FROM users WHERE chat_id LIKE (%s)", (chat_id,))

            log.info("Delete user by chat_id" + chat_id)
```

___//Скриптutils.py___

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from telebot import types
from userdao import *

userdao_instance = UserDao()

token = '358032242:AAERdFqaThpa_nSQDlBm5dzyaJ7Nj2rkNos'

USER_LOGGING = 'USER_LOGGING'
USER_ACCEPTED = 'USER_ACCEPTED'
ADMIN_SET_ADMIN = 'ADMIN_SET_ADMIN'
ADMIN_RESET_ADMIN = 'ADMIN_RESET_ADMIN'
```

```
ADMIN_ADD_USER = 'ADMIN_ADD_USER'
ADMIN_REMOVE_USER = 'ADMIN_REMOVE_USER'
USER_UPDATE_NAME = 'USER_UPDATE_NAME'
USER_UPDATE_EMAIL = 'USER_UPDATE_EMAIL'
USER_REMOVE_CONF = 'USER_REMOVE_CONF'

HELLO_MESSAGE = ('Приветствую!\n'
                 'Бот преназначен для управления электроникой и для сбора параметров с датчиков.\n'
                 '/help - помощь');
HELP_UNAUTH_MESSAGE = ('Взамен клавиатуры можно использовать команды:\n'
                       '/start - приветствие\n'
                       '/login - вход, запрос email\n'
                       '/help - помощь\n');
HELP_ADMIN_MESSAGE = ('Взамен клавиатуры можно использовать команды:\n'
                      '/main - меню управления системой включает в себя управление освещением, считывание данных с датчиков и меню статистики\n'
                      '/phase_a - фаза A вкл/выкл\n'
                      '/phase_b - фаза B вкл/выкл\n'
                      '/phase_c - фаза C вкл/выкл\n'
                      '/tmp_in - данные по T внутри\n'
                      '/tmp_out - данные по T снаружи\n'
                      '/humid - данные по влажн. возд.\n'
```

```
'/press - данные по атм. давл.\n'
'/electr - параметры электр. сети.\n'

'\n/stat - меню статистики\n'
'/tmp_in_d - дневн. темп. внутри\n'
'/tmp_out_d - дневн. темп. снаружи\n'
'/tmp_in_w - темп. внутри за нед.\n'
'/tmp_out_w - темп. снаружи за нед.\n'
'/humid_d - дневная влажн.\n'
'/humid_w - влажн. за нед.\n'
'/press_d - дневное атм. давл.\n'
'/press_w - атм. давл. за нед.\n'

'\n/admin - меню админа. Здесь можно изменять количество пользователей и назначать администраторов\n'
'/users - список пользователей\n'
'/add - добавить пользователя\n'
'/remove - удалить пользователя\n'
'/set_admin - назначить админа\n'
'/reset_admin - убрать админа\n'

'\n/profile - меню настройки профиля позволяет сменить имя и email и удалить профиль\n'
'/update_name - изменить имя\n'
```

```
                                '/update_email - изменить email\n'
                                '/remove_profile - удалить профиль\n'

                                '\n/logout - выход\n'
                                '/help - помощь\n');
HELP_USER_MESSAGE = ('Взамен клавиатуры можно использовать команды.\n'
                                '/main - меню управления системой включает в себя управление освещением, считывание данных с датчиков и меню статистики\n'
                                '/phase_a - фаза A вкл/выкл\n'
                                '/phase_b - фаза B вкл/выкл\n'
                                '/phase_c - фаза C вкл/выкл\n'
                                '/tmp_in - данные по T внутри\n'
                                '/tmp_out - данные по T снаружи\n'
                                '/humid - данные по влажн. возд.\n'
                                '/press - данные по атм. давл.\n'

                                '\n/stat - меню статистики\n'
                                '/tmp_in_d - дневн. темп. внутри\n'
                                '/tmp_out_d - дневн. темп. снаружи\n'
                                '/tmp_in_w - темп. внутри за нед.\n'
                                '/tmp_out_w - темп. снаружи за нед.\n'
                                '/humid_d - дневная влажн.\n'
```

```
                '/humid_w - влажн. за нед.\n'
                '/press_d - дневное атм. давл.\n'
                '/press_w - атм. давл. за нед.\n'

                '\n/profile - меню настройки профиля позволяет сменить имя и email и удалить профиль\n'
                '/update_name - изменить имя\n'
                '/update_email - изменить email\n'
                '/remove_profile - удалить профиль\n'

                '\n/logout - выход\n'
                '/help - помощь\n');

markup_unauth = types.ReplyKeyboardMarkup(resize_keyboard=True)
markup_unauth.row('Вход')
markup_unauth.row('Помощь')

markup_admin = types.ReplyKeyboardMarkup(resize_keyboard=True)
markup_admin.row('Управление системой')
markup_admin.row('Операции с пользователями')
markup_admin.row('Настройки профиля')
markup_admin.row('Помощь')
markup_admin.row('Выход')
```

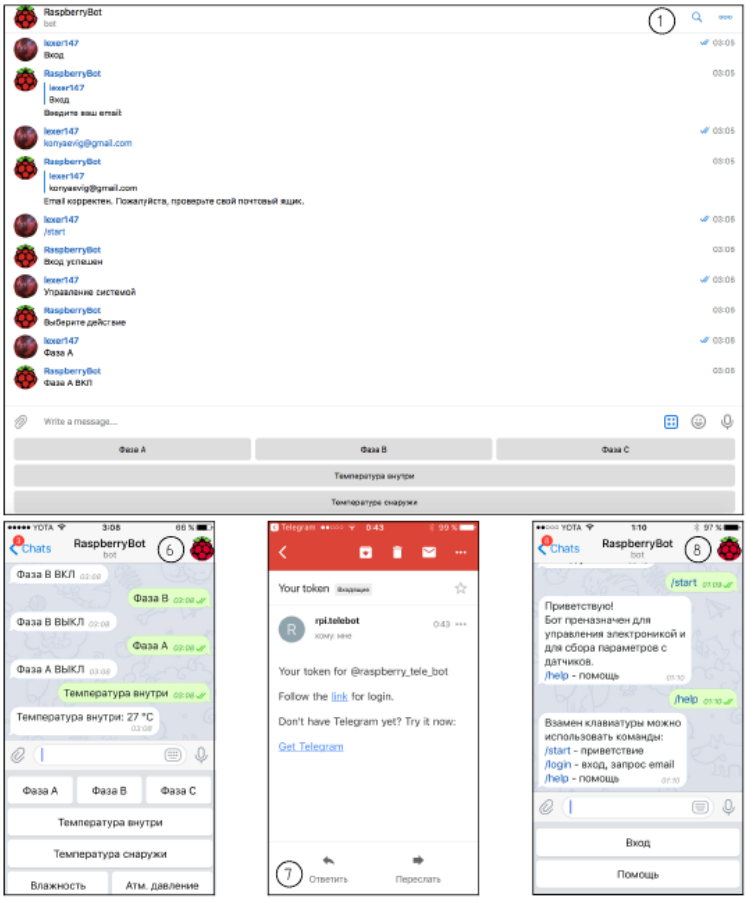
```
markup_admin_ops = types.ReplyKeyboardMarkup(resize_keyboard=True)
markup_admin_ops.row('Список пользователей')
markup_admin_ops.row('Добавить пользователя')
markup_admin_ops.row('Удалить пользователя')
markup_admin_ops.row('Назначить админа')
markup_admin_ops.row('Убрать админа')
markup_admin_ops.row('Закрыть меню админа')

markup_profile_ops = types.ReplyKeyboardMarkup(resize_keyboard=True)
markup_profile_ops.row('Изменить имя')
markup_profile_ops.row('Изменить email')
markup_profile_ops.row('Удалить профиль')
markup_profile_ops.row('Закрыть меню профиля')

markup_user = types.ReplyKeyboardMarkup(resize_keyboard=True)
markup_user.row('Управление системой')
markup_user.row('Настройки профиля')
markup_user.row('Помощь')
markup_user.row('Выход')

markup_main = types.ReplyKeyboardMarkup(resize_keyboard=True)
markup_main.row('Фаза A', 'Фаза B', 'Фаза C')
```

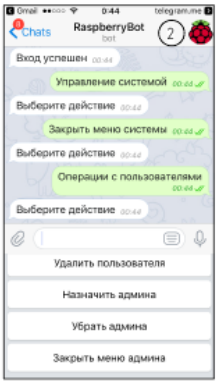
```
markup_main.row('Температура внутри')
markup_main.row('Температура снаружи')
markup_main.row('Влажность', 'Атм. давление')
markup_main.row('Параметры электр. сети')
markup_main.row('Статистика')
markup_main.row('Закрыть меню системы')

markup_stat = types.ReplyKeyboardMarkup(resize_keyboard=True)
markup_stat.row('T°C внутри за сутки')
markup_stat.row('T°C снаружи за сутки')
markup_stat.row('T°C внутри за неделю')
markup_stat.row('T°C снаружи за неделю')
markup_stat.row('Влажность воздуха за сутки')
markup_stat.row('Влажность воздуха за неделю')
markup_stat.row('Атм. давление за сутки')
markup_stat.row('Атм. давление за неделю')
markup_stat.row('Закрыть меню статистики')
def fill_users_with_roles():
        users_chatid_roles = {}
        users = userdao_instance.get_all_with_roles()
        for x in xrange(0,len(users)):
                chat_id = users[x][3]
```

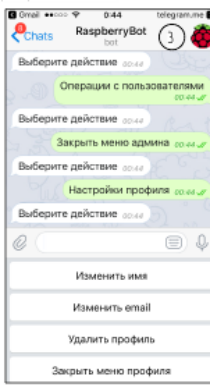
```
                role = users[x][6]
                users_chatid_roles[chat_id] = role
        return users_chatid_roles
def extract_token(text):
        return text.split()[1] if len(text.split()) > 1 else None
```

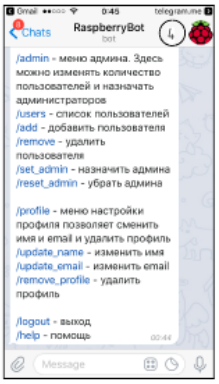
ПРИЛОЖЕНИЕ К

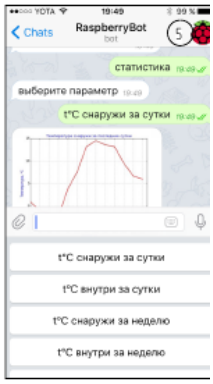
Интерфейс пользователя на ПК и на смартфоне



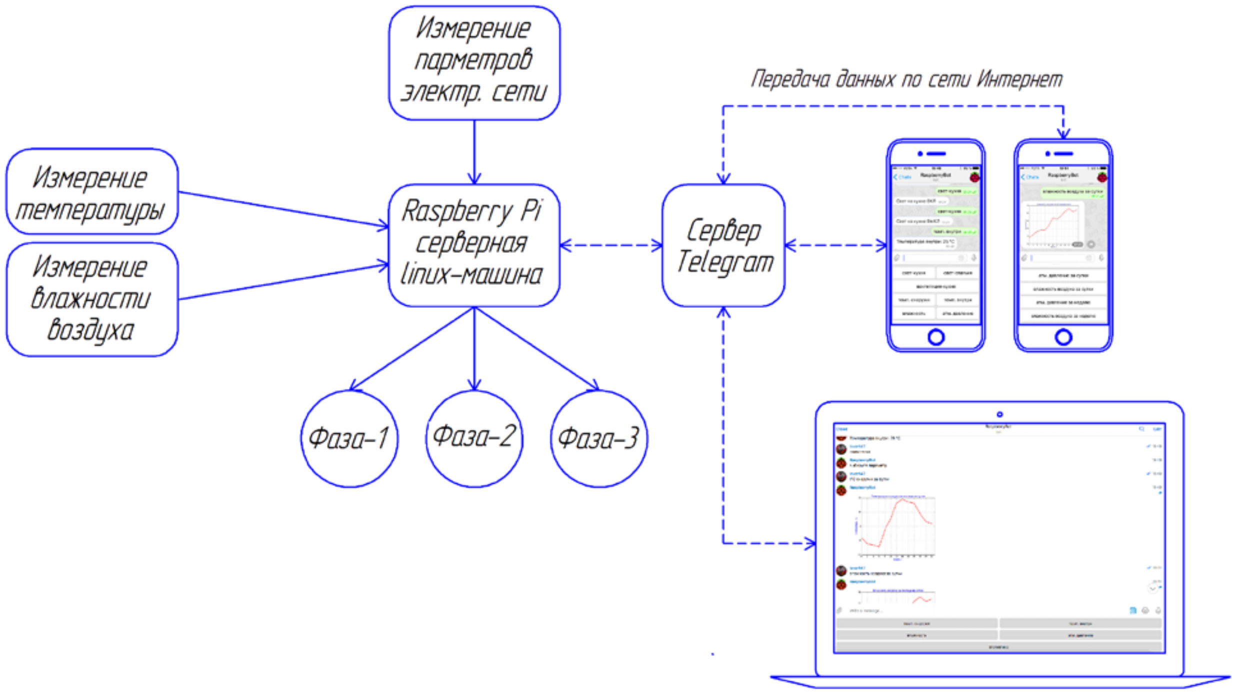








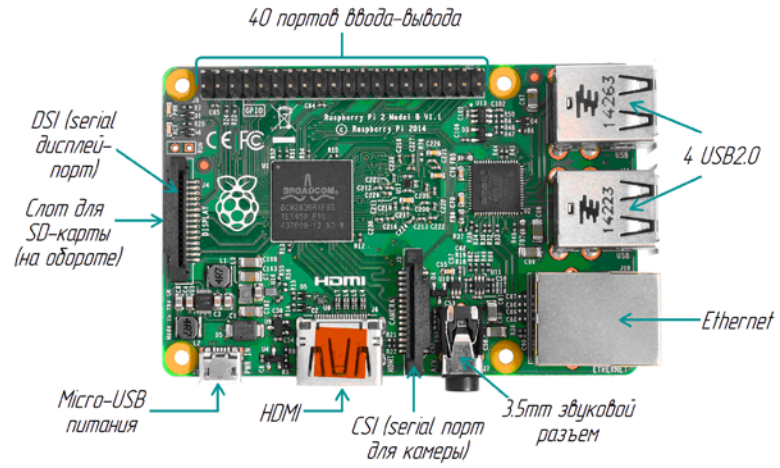
Общая структурная схема системы



Наименование выводов Raspberry Pi

Пин №	Назначение	Назначение	Пин №
01	3.3v DC Power	DC Power 5v	02
03	GPIO-02 (SDA1, I2C)	DC Power 5v	04
05	GPIO-03 (SCL1, I2C)	GROUND	06
07	GPIO-04 (GPIO_GCLK)	(TXD0) GPIO-14	08
09	GROUND	(RXD0) GPIO-15	10
11	GPIO-17 (GPIO_GEN0)	(GPIO_GEN1) GPIO-18	12
13	GPIO-27 (GPIO_GEN2)	GROUND	14
15	GPIO-22 (GPIO_GEN3)	(GPIO_GEN4) GPIO-23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO-24	18
19	GPIO-10 (SPI_MOSI)	GROUND	20
21	GPIO-09 (SPI_MISO)	(GPIO_GEN6) GPIO-25	22
23	GPIO-11 (SPI_CLK)	(SPI_CE0_N) GPIO-08	24
25	GROUND	(SPI_CE1_N) GPIO-07	26
27	ID_SD (I2C ID EEPROM)	(I2C ID EEPROM) ID_SC	28
29	GPIO-05	GROUND	30
31	GPIO-06	GPIO-12	32
33	GPIO-13	GROUND	34
35	GPIO-19	GPIO-16	36
37	GPIO-26	GPIO-20	38
39	GROUND	GPIO-21	40

Внешний вид и основные составляющие Raspberry Pi

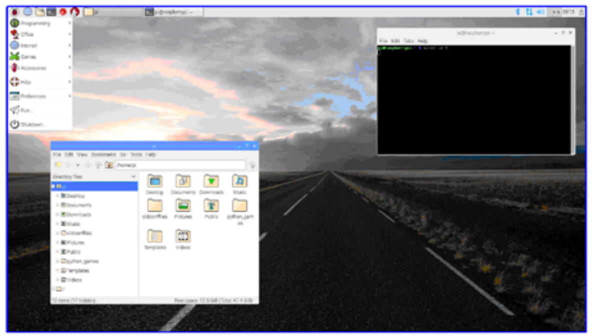


900 Мгц процессор ARM CORTEX A7 (32бит)
Оперативная память – 1 Гб

Основные поддерживаемые операционные системы:

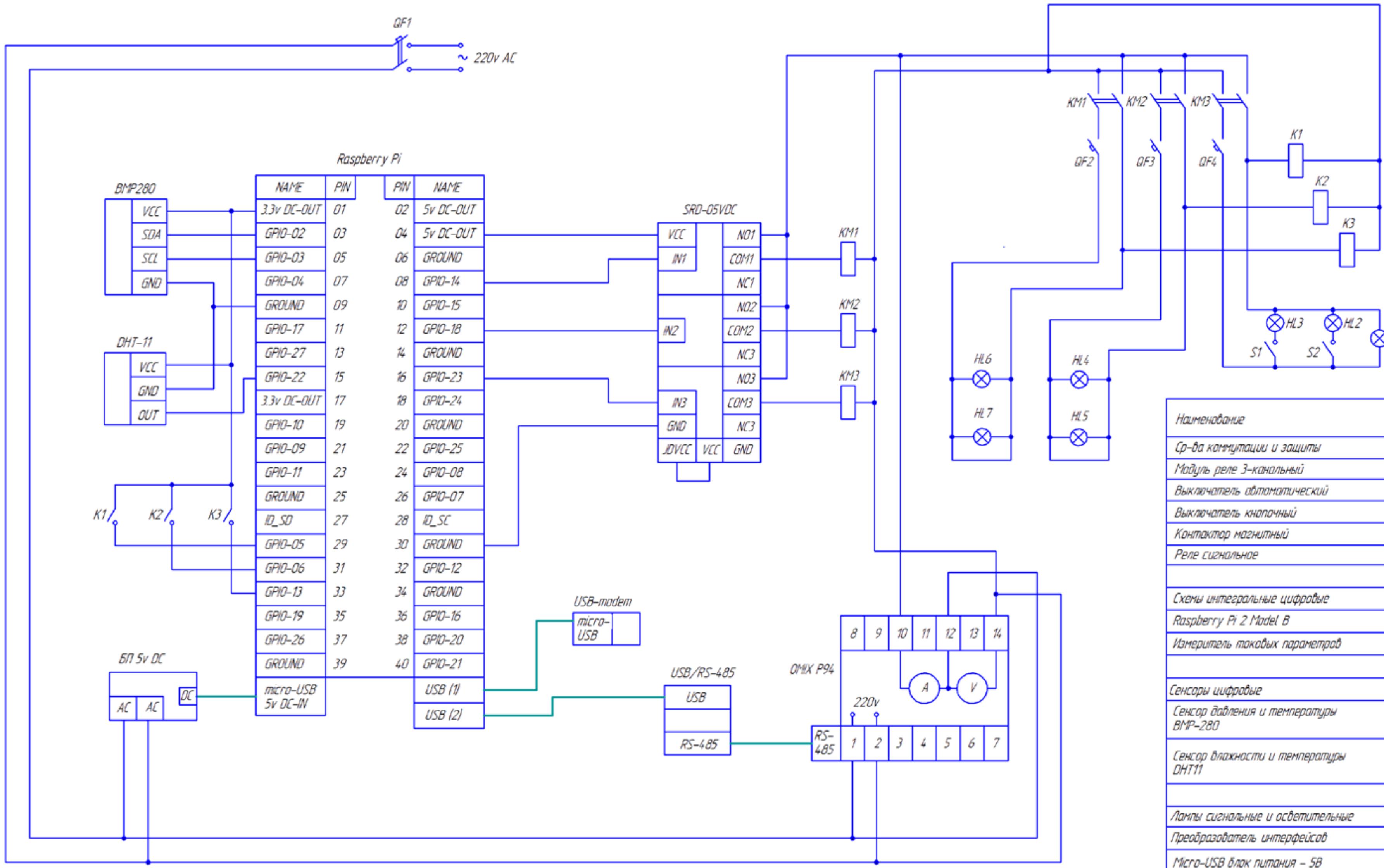
Windows 10 Raspbian Jessie

Демонстрация рабочего стола Raspbian



					ВКР.134008.15.03.04.СХ		
Изм.	Лист	№ докум.	Подп.	Дата	Общая структура. Сведения о платформе	Лит. У	Масса / Масштаб
Разраб.		Коняев И.А.				Лист 1	Листов 7
Пров.		Рыбалев А.Н.					
Т.контр.		Рыбалев А.Н.			Система беспроводного мониторинга на базе GSM-технологии	АмГУ гр 341-об	
Н.контр.		Скрипко О.В.					
Утв.		Остапенко А.А.					

Копировал Формат А1



Наименование		Обозначение на сх.
Ср-ва коммутации и защиты		
Модуль реле 3-канальный	1	SRD-05VDC
Выключатель автоматический	4	QF1-QF4
Выключатель кнопочный	2	S1, S2
Контактор магнитный	3	KM1-KM3
Реле сигнальное	3	K1-K3
Схемы интегральные цифровые		
Raspberry Pi 2 Model B	1	Raspberry Pi
Измеритель токовых параметров	1	OMIX P94
Сенсоры цифровые		
Сенсор давления и температуры BMP-280	1	BMP-280
Сенсор влажности и температуры DHT11	1	DHT-11
Лампы сигнальные и осветительные	7	HL1-HL7
Преобразователь интерфейсов	1	USB/RS-485
Micro-USB блок питания – 5В	1	БП 5v DC
4G USB-модем	1	USB-modem

ВКР.134008.15.03.04.СХ

Изм.	Лист	№ докум.	Подп.	Дата
Разраб.		Коняев И.А.		
Пров.		Рубашев А.Н.		
Т.контр.		Рубашев А.Н.		
Н.контр.		Скрипко О.В.		
Утв.		Остапенко А.А.		

Детальная электрическая схема устройства и спецификация

Лит.	Масса	Масштаб
У		

Лист 2 Листов 7

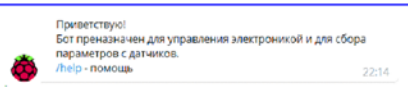
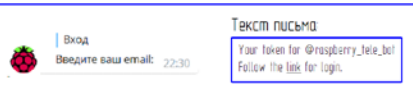
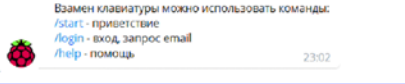
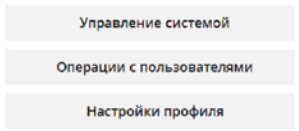
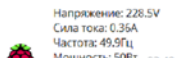
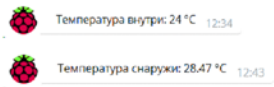
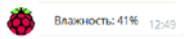
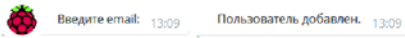
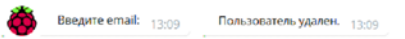
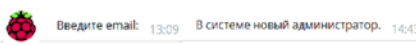
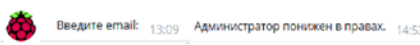
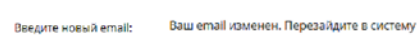
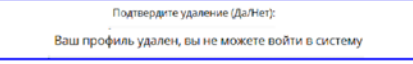
Система беспроводного мониторинга на базе GSM-технологии

АмГУ гр 341-об

Копировал Формат А1

ВКР.134008.15.03.04.ВО

Таблица запросов клиента и ответов сервера

Запрос клиента	Ответ сервера	Пояснение
"/start"		Команда приветствия и авторизации. Авторизация производится по ссылке из email: telegram.me/raspberry_tele_bot?start=token
"/login" – "Вход"		По команде запрашивается email. После проверки email-а в БД на него отправляется письмо с ссылкой для входа в систему.
"/help" – "Помощь"		Команда вызывает справку. Ее содержание отличается в зависмости от пользовательских прав.
"/main" – "Управление системой" "/admin" – "Операции с пользователями" "/profile" – "Настройки профиля"		Команда "/main" открывает доступ к управлению системой (обновляется клавиатура); "/admin" – позволяет войти в меню администратора; "/profile" – раздел пользовательских настроек.
"/phase_a" – "Фаза A" "/phase_b" – "Фаза B" "/phase_c" – "Фаза C"		Включение и выключение фаз A, B, C.
"/electr" – "Параметры электр. сети"		Получение параметров электрической сети.
"/tmp_in" – "Температура внутри" "/tmp_out" – "Температура снаружи"		Получение условных показателей температуры снаружи и внутри помещения.
"/humid" – "Влажность"		Получение влажности воздуха.
"/users" – "Список пользователей"		Команда выводит список пользователей в виде: "Имя" "email" "роль". Администраторский доступ.
"/add" – "Добавить пользователя"		Добавление нового пользователя в базу. Администраторский доступ.
"/remove" – "Удалить пользователя"		Удаление пользователя из базы. Администраторский доступ.
"/set_admin" – "Назначить админа"		Назначение нового администратора по email.
"/reset_admin" – "Убрать админа"		Понижение администратора по email.
"/update_name" – "Изменить имя"		Изменение имени пользователя.
"/update_email" – "Изменить email"		Изменение email-а пользователя.
"/remove_profile" – "Удалить профиль"		Полное удаление профиля с подтверждением.

Продолжение таблицы запросов и ответов (статистика)

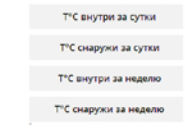
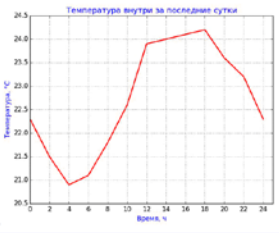
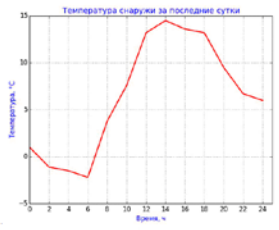
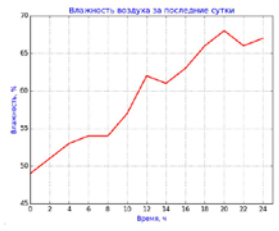
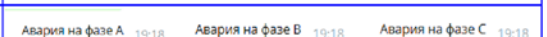
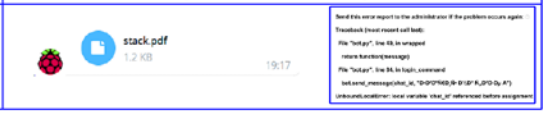
Запрос клиента	Ответ сервера	Пояснение
"/stat" – "Статистика"		Переход в подменю статистики.
"/tmp_in_d" – "T°C внутри за сутки" "/tmp_in_w" – "T°C внутри за неделю"		Команды предназначены для получения температурных графиков за неделю и за сутки снаружи помещения
"/tmp_out_d" – "T°C внутри за сутки" "/tmp_out_w" – "T°C внутри за неделю"		Команды предназначены для получения температурных графиков за неделю и за сутки снаружи помещения
"/humid_d" – "Влажность воздуха за сутки" "/humid_w" – "Влажность воздуха за неделю"		Команды предназначены для получения графиков влажности воздуха за неделю и за сутки

Табл. информирования при возникновении неисправностей

Тип неисправности	Информирование клиента
Авария на фазе A, B или C	
При программной ошибке, связанной с запросом пользователя, отправляется PDF-док. со стеком вызовов	

Изм.	Лист	№ докум.	Подп.	Дата	ВКР.134008.15.03.04.ВО		
Разраб.		Коняев И.А.			Основные команды для запросов к серверу	Лит.	Масса	Масштаб
Пров.		Рыбалев А.Н.				У		
Т.контр.		Рыбалев А.Н.				Лист 3	Листов 7	
Н.контр.		Скшинта П.В.			Системе беспроводного мониторинга на базе GSM-терминалов	АмГУ гр. 341-об		
Утв.		Коваленко А.А.						

Копировал Формат А1

Перв. примен. | Справ. № | Подп. и дата | Инв. № подл. | Подп. и дата | Взам. инв. № | Инв. № дубл. | Подп. и дата

bot.py (main script)

Методы для управления и получения актуальных данных:

Метод (функция)	Действие
phase_a_command(message);	Включение или выключение фазы A.
phase_b_command(message);	Включение или выключение фазы B.
phase_c_command(message);	Включение или выключение фазы C.
tmp_in_command(message);	Получение температуры внутри помещения.
tmp_out_command(message);	Получение температуры снаружи помещения.
humid_command(message);	Получение влажности воздуха.
press_command(message);	Получение атмосферного давления.
electricity_command(message);	Получение параметров электрической сети.

Методы для статистики:

Метод (функция)	Действие
tmp_out_d_command(message);	Температура снаружи за сутки.
tmp_in_d_command(message);	Температура внутри за сутки.
tmp_out_w_command(message);	Температура снаружи за неделю.
tmp_in_w_command(message);	Температура внутри за неделю.
humid_d_command(message);	Суточная влажность.
humid_d_command(message);	Влажность за неделю.

Экземпляры классов:

omix_instance – Объект класса ModbusSerialService();
userdao_instance – Объект класса UserDao();
bme_instance – Объект класса BME280();
dht_instance – Объект класса DHT11();
bot – Объект класса TeleBot();
plot_instance – Объект класса PlotService();
mail_instance – Объект класса MailService();
GPIO – Библиотечный класс по входам/выходам;

Методы административного контроля, авторизации и пользовательских настроек:

Метод (функция)	Действие
start_command(message);	Авторизации через ссылку из email.
login_command(message);	По команде /login производится запрос email.
email_auth(message);	Проверка email и отправка письма на с актив. ссылкой.
logout_command(message);	Процедура выхода из системы.
add_user_command(message);	По команде /add запрашивается email на добавление.
email_add(message);	Проверка email и добавление пользователя.
remove_user_command(message);	По команде /remove запрашивается email на удаление.
email_remove_user(message);	Проверка email и удаление пользователя.
get_users_command(message);	Получение списка пользователей.
set_admin_command(message);	По команде /set_admin запраш. email для повыш. польз.
email_set_admin(message);	Проверка email и назначение нового админа.
reset_admin_command(message);	По ком. /reset_admin запраш. email для пониж. польз.
email_reset_admin(message);	Проверка email и понижение админа в правах.
update_name_command(message);	По команде /update_name запрашивается новое имя.
name_update_user(message);	Обновление введенного имени пользователя.
update_email_command(message);	По команде /update_email запрашивается новый email.
email_update_user(message);	Обновление введенного email-а пользователя.
remove_profile_command(message);	По команде /remove_profile требуется подтвержд.
conf_remove_user(message);	Обработка подтверждения и удаление пользователя.
help_command(message);	По команде /help вызывается справка.
main_enter_command(message);	По команде /main откр. меню управления системой.
admin_enter_command(message)	По команде /admin откр. меню администрирования.
profile_enter_command(message)	По команде /profile откр. меню настроек профиля.
stat_enter_command(message)	По команде /stat откр. меню статистики.
check_user_auth(chat_id)	Проверка пользователя на авторизованность.
check_user_role(chat_id)	Проверка пользовательской роли.

dht_instance
dht11.py

plot_instance
plots.py

mail_instance
mail.py

bme_instance
BME280.py

userdao_instance
userdao.py

omix_instance
omix.py

Изм.	Лист	№ докум.	Подп.	Дата	ВКР.134008.15.03.04.СХ	
Разраб.		Кенжин И.А.			Структура главного программного модуля	Лит. / Масса / Масштаб
Руков.		Гибадов А.Н.				У
Т.контр.		Гибадов А.Н.				Лист 4 / Листов 7
Н.контр.		Кацитка О.В.			Система беспроводного мониторинга на базе GSM-технологии	АмГУ гр 341-об
Утв.		Останина А.А.			Кацитка	Формат А1

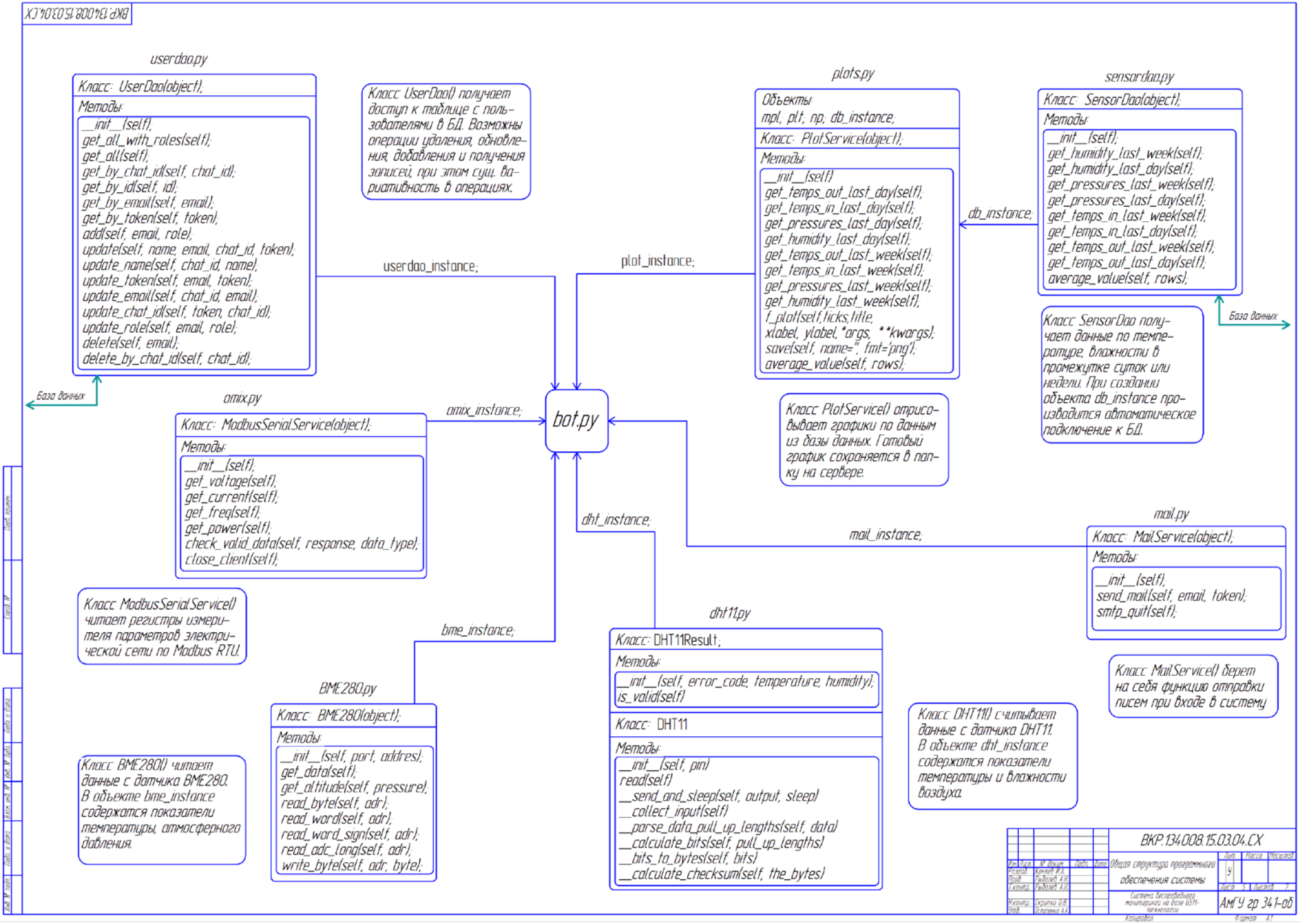

userdao.py
Класс: UserDao(object);
Методы:
__init__(self);
get_all_with_roles(self);
get_all(self);
get_by_chat_id(self, chat_id);
get_by_id(self, id);
get_by_email(self, email);
get_by_token(self, token);
add(self, email, role);
update(self, name, email, chat_id, token);
update_name(self, chat_id, name);
update_token(self, email, token);
update_email(self, chat_id, email);
update_chat_id(self, token, chat_id);
update_role(self, email, role);
delete(self, email);
delete_by_chat_id(self, chat_id);
Класс UserDao() получает доступ к таблице с пользователями в БД. Возможны операции удаления, обновления, добавления и получения записей, при этом сущ. вариативность в операциях.
userdao_instance;
База данных
plots.py
Объекты:
mpl, plt; np; db_instance;
Класс: PlotService(object);
Методы:
__init__(self)
get_temps_out_last_day(self);
get_temps_in_last_day(self);
get_pressures_last_day(self);
get_humidity_last_day(self);
get_temps_out_last_week(self);
get_temps_in_last_week(self);
get_pressures_last_week(self);
get_humidity_last_week(self);
f_plot(self,ticks,title,
xlabel, ylabel, *args, **kwargs);
save(self, name='', fmt='png');
average_value(self, rows);
plot_instance;
db_instance;
sensordao.py
Класс: SensorDao(object);
Методы:
__init__(self);
get_humidity_last_week(self);
get_humidity_last_day(self);
get_pressures_last_week(self);
get_pressures_last_day(self);
get_temps_in_last_week(self);
get_temps_in_last_day(self);
get_temps_out_last_week(self);
get_temps_out_last_day(self);
average_value(self, rows);
База данных
Класс SensorDao получает данные по температуре, влажности в промежутке суток или недели. При создании объекта db_instance производится автоматическое подключение к БД.
Класс PlotService() отрисовывает графики по данным из базы данных. Готовый график сохраняется в папку на сервере.
omix.py
Класс: ModbusSerialService(object);
Методы:
__init__(self);
get_voltage(self);
get_current(self);
get_freq(self);
get_power(self);
check_valid_data(self, response, data_type);
close_client(self);
omix_instance;
bot.py
dht_instance;
mail_instance;
mail.py
Класс: MailService(object);
Методы:
__init__(self);
send_mail(self, email, token);
smtp_quit(self);
Класс MailService() берет на себя функцию отправки писем при входе в систему
Класс ModbusSerialService() читает регистры измерителя параметров электрической сети по Modbus RTU.
bme_instance;
BME280.py
Класс: BME280(object);
Методы:
__init__(self, port, addres);
get_data(self);
get_altitude(self, pressure);
read_byte(self, adr);
read_word(self, adr);
read_word_sign(self, adr);
read_adc_long(self, adr);
write_byte(self, adr, byte);
Класс BME280() читает данные с датчика BME280. В объекте bme_instance содержатся показатели температуры, атмосферного давления.
dht11.py
Класс: DHT11Result;
Методы:
__init__(self, error_code, temperature, humidity);
is_valid(self)
Класс: DHT11
Методы:
__init__(self, pin)
read(self)
__send_and_sleep(self, output, sleep)
__collect_input(self)
__parse_data_pull_up_lengths(self, data)
__calculate_bits(self, pull_up_lengths)
__bits_to_bytes(self, bits)
__calculate_checksum(self, the_bytes)
Класс DHT11() считывает данные с датчика DHT11. В объекте dht_instance содержатся показатели температуры и влажности воздуха.
ВКР.134008.15.03.04.СХ
Общая структура программного обеспечения системы
Система беспроводного мониторинга на базе ESP-технологии
АмГУ гр 341-об

Таблица humid_s

humid_s:

Columns:

```
id SERIAL PRIMARY KEY,
humidity INTEGER NOT NULL,
date_time TIMESTAMP DEFAULT now()
```

Таблица humid_s хранит данные по влажности воздуха. Запись в таблицу производится раз в 2 часа для адекватной выборки

Таблица user_roles

user_roles:

Columns:

```
user_id INTEGER NOT NULL,
role    VARCHAR,
CONSTRAINT user_roles_idx UNIQUE (user_id, role),
FOREIGN KEY (user_id) REFERENCES users (id) ON DELETE CASCADE
```

N

Таблица user_roles хранит пользовательские роли в отношении N к 1. Выдаются по вторичному ключу. Имеют индекс уникальности по id пользователя.

1

Таблица users

users:

Columns:

```
id SERIAL PRIMARY KEY,
name VARCHAR NOT NULL DEFAULT 'NULL',
email VARCHAR NOT NULL,
chat_id VARCHAR NOT NULL DEFAULT 'NULL',
token    VARCHAR NOT NULL DEFAULT 'NULL'
```

sensordao.py

userdao.py

Таблица temps_in

temps_in:

Columns:

```
id SERIAL PRIMARY KEY,
humidity INTEGER NOT NULL,
date_time TIMESTAMP DEFAULT now()
```

Таблица temps_in хранит данные по температуре воздуха внутри помещения. Запись раз в 2 часа.

Таблица temps_out

temps_out:

Columns:

```
id SERIAL PRIMARY KEY,
humidity INTEGER NOT NULL,
date_time TIMESTAMP DEFAULT now()
```

Таблица temps_in хранит данные по температуре воздуха внутри помещения. Запись раз в 2 часа.

Таблица users хранит всю информацию по пользователю. Его имя, chat-id, email и token. Роли в таблице user_role в отношении 1 к N.

Изм.	Лист	№ докум.	Подп.	Дата	
Разраб.		Конягов И.А.			
Пров.		Рубанов А.Н.			
Т.контр.		Рубанов А.Н.			
Н.контр.		Скрипко О.В.			
Утв.		Остроух А.А.			

ВКР.134008.15.03.04.СХ

Структура базы данных

Лит.	Масса	Масштаб
У		

Лист 8	Листов 7

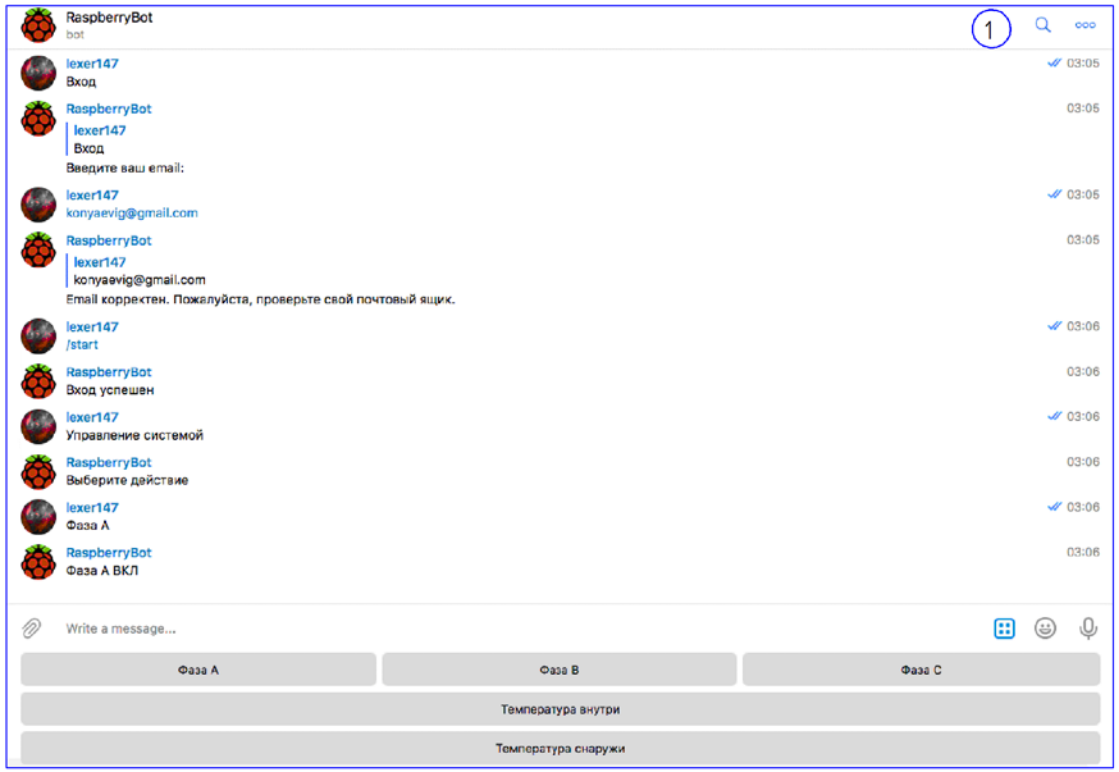
Система беспроводного мониторинга на базе ESP-технологии

АмГУ гр 341-об

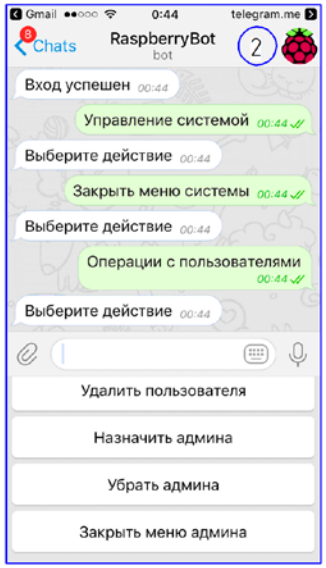
Копировал

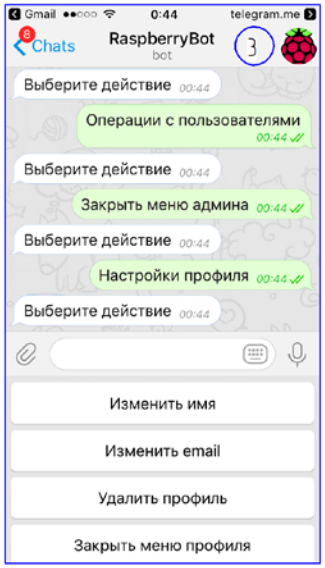
Формат А1

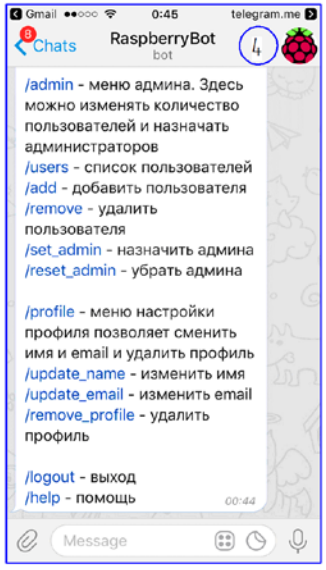
Общий вид на ПК (1)

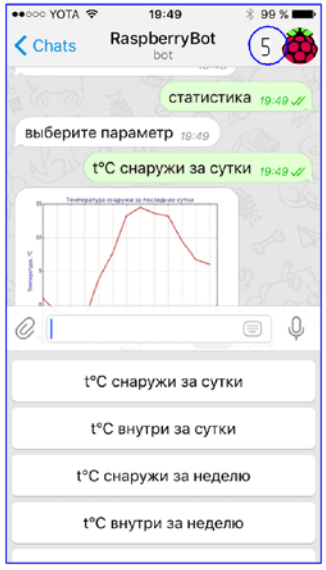


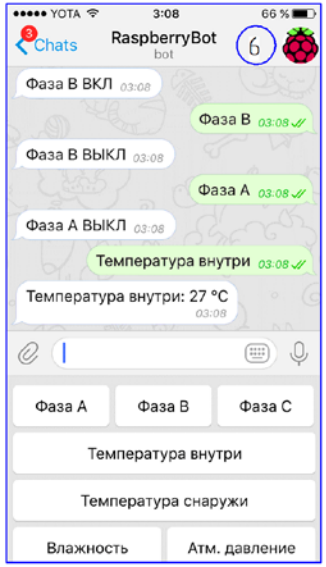
Общий вид на смартфоне (2-8)

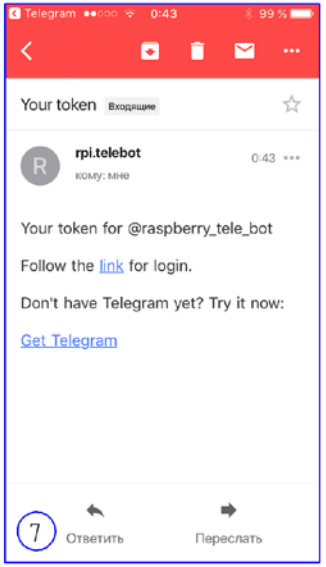


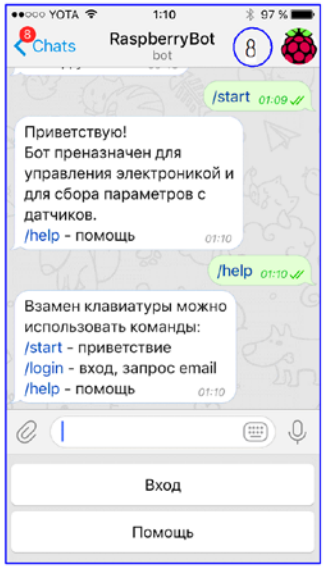












Изм.	Лист	№ докум.	Подп.	Дата	ВКР.134.008.15.03.04.ВО		
Разраб.		Конюев И.А.			Интерфейс пользователя на ПК и смартфоне	Лит.	Масса	Масштаб
Пров.		Рыбалев А.Н.				У		
Т.контр.		Рыбалев А.Н.				Лист 8	Листов 8	
Н.контр.		Куркина О.В.			Система беспроводного мониторинга на базе GSM-технологии	АмГУ гр 341-об		
Утв.		Осауленко А.А.						
Копировал						Формат А1		