RRoble Double Dimmer

Single and Double Dimmer with IoT over WiFi

Revision History		
Version date	**Author**	**Change Summary**
Tuesday, June 25, 2019	Jesus Amozurrutia	First Draft
Saturday, November 07, 2020	Jesus Amozurrutia	Version 1
Friday, April 30, 2021	Jesus Amozurrutia	Version 2
Monday, November 01, 2021	Jesus Amozurrutia	Version 3
Tuesday, January 31, 2023	Jesus Amozurrutia	Version 4

Contents

1. Introduction

This project is quite involved, so first, why go this route?

There are essentially two ways to control the lights at home, Smart Bulbs or with Smart Light Switches / dimmers, both with advantages and disadvantages.

Smart Light Switches are cheaper. Purely based on cost, smart light switches are likely to be far cheaper than smart bulbs, especially since one light switch can control multiple light fixtures, depending on how your house's wiring is set up.

You can use any bulb with Smart Light Switches. Since the traditional light bulb industry is very mature, the range of styles and accessories is much broader. In recent years, new smart lights solutions have appeared that allow for flexibility and different styles, but for the bulk of the lights it is a home or office, the Smart Switch is still the best option.

Smart Light Switches can control your existing fixtures without complex modifications.

The main disadvantage is that special wiring is required in the electric boxes where the switches are to be installed (the wires for hot, neutral, as well as the wire for the light are required, although there are already some options that do not require both cables, this project has not reached that point.). In most recent installations, this is not an issue, but it can be a problem for older installations; also, there are different standards in each country.

While smart bulbs offer more flexibility and have the possibility of adding color, they are more expensive and the need for this level of control, from my point of view, only makes sense in a few specific areas and in the end, you can combine the two solutions.

For these reasons, I leaned towards Smart Light Switches for the bulk of my lights.

Now since there are several options on the market, why go the DIY path?

Mainly I return to the point of the range of styles and accessories. In the arena of traditional light switches, there are thousands of colors, styles, textures and accessory options that are not yet available in smart switches.

On the other hand, inexpensive smart switches do not offer enough functionality, and some of the more expensive ones did not fit the form factor of my installation. I have also had the experience that my visitors who are not familiar with devices find some of these devices unintuitive.

By taking the DIY path, I can get the flexibility in style, form factor, and functionality at a reasonable cost.

With that out of the way, let's go to the project itself.

2. IoT Double Dimmer

This project consists of an AC light dimmer for one or two incandescent light bulbs or dimmable LED bulbs, with mains AC voltage of 110 to 220 V, 50 or 60Hz, up to 400W, using the WiFi enabled microcontroller ESP-12E / F.

The Dimmer functionality is achieved with the phase control method, using a TRIAC for each bulb, varying the total power applied. This method works very well with incandescent light bulbs and with LED lamps that have the Dimmable specification.

Features:

- Lights can be dimmed from 10% to 100% power.
- Supports transitions between brightness levels.
- One push button per light bulb to control the ON/OFF state and brightness.
    - Single click toggles ON/OFF state.
    - Continues press changes brightness.
- A LED indicator for each button/light bulb.
- LED indicators change intensity if light is ON or OFF.
- LED indicators intensity can be set remotely or turned off for day / night mode.
- Smart control and configuration using MQTT over WiFi; An MQTT broker is required.
- ON/OFF state and brightness can be set via the push buttons or remotely via WiFi/MQTT.
- Integrated into Home-Assistant or other MQTT compatible IoT Hubs, with auto discovery.
- Setup Portal over WiFi, to configure WiFi/MQTT parameters.
- The Setup Portal is started by a double reset or optionally with multi-click of any of the light buttons.
- On Power Up the device resets to the last known state, using MQTT retain or EEPROM.
- Optional double/triple clicks to trigger additional actions.
- Advanced configuration through MQTT.
    - Dimming time.
    - Default transition speed.
    - Minimum power can be set for each light.
    - Minimum Turn ON Brightness (some cheap LED bulbs require some a minimum brightness level to actually turn ON).
- Each light can be set to "Switch Mode" in case the Dimmer functionality is not required or supported.
- The Dimmer auto detects AC frequency.

The finished product.

2.1. Definitions

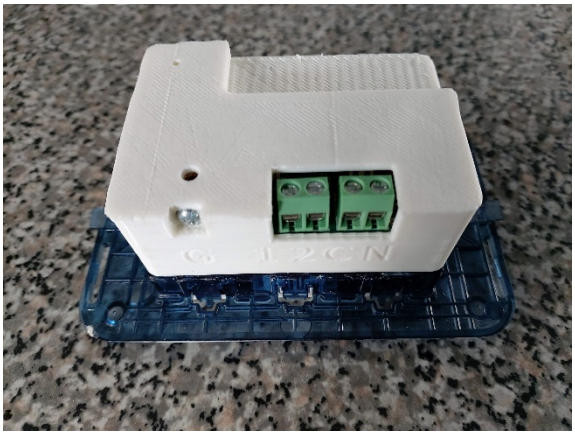
- Device: Refers to the component as a whole. It is used to identify the status of the device and to set the configuration.
- Instance: It refers to each of the lights. The status of each of the instances can be set individually.
- Brightness: Relative brightness for each of the light bulbs. Brightness is scaled from 0, for minimum brightness, to 100, for maximum brightness.
- Zero Crossing is a pulse that detects every time the AC sinusoidal electrical voltage signal crosses the zero volt zone. It is used to synchronize the phase control.
- Power: The electrical power applied to the light bulb to achieve the desired brightness. Power is regulated using phase control, by varying the time to trigger the TRIAC after the zero crossing detection. The applied power is linear and proportional to the brightness value.
- Minimum Power: The minimum power applied to the light bulb when the brightness has a value of zero. The default value is 4% and it can be configured to any value between 4% and 50%.
- Duty Cycle: The time that the TRIAC is ON in relation to the duration of the AC power signal period.
- OTA: Over The Air updates. Allow the firmware to be updated over WiFi.

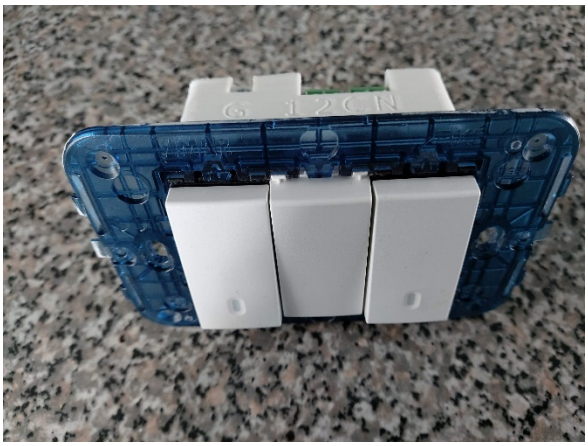
3. Hardware

The hardware is based on an ESP-12E/F micro controller, using TRIACs to control the total power applied to the light bulbs. The controller board is mounted over some Vimar® wall switches. I choose this particular brand because it features a wide range of styles and colors for the decorative plates and buttons. Finally, a 3D printed cover to protect the circuit.

The parts I am using are:

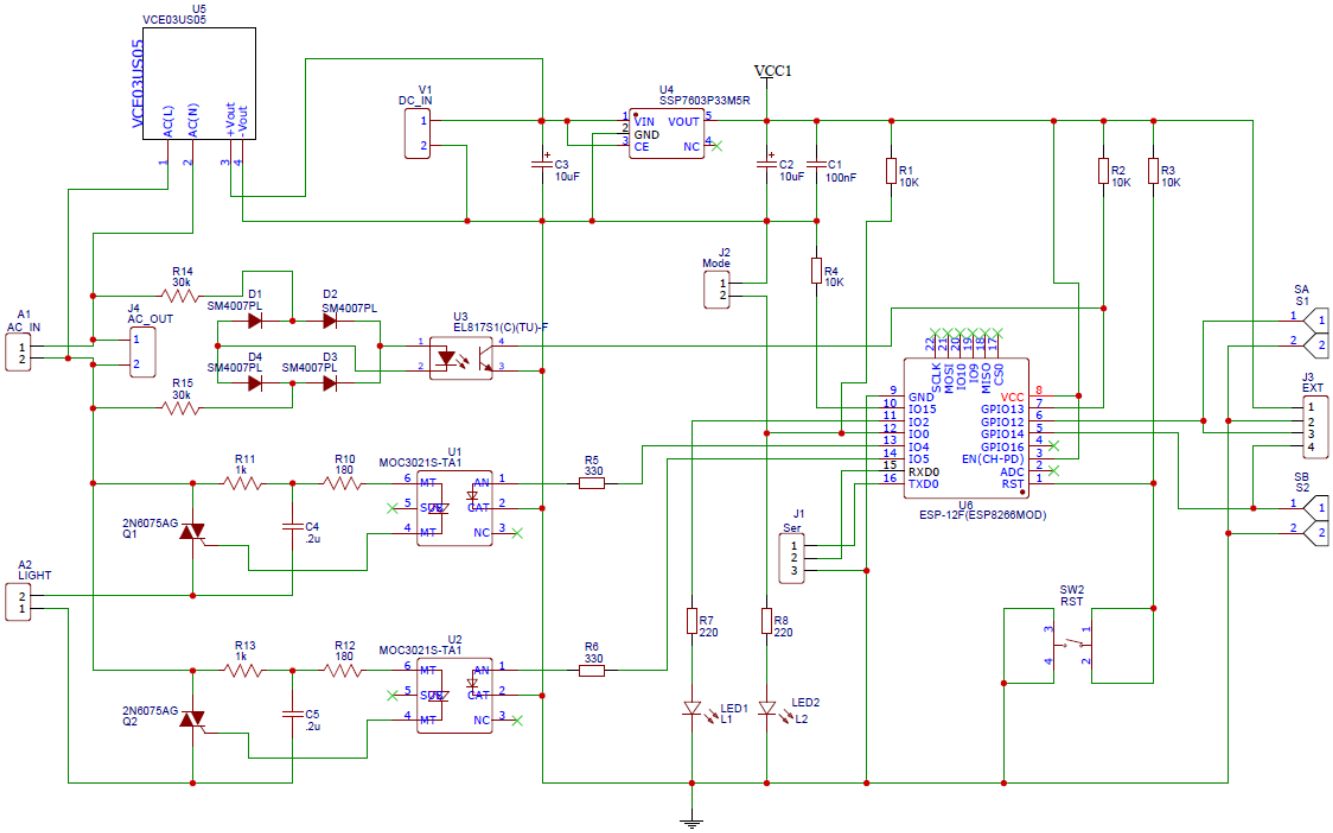
- https://www.vimar.com/en/int/catalog/product/index/code/19613 (Mounting frame)
- https://www.vimar.com/en/int/catalog/product/index/code/19008.B (Push Button)
- https://www.vimar.com/en/int/catalog/product/index/code/19041.B (Blank module)





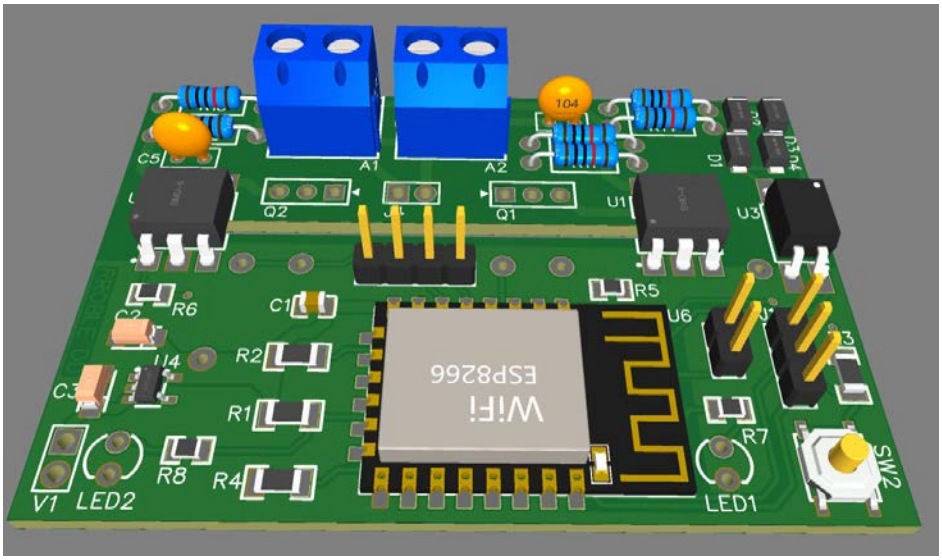
Full assembly.

3.1. Schematic diagram



3.2. PCB

The latest version of the PCB is based on surface mount components, to minimize board dimensions.



PCB with components (Missing TRIACs and LEDs).

The details of the PCB are available here:

https://oshwlab.com/jamozu/multi_project_copy_copy_copy_copy_copy

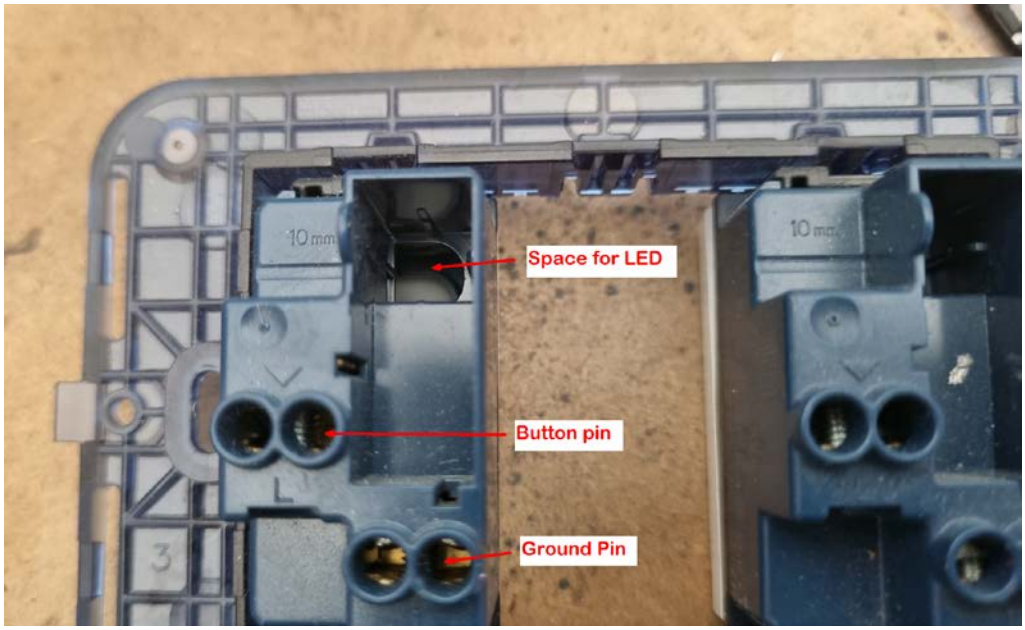
I have used JLC's services to manufacture the PCBS and they also offer assembly service for surface mount components.

3.3. Bill of material

ID	Designator	Quantity	Name	Description	LCSC Assembly
2	A1, A2	1	TERM	Screw terminal	
3	C1	1	100nF	Capacitor	Yes
4	C2,C3	2	10uF	Electrolytic capacitor	Yes
5	C4,C5	2	.2u	Capacitor	
6	D1,D2,D3,D4	4	SM4007PL	Diode	Yes
7	J1	1	3 Pin Header	2.5 mm Header	
8	J2	1	2 Pin Header	2.5 mm Header	
9	J3	1	4 Pin Header	2.5 mm Header	
10	J4	1	2 Pin Header	2.5 mm Header	
11	LED1	1	5mm LED	LED	
12	LED2	1	5mm LED	LED	
13	Q1,Q2	2	2N6075AG	TRIAC	
14	R1,R2,R3,R4	4	10K Ohm Resistor	SMT Resitor	Yes
15	R5,R6	2	330 Ohm Resistor	SMT Resitor	Yes
16	R7,R8	2	220 Ohm Resistor	SMT Resitor	Yes
17	R10,R12	2	180 Ohm Resistor	Resitor 1/2 W	
18	R11,R13	2	1K Ohm Resistor	Resitor 1/2 W	
19	R14,R15	2	30K Ohm Resistor	Resitor 1/2 W	
20	SA	1	19008.B	Vimar Push Button	
21	SB	1	19008.B	Vimar Push Button	
22	SW2	1	RST	Push Button	Yes
23	U1,U2	2	MOC3021S-TA1	TRIAC Opto-cupler	Yes
24	U3	1	EL817S1(C)(TU)-F	Opto transistor	Yes
25	U4	1	SSP7603P33M5R	3.3V Voltage regulator	Yes
26	U5	1	VCE03US05	5V AC-DC converter	
27	U6	1	ESP-12F(ESP8266MOD)	ESP Microcontroller	

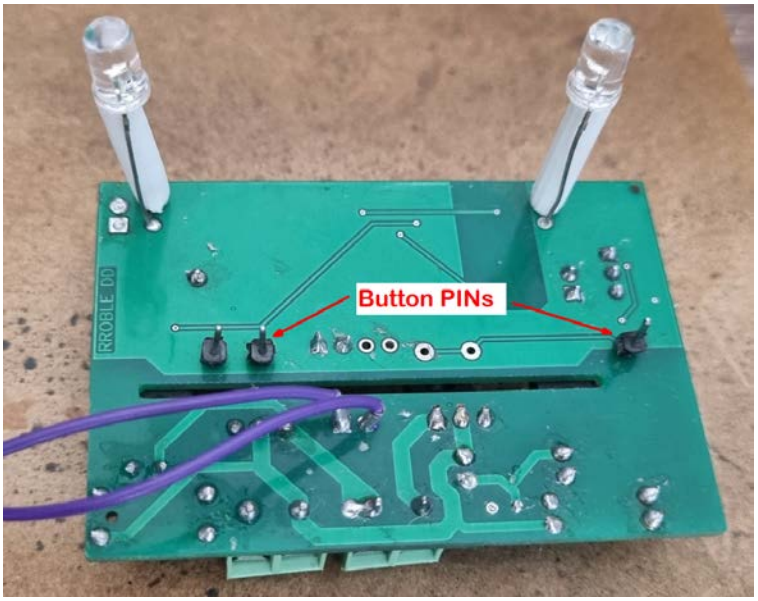
3.4. Assembly

The PCB is installed over the Vimar pushbuttons. These have a hole for the LED indicators and the cable clips are used to hold the button pins. The second cable clip for each pushbutton connects to the ground of the 5V power supply.

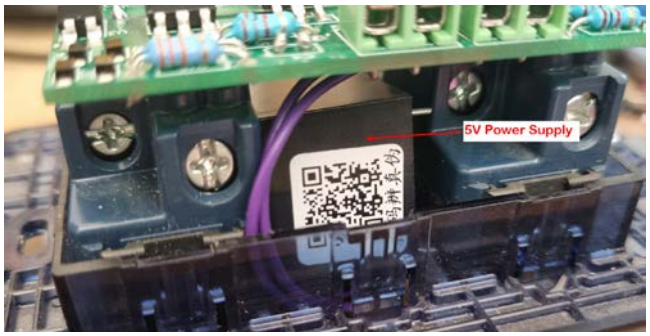


Push button assembly details.

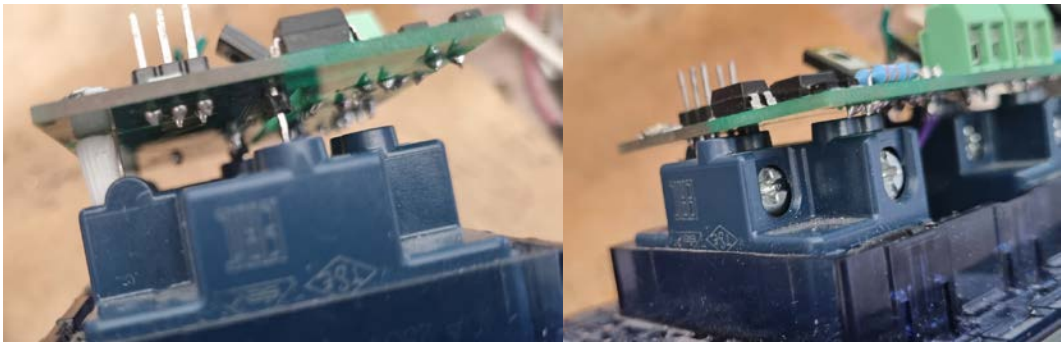
On the back, the PCB has the LEDs and button pins, as well as the AC cables for the 5V power supply.



The 5V power supply is installed in the space between the pushbuttons.



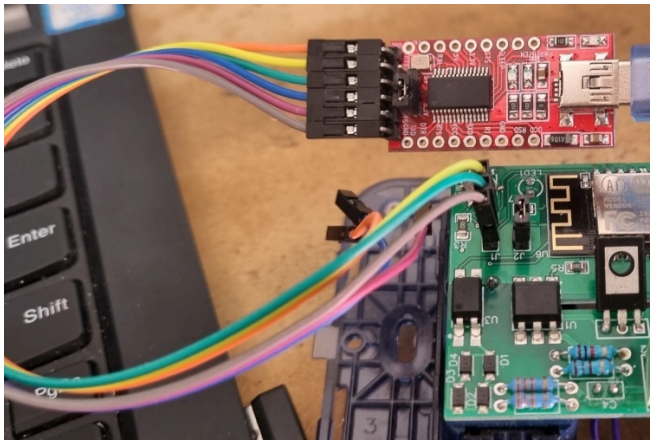
The LEDs and button pins are inserted into the corresponding slots on the push button and the push button screw is tightened once the board is flush with the push button. For best results you can use a little conductive paste in the cable clips.



4. Flashing Software

Once the hardware is assembled and before placing the cover, it is necessary to flash the software for the first time.

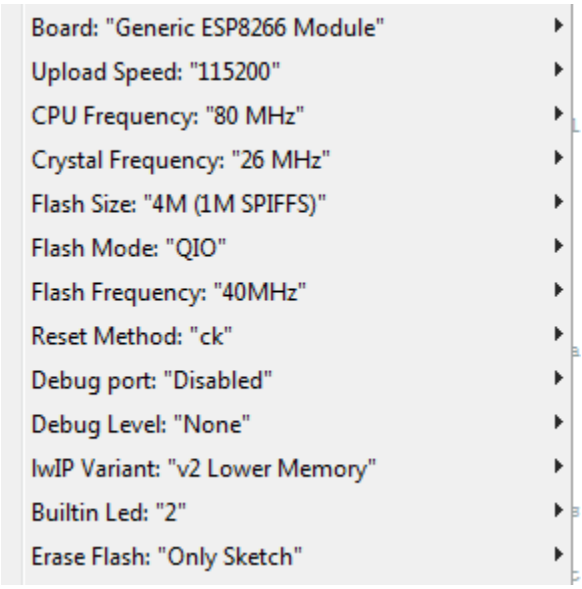
To connect the device with the computer an FTDI configured for 3.3V is required. Connect the PC via USB with a 3.3V FTDI adapter with jumper cables. Only 3 pins are used; TX0, RX0 and GND. Set a jumper in J2.



To power the device, you can connect 5V across V1 or 3.3V on pins 1 and 2 of J3. It is not recommended to flash by connecting to the mains power, due to the risk of electrocution or damage to the PC.

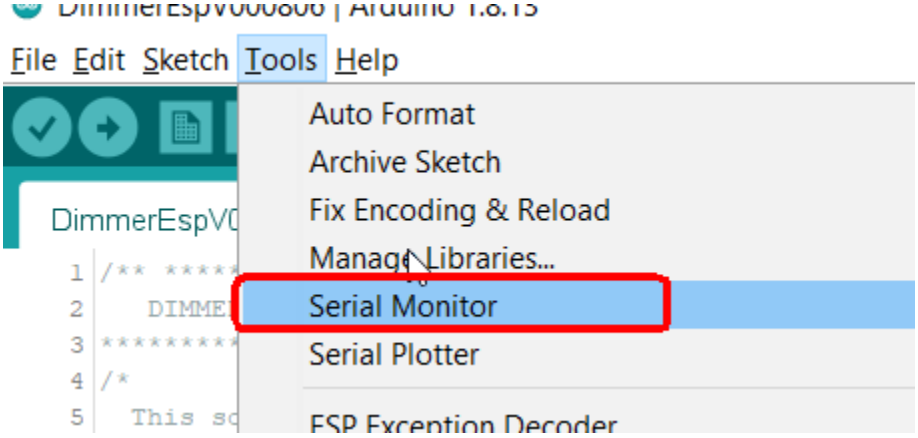
Open Arduino IDE and open the “DimmerEsp_Current.ino” sketch.

Under the Tools menu, set the Board to “Generic ESP8266 Module” and the set the following parameters for the board:



The file system (SPIFFS) is required to save configuration parameters.

Open the Serial Monitor to make sure that there is communication between the computer and the device and that it boots in the correct mode:



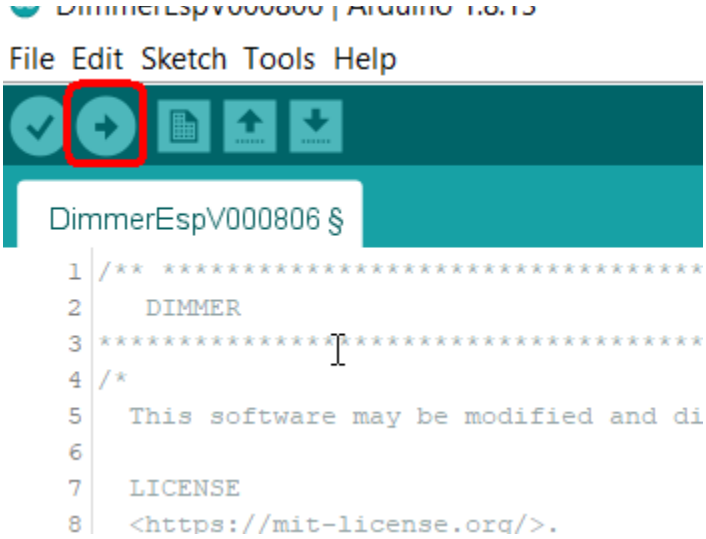
Configure the serial monitor to a baud rat of 74880.

Apply power to the device, then the ESP will boot on flash mode (1). You should see something like this in the serial monitor:

```
ets Jan  8 2021,rst cause 1, boot mode:(1,2)
```

At this point the J2 jumper can be removed.

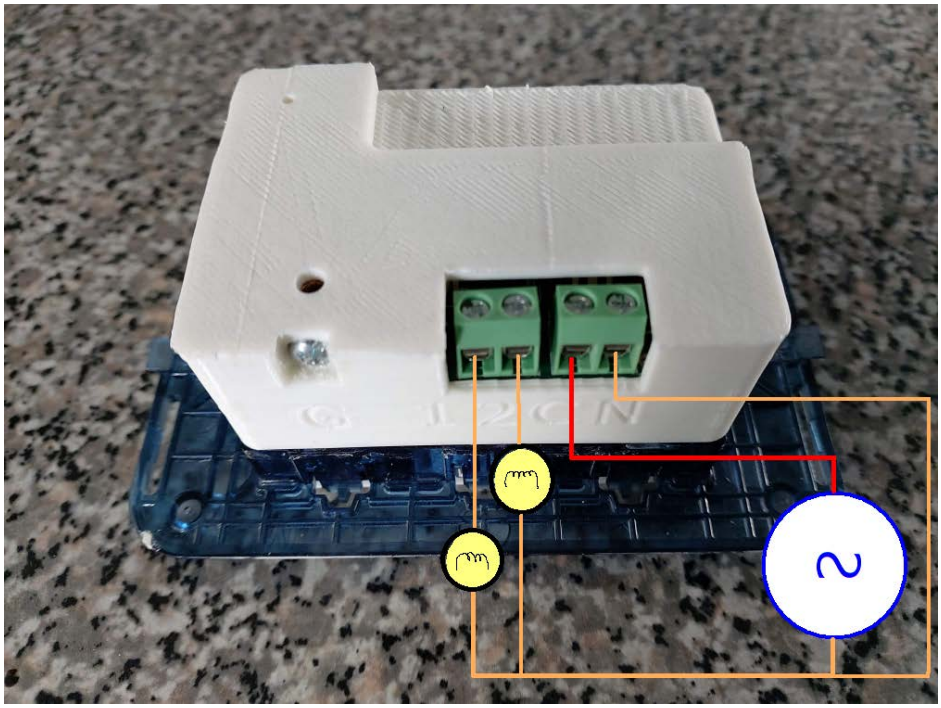
Click the upload button on the Arduino IDE to start flashing.



The device is ready to operate. Remove the jumper from J2 if it is still in place and put the cover; now it can be connected to the mains power and lights through the screw terminals.

5. Installation

Install the dimmer according to the diagram:



6. Setup portal

Once the device is connected, you can configure the parameters of the WiFi network with the Setup Portal.

To activate the Setup Portal:

- If it is not already screwed into the wall, a double reset can be done on the back of the device;
- Otherwise you can activate the portal by doing 6 to 8 clicks on any of the buttons within 4 seconds.

The device indicators start flashing every second. On a computer or mobile device, search for a WiFi network with a name like “RDimmer_XXXXXXXXXXX” and select it.

Open a browser. It should automatically direct you to the Setup Portal. In case the portal does not open, you can go to the address “http://192.168.4.1/“ to open it manually.

Follow the on-screen instructions to set the WiFi configuration and MQTT parameters.

For more information on the Setup Portal see: https://github.com/tzapu/WiFiManager

7. Advanced Configuration

Various device parameters can be configured remotely using MQTT commands.

7.1. Home Assistant

If the Broker that was configured with the Setup Portal is part of Home Assistant or other MQTT compatible hubs, the device should work out of the box. By default, auto-discovery is enabled, so Home Assistant should discover the dimmer as soon as the initial configuration is done.

7.2. Configuration

Various parameters can be configured through MQTT commands. Those commands can be sent using any MQTT program, for example **mosquitto** that comes as part of Home Assistant installation with MQTT. To run **mosquitto** you can use the SSH add-on.

For example for a device called “bedroom” in the namespace “MyHome” we can change the minimum power applied to the light bulbs (We have found this may be necessary with some LED lamps that do not turn on at low power levels):

```
mosquitto_pub -h my.broker.net -p 1883 -t "MyHome/light/bedroom/set" -m '{"action":"config",
"secret":"secret", "minPow1":10, "minPow2":10}' -u USER -P PASS
```

- Set the second light to Switch mode:

```
mosquitto_pub -h my.broker.net -p 1883 -t "MyHome/light/bedroom/set" -m '{"action":"config",
"secret":"secret", "LgtDimm2":false}' -u USER -P PASS
```

- Set the name for both lights:

```
mosquitto_pub -h my.broker.net -p 1883 -t "MyHome/light/bedroom/set" -m '{"action":"config",
"secret":"secret", "name1":"bedroom_left", "name2":"bedroom_right"}' -u USER -P PASS
```

The parameters that can be configured using MQTT are:

Parameter	Type	Description	Values	Default
myId	string [20]	Device ID.	[a-z0-9_.]	MAC Address
retain	bool	MQTT Retain flag. If this flag is set, the last known state for the light is preserved by the MQTT broker.	TRUE = On; FALSE = Off	TRUE
keepAlive	int	MQTT Keep alive in seconds.	60 – 10000	Default [59]
dimmTime	int	Time to go from 0% brightness to 100% brightness or vice versa (milliseconds)	500 - 20000	2500
edgeTime	int	Delay at the edge of the dimmable range in milliseconds (makes it easy to select the Minimum or Maximum brightness).	100 – 3000	800
LedInstOn	int	LED indicator intensity when the lights are ON.	0 – 100	100
LedInstOff	int	LED indicator intensity when the lights are OFF.	0 - 100	25
LedDefault	bool	Default state for LED Indicators.	TRUE = On; FALSE = Off	TRUE
transitionOn	int	Transitions speed to turn ON with the button, in seconds.	0 – 12	0
transitionOff	int	Transitions speed to turn OFF with the button, in seconds.	0 – 12	0
name1	string [40]	Name of first light. It must be unique inside the MQTT namespace. If the name is not defined, it uses an auto generated INSTANCE_ID based on the MAC address.	[a-z_0-9]	“”
name2	string [40]	Name of second light. It must be unique inside the MQTT namespace. If the name is not defined, it uses an auto generated INSTANCE_ID based on the MAC address.	[a-z_0-9]	“”
LgtDimm1	bool	Light 1 mode.	TRUE = Dimmer; FALSE = Switch	TRUE
LgtDimm2	bool	Light 2 mode.	TRUE = Dimmer; FALSE = Switch	TRUE
LgtTrns1	bool	Light 1 transition. Enables the fade effect.	TRUE = Enabled; FALSE = Disabled	TRUE

LgtTrns2	bool	Light 2 transition. Enables the fade effect.	TRUE = Enabled; FALSE = Disabled	TRUE
brightness1	int	Default brightness for Light 1.	1 – 100	100
brightness2	int	Default brightness for Light 2.	1 – 100	100
minPow1	int	Minimum Power % for Light 1.	4 – 50	10
minPow2	int	Minimum Power % for Light 2.	4 – 50	10
minTUB1	int	Minimum Turn ON brightness for Light 1. It is used for some LED bulbs that do not turn on at low power levels.	0 – 50	0
minTUB2	int	Minimum Turn ON brightness for Light 2.	0 – 50	0

8. Software

The hardware is compatible with libraries such as ESPHome, using the AC Dimmer component, but this is out of the scope of this document.

For this project, I decided to create my own software to have more granular control of the features, which is what is described below.

If you have already reached this point, your can have a working dimmer. This section is part of the software's internal documentation. If you want to know more details and how you can improve the project, continue with this section.

The software is developed in C++ with the ***Arduino®*** platform .

8.1. Dimmer

The dimmer function as follows:

- A single button controls the State and brightness of each light bulb.
  - Short press = Toggle State ON/OFF.
  - Long press = Change brightness from 100% to 0% and back, while the button remains pressed.
  - Button release after a Long Press = Sets the brightness and calculates the powe/duty-cycle.
- Double click and triple click can be enabled to trigger an MQTT action that can be used for any additional function using the MQTT hub.
- If the light State is OFF, the signal to trigger the TRIACs stays down.
- If the light State is ON, the ESP waits for the Zero Crossing Pulse (ZCP) signal and triggers the TRIAC according to the current duty cycle.
- The minimum power level that can be applied to any Light is 4%. The minimum power can be adjusted for each Light through MQTT configuration.
- Brightness value can be set between 0 and 100: 100 = Maximum brightness; 0 = Minimum brightness, where 0 corresponds to the minimum power and 100 to 100% power.
- The duty-cycle is calculated so that the total power has a linear relation with the brightness level.

- The dimmer can be configured as a switch. In this case the TRIAC only turns ON or OFF.
- The sketch can be compiled in Switch Mode. In this mode, the dimmer functions are disabled and the operation of the LED indicators changes to reflect which instance is ON. See SWITCH_MODE and HIGH_IMPEDANCE on the sketch.

The zero-crossing pulse (**ZCP**) is connected to a pin on the ESP that handles hardware interrupts.

On power up, the device detects the AC power frequency and the duration (width) of the **ZCP**. Once the AC parameters are obtained, the device enters normal operation mode.

In normal operation mode, the ESP gets a hardware interruption in the rise part of the **ZCP**. At this point, if the light is ON, it sets the timer interruptions used to trigger the TRIAC(s) based on the current duty-cycle.

If one or both lights are turned on, up to 4 timers can be defined:

1) **Trigger first set of TRIACs**. This can trigger Light 1 and/or Light 2. If both lights have the same duty-cycle they will trigger at the same time.

2) **Trigger second set of TRIACs**. (conditional) This can trigger Light 1 and/or 2. If the lights have a different duty-cycle values the light with the lower duty-cycle is triggered with the second timer.

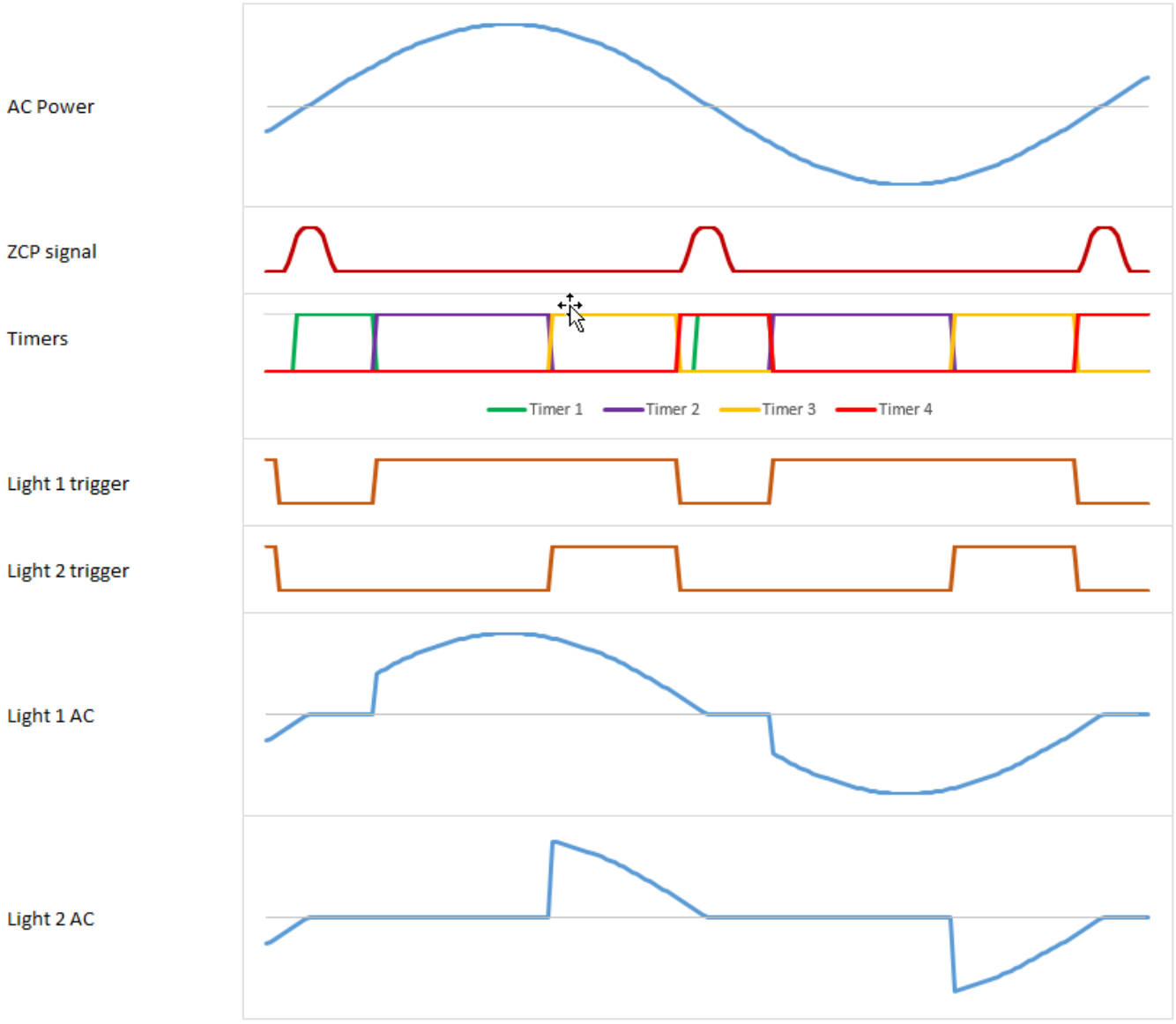
3) **Turn Off trigger signal**. At 95% of the time before the next **ZCP**, the trigger signal for the TRIACs is turned off.

4) **Repeat cycle at step 1**. The last timer is the time between the **Turn Off trigger signal** and the trigger of the next cycle. This timer is overridden by the next **ZCP** interrupt, but in case the pulse is not detected, the cycle is repeated anyway; this prevents light flickering in case of AC noise.

The timers use ESP's timer_1 interrupt and are triggered in sequence using a single hardware timer.

The following graph shows the operation of a double dimer, with a light on at 95% power and the second light on at 27% power.

You can see the **ZCP** signal, the calculated timers, the signals to trigger the TRIACs and the power applied to the light bulbs.



The program loop takes care of the rest of the tasks like:

- Detects the button status and sets the ON/OFF/Duty-cycle of each light.
- Sets the indicator LEDs brightness.
- Monitor WiFi connectivity.
- Schedule MQTT out messages.

The incoming MQTT messages are handled by a callback function.

8.2. WiFi

The ESP microcontroller communicates with an MQTT broker, using WiFi in normal operation mode; in this case the device is a client in an existing WiFi network.

The device can be set for the initial configuration mode; in this case, the device uses WiFi in access point mode and an internal web server catches the requests for initial configuration.

8.2.1. MQTT Topics

The Dimmer uses 2 or 3 **MQTT** topics for publishing:

1) Device state:

`NAMESPACE/light/DEVICE_ID/state`

2 & 3) Status of each Light:

`NAMESPACE/light/INSTANCE_ID/state`

It also subscribes to 2 or 3 **MQTT** topics:

1) Configure the device:

`NAMESPACE/light/DEVICE_ID/set`

2 & 3) Set the light status or request feedback:

`NAMESPACE/light/INSTANCE_ID/set`

The **`NAMESPACE`** can be configured on the WiFi setup portal or trough MQTT

The **`DEVICE_ID`** is set using the MAC Address in the form: *XXXXXXXXXXXX*.

`INSTANCE_ID`s are defined as:

- **N***XXXXXXXXXXXX***1**
- **N***XXXXXXXXXXXX***2**

The Instance ID can be overridden using the configuration parameters "***name1***" and "***name2***".

8.2.2. MQTT Messages

The Dimmer publishes the following messages:

- **`NAMESPACE/light/DEVICE_ID/state`**
  - ***{"state":"golive"}*** → This message is published after the device is turned on, or after a reset. This is can be used to obtain the last known state in the IoT Hub after a reset or power outage
  - ***{" state ":"online"}*** → Published periodically every KEEPALIVE seconds.
  - ***{" state ":"reset"}*** → Published before the device resets.
  - ***{"feedback":"configuration saved"}*** → Published after a configuration change is saved.
- **`NAMESPACE/light/INSTANCE_ID/state`**
  - ***{"state":["ON"|"OFF"], "brightness":[0-100], "power":[0-100], "indicator":["ON"|"OFF"]}***

The dimmer is constantly listening on the feedback topics and it accepts the following messages:

- **`NAMESPACE/light/DEVICE_ID/set`**

- *{"action":"config",*
  "secret":CONFIG_SECRET,
  "mqttbroker":MQTT_BROKER_NAME-IP,
  "mqttport":MQTT_PORT,
  "mqttuser":MQTT_USER,
  "mqttpass":MQTT_PASSWD,
  "namespc":MQTT_NAME_SPACE,
  "mqttdisc":MQTT_AUTO_DISCOVERY,
  "mqttkey":MQTT_SHARED_KEY,
  "keepAlive":KEEP_ALIVE_SECONDS}
- *{"action":"config",*
  "secret":CONFIG_SECRET,
  "LgtMode":BOOL,
  "LgtDimm1":BOOL,
  "LgtTrns1":BOOL,
  "LgtDimm2":BOOL,
  "LgtTrns2":BOOL,
  "LedBright":INT,
  "LedDimm":INT,
  "dimmTime":INT,
  "edgeSpeed":INT,
  "transitionOn":INT,
  "transitionOff":INT,
  "name1":LIGHT1_NAME,
  "name2":LIGHT2_NAME,
  "minPow1":LIGHT1_NAME,
  "minPow 2":LIGHT2_NAME,
  "brightness1":LIGHT1_NAME,
  "brightness2":LIGHT2_NAME}
- *{"action":"getStatus","instance":NUMBER}*
- *{"action":"indicators","bright":NUMBER,"dimm":NUMBER}*
- *{"action":"reset"}*

- **`NAMESPACE/light/INSTANCE_ID/set`**
  - *{"state":["ON"|"OFF"],"brightness":[0-100],"transition":[0-10],"indicator":["ON"|"OFF"]}*
  - *{"action":"getState"}*
  - *{"action":"getConfig"}*

8.2.3. Configuration

The device can be configured trough the WiFi Portal and through MQTT messages.

WiFi Portal

The portal is used for the initial device configuration. It allows to configure the **SSID** and Key of the WiFi network, as well as the MQTT broker and optionally the IDs for each light.

To enter the configuration mode, you need physical access to the reset button on the back of the Dimmer or do a multiple click (6 clicks within 4 seconds).

A Wi-Fi-enabled ***device***, such as a phone, tablet or computer, is also required.

To enter the configuration, perform a double reset; press the reset button and wait a second and press a second time, or do a multi-click. The LED Indicators start flashing once per second.

Open the WiFi configuration on the ***device*** and look for an **SSID** called something like "RDimmer_XXXXXXXXXXX", connect to the network and then point the browser to “http://192.168.4.1/”

Follow the on screen instructions.

MQTT Testing

- Turn On the Living Room light at 50% brightness:

```
mosquitto_pub -h my.broker.net -p 1883 -t " MyHome/light/living_room/set" -m '{"state":"ON",
"brightness":50}' -u USER -P PASS
```

8.3. Home Assistant

By default, auto-discovery is enabled, so Home Assistant should discover the dimmer as soon as the initial configuration is done.

It is possible to configure the light manually like:.

8.3.1. configuration.yaml

The parameters for the configuration YAML file are:

```
    light:
      - platform: mqtt
        schema: json
        name: INSTANCE_ID
        title: Friendly_Name
        state_topic: "NAMESPACE/light/INSTANCE_ID/state"
        command_topic: "NAMESPACE/light/INSTANCE_ID/set"
        brightness: true
        brightness_scale: 100
        optimistic: false
        qos: 0
```

8.3.2. dashboard.dash

If the installation has the HADashboard, the configuration for each light in the dashboard configuration file is:

```
    lightN:
        widget_type: light
        title: INSTANCE_NAME
        entity: light.INSTANCE_ID
```

8.4. Persistent status

In the event of a blackout or if the power supply is cut, or after a reset, the dimmer can return to its last known state using one of 2 mechanisms:

- EEPROM: The Dimmer saves each state change in EEPROM and retrieves it on boot. See *PERSISTENT_STATE* parameter in the sketch configuration.
- Retain Flag: By using the "retain" flag on MQTT messages, the dimmer saves its last state in the broker. The MQTT broker must be configured to accept retained messages.

The device can follow one of the following go live sequences, depending on the retain configuration:

- EEPROM: If "*PERSISTENT_STATE*" is defined and the "retain" parameter is turned OFF (set to FALSE), on power up or after a reset the device enters the "**Go Live**" sequence:
    a. The device reads the last known state from EEPROM.
    b. Start dimming.
    c. Connects to WiFi.
    d. Connects to the MQTT Broker.
    e. Publishes a "golive" message on the device's "Sate Topic" (NAMESPACE/INSTANCE_TYPE/DEVICE_ID/state).
    This can be optionally used by the Broker or Hub to override the initial state.
    f. Subscribes to the "Instance Set" topics (NAMESPACE/INSTANCE_TYPE/INSTANCE_ID/set) and listens for commands.
    g. Continues normal operation.
    Any state change is published to the "Instance State" topics with the retain flag turned OFF.
- Retain: If the "retain" parameter is turned ON (set to TRUE), on power up or after a reset the device enters a different "**Go Live**" sequence:
    - Start dimming.
    - Connects to WiFi.
    - Connects to the MQTT Broker.
    - Subscribes to the "Instance State" topics (NAMESPACE/INSTANCE_TYPE/INSTANCE_ID/state).
    - The broker sends the last known state back to the device, per MQTT standards with retain.
    - After getting the last known state from the broker for all instances the dimmer, or if the initial state is not received after 60 seconds (see GO_LIVE_DURATION), the device un-subscribes from the "Instance State" topics.
    - Subscribes to the "Instance Set" topics (NAMESPACE/INSTANCE_TYPE/INSTANCE_ID/set)
    - Continues normal operation.
    Any state change is published to the "Instance State" topics with the retain flag turned ON.

If WiFi or the MQTT Broker are not available during power up, the device will resume normal operation after 60 seconds. Also, if the user sets the status manually before the device can reach the MQTT broker, the "**Go Live**" sequence is canceled.

With this method the dimmer can establish its initial state with an MQTT broker using the retain standard and without additional functionality required by the Hub.

If Home Assistant (HA) is used, this method is preferred as it reduces the risk of HA and the dimmer being out of sync. In the event of temporary network failures or rebooting of any of the involved devices.

Any user activity, like pressing a button cancels the "**Go Live**" sequences.

8.5. OTA Update

This sketch supports OTA updates for development purposes, using the Arduino IDE or an HTTP server.

Currently, HTTP updates only support unsecured http, therefore it is only recommended in a local environment for development purposes.

It is possible to implement secure https, as in the next example, if a public deployment is required: https://gist.github.com/igrr/24dd2138e9c8a7daa1b4, but at the moment it is out of scope.

> *Note: Temporarily disable the firewall in the computer while uploading a new image to the controller, to avoid disconnection problems when using OTA with the Arduino IDE.*

8.6. Libraries

The software uses the following libraries:

- ESP8266 core for Arduino (https://github.com/esp8266/Arduino).
- ESP8266 WiFi Connection manager (https://github.com/tzapu/WiFiManager).
- Double Reset Detect (https://github.com/jenscski/DoubleResetDetect).
- Arduino Json (https://github.com/bblanchon/ArduinoJson/)
- Asynchronous MQTT Client (https://github.com/marvinroger/async-mqtt-client/releases/tag/v0.8.1).