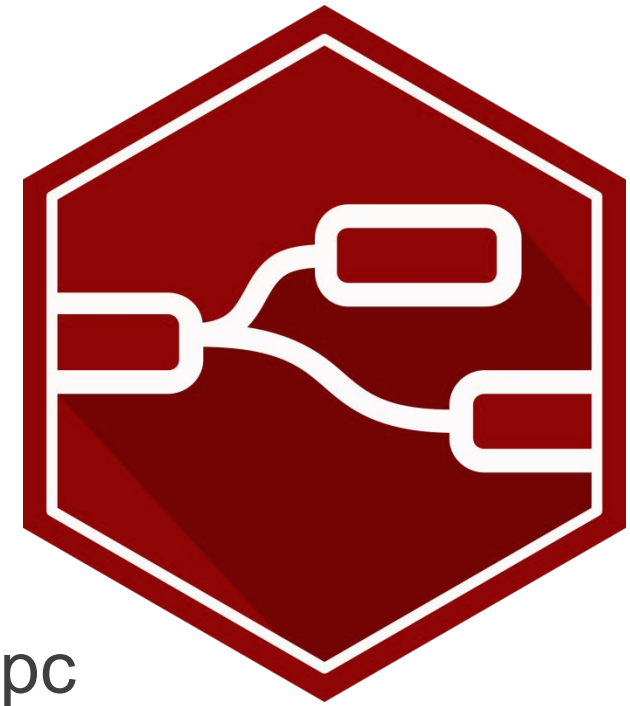
Node-RED для умного дома

управление освещением по датчикам движения и освещения



Докладчик: Епанов Андрей, 2(м) курс

Одним из наиболее важных факторов ограничивающих развитие интернета вещей является отсутствие удобных средств разработки правил взаимодействия устройств IoT между собой. Для решения этой задачи был разработан Node-RED, позволяющий через браузер построить схему взаимодействия устройств между собой и внешними системами.

Данное решение удобно как промежуточное для связи устройств различного типа между собой и/или же с системой автоматизации или, например, СУБД или иным облаком. С использованием дополнительных пакетов Node-RED можно использовать для создания простых систем автоматизации умного дома, но решение будет относительно ограниченных ввиду неполного покрытия функциональных потребностей умного дома.

Node-RED работает на Node.JS, и был разработан для работы на относительно малопроизводительных системах, таких как:

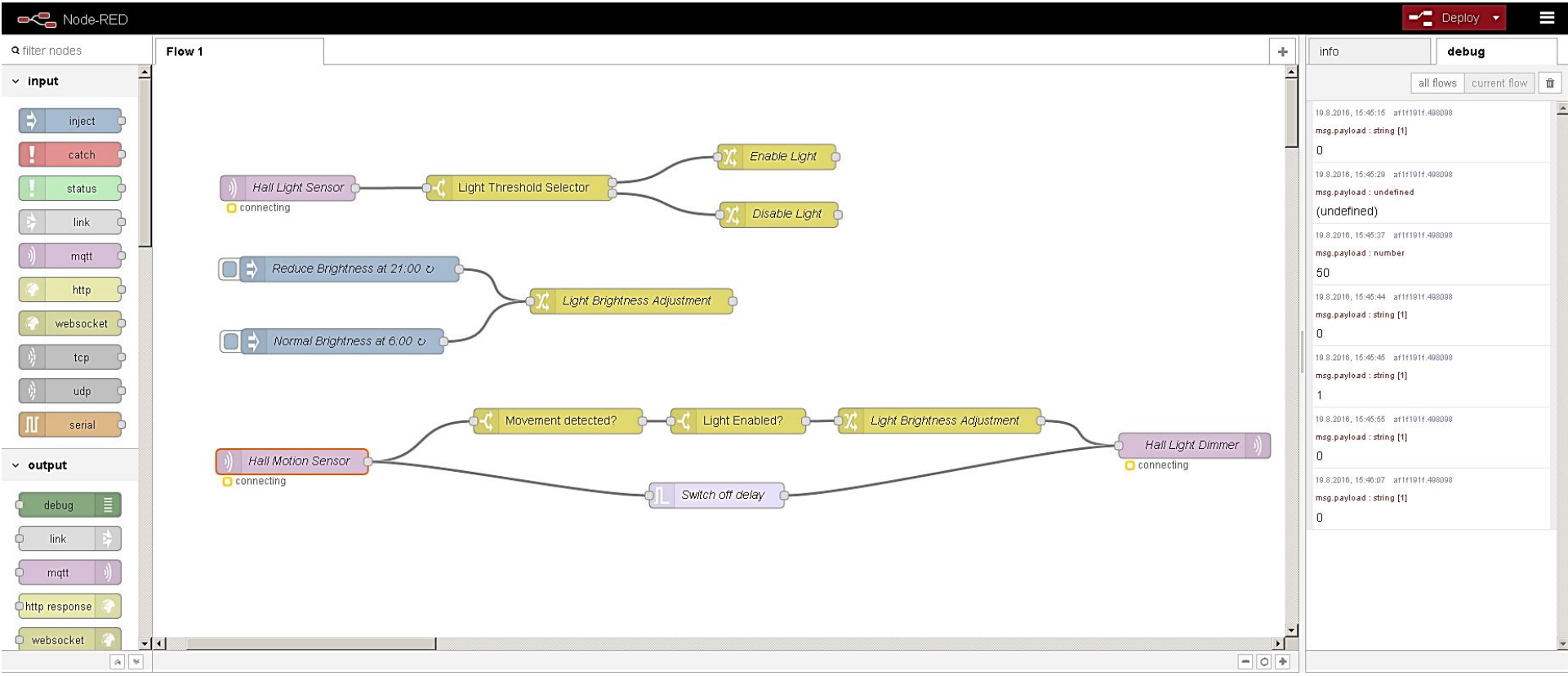
- Raspberry Pi
- BeagleBone Black
- Arduino (но можно крутить просто на локальной машине. Или вообще в облаке)

С учётом озвученных факторов Node-RED удобно использовать на шлюзах между различными сетями устройств интернета вещей, функционирующих на собственных, как правило, более простых протоколах и традиционным интернетом, построенных на TCP/IP, UDP. В этом случае он позволит более оптимально использовать свободные ресурсы шлюза, работающего, как правило,

Что такое NODE-RED ?

Node-RED — это инструмент с открытым исходным кодом, который позволяет создавать приложения полностью в графическом редакторе без написания какого-либо кода, просто соединяя готовые компоненты. Компоненты - это предварительно прописанные части кода, выполняющие желаемое действие, например взаимодействие с внешними устройствами, онлайн-службы и т.д. Для связи компонентов друг с другом используются графические линии связи, по которым пересылаются данные между узлами. Связывать различные блоки можно просто мышкой и при этом не иметь каких-либо особых навыков в программировании. Когда есть идея, но лень писать код, то в первую очередь стоит вспомнить о Node-RED и подобных инструментах. Ну а если идея взлетит, то при желании код всегда можно дописать потом.

Разработка в Node-RED ведется через обыкновенный браузер, само приложение можно запустить на различных платформах – PC, Raspberry Pi, в облаке и т.д.


Node-RED
Deploy
filter nodes
Flow 1
input
inject
catch
status
link
mqtt
http
websocket
tcp
udp
serial
output
debug
link
mqtt
http response
websocket
Hall Light Sensor
connecting
Light Threshold Selector
Enable Light
Disable Light
Reduce Brightness at 21:00
Normal Brightness at 6:00
Light Brightness Adjustment
Hall Motion Sensor
connecting
Movement detected?
Light Enabled?
Light Brightness Adjustment
Switch off delay
Hall Light Dimmer
connecting
info
debug
all flows
current flow
19.8.2016, 15:46:15 af1f191f.498098
msg.payload : string [1]
0
19.8.2016, 15:46:29 af1f191f.498098
msg.payload : undefined
(undefined)
19.8.2016, 15:46:37 af1f191f.498098
msg.payload : number
50
19.8.2016, 15:46:44 af1f191f.498098
msg.payload : string [1]
0
19.8.2016, 15:46:45 af1f191f.498098
msg.payload : string [1]
1
19.8.2016, 15:46:55 af1f191f.498098
msg.payload : string [1]
0
19.8.2016, 15:46:07 af1f191f.498098
msg.payload : string [1]
0

Node-RED для умного дома

Node-RED можно использовать для написания сценариев и правил для умного дома. То есть для автоматизации. Сама связь с различными исполнительными устройствами, веб-сервисами и датчиками можно реализовать, к примеру, на OpenHAB. Почему не сделать автоматизацию там же? Несколько причин:
— В OpenHAB сценарии и правила пишутся на своем языке.
— Сама отладка правил практически невозможна – если правило не работает, сложно разобраться почему
— Правила дложны быть независимы от железа, каналов связи, платформ и самого ПО для коммуникации с устройствами. Чтобы легко перейти на другую платформу УД*, например Domoticz, MajorDomo, FHEM и взять уже написанные правила с собой, а не переписывать их заново под новую платформу УД.

*УД – Умный
Дом

Управление освещением по датчикам движения и освещенности

Приступим к реализации. Собственно, оригинальная задача проста и тривиальна:
Есть управляемые LED споты для освещения. Нужно, чтобы свет загорался по движению и выключался сам через 10 секунд.

Немного усложним задачу:
— Свет должен включаться только когда на улице темно
— Интенсивность света должна зависеть от времени – до 9 часов вечера свет должен включаться с полной интенсивностью, а после только на 10%, как подсветка.

Датчики, исполнители и пр. железо

Пусть в нашем умном доме все эти протоколы приводятся к одному – MQTT, а через него уже происходит общение с Node-RED.
И так какие же датчики и исполнительные устройства имеются?

1. Датчик движения. Публикует сообщение OPEN в топик /myhome/state/Hall_motion, когда детектирует движение и CLOSED, если в течении 2-х секунд движения нет.

2. Датчик освещенности. Он измеряет яркость уличного освещения в диапазоне 0-1000 Люкс. Публикует раз в минуту сообщение в топик /myhome/state/Lumin_Hall с уровнем текущей освещенности.

3. Диммер управления LED лампами. Он подписан на топик /myhome/command/Light_Hall/state. Если записать туда 0 – свет выключится. 100 – включится на максимальную яркость. 1-99 – будут менять интенсивность освещения. Для ночной подсветки достаточно интенсивности 1.

Входы и выходы

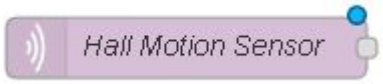
В первую очередь создаем нужные входы и выходы для нашего алгоритма. Это будут MQTT клиенты, подписывающиеся на соответствующие топики. Их перетаскиваем из библиотеки слева и настраиваем.

Примечание: Отображаемые названия блоков можно изменить в их настройках.

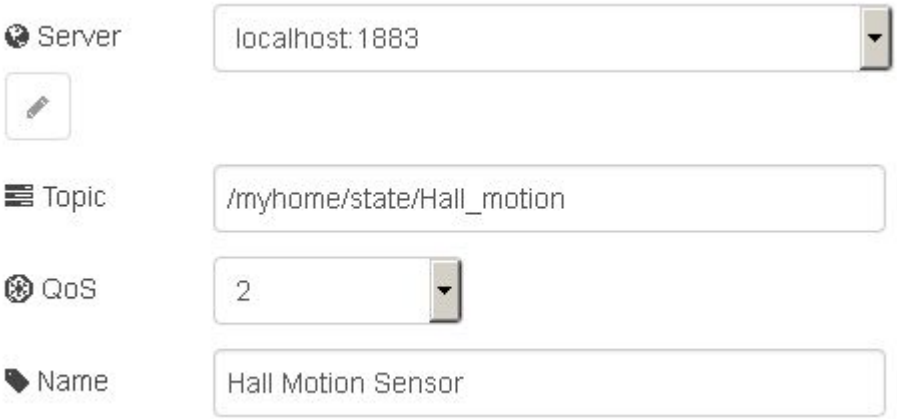
Для датчика освещенности создаем узел Hall Light Sensor из MQTT Input:

В его настройке достаточно прописать адрес MQTT брокера и топик.

Для датчика движения создаем узел Hall Motion Sensor:



У него все то же самое, только прописываем другой топик /myhome/state/Hall_motion.



Как можно заметить адрес и параметры брокера этот узел уже перенял из предыдущего, так что по новой их вводить не надо.

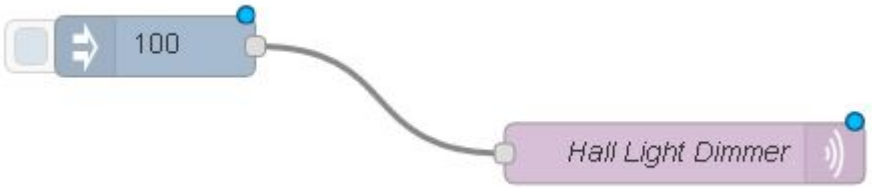
Осталось добавить MQTT Output, чтобы сделать выход для LED диммера. Перетаскиваем MQTT Output и называем Hall Light Dimmer.

В параметрах надо опять же только указать нужный топик, в который будут слаться сообщения для управления диммером — /myhome/command/Light_Hall/state

В результате мы получили три узла для нашего Flow.

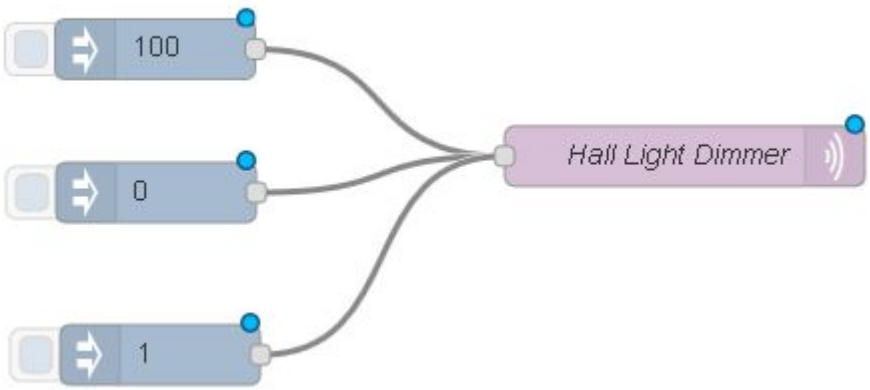
Не мешало бы их проверить на функциональность. Это легко.
К Input блокам подключаем Debug output.

А к выходному блоку подключаем Inject Input.



В настройках этого узла надо поменять payload на уровень желаемой яркости светильника. Например в данном случае это 100.

Мы можем путем copy-paste создать несколько идентичных Inject блоков, поменять яркость, и подключить к выходу вот так:



Так тоже будет работать.

Пришло время проверить. Тыкаем кнопку Deploy:

Под MQTT узлами у вас должно появиться маленькое сообщение:

Это означает, что они подключились к MQTT брокеру. Если все прошло по плану, то в правой вкладке debug у вас должны начать появляться сообщения от датчиков, а если кликать мышкой по прямоугольникам слева от Inject узлов, должна меняться интенсивность освещения у светильника, подключенного к диммеру. Если все работает, можно идти дальше.

Цепь управления освещением по датчику движения

Для простоты будем называть отдельные связанные между собой блоки цепями. В первую очередь попробуем сделать простую цепь, которая бы включала свет по датчику движения и отключала его спустя какое-то время. Для этого согласно концепции Node-RED нам нужно, чтобы нужное сообщение от датчика движения дошло до диммера через определенные блоки и имело определенный текст. Сначала разберемся с включением света.

В первую очередь выделим из всех сообщений от датчика движения сообщение с текстом OPEN — это значит, что движение появилось. Для этого используем блок switch. Подключим его к выходу уже созданного нами блока Hall Motion Sensor.

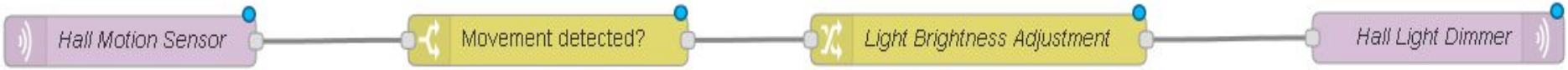
Настроим его так, чтобы блок пускал на выход только сообщения с текстом

Как видно из картинки, блок будет сравнивать payload с текстом OPEN и направлять сообщения на выход один, если текст совпадает.

Наш диммер требует, чтобы на вход ему подавались сообщения с нужной яркостью 0...100. Текст OPEN он не поймет. Поэтому используем блок Change, чтобы поменять текст сообщения.

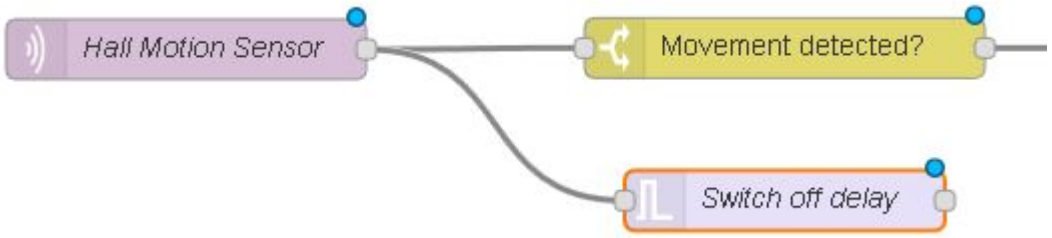
В настройках этого блока записываем требуемое изменение – message payload изменяем на 100 — требуемую интенсивность освещения.

И наконец подключаем это все ко входу нашего диммера:



Если запустить данную цепь, то можно убедиться, что она работает – свет будет включаться по движению. Осталось сделать так, чтобы он еще и выключался.

Д[illegible]го к выходу датчика



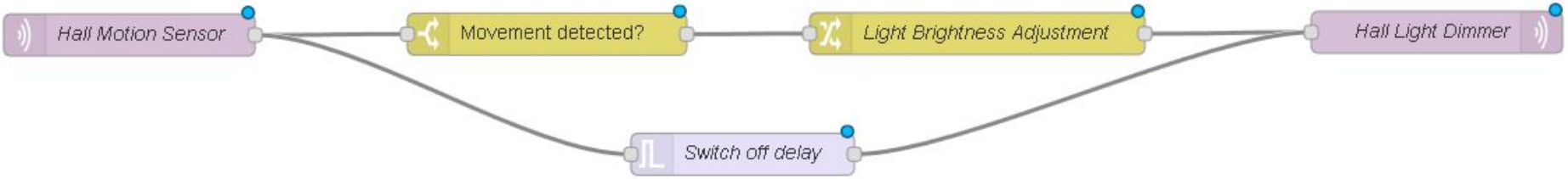
Блок триггер позволяет генерировать сообщения с задержкой, а также может быть сброшен определенным сообщением. Настроим его так:

Данная настройка означает, что при поступлении первого сообщения триггер не посылает ничего, но запускает выдержку времени в 8с и по ее истечении посылает сообщение с текстом 0. Также триггер сбрасывается, если ему на вход поступает сообщение с текстом OPEN. Что же это означает в нашем случае?

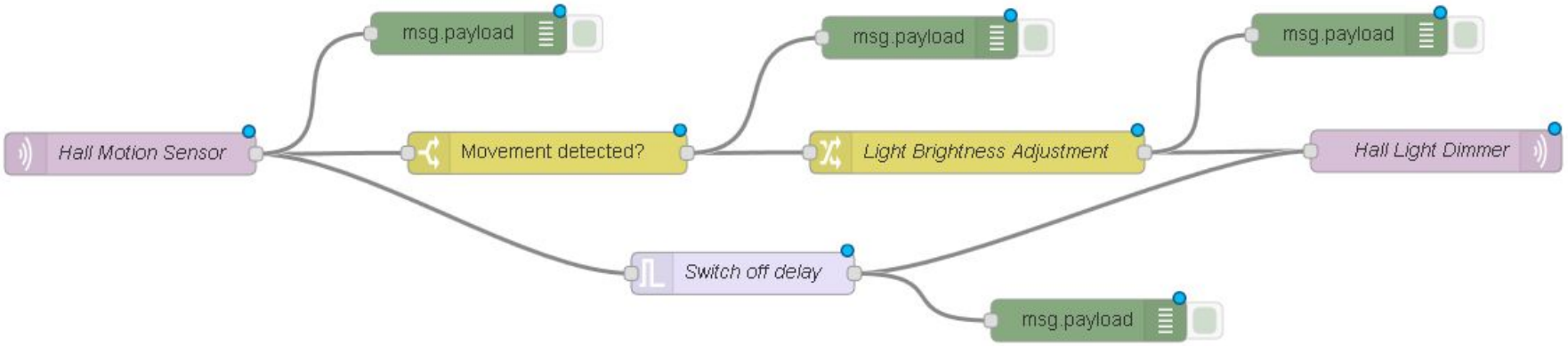
Предположим что датчик движения выдал сообщение OPEN. Данное сообщение вернет триггер к исходному состоянию без какой либо реакции. Далее через какое-то время датчик движения выдаст сообщение CLOSED. Это сообщение запустит выдержку времени и через 8 секунд после этого триггер выдаст сообщение 0.

Если в течении этой выдержки времени опять поступит сообщение OPEN, то триггер опять вернется к исходному состоянию и будет ждать следующего сообщения(которое логично будет CLOSED). При этом триггер не выдаст никаких сообщений.
То есть таким образом мы создали таймер, который будет служить нам для автоматического отключения света после заданной выдержки. Если вспомнить описание датчика движения, то становится понятным почему здесь задается 8 секунд, а не 10 – 2 секунды добавляется за счет выдержки самого датчика движения.

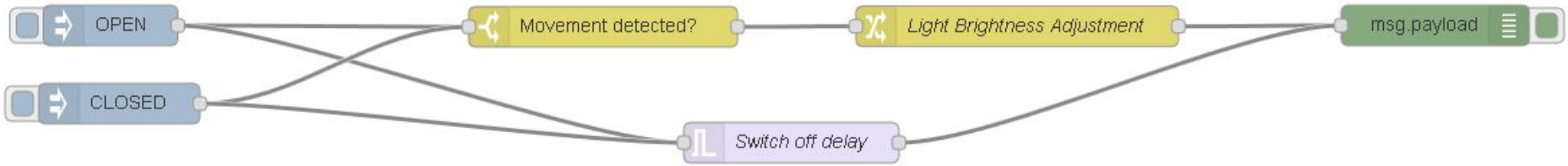
Осталось подключить выход триггера к диммеру и можно запускать цепь на проверку.



Вы также можете обвешать вашу цепь Debug блоками, чтобы разобраться с принципом работы и тогда цепь будет выглядеть вот так:



А можно сделать вообще вот так:



И тогда клацая мышкой на прямоугольниках слева от Inject блоков вы можете отладить цепь вообще без железа – на своем лаптопе во время поездки в метро или даже планшете.

Цепь включения/отключения света в зависимости от яркости уличного освещения

Теперь нам надо сделать так, чтобы свет не включался, если на улице слишком светло. Для этого нам нужно добавить еще одну цепь и слегка изменить основную цепь, чтобы учесть данный фактор.
Сначала возьмем уже знакомый нам switch блок и подключим его к выходу датчика освещенности.

Настройку этого блока сделаем таким образом, чтобы он направлял сообщения от датчика освещения на один из выходов, в зависимости от текущей освещенности.

Условие выше означает, что если освещенность меньше 10 люкс, то сообщение будет направлено на выход 1. А иначе оно пойдет на выход 2.
Не забываем, что надо выбрать

Опцию, чтобы сообщение было направлено только на один из выходов.

Исходя из принципа работы логично получается, что если сообщение появилось на первом(верхнем) выходе, нам надо включить детектирование движения, а если на втором, то выключить.

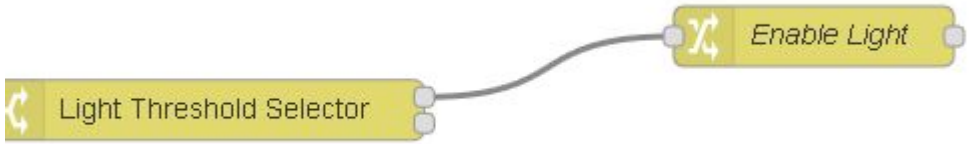
Тут, конечно, возможен 1000 и 1 способ, самый простой из которых – просто блокировать сообщение на включение света от датчика движения, если на улице светло. Реализуем это.

Следует отметить, что в Node-RED механизм исполнения реализован на основе сообщений. Т.е. нет сообщения – нет события, нет и реакции. Это накладывает определенную специфику в случае, если разные сообщения приходят асинхронно, т.е в разные моменты времени. Например в данном случае сообщение от датчика освещения асинхронно по отношению к с сообщениям от датчика движения. Поэтому, чтобы учесть эффект датчика освещения, нам надо запомнить информацию, которая была в его сообщении и затем применить ее в следующий раз, когда придет сообщение от датчика движения.

На помощь в этом случае приходит контекст – поле, где можно хранить информацию во время исполнения flow. Контекст бывает глобальным по отношению ко всей среде, или локальным по отношению к конкретному flow или вообще только по отношению к конкретному блоку.

В нашем случае мы будем использовать контекст, локальный по

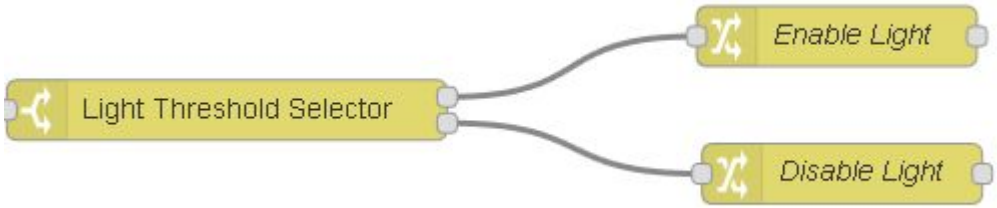
Создадим блок change и подключим его к первому выходу Light Threshold Detector



Как мы помним, на этом выходе появляется сообщение в том случае, если датчик освещенности отрапортовал, что освещение на улице менее 10 Люкс. Рассмотрим конфигурацию блока change

В данном случае мы используем правило Set, чтобы присвоить переменной flow.Light_enabled значение Yes. Таким образом мы присвоили значение глобальной переменной, которое мы можем использовать в других блоках.

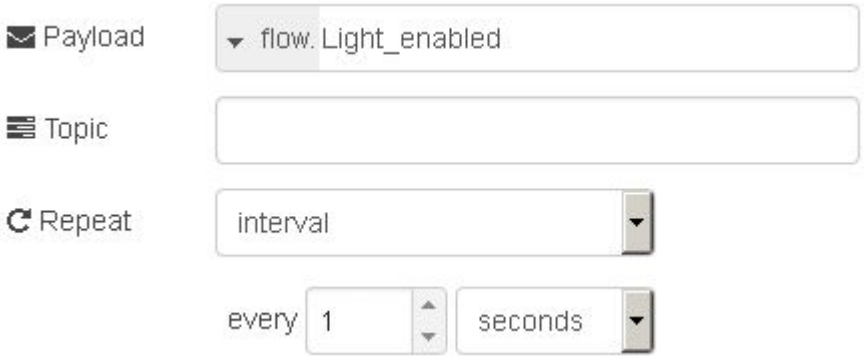
Аналогичным образом создадим второй блок change и подключим его на второй выход.



Его конфигурация будет такой:

Чтобы узнать правильно ли работает такой детектор мы можем создать простую цепь с блоками Inject и Debug.

При этом в настройках блока Inject укажем, что он должен выдавать каждую секунду значение переменной flow.Light_enabled



Тогда результат работы датчика освещенности можно легко наблюдать в вкладке Debug

19.8.2016, 15:22:01 Light_enabled
msg.payload : string [3]
Yes

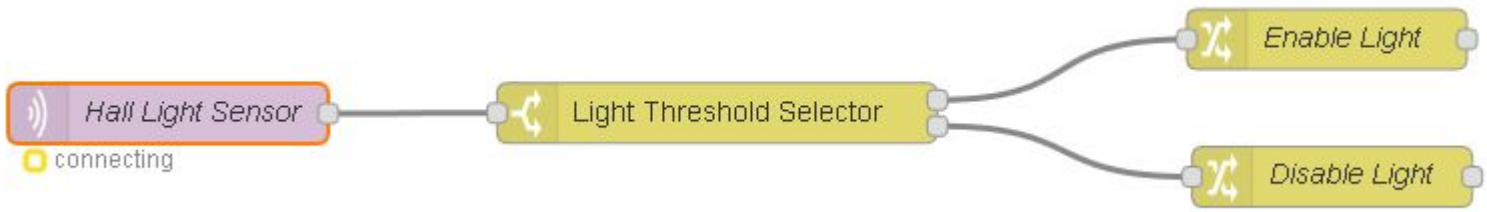
19.8.2016, 15:22:02 Light_enabled
msg.payload : string [3]
Yes

19.8.2016, 15:22:03 Light_enabled
msg.payload : string [3]
Yes

19.8.2016, 15:22:04 Light_enabled
msg.payload : string [2]
No

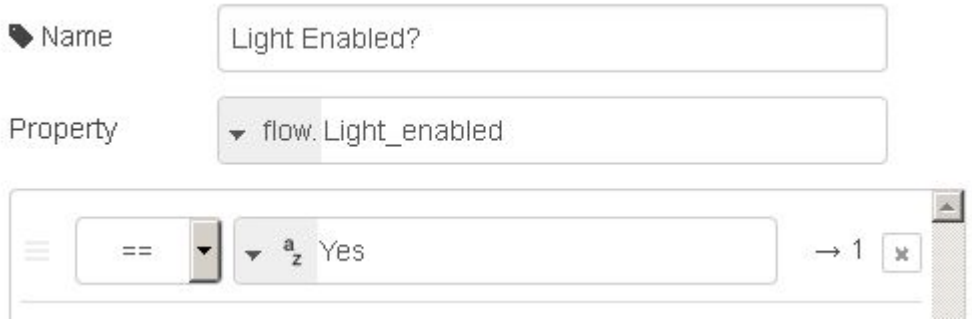
19.8.2016, 15:22:05 Light_enabled
msg.payload : string [2]
No

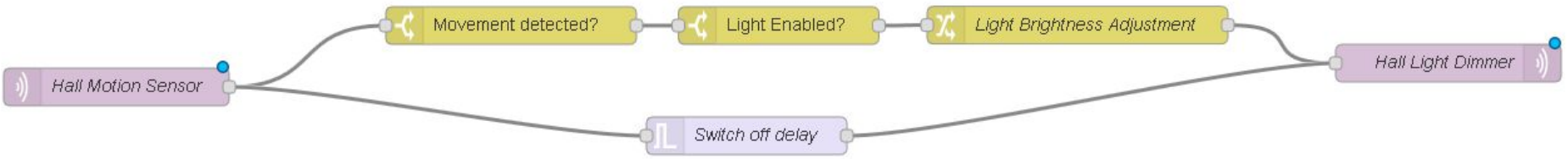
Общая цепь включения/отключения света по датчику освещения будет выглядеть следующим образом



Теперь, чтобы учесть этот эффект в цепи управления по датчику движения, нам достаточно вставить switch блок в цепь пути включения света.

И настроить его так, чтобы он пропускал сообщения от датчика движения только, если наша глобальная переменная flow.Light_enabled имеет значение Yes – т.е. на улице темно.

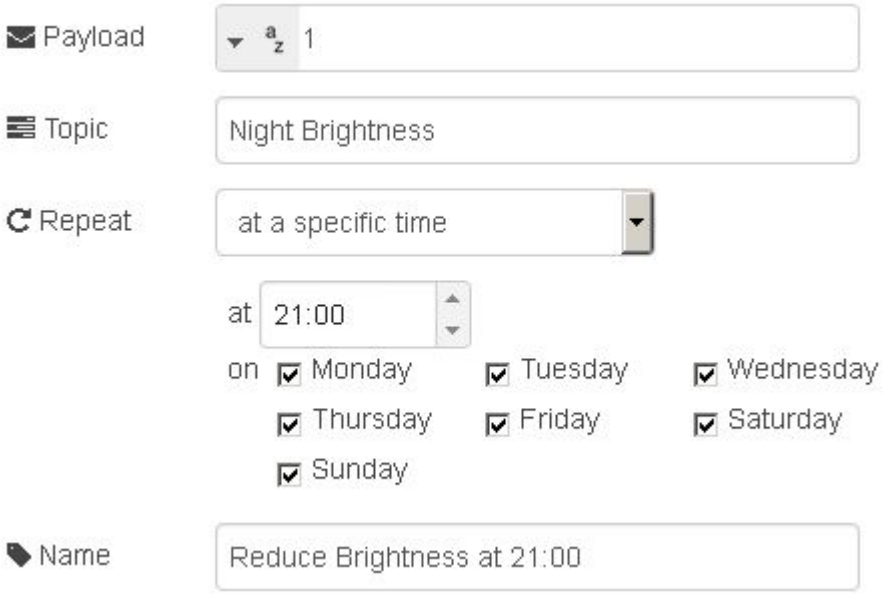




Изменение яркости светильника в зависимости от времени

Осталось совсем немного. Мы хотим изменять интенсивность LED светильника. Т.е. если у нас темно, но время до 9 часов вечера, свет должен включаться на полную мощность. Но после девяти – только на малую мощность, как ночная подсветка.
Для этого создадим блок Inject:

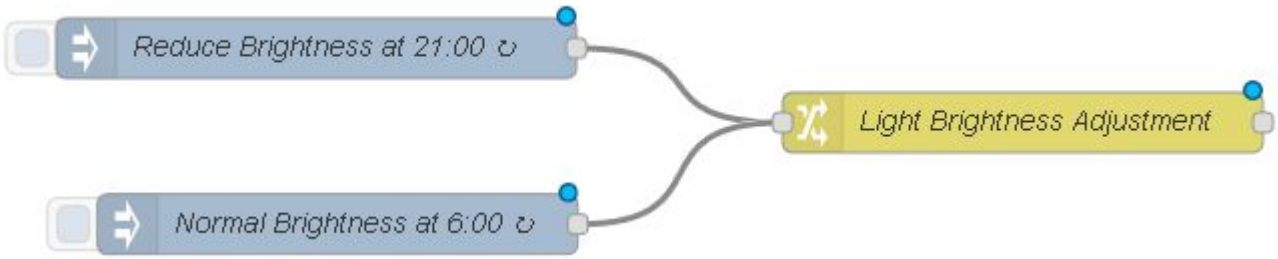
Его можно настроить таким образом, чтобы он выдавал сообщение в определенное время. Настроим его на 21:00 каждого дня.



При этом заметим, что выдавать он будет сообщение со значением 1. Это будет нашей нужной интенсивностью подсветки в ночном режиме. Чтобы использовать это значение в цепи управления светом, используем тот же трюк с глобальными переменными. Создадим блок change:

И настроим его так, чтобы переменной flow.Light_Brightness присваивалось значение из сообщения.

Чтобы возвращать первоначальную яркость по утрам, создадим второй блок inject, который будет выполняться в 6 часов утра и выдавать значение яркости 100. Подключим его туда же.

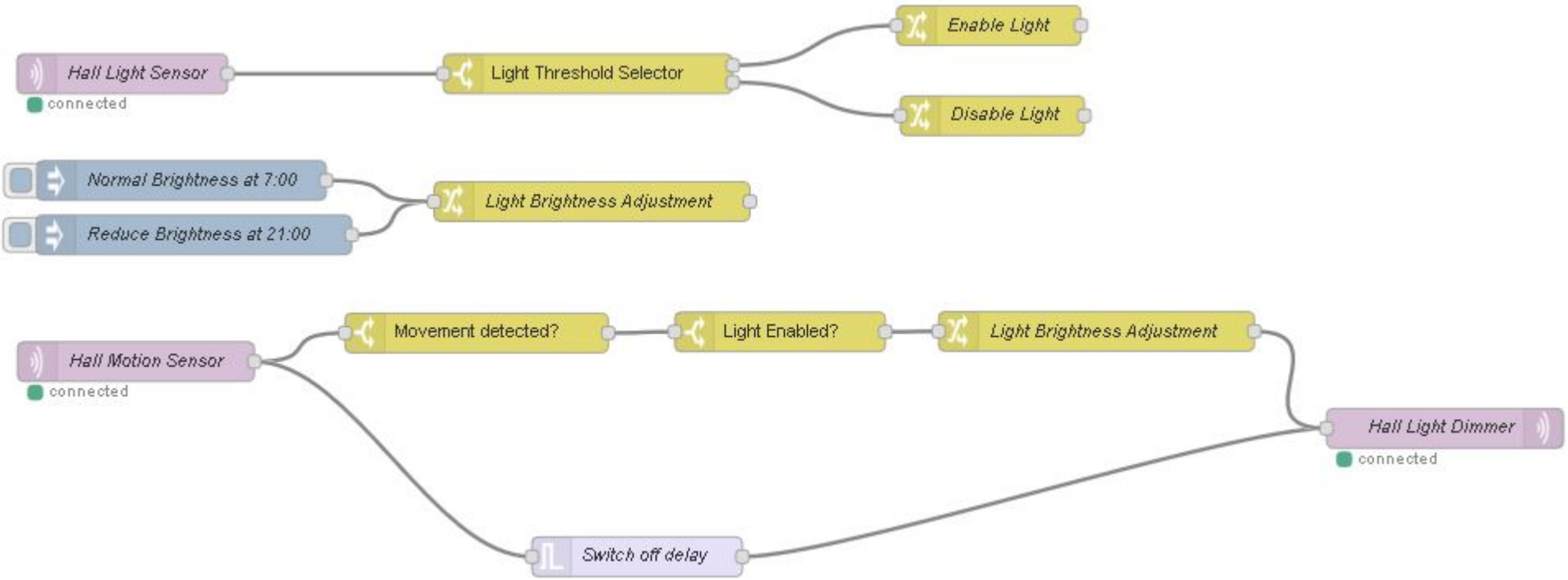


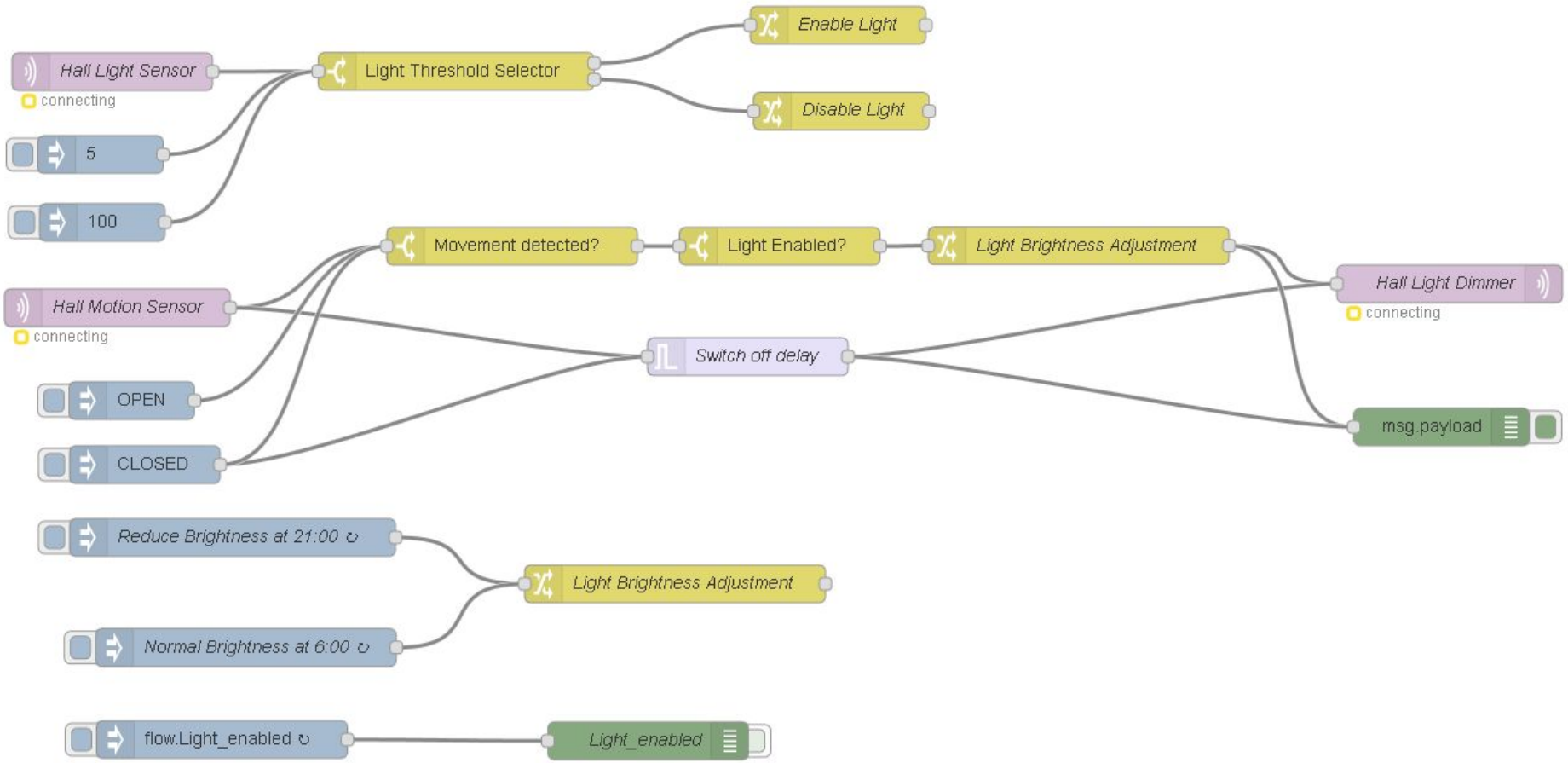
Таким образом переменной flow.Light_brightness будет присваиваться значение 1 каждый вечер в 9 часов, и значение 100 каждое утро в 6 часов. Осталось только применить это в основной цепи. Для этого у нас уже есть блок Light Brightness Adjustment

Для которого нам только нужно изменить настройку, чтобы он присваивал не константу, а значение переменной flow.Light_brightness.

Финальный результат

Итоговый flow, очищенный от отладочных блоков выглядит опрятно и чисто. При создании мы использовали только стандартные блоки из инсталляции Node-RED. На сайте flows.nodered.org, однако, существует более 800 дополнительных блоков и библиотек, которые позволяют добавить множество различных вещей.




Hall Light Sensor
connecting
Light Threshold Selector
Enable Light
Disable Light
5
100
Movement detected?
Light Enabled?
Light Brightness Adjustment
Hall Light Dimmer
connecting
Hall Motion Sensor
connecting
Switch off delay
OPEN
CLOSED
msg.payload
Reduce Brightness at 21:00
Light Brightness Adjustment
Normal Brightness at 6:00
flow.Light_enabled
Light_enabled

А вот и json код, который можно легко импортировать в любую версию Node-RED и протестировать.

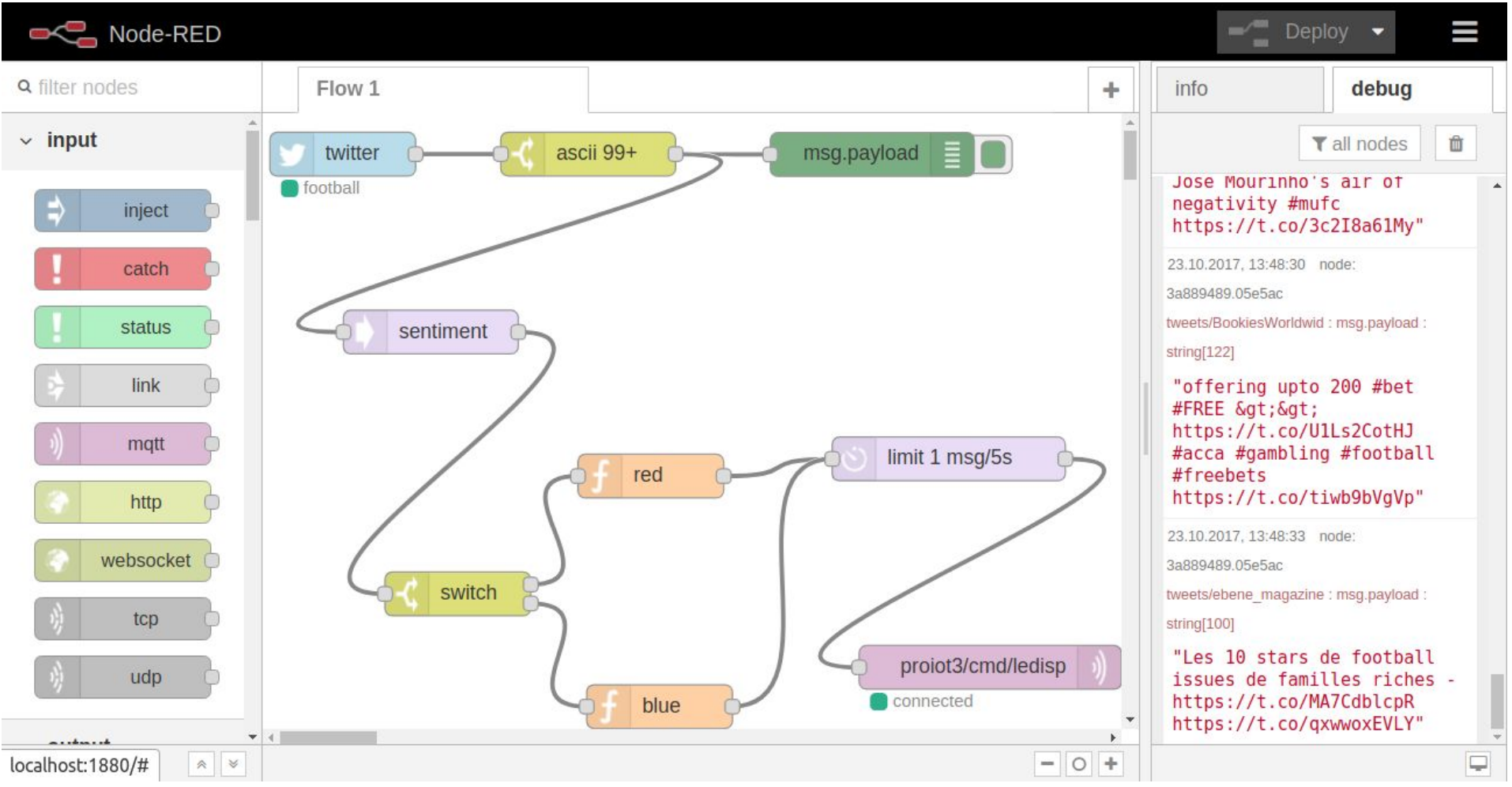

```
[{"id":"5afd41b4.d61318","type":"switch","z":"2384634b.17767c","name":"Movement detecte
d?","property":"payload","propertyType":"msg","rules":[{"t":"eq","v":"OPEN","vt":"str"
}],"checkall":"false","outputs":1,"x":562,"y":285,"wires":[["381b0d6d.a0bd7a"]]},{"id":
"35bac8e.57dd5b8","type":"trigger","z":"2384634b.17767c","op1":"5","op2":"0","op1type":
"nul","op2type":"val","duration":"8","extend":false,"units":"s","reset":"OPEN","name":
"Switch off delay","x":750,"y":373,"wires":[["e995e130.1e2118","af1f191f.498098"]]},{"i
d":"d85623d1.29b058","type":"change","z":"2384634b.17767c","name":"Light Brightness Adj
ustment","rules":[{"t":"set","p":"payload","pt":"msg","to":"Light_brightness","tot":"fl
ow"}],"action":"","property":"","from":"","to":"","reg":false,"x":1013.9999389648438,
"y":284.63330078125,"wires":[["e995e130.1e2118","af1f191f.498098"]]},{"id":"934ff922.ca
34f","type":"inject","z":"2384634b.17767c","name":"","topic":"","payload":"OPEN","paylo
adType":"str","repeat":"","crontab":"","once":false,"x":258.5,"y":408,"wires":[["5afd41
b4.d61318"]]},{"id":"ea0e2e99.52a6f8","type":"inject","z":"2384634b.17767c","name":"",
"topic":"","payload":"CLOSED","payloadType":"str","repeat":"","crontab":"","once":false
,"x":269,"y":459,"wires":[["35bac8e.57dd5b8","5afd41b4.d61318"]]},{"id":"4187db59.93c2d
c","type":"mqtt in","z":"2384634b.17767c","name":"Hall Light Sensor","topic":"/myhome/s
tate/Lumin_Hall","qos":"2","broker":"bfc8eee2.a46c9","x":243,"y":146,"wires":[["c94e7c
4.849f48"]]},{"id":"c94e7c4.849f48","type":"switch","z":"2384634b.17767c","name":"Light
 Threshold Selector","property":"payload","propertyType":"msg","rules":[{"t":"lt","v":
"10","vt":"num"},{"t":"else"}],"checkall":"false","outputs":2,"x":517.3333129882812,"y"
:145.7166748046875,"wires":[["48e6a07a.962798"],["ca8b6623.f11c7"]]},{"id":"381b0d6d.a0
bd7a","type":"switch","z":"2384634b.17767c","name":"Light Enabled?","property":"Light_e
nabled","propertyType":"flow","rules":[{"t":"eq","v":"Yes","vt":"str"}],"checkall":"tru
e","outputs":1,"x":775.5,"y":285,"wires":[["d85623d1.29b058"]]},{"id":"48e6a07a.962798"
,"type":"change","z":"2384634b.17767c","name":"Enable Light","rules":[{"t":"set","p":"L
ight_enabled","pt":"flow","to":"Yes","tot":"str"}],"action":"","property":"","from":"",
"to":"","reg":false,"x":822,"y":109,"wires":[[]]},{"id":"ca8b6623.f11c7","type":"chang
e","z":"2384634b.17767c","name":"Disable Light","rules":[{"t":"set","p":"Light_enabled"
,"pt":"flow","to":"No","tot":"str"}],"action":"","property":"","from":"","to":"","reg":
false,"x":824.6666259765625,"y":177.51666259765625,"wires":[[]]},{"id":"b6ea27c1.c33cd"
,"type":"inject","z":"2384634b.17767c","name":"","topic":"","payload":"Light_enabled",
```

Итоги

В данной статье продемострировано и рассказано, как просто можно реализовать повседневные алгоритмы домашней автоматизации в среде Node-RED. Также показаны основные преимущества данной среды программирования, такие как:
— логичное графическое представление связей и функций
— простоту программирования и легкость отладки пользовательских сценариев
— аппаратную и платформенную независимость полученных алгоритмов – данный сценарий будет одинаково хорошо работать и с OpenHAB, и с ioBroker и с любыми другими платформами умного дома, которые поддерживают протокол MQTT.
— простоту обмена готовыми алгоритмами между пользователями благодаря Copy-Paste JSON кода и наличию онлайн платформы для обмена удачными решениями.

Node-RED может и многое другое – например получать погоду из интернета, рассылать уведомления на твиттер или работать с термостатами Nest. С его помощью можно даже создать бота для Telegram. И на основе этого можно создать множество других интересных и полезных алгоритмов автоматизации.

На рисунке ниже Node-RED читает сообщения Твиттера на тему футбола, пропускает только сообщения в кодировке ASCII и с длиной более 99 символов, анализирует тональность текста и периодически выводит всё это дело на устройство с поддержкой протокола MQTT, разработанное в предыдущем материале. Красные лампочки указывают на грустные сообщения Твиттера, синие на радостные.



Управление выключателем xiaomi с реле с помощью кнопки xiaomi

- однократное - включаем первую нагрузку
- двойное - включаем вторую нагрузку
- длинное - выключаем все