Tasmota Templates
Use these copy paste templates on your device, for which you will have to install Tasmota first.
https://dgit.in/may20-01

Tasmota 8.2 Elliot
Open source firmware for ESP8266 devices, with web based controls and more.
https://dgit.in/may20-02

SMART LIGHT AUTOMATION

Make a personal, and private home automation system, with full control over your smart bulb's hardware. No proprietary black box spying here!

Aaruni Kaushik | feedback@digit.in

A trademark feature of all futuristic high tech homes is smart appliances, and central control hubs. As examples take the lights that change colour depending on the time, weather, or when you come home or leave. Also, apps on your PC that control the automated system and your smartphone that can run a manual overrides whenever required! However, such convenience often comes at the cost of your own privacy. To get advertised functionality, all big companies have to transfer your data to their computers for operations. What if there was a way to get the best of both worlds? If that got your attention, continue reading, and by the end of this article, you'll have your own private smart infrastructure!

QUICK OVERVIEW

The process, in broad strokes, is simple. Your goal is to get automation functionality from a smart bulb, with all of your data never leaving the LAN. To do this, you need to set up your own command and control server with Home-Assistant (HASS), a free and open source home automation solution. You also flash the firmware of the bulb with Tasmota, a free and open source firmware for ESP8266-based Wi-Fi devices. And finally, you will have to configure an MQTT broker so that HASS and the bulb can talk to each other, and you can control the bulb using a shiny UI. The entire process takes about 20 minutes if you know what you are doing, and a couple hours if you're a beginner.

The flashing procedure is not officially supported, and will likely void the hardware warranty of your bulb. Note that, we have flashed 6 bulbs, and nothing has gone wrong, but your mileage may vary. You have been warned! Additionally, we won't be discussing best practices for setting up HASS. That's an entire article in its own right. You don't need HASS, you can control the bulb from Tasmota's webUI, or just

Wipro Garnet
If light bulbs are not your thing, check out these really cool dimmable LED tuble lights from Wipro. **https://dgit.in/may20-03**

Amazon Solimo
These Amazon Solimo light bulbs also use the ESP8366 board, and this DIY works on them too. **https://dgit.in/may20-04**

MQTT messages, but HASS provides a nice interface of control.

REQUIREMENTS

For flashing, you will need an up to date Debian-based Linux computer with Wi-Fi (For example, an old laptop, or a Raspberry Pi, or even a live disk. A VM will probably not work.)

- For the HASS server, an up to date Linux computer with Wi-Fi (An old laptop, Raspberry Pi, or even a VM. Note that a live environment is not optimal as you will have to install the server at every reboot.)
- A Wi-Fi device (A phone, a tablet, or another laptop connected to Wi-Fi)
- Docker / alternate method to install HASS (Optional, only if you want app UI and automations)
- Tuya-convert
- A compatible Wi-Fi bulb
- A few hours on a good weekend!
- Home Assistant: Home-Assistant is a free and open source home automation and control solution. There are many ways and compatible devices to install it. Personally, we ran it in a python virtual environment on our home server. But for this guide, we recommend the docker method, as this is just a single command and requires minimal configuration.

TUYA AND TUYA-CONVERT

Tuya is a China-based manufacturer of smart electronics. As long as you have a Tuya device, you can flash custom firmware for the device in an OTA style update. No soldering required! Some Tuya devices available on the market include the Wipro Garnet line of smart lights, and the Amazon Solimo brand of smart LEDs.

WARNING: Remember to check your exact model number against the device list on Tasmota wiki before purchase. From personal experience, we can only confirm that Wipro Garnet NS9001 works.

The first time a Tuya device connects to the internet, it calls home to fetch updates. We use a program called tuya-convert which pretends to be the update server for the bulb, and installs a firmware of our choice into the device. We need a Debian-based computer because the program runs apt-get in the background. This can be modified for use with other distributions of Linux, but this is not covered in this article. While this article is written with a Debian base in mind, it can easily be modified for other Linux distributions.

TASMOTA

ESP8266 is a Wi-Fi enabled microcontroller commonly used in IoT devices. Tasmota is a free and open source custom firmware for ESP8266 based devices. It works for a wide range of devices including bulbs, switches, and kettles. You can configure the firmware to work for your device by specifying a device template, so it knows what device it is installed on. With Tasmota installed, you get features such as full brightness and colour control, timed executions, etc, with complete control over your device. No proprietary spying blackboxes. Sadly, you lose the ability to voice control and app control the bulb with just Tasmota, as it provides only a web interface and an MQTT interface, but the functionality can be added back with Home-Assistant.

UPDATES

You should first update your computer

```
sudo apt-get update &&
sudo apt-get upgrade
```

If you are running a live environment, you may need to enable universe repository before running updates.

```
sudo add-apt-repository
universe
```

MQTT

Next, install an MQTT broker so that the smart bulb and HASS can talk to each other. (Do this on the 'server' PC.)

```
sudo apt-get install mos-
quitto mosquitto-clients
```

Now, we need to open the MQTT port on your computer firewall.

```
sudo ufw allow 1883; sudo
ufw enable
```

Tasmota + Home Assistant
Wi Fi Connection
App Control
Voice Control
Energy Saving
Dimmable
Timer Function

The App control and Voice control functionality is provided by Home Assistant, everything else by Tasmota.

HOME ASSISTANT

We need to remove any conflicting packages on your system (server PC).

```
sudo apt-get remove docker
docker-engine docker.io con-
tainerd runc
```

Then, we will need to install the packages required for docker

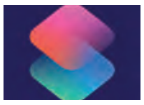
```
sudo apt-get install -y
software-properties-common
apparmor-utils apt-transport-
https avahi-daemon ca-certif-
icates curl dbus jq network-
manager socat ufw
```

Modem manager service needs to be disabled since it can cause conflicts

```
sudo systemctl disable
ModemManager.service
```

Install Docker Community Edition-

```
curl -fsSL get.docker.com | sh
```


iOS Shortcuts Guide
iOS has powerful home automation tools, and this in-depth guide should give you a headstart. **https://dgit.in/may20-05**

IFTTT applets
Here are a bunch of IFTTT applets to use with yoursmartlights, including GPS and time based ones. **https://dgit.in/may20-06**

Check if docker was installed properly by running the hello-world image.

```
sudo docker run hello-world
```

If this produces errors, you must debug them before you continue. Next, install home assistant in docker

```
curl -sL "https://raw.githubusercontent.com/home-assistant/hassio-installer/master/hassio_install.sh" | sudo bash -s
```

Now visit your home assistant installation by typing http://localhost:8123 into your browser. Follow the on screen instructions.

After this, you will need to enable the MQTT integration.

```
Head to settings -> integrations.
```

On the bottom right corner of the screen, you should see a button to add integrations. Click on that. On the resulting dialogue box, choose MQTT.

Enter the following settings -

```
Broker: localhost, Port: 1833, Enable Discovery: True
```

TUYA-CONVERT

In this step, you will flash Tasmota onto your smart bulb. This step is crucial in stopping your data from leaking all over the internet via the proprietary firmware the bulb already has.

This step needs to be followed on the "flashing" computer. The one with hardware access to its Wi-Fi chip.

First you need to clone the repository from github.

```
sudo apt-get insall -y git
git clone https://github.com/ct-Open-Source/Tuya-convert
```

Then you should run the "pre requisites" script. (Note, if you have not already, you should enable the universe repository and update your computer. Otherwise the next step produces errors.)

```
cd Tuya-convert; ./install_prereq.sh
```

Make sure the above step completes without errors. Next, within the Tuya-convert directory execute

```
cat config.txt
```

You should see a line similar to

```
WLAN=wlan0
```

Make sure it matches your Wi-Fi chip. To check the name of your Wi-Fi chip, execute

```
ifconfig | grep wl
```

The first word in the output is the name of your Wi-Fi chip.

```
wlp2s0: flags=4099 mtu 1500
```

Edit config.txt to reflect the right WLAN name. And then finally, you can start the flashing process!

```
./start_flash.sh
```

The following steps are accurate at the time of writing. It is possible that something changes in future updates to Tuya-convert. When in doubt, follow the on screen prompt. Follow the on screen guide, inputting consent as required, until you reach the point where your laptop is broadcasting vtrust-flash (At this point the process pauses for you to press enter). Connect a Wi-Fi device to vtrust-flash. All it needs to do now is connect to Wi-Fi and go to the network login page. Plug your smart bulb into a socket and switch it on. It should immediately start blinking fast. If, for some reason, it isn't blinking fast, you can power cycle it twice to set it to the required mode. To power cycle it, switch it ON-OFF-ON-OFF-ON. Wait a moment for the bulb to enter flashing mode. With your bulb in flashing mode, and a device connected to vtrust-flash , hit enter on the console.

The program will now connect to your smart bulb, make a backup of its original firmware, and then prompt you to choose between installing Tasmota and another firmware. Hit the number choice for Tasmota. The program will upload the Tasmota firmware to your bulb and reboot it. Close the program when it asks you to flash more devices. Now, wait for the bulb to restart, and then connect to its Wi-Fi. It should be named 'something Tasmota-1111'. It is possible to do this and the next configuration steps with your phone browser, but its recommended you do this from your computer.

TASMOTA CONFIGURATION

The first time you connect to your Tasmota device via its Wi-Fi AP, you will need to enter the Wi-Fi name and password of your router. It will then reboot and connect to your Wi-Fi. Once this is done, you will need your bulb's IP address to continue. This can be looked up from your router's status page, or using the "LAN neighbours" features of the "Wi-Fi Analyzer" app on Android. Enter your bulb's IP address into your browser. You will be greeted by Tasmota's webUI menu. Head to `Configuration -> Configure MQTT`, and enter your MQTT broker's IP. If you've been following this article, that's the same IP address as your "server" computer. Now, hit "Save". Your bulb should reboot again. At this point, your bulb can talk to your MQTT broker. Your MQTT broker is already talking to your install of HASS. Now, all that remains is to tell Tasmota what hardware its running on.

TASMOTA TEMPLATE CONFIGURATION

You now need to configure the template of your firmware to match your device. This is just mapping the software functionality to the hardware ports of your device. Your first try should be to simply copy the template of from your light's page at https://templates.blakadder.com/bulb.html.

*Note that in our experiments Wipro Garnet did *NOT* follow the template given on blakadder.*

If the template from the website doesn't give you the expected functionality , and your light is Wipro Garnet NS9001 810lm, you could try the following template :

```
{"NAME":"WiproSmartBulb","GPIO":[0,0,0,0,38,37,0,0,40,39,41,0,0],"FLAG":0,"BASE":18}
```

If the above steps fail for you, you will have to get your hands dirty and investigate. Its not hard, it just requires some extra time.

STEP 1: Go into console, and type the command Color 255,0,0,0,0

Alexa routines
An outline of various approaches that can be used to make Alexa work with your smart lights. **https://dgit.in/may20-07**

Best smartlights
Whether it is for family rooms, value or security, here are the best smartlights for every purpose. **https://dgit.in/may20-08**

This sets the "red color" (PWM1) in the software to full brightness, and all other colours in software to 0. Now, all we need to do is map PWM1 to the different GPIO ports in the template configuration one by one. The bulb will restart after each template change.

STEP 2: Go into template and map PWM1(37) to GPIO4. Make all the other maps go to 0. Click on save. Note what colour the bulb changes to.

STEP 3: Map PWM1(37) to GPIO5. Make all other maps go to 0. Click on save. Note what colour the bulb changes to.

STEP 4: Map PWM1(37) to GPIO12. Make all other maps go to 0. Click on save. Note what colour the bulb changes to.

STEP 5: Map PWM1(37) to GPIO13. Make all other maps go to 0. Click on save. Note what colour the bulb changes to.

STEP 6: Map PWM1(37) to GPIO14. Make all other maps go to 0. Click on save. Note what colour the bulb changes to.

At the end of Step 6 , you should have a map similar to this

GPIO4	Green
GPIO5	Red
GPIO12	Yellow
GPIO13	Blue
GPIO14	White

If you don't get all the colour channels you expect from the above GPIO ports, you can try setting random GPIO ports to PWM1 (37), till you find the date for all channels. RGB lights will only have 3 channels, RGBW will have 4 channels, and RGBWY will have 5 channels which need mapping.

Once you have all your data, all you need to do is go into the template configuration and map the right colour channels to the right GPIO pins. You will need to do this according to the following table-

Color	PWM Pin	Code Number
Red	PWM1	37
Green	PWM2	38
Blue	PWM3	39
Yellow	PWM4	40
White	PWM5	41

So, if your GPIO to Color maps were as in the example above, you would get a mapping as follows:

GPIO	PWM	Code Number
GPIO0	None	0
GPIO1	None	0
GPIO2	None	0
GPIO3	None	0
GPIO4	PWM2	38
GPIO5	PWM1	37
GPIO9	None	0
GPIO10	None	0
GPIO12	PWM4	40
GPIO13	PWM3	39
GPIO14	PWM5	41
GPIO15	None	0
GPIO16	None	0
ADC0	None	0

FADE AND SPEED

This is an optional step that allows you set fade and speed options on your bulb. This is optional, but it is highly recommended, as lights with colours that just snap into place can get a little jarring.

Open up the Tasmota console. Enter in the following two commands-

```
Fade ON
Speed 20
```

This will make the bulb fade between settings. You can set speed to any integer between 1 and 20, where 1 is the shortest time between settings, and 20 is the longest.

SLEEP

If you notice your light is flickering when turned on, you can try setting Sleep to a less aggressive setting. Your light will consume slightly more power when turned off, but the trade off in terms of quality of lighting is worth it.

```
Sleep 0
```

Once you have HASS properly set up, you can automate your light to set Sleep to 0 when its on for the right behaviour, and set Sleep to 200 for maximum power savings whenever it turns off.

```
SetOption19
```

Normally, you would have to manually configure your light and HASS to work together, but Tasmota provides a handy option to play nice with HASS when MQTT autodiscovery is enabled. Head to your Tasmota web UI, and click on console. Type in SetOption19 ON. Now, when you restart your setup, everything will just magically work together! If everything has gone according to keikaku, you will end up with a nice bulb interface in your HASS UI.

BONUS : AUTOMATE SLEEP

If you're experiencing weird flickering, you can automate changing the sleep setting based on the state of the light, to get maximum saving when the light is off, and maximum performance when the light is on. In your HASS interface, head to `Configuration -> Automations`. Click on the new button on the bottom right corner. On the resulting dialogue box, click "skip". Name your automation something informative like "PowerOnSleep0". As your trigger, select "State". In the entity, select your light. Enter From as "off", and To as "on". As your action, select "Call Service". Select your service to be mqtt.publish. In the payload, enter the following :

```
topic: Tasmota/cmnd/SLEEP
payload: 0
```

Hit save. Add another automation, with the opposite trigger, and payload: 200. Note : The topic will change based on your Tasmota MQTT settings. Tasmota/cmnd/{command name} is the default topic. d