```
/**
 * @params
 *        - fPort: 85
 *        - bytes: Payload
 */
function Decode(fPort, bytes) {
    var decoded = {};

    decoded.devEUI          =   LoRaObject.devEUI;
    decoded.base64          =   LoRaObject.data;
    decoded.frequency       =   LoRaObject.txInfo.frequency;
    decoded.dataRate        =   LoRaObject.txInfo.dataRate;
    decoded.uplink          =   readUplinkInfo(bytes);

    return decoded;
}

/* ****************************************
 * uplink info
 ****************************************/
function readUplinkInfo(bytes) {
    var uplink = {};

    uplink.flow                     = readUInt32(0,bytes) / 100;
    uplink.total                    = readUInt32(4,bytes) / 100;
    uplink.vel                      = readUInt16(8,bytes) / 100;
    uplink.SignalQuality            = readUInt8(10,bytes);
    uplink.TimeStamp                = readUInt32(11,bytes);
    uplink.time                     = GetTime(uplink.TimeStamp);

    return uplink;
}

/* ****************************************
 * bytes to number
 ****************************************/
function readUInt16LE(bytes) {
    var value = (bytes[2] << 8) + bytes[3];
    return value & 0xffff;
}

function readInt16LE(bytes) {
    var ref = readUInt16LE(bytes);
    return ref > 0x7fff ? ref - 0x10000 : ref;
```

```
}

function readUInt32(index,bytes) {
	var value = (bytes[index] << 24 & 0xff000000) | (bytes[index + 1] << 16 & 0x00ff0000)	|
(bytes[index + 2] << 8 & 0x0000ff00) | (bytes[index + 3] & 0x000000ff);

	return value & 0xffffffff;
}


function readUInt16(index,bytes) {
	var value = ((bytes[index] << 8) & 0xff00) | ((bytes[index + 1]) & 0x00ff);

	return value & 0xffff;
}

function readUInt8(index,bytes) {
	var value = ((bytes[index] ) & 0xff);

	return value & 0xff;
}
/* ******************************************
 * timestamp to real time
 *****************************************/
function GetTime(timestamp){
   var date = new Date(parseInt(timestamp)*1000);
   var Year = date.getFullYear();
   var Moth = (date.getMonth() + 1 < 10 ? '0' + (date.getMonth() + 1) : date.getMonth() + 1);
   var Day = (date.getDate() < 10 ? '0' + date.getDate() : date.getDate());
   var Hour = (date.getHours() < 10 ? '0' + date.getHours() : date.getHours());
   var Minute = (date.getMinutes() < 10 ? '0' + date.getMinutes() : date.getMinutes());
   var Sechond = (date.getSeconds() < 10 ? '0' + date.getSeconds() : date.getSeconds());
   var	GMT =	Year + '-' + Moth + '-' + Day + '	'+ Hour +':'+ Minute	+ ':' + Sechond;
   return GMT;
}


/* ******************************************
 * Base64 to hex
 *****************************************/
var Base64table = {};
Base64table = [
		'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P',
		'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
```

```
            'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
            'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '+', '/'
        ]
function Base64toHex(base64Str) {
            var i = 0;
            var bin = [];
            var x = 0, code = 0, eq = 0;
            while (i < base64Str.length) {
                var c = base64Str.charAt(i++);
                var idx = Base64table.indexOf(c);
                if (idx == -1) {
                    switch (c) {
                        case '=': idx = 0; eq++; break;
                        case ' ':
                        case '\n':
                        case "\r":
                        case '\t':
                            continue;
                        default:
                            throw { "message": "\u0062\u0061\u0073\u0065\u0036\u0034\u002E\u0074\u0068\u0065\u002D\u0078\u002E\u0063\u006E\u0020\u0045\u0072\u0072\u006F\u0072\u003A\u65E0\u6548\u7F16\u7801\uFF1A" + c };
                    }
                }
                if (eq > 0 && idx != 0)
                    throw { "message": "\u0062\u0061\u0073\u0065\u0036\u0034\u002E\u0074\u0068\u0065\u002D\u0078\u002E\u0063\u006E\u0020\u0045\u0072\u0072\u006F\u0072\u003A\u7F16\u7801\u683C\u5F0F\u9519\u8BEF\uFF01" };

                code = code << 6 | idx;
                if (++x != 4)
                    continue;
                bin.push(code >> 16);
                bin.push(code >> 8 & 0xff);
                bin.push(code & 0xff)
                code = x = 0;
            }
            if (code != 0)
                throw { "message": "\u0062\u0061\u0073\u0065\u0036\u0034\u002E\u0074\u0068\u0065\u002D\u0078\u002E\u0063\u006E\u0020\u0045\u0072\u0072\u006F\u0072\u003A\u7F16\u7801\u6570\u636E\u957F\u5EA6\u9519\u8BEF" };
```

```
        if (eq == 1)
            bin.pop();
        else if (eq == 2) {
            bin.pop();
            bin.pop();
        } else if (eq > 2)
            throw { "message":
"\u0062\u0061\u0073\u0065\u0036\u0034\u002E\u0074\u0068\u0065\u002D\u0078\u002E\
u0063\u006E\u0020\u0045\u0072\u0072\u006F\u0072\u003A\u7F16\u7801\u683C\u5F0F\u9
519\u8BEF\uFF01" };

        return bin;

}
```