RedPitaya Anleitung

# Toolchain

Benötigte Tools unter Windows:

* Putty (<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>)
* WinSCP (<https://winscp.net/eng/download.php>)
* Vivado (siehe Vivado Install)
* Visual Studio Code [https://code.visualstudio.com/](https://code.visualstudio.com/O) oder ein anderes Tool zum Bearbeiten von C Code z.B. Notepad++
* Python + IDE (z.B. <http://winpython.github.io/> )
  + Download + Ausführen
  + Ordner an gewünschten Ort kopieren
  + WinPython Control Pannel.exe ausführen 🡪 Advanced 🡪 Register distribution…

1. LAN Verbindung mit RedPitaya herstellen (PC muss IP aus dem Adressraum 192.168.178.x haben)
2. Ping an 192.168.178.15 oder im Browser eingeben (wenn erfolgreich weiter mit 4. sonst 3.
3. Laut <https://redpitaya.readthedocs.io/en/latest/quickStart/connect/connect.html> verbinden und IP auf 192.168.178.15 ändern (wenn mehrere RedPitaya in einem Netz verschiedene IP‘s vergeben und diese notieren notieren!)
4. Via Putty (SSH) auf RedPitaya zugreifen   
   Host IP: 192.168.178.15  
   Port: 22  
   Connection type: SSH   
   Benutzer: root  
   Password: root

# Tutorial

Siehe <http://antonpotocnik.com/?p=487360>

# Software desigen

## IP Cores

* Slice 🡪 vewenden einzelner bit’s / abschneiden von bit’s
* Binary Counter 🡪 zählt binärzahl am Ausgang hoch
* AXI GPIO 🡪 GPIO Interface zu Prozessor (Run Connection Automation fügt weitere benötigte Blocks hinzu)  
  alternativ kann „axi\_configuration\_v1\_0“ genutzt werden
  + Adresse über Adress Editor einstellbar
  + Über Konsole lesen (monitor „adresse“)
  + Über Konsole schreiben (monitor „adresse“ „wert“)
  + In C Datei lesen/schreiben
    - void \*adress; //create pointer
    - adress=mmap(NULL, sysconf(\_SC\_PAGESIZE), PROT\_READ|PROT\_WRITE, MAP\_SHARED, fd, 0xXXXXXXXX); //map memory
    - \*((uint32\_t \*)(adress)) = XXX; //write to buffer
    - value = \*((uint32\_t \*)(adress)); //read from buffer (GPIO1)
    - value = \*((uint32\_t \*)(address+8)); //read from buffer (GPIO2)
    - munmap(adress, sysconf(\_SC\_PAGESIZE)); //unmap memory

## Custom IP Cores

Um die custom Cores verfügbar zu machen muss das make\_cores.tcl Script mit „source make\_cores.tcl“ in Vivado ausgeführt werden, hierzu muss vorher mit cd in den root Ordner navigiert werden.

Um eigene Cores zu erstellen

* Template in core Ordner kopieren
* Verilog Code schreiben
* core\_config.tcl anpassen
* core testen
* in information\IP\_Core\_description.docx dokumentieren
* source make\_cores.tcl ausführen

Alternative können auch in den Projekten nur die IP Cores erstellt werden welche benötigt werden. Hierzu muss folgende Zeile ein kommentiert werden „set cores [list \ core1 \ core2]” und die Zeile „set cores [list]“ auskommentiert. Für core1 und core2 können dann die Namen der Core Ordner eingefügt werden. Eventuell muss vorher der cores Ordner aus tmp gelöscht werden.

## Eigener Code (siehe <http://antonpotocnik.com/?p=488784>)

1. Project Manager 🡪 Add Sources 🡪 Add or create design sources
2. Create File 🡪 Typ auswählen 🡪 Namen geben
3. I/O Port Definitions sind optional, diese können Später auch in der Datei vergeben werden
4. Modul testen
5. Einbinden in Bockdiagramm: rechtsklick🡪 Add Module…

# Programming FPGA

1. \*.bit Datei mit WinSCP in RedPitaya kopieren (zu finden in tmp/Project/\*.runs/impl)
2. Ausführen mit (in Konsole des RedPitaya 🡪 Putty)
   1. Putty starten und per SSH auf RedPitaya zugreifen
   2. Bitfile laden z.B.  
      cat /opt/redpitaya/fpga/fpga\_X.XX.bit > /dev/xdevcfg  
      🡨 das führt die original FPGA Software aus  
      bzw.: cat /usr/src/bitfile.bit > /dev/xdevcfg  
      🡨 wenn das eigene file in usr/src/ liegt
   3. Das bitfile muss nach reboot neu geladen werden
      1. Alternativ kann es in einer \*.c datei mit folgendem Befehl geladen werden  
         system("cat /opt/redpitaya/fpga/UscopeV0.04.bit > /dev/xdevcfg");
      2. Oder per Crontab beim booten
         1. [sudo apt-get update]
         2. [sudo apt-get install cron]
         3. crontab -e
         4. eine Zeile mit folgendem Inhalt einfügen:  
            @reboot cat /opt/redpitaya/fpga/fpga\_X.XX.bit > /dev/xdevcfg
         5. Achtung! Damit die Standardfunktionen des RedPitaya wieder funktionieren muss die Zeile gelöscht werden

# Daten mit PC auslesen (siehe <http://antonpotocnik.com/?p=514765>)

## Server

Zum Auslesen der Daten über die Ethernet Schnittstelle muss ein Server auf dem RedPitaya angelegt werden. Hierzu kann das Server Template verwendet werden. Anpassungen an das Bitfile auf dem FPGA und das Python Programm auf dem Rechner sind in folgenden Zeilen nötig:

45 🡪 ein kommentieren um Bitfile automatisch zu laden und Name anpassen  
 54/55 🡪 Pointer auf die verwendeten Adressen anpassen, ergänzen, löschen  
 94ff 🡪 case Anweisungen anpassen  
 119 🡪 Methode um auf zu zeigen, dass Messung beendet ist anpassen  
 122ff 🡪 Daten auslesen anpassen

Starten des Servers:

* Server.c mit WinSCP in RedPitaya kopieren (/usr/src)
* Mit „gcc -o server /usr/src/server.c“ kompilieren (via Putty)
* Mit „./server“ starten (via Putty)

Automatisch beim Hochfahren ausführen:

* In Putty Konsole crontab -e
* Zeile „@reboot ./server“ einfügen
* Achtung! Damit die Standardfunktionen des RedPitaya wieder funktionieren muss die Zeile gelöscht werden

## Python

Um die vom Server zur Verfügung gestellten Daten aus zu lesen, kann das „client\_connection.py“ Template genutzt werden. Dieses kann durch eine GUI z.B. mittels PyQt erweitert werden.

# Python user interface

Tutorial: <https://realpython.com/qt-designer-python/#getting-started-with-qt-designer>

<https://www.youtube.com/watch?v=rZcdhles6vQ&list=PLCC34OHNcOtpmCA8s_dpPMvQLyHbvxocY>

Installation (in WinPython command prompt eingeben):

* pip install PyQt5
* pip install PyQt5Designer
* möglicherweise pip3 install --upgrade matplotlib
* desinger.exe ist im Scripts Ordner der Python Installation zu finden z.B.: C:\WPy64-3950\python-3.9.5.amd64\Scripts

Code aus \*.ui Datei generieren (in WinPython command prompt eingeben):

* pyuic5 -x <source> -o <output> (wobei source und output den Dateinamen inklusive Pfad entsprechen es ist auch möglich vorher in den Ordner zu navigieren)
* Die generierte Datei sollte nicht geändert werden, da bei erneutem Generieren alle Änderungen verloren gehen

Matplotlib Diagramm einfügen:

* Widget Container einfügen
* Objekt benennen
* Nach einbinden kann dieser Name als Methode der Klasse Ui\_MainWindow verwendet werden.
* Somit kann ein Matplotlib Diagramm in den Container eingebunden werden (siehe „templateMain.py“)

Server Client:

* Der Server auf dem RedPitaya kann über einen TCP Socket angesprochen werden, hierzu wird im Template „clientConnection.py“ eine Klasse angelegt
* Im „templateMain.py“ wird ein Beispiel gezeigt wie diese Klasse zu verwenden ist.