Anleitung Pynq-Z1

Inhaltsverzeichnis

1 Übersicht und Aufbau

Das **PYNQ-Z1** besteht aus einem **Zynq-XC7Z020-1CLG400C SoC**, welches wiederum aus einer **ARM Cortex-A9 CPU** sowie einem **Artix-7 FPGA** besteht. Das Board verfügt über **512 MB DDR3-RAM**. Der FPGA kann mittels des **PYNQ**-Frameworks in Python programmiert werden, ohne dass vorher - wie bei klassischem Hardware-design - programmierbare Logikschaltungen entworfen werden müssen. Statt dessen wird der FPGA über (hardware) Bibliotheken sowie deren APIs angesprochen. [1]

Die Software des Z1 läuft auf einem ARM Cortex-A9 (ARMv7-A) 2-Kern-2-Thread Prozessor, welcher mit 650MHz taktet und umfasst

- Ein Linux-Betriebssystem, welches Python bereitstellt.
- Jupyter Notebook zum programmieren des Z1 / FPGAs.
- Hardwarebibliotheken sowie APIs für den FPGA.

1.1 Integrierte Hardware

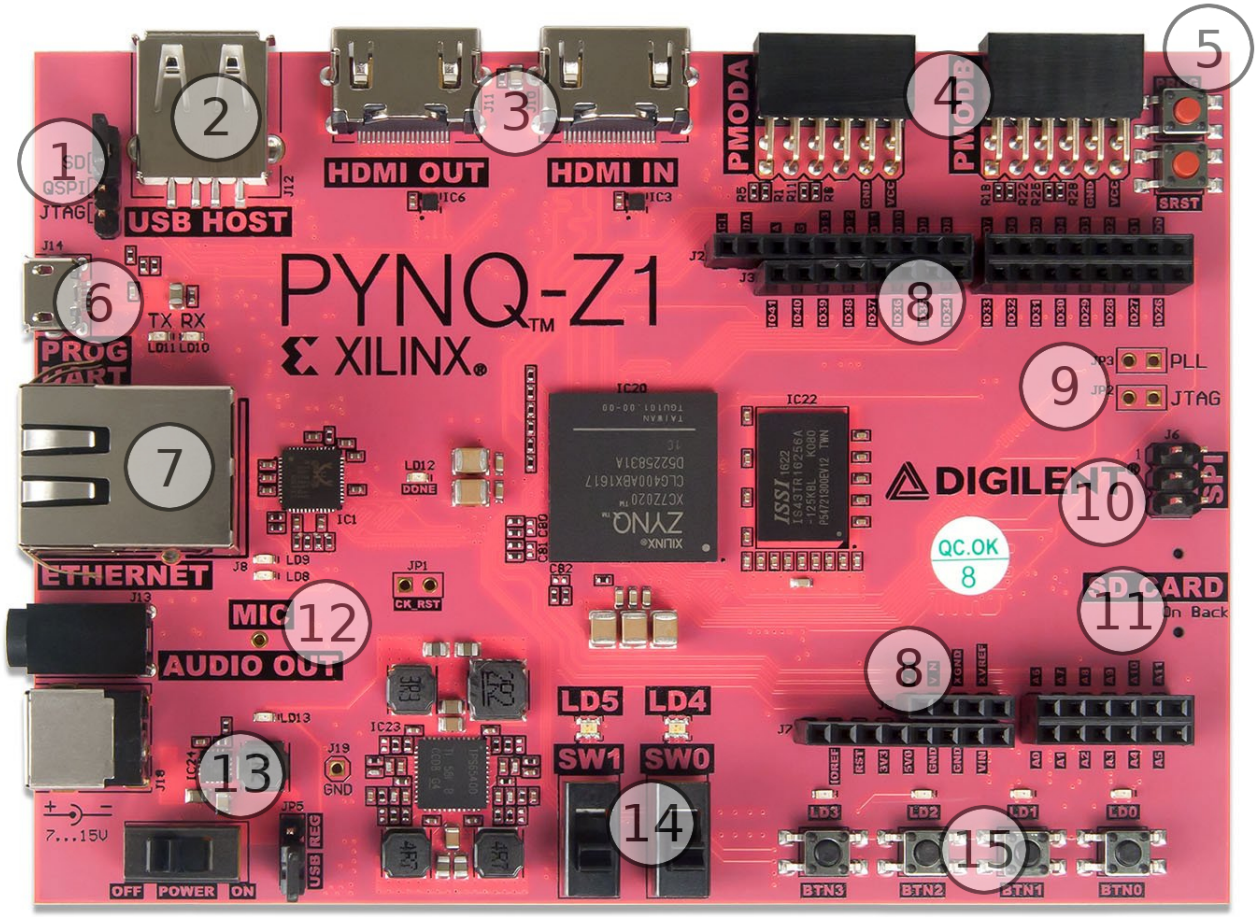

Abbildung 1: Übersicht PYNQ-Z1

- **1 - Boot Jumper**
  Wird zum wählen des Bootmodus gesetzt. Das Board kann über **SD-Karte**, **Quad SPI** oder **JTAG** gebootet werden.
- **2 - USB HOST**
  USB 2.0 Controller des Boards. Hier kann USB-Hardware angeschlossen werden.

- **3 - HDMI OUT/IN**
  HDMI Type-A Ports, welche direkt mit dem FPGA verbunden sind. Die Ports sind mit DVI-D kompatibel.
- **4 - PMOD A/B**
  High-Speed Pmod ports, welche 2 3.3V, 2 GND sowie 8 Logik-Signale bereitstellen.

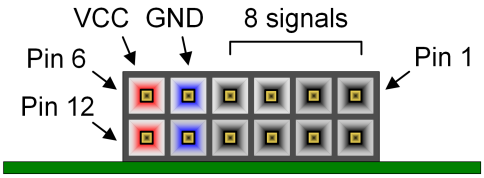

Abbildung 2: Pinbelegung Pmod Ports

Achtung: Da die Pins nicht gegen Kurzschluss oder Überspannung (**>3.3V**) geschützt sind, muss beim verdrahten besonders aufgepasst werden.

- **5 - PROG/SRST**
  Über den **PROG**-Knopf wird die PL zurückgesetzt und muss erneut konfiguriert werden.
  Mittels des **SRST**-Knopfes wird das gesamte System (inklusive angebrachter Shields, welche auf das **CK_RST**-Signal reagieren) zurückgesetzt. Effektiv wird hierdurch ein Neustart des Boards erzwungen - wobei der Bootmodus nicht erneut überprüft wird.
- **6 - PROG/UART**
  Dient zur Kommunikation über UART (**115200-8-1-N-N**) und zur Stromversorgung des Boards.
- **7 - ETHERNET**
  10/100/1000 FDX Netzwerkschnittstelle. Das Board versucht nach dem Anschalten über DHCP eine IP-Adresse zu erhalten und fällt auf **192.168.2.99** zurück, falls kein DHCP Server verfügbar ist.
- **8 - Arduino / chipKIT Shield Header**
  Header, welcher speziell für Arduino / chipKIT Shields entworfen wurde.

  Achtung: Es dürfen keine Shields benutzt werden, die einen digitalen / analogen Output **>3.3V** haben, da dies das Board beschädigen kann.
- **9 - PLL/JTAG**
  Wenn das Board zum Booten über JTAG konfiguriert ist (Siehe **1.1, Boot Jumper**), kann mittels **JP2 (JTAG)** zwischen **Cascaded JTAG Chain Mode** und **Independent JTAG Boot Mode** gewechselt werden.
- **10 - SPI**
  Dient zur Kommunikation über SPI. - beispielsweise für Shields.

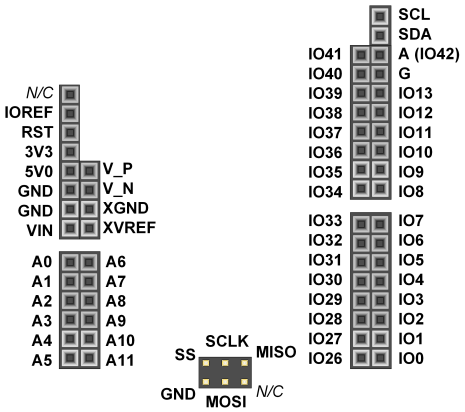

Abbildung 3: Pinbelegung Shield / SPI-Header

- **11 - SDCARD**
  Steckplatz für MicroSD Karten, auf denen beispielsweise das Betriebssystem gespeichert ist und gebootet wird. Für das Betriebssystem wird mindestens eine Klasse-4 Karte mit 8 GB Speicherplatz empfohlen.
  Siehe auch Abschnitt 2.

- **12 - MIC/AUDIO OUT**
  Hier befinden sich sowohl ein Mono-Audioausgang, welcher direkt vom Zynq FPGA angesteuert werden kann, als auch ein omnidirektionales MEMS (microelectro-mechanical system) Mikrofon, welches ein Pulsdichte-moduliertes Signal an den FPGA abgibt.

- **13 - Ext. Power, Power Switch, Power Jumper**
  Über den **Power Jumper** wird eingestellt, ob das Board über Anschluss **J14 (PROG / UART)**, Stellung **USB**, oder über Anschluss **J18 (External Power)**, Stellung **REG** mit Strom versorgt wird.
  Externe Netzteile müssen folgende Voraussetzungen erfüllen:

  – Hohlstecker mit positivem Innenleiter und 2,1mm Innendurchmesser.

  – Ausgangsspannung zwischen 7VDC und 15VDC.

  Eingangsspannungen >15VDC können das Board beschädigen.

- **14 - Schalter, RGB LEDs**
  2 Schiebeschalter sowie 2 RGB-LEDs, welche direkt im dem Zynq FPGA interagieren können. Die Schalter sind gegen Kurzschlüsse gesichert (welche beispielsweise auftreten können wenn ein FPGA Pin, der an einem Schalter liegt, als Output definiert wird).
- **15 - Taster, LEDs**
  Taser nur grüne LEDS, welche ebenfalls direkt mit dem Tynq FPGA verbunden sind. Wie die Schiebeschalter sind auch die Taster gegen Kurzschlüsse gesichert.

2 Vorbereiten des Boards

2.1 Voraussetzungen

Zum Einrichten des Boards wird mindestens folgendes benötigt:

- MicroSD Karte mit mindestens 8 GB Speicherplatz.
- MicroUSB Kabel.
- Ethernetkabel.
- Pynq Z1 Image. [2]
- Eine Möglichkeit, auf MicroSD Karten zu schreiben (und gegebenenfalls SD->MicroSD Adapter).

Ferner wird ein Programm benötigt, welches Dateisystem-Abbilder auf SD-Karten schreiben kann. Das Beschreiben wird im folgenden mit dem **Pynq Z1 v2.1** Image anhand einiger Beispiele erklärt.

2.2 Einrichten der MicroSD Karte

2.2.1 Linux / OSX - *dd*

Zum beschreiben der Karte mit *dd* sind folgende Schritte auszuführen:

- Ein Terminal öffnen.
- Herunterladen des Images mittels *wget*:

```
wget http://files.digilent.com/Products/PYNQ/pynq_z1_v2.1.img.zip
```

- Entpacken des Archives:

```
unzip xzvf pynq_z1_v2.1.img.zip
```

- Herausfinden des Laufwerkes:
  Zunächst muss bestimmt werden, welches Gerät für die SD Karte angelegt wird. Hierfür kann nach dem Verbinden *dmesg* benutzt werden:

```
dmesg
```

  Es sollte dann folgende (bzw. ähnliche) Ausgabe zu sehen sein:

```
usb-storage 1-1.1:1.0: USB Mass Storage device detected
scsi host6: usb-storage 1-1.1:1.0
scsi 6:0:0:0: Direct-Access Mass Storage Device 1.00 PQ: 0 ANSI: 0 CCS
sd 6:0:0:0: Attached scsi generic sg2 type 0
sd 6:0:0:0: [sdb] 7744512 512-byte logical blocks: (3.97 GB/3.69 GiB)
sd 6:0:0:0: [sdb] Write Protect is off
sd 6:0:0:0: [sdb] Mode Sense: 03 00 00 00
sd 6:0:0:0: [sdb] No Caching mode page found
sd 6:0:0:0: [sdb] Assuming drive cache: write through
sdb: sdb1 sdb2
sd 6:0:0:0: [sdb] Attached SCSI removable disk
EXT4-fs (sdb2): 24 orphan inodes deleted
EXT4-fs (sdb2): recovery complete
EXT4-fs (sdb2): mounted filesystem with ordered data mode. Opts: (null)
```

Listing 1: Beispielausgabe für MicroSD-USB Adapter. Manche Kartenleser zeigen statt **/dev/sd* /dev/mmcblk*** o.ä. an.

Hierbei ist Zeile 10 zu beachten, welche die gefundenen Partitionen anzeigt. Bei einer neuen / unformatierten Karte sind in der Regel keine Partitionen angelegt.

- Kopieren des Images:
  Wenn die Karte Partitionen enthält welche automatsich eingehangen wurden, müssen diese zunächst mittels *umount* ausgehangen werden. Am oben genanten Beispiel wäre dann

```
1  umount /dev/sdb?
```

  auszuführen.
  Mittels *dd* kann jetzt das Image auf die Karte geschrieben werden:

```
1  sudo dd if=pynq_z1_v2.1.img of=/dev/sdb bs=4M status=progress
```

  Wichtig: Das Image muss direkt auf das Gerät kopiert werden, nicht auf eine Partition. Der Vorgang kann einige Zeit in Anspruch nehmen. Wenn *dd* fertig ist, kann die Karte entfernt und mit **3 - Inbetriebnahme** fortgefahren werden.

2.2.2 Windows - *Win32 Disk Imager*

Win32 Disk Imager [3] ist ein Windows-Tool zum beschreiben von entfernbaren Laufwerken bzw. zur Sicherung von Abbildern entfernbarer Laufwerke.

- Herunterladen des Pynq Images:
  Das Image kann mittels Webbrowser über die URL `http://files.digilent.com/Products/PYNQ/pynq_z1_v2.1.img.zip` bezogen werden. Nach dem Herunterladen muss das Image noch entpackt werden.
- Anschließen der MicroSD-Karte:
  Die MicroSD-Karte kann nun entweder über ein Kartenlesegerät oder über einen USB-Adapter angeschlossen werden. Nach dem anschließen wird der Karte von Windows ein Laufwerksbuchstabe zugewiesen. Wir gehen im weiteren davon aus, dass der Karte das Laufwerk **I:** zugewiesen wurde.
- Schreiben des Images auf die MicroSD-Karte:
  Nach dem staren von *Win32 Disk Imager* wird folgendes Fenster angezeigt:

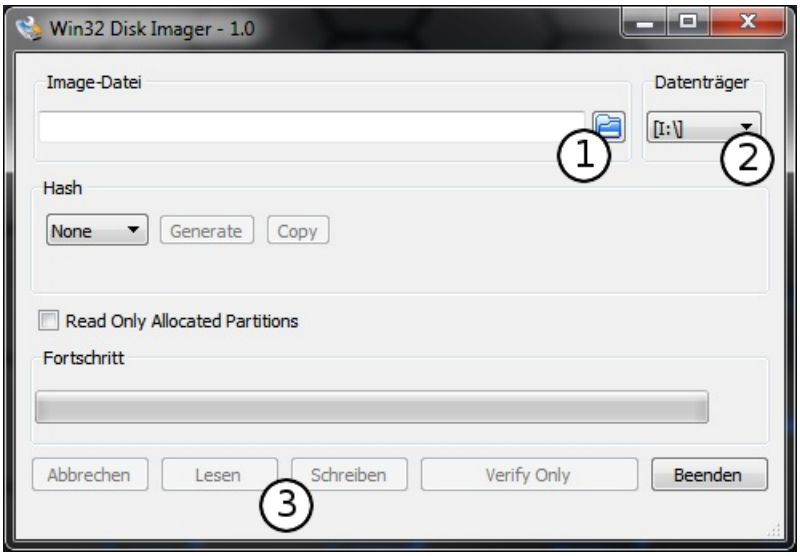

Abbildung 4: Win32 Disk Imager v1.0.0

Die für das Beschreiben wichtigen Schlatflächen sind unter **1**, **2** und **3** angezeigt:

- **1** - Hier wird über das Ordner-Icon der Pfad zur Image-Datei eingestellt.
- **2** - Hier wird der Laufwerksbuchstabe des Ziellaufwerkes angegeben.

– **3** - Über diese Schaltflächen wird unter anderem das Schreiben gestartet.

Über Schlatfläche **1** wird nun zum Image navigiert. Als Beispiel sei hier der Pfad
E:/Downloads/pynq_z1_v2.1.img/pynq_z1_v2.1.img
gegeben.
Über Schaltfläche **2** wird das der Karte zugewiesene Laufwerk ausgewählt. In unserem Fall wäre dies **I:**. (Siehe 2.2.2 - Anschließen der MicroSD Karte).

Durch einen Klick auf **Schreiben** wird der Schreibvorgang gestartet. Dieser kann einige Zeit in Anspruch nehmen. Nach dem Schreiben kann die Karte entfernt und mit **3 - Inbetriebnahme** fortgefahren werden.

3 Inbetriebnahme

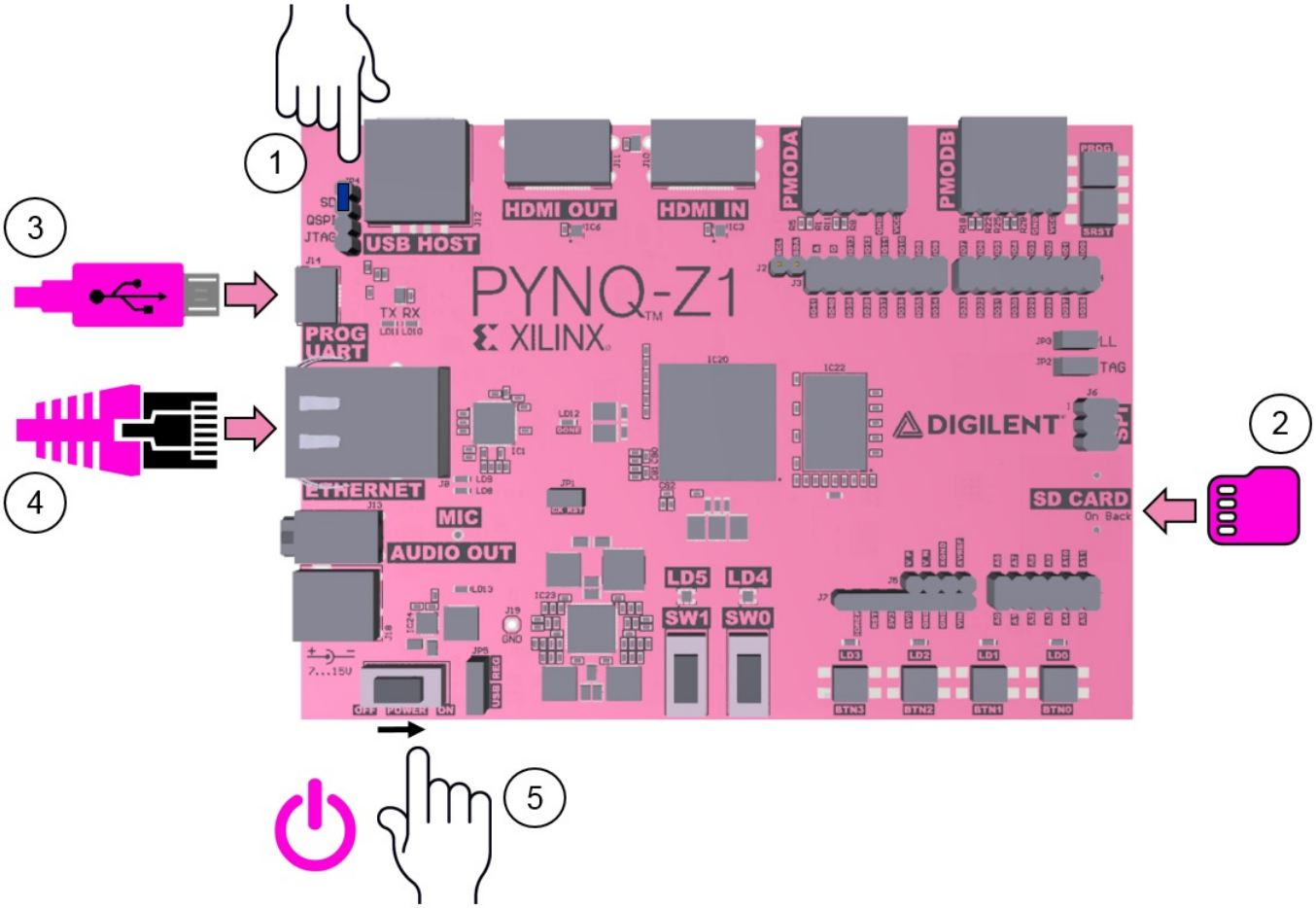

Abbildung 5: Schritte zur Inbetriebnahme

Zur Inbetriebnahme mit Stromversorgung über USB sind grundsätzliche folgende Schritte auszuführen:

- Einstellen der Bootmethode. Siehe 3.1.
- Einsetzen der MicroSD-Karte.
- Verbinden des MicroUSB Kabels.
- Verbinden des Ethernet-Kabels.
- Einstellen der Stromquelle. Siehe 3.2.
- Einschalten des Boards.

3.1 Bootmethode

Der Bootmodus wird über den Boot Jumper (siehe 1.1) eingestellt. Zum Booten über SD-Karte muss dieser in der Position **SD** (siehe Abbildung 1) sein.

3.2 Einschalten des Boards

Vor dem Einschalten des Boards muss darauf geachtet werden, dass der Power Jumper (siehe 1.1 / Abbildung 1) entsprechend der gewünschten Stromversorgung gesetzt ist. Für Stromversorgung über USB muss der Jumper auf **USB**, für Stromversorgung über externes Netzteil auf **REG** gesetzt sein.

3.2.1 USB

Zur Stromversorgung über USB muss das Board mittels USB-MicroB Kabel mit einer geeigneten Stromversorgung (USB 2.0 Hostcontroller bzw. Ladegerät mit min. 500mA Leistung) verbunden sein. Der **Power Jumper** muss auf **USB** gesetzt sein.

Der USB 2.0 Port des Boards kann selber nur 500mA zur Verfügung stellen. Sollten viele externe Peripheriegeräte bzw. externe USB-Geräte verwendet werden, kann der Stromverbrauch über 500mA steigen. Sollte das Stromlimit erreicht / überschritten werden, wird das Board zurückgesetzt sobald die Betriebsspannung unter einen Mindestwert sinkt.

Für Anwendungen die einen erhöhten Stromverbrauch bzw. keine USB-Quelle in der Nähe haben ist eine Spannungsversorgung über eine externe Stromquelle empfohlen.

3.2.2 Externes Netzteil

Die externe Stromversorgung benötgt ein Netzteil, welches zwischen 7VDC - 15VDC Spannung liefert und über einen Hohlstecker mit 2.1mm Innendurchmesser und positiv gepoltem Innenleiter verfügt. Die Spannungsregelung des Boards findet über die integrierte **TPS65400 PMU** statt.

Neben einem Netzteil kann das Board auch über eine Batterie am Jumper **J7** am **Shield Header** (siehe Abbildung 1) versorgt werden. Der positive Pol muss hierbei mit dem Pin **VIN**, der negative mit dem Pin **GND** verbunden werden.

In beiden Fällen muss der **Power Jumper** in der Stellung **REG** sein.

3.3 Bootvorgang

Nach dem einschalten wird zunächst die rote Power LED **LD13** aufleuchten. Nach einigen Sekunden wird **LD12** aufleuchten um anzuzeigen, dass das Board betriebsbereit ist. Zu diesem Zeitpunkt beginnt der eigentliche Bootvorgang.

Während des Bootens können **LD10** und **LD11** aufleuchten, was UART / serielle Kommunikation anzeigt. Gegebenenfalls zeigen **LD8** und **LD9** Netzwerkaktivität.

Nach etwa 30 Sekunden werden die RGB LEDs **LD4** und **LD5** und die grünen LEDs **LD3**, **LD2**, **LD1** und **LD0** mehrfach aufleuchten. **LD3**, **LD2**, **LD1** und **LD0** bleiben an, **LD4** und **LD5** bleiben aus.

Das Board ist jetzt vollständig hochgefahren und kann benutzt werden. Über einen Webbrowser kann jetzt auf die Jupyter Notebook-Entwicklungsumgebung zugegriffen werden.

Die Adresse ist abhängig davon, ob das Board an einem Netzwerk mit **DHCP** / **DNS** angeschlossen oder statisch adressiert ist.

3.3.1 Adressierung mit DHCP

Sollte das Board an einem Netzwerk mit DHCP-Server angeschlossen sein, wird es versuchen, bei starten eine IP-Adresse vom DHCP-Server zu erhalten. Als Hostname ist standardmäßig **pynq** eingestellt. Das Jupyter Notebook kann nach erfolgreichem Bootvorgang per Browser über `http://pynq:9090` aufgerufen werden.

Sollte die Namensauflösung nicht funktionieren, so muss im DHCP Server nachgesehen werden, welche Adresse vom Board belegt wurde. Entsprechend muss dann auf `http://<Adresse_des_Boards>:9090` zugegriffen werden.

3.3.2 Adressierung ohne DHCP

Sollte das Board an einem Netzwerk ohne DHCP Server angeschlossen sein (beispielsweise direkt an die Ethernet-Schnittstelle eines Computers) so wird es sich nach erfolgloser Suche selbst die Adresse **192.168.2.99/24** zuweisen. Der Schnittstelle am Computer muss entsprechend eine Adresse im gleichen Subnetz zugewiesen werden.

Das Jupyter Notebook kann dann über `http://192.168.2.99` aufgerufen werden. Ohne weitere Einstellungen (z.B. Netzwerkbrücken) kann das Board in dieser Konfiguration nicht auf das Internet zugreifen.

4 Arbeiten mit dem Board

4.1 Zugriff über einen Browser

Unabhängig von der Adressierung (siehe 3.3.1 und 3.3.2) muss man sich beim Zugriff auf das Jupyter Notebook zunächst einloggen:

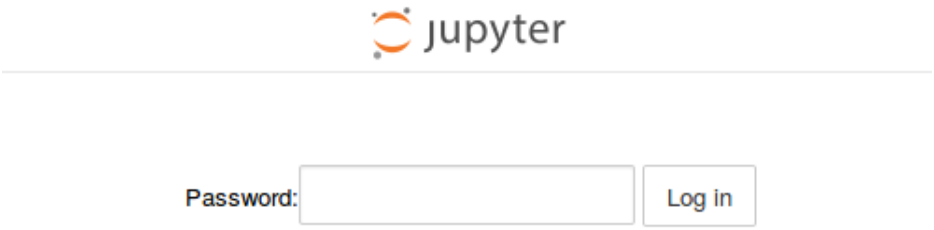

Abbildung 6: Loginmaske von Jupyter Notebook

Das Standardpasswort lautet:

```
xilinx
```

4.2 Zugriff über SSH

Neben Jupyter Notebook kann auch per **SSH** auf das Board zugegriffen werden. Der Standardlogin ist auch hier:

```
xilinx:xilinx
```

Bei einem statisch adressierten Board müsste man also

```
ssh xilinx@192.168.2.99
```

in ein Terminal eingeben. Bei funktionierendem DHCP und DNS entsprechend:

```
ssh xilinx@pynq
```

5 Jupyter Notebook

Nach dem Login erreicht man die Oberfläche von **Jupyter Notebook**:

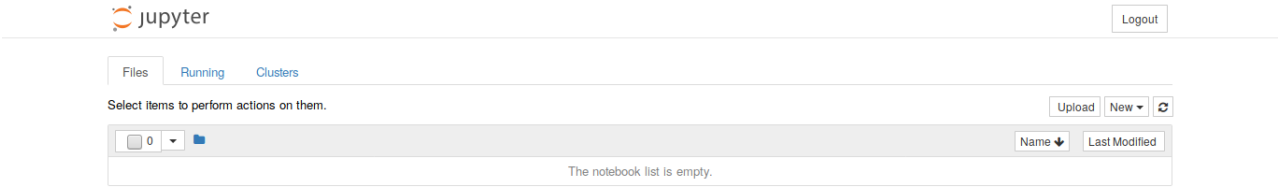

Abbildung 7: Jupyter Oberfläche

Bei der Installation des Betriebssystems wurden bereits einige Notebooks / Codebeispiele vorinstalliert. Über die Reiter **Files, Running, Clusters (1)** kann zwischen der Dateisystem-Ansicht, einer Übersicht über aktive Notebooks sowie einer Übersicht über verteilte Aufträge gewechselt werden. Über den **Upload-Knopf (2)** können Dateien in das aktuelle Verzeichnis hochgeladen werden. Mittels des **New-Menüs (2)** können neue Notebooks, Ordner oder Terminalsitzungen gestartet werden.

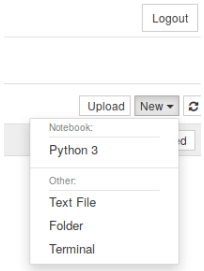

Abbildung 8: Jupyter Menü

Mittels Jupyter können Notebooks für verschiedene Programmiersprachen erstellt werden (die Module für die jeweilige Sprache werden hierbei als Kernel bezeichnet). Bei PYNQ ist ausschließlich der Kernel für Python 3 installiert.

Nach dem erstellen eines neuen Notebooks bzw. dem Öffnen eines exisitierenden Notebooks landet man in der Programmieroberfläche von Jupyter:

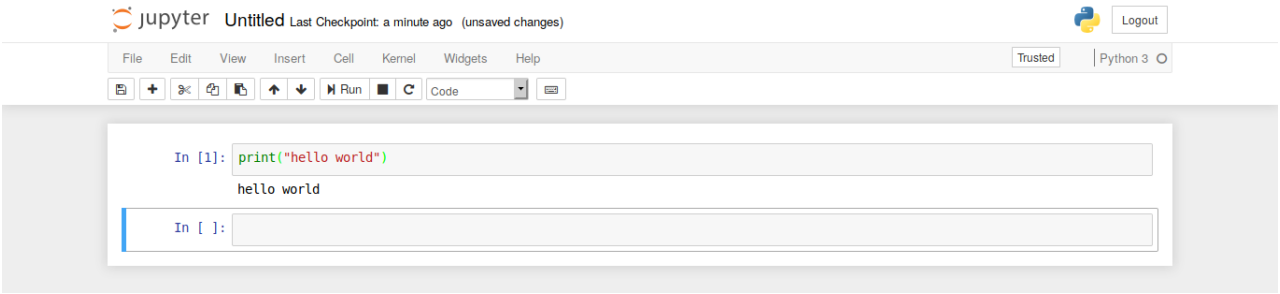

Abbildung 9: Jupyter Notebook Python-Oberfläche

Der Code kann hierbei in Zellen unterteilt werden, welche unanhängig voneinander ausgeführt werden. Um eine Zelle auszuführen muss diese zunächst durch klicken in die Zelle aktiviert werden. Die Ausführung kann dann mittels des **Run-Knopfes (1)** gestartet werden.

Literatur

[1] Digilent. PYNQ Z1 Reference Manual, 04.04.2019.
https://reference.digilentinc.com/_media/reference/programmable-logic/pynq-z1/pynq-rm.pdf.

[2] Pynq. PYNQ Z1 Image, 10.04.2019.
https://pynq.readthedocs.io/en/v2.2.1/getting_started/pynq_image.html.

[3] Tobin Davis, Justin Davis. Win32 Disk Imager 1.0.0, 12.04.2019.
https://sourceforge.net/projects/win32diskimager/files/Archive/Win32DiskImager-1.0.0-binary.zip/download.