Decoder LoRaWAN – plug bluetooth gateway (TTN/Chirpstack)

Decoders · https://www.lansitec.com/es/docs/decoders/decodificador-lorawan-enchufe-bluetooth-puerta-de-enlace-ttn-chirpstack/

decode-plug-bluetooth-gateway-ttn_chirpstackv4.jsDownload

Below the full decoder code:

```
// Decode uplink function.
//
// Input is an object with the following fields:
// - bytes = Byte array containing the uplink payload, e.g. [255, 230, 255, 0]
// - fPort = Uplink fPort.
// - variables = Object containing the configured device variables.
//
// Output must be an object with the following fields:
// - data = Object representing the decoded payload.

// Plug Bluetooth Gateway
function decodeUplink(input) {
  // bytes
  var bytes = input.bytes;
  // type
  var uplinkType = (bytes[0] >> 4) & 0x0f;

  switch (uplinkType) {
    case 0x04:
      return { data: decodeRegistration(bytes) };

    case 0x02:
      return { data: decodeHeartbeat(bytes) };

    case 0x06:
      return { data: decodeBeacon(bytes) };

    case 0x0e:
      return { data: decodeAlarmBeaconList(bytes) };

    case 0x0f:
```

```
      return { data: decodeAcknowledgment(bytes) };

    default:
      return null;
  }
}

// type: 0x4 Registration
function decodeRegistration(bytes) {
  var data = {};
  data.type = "Registration";
  // adr
  data.adr = ((bytes[0] >> 3) & 0x1) == 0 ? "OFF" : "ON";
  // mode
  var modeValue = bytes[0] & 0x07;
  if (modeValue == 0x01) {
    data.mode = "AU915";
  } else if (modeValue == 0x03) {
    data.mode = "CN470";
  } else if (modeValue == 0x04) {
    data.mode = "AS923";
  } else if (modeValue == 0x05) {
    data.mode = "EU433";
  } else if (modeValue == 0x06) {
    data.mode = "EU868";
  } else if (modeValue == 0x07) {
    data.mode = "US915";
  }
  // smode
  var smodeValue = bytes[1];
  if (smodeValue == 0x01) {
    data.smode = "AU915";
  } else if (smodeValue == 0x04) {
    data.smode = "CN470";
  } else if (smodeValue == 0x08) {
    data.smode = "AS923";
  } else if (smodeValue == 0x10) {
    data.smode = "EU433";
  } else if (smodeValue == 0x20) {
    data.smode = "EU868";
  } else if (smodeValue == 0x40) {
    data.smode = "US915";
  }
  // power
```

```
  data.power = ((bytes[2] >> 3) & 0x1f) + "dBm";
  // continuousBleReceiveEnable
  data.continuousBleReceiveEnable =
    ((bytes[2] >> 1) & 0x1) == 0 ? "Disable" : "Enable";
  // powerDownFlag
  data.powerDownFlag =
    (bytes[2] & 0x1) == 0 ? "Normal status" : "Automatic shutdown status";
  // dr
  data.dr = (bytes[3] >> 4) & 0x0f;
  // deviceType
  var deviceType = (bytes[3] >> 1) & 0x02;
  if (deviceType == 0) {
    data.deviceType = "OutdoorGateway";
  } else if (deviceType == 1) {
    data.deviceType = "IndoorGateway";
  } else if (deviceType == 2) {
    data.deviceType = "PlugGateway";
  }
  // rssiSortMethod
  data.rssiSortMethod =
    (bytes[3] & 0x01) == 0 ? "SortByAverage" : "SortByTheMaximumValue";
  // positionReportInterval
  data.positionReportInterval =
    (((bytes[4] << 8) & 0xff00) | (bytes[5] & 0xff)) * 5 + "s";
  // bleReceivingDuration
  data.bleReceivingDuration = bytes[6] * 30 + "s";
  // heartbeatInterval
  data.heartbeatInterval = bytes[7] * 30 + "s";
  // beaconQTY
  data.beaconQTY = (bytes[8] & 0xff) + "s";
  // crc
  data.crc = ((bytes[9] << 8) & 0xff00) | (bytes[10] & 0xff);
  return data;
}

// type: 0x2 Heartbeat
function decodeHeartbeat(bytes) {
  var data = {};
  // type
  data.type = "Heartbeat";
  // battery
  data.battery = bytes[1] + "%";
  // rssi
  data.rssi = bytes[2] * -1 + "dBm";
```

```
  // snr
  data.snr = (((bytes[3] << 8) & 0xff00) | (bytes[4] & 0xff)) / 100 + "dB";
  // version
  data.version = bytes[5] & 0xff;
  // chargeState
  var chargeState = bytes[6] & 0xff;
  if (chargeState == 0x00) {
    data.chargeState = "Not charging";
  } else if (chargeState == 0x50) {
    data.chargeState = "Charging";
  } else if (chargeState == 0x60) {
    data.chargeState = "Charging completed";
  }
  // crc
  data.crc = ((bytes[7] << 8) & 0xff00) | (bytes[8] & 0xff);

  return data;
}

// type: 0x6 Beacon
function decodeBeacon(bytes) {
  var data = {};
  data.type = "Beacon";
  data.length = bytes[0] & 0x0f;
  for (let i = 0; i < data.length; i++) {
    var index = 1 + 6 * i;
    var uuidTailNumber = bytes[index];
    var major = ((bytes[index + 1] << 8) | bytes[index + 2])
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var minor = ((bytes[index + 3] << 8) | bytes[index + 4])
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var rssi = bytes[index + 5] - 256 + "dBm";

    data["uuidTailNumber" + (i + 1)] = uuidTailNumber;
    data["beacon" + (i + 1)] = major + minor;
    data["rssi" + (i + 1)] = rssi;
  }

  return data;
}
```

```
// type: 0xe AlarmBeaconList
function decodeAlarmBeaconList(bytes) {
  var data = {};
  data.type = "AlarmBeaconList";
  data.length = bytes[0] & 0x0f;
  for (let i = 0; i < data.length; i++) {
    var index = 1 + 6 * i;
    var uuidTailNumber = bytes[index];
    var major = ((bytes[index + 1] << 8) | bytes[index + 2])
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var minor = ((bytes[index + 3] << 8) | bytes[index + 4])
      .toString(16)
      .toUpperCase()
      .padStart(4, "0");
    var rssi = bytes[index + 5] - 256 + "dBm";

    data["uuidTailNumber" + (i + 1)] = uuidTailNumber;
    data["beacon" + (i + 1)] = major + minor;
    data["rssi" + (i + 1)] = rssi;
  }

  return data;
}

// type: 0xf Acknowledgment
function decodeAcknowledgment(bytes) {
  var data = {};
  data.type = "Acknowledgment";
  data.result = (bytes[0] & 0x0f) == 0 ? "Success" : "Failure";
  data.msgId = (bytes[1] & 0xff).toString(16).toUpperCase();

  return data;
}
```