AN2202
Application note

Getting started guide for *u-blox MAX-M10S* GNSS module mounted on *Move-X Cicerone* board

Introduction

The purpose of this document is to explain the usage of *u-blox* library provided along with the *Move-Xduino* Arduino support package available at: https://www.github.com/Move-X/Move-Xduino).

Mounted on the Cicerone board there is a GNSS *MAX-M10S* module; please visit https://www.u-blox.com/en/product/max-m10-series if you want further details and documentation of the module.

From this application note you will learn how the GNSS module interacts with *Move-X MAMWLE*, how it is connected and some useful examples to start with.

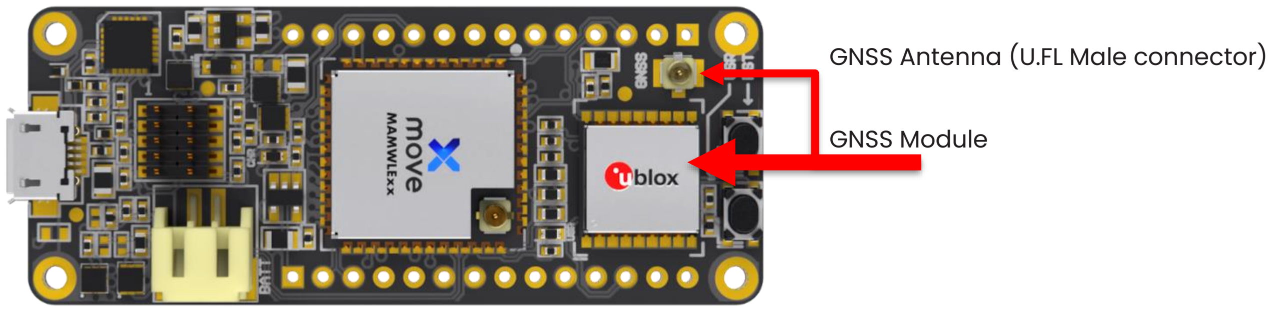
For more resources, news, and links to official distributors for *Move-X* products, please visit *Move-X* website: https://www.move-x.it/.

For details on the Cicerone boards, documentation and buying options please visit the official website: https://www.move-x.it/cicerone-board/.

Contents

u-blox MAX-M10S GNSS module

The *MAX-M10S* is a standard precision GNSS module that provides excellent sensitivity and acquisition time for constellations in GNSS L1 band such as *GPS, GLONASS, Galileo, BeiDou, etc*.



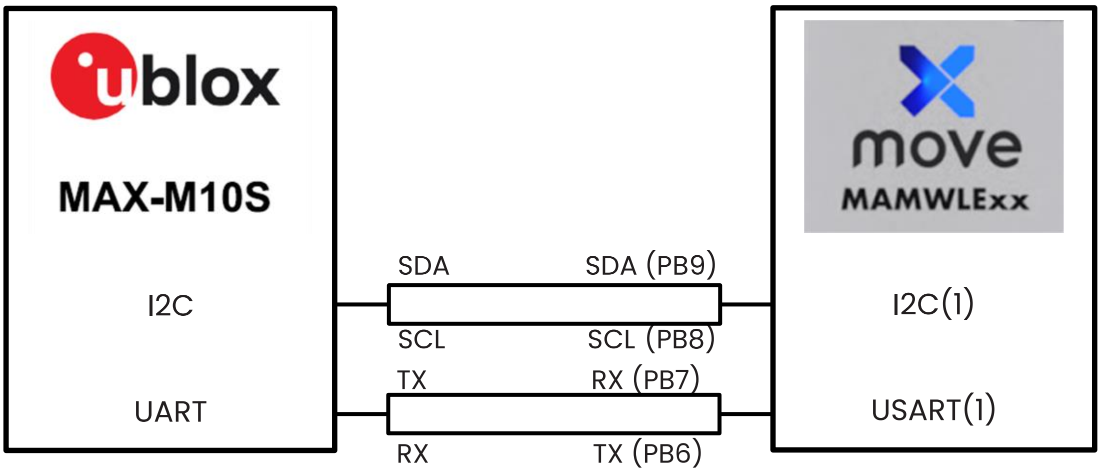
The *MAX-M10S* module integrates an LNA + SAW filter and so it is suitable for passive antennas, and thanks to its low power consumption is a perfect fit for batty-powered applications.

For more information about the MAX-M10 please visit:
https://www.u-blox.com/en/product/max-m10-series

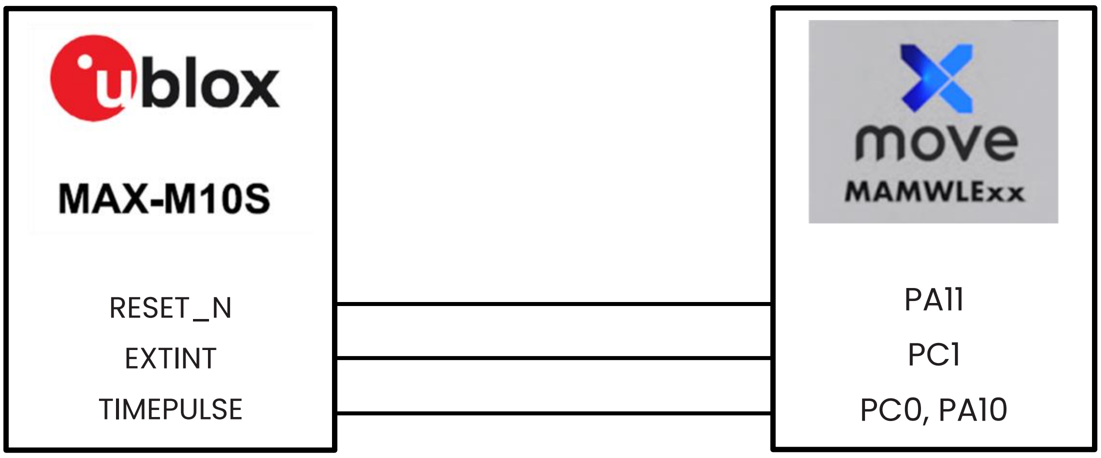
To use the GNSS module you will need a **Passive** Antenna for L1 (1 – 2 GHz) frequency band with a U.FL Female connector.

Communication with *Move-X MAMWLE*

The *MAX-M10S* module talks with microcontrollers using Asynchronous Serial (UART) and/or Synchronous Serial (I2C) communication supporting UBX (proprietary binary *u-blox* messages) and *NMEA* (standard ASCII navigation messages).
On *Cicerone* board *u-blox* module is fully connected so you can use I2C or UART as you prefer.
Following is a simplified scheme of connections, with pin names of both modules:



This GNSS module offers others GPIO pins that you can use: TIMEPULSE, EXTINT, RESET_N. They are connected as follow:



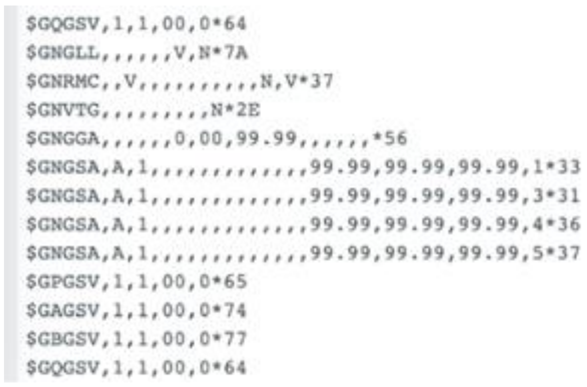
In *Arduino IDE* those ports/pins are mapped as below:

MAX-M10S	Arduino IDE
I2C	Default “Wire”
UART	Serial1
RESET_N pin	UBLOX_RESETN
EXTINT pin	UBLOX_EXTINT
TIMEPULSE pin	UBLOX_TIMEPULSE

Demo Code inside Cicerone Board

The Cicerone Board comes with a preloaded Demo Firmware for the GNSS module evaluation. The GNSS module by U-blox communicates, by default, with a serial UART interface. The preloaded firmware inside Cicerone Board let you access the GNSS UART interface from the USB connection.

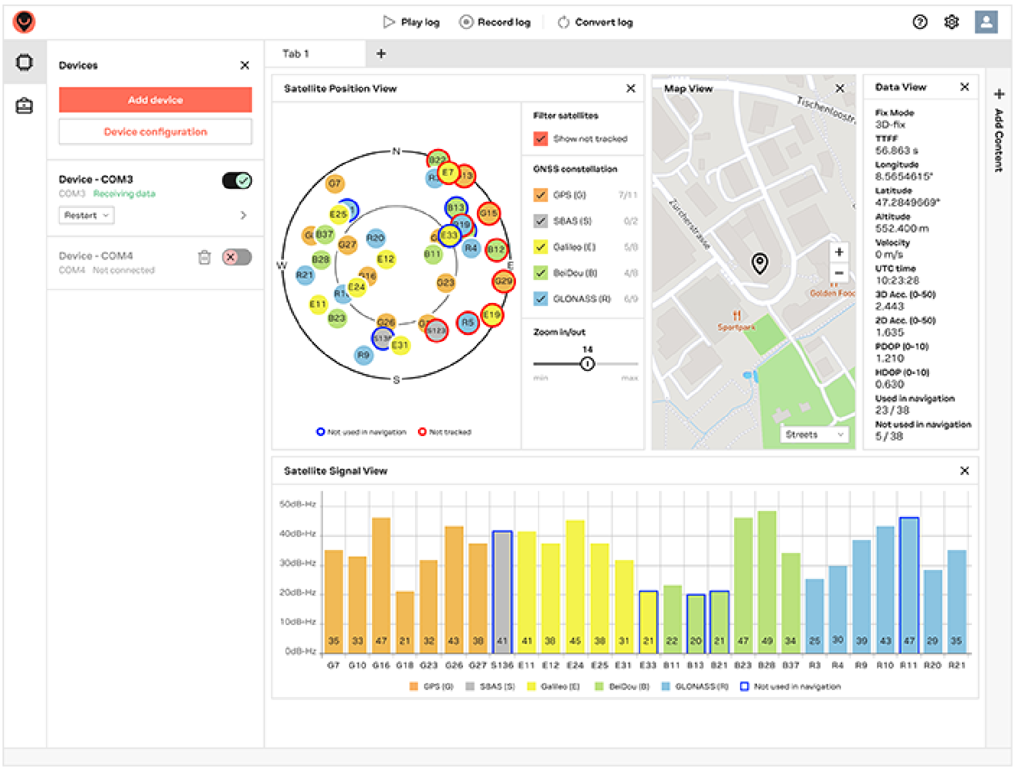
If you open a serial connection with a terminal (like the Serial Monitor in Arduino IDE), setting baudrate to 9600, you will get something like in figure.



```
$GQGSV,1,1,00,0*64
$GNGLL,,,,,,V,N*7A
$GNRMC,,V,,,,,,,,,,N,V*37
$GNVTG,,,,,,,,,N*2E
$GNGGA,,,,,,0,00,99.99,,,,,,*56
$GNGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99,1*33
$GNGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99,3*31
$GNGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99,4*36
$GNGSA,A,1,,,,,,,,,,,,,99.99,99.99,99.99,5*37
$GPGSV,1,1,00,0*65
$GAGSV,1,1,00,0*74
$GBGSV,1,1,00,0*77
$GQGSV,1,1,00,0*64
```

Those are ascii-formatted message strings of NMEA protocol. It is a standard protocol for GNSS positioning.

To evaluate better the GNSS module you can use the free *"u-center 2"* application by U-blox. You can download it from https://www.u-blox.com/en/product/u-center clicking on "Download u-center 2 – For M10 products only".



Data View

Fix Mode
3D-fix
TTFF
56.863 s
Longitude
8.5654615°
Latitude
47.2849669°
Altitude
552.400 m
Velocity
0 m/s
UTC time
10:23:28
3D Acc. (0-50)
2.443
2D Acc. (0-50)
1.635
PDOP (0-10)
1.210
HDOP (0-10)
0.630
Used in navigation
23 / 38
Not used in navigation
5 / 38

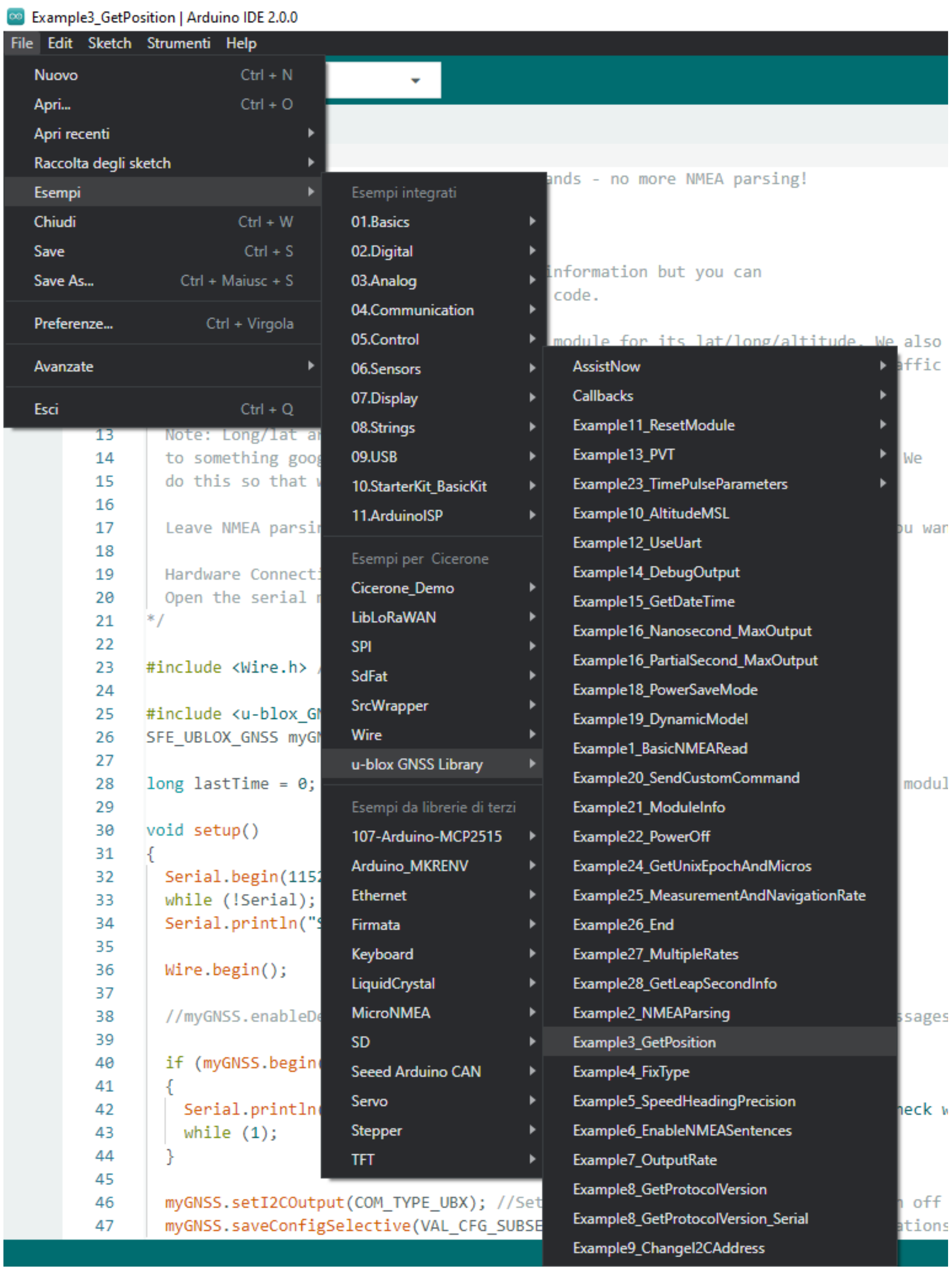
G7	G10	G16	G18	G23	G26	G27	S136	E11	E12	E24	E25	E31	E33	B11	B13	B21	B23	B28	B37	R3	R4	R9	R10	R11	R20	R21
35	33	47	21	32	43	38	41	41	38	45	38	31	21	22	20	21	47	49	34	25	30	39	43	47	29	35

If you want to re-flash this demo firmware, you can refer to the “SerialPassthrough” example of Arduino IDE from “File -> Examples -> [Built-in examples] 04.Communication”

Note: To get a position after first power connection, the GNSS module needs up to 30 seconds (in outdoor conditions)

Arduino IDE Examples

The *Move-Xduino* support provides an integrated library for *u-blox* GNSS module, with lots of available examples. You can find them under *File->Examples->u-blox GNSS Library* when *Cicerone* board is selected.

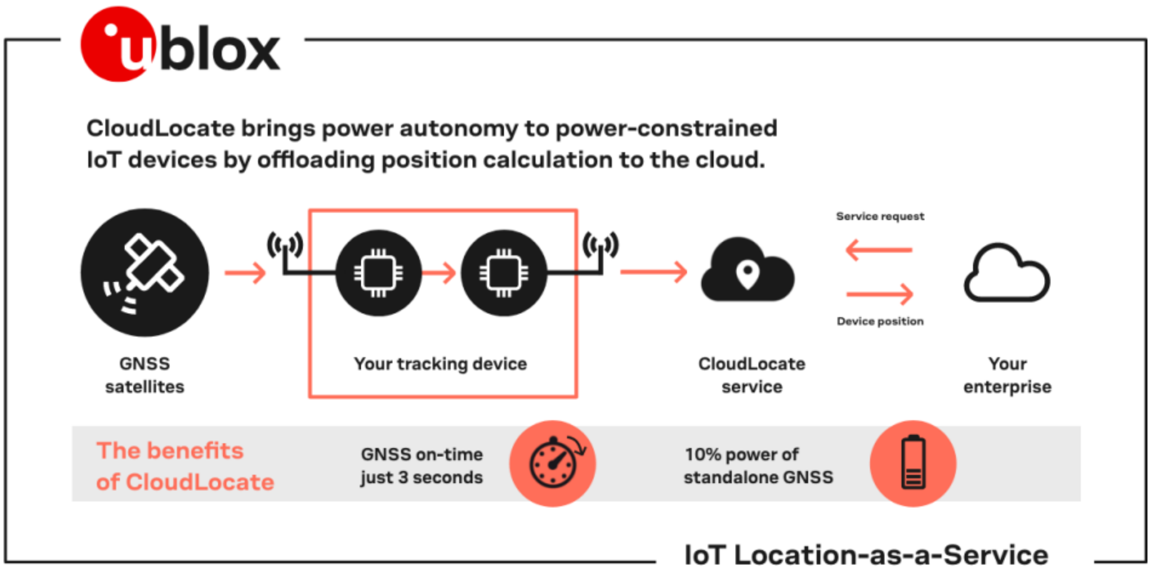


```
ands - no more NMEA parsing!

information but you can
code.

module for its lat/long/altitude. We also
affic
  Note: Long/lat ar
  to something goog
  do this so that w

  Leave NMEA parsi

  Hardware Connecti
  Open the serial m
*/

#include <Wire.h> /

#include <u-blox_GN
SFE_UBLOX_GNSS myGN

long lastTime = 0;

void setup()
{
  Serial.begin(1152
  while (!Serial);
  Serial.println("S

  Wire.begin();

  //myGNSS.enableDe

  if (myGNSS.begin(
  {
    Serial.println(
    while (1);
  }

  myGNSS.setI2COutput(COM_TYPE_UBX); //Set
  myGNSS.saveConfigSelective(VAL_CFG_SUBSE
```

For a description of the example's behavior, please look at the comments inside the code.

LoRaWAN_CloudLocate Example

The *Cicerone_Demo* example combines LoRaWAN communication and GNSS positioning. It leverages CloudLocate service by u-blox for cloud-based GNSS positioning, allowing energy saving on battery-powered devices such as Cicerone board by Move-X.
For more information, purchasing options and documentation about the **Move-X Cicerone** board please visit the website: https://www.move-x.it/cicerone-board/.


ublox
CloudLocate brings power autonomy to power-constrained IoT devices by offloading position calculation to the cloud.
GNSS satellites
Your tracking device
CloudLocate service
Service request
Device position
Your enterprise
The benefits of CloudLocate
GNSS on-time just 3 seconds
10% power of standalone GNSS
IoT Location-as-a-Service

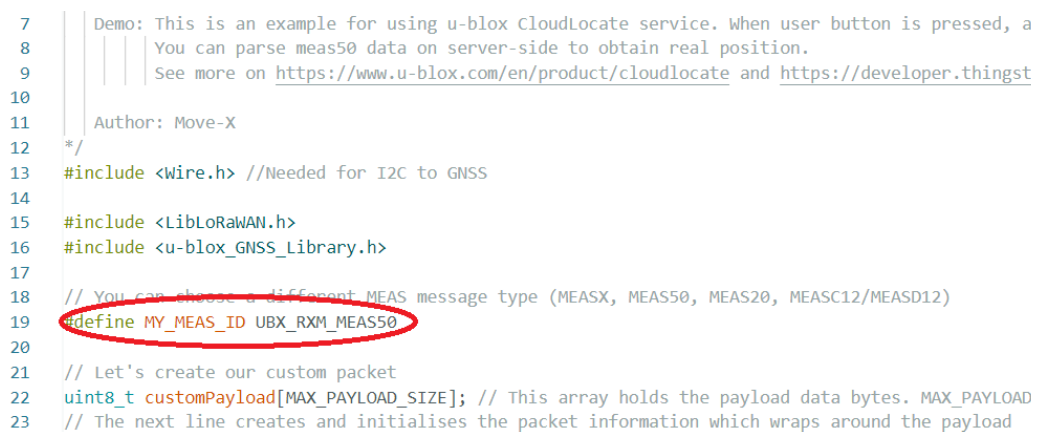
The example is based on LoRaWAN_End_Node and Example20_SendCustomCommand.
The goal is to send a compact raw measurement of tracked satellites to server application using LoRaWAN connectivity, polling UBX MEAS50 message (fixed 50 bytes payload) from GNSS module. This example aims to be a quick start to easily implement a CloudLocate device for testing and prototyping purposes.
Usage of u-blox CloudLocate service requires the user to have a *u-blox Thingstream* account and pay for the subscription. Then, from the account it is possible to register a new *Location* thing under the Location services and retrieve the credentials under device's credentials tab. The credentials can be used to access the CloudLocate position decoding APIs that allow to convert the MEAS50 data from the end device into valid coordinates. For more details about the usage and implementation of u-blox CloudLocate you can visit the following links:

- https://developer.thingstream.io/guides/location-services/cloudlocate-getting-started
- https://developer.thingstream.io/guides/location-services/cloudlocate-getting-started/compact-raw-measurements

Thanks to Move-Xduino implementation of LoRaWAN stack the example sketch is very simple and consists of the following steps:

1) A Join request is made to LoRaWAN Network Server to access the chosen LoRaWAN network. Usage of the LoRaWAN stack, network access keys usage and setup are better discussed in the AN2201 application note.
2) The user button of the Cicerone board triggers a GNSS position measurement. Related MEAS50 data is transferred from u-blox module to the MAMWLE, which is responsible of sending the data packet to the network server where the user can retrieve it through its APIs.

The MEAS50 (50 bytes long) position measurement is not the only kind of acquisition available, other options are MEAS20 (20 bytes long) and MEAS12 (12 bytes long). The rule of thumb in choosing the best compromise is that a shorter data packet translates into worse accuracy of the decoded position.
The kind of measurement used can be changed in the sketch by mean of the *MY_MEAS_ID* define.



```
  Demo: This is an example for using u-blox CloudLocate service. When user button is pressed, a
        You can parse meas50 data on server-side to obtain real position.
        See more on https://www.u-blox.com/en/product/cloudlocate and https://developer.thingst

  Author: Move-X
*/
#include <Wire.h> //Needed for I2C to GNSS

#include <LibLoRaWAN.h>
#include <u-blox_GNSS_Library.h>

// You can choose a different MEAS message type (MEASX, MEAS50, MEAS20, MEASC12/MEASD12)
#define MY_MEAS_ID UBX_RXM_MEAS50

// Let's create our custom packet
uint8_t customPayload[MAX_PAYLOAD_SIZE]; // This array holds the payload data bytes. MAX_PAYLOAD
// The next line creates and initialises the packet information which wraps around the payload
```

Note: the example does not discuss data payload retrieve from the network server because it depends on the provider chosen by the user. Please follow the network server's documentation for information about this.

For examples of MEAS50, MEAS20, MEAS12 and documentation about the APIs for implementing the decoding using CloudLocate service, please visit the following links:
- https://developer.thingstream.io/guides/location-services/cloudlocate-getting-started/compact-raw-measurements

Document revision

Revision 1.1 - 12/12/2022

Revision 1.0 - 23/09/2022