MQTT / HTTP API

Table of Contents

Overview

This API documentation covers various functionalities such as retrieving device statistics, screen mirroring, notifications, customapps, sound playing, and mood lighting. You can interact with these features via both MQTT and HTTP protocols.

Status Retrieval

Access various device statistics like battery, RAM, and more:

MQTT Topic	HTTP URL	Description
`[PREFIX]/stats`	`http://[IP]/api/stats`	General device stats (e.g., battery, RAM)
`[PREFIX]/stats/effects`	`http://[IP]/api/effects`	List of all effects
`[PREFIX]/stats/transitions`	`http://[IP]/api/transitions`	List of all transition effects
`[PREFIX]/stats/loop`	`http://[IP]/api/loop`	List of all apps in the loop

> **Note: MQTT also broadcasts other data, such as button presses and the current app.**

LiveView

Retrieve the current matrix screen as an array of 24bit colors:

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/sendscreen`	`http://[IP]/api/screen`	-	GET

When triggering the MQTT API, AWTRIX sends the array to `[PREFIX]/screen`.

Extras:

- Access a live view of the screen in your browser: `http://[IP]/screen`.
- Options to download a screenshot or generate a GIF from the current display content.
- `http://[IP]/fullscreen` gives you a fullscreen liveview. Here you can optionally set the `fps` as parameter (standard 30)

Power Control

Toggle the matrix on or off:

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/power`	`http://[IP]/api/power`	`{"power": true}` or `{"power": false}`	POST

Send the board in deep sleep mode (turns off the matrix as well), good for saving battery life:

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/sleep`	`http://[IP]/api/sleep`	`{"sleep": X}` where X is number of seconds	POST
AWTRIX will only wakeup after time or if you press the middle button once. There is no way to wake up via API.			

Sound Playback

Play a RTTTL sound from the MELODIES folder:

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/sound`	`http://[IP]/api/sound`	`{"sound":"alarm"}`	POST

Play a RTTTL sound from a given RTTTL string:

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/rtttl`	`http://[IP]/api/rtttl`	`rttl string`	POST

Mood Lighting

Set the entire matrix to a custom color or temperature:

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/moodlight`	`http://[IP]/api/moodlight`	See below	POST

> ⚠️ **Caution: Using this function results in a higher current draw and heat, especially when all pixels are lit. Ensure you manage brightness values responsibly.**

To disable moodlight, send an empty payload.

Example:

```json
{
  "brightness": 170,
  "kelvin": 2300
}
```

Possible moodlight options:

```json
{"brightness":170,"kelvin":2300}
or
{"brightness":170,"color":[155,38,182]}
or
{"brightness":170,"color":"#FF00FF"}
```

Colored Indicators

Colored indicators serve as small notification signs displayed on specific areas of the screen:

- Upper right corner: Indicator 1
- Right side: Indicator 2
- Lower right corner: Indicator 3

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/indicator1`	`http://[IP]/api/indicator1`	`{"color":[255,0,0]}`	POST
`[PREFIX]/indicator2`	`http://[IP]/api/indicator2`	`{"color":[0,255,0]}`	POST
`[PREFIX]/indicator3`	`http://[IP]/api/indicator3`	`{"color":[0,255,0]}`	POST

Color Options:

- Use an RGB array, e.g., `{"color":[255,0,0]}`
- Use HEX color strings, e.g., `{"color":"#32a852"}`

Hide Indicators:

- To hide the indicators, send the color black (`{"color":[0,0,0]}`) or use the shorthand `{"color":"0"}` . Alternatively, send an empty payload.

Additional Effects:

- **Blinking**: To make the indicator blink, add the key `"blink"` with a value specifying the blinking interval in milliseconds.
- **Fading**: To make the indicator fade on and off, add the key `"fade"` with a value specifying the fade interval in milliseconds.

Custom Apps and Notifications

With AWTRIX 3, you can design custom apps or notifications to showcase your unique text and icons.

Interaction

- **MQTT**: Send a JSON object to `[PREFIX]/custom/[app]` , where `[app]` denotes your app's name (excluding spaces).
- **HTTP API**: Incorporate the app name in the query parameter (`name=[appname]`).
- **Updating**: To refresh a custom page, dispatch a modified JSON object to the identical endpoint. The display updates instantly.
- **One-Time Notification**: Use the same JSON format. Direct your JSON object to `[PREFIX]/notify` or `http://[IP]/api/notify` .

MQTT Topic	HTTP URL	Payload/Body	Query Parameters	HTTP Method
`[PREFIX]/custom/[appname]`	`http://[IP]/api/custom`	JSON	name=[appname]	POST
`[PREFIX]/notify`	`http://[IP]/api/notify`	JSON	-	POST

JSON Properties

Below are the properties you can utilize in the JSON object. **All keys are optional**; only include the properties you require.

Key	Type	Description	Default	Custom App	Notification
`text`	string	The text to display. Keep in mind the font does not have a fixed size and `I` uses less space than `W`. This facts affects when text will start scrolling	N/A	X	X
`textCase`	integer	Changes the Uppercase setting. 0=global setting, 1=forces uppercase; 2=shows as it sent.	0	X	X
`topText`	boolean	Draw the text on top.	false	X	X
`textOffset`	integer	Sets an offset for the x position of a starting text.	0	X	X
`center`	boolean	Centers a short, non-scrollable text.	true	X	X
`color`	string or array of integers	The text, bar or line color.	N/A	X	X
`gradient`	Array of string or integers	Colorizes the text in a gradient of two given colors	N/A	X	X
`blinkText`	Integer	Blinks the text in an given interval in ms, not compatible with gradient or rainbow	N/A	X	X
`fadeText`	Integer	Fades the text on and off in an given interval, not compatible with gradient or rainbow	N/A	X	X
`background`	string or array of integers	Sets a background color.	N/A	X	X
`rainbow`	boolean	Fades each letter in the text differently through the entire RGB spectrum.	false	X	X
`icon`	string	The icon ID or filename (without extension) to display on the app. You can also send a **8x8 jpg** as Base64 String	N/A	X	X
`pushIcon`	integer	0 = Icon doesn't move. 1 = Icon moves with text and will not appear again. 2 = Icon moves with text but appears again when the text starts to scroll again.	0	X	X
`repeat`	integer	Sets how many times the text should be scrolled through the matrix before the app ends.	-1	X	X
`duration`	integer	Sets how long the app or notification should be displayed.	5	X	X
`hold`	boolean	Set it to true, to hold your **notification** on top until you press the middle button or dismiss it via	false		X

Key	Type	Description	Default	Custom App	Notification
		HomeAssistant. This key only belongs to notification.			
`sound`	string	The filename of your RTTTL ringtone file placed in the MELODIES folder (without extension).	N/A		X
`rtttl`	string	Allows to send the RTTTL sound string with the json.	N/A		X
`loopSound`	boolean	Loops the sound or rtttl as long as the notification is running.	false		X
`bar`	array of integers	Draws a bargraph. Without icon maximum 16 values, with icon 11 values.	N/A	X	X
`line`	array of integers	Draws a linechart. Without icon maximum 16 values, with icon 11 values.	N/A	X	X
`autoscale`	boolean	Enables or disables autoscaling for bar and linechart.	true	X	X
`progress`	integer	Shows a progress bar. Value can be 0-100.	-1	X	X
`progressC`	string or array of integers	The color of the progress bar.	-1	X	X
`progressBC`	string or array of integers	The color of the progress bar background.	-1	X	X
`pos`	integer	Defines the position of your custom page in the loop, starting at 0 for the first position. This will only apply with your first push. This function is experimental.	N/A	X	
`draw`	array of objects	Array of drawing instructions. Each object represents a drawing command. See the drawing instructions below.		X	X
`lifetime`	integer	Removes the custom app when there is no update after the given time in seconds.	0	X	
`lifetimeMode`	integer	0 = deletes the app, 1 = marks it as staled with a red rectangle around the app	0	X	
`stack`	boolean	Defines if the **notification** will be stacked. `false` will immediately replace the current notification.	true		X
`wakeup`	boolean	If the Matrix is off, the notification will wake it up for the time of the notification.	false		X

Key	Type	Description	Default	Custom App	Notification
`noScroll`	boolean	Disables the text scrolling.	false	X	X
`clients`	array of strings	Allows forwarding a notification to other awtrix devices. Use the MQTT prefix for MQTT and IP addresses for HTTP.			X
`scrollSpeed`	integer	Modifies the scroll speed. Enter a percentage value of the original scroll speed.	100	X	X
`effect`	string	Shows an **effect** as background.The effect can be removed by sending an empty string for effect		X	X
`effectSettings`	json map	Changes color and speed of the **effect**.		X	X
`save`	boolean	Saves your custom app into flash and reloads it after boot. Avoid this for custom apps with high update frequencies because the ESP's flash memory has limited write cycles.		X	
`overlay`	string	Sets an effect overlay (cannot be used with global overlays).		X	X

Color: Accepts a hex string or an R,G,B array: `"#FFFFFF"` or `[255,255,0]` .

Overlay effects:

- "clear"
- "snow"
- "rain"
- "drizzle"
- "storm"
- "thunder"
- "frost"

Example

Here's a sample JSON to present the text "Hello, AWTRIX 3!" in rainbow colors for a duration of 10 seconds:

```json
{
  "text": "Hello, AWTRIX 3!",
  "rainbow": true,
```

```
  "duration": 10
}
```

Drawing Instructions

> ! Please note: Depending on the number of objects, the RAM usage can be very high. This could cause freezes or reboots. It's important to be mindful of the number of objects and the complexity of the drawing instructions to avoid performance issues.

Each drawing instruction is an object with a required command key and an array of values depending on the command:

Command	Array Values	Description
`dp`	`[x, y, cl]`	Draw a pixel at position (`x` , `y`) with color `cl`
`dl`	`[x0, y0, x1, y1, cl]`	Draw a line from (`x0` , `y0`) to (`x1` , `y1`) with color `cl`
`dr`	`[x, y, w, h, cl]`	Draw a rectangle with top-left corner at (`x` , `y`), width `w` , height `h` , and color `cl`
`df`	`[x, y, w, h, cl]`	Draw a filled rectangle with top-left corner at (`x` , `y`), width `w` , height `h` , and color `cl`
`dc`	`[x, y, r, cl]`	Draw a circle with center at (`x` , `y`), radius `r` , and color `cl`
`dfc`	`[x, y, r, cl]`	Draw a filled circle with center at (`x` , `y`), radius `r` , and color `cl`
`dt`	`[x, y, t, cl]`	Draw text `t` with top-left corner at (`x` , `y`) and color `cl`
`db`	`[x, y, w, h, [bmp]]`	Draws a RGB888 bitmap array `[bmp]` with top-left corner at (`x` , `y`) and size of (`w` , `h`)

Example

Here's an example JSON object to draw a red circle, a blue rectangle, and the text "Hello" in green:

json

```
{"draw":[
 {"dc": [28, 4, 3, "#FF0000"]},
 {"dr": [20, 4, 4, 4, "#0000FF"]},
```

```
 {"dt": [0, 0, "Hello", "#00FF00"]}
]}
```

Display Text in Colored Fragments

AWTRIX 3 allows you to present text where specific fragments can be colorized. Use an array of fragments with `"t"` representing the text fragment and `"c"` denoting the color's hex value.

json

```
{
  "text": [
    {
      "t": "Hello, ",
      "c": "FF0000"
    },
    {
      "t": "world!",
      "c": "00FF00"
    }
  ],
  "repeat": 2
}
```

Sending Multiple Custom Apps Simultaneously

AWTRIX 3 enables you to dispatch multiple custom apps in a single action. Instead of transmitting one custom app object, you can forward an array of objects.

e.g. MQTT Topic: `/custom/test`

json

```
[
  {"text":"1"},
```

```
  {"text":"2"}
]
```

Handling of Multiple Custom Apps:

- **Suffix Assignment**: Internally, the app name receives a suffix, turning it into formats like `test0`, `test1`, etc.
- **Updates**: You can refresh each app individually or update all of them collectively.
- **Deletion**:
  - When erasing apps, AWTRIX doesn't match the exact app name. Instead, it identifies apps that begin with the specified name.
  - To expunge all associated apps, send an empty payload to `/custom/test`. This action will remove `test0`, `test1`, and so on.
  - To eradicate a single app, direct the command to, for instance, `/custom/test1`.
  - Caution: Deleting just one app may upset the sequence of the remaining apps in the loop, as there's no provision for placeholders to retain order.

Delete a Custom App

To remove a custom app, dispatch an empty payload/body to the associated topic or URL.

Dismiss Notification

Easily dismiss a notification that was configured with `"hold": true`.

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/notify/dismiss`	`http://[IP]/api/notify/dismiss`	Empty payload/body	POST

Switch Apps

Navigate to the next or preceding app.

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/nextapp`	`http://[IP]/api/nextapp`	Empty payload/body	POST
`[PREFIX]/previousapp`	`http://[IP]/api/previousapp`	Empty payload/body	POST

Switch to Specific App

Directly transition to a desired app using its name.

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/switch`	`http://[IP]/api/switch`	`{"name":"Time"}`	POST

Built-in App Names:

- `Time`
- `Date`
- `Temperature`
- `Humidity`
- `Battery`

For custom apps, employ the name you designated in the topic or HTTP parameter. In MQTT, if `[PREFIX]/custom/test` is your topic, then `test` would be the app's name.

Change Settings

Adjust various settings related to the app display.

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/settings`	`http://[IP]/api/settings`	JSON	GET/POST

JSON Properties

You can adjust each property in the JSON object according to your preferences. Including a property is optional.

Key	Type	Description	Value Range	Default
`ATIME`	number	Duration an app is displayed in seconds.	Positive integer	7
`TEFF`	number	Choose between app transition effects.	0-10	1
`TSPEED`	number	Time taken for the transition to the next app in milliseconds.	Positive integer	500
`TCOL`	string/array of	Global text color.	RGB array or hex color	N/A

Key	Type	Description	Value Range	Default
	ints			
TMODE	integer	Changes the time app style.	0-4	1
CHCOL	string/array of ints	Calendar header color of the time app.	RGB array or hex color	#FF0000
CBCOL	string/array of ints	Calendar body color of the time app.	RGB array or hex color	#FFFFFF
CTCOL	string/array of ints	Calendar text color in the time app.	RGB array or hex color	#000000
WD	boolean	Enable or disable the weekday display.	true / false	true
WDCA	string/array of ints	Active weekday color.	RGB array or hex color	N/A
WDCI	string/array of ints	Inactive weekday color.	RGB array or hex color	N/A
BRI	number	Matrix brightness.	0-255	N/A
ABRI	boolean	Automatic brightness control.	true / false	N/A
ATRANS	boolean	Automatic switching to the next app.	true / false	N/A
CCORRECTION	array of ints	Color correction for the matrix.	RGB array	N/A
CTEMP	array of ints	Color temperature for the matrix.	RGB array	N/A
TFORMAT	string	Time format for the TimeApp.	Varies (see below)	N/A
DFORMAT	string	Date format for the DateApp.	Varies (see below)	N/A
SOM	boolean	Start the week on Monday.	true / false	true
BLOCKN	boolean	Block physical navigation keys (still sends input to MQTT).	true / false	false
UPPERCASE	boolean	Display text in uppercase.	true / false	true
TIME_COL	string/array of ints	Text color of the time app. Use 0 for global text color.	RGB array or hex color	N/A
DATE_COL	string/array of ints	Text color of the date app. Use 0 for global text color.	RGB array or hex color	N/A
TEMP_COL	string/array of ints	Text color of the temperature app. Use 0 for global text color.	RGB array or hex color	N/A
HUM_COL	string/array of ints	Text color of the humidity app. Use 0 for global text color.	RGB array or hex color	N/A

Key	Type	Description	Value Range	Default
`BAT_COL`	string/array of ints	Text color of the battery app. Use 0 for global text color.	RGB array or hex color	N/A
`SSPEED`	integer	Scroll speed modification.	Percentage of original scroll speed	100
`TIM`	boolean	Enable or disable the native time app (requires reboot).	`true` / `false`	true
`DAT`	boolean	Enable or disable the native date app (requires reboot).	`true` / `false`	true
`HUM`	boolean	Enable or disable the native humidity app (requires reboot).	`true` / `false`	true
`TEMP`	boolean	Enable or disable the native temperature app (requires reboot).	`true` / `false`	true
`BAT`	boolean	Enable or disable the native battery app (requires reboot).	`true` / `false`	true
`MATP`	boolean	Enable or disable the matrix. Similar to `power` Endpoint but without the animation.	`true` / `false`	true
`VOL`	integer	Allows to set the Volume of the Buzzer and DFplayer	0-30	true
`OVERLAY`	string	Sets a global effect overlay (cannot be used with app specific overlays)	Varies (see below)	N/A

Color Values: Can either be an RGB array (e.g., `[255,0,0]`) or a valid 6-digit hexadecimal color value (e.g., "#FF0000" for red).

Overlay effects:

- "clear"
- "snow"
- "rain"
- "drizzle"
- "storm"
- "thunder"
- "frost"

Timeformats:

bash

```
%H:%M:%S     13:30:45
%l:%M:%S     1:30:45
%H:%M        13:30
%H %M        13.30 with blinking colon
%l:%M        1:30
%l %M        1:30 with blinking colon
%l:%M %p     1:30 PM
%l %M %p     1:30 PM with blinking colon
```

Dateformats:

bash

```
%d.%m.%y     16.04.22
%d.%m        16.04
%y-%m-%d     22-04-16
%m-%d        04-16
%m/%d/%y     04/16/22
%m/%d        04/16
%d/%m/%y     16/04/22
%d/%m        16/04
%m-%d-%y     04-16-22
```

Transition effects:

bash

```
0 - Random
1 - Slide
2 - Dim
3 - Zoom
4 - Rotate
5 - Pixelate
6 - Curtain
7 - Ripple
8 - Blink
```

```
9 - Reload
10 - Fade
```

Update

You can initiate the firmware update either through the update button in HA or using the following:

MQTT Topic	HTTP URL	Payload/Body	HTTP Header	HTTP Method
`[PREFIX]/doupdate`	`http://[IP]/api/doupdate`	JSON	empty payload/body	POST

Reboot Awtrix

If you need to restart the Awtrix:

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`[PREFIX]/reboot`	`http://[IP]/api/reboot`	-	POST

Erase Awtrix

WARNING: This action will format the flash memory and EEPROM but will not modify the WiFi Settings. It essentially serves as a factory reset.

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`N/A`	`http://[IP]/api/erase`	-	POST

Clear Settings

WARNING: This action will reset all settings from the settings API. It does not reset the flash files and WiFi Settings.

MQTT Topic	HTTP URL	Payload/Body	HTTP Method
`N/A`	`http://[IP]/api/resetSettings`	-	POST