DEEPRobotics

Lynx M20 Series Software Development Manual(Beta)

V0.0.4-0 July 10, 2025

Document Description

Version	Date	Modification contents
0.0.1-0	2025/6/18	New document
0.0.4-0	2025/7/10	Synchronized with the Chinese version

※ The final interpretation rights belong to DEEP Robotics.

CONTENTS

1 Inspection Communication Protocol 5

1.1 Protocol Overview 5

1.1.1 Protocol Hierarchies..... 5

1.1.2 Protocol Port Number..... 5

1.1.3 Interaction Mechanism 5

1.1.4 Application Protocol Data Unit..... 6

1.1.5 Protocol Header Structure..... 6

1.1.6 ASDU Structure 7

1.2 ASDU Message Set (Control Type) 9

1.2.1 Heartbeat 9

1.2.2 Usage Mode..... 9

1.2.3 Motion State..... 10

1.2.4 Gait Switching..... 12

1.2.5 Motion Control(Axis Command)..... 13

1.2.6 Flashlight..... 14

1.3 ASDU Message Set (Response Type) 16

1.3.1 Obtain Real-time State 16

1.3.2 Obtain Abnormal Status..... 29

Appendix 1: UDP Sample Code 33

Appendix 2: Obtaining camera video stream 35

1 Inspection Communication Protocol

1.1 Protocol Overview

This protocol is based on the TCP or UDP (selectable according to specific development needs) and applies to the communication between the robot and the host computer (external board card or system).

1.1.1 Protocol Hierarchies

The layer of this protocol in OSI model, and the data structure of protocol stack, as shown in Table below:

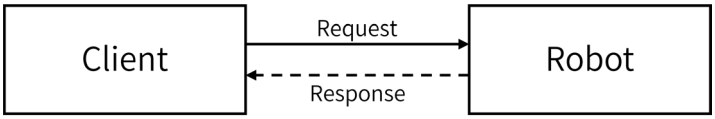
<table>
<tr><td>Inspection Protocol</td><td>Application layer (layer 7)</td></tr>
<tr><td rowspan="2">TCP/IP or UDP/IP protocol</td><td>Transport layer (layer 4)</td></tr>
<tr><td>Network layer (layer 3)</td></tr>
<tr><td rowspan="2">Ethernet</td><td>Link layer (layer 2)</td></tr>
<tr><td>Physical layer (layer 1)</td></tr>
<tr><td colspan="2">Notes: Layer 5 and layer 6 are not used</td></tr>
</table>

1.1.2 Protocol Port Number

When using this protocol, the robot is the TCP/UDP server, and the host computer (external board card or system) is the TCP/UDP client. The UDP server address and port number for the protocol is 10.21.31.103:30000, and the TCP server address and port number is 10.21.31.103:30001.

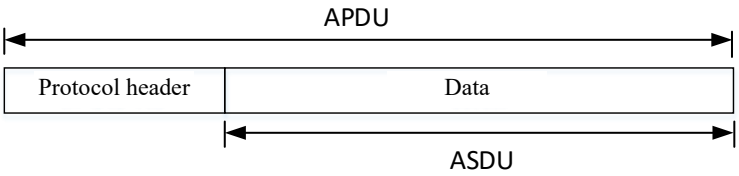
1.1.3 Interaction Mechanism

As shown in Figure below, this protocol uses the request/response mechanism. The host computer can actively issue the requests such as data query and control command to the robot, then robot responds to some of the requests (for details on the response requests, please refer to Section 1.3).



1.1.4 Application Protocol Data Unit

This protocol, APDU (Application Protocol Data Unit), adopts the structure of "Protocol Header + ASDU (Application Service Data Unit)", and each APDU can carry 1 ASDU.



1.1.5 Protocol Header Structure

The length of the protocol header is fixed to 16 bytes, as shown in Table below.

No.	Content	Length	Value	Remarks
1	Synchronization character	1	0xeb	Fixed
2	Synchronization character	1	0x91	Fixed
3	Synchronization character	1	0xeb	Fixed
4	Synchronization character	1	0x90	Fixed
5	Length	2		The length of the ASDU byte segment in APDU, which is in little endian, with the low bytes first. The maximum length of ASDU is

				limit to 65,535 bytes.
6	Message ID	2		The unique identifier of each frame of the message, used to identify the corresponding relationship between the request frame and the response frame. The value is controlled by the requester, and the response frame replies with the same value. It increments from 0, then starts at 0 again after reaching 65,535. It is in little endian, with the low bytes first.
7	ASDU Structure	1		ASDU data format type identification bit; When using XML format, the value is 0x00; When using JSON format, the value is 0x01.
8	Reserved	7	0x00	Reserve 7 bytes

1.1.6 ASDU Structure

The ASDU data content of this protocol is in JSON/XML format (Refer to Section 1.1.5; must be specified in the protocol header), containing the following general fields:

Field	Meaning
Type	Message type
Command	Message command code
Time	Message sending time (local time zone),

	with the format of: YYYY-MM-DD HH:MM:SS
Items	Parameters of the message

[Notes] It is recommended to use JSON format, which has faster processing performance and more comprehensive data structures for ASDU, and supports more message types.

There is a JSON ASDU case:

```
{
    "PatrolDevice": {
        "Type": 1002,
        "Command": 1,
        "Time": "2023-01-01 00:00:00",
        "Items": {
        }
    }
}
```

There is a XML ASDU case:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
    <Type>1002</Type>
    <Command>1</Command>
    <Time>2023-01-01 00:00:00</Time>
    <Items/>
</PatrolDevice>
```

1.2 ASDU Message Set (Control Type)

1.2.1 Heartbeat

You can send heartbeat commands to the robot by this request.

Type	Command	Message type
100	100	Heartbeat

[Notes] It is recommended to send this command at a frequency of no less than 1 Hz. The robot will report real-time status and abnormal status information to the IP and port that continuously sends heartbeat commands. For details, please refer to Section 1.3.

JSON request:

```
{
    "PatrolDevice": {
        "Type": 100,
        "Command": 100,
        "Time": "2023-01-01 00:00:00",
        "Items": {
        }
    }
}
```

XML request:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
    <Type>100</Type>
    <Command>100</Command>
    <Time>2023-01-01 00:00:00</Time>
    <Items/>
</PatrolDevice>
```

In this request message, the Items field does not contain any parameter item.

1.2.2 Usage Mode

You can switch the usage mode of the robot by this request and determine whether the operation was successful by referring to the response information in Section 1.3.

Type	Command	Message type
1101	5	Usage Mode Switching

JSON request:

```
{
    "PatrolDevice": {
        "Type": 1101,
        "Command": 5,
        "Time": "2023-01-01 00:00:00",
        "Items": {
            "Mode": 0
        }
    }
}
```

XML request:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
    <Type>1101</Type>
    <Command>5</Command>
    <Time>2023-01-01 00:00:00</Time>
    <Items>
        <Mode>0</Mode>
    </Items>
</PatrolDevice>
```

In this request message, the Items field contains the following parameters:

Parameters	Meaning	Type	Value
Mode	The usage mode of the robot	int	Regular Mode = 0 Navigation Mode = 1

[Notes] In regular mode, axis commands are supported (refer to Section 1.2.5). In navigation mode, navigation tasks are supported.

1.2.3 Motion State

You can switch the motion state information of the robot by this request and determine whether the operation was successful by referring to the response information in Section

1.3.

Type	Command	Message type
2	22	Motion State Switching

JSON request:

```
{
    "PatrolDevice": {
        "Type": 2,
        "Command": 22,
        "Time": "2023-01-01 00:00:00",
        "Items": {
            "MotionParam": 0
        }
    }
}
```

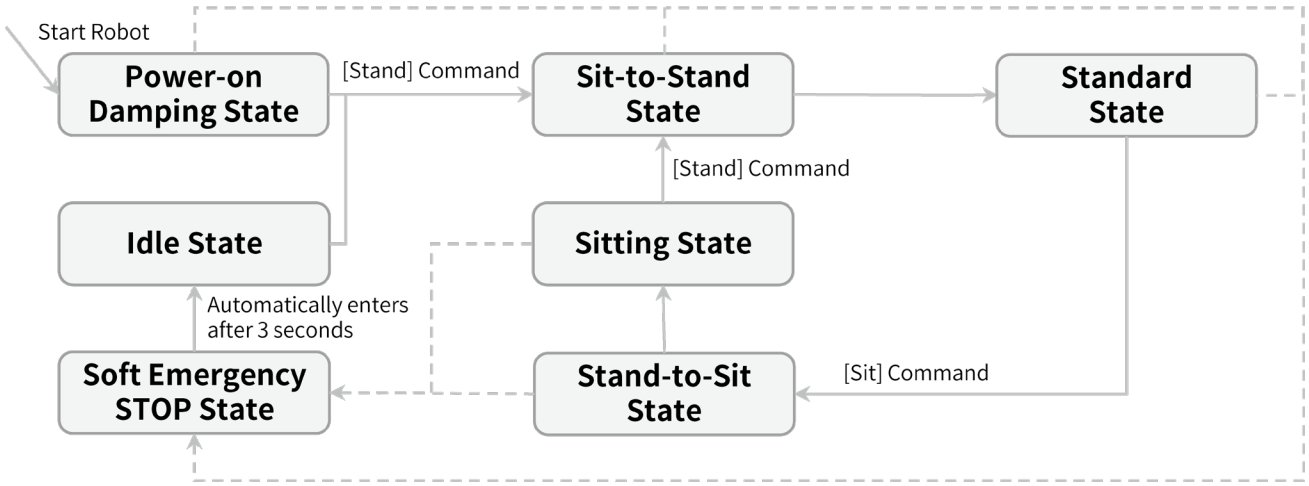
XML request:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
    <Type>2</Type>
    <Command>22</Command>
    <Time>2023-01-01 00:00:00</Time>
    <Items>
        <MotionParam>0</MotionParam>
    </Items>
</PatrolDevice>
```

In this request message, the Items field contains the following parameters:

Parameters	Meaning	Type	Value
MotionParam	Robot motion state[1]	int	Idle = 0 Stand = 1 Soft Emergency Stop = 2 Power-on Damping = 3 Sit = 4 Standard= 6

[1] The conversion relationship of the robot's motion state is shown in the figure below:



1.2.4 Gait Switching

You can switch the gait of the robot by this request and determine whether the operation was successful by referring to the response information in Section 1.3.

Type	Command	Message type
2	23	Gait Switching

JSON request:

```
{
    "PatrolDevice": {
        "Type": 2,
        "Command": 23,
        "Time": "2023-01-01 00:00:00",
        "Items": {
            "GaitParam": 0
        }
    }
}
```

XML request:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
    <Type>2</Type>
    <Command>23</Command>
    <Time>2023-01-01 00:00:00</Time>
    <Items>
        <GaitParam>0</MotionParam>
    </Items>
</PatrolDevice>
```

In this request message, the Items field contains the following parameters:

Parameters	Meaning	Type	Value
GaitParam	Robot gait	int	Basic = 1 Stair = 14

1.2.5 Motion Control(Axis Command)

You can control the motion of the robot by this request.

Type	Command	Message type
2	21	Motion Control

[Notes] This command is only supported in Regular Mode.

JSON request:

```
{
    "PatrolDevice": {
        "Type": 2,
        "Command": 21,
        "Time": "2023-01-01 00:00:00",
        "Items": {
            "X": 0.0,
            "Y": 0.0,
            "Z": 0.0,
            "Roll": 0.0,
            "Pitch": 0.0,
            "Yaw": 0.0
        }
    }
}
```

XML request:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
    <Type>2</Type>
    <Command>21</Command>
    <Time>2023-01-01 00:00:00</Time>
    <Items>
        <X>0.0</X>
        <Y>0.0</Y>
```

```
        <Z>0.0</Z>
        <Roll>0.0</Roll>
        <Pitch>0.0</Pitch>
        <Yaw>0.0</Yaw>
    </Items>
</PatrolDevice>
```

In this request message, the Items field contains the following parameters:

Parameters	Meaning	Type	Value
X	Forward and backward speed	float	[-1,1] (Values between [-1,1] represent the ratio of the current command speed to the maximum speed)
Y	Left and right movement speed	float	[-1,1] (Values between [-1,1] represent the ratio of the current command speed to the maximum speed)
Z	Vertical movement speed	float	[-1,1] (Values between [-1,1] represent the ratio of the current command speed to the maximum speed)
Roll	Roll angle	float	[-1,1] (Values between [-1,1] represent the ratio of the current command speed to the maximum speed)
Pitch	Pitch angle	float	[-1,1] (Values between [-1,1] represent the ratio of the current command speed to the maximum speed)
Yaw	Yaw angle	float	[-1,1] (Values between [-1,1] represent the ratio of the current command speed to the maximum speed)

[Notes] Recommended frequency for translation/rotation control: 20 Hz. Only X, Y, and Yaw are effective in Basic and Stair gaits.

1.2.6 Flashlight

You can turn the robot's front and rear lights on or off by this request and determine whether the operation was successful by referring to the response information in Section 1.3.

Type	Command	Message type
1101	2	Flashlight

JSON request:

```
{
    "PatrolDevice": {
        "Type": 1101,
        "Command": 2,
        "Time": "2023-01-01 00:00:00",
        "Items": {
            "Front": 0
```

```
            "Back": 0
        }
    }
}
```

XML request:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
    <Type>1101</Type>
    <Command>2</Command>
    <Time>2023-01-01 00:00:00</Time>
    <Items>
        <Front>0</Front>
        <Back>0</Back>
    </Items>
</PatrolDevice>
```

In this request message, the Items field contains the following parameters:

Parameters	Meaning	Type	Value
Front	Front flashlight of the robot	int	Light off= 0 Light on= 1
Back	Back flashlight of the robot		

1.3 ASDU Message Set (Response Type)

1.3.1 Obtain Real-time State

The robot will actively report status information to the IP address and port that sent the heartbeat command (refer to section 1.2.1).

1.3.1.1 Basic Status Report

This message type is actively reported at 2 Hz, you can obtain the current basic status of the robot through the response of this message type.

Type	Command	Message type
1002	6	Obtain Basic Status

JSON response:

```
{
    "PatrolDevice": {
        "Type": 1002,
        "Command": 6,
        "Time": "2023-01-01 00:00:00",
        "Items": {
            "BasicStatus": {
                "MotionState": 0,
                "Gait": 0,
                "Charge":0,
                "HES": 0,
                "ControlUsageMode": 0,
                "Direction": 0,
                "OOA": 0,
                "PowerManagement": 0,
                "Sleep": false,
                "Version": STD
            }
        }
    }
}
```

XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<PatrolDevice>
    <Type>1002</Type>
    <Command>6</Command>
    <Time>2023-01-01 00:00:01</Time>
    <Items>
        <BasicStatus>
            <MotionState>0</MotionState>
            <Gait>0</Gait>
            <Charge>0</Charge>
            <HES>0</HES>
            <ControlUsageMode>0</ControlUsageMode>
            <Direction>0</Direction>
            <OOA>0</OOA>
            <PowerManagement>0</PowerManagement>
            <Sleep>false</Sleep>
            <Version>STD</Version>
        </BasicStatus>
    </Items>
</PatrolDevice>
```

The Items field contains the following parameters:

Parameters	Meaning	Type	Value
MotionState	Robot motion state[1]	int	Idle = 0 Stand = 1 Soft Emergency Stop = 2 Power-on Damping = 3 Sitting = 4 Standard= 6
Gait	Robot gait	int	Basic = 1 Stair = 14
Charge	Robot charging status	int	Idle = 0 Enter charge dock = 1 Charging = 2 Exiting charge dock = 3 Robot error = 4 Robot is on the dock but not charged = 5
HES	Hard emergency stop status	int	Not triggered = 0

Parameters	Meaning	Type	Value
			triggered = 1
ControlUsageMode	Robot control usage mode	int	Regular mode = 0 Navigation mode = 1
Direction	The robot's forward direction[2]	int	Front = 0 Back = 1
Version	Device version	/	Lynx M20 = STD Lynx M20 Pro = PRO

[2] The robot's forward direction is defined as the movement direction when a positive X-axis velocity is commanded. If the forward direction is set to "Back", the side with the Hard Emergency Stop is considered forward direction.

1.3.1.2 Motion Control Status Report

This message type is actively reported at 10 Hz, you can obtain the current motion control status of the robot through the response of this message type.

Type	Command	Message type
1002	4	Obtain Motion Control Status

JSON response:

```
{
    "PatrolDevice": {
        "Type": 1002,
        "Command": 4,
        "Time": "2023-01-01 00:00:00",
        "Items": {
            "MotionStatus": {
                "Roll": 0.0,
                "Pitch": 0.0,
                "Yaw": 0.0,
                "OmegaZ": 0.0,
                "LinearX": 0.0,
                "LinearY": 0.0,
                "Height": 0.0,
                "Payload":0.0,
```

```
                "RemainMile":0.0
            },
            "MotorStatus": {
                "LeftFrontHipX": 0.0,
                "LeftFrontHipY": 0.0,
                "LeftFrontKnee": 0.0,
                "LeftFrontWheel": 0.0,
                "RightFrontHipX": 0.0,
                "RightFrontHipY": 0.0,
                "RightFrontKnee": 0.0,
                "RightFrontWheel": 0.0,
                "LeftBackHipX": 0.0,
                "LeftBackHipY": 0.0,
                "LeftBackKnee": 0.0,
                "LeftBackWheel": 0.0,
                "RightBackHipX": 0.0,
                "RightBackHipY": 0.0,
                "RightBackKnee": 0.0,
                "RightBackWheel": 0.0,
            },
        }
    }
}
```

XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
    <Type>1002</Type>
    <Command>4</Command>
    <Time>2023-01-01 00:00:01</Time>
    <Items>
        <MotionStatus>
            <Roll>0.0</Roll>
            <Pitch>0.0</Pitch>
            <Yaw>0.0</Yaw>
            <OmegaZ>0.0</OmegaZ>
            <LinearX>0.0</LinearX>
            <LinearY>0.0</LinearY>
            <Height>0.0</Height>
            <Payload>0.0</Payload>
            <RemainMile>0.0</RemainMile>
        </MotionStatus>
        <MotorStatus>
            <LeftFrontHipX>0.0</LeftFrontHipX>
```

```
            <LeftFrontHipY>0.0</LeftFrontHipY>
            <LeftFrontKnee>0.0</LeftFrontKnee>
            <LeftFrontWheel>0.0</LeftFrontWheel>
            <RightFrontHipX>0.0</RightFrontHipX>
            <RightFrontHipY>0.0</RightFrontHipY>
            <RightFrontKnee>0.0</RightFrontKnee>
            <RightFrontWheel>0.0</RightFrontWheel>
            <LeftBackHipX>0.0</LeftBackHipX>
            <LeftBackHipY>0.0</LeftBackHipY>
            <LeftBackKnee>0.0</LeftBackKnee>
            <LeftBackWheel>0.0</LeftBackWheel>
            <RightBackHipX>0.0</RightBackHipX>
            <RightBackHipY>0.0</RightBackHipY>
            <RightBackKnee>0.0</RightBackKnee>
            <RightBackWheel>0.0</RightBackWheel>
        </MotorStatus>
    </Items>
</PatrolDevice>
```

In this response message, the Items field contains the MotionStatus and MotorStatus parameter groups:

The MotionStatus parameter group provides feedback on the motion status of the robot:

Parameters	Meaning	Type
Roll/Pitch/Yaw	The posture angle of robot (rad)	float
OmegaZ	Z-axis angular velocity of robot (rad/s)	float
LinearX / LinearY	X-axis/Y-axis linear velocity of robot (m/s)	float
Height	The height of the robot (m)	float
Payload	Invalid parameter	/
RemainMile	Estimated remaining range (km)	float

The MotorStatus[3] parameter group provides feedback on the motion status of each joint of the robot:

Parameters	Meaning	Type
*HipX	Angle of hip joint for abduction and adduction (rad)	float
*HipY	Angle of hip joint for flexion and extension (rad)	float

Parameters	Meaning	Type
*Knee	Angle of knee joint (rad)	float
*Wheel	Speed of wheel joint (rad/s)	float

[3] The asterisk (*) to the left of a parameter item in the MotorStatus parameter group indicates the position of an omitted parameter name. The left side is the side where the battery bin is located. The back side is the side with the Hard Emergency STOP button. For example, LeftBackHipX refers to the left front HipX joint for abduction and adduction, and the parameter value indicates the angle of the joint. The names and meanings of other parameters follow the same principle.

1.3.1.3 Device State Report

This message type is actively reported at 2 Hz, you can obtain the current device state of the robot through the response of this message type.

Type	Command	Message type
1002	5	Obtain Device state

JSON response:

```
{
    "PatrolDevice": {
        "Type": 1002,
        "Command": 5,
        "Time": "2023-01-01 00:00:00",
        "Items": {
            "BatteryStatus": {
                "VoltageLeft": 0.0,
                "VoltageRight": 0.0,
                "BatteryLevelLeft": 0.0,
                "BatteryLevelRight": 0.0,
                "battery_temperatureLeft": 0.0,
                "battery_temperatureRight": 0.0,
                "chargeLeft":false,
                "chargeRight":false
```

```
            },
            "DeviceTemperature": {
                "LeftFrontHipXMotor": 0.0,
                "LeftFrontHipXDriver": 0.0,
                "LeftFrontHipYMotor": 0.0,
                "LeftFrontHipYDriver": 0.0,
                "LeftFrontKneeMotor": 0.0,
                "LeftFrontKneeDriver": 0.0,
                "LeftFrontWheelMotor": 0.0,
                "LeftFrontWheelDriver": 0.0,
                "RightFrontHipXMotor": 0.0,
                "RightFrontHipXDriver": 0.0,
                "RightFrontHipYMotor": 0.0,
                "RightFrontHipYDriver": 0.0,
                "RightFrontKneeMotor": 0.0,
                "RightFrontKneeDriver": 0.0,
                "RightFrontWheelMotor": 0.0,
                "RightFrontWheelDriver": 0.0,
                "LeftBackHipXMotor": 0.0,
                "LeftBackHipXDriver": 0.0,
                "LeftBackHipYMotor": 0.0,
                "LeftBackHipYDriver": 0.0,
                "LeftBackKneeMotor": 0.0,
                "LeftBackKneeDriver": 0.0,
                "LeftBackWheelMotor": 0.0,
                "LeftBackWheelDriver": 0.0,
                "RightBackHipXMotor": 0.0,
                "RightBackHipXDriver": 0.0,
                "RightBackHipYMotor": 0.0,
                "RightBackHipYDriver": 0.0,
                "RightBackKneeMotor": 0.0,
                "RightBackKneeDriver": 0.0,
                "RightBackWheelMotor": 0.0,
                "RightBackWheelDriver": 0.0
            },
            "Led": {
                "Fill": {
                    "Front":1,
                    "Back":1
                }
            },
            "GPS": {
                "Latitude":0.0,
                "Longitude":0.0,
```

```
                "Speed":0.0,
                "Course":0.0,
                "FixQuality":0.0,
                "NumSatellites":0,
                "Altitude":0.0,
                "HDOP":0.0,
                "VDOP":0,
                "PDOP":0.0,
                "VisibleSatellites":0
            },
            "DevEnable":{
                "FanSpeed":100,
                "LoadPower":1,
                "LedHost":1,
                "LedExt":1,
                "FP":1,
                "Lidar":{
                    "Front":1,
                    "Back":1
                },
                "GPS":1,
                "Video":{
                    "Front":1,
                    "Back":1
                }
            },
            "CPU":{
                "CPU103":{
                    "Temperature":0.0,
                    "FrequencyInt":0.0,
                    "FrequencyApp":0.0
                },
                "CPU105":{
                    "Temperature":0.0,
                    "FrequencyInt":0.0,
                    "FrequencyApp":0.0
                },
                "CPU106":{
                    "Temperature":0.0,
                    "FrequencyInt":0.0,
                    "FrequencyApp":0.0
                }
            }
        }
```

```
    }
}
```

XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
    <Type>1002</Type>
    <Command>5</Command>
    <Time>2023-01-01 00:00:01</Time>
    <Items>
        <BatteryStatus>
            <VoltageLeft>0.0</VoltageLeft>
            <VoltageRight>0.0</VoltageRight>
            <BatteryLevelLeft>0.0</BatteryLevelLeft>
            <BatteryLevelRight>0.0</BatteryLevelRight>
            <Battery_temperatureLeft>0.0</Battery_temperatureLeft>
            <Battery_temperatureRight>0.0</Battery_temperatureRight>
            <chargeLeft>false</chargeLeft>
            <chargeRight>false</chargeRight>
        </BatteryStatus>
        <DeviceTemperature>
            <LeftFrontHipXMotor>0.0</LeftFrontHipXMotor>
            <LeftFrontHipXDriver>0.0</LeftFrontHipXDriver>
            <LeftFrontHipYMotor>0.0</LeftFrontHipYMotor>
            <LeftFrontHipYDriver>0.0</LeftFrontHipYDriver>
            <LeftFrontKneeMotor>0.0</LeftFrontKneeMotor>
            <LeftFrontKneeDriver>0.0</LeftFrontKneeDriver>
            <LeftFrontWheelMotor>0.0</LeftFrontWheelMotor>
            <LeftFrontWheelDriver>0.0</LeftFrontWheelDriver>
            <RightFrontHipXMotor>0.0</RightFrontHipXMotor>
            <RightFrontHipXDriver>0.0</RightFrontHipXDriver>
            <RightFrontHipYMotor>0.0</RightFrontHipYMotor>
            <RightFrontHipYDriver>0.0</RightFrontHipYDriver>
            <RightFrontKneeMotor>0.0</RightFrontKneeMotor>
            <RightFrontKneeDriver>0.0</RightFrontKneeDriver>
            <RightFrontWheelMotor>0.0</RightFrontWheelMotor>
            <RightFrontWheelDriver>0.0</RightFrontWheelDriver>
            <LeftBackHipXMotor>0.0</LeftBackHipXMotor>
            <LeftBackHipXDriver>0.0</LeftBackHipXDriver>
            <LeftBackHipYMotor>0.0</LeftBackHipYMotor>
            <LeftBackHipYDriver>0.0</LeftBackHipYDriver>
            <LeftBackKneeMotor>0.0</LeftBackKneeMotor>
            <LeftBackKneeDriver>0.0</LeftBackKneeDriver>
            <LeftBackWheelMotor>0.0</LeftBackWheelMotor>
```

```
            <LeftBackWheelDriver>0.0</LeftBackWheelDriver>
            <RightBackHipXMotor>0.0</RightBackHipXMotor>
            <RightBackHipXDriver>0.0</RightBackHipXDriver>
            <RightBackHipYMotor>0.0</RightBackHipYMotor>
            <RightBackHipYDriver>0.0</RightBackHipYDriver>
            <RightBackKneeMotor>0.0</RightBackKneeMotor>
            <RightBackKneeDriver>0.0</RightBackKneeDriver>
            <RightBackWheelMotor>0.0</RightBackWheelMotor>
            <RightBackWheelDriver>0.0</RightBackWheelDriver>
        </DeviceTemperature>
        <Led>
            <Fill>
                <Front>0</Front>
                <Back>0</Back>
            </Fill>
        </Led>
        <GPS>
            <Latitude>0.0</Latitude>
            <Longitude>0.0</Longitude>
            <Speed>0.0</Speed>
            <Course>0.0</Course>
            <FixQuality>0.0</FixQuality>
            <NumSatellites>0</NumSatellites>
            <Altitude>0.0</Altitude>
            <HDOP>0.0</HDOP>
            <VDOP>0.0</VDOP>
            <PDOP>0.0</PDOP>
            <VisibleSatellites>0</VisibleSatellites>
        </GPS>
        <DevEnable>
            <FanSpeed>100</FanSpeed>
            <LoadPower>0</LoadPower>
            <LedHost>0</LedHost>
            <LedExt>0</LedExt>
            <FP>0</FP>
            <Lidar>
                <Front>0</Front>
                <Back>0</Back>
            </Lidar>
            <GPS>0</GPS>
            <Video>
                <Front>0</Front>
                <Back>0</Back>
            </Video>
```

```
        </DevEnable>
        <CPU>
            <CPU103>
                <Temperature>0.0</Temperature>
                <FrequencyInt>0.0</FrequencyInt>
                <FrequencyApp>0.0</FrequencyApp>
            </CPU103>
            <CPU105>
                <Temperature>0.0</Temperature>
                <FrequencyInt>0.0</FrequencyInt>
                <FrequencyApp>0.0</FrequencyApp>
            </CPU105>
            <CPU106>
                <Temperature>0.0</Temperature>
                <FrequencyInt>0.0</FrequencyInt>
                <FrequencyApp>0.0</FrequencyApp>
            </CPU106>
        </CPU>
    </Items>
</PatrolDevice>
```

In this response message, the Items field contains six parameter groups: BatteryStatus、DeviceTemperature、Led、GPS、DevEnable and CPU.

The BatteryStatus[4] parameter group provides feedback on the status of batteries of the robot:

Parameters	Meaning	Type	Value
Voltage*	Voltage of the battery (V)	float	
BatteryLevel*	Percentage of remaining battery power (%)	float	[0,100]
Battery_temperature*	Temperature of battery (°C)	float	
Charge*	Charging status of battery	bool	true = Charging false = Not Charging

The DeviceTemperature[5] parameter group provides feedback on the temperature information of each joint motor and driver:

Parameters	Meaning	Type	Value
*Motor	Temperature of motor (°C)	float	

Parameters	Meaning	Type	Value
*Driver	Temperature of driver (°C)	float	

The LED parameter group provides feedback on the front and back flashlight status of the robot:

Parameters	Meaning	Type	Value
Fill:Front / Back	Status of front / back flashlight	int	Off = 0, On = 1

The GPS parameter group provides feedback satellite positioning module positioning data:

Parameters	Meaning	Type	Value
Latitude	Latitude of the robot in the world coordinate system (deg)	float	Positive values indicate north of the equator; Negative values indicate south of the equator.
Longitude	Longitude of the robot in the world coordinate system (deg)	float	Positive values indicate east of the prime meridian; Negative values indicate west of the prime meridian.
Speed	Ground speed (km/h)	float	
Course	Robot course in the world coordinate system (deg)	float	The angle between the direction of travel and true north
FixQuality	Robot positioning quality	float	The higher the value, the more accurate the positioning.
NumSatellites	Number of satellites involved in positioning	int	
Altitude	Altitude of the robot (m)	float	
HDOP	Position Dilution of Precision Comprehensive indicator reflecting	float	[0.5, 99.9] A smaller value indicates

Parameters	Meaning	Type	Value
	positioning accuracy		higher precision.
VDOP	Horizontal Dilution of Precision Reflects positioning accuracy in the horizontal direction	float	
PDOP	Vertical Dilution of Precision Reflects positioning accuracy in the vertical direction	float	
VisibleSatellites	Total number of visible satellites	int	

The DevEnable parameter group provides feedback on the operating status of the robot's internal components. The meanings of some of the parameters are as follows:

Parameters	Meaning	Type	Value
Lidar:Front/Back	Front and back power switch	int	Off = 0, On = 1
GPS	Satellite positioning module power switch	int	Off = 0, On = 1
Video:Front/Back	Front and back camera power switch	int	Off = 0, On = 1

The CPU[6] parameter group provides feedback on the operating status of each CPU inside the robot:

Parameters	Meaning	Type	Value
Temperature	Maximum CPU temperature	°C	
FrequencyInt	Efficiency core (A55) usage (%)	%	
FrequencyApp	Performance core (A76) usage (%)	%	

[4] The asterisk (*) to the right of the parameter item indicates the position of the omitted parameter name. The right side is the side closer to the hard emergency stop button. For example, 'BatteryLevelRight' refers to the remaining battery percentage on the right side (the side closer to the hard emergency stop button). The meanings of other parameters follow the same principle.

[5] The asterisk (*) on the left side of the parameter item indicates the omitted parameter name information. Please refer to [3] to determine the joint name. For example, RightFrontKneeDriver refers to the right front knee joint driver, and the parameter value indicates the temperature information of the right front knee joint driver. The names and meanings of other parameters follow the same principle.

[6] In the CPU parameter group, CPU103, CPU105, and CPU106 correspond to the CPUs of the three hosts inside the robot, respectively. CPU106 is only effective on the Lynx M20 Pro.

1.3.2 Obtain Abnormal Status

You can obtain the current abnormal status information of the robot by this request.

Type	Command	Message type
1002	3	Obtain Abnormal Status

[Notes] The response will actively report to the IP and port that sent the heartbeat command at a fixed frequency of 2 Hz, when the status changes (such as when an abnormal occurs or is resolved), the robot will perform an additional report.

JSON response:

```
{
    "PatrolDevice": {
        "Type": 1002,
        "Command": 3,
        "Time": "2023-01-01 00:00:00",
        "Items": {
            "ErrorList":[
            {
                "errorCode": errorCode,
                "component": value,
            },
            {
                "errorCode": errorCode,
```

```
                "component": value,
            }
            ···//Other omitted errors
            ]
        }
    }
}
```

XML response:

```
<?xml version="1.0" encoding="UTF-8"?>
<PatrolDevice>
  <Type>1002</Type>
  <Command>3</Command>
  <Time>2023-01-01 00:00:00</Time>
  <Items>
    <ErrorList>
      <errorCode>errorCode</errorCode>
      <component>value</component>
    </ErrorList>
    <ErrorList>
      <errorCode>errorCode</errorCode>
      <component>value</component>
    </ErrorList>
  </Items>
</PatrolDevice>
```

In this response message, the items field contains the following parameters:

Parameters	Meaning	Type	Value
errorCode	Error code	int	Check the table below
value	Location of the component where the error occurred[7]	int	Use bits as part numbers

[7] Please refer to Section "1.3.1.1 Motion Control Status Reporting" for joint numbering, which follows the order of parameters in the <MotorStatus> group. Joints are numbered from low to high bits in the order of FL/FR/BL/BR + HipX/HipY/Knee/Wheel. For example, 0x21 (0000 0000 0010 0001) indicates that the 1st joint (bit 0, LeftFrontHipX) and the 6th joint (bit 5, RightFrontHipY) have errors corresponding to the errorCode. Other joint bits follow the same convention. For battery numbering, bit 0 represents the right-side battery

(the side near the hardware emergency stop button), and bit 1 represents the left-side battery.

The meaning of the value of errorCode is shown in the following table:

Value	Meaning	Value	Meaning
0x8001	Motor Temperature Warning	0x8108	Battery Cell Undervoltage Protection
0x8002	Motor Over-temperature Protection	0x8112	Battery Discharge Low-temperature Protection
0x8003	Motor Temperature Critical Shutdown	0x8115	Battery Charging Over-temperature Protection
0x8007	Joint Driver Over-temperature	0x8116	Battery Charge Low-temperature Protection
0x8008	Driver Undervoltage Protection	0x8117	Battery Cell Overvoltage Protection
0x8009	Driver Overvoltage Protection	0x8118	Battery Pack Overvoltage Protection
0x8012	Joint Driver Communication Timeout	0x8119	Battery Pack Undervoltage Protection
0x8016	No Encoder Value	0x8120	Battery Charging Overcurrent Protection
0x8020	Driver Overcurrent Protection	0x8121	Battery Discharge Overcurrent Protection
0x8021	Temperature Sensor Disconnected	0x8122	Short Circuit Protection
0x8022	Joint Angle Limit Exceeded	0x8123	Battery Front-end Detection IC Error
0x8024	Joint Data is NaN	0x8124	Battery Software MOS Lock
0x8025	Joint Data Update Error	0x8125	Battery Discharge Over-temperature Warning
0x8027	Body Attitude Error	0x8126	Battery Discharge Low-temperature Warning
0x8028	Driver Status Error	0x8127	Battery Charging Over-temperature Warning

Value	Meaning	Value	Meaning
0x8029	Motion Attitude Error	0x8128	Battery Charging Low-temperature Warning
0x8030	Joint Driver Over-temperature Warning	0x8129	Battery Output Minimum Voltage Warning
0x8102	Low Battery Warning	0x8201	CPU Usage Overload Warning
0x8103	Protected Battery Level	0x8202	CPU Temperature Overheat Warning
0x8106	Battery Output Minimum Voltage Protection	0x8211	CPU Usage Overload Protection
0x8107	Battery Discharge Over-temperature Protection	0x8212	CPU Temperature Overheat Protection

Appendix 1: UDP Sample Code

Example code for sending an autonomous charging command:

```
#include <iostream>
#include <cstring>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>

#define SERVER_IP "10.21.31.103"
#define PORT 30000
#define BUFFER_SIZE 1024

struct udpMessage
{
    unsigned char header[16];
    unsigned char data[BUFFER_SIZE];
};


int main() {
    // Create a UDP socket
    int client_fd = socket(AF_INET, SOCK_DGRAM, 0);
    if (client_fd < 0) {
        perror("socket creation failed");
        return -1;
    }

    // Set server address
    struct sockaddr_in server_addr;
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(PORT);
    if (inet_pton(AF_INET, SERVER_IP, &server_addr.sin_addr) <= 0) {
        perror("Invalid address/ Address not supported");
        close(client_fd);
        return -1;
    }

    udpMessage message;
    message.header[0] = 0xeb;
    message.header[1] = 0x90;
```

```
    message.header[2] = 0xeb;
    message.header[3] = 0x90;

    // JSON string for sending
    const char *data = R"(
    {
        "PatrolDevice":{
            "Type":2,
            "Command":24,
            "Time":"2023-01-01 00:00:00",
            "Items":{
            }
        }
    }
    )";

    unsigned short dataLength = strlen(data);

    message.header[4] = dataLength & 0xFF;
    message.header[5] = (dataLength >> 8) & 0xFF;
    message.header[6] = 0x01;
    message.header[7] = 0x00;
    message.header[8] = 0x01;

    memcpy(message.data, data, strlen(data));

    // Send data
    ssize_t send_len = sendto(client_fd, &message, dataLength + 16, 0, (struct sockaddr *)&server_addr,
sizeof(server_addr));
    if (send_len < 0) {
        perror("sendto failed");
        close(client_fd);
        return -1;
    }

    std::cout << "Message sent successfully." << std::endl;

    // Close the socket
    close(client_fd);
    return 0;
}
```

Appendix 2: Obtaining camera video stream

The front and rear wide-angle cameras of the Lynx M20 use the RTSP protocol for streaming. The RTSP addresses are as follows:

Camera Position	RTSP Address
Front Wide-angle Camera	rtsp://10.21.31.103:8554/video1
Rear Wide-angle Camera	rtsp://10.21.31.103:8554/video2

Developers can pull the RTSP video streams of the front and rear wide-angle cameras using the above addresses.