## С чего начиналось

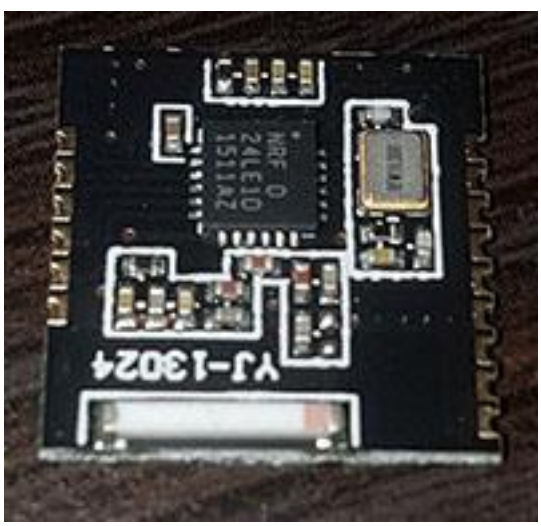
В свое время заменил дома выключатели на livolo и maifom с управлением по RF 433MHz. Настроил управление через Broadlink и программу e-control . Потом перешел на iobroker с драйвером Broadlink. Система понравилась и работало все нормально, но без обратной связи это всего лишь универсальный пульт управления, трудно поддающийся автоматизации. Поискал готовые решения с обратной связью. Выключатели с Wi-Fi на базе ESP8266 не подходили, так как всем им необходим еще нулевой провод для питания. Появились уже, правда, zigbee выключатели livolo без нулевого провода со своим шлюзом со своим облаком. То есть был вариант заменить только выключали, оставив стекла. Но на тот момент шлюз livolo еще не соединили с брокером. Да и покупать еще раз все выключатели с неясными на тот момент перспективами интеграции в систему умного дома не хотелось. Проще, быстрее и дешевле было переделать имеющиеся.

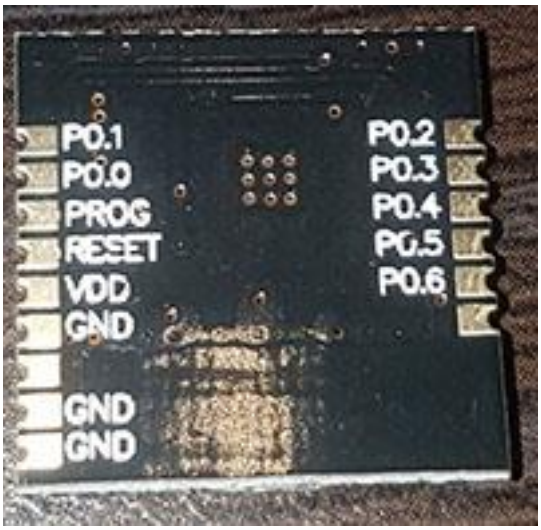
## Выключатели

Поискал по интернету варианты переделки livolo без отдельного питания по нулевому проводу. Есть варианты на ESP8266, NRF24LO1 + AVR/STM, NRF5. Но все они требуют доработки блока питания как минимум. Далее или дополнительной печатной платы (NRF24LO1 + AVR/STM), или замены основной платы процессора (NRF5). В livolo используется микроконтроллер PIC16F690(0.22ma@1MHz), наверняка работающий на минимально необходимой для выполнения штатных функций частоте и вообще использующийся по максимуму. А его замена на другой процессор с приемопередающим модулем помимо необходимости изготовления новой платы приведет к кратному увеличению энергопотребления. Например, до 5 ma в лучшем случае тут: https://www.openhardware.io/view/486/NRF51822-Livolo-2-channel-1-way-EU-switchVL-C700X-1_Ver_C2

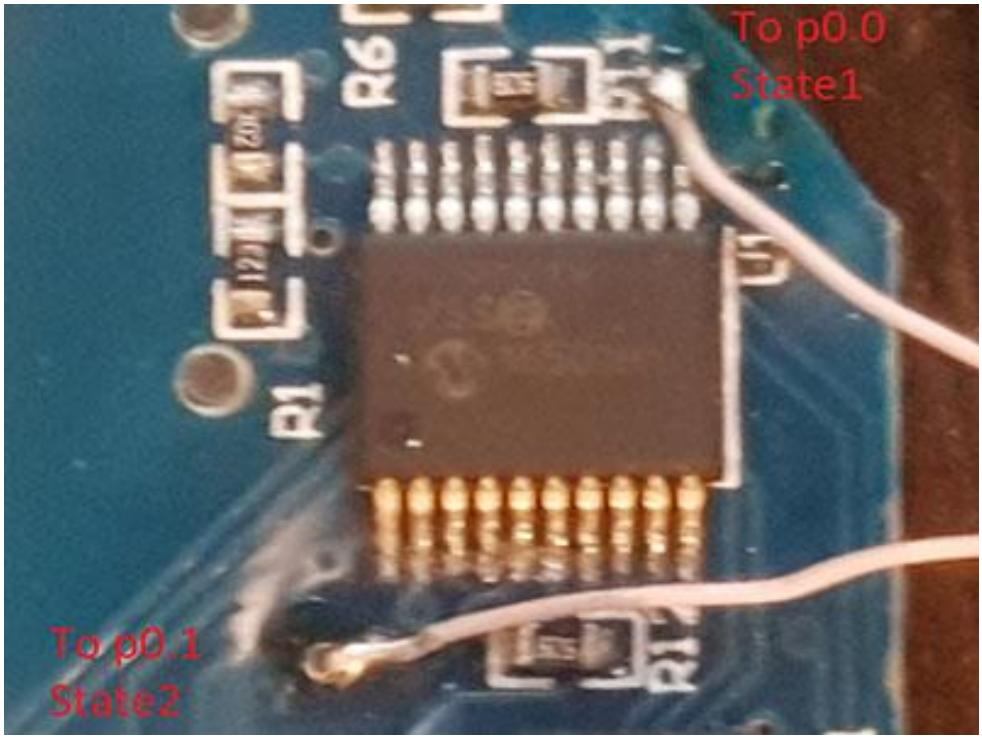
Все эти варианты довольно трудоемкие, часть их отрывочно документирована или не документирована вовсе. А потому решил сделать еще один с нуля. Как мне кажется, выгоднее не трогать родной PIC процессор, а приемо-передающую часть выполнить в виде отдельного контроллера, работающего в активном режиме меньшую часть времени и спящего большую его часть. В этом случае выключатель сохранит все штатные функции, то есть этот вариант пригоден и для проходных выключателей. Считывать состояния выключателей 1 и 2 можно с выходов на светодиоды, это выводы 11 и 5 PIC. Управлять выключателями можно, генерируя коды команд и подавая их на вход, предназначенный для подключения RF приемника. Штатный RF 433MHz приемник придется удалить с платы. Приемопередающие модули лучше использовать диапазона 2.4 GHz. Самый популярный из них NRF24LO1 от Nordic. При максимальной мощности передатчика 1мВт (0dBm) и максимальной чувствительности приемника -94dBm на скорости 250kbps модуль NRF24LO1 требует при приеме/передаче соответственно 12.4ma и 11.1ma. Обычно его используют с контроллером AVR или STM серий. Вариант неплохой, но нужно все это собирать на отдельной печатной плате. Мне же понравилась микросхема NRF24LE1, тоже от Nordic. Это тот же модуль NRF24LO1 плюс встроенный микроконтроллер с быстрым(~8x) 8051 ядром (MCU) в одном корпусе. С программированием 8051 я сталкивался, нужно только было найти программатор. Конечно, по современным меркам ресурсы контроллера не впечатляют (16MHz MCU, 16кбайт flash памяти и 256байт + 2кбайт ОЗУ), но он довольно экономичен (заявлено потребление MCU 2.5ma на частоте 8MHz), имеет неплохую периферию и много разных режимов работы, включая несколько режимов сна. Этот контроллер используется довольно редко. Я встречал только примеры беспроводных датчиков с батарейным

питанием. Как мне кажется, это связано с тем, что используемое абсолютно во всех конструкциях программирование с подключением SDK от Nordic не позволяет эффективно использовать все возможности контроллера. Чтобы в этом убедиться, достаточно дизассемблировать полученный таким образом код. Особенно поразило меня использование целых процедур(!) вместо одной(!) команды при простом битовом вводе и выводе. Однако, если использовать от Nordic только родную документацию (https://infocenter.nordicsemi.com/pdf/nRF24LE1_PS_v1.6.pdf) и заявленную совместимость контроллера с архитектурой 8051, а также ассемблер для этого семейства, можно получить быстрый и компактный код (например, 4.9кбайт для шлюза и 2.6кбайт для выключателя). А наличие различных режимов работы с разной тактовой частотой позволяет «обменять» быстрый код на пропорциональное снижение производительности и потребления MCU. У меня NRF24LE1 в выключателе работает на частоте 1MHz. Этого вполне достаточно для работы с радио. Для уменьшения потребления приемопередатчик (1мВт -94dBm @250kbps) включается два раза в секунду до 33 ms каждый для приема команд и передаче состояния, а общее время работы процессора в активном режиме примерно 100 ms каждую секунду. Остальное время процессор находится в режиме сна, кроме случаев, когда нужно передать код на выключатель. Код передается до срабатывания выключателя до 8 раз в каждом полусекундном окне, но процессор работает при этом на минимальной частоте 125kHz. Перед переходом в режим сна каждые 8 секунд программа считывает данные с датчика температуры DS18B20 при его наличии. Такой режим работы дает среднее потребление около 1ma при максимальной мощности передатчика и максимальной чувствительности приемника. Это заметно лучше, чем по ссылке выше, и позволяет работать даже без переделок в блоке питания выключателя со светодиодными лампами от 5-10 Вт. И только при записи во флешь память документация на контроллер требует работы на максимальной частоте 16 MHz без сна, и соответственно, увеличивается энергопотребление. Чтобы не допустить ошибок при записи флешь памяти, программа следит, чтобы во время обновления был включен хотя бы один выключатель, тогда его блок питания способен отдать в нагрузку больший ток. Минусом такого режима работы является наличие задержки в 1-2 секунды при приеме и исполнении команд. Микроконтроллер NRF24LE1 выпускается в 3-х разных корпусах, с 24, 32 и 48 выводами. Есть также OTP и Flash версии микросхемы. Я ориентировался на возможности Flash версии, так как она позволяет оперативно менять прошивку. И я использовал микросхему на готовой плате. Плата обычно включает в себя не только саму микросхему, но и антенну с кварцем и сопутствующими элементами. Найти ее в самом компактном исполнении можно, например, здесь: https://ru.aliexpress.com/item/32665457728.html?spm=a2g0s.9042311.0.0.4de033edB2Yt4h

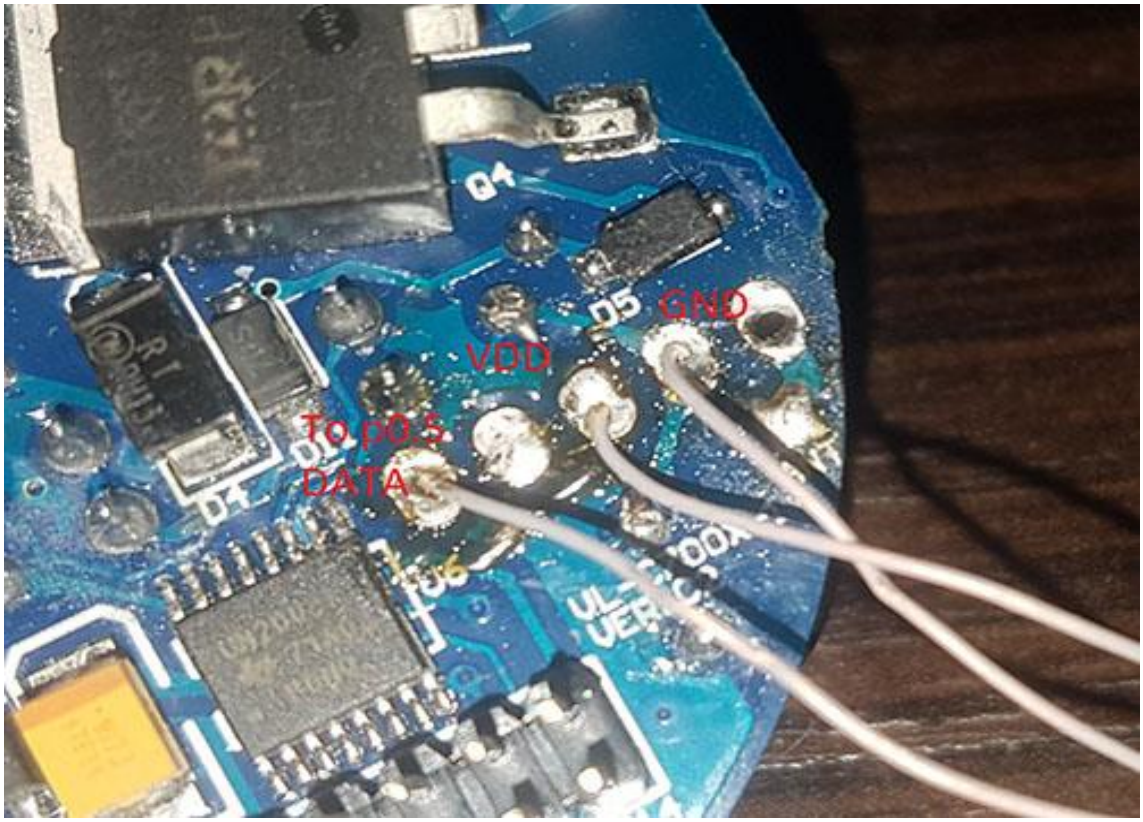


Эта плата с NRF24LE1 по размеру лишь немного больше NRF24LO1. Точки подключения для мониторинга состояния выключателей:



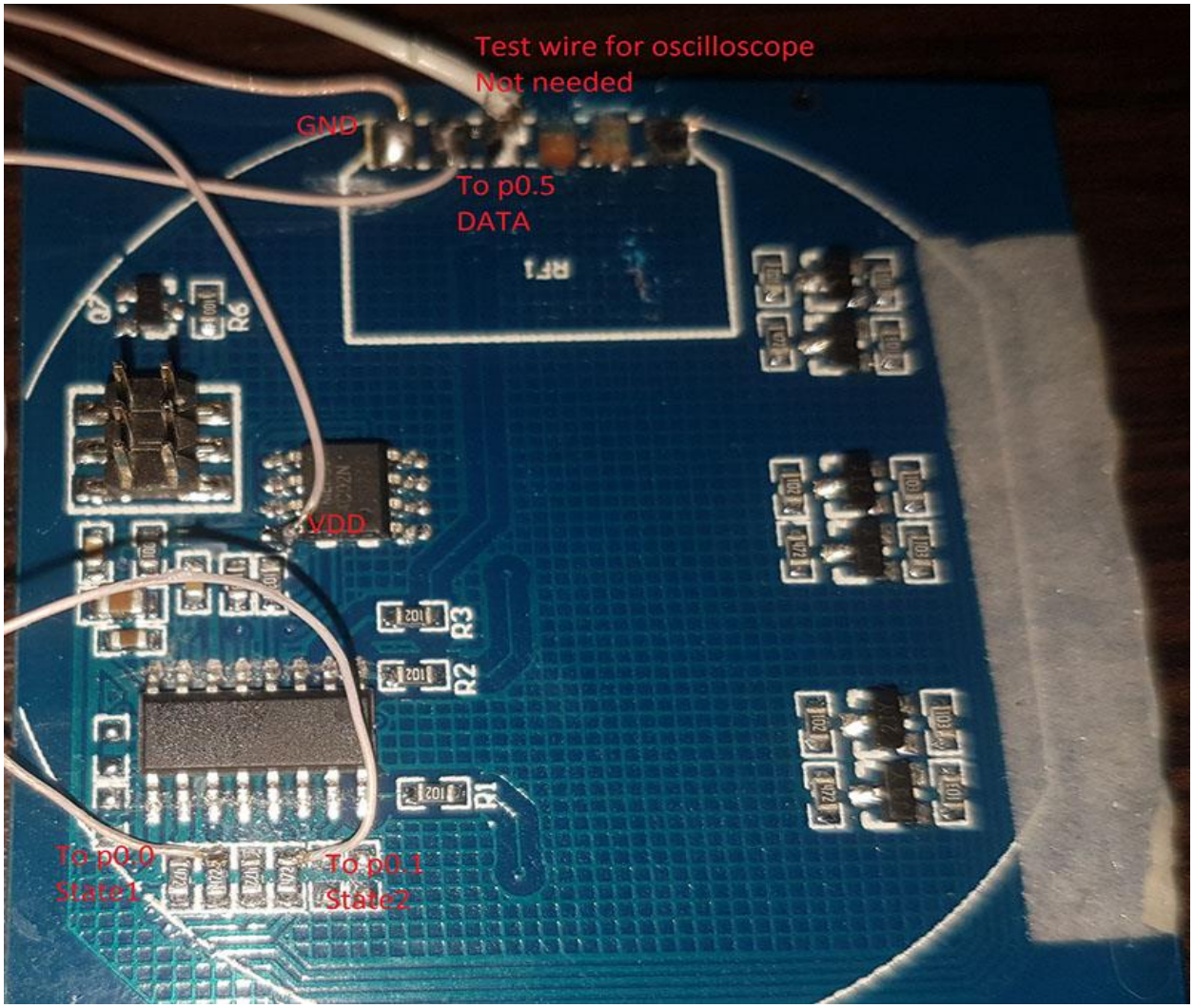

Точки подключения для питания NRF24LE1 и управления выключателем:




Для упрощения конструкции вывод DER разрешения приемника не используется, а управляющий код передается до срабатывания выключателя, максимум 4 секунды. За это время как минимум 4 раза выключатель должен включить приемник и принять код. Программируем плату NRF24LE1 при помощи программатора (об этом ниже) и

припаиваем 5 проводов (NRF gnd <-> RF gnd, NRF vdd <-> RF +3v, NRF p0.5(control) <-> RF DATA, NRF p0.0(state1) <-> PIC 11(LED), NRF p0.1(state2) <-> PIC 5(LED)) от NRF24LE1 к выключателю. После чего собираем выключатель, допиливаем паз в металлическом креплении и, обернув плату скотчем, фиксируем ее тем же скотчем как, чтобы антенна немного выступала над креплением. Для уменьшения пульсаций я все же меняю емкость в блоке питания livolo  по 12в с 330uF*25V на 1000uF*25V. Готовая конструкция у меня выглядит так:


Для проверки протокола rcswitch  подобным образом переделал  несколько выключателей maifom. Похожую конструкцию имеют и выключатели Vhome. Точки подключения:



Тут также нужно аккуратно снять RF приемник. Стоит отметить, что, в отличие от livolo, где приемник включается кратковременно, а выключенный приемник на выходе имеет низкий уровень, в этих выключателях приемник работает постоянно, а потому на входе контроллера всегда присутствует шум эфира, что контроллер выключателя воспринимает как высокий уровень. Это нужно учитывать при программировании NRF24LE1, в противном случае выключатель не воспринимает приходящий от нее код и не обучается, хотя и честно пищит при приходе команды. При переделке других моделей выключателей желательно уточнить этот момент, используя осциллограф, и, при необходимости скорректировать программу. Состояния выключателей можно снимать с выходов контроллера, идущих через разъем на оптроны управления симисторами, это 11 и 13 выводы для 1 и 2 канала соответственно, а питание и управление с контактов RF приемника, как и в livolo. У меня снять приемник получилось не очень, отвалилась площадка +3v. Дорожки не самая большая проблема, при выпаивании приемника в одном выключателе перестали нормально работать сенсоры, похоже, что не хватило аккуратности, что-то повредил.

В блоке питания выключателя я тоже увеличивал емкость, стоящую перед 3-х вольтовым стабилизатором HT7130, с 220uF*16v до 2000uF*16v. Я ставил 2 конденсатора по 1000uF, больше не влезло. И тем не менее, при подключении NRF24LE1, в отличие от ливоло, где напряжение перед стабилизатором 10-12 вольт, в их блоках питания в выключенном состоянии напряжение перед питающим процессор 3-х вольтовым стабилизатором может иногда опускаться до 3.5 вольт и даже менее. При этом стабилизатор выходит из режима стабилизации, и помехи из сети могут привести к самопроизвольному включению выключателя. Во включенном же состоянии там обычно от 5 до 9 вольт. Возможно, это зависит от нагрузки, но 3 таких выключателя иногда включались, а 4 у меня работают более полугода без нареканий. Блок питания сильно не копал, может быть можно мелкими переделками устранить этот эффект. Двойные выключатели maifom, как, возможно, и другие симисторные модели, изначально имеют одну особенность: при включении только одной половины светодиодные лампы иногда заметно мерцают. У ливоло подобного не замечал.

О программе. Она написана на ассемблере 8051 и занимает в памяти до 3 кбайт. Может считывать состояния 2-х выключателей (выводы p0.0 и p0.1) и управлять ими, используя выход p0.5 контроллера для генерации команд по протоколу livolo или rcwitch (это maifom, vhome и т.п.). Для управления используются 2 кода на переключение, по одному на каждый выключатель. Код посылаются до срабатывания выключателя и только тогда, когда текущее состояние выключателя не соответствует заданному. Естественно, для нормальной работы нужно будет обучить выключатель этим кодам. Процедура ничем не отличается от обучения выключателя по RF. Возможно также управление другой техникой, которая включается и выключается одной кнопкой, например, кондиционером. Для этого используются выходы p0.2 и p0.3, на которые подается импульс на переключение первого и второго канала соответственно. Полярность импульсов можно корректировать в прошивке. И в каждом конкретном случае нужен будет дополнительный стабилизатор на 3.3 в и схема согласования уровней для управления и чтения состояния. Ну и для контроля температуры можно подключить к выходу p0.4 датчик DS18B20. Для нормальной работы шины программа включает внутренний резистор с p0.4 на 3.3в, а потому внешний резистор не нужен. На сервер передается 4 параметра. Первые 2 отражают состояние выключателей, 3-й параметр – это температура (-0.0 при отсутствии датчика), и 4-й – напряжение питания NRF24LE1, до 3.59в. Поддерживается обновление прошивки по радиоканалу.

Для компиляции программы я использую Telemark Cross Assembler. Найти его можно здесь: http://old-dos.ru/index.php?page=files&mode=files&do=show&id=1385

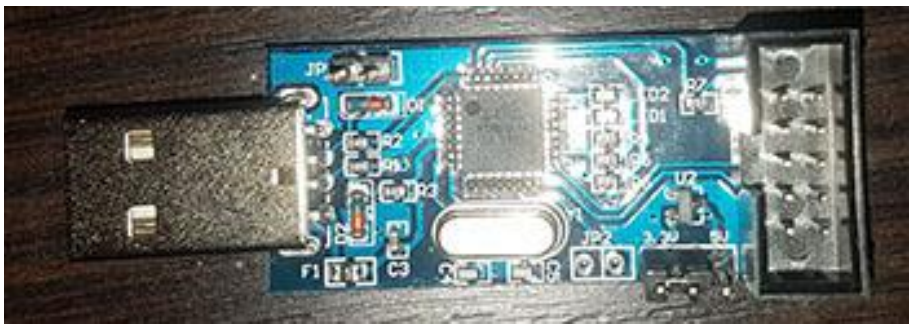
Для каждого устройства перед компиляцией нужно установить несколько основных параметров. Это номер канала, в исходнике RFCH, диапазон значений 0 … 7Dh(0 … 125), а также адрес устройства (RRFID), для каждого устройства он свой, значения 0 … 0Fh (0…15), до 16 устройств на один канал и шлюз. Я рекомендую использовать каналы в диапазоне 100-125(64h – 7Dh), это частоты 2500-2525MHz. Они расположены выше частот Wi-Fi, zigbee и ble. Также нужно выбрать протокол управления выключателем (RCPRT =0-livolo,RCPRT <>0 – rcswitch) и формат вывода в MQTT состояний выключателей (STVAL=0 - 0/1, STVAL=0feh - on/off, STVAL=0ffh - true/false). После установки параметров прошивку надо скомпилировать. Для Telemark Assembler, команда для получения бинарного файла выглядит так:

tasm.exe -51 -b -fff NRFCLV00.A51

Если нужен Intel Hex формат:

tasm.exe -51 –g0 -fff NRFCL00.A51

Далее нужно полученный файл прошить в микросхему. Для этого нужен специальный программатор. Отсутствие такого программатора, как мне кажется, и есть главная причина прохладного отношения к контроллеру NRF24LE1. Я работал с двумя разными программаторами, и у каждого есть свои плюсы и минусы. Первый их них перешитый USBASP на ATMega8. Его фото и назначение контактов разъема:



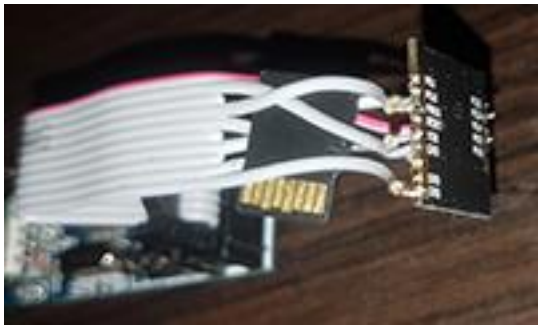
| | |
|---|---|
| 1. FMOSI | 2. VDD (+3.3) |
| 3 — | 4. PROG |
| 5. FCSN | 6 — |
| 7. FSCK | 8 — |
| 9. FMISO | 10. GND |

Программа и подробности по прошивке USBASP для программирования NRF24LE1 тут: http://homes-smart.ru/index.php/oborudovanie/bez-provodov-2-4-ggts/55-programmirovanie-nrf24le1-cherez-usbasp.

Нужен USBASP именно на ATMega8, только для нее я нашел прошивку на этом сайте. Я использовал 24 пиновый корпус (24LE1D) в выключателе и 32 пиновый (24LE1E) в шлюзе. Подключение к разным вариантам корпусов NRF24LE1:

| | 24 pin-4×4 | 32 pin-5×5 | 48 pin-7×7 |
|---|---|---|---|
| FCSN | P0.5 | P1.1 | P2.0 |
| FMISO | P0.4 | P1.0 | P1.6 |
| FMOSI | P0.3 | P0.7 | P1.5 |
| FSCK | P0.2 | P0.5 | P1.2 |

Подключение NRF24LE1 в корпусе с 24 выводами (с SD картой лучше видно, что и куда припаяно ☺):




Этот программатор принимает бинарный файл. К плюсам можно отнести возможность работы контроллера в программаторе сразу после прошивки. К минусам 5 вольтовые уровни на SPI шине, даже при установленной на 3.3v перемычке. При подключении программатора с контроллером к USB высокий уровень с выходов SPI через защитные диоды попадает на питание NRF24LE1 и, в случае с прошивкой для выключателя, из-за низкого потребления повышает питание до 3.7 и выше. У меня был один случай выхода из строя контроллера с КЗ по питанию при длительной работе в таком режиме. После первого же обращения программы к программатору выставляются низкие уровни по SPI и питание возвращается к 3.3 вольтам. Есть еще на этом же сайте проект по прошивке NRF24LE1 при помощи Raspberry PI: http://homes-smart.ru/index.php/oborudovanie/bez-provodov-2-4-ggts/54-programmirovanie-nrf24le1-cherez-raspberry-pi. Тоже пробовал, но вариант с прошивкой по USB мне показался удобнее.

Можно найти на алиэкспрессе программатор специально для NRF24LE1: https://ru.aliexpress.com/item/32459650560.html?spm=a2g0s.9042311.0.0.4de033edB2Yt4h


Заказывал его в 2-х разных местах, и один из продавцов прислал архив с программой. Она была на китайском языке, но оказалось, что это просто китайская версия программы от Nordic, а английскую версию я нашел тут: http://nic.vajn.icu/PDF/wireless/Nordic/nRFFlasherV1.20b/. Прошить бинарный файл этим программатором у меня не получилось, но файл формата Intel HEX он принимает нормально. Уровни в нем 3.3в, но после программирования питание с NRF24LE1 снимается. Для работы микросхему придется отсоединять от программатора и подключать к другому источнику питания.

Программатор нужен только один раз для каждой микросхемы. В дальнейшем обновлять прошивки микросхем можно будет через шлюз по Wi-Fi.

# Шлюз

Управлять выключателями удобнее всего по протоколу MQTT. Все системы умного дома или имеют в своем составе сервер MQTT, или умеют с ним работать. Для связи с MQTT сервером необходим шлюз, лучше Wi-Fi. Учитывая небольшую мощность NRF24, порядка 1мВт (это в 100 раз меньше мощности Wi-Fi роутера), для обеспечения надежной связи с выключателями может потребоваться несколько шлюзов, как минимум по одному на каждый этаж в многоэтажном доме. И лучшее место установки шлюза за подвесным потолком в коридоре или холле, где сходится большинство комнат и, соответственно, выключателей в доме. И меньше предметов и людей будет на линии шлюз – выключатели.

Кратко по протоколу работы шлюза с контроллерами NRF24LE1 в выключателях (далее для краткости клиентами). Обмен между шлюзом и клиентами идет 24 байтовыми пакетами. Каждый клиент имеет свой адрес от 0 до 15, а приемник клиента настроен принимать пакет от шлюза только тогда, когда адрес в пакете совпадает с адресом клиента. Шлюз по очереди опрашивает состояние каждого клиента, посылая ему адресный пакет. Пакет передается примерно 1ms, потом шлюз включает приемник и ждет ответ в течение 9ms. Если ответа нет, то этот цикл повторяется до 127 раз. По времени это будет примерно 1.1 секунды, то есть клиент за это время должен 2 раза проснуться и послушать эфир в течение 11ms. При приеме запроса клиент посылает ответ и включает приемник на 9 ms. Если за это время ничего не принято, пакет считается успешно отправленным, и клиент переходит в режим сна. Если шлюз за 1.1 секунду не получил ответ от клиента, то увеличивается счетчик неуспешных попыток, и по достижении 15 неуспешных попыток связь с клиентом считается утерянной. И на сервере MQTT во всех параметрах этого клиента появится строка “NoRFL”. После чего шлюз переходит к опросу следующего клиента. Если от сервера MQTT приходит команда, то она передается нужному клиенту в следующем же цикле. Для того, чтобы не слать запросы на все 16 клиентов и ждать от каждого ответ в течение 1.1 секунды, шлюз должен знать адреса реально подключенных устройств. Для этого используется таблица имен параметров. Устройств может быть 16, в каждом по 4 параметра, то есть в таблице всего 64 имени. Таблица хранится в памяти шлюза, а точнее в памяти его NRF24LE1. Шлюз опрашивает клиента, если имя хотя бы одного его параметра есть в этой таблице. Это сильно сокращает время опроса всех устройств. На сервер MQTT также передаются только те параметры, имена которых есть в таблице.

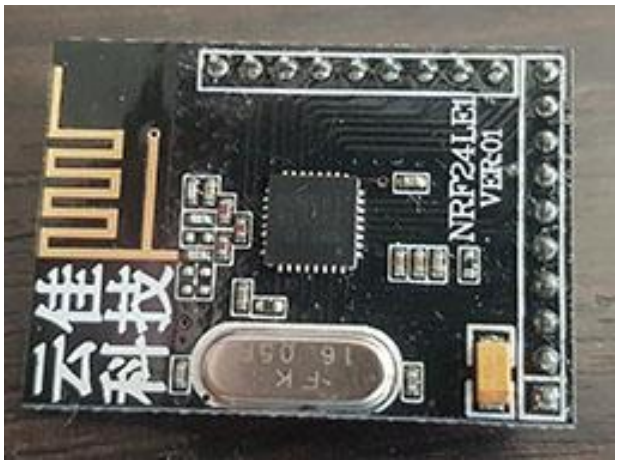
В качестве шлюза выбрал комбинацию ESP8266 + NRF24LE1. ESP осуществляет общее управление, связь с MQTT, web интерфейс, а общением с NRF клиентами занимается NRF24LE1. Общение между NRF и ESP идет текстовыми строками по RS232 на скорости 38400. Так как от ESP требуется только 3 вывода: Tx-Rx RS232 и один Reset NRF, то можно использовать самую простую и «безногую» ESP-01. Для прошивки ESP я использовал адаптированный код по ссылке: https://github.com/seb821/espRFLinkMQTT . Добавлены процедуры работы с NRF24, заменена на асинхронную MQTT библиотека. Файл для программирования один для всех шлюзов, называется espnrf1.bin для ESP с 1M памяти или espnrf4.bin для ESP с 4M памяти, находится он в папке espgateway. Если же нужно заново скомпилировать прошивку, нужно скачать zip файлы 2-х дополнительных библиотек: Asynchronous MQTT client (https://github.com/marvinroger/async-mqtt-client) и Async TCP Library for ESP8266 (https://github.com/me-no-dev/ESPAsyncTCP). Затем подключить библиотеки Sketch > Include Library > Add . ZIP Library. А библиотеки ArduinoJson и PubSubClient для компиляции не нужны.

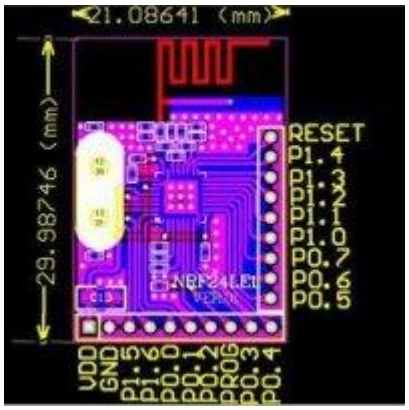
Программируется любым программатором ESP, например, таким:

Саму программу можно скачать с официального сайта espressif:

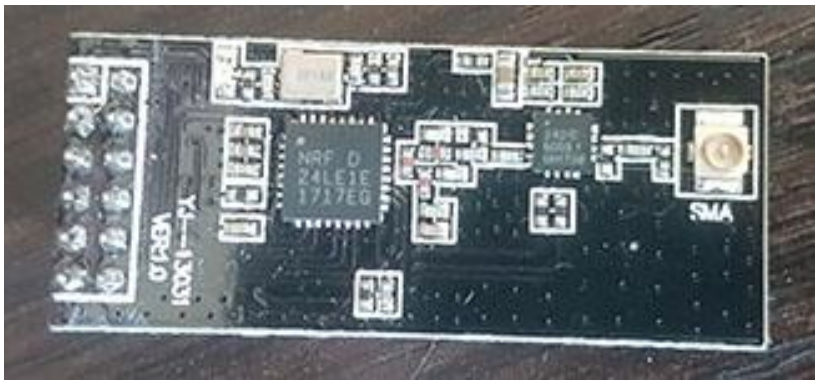
https://www.espressif.com/sites/default/files/tools/flash_download_tools_v3.6.8.zip

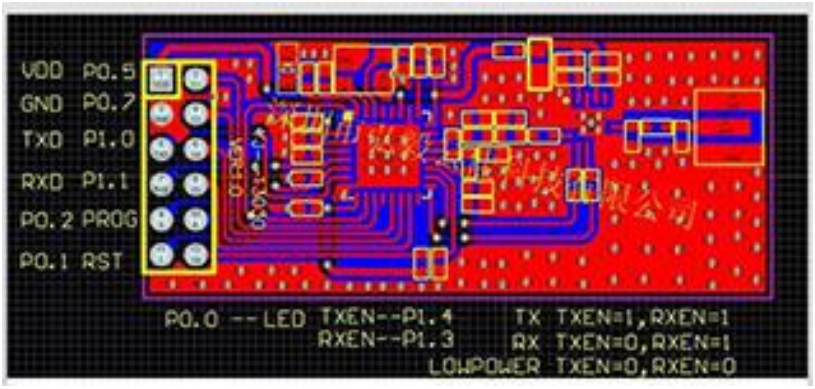
В шлюзе я использовал 32-контактный вариант NRF24LE1, там антенна побольше, а размеры платы не так критичны Внешний вид и назначение выводов на фото ниже:




Программой поддерживается и вариант NRF24LE1 с усилителем RFX2401C:






В этом модуле TX чипа RFX2401C управляется P1.4, RX чипа – P1.3, светодиод P0.0-GND светится при приеме RF пакета. Но ощутимого прироста дальности при работе с обычными NRF24LE1 ожидать не стоит, так как по передаче усилитель дает выигрыш 20dBm, а по приему только 6dBm. А использование в выключателях NRF24LE1 с усилителем невозможно, не хватит мощности их блока питания, не говоря уже о размерах платы.

В каждом шлюзе прошивка для NRF24LE1 своя, отличается двумя параметрами. Это RRFID, который должен быть на 1 меньше номера шлюза (0 для nrf1 … 8 для nrf9), и RFCH – номер радиоканала. После установки параметров прошивку надо скомпилировать. Для Telemark Assembler, команда для получения бинарного файла выглядит так:

tasm.exe -51 -b -fff NRFSRV00.A51

Если нужен Intel Hex формат:

tasm.exe -51 –g0 -fff NRFSRV00.A51

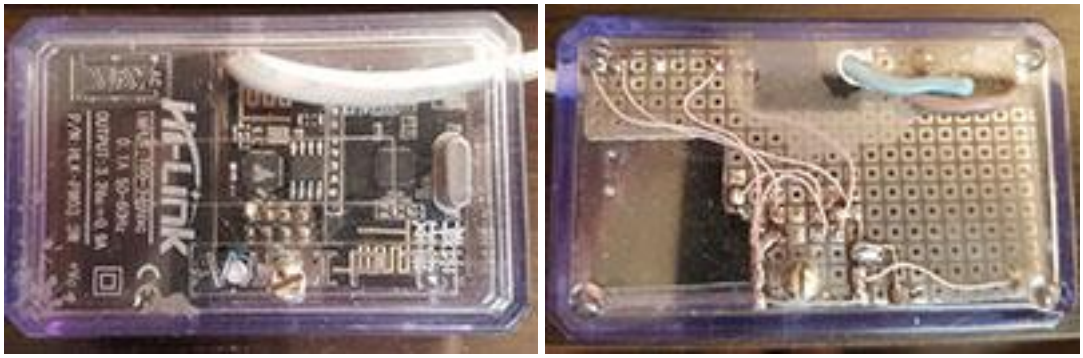
Затем программатором прошиваем микросхему. После прошивки ESP и NRF24 можно собирать схему, где дополнительно еще нужен источник питания, например, Hi-Link на 3.3v 0.9a, и емкость по питанию 22-100uF. Я использовал для монтажа макетную плату с вырезами напротив антенн и провод мгтф. Соединений немного:

*ESP gpio1(tx) <-> NRF(32pin) p0.4(rx),*

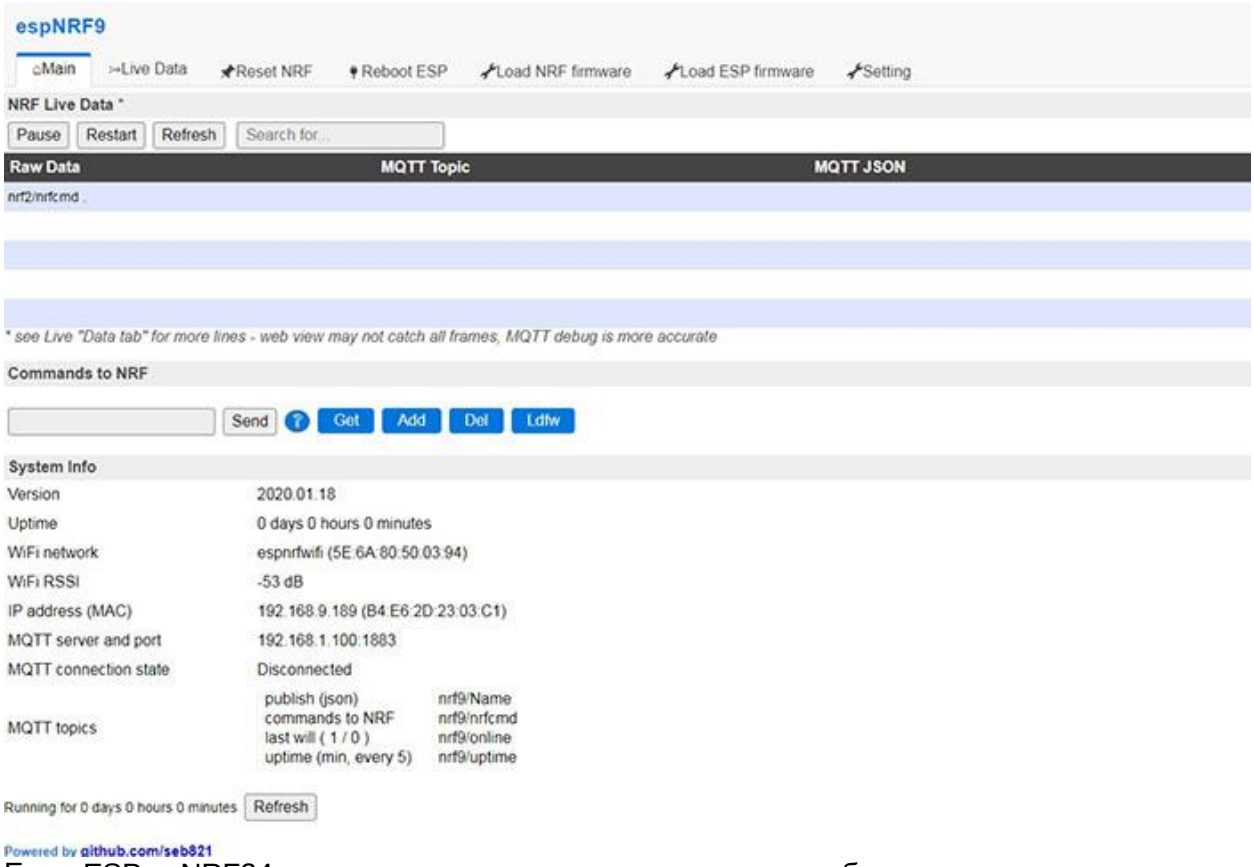
*ESP gpio3(rx) <-> NRF(32pin) p0.3(tx),*

*ESP gpio02 <-> NRF reset.*

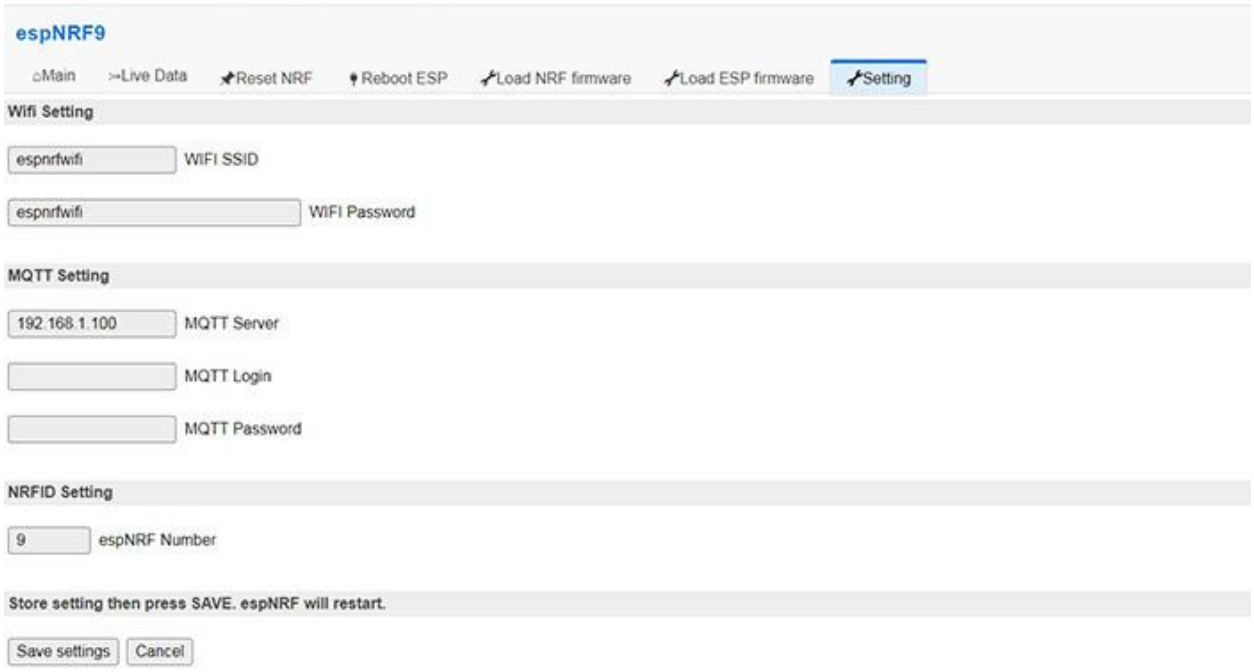
Ну и соединить общий и питание у всех элементов схемы. Фото готового шлюза:



После сборки включаем питание шлюза, создаем в роутере гостевой сегмент WiFi с именем espnrfwifi и паролем espnrfwifi. Через некоторое время шлюз должен к ней присоединиться. В роутере ищем в списке подключенных устройств espNRF9, определяем его IP адрес и вводим его в Chrome или Opera (в IE11 команды не проходят, надо разбираться). Откроется главная страничка.




Если ESP и NRF24  правильно соединены и прошиты, а таблица имен еще пустая, то в окне Raw Data через некоторое время появится nrf1/nrfcmd . (для RRFID=0 в прошивке NRF24). Но, пока не совпадают номера nrf в Raw Data и espNRF Number(номер шлюза), команды работать не будут. Далее открываем setting:



Выставляем параметры Wi-Fi, MQTT, espNRF Number(1…9), нажимаем Save setting. Шлюз перезапустится и подключится уже к сети Wi-Fi с введенными параметрами и к MQTT серверу. При недоступности MQTT сервера шлюз рестартует примерно каждые 5 минут, а при недоступности Wi-Fi пытается подключиться к сети espnrfwifi.

Осталось заполнить таблицу имен. Для добавления имени в web интерфейсе есть команда:

add n p namepar,

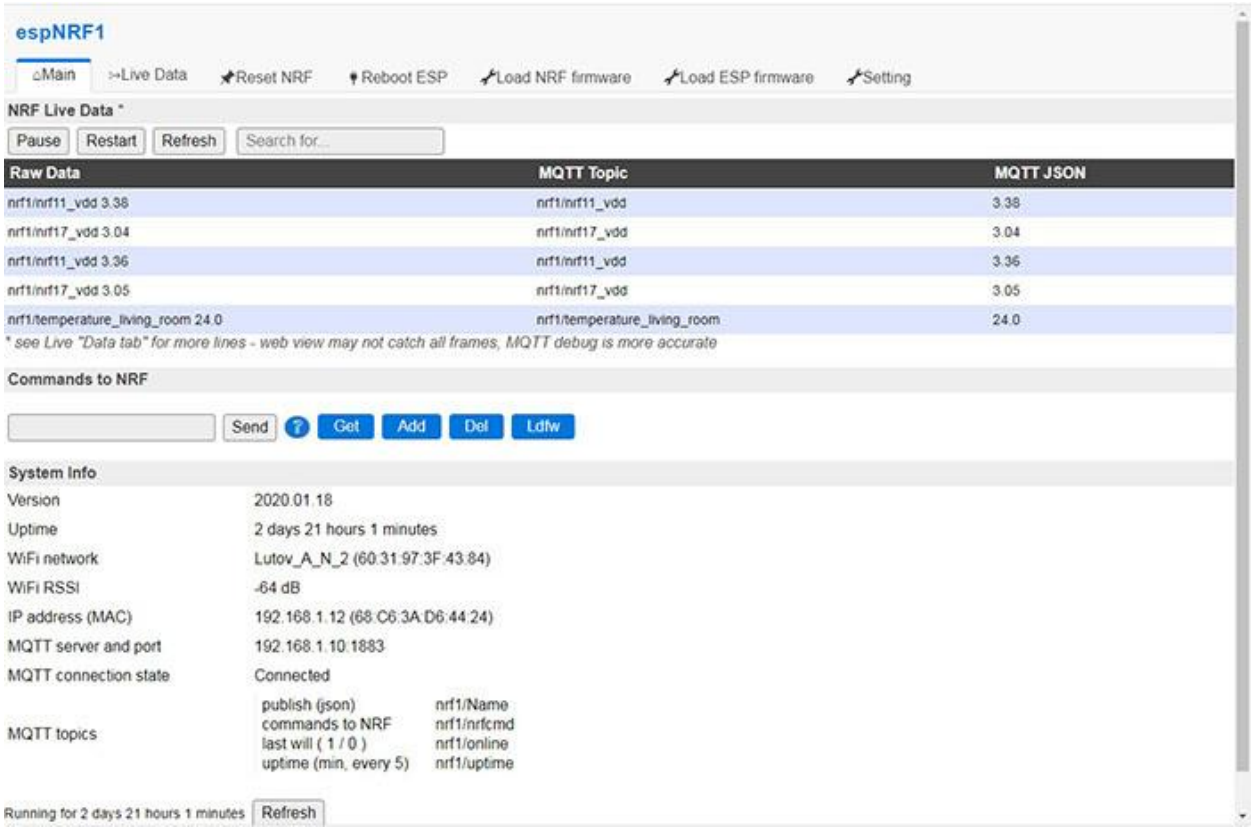
где n(1…16) – номер клиента, p(1…4) – номер параметра, namepar – имя параметра.

Параметры разделяются пробелом, в имени пробелов быть не должно. Например:

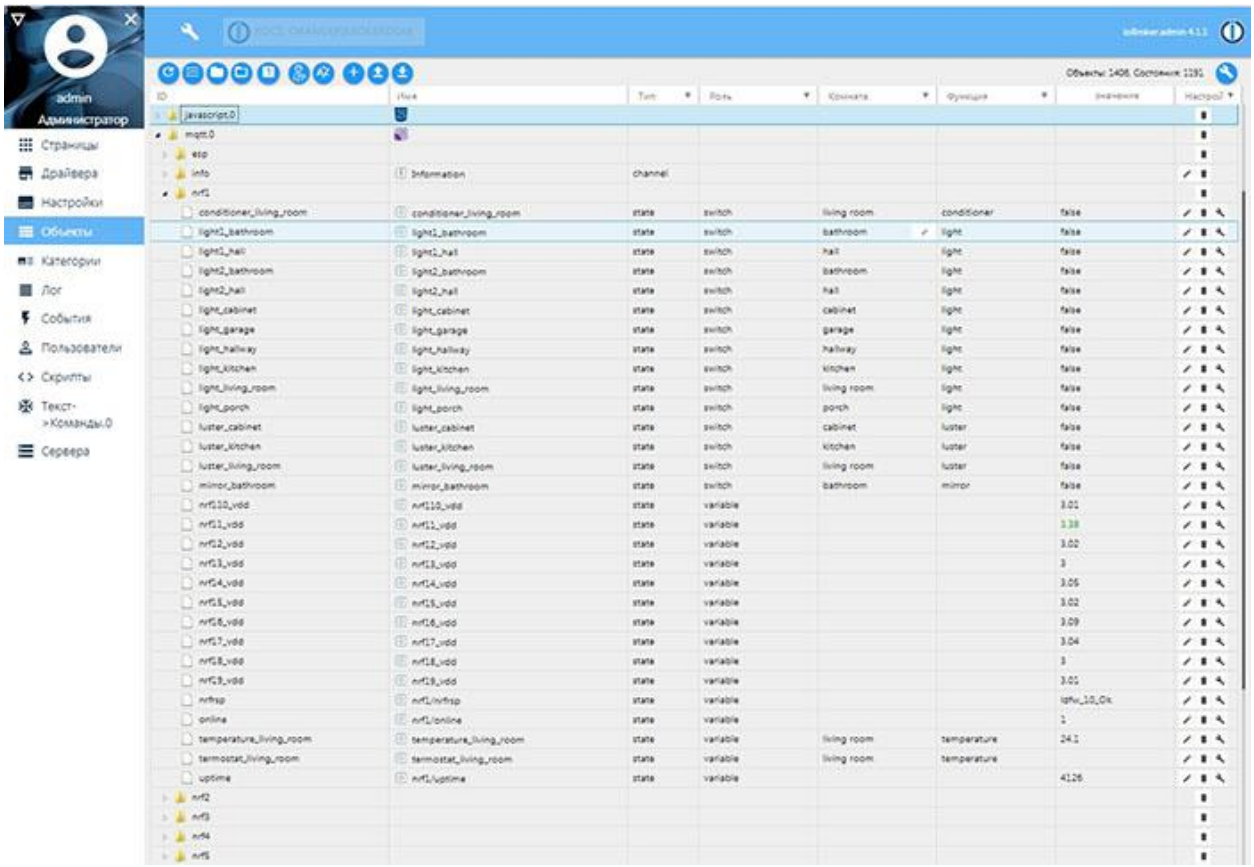
«add 1 1 light_living_room», «add 1 3 temperature_living_room», «add 1 4 nrf11_vdd».

Набираем ее в окошке «Commands to NRF» или нажмаем кнопку «Add». Вводим или корректируем недостающие параметры, затем нажимаем «Send» или ввод. Если команда выполнена без ошибок, шлюз ответит «Ok». Если «Error», значит имя уже есть в таблице, и его сначала нужно удалить. Для удаления имени используется команда «del n p» с теми же параметрами. Для быстрой очистки таблицы предусмотрена команда «del all». Для просмотра имени в таблице нужно ввести команду «get n p».

Сразу же после ввода имени шлюз начнет опрос соответствующего устройства. Главная страничка подключенного к серверу MQTT шлюза с заполненной таблицей имен выглядит примерно так:



Каждый шлюз создает свою папку nrf1….nrf9 на MQTT сервере. В папке создается топик nrfcmd для приема команд, nrfrsp для вывода результатов, uptime для отображения времени подключения, online для отображения наличия связи с espNRF. Имя и значение сразу появятся в топике nrf1….nrf9, если есть связь с устройством, а если связь не установлена, то значение всех параметров будет «NoRFL». У меня в iobroker-е это выглядит так:



Для включения нужно ввести 1, on или true в соответствующий топик, для выключения 0, off или false. Выключатель, получив команду, сразу же шлет подтверждение. Один из 3-х вариантов можно выбрать при прошивке. После чего в течение 4-х секунд пытается выполнить команду. Если команду выполнить не удалось, например, выключатель не обучен, через 4 секунды отсылается его реальное состояние. Реальное состояние отсылается сразу при попытке изменить данные только для чтения, например, напряжение или температуру. Если же выключатель недоступен, подтверждения не будет. При локальном изменении состояния выключателя новое состояние придет в следующем цикле его опроса. По времени это максимум до 16 секунд при 16 активных выключателях, 10 секунд при 10 и т.д.

## Обновление

Для обновления прошивки ESP8266 и NRF24LE1 используются команды «Load ESP firmware» и «Load NRF firmware» соответственно. Для загрузки обновления ESP8266 используется тот же файл, что и для прошивки программатором. Для обновления же NRF24LE1, что в шлюзе, что в клиенте, бинарный файл прошивки, полученный после компиляции, нужно сначала обработать программой nrfcrc.exe. Программа обрезает файл до длины 6144 байт и добавляет в конец файла CRC. После чего командой «Load NRF

firmware» обновление загружается в буфер шлюза. Далее для загрузки на устройство и запуска обновления используется команда «Ldfw n», n(1…16) – номер клиента. После ввода команды обновление передается на нужное устройство и при отсутствии ошибок сохраняется и затем запускается на исполнение. Для обновления NRF24LE1 на самом шлюзе используется команда «Ldfw 0». Для исключения ошибок при прошивке каждое устройство сравнивает перед прошивкой свой тип прошивки (RFWID=0 - шлюз, RFWID=1 – выключатель), а также свой адрес устройства (RRFID) с принятым в обновлении. Кроме того, обновление выключателя возможно только, если включен одинарный выключатель или хотя бы одна половина двойного. Только тогда его блок питания способен обеспечить ток, достаточный для успешной прошивки микросхемы.

## Заключение

Шлюзы и выключатели довольно стабильно работают у меня в течение полугода. Замеченные баги правлю и обновляю прошивки на гите. Возможно, некоторые моменты не освещены, но в основном информации должно быть достаточно для повторения.