M2MLIGHT IOT PLATFORM

USER GUIDE

Version 1.5

Date: Oct 30, 2023

Contenido

1. INTRODUCTION..... 2
2. REGISTER USER..... 2
3. SENSORS..... 3
4. ACTUATORS..... 8
5. ALERTS..... 10
6. SUBDOMAINS..... 11
7. REST SERVICES AND M2MDATA ARDUINO LIBRARY..... 12
8. MQTT Protocol..... 15
9. API REST JSON..... 17

1. INTRODUCTION

This platform is aimed at enthusiasts who have a basic knowledge of Internet of Things (IoT) devices. M2mlight.com is a free basic platform service for IoT devices. You can store your cameras, sensors, actuators and alerts data in the cloud. Then, using your desktop or smartphone via the Internet, you can see the videos, graphs, maps and statistics of these devices.

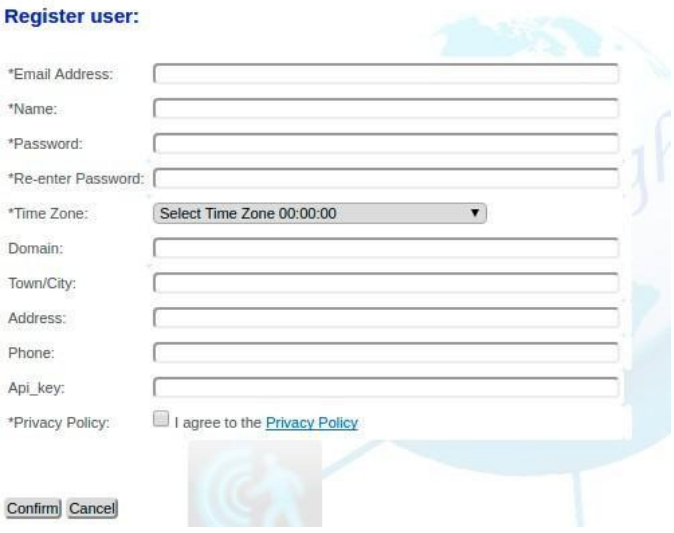
Cameras, sensors and actuators can receive and send data using a microcontroller like Arduino, Raspberry Pi or Esp8266; and then, through a gateway or router can send this information to the m2mlight platform in the cloud.

HTTP and RTSP protocols are used for cameras; and, sensors, actuators and alarms use the HTTP and MQTT protocols

Examples of using this platform you can see in this link caletagreen.com.

2. REGISTER USER

In order to use this platform you need to register as a user using the option **Login** and then **Register User.**



A user account has the following main fields:

- **Email Address:** required. Your Email Address will be your user identifier in m2mlight.com. All messages from cameras, sensors, actuators and alerts will be sent to this address.

- **Name:** required. Your user name.

- **Password:** required. Your password of at least 7 characters

- **Time Zone:** required. Pick your Time Zone. It is important to stored your data with the appropriate time.

- **Domain:** optional. The domain of your home or small business. For example: caletagreen.com. This field is not required but it is necessary if you want to use the subdomains option.

- **Town/City, Address and Phone:** optional.

- **Api_key:** informative. This api_key is an unique identifier of the user.

- **Privacy Policy:** required. You have to indicate that you agree to the Privacy Policy of this website. You can read this policy in the indicated link.

If you want to change your account information, including your password, you have to use the **Account**
option.

3. SENSORS

A sensor is a device that detects changes in the ambient conditions and transmits this information. For example, there are sensors for temperature, light, humidity, motion, electric current, power consumption, etc.

The SENSORS option allows you to manage your sensors. You can add, delete and edit a sensor. With this option you define sensors that have periodically values: singles values or location values (longitude and latitude). In this platform, coordinates values obtained from GPS devices are considered sensors.

EV1_STATE	Value	Automatic	state	Disable	0	0		0	8nTd1njcn4	Start Stop Edit Delete Graph Data
EV2_STATE	Single Value	Automatic	state	Disable	0	0		0	Eajh1njcn5	Start Stop Edit Delete Graph Data
Humedad del suelo 1	Single Value	Automatic	Porcentaje	Disable		30	Jm4Y1njcib BtXI1njcic	0	YKsQ1njcji	Start Stop Edit Delete Graph Data
S1_oficina_pruebas	Single Value	Automatic	HR	Disable	0	60	6q7H1njcjw QY0e1njcjx	0	zunH1njcmr	Start Stop Edit Delete Graph Data
S2_oficina_pruebas	Single Value	Automatic	HR	Disable	0	60	mu5B1njcjz 7yll1njck0	0	jKcY1njcn2	Start Stop Edit Delete Graph Data
Sensor_viento_oficina_pruebas	Single Value	Automatic	m/s	Disable	0	5		1	4Soa1njcn1	Start Stop Edit Delete Graph Data
Viento	Single Value	Automatic	m/seg	Disable		15	Jm4Y1njcib BtXI1njcic	0	mzR11njcjj	Start Stop Edit Delete Graph Data

Add new sensor:

*Name:
Sensor type: Single value
Control type: Manual
*Measure:
Email messages: Disable
Interval(sec):
Set Point Value:
Action Under Set Point:
Action Over Set Point:
Sample Time(min): 0

Add Cancel

For a sensor, you can upload data and see its status. For example, you can see if the sensor's values are out of the defined Set Point Value.

In the second part of this dialog, you can add or register a new sensor. The maximum number of sensors you can register is 10.

A **sensor** has the following fields:

- **Name:** required. The sensor name. You can not have two sensors with the same name.

- **Sensor type:** you have two alternatives, Single value (default) or Location. If you choose Location the measure field is always Degrees (longitude and latitude).

- **Control type:** you have two alternatives, Manual (default) or Automatic. If you choose Automatic will exist a synchronization with a remote controller. MQTT messages will be sent to the controller and the system will wait for a positive response.

- **Measure:** required. A standard unit used to express the size, amount, or degree of something. For example: Watts.

- **Email-messages:** you can choose to enable or disable. Enable, if you want to receive email messages when the sensor is not registering values or values reach the Set Point. Disable, if you do not receive email messages.

- **Interval (sec):** optional. The interval in seconds of two consecutive sensor values.

- **Set Point Value:** optional. The sensor value that controls an actuator.

- **Action Under Set Point:** optional. The actuator api_key that is triggered if sensor values are under the set point value.

- **Action Over Set Point:** optional. The actuator api_key that is triggered if sensor values are over the set point value.

- **Sample Time (min):** optional, default is 0. The time to obtain an average of a sample of values. It is used when the sensor values are widely dispersed over time.

In the first part of this dialog, you will find the list of your sensors. Using the **Edit** button you can update the sensor properties. With the **Delete** button you remove a sensor and its register values; you need to confirm this previously. The **Stop** button sets the disabled status and the **Start** button sets the enable status. If you have defined an Automatic option, all these actions will be executed if there is a positive response from the controller.

The **Status** of a sensor can be:

Sensor is enabled and it is registering values Sensor is disable

Sensor is not registering values

Sensor values have reached the Set Point

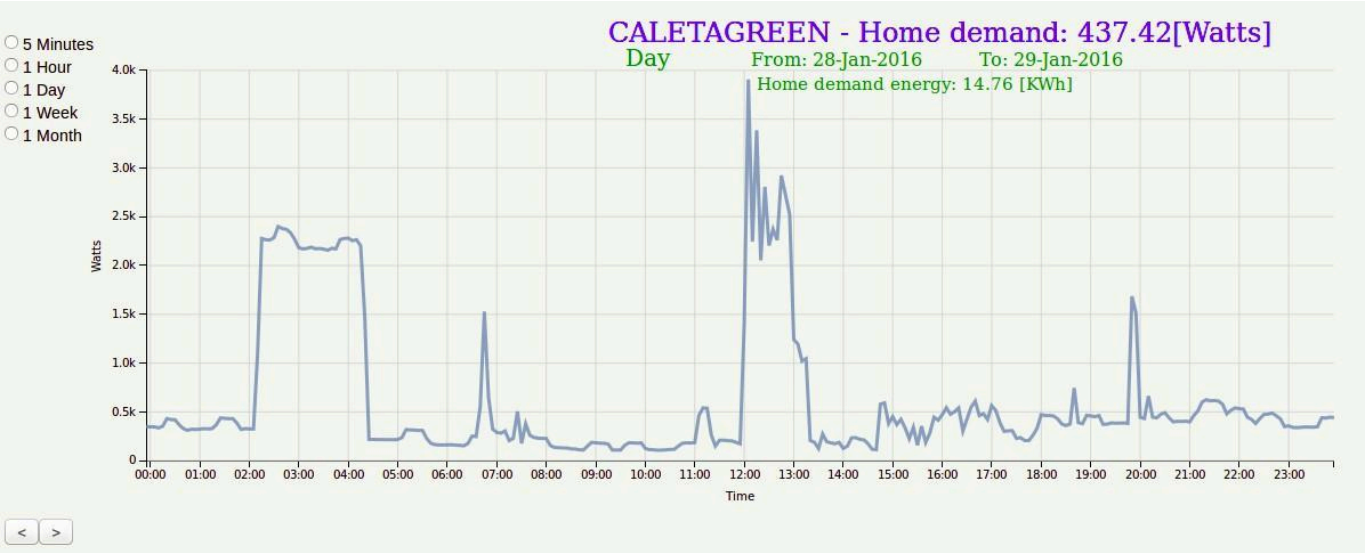
Daemons process periodically check each sensor and set this status. If a sensor has not registered a value in the cloud in the last 10 minutes, the system changes its state and sends an email message to the user if Email-messages property is enabled.

For a sensor, the minimum interval between two inserted values is 10 seconds. Namely**,** if you try to insert a value in a range of less than 10 seconds, the value will not be inserted.

Also, for a sensor, the minimum interval between two emails sent is 3,600 seconds (one hour) **.** Namely**,** if there is continuous news, you will receive a separate email for at least one hour.

Api_key is the unique identifier of the sensor and you can not change this value. You can add sensor values using this api_key and the Server functions or M2MData library.

The **Graphics** button allows you to see line graphs of this data, graphs by 5 minutes, an hour, a day, a week or a month. These graphics work with most popular browsers: Chrome and Firefox.



Using the **Data** button you can display and download stored data.

User: CALETAGREEN

Sensor Name: SolarHeater - Tank temperature

Show the most recent 40 **sensor values** Go!

Show all sensor values between 2016-09-19 18:42:11 **and** 2016-09-19 18:42:11 Go!

Download

Date	Value
2016-09-19 18:41:56	36.4
2016-09-19 18:41:15	36.5
2016-09-19 18:40:58	36.5
2016-09-19 18:40:42	36.5
2016-09-19 18:40:15	36.5
2016-09-19 18:39:57	36.5
2016-09-19 18:39:41	36.5
2016-09-19 18:39:21	36.5
2016-09-19 18:39:04	36.5
2016-09-19 18:38:34	36.4
2016-09-19 18:38:01	36.6
2016-09-19 18:36:46	36.5
2016-09-19 18:36:00	36.5

In the case of location sensors, with the **Map** button, you can see a Google Map with the location.

With the **Live Sensor Panel** button, you can see all sensor values on line in one screen. The screen is refreshed every 5 seconds.

Live Sensor Panel

Api_key	Name	Status	Measure	Last value	At time
5EUmWZfnB	MainPanel - Cooktop power consumption	Enable	Watts	2.89	2016-10-03 12:21:46
10OjcBi0AQ	MainPanel - Current phase 1 demand	Enable	Amps	0.48	2016-10-03 12:21:16
11ftawqPLz	MainPanel - Current phase 2 demand	Enable	Amps	1.40	2016-10-03 12:21:20
9vnsWkQpz	MainPanel - Home demand	Enable	Watts	234.17	2016-10-03 12:21:23
8YszrStin	MainPanel - Voltage 210V	Enable	Volts	217.02	2016-10-03 12:21:42
45dKXLsknB	OnGrid - AC Power microinverter 1	Enable	Watts	3295.72	2016-10-03 12:21:46
46CcZds67D	OnGrid - AC Power microinverter 2	Enable	Watts	95.29	2016-10-03 12:21:08
55nvHUWjwD	OnGrid - Generated power	Enable	Watts	113.70	2016-10-03 12:21:25
42HPnFgkYI	OnGrid - IAC microinverter 1	Enable	Amps	27.41	2016-10-03 12:21:41
435xKlbSoB	OnGrid - IAC microinverter 2	Enable	Amps	0.65	2016-10-03 12:21:44
4DT75t9zRW	OnGrid - IDC Battery	Enable	Amps	-0.44	2016-10-03 12:21:10
36ng6PeAqR	OnGrid - IDC solar panels	Enable	Amps	8.68	2016-10-03 12:21:22
37PTaQH5uh	OnGrid - IDC wind turbine	Enable	Amps	-0.79	2016-10-03 12:21:24
416Xq3JLfU	OnGrid - VAC microinverters	Enable	Volts	120.35	2016-10-03 12:21:13
4402vFnLa3	OnGrid - VDC Battery	Enable	Volts	13.91	2016-10-03 12:21:15
3915CHZzVL	OnGrid - VDC solar panels	Enable	Volts	13.11	2016-10-03 12:21:17
38QO6xdDTZ	OnGrid - VDC wind turbine	Enable	Volts	12.89	2016-10-03 12:21:20
4nGTZiA0H	SolarHeater - Heater power consumption	Enable	Watts	0.89	2016-10-03 12:20:18
1FEUqLO6i	SolarHeater - Tank temperature	Enable	Celsius deg	45.28	2016-10-03 12:20:40

If you do not want to see a sensor in this screen you can change the status to disable with the Stop button.

4. ACTUATORS

An actuator is a component that is responsible for controlling a mechanism or system. For example, turning on/off lights, turning on/off a sound alarm, opening or closing a door, etc.

The ACTUATORS option allows you to manage your actuators. You can add, delete and edit an actuator. With this option you can do a remote control (manual or automatic) of an actuator. Also, you can display and download registered actions of an actuator. In the second part of this dialog, you can add or register a new actuator**.**

An **actuator** has the following fields:

- **Name:** required. The actuator name. You can not have two actuators with the same name.
- **Email messages:** you can choose to enable or disable. Enable, if you want to receive email messages when an action occurs. Disable, if you do not want to receive email messages.
- **Control Type:** you can choose Manual or Automatic. If you want to have automatic remote control you have to choose the Automatic option.
- **Ip or domain:** optional. Ip address or domain of controller device. For example, the ip address of an arduino with ethernet shield.
- **Port:** optional. The port of the controller device.
- **Action:** it can be ON or OFF. This action name will be sent in the message to the

controller device.

- **Start HH:MM:** optional. The start time (HH:MM) for the action. It is the time to execute the action daily.
- **Week-day:** enable or disable for each day. Enable means that the action must be executed on this day. This property allows you to choose the days of the week when action will be executed.

In the first part of this dialog, you will find the list of your actuators. Using the **Edit** button you can update the actuator properties. With the **Delete** button you remove an actuator and its register values; you need to confirm this previously. The **Stop** button sets the disabled status and the **Start** button sets the enable status.

The **Status** of an actuator can be:

- ✅ Actuator is enable
- ❌ Actuator is disable

For an actuator, the minimum interval between two emails sent due to an actuator action is 10 seconds.

Api_key is the unique identifier of the actuator and you can not change this value. You can manage actuators using this api_key and the Server functions or M2MData library.

The **Do Now** button executes immediately an actuator action at defined start HH:MM. It sends a message (http call) to the controller device. This message is displayed on the screen and registered as an actuator manual action.

For a sensor, the **Data** button allows to see the executed actuators actions.

5. ALERTS

An alert is a message sent by e-mail that gives you useful information. You can define an alert when a certain event occurs. For example: sensors of motion and occupancy, proximity sensors, contact sensors (open door or window), etc.

The ALERTS option allows you to manage your alerts. You can add, delete and edit an alert. Also, you can display and download history stored data.

In the second part of this dialog, you can add or register a new alert**.** An **alert** has the following

fields:

- **Name:** required. The sensor name. You can not have two sensors with the same name.
- **Message:** required. The message to be sent.
- **E-mail interval:** interval between messages in seconds (default is 3,600 seconds that is equal to an hour). The minimum is 60 seconds.

In the first part of this dialog, you will find the list of your alerts. Using the **Edit** button you can update the alert properties. With the **Delete** button you remove an alert and its register values; you need to confirm this previously. The **Stop** button allows to disable an alert and the **Start** button allows it to enable it. To send an e-mail the status must be enabled.

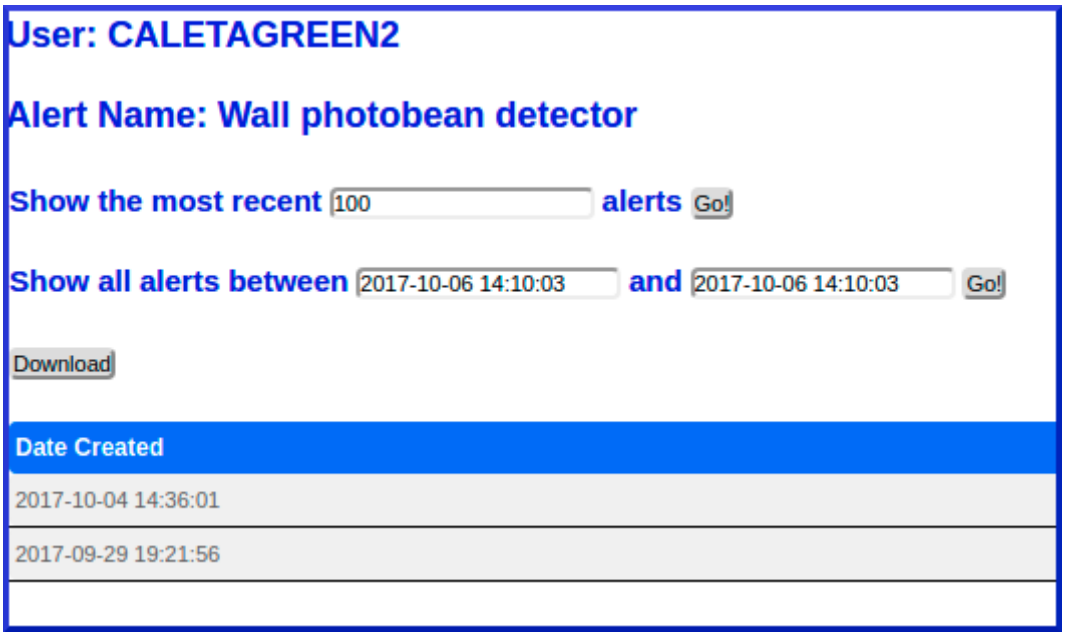
The **Status** of an alert can be:

- ✅Alert is enable
- ❌Alert is disable

Api_key is the unique identifier of the alert and you can not change this value.

You can add alerts values using this api_key and the Server functions or M2MData library.

Using the **Data** button you can display and download stored data alerts sent.



6. SUBDOMAINS

M2Mlight platform offers a DNS (Domain Name Server) service if you have a domain name. You can use this option if you need an internal DNS at your home.

First, register your domain name in your m2mlight account. Next register here your subdomains and set in your main router the DNS equal to pdns.caletagreen.com.

A subdomain is a domain that is a part of a main domain. For example, the subdomain gardencontrol.caletegreen.com is part of the domain caletagreen.com. It is useful when you need to give a name to an internal IP address and then use this name internally or externally together with a port.

This local DNS is not related to the other options of cameras, sensors, alerts, etc. That is, you can use this Subdomains option regardless of the other options.

With the second part of this dialog, you can add or register a new sub-domain. First, you need to register a domain using the Register User Dialog.

A sub-domain has the following fields:

- **Name:** the sub-domain name. You can not have two sub-domains with the same name.
- **Ip address:** the internal Ip address

In the first part of this dialog, you will find the list of your sub-domains. Using the **Edit** button you can update the sub-domains properties. With the **Delete** button you remove a sub-domain.

7. REST SERVICES AND M2MDATA ARDUINO LIBRARY

REST services (server functions) , MQTT protocols and the M2M Arduino library allow the use of the M2Mlight platform. First, you need to register the sensor, actuator or alert and then obtain the api_key in m2mlight.com.

You can find examples of this use in https://github.com/m2mlight/m2mData and projects located on www.caletagren.com.

These REST services and Arduino library can be used to send and receive values between the m2mlight platform and a program in a microcontroller or computer (arduino, raspberry, esp8266, etc.) or in a mobile device (tablet or smartphone).

The communication with the m2mlight platform can be using Ethernet (Arduino Ethernet or ENC28J60 shields), GSM (SIM800L module) or any device that connects to the Internet.

You can download m2mData library and see more information in: https://github.com/m2mlight/m2mData

7.1. REST services

To upload/download data to m2mlight platform using http from a browser or program, you can employ the following REST services:

a) Sensors:

-> Name: **send_sensor_value**

Description: upload (save) a single sensor "value" of a sensor identified by "api_key"

Call example:
https://m2mlight.com/iot/send_sensor_value?api_key=250hhCIHeO&value=200

Return: 0 if a value is successfully inserted, -1 for invalid api_key or disabled status, -2 if the interval between two values is less than 10 seconds.

-> Name: **send_sensor_location**

Description: upload (save) a location sensor "latitude" and "longitude of a sensor identified by "api_key"

Call example:
https://m2mlight.com/iot/send_sensor_location?api_key=26hyCIHeO&latitude=-0.158382&longitude=-78.45534

Return: 0 if values are successfully inserted, -1 for invalid api_key or disabled status, -2 if the interval between two values is less than 10 seconds.

-> Name: **read_sensor_value**

Description: return the last stored value of a single sensor identified by "api_key"

Call example: https://m2mlight.com/iot/read_sensor_value?api_key=300uhxXsAH

b) Actuators:

-> Name: **send_email_action**

Description: send an email related to an actuator action identified by api_key. The minimum

interval between two emails is 10 seconds.

Call example: http://m2mlight.com/iot/send_email_action?api_key=LrkD1njci8

Return: 0 if the mailing is successful

-> Name: **read_actuator_hour**

Description: return the start hour (HH:MM) of an actuator identified by "api_key"

Call example: http://m2mlight.com/iot/read_actuator_hour?api_key=180SgN4pHk

-> Name: **update_actuator_hour**

Description: update the start hour "start_hour" of the actuator identified by "api_key"

Call example:

http://m2mlight.com/iot/update_actuator_hour?api_key=a6ruhxX&start_hour=10:30

Return: -1 if HH:MM format is not correct

c) Alerts

-> Name: **send_email_alert**

Description: send an email related to an alert identified by api_key. The minimum interval between two emails is the email_time_interval property

Call example: http://m2mlight.com/iot/send_email_alert?api_key=6trkDnjci8

Return: 0 if the mailing is successful

You can use these functions inside an C++ code (Arduino or Esp8266) or Python code (Raspberry). If you are using an Arduino it is better to use the M2MData library.

If you are using a GSM/GPRS chip like SIM800L then you can connect to m2mlight.com directly with http commands or mqtt protocol. See example_use_sim800.ino.

7.2. M2MData Arduino library

The m2mData library is an Arduino library that allows you to store and retrieve data of sensors, actuators and alerts to/from m2mlight. You can use the http or mqtt protocol.

You can download this library from this link: https://github.com/m2mlight/m2mData

You need to create a m2mData folder in your Arduino Libraries and download the files: m2mData.h and m2mData.cpp

When Arduino is connected to Internet, you have the following functions:

a) Sensors

// Store a single "value" of a sensor identified with "api_key" using the http protocol void sendValue(char api_key[], float value);

// Store coordinates values of a location sensor identified with "api_key" void sendCoordinates(char api_key[], float latitude, float longitude);

// Return the last single value of a sensor identified with "api_key" float readValue(char api_key[]);

b) Actuators

// Send an email about an action of an actuator identified with "api_key" void sendEmailAction(char api_key[]);

// Return the star hour (HH:MM) of an actuator identified with "api_key" String readActuatorHour(char api_key[]);

// Update the start hour "start_hour" (HH:MM) of an actuator identified with "api_key" void updateActuatorHour(char api_key[], char start_hour[]);

c) Alerts

// Send an email of an alert identified with "api_key" void sendEmailAlert(char api_key[]);

8. MQTT Protocol

If you are connected to Internet, you can use mqtt messages to communicate to m2mlight.com

m2mlight.com is a mqtt broker, as such it receives and sends mqtt messages. If you choose Automatic in the control type option in your sensor and actuators, m2mlight sends mqtt messages in order for a controller to define its behavior with sensors, actuators and alerts.

A controller is a device that has a microcontroller (ATMEGA328, ESP8266, STM..,etc) and some sensors and actuators connected at its pines.

8.1 MQTT Messages from m2mlight.com to controller

a) To update information for a **sensor**:

- Topic: /user_api_key → Ex.: /3EI81njci3

- Payload: sensXXXX&o=I&a=YYY&s=Y&t=Y&e=YYY&u=YYY&v=YYY Where:
- XXXX is a aleatory number created by m2mlight.com (message id)
- Operation: o= I(insert) o U(Update) o D(Delete)
- Sensor api_key: a (Ex. a= jKcY1njcn2)
- Status: s (0=disable, other value=enable)
- Sample time: t (Ex, t=120)
- Set point: e (Ex. e=5)
- Action Under Set Point: u (Ex. u=6q7H1njcjw)
- Action Over Set Point: v (Ex. v=QY0e1njcjx)

b) To update information for an **actuator**:

- Topic: /user_api_key → Ex.: /3EI81njci3
- Payload: actuXXXX&o=Y&a=YYY&s=Y&c=Y&m=Y Where:
- XXXX is a aleatory number created by m2mlight.com (message id)
- Operation: o= I(insert) o U(Update) o D(Delete)
- Actuator api_key: a (Ex. a= mu5B1njcjz)
- Status: s (0=disable, other value=enable)
- Action: a (0=OFF, 1=ON)
- Control Type: m (0=Automatic, 1=Manual)

c) To execute an action immediately for an **actuator**:

- Topic: /user_api_key → Ex.: /3EI81njci3
- Payload: dnow&akey=yuWQ1njcig&actnow=ON Where:
- Actuator api_key: akey (Ex. akey= mu5B1njcjz)
- Action: actnow (Ex. actnow=ON)

8.2. MQTT Messages from controller to m2mlight.com

a) To load a “value” of a sensor with sensor_api_key of a user identified by user_api_key:

- Topic: /user_api_key/sensor → Ex.: /3EI81njci3/sensor
- Payload: sensor_api_key&value → Ex.: jKcY1njcn2&12.34

b) To send an “alert” of alert_api_key of a user identified by user_api_key:

- Topic: /user_api_key/alert → Ex.: /3EI81njci3/alert
- Payload: alert_api_key → Ex.: jqma1njcj9

c) To send a reply for an update of information about sensor or actuator:

- Topic: /user_api_key/reply → Ex.: /3EI81njci3/reply
- Payload: sensXXXX&OK or sensXXXX&KO or actuXXXX&OK or actuXXXX&KO

Where XXXX is a aleatory number created by m2mlight.com (message id)

8.3. MQTT protocol in M2Mlibrary

In M2Mlibrary you have this function:

// Store a single "value" of a sensor identified with "api_key" using the mqtt protocol void sendValueMqtt(char api_key[], float value);

9. API REST JSON

This API complies with the OpenAPI standard.

a) **Definition**

Description: API for recording real-time sensor readings.

URL Base: https://m2mlight.com

b) Authentication

Method: API Key.

Link: Header HTTP.

Header: X-API-Key: 250hhCIHeO

c) Documentation for readings

The data will be sent as follows:

Endpoint: /sensores/lecturas

Method: POST

Data format: JSON

Payload:

Field	Type	Required	Description
Value	Numeric	Yes	The value measured by the sensor (e.g., 20.5)
Unit	String	No	The unit of measurement (in "celsius", "%").

d) Output codes

201 Created: The reading was saved correctly in the database.

400 Bad Request: Required data is missing or the JSON is malformed

401 Unauthorized: The API key is invalid or was not sent.

405 Method Not Allowed: An attempt was made to use a method other than POST.

e) Practical Example

Example with cURL:

```
curl -X POST https://m2mlight.com \
  -H "Content-Type: application/json" \
  -H "X-API-Key: 250hhCIHeO" \
  -d '{"valor": 25.4, "unidad": "celsius"}'
```