Zephyr® @ THE LINUX FOUNDATION OPEN SOURCE SUMMIT EUROPE

How to create an Asset Tracker with Zephyr and Thingsboard in no Time

Tobias Marquardt, grandcentrix

About me

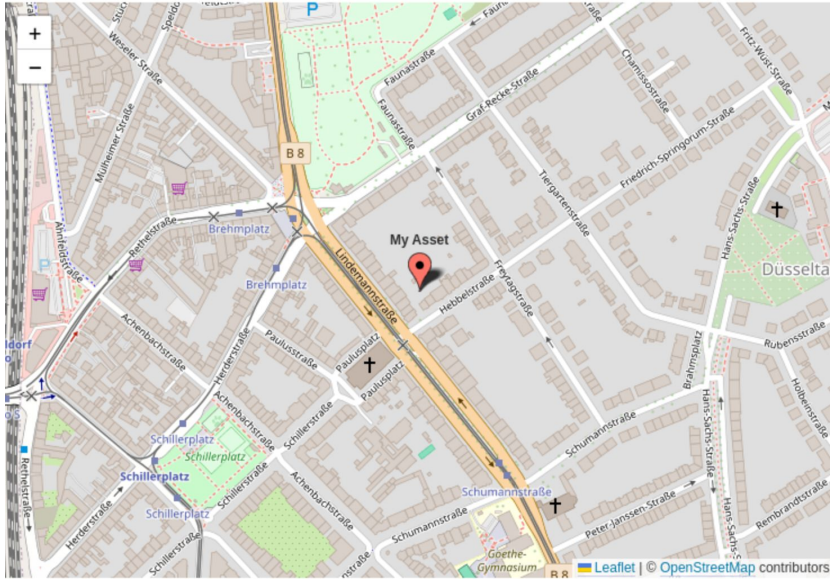
- Tobias Marquardt
- Embedded Software Developer @grandcentrix (Cologne, Germany)
- Working on IoT solutions
- Embedded Linux
- Zephyr

tobias.marquardt@grandcentrix.net

github.com/worblehat

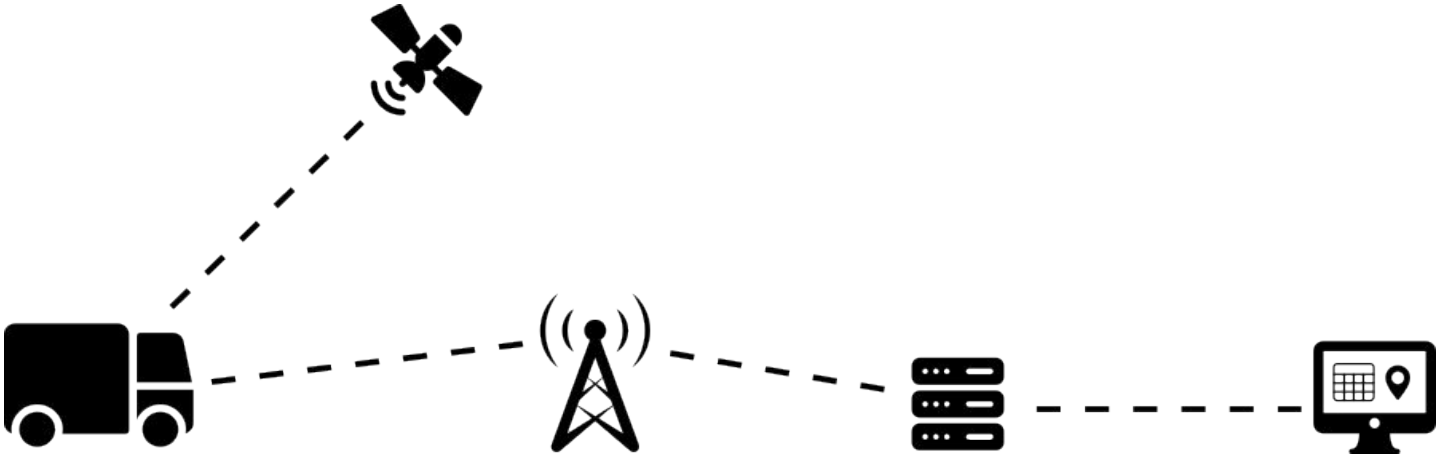
Imagine you want to build an IoT Asset Tracker

- Track position of physical asset (outdoors)
- Battery powered device
- Wireless communication
- Live monitoring
- Visualization

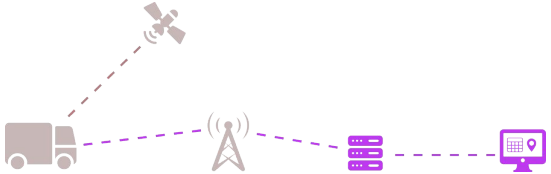


Get the sample code:
github.com/grandcentrix/ossumit-asset-tracker

What do we need?



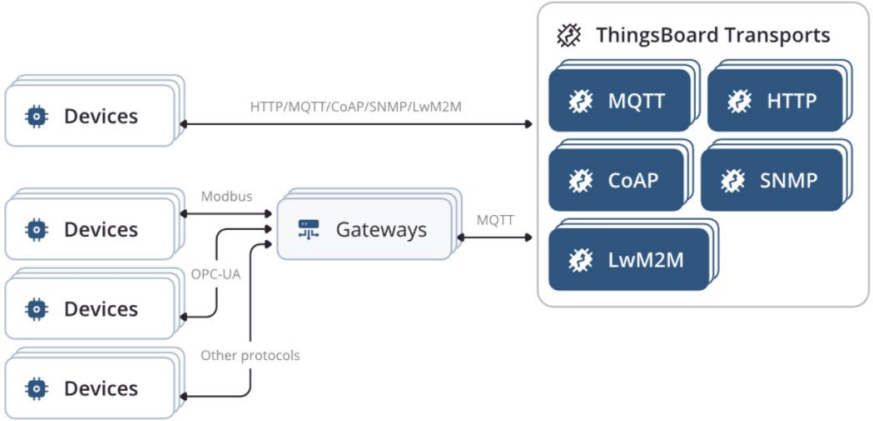
ThingsBoard



An open-source IoT Platform…

Features:

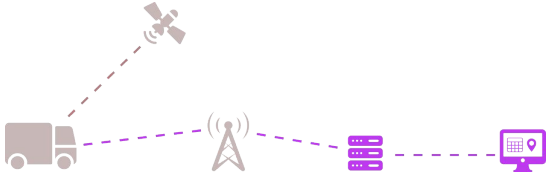
- Collect data from devices
- Visualize data with dashboards
- Device provisioning and management
- OTA updates
- And much more…

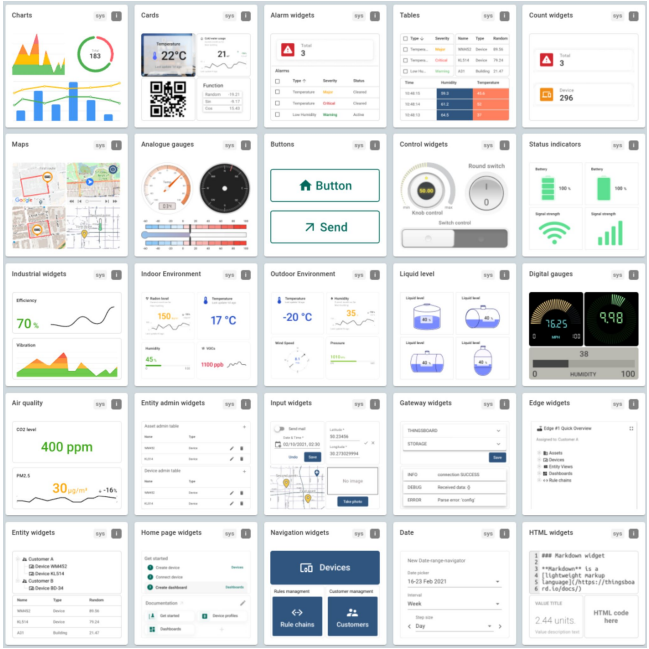


Why Thingsboard?

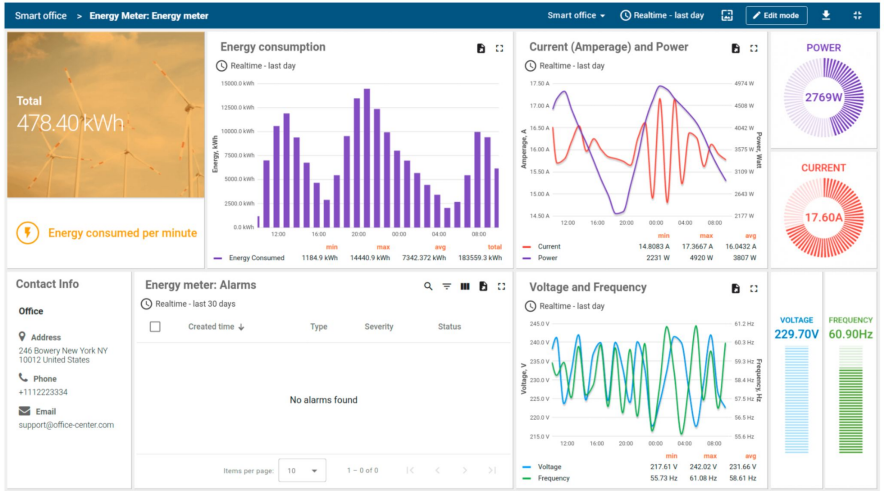
- It's open source
- It's super easy to get started
- Device API supports various protocols
- Your choice: hosted/self-hosted, cloud/on-premise

Data Visualization



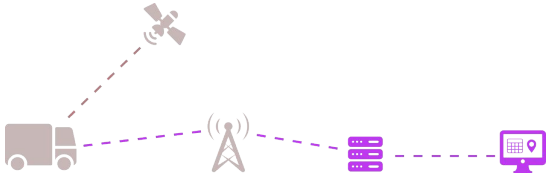


Widget Library



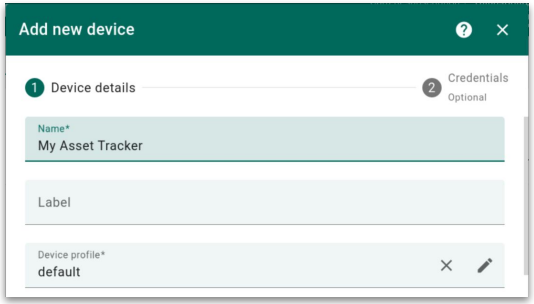
Dashboard

Prepare Thingsboard for our asset tracker

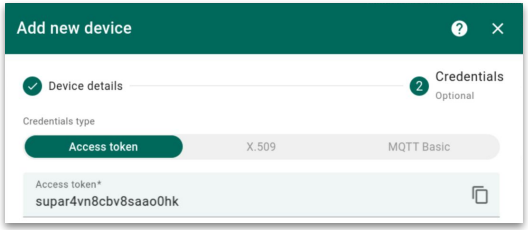


1. Login to thingsboard.cloud

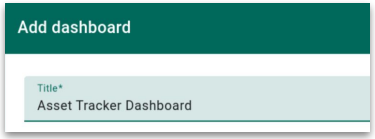
2. Create a device



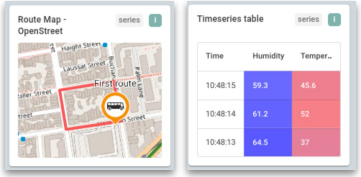
3. Get device access token



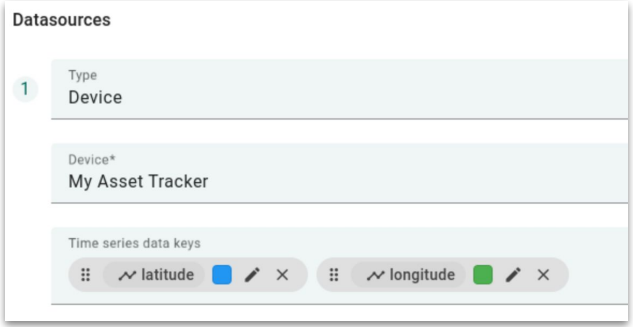
4. Create a dashboard



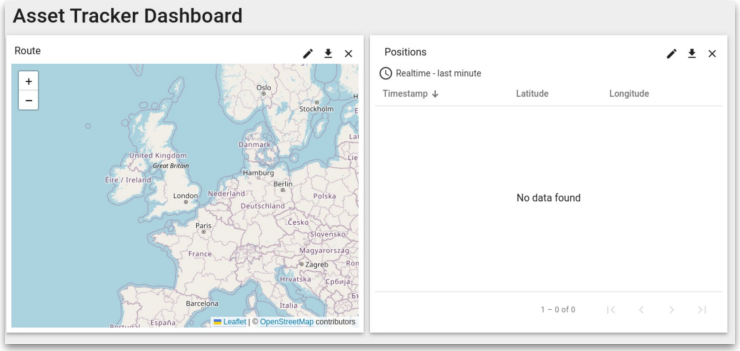
5. Select widgets



6. Configure widgets



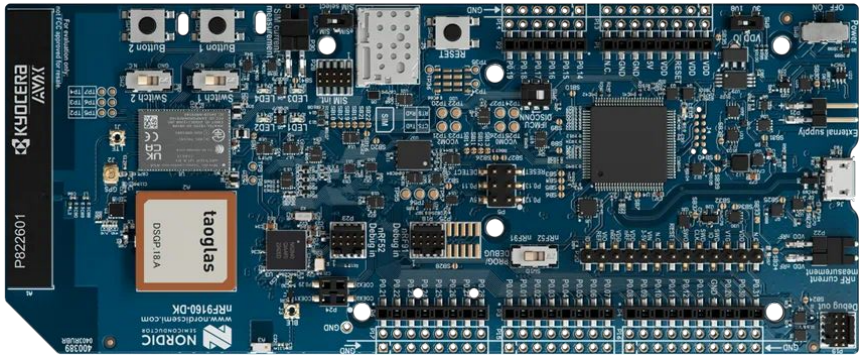
7. Place widgets on dashboard



THE LINUX FOUNDATION
OPEN SOURCE SUMMIT
EUROPE

Device

Hardware: nRF9160 DK



- ARM Cortex M33
- Modem with:
  - LTE-M / NB-IoT
  - GPS

Software: Zephyr + nRF Connect SDK

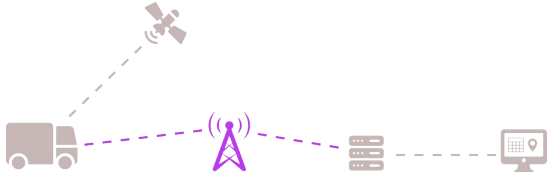
nRF9160 DK image by Nordic Semiconductor

GNSS with Zephyr

- Using **nRF Connect SDK Modem Library**
- Enable GPS (alongside LTE-M and NB-IoT):
  ```
  CONFIG_LTE_NETWORK_MODE_LTE_M_NBIOT_GPS=y
  ```
- Configure periodic navigation mode (i.e. receiver sleeps most of the time):
  ```
  nrf_modem_gnss_fix_interval_set(120);
  nrf_modem_gnss_fix_retry_set(30);
  ```
- Register event handler:
  ```
  nrf_modem_gnss_event_handler_set(handle_gps_event);
  ```
- Start GNSS:
  ```
  nrf_modem_gnss_start();
  ```
- Read data in event handler:
  ```
  if (event == NRF_MODEM_GNSS_EVT_PVT)
      nrf_modem_gnss_read(&pvt_data, sizeof(pvt_data), NRF_MODEM_GNSS_DATA_PVT);
  ```

Since Version 3.6.0 there's also a GNSS subsystem in Zephyr

LTE-M and NB-IoT



- Cellular networks for IoT Devices (LPWAN)
  ⇒ Energy efficiency, long range, building penetration, scalability

LTE-M

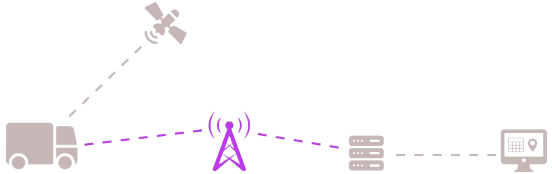
- High data rates (up to 7 mbit/s)
- Low latency (suitable for TCP, MQTT etc.)
- Supports cell handover (good for moving devices)

NB-IoT

- Low data rates (max. 159 kbit/s)
- No cell handover
- Higher latency (not suitable for TCP)
- More global coverage
- Longer range, higher building penetration

- Cellular connectivity is the weakest link in an asset tracker
  ⇒ Use everything we can get: LTE-M (preferred) <u>and</u> NB-IoT (as fallback)

LTE-M and NB-IoT with Zephyr



- Using **nRF Connect SDK Modem and Link Control Library**
- Configuration:

```
CONFIG_LTE_LINK_CONTROL=y
CONFIG_LTE_NETWORK_MODE_LTE_M_NBIOT_GPS=y
CONFIG_LTE_MODE_PREFERENCE_LTE_M=y
CONFIG_LTE_EDRX_REQ=y
CONFIG_LTE_PSM_REQ=y
```

  } Power saving features

- Enable LTE:

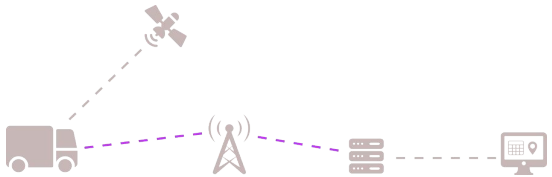
```
lte_lc_func_mode_set(LTE_LC_FUNC_MODE_ACTIVATE_LTE);
```

- Connect:

```
lte_lc_connect_async(event_handler);    // Handler for registration status events etc.
```

- Then use BSD sockets as usual…

CoAP



Remember: NB-IoT implies UDP (no TCP)

CoAP feels like HTTP but:

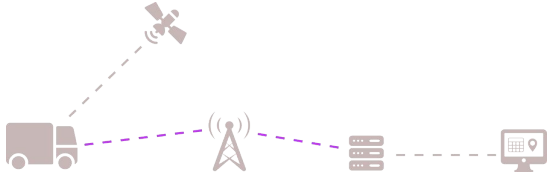
- Over UDP
- Simpler
- More compact representation
- Designed for constrained devices and low throughput networks

Reliability:

- UDP gives no guarantees
- CoAP provides an <u>optional</u>, lightweight reliability mechanism (“confirmable messages”)

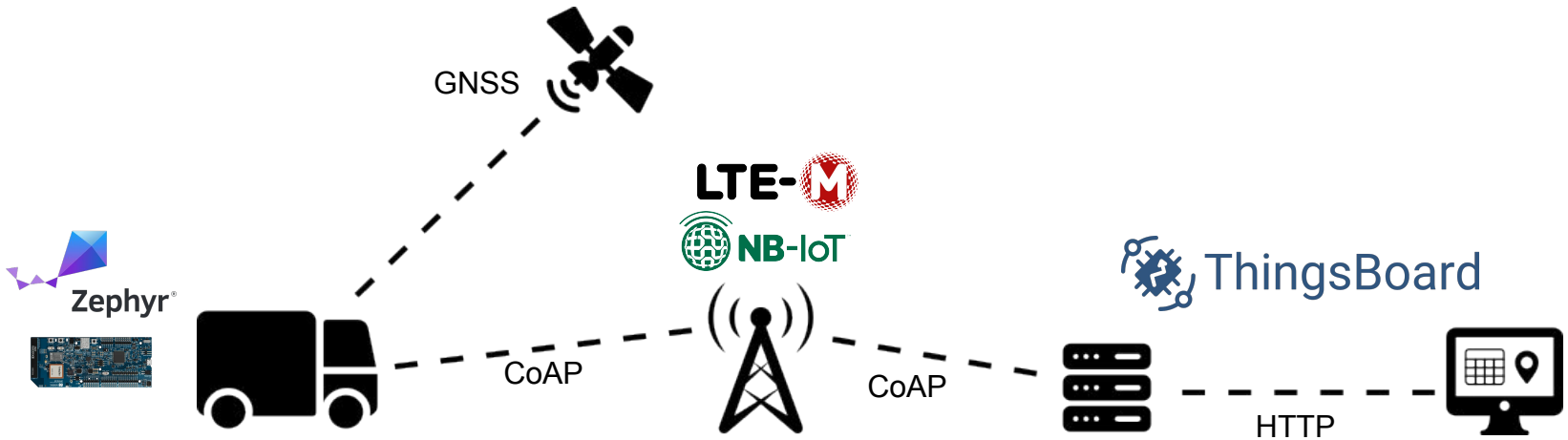
⇛ Application can decide if reliability mechanisms are needed or not

Client-side CoAP with Zephyr

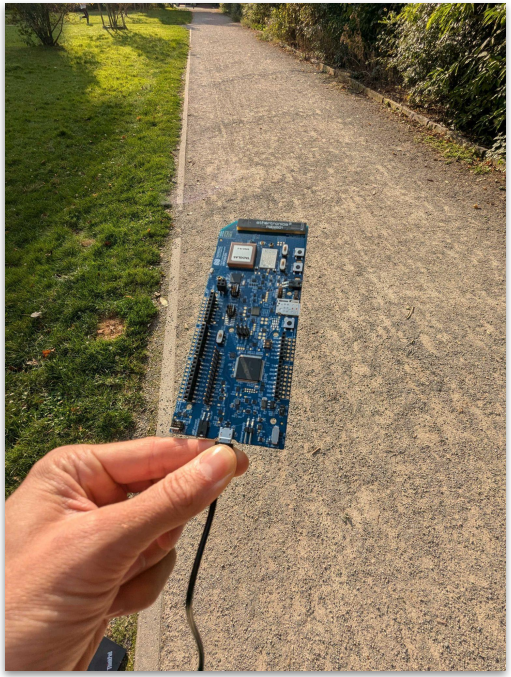


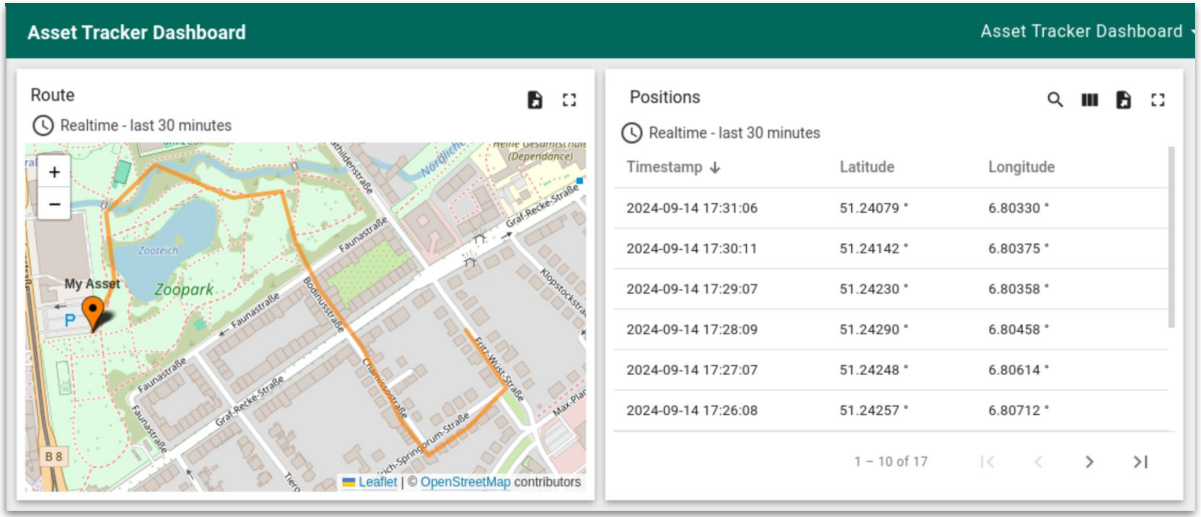
```
CONFIG_COAP=y
CONFIG_COAP_CLIENT=y
```

```
struct sockaddr_storage server_addr;
// (...) resolve hostname and set server_addr
int sock = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);

char *uri = "api/v1/sgp5r4vn8cbv8saao0hk/telemetry";
char *payload = "{\"latitude\": \"50.92950\", \"longitude\": \"6.96200\"}";

struct coap_client_request request = {
    .path = uri,
    .method = COAP_METHOD_POST,
    .confirmable = false,
    .fmt = COAP_CONTENT_FORMAT_APP_JSON,
    .payload = (uint8_t *)payload,
    .len = strlen(payload),
    .cb = response_handler
};

coap_client_req(&client, sock, (struct sockaddr *)&server_addr, &request, NULL);
```


GNSS
LTE-M
NB-IoT
Zephyr
ThingsBoard
CoAP
CoAP
HTTP

A walk in the park…






```
*** Booting nRF Connect SDK v2.7.0-5cb85570ca43 ***
*** Using Zephyr OS v3.6.99-100befc70c74 ***
[00:00:00.368,316] <inf> main: Initializing modem library
[00:00:00.605,102] <inf> nrf_modem: Modem FW version: mfw_nrf9160_1.3.5
[00:00:00.605,651] <inf> nrf_modem: Modem FW UUID: dc8e7aff-5331-409e-be02-5529dffa5537
[00:00:00.636,718] <inf> lte: Connecting to LTE network
[00:00:09.874,237] <inf> lte: Network registration status: not registered - searching
[00:00:11.997,344] <inf> lte: Network registration status: registered - roaming
[00:00:11.997,833] <inf> thingsboard: Initializing socket for coap://coap.thingsboard.cloud:5683
[00:00:38.867,431] <inf> thingsboard: Resolved coap.thingsboard.cloud to 44.196.172.187
[00:00:38.870,635] <inf> gps: Starting GPS
[00:02:23.755,676] <inf> main: Sending GPS telemetry: {"latitude":"51.2408190","longitude":"6.8108315",
[00:02:23.756,622] <dbg> gps: handle_gps_event: GPS enters sleep after achieving fix
[00:02:24.801,849] <inf> thingsboard: Received response with code 2.01
[00:02:38.885,986] <dbg> gps: handle_gps_event: GPS has woken up
[00:02:42.929,321] <inf> main: Sending GPS telemetry: {"latitude":"51.2406618","longitude":"6.8109913",
[00:02:42.930,023] <dbg> gps: handle_gps_event: GPS enters sleep after achieving fix
[00:02:43.952,423] <inf> thingsboard: Received response with code 2.01
```

Thank you for attending!

- Sample Code: https://github.com/grandcentrix/ossumit-asset-tracker
- We are also working on a Thingsboard SDK for Zephyr: https://github.com/grandcentrix/thingsboard-sdk
- If you want to build IoT solutions join our team: https://grandcentrix.team/
- Come talk to me later!