MQTT command overview

This document provides an overview of all the commands that can be used to communicate with the Jimbi over MQTT.

Every command/request is preceded by either a "GET" to request the status or a "SET" to update the status. All request messages will have a response message identical to the default “PUSH” messages which are triggered when new patterns or colors are triggered. All the information is organized in a JSON structure which is send over MQTT, this makes it possible to update multiple channels simultaneously. Spaces in JSON data are allowed for formatting but are not recommended due to data overhead.

The Jimbi can be configured as a MQTT client through the web interface.
Provide al the required information and save the settings. After a reboot of the Jimbi the client will automatically try to connect to the server.

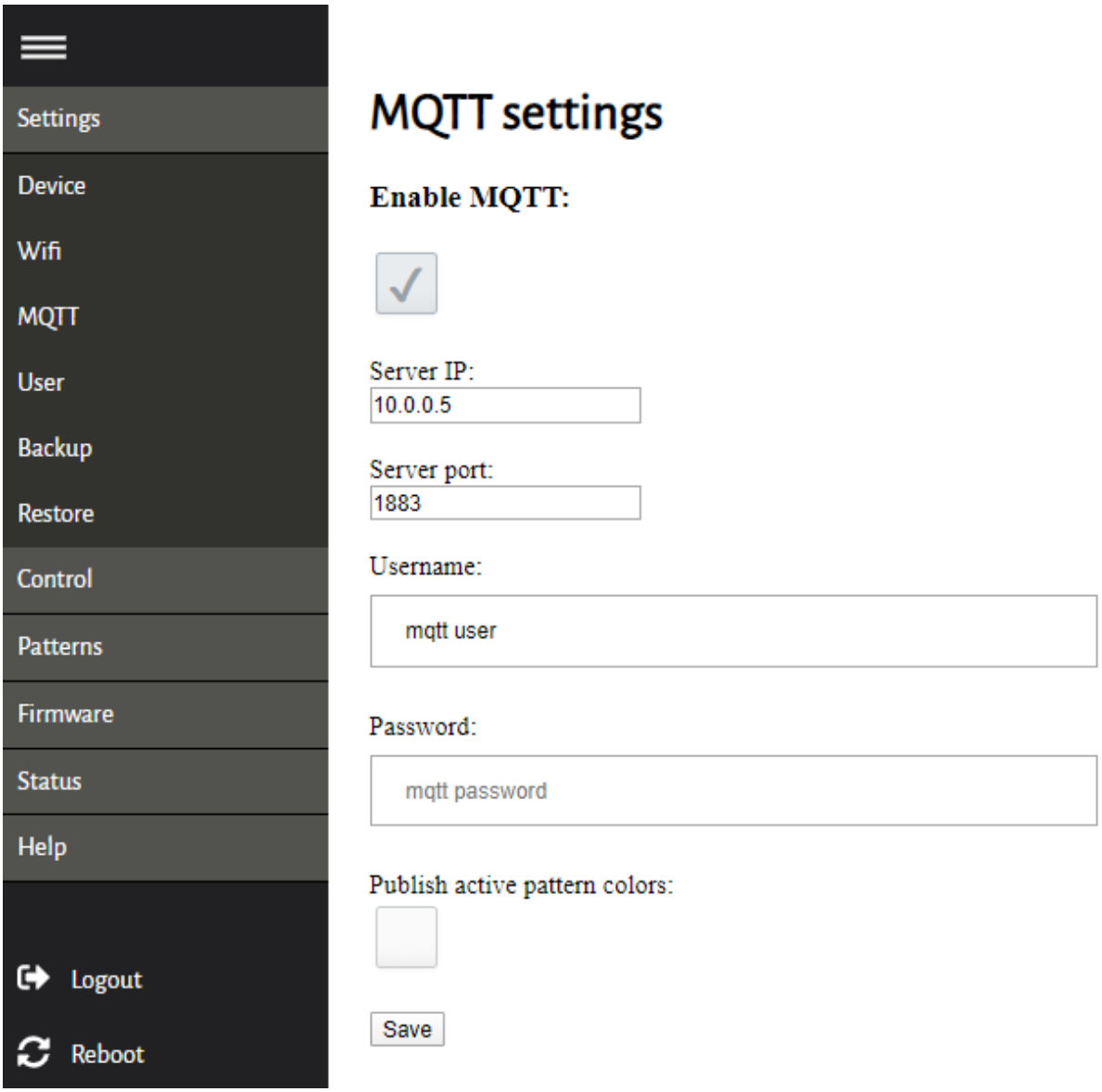

Figure 1: MQTT configuration page

Table of Contents

PWM Channel Message

With the PWM channel message it's possible to directly set the value of the PWM outputs.

SET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Channels: 1 - 48
- Values: 0 – 255

JSON data:
{"SET": {"PWM": {"channel1": value1,"channel2": value2}}}

Example
{"SET": {"PWM": {"1": 50,"2": 60,"5": 70,"48": 3}}}

Formatted:
```
{
  "SET":  {
          "PWM":  {
            "1":  50,
            "2":  60,
            "5":  70,
            "48":  3
          }
  }
}
```

GET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Channels: 1 - 48
- Values: NA

JSON data:
{"GET": {"PWM": {"channel1": NA,"channel2": NA}}}

Example
{"GET": {"PWM": {"1": "NA","2": "NA","5": "NA","48": "NA"}}}

Formatted:

```
{
  "GET":  {
    "PWM":  {
      "1":  "NA",
      "2":  "NA",
      "5":  "NA",
      "48":  "NA"
    }
  }
}
```

PUSH Message

Direction:
- Source: Jimbi
- Destination: MQTT

Topic: <device name>/<jsondata>

Parameters:
- Channels: 1 - 48
- Values: 0 - 255

JSON data:
{"PUSH": {"PWM": {"channel1": value1,"channel2": value2}}}

Example
{"PUSH": {"PWM": {"1": 50,"2": 60,"5": 70,"48": 3}}}

Formatted:
```
{
  "PUSH":  {
    "PWM":  {
      "1":  50,
      "2":  60,
      "5":  70,
      "48":  3
    }
  }
}
```

Figure 1

RGB Color Message

With the RGB LED Color message it is possible to set the color of a RGB or RGBW Led strip. When a RGBW LED Strip is connected the value is converted from RGB to RGBW

SET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Channels: 1 - 16
- Color: R, G and/or B
- Values: 0 - 255

JSON data:
{"SET": {"RGB": {"channel1": {"R": Rvalue,"G": Gvalue,"B": Bvalue},"channel2": {"R": Rvalue,"G": Gvalue,"B": Bvalue}}}}

Example
{"SET": {"RGB": {"1": {"R": 100,"G": 0,"B": 0},"2": {"R": 0,"G": 100,"B": 0}}}}

Formatted:

```
{
  "SET":  {
    "RGB":  {
      "1":  {
        "R":  100,
        "G":  0,
        "B":  0
      },
      "2":  {
        "R":  0,
        "G":  100,
        "B":  0
      }
    }
  }
}
```

GET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Channels: 1 - 16
- Color: R, G and/or B
- Values: NA

JSON data:
{"GET": {"RGB": {"channel1": {"R": "NA","G": "NA","B": "NA"},"channel2": {"R": "NA","G": "NA","B": "NA"}}}}

Example
{"GET": {"RGB": {"1": {"R": "NA","G": "NA","B": "NA"},"2": {"R": "NA","G": "NA","B": "NA"}}}}

Formatted:

```
{
  "GET":  {
    "RGB":  {
      "1":  {
        "R":  "NA",
        "G":  "NA",
        "B":  "NA"
      },
      "2":  {
        "R":  "NA",
        "G":  "NA",
        "B":  "NA"
      }
    }
  }
}
```

PUSH Message

Direction:
- Source: Jimbi
- Destination: MQTT

Topic: <device name>/<jsondata>

Parameters:
- Channels: 1 - 16
- Color: R, G and/or B
- Values: 0 - 255

JSON data:
{"PUSH": {"RGB": {"channel1": {"R": Rvalue,"G": Gvalue,"B": Bvalue},"channel2": {"R": Rvalue,"G": Gvalue,"B": Bvalue}}}}

Example
{"PUSH": {"RGB": {"1": {"R": 100,"G": 0,"B": 0},"2": {"R": 0,"G": 100,"B": 0}}}}

Formatted:

```
{
  "PUSH":  {
    "RGB":  {
      "1":  {
        "R":  100,
        "G":  0,
        "B":  0
      },
      "2":  {
        "R":  0,
        "G":  100,
        "B":  0
      }
    }
  }
}
```


Figure 1

RGBW Color Message

With the RGBW LED Color message it's possible to set the color of a RGBW LEDStrip.

SET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Channels: 1 - 12
- Color: R, G, B and/or W
- Values: 0 - 255

JSON data:
{"SET": {"RGBW": {"channel1": {"R": Rvalue,"G": Gvalue,"B": Bvalue,"W": Wvalue},"channel2": {"R": Rvalue,"G": Gvalue,"B": Bvalue,"W": Wvalue}}}}

Example
{"SET": {"RGBW": {"1": {"R": 100,"G": 0,"B": 0,"W": 0},"2": {"R": 0,"G": 100,"B": 0,"W": 0}}}}

Formatted:

```
{
  "SET":  {
    "RGBW":  {
      "1":  {
        "R":  100,
        "G":  0,
        "B":  0,
        "W":  0
      },
      "2":  {
        "R":  0,
        "G":  100,
        "B":  0,
        "W":  0
      }
    }
  }
}
```

GET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Channels: 1 - 12
- Color: R, G, B and/or W
- Values: NA

JSON data:
{"GET": {"RGBW": {"channel1": {"R": "NA","G": "NA","B": "NA","W": "NA"},"channel2": {"R": "NA","G": "NA","B": "NA","W": "NA"}}}}

Example
{"GET": {"RGBW": {"1": {"R": "NA","G": "NA","B": "NA","W": "NA"},"2": {"R": "NA","G": "NA","B": "NA","W": "NA"}}}}

Formatted:

```
{
  "GET":  {
    "RGBW":  {
      "1":  {
        "R":  "NA",
        "G":  "NA",
        "B":  "NA",
        "W":  "NA"
      },
      "2":  {
        "R":  "NA",
        "G":  "NA",
        "B":  "NA",
        "W":  "NA"
      }
    }
  }
}
```

PUSH Message

Direction:
- Source: Jimbi
- Destination: MQTT

Topic: <device name>/<jsondata>

Parameters:
- Channels: 1 - 12
- Color: R, G, B and/or W
- Values: 0 - 255

JSON data:
{"PUSH": {"RGBW": {"channel1": {"R": Rvalue,"G": Gvalue,"B": Bvalue,"W": Wvalue},"channel2": {"R": Rvalue,"G": Gvalue,"B": Bvalue,"W": Wvalue}}}}

Example
{"PUSH": {"RGBW": {"1": {"R": 100,"G": 0,"B": 0,"W": 0},"2": {"R": 0,"G": 100,"B": 0,"W": 0}}}}

Formatted:

```
{
  "PUSH":  {
    "RGBW":  {
      "1":  {
        "R":  100,
        "G":  0,
        "B":  0,
        "W":  0
      },
      "2":  {
        "R":  0,
        "G":  100,
        "B":  0,
        "W":  0
      }
    }
  }
}
```


Figure 1

Static Color Message

With the Static Color message it is possible to request the current static colors or overrule the current static colors from the values loaded from the settings at start-up This new setting will be lost after a reboot of the device (stored settings will be loaded again)

SET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- TR values:
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
  - IDLE - Idle (default, empty or invalid trigger value will also save to IDLE
- RGB(W) values:
  - #FFFFFF - sent RGB value (White is done automatically based on RGB value and output setting)
  - #FFFFFFFF - sent RGBW value

JSON data:
{"SET": {"SC": [{"TR": value,"RGB/RGBW": value"}]}}

Examples

{"SET": {"SC": [{"TR": "IDLE","RGB": "#FFFFFF"}]}}
{"SET": {"SC": [{"TR": "IDLE","RGBW": "#FFFFFFFF"}]}}
{"SET": {"SC": [{"TR": "IDLE","RGB": "#FFFFFF"},{"TR": "TR1R","RGBW": "#FFFFFFFF"}]}}

Formatted:

{
  "SET": {
    "SC": [
      {
        "IDLE",
        "RGB": "#FFFFFF"
      },
      {
        "TR1R",
        "RGBW": "#FFFFFFFF"
      }
    ]
  }
}

GET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Values:
  - NA - Request all static colors
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
  - IDLE - Idle

JSON data:
{"GET": {"SC": "NA"}}

Example
{"GET": {"SC": "NA"}}

Formatted:

```
{
  "GET":  {
    "SC":  "NA"
  }
}
```

PUSH Message

Direction:
- Source: Jimbi
- Destination: MQTT

Topic: <device name>/<jsondata>

Parameters:
- TR values:
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
  - IDLE - Idle
- RGBW values:
  - #FFFFFFFF (The color will always be pushed as RGBW color)

JSON data:
{"PUSH": {"SC": [{"TR": value,"RGBW": value},{"TR": value,"RGBW": value},{"TR": value,"RGBW": value},{"TR": value,"RGBW": value},{"TR": value,"RGBW": value},{"TR": value,"RGBW": value},{"TR": value,"RGBW": value},{"TR": value,"RGBW": value},{"TR": value,"RGBW": value}]}}

Example
{"PUSH": {"SC": [{"TR": "TR1R","RGBW": "#FFFFFFFF"},{"TR": "TR1F","RGBW": "#FFFFFFFF"},{"TR": "TR2R","RGBW": "#FFFFFFFF"},{"TR": "TR2F","RGBW": "#FFFFFFFF"},{"TR": "TR3R","RGBW": "#FFFFFFFF"},{"TR": "TR3F","RGBW": "#FFFFFFFF"},{"TR": "TR4R","RGBW": "#FFFFFFFF"},{"TR": "TR4F","RGBW": "#FFFFFFFF"},{"TR": "TRSR","RGBW": "#FFFFFFFF"},{"TR": "TRSF","RGBW": "#FFFFFFFF"},{"TR": "IDLE","RGBW": "#FFFFFFFF"}]}}

Formatted:

```
{
  "PUSH":  {
    "SC":  [
      {
        "TR1R",
        "RGBW":  "#FFFFFFFF"
      },
      {
        "TR1F",
        "RGBW":  "#FFFFFFFF"
      },
      {
        "TR2R",
        "RGBW":  "#FFFFFFFF"
      },
```

```
    {
      "TR2F",
      "RGBW":  "#FFFFFFFF"
    },
    {
      "TR3R",
      "RGBW":  "#FFFFFFFF"
    },
    {
      "TR3F",
      "RGBW":  "#FFFFFFFF"
    },
    {
      "TR4R",
      "RGBW":  "#FFFFFFFF"
    },
    {
      "TR4F",
      "RGBW":  "#FFFFFFFF"
    },
    {
      "TRSR",
      "RGBW":  "#FFFFFFFF"
    },
    {
      "TRSF",
      "RGBW":  "#FFFFFFFF"
    },
    {
      "IDLE",
      "RGBW":  "#FFFFFFFF"
    }
   ]
  }
}
```

Figure 1

Brightness Message

With the brightness message it's possible to set the brightness of all channels simultaneously.

SET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Brightness: 0 - 100

JSON data:
{"SET": {"BR": value}}

Example
{"SET": {"BR": 50}}

Formatted:
```
{
  "SET":  {
    "BR":  50
  }
}
```

GET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Brightness: NA

JSON data:
{"GET": {"BR": "NA"}}

Example
{"GET": {"BR": "NA"}}

Formatted:
```
{
  "GET":  {
    "BR":  "NA"
  }
}
```

PUSH Message

Direction:
- Source: Jimbi
- Destination: MQTT

Topic: <device name>/<jsondata>

Parameters:
- Brightness: 0 - 100

JSON data:
{"PUSH": {"BR": value}}

Example
{"PUSH": {"BR": 50}}

Formatted:
```
{
  "PUSH":  {
    "BR":  50
  }
}
```

Figure 1

Trigger Message

The trigger message can have two directions: from the Jimbi to inform the user that an input has changed (PUSH) or to the Jimbi to simulate an input (SET)

SET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- TR values:
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
  - IDLE - Idle

JSON data:
{"SET": {"TR": value}}

Example
{"SET": {"TR": "TR1R"}}

Formatted:
```
{
  "SET":  {
    "TR1R"
  }
}
```

PUSH Message

Direction:
- Source: Jimbi
- Destination: MQTT

Topic: <device name>/<jsondata>

Parameters:
- TR values:
  - NA - Start directly
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
  - IDLE - Idle

JSON data:
{"PUSH": {"TR": value}}

Example
{"PUSH": {"TR": "TR1R"}}

Formatted:
```
{
  "PUSH":  {
    "TR1R"
  }
}
```

Figure 1

Trigger Block Message

With the trigger block message it is possible to disable a trigger when it is received, even though action for a trigger is configured on the webpage

SET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- TR values:
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
- BLK values:
  - T - True (Block the trigger)
  - F - False (Allow the trigger)

JSON data:
{"SET": {"TRBLK": {TR value,BLK value}}}

Example
{"SET": {"TRBLK": {"TR1R": "T"},{"TR1F": "F"}}}

Formatted:
```
{
  "SET":  {
    "TRBLK":
      {
        "TR1R":  "T"
      },
      {
        "TR1F":  "F"
      }
  }
}
```

GET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- TR values:
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
  - NA - Request all triggers

JSON data:
{"GET": {"TRBLK": {TR value: "NA"}}}

Example
{"GET": {"TRBLK": {"TR1R":"NA"}}}

{"GET": {"TRBLK": {"NA": "NA"}}}

Formatted:
```
{
  "GET":  {
    "TRBLK":
      {
        "TR1R": "NA"
      }
  }
}
```

PUSH Message

Direction:
- Source: Jimbi
- Destination: MQTT

Topic: <device name>/<jsondata>

Parameters:
- TR values:
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
- BLK values:
  - T - True (Block the trigger)
  - F - False (Allow the trigger)

JSON data:
{"PUSH": {"TRBLK": {TR value: BLK value}, ... }}

Example

{"PUSH":{"TRBLK":{"TR1R":"F"},{"TR1F":"F"},{"TR2R":"F"},{"TR2F":"F"},{"TR3R":"F"},{"TR3F":"F"},{"TR4R":"F"},{"TR4F":"F"}{"TRSR":"F"},{"TRSF":"F"}}}

Formatted:


```
{
  "PUSH":  {
    "TRBLK":
      {
        "TR1R": "F"
      },
      {
        "TR1F": "F"
      },
      {
        "TR2R": "F"
      },
      {
        "TR2F": "F"
      },
      {
        "TR3R": "F"
      },
      {
        "TR3F": "F"
      },
      {
```

```
      "TR4R": "F"
    },
    {
      "TR4F": "F"
    },
    {
      "TRSR": "F"
    },
    {
      "TRSF": "F"
    }
  ]
 }
}
```


Figure 1

Pattern message

With the pattern message it's possible to start a specific pattern, request the currently active pattern or request the pattern configured for a trigger.
If no trigger is specified (TR=NA) the pattern will start directly, otherwise the current trigger loaded from the settings at start-up will be overruled.

This new setting will be lost after a reboot of the device (stored settings will be loaded again)

SET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- TR values:
  - NA - Start directly
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
  - IDLE - Idle
- NAME values: ASCII
  - NA - Clear trigger
  - <pattern name>

JSON data:
{"SET": {"PT": {"TR": value},"NAME": "<pattern name>"}}

Example
{"SET": {"PT": {"TR": "NA""NAME": "Upstairs"}}}

Formatted:
```
{
  "SET":  {
    "PT":  {
      "NA",
      "NAME":  "Upstairs"
    }
  }
}
```

GET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- TR values:
  - NA - Running pattern
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
  - IDLE - Idle

JSON data:
{"GET": {"PT": {"TR": value}}}

Example
{"GET": {"PT": {"TR": "NA"}}}

Formatted:
```
{
  "GET":  {
    "PT":  {
      "NA"
    }
}
```

PUSH Message

For the PT message two different PUSH messages can be received depending on the initiater of the message.
If the running pattern changes, a PUSH message with only the name of the currently running pattern is sent.
But if the running pattern is requested through a GET command the message contains both a NAME and TR value from the requested trigger.

Direction:
- Source: Jimbi
- Destination: MQTT

Initiated by a change in running pattern:

Topic: <device name>/<jsondata>

Parameters:
- Values:
  - NA - if no pattern is active
  - <pattern name>

JSON data:
{"PUSH": {"PT": "<name of running pattern>"}}

Example
If a pattern is running:
{"PUSH": {"PT": "Upstairs"}}

If no pattern is running:
{"PUSH": {"PT": "NA"}}

Formatted:
```
{
  "PUSH":  {
    "PT":  "NA"
  }
}
```

Initiated by a GET request:

Topic: <device name>/<jsondata>

Parameters:

- TR values:
  - NA - Running pattern
  - TR1R - Trigger 1 rising edge
  - TR1F - Trigger 1 falling edge
  - TR2R - Trigger 2 rising edge
  - TR2F - Trigger 2 falling edge
  - TR3R - Trigger 3 rising edge
  - TR3F - Trigger 3 falling edge
  - TR4R - Trigger 4 rising edge
  - TR4F - Trigger 4 falling edge
  - TRSR - Trigger service key rising edge
  - TRSF - Trigger service key falling edge
  - IDLE - Idle
- NAME values:
  - <pattern name>

JSON data:
{"PUSH": {"PT": {"TR": value,"NAME": <name of running pattern>}}}

Example

{"PUSH": {"PT": {"TR": "NA","NAME": "Upstairs"}}}

{"PUSH": {"PT": {"TR": "IDLE","NAME": "NA"}}}

Formatted:

```
{
  "PUSH":  {
    "PT":  {
      "NA",
      "NAME":  "Upstairs"
  }
}
```

Figure 1

Pattern List Message

With the pattern list message it's possible to request the available patterns on the Jimbi

GET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Values: 0 - 99*

*) 0 - All available patterns
n - First n number of patterns

JSON data:
{"GET": {"PTL": <nr. of patterns>}}

Example
{"GET": {"PTL": 0}}

Formatted:
```
{
  "GET":  {
    "PTL":  0
  }
}
```

PUSH Message

Direction:
- Source: Jimbi
- Destination: MQTT

Topic: <device name>/<jsondata>

Parameters:
- Values PTL: <name of pattern n>

JSON data:
{"PUSH": {"PTL": ["<name of pattern 1>","<name of pattern 2>","<name of pattern 3>","<name of pattern 4>","<name of pattern n>"]}}

Example
{"PUSH": {"PTL": ["TestUpstairs","TestDownstairs","TestPWM","TestRGB","TestRGBW"]}}

Formatted:

```
{
  "PUSH":  {
    "PTL":  [
      "TestUpstairs",
      "TestDownstairs",
      "TestPWM",
      "TestRGB",
      "TestRGBW"
    ]
  }
}
```


Figure 1

IP message

With the IP message it is possible to request the assigned IP address from the Jimbi module

GET Message

Direction:
- Source: MQTT
- Destination: Jimbi

Topic: <device name>/<jsondata>

Parameters:
- Type: "full", "number" *

*) "full" = returns the full IP address: 192.168.1.10
"number" = returns the last number of the ip address: 10

JSON data:
{"GET": {"IP": "<length>"}}

Example
{"GET": {"IP": "full"}}

Formatted:
```
{
  "GET":  {
    "IP":  "full"
  }
}
```

PUSH Message

Direction:
- Source: Jimbi
- Destination: MQTT

Topic: <device name>/<jsondata>

Parameters:
- Value: <IP Address>

JSON data:
{"PUSH": {"IP": "<IP address>"}}

Example
{"PUSH": {"IP": "192.168.1.10"}}

Formatted:
```
{
  "PUSH":  {
    "IP":  "192.168.1.10"
  }
}
```

Figure 1