射频网关 (433mhz/315mhz)

更换有源接收器模块

在 0.9.7 版本中，支持在 RF、RF2、RTL_433 和 Pilight 接收器模块之间切换有源信号接收器和解码器的能力。

切换有源接收模块

有源接收器模块可在 RF、RF2、RTL_433 和 Pilight 网关模块之间切换，无需重新部署 openMQTTGateway 包即可更改信号解码器。将 JSON 消息发送到所需接收器的命令主题将更改活动接收器模块。

要启用 RF 网关模块，请将 json 消息发送到 RF 网关模块命令主题，其中键为"活动"和任何值。此时的值被忽略。

例子： `mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTto433" -m '{"active":true}'`

要启用 PiLight 网关模块，请向 PiLight 网关模块命令主题发送一条 json 消息，其中键为"活动"和任何值。此时的值被忽略。

例子： `mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTtoPilight" -m '{"active":true}'`

要启用 RF2 网关模块，请向 RF2 网关模块命令主题发送一条 json 消息，其中键为"活动"和任何值。此时的值被忽略。

例子： `mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTtoRF2" -m '{"active":true}'`

要启用 RTL_433 网关模块，请将 json 消息发送到 RTL_433 网关模块命令主题，其中键为"活动"和任何值。此时的值被忽略。

例子： `mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTtoRTL_433" -m '{"active":true}'`

状态信息

openMQTTGateway 状态消息包含一个键 `actRec` ，它是当前活动的接收器模块。

1 - PiLight 2 - RF 3 - RTL_433 4 - RF2

基于 RCSwitch 的网关

从射频信号接收数据

使用 mosquitto 订阅所有消息或打开您的 MQTT 客户端软件：

```
sudo mosquitto_sub -t +/# -v
```

通过按下遥控按钮或其他按钮生成您的射频信号，您应该会看到：

禁用传输功能以保护 PIN

要禁用传输功能以允许使用另一个引脚，请将以下内容添加到 config_rf.h 文件：

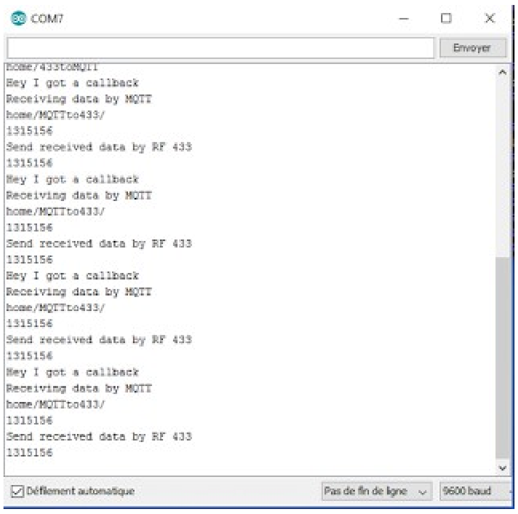
```
#define RF_DISABLE_TRANSMIT
```

通过 MQTT 发送数据以将其转换为射频信号

```
mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTto433" -m '{"value":1315156}'
```

此命令将通过 RF 发送代码 1315156 并使用默认参数（协议 1，延迟 350）

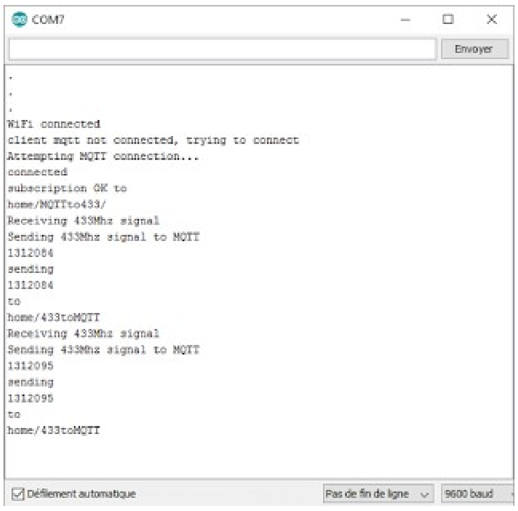
MQTT发布数据时收到的Arduino IDE串口数据



```
home/433toMQTT
Hey I got a callback
Receiving data by MQTT
home/MQTTto433/
1315156
Send received data by RF 433
1315156
Hey I got a callback
Receiving data by MQTT
home/MQTTto433/
1315156
Send received data by RF 433
1315156
Hey I got a callback
Receiving data by MQTT
home/MQTTto433/
1315156
Send received data by RF 433
1315156
Hey I got a callback
Receiving data by MQTT
home/MQTTto433/
1315156
Send received data by RF 433
1315156
```

我们看到 Arduino 在 MQTT 主题“MQTTto433”上接收到值 1315156 并通过 RF 发送数据

433Mhz接收数据时收到的Arduino IDE串口数据



```
.
.
.
WiFi connected
client mqtt not connected, trying to connect
Attempting MQTT connection...
connected
subscription OK to
home/MQTTto433/
Receiving 433Mhz signal
Sending 433Mhz signal to MQTT
1312084
sending
1312084
to
home/433toMQTT
Receiving 433Mhz signal
Sending 433Mhz signal to MQTT
1312095
sending
1312095
to
home/433toMQTT
```

通过具有高级射频参数的 MQTT 发送数据

射频发送支持三个高级参数；比特长度、RF 协议和 RF 脉冲长度，如果你想使用不同的 RCswitch 协议，在你的有效载荷中放入协议号 2，“协议”：2。

如果您想在主题"长度"中使用不同于 24 的位数：例如 24

示例： `mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTto433" -m '{"value":1315156,"protocol":2,"length":24,"delay":315}'` 将使 RCSwitch 使用协议 2，脉冲长度为 315ms，位数为 24

重复射频信号 OpenMQTTGateway 接收

为了重复网关接收到的射频信号，一旦在config_RF.h中将以下参数设置为true

```
#define repeatRFwMQTT true
```

多次重复射频信号

您可以向 MQTTto433 JSON 消息添加"重复"键/值以覆盖默认的重复次数。

例子： `home/OpenMQTTGateway/commands/MQTTto433 {"value":1315156,"protocol":1,"length":24,"delay":317, "repeat":10}`

设置 CC1101 收发模块的发送和接收频率

CC1101 模块的默认发送频率为 433.92 Mhz，可以通过在发送消息中包含频率来更改此频率。参数是 `mhz` ，有效值为 300-348 Mhz、387-464Mhz 和 779-928Mhz。实际频率支持取决于您的 CC1101 板。

```
home/OpenMQTTGateway/commands/MQTTto433 {"value":1150,"protocol":6,"length":12,"delay":450,"repeat":8,"mhz":303.732}
```

CC1101 模块的默认接收频率为 433.92 Mhz，可以通过发送带有该频率的消息来更改。参数是 `mhz` ，有效值为 300-348 Mhz、387-464Mhz 和 779-928Mhz。实际频率支持取决于您的 CC1101 板

```
home/OpenMQTTGateway/commands/MQTTto433 {"mhz":315.026}
```

收到的消息将包括频率，当以不同的频率发送时，模块随后返回接收频率。即在 303.732 Mhz 上发送消息，然后在 433.92 Mhz 上接收消息

```
{"value":4534142,"protocol":6,"length":26,"delay":356,"mhz":315.026}
```

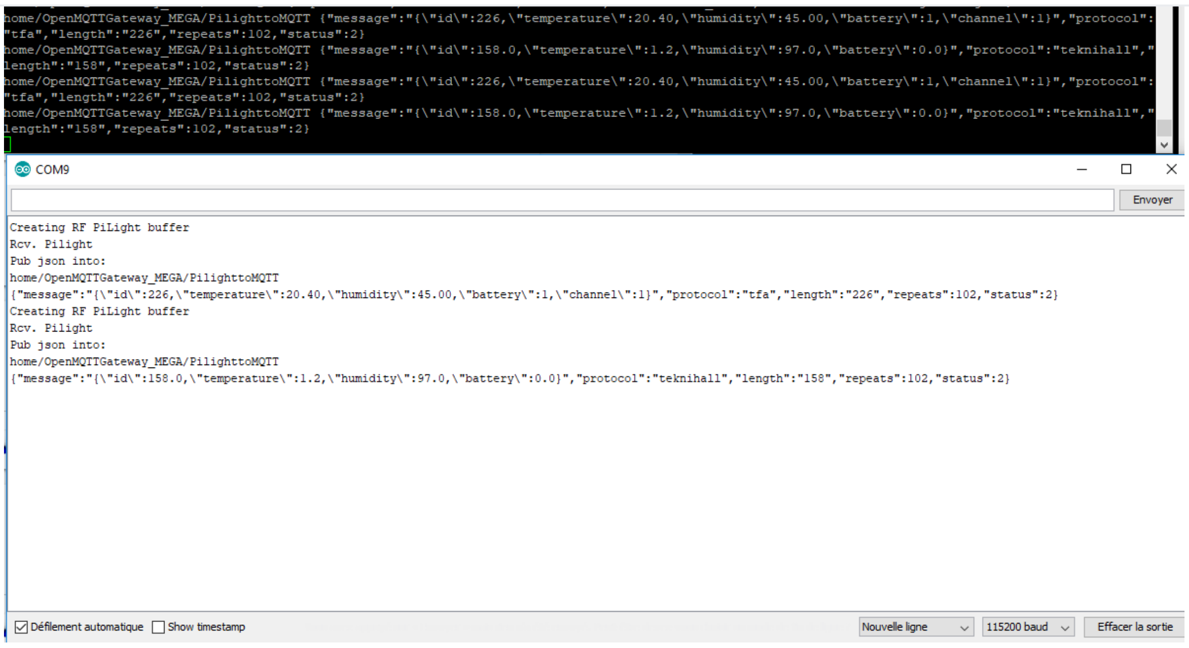
导光网关

从射频信号接收数据

使用 mosquitto 订阅所有消息或打开您的 MQTT 客户端软件：

```
sudo mosquitto_sub -t +/# -v
```

通过按下遥控按钮或其他按钮生成您的射频信号，您将看到：



```
home/OpenMQTTGateway_MEGA/PilighttoMQTT {"message":"{\"id\":226,\"temperature\":20.40,\"humidity\":45.00,\"battery\":1,\"channel\":1}","protocol":
"tfa","length":"226","repeats":102,"status":2}
home/OpenMQTTGateway_MEGA/PilighttoMQTT {"message":"{\"id\":158.0,\"temperature\":1.2,\"humidity\":97.0,\"battery\":0.0}","protocol":"teknihall","
length":"158","repeats":102,"status":2}
home/OpenMQTTGateway_MEGA/PilighttoMQTT {"message":"{\"id\":226,\"temperature\":20.40,\"humidity\":45.00,\"battery\":1,\"channel\":1}","protocol":
"tfa","length":"226","repeats":102,"status":2}
home/OpenMQTTGateway_MEGA/PilighttoMQTT {"message":"{\"id\":158.0,\"temperature\":1.2,\"humidity\":97.0,\"battery\":0.0}","protocol":"teknihall","
length":"158","repeats":102,"status":2}
```

```
Creating RF PiLight buffer
Rcv. Pilight
Pub json into:
home/OpenMQTTGateway_MEGA/PilighttoMQTT
{"message":"{\"id\":226,\"temperature\":20.40,\"humidity\":45.00,\"battery\":1,\"channel\":1}","protocol":"tfa","length":"226","repeats":102,"status":2}
Creating RF PiLight buffer
Rcv. Pilight
Pub json into:
home/OpenMQTTGateway_MEGA/PilighttoMQTT
{"message":"{\"id\":158.0,\"temperature\":1.2,\"humidity\":97.0,\"battery\":0.0}","protocol":"teknihall","length":"158","repeats":102,"status":2}
```

☑ Défilement automatique ☐ Show timestamp — Nouvelle ligne — 115200 baud — Effacer la sortie

通过 MQTT 发送数据以传输射频信号

使用已知协议

在 `mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTtoPilight" -m '{"message":"{\"systemcode\":12,\"unitcode\":22,\"on\":1}","protocol":"elro_400_switch"}'`

离开 `mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTtoPilight" -m '{"message":"{\"systemcode\":12,\"unitcode\":22,\"off\":1}","protocol":"elro_400_switch"}'`

这些命令将通过射频传输信号以启动 elro_400 开关。

使用原始信号

You can transmit raw signal data by using the "raw" protocol. This uses the Pilight pulse train string format. One such example string, representing a transmission for Nexus protocol weather stations, looks like this: `c:03020202010102020102010101010101010202020201020102020202020101010201010202;p:500,1000,2000,4000;r:12@` . This string represents pulses and gaps directly.

Each number in the list after `p:` that ends with `;` stands for pulse and gap lengths in microseconds (µs). In this example, we have a list containing lengths of 500µs, 1000µs, 2000µs, and 4000µs.

Each number after `c:` and ended by `;` represents a code that references the `p:` list by index. In this example, the first 4 numbers after `c:` are 0, 3, 0, and 2, which reference `p:` [0] = 500, `p:` [3] = 4000, `p:` [0] = 500, and `p:` [2] = 2000, respectively. In the language of digital radio transceiving, the most basic unit is usually a pulse and gap pair; in other words, 0s and 1s are represented by a pulse followed by a gap (lack of pulse) and the time lengths of these pulses and gaps. Different protocols have different pulse lengths and gap lengths representing 0, and a different one representing 1. Because of this pulse-gap nature, the codes in `c:` must be taken as pairs; the first number in a pair represents the length of the pulse, and the second number the subsequent gap. In this example, the first pair, 03, represents a

The number after `r:` represents how many times the message in the string is to be repeated. The `r:` block is optional. The default number of repeats if `r:` is not specified is 10. Greater than about 100 repeats will cause a crash due to memory usage. If this example were written without specifying repeats, it would look like this:

```
{"raw":"c:03020202010102020102010101010101010202020201020102020202020101010201010202;p:500,1000,2000,4000@"}
```

The entire string must end in a `@` . Each block must end in a `;` , but if it is the last block in the string, the `@` replaces the `;` . Since the `r:` block is optional, this last block could be either `p:` or `r:` .

The JSON for the MQTT message to `home/OpenMQTTGateway/commands/MQTTtoPilight` should specify the pulse train string as the value for the "raw" key:

```
{"raw":"c:03020202010102020102010101010101010202020201020102020202020101010201010202;p:500,1000,2000,4000;r:1
```

e.g. `mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTtoPilight" -m`
`'{"raw":"c:03020202010102020102010101010101010202020201020102020202020101010201010202;p:500,1000,2000,4000;r:1`

RF with SONOFF RF BRIDGE

Receiving data from RF signal

Subscribe to all the messages with mosquitto or open your MQTT client software:

```
sudo mosquitto_sub -t +/# -v
```

Generate your RF signals by pressing a remote button or other and you will see:

```
home/OpenMQTTGateway/SRFBtoMQTT {"raw":"2B660186042E00E7E5","value":"59365","delay":"1111'
```

The first parameter is the raw value extracted from the RF module of the Sonoff bridge. The data are in hexadecimal and correspond to the details below: https://www.itead.cc/wiki/images/5/5e/RF_Universal_Transeceive_Module_Serial_Protocol_v1.0.pdf OpenMQTTGateway process the raw value to extract the other decimal values that can be reused to reproduce a signal (raw value can also be reused).

NOTE: currently the device doesn't receive correct values from Switches remote control

Send data by MQTT to convert it on RF signal

```
mosquitto_pub -t "home/OpenMQTTGateway/commands/MQTTtoSRFB" -m '{"value":1315156}'
```

This command will send by RF the code 1315156 and use the default parameters: Repeat = 1 Low time= 320 High time= 900 SYNC = 9500

Send data by MQTT with advanced RF parameters

RF bridge sending support four advanced parameters; Repeat, Low time, High time & Sync if you want to repeat your signal sending put into your json payload "repeat":2, 2 means 2 repetitions of signal if you want to use a high time of 845 put inside your json payload "Thigh":845

if you want to use a sync time of 9123 put inside your json payload "Tsyn":9123

Example: `mosquitto_pub -t home/OpenMQTTGateway/commands/MQTTtoSRFB/Tlow_315/Thigh_845/Tsyn_9123 -m '{"value":"33151562","delay":"9123","val_Thigh":"845","val_Tlow":"315"}'` will make RF Bridge send a signal with the use of listed parameters 315, 845, 9123...

`mosquitto_pub -t home/OpenMQTTGateway/commands/MQTTtoSRFB/Raw -m '{"raw":"267A013603B6140551"}'` will make RF Bridge send a signal with the use of advanced parameters defined in the raw string

RF2 gateway KAKU

RF2 gateway enables to send command to RF devices with the KAKU protocol. DIO chacon devices are an example. It uses the same pinout as the RF gateway and both gateways can be used on the same setup.

Receiving RF codes with the KAKU protocol is not compatible with ZgatewayRF , so as to get the code of your remotes you should comment ZgatewayRF in User_config.h. Transmitting can be done with both ZgatewayRF and ZgatewayRF2

Receiving data from KAKU signal

Subscribe to all the messages with mosquitto or open your MQTT client software:

```
sudo mosquitto_sub -t +/# -v
```

Generate your RF signals by pressing a remote button or other and you will see :

```
home/OpenMQTTGateway/RF2toMQTT
{"unit":0,"groupBit":0,"period":273,"address":8233228,"switchType":0}
```

Send data by MQTT to convert it on KAKU signal

Once you get the infos publish the parameters with mqtt like that for off:

```
mosquitto_pub -t home/OpenMQTTGateway/commands/MQTTtoRF2 -m "
{"unit":0,"groupBit":0,"period":273,"address":8233228,"switchType":0}"
```

for on:

```
mosquitto_pub -t home/OpenMQTTGateway/commands/MQTTtoRF2 -m "
{"unit":0,"groupBit":0,"period":273,"adress":8233228,"switchType":1}"
```

rtl_433 device decoders

This feature is only available on a ESP32 based device with a CC1101 transceiver connected due to the resource requirements of the rtl_433 device decoders. At the present time only Pulse Position Modulation (OOK_PPM) and Pulse Width Modulation (OOK_PWM) based decoders are available.

```
[illegible]
Registering protocol [4] "Acurite 986 Refrigerator / Freezer Thermometer"
Registering protocol [5] "Acurite 606TX Temperature Sensor"
Registering protocol [6] "Acurite 00275rm,00276rm Temp/Humidity with optional probe"
Registering protocol [7] "Acurite 590TX Temperature with optional Humidity"
Registering protocol [8] "Akhan 100F14 remote keyless entry"
Registering protocol [9] "AlectoV1 Weather Sensor (Alecto WS3500 WS4500 Ventus W155/W044 (
Registering protocol [10] "Ambient Weather TX-8300 Temperature/Humidity Sensor"
Registering protocol [11] "Auriol AFW2A1 temperature/humidity sensor"
Registering protocol [12] "Auriol HG02832, HG05124A-DCF, Rubicson 48957 temperature/humid
Registering protocol [13] "BlueLine Power Monitor"
Registering protocol [14] "Blyss DC5-UK-WH"
Registering protocol [16] "Bresser Thermo-/Hygro-Sensor 3CH"
Registering protocol [18] "Burnhard BBQ thermometer"
Registering protocol [19] "Calibeur RF-104 Sensor"
Registering protocol [20] "Cardin S466-TX2"
Registering protocol [21] "Chuango Security Technology"
Registering protocol [22] "Companion WTR001 Temperature Sensor"
Registering protocol [25] "Ecowitt Wireless Outdoor Thermometer WH53/WH0280/WH0281A"
Registering protocol [26] "Eurochron EFTH-800 temperature and humidity sensor"
Registering protocol [30] "Esperanza EWS"
Registering protocol [32] "Fine Offset Electronics, WH2, WH5, Telldus Temperature/Humidity
Registering protocol [33] "Fine Offset Electronics, WH0530 Temperature/Rain Sensor"
Registering protocol [34] "Fine Offset WH1050 Weather Station"
Registering protocol [35] "Fine Offset Electronics WH1080/WH3080 Weather Station"
Registering protocol [37] "FT-004-B Temperature Sensor"
Registering protocol [38] "Generic wireless motion sensor"
Registering protocol [39] "Generic Remote SC226x EV1527"
Registering protocol [40] "Generic temperature sensor 1"
Registering protocol [41] "Globaltronics QUIGG GT-TMBBQ-05"
Registering protocol [42] "Globaltronics GT-WT-02 Sensor"
Registering protocol [43] "Globaltronics GT-WT-03 Sensor"
Registering protocol [44] "Microchip HCS200 KeeLoq Hopping Encoder based remotes"
Registering protocol [45] "Honeywell ActivLink, Wireless Doorbell"
Registering protocol [46] "HT680 Remote control"
Registering protocol [47] "inFactory, nor-tec, FreeTec NC-3982-913 temperature humidity s
Registering protocol [49] "Interlogix GE UTC Security Devices"
Registering protocol [51] "Kedsum Temperature & Humidity Sensor, Pearl NC-7415"
Registering protocol [52] "Kerui PIR / Contact Sensor"
Registering protocol [53] "LaCrosse TX Temperature / Humidity Sensor"
Registering protocol [54] "LaCrosse TX141-Bv2, TX141TH-Bv2, TX141-Bv3, TX141W, TX145wsdth
Registering protocol [55] "LaCrosse/ELV/Conrad WS7000/WS2500 weather sensors"
Registering protocol [56] "LaCrosse WS-2310 / WS-3600 Weather Station"
Registering protocol [58] "Maverick et73"
Registering protocol [60] "Missil ML0757 weather station"
Registering protocol [64] "Nexus, FreeTec NC-7345, NX-3980, Solight TE82S, TFA 30.3209 ter
Registering protocol [66] "Opus/Imagintronix XT300 Soil Moisture"
Registering protocol [67] "Oregon Scientific SL109H Remote Thermal Hygro Sensor"
Registering protocol [69] "Philips outdoor temperature sensor (type AJ3650)"
Registering protocol [70] "Philips outdoor temperature sensor (type AJ7010)"
Registering protocol [71] "Prologue, FreeTec NC-7104, NC-7159-675 temperature sensor"
Registering protocol [73] "Quhwa"
Registering protocol [75] "Rubicson Temperature Sensor"
Registering protocol [76] "Rubicson 48659 Thermometer"
Registering protocol [77] "Conrad S3318P, FreeTec NC-5849-913 temperature humidity sensor
Registering protocol [78] "Silvercrest Remote Control"
Registering protocol [81] "Solight TE44/TE66, EMOS E0107T, NX-6876-917"
Registering protocol [82] "Springfield Temperature and Soil Moisture"
Registering protocol [83] "TFA Dostmann 30.3221.02 T/H Outdoor Sensor"
Registering protocol [84] "TFA Drop Rain Gauge 30.3233.01"
Registering protocol [85] "TFA pool temperature sensor"
Registering protocol [86] "TFA-Twin-Plus-30.3049, Conrad KW9010, Ea2 BL999"
Registering protocol [87] "Thermopro TP11 Thermometer"
Registering protocol [88] "Thermopro TP08/TP12/TP20 thermometer"
Registering protocol [90] "TS-FT002 Wireless Ultrasonic Tank Liquid Level Meter With Tempe
Registering protocol [91] "Visonic powercode"
Registering protocol [92] "Waveman Switch Transmitter"
Registering protocol [93] "WG-PB12V1 Temperature Sensor"
Registering protocol [94] "WS2032 weather station"
Registering protocol [95] "Hyundai WS SENZOR Remote Temperature Sensor"
Registering protocol [96] "WT0124 Pool Thermometer"
Registering protocol [98] "X10 Security"
```

Change receive frequency

Default receive frequency of the CC1101 module is 433.92 Mhz, and this can be can changed by sending a message with the frequency. Parameter is `mhz` and valid values are 300-348 Mhz, 387-464Mhz and 779-928Mhz. Actual frequency support will depend on your CC1101 board

```
home/OpenMQTTGateway/commands/MQTTtoRTL_433 {"mhz":315.026}
```

Minimum Signal Strength

Default minimum signal strength to enable the receiver is -82, and this setting can be changed with the following command.

```
home/OpenMQTTGateway/commands/MQTTtoRTL_433 {"rssi":-75}
```

Enable rtl_433 device decoder verbose debug

This function does not work when all available decoders are enabled and triggers an out of memory restart.

```
home/OpenMQTTGateway/commands/MQTTtoRTL_433 {"debug":4}
```

Retrieve current status of receiver

```
home/OpenMQTTGateway/commands/MQTTtoRTL_433 {"status":1}
```

```
{"model":"status",
"protocol":"debug",
"debug":0,                  - rtl_433 verbose setting
"duration":11799327,        - duration of current signal
"Gap length":-943575,       - duration of gap between current signal
"signalRssi":-38,           - most recent received signal strength
"train":1,                  - signal processing train #
"receiveMode":0,              - is the receiver currently receiving a signal
"currentRssi":-89,            - current rssi level
"minimumRssi":-82,            - minimum rssi level to start signal processing
"pulses":0,                   - how many pulses have been recieved in the current signal
"StackHighWaterMark":5528,    - ESP32 Stack
"freeMem":112880}             - ESP32 memory available
```

编辑这个页面

最后更新时间： 2022 年 2 月 8 日，上午 12 时 42 分 37 秒

← 控制器 设备 →