Homework
Software Architecture for Storing Data Coming From Measurement Devices

Contents

Figure of Tables

Scope

This document intends to use as homework to explain details about the software that receives data coming from the battery measurement devices.

Context

Every device, dispatch its measurements through a network (CAN, MODBUS, Ethernet, etc.) An embedded device on which we're developing software on it receives all the data by the DataReceive handler and takes an action to store them in a database.

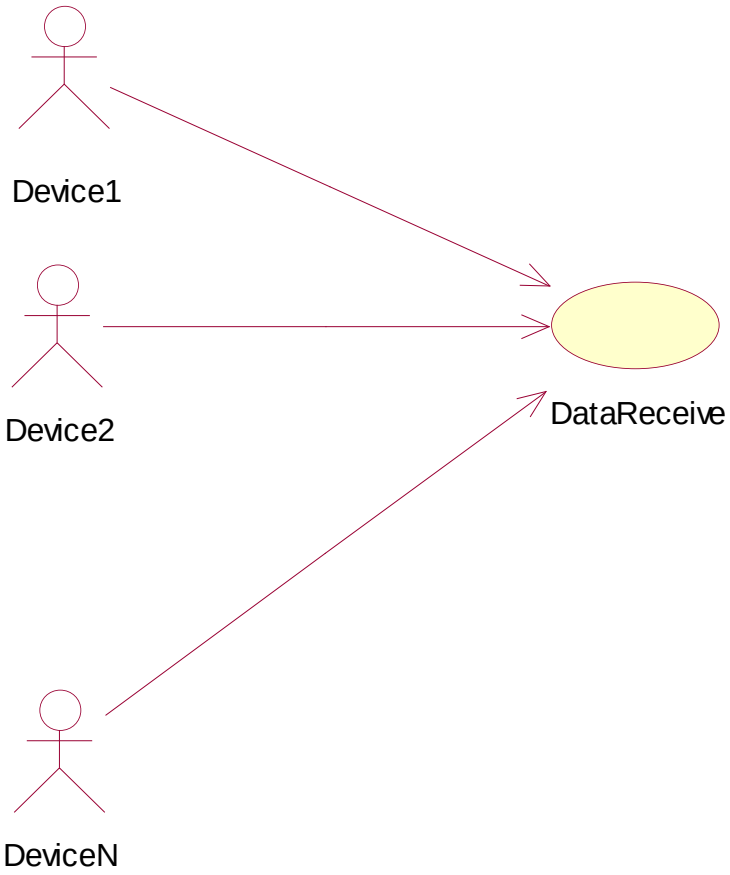

Figure 1 Context Diagram

Infrastructure

The picture below shows us the device our code run on it. The device should have an interface between the Devices mentioned above. The device may be one of the embedded devices. They can be COTS (Commercial on The Shelf) single board computers (SBC), custom-designed boards, or similar devices. Devices need to reach a database preferably one of the modern databases (e.g., PostgreSQL) over the internet or a local network.

The monitor should show us the momentary stored and received packages.

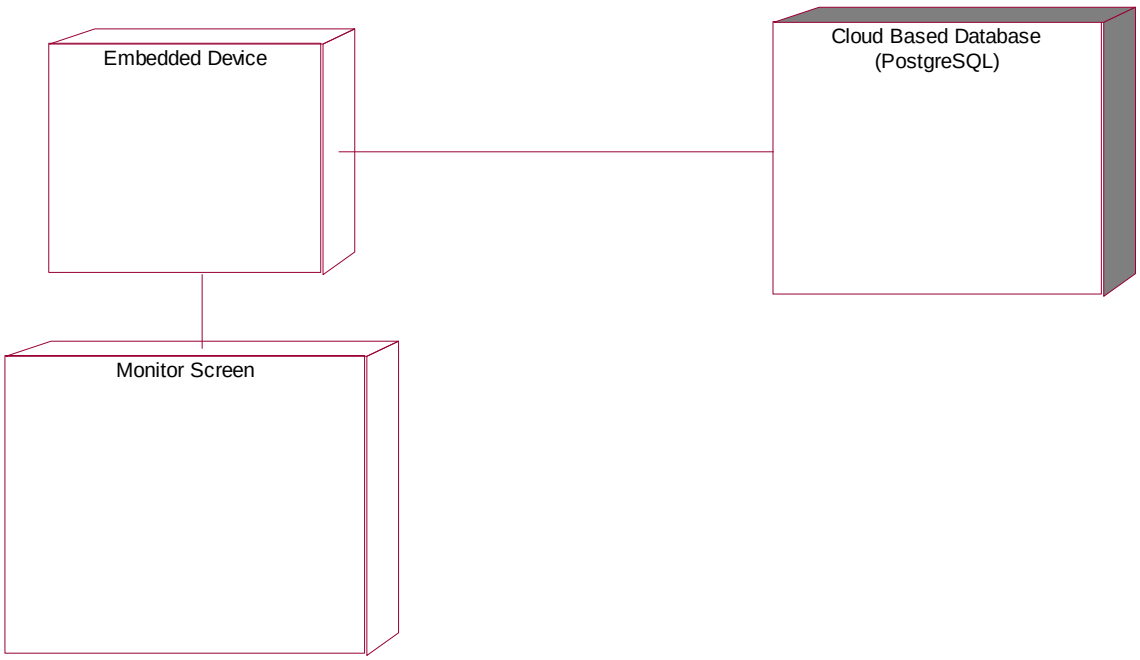

Figure 2 Container Diagram

Components

The software components need to be used to accomplish the project goals.

- libpqxx is the connection library of PostgreSQL database
- Database is the server installation of PostgreSQL database
- DataStore is the task we use to store data incoming from the measurement devices.
- DataReceive is the task we use to receive and parse the data coming from measurement devices.
- Monitor is the task we show the packages incoming from measurement devices and stored.
- DataReceiver is the main code snippet to manage tasks mentioned above

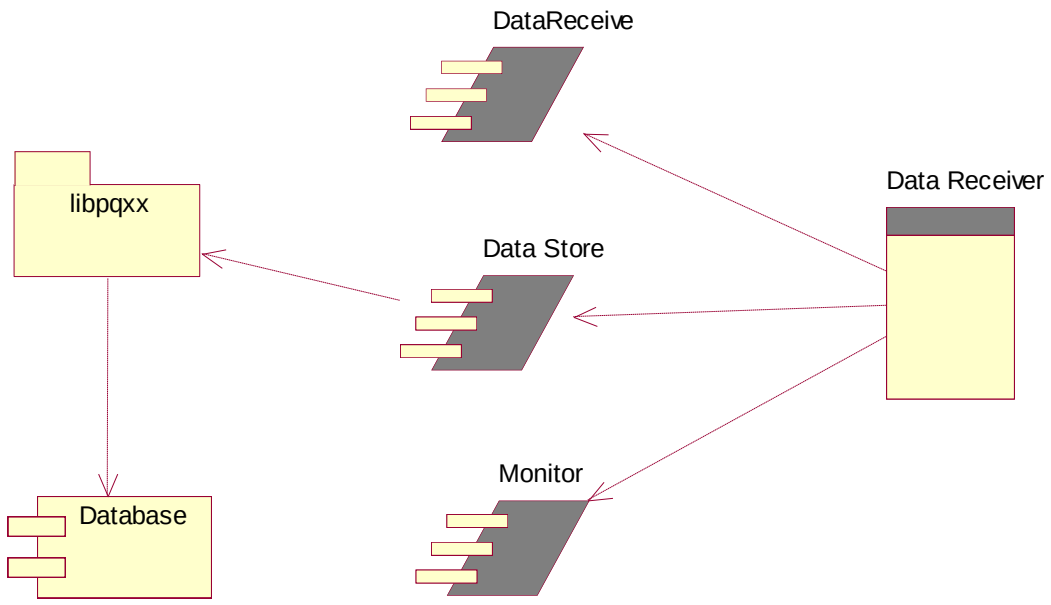

Figure 3 Component Diagram

Code Diagram

The code will be developed to receive and store data coming from measurement devices. The code will be developed on modern C++ (C++20 preferably). It's going to be a CMAKE project to be adaptable to any IDE (Integrated development environment)

The database includes two major tables. One of them is Device which responsible device id's and other one is the Messages table which responsible messages. The messages will be stored as binary to adapt software for future changes.

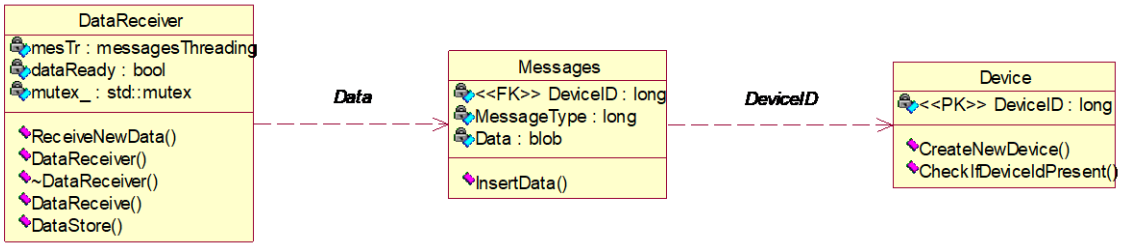

Figure 4 Class Diagram

Theory of operation as follows;

when DataReceiver Class received a data from one of the measurement devices, first, need to check its ID if previously created in the Device table. If it's not new device id will be created on the table.

Then, received message, would be parsed regarding message types. Message type would be one of the (capacity, health, alarm conditions, etc..) Messages Class would be the responsible class for this task.