Bash • Carputer • Photo Frame • Home Server • Android Development

ODROID Magazine

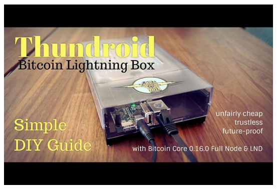
Year Five
Issue #54
Jun 2018

Game Station TURBO

OLD SCHOOL NOSTALGIA GAME CONSOLE DESIGN ON YOUR XU4!

SOLAR POWERED MICROSERVER: INFRASTRUCTURE WHENEVER YOU NEED

THUNDROID: THE PERFECT BITCOIN LIGHTNING NODE



Thundroid: The Perfect Bitcoin Lightning Node

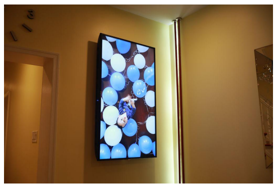
June 1, 2018

Bitcoin is a native digital currency for the internet age. It could be considered just another international currency, but one without a native country so it defies borders, trade policies, and arbitrary inflation. In the 2008 whitepaper (https://bitcoin.org/bitcoin.pdf) by the pseudonymous Satoshi Nakamoto it is described as "…a purely peer-to-peer ▶

Linux Gaming: Nintendo 64 Emulation – Part 1

June 1, 2018

It took a while to get N64 emulation to work on all the ODROID boards under Linux. However, now that it's functioning, it's quite fun and opens up lots of opportunities for classic gaming. Hopefully in the future, we will see more improvement and have even better support for N64 ▶



Digital Photo Frame: 55 inch 4K Digital Photo Frame Display for Around $400

June 1, 2018

There are lots of tutorials on how to make an awesome digital photo frame with a Raspberry Pi.

OS Spotlight: ODROID GameStation Turbo

June 1, 2018

One of the biggest projects that I am working on for the ODROID community is the ODROID GameStation Turbo image, which works as a frontend for both games and media playback. It's intended as an entertainment system that allows you to control your ODROID just by using a game controller ▶

OGST Gaming Console Kit for the ODROID-XU4

June 1, 2018

The OGST Gaming Console Kit for the ODROID-XU4 kit allows you to build your own gaming console with a powerful ODROID-XU4 or ODROID-XU4Q. Its attractive design includes a fancy 2.4" LCD to show programmable game logo animations, and is specifically designed to work with the popular ODROID GameStation Turbo disk ▶

ODROID GameSir G3w USB Controller Joystick

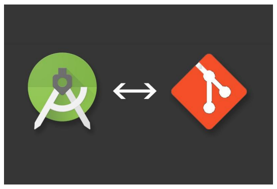
June 1, 2018

The GameSir G3w is a high-quality gamepad that adopts a 32-bit MCU chip, with a computing capability that is up to 48 million operations per second. And it is supported on the official Linux and Android operating systems offered by Hardkernel.

Solar Powered Microserver

June 1, 2018

Blackouts are not just annoying, but cause a series of problems impacting almost all aspects of modern life, so after months in the dark, I took the plunge an built a small solar power system.The ODROID C2 has a 64-bit quad-core ARM CPU, 2 GB of RAM, and support eMMC ▶



Android Development: Using GitHub

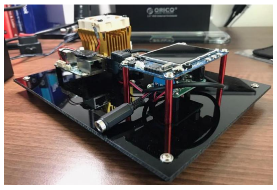
June 1, 2018

Welcome back, appdev initiates! If you're like me, you're more than ready to increase your app development skills. As mentioned previously, the 800-pound gorilla of online open source projects is GitHub. There are several Git-based choices in the marketplace, but for our purposes, we will use GitHub for this column, ▶

Linux Gaming on ODROID: Saturn Games – Part 4

June 1, 2018

We are back again with the ODROID-XU4/XU4 and Sega Saturn games. This time I want to look into games starting with the letter "S" like in Saturn, or as in "shmup". There are so many great games for the Sega Saturn that start with the letter "S" that I enjoy ▶



ODROID-XU4 Home Server

June 1, 2018

Back in December 2017, I rebuilt my Odroid XU4 home media server (https://goo.gl/6tT6rt) because I was having some issues with the previous setup. Unfortunately, that rebuild was not focusing on aesthetics or cable management, just functionality, because I needed the server up and running and did not care how it ▶

Carputer – 7″ Touch Screen Android

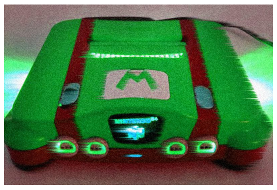
June 1, 2018

This is a 7″ Touch Screen Android Carputer with super accurate USB GPS, Bluetooth 4, 3.5MM Audio in/out, WiFi, and an adjustable magnetic screen. A sketchup file is available for additional customization and resizing as needed at https://www.thingiverse.com/thing:2720349. As an IT Field service tech of 28 years, I built this ▶

Introduction to BASH Basics – Part 2: Useful BASH commands for Single Board Computers

June 1, 2018

Last time, we learned about the 'ls' and 'tree' commands. While looking at things is nice, it's more fun to actually do something with our data. This article contains a list of the common commands for manipulating data. The command and its explanation are kept very brief to avoid writing ▶



Linux Gaming: Nintendo 64 Emulation – Part 2

June 1, 2018

Part 1 of this article introduced the latest version of the Nintendo 64 emulator for Linux and compared its performance on all of the current ODROID boards. This second part presents an overview of some of the more popular Nintendo 64 games, including Mario Kart, Mario Party, Paper Mario, Star ▶️

Thundroid: The Perfect Bitcoin Lightning Node

June 1, 2018 By @stadicus ODROID-XU4, Tutorial

Bitcoin is a native digital currency for the internet age. It could be considered just another international currency, but one without a native country so it defies borders, trade policies, and arbitrary inflation. In the 2008 whitepaper (https://bitcoin.org/bitcoin.pdf) by the pseudonymous Satoshi Nakamoto it is described as "…a purely peer-to-peer version of electronic cash [which] would allow online payments to be sent directly from one party to another without going through a financial institution."

Being peer-to-peer means that Bitcoin does not rely on a middleman such as a bank, and can be transferred as a bearer asset, like physical cash, without asking anyone for permission. It does not need to be stored physically as it is secured by a cryptographic key, so it can be transferred within minutes to anyone anywhere in the world. One key component of this free open-source financial system is the blockchain, a ledger that keeps track of who owns how many bitcoin and that is stored as an identical copy by all users that decide to run a full Bitcoin node. You can learn more at bitcoin.org.

Bitcoin is an economic experiment of epic scope, and its success is by no means certain. In any case, Bitcoin as a new technology is an incredibly interesting endeavor, especially due to its interdisciplinary nature and low barrier to entry. Bitcoin as sound money– being scarce and non-inflationary, challenging money as one of the last true monopolies of nation states– could have a major impact on economic principles and society as a whole.

At the moment, Bitcoin is more a store of value and not really suited for small everyday transactions. Truly decentralized blockchains are a scarce resource and cannot scale to accommodate all global transactions. If you think about it, it cannot be good practice to store every coffee purchase redundantly all over the world for all eternity. That would be like forcing everyone to download everyone else's email as well.

These limitations are a great motivator to build better technology on top of the Bitcoin blockchain to scale exponentially, as opposed to simply making everything bigger for the sake of linear scaling.

This is where the “Lightning Network” comes in. As one of several new blockchain extensions, it promises to accommodate nearly unlimited transactions, with instant confirmation, minimal fees, and increased privacy. It sounds almost too good to be true, but this technology is well researched, committed to the cypherpunk open-source ethos, and leverages the solid underpinnings of Bitcoin. **Learn more.**

To preserve the decentralized nature of this monetary system, it is important that everybody has at least the possibility to run their own trustless Bitcoin node, preferably on cheap hardware like ODROID.

NOTE: Please be aware that while Bitcoin has been battle-tested for almost a decade, the Lightning Network is still in beta and under heavy development. This guide also allows you to set up your Bitcoin node while ignoring the Lightning part. Read the “Financial Best Practices” section before committing real Bitcoin to your node.

Purpose

This guide allows you to be your own bank. The aim is to set up a Bitcoin and Lightning node that:

- is as fully validating Bitcoin Full Node and does not require any trust in a 3rd party
- is reliably running 24/7
- is part of and supports the decentralization of the Lightning network by routing payments
- can be used to send and receive personal payments using the command line interface.

This server is set up without graphical user interface and is operated remotely using the Secure Shell (SSH) command line. It can also function as a personal Bitcoin backend for the desktop Electrum wallet.

Target audience

While this guide strives to give simple and foolproof instructions, the goal is also to do everything ourselves–no shortcuts that involve trust in a 3rd party allowed. This makes this guide quite technical, but I have tried to make it as straightforward as possible for you to gain a basic understanding of the how and why.

A word of caution

All components of the Lightning network are still under development and we are dealing with real money here. This guide follows a conservative approach: first setup and test everything on Bitcoin testnet, then once you are comfortable enough to put real money on the line, switch to Bitcoin mainnet with a few simple changes.

Preparations

After publishing the “Beginner’s Guide to Lightning on a Raspberry Pi,” I started to explore other hardware since the Raspberry Pi has drawbacks, mainly in the area of performance and the the hassle of attaching external storage, which is important when storing the big Bitcoin blockchain.

Hardkernel’s ODROID-HC2 (**http://www.hardkernel.com/main/products/prdt_info.php**)–or the HC1 for a smaller form factor–as a Linux-based mini PC is a perfect fit. Compared to a Raspberry Pi, it has the following advantages:

- Price comparable to Raspberry Pi
- More powerful (8 core CPU, 2 GB RAM, Gigabit Ethernet)
- Internal hard disk housing, direct connection using SATA3
- Only one power adapter for everything

Not available are features like an HDMI output, built-in Wifi, or GPIO pins, but these are not relevant to this project. The performance is way better, so it seems more future-proof as Bitcoin and Lightning are certain to evolve.

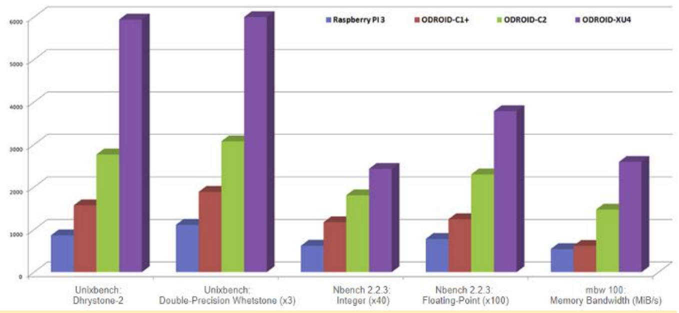


Figure 1 – Performance of ODROID-HC2 is identical to XU4 (which is more of a media pc)

Once I decided on this setup, I ordered the ODROID-HC2 and, after setting it up and running it for two months now, I think this is as good as it gets for a low-cost Bitcoin/Lightning node. As this project needs a cheesy name as well, I'll call my node Thundroid.

I ordered the following items directly from Hardkernel in Singapore. There are resellers available worldwide but unfortunately not for Switzerland.

- ODROID-HC2 – $54 (http://www.hardkernel.com/main/products/prdt_info.php?g_code=G151505170472)
- Power adapter and cord – $7 (http://www.hardkernel.com/main/products/prdt_info.php?g_code=G151578376044)
- ODROID-HC2 case (optional) – $5 (http://www.hardkernel.com/main/products/prdt_info.php?g_code=G151)

You also need the following:

- Micro SD card: 16 GB, including an adapter to your regular computer
- Internal hard disk: 500 GB or more, SATA, 3.5″ HDD, or 2.5″ HDD/SSD
- Network RJ45 cable

Assembly is easy: just insert the hard disk and affix it with the the screws included with your ODROID. If you ordered the plastic case, slide it over the metal body.

Installing the operating system

We use Ubuntu 16.04.03 LTE (Minimal, Bare OS) that is provided by Hardkernel. Download the image from the ODROID-XU4 section on wiki.odroid.com

Exact file used: https://odroid.in/ubuntu_16.04lts/ubuntu-16.04.3-4.14-minimal-odroid-xu4-20171213.img.xz

Download the image, flash it on your MicroSD card, put it into your Thundroid, connect it to your network via cable, and connect the power adapter. The initial boot can take several minutes.

Configure your network router to assign a static IP address to your Thundroid.

Working on your Thundroid

Write down your passwords

You will need several passwords and I find it easiest to write them all down in the beginning, instead of bumping into them throughout the guide. They should be unique and very secure, at least 12 characters in length. Do not use uncommon special characters, blanks, or quotes (‘ or “).

- User password
- Bitcoin RPC password
- Lightning API password
- Lightning seed passphrase

Store a copy of your passwords somewhere safe (preferably in a password manager like KeePass or LastPass) and keep your original notes out of sight once your system is up and running.

The command line

Everything is configured on the Linux command prompt. Throughout this guide I use the following notation:

#: this is a comment, just for information $: This is a single-line command to enter (without the $) and confirm with the enter key No prefix: This is either an output of the command above or something you can copy/paste into a file

- Auto-complete commands: When you enter commands, you can use the ‘Tab’ key for auto-completion, eg. for commands, directories or filenames.
- Command history: By pressing up and down on your keyboard, you can recall your previously entered commands.
- Use admin privileges: Our users has no admin privileges. If a command needs to edit the system

configuration, we need to use the ‘sudo’ (“superuser do”) command as prefix. Instead of editing a system file with ‘nano /etc/fstab’, we use ‘sudo nano /etc/fstab’.

- Using the Nano text editor: We use the Nano editor to create new text files or edit existing ones. It’s not complicated, but to save and exit is not intuitSave: hit ‘Ctrl-O’ (for Output), confirm the filename, and hit the ‘Enter’ Exit: hit ‘Ctrl-X’

* **Copy / Paste**: If you are using Windows and the PuTTY SSH client, you can copy text from the shell by selecting it with your mouse (no need to click anything), and paste stuff at the cursor position with a right-click anywhere in the SSH window.

Connecting to Thundroid

It’s time to connect via SSH and get to work. For that, a Secure Shell (SSH) client is needed. Install, start and connect:

- Windows: I recommend to use the SSH client [KiTTY](http://kitty.9bis.com). You can copy text from the shell by selecting it with your mouse (no need to click anything), and paste stuff with a right-click.
- Mac OS: Built-in SSH client (http://osxdaily.com/2017/04/28/howto-ssh-client-mac/)
- Linux: Just use the native command, eg. ssh root@192.168.0.20

Use the following SSH connection settings:

- Host name: the static address you set in the router, eg. 192.168.0.20
- Port: 22
- Username: root
- Password: odroid

Basic configuration

You are now on the command line of your own Bitcoin node. First, we take care of the basic configuration. Enter the following commands:

```
# change root password to [password A]
$ passwd

# update the operating system
$ apt update
$ apt upgrade
$ apt dist-upgrade
$ apt install linux-image-xu3
# answer [y], then [no] (do not abort)

# install some additional software
$ apt install htop git curl bash-completion jq

# set time zone & localization
$ dpkg-reconfigure tzdata
$ dpkg-reconfigure locales
```

When using the Nano text editor, you can use the same keyboard shortcuts to save (Ctrl-O, confirm or change filename, and press Enter) and exit (Ctrl-X).

```
# change hostname (replace "odroid" with
"thundroid" :) in both files
$ nano /etc/hostname
$ nano /etc/hosts

# create user "admin", set [password A] and
make it a superuser
$ adduser admin
$ adduser admin sudo

# create user "bitcoin" and set password
[password A]
$ sudo adduser bitcoin
```

Mounting the hard disk

The external hard disk is attached to the file system and can be accessed as a regular folder. This is called “mounting.” As a server installation, the Linux native file system Ext4 is the best choice for the external hard disk.

NOTE: All data on this hard disk will be erased with the following steps!

```
# get NAME for hard disk
$ lsblk -o UUID,NAME,FSTYPE,SIZE,LABEL,MODEL

# format hard disk (use [NAME] from above, e.g
/dev/sda1)
$ mkfs.ext4 /dev/[NAME]

# get UUID for hard disk, copy into notepad
$ lsblk -o UUID,NAME,FSTYPE,SIZE,LABEL,MODEL

# edit fstab and enter new line (replace UUID)
at the end, save & exit
$ nano /etc/fstab
```

```
UUID=123456 /mnt/hdd ext4 noexec,defaults 0 0

# create mount point, mount, check and set
owner
$ mkdir /mnt/hdd
$ mount -a
$ df /mnt/hdd
Filesystem 1K-blocks Used Available Use%
Mounted on
/dev/sda1 961300808 600388836 312057600 66%
/mnt/hdd

$ chown -R bitcoin:bitcoin /mnt/hdd/
```

Moving the Swap File

The use of a swap file can degrade the SD card very quickly. Therefore, we will move it to the external hard disk.

```
# install necessary software package
$ apt install dphys-swapfile

# change configuration file to use swapfile on
external hard disk
$ nano /etc/dphys-swapfile
CONF_SWAPFILE=/mnt/hdd/swapfile

# enable new swap configuration
$ sudo dphys-swapfile setup
$ sudo dphys-swapfile swapon

# reboot, login as "admin" and delete old
swapfile
$ restart shutdown -r now
$ sudo rm /var/swap
```

Hardening your Thundroid

Your Thundroid will be visible from the internet and therefore needs to be secured against attacks. A firewall controls what traffic is permitted and closes possible security holes. Login as “admin” (we will not use “root” again).

UFW: Uncomplicated Firewall

The firewall denies all connection attempts from other peers by default and allows only specific ports to be used.

WARNING: The line ‘ufw allow from 192.168.0.0/24 …’ below assumes that the IP address of your Thundroid is something like ‘192.168.0.???”, the variable being any number from 0 to 255. If your IP address is ‘12.34.56.78’, you must adapt this line to ‘ufw allow from 12.34.56.0/24 …’. Otherwise you will lock yourself out for good.

```
# change session to "root"
$ sudo su
$ apt install ufw
$ ufw default deny incoming
$ ufw default allow outgoing

# make sure to use the correct subnet mask
(see warning above)
$ ufw allow from 192.168.0.0/24 to any port 22
comment 'allow SSH from local LAN'

$ ufw allow 9735 comment 'allow Lightning'
$ ufw deny 8333 comment 'deny Bitcoin mainnet'
$ ufw allow 18333 comment 'allow Bitcoin
testnet'
$ ufw enable
$ systemctl enable ufw
$ ufw status

# exit "root" session back to "admin"
$ exit
```

Fail2ban

Fail2ban monitors SSH login attempts and bans a remote peer for 10 minutes after five unsuccessful tries. This makes a brute-force attack unfeasible, as it would simply take too long.

```
$ sudo apt install fail2ban
```

SSH Keys

One of the best options to secure the SSH login is to completely disable the password login and require a SSH key certificate. Only someone with physical possession of the private key can login.

Set up SSH keys for the “admin” user by following this article: Configure “No Password SSH Keys Authentication” with PuTTY on Linux Servers

You should now three generated files. Keep them safe, as we will now disable the password login.

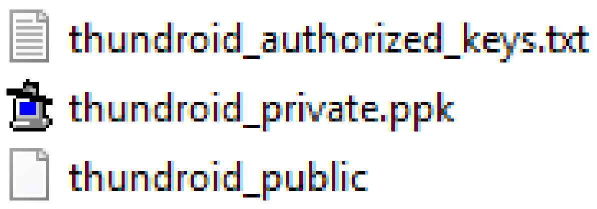

thundroid_authorized_keys.txt
thundroid_private.ppk
thundroid_public

Figure 2 – SSH Keys filelist

(Figure 2 – SSH Keys filelist)

- Logout (‘exit’) and make sure that you can login as “admin” with your SSH key
- Edit SSH config file

```
$ sudo nano /etc/ssh/sshd_config
```

- Change settings “ChallengeResponseAuthentication” and “PasswordAuthentication” to “no” (uncomment the line by removing # if necessary), save, and exit.

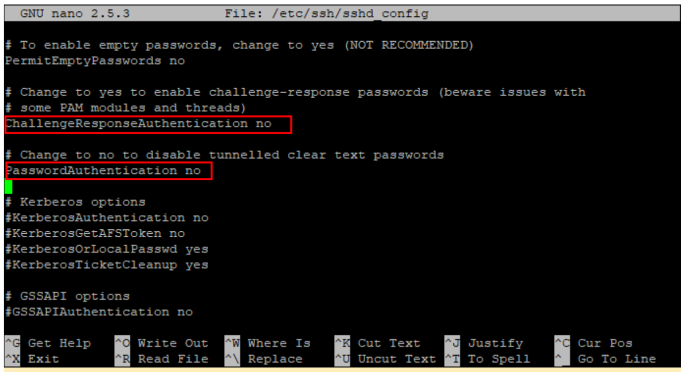

```
  GNU nano 2.5.3             File: /etc/ssh/sshd_config

# To enable empty passwords, change to yes (NOT RECOMMENDED)
PermitEmptyPasswords no

# Change to yes to enable challenge-response passwords (beware issues with
# some PAM modules and threads)
ChallengeResponseAuthentication no

# Change to no to disable tunnelled clear text passwords
PasswordAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosGetAFSToken no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

# GSSAPI options
#GSSAPIAuthentication no

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify     ^C Cur Pos
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell    ^_ Go To Line
```

Figure 3 – SSH config

```
# copy the ssh key to user "root", just in
case
$ sudo mkdir /root/.ssh
$ sudo cp /home/admin/.ssh/authorized_keys
/root/.ssh/
$ sudo chown -R root:root /root/.ssh/
$ sudo chmod -R 700 /root/.ssh/
$ sudo systemctl restart ssh

# exit and login again with your private key
$ exit
```

You can now only login with “admin” or “root” and your SSH key. As you cannot connect a screen to the ODROID, SSH is your only option.

REMINDER: Backup your SSH key! There is no fallback login! In a worst-case scenario, you will need to flash the MicroSD card and set up the system again; all the important stuff is still on the hard drive.

Increase your open files limit

In case your Thundroid is swamped with internet requests–honest or malicious due to a DDoS attack–you will quickly encounter the ‘can’t accept connection: too many open files’ error. This is due to a limit on open files (representing individual TCP connections) that is set too low.

Edit the following three files, add the additional line(s) right before the end comment, save and exit.

```
$ sudo nano /etc/security/limits.conf
* soft nofile 128000
* hard nofile 128000
root soft nofile 128000
root hard nofile 128000
```

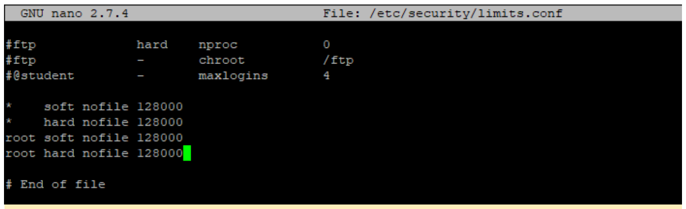

```
  GNU nano 2.7.4                              File: /etc/security/limits.conf

#ftp             hard    nproc           0
#ftp             -       chroot          /ftp
#@student        -       maxlogins       4

*    soft nofile 128000
*    hard nofile 128000
root soft nofile 128000
root hard nofile 128000

# End of file
```

Figure 4 – Edit pam.d/limits.conf.png

```
$ sudo nano /etc/pam.d/common-session
session required pam_limits.so
```

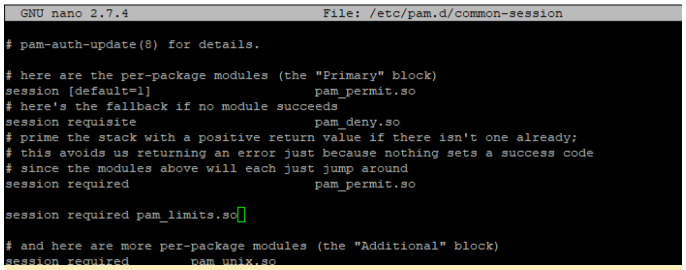

```
  GNU nano 2.7.4                              File: /etc/pam.d/common-session

# pam-auth-update(8) for details.

# here are the per-package modules (the "Primary" block)
session [default=1]                     pam_permit.so
# here's the fallback if no module succeeds
session requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
session required                        pam_permit.so

session required pam_limits.so

# and here are more per-package modules (the "Additional" block)
session required        pam_unix.so
```

Figure 5

```
$ sudo nano /etc/pam.d/common-session-
noninteractive
session required pam_limits.so
```

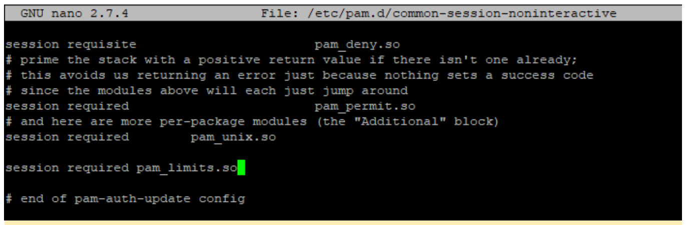

```
  GNU nano 2.7.4                  File: /etc/pam.d/common-session-noninteractive

session requisite                       pam_deny.so
# prime the stack with a positive return value if there isn't one already;
# this avoids us returning an error just because nothing sets a success code
# since the modules above will each just jump around
session required                        pam_permit.so
# and here are more per-package modules (the "Additional" block)
session required        pam_unix.so

session required pam_limits.so

# end of pam-auth-update config
```

Figure 6 – Edit pam.d/common-session-noninteractive

Bitcoin

The foundation of the Lightning node is a fully trustless Bitcoin node (https://bitcoin.org/en/bitcoin-core/). It keeps a complete copy of the blockchain and validates all transactions and blocks. By doing all this work ourselves, nobody else needs to be trusted.

In the beginning, we will use the Bitcoin testnet to familiarize ourselves with its operations. This sync is handled directly by the Thundroid and should not take longer than a few hours. Just let it sync overnight.

Installation

We will download the software directly from bitcoin.org, verify its signature to make sure that we use an official release, and install it.

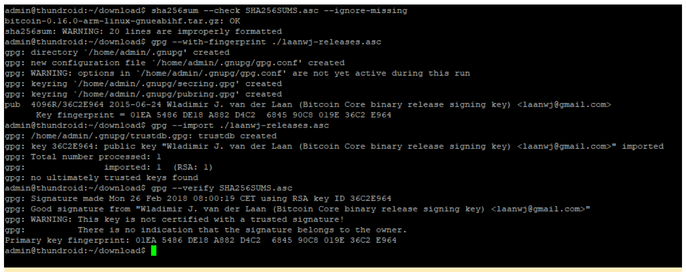
Login as "admin" and create a download folder:

```
$ mkdir /home/admin/download
$ cd /home/admin/download
```

We download the latest Bitcoin Core binaries and compare the file with the signed checksum. This is a precaution to make sure that this is an official release and not a malicious version trying to steal our money.

Get the latest download links at bitcoin.org/en/download. They change with each update. Run the following commands with adjusted filenames and check the output where indicated.

```
# download Bitcoin Core binary
$ wget https://bitcoin.org/bin/bitcoin-core-
0.16.0/bitcoin-0.16.0-arm-linux-
gnueabihf.tar.gz
$ wget https://bitcoin.org/bin/bitcoin-core-
0.16.0/SHA256SUMS.asc
$ wget https://bitcoin.org/laanwj-releases.asc

# check that the reference checksum matches
the real checksum
# (ignore the "lines are improperly formatted"
warning)
$ sha256sum --check SHA256SUMS.asc --ignore-
missing
> bitcoin-0.16.0-arm-linux-gnueabihf.tar.gz:
OK

# manually check the fingerprint of the public
key
$ gpg --with-fingerprint ./laanwj-releases.asc
> 01EA 5486 DE18 A882 D4C2 6845 90C8 019E 36C2
E964

# import the public key of Wladimir van der
Laan, verify the signed checksum file
# and check the fingerprint again in case of
malicious keys
$ gpg --import ./laanwj-releases.asc
$ gpg --verify SHA256SUMS.asc
> gpg: Good signature from Wladimir ...
> Primary key fingerprint: 01EA 5486 DE18 A882
D4C2 6845 90C8 019E 36C2 E964
```



```
admin@thundroid:~/download$ sha256sum --check SHA256SUMS.asc --ignore-missing
bitcoin-0.16.0-arm-linux-gnueabihf.tar.gz: OK
sha256sum: WARNING: 20 lines are improperly formatted
admin@thundroid:~/download$ gpg --with-fingerprint ./laanwj-releases.asc
gpg: directory `/home/admin/.gnupg' created
gpg: new configuration file `/home/admin/.gnupg/gpg.conf' created
gpg: WARNING: options in `/home/admin/.gnupg/gpg.conf' are not yet active during this run
gpg: keyring `/home/admin/.gnupg/secring.gpg' created
gpg: keyring `/home/admin/.gnupg/pubring.gpg' created
pub  4096R/36C2E964 2015-06-24 Wladimir J. van der Laan (Bitcoin Core binary release signing key) <laanwj@gmail.com>
      Key fingerprint = 01EA 5486 DE18 A882 D4C2  6845 90C8 019E 36C2 E964
admin@thundroid:~/download$ gpg --import ./laanwj-releases.asc
gpg: /home/admin/.gnupg/trustdb.gpg: trustdb created
gpg: key 36C2E964: public key "Wladimir J. van der Laan (Bitcoin Core binary release signing key) <laanwj@gmail.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: no ultimately trusted keys found
admin@thundroid:~/download$ gpg --verify SHA256SUMS.asc
gpg: Signature made Mon 26 Feb 2018 08:00:19 CET using RSA key ID 36C2E964
gpg: Good signature from "Wladimir J. van der Laan (Bitcoin Core binary release signing key) <laanwj@gmail.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 01EA 5486 DE18 A882 D4C2  6845 90C8 019E 36C2 E964
admin@thundroid:~/download$
```

Figure 7 – Commands to check bitcoind signature

Extract the Bitcoin Core binaries, install them, and check the version.

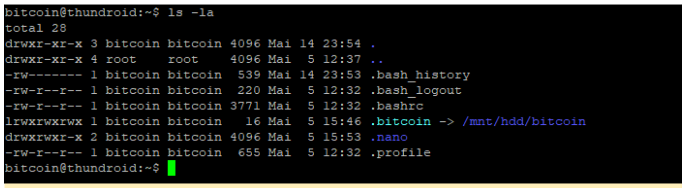
```
$ tar -xvf bitcoin-0.16.0-arm-linux-
gnueabihf.tar.gz
$ sudo install -m 0755 -o root -g root -t
/usr/local/bin bitcoin-0.16.0/bin/*
$ bitcoind --version
> Bitcoin Core Daemon version v0.16.0
```

Prepare Bitcoin Core directory

We use the Bitcoin daemon, called "bitcoind", that runs in the background without user interface and stores all data in a the directory '/home/bitcoin/.bitcoin'. Instead of creating a real directory, we create a link that points to a directory on the external hard disk.

```
# change to user "bitcoin"
$ sudo su bitcoin

# add symbolic link that points to the
external hard drive
$ mkdir /mnt/hdd/bitcoin
$ ln -s /mnt/hdd/bitcoin
/home/bitcoin/.bitcoin

# Navigate to home directory and check the
symbolic link (the target must not be red).
$ cd
$ ls -la
```

The content of this directory will actually be on the external hard disk.



```
bitcoin@thundroid:~$ ls -la
total 28
drwxr-xr-x 3 bitcoin bitcoin 4096 Mai 14 23:54 .
drwxr-xr-x 4 root    root    4096 Mai  5 12:37 ..
-rw------- 1 bitcoin bitcoin  539 Mai 14 23:53 .bash_history
-rw-r--r-- 1 bitcoin bitcoin  220 Mai  5 12:32 .bash_logout
-rw-r--r-- 1 bitcoin bitcoin 3771 Mai  5 12:32 .bashrc
lrwxrwxrwx 1 bitcoin bitcoin   16 Mai  5 15:46 .bitcoin -> /mnt/hdd/bitcoin
drwxrwxr-x 2 bitcoin bitcoin 4096 Mai  5 15:53 .nano
-rw-r--r-- 1 bitcoin bitcoin  655 Mai  5 12:32 .profile
bitcoin@thundroid:~$
```

Figure 8 – Verify .bitcoin symlink

Configuration

The configuration file for bitcoind needs to be created. Open it with Nano and paste the configuration below. Save and exit.

```
$ nano /home/bitcoin/.bitcoin/bitcoin.conf

# Thundroid: bitcoind configuration
# /home/bitcoin/.bitcoin/bitcoin.conf

# remove the following line to enable Bitcoin
mainnet
testnet=1

# Bitcoind options
server=1
daemon=1
txindex=1
disablewallet=1

# Connection settings
rpcuser=bitcoin
rpcpassword=PASSWORD_[B]
zmqpubrawblock=tcp://127.0.0.1:29000
zmqpubrawtx=tcp://127.0.0.1:29000

# Optimizations for limited hardware
dbcache=100
maxorphantx=10
maxmempool=50
maxconnections=40
maxuploadtarget=5000
```

NOTE: Change rpcpassword to your secure ‘password [B]’, otherwise your funds might get stolen.

Autostart bitcoind

The system needs to run the bitcoin daemon automatically in the background, even when nobody is logged in. We use “systemd“, a daemon that controls the startup process using configuration files.

Exit the “bitcoin” user session back to user “admin”

```
$ exit
```

Create the configuration file in the Nano text editor and copy the following paragraph. Save and exit.

```
$ sudo nano
/etc/systemd/system/bitcoind.service

# Thundroid: systemd unit for bitcoind
# /etc/systemd/system/bitcoind.service

[Unit]
Description=Bitcoin daemon
After=network.target

[Service]
ExecStart=/usr/local/bin/bitcoind -daemon -
conf=/home/bitcoin/.bitcoin/bitcoin.conf -
pid=/run/bitcoind/bitcoind.pid
# Creates /run/bitcoind owned by bitcoin
RuntimeDirectory=bitcoind
User=bitcoin
Group=bitcoin
Type=forking
PIDFile=/run/bitcoind/bitcoind.pid
Restart=on-failure

# Hardening measures
####################

# Provide a private /tmp and /var/tmp.
PrivateTmp=true

# Mount /usr, /boot/ and /etc read-only for
the process.
ProtectSystem=full

# Disallow the process and all of its children
to gain
# new privileges through execve().
NoNewPrivileges=true

# Use a new /dev namespace only populated with
API pseudo devices
# such as /dev/null, /dev/zero and
/dev/random.
PrivateDevices=true

# Deny the creation of writable and executable
memory mappings.
```

```
MemoryDenyWriteExecute=true

[Install]
WantedBy=multi-user.target
```

Enable the configuration file:

```
$ sudo systemctl enable bitcoind.service
```

Copy ‘bitcoin.conf’ to user “admin” home directory for RPC credentials:

```
$ mkdir /home/admin/.bitcoin
$ sudo cp /home/bitcoin/.bitcoin/bitcoin.conf
/home/admin/.bitcoin/
```

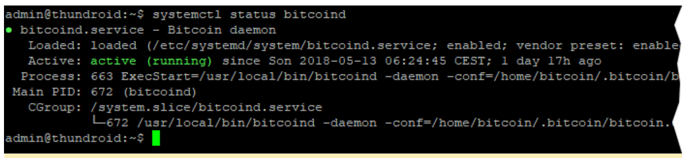
Restart the Thundroid

```
$ sudo shutdown -r now
```

Verification of bitcoind operations

After rebooting, the bitcoind should begin to sync and validate the Bitcoin blockchain. Wait a bit, reconnect via SSH and login with the user “admin”. Check the status of the Bitcoin daemon that was started by systemd (exit with ‘Ctrl-C’).

```
$ systemctl status bitcoind
```



```
admin@thundroid:~$ systemctl status bitcoind
● bitcoind.service - Bitcoin daemon
   Loaded: loaded (/etc/systemd/system/bitcoind.service; enabled; vendor preset: enable
   Active: active (running) since Son 2018-05-13 06:24:45 CEST; 1 day 17h ago
  Process: 663 ExecStart=/usr/local/bin/bitcoind -daemon -conf=/home/bitcoin/.bitcoin/b
 Main PID: 672 (bitcoind)
   CGroup: /system.slice/bitcoind.service
           └─672 /usr/local/bin/bitcoind -daemon -conf=/home/bitcoin/.bitcoin/bitcoin.
admin@thundroid:~$
```

Figure 9 – Bitcoind status

See bitcoind in action by monitoring its log file (exit with ‘Ctrl-C’):

```
$ sudo tail -f
/home/bitcoin/.bitcoin/testnet3/debug.log
```

Use the Bitcoin Core client ‘bitcoin-cli’ to get information about the current blockchain:

```
$ bitcoin-cli getblockchaininfo
```

NOTE: When “bitcoind” is still starting, you may get an error message like “verifying blocks.” That’s normal, just give it a few minutes.

Among other information, the “verificationprogress” is shown. Once this value reaches almost 1 (0.999…), the blockchain is up-to-date and fully validated.

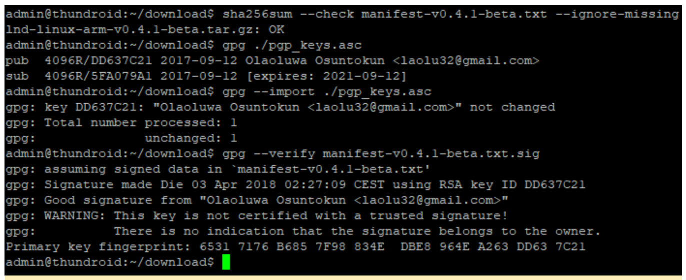
Lightning Network

We will download and install the LND (Lightning Network Daemon) by Lightning Labs **http://lightning.engineering/**. Check out their Github repository (**https://github.com/lightningnetwork/lnd/blob/master/README.md**) for a wealth of information about their open-source project and Lightning in general.

Install LND

Now to the good stuff: Download, verify, and install the LND binaries.

```
$ cd /home/admin/download
$ wget
https://github.com/lightningnetwork/lnd/releas
es/download/v0.4.1-beta/lnd-linux-arm-v0.4.1-
beta.tar.gz
$ wget
https://github.com/lightningnetwork/lnd/releas
es/download/v0.4.1-beta/manifest-v0.4.1-
beta.txt
$ wget
https://github.com/lightningnetwork/lnd/releas
es/download/v0.4.1-beta/manifest-v0.4.1-
beta.txt.sig
$ wget
https://keybase.io/roasbeef/pgp_keys.asc

$ sha256sum --check manifest-v0.4.1-beta.txt -
-ignore-missing
> lnd-linux-arm-v0.4-beta.tar.gz: OK

$ gpg ./pgp_keys.asc
> pub 4096R/DD637C21 2017-09-12 Olaoluwa
Osuntokun <laolu32@gmail.com>
> sub 4096R/5FA079A1 2017-09-12 [expires:
2021-09-12]
> 65317176B6857F98834EDBE8964EA263DD637C21

$ gpg --import ./pgp_keys.asc
$ gpg --verify manifest-v0.4.1-beta.txt.sig
> gpg: Good signature from "Olaoluwa Osuntokun
<laolu32@gmail.com>" [unknown]
> Primary key fingerprint: 6531 7176 B685 7F98
834E DBE8 964E A263 DD63 7C21
```



```
admin@thundroid:~/download$ sha256sum --check manifest-v0.4.1-beta.txt --ignore-missing
lnd-linux-arm-v0.4.1-beta.tar.gz: OK
admin@thundroid:~/download$ gpg ./pgp_keys.asc
pub  4096R/DD637C21 2017-09-12 Olaoluwa Osuntokun <laolu32@gmail.com>
sub  4096R/5FA079A1 2017-09-12 [expires: 2021-09-12]
admin@thundroid:~/download$ gpg --import ./pgp_keys.asc
gpg: key DD637C21: "Olaoluwa Osuntokun <laolu32@gmail.com>" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
admin@thundroid:~/download$ gpg --verify manifest-v0.4.1-beta.txt.sig
gpg: assuming signed data in `manifest-v0.4.1-beta.txt'
gpg: Signature made Die 03 Apr 2018 02:27:09 CEST using RSA key ID DD637C21
gpg: Good signature from "Olaoluwa Osuntokun <laolu32@gmail.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: 6531 7176 B685 7F98 834E  DBE8 964E A263 DD63 7C21
admin@thundroid:~/download$
```

Figure 10 – Checksum LND

```
$ tar -xzf lnd-linux-arm-v0.4.1-beta.tar.gz
$ ls -la
$ sudo install -m 0755 -o root -g root -t
/usr/local/bin lnd-linux-arm-v0.4.1-beta/*
$ lnd --version
> lnd version 0.4.1-beta commit=
```

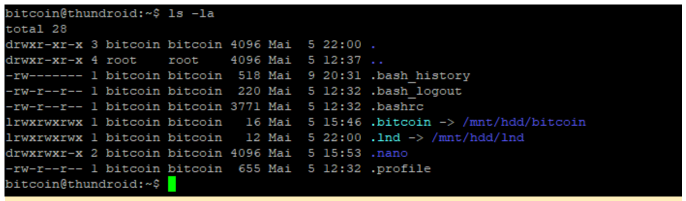
LND configuration Now that LND is installed, we need to configure it to work with Bitcoin Core and run automatically on startup.

Open a “bitcoin” user session:

```
$ sudo su bitcoin
```

Create the LND working directory and the corresponding symbolic link:

```
$ mkdir /mnt/hdd/lnd
$ ln -s /mnt/hdd/lnd /home/bitcoin/.lnd
$ cd
$ ls -la
```



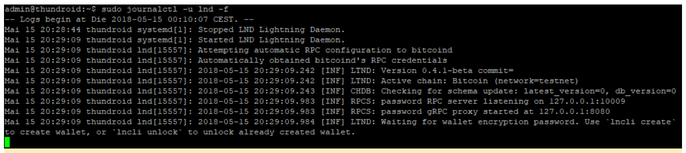
```
bitcoin@thundroid:~$ ls -la
total 28
drwxr-xr-x 3 bitcoin bitcoin 4096 Mai  5 22:00 .
drwxr-xr-x 4 root    root    4096 Mai  5 12:37 ..
-rw------- 1 bitcoin bitcoin  518 Mai  9 20:31 .bash_history
-rw-r--r-- 1 bitcoin bitcoin  220 Mai  5 12:32 .bash_logout
-rw-r--r-- 1 bitcoin bitcoin 3771 Mai  5 12:32 .bashrc
lrwxrwxrwx 1 bitcoin bitcoin   16 Mai  5 15:46 .bitcoin -> /mnt/hdd/bitcoin
lrwxrwxrwx 1 bitcoin bitcoin   12 Mai  5 22:00 .lnd -> /mnt/hdd/lnd
drwxrwxr-x 2 bitcoin bitcoin 4096 Mai  5 15:53 .nano
-rw-r--r-- 1 bitcoin bitcoin  655 Mai  5 12:32 .profile
bitcoin@thundroid:~$
```

Figure 11 – Check symlink LND

Create the LND configuration file and paste the following content (adjust to your alias). Save and exit.

```
'$ nano /home/bitcoin/.lnd/lnd.conf'

# Thundroid: lnd configuration
# /home/bitcoin/.lnd/lnd.conf

[Application Options]
debuglevel=info
debughtlc=true
maxpendingchannels=5
alias=YOUR_NAME [LND]
color=#68F442

[Bitcoin]
bitcoin.active=1

# enable either testnet or mainnet
bitcoin.testnet=1
#bitcoin.mainnet=1

bitcoin.node=bitcoind

[autopilot]
autopilot.active=1
autopilot.maxchannels=5
autopilot.allocation=0.6
```

Additional information



```
admin@thundroid:~$ sudo journalctl -u lnd -f
-- Logs begin at Die 2018-05-15 00:10:07 CEST. --
Mai 15 20:28:44 thundroid systemd[1]: Stopped LND Lightning Daemon.
Mai 15 20:29:09 thundroid systemd[1]: Started LND Lightning Daemon.
Mai 15 20:29:09 thundroid lnd[15557]: Attempting automatic RPC configuration to bitcoind
Mai 15 20:29:09 thundroid lnd[15557]: Automatically obtained bitcoind's RPC credentials
Mai 15 20:29:09 thundroid lnd[15557]: 2018-05-15 20:29:09.242 [INF] LTND: Version 0.4.1-beta commit=
Mai 15 20:29:09 thundroid lnd[15557]: 2018-05-15 20:29:09.242 [INF] LTND: Active chain: Bitcoin (network=testnet)
Mai 15 20:29:09 thundroid lnd[15557]: 2018-05-15 20:29:09.243 [INF] CHDB: Checking for schema update: latest_version=0, db_version=0
Mai 15 20:29:09 thundroid lnd[15557]: 2018-05-15 20:29:09.983 [INF] RPCS: password RPC server listening on 127.0.0.1:10009
Mai 15 20:29:09 thundroid lnd[15557]: 2018-05-15 20:29:09.983 [INF] RPCS: password gRPC proxy started at 127.0.0.1:8080
Mai 15 20:29:09 thundroid lnd[15557]: 2018-05-15 20:29:09.984 [INF] LTND: Waiting for wallet encryption password. Use `lncli create`
to create wallet, or `lncli unlock` to unlock already created wallet.
```

Figure 12 – sample-lnd.conf

In the LND project repository:

Exit the “bitcoin” user session back to “admin”

```
$ exit
```

Create LND systemd unit and with the following content. Save and exit.

```
$ sudo nano /etc/systemd/system/lnd.service

# Thundroid: systemd unit for lnd
# /etc/systemd/system/lnd.service

[Unit]
Description=LND Lightning Daemon
Wants=bitcoind.service
After=bitcoind.service

[Service]
ExecStart=/usr/local/bin/lnd
PIDFile=/home/bitcoin/.lnd/lnd.pid
User=bitcoin
Group=bitcoin
LimitNOFILE=128000
Type=simple
KillMode=process
TimeoutSec=180
Restart=always
RestartSec=60
```

```
[Install]
WantedBy=multi-user.target
```

Enable and start LND

```
$ sudo systemctl enable lnd
$ sudo systemctl start lnd
$ systemctl status lnd
```

Monitor the LND logfile in realtime (exit with ‘Ctrl-C’)

```
$ sudo journalctl -f -u lnd
```

LND wallet setup

Once LND is started, the process waits for us to create the integrated Bitcoin wallet. It does not use the bitcoind wallet.

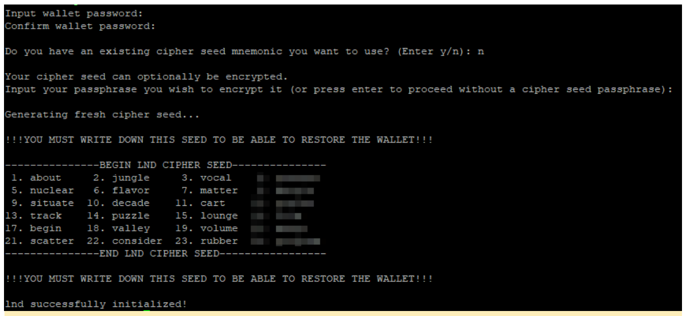
Start a “bitcoin” user session

```
$ sudo su bitcoin
```

Create the LND wallet

```
$ lncli create
```

If you want to create a new wallet, enter your ‘password [C]’ as wallet password, select ‘n’ regarding an existing seed and enter the optional ‘password [D]’ as seed passphrase. A new cipher seed consisting of 24 words is created.



```
Input wallet password:
Confirm wallet password:

Do you have an existing cipher seed mnemonic you want to use? (Enter y/n): n

Your cipher seed can optionally be encrypted.
Input your passphrase you wish to encrypt it (or press enter to proceed without a cipher seed passphrase):

Generating fresh cipher seed...

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

---------------BEGIN LND CIPHER SEED---------------
 1. about     2. jungle     3. vocal    [redacted]
 5. nuclear   6. flavor     7. matter
 9. situate  10. decade    11. cart
13. track    14. puzzle    15. lounge
17. begin    18. valley    19. volume
21. scatter  22. consider  23. rubber
---------------END LND CIPHER SEED-----------------

!!!YOU MUST WRITE DOWN THIS SEED TO BE ABLE TO RESTORE THE WALLET!!!

lnd successfully initialized!
```

Figure 13 – LND new cipher seed

These 24 words, combined with your passphrase (optional ‘password [D]’) is all that you need to restore your Bitcoin wallet and all Lighting channels. The current state of your channels, however, cannot be recreated from this seed, as this is still under development for LND.

NOTE: This information must be kept secret at all times. Write these 24 words down manually on a piece of paper and store it in a safe place. This piece of paper is all an attacker needs to completely empty your wallet! Do not store it on a computer. Do not take a picture with your mobile phone. This information should never be stored anywhere in digital form.

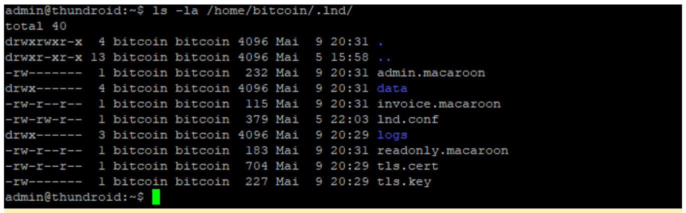
Exit “bitcoin” user session

```
$ exit
```

Assign LND permissions to “admin”

Check if permission files ‘admin.macaroon’ and ‘readonly.macaroon’ have been created. If not, see open LND issue #890 (https://github.com/lightningnetwork/lnd/issues/890).

```
$ ls -la /home/bitcoin/.lnd/
```



```
admin@thundroid:~$ ls -la /home/bitcoin/.lnd/
total 40
drwxrwxr-x  4 bitcoin bitcoin 4096 Mai  9 20:31 .
drwxr-xr-x 13 bitcoin bitcoin 4096 Mai  5 15:58 ..
-rw-------  1 bitcoin bitcoin  232 Mai  9 20:31 admin.macaroon
drwx------  4 bitcoin bitcoin 4096 Mai  9 20:31 data
-rw-r--r--  1 bitcoin bitcoin  115 Mai  9 20:31 invoice.macaroon
-rw-rw-r--  1 bitcoin bitcoin  379 Mai  5 22:03 lnd.conf
drwx------  3 bitcoin bitcoin 4096 Mai  9 20:29 logs
-rw-r--r--  1 bitcoin bitcoin  183 Mai  9 20:31 readonly.macaroon
-rw-r--r--  1 bitcoin bitcoin  704 Mai  9 20:29 tls.cert
-rw-------  1 bitcoin bitcoin  227 Mai  9 20:29 tls.key
admin@thundroid:~$
```

Figure 14 – Check macaroon

Copy permission files and TLS cert to user “admin” to use ‘lncli’.

```
$ mkdir /home/admin/.lnd
$ sudo cp /home/bitcoin/.lnd/tls.cert
/home/admin/.lnd
$ sudo cp /home/bitcoin/.lnd/admin.macaroon
/home/admin/.lnd
$ sudo chown -R admin:admin /home/admin/.lnd/
```

Make sure that ‘lncli’ works by unlocking your wallet (enter ‘password [C]’) and getting some node information.

```
$ sudo systemctl restart lnd
$ lncli unlock
```

Monitor the LND startup progress until it has caught up with the testnet blockchain (about 1.3m blocks at the moment). This can take up to 2 hours. After that, you’ll see a lot of very fast chatter. Exit with ‘Ctrl-C’.

```
$ sudo journalctl -f -u lnd
```

Get some testnet Bitcoin

Now your Lightning node is ready. To use it in testnet, you can get some free testnet bitcoin from a faucet.

Generate a new Bitcoin address to receive funds on-chain

$ lncli newaddress np2wkh

```
'> "address":
"2NCoq9q7............dkuca5LzPXnJ9NQ"
```

Get testnet bitcoin:

```
<https://testnet.manu.backend.hamburg/faucet>
```

Check your LND wallet balance.

```
$ lncli walletbalance
```

Monitor your transaction (the faucet shows the TX ID) on a Blockchain explorer:

```
<https://testnet.smartbit.com.au>
```

```
LND in action
```

As soon as your funding transaction is mined and confirmed, LND will start to open and maintain channels. This feature is called "Autopilot" and is configured in the "lnd.conf" file. If you would like to maintain your channels manually, you can disable the autopilot.

Get yourself a payment request on StarBlocks (https://starblocks.acinq.co/#/) or Y'alls (https://yalls.org/) and move some coins!

Some commands to try:

List all arguments for the command line interface (cli)

```
$ lncli
```

Get help for a specific argument

```
$ lncli help [ARGUMENT]
```

Find out some general stats about your node:

```
$ lncli getinfo
```

Connect to a peer (you can find some nodes to connect to here: https://1ml.com/)

```
$ lncli connect [NODE_URI]
```

Check the peers you are currently connected to:

```
$ lncli listpeers
```

Open a channel with a peer:

```
$ lncli openchannel [NODE_PUBKEY]
[AMOUNT_IN_SATOSHIS] 0
```

Keep in mind that [NODE_URI] includes @IP:PORT at the end, while [NODE_PUBKEY] doesn't.

Check the status of your pending channels:

```
$ lncli pendingchannels
```

Check the status of your active channels:

```
$ lncli listchannels
```

Before paying an invoice, you should decode it to check if the amount and other information are correct:

```
$ lncli decodepayreq [INVOICE]
```

Pay an invoice:

```
$ lncli payinvoice [INVOICE]
```

Check the payments that you sent:

```
$ lncli listpayments
```

Create an invoice:

```
$ lncli addinvoice [AMOUNT_IN_SATOSHIS]
```

List all invoices:

```
$ lncli listinvoices
```

To close a channel, you need the following two arguments that can be determined with 'listchannels' and are listed as "channelpoint": 'FUNDING_TXID' : 'OUTPUT_INDEX' .

```
$ lncli listchannels
$ lncli closechannel [FUNDING_TXID]
[OUTPUT_INDEX]
```

To force close a channel (if your peer is offline or not cooperative), use

```
$ lncli closechannel --force [FUNDING_TXID]
[OUTPUT_INDEX]
```

See Lightning API reference (http://api.lightning.community/) for additional information

Outlook: Prepare for Bitcoin mainnet

In part 2 of this guide we will move the Thundroid Bitcoin & Lightning node to the Bitcoin mainnet, that uses a different blockchain. Like the small testnet blockchain, the mainnet blockchain records all Bitcoin transactions and basically defines who owns how many bitcoin. This is the most crucial of all information and we should not rely on someone else to provide this data. To set up our Bitcoin Full Node on mainnet, we need to:

- Download the whole blockchain (~ 200 GB)
- Verify every Bitcoin transaction that ever occurred and every block ever mined
- Create an index database for all transactions, so that we can query it later on
- Calculate all bitcoin address balances (called the UTXO set)

See "Running a Full Node" (https://bitcoin.org/en/full-node) for additional information.

You can imagine that the Thundroid is not quite up to this huge task. The download is not the problem, but to initially process the whole blockchain would take weeks due to its resource restrictions. We need to download and verify the blockchain with Bitcoin Core on a regular computer and then transfer the data to the Thundroid. This needs to be done only once. After that, the Thundroid can easily keep up with new blocks.

For the switch to mainnet, the mainnet blockchain should be ready, so we'll start this task now.

Using a regular computer

This guide assumes that you will use a Windows machine for this task, but it works with most operating systems. You need to have about 250 GB free disk space available, internally or on an external hard disk. As indexing creates heavy read/write traffic, the faster your hard disk, the better. An internal drive or an external USB3 hard disk will be significantly faster than one with a USB2 connection.

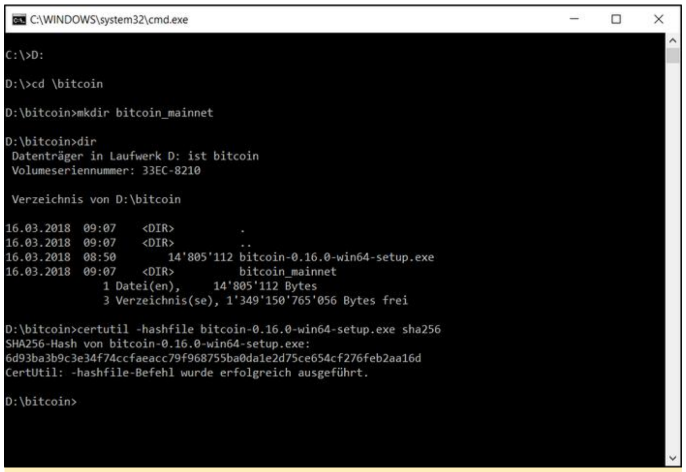
Download and verify Bitcoin Core

Download the Bitcoin Core installer from bitcoin.org/download and store it in the directory you want to use to download the blockchain. To check the authenticity of the program, calculate its checksum and compare it with the checksums provided.

In Windows, I'll preface all commands you need to enter with '>' , so with the command '> cd bitcoin' , just enter 'cd bitcoin' and hit enter.

Open the Windows command prompt ('Win+R', enter 'cmd', hit 'Enter'), navigate to the bitcoin directory (for me, it's on drive 'D:', check in Windows Explorer) and create the new directory 'bitcoin_mainnet'. Then calculate the checksum of the already downloaded program.

```
> G:
> cd itcoin
> mkdir bitcoin_mainnet
> dir
> certutil -hashfile bitcoin-0.16.0-win64-
setup.exe sha256
6d93ba3b9c3e34f74ccfaeacc79f968755ba0da1e2d75c
e654cf276feb2aa16d
```



```
C:\>D:

D:\>cd \bitcoin

D:\bitcoin>mkdir bitcoin_mainnet

D:\bitcoin>dir
 Datenträger in Laufwerk D: ist bitcoin
 Volumeseriennummer: 33EC-8210

 Verzeichnis von D:\bitcoin

16.03.2018  09:07    <DIR>          .
16.03.2018  09:07    <DIR>          ..
16.03.2018  08:50        14'805'112 bitcoin-0.16.0-win64-setup.exe
16.03.2018  09:07    <DIR>          bitcoin_mainnet
               1 Datei(en),     14'805'112 Bytes
               3 Verzeichnis(se), 1'349'150'765'056 Bytes frei

D:\bitcoin>certutil -hashfile bitcoin-0.16.0-win64-setup.exe sha256
SHA256-Hash von bitcoin-0.16.0-win64-setup.exe:
6d93ba3b9c3e34f74ccfaeacc79f968755ba0da1e2d75ce654cf276feb2aa16d
CertUtil: -hashfile-Befehl wurde erfolgreich ausgeführt.

D:\bitcoin>
```

Figure 15 – Windows Command Prompt: verify checksum

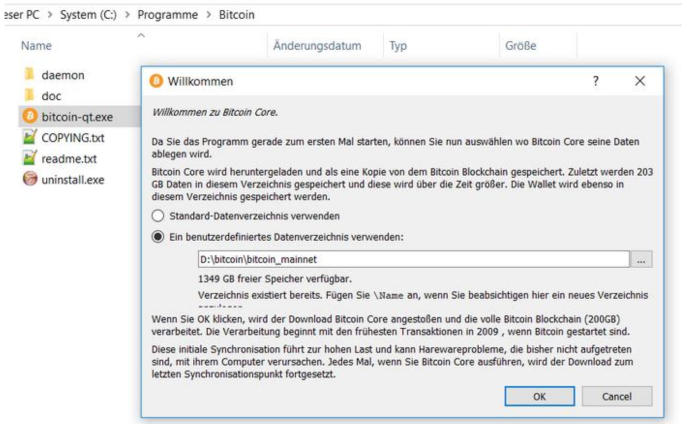
You can check this checksums with the the reference checksums on your Thundroid from the file we downloaded previously and have already checked for authenticity. Compare the following output with the checksum of your Windows Bitcoin Core download.

```
# on Thundroid, with user "admin"
$ cat /home/admin/download/SHA256SUMS.asc |
grep win
```

```
7558249b04527d7d0bf2663f9cfe76d6c5f83ae90e5132
41f94fda6151396a29 bitcoin-0.16.0-win32-
setup.exe
60d65d6e57f42164e1c04bb5bb65156d87f0433825a1c1
f1f5f6aebf5c8df424 bitcoin-0.16.0-win32.zip
6d93ba3b9c3e34f74ccfaeacc79f968755ba0da1e2d75c
e654cf276feb2aa16d bitcoin-0.16.0-win64-
setup.exe
42706da1a95b2db8c5808529f73c2063a0dd770f71e0c8
506bfa86dc0f3403ef bitcoin-0.16.0-win64.zip
```

Installing Bitcoin Core

Execute the Bitcoin Core installation file. You might need to right-click and choose “Run as administrator.” Install it using the default settings. Start the program ‘bitcoin-qt.exe’ in the directory “C:Program FilesBitcoin”. Choose your new “bitcoin_mainnet” folder as the custom data directory.



Willkommen zu Bitcoin Core.

Da Sie das Programm gerade zum ersten Mal starten, können Sie nun auswählen wo Bitcoin Core seine Daten ablegen wird.

Bitcoin Core wird heruntergeladen und als eine Kopie von dem Bitcoin Blockchain gespeichert. Zuletzt werden 203 GB Daten in diesem Verzeichnis gespeichert und diese wird über die Zeit größer. Die Wallet wird ebenso in diesem Verzeichnis gespeichert werden.

○ Standard-Datenverzeichnis verwenden

● Ein benutzerdefiniertes Datenverzeichnis verwenden:

D:\bitcoin\bitcoin_mainnet

1349 GB freier Speicher verfügbar.

Verzeichnis existiert bereits. Fügen Sie \Name an, wenn Sie beabsichtigen hier ein neues Verzeichnis

Wenn Sie OK klicken, wird der Download Bitcoin Core angestoßen und die volle Bitcoin Blockchain (200GB) verarbeitet. Die Verarbeitung beginnt mit den frühesten Transaktionen in 2009 , wenn Bitcoin gestartet sind.

Diese initiale Synchronisation führt zur hohen Last und kann Harewareprobleme, die bisher nicht aufgetreten sind, mit ihrem Computer verursachen. Jedes Mal, wenn Sie Bitcoin Core ausführen, wird der Download zum letzten Synchronisationspunkt fortgesetzt.

Figure 16 – Bitcoin Core directory selection

Bitcoin Core opens and starts immediately syncing the blockchain. Now, we need to set one very important additional setting in the “bitcoin.conf” file. If not set, the the whole blockchain will be useless and needs to be re-validated. Using the menu, open ‘Settings’ / ‘Options’ and click the button ‘Open Configuration File’. Enter the following line:

```
$ txindex=1
```

If your computer has a lot of memory, you can increase the database in-memory cache by adding the following line (with megabytes of memory to use, adjusted to your computer) as well:

```
$ dbcache=6000
```

Save and close the text file, quit Bitcoin Core using ‘File’ / ‘Exit’, and restart the program. The program will start syncing again.

Let the blockchain sync for now, this will take a day or two.

Before proceeding to mainnet

In part 2 of this guide, we will transition to the Bitcoin mainnet. This will be the point of no return. Up until now, you can just start over. Experiment with testnet bitcoin. Open and close channels on the testnet. It’s important that you feel comfortable with Thundroid operations, before putting real money on the line.

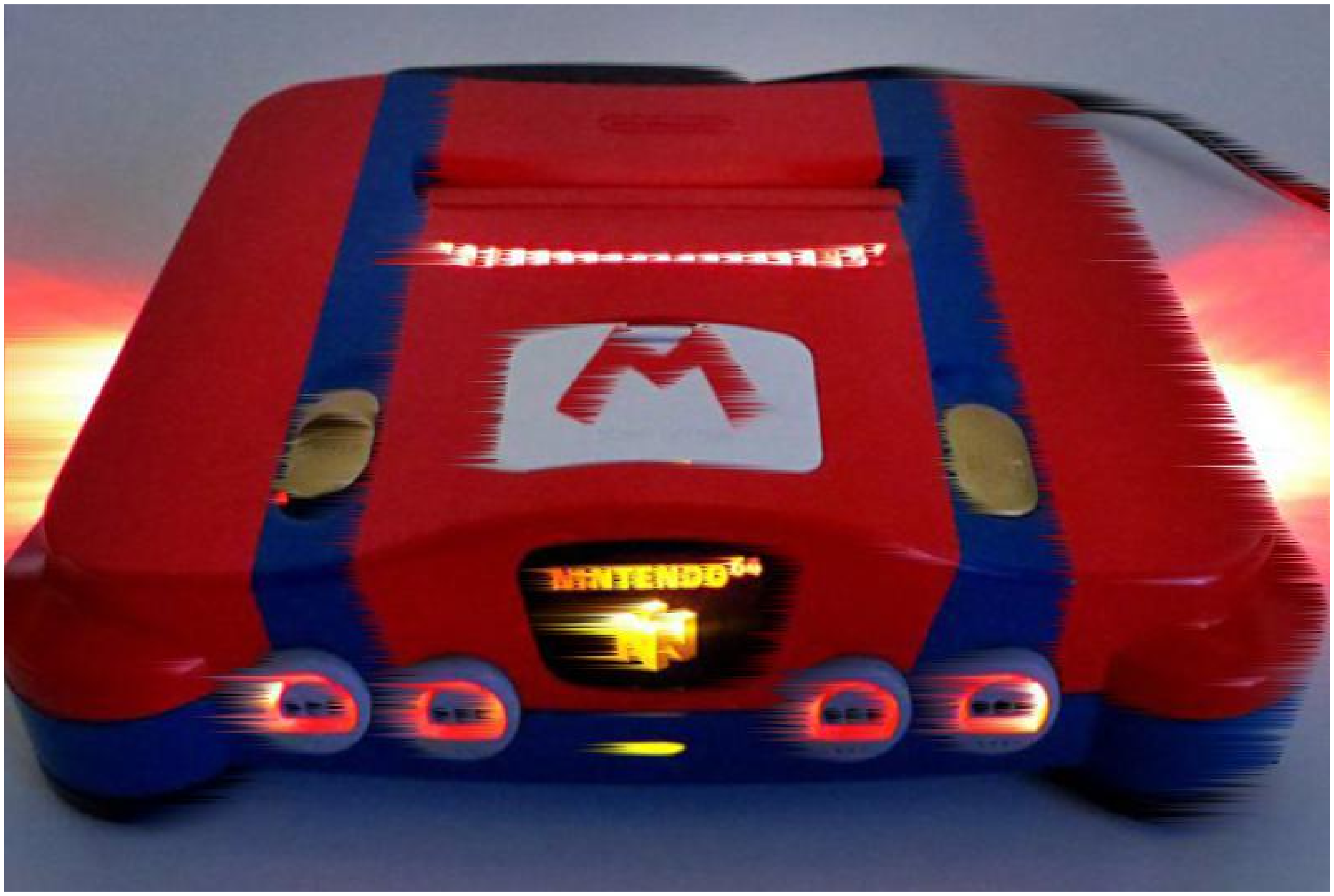
Once you switch to mainnet and send real bitcoin to your Thundroid, you have “skin in the game.”

- Make sure your RaspiBolt is working as expected. Get a some practice with ‘bitcoin-cli’ and its options. See Bitcoin Core RPC documentation (**https://bitcoin-rpc.github.io/**)
- Do a dry run with ‘lncli’ and its many options. See Lightning API reference (**http://api.lightning.community/**)
- Try a few restarts (‘sudo shutdown -r now’). Is everything starting fine?

See you soon in part 2, “The Perfect Bitcoin Lightning Node.”

Linux Gaming: Nintendo 64 Emulation – Part 1

June 1, 2018 By Tobias Schaaf Gaming, Linux



Nintendo 64 emulation has recently evolved to run on all ODROID devices, using either the Mupen64plus standalone emulator or the Libretro core for Retroarch. Now that it's widely available, I decided to do a comparison not only between the standalone version and the Libretro core, but also between the different ODROID platforms, in order to evaluate their capabilities in terms of emulating a Nintendo 64 (N64) console. Please note that this article covers only Linux emulation, and does not extend to Android, although there are several Nintendo 64 emulators available for Android, such as Mupen64plus and N64oid.

General information

It took a while to get N64 emulation to work on all the ODROID boards under Linux. However, now that it's functioning, it's quite fun and opens up lots of opportunities for classic gaming. Hopefully in the future, we will see more improvement and have even better support for N64 emulators on ODROID devices under Linux. For now, there are a few restrictions. Only the XU3 is able to use the Libretro core under Linux, which has better graphics, and is easier to control than the standalone Mupen64plus emulator. Mupen64plus runs on all other ODROID devices, such as the Exynos 4 series (X, X2, U2, and U3) as well as the newer but less powerful ODROID-C1. Both versions offer different plugins and methods of playing the games.

Graphics plugins

Whether you use Mupen64plus or the Libretro core, different plugins are used to display the game graphics. Mupen64plus is able to use a video plugin called rice, and another one called glide64mk2. The Libretro core offers rice, glide64 and one called gln64. While testing, I found that the best videocore depends on the game. However, it seems that glide offers better graphics features than the rice plugin, but has some minor glitches that are not present in the rice video plugin. Using the standalone Mupen64plus, rice is unable to perform aspect ratio scaling, and always scales the game to the full size of your video resolution. This distorts the picture, causing the characters and objects to appear wider than normal. The ODROID-C1 performs best when using the rice video plugin, since glide64mk2 doesn't work unless the color depth is reduced to 16 bit, which causes the transparency effects to become disabled. This will also cause issues if you try watching movies or want to start other applications that require more than 16 bit color depth. Since the initial tests on the C1 went poorly, I decided to retest every game in 16 bit using the glide64mk2 video core. There seems to be a workaround using framebuffer drivers instead of X11 drivers by adding some scripts in order to switch resolution and color depth, but since my ODROID GameStation Turbo image uses X11 drivers by default, I don't take the time to perform framebuffer tests. The glide plugin on the Exynos 4 series devices (X, X2, U2 and U3) is working well, and respects aspect ratio with an overall good quality, but it can be a bit slower than rice on some games. Glide also seems to render a darker picture than rice does, which is most likely due to some missing shader options with regards to shadows. The glide64mk2 plugin on the Exynos 4 devices is the preferred graphics plugin for Mupen64plus standalone. The XU3 can use rice, glide64 and gln64, but glide64 seems to be the best plugin for now on the ODROID.

Controllers

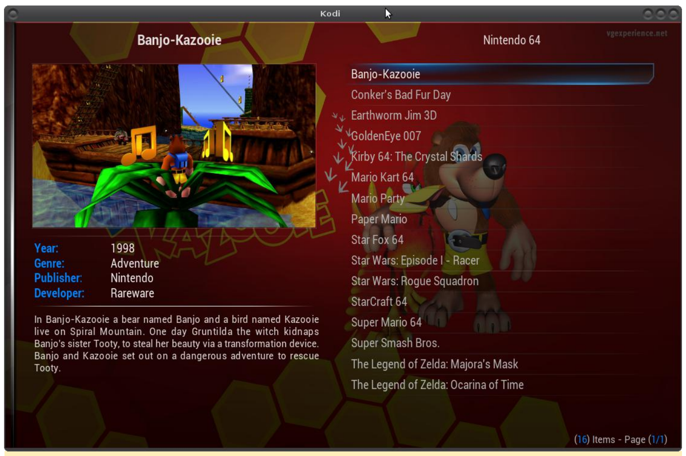
Joysticks are fortunately working fine on all ODROID devices, which means that all emulators (mupen64plus and libretro core) are fully supported with any game controller. The Mupen64plus emulator configures the controllers automatically, but not all controllers work perfectly with the default settings. Thanks to Retroarch on the XU3, you can setup any controller you want by manually configuring the buttons, so every controller is 100% supported. Normally you should be able to activate rumbling support of the controller, but I had trouble getting it enabled on all emulators and controllers. I was able to use it with some PS3-style controllers on the Mupen64plus standalone emulator, but I wasn't able to use rumbling with the Libretro core.

Sound

Sound is working well on all emulators, and I haven't found any major issues with it. Although one game had a delay in sound, which caused effects not to be synced with the action on the screen, that was an exception, and I haven't seen this issue with any other game.

Game selection

Are you ready to play your favorite Nintendo 64 games on the ODROID? Well, that's exactly what we want to try and find out: do your favorite games work? To answer that question, I did some research on what are generally considered the best games on the N64, then picked some of them to test, as shown in Figures 1 and 2. Hopefully you will find some of your favorite games in this list as well.



Banjo-Kazooie

Year: 1998
Genre: Adventure
Publisher: Nintendo
Developer: Rareware

In Banjo-Kazooie a bear named Banjo and a bird named Kazooie live on Spiral Mountain. One day Gruntilda the witch kidnaps Banjo's sister Tooty, to steal her beauty via a transformation device. Banjo and Kazooie set out on a dangerous adventure to rescue Tooty.

Banjo-Kazooie
Conker's Bad Fur Day
Earthworm Jim 3D
GoldenEye 007
Kirby 64: The Crystal Shards
Mario Kart 64
Mario Party
Paper Mario
Star Fox 64
Star Wars: Episode I - Racer
Star Wars: Rogue Squadron
StarCraft 64
Super Mario 64
Super Smash Bros.
The Legend of Zelda: Majora's Mask
The Legend of Zelda: Ocarina of Time

Figures 1 – List of some favorite N64 games from many different genres

Banjo-Kazooie

Banjo-Kazooie is a mix of jump-and-run platformer and action adventure. You play as Banjo the bear trying to save his sister who was abducted by a witch. He has a friend Kazooie the Bird, with whom you need to solve a couple of puzzles. Like most of the Rareware games, this one is fun and has a cute comic style.

Figure 2 – Banjo Kazooie)

U3

Generally, the game is running acceptably on the U3. It's sometimes a little laggy, especially during the intro. The introductions of Rare games are normally rather long and can't be skipped. On the XU3 and the Libretro core, you have the ability to increase the emulator speed, so that the introduction is over faster. I haven't seen that option on the standalone Mupen64plus emulator yet, which means a long wait time. Also, the fonts are not correct on the standalone emulator, which is slightly annoying. The game felt a little laggy after playing it for a while. I used the configuration options of the emulator to activate frameskipping with a maximum of three frames, which increased gameplay to full speed. With that setting, it was a really nice game to play, with only the font issue remaining. I left the frameskipping option set to three frames for all of the other games.

C1 – rice plugin

For the C1, I used the standalone emulator Mupen64plus, using the rice plugin, since I did not want to change the colors to 16 bit. Also, rice is a little bit faster than glide64mk2, and is better suited for the C1. I also had to activate frameskip for the rice plugin in order to get the game to run smoothly. Without frameskipping, the sound was lagging and was not a good experience. Although rice does not respect aspect ratio, it doesn't look bad. The issues with the fonts that happened with glide64mk2 do not exist on rice, so the fonts look normal. With frameskip activated, the game ran surprisingly fast on the C1, which was unexpected. If the game were to support a proper aspect ratio, it would run perfectly fine on the C1.

C1 – glide64mk2

Banjo-Kazooie runs on the ODROID-C1 using glide64mk2, but is extremely slow and no fun to play. The Rice plugin in 32 bit color is working much better than glide64mk2 in 16 bit.

XU3

XU3 uses the Mupen64plus Libretro core and Retroarch to emulate the game, and the experience on the XU3 is the best of all three platforms. The emulator runs glide2gl as a video plugin, which seems to be much better than the older glide64mk2, and does not render the colors as darkly. The Libretro core is missing the frameskip feature that the Mupen64plus emulator offers, which means that it can only perform as fast as the board that it runs on, which can lead to slowdowns, depending on the scenes. In Banjo-Kazooie, this happens in the introduction, but it's not bad. The graphics look much better using Libretro, and the game is fully playable.

Conker's Bad Fur Day

Conker's Bad Fur Day is another game from Rareware, and is similar to Banjo-Kazooie. However, it's not suitable for small children due to its reference to drugs and alcohol along with harsh language, despite its comic style. You will also find a few characters that are the same in both games. This game is a mixture of many genres, mostly jump-and-run and action adventure, but it feels more like a first-person shooter with a mix of other genres as well. The game is actually one of my favorites for the Nintendo 64, and some other people already gave it nice reviews: http://bit.ly/1bo6odW. I highly recommend the game for adults and teens.

Figure 3 – Conker's Bad Fur Day

U3

The U3 with the Mupen64plus standalone emulator is a little bit too slow for Conker's Bad Fur Day, and there are scenes where it feels somewhat laggy, which affects the controls. Sometimes they react too slowly, which is annoying during the jumping puzzles. The glide64mk2 plugin also makes the graphics very dark, especially during the cutscenes in the castle. When inside dark rooms, it's nearly all black in some spots.

U3 – rice plugin

While the U3 is having speed issues, the C1 fails completely because the C1 is simply not powerful enough to run a demanding game like Conker's Bad Fur Day. The rice graphics plugin also has many issues with this game, such as black borders and distorted graphics, which is not fun to look at. Although the game is generally working, it's rather slow. Some scenes are actually fast enough to be considered playable, but are far from full speed. Therefore, I would consider this game unplayable on the ODROID-C1.

U3 – glide64mk2

Conker runs better using the glide64mk2 plugin than with the rice plugin. It's still not full speed, but if you can tolerate a little lag, it's playable.

XU3

The XU3 offers the best gaming experience when running Conker's Bad Fur Day. The game, although not running at full speed, is mostly smooth on the XU3. The glide2gl plugin looks really good and only has a few issues with the game. I am not very far in the game right now, so I can't compare how the later levels perform, especially with driving tanks and using sniper modes.

Earthworm Jim 3D

Earthworm Jim is a nice platform action shooter about an earthworm named Jim, who got hit by an advanced space suit, transforming him into a hero. While the game was first a success on the SNES, SEGA Genesis, and even the Playstation 1, with the N64, it went a step further and transformed the game from a 2D platformer into a 3D action game.

Figure 4 – Earthworm Jim 3D

U3

The gaming experience of Earthworm Jim 3D on all ODROIDs is very nice. The U3 runs the emulator very fast and fluently, with some occasional minor graphical glitches. Since the game is rather colorful and bright, with light rooms and no shadows or dark corners, the glide64mk2 graphical darkness that affects other games is not a factor while playing Earthworm Jim 3D, which really improves the gaming experience.

C1 – rice plugin

Although the introduction and demo gameplay are fast, I couldn't start the game. The first scene where you talk to one of your friends is not only laggy, but also the window that is supposed to show the text remains empty and does not render anything. Clicking a button is also unresponsive, so you're stuck before the game even starts. It's likely that this is only a rice bug, but since I haven't tested glide64mk2 on the C1, I can't tell how well it's performing under that plugin. Therefore, I can only say that Earthworm Jim is not working on the C1.

C1 – glide64mk2

This game runs really well using the glide64mk2 plugin, with no graphical issues or slowdowns. Everything works as expected.

XU3

Since the C1 and U3 are powerful enough to play the game fluently, it's really no surprise that the gaming experience on the XU3 is perfect as well. If you like a good action platformer, this game is definitely a must have, but it does have some minor glitches. Some of the objects that you can collect are not displayed correctly. They seem to be too high, and you often only see a shadow where the object is supposed to be. You can still collect them, they are just invisible, although the game still works fine otherwise.

GoldenEye 007

GoldenEye 007 is found on everyone's top 10 list for the N64, since the game was revolutionary for it's time. Not only did it offer nice graphics, but it was known for its awesome multiplayer mode. The single player story and missions are very exciting as well and are fun to play as well. GoldenEye is a rather serious first person shooter with just the right touch of secret agent work. Although not as spectacular as Cate Archer, James Bond fights or sneaks through different levels and has to defend himself against enemy guards and spies. However, Cate Archer will always be my favorite spy in harms way. Although this game has very good reviews for the N64, I really don't like first-person shooters on consoles. Therefore it's not one of my favorite games, although it's nice to play.

Figure 5 – GoldenEye 007

U3

The game runs very well on the U3. Except for a short scene in the introduction, there was no slowdown either inside or outside of buildings. I had some issues when using a wireless XBox 360 controller with the right analog stick, which made game movement difficult. However, using only the left analog stick seemed to be a good workaround for playing the game.

C1 – rice plugin

The C1 has graphical issues with the rice plugin with this game as well. Neither the logo, nor the introduction are visible, and both are hidden behind a black border. The scene that caused a massive slowdown to 8 fps on the U3 is too much for the C1, and the emulator stops completely and eventually crashes. Observing the ODROID while running the game, I can tell that when the slowdown happens, the RAM usage skyrockets to the point where no RAM is available. After that, it uses the swap partition that I created, and after that, the out of memory killer terminates the emulator, which doesn't happen on the U3. I then switched to the rice plugin on the U3, and although the emulator was much slower than with glide64mk2, it was working properly with no black screen or memory issue, and did not crash. I therefore concluded that it was only an issue with the C1.

C1 – glide64mk2

The game is working with glide64mk2, but the speed varies from nearly full speed to laggy. It's playable, but not as good as on the U3 or XU3.

XU3

The game runs at a decent speed on the XU3 using the Libretro core. The graphics look really good, but it has occasional slow downs, although not in a way that prevents playing the game. I was also able to use the XBox 360 controller without any issues.

Kirby 64: The Crystal Shards

Who does not recognize little Kirby? This game is very kid friendly and has cute graphics. The pink marshmallow-like buddy can suck in its enemies and swallow them in order to absorb their powers. The N64 version has beautiful 3D graphics and is rather easy to play, which makes it perfect for children. Although the game is rendered in 3D, the levels are very linear. You can go left, right, up and down, but are not able to walk freely on the map, which probably greatly reduced the map size and allowed extra performance for effects. The game looks similar to Mario 64, but without the free movement in all directions.

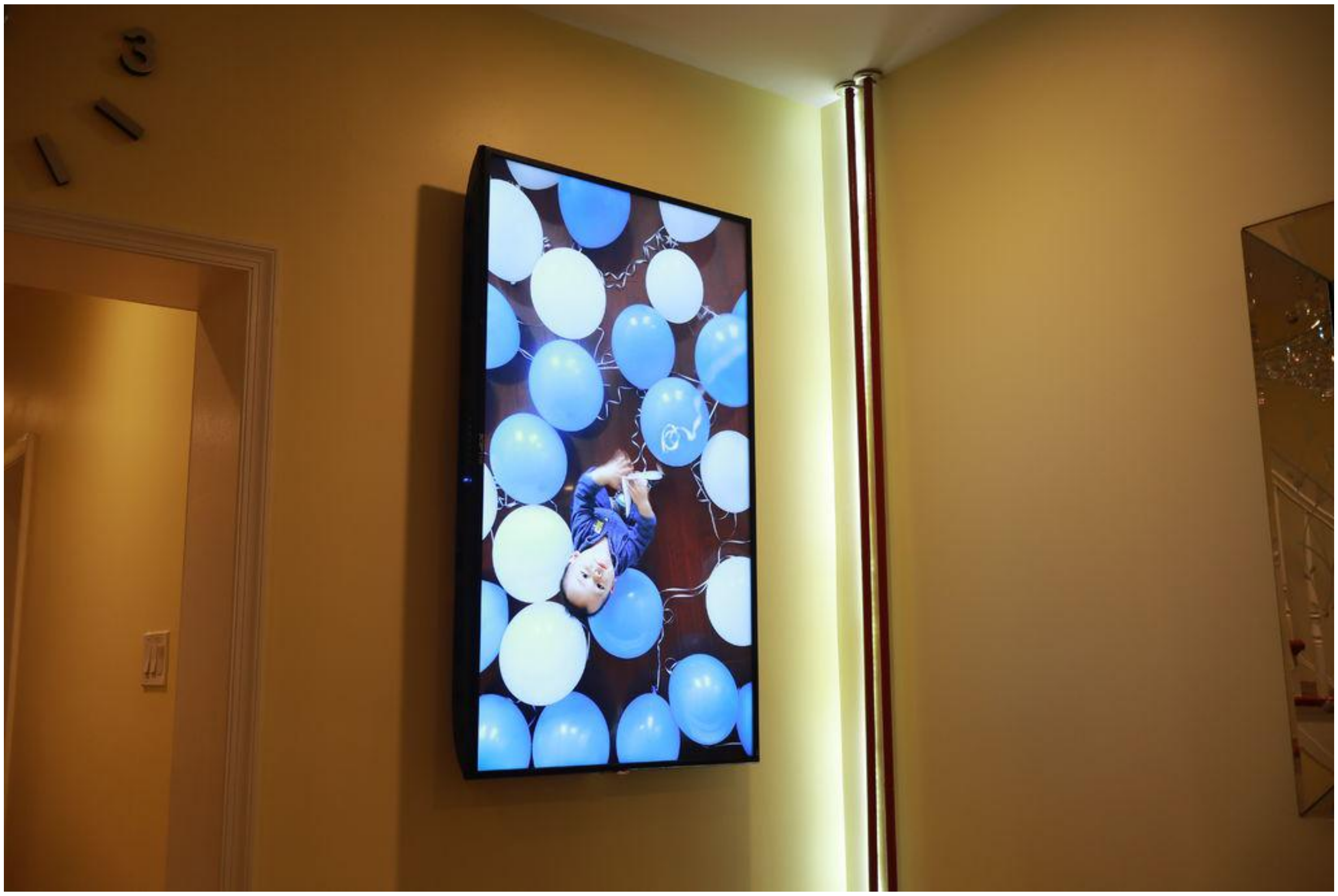
Figure 6 – Kirby 64: The Crystal Shards

U3

The general experience is very good, and the game runs perfectly fine in full speed. However, it has some graphical glitches with the ground and shadows which

Digital Photo Frame: 55 inch 4K Digital Photo Frame Display for Around $400

June 1, 2018 By @tung256 ODROID-C2, Tinkering



There are lots of tutorials on how to make an awesome digital photo frame with a Raspberry Pi. Unfortunately, the Raspberry Pi does not support 4K resolution. The ODROID-C2 can easily handle 4K resolution, but none of those Raspberry Pi tutorials work for the C2 unit. It took me 30+ hours to get where I am today. You can buy one from Memento (mementosmartframe.com) for $900 USD for a 35 inch, 4K frame, or a Samsung frame TV for around $1300 USD.

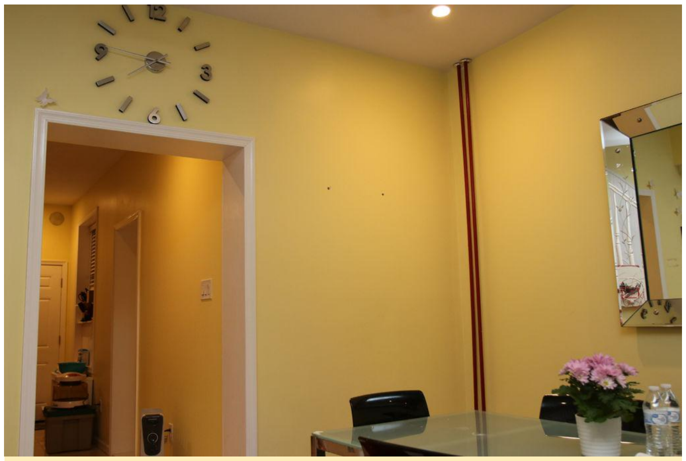


Figure 1 – This wall has been empty for three years

I wanted to have a nice poster or backlit display, but those are costly and can only display one photo at a time. After a bit of a wait, a 55 inch 4K TV was available at Walmart for $260 USD. Add $26 USD for a three year warranty and taxes, and the out-the-door cost was $306 USD.

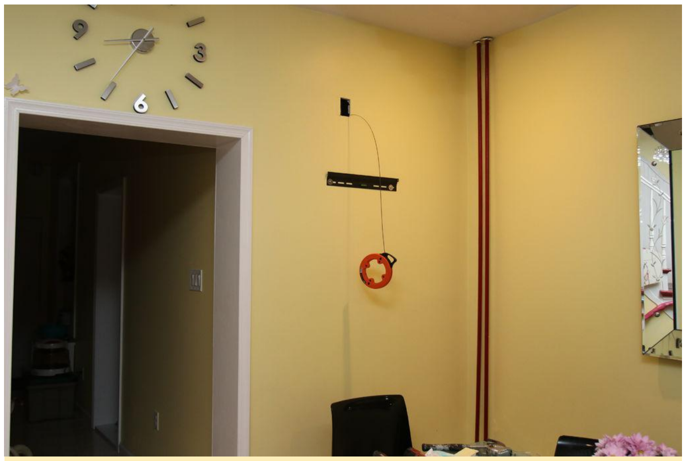


Figure 2 – Add an outlet near the TV for a clean look

I was able to fish a Romex 12 gauge wire from a nearby outlet. Be sure to turn off the electricity before doing this work! There are detailed tutorials available on YouTube for installing an outlet. I actually forgot to take photos and videos while I installed my own. Sorry!

Gather All the Materials You Will Need

- 4K TV 55 inch Sceptre from Walmart, $260 USD as of April 30th, 2018.
- ODROID-C2 with barrel plug power adapter, $65 USD with 4 day shipping in USA. Do not use the micro USB cable to power, it is not good enough. I bought my unit and power cable from Ameridroid.com
- High speed HDMI cable from Monoprice.com, $5 USD. Certified to work with 4K 60hz 4:4:4 chroma
- 32GB flash drive to hold photos, $15 USD.
- 8GB microSD card, $4 USD.
- Wireless USB and keyboard combo, $30 USD.
- Optional USB WiFi adapter, $10 USD.

All prices subject to change. My costs were actually lower because i got some items used or already had on hand for other projects.

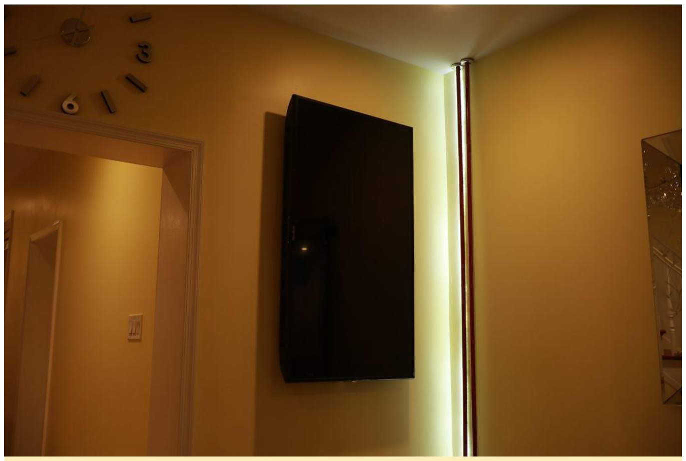


Figure 3 – The ODROID-C2 photo frame mounted on the wall

Install Ubuntu mate from https://wiki.odroid.com/odroid-c2/os_images/ubuntu/v2.4. Burn your downloaded ISO image onto the microSD card using Win32disk imager. Insert your microSD card into the ODROID-C2. Connect the HDMI cord from the C2 to the TV. Turn everything on and make sure all the materials are connected to the ODROID-C2. You can skip connecting the USB drive and USB WiFi for now. Let Ubuntu initialize and finish everything. Once all done, it will ask you to log in.

User ID: odroid Password: odroid

Enable auto login so you don't have to manually log in each time. Go to menu Applications/System Tools/Mate Terminal and type in:

```
$ sudo nano
/usr/share/lightdm/lightdm.conf.d/60-lightdm-
gtk-greeter.conf
```

If it asks for a password, type "odroid". Type in the autologin line. so the final file should look like this:

```
[Seat:*] greeter-session=lightdm-gtk-greeter

autologin-user=odroid
```

Press CTRL+X to exit, or Y to save file. Install FEH to view photos. In the same terminal window, type the following commands, typing Y to confirm, if it asks:

```
$ sudo apt-get install feh
```

My photos are edited to be 4K resolution at 3840 x 2160 pixels. Copy your photos to a USB drive and insert into the ODROID-C2. Use your mouse to navigate to the directory /home/odroid and right-click to create a Create Document/empty file. I named it pixx.sh, but you can name it whatever you want. Open pixx.sh, add these codes in:

```
$ sleep 15

$ feh --quiet --fullscreen --borderless --
hide-pointer --randomize --slideshow-delay 30
/media/odroid/38C1-602E/*
```

Your USB drive name will be different from mine. In my case, the drive is named "38C1-602E." To find your USB name, just navigate to media/odroid and you will see. Change the slideshow delay value of 30 seconds to whatever you want. Save the file and close. Right click on pixx.sh to view its properties. Make it "executable" in one of the options.

Add pixx.sh to autostart menu. Go to the menu System/Preferences/Personal/Startup Applications and click on Add:

- Name=slideshow
- Command=(choose the pixx.sh file wherever you saved it)
- Comment= slideshow autostart, click on Add and close

Disable the screensaver by going to System/Control Center and choosing Look and Feel, and then ScreenSaver. Disable "Activate Screensaver" and anything else that would trigger idle mode. I've forgotten all the settings, but they are all here. Go back to the terminal window and type:

```
$ sudo reboot
```

This will reboot the C2. Once rebooted, it should automatically login, wait for 15 seconds, and then start playing photos from your USB drive. Hit ESC on your keyboard to exit FEH if you need. I leave mine running nonstop. I only turn off the TV as needed.

Optional Steps

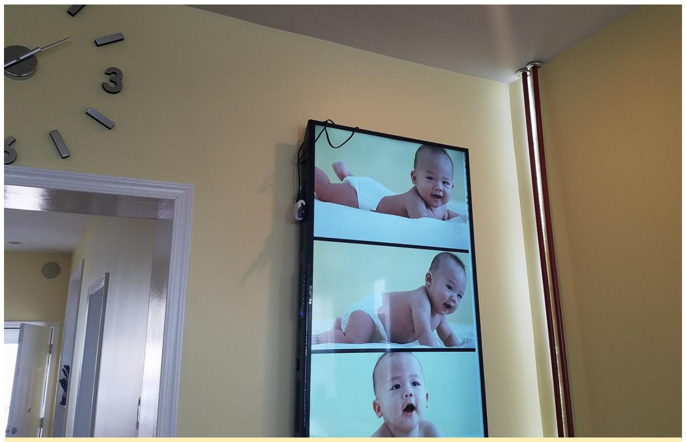


Figure 4 – The photo frame fully assembled and displaying family photos

Add more photos to the USB flash drive by copying and pasting into the USB drive manually or via FTP. To upload via FTP–so you don't have to physically disconnect the USB drive–make sure the USB WiFi adapter is plugged in. Go to the upper right hand corner of Ubuntu menu and connect to your WiFi network. Download FileZilla and connect to the ODROID via SFTP protocol, not FTP protocol. Enter your C2 IP address in host field:

user: odroid password: odroid login type: normal

Upload to your media/odroid/usb drive name and reboot the C2 for FEH to load the new photos into memory. To rotate the display screen into portrait mode, go to the terminal and type in:

```
$ sudo nano /etc/X11/xorg.conf
```

Add in this line:

```
Option "Rotate" "CCW"
```

So the end result looks like this:

```
Section "Device"
Identifier "Mali FBDEV"
Driver "fbturbo"
Option "fbdev" "/dev/fb0"
Option "Rotate" "CCW"
Option "SwapbuffersWait" "true"
EndSection
```

Exit and save, then reboot the computer:

```
$ sudo reboot
```

Once your TV has loaded up again, the display should be rotated into portrait mode.

Troubleshooting

- Ubuntu rebooting itself There is not enough power. Don’t power up the C2 unit via the micro USB port. Power the C2 unit via the barrel plug that Ameridroid.com sells for around $7 USD. I spent at least 20 hours on troubleshooting this.
- TV shows blank screen or no signal Try another HDMI cable or Try another power cable for the C2.

OS Spotlight: ODROID GameStation Turbo

June 1, 2018 By Tobias Schaaf Gaming, ODROID-C2, ODROID-XU4

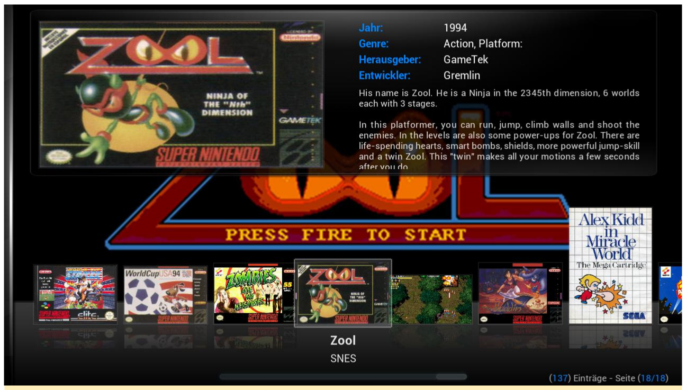
One of the biggest projects that I am working on for the ODROID community is the ODROID GameStation Turbo image, which works as a frontend for both games and media playback. It's intended as an entertainment system that allows you to control your ODROID just by using a game controller in your hand without ever having to touch the keyboard in order to watch movies, listen to music, or play your favorite games. For a better understanding of the usefulness of the image, I want to give you an inside view on how the image was created, what motivation I had, and how you can adapt it to meet your own needs.

Motivation

The first ARM based devices I considered was actually the Open Pandora, but by the time I was ready to to buy one, it was not available. However, even when it came available again, it was so expensive that I couldn't afford it. Finally, when I had enough money, I was already skeptical and was looking into other options. The Pandora board is an ARM-based single core device with just 1GHz and only 512MB RAM for $700, so was it really worth it? Well, although the community was and continues to be awesome, and it's a fully portable device (like a Nintendo DS), it was way too expensive, in my opinion, for what it could do. By that time there were better devices available, including the ODROID. After seeing the ODROID-X2 in an article on a German IT News page, I got really hooked up to it. By the time Ubuntu was announced for the ODROID, I bought myself an X2. However, what I wanted the Pandora for was to play games, and the ODROID didn't have too many games at that time (2012). As my nephews got older, I figured that I could make something really nice for them that would grow with them as they grew. First, the ODROID could be a console for playing games, and later, it would function as a PC in order to do homework and learn Linux. That was my goal and motivation for creating the image.

Steps to success

The first step towards achieving that goal was to generate content, so lots of games and emulators had to be ported to the ODROID. If you read the ODROID Forum's Ubuntu (All Linuxes) section, you will find many games and programs there that I ported myself. It was hard work, since I went from knowing basically nothing about porting games or compiling software on Linux to what I know now. ODROID was a great help in learning new skills and getting better at knowing that kind of Linux stuff. Now, I know how to optimize certain programs, how to set different optimization flags, and when those flags are needed. I learned more about how ARM CPUs work and, especially the hardware differences between the Hardkernel boards. My first project was then to port lots of games, and compile some emulators as well. If you've read my columns from the previous issues of ODROID Magazine, you will find lots of informations on what games are actually running on the ODROID, and it keeps getting bigger. The next big step was to make it easy, even for children, to use and work with Linux and play games. I started building Gamestation Turbo from the Linaro Ubuntu 12.04 Image. I preferred that over all other operating systems because of its Unity Desktop. Unity is easy to use and understand even for people that never used Linux before. It might not be the best desktop environment for all applications, but it's colorful, and easy to handle. For someone who has never used Linux before, it's a very nice way to get started with it. My first approach was to get the programs easy enough to run on all system. I gave all applications and games that I created a .desktop icon file, so you can find it in Unity or just place a shortcut on the desktop. This worked fine for games, but not for emulators, since emulators normally use their own file browser interface to load the ROMs. Although adults might be able to handle starting all of the games manually, kids will have no clue what certain words mean, and it's hard to see which games are available, or what to search for on Unity. It was immediately clear that I needed some kind of front end in which to start the games. I had already used XBMC on an old PC that functioned as a Home Theater PC (HTPC), and subsequently discovered a nice XBMC addon called Rom Collection Browser (RCB). RCB allows you to organize the emulator ROMs in the same way that you can organize your video collection. It's even able to download preview images and covers and gave a short description to the games, just like video services do for movies.


Jahr: 1994
Genre: Action, Platform:
Herausgeber: GameTek
Entwickler: Gremlin

His name is Zool. He is a Ninja in the 2345th dimension, 6 worlds each with 3 stages.

In this platformer, you can run, jump, climb walls and shoot the enemies. In the levels are also some power-ups for Zool. There are life-spending hearts, smart bombs, shields, more powerful jump-skill and a twin Zool. This "twin" makes all your motions a few seconds after you do

Figure 1 – Rom Collection Browser in ODROID GameStation Turbo

Knowing this, the idea came up to use XBMC as a frontend and set it up in a way for children to play and have fun with, or better to say to set it up in a way that even a child could play with it. During that time, hardware accelerated XBMC and video playback was out of question since it development hadn't yet been completed. The XBMC version that came with Ubuntu 12.04 was XBMC 11 (Eden), which was working, but not very fast due to software decoding. Although the Menu was working smoothly, video playback was not smooth. Still, it was working well enough that I could test out the Rom Collection Browser, and experimented with how to set everything up. When the first image of XBMC 12 (Frodo) for ODROID came out, it still did not support hardware-accelerated movie playback, but did come with OpenGL ES 2 support. Things got a little difficult to manage around that time, since compiling Hardkernel's XBMC source code didn't work for me, and the version provided has no joystick support, which I considered very crucial to my plans I decided that, since it was planned as a gaming platform, video playback was not the most important feature, and you still could play everything that was not HD smoothly as long as it was 720p or lower. For children, it generally doesn't matter if their favorite anime or cartoon is in HD or just SD. Well, it was about that time that a working hardware accelerated XBMC image was released, and I was able to rebuild the image with the necessary joystick support. Shortly before I released the first version of GameStation Turbo, I moved over to a fully working XBMC version.

Parts that were included

After i decided how the image should work, it was important to put all the tiny pieces together into a nicely packaged image, and for this, some work and different kinds of programs were necessary to achieve what I wanted to have. The first priority was the Operating System, which had to be very stable, easy to maintain, and with an interface that many people are already familiar with. The only choice here was between Ubuntu 12.04 and Debian Wheezy. Every other image was either unstable (Debian Jessie/Sid) or wouldn't be supported for very long (Ubuntu 13.04 or newly released 13.10). Ubuntu 12.04 is an LTS version that is supported until 2017, which is always good, however, Debian Wheezy outperformed Ubuntu 12.04. I also found that while developing for Debian Wheezy, the programs were most likely to run on Ubuntu 12.04 and higher without any issue, but not the other way around. So, I decided to use Debian Wheezy and LXDE, which uses less than 150 MB RAM even with XBMC and a couple of other programs running. After that, it was a question of putting together the right kind of software to turn the ODROID into a gaming machine.

Rom Collection Browser

I used Rom Collection Browser as a base to install different kind of emulators such as Retoarch, Mednafen, PPSSPP and ScummVM. Once the basic setup was done, it turned out that not everything was working with a gamepad out of the box, so I added antimicro which is able to map certain keys to a joystick button to fill the gaps where the joystick drivers did not work. I also maintain my own kernel builds and include the header files as well, since some parts of the kernel provided by Hardkernel did not meet my needs, and header files were not included. Besides that, there was a huge space difference between hardkernel's kernel modules and the one that I produced. The size was Hardkernel's build was over 300MB, but mine was only 16MB of my own build, which was achieve just by stripping the modules. My scripts also allows users to install or uninstall kernel packages, instead of just copying the kernel directly over the existing files.

Complications

The biggest problem for me was how to get all the parts to work with each other, and make it easy for people to use the image, even if they do not have knowledge on how to set it all up. The Rom Collection Browser was somewhat difficult to use for a beginner, since you had to choose the emulator, starting parameters and give the ROM files standard extension to set it up and get it to run. So I had to come up with a system that made it rather easy for a user to deal with that. There was another issue. I wanted to have full Joystick (GamePad) control, but some emulators required keys as well, such as Retroarch and Mednafen which required the ESC-key to end the current game and go back to XBMC, and also MAME games which required to enter an "OK" to continue.

Configuration

One problem with preconfiguring the Rom Collection Browser was that it requires the full path of where the emulator and ROMs are located, and what file extension is used to search for ROMs. This can be rather confusing for someone that has never worked with the Rom Collection Browser. That's why I pre-selected the emulator and games, and created a folder structure where the ROMs should be placed, in order for the Rom Collection Browser to find the games. Additional emulators can be added by pressing the C key in the Rom Collection Browser and selecting "Add a new ROM collection". There you have to give the path to the emulator, the path to the ROMs, the path where it should store information and pictures and the extension of the ROMs it should look for. The configuration file for ROM Collection Browser is stored in

```
/home/odroid/.xbmc/userdata/addon_data/script.
game.rom.collection.browser/config.xml
```

By editing this file, you can alter other options as well, such as if a .zip file should be extracted into a temporary folder, and whether to look for a ROM inside of a .zip file (which, for example, has to be deactivated for MAME games). If you're experienced enough, you can even add new collections directly in this file.

Starting an emulator

Although starting a ROM directly through the emulator will definitely work, it has a couple of disadvantages. First of all, XBMC will still be running in the background and will use some of the resources needed for a better gaming experience. Second, as mentioned before, some emulators need extra keys that are not mapped to a button. If using a joystick that is not supported, you need need antimicro to map the buttons for you. If so, you need to make sure that antimicro is started when you need it, which might not always be the case. Directly starting antimicro along with the emulator didn't work either. To solve these and other issues, I let XBMC run a small scriptInstead of directly starting the emulator. In that script, the emulator is started, which then runs the ROM file which is given to the script as a command-line parameter from XBMC. That way, I can define different steps to make sure the emulator works the best.

```
Example: running an SNES game with Retroarch:
#!/bin/sh
/usr/bin/killall -STOP xbmc.bin
if [ `ps aux | grep antimicro | grep -v grep |
wc -l` -lt 1 ]; then
antimicro --tray --profile
/home/odroid/joydev.xml &
else
/usr/bin/killall -CONT antimicro
fi
/usr/local/bin/retroarch -L
/usr/local/share/retroarch/cores/working/snes9
x_next_libretro.so "$1"
/usr/bin/killall -CONT xbmc.bin
/usr/bin/killall -STOP antimicro
```

Reading through the above code, you can see that XBMC is set to suspend mode, which means it won't use any processing power while we run our emulator. After that, I make a check to see whether antimicro is running, and either load it with the required profile file, or resume it in case it's still running. Then, I call the actual emulator. Here I can pass command parameters which allows me to configure the emulator. After the emulator is terminated by exiting the emulator, XBMC is resumed and antimicro is suspended. Just after the script is completely done, it switches control back to XBMC. This allows for some cleanup work that may be necessary. I wrote quite some scripts to adapt to different circumstances. For example, the ScummVM and Amiga script is a little bit more complicated; but all in all it's pretty much always the same.

- Suspend the processes you don't need (for example XBMC)
- Setup your environment by preparing the system with the stuff you need (for example, loading antimicro with the right profile)
- Call the emulator and give it the parameters that you think you'll need. The "$1" represents the ROM file that is getting passed by XBMC as a parameter.
- Do some clean up work and resume the processes that you suspended earlier

All the scripts that I used for launching emulators are located in /usr/local/bin/, where you can review, improve or add your own scripts.

FAQ

Every now and then, I receive some questions about my image which I would like to address here as a FAQ.

Where do I have to put the ROM files for my games? Navigate to /home/odroid/ROMS, where you will find a structure of folders already created for each type of ROM you want to play, such as GBA and SNES. Please check the forum post at **http://bit.ly/1nVvQqz** for details on which file extensions are supported.

Is there a way to load ROMs from an external storage? Copy the contents of /home/odroid/ROMS to your external storage device and then auto-mount the external device to /home/odroid/ROMS by adding it to /etc/fstab, or using /etc/rc.local to make it permanent.

What joystick/gamepad are supported? I built the image for use with an Xbox 360 wireless controller and Xbox

360 wireless USB receiver. So if you have that hardware, the image should work out of the box with no modification necessary, unless I forgot something again. Besides that, every joystick/gamepad that is supported by Linux should work as well, but you will have to adapt settings for your device. Therefore you have to change the joystick settings on the individual emulators. Running Mednafen, you can simply press ALT+SHIFT+1 to setup controlls for your device. The setup program is easy to understand. The second player, if supported by emulator can be setup with ALT+SHIFT+2, and so on. For Retroarch, it's a bit more complicated. Quit XBMC, open a terminal, and type retroarch-joyconfig, then follow the instructions on the screen. At the end, you will get a long list of configuration parameters in the Terminal window. Copy this list, then open the file /home/odroid/.config/retroarch/retroarch.cfg, where you will find the same parameters listed. Replace the already existing parameters with the ones you got from retroach-joyconfig, and your device should work in retroarch.

XBMC unfortunately does not support a lot of devices for joystick support. Although Xbox 360 controllers are working fine, others do nothing at all. With PPSSPP, you can change the controller configuration within the emulator by just going into the menu. However, on PPSSPP, the way controllers are implemented is rather sluggish so only a few really work well. In worst case, you can't even use the keyboard anymore. since the controller settings won't allow you to hit certain direction keys. If that happens, delete the file /home/odroid/.config/ppsspp/PSP/SYSTEM/controls.ini and start over. If all else fails, remove any mappings for the controller and keep the settings for Keyboard only, which should always work. Then, use your best buddy, antimicro! If you use a different joystick device and really have trouble getting PPSSPP or XBMC to

OGST Gaming Console Kit for the ODROID-XU4

June 1, 2018 By Justin Lee Gaming, ODROID-XU4, Tutorial

The OGST Gaming Console Kit for the ODROID-XU4 kit allows you to build your own gaming console with a powerful ODROID-XU4 or ODROID-XU4Q. Its attractive design includes a fancy 2.4" LCD to show programmable game logo animations, and is specifically designed to work with the popular ODROID GameStation Turbo disk image. The Gaming Console Kit is available at the Hardkernel store for USD$24, and pairs well with the new GameSir G3w USB Controller Joystick. The ODROID-XU4 or XU4Q, microSD card or eMMC module, 5V/4A PSU and USB game controller/joystick are not included.

Kit contents

- A. Top case
- B. Bottom case
- C. OGST-XU4 LCD Board
- D. USB2.0 extension cable
- E. 30-pin flat cable
- F. USB-port cover
- G. rubber feet set
- H. screws
- I. USB3.0 extension cable *(For optional 2.5" USB HDD storage)*

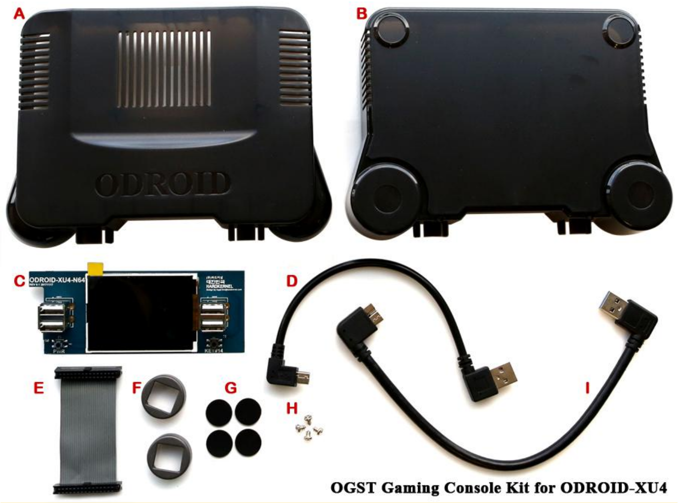


Figure 1 – Annotated GameStation kit

For more information, please visit our GameStation WiKi page.

Assembly

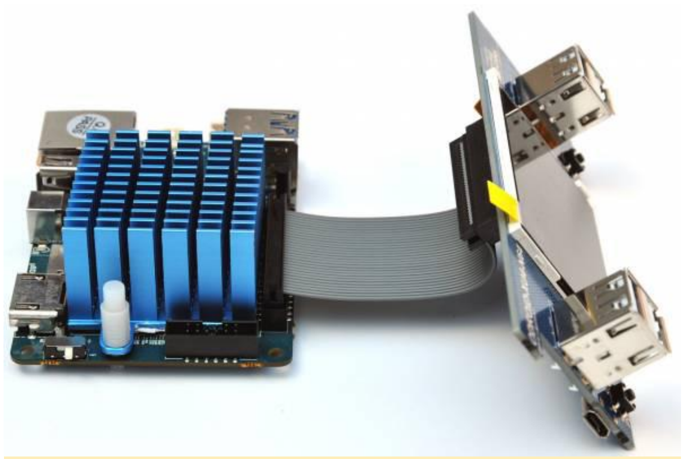


Figure 2 – Use a 30-pin flat cable (E) to connect the XU4 and the LCD board (C). Then remove the LCD surface protect film. The ODROID-XU4 is sold separately

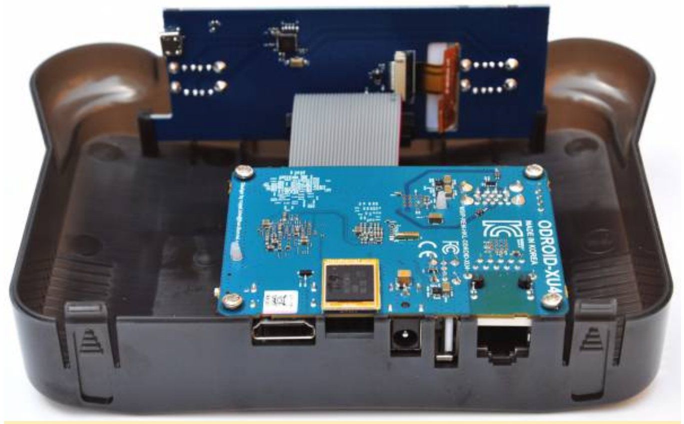


Figure 3 – Assemble the XU4 + LCD board in the top case (A) and fasten the screws (H)

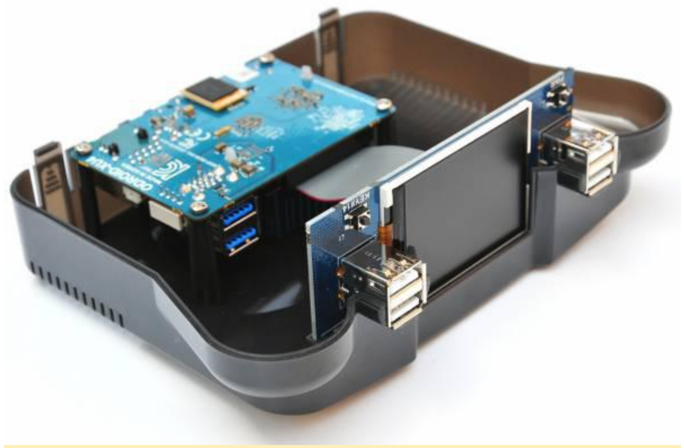


Figure 4 – Connect the XU4 board and the LCD board with the USB2.0 cable (D)

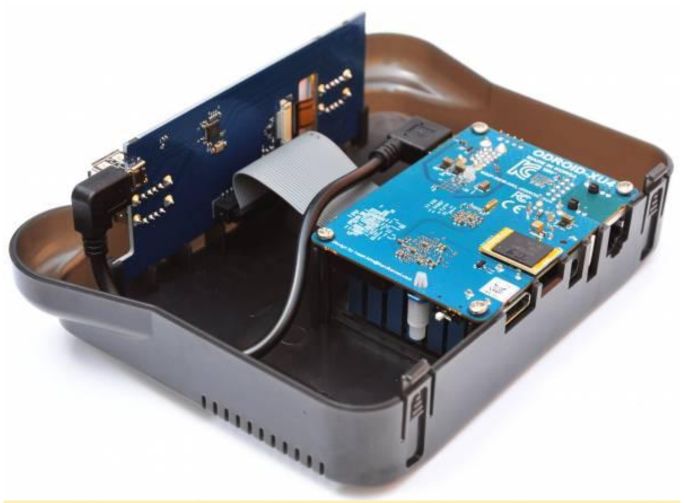


Figure 5 – Connect the XU4 board and the LCD board with the USB2.0 cable (D)

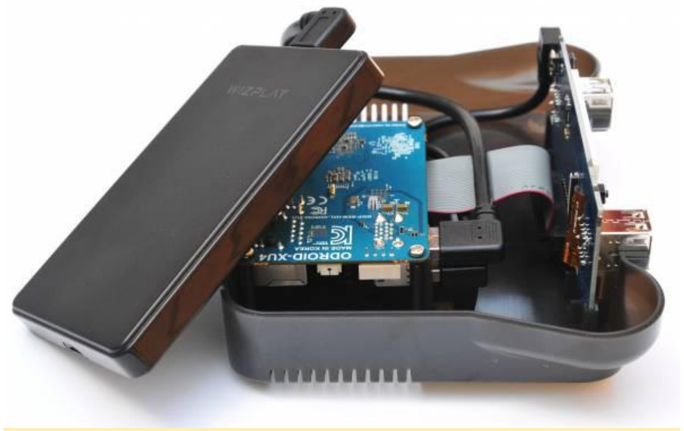


Figure 6 – Connect the XU4 board and the LCD board with the USB2.0 cable (D)

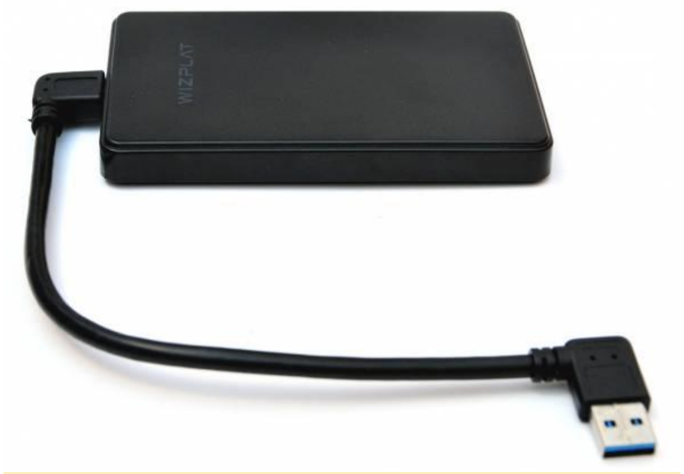


Figure 7 – You can connect a 2.5" HDD box using USB3.0 cable(I) included in the package and secure the HDD enclosure to the bottom case with velcro tape. The package does not include an HDD. If your USB HDD storage consumes more than 500mA, you might need to use a 5V/6A PSU instead of 5V/4A PSU

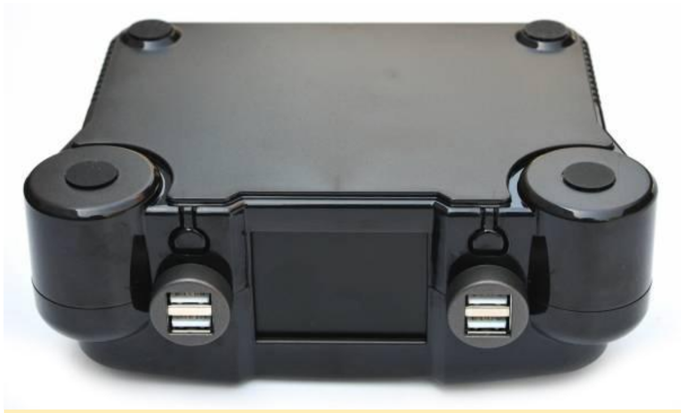
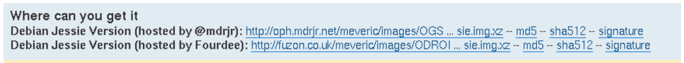


Figure 8 – Assemble the bottom case (B) and fit the USB covers (F). Then attach the rubber feet (G) to the bottom case

Software

ODROID GameStation Turbo is an entertainment based ODROID OS image aimed for watching movies on Kodi, listening to music, playing games from your Desktop as well as playing retro games over different emulators from within Kodi, EmulationStation or Attract-Mode as a frontend.

The image can be downloaded from the ODROID Forum in the ODROID GameStation Turbo for XU3/XU4 thread. Scroll down until you find the text shown in Figure 9.



Where can you get it
Debian Jessie Version (hosted by @mdrjr): http://oph.mdrjr.net/meveric/images/OGS ... sie.img.xz -- md5 -- sha512 -- signature
Debian Jessie Version (hosted by Fourdee): http://fuzon.co.uk/meveric/images/ODROI ... sie.img.xz -- md5 -- sha512 -- signature

Figure 9 – ODROID GameStation Turbo download locations

Click to select one of the links. The two links here should be identical but if one is slow for you, you can try the other link. The md5, sha512 and sig files are for people that want to make sure the downloaded file is not corrupted and wasn't tampered with, but you can ignore them if you don't know what to do with them.

Once you've downloaded the image, follow the guide at the easiest way to install new OS to flash the image on your SD card or eMMC card. You'll need at least an 8GB SD/eMMC card to use this image, but due to the nature of the image (games and videos) I suggest using larger SD/eMMC cards such as 32 GB or 64 GB (or using an additional external storage unit).

If you use an SD card, I suggest using a SanDisk Ultra (not Ultra Plus, or Extreme), since it seems the most compatible brand/model available (it's also rather cheap), and other models may have issues (limitation of the XU4 hardware). Once you've flashed the image on your SD or eMMC card, plug it into the ODROID and boot the system.

When you first boot the image, it will boot into the desktop and start Kodi, and will then reboot after a short while to finish some first boot maintenance. You don't have to do anything, it will happen automatically and you should simply wait for the system to reboot. This may take up to 5 minutes, but generally should happen much faster.

After the system is rebooted, it will boot back into the desktop and into Kodi. Exit Kodi by pressing "S" on the keyboard and select Exit/Quit Kodi. You're now on the desktop of your OS and can access all programs that you want, like the web browser, music player, install new programs, or can go back to Kodi to watch movies. But let's start with some basics and then put some games on the system so we can play them.

At this point, you will want to be connected to the Internet. Either plug in your WiFi Module or Ethernet cable onto your ODROID. On the desktop of your OS, make sure you are connected to the internet. To do so, click on the top right menu to check/connect to internet. Select the desired WiFi network and enter your password.

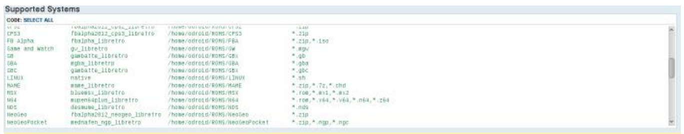
Next, open a new terminal window, either by pressing the MATE Terminal symbol in the Quickstart list on the top of the screen, or on the Desktop, or by pressing the key-combination CTRL + ALT + T. In the newly opened terminal Window type the following commands to upgrade the system.

```
$ su
$ apt-get update && apt-get upgrade && apt-get
dist-upgrade
```

This will log you in as root and ask for your root password which is "odroid" (without quotes) by default. After you're logged in as root, the system will update to make sure you have all the latest drivers and programs installed. After all the updates are installed, reboot the system.

Installing ROMs

To play some retro games, you first have to copy so called ROM files onto the system into the correct folders. A list of supported ROMS can be found on the ODROID GameStation Turbo for XU3/XU4 site where you find a table with the supported systems, as shown in Figure 10.



Supported Systems

CODE: SELECT ALL

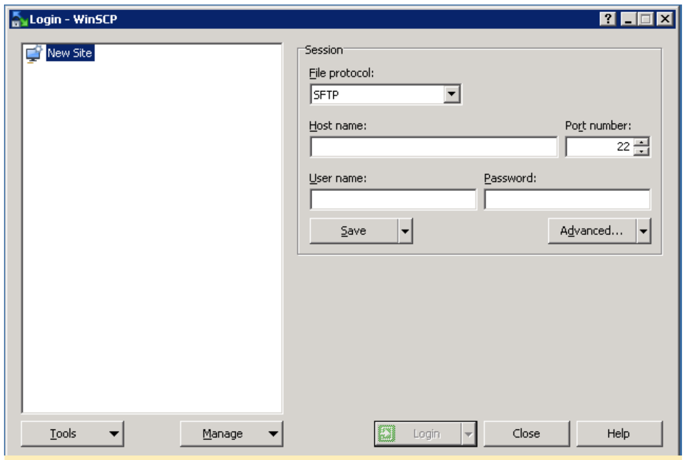
```
CPS2            fbalpha2012_cps2_libretro     /home/odroid/ROMS/CPS2           *.zip
CPS3            fbalpha2012_cps3_libretro     /home/odroid/ROMS/CPS3           *.zip
FB Alpha        fbalpha_libretro              /home/odroid/ROMS/FBA            *.zip,*.iso
Game and Watch  gw_libretro                   /home/odroid/ROMS/GW             *.mgw
GB              gambatte_libretro             /home/odroid/ROMS/GBx            *.gb
GBA             mgba_libretro                 /home/odroid/ROMS/GBA            *.gba
GBC             gambatte_libretro             /home/odroid/ROMS/GBx            *.gbc
LINUX           native                        /home/odroid/ROMS/LINUX          *.sh
MAME            mame_libretro                 /home/odroid/ROMS/MAME           *.zip,*.7z,*.chd
MSX             bluemsx_libretro              /home/odroid/ROMS/MSX            *.rom,*.mx1,*.mx2
N64             mupen64plus_libretro          /home/odroid/ROMS/N64            *.rom,*.v64,*.v64,*.n64,*.z64
NDS             desmume_libretro              /home/odroid/ROMS/NDS            *.nds
NeoGeo          fbalpha2012_neogeo_libretro   /home/odroid/ROMS/NeoGeo         *.zip
NeoGeoPocket    mednafen_ngp_libretro         /home/odroid/ROMS/NeoGeoPocket   *.zip,*.ngp,*.ngc
```

Figure 9 – List of supported ODROID GameStation Turbo ROMs

Let's focus on one of my favorite consoles, Gameboy Advance (GBA), and use this as an example. As you can see in the list above, ROMS for GBA need to be placed in the folder /home/odroid/ROMS/GBA and have to have a file extension of .gba. Other extensions such as .zip, .rar or .7z are not allowed at this point.

There are several different ways to get ROMs on your system. You can download them from a web browser, which is the easiest way. Download the ROM you want to use, then place the extracted file in the folder mentioned above.

If you already have a collection of ROMS on a PC and don't want to download them again, you can copy from your PC to your ODROID. In Windows, download and install a tool called WinSCP. It will allow you to connect from your Windows PC to your ODROID and copy files over to it. For this, start WinSCP on your Windows PC and create a new connection, as shown in Figure 10.



Login - WinSCP

New Site

Session

File protocol: SFTP

Host name:

Port number: 22

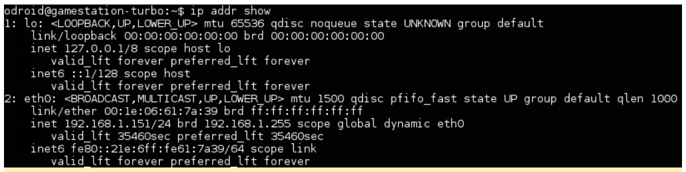
User name:

Password:

Figure 10 – Using WinSCP to connect to ODROID GameStation Turbo

For the Host name, use the IP address of your ODROID. To find the IP, go to your ODROID, open a terminal and type the following command, which will output information similar to Figure 11:

```
$ ip addr show
```



```
odroid@gamestation-turbo:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:1e:06:61:7a:39 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.151/24 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 35460sec preferred_lft 35460sec
    inet6 fe80::21e:6ff:fe61:7a39/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 11 – Output of the "ip addr show" command

If you're connected via LAN, the IP of your ODROID can be found under eth0:, or if you're connected via wireless network, the IP will be under wlan0:, below the label inet. In the above example the IP would be: 192.168.1.151. The user name and password should both be "odroid" (without quotes), if you haven't changed it yet. You can now navigate on your ODROID to /home/odroid/ROMS/GBA/ and copy your games from your collection on your PC over to the ODROID.

For Linux users, I assume you already know how to copy files from one PC to another. Tools like "scp" or "mc" should be common to you and you should know how to copy files. You can alternative use Filezilla, a file manager available for Windows, Linux and Mac which can be used like above WinSCP on all systems.

Once you've copied the ROMS over to the ODROID you're ready to go! On the OS screen, open up EmulationStation. Go to Applications -> Games -> EmulationStation and your games should already be listed.

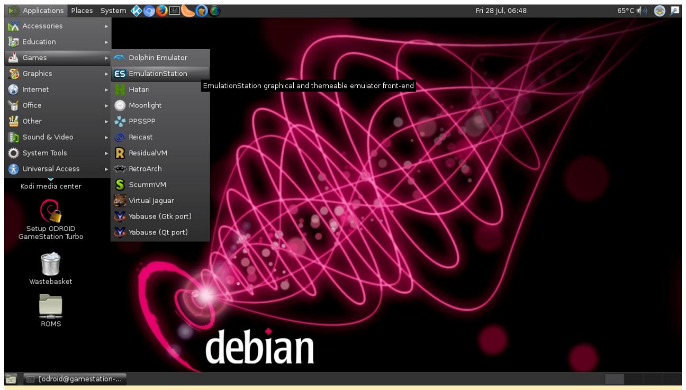


Figure 12 – Starting EmulationStation from the ODROID GameStation Turbo desktop

When you start EmulationStation for the first time, you will be asked to setup your controller. Please note that this does not affect the emulator and games you want to run, but is only used to navigate inside EmulationStation. You will need the directions (up, down, left, right), the select and start button (for menus), and A and B (select and go back). Every other key that EmulationStation asks you to configure is not used, and should be skipped by pressing and holding one button/key until everything that is not needed is skipped (not defined).

ODROID GameSir G3w USB Controller Joystick

June 1, 2018 By Justin Lee Android, Gaming, Tutorial

The GameSir G3w is a high-quality gamepad that adopts a 32-bit MCU chip, with a computing capability that is up to 48 million operations per second. It allows the device to be highly sensitive and accurate with respect to overall performance. The floating D-pad makes it easier to discern between the eight compass points. The analog joysticks have 360-degree positioning of ultra-accuracy.

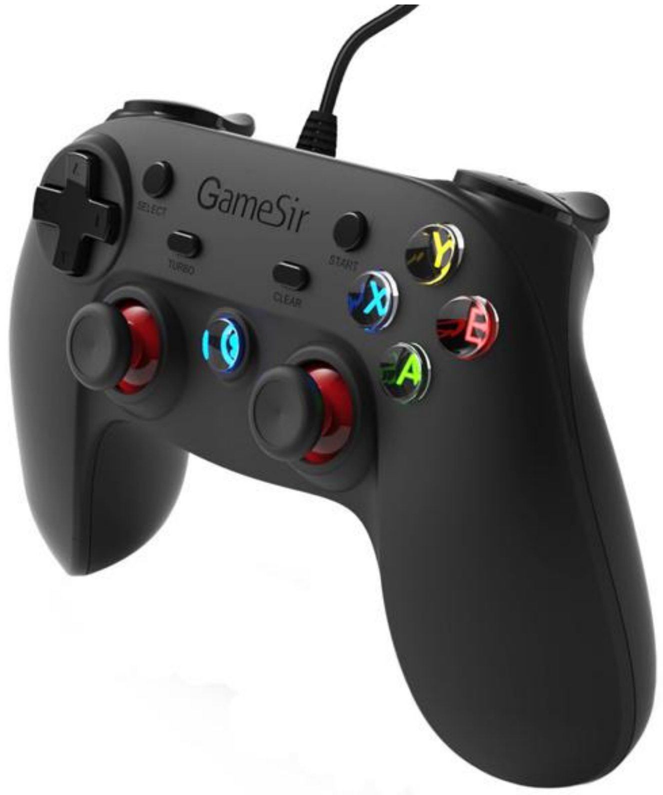
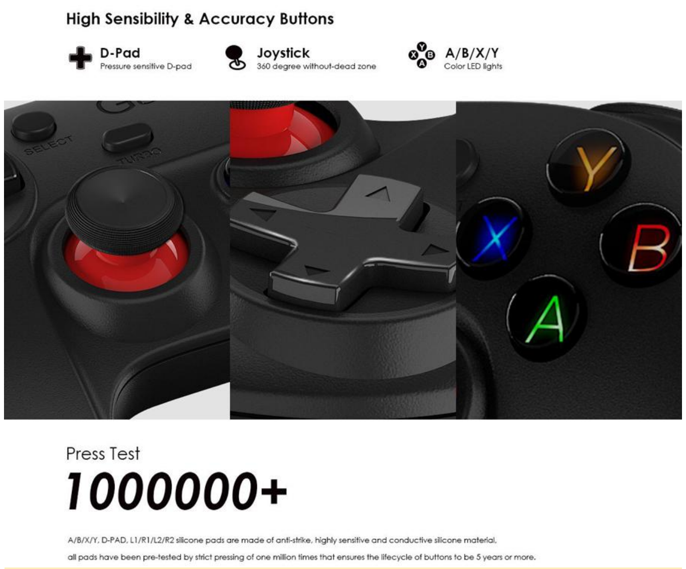


Figure 1 – The GameSir gamepad is Hardkernel's latest high-quality gaming accessory

It also boasts well designed ergonomic function buttons, is solid and responsive, with a firm action, and can be pressed with no effort. The pressure sensitivity of R2/L2 allows players to perceive the press depth accurately, making it much easier to play car racing and drifting games. The USB cable length is 2 meters (6.6 ft) for comfortable usage.



A/B/X/Y, D-PAD, L1/R1/L2/R2 silicone pads are made of anti-strike, highly sensitive and conductive silicone material.
all pads have been pre-tested by strict pressing of one million times that ensures the lifecycle of buttons to be 5 years or more.

Figure 2

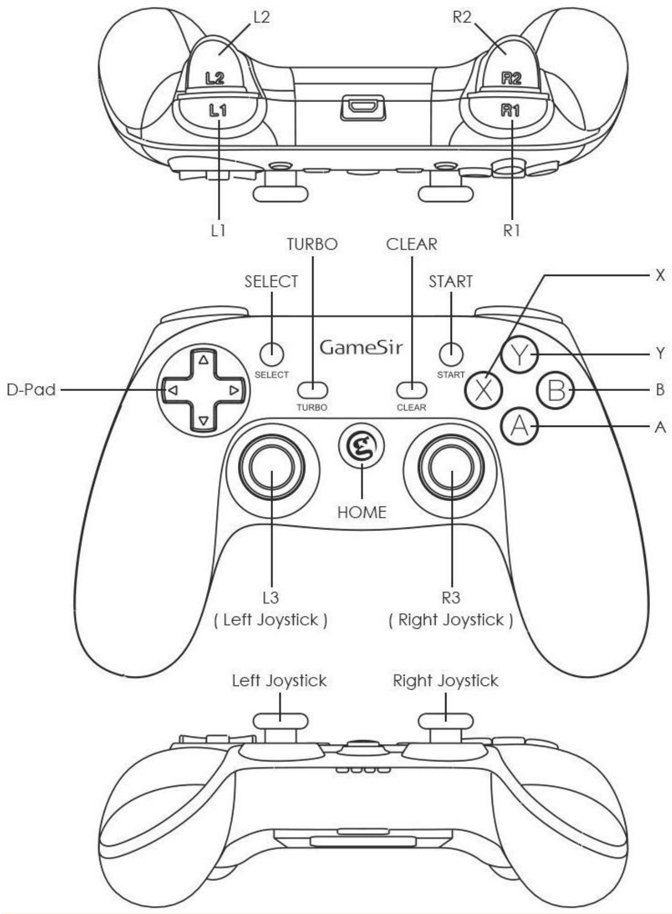


Figure 3

Figure 4

This gamepad works with Hardkernel's very own OGST console kit offering (https://goo.gl/5fSR48). It is supported on the official Linux and Android operating systems offered by Hardkernel.

Configuration

Show below is a reference udev configuration file for Libretro (/usr/share/libretro/autoconfig/udev/xiaoji_GameSir_G3w.cfg)

```
input_driver = "udev"
input_device = "Gamesir-G3w"
input_device_display_name = "xiaoji GameSir
G3w"
# Hex vid:pid is found using "dmesg -w" or
"tail -f /var/log/syslog"
# and converted to Decimal using
# http://www.binaryhexconverter.com/hex-to-
decimal-converter
# Hex vid:pid = 20BC:5500 -> Decimal vid:pid =
8380:21760
input_vendor_id = "8380"
input_product_id = "21760"
input_b_btn = "0"
input_y_btn = "3"
input_select_btn = "10"
input_start_btn = "11"
input_up_btn = "h0up"
input_down_btn = "h0down"
input_left_btn = "h0left"
input_right_btn = "h0right"
input_a_btn = "1"
input_x_btn = "4"
input_l_btn = "6"
input_r_btn = "7"
```

```
input_l2_btn = "8"
input_r2_btn = "9"
input_l3_btn = "13"
input_r3_btn = "14"
input_l_x_plus_axis = "+0"
input_l_x_minus_axis = "-0"
input_l_y_plus_axis = "+1"
input_l_y_minus_axis = "-1"
input_r_x_plus_axis = "+2"
input_r_x_minus_axis = "-2"
input_r_y_plus_axis = "+3"
input_r_y_minus_axis = "-3"
input_b_btn_label = "A"
input_y_btn_label = "X"
input_select_btn_label = "Select"
input_start_btn_label = "Start"
input_up_btn_label = "D-Pad Up"
input_down_btn_label = "D-Pad Down"
input_left_btn_label = "D-Pad Left"
input_right_btn_label = "D-Pad Right"
input_a_btn_label = "B"
input_x_btn_label = "Y"
```

Solar Powered Microserver

June 1, 2018 By Roberto Rosario Tinkering, Tutorial

As you have probably read already, less than a year ago Puerto Rico experienced one of the worst natural disasters in recorded history (https://www.mayan-edms.com/post/hurricane-maria). The situation after hurricane Maria has been compounded by an already deficient energy infrastructure. Frequent blackouts are now a common occurrence of daily life.

Blackouts are not just annoying, but cause a series of problems impacting almost all aspects of modern life. Being a technology worker, blackouts directly impact my bottom line and my primary source of income.

After months in the dark, I took the plunge an built a small solar power system. To make use of solar energy it is very common to use a voltage inverter to boost the 12 volts DC produced by the solar panels and stored by the batteries to 120 volts AC which is what most electronic devices use. Using an inverter introduces some conversion losses in the system. In order to minimize these losses, I started converting as many devices as I could to work with 12 volts DC. This way they could be operated directly from the solar system batteries. It was time to start converting my tools of work too.

One of the systems I need for my daily work are computers and servers. I have some hosted on the cloud and others locally. Constant power outages mean a lot of the charge in my solar system batteries is being wasted keeping the servers and computers working. There had to be a better way to keep the equipment I needed running and optimize their electricity use.

Instead of a few larger computers, I decided to start scaling down the devices I used. More devices meant I could spread the services among them.

Solar Powered server assembled

Single board computers or SBCs are all very common now with a big variety to choose from. These SBCs are usually powered by USB so they already work with DC current. One my favorite single board computers is the ODROID C2. The ODROID C2 is produced by a responsible company, it's well documented and is very well supported. The ODROID C2 has a 64-bit quad-core ARM CPU, 2 GB of RAM, and support eMMC for storage. It is one of the most powerful SBCs in the market in its price range. With this in mind, I set out to convert some of my existing ODROID and SBCs to replace the common servers and computers I normally use.

Figure 2 – The common plastic and 3D printed enclosures for these SBCs are not enough to protect them and do not leave any space for additional components. Aluminum enclosures are a much better choice

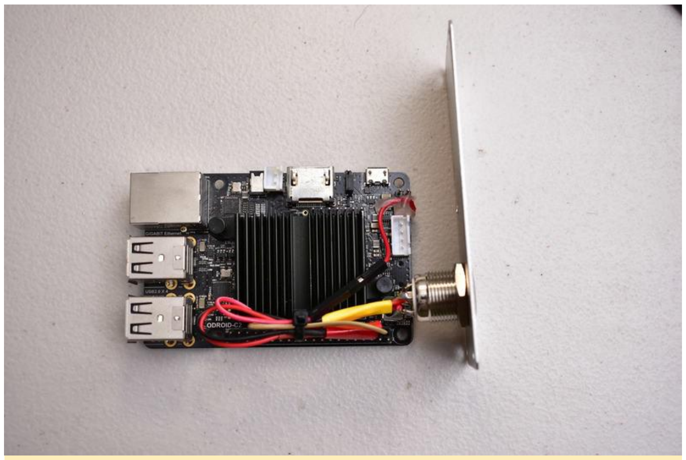


Figure 3 – This is the ODROID C2 with power indicator and button installed. The switch is not connected to anything and left for future use (maybe via GPIO). The power indicator is soldered to heartbeat LED on the board

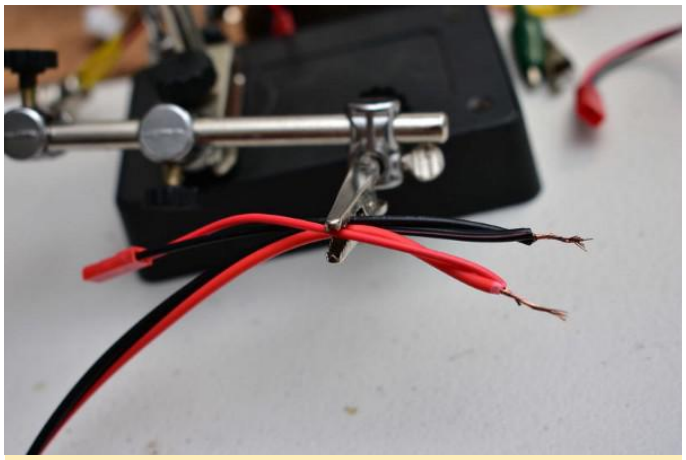


Figure 4 – The power cable

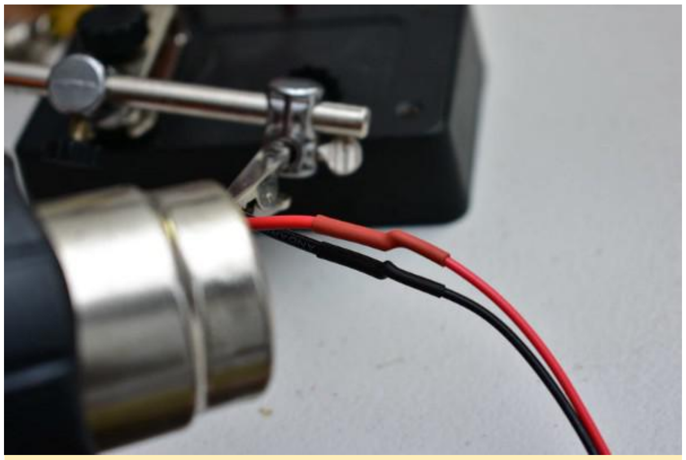


Figure 5 – The JST plug goes to the voltage converter inside the aluminum box. All cable bonds are protected with heat-shrink tubing

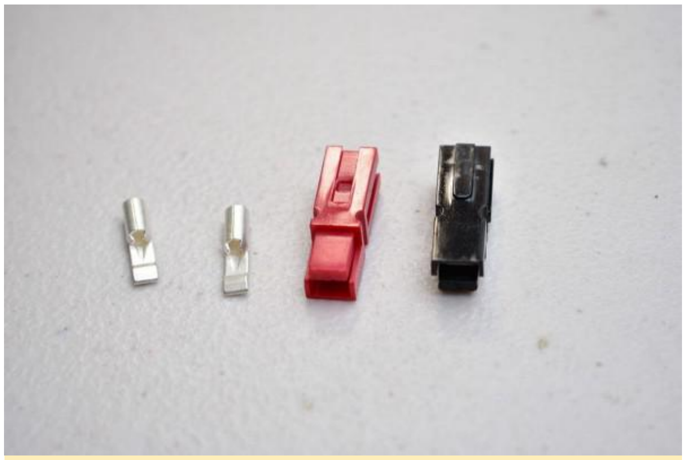


Figure 6 – The Anderson connector goes on the outside of the cable

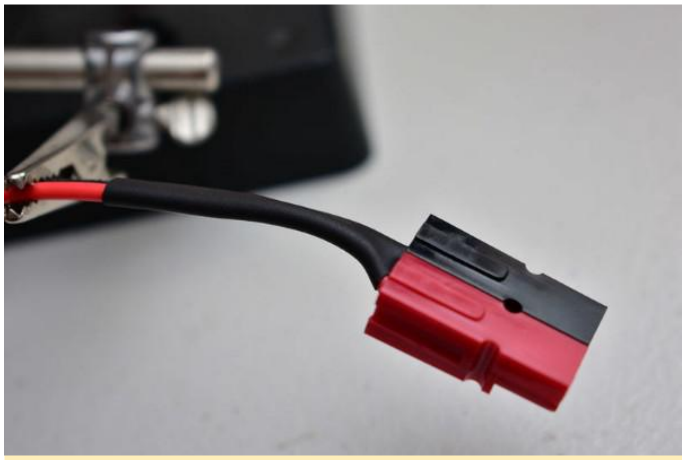


Figure 9 – I added some heat-shrink tubing to work as cable strain support

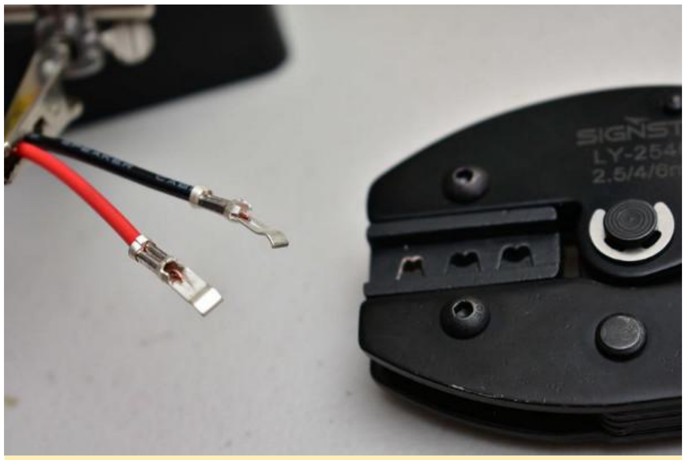


Figure 7 – I crimped then and added a bit of solder for best conductivity

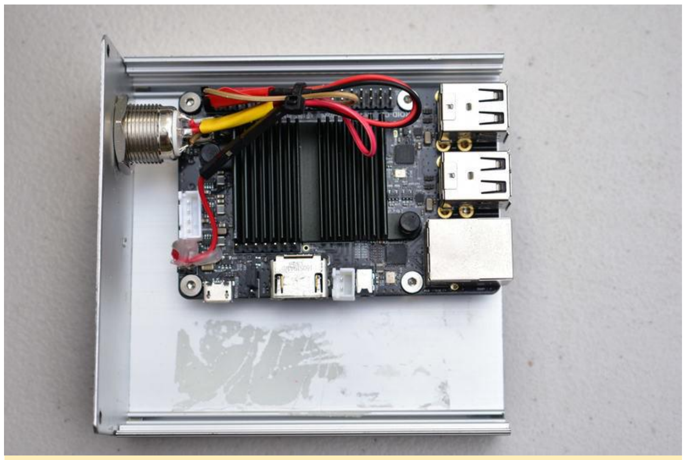


Figure 10 – The ODROID mounted using brass standoffs. The residue is from double-sided tape from a previous attempt

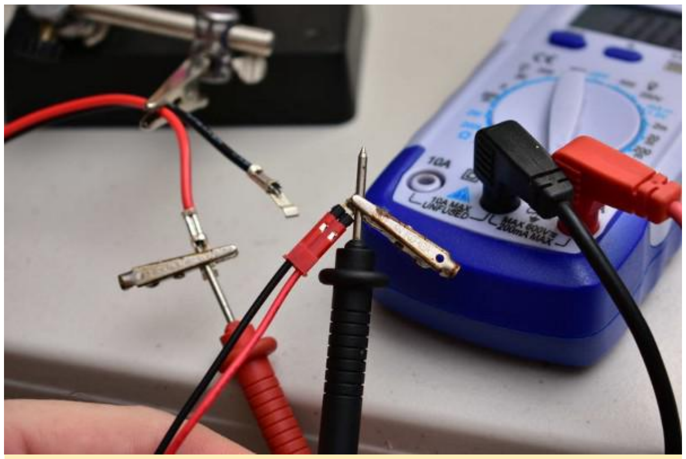


Figure 8 – Here I am testing for continuity

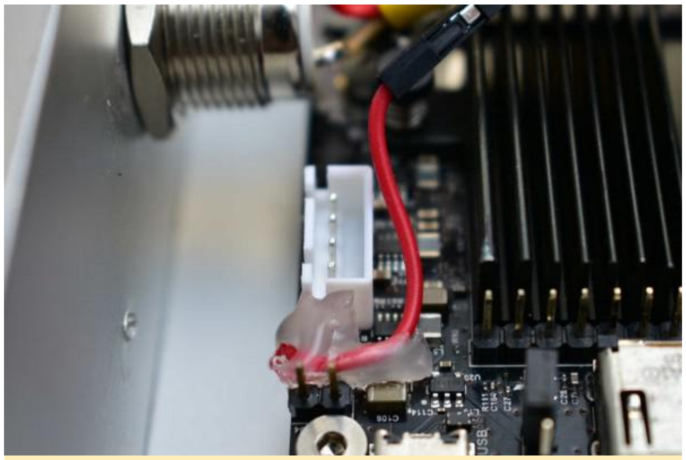


Figure 11 – The serial console and power plugs were replaced as shown

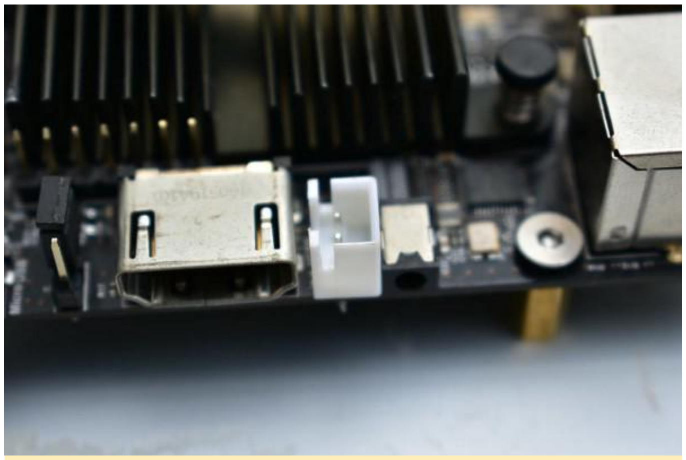


Figure 12 – The serial console and power plugs were replaced as shown

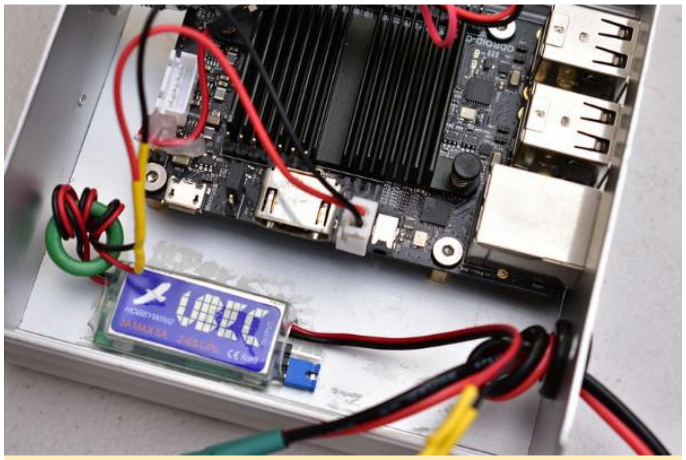


Figure 13 – Here is the Voltage converter mounted with double-sided tape. The converter takes the 12 volts from the solar system and reduces it to 5 volts

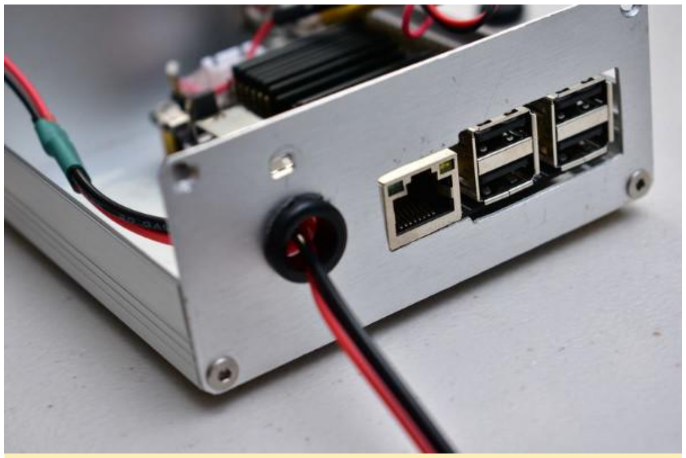


Figure 14 – This particular converter can supply 3 amps continuously

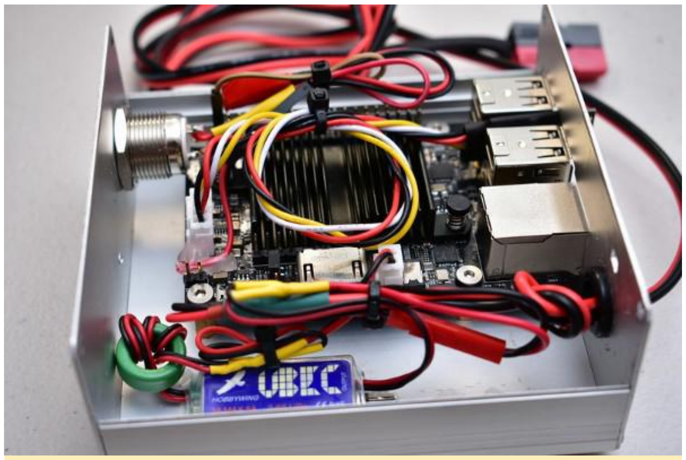


Figure 15 – Everything mounted and all cables tied properly

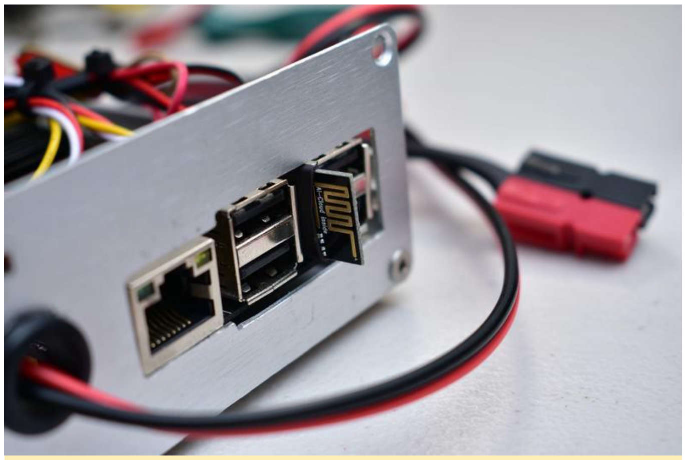


Figure 16 – The ESP8266 (ESP-01) programmed to work as a WiFi to serial console bridge. This will allow access to the console of the ODROID even if communications fail. This is perfect for debugging and management

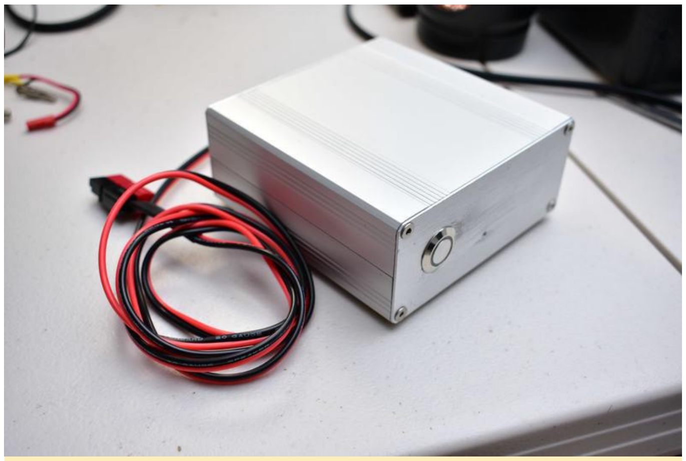


Figure 17 – Here is the box fully assembled. The holes for the USB, network connector and the power button were cut by hand and there was some scuffing. This can be avoided using painters tape or with better tools like a CNC or a bench drill

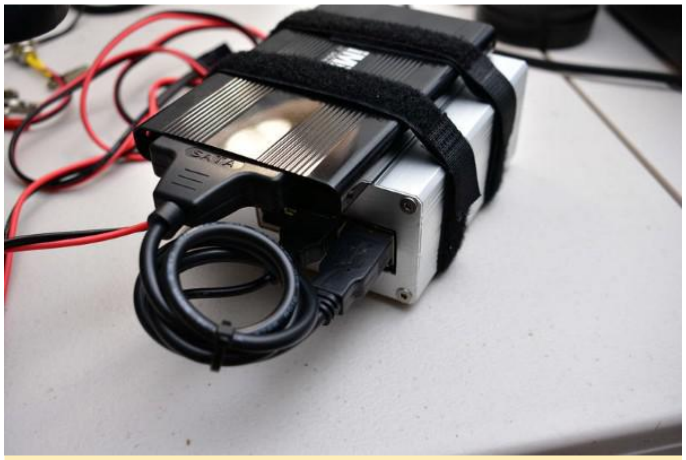


Figure 18 – The ODROID-C2 supports eMMC storage and this one has a 16 GB card which is used for system files. For storage, an external SSD SATA hard drive is used with a SATA to USB adapter

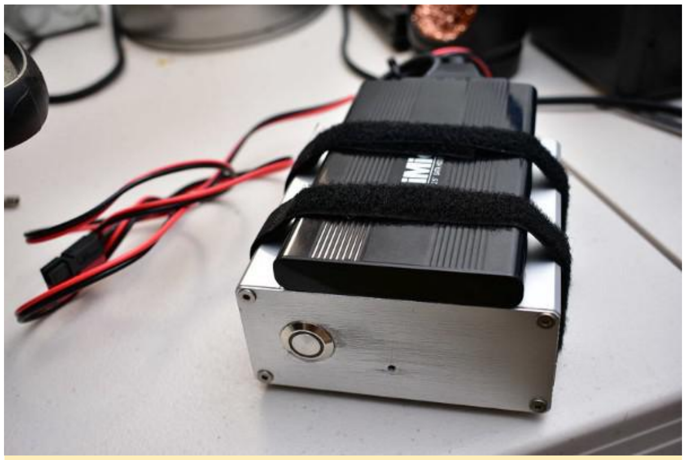


Figure 19 – The ODROID's USB ports are USB 2. This means that the access to the storage will not the ultra fast but it is adequate for the work this box will perform

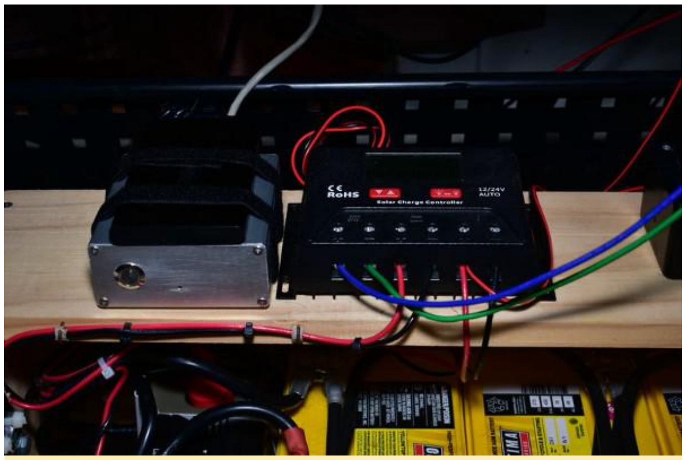
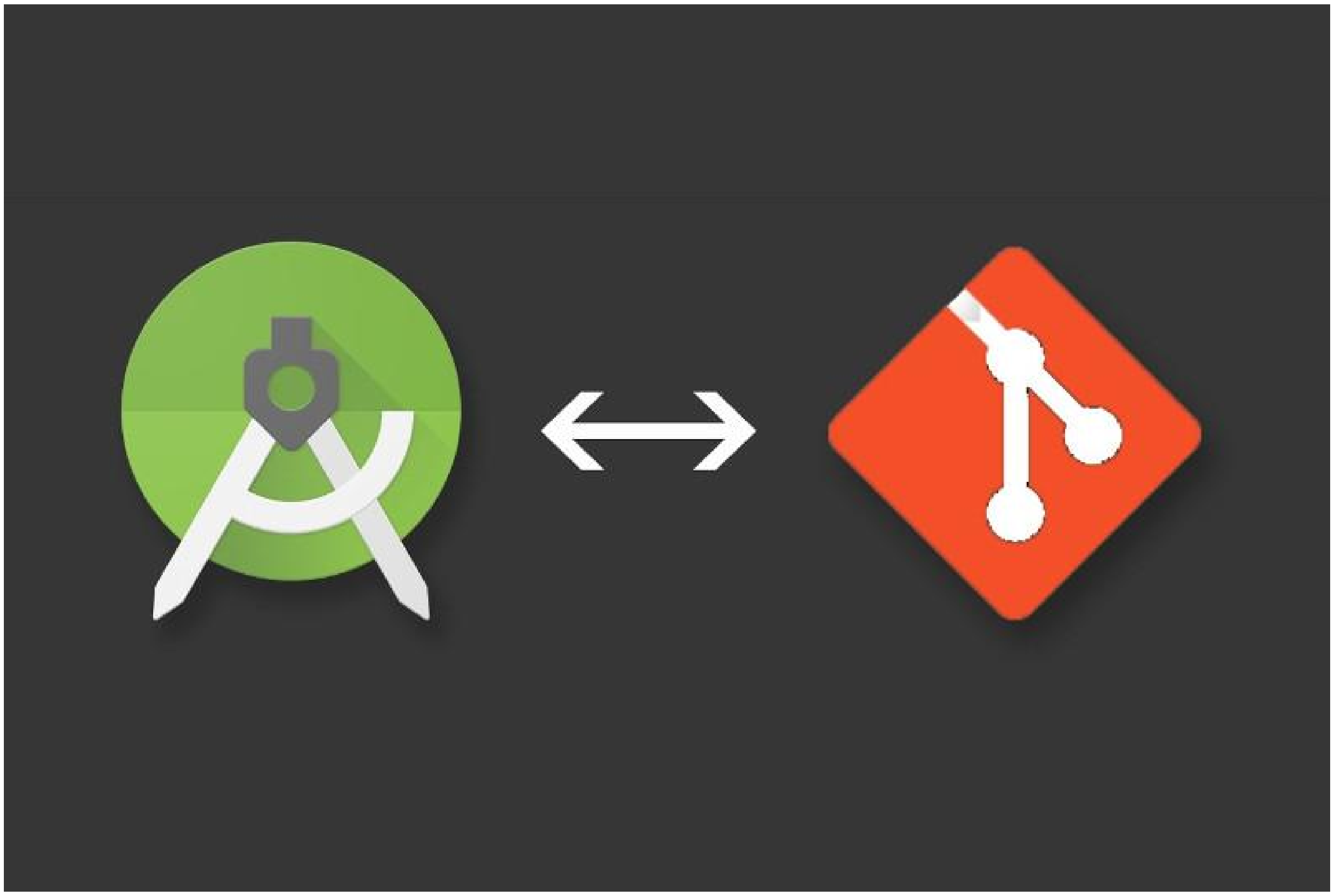


Figure 20 – Here is the microserver placed next to the solar system charge controller and connected to the Anderson power distribution box

For comments, questions, and suggestions, please visit the original article at https://medium.com/@siloraptor/solar-powered-microservers-for-a-post-hurricane-maria-puerto-rico-ca83027d20ac.

Android Development: Using GitHub

June 1, 2018 By Randy Hall Android, Tutorial



Welcome back, appdev initiates! If you're like me, you're more than ready to increase your app development skills. As mentioned previously, the 800-pound gorilla of online open source projects is GitHub. There are several Git-based choices in the marketplace, but for our purposes, we will use GitHub for this column, with the upside being that all of the projects, and mistakes, will be available to the community to view, give comments, and improve.

The good news for Android Studio is that it treats GitHub as a first-class source code repository host, that is to say, you can make other hosted Git providers work, but it's just not quite as slick. This only means that if you choose to go your own way with something other than GitHub as your source control tool of choice, then we'll leave it as an exercise for you, dear reader, to complete and let us know how it goes in the ODROID forums. With that, let's connect the IDE with our hosted source control provider.

Get on GitHub

If you look back at the code in our last installment, it was very basic. It took Android Studio only a few moments to construct it, and then we had a project folder complete with build scripts, app code, and layout files. The downside of all this automation in integrated development environment tools is that it creates a lot of files, and keeping track of them (and the changes that get made to them) can be a big headache. This is where hosted sites like GitHub really help, because they allow you to keep a backup of all of your precious code files, ready to be recalled in case of disk crashes, equipment loss, or any number of other pitfalls. So, starting with the project we made last time loaded into Android Studio, we can head up to the "VCS" menu, select "Import into Version Control" and select "Share Project on GitHub." If you haven't already gone through the process of creating a GitHub account, which was mentioned last time, now's your chance.

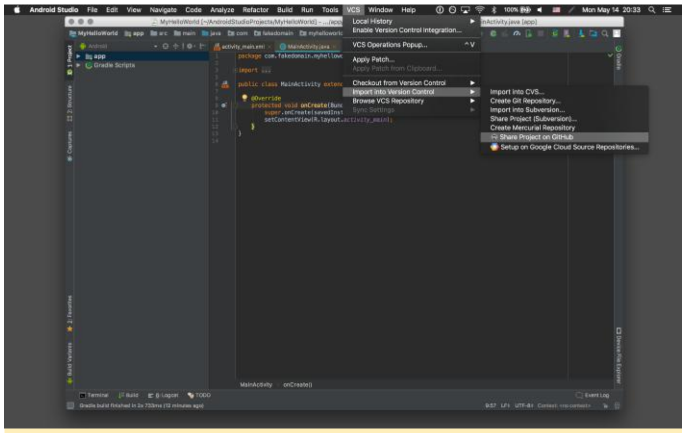
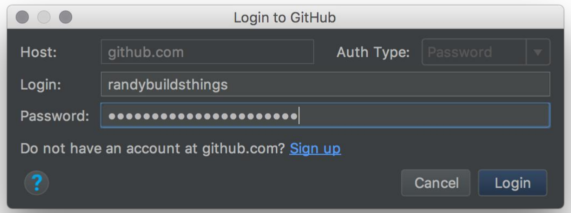

VCS
Local History
Enable Version Control Integration...
VCS Operations Popup...
Apply Patch...
Apply Patch from Clipboard...
Checkout from Version Control
Import into Version Control
Browse VCS Repository
Sync Settings
Import into CVS...
Create Git Repository...
Import into Subversion...
Share Project (Subversion)...
Create Mercurial Repository
Share Project on GitHub
Setup on Google Cloud Source Repositories...

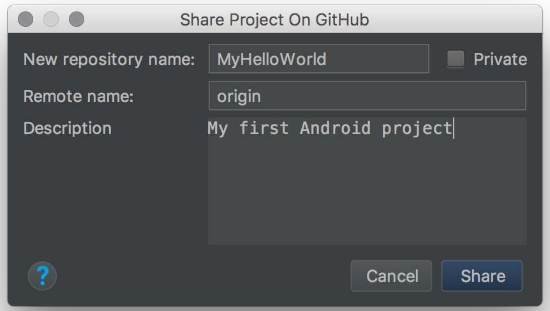
figure 01 – Set version control to GitHub

Once you have the account confirmed and set up, you can come back to Android Studio and inside the "Login to GitHub" dialog, select "Password" for Auth Type and enter your username and password. You will then be shown the "Share Project on GitHub" dialog, which lets you change the repository, or "repo", name, remote name and description. For now, leave the repo name and remote name alone, and you can put in a simple description for the project. I chose something simple, like "My first Android project". You can then confirm which files will be initially sent to GitHub, it's fine to accept the default set, and click "OK". In a few short moments, you'll be greeted with the notification that your project is now on GitHub. You can click the link in the notification and check out your online source code.

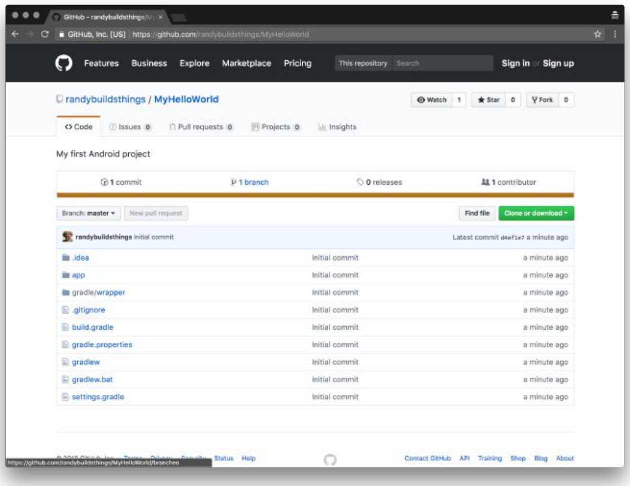

Login to GitHub

Host: github.com
Auth Type: Password
Login: randybuildsthings
Password: ••••••••••••••••••••••
Do not have an account at github.com? Sign up

figure 02 – Login to GitHub


Share Project On GitHub

New repository name: MyHelloWorld
Private
Remote name: origin
Description: My first Android project


randybuildsthings / MyHelloWorld

My first Android project

1 commit · 1 branch · 0 releases · 1 contributor

Branch: master

randybuildsthings Initial commit — Latest commit a minute ago

.idea	Initial commit	a minute ago
app	Initial commit	a minute ago
gradle/wrapper	Initial commit	a minute ago
.gitignore	Initial commit	a minute ago
build.gradle	Initial commit	a minute ago
gradle.properties	Initial commit	a minute ago
gradlew	Initial commit	a minute ago
gradlew.bat	Initial commit	a minute ago
settings.gradle	Initial commit	a minute ago

figure 04 – GitHub project page

The Most Basic App Ever Made™

With all the fuss about getting our code into version control, we really haven't spent much time looking at what Android Studio generated for us. Let's do that now. In creating this app, Android Studio generates a lot of files, but only a few of them are really vital to figure out what's going on. These are the files we care about right now:

- app/manifests/AndroidManifest.xml
- java/com/fakedomain/myhelloworld/MainActivity.java
- res/layout/activity_main.xml
- res/values/strings.xml

AndroidManifest.xml

This is useful from the point of view that it actually tells the Android device what code will be run when the app is first launched. If you double-click on the file and view it in the IDE, it's not much to look at, but everything in there is important to properly run your app. We will devote time in another installment to really delve what the manifest does for your app, but for today it is enough to know that Android Studio will take care of putting new activities and other app features into the file for you, at least until you're more comfortable doing it yourself.

MainActivity.java

The file is only a few lines of code, but it directs the reader to one important resource that you will definitely spend time with: the activity_main.xml layout file. This is where the "look" of the app is defined, and where it can be changed using Android Studio's visual layout tools. If you double-click on the activity_main.xml file, the IDE should present you with its visual layout editor. If you see a layout with an XML file pane next to it, you may have chosen the "Text" view of the layout. If you click the "Design" tab near the bottom of the layout pane, you will be switched back. While the layout is sparse, there's still a lot going on here. In fact, if you look closely, our "Hello World" text is already in place in the middle of the view. That's the thing about the Empty Activity: it's not entirely empty, but has a simple TextView element in the layout to get you started. And what better sample text to put into a text widget than our favorite two word starter phrase?

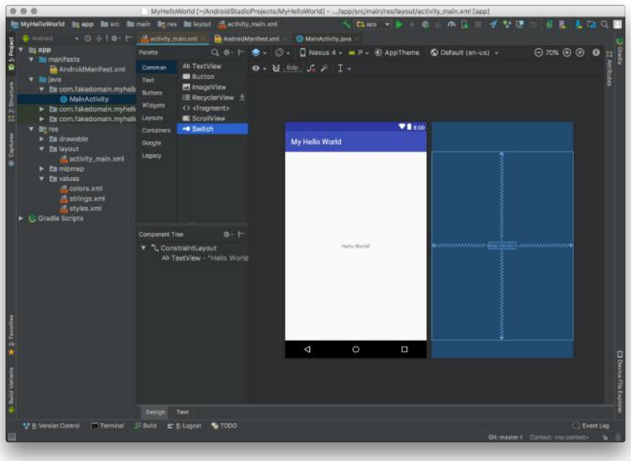
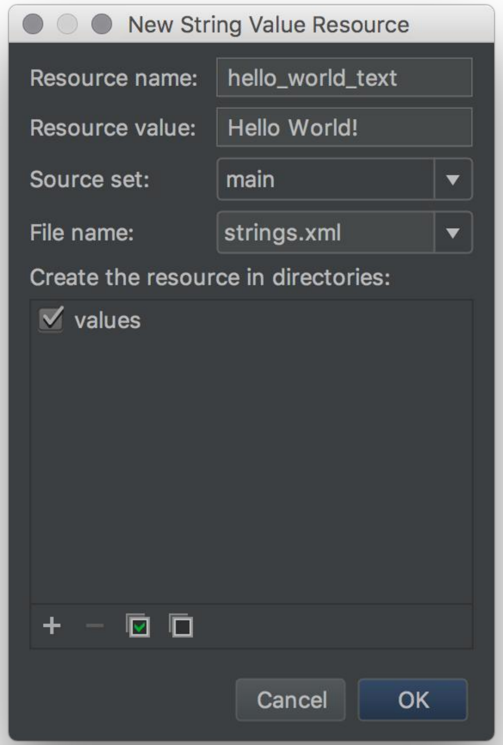


figure 05 – Design view

strings.xml

Strings.xml is the last file to keep an eye on. The reason I mention this is that, as an application developer, good practices are essential, and the earlier you adopt them, the better. For throwaway one-off projects, it's fine to go around typing in fixed text values for different buttons, text views, and navigation controls, but in the real world, there are many languages, and we'd be truly short-sighted, if not arrogant, to assume that English is the only language your app should use. So, let's tweak a setting to make use of this file. If you double-click on the "Hello World" text in the layout, you will see the Attributes pane appear. As it turns out the TextView text attribute will most likely be highlighted, since you double-clicked it in the layout editor. To the right of that attribute box, you should see what looks like an ellipsis (...), which you can click now to display the resource window for this project. Inside are the resources that you can make use of for the attribute you're editing.



New String Value Resource

Resource name: hello_world_text
Resource value: Hello World!
Source set: main
File name: strings.xml
Create the resource in directories:
✓ values

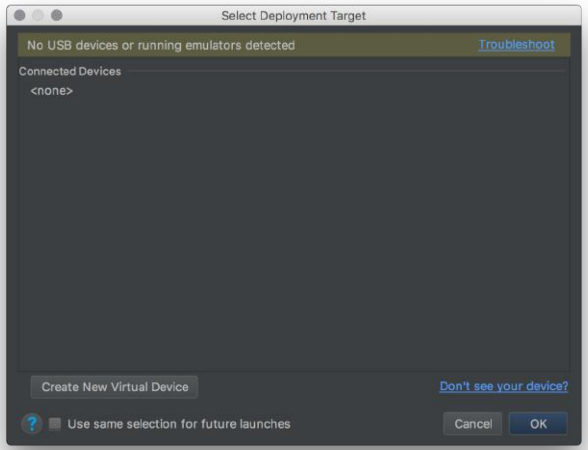
Figure 06 – String Value Resource Editor

Since we want to make a change, click on the "Add new resource" menu item in the top right of the window and then select "New string Value...", which presents you with the "New String Value Resource" dialog. For "Resource name", choose something sensible like "hello_world_text", and then set the

Resource Value to “Hello World!” You can then click OK and the value in the text attribute will have changed to “@strings/hello_world_text” but the layout will still display the value, which is “Hello World!” If you now look at strings.xml in the code editor, you’ll see that there is a new line in the file, which Android Studio put there for you. Brilliant!

Let’s run it!

Now we need to build this and see if it will run. At the top of the IDE window, the center of the toolbar has a green hammer icon. Click it, and the project will be built. It might take a few moments depending on the speed of your development machine, but it will ultimately get done. As an aside, as much as I love ODROID devices for running Android apps and Linux in general, and even the powerful ODROID-XU4 is not particularly well-suited for app development of this nature. It’s just that there’s a lot of memory involved in building software, and building a nontrivial app would ultimately lead to thrashing your eMMC or SD card media when Android Studio runs low on memory, shortening the life of your flash media at the very least. But hey, it’s your money, and if ODROID is what you have to build with, it can do the job, you might just have to wait longer for certain tasks. Now that the app is built, if you press the green “Play” triangle, the app will “run” and try to connect to an Android device. What’s that, you say? Don’t have an Android device handy? You haven’t gone to the ODROID Wiki to connect your ODROID-C2 using it’s USB OTG (https://wiki.odroid.com/getting_start/adb_fastboot) port? You haven’t started Network ADB on your ODROID-XU4 (https://wiki.odroid.com/odroid-xu4/troubleshooting/adb_over_tcpip)? While it’s always more valuable to test an app on actual hardware, if all else fails, you can always emulate an Android device.


Select Deployment Target

No USB devices or running emulators detected

Connected Devices

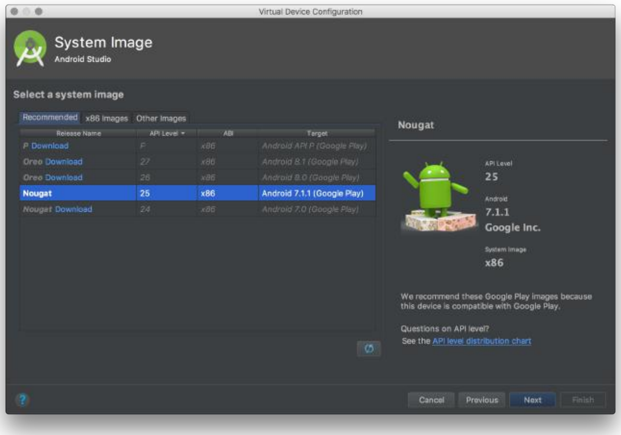
<none>

Troubleshoot
Create New Virtual Device
Don't see your device?
Use same selection for future launches
Cancel
OK

Figure 07 – Select Deployment Target

Get Virtual (Devices)

So, if you’re seeing the “Select Deployment Target” and it is empty, you can click on “Create New Virtual Device” and be treated to a catalog of various Android benchmark devices that you can emulate on your computer, from TV to tablet, to phone to Android Wear devices. This time around, let’s create a middle-of-the-road phone device, like a Nexus 5X. Once you select it, click “Next” and go ahead and choose one of the “recommended” system images, which as of this writing would be Nougat, Oreo or the “P” Preview. You’ll probably have to click the “download” link next to the version to download the image prior to proceeding, and yes, it might take a while.


Virtual Device Configuration

System Image
Android Studio

Select a system image

Release Name	API Level	ABI	Target
P Download	P	x86	Android API P (Google Play)
Oreo Download	27	x86	Android 8.1 (Google Play)
Oreo Download	26	x86	Android 8.0 (Google Play)
Nougat	25	x86	Android 7.1.1 (Google Play)
Nougat Download	24	x86	Android 7.0 (Google Play)

Nougat

API Level
25

Android
7.1.1
Google Inc.

System Image
x86

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?
See the API level distribution chart

figure 08 – Virtual Device Configuration Editor

Once you have the image downloaded, click “Next”, give the virtual device a name (or keep the default that Android Studio generates), change the screen orientation to landscape if you’re into horizontal screens, and click "Finish". At last, you have a device to launch your app!

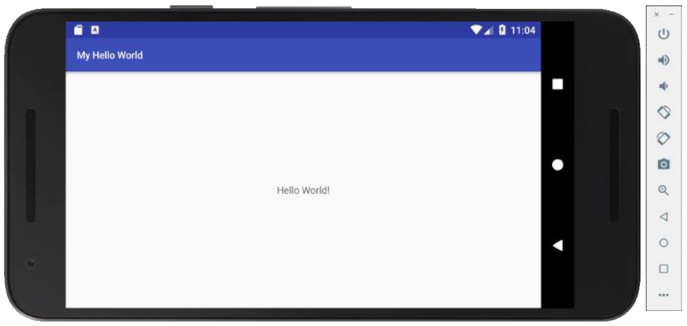


figure 09 – Virtual Device Running Our App

If all goes according to plan, the virtual device will boot up the Android version you downloaded, and will place you at the launch screen. A few moments after that, Android Studio will connect to the virtual device using ADB and launch the app. It might not be as satisfying as having your app running on an actual Android device, but then again, you'll never have the excuse that you don't have an Android device when you need one.

Commit those changes

All that coding is hard work, and at the end of every coding session, you should always make it part of your practice to commit your work in progress. I'm the first to say that when it's you working on a code repository by yourself, nobody is going to notice that your code doesn't build correctly, or has a ton of bugs and errors in it. I invite you to get over it. The important thing is that you keep a backup in the event that something terrible happens and you're suddenly without your workstation and its file storage. To do this, you can go to the VCS menu again, and select "Commit", there is a keyboard shortcut for this, and also an entire pane of the IDE window that is for version control. Make sure to take a look at it and get familiar. You will then see that two files at least have been changed: the activity_main.xml and strings.xml files. You can take a look at the changes to make sure they are what you're expecting to check in. Now you should enter a commit message, which will log a narrative of what changes are in the code that you're committing to the repo. I will enter something terse, like "changed text to use strings.xml". Hover your mouse pointer over the "Commit" button in the bottom right corner, and you'll notice that there are some additional options. For our GitHub demo, let's go ahead and select "Commit and Push…" which will present you with a dialog that lets you add tags to the push to GitHub. Click "Push" and you're done! I can't emphasize this enough: make sure you check in your code changes at the end of your work day, and you'll never regret going to sleep (and you might even sleep better, but don't hold me to that!)

What's next?

Since this is a new column, I thought making a Hello World app would be a fun start. But honestly, it really is the Most Basic App Ever Made™. I'm now actively looking at options for working on an app that makes sense to deploy to an ODROID-XU4 or ODROID-C1+/C2 for fun and possibly good learning. Some ideas that come easily to mind include:

- An Android TV-based game, running on an ODROID-XU4 with LineageOS Android TV loaded, that uses multiple players' own Android mobile or tablet devices to control and play a turn-based card or tile game.
- A custom Android home automation touchscreen, running on an ODROID-XU4 or ODROID-C2 with LineageOS, that uses background services to communicate with other home automation servers and devices to control and monitor your home without using cloud-based services.
- An Android Auto, running on an ODROID-C2 with a touchscreen, that collects information from your car and allows your Android mobile or tablet to connect and play media, navigate, communicate and inform the time you spend behind the wheel.

If the list seems daunting or you feel yourself recoiling in horror that this is all too complicated, just relax. I'm here to help, and I will do my best to explain everything as plainly as possible. I recognize that every one of those projects above is very much a combination of hardware elements and software elements working together to provide a solution. So you can expect that there will be editions of this column that will be hardware-focused in order to set up or at least mock up a hardware platform that we can then use to write apps for. There will also be editions of this column where we will break down bigger problems into chunks to which we can then apply Android-based app architecture practices in order to make them work the way Android device users expect them to. I think it will be a much more rewarding experience for everyone, and it will all be made open source and available online for future Android developers to build on. If you think these ideas are terrible and you are convinced you have a better one, I want to hear about it! You can always comment on the interactive version of ODROID Magazine for this article, or visit the ODROID Magazine forum at https://forum.odroid.com. The best part about this column is that the code I work on is online for you to check out as well. Feel free to visit my GitHub account at https://GitHub.com/randybuildsthings and laugh at many of the goofy things I do on there. I use GitHub mostly for playing around, but I look forward to seeing you there.

Linux Gaming on ODROID: Saturn Games – Part 4

June 1, 2018 By Tobias Schaaf Gaming, Linux, ODROID-XU4

We are back again with the ODROID-XU4/XU4 and Sega Saturn games. This time I want to look into games starting with the letter "S" like in Saturn, or as in "shmup". There are so many great games for the Sega Saturn that start with the letter "S" that I enjoy playing, that I had to make this into its own section of this series.

Samurai Spirits IV – Amakusa Kourin

Samurai Spirits IV is a very nice "beat em up" game similar to King of Fighter, or Street Fighter, but offers the addition of weapons similar to Last Blade.

Figure 01 – Title screen of Samurai Spirits IV for the Sega Saturn

This game requires a 1MB memory expansion to work properly, so make sure you have the correct cartridge selected. I really enjoy this game. It's one of the best 2D fighting games I have played. The blinking health bars and shadows are the only elements I don't like about this game, but the rest is just perfect.

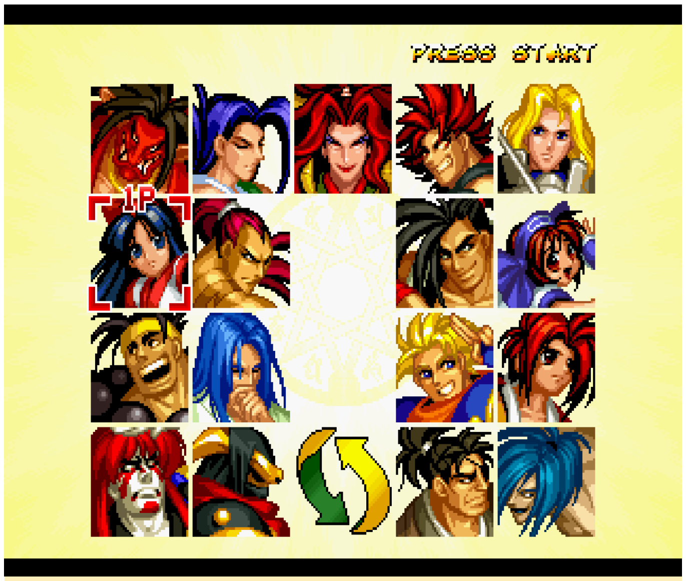


Figure 02 – There's a good amount of fighters to choose from in this game

The game is very interesting, since the different characters are not just different in graphics but also in fighting styles. Moves are easy to pull off, and even button smashing can work fine for you if you figure out what moves you need to combine.

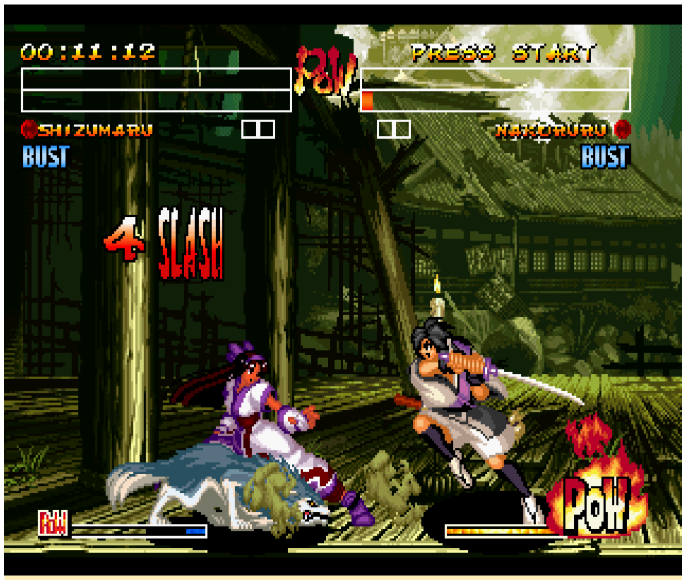


Figure 03 – The fighting animations are really good and the graphics and background really fit the setting

You can even disarm your enemy and weaken them until they pick up their weapons again. The music is nice, there is a lot of animation going on both for the characters as well as the background. I'm not a too big fan of fighting games, but I spends quite a few hours playing this one and will continue playing it too.

Saturn Bomberman

I don't think Bomberman needs to be introduced, but some say the Sega Saturn version of Bomberman is the best version in existence, and that each version after this version was compared with the Sega Saturn version.

Figure 04 – Saturn Bomberman

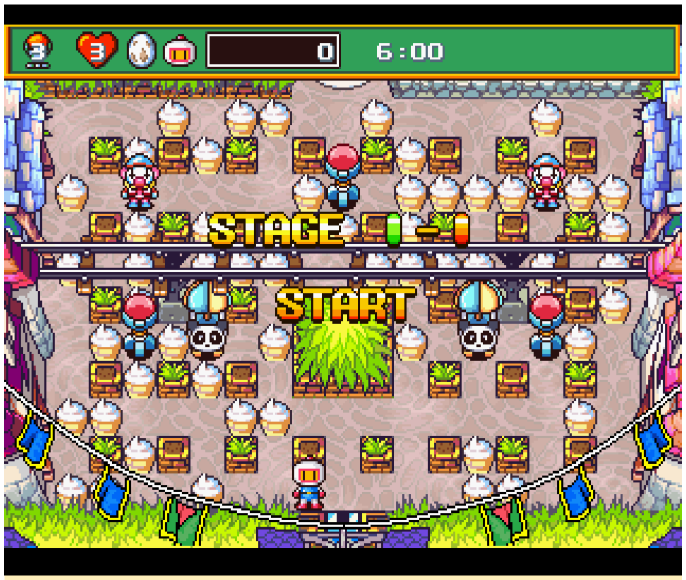


Figure 05 – Cute Animations, bright color, very good gameplay it won't get any better than that when it comes to Bomberman

Saturn Bomberman is really a gem, with nice anime cut-scenes, wonderful bright colors, all kinds of power-ups and an all so familiar gameplay. If you like Bomberman, you should definitely play the Sega Saturn version! Oh and did I mention it supports up to 10 players on one screen?

Sexy Parodius

This shooter is similar to Parodius which I showcased last time, and is a Gradius-Parody-Series kinda "Cute 'em up".

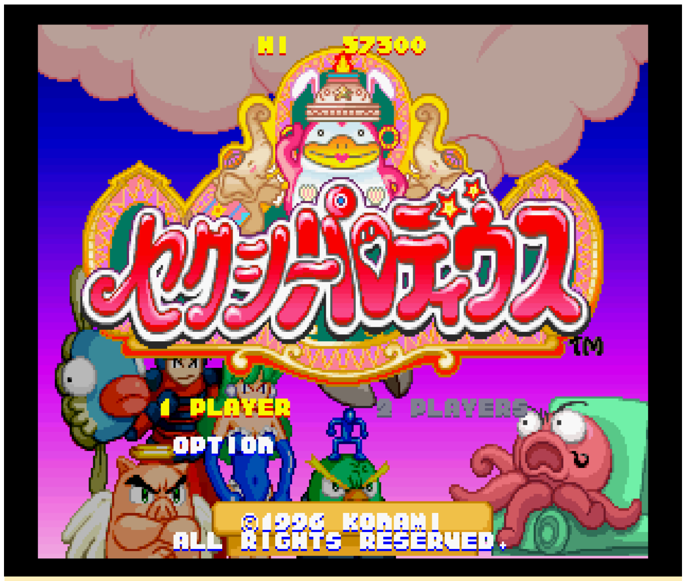


Figure 06 – Sexy Parodius title screen on the Sega Saturn

It has a slight twist at the end of a section where you kill a couple of bosses in a row, and if you beat them, a "sexy picture" is slowly revealed, but if you run out of lives it's over. You have an unlimited amount of continues, so you can still be playing the next level, but the picture level will be over.

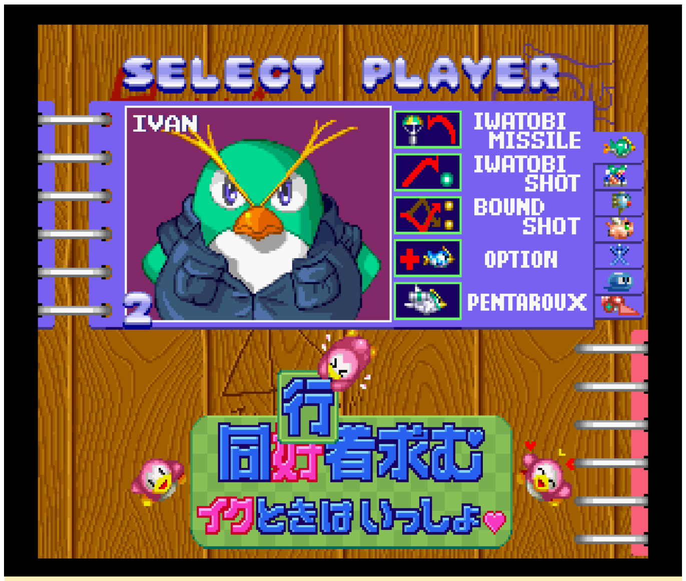


Figure 07 – Sexy Parodius character select screen

Aside from that, the game is a nice "cute 'em up" but actually rather hard and you might die quite often, but as I said, luckily there are unlimited continues.

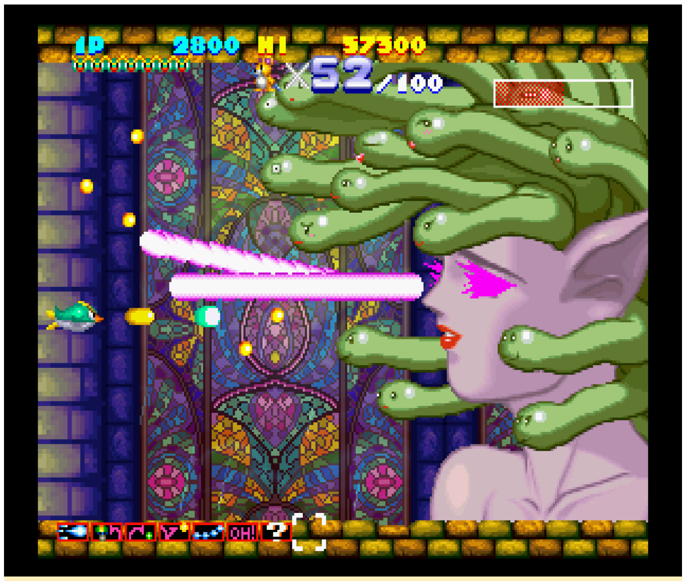


Figure 08 – Boss Fight in Sexy Parodius

Shienryu

Shienryu is another shoot 'em up. It's a little bit simpler in its game mechanics, as you can't choose different crafts, and there are only three different types of weapons.

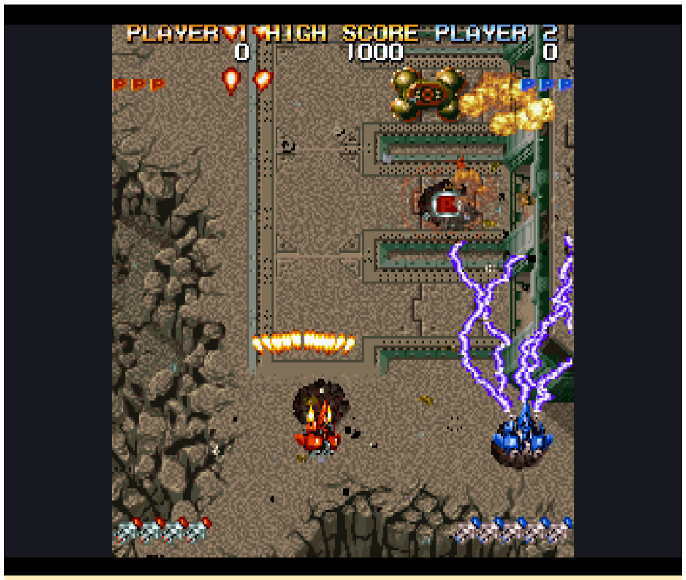


Figure 09 – Shienryu

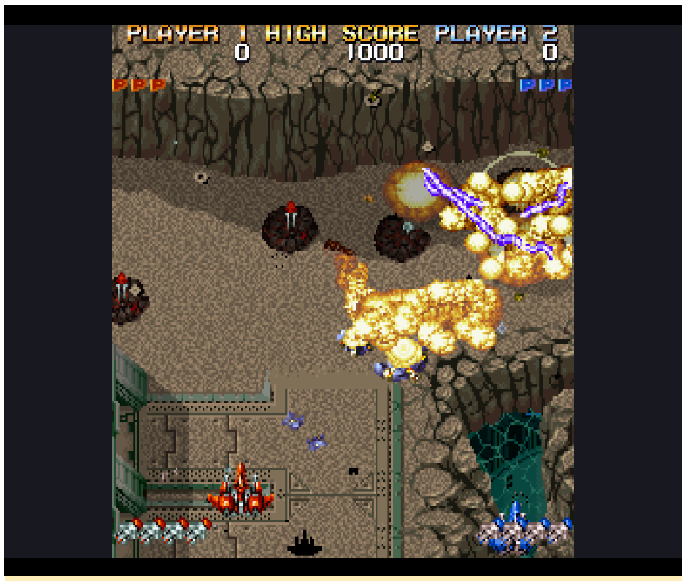


Figure 10 – Very good looking graphics for up to two players lots of explosions on and enemies on the screen

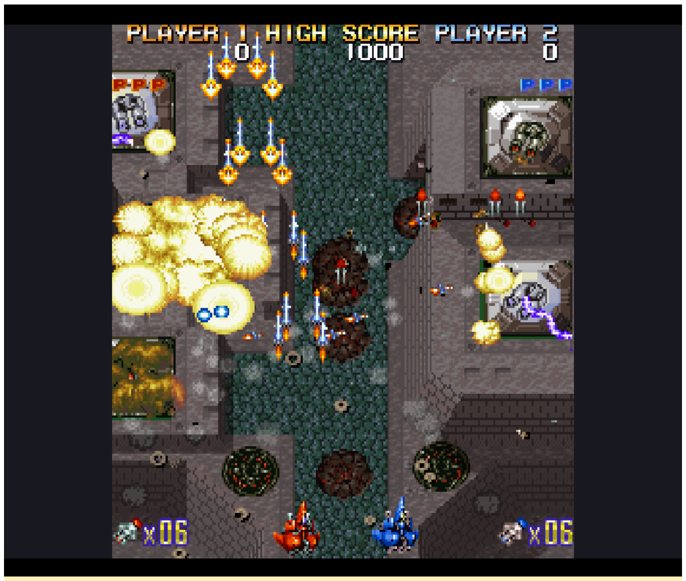


Figure 11 – Rockets can deal a lot of damage and some are self aiming

You either can use a spread shot, a lightning attack that auto aims at all targets and jumps from one enemy to the next, or a rocket attack which is a combination of self aiming missiles and straight forward rockets. Each of the weapons can be increased in fire-power multiple times by collecting power-ups. Aside from that, you have “bombs” which are special attacks designed to deal a lot of damage on a large section of the screen, as found in many other games as well. The game plays very well on the ODROID-XU3/XU4. The graphics are probably the most impressive part of the game, with plenty of details and animations, transparent clouds, as well as a good amount of parallax scrolling. It’s one of the best looking “shmups” for the system.

Figure 12 – No “shmup” is complete without bosses to defeat

If you like “shmups” this is definitely one you should try out. The game is also available for MAME and PS1, but the Saturn version is probably the best.

Shinrei Jusaishi Taroumaru

This game is kinda unique, it looks and feels a little like the Shinobi series. However, instead if throwing knifes or hitting the enemy with your fists, you’re using your inner energy to shoot them with lightning bolts.

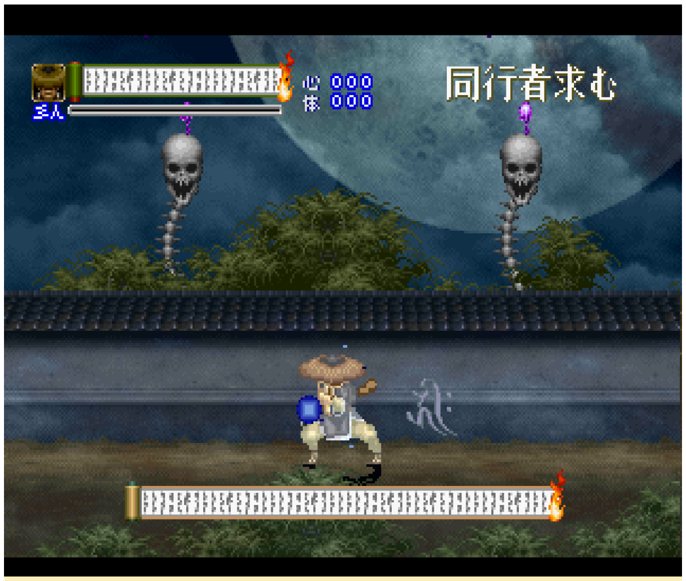


Figure 13 – Start Screen of Shinrei Jusaishi Taroumaru

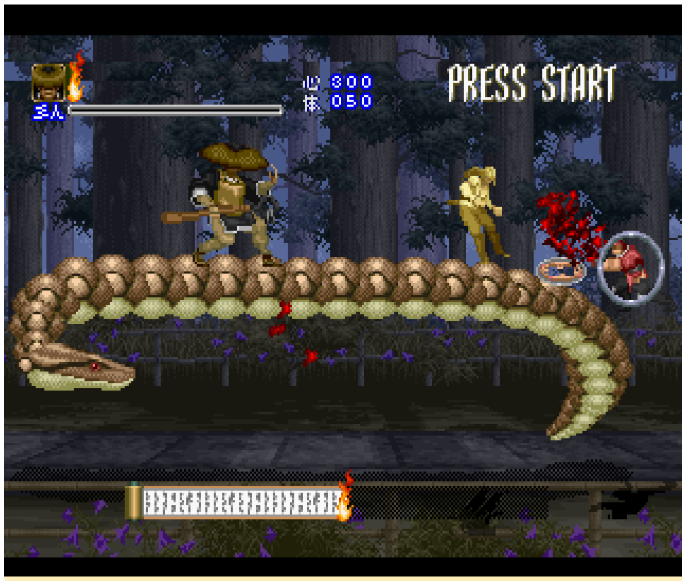


Figure 14 – Charging your attack before the strike to do extra damage

In this game, you walk from left to right fighting off hordes of enemies ranging from moving flames, then to ninjas and demon creatures, and every so often you fight off different boss monster which vary much in strength and ways to kill them. The game is a little bit hard to control in my opinion. You have an auto aiming feature that only locks on if an enemy is in front of you. If an enemy is behind you, you have to turn around, but also if an enemy is right above you, you won't lock on him either.

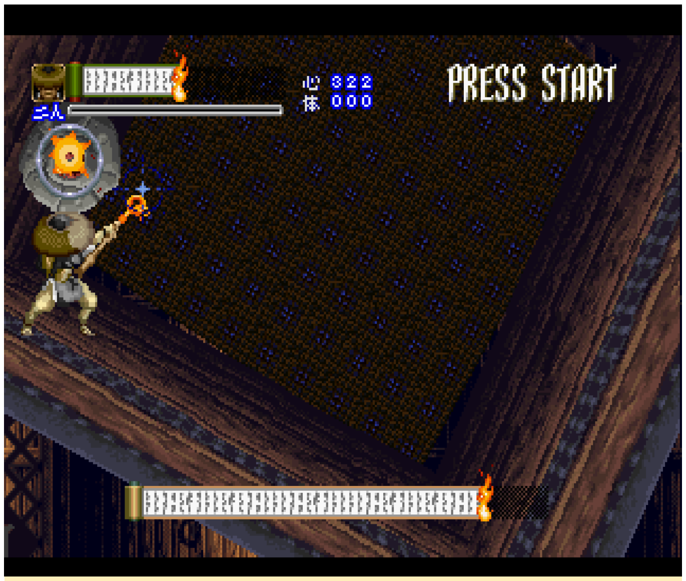


Figure 15 – Fighting a boss together with a captured enemy

Figure 16 – The game even has rotating rooms

The variety of enemies, especially the bosses, is quite impressive, and the different levels and scenes you are fighting in are very good. You won't get bored with seeing the same background over and over again. The game can be quite hard though, but is still very fun to play. You can even "capture" some of the enemies and let them fight for you. You can also reflect some attacks or make projectiles aimed at you disappear that way. The game is quite fun to play, but has no save feature or anything, it's probably meant to be played in one session and can be finished in about an hour or so.

Sonic Wings Special

It seems that “S” does not only stand for Saturn but also for “shmup”, as this is yet again another “shoot ‘em up” game for the Sega Saturn. Once again, the gameplay is rather simplistic in this game. There are no different weapons to collect, and there is no charged attack or anything similar; just your standard attack and a “bomb” special attack.

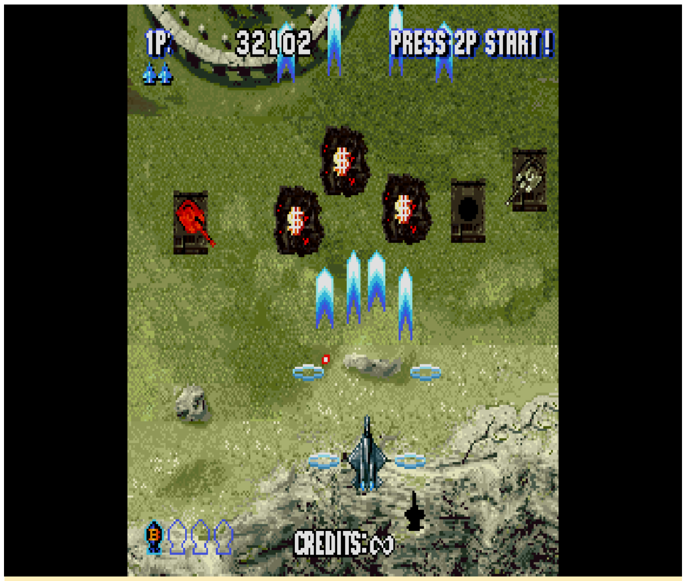


Figure 17 – Sonic Wings Special

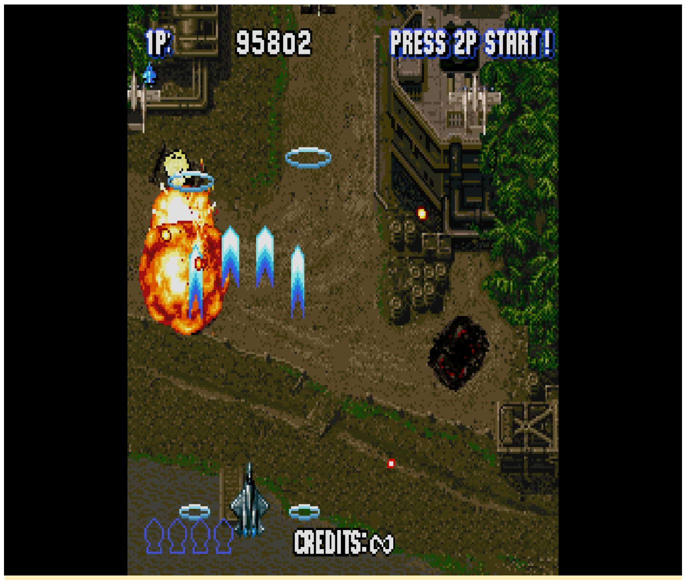


Figure 18 – Nice graphics and explosions but no parallax scrolling or transparency in this game

The lack of different weapons is handled by a vast amount of aircrafts you can command: you can select between 10 different aircrafts, each with their own weapons and special attacks. Another nice addition is that your C button acts as turbo fire button. This means you don’t have to press fire over and over again yourself.

Figure 19 – Lots of different backgrounds and enemies

Figure 20 – Many unique Bosses to kill best keep your specials for these

Even if you don’t die, your weapons will lose their power level if you do not pick up more power-ups over time, and once you get hit, you lose your power-ups and are normally just left stranding with one power-up from your destroyed aircraft. Sometimes it’s good to just let the power-ups fly around on the screen for a while before you collect them.

Steam Hearts

Here is yet another shoot ‘em up for the Sega Saturn. Steam Hearts is once again a rather good looking Shmup on the Saturn, with some level of parallax scrolling and very nice animations.

Figure 21 – Steam Hearts title screen on the Sega Saturn

What I like about this game is, that a single hit does not necessarily mean you're out of the game. Your craft can take a couple of hits until you go down, which is quite nice for a change.

Figure 22

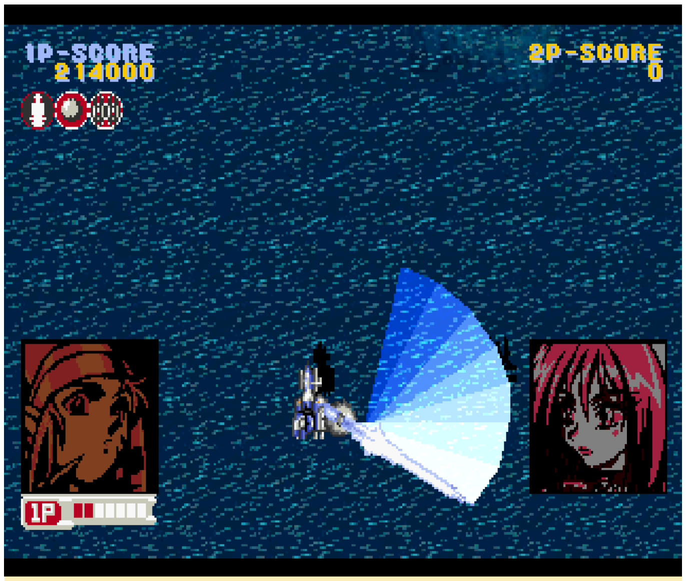


Figure 23 – One of your weapons is a giant sword You can swing to deflect bullets

You have a couple of different primary and secondary weapons you can collect and up to three secondary weapons which all have different styles. This gives you a very wide range of options to combine your powers. These secondary weapons are also used for your special attack and will be "used" for this, which means you'll lose that secondary weapon after you use your special attack. Special attacks are normally stronger forms of your primary attack and not like in many other games a huge screen filling extremely powerful attack. For example, if you have self-aiming laser as a secondary attack and activate the special attack, your rate and strength of these self-aiming laser will increase for a short while, and the Sword will increase its size to fill the entire screen. Things like this can be extremely helpful, but not overpowering, like in other games where you can kill an boss with just one or two special attacks.

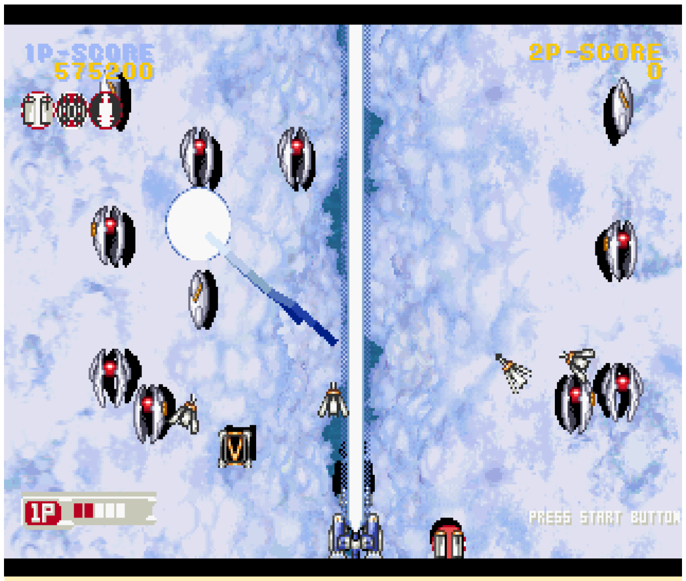


Figure 24 – The laser is probably the strongest primary weapon

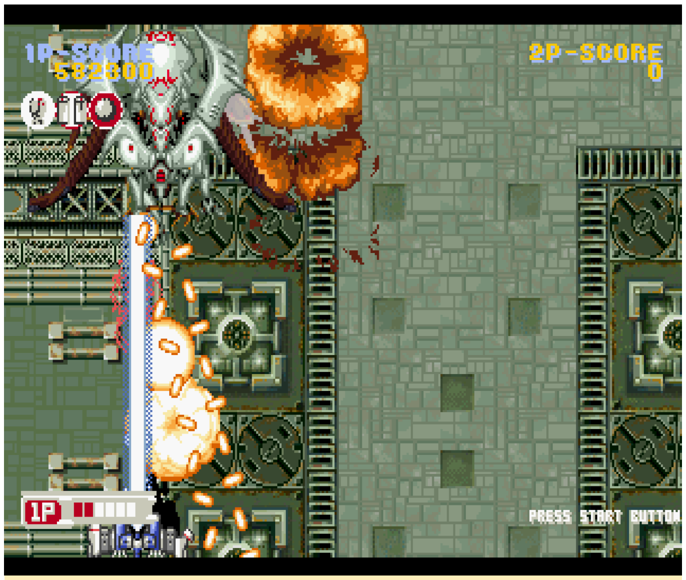


Figure 25 – Killing a boss with the laser is a lot faster than with other weapons

All in all, it's a pretty solid shooter. You can select between a male and female main character and either a blue or red craft.

Strikers 1945

Strikers 1945 is the last shoot 'em up for this series. It came out for many different systems, and there's even a PSP port of the game.

Figure 26 – Strikers 1945 for the Sega Saturn running on an ODROID-XU3

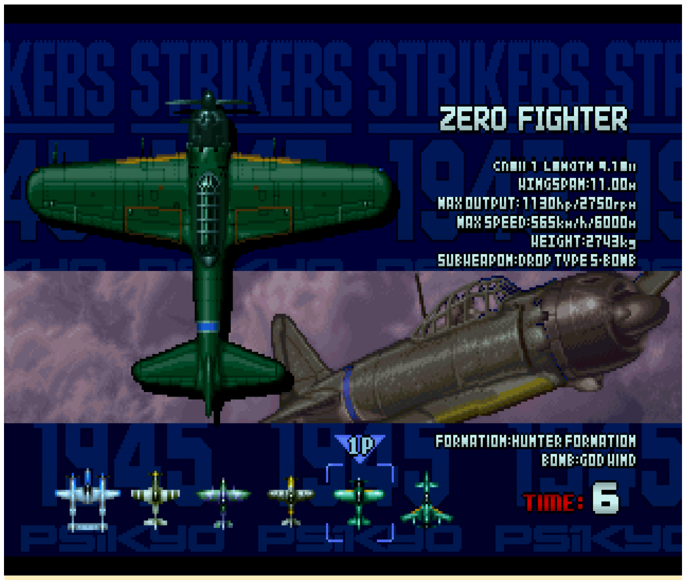


Figure 27 – Select your favorite Aircraft each has different weapons

The game as such is not so special, since it has no parallax scrolling, although graphics and animations are quite good, but nothing that has not be seen in other games. Still, the gameplay is quite fun, the level are short, and you face a new boss on each stage which are randomly thrown at you. No run is the same, as the order of the level changes each time which gives it a very good replay value. The levels are interesting and the planes are different enough that you want to try them all. Strikers 1945 might not be the best shmup, but it does everything right. The gameplay is fast and challenging, but not unfair. The graphics are nice even if it doesn't use parallax scrolling. You have great music, an intro video, unlimited continues, auto-fire on the right shoulder button, and more. Overall, Strikers 1945 is a very solid shooter and it runs very good on the ODROID-XU4/XU3. If you like shoot 'em ups, you should definitely try this one.

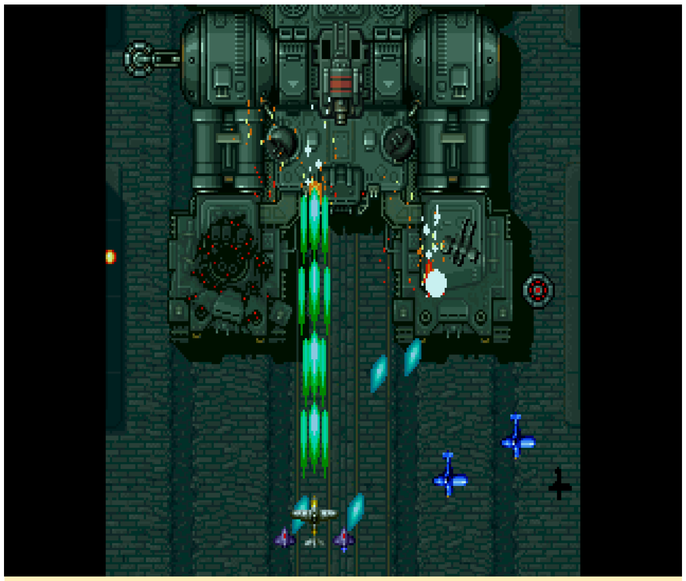


Figure 28 – Boss Battle in Striker 1945

Honorable Mentions

Saturn Bomberman Fight!!

This game is kinda interesting when it comes to Bomberman style games. You do not die on the first bomb that explodes next to you, but you have a health-bar. You can throw bombs, jump over boxes, you ride animals, and more. The game is quite different from the original Bomberman, but is rather fun to play. It also brings Bomberman and his friends into the 3D world which also gives the game a nice little twist.

Shichisei Toushin Guyferd – Crown Kaimetsu Sakusen

The first time you see the intro, you're instantly reminded of an old TV show called Power Rangers, but no, this is Guyferd, which is similar but still different. Unfortunately, I don't understand a word of what is spoken, which is a shame, as the game offers many, many video cutscenes probably from the TV show itself (and I mean A LOT of em!). It plays like a dungeon crawler, where you walk around from the first-person perspective and turn left or right and walk back and forth, until you find items or an enemy, which is when you start to fight. The game is very interesting, so it's a shame that I understand so little about it.

Shining Force III

If you want to play this game for the Sega Saturn, you'd better insert a Memory Backup Card to save your game progress, since the internal memory of the Saturn will probably not be sufficient. I don't really enjoy pre-rendered graphics like in Donkey Kong for SNES or like here in Shining Force III. Still, the game is a very interesting RPG game using a lot of 3D elements and effects, and is one of the better 3D games for the Saturn. It's huge, and from what I heard, the game is suppose to be very "epic" and the best in the series. It plays a little slowly, with lots of text and only little action at the start, and I haven't had time to really play it seriously to give a good review. It's been praised a lot, so I guess it's safe to recommend it.

Silhouette Mirage

Silhouette Mirage is a very interesting game which I do not fully understand yet. You can run, jump, and shoot energy balls from your fingers, and have to defeat Mirages and Silhouettes. There is a lot of shooting, bashing and such that you can do. It includes a tutorial that I never finished, as it takes a long time, and just shows you the different attack combos you can do. The graphics are nice, and the music is catchy. It's a nice game, although I don't understand what I'm really doing.

Street Fighter Alpha – Warrior's Dream

This game might also require you to use a backup memory card as well if you want to keep your high scores, or you have to make sure the system memory has at least 32 free blocks. The game is what you expect from a Street Fighter game. It has nice graphics, good animations, and is good overall, but not extraordinary.

Street Fighter Zero

I couldn't get Street Fighter Zero 3 to work, but it's a better version than the Zero version, and with the 4MB memory expansion, it offers a lot more animations. Still, Street Fighter Zero is an excellent fighting game in the Street Fighter series with nice graphics and animations. There are lots of special moves, and it runs very well on the ODROID XU3/XU4. If you are a fan of the Street Fighter series, you should definitely check this one out.

Street Racer

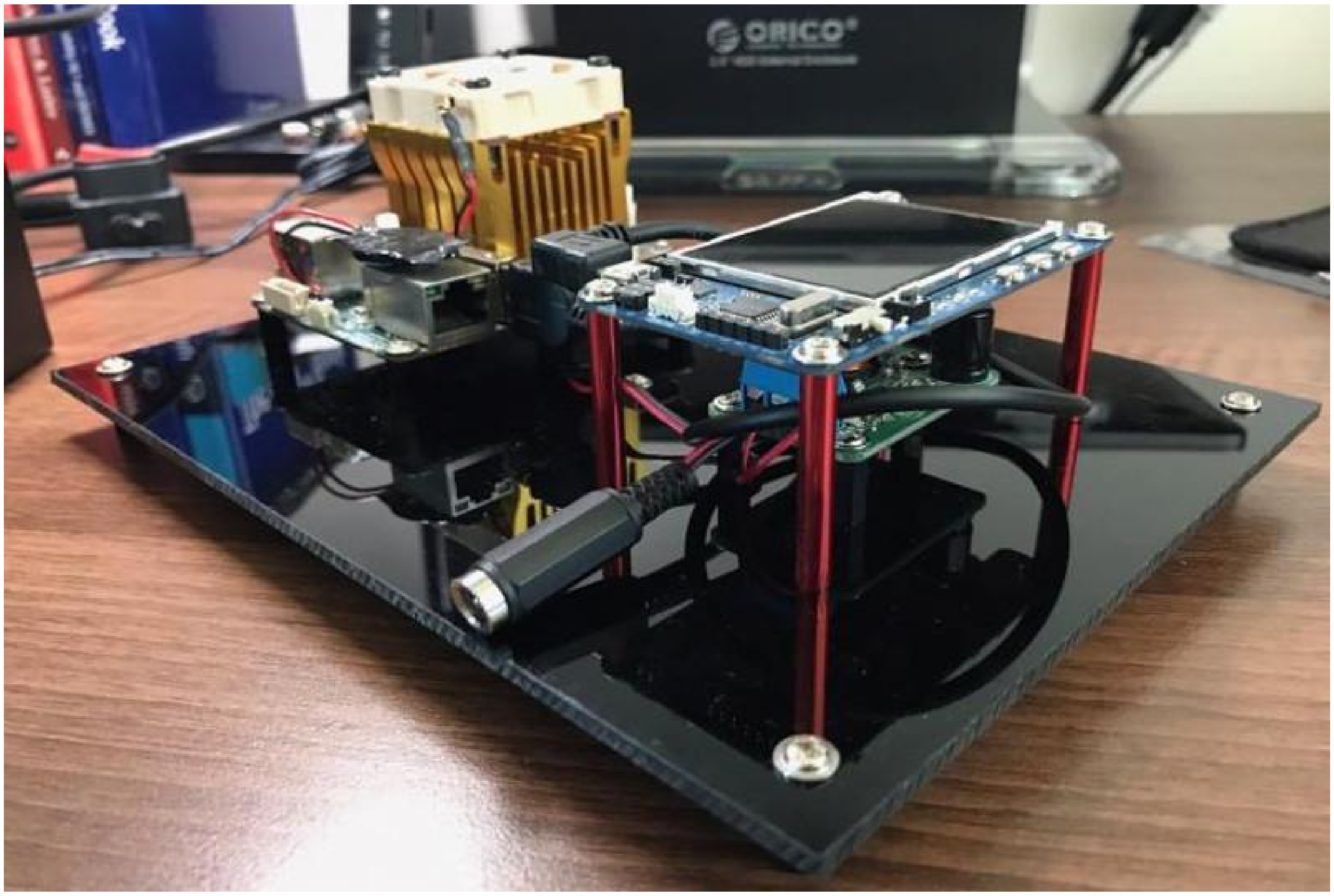
Street Racer is a nice and interesting little racing game with funny cartoon graphics similar to Mario Kart. It has some graphical issues as it seems that the sky is actually on the ground, and it looks like you drive on it. This makes it hard to see the actual track, but not impossible, and you can still enjoy the game a lot. Another bonus is that you can manage your system memory and backup memory save states with this game.

Striker 1945 2

Striker 1945 2 is the second installment in the 1945 series. It is a very nice shoot 'em up, but is unfortunately a little bit too laggy. The frame skipping feels very jumpy, and deactivating frameskipping will make you play in slow motion, which is tolerable but not as fun.

ODROID-XU4 Home Server

June 1, 2018 By Igor Kromin ODROID-XU4, Tinkering



Back in December 2017, I rebuilt my Odroid XU4 home media server (https://goo.gl/6tT6rt) because I was having some issues with the previous setup. Unfortunately, that rebuild was not focusing on aesthetics or cable management, just functionality, because I needed the server up and running and did not care how it looked. In March 2018, I received my 34″ curved screen monitor (https://goo.gl/WekKF9) and then cable management and aesthetics became important, so I decided to do something about the server.

I am not using a fully enclosed case, since I like the look of the separate components being mounted on a single sheet of acrylic, which gives it an industrialized look. I would like to have a top clear sheet, but I have not been able to find stand-offs that are tall enough to support one.

The issue that I was trying to solve here was around cable management. I had two separate power supplies: one for the HDD dock and then another for the XU4 itself. This was not ideal, especially since I had to use an international power adapter on the XU4 power supply (US to Australian). It was messy, since I wanted to use one power supply, but I needed two voltages: 12V for HDD dock and 5V for the XU4.

My solution was to use the 12V HDD dock power supply, and get a 12V to 5V DC-DC converter (https://goo.gl/gKjnzG) to power the XU4. The power supply could provide enough current for both. That resolved the voltage issue. Next, I had to get the 12V to the DC-DC converter. I already had a power splitter cable on the HDD dock that was powering the cooling fan and the dock itself, so I wanted to reuse it. That meant rewiring the fan. This part was easy. I took apart the dock, drilled a small hole to get fan wires through and wired the fan directly into its DC jack. It was a total hack but it did the trick.

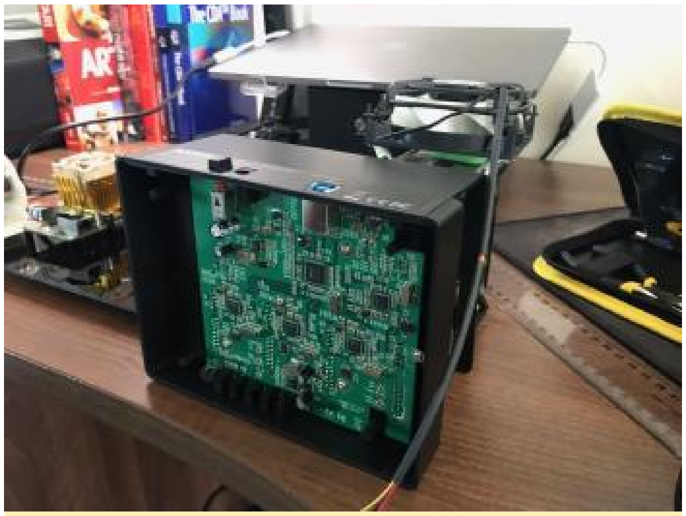


Figure 1 – Rewiring the fan was easy

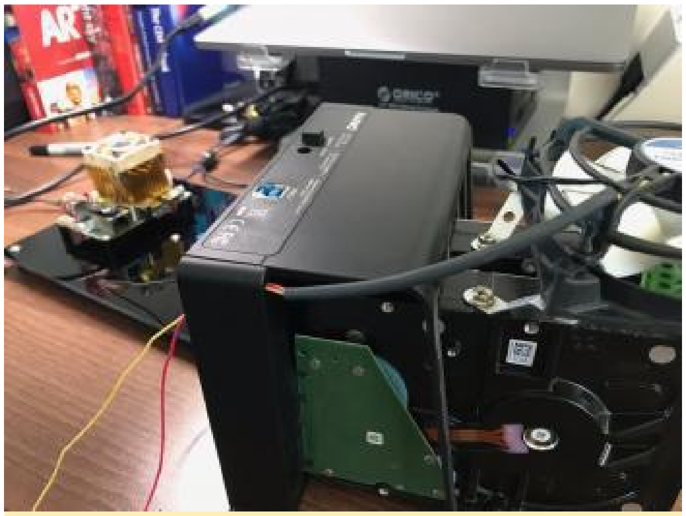


Figure 2 – Another view of the fan wiring

Figure 3 – The rest of the components for the project before assembly

Now that I had the wiring more or less figured out, I needed to get started on mounting everything in place. I had a sheet of 3mm black acrylic around, so decided to use it as the base (after cutting it down to size). I used a whole bunch of black nylon standoffs (https://goo.gl/auDiwD) to keep the XU4 and the DC-DC converter in place. I also used the same standoffs under the base: one in each corner with a center-mounted standoff.

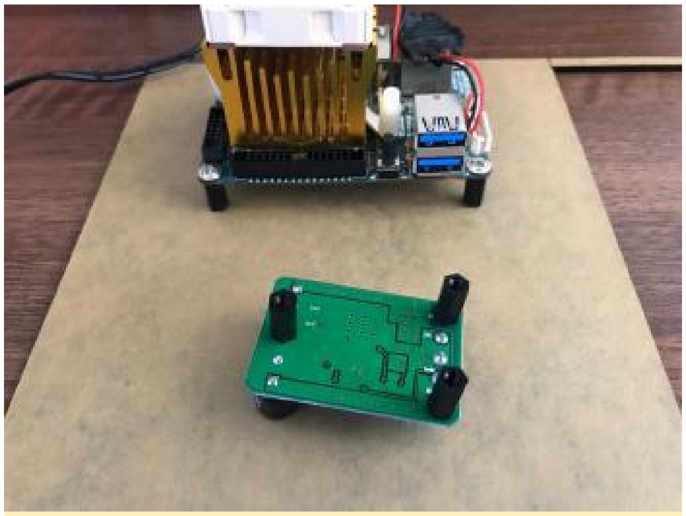


Figure 4 – The standoffs for the XU4 and the DC-DC converter

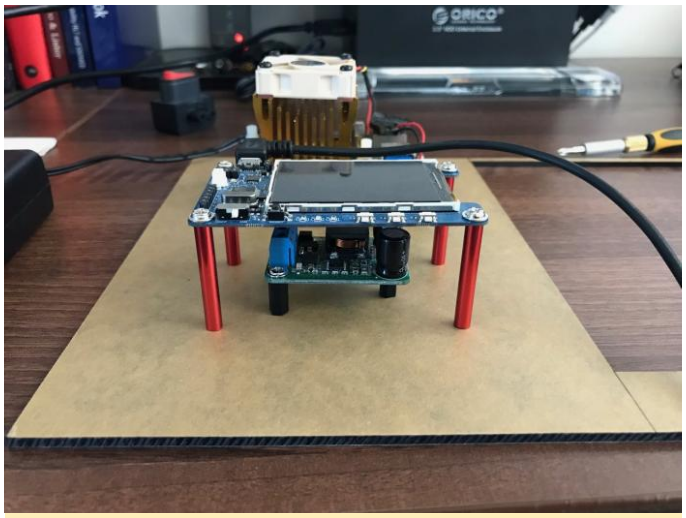


Figure 5 – Another view of the standoffs for the XU4 and the DC-DC converter

To keep everything neat, I wanted to stack the display over the top of the DC-DC converter. For that, I used some nice 35mm red aluminium standoffs (https://goo.gl/G6JTsv). The height worked out really well.

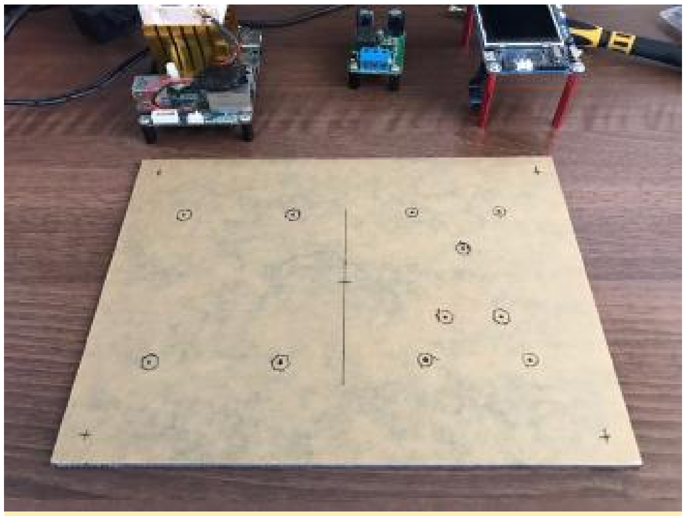


Figure 6 – The acrylic sheet ready for drilling

Then it was a matter of marking out all the standoff positions and drilling holes for them.

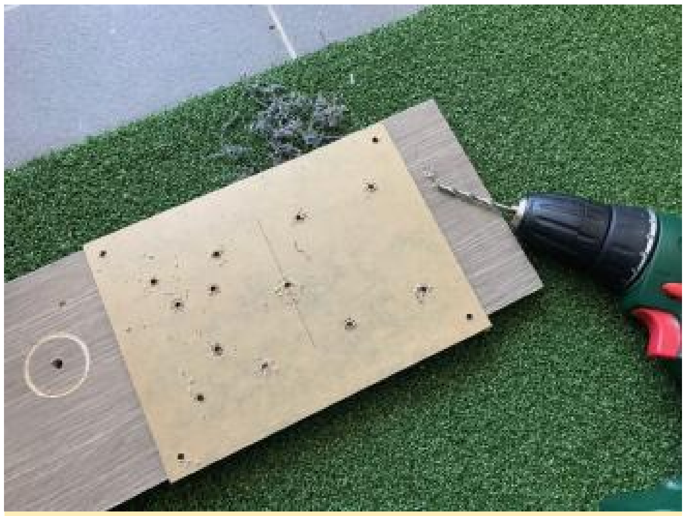


Figure 7 – Drilling the acrylic sheet

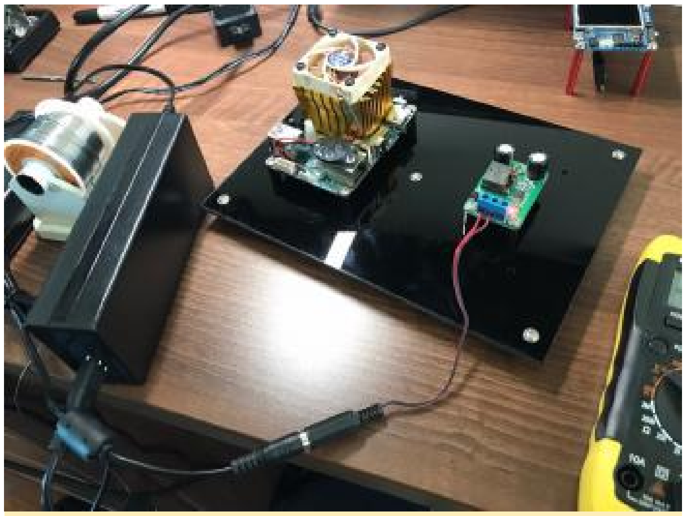


Figure 8 – The components mounted on the acrylic sheet

After securing the XU4 and the DC-DC converter in place. I did the basic wiring from the power supply to the DC-DC converter and made sure the voltages were all correct. Once the voltages were verified, I finalized the wiring between the DC-DC converter and the XU4. The right angle DC jack (https://goo.gl/GvN8KY) came in handy here.

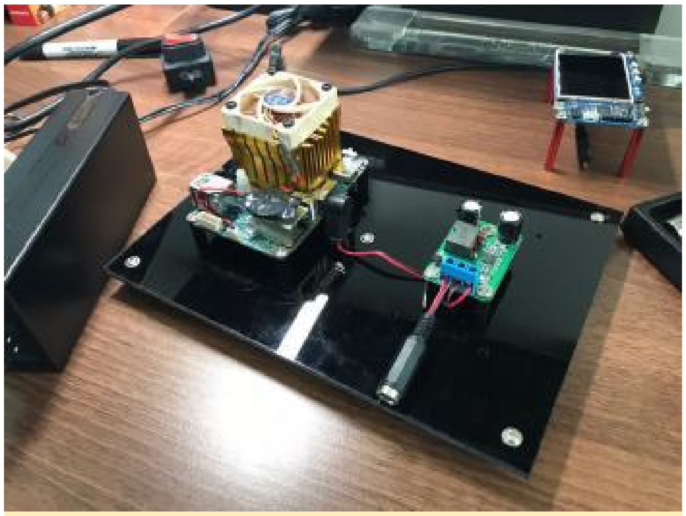


Figure 9 – The components with the wiring tidied up

After connecting everything, the server was ready to go. It sits behind my monitor and looks neat and tidy with no cables showing.

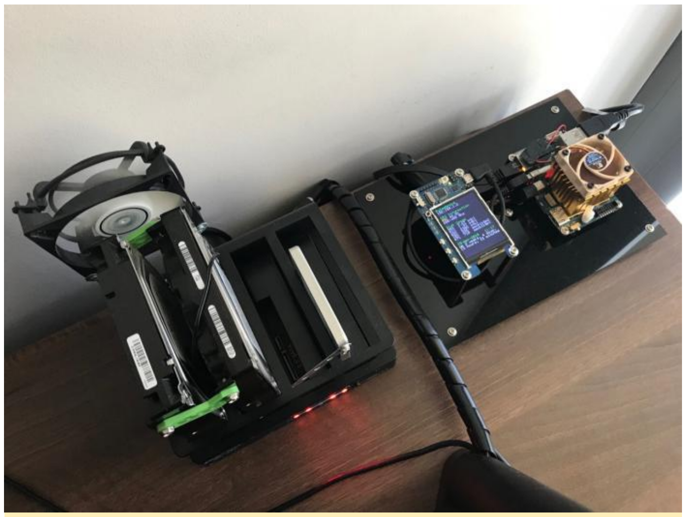


Figure 10 – The final view of the server with cable management and acrylic sheets

The heatsink on the XU4 is not the original one. I replaced it a while back as described in another article caleld ODROID-XU4 standard heatsink and fan replacement. For comments, questions and suggestions, please visit the original article at https://goo.gl/6TL9BT. If you'd like to read more articles by Igor, go to https://www.igorkromin.net/index.php/category/odroidxu4/.

Carputer – 7″ Touch Screen Android

June 1, 2018 By Steve Jackson (@zurkeyon@gmail.com) Android

This is a 7″ Touch Screen Android Carputer with super accurate USB GPS, Bluetooth 4, 3.5MM Audio in/out, WiFi, and an adjustable magnetic screen. A sketchup file is available for additional customization and resizing as needed at https://www.thingiverse.com/thing:2720349. As an IT Field service tech of 28 years, I built this unit out of necessity to replace the smaller screen of the phone, and to provide more utility while on the road. There are pre-built units that you can certainly purchase to accomplish the same task, but I already had the ODROID-XU4 and Vu7 Screen from another project, so I opted to re-purpose them in something that would get more utilization.

Please be aware that some of the following apps/features are illegal to use while driving. You cannot use apps that allow you to type while you operate a motor vehicle in most states. Check your local laws and do not violate them in your use of this device. I am not responsible for improper use or application of this device in violation of the law. This device release is for educational/tinkering only. Pull over, stop safely, then use it. I am typically in a parking lot before or after a service call when using this device, or I stop and get safe, then do what I need to do. Be smart in using this device, and don't get yourself killed over what is essentially a nerd toy with some business applications.

This carputer project is an inexpensive way to modernize your vehicle

GPS

The USB GPS Units are way more responsive and accurate than the ones in your phone. This is a huge time saver, as it allows me to better anticipate my turns and avoid waiting at extra lights and making U-turns. It also instantly locates satellites with no waiting.

RepairShopr dispatch portal

I use this calendar/portal to see my appointments in real time, and to pull up the customer's address for quick-queue of GPS trips. Zero typing and zero talking to Google. With a Bluetooth RII keyboard, I can also sit and make site notes in the privacy of my truck, while listening to my news or music.

Music streaming

I use Spotify, Pandora, and a multitude of other free music services to provide my truck with sound. These all integrate well into the setup and allow for controls from other programs, as well as the the hard buttons on the car stereo head-unit.

Command Center

The command center includes a digital speedometer (GPS), app wheel, date, time, weather, audio controls, and even an audio visualizer, which is all possible through Car Launcher Pro.

WAZE

WAZE is a speed-trap, cop, and accident finder with real time traffic information, alternative GPS to Google Maps, and much more. This even shows you the posted speed limits to a pretty good degree of accuracy, although it's not always right, so keep an eye on the street signs.

Torque PRO

You can pair Torque PRO together with an ELM327 or similar Bluetooth OBD-II scanner and you end up with a real time engine information system. It features a full spectrum of digital gauges that are customizable into many different configurations. For cars with or without Turbo, there are boost and vacuum gauges, acceleration gauges, and it even reads engine codes for most OBD-II cars and trucks. There are other apps out there that do this, but Torque Pro, which is a paid app, works perfectly for my needs. It is nice and stable on the ODROID-XU4 Android images.

WiFi boundary testing and ranging

WiFi testing a large property for wireless coverage, like an apartment complex or hotel, can be tedious if you have to walk it. WiFi Analyzer is a great utility that provides an audible ping for WiFi strength. It allows you to drive a property and map the WiFi boundaries quickly onto the site plan, exposing coverage gaps with ease, all from the comfort of your AC, which is great when you are down here in sunny Florida. This tool allows me to stay online 24 hours a day with something a lot more useful than a tiny phone screen. Whenever my phone is near, my hotspot connects up and I'm all set.

Additional Uses

- Digital LED signage, using Neopixels and a decent battery setup to run them
- Custom LED, neon, and other lighting controls for effects and show lighting
- Custom fuel system Interface and No2 controls
- Private investigator/Detective vehicle and GPS tracking interface for investigations
- Backup camera or dash cam
- Vehicle deployable FPV drone base station for site/building inspections and security
- Rear headrest movies/video gaming system for entertainment of passengers and children
- EV charge reporting and electric motor, AC, regenerative, and other controls

Components

- ODROID-XU4 SoC computer with eMMC of 32GB + RTC Battery to keep time: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G143452239825, http://www.hardkernel.com/main/products/prdt_info.php?g_code=G145628097465, http://www.hardkernel.com/main/products/prdt_info.php?g_code=G137508214939, which is a fan-cooled heatsink is strongly recommended for the XU4 in this setting.
- ODROID-VU7 or VU7+5v TouchScreen: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G144549683088
- 4-5 Port powered USB hub: My build includes a modified power in cable to get its 5v power from DC-DC converter listed below (https://goo.gl/WZmPUA)
- 1-2 foot optional USB extension cord (https://goo.gl/GDQzkv)
- 12v to 5v power supply to run both the ODROID-XU4, and the powered USB Hub (8 AMP+) at https://goo.gl/mwT1ng. You will also need a single 40mm x 40mm x 10mm 12V DC fan for cooling up to 10-12 amps
- ODROID USB Nano WiFi adapter: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G145431421052
- ODROID USB Bluetooth 4 adapter: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141656580698
- Short HDMI cable, as thin as possible, for screen video: The ODROID-CU7 included HDMI cable is far too thick to use (https://goo.gl/HuQdFy)
- Short USB to USB micro cable, as thin as possible, for screen power and touch: The ODROID-VU7 Screen comes with this part.
- K and J Magnetics round neodymium magnets (2), to hold screen in place: https://goo.gl/SeuHpM
- Ar oll of ABS material, in any color you like: https://goo.gl/nYJPB4. This works okay, but you get what you pay for. I'm sure I could have found better.
- Decent size twisted pair wire. Nothing too small, as it's going to be moving 5+ amps.
- ODROID USB 5.1 Sound Card with line-in, the for the mic so you can talk to Google: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141629380416
- Gmouse USB GPS unit, compatible with ODROID.: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G142502154078
- DC plug for ODROID-XU4: http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141440511056
- PC microphone https://goo.gl/YQGT9p
- Sally Hansen Insta-Dri nail polish in Back to Black: This is nice for finishing the screen face with a glossy finish, which is available at Walgreens and similar stores.
- An on/off power toggle switch

Uses a combination of M2 and M3 hardware. I also had a few small coarse thread screws laying around from another project, similar in size to coarse thread M2. But you should be able to see what is needed from the hole sizes.

Upcoming features

Auto On/Off with vehicle ignition: Unfortunately, I have not had enough time nor the desire to work this circuit out and implement it into the design. Anyone who wants to can certainly customize one up and show us how it's done!

Hard buttons / Joystick control: I was considering using a thumb stick with a push button for hard controls, but couldn't decide where to place such a thing.

Double Din, fully independent version with built in 4-channel amp,and line-in/out: Clamping system added, for easier mounting into the DIN pocket. The combination of a wedge fit and screws at the moment makes taking in and out difficult.

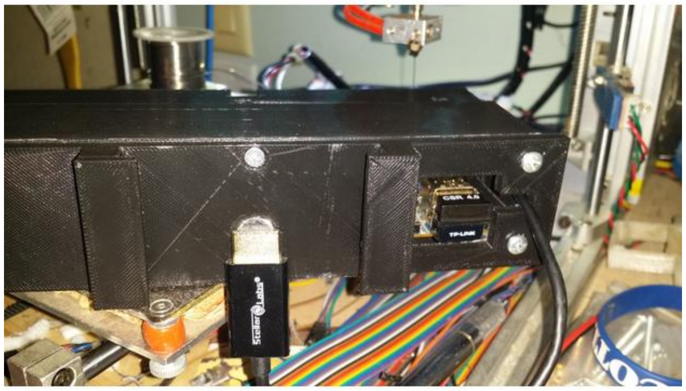
Sound reactive lighting controls or HAT**:** This would really set this device off from the other stuff out there. No off-the-shelf Android head unit can currently do this without additional components. A HAT would allow the Xu4 to be a true custom car command center.

Printing

Use ABS or PETG for best results. Expect warping, working at the extreme edges of your print surface, and layer separation issues. A heated enclosure works well. Use cardboard, corkboard, tin foil, and a heat lamp or track lights; just watch for heat and don't leave it unattended.

Carputer case for screen

Side view of computer enclosure



Back view of computer enclosure

Post-Printing Assembly

After printing the parts, install the eMMC into the ODROID and install your Android OS. Do a dry run without installing the unit into your vehicle and make sure everything is working correctly. I strongly recommend running the unit for several hours to ensure there are no heat issues with your power supply.

Screen fitted to computer enclosure

DROK supply for me was best when set to about 5.7v. The VU7 is known to slightly flicker when the XU4 is working hard due to current drop across the 5v rail. The power gub tries to fix this, but you gotta turn up the juice a bit. 5.7 was my "sweet spot" for the least amount of flickering.

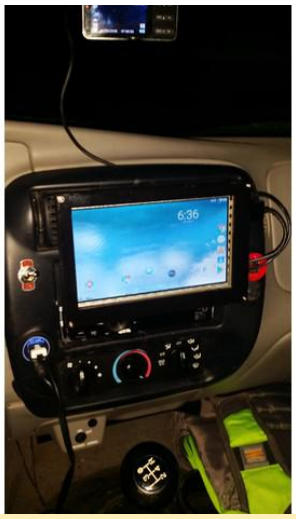
The setup works the CPUs very heavily! Warm them all the way up and put the XU4 under as much load as possible. Benchmarking programs work well for this. You don't want a fire in your dashboard. If any components feel too hot to hold in your hand, they are running far too hot to put into an enclosed dashboard, which means you need to add ventilation fans.

Dashboard opening

Find spots for the USB hub and power that do not come into contact with a chassis ground, or you could have issues with shorts. Make sure all items are properly enclosed with enough venting to protect the electronics. The Din body screws into the surround, or can be used with threaded set tabs and tightened into place that way.

Having a line out to an AUX or other source is still an issue on this project. I am still looking for a clean AUX out for the **ODROID-XU4** while its connected to automotive 12 volts. For the moment, the unit only provides clean audio via Bluetooth to your existing head unit in your vehicle.

Some have suggested to try the **line-out 3.5mm jack** on the Boombonnet, and just leaving the little speakers off of it entirely. Being I2S, it may not suffer from as much noise. The power supply I used here injected 50% noise into the audio signal, making it totally unusable.


Carputer fitted into dashboard

Noise and ground loop filters cut out bass frequencies and other frequencies, and distort the sound on any system using an amplifier and subwoofer. Trust me. I tried them all. None of them sounded good after being placed inline. The noise was gone, but so were 50% of my sound frequencies.

Additional considerations

Make sure to use care when using Neodymium magnets in any project. They are known for coming together and shattering, causing sharp metal to fly off at high speeds, possibly blinding the user, so be careful! Make sure your magnets are faced the correct way before gluing them into final position, or you will be prying them out and likely shattering them in the process.

Every single attempt to print this in PLA ended in failure within a few days. It was becoming warped in the car due to being inside the hot dashboard. Don't bother trying it; it won't work.

To print the components, and for further details, please visit the original article at https://www.thingiverse.com/thing:2720349.

Introduction to BASH Basics – Part 2: Useful BASH commands for Single Board Computers

June 1, 2018 By Erik Koennecke Linux, Tutorial

Last time, we learned about the 'ls' and 'tree' commands. While looking at things is nice, it's more fun to actually do something with our data. This article contains a list of the common commands for manipulating data. The command and its explanation are kept very brief to avoid writing a Linux encyclopedia and boring you all to death. They should just bring the existence of a BASH command to your attention.

As a reminder, whenever we want to learn more about something, typing 'man ' into a Terminal window will give the usage syntax as well as all the options available for this program. Be careful, though, because the man pages often have content accumulated over decades and can be hard to digest, especially for beginners. If you want to hit the ground running, open www.explainshell.com in your browser to get commands explained, like the examples mentioned here. For useful starters to build your own commands, use the TLDR pages at tldr.ostera.io. Compare the manual for the tar command with the TLDR version and you will see what I mean.

Basic BASH commands

The most used and most basic commands are:

- cd – change directory. 'cd /' changes to the root directory, 'cd ~' back to the user's home directory.
- mkdir – make a directory. Try 'mkdir ~/test' to make a subdirectory in your home folder named test.
- cp – copy files. 'cp /path/to/file /home/odroid/test/' generates a copy at the freshly made directory from before. With the -R option, you can recursively copy files and directories.
- mv – move files. The syntax is the same as cp, but the file gets removed after the copy was successfully made.

- rm – remove files

```
$ rm exampleFile.txt
```

- cat – concatenate files. cat reads from standard input and writes to standard output if no other options are given. If you use cat file, it writes the contents of file to the console, the standard output. Try 'cat /etc/fstab' for an example.
- less – If you want to read longer files on the console. 'less filename' shows the content of file and pauses after each page. You can advance with space and go back with b and q to quit.

```
$ less exampleFile.txt
```

- head and tail – By default, head file shows the first 10 lines of file, while tail of a file shows the last 10 lines of file.

```
$ tail exampleFile.txt
$ head exampleFile.txt
```

- tar and zip – For archiving files. tar cf archive.tar /home/odroid/test/ archives the test directory and all of its contents. If you add the switch z (tar czf), the archive gets zipped afterwards.
- sudo and su – For everything requiring elevated rights, you prepend a sudo to run the command as root, the super user. cat /etc/sudoers will fail. If you use sudo cat /etc/sudoers, you can read the file. su is 'switch user' If you have assigned a password to the root user or have other users on the system, you can switch to them with this command. With sudo -i, you can also switch to an elevated command prompt and work as root until you exit with.

```
$ sudo -i
```

- exit – to exit the elevated prompt or the shell. Useful also for remote connections.

```
$ exit
```

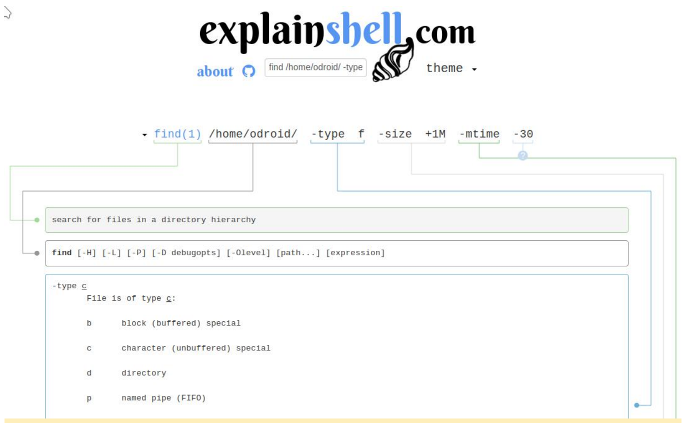
- ssh – You can login to a remote console of your ODROID with this. This is where the BASH console really shines. Sit at your PC, log into the ODROID where ever it is and work as if you are sitting in front of it. It is very inconvenient to use a GUI in this case, this is why the console and BASH on your SBC are more important. On a desktop PC, you may be able to avoid the command line most of the time. To connect with ssh you need the computer's or ODROID's IP address and the username you want to login as. For example the user is 'odroid' and the IP is '192.168.0.1'.

```
$ ssh odroid@192.168.0.1
```

- grep – Shows files which contain a certain text pattern. You want to know which files in the /etc/ directory contain the word 'Ubuntu'? grep -rnw '/etc/' -e 'Ubuntu' gives you the answer.

```
$ grep -rnw '/etc/' -e Ubuntu
```

- find – You can use find to find files with almost all imaginable criteria, and act on them. Simplest example: find /home/ -name '*test*' finds all files and directories which contain 'test'. find /home/odroid/ -type f -size +1M -mtime -30 is a more elaborate example. It finds all files (not directories) in /home/odroid bigger than 1 MB modified in the last 30 days. You can also use Explainshell.com to see what it does.



```
find(1) /home/odroid/ -type f -size +1M -mtime -30

search for files in a directory hierarchy

find [-H] [-L] [-P] [-D debugopts] [-Olevel] [path...] [expression]

-type c
       File is of type c:

       b      block (buffered) special

       c      character (unbuffered) special

       d      directory

       p      named pipe (FIFO)
```

Figure 01 – explainshell.com explaining our find command

- sort – sorts files.
- df and du – disk free and disk usage show how much space you have left with df or how much your individual files and folders use with du -ch.
- ps and kill, killall – ps lists running processes, if you want to see everything running, use ps aux. The process number used with kill, so kill , will kill the process by sending a signal to terminate the process. killall kills all processes with this name.

```
$ ps aux
$ kill
```

- mount and umount – Mount external devices and drives with mount /dev/ /path, unmount them with umount /dev/.
- dd – With dd, you can write from everywhere to everywhere. It's also called the disk destroyer for a reason, so be careful! dd if=/dev/mmcblk0 of=/home/odroid/mbrbackup bs=512 count=1 makes a copy of the master boot record. Very nice when you want to flash a new SD card.

```
$ dd  if=/input/path of=/output/path
```

- shutdown and reboot – they shutdown and reboot your SBC. Several options available.

BASH expansion and wildcards

- * and ? – ls *.txt lists all files ending with .txt. With ls Image??.jpg, you can list files with names like Image00.jpg or ImageAB.jpg.
- [abc] – ls /dev/sd[abcd]1 lists the partitions /dev/sda1, /dev/sdb1, /dev/sdc1 and /dev/sdd1.
- | as selector – With mount | grep 'mmc\|sd', you can list only mounted eMMC, SD cards or external drives or sticks. It expands to 'everything with mmc OR sd in the name'.

Advanced BASH commands – redirection, data flow and loops

- stdin, stdout and stderr – These three files (remember everything in UNIX is a file) always exist. stdin is the keyboard, stdout is the screen and stderr are the error messages output to the screen.
- | – You can pipe output from one program to the next. The find example output from above is much easier to read if you use find /home/odroid -type f -size +1M -mtime -30 -exec du -h {} ; 2>/dev/null | sort -n. The output from find gets used by du to show the sizes, and all errors from stderr go to /dev/null, the sink for unnecessary data. Once again, if paste that command in explainshell.com you will see a broken down view of what each part does.

```
$ find /home/odroid -type f -size +1M -mtime
-30 -exec du -h {} ; 2>/dev/null | sort -n
```

- /dev/zero and /dev/null are the source for null bytes and the sink for everything you don't need.
- > and <, >> and << - These redirect output or input to files. ls *.txt > texts.list generates a list of text files in the current directory. ls Documents/*.txt >> texts.list would append the list of .txt files in the Documents directory to it. 2> is the stderr output. The 2>/dev/null part in the example above redirects all error messages away from the console into the Big Bit Bucket.

for, do, done – BASH loops can be constructed with these commands.

if, then, else – Advanced loop construct, mainly for scripts.

Practical application example

Let's do an example to see the power of the shell and of loops. Your Pictures folder contains several subdirectories, each of which you want to compress into an individual archive to mail to someone else. This is easily done with the GUI if you have just your folders from your travels labeled 'USA', 'South Korea' and 'Spain': right-click, select compress, and you're done.

Now think about the same problem, but with folders generated by a program from a camera attached to the ODROID. If you have 100 folders or more, this task becomes mind-numbingly repetitive. Here is where BASH comes to the rescue!

If we use the 'ls -F' command to list a directory, symbols are added to the names according to their type. Directories have a slash appended, which is also how BASH sees them. Aid to memory: The slash could be part of the path name to a file in this directory, so they get a trailing slash:

```
$ for i in */; do zip -r "${i%/}.zip" "$i";
done
```

This command gets all directories in the directory we are in: the for i in */; part. For each of them, we do the following: zip -r "${i%/}.zip", where $i is the i variable we introduced before, i%/ is the name without the slash, and “${i%/}.zip” is the directory name with .zip attached; directory 2018 05 20 would give the archive 2018 05 20.zip. The quotes avoid that names with spaces are treated as two different names instead, which could have disastrous consequences. After this command has run for all i, we are done.

A short one-liner saves a lot of tedious work. This is what makes BASH so attractive. In the next article, we will look at startup and login and customizing the BASH prompt.

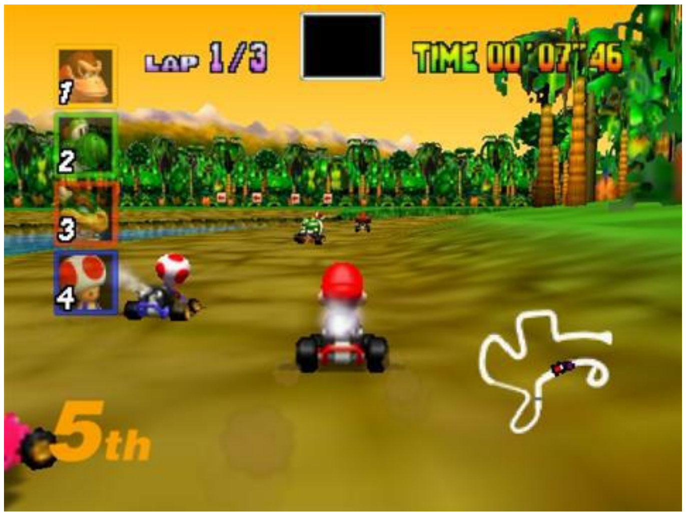
Linux Gaming: Nintendo 64 Emulation – Part 2

June 1, 2018 By Tobias Schaaf Gaming, Linux

Part 1 of this article introduced the latest version of the Nintendo 64 emulator for Linux and compared its performance on all of the current ODROID boards. This second part presents an overview of some of the more popular Nintendo 64 games, including Mario Kart, Mario Party, Paper Mario, Star Fox, Star Wars, Starcraft, Super Mario, Super Smash Bros, and Legend of Zelda.

Mario Kart 64

Mario Kart is very well-known franchise from Nintendo as a racing game, starring the most famous characters from Nintendo like Mario, Luigi, Peach, Yoshi, Donkey Kong, Bowser and others. One of the big benefits of this game is that you can play it with up to 4 players at the same time. I'm not really a fan of the series, especially the Nintendo 64 version, which is graphically poor in my opinion. Although the N64 is known for its 3D capabilities, Mario Kart 64 uses mostly 2D sprites, which don't look good. The only 3D elements of the game are the ground that you are driving on, and some obstacles and bridges, which makes the game very unattractive to play.



U3

When I first ran the game without frameskip, it was rather laggy. Since the game mostly uses old 2D sprites, it really made me wonder why this game needs so much CPU power. However, once I activated frame skipping, it worked fine on the U3. There is some small delay in the sound while using the menu, but nothing that's really troublesome. In-game racing works fine without lags or slowdowns, and multiplayer with several controllers is working perfectly as well.

C1

While the menu is slow, the in-game experience is good and seems to work at full speed using the Rice plugin. It's definitely playable, although you get a much better experience on the U3 or XU3 rather than the C1. When I re-tested it using glide64mk2, the game ran fine, although it had some glitches with the shadows and ground textures.

XU3

Mario Kart 64 had no issues on the XU3. It ran at full speed and could easily be controlled with an the XBox 360 controller.

Mario Party

Mario Party is a type of board game in which you play with or against up to 4 players in different kinds of mini-games. The game is quite fun, although sometimes I have a hard time figuring out the controls for certain mini-games. It's probably suited for all ages, from small children to adults as a party game, or just to have some fun.

Figure 2 – Mario Party

U3

The U3 experience was flawless, and the game ran at full speed without any issues. I saw a flickering screen on the split screen of one mini-game once, but the moment the action started, it was gone and therefore fully playable.

C1 – rice plugin

The menu was a little slow at first, and when I was actually on the map to select a game, I was rewarded with a very fluent movement, like in Mario Kart. However, when I tried to start a game I only saw a white screen. I heard everything running in the background, and clicking buttons triggered certain actions which I could hear, but I could not see anything besides a blank screen. When I tried again with a different game mode, I was able to see a few parts of the game, but major parts were missing, and the moment that I started a mini-game, I only got a black screen.

C1 – glide64mk2

While the game didn't run using the rice plugin, it worked fine with glide64mk2, although it was a little slow. Most scenes in game are full speed, so I consider this game playable under glide64mk2 in 16-bit.

XU3

The XU3 had no issues at all playing this game. It ran smoothly, which was not surprising considering that it also ran well on the U3. Overall, the gaming experience was quite nice.

Paper Mario

Paper Mario is a mix between a jump and run game like Super Mario and an RPG game like Final Fantasy. It has nice graphics, and although the world is 3D, Mario himself is only 2D. He's actually a paper figure. The gameplay is very unique and is really fun to play. It's hard to describe, but you should definitely give this one a try!

Figure 3 – Paper Mario

U3

The U3 experience for Paper Mario is really good. I encountered a few graphical issues with shadow, fonts and speech bubbles. I could not read what the stars were saying, but that's about as bad as it gets. I could still read everything else, so it didn't actually interfere with game play. The overall speed was very good, and I enjoyed the game a lot on the U3.

C1 – rice plugin

The experience on the C1 is hard to describe. At first, the game was not working at all. After a laggy introduction, the main menu did not show up. After 10 or 15 minutes, another type of introduction seemed to show up, which was basically just a scrolling background picture. Another 10 to 20 minutes later, the picture changed again and suddenly I saw the start menu. I created a new save state and started a new game. Again, I was presented with a single background picture. It seems the game is not working at all on the C1, or it might take hours for it to start. The C1 should be able to play the game in rather a decent speed, but unfortunately, the faulty drivers and graphics support prevent the system from working properly.

C1 – glide64mk2

This game works with glide64mk2 at full speed. Similar to the U3, it has glitches with the shadows and ground textures, but besides that, the game is running very well.

XU3

The libretro core did a very good job with this game. None of the U3 glitches with glide64mk2 could be seen. The shadows were perfect, speech bubbles were fine, and I could read what the stars were saying. The overall speed was perfect as well. I really like playing this game on the XU3.

Star Fox 64

Star Fox 64 is a remake of the Super Famicom/SNES game Star Fox, which was one of the first 3D space shooters. The N64 version was famous for its very good graphics and especially for its voice acting. The often funny lines of your comrades through the radio, the intense battles, and the good graphics make this game really fun to play.

Figure 4 – Star Fox 64

U3

The game runs very well on the U3. It had some slow downs on the galaxy map where you select the mission, and the shadows are too dark. The lighting does not work correctly which means the game is very dark in some scenes. Besides that, the game works perfectly well at full speed.

C1 – rice plugin

The C1 does well with this game. The rice video plugin looks a lot better when rendering shadows than the glide64mk2 on the U3, so the scenes are not as dark. Besides that, the performance of the C1 is slower than on the U3, and the mission briefing is slightly laggy. While the U3 has a slowdown on the Galaxy map where you can select your mission, the C1 hangs very badly, but since it's just for selecting your mission it doesn't affect game play that much. When you're finally on the hunt and shooting through the game, the game runs at full speed without issues, and is actually nice to play on the C1.

C1 – glide64mk2

Similar to the U3, the gaming experience is rather good. It's about the same speed as on the U3 and has the same issues with the shadow, but besides that, the gaming experience is nice and only slows down on the galaxy map.

XU3

As usual, the XU3 experience is the best. The game runs smoothly, but slows down on the galaxy map. The graphics look great on the XU3, and the game runs very smoothly.

Star Wars Episode 1 – Racer

I played this game many years ago on the PC with my 3DFX Voodoo graphics card, which used the "glide" that's included in some of the graphic plugins for mupen64plus. The game is about the Pod Racer in Episode 1 of Star Wars. It's a very fast racing game with nice graphics and destroyable objects, and you can upgrade your pod to make it faster or easier to handle. This game actually uses the memory expansion pak on the N64 which improved the graphics, and the rumble pak is also supported. However, the N64 version doesn't compare to the PC version in terms of graphics, and is also missing the multiplayer mode, although it's still a nice racing game.

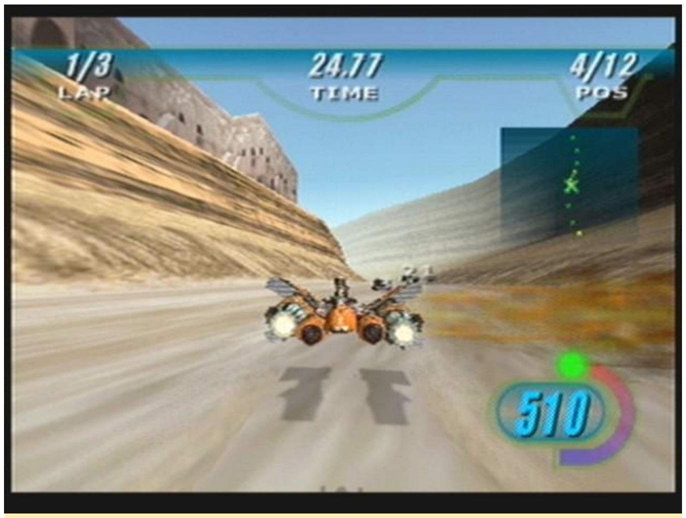


Figure 5 – Star Wars Episode 1 – Racer

U3

The experience on the U3 is very good. The game runs fluently and quickly, and doesn't seem to have glitches. Some of the shadows are too dark, but that's something you only experience in the menu.

C1 – rice plugin

Once again, the C1 has issues with this game related to the rice video plugin, since the same issues happen on the U3 when the video plugin is switched to rice. The picture was distorted and cut off in some scenes. The game works perfectly fine using glide64mk2 at full speed with no issues.

XU3

The game works very well on the XU3. I finally figured out how to use the booster, and I also saw a two player option. It seems that if the game finds more than one controller connected, it offers a multiplayer option. The gaming experience was flawless and at full speed.

Star Wars: Rogue Squadron

This is named as one of the best N64 games ever made, where you fly an X-Wing to right against the evil Empire. I played the game on the PC when it came out, and it was quite fun. I was looking forward to trying it on the ODROID. I've read that this game requires the memory expansion pack in order to launch. However, no matter what I tried, I wasn't able to get this game to work on any platform or with any graphics plugin. Both the mupen64plus and libretro core emulators either crashed or stopped responding.

Picture 6 – Star Wars: Rogue Squadron

StarCraft 64

StarCraft is a very famous RTS game. It's one of the best strategy games ever made, and is still played in professional gaming tournaments. The Nintendo 64 game is a very good remake with reduced graphics, stripped videos, and minimal music. It's a nice strategy game, and I found it interesting that I was able to play it on a Nintendo 64 emulator.

Picture 7 – StarCraft 64

U3

The game runs surprisingly well on the U3. There are some speed issues on the menu, but as soon as you are in the game, it works well, although the sound is a little bit delayed, especially in bigger battles. You can hear units die after they have already disappeared from the screen.

C1 – rice plugin

StarCraft 64 ran surprisingly well on the C1. It seems to work best using the rice video plugin. However, when using the glide64mk2 plugin, the menu is so slow that you can't select the mission that you want to play. Therefore, the game is not playable under glide64mk2. The in-game speed would probably be fine, but since I couldn't get past the menu, there is no way to tell.

XU3

I actually had a lot of issues getting StarCraft 64 to run on the XU3. The game was very laggy at first, and switching from glide64 to rice or gln64 exhibited strange issues. Rice and gln64 were really fast on the XU3 menu, and everything was full speed. But both rice and gln64 had major graphical problems, which made the game unplayable. After some investigation on the slowdown of glide64, I found out that reducing the rendering resolution increased the speed. The game is displayed in 1080p no matter which resolution you choose, but the resolution at which the characters and objects are rendered can be changed on the XU3. I found that using a resolution of 800×600 or below gave the best performance.

Super Mario 64

Super Mario 64 was the launch title for the N64, and what a launch title it was! This game boosted the N64 to the top of its class by showing what the console was capable of, and once again, made Mario the star of the Nintendo franchise.

Figure 8 – Super Mario 64

U3

On the U3, Mario 64 has some glitches with shadows, textures and lighting, but besides that, the game runs at full speed.

C1 – rice plugin

Mario 64 seems to be running a little below full speed on the C1, but it is still playable with the rice graphics plugin. The speed is slightly better with glide64mk2 than with the rice plugin, but it occasionally drops below full speed. It also has the same issues as the U3 glide64mk2 plugin with rendering ground textures and shadows.

XU3

The game is running fine on the XU3, with no issues or glitches.

Super Smash Bros

This game introduced a new genre of brawler games. It was a major success on the N64, and led to a lot of sequels. You can choose between famous Nintendo characters such as Mario, Yoshi, Princess Peach and many more, and fight against other characters.

Figure 9 – Super Smash Bros

U3

The gaming experience for Super Smash Bros on the U3 with mupen64plus and glide64mk2 plugin is very nice. Even the menu is working at a decent speed. There are some glitches with shadow and text, but nothing serious, and only the text issue is noticeable.

C1 – rice plugin

The game was too slow under rice to be playable. The menu, introduction, and gameplay were laggy. However, Super Smash Bros runs much better with the glide64mk2 plugin, and you can actually play it full speed, although it has the same glitches as the U3 version.

XU3

While in the menu, there is some lagging and slow downs, but the game runs perfectly fine otherwise. It was really fun to play.

The Legend of Zelda: Majora's Mask

I don't know much about the Legend of Zelda games on the N64, but I do know that this game involves having 72 hours to save the world, and you have different masks to help you in your cause. You can use the "Ocarina of Time" to travel back in time and start the 72 hours over and over again until you finished the game.

Figure 10 – The Legend of Zelda: Majora's Mask

U3

Although the game speed is very good, the glide64mk2 plugin once again has issues with being too dark. Since it can't do the blurry effect, the game stays at full speed the entire time. However, because it's too dark, it's sometimes hard to find a way, but it's not as dark as it is when played on the XU3, where nothing is visible. I consider this fully playable.

C1 – rice plugin

The game worked surprisingly well on the ODROID-C1 with the rice plugin. There were no graphical issues, but the introduction and some scenes were slightly laggy. Overall, the game is very playable on C1 with the rice plugin.

C1 – glide64mk2

The game runs at nearly full speed, but suffers from the same darkness issue as the U3. Rice is probably the best plugin for use with this game when played on the ODROID-C1.

XU3

The overall experience of the game is quite good. When there are cutscenes with the blurring effect, the game slows down and becomes laggy. However, since only occurs in cutscenes, the gameplay is fine. However, there's another issue which is related to glide plugin, which is that the graphics are too dark, makes it hard to figure out which way to go. It got so dark that I switched to the gln64 plugin, which had some minor glitches with the ground, but otherwise worked perfectly at full speed. It was not so dark that you couldn't see where to go, so using gln64 as a plugin for this game worked great.

The Legend of Zelda: Ocarina of Time

This is the predecessor of Majora's Mask. I actually had a hard time enjoying the game, but I know that it is supposed to get better over time, and there must be a reason why so many have it on their top 10 list, so I gave it a try.

Figure 11 – The Legend of Zelda: Ocarina of Time

U3

Generally the game works fine and is at full speed, with some minor issues with shadows and ground textures. In some places, it is too dark, but it is still fully playable.

C1 – rice plugin

Similar to the other Legend of Zelda game, this one works very nicely on the ODROID-C1 using the rice plugin. With the glide64mk2 plugin, the game was not entirely full speed, and exhibits the typical ground texture and shadow issues.

XU3

The experience on the XU3 is superb. I didn't see any glitches or slow downs so far, although I didn't get very far in the game. It's a really nice experience.

High Resolution Textures

After trying out different games, I checked on what else could be done with the emulators, and I found out that there are some high resolution texture packs that offer much better graphics. I tried a few of them to see what they look like in order determine if they would work on the ODROIDs. Mupen64plus standalone emulator offers the possibility to use high resolution textures for N64 games which can improve gaming experience by giving a new look to the games, but this option is not available for other emulators.

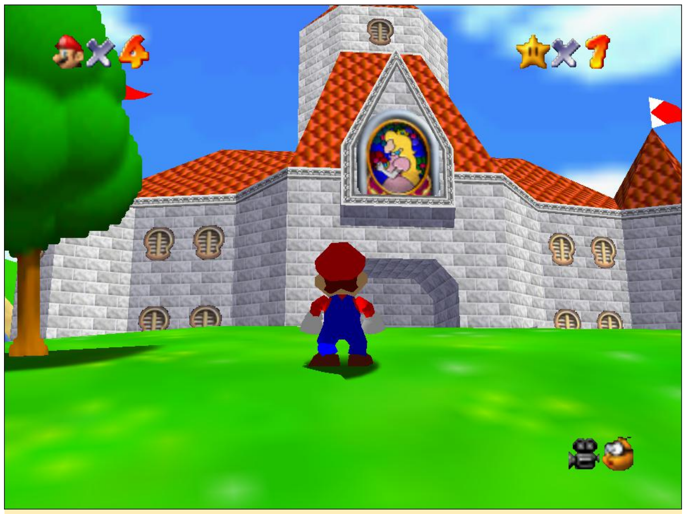


Figure 12 – Super Mario 64 with standard textures

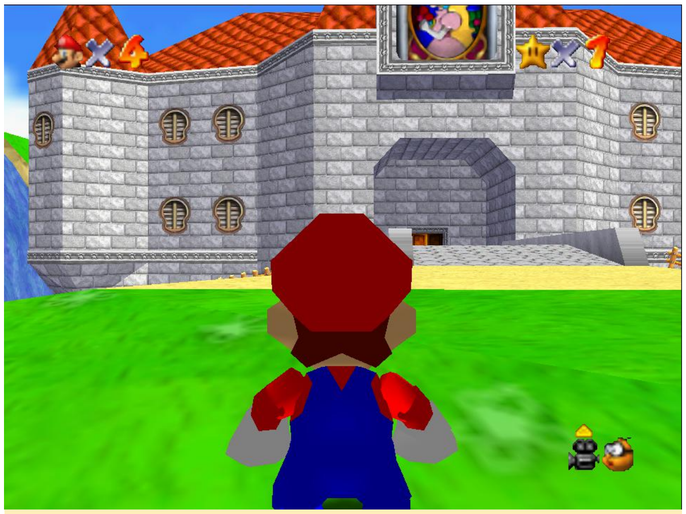


Figure 13 – Super Mario 64 with high resolution textures

To use the high resolution textures, download them from http://bit.ly/1Jvpahr and copy them to the directory ~/.local/share/mupen64plus/hires_texture/. Some of the textures are complete rewrites of the game graphics. Make sure to place the textures in a folder with the "short name" of the game in capital letters. For example, Mario 64 is "SUPER MARIO 64", and Mario Kart 64 is "MARIOKART64".

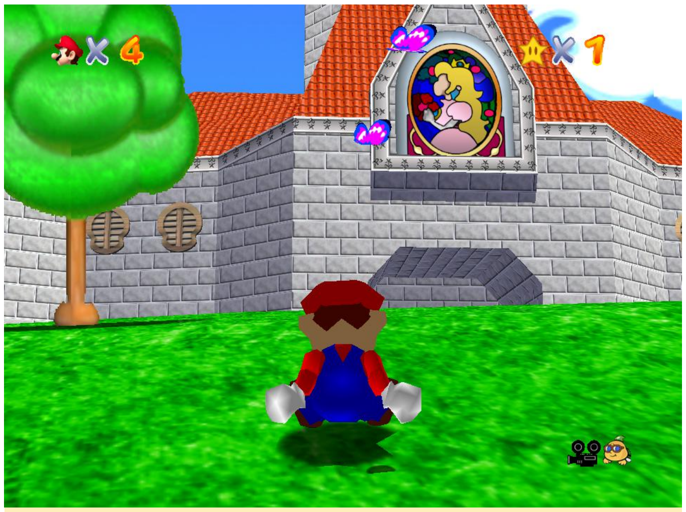


Figure 14 – Complete remake of the Mario 64 textures

Conclusion

Nintendo 64 emulation is generally working very well on ODROID devices, especially on the U3 and XU3. The C1 has a lot of issues which prevent it from offering the same gaming experience as on the other ODROID devices. The rice plugin, which works without having to change color depth settings on your image, has major issues with many games, but does a rather good job on other games. The glide64mk2 plugin only works under 16-bit, and although most games are running nicely, the ones that do run better with the rice plugin require a reboot in order to be able to use it, since rice isn't working with 16-bit. This leaves me rather unsatisfied, since I always had to reboot the entire ODROID in order to switch between different graphics plugins on the C1. The U3 and XU3 can do this without rebooting the entire system, which makes it much easier to switch between the plugins. Also, using 16-bit color depth prevents different applications such as XBMC from running properly, which causes you to choose a emulator frontend that actually supports 16-bit mode, or else you are forced to start N64 games through a Terminal window. This all makes me believe that C1 is not really suitable for N64, at least under Linux. I think that the best way to play N64 games on the C1 is probably through the Android app or a highly modified version using fbdev drivers and some scripts that are able to switch color depths and applications to run. That setup would be very inconvenient and certainly not suitable for beginners. The U3 and XU3 both measure up very well when it comes to N64 emulation. Being able to switch between graphic cores easily is a big benefit over the C1. N64 games seem to need some occasional tweaking, and if you look at the configuration options for either glide64mk2 or rice on the mupen64plus standalone emulator, there are a lot of options to choose from. The XU3 is the only board that can use libretro core of mupen64plus with Retroarch at the moment. It integrates the controllers very nicely, and you can easily adapt your gamepad layout to your own needs and have various controllers supported. Also, the XU3 has extra CPU power, which often make the difference between full speed or "nearly" full speed. The U3 does a very good job in emulating N64 games, and being able to use high resolution textures in mupen64plus is really a cool thing to have.