X86 Exagear Emulation • Android Gaming • Meta Package Installation

ODROID Magazine

Year Two
Issue #14
Feb 2015

Docker

Deploying ready-to-use containers for running complex system environments

OS Spotlight: Ubuntu Studio

- Interfacing ODROID-C1 with 16 Channel Relay
- ODROID-C1 Minimal Install
- Device Configuration for Android Development
- Remote Desktop using Guacamole

Play with the Weather Board

What we stand for.

We strive to symbolize the edge of technology,
future, youth, humanity, and engineering.

Our philosophy is based on Developers.
And our efforts to keep close relationships with
developers around the world.

For that, you can always count on having the quality
and sophistication that is the hallmark of our products.

Simple, modern and distinctive.
So you can have the best to accomplish
everything you can dream of.

HARDKERNEL

Pollin ELECTRONIC

We are now shipping the ODROID U3 devices to EU countries! Come and visit our online store to shop!

Address: Max-Pollin-Straße 1
85104 Pförring Germany

Telephone & Fax
phone : +49 (0) 8403 / 920-920
email : service@pollin.de

Our ODROID products can be found at
http://bit.ly/1tXPXwe

Now that ODROID Magazine is in its second year, we've expanded into several social networks in order to make it easier for you to ask questions, suggest topics, send article submissions, and be notified whenever the latest issue has been posted. Check out our Google+ page at http://bit.ly/1D7ds9u, our Reddit forum at http://bit.ly/1DyClsP, and our Hardkernel subforum at http://bit.ly/1E66Tm6.

If you've been following the recent Docker trends, you'll be excited to find out about some of the pre-built Docker images available for the ODROID, detailed in the second part of our Docker series that began last month. For those who want to try x86 emulation, Tobias presents an overview of an application called Exagear, which allows many Windows applications to run on ARM architecture, including Skype. Venkat brings us the technical details for installing Guacamole, which enables remote desktop viewing from a browser, and Nanik continues his Android Development series with a look into device configuration.

For those interested in setting up a weather station, the ODROID Weather Board (http://bit.ly/1wtPdgP) makes a perfect addon, and Jussi had some fun by remotely monitoring meteorological conditions with it. Tinkering enthusiasts will enjoy the feature on connecting a 16-channel relay to the CI, and and musicians and artists can learn more about Ubuntu Studio, which is free to download and install on any Ubuntu distribution, providing lots of open-source media tools for creating and producing art, videos and music.

Android and mobile gaming has become very popular in recent years, and Bruno continues to present his favorite games for the ODROID, including Plants Vs. Zombies 2, Fish out of Water, and Pew Pew. If you have a favorite game that you'd like to see reviewed, create a post on the ODROID Magazine subforum or make a note on our Google+ page, and we may feature it in an upcoming issue!

ODROID Magazine, published monthly at http://magazine.odroid.com, is your source for all things ODROIDian.
Hard Kernel, Ltd. • 704 Anyang K-Center, Gwanyang, Dongan, Anyang, Gyeonggi, South Korea, 431-815
Makers of the ODROID family of quad-core development boards and the world's first ARM big.LITTLE architecture based single board computer.
Join the ODROID community with members from over 135 countries, at http://forum.odroid.com, and explore the new technologies offered by Hardkernel at http://www.hardkernel.com.

HARDKERNEL

ODROID Magazine

Rob Roy, Chief Editor

I'm a computer programmer living and working in San Francisco, CA, designing and building web applications for local clients on my network cluster of ODROIDs. My primary languages are jQuery, Angular JS and HTML5/CSS3. I also develop pre-built operating systems, custom kernels and optimized applications for the ODROID platform based on Hardkernel's official releases, for which I have won several Monthly Forum Awards. I use my ODROIDs for a variety of purposes, including media center, web server, application development, workstation, and gaming console. You can check out my 100GB collection of ODROID software, prebuilt kernels and OS images at `http://bit.ly/1fsaXQs`.

Bo Lechnowsky, Editor

I am President of Respectech, Inc., a technology consultancy in Ukiah, CA, USA that I founded in 2001. From my background in electronics and computer programming, I manage a team of technologists, plus develop custom solutions for companies ranging from small businesses to worldwide corporations. ODROIDs are one of the weapons in my arsenal for tackling these projects. My favorite development languages are Rebol and Red, both of which run fabulously on ARM-based systems like the ODROID-U3. Regarding hobbies, if you need some, I'd be happy to give you some of mine as I have too many. That would help me to have more time to spend with my wonderful wife of 23 years and my four beautiful children.

Bruno Doiche, Senior Art Editor

Made a pact with the fianceé to sweep the floor everyday, so he got himself a Roomba.

Nicole Scott, Art Editor

I'm a Digital Strategist and Transmedia Producer specializing in online optimization and inbound marketing strategies, social media directing, and media production for print, web, video, and film. Managing multiple accounts with agencies and filmmakers, from Analytics and Adwords to video editing and DVD authoring. I own an ODROID-U3 which I use to run a sandbox web server, live in the California Bay Area, and enjoy hiking, camping and playing music. Visit my web page at http://www.nicolecscott.com.

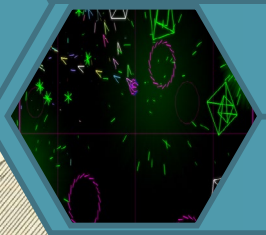
James LeFevour, Art Editor

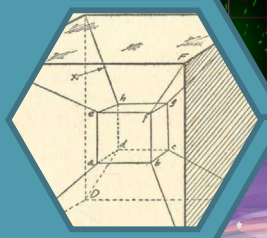
I am a Digital Media Specialist who is also enjoying freelance work in social network marketing and website administration. The more I learn about ODROID capabilities the more excited I am to try new things I'm learning about. Being a transplant to San Diego from the Midwest, I am still quite enamored with many aspects that I think most West Coast people take for granted. I live with my lovely wife and our adorable pet rabbit; the latter keeps my books and computer equipment in constant peril, the former consoles me when said peril manifests.

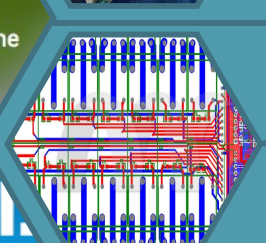
Manuel Adamuz, Spanish Editor

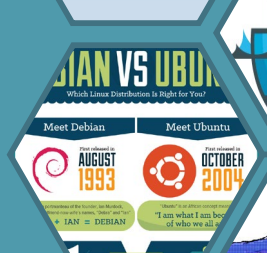
I am 31 years old and live in Seville, Spain, and was born in Granada. I am married to a wonderful woman and have a child. A few years ago I worked as a computer technician and programmer, but my current job is related to quality management and information technology: ISO 9001, ISO 27001, and ISO 20000. I am passionate about computer science, especially microcomputers such as the ODROID and Raspberry Pi. I love experimenting with these computers. My wife says I'm crazy because I just think of ODROIDs! My other great hobby is mountain biking, and I occasionally participate in semi-professional competitions.

INDEX









X86 EMULATION

A LOOK INTO EXAGEAR

by Tobias Schaaf

Eltechs ExaGear desktop is a virtual machine that implements a software-based x86 Linux container on ARM and allows you to run Intel x86 applications directly. It is like QEMU but 4.5 times faster. You can even run Windows applications on your ARM device if you install Wine. When ExaGear was first announced, I had doubts about its capabilities, and never thought I would actually use it. However, when I installed it a short time ago, I was actually surprised, and want to share my experience and results with x86 emulation on ARM devices using ExaGear.

Overview

ExaGear is not free, and a license must be purchased from Eltechs at http://bit.ly/YbPqc5. ExaGear comes with installable Debian packages and a stripped-down Ubuntu 12.04 x86 image. The package should work on all Debian-based systems such as the official Ubuntu 14.04 images from HardKernel, as well as any Debian image, such as my ODROID GameStation Turbo image.

ExaGear works by running x86 applications on your ARM-based ODROID board using the kernel and drivers coming from your board. It seems to simply translate x86 function calls into ARM equivalents, and for this purpose, it is rather efficient.

What can you do

Since ExaGear comes with a minimal Ubuntu 12.04 image, you can install and (theoretically) run any program that is compatible with Ubuntu 12.04, which gives a wide range of applications. Basically everything that's in the Ubuntu repository can be installed, including applications that come from Ubuntu partners, like Zentyal and Steam. Generally, everything that runs under the native Linux window management runs surprisingly fast with ExaGear.

What can't you do

Although you can install everything that come with or exists for Ubuntu 12.04, you are still limited with what you can do with ExaGear. For instance, you can't use any hardware-accelerated applications since the drivers do not support this. Anything that requires OpenGL, for example, will only run in software emulation through MESA software version of

Exagear runs x86 applications on ARM devices, including the ODROID family

OpenGL. This means that not all functions are supported, and the graphics are rather slow.

For example, you can install Steam on the ODROID, but you can't run it, since some functions are missing and it won't start. Any applications that requires fancy graphics won't run, and the same applies to Windows applications. Although you can run many Windows x86 applications, those that use heavy DirectX components like 3D games won't work, or will be very slow, and actually will make your ODROID run very hot.

Real examples

With all that said, you might wonder, what is ExaGear good for and what can you expect? Check out Figure 1 for an example of ExaGear working. As you can see,– it is quite capable but with some limitations.

I have found that many programs have issues with the sound. If there are multiple sound samples playing at the same time the sound gets scratchy with audible delays. So, although gaming is possible to a certain degree, don't expect it to be perfect.

Skype and TeamViewer

One of the most useful things that you can do with ExaGear is to run applications that are widely used in x86 envi-

Figure 1 - Windows game Total Annihilation running on the ODROID-XU3

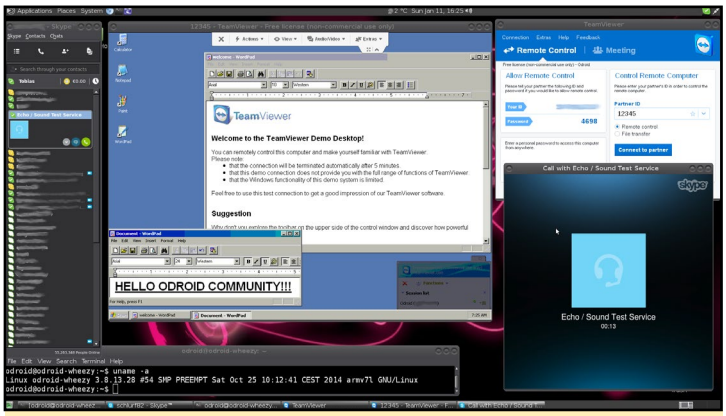


Figure 2 - Skype and TeamViewer running simultaneously using ExaGear on an ODROID-U3

ronments, but that do not exist for ARM. Skype and Teamviewer are two of the best examples. As previously mentioned, if a program does not require any special mechanism, they run perfectly fine on ODROID devices.

Both programs can be used without any restrictions. I have already made calls using Skype on my TV using an ODROID-U3 already, even with a camera for a video call, which runs flawlessly. The same experience applies to for TeamViewer, which in fact uses Wine, and means that it's actually running a Windows application directly on the ODROID. As shown in Figure 2, you can even run both Skype and TeamViewer simultaneously.

Shortcuts

The "normal" way to start a program with ExaGear would be to open a Terminal session and type in the command "exagear". After that, you are in the x86 environment, where you once again can start certain applications via a command line, for example, typing "skype" to start Skype.

Although this method works, it's a little bit complicated, especially since you can not close this terminal window without terminating the x86 program as well. Therefore, I want to show you how to start programs directly from your ARM environment without using the exagear command.

Skype

The first thing to do is to find out how programs are started via ExaGear, and see if we can replicate it. To determine this, start Skype using the "normal" method, which assumes that you already have Skype installed in your ExaGear environment. First, start Skype from the terminal:

```
$ exagear
```

Starting the shell in the guest image /opt/exagear/images/ubuntu-1204lts

```
$ skype
```

Open a new terminal, or a new tab in your current terminal window, and type:

```
$ ps aux | grep skype
```

You will find a line similar to this:

```
odroid    3125  5.7 15.1 828424 314764 ?       Sl
16:00   2:59 /opt/exagear/bin/ubt_x32a32_al --path-
prefix /opt/exagear/images/ubuntu-1204lts --vpaths-
list /opt/exagear/images/ubuntu-1204lts/.exagear/
vpaths-list --hifd-base 4095 -f /usr/bin/skype -
skype
```

If your terminal is too short, you might not see the full command line. In that case, use this command instead, which redirects the output to a file called skype.txt, which you can open with any text editor.

```
$ ps aux | grep skype > skype.txt
```

Among other information, we can see the command that is used to start Skype:

```
/opt/exagear/bin/ubt_x32a32_al --path-prefix /opt/
exagear/images/ubuntu-1204lts --vpaths-list /opt/
exagear/images/ubuntu-1204lts/.exagear/vpaths-list
--hifd-base 4095 -f /usr/bin/skype -- skype
```

Let's experiment to see if this command actually works. Quit Skype, then open a new terminal session so that you are no longer in the ExaGear environment, and try out this newly found command.

If everything works as intended, you will see Skype start up normally, which means that it was launched directly from your

ARM environment without the extra step of opening the ExaGear environment first. Although this does not seem much of a difference, since you are still using the terminal to start Skype, it's just the first step.

To make things easier, let's prepare a start shortcut for Skype. We don't actually have to do that much work, since such a starter already exists in the ExaGear environment. Let's use that instead of rewriting everything. To do so, open a new terminal and logon as root:

```
$ su
Password:
$ echo "/opt/exagear/bin/ubt_x32a32_al --path-prefix
/opt/exagear/images/ubuntu-1204lts --vpaths-list /
opt/exagear/images/ubuntu-1204lts/.exagear/vpaths-
list --hifd-base 4095 -f /usr/bin/skype - skype" > /
usr/local/bin/skype
$ chmod +x /usr/local/bin/skype
$ cp /opt/exagear/images/ubuntu-1204lts/usr/share/
applications/skype.desktop /usr/local/share/applica-
tions/
$ cp /opt/exagear/images/ubuntu-1204lts/usr/share/
pixmaps/skype.png /usr/local/share/pixmaps
```

Once these steps are completed, we already have everything that we need in order to start Skype. If you open the program list by clicking the Start button and navigate to Internet, you should find a new icon labelled "Skype". From now on, whenever you want to start Skype, just click on that icon, just like you would in a real x86 environment, and after a short wait, Skype will start up just like a native ARM application.

TeamViewer

Running TeamViewer directly from a shortcut works similarly to Skype, but is slightly more complicated. TeamViewer has a small problem, since it needs the TeamViewer daemon service, called teamviewerd, to start before the actual TeamViewer program. The TeamViewer daemon may only be started as root, so we would have to start ExaGear as the root user, then start the teamviewerd service, then logon a second time as a normal user in ExaGear, and finally start TeamViewer. We want to end up with the same simple and direct way of starting TeamViewer as we have with Skype already, so let's apply some Linux magic in order to get it working as desired. To begin, open a new terminal and create the following script:

```
$ su
Password:
$ cat <<\EOF > /etc/init.d/teamviewerd_exagear
#! /bin/sh
### BEGIN INIT INFO
# Provides:          teamviewerd_exagear
# Required-Start:    $local_fs $remote_fs
# Required-Stop:
# X-Start-Before:
# Default-Start:     2 3 4 5
# Default-Stop:          0 1 6
# Short-Description: starts teamvierwer deamon
# Description: Starts the teamvierwer deamon for
teamviewer useage through exagear.
### END INIT INFO
set -e
case "$1" in
 start)
      # only run as root
      if [ `id -u` -eq 0 ];
      then
                /opt/exagear/bin/ubt_x32a32_al
--path-prefix /opt/exagear/images/ubuntu-1204lts
--vpaths-list /opt/exagear/images/ubuntu-1204lts/.
exagear/vpaths-list --hifd-base 4095 -f /opt/team-
viewer/tv_bin/teamviewerd -- /opt/teamviewer/tv_bin/
teamviewerd
      fi
      ;;
  stop)
      PID=`ps aux | grep "/opt/exagear/bin/ubt_
x32a32_al --path-prefix /opt/exagear/images/ubuntu-
1204lts --vpaths-list /opt/exagear/images/ubuntu-
1204lts/.exagear/vpaths-list --hifd-base 4095 -f /
opt/teamviewer/tv_bin/teamviewerd -- /opt/teamview-
er/tv_bin/teamviewerd" | grep -v grep | awk '{print
$2}'`
      if [ ! -z $PID ];
      then
              kill $PID
      fi
      ;;
  *)
      echo "Usage: $N {start|stop}" >&2
      exit 1
      ;;
 esac
 exit 0
EOF
$ chmod +x /etc/init.d/teamviewerd_exagear
$ update-rc.d teamviewerd_exagear defaults
```

This creates a service script that can be started as root, which cal also run each time you start your ODROID:

```
$ service teamvierwerd_exagear start
```

Then, we create a launcher shortcut for TeamViewer, like we did for Skype:

```
$ su
Password:
$ cp /opt/exagear/images/ubuntu-1204lts/usr/share/
applications/teamviewer-teamviewer10.desktop /usr/
local/share/applications/
$ cp /opt/exagear/images/ubuntu-1204lts/opt/team-
viewer/tv_bin/desktop/teamviewer.png /usr/local/
share/pixmaps/
$ cp /usr/local/bin/skype /usr/local/bin/teamviewer
$ pico /usr/local/bin/teamviewer
$ pico /usr/local/share/applications/teamviewer-
teamviewer10.desktop
```

For the last two steps, we need to edit the files slightly. In the file /usr/local/bin/teamviewer, replace the word "skype" two times with the word "teamviewer", then save the file by pressing Ctrl-X, answering with "y" for yes, then pressing Enter twice. Next, in the file /usr/local/share/applications/teamviewer-teamviewer10.desktop, change the Icon patch to include just "teamviewer" and nothing else, then save that file as well. Now, we can either start the teamviewerd_exagear service manually, or reboot the ODROID and then launch TeamViewer in the same way as we did with Skype, using the Start button's Internet submenu.

Caveats

There are some quirks with ExaGear that make things a little bit harder to use. For example, the file access in the ExaGear environment is somewhat slow. Starting TeamViewer can take anywhere from 30 seconds up to a minute on an ODROID-U3, which happens regardless of whether you're using an eMMC module or SD card. You should occasionally run "apt-get update" in your ExaGear environment in order to update the package lists, or else some packages may not be installed during the TeamViewer or Skype installation.

If you perform a system update using "apt-get upgrade" and/or "apt-get dist-upgrade" command, you might encounter several issues, since the image was highly modified. I noticed that a few things were forgotten, such as altering the initramfs-tools to disable the creation of an initrd.img file, which isn't possible anyway. Also, some packages will fail to update, which requires some Linux expertise to fix, but an upgrade is probably not really necessary once everything is working.

Another issue is that ExaGear distributes tasks over all CPU cores, which is generally a very good thing since it uses all the power it can get, but it also can lead to a very hot CPU if an application uses a lot of CPU power. For example, I was running a Windows application called Blender on my XU3 using ExaGear and Wine, which resulted in all 8 cores running at 100%, and even with the fan spinning at its maximum speed, the temperature rose to over 94°C (200°F)!

Overall, I really like what you can do with ExaGear, and although I was very skeptical when it was first announced, I have to say it's doing a very good job.

Wine

If you use ExaGear with Wine, a very convenient program is PlayOnLinux, which allows you to easily configure and install Windows applications under Wine. If you try to run full-screen applications such as games using Wine, you need to configure Wine to run in a fake desktop with a size of 800x600 or 1024x768, rather than allow it to run natively in Linux. PlayOnLinux may spare you some of these resolution problems, especially with the C1, which cannot change resolutions on the fly. PlayOnLinux also makes recovery easier when a program hangs, since it is able to actually close the specific program.

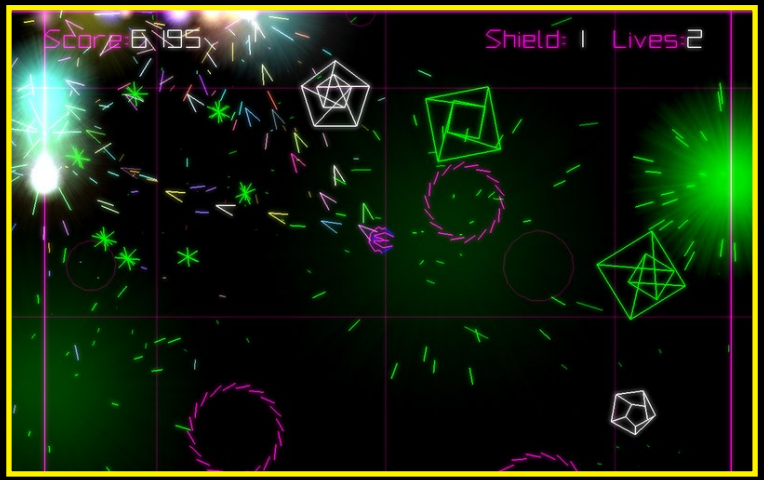
PEW PEW

SHOOT 'EM UP FUN WITH SPACE AND ASTEROIDS!

by Bruno Doiche

When classic games are re-invented on modern hardware, they are just the best! Pew Pew is a multi-directional shoot 'em up for Android. It's basically megatons of enemies with many different game modes, combined with sweet smooth retro graphics. Win medals, unlock ships, and compete on the online ladder.

```
https://play.google.com/store/apps/
details?id=com.jyaif.pewpew
```



TASKSEL

EASILY INSTALL UBUNTU METAPACKAGES FROM THE CLI

edited by Rob Roy

Tasksel is a Debian and Ubuntu compatible tool that facilitates the installation of multiple related packages as a coordinated "task" onto your system, allowing single-click installations of web server bundles, desktop environments, and software suites. The installation function is similar to that of meta-packages, and most of the tasks available from tasksel are also available from the Ubuntu package managers, such as Synaptic Package Manager.

Installation

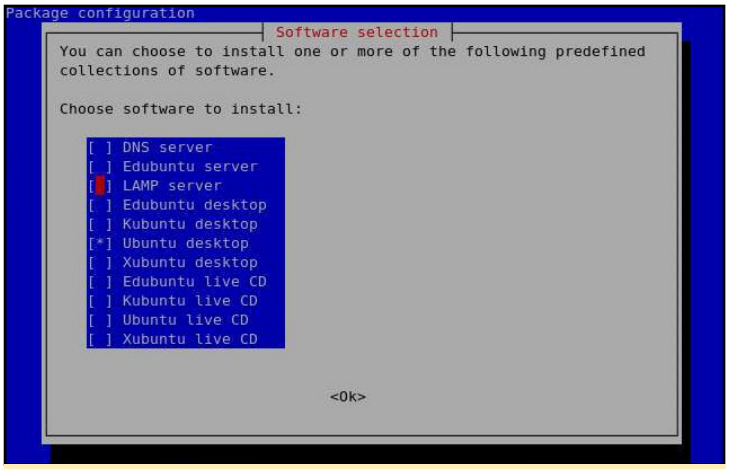
If tasksel is not already installed, it may be downloaded with the following command:

```
$ sudo apt-get install tasksel
```

Usage

To run tasksel from the command line, type the following into a Terminal window, which will show the menu seen in below:

```
$ sudo tasksel
```



The main Tasksel menu allows single-click package installation

Already-installed tasks are indicated with an asterisk beside their name. Select a task by scrolling down and pressing space, which will put an asterisk beside the selected task and mark it for installation. Removing an asterisk marks the task for removal. After pressing "OK", the selected the task installations and/or removals will take place using apt-get.

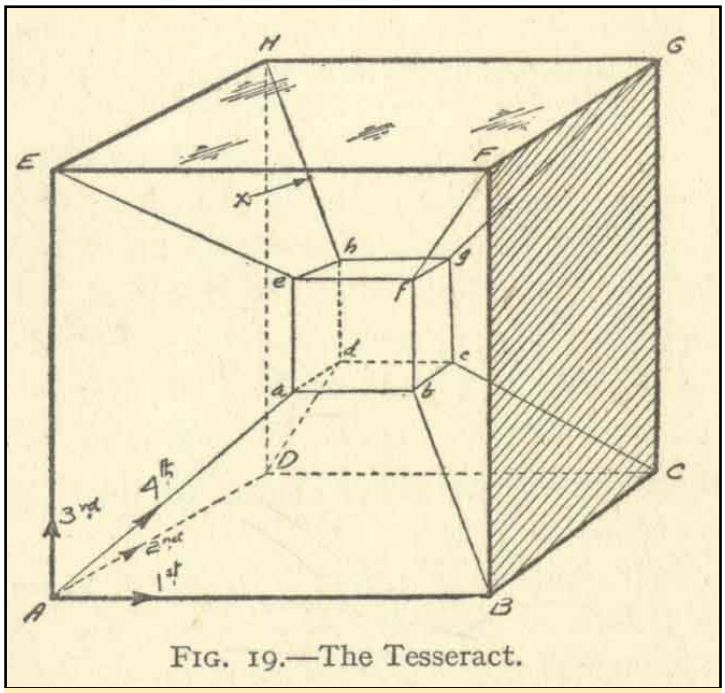


FIG. 19.—The Tesseract.

Does going through the thousands of packages available for Ubuntu seem like 4th dimensional physics? Use Tasksel instead!

Command line arguments

You can also directly specify which task to install on the command line. For instance, to add the Apache-MySQL-PHP stack to an existing system, type:

```
$ sudo tasksel install lamp-server
```

For complete options, see the tasksel manual by typing:

```
$ man tasksel
```

Usage

Tasks can also be installed with apt-get using the syntax:

```
$ sudo apt-get install <task_name>
```

For example, the following commands will install the Ubuntdesktop, Kubuntdesktop and LAMP server, respectively:

```
$ sudo apt-get install ubuntu-desktop
$ sudo apt-get install kubuntu-desktop
$ sudo apt-get install lamp-server
```

Tasks list

Tasks are defined in .desc files located in the /usr/share/tasksel directory. The default list available in Ubuntu may be viewed with the following command:

```
$ grep Task /usr/share/tasksel/ubuntu-tasks.desc
```

Package descriptions

Below is a list of the tasks in Ubuntu 14.04 Trusty Tahr that can be installed using the tasksel meta package manager. The supported packages may change between Ubuntu versions, so be sure to run the following command in order to see the latest list:

```
$ tasksel --list

server                    Basic Ubuntu server
openssh-server            OpenSSH server
dns-server                DNS server
lamp-server               LAMP server
mail-server               Mail server
postgresql-server         PostgreSQL database
print-server              Print server
samba-server              Samba file server
tomcat-server             Tomcat Java server
cloud-image               Ubuntu Cloud Image (instance)
virt-host                 Virtual Machine host
ubuntustudio-graphics     2D/3D creation and editing suite
ubuntustudio-audio        Audio recording/editing suite
edubuntu-desktop-gnome    Edubuntu desktop
kubuntu-active            Kubuntu Active
kubuntu-desktop           Kubuntu desktop
kubuntu-full              Kubuntu full
ubuntustudio-font-meta    Large selection of font packages
lubuntu-desktop           Lubuntu Desktop
lubuntu-core              Lubuntu minimal installation
mythbuntu-desktop         Mythbuntu additional roles
mythbuntu-frontend        Mythbuntu frontend
mythbuntu-backend-master  Mythbuntu master backend
mythbuntu-backend-slave   Mythbuntu slave backend
ubuntustudio-photography  Photograph touchup/editing suite
ubuntustudio-publishing   Publishing applications
ubuntu-gnome-desktop      Ubuntu GNOME desktop
ubuntu-desktop            Ubuntu desktop
ubuntu-usb                Ubuntu desktop USB
ubuntustudio-video        Video creation and editing suite
xubuntu-desktop           Xubuntu desktop
edubuntu-dvd-live         Edubuntu live DVD
kubuntu-active-live       Kubuntu Active Remix live CD
kubuntu-live              Kubuntu live CD
kubuntu-dvd-live          Kubuntu live DVD
lubuntu-live              Lubuntu live CD
ubuntu-gnome-live         Ubuntu GNOME live CD
ubuntustudio-dvd-live     Ubuntu Studio live DVD
ubuntu-live               Ubuntu live CD
ubuntu-usb-live           Ubuntu live USB
xubuntu-live              Xubuntu live CD
```


FISH OUT OF WATER

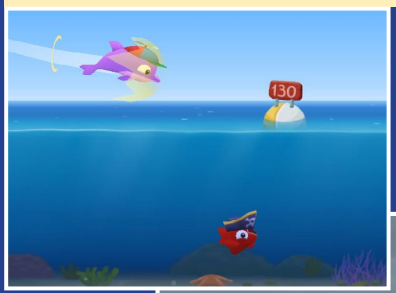
SKIM YOUR MOUSE AROUND ON VIRTUAL SEAS

by Bruno Doiche

If you are in the mood for a totally fun casual game, take a look at Fish Out of Water. It's not a typical instantly addictive game, but has a different appeal of a casual game. You just throw your characters across the water and try to skim them as far as possible in order to get a high score. Created by Halfbrick, the creators of the classic Fruit Ninja game, this pastime is certainly worth your attention!

https://play.google.com/store/apps/details?id=com.halfbrick.FishOutOfWater

This game features fish wearing hats! Isn't that awesome?



REMOTE DESKTOP

USING GUACAMOLE

by Venkat Bommakanti

Are you looking for a clientless, browser-based remote desktop solution for accessing an ODROID such as a remotely located C1? Guacamole is perfect for your project! All you need is a web browser with HTML5 support in order to be able to use your device from a smartphone, laptop or desktop PC, since no plugin or client-side software is required.

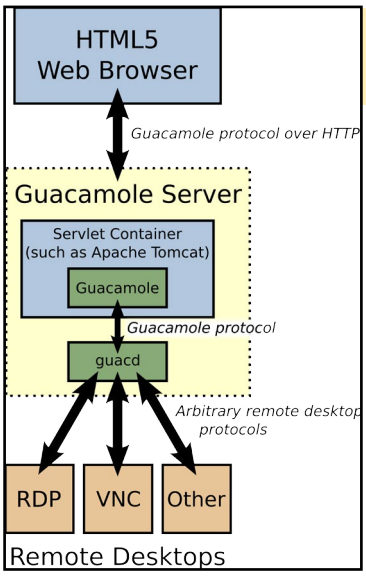


Figure 1: Typical Guacamole server architecture

Guacamole is an HTML5 protocol-agnostic remote-desktop web application, which supports several remote desktop protocols such as VNC, SSH and RDP. Figure 1, obtained from guac-dev.org, illustrates the architecture, which essentially consists of three (3) components:

- a guacamole servlet hosted by Apache tomcat (servlet container) that fields web-based requests.
- a guacd daemon that talks the guacamole protocol with the servlet.
- a guacamole-client, a fully server-side resident component, that serves HTML5 user-interface (UI) content

Requirements

1. An ODROID-C1 - while this article targets a C1, it can apply to a U3 or an XU3 series board
2. C1 accessories such as HDMI cable, CAT 5E+ ethernet cable or WIFI 3 dongle, PSU, and RTC battery
3. A 16GB+ eMMC 5.0 card with the latest XU3-Lite specific lubuntu desktop image like ubuntu-14.04.1lts-lubuntu-odroid-c1-20150102.img and/or a 16GB+ Class 10 MicroSD card with an SDCard reader/writer
4. A network where the device has access to the Internet and the ODROID forums
5. Network access to the C1 via utilities like PuTTY, FileZilla, TightVNC Viewer (MS Windows 7+) and Terminal (Mac, linux) from a development machine
6. Apache tomcat 6
7. Guacamole 0.8.3

Preparing Lubuntu

Install the latest C1 image onto the eMMC card, then attach the eMMC card to the C1. With the HDMI display attached, boot up the system. The first step is to run the ODROID Utility, then expand the installation partition to use all of the eMMC by selecting the "Resize your root partition" option. Reboot, then run the ODROID Utility again, configure and update all remaining aspects of the system such as the kernel and video drivers, then reboot the system again.

Install related software

Run the following commands to install the necessary guacamole 0.8.3 web-application software:

```
$ sudo apt-get install guacamole guacamole-tomcat
guacd
$ sudo apt-get install libguac-client-vnc0 libguac-
client-rdp0 libguac-client-ssh0
```

Setup user accounts

The installation process installs the following files:

```
$ cd /etc/guacamole
$ ls -lsa
 4 -rw-r--r--   1 root root             1099 Sep 21
2013 guacamole.properties
 4 -rw-r-----   1 root guacamole-web  1030 Sep 21
2013 user-mapping.xml
```

User accounts need to be setup in the user-mapping.xml file, which is done by editing the file to match the following:

```
<user-mapping>
      <!-- Example user configurations are given be-
low.  For more information,
      see the user-mapping.xml section of the Guaca-
mole configuration
      documentation: http://guac-dev.org/Configur-
ing%20Guacamole -->

      <!-- Per-user authentication and config informa-
tion -->
      <authorize username="USERNAME"
password="PASSWORD">
      <protocol>vnc</protocol>
      <param name="hostname">localhost</param>
      <param name="port">5900</param>
      <param name="password">VNCPASS</param>
      </authorize>

      <!-- Another user, but using md5 to hash the
password
      (example below uses the md5 hash of the pass-
word "odroid") -->
      <authorize
              username="odroid"
              password="54e6a0bc46148912360a9f6bd82352
aa"
              encoding="md5">
      <connection name="vnc-conn">
              <protocol>vnc</protocol>
              <param name="hostname">localhost</param>
              <param name="port">5900</param>
              <param name="password">VNCPASS</param>
       </connection>
      <connection name="ssh-conn">
              <protocol>ssh</protocol>
              <param name="hostname">localhost</param>
       </connection>
      </authorize>
</user-mapping>
```

Note that this setup is coded to use the odroid user account, which matches the default Linux user account on the C1 for convenience. This account has two connection options: vnc-conn and ssh-conn, which are intended to illustrate the various connection possibilities. The vnc-port and vnc-password used by the X11VNC vnc-server setup are 5900 and VNCPASS respectively. Its password is the md5 hash equivalent of the password odroid:

```
$ echo -n odroid | md5sum
54e6a0bc46148912360a9f6bd82352aa  -
```

The guacamole login information for this user is:

```
username:     odroid
password:     odroid
```

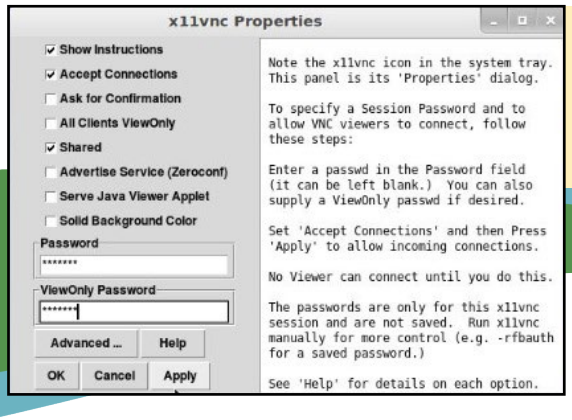
The default guacamole.properties file does not need to be altered for the setup used in this article. However, the user id that tomcat6 requires needs access to this file. After determining the tomcat6 user id, set up the file linkage using the following commands:

```
$ sudo cat /etc/passwd | grep tomcat
tomcat6:x:115:122::/usr/share/tomcat6:/bin/false

$ sudo mkdir /usr/share/tomcat6/.guacamole
$ sudo ln -s /etc/guacamole/guacamole.properties /
usr/share/tomcat6/.guacamole

Ensure that the tomcat6 server's connector configura-
tion (/etc/tomcat6/server.xml) matches the following
code snippet, then reboot:

<Connector port="8080" protocol="HTTP/1.1" connec-
tionTimeout="20000" URIEncoding="UTF-8" redirect-
Port="8443" />
```

Setup X11VNC server

Guacamole requires the setup of a supported vnc-server, so that the user may access the desktop remotely. The official C1 image already includes the X11VNC server, and no new vnc-server software needs to be installed. To create a more robust and secure system, it is advisable to disable direct system access via ports such 5900. To do so, permit direct vnc-server access only from localhost, which is illustrated in Figures 2 and 3.

X11VNC server setup

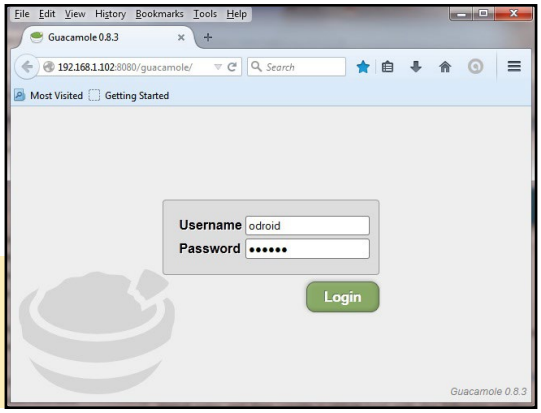


Another X11VNC server setup

The password used here, which is VNCPASS, is the same one that is specified in guacamole user settings file. Save the changes and continue to the next section to ensure that the vnc-server is running properly.

Access the desktop

The guacamole web application is set to listen on port 8080 through tomcat6. From a development machine, such as a Windows 7 desktop, launch an HTML5-capable web-browser and point it to the address http://<c1s-ip-address>:8080/guacamole. A login screen should appear, as shown at right. Enter the guacamole user account information, then click on the Login button.



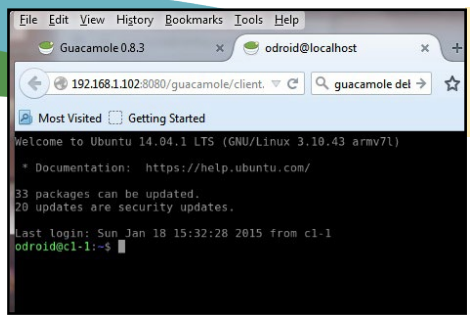
Guacamole login screen

Remote desktop connection options

Upon successful login, the home screen should be shown. Note that the application offers the two options configured earlier for this user: vnc-conn and ssh-conn.

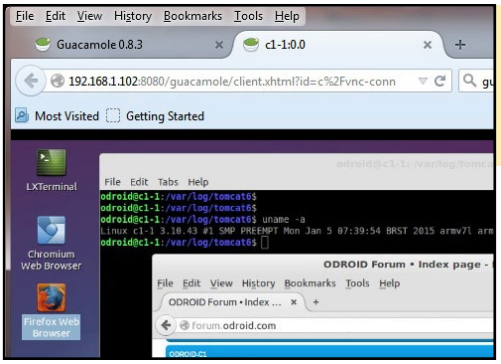
Connection options

On the very first access, the Recent Connections section will be blank, so click on the ssh-conn option first, which opens a terminal-like session on the C1. Enter the guacamole account login information and proceed with the the SSH connection, which should open in a new tab as seen below.



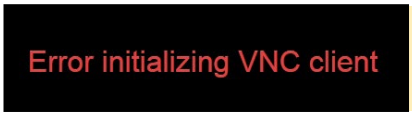
SSH connection

After experimenting with this session, close the SSH tab and go back to the original session tab. Next, click on the ssh-conn option, which will again open a new tab like below.



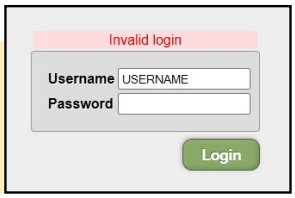
Desktop through VNC connection

If the server side vnc-server is not correctly configured, you may see a screen similar the error below. Figure 9 shows the.

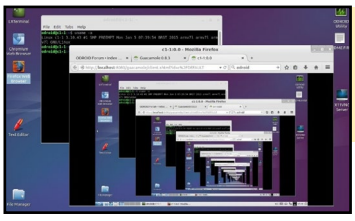


The result of a VNC server error

Error message that will result from entering the wrong user credentials



If, for some reason, you try to access the desktop from the ODROID-C1 itself, you may see an interesting phenomenon that reflects the recursive nature of the access.



Oops! An example of recursive desktop access

UBUNTU STUDIO

A UNIQUE SET OF OPEN-SOURCE MULTIMEDIA-FOCUSED TOOLS

edited by Rob Roy

Ubuntu Studio is a free and open source operating system that is packaged as an official flavor of Ubuntu, intended for creative people to produce art. It is the most widely used multimedia-oriented GNU/Linux distribution in the world, and comes pre-installed with a selection of the most common free multimedia applications available. It is free to download and use, so that you can get the source code, study it and modify it to suit your needs.

Installation

The various Ubuntu Studio packages are available via Synaptic Package Manager or the tasksel application. To install all of the available software suites, type the following into a Terminal window:

```
$ sudo apt-get install ubuntustudio-dvd-live ubuntustudio-video ubuntustudio-publishing ubuntustudio-photography ubuntustudio-font-meta ubuntustudio-audio ubuntustudio-graphics
```

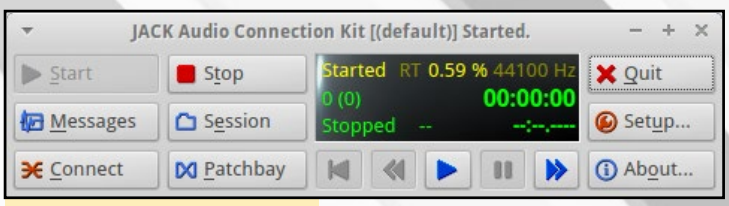
Community project

Ubuntu Studio is a community effort, created by volunteers, targeted towards all skill levels, from beginner to professional, and aims to be easily installed and simple to use, as well as providing all the tools necessary for any type of media content creation. As an officially recognized derivative of Ubuntu, Ubuntu Studio is supported by Canonical Ltd., the producers of Ubuntu, along with an amazing and continually increasing community. Ubuntu Studio is released every six months, but a long term release (LTS) version is released only every 2 years.

Audio production

Ubuntu Studio makes available some of the most popular and recently updated audio software in the Linux world, some of which are detailed below. When used with a MIDI instrument such as a keyboard, or recording devices such as a USB or standard microphone, it provides a enormous set of tools for producing high-quality music and audio creations.

Jack



QJackCtl

Jack is a low latency capable audio and MIDI server, designed for professional audio use. It enables all Jack-capable applications to connect to each other. A common program for controlling the jack server is Qjackctl, as shown above. Jack provides low latencies of less than 5ms with the right hardware, completely flexible connections, and also acts as a transport for Jack-aware applications.

Ardour

Ardour is a Digital Audio Workstation (DAW), suitable for recording, mixing and mastering. Some of its features include:

Unlimited audio tracks and buses

Non-destructive, non-linear editing with unlimited undo

Anything-to-anywhere signal routing

Unlimited pre- and post-fader plugins

32 bit floating point audio path

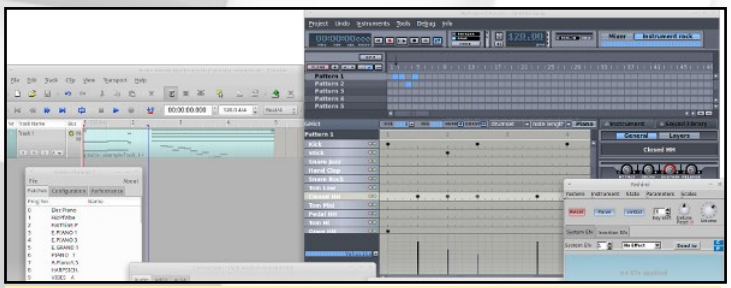
Ardour Digital Workstation

Automatic track delay compensation
Sample accurate automation
Standard file formats (BWF, WAV, WAV64, AIFF, CAF and more)
More than 200 LADSPA & LV2 plugins freely available
MIDI CC control with 1 click
Level 2 MIDI Machine Control
MIDI Timecode (MTC) Master or Slave
Full integration with all JACK applications
Video-synced playback, pull up/pull down

Sequencers and Synthesizers

Ubuntu Studio also comes installed with other notable applications such as:

Audacity, an Audio Wave Editor
Qtractor, a MIDI-capable Digital Audio Workstation
Hydrogen, a drum machine and sequencer
Yoshimi, a software-based synthesizer



Audacity, Qtractor, Hydrogen and Yoshimi

Virtual guitar amps

Rakarrack and Guitarix are two popular guitar amp simulators that let you create and use software-based amplifiers with your electric or electro-acoustic guitar.

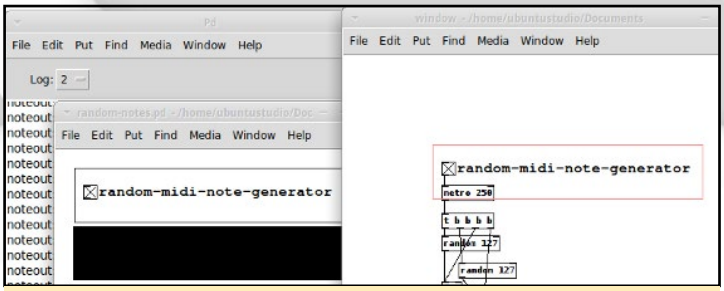
Guitarix

Gladish

Gladish, an alternative to Qjackctl, allows you to start applications, make connections between them, and save the whole configuration to a file for later use.

Audio programming

There are numerous easy-to-use audio programming environments available in Ubuntu Studio, such as Pure Data, Super Collider, Csound and Chuck. These programs can be used to create software that influences audio waveforms for complete control over the effects used in samples and recordings.

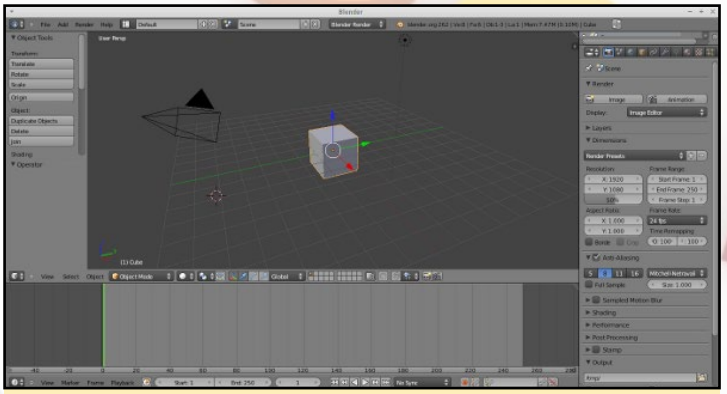


Puredata

Blender

Blender (www.blender.org) is a full-fledged 3D content creation suite, allowing you to create 3D models and animated scenes. Blender also has its own game engine, and is vastly expandable with addons. Features of Blender include:

3D Solids and character modeling
Scene animation
Physics and particle functions
Shading
Game engine (create a whole game using only Blender)
Imaging and compositing
Highly extensible

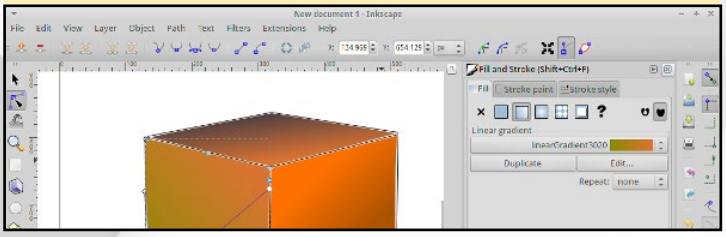


Blender

Inkscape

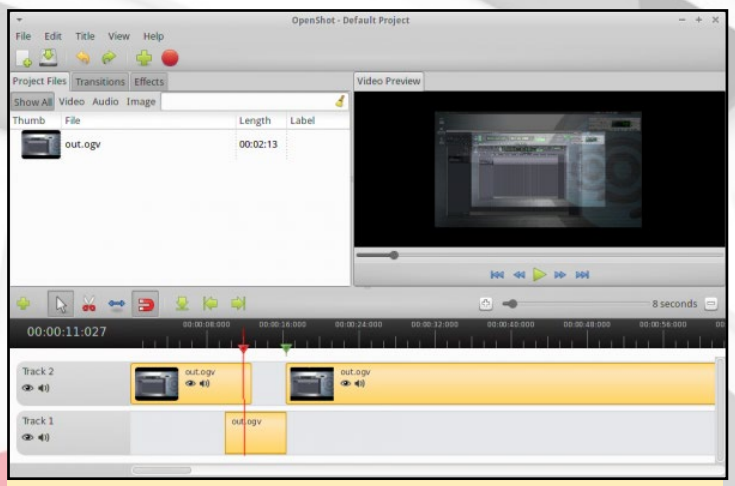
Inkscape (www.inkscape.org) is a superb vector graphics editor, with capabilities similar to Illustrator, CorelDraw, or Xara X, using the W3C standard Scalable Vector Graphics (SVG) file format.

InkScape



Openshot

Openshot (www.openshot.org) is a simple video editor for Linux, making it easy to add videos, photos and music for DVDs, youtube clips and a range of other formats.



OpenShot

FFMPEG

FFMPEG is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much any format, supporting the most obscure ancient codecs up to cutting-edge modern ones. It contains libavcodec, libavutil, libavformat, libavdevice, libswscale and libswresample, which can be used by other applications, as well as ffmpeg, ffserver, ffplay and ffprobe which can be used by end users for transcoding, streaming and playing.

DVDStyler

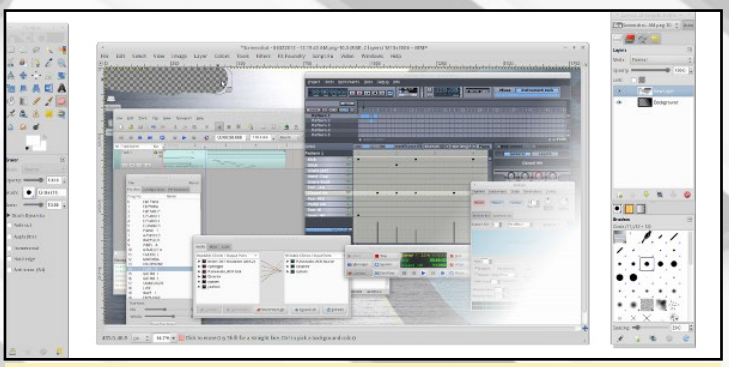
DVDStyler may be used to create custom, professional looking DVDs.

User-friendly interface with support of drag & drop
Multiple subtitles and audio tracks
Design your own DVD menu or select a template
Create a photo slide show
support of AVI, MOV, MP4, MPEG, OGG, WMV and other file formats
support of MPEG-2, MPEG-4, DivX, Xvid, MP2, MP3, AC3 and other audio and video formats

DVD Styler

GIMP

GIMP (www.gimp.org) stands for Gnu Image Manipulation Program, and is a similar to Photoshop. It is highly expandable via add-ons, and contains many features for visual effects, cropping tools, and much more.

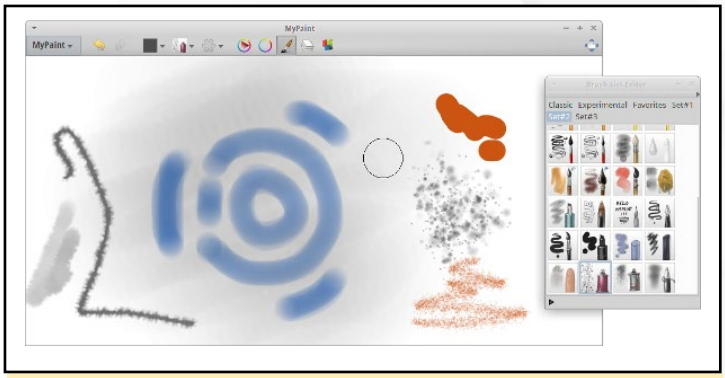


GIMP

MyPaint

MyPaint is a digital painting tool, designed to work with graphic tablets. It comes with a large collection of brushes, including ink and charcoal.

Designed for pressure sensitive graphics tablets
Simple and minimalistic user interface
Extensive brush creation and configuration options
Unlimited canvas
Basic layer support



My Paint

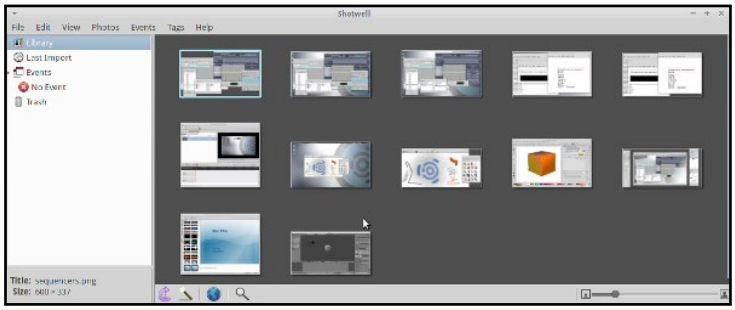
Darktable

Darktable is a photography workflow application and RAW developer, featuring a virtual light table and darkroom for photographers. It manages your digital negatives in a database, lets you view them through a zoomable light table, and enables you to develop images and enhance them.

Shotwell

You can use Shotwell to organize your library of photos, with an emphasis on keeping things simple:

Import multiple formats
Edit tags
Convert formats
Simple editing on the fly
Publish directly to social sites



Shotwell

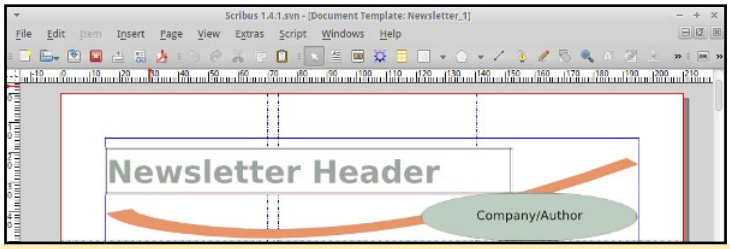
Calibre

Calibre is a popular program that makes things easy for new users by providing excellent templates for common formats, such as kindle, various types of tablets and other hardware readers, and more.

Calibre

Scribus

Another great tool for desktop publishing is Scribus, which will let you create professional PDF publications. It includes many pre-built templates for posters, business cards, brochures and more.



Scribus

LibreOffice

LibreOffice is a powerful office suite that embeds several applications similar to those found in Microsoft Office. For example, with LibreOffice Writer you can create text and save it in any format you like, including MS Office formats, as well as exporting to PDF.

PLANTS VS. ZOMBIES™ 2

CLASSICS NEVER DIE, ESPECIALLY WHEN UNDEAD

by Bruno Doiche

Plants vs. Zombies is the sort of classic that precedes the tablet/smartphone age, so it is no surprise that it would be recently improved for our amusement. Like the original, it is a freemium game, and requires lots of planning and real-time strategy in order to keep the undead from attacking your house. Enjoy defeating your endless waves of zombies!

```
https://play.google.com/store/apps/details?id=com.
ea.game.pvz2_row
```

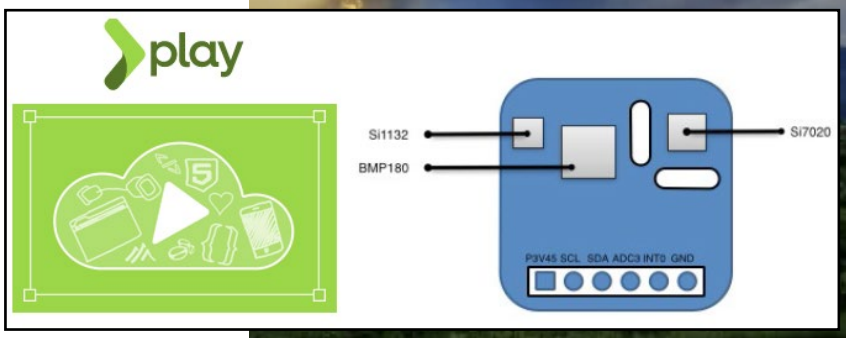
A fun travel through time and zombies galore!

PLAY WITH THE WEATHER BOARD

TECHNOLOGY FOR ALL FOUR SEASONS

by Jussi Opas

Hardkernel's Weather Board, which is an inexpensive add-on for the ODROID-SHOW peripheral, records weather phenomenon such as UV index, barometric pressure, altitude, relative humidity, illumination, and temperature. In this article, we show how to read the Weather Board sensor values using Java, as well as integrate the sensor reader into a Java-based web service, implemented with the modern Play framework. By using the web service, the weather board sensor values can be viewed from a browser page via a home network or over the Internet.



Use case

Let's assume that somewhere in the countryside, far away from home, there is a summer house or cottage. The cottage has been set up to rest over winter time, and its heating has been downgraded to save energy, to maintain dry conditions inside the cottage. To know the current condition of a cottage, one could drive a long distance, or ask a neighbor to check out the situation such as whether the heat is on, or if a window is broken. In this case, a Weather Board with sensors would help. If there is an Internet connection and continuous power available, we can set up the Weather Board hardware along with a computer to continuously measure the conditions within the cottage.

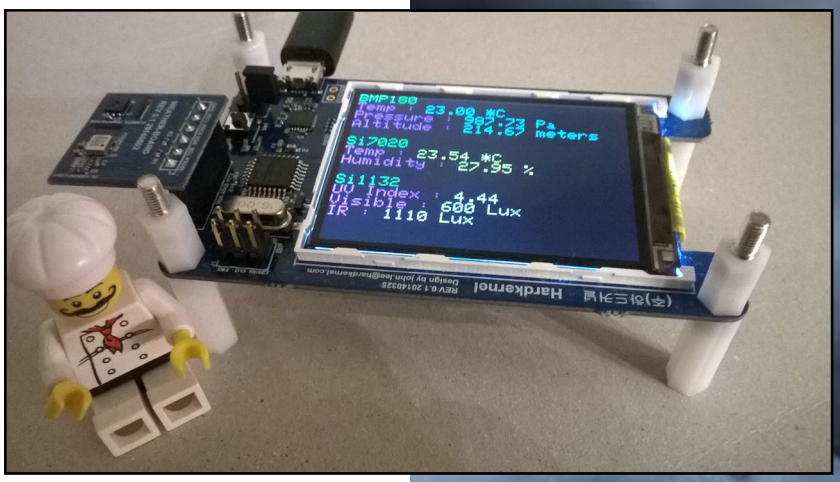
To get a measurement, we must have

1) a weather board connected to computer,
2) know how to read sensor values from the weather board, continuously or on an ad-hoc basis, and
3) implement the measurements as a service.

Initiation

The first requirement can be fulfilled by purchasing a weather board together with an ODROID-SHOW board from Hardkernel. Programming and monitoring the system using an ODROID computer would be a plus, but is not ab-

solutely necessary. Basically, any computer with Java capability and a USB port will suffice, since the SHOW can be connected universally via USB cable. The size of the Weather Board with sensors is 20x20mm, and the size of the ODROID-SHOW is 48x83mm. Here is the weather board connected to a SHOW along with a Lego chef for scale.

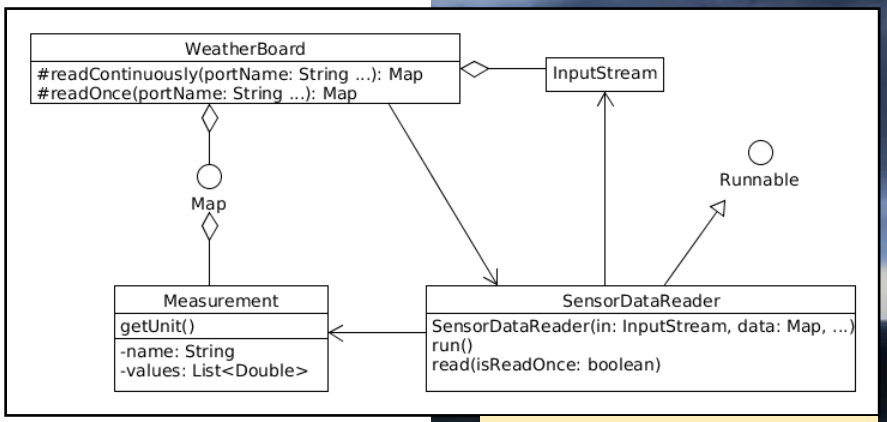


The sensors in the Weather Board are SI7020 for humidity, BMP180 for air pressure, and SI1132 for light sensing. One can easily set up the SHOW board by loading the sample software code from Hardkernel's wiki page. After loading sensor software into the microcontroller of the ODROID-SHOW board, the values are displayed on the TFT screen.

Read sensor values

The source code of the Qt-based desktop application shows how sensor values may be read from a continuous stream. However, we want to have access to the sensor values from Java. In principle, we could write a C or C++ based program that reads the stream from serial connection and then writes it into a file at appropriate intervals. Then, a Java program could read the sensor values from that same file. However, if a Java program reads the data stream directly from serial connection, then no intermediate file is needed.

We wrote a Java implementation for accessing the sensor values directly by abstracting the reading process as three Java classes: Measurement, SensorDataReader and WeatherBoard. The Measurement class represents sensor values and their treatment, value, unit and recognition as they are read from the data stream. There are 7 distinct sensor values that are being delivered by the serial port.



Weather Board model

The SensorDataReader class knows how to read input data stream correctly by interpreting the delimiter between sensor values within the data stream, and deciding when to stop reading. Last, but not least, is the WeatherBoard class that implements a method for opening the serial port for reading, and delivering the result as a map of measurements. The Weather

Board class can invoke reading as an ad-hoc operation or as a continuous thread. Although the abstraction has the capability of producing a continuous stream of sensor values, we have only been using the ad-hoc read methods. Sample source code for reading serial port with Java are available from the ODROID forums at http://bit.ly/1GsQKw8.

Hardkernel's pre-built Lubuntu images already have Java installed. They also have a library that permits binding input to serial port. The two RX*.jar files are located in /usr/share/java folder, and the respective native libraries are located as /usr/lib/jni/librxtx*.so files. For those operating systems that don't have these installed yet, download and install them with the following command:

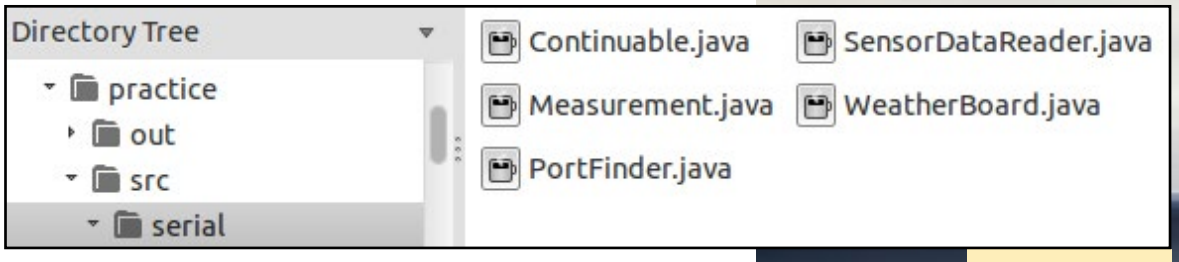
```
$ sudo apt-get install librxtx-java
```

Another alternative is to download a Java to serial port binding from the Debian repositories. After extraction, everyone can locate libraries in accordance to own preferences, as shown at http://bit.ly/15wqqke. During development we used this method.

Before running the script, each platform (ARM, X86, Linux and Windows) must have correctly compiled native libraries available. They must be properly referenced by command line compilation, or within an Interactive Development Environment (IDE) tool. With an IDE, we added "-Djava.library.path=/usr/lib/jni" as an option to the virtual machine so that native libraries were used at run time. The Java abstraction and implementation may be useful with other Java service containers as well.

Command line

We stored the Java source code into the practice folder using the command line, as shown in Figure 4.



Practice Folder

We also created the practice/out directory where compiled Java classes will be located. First, change to the practice directory:

```
$ cd practice
```

Then, create a file called Manifest.txt file inside the practice directory. The content should be as follows:

```
Main-Class: serial.WeatherBoard
```

Next, write javac commands into build.sh:

```
#!/bin/bash
javac src/serial/*.java -d ./out -cp /usr/share/java/
RXTXcomm.jar:/usr/share/RXTXcomm-2.2pre2.jar
```

Finally, write jar packaging command into pack.sh:

```
#!/bin/bash
clear
echo "make wboard.jar"
current=.
cd out
jarfile=../wboard.jar
classes=./serial
jar cmf ../Manifest.txt $jarfile $classes/*
cd $current
```

Now, one can compile and package the source code into the wboard.jar file:

```
$ sh build.sh
$ sh pack.sh
```

To be able to access the serial port, users must have the correct privileges. In Debian, the command "sudo adduser <user> dialout" adds a user to the dialout group. To make the change effective, one must run the newgrp command, or logout and login again.

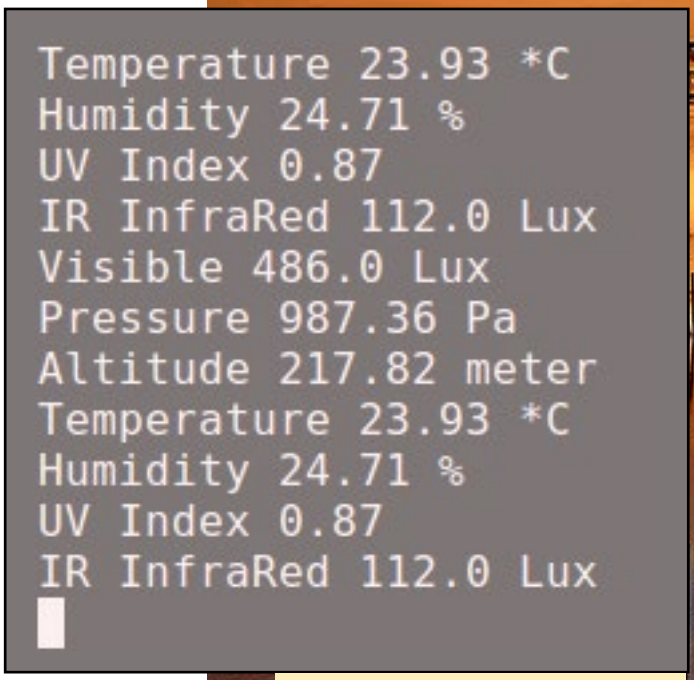
The last file to write is run.sh, which links the self-made wboard.jar, the installed RXTX*.jar libraries and native libraries in /usr/lib/jni, then invokes serial port reading:

```
#!/bin/bash
java -cp wboard.jar:/usr/share/java/
RXTXcomm.jar:/usr/share/java/RXTXcomm-
2.2pre2.jar -Djava.library.path=/usr/
lib/jni serial.WeatherBoard
```

One can then invoke continuous serial port monitoring:

```
$ sh run.sh
```

The Terminal window will show a continuous flow of sensor values, as demonstrated in here.



Sensor values in Terminal window

Play framework

Play is a modern development and deployment friendly, non-JEE based Java service container. The services may be programmed with Java and/or Scala, and to develop web pages and their layouts, one must know also HTML and CSS. Play may be downloaded from the home pages of the Play framework at http://bit.ly/1uz0UU0. We used Play version 2.2.2:

```
$ cd ~ && mkdir Applications && cd Applications
$ wget http://downloads.typesafe.com/play/play-
2.2.2.zip
```

If wget does not work, then just visit the Play framework's download pages and click on the appropriate link. After downloading is completed, unzip the file:

```
$ sudo apt-get install p7zip-full
$ 7z x play-2.2.2.zip
```

For development purposes, we extracted Play into the ~/Applications folder, then added the "play" command to the PATH variable, which may be done by opening the file ~/.profile in a text editor and adding the following lines:

```
if [ -d "$HOME/Applications/play-2.2.2" ] ; then
    PATH="$HOME/Applications/play-2.2.2:$PATH"
fi
```

The definition can be immediately used by running the "source ~/.profile" command. Start a sample application by typing the following:

```
$ cd ~/Applications/play-2.2.2/samples/java/hello-
world
$ play
```

Play will now start, showing a Play prompt. Next, start the application:

```
[helloworld] $run
```

The service will then be ready to go. Open a browser and navigate to "http://localhost:9000", which will show the sample web application. Memory usage of the Play service may be defined in the last line of the file ../play-2.2.2/framework/build, using values similar to -Xms128M -Xmx256M or -Xms32M -Xmx64M. To create a new project, type the following command:

```
play new weather
```

This creates a new Play project with the name weather. Next, invoke the newly created project:

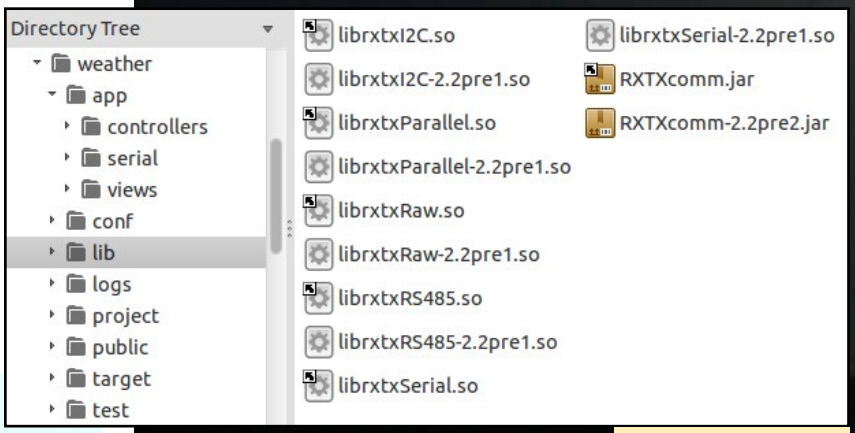
```
cd weather
play
run
```

Play application and web pages can be developed without an IDE, because the application will be compiled when any of the files have changed and the browser page is refreshed. Any errors are displayed in the browser page as well.

Libraries

External Java libraries are automatically compiled into the Play application, as long as they are located into the weather application's lib folder. In our case, it means that .jar files are located in the /weather/lib directory, which contains the RXTXcomm.jar and RXTXcomm-2.2pre2.jar files. We placed also the ARM specific native .so files in the same folder.

However, only the .jar files are included with this method. We defined additionally explicit loading of the native libraries by creating a Global.java file in the /weather/app/controllers folder:



Added libraries

```
import play.*;
public class Global extends GlobalSettings {
      @Override
      public void beforeStart(Application app) {
              super.beforeStart(app);
                 ...
                 System.load("/home/odroid/Applica-
tions/weather/lib/librxtxSerial.so");
                 System.load("/home/odroid/Applica-
tions/weather/lib/librxtxSerial-2.2pre1.so");
                 ...
        }
}
```

We must also include the library path when invoking the Play service:

```
$ vi invoke.sh
```

Insert the following instructions into the file:

```
play -Djava.library.path=$HOME/Applications/weather/
lib start
```

Then, launch the application:

```
$ chmod +x invoke.sh
$ ./invoke.sh
```

Play Application

The weather/conf/routes file contains the following definitions:

```
GET     /weather  controllers.Application.getWeather()
GET     /refresh  controllers.Application.refreshWeather()
```

Two web pages are defined. The first one is consistent with the concept of the Play framework, as shown in the file /view/index.scala.html:

```
@(weatherForm: Form[Application.Weather])
@import helper._
@main(title = "Play with Weather Board") {
    <h1>Sensors</h1>
    <ul>
        <li>Si7020, humidity</li>
        <li>SI1132, ambient light</li>
        <li>BMP180, air pressure</li>
    </ul>
    @form(action = routes.Application.getWeather, args = 'id -> "weatherform") {
        @inputText(
            weatherForm("name").copy(value=Some("/dev/ttyUSB0")),
            args = '_label -> "Port name", 'size -> 9
        )
        @inputText(
            weatherForm("time").copy(value=Some("5")),
            args = '_label -> "First read time as sec?", 'size -> 3
        )
        <p class="buttons">
            <input type="submit" value="Get values">
        <p>
    }
}
```

There are two fields defined, but the name of the port cannot be edited. However, the maximum sensor read time can be edited. The definition of the result page (/view/weather.scala.html) is still more succinct:

```
@(name: String, time: Int, measurements: List[String])
```

```
@main("Play with Weather Board") {
    <h1>Sensors' values</h1>
    <ul>
        @for(measurement <- measurements) {
            <li>@measurement</li>
        }
    </ul>
    <p class="buttons">
        <a href="@routes.Application.index">Back to
sensors page</a>
        <a href="@routes.Application.
refreshWeather">Refresh</a>
    </p>
}
```

This definition is automatically transformed to an invocation of Weather Board reading by the Play framework. Below is our implementation of the weather sensor service:

```
public class Application extends Controller {
    public static Weather data;

    public static class Weather {
        public String name;
        @Min(1) @Max(60) public Integer time;
        public List<String> measurements;
    }

    public static Result index() {
        return ok(index.render(form(Weather.class)));
    }

    public static Result getWeather() {
        Form<Weather> form = form(Weather.class).
bindFromRequest();
        if (form.hasErrors()) {
            return badRequest(index.render(form));
        } else {
            data = form.get();
            try {
                data.measurements =
readMeasurements(data.name, data.time, 0);
            } catch (Exception e, data.name) {
                return handleErrors(e);
            }
            return ok(weather.render(data.name, data.
time, data.measurements));
        }
    }

    public static Result refreshWeather() throws Ex-
```

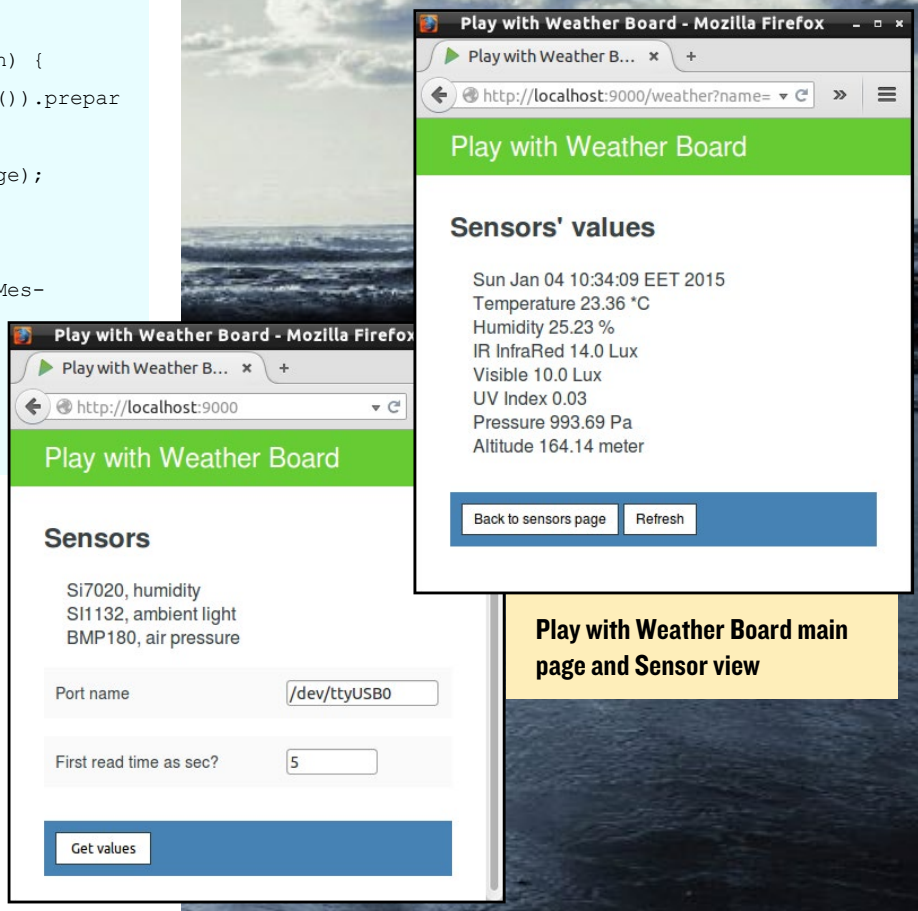
```
ception {
        List<String> measurements =
readMeasurements(data.name, data.time, 0);
        return ok(weather.render(data.name, data.
time, measurements));
    }

    static List<String> readMeasurements(String port,
int time, int logLevel) throws Exception {
         WeatherBoard board = new WeatherBoard();
         return transform(board.
readContinuously(port, time, logLevel));
    }

    private static List<String> transform(Map<String,
Measurement> data) {
        List<String> measurements = new
ArrayList<String>();
      String today = (new Date()).toString();
      measurements.add(today);
        for (Measurement measurement: data.values())
{
            measurements.add("" + measurement);
        }
        return measurements;
    }

    private static Result handleErrors(Exception e,
String portName) {
        if (e instanceof NoSuchPortException) {
           String message = (new PortFinder()).prepar
ePortErrorMessage(portName);
           return internalServerError(message);
        } else {
           e.printStackTrace();
           return internalServerError(e.getMes-
sage());
        }
    }
}
```

The style sheet is based on the Play's hello-world sample project and is not included here.

Web pages

The resulting web pages are shown here, with the main page on the left.

Pressing the Get values button invokes continuous reading of sensor values. The results of the read are shown in the sensors' values page. The page may also be refreshed to show the most



Play with Weather Board main page and Sensor view

recent values. In the case that the port is not available, an error message will be shown:

```
Port /dev/ttyUSB0 was not found, please try another.
Available ports are:
/dev/ttyUSB1
/dev/ttyACM99
```

With this information, one can then type /dev/ttyUSB1 into the main page and get sensor values again.

Conclusion

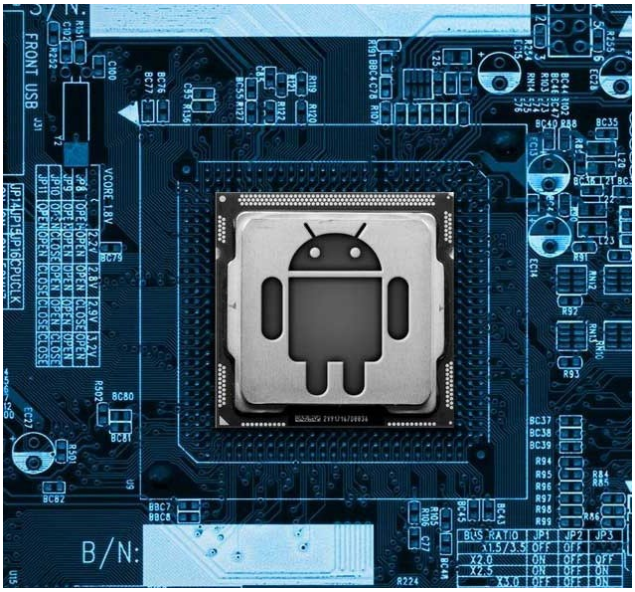
Once the steps are completed, all family members can inspect the weather in the cottage from their home locations as long as they are connected via Ethernet. However, only one reader can consume the output stream from sensors at a time, since the Qt and C++ based desktop application cannot get values while a threaded Java reader is consuming the data stream from serial port. The Java reader program reads 7 measurement values, then stops. Although, it possible that 7 values can never be read, because there will be not that many measurements available. For instance, at night time, when lights are off, there will be no value available for visible light. Tuning of this time limit for first time invocation is also possible via the web page.

If several users were aimed to be served in a responsive way, then the service should itself maintain a file or database of the latest values in order to give a rapid response based on that. We didn't store values into files, and instead, the server holds a static state that stores the latest sensor values. From the static state, each request can be served as a non-blocking response. The service is so small that it can be deployed with any ODROID that has Java available and native libraries for reading serial port, even the inexpensive C1 model. For development purposes, it is faster to use a ODROID-U3 or ODROID-XU3 instead.

ANDROID DEVELOPMENT

DEVICE CONFIGURATION

by Nanik Tolaram



You may be familiar with the many different Android devices that are currently available, with enormous disparity in hardware such as screen size, available peripherals, and intended purpose such as mobile phones and automotive installations. Android runs nearly anywhere and everywhere, but you may be wondering how it is possible that it can function on so many different hardware configurations, processors, while using a variety of sensors and inputs. Every day, we read about new devices that have been released with supported peripherals from previously unknown vendors. How is it possible to integrate so many different kind of hardware inside Android devices? Most of the integration is due to the power of Linux, since it's a mature ecosystem that allows hardware vendors to create their own product with their own drivers. However, there are still many vendors that do not want to release the code for their software drivers as open source projects, so those drivers are packaged as binary files, which are referred to as Binary Large Objects (BLOBs) in Linux vernacular.

Different devices requires separate configuration and settings, which are necessary in order to be packaged as part of the Android build process. Android is very flexible when it comes to device configuration, but there are certain rules that need to be followed in order to ensure that there is consistency between devices. The process behind the rules and the integration of different drivers during the build process is beyond the scope of this article, and will be discussed in future installments. For now, we will only look at how devices are being configured inside Android and how it is relevant to the build process, using the ODROID-U3 running Android Kitkat 4.4.4 as an example.

device/

The standard way for Android to find device configurations is to look inside the device/ folder as shown in Figure 1. ODROID boards configuration are found inside device/hardkernel/.

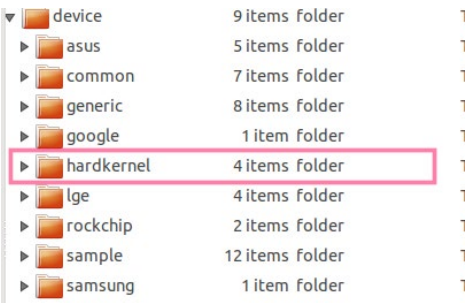


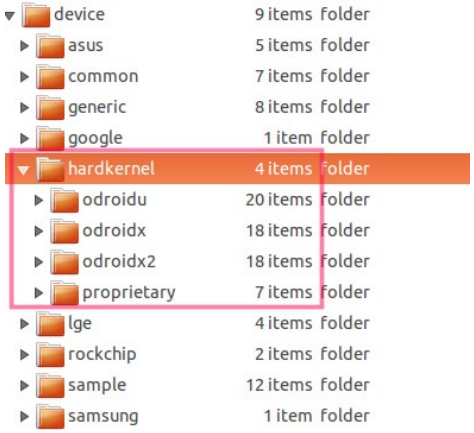
Figure 1 : Hardkernel board configuration

Along with the ODROID boards, you can also find configurations for Nexus-based devices that come standard with AOSP, such as the Google Nexus Tablet 7 2012 (grouper) and the Nexus 7 2013 (flo). Figure 2 shows the details of what is inside the hardkernel directory, which contains a few of Hardkernel's boards: the ODROID-U (ODROID-U3) , ODROID-X and ODROID-X2. Inside these directories you can find drivers, configuration and build scripts that are specific to ODROIDs. In Figure 3, you can see the contents of the odroidu/ directory containing the configuration details for the ODROID-U3 board. We will take a look at each of the files and directories in the following sections.

bluetooth/

The bluetooth directory contains a single file called bdroid_buildcfg.h, which is basically used as part of Android's Bluedroid bluetooth stack. This file contains many configuration options such as the device name, and the kind of bluetooth support that the device offers.

Figure 2 : ODROID board configuration



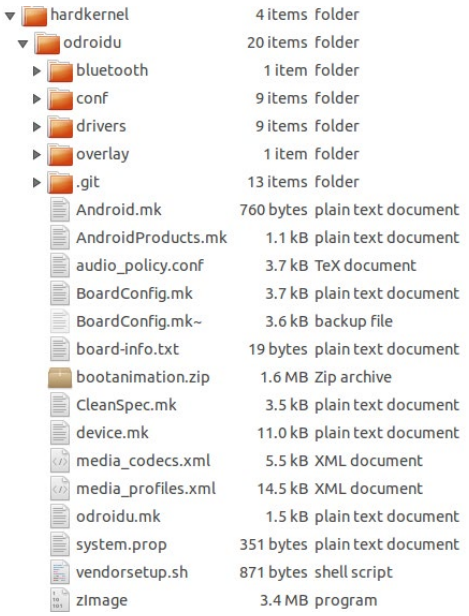


Figure 3 : ODROID-U3 configuration

conf/

The conf directory contains configuration files such as codecs, fstab (block device), the .rc file, and the ueventd file, as shown in Figure 4.

drivers/

Binary drivers that are needed for peripherals such as WiFi, Ethernet and Audio can be found here. These files are copied during the build process to /system/lib/modules/.

Figure 4 : conf directory

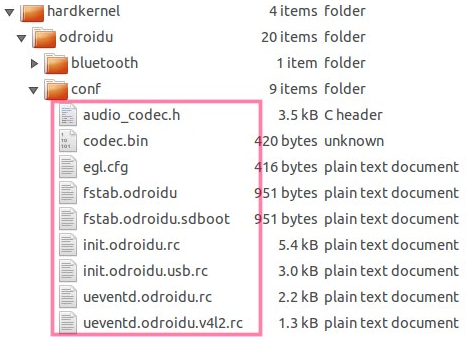
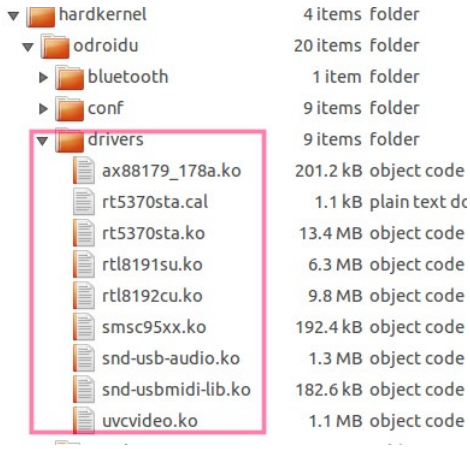


Figure 5 : drivers directory



overlay/

This directory contains configurations that exist inside the system applications which may need to be modified by the user. As can be seen in Figure 6, there are a few .xml files inside the res/ folder, and if we take a look at one of the files such as power_profile.xml, you will see the following configuration:

```
<item name="bluetooth.active">10</item> <!-- Bluetooth data transfer, ~10mA -->
  <item name="bluetooth.on">0.1</item>  <!-- Bluetooth on & connectable, but not connected, ~0.1mA -->
  <item name="wifi.on">3</item> <!-- ~3mA -->
```

The above configuration (bluetooth.active, bluetooth.on and wifi.on) will be used to replace the original content inside the framework/base/core/res/res/xml/power_profile.xml file.

proprietary/

This particular directory is a very interesting, since it contains many proprietary files that are packaged together as part of the Android image. The first directory is the apk/ folder which contains application .apk files. You will see the apk shown in Figure 7 when you build your own Android image files, or when you download the pre-built one from

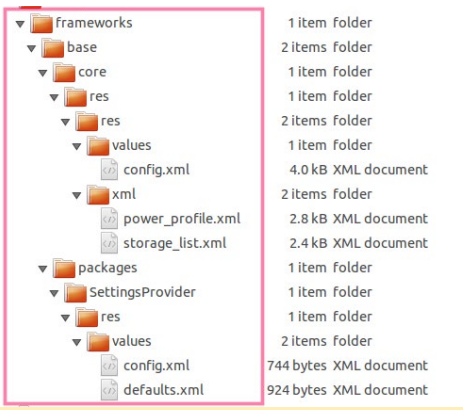


Figure 6 : overlay directory

http://bit.ly/1xkxreJ. The file that instructs the build process to include these apks can be seen inside device/hardkernel/odroidu/device.mk as shown in Figure 8a and 8b.

The propietary/bin directory contains files that are relevant to touch and keyboard devices. One of the files is called Vendor_2808_Product_81c9.idc, which correspond to the ODROID-VU touch screen device, as detailed in the August 2014 ODROID Magazine on page 30. Most of the files inside this directory

Figure 7 : apk folder

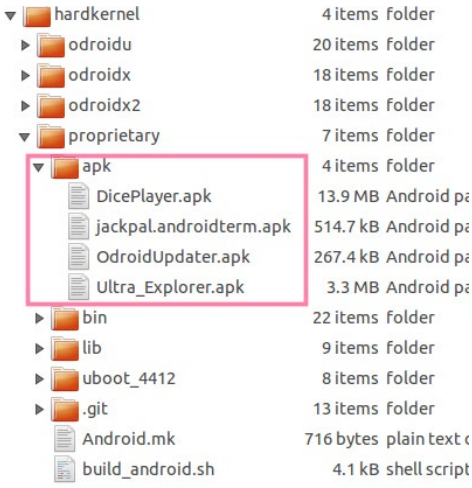


Figure 8a: device.mk

```
PRODUCT_COPY_FILES += \
    device/hardkernel/proprietary/apk/DicePlayer.apk:system/app/DicePlayer.apk \
    device/hardkernel/proprietary/lib/libSoundTouch.so:system/lib/libSoundTouch.so \
    device/hardkernel/proprietary/lib/libdice_kk.so:system/lib/libdice_kk.so \
    device/hardkernel/proprietary/lib/libdice_loadlibrary.so:system/lib/libdice_loadlibrary.so \
    device/hardkernel/proprietary/lib/libdice_software.so:system/lib/libdice_software.so \
    device/hardkernel/proprietary/lib/libdice_software_kk.so:system/lib/libdice_software_kk.so \
    device/hardkernel/proprietary/lib/libffmpeg_dice.so:system/lib/libffmpeg_dice.so \
    device/hardkernel/proprietary/lib/libsonic.so:system/lib/libsonic.so
```

Figure 8b: device.mk

```
PRODUCT_COPY_FILES += \
    device/hardkernel/proprietary/apk/Ultra_Explorer.apk:system/app/Ultra_Explorer.apk \
    device/hardkernel/proprietary/apk/jackpal.androidterm.apk:system/app/jackpal.androidterm.apk \
    device/hardkernel/proprietary/lib/libjackpal-androidterm4.so:system/lib/libjackpal-androidterm4.so
```

```
touch.deviceType = touchScreen
touch.orientationAware = 1

device.internal = 1

keyboard.layout = qwerty
keyboard.characterMap = qwerty2
keyboard.orientationAware = 1
keyboard.builtIn = 1

cursor.mode = navigation
cursor.orientationAware = 1
```

Figure 9 : Vendor_2808_Product_81c9.idc

are copied into the image file, and are subsequently used by the Android input subsystem in order to understand the configuration of the touch devices that are available for use. Figure 6 shows the content of the ODROID-VU touch device configuration file.

The proprietary/lib folder contains a number of .so library files that are used by .apk files, as shown in Figure 10. The last directory, proprietary/uboot_4412, is important because it contains the proprietary Samsung bootloader and ODROID uboot binary. Without these files, the board would not be able to boot up! The proprietary bootloader source code is not available, but the ODROID uboot code may be downloaded from http://bit.ly/1ydj3cb, as seen in Figure 11. The other file, called zImage, is the pre-built kernel image, created during the build process, that is also used for booting up the board.

Figure 10 : .so library files

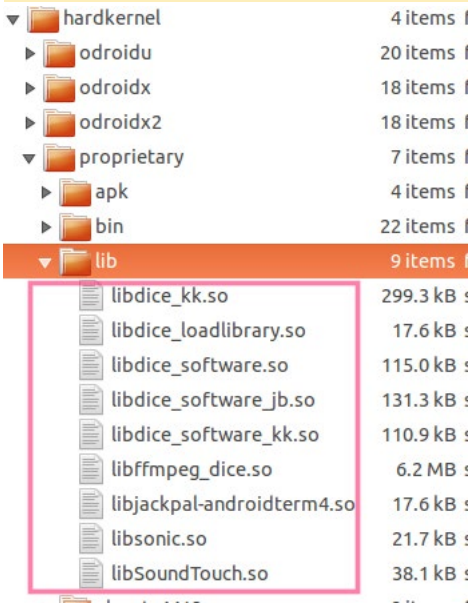
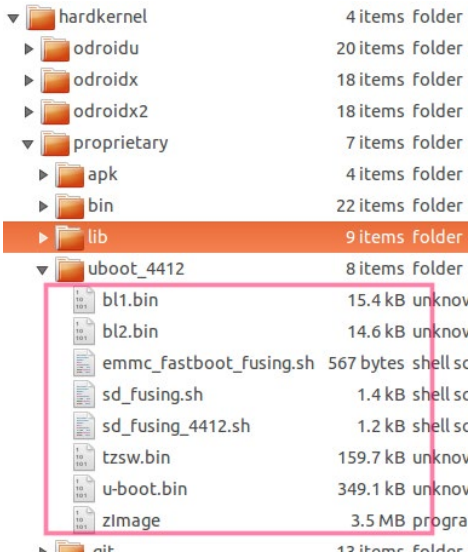


Figure 11 : bootloader files

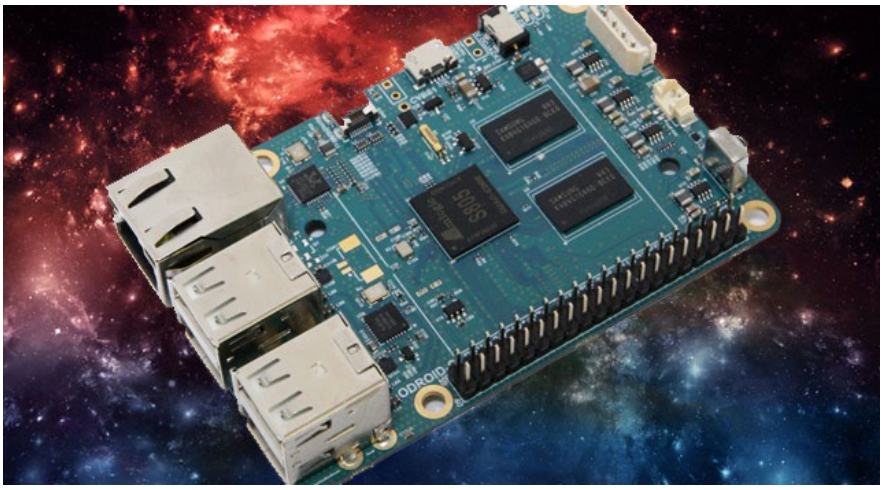


Android's device configuration architecture is designed to work many types of peripherals

ODROID-C1 MINIMAL INSTALL

GET BACK TO THE BASICS

by segfault@kill-9.me



My Odroid C1 is equipped with an 8GB eMMC. However, there are no "console" versions yet of any the operating systems provided by Hardkernel. The only ones available at the time of this writing have all of the GUI software installed, which for my purpose is unnecessary. This means that nearly 4GB of disk space is used out of the box. With some trial and error, I managed to strip out everything that I don't need for my Odroid C1 and condensed it into a few lines of code. There's also a few caveats to the eMMC images as well, some of which are security issues:

1. SSH Server Keys are not regenerated upon first boot.
2. udev rules for network devices aren't purged before the image is created. This causes the main ethernet interface to get listed as eth1 instead of eth0.
3. NetworkManager is installed. You'll need to edit /etc/network/interfaces, then disable NetworkManager.

To begin, login as root using the "su" command before following the steps below. It's recommended to copy-and-paste the commands!

SSH keys

```
# dpkg-reconfigure openssh-server
```

Fix udev

```
# echo > /etc/udev/rules.d/70-persistent-net.rules
# reboot
```

Remove Network-Manager

```
# echo >> /etc/network/interfaces
# echo auto eth0 >> /etc/network/interfaces
# echo iface eth0 inet dhcp >> /etc/network/interfaces
# stop network-manager
# echo "manual" | tee /etc/init/network-manager.override
# reboot
```

Strip out X11 packages

```
# apt-get remove abiword abiword-common abiword-plugin-grammar abiword-plugin-mathview ac-countsservice acl alsa-base alsa-utils anthy anthy-common apport apport-gtk apport-symptoms aria2 aspell audacious audacious-plugins:armhf audacious-plugins-data autopoint axel bind9-host blueman bluez bluez-alsa:armhf bluez-cups camorama chromium-browser chromium-browser-l10n chromium-codecs-ffmpeg-extra colord cups cups-browsed cups-bsd cups-client cups-common cups-core-drivers cups-daemon cups-driver-gutenprint cups-filters cups-filters-core-drivers cups-ppdc cups-server-common deadbeef dmz-cursor-theme docbook docbook-dsssl docbook-to-man docbook-xml docbook-xsl evince evince-common evolution-data-server-common extra-xdg-menus faenza-icon-theme fbset ffmpegthumbnailer filezilla filezilla-common file-roller flite1-dev:armhf fontconfig font-config-config fonts-arabeyes fonts-arphic-ukai fonts-arphic-uming fonts-dejavu fonts-dejavu-core fonts-dejavu-extra fonts-droid fonts-farsiweb fonts-freefont-ttf fonts-kacst fonts-kacst-one fonts-khmeros fonts-khmeros-core fonts-lao fonts-liberation fonts-lklug-sinhala fonts-lyx fonts-manchufont fonts-mgopen fonts-nafees fonts-nanum fonts-nanum-coding fonts-opensymbol fonts-sil-abyssinica fonts-sil-ezra fonts-sil-gentium fonts-sil-gentium-basic fonts-sil-padauk fonts-sil-scheherazade fonts-takao-gothic fonts-takao-mincho fonts-takao-pgothic fonts-thai-tlwg fonts-tibetan-machine fonts-tlwg-garuda fonts-
```

tlwg-kinnari fonts-tlwg-loma fonts-tlwg-mono fonts-tlwg-norasi fonts-tlwg-purisa fonts-tlwg-sawasdee fonts-tlwg-typewriter fonts-tlwg-typist fonts-tlwg-typo fonts-tlwg-umpush fonts-tlwg-waree fonts-ukij-uyghur fonts-unfonts-core foomatic-db-compressed-ppds fuse libfontconfig1 libgtk2.0.0 libpango-1.0.0 libqt5widgets5 snappy transmission-qt transmission-cli foomatic-filters gir1.2* zenity-common yelp-xsl yasm xvfb xtrans-dev xsltproc xserver-common xscreensaver-data xscreensaver-screensaver-bsod x11-common xfconf xfce4-dev-tools xfce4-power-manager-data xdg-utils xauth xbitmaps xdg-user-dirs xkb-data xorg-docs-core xorg-sgml-doctools autoconf autoconf2.13 automake autotools-dev avr-libc avrdude binutils binutils-avr build-essential ccache cdbs cmake cmake-data comerr-dev command-not-found-data cpp cpp-4.8 desktop-file-utils dpkg-dev fakeroot firefox-locale-en flex bison g++ g++-4.8 gcc gcc-4.8 gcc-avr gconf-service gconf-service-backend gconf2 gconf2-common gdb gdebi-core genisoimage giblib1:armhf glmark2-data glmark2-es2 gnome-accessibility-themes gnome-desktop-data gnome-desktop3-data gnome-menus gnome-panel-data gnome-pkg-tools gnome-themes-standard-data gnumeric-common gobject-introspection gstreamer0.10-nice:armhf gstreamer0.10-plugins-base:armhf gsfonts gsettings-desktop-schemas gstreamer0.10-plugins-base:armhf gstreamer1.0-alsa:armhf gstreamer1.0-doc gstreamer1.0-libav:armhf gstreamer1.0-plugins-bad-doc gstreamer1.0-plugins-base:armhf gstreamer1.0-plugins-base-apps gstreamer1.0-plugins-base-doc gstreamer1.0-plugins-good-doc gstreamer1.0-pulseaudio:armhf gstreamer1.0-tools gvfs-common gvfs-libs:armhf hicolor-icon-theme hplip-data hunspell-en-us imagemagick-common java-common jade joe kerneloops-daemon ladspa-sdk laptop-detect lightdm link-grammar-dictionaries-en lintian linux-sound-base lubuntu-lxpanel-icons openjdk-7-jre ca-certificates-java aspell aspell-en fonts-dejavu-core lxmenu-data lxsession-data m4 make maliddx mc mc-data mesa-utils mesa-utils-extra metacity-common mircommon-dev:armhf mobile-broadband-provider-info modemmanager mysql-common nautilus-data netpbm obex-data-server openprinting-ppds p11-kit p11-kit-modules:armhf pastebinit pcmciautils pidgin-data policykit-desktop-privileges poppler-data printer-driver-c2esp printer-driver-foo2zjs-common printer-driver-min12xxw pulseaudio python-cups python-cupshelpers qpdf quilt rfkill samba-common samba-common-bin samba-libs:armhf sgml-base sgml-data sgmlspl smbclient sound-theme-freedesktop swig swig2.0 sylpheed-doc system-config-printer-common system-config-printer-udev t1utils transmission-common tsconf ttf-bengali-fonts ttf-devanagari-fonts ttf-gujarati-fonts ttf-indic-fonts-core ttf-kannada-fonts ttf-malayalam-fonts ttf-oriya-fonts ttf-punjabi-fonts ttf-tamil-fonts ttf-telugu-fonts ttf-ubuntu-font-family usbmuxd uvcdynctrl uvcdynctrl-data valgrind whoopsie wireless-tools wpasupplicant wvdial x11-xfs-utils xbmc xinput xserver-xorg-core xfce4-power-manager xfonts-100dpi xfonts-base xfonts-mathml xfonts-scalable xfonts-utils xinit xdg-user-dirs-gtk xdg-user-dirs xarchiver x11proto-xinerama-dev x11proto-xf86vidmode-dev x11proto-xf86dri-dev x11proto-xcmisc-dev x11proto-video-dev x11proto-record-dev x11proto-randr-dev x11proto-present-dev x11proto-kb-dev x11proto-dri3-dev x11proto-dri2-dev x11proto-dmx-dev x11proto-bigreqs-dev dictionaries-common libavc1394-0:armhf libavresample1:armhf libavutil52:armhf libbluetooth3:armhf libbluray1:armhf libbonobo2-common libbonoboui2-common libboost-atomic1.54.0:armhf libboost-chrono1.54.0:armhf libboost-date-time1.54.0:armhf libboost-serialization1.54.0:armhf libboost-system1.54.0:armhf libboost-thread1.54.0:armhf libbs2b0 libburn4 libcaca0:armhf libcamel-1.2-45 libcdaudio1 libcddb2 libcdio-cdda1 libcdio-paranoia1 libcdio13 libcdparanoia0:armhf libcdt5 libcec libcgraph6 libcogl15:armhf libcolamd2.8.0:armhf libcolord1:armhf libcolorhug1:armhf libcompfaceg1 libcrack2:armhf libcroco3:armhf libcue1 libcups2:armhf libcupscgi1:armhf libcupsfilters1:armhf libcupsimage2:armhf libcupsmime1:armhf libcupsppdc1:armhf libdatrie1:armhf libdc1394-22:armhf libdca0:armhf libdirac-decoder0:armhf libdirac-encoder0:armhf libdiscid0:armhf libdjvulibre-text libdjvulibre21:armhf libdmx1:armhf libdrm-nouveau2:armhf libdrm-radeon1:armhf libdv4:armhf libdvdnav4:armhf libdvdread4:armhf libegl1-mesa:armhf libdrm-omap1:armhf

libegl1-mesa-drivers:armhf libexo-common libexo-helpers libfaad2:armhf libfakeroot:armhf libfftw3-bin libfftw3-double3:armhf libfftw3-single3:armhf libflac8:armhf libfontembed1:armhf libfontenc1:armhf libframe6:armhf libfreetype6:armhf libfribidi0:armhf libfs6:armhf libftdi1:armhf libfuse2:armhf libgbm1:armhf libgck-1-0:armhf libgcr-3-common libgcr-base-3-1:armhf libgda-5.0-common libgdk-pixbuf2.0-0:armhf libgdk-pixbuf2.0-common libgdome2-0 libgdome2-cpp-smart0c2a libgeis1:armhf libgeoclue0:armhf libgeoip1:armhf libgif4:armhf libgirepository-1.0-1 libgl1-mesa-dri:armhf libgl1-mesa-glx:armhf libglapi-mesa:armhf libgles1-mesa:armhf libgles2-mesa:armhf libglib2.0-doc libglib-perl libgme0 libgmpxx4ldbl:armhf libgnome-keyring-common libgnome-keyring0:armhf libgnome-menu-3-0 libgnomecanvas2-common libgnomeui-common libgoffice-0.10-10-common libgomp1:armhf libgphoto2-port10:armhf libgraphite2-3:armhf libgs9-common libgsf-1-114 libgsf-1-common libgsl0ldbl libgsm1:armhf libgstreamer-plugins-base0.10-0:armhf libgstreamer-plugins-base1.0-0:armhf libgstreamer0.10-0:armhf libgstreamer1.0-0:armhf libgtk-3-common libgtk2.0-common libgtop2-7 libgtop2-common libgudev-1.0-0:armhf libguess1:armhf libgusb2:armhf libgutenprint2 libgweather-common libhogweed2:armhf libhpmud0 libhunspell-1.3-0:armhf libibus-1.0-5:armhf libical1 libid3tag0 libieee1284-3:armhf libijs-0.35 libilmbase6:armhf libimage-exiftool-perl libiptcdata0 libisofs6 libiw30:armhf

libjack-jackd2-0:armhf libjasper1:armhf libjavascriptcoregtk-3.0-0:armhf libjbig0:armhf libjbig2dec0 libjna-java libjpeg-turbo8:armhf libjpeg8:armhf libjs-jquery libjte1 libkate1 liblavjpeg-2.1-0 liblcms2-2:armhf liblircclient0 libllvm3.4:armhf liblockfile-bin liblockfile1:armhf libloudmouth1-0 liblqr-1-0:armhf libltdl7:armhf liblua5.2-0:armhf libmad0:armhf libmbim-glib0:armhf libmeanwhile1 libmenu-cache-bin libmenu-cache3 libmessaging-menu0 libmicrohttpd10 libmikmod2:armhf libmimic0 libmirprotobuf0:armhf libmjpegutils-2.1-0 libmms0:armhf libmodplug1 libmp3lame0:armhf libmpcdec6 libmpeg2-4:armhf libmpeg2encpp-2.1-0 libmpg123-0:armhf libmplex2-2.1-0 libmtdev1:armhf libmtp-common libmtp-runtime libmtp9:armhf libnetpbm10 libnettle4:armhf libobt2 libogg0:armhf libopenaldata libopenal1:armhf libopencvcalib3d2.4:armhf libopencvcore2.4:armhf libopencv-features2d2.4:armhf libopencv-flann2.4:armhf libopencv-gpu2.4:armhf libopencvimgproc2.4:armhf libopencvml2.4:armhf libopencv-photo2.4:armhf libopencv-stitching2.4:armhf libopencv-video2.4:armhf libopenjpeg2:armhf libopenobex1 libopenvg1-mesa:armhf liborbit-2-0:armhf liborc-0.4-0:armhf libots0 libp11-kit-gnome-keyring:armhf libpam-gnome-keyring:armhf libpaper-utils libpaper1:armhf libpathplan4 libpcsclite1:armhf consolekit libpixman-1-0:armhf libpixman-1-0-dbg:armhf libplist1:armhf libpolkit-agent-1-0:armhf libpolkit-backend-1-0:armhf libpolkit-gobject-1-0:armhf libpostproc52 libprotobuf8:armhf libprotobuf-

lite8:armhf libproxy1:armhf libqmi-glib0:armhf libqpdf13:armhf libquvi-scripts libraptor2-0:armhf librarian0 librasqal3:armhf libraw1394-11:armhf librxtx-java libsamplerate0:armhf libsane-common libsbc1:armhf libschroedinger-1.0-0:armhf libsecret-1-0:armhf libsecret-common libshairplay libsidplayfp:armhf libsoundtouch0:armhf libsp1c2 libspeex1:armhf libspeexdsp1:armhf libsrtp0 libt1-5 libtag1-vanilla:armhf libtag1c2a:armhf libtagc0:armhf libtcl8.6:armhf libtelepathy-glib0:armhf libthai-data libtheora0:armhf libudisks2-0:armhf libusbmuxd2 libv4l2rds0:armhf libva1:armhf libvdpau1:armhf libvisual-0.4-0:armhf libvo-aacenc0:armhf libvo-amrwbenc0:armhf libvpx1:armhf libvte-2.90-common libvte-common libwavpack1:armhf libwayland-client0:armhf libwayland-cursor0:armhf libwayland-server0:armhf libwbclient0:armhf libwebcam0 libwebkitgtk-3.0-common libwebp5:armhf libwebpdemux1:armhf libwebpmux1:armhf libwhoopsie0 libwildmidi-config libwildmidi1:armhf libwnck-3-common libwnck-common libwpd-0.9-9 libwpg-0.2-2 libwps-0.2-2 libwvstreams4.6-base libwvstreams4.6-extras libx11-6:armhf libx11-data libx11-xcb1:armhf libx264-142:armhf libxapian22 libxau6:armhf libxcb-dri2-0:armhf libxcb-dri3-0:armhf libxcb-glx0:armhf libxcb-icccm4:armhf libxcb-image0:armhf libxcb-keysyms1:armhf libxcb-present0:armhf libxcb-randr0:armhf libxcb-render0:armhf libxcb-shape0:armhf libxcb-shm0:armhf libxcb-sync1:armhf libxcb-util0:armhf libxcb-

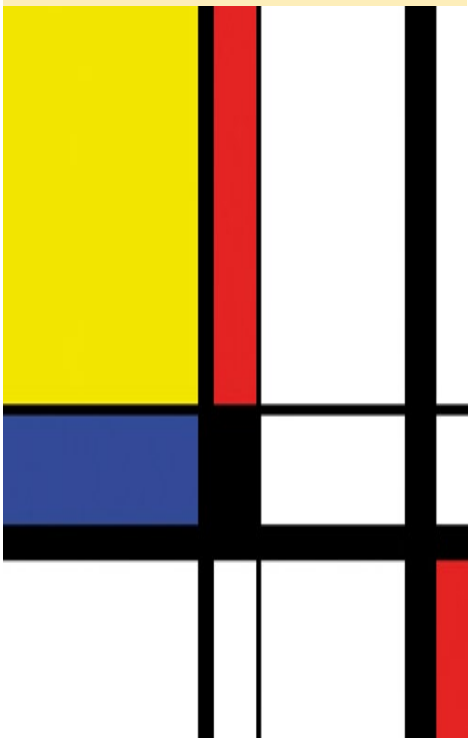
```
xf86dri0:armhf libxcb-
xfixes0:armhf libxcb-xv0:armhf
libxcb1:armhf
libxcomposite1:armhf
libxcursor1:armhf
libxdamage1:armhf
libxdmcp6:armhf libxdot4
libxext6:armhf libxfce4ui-common
libxfce4util-common libxfce4util6
libxfixes3:armhf libxi6:armhf
libxinerama1:armhf
libxkbfile1:armhf libxp6:armhf
```

Purge packages

```
# dpkg --list | grep ^rc | awk
-F" " ' { print $2 } ' | xargs
apt-get -y purge
```

After applying these changes, my disk usage dropped to just under 1GB. Keep in mind that following these instructions completely removes the GUI so that the HDMI no longer works, which leaves only the serial port and SSH available for managing the ODROID-C1. Please leave any comments, suggestions and feedback on the original post at http://bit.ly/1CDvNIO.

A minimal installation on an ODROID-C1 is a work of fine art, like a Mondrian painting



ODROID MAGAZINE ON GOOGLE+

FOLLOW US FOR THE LATEST UPDATES

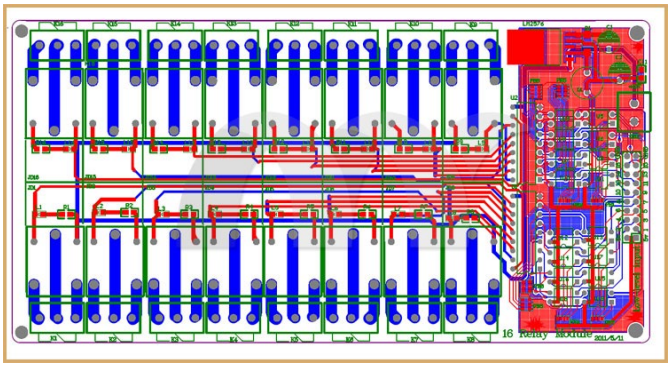
by Rob Roy

Would you like to know when the newest edition of your favorite online magazine has been released? Add ODROID Magazine to your Google+ circle to be notified immediately as soon as the next issue has been posted. Find out more about the ODROID Magazine community page at http://bit.ly/14rsCIr.

HARDWARE TINKERING

INTERFACING THE ODROID-C1 WITH A 16-CHANNEL RELAY

by @vzool



After I got my ODROID-C1, I read through its specifications and datasheet, and a big idea came to mind. My project was to create an easily accessible interface between the ODROID-C1 and SainSmart 16-Channel Relay Module (http://bit.ly/17ZbxrG). The relay module may be used to control appliances, lights, or any other gadget or device that requires 12V power.

By following the steps below, the relay may be programmed remotely via a web page, to be accessed by any type of device such as a PC or cell phone. The web page can then be published to the Internet, or locally via intranet, depending on your needs. For the software base, I used the the original Linux OS which came with my ODROID-C1 when I bought it.

Hardware requirements

1- 16x Transistors 2n2222 NPN: http://ebay.to/1CtxlVn

2- 16x Resistors 10 Ohm 1/4 Watt: http://ebay.to/1AD3ojt

3- 16x Resistors 2K2 Ohm 1/4 Watt: http://bit.ly/1xpRjx6

4- 1x Solderless Plug-in BreadBoard: http://amzn.to/14Z0Rar

5- 1x 16-Channel Relay Module: http://bit.ly/1yAXMLG

6- 40x female-male breadboard connections: http://amzn.to/1DVpRP4

Software Configuration

Install and configure the wiringPi application by typing the following commands into a Terminal window:

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

I used PHP for server side scripting, but you can implement it using any language that you want. First, install the necessary packages:

```
$ sudo apt-get install apache2
php5 libapache2-mod-php5 nano
```

Next, open the Apache configuration file:

```
$ nano /etc/apache2/sites-
available/000-default.conf
```

Change the following line:

```
DocumentRoot /var/www/html
```

to

```
DocumentRoot /var/www
```

Then, open the php5 configuration file:

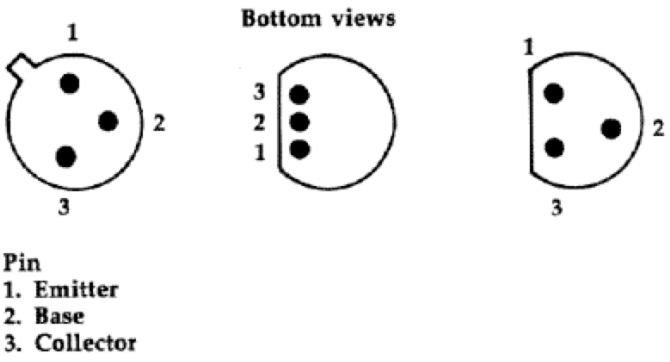
```
$ nano /etc/php5/apache2/php.ini
```

Change the option short_open_tag to On, which is Off by default:

```
short_open_tag = On
```

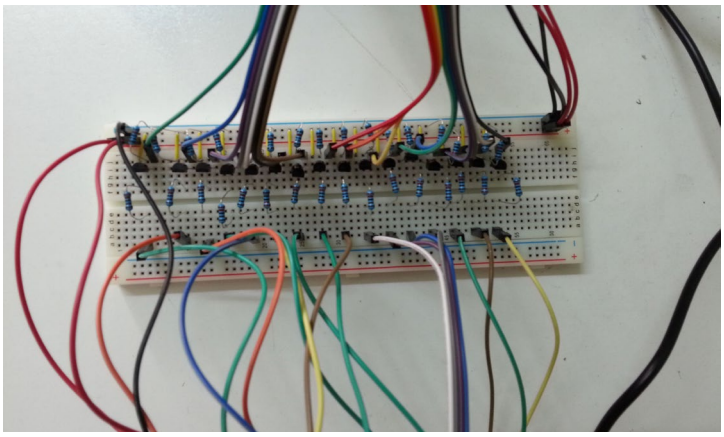
Create a new PHP file, and paste the following code:

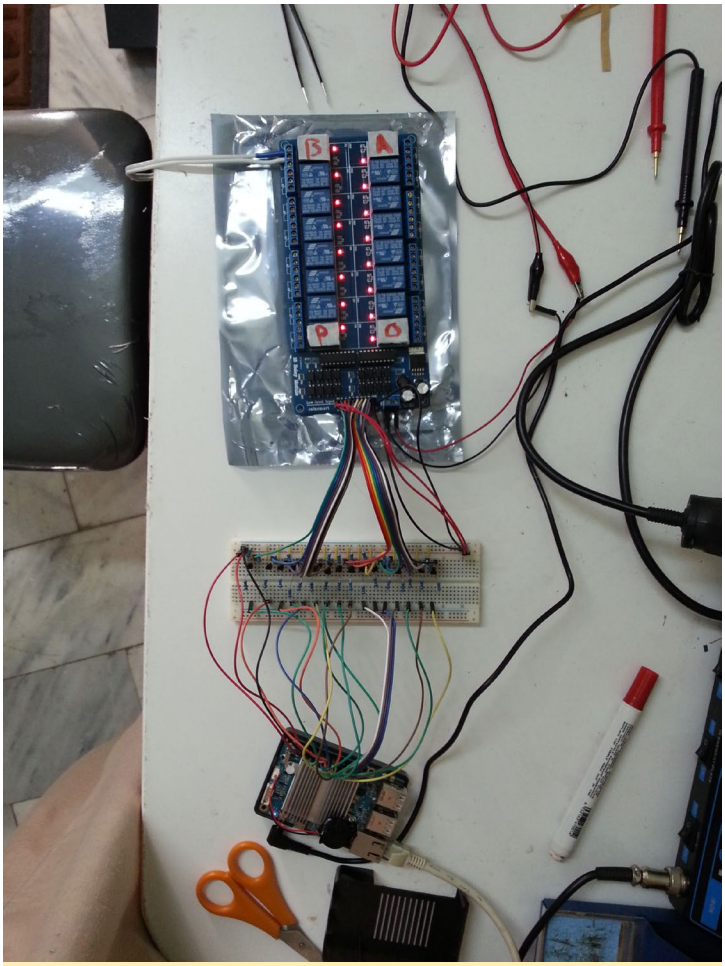
```
$ nano /var/www/odroid-c1.php

<?
// GPIO Configuration
$GPIO = array(
    array('Char' => 'A', 'Wiring-
Pi' => '14', 'Header' => '23'),
    array('Char' => 'B', 'Wiring-
Pi' => '7', 'Header' => '7'),
    array('Char' => 'C', 'Wiring-
Pi' => '22', 'Header' => '31'),
    array('Char' => 'D', 'Wiring-
Pi' => '1', 'Header' => '12'),
    array('Char' => 'E', 'Wiring-
Pi' => '12', 'Header' => '19'),
    array('Char' => 'F', 'Wiring-
Pi' => '5', 'Header' => '18'),
    array('Char' => 'G', 'Wiring-
Pi' => '13', 'Header' => '21'),
    array('Char' => 'H', 'Wiring-
Pi' => '10', 'Header' => '24'),
    array('Char' => 'I', 'Wiring-
Pi' => '21', 'Header' => '29'),
    array('Char' => 'J', 'Wiring-
Pi' => '3', 'Header' => '15'),
    array('Char' => 'K', 'Wiring-
Pi' => '24', 'Header' => '35'),
    array('Char' => 'L', 'Wiring-
Pi' => '0', 'Header' => '11'),
    array('Char' => 'M', 'Wiring-
Pi' => '4', 'Header' => '16'),
    array('Char' => 'N', 'Wiring-
Pi' => '2', 'Header' => '13'),
```

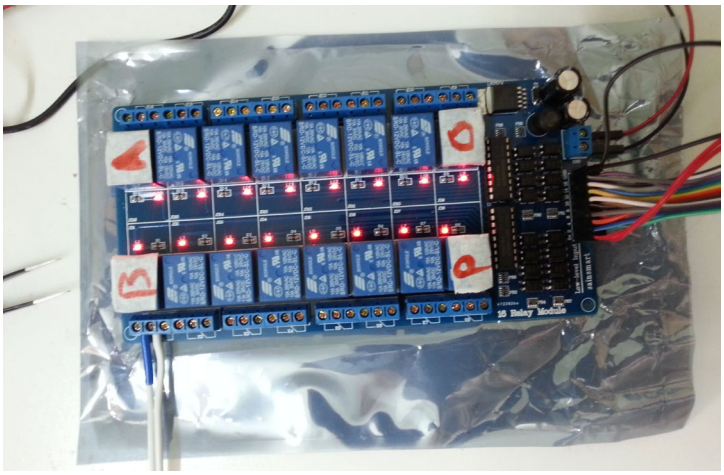


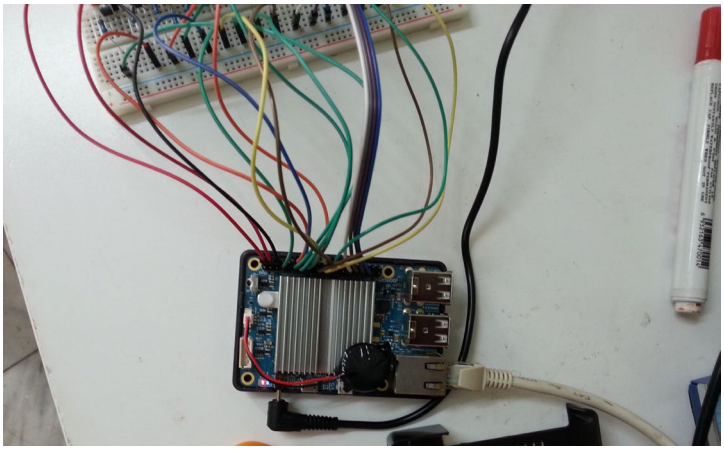
Pin
1. Emitter
2. Base
3. Collector

Typical pin diagrams for 2n2222 npn transistors

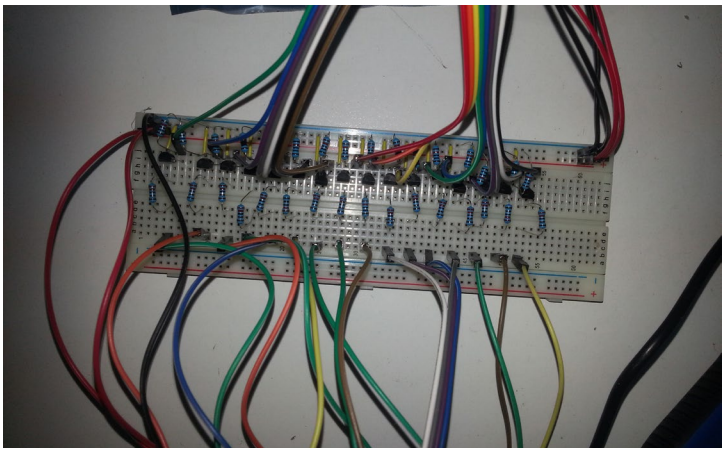








Photos of the wiring for the relay



```
    array('Char' => 'O', 'Wiring-
Pi' => '23', 'Header' => '33'),
    array('Char' => 'P', 'Wiring-
Pi' => '6', 'Header' => '22'),

    array('Char' => null, 'Wir-
ingPi' => '11', 'Header' =>
'26'),
    array('Char' => null, 'Wir-
ingPi' => '26', 'Header' =>
'32'),
    array('Char' => null, 'Wir-
ingPi' => '27', 'Header' =>
'36'),
);

//user => password
$users = array("admin" =>
"pass");

/*==============================
==============================
==========================*/
/*##############################
Authenticated Access Security ###
##########################*/
/*==============================
==============================
==========================*/
$realm = 'Restricted Area!';
$wrong_credential_message =
"<h1>401 Restricted Area: Failed
to Authenticate!</h1>";


if (empty($_SERVER['PHP_AUTH_DI-
GEST'])) {
    header('HTTP/1.1 401 Unau-
thorized');
    header('WWW-Authenticate: Di-
gest realm="'.$realm.

'",qop="auth",nonce="'.
uniqid().'",opaque="'.
md5($realm).'"');

    die('Text to send if user
hits Cancel button');
}

// analyze the PHP_AUTH_DIGEST
variable
if (!($data = http_digest_
parse($_SERVER['PHP_AUTH_DI-
GEST'])) ||
    !isset($users[$data['userna
me']]))
    die($wrong_credential_mes-
sage);

// generate the valid response
$A1 = md5($data['username']
. ':' . $realm . ':' .
$users[$data['username']]);
$A2 = md5($_SERVER['REQUEST_
METHOD'].':'.$data['uri']);
$valid_response = md5($A1.':'.$da
ta['nonce'].':'.$data['nc'].':'.
$data['cnonce'].':'.$data['qop']
.':'.$A2);

if ($data['response'] != $valid_
response)
    die($wrong_credential_mes-
sage);

// function to parse the http
auth header
function http_digest_parse($txt){
    // protect against missing
data
    $needed_parts =
array('nonce'=>1, 'nc'=>1,
'cnonce'=>1, 'qop'=>1, 'user-
name'=>1, 'uri'=>1, 're-
sponse'=>1);
    $data = array();
    $keys = implode('|', array_
keys($needed_parts));

    preg_match_all('@
(' . $keys . ')=(?:([\'"])
([^\2]+?)\2|([^\s,]+))@', $txt,
$matches, PREG_SET_ORDER);

    foreach ($matches as $m) {
        $data[$m[1]] = $m[3] ?
$m[3] : $m[4];
        unset($needed_
parts[$m[1]]);
    }

    return $needed_parts ? false
: $data;
}
/*==============================
==============================
==========================*/
/*##############################
Authenticated Access Security ###
##########################*/
/*==============================
==============================
==========================*/

if(!function_exists("php_cli")){
    function php_cli($cmd, $auto_
reload = true){
        $result = trim(shell_
exec($cmd));
        if($auto_reload)
header('Location: '.$_
SERVER['REQUEST_URI']);
        return $result;
    }
}

if(!function_exists("gpio_ref")){
    function gpio_ref($char,
$GPIO){
        foreach($GPIO as $g){
            if($g['Char'] ===
$char)
                return $g;
        }
        return null;
    }
}

$gpio = -1;
$mode = -1;
$status = -1;

if($_POST){

    try{
        if(isset($_POST['gpio']))
{
            $gpio = $_
POST['gpio'];
            $mode = $_
POST['mode'];
            $status = $_
```

```
POST['status'];

            php_cli("gpio mode
$gpio $mode && gpio write $gpio
$status");
        }

        if(isset($_POST['on_
all'])){
            foreach($GPIO as $g){
                $auto_reload =
end($GPIO) === $g;
                php_cli("gpio
mode {$g['WiringPi']} out && gpio
write {$g['WiringPi']} 1", $auto_
reload);
            }
        }

        if(isset($_POST['off_
all'])){
            foreach($GPIO as $g){
                $auto_reload =
end($GPIO) === $g;
                php_cli("gpio
mode {$g['WiringPi']} out && gpio
write {$g['WiringPi']} 0", $auto_
reload);
            }
        }

        if(isset($_POST['cmd'])){
            $header = gpio_
ref(strtoupper($_POST['cmd']),
$GPIO);
            $header =
$header['WiringPi'];
            $result = php_
cli("gpio read $header", false);
            if($result === '1'){
                php_cli("gpio
write $header 0 && gpio mode
$header in");
            }else{
                php_cli("gpio
mode $header out && gpio write
$header 1");
            }
        }

    }catch(Exception $ex){
        echo "<h1>$ex</h1>";
    }
}

$GPIO_STATUS = array();
foreach($GPIO as $g){
    if(!$g['Char'])continue;
    $GPIO_STATUS[$g['Char']]
= php_cli("gpio read
{$g['WiringPi']}", false);
}
?>

<center>
    <h1>^_^ Welcome to My Home
Infrastructure Panel(HIP) ^_^</
h1>
</center>
<hr/>
<div id='cmd_button'>
<?foreach($GPIO as $g):?>
    <?if(!$g['Char'])continue;?>
    <form method="post">
        <button class='<?=$GPIO_
STATUS[$g['Char']] == "1" ?
"on" : "off"?>' name='cmd'
type='submit' value='<?=$g['Ch
ar']?>'><?=$g['Char']?> -
 <?=$GPIO_STATUS[$g['Char']]
== "1" ? "ON" : "OFF"?></button>
    </form>
<?endforeach?>
</div>

<table border='1' width='100%'>
    <tr align='center'>
        <td>
            <form method="post">
                    <select
name='gpio'>

<?foreach($GPIO as $g):?>

<? $selected = $gpio ==
$g['WiringPi'] ? "SELECTED" :
""?>

<option <?=$selected?> value='<
?=$g['WiringPi']?>'>Header PIN
<?=$g['Header']?> ### WiringPi
<?=$g['WiringPi']?></option>
<?endforeach?>
                    </select>

                    <select
name='mode'>
                        <option
<?= $mode == "out" ? "SELECTED" :
""?> value='out'>OUT</option>
                        <option
<?= $mode == "in" ? "SELECTED" :
""?> value='in'>IN</option>
                    </select>

                    <select
name='status'>
                        <option
<?= $status == 1 ? "SELECTED" :
""?> value='1'>ON</option>
                        <option
<?= $status == 0 ? "SELECTED" :
""?> value='0'>OFF</option>
                    </select>

                    <input
type='submit' value='Execute'/>
            </form>
        </td>
        <td>
            <form method="post">
                    <input
name='on_all' type='submit'
value='ON ALL'/>
                    <input
name='off_all' type='submit'
value='OFF ALL'/>
            </form>
        </td>
    </tr>
</table>
<center>
    <h6>powered by <a
href='http://www.hardkernel.
com/main/products/prdt_info.
php?g_code=G141578608433' tar-
get='_blank'>ODROID-C1</a> Coded
by <a href='https://plus.google.
com/u/0/109727413094063366437'
target='_blank'>vZool</a></h6>
</center>
<style type="text/css">
```

```
form button{
   width: 50%;
   height: 10%;
    float: right;
}
.on, .off{
   font-weight: bold;
   font-size: 2em;
}
.on{
   color: green;
}
.off{
   color: red;
}
</style>
```

Save the file, then open a web browser and point it to http://<your-device-ip>/odroid-c1.php in order to access the Home Infrastructure Panel (HIP). The default user is admin and password is pass, which may be changed on line 27 of the PHP code.

Power

The 16-Channel Relay Module needs more power than the ODROID-C1 can provide via the GPIO pins, so I used my laboratory power supply unit (PSU) for the project. For your own hardware, you will need an adapter or similar source for 12V 0.5A power, which should be connected to the external power socket which is located to the right of the low level input pins.

More information

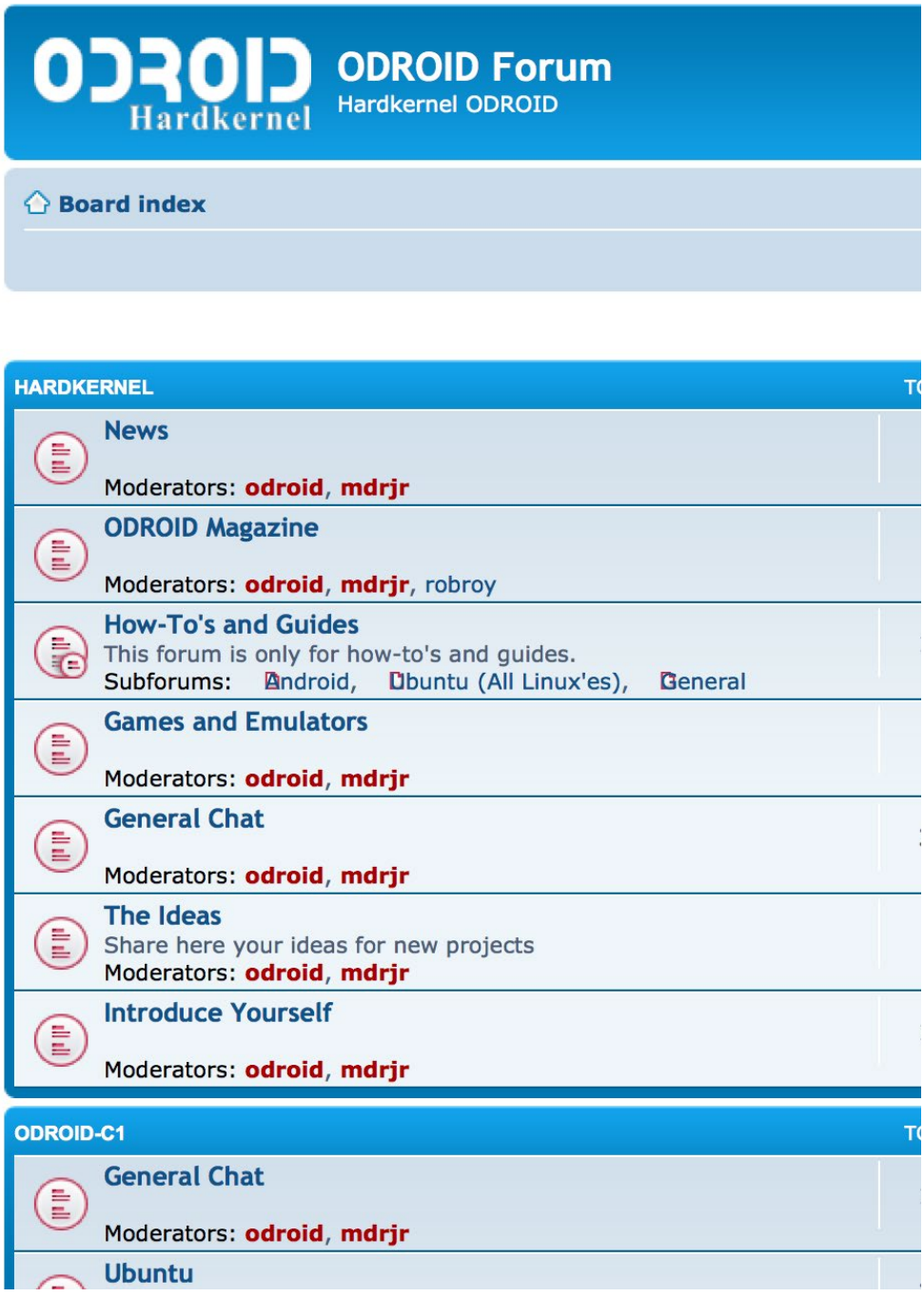
For more information about Raspberry Pi-stye PC and Relay interface, check out the collection of home automation videos at http://bit.ly/1BX5Wxj. Specific details about wiring the relay may be found at http://bit.ly/15r7Byv. For reference, the ODROID-C1 datasheet is available at http://bit.ly/1KWdJiM. If you would like to ask questions or leave feedback regarding this project, please visit the original post at http://bit.ly/15n93SQ.

ODROID FORUMS

THE PERFECT PLACE TO COMMUNICATE WITH HARDKERNEL DEVELOPERS

by Rob Roy

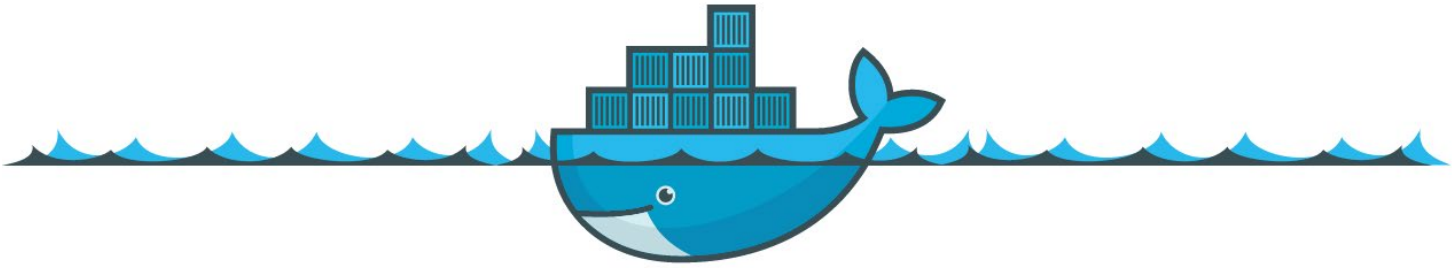
The ODROID forums have been the central meeting place for the growing Hardkernel community for several years, with over 8500 members as of February 2015. You can discuss ODROIDs with Mauro, the lead Linux kernel developer, and Justin, the CEO of Hardkernel, along with a growing team of developers who donate their time to helping you get the most out of your ODROID. Check it out at http://forum.odroid.com!



DOCKER: DEVELOP, SHIP AND RUN ANY APPLICATION, ANYWHERE

PART 2 - PRE-BUILT IMAGES

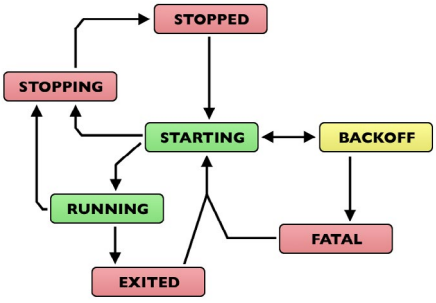
by Fred Meyer



The previous article in my series on Docker detailed how to set up Docker on the ODROID platform, along with an overview on how containers work. This article presents several pre-built images, available for free download, that are designed to help you get Docker up and running quickly with minimal setup.

Since initd/systemd are not available inside the container, a Docker container is designed to run only one foreground process. However, sometimes there is the need to run two processes concurrently, such as a web server, in order to provide a GUI for configuring or controlling a running backend process. When the container starts or stops, the processes need to mimic that command. There are two preferred options available. One is a very small/lean initd/systystemd replacement, like runit (http://bit.ly/1zp2o7s), and the other is to use supervisor-daemon (http://bit.ly/1yb6w95).

Subprocess transitions



Ubuntu and Debian

I prepared a pre-built Docker image on top of the official Hardkernel Ubuntu 14.04 image that includes supervisor-daemon as a foreground process. Inside supervisord, the SSH-daemon is already enabled and running on port 22. Fetch and run the Ubuntu image with the following commands:

```
$ sudo docker pull hominidae/
armhf-supervisord
$ sudo docker run -d -p 8022:22
hominidae/armhf-supervisord
```

Next, SSH into that mapped port 8022 on your ODROID host, using the default configured username and password of ubuntu/ubuntu. Please refer to http://bit.ly/1CBw8f4 and http://bit.ly/1xi309d for more information.

As another available pre-built option, there is a Debian Wheezy image with runit enabled, and the SSH-daemon running on port 22. Fetch and run the Debian image using the following commands:

Debian Wheezy runs Docker very well


```
$ sudo docker pull hominidae/
armhf-wheezy
$ sudo docker run -d -p 9022:22
hominidae/armhf-wheezy \ /usr/
sbin/runsvdir-start
```

Connect using SSH via port 9022 on your ODROID host, using the default configured username and password of wheezy/wheezy.

Arch Linux base image

If you prefer Arch Linux, download my Arch Linux base image by typing the following:

```
$ docker pull hominidae/armhf-
archLinux
```

Should you prefer to create the image

Arch Linux runs Docker too

yourself, you can do so using the following script:

```
#!/usr/bin/env bash
# Generate a minimal filesystem
for Arch Linux
# and load it into the local
docker as "armhf-archLinux"
# requires root
# based on https://github.com/
docker/docker/blob/master/con-
trib/mkimage-arch.sh
set -e

hash pacstrap &>/dev/null || {
  echo "Could not find pacstrap.
Run pacman -S arch-install-
scripts"
  exit 1
}

hash expect &>/dev/null || {
  echo "Could not find expect.
Run pacman -S expect"
  exit 1
}

ROOTFS=$(mktemp -d ${TMPDIR:-/
var/tmp}/rootfs-archLinux-
XXXXXXXXXX)
chmod 755 $ROOTFS

# packages to ignore for space
savings
PKGIGNORE=Linux,jfsutils,lvm2,cry
ptsetup,groff,man-db,man-pages,md
adm,pciutils,pcmciautils,reiserfs
progs,s-nail,xfsprogs

expect <<EOF
  set send_slow {1 .1}
  proc send {ignore arg} {
     sleep .1
     exp_send -s -- \$arg
  }
  set timeout 60

  spawn pacstrap -C ./mkimage-
arch-pacman.conf -c -d -G -i
$ROOTFS base haveged --ignore
$PKGIGNORE
  expect {
     -exact "anyway? \[Y/n\] " {
send -- "n\r"; exp_continue }
     -exact "(default=all): " {
send -- "\r"; exp_continue }
     -exact "installation? \
[Y/n\]" { send -- "y\r"; exp_con-
tinue }
  }
EOF

arch-chroot $ROOTFS /bin/sh -c
"haveged -w 1024; pacman-key
--init; pkill haveged; pacman -Rs
--noconfirm haveged"
arch-chroot $ROOTFS /bin/sh -c
"ln -s /usr/share/zoneinfo/UTC /
etc/localtime"
echo 'en_US.UTF-8 UTF-8' > $ROOT-
FS/etc/locale.gen
arch-chroot $ROOTFS locale-gen
arch-chroot $ROOTFS /bin/sh -c
'echo "Server = http://mirror.
archLinuxarm.org/\$arch/\$repo" >
/etc/pacman.d/mirrorlist; pacman
--noconfirm -Sy; pacman --noconfirm
-S archLinuxarm-keyring'

# udev doesn't work in contain-
ers, rebuild /dev
DEV=$ROOTFS/dev
rm -rf $DEV
mkdir -p $DEV
mknod -m 666 $DEV/null c 1 3
mknod -m 666 $DEV/zero c 1 5
mknod -m 666 $DEV/random c 1 8
mknod -m 666 $DEV/urandom c 1 9
mkdir -m 755 $DEV/pts
mkdir -m 1777 $DEV/shm
mknod -m 666 $DEV/tty c 5 0
mknod -m 600 $DEV/console c 5 1
mknod -m 666 $DEV/tty0 c 4 0
mknod -m 666 $DEV/full c 1 7
mknod -m 600 $DEV/initctl p
mknod -m 666 $DEV/ptmx c 5 2
ln -sf /proc/self/fd $DEV/fd

tar --numeric-owner -C $ROOTFS
-c .  | docker import - armhf-
archLinux
docker run -i -t armhf-archLinux
echo Success.
sleep 2
rm -rf $ROOTFS
```

You also need to download the mkimage-arch-pacman.conf file from the same repository at http://bit.ly/1IQe2K0, making sure to set the architecture to "armv7hf".

Needful Things

The space that a Docker image consumes can get very large as you iterate over it by stopping, starting and modifying it. There are some tweaks that can be done in order to strip the bloat and save space on your eMMC or SD, as demonstrated at http://bit.ly/1wjki47.

Docker needs lots of disk space

Here is some useful information to be found about using "save" or "export" commands with Docker: http://bit.ly/1GdEu2t. For details on using networking and mapping ports with a Docker container, visit http://bit.ly/1EclIDL.

Running a container in daemon mode will start its foreground process. You can also enter into a shell in the running, daemonized container using the exec command:

```
$ sudo docker exec -i -t <id> /
bin/bash
```

Ubuntu 14.04 with ReadyMedia

A container running minidlna and ReadyMedia in order to stream music to my various audio devices was one

Digital Living Network Alliance (DLNA)

of the first images that I built, which is still running strong in my home. The container is based on Ubuntu 14 and includes build tools, libs for codecs and minidlna dependencies, as well as the minidlna sources.

Here's the Dockerfile:

```
#
# MiniDLNA Dockerfile
#

# Pull base image.
# FROM hominidae/armhf-supervi-
sord
FROM hominidae/armhf-ubuntu:14.04

MAINTAINER hominidae

# Install MiniDLNA(ReadyMedia).
RUN apt-get update && apt-get
upgrade -y

# build tools and codecs
RUN apt-get install -y wget
build-essential libavutil-dev \
   libavcodec-dev libavformat-
dev libjpeg-dev libsqlite3-dev
libid3tag0-dev \
   libogg-dev libvorbis-dev lib-
flac-dev libexif-dev gettext

# download minidlna source code,
build and install
RUN \
 cd /tmp && \
 wget http://downloads.source-
forge.net/project/minidlna/
minidlna/1.1.4/minidlna-
1.1.4.tar.gz && \
 tar xvzf minidlna-1.1.4.tar.gz
&& \
 cd minidlna-1.1.4 && \
 ./configure && \
 make && make install

# add config file.
ADD minidlna.conf /etc/minidlna.
conf

# Define mountable directories.
VOLUME ["/data"]

# Define working directory.
WORKDIR /data

# Define default command.
CMD ["/usr/local/sbin/
minidlnad","-d"]

# Expose ports.
#   - 1900: UPnP
#   - 8200: HTTP
EXPOSE 1900/udp
EXPOSE 8200

# supervisor configuration für
minidlna
#ADD minidlna_d.conf /etc/super-
visor/conf.d/minidlna_d.conf
#
#CMD ["supervisord", "-c", "/etc/
supervisor/supervisord.conf"]
```

This Dockerfile will inject your minidlna.conf configuration file and provision a volume called "/data" inside the container. You'll have to point your minidlna to that directory in order to store and find your media, as well as map the host directory/filesystem into that container. Should you opt to run minidlna together with an SSH daemon, just use the supervisord-image as the base image with the FROM directive, and add the appropriate supervisord.conf into the image.

The image exposes the upnp/dlna udp port (1900) and http UI port (8200) for minidlna. However, for a renderer to find and receive the broadcasted advertisements of minidlna, you will need to start/run the image with the "--net=host" command option.

This is how I start the image locally, assuming your media is attached to the ODROID and is available at /media/mediadata/my-music on the host:

```
$ sudo docker run
--name=minidlna_d --rm=true
--net=host -p 1900:1900/udp -p
8200:8200 -v /media/mediadata/my-
music:/data:ro hominidae/armhf-
minidlna
```

Cups and Cloud Print

You can also add a Google Cloud Print server to your home and enable your local printer(s) for use with Google Print. This is another one of my favourite containers, as I already own some networked printers, but all of them lack Google Cloud Print capabilities. With this container running, I can use the printers from my Android phone, tablet and ChromeBrowser/OS, which adds convenience for my family, as well as upgrading my printer's capabilities.

Common Unix Printing System (CUPS)

Cloud Print lets you print to any printer

This particular image is built on Debian Wheezy, and adds two major components: cupsd, the UNIX printing daemon, and a Python script (http://bit.ly/1IVP766) that is able to connect to the Google printing API. This time, we are not using a Dockerfile, but will build

the image interactively. To begin, start the Debian Wheezy base image:

```
$ sudo docker run -t -i
--net=host hominidae/armhf-wheezy
/bin/bash
```

Note that we're using the --net=host option in order to enable full IP access for the container. If you have multiple containers running with sshd inside, you should reconfigure the ports or disable all ssh-daemons except one, including the one on your host. You can also enter into a shell in the running, daemonized container using the exec command:

```
$ sudo docker exec -i -t <id> /
bin/bash
```

Next, enter the container and install cups:

```
$ apt-get update && apt-get up-
grade -y && apt-get install -y
cups
```

Then, check and configure the cupsd configuration file at /etc/cups/cupsd.conf, enable the Web-UI of cupsd using the default port 631, and allow access for admin:

```
[...]
# Only listen for connections
from the local machine.
Listen 0.0.0.0:631
[...]
# Restrict access to the serv-
er...
<Location />
 Order allow,deny
 Allow all
</Location>

# Restrict access to the admin
pages...
<Location /admin>
 Order allow,deny
 Allow all
</Location>

# Restrict access to configuration
files...
<Location /admin/conf>
 AuthType Default
 Require user @SYSTEM
 Order allow,deny
 Allow all
</Location>
[...]
```

Add a user as Admin for the printing system to the group lpadmin and create a password:

```
$ sudo adduser cupsadmin
[...]
$ sudo usermod -a -G lpadmin cup-
sadmin
```

Enable cupsd startup with runit:

```
$ sudo mkdir /etc/service/cupsd
$ sudo cp /etc/service/sshd/run /
etc/service/cupsd/run
```

Edit the run-file /etc/service/cupsd/run to look like this:

```
#!/bin/sh
#
# start cupsd
exec /etc/init.d/cups start
# end
```

Logout from your container and fetch the container ID, then stop the Wheezy image:

```
$ sudo docker ps -a
$ sudo docker stop <id>
```

Create a new image from the container, so we have a state to return to later, just in case, then start the new cupsd container:

```
$ sudo docker export <id> | sudo
docker import - armhf-cupsd
$ sudo docker run -d --net=host
armhf-cupsd \ /usr/sbin/runsvdir-
start
```

After entering the container via SSH to check if cupsd is running, point your browser to https://<your-odroid-ip>:631/admin and configure your printer. Use the cupsadmin/cupsadmin credentials to authenticate when asked.

I always have a ppd-file for my printers ready; upload it via the Web-Admin interface. Note that I did not test with USB-based printers, since my printers are already network-enabled. You could try and add the /dev filesystem as a volume, using the -v parameter, or the --device option to the run command when starting the container, but I am not sure about the security implications.

Finally, print a test page from the cups maintenance interface. If you need some more advice on cups, please consult the tutorial available at http://bit.ly/1KT6sjD.

Repeat the steps for all of your printers that you want to manage through this cupsd instance. Again, create a new image from the running container, this time including your configured printers:

```
$ sudo docker export <id> | sudo
docker import - armhf-cups_print-
ing
```

Then, add the Cloud Print python stuff, using the pre-built Debian packages from David Steele's PPA at http://bit.ly/17WkvpF. Once inside the container, run the commands to add the Cloud Print PPA. Add a reference to the Cloud Print PPA to your /etc/apt/sources.list file (as root):

```
$ deb http://davesteele.github.
io/Cloud Print-service/repo \
Cloud Printppa main
```

Add the repository key to your apt key ring and install the packages:

```
$ wget http://davesteele.github.
io/key-366150CE.pub.txt
```

```
$ sudo apt-key add key-366150CE.pub.txt
$ sudo apt-get update
$ sudo apt-get install Cloud Print Cloud Print-ser-
vice
```

After completing these steps, the packages will be automatically updated whenever "apt-get upgrade" is run.

Register the printer

Google accounts that have 2-step verification enabled need to use an application-specific password, as described at http://bit.ly/1CBGfAy. Configure these account credentials before you try to connect to the printing service.

Next, run the Cloud Print python app, using your own credentials, which will register any printers that are configured in cupsd with Google Cloud Print:

```
$ /etc/init.d/Cloud Printd login
Accounts with 2 factor authentication require an
application-specific password
Google username: <your-id>@gmail.com
Password: <your_app_pwd>
Added Printer Brother_MFC-9120CN
```

Google Cloud Print Screenshot

Next, check the setup using Google Print (http://bit.ly/1yvWQou), which should show the printer as available, and do a test print. Then, enable the Cloud Print daemon with the runit services:

```
$ sudo mkdir /etc/service/Cloud Print
$ sudo cp /etc/service/sshd/run /etc/service/Cloud
Print/run
```

Edit the run-file /etc/service/Cloud Print/run to look like this:

```
#!/bin/sh
#
# start Cloud Print daemon
exec /etc/init.d/Cloud Printd start
# end
```

As the final step, exit from the container, stop it and create the final image in order to save your work:

```
$ sudo docker export <id> | sudo docker import -
armhf-Cloud Printd
```

You can now re-run your Cloud Print-server docker container:

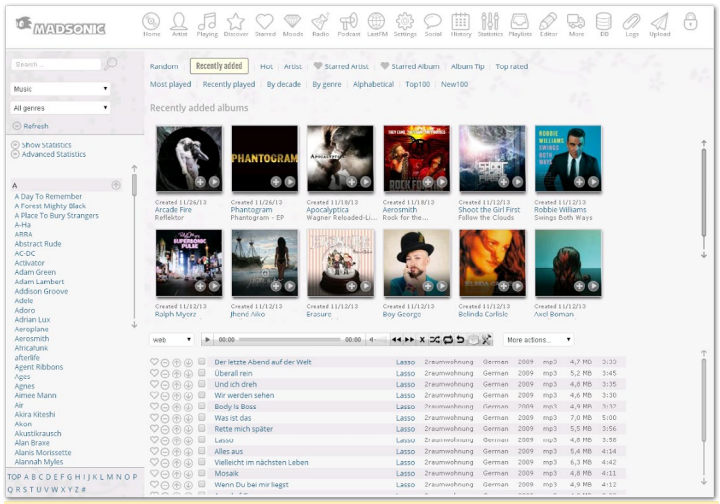
```
$ sudo docker run -d --net=host armhf-Cloud Printd \
/usr/sbin/runsvdir-start
```

Note that this container and image carries your Cloud Print credentials with Google. Although the credentials are stored in an encrypted format, I suggest not pushing this image to a public repository in order to avoid identity theft.

Arch Linux with Madsonic

Madsonic lets you stream your media over the web

Madsonic is a fork of the well known Subsonic application. Although it is capable of running a UPnP/DLNA-daemon like minidlna, its main purpose is to give access to your media and stream it over the web. There already is a Dockerfile available for x86 architecture at http://bit.ly/1EcvNRb. Since Madsonic is a Java app, this shouldn't impose a compatibility problem on ARM architecuter. However, Madsonic needs some transcoding plugins in order to be able to transcode the media for playback with different devices.



Madsonic Demo page

At the time of writing this article, I haven't been able to find ARM based transcoding libs or the source code for Madsonic. Nevertheless, the following is some brief information on how to build Madsonic on your ODROID, albeit without

transcoding capabilities. Start with the Dockerfile, which was adopted from the user binhex's examples:

```
FROM hominidae/armhf-base-arch-Linux
#MAINTAINER binhex
MAINTAINER hominidae

# install application
#####################

# update package databases from the server
RUN pacman -Sy --noconfirm

# install pre-req for application
RUN pacman -S libcups jre7-openjdk-headless fontconfig unzip --noconfirm

# make destination folders
RUN mkdir -p /var/madsonic/media
RUN mkdir -p /var/madsonic/transcode

# download madsonic
ADD http://madsonic.org/download/5.1/20140823_madsonic-5.1.5080-standalone.zip /var/madsonic/madsonic.zip

# unzip to folder
RUN unzip /var/madsonic/madsonic.zip -d /var/madsonic

# remove zip
RUN rm /var/madsonic/madsonic.zip

# force process to run as foreground task
RUN sed -i 's/-jar madsonic-booter.jar > \${LOG} 2>\&1 \&/-jar madsonic-booter.jar > \${LOG} 2>\&1/g' /var/madsonic/madsonic.sh

# install transcoders
#####################

# download madsonic transcoders
#ADD http://madsonic.org/download/transcode/20140819_madsonic-transcode_latest_x64.zip /var/madsonic/transcode/transcode.zip

RUN pacman -S ffmpeg lame flac --noconfirm

# unzip to folder
#RUN unzip /var/madsonic/transcode/transcode.zip -d /var/madsonic/transcode

# remove zip
#RUN rm /var/madsonic/transcode/transcode.zip

# copy transcode script to madsonic install dir (copies transcoders to madsonic install dir)
ADD transcode.sh /var/madsonic/transcode.sh

RUN cd /var/madsonic/transcode && ln -s "$(which ffmpeg)" && ln -s "$(which flac)" && ln -s "$(which lame)" && ls -la

# docker settings
#################

# set env variable for java
ENV JAVA_HOME /usr/lib/jvm/java-7-openjdk/jre

# map /config to host defined config path (used to store configuration from app)
VOLUME /config

# map /media to host defined media path (used to read/write to media library)
VOLUME /media

expose port for http
EXPOSE 4040

# expose port for https
EXPOSE 4050

# expose UPnP - DLNA ports
EXPOSE 1900/udp
EXPOSE 2869

# set permissions
#################

# change owner
RUN chown -R nobody:users /var/madsonic

# set permissions
RUN chmod -R 775 /var/madsonic

# add conf file
###############

ADD madsonic.conf /etc/supervisor/conf.d/madsonic.conf

# cleanup
#########

# completely empty pacman cache folder
RUN pacman -Scc --noconfirm

# run supervisor
################

# run supervisor
CMD ["supervisord", "-c", "/etc/supervisor/supervisor.conf", "-n"]
```

The base image with supervisord enabled, that is required for Madsonic, is also available from the git archive of user binhex at http://bit.ly/1KT6Z5k. However, you can opt to create the image from my armhf-archLinux image at http://bit.ly/1Ecx76C by adapting the Dockerfile. Don't forget to fetch the configuration files for Madsonic and supervisord from the arch-madsonic git-archive before you run the build on the armhf-madsonic image.

Debian Wheezy with FreeSwitch & FusionPBX

This container has the capability of hosting a fully functional IP Private Branch Exchange (IP-PBX) with a convenient web-based user interface. Incidentally, Hardkernel's business partner Sipbox (sipbox.co.uk) sells a complete IP-PBX system, based on an ODROID-U3, using the same software stack, which runs natively without Docker.

Fusion PBX offers a custom IP-PBX

Free Switch is a telephony platform

To begin, refer to http://bit.ly/1AuGkDy for details on the base setup. The included script will pull the complete source code and build a sipbox-like system on your ODROID. However, unlike the original version, we will use a Docker container to host the files. Make sure to have approximately 1.5GB of storage available before setting up this system.

To begin, use the Debian Wheezy image as a base. Enter the container, become root and fetch the install-script:

```
# apt-get install wget -y
# cd /usr/src
# wget http://bit.ly/Rfvxy5
# chmod 755 install_fusionpbx.sh
```

Configure some variables that control the build by editing the install-script. First, enable the nginx and sqllite3 components, since these are the most resource-friendly, then run the script:

```
[...]
#---------
#VARIABLES
#---------
#Variables are for the auto in-
stallation option.

#for apache set to a, for nginx/
php-fpm set to n -> for an auto
install, user mode will prompt
APACHENGINX=n

# for mysql set m.  for sqlite
set s.  for postgresql set p
SQLITEMYSQL=s
[...]
```

Opt to install both components, which will take quite a while. Some of the options are interactive, so stand by and monitor its progress.

```
# ./install_fusionpbx.sh install-
both auto
```

While building, the script will detect that the local version of itself has changed compared to the one in the repository, because we've edited it. Just select *not* to use the "newer" version from the repository by pressing "y" when the following message is displayed:

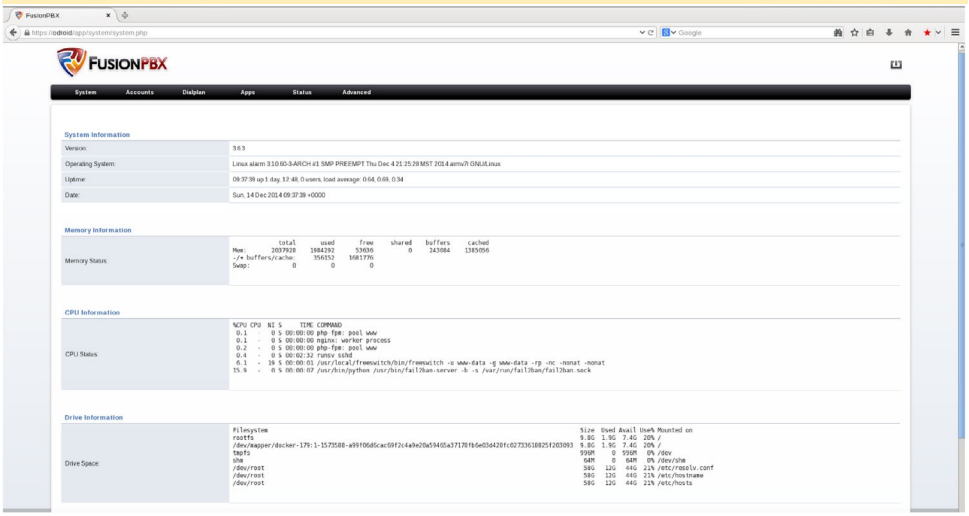
```
there is a new version of this
script.
 It is PROBABLY a good idea use
the new version
 the new file is saved in /tmp/in-
stall_fusionpbx.latest
 to see the difference, run:
 diff -y /tmp/install_fusionpbx.
latest /usr/src/install_fusion-
pbx.sh
Continue [y/N]?
```

Once everything has been built, both components will be installed and activated. The script will prompt you to connect your browser towards your fusionpbx instance in order to set up accounts and save a basic configuration. Once you finish saving, the script will finalize its build.

In order to enable the services in Docker, you will need to add them to the runit service. When the install script finishes, you can examine which services have alrady been started. All start scripts have been installed into /etc/init.d/.

Create a directory for each service un-

Fusion PBX Screenshot



der /etc/service/ and create a run-script, which invokes the script in /etc/init.d/. You will want to create a script for each of the following services, resulting in this runit dir-tree:

```
/etc/service
|-- cron
|   |-- run
|-- dbus
|   |-- run
|-- fail2ban
|   |-- run
|-- freeswitch
|   |-- run
|-- nginx
|   |-- run
|-- ntp
|   |-- run
|-- php5-fpm
|   |-- run
|-- sshd
    |-- run
```

For the final step, exit from the container, stop it, and create the final image to save your work:

```
$ sudo docker export <id> | sudo
docker import - armhf-mysipbox
```

You can now re-run your sipbox Docker container:

```
$ sudo docker run -d --net=host
armhf-mysipbox \ /usr/sbin/runsv-
dir-start
```

Consult the excellent fusionpbx wiki on how to configure your IP-PBX with your SIP-Provider and SIP-Devices/-Extensions at http://bit.ly/1sOugPY.

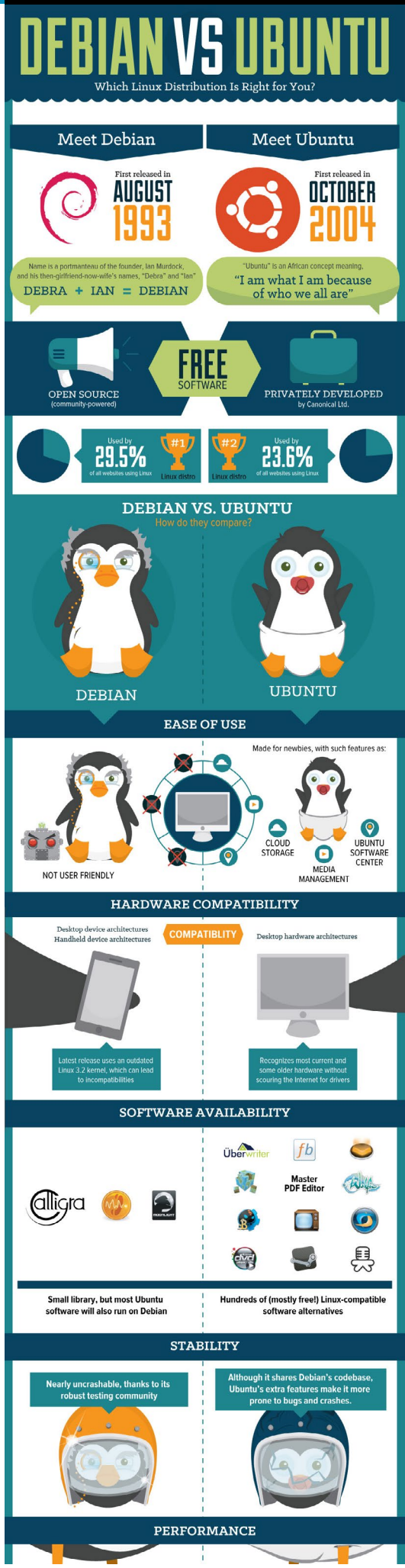
Other Docker ideas

Can you think of an application which you require on your ODROID that you would like to run inside a Docker container? There's a good chance that it has already been done, at least for the x86 architecture. Just browse the Docker Hub at https://hub.docker.com to see what is available. Many of the contributors offer their Dockerfile either directly or through a Github project. Remember that the Dockerfile is simply a type of start/build-script for creating or enhancing Docker images. As long as the Dockerfile is not importing any x86 or non-ARMHF specific binaries, they will work on the ODROID platform. The base armhf images from the first installment of this article should give you a good head start.

My next project will be about employing a smart home automation solution on my ODROID, along with the world of the Internet Of Things, using Docker containers, of course!

References

http://docs.docker.com
http://bit.ly/1CQs1hl
http://bit.ly/1Cop8Wj

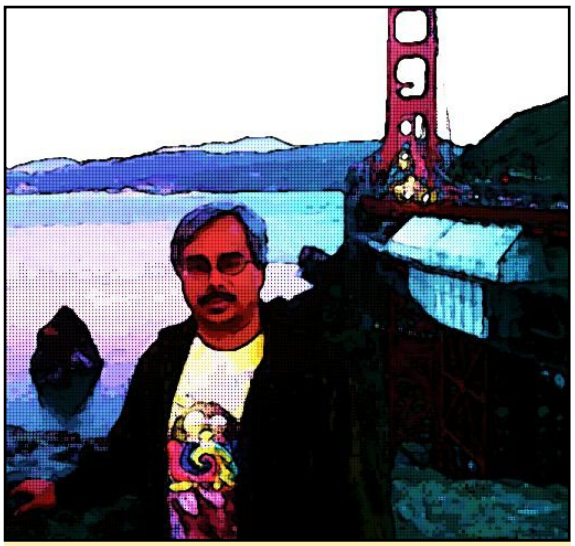


CLICK TO VIEW MORE

MEET AN ODROIDIAN

VENKAT BOMMAKANTI
JACK OF ALL TRADES

edited by Rob Roy



Venkat at the Golden Gate Bridge in San Francisco, California

Please tell us a little about yourself.

I am from South India, but have made the Bay Area in California my home. It's the one place on earth that permits a Chemical Engineer to dabble with robotics, banking, ip-telephony, biotechnology and networking.

How did you get started with computers?

In the early 1980s, I was in graduate school, and for the very first time had the chance to use an IBM 3081 using Hollerith punch cards and brand new VT220s. I had a donated Sinclair Z81 for my process control simulation thesis using UCSD Pascal. I have been hooked ever since, and bought my first PC at the young age of 30, which was an Intel-based 486 PC, at a price of $3000. Kids today are very lucky because, thanks to folks like Hardkernel, they can actually afford computers.

What drew you to the ODROID platform?

In my most recent job at the local networking giant, I had the once-in-a-lifetime opportunity to help migrate nOS to Linux, which started my journey into Linux-based embedded systems. I felt that x86-based Linux systems were too complicated, so I started looking at efficient Linux systems, and there it was: ARM to the rescue. I started with other platforms, but quickly settled on the ODROID devices, which offer the greatest bang for the buck, and are one of the most well-supported general purpose Linux systems available.

Which ODROID is your favorite?

That's a tough question to answer. From my Linux point of view, I think that the U2/U3 put Hardkernel on the map. They are both very special devices that embody all the right levels of important basics: form factor, interfaces, memory, power efficiency, computation power and reasonably quick native compiling. They make great development systems for testing out any full-featured Linux image. Conceptually, I think that they begat both the XU3-Lite and the C1, which basically address the different price-points driven by the balance between functionality and affordability. I love all of the ODROIDs, for different reasons.

Your technical articles are very detailed, how do you produce a typical feature for ODROID Magazine?

My first job was that of a teaching assistant in graduate school. It laid the foundation for being a meticulous learner. I quickly became aware that no two people learn the same way, so I gathered enough knowledge in order to address the lowest common denominator: the slowest learner (like myself). I don't like when material is presented without background information, which leads to frustration for some. Most often, outdated and hidden steps can't be replicated successfully without detailed instructions. So, I try to document every possible caveat in the process, to promote the longevity of the material.

Are you involved with any other computer projects unrelated to the ODROID?

Small form factor powerhouses like the ODROIDs, Beaglebone Black and, to a lesser extent, the Raspberry Pi and Arduino, have piqued my curiosity by interfacing the sensory world to the computational realm. I'm also looking into interfacing appropriate field-programmable gate array (FPGA) boards. Being a hands-on type of person, I'm anxious to get my home automation and car computer projects going.

What hobbies and interests do you have apart from computers?

Like everyone else, I try to appreciate life using all of my □a□□yatana, which is Sanskrit for the 6 basic senses. Through vision, I enjoy nature's beauty and try to capture some of it, if possible. I love to listen to woozy blues, bluegrass jams, Indian and Western classical instrumentals all day long. I wish there was an ODROID-based time machine that could take me to the 1960s music scene. I enjoy spicy cuisine from all over the world, satisfying my gustatory and olfactory curiosities. I experiment with cooking by mixing ingredients from various cuisines. I let my ODROID Magazine articles keep my mind busy and enjoy hiking, cycling and woodworking to clear my mind. I also volunteer at a couple of Bay Area nonprofits, to give back for so much that I have been blessed with.

What type of hardware innovations would you like to see for future Hardkernel boards?

There are five hardware improvements that I would like to see for ODROIDs:

• Simultaneously producing two types of boards - one maxed out with all the connectors and risers, and one with the bare minimum single level connectors and risers. For instance, a small footprint build would have only 2 USB ports side by side, with a boot media connector and power jack, which would be ideal for a light-weight, low-height minimalist embedded system.

• Moving all of the ports, media receptacles, and connectors to face the top of the device, which would allow changing cables, peripherals or boot media to be much easier, especially when the ODROID is encased.

• Relocating ports on the VU so that the connections are on the back instead of the side.

• Producing adapter boards and shields so that add-ons made for other popular boards (such as the Raspberry Pi or Beaglebone Black) can be reused with minimal effort on any ODROID. These adapter boards would take care of varying voltage levels, enabling circuit protections, and providing other compatibility features.

• Incorporating useful functionality from other best-of-breed single board computers, such as the equivalent of the Programmable Real-time Unit (PRU) that is available for the Beaglebone Black.

Although it is not a hardware innovation, I think that Hardkernel could further popularize ODROIDs by emulating the early success story of the Raspberry Pi. Sun started the Raspberry Pi legacy by giving away basic systems to Computer Science schools. As a result, every graduate then either wanted to work at Sun or use Sun systems at their jobs. Similarly, if possible, Hardkernel could allocate marketing funds towards donating boards to schools through competitions related to Science, Technology, Engineering and Mathemetics (STEM) that could be available for kids to enter. The beginning programmers could use an affordable ODROID-C1, enter the competition, and use their winnings to purchase higher-end Hardkernel devices. Familiarity, popularity and word-of-mouth will create a larger, vibrant young community of ODROID users.

What advice do you have for someone wanting to learn about programming?

An interesting job interview comes to mind, where the manager asked me, "What religion do you subscribe to?" I was taken aback, since I was never asked such a question in a progressive place like California. He saw my perplexion and quickly corrected himself by saying, "I mean, which programming language do you prefer?" I said, "Well, in that case, I'm irreligious!"

Basically, I believe there is no single universally applicable programming panacea. Borrowing from my woodworking vernacular, I see all of these languages and utilities as a variety of tools in a programmer's toolbox. There is always an appropriate tool for a given job, and other similar tools can also be used to achieve the final result. I strongly believe in tool reuse, rather than inventing a solution using a fancier tool.

Linux and Unix are great at tool reuse, and include some very powerful single-purpose tools. In many cases, threading together multiple tools via scripts can quickly produce a working solution. One can then optimize parts of the solution, using other tools or languages.

My toolbox contains shell scripting, Python, Java, PERL, JavaScript, LUA, C, C++, ANTLR, XML, JSON, SQL, GDB, Valgrind, Wireshark and various other mature frameworks and stacks. I'm also learning Go and Dart, to see where I can apply them in the future. I wish there were more than 24 hours in a day, since there is so much to learn and try!

ODROID Magazine is now on Reddit!

Check out the ODROID Talk Subreddit at http://www.reddit.com/r/odroid