AQS Documentation

Contents

Introduction

The Air Quality Server (AQS) is a lightweight tool that can be used to receive, store and serve air quality data from the third generation air quality sensor from Exploratory Engineering. It can both be run as a more permanent server in a datacenter in order to receive, process, store and pass on data, but it is also designed for ad-hoc experimentation by researchers and integrators. Some care has been taken in order to make both scenarios possible. It should be trivial to download, build and run the server on your personal machine, as well as running it in a datacenter.

Data reception

Currently the server supports two different ways of receiving data - you can either configure it to connect to IoT-GW/Horde and have it listen to the data stream belonging to a given *collection*.

We are also in the process of developing a solution that will allow the server to use MIC as the source of messages, but this is not supported at the time of writing.

For experimentation purposes it is also possible to use a UDP-based protocol for injecting messages into a running server.

Database storage

The AQS makes use of a local SQLite 3 database where it stores calibration data as well as the messages (datapoints) it has received from the network. We chose to use SQLite 3 as the database as a convenience because the data rates and amounts of data are fairly modest for the number of sensors we are dealing with. At some later point we may support PostgreSQL so that the Air Quality Server can deal with higher volumes.

The choice of embedding the database was made to make life simpler for researchers and people who simply want to experiment with the system, as there is no need to set up and manage a separate database system. Everything is taken care of by the server. The database itself is located in a single file, which makes it easy to copy the data around and perform ad-hoc queries on it using the `sqlite3` command line utility.

Open source

The AQS is published under the Apache 2.0 license, and you are encouraged to contribute to its development. Bug reports and feature requests filed through the issue tracking on the Github project are welcome. Pull requests with code are even more welcome. The Git repository is located at https://github.com/exploratoryengineering/air-quality-sensor-node.

Building AQS

Check out source from Github

The AQS is hosted on Github at and you can clone it by issuing the command:

```
git clone https://github.com/ExploratoryEngineering/air-quality-sensor-node.git
```

This will copy not only the server, but also the hardware designs for the sensor, the firmware source code for the device, and the server. In order to build the server change directory to the `server` directory.

Fetch dependencies

- Go 1.14 or newer
- Protobuf Compiler version 3.0.0 or newer
- Protobuf Code generator for Go
- SQLite3 libraries

Installing Go

For information on how to install Go, please refer to https://golang.org/doc/install.

Installing Protobuf

To install the protobuf compiler on macOS the easiest route is to use Homebrew and install it with:

```
brew install protobuf
```

or on Linux

```
apt install protobuf-compiler
```

You then need to install the protobuf code generator for Go, which you can install with the command, which should work on all platforms Go has been ported to:

```
go install google.golang.org/protobuf/cmd/protoc-gen-go
```

Installing libsqlite3

On macOS you can install the latest SQLite version (including libraries) using

```
brew install sqlite
```

On Linux you can do this with

```
apt install libsqlite3-dev
```

Building AQS with make

We use `make` to build the server. Per default the `Makefile` is set up to build the server for macOS. You can either edit the `Makefile` and change the default values for `GOOS` and `GOARCH` to fit the platform you are building on, or you can specify these variables on the command line.

For instance in order to build for Linux you would

```
GOOS=linux GOARCH=amd64 make
```

This should produce a `bin` directory containing a runnable binary called `ac`. To test that you have succeeded in building it you can run `bin/aq` and it should produce output that looks something like this:

```
$ bin/aq -h
Usage:
  aq [OPTIONS] <command>

Application Options:
  -c, --collection=<collectionID>    Horde collection id to listen to (default: 17dh0cf43jg007)
  -d, --db=<file>                    Data storage file (default: aq.db)
  -v, --verbose

Help Options:
  -h, --help                         Show this help message

Available commands:
  convert  Convert calibration data
  fetch    Fetch historical data
  import   Import a calibration data
  list     List calibration data
  rm       Remove calibration data
  run      Run server
  show     Show calibration data for device
```