Grüezi YouTubers. Here is the guy with the Swiss accent. With a new episode around sensors and microcontrollers.

In video #140 we hacked the new IKEA Tradfri Smart lighting system using the remote control as a basis. And, as I said already in video #144: It worked, but had some major disadvantages. Today, we will use an IKEA gateway and do it right!

* We will eliminate the problems of the current hack
* We will install and pair the gateway with two different bulbs
* We will learn how the system communicates
* We will switch the lights using node-red. Like that, we can use MQTT or many other tools to switch the lights

If you are interested

* We will switch the lights with a normal Linux command
* We will install a Python wrapper. You can use this wrapper as a basis for your own programs
* We will see the CoAP protocol , which was introduced in the last video, in action
* At the end, as promised, I will reveal a “hot” secret about the IKEA bulbs.

The major problem with the last hack was, that the remote control only toggled the on and off switch. So, we were never sure, if the bulb was really on or off. This works ok if you switch by hand, because you just press the button again, if something went wrong. For home automation systems, however, we need a concept with clear commands like “off” or “on”. And it would be nice, if we could read the actual state back to check, if everything went right.

Ikea sells a Gateway for its lighting system, which promises to be able to do exactly that. You get it together with a Smartphone app for around 30 dollars.

If we have a look at the architecture, we can start with the bulbs. They are connected to a remote control through the Zigbee protocol. In the first IKEA video, I thought, it would be a good idea to hack this Zigbee protocol. But the price for a Zigbee adapter is similar to a IKEA gateway. So, I decided to buy one of those. I leave a link in the description, if you are interested in hacking the Zigbee protocol. The Gateway is also connected via Zigbee to the bulbs. And we can connect a Smartphone app to the gateway. This app looks nice, but I do not see a big advantage by replacing a remote control with an app. It is not very user friendly. The app communicates via your home network and the CoAP protocol with the gateway. So, the attack plan is to connect a Raspi to the gateway and emulate the commands normally sent by the app. And fortunately, there is information around on how to do that.

But let’s start with building the original IKEA setup: We have to install the gateway and connect it to mains and to our network. Be aware, that this has to be done via an Ethernet cable. Now, we have to install the app, and connect it to the gateway. Here, we need the code printed on the back of the gateway. It is the key for the CoAP encryption. Next, we have to pair a remote control with the gateway, and, as a last step, we have to pair the remote with each bulb. Now, everything is working, and we can switch the bulbs with our app, either each bulb alone, or as a group.

Now we have to find the IP address of the gateway. I usually use Advanced IP scanner to do so. Fortunately, the MAC address is also printed on the Gateway. So, we just have to search for it in our IP scanner.

Now, we should already be able to ping the gateway to check communication. First step done.

Next, we have to make our Raspberry capable to use the secure CoAPS protocol. You find a few “how-to’s” in the internet. I enclose a link in the comments.

But fortunately, I found an easier way: We use node-red, and I suggest you run Peter Scargill’s script to install Node-Red, Mosquito, and many other goodies as a prerequisite. And if you do not know this powerful script, you might consider watching video #126 first. If you do not know Node-Red, you can also watch this video first.

Then, you start your browser and navigate to node-red. In the palette, you find a contribution called node-red-contrib-Tradfri. Just install this contribution and you are done. A CoAP client is installed together with this contribution, as we will see later.

Now, we can include our Tradfri gateway into a flow. We find two different nodes: An output node called “tradfri” and a function node called “tradfri get”. Let’s start with “tradfri” to switch the lamps.

If we double click, as usual, we can configure the node. The first time, we have to configure also the gateway, here called “hub”. We give it a name and enter the security code from the bottom of the gateway. Then, we just click on the “coap-client-raspbian and the path to our CoAP client is entered. Please note this path down for the second part of the video.

After the definition of the gateway, we can press this button to auto-discover our bulbs. And really, the gateway announces two bulbs, a clear and an opal one. We can select either one, or also the whole group. This is due to the discovery function of CoAP we discussed in the last video. Cool. Now, we are ready to rumble. We inject a “on” and a “off” message to the tradfri node and deploy the flow. And really, if we press “on, the bulb switches on. Really simple.

Now, we can build a second flow for the second lamp with “copy paste”. Just replace the bulb in the definition, and you are ready to go.

In the next step, we want to read the status of the lamps. We use the “tradfri get” node, and, because we already defined the gateway, it is available here. To read the status, we have to inject a string and we get a message in the debug window. If we have a close look at the message, we see the remote control and our two bulbs. And they are numbered from 65536 up. These numbers of the bulbs will be important for the second part of the video.

And we see here also the three parameters: ON, color, and brightness. These are the tree parameters we can influence either by our remote control or by the gateway. To do so, we have to inject the right payload into the Tradfri node. For the interested viewers, I wrote a small function which reads the input and decides, which message to send to the gateway. For example, if the input is “warm”, a message “color: warm is sent to the gateway. If the input is “half”, the message brightness: 128 is sent to the bulb. Brightness 255 would be full intensity. You find a link to the file for the flow in the comments, and in video #128 around minute 16 you can see, how simple it is to import this flow into your node red installation. And this flow reads the status of one particular bulb: Based on the status of the bulb, ON is either true or false.

To finish the node-red setup I use the big-timer node built by Peter Scargill. This is a very versatile node, and I set it up that it will switch the bulb on at dusk and switches it off at dawn. So, everything is done. Instead of Big-timer, you can connect your MQTT node to your flow if you want to switch the bulbs with MQTT messages. Or with twitter, or with an email… Nearly no limits!

Viewers, who want to stick with node-red can now finish. If you want to program yourself, or look a little under the hood, you can continue the video. We will now move to the CoAP client and also to a Python example program.

To do that, we have to navigate to the directory of the CoAP client we noted down in the step before. Then, we can execute a simple call:

./coap-client-raspbian -m get -u "Client\_identity" -k "djed8rbxLWVOdUWm" "coaps://192.168.0.15:5684/15001"

This command calls the CoAP client with the user “client identity” and the password from the back of the gateway. The address of the gateway is 192.168.0.15 and the port is now 5684, because we use CoAPS, the secure protocol. And 15001 is the number for the gateway. In the response, we find the three devices from before, numbered 65536, 37, and 38. If we add 65537 to the command, we get the information about one particular bulb . Here, the properties only have numbers. You find the details of these numbers in my blogpost.

If we issue the put command

./coap-client-raspbian -m put -u "Client\_identity" -k "djed8rbxLWVOdUWm" -e '{ "3311": [{ "5850": 1 }] }' "coaps://192.168.0.15:5684/15001/65537"

with some additional parameters, the bulb switches on or off.

If you want to use this client in a Python program, you can install a so called “Python Wrapper”. It consists of a few python programs and is installed in your home directory. If you navigate into its directory, you can use it with simple commands like

./tradfri-status.py

To discover the bulbs, or

./tradfri-lights.py -l 65537 -a power -v on

To switch one of the lamps.

That’s it for today. Summarized:

* The new hack can definitively switch the lamps on or off, warm or cool, and not, as the old hack, only toggle through the states
* We paired the new gateway with two different bulbs
* We know now how the system communicates and its parameters
* We installed the node-red contribution for Tradfri and developed flows to switch the lights and read their status. Based on these flows, you can easily connect your Tradfri system to MQTT or any other service
* We switched the lights with a normal Linux command using the CoAP client
* We installed a Python wrapper and used it to discover and control our bulbs
* And, of course, all was based on our new discovery: The CoAP or its secure sister, the CoAPS protocol

By the way: The Remote control still works in parallel with the Gateway!

Did I forget something? Oh, yes: The “hot” secret! Here it is.

I hope, this video was useful or at least interesting for you. If true, then like. Bye