Документация

Описание устройства

Протокол передачи данных

AquaController Documentation (/Index.Php) 17 января 2017 Просмотров: 72

Для связи с устройством используется встроенный модуль связи ESP8266 модель ESP-07. Для обмена данных используется порт **8888** и сетевой протокол передачи данных **UDP**. Для обменом информации между устройством и клиентом пользователя используется внутренний протокол устройства основанный на JSON:

Устройство связи принимает команды трех типов: **GET**, **POST**, **INFO**:

Команды типа GET позволяют делать запрос на получения данных от устройства.

Синтаксис команд: все команды отправляются только в строчном виде. В устройстве присутствует внутренний валидатор запросов, однако проверки на валидность параметров не предусмотрено, и вся валидация параметров должна происходить на стороне клиента, поэтому при разработке необходимо учитывать этот момент. Так как использование не валидных параметров может вызвать неоднозначную работу устройства.

Формат команд GET:

{"status":"get","message":"device"}

status - тип отправляемой команды: GET, POST, INFO

message - команда устройству:

- device - информация об устройстве.
- canal - информация об состоянии каналов устройства
- timer_daily_state - информация о ежедневных таймерах устройства
- timer_hours_state - информация о ежечасных таймерах устройства
- timer_seconds_state - информация о секундных таймерах устройства
- temp_state - информация о температурных настройках устройства
- temp_sensors - информация о температурных датчиках устройства

data - дополнительный параметр для передачи параметров запроса, в запросе типа GET не используется.

На каждый запрос GET устройство шлет ответ в виде своего внутреннего JSON. Такие запросы и примеры таких ответов представлены ниже:

Информация об устройстве:

- **Запрос:**

```
{
        "status": "get",
        "message": "device"
}
```

- **Ответ:**

```
{
        "status": "get",
        "message": "device",
        "data": {
                "version": "AQ_CH08W",
                "max_timer": 8,
                "max_temp_sensor": 4,
                "min_temp": 1600,
                "max_temp": 3500
        }
}
```

- *"version": "AQ_CH08W" - версия устройства*
- *"max_timer": 8 - максимальное количество таймеров (ежедневных, ежечасных, секундных)*
- *"max_temp_sensor": 4 - максимально возможное количество подключенных температурных датчиков*
- *"min_temp": 1600 - порог минимальной температуры*

- *"max_temp": 3500 - порог максимальной температуры*

Информация об состоянии каналов устройства:

- **Запрос:**

```
{
        "status": "get",
        "message": "canal_state"
}
```

- **Ответ:**

```
{
        "status": "get",
        "message": "canal_state",
        "data": {
                "canal": [2, 2, 1, 1, 1, 1, 1, 1],
                "canal_timer": [3, 2, 1, 1, 1, 1, 1, 1]
        }
}
```

- *"canal": [2, 2, 1, 1, 1, 1, 1, 1] - состояние канала (1 - off, 2 - on)*
- *"canal_timer": [3, 2, 1, 1, 1, 1, 1, 1] - настройки канала (1 - off, 2 - on, 3 - auto)*

Информация о ежедневных таймерах устройства:

- **Запрос:**

```
{
        "status": "get",
        "message": "timer_daily_state"
}
```

- **Ответ:**

```
{
        "status": "get",
        "message": "timer_daily_state",
        "data": {
                "daily_timer_hour_start": [0, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_hour_end": [0, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_min_start": [0, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_min_end": [0, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_state": [0, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_canal": [0, 0, 0, 0, 0, 0, 0, 0]
        }
}
```

- *"daily_timer_hour_start": [0, 0, 0, 0, 0, 0, 0, 0] - час включения таймера (0...23)*
- *"daily_timer_hour_end": [0, 0, 0, 0, 0, 0, 0, 0] - час выключения таймера (0...23)*
- *"daily_timer_min_start": [0, 0, 0, 0, 0, 0, 0, 0] - минута включения таймера (0...59)*
- *"daily_timer_min_end": [0, 0, 0, 0, 0, 0, 0, 0] - минута выключения таймера (0...59)*
- *"daily_timer_state": [0, 0, 0, 0, 0, 0, 0, 0] - состояние таймера (0 - off, 1 - on)*
- *"daily_timer_canal": [0, 0, 0, 0, 0, 0, 0, 0] - канал управляемый таймером (0...max_canal)*

Информация о ежечасных таймерах устройства:

- **Запрос:**

```
{
        "status": "get",
        "message": "timer_hours_state"
}
```

- **Ответ:**

```
{
        "status": "get",
        "message": "timer_hours_state",
        "data": {
                "hours_timer_min_start": [0, 0, 0, 0, 0, 0, 0, 0],
                "hours_timer_min_stop": [0, 0, 0, 0, 0, 0, 0, 0],
                "hours_timer_state": [0, 0, 0, 0, 0, 0, 0, 0],
                "hours_timer_canal": [0, 0, 0, 0, 0, 0, 0, 0]
        }
}
```

- *"hours_timer_min_start": [0, 0, 0, 0, 0, 0, 0, 0] - минута включения таймера (0...59)*

- *"hours_timer_min_stop": [0, 0, 0, 0, 0, 0, 0, 0] - минута выключения таймера (0...59)*
- *"hours_timer_state": [0, 0, 0, 0, 0, 0, 0, 0] - состояние таймера (0 - off, 1 - on)*
- *"hours_timer_canal": [0, 0, 0, 0, 0, 0, 0, 0] - канал управляемый таймером (0...max_canal)*

Информация о секундных таймерах устройства:

- **Запрос:**

```
{
        "status": "get",
        "message": "timer_seconds_state"
}
```

- **Ответ:**

```
{
        "status": "get",
        "message": "timer_seconds_state ",
        "data": {
                "second_timer_hour_start": [0, 0, 0, 0, 0, 0, 0, 0],
                "second_timer_min_start": [0, 0, 0, 0, 0, 0, 0, 0],
                "second_timer_duration": [0, 0, 0, 0, 0, 0, 0, 0],
                "second_timer_state": [0, 0, 0, 0, 0, 0, 0, 0],
                "second_timer_canal": [0, 0, 0, 0, 0, 0, 0, 0]
        }
}
```

- *"second_timer_hour_start": [0, 0, 0, 0, 0, 0, 0, 0] - час включения таймера (0...23)*
- *"second_timer_min_start": [0, 0, 0, 0, 0, 0, 0, 0] - минута включения таймера (0...59)*
- *"second_timer_duration": [0, 0, 0, 0, 0, 0, 0, 0] - длительность работы таймера в секундах (0...255)*
- *"second_timer_state": [0, 0, 0, 0, 0, 0, 0, 0] - состояние таймера (0 - off, 1 - on)*
- *"second_timer_canal": [0, 0, 0, 0, 0, 0, 0, 0] - канал управляемый таймером (0...max_canal)*

Информация о температурных настройках устройства:

- **Запрос:**

```
{
        "status": "get",
        "message": "temp_state"
}
```

- **Ответ:**

```
{
        "status": "get",
        "message": "temp_state",
        "data": {
                "temp_timer_state": [1, 0, 0, 0, 0, 0, 0, 0],
                "temp_timer_min_start": [2000, 2250, 0, 0, 0, 0, 0, 0],
                "temp_timer_max_end": [2600, 2850, 0, 0, 0, 0, 0, 0],
                "temp_timer_canal": [1, 0, 0, 0, 0, 0, 0, 0]
        }
}
```

- *"temp_timer_state": [1, 0, 0, 0, 0, 0, 0, 0] - состояние температурного таймера*
- *"temp_timer_min_start": [2000, 2250, 0, 0, 0, 0, 0, 0] - температура включения канала (min_temp...max_temp, кратна 50)*
- *"temp_timer_max_end": [2600, 2850, 0, 0, 0, 0, 0, 0] - температура выключения канала (min_temp...max_temp кратна 50)*
- *"temp_timer_canal": [1, 0, 0, 0, 0, 0, 0, 0] - канала управляемый таймером (0...max_canal)*

Информация о температурных датчиках устройства:

- **Запрос:**

```
{
        "status": "get",
        "message": "temp_sensors"
}
```

- **Ответ:**

```
{
        "status": "get",
        "message": "temp_sensors",
        "data": {
                "temp_sensors": [2400, 2650, 0, 0, 0, 0, 0, 0]
        }
}
```

- *"temp_sensors": [2400, 2650, 0, 0, 0, 0, 0, 0] - температура датчиков (min_temp...max_temp)*

Формат команд POST

Команды POST отличаются от команд GET только тем что в запросе необходимо в поле **data** указывать параметры запроса и поле **status** содержит параметр **post**. Формат данных для поля **data** аналогичен данным приходящим от устройства при запросе командой GET.

Примеры возможных запросов:

Изменение состояния каналов устройства:

- **Запрос:**

```
{
        "status": "post",
        "message": "canal_state",
        "data": {
                "canal_timer": [1, 0, 0, 0, 0, 0, 0, 0]
        }
}
```

Обратите внимание для изменения настроек каналов, отправлять нужно только параметр **canal_timer**, параметр **canal** не предназначен для изменения, он отвечает только за текущее состояние канала в зависимости от настроек таймеров или ручных настроек.

Изменение ежедневных таймеров устройства:

- **Запрос:**

```
{
        "status": "post",
        "message": "timer_daily_state",
        "data": {
                "daily_timer_hour_start": [1, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_hour_end": [2, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_min_start": [5, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_min_end": [55, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_state": [1, 0, 0, 0, 0, 0, 0, 0],
                "daily_timer_canal": [3, 0, 0, 0, 0, 0, 0, 0]
        }
}
```

Изменение ежечасных таймеров устройства:

- **Запрос:**

```
{
        "status": "post",
        "message": "timer_hours_state",
        "data": {
                "hours_timer_min_start": [45, 0, 0, 0, 0, 0, 0, 0],
                "hours_timer_min_stop": [56, 0, 0, 0, 0, 0, 0, 0],
                "hours_timer_state": [1, 0, 0, 0, 0, 0, 0, 0],
                "hours_timer_canal": [3, 0, 0, 0, 0, 0, 0, 0]
        }
}
```

Изменение секундных таймеров устройства:

- **Запрос:**

```
{
        "status": "post",
        "message": "timer_seconds_state",
        "data": {
                "second_timer_hour_start": [5, 0, 0, 0, 0, 0, 0, 0],
                "second_timer_min_start": [26, 0, 0, 0, 0, 0, 0, 0],
                "second_timer_duration": [125, 0, 0, 0, 0, 0, 0, 0],
                "second_timer_state": [1, 0, 0, 0, 0, 0, 0, 0],
                "second_timer_canal": [2, 0, 0, 0, 0, 0, 0, 0]
        }
}
```

Изменение температурных настроек устройства:

- **Запрос:**

```
{
        "status": "post",
        "message": "temp_state",
        "data": {
                "temp_timer_state": [0, 0, 0, 0, 0, 0, 0, 0],
                "temp_timer_min_start": [2200, 2250, 0, 0, 0, 0, 0, 0],
                "temp_timer_max_end": [2700, 2850, 0, 0, 0, 0, 0, 0],
                "temp_timer_canal": [5, 0, 0, 0, 0, 0, 0, 0]
        }
}
```

При POST запросе нет необходимости указывать все параметры в поле **data**. При передаче значений в устройство можно отправлять только те данные которые изменились на клиенте. На каждый выполненный **POST** запрос, ответом будет ответ аналогичный **GET** с тем же **message** параметром, но уже с измененными данными.

Пример:

- **Запрос**

```
{
        "status": "post",
        "message": "canal",
        "data": {
                "canal_timer": [1, 1, 1, 2, 3, 2, 2, 2]
        }
}
```

- **Ответ**

```
{
        "status": "get",
        "message": "canal",
        "data": {
                "canal": [1, 1, 1, 2, 1, 2, 2, 2],
                "canal_timer": [1, 1, 1, 2, 3, 2, 2, 2]
        }
}
```

- Обратите внимание что ответ на POST приходит всегда в полном виде независимо от тех параметров которые вы отправляете.

Формат команд INFO:

В данный момент запросы формата INFO используются только для внутренней передачи данных между Arduino и модулем связи ESP8266. В частности используется ESP8266 для передачи своего состояния и логирования процессов. Использование внешними клиентами в данной версии не предусмотрено.

Пример отправки логов WIFI соединения на устройство:

```
{
        "status": "info",
        "message": "device_log",
        "log": ""
}
```

‹ Назад (/index.php/other-pages/tag) Вперёд › (/index.php/doc/class-layout/3-columns)