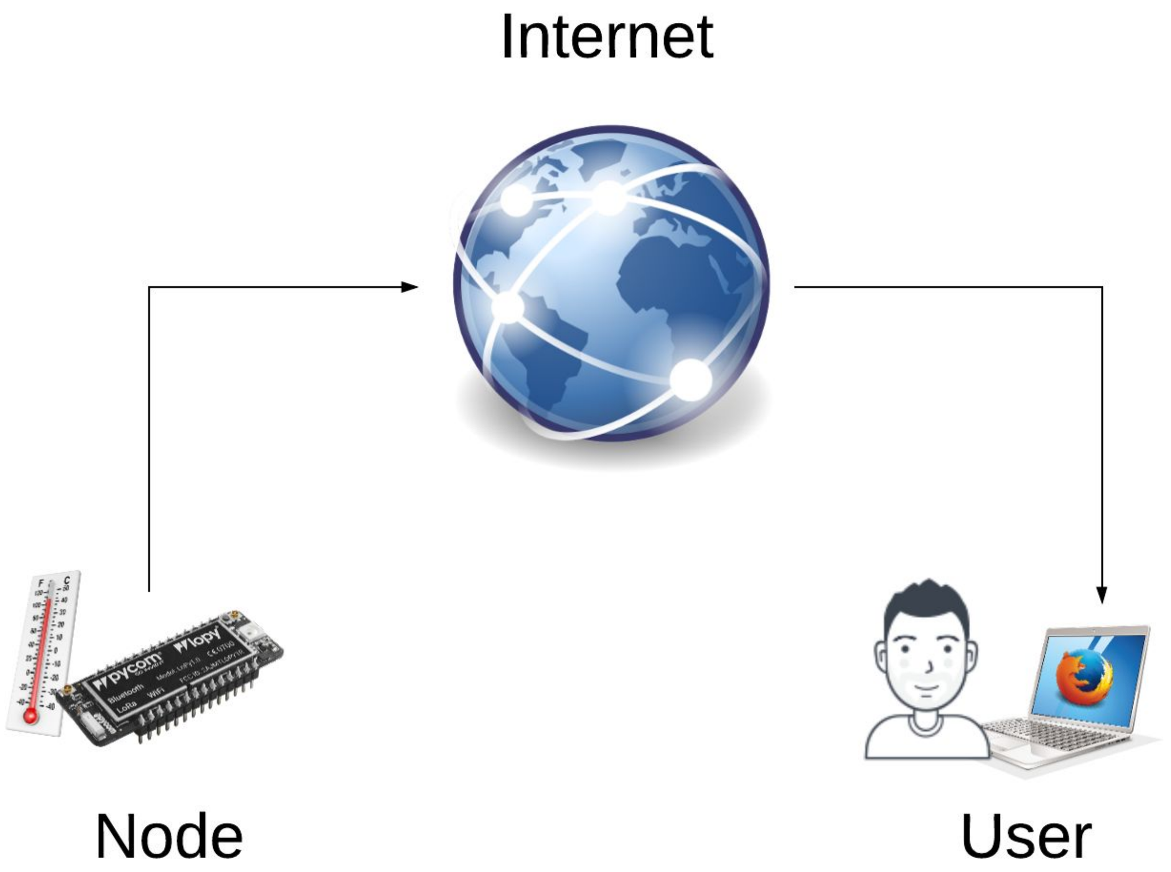
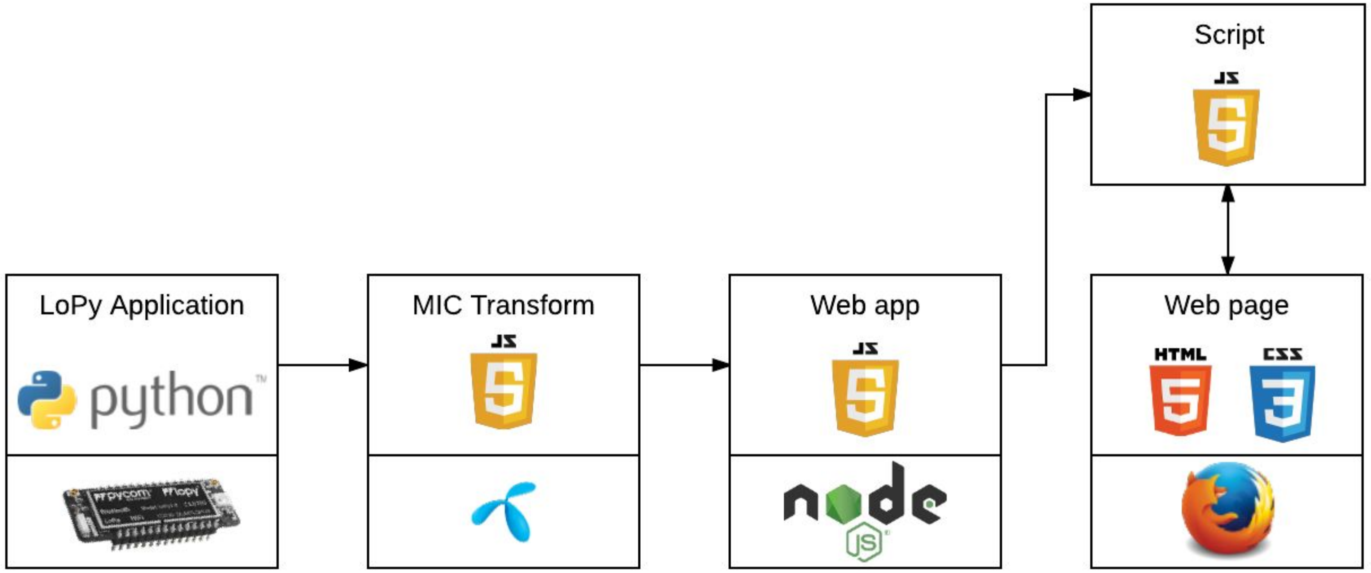
telenor start iot

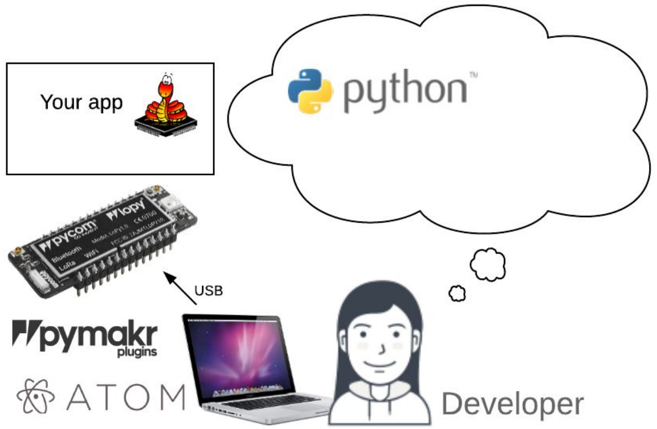
Powered by Wireless Trondheim


Internet
Node
User

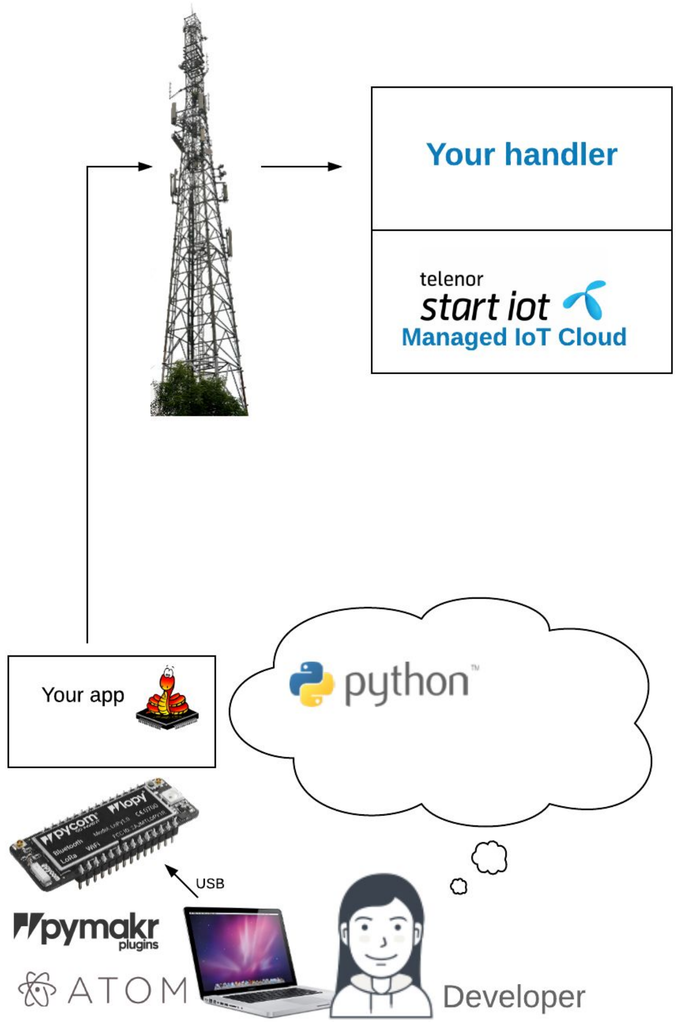
Code


Script
JS
LoPy Application
python
MIC Transform
JS
Web app
JS
node JS
Web page
HTML
CSS

Node



Handler

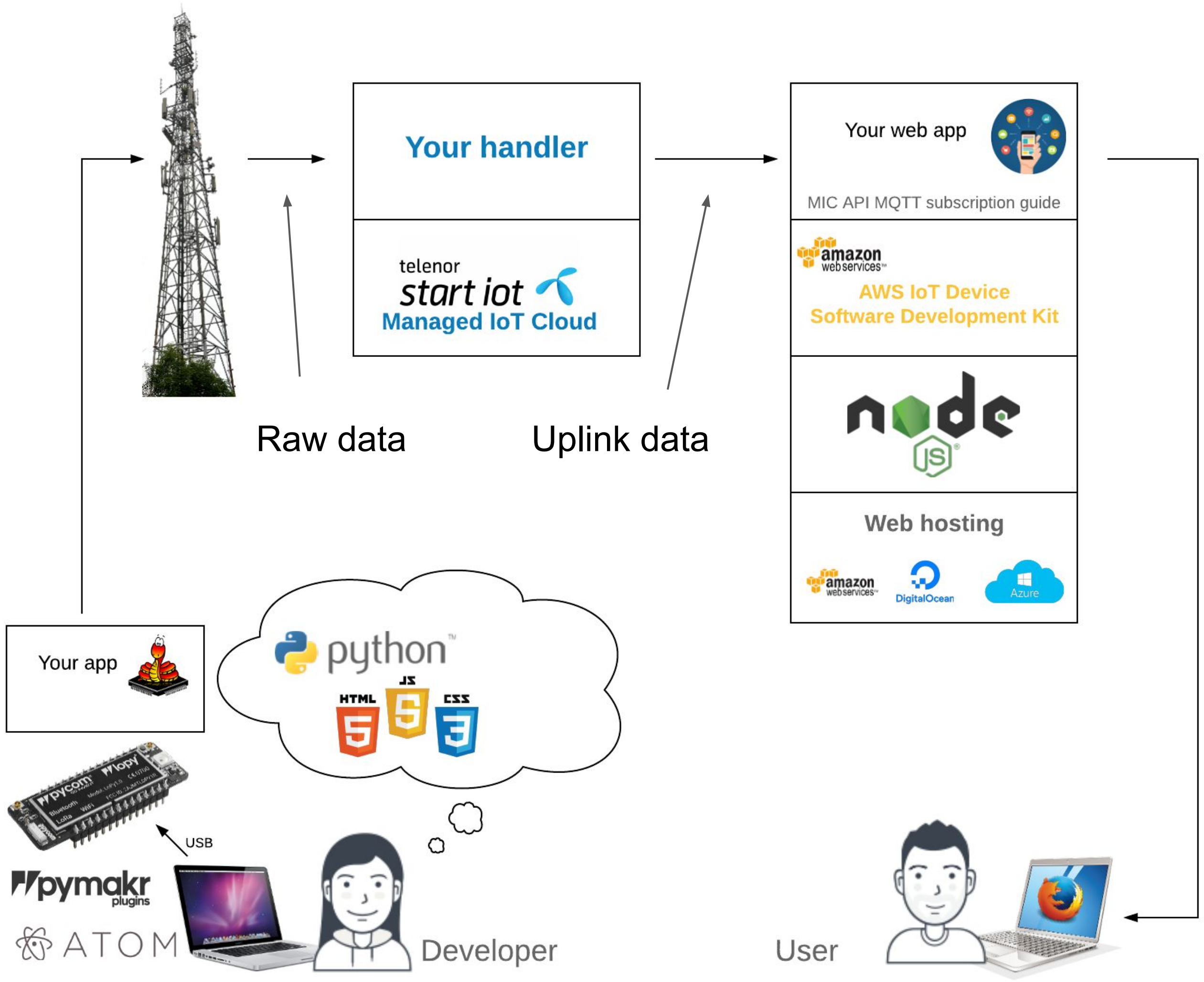

Your handler
telenor
start iot
Managed IoT Cloud
Your app
python
USB
pymakr plugins
ATOM
Developer

Device EUI

Use REPL for this

```
import network
import binascii
binascii.hexlify(network.LoRa().mac())
```

Application



LoPy code

```python
from startiot import Startiot
import pycom
import time

pycom.heartbeat(False)      # disable the blue blinking
iot = Startiot()

print("Connecting....")
pycom.rgbled(0x0F0000)      # Red light when not connected
iot.connect()
pycom.rgbled(0x00000F)      # Blue light when connected

count = 0

while True:
  print("Send data...",  count)
  data = "TEMP,%s" % (count)
  count = count + 1
  iot.send(data)
  time.sleep(60)
```

Transform

Name is predefined in MIC.

This is the raw data sent from LoPy

```
var variables = payload.toString('ascii').split(',');

return {
      'temperature'  : variables[1]
};
```

Transform

Raw data (string)

```
TEMP,49
```

Uplink data (data structure)

```
{
    state: {
        reported: {
            tcxn: {
                connection_status: 2,
                cellular: {
                    rssi: 16
                }
            } ,
            lsnr : -0.5,
            latlng : ‘63.4184,10.4002’,
            temperature: 49
        }
    } ,
    preventMessageRepublish : true
}
```

Web app

```
var awsIot = require('aws-iot-device-sdk');
var io = require('socket.io')(3000);

var thingName = '00000901'; // Replace with your own thing name

var device = awsIot.device({
   keyPath: './certs/privkey.pem',
  certPath: './certs/cert.pem',
    caPath: './certs/ca.pem',
  clientId: thingName,
      host: 'a31ovqfkmg1ev8.iot.eu-west-1.amazonaws.com'
});

device.on('connect', function() {
  console.log('Client connected');
  device.subscribe('$aws/things/' + thingName + '/shadow/update');
});

device.on('message', function(topic, payload) {
  console.log('Message: ', topic, payload.toString());

  // Broadcast the message to any connected socket clients
  io.emit('broadcast', {topic, message: payload.toString()});
});
```

Web page

```html
<html>
  <head>
      <script src="https://code.jquery.com/jquery-1.12.0.min.js"></script>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.3/socket.io.js"></script>
      <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.6.0/Chart.min.js"></script>
  </head>
  <body>
      <canvas id="output" width="600" height="250"></canvas>

      <script> // Include script from Script slide
      </script>
  </body>
</html>
```

Script

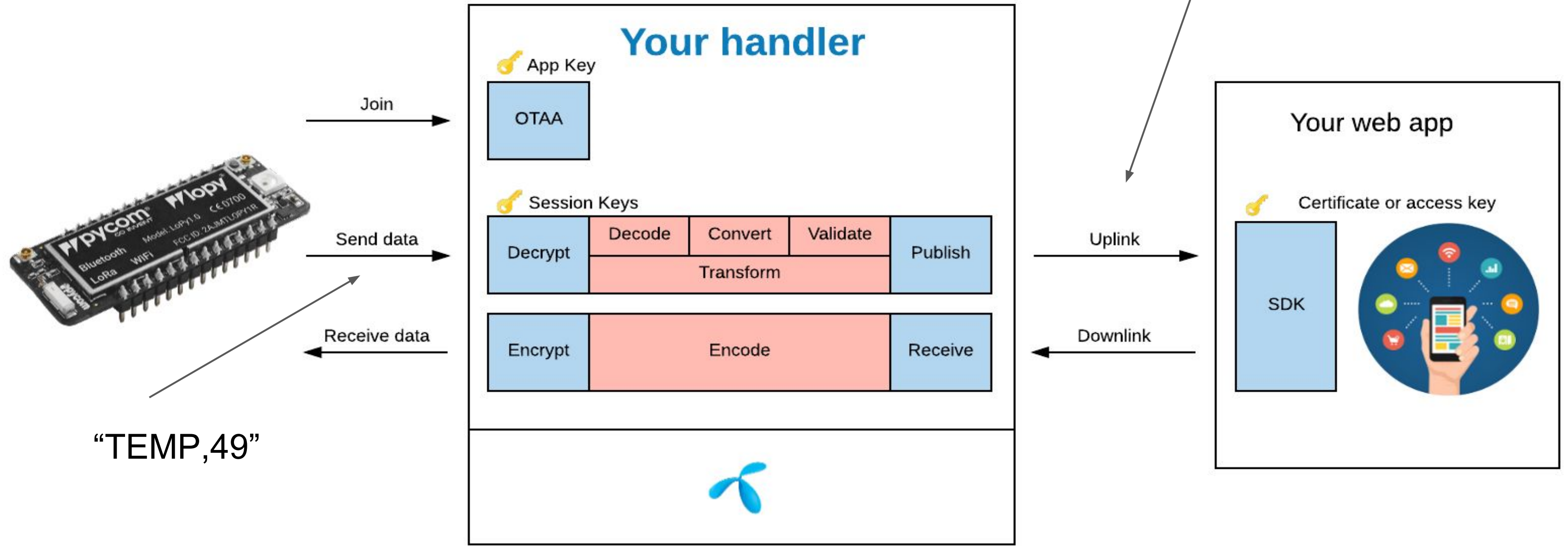
```
// Create the Chartjs element
var ctx = document.getElementById('output').getContext('2d');

var myChart = new Chart(ctx, {
                     type: 'line',
                     data: {
                         labels: [],
                         datasets: [{
                             label: 'Temperature',
                             data: []
                         }]
                     }
                 });

// Function to dynamically add data to the chart and update it
function addData(chart, label, data) {
    chart.data.labels.push(label);
    chart.data.datasets.forEach((dataset) => {
        dataset.data.push(data);
    });

    chart.update();
}

// Init Socket.io and add data to chart when broadcasted
var socket = io('http://localhost:3000');

socket.on('broadcast', function(data) {
    var payload = JSON.parse(data.message);
    addData(myChart, new Date(), payload.state.reported.temperature);
});
```

MIC data structure

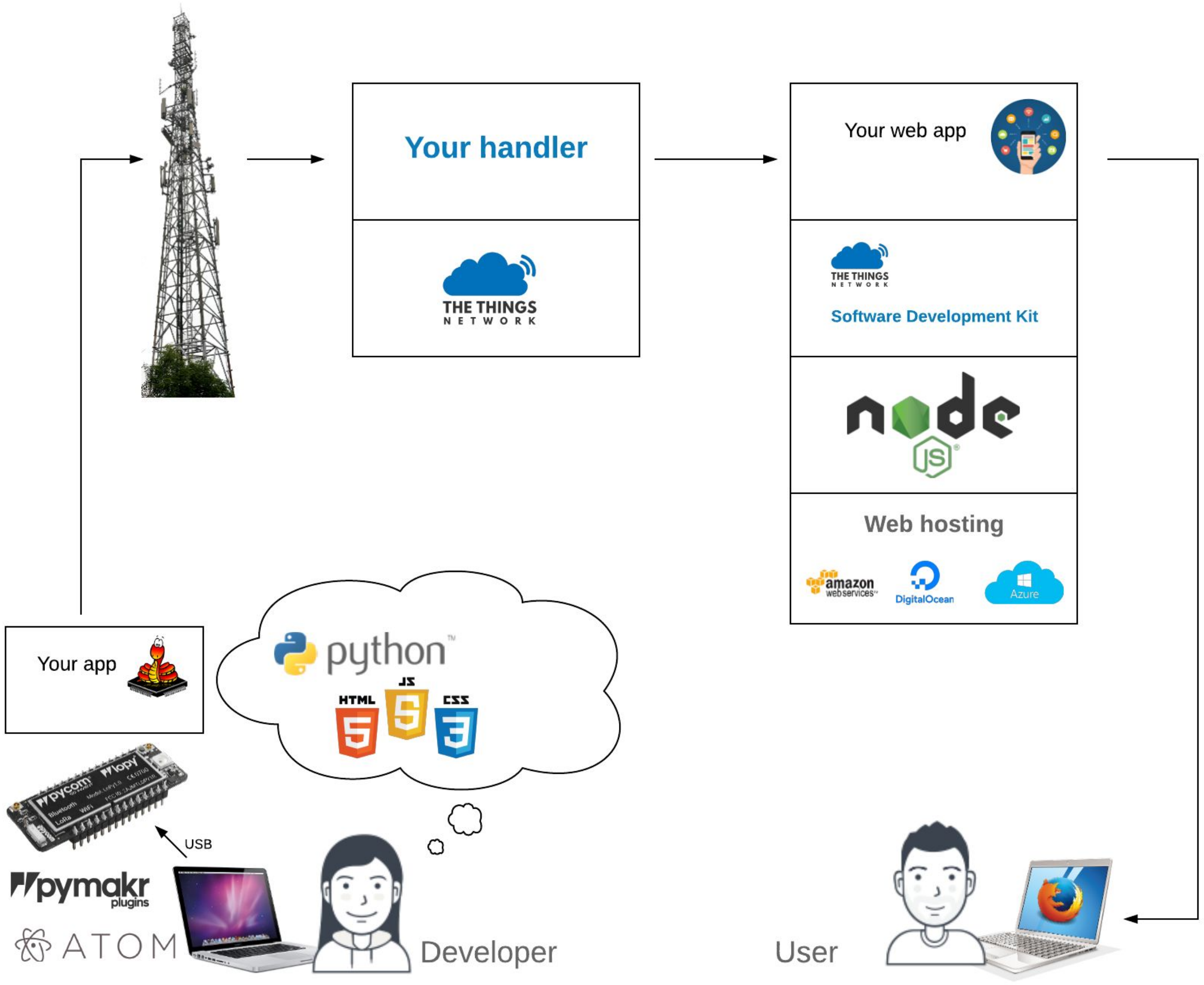
Security

MQTT: Uplink data structure

“TEMP,49”



TTN


Your handler
THE THINGS NETWORK
Your web app
THE THINGS NETWORK
Software Development Kit
node JS
Web hosting
amazon web services
DigitalOcean
Azure
Your app
python
HTML
JS
CSS
pycom
lopy
USB
pymakr plugins
ATOM
Developer
User

Data

MIC data structure

```
{
   state: {
      reported: {
         tcxn: {
            connection_status: 2,
            cellular: {
               rssi: 16
            }
         } ,
         lsnr : -0.5,
         latlng : ‘63.4184,10.4002’,
         temperature: 49
      }
   } ,
   preventMessageRepublish : true
}
```

TTN data structure

```
{
   app_id: 'kakemonster',
   dev_id: 'malopy',
   hardware_serial: '70B3D5499A4CE82A',
   port: 2,
   counter: 5892,
   payload_raw: <Buffer 54 45 4d 50 2c 35 38 39 31>,
   payload_fields: {
      temperature: '5891'
   },
   metadata: {
      time: '2017-10-10T11:34:50.160318598Z',
      frequency: 867.9,
      modulation: 'LORA',
      data_rate: 'SF7BW125',
      coding_rate: '4/5',
      gateways: [ [Object] ]
   }
}
```

TTN gateways

```
[ {
    gtw_id: 'trt-samf-loragw01',
    gtw_trusted: true,
    timestamp: 3869682171,
    time: '2017-10-10T11:37:47Z',
    channel: 4,
    rssi: -118,
    snr: -9.25,
    rf_chain: 0,
    latitude: 63.422485,
    longitude: 10.395755,
    altitude: 20
  } , {
    gtw_id: 'eui-008000000000bc6c',
    timestamp: 4068030371,
    time: '2017-10-10T11:33:13.36681Z',
    channel: 4,
    rssi: -115,
    snr: -4.2,
    rf_chain: 0,
    latitude: 63.42883,
    longitude: 10.3857,
    altitude: 21
  } ]
```