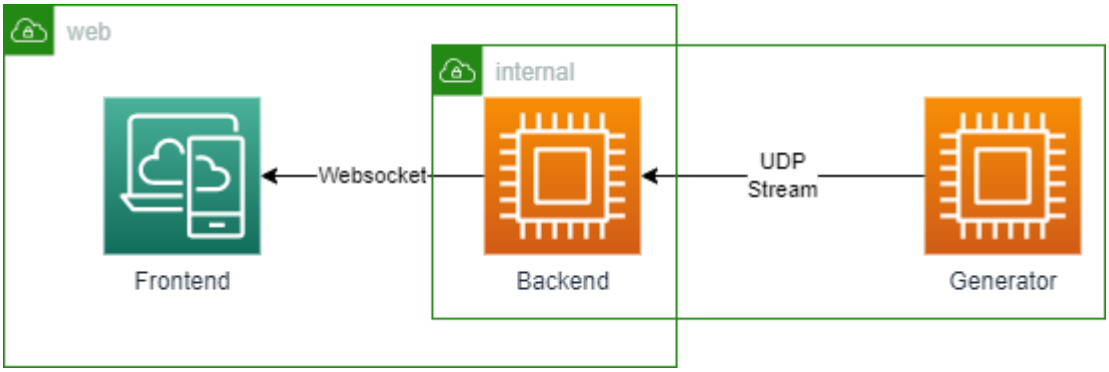
Overview

Three components:

1. Generator: creates and sends out signal samples
2. Backend: receives samples and passes them on to the frontend
3. Frontend: UI that displays and manipulates the signal

Architecture



Docker

The application is composed of three docker containers representing the three components.

Docker compose configures the containers and allows to start the application with a single command `docker-compose up --build`.

Generator

The generator has two tasks:

1. creating signal samples
2. sending them out as UDP packages to our backend

A UDP package currently consists of ten samples representing the voltage values for ten oscilloscope channels.

Building and managing dependencies of the generator is done using cargo, which is the package manager for rust.

Backend

The backend is a node server. The server fulfills two tasks:

1. listen on a UDP socket for the samples sent out by the generator
2. transfer the samples to the frontend via a WebSocket

Frontend

The frontend consists of types of components

1. The cartesian plot that displays the most recently received samples as a continuous signal plot
2. UI elements that manipulate the representation of the samples

Code

Generator

The generator is written in rust. To create signals we use dasp::Signal crate. To achieve a precise outgoing package/s rate up to 10_000 we use tokio to measure the time between packages sent out.

For testing, we have unit tests that can be run with `cargo test`.

We follow this coding guideline: https://github.com/rust-lang/fmt-rfcs.

The guideline can be automatically enforced by running `cargo fmt`.

Backend

As we use a Node.js server the backend is written in Javascript.

Frontend

The frontend is written in Javascript and uses Vite to build the UI components.

The signal is plotted using webgl-plot which uses WebGL for fast GPU rendering.

For testing we use Cypress.

To achieve uniform code in the frontend we set up a linter/prettier. The configuration can be found in the following files:

`Apps/frontend/.eslintrc.json`

`Apps/frontend/.prettierrc`