HOMEBRIDGE-HTTP

and RabbitMQ

Version 1.0

Homebridge-http and RabbitMQ extension allows you to send command strings to your localhost RabbitMQ server instead of making HTTP requests. This tutorial assumes that you have https://github.com/nfarina/homebridge program set up and working.

First step: tweak homebridge-http

Before doing anything, make sure nodejs, npm, and homebridge are installed. Run homebridge in terminal and test if your iPhone can see it (using Apple's new "Home" app). Close homebridge and quit terminal.

1. Install homebridge-http module (information on how to do that: here)

2. Go to **/usr/local/lib/node_modules/homebridge-http/** (or wherever it is) and open **index.js** with a simple text editor

3. Press CTRL+A and backspace to delete everything. Go to page 4 of this PDF, copy pink code and paste it in index.js. Save and close.

Note: If you already made some changes to index.js and want to keep them, follow this link

Setup homebridge config.json file:

Adding virtual accessories that send commands to rabbitmq is very simple. Just open your config.json file (located in home/.homebridge) and add any of these three accessories inside the accessories array (you can change blue text to anything you want).

Switch:

```
{
        "accessory":          "Http",
        "service":            "RabbitmqSwitch",
        "name":               "Swimming Pool Pump",
        "powerState_action":  "rabbitmq",
}
```

Lightbulb:

```
{
        "accessory":          "Http",
        "service":            "RabbitmqSwitch",
        "name":               "Lamp",
        "powerState_action":  "rabbitmq",
        "brightness_action":  "rabbitmq"
}
```

Thermostat:

```
{
        "accessory":                          "Http",
        "service":                           "RabbitmqSwitch",
        "name":                               "Cooler",
        "targetHeatingCoolingState_action":   "rabbitmq",
        "targetTemperature_action":           "rabbitmq",
        "temperatureDisplayUnits_action":     "rabbitmq"
}
```

"rabbitmq" action means that a string containing the accessory name, method, and value will be sent to the rabbitmq localhost server whenever you change something with Siri or an app on your iOS device. The string sent to the "hap" rabbitmq queue will look something like this: "update Cooler targetTemperature 16.3".

Listen to commands in your app

On RabbitMQ website, you can find tutorials for your favorite programming languages (Java, Pyton, NodeJS, Swift...), how to include RabbitMQ library into your project and how to listen to the queues (in this case the "hap" queue).

In your app you will then develop a function that will accept this string (example: "update Cooler targetTemperature 16.3") and do something like controling gpio pins, opening a program or playing a sound.

🎁 Bonus: Sending terminal commands

If you don't wanna hassle with rabbitmq, you can simply execute terminal commands. Just replace `"rabbitmq"` with any terminal command you'd like.

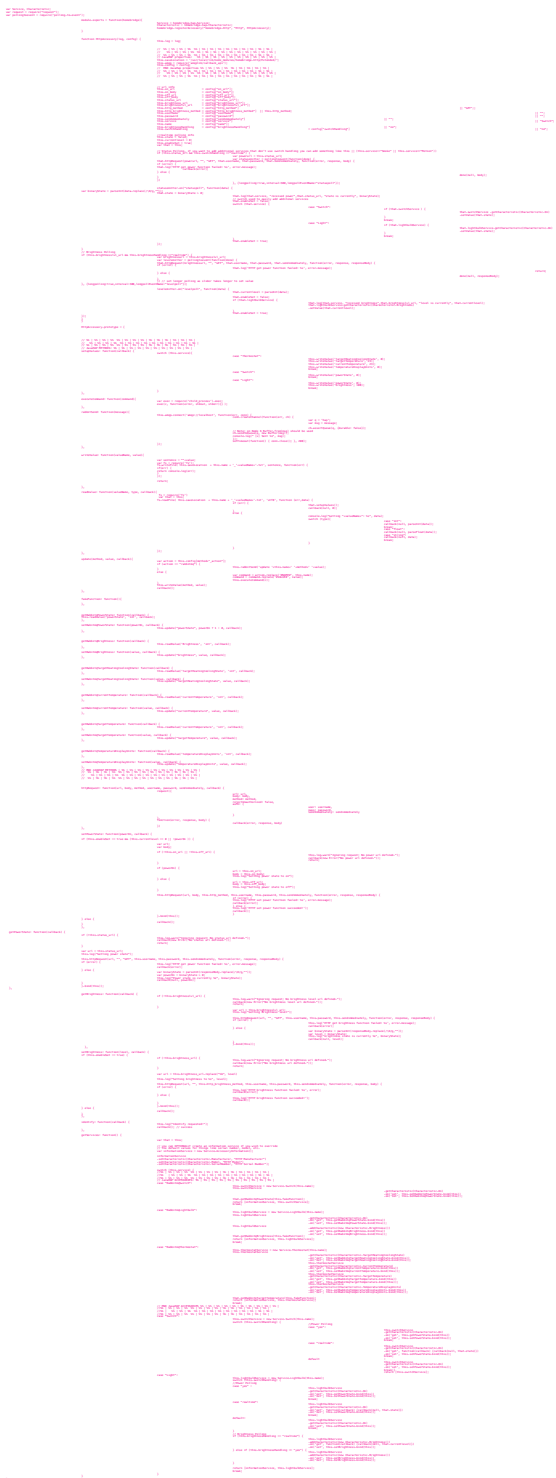
Example1: `"open -a iTunes"`
Example2: `"sudo shutdown -h now"`.

You can also use variables in those terminal commands like #NAME# (the name of accessory) and #VALUE# (powerState, brightness, targetTemperature). Hint: in Example2, there is a java app that accepts argument "-setLights" and a number.

Example1: `"osascript -e 'set Volume #VALUE#'"`
Example2: `"java -jar ChristmasLights.jar -setLights #VALUE#'"`

The #VALUE# variable is always a number:
- PowerState: 0 = off, 1 = on
- Brightness: 0 - 100
- TargetTemperature: any number
- HeatingCoolingStates: 0 = off, 1 = heat, 2 = cool, 3 = auto
- TemperatureDisplayUnits: 0 = Celsius, 1 = Fahrenheit

//
//PAGE 4: - SIMPLY COPY THIS WHOLE PAGE
//Tweaked …homebridge-http/index.js contents:
//The code below is scrambled so you can copy easily:



//
//
// Copy these lines too