**IoT Use Case: Realtime Truck Position Monitoring & Geofencing**

# Overview

The monitoring of the GPS position of a vehicle as well as a notification when a defined event occurs, e.g. exceeding a threshold value of a sensor measurement or entering a defined geo-area (geofencing) are frequent tasks in the field of eLogistics.  
In this use case, the IoT perspective is considered. Integration into a business context with a corresponding data model is not described here. However, the use of standard interfaces makes it easy to transfer to a complex scenario.

This example will demonstrate the use of IoT technologies to solve the above examples.

# Technical Requirements and Boundary conditions

Requirements:

* Visualization of realtime position and track data e.g. speed
* Notification e.g. via SMS when reaching a distance to the destination

Hardware for Sense Layer:

* GPS-Position: Neo 6m GPS Modul
* MicroController: Arduino NodeMCU (or RaspPi)

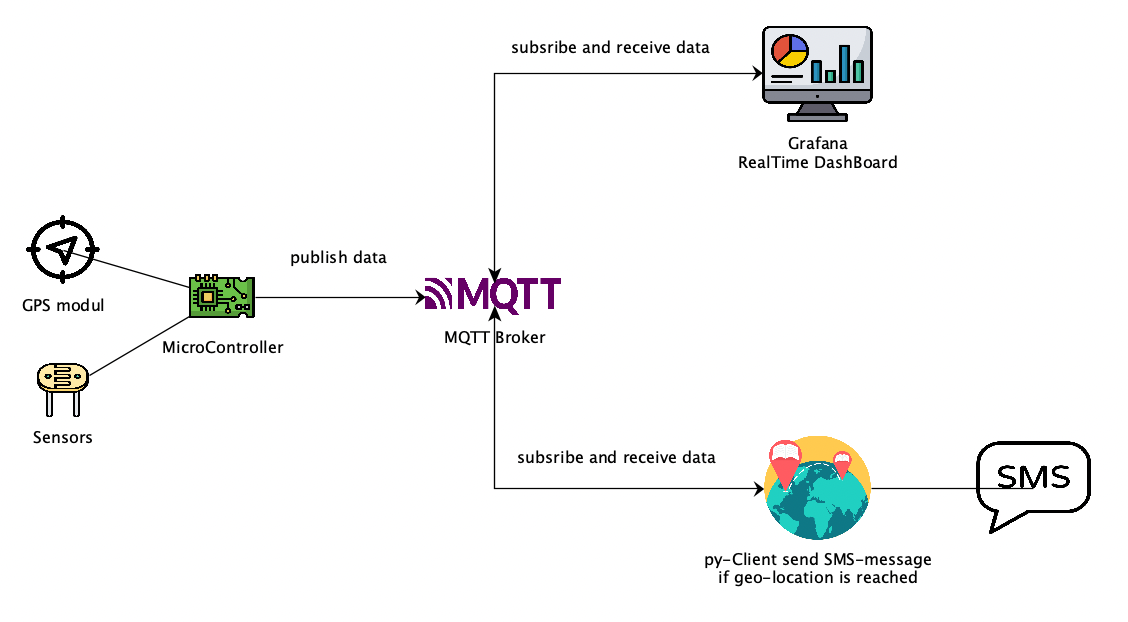
Recommended software development environment:

* Arduino IDE
* python
* Grafana (cloud-based)
* For tests: MQTT.fx

# The Pragmatic Solution

It exists different architectures to solve the problem, a very generic approach for loose coupling of different modules is the Broker Architecture using a publish/subscribe mechanism. In this way, a client can collect the data (microcontroller with sensors) and publish it to the broker. Participants can then log on to this as various services and process the data differently, e.g. by visualizing the data as a stream in real time, writing it to a database or sending an event-driven message regarding the gps-position.

## Architecture & Data Model



# Tasks for the students:

## Create a business case

1. Define a meaningful business case for a concrete scenario e.g. Transportation of Fruits.
2. Document the following steps:
   1. Introduction
   2. Problem Statement and Requirements
   3. Literature Review
   4. Conception including UML-Diagramms, Software Architecture, Data Model and Wireframes
   5. Test conception
3. Next Steps to do: how can we integrate the small IoT-Case into that concreate scenario?

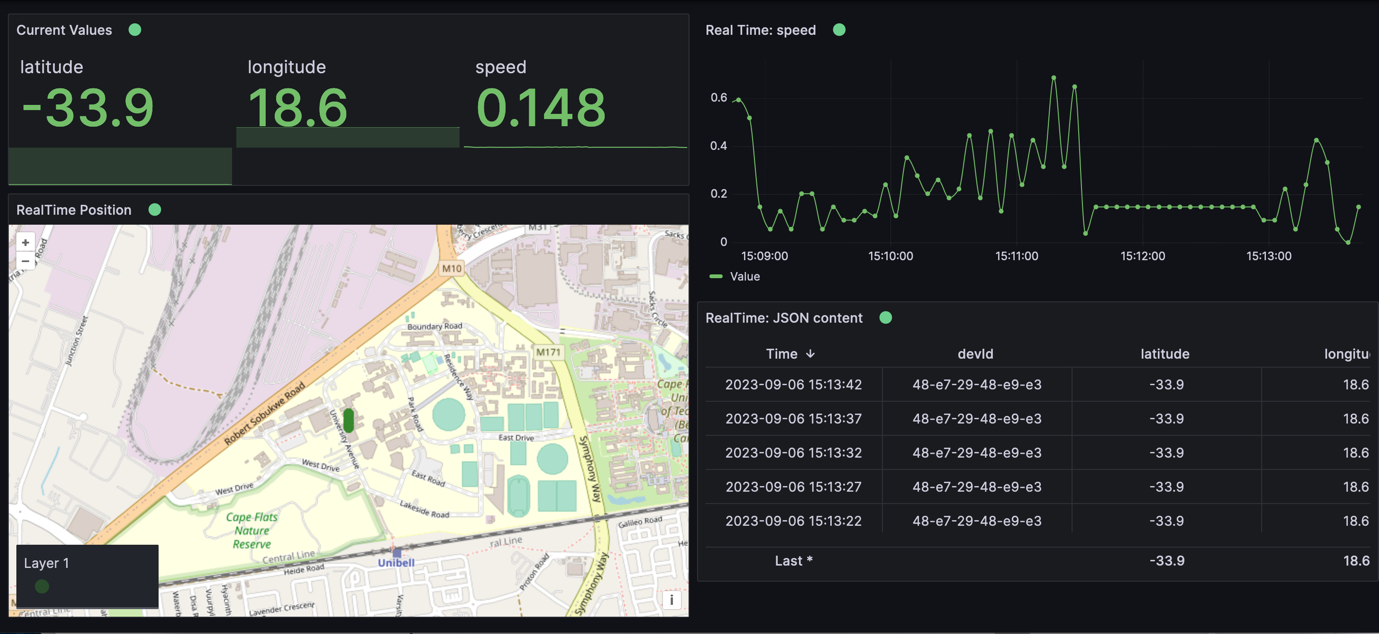
## Programming

1. Programming MicroController to send data to MQTT-Broker over WiFi:
   1. Learn the basics 🡪 <https://www.section.io/engineering-education/getting-started-with-nodemcu/> and <https://randomnerdtutorials.com/esp8266-nodemcu-guides-sensors-modules/>
   2. **Learn to connect/reconnect to WiFi** including parameterize network access in case of changing network without reprogramming (use of a HotSpot for initial input of network data) 🡪 for NodeMCU <https://randomnerdtutorials.com/wifimanager-with-esp8266-autoconnect-custom-parameter-and-manage-your-ssid-and-password/>
   3. **Learn to read GPS-Data** 🡪 for NodeMCU <https://circuitdigest.com/microcontroller-projects/interfacing-gps-with-nodemcu-esp12>
   4. **Learn to send data to MQTT-Broker** 🡪 <https://docs.arduino.cc/tutorials/uno-wifi-rev2/uno-wifi-r2-mqtt-device-to-device>, for tests use the mqtt.fx application
   5. **Learn to send data in json-Format** 🡪 for NodeMCU <https://arduinojson.org/>
2. Create a python client to subscribe for the data and create a notification in case of geo-distance is reached
   1. **Learn the basics** 🡪 <https://www.w3schools.com/python/>
   2. **Learn to implement geo-fencing using geopy**: <https://unprotect.it/snippet/geofencing/183/>
   3. **Learn to communicate with MQTT-Broker** 🡪 <https://pypi.org/project/paho-mqtt/>
   4. **Learn to send a SMS from py-client using twilio** 🡪 <https://www.twilio.com/docs/sms/quickstart/python>
3. Create a DashBoard using Grafana and the MQTT-plugin to stream realtime-Data
   1. **Learn to create a Dashboard using Grafana** 🡪 <https://grafana.com/docs/grafana/latest/getting-started/>
   2. Learn to use Grafana in a cloud 🡪 <https://grafana.com/>
   3. **Learn to use MQTT plugin** 🡪 <https://grafana.com/blog/2021/08/12/streaming-real-time-sensor-data-to-grafana-using-mqtt-and-grafana-live/> hint: use a json-format to send data

# Template Solution:

* MicroController: <https://github.com/dany-meyer/uwc_tests.git> 🡪 NodeMCU\_arduino\_ide\_speedoMeter
* Using the MQTT.fx for tests: observe the data coming from microcontroller or simulate to publish data.

GrafanaDashBoard



Hints:

* DataSource: Connection to BROKER
* Use a json including the key’s longitude and latitude
* Use for map-Visualization:Geomap / configure the topic / configure the Map: MapViev=Fit to data; location mode=auto; base map layer🡪layer typ = open street map

Test Environment for Teacher: toDo