



Ifxapp

A reusable Python-based Qt widget library for radar applications

Kleinmeier Benedikt

Nov 19, 2024

Table of contents

1	Motivation	3
2	Ifxapp: Overview	7
3	Ifxapp: Demos	16

Motivation

Motivation: Ifxapp

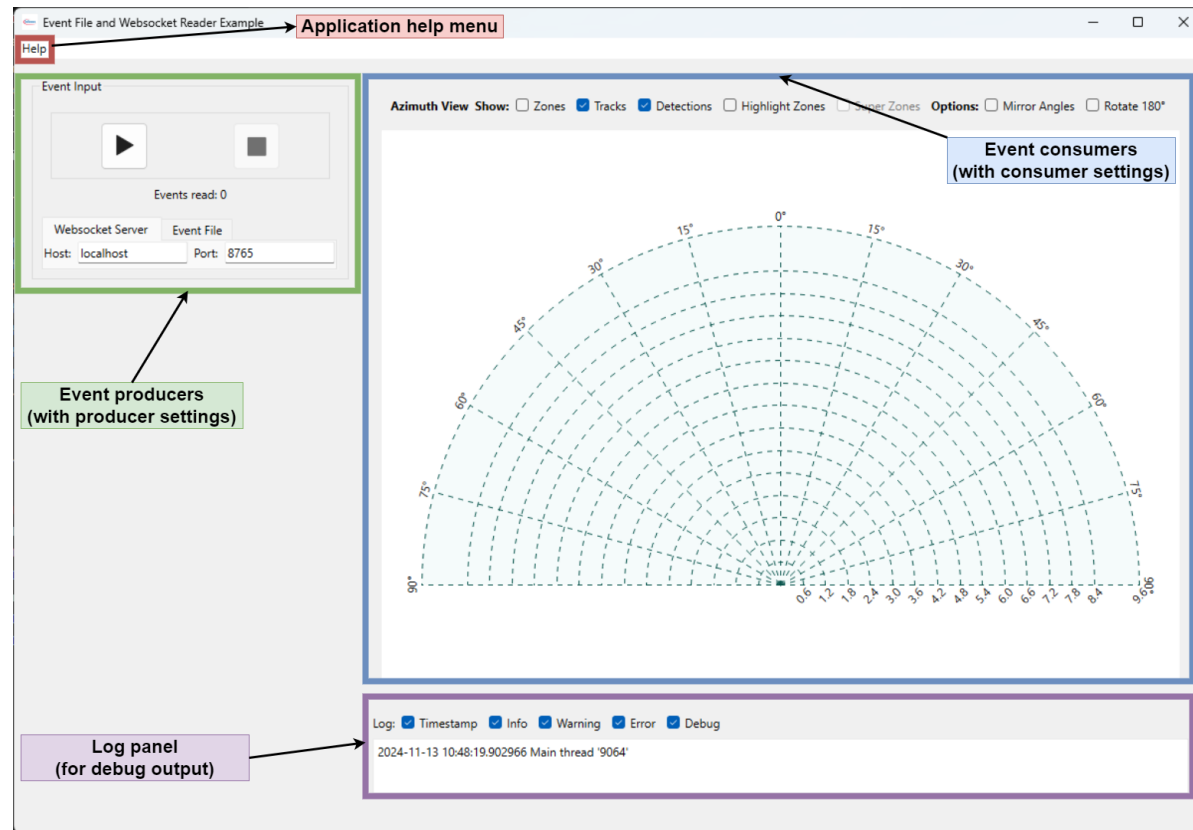
Current status: PSS RFS develops **different applications** (SmartTV, Air Condition, Security Camera & Doorbell), in **different teams** spread **around the globe!**

How to lift synergies between these development efforts?



Motivation: Ifxapp

By providing a reusable GUI library for radar data visualization!



Benefits:

- Common look & feel
- Can be used across different projects
- Builds up abstractions

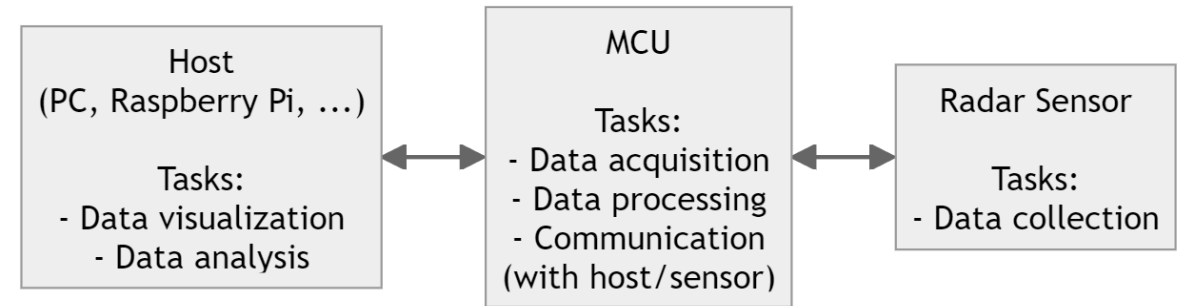
I.e., embedded devs can focus on data communication, algo devs on algo development etc.

Assumptions

- An **application** usually runs on a microcontroller (MCU) and **consists of**:
 - An **algorithm*** which processes data from a radar sensor (Smarter) and
 - **Code to communicate** to a PC and to the actual radar sensor.

* The algorithm is usually implemented in Matlab and converted to C by using Matlab's code generation feature.

An application is usually also **complemented by** a small **Python-based GUI** to visualize the radar data on host side.



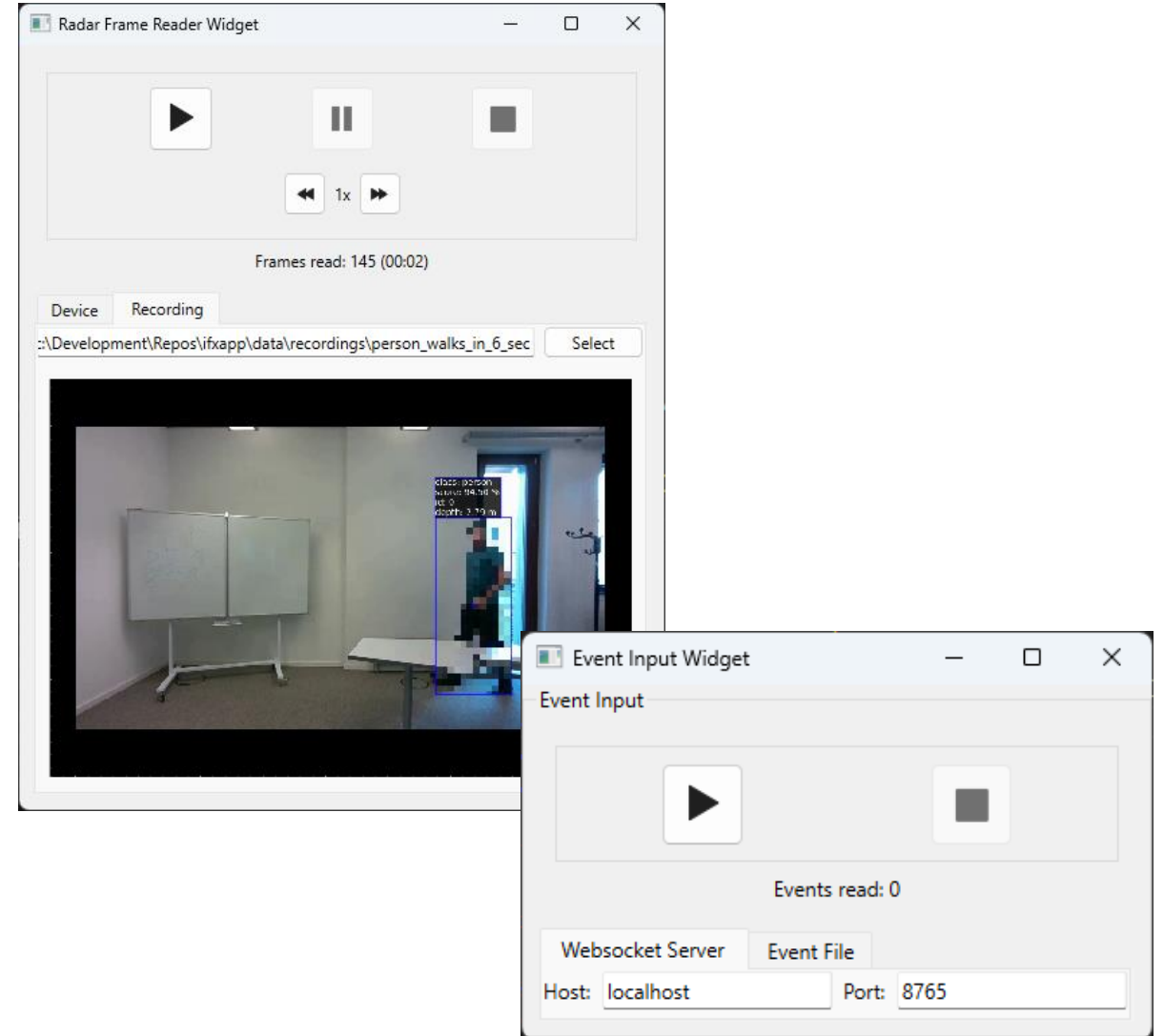
Hardware components of an application

Ifxapp: Overview

Project goals

Ifxapp:

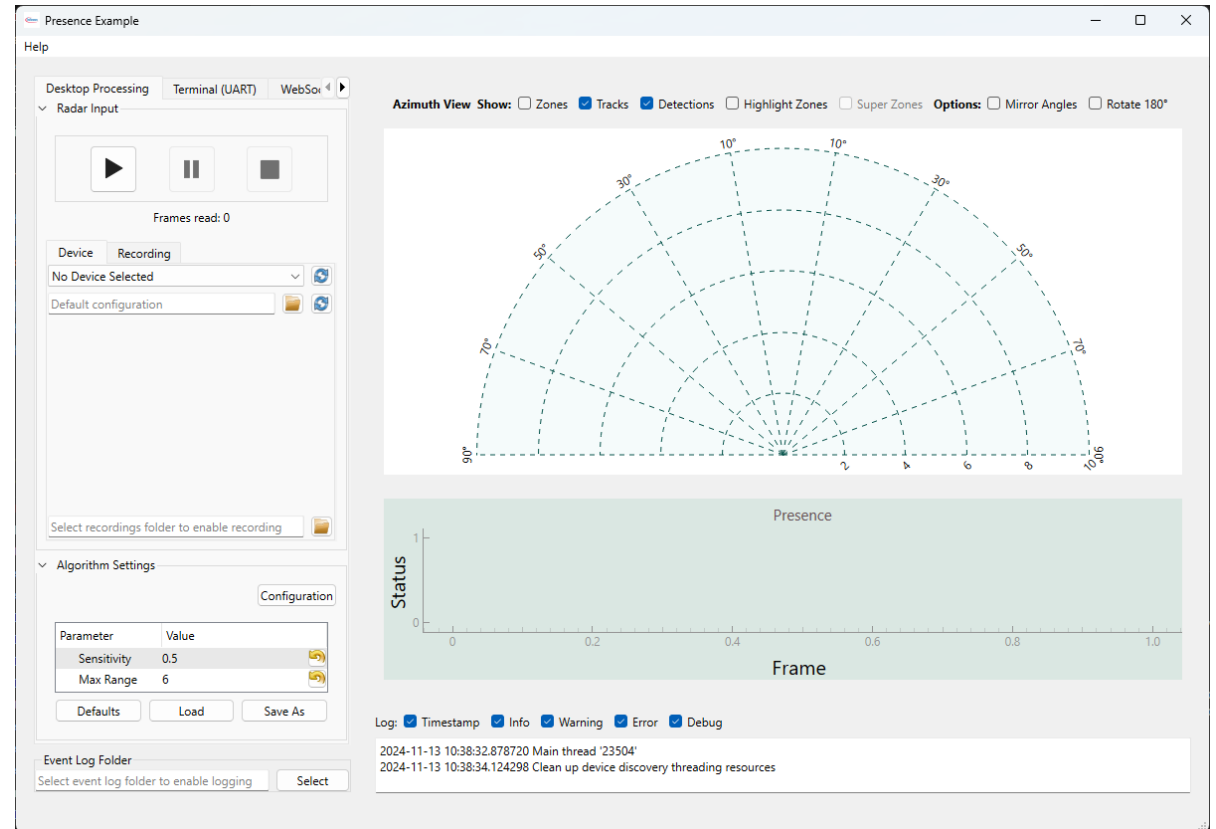
1. Offers **Qt GUI widgets** with a common look-and-feel to visualize algorithm events defined in [ifxalgoapi](#).
I.e., ifxapp consumes ifxalgoapi events instead of ADC raw data!



Project goals

Ifxapp:

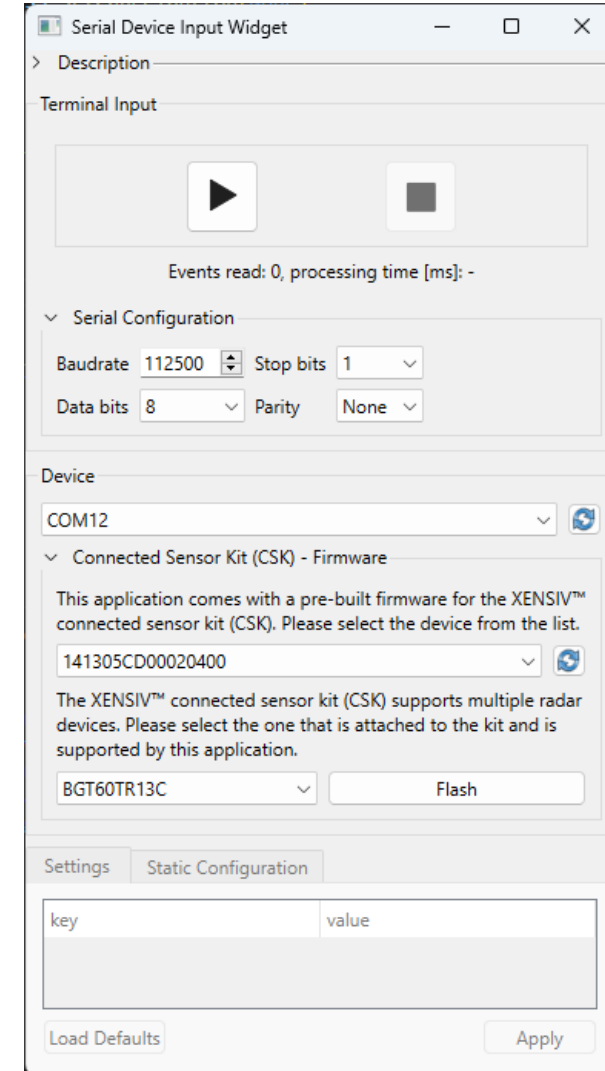
1. Offers Qt GUI widgets with a common look-and-feel to visualize algorithm events defined in [ifxalgoapi](#).
I.e., ifxapp consumes ifxalgoapi events instead of ADC raw data!
2. Allows to quickly build **GUI main windows** for customer demos but also during algorithm development.



Project goals

Ifxapp:

1. Offers Qt GUI widgets with a common look-and-feel to visualize algorithm events defined in [ifxalgoapi](#). I.e., ifxapp consumes ifxalgoapi events instead of ADC raw data!
2. Allows to quickly build GUI main windows for customer demos but also during algorithm development.
3. Offers **access to radar sensors** (via Strata firmwares and evaluation kits like CSK).



Ifxapp: Feature overview and event sources

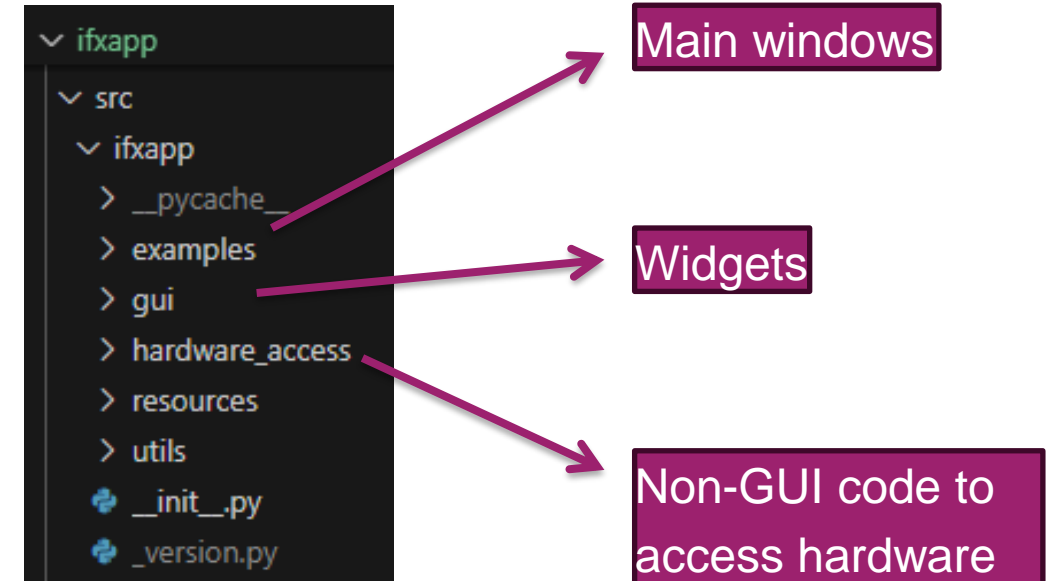
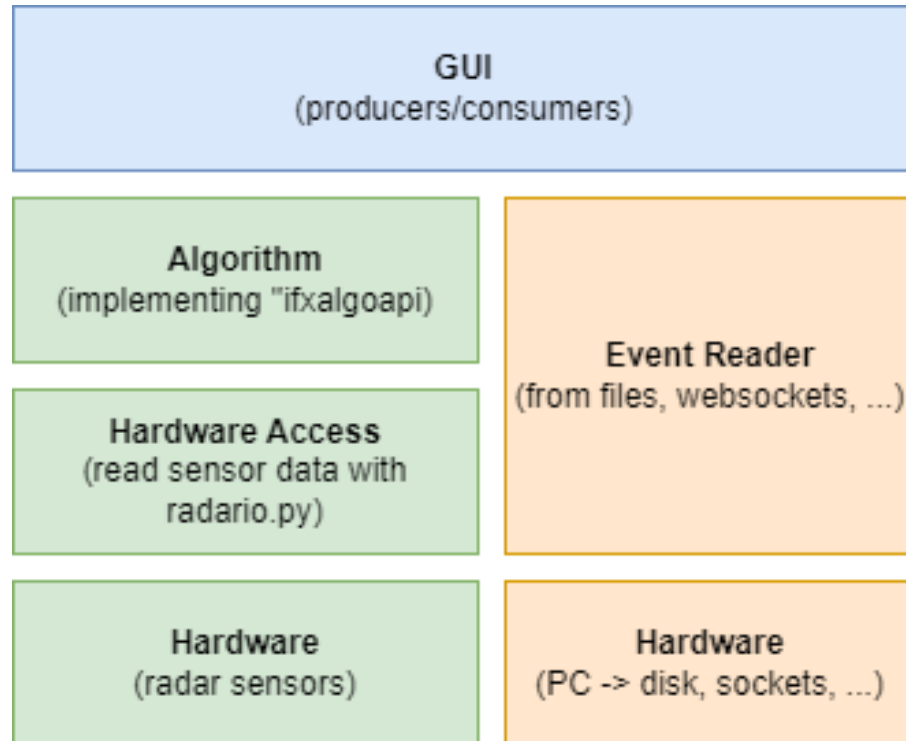
Ifxapp can read radar data (algorithm events) from the following sources:

- Files
- Radar sensors
 - Sensor on-chip processing chains (e.g. motion detector)
 - Access radar processing chains on embedded systems (i.e., directly fetch algorithm output from a MCU FW like CSK)
- Websockets (e.g., process radar raw frames in Matlab and send it to ifxapp via websockets)

Ifxapp visualizes:

- [ifxalgoapi](#) events (AlgoPresenceEvent, AlgoTrackingEvent, ...)
- Video recordings from [ifxdaq](#)
- Planned: Custom plots to enable developers to debug their algorithm

Software architecture

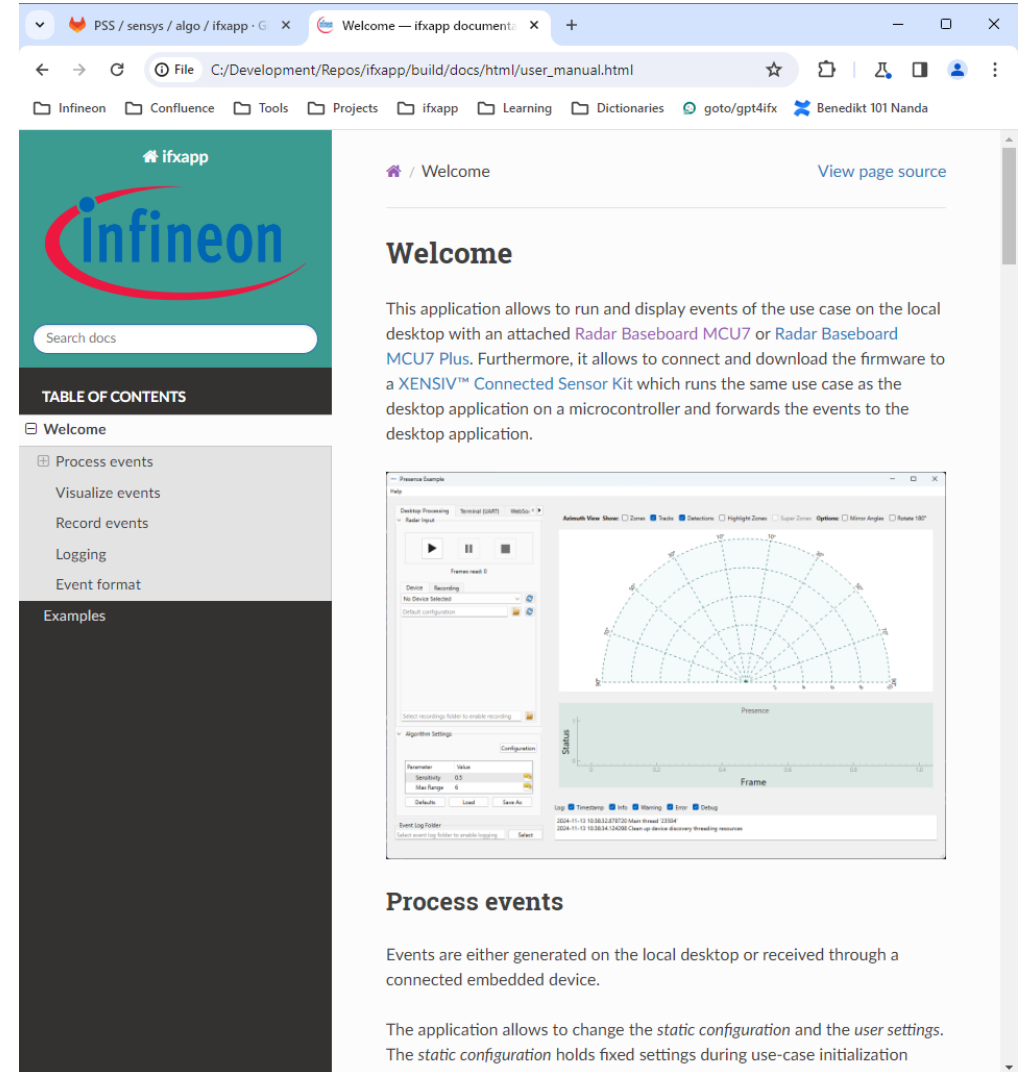


Project setup

- Repo on [GitLab](#)
- Ticket-driven approach with feature branches and merge requests
- Release notifications on [Webex](#)
 - A Git tag triggers (1) Python .whl creation (2) HTML doc and (3) executable generation (with cx_Freeze)
 - <https://gitlab.acme.com/pss/ifxapp/-/releases>
- HTML documentation (currently not uploaded to Artifactory)

Dev tools:

- Python 3.9
 - pre-commit 3.7.1
 - cx_Freeze 7.2.2
 - pytest 8.2.1
 - ruff 0.7.1
 - Sphinx 5.2.1
- VS Code 1.87
- Code coverage: ~30%



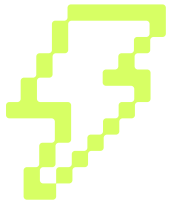
The image shows two screenshots. The top screenshot is a web browser displaying the 'Welcome' page of the 'ifxapp' documentation. The browser's address bar shows the file path 'C:/Development/Repos/ifxapp/build/docs/html/user_manual.html'. The page features the Infineon logo, a search bar, and a 'TABLE OF CONTENTS' section with links to 'Welcome', 'Process events', 'Visualize events', 'Record events', 'Logging', 'Event format', and 'Examples'. The bottom screenshot shows the 'ifxapp' application interface. It has a sidebar with the same 'TABLE OF CONTENTS' and a main area displaying a 'Presence Sample' window. This window includes a 'Radar Input' section with play/pause buttons, a 'Radar View' showing a radar plot, and a 'Status' section with a timeline. The application also has a 'Configuration' section with fields for 'Resonator', 'Frequency', and 'Max Range', and a 'Log' section at the bottom.

Project setup

- Repo on [GitLab](#)
- Ticket-driven approach with feature branches and merge requests
- Release notifications on [Webex](#)
 - A Git tag triggers (1) Python .whl creation (2) HTML doc and (3) executable generation (with cx_Freeze)
 - <https://gitlab.acme.com/pss/ifxapp/-/releases>
- HTML documentation (currently not uploaded to Artifactory)

Dev tools:

- Python 3.9
 - pre-commit 3.7.1
 - cx_Freeze 7.2.2
 - pytest 8.2.1
 - ruff 0.7.1
 - Sphinx 5.2.1
- VS Code 1.87
- Code coverage: ~30%



Outlook

- **Current focus:**

- Get stable release out the door with minimal feature set to unblock projects.

- **Next:**

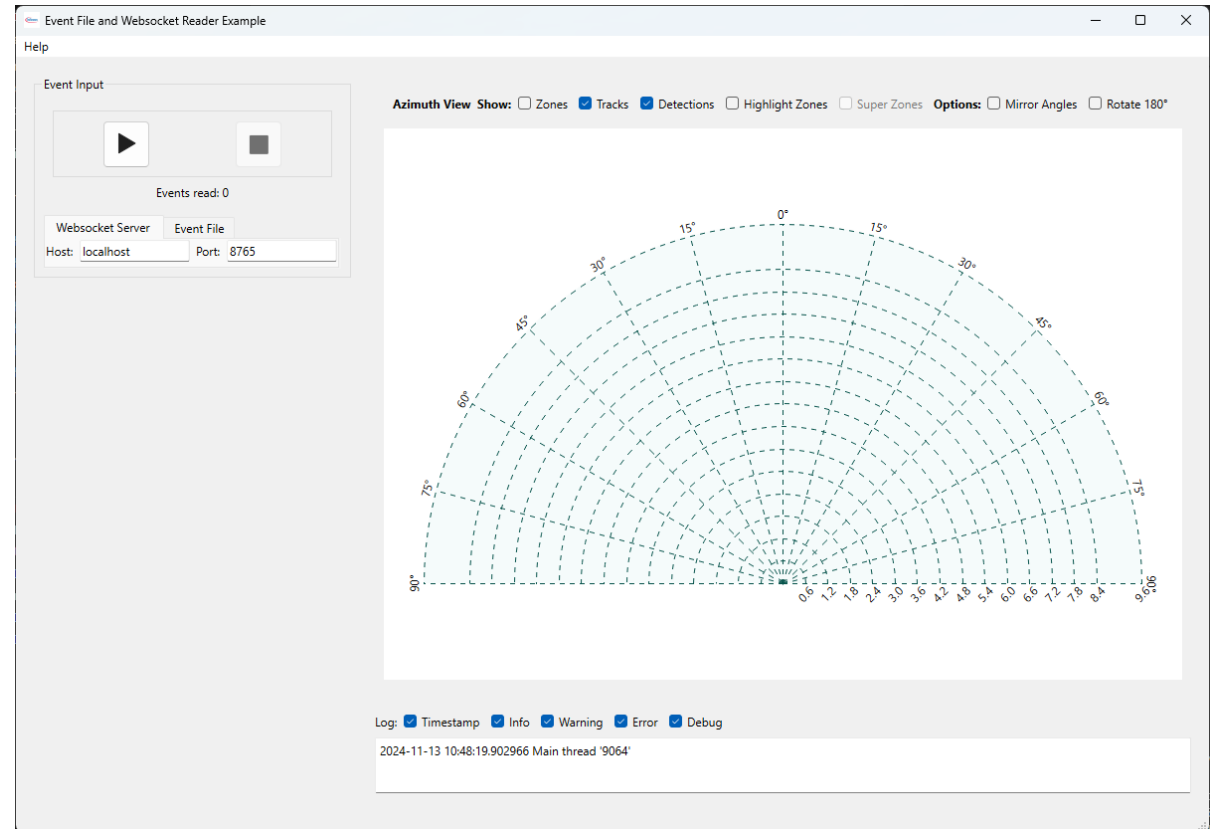
- Code refactoring (use design patterns, mitigate technical debts etc.)
- Increase code coverage (even for a GUI project it is too low)
- Implement new features

Ifxapp: Demos

Demos

Demos can be found under [src/ifxapp/examples/](#) folder:

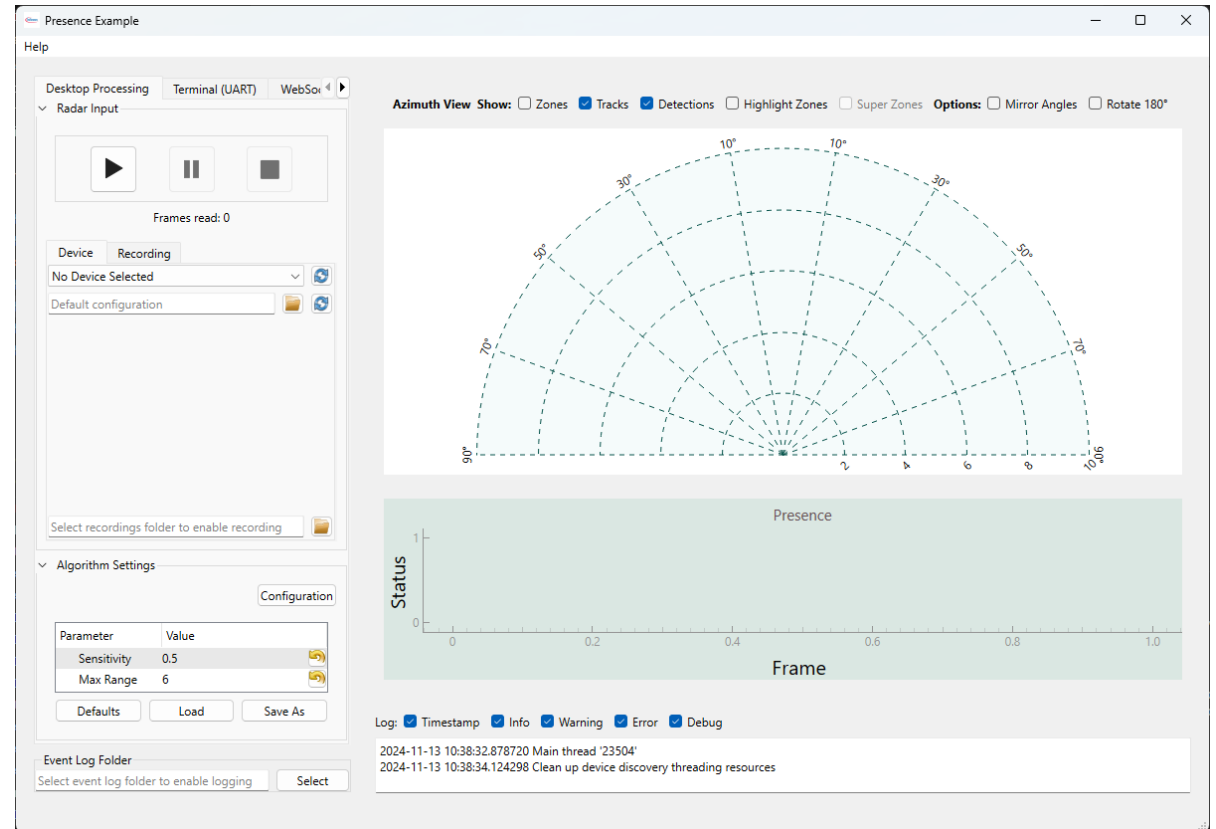
- [event_file_and_websocket_reader_example.py](#)
- [presence_example.py](#)
- [smartar_motion_detector_example.py](#)



Demos

Demos can be found under [src/ixapp/examples/](#) folder:

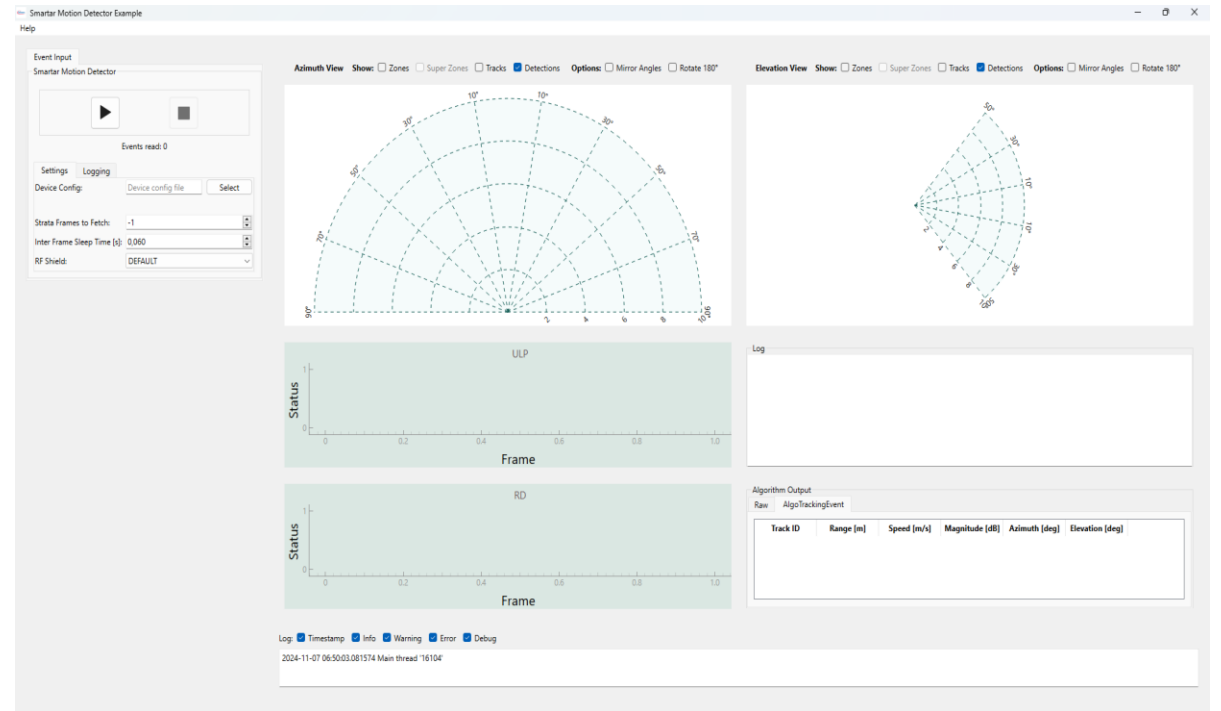
- `event_file_and_websocket_reader_example.py`
- [presence_example.py](#)
- `smartar_motion_detector_example.py`



Demos

Demos can be found under [src/ixapp/examples/](#) folder:

- [event_file_and_websocket_reader_example.py](#)
- [presence_example.py](#)
- [smarter_motion_detector_example.py](#)



How to get started

Try out ifxapp applications and GUI widgets:

1. Clone and initialize the repo and install ifxapp Python dependencies from `requirements.txt`

```
git clone git@gitlab.acme.com:pss/ifxapp.git
cd ifxapp
./scripts/configure_repo.sh
code vscode.code-workspace &
```

Note: On Windows, use "Git Bash for Windows" to execute these commands.

2. Start a demo application:

```
source .venv-3.9/Scripts/activate
python src/ifxapp/examples/event_file_and_websocket_reader_example.py
```

3. Start individual ifxapp widgets, for instance:

```
source .venv-3.9/Scripts/activate
python src/ifxapp/gui/event_consumers/segment_plot.py
```