In video @111 I started to hack the new Sonoff SC sensor device and in video #113 I continued. From there, we still have some work open:

We want to use the microphone to detect noise, for example if we are on holiday

We have to transfer our code to the Arduino and to the ESP on the SonoffSC

We want to finish the ESP code that it automatically reads the values created by the Arduino and transfers them as messages to an MQTT broker

We want to make the ESP8266 code IOTappstory.com compatible

So, let’s start.

First, we have to understand how the microphone works. For this, we have a short look at the diagram of the device and we see, that the microphone is connected to Analog Pin 2. Because microphones only deliver small voltages, two operational amplifiers amplify the signal and also filter some noise. I connect now my oscilloscope with A2. If it is quiet in the room, the signal is around 2.5 volt. But we see also periodic voltages of zero and 5 volt, even if it is quiet. Not good. Where do these spikes come from? Experienced ESP8266ers discover, that they might come from the Wi-Fi signal emitted by this chip. So, this would be a typical interference. To prove it, I switch the Wi-Fi of the ESP off. And here you see the result. The spikes are gone and we have a relatively clean signal.

This is not good news and we have to pay attention to that later on. So, let’s keep Wi-Fi off for the moment and watch the “real” signal. As expected, it is the amplified signal of the microphone without any conditioning. Whistling creates a relatively clean sine wave. If we measure now the voltage with analogRead(), we get all values from nearly zero to 5 volts. So, what is the right value? Our eyes quickly see, that we have to get the difference between the two peaks of the signal. To translate this into “Arduino code” I read many values, let’s say 50 values as fast as possible. During this time, I search for the maximum and the minimum. The chance, that I get close to the real maximum and the minimum is bigger, if I can read fast. After these 50 values, I subtract the minimum from the maximum and get the peak value. And then, I average these values to smooth the outcome.

So, we can upload our sketch to the SonoffSC and check, if it runs also there. This is done the normal way. Because we have no bootloader, I use the ICSP header and a USBASP programmer for uploading. You find many videos about how this is done. First, I upload a simple test sketch to check the readings of the A2 pin with serial plotter. I connect a FTDI to the serial of the Arduino and disconnect the Arduino from the ESP.

To create a constant signal, I use the waveform generator and a loudspeaker. The initial frequency is 1000 Hz. We see, that we get a relatively constant signal. And it changes with the intensity, which is good. But unfortunately, you still see the spikes.

To test, if these spikes do not come from my own wires soldered to the A2 pin, I de-solder everything. Unfortunately, the spikes do not disappear. So, the noise measurement cannot be used without synchronizing the Wi-Fi with the measurement of the noise. Otherwise, we would have too many false alarms. Because noise detection is not important for me, I leave this to somebody else. So we are ready to upload the real sketch to the Arduino and go back to the ESP.

Here, we have to connect it to an MQTT broker. Tinkerman uses the “AsyncMqttClient” library. I do not change that. And he also uses a library which deals with the whole serial connection. Which also works fine. So, I do not change it. He puts all definitions into a separate file called “general.h”. I like this concept. Here, we see, that we need quite a few definitions to deal with mqtt. For example, we have to define the server address and port. And we have to define all the topics. In the IOTappStory concept, we do not want to have these data hard coded in the sketch. We want to be able to change it later on. In the current version of IOTappstory.com, we use Wi-Fi Manager to do that. But in this case, it is not very handy to enter all these names and addresses by hand on your mobile. This is, why I store these parameters on the “ESP disk drive” called SPIFFS. I introduced this method in video #121.

But this is not enough. It is now also possible to download the content of the SPIFFS over the air from IOTappstory.com. So, you store the configuration on the IOTappstory.com and, if you changed it, your ESP device uploads it at the next opportunity to its internal disk, just as we do it with a new sketch. I will show you in a future video, how this is done. For now, just make sure you upload the SPIFFS file with the configuration as shown in the last video. Otherwise the sketch uses the default values of the sketch.

Now, we disconnect the Arduino programmer and connect the serial connections between the Arduino and the ESP and start the device. We still can monitor the traffic between the two devices using a FTDI connector. And if we go to cloudMQTT, we see the results coming.

Summarized,

We discovered, that the sonoff SC has two processors, an Arduino and a ESP8266

The Arduino is used to mesure the results of the Dust, the noise, the temperature and humidity, and the light sensor

The ESP8266 transports these values to the cloud

The connection between the two is done by serial connection

Tinkerman wrote an initial sketch to use this device and connect it to an MQTT broker.

I reduced some of the features of the sketch, and adaped it to IOTappstory.com

We also discovered how noise is measured and found a problem, which is not solved yet.

This is the end of this hack. It is now up to you to use this sensor node for your projects.

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