Odroid-N2 Documentation

Cody Roberson - Arizona State University

March 23 2023

Contents

1 Initial Setup

1.1 OS Image

Ubuntu Minimal 20.04 was downladed from the following page. ODroid OS Installation Guide

The file was imaged onto a standard 16Gb Micro-SD Card and loaded into the Odroid-N2. The device was connected to a regular mouse, keyboard, ethernet, and moniter.

1.2 Login

The username was

```
root
```

The password was:

```
odroid
```

1.3 Updates

The os offered an update to 22.04 and so the command

```
do-release-upgrade
```

was issued and the system updated to Ubuntu 22.04 Minimal. Old packages were removed. Vim was installed

```
apt-get install vim -y
```

1.4 Networking

A secondary ip address was configured using nmcli enabling a direct pc to odroid network connection.‘

```
nmcli con mod "Wired connection 1" +ipv4.addresses "192.168.99.2"/24
reboot
```

The subnetmask is 255.255.255.0
Users can ssh into the odroid using:

```
ssh root@192.168.99.2
```

provided they have configured their own connection correctly.

1.5 Python

The dependencies of Python3 were installed and Python was cloned from github.

```
sudo apt-get install build-essential gdb lcov pkg-config \
  libbz2-dev libffi-dev libgdbm-dev libgdbm-compat-dev liblzma-dev \
  libncurses5-dev libreadline6-dev libsqlite3-dev libssl-dev \
  lzma lzma-dev tk-dev uuid-dev zlib1g-dev

git clone https://github.com/python/cpython.git --depth=1 --branch=v3.11.2
```

Having setup the base, python was built and installed

```
cd python
mkdir build
cd build
../configure --enable-optimizations --enable-shared
make altinstall
cp libpython3.* /usr/lib/aarch64-linux-gnu/
cd
```

1.6 Python Virtual Environment

In the terminal, a command was issued to create a PVE in the root home directory. And the Python Virtual Environment was activated.

```
python3.11 -m venv py3

source py3/bin/activate
```

Pip was then updated

```
pip install --upgrade pip
```

and the basic necessities were installed

```
pip install matplotlib jupyterlab ipython numpy scipy
```

2 Jupyter Lab

Some minor configuration was setup in

```
.jupyter/jupyter_lab_config.py
```

the password was configured to be

```
odroid
```

Jupyter-notebook was then launched.

```
cd
jupyter-lab --allow-root --ip=<network ip goes here> jnotebooks/
```

A hello world notebook was created to test that everything appeared to be in working order and that was indeed the case. $e^{-i\omega t} * e^{i\pi/2}$ was plotted.

3 Board Control

(Now that the OS was configured, a complete SD Card image was taken.)
We referred to the documentation found here: Android N2 Wiki to enable and install the gpio related hw/sw such as SPI/I2C/UART. However, it seems that these devices are enabled by default and no additional configuration is needed. The board controls will be facilitated via i2c and as such, wiring pi was planned for installation.

3.1 wiring pi

Wiring pi was installed according to documentation.

```
add-apt-repository ppa:hardkernel/ppa
apt update
apt install odroid-wiringpi
source py3/bin/activate
pip install odroid-wiringpi
```

At this point it, the odroid is development ready.

4 Requirements

- The term 'useability' refer to how complex the procedure is to run the system and the relative skill level required.
- The term 'testability' shall refer to the complexity and skill needed to test elements of the system using the software.
- The 'EU' or End User shall be the end entity of which this product is delivered.
- The 'HE' or Hardware Engineer shall be the hardware engineer Eric Weeks who will be designing the daughter boards of the VME backplane.
- The 'SE' or Software Engineer shall be Cody Roberson. Whom shall provide the software and control scheme that the EU shall utilize.

4.1 The EU shall interface with the Odroid

4.1.1 Interaction Trade Study

This is an **open requirement**. Therefore several possibilites exist that will impact implementation. The EU could require that all EU-to-Odroid control take place:

1. Through an Ethernet API
2. Through a <generic> serial bus command API
3. Directly use a UI running on the Odroid
4. Web UI hosted by the Odroid

The considerations are merited as follows:

- Cost, Useability
  - The cheapest solution for the SE would be a dead simple python script that haa functions which are used to control I2C devices. Its the most adaptable as well but is not very useable. This could make testing slow
  - A API over ethernet would be the second cheapest solution. Its a python script that will listen for requests and make the appropriate function calls. The code can still be highly modifiable since its a small step up from the previous option. The EU can implement their own code to run on the odroid or a host pc which makes requests to the api.
  - A UI on the odroid would increase complexity and start to get relatively expensive. It could take the form of a Terminal Interface or a Web Interface or GUI. All of which would need to be implemented. A desktop may need to be installed as well. Useability would be very high however, modifiablility would nontrivially decrease. Testability would be somewhat easier however, debugging would be costlier taking into account reduced modifiability.
- Ease of use for the HE to test his designs with. It should be something simple, robust, and quick. It should also provide some dianostic utility that removes code errors as a factor.

- Ultimately, this requirement is predicated on how the EU intends to utilize the device in their system. Will the user control this as a subsystem of another System or Standalone. Will the EU prefer manual or programmed control?

Conclusion Implementing just the python script would be the cheapest overall and is therefore the best option going forward considering time available.

4.1.2 EU and HE shall utilize a basic python library

Rationale: See Quick Study.

4.2 The odroid shall control the daughter boards within the VME backplane it's connected to

The VME backplane we're interested in using is Vector Electronics and Tech Series 2151 Chassis.

4.3 The user shall have the ability to preserve configuration

5 Design and Implementation

5.1 Operations

1. Enumerate connected boards
2. Request voltages for a given board
3. Request current for a given board
4. Set voltage for a given board
5. Set currnet for a given board
6. set a wiper for a given board
7. get a wiper for a given board
8. save config to given file
9. load config from a given file
10. get all available data from all of the boards

6 Verification and Validation