ODROID Boot Logo • emuELEC • N2 Reviews and Benchmarks • MAME

ODROID Magazine

Year Six
Issue #66
Jun 2019

ZONEMINDER

BUILD YOUR OWN SURVEILLANCE SOLUTION

GOOGLE STADIA:

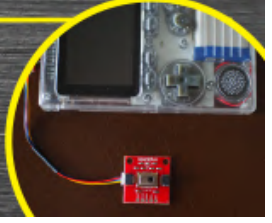
THE STREAMING GAME SERVICE THAT WILL BREATHE NEW LIFE INTO YOUR ODROID-XU4

MONKU RETRO GAMING:

ADDING CUSTOM BUTTONS AND CONTROLLERS

I ARE GO:

IR VISION TO YOUR ODROID-GO



emuELEC: Alpha Version Now Available For The ODROID-N2

June 1, 2019

EmuELEC is an emulation image that lets you run retro games: Atari, NES, SNES, TG16, PSP, PSX, N64, Genesis/Megadrive, and more.

Custom Boot Logo: Creating a custom boot image for the ODROID-C2

June 1, 2019

You can create a custom boot logo for the ODROID-C2 by following the simple instructions below. The logo will appear in the first few seconds of booting while the operating system loads. The basic image format of the ODROID-C2 boot logo file is described in the next section. Format Image ▶

ZoneMinder on ODROID-XU4: Build your own surveillance solution

June 1, 2019

ZoneMinder (ZM) is an integrated set of applications that provide a complete surveillance solution that allows the user to capture, analyze, record, and monitor an area through any CCTV or security cameras.

How to Build a Monku Retro Gaming Console - Part 1: Adding Custom Buttons And Controllers

June 1, 2019

This is a continuation of the Retro Gaming Console article from the May 2019 issue, where we learned how to assemble the basics of a retro gaming console.

I aRe GO: Qwiic-ly Add IR Vision to Your ODROID-GO

June 1, 2019

Now that you're all up to speed on the Qwiic Adapter article, here's another Qwiic project that will give your ODROID-GO the gift of infrared (IR) vision.

The G Spot: Your Go-to Destination for all Things Android Gaming - Google Stadia

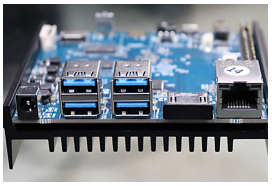
June 1, 2019

I would be remiss if I didn't at least mention Google Stadia. This is a streaming game service that promises to breathe new life into your ODROID-XU4 gaming experience—even the lower-powered ODROID-C1 should, in theory, benefit from this Google service. What the heck is Google Stadia? Simply put, as announced ▶

Custom Boot Logo: Creating a custom boot image for the ODROID-N2

June 1, 2019

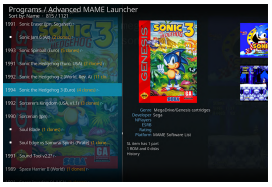
Ever wanted to create a custom boot logo for the ODROID-N2? By following the simple instructions below, the logo will appear in the first few seconds of booting while the operating system loads. Enjoy!



ODROID-N2 Review: Good Performance in Linux Benchmarks

June 1, 2019

Hardkernel's newest single board computer, the ODROID-N2 is our hottest machine, come see it!



Kodi and Advanced Mame on ODROID-XU4 - Part 1

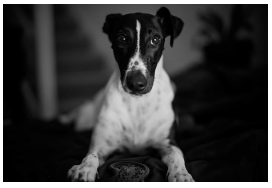
June 1, 2019

This is a guide to setup Kodi with Mame, on the ODROID-XU4 SBC, making it a nice media and game center. It lists all the steps to install the needed software on Ubuntu Linux.

ODROID-N2: Benchmarks

June 1, 2019

The benchmarks of Hardkernel's ODROID-N2, a new board replacing the cancelled ODROID-N1.



Meet An ODROIDian: Ry (@lordhardware)

June 1, 2019

Our monthly column presenting a member of our outstanding community, come! we have cookies!

emuELEC: Alpha Version Now Available For The ODROID-N2

June 1, 2019 By @shanti Gaming, ODROID-N2

EmuELEC is an emulation image that lets you run retro games: Atari, NES, SNES, TG16, PSP, PSX, N64, Genesis/Megadrive, and more. To view emuELEC in action, please check out the video at https://youtu.be/ZPq0lt1XcI0. This is the first release of my project, which is based on CoreELEC. CoreELEC is a "just enough OS" Linux distribution for running Kodi on popular low-cost hardware, which is itself a minor fork of LibreELEC. However, emuELEC is a pure emulation project that does not include Kodi.

Figure 1 - Main Menu for the emuELEC emulation operating system

WiFi should be easy to configure by editing the file called wifi.txt in the root of the sdcard (edit it right after you burn the image to the SD card), or by creating a wifi.txt file in /storage/.config/wifi.txt. Put your Wifi details inside the file as shown below (don't put anything else):

YOURSSID:YOURPASSWORD

Figure 2 - Settings for the ODROID emuELEC emulation operating system

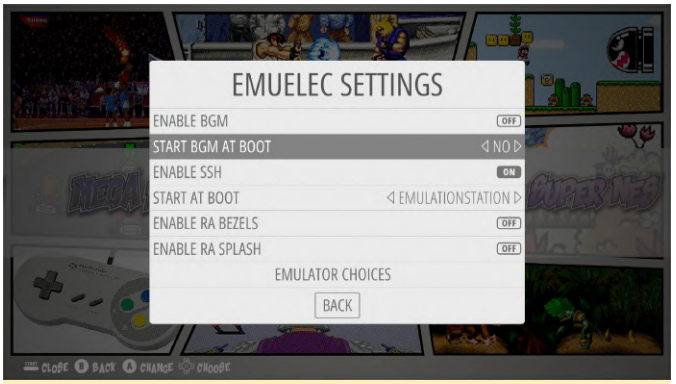


Figure 3 - Settings for the ODROID emuELEC emulation operating system

Then you can select the wifi script in the "Retropie" category in Emulation Station. If all went well you should have Wifi. For now it only connects to wlan0, and for the N2 you will need a wifi dongle. To add background music, copy the MP3s to /storage/roms/BGM and enable BGM in Emulation Station.

SSH user: root SSH password: emuelec

The image includes:

- Emulationstation
- advancemame
- PPSSPP
- Reicast
- Retroarch with a lot of included cores
- Support for background music (it reads mp3s in /storage/roms/BGM/)
- Easy external USB roms (just put a file named emuelecroms inside your /roms folder in the USB)
- Theme downloader

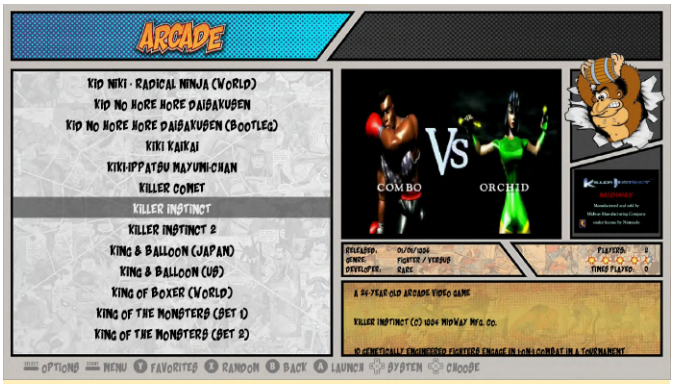


Figure 4 - Arcade Emulator running on the ODROID emuELEC image

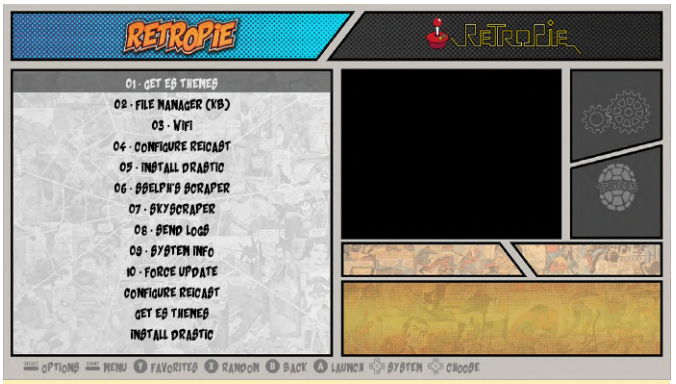


Figure 5 - RetroPie running on the ODROID emuELEC image

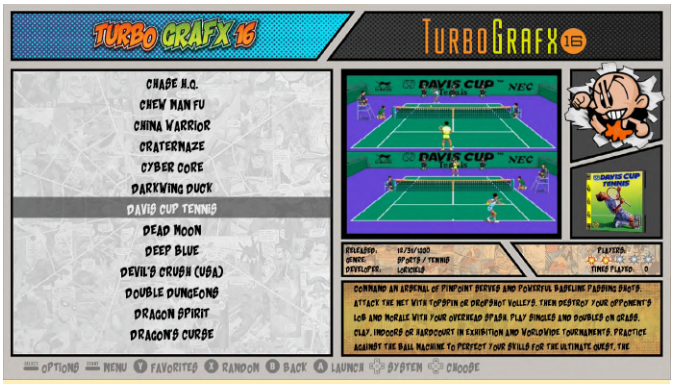


Figure 6 - TurboGrafx running on the ODROID emuELEC image

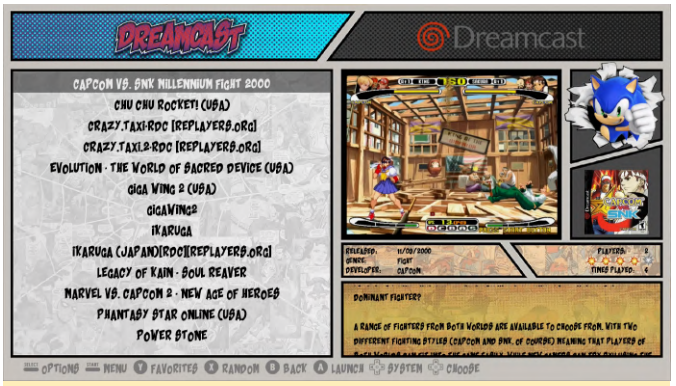


Figure 7 - Dreamcast emulator running on the ODROID emuELEC image

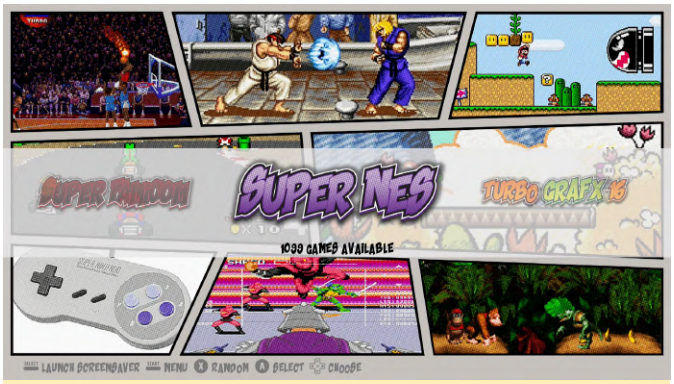


Figure 8 - SuperNES emulator running on the ODROID emuELEC image

Known Issues:

- When you exit an emulator, sometimes the frame buffer does not refresh, and the image stays frozen, but you can still control it and hear sound
- After returning to Emulation Station from an emulator (if the first issue does not happen), the screen has a lot of tearing
- If you use video snaps, after booting it takes a bit to load the videos, but once loaded it works as it should
- Sometimes (very rare), you can hear white noise for a fraction of a second
- Some of the included RetroPi scripts do not fully work
- If you have BGM enabled and Emulation STation goes into screensaver mode, the music will continue to play
- To install, just burn the image to a microSD using etcher (or your favorite program). I hope you like it, and please let me know of other issues you might find. Keep in mind that this is an alpha release, and some things might not work as they should or maybe not at all.

Figure 9 - Street Fighter runs smoothly on the ODROID emuELEC image

The latest release of emuELEC is available at https://github.com/shantigilbert/EmuELEC/releases/tag/v2.5.2. For comments, questions and suggestions, please visit the original post at https://forum.odroid.com/viewtopic.php?f=182&t=34355, or on Discord at https://discord.gg/QqGYBzG.

Custom Boot Logo: Creating a custom boot image for the ODROID-C2

June 1, 2019 By Justin Lee Android, Linux, ODROID-C2, Tinkering, Tutorial

You can create a custom boot logo for the ODROID-C2 by following the simple instructions below. The logo will appear in the first few seconds of booting while the operating system loads. The basic image format of the ODROID-C2 boot logo file is described in the next section.

Format

```
Image Format : 24-bit Windows BMP image or 24-bit
Windows Gzipped BMP image (Without meta-data)
Image Size : 1280 by 720
Color Depth : 24bpp
File Name : 'boot-logo.bmp' or 'boot-logo.bmp.gz'
```

A sample bmp file can be found at: https://wiki.odroid.com/_media/en/boot-logo.bmp.gz We recommend using GIMP or KolourPaint as editing programs. Here are the advanced options:

GIMP

- Export as Windows BMP - Compatibility Options : Do not write Color Space Information - Advanced Options : 24 bits Color - Name : "boot-logo.bmp"

KolourPaint

- Save Image as - Filter : Windows BMP image - Convert to : 24-bit Color

Size Limitation

You must keep the size of your bmp file under 2MB, because the logo partition is limited to 2MB. Gzip BMP format is supported, so if the size is over 2MB, you can use bmp.gz file.

```
$ gzip boot-logo.bmp
$ ls -l boot-logo.bmp.gz
```

Auto scaling option

On the ODROID-C2 uboot, image scaling for boot logo is supported, so displayed boot logo will be fixed

automatically for output mode as described in boot.ini. For example, in the case of using mode “1024x600p60hz”, the boot logo will be displayed as 1024×600 even though actual size of bmp file is 1280×720.

Replacing the boot logo

ODROID-C2 scans the existence of the following three parts in numerical order. boot-logo.bmp in VFAT partition boot-logo.bmp.gz in VFAT partition logo data in Android LOGO partition

Android

On Android, you can replace boot logo with your custom image. There are two methods to change boot logo image. Add an image into VFAT partition. Rewrite image data into Android LOGO partition using fastboot.

Method 1: VFAT Copy the new boot-logo.bmp, or boot-logo.bmp.gz, to the VFAT partition.

Method 2: Android Logo Partition If you want to replace the logo data in the logo partition, please follow this guide:

First, you must get into your U-Boot command line while pressing ‘ENTER’ key when your ODROID-C2 is powered up and execute fastboot command from U-Boot and connect with your desktop using micro USB cable.

```
odroidc2# fastboot
```

Next, run the fastboot command from your desktop.

HOST PC

```
$ fastboot flash logo boot-logo.bmp.gz
```

or

```
$ fastboot flash logo boot-logo.bmp
```

Warning

If you will use bmp data on the logo partition, make sure there is NO boot-logo.bmp.gz file on your VFAT area, because U-Boot checks, first, if there are boot-logo.bmp/boot-logo.bmp.gz on VFAT area and then checks the logo partition.

Ubuntu

With Ubuntu, the display logo option is NOT included by default. So, you need to add a boot logo image into VFAT partition. The method using the LOGO partition is not available on Ubuntu.

How to Add showlogo Command in boot.ini

1080p60hz

On U-Boot, the default logo display logic works with 1080p60hz display resolution. So you don't need to add/modify any related commands, but make sure the boot logo file exists in the aforementioned locations. Another resolution other than 1080p60hz, you should add the commands to your boot.ini before bootcmd is executed. Please check if there is 'showlogo' command in your boot.ini first. If not, refer to the following:

Android

```
showlogo ${hdmimode}
```

Ubuntu

```
# Boot Arguments
if test "${display_autodetect}" = "true"; then usb
pwren; hdmitx edid; fi
if test "${m}" = "custombuilt"; then setenv cmode
"modeline=${modeline}"; fi

### You should add the following lines after
**hdmitx edid** command.
showlogo ${m}
setenv logoopt "osd1,loaded,0x3f800000,${m}"

# Boot Arguments - Add logo args on the existing
bootargs parameter
setenv bootargs "root=UUID=e139ce78-9841-40fe-
8823-96a304a09859 rootwait ro ${condev}
no_console_suspend hdmimode=${m} ${cmode}
m_bpp=${m_bpp} vout=${vout} fsck.repair=yes
net.ifnames=0 elevator=noop disablehpd=${hpd}
max_freq=${max_freq} maxcpus=${maxcpus}
monitor_onoff=${monitor_onoff}
disableuhs=${disableuhs}
mmc_removable=${mmc_removable}
usbmulticam=${usbmulticam} ${hid_quirks}
logo=${logoopt}"
```

Custom Native Image Resolution

If you want to use a native resolution for a bmp image like 1920×1080, 1024×600 (for VU7+) or 800×480 (for VU7), set arg[2]/arg[3] of showlogo command as follows:

```
# help showlogo
showlogo - Displaying BMP logo file to HDMI screen
with the specified resolution

Usage:
showlogo  [ ]
        resolution - screen resoltuion on HDMI
screen
                '1080p60hz' will be used by
default if missing
        bmp_width (optional) - width of logo bmp
file
                '1280' will be used by default if
missing
        bmp_height (optional) - height of logo bmp
file
                '720' will be used by default if
missing
```

Replace boot logo image with yours as described in previous sections and then modify 'showlogo' command in boot.ini. Here are some examples:

Logo image size of width 1920 and height 1080

If your monitor's resolution is 1920×1080 and you want to set a bmp file in 1920×1080, Set the command in boot.ini with:

```
setenv hdmimode "1080p60hz"
showlogo ${hdmimode} 1920 1080
```

Logo image size of width 1024 and height 600

```
setenv hdmimode "1024x600p60hz"
showlogo ${hdmimode} 1024 600
```

Logo image size of width 800 and height 480

```
setenv hdmimode "800x480p60hz"
showlogo ${hdmimode} 800 480
```

Workaround for Logo Splash Issue with VU7+

For some specific cases, when VU7+ is using an extra power source, strange colors and a flashing screen issue during display initialization on u-boot stage might be visible. To resolve this issue, the following workaround will fix it:

Update u-boot

Click the following link to download the boot loader to fit displays with 1024x600p60hz, DVI mode: http://dn.odroid.com/S905/BootLoader/ODROID-C2/c2_vu7plus_splash_20180720 Once downloaded, copy tar.gz file to /media/boot and boot the C2 system

```
# cd /media/boot
# tar xvfz c2_vu7plus_splash_20180720.gz
# cd ./sd_fuse
# ./sd_fusing.sh /dev/mmcblk0
```

(reboot)

Setup boot.ini

Some points in boot.ini should be adjusted.

```
### set "display_autodetect" as "false"
setenv display_autodetect "false"

### block a default "m" and change "m" as
"1024x600p60hz"
setenv m "1024x600p60hz"

### HDMI DVI/VGA modes
### set "vout" as "dvi"
setenv vout "dvi"

### turn on USB power
usb pwren

### add "logoopt"
setenv logoopt "osd1,loaded,0x3f800000,${m}"

### add "logo=${logoopt}" in "bootargs"
setenv bootargs "root=UUID=e139ce78-9841-40fe-
8823-96a304a09859 rootwait ro ${condev}
no_console_suspend logo=${logoopt} hdmimode=${m}
${cmode} m_bpp=${m_bpp} vout=${vout}
fsck.repair=yes net.ifnames=0 elevator=noop
disablehpd=${hpd} max_freq=${max_freq}
```

```
maxcpus=${maxcpus} monitor_onoff=${monitor_onoff}
disableuhs=${disableuhs}
mmc_removable=${mmc_removable}
usbmulticam=${usbmulticam} ${hid_quirks}"
```

References

You can refer to the history of this issue from these ODROID Forum pages: https://forum.odroid.com/viewtopic.php?f=141&t=29262#p209113https://forum.odroid.com/viewtopic.php?f=141&t=31590#p229069

Sample Source Code

This section describes U-boot code change history. In the case of 1024x600p60hz, you don't need to follow this section, but refer to this section and use the pre-built U-boot. https://wiki.odroid.com/odroid-c2/application_note/software/bootlogo#update_u-boot

u-boot/board/hardkernel/odroidc2/odroidc2.c : in function board_late_init https://github.com/hardkernel/u-boot/blob/odroidc2-v2015.01/board/hardkernel/odroidc2/odroidc2.c#L468

```
#ifdef CONFIG_DISPLAY_LOGO
  /* run_command("showlogo 1080p60hz", 0); */
  run_command("showlogo 1024x600p60hz", 0);
#endif

u-boot/common/cmd_showlogo.c
https://github.com/hardkernel/u-
boot/blob/odroidc2-
v2015.01/common/cmd_showlogo.c#L119
        /*
  if (NULL == getenv("vout_mode"))
    setenv("vout_mode", "hdmi");
        */
        setenv("vout_mode", "dvi");
```

This guide is available on the ODROID wiki at the follow link: https://wiki.odroid.com/odroid-c2/application_note/software/bootlogo

Android Boot Animation

In the case of Android, you can use bootanimation.zip method to show your custom logo using animation. Please refer to this reference site: https://android.googlesource.com/platform/frameworks/base/+/master/cmds/bootanimation/FORMAT.md

The system selects a boot animation zip file from the following locations.

```
/system/media/bootanimation.zip
/oem/media/bootanimation.zip
```

Before the copy process, you need to change root filesystem permission to r/w and copy your bootanimation.zip into /system/media/ folder.

```
shell@odroidc2:/ $ su
root@odroidc2:/ #
root@odroidc2:/ # mount -o rw,remount /
[  357.892532@2] EXT4-fs (mmcblk0p2): re-mounted.
Opts: (null)
```

Samples of boot animation

Here are sample capture videos with ODROID-C2 Android Marshmallow. To display these boot animation samples, the following reference files are used: [Please note the sources of the files.] https://forum.xda-developers.com/android/themes/bootanimation-android-marshmallow-t3180984

Figure 1 - Boot Animation Example 1

(figure 01 - boot animation example)

Directory Layout

desc.txt part0 part1 part2 part3 part4 desc.txt

```
814 214 60
c 1 30 part0
c 1 0 part1
c 0 0 part2
c 1 64 part3
c 1 15 part4
```

Size of png files is 814 by 214.

Figure 2 - Boot Animation Example 2

Boot Animation Example 2

(figure 02 - boot animation example)

Directory Layout

desc.txt Part0 Part1 desc.txt

```
800 1280 24
p 1 0 Part0
p 0 0 Part1
```

The size of png files is 800 by 1280.

The following guide can be found on the ODROID wiki at **https://wiki.odroid.com/odroid-c2/application_note/software/bootlogo**.

ZoneMinder on ODROID-XU4: Build your own surveillance solution

June 1, 2019 By Michele Matacchione ODROID-XU4, Tutorial

2ZoneMinder (ZM) is an integrated set of applications that provide a complete surveillance solution that allows the user to capture, analyze, record, and monitor an area through any CCTV or security cameras.

Main features

- Monitor from anywhere: ZoneMinder has a full-featured, web-based interface you can access from any Internet-accessible device
- Use any camera: ZoneMinder allows you to use any analog or IP-enabled camera
- Control of your data: ZoneMinder is fully on premises; it allows you to own your data and control where it goes
- Run small or super big systems: suitable for home and small business use, as well as multi-server enterprise deployments. It is compatible with many platforms, including ARM technology (ODROID is built on an ARM platform)
- Keep track of what matters: ZoneMinder allows you to browse information intuitively. Drill down to what you want to see in a matter of seconds
- Actively maintained and free of charge: a team committed to open source actively maintains ZoneMinder

Recently, I moved the ZoneMinder application from my old Radxa Rock Pro (an ARM board) to the more powerful ODROID-XU4. The better and easier to install working setup I found is ODROID-XU4 - Ubuntu 16.04.3 LTS – ZoneMinder 1.29*.

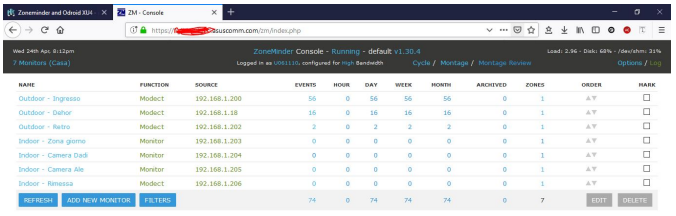


Figure 1 – ZoneMinder console, configured with 7 cameras

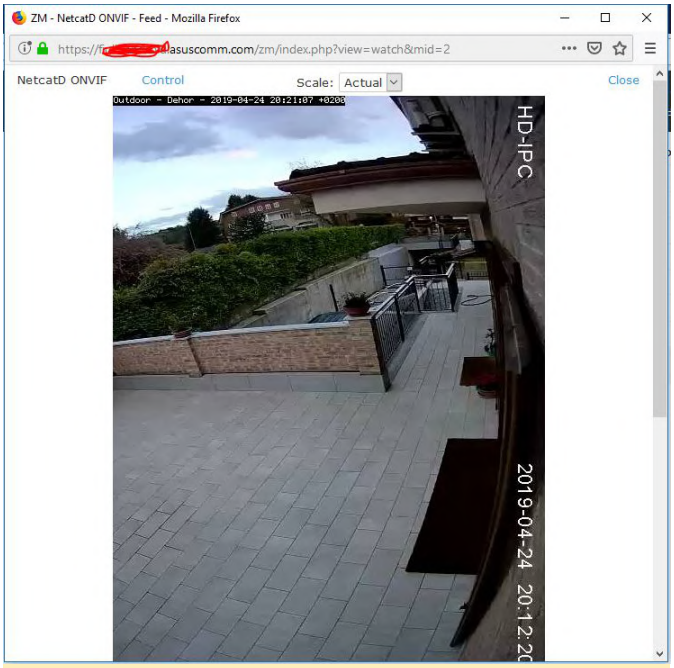


Figure 2 – ZoneMinder watching camera #2

Installation

Let's install ZoneMinder on our ODROID-XU4 board. On your SD card, install the Ubuntu 16.04.3 LTS image (the upstream Release 4.14.y) provided by Hardkernel at the address: **https://wiki.odroid.com/odroid-xu4/os_images/linux/ubuntu_4.14/20171213**

Then upgrade the system:

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

Now install LAMP (Linux, Apache, MySQL, PHP) on the board:

```
$ sudo apt install apache2
$ sudo apt install mysql-server
$ sudo apt install php libapache2-mod-php php-mysql
```

Now install ZoneMinder 1.29:**

```
$ sudo -i
```

Tweak MySQL configuration (not needed for ZM 1.32 or greater):

```
$ rm /etc/mysql/my.cnf (this removes the current symbolic link)
$ cp /etc/mysql/mysql.conf.d/mysqld.cnf /etc/mysql/my.cnf

$ nano /etc/mysql/my.cnf
```

In the [mysqld] section add the following:

```
$ sql_mode = NO_ENGINE_SUBSTITUTION
```

Then restart MySQL:

```
$ systemctl restart mysql
```

Now install ZoneMinder:

```
$ apt-get install zoneminder
```

Create ZM database (not needed for ZM 1.32 or greater)::

```
$ mysql -uroot -p < /usr/share/zoneminder/db/zm_create.sql
$ mysql -uroot -p -e "grant lock tables,alter,drop,select,insert,update,delete,create,index,alter routine,create routine, trigger,execute on zm.* to 'zmuser'@localhost identified by 'zmpass';"
```

Add permissions:

```
$ chmod 740 /etc/zm/zm.conf
$ chown root:www-data /etc/zm/zm.conf
$ chown -R www-data:www-data /usr/share/zoneminder/
```

Enable modules and ZoneMinder configuration:

```
$ a2enmod cgi
$ a2enmod rewrite
$ a2enconf zoneminder

$ a2enmod expires
$ a2enmod headers
```

Enable ZoneMinder to system startup:

```
$ systemctl enable zoneminder
$ systemctl start zoneminder
```

Configure php.ini with the correct timezone:

```
$ nano /etc/php/7.0/apache2/php.ini
```

Insert your timezone:

```
$ [Date]
$ ; Defines the default timezone used by the date
```

```
functions
$ ; http://php.net/date.timezone
$ date.timezone = America/New_York
```

Restart Apache:

```
$ systemctl reload apache2
```

Done!

Now, you can find the ZM web page on **http://IP_of_the_board/zm** and add cameras.*** The same procedure is valid for ZM 1.32 (the last stable ZM release), thus avoiding commands not needed for ZM 1.32. In this case, before installing ZM you will need to add the ZoneMinder repo by Isaac Connor (the ZM maintainer on Ubuntu):

```
$ add-apt-repository ppa:iconnor/zoneminder-1.32
```

Note that ZM 1.32 is available for Ubuntu 18.04, too. Please take into consideration that the ffmpeg in 18.04 Hardkernel image is not compatible and should be tweaked (see **https://forum.odroid.com/viewtopic.php?f=95&t=33745#p247653**)

Notes

- There are other possible combinations, to be discussed later;
- There is no need to install dedicated ZoneMinder repository since ZM 1.29 is already available on Ubuntu 16.04 (xenial) repos
- All types of cameras can work: ffmpeg and mjpeg cameras connected via Wi-Fi, Ethernet cable, or USB

Other tweaks and tricks

After installation, check if ZM is working well on your LAN. See previous posts for ZM installation and FFMPEG modification, if you use HK 18.04 distro.

Open a Web browser (Firefox is recommended): **http://LANserverip/zm** (e.g. **http://192.168.1.200/zm**). If the ZoneMinder page appears, you are in good shape.

Set the ZM access in authenticated way: Go in Options/System. Flag on OPT_USE_AUTH. Close the browser, then open the browser and you will be asked to enter Userid/password. Use admin/admin*.

Modify userid/password: Go into Options/Users. Change username from admin to whatever1, set the new password to whatever2. Close the browser, then open the browser and you will be asked to enter Userid/password. Use whatever1/whatever2*

Check that AUTH_TYPE = builtin. If no, then set it: Check AUTH_RELAY = hashed and again if no, then set it. Change AUTH_HASH_SECRET to whatever*

Enable https for ZoneMinder: This is the easy way. There is another method using letsencrypt certificates. Go in terminal and add SSL to Apache2 by first creating a self-signed certificate:

```
$ make-ssl-cert generate-default-snakeoil --force-overwrite
```

This will create the following files:

```
/etc/ssl/private/ssl-cert-snakeoil.key
/etc/ssl/certs/ssl-cert-snakeoil.pem
```

Activate Apache SSL module:

```
$ a2enmod ssl
```

Activate Apache default ssl virtual host:

```
$ a2ensite default-ssl
```

Restart Apache:

```
$ service apache2 restart
```

You should now be able to access the Web server using **https://LANserverip/zm** (e.g. **https://192.168.1.200/zm**).

Enable port-forwarding of 443 port on your router. On an ASUS model, go to WAN/Port Forwarding:

Service name: HTTPS_Port_Forwarding Port: 443 Local IP: LANserverip (e.g.: 192.168.1.200) Local port: 443 Protocol: TCP

You should now be able to access the Web server using **https://WANserverip/zm** (e.g. **https://5.157.104.224/zm**).

You can dynamically link your WANserverip to a static host name (e.g. **https://hostname/zm**) through Dynamic DNS functionalities on your router (e.g., on ASUS and DLink) or using services like NOIP. This way you can securely expose ZoneMinder service in WAN--

authenticated access and accessible in encrypted https, so all information flow on WAN will be encrypted--through a static Web address.

If you are asked to restart ZoneMinder, go to Terminal and enter the following command:

```
$ sudo service zoneminder restart
```

How to configure the camera There are four ways to do this--three easy, the fourth manual:

1. Use PRESETS 2. Use ONVIF automatic detection 3. Google it: Search for “inserting {camera model} ZoneMinder" 4. Manual configure

For your inspiration, the following are configurations for my three types of cameras:

A: FOSCAM FI8910W (old PTZ MJPEG model used for indoor)

General Name: whatever Server: None Source Type: Remote Function: Monitor (to view only)

Source Remote Protocol: HTTP Remote Method: Simple Remote Host: 192.168.1.203 Remote Host Port: 200 Remote Host Path: /videostream.cgi?user=user-on-camera&pwd=pwd-on-camera&rate=3 Target Colorspace: 24 bit color Capture Width: 640 Capture Height: 480

Leave other values as the defaults, other sheet Control on PTZ commands in next session.

B: FOSCAM FI8918W (PTZ FFMPEG model used for indoor)

General Name: whatever Server: None Source Type: Ffmpeg Function: Modect (to view and to record on motion detection)

Source Source Path: rtsp://user-on-camera:pwd-on-camera@192.168.1.206:200/videoSub Remote Method: TCP Options: blank Target Colorspace: 24 bit color Capture Width: 640 Capture Height: 480

Leave other values as the defaults, other sheet Control on PTZ commands in next session.

C: Jidetech POE PTZ FFMPEG dome model used for outdoor and LEFTEK POE PTZ FFMPEG dome model used for outdoor

General Name: whatever Server: None Source Type: Ffmpeg Function: Modect (to view and to record on motion detection)

Source Source Path: rtsp://user-on-camera:pwd-on-camera@192.168.1.200:554/1/h264minor Remote Method: TCP Options: blank Target Colorspace: 24 bit color Capture Width: 640 Capture Height: 480

Leave other values as the defaults, other sheet Control on PTZ commands in next session.

How-to record on SSD

- Install cifs-utils

```
$ sudo apt-get install cifs-utils
```

-Create directory to mount SSD

```
$ mkdir /home/odroid/Documents/STORAGE
```

-Edit /etc/fstab

```
$ sudo nano /etc/fstab
```

Insert the following lines:

Note: the correct UUID of SSD can be discovered through ls -l /dev/disk/by-uuid/*

```
#SSD
UUID=BE28A67028A626FD
/home/odroid/Documents/STORAGE auto
nosuid,nodev,nofail 0 0
```

Press CTRL+O, then CTRL+X

```
$ sudo mount -a
```

Reboot, if needed.

```
$ mkdir /home/odroid/Documents/STORAGE/ZM-IMAGES
$ mkdir /home/odroid/Documents/STORAGE/ZM-
IMAGES/images
$ mkdir /home/odroid/Documents/STORAGE/ZM-
IMAGES/events
$ mkdir /home/odroid/Documents/STORAGE/ZM-
IMAGES/temp

$ sudo nano /etc/fstab
```

Insert the following lines to create a bind:

/home/odroid/Documents/STORAGE/ZM-IMAGES/images /var/cache/zoneminder/images none defaults,bind 0 0 /home/odroid/Documents/STORAGE/ZM-IMAGES/events /var/cache/zoneminder/events none

defaults,bind 0 0 /home/odroid/Documents/STORAGE/ZM-IMAGES/temp /var/cache/zoneminder/temp none defaults,bind 0 0

Press CTRL+O, then CTRL+X

```
$ sudo mount -a

$ sudo chown -R www-data:www-data /home/odroid/Documents/STORAGE/ZM-IMAGES
```

A similar procedure could be followed to mount a USB HD or a network storage disc.

How to record on motion detection In Console, click on column zones. For the camera, you want MOTION DETECTION. Click again on the picture. Set the area you want to check for detection. Leave unchanged if you want to check the full area. Now set the following:

Name: whatever Type: Active Unit: Pixels Alarm Colour: whatever Alarm check method: Blobs Min-Max Pix Thresh: 40 - 0 Filter: 5 -5 Min/max Alarmed area: 200 - 0 Min/max Filtered area: 20 - 0 Min/max Blob area: 20 - 0 Min/max Blob: 2 - 0 Overload Frame: 2

Save. Put in Console and the camera in MODECT mode (column function).

How-to implement PTZ commands

Enable PTZ commands: go in Options/System: flag on OPT_CONTROL. In Console, click in column source, for the camera you want to enable.

PTZ:

Control Controllable: flag Control Type: choose your camera model Control device: according tp your model (in my case blank) Control address: camera_ip:camera_port (in my case 192.168.1.200:8999)

How to Build a Monku Retro Gaming Console - Part 1: Adding Custom Buttons And Controllers

June 1, 2019 By Brian Ree ODROID-C2

This series will teach you how to assemble the basics of a retro gaming console using Monku and an ODROID-C1+ / C2. This first installment will show you how to build the custom buttons and controllers for the project.

Tools Needed

- A small screwdriver set that contains various small phillips head screwdrivers.
- A soldering kit including soldering iron with fine tip, solder with flux, temperature control, if possible.
- Masking tape or painter's tape.
- A clean static free work surface.
- Drill and a good selection of drill bits.
- Monitor or TV with HDMI support to test the device.

I used this soldering kit (https://amzn.to/2Jyy85h). It costs around $19.00 and although on the cheap side, it comes with everything you will need for this build and the soldering iron comes with multiple tips including a fine tip and has a temperature control which is awesome. Now I had not soldered anything since high school and I was able to do this and not mess it up. I also do not have the steadiest hands as I have probably had about 3 cups of coffee before any given moment in time. If I can do it, you can do it.

1: Parts Needed

- ODROID-C1+ / ODROID-C2 x1: $35.00 / $46.00 (https://www.hardkernel.com/shop/odroid-c1/)
- Case x1: $4.50 (https://www.hardkernel.com/shop/odroid-c2-c1-case-black/)
- Button Set: $7.99 (https://amzn.to/2HNRekI)
- Breadboard Cable Set x1: $7.99 (https://amzn.to/2wgeXo8)
- 64GB Micro SD Card x2: $16.99 (https://amzn.to/2JyNjLZ)

- HDMI Cable x1: $1.00 (https://www.hardkernel.com/shop/hdmi-1-4-cable-type-a-a/)
- Micro USB Cable x1: $1.50 (https://www.hardkernel.com/shop/micro-usb-cable/)
- USB Charging Block 5V/2A x3: $11.99 (https://amzn.to/2VXIfY0)
- GameSir Wired Controller x1: $17.00 (https://bit.ly/2JxEu4V)

The total project cost as described above and not including shipping or needed tools is around $104. If you exclude the custom control buttons or can buy the parts individually, however, you can save around $16. Also you do not need two micro SD cards. I like to have a spare in case one is corrupt and the dual set listed above has a great price. Also, the card is well rated and from my personal experience I have only had one fail unexpectedly out of 12 or so I have been using regularly during the configuration and development of these devices. If you have an HDMI cable, a micro USB cable, and a USB charger 5V/2A then you can save even more on the cost of this project. While the price listed is around $104 you can probably get it done for around $80. Not too bad, once you see what these things can do.

2: Introduction and Tutorial Goals

This article will show you in detail how to build a Monku Retro 1 (ODROID-C1+) or Monku Retro 2 (ODROID-C2) video game console from scratch. First, some of the hardware basics: the hardware button provides a hard reset of the device. The software button hooks up to GPIO pins and is setup to record how long it is being held and execute a certain command for the recorded time period. So, for instance, a 2 second hold is a software reboot, a 4 second hold is a software shutdown etc. If you do want to add custom control buttons we will cover that here in detail. Everything from the parts, to the pins, to how to solder it all up. This feature is optional. Your device will work fine without them but in the event of a crash you will have to power cycle the console by hand and this action could corrupt the Linux file system. The hardware reset button makes this a little easier, but the software control button with it's OS shutdown and restart calls are much safer to use. You can always go back and add them in, not a problem. We will also cover the software setup including installing and configuring Ubuntu, retroarch, and antimicro in part 2 and 3 of this tutorial series.

R1 / C1+ Features:

- ODROID Goodness
- Custom Software Control Button
- Custom Hardware Reset
- Support for Atari 2600, Atari 7800, ColecoVision, MSX-1, MSX-2, NES, GameBoy, GameBoy Color, Sega SG-1000, Sega Mark III, and Sega Master System configured and ready to go
- Retroarch with XBM, custom scripts to monitor the software button, start retroarch, maintain antimicro
- Configured for low memory usage and for use with included controller
- Every ROM tested to see if it loads and properly associated with its emulator
- Full linux desktop environment when not in game kiosk mode via antimicro

R1 / C1+ Software Button Functions:

- 02 Second Hold: Software reset
- 04 Second Hold: Software shutdown
- 06 Second Hold: Turn off game kiosk mode
- 08 Second Hold: Change to 1024x768x32bpp resolution and reboot
- 10 Second Hold: Change to 720px32bpp resolution and reboot.

R2 / C2 Features:

- ODROID Goodness
- Custom Software Control Button
- Custom Hardware Reset
- Support for Atari 2600, Atari 7800, Atari Lynx, ColecoVision, MSX-1, MSX-2, NES, GameBoy, GameBoy Color, Virtual Boy, SNES, GameBoy Advance, WonderSwan Pocket/Color, NEO
- GEO Pocket/Color, Sega SG-1000, Sega Mark 3, Sega Master System, Sega Genesis, Sega GameGear, NEC Turbo Graphics 16, and NEC Super Graphics emulators configured and ready to go
- Retroarch with XBM, custom scripts to monitor the software button, start retroarch, maintain antimicro

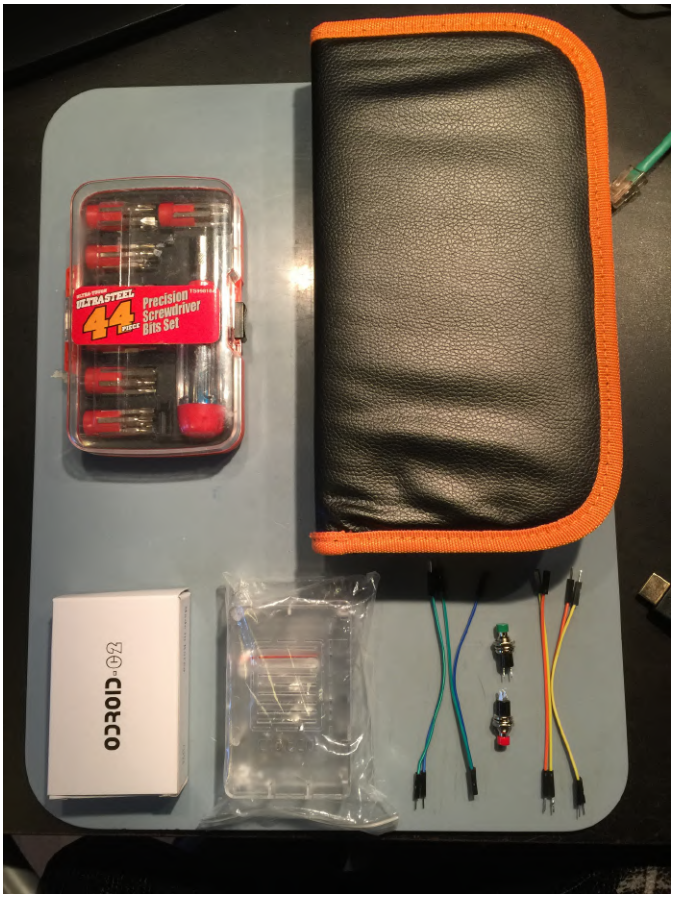
- Configured for low memory usage and for use with included controller
- Full linux desktop environment gamepad control when not in game kiosk mode via antimicro

R2 / C2 Software Button Functions:

- 02 Second Hold: Software reset
- 04 Second Hold: Software shutdown
- 06 Second Hold: Turn off game kiosk mode
- 08 Second Hold: Set video to auto for VGA mode, possibly alter retroarch.cfg for USB audio if present
- 10 Second Hold: Set video mode to 720p, alter retroarch.cfg for HDMI audio

3: The Setup

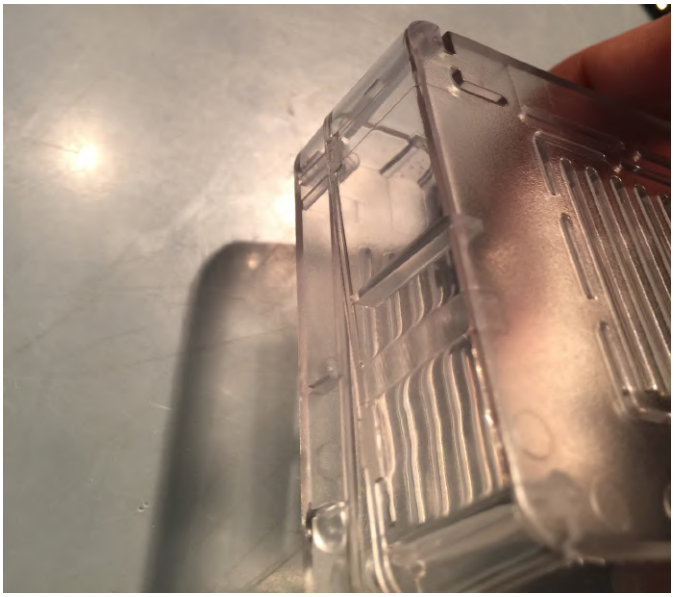
First things first - let's go over the tools and parts, lay them out, and get ready to build. We have an electronics screwdriver set. If you have built an ODROID-GO the same screwdriver set should work fine here. Notice we have our device, an ODROID-C2 is depicted below, this tutorial applies equally to the ODROID-C1+ or the ODROID-C2 version of this device. The ODROID-C1+ I use for 8-bit retro gaming the ODROID-C2 I like to use for all 8-bit and 16-bit systems plus the handhelds--it runs them wonderfully. You can probably run them on a C1+ but I like the C2 more for 16-bit games, for some reason. We have our custom control buttons, jumpers, case, and tools all ready to go.



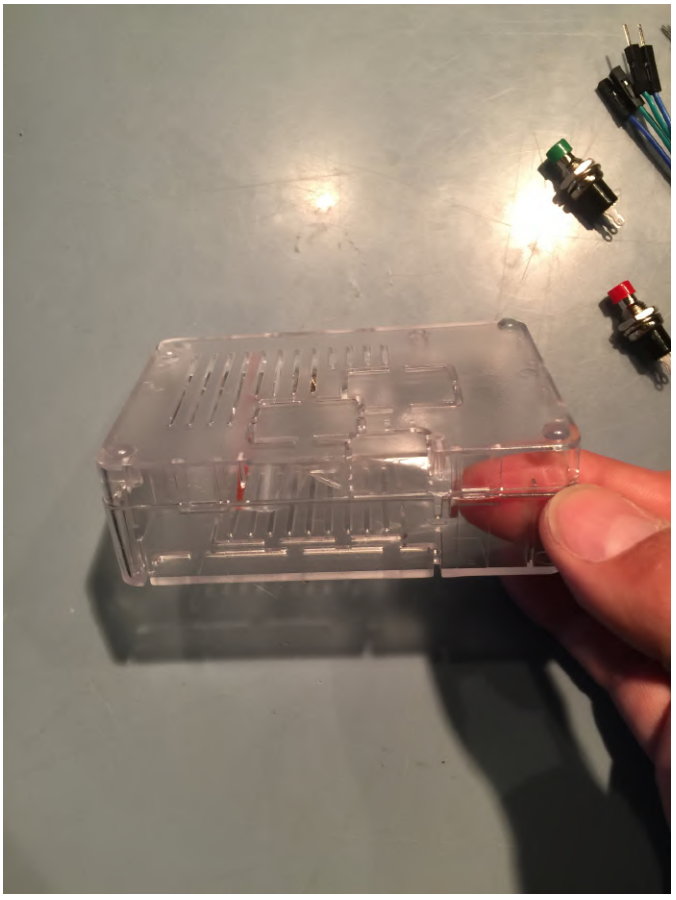
We have our soldering iron, notice the stand and sponge for cleaning the iron and temperature control on the iron. The temperature control is crucial, I think, but you can probably get away with not having one especially if you are good at soldering. The temperature control lets you adjust things so that you are applying high heat for a small amount of time. I have read a few tutorials and watched a few videos and from what I gather lower heat for a long time is not as good as high heat for a very short period of time. I do feel that it makes things cleaner, I am not sitting there trying to hold the iron to a connector waiting for a glob of solder to finally melt. One other thing is you really need is a fine tip on the iron. Again if you have skills maybe you can get by with a different tip, but I do not and a fine soldering tip was a life saver for me. Tip: If you do not have a fancy circuit board holding device to aid you in your soldering I have found that masking tape works really well. Ok, so it will melt, but it should not be getting hot. I will show you how to use the tape to secure what your soldering so you can get a good clean solder joint. It works just as good as the fancy circuit board holders, is super cheap, and has more

flexibility. If you really wanted, you could solder upside down, hanging from the ceiling if you use enough tape.

The case can be a little tricky to open. It should require little to no effort to separate it but it takes a little trick or two to do it that way. You could pop it open I suppose but broken fastening clips really irks me so I take the careful route. The first thing you want to do is unclasp the back of the case. This can be done by applying a slight pressure on the bottom of the case pushing it to the right, while pushing the top of the case to the left with a slight lift as shown below. Small forces here, it will not pop open, but it will separate.

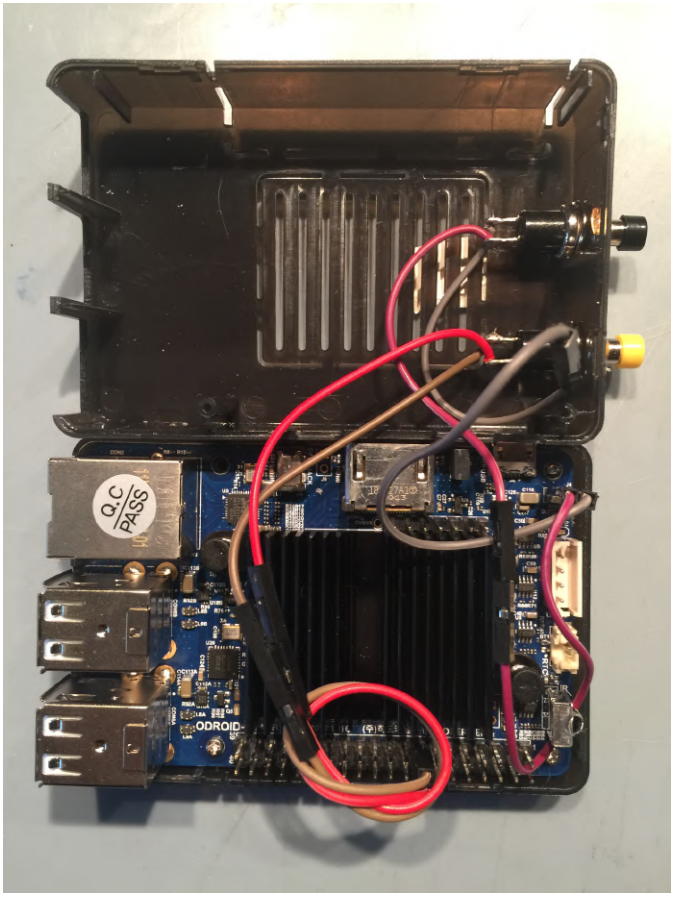


Once the back clip is separated we can move over to the side that has two clips. Tip: The next two clips on the side are a bit tricky. You can get the front one open by putting your finger inside the case and applying the same small forces we used to get the back clip open. To get the last clip on that side open you will need to use your fingernail. Slide the fingernail into the seam on the case at the back clip, the first one we separated, you can drag your fingernail around to the side that has the two clips and it will separate the remaining clip. It takes a little doing, but do not worry take your time. You may inadvertently reconnect a clip; that's OK, just start over. Main thing here is to have a non-broken case to house our beautiful console.



Now that we have the case open you can access the prize inside, a small bag of screws. Tip: If you have any left over ODROID-GO screws they are slightly bigger than the included screws and I find that they work better and are less prone to stripping. Now that the case is open we have to make a decision about the SD card door Normally the case will have no SD card access unless you pop out the SD card door. So, if your console is all closed up and your SD card dies you will have to very carefully cut open the SD card door or wrestle the case open. I like to open the SD card door for development models so I can easily pop in and out SD cards.

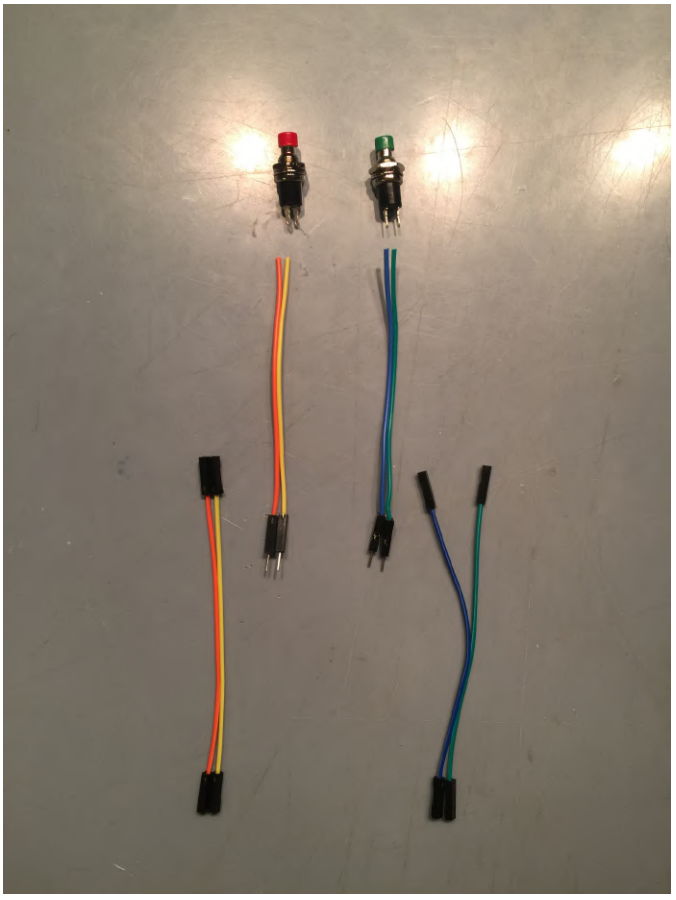
To cut out the SD card door you'll need a razor or small sharp knife. There is not much too it. You will need a few minutes, there is no super fast way to pop it out without risking damage to the case. Scratch away at the connection point with the razor. Drag it across the points one at a time scratching and cutting into the small connection plastic. Tip: You can place the bottom of the case standing up on its side, that way you can put a lot of downward pressure on two of the SD card door's plastic connections without risking damage to the case. An open SD card door is depicted below on a Monku Retro 1 (ODROID-C1+).

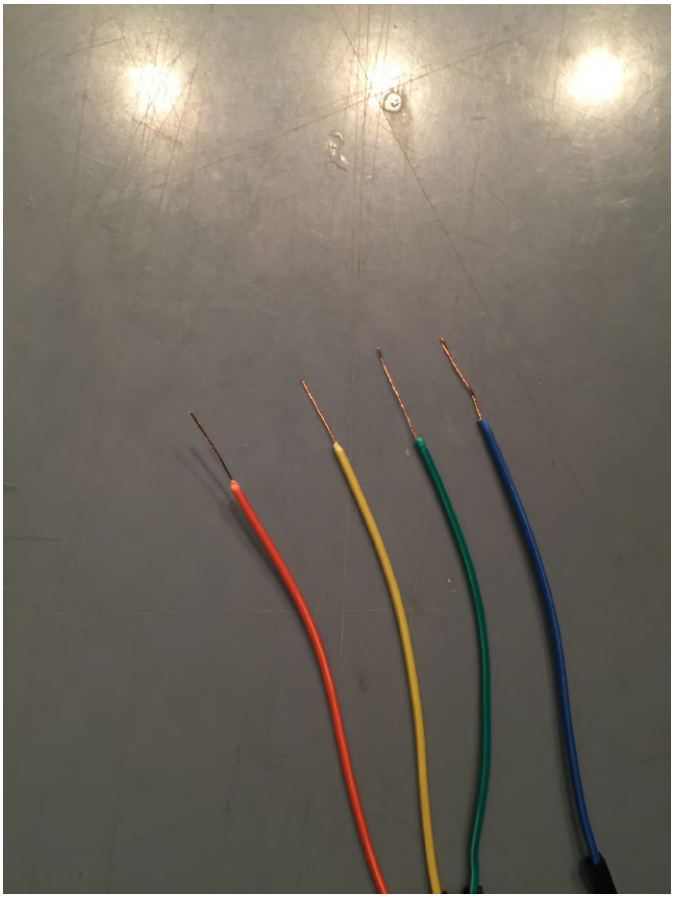


A quick aside, below is a photo of a completed Monku Retro 1 (ODROID-C1+) with custom control buttons and jumper connections. I like using 4 jumpers per button, I know you can get by with only two but I like having the extra connections.For this option, you'll need 4 female to female jumpers, and 4 anything to male connectors. There are other combinations just make sure you lay out the jumpers and envision their use. Make sure all the connections match up. You will have to cut off one side of the jumpers for the wires that are soldered to the buttons.

4: The Soldering

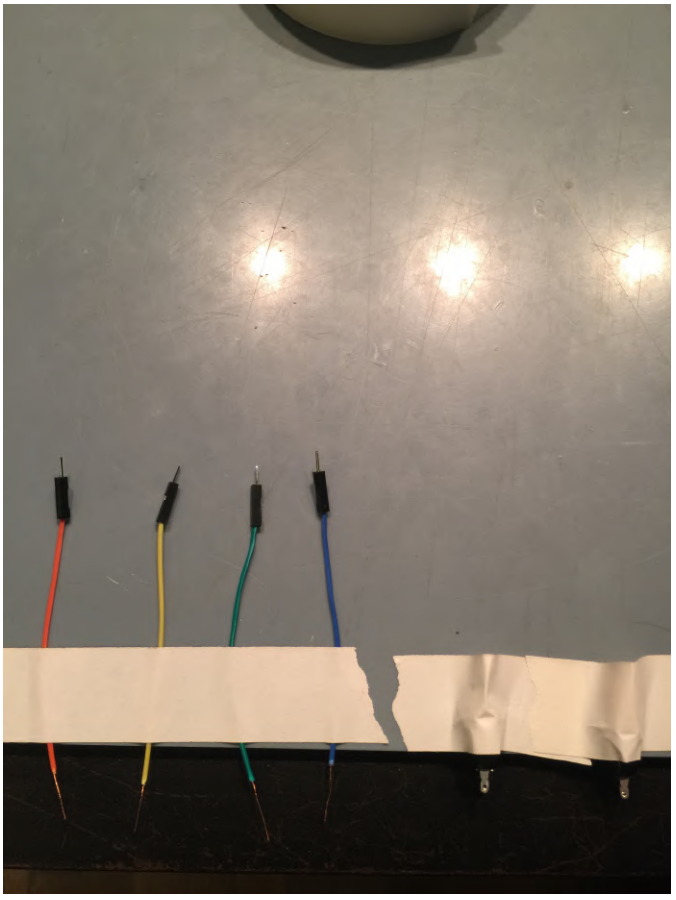
Ok so now we will be getting ready to solder the connection we need for the hardware reset, direct power toggle of the board. Now would be a good time to get your soldering iron. Plug it in and let it get all heated up.We will be using around 400 degrees, if you have temperature control, for the wires and buttons. We are going to layout the jumpers and buttons below to visualize them. I normally use jumper pins from the ODROID-GO expansion slot but I realize not everyone has those, so I used jumpers on this build. In this photograph, I used the wrong colors, oops.

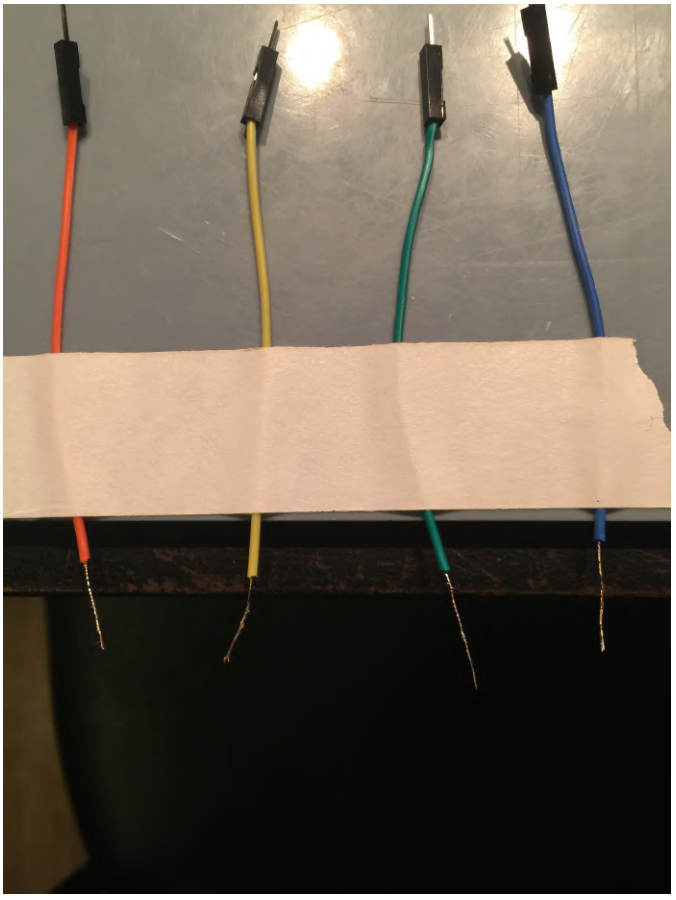




Let’s take the jumpers that are closer to the buttons as shown above. We then cut the connectors off the one side. Now you can use a scissors or your fingernail to remove the wire casing. Either gently roll the wire along one blade of your scissors with a little pressure until there is a break in the casing. Same strategy applies to using a fingernail. Once the casing gets a little white and shows signs of coming apart you can usually pinch and pull just the casing off of the wire. Once that is done hold the very tip of the copper wires and rotate the cable so that they are all nicely twisted up. Do this for the other jump cables as well.

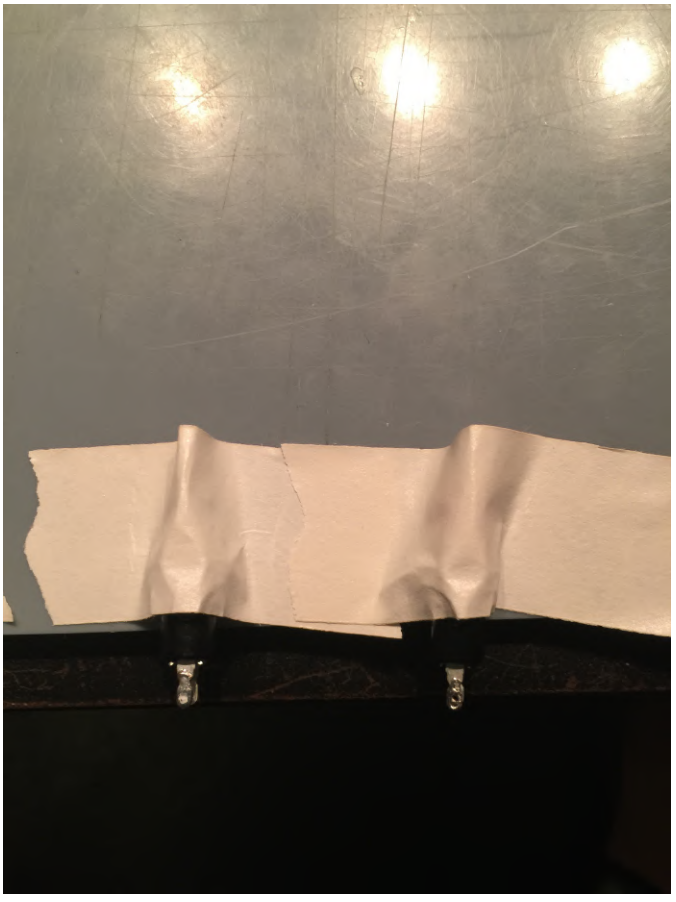
Make sure to check the status of your soldering iron. I accidentally started melting the plastic case on my screwdriver set once during a build. Lay out the four wires and two buttons. Get your iron ready, clean the tip with your wet sponge and tin the iron with a little solder and flux. I will assume you have solder with the flux inside it from this point on We are going to tin all the connections here before we connect them. Solder binds almost instantly to properly tinned connections. I do not tin the boards though, I follow a "as little time with the iron as possible" mantra with the board itself. Notice we are using the masking tape to secure each piece.



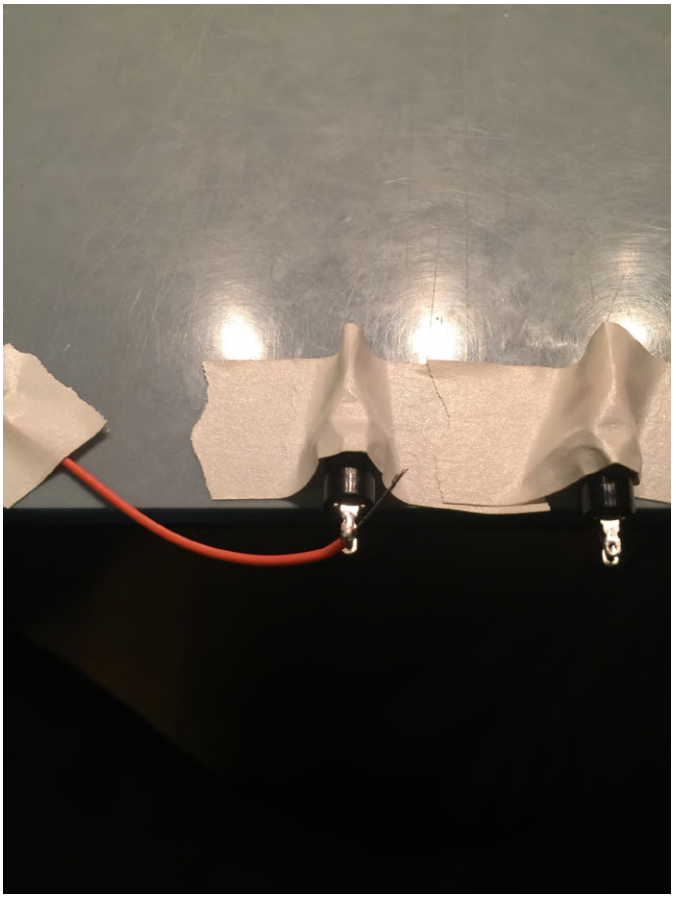


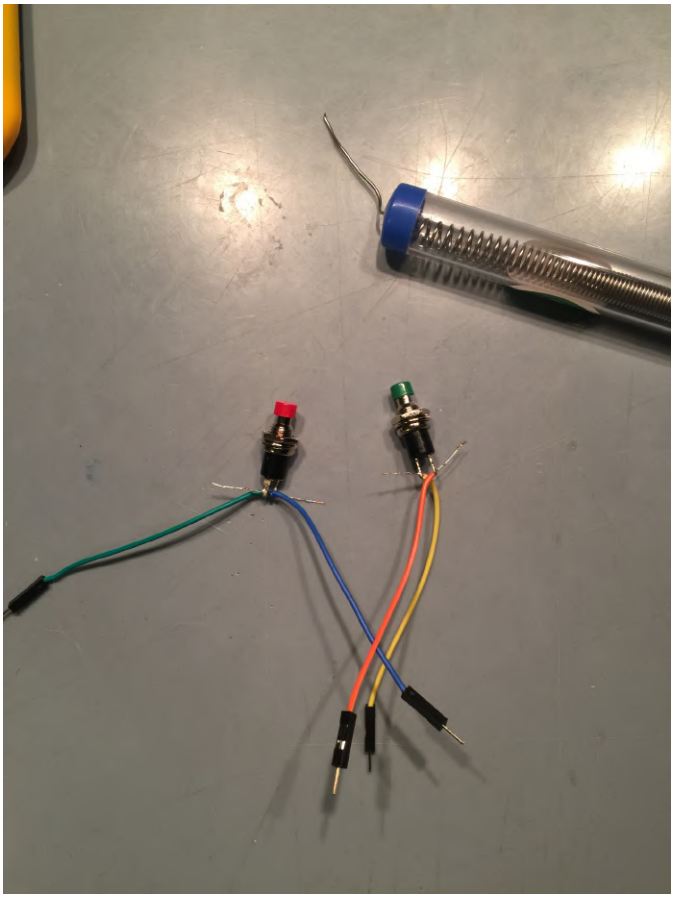
Hold the soldering iron underneath the wire and just dab the solder to it, the copper braid will absorb the solder and you can spread it by moving the iron down the wire. You do not need a lot, just a little on each one Do all four wires.

Now you want to tin the button contact points. Don't worry if you fill one of the holes on the contact point, it will not be a problem.

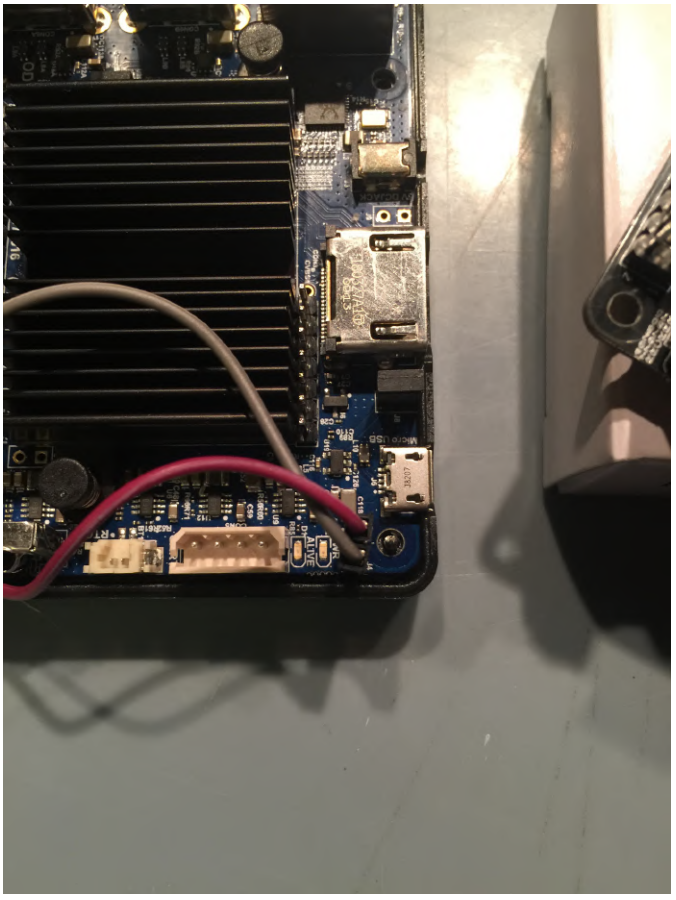


Place one of the wires that we stripped and tinned, keeping the colors you choose consistent throughout the construction, inside the contact hole of the button. You may have to apply the iron slightly to allow the wire to pass through. Bend and secure the wire to the side of the button with some tape. Now solder the contact point. Do this for both connections on both buttons. Use a scissors to snip off the excess wire. You should have something like what's shown below.

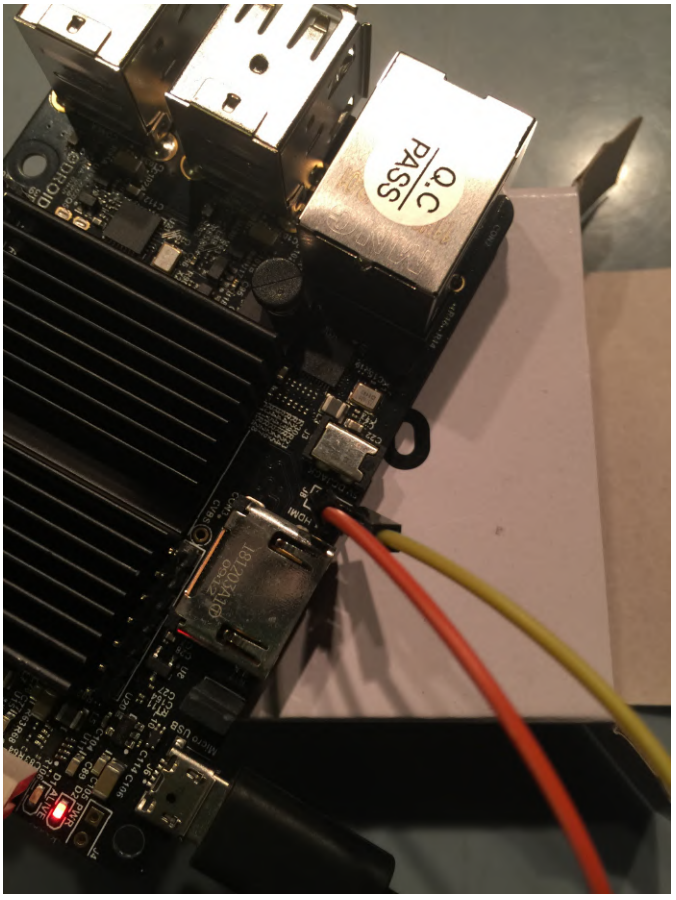




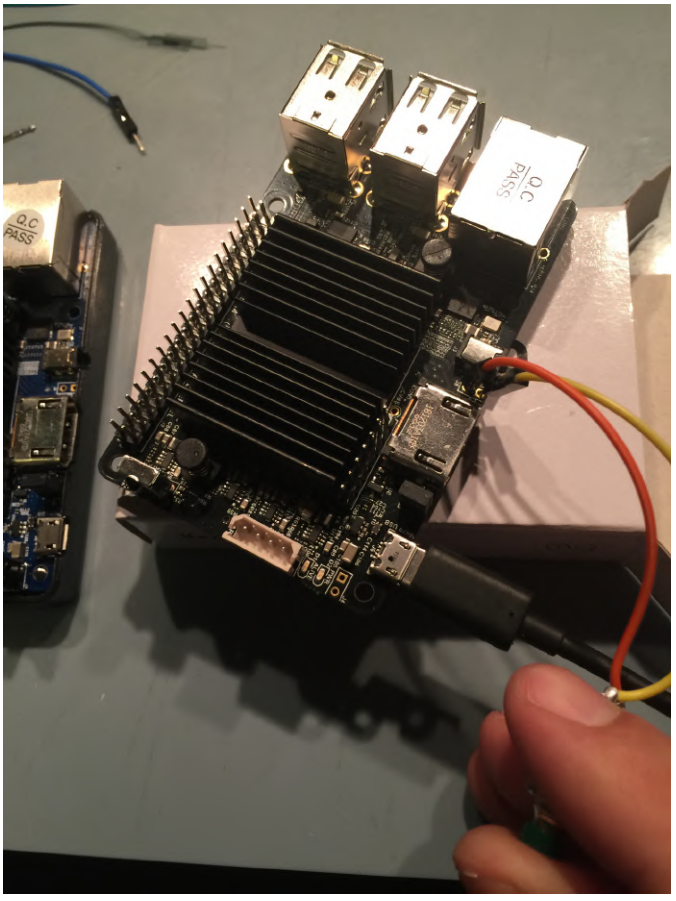
ALERT: There is a difference between the C1+ and the C2 so you have to stop at this point and double check which board you have and in the case of the ODROID-C2, which version of that board. Below is a photo of the C1+ with power button soldered on the board by the "pwr" text. The C2 board has the same marking but it may not be the actual power button location. You can use the buttons you just made to find out which location is the power button point. Place the male jumper leads into the holes in one of the available positions, J8 or pwr, give them a slight squeeze with your finger so they make contact. With the power on, but no SD card or anything in, push the button. If the red LED goes off you've found the power button. If nothing is happening you might have a bad button try the other one you made. If still nothing is happening you might have a bad contact try and short the two little holes at the available locations until you identify which one controls the power.

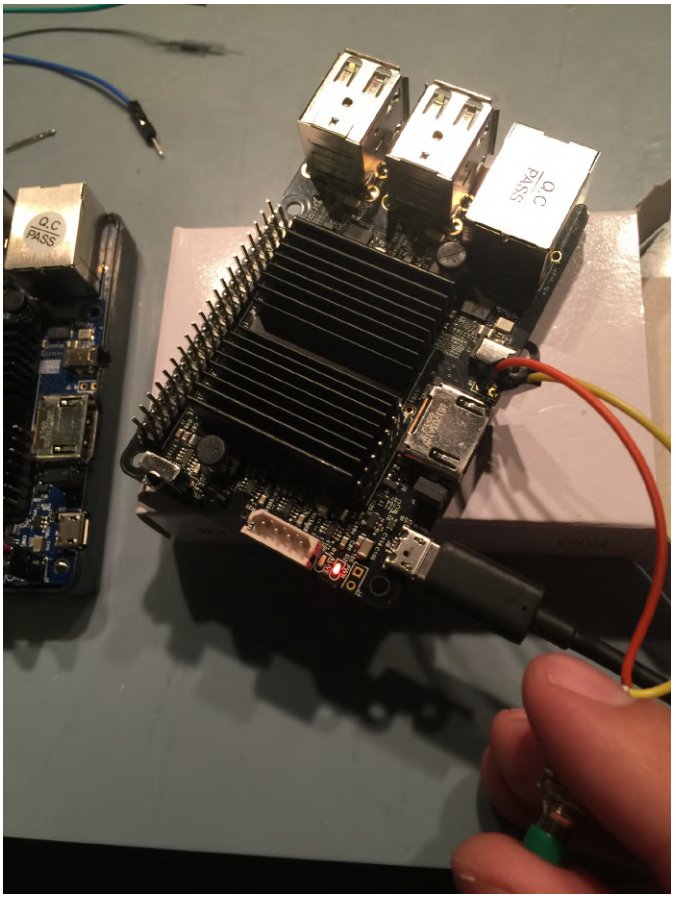


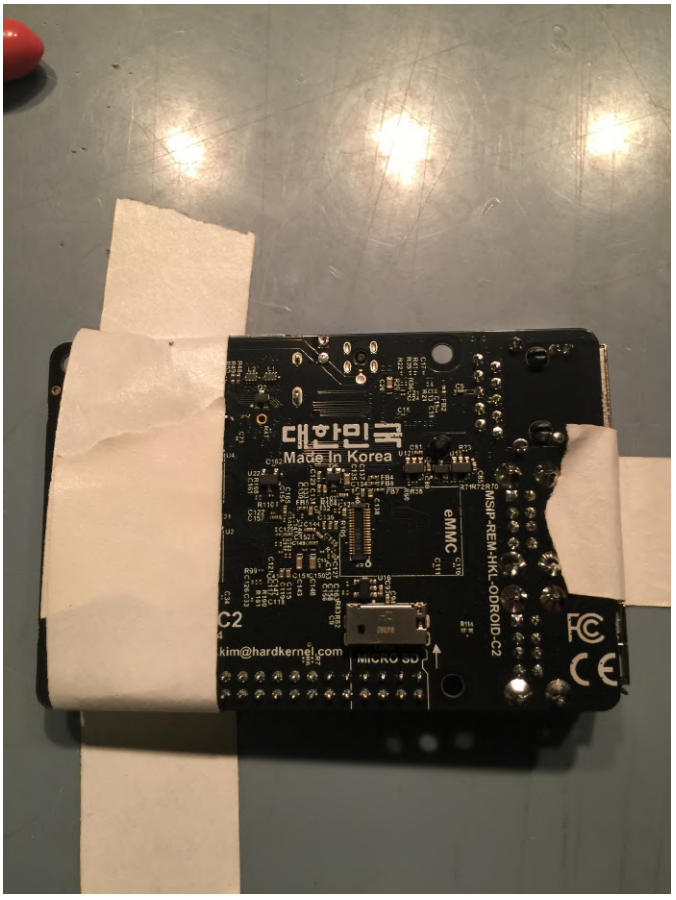
NOTE: If you are working with an ODROID-C1+ this is the place on the board you will be adding the jumpers for the power switch. Make sure you have verified it with the process outlined above.



The photos below show the button test in action.

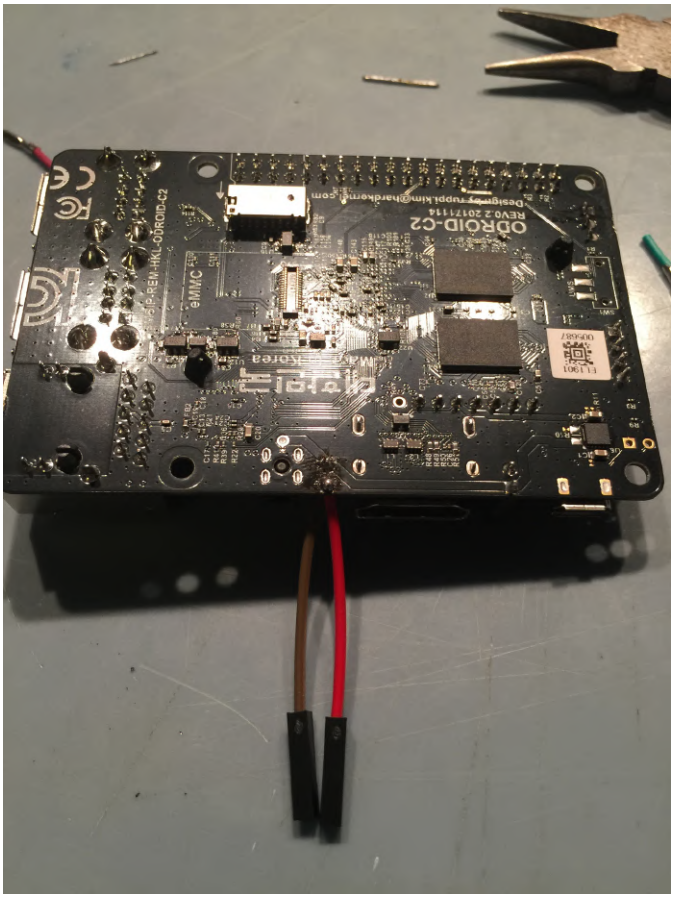


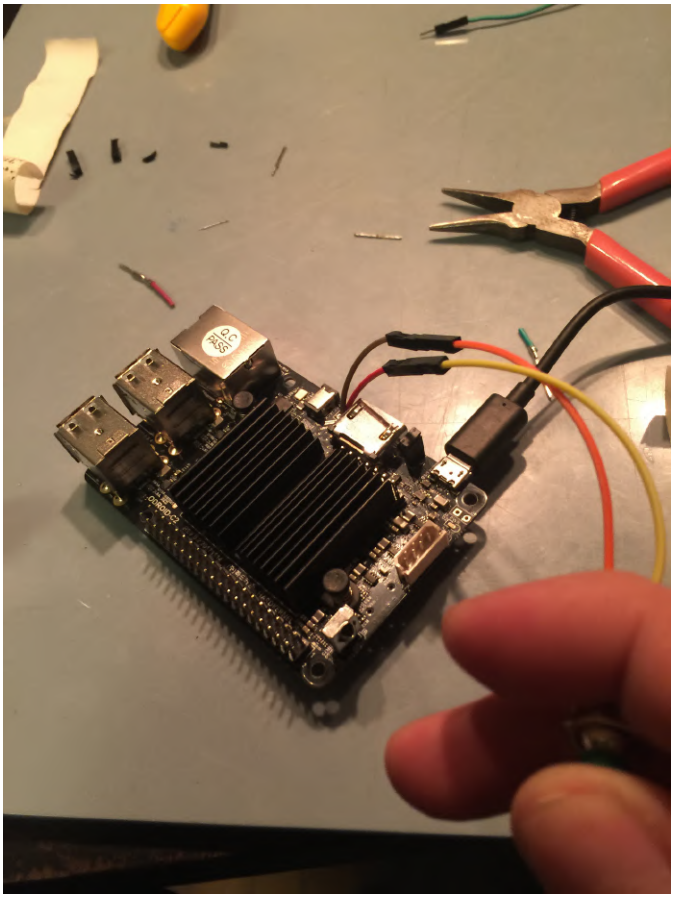




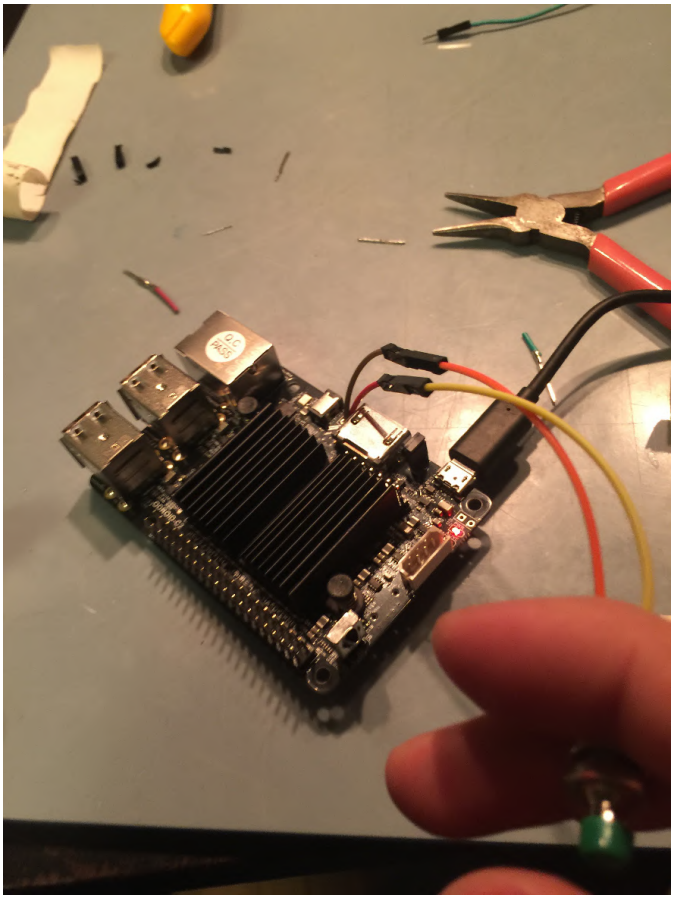
The next step is to solder the board. Turn your soldering iron heat up to 450 or so. Secure the board to your work surface, make sure you have it steady such that it won't move when the soldering iron touches it. I normally would use pins from the ODROID-GO expansion clip but not everyone has one lying around so instead of plugging jumpers into the pins we'll solder the jumper to the board and plug into the other side of them.

Strip and twist the wires like we did for the buttons. Start with the hole furthest away on the board, push the wire through and solder it on, little bit of solder, little bit of time. Repeat for the hole closer to the edge. At this point, you should have something like pictured below. For the C1+ this connection - near the corner of the board - is a bit tight, just be careful, take your time.



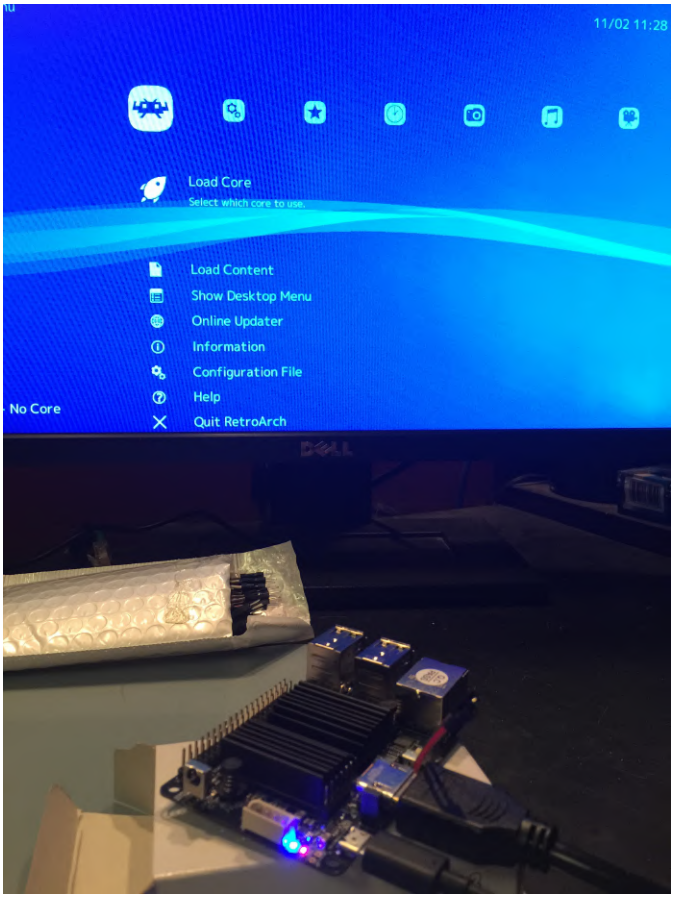


Congratulations, we are done soldering. Let's test it. Plug in one of your working buttons. Power up the unit without an SD card. You should be able to get the red LED to toggle just like before.



If you have a working C1+ / C2 SD card now is a good time to test and see that everything is working fine. Looking good. If you do not have an SD card, no

worries, we'll be at the software step soon. The only thing left on this side of things is the case holes and mounting the buttons.



In the next installment of this tutorial, we will look at building a case for the console.

References

http://middlemind.com/tutorials/odroid_go/mr1_build.html

I aRe GO: Qwiic-ly Add IR Vision to Your ODROID-GO

June 1, 2019 By Dave Prochnow Development, ODROID-GO

Remember the SparkFun Electronics (SFE) I2C Qwiic Adapter project that we built in the **May 2019 issue of ODROID Magazine**? Oh, you don't recall reading that article? Okay, I'll wait here while you go back and refresh your recollection. Now that you're all up to speed on the Qwiic Adapter article, here's another Qwiic project that will give your ODROID-GO the gift of infrared (IR) vision.

This particular type of IR vision is supplied by a Panasonic AMG8833 IR array sensor (see Figure 1). Inside this sensor is an 8x8 thermopile "grid" array. What exactly is a thermopile? Well, it's a fancy word for a pile of thermos. No, seriously, a thermopile is a device housing a series of thermocouples, each of which measures temperature via a pair of dissimilar metal wires. In turn, these wires measure the difference in potential that is created at the wires' junction. Whew! You know what? I like the 'pile of thermos' definition better.

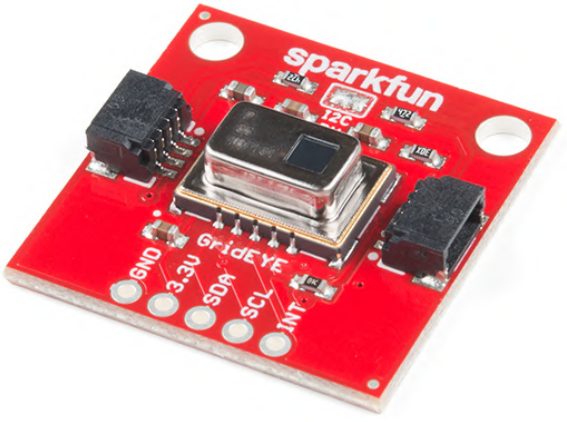


Figure 1 - The SparkFun Grid_EYE Qwiic Sensor board. Photograph courtesy of SparkFun Electronics

Granted, an 8x8 visual array is very low resolution, but objects placed 200-300mm in front of the Panasonic sensor can provide a colorful visual treat on the

ODROID-GO LCD. By adding the Grid-EYE to the Qwiic system, SFE have made IR vision a simple plug-and-go project for the ODROID-GO.

Step-by-Step

1. Insert the Qwiic Adapter for ODROID-GO PCB into the general purpose input/output (GPIO) female header located along the top edge of the ODROID-GO. If you don't already have this adapter PCB, you can learn how to make one in the May 2019 issue of ODROID Magazine.

2. Plug a Qwiic cable (e.g., 100 mm; SFE PRT-14427) into the Qwiic adaptor board.

3. Attach the SparkFun Grid-EYE infrared array breakout – AMG8833 (Qwiic) (SFE SEN-14607) into the other end of the cable. An ODROID-GO is ready to Qwiic-ly scan for heat signatures in Figure 2.

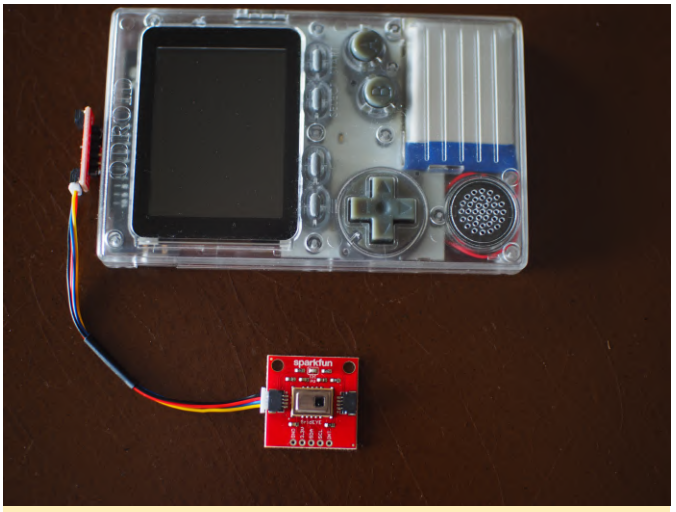


Figure 2 - All hooked up and looking for some hot stuff

4. Enter this super-short Arduino sketch (refer to Figure 3 – Figure 3c) into the Arduino Integrated Development Environment (IDE) and compile/load the sketch onto your ODROID-GO. Remember to create your own Display.h library subset (as described in the Qwiic Adapter project) from the ODROID-GO Master Library. Otherwise, the sketch will not compile inside the Arduino IDE.

```
/*
  Using the Panasonic Grid-EYE Sensor
  By: Nick Poole
  SparkFun Electronics
  Date: January 12th, 2018

  MIT License: Permission is hereby granted, free of charge, to any person obtaining a copy of this
  software and associated documentation files (the "Software"), to deal in the Software without
  restriction, including without limitation the rights to use, copy, modify, merge, publish,
  distribute, sublicense, and/or sell copies of the Software:
  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING
  BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
  NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
  DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
  OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

  Feel like supporting our work? Buy a board from SparkFun!
  https://www.sparkfun.com/products/14568

  Hardware Connections:
  Attach the Qwiic Grid-EYE to the Qwiic Adapter board mounted on your ODROID-GO
  Display on the ODROID-GO @ 320x240
*/

#include <SparkFun_GridEYE_Arduino_Library.h>
#include <Wire.h>
#include <Display.h>

// Use these values (in degrees C) to adjust the contrast
#define HOT 40
#define COLD 20

// This table can be of type int because we map the pixel
// temperature to 0-3. Temperatures are reported by the
// library as floats
```

Figure 3 - Include these three libraries at the beginning of your sketch

```
// library as floats
int pixelTable[64];

GridEYE grideye;

ILI9341 lcd = ILI9341();

void setup() {

  // Start your preferred I2C object
  Wire.begin();
  // Library assumes "Wire" for I2C but you can pass something else with begin() if you like
  grideye.begin();
  // Pour a bowl of serial
  Serial.begin(115200);

  // Setup LCD
  lcd.begin();
  lcd.setRotation(1);
  lcd.fillScreen(BLACK);
  lcd.setBrightness(255);
  lcd.setTextFont(1);
  lcd.setTextSize(2);
  lcd.setCharCursor(16, 3);
  lcd.setTextColor(LIGHTGREY);
  lcd.println("ODROID-GO");
  lcd.setCharCursor(17, 6);
  lcd.println("Grid-EYE");
  lcd.setTextSize(3);

}

void loop() {
```

Figure 3a - Setup the ODROID-GO LCD for displaying your IR data

```
void loop() {

  // Initiate the text cursor position
  lcd.setCharCursor(0, 0);

  // loop through all 64 pixels on the device and map each float value to a number
  // between 0 and 3 using the HOT and COLD values we set at the top of the sketch
  for(unsigned char i = 0; i < 64; i++){
    pixelTable[i] = map(grideye.getPixelTemperature(i), COLD, HOT, 0, 3);
  }

  // loop through the table of mapped values and print a character corresponding to each
  // pixel's temperature. Add a space between each. Start a new line every 8 in order to
  // create an 8x8 grid
  for(unsigned char i = 0; i < 64; i++){
    if(pixelTable[i]==0){
      Serial.print(".");
      lcd.setTextColor(BLACK, BLACK);
      lcd.print("*");}
    else if(pixelTable[i]==1){
      Serial.print("O");
      lcd.setTextColor(BLUE, BLACK);
      lcd.print("*");}
    else if(pixelTable[i]==2){
      Serial.print("O");
      lcd.setTextColor(ORANGE, BLACK);
      lcd.print("*");}
    else if(pixelTable[i]==3){
      Serial.print("O");
      lcd.setTextColor(RED, BLACK);
      lcd.print("*");}
    Serial.print(" ");
    if((i+1)%8==0){
      Serial.println();
```

Figure 3b - Mapping thermal colors to the LCD and Serial port

```
      Serial.println();
      lcd.println();
    }
  }

  // in between updates, throw a few linefeeds to visually separate the grids. If you're using
  // a serial terminal outside the Arduino IDE, you can replace these linefeeds with a clearscreen
  // command
  Serial.println();
  Serial.println();


  // toss in a delay because we don't need to run all out
  delay(100);

}
```

Figure 3c - Finally, add a short delay to the screen and Serial port updates

Disconnect the ODROID-GO, turn it on, and go look for some hot stuff to measure. A hot mug of coffee is an ideal test subject (see Figure 4). Your new IR vision on the ODROID-GO is now ready to show you things in your life that are too hot to handle (see Figure 5).

Figure 4 - Here's a handy setup for using your ODROID-GO to measure IR heat

Figure 5 - A display from the same ODROID-GO setup shown in Figure 4

The G Spot: Your Go-to Destination for all Things Android Gaming - Google Stadia

June 1, 2019 By Dave Prochnow Android, Gaming, ODROID-C0, ODROID-C1+, ODROID-XU4

I would be remiss if I didn't at least mention Google Stadia. This is a streaming game service that promises to breathe new life into your ODROID-XU4 gaming experience—even the lower-powered ODROID-C1 should, in theory, benefit from this Google service.

What the heck is Google Stadia? Simply put, as announced at Google Developer Conference 2019, Google Stadia is a cloud-based streaming gaming system that will enable anyone to play any game on any device at any time. Likewise, when this system is used in conjunction with Google's YouTube network, you'll be able to go from watching a game trailer to playing that game in just a matter of minutes.

Aimed at big-budget AAA games, Google Stadia promises to revolutionize today's gaming industry. In fact, during an April Alphabet financial earnings telephone conference, Google boss Sundar Pichai stated that big-name game producers have "...genuine excitement, because I think they see the opportunity for a shift, a point of inflection, but they realized the technical challenge of pulling something like this off. But once they get their hands on the technology and then they see the experience, I think that really wins people over."

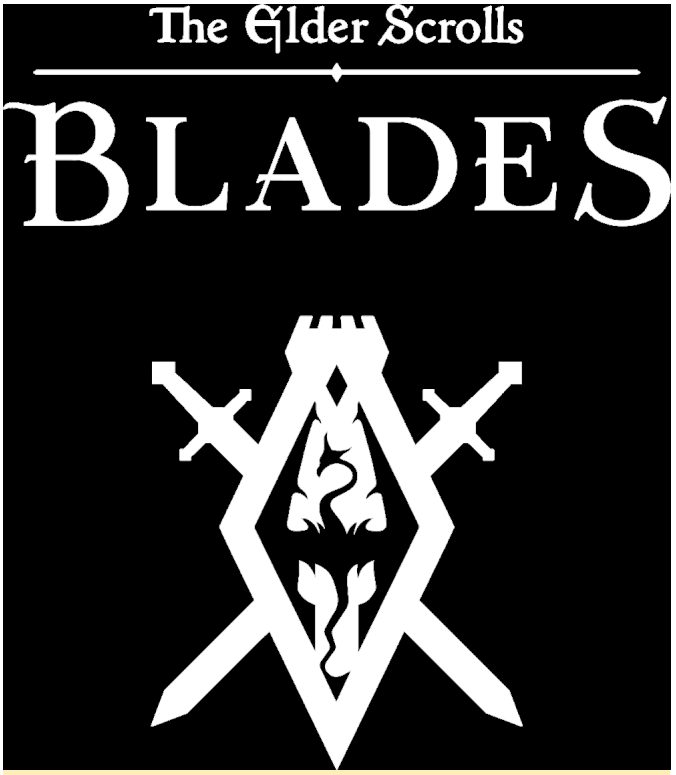


Figure 1 - Google Stadia Logo

On paper, the promise of Stadia sounds almost too good to be true: game where you want, when you want; play instantly—no software updates, drivers, or installers; game in 4K HDR at 60 frames per second (fps); and there is a built-in upgrade roadmap that will be maintained by Google. However, there is one gigantic gotcha (and you knew that there would be one, didn't you?). In order to utilize Stadia on your device(s), you will need a high-bandwidth, rock-solid Internet connection. And for many users, good luck with seeing that kind of performance throughout game play.

You can watch the Google Stadia infomercial, aptly named 'The Future of Gaming from Google,' here:

During this Alphabet financial conference call, neither the cost nor the starting date were revealed. According to Mr. Pichai, "With Stadia, you will be able to play advanced AAA games on any type of screen instantly, without ever needing to download the game or install updates. The reception from gamers in the industry has been incredible and we look forward to sharing more when it launches later this year."

This vagueness was addressed more directly by Google VP Phil Harrison, stating that "two more events" would be scheduled by Google in order to provide all of the "meat and potatoes" info for gamers prior to the launch of Stadia.

That's (Not) All Folks!

Are you a Cartoon Network fan? Then the free Cartoon Network Arcade game has got you covered. Crammed full of games that are populated with your fave Cartoon Network characters, there is also a video tie-in between the game and the TV network for grabbing Cartoon Network characters. It's a win-win for satisfying your cartoon habit 24/7.

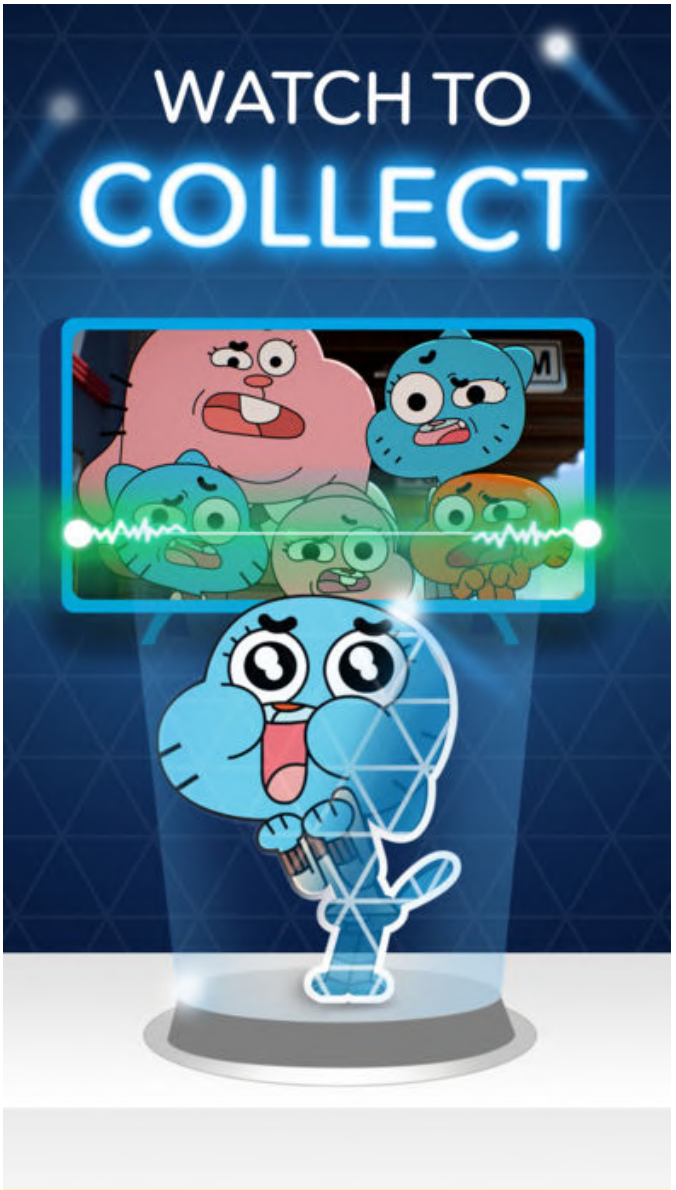


Figure 2 - Play as your favorite characters on Cartoon Network Arcade

https://youtu.be/l6w6P0-Sdo4

Update: Elder Scrolls: Blades (Figure 3 - Elder Scrolls: Blades is a step closer to becoming reality)

As we mentioned last month, the official release of Elder Scrolls: Blades from Bethesda is getting closer to becoming a reality. This month we all received an Easter present: an early-access Beta release that can now be played by anyone. There's just the trivial matter of signing up for early access, then you'll be catapulted into battle. Just follow this link to learn more about this landmark game:

https://elderscrolls.bethesda.net/en/blades

Going Downhill Fast

Hey Linux gamers; remember Tux Racer, a game featuring that adorable penguin skiing pell-mell downhill looking for fish? Well, Android gamers can now share the excitement of traversing down snow-covered mountains with the cleverly named Grand Mountain Adventure. Launched by Toppluva AB in late March, Grand Mountain Adventure combines fun downhill challenges along with mountain skiing exploration. A perfect game to play while sitting next to the pool.

Watch this chilly trailer:

https://youtu.be/n-mr4Z5oGWc

Do You Want to Play an Android Game? ODROID Magazine's May Top Ten 10. The War of Mine–$14.99/developer expansion pack $1.99 9. Asphalt 9: Legends–FREE 8. Minecraft–$6.99 7. Crashlands–$4.99 6. Fire Emblem Heroes–FREE 5. Riptide GTP Series–$2.99 4. Shadowgun Legends–FREE 3. Elder Scrolls: Blades–FREE; this beta rockets to the top 2. Fortnite–FREE 1. PUBG Mobile–FREE

Custom Boot Logo: Creating a custom boot image for the ODROID-N2

June 1, 2019 By Justin Lee ODROID-N2, Tinkering, Tutorial

You can create a custom boot logo for the ODROID-N2 by following the simple instructions below. The logo will appear in the first few seconds of booting while the operating system loads. The basic image format of the ODROID-N2 boot logo file is described in the next section.

Format

```
Image Format : 24-bit Windows BMP image or 24-bit Windows Gzipped BMP image (without meta-data)
Image Size : 1280 by 720
Color Depth : 24bpp Color
File Name : 'boot-logo.bmp' or 'boot-logo.bmp.gz'
```

The file name should be 'boot-logo.bmp' or 'boot-logo.bmp.gz'. A sample bmp file can be downloaded from **https://wiki.odroid.com/_media/en/boot-logo.bmp.gz**. We recommend using GIMP or KolourPaint.

GIMP - Export as Windows BMP - Compatibility Options : Do not write Color Space Information - Advanced Options : 24 bits Color - Name : "boot-logo.bmp"

KolourPaint - Save Image as - Filter : Windows BMP image - Convert to : 24-bit Color

Size Limitation

You must keep the size of your logo file under 2MB because, the logo partition of Android is limited to 2MB. Gzip BMP format is supported, so if the size is over 2MB, you can use a bmp.gz file.

```
$ gzip boot-logo.bmp
$ ls boot-logo.bmp.gz
```

Auto scaling option

On ODROID-N2 uboot, image scaling for a boot logo is supported, so that a displayed boot logo will be fixed

automatically for output mode as described in boot.ini. For example, in case your using the “1024x600p60hz” mode, the boot logo will be displayed as 1024×600 even though the actual size of bmp file is 1280×720.

Replacing the boot logo

ODROID-N2 scans the existence of the following three parts in numerical order. boot-logo.bmp in VFAT partition boot-logo.bmp.gz in VFAT partition logo data in Android LOGO partition

Android

On Android, you can replace the boot logo with your own custom image. There are two methods to change the boot logo image: Add a image into VFAT partition. Rewrite image data into Android LOGO partition using fastboot.

Method 1. VFAT Copy the new boot-logo.bmp, or boot-logo.bmp.gz, to the VFAT partition.

Method 2. Android Logo Partition First, you must get into your U-Boot command line while pressing ENTER key when your ODROID-N2 is powered up. Execute the fastboot command from U-Boot and connect with your desktop using USB cable.

```
odroidn2# fastboot
```

Next, run fastboot command from your desktop.

HOST PC

```
$ fastboot flash logo boot-logo.bmp.gz
```

or

```
$ fastboot flash logo boot-logo.bmp
```

If you will use bmp data on the logo partition, make sure there is NO boot-logo.bmp.gz file in your VFAT area, because U-Boot first checks if there is a boot-logo.bmp/boot-logo.bmp.gz in the VFAT area and then checks the logo partition.

Ubuntu

With Ubuntu, a LOGO option is NOT included by default. So, you will need to add a boot logo image into the VFAT partition. A method using a LOGO partition is not available on Ubuntu.

Adding the showlogo command: 1080p60hz case

On U-Boot, the default logo display logic works with 1080p60hz display resolution. So you don't need to add or modify any related commands. Just make sure, however, that the boot logo file exists in the aforementioned locations.

Resolutions other than 1080p60hz

You should add the commands to your boot.ini before bootcmd is executed. Please check if there is a 'showlogo' command in your boot.ini first. If not so, refer to the following:

```
### Boot Arguments
if test "${display_autodetect}" = "true"; then
hdmitx edid; fi
if test "${hdmimode}" = "custombuilt"; then setenv
cmode "modeline=${modeline}"; fi

### Add showlogo with ${hdmimode} size
hdmitx mode ${voutmode}
showlogo ${hdmimode}
```

How to use the custom image with Native resolution

If you want to use a native resolution of the bmp image, like 1920×1080, 1024×600 (for VU7+) or 800×480 (for VU7), set arg[2]/arg[3] of the showlogo command as follows:

```
odroidn2 # help showlogo
showlogo - Displaying BMP logo file to HDMI screen
with the specified resolution

Usage:
showlogo  [ ]
        resolution - screen resoltuion on HDMI
screen
                '1080p60hz' will be used by
default if missing
        bmp_width (optional) - width of logo bmp
file
                '1280' will be used by default if
missing
        bmp_height (optional) - height of logo bmp
file
                '720' will be used by default if
missing
```

Replace the boot logo image with yours as described in the previous sections and then modify 'showlogo' command in boot.ini. Here are examples:

Logo image size of width 1920 and height 1080

If your monitor's resolution is 1920×1080 and you want to set a bmp file in 1920×1080, set command in boot.ini as follows:

```
setenv hdmimode "1080p60hz"
showlogo ${hdmimode} 1920 1080
```

Logo image size of width 1024 and height 600

```
setenv hdmimode "1024x600p60hz"
showlogo ${hdmimode} 1024 600
```

Logo image size of width 800 and height 480

```
setenv hdmimode "800x480p60hz"
showlogo ${hdmimode} 800 480
```

Android Boot Animation

In case of Android, you can use the bootanimation.zip method to show your custom logo using animation. Please refer to this reference site: **https://android.googlesource.com/platform/frameworks/base/+/master/cmds/bootanimation/FORMAT.md** The system selects a boot animation zip file from the following locations.

```
/system/media/bootanimation.zip
/oem/media/bootanimation.zip
```

Before the copy process, you need to change the root filesystem permission to r/w and copy your bootanimation.zip into /system/media/ folder.

```
console:/ $ su
console:/ # mount -o rw,remount /system
[  173.674067@2] EXT4-fs (mmcblk0p11): re-mounted.
Opts:
block_validity,delalloc,barrier,user_xattr,acl,inode_readahead_blks=8
```

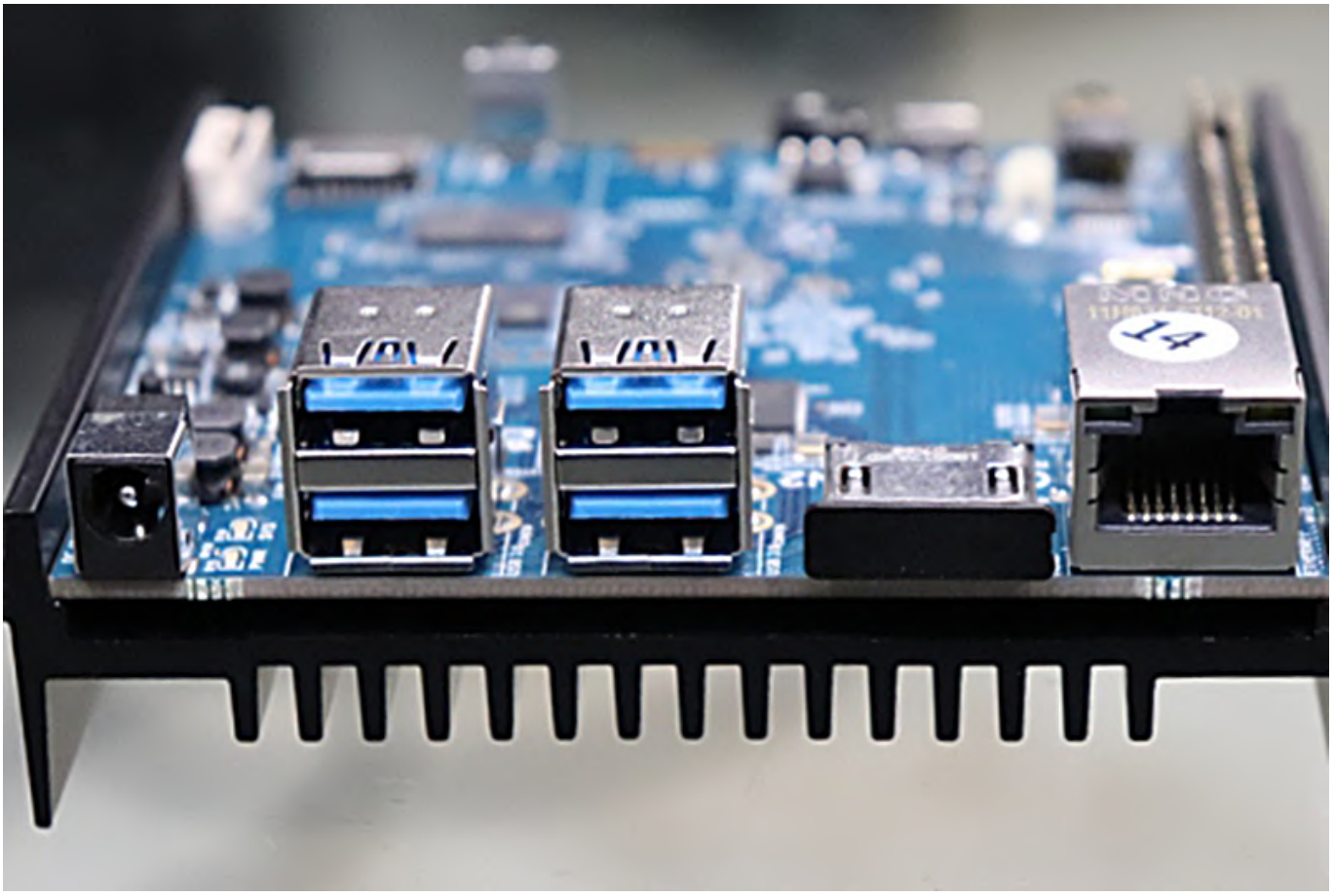
Please make sure of the correct file permission for bootanimation.zip:

```
console:/ # chmod 755
/system/media/bootanimation.zip
console:/ # ls -al /system/media/bootanimation.zip
-rwxr-xr-x 1 root root 5030729 2019-03-27 00:41
/system/media/bootanimation.zip
```

This guide is available on the ODROID wiki at **https://wiki.odroid.com/odroid-n2/application_note/bootlogo_n2**.

ODROID-N2 Review: Good Performance in Linux Benchmarks

June 1, 2019 By Michael Larabel from Phoronix.com ODROID-N2



Hardkernel's newest single board computer is the ODROID-N2, which they sent over to me a few weeks ago for benchmarking. The ODROID-N2 is built around the Amlogic S922X SoC and features four Cortex-A73 cores and two Cortex-A53 cores, options for 2GB or 4GB of DDR4 system memory, eMMC connectivity, Gigabit Ethernet, and four USB 3.0 ports for starting out just above $60 USD.

Figure 1 - top view of ODROID-N2

The ODROID-N2's use of an Amlogic S922X big.LITTLE design makes for an interesting arrangement with the four Cortex-A73 cores clocking up to 1.8GHz and the two Cortex-A53 cores able to hit 1.9GHz. This SoC uses the Mali G52 Bifrost GPU, which eventually should see nice driver support via the open-source Panfrost graphics driver stack.

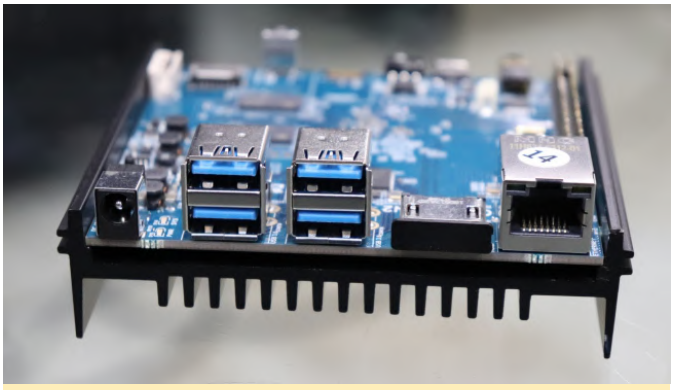


Figure 2 - edge view of ODROID-N2

The ODROID-N2 employs DDR4 memory which offers much greater performance potential than the ODROID-N1 and others relying upon DDR3 system

memory. The aluminum heatsink on the bottom of the SBC is designed for ensuring sufficient cooling potential of the SoC and RAM. The SoC being placed on the bottom of the PCB is a bit of an usual design feature when compared to most budget ARM SBCs that we see, but it works out well with the metal heatsink serving as a nice base.

Figure 3 - view of the bottom and heatsink

Connectivity for the ODROID-N2 includes HDMI 2.1, Gigabit Ethernet, four USB 3.0 host ports, micro-USB 2.0 OTG port, 40-pin GPIO header, and a DC power jack.

Figure 4 - top view of board

The ODROID-N2 officially supports Ubuntu 18.04 LTS and Android 9 Pie BSPs at this time though it's likely more Linux distributions will be supporting the N2 as the year progresses. The Ubuntu 18.04 LTS AArch64 image for the ODROID-N2 is using a Linux 4.9-based kernel.

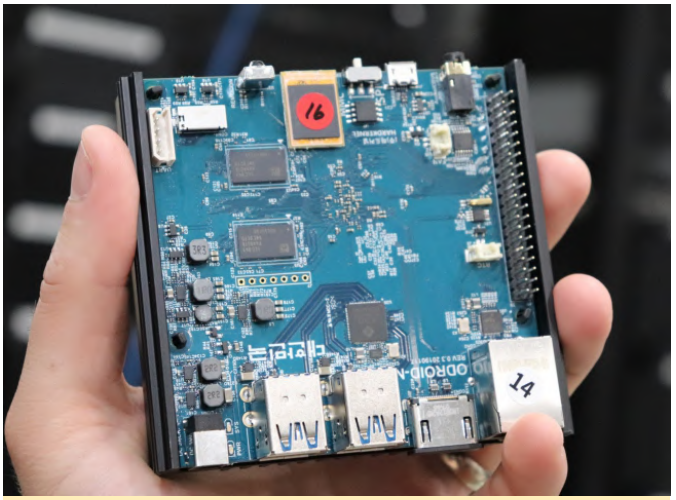


Figure 5 - top top view with emm card installed

The ODROID-N2 with 2GB of RAM is priced at around $65 USD while the 4GB version is positioned at just above $80 USD. Hardkernel kindly sent over the ODROID-N2 for benchmarking on Phoronix.

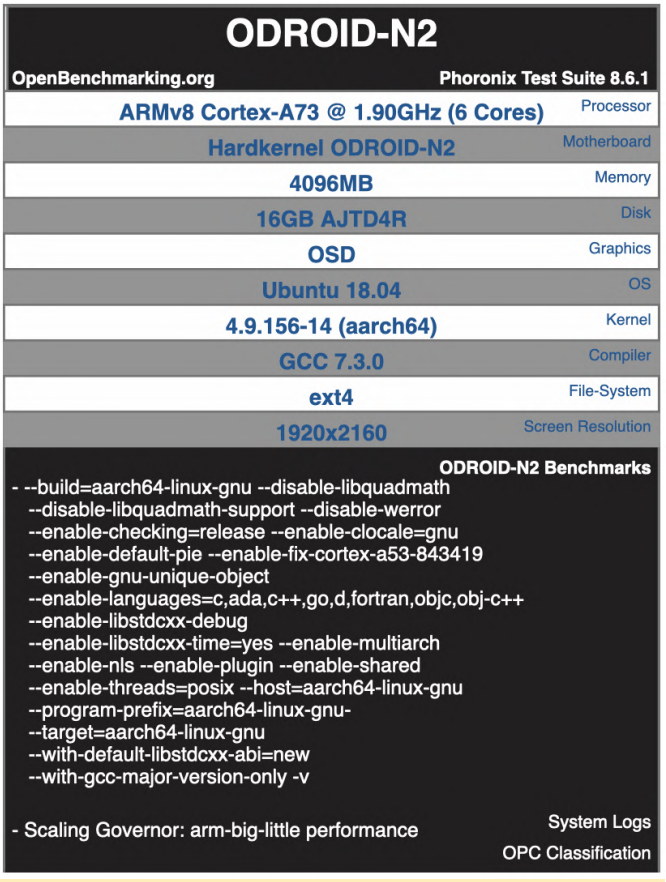


Figure 6 - specification overview

For getting an idea for the performance potential of the ODROID-N2 4GB, here are some initial benchmarks comparing it to the NVIDIA Jetson TX1, Jetson TX2, Jetson AGX Xavier, Jetson Nano, Raspberry Pi 3 Model B+, ASUS TinkerBoard, ODROID-C2, and ODROID XU-4.

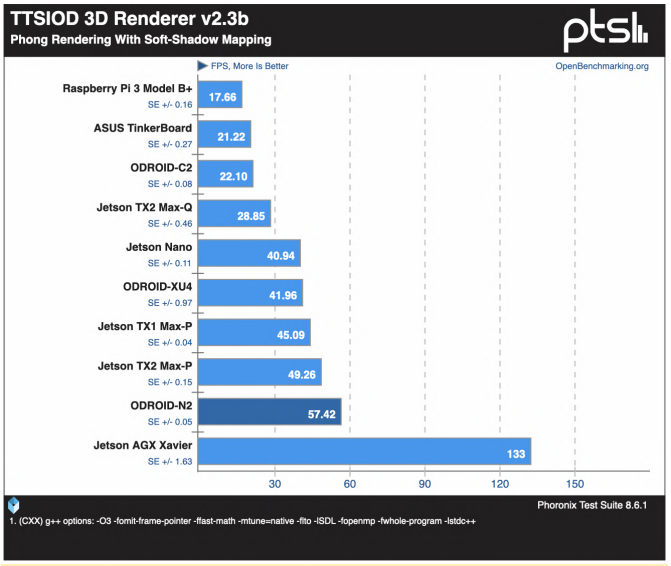


Figure 7 - TTSIOD 3D Renderer benchmark

Right out of the gate, the ODROID-N2 was offering very capable CPU performance in the multi-threaded TTSIOD 3D Renderer benchmark. The ODROID-N2 delivered better CPU performance than the $99 Jetson Nano and the Jetson TX2 while obviously coming in short of the premium Jetson AGX Xavier. But of the boards tested, the ODROID-N2 is a competitive SBC, especially with its $65~82 USD price-tag.

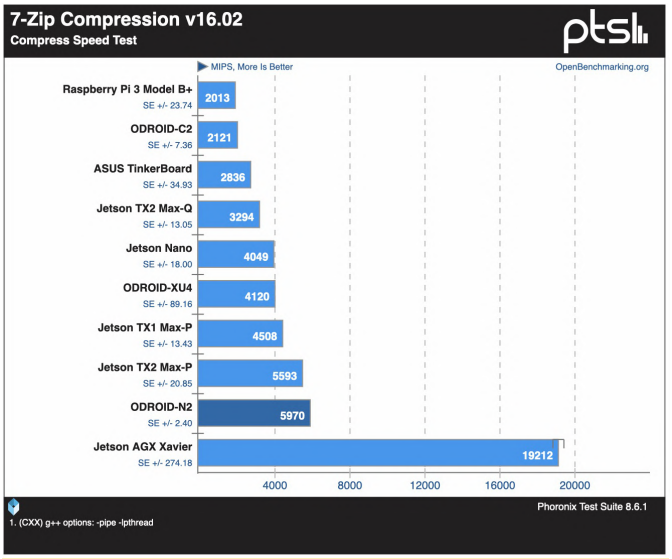


Figure 8 - 7-Zip compression benchmark

The positioning was similar with the 7-Zip compression benchmark where the ODROID-N2 was only outperformed by the much more expensive AGX Xavier while being a big upgrade over the likes of the Raspberry Pi 3.

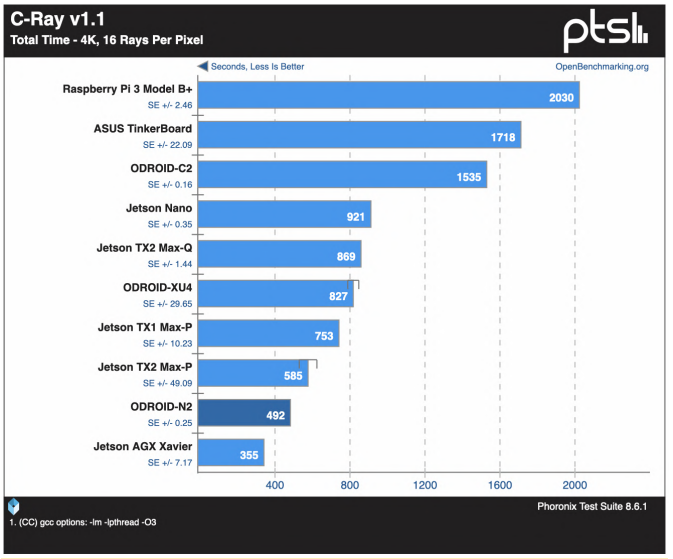


Figure 9 - C-Ray benchmark

The six-core ODROID-N2 also performs very well with the C-Ray multi-threaded ray-tracer.

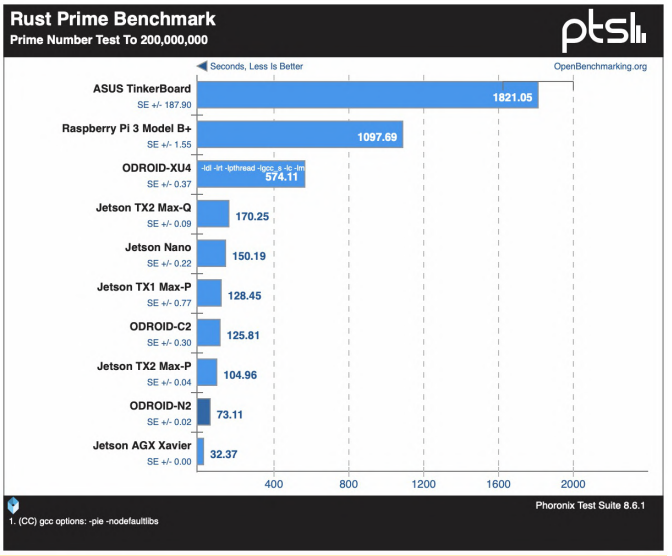


Figure 10 - Rust Prime benchmark

The ODROID-N2 also performed well with Rust.

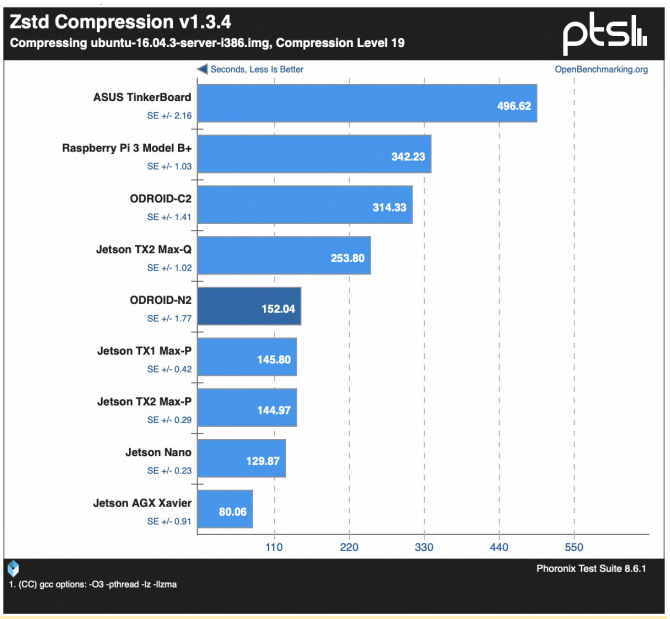


Figure 11 - Zstd compression benchmark

With the multi-threaded Zstd compression, the Jetson Nano and TX1/TX2 had a slight advantage but overall the ODROID-N2 was still performing great for its price in relation to the Raspberry Pi 3 Model B+, ASUS TinkerBoard, and others.

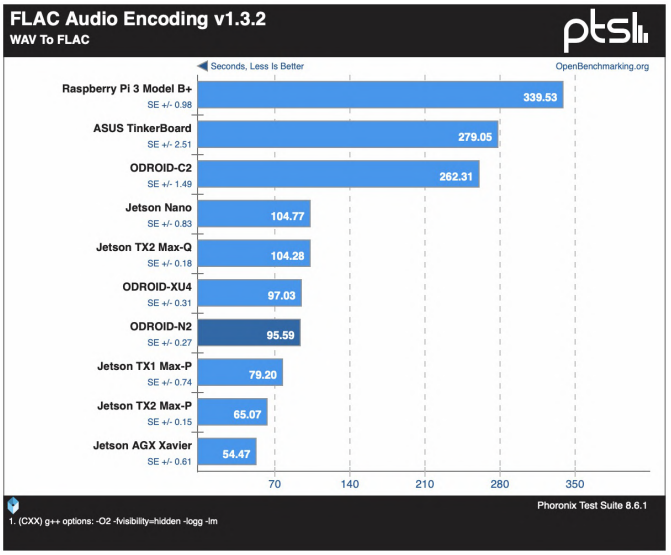


Figure 12 - FLAC audio benchmark

For the single-threaded FLAC audio encode benchmark, the ODROID-N2 delivered similar performance to the ODROID-XU4.

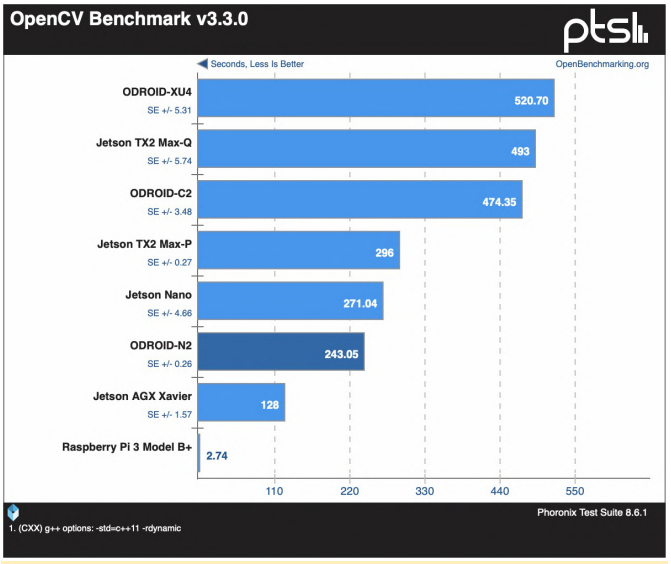


Figure 13 - OpenCV benchmark

The OpenCV performance out of the CPUs were also good. (Ignore the Raspberry Pi run as it was aborting early.)

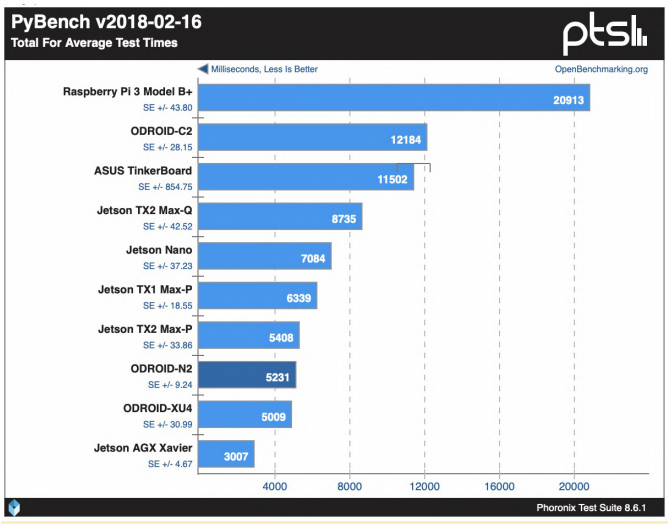


Figure 14 - PyBench benchmark

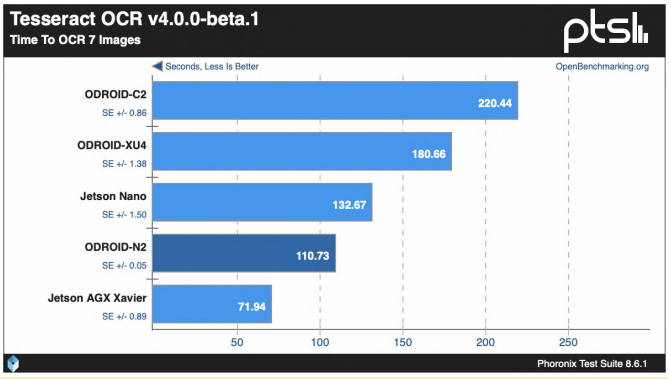


Figure 15 - Tesseract OCR

Overall, the ODROID-N2 delivers very good performance for its sub-$100 price tag. We are very happy with the performance out of Hardkernel's ODROID-N2. The performance with its four Cortex-

A73 and two Cortex-A53 cores was very competitive especially considering the board costs just over $80 USD for the 4GB RAM version (or $65 USD if only needing 2GB of RAM). The performance is very good, in fact, and opens up the ODROID-N2 for serving multiple hobbyist use-cases and even as a very lightweight ARM Linux desktop or other environments. What we also like are: - 4GB DDR4 system memory - 4 x USB 3.0 ports - Gigabit Ethernet - The Mali Bifrost graphics are in the process of being freed thanks to the Panfrost DRM/Gallium3D driver effort For those wanting to see how your own Linux board(s) compare to the performance of the ODROID-N2, I uploaded additional results via this OpenBenchmarking.org result file: **https://openbenchmarking.org/result/1904211-HV-ODROIDN2942**

After installing the Phoronix Test Suite, available at **https://www.phoronix-test-suite.com/**, simply run phoronix-test-suite benchmark 1904211-HV-ODROIDN2942 for your very own side-by-side, automated benchmark comparison from start to finish.

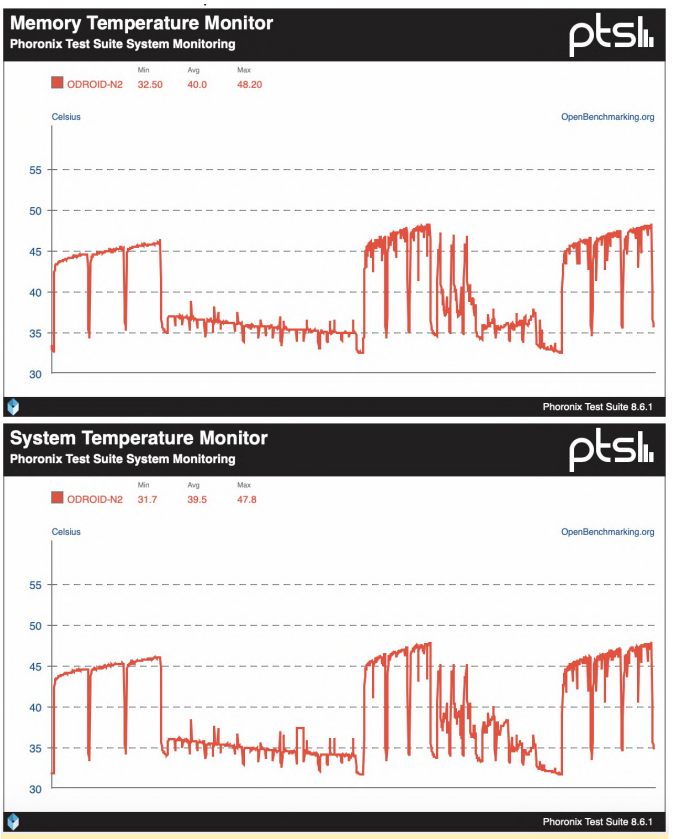


Figure 16 - Temperature Monitor Values

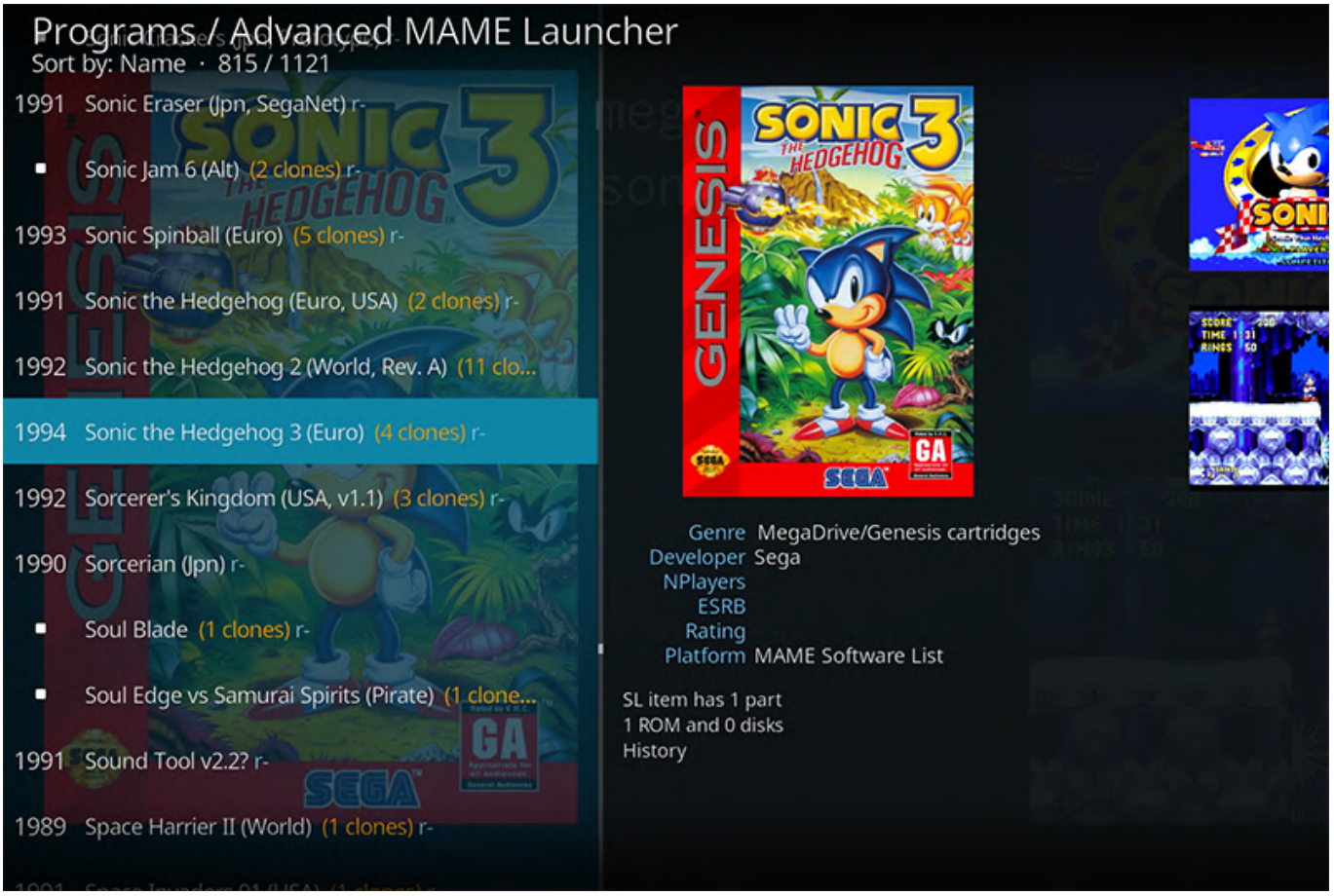
For those wondering about the thermal performance of the ODROID-N2, thanks to the large aluminum heatsink I haven't seen any noticeable thermal throttling or other issues. Here are various tests of the exposed SoC and DDR temperatures under different workloads. Under various single and multi-threaded workloads, the SoC and DDR4 memory never cracked 50 degrees Celsius under load and the average temperature under load was just 40 degrees thanks to this large passive heatsink. The minimum temperature during idle and starting out the benchmarks was just 31 degrees. More data in this OpenBenchmarking.org result file: **https://openbenchmarking.org/result/1904255-HV-ODROIDN2T59&obr_nbp=1**

Thanks to Hardkernel for sending over the ODROID-N2 for testing and those wanting to learn more about this ARM SBC can do so at Hardkernel.com.

The above article by Michael Larabel, was published on his website and can be found at **http://www.phoronix.com/vr.php?view=27780**.

Kodi and Advanced Mame on ODROID-XU4 - Part 1

June 1, 2019 By David Bellot Gaming, Linux, ODROID-XU4, Tutorial



This is a guide to setup Kodi with Mame, on the ODROID-XU4 SBC, making it a nice media and game center. It lists all the steps to install the needed software on Ubuntu Linux. Installing Mame is optional as well as configuring the USB drive, the joystick, etc. You can stop at the Kodi level or go further and have a fully functional video, music and game machine. Not all the steps are straightforward. Some steps may require consulting information online. I have done all the research for you and have provided resource links, so you have the information to understand necessary concepts. The first part on building Kodi is self-contained. The part on Mame will require external additional resources. The rest is self-contained again.

Install Linux and Kodi

To install all of this, follow these steps:

1. The first step is ideally done on another Linux box. If you do not have one yet, it is good to migrate to Linux as soon as you can. Download an image from: https://wiki.odroid.com/ODROID-XU4/os_images/linux/ubuntu_4.14/20181203

2. Ensure the image is valid with md5sum and compare the result with the corresponding md5sum file at https://odroid.in/ubuntu_18.04lts/XU3_XU4_MC1_HC1_HC2/.

```
$ md5sum ubuntu-18.04.1-4.14-mate-ODROID-XU4-20181203.img.xz
```

3. Assuming you are doing all of this on a Linux box, expand the file you downloaded:

```
$ xz -d ubuntu-18.04.1-4.14-mate-ODROID-XU4-20181203.img.xz
```

4. Plugin your SD card on an USB port of your Linux box.

5. Search for the device with lsblk.

```
$ lsblk
```

Note: Let us assume your SD card is on `/dev/sdX`. Be very careful to choose the correct device or you will risk erasing the content of another disk.

6. Write the image to the SD card.

```
$ sudo dd if=ubuntu-18.04.1-4.14-mate-ODROID-XU4-20181203.img.xz of=/dev/sdX bs=1M conv=fsync
$ sudo sync
```

7. Unmount your SD card either from your desktop environment or on the command line:

```
$ sudo umount /dev/sdX
```

8. Insert your SD card in your ODROID-XU4 and boot it up. Then login on the desktop with the credentials as given at: **https://bit.ly/30syIGW**

At this stage, you have a fully functional ODROID-XU4 desktop. The next step will be to install Kodi and remove this direct desktop access so that your box will directly boot on the Kodi interface more like a regular TV set.

9. Open a terminal (search in the menu on the top-left of the screen).

10. Update the packages:

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

During the update, the first time you boot your machine, you may get an error regarding a lock file with dpkg (`/var/lib/dpkg/lock`). It means that one of the automated update procedures is still running, which is very likely relevant to the new install. Just let it run until the end and try the commands again. If you get a message about boot.ini being replaced, just select 'OK'. If you get a question about `/etc/apt-fast.conf`, answer Y (for Yes).

11. Install Kodi:

```
$ sudo apt install kodi kodi-bin kodi-data libcec4 python-libcec openbox
```

12. The Mali graphical driver is installed by default and works well. In a terminal run `glmark2-es2` and you should see a demo (an image of a horse). The FPS should be around 300 frames/second. Run `glmark2` and you should see the same demo. The FPS should be around 30 frames/second--terrible. The reason is because the Mali driver only supports OpenGL ES (Embedded System) and not plain OpenGL. The solution is to use a library called `gl4es` (**https://github.com/ptitSeb/gl4es**), which you can find on Github.

13. Install `gl4es`

In a terminal, install the following development tools:

```
$ sudo apt install git cmake xorg-dev gcc g++ build-essential
```

Get the source code of gl4es. So, gl4es is an implementation of OpenGL using OpenGL ES (instead of implementing the library directly). Therefore, for systems like ODROID-XU4 with a driver supporting only OpenGL ES, it is possible to use software based on OpenGL too with hardware acceleration:

```
$ git clone https://github.com/ptitSeb/gl4es.git
$ cd gl4es
```

Compile it:

```
$ mkdir build
$ cd build
$ cmake .. -DODROID=1
$ make
```

It will compile and display its progress with a lot of messages. Towards the end, if the compilation is successful, you should see the following:

```
...
[ 96%] Building C object src/CMakeFiles/GL.dir/glx/utils.c.o
[ 98%] Linking C shared library ../lib/libGL.so.1
[100%] Built target
...
```

Save the Mesa version of OpenGL (the one installed by default):

```
$ sudo mv /usr/lib/arm-linux-gnueabihf/libGL.so.1 /usr/lib/arm-linux-gnueabihf/save.libGL.so.1
$ sudo mv /usr/lib/arm-linux-gnueabihf/libGL.so.1.0.0 /usr/lib/arm-linux-gnueabihf/save.libGL.so.1.0.0
```

and copy your new compiled version:

```
$ sudo cp lib/libGL.so.1 /usr/lib/arm-linux-gnueabihf/
$ sudo ldconfig
```

Now check that everything is fine by running:

```
$ glxinfo | head -6
```

If you see the following, then you are good to go (of course, the date on the second line will be different for you):

```
LIBGL: Initialising gl4es
LIBGL: v1.1.1 built on Mar 28 2019 06:22:26
LIBGL: Using GLES 2.0 backend
LIBGL: loaded: libGLESv2.so
LIBGL: loaded: libEGL.so
LIBGL: Using GLES 2.0 backend
```

Now you can run glmark2 again:

```
$ glmark2
```

This time you should see FPS around 300 to 600. At this stage, your ODROID-XU4 also has full OpenGL support. You can even use software like Blender (**https://www.blender.org/**).

14. For security reasons, Kodi and its associated programs will be run by a user with limited privileges, with no password and automatic login. We will call this user, `kodi`:

```
$ sudo adduser --disabled-password --gecos "" kodi
```

The output will be:

```
Adding user `kodi' ...
Adding new group `kodi' (1000) ...
Adding new user `kodi' (1000) with group `kodi' ...
Creating home directory `/home/kodi' ...
Copying files from `/etc/skel' ...
```

15. Assign some privileges to this user:

```
$ sudo usermod -a -G cdrom,video,plugdev,users,dialout,dip,input,netdev,audio,pulse,pulse-access,games kodi
```

16. Add options to allow Kodi to start its own X server:

```
$ sudo sed -ie 's/allowed_users=console/allowed_users=anybody/g' /etc/X11/Xwrapper.config
$ sudo sed -ie "$aneeds_root_rights = yes" /etc/X11/Xwrapper.config
```

17. Add the Kodi service to `systemd` (better to copy and paste the following rather than typing it):

```
$ cat << EOF | sudo tee /etc/systemd/system/kodi.service
 [Unit]
 Description = Kodi Media Center
 After = systemd-user-sessions.service network.target sound.target mysql.service dbus.service polkit.service upower.service
 Wants = mysql.service
 [Service]
 User = kodi
 Group = kodi
 Type = simple
 #PAMName = login # you might want to try this one, did not work on all systems
 ExecStart = /usr/bin/xinit /usr/bin/dbus-launch --exit-with-session openbox --startup /usr/bin/kodi-standalone -- :0 -nolisten tcp vt7
 Restart = on-abort
 RestartSec = 5

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

A quick note on the first line: the `sudo` command works on `tee` to write with super-user rights to the file, because if I would do it on `cat` and follow the same pattern as the other commands in this document, the redirection to the file would not work. Indeed, the redirection is done by the shell (belonging to the user `odroid`) and not by the command. So, `sudo cat` would not work.

18. Enable Kodi at boot time:

```
$ sudo systemctl enable kodi
$ sudo systemctl set-default multi-user.target
```

19. On the Hardkernel Wiki (**https://bit.ly/2Qjvcdf**), it is said that a recent Canonical EGL package blocked access to the Mali GPU. It is quite true. The fix is:

```
$ sudo apt install mali-x11 --reinstall
```

At the time of this publication, it fixes the loss of Mali GPU access bug. So it is highly recommended to apply this fix. In fact, if you update the system as I said before, you will have the bug and you will fix it with the reinstall `mali-x11` as above.

20. Now you have a fully functional Kodi system. If you want to reboot, your ODROID-XU4 will directly start into Kodi. If you want to add more features, like MAME (**https://www.mamedev.org**, **https://en.wikipedia.org/wiki/MAME**), an external USB drive, a joystick, a firewall, you can follow the rest of this article.

If you want to stop and enjoy your ODROID-XU4 now, you can reboot it:

```
$ sudo reboot
```

Everything in the following sections can be done either by connecting to your ODROID-XU4 with `ssh` or on screen, directly (you will need a keyboard). If you want to connect on the screen directly, after rebooting your ODROID-XU4 on Kodi, you can press `Ctrl+Alt+F1` to switch to a terminal (text screen). At any time, you can go back to Kodi by pressing `Alt+F7`.

Install an USB external drive

1. Switch to a terminal or login with `ssh`.

2. Install `usbmount` to automount USB drives:

```
$ sudo apt install usbmount
```

There is a bug in the version 0.0.22 of `usbmount` as provided by the stock Ubuntu Linux when connecting 2 USB drives at the same time. The bug has been fixed with `usbmount 0.0.24` which is not yet on the Ubuntu repository. You can upgrade it manually, if you like.

If your second drive (or even the first) does not mount properly when you plug it in, you can try to upgrade `usbmount` as follows:

```
$ git clone https://github.com/rbrito/usbmount.git
$ cd usbmount
$ sudo apt-get update && sudo apt-get install -y
debhelper build-essential fakeroot
$ dpkg-buildpackage -us -uc -b
$ cd ..
$ sudo dpkg -i usbmount_0.0.24_all.deb
```

3. By default, usbmount will mount the external USB drive with the `sync` option. `sync` means that all write access to the disk will be immediately flushed to the disk. It can dramatically slow down all the disk operations.

```
$ sudo sed -ie
's/^MOUNTOPTIONS=.*/MOUNTOPTIONS="noexec,nodev,noa
time,nodiratime"/' /etc/usbmount/usbmount.conf
```

I assume it is safe to use the `async` option here because you're not supposed to unplug this disk at any time. When I did that, the write speed of my external hard drive has been multiplied by 10.

Install MAME to play arcade games from Kodi

This section is long and will require external resources if you want to have all the bells and whistles of a full-featured MAME installation. From time to time, we will refer to external guides too.

1. Install MAME

```
$ sudo apt install mame mame-data mame-doc mame-
extra mame-tools libsdl2-gfx-1.0-0 libsdl2-image-
2.0-0 libsdl2-mixer-2.0-0 libsdl2-net-2.0-0
libsdl2-net-2.0-0 python-pil
```

Create a new config file with the correct paths, using OpenGL ES 2, the ALSA audio driver, tuning the speed and using all the cores:

```
$ cat << EOF | sudo tee /etc/mame/mame.ini
 inipath $HOME/.mame;/etc/mame
 rompath /home/kodi/AML-ROMs/
 samplepath /home/kodi/AML-assets/samples/
 ctrlrpath /home/kodi/AML-assets/ctrlr
 hashpath /usr/share/games/mame/hash
 cfg_directory $HOME/.mame/cfg
 nvram_directory $HOME/.mame/nvram
 memcard_directory $HOME/.mame/memcard
 input_directory $HOME/.mame/inp
 state_directory $HOME/.mame/sta
 snapshot_directory $HOME/.mame/snap
 diff_directory $HOME/.mame/diff
 comment_directory $HOME/.mame/comments
 video auto
 render opengles2
```

```
 audiodriver alsa
 samplerate 32000
 numprocessors 8
 mouse 1
 uimodekey INSERT
 autoframeskip 1
EOF
$ sudo chmod ugo+r /etc/mame/mame.ini
```

Then we need to remove the `gl4es` startup message to make MAME happy. Long story short: a Kodi plugin will extract the games' database `MAME.xml` to the standard output by running `mame`. When `mame` starts, `gl4es` is initialized and displays a nice message, like the one above. However, the Kodi plugin captures the standard output which is supposed to be an XML file, except that we have this welcome message on top of it from `gl4es`. So, when Kodi tries to read the XML file (in fact a Python library tries, too), it fails:

```
$ cat << EOF | sudo tee /usr/local/bin/mame
 #!/bin/sh
 export LIBGL_SILENTSTUB=1
 export LIBGL_NOBANNER=1
 /usr/games/mame "$@"
 EOF
$ sudo chmod ugo+x /usr/local/bin/mame
```

When configuring the AML add-on in Kodi, we will use this new mame command we have just created

2. Install and configure Advanced Mame Launcher plugin from Kodi

We are going to follow the official guide of a Kodi plugin called, Advanced Mame Launcher (AML) (**https://forum.kodi.tv/showthread.php?tid=304186**) and adapt a few steps to follow our setup. Here, I assume that we have an external USB drive connected to the ODROID-XU4. It will be helpful to store data for MAME.

Assuming you are on a text terminal (see above on how to switch to a text terminal from Kodi), the first step is to create directories to store MAME data. The external USB drive is mounted to `/media/usb0` and by default to `/media/usb,` too. We will assume this configuration is the case from now on.

```
$ cd /media/usb
$ sudo mkdir mame
$ sudo chown kodi.kodi mame
$ sudo su - kodi
$ cd /media/usb/mame
$ mkdir -p AML-ROMs AML-CHDs AML-SL-ROMs AML-SL-
CHDs AML-assets/samples/
$ cd AML-assets
$ mkdir artpreviews artwork cabinets clearlogos
covers_SL cpanels fanarts fanarts_SL flyers
manuals manuals_SL marquees PCBs snaps snaps_SL
titles titles_SL videosnaps videosnaps_SL
$ cd
$ ln -s /media/usb/mame/* .
$ exit
```

Then we need to fill in the directories with the latest data for Mame as described in **https://forum.kodi.tv/showthread.php?tid=304186**. We use the following script to download and install everything automatically:

```
$ sudo su - kodi
$ cd /media/usb/mame/AML-assets
# catlist.ini catver.ini genre.ini genre_OWS.ini
mature.ini not_mature.ini
$ wget http://www.progettosnaps.net/catver/ -q -O
- | grep 'download?tipo=catver' | sed "s#.*href="\
(.*\.zip\)".*#wget -q
'http://www.progettosnaps.net□' -O file.zip#"|sh
$ unzip -jq file.zip '*.ini'
$ rm file.zip
# nplayers.ini
$ wget http://nplayers.arcadebelgium.be/ -q -O - |
grep -E 'nplayers[[ :digit: ]]{4}\.zip' | sed
"s#.*href="\(http://nplayers.*zip\)">.*#wget -q
'□' -O file.zip#" | sh
$ unzip -jq file.zip nplayers.ini
$ rm file.zip

# bestgames.ini
$ wget http://www.progettosnaps.net/bestgames/ -q
-O - | grep 'download?tipo=bestgames' | sed
"s#.*href="\(.*\.zip\)".*#wget -q
'http://www.progettosnaps.net□' -O file.zip#"|sh
$ unzip -jq file.zip '*.ini'
$ rm file.zip

# series.ini
$ wget http://www.progettosnaps.net/series/ -q -O
- | grep 'download?tipo=series' | sed "s#.*href="\
(.*\.zip\)".*#wget -q
'http://www.progettosnaps.net□' -O file.zip#"|sh
$ unzip -jq file.zip '*.ini'
```

```
$ rm file.zip

# history.dat
$ wget https://www.arcade-history.com/?
page=download -q -O - | grep -o 'href="
[^"]*\.zip"' | sed 's#href="\.\.\(.*zip\)"#wget
https://www.arcade-history.com -q -O
file.zip#'|sh
$ unzip -jq file.zip history.dat

# mameinfo.dat
$ wget 'http://mameinfo.mameworld.info' --
header="User-Agent: Firefox/70.0" -q -O - |grep -o
'href="[^"]*Mameinfo.*\.zip"'|sort|tail -1| sed
's#href="\(.*zip\)"#wget --header="User-Agent:
Firefox/70.0"  -q -O file.zip#'|sh
$ unzip -qjp file.zip *.7z > mameinfo.7z
$ 7z e '-i!mameinfo.dat' mameinfo.7z > /dev/null
$ rm file.zip mameinfo.7z

# gameinit.dat
$ wget http://www.progettosnaps.net/gameinit/ -q -
O - | grep 'download?tipo=gameinit' | sed
"s#.*href="\(.*\.zip\)".*#wget -q
'http://www.progettosnaps.net' -O file.zip#"|sh
$ unzip -jq file.zip english/gameinit.dat
$ rm file.zip

# command.dat
$ wget http://www.progettosnaps.net/command/ -q -O
- | grep 'download?tipo=command' | sed
"s#.*href="\(.*\.zip\)".*#wget -q
'http://www.progettosnaps.net' -O file.zip#"|sh
$ unzip -jq file.zip Longhand/command.dat
$ rm file.zip
$ exit
```

3. Modify `mame.ini` to reflect the new directory structure:

```
$ sudo sed -i 's#^rompath \+.*$#rompath
/home/kodi/AML-ROMs/#' /etc/mame/mame.ini
$ sudo sed -i 's#^samplepath \+.*$#samplepath
/home/kodi/AML-assets/samples/#'
/etc/mame/mame.ini
```

4. Add assets.

Go back to Kodi: type 'Ctrl+D' on your terminal, then 'Atl+F7' to go back to Kodi.

In **https://forum.kodi.tv/showthread.php?tid=304186**, you can follow the paragraph called "Setting up MAME assets and Software List assets" to add more resources and assets. This is the way to get extra pictures, logos, etc. You can read the page at **http://forum.pleasuredome.org.uk/index.php?showtopic=30715** about the MAME Extra packages to understand all the types of resources you can find on the Net for Mame.

On the above mentioned page, you can find links to Mame resources on the Internet. They come in huge packages you simply have to move to `/media/usb/mame,` that we created before. Let me give you some examples about those packages, which can be useful in the process.

If you find a package called `MAME 0.xxx EXTRAs` where `xxx` is a Mame version number, go into the directory you have downloaded. In this directory, you will find more directories. Move their content to the corresponding directory in `/media/usb/mame/AML-assets/`

Another similar package might be called something like `MAME 0.xxx Multimedia`. You should process it the same way

However, you will find `.zip` files too. You can simply unzip them and move them to the final directory as above. Some directories do not have the same name, so keep the names we created above. In this case, transfer the content of the directory instead of the directory itself.

If you find packages with `bios-devices` in their name you do not need them if you already downloaded the very big packages with `ROMs` in their name. They are just a subset for a different situation. You can read more about all those packages at: **http://forum.pleasuredome.org.uk/index.php?showtopic=34705**. These packages are made when you want a minimal version of the ROMs; for example, if you are running short on disk space.

There are packages called Rollbacks Roms too (**https://bit.ly/2HL6Bdt**). You only need them if you have a ROM manager for Mame which can deal with multiple versions. You will not need them in this tutorial.

There are the Software List ROMs and CHDs. Obviously, their content will go into the `/media/usb/mame/AML-SL-ROMS` and

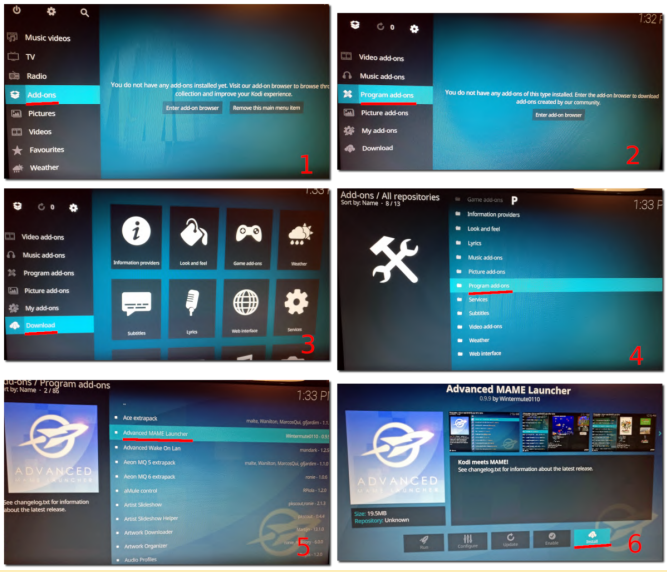
`/media/usb/mame/AML-SL-CHDs` directories we have created before. `SL` stands for `Software List`.

Finally, you might have messed up a bit with users' permission when downloading and moving files. So you want to make things well by assigning the `kodi` user to the `mame` directory:

```
$ sudo chown -R kodi.kodi /media/usb/mame
```

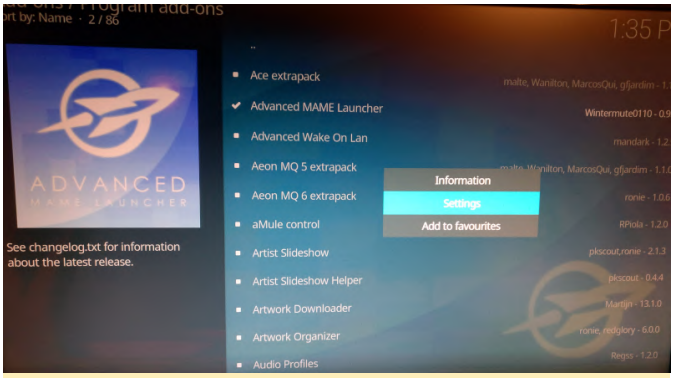
5. Install and configure the AML plugin. You will find it in Program Adds-on. It is called _Advanced Mame Launcher_.

In Kodi, go to **Settings**, **Addon settings**, **Install from repository**. In **Program add-ons**, look for **Advanced Mame Launcher** and install it. Follow this picture:



Installating the Advanced Mame Launcher

After it is installed, right-click on the add-ons logo and go to _Settings_:

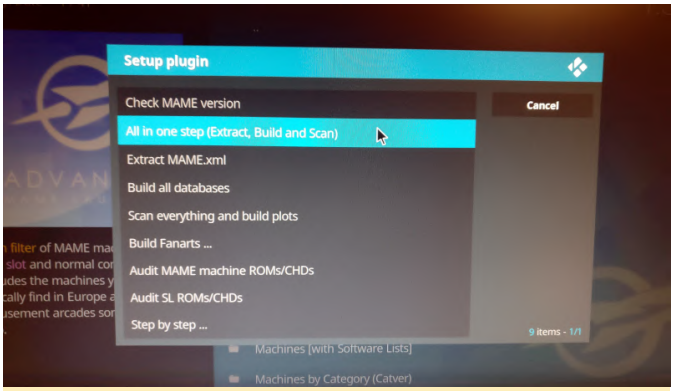


Advanced Mame Launcher settings selection

Change the paths to the executable file and the data directories as shown on this picture:

Updating the paths in the Advanced Mame Launcher settings

Go back to Kodi's initial screen and look for the AML plugin, in general **Add-ons**, **Program Add-ons**, **Advanced Mame Launcher**. Select any row, open the context menu with a right-click and select **Setup plugin**. Run a full setup and configuration of the plugin by choosing the _All in one step_ options in the context menu of the plugin.



Choosing the All In One Step option in the Advanced Mame Launcher plugin

This step can take several minutes to an hour. You will see a lots of progress bars. If you did everything well before, it should work without error messages. The plugin is now configured and ready, and Mame is now ready to be used.

In the next installment of this article, we will look at how to install a joystick. For comments, questions, and suggestions, please visit the rogiainl article at https://forum.odroid.com/viewtopic.php?f=52&t=34760.

ODROID-N2: Benchmarks

June 1, 2019 By Carlos Eduardo ODROID-N2

I recently received from Hardkernel an ODROID-N2, a new board replacing the cancelled ODROID-N1. Inside this package, I found the ODROID-N2 board, power supply, the clear case, WiFi USB adapter, and a 32GB eMMC card. The eMMC is way faster than SD cards.

Figure 01 - ODROID-N2 top view

Figure 02 - ODROID-N2 front view

The nice thing about the ODROID-N2 is that is uses a different SOC, an Amlogic S922X, giving a new perspective compared to most RK3399 top-end boards we see these days.

Some board specs: Hexa-core Amlogic S922X CPU with quad ARM Cortex-A73 and dual Cortex-A53 cores 4GB DDR4 RAM 1Gbps Ethernet 4 USB 3.0 ports, USB 3.0 hub behind a single USB 3.0 port from the SOC

More details can be found at https://www.hardkernel.com/shop/odroid-n2-with-4gbyte-ram/. My tests are always focused on server and console workloads. There are lots of benchmarks on Youtube and other blogs running games on Android or Linux desktop.

With the latest features from Docker, where you can build ARM images as easy as for x86, there are almost no drawbacks to use an ARM SBC for your server needs. I hope Hardkernel and Amlogic upstreams the patches to support this board. Now you need to use Hardkernel's own Kernel tree (https://github.com/hardkernel/linux) or more details on their wiki (https://wiki.odroid.com/odroid-n2/odroid-n2).

First, I installed DietPi, a lightweight Linux distribution based on Debian. They already have an image for the ODROID-N2. I just downloaded and unpacked the file and flashed it to the eMMC memory using Balena Etcher. Remember to purchase the eMMC-USB reader to make your life easy. The eMMC-USB reader is available at: https://www.hardkernel.com/shop/emmc-module-reader-board-for-os-upgrade/

CPU/Memory Benchmark

Here I compare synthetic benchmarks (DietPi benchmark and 7zip). These tests give a brief overview of the performance for the boards.

ODROID-N2 4GB

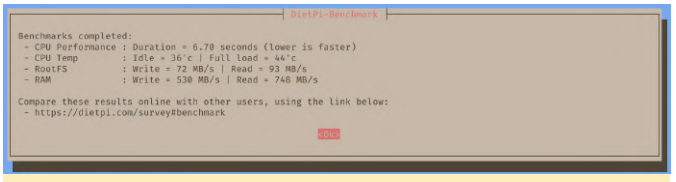


Figure 03 - ODROID-N2 Diet-pi benchmarks

```
carlosedp@DietPi:~ $ 7zr b

7-Zip (a) [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,6 CPUs LE)

LE
CPU Freq:   864  1368  1797  1796  1797  1797  1797  1797  1797

RAM size:    3711 MB,  # CPU hardware threads:   6
RAM usage:   1323 MB,  # Benchmark threads:      6

                       Compressing  |                 Decompressing
Dict     Speed Usage    R/U Rating  |      Speed Usage    R/U Rating
         KiB/s     %   MIPS   MIPS  |      KiB/s     %   MIPS   MIPS

22:       6017   516   1135   5853  |     121781   550   1890  10386
23:       6148   532   1179   6265  |     119234   549   1879  10317
24:       6099   535   1227   6559  |     115212   543   1861  10112
25:       6185   532   1328   7063  |     113745   544   1862  10123
----------------------------------  | ------------------------------
Avr:              528   1217   6435  |               546   1873  10234
Tot:              537   1545   8335
```

Figure 04 - ODROID-N2 7zr b benchmarks

Firefly RK3399–4GB

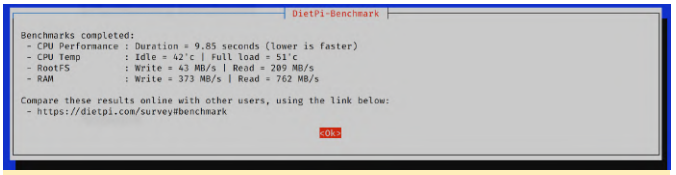


Figure 05 - RK3399 Diet-pi benchmarks

```
carlosedp@Firefly3399:~ $ 7zr b

7-Zip (a) [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,6 CPUs LE)

LE
CPU Freq:  1785  1791  1790  1790  1791  1791  1790  1791  1790

RAM size:    3811 MB,  # CPU hardware threads:   6
RAM usage:   1323 MB,  # Benchmark threads:      6

                       Compressing  |                 Decompressing
Dict     Speed Usage    R/U Rating  |      Speed Usage    R/U Rating
         KiB/s     %   MIPS   MIPS  |      KiB/s     %   MIPS   MIPS

22:       4693   505    905   4566  |      93626   524   1523   7985
23:       4195   480    890   4274  |      90674   522   1502   7846
24:       4199   524    861   4515  |      89313   526   1490   7839
25:       4233   562    860   4833  |      86415   526   1463   7691
----------------------------------  | ------------------------------
Avr:              518    879   4547  |               525   1495   7840
Tot:              521   1187   6194
```

Figure 06 - RK3399 7zr b benchmarks

On average, the ODROID-N2 is 30–35% faster than Firefly RK3399, which is my default board. Also, it has way better memory throughput, up to 40% faster. I also benchmarked other RK3399 boards in the past and they all score similar numbers.

Java Benchmarks

Here I ran some Java benchmarks aligned with my previous post comparing results on SPECjvm2008. On those tests, I ran in the Firefly RK3399 as well so the results will be aligned with the other benchmarks that have already done here. The benchmarks were run in a Docker container with the parameters:

```
$ docker run -it –rm -v $(pwd):/test
openjdk:8u181-jdk-stretch bash
$ java -jar SPECjvm2008.jar -wt 30s -it 1m -bt 6 -
i 3 -ikv -ict [benchmark]
```

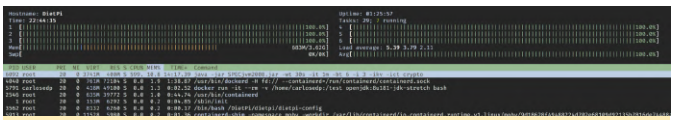


Figure 07 - Here are the Core speeds and a temperature during the benchmarks (100% on all cores)

```
DietPi CPU Info
Use dietpi-config to change CPU / performance options

Architecture  |  aarch64
Temperature   |  58'C : 136'F (Running warm, but safe)
Governor      |  interactive

              Current Freq    Min Freq    Max Freq
CPU0      |     1896 MHz      100 MHz     1896 MHz
CPU1      |     1896 MHz      100 MHz     1896 MHz
CPU2      |     1800 MHz      100 MHz     1800 MHz
CPU3      |     1800 MHz      100 MHz     1800 MHz
CPU4      |     1800 MHz      100 MHz     1800 MHz
CPU5      |     1800 MHz      100 MHz     1800 MHz
```

Figure 08 - CPU information

Benchmark	OpenJDK Java 1.8.0_181	Java 1.8.0 - OdroidN2	Performance Diff (positive is faster)
compress	58.72	64.5	9.84%
crypto.aes	35.85	43.49	21.31%
crypto.rsa	497.01	654.92	31.77%
crypto.signverify	659.5	831.44	26.07%
crypto composite	227.35	287.16	26.31%
derby	126.33	182.43	44.41%
mpegaudio	57.82	77.58	34.18%
scimark.large	41.06	44.14	7.50%
scimark.small	86.01	91.68	6.59%
serial	58.15	81.32	39.85%
sunflow	30.42	43.32	42.41%
xml	169.83	246.37	45.07%
		Average:	27.94%

Figure 09 - SPEC 2008 result chart

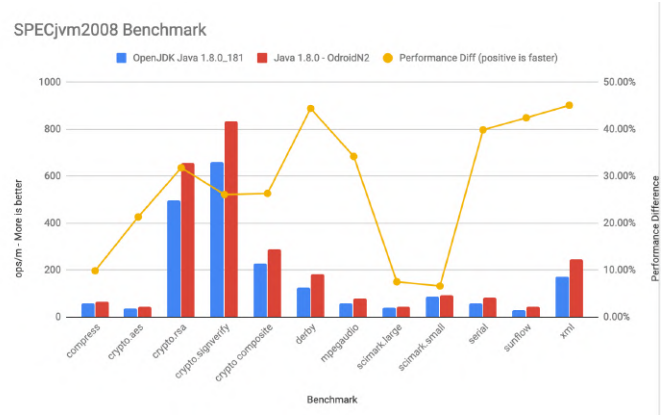


Figure 10 - SPEC 2008 result plot

As can be seen, the performance increase of around 30% persisted compared to RK3399.

Network

Here I test the network using iperf3. I test both TX and RX using the 1Gbps Ethernet connected to the same switch as the other computer. For the test server, I used my Macbook Pro connected with a 1Gbps Ethernet adapter. Here you can see the results from the ODROID-N2:

```
carlosedp@DietPi:~ $ iperf3 -c 192.168.15.141
Connecting to host 192.168.15.141, port 5201
[  4] local 192.168.15.15 port 42276 connected to 192.168.15.141 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec   103 MBytes   859 Mbits/sec    0   3.98 MBytes
[  4]   1.00-2.00   sec   101 MBytes   851 Mbits/sec    0   4.01 MBytes
[  4]   2.00-3.00   sec   100 MBytes   837 Mbits/sec    8   2.00 MBytes
[  4]   3.00-4.05   sec  98.8 MBytes   792 Mbits/sec    8   1.02 MBytes
[  4]   4.05-5.01   sec  95.0 MBytes   830 Mbits/sec   21    235 KBytes
[  4]   5.01-6.04   sec  98.8 MBytes   801 Mbits/sec   64    212 KBytes
[  4]   6.04-7.00   sec   100 MBytes   876 Mbits/sec    0    382 KBytes
[  4]   7.00-8.00   sec   101 MBytes   847 Mbits/sec    0    430 KBytes
[  4]   8.00-9.00   sec  99.6 MBytes   837 Mbits/sec    0    458 KBytes
[  4]   9.00-10.00  sec  97.5 MBytes   817 Mbits/sec   23    358 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec   995 MBytes   834 Mbits/sec  124             sender
[  4]   0.00-10.00  sec   991 MBytes   831 Mbits/sec                  receiver
```

Figure 11 - iperf3 benchmark

Reverse traffic mode gets lower numbers but, I saw similar numbers while testing the RK3399 board.

```
carlosedp@DietPi:~ $ iperf3 -R -c 192.168.15.141
Connecting to host 192.168.15.141, port 5201
Reverse mode, remote host 192.168.15.141 is sending
[  4] local 192.168.15.15 port 42280 connected to 192.168.15.141 port 5201
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-1.00   sec  29.8 MBytes   250 Mbits/sec
[  4]   1.00-2.00   sec  57.2 MBytes   480 Mbits/sec
[  4]   2.00-3.00   sec  78.5 MBytes   659 Mbits/sec
[  4]   3.00-4.00   sec  49.5 MBytes   415 Mbits/sec
[  4]   4.00-5.00   sec  60.0 MBytes   502 Mbits/sec
[  4]   5.00-6.00   sec  60.7 MBytes   510 Mbits/sec
[  4]   6.00-7.00   sec  63.0 MBytes   528 Mbits/sec
[  4]   7.00-8.00   sec  72.6 MBytes   608 Mbits/sec
[  4]   8.00-9.00   sec  63.5 MBytes   533 Mbits/sec
[  4]   9.00-10.00  sec  85.6 MBytes   719 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  4]   0.00-10.00  sec   621 MBytes   521 Mbits/sec                  sender
[  4]   0.00-10.00  sec   621 MBytes   521 Mbits/sec                  receiver
```

Figure 12 - iperf3 reverse traffic benchmarks

I tried disabling network checksum offload, a known issue on Rockchip SOCs, but the performance numbers didn't change.

Conclusion

The ODROID-N2 board has a huge potential and is the most powerful ARM SBC I've seen. It's use cases are infinite ranging from a home/mini server to a full-featured media center or desktop running almost any workload either installed or on containers. Also, it's fantastically suited for a Kubernetes cluster with multiple nodes.

Its power consumption is amazing and can be always on with only 2.8W while idle and 6.5W while benchmarking with all 6 cores at 100%. Furthermore, it's easy to flash new images, using eMMC, and the connectivity is plenty for most uses. I would love to see a PCI-E slot or an M.2. connector for NVMe drives. I found a document that states that the S922X SOC contains 1 PCI-E lane that in ODROID-N2 case was used for the USB 3.0 port. Other companies could provide this PCI-E lane as a M.2 connector or PCI-E slot.

All in all, I highly recommend the board and place it on top of my list with its features and within a reasonable price point for its performance and features. I think I might transform it soon into an ARM64 desktop with a full-featured Linux distribution like Ubuntu or if I can port Fedora.

This article was written by Carlos Eduardo and published on his medium page, which is available at **https://medium.com/@carlosedp/hardkernel-odroid-n2-review-and-benchmarks-b6996b002582**.

Meet An ODROIDian: Ry (@lordhardware)

June 1, 2019 By Rob Roy Meet an ODROIDian

Please tell us a little about yourself. I'm from Australia, was born in the 1990s, and grew up in a small town a few hours away from the closest city. I didn't have access to much technology growing up, but managed to buy an Atari 2600 at a community swap meet, which was my first introduction to anything interactive. Since then, technology has certainly been a passion of mine.

I work as a solutions architect for a small IT firm in Australia, much closer to the city now that I'm older. I studied programming at university but dropped out to study film. I figured, if someone is going to critique my creative output, let them critique from an artistic standpoint! I have a partner who I've been with for a decade now who works in finance. They don't share my hobby for electronics, but at least they tolerate it.

How did you get started with computers? When I moved to the suburbs as a kid, my family didn't have the money for a computer, but I used to collect the price guides from every computer store in the area and asked constant questions at the local market of all the PC gear sellers. When I was 9, I put my first computer together from parts I pulled from hard-rubbish (council hard waste, a yearly collection of all your unwanted broken items). It was an IBM with a 20MB hard drive (just enough space to install Commander Keen) with a Matrox card that worked every third boot.

What attracted you to the ODROID platform? I've always enjoyed tinkering, and honestly the price vs performance of my first was the biggest draw.

How do you use your ODROIDs? I've got a XU3, which is currently acting as the brain for my ducted heater system. My Odroid-W is burning a hole in a box and makes me guilty every time i look at it. My Odroid-Go is doing a pretty great job of serving the purpose intended for my Odroid-W however.

Figure 1 - Ry has a nice music workshop

Figure 2 - He has a variety of instruments that he plays

Which ODROID is your favorite and why? It has to be the Odroid-Go. When i got older and got a little money of my own, the Gameboy Pocket was one of the first things I got my hands on. The Go is a wonderful nostalgia trip without the incredible price tag that all those AAA batteries ended up being.

Figure 3 - The ODROID-Go lets Ry recreate his nostalgic Gameboy Pocket memories

What innovations would you like to see in future Hardkernel products? I love to see small, energy efficient systems. I was always fascinated when people managed to get incredibly complex programs or functions running on hardware that, by all rights, shouldn't have been capable. I honestly think we live in a golden age where the capabilities of our hardware far outweigh our abilities to efficiently utilise them so in many ways, that we've grown lazy in our programming. Seeing the things people do on embedded systems gives me a lot of hope.

What hobbies and interests do you have apart from computers? I'm pretty passionate about music, I built a studio in my garage and tend to stay in there during holidays to pluck or drum away.

Figure 4 - Ry has a box of unfinished projects in his workshop that includes an ODROID-W

What advice do you have for someone wanting to learn more about programming? Just keep at it.

Programming is a double-edged sword. It's about learning logical rules and how to apply them, but also requires you to think laterally and be creative to solve the problems you run into. On the one hand, it’s just like learning a new language, while on the other hand, being asked to write poetry with the new language's expressive nuance in mind. You never get good by regretting you did more of something, but you can be incredibly satisfied by even the smallest accomplishment.