Date : 2023.7.19

3D LiDAR
YLM-X001

COMMUNICATION SPECIFICATIONS FOR SENSOR SETTING

Symbol	Amended Reason			Pages	Date	Corrector	Amendment No.
Approved by	Checked by	Drawn by	Designed by	Title	YLM series Communication Specifications for sensor setting		
KAMITANI	HOSODA	ARIMORI	ARIMORI	Drawing No.	C-42-04550		1/27

Index

1. Overview

This is the communication specifications for YLM series sensor setting. The communication protocols described in this document are for sensor setting and differ from the communication protocols for point cloud data transmission. For the communication protocols for point cloud data acquisition, refer to the communication specifications for data acquisition.

1.1 Caution when disconnecting power

THIS SENSOR MAY BE DAMAGED IF THE POWER IS SWITCHED OFF DURING SCANNING. STOP THE POWER SUPPLY WITH /disable AND THEN TURN OFF THE POWER.

1.2 Virtualized sensor

Up to 8 FOV (Field of view) per unit can be set in this sensor. Different settings can be applied to each FOV and the measurement data can be transmitted by each FOV so that 1 unit can be treated as multiple sensors. Each of these virtual multiple sensors is called a virtualized sensor.

2. Communication

2.1 Communication interface

The communication interface of this sensor is Ethernet.

●Ethernet 1000BASE-TX

Communicated by HTTP and via REST API described in 4. REST API. The default setting of network address is as follows;

IP address: 192.168.0.10
Subnet mask: 255.255.255.0
Base port number: 10940
*The port number of the virtualized sensor is incremented by 1 from the base port number. 10940, 10941, …10947)

3. Parameter for sensor setting

This chapter describes configurable parameters. Each parameter can be altogether set with /scan_parameters described in 4. REST API. Endpoints for individual setting are also available for some parameters. The list of configurable parameters as follows.

Table 1

Parameter	Endpoint	Minimum value	Maximum value	Description
angle_range	/angle_range	-45[°]	45[°]	Set the minimum value/maximum value of FOV. The value can be set in one degree increments.
fps_multiple	/fps_multiple	1	31	Set the frame rate ratio for multiple FOVs. Can be set in one degree increments.
binning	/binning	1	4	Set the binning of the light-receiving pixels. The following settings are available. 1: without binning (low frame rate) 2: 2×2 binning 4: 4×4 binning
nn_level	/nn_level	0	5	Set NN filter (nearest neighbor filter). Noise can be excluded by activating the filter, but the resolution will be reduced. The following settings are available. 0: No filtering. 1: Requires 3 neighbors in 3×3 pixel window. 2: Requires 5 neighbors in 5×5 pixel window. 3: Requires 5 neighbors in 6×6 pixel window. 4: Requires 7 neighbors in 7×7 pixel window. 5: Requires 11 neighbors in 9×9 pixel window
inte_time_index	/inte_time_index	0	2	Set the accumulation time of the reflected light on the pixel for each direction of laser irradiation. 0: 10μs 1: 15μs 2: 20μs
snr_threshold	/snr_threshold	0	511.87	Set the SNR threshold value. Pixels with an SNR lower than this value are assumed to be pixels that could not be measured correctly and the distance is set to 0.
power_index	/power_index	0	2	Set the output laser power. The maximum output is 100%. 0: 50% 1: 75% 2: 100%
max_range_index	/max_range_index	0	1	Set the measurable maximum distance. 0: 25.2m 1: 32.4m

user_tag	-	0	4095	Identifier attached to the Type D header. When changing a parameter during scanning, this can be used to determine whether the received measurement data is the data after the parameter change. Refer to the communication specifications for data acquisition for Type D header.
frame_average	/frame_average	0	31	Set the number of moving averaging process in time to reduce noise. Frames are still emitted over the network at the nominal frame rate but each is an average of the previous N frames. A value of 0 and 1 both results in no frame averaging.
interleave	/interleave	false	true	Set the scanning order: sequential or interleaved in time. false: sequential true: interleaved

4. REST API

This chapter describes each endpoint of REST API. The Data is transmitted/received with json format. The commands available for all endpoints are GET and POST only. Sending commands not listed in the endpoint usage examples will return HTTP 405 error. Besides, setting parameters outside the range will return HTTP 422 error.

4.1 /state

GET /state

Get the current state of the sensor. Possible return values are "READY", "ENERGIZED", and "SCANNING".

Example: Get the current state of sensor.

Transmission)

```
GET /state HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "state": "ENERGIZED"
}
```

4.2 /start_scan

POST /start scan

Start scan. This operation succeeds only if the system has been calibrated and is in “ENERGIZED” state. In case of failure, HTTP 555 error is returned. Failure also occurs if the configured scanning pattern exceeds the configurable number of entries. For more information on the number of entries, see 4.6.1 Limitation.

Example: Start scan.

Transmission)

```
POST /start_scan HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.3 /stop_scan

POST /stop_scan

Stop scan. This operation succeeds only in “ENERGIZED” or "scanning” state. In case of failure, HTTP 555 error is returned.

Example: Stop scan.
Transmission)

```
POST /stop_scan HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.4 /scan_parameters
Endpoints for retrieving and setting parameters. See Table 1 Sensor parameters for parameter types.

GET /scan_parameters/opts
Get a list of sensor parameters and the configurable range.

GET /scan_parameters
Get the parameters set to the sensor.

POST /scan_parameters
Set parameters for the sensor. In case of using multiple virtualized sensors, increase the number of elements in the array for each parameter by the number of virtualized sensors used. In addition, all parameters must be set when adding a new virtualized sensor.

Example: Get a list of sensor parameters and the configurable range.
Transmission)

```
GET /scan_parameters/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
HOST: 192.168.0.10
Content-Type: application/json

{
    "angle_range":{
          "low": -45,
          "high": 45
     },
     "fps_multiple":{
          "low": 1,
          "high": 31
     },
     "power_index":{
          "options":[
               0,
               1,
               2
           ]
     },
     "inte_time_index":{
           "options":[
               0,
               1,
```

```
                2
            ]
        },
        "max_range_index":{
            "options":[
                0,
                1
            ]
        },
        "binning":{
            "options":[
                1,
                2,
                4
            ]
        },
        "frame_average":{
            "low":0,
            "high": 31
        },
        "interleave":{
            "options":[
                true,
                false
            ]
        },
        "snr_threshold":{
            "low": 0.0,
            "high": 511.87
        },
        "nn_level":{
            "options":[
                0,
                1,
                2,
                3,
                4,
                5
            ]
        },
        "user_tag":{
            "low": 0,
            "high": 4095
        }
}
```


Example: Get the parameters set to the sensor
Transmission)

```
GET /scan_parameters HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
HOST: 192.168.0.10
Content-Type: application/json

{
        "angle_range":[
```

```
            [
                -45,
                45
            ],
            [
                -10,
                10
            ]
        ],
        "fps_multiple":[
            1,
            2
        ],
        "power_index":[
            2,
            1
        ],
        "inte_time_index":[
            1,
            0
        ],
        "max_range_index":[
            0,
            1
        ],
        "user_tag":[
            10,
            20
        ],
        "binning":[
            2,
            4
        ],
        "frame_average":[
            1,
            2
        ],
        "nn_level":[
            0,
            1
        ],
        "snr_threshold":[
            1.25,
            2.25
        ],
        "interleave": false
}
```

Example: Set parameters for the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /scan_parameters HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

{
        "angle_range":[
```

```
                [
                    -45,
                    45
                ],
                [
                    -10,
                    10
                ]
            ],
            "user_tag":[
                10,
                20
            ],
            "snr_threshold":[
                1.25,
                1.44
            ],
            "nn_level":[
                0,1
            ],
            "power_index":[
                2,
                1
            ],
            "inte_time_index":[
                1,
                0
            ],
            "binning":[
                2,
                4
            ],
            "frame_average":[
                1,
                2
            ],
            "max_range_index":[
                0,
                1
            ],
            "fps_multiple":[
                1,
                2
            ],
            "interleave": false
}
```


Response)


```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.5 /angle_range

Endpoints for retrieving and setting the measurement range of the sensor. Other parameters can be also set together in /scan _parameters.

GET /angle_range/opts

Get the configurable measurement range.

GET /angle_range

Get the current measurement range.

POST /angle_range

Set the measurement range to the sensor.

Example: Get the configurable measurement range.

Transmission)

```
GET /angle_range/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "low": -45,
        "high": 45
}
```

Example: Get the current measurement range.

Transmission)

```
GET /angle_range HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
HOST: 192.168.0.10
Content-Type: application/json

{
        "angle_range": [
            [
                -15,
                15
            ],
            [
                30,
                45
            ]
        ]
}
```

Example: Set the measurement range to the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /angle_range HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

[[-45,45],[-10,10]]
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.6 /fps_mupltiple

Endpoints for retrieving and setting the relative frame rate of virtualized sensors. For example, 1 is set for virtualized sensor 1 and 3 for virtualized sensor 2, the FOV set for virtualized sensor 2 is scanned three times for every one scanning of the FOV set for virtualized sensor 1. Other parameters can be also set together in /scan _parameters.

GET /fps_multiple/opts

Get the configurable range of relative frame rate.

GET /fps_multiple

Get the relative frame rate set to the sensor.

POST /fps_multiple

Set the relative frame rate for the sensor.

Example: Get the configurable range of relative frame rate.
Transmission)

```
GET /fps_multiple/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "low": 1,
        "high": 31
}
```

Example: Get the relative frame rate set to the sensor.
Transmission)

```
GET /fps_multiple HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{
        "fps_multiple": [
            1,
            1
        ]
}
```

Example: Set the relative frame rate for the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /fps_multiple HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

[1,2]
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.6.1 Limitation

The following limitations should be noted when setting the relative frame rate.
This sensor uses a scan table to manage the scanning angle. Up to 512 entries can be defined in this scan table.
The total number of entries in the scan table created from the relative frame rates and FOV of eight virtualized sensors is determined by the following formula.

$$N_{scan\_table} = \sum_{i=0}^{7} \text{(the number of entries correspoding to the measurement range)(fps\_multiple)}$$

Based on this formula, the total number of entries should not exceed 512. As measured in 1 degree increments, (the number of entries corresponding to the measurement range) is (set FOV +1). Below is a calculation example of the number of entries when setting different relative frame rates for 2 virtualized sensors.

Example)
Virtualized sensor 1 (FOV0)
FOV: -45 degrees ~45 degrees, fps_multiple: 1
Virtualized sensor 2 (FOV1)
FOV: -7 degrees ~7 degrees, fps_muptiple: 3

The number of entries = (90+1) × (14+1) × 3=136

The following figures describe 1 frame of measurement in setting above.

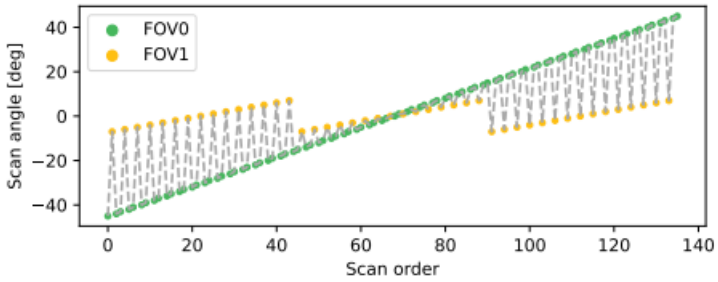

Figure 1 interleave = true

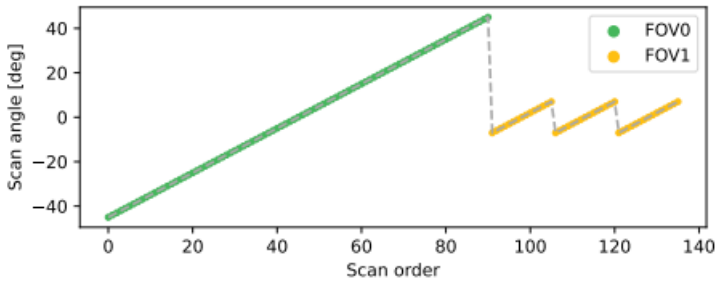

Figure 2 interleave =false

As per the relative frame rate setting, every time one measurement is taken in FOV0, three measurements are taken in FOV1. Figure 1 and 2 show the difference of scanning order by interleave setting. If interleave=true, FOV0 and FOV1 are alternately measured as shown in Figure 1. If Interleave= false, FOV1 is measured after FOV0 has been completely measured as seen in Figure 2.

4.7 /binning

Endpoints for retrieving and setting the binning level of the sensor. The smaller the value of binning level, the higher the resolution. Other parameters can be also set together in /scan_parameters.

GET /binning/opts

Get the configurable binning level of the sensor.

GET /binning

Get the binning level of the sensor.

POST /binning

Set the binning level of the sensor.

Example: Get the configurable binning level of the sensor.
Transmission)

```
GET /binning/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "options": [
            1,
            2,
            4
        ]
}
```

Example: Get the binning level of the sensor.
Transmission)

```
GET /binningHTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "binning": [
            2,
            2
        ]
}
```

Example: Set the binning level of the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /binningHTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

[1, 2]
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.8 /nn_level

Endpoints for retrieving and setting NN filter (nearest neighbor filter) level applying to the measurement data. Other parameters can be also set together in /scan_parameters.

GET /nn_level/opts

Get the configurable NN filter level.

GET /nn_level

Get the NN filter level set to the sensor.

POST /nn_level

Set the NN filter level for the sensor.

Example: Get the configurable NN filter level.

Transmission)

```
GET /nn_level/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "options":[
            0,
            1,
            2,
            3,
            4,
            5
        ]
}
```

Example: Get the NN filter level set to the sensor.

Transmission)

```
GET /nn_level HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "nn_level": [
            0,
            1
        ]
}
```

Example: Set the NN filter level for the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /nn_level HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

[1, 2]
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.9 /inte_time_index

Endpoints for retrieving and setting the time to integrate reflected light to pixels. Other parameters can be also set together in /scan_parameters.

GET /inte_time_index/opts
Get the configurable integration time.

GET /inte_time_index
Get the integration time set to the sensor.

POST /inte_time_index
Set the integration time to the sensor.

Example: Get the configurable integration time.
Transmission)

```
GET /inte_time_index/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "options": [
            0,
            1,
            2
        ]
}
```

Example: Get the integration time set to the sensor.
Transmission)

```
GET /inte_time_index HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1200 OK
Content-Type: application/json

{
        "inte_time_index": [
            0,
            1
        ]
}
```

Example) Set the integration time to the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /inte_time_index HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

[1, 0]
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.10 /snr_threshold

Endpoints for retrieving and setting the SNR (Signal to Noise Ratio) threshold. The distance value of a pixel that is below this threshold is 0. Other parameters can be also set together in /scan_parameters.

GET /snr_threshold/opts
Get the configurable range of SNR threshold.

GET /snr_threshold
Get the SNR threshold set to the sensor.

POST /snr_threshold
Set the SNR threshold to the sensor.

Example: Get the configurable range of SNR threshold.
Transmission)

```
GET /snr_threshold/opts HTTP/1.1
HOST:192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "low": 0.0,
        "high": 511.87
}
```

Example: Get the SNR threshold set to the sensor.
Transmission)

```
GET /snr_threshold HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "snr_threshold": [
            7.34,
            4.56
        ]
}
```

Example: Set the SNR threshold to the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /snr_threshold HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

[2.51, 3.42]
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.10.1 /power_index

Endpoints for retrieving and setting laser power. Other parameters can be also set together in /scan_parameters.

GET /power_index/opts

Get the configurable laser power.

GET /power_index

Get the laser power set to the sensor.

POST /power_index

Set the laser power to the sensor.

Example: Get the configurable laser power.
Transmission)

```
GET /pwoer_index/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "options": [
            0,
            1,
            2
        ]
}
```

Example: Get the laser power set to the sensor.
Transmission)

```
GET /power_index HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "power_index": [
            0,
            1
        ]
}
```

Example: Set the laser power to the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /power_index HTTP/1.1
HOST: 192.168.0.110
Content-Type: application/json

[1, 0]
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.11 /max_range_index

Endpoints for retrieving and setting the measurable maximum distance. Other parameters can be also set together in /scan_parameters.

GET /max_range_index/opts
Get the measurable maximum distance.

GET /max_range_index
Get the measurable maximum distance set to the sensor.

POST /max_range_index
Set the measurable maximum distance to the sensor.

Example: Get the configurable maximum measurement distance.
Transmission)

```
GET /max_range_index/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "options": [
            0,
            1
        ]
}
```

Example: Get the maximum measurement distance set to the sensor.
Transmission)

```
GET /max_range_index HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "max_range_index": [
            0,
            0
        ]
}
```

Example: Set the maximum measurement distance to the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /max_range_index HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

[1, 1]
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.12 /frame_average

Endpoints for setting the number of moving averaging process in time. Other parameters can be also set together in /scan_parameters.

GET /frame_average/opts
Get the configurable number of averaging.

GET /frame_average
Get the number of averaging set to the sensor.

POST /frame_average
Set the number of averaging to the sensor.

Example: Get the configurable number of averaging.
Transmission)

```
GET /frame_average/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "low": 0,
        "high": 31
}
```

Example: Get the number of averaging set to the sensor.
Transmission)

```
GET /frame_averageHTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "frame_average": [
            0,
            0
        ]
}
```

Example: Set the number of averaging to the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /frame_averageHTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

[1, 1]
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.13 /interleave

Endpoints for setting the scanning order. Measurements are sequentially taken between virtualized sensors in this parameter=true. However, measurements are taken in order from the virtual sensor with the lowest number if this parameter=false. Other parameters can be also set together in /scan_parameters.
*This setting is applied to all virtualized sensors.

GET /interleave/opts
Get the configurable parameter.

GET /interleave
Get the interleave setting of the sensor.

POST /interleave
Set interleave=true or false to the sensor.

Example: Get the configurable parameter.
Transmission)

```
GET /interleave/opts HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "options": [
            true,
            false
        ]
}
```

Example: Get the interleave setting of the sensor.
Transmission)

```
GET /interleave HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

{
        "interleave": false
}
```

Example: Set interleave=true or false to the sensor.
In this example, parameters are set to two virtualized sensors.
Transmission)

```
POST /interleave HTTP/1.1
HOST: 192.168.0.10
Content-Type: application/json

true
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"SUCCESS"
```

4.14 /restart

Restart the sensor to return to the default “ENERGIZED” state. This enables the power supply to be safely disconnected.

POST /restart

Example: Restart the sensor.
Transmission)

```
POST /restart HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"Success"
```

4.15 /disable

Stop scanning and disable power to the sensor.

POST /disable

Example: Disable power to the sensor.

Transmission)

```
POST /disable HTTP/1.1
HOST: 10.20.30.40
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

"Sensor head disabled and powered down."
```

4.16 /logs

Retrieve logs for debug.

GET /logs

Encrypted zip files containing system logs and diagnostic data can be downloaded.

```
GET /logs HTTP/1.1
HOST: 192.168.0.10
```

4.17 /messages

Endpoints for retrieving messages from API.

GET /messages

```
GET /messages HTTP/1.1
HOST: 192.168.0.10
```

Example: Get messages from API

Transmission)

```
GET /messages HTTP/1.1
HOST: 192.168.0.10
```

Response)

```
HTTP/1.1 200 OK
Content-Type: application/json

["System Bootup Complete"]
```

5. References

Title of document	Reference (Document number or URL)
YLM series Communication Specifications for data acquisition	C-42-04549