Documentation Red Pitaya

# Introduction and Goal (Alexander Schmid)

The Red Pitaya STEMlab 125-10 is a “Multifunction Lab Instrument”.

More concretely, it is a single-board computer consisting of a dual-core ARM Cortex-A9 MPCore CPU, a Xilinx Zynq 7010 FPGA, a number of digital IO pins and four coaxial RF I/Os (two inputs and two outputs).

The Red Pitaya runs a custom Ubuntu-based operating system with an nginx-based web server. This allows the user to connect to the Red Pitaya over a local network and, via a web-browser, interact with a number of included applications. Among these applications are an oscilloscope, signal generator, logic analyzer and others.

The included logic analyzer supports reading data from the Red Pitaya’s digital I/O-pins and decoding the I2C, SPI and UART protocols. This limited number of supported protocols has proven to be insufficient.

The goal of this project is to support the analyzing of additional protocols with the Red Pitaya, especially the CAN protocol.

Because the included logic analyzer application is closed-source, this will necessitate building a new logic analyzer application from scratch.

# Implementation

## The Red Pitaya’s architecture

## Building the backend

## Data acquisition

## Analyzing the acquired data

## The User Interface

# Current State and Future Directions

# Important/helpful links to get start or to deal with problems

Information about rp.h : [https://github.com/RedPitaya/RedPitaya/blob/master/api/include/redpitaya/rp.h#L145](https://github.com/RedPitaya/RedPitaya/blob/master/api/include/redpitaya/rp.h" \l "L145)

Examples of RF Input Output: [https://redpitaya.readthedocs.io/en/latest/appsFeatures/examples/genRF-exm1.html#code-c](https://redpitaya.readthedocs.io/en/latest/appsFeatures/examples/genRF-exm1.html" \l "code-c)

Information about IOs: <https://redpitaya.readthedocs.io/en/latest/developerGuide/125-14/fastIO.html>

Specs of the Redpitaya: <https://www.redpitaya.com/f146/specifications>

Sampling rates, etc: [https://redpitaya.readthedocs.io/en/latest/appsFeatures/examples/acqRF-samp-and-dec.html#s-rate-and-dec](https://redpitaya.readthedocs.io/en/latest/appsFeatures/examples/acqRF-samp-and-dec.html" \l "s-rate-and-dec)

Ringbuffer stuff: <https://github.com/RedPitaya/RedPitaya/issues/100>

# Procedure Initial Start

# Recurrent work

## Building App

Alex, da hast du dein Video, vllt einfach kurz beschreiben oder/und aufs video verweisen

# Known issues and workaround

## The included applications do not start, the progress ring simply keeps spinning

If the browser’s javascript console shows „Uncaught ReferenceError: AnalyticsCore is not defined“, you most likely have an ad-blocker enabled in your browser. The included applications will not work with an ad-blocker. Simply disable it for your Red Pitaya’s URL and reload the page.

## Filesystem is read-only

Call `rw` inside of the RP console to make writeable

## Call the Redpitaya webserver fails within Windows

## SSH connection breaks

Simple restart should work. We don’t know why it happens but maybe it’s a temperature problem.

## Acquiring data never stops

If you call the “rp\_App\_Init()” to initialize the Webapp, replace it with “rp\_Init()”. Until now we didn’t notice any different behavior, only the acquiring process stops correctly.

To stop the acquiring process, its needed to push the FPGA image again via “cat /opt/redpitaya/fpga/fpga\_0.94.bit > /dev/xdevcfg”.

## Ws.app is null by starting up the Webapp

The name of the application (in the app.js) must be the same as the name of the folder

## Cannot read fpga file

Maybe you modified the fpga.conf of your project in windows, than everytime you open the file in nano, the file gets parsed from “DOT format” and somehow the path gets corrupted. Just copy a file from a working project or build up the file from scratch, but do it in Linux.

# Our VPN solution for „home office“

1. Build and configure a wireguard server
2. Build certificates for every client.
3. Install wireguard client software on your pc
4. Wireguard doesn’t work on the redpitaya directly therefore we did a little workaround
   1. Install wireguard on a raspberry or similar (following link should contain tutorial)
      1. https://www.sigmdel.ca/michel/ha/wireguard/wireguard\_02\_en.html
   2. Configure and connect the raspi
   3. Activate vnc server on the Raspberry
   4. Connect the redpitaya via lan to the raspberry
   5. You should now be able to control the raspi via vncViewer
      1. It is very important to use (real vncViewer) other clients didn’t work in our tests (https://www.realvnc.com/de/connect/download/viewer/)
   6. And on the raspi the Redpitaya should be available
   7. Finally build autostart (build in wireguard service)
      1. https://www.sigmdel.ca/michel/ha/wireguard/wireguard\_02\_en.html

4.1 Install (Ubuntu/Mint/Arch/Manjaro):

* `sudo apt install wireguard` (Ubuntu/Mint)
* `sudo pacman -s wireguard-tools` (Arch/Manjaro)
* Save .conf file to /etc/wireguard
* To start VPN: `wg-quick up <nameofconf>`
* To stop VPN: `wg-quick down <nameofconf>`
* If error `resolvconf: command not found` run `sudo ln -s /usr/bin/resolvectl /usr/local/bin/resolvconf`
* `sudo wg show all` shows the current connection

4.2 Install (Windows)

* Install: https://download.wireguard.com/windows-client/wireguard-installer.exe
* Run and import config file