Smart Node

Product: - Door Lock Controller APIs

v3.0

The APIs between the Home automation hardware and smartphone application

1.1 Introduction

Here, in following document we have listed the APIs needed to communicate with the hardware To be more specific we can use UDP or TCP for local communication with the hardware. In addition, the MQTT protocol is used for over the internet communication.

Here, the local broadcast command(“MST”) and the device authentication command(“LIN”) are the only two commands that works in local network. All other commands works with UDP as well as MQTT.

1.2 UDP communication

UDP is used for only local hardware communication. UDP is comparatively lightweight and quick response protocol compared to TCP. Since, we need one to many type communication UDP serves our purpose well. The first thing about our hardware UDP connection is that our hardware will always broadcast all commands/responses on UDP port 13000. So, if you are want to communicate with the hardware you need to send the commands to UDP port 13001 to that particular hardware IP or you can also broadcast that command (We recommend you to send to the command to the specific IP because it will reduce un-necessary work load to other hardware devices). Our hardware will respond on both UDP as well as MQTT to any command that comes from either UDP or MQT. Therefore, any action taken by one app user will be live reflected on all other user’s smartphone as well.

1.3 TCP communication

For TCP communication, please use port 13002. All commands will work on TCP and UDP both. To discover all the available Smart Node devices in the local network, please broadcast "MST" command in UDP and fetch the response to find all the IPs of that hardware for all the further TCP communication.

1.4 MQTT communication

MQTT is a communication protocol made for low powered IOT devices. It is a lightweight and a reliable messaging system for over internet communication. MQTT is a publish and subscribe based messaging system. It includes one Broker (kind of server) to manage all commands to be handle on its individual topics. In our case, we have our own broker hosted at a specific server location. To connect to the broker, it is mandatory to provide the correct username and password. Every individual hardware has its own unique topics for communication. To get that topic information, you first need to successfully login to that device. In successful login response, you will get “slave id”, “token” and “encryption key”. This are the main parameters you must need for further command. (The login process for any hardware will be done only on Local LAN network. We have discussed its process flow in this document further)

For debug purpose, you can use “Packet Sender” software on Windows/Mac PC to check UDP/TCP commands. You can use “MQTT Lens” an chrome extension to debug MQTT commands. Keeping in mind that our hardware will respond same response multiple time on UDP to resolve problem of package drop, but on MQTT it will be responded a single time only.

Command Flow

Here are the list of commands to discover the devices in local network, to add a device in the application, find status of the nodes, to control the nodes and many more.

To add a new hardware to the application

To add a new device to our application, it is mandatory that the application and the hardware be in the same network. To identify if the device is present in local network or not, you need to broadcast ‘MST’ command on UDP. As a result, all the devices in the same network will respond with its name and device Id. Here, if the local network includes multiple devices then the responses will be multiple. In this way, you can also identify the individual IPs of that hardware device.

1. ‘MST’ command

This command is used to discover devices in the local network. All Smart Node device in the local network will reply as shown.

Command from App to Hardware

```
{“cmd”:”MST”}
```

Response from the Hardware

```
{"cmd":"MST","device_id":1638929,"m_name":"SN-20002301060943I6","vendor":"VDT","wifi_ver":"1.2.5","hw_ver":"L.1.0","serial":49,"type":"standalone"}
```

Key	Value Type	Ignorable	Description
m_name	string	No	The user defined name of the device
device_id	string	No	The temporary unique number of each hardware which is needed to add a new device to the application
hw_ver	string	No	The hardware version of the device
wifi_ver	string	No	The firmware version of the Wi-Fi chip
vendor	string	Yes	“VDT” (no other possibility as of now)
serial	integer	No	It is a simple counter which keeps increasing after every new command. Mainly used to detect multiple UDP responses for a single command
type	string	Yes	“standalone” (no other possibility as of now)

2. ‘LIN’ command

To add a new curtain device to our application, it is mandatory that the application and the hardware be in the same network. This ‘LIN’ command is sent from the application to get a response from the hardware. The hardware responds with all the important parameters which are needed to operate the device. Here, we have shown a sample command to be sent by the app to add a curtain device. In this command, the app need to add “device_id” parameter which we get from the response of MST command.

Command from App to Hardware

```
{"cmd":"LIN","user":"Admin","pin":"1234","device_id":"4511681"}
```

Key	Value Type	Ignorable	Description
user	string	Yes	“Admin” (no other possibility as of now)
pin	string	No	The pin of the device
device_id	string	No	A temporary unique number of that device(received from MST response)

Response from the Hardware

If the login credentials are correct

```
{"cmd":"LIN","wifi_ver":"1.2.5","hw_ver":"L.1.0","device_id":1638929,"slave":"2000230106094316","status":"success","type":"standalone","encryption_key":"4f6be25e79f570ec0883ff7a96118da4","topic":"2000230106094316","token":"001d08f4e0cbb7a38e7a","nodes":2,"dimmer_support":[0,0],"vendor":"VDT","CNF":"","serial":71}
```

Key	Value Type	Ignorable	Description
slave	string	No	The serial number of the device (It is needed for all further communication)

device_id	string	No	The temporary unique number of each hardware(received from MST response)
status	string	No	“success” (if the credentials are correct otherwise “error”)
type	string	Yes	“standalone” (no other possibility as of now)
nodes	integer	No	The total number of nodes in the device
dimmer_support	array of integer	Yes	
encryption_key	string	Yes	AES 256 encryption key (for future use)
topic	string	Yes	It is always same as slave
token	string	No	A unique authentication string. To operate/control the device, we need to send this token with the rest of the commands. (It will only change if anyone factory resets the device)
hw_ver	string	NO	The hardware version of the device
wifi_ver	string		The firmware version of the Wi-Fi chip

If the credentials are incorrect then the device will respond as follows.

```
{"cmd":"LIN","wifi_ver":"1.2.4","hw_ver":"L.1.0","device_id":2695322,"status":"error","type":"unknown","serial":238}
```

Key	Value Type	Description
status	string	“error”

3. ‘STS’ command

This command is used to get the current state of the curtain. The command includes the information such as open/close status, child lock status as well as the total number of schedules set for each curtain.

Command from App to Hardware

{"cmd":"STS","slave":"2000230106094316","token":"14e50a705a14256f18b8"}

Response from the Hardware

{"cmd":"STS","slave":"2000230106094316","m_name":"SN-2000230106094316","dimmer":[255,255],"auto_off":[3,3],"auto_off_sts":[0,0],"button":"0102","val":"00","dimmer_type":"XX","dval":"XX","touch_lock":"NN","user_locked":"NN","schedule_info":"0000","wifi_ver":"1.2.5","arm_ver":"1.0.0.5","hw_ver":"L.1.0","WiFi":"SmartNode","CNF":"","tag":"formal","temperature":103,"triac_t":6503.6,"signal":76,"serial":77}

Key	Value Type	Ignorable	Description
slave	string	No	The serial number of the device
val	string	No	The status of each node (0 = off, A = on)
dimmer	array of Integer	Yes	
dimmer_type	string	Yes	
touch_lock	string	No	The child lock status of each node (N = no, Y = yes)

m_name	string	No	The user defined name of the device
schedule_info	string	No	The total number of schedules set in each individual nodes
tag	string	No	The reason why the status is sent Possible values : formal/schedule/scene/auto-off

Device Action commands

The Device action commands are used to open/close the curtain or child lock/unlock. Here shown are some sample commands and its response from our hardware. In all the commands shown below, there are two parameters that are mandatory to include: **slave and token**.

4. ‘UPD’ command

This command is used to open/close the curtain. The response for UPD command is “SET” command. We need to keep in mind that any manual change done by the user (through touch/physical switch on switch-board), the device will respond the same SET command as well.

Command from App to Hardware

```
{"cmd":"UPD","slave":"2022050711485903","token":"11c8b49f8b76624",
"by":"M6083fbfba74176263cd4badf","node":2,"val":"0","d_val":255}
```

Key	Value Type	Description
slave	string	Serial number of the device
node	integer	The node number of the device
val	string	To switch on or off.

		0 means to switch off A means to switch on
d_val	integer	The dimmer value to be sent in 0-100% range and if the node is non dimmable it will send 255 as value

Response from the Hardware

```
    { "cmd": "SET","slave": "2000230106094316","button": "01", "node":
1"val": "A","dval": "X","dimmer": 255,"touch_lock": 0,"user_locked":
0,"schedule_info": 0,"auto_off_sts": 0,"auto_off": 3,"tag": 2,"serial": 81}
```

Key	Value Type	Description
button	string	The switch number of the device (01,02,03,...)
node	integer	The switch number of the device (1,2,3,...)
val	string	status (A = on, 0=off)
dimmer	integer	This will give dimmer values in 0-100% range and if the particular node is non dimmable it will send 255 as value.

5. ‘000’ command (scene execution)

This command is used to open/close all the curtains of the same device simultaneously using a single command. The response for “000” command is “STS” command.

Command from App to Hardware

```
    {"cmd":"000","slave":"2000230106094316","token":"40119c87725d
48  331f0a","data":"AXAX","dimmer":[100,100]}
```

Key	Value Type	Description
slave	string	Serial number of the device
data	string	data : *NNS* *NN*: node number of the device (01,02,03,...) *S*: To convert that node to dimmable or non-dimmable (Y = dimmable, N = only on/off)
dimmer	array of Integer	This will give dimmer values in 0-100% range and if the node is non dimmable it will send 255 as value

Response from the Hardware

{"cmd":"STS","slave":"2000230106094316","m_name":"SN-2000230106094316","dimmer":[255,255],"auto_off":[3,3],"auto_off_sts":[0,0],"button":"0102","val":"00","dimmer_type":"XX","dval":"XX","touch_lock":"NN","user_locked":"NN","schedule_info":"0000","wifi_ver":"1.2.5","arm_ver":"1.0.0.5","hw_ver":"L.1.0","WiFi":"SmartNode","CNF":"","tag":"formal","temperature":103,"triac_t":6503.6,"signal":76,"serial":77}

tag = “scene” (reason for this status command. Here it is scene execution)

Key	Value Type	Ignorable	Description
slave	string	No	The serial number of the device
tag	string	Yes	
val	string	No	The status of each node (0 = off, A = on)
dimmer	array of Integer	Yes	
dimmer_type	string	Yes	
touch_lock	string	No	The child lock status of each node (N = no, Y = yes)
m_name	string	No	The user defined name of the device
schedule_info	string	No	The total number of schedules set in each individual nodes
tag	string	No	The reason why the status is sent Possible values : formal/schedule/scene/auto-off

All other keys are explained in the STS section