ArcelorMittal — Documento de Engenharia

Sistema de Rastreamento de Carros Torpedo

Especificação Técnica, Arquitetura e Operação

Número do Documento:

Revisão: 1.0

Data: 02/02/2026

Elaborado por: Luciano Rocha / Automação e TI Industrial

Histórico de Revisões

Rev.	Data	Descrição	Responsável
1.0	02/02/2026	Emissão inicial	Luciano Rocha

1. Introdução

Este documento descreve a arquitetura, componentes, fluxos e procedimentos para o Sistema de Rastreamento de Carros Torpedo, visando rastreamento contínuo por GPS e BLE com comunicação LoRaWAN, integração ao Network Server da Kore Brasil e entrega de payloads JSON ao Datalake da ArcelorMittal.

2. Escopo

Abrange hardware (trackers LW001-BG-PRO B e beacons M2), infraestrutura LoRaWAN, integração de dados.

3. Glossário e Siglas

BLE — Bluetooth Low Energy;

PDOP — Position Dilution of Precision;

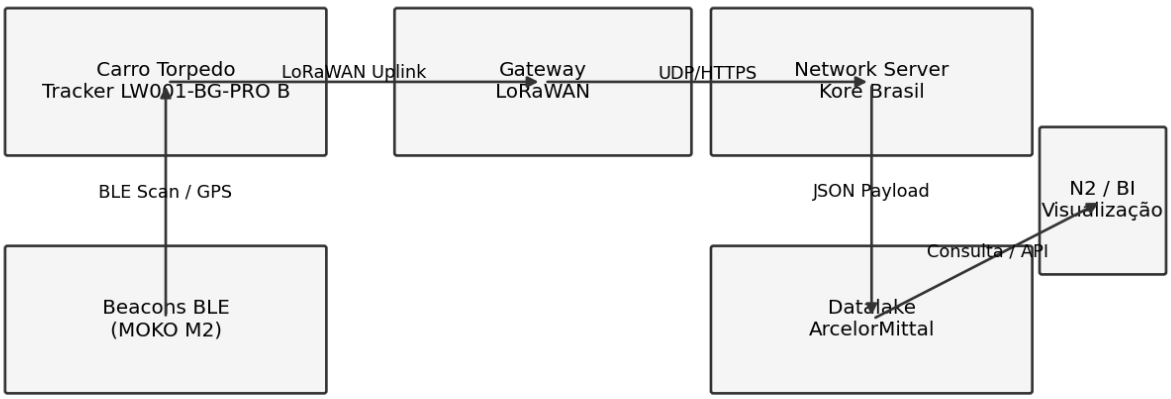
RSSI — Received Signal Strength Indicator;

LoRaWAN — Long Range Wide Area Network.

4. Arquitetura do Sistema

Quantidade de rastreadores: 23 unidades LW001-BG-PRO B,

Quantidade de beacons BLE M2: 30 unidades



Observação: BLE fornece posição aproximada; GPS fornece posição exata quando disponível.

5. Componentes

• Rastreadores LW001-BG-PRO B: posicionamento GPS e fallback BLE; alimentação por bateria interna e bateria externa 50Ah.

• Beacons BLE M2: broadcasting sob UUID da aplicação; utilizados para posicionamento aproximado por proximidade de MAC/RSSI.

• Gateways LoRaWAN: recepção e envio ao Network Server.

• Network Server (Kore Brasil): decodificação Base64, enriquecimento e envio como JSON.

• Datalake ArcelorMittal: armazenamento e consumo analítico/operacional.

6. Endereçamento dos Beacons (MAC)

Beacon	MAC
1	EA5639205EB5
2	F847D9B4C045
3	EEC0ED074751
4	D72AC0EE3145
5	FC15B6F05CDA
6	F0B8FBE471FB
7	F440B85F4347
8	CA29381D7730

9	CA3A46BF915D
10	C8957CA525E1
11	E71E10D418EE
12	D325647BEABD
13	E7A4C10309C6
14	C7698D919FA9
15	E6A8376633F2
16	EC219D641AB2
17	E1D7F77FECA7
18	C8B62BB208A3
19	E56C622CC448
20	EDEDACAD7373
21	E7B3FDBF8B52
22	D08D272E9048
23	DCB0A1FC6DD5
24	D8978C863BF5
25	DF4998414218
26	C540134CB92E
27	D5E5771C5DF3
28	F371F348C62F
29	F4EB34FCB347
30	FCEB6A1E81C8

7. Fluxo de Dados

O fluxo de dados compreende o uplink LoRaWAN, processamento no Network Server, decodificação para JSON e ingestão no Datalake. A consulta é realizada via SQL conforme abaixo.

Consulta SQL para consumo analítico e N2

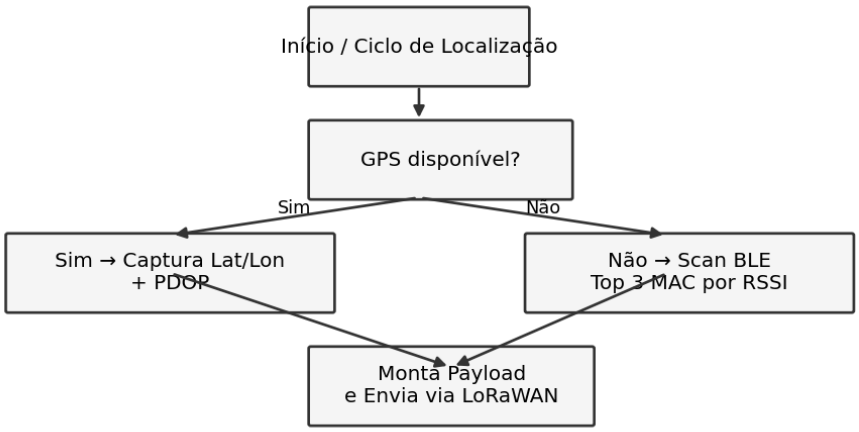
Consulta SQL:

```
SELECT *
  FROM FCB_PRD_SILVER_CORP.LORAWAN.DEVICE_MESSAGE
 WHERE DEV_APPNAME = 'SEU-TRACKER-CARRO_TORPEDO'
ORDER BY
```

8. Estratégia de Posicionamento (GPS/BLE)

Quando o GPS está disponível, envia lat/lon e PDOP. Na indisponibilidade, realiza varredura BLE e reporta os três MACs mais fortes (RSSI).

UUID da aplicação BLE: 0x000102030405060707090a0b0c0d0e0f



9. Instalação e Energia

Os trackers são alojados em caixas plásticas com proteção IP67 e alimentados por bateria interna e bateria externa de 50Ah no painel.

10. Configuração dos Dispositivos

Conforme telas no ANEXO A.

11. Decodificação de Payload (Decoder JS)

Decoder corrigido está disponível no ANEXO B para decodificação dos payloads no Networkserver da Kore.

Anexo A — Imagens do Rascunho

Figura 1 — Mapa de instalação dos beacons na área.)

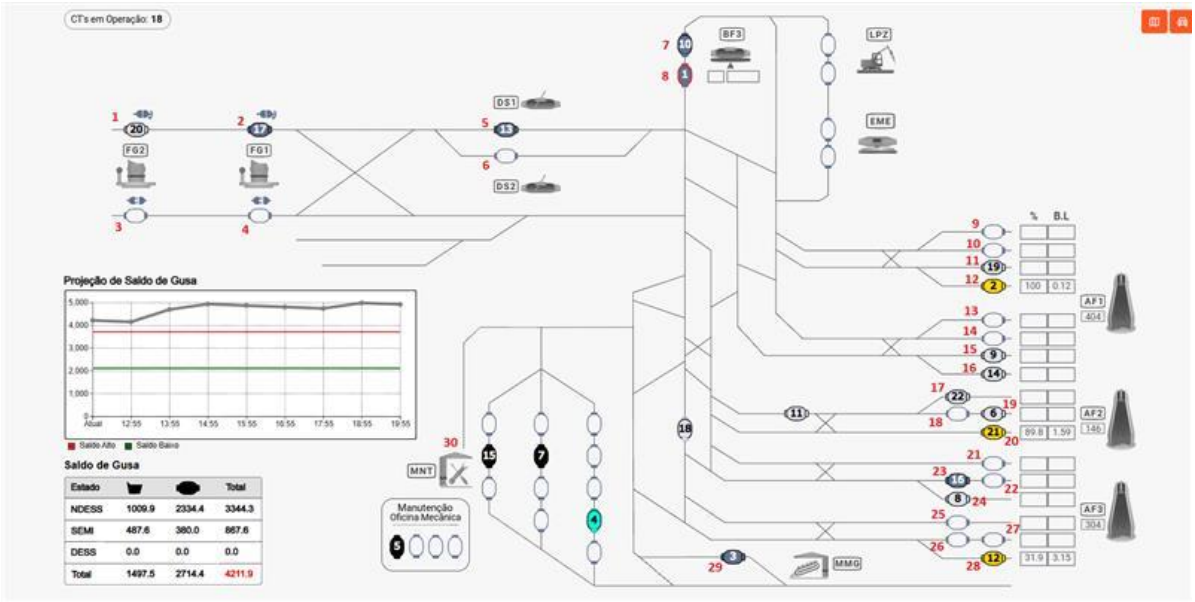


Figura 2 — Beacons BLE Moko Smart modelo M2.

Figura 3 — Especificações gerais dos beacons

General specifications	
Main Chip	Nordic nRF52810
Bluetooth	Bluetooth 4.2
Dimension	70mm x 46mm x 21mm
Range	Up to 160 meters (in the open area and no obstacles)
Weight	33.5g (With battery)
Material	ABS+PC
Waterproof	IP67
Color	Gray
Installation	Hang, Sticker, Screw
Button	Mechanical button
LED	Single red LED

Figura 4 — Especificação geral dos beacons.

General specifications	
Sensor	Omni polar Hall-effect sensor for magnetic switch 3-axis accelerometer sensor (optional)
Operating temperature	General -20°C / + 60°C -40°C / + 85°C can be customized
Storage temperature	-20°C / + 70°C (without battery) 10°C / + 25°C (with battery)
Humidity	0% ~ 95% (non-condensing)
Antenna Type	PCB on-board antenna
Power supply	Replaceable 1000mAh lithium coin CR2477 battery

Figura 5 — Imagem ilustrativa dos rastreadores GPS+BLE.

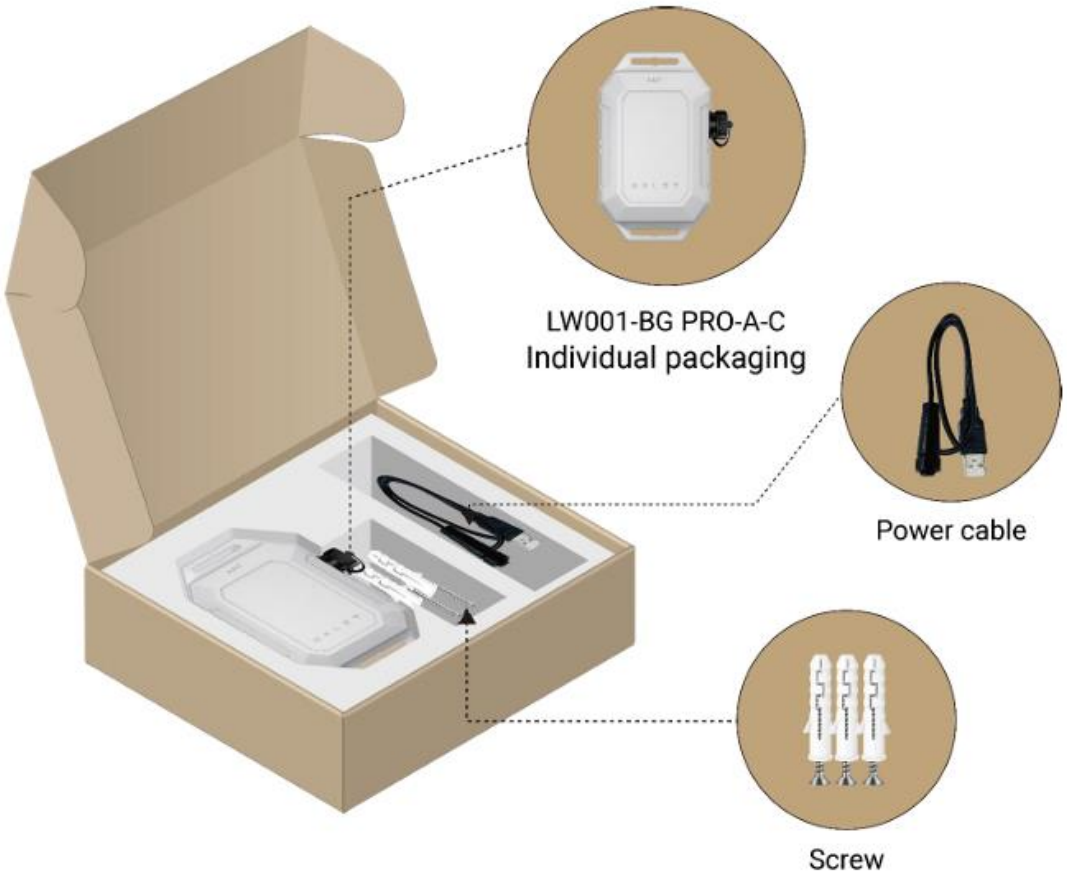


Figura 6 — Painel montado com rastreador e bateria de 50Ah.

Figura 7 — Imagem importada do rascunho (image7.jpeg)

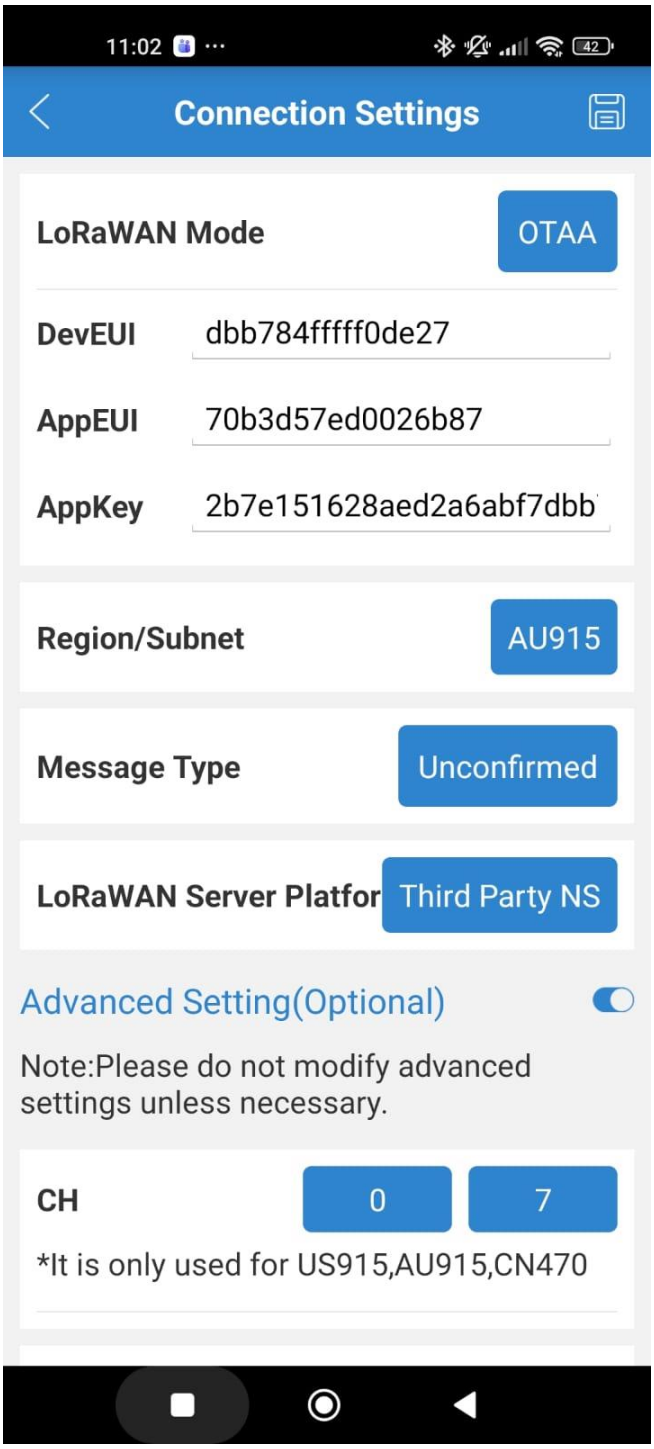


Figura 8 — Imagem importada do rascunho (image8.jpeg)

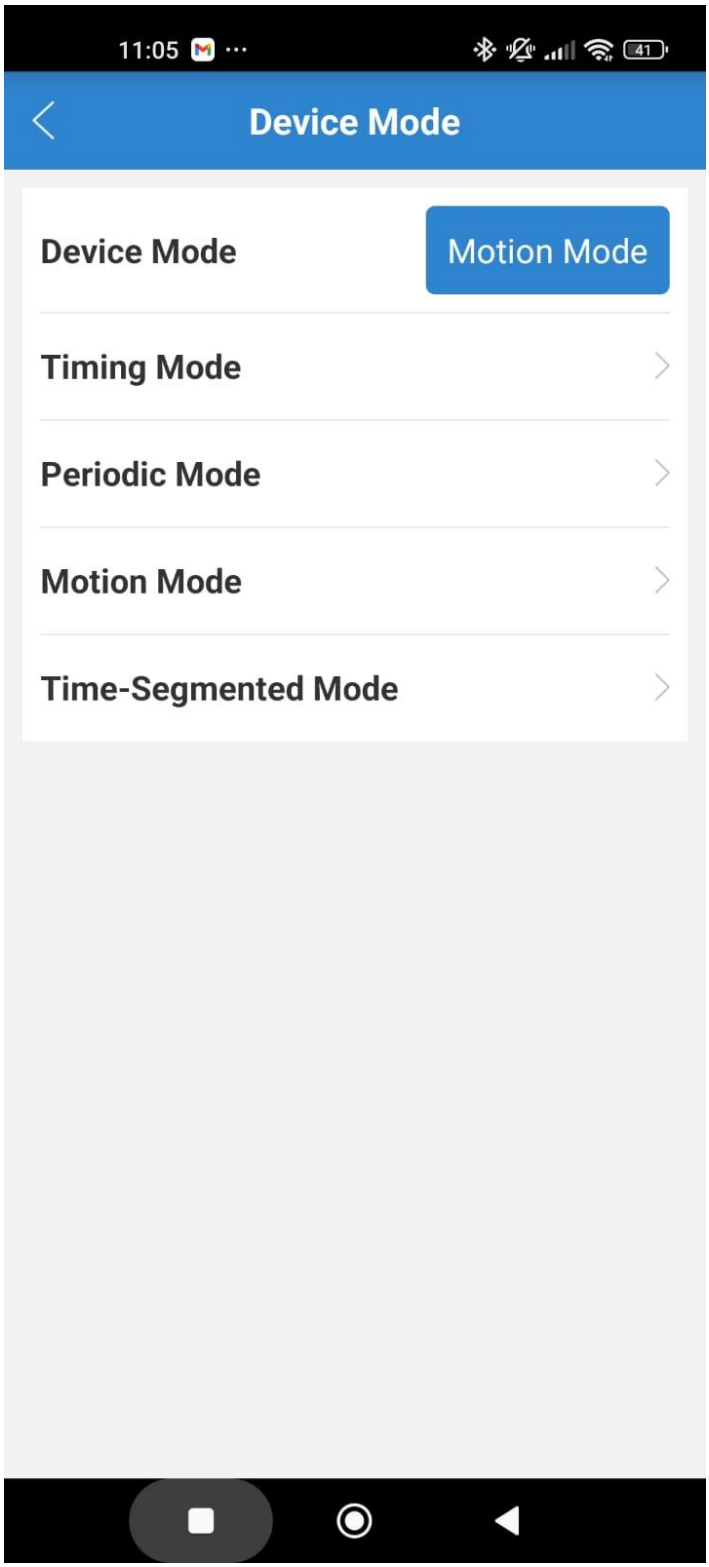


Figura 9 — Imagem importada do rascunho (image9.jpeg)

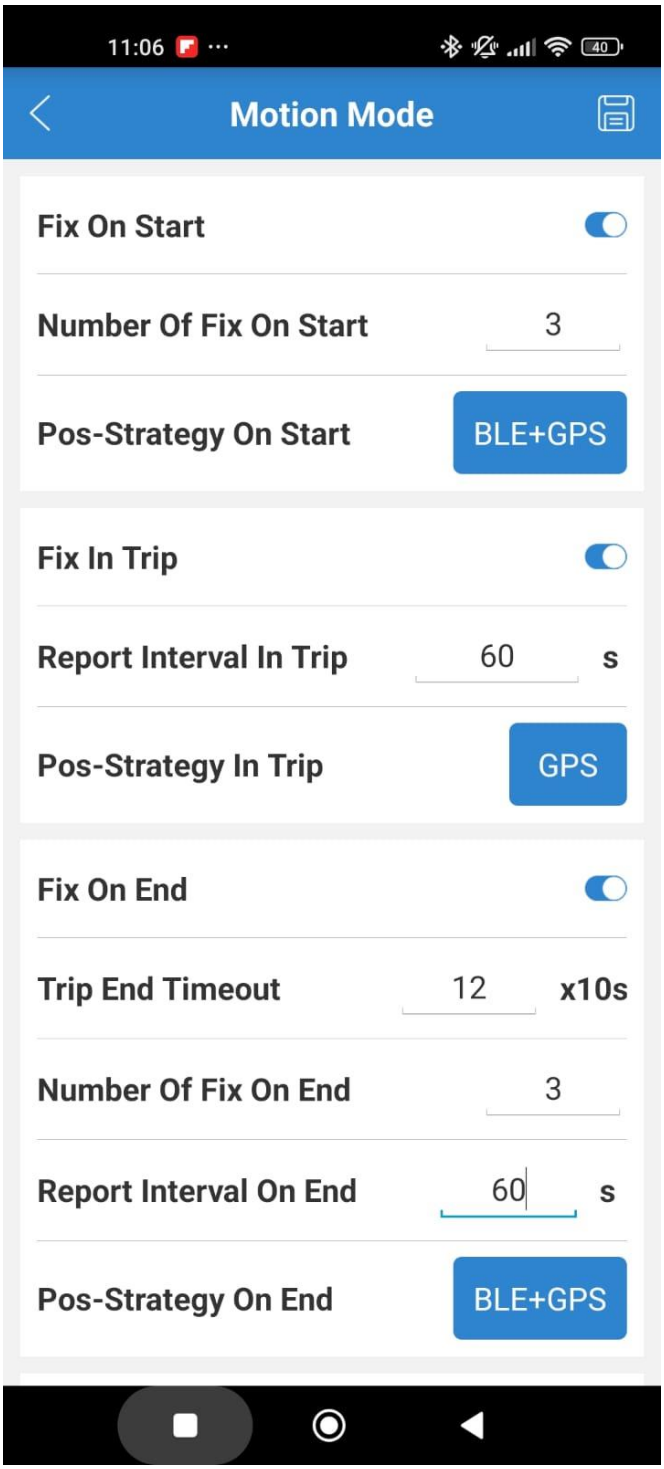


Figura 10 — Imagem importada do rascunho (image10.jpeg)

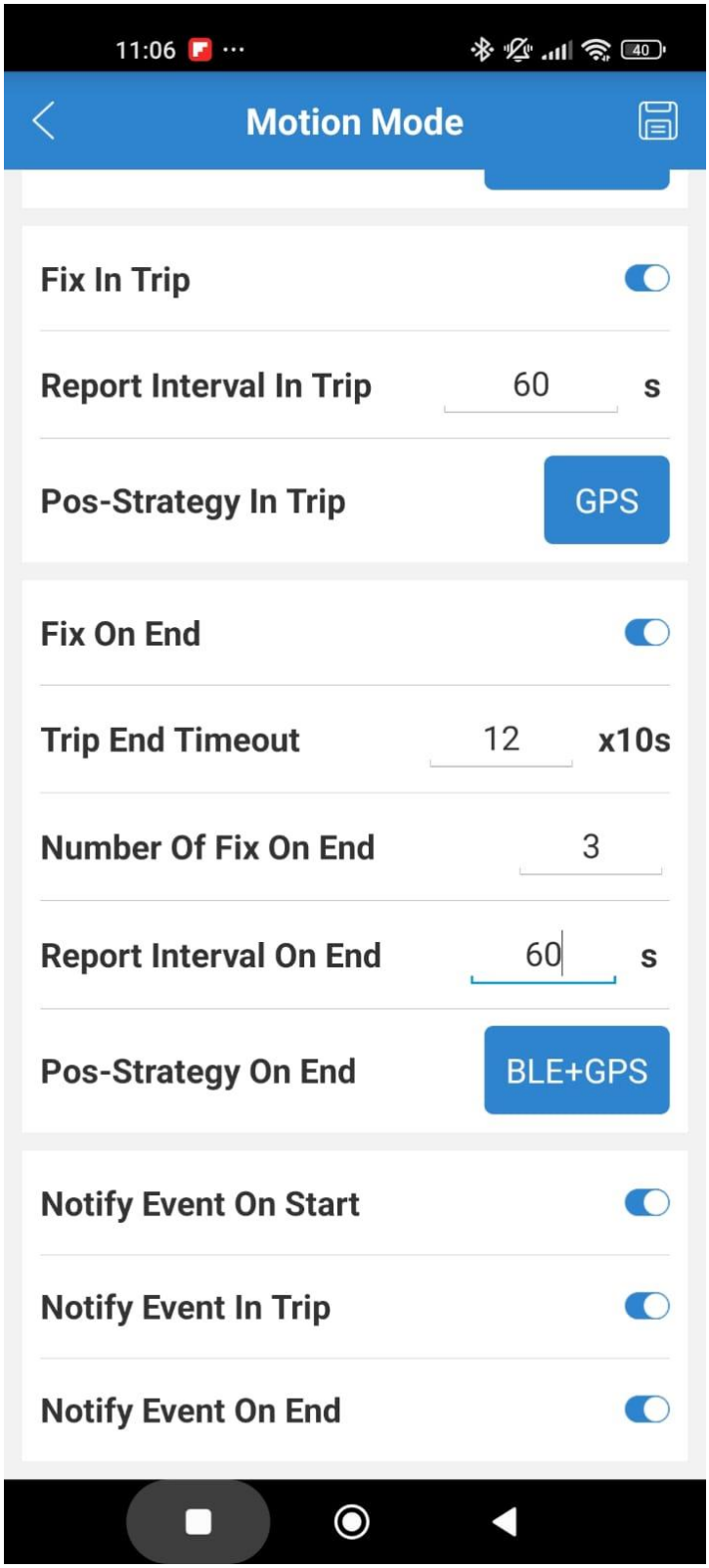


Figura 11 — Imagem importada do rascunho (image11.jpeg)

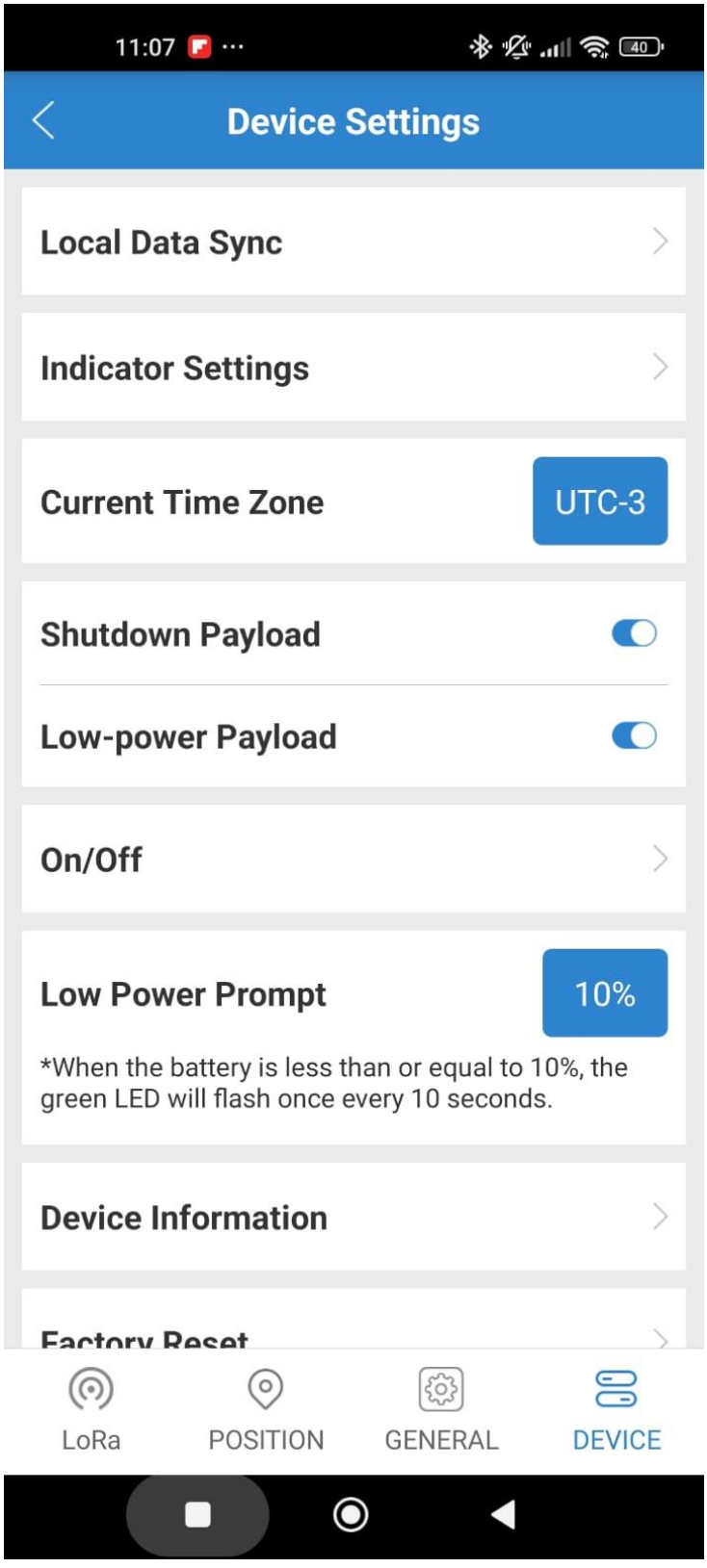
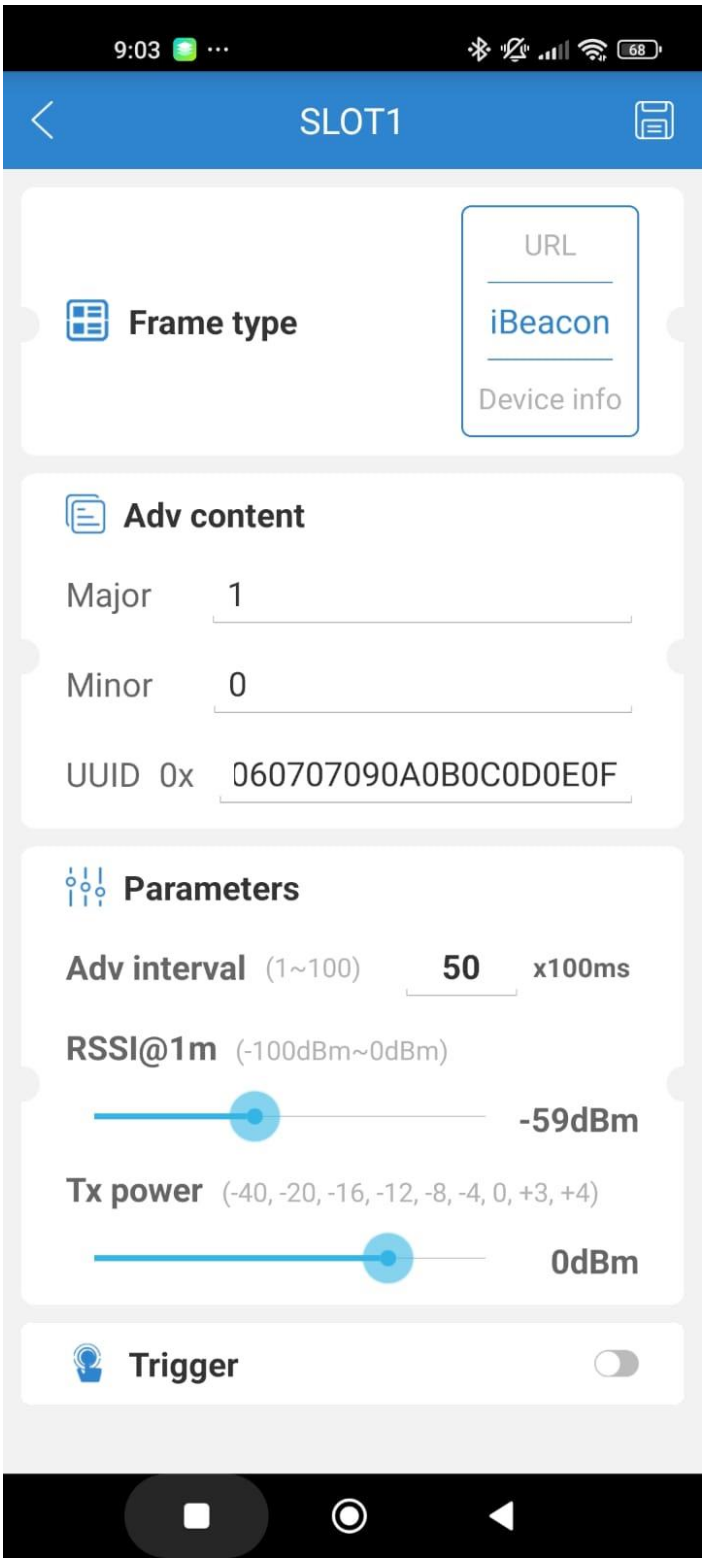


Figura 12 — Imagem importada do rascunho (image12.jpeg)



UUID da aplicação: 0x000102030405060707090a0b0c0d0e0f

Anexo B — Decoder JavaScript (Completo)

```
var payloadTypeArray = ['Heartbeat', 'Location Fixed', 'Location Failure', 'Shutdown',
'Shock', 'Man Down detection', 'Tamper Alarm', 'Event Message', 'Battery Consumption',
'', '', 'GPS Limit'];
var operationModeArray = ['Standby mode', 'Periodic mode', 'Timing mode', 'Motion mode'];
var rebootReasonArray = ['Restart after power failure', 'Bluetooth command request',
'LoRaWAN command request', 'Power on after normal power off'];
var positionTypeArray = ['WIFI positioning success', 'Bluetooth positioning success',
'GPS positioning success'];
var posFailedReasonArray = [
    'WIFI positioning time is not enough (The location payload reporting interval is set
too short, please increase the report interval of the current working mode via MKLoRa
app)',
    'WIFI positioning strategies timeout (Please increase the WIFI positioning timeout
via MKLoRa app)',
    'WIFI module is not detected, the WIFI module itself works abnormally',
    'Bluetooth positioning time is not enough (The location payload reporting interval is
set too short, please increase the report interval of the current working mode via MKLoRa
app)',
    'Bluetooth positioning strategies timeout (Please increase the Bluetooth positioning
timeout via MKLoRa app)',
    'Bluetooth broadcasting in progress (Please reduce the Bluetooth broadcast timeout or
avoid Bluetooth positioning when Bluetooth broadcasting in process via MKLoRa app)',
    'GPS position time budget over (Pls increase the GPS budget via MKLoRa app)',
    'GPS coarse positioning timeout (Pls increase coarse positioning timeout or increase
coarse accuracy target via MKLoRa app)',
    'GPS fine positioning timeout (Pls increase fine positioning timeout or increase fine
accuracy target via MKLoRa app)',
    'GPS positioning time is not enough (The location payload reporting interval is set
too short, please increase the report interval of the current working mode via MKLoRa
app)',
    'GPS aiding positioning timeout (Please adjust GPS autonomous latitude and autonomous
longitude)',
    'GPS cold start positioning timeout (The gps signal current environment isn’t very
good, please leave the device in a more open area)',
    'Interrupted by Downlink for Position',
    'Interrupted positioning at start of movement(the movement ends too quickly,
resulting in not enough time to complete the positioning)',
    'Interrupted positioning at end of movement(the movement restarted too quickly,
resulting in not enough time to complete the positioning)'
];
var shutdownTypeArray = ['Bluetooth command to turn off the device', 'LoRaWAN command to
turn off the device', 'Magnetic to turn off the device'];
var eventTypeArray = [
    'Start of movement',
    'In movement',
    'End of movement',
    'Uplink Payload triggered by downlink message'
];
function Decoder(bytes, fPort, groupID) {
    var payloadList = [];
    var payload_type = payloadTypeArray[fPort - 1];
    if (fPort <= 10) {
        var operationModeCode = bytes[0] & 0x03;
        var operation_mode = operationModeArray[operationModeCode];
        payloadList.push(getPayloadData('operation_mode', operation_mode, groupID));
        var batteryLevelCode = bytes[0] & 0x04;
        var battery_level = batteryLevelCode == 0 ? 'Normal' : 'Low battery';
        payloadList.push(getPayloadData('battery_level', battery_level, groupID));
        var tamperAlarmCode = bytes[0] & 0x08;
        var tamper_alarm = tamperAlarmCode == 0 ? 'Not triggered' : 'Triggered';
```

```
        payloadList.push(getPayloadData('tamper_alarm', tamper_alarm, groupID));
        var manDownStatusCode = bytes[0] & 0x10;
        var mandown_status = manDownStatusCode == 0 ? 'Not in idle' : 'In idle';
        payloadList.push(getPayloadData('mandown_status', mandown_status, groupID));
        var motionStateSinceLastPaylaodCode = bytes[0] & 0x20;
        var motion_state_since_last_paylaod = motionStateSinceLastPaylaodCode == 0 ? 'No'
: 'Yes';
        payloadList.push(getPayloadData('motion_state_since_last_paylaod',
motion_state_since_last_paylaod, groupID));
        if (fPort == 2 || fPort == 3) {
            var positioningTypeCode = bytes[0] & 0x40;
            var positioning_type = positioningTypeCode == 0 ? 'Normal' : 'Downlink for
position';
            payloadList.push(getPayloadData('positioning_type', positioning_type,
groupID));
        }
        var temperature = signedHexToInt(bytesToHexString(bytes, 1, 1)).toString() +
'°C';
        payloadList.push(getPayloadData('temperature', temperature, groupID));
        var ack = bytes[2] & 0x0f;
        payloadList.push(getPayloadData('ack', ack, groupID));
        var battery_voltage = ((28 + ((bytes[3] >> 4) & 0x0f)) / 10).toString() + 'V';
        payloadList.push(getPayloadData('battery_voltage', battery_voltage, groupID));
        var chargingCode = bytes[0] & 0x80;
        var charging_status = chargingCode == 0 ? 'not charging' : 'charging';
        payloadList.push(getPayloadData('charging_status', charging_status, groupID));
    }
    if (fPort == 1) {
        var rebootReasonCode = bytesToInt(bytes, 5, 1);
        // dev_info.reboot_reason_code = rebootReasonCode;
        var reboot_reason = rebootReasonArray[rebootReasonCode];
        payloadList.push(getPayloadData('reboot_reason', reboot_reason, groupID));
        var majorVersion = (bytes[6] >> 6) & 0x03;
        var minorVersion = (bytes[6] >> 4) & 0x03;
        var patchVersion = bytes[6] & 0x0f;
        var firmwareVersion = 'V' + majorVersion.toString() + '.' +
minorVersion.toString() + '.' + patchVersion.toString();
        var firmware_version = firmwareVersion;
        payloadList.push(getPayloadData('firmware_version', firmware_version, groupID));
        var activityCount = bytesToInt(bytes, 7, 4);
        var activity_count = activityCount;
        payloadList.push(getPayloadData('activity_count', activity_count, groupID));
    }
    else if (fPort == 2) {
        var parse_len = 5; // a documentação está errada, o common header vale para todos
os tipos de payload
        var positionTypeCode = bytes[parse_len++];
        var position_success_type = positionTypeArray[positionTypeCode];
        payloadList.push(getPayloadData('position_success_type', position_success_type,
groupID));
        var year = bytes[parse_len] * 256 + bytes[parse_len + 1];
        parse_len += 2; // parse_len = 6
        var mon = bytes[parse_len++]; // parse_len = 6
        var days = bytes[parse_len++];// parse_len = 7
        var hour = bytes[parse_len++];// parse_len = 8
        var minute = bytes[parse_len++];// parse_len = 9
        var sec = bytes[parse_len++];// parse_len = 10
        var timezone = bytes[parse_len++];// parse_len = 11
        if (timezone > 127) timezone-= 256; //convertendo para inteiro com sinal
        payloadList.push(getPayloadData('timezone', timezone, groupID));
        var timestamp = '';
        if (timezone > 0x80) {
```

```
            timestamp = year.toString() + '-' + mon.toString() + '-' + days.toString() +
' ' + hour.toString() + ':' + minute.toString() + ':' + sec.toString()
                + '  TZ:' + (timezone - 0x100).toString();
        }
        else {
            timestamp = year.toString() + '-' + mon.toString() + '-' + days.toString() +
' ' + hour.toString() + ':' + minute.toString() + ':'
                + sec.toString() + '  TZ:' + timezone.toString();
        }
        payloadList.push(getPayloadData('timestamp', timestamp, groupID));
        var datalen = bytes[parse_len++]; // parse_len = 12
        if (positionTypeCode == 0 || positionTypeCode == 1) {
            for (var i = 0; i < (datalen / 7); i++) { // Linha recomendada pelo GPT
                var mac = bytesToHexString(bytes, parse_len, 6); // parse_len = 13
                payloadList.push(getPayloadData('positioning_success_mac' + i.toString(),
mac, groupID));
                parse_len += 6;
                var rssi = (bytes[parse_len++] - 256).toString() + 'dBm'; // parse_len =
19
                payloadList.push(getPayloadData('positioning_success_rssi' +
i.toString(), rssi, groupID));
            }
        }
        else {
            var lat = bytesToInt(bytes, parse_len, 4); //parse_len = 13
            parse_len += 4;
            var lon = bytesToInt(bytes, parse_len, 4);
            parse_len += 4;

            if (lat >= 0x80000000) // Trecho recomendado pelo GPT
                lat -= 0x100000000;
            if (lon >= 0x80000000)
                lon -= 0x100000000;

            if (lat > 0x80000000)
                lat = lat - 0x100000000;
            if (lon > 0x80000000)
                lon = lon - 0x100000000;
            var latitude = (lat / 10000000);
            var longitude = (lon / 10000000);
            var location_1 = {
                'variable': 'location',
                'value': 'My Address',
                'location': {
                    'lat': latitude,
                    'lng': longitude,
                },
                'group': groupID,
                'metadata': {
                    'color': '#add8e6'
                },
            };
            payloadList.push(location_1);
            var pdop_1 = (bytes[parse_len] / 10).toFixed(1).toString();
            payloadList.push(getPayloadData('pdop', pdop_1, groupID));
        }
    }
    else if (fPort == 3) {
        var parse_len = 5; //Valor original 3, porem no fPort 2 nao funcionou
        var failedTypeCode = bytesToInt(bytes, parse_len++, 1);
        var reasons_for_positioning_failure = posFailedReasonArray[failedTypeCode];
        payloadList.push(getPayloadData('reasons_for_positioning_failure',
```

```
reasons_for_positioning_failure, groupID));
        var datalen = bytes[parse_len++];
        if (failedTypeCode <= 5) //wifi and ble reason
         {
            if (datalen) {
                for (var i = 0; i < (datalen / 7); i++) {
                    var mac = bytesToHexString(bytes, parse_len, 6);
                    payloadList.push(getPayloadData('positioning_success_mac' +
i.toString(), mac, groupID));
                    parse_len += 6;
                    var rssi = (bytes[parse_len++] - 256).toString() + 'dBm';
                    payloadList.push(getPayloadData('positioning_success_rssi' +
i.toString(), rssi, groupID));
                }
            }
        }
        else if (failedTypeCode <= 11) //gps reason
         {
            var pdop = '';
            if (bytes[parse_len] != 0xff)
                pdop = (bytes[parse_len] / 10).toString();
            else
                pdop = 'unknow';
            payloadList.push(getPayloadData('pdop', pdop, groupID));
            parse_len++;
            var gps_satellite_cn = bytes[parse_len].toString() + '-' + bytes[parse_len +
1].toString() + '-' + bytes[parse_len + 2].toString() + '-' + bytes[parse_len +
3].toString();
            payloadList.push(getPayloadData('gps_satellite_cn', gps_satellite_cn,
groupID));
        }
    }
    else if (fPort == 4) {
        var shutdownTypeCode = bytesToInt(bytes, 5, 1);
        var shutdown_type = shutdownTypeArray[shutdownTypeCode];
        payloadList.push(getPayloadData('shutdown_type', shutdown_type, groupID));
    }
    else if (fPort == 5) {
        var number_of_shocks = bytesToInt(bytes, 5, 2);
        payloadList.push(getPayloadData('number_of_shocks', number_of_shocks, groupID));
    }
    else if (fPort == 6) {
        var total_idle_time = bytesToInt(bytes, 5, 2);
        payloadList.push(getPayloadData('total_idle_time', total_idle_time, groupID));
    }
    else if (fPort == 7) {
        var parse_len = 5; // common head is 5 byte
        var year = bytesToInt(bytes, parse_len, 2).toString();
        parse_len += 2;
        var mon = bytes[parse_len++].toString();
        var days = bytes[parse_len++].toString();
        var hour = bytes[parse_len++].toString();
        var minute = bytes[parse_len++].toString();
        var sec = bytes[parse_len++].toString();
        var timezone = bytes[parse_len++];
        payloadList.push(getPayloadData('timezone', timezone, groupID));
        var timestamp = '';
        if (timezone > 0x80) {
            timestamp = year + '-' + mon + '-' + days + ' ' + hour + ':' + minute + ':' +
sec + '  TZ:' + (timezone - 0x100);
        }
        else {
```

```
            timestamp = year + '-' + mon + '-' + days + ' ' + hour + ':' + minute + ':' +
sec + '  TZ:' + timezone;
        }
        payloadList.push(getPayloadData('timestamp', timestamp, groupID));
    }
    else if (fPort == 8) {
        var eventTypeCode = bytesToInt(bytes, 5, 1);
        var event_type = eventTypeArray[eventTypeCode];
        payloadList.push(getPayloadData('event_type', event_type, groupID));
    }
    else if (fPort == 9) {
        var parse_len = 5;
        var gps_work_time = bytesToInt(bytes, parse_len, 4);
        payloadList.push(getPayloadData('gps_work_time', gps_work_time, groupID));
        parse_len += 4;
        var wifi_work_time = bytesToInt(bytes, parse_len, 4);
        payloadList.push(getPayloadData('wifi_work_time', wifi_work_time, groupID));
        parse_len += 4;
        var ble_scan_work_time = bytesToInt(bytes, parse_len, 4);
        payloadList.push(getPayloadData('ble_scan_work_time', ble_scan_work_time,
groupID));
        parse_len += 4;
        var ble_adv_work_time = bytesToInt(bytes, parse_len, 4);
        payloadList.push(getPayloadData('ble_adv_work_time', ble_adv_work_time,
groupID));
        parse_len += 4;
        var lora_work_time = bytesToInt(bytes, parse_len, 4);
        payloadList.push(getPayloadData('lora_work_time', lora_work_time, groupID));
        parse_len += 4;
    }
    else if (fPort == 12) {
        var operationModeCode = bytes[0] & 0x03;
        var operation_mode = operationModeArray[operationModeCode];
        payloadList.push(getPayloadData('operation_mode', operation_mode, groupID));
        var batteryLevelCode = bytes[0] & 0x04;
        var battery_level = batteryLevelCode == 0 ? 'Normal' : 'Low battery';
        payloadList.push(getPayloadData('battery_level', battery_level, groupID));
        var tamperAlarmCode = bytes[0] & 0x08;
        var tamper_alarm = tamperAlarmCode == 0 ? 'Not triggered' : 'Triggered';
        payloadList.push(getPayloadData('tamper_alarm', tamper_alarm, groupID));
        var manDownStatusCode = bytes[0] & 0x10;
        var mandown_status = manDownStatusCode == 0 ? 'Not in idle' : 'In idle';
        payloadList.push(getPayloadData('mandown_status', mandown_status, groupID));
        var motionStateSinceLastPaylaodCode = bytes[0] & 0x20;
        var motion_state_since_last_paylaod = motionStateSinceLastPaylaodCode == 0 ? 'No'
: 'Yes';
        payloadList.push(getPayloadData('motion_state_since_last_paylaod',
motion_state_since_last_paylaod, groupID));
        var positioningTypeCode = bytes[0] & 0x40;
        var positioning_type = positioningTypeCode == 0 ? 'Normal' : 'Downlink for
position';
        payloadList.push(getPayloadData('positioning_type', positioning_type, groupID));
        var lorawan_downlink_count = bytes[1] & 0x0f;
        payloadList.push(getPayloadData('lorawan_downlink_count', lorawan_downlink_count,
groupID));
        var battery_voltage = ((22 + ((bytes[1] >> 4) & 0x0f)) /
10).toFixed(1).toString() + 'V';
        payloadList.push(getPayloadData('battery_voltage', battery_voltage, groupID));
        var parse_len = 2;
        lat = bytesToInt(bytes, parse_len, 4);
        parse_len += 4;
        lon = bytesToInt(bytes, parse_len, 4);
```

```
        parse_len += 4;
        if (lat > 0x80000000)
            lat = lat - 0x100000000;
        if (lon > 0x80000000)
            lon = lon - 0x100000000;
        var latitude = lat / 10000000;
        payloadList.push(getPayloadData('latitude', latitude, groupID));
        var longitude = lon / 10000000;
        payloadList.push(getPayloadData('longitude', longitude, groupID));
        var pdop_2 = (bytes[parse_len] / 10).toFixed(1).toString();
        payloadList.push(getPayloadData('pdop', pdop_2, groupID));
    }
    return payloadList;
}
/*
    整型数组指定部分转换成对应的 Hex 字符串
    bytes:里面全部为整数,
    start:开始转换的位置
    len:需要转换的长度
*/
function bytesToHexString(bytes, start, len) {
    if (bytes.length == 0 || start >= bytes.length || (start + len) > bytes.length)
        return '';
    var hexStr = '';
    for (var i = 0; i < len; i++) {
        var tempBytes = bytes[start + i];
        var data = tempBytes.toString(16);
        if (tempBytes < 16) {
            data = '0' + data;
        }
        hexStr += data;
    }
    return hexStr;
}
/*
    整型数组指定部分十六进制转换成对应的 10 进制整数
    bytes:里面全部为整数,
    start:开始转换的位置
    len:需要转换的长度
*/
function bytesToInt(bytes, start, len) {
    if (bytes.length == 0 || start >= bytes.length || (start + len) > bytes.length)
        return 0;
    var value = 0;
    for (var i = 0; i < len; i++) {
        var m = ((len - 1) - i) * 8;
        value = value | bytes[start + i] << m;
    }
    return value;
}
function timezone_decode(tz) {
    var tz_str = 'UTC';
    tz = tz > 128 ? tz - 256 : tz;
    if (tz < 0) {
        tz_str += '-';
        tz = -tz;
    }
    else {
```

```
        tz_str += '+';
    }
    if (tz < 20) {
        tz_str += '0';
    }
    tz_str += String(tz / 2);
    tz_str += ':';
    if (tz % 2) {
        tz_str += '30';
    }
    else {
        tz_str += '00';
    }
    return tz_str;
}
function parse_time(timestamp, timezone) {
    timezone = timezone > 64 ? timezone - 128 : timezone;
    timestamp = timestamp + timezone * 3600;
    if (timestamp < 0) {
        timestamp = 0;
    }
    var d = new Date(timestamp * 1000);
    //d.setUTCSeconds(1660202724);
    var time_str = '';
    time_str += d.getUTCFullYear();
    time_str += '-';
    time_str += formatNumber(d.getUTCMonth() + 1);
    time_str += '-';
    time_str += formatNumber(d.getUTCDate());
    time_str += ' ';
    time_str += formatNumber(d.getUTCHours());
    time_str += ':';
    time_str += formatNumber(d.getUTCMinutes());
    time_str += ':';
    time_str += formatNumber(d.getUTCSeconds());
    return time_str;
}
function formatNumber(number) {
    return number < 10 ? '0' + number.toString() : number.toString();
}
/*
    有符号十六进制字符串转十进制
*/
function signedHexToInt(hexStr) {
    var twoStr = parseInt(hexStr, 16).toString(2); // 将十六转十进制，再转 2 进制

    var bitNum = hexStr.length * 4; // 1 个字节 = 8bit , 0xff 一个 'f'就是 4 位
    if (twoStr.length < bitNum) {
        while (twoStr.length < bitNum) {
            twoStr = '0' + twoStr;
        }
    }
    if (twoStr.substring(0, 1) == '0') {
        // 正数
        return parseInt(twoStr, 2); // 二进制转十进制
    }
    // 负数
    var twoStr_unsign = '';
    var tempValue = parseInt(twoStr, 2) - 1; // 补码：(负数)反码+1，符号位不变；相对十进制来说也
```

```
是 +1，但这里是负数，+1 就是绝对值数据-1
    twoStr_unsign = tempValue.toString(2).substring(1, bitNum); // 舍弃首位(符号位)
    // 去除首字符，将 0 转为 1，将 1 转为 0   反码
    twoStr_unsign = twoStr_unsign.replace(/0/g, 'z');
    twoStr_unsign = twoStr_unsign.replace(/1/g, '0');
    twoStr_unsign = twoStr_unsign.replace(/z/g, '1');
    return parseInt('-' + twoStr_unsign, 2);
}
function getPayloadData(type, value, groupID) {
    return {
        'variable': type,
        'value': value,
        'group': groupID
    };
}
var payloadd = payload.find(function (x) { return ["payload_raw", "payload",
"data"].includes(x.variable); });
var portt = payload.find(function (x) { return ["port", "fport",
"f_port"].includes(x.variable); });

if (payloadd.value && portt.value) {
    try {
        // Convert the data from Hex to Javascript Buffer.
        var buffer = hexToNumberArray(payloadd.value);
        // payload.push(...Decoder(buffer, port.value, payload_raw.group));
        payload = payload.concat(Decoder(buffer, portt.value, payloadd.group));
    }
```