MODULO DIY Mode Overview

Do you want to control the MODULO device via your own app or home automation platform? DIY Mode helps!

The DIY Mode is designed for IoT home automation users and developers who would like to control the MODULO device via existing home automation open-source platform or local HTTP client instead of eWeLink App. In DIY Mode, when the device is connected with the network, it will publish its services and capabilities according to the mDNS/DNS-SD standard. Before publishing the service, the device has enabled the HTTP server on the port declared by the DNS SRV record. Device exposes the capabilities through an HTTP-based RESTful API. Users can obtain device information, control the device by sending an HTTP API request.

Supported Device

Product Class	Device	Firmware version note
Single Channel DIY Plug	BASICR3 RFR3 MINI	Firmware 3.5.0 refers to v2.1 API protocol Firmware 3.3.0 refers to v1.4 API protocol
Single Channel DIY Dimmer	D1	Firmware 3.5.0 refers to v2.1 API protocol

eWeLink Mode and DIY Mode

The MODULO devices [1] can work in either eWeLink mode or DIY Mode, In eWeLink mode, the device is connected with eWeLink cloud and controlled by eWeLink APP, while in DIY Mode, device publishes its capability service and is controlled by HTTP Post request.

The steps of entering the DIY Mode and connecting to an existing WiFi network:

1. Entering the Compatible Pairing Mode (AP) by long press the paring button for 5 seconds after power on
2. Connecting the Access Point named **ITEAD-XXXXXXXXXX** with default password **12345678** via mobile phone or PC
3. Browser visits http://10.10.7.1/
4. Filling in the existing WiFi network SSID and password
5. Entering **DIY Mode** successfully with specific WiFi network connected.

Example for Single Channel DIY Plug (BASICR3, RFR3, MINI) enters DIY Mode:

1. Power on;

2. Long press the button for 5 seconds to enter Compatible Pairing Mode (AP)
   User tips: If the device has been paired with eWeLink APP, reset the device is necessary by long press the pairing button for 5 seconds, then press another 5 seconds for entering Compatible Pairing Mode (AP)
3. The LED indicator will blink continuously
4. From mobile phone or PC WiFi setting, an Access Point of the device named **ITEAD-XXXXXXXXXX** will be found, connect it with default password **12345678**
5. Open the browser and access http://10.10.7.1/
6. Next, fill in WiFi SSID and password that the device would have connected with
7. Succeed, now the device is in DIY Mode.

Example for Single Channel DIY Dimmer (D1) enters DIY Mode:

1. Power on;
2. Long press the pairing button of RM433 remote controller for 5 seconds to enter Compatible Pairing Mode (AP)
   User tips: If the device has been paired with eWeLink APP, reset the device is necessary by long press the pairing button of RM433 remote controller for 5 seconds, then press another 5 seconds for entering Compatible Pairing Mode (AP)
3. The dimmable Light which is connected with D1 will blink continuously (promptly jump 100% to 1%, 1% to 100% …)

Steps of 4 - 7 are same as the example of Single Channel DIY Plug.

Note:

- The user settings will be cleaned once the operation mode is changed from one to another.
- The WiFi router or AP should work in 2.4GHz and support mDNS service.
- LED blinking meanings
  Fast single blinking – The device does not connect to the WiFi network;
  Fast double blinking – The device connects to the WiFi successfully and is able to be discovered through mDNS and respond the request from LAN network.
- Once the device is already in DIY Mode, the WiFi configuration page of http://10.10.7.1/ is not accessible.
- If a wrong WiFi SSID or password was entered, the device will fail to connect with specific WiFi network, with 20 seconds timeout mechanism, the device stop connecting WiFi network, please try again with the example steps of 1-7.
- Official firmware upgrade is only available in eWeLink APP.
- The protocol v1.4 can be accessed via https://github.com/itead/Modulo_Devices_DIY_Tools

DIY Mode LAN Discovery Mechanism

DIY Mode LAN discovery implements IETF Multicast DNS protocol and DNS-Based Service Discovery protocol. [2]-[8]

Device mDNS Service Info Publish Process

The device publishes its own service (i.e. device capability) according to the mDNS/DNS-SD standard discovery protocol when the device is connected to LAN (Local Area Network).

The fields definition as follow:

Attribute	Description	Example
IP Address	The LAN IP Address is obtained through DHCP instead of the Link-Local address of IPv4/IPv6	
Hostname	The Hostname must be unique in LAN; Format: eWeLink_[Device ID]	eWeLink_10000000d0
Service Type	_ewelink._tcp	
Service Instance Name	The Service Instance Name must be unique in LAN; Max: 63 bytes (21 UTF8 Characters)	same as Hostname
TXT Record	One or more strings; No exceeded 255 bytes for each string; No exceeded 1300 bytes for the entire TXT record;	

TXT Record note：

1. TXT Record must contain below strings:
   “**txtvers**=1”, “**id**=[device ID]”, “**type**=[device type]”, “**apivers**=[device API interface version]”, “**seq**=[TXT Record serial number]”, “**data1**=[device information]”;
2. Optional strings:
   “**data2**=[device information]”, “**data3**=[device information]”, “**data4**=[device information]”
3. “**seq**=[TXT record sequence number]” indicates the order in which the TXT records are updated (the order in which the device status is updated). It is recommended to be a positive integer that increments from 1 (reset to 1 when the device restarts);
4. When the device information is longer than 249 bytes, the first 249 bytes must be stored in data1, and the remaining bytes are divided by length 249, which are stored in data2, data3, and data4. The complete device information format is a JSON object.

For BASICR3, RFR3, MINI mDNS txt record example:

data1={“switch”:”on”,”startup”:”stay”,”pulse”:”on”,”pulseWidth”:2000,”rssi”:-67,”fwVersion”:”3.5.0”}

For D1 (Dimmer) mDNS txt record example:

data1={“switch”: “on”,”mode”: 0,”brightness”: 50,”brightMin”:0,”brightMax”:255,”startup”: “on”,”rssi”:-67,”fwVersion”:”3.5.0”}

Whenever content other than seq changes, such as Service Instance Name is modified, device information is updated, etc., the device must multicast the corresponding DNS record (including the incremented seq) according to the mDNS/DNS-SD standard.

Discovery Process for Device Service

The discovery process must follow the mDNS/DNS-SD Discovery protocol to discover the Modulo DIY Mode device with “_ewelink._tcp” service type when your application or client connect with Internet (WiFi or Ethernet);

Here is the discovery process:

1. Search in the LAN for all devices with the service type _ewelink._tcp through the DNS PTR record.
2. Get the Hostname and Port of device service via parsing out the device DNS SRV record. (The default port is 8081)
3. Get device IP address via DNS A record or by other means.
4. Get the info of “device ID”, “Service Type”, “device API interface version” and “device information” via parsing out the device DNS TXT Record.

Note:

- When the “device type” of the device service does not match with the “device type” of your application or client, or the device API interface version of the device service is higher than your application or client’s, the application or client should not parse out the “device information” and call the device API interface, but prompt the specific reason for users why the device cannot be controlled via LAN and suggest to upgrade the application or client.
- The application or client get the IP address of the device via DNS A record when the device API interface is about to be called.

RESTful API Control Protocol（HTTP POST）

The device must open the HTTP server in the port declared by the DNS SRV record before the device publishes its services; the device publishes the capabilities through a HTTP-based RESTful API. Because of the LAN’s security and device’s limited computing power, this document recommends that the device provides HTTP instead of HTTPS interface.

The device type and API interface version of each product is shown as below:

Product	type	apivers
BASICR3 RFR3 MINI	diy_plug	1
D1	diylight	1

RESTful API Request and Response Format

URL: http://[ip]:[port]/[path]
Return value format: json
Method: HTTP post
RESTful API Request works in POST method and JSON formatted request body.

```
{
  "deviceid": "100000140e",
  "data": {
     "switch": "on"
    }
}
```

Attribute	Type	Example	Optional	Description
deviceid	String	100000140e	Yes	The device ID for this request.
data	Object	{“switch”: “on”}	No	Object type, Specific device information setting when controlling the device. Empty object when check the device information

RESTful API Response works in 200 OK HTTP response code and JSON formatted response body.

```
{
  "seq": 2,
  "error": 0,
  "data": {
     "signalStrength": -67
  }
}
```

Attribute	Type	Optional	Description
seq	Number	No	The order of device status update (also the order of TXT Record update)
error	Number	No	Whether the device successfully sets the specified device information. - **0:** successfully - **400:** The operation failed and the request was formatted incorrectly. The request body is not a valid JSON format. - **401:** The operation failed and the request was unauthorized. Device information encryption is enabled on the device, but the request is not encrypted. - **404:** The operation failed and the device does not exist. The device does not support the requested deviceid. - **422:** The operation failed and the request parameters are invalid. For example, the device does not support setting specific device information.
data	Object	No	Object type, it returns specific device info when check the device information

Note:

- Due to the device computing capability, the time interval of each HTTP request should be no less than 200ms.
- The default Port: 8081