Red Pitaya Logic Analyzer

Alexander Schmid, Sebastian Ederer, Michael Schneider, Florian Henneke

OSTBAYERISCHE TECHNISCHE HOCHSCHULE REGENSBURG

ABSTRACT

The Red Pitaya is a single board computer for electrical measurement that contains an FPGA in addition to the CPU. In this project a logic analyzer software was developed for the board. The logic analyzer software can be accessed via a web interface. Over the course of this project, the server side application, the user interface as well as the data structures for communication over a web-socket connection were developed.

MOTIVATION

Until now, there are no logic analyzer solutions that can be accessed via mobile or tablet devices. The Red Pitaya offers the infrastructure for web applications out of the box as well as a built in logic analyzer. However, the built in logic analyzer only supports very few protocols and is also closed source. Therefore, it cannot be extended to decode more protocols. In order to decode all commonly used protocols, including the CAN bus, a new logic analyzer software is developed for the Red Pitaya from the ground up.

OBJECTIVES

The new logic analyzer shall be accessible on mobile and tablet devices in addition to a desktop computer. Furthermore, it should be able to decode all commonly used protocols and extending the logic analyzer to support new protocols should be possible.

CONCEPT

A web app for the Red Pitaya always consists of both frontend as well as the backend. The frontend consists of HTML and JavaScript that is processed by the clients web browser. Whereas, the backend consists of C++ code executed on the Red Pitaya itself. The frontend and the backend communicate via a web socket connection over which JSON encoded data structures are sent.

The tasks of a logic analyzer include:

- The acquisition of data of an external source.
- The decoding of that data according to a certain protocol by taking the users settings into account.
- Displaying the raw and decoded data.

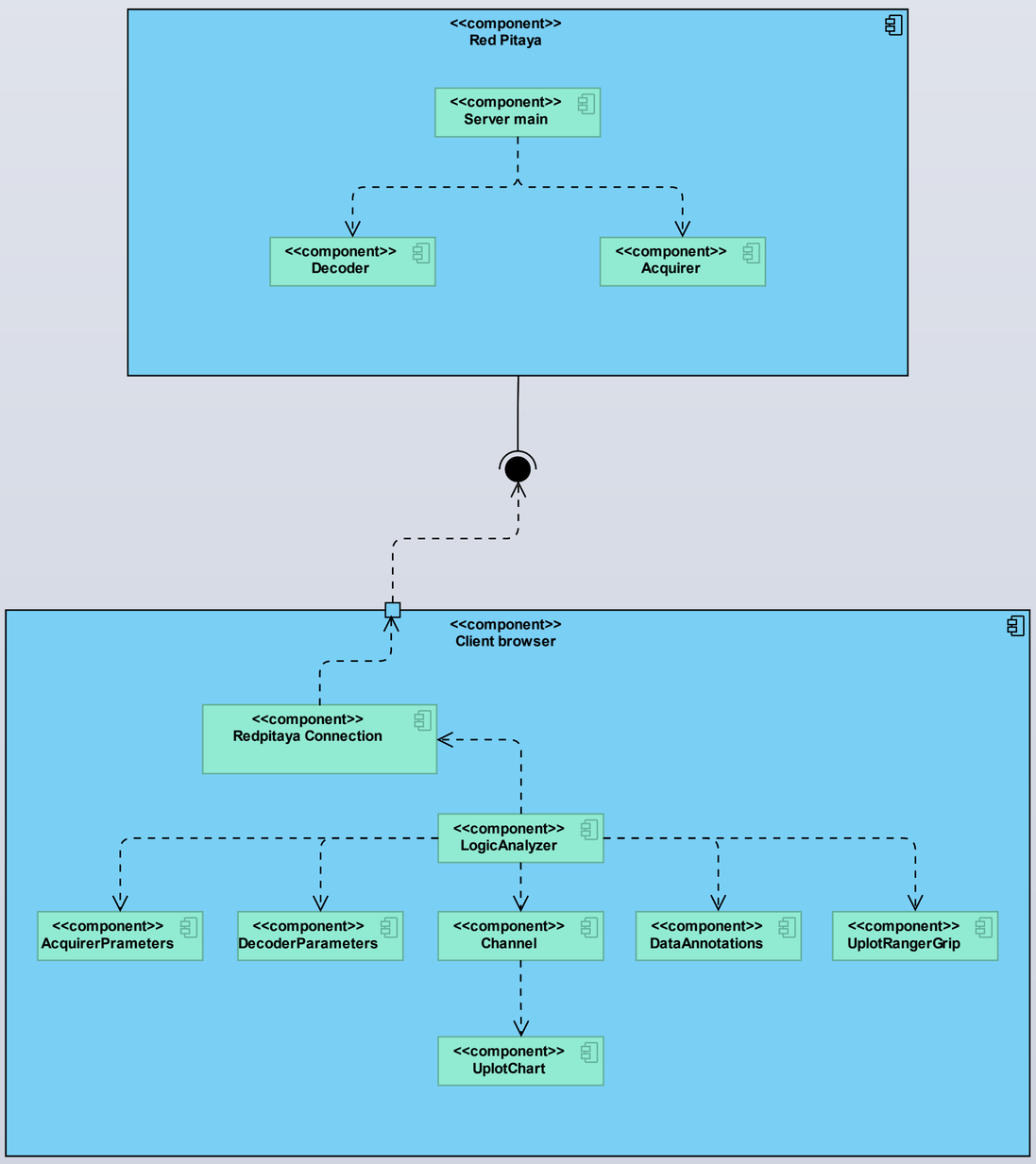

Fig. 1: Logic analyzer component diagram

IMPLEMENTATION and RESULTS

Acquisition:

The acquisition is performed by using the Red Pitaya's analog high frequency inputs that are processed and buffered by the FPGA. The entire buffer can then be read out by the CPU and sent to the client's device over the web socket connection. Before the acquisition is performed, the user is able to adjust different parameters of the acquisition such as the sample rate, the number of samples or the gain of each input channel in the web UI.

Decoding:

After acquiring the data, they get decoded according to the protocol and protocols settings, specified by the user in the web UI. The decoding is performed by the libsigrokdecode library running on the Red Pitaya. libsigrokdecode is a C-library that can invoke several protocol decoders written in Python. It currently supports 130 different protocols out of the box, why it was chosen as the method for decoding data on the Red Pitaya instead of decoding the data in the web UIs JavaScript. The decoded data then gets sent to the frontend via the web socket connection where it gets displayed.

Displaying:

The data are displayed using the uPlot library. uPlot was chosen over other chart libraries as it is quite small and fast in comparison. With around 40KB in size it can create a chart containing 150 000 data points in 135ms. Unlike some other chart libraries, uPlot uses WebGL instead of SVG based chart rendering, which is usually more performant.

UI:

Tying all these three tasks together is the UI, implemented using Vue 3. Vue is a modern JavaScript framework for frontend development that is lightweight, has a large community, good documentation and is simple as well as flexible. Moreover, Bootstrap, a CSS library, is used for responsive design in order to display the UI according to the size of the client's screen.

Current status:

At the date of writing this poster, the following features have been implemented fully:

- Setting all relevant acquirer options sampling up to 16384 samples from the two analog inputs.
- For all decoders in libsigrokdecode: querying and setting the decoder options and decoder channels. At this point, only the UART decoder has been fully tested and confirmed working.
- Displaying the acquired data along with the decoder annotations as well as scrolling and zooming synchronously between the various channel's data.
- Support for touchscreen-based devices.

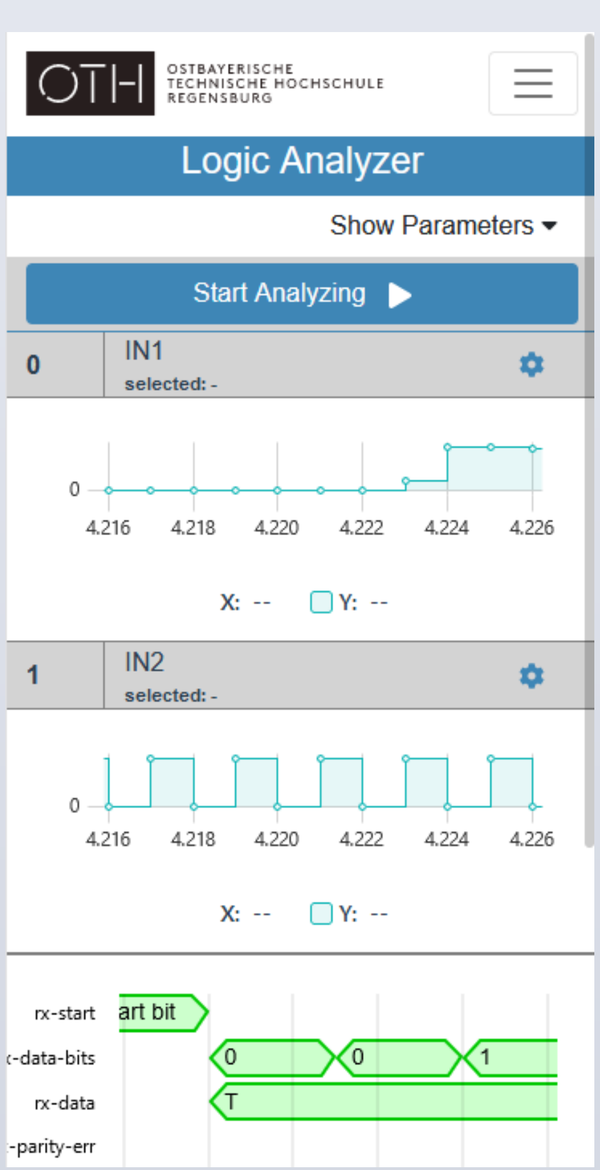

Fig. 2: Mobile view of the logic analyzer

REFERENCES

- https://github.com/leeoniya/uPlot
- https://sigrok.org/wiki/Protocol_decoders
- https://redpitaya.com/stemlab-125-10/
- https://www.codeinwp.com/blog/angular-vs-vue-vs-react/

CONTACT

Prof. Dr. Daniel Muench
daniel-muench@oth-regensburg.de