# 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

### Sensors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| S\_ID | C\_ID | Тип | Контроллер | Описание |
| 1 | 1 | DHT22 | NodeMcu01BedRoom | Измерение температуры и влажности в спальне. |
| 2 | 2 | MQ-4 | NodeMcu02Gas | Измерение уровня газа в помещении. |
| 8 | 3 | AM312 | NodeMcu03Door | Определение движения перед входной дверью. |
| 9 | 3 | AM312 | NodeMcu03Door | Определение движения между входными дверями. |
| 10 | 3 | AM312 | NodeMcu03Door | Определение движения в коридоре. |
| 4 | 4 | DHT22 | NodeMcu04ChildrenRoom | Измерение температуры и влажности в детской. |
| 5 | 2 | DHT22 | NodeMcu02Gas | Измерение температуры в кухне. |
| 6 | 2 | AM312 | NodeMcu02Gas | Определение движения на кухне |
| 7 | 2 | DHT22 + AM312 | NodeMcu02Gas | Измерение температуры и определение движения у плиты. Предупреждение о забытых на плите кастрюлях и сковородках. |

S\_ID – Densor ID.

C\_ID – Controller ID.

### 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)

#### Файлы:

Файлы кода микроконтроллера

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | NodeMcu01BedRoom\_boot.py | boot.py для ESP8266 |
| 2 | NodeMcu01BedRoom\_main.py | main.py для ESP8266 |
| 3 | ssd1306.py | Библиотека ssd1306.py для Node ESP8266MCU |
| 4 | david\_pass.json | Файл с паролями |

Файлы моделей корпусов

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | NodeMcu01BedRoomStand.FCStd | Модель FreeCAD корпуса для модуля. |
| 2 | NodeMcu01BedRoomStand.stl | Модель корпуса для модуля stl. |
| 3 | NodeMcu01BedRoomStand.gcode | Модель корпуса для модуля gcode. |

#### 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 их на сервер.

Информация выводится на экран.

#### Задача

1. Подключается к Wi-Fi сети. И отправляет нотификацию на сервер посредством http get запроса:

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** | **Комментарий** |
| 1 | ESP8266 | 1 |  |
| 2 | Плата - переходник ESP8266 | 1 | https://roboshop.spb.ru/board-esp8266 |
| 3 | Стабилизатор напряжения AMS1117-3.3 | 1 | https://roboshop.spb.ru/AMS1117-3-3 |
| 4 | DHT22 датчик температуры и влажности. | 1 |  |
| 5 | OLED дисплей 0.96 inch SSD1306 | 1 |  |
| 6 | Micro-USB | 1 | https://roboshop.spb.ru/micro-USB-female-board |
| 7 | Блок питания с Micro-USB | 1 |  |

#### Схема подключения



### 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 (ver. 2)

#### Файлы:

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | david\_pass.json | Файл с паролями |
| 2 | NodeMcu02BedRoom\_boot.py | boot.py на NodeMCU |
| 3 | NodeMcu02Gas\_main.py | main.py на NodeMCU |
| 4 | ssd1306.py | ssd1306.py на NodeMCU |

#### Таймеры

|  |  |  |
| --- | --- | --- |
| № | Описание | Интервал, сек. |
| 1 | Regular climate and gas sensors report interval to server | 900 |
| 2 | Gas check interval | 5 |
| 3 | Check motion sensor interval | 1 |
| 4 | Delay for the light to switch off | 300 |
| 5 | Delay to check the oven | 10 |
| 6 | No motion and high temperature near oven emergency delay | 600 |

#### 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 расположены блоки с датчиком газа и датчиком температуры в разных корпусах.

Микроконтроллер расположен в коробе вытяжки.

В случае обнаружения движения включается подсветка и горит в течение времени, определенном в таймере №4 (Delay for the light to switch off).

При отсутствии движения в течение таймера №6 (No motion and high temperature near oven emergency delay) 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 на блоке с OLED дисплеем горит зеленый светодиод. Модуль считывает данные через интервал, определенный таймером №2 (Gas check interval) с датчика газа. Сверяет полученные данные с заданным пороговым значением. В случае превышения порогового значения зажигается красны светодиод и отправляется аварийный GET запрос на сервер c параметром type=1.

Каждый промежуток времени, определенный таймером №1 (Regular climate and gas sensors report interval to server) модуль отправляет данные на сервер с помощью GET запроса с параметром type=0.

Информация на дисплее 16 символов, 5 строк (каждый символ 8 Х 10 пикселей):

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 (8) | 2 (16) | 3 (24) | 4 (32) | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 | T | e | m | p | . | : |  | X | X | X | X |  | C |  | •  (sent to server) |  |
| 20 | H | u | m | . | : |  |  | X | X | X | X |  | % |  | •  (sent to server) |  |
| 30 | G | a | s | : |  |  | X | X | X | X | X |  | % |  | •  (sent to server) |  |
| 40 |  | •  (Motion detected) |  | S | e | c | . | : |  | X | X | X | X |  |  |  |
| 60 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

#### 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. Через интервал Т№1 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. Через интервал Т№2 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. Через интервал Т№1 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. Через интервал Т№2 считываются данные с датчика движения.
2. В случае обнаружения движения включается реле, запускается таймер и загорается диодная лента.
3. Отправляется http get запрос на Главный Компьютер:

http://192.168.1.44:80/motion;sensor=6

1. При отсутствии движения в течение Т№4 реле выключается.
2. Через интервал Т№5 проверяется температура и считываются время, прошедшее с последнего движения.
3. В случае превышения пороговой температуры и отсутствии движения более Т№6 передается оповещение на Главный Компьютер посредством http get запроса

http://192.168.1.44:80/oven;sensor=4&temperature=666&type=1 – аварийный отчет. Главный Компьютер должен подать сигнал (голосом) о том, что на плите что-то стоит на газу.

#### Элементная база

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Элемент** | **Кол-во** | **Питание, В** | **Управление, В** | **Ток, А** |
| NodeMcu (sensor\_id=2) | 1 |  |  |  |
| MQ-4 датчик газа | 1 | 5 |  | 0.19 |
| DHT22 | 1 |  |  | 0.0003 |
| RGB светодиод | 1 |  |  |  |
| OLED дисплей 0.96 inch SSD1306 | 1 |  |  | 0.015 |
| Датчик движения AM312 | 1 |  |  | 0.0001 |
| Светодиодная лента | 1 |  |  |  |
| Реле (JQC-3FF-S-Z 5VDC) | 1 |  |  | 0.07 |

#### Схемы подключения

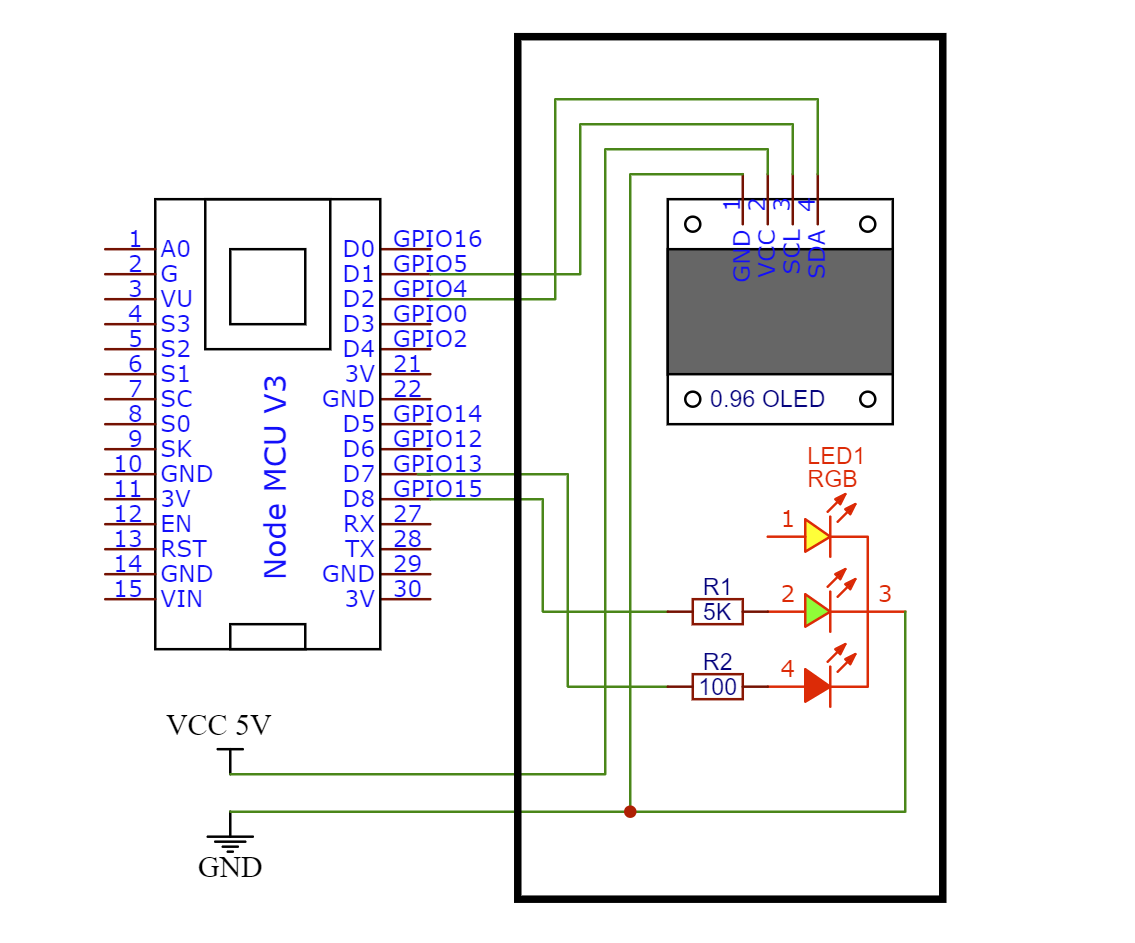


Figure 1. Блок с OLED дисплеем

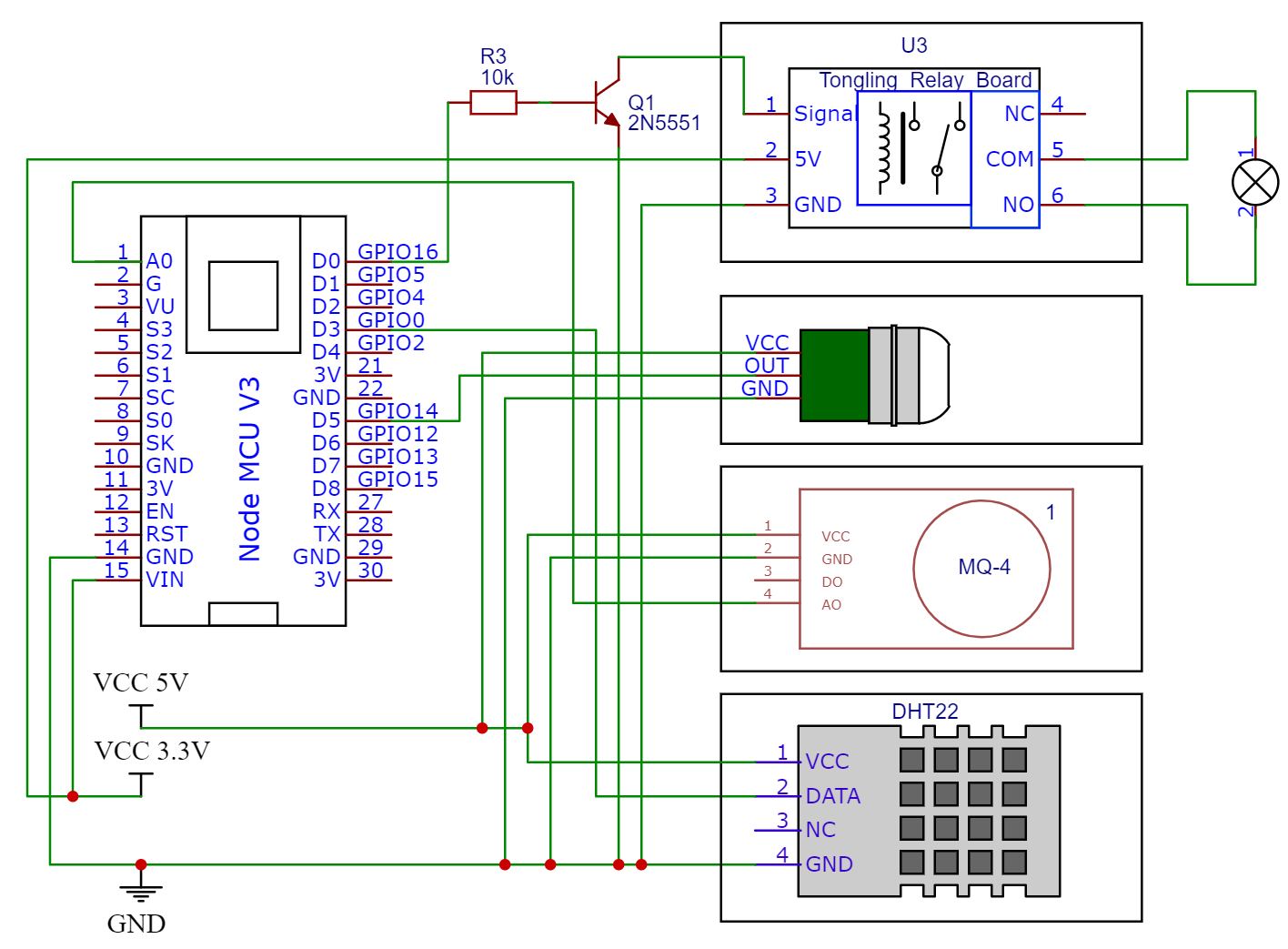


Figure 2. Блоки датчиков

#### Корпусы

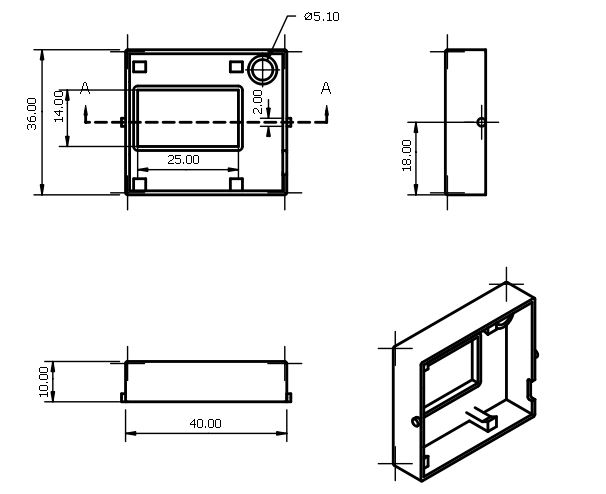


Рисунок 1. Корпус Oled дисплея

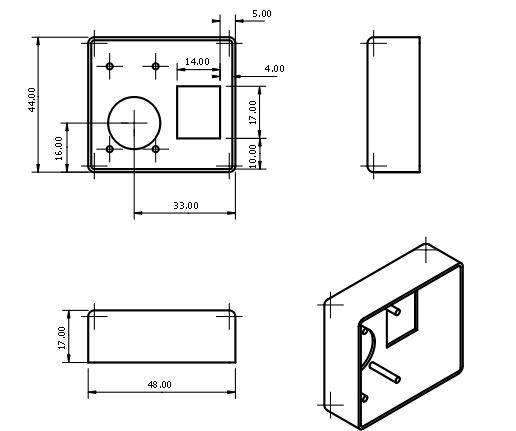


Рисунок 2. Корпус датчиков DHT и MQ-4

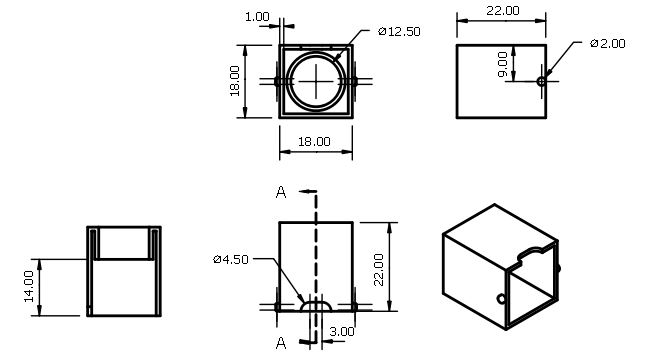


Рисунок 3. Корпус датчика движения

#### Список тестов

|  |  |  |
| --- | --- | --- |
| № | Описание | Результат |
|  | При включении питания на дисплее отобразилась надпись:  «Connecting to  <название\_сети> by WiFi...» | ОК |
|  | После подключения к сети WiFi на дисплее отобразилась надпись:  «Connected:  <ip\_address>» | ОК |
|  | При включении питания выполнилась синхронизация по времени и на дисплее отобразилась надпись:  «Time set:  YYYY-MM-DD  HH:MI:SS»  Если синхронизация не выполнилась, появилась надпись:  «Time not set.» |  |
|  | При включении питания отправлена нотификация на сервер:  http://192.168.1.44:80/connected;sensor=5&ip=<ip\_addr>&ver=<ver> | ОК |
|  | При включении питания после успешной отправки нотификации на сервер на дисплее отобразилась надпись:  «Server response  Status: 200» | ОК |
|  | При включении питания, в случае неуспешной отправки нотификации на сервер (сервер не поднят), на дисплее отобразилась надпись:  «Server response  Status: <код\_ошибки>» | ОК |
|  | Отсутствие Wi-Fi сети при старте контроллера (на этапе boot.py). |  |
|  | Пропадание Wi-Fi сети во время работы контроллера (на этапе main.py). |  |
|  | Отсутствие сервера при старте контроллера (на этапе boot.py). |  |
|  | Пропадание сервера во время работы контроллера (на этапе main.py). |  |
|  | Каждые 15 минут обновляются данные температуры и влажности на дисплее. | ОК |
|  | Каждые 15 минут отправляется нотификация на сервер с данными температуры и влажности:  http://192.168.1.44:80/climate;sensor=2&readattempt=<att>&temperature=<temp\_val>&humidity=<hum\_val> | ОК |
|  | Каждые 15 минут отправляется нотификация на сервер с данными температуры и влажности.  В случае успешной отправки нотификации на сервер, напротив значений температуры и влажности на дисплее отображается точка. | ОК |
|  | Каждые 15 минут отправляется нотификация на сервер с данными температуры и влажности.  В случае неуспешной отправки нотификации на сервер, напротив значений температуры и влажности на дисплее не отображается точка. | ОК |
|  | Каждые 15 минут отправляется нотификация на сервер с данными датчика газа type=0:  http://192.168.1.44:80/gas;sensor=2&sensorValue=<gaz\_val>&type=0 | ОК |
|  | Каждые 5 секунд обновляются данные датчика газа на дисплее. | ОК |
|  | Светодиод индикации газа обновляется каждые 5 секунд и загорается красным в случае превышения первого и второго порогового значения. | ОК |
|  | Buzzer начинает пищать в случае превышения второго порогового значения датчика газа. Время обновления – 5 секунд. | ОК |
|  | В случае превышения второго порогового значения датчика газа на сервер каждые 5 секунд отправляется нотификация type=1:  http://192.168.1.44:80/gas;sensor=2&sensorValue=<gaz\_val>&type=1 | ОК |
|  | Каждую секунду считываются данные с датчика движения и обновляется информация на дисплее в виде точки в нижней строке в случае обнаружения движения. | ОК |
|  | В случае обнаружения движения включается реле освещения. | ОК |
|  | В случае обнаружения движения отправляется http get запрос на Главный Компьютер:  http://192.168.1.44:80/motion;sensor=6 |  |
|  | В случае отсутствия движения в течение 5 минут реле освещения отключается. | ОК |
|  | Каждую минуту проверяется температура и проверяется время, прошедшее с последнего обнаружения движения. В случае превышения пороговой температуры и отсутствия движения более 10 минут передается нотификация на главный компьютер type=1:  http://192.168.1.44:80/oven;sensor=7&temperature=<t\_val>&type=1 | ОК |
|  | Каждую минуту проверяется температура и проверяется время, прошедшее с последнего обнаружения движения. В случае превышения пороговой температуры и отсутствия движения более 10 минут включается Buzzer.  При обнаружении движения Buzzer выключается. | ОК |
|  | Если датчик освещенности показывает значение выше порогового, то освещение не включается. | N/A |
|  | На экране отображается кол-во секунд, оставшихся до отключения освещения. | N/A |

### 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 NodeMcu03Door (ver. 2)

#### Остановились на

Писать программу.

#### Файлы:

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 | david\_pass.json | Файл с паролями |
| 2 | NodeMcu03Door\_boot.py | boot.py на NodeMCU |
| 3 | NodeMcu03Door\_main.py | main.py на NodeMCU |

#### Таймеры

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Обозначение | Интервал, сек. |
| 1 | Интервал выключения освещения перед входной дверью и отправки http GET запроса на Главный Компьютер. | t\_sw1 | 30 |
| 2 | Интервал выключения освещения между входными дверями и отправки http GET запроса на Главный Компьютер. | t\_sw2 | 30 |
| 3 | Интервал выключения освещения в коридоре и отправки http GET запроса на Главный Компьютер. | t\_sw3 | 60 |

#### Логика

Один микроконтроллер обслуживает три датчика движения, датчик освещенности 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 включается реле включения освещения перед входной дверью. При этом также подается сигнал на pin для включения камеры.

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

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

#### Элементная база

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Элемент** | **Кол-во** | **Питание, В.** | **Управление, В.** | **Ток, А** | **Комментарий** |
|  | ESP-12E (sensor\_id=3) | 1 | 5 / 3.3 | 3.3 | 0.5 |  |
|  | Датчик движения AM312 | 3 | 5 / 3.3 | 3.3 | 0.0001 |  |
|  | Фоторезистор LDR GL5539 | 1 | - | 3.3 |  |  |
|  | MOSFET FQP30N06L | 3 |  |  |  |  |
|  | Лампа или диодная лента | 3 | 12 |  |  |  |
|  | Блок питания 12В | 1 | 220 |  |  |  |

#### Схема подключения

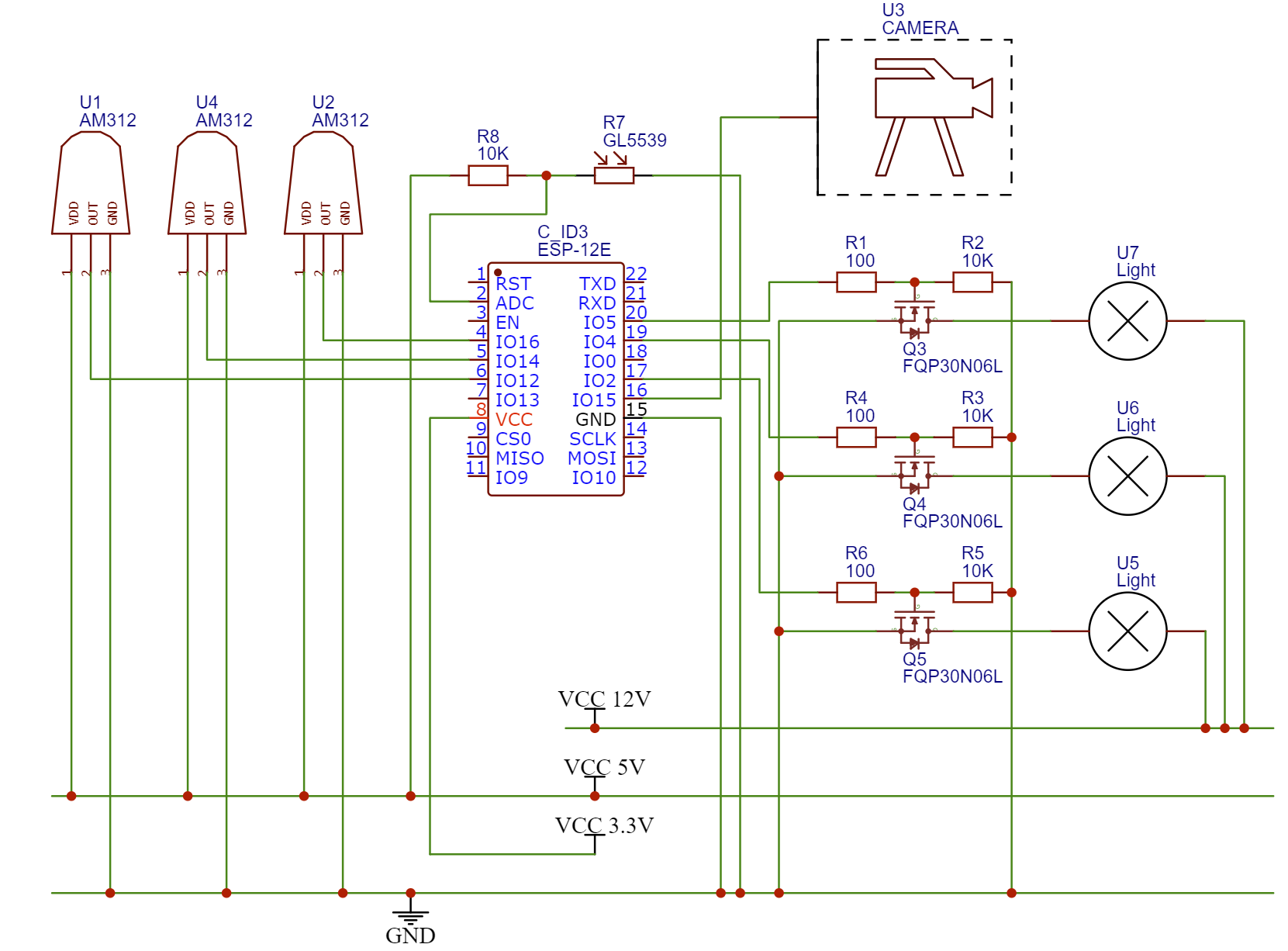


Figure 3. NodeMcu03Door

#### Корпус

#### Список тестов

Результаты тестирования версии 200814

Основные кейсы

|  |  |  |
| --- | --- | --- |
| № | Описание | Результат |
|  | При включении питания отправлена нотификация на сервер:  http://192.168.1.44:80/connected;sensor=3&ip=<ip\_addr>&ver=<ver> | OK |
|  | В случае обнаружения движения перед входной дверью и при уровне освещенности ниже порогового включается реле освещения на t\_sw1 секунд. | OK |
|  | В случае обнаружения движения перед входной дверью и при уровне освещенности ниже порогового отправляется http get запрос на Главный Компьютер: http://192.168.1.44:80/motion;sensor=8  Запрос отправляется каждые t\_sw1 секунд при наличии движения. | OK |
|  | В случае обнаружения движения перед входной дверью и при уровне освещенности ниже порогового подается активный сигнал на Pin cam. Активный сигнал сохраняется t\_sw1 секунд. | OK |
|  | В случае обнаружения движения перед входной дверью и при уровне освещенности выше порогового реле освещения не включается.  Проверка осуществляется каждые t\_sw1 секунд. | OK |
|  | В случае обнаружения движения перед входной дверью и при уровне освещенности выше порогового отправляется http get запрос на Главный Компьютер: http://192.168.1.44:80/motion;sensor=8  Запрос отправляется каждые t\_sw1 секунд при наличии движения. | OK |
|  | В случае обнаружения движения перед входной дверью и при уровне освещенности выше порогового подается активный сигнал на Pin cam. Активный сигнал сохраняется t\_sw1 секунд. | OK |
|  | В случае обнаружения движения между входными дверями включается реле освещения на t\_sw2 секунд. | OK |
|  | В случае обнаружения движения между входными дверями отправляется http get запрос на Главный Компьютер: http://192.168.1.44:80/motion;sensor=9  Запрос отправляется каждые t\_sw2 секунд при наличии движения. | OK |
|  | В случае обнаружения движения в коридоре включается реле освещения на t\_sw3 секунд. | OK |
|  | В случае обнаружения движения в коридоре отправляется http get запрос на Главный Компьютер: http://192.168.1.44:80/motion;sensor=10  Запрос отправляется каждые t\_sw3 секунд при наличии движения. | OK |
|  |  |  |

Дополнительные кейсы

|  |  |  |
| --- | --- | --- |
| № | Описание | Результат |
|  | Отсутствие Wi-Fi сети при старте контроллера (на этапе boot.py) |  |
|  | Пропадание Wi-Fi сети во время работы контроллера (на этапе main.py) |  |
|  | Отсутствие сервера при старте контроллера (на этапе boot.py) |  |
|  | Пропадание сервера во время работы контроллера (на этапе main.py) |  |
|  |  |  |

### 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



# Проблемы

## Проблема №1. Зависание модуля NodeMcu02Gas (ver. 2)

Модуль зависает в районе 23:00. Модуль зависает только при подключении на кухне.

Это повторялось 3 раза. Замена модуля не помогла. Модуль с тестовой программой тоже завис. При этом модуль с базовой прошивкой не завис в лабораторных условиях.

Отличия лабораторных условий от базовых.

1. В лабе датчик DHT22.
2. В лабе установлено реле.
3. В лабе для питания контроллера используется USB.
4. В прод экран подключен через длинный кабель.

# 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

#### Файлы:

Файлы кода микроконтроллера

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 |  |  |
| 2 |  |  |

Файлы моделей корпусов

|  |  |  |
| --- | --- | --- |
| № | Название | Описание |
| 1 |  |  |
| 2 |  |  |

#### Таймеры

|  |  |  |  |
| --- | --- | --- | --- |
| № | Описание | Обозначение | Интервал, сек. |
| 1 |  |  |  |
| 2 |  |  |  |

#### Логика

#### Задача

Obsolete. Использовать раздел «Список тестов».

#### Элементная база

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **№** | **Элемент** | **Кол-во** | **Питание, В.** | **Управление, В.** | **Ток, А** | **Комментарий** |
|  |  |  |  |  |  |  |

#### Схема подключения

#### Корпус

#### Список тестов

Основные кейсы

|  |  |  |
| --- | --- | --- |
| № | Описание | Результат |
|  |  |  |

Дополнительные кейсы

|  |  |  |
| --- | --- | --- |
| № | Описание | Результат |
|  |  |  |