GPS-Tracker documentation

Authors

- Dmitrii Semenov
- Radoslav Tomčala

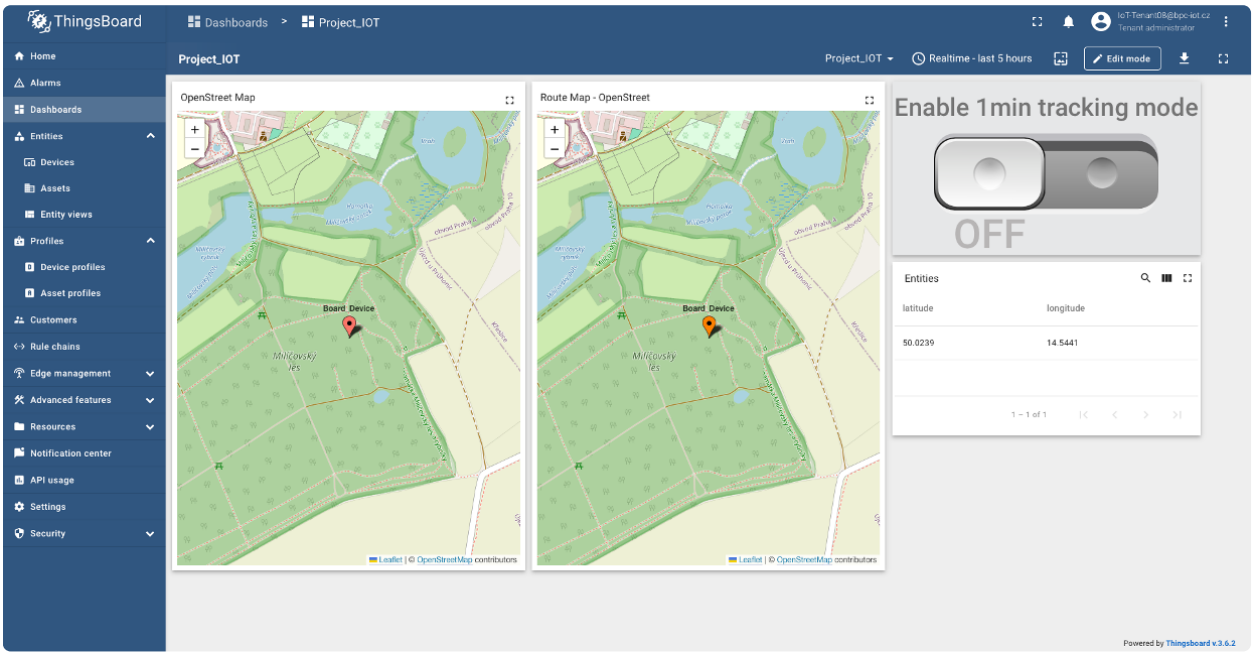
Use Case

The GPS-tracker is a portable device designed for use by delivery services to easily monitor package locations. In cases of emergency or theft, it can switch from a standard update period of 30 minutes to a quick 1-minute refresh rate. The last known location is displayed on the Thingsboard portal, allowing monitoring of the delivery. The device operates on battery power to maintain portability.

Instructions

After being connected to power, the device automatically configures itself and should begin sending coordinates to the desired server in about 20 seconds. Afterward, it enters power-saving mode until the waiting time expires and proceeds to send another message. When connected to the server, you have the ability to change the frequency of incoming data by flipping a switch on a dashboard. The device doesn't recognize this change immediately, but it must wait for another uplink to receive a confirmation message with the current switch status. If the state has changed, the device adapts to the desired mode of operation.

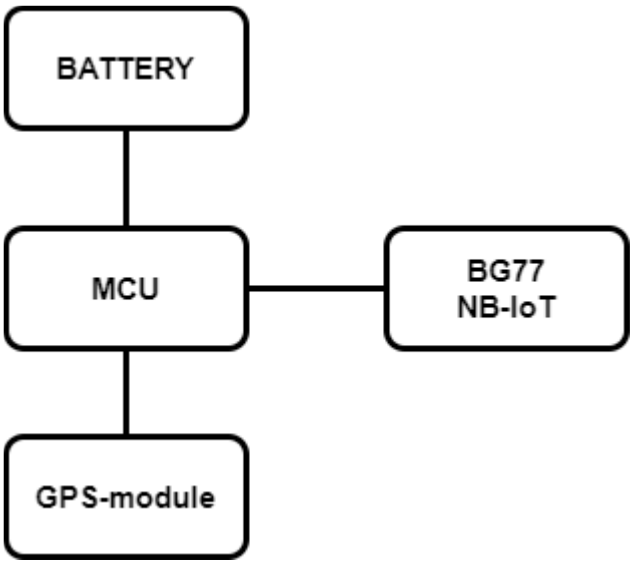
Showcase



https://github.com/dmitrii-semenov/GPS_IOT/blob/main/demo.mov

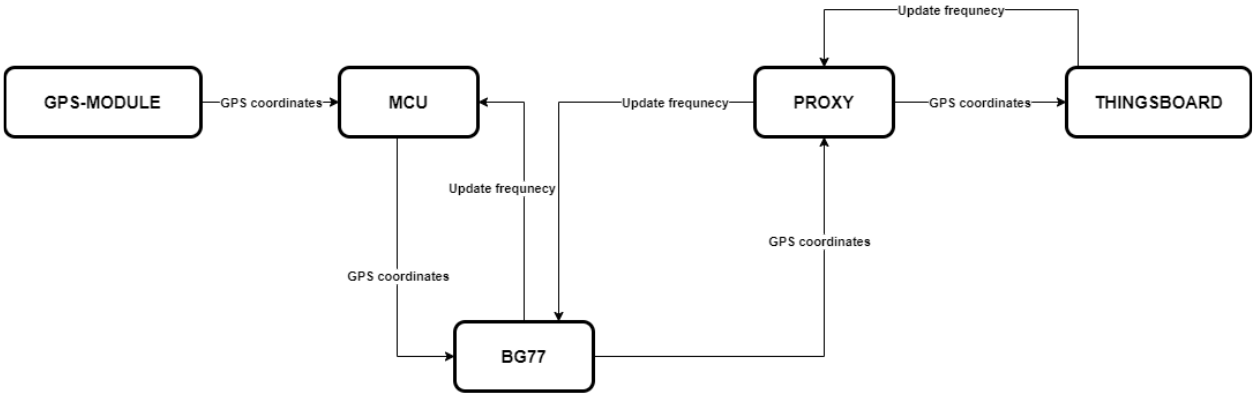
Hardware

The main controlling unit is the MCU (Microcontroller Unit), specifically Raspberry Pico, connected to a BG77 module providing NB-IoT capability. In the full version, the MCU would connect to a battery pack, and a GPS module would be added via a serial port.



Data Transmission

The NB-IoT module sends data according to the set interval to a proxy server using UDP communication protocol. This approach bypasses the requirement of Thingsboard to use TCP-based MQTT and minimizes packet quantity. Upon receiving an uplink, the proxy sends a confirmation message that also includes the status of a fast monitoring switch.



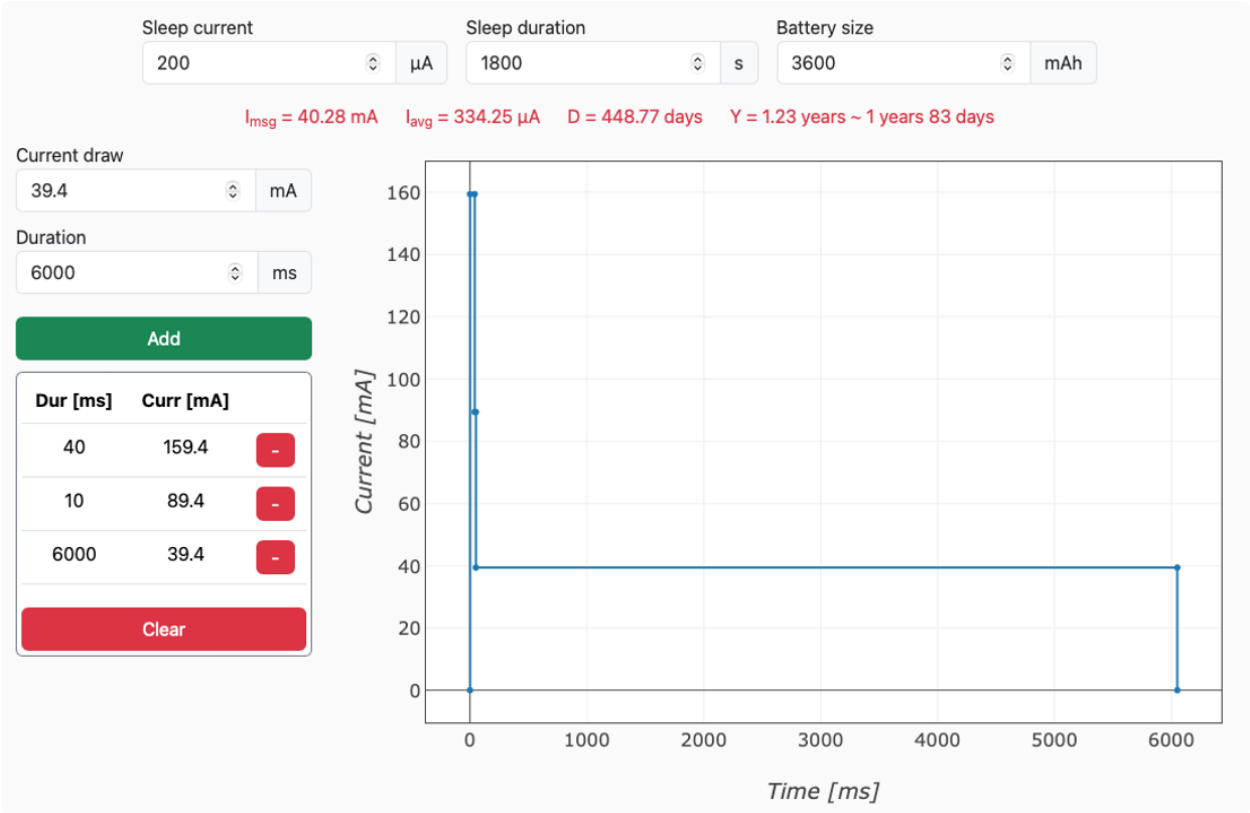
GPS Coordinates Format

```
coordinates = [(longitude, latitude)]
```

Response Examples

```
message = "OK,True"  # Uplink was received and forwarded, quicker updates enabled
message = "OK,False"  # Uplink was received and forwarded, quicker updates
disabled
```

Theoretical battery live



Module	Average Current	Total average current	Time
Network registration			
BG77	27.41 mA	36.81 mA	10 s
MCU RP2040 - Normal	9.4 mA		
Total TX time			
BG77 - TX	250 mA	159.4 mA	4x10 ms = 40 ms
MCU RP2040 - Normal	9.4 mA		
Total RX wait time			
BG77 - RPC_CONNECTED	30 mA	39.4 mA	6 s
MCU RP2040 - Normal	9.4 mA		
Total RX time			
BG77 - RX	80 mA	89.4 mA	10 ms
MCU RP2040 - Normal	9.4 mA		
PSM mode with 30 min interval			
BG77 - PSM	10 uA	200 uA	1800 s
MCU RP2040 - Sleep	190 uA		
PSM mode with 1 min interval			
BG77 - PSM	10 uA	200 uA	60 s
MCU RP2040 - Sleep	190 uA		

Used Solutions and Issues

For data transmission, NB-IoT technology was chosen due to its coverage, low hardware cost and low power consumption. Also because of the nature of this project, the main disadvatanges as high latency, low amount of data transition and possible lags do not affect us in any significant way.

However, as it's a paid solution, minimizing sent and received data was essential. Hence, UDP protocol was utilized for its fire-and-forget nature. To conserve power, the PSM (Power Saving Mode) was employed, as no data is expected during sleep periods. Although a sleep mode for the MCU was considered, issues with the RP2040 prompted its continuous operation. This issue could be solved by using a different MCU, such as the ESP32. When it comes to battery power and a GPS module, neither of these components are included in the solution due to limitations of the development board used. Although the battery issue could potentially be resolved, the larger problem lies with the GPS module. The issue is from the BG77 module utilizing the serial port of the onboard MCU, and the capability to add another device is not implemented on the board. While both of these issues are fixable, they extend beyond the scope of this project.