sACN Library Usage Instructions

April 2020

Installation

The library can be installed by putting the Lib folder within an accessible path and then adding sacn as dependency within the cargo.toml of your project.

[dependencies]
sacn = { path = "../Lib"}

The library can then be accessed within your project using an extern crate import

extern crate sacn;

Usage Documentation

The documentation demonstrating usage of the library is included in the docs/sacn folder. This documentation can be re-generated using the cargo doc command. This command includes the first argument to not document external dependencies and the second argument to include documentation of non-public items. If you are just utilising the library rather than adding features then the second argument can be omitted to reduce the documentation size. The last argument opens the documentation automatically after generation.

cargo doc --no-deps --document-private-items --open

Running the demo programs

The demo programs which are used to demonstrate the library in use as-well-as for the tests which require them can be built and run using the commands.

cargo run --bin demo_src <interface_IP> <source_name>
cargo run --bin demo_rcv <interface_IP>

Where:
<interface_IP> is the IP of the interface that the demo program should use.
<source_name> is the name of the source within the sACN packets.

To see the usage instructions for the demo programs once running type 'h' to display the help.

Testing the library - Compile Errors / Warnings

To check that the library compiles as expected without errors or warnings (none should be expected) run:

cargo check

Testing the library - Unit Tests

The library unit tests are created using the standard rust unit testing system and can therefore be run directly from cargo.

cargo test

Testing the library - Single Machine Integration Tests

The single machine integration tests are also run from cargo but require some machine configuration. The machine should be setup to use multiple IP addresses within the same subnet with the defaults required for the test being *"192.168.0.6", "192.168.0.7"* and *"192.168.0.8"* for IPv4 and "2a02:c7f:d20a:c600:a502:2dae:7716:601b", "2a02:c7f:d20a:c600:a502:2dae:7716:601c" and "2a02:c7f:d20a:c600:a502:2dae:7716:601d" for IPv6. Theses defaults can be changed and different addreses used by modifying the 'TEST_NETWORK_INTERFACE_IPV4' constant at the top of the ipv4_tests.rs file to change the IPv4 addresses and modifying the 'TEST_NETWORK_INTERFACE_IPV6' constant at the top of the ipv6_tests.rs file.

Once configured the tests can be run using the command below. Note that some of these tests require the network to support IP multicast.

cargo test -- --ignored --test-threads=1

Testing the library - Code Coverage

To view the code coverage of the library the grcov tool from Mozilla is used. Once setup (as detailed within the grcov documentation) the following commands can be run to install the nightly toolchain (required for grcov), build the library and run the tests.

rustup toolchain install nightly
rustup run nightly cargo build
rustup run nightly cargo test -- --ignored --test-threads=1

Testing the library - Multi-Machine Integration Tests

The setup of these tests is detailed in the report. These tests require a specific machine configuration with a shared file system as detailed within the report. Once set up these tests can be run by running the below command on the 'master' machine from within the script-testing folder.

sh ./test.sh

Testing the library - Interoperability Testing

The library was tested for interoperability with three external programs, Avolites Titan v11.4, sACNView version 2.1.0 and Vectorworks Vision Plus 2019 version 24.0.6.521266. This testing is detailed within the report and in the 'CS4099 - Interoperability Testing.pdf' file. The Interoperability Testing folder also includes videos showing the various tests and the results with details allowing recreation if desired.