Code Challenge
Maritime IoT Gateway

We are happy and honoured you took the chance to apply for the new role as the Lead Software Developer IoT & Edge for our Digital Solutions team and want to thank you for that already.

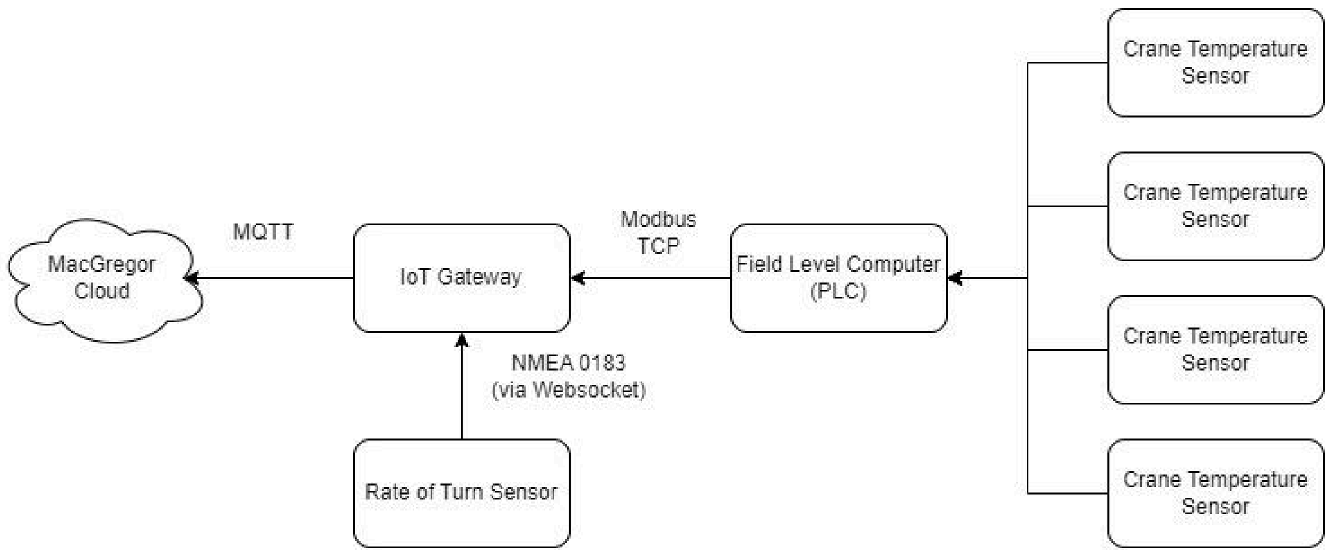
While job interviews are great to get to know each other, they surely are not the greatest place for you to show what you really shine in: Coding. Therefore we want to give you that opportunity with our little Code Challenge. We aimed at making it an interesting and fun activity for a cosy winter afternoon, and we hope you will enjoy it.

General Requirements

- From the time you received the code challenge, you will have **2 days** from the time you received the introduction & prepared material to complete it. However we believe the challenge can be finished successfully in 2 - 4 hours.
- On Completion, please send your results back to: tim.bischoff@macgregor.com, or make them publicly available in a repository of your choice (e.g. Github) and share the link with Tim. Alternatively, you could also use a sharing service as https://wetransfer.com/ .
- If you weren't able to complete it by that time, please send back all you have ready by then. Please also include a note about what caused you trouble so we can understand why you didn't finish. Don't worry, it's not the end of the world!
- If you have any questions during the code challenge, feel free to reach out to Tim who will be on standby:
    - via email (tim.bischoff@macgregor.com),
    - Google meet (with same mail as above),
    - by phone: +49 1578 602 123 7
- You are free to perform the code challenge in your desired programming language, however if you don't know what to choose, we always like to read Python code ;-)
- Just build the application how you normally would, don't take our simulation script as a reference.

Challenge description

Scenario



Imagine you are the lead developer for MacGregor's IoT Gateway which is installed on vessels to fetch data from cranes (just like the one shown in title picture), putting it together and sending it to our cloud application for further processing. You are working together with a team of 3 more developers.

Today you have been asked to implement a feature to read data from two new sensor types:
- 4 temperature sensors for the temperature of the luffing winch motors.
- A Rate of turn (ROT) sensor which indicates the speed of the crane's boom moving around its vertical axis.
- In addition to collecting the data, send it towards the cloud, using MQTT (with your application being the client)

The feature is supposed to be rolled out in multiple customer installations.

Preparation

- We created a simulation script that will constantly send out random measurement values via the localhost websocket onto the computer it is running on. The script is written in Python (version 3.11)
    - You can get the simulation script on Github: https://github.com/ChiquiTi2/CraneIoT
    - You can execute it by calling crane_simulation.py; it can be terminated by Keyboard Interrupt (Ctrl + C)
- As a precondition, make sure you have the following set up to test your client application with our server script:
    - Have Python installed in version 3.11 or newer
    - We recommend you use pip to install the required dependencies, as listed in the included requirements.txt
- It's recommended to keep the server running in one terminal window and open another one to test your client application

Technical details & constraints

General Sensor Data handling

Modbus TCP temperature sensors

- In total, 4 temperature sensors shall be connected for the luffing motors of the crane.
- The temperature sensors are measuring in a range of 0 - 100°C, while the regular temperature is expected to be between 20 - 80°C
- The sensors are all connected to a central PLC (Programmable Logic controller, i.e. kind of a field level computer). The PLC has a built in Modbus TCP Server constantly running. The sensor measurements are mapped to the holding registers according to below table.

- Please note: For the code challenge, the simulation server script we provided will take the role of the PLC, i.e. keeping the Modbus TCP Server running

Holding Register	Measurement
0	Luffing motor 1 temperature (PS Winch)
1	Luffing motor 2 temperature (STB Winch)
2	Luffing motor 3 temperature (PS Winch)
3	Luffing motor 4 temperature (STB Winch)

- The sensor measurements have an update frequency of 0.5 Hz
- The Modbus server detailed settings are listed in the table below.
  - Please note that these are default values for testing purposes, however there are customer projects where the settings will be different.

Setting type	Setting value
IP	127.0.0.1
Host	localhost
Port	8889
Transport Layer Protocol	TCP
Unit ID	1

- In general, Modbus supports multiple options to store data. Our measurements are stored in holding registers of 16 bit length
- In Modbus TCP, the client is responsible to send the request of reading the holding registers. The specification of that request is shown below:

Function code	3 (Read Multiple Holding Register)
Required information for request	Starting address of holding reg. (2 bytes) Quantity of holding reg. (2 bytes)

- The timeout settings are not predefined by the protocol itself, as the adequate time depends on the network architecture. A setting of 100 ms should in general be safe for most networks and adequate for the use case over here. Ultimately it would depend on the details of a customer project, though.
- If the Modbus server isn't able to act according to the request, it will send an Exception response instead.
  - For the Gateway application, it's sufficient to aggregate the exceptions and log a general Modbus error. As a result, the application should just retry with the next request cycle.
- Further details can be found on Wikipedia (https://en.wikipedia.org/wiki/Modbus) or from the Modbus Foundation (https://modbus.org/specs.php)

NMEA Rate of Turn Sensor

- The measurement can be anything within 0 - 999 °/min
  - Plausible values are 0 - 360°/min for this application
  - The measurement contains 1 decimal places
  - The publishing frequency of measurement updates is 0.5 Hz
- NMEA is a typical protocol used in the maritime industry, especially for navigation components. Originally, it was bound to a serial communication (NMEA 0183)
  - For our example, we chose to set up a connection based on WebSocket instead of serial communication
- NMEA is characterised by ASCII sentences in a standardised format, as shown in below table. It also shows an example specific for the Rate of Turn Sensor connected here.

<protocol id>	<Talker ID>	<sentence type>	data	<end of data>	<checksum >	<end of sentence>
$	MG	ROT,	2.0, A (the letter shows the status, A = valid data, V = invalid data)	*	33	<CR><LF> (carriage return & line feed)

MQTT Connection

- The gateway will act as a publishing MQTT client. It will publish the measurements towards the topics as specified in below table

Measurement	Topic
Temp. Luff. Mot. 1	ows-challenge/mv-sinking-boat/main-crane/luffing/temp-mot-1
Temp. Luff. Mot. 2	ows-challenge/mv-sinking-boat/main-crane/luffing/temp-mot-2
Temp. Luff. Mot. 3	ows-challenge/mv-sinking-boat/main-crane/luffing/temp-mot-3
Temp. Luff. Mot. 4	ows-challenge/mv-sinking-boat/main-crane/luffing/temp-mot-4
ROT	ows-challenge/mv-sinking-boat/main-crane/rot

- For the code challenge, we will use a free public broker. Detailed information and a general dashboard is available under https://mqtt-dashboard.com, in addition some details are listed in below table

Host	broker.hivemq.com
TCP Port	1883
Websocket Port	8000
TLS TCP Port	8883
TLS Websocket Port	8884
username / password	not used
client id	ows-challenge-{date, format ddmmyy}

- For the publishing, following details apply:
  - If data doesn't change (i.e. not differing more than 1°/min or °C from the previous value), for the sake of saving traffic related costs no update shall be sent to the broker; however at least every 5 minutes an update should be send even if the data hasn't changed significantly
  - The connection should be kept alive for max. 10 min even if no data is sent
  - each message should be sent at least once
  - issue a LWT (Last Will & Testament) message to each topic with the content: "connection lost"
- As the time of measurement is important to track, publish it together with the measured value in a format like:
  "10.0°C, Valid, 2023-11-18 at 18:30 UTC"
  - If a problem exists with the measurement, assign a "Invalid" tag