Shinobi CCTV • Home Automation • HID Gadget • GPU Mining • Gaming

ODROID Magazine

Year Five
Issue #52
Apr 2018

PORTABLE Sound STUDIO

RECORD MUSIC WITH AN ODROID-XU4Q ANYTIME, ANYWHERE

NEXTCLOUD SERVER:

CREATING A NETWORK ACCESS STORAGE (NAS) WITH AN ODROID-HC2

ARCADE BOX ON ODROID:

TAKING ADVANTAGE OF THE ARM BOARD WITH THE BEST GPU FOR AMAZING GAMES

HID Gadget Device: Using The ODROID-C2

April 1, 2018

The following guide describes how to setup the ODROID-C2 as a HID gadget device

Shinobi Closed Circuit TV (CCTV): Creating a Video Monitoring System Using the ODROID-HC2

April 1, 2018

It is my pleasure to share my experience creating a home CCTV video monitoring system using the ODROID-HC2 and Shinobi CCTV software. Hopefully this helps someone out there who is looking to have a video monitoring system of their own. In my opinion, traditional home alarm systems generally just cause ▶

Portable Sound Studio: Record Music with an ODROID-XU4Q Anytime, Anywhere

April 1, 2018

There is a lack of inexpensive hardware systems in today's market that can handle the variety of live recordings that make up the majority of my business. However, I have to gut my studio and re-patch my PA system to take it on the road. I began wondering if I ▶

Linux Gaming: Saturn Games – Part 3

April 1, 2018

We are back with this month's look at Sega Saturn games for the ODROID-XU3/XU4. Last time I talked about a lot of shoot 'em ups, but this article has a good mix of different genres, although I did pick quite a few "mecha" games I enjoy. Since this is the ▶

Arcade Box

April 1, 2018

We made our own arcade box with simple GPIO buttons and joysticks, and called it the ODROID Arcade Box

Nextcloud Server: Creating a Network Access Storage (NAS) with an ODROID-HC2

April 1, 2018

This guide will walk you through setting up a NAS (Network Attached Storage) on the ODROID-HC2 single-board computer. The guide is written for those with no Linux experience and minimal computer building experience. And it contains the content involved reading a few dozen guides.

Control Any Electrical Device With An ODROID-C2: A Sample Project

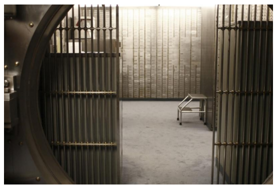
April 1, 2018

We will present you with a way to control almost any electrical device with a single click from any other device that has access to the web

Prospectors, Miners and 49er's – Part 2: Dual GPU-CPU Mining on the XU4/MC1/HC1/HC2

April 1, 2018

Here is brief summary of the kernels and crypto algorithms that were modified for the Odroid and the test results.



Secure Storage: Creating an Encrypted File System in Linux

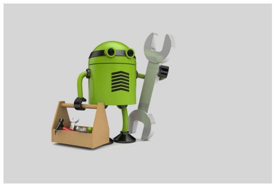
April 1, 2018

Learn how to encrypt your file system on an ODROID using LUKS as the key setup.

Tvheadend

April 1, 2018

The following instructions show how to compile the TVH code for the ODROID-C2



Android Development: So, You Want to Be an App Developer?

April 1, 2018

So, you want to be an app developer? Let's see what I can do to help you with that! A casual survey of the back issues of ODROID Magazine has shown that there are plenty of in-depth, detailed articles about Android on ODROID. What the community has been missing is ▶

Setting Up Your ODROID: ODROID-XU4 As A General Purpose NAS

April 1, 2018

I wanted my ODROID-XU4 to do much more than being a plain old NAS, and I wanted to keep using Ubuntu in order to benefit from newer packages. This presents an opportunity to gain new knowledge.

Meet An ODROIDian: Ernst Mayer, Mathematician Extraordinaire

April 1, 2018

I've been living and working in Silicon Valley for roughly the last 20 years, doing algorithmic and coding work first for several tech startups and then for larger firms. Most of that work was related to EDA software for chip design. Last year, the 4-5 months after buying my C2 ▶

HID Gadget Device: Using The ODROID-C2

April 1, 2018 By @Ashren0 Linux, ODROID-C2, Tutorial

The following guide describes how to setup the ODROID-C2 as a HID gadget device, in this case it will be used as either a keyboard or a basic gamepad.

The following steps could be adapted for any another device that

- Support for USB OTG device mode
- Has a Linux kernel at above version 3.19 with FunctionFS HID module built, we will use Linux version 4.15.

Before we begin, it should be noted that every command has to be run as root.

Figure 1 – The ODROID-C2 can act as an HID device using USB OTG device mode

Preparation

The easiest method for running a recent kernel, for now, is to use the ArchLinuxARM image at https://archlinuxarm.org/platforms/armv8/amlogic/odroid-c2. After following the instructions and having botted into the default installation, update the system to use the mainline kernel (4.15 at the time of writing):

```
$ sudo pacman -Syu
$ sudo pacman -R uboot-odroid-c2
$ sudo pacman -S uboot-odroid-c2-mainline
linux-aarch64 dtc
```

Make sure NOT to reboot the device yet if you are booting from eMMC. The default mainline installation above has some quirky defaults that render the system read-only and disable the OTG module. We must first disassemble the Device Tree Blob, or DTB, file used by the C2 for initializing onboard peripherals into its source form the DTS

```
$ cd /boot/dtbs/amlogic/
$ sudo cp -p meson-gxbb-odroidc2{,_backup}.dtb
$ sudo dtc -I dtb -O dts meson-gxbb-
odroidc2.dtb > meson-gxbb-odroidc2.dts
```

Once done, edit the source:

```
$ sudo nano meson-gxbb-odroidc2.dts
```

In the section [mmc@74000], change the following line to re-enable write access to the eMMC (even if not using eMMC now, it’s always good to change it):

```
max-frequency = <0xbebc200>;
```

to:

```
max-frequency = <0x8f0d180>;
```

And in the section [usb@c9000000], change:

```
dr_mode = "host";
```

to:

```
dr_mode = "peripheral";
```

usb@c9000000 is the OTG peripheral which is in “host” mode by default here. Be careful not to touch anything regarding usb@c910000 which is the 4-ports USB hub. Once this is done, we can rebuild the DTB file and reboot:

```
$ sudo dtc -I dts -O dtb meson-gxbb-
odroidc2.dts > meson-gxbb-odroidc2.dtb
$ sudo reboot
```

We can check the running kernel with the following command:

```
$ uname -r
```

It should be 4.15+ and if everything worked fine, the following command should show a symbolic link [c9000000.usb]:

```
$ ls /sys/class/udc/
```

Configuration

Consider the following python3 script that performs setup and teardown of the device automatically:

```
import sys
import os
import shutil
import pwd
import asyncio
import subprocess
import argparse
import atexit

class HIDReportDescriptorKeyboard(object):
def __len__(self):
return 8

def __bytes__(self):
return bytes([
 0x05, 0x01, # Usage Page (Generic Desktop
Ctrls)
 0x09, 0x06, # Usage (Keyboard)
 0xA1, 0x01, # Collection (Application)
 0x05, 0x07, # Usage Page (Kbrd/Keypad)
 0x19, 0xE0, # Usage Minimum (0xE0)
 0x29, 0xE7, # Usage Maximum (0xE7)
 0x15, 0x00, # Logical Minimum (0)
 0x25, 0x01, # Logical Maximum (1)
 0x75, 0x01, # Report Size (1)
 0x95, 0x08, # Report Count (8)
 0x81, 0x02, # Input (Data,Var,Abs,No
Wrap,Linear,Preferred State,No Null Position)
 0x95, 0x01, # Report Count (1)
 0x75, 0x08, # Report Size (8)
 0x81, 0x03, # Input (Const,Var,Abs,No
Wrap,Linear,Preferred State,No Null Position)
 0x95, 0x05, # Report Count (5)
 0x75, 0x01, # Report Size (1)
 0x05, 0x08, # Usage Page (LEDs)
 0x19, 0x01, # Usage Minimum (Num Lock)
 0x29, 0x05, # Usage Maximum (Kana)
 0x91, 0x02, # Output (Data,Var,Abs)
 0x95, 0x01, # Report Count (1)
 0x75, 0x03, # Report Size (3)
 0x91, 0x03, # Output (Const,Var,Abs)
 0x95, 0x06, # Report Count (6)
 0x75, 0x08, # Report Size (8)
 0x15, 0x00, # Logical Minimum (0)
 0x25, 0x65, # Logical Maximum (101)
 0x05, 0x07, # Usage Page (Kbrd/Keypad)
 0x19, 0x00, # Usage Minimum (0x00)
 0x29, 0x65, # Usage Maximum (0x65)
 0x81, 0x00, # Input (Data,Array,Abs,No
Wrap,Linear,Preferred State,No Null Position)
 0xC0, # End Collection
])

class HIDReportDescriptorGamepad(object):
def __len__(self):
return 4

def __bytes__(self):
return bytes([
 0x05, 0x01, # USAGE_PAGE (Generic Desktop)
 0x15, 0x00, # LOGICAL_MINIMUM (0)
 0x09, 0x04, # USAGE (Joystick)
 0xa1, 0x01, # COLLECTION (Application)
 0x05, 0x02, # USAGE_PAGE (Simulation
Controls)
 0x09, 0xbb, # USAGE (Throttle)
 0x15, 0x81, # LOGICAL_MINIMUM (-127)
 0x25, 0x7f, # LOGICAL_MAXIMUM (127)
 0x75, 0x08, # REPORT_SIZE (8)
 0x95, 0x01, # REPORT_COUNT (1)
 0x81, 0x02, # INPUT (Data,Var,Abs)
 0x05, 0x01, # USAGE_PAGE (Generic Desktop)
 0x09, 0x01, # USAGE (Pointer)
 0xa1, 0x00, # COLLECTION (Physical)
 0x09, 0x30, # USAGE (X)
 0x09, 0x31, # USAGE (Y)
 0x95, 0x02, # REPORT_COUNT (2)
 0x81, 0x02, # INPUT (Data,Var,Abs)
 0xc0, # END_COLLECTION
 0x09, 0x39, # USAGE (Hat switch)
 0x15, 0x00, # LOGICAL_MINIMUM (0)
 0x25, 0x03, # LOGICAL_MAXIMUM (3)
 0x35, 0x00, # PHYSICAL_MINIMUM (0)
 0x46, 0x0e, 0x01, # PHYSICAL_MAXIMUM (270)
 0x65, 0x14, # UNIT (Eng Rot:Angular Pos)
 0x75, 0x04, # REPORT_SIZE (4)
 0x95, 0x01, # REPORT_COUNT (1)
 0x81, 0x02, # INPUT (Data,Var,Abs)
 0x05, 0x09, # USAGE_PAGE (Button)
 0x19, 0x01, # USAGE_MINIMUM (Button 1)
 0x29, 0x04, # USAGE_MAXIMUM (Button 4)
 0x15, 0x00, # LOGICAL_MINIMUM (0)
 0x25, 0x01, # LOGICAL_MAXIMUM (1)
 0x75, 0x01, # REPORT_SIZE (1)
 0x95, 0x04, # REPORT_COUNT (4)
 0x55, 0x00, # UNIT_EXPONENT (0)
 0x65, 0x00, # UNIT (None)
 0x81, 0x02, # INPUT (Data,Var,Abs)
 0xc0 # END_COLLECTION
])

class HidDaemon(object):
 def __init__(self, vendor_id, product_id,
manufacturer, description, serial_number,
hid_report_class):
 self._descriptor = hid_report_class()
 self._hid_devname = 'odroidc2_hid'
 self._vendor = vendor_id
 self._product = product_id
 self._manufacturer = manufacturer
 self._desc = description
 self._serial = serial_number
 self._libcomposite_already_running =
self.check_libcomposite()
 self._usb_f_hid_already_running =
self.check_usb_f_hid()
 self._loop = asyncio.get_event_loop()
 self._devname = 'hidg0'
 self._devpath = '/dev/%s' % self._devname

def _cleanup(self):
 udc_path =
'/sys/kernel/config/usb_gadget/%s/UDC' %
self._hid_devname
 if os.path.exists(udc_path):
 with open(udc_path, 'w') as fd:
 fd.truncate()
 try:
 shutil.rmtree('/sys/kernel/config/usb_gadget/
%s' % self._hid_devname, ignore_errors=True)
 except:
 pass
 if not self._usb_f_hid_already_running and
self.check_usb_f_hid():
 self.unload_usb_f_hid()
 if not self._libcomposite_already_running and
self.check_libcomposite():
 self.unload_libcomposite()

@staticmethod
def check_libcomposite():
 r = int(subprocess.check_output("lsmod | grep
'libcomposite' | wc -l", shell=True,
close_fds=True).decode().strip())
 return r != 0

@staticmethod
def load_libcomposite():
 if not HidDaemon.check_libcomposite():
```

```
 subprocess.check_call("modprobe
libcomposite", shell=True, close_fds=True)

@staticmethod
def unload_libcomposite():
 if HidDaemon.check_libcomposite():
 subprocess.check_call("rmmod libcomposite",
shell=True, close_fds=True)

@staticmethod
def check_usb_f_hid():
 r = int(
 subprocess.check_output("lsmod | grep
'usb_f_hid' | wc -l", shell=True,
close_fds=True).decode().strip())
 return r != 0

@staticmethod
def load_usb_f_hid():
 if not HidDaemon.check_libcomposite():
 subprocess.check_call("modprobe usb_f_hid",
shell=True, close_fds=True)

@staticmethod
def unload_usb_f_hid():
 if HidDaemon.check_libcomposite():
 subprocess.check_call("rmmod usb_f_hid",
shell=True, close_fds=True)

def _setup(self):
f_dev_name = self._hid_devname
os.makedirs('/sys/kernel/config/usb_gadget/%s/
strings/0x409' % f_dev_name, exist_ok=True)
os.makedirs('/sys/kernel/config/usb_gadget/%s/
configs/c.1/strings/0x409' % f_dev_name,
exist_ok=True)
os.makedirs('/sys/kernel/config/usb_gadget/%s/
functions/hid.usb0' % f_dev_name,
exist_ok=True)
with
open('/sys/kernel/config/usb_gadget/%s/idVendo
r' % f_dev_name, 'w') as fd:
fd.write('0x%04x' % self._vendor)
with
open('/sys/kernel/config/usb_gadget/%s/idProdu
ct' % f_dev_name, 'w') as fd:
fd.write('0x%04x' % self._product)
with
open('/sys/kernel/config/usb_gadget/%s/bcdDevi
ce' % f_dev_name, 'w') as fd:
fd.write('0x0100')
with
```

```
open('/sys/kernel/config/usb_gadget/%s/bcdUSB'
% f_dev_name, 'w') as fd:
fd.write('0x0200')

with
open('/sys/kernel/config/usb_gadget/%s/strings
/0x409/serialnumber' % f_dev_name, 'w') as fd:
fd.write(self._serial)
with
open('/sys/kernel/config/usb_gadget/%s/strings
/0x409/manufacturer' % f_dev_name, 'w') as fd:
fd.write(self._manufacturer)
with
open('/sys/kernel/config/usb_gadget/%s/strings
/0x409/product' % f_dev_name, 'w') as fd:
fd.write(self._desc)

with
open('/sys/kernel/config/usb_gadget/%s/configs
/c.1/strings/0x409/configuration' %
f_dev_name, 'w') as fd:
fd.write('Config 1 : %s' % self._desc)
with
open('/sys/kernel/config/usb_gadget/%s/configs
/c.1/MaxPower' % f_dev_name,'w') as fd:
fd.write('250')

with
open('/sys/kernel/config/usb_gadget/%s/functio
ns/hid.usb0/protocol' % f_dev_name, 'w') as
fd:
fd.write('1')
with
open('/sys/kernel/config/usb_gadget/%s/functio
ns/hid.usb0/subclass' % f_dev_name, 'w') as
fd:
fd.write('1')
with
open('/sys/kernel/config/usb_gadget/%s/functio
ns/hid.usb0/report_length' % f_dev_name, 'w')
as fd:
fd.write(str(len(self._descriptor)))
with
open('/sys/kernel/config/usb_gadget/%s/functio
ns/hid.usb0/report_desc' % f_dev_name, 'wb')
as fd:
fd.write(bytes(self._descriptor))

os.symlink(
'/sys/kernel/config/usb_gadget/%s/functions/hi
d.usb0' % f_dev_name,
'/sys/kernel/config/usb_gadget/%s/configs/c.1/
hid.usb0' % f_dev_name,
target_is_directory=True
)

with
open('/sys/kernel/config/usb_gadget/%s/UDC' %
f_dev_name, 'w') as fd: fd.write('
'.join(os.listdir('/sys/class/udc')))

def run(self):
if not self._libcomposite_already_running:
self.load_libcomposite()
atexit.register(self._cleanup)

# Setup HID gadget (keyboard)
self._setup()

# Use asyncio because we can then do thing on
the side (web ui, polling attached devices
using pyusb ...)
try:
self._loop.run_forever()
except KeyboardInterrupt:
pass

if __name__ == '__main__':
user_root = pwd.getpwuid(0)
user_curr = pwd.getpwuid(os.getuid())
print('Running as <%s>' % user_curr.pw_name)
if os.getuid() != 0:
print('Attempting to run as ')
sys.exit(os.system("/usr/bin/sudo /usr/bin/su
root -c '%s %s'" % (sys.executable, '
'.join(sys.argv))))
parser = argparse.ArgumentParser()
parser.add_argument('hid_type', choices=
['keyboard', 'gamepad'])
args = parser.parse_args()
if args.hid_type == 'keyboard':
print('Emulating: Keyboard')
# Generic keyboard
hid = HidDaemon(0x16c0, 0x0488, 'author',
'ODROID C2 KBD HID', 'fedcba9876543210',
HIDReportDescriptorKeyboard)
hid.run()
elif args.hid_type == 'gamepad':
print('Emulating: Gamepad')
# Teensy FlightSim for the purpose of this
example (and since it's intended for DIY, it
fits ou purpose)
hid = HidDaemon(0x16c0, 0x0488, 'author',
'ODROID C2 GAMEPAD HID', 'fedcba9876543210',
HIDReportDescriptorGamepad)
hid.run()
```

The classes HIDReportDescriptorKeyboard and HIDReportDescriptorGamepad are where we describe our device such as its type, buttons, and axis count. Note that the vendorId and productId are also important since even if you describe your device as a gamepad, if the VID/PID are those of a keyboard, the operating system will most likely identity it as a keyboard.

Next, run the following command, which requires root privileges, in order to create a /dev/hidg0 device to which you can freely write:

```
$ sudo python3 script.py keyboard
```

or

```
$ sudo python3 script.py gamepad
```

We can then test it with the keyboard argument:

```
$ sudo sleep 5 && echo -ne "" > /dev/hidg0 &&
echo -ne "" > /dev/hidg0
```

This command will write “A” (or “Q” if your use an azerty layout), after 5 seconds. Now, to test it with the gamepad argument use:

```
$ sudo sleep 5 && echo -ne "�" > /dev/hidg0
&& echo -ne "" > /dev/hidg0
```

This will trigger the fourth button on the gamepad device.

Making different devices

The two examples use very basic descriptors which use the following bit format when writing to /dev/hidg0:

Keyboard (6-rollover)

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8
Modifiers	Reserved	Key 1	Key 2	Key 3	Key 4	Key 5	Key 6

Gamepad

BYTE 1								BYTE 2							
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8

Throttle (-127 to 127)								X-Axis (-127 to 127)							

								BYTE 4							
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Y-Axis (-127 to 127)								B 4	B 3	B 2	B 1	(1)		(2)	

(1)When equals to 0b11 the HAT buttons are all set to non active, which should be a default. (2)HAT buttons bit-mask, given that bits 5-6 are set to 0 (a)0b00 => UP (b)0b01 => RIGHT (c)0b10 => DOWN (d)0b11 => LEFT

Conclusion

This is just a very basic example but it shows the options available to us. Delving deeper into the HID and USB specifications should make for plenty of use cases:

- Simulating a device, such as a specific keyboard, gamepad, mouse for development purposes.
- Making a device appear as another. For legacy devices.
- Pure USB debugging / reverse engineering (USBProxy project at https://github.com/dominicgs/USBProxy).
- Penetration testing.
- And surely many others I didn't think of.

For comments, questions, and suggestions about the HID gadget, please visit the original thread at https://forum.odroid.com/viewtopic.php?f=139&t=30267.

Sources

USB HID 1.1 specs http://www.usb.org/developers/hidpage/HID1_11.pdf

Where if found the key scancodes for the keyboard https://gist.github.com/MightyPork/6da26e382a7ad91b5496ee55fdc73db2

To better understand report descriptors https://hamaluik.com/posts/making-a-custom-teensy3-hid-joystick/ and http://eleccelerator.com/tutorial-about-usb-hid-report-descriptors/

Tool for making descriptors (pretty clunky but official) http://www.usb.org/developers/hidpage#HID%20Descriptor%20Tool

Keyboard report descriptor http://isticktoit.net/?p=1383

Utility to check report descriptors http://eleccelerator.com/usbdescreqparser/

OTG peripheral DTB modification https://community.nxp.com/thread/383191

C2 mainline kernel support (frequency tweaks) https://forum.odroid.com/viewtopic.php?f=135&t=22717

Shinobi Closed Circuit TV (CCTV): Creating a Video Monitoring System Using the ODROID-HC2

April 1, 2018 By Dylan Tutorial, ODROID-HC2

It is my pleasure to share my experience creating a home CCTV video monitoring system using the **ODROID-HC2** and Shinobi CCTV software. Hopefully this helps someone out there who is looking to have a video monitoring system of their own.

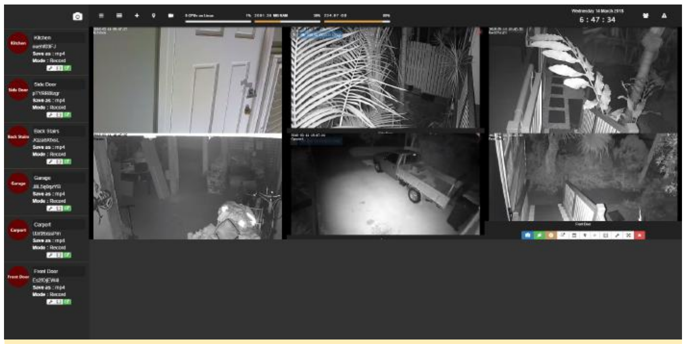


Figure 1 – Shinobi CCTV Dashboard

In my opinion, traditional home alarm systems generally just cause a nuisance for neighbours, and they must be armed to be effective. In comparison, a CCTV system is quiet, but also provides records, which in my experience is a more effective deterrent to potential criminals. A good CCTV system records 24/7 and flags motion events.

Other benefits of a CCTV system include, its usefulness in telling you when packages have been delivered to the front door; lets you know who is at the door; and monitor the BBQ smoker outside while playing video games inside.

Warnings and Disclaimer

To honor others' privacy, install cameras with only a view of your property, and not that of the neighbour's. It is also wise to review your local rules to understand any additional requirements covering signage, restrictions on audio recording, and such. Although I only recommend installing the cameras outside your home, one exception may be to use a camera internally as a baby monitor. However, as one of my friends found, it is important to make sure your access/security to this camera is watertight!

System fundamentals

Structured Cabling in the form of wired CAT6 or better will provide robust, secure communications for your CCTV system. I installed 20 ethernet outlets throughout my home with eight in the ceiling void for the purpose of connecting cameras. Structured cabling is normally something installed by an electrician, but with patience and research you can safely do this yourself. Numerous online resources are available to research the topic.

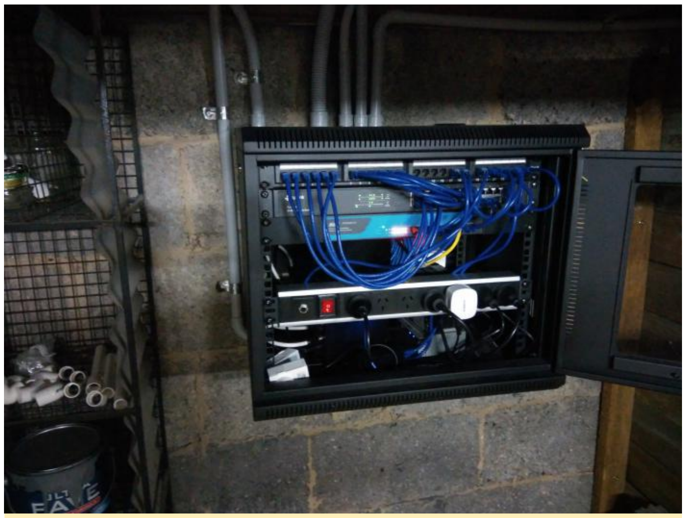


Figure 2 – Structured Cabling

Cameras are of two types: analog or digital. Analog cameras need more hardware to operate on a network based system. For video monitoring, I prefer digital cameras as they connect using ethernet, offer flexible configuration and generally cost the same if not less. I chose cameras powered over ethernet (PoE) which means I can provide power and communications over CAT6. The open standard for network cameras is ONVIF (https://www.onvif.org). I recommend ONVIF compatible cameras, such as Foscam FI9853EP, to mitigate compatibility issues.

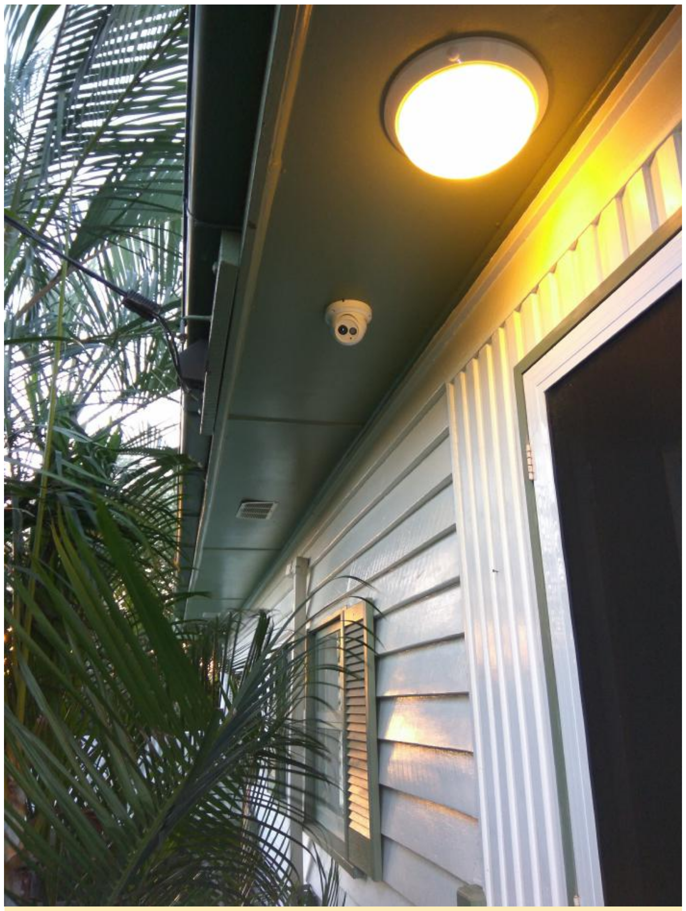


Figure 3 – Example Camera Install

ODROID-HC2 Setup

The ODROID-HC2 is perfectly suited for my CCTV video monitoring system. It offers gigabit network speed, 3.5" or 2.5" HDD/SDD native SATA interface and no unnecessary bells or whistles. I want to store footage for at least 30 days using inexpensive hard drive storage media.

I Initialised the ODROID-HC2 using the latest ODROID-XU4 Ubuntu minimal OS Image. Flash the image onto a microSD using WinDiskImager, then insert into the ODROID along with an ethernet connection and 3.5" hard-drive. Power up with a compatible 12 volt PSU. It took about 2 minutes to initialise the ODROID and on the next boot it had SSH access over a DHCP assigned IP.

Apply any available updates:

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get dist-upgrade
```

Mount the hard-drive. The hard-drive should show up as /dev/sda but run the following command to check:

```
$ sudo fdisk -l
```

The following assumes the hard drive shows as /dev/sda. Create a single partition using fdisk. Reference fdisk documentation for partitioning the drive.

```
$ sudo fdisk /dev/sda
```

After you have a partition, format the partition:

```
$ sudo mkfs.ext4 /dev/sda
```

The ODROID-HC2 only has space for a single hard drive. As such, I just mount the disk referencing /dev/sda. You can mount using the UUID if you prefer. Add the following line to fstab:

```
$ sudo nano /etc/fstab
/dev/sda /media/CCTV ext4 defaults 0 2
```

The following script helps park the hard drive on shutdown. Download and install the script:

```
$ wget
https://dn.ODROID.com/5422/script/ODROID.shutd
own
$ sudo install -o root -g root -m 0755
./ODROID.shutdown /lib/systemd/system-
shutdown/ODROID.shutdown
```

I shared the entire hard drive using Samba. This gives me the option to retrieve recordings over a network share.

```
$ sudo apt-get install samba samba-common-bin
```

Next, configure Samba:

```
$ sudo nano /etc/samba/smb.conf

#===== Share Definitions =====
[CCTV]
comment = CCTV
path = /media/CCTV/
browsable = yes
writable = yes
guest ok = yes
read only = no
```

Restart the device and ensure you can read/write to the hard drive.

Shinobi Setup

Use the install script to install Shinobi CCTV and associated prerequisites:

```
$ sudo apt-get install curl

$ bash <(curl -s
https://raw.githubusercontent.com/ShinobiCCTV/
Shinobi-Installer/master/shinobi-install.sh)
```

Install all dependencies. I chose the Shinobi Pro branch with the remaining options as defaults. Access the Shinobi administrator view using accessing the link: https://[your-ODROID-HC2-ip]:8080/super on your browser.

Then, add a new account, add the hard-drive to the configuration and save it:

```
"addStorage": [
 {
 "name": "second",
 "path": "/media/sda/CCTV"
 }
```

You may also want to change the email settings, API keys and Shinobi superuser password.

Now login to the primary Shinobi CCTV interface in your browser at https://[your ODROID]:8080 using the new account. Click on the plus [+] symbol with a tooltip ‘Add Monitor’. Configure your camera as follows (your ideal settings may vary):

```
Identity
Mode: Record
Monitor ID: (leave default)
Name: (i.e. Front Door)
Retry Connection: 0
Storage Location: second

Input
Input Type: H264 (or that supported by your
camera)
Connection Type: RSTP
RTSP Transport: UDP
Username: (Camera login)
Password: (Camera password)
Host: (Camera IP)
Port: (Camera RTSP Port)
Force Port: No
Path: (RSTP Path)
```

```
Analyzation Duration: 100000
Probe Size: 100000
Accelerator: No

Stream
Stream Type: FLV
FLV Stream Type: Websocket
Max Latency: 20000
Video Encoder: Copy
Audio Encoder: No Audio
TV Channel: No

Stream Timestamp
Enabled: No

Stream Watermark
Enabled: No

JPEG API Snapshot (cgi-bin)
Enabled: No

Recording
Record File Type: MP4
Video Codec: Copy
Audio Codec: No Audio
Double Quote Directory: No
Recording Segment Interval: 15

Custom
(leave blank)

Logging
Log Level: (set to silent once camera is
stable)
Save Log in SQL: No
```

You will need to review your camera setup manual to establish options. The JPEG API should work for most cameras, but unfortunately not mine. I prefer UDP for camera streams. If you plan to stream over the internet or route through a busy LAN, choose TCP.

HTTP Live Streaming (HLS) is supposed to be superior to Flash Video (FLV) streaming but I could not make it display smoothly.

There are many more customizations for Shinobi. You can configure a motion detector, email alerts and customer scripting to name a few. I found the following Shinobi pages useful:

https://goo.gl/47M2Ty
https://shinobi.video/docs/cameras
https://goo.gl/kvKax1

You will need to setup port-forwarding or other methods to view Shinobi over the internet. I was able to view all streams smoothly at from a friend's place. Viewing live streams on a mobile phone however seems to be problematic but recording playback is flawless. I expect that mobile phone live view should improve with tweaking or future updates.

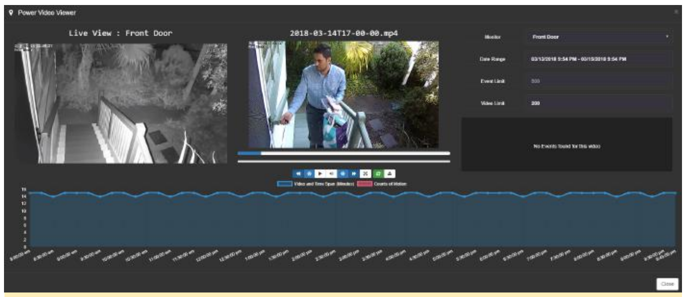


Figure 4 – Shinobi CCTV Power Viewer

I am capturing and recording 6 cameras at 720p 2M 15fps. My ODROID-HC2 sits between 1% to 5% CPU and 36% memory use. So far the system has run stable for 2 weeks with no signs of degrading. No crashes, reboots or other abnormal behaviour. It took a few days of trial and error to get the system running the way I like.

For comments, questions, and suggestions, please visit the original article at https://goo.gl/tJprLA.

Portable Sound Studio: Record Music with an ODROID-XU4Q Anytime, Anywhere

April 1, 2018 By Stephen Baldassarre Linux, ODROID-XU4

Stephen Baldassarre is a drummer working in two active bands who does video and film production. More importantly, he is also the engineer for Golden Clam Machine Studio in Boise, Idaho. Clients include Steve Schwarz of the metal band Pyrael and the non-profit organization Story Story Night. Stephen recently posted a Youtube video about his somewhat unusual recording rig, which is what piqued ODROID Magazine's attention about his setup.

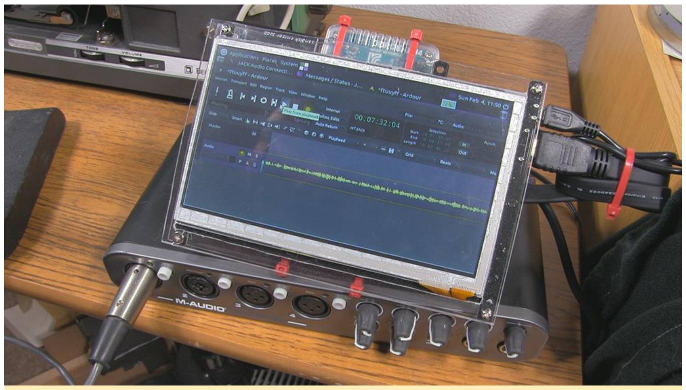


Figure 1 – An ODROID-XU4Q recording voice-over via USB interface

Hardware requirements

- ODROID-XU4Q
- 40mm 5V fan
- ODROID-VU7
- 5V/6A US Power Supply
- ODROID-XU4 clear case
- SanDisk 64GB MicroSD system drive

- SanDisk 64GB USB drive (for recording small track counts)
- USB3.0 to SATA Bridge Board Plus with SanDisk 240GB SSD (for recording large track counts)
- Plexiglass sheet
- 4 x small zip-ties to hold it all together

Software requirements

- Ubuntu (ubuntu-16.04.3-4.14-mate-odroid-xu4-20171212)
- JACK Audio Connection Kit – Qt GUI V0.4.2: 4/7/2016
- Ardour 4.6.0 "Evening Star"

Stephen, why did you make this?

I use computers for audio and video production on a daily basis but try to rely on dedicated solutions as much as possible. Unfortunately, there is a lack of inexpensive hardware systems in today's market that can handle the variety of live recordings that make up the majority of my business. I have been using the M-Audio Fast Track Ultra (https://goo.gl/bpx3Ly), an USB interface with six analogue inputs, with a laptop computer running Sony Vegas for smaller live recording projects like 'Story Story Night', which is largely speech with some live music in between story tellers. Every show is recorded, mixed and uploaded to various streaming services for all to hear. I do not like bringing my laptop to shows because there's a lot of things that can go wrong, including Microsoft Windows-related glitches or theft. For more demanding shows, like a 3-hour rock concert, I have been taking my HD24, which is much more reliable and has more audio inputs than the Fast Track. However, I have to gut my studio and re-patch my PA system to take it on the road. I began wondering if I could make a sort of modular system to cover all my live recording needs and maybe starting a cottage industry of making/selling them. I thought some kind of single-board computer might be able to handle this task. I could not find any examples on the internet of anybody doing it, so I just took a chance on the ODROID-XU4Q, due to its better specification.

What hardware did you use?

I sort of worked backwards on that. My colleague Steve Schwarz, an avid GNU/Linux fan, has been suggesting I try Ardour for years. Ardour 4 requires at least 2GB of RAM, so that eliminated most other SBCs. I liked the idea of a passive heat sink: quiet operation, less power, less chance of failure. I did not want to deal with a mouse and keyboard on-location, so a touch screen was the obvious solution. I chose the ODROID-VU7 as I wanted a somewhat larger screen and did not feel like the resolution was particularly important. I was not going to mix on it; I just wanted to record directly to an USB drive and later move the files to my studio computer or HD24 to start mixing. Obviously, an USB drive would not handle a lot of tracks, so I also got a USB3 to SATA adapter with 240GB SSD for larger shows.

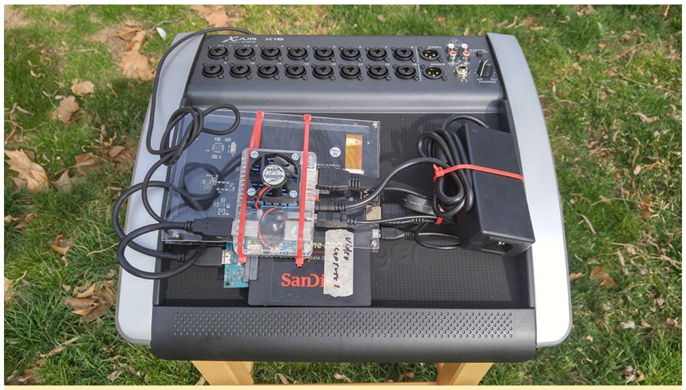


Figure 1 – Back of XU4Q with fan and SSD sandwiched between Plexi layers

How did you create the setup?

I was not sure what kind of power requirements I would have, so I opted for a 5V 6A power supply. Most Ardour users seem to use Ubuntu, so the OS was actually the last decision I made. I copied an Ubuntu 16.04 image to a MicroSD card I already had. I had some Plexiglass left over from an unrelated project, so I cut out a rectangle the size of the VU7 and used the supplied standoffs and screws to mount the Plexiglass to it. I drilled some holes in the back, the same size and spacing as the feet on the ODROID-XU4's case, then strapped the ODROID-XU4Q in place with a couple pairs of zip-ties. Channels were cut into the Plexiglass with a small file to keep the zip-ties from sliding. The ODROID-XU4Q is off-set so it acts as a stand for the screen, holding it at an angle if I set it on a table. It looks a little funny but it lets me take everything apart very quickly if needed.

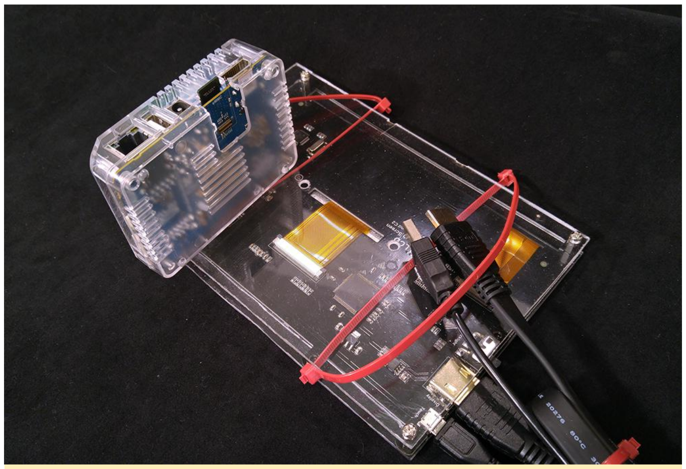


Figure 2 – Back of VU7 showing makeshift mounting plate

How do you use your portable studio?

Initially, the Fast Track lived in the back of my PA all the time, with various outputs from my audio mixer feeding into it. For most shows, I put the main voice on input-1, and all other voices on input-2. Instruments are sub-mixed to stereo and fed to 3-4 while audience mics are 5-6. Most of my clients work on extremely tight budgets. Like with SSN, I only have 2 ½ hours of clock time to turn a 2-hour show into two 45-minute podcasts. I have become rather adept at getting good mixes on the fly, but it is always best to keep the vocals separate. The audience mics, which are useless for the live sound, were fed directly to the interface.

Figure 3 – 400 guests find seats at "Story Story Night"

What I am usually doing off-late is using a Behringer X18 digital mixer with USB interface capabilities. That lets me record 6-track sub-mixes for small jobs but I can have up to 16 independent tracks if I want them, all without reconfiguring anything. I have also tested up to 26 tracks from an X32, a 32-channel digital mixer with mixed (no pun intended) results. Needless to say, I am glad I got in the habit of having a backup recording system at every show: https://www.youtube.com/watch?v=4TVOxfPE2ps.

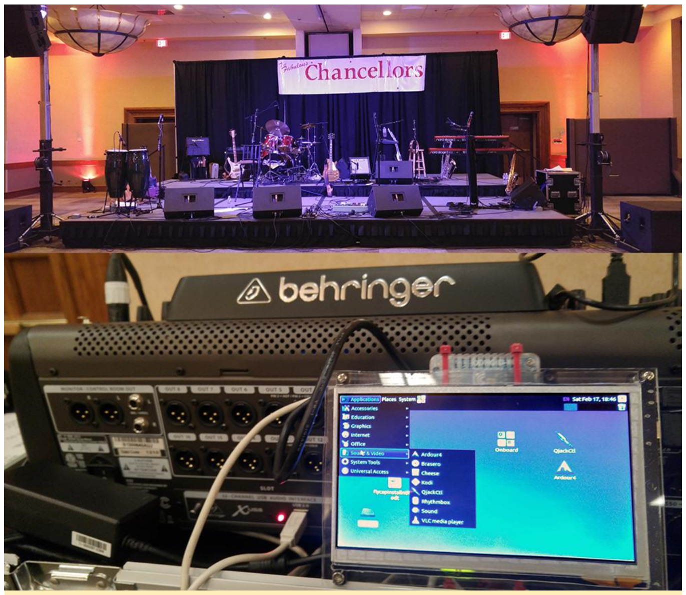


Figure 5 – The Fabulous Chancellors needs a lot of tracks but one tiny computer

What problems did you encounter during the build and how did you solve them?

I had no experience with SBCs or Linux before this project, so I could not have done it without the help of Steve Schwarz. He lives in New York, so getting advice required patience. Incidentally, I came up with this Ubuntu/Ardour recording system because of his suggestions and he put an HD24 and analogue console in his studio because of me. Anyway, I did not know that everything important happens through the terminal when I got started. The only command I really needed to know, though, was:

```
$ sudo apt-get install ardour
```

I am so glad that Ardour 4 was in the Ubuntu repository so it would just download and install along with all its dependencies. Getting it working properly took some experimentation. I discovered that Ardour did not like being installed if I updated the OS, so I left it as the stock "ubuntu-16.04.3-4.14-mate-odroid-xu4-20171212" install.

The ODROID-XU4Q has a tendency to overheat and crash when recording many tracks at a time. Adding a fan to the ODROID-XU4Q's oversized heat sink helps immensely. It is very sensitive to static electricity, so I need to figure out some kind of shielded housing. In the meantime, I have gotten in the habit of grounding myself before touching it during recording sessions.

With the Fast Track, the main outputs are disabled and the alternate outputs are drowned in reverb. It is possible to do overdubs (record more tracks later) but I do not recommend it. This seems to be an intentional handicap when not using M-Audio's proprietary drivers.

JACK is very important with the Fast Track because Ardour will not connect directly to the ALSA drivers in this case. JACK must be opened and started for Ardour to connect. Ardour will not start JACK automatically, and cannot be set to "real time" or Ardour will not open or create sessions. It's also important to make sure sample rates match before running Ardour. If using a digital mixer, Ardour must match the mixer's internal clock before connecting. JACK is not necessary with the Behringer digital mixers, and only the Fast Track Ultra, X18 and X32 were tested.

Since adding a fan, I have bench tested the ODROID-XU4Q recording 16 tracks for two hours straight without issue. Before adding the fan, the heatsink was hot to the touch after five minutes. With the fan, it stays in low-speed about 2/3rds of the time with brief bursts of higher speed and the sink feels cool. CPU and memory usage are fairly minimal so I do not doubt it can record 32 tracks at a time for sustained periods. If I have one real regret with this project, it is that I should have spent the extra $25 on the 8" high-res screen because many program windows don't fit in the 800×480 space. I will likely get a larger screen soon.

Linux Gaming: Saturn Games – Part 3

April 1, 2018 By Tobias Schaaf Gaming, ODROID-XU4

We are back with this month's look at Sega Saturn games for the ODROID-XU3/XU4. Last time I talked about a lot of shoot 'em ups, but this article has a good mix of different genres, although I did pick quite a few "mecha" games I enjoy. Since this is the most games I've covered in a single article so far, I've made an effort to keep each description shorter this time.

Although these games were tested on the ODROID-XU3/XU4, they should work just as well, if not better, on the upcoming ODROID-N1 as well. My initial tests for the ODROID-N1 were quite good, so it's no surprise that the N1 handles Sega Saturn emulation well.

Macross – Do you remember Love

What I really stands out for me in this game are the amazing anime cutscenes. These are not your regular JRPG-styled cutscenes like in Popfull Mail; they're straight out of the Japanese anime series Macross itself. The scenes are incredibly detailed, making it hard to believe you're watching a cutscene in a game and not the actual anime.

Similar cutscenes can be found within the game itself: when you encounter a boss and when the screen shows an anime-style conversation between your character and the boss. The game is completely in Japanese, but it doesn't interfere with enjoyment of the game, although you might miss some minor plot elements.

In this game you fly a transformable aircraft which can be turned into a mecha. Each stage has its advantages and disadvantages. Aircraft mode is very fast and easy to maneuver, but your weapons are somewhat weaker. Changing into a mecha strengthens your attacks, but you become slower, more sluggish in your movements, bigger, and with that easier to hit. Fighting through different stages, your form can be limited to one or two of the three options depending on your stage.

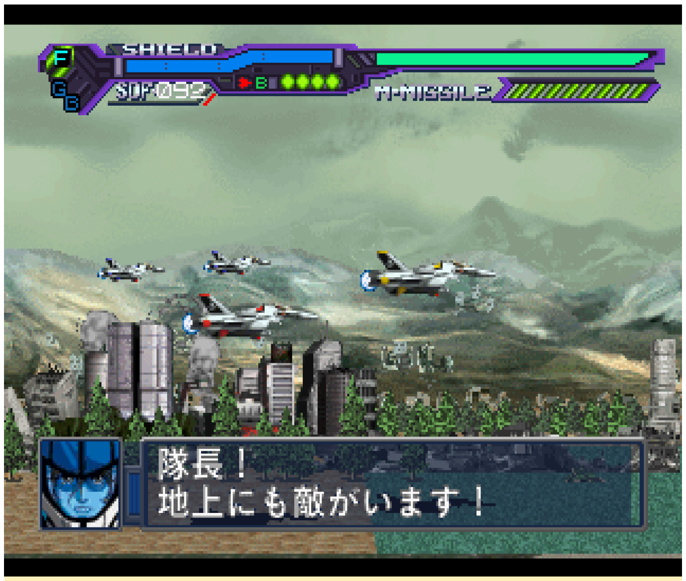


Figure 1 – Macross has three fighter stages, each with their own strengths and weaknesses

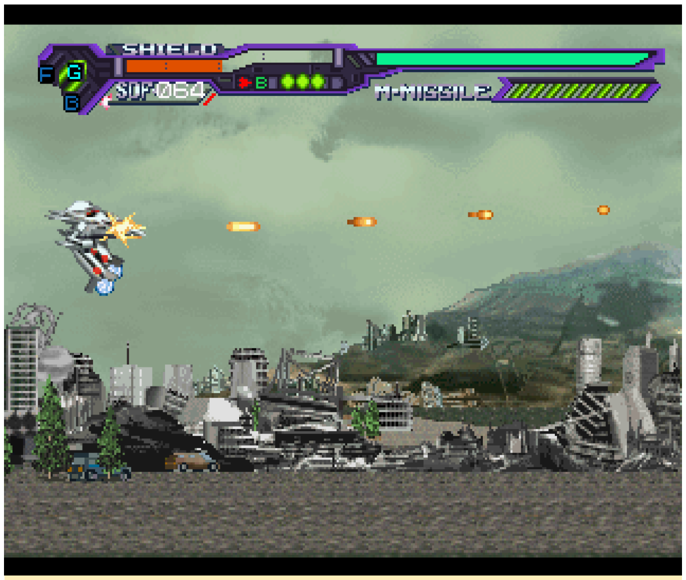


Figure 2 – Macross has three fighter stages, each with their own strengths and weaknesses

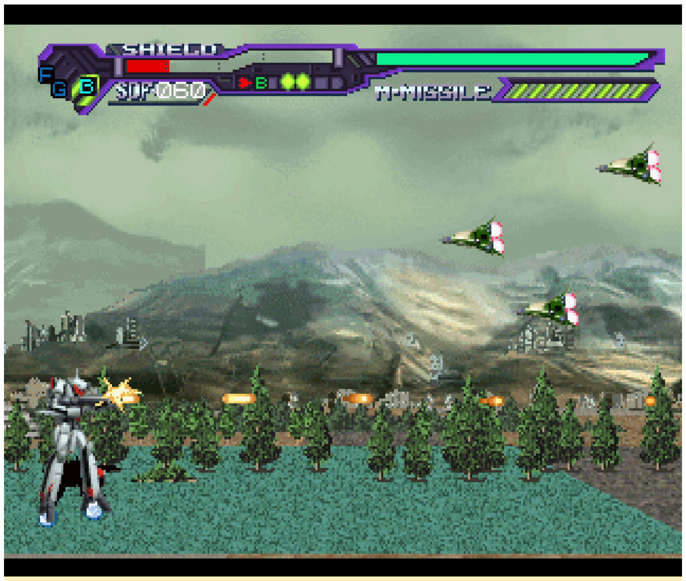


Figure 3 – Macross has three fighter stages, each with their own strengths and weaknesses

You also have three different weapons: auto-aiming missiles which you load and auto-target by holding a button, a fast machine gun, and a bomb that covers most of the screen, which helps you if you're surrounded by enemies. As the enemies come at you in three planes (foreground, middleground, and background), the auto-aiming missiles are likely going to be your main attack.

The game allows you to save your game progress to the system memory. This is one of the few games that actually fits into the system memory, without complaining that the system memory has not enough space. The game is a good, solid shooter that's fun to play and the anime cutscenes are simply amazing.

Magic Knight Rayearth

This game is based on an anime of the same name about three school girls who are transported to a magical world where the girls are told they are the Magic Knights and are suppose to save the world. The game starts with a long intro that features a mix of anime cutscenes, in-game graphics, and a lot of dialogue.

Magic Knight Rayearth was translated by Working Designs, a company that spent many years to porting Japanese games to the North American market and doing a incredible job, with accurate translations and without monotone voice acting. This was the last game they ever ported for Sega and the last official North America release for the Sega Saturn.

The game plays like an action-RPG similar to The Legend of Zelda or Beyond Oasis. You control the movements of all three girls at the same time, but only one is active. Hikaru (red girl) is part of a kendo club and uses a sword which is kind of short and hard to handle. Umi (blue girl) is part of a fencing club and uses a longer rapier for her attacks. Fuu (green girl) is part of an archery club and fights with a bow. They can also use magic: Hikaru possesses fire-based skills; Umi is skilled with water; and Fuu uses wind-based skills and healing spells.

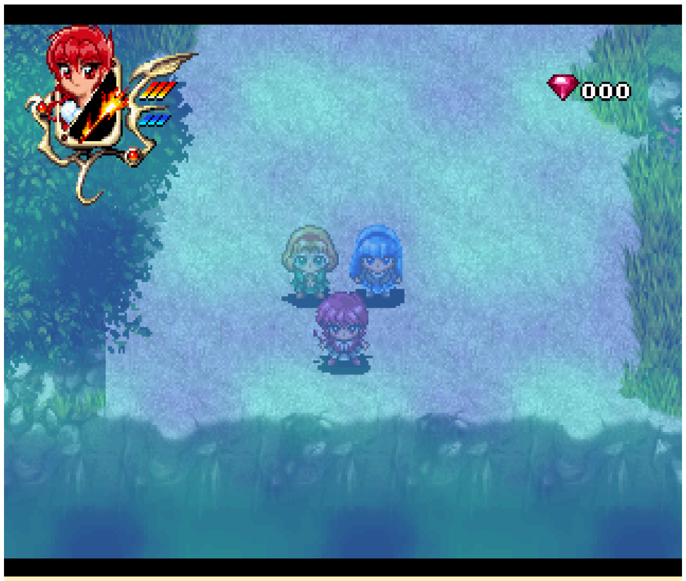


Figure 4 – The three main characters of Magic Knight Rayearth, starting off on their adventure

The dialog in the game is well written, with voice-overs in some parts of the game. Every now and then you'll encounter new people or situations, and with that, you are treated to another nice anime cutscene. All in all, this is a fun game to play. Roaming through different dungeons, you're fighting lots of monsters while trying to find hidden treasures. Occasionally you find items that upgrade your health or magic, but each item only works once, so make sure you think about which girl is going to take it.

I like the bright colors and understandable gameplay. I also like that you can save your game process wherever you want and are encouraged to do so. The translation was well done, making this game a masterpiece for the Sega Saturn. I haven't played much of this game, but I've seen a lot that I liked and I plan to keep playing for quite a while. You should definitely check this game out.

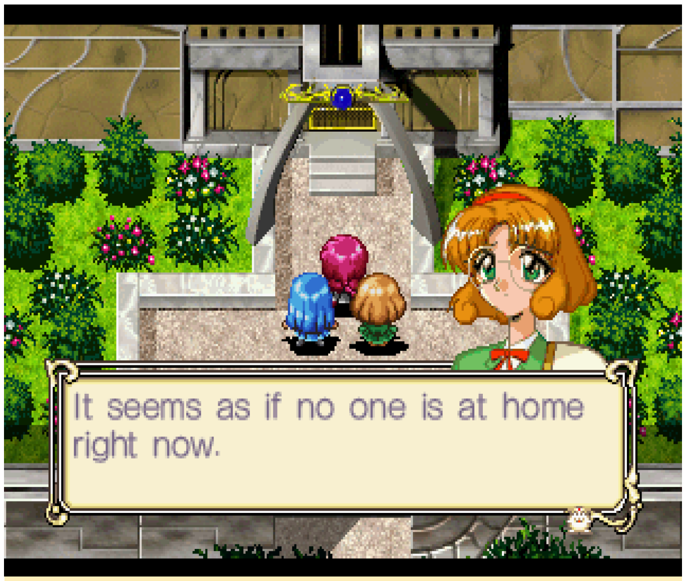


Figure 5 – Arriving at Presea's home to get your first weapons

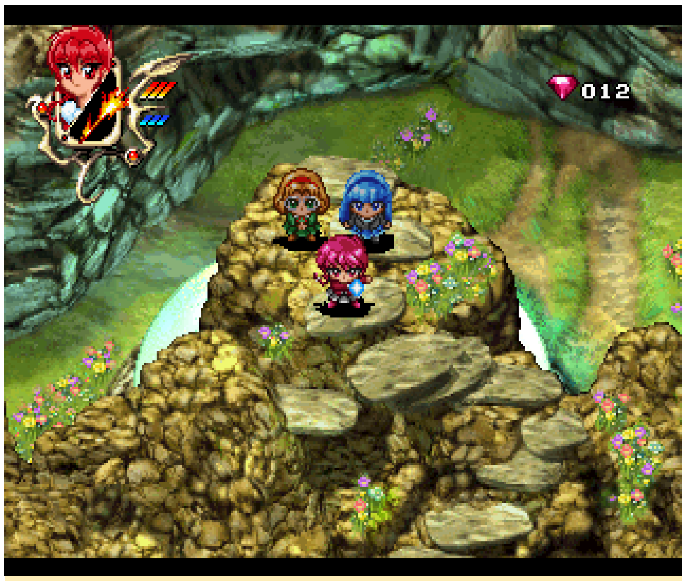


Figure 6 – On the way back from your first mission to gather Escudo

Mega Man 8–Anniversary Edition

I'm not a huge fan of the Mega Man series, but of all the games in this series I have played, this is probably the one I like the most. When it comes to graphics, I like good comic-style games. Games like Monkey Island 3, or Mega Man 8 are timeless due to their graphic style. They looked awesome back then and they still look awesome today. If you like 2D games this is as good as it gets. Although the game also exists for the PlayStation One, the Saturn version offers better music (PCM VS Midi), additional bosses (Cut Man and Wood Man), additional artworks, and so forth. The PS1 version also suffered from some minor graphical glitches.

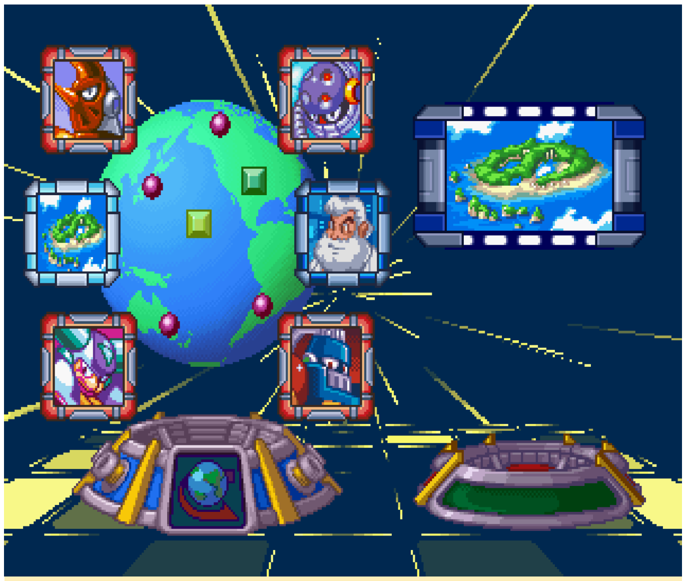
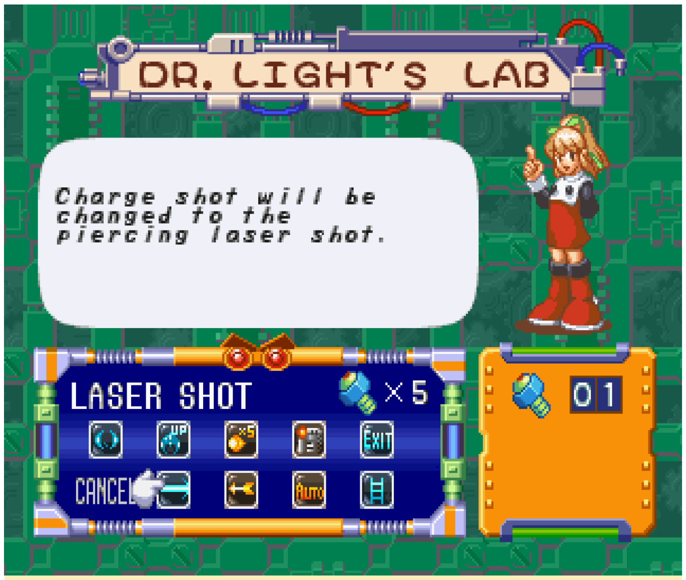


Figure 7 – The level select menu in Mega Man 8

From the level selection screen you can go to different worlds fight and different enemies and bosses. If you defeat them you get their powers to use against other enemies, same as all the Mega Man games. From the level selection screen you can also go "home" where you can exchange collected "screws" for special upgrades, such as different attack styles and other goodies.



Charge shot will be changed to the piercing laser shot.

Figure 8 – Upgrade-screen for Mega Man

Some of these upgrades make a big difference. A charged attack that no longer hits just one enemy but all enemies in line sounds like something I'd like to have. Mega Man 8 offers different level with different enemies, graphic style, and challenges.

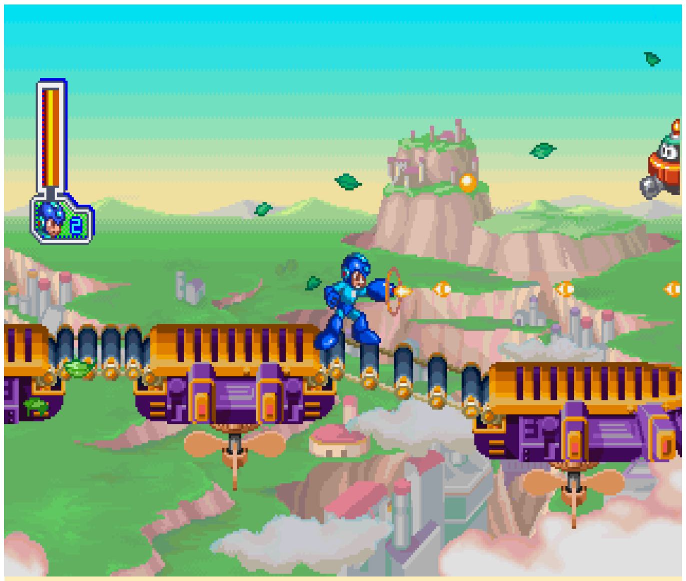


Figure 9 – There are different levels in Mega Man 8, such as up in the sky

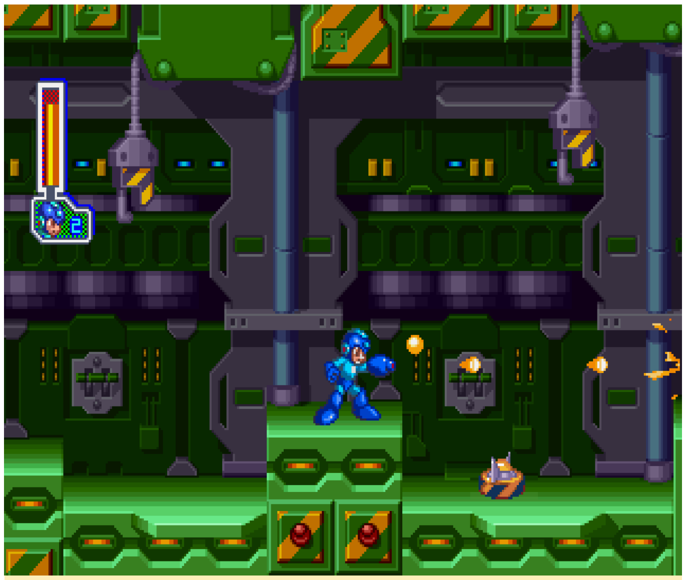


Figure 10 – There are different levels in Mega Man 8, including inside a machine dungeon

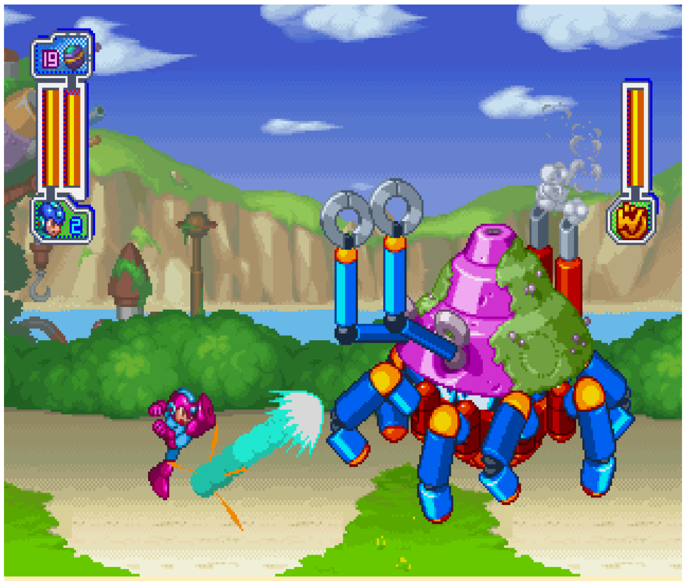


Figure 11 – No Mega Man game is complete without boss battles!

Generally, this game is a fest of colors and beautifully drawn graphics. The awesome soundtrack just completes this wonderful experience. I highly recommend this Sega Saturn game for the ODROID-XU3/XU4/N1.

Mobile Suit Gundam Side Story I, II, and III

I've always said the Sega Saturn wasn't made for 3D games and that developers should have stuck with 2D games. Boy, was that a lie!

It's true that there are many consoles with better 3D capabilities than Sega Saturn, but the Saturn had its gems: games that made the most out of what the console had to offer, while still keeping up speed and playability. The Gundam series certainly fits that description.

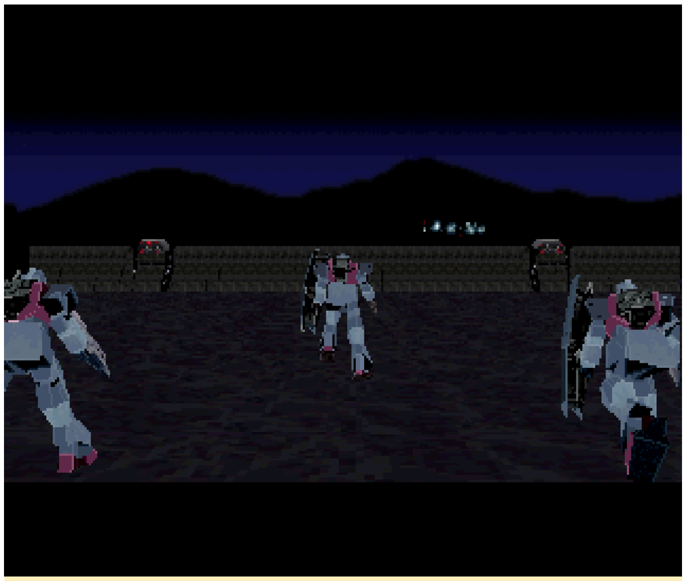


Figure 12 – The 3D graphics in Mobile Suit Gundam Side Story are surprisingly good for the Sega Saturn

My first Gundam 3D game was actually for the Sega Dreamcast. I instantly fell in love with it, but seeing the games for the Sega Saturn, I'm positive I would have also enjoyed them as a first experience. I definitely enjoy them now and I was surprised how well they played on my ODROID. I enjoy this game a lot, even if it's sometimes hard to figure out the goal, as the game is completely in Japanese. Still, the gameplay is easy to understand and you can get into the action right away.

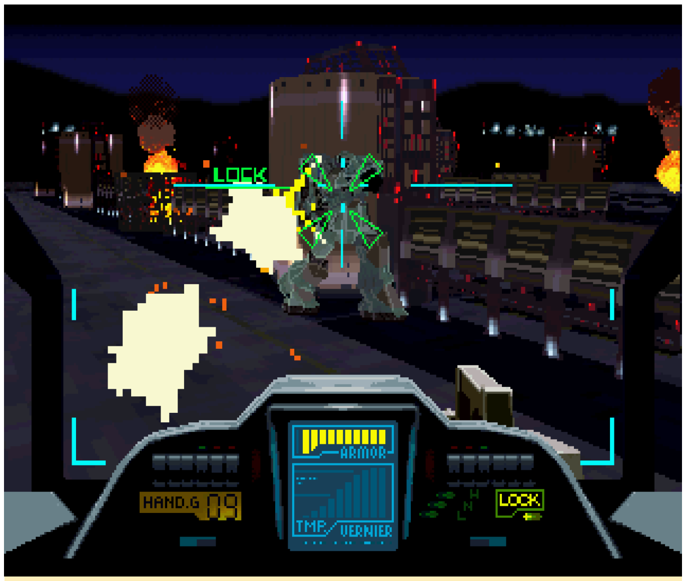


Figure 13 – Target an enemy and pound him until he's destroyed

Use your Gundam to hunt down different enemies using a machine gun, your cannon, area weapon (like a grenade or missile), or a sword. I have to admit I haven't entirely figured out how to make the sword work yet. The mission briefing is a little bit hard to understand since it's in Japanese. It's fully voiced, which is nice, but I don't understand a word of it.

The first game in the series has a very minimal mission interface. You get an overall map but rarely see any mission points so it's hard to figure out your goal. From the second game, the information screen gets a little bit more talkative and you can figure out from the icons, arrows, and blinking objects what your goal is gonna be.

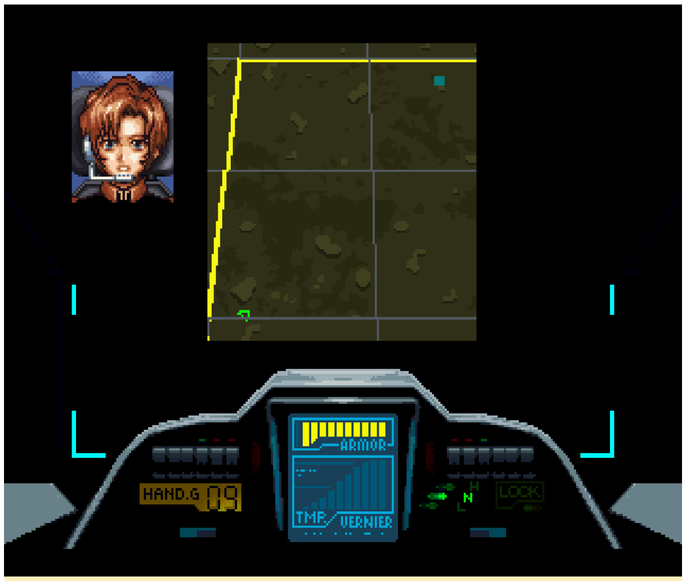


Figure 14 – Your mission map in Mobile Suit Gundam 2 and 3, which you see before each mission

Each level involves fighting different kinds of enemy Mecha/Gundam and an occasional harder boss that you need to defeat or scare off. These bosses can be really challenging as they move very quickly and may be hard to hit with your main cannon. Over the course of the game, and from one game of the series to the next, your Gundam will change, gaining better weapons that allow you to deal more damage and travel faster as you progress.

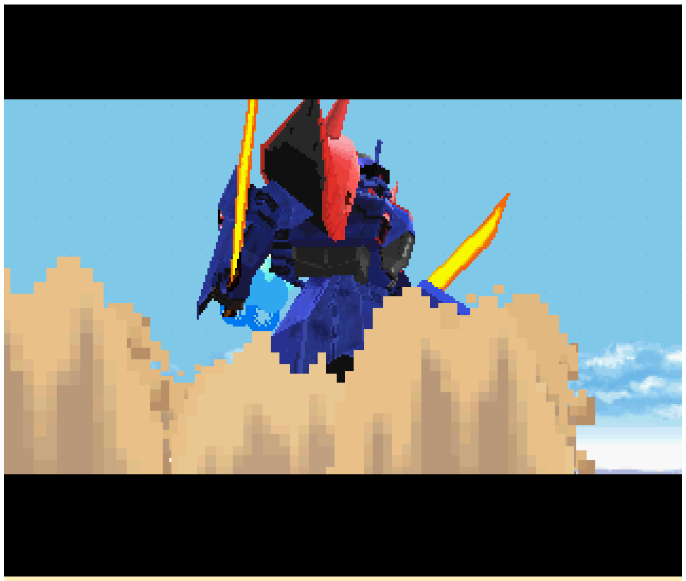


Figure 15 – New boss approaching...cutscenes are all in in-game graphics

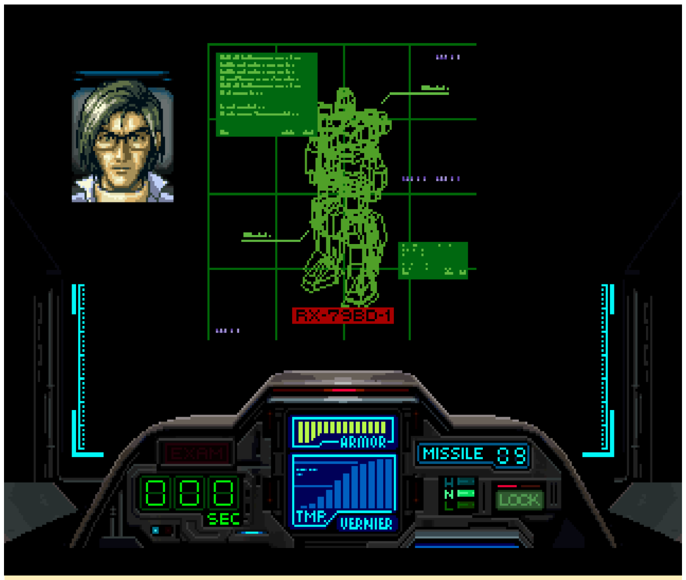


Figure 16 – Quick overview of your Gundam and its system

I really enjoy playing this game on the ODROID and highly recommend it to anyone that likes a good action game.

Mobile Suit Z Gundam

Yes, this is another Gundam game, and not the last one, as there is one more in the honorable mention section. However, this one is different than the previous ones. It's not a 3D game, but 2D, which is closer to the Macross game mentioned earlier.

Mobile Suit Z allows you to switch your Gundam between a robot-style and a aircraft-style. Similar to Macross, your aircraft is faster and shoots quicker, but the Gundam is much more powerful, with access to additional attacks and weapons such as the ability to aim at background and foreground enemies, and a very strong and fast saber to kill your enemy.

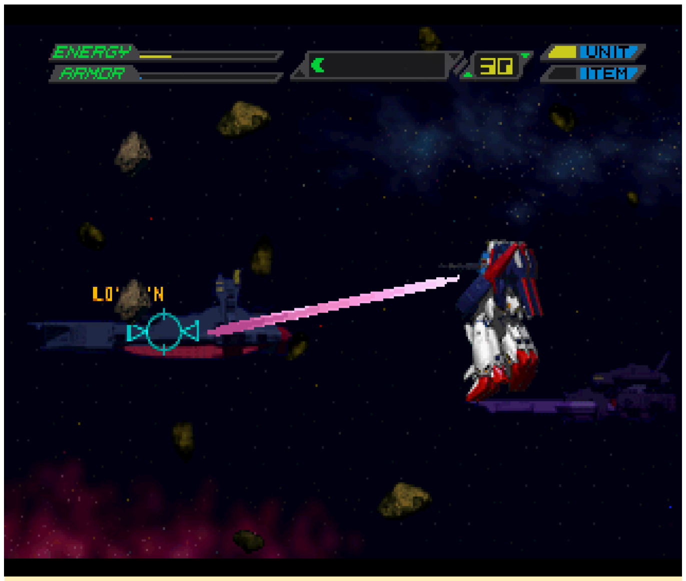


Figure 17 – Aiming at an enemy in the background only works in Gangnam–erm–"Gundam-Style"

Compared to the previously-mentioned Gundam games, this game is filled with anime cutscenes. Intro, before missions, after missions and even sometimes during missions you have anime cutscenes which fit the scenario quite well. Compared to Mobile Suit Z Gundam Zenpen Z no Kodou, this version is much easier to control as all your actions are mapped to different buttons, making it very easy to switch between attacks or aim at enemies in the background while slicing up enemies in the mid-ground.

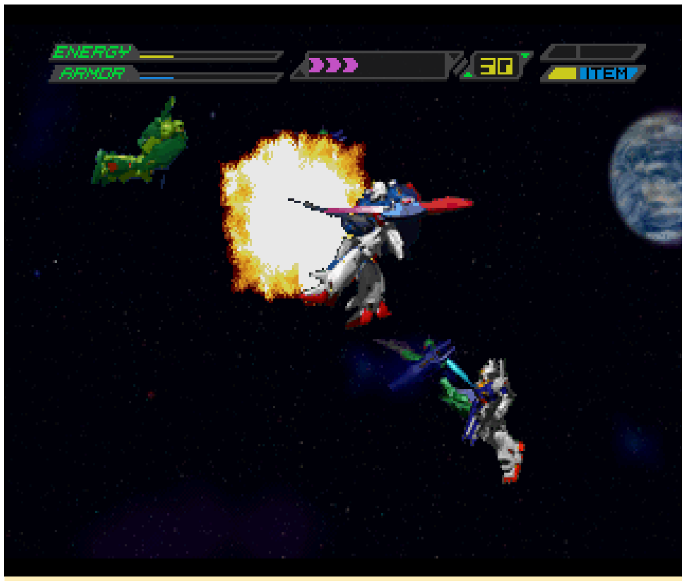


Figure 18 – Slicing up some enemies with your saber is always rewarding and fun to look at

At the end of a stage you'll often encounter a boss which can take a lot more damage than other random enemies that are thrown at you all the time. I like is that there is always a conversation going on between the characters which makes it feel like you are right there in the moment.

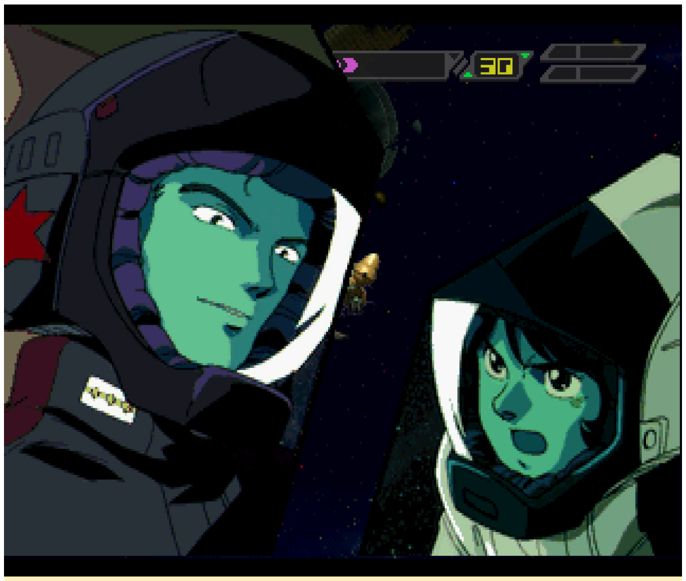


Figure 19 – Before each boss fight there is a cutscene featuring you and your enemy

In the upper left corner of the screen there are two bars: "Energy", which is your health, and "Armor." The latter regenerates over time so taking a few hits is not that bad. After each stage a progress screen shows you how well you did and how your character and Gundam improved. Over time, you'll get a lot stronger, take more hits, regenerate a lot faster, and aim at more enemies at once. As usual, the game is done entirely in Japanese, but the gameplay is easy enough to understand.

Pocket Fighters

This game is best played with the 4MB memory expansion. It features many animated clips, and the extra memory helps to improve the animations. If you compare the Sega Saturn with the PS1, the two versions of Pocket Fighters are very close, but the Sega Saturn still has a bit more animations here and there, and an "Around the World" kick that is not present in the PS1 version. However, the PS1 version is available in English, while Saturn is Japanese only, although this doesn't really affect the game. Since the Sega Saturn version runs very well on the XU3/XU4 (or

ODROID-N1) it's up to you which version you want to play.

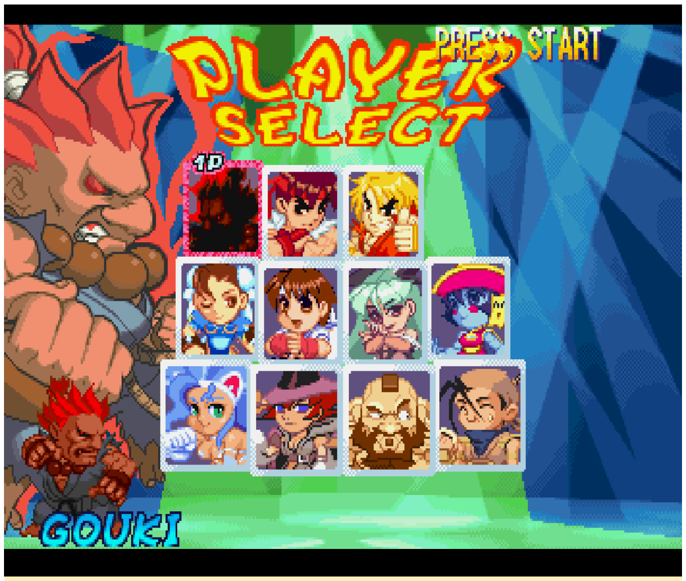


Figure 20 – Originally you only see three male characters (see picture in the middle) but when you go to the left or right of either Ruy or Ken you'll find two more "hidden" characters to choose from

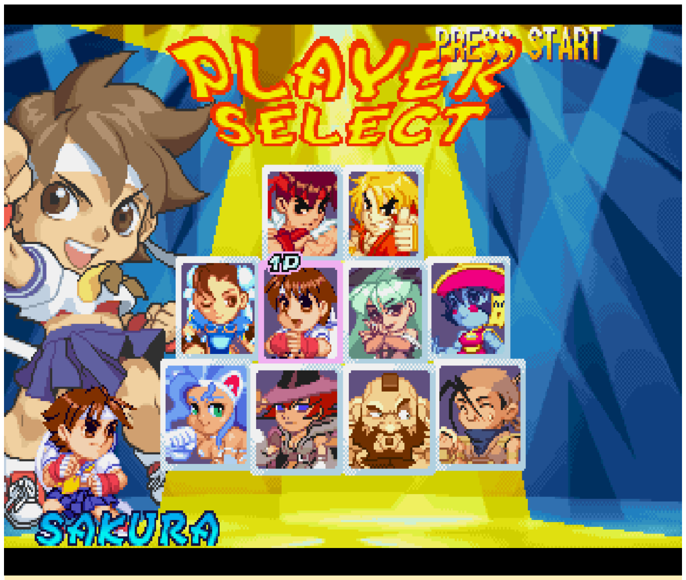


Figure 21 – Originally you only see three male characters (see picture in the middle) but when you go to the left or right of either Ruy or Ken you'll find two more "hidden" characters to choose from

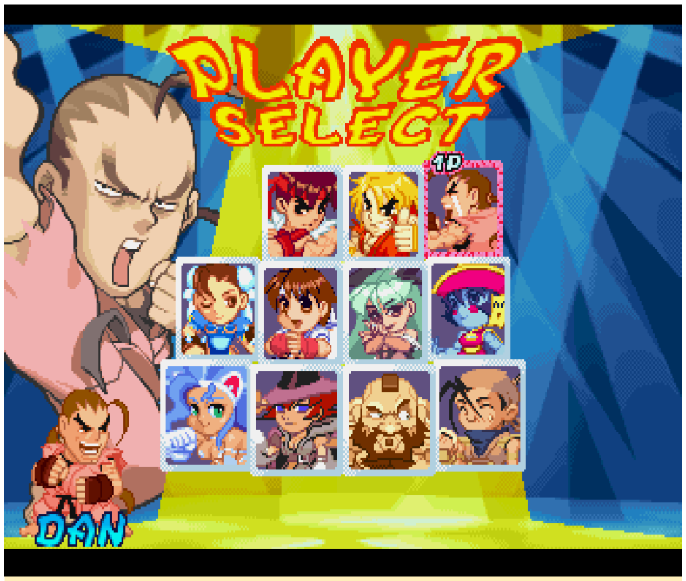


Figure 22 – Originally you only see three male characters (see picture in the middle) but when you go to the left or right of either Ruy or Ken you'll find two more "hidden" characters to choose from

This game plays like most fighting games, but with a fun "chibi" character style. During some combos the characters change their costumes numerous times, making the game look quite funny. It's definitely not a fighting game that takes itself too seriously, giving it a unique style.

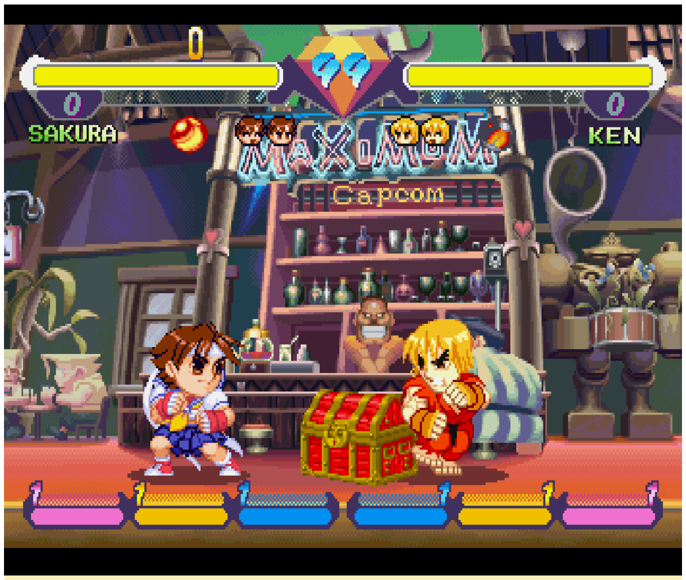


Figure 23 – A treasure chest with gems inside stands between the fighters at the start of each fight

This game is also known as "Super Gem Fighters" on arcade machines. This name refers to you collect gems during fights either by opening a treasure chest or hitting the enemy.

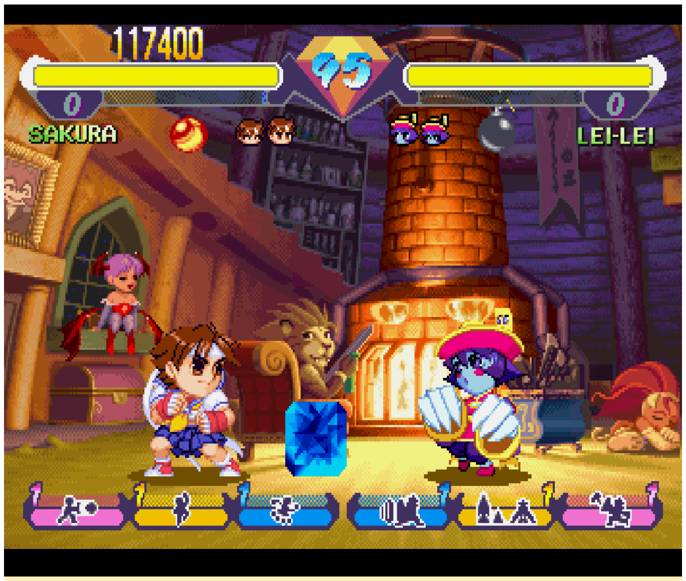


Figure 24 – The gems you earn from fighting give the game its second name: Super Gem Fighters

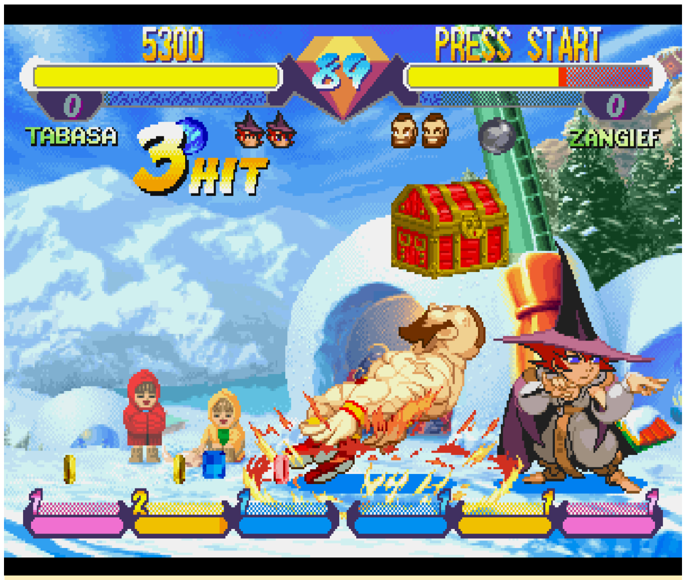


Figure 25 – The gems you earn from fighting give the game its second name: Super Gem Fighters

If you're able to hit a successful combo, the enemy will drop a treasure chest which has several gems of different size inside. This is probably probably one of my favorite fighting games, regardless of the system.

Robo Pit

This is another 3D Sega Saturn game that doesn't disappoint. The graphics aren't all that great, in fact, they are rather simple, but I think that is what actually saves this game. It runs rather well, but I don't think it's actually running at full spee, but it's very playable and you probably won't notice that it's a bit slow.

In this game, you can build your own robot and use it to fight other robots in an arena (I guess that's why it's called Robo Pit in the first place). Your robot is customizable with a wide range of different body types, legs (for driving, jumping, or even flying), arms (which hold different weapons), and even eyes.

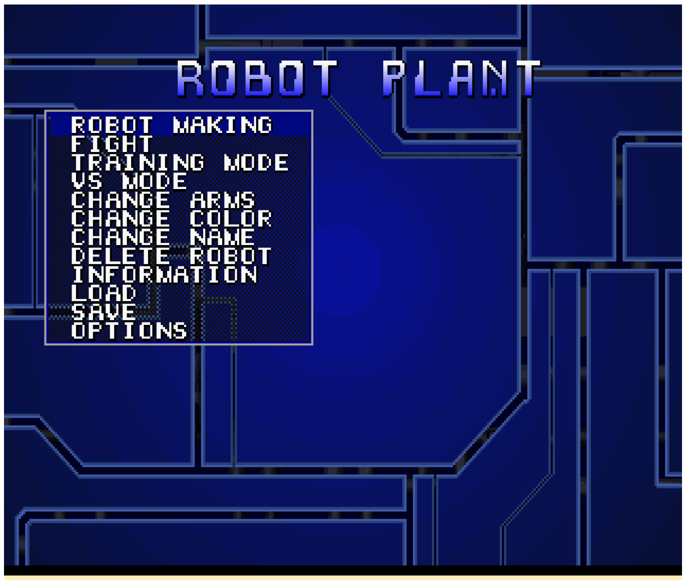


Figure 26 – The game menu is very simplistic, but it allows you to access all aspects of your robot and your career

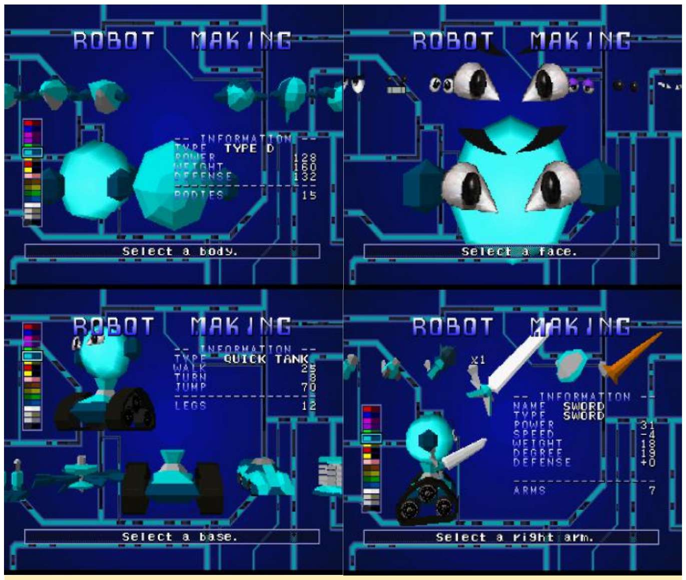
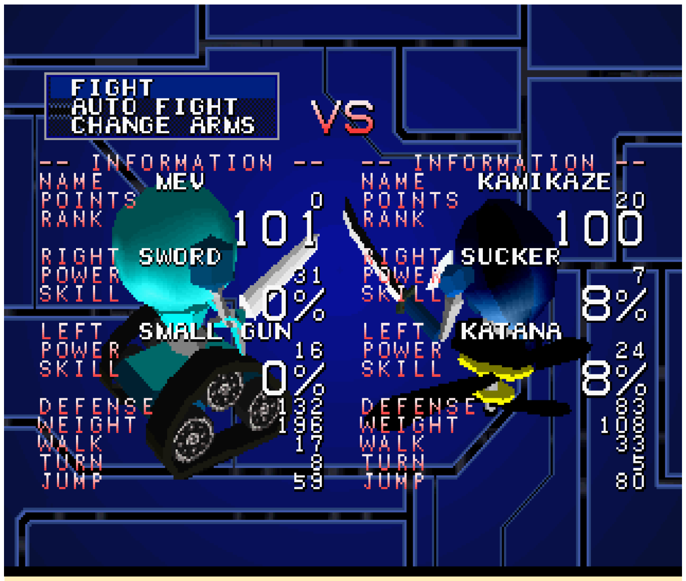


Figure 27 – Build your very own robot based on your preferences

After creating and naming your robot, you march into battle. The battle are quite simple: smash, slice, shoot, or punch your enemy until his health bar hits zero, before your own bar goes down. Each fight gives you points which increase your rank, allowing you to fight even more, stronger enemies. After each fight your stats increase depending on how you did in battle. If you only use your left arm to attack, your left arm skill will increase while the right arm will stay the same.

You get extra hit points, experience for your left and right arm, and defense depending on how you did in battle.



```
FIGHT
AUTO FIGHT
CHANGE ARMS                VS

-- INFORMATION --           -- INFORMATION --
NAME      MEV               NAME      KAMIKAZE
POINTS            0         POINTS            20
RANK            101         RANK             100

RIGHT  SWORD                RIGHT  SUCKER
POWER            31         POWER              7
SKILL            0%         SKILL             8%

LEFT   SMALL GUN            LEFT   KATANA
POWER            16         POWER             24
SKILL            0%         SKILL             8%

DEFENSE         132         DEFENSE           83
WEIGHT          196         WEIGHT           108
WALK             17         WALK              33
TURN              8         TURN               5
JUMP             59         JUMP              80
```

Figure 28 – Select an enemy and beat him – it's as easy as that!

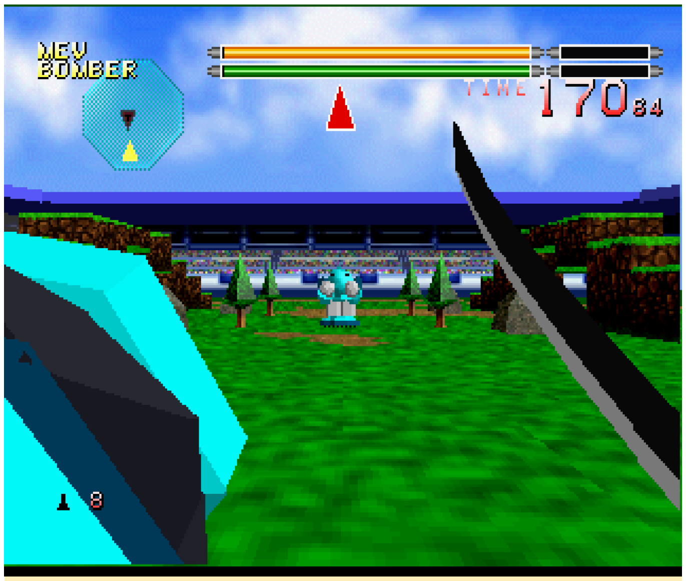


Figure 29 – Arenas are not very detailed, but it fits the game

If you beat an enemy, you will get his arms and can use them as your own weapons. If you lose a fight, you will lose your own arm and have to switch with something in your inventory, so it's best to keep some spares around, or collect them from weaker enemies.

The graphics are rather simple, especially the arena, but it fits the game perfectly. You can switch from a first-person perspective to two different third-person perspectives, whichever way you prefer. It's quite a funny little game with not much story or deep stimulation. Just some good old "beat the crap out of your enemy and get a bonus." It's a nice game to kill half-hour or so, and I really like playing it on the ODROID.

Honorable Mentions

Marvel Super Heroes

I normally don't like fighting games that much, but this game is actually quite good, and I only placed it here cause I already have so many other games above. Honestly this is a really good game. The character sprites are huge, and it features vibrant colors and awesome gameplay. It runs with a 1MB or 4MB RAM expansion pack for more animations.

Mega Man X3

Although this is not a bad game, I really prefer Mega Man 8. Mega Man X3 looks a lot more serious than other Mega Man games. In the intro video, cars and buildings in town are completely destroyed, giving the game a more sinister feeling. The characters are slightly bigger and seem more mature. I always pictured Mega Man as similar to Astro Boy: more of a kid fighting imaginative robot monsters. Mega Man X3 seems to be more serious than that. There are also some minor graphical glitches when walking, but nothing serious. It's a good game, though.

There's also Mega Man X4 but it seems to have issues with the controls. Using yabause-qt you are stuck when you try to go into the game, but with yabause-gtk it seems to work. It's still not worth switching emulators, in my opinion.

Metal Black

Actually an interesting shoot 'em up game. It's very basic: a bunch of enemies come at you, you kill them, and every now and then you encounter a boss. What is different about this game is that your "power-ups" simply fall from the sky like flowers in the wind and you collect them. The more you collect, the stronger your main attack becomes (6 levels). The same energy is also used to start a "mega-attack" which deals a lot more damage and normally kills everything on the screen. The only issue I have with this game is that once you start your special attack, it completely drains your energy. I haven't found a way to stop it. This means that after a special attack you're very weak and do very little damage.

Metal Slug – Super Vehicle-001

This game is similar to the NeoGeo or PS1 version, although slightly superior to the PS1 version as the extra RAM on the Sega Saturn allowed for more sprites for animation. This version works fine although I have experienced some graphical glitches here and there. This game requires either a 1MB or 4MB memory extension pack to work.

Mobile Suit Z Gundam Zenpen Z no Kodou

This game is similar to Mobile Suit Z Gundam, but with slightly different controls. Instead of mapping each button of the controller for one attack, you have to use buttons to switch between weapons and use a single button for attack. It's slightly inconvenient, therefore not as good as Mobile Suit Z Gundam.

Nights Into Dreams…

Some would probably kill me if I didn't mention the game. Some might still kill me because it's only a "footnote". It's an interesting game, but I only play it for a couple of minutes at a time. It's not a go-to game for me as I rarely play it, but I can see why some people really like it. It works ok on the XU4, but slightly too slow. The 3D part slows the game down, so it might actually run better on the N1 or with real 3D.

Norse by Norsewest (aka Lost Vikings 2)

This is actually quite a good puzzle game, it's just that I'm not a huge fan of these kind of games. Although I played the first Lost Vikings game on the Amiga CD32 for quite a while, it's not one of my favorite games either. In this game you control three Vikings with different abilities, guiding them through different level, solving puzzles to defeat the evil Tomator! It's a funny little game, with lots of voice acting and nice music, but it's just not for me.

Panzer Dragoon Series

Many people seem to like Panzer Dragoon, and they are not bad games, I just think they should have not been brought to life on the Sega Saturn. The Saturn was all that good with 3D graphics. These games are purely 3D, and as such, they look rather bad, at least on the ODROID. They are also rather slow on the ODROID-XU4. They might have been good games on the real Sega Saturn, but on ODROID, they are just not that great. I kind of like Panzer Dragoon Saga (an RPG game in the series) but the graphics aren't good. I bet these games would have been way better on the Sega Dreamcast.

The last game of the series, Panzer Dragoon Orta, was released for the original Xbox, and gives you an idea of what the game could have looked like had it been ported to the Dreamcast instead. Still, the Panzer Dragoon Saga RPG won several prizes and it's easy to see why. However, it would be better if the graphics had aged better.

Parodius

Parodius actually comes with two games: Parodius and Fantastic Journey. It plays very much like Gradius and is suppose to be a parody of it. It has an interesting comic style and doesn't take itself too seriously. You have plenty of different characters to choose from, all of which have their own individual weapon style and upgrades, making it fun to experiment with the different characters.

It also comes with an auto and semi-auto mode, where the PC handles your weapon upgrades so you can concentrate on shooting. This works well, and you will have a very strong attack in no time, as power ups are quite common. However, as with all Gradius-style games, you lose everything when you get destroyed.

Princess Crown

This is a lovely action RPG game with big character sprites and a real-time fighting system. I really like it, but would like it more if only I could understand what they are saying, since the game is completely in Japanese. It's a shame, as I would have really enjoyed playing it.

Purikura Daisakusen

This is a virtually perfect arcade port. If you know the arcade version, you know the Sega Saturn version. You could probably call it a cute 'em up, but instead of flying or any scrolling action, you actually walk by foot and fight your way through the levels. You have a tiny companion that slowly evolves as you progress, and has a special attack that can hit all enemies on the screen. I like this game due to the colorful sprites and lovely backgrounds, but the constantly "pew pew" sound effects for the shooting can get a little annoying.

Rabbit

This is another game that I only put down here because it's already so full up top. Rabbit is another fighting game where you fight together with a "beast soul." You and your enemy have an animal soul with you that can be called forth, allowing you to do special attacks. Once you've beaten an enemy, you collect their beast soul and can use it as an special attack on your next enemy. It's a very interesting fighting game, although I don't like all of the character sprites because some of them don't look very polished. There are games with better graphics out there, but at least the fighting style is unique.

Radiant Silvergun

This is the first shoot 'em up I really played for the Sega Saturn. I also covered it in my first article about the Sega Saturn for ODROID back in September 2016. What I really like about this game is availability of the different attacks. You have six different attacks, from auto-aiming over strong frontal attack, to attacks that shoot at enemies behind you. There are even electrical sparks that get stronger the longer you attack. The Sega Saturn controller had six action buttons, meaning that all attacks were mapped to a different button. There are more buttons to use, as the shoulder buttons activated a sword which could both attack and collect enemy bullets, making it perfect for avoiding hits. You could also launch an even bigger attack that would strike nearly the entire screen, hitting many enemies at once or simply dealing extra damage to bosses.

Radiant Silvergun was my first shoot 'em up for the Sega Saturn and a good one at that. Although it's made as a 3D game it plays rather nicely, which is somewhat rare on the Saturn. My default opinion about the Saturn is that it excels at 2D graphics and should have stayed away from 3D.

Rayman

I know that Rayman exists for many systems, and the PS1 version comes very close to the Saturn version, but the Saturn version is considered slightly superior. It has additional animation between loading screens, in between stages, and at certain bosses. The sound and music is also said to be better on the Saturn. Aside from that, the game itself is pretty much the same on all systems.

Arcade Box

April 1, 2018 By Brian Kim, Charles Park, and John Lee Gaming, ODROID-XU4, Tinkering

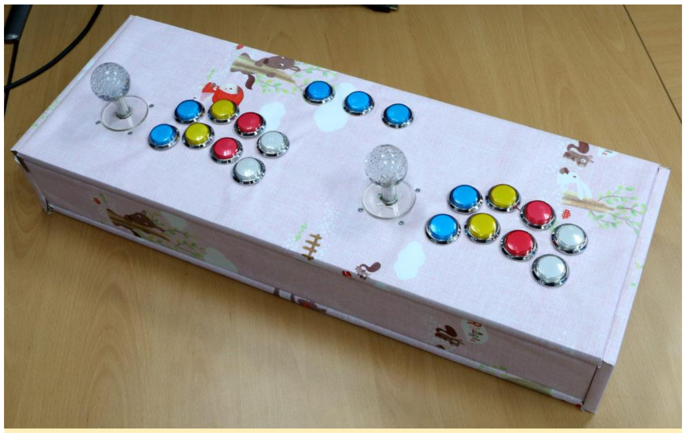


Figure 1 – ODROID Arcade Box

ODROIDs have better performance than the competitor boards, especially in video rendering, which means that ODROID boards are very suitable for playing games, which many ODROID users do. There are already several game platform operating systems available, such as Lakka (http://bit.ly/1NO8BBC) and ODROID GameStation Turbo (http://bit.ly/1ASFO5O). In order to enjoy our gaming sessions more, we made our own arcade box with simple GPIO buttons and joysticks, and called it the ODROID Arcade Box. We choose an ODROID-XU4 for this project because it has the best GPU performance of all the current ODROID devices. This article describes how to recreate the ODROID Arcade Box for yourself.

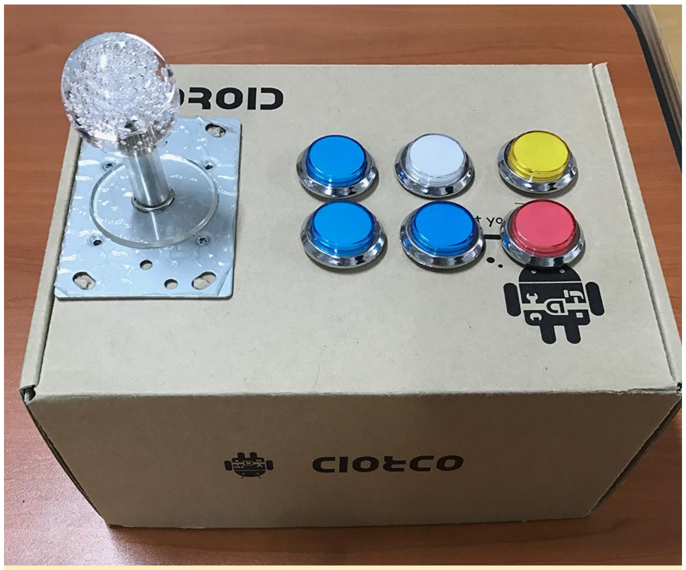


Figure 2 – Our first simple prototype

Requirements

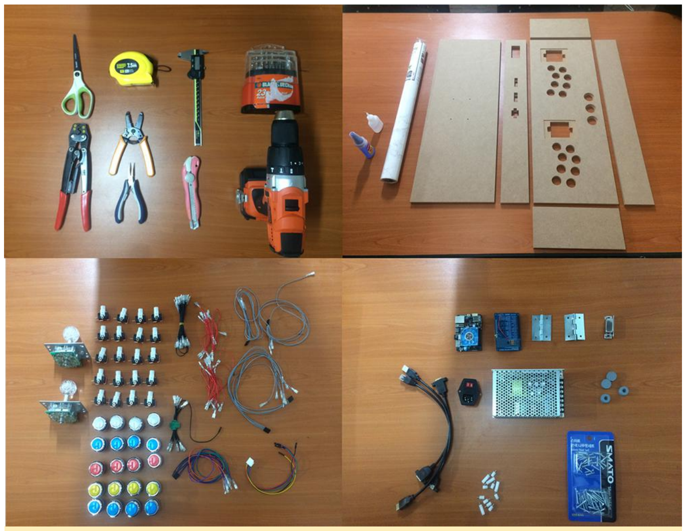


Figure 3 – Tools and parts

We decided to make the ODROID Arcade Box using MDF (Medium-Density Fibreboard). The XU4 Shifter Shield is also useful in order to utilize the expansion pins of the ODROID-XU4. Joysticks, buttons and cables were the input components, and an SMPS (Switched-Mode Power Supply) was used for the power supply. The detailed tools and parts list are listed below:

- 12T MDF Panel
- 2EA 600×220
- 2EA 600×75
- 2EA 220×75
- Drill
- Crimper
- Stripper
- Measuring tape
- Utility knife
- Long Nose Plier
- ODROID-XU4
- XU4 Shifter shield
- SMPS
- HDMI, USB, Ethernet extenders
- Power socket & Switch
- 2EA Hinges
- Door catcher
- 4EA foot rubber
- Screws
- 19EA Buttons
- 2EA Joystick
- Wires
- Terminals

The ODROID Arcade Box needs a total of 27 inputs (19 inputs for buttons and 8 inputs for joysticks). The ODROID-XU4's digital 24 GPIO inputs are not sufficient for all 27 inputs, so we created two additional ADC ports for the three additional buttons. The ADC input values are based on input voltage, and the digital and analog input values are processed in the GPIO key daemon, which is described below.

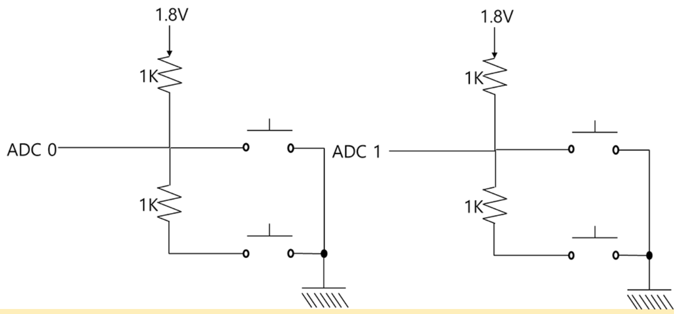


Figure 4 – Expansion ports schematic

Design and assembly

The panels of the ODROID Arcade Box must be designed and manufactured so that the buttons and joysticks are well placed. We chose 12T MDF considered for price and durability. Your design can be done with any familiar CAD tool such as Google Sketch or SolidWorks. Although there are many layout templates for joypad panels available, we chose a standard Japanese arcade layout.

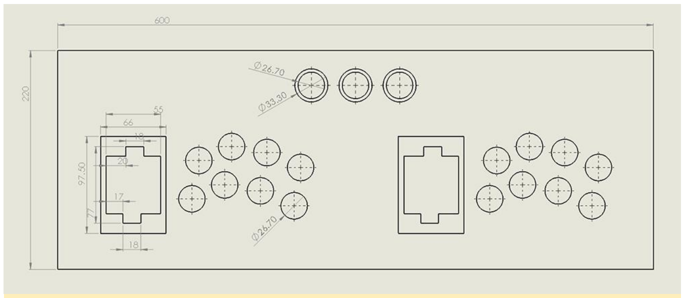


Figure 5 – Joypad Layout Blueprint

The first step of assembly is to attach the sheet to the MDF panel. This step was easy, but took longer than the other steps. After that, we inserted the joysticks, power socket, switch and buttons on the top MDF panel. The HDMI, Ethernet and USB extenders were inserted on the back of the MDF panel. The next step was to assemble each MDF panel by using a drill to make holes in it, then using screws to hold it together.

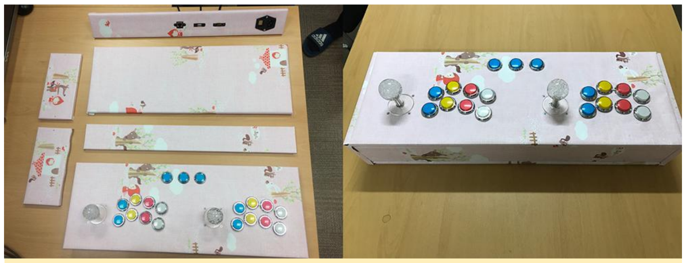
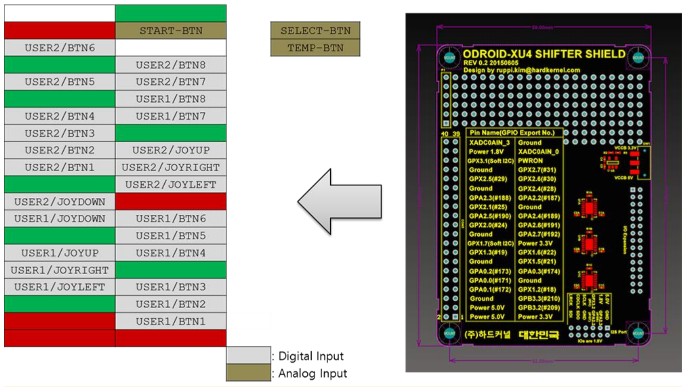


Figure 6 – Assembled ODROID Arcade Box Outline

The last step of assembling the ODROID Arcade Box was wiring the ODROID-XU4 expansion pins to the input components. In this project, we designed the external GPIO inputs, as shown in Figure 5. The Select and Temp buttons are connected to ADC expansion ports, as shown in Figure 3.



	🟩
🟥	START-BTN
USER2/BTN6	
🟩	USER2/BTN8
USER2/BTN5	USER2/BTN7
🟩	USER1/BTN8
USER2/BTN4	USER1/BTN7
USER2/BTN3	🟩
USER2/BTN2	USER2/JOYUP
USER2/BTN1	USER2/JOYRIGHT
🟩	USER2/JOYLEFT
USER2/JOYDOWN	🟥
USER1/JOYDOWN	USER1/BTN6
🟩	USER1/BTN5
USER1/JOYUP	USER1/BTN4
USER1/JOYRIGHT	🟩
USER1/JOYLEFT	USER1/BTN3
🟩	USER1/BTN2
🟥	USER1/BTN1
🟥	🟥

SELECT-BTN
TEMP-BTN

▢	: Digital Input
▨	: Analog Input

ODROID-XU4 SHIFTER SHIELD
(주)하드커널 대한민국

Figure 7 – External GPIO mappings for the Buttons and Joysticks

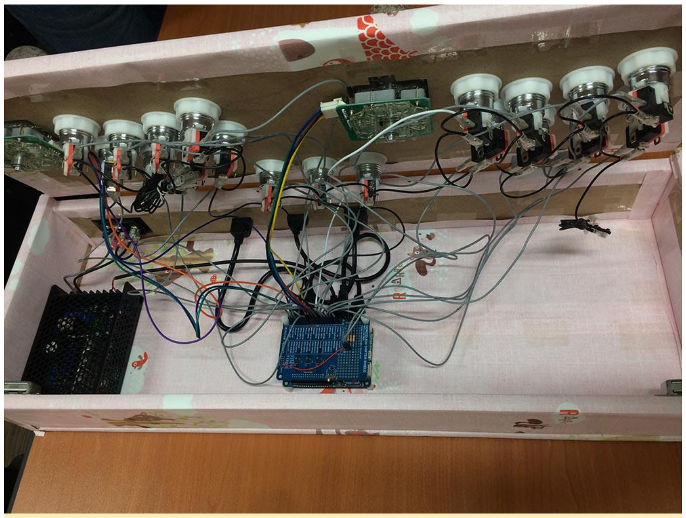


Figure 8 – ODROID Arcade Box Wiring

Software Setup

We developed a new GPIO key daemon called gpio_keyd (http://bit.ly/2ljOZKg). The gpio_keyd daemon is able to map GPIO inputs and key events using uinput and wiringPi, which is a pin-based GPIO access library. It's designed to be familiar to people who have used the Arduino wiring system. Although the upstream wiringPi library supports only Raspberry Pi, Hardkernel offers a wiringPi fork for ODROIDs in its GitHub repository (http://bit.ly/1Eq3UpF). The module uinput is a Linux kernel module that handles the input subsystem from user land. It can be used to create and handle input devices from an application.

We choose ODROID GameStation Turbo (http://bit.ly/1ASFO5O) as the software platform for our ODROID Arcade Box, which has uinput built in. You should make sure the uinput device file exists in your chosen operating system, because some of them do not have uinput devices.

```
$ ls /dev/uinput
```

If your operating system does not have a /dev/uinput device file, then it will be necessary to rebuild and install a new kernel with the INPUT_UINPUT option configuration option set. The Wiki page at http://bit.ly/1YIToBI describes how to build and install the kernel image from source code.

```
$ make menuconfig

 Device Drivers -> Input device support
 -> Generic input layer
 -> Miscellaneous device
 -> User level driver support <*>
```

Note that wiringPi must be installed before installing gpio_keyd. On the ODROID GameStation image, the sudo commands must be run as root, because the “odroid” account is not designated as a sudo user.

```
$ git clone
https://github.com/hardkernel/wiringPi.git
$ cd wiringPi
$ sudo ./build
```

Download the gpio_keyd source code, which is available from our GitHub repository. The gpio_keyd build and installation methods are very simple:

```
$ git clone
https://github.com/bkrepo/gpio_keyd.git
$ cd gpio_keyd
$ make
$ sudo make install
```

The gpio_keyd script refers to /etc/gpio_keyd.conf as the default for GPIO and key mapping information. The configuration file was modified for 27 inputs of ODROID Arcade Box. Some keys are already used in the game emulator, so we had to change the emulator key settings in order to avoid key collisions between the emulator and GPIO input keys. Note that field in the configuration file refers to the wiringPi number, not the GPIO and pin number (http://bit.ly/2lbzPIB).

Configuration file sample for 27 inputs: /etc/gpio_keyd.conf

```
# Digital input
#
# User 1
KEY_LEFT digital 15 0
KEY_RIGHT digital 1 0
KEY_UP digital 4 0
KEY_DOWN digital 16 0
KEY_A digital 2 0
KEY_S digital 3 0
KEY_D digital 30 0
KEY_F digital 21 0
KEY_Z digital 8 0
KEY_X digital 9 0
KEY_C digital 7 0
KEY_V digital 0 0
# User 2
KEY_BACKSLASH digital 12 0
KEY_SLASH digital 13 0
KEY_SEMICOLON digital 14 0
KEY_LEFTBRACE digital 5 0
KEY_Y digital 26 0
KEY_U digital 27 0
KEY_I digital 22 0
KEY_O digital 23 0
KEY_H digital 6 0
KEY_J digital 10 0
KEY_K digital 11 0
KEY_L digital 31 0

# Analog input
#
KEY_B analog 0 0
KEY_N analog 0 2045
KEY_M analog 1 2045
```

To run gpio_keyd daemon at every startup is convenient for ODROID Arcade Box.

```
/etc/init.d/gpio_keyd
#! /bin/sh
### BEGIN INIT INFO
# Provides: gpio_keyd
# Required-Start: $all
# Required-Stop:
# Default-Start: 2 3 4 5
# Default-Stop:
# Short-Description: Run /usr/bin/gpio_keyd if
it exist
### END INIT INFO

PATH=/sbin:/usr/sbin:/bin:/usr/bin

. /lib/init/vars.sh
. /lib/lsb/init-functions

do_start() {
 if [ -x /usr/bin/gpio_keyd ]; then
 /usr/bin/gpio_keyd -d
 ES=$?
 [ "$VERBOSE" != no ] && log_end_msg $ES
 return $ES
 fi
}

case "$1" in
start)
do_start
```

```
;;
restart|reload|force-reload)
echo "Error: argument '$1' not supported" >&2
exit 3
;;
stop)
killall gpio_keyd
exit 0
;;
*)
echo "Usage: $0 start|stop" >&2
exit 3
;;
Esac
$ sudo chmod +x /etc/init.d/gpio_keyd
$ sudo update-rc.d gpio_keyd defaults
$ sudo reboot
```

In the above commands, the gpio_keyd script runs as a daemon using the "-d" option. Usage of gpio_keyd can be checked with the "–h" option. Double-check the keys used by the game or the emulator, then set the gpio_keyd settings correctly. Then, you are ready to play and enjoy your games with your new ODROID Arcade Box.

Figure 9 – The King of Fighters 98, John vs. Brian

Nextcloud Server: Creating a Network Access Storage (NAS) with an ODROID-HC2

April 1, 2018 By Autonomous Linux, Tutorial, ODROID-HC2

This guide will walk you through setting up a NAS (Network Attached Storage) on the ODROID-HC2 single-board computer. There is no reason why it wouldn't work for the HC1 or XU4, however the XU4 uses USB3 instead of SATA. The guide is written for those with no Linux experience and minimal computer building experience. I'm writing it because I had minimal Linux experience and it was a pain to figure out how to set it up, and involved reading a few dozen guides.

Parts List

- ODROID-HC2 : Home Cloud Two
- 12V/2A Power Supply Unit US Plug for ODROID-HC2
- ODROID-USB-UART Module Kit
- SanDisk 32GB Ultra Class 10 SDHC UHS-I Memory Card, up to 80MB, Grey/Black (SDSDUNC-032G-GN6IN)
- AmazonBasics RJ45 Cat-6 Ethernet Patch Cable – 3 Feet (0.9 Meters)
- ODROID-HC2 Cases (Black)
- WD Red 2TB NAS Hard Disk Drive – 5400 RPM Class SATA 6 Gb/s 64MB Cache 3.5 Inch – WD20EFRX

The HC2 is superior to the Raspberry Pi 3 in that it has gigabit ethernet, a SATA hard drive connection, a faster 2 Ghz CPU and 2 GB of memory. It is purposely built to be a network attached storage device or light weight server. The Raspberry Pi 3 is superior in the amount of support available from manufacturers and the community. I opted for the HC2 over the HC1 because high capacity 3.5-inch drives are cheaper than 2.5-inch drives. The HC1 is more energy efficient, if that is a concern. The HC2 does not have an HDMI output so everything must be set up before the image is burned to the MicroSD card, over a VNC remote connection or over a serial connection via Command Line. This guide will cover setup with a serial connection but it is worth setting up a VNC server later, as Linux is easier to use with a GUI if you don't know console commands very well.

The HC2 needs a 12 volt/2 amp power supply, a MicroSD Card, and a 3.5 inch SATA hard drive. I used a 3TB Toshiba hard drive I had laying around. The HC2 with the lid only supports hard drives up to 27mm tall.

Additionally, if the included screws are too short, you may need screws to attach the hard drive to the aluminum HC2 fixture.

You will also need an ethernet cable to connect to your router and the internet. You will also need an ODROID-USB-UART to connect to the serial port on the HC2.

Pick up a HC2 case for $5 to close everything up when you are done.

Assembly

Assembly is extremely simple. Place the hard drive on the bottom of the HC2 and slide it into the SATA port. Supporting the hard drive and the aluminum body, flip the two pieces over ensuring the hard drive does not put weight on the SATA connector. Once you have it upside down, use screws to attach the bottom of the hard drive to the HC2. Flip it back over and connect the ethernet cable from the HC2 to your router. Connect the USB-UART to the serial port on the HC2. Plug the USB end into your computer. Do not connect the power supply yet.

Software

Download ODROID's ubuntu-16.04.3-4.9-mate-odroid-xu4-20171025.img.xz release at https://odroid.in/ubuntu_16.04lts. Ubuntu is a Linux distribution that is free and stable. While that is downloading, download Etcher: https://etcher.io/. Etcher is software that will flash the Ubuntu image onto your MicroSD card. Put your MicroSD card into your computer's MicroSD adapter and start Etcher. Select the Ubuntu image and the MicroSD card then click "Flash." It will take a few minutes to run. When it finishes, Windows will pop up a message about formatting a drive before using it. Click "Cancel."

Take your flashed MicroSD card and insert it into the HC2's MicroSD slot.

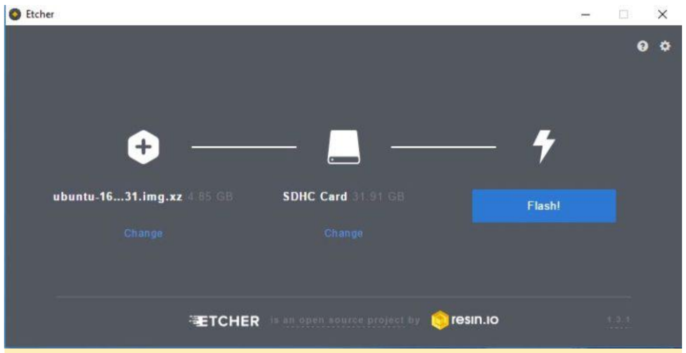

Etcher
ubuntu-16...31.img.xz 4.85 GB
SDHC Card 31.91 GB
Flash!
Change
Change
ETCHER is an open source project by resin.io

Figure 1 – Flashing a drive with Etcher

Download drivers for the USB-UART from here: https://goo.gl/opafDn

Extract and install the drivers. Once the drivers are installed, you will find it listed as a COM port in the device manager.

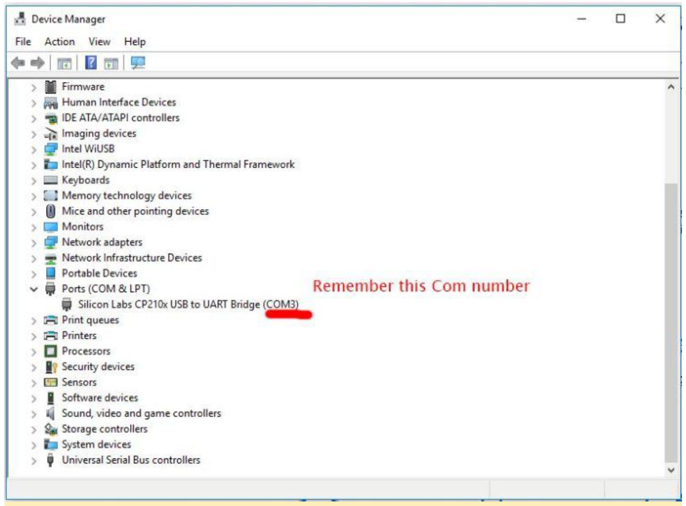
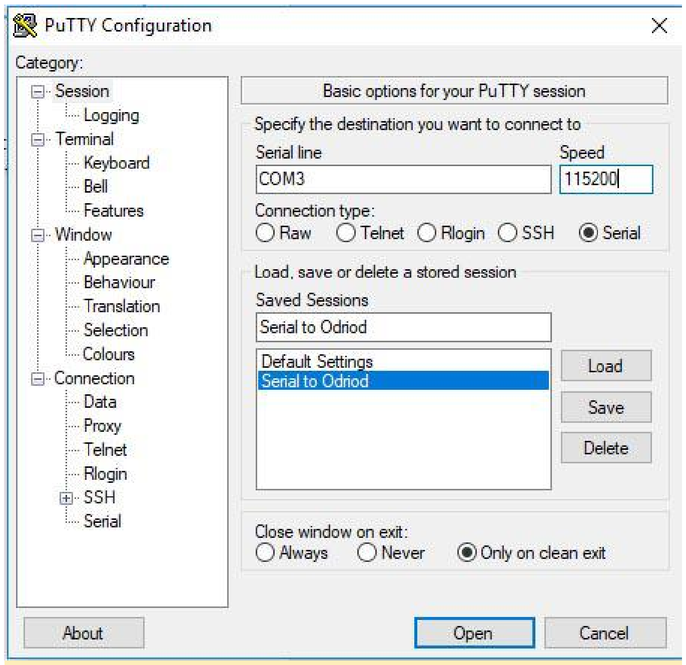


Figure 2 – Make note of the COM port of the USB-UART device

Next, download and install PuTTY from http://the.earth.li/~sgtatham/putty/latest/x86/putty.exe. PuTTY is a free serial console program we will be using to send commands to the HC2. PuTTY will prompt you for connection setting. Type the COM number from the device manager 115200 for the speed and check serial for the connection type. Save your configuration in case you need to use it in the future.



Specify the destination you want to connect to

Serial line: COM3 Speed: 115200

Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Load, save or delete a stored session

Saved Sessions: Serial to Odriod

Default Settings
Serial to Odriod

Close window on exit: Always Never [✓ Only on clean exit]

Figure 3 – PuTTY configuration

Configuration

Click “Open” in PuTTY and connect power to the HC2. The console window will show all the boot-up messages and eventually give you a login prompt. The default login info is:

User: odroid Password: odroid

Root Access credentials:

User: root Password: odroid

We are going to login and install updates first. The sudo command is “Super User Do” command to run commands as Admin or Root. Type the following commands:

```
$ odroid login: odroid
$ Password: odroid

$ sudo apt update
$ sudo apt upgrade
$ sudo apt dist-upgrade
$ sudo apt install linux-image-xu3
```

When you install the linux-image-xu3, it will throw out a prompt telling you that updating the kernel is dangerous. Do you want to abort the update? Select “NO.” Sometimes it will throw errors about files being in use on your first boot. If it does, enter the following command, and pick up where you left off:

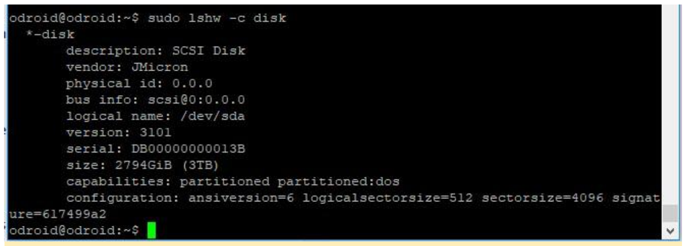
```
$ sudo reboot
```

Partition the Hard Drive

Log back in and type:

```
$ sudo apt-get install lshw
$ sudo lshw -C disk
```

This will list all attached storage.



```
odroid@odroid:~$ sudo lshw -c disk
  *-disk
       description: SCSI Disk
       vendor: JMicron
       physical id: 0.0.0
       bus info: scsi@0:0.0.0
       logical name: /dev/sda
       version: 3101
       serial: DB00000000013B
       size: 2794GiB (3TB)
       capabilities: partitioned partitioned:dos
       configuration: ansiversion=6 logicalsectorsize=512 sectorsize=4096 signat
ure=617499a2
odroid@odroid:~$
```

Figure 4 – Listing the attach storage

You’ll want the logical name for the hard drive in this case “/dev/sda”. Next, we will partition the disk:

```
$ sudo fdisk /dev/sda
```

Create a new partition type by pressing “n” for a new partition, “p” for primary partition, then “1”. For the partition start and end hit enter twice to use the defaults of the entire drive, then type “w” to write the partition table and exit fdisk.

Format the hard drive

In this guide, we are going to format the hard drive as ext4. If you ever have to recover the data on the drive, ext4 is not natively supported by Windows. This is not major issue since Ubuntu/Nextcloud will be handling all of the read/write operations. If you need the drive to be directly compatible with Windows or Mac, format it as vfat(fat32). Be aware that vfat can only handle files up to 4GB and partitions up to 2TB. Make multiple partitions to work around the partition size limit.

Type the following command to format the drive:

```
$ sudo mkfs -t ext4 /dev/sda1
```

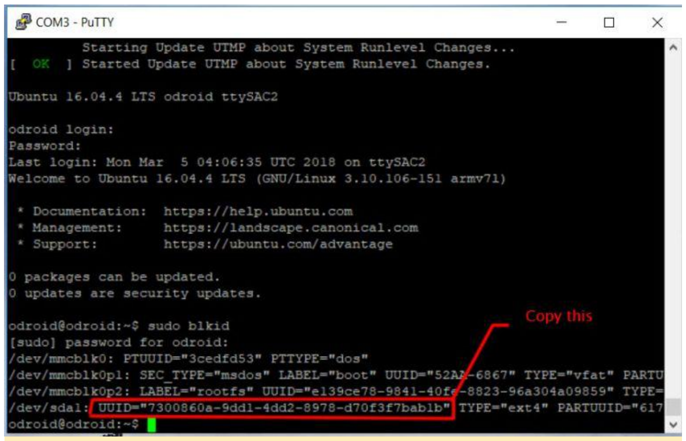
When it finishes formatting, we are going to set up the hard drive to mount on boot. We are going to create a directory in Ubuntu to mount the hard drive to. I am naming mine “SATAHD” but you can use any name you want:

```
$ sudo mkdir /media/SATAHD
```

Next, we need the UUID of the hard drive. When a hard drive is partitioned, a UUID is generated and assigned to the partition. Our hard drive is physically identified as "/dev/sda1" with "sda" being the drive and "1" being the first partition. If the hard drive was swapped, /dev/sda1 would be the first partition of the new drive. However the UUID of the partition is unique to only that partition. If you want Ubuntu to automatically mount whatever drive you put in it as SATAHD you may use /dev/sda1 instead of the UUID:

```
$ sudo blkid
```

Search the results for "/dev/sda1 uuid=", then copy the UUID to notepad or scratch paper.



```
COM3 - PuTTY
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Started Update UTMP about System Runlevel Changes.

Ubuntu 16.04.4 LTS odroid ttySAC2

odroid login:
Password:
Last login: Mon Mar  5 04:06:35 UTC 2018 on ttySAC2
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 3.10.106-151 armv7l)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

odroid@odroid:~$ sudo blkid
[sudo] password for odroid:
/dev/mmcblk0: PTUUID="3cedfd53" PTTYPE="dos"
/dev/mmcblk0p1: SEC_TYPE="msdos" LABEL="boot" UUID="52AA-6867" TYPE="vfat" PARTU
/dev/mmcblk0p2: LABEL="rootfs" UUID="e139ce78-9841-40fe-8823-96a304a09859" TYPE=
/dev/sda1: UUID="7300860a-9dd1-4dd2-8978-d70f3f7bab1b" TYPE="ext4" PARTUUID="617
odroid@odroid:~$
```

Figure 5 – Make note of the UUID of the drive

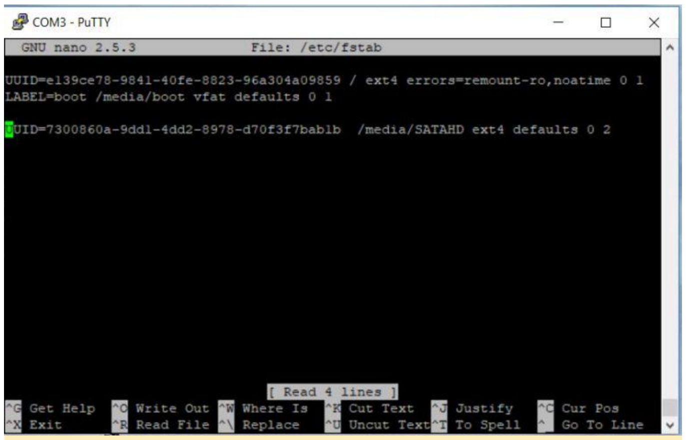
Next, we are going to set the drive to automatically mount on boot. The "sudo nano" command is a terminal based text editor:

```
$ sudo nano -Bw /etc/fstab
```

Use your arrow keys to go down to the bottom of the file and add the following line:

```
UUID=7300860a-9dd1-4dd2-8978-d70f3f7bab1b
/media/SATAHD EXT4 defaults 0 2
```

Use the UUID you copied in the last step instead of the one I used. "/media/SATAHD" is the mount point and the directory we created earlier. ext4 is the file system we used in formatting the drive. If you used something other than ext4, substitute the correct file system.



```
COM3 - PuTTY
  GNU nano 2.5.3              File: /etc/fstab

UUID=e139ce78-9841-40fe-8823-96a304a09859 / ext4 errors=remount-ro,noatime 0 1
LABEL=boot /media/boot vfat defaults 0 1

UUID=7300860a-9dd1-4dd2-8978-d70f3f7bab1b  /media/SATAHD ext4 defaults 0 2

                                [ Read 4 lines ]
^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 6 – Editing /etc/fstab to auto-mount the drive on boot

Hold CTRL+X to exit. Press "Y" to save changes and Enter to save over the old file, then reboot the computer:

```
$ sudo reboot
```

Installing Nextcloud

We are going to install Nextcloud with a snap package. If you want to manually install and configure Nextcloud, the documentation is available at http://goo.gl/DS5ZjY. Type the following command to install it:

```
$ sudo snap install nextcloud
```

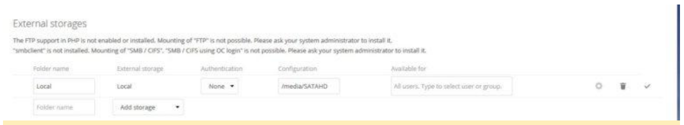
Nextcloud has a list of authorized domains or addresses. By default, the first IP address you access Nextcloud from will be added to that list. After that you must manually add addresses. I prefer to have my router automatically give the same IP address to the HC2 using the MAC address. You can set your IP address to be static self-assigned instead of DHCP if your router doesn't support assigning IP addresses this way.

To list the eth0 IP address of your HC2, type the following command:

```
$ ifconfig -a
```

Open your web browser and type in the IP address of your HC2. You will be greeted with a "Make a new admin account" screen. Enter your choice of username and password. On the next screen, close the "Welcome to Nextcloud" pop up. Click the gear in the top right corner. Select "apps." Scroll down the list of apps until you find “External storage support.” On the right side, click the Enable button.

Click the gear in the top right corner again. Select “Admin.” On the left side bar select “External Storage.” Click the “Add Storage” drop down box and select “Local.” Fill in the location as “/media/SATAHD” and hit the check box on the right to save the changes.


External storages

The FTP support in PHP is not enabled or installed. Mounting of "FTP" is not possible. Please ask your system administrator to install it.
"smbclient" is not installed. Mounting of "SMB / CIFS", "SMB / CIFS using OC login" is not possible. Please ask your system administrator to install it.

Folder name	External storage	Authentication	Configuration	Available for
Local	Local	None	/media/SATAHD	All users. Type to select user or group.
Folder name	Add storage			

Figure 7 – Adding storage in Nextcloud

Your hard drive will now appear on the home screen as “Local.” If you need to change the server’s config.php file, type:

```
$ sudo nano
/var/snap/nextcloud/current/nextcloud/config/c
onfig.php
```

The setting you will most likely have to mess with is the trusted domains, should your IP address change. As a good security practice you should change your password for the ODROID account and root. Type:

```
$ sudo passwd
$ and
$ passwd
```

You can enable HTTPS, but web browsers won’t like self-signed certificates. Do it right and use Let’s Encrypt. You can also set up your Nextcloud to be accessible through a Dynamic DNS but that will be another tutorial. For comments, questions, and suggestions, please visit the original article at **http://autonomousdev.net/index.php/2018/03/06/odroid-hc1-2-ubuntu-nas-with-nextcloud-setup-guide/**.

Further reading

https://help.nextcloud.com/t/adding-a-new-trusted-domain/26 https://github.com/nextcloud/nextcloud-snap
https://docs.nextcloud.com/server/13/admin_manual/

Control Any Electrical Device With An ODROID-C2: A Sample Project

April 1, 2018 By Miltiadis Melissas (miltos) ODROID-C2, Tinkering

It was always a 20th-century dream that there will be an era when every electrical apparatus at home will be controlled by a single click from any web enabled device, such as a PC, tablet, or smart TV, from everywhere. This era has come, and today we will present you with a way to control any electrical device with a single click from any other device that has access to the web.

We will use a spotlight as an example, but this could be easily substituted with a refrigerator, a washing machine, or an electric coffee pot, for example. We did one simplification, however, which is to electrify the spotlight with 12V DC instead of 220V AC current, primarily for safety reasons. We encourage the users of this guide to do the same, as it is very easy to expose oneself to hazardous electric shocks!

The relay module we use with the **ODROID-C2** in this project can easily be connected to a 220V power source, driving any electric device (up to 10A). Experienced users can try to work with these voltages, making sure to take all safety precautions. Let us delve into the endless potential of the ODROID-C2.

Hardware requirements

- ODROID-C2 (http://bit.ly/1oTJBya)
- 5V 1/2/4/8 Channel Relay Board Module ARM AVR DSP PIC (http://ebay.eu/2ncLWD8)
- Lamba JM/84211 3W 3000K 12V or any other compatible spotlight
- Dupon wires female-to-female, male-to-male (http://ebay.eu/2mDWf6Q)
- 4 X LS 14500 3.7V 2300mAh Li-ion batteries (http://ebay.eu/2m0F7EI)

Software requirements

- Ubuntu 16.04 v2.0 from Hardkernel (http://bit.ly/2cBibbk)
- Python 2.3 or 3.3 preinstalled with Ubuntu
- WiringPi Library for controlling the ODROID-C2 GPIO pins. You can learn how to install this at http://bit.ly/2ba6h8o
- CoffeeCup Free HTML Editor (http://bit.ly/2lCxgB8)
- PuTTY * – We are going to need to be able to connect to our ODROID-C2 via SSH, and PuTTY is the perfect client to do this (http://bit.ly/2kFVngX)
- FileZilla – We are going to need a way to transfer files onto the ODROID-C2 using SFTP, which is FTP over SSH (http://bit.ly/1gEw9op)

Connecting it together

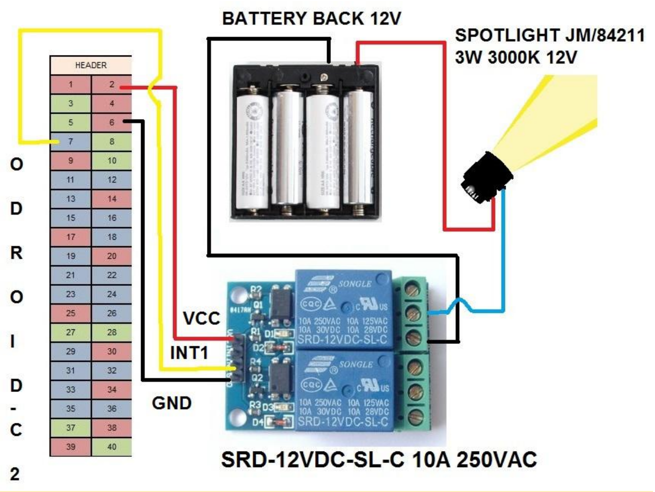


Figure 1 – Block diagram

The design of this project is very simple, and the Songle relay plays the most critical role. We have connected the GND of the Songle relay with the pin6 of ODROID-C2 (GND). The VCC pin of the relay is connected directly to pin2 of ODROID-C2, which provides 5V to this circuit and electrifies the electric coil of the relay). Finally, the INT1 pin of the relay is connected with pin7 of our ODROID, which is the pin that actually controls the relay, which is the ONs and OFFs of this device. From the other side of the Songle relay, there is a simple switch on which we have connected the spotlight through the battery or the mains. Please refer to the schematic in Figure 1 for a clear idea of this circuity. As a technical reference regarding the ODROID-C2 pins, we have used the excellent PIN Map provided by Hardkernel at http://bit.ly/2aXAlmt. According to this map, pin2=5V, pin6=GND and pin7=GPIOX.BIT21 (General Purpose Input/Output Pin). All of the connections were made by using the Dupon female-to-female, male-to-male or male-to-female wires. Now that our hardware is ready, let us see how to build the software and bring it all together.

Designing a simple web page

We used the free CoffeeCup HTML editor to design a simple HTML web page for controlling the spotlight. On this page, we added the images of two buttons in order to control the spotlight, represented by the ON and the OFF buttons. Please refer to Figure 2 for a view of this page. The whole project is controlled by using a Web UI. In order to achieve this, we have hyperlinked those buttons to the relevant Python scripts songleon.py and songleoff.py that control the ON and OFF of the spotlight. Instructions on how to write those programs in Python are provided on the section called “Connecting the Application to the Web below.

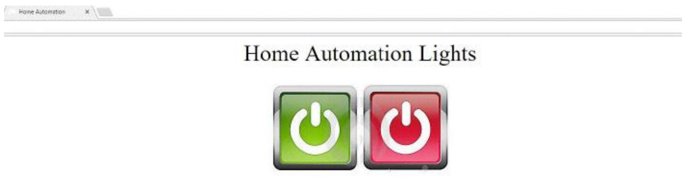


Figure 2 – Web page

When you finally design your website, make sure that your home page is called index.php and not index.html, just to keep things uniform. However, we are only going to be using two PHP scripts, songleon.php and songleoff.php, to control the spotlight. The Python and PHP code we need to write is very simple and well documented.

Installing the server

In order to use the ODROID-C2 as a web server in this project, we have to install all of the necessary server software components. In addition, since we want a simple HTML server, we will install Apache with PHP (server-side scripting-language) support on the ODROID-C2. The following steps can be performed with PuTTY. Accessing the ODROID-C2 with this SSH client is well documented. All you need is the ODROID-C2’s IP address.

Apache server software is the most widely used web server software today. Here’s how to install Apache with PHP support:

```
odroid@odroid:~# sudo apt-get install apache2
php libapache2-mod-php
```

When prompted to continue, enter “y” for yes. Next, enable and start Apache:

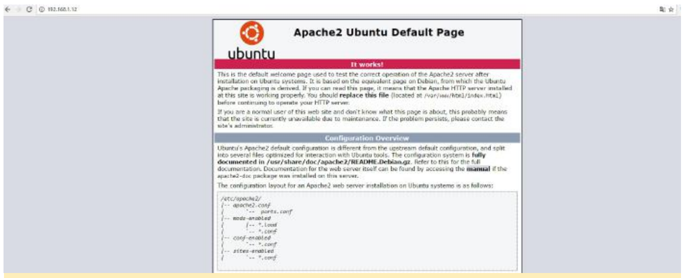
```
odroid@odroid:~# systemctl enable apache2
 odroid@odroid:~# systemctl start apache2
 odroid@odroid:~# systemctl status apache2
```

Test Apache

Open your web browser and navigate to http://localhost/ or http:///. This is the address of your ODROID-C2 on your local network. You can find it by just typing:

```
odroid@odroid:~# ifconfig
```

You will see a page similar to one shown in Figure 3.



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

Figure 3 – Apache test

Test PHP

To test PHP, create a sample testphp.php file in Apache document root folder:

```
odroid@odroid:~# sudo nano
/var/www/html/testphp.php
```

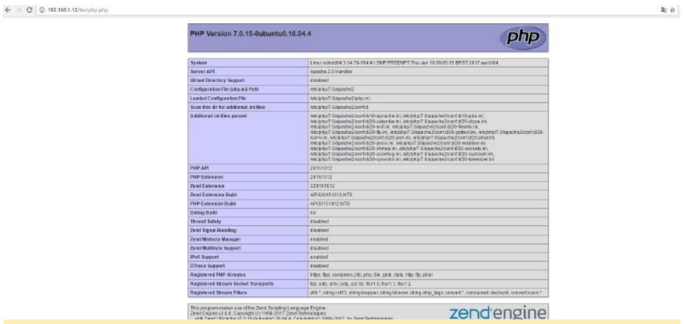
Add the following lines and save the file:

```
```

Restart the Apache service.

```
$ sudo systemctl restart apache2
```

Navigate to http://server-ip-address/testphp.php. It will display all the details about PHP such as, version, build date and commands, to name a few. Pleaser refer to Figure 4 below.



PHP Version 7.0.15-0ubuntu0.16.04.4

System	Linux odroid64 3.14.79-104 #1 SMP PREEMPT Thu Jan 19 09:05:15 BRST 2017 aarch64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.0.0, Copyright (c) 1998-2017 Zend Technologies
with Zend OPcache v7.0.15-0ubuntu0.16.04.4, Copyright (c) 1999-2017, by Zend Technologies

Figure 4 – PHP information

Installing FTP

Install an FTP server such as vsftpd using the following command:

```
$ sudo apt-get install vsftpd
```

Edit the FTP configuration file by typing:

```
$ sudo nano /etc/vsftpd.conf
```

and make the the following changes:

- Hit ctrl+W and search for anonymous_enable=YES, and change it to anonymous_enable=NO
- Remove the # from in front of local_enable=YES
- Remove the # from in front of write_enable=YES
- Skip to the bottom of the file and add force_dot_files=YES
- Hit ctrl+X to exit and enter y to save and hit to confirm:

Then, restart vsftpd:

```
$ sudo service vsftpd restart
```

Publishing to the web

By now, you should have a website that you can transfer over to the ODROID-C2. Once you have performed all the previous steps and have verified that you can view your website on another computer, we can move onto making the website turn ON our lamp using the 5V 1/2/4/8 Channel Relay Board Module ARM AVR DSP PIC.

Inside your website directory, create a new PHP file called songleon.php containing the following code snippet, then save the file:

```
```

Next, create a folder in the website directory called “scripts”, then create a subfolder inside it called "lights", and inside there, create a new file called songleon.py. This will be the python script that turns our lamp on. Inside there, enter the following code, then save the file:

```
import wiringpi2 as odroid
 odroid.wiringPiSetup()
 odroid.pinMode(7,1)
 odroid.digitalWrite(7,0)
```

Go back to your web page in design/edit mode, and make sure the hyperlink for your "on" button links to the songleon.php. Now, when you click the button, the songleon.php script will execute the songleon.py python script, resulting in the lamp turning on. We are finally ready to make it turn off.

Inside the website directory, create a new file called songleoff.php. Inside this, file enter the following code snippet, then save it:

```
 <!--?php system("echo odroid | sudo -S python
/var/www/html/scripts/lights/songleoff.py");
header( 'Location: 'index.php' ) ; ?-->
```

Again, make sure your file path is the same, so that this works. Also, set your redirection rules to redirect to the page of your choice. Then, make a new file in the scriptslights folder called sognleoff.py. Inside this file, enter the following code, then save the file:

```
import wiringpi2 as odroid
odroid.wiringPiSetup()
odroid.pinMode(7,1)
odroid.digitalWrite(7,1)
```

Add a hyperlink to songleoff.php to your "off" button, which should make your lamp turn off. You now have a website that can control your lights!

Transfer to Apache

It is very easy to login into your ODROID-C2 apache web server with Filezilla as soon as you know the ODROID-C2's IP address. If you have previously logged in with SSH into your ODROID-C2 with PuTTY, you can find it out by typing:

```
odroid@odroid:~# ifconfig
```

You have to give your name and password of "odroid" and "odroid". You will be immediately taken to the ODROID-C2 root file system. From there, navigate to /var/www/html/ folder, and inside this directory, copy the files from your local drive to the above directory. Here are all the files and folders that you have to copy from this local directory:

- index.php
- songleon.php
- songleoff.php and finally the folder /scripts/lights/ with
- songleon.py and songleoff.py

Now you are done with the main part of project. One final word of advice: in order for you to have access to the execution of the scripts controlling the spotlight (songleon.py and songleoff.py), you have to change the permissions/rights of all the files and folders previously mentioned. We recommend just for the sake of this project to give them full access with read, write, and execute privileges for root:

```
$ sudo chown 755 /var/www/html/index/php
```

In addition, you have to modify the sudoers file with nano editor:

```
$ sudo nano /etc/sudoers
```

Provide your password for any modifications and add the following line:

```
www-data ALL=(ALL) NOPASSWD ALL after this
line: %sudo ALL=(ALL:ALL) ALL
```

Testing the applications

Let's see if everything is working. From your computer, laptop, or tablet, navigate to your ODROID-C2's IP address in the browser, and click the "on" button. Are your spotlights lightening up your room? Now it's the click the "off" button. Click it and see your room's spotlight switched off. Success!

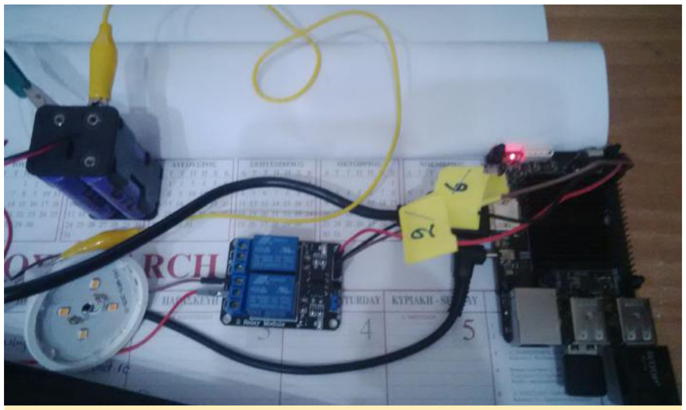


Figure 5 – Hardware setup

Final notes

We could give you further guidance on how to control any electrical device remotely from the office, during travelling, or you are in an emergency. This is not a difficult step now that you've got the basic circuit working in your local network, but be advised that such a step comes with a security risk. Hackers may be interested in controlling your server by taking advantage of a weak password, a commonly used port, or wrong router settings. For that reason, we advise you to make your network secure and even change the password for the user ODROID-C2 to a stronger one if you are implementing electric device control. You now have the knowledge you need to build something innovative and inspiring for you and your peers.

Prospectors, Miners and 49er's – Part 2: Dual GPU-CPU Mining on the XU4/MC1/HC1/HC2

April 1, 2018 By Edward Kisiel (@hominoid) Linux, Tutorial

Last month's article introduced Dual GPU-CPU mining on the Odroid XU4/MC1/HC1/HC2. This month we'll update the community on the progress of improvements to the original work and discuss some basic GPU tuning. The removal of all the OpenCL AMD dependencies and INTEL assembler for the OpenCL kernels and crypto algorithms is now complete for sgminer-arm. Genesis Mining also recently released a new version of sgminer-gm 5.5.6. Those changes have been incorporate into the completed newly released sgminer 5.5.6-ARM-RC1. Here is brief summary of the kernels and crypto algorithms that were modified for the Odroid and the test results.

ocl/build_kernel.c

algorithm/cryptonight.c

– INTEL assembler optimizations

algorithm/neoscrypt.c

– AMD architecture optimizations

kernel/cryptonight.cl

– AMD OpenCL extensions

kernel/equihash.cl

– AMD OpenCL extensions and AMD architecture optimizations

kernel/ethash.cl

– AMD OpenCL extensions

kernel/ethash-genoil.cl

– AMD architecture optimizations

kernel/ethash-new.cl

– AMD architecture optimizations

kernel/lyra2re.cl

– AMD OpenCL extensions

kernel/lyra2rev2.cl

– AMD OpenCL extensions

kernel/whirlpoolx.cl

– AMD architecture optimizations

kernel/wolf-aes.cl

– AMD OpenCL extensions

kernel/wolf-skein.cl

– AMD OpenCL extensions

Choices had to be made about specific coin algorithms and OpenCL kernels that had architecture specific setting (not AMD extensions) as indicated. 70% of the OpenCL kernels share one or more of the same AMD OpenCL extensions, that were modified and tested with the cryptonight kernel, which also uses 2 OpenCL helper kernels (wolf-aes.cl and wolf-skein.cl). It appears that ethash-new.cl is not used for any coins which would leave only 2 unproven in anyway, whirlpoolx.cl and ethash-genoil.cl. The others had only AMD and or Nvidia architecture optimizations that were removed. The most conservative approach possible was used in modifications so they would run on a wide range of current and future GPU's, but there is always room for technical and human error. The sgminer-arm implementation should be CPU and GPU agnostic which raises the possibility for adding some ARM-Mali optimization based on specific architectures (ARMv7, ARMv8, Mali-T628, Mali-T860) in the future.

Tuning the GPU

When first trying to figure out what the settings should be for a coin you haven't mined, start very conservatively with all of the setting and work your way up using trial and error until it starts to fail or the performance starts to drop. Here are some settings that are a good place to start:

```
./sgminer -k algorithm -o
stratum+tcp://your.pool.com:3333 -u user -p
password -I 3 -w 32 -d 0,1 --thread-
concurrency 8192
```

Keep in mind that on all ARM-Mali SOC, the GPU shares the main memory with the CPU so there is a dynamic between the two when your dual mining as well. That is why generally you loss some performance dual CPU/GPU compared to only mining on the CPU.

When you start sgminer-arm and the settings are wrong or too much for the GPU, it can manifest itself in a lot of different ways. The OpenCL kernel can crash, hang or indicate different error messages. Below is a typical error for a GPU tuning problem. It couldn't build the OpenCL kernel.

```
[19:28:21] Error -6: Creating Kernel from
program.  (clCreateKernel)
[19:28:21] Failed to init GPU thread 1,
disabling device 1
```

Another common error message is that the OpenCL kernel is trying to allocate more memory then it has available. These both indicate one or more of the GPU settings need to be reduced(Intensity, Work Size, Number of threads or Thread-concurrency).

```
[19:28:16] Maximum buffer memory device 0
supports says 522586112
[19:28:16] Your settings come to 536870912
```

When Dual Mining, get the GPU tuned and running by itself and then get the CPU tuned and running by itself. Then try to run both together but expect to adjust them again accordingly (usually the CPU). Most CPU mining software is going to try and use every system resource it can. You may have to manually set parameters for the CPU miner instead of letting it choose. Likewise, there are situations where there is such tight memory usage that any other normal process trying to start may cause a system problem (crash, errors etc). For CPU mining, none of the HK Image releases use swap by default so Hugh Pages can't be enabled. So in general, there should be similar performance regardless of the CPU miner software your using.

Cooling, Power Utilization and System Monitoring

You have to monitor CPU temperatures while tuning until you know what your mining rig setup is capable of with the crypto-algorithms your using. System damage can and is likely to occur without adequate cooling while mining! Generally speaking, OEM cooling is not sufficient without significantly reducing the CPU frequency. It is one of many reasons a system can crash or reboot while single or dual mining. Even small ambient temperature changes can significantly impact your system and cause damage. Monitor the ambient and system temperatures on a regular basis while mining. Use watchtemp.sh if you don't have some other means.

```
watchtemp.sh
#!/bin/bash
z=0
echo "T, Freq4,   Freq5,   Freq6,   Freq7,
T4, T5, T6, T7, TGPU"

while true :
do
     fa=`cat
/sys/devices/system/cpu/cpu4/cpufreq/scaling_c
ur_freq`
     fb=`cat
/sys/devices/system/cpu/cpu5/cpufreq/scaling_c
ur_freq`
     fc=`cat
/sys/devices/system/cpu/cpu6/cpufreq/scaling_c
ur_freq`
     fd=`cat
/sys/devices/system/cpu/cpu7/cpufreq/scaling_c
ur_freq`
     s1=`cat
/sys/devices/virtual/thermal/thermal_zone0/tem
p`
     s1t=$(($s1/1000))
     s2=`cat
/sys/devices/virtual/thermal/thermal_zone1/tem
p`
     s2t=$(($s2/1000))
     s3=`cat
/sys/devices/virtual/thermal/thermal_zone2/tem
p`
     s3t=$(($s3/1000))
     s4=`cat
/sys/devices/virtual/thermal/thermal_zone3/tem
p`
     s4t=$(($s4/1000))
         g1=`cat
/sys/devices/virtual/thermal/thermal_zone4/tem
p`
     g1t=$(($g1/1000))

     echo $z, $fa, $fb, $fc, $fd, $s1t, $s2t,
$s3t, $s4t, $g1t
     sleep 2
     (( z += 2 ))
done
```

No power utilization testing has been, only thermal testing. See last months article for a preliminary thermal test. Lots of resource are used simultaneously while dual mining and some crypto-algorithms use considerable more power than others. For example, scrypt2(VRM) mining uses approximately 20-25% more power then cryptonight(Monero) algorithm. Monitor or do a power utilization study for a better understanding of ARM-Mali power usage before or while dual mining different crypto-algorithms.

When dual mining be conservative so you allow the rest of the OS to function, keep an eye on temperature and be aware of power usage until you prove out both the CPU and GPU configurations and then you can lean into it more. Dual mining pushes these system to the limits. This is a new frontier for ARM SBCs, so keep in mind you are on the sharp edge of extreme system utilization.

Crypotnight (Monero Coin) testing on an Odroid-XU4 GPU only:

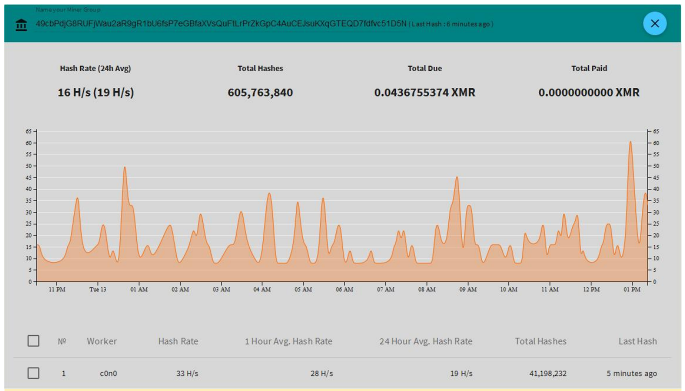
```
sgminer 5.5.6-ARM-RC1 - Started: [2018-03-13
03:06:15] - [0 days 12:38:48]
----------------------------------------------
----------------------------------
(5s):22.52 (avg):23.85h/s | A:700000  R:10000
HW:48  WU:0.187/m
ST: 1  SS: 0  NB: 421  LW: 48993  GF: 21  RF:
0
Connected to pool.supportxmr.com (stratum)
diff 5K as user
49cbPdjG8RUFjWau2aR9gR1bU6fsP7eGBfaXVsQuFtLrPr
ZkGpC4AuCEJsuKX
Block: e368fdd9...  Diff:905.5  Started:
[15:44:30]  Best share: 325K
----------------------------------------------
----------------------------------
[P]ool management [G]PU management [S]ettings
[D]isplay options [Q]uit
GPU 0:                |  13.41/ 13.40h/s | R:
2.6% HW:24 WU:0.103/m I: 7
GPU 1:                |  10.45/ 10.45h/s | R:
0.0% HW:24 WU:0.084/m I: 7
```

```
----------------------------------------------
----------------------------------
[13:41:07] Accepted 035e18c0 Diff 19.5K/5K GPU
0
[13:54:00] Accepted 058d63f3 Diff 11.8K/5K GPU
0
[13:55:45] Accepted 0b3c0178 Diff 5.83K/5K GPU
0
[14:04:37] Accepted 054a51d3 Diff 12.4K/5K GPU
1
[14:15:02] Accepted 014287d0 Diff 52K/5K GPU 1
[14:19:56] Accepted 018036d4 Diff 43.7K/5K GPU
1
[14:20:16] pool.supportxmr.com stale share
detected, submitting (user)
[14:20:16] Accepted 052596c8 Diff 12.7K/5K GPU
0
[14:36:38] Stratum connection to
pool.supportxmr.com interrupted
[14:40:09] Stratum connection to
pool.supportxmr.com interrupted
[14:42:19] Accepted 0b9ad8d2 Diff 5.65K/5K GPU
0
[14:44:23] Accepted 04b3a1c8 Diff 13.9K/5K GPU
0
[14:44:43] Accepted 011e41a0 Diff 58.6K/5K GPU
0
[14:54:26] pool.supportxmr.com stale share
detected, submitting (user)
[14:54:26] Accepted 060914e6 Diff 10.9K/5K GPU
0
[14:57:14] pool.supportxmr.com stale share
detected, submitting (user)
[14:57:14] Accepted 0a2b9f80 Diff 6.44K/5K GPU
1
[15:04:20] pool.supportxmr.com stale share
detected, submitting (user)
[15:04:20] Accepted 076a4aeb Diff 8.84K/5K GPU
0
[15:05:37] Accepted 09d5465e Diff 6.66K/5K GPU
0
[15:10:32] Accepted 0a066760 Diff 6.54K/5K GPU
1
[15:16:15] pool.supportxmr.com stale share
detected, submitting (user)
[15:16:16] Accepted 06082f75 Diff 10.9K/5K GPU
1
[15:18:06] pool.supportxmr.com stale share
detected, submitting (user)
[15:18:06] Accepted 0ce5243a Diff 5.08K/5K GPU
1
[15:18:30] Accepted ccf47695 Diff 81.9K/5K GPU
1
[15:30:46] Accepted 0857db6a Diff 7.86K/5K GPU
0
[15:30:57] Accepted 090f5ea6 Diff 7.23K/5K GPU
1
[15:31:34] Accepted 071e4b0f Diff 9.21K/5K GPU
1
[15:38:34] Accepted 0a0c5007 Diff 6.52K/5K GPU
0
[15:44:56] Accepted 88acc80f Diff 123K/5K GPU
0
```

The test was run for more than 12 hours and everything ran smoothly. The summary shows 2 actual rejected shares I have a relatively slow Internet connection, so any Stratum server disconnects, and purported stale shares are not unusual.

```
Summary of runtime statistics:

[15:46:02] Started at [2018-03-13 03:06:15]
[15:46:02] Pool:
stratum+tcp://pool.supportxmr.com:3333
[15:46:02] Runtime: 12 hrs : 39 mins : 47 secs
[15:46:02] Average hashrate: 0.0 Kilohash/s
[15:46:02] Solved blocks: 0
[15:46:02] Best share difficulty: 325K
[15:46:02] Share submissions: 142
[15:46:02] Accepted shares: 140
[15:46:02] Rejected shares: 2
[15:46:02] Accepted difficulty shares: 700000
[15:46:02] Rejected difficulty shares: 10000
[15:46:02] Reject ratio: 1.4%
[15:46:02] Hardware errors: 48
[15:46:02] Utility (accepted shares / min):
0.18/min
[15:46:02] Work Utility (diff1 shares solved /
min): 0.19/min

[15:46:02] Stale submissions discarded due to
new blocks: 0
[15:46:02] Unable to get work from server
occasions: 21
[15:46:02] Work items generated locally: 49055
[15:46:02] Submitting work remotely delay
occasions: 0
[15:46:02] New blocks detected on network: 421

[15:46:02] Summary of per device statistics:

[15:46:02] GPU0                | (5s):13.48
(avg):13.40h/s | A:380000 R:10000 HW:24
```

```
WU:0.103/m
[15:46:02] GPU1                  | (5s):10.45
(avg):10.45h/s | A:320000 R:0 HW:24 WU:0.084/m
[15:46:02]
```


Name your Miner Group

49cbPdjG8RUFjWau2aR9gR1bU6fsP7eGBfaXVsQuFtLrPrZkGpC4AuCEJsuKXqGTEQD7fdfvc51D5N (Last Hash : 6 minutes ago)

Hash Rate (24h Avg)	Total Hashes	Total Due	Total Paid
16 H/s (19 H/s)	605,763,840	0.0436755374 XMR	0.0000000000 XMR

№	Worker	Hash Rate	1 Hour Avg. Hash Rate	24 Hour Avg. Hash Rate	Total Hashes	Last Hash
1	c0n0	33 H/s	28 H/s	19 H/s	41,198,232	5 minutes ago

0 5 10 15 20 25 30 35 40 45 50 55 60 65
11 PM Tue 13 01 AM 02 AM 03 AM 04 AM 05 AM 06 AM 07 AM 08 AM 09 AM 10 AM 11 AM 12 PM 01 PM

Figure 1 – The pool results verify the summary results

With the increasing modification of sgminer a git was setup for ease of use and future modification. Likewise, the installation process has changed and no longer requires any AMD_SDK, only the ARM Computer Vision and Machine Learning library. Below is the new procedure.

Download and install the latest ARM Computer Vision and Machine Learning library from https://github.com/ARM-software/ComputeLibrary/releases. Note that they have separated the Linux and Android libraries so that it now fits on a 8GB SD card. Use the following command to extract the files:

```
$ tar -xvzf `filename to extract`
```

Next, install the dependencies and copy the OpenCL headers:

```
$ apt-get install automake autoconf pkg-config
libcurl4-openssl-dev libjansson-dev libssl-dev
libgmp-dev make g++ git libgmp-dev
libncurses5-dev libtool opencl-headers mali-
fbdev
$ cp ./arm_compute-v18.03-bin-
linux/include/CL/* /usr/include/CL/
```

Download sgminer-5.5.6-ARM-RC1 with the following command:

```
$ git clone
https://github.com/hominoids/sgminer-arm
```

Then, compile the source code:

```
$ cd sgminer-arm
$ git submodule init
$ git submodule update
$ autoreconf -fi
$ CFLAGS="-Os -Wall -march=native -std=gnu99 -
mfpu=neon" ./configure --disable-git-version -
-disable-adl --disable-adl-checks
```

You can optionally use the following command to be more explicit as to where you placed the library and headers:

```
$ CFLAGS="-Os -Wall -march=native -std=gnu99 -
mfpu=neon -I/opt/arm_compute-v18.03-bin-
linux/include/CL" LDFLAGS="-L/opt/arm_compute-
v18.03-bin-linux/lib/linux-armv7a-neon-cl"
./configure --disable-git-version --disable-
adl --disable-adl-checks

$ make -j5
```

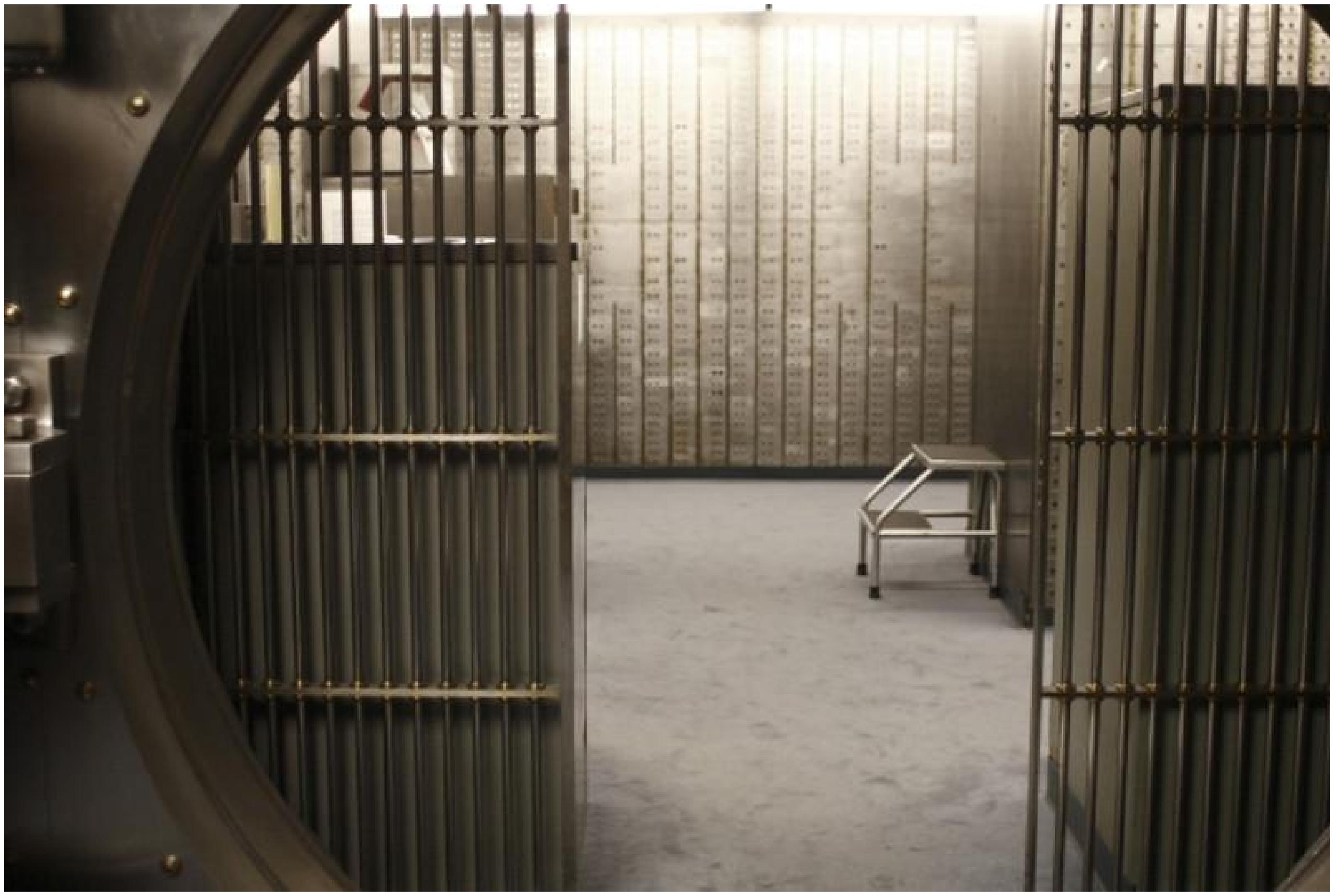
Here is the script and settings used for the testing of XMR-Monero coin using the cryptonight algorithm:

```
#!/bin/bash

export GPU_FORCE_64BIT_PTR=1
export GPU_USE_SYNC_OBJECTS=1
export GPU_MAX_ALLOC_PERCENT=100
export GPU_SINGLE_ALLOC_PERCENT=100
export GPU_MAX_HEAP_SIZE=100

./sgminer -k cryptonight -o
stratum+tcp://pool.supportxmr.com:3333 -u
username -p password -I 7 -w 32 -d 0,1 --
thread-concurrency 8192 --monero --pool-no-
keepalive
```

The ODROID Community now has the only multi-algorithm Linux ARM-Mali OpenCL GPU miner that I'm aware of in the crypto community! Remember to check the forum for more information and updates at https://forum.odroid.com/viewtopic.php?f=98&t=29571.

Secure Storage: Creating an Encrypted File System in Linux

April 1, 2018 By @Anand.moon ODROID-HC2



In Linux, encryption is done through dm-crypt using LUKS as the key setup using kernel crypto API. This feature is part of Linux Kernel 4.14.18-106 and above, additionally we need Exynos5422 Slim SSS (Security Sub-System) driver which supports AES, SHA-1, SHA-256, HMAC-SHA-1, and HMAC-SHA-256 encryptions. The device-mapper target provides transparent encryption of block devices using the kernel crypto API.

- AES cipher with support of aes-cbc/aes-ctr
- Cipher-block chaining (CBC)
- Counter (CTR) known as integer counter mode (ICM) and segmented integer counter (SIC) mode
- ESSIV ("Encrypted salt-sector initialization vector") allows the system to create IVs based on a hash including the sector number and encryption key
- SHA-256 is the hashing algorithm used for key derivation
- XTS is counter-oriented chaining mode
- PLAIN64 is an IV generation mechanism that simply passes the 64-bit sector index directly to the chaining algorithm as the IV

```
Parameters: < cipher > < key > < iv_offset > <
device path > < offset > [< #opt_params > <
opt_params >]
< cipher >
Encryption cipher, encryption mode and Initial
Vector (IV) generator.
The cipher specifications format is:
cipher[ :keycount ]-chainmode-ivmode[ :ivopts
]
Examples:
aes-cbc-essiv:sha256
aes-xts-plain64
Serpent-xts-plain64

Cipher format also supports direct
specification with kernel crypt API
format (selected by capi: prefix). The IV
specification is the same
```

```
as for the first format type.
This format is mainly used for specification
of authenticated modes.
The crypto API cipher specifications format
is:
Capi:cipher_api_spec-ivmode[ :ivopts ]
Examples:
capi:cbc(aes)-essiv:sha256
capi:xts(aes)-plain64
Examples of authenticated modes:
capi:gcm(aes)-random
capi:authenc(hmac(sha256),xts(aes))-random
capi:rfc7539(chacha20,poly1305)-random
```

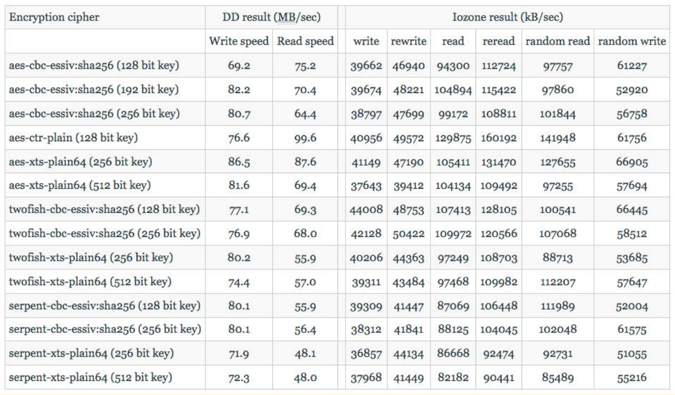
Cryptsetup benchmark testing with DRAM

```
# root@odroid:~# cryptsetup benchmark

# Tests are approximate using memory only (no
storage IO).
PBKDF2-sha1 297552 iterations per second
PBKDF2-sha256 195338 iterations per second
PBKDF2-sha512 125068 iterations per second
PBKDF2-ripemd160 247305 iterations per second
PBKDF2-whirlpool 27935 iterations per second
# Algorithm | Key | Encryption | Decryption
aes-cbc 128b 73.8 MiB/s 97.7 MiB/s
serpent-cbc 128b 40.9 MiB/s 42.9 MiB/s
twofish-cbc 128b 58.0 MiB/s 62.0 MiB/s
aes-cbc 256b 59.8 MiB/s 74.0 MiB/s
serpent-cbc 256b 41.5 MiB/s 42.7 MiB/s
twofish-cbc 256b 59.1 MiB/s 62.0 MiB/s
aes-xts 256b 110.6 MiB/s 95.2 MiB/s
serpent-xts 256b 41.6 MiB/s 42.2 MiB/s
twofish-xts 256b 59.7 MiB/s 61.9 MiB/s
aes-xts 512b 86.1 MiB/s 72.5 MiB/s
serpent-xts 512b 42.1 MiB/s 42.4 MiB/s
twofish-xts 512b 60.7 MiB/s 61.6 MiB/s
```

Cryptsetup benchmark testing with HDD

Figure 1 shows test results from an ODROID-HC2 using a WD 4TB 5400RPM NAS HDD, which may vary depending on the type of hard drive used:

```
$ iozone -e -I -a -s 100M -r 4k -r 16k -r 512k
-r 1024k -r 16384k -i 0 -i 1 -i 2
```



Encryption cipher	DD result (MB/sec)		Iozone result (kB/sec)					
	Write speed	Read speed	write	rewrite	read	reread	random read	random write
aes-cbc-essiv:sha256 (128 bit key)	69.2	75.2	39662	46940	94300	112724	97757	61227
aes-cbc-essiv:sha256 (192 bit key)	82.2	70.4	39674	48221	104894	115422	97860	52920
aes-cbc-essiv:sha256 (256 bit key)	80.7	64.4	38797	47699	99172	108811	101844	56758
aes-ctr-plain (128 bit key)	76.6	99.6	40956	49572	129875	160192	141948	61756
aes-xts-plain64 (256 bit key)	86.5	87.6	41149	47190	105411	131470	127655	66905
aes-xts-plain64 (512 bit key)	81.6	69.4	37643	39412	104134	109492	97255	57694
twofish-cbc-essiv:sha256 (128 bit key)	77.1	69.3	44008	48753	107413	128105	100541	66445
twofish-cbc-essiv:sha256 (256 bit key)	76.9	68.0	42128	50422	109972	120566	107068	58512
twofish-xts-plain64 (256 bit key)	80.2	55.9	40206	44363	97249	108703	88713	53685
twofish-xts-plain64 (512 bit key)	74.4	57.0	39311	43484	97468	109982	112207	57647
serpent-cbc-essiv:sha256 (128 bit key)	80.1	55.9	39309	41447	87069	106448	111989	52004
serpent-cbc-essiv:sha256 (256 bit key)	80.1	56.4	38312	41841	88125	104045	102048	61575
serpent-xts-plain64 (256 bit key)	71.9	48.1	36857	44134	86668	92474	92731	51055
serpent-xts-plain64 (512 bit key)	72.3	48.0	37968	41449	82182	90441	85489	55216

Figure 1 – Cryptsetup benchmark test results for an ODROID-HC2

Encrypt the hard drive using cryptsetup

Install cryptsetup, and so as not to need rebooting, start the dm-crypt modules.

```
$ sudo apt-get install cryptsetup
$ sudo modprobe dm-crypt sha256 aes
```

Test verify the cryptsetup and dm-crypt are working:

```
$ fallocate -l 128MiB /tmp/test.bin
$ dd if=/dev/urandom of=/tmp/testkey.key
bs=128 count=1
$ sync
$ cryptsetup luksFormat --debug -q -d
/tmp/testkey.key --cipher aes-cbc-essiv:sha256
-h sha256 -s 128 /tmp/test.bin

$ fallocate -l 128MiB /tmp/test.bin
$ dd if=/dev/urandom of=/tmp/testkey.key
bs=128 count=1
$ sync
$ cryptsetup luksFormat --debug -q -d
/tmp/testkey.key --cipher aes-ctr-plain -h
sha256 -s 128 /tmp/test.bin
```

Once the you verify the cryptsetup is working fine, you can start encrypting the disk. Note that this is full disk encryption, so the disk needs to be formatted.

```
$ sudo wipefs -a /dev/sda1
/dev/sda1: 6 bytes were erased at offset
0x00000000 (crypto_LUKS): 4c 55 4b 53 ba be
```

Create a key to unlock the volume

Luks encryption supports multiple keys. These keys can be passwords entered interactively, or key files passed as an argument while unlocking the encrypted partition.

```
$ sudo dd if=/dev/urandom of=/root/keyfile
bs=1024 count=4
$ sudo chmod 400 /root/keyfile
```

To create the encrypted partition on /dev/sda1, luks is used. The encryption of the partition will be managed using the cryptsetup command.

```
$ sudo cryptsetup --verify-passphrase
luksFormat /dev/sda1 -c aes-cbc-essiv:sha256 -
h sha256 -s 128
```

This will ask for passphrase which should be long (more than 8 characters), which should be noted. The following steps demonstrate how to open up the encrypted drive and map the dp-crypt to filesystem Next, unlock the drive using the passphrase we just gave,then create a filesystem on the device.

```
$ sudo cryptsetup luksOpen /dev/sda1
securebackup
```

Format the partition:

```
$ sudo mkfs -t ext4 -m 1
/dev/mapper/securebackup
```

Add a luks key to support auto mounting at boot time:

```
$ sudo cryptsetup -v luksClose securebackup
$ sudo cryptsetup luksAddKey /dev/sda1
/root/keyfile
```

Update the /etc/crypttab file, to refer to the keyfile:

```
$ cat /etc/crypttab
# < target name > < source device > < key file
> < options >
securebackup /dev/sda1 /root/keyfile luks
```

We need to tell the dm-crypt subsystem that this stick must be mounted before the encrypted HDD partition. To do this, open the /etc/default/cryptdisks file and look for the line CRYPTDISKS_MOUNT=“”:

```
$ cat /etc/default/cryptdisks
# Run cryptdisks initscripts at startup?
Default is Yes.
CRYPTDISKS_ENABLE=Yes

# Mountpoints to mount, before cryptsetup is
invoked at initscripts. Takes
# mountpoins which are configured in
/etc/fstab as arguments. Separate
# mountpoints by space.
# This is useful for keyfiles on removable
media. Default is unset.
CRYPTDISKS_MOUNT="/root/keyfile"

# Default check script. Takes effect, if the
'check' option is set in crypttab
# without a value.
CRYPTDISKS_CHECK=blkid

# Default precheck script. Takes effect, if
the 'precheck' option is set in
# crypttab without a value.
# Default is 'un_blkid' for plain dm-crypt
devices if unset here.
CRYPTDISKS_PRECHECK=
```

Verify that the drive is mapped to the crypto device:

```
$ sudo cryptsetup luksOpen /dev/sda1
securebackup
$ lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
mmcblk1 179:0 0 29.8G 0 disk
|-mmcblk1p1 179:1 0 128M 0 part /media/boot
`-mmcblk1p2 179:2 0 29.7G 0 part /
sda 8:0 0 149G 0 disk
`-sda1 8:1 0 149G 0 part
`-securebackup 254:0 0 149G 0 crypt
```

In order to auto mount the disk on next boot up, you need to update the /etc/fstab entry:

```
$ mkdir -p /media/secure
$ sudo cat /etc/fstab
UUID=e139ce78-9841-40fe-8823-96a304a09859 /
ext4 errors=remount-ro,noatime 0 1
LABEL=boot /media/boot vfat defaults 0 1
/dev/mapper/securebackup /media/secure ext4
defaults,rw 0 2
```

You will be able to manually mount drive the disk if the previous steps were successful:

```
$ mount /dev/mapper/securebackup /media/secure
```

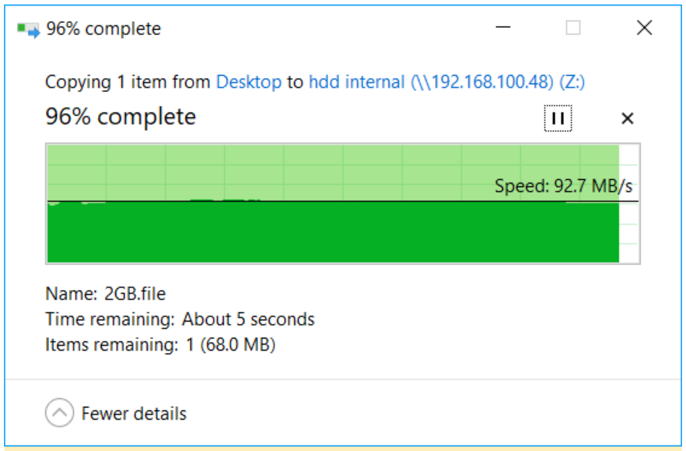
CIFS/Samba performance on an encrypted HDD

Figures 2 – 7 show performance using the following hardware configuration:

- HC2 + ubuntu-16.04.3-4.14-minimal-odroid-xu4-20171213.img.xz with updated 4.14.18-106 kernel
- 8GB MicroSD card
- Seagate 8TB HDD (ST8000AS0002)
- Encryption: aes-xts-plain64 (256 bit key, SHA256 hash)

Samba was using the following configuration:

```
[HDD INTERNAL]
comment = NAS
path = /media/internal
valid users = odroid
writable = yes
create mask = 0775
directory mask = 0775
# Tweaks
write cache size = 524288
getwd cache = yes
use sendfile = yes
min receivefile size = 16384
socket options = TCP_NODELAY IPTOS_LOWDELAY
```


96% complete

Copying 1 item from Desktop to hdd internal (\\192.168.100.48) (Z:)

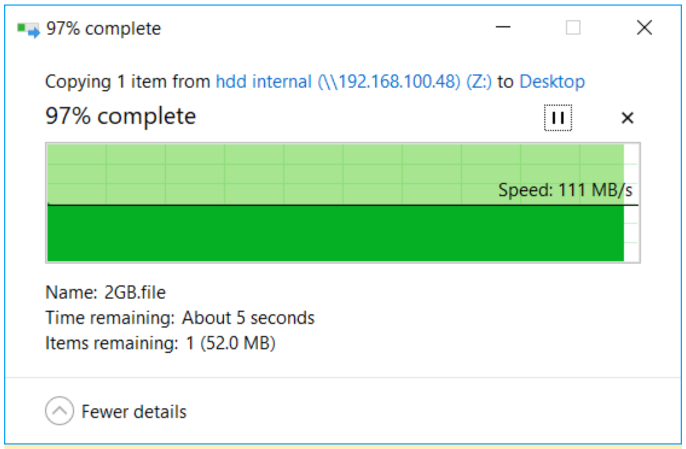
96% complete

Speed: 92.7 MB/s

Name: 2GB.file
Time remaining: About 5 seconds
Items remaining: 1 (68.0 MB)

Fewer details

Figure 2 – Before encryption test 1


97% complete

Copying 1 item from hdd internal (\\192.168.100.48) (Z:) to Desktop

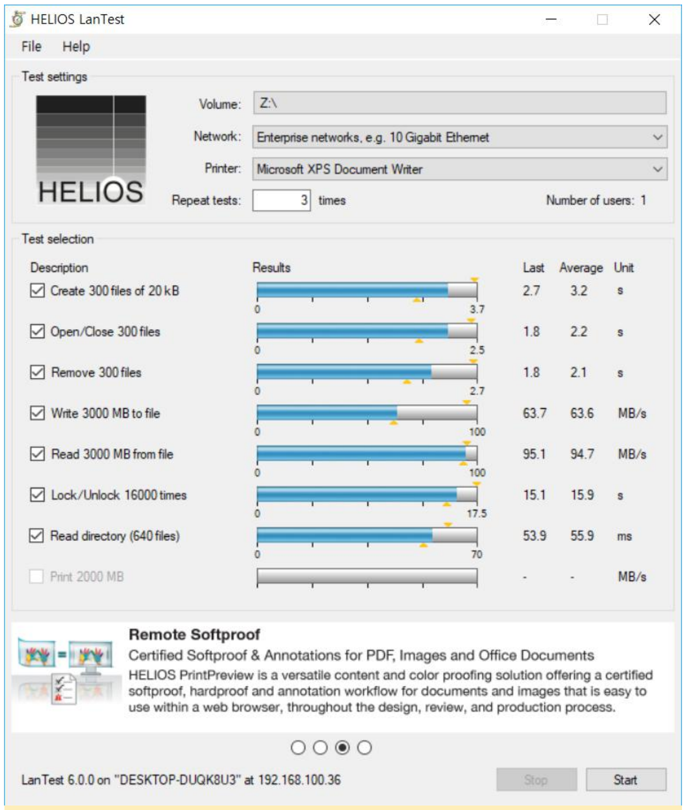
97% complete

Speed: 111 MB/s

Name: 2GB.file
Time remaining: About 5 seconds
Items remaining: 1 (52.0 MB)

Fewer details

Figure 3 – Before encryption test 2


HELIOS LanTest

File Help

Test settings

Volume: Z:\
Network: Enterprise networks, e.g. 10 Gigabit Ethernet
Printer: Microsoft XPS Document Writer
Repeat tests: 3 times
Number of users: 1

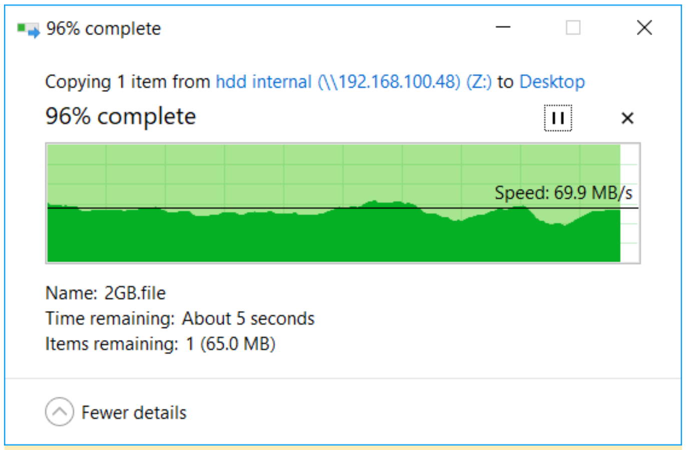
Test selection

Description	Results	Last	Average	Unit
☑ Create 300 files of 20 kB	0 – 3.7	2.7	3.2	s
☑ Open/Close 300 files	0 – 2.5	1.8	2.2	s
☑ Remove 300 files	0 – 2.7	1.8	2.1	s
☑ Write 3000 MB to file	0 – 100	63.7	63.6	MB/s
☑ Read 3000 MB from file	0 – 100	95.1	94.7	MB/s
☑ Lock/Unlock 16000 times	0 – 17.5	15.1	15.9	s
☑ Read directory (640 files)	0 – 70	53.9	55.9	ms
☐ Print 2000 MB		-	-	MB/s

Remote Softproof
Certified Softproof & Annotations for PDF, Images and Office Documents
HELIOS PrintPreview is a versatile content and color proofing solution offering a certified softproof, hardproof and annotation workflow for documents and images that is easy to use within a web browser, throughout the design, review, and production process.

LanTest 6.0.0 on "DESKTOP-DUQK8U3" at 192.168.100.36

Figure 4 – HELIOS LanTest Before Encryption


96% complete

Copying 1 item from hdd internal (\\192.168.100.48) (Z:) to Desktop

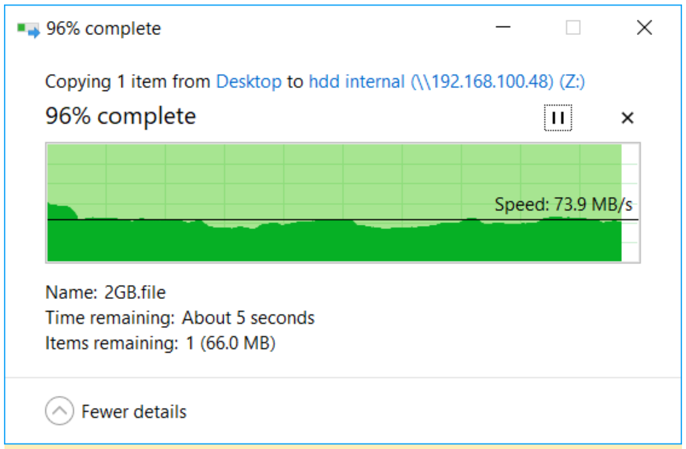
96% complete

Speed: 69.9 MB/s

Name: 2GB.file
Time remaining: About 5 seconds
Items remaining: 1 (65.0 MB)

Fewer details

Figure 5 – After encryption test 1


96% complete

Copying 1 item from Desktop to hdd internal (\\192.168.100.48) (Z:)

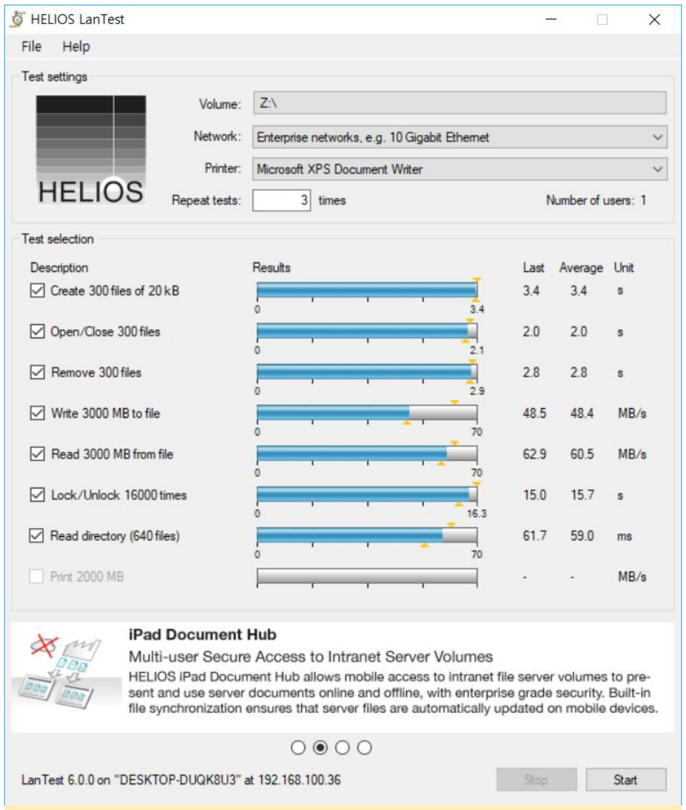
96% complete

Speed: 73.9 MB/s

Name: 2GB.file
Time remaining: About 5 seconds
Items remaining: 1 (66.0 MB)

Fewer details

Figure 6 – After encryption test 2



Test settings

Volume: Z:\
Network: Enterprise networks, e.g. 10 Gigabit Ethernet
Printer: Microsoft XPS Document Writer
Repeat tests: 3 times
Number of users: 1

Test selection

Description	Results	Last	Average	Unit
☑ Create 300 files of 20 kB	0 – 3.4	3.4	3.4	s
☑ Open/Close 300 files	0 – 2.1	2.0	2.0	s
☑ Remove 300 files	0 – 2.9	2.8	2.8	s
☑ Write 3000 MB to file	0 – 70	48.5	48.4	MB/s
☑ Read 3000 MB from file	0 – 70	62.9	60.5	MB/s
☑ Lock/Unlock 16000 times	0 – 16.3	15.0	15.7	s
☑ Read directory (640 files)	0 – 70	61.7	59.0	ms
☐ Print 2000 MB		-	-	MB/s

iPad Document Hub
Multi-user Secure Access to Intranet Server Volumes
HELIOS iPad Document Hub allows mobile access to intranet file server volumes to present and use server documents online and offline, with enterprise grade security. Built-in file synchronization ensures that server files are automatically updated on mobile devices.

LanTest 6.0.0 on "DESKTOP-DUQK8U3" at 192.168.100.36

Figure 7 – HELIOS LanTest After Encryption

For the original article, please see the Wiki post at https://wiki.odroid.com/odroid-xu4/software/disk_encryption.

Tvheadend

April 1, 2018 By @Pepes ODROID-C2, Tutorial

Tvheadend (TVH) is server software that can read video streams from LinuxTV sources and publish them as streams viewable on devices like Smart TVs over an IP internet. It can be used as a recorder too. Input sources can include: DVB-S, DVB-C/T, ATSC, IPTV and SAT>IP, to name a few. Multiple TVH servers can be combined to form a network. The following instructions show how to compile the TVH code for the ODROID-C2.

Installation

Install the necessary software components using the following commands:

```
$ sudo apt-get install cmake git libssl-dev
libdvbcsa-dev ffmpeg liburiparser-dev openssl
libavahi-client-dev zlib1g-dev libavcodec-dev
libavutil-dev libavformat-dev libswscale-dev
libavresample-dev dvb-apps libiconv-hook-dev
```

Now, we need to download source code:

```
$ wget
https://github.com/tvheadend/tvheadend/archive
/master.zip
```

Move it to a working directory and then extract it:

```
$ cd ~
$ unzip master.zip -d tvheadend
```

Enter the relevant directory and build:

```
$ cd tvheadend/tvheadend-master
$ ./configure
$ make
```

You should observe build errors at this point.

To rectify this, download the following files to ~/tvheadend/tvheadend-master from the git repository:

- config.guess (https://goo.gl/3wJNcP)
- config.sub (https://goo.gl/NKzVes)

Replace the original files with these versions:

```
$ cp config.guess build.linux/ffmpeg/libtheora
$ cp config.sub build.linux/ffmpeg/libtheora
```

Now we can repeat the build steps:

```
$ make
$ sudo make install
```

For the first time that you run Tvheadend, you will need to run it with the option -C:

```
$ tvheadend -C
```

You will then need to set up the configuration through the UI (IP:9981) and terminate it by pressing CTRL+C. Then you can run it with other options like:

```
$ screen -m -s tvheadend
```

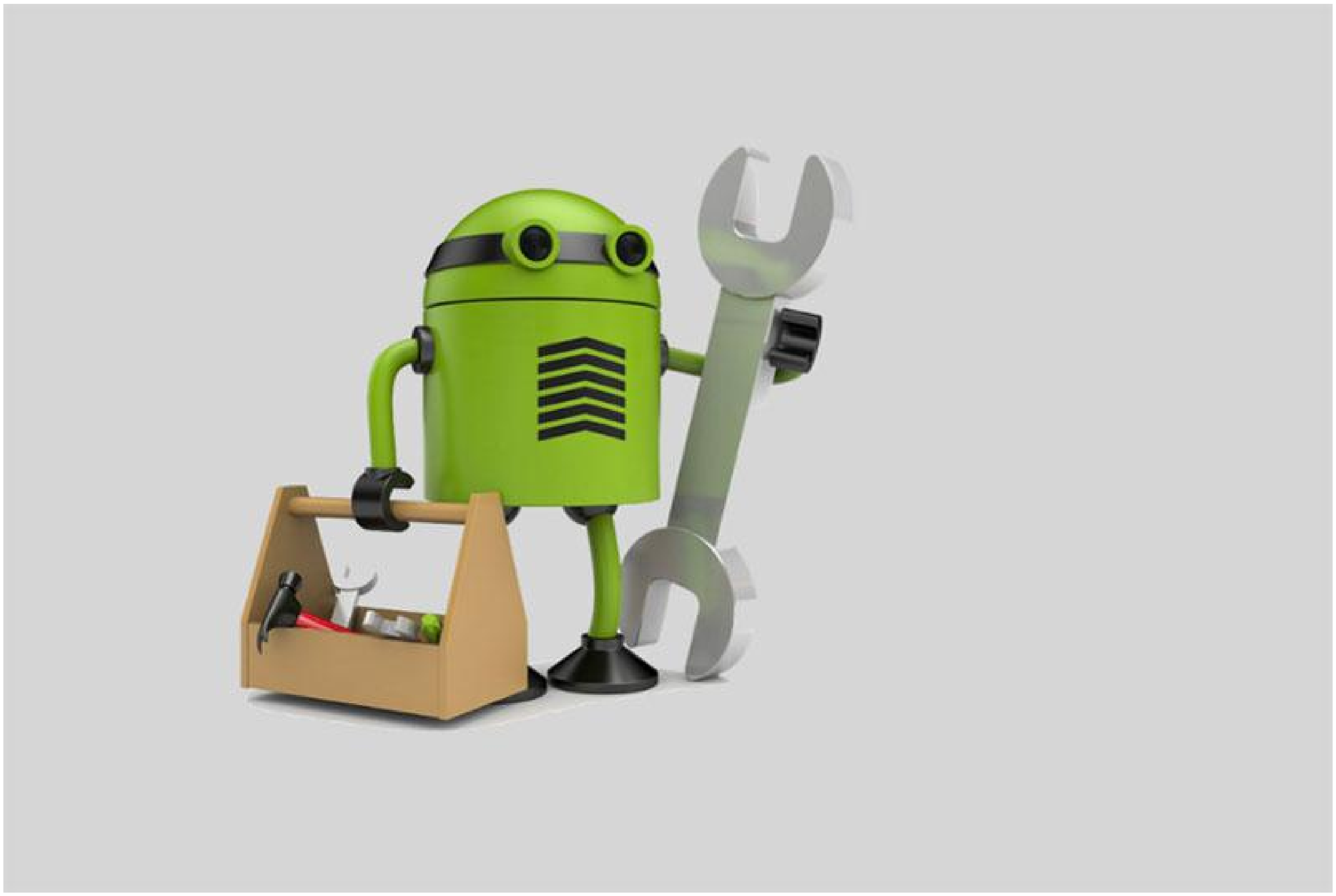
or:

```
$ tvheadend
```

For the latter, you need to have PuTTY connected all the time, otherwise it will terminate.

Android Development: So, You Want to Be an App Developer?

April 1, 2018 By Randy Hall Android, Tutorial



So, you want to be an app developer? Let's see what I can do to help you with that! A casual survey of the back issues of ODROID Magazine has shown that there are plenty of in-depth, detailed articles about Android on ODROID. What the community has been missing is a series of "So, you want to get started with this" articles. Sure, there are plenty of YouTube videos with questionable audio and video quality, as well as blogs and tutorials online that you may or may not be able to find through a search, but rather than muddle through all that noise, I thought it would be helpful to throw my proverbial hat into the "Getting Started" ring.

Aside from being capable single-board Linux computers, ODROID models such as the C1+, C2, and even the XU4 are well-suited for Android app development, whether for handheld, tablet, or automotive use. I own a well-loved ODROID-C1 that I purchased a couple years ago and while it may be a bit older than the latest generation of available hardware, the brain inside is the same as the current ODROID-C0 and C1+ and is still a valid, affordable test platform for trying out Android apps.

Walk before you run

Before we dive into using ODROID boards for app development, we need to cover a few basics. First, you will need to have a way to create the software that you will run on the Android OS. In order to do that, you'll need a development environment, which is a collection of tools that work together to create the application. Android has a few options for developing programs in an Integrated Development Environment (IDE). We will focus on the "official" one, Android Studio.

Second, you will need to have a safe place to store your work. While personally I'm a fan of the self-hosting, I recognize that any good backup strategy involves storing a copy of your data offsite. So make sure to sign up for an online git-based source code repository on a hosted git provider such as Github, Gitlab, or Bitbucket.

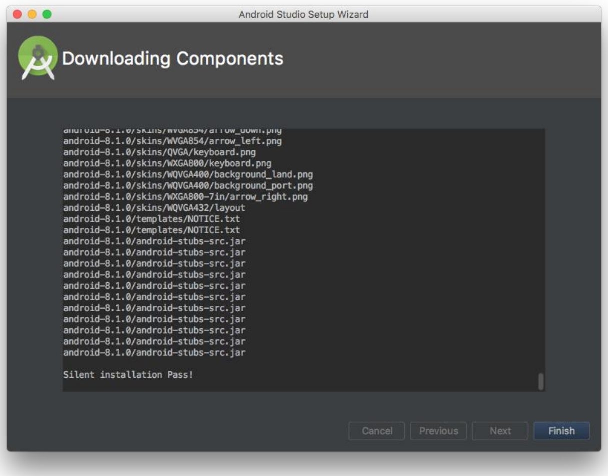
Finally, let's take a moment to outline what a sane software development process should look like, both in general and specifically for Android:

- Setting up the development environment
- Set up Android on ODROID
- Create the Hello World app
- Deploy app to Android/ODROID
- Iterate!

Setting up the development environment

When you set up a development environment, you will most likely do it on one of three operating systems: Linux, Windows or macOS. For this column, I will install Android Studio on my Mac but you will find the process substantially similar, regardless of platform. Begin by visiting the Android Studio download page at https://developer.android.com/studio. The page will generally select the correct download for your operating system (OS), but you can always download any of the installers or bundles for whatever OS you wish.

As of this writing, the Android Studio installation instructions are very straightforward and include specific steps for each supported OS, which are available at https://developer.android.com/studio/install.html. The installation on my Mac was detailed and remarkably painless. The installer downloads a variety of Android libraries needed for development and provides the Intel HAXM hypervisor to assist in Android emulation down the road. In just a few minutes, the job is done, as shown in Figure 1.



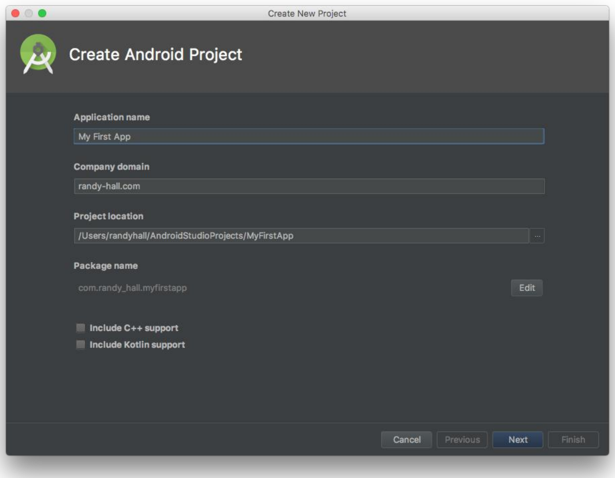
```
android-8.1.0/skins/WVGA854/arrow_left.png
android-8.1.0/skins/QVGA/keyboard.png
android-8.1.0/skins/WXGA800/keyboard.png
android-8.1.0/skins/WQVGA400/background_land.png
android-8.1.0/skins/WQVGA400/background_port.png
android-8.1.0/skins/WXGA800-7in/arrow_right.png
android-8.1.0/skins/WQVGA432/layout
android-8.1.0/templates/NOTICE.txt
android-8.1.0/templates/NOTICE.txt
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar
android-8.1.0/android-stubs-src.jar

Silent installation Pass!
```

Figure 1 – Downloading components

Once you click "Finish", the app will start and you'll be confronted by a list of choices. For now, select "Start a new Android Studio project" and have a look around. Of course, you'll be immediately plunged into the world of Java programming on the "Create Android Project" screen as you'll be asked to come up with a "company domain.", which I'll explain in the next section.

What domain?

In Java, all object classes are divided into namespaces, which are basically groupings where you can safely create Java classes, which can be named in whatever way you want, that won't clobber someone else's Java class definitions. It's similar in concept to domain names on the web, which is why Android Studio refers to it as a "company domain", as illustrated in Figure 2. For now, you can put any domain you want, but definitely don't use a domain that you don't own or control.


Create New Project

Create Android Project

Application name
My First App

Company domain
randy-hall.com

Project location
/Users/randyhall/AndroidStudioProjects/MyFirstApp

Package name
com.randy_hall.myfirstapp

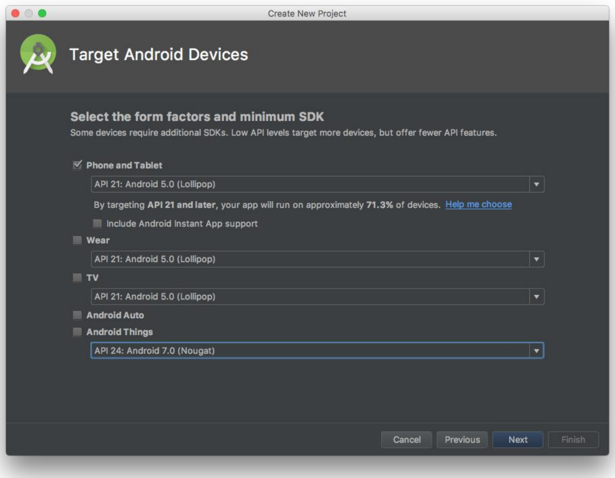
- [ ] Include C++ support
- [ ] Include Kotlin support

Figure 2 – Create Android project

You can see the resulting namespace for this new project by looking at the “Package Name” field on that screen, which in my case shows “com.randy_hall.myfirstapp”. For now, understand that this is how Java and, by extension, Android, works.

What Device?

Next, we’ll look at what level of Android API we want to use for this project. This is both an incredible advantage of Android, and one of its most frustrating shortcomings. There is such a wide variety of software versions of Android out in the world that when you choose to support one API level over another, you inherently exclude a number of Android devices from using your app. For now, let’s be reasonable and choose “API 21: Android 5.0 (Lollipop)” as shown in Figure 3, which, according to Android Studio, will mean that our app will run on about 71.3% of Android devices as of this writing. As new Android devices are introduced and old phones get upgraded or decommissioned, this number will generally increase over time.


Create New Project

Target Android Devices

Select the form factors and minimum SDK
Some devices require additional SDKs. Low API levels target more devices, but offer fewer API features.

- [x] Phone and Tablet
  API 21: Android 5.0 (Lollipop)
  By targeting **API 21 and later**, your app will run on approximately **71.3%** of devices. Help me choose
  - [ ] Include Android Instant App support
- [ ] Wear
  API 21: Android 5.0 (Lollipop)
- [ ] TV
  API 21: Android 5.0 (Lollipop)
- [ ] Android Auto
- [ ] Android Things
  API 24: Android 7.0 (Nougat)

Figure 3 – Target Android devices

Adding activity

Android Studio can help get you off the ground a bit more quickly by providing some scaffolding, which are templates of common application patterns that provide a starting point to begin your work. There are several to choose from, which we will talk more about in the future, but for now let’s pick “Empty Activity”, as shown in Figure 4.

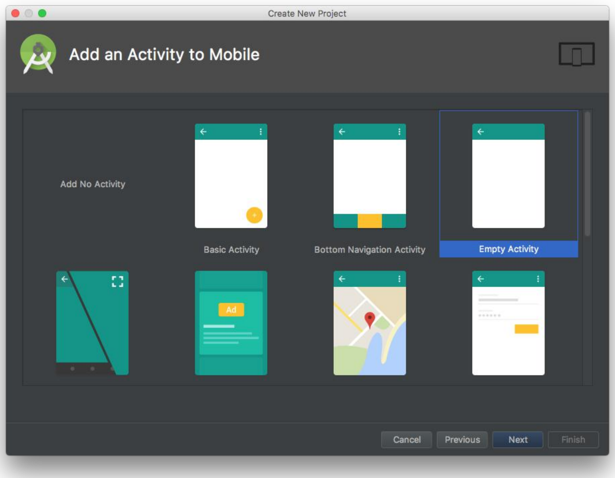
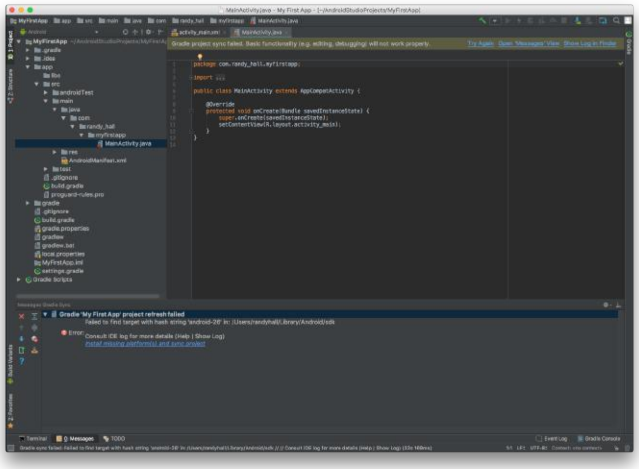


Figure 4 – Add an activity to mobile

You will be asked to configure the empty activity by giving it a name and a layout name. Both of these will be used to generate code for the project, so you can name it however you like. I will stick with the defaults of “MainActivity” and “activity_main”, but you can always change them later in the code.

Once the MainActivity has been generated, things start moving forward! We have clicked a lot, and Android Studio has done some background work, but now we are in the IDE, ready to get started. You may notice right away that the amount of code generated is relatively small. That's in large part because we chose an Empty Activity, which is remarkably empty! If we had chosen another activity to generate, we would have been faced with considerably larger amount of code to look through.

Along the way, I got a "Gradle Sync" error, as shown in Figure 5, which means that I don't have the most recent Android API installed. I clicked the "Install missing platform(s)" and hit refresh, after which the error repeated itself for the build tools. Installing both of these pieces (if they weren't already there) got me past that hurdle and on to the workspace, ready for coding.



Gradle project sync failed. Basic functionality (e.g. editing, debugging) will not work properly.

```
package com.randy_hall.myfirstapp;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```
Gradle 'My First App' project refresh failed
        Failed to find target with hash string 'android-26' in: /Users/randyhall/Library/Android/sdk
Error: Consult IDE log for more details (Help | Show Log)
        Install missing platform(s) and sync project
```

Figure 5 – Gradle Sync error

Git with it

Now we are ready to move to the next step, which is creating a hosted git account to store and perhaps share our work. For this, we will use one of the slightly less well-known hosted git providers, Github (https://www.github.com).

Hosted git providers like Github, Gitlab, and Bitbucket are all based on the widely used open-source git version control software. We can get into the philosophy behind git in more detail in another article, but for now it's enough to know that the method we're using for backing up our projects is very well-supported in the app development world.

Creating a new account at Github is easy. You need to choose an original username, provide a valid email account to activate the Github account, and select a password. Once you've done that, Github will pitch you on subscribing for additional services, but you can choose the default free plan, which means only that your repositories will be publicly visible. If that's a problem for you, you can always opt for Gitlab, which provides a free private git repository option, or pay a subscription to Github for the privilege of keeping private repos.

Ready to run?

Once you're signed up, you're just about ready to connect Android Studio to your Github account. We will pick that up in the next installment, and then proceed to create the most typical of all applications: the "Hello World" app. The world of Android app development is now open!

Despite what you might think, I don't write these articles for my own well-being. I would love to hear about what kinds of app development you're interested in doing. Your feedback will inform and influence how these articles evolve over time! If you are interested in participating, you can comment on the interactive version of ODROID Magazine for this article, or visit the ODROID Magazine forum at https://forum.odroid.com.

Setting Up Your ODROID: ODROID-XU4 As A General Purpose NAS

April 1, 2018 By Adrian Popa Linux, ODROID-XU4, Tutorial

I purchased my ODROID-XU4 with the intent of converting it into a NAS. However, I did not want to settle on a specialized NAS distro like OpenMediaVault because I wanted my ODROID-XU4 to do much more than being a plain old NAS. For instance, I plan on transcoding TV shows recorded from my TV to the H264 standard, using the ODROID-XU4's hardware encoder (as described in http://bit.ly/2jnv4Za), and also make use of the GPIO pins later on. One more issue I had with OpenMediaVault is that it runs on top of Debian, and I wanted to keep using Ubuntu in order to benefit from newer packages.

I would be losing much of the convenience of using a specialized distro and consequently have to discover alternate ways of doing things in a simple and user-friendly way. This presents an opportunity to gain new knowledge.

These are the steps we will need to take:

- Install Webmin (http://bit.ly/J5WtfI) for easier management
- Mount the disks
- Set up network shares (Samba/NFS)
- Install Owncloud
- Secure and optimize the OS

These instructions presume that you have medium or higher level system expertise.

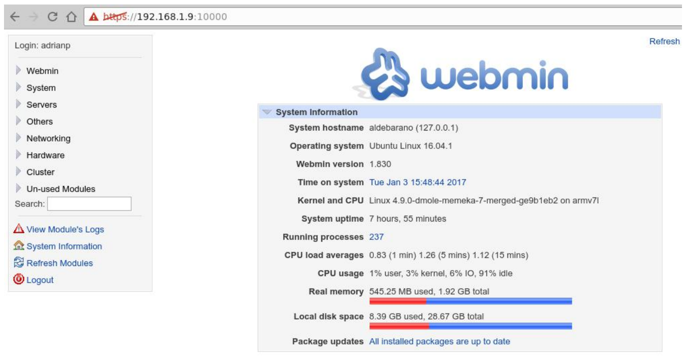
Webmin

Every NAS needs a nice web GUI. Unfortunately, OpenMediaVault's GUI is not an option, and after searching for a long while for alternatives I settled on using Webmin. Webmin has been around since 1997 and has solid support for general server maintenance tasks. It has the advantage that even inexperienced users can find their way around and with the integrated help in order to set set up and manage all kinds of servers like Apache, MySQL, Mail, DNS, and more. It has solid support for RAID and LVM management, and also supports Samba and NFS file sharing. Unfortunately, it lacks support for newer services like Transmission or Owncloud, but I can always configure them manually.

To install it, follow the steps below:

```
$ echo "deb
http://download.webmin.com/download/repository
sarge contrib" | sudo tee
/etc/apt/sources.list.d/webmin.list
$ wget http://www.webmin.com/jcameron-key.asc
$ sudo apt-key add jcameron-key.asc
$ rm jcameron-key.asc
$ sudo apt-get update
$ sudo apt-get install libapt-pkg-perl libnet-
ssleay-perl libauthen-pam-perl libio-pty-perl
apt-show-versions apt-transport-https
$ sudo apt-get install webmin
$ sudo systemctl enable webmin
$ sudo systemctl start webmin
```

You can login to your device's IP address on port 10000 to use the web interface: https://odroid-ip:10000. However after you log-in (with any system user with sudo access) you will likely be unimpressed by the default interface. It looks like it is out of the 1990's.



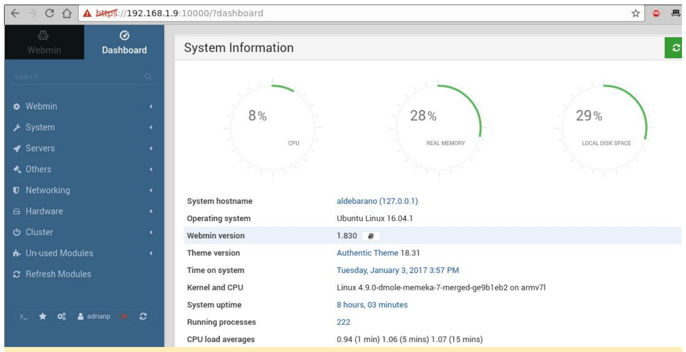
System hostname	aldebarano (127.0.0.1)
Operating system	Ubuntu Linux 16.04.1
Webmin version	1.830
Time on system	Tue Jan 3 15:48:44 2017
Kernel and CPU	Linux 4.9.0-dmole-memeka-7-merged-ge9b1eb2 on armv7l
System uptime	7 hours, 55 minutes
Running processes	237
CPU load averages	0.83 (1 min) 1.26 (5 mins) 1.12 (15 mins)
CPU usage	1% user, 3% kernel, 6% IO, 91% idle
Real memory	545.25 MB used, 1.92 GB total
Local disk space	8.39 GB used, 28.67 GB total
Package updates	All installed packages are up to date

Figure 1 – Stock Webmin interface

The first thing we must do is beautify it via a theme. The best-looking theme is called "Authentic Theme", which brings in a lot of features, including being mobile-friendly. You can get the latest version from http://bit.ly/2jf468e and install it using the following command:

```
$ wget https://github.com/qooob/authentic-
theme/releases/download/19.12/authentic-theme-
19.12.wbt.gz
```

Navigate inside Webmin to "Webmin Configuration -> Webmin Themes -> Install themes -> From uploaded file" and select the newly downloaded theme. After a short wait and refresh later, you will be presented with the following page:



System hostname	aldebarano (127.0.0.1)
Operating system	Ubuntu Linux 16.04.1
Webmin version	1.830
Theme version	Authentic Theme 18.31
Time on system	Tuesday, January 3, 2017 3:57 PM
Kernel and CPU	Linux 4.9.0-dmole-memeka-7-merged-ge9b1eb2 on armv7l
System uptime	8 hours, 03 minutes
Running processes	222
CPU load averages	0.94 (1 min) 1.06 (5 mins) 1.07 (15 mins)

Figure2 – Webmin with Authentic theme

You can explore Webmin's features by using the search tool in the interface. Note that you can install third-party modules available from http://bit.ly/2jf6KLd.

Mounting disks

First, you will need to decide if you are going to use RAID or LVM with your disks, and which filesystem you would use. I will not go into details about setting RAID/LVM because the subject has been discussed in previous articles. However, even without having a lot of expertise, you can use Webmin to do the heavy lifting for you and use the built-in help to learn more. Webmin will prompt you to install any missing dependencies. Once you have your partitions ready, you can start mounting them.

The traditional method of mounting is to use /etc/fstab (and Webmin has a comprehensive module to handle that as well), but you may run into problems if you start your system with the disk not attached (systemd likes to wait around for the disk). I prefer to use autofs, which mounts disks (local or network-based) on demand and unmounts them when not in use. Unfortunately, it's not managed by webmin, so you will need to use the shell:

```
$ sudo apt-get install autofs
```

You will need to edit /etc/auto.master and add a mount entry for your disk, specifying the base directory and its configuration file:

```
$ sudo vi /etc/auto.master
# add at the end your mountpoint
/media/yourdisk /etc/auto.yourdisk --timeout
20
```

In the command above, replace your disk with the path you want to use. Next, edit this configuration file and add your partitions and their mount parameters, using the command “blkid” to find the correct UUID for the disk:

```
$ sudo blkid
$ sudo vi /etc/auto.yourdisk
xfs-partition   -
fstype=xfs,dev,exec,suid,noatime
:UUID=9d2d675d-cb08-45b2-b222-c981a8d00c06
```

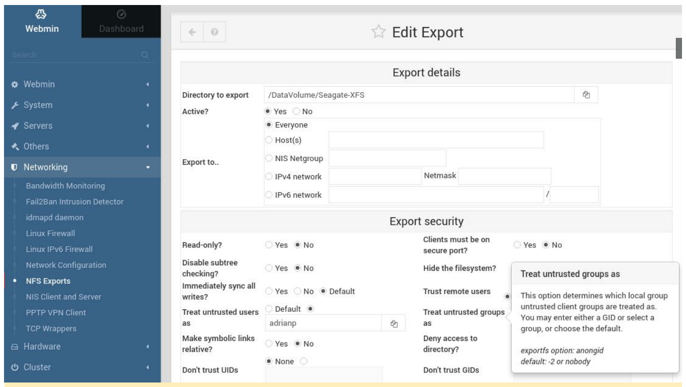
Restart autofs, and when you access /media/yourdisk/xfs-partition your partition will be mounted automatically:

```
$ sudo service autofs restart
```

You will need to take care of the mount parameters because each filesystem has their own parameters and they might impact performance. For instance, without activating the parameter big_writes on NTFS, you will get very poor performance. If in doubt, you can cheat and use Webmin to create entries in /etc/fstab, test them to ensure the parameters are ok, and migrate them to autofs’s layout later (that’s what I used). To force automounted disks to be unmounted, you can simply restart the autofs service.

Set up file shares

To set up Samba shares (and also install Transmission for torrent downloading) you can follow the guide “Designing your own seedbox” featured in Odroid Magazine http://bit.ly/2j3xpaK. You also can also experiment with Webmin’s interface and easily create shares and users with a few clicks. For example, Figure 3 shows the “Create NFS share” dialog. Clicking on the form items shows a contextual help menu that explains well, what that item does. This can help you with things you might not be familiar with.



Edit Export

Export details

Directory to export: /DataVolume/Seagate-XFS
Active? [✓ Yes] No
Export to.. [✓ Everyone] Host(s) NIS Netgroup IPv4 network Netmask IPv6 network /

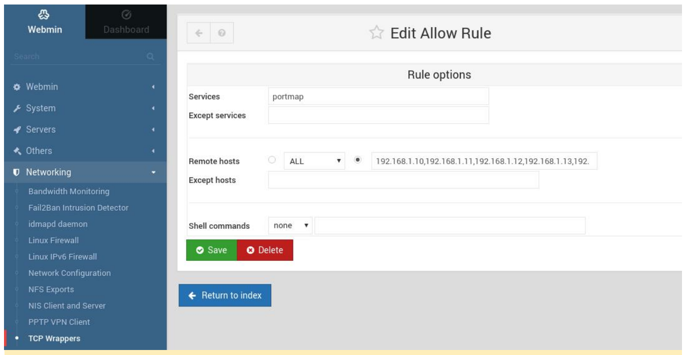
Export security

Read-only? Yes [✓ No]
Clients must be on secure port? Yes [✓ No]
Disable subtree checking? Yes [✓ No]
Hide the filesystem?
Immediately sync all writes? Yes No [✓ Default]
Trust remote users
Treat untrusted users as: Default [✓] adrianp
Treat untrusted groups as
Make symbolic links relative? Yes [✓ No]
Deny access to directory?
Don't trust UIDs: [✓ None]
Don't trust GIDs

Treat untrusted groups as
This option determines which local group untrusted client groups are treated as. You may enter either a GID or select a group, or choose the default.
exportfs option: anongid
default: -2 or nobody

Figure3 – Create NFS share

When creating Samba/NFS shares, take security into consideration from the start. Samba authenticates by userid and password, but NFS authenticates users only by IP. If you know which hosts in your network may have access to specific shares, specify it in the configuration. For example, an NFS share might be exported to “Everyone”, but access can still be limited with iptables or /etc/hosts.allow and /etc/hosts.deny (which are used by TCP Wrappers Webmin module).



Edit Allow Rule

Rule options

Services: portmap
Except services
Remote hosts: ALL [✓ 192.168.1.10,192.168.1.11,192.168.1.12,192.168.1.13,192.]
Except hosts
Shell commands: none

Figure 4 – . /etc/hosts.allow configuration for NFS to limit access from a few hosts

Used with its default configuration, Samba will give decent performance, but with the tweaks below, extracted from the ODROID forums, you should get fewer “pauses” in large file transfers. Add the lines below to the [global] section of your /etc/samba/smb.conf:

```
write cache size = 524288
getwd cache = yes
use sendfile = yes
min receivefile size = 16384
```

Install Owncloud

Owncloud is a personal “cloud” service that lets you share files with people over the Internet. I am not going to go into installation details, because they have been discussed in a previous Magazine article http://bit.ly/2kgVZpn, but there are some things I would like to point out.

First of all, the installation is quite simple on Ubuntu 16.04. I used the guide at http://do.co/2bzxhxG and was up and running in 10 minutes. If you have a DNS name (e.g. dynamic DNS for your home) you should take the time to get a valid SSL certificate from Let's Encrypt (http://bit.ly/1qmIXlY) using steps listed at http://do.co/2bQpv4M.

You basically need to install the following prerequisites before installing OwnCloud:

```
$ sudo apt-get install php
  libapache2-mod-php php-mcrypt
  php-mysql php-bz2 php-curl
  php-gd php-imagick php-intl
  php-mbstring php-xml php-zip
  mysql-server apache2
```

Next you install the OwnCloud repository for Ubuntu and refresh the available packages:

```
$ sudo curl
https://download.owncloud.org/download/
repositories/stable/Ubuntu_16.04/Release.key
| sudo apt-key add -
$ echo 'deb
https://download.owncloud.org/download/
repositories/stable/Ubuntu_16.04/ /'
| sudo tee
/etc/apt/sources.list.d/owncloud.list
$ sudo apt-get update
```

Finally, you can install OwnCloud:

```
$ sudo apt-get install owncloud
$ sudo systemctl reload apache2
```

You will also need to create a database user for OwnCloud:

```
$ sudo mysql -u root
> CREATE DATABASE owncloud;
> GRANT ALL ON owncloud.* to
'owncloud'@'localhost' IDENTIFIED BY
'databasePassword';
> FLUSH PRIVILEGES;
> exit
```

After all this work, you can login through the web interface at https:///owncloud and finish the installation. Since the point of OwnCloud is to be accessible to the Internet, you should take some time to harden your installation, as described at http://bit.ly/2jOTe1F. In my case, I want to run the OwnCloud service on a different port (so that external users don't have access to my internal sites), to set iptables rules to allow access only from my country (based on geo-ip data), and set up fail2ban to protect me against automated password guesses.

In order to run the OwnCloud virtual host on a different port you need to make a few adjustments to your apache config:

```
$ sudo cp /etc/apache2/sites-
available/default-ssl.conf
  /etc/apache2/sites-available/owncloud.conf
$ cd /etc/apache2/sites-available
$ sudo ln -s ../sites-available/owncloud.conf
  020-owncloud.conf
```

Next, edit /etc/apache2/sites-available/owncloud.conf and make the following changes:

Add "Listen 8443" as the first row Change the VirtualHost definition to use port 8443 instead of 443 () Change DocumentRoot to point to your owncloud installation "DocumentRoot /var/www/owncloud"

When done, you can restart the Apache daemon, and you should be able to access only your OwnCloud instance on https://:8443/.

To get started with GeoIP firewall rules, you'll need to have the kernel sources (or kernel headers) available. Next, you can install the extra iptables modules with the following command:

```
$ sudo apt-get install
  xtables-addons-dkms
  xtables-addons-common
  xtables-addons-source
```

The dkms package may fail to install cleanly because some of the modules fail to compile against kernel 4.9/4.14. You can disable the failed modules and recompile the rest by setting the following settings to "n" instead of "m" in the file /var/lib/dkms/xtables-addons/2.10/build/mconfig:

```
$ sudo vi /var/lib/dkms/xtables-
addons/2.10/build/mconfig
build_ACCOUNT=n
build_LOGMARK=n
build_SYSRQ=n
build_pknock=n
build_psd=n
```

Next you will need to manually compile the rest:

```
$ cd /var/lib/dkms/xtables-addons/2.10/build/
$ sudo autoconf
$ sudo ./configure
$ sudo make
$ sudo make install
```

Before using the geoip module, you will need to initialize the geoip database (the prefix to country mapping). You may need to repeat this step from time to time to benefit from the latest data:

```
$ sudo apt-get install libtext-csv-xs-perl
$ sudo mkdir /usr/share/xt_geoip
$ sudo /usr/lib/xtables-addons/xt_geoip_dl
$ sudo /usr/lib/xtables-addons/xt_geoip_build
-D /usr/share/xt_geoip
/root/GeoIPCountryWhois.csv
```

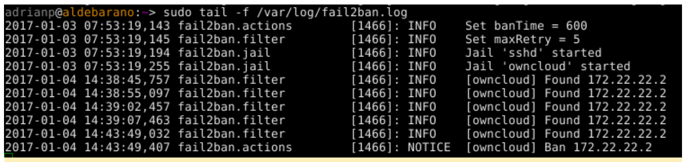
All that is left to do now is to create and test the iptables rules to allow only traffic that you want to reach your owncloud setup. An example rule looks like this:

```
$ sudo iptables -N geo-owncloud
$ sudo iptables -A INPUT -p tcp -m tcp --dport
8443 -j geo-owncloud
$ sudo iptables -A geo-owncloud -s
192.168.1.0/24 -j ACCEPT
$ sudo iptables -A geo-owncloud -m geoip ! --
src-cc RO -j DROP
```

Do not forget to save your rules and apply them at startup (either with iptables-save or with webmin). More details about geoip can be found at http://bit.ly/2jnwUJD.

Configuring fail2ban is not very complicated once you follow the tutorial at http://bit.ly/2kipXxn. Remember to install fail2ban first (and test it with some false credentials):

```
$ sudo apt-get install fail2ban
```



```
adrianp@aldebarano:~> sudo tail -f /var/log/fail2ban.log
2017-01-03 07:53:19,143 fail2ban.actions        [1466]: INFO    Set banTime = 600
2017-01-03 07:53:19,145 fail2ban.filter         [1466]: INFO    Set maxRetry = 5
2017-01-03 07:53:19,194 fail2ban.jail           [1466]: INFO    Jail 'sshd' started
2017-01-03 07:53:19,255 fail2ban.jail           [1466]: INFO    Jail 'owncloud' started
2017-01-04 14:38:45,757 fail2ban.filter         [1466]: INFO    [owncloud] Found 172.22.22.2
2017-01-04 14:38:55,097 fail2ban.filter         [1466]: INFO    [owncloud] Found 172.22.22.2
2017-01-04 14:39:02,457 fail2ban.filter         [1466]: INFO    [owncloud] Found 172.22.22.2
2017-01-04 14:39:07,463 fail2ban.filter         [1466]: INFO    [owncloud] Found 172.22.22.2
2017-01-04 14:43:49,032 fail2ban.filter         [1466]: INFO    [owncloud] Found 172.22.22.2
2017-01-04 14:43:49,407 fail2ban.actions        [1466]: NOTICE  [owncloud] Ban 172.22.22.2
```

Figure 5 – Fail2Ban doing its job on failed logins

Since we have added a special port for owncloud, we will need to tweak fail2ban's configuration to account for that. Edit /etc/fail2ban/jail.local and append "port 8443" to the port line and restart fail2ban:

```
$ sudo vi /etc/fail2ban/jail.local
port = http,https,8443
$ sudo service fail2ban restart
```

To manually lift the ban for a blacklisted IP address you can run the following command:

```
$ sudo fail2ban-client set owncloud unbanip
172.22.22.2
```

Assign tasks to specific CPUs

The ODROID-XU4 comes with two types of CPU cores: 4 little cores that are low power and are best suited for background tasks and 4 big cores which are designated for more powerful tasks. The official kernel comes with a "magic" scheduler from Samsung which knows the processor's true power, and can switch tasks from the little cores to the big cores when load is high. There may be special cases where you want to run specific tasks exclusively on the big or little cores, either to maximize performance, or to minimize temperature. We can use use cgroups as noted in http://bit.ly/2jP6KlU.

"cgroups" is a feature of modern kernels that allows allocation of resources for various processes. In our case we will need the "cpuset" cgroup to create a "littlecores" and a "bigcores" group. Each group will force processes to run on specific cores by setting the affinity. So, littlecores will have cpus 0-3 and bigcores 4-7. Fortunately, creating the cgroups is easy:

```
# mkdir -p /sys/fs/cgroup/cpuset/littlecores
  /sys/fs/cgroup/cpuset/bigcores
# echo "0-3" > /sys/fs/cgroup/cpuset/
littlecores/cpuset.cpus
# echo "0"> /sys/fs/cgroup/cpuset/
littlecores/cpuset.mems
# chmod -R 777 /sys/fs/cgroup/cpuset/
```

```
littlecores
# echo "4-7"> /sys/fs/cgroup/cpuset/
bigcores/cpuset.cpus
# echo "0"> /sys/fs/cgroup/cpuset/
bigcores/cpuset.mems
# chmod -R 777 /sys/fs/cgroup/cpuset/
bigcores
```

Unfortunately, the commands will only last until the next reboot. So, let us create a service to set them as early as possible on boot:

```
$ sudo wget -O
/etc/systemd/system/cpuset.service
https://raw.githubusercontent.com/mad-ady/
odroid-ODROID-XU4-
optimizations/master/cpuset.service
$ sudo systemctl enable cpuset
$ sudo systemctl start cpuset
```

At this point, the cgroups are created, but they are not actively used by anyone. To manually start a process in a specific cgroup, you can use cgexec:

```
$ sudo apt-get install cgroup-tools
$ cgexec -g cpuset:bigcores sysbench --
test=cpu
--cpu-max-prime=100000 --num-threads=8 run
```

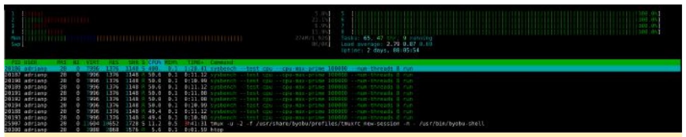


Figure 6 – 8 sysbench threads are forced to run on 4 specific cores

We are only halfway there. We will need to tell specific processes to run on the little cores and the others to run on the big cores. This is where you need to make a list and decide what you want. Start with a list of active services from webmin (System -> Bootup and Shutdown) and disable anything you will not be using. In my case I have disabled the following services: ModemManager, NetworkManager-wait-online, NetworkManager, accounts-daemon, alsa-restore, alsa-state, apport, apport-forward.socket, bluetooth, cups-browsed, cups.path, cups.service, cups.socket, lightdm, lxc-net, lxc, lxcfs, plymouth*, rsync, saned, speech-dispatcher and whoopsie.

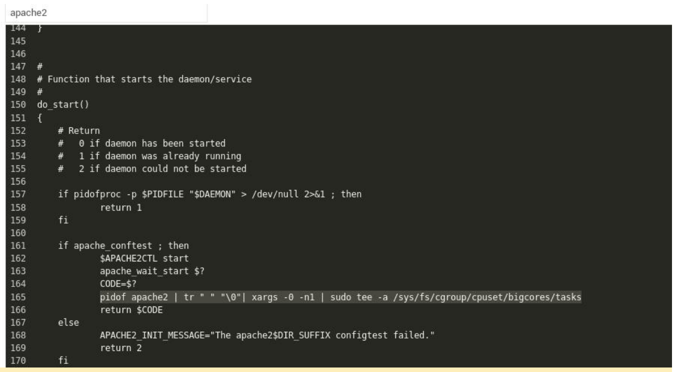
You will need to edit the startup scripts for the services you want and have them add their PID to the correct cgroup. Once the main process (and its children) are part of the correct cgroup, any new children will inherit the cgroup. My plan was to add things like MySQL, Apache, Samba, NFS and even Webmin to the big group and things like SSH (and all my shell activity), cron, Munin, and Transmission to the little group. This allows processes that are involved in the NAS functionality to be snappy, while other tasks can happily run on the little cores. If you are also using the X11 GUI, you might want to add lightdm to the "bigcores" group as well.

There are two types of startup scripts – systemd native scripts and legacy sys-v (/etc/init.d/). When editing a systemd script (for example nfs-mountd.service) you will need to add something like this to the [Service] section:

```
ExecStartPost=-/bin/sh -c 'echo $MAINPID | tee
-a /sys/fs/cgroup/cpuset/bigcores/tasks'
```

When editing an older sys-v script, it is trickier. You will need to find the start function, extract the PID(s) of the newly started process and add it to the tasks list. Below is an example for changing the apache startup script:

```
pidof apache2 | tr " " ""| xargs -0 -n1 | sudo
tee -a /sys/fs/cgroup/cpuset/bigcores/tasks
```



apache2

```
}


#
# Function that starts the daemon/service
#
do_start()
{
    # Return
    #   0 if daemon has been started
    #   1 if daemon was already running
    #   2 if daemon could not be started

    if pidofproc -p $PIDFILE "$DAEMON" > /dev/null 2>&1 ; then
            return 1
    fi

    if apache_conftest ; then
            $APACHE2CTL start
            apache_wait_start $?
            CODE=$?
            pidof apache2 | tr " " "\0"| xargs -0 -n1 | sudo tee -a /sys/fs/cgroup/cpuset/bigcores/tasks
            return $CODE
    else
            APACHE2_INIT_MESSAGE="The apache2$DIR_SUFFIX configtest failed."
            return 2
    fi
```

Figure 7 – Changing apache's startup configuration

Take care to restart each service after changing it and make sure to check that the process PID is in the correct cpuset tasks file. Do a full system reboot and check again after restart. If this sounds too complicated and mostly want to run tasks on the big cores, there is a way to cheat. You can simply set systemd's affinity, and all of its children processes will inherit it. The affinity can be controlled by the

CPUAffinity parameter in /etc/systemd/system.conf, but keep in mind you'll be wasting CPU cores.

Disk longevity

In order to prolong the life of your disk(s), you may want to spin them down after a period of inactivity. If you are using SSDs, you can skip this section because it only applies to old mechanical disks. Disks may receive a "stop" command to spin down either from an internal controller, from the USB-SATA bridge or directly from the operating system. However, sometimes the controllers are not tuned correctly and a stop command never arrives. This causes the disk to keep spinning which generates a lot of heat and can cause the drive to fail sooner than normal.

The normal way to handle this is to tell the disk to spin down after a period of inactivity, which can be done with hdparm:

```
$ sudo apt-get install sdparm hdparm
```

To manually set the disk to sleep after 10 minutes of inactivity, you can run the following command:

```
$ sudo hdparm -S 120 /dev/sda
```

If you get errors (like "bad/missing sense data"), hdparm might not help you for that disk.

To handle disk mobility, it would be better to let udev run the command after a disk has been plugged in. Since different disks might have different roles, and you may want different sleep timers (e.g. one disk is for backups and should sleep sooner, other is active and should sleep later), I decided on setting the UDEV rule based on the disk's serial number. You can get this serial number by looking in dmesg when plugging in a disk:

```
[1885221.800435] usb 4-1.3: Product: My
Passport 0730
[1885221.800436] usb 4-1.3: Manufacturer:
Western Digital
[1885221.800437] usb 4-1.3: SerialNumber:
575844314141305636323937
```

To set up the rule, create a file like this and reload udev:

```
$ sudo vi /etc/udev/rules.d/90-disk.rules
ACTION=="add", ENV{DEVNAME}=="/dev/sd?",
SUBSYSTEM=="block",
ENV{ID_SERIAL_SHORT}=="57584431414130563632393
7", RUN+="/sbin/hdparm -S 120 $env{DEVNAME}"
$ sudo udevadm control -R
```

If the hdparm cannot put your disk to sleep, then try other alternatives like sdparm, which can send a SCSI command to your disk, like ordering it to shut down in that instant:

```
$ sudo sdparm -C stop /dev/sda
```

There are tools like hd-idle (http://bit.ly/2j3zWSk) or periodic scripts you can run to put your disk to sleep. In my case they did not work, but make sure to try them manually before settling on a solution. Here is a manual script which checks a disk (identified by a partition's UUID) for activity in a 10s window, and if there was no disk activity (data transferred), it uses sdparm to stop the disk. You can run it via cron:

```
$ sudo wget -O /usr/local/bin/hdd-idle.sh
http://bit.ly/2k6LK7Y
$ sudo chmod a+x /usr/local/bin/hdd-idle.sh
$ sudo /usr/local/bin/hdd-idle.sh "4283-E975"
```

You must be aware that there are tools and services which will wake up your disk periodically, even if no data is transferred. Such tools include smartctl (from smartmontools) and smartd. The smartd service periodically checks disk health, and if not correctly configured, it may keep your disk up needlessly. You can consult the thread at http://bit.ly/2kh6b17 in case you do not know what is keeping your disk awake. You should be able to infer the disk's state by running this command: $ sudo smartctl -d sat -n standby -a /dev/sda

If it exits with an error, your disk is still in standby and should have been spun-down.

Flash disk performance

One more thing to keep in mind when using flash storage (eMMC or SSD) is that they need periodic trimming to maintain their speed. Basically, in order to write to a storage block, you need to erase it first and this takes longer than writing to it. Normal filesystems do not do this erase when deleting data, so after a while disk performance drops significantly. To “revive” the disk, the trim operation informs the disk controller to erase all empty blocks, thus restoring write speeds. The trim operation must be supported by the filesystem and the disk controller. Again, using cron once a week to run fstrim can save you from slowdowns in the long term:

```
$ sudo crontab -e
#trim the eMMC once a week
15 0 0 * *      /sbin/fstrim / >/dev/null 2>&1
```

Governor

Performance and heat are also directly dependent on what governor you are using for the CPU. Keeping “performance” on gets you top performance, but also generates a lot of heat. In my tests, the best combination was a modified “ondemand” governor based on the recommandations at http://bit.ly/2jfaDjw. To enable it, make sure you select governor = ondemand in /media/boot/boot.ini, and set the rest of the parameters inside /etc/rc.local (test out the commands before). The commands below work for a 4.9/4.14 kernel and may differ for the 3.10 kernel:

```
$ sudo vi /etc/rc.local
echo 1 >
/sys/devices/system/cpu/cpufreq/ondemand/io_is
_busy
echo 10 >
/sys/devices/system/cpu/cpufreq/ondemand/sampl
ing_down_factor
echo 80 >
/sys/devices/system/cpu/cpufreq/ondemand/up_th
reshold
```

With the setting above, the CPU will ramp up frequency sooner and will consider IO usage as CPU, making IO intensive tasks influence the CPU frequency. This allows you to have great performance when needed and low heat when idle. In my usage, the little cores idle around 300MHz while the big cores idle at 200MHz.

Network performance – MTU

If you have a Gigabit network with proper cables, you can increase the MTU (Maximum Transmission Unit) on the ODROID-XU4’s onboard network. This will allow it to send larger packets which have less overhead and generate fewer interrupts on the receiving end. However, to benefit from it, you will need to have network devices (switches/routers) and end devices which support Jumbo frames. Ideally, Jumbo frames would need to be enabled on all network devices in your LAN, otherwise you might see dropped traffic or even devices unable to send large traffic to each other. For example, SSH works because it uses small packets, but getting a web page or transferring a file stalls the connection. If you do decide to enable jumbo frames, the ODROID-XU4’s magic MTU value is 6975 (**http://bit.ly/2jP9zDl**). You can enable it on the ODROID-XU4 inside /etc/rc.local:

```
$ sudo vi /etc/rc.local
#MTU
/sbin/ifconfig eth0 mtu 6975 up
```

Fastest transfers over sshfs/scp/sftp

Since SSH is a very flexible protocol and supports tunnelling and file transfer, it would be wise to use it at full speed. If you attempt a secure copy (scp) transfer on an ODROID-XU4 with the sshd process tied to the little cores, you will get about 15MB/s top speed. If you tie the sshd process to the big cores you get 40MB/s. If you are feeling adventurous and do not mind sacrificing some security, you can squeeze 50MB/s by lowering the encryption algorithm used. I did that by starting a different sshd instance (on port 2222) with different settings:

```
$ sudo wget -O /etc/systemd/system/ssh-
big.service
https://raw.githubusercontent.com/mad-ady/
odroid-xu4-optimizations/master/ssh-
big.service
$ sudo wget -O /etc/ssh/sshd_config_big
https://raw.githubusercontent.com/mad-ady/
odroid-xu4-
optimizations/master/sshd_config_big
$ sudo systemctl enable ssh-big
$ sudo systemctl start ssh-big
```

To mount or transfer a file using this new ssh service you will need to specifically specify the cipher (since it is disabled by default because it is considered weak). You can do so in an entry in ~/.ssh/config on the client:

```
Host odroid-big
Hostname odroid-ip
Port 2222
Ciphers arcfour
Compression no
```

To transfer files you can simply use the following command:

```
$ scp bigfile odroid-big:/remote/path
```

Tune systemd timeouts

It can be irritating to wait around for systemd to finish waiting for something that will never finish. You can tweak systemd's timeouts by modifying the global timeout settings in /etc/systemd/system.conf:

```
DefaultTimeoutStartSec=20s
DefaultTimeoutStopSec=10s
```

Note that some services (like networking) set explicit timeouts and you'll need to change those as well:

```
$ sudo vi /etc/systemd/system/network-
online.target.wants/
networking.service
TimeoutStartSec=30sec
```

Performance

Here are some performance metrics you can expect with the tweaks above and a Gigabit network. The client is an ODROID-C2 running Ubuntu 16.04, while the server is the ODROID-XU4. The download and upload directions are relative to the ODROID-XU4. The disk attached to the ODROID-XU4 has a write speed of 110MB/s. File transfers transferred an 8GB file filled with zeros (dd if=/dev/zero of=zero bs=1M count=8000 conv=fsync). Please note that part of the performance depends on your client as well. I was able to get better performance with a Linux PC than with the ODROID-C2 as a client.

Meet An ODROIDian: Ernst Mayer, Mathematician Extraordinaire

April 1, 2018 By Rob Roy Meet an ODROIDian

Please tell us a little about yourself. I've been living and working in Silicon Valley for roughly the last 20 years, doing algorithmic and coding work first for several tech startups and then for larger firms. Most of that work was related to EDA software for chip design. I've actually been semi-retired (in the sense that I still work, but mostly on my own research projects, not for pay) since getting laid off from my last large-company gig 6 years ago. I currently live in Cupertino, the heart of Apple country, though I never worked there. It's nice being close to the coastal hills, but the real estate prices and rents are really high. My sister and her family (husband and twin 9-year-old boys) live in the North bay, so I see them fairly often. I actually come from a science but non-computer-science background: my graduate degrees from the University of Michigan are in Aerospace Engineering and Mathematics. My PhD thesis was in theoretical fluid mechanics, specifically vortex flows. Lots of differential equations, applied mathematics and numerical analysis. My coding background coming out of college was scientific computing, that is, Fortran. I taught myself the rudiments of C and C++ after moving to Silicon Valley, and learned most of the rest of what I needed to know about those languages and CS-style algorithmics and data structures on the job.

Figure 1 – Ernst visiting the Canadian Rocky Mountains in 1986

How did you get started with computers? In the context of the kind of algorithmic and programming work, I ended up doing both by way of a career and ongoing research, it's important to note that I was, based on my early-college experiences, one of the most unlikely code geeks ever. I was a freshman at the University of Michigan in Fall 1981, and as such, was a member of one of the last few engineering-program gradating classes which was made to suffer through the freshman Fortran programming for engineers course as then constituted. The computer center was housed in a nondescript structure which was literally under a pedestrian-overpass bridge, and formally named the North University Building, but known to everyone by its acronym, NUBS. Everything was based around a then-standard mainframe-based tiered-price timeshare setup, said mainframe being an Amdahl system which I'm sure made for a suitably budget-priced alternative to the market-leading IBM 360 series. The real problem, as so often is the case, lay in the software: a non-IBM system compiler meant a non-IBM compiler, and while I don't know what alternative compiler offerings Amdahl Corp. may have had, I do know that the users of the system were stuck with a somewhat-experimental compiler from Waterloo U. in Canada. The problem with it was that the error messaging from same was so cryptic (often just obscure hexadecimal error codes, lacking even a program line number), so that for us newbies it effectively amounted to a binary syntax-error messaging system: 1 = "there are >= 1 syntax errors in your code, don't ask me where, so good luck finding them", and 0 = "congratulations! there are 0 syntax errors in your code, here are your (probably wrong) outputs." Even that would have been annoying but workable, but for the second major issue was that there were exceedingly few line-printer terminals and even fewer actual interactive terminals available, all of those were 100% occupied by computer science grad students, so we were limited to punched-card machines, for which there were also often wait lines. So once you finally got a seat at one of those and transcribed your handwritten initial attempt at a program for the current assignment on your little deck of playing-card-thick paper cards, you had to vacate your seat, take your card deck over to the nearest riffle-reader machine, maybe wait in line some more there, then head over to the paper-printout dispenser window to retrieve your program listing and, if you were lucky, get your outputs. Got a syntax-errors-remain crypto-message? Spend time poring over your program listing, identify likely error(s), then proceed back to the wait line outside the punch-card machine room. No syntax errors but errors in your outputs? More of the same. The final insult was that us bottom-rungers were allotted some ridiculously small amount of timeshare-account credits. I seem to recall $2 of said funny-money credits for the entire semester's projects, with no option to add more. Given the pricing system which was in place, the only way to turn said amount into a remotely-reasonable number of the above-described debug cycles was to use the facility in the dead of night, when prices were at their cheapest. The net result was that even the simplest 50-line programming assignment almost invariably turned into a hellish all-night work session.

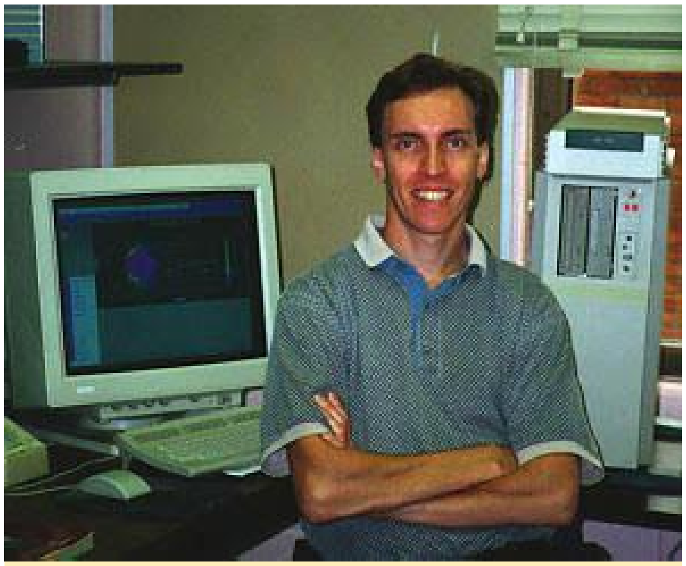


Figure 2 – Ernst in his office at Case Western Reserve University in 1997, in front of a DEC Alpha workstation, the first true 64-bit RISC architecture, which was a fine system for its day, but his ODROID-C2 has an order of magnitude more computing power

As a consequence, for the rest of my undergraduate days, the only kind of coding I did was on my trusty HP-41C programmable calculator. If someone from the future had come back and told my then-self that I would end up writing (almost entirely from scratch) and maintaining a program consisting of on the order of a half-million lines of code, I would've told them they were crazy. Of course fate, as it so often does, had other plans. While in graduate school, I earned extra money by working roughly 20 hours per week doing carpentry and maintenance work for a local landlord, and spent as many of my remaining waking hours as were left available indulging my love of "bashing around in the great outdoors": rock climbing and summer mountaineering trips, cycling, martial arts.

In the summer of 1987, having completed my master's degree I was preparing to start a PhD program in experimental fluid dynamics when during a mountain-biking session I took a nasty headfirst spill and ended up with a broken neck and paralysis from the chest down. So crawling around an equipment-filled experimental lab was out; math and computer work were in. Thankfully by then the university computing labs had moved to workstations and PCs, so I was able to do most of my graduate-research coding on DEC Vax workstations, with quality software, a vastly different experience than I'd had as a freshman.

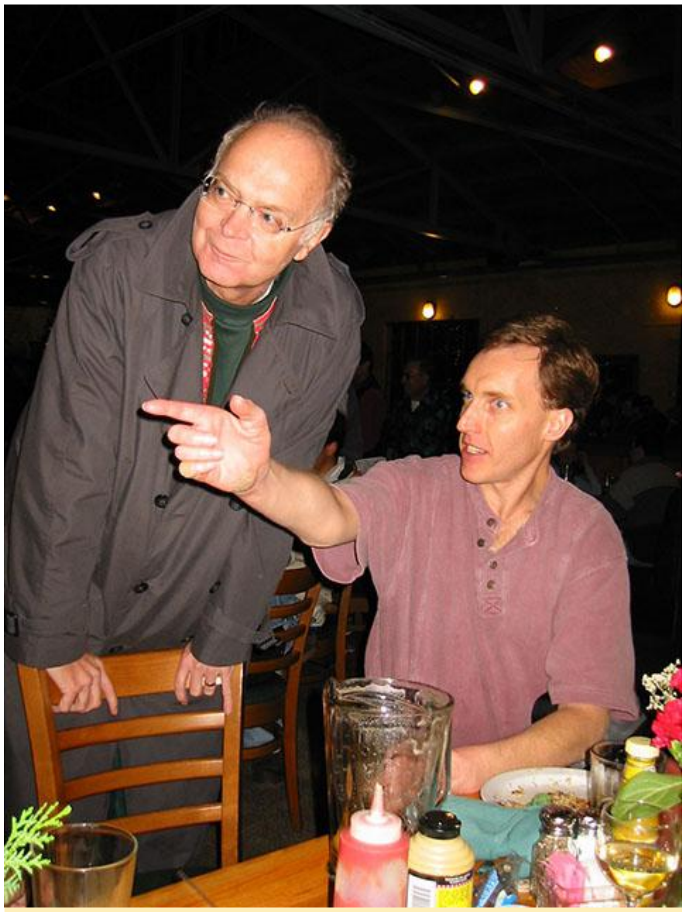


Figure 3 – Ernst with Stanford's Donald Knuth and a bunch of fellow Mersenners at the Mountain View Tied House to celebrate the discovery of the 39th Mersenne prime, M(13466917), December 2001

What attracted you to the ODROID platform? As I noted in last month's prime numbers article in ODROID Magazine, after spending much of the past five years writing assembly code for the various flavors of the x86 SIMD vector-arithmetic instruction set (SSE2, AVX, AVX2+FMA3, AVX512), last year I was also considering a first foray into adding SIMD support to a non-x86 processor family, and the ARMv8 CPU family's 128-bit vector support nicely fit the bill. After doing some homework regarding a low-cost but still reasonably high-performance development platform for that coding effort, the ODROID-C2 emerged as the top choice. I get about 50% greater throughput for my code on the C2 than on Raspberry Pi3. I also got performance benchmarks on a pre-release version of the ODROID N1 thanks to an ODROID forum user who was selected as one of the beta testers for the N1, and it looks very promising, using both of the N1's

CPUs (dual-core Cortex a72 and quad-core Cortex a53), I get more or less the same throughput for the "little" a53 socket as for a C2 (which is based on the same quad-core CPU), and the 'big' dual-core a72 CPU delivers about 1.5 times that throughput. Running jobs on both sockets simultaneously cuts about 10% off each of those single-socket throughputs so we don't get quite 2.5 times the C2's total throughput, but it's still more than double that of the C2. So the C2 was definitely a good choice as my first ODROID.

How do you use your ODROIDs? Last year, the 4-5 months after buying my C2 were spent doing heavy-duty inline-assembly coding, debug and performance tuning. Since releasing the ARMv8 code I've used my C2 pretty much the same way I hope other users of my code will do, large-prime-hunting for the GIMPS distributed-computing project. (In case anyone is wondering, I neither chose the project acronym nor take any offense from it.) It's also handy to have a machine with a different version of Linux and GCC installed than either my Macbook or my big Intel quad-core system, in case I need to track down a build issue that looks like it may be compiler or OS-version related.

Which ODROID is your favorite and why? The ODROID-C2 of course, at least until the N1 goes on sale.

What innovations would you like to see in future Hardkernel products? I'm a throughput hog, so I guess my answer boils down to "more sockets!" In particular, I'm interested in how many ODROID boards it would take to compete with a high-end Intel quad system, and how the respective prices for those similar-throughput options compare. Based on the relative performance of my Intel Haswell quad and the ODROID-C2 and N1, we'd need around 20 or so C2's to match the Haswell, and around 10 N1s. Once one gets to "around 10" the notion of that kind of compact multi-board bundle becomes no longer unreasonable to contemplate. The price based on the estimated retail price of the N1 is still a little higher than one would like, but not by much. Anyway, those are the kinds of daydreams this ODROIDer has. I also think linux micro-PCs are a great way to get kids interested in computers in a way that avoids the "walled garden" effect of PCs running proprietary OSes; I think maybe some kind of educational-outreach initiative to get such systems into the hands of low-income school children would be worthwhile for the Hardkernel folks to look into. That's the kind of thing that might attract government or private-foundation grant money to sponsor it.

What hobbies and interests do you have apart from computers? I've always been a person who likes to work not only with his head but also with his hands. One thing coding and mathematics lack is the tangible satisfaction that comes with physically building something, so I always like to have some kind of handicraft project going to fulfill that need. A couple years ago I built a really sturdy workbench using salvage lumber, mostly discarded heavy-duty computer-equipment shipping pallets. This winter's project was to build a display mount for a large (45kg) iron meteorite I'd bought some years back, out of a travertine limestone base topped with a block of natural sandstone drilled to hold three lengths of steel rod to act as a tripod to cradle the meteorite. The drilling proved to be the hardest part – one expects sandstone to be fairly soft and easy to work, but this block was sand which had apparently eroded from some kind of hard mineral, it ended up taking a diamond-encrusted hollow-core drill and hours of steady heavy pressure using a drill press and water to lubricate things. I went through a lot of ibuprofen that week!

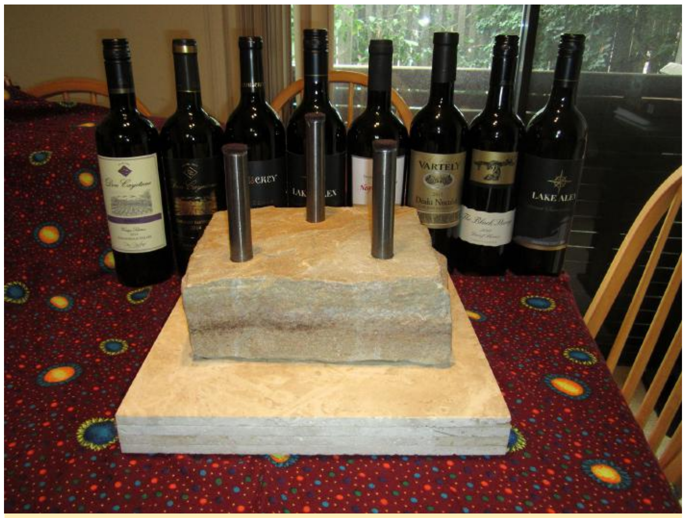
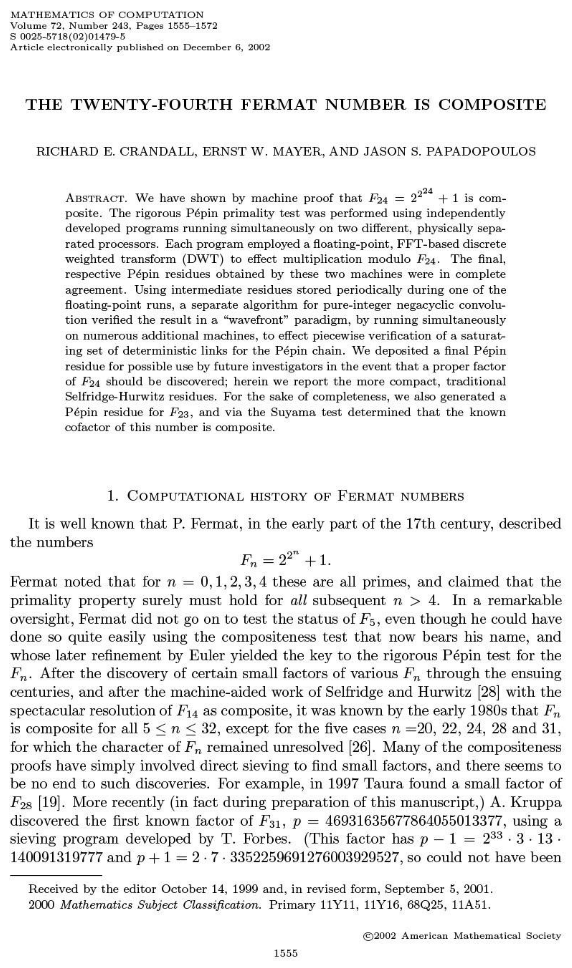


Figure 4 – Ernst is currently building a display mount for his iron meteorite

What advice do you have for someone wanting to learn more about programming and mathematics? Find a problem that really interests you which can serve as a learning vehicle for these subjects. I've had several such in my career, since my PhD research had aspects of differential equations, asymptotic analysis, perturbation theory, and linear algebra and eigensystems. My prime number work involves large-integer arithmetic and signal-processing algorithms, vector-arithmetic assembly code, plus a fascinating rich history featuring some of math's brightest luminaries. The world of science is full of such interesting problems; believe me, you'll know when one such grabs a hold of you. Making time in our distraction-filled and money-ruled modern world to pursue it, that is perhaps the trickiest issue.



MATHEMATICS OF COMPUTATION
Volume 72, Number 243, Pages 1555–1572
S 0025-5718(02)01479-5
Article electronically published on December 6, 2002

THE TWENTY-FOURTH FERMAT NUMBER IS COMPOSITE

RICHARD E. CRANDALL, ERNST W. MAYER, AND JASON S. PAPADOPOULOS

ABSTRACT. We have shown by machine proof that $F_{24} = 2^{2^{24}} + 1$ is composite. The rigorous Pépin primality test was performed using independently developed programs running simultaneously on two different, physically separated processors. Each program employed a floating-point, FFT-based discrete weighted transform (DWT) to effect multiplication modulo F_{24}. The final, respective Pépin residues obtained by these two machines were in complete agreement. Using intermediate residues stored periodically during one of the floating-point runs, a separate algorithm for pure-integer negacyclic convolution verified the result in a "wavefront" paradigm, by running simultaneously on numerous additional machines, to effect piecewise verification of a saturating set of deterministic links for the Pépin chain. We deposited a final Pépin residue for possible use by future investigators in the event that a proper factor of F_{24} should be discovered; herein we report the more compact, traditional Selfridge-Hurwitz residues. For the sake of completeness, we also generated a Pépin residue for F_{23}, and via the Suyama test determined that the known cofactor of this number is composite.

1. COMPUTATIONAL HISTORY OF FERMAT NUMBERS

It is well known that P. Fermat, in the early part of the 17th century, described the numbers

$$F_n = 2^{2^n} + 1.$$

Fermat noted that for $n = 0, 1, 2, 3, 4$ these are all primes, and claimed that the primality property surely must hold for *all* subsequent $n > 4$. In a remarkable oversight, Fermat did not go on to test the status of F_5, even though he could have done so quite easily using the compositeness test that now bears his name, and whose later refinement by Euler yielded the key to the rigorous Pépin test for the F_n. After the discovery of certain small factors of various F_n through the ensuing centuries, and after the machine-aided work of Selfridge and Hurwitz [28] with the spectacular resolution of F_{14} as composite, it was known by the early 1980s that F_n is composite for all $5 \leq n \leq 32$, except for the five cases n =20, 22, 24, 28 and 31, for which the character of F_n remained unresolved [26]. Many of the compositeness proofs have simply involved direct sieving to find small factors, and there seems to be no end to such discoveries. For example, in 1997 Taura found a small factor of F_{28} [19]. More recently (in fact during preparation of this manuscript,) A. Kruppa discovered the first known factor of F_{31}, $p = 46931635677864055013377$, using a sieving program developed by T. Forbes. (This factor has $p - 1 = 2^{33} \cdot 3 \cdot 13 \cdot 140091319777$ and $p + 1 = 2 \cdot 7 \cdot 3352259691276003929527$, so could not have been

Received by the editor October 14, 1999 and, in revised form, September 5, 2001.
2000 *Mathematics Subject Classification.* Primary 11Y11, 11Y16, 68Q25, 11A51.

©2002 American Mathematical Society

1555

Figure 5 – Ernst's first contribution to the field of computational number theory in 2002