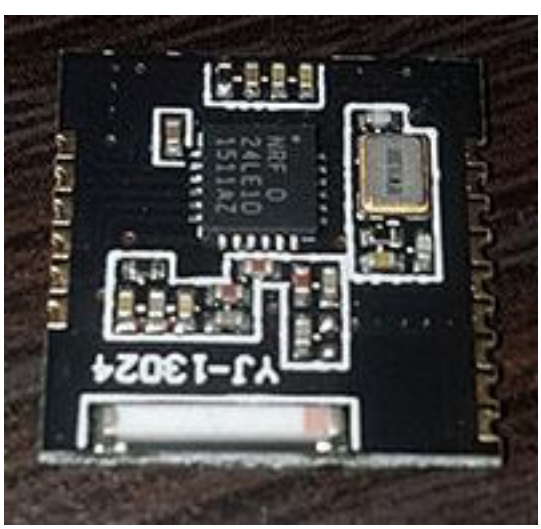
Доработка выключателей

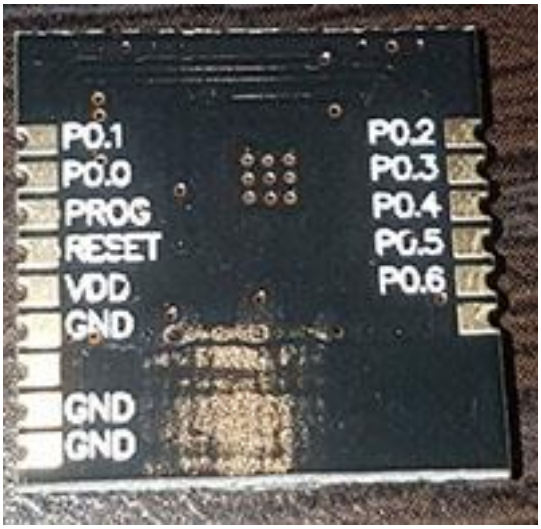
В свое время заменил дома выключатели на livolo и maifom с управлением по RF 433MHz. Настроил управление ими через Broadlink. Потом запустил iobroker, система понравилась, пользуюсь ею и сейчас. Есть в ней и драйвер Broadlink. Работало все нормально, но все же хотелось бы иметь обратную связь. Поискал по интернету варианты переделки livolo, но интересных мне решений не нашел. 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

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 сохраняет 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. Сам приемник придется удалить с платы. Приемопередающий модуль диапазона 2.4 GHz NRF24LO1 при мощности 0 dBm и скорости 250 kbit требует при приеме/передаче соответственно 12.4ma и 11.1ma, и я не нашел цифр ниже этих у других модулей. И нужно еще от 1 до 10ma для работы обслуживающего его контроллера. В качестве приемопередатчика я выбрал NRF24LE1. Это тот же модуль NRF24LO1(12.4 ma /11.1ma) плюс встроенный микроконтроллер с 8051 ядром. Конечно, ресурсы его ограничены, что потребует более серьезного программирования, но и потребление 2.5ma на частоте 8MHz. У меня NRF24LE1 работает на частоте 1MHz. Этого оказалось вполне достаточно для работы с радиомодулем. Приемопередатчик 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 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. Это 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.

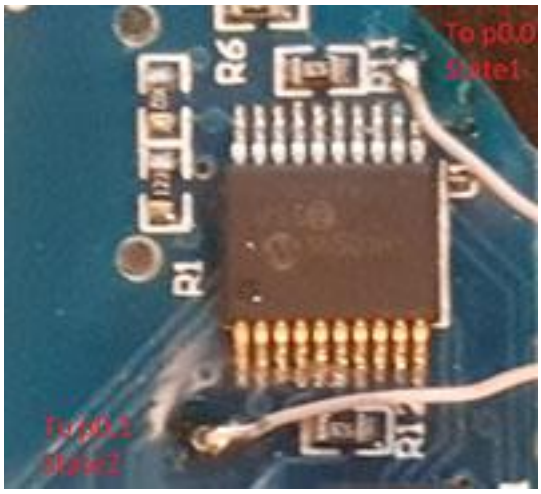
NRF24LE1 выпускается в 3-х разных корпусах, с 24, 32 и 48 выводами. Плата обычно включает в себя не только саму микросхему, но и антенну с кварцем и

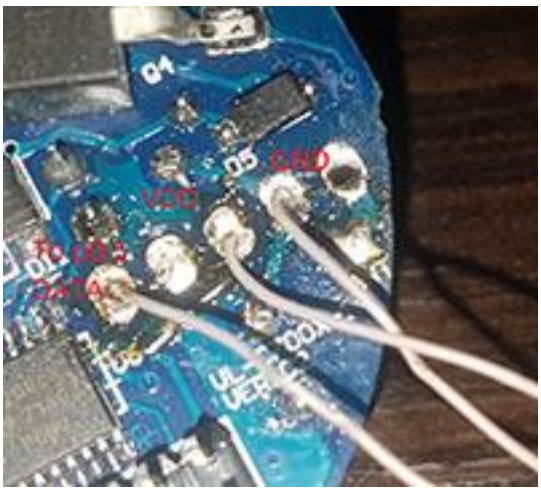
сопутствующими элементами. Найти ее в самом компактном исполнении можно здесь: https://ru.aliexpress.com/item/32665457728.html?spm=a2g0s.9042311.0.0.4de033edB2Yt4h





Эта плата с NRF24LE1 по размеру лишь немного больше NRF24LO1. Нужно только запрограммировать контроллер и припаять 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 обернуть плату скотчем, допилить паз в металлическом креплении и зафиксировать плату тем же скотчем как, чтобы антенна немного выступала над креплением. Для уменьшения пульсаций я все же меняю емкость в блоке питания livolo 330uF*25V на 1000uF*25V. Готовая конструкция у меня выглядит так:

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. 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. Ну и для контроля температуры можно подключить к выходу p0.4 датчик DS18B20. На сервер передается 4 параметра. Первые 2 отражают состояние выключателей, 3-й параметр – это температура, и 4-й – напряжение питания NRF24LE1. 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

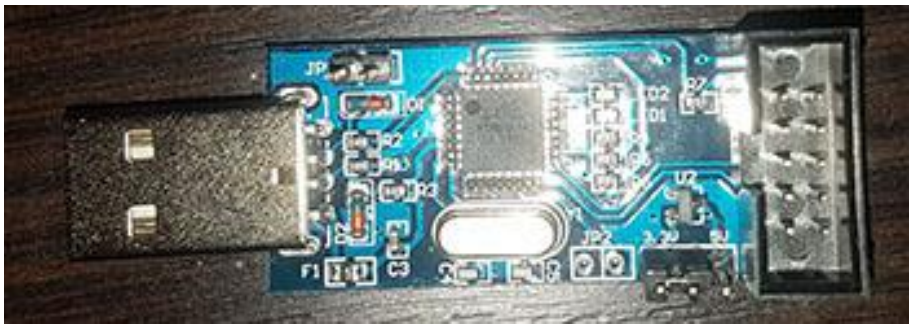
Для каждого устройства перед компиляцией нужно установить несколько основных параметров. Это номер канала, в исходнике RFCH, диапазон значений 0 … 7Eh(0 … 125), а также адрес устройства (RRFID), для каждого устройства он свой, значения 0 … 0Fh (0…15), до 16 устройств на один канал и шлюз. Также нужно выбрать протокол управления выключателем (RCPRT =0-livolo,RCPRT <>0 – rcswitch) и формат вывода 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:



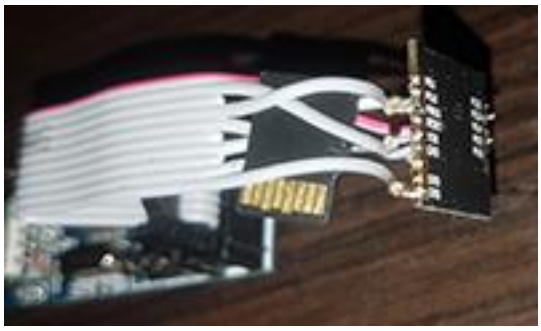
1. FMOSI	2. VDD (+3.3)
3 —	4. PROG
5. FCSN	6 —
7. FSCK	8 —
9. FMISO	10. GND

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 по Wi-Fi.

Шлюз

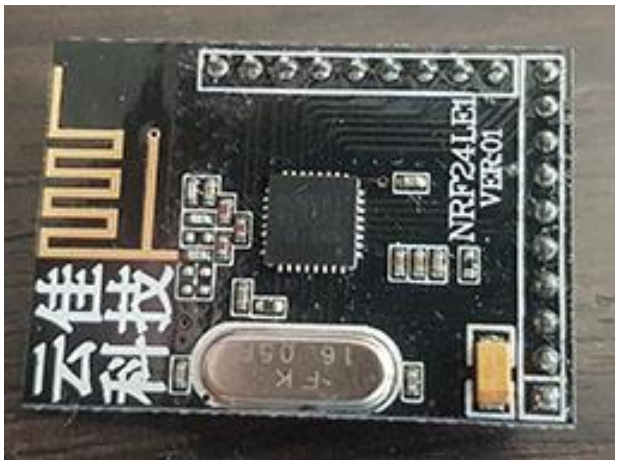
Управлять выключателями удобнее всего по протоколу MQTT. Все системы умного дома или имеют в своем составе сервер MQTT, или умеют с ним работать. Для связи с MQTT сервером необходим шлюз, лучше Wi-Fi. Учитывая небольшую мощность NRF24, порядка 1мВт, для обеспечения надежной связи с выключателями может потребоваться несколько шлюзов, как минимум один на этаж в многоэтажном доме. И лучшее место установки шлюзов за подвесным потолком в коридоре или холле, где сходится большинство комнат и, соответственно, выключателей в доме.

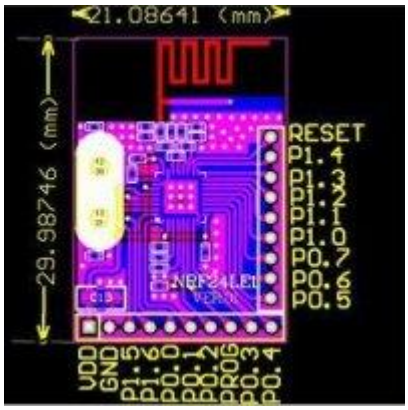
Кратко по протоколу работы шлюза с контроллерами NRF24LE1 в выключателях (далее для краткости клиентами). Обмен между шлюзом и клиентами идет 24 байтовыми пакетами. Каждый клиент имеет свой адрес от 0 до 15, а приемник клиента настроен принимать пакет от шлюза только тогда, когда адрес в пакете совпадает 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 успеет 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. Строка имеет вид: nrf[номер_шлюза]/[имя_параметра] [значение_параметра].

Так как это не стандартный протокол, то и шлюз пришлось делать самостоятельно. 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 RS232, то можно использовать самую простую и «безногую» ESP-01. Для прошивки ESP я использовал адаптированный для работы с NRF код по ссылке: https://github.com/seb821/espRFLinkMQTT . Спасибо seb821! Файл

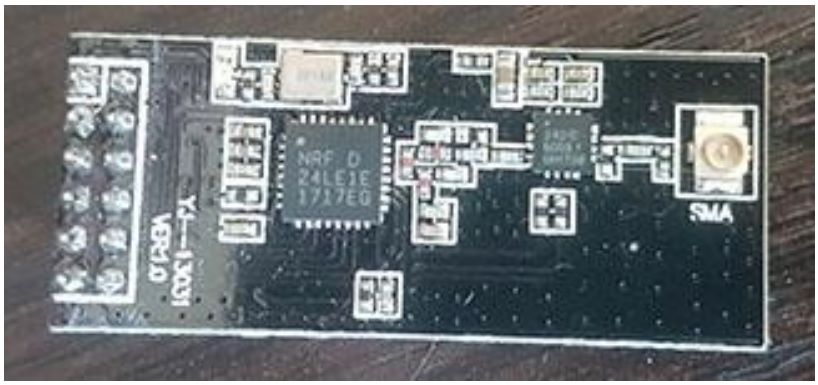
для программирования один для всех шлюзов, называется espnrf.ino.bin и находится он в папке espgateway. Программируется любым программатором ESP, например, таким:

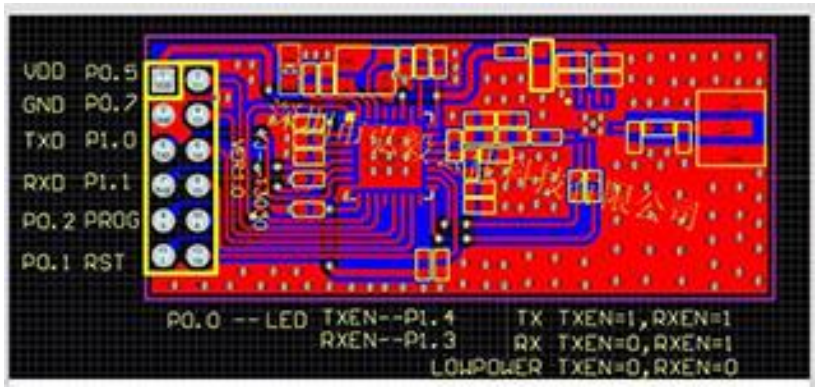
В шлюзе я использовал 32-контактный вариант NRF24LE1, там антенна побольше:





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 ожидать не стоит, так как по передаче усилитель дает выигрыш 20dBm, а по приему только 6dBm.

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 3.3в и емкость по питанию 22-100uF. При этом схема получается даже проще, чем при использовании NRF24LO1. Ее проще написать, чем рисовать:

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. Фото готового шлюза: