# Main server

## Рабочие (основные) модули Главного Компьютера



### Модуль david\_web\_server

|  |  |
| --- | --- |
| Параметр | Описание |
| Файл | david\_web\_server.py |
| Запуск | systemctl start david.service |

#### User Story:

Микроконтроллер отправляет http get запрос через wi-fi на IP адрес главного компьютера.

Модуль david\_web\_server принимает запрос, сохраняет событие в лог файл и сохраняет полученные данные в базу данных.

#### Задача:

1. Принимает http запросы от микроконтроллеров.

2. Сохраняет полученные данные в базу данных.

3. Логирует события в файл.

### Модуль david\_user\_interface

|  |  |
| --- | --- |
| Параметр | Описание |
| Файл | david\_user\_interface.py |
| Запуск | Imported module |

#### User Story:

Различные модули используют методы данного модуля для взаимодействия с пользователем.

В качестве параметров передаются каналы информирования.

Каждый метод возвращает: unsuccessful, successful, confirmed.

unsuccessful - в случае неуспешной попытки проинформировать пользователя по любой причине,

successful - сообщение отправлено, но подтверждение не получено или не предполагается,

confirmed - сообщение отправлено и получено подтверждение о доставке.

Каналы информирования:

а. Голосовое сообщение,

б. Информирование на сайте WEB\_UI,

в. Информирование через GSM канал (вызов, смс),

г. Информирование через мессенджер (WhatsApp и пр.)

#### Задача:

1. Получает задачу на информирование пользователя.

2. Делает попытку проинформировать пользователя.

3. Возвращает результат.

Действия (для логирования):

а. Получает задачу на информирование пользователя: get\_task

б. Делает попытку проинформировать пользоателя: inform\_user\_attempt

в. Возвращает результат: inform\_user\_result

### Модуль Django Математика

Логика:

(user story)

Пользователь заходит на сайт под своим аккаунтом, решает примеры чтобы получить оценку.

Администратор контролирует статистику решений.

(use case)

Main success scenario

Extensions - сценарии ошибок.

Variations - варианты успешных сценариев.

Метод запуска:

Задача:

1.

2.

3.

Действия (для логирования):

а.

б.

в.

### Модуль david\_climate\_check

Файл: david\_climate\_check.py

Задача:

1. Запрашивает климатические данные из базы данных david\_db.sqlite.

2. Проигрывает голосовое оповещение в случае превышения заданных порогов.

3. Логирует события в файл.

Действия (для логирования):

а. Проверяет наличие файлов логов и звуковых сэмплов.

б. Читает базу данных.

в. Проигрывает звуковой файл.

### Модуль david\_currency\_check

Файл: david\_currency\_check.py

Логика:

Если новое значение в процентном отношении больше предыдущего

или новое значение выходит за границы нормальных значений, идет информирование во WA.

Метод запуска:

crontab

Задача:

1. Получает курс валют с сайта ЦБРФ.

2. Записывает полученный курс в базу данных.

3. Проверяет изменения курса валют.

4. Информирует пользователя по WA об изменениях.

Действия (для логирования):

а. Выполняет http запрос.

б. Записывает в базу данных.

в. Читает базу данных. Выполняет проверку полученных данных.

г. Отправляет сообщение в WA.

### Модуль david\_gas\_check

Файл: david\_gas\_check.py

Логика:

Метод запуска:

crontab

Задача:

1. Запрашивает данные измерения датчика газа из базы данных david\_db.sqlite.

2. Проигрывает голосовое оповещение в случае превышения заданных порогов.

3. Логирует события в файл.

Действия (для логирования):

а. Проверяет наличие файлов логов и звуковых сэмплов: check\_file

б. Читает базу данных: get\_data\_from\_db

в. Проигрывает звуковой файл: play\_audio

### Модуль "Список покупок" (проект)

Модуль анализа речи подает сигнал на GPIO порт главного компьютера.

При получении сигнала с GPIO порта активируется программа, которая дает отклик, например: "Слушаю".

Далее по команде "Запиши в список покупок" переключается в подпрограмму записи покупок и выдает ответ: "Записываю".

Далее дается наименование товара.

Модуль произносит услашанное и ждет подтверждения.

Если подтверждение получено, наименование записывается в базу с датой создания и идентификатором списка покупок.

Должна быть реализована возможность закрывать список покупок и получать список покупок в виде речи и в виде сообщения в WhatsApp.

## Вспомогательные модули Главного Компьютера

### Модуль david\_healthcheck

Файл: david\_healthcheck.py

Логика:

Метод запуска:

crontab, запрос пользователя.

Задача:

1. Проверяет наличие файла базы данных: check\_file

2. Запрашивает данные измерения датчиков из базы данных david\_db.sqlite.

3. Запрашивает данные процессорной загрузки, загрузки памяти и температуры CPU (для Raspberry Pi)..

4. Записывает полученные данные процессорной загрузки, загрузки памяти и температуры CPU в лог файл.

5. Готовит информационное сообщение отчет для пользователя.

6. Отправляет информационное сообщение для пользователя на mail.

Действия (для логирования):

а. Проверяет наличие файлов логов, файла базы данных и звуковых сэмплов: check\_file

б. Запрашивает данные измерения датчиков из базы данных david\_db.sqlite: get\_data\_from\_db

в. Запрашивает данные процессорной загрузки, загрузки памяти и температуры CPU: system\_check

г. Готовит информационное сообщение отчет для пользователя: healthcheck\_report

### Модуль david\_unittest

Файл: david\_unittest.py

Метод запуска:

Ручной запуск

Задача:

1. Делает бэкап базы данных.

2. Выполняет тесты.

3. Восстанавливает базу данных из бэкапа.

### Модуль david\_lib

Файл: david\_lib.py

Метод запуска:

Импортируемая библиотека.

Задача:

1. Содержит переменные и функции, используемые другими модулями.

### Модуль david\_db\_create

Файл: david\_db\_create.py

Задача:

1. Создает базу данных david\_db.sqlite.

2. Пересоздает базу данных david\_db.sqlite.

### Модуль voice\_recorder

Файл: voice\_recorder.py

Задача:

Запись голосовых сэмплов для проигрывания на Главном Компьютере.

# Микроконтроллеры

### Микроконтроллер NodeMcu01BedRoom

Файл: NodeMcu01BedRoom.pde

Задача:

1. Считывает данные с датчика DHT22

2. Отправляет данные на Главный Компьютер посредством http get запроса.

Элементная база:

1. NodeMcu,

2. DHT22 датчик температуры и влажности.

Схема подключения:



### Микроконтроллер NodeMcu01BedRoom (ver. 2)

#### Файл:

NodeMcu01BedRoom\_boot.py (boot.py на NodeMCU)

NodeMcu01BedRoom\_main.py (main.py на NodeMCU)

NodeMcu01BedRoom\_ssd1306.py (ssd1306.py на NodeMCU)

#### Логика

При подключении питания выполняется скрипт boot.py.

Этот скрипт выполняет подключение к сети, и отправляет нотификацию на сервер.

Вся информация выводится на экран.

Если подключение к сети прошло успешно, выполняется скрипт main.py.

Этот скрипт каждые 15 минут считывает данные с датчика температуры/влажности и отправляет на сервер.

Информация выводится на экран.

#### Задача

1. Подключается к Wi-Fi сети. И отправляет нотификацию на сервер:

http://192.168.1.44:80/connected;sensor=1&ip=<ip\_addr>&ver=<ver>

1. Выводит информацию о статусе подключения на OLED дисплей.
2. Каждые 15 минут считывает данные с датчика DHT22.
3. Отправляет данные на Главный Компьютер посредством http get запроса:

http://192.168.1.44:80/climate;sensor=1&readattempt=0&temperature=<temp\_val>&humidity=<hum\_val>

1. Выводит информацию о климатических данных на OLED дисплей.

#### Элементная база

|  |  |
| --- | --- |
| **Элемент** | **Кол-во** |
| NodeMcu (sensor\_id=1) | 1 |
| DHT22 датчик температуры и влажности. | 1 |
| OLED дисплей 0.96 inch SSD1306 | 1 |

#### Схема подключения

OLED NodeMcu

SDA <-> D2 (4)

SCL <-> D1 (5)

GND <-> G

VCC <-> 3V

DHT11 NodeMcu

out <-> D3 (0)

- <-> G

+ <-> 3V

### Микроконтроллер NodeMcu02Gas

**Файл:**

NodeMcu02Gas.pde

**Логика**:

В рабочем состоянии горит зеленый светодиод. Модуль считывает данные каждые 5 секунд с датчика газа. Сверяет полученные данные с заданными пороговым значениями. Задано два пороговых значения. В случае превышения первого порогового значения зажигается желтый светодиод. В случае превышения второго порогового значения зажигается красны светодиод, включается звуковой сигнал и отправляется аварийный GET запрос на сервер c параметром type=1.

Каждые 15 минут модуль отправляет данные на сервер с помощью GET запроса с параметром type=0.

**Задача**:

1. Подключается к Wi-Fi сети.

http://192.168.1.44:80/connected;sensor=2&ip=<ip\_addr>&ver=<ver>

1. Считывает данные с датчика газа.
2. Отправляет данные на Главный Компьютер посредством http get запроса.

<http://192.168.1.44:80/gas;sensor=2&sensorValue=666&type=0> – регулярный отчет

<http://192.168.1.44:80/gas;sensor=2&sensorValue=666&type=1> – аварийный отчет

**Элементная база**:

1. NodeMcu,

2. MQ-4 датчик газа.

3. Три светодиода (зеленый, желты, красный) или один трехцветный светодиод.

4. Звуковой сигнализатор.

**Схема подключения**:



### Микроконтроллер NodeMcu02Gas (проект)

#### Файл:

#### Логика

На кухне расположены: датчик движения, датчик газа, датчик температуры, OLED дисплей, светодиодная лента подсветки.

Справа от плиты расположен блок с OLED дисплеем и RGB светодиодом. На дисплей выводится информация температуры, влажности, уровня газа. Светодиод отображает цветом пороги уровня газа.

Слева от вытяжки расположен блок с датчиком движения.

За вытяжкой расположен блок с датчиком газа и датчиком температуры.

Микроконтроллер расположен в коробе вытяжки.

В случае обнаружения движения включается подсветка и горит в течение 5 минут после последнего движения. При отсутствии движения в течение 10 минут и при превышении температуры определенного порога, передается оповещение на главный компьютер, который должен подать сигнал (голосом) о том, что на плите что-то стоит на газу.

В рабочем состоянии горит зеленый светодиод. Модуль считывает данные каждые 5 секунд с датчика газа. Сверяет полученные данные с заданными пороговым значениями. Задано два пороговых значения. В случае превышения первого порогового значения зажигается желтый светодиод. В случае превышения второго порогового значения зажигается красны светодиод, включается звуковой сигнал и отправляется аварийный GET запрос на сервер c параметром type=1.

Каждые 15 минут модуль отправляет данные на сервер с помощью GET запроса с параметром type=0.

Выполняется обновление OLED дисплея и данные сохраняются в файл. Через определенные промежутки времени данные выгружаются из файла в главный компьютер.

#### Задача

1. Подключается к Wi-Fi сети. И отправляет нотификацию на сервер.

http://192.168.1.44:80/connected;sensor=2&ip=<ip\_addr>&ver=<ver>

1. Выводит информацию о статусе подключения на OLED дисплей.
2. Каждые 15 минут считывает данные с датчика DHT22.
3. Выводится информация с датчика DHT22 на OLED дисплей
4. Отправляет климатические данные на Главный Компьютер посредством http get запроса:

http://192.168.1.44:80/climate;sensor=2&readattempt=0&temperature=<temp\_val>&humidity=<hum\_val>

1. Каждые 5 секунд считывает данные с датчика газа MQ-4.
2. Данные с MQ-4 сравниваются с пороговыми и обновляется цвет светодиода.
3. Выводится информация MQ-4 на OLED дисплей.
4. В случае аварийных данных с MQ-4 включается биппер и отправляет аварийный http get запрос.

http://192.168.1.44:80/gas;sensor=2&sensorValue=666&type=1 – аварийный отчет

1. Каждые 15 минут отправляет данные MQ-4 на Главный Компьютер посредством http get запроса.

http://192.168.1.44:80/gas;sensor=2&sensorValue=666&type=0 – регулярный отчет

1. Каждую секунду считываются данные с датчика движения.
2. В случае обнаружения движения включается реле, запускается таймер и загорается диодная лента.
3. При отсутствии движения в течение 5 минут реле выключается.
4. Каждую 1 минуту проверяется температура и считываются время, прошедшее с последнего движения.
5. В случае превышения пороговой температуры и отсутствии движения более 10 минут передается оповещение на Главный Компьютер посредством http get запроса

http get запрос нужно разработать и разработать обработчик этого запроса на Главном компьютере. Главный Компьютер должен подать сигнал (голосом) о том, что на плите что-то стоит на газу.

1. Включается биппер сигнализирующий о том, что на плите что-то стоит на газу.

#### Элементная база

|  |  |
| --- | --- |
| **Элемент** | **Кол-во** |
| NodeMcu (sensor\_id=2) | 1 |
| MQ-4 датчик газа | 1 |
| RGB светодиод | 1 |
| Звуковой сигнализатор | 1 |
| OLED дисплей 0.96 inch SSD1306 | 1 |
| Датчик движения AM312 | 1 |
| Светодиодная лента | 1 |
| Реле | 1 |

#### Схема подключения

### Микроконтроллер NodeMcu03Door (ver. 2)

#### Файл:

NodeMcu03Door.pde

#### Логика

Один NodeMcu обслуживает три датчика движения, датчик освещенности, датчик открывания двери и три реле.

Первый датчик движения расположен перед входной дверью. Он работает совместно с датчиком освещенности. Срабатывает при фиксации движения и при уровне освещенности выше заданного порогового значения. При срабатывании отправляется GET запрос на главный компьютер и переключается реле включения освещения перед входной дверью.

Второй датчик движения расположен между входными дверями. При срабатывании отправляется GET запрос на главный компьютер и переключается реле включения освещения между входными дверями.

Третий датчик движения расположен в помещении за входной дверью. При срабатывании отправляется GET запрос на главный компьютер и переключается реле включения освещения в коридоре.

Датчик открывания двери срабатывает при открывании первой входной двери и отправляет GET запрос на главный компьютер. При этом также подается сигнал на один из pin для включения микрофона или камеры.

Если датчик движения внутри помещения зафиксировал движение после фиксации движения между входными дверями, загорается зеленый светодиод. Если наоборот, то красный. Данная индикация нужна для определения закрыта ли вторая дверь. Светодиод расположен между входными дверями.

#### Задача

#### Элементная база

|  |  |
| --- | --- |
| **Элемент** | **Кол-во, шт.** |
| NodeMcu | 1 |
| Датчик движения AM312 | 3 |
| Датчик освещенности | 1 |
| Реле | 3 |
| Датчик открывания двери | 1 |
| Резистор 100 Ом | 3 |
| RGB LED | 1 |

#### Схема подключения

### Микроконтроллер NodeMcu03Door

Файл: NodeMcu03Door.pde

Задача:

1. Считывает данные с датчика движения.

2. Отправляет события на Главный Компьютер посредством http get запросов.

3. Включает освещение по событиям с датчика движения.

Элементная база:

1. NodeMcu,

2. Датчик движения AM312,

3. Светодиод "желтая линза" 5 мм.,

4. Резистор 100 Ом (2 шт.),

5. Адаптер питания на 5 В.,

6. Блок питания с разъемом 5.5 х 2.1.

Схема подключения:



### Микроконтроллер NodeMcu04Entrance (проект)

Файл:

Логика:

Датчик установлен в коридоре. При обнаружении движения включается светодиод и отправляется сообщение на главный компьютер.

Данные записываются в базу данных и логируются.

Далее идет проверка было ли в пределах 15-ти секунд перед этим срабатывание датчика NodeMcu03Door.

Если было, то через 25 секунд проигрывается приветствие.

В приветствии можно добавить информацию о состоянии системы, климатические данные в квартире и пр. информацию.

Задача:

1.

2.

3.

Элементная база:

1. NodeMcu,

2. Датчик движения AM312,

3. Светодиод "желтая линза" 5 мм.,

4. Резистор 100 Ом (2 шт.),

5. AC-DC преобразователь 5 В.,

Схема подключения:

### Микроконтроллер NodeMcu04ChildrenRoom

#### Файл:

NodeMcu04ChildrenRoom.pde

#### Логика

#### Задача

1. Считывает данные с датчика DHT11
2. Отправляет данные на Главный Компьютер посредством http get запроса.

#### Элементная база

|  |  |
| --- | --- |
| **Элемент** | **Кол-во** |
| ESP01 | 1 |
| DHT11 | 1 |

#### Схема подключения



# Templates

### Модуль Template

|  |  |
| --- | --- |
| Параметр | Описание |
| Файл |  |
| Запуск |  |

#### User Story:

#### Use Case

Main success scenario

Extensions - сценарии ошибок.

Variations - варианты успешных сценариев.

#### Задача:

Действия (для логирования):

а.

### Микроконтроллер Template

#### Файл:

#### Логика

#### Задача

#### Элементная база

|  |  |
| --- | --- |
| **Элемент** | **Кол-во** |
|  |  |

#### Схема подключения