What if your IOT device would behave like your Smartphone or PC? You select an app in some place like Google play and your smartphone downloads and installs it directly from the net over the air? And in addition, it would also update to a new version, if appropriate? In video #78 I introduced such a new concept: The IOTappstore. Today, I will go a step further and will introduce the publicly available iotappstory.com service. Every app has its own story…

The first version was not easy to be implemented. Viewers asked me about a simplification and I promised it as a “Christmas Gift”. Fortunately, I got help from Onno, also one of my viewers. He is a web developer.

Together, we now can present our 2016 Christmas gift: The official launch of IOTappstory.com. Of cause, we hope, that it will write “history”!

Please keep in mind, that this is a hobby project and the first version. So, if you find errors or possibilities for enhancements, feel free to post a comment.

In the past, I published all my sketches for the ESP8266 on Github, you had to download them, sometimes install additional libraries, compile it, and upload it to your devices. And make sure, if you installed it in an awkward place, make sure you were able to update it over the air.

From today on, you can just login to iotappstory.com, select your board and the sketch you want, and press a button for about 3 seconds. A few seconds later, the sketch is installed over the air on your board and you can use it. Immediately. No hassle. And the update will also be loaded over the air. And you can change credentials and other parameters, also over the air. Cute!

Now this is only part of the story. If you are a maker or even a manufacturer of devices, you can upload your own compiled sketches to IOTappstory.com and use them either for you, or also for anybody else. So, issues like “how do I manage my ten ESP8266 devices in my Home Automation system are a thing of the past.

But let’s start with explaining the concept and the parts:

Since a few months, the ESP8266 can load its programs from a webserver. This is the functionality we use for appstory.com. If we create a sketch just for us and one device, we include the WiFi and other credentials and other parameters like web services, or names of devices, directly in our sketch. All ESP8266 examples are done like that. This works perfectly if we want to use the sketch just for us and only for one particular device in our home. If we have more than one device, we have to give each a different name, and maybe want to change their relationship later on. Also, if we want to give our “invention” to our colleague or if we want to use it in another house, we have to change the credentials and recompile the code. Even if we change the WiFi password, we have to recompile and Upload a new code to all our devices. I can imagine nicer tasks like that.

This I, why a library was introduced: The WiFimanager. With this library you can create code, where you can change credentials with your browser on the smartphone or the PC. You can also change any other constant of your device if you want. IotAppstory.com uses a fork of this library to do exactly that. This is the second important part of the infrastructure.

Summarized, you get code examples you can use for your own project which includes all the needed functionality to separate code from data, to download and update sketches over the air, and change credentials and parameters using your browser or Smartphone. And of course, you can use this without having your own webserver.

For a first test I prepared two sketches an loaded them to IOTappstory.com: A fast blink and a slow blink.

So, let’s try to load the slowblink example to one of our boards. To do that, we need a few steps and, at the beginning, it gets a little complicated. This is, because the 2.4 release of the ESP8266 in the Arduino IDE is still not released. So, we have to do a little patching. But I tried to make it as simple as possible.

The first step is the pre-implementation of the so called MD5 functionality into your programming environment. Please select directory C:\Users\”your user”\AppData\Local\Arduino15\packages\esp8266\hardware\esp8266. Here, you should find a directory “2.3.0-rc2”. If not, you should change the this part in the preferences part of your Arduino IDE from “stable” to staging. If you go now to the boards menu you should get version “2.3.0-rc2”. Now you just have to install it.

Now you can download the files from my github. There, you find a directory with the name “2.3.0-rc2”. If you copy this directory, it will replace a few files in your IDE. You have to accept this and step #1 is done. Fortunately, this step is only necessary once per PC.

The second step is only necessary if you want to use a board or module which was never used with IOTappstory.com or which runs a sketch which is not compatible to IOTappstory.com. I just unpack a brand new Wemos mini board for that.

Please take now the IOTappstoryLoader.ino sketch from github and, if you want, insert your default credentials here. This is for your comfort, because you do not need to enter them later manually. You can leave them also blank as I do now. Loased to the Wemos mini, this sketch will download the first sketch form IOTappstory.com. This step is only necessary once per device, if you always use sketches which comply the standards of IOTappstory.com. These sketches do not only have the ability to load a new version of your current code over the air, they also have the ability to download any other code from iotappstory.com which is compiled for their architecture.

After uploading this sketch, please press the reset button for this first time. This is a small “bug” of the ESP8266. If you do not do that, you will get later a wdt reset and a board which is blocked. Also there, usually, a reset helps.

Please keep the Serial cable connected and watch, what happens when the sketch starts.

First, it displays the MACaddress of your device. This is a unique address of a device connected to the internet and will be used to identify this device on IOTappstory.com. Next, it creates an Accesspoint with the name “initloader”. If we connect to this network, a splash screen opens up and we should see this screen. If not, please enter the address 192.168.4.1 as an address. If we go now to the “info Tab” we also find the Station macaddress and can copy-it for later usage.

In the “configuration” tab, we see all constants to be filled in. Fortunately, the information for the IOTappstory.com are already taken from the default. The WiFi credentials, however, have to be entered here, because we did not supply default values. You just select an available Network, enter the password, and press “save”. Now, you have to wait at least 10 seconds. In the meantime, we can watch in the serial monitor what happens. If you get a message “failed to connect”, your credentials were wrong.

After the 10 seconds, we press this link to check how it went. We see now, that the ESP is also connected to our local Wi-Fi. Everything ok. Now we just press “Exit portal” and the configuration is saved in EEPROM, as we see in the Serial monitor. We see also, that the loader cannot load any sketch. This is, because we did not define what to load.

Because this step is needed for all fresh boards, you might just load a few more boards for future usage…

So, step 2 is finished and we can go on to step #3: We have to enter the information about our board into IOTappstory.com. We go to iotappstory.com and, if you do not have an account, create one. It is free of charge for hobby usage. Then, you open up the control panel and go to devices. Because I just created a new account, there are no devices available. So, we have to add one or more new devices. You can give the board a name and even upload a picture for easy reference. Maybe you give it a name like “blue Wemos Mini” and put a blue point on it. Or you use a device, which is already built into one of your projects. I name it “blue Wemos”. Next, you select the board type and enter the Macaddress which was copied before. Make sure, that every number has 2 places. The other fields are optional. Press “save” and you are done. If you want to enter more than one device, press “save+1 and you can immediately enter the next one.

Currently, the new device is offline and we have to set it “online”. Now we can use it for the future.

Because the slowblink app is already available on the platform, we can directly start our first project using our device. We add a new project in the projects rider and give it also a name. Also here, you can upload a picture of your project. The project brings the device and the App together. So, we select the newly created device and the app we want to be downloaded. If we press now “save” we just have to set this project “online” and we are ready to go.

Later on, we will need a “flash” button. Unfortunately, the Wemos has no such button. So, I add a small button between GPIO0 or D3 and GND. I also had to add a 100nF capacitor to debounce the button. The value is not critical.

For the last step, we reset the Wemos mini.

Serial is still visible and we can see and understand, what happens. In the future, of course, we will not need a serial connection anymore. The OTA upload starts automatically. This is, while the infrastructure detected a difference between the sketch stored on your device and the sketch defined in iotappstoy.com. After a while, the ESP reboots, and the downloaded sketch starts to run. But, unfortunately, the LED does not start to blink. This is, because I wrote the sketch in a flexible way. You can select the pin for your LED after downloading of the sketch. So, your colleague could use a different pin than you. And because I did deliberately not enter a default pin,

Pin D8 or GPIO15 is selected.

Now, you have to press the flash button we built before for more than 6 seconds and the Wemos enters directly configuration mode and starts again its network. This time, its name is: slowblink.

If the sketch only would need Wi-Fi credentials, you would be good to go without this step and the Wemos would have started directly with blinking. But now, we have to connect to the “slowblink” network and enter the configuration again. Now, you see the new field “BlinkPin” and you can enter D0 to D7. If you enter a wrong string, or if you enter D3 (flash button), pin D8 will be selected. The rest of the fields are ok for now.

After “Save”, we again wait for the 10 seconds, and then press “Exit”. Only after this action, also the pin definition is stored in EEPROM for future use. If you do not wait the 10 seconds, it might not work properly.

The board reboots now automatically and starts with the new sketch. We see it in the Serial Monitor. Because we chose the internal LED, it starts to flash.

If you now connect a LED between the selected pin and 3.3v, it starts to flash. I connected the green LED to D1 and the red one to D2. If I go now back to the configuration menu with a long press of the flash button, the red LED starts to blink. Cool.

If you press the flash button for 3 seconds, the sketch again checks Appstory.com. But, because the MD5 sum of the sketch on your board and in Appstory.com is the same, no download is initiated this time.

Just for fun, you can now try to replace your slowblink by the fastblink sketch. Just change your project in iotappstory.com, and press the flash button for 3 seconds. This time, the sketch is replaced and, without entering the credentials or the pin again, the LED starts to blink fast. This is, because these constants are still stored in the EEPROM and can be reused by the new sketch.

I would suggest that you do this without Serial connection of your ESP. Just to show, that the update really comes over the air and

You find a few other example files like the SonoffReceiver and SonoffSender files of video #97 and the IFTTT button from video #101. The Sonoff receiver has to be loaded on a Sonoff device with 1M flash, the other two on a NodeMCU or an ESP-12E module with 4M.

This is the end of this episode. Later, I will show you, how you can build your own projects using this infrastructure.

I hope, this video was useful or at least interesting for you. Merry Christmas!