Cluster User and File Management • Remote desktop for ODROID U3

ODROID Magazine

Year One
Issue #7
Jul 2014

WEATHER ENGINEERING

HARDKERNEL'S NEW WEATHER BOARD DEVICE LETS YOU MONITOR WEATHER CONDITIONS WITH YOUR ODROID

CONFIGURE YOUR UNMANNED GROUND VEHICLE INTERFACE

- PORTABLE SOLAR COMPUTING WITH AN ODROID U3
- MAKE AN ODROID X/X2 HARDWARE DUAL BOOT SWITCH
- BUILD NATIVE ANDROID APPLICATIONS WITH RED

OS SPOTLIGHT: GAMESTATION TURBO

CONSOLE EMULATION AND MEDIA PLAYBACK FOR YOUR ODROID

What we stand for.

We strive to symbolize the edge 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://www.pollin.de/shop/suchergebnis.html?S_TEXT=odroid&log=internal`

EDITORIAL

Our Robotics columnist, Chris, has released the next installment of his Unmanned Ground Vehicle article, and shows us in this issue how to interface with the motors and sensors. We're almost ready to start building our own version of this awesome robot! We also take an inside look at GameStation Turbo, the popular gaming and console emulation OS image for the ODROID X and U series. Our HPC columnists Cooper and Anthony outline methods of keeping a High Performance cluster of ODROIDs working smoothly with an in-depth article about best practices for HPC user and file management, and we share a great home project where an ODROID-U3 is powered using only solar energy, which stays running even when the weather doesn't cooperate.

Speaking of weather, the new ODROID Weather Board is Hardkernel's latest ODROID-SHOW peripheral, and bundles 6 sensitive environmental sensors at an affordable price: UV index, barometric pressure, altitude, relative humidity, illumination and temperature, and is fully compatible with the Arduino programming system. Purchase your own Weather Board from the online Hardkernel Store at http://bit.ly/1wtPdgP.

Hardkernel will be exhibiting at ARM Techcon on October 1-3, 2014 in Santa Clara, California, USA. If you live in the Silicon Valley or Bay Area, register before August 8th and receive a FREE pass to the Expo ($59 at the door) so that you can come hang out at the Hardkernel boot for an opportunity to chat with Justin, Lisa, Robroy and Mauro! More information about ARM Techcon can be found at http://www.armtechcon.com, and visit http://ubm.io/1qPo8Ci to get your complimentary early bird pass. We'd love to see you there!

Recently joining the ODROID Magazine team is our new Assistant Art Editor, Nicole Scott, who lives in the San Francisco Bay area and has experience in video, audio, music, web, and multimedia production. Welcome aboard, Nicole!

Would you like to write for ODROID Magazine? We are currently looking for several columnists to write on a variety of topics such as Linux basics, Kali hacking, ARM world news, Bash scripting and any other subject that you wish to share. Gift packages are awarded for published articles, and regular columnists can earn top-end hardware! Email your article in Google Docs or Office format odroidmagazine(at)gmail.com .

Look for more exciting announcements in the next issue, including the newest generation of the ODROID-XU family, as well as an affordable 9" Touchscreen USB LCD monitor, both of which are available for purchase at the Hardkernel store.

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, Art Editor

Is counting the days to his long earned vacations! While that doesn't happen, he is working hard to keep things going on smoothly at work and play. And loony as always.

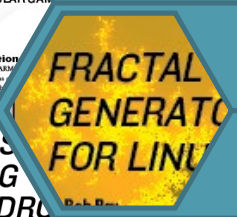
Manuel Adamuz, Spanish Editor

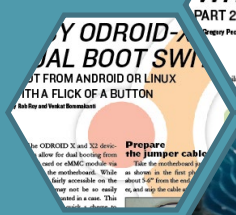
I am 31 years old and live in Seville, Spain, and was born in Granada. I have recently become a father, and my son is now 5 months old. It is an incredible experience! A few years ago I worked as a computer technician and programmer, but my current job is related to quality management and information technology: ISO 9001, ISO 27001, and ISO 20000. I am passionate about computer science, especially micro computers such as the ODROID, Raspberry Pi, etc. My other great hobby is mountain biking, and I occasionally participate in semi-professional competitions.

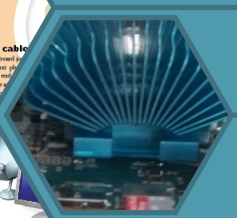
Nicole Scott, Assistant Art Editor

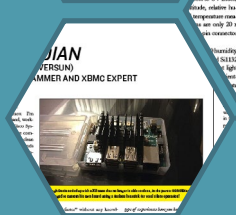
I am currently a Digital Strategist and Transmedia Producer who specializes in online optimization and inbound marketing strategies, social media directing and team coordination, as well as media production for print, TV, film, and web. I also have experience in graphic and website design, social networking management and advertising, 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. Check out my web page at http://www.nicolecscott.com.

INDEX









HIGH PERFORMANCE COMPUTING AT HOME

CLUSTER USER AND FILE MANAGEMENT

by Cooper Filby and Anthony Skjellum - Runtime Computing Solutions LLC
http://www.runtimecomputing.com

In the February and March 2014 Articles of ODROID Magazine, we outlined how to configure the networking for a cost-effective and efficient ODROID cluster managed by a central head node. In this article, we continue to expand the robustness of our cluster by incorporating services such as LDAP and NFS/AutoFS into our head node and compute nodes in order to accommodate a wider range of uses and users. With LDAP configured, we will be able to have users log into any node of the cluster with the same credentials and permissions which, when combined with NFS/AutoFS, will permit us to configure network-mounted home directories and shares, creating a uniform user experience when connecting to any node in our cluster.

To try out this HPC project, you'll need:

1. 2x ODROIDs - in our examples, we will be using XU+Es running Ubuntu 13.09 server. More ODROIDs can easily be included as well to create a bigger cluster

2. 1x Ethernet Switch (preferably GigabitEthernet, also called 1000-BaseT)

3. 3x Ethernet Cables (plus 1 cable for each additional ODROID)

4. 1x USB Gigabit Ethernet Adapter (ideally one for each ODROID)

Cool looking cluster huh? Well, after reading this article, you'll know how to build one too!

5. A machine with a display with a GUI, network connectivity and Apache Directory studio

Configuring SLDAP

Up until this point in previous articles, we have been operating under the assumption that anyone using our ODROID cluster would rely on the default odroid user account that comes with precompiled Ubuntu Server images. While using a single login is sufficient for one (or maybe two) users, this single login strategy is hardly a secure and scalable model for an expanding cluster. To address this need, we will install and configure a Lightweight Active Directory Protocol (LDAP) service for managing users that are a part of our cluster.

Of course, if you already have an LDAP service running on your network, you could go ahead and skip this step and configure your nodes to use that for user authentication instead. However, before installing and configuring LDAP, we will need to decide how we wish to organize our user accounts. While LDAP configurations will vary by organization, it's a common practice to use the organization's domain name for as the LDAP search base, as we will show below. To get started, we will begin by installing the LDAP service on our head node as follows:

```
sudo apt-get install slapd ldap-utils
```

During the installation, you will be given a chance to select your administrator password for the LDAP service. Once the install completes, our LDAP server will be a blank slate for our further setup. To get started with our configuration, we run:

```
sudo dpkg-reconfigure slapd
```

While the options may look confusing at first, for our purposes we can choose the default options for everything except the DNS Domain name, Organization name, and Administrator password fields. What you select for the DNS Domain name and Organization name are up to you, in this case we used 'ocluster.rtcomputingsolutions.com' for both entries (you will use your own domain, and prefix it with the name of your cluster, which might be odroid or another name you prefer).

Success! You now have a running LDAP server! Unfortunately, we don't have any real information stored in the LDAP service just yet, such as Users or Group information. While there are a number of ways that you could go about creating containers for users and groups, like using command line tools, we personally prefer using Apache Directory Studio (`http://directory.apache.org/studio`) to take care of our initial setup. Afterwards, it will be much more convenient to utilize command line scripts, such as the one included in the Appendix of this article.

Users and Groups

A good next networking question toNow that we have our service started, we need to start adding user accounts (authorization) to allow people to sign in using those credentials (authentication). To get started with using Apache Directory Studio, we need to create a new connection to our LDAP server, which we can do using the following steps:

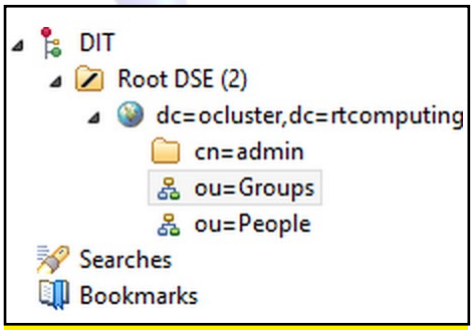
1. In the bottom left corner of ApacheDS, select the LDAP icon to create a new Connection.
2. Input a name for the connection, then the hostname or IP for your head node, and select 'Check Network Parameter' to verify that ApacheDS can communicate with your server, and hit 'Next'. If ApacheDS is unable to connect, check that your machine can ping your head node and that slapd is running.
3. On the Authentication screen, enter the admin login credentials using the Domain Component (dc) information that you entered during the setup of slapd. Using the slapd configuration listed above, we entered 'cn=admin,dc=ocluster,dc=rtcomputingsolutions,dc=com' and the corresponding password we entered when we ran dpkg-reconfigure slapd (the series of dc= sections will differ depending on what you input during setup). After verifying that 'Check Authentication' is successful, hit 'Finish'.

Again, your specific LDAP setup will vary, but in this instance we will create two Organizational Units (OU's), called People and Groups using ApacheDS. To add a new entry, right click on our Domain Component (dc, ex:'dc=ocluster,dc=rtcomputingsolutions,dc=com') entry in the LDAP Browser and select: New -> New Entry.

We can then create our People and Groups OU's using the following steps:

1. Select 'Create Entry from scratch' and select 'Next >'
2. Find and select 'organizationalUnit' from the 'Available object classes' list, select 'Add' and then 'Next >'.
3. On the row containing 'RDN', enter 'ou' into the first field, and the name of the OU you are creating in the second field after the '=' (in this case, either People or Groups) and hit 'Next >'.
4. Review the Attributes and select 'Finish' when you are satisfied with the entry.
5. Repeat steps 1-4, entering the other organizational unit name on step 3.

If steps 1-5 went well, you should now see an LDAP Browser view that looks something like this:



LDAP Browser in action

With these two organizational units in place, we can go ahead and start creating our users and groups for our cluster. In this example, we are going to create one cluster admin user and two cluster groups using ApacheDS, and then provide a sample python script that can be used to create new accounts from the command line.

First, we will create two groups, one called 'ouser' and one called 'osudo', giving us some control on who has administrative access throughout our cluster through the eventual use of sudoers files. We can create these groups following the steps below:

1. Right click 'ou=Groups' and select 'New -> New Entry'
2. Select 'Create entry from scratch' and 'Next >'
3. From the 'Available object classes' list, select 'posixGroup', 'Add' and then 'Next >'.
4. On the RDN row, enter 'cn' = 'osudo', and select 'Next >'.
5. Next, you will be prompted for

a gid number for this group, what you pick is up to you, although we highly recommend you use a number at least above 1000 to avoid potential conflicts. In this case, we will use '2001'. After entering your chosen value, hit 'OK'.

6. Confirm your attributes and hit 'Finish' to create the group.

7. Repeat steps 1-6 to create the second group, using 'ouser' and '2002' for the common name and gid respectively.

With these created, let's go ahead and create an LDAP administrator account, 'oadmin':

1. Right click 'ou=People' and select 'New -> Entry'

2. Select 'Create entry from scratch' and 'Next >'

3. From the 'Available object classes' list, select and 'Add' both the 'posixAccount' and 'inetOrgPerson' fields, and then 'Next >'.

4. On the RDN row, enter 'uid' = 'oadmin' (where the uid corresponds to the username) and select 'Next >'.

5. The next window will prompt you to enter the uidNumber. Again, we recommend you pick a number at least above 1000 to avoid conflicts. In this instance, we use '2200'. After entering your chosen value, hit 'Ok'.

6. On the Attributes menu, you will notice several fields highlighted red that we will need to populate before creating the account. First, we can populate the Common Name (cn) field with a name for our user, in this case 'Ocluster Admin'. The gidNumber field is the primary group for this user, which in this case will be '2001' for the osudo group. Next, we need to fill out the homeDirectory field, which will tell the LDAP account which directory to use to access user specific files. In this instance, we will use the directory '/nethome/oadmin', where /nethome is the folder that will we configure with AutoFS and NFS. The last field is for the Surname (sn) field, where we used 'Admin'.

7. Finally, we need to add the 'userPassword' and 'loginShell' attributes to our oadmin account. We can add these attributes by right clicking on the window and selecting 'New Attribute', typing in the desired attribute name and hitting 'Finish'. In the case of the userPassword attribute, you will be prompted to enter in your password and your Hashing Method (what you pick is up to you, but we strongly recommend against using Plaintext). You can then add the loginShell attribute by repeating the above step, using '/bin/bash' as the corresponding value.

8. After you confirm that everything looks correct, hit 'Finish' to create your new user.

If everything checks out, the 'oadmin' account should appear under the People ou in the LDAP Browser. You can then go ahead and create a few more accounts as you see fit, or you can check out the appendix for a simple Python script that you can use to create users outside of Apache Directory Studio. Before we can get to using these accounts, however, we need to set up our NFS server and configure our nodes to use both LDAP and AutoFS.

Configuring NFS

The Network File System (NFS) service is essential to any well rounded cluster. With NFS properly configured we can make use of network mounted home directories, meaning users' files follow them around the different nodes in the cluster. Furthermore, we can use NFS to setup general purpose shared folders for compiled binaries or cluster configuration settings. To get started with setting up our NFS server, run:

```
sudo apt-get install nfs-kernel-server
```

It's worth noting that in the past, we have encountered issues attempting to install nfs-kernel-server using apt. If this is the case for you, try building and installing unfsd from source, which is available online. Next, we need to create two folders that we can export from the head node and mount on the compute nodes, nethome and opt:

```
sudo mkdir -p /srv/nfs4/nethome
sudo mkdir -p /srv/nfs4/opt
sudo chmod 777 /srv/nfs4/nethome
```

In this case, we're going to replace the existing /opt directory with our NFS directory, and nethome will be used to store our home directories. With these directories created, we now need to edit /etc/exports using our favorite text editor (as root) and add the following lines:

```
/srv/nfs4/nethome
192.168.128.0/24(rw,sync,no_root_
squash,no_subtree_check)
/srv/nfs4/opt
192.168.128.0/24(rw,sync,no_root_
squash,no_subtree_check)
```

In essence, the first segment denotes the folder to share, while the second segment denotes the hosts to export to (our internal network) and the options for exporting. With all these settings in place, we can now go ahead and restart the nfs service:

```
sudo /etc/init.d/nfs-kernel-server restart
```

Finally, we can 'mount' these directories on the head node with the following commands:

```
sudo ln -s /srv/nfs4/nethome/ /
nethome
sudo rm -rf /opt
sudo ln -s /srv/nfs4/opt/ /opt
```

With this configuration in place, we can now go ahead and configure our nodes to use AutoFS and LDAP.

Configuring Client LDAP Authentication

Fortunately, configuring LDAP on all of our cluster nodes is a good deal simpler than setting up the LDAP service on the head node; we can get started by installing the nscd and the ldap client meta package:

```
sudo apt-get install ldap-auth-
client nscd
```

The ldap-auth-client configuration installation will prompt you to fill in information about your LDAP server, we've shown our answers below:

```
 LDAP Server URI: ldap://odroid1.
ocluster.rtcomputingsolutions.
comSearch Base: dc=ocluster,dc=rt
computingsolutions,dc=com
LDAP Version: 3
Local root Database admin: No
LDAP database require login?: No
```

We actually encountered some difficulties configuring LDAP on the head node when using the fully qualified domain name. To get around this, we just put localhost, although your mileage may vary. You may have also noticed that we changed up the hostnames and domain names of our cluster, where odroid1 is now our head node instead of odroid-server0, and are compute nodes are named odroid2 and up.

After the install finishes, we will need to run the following commands to finalize configuration of our cluster:

```
sudo ln -s /lib/arm-linux-gnue-
abihf/nss_ldap.so /lib/libnss_
ldap.so.2
sudo auth-client-config -t nss -p
lac_ldap
sudo pam-auth-update
sudo reboot
```

If all went well, you should be able to log in using the credentials for the oadmin account, or any new accounts we configure. Of course, we still don't have any home directories created for our users; fortunately, LDAP provides us a way to automatically create home directories. Create the file /usr/share/pam-configs/my_mkhomedir as root, and add the following contents:

```
Default: yes
Priority: 900
Session-Type: Additional
Session: required pam_mkhomedir.
so umask=0077 skel=/etc/skel
```

Run sudo pam-auth-update to enable the settings. Upon logging in as oadmin, your home directory will be created for you automatically under /nethome/oadmin. Note that we can get away with only having this enabled on the head node, as external users have to connect to the rest of the cluster through the head node (although having this enabled on all cluster nodes won't create any problems).

Configuring Client AutoFS

AutoFS is often used in conjunction with NFS services, as it allows us to automatically mount remote resources upon rebooting. To get our NFS shares mounted on our compute nodes (we've already 'mounted' them on our head node), we need to start by installing autofs:

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

Next, we need to create the nethome directory that will be one of our NFS mount points:

```
sudo mkdir /nethome
sudo chmod 777 /nethome
```

Lastly, we need to modify our autofs configuration files to mount the opt and nethome directories. The files we've modified (or created) and

BASH BANG!

BECAUSE COMMAND LINE NEVER GETS OUT OF FASHION

by Bruno Doiche

After getting used to running your own commands via terminal, nothing could be better than having some extra tricks up your sleeve, right? Take notes on this pair of time savers and make it look like you learned Linux years ago!

1. Use !! to refer to the previous command:

```
$ apt-get install package
$ sudo !!
```

2. Look for a certain command but don't execute it:

```
!<command>:p
```

FIND WHAT YOU NEED WITH GREP

THE QUICK WAY TO LOOK FOR CONTENT

by Bruno Doiche

Need to edit that system file for a certain string, but don't remember which file was that? Grep your way out of this simply by typing:

```
grep 'string' <path>
```

You can even expand your search with the -i and -R arguments, -i to ignore case, -R to find recursively on directories. Just remember that grep doesn't distinguish which type of file it is looking at, so exclude your multimedia file systems to save time.

their contents have been shown below:

```
/etc/auto.master
+dir:/etc/auto.master.d
+auto.master
/nethome    /etc/auto.nethome
--timeout=60
/-        /etc/auto.direct
--timeout=60
```

```
/etc/auto.nethome
*    odroid1.ocluster.rtcomput-
ingsolutions.com:/srv/nfs4/neth-
ome/&
```

```
/etc/auto.direct
    /opt    odroid1.ocluster.
rtcomputingsolutions.com:/srv/
nfs4/opt
```

With this information in place, we can apply these autofs settings with the following command:

```
sudo service autofs restart
```

If everything is configured correctly, you should be able to add files into /opt and /nethome (pending permissions) and see them on other nodes in the cluster that are mounting the same autofs share. If AutoFS doesn't seem to start up when you restart the machine, you can add '/usr/sbin/service autofs restart' to /etc/rc.local.

Conclusions

Some work is required up front to make our cluster significantly more scalable in the long run (and low effort to maintain in particular), and unlike the steps and configurations shown in our previous articles, there is a wider margin for error and a potential need for debugging. Even if you aren't building a massive cluster, experience with LDAP, NFS, and AutoFS are often part of a Linux administrator's toolbelt, making this a worthwhile effort in its own right. With these services in place and the addition of three more compute nodes, we now have an ARM infrastructure we can rightly call a cluster. In future articles, we will demonstrate various ways to make use of our newly commissioned cluster, such as utilizing MPI, and further ways to improve it, such as using Puppet.

Acknowledgements

We would like to give special thanks to Dr. Kenneth Sloan, Director of the University of Alabama at Birmingham's 3D Print Lab (http://bit.ly/1j2AOjI) in the Department of Compuer and Information Sciences. With their support, we were able to 3D-print the cluster stand shown in the picture on page 2.

Sample Code

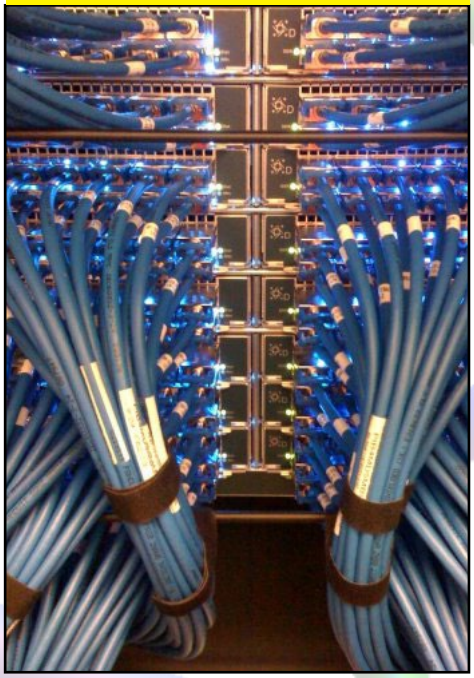
Note this is just an example that we used previously. Feel free to expand upon it to create new users. This also requires the 'python-ldap' package.

```
addUser.py
#!/usr/bin/python
import base64
import getpass
import hashlib
import ldap
import ldap.modlist as modlist
import os
import random
import string
import sys

# LDAP Vars
HOST="ldap://localhost:389"
LDAP_BASE = "dc=ocluster,dc=rtcom
putingsolutions,dc=com"
ADMIN_CN = "cn=admin," + LDAP_
BASE
PEOPLE_BASE = "ou=People," +
LDAP_BASE
ADMIN_PASS = getpass.
getpass("LDAP Admin Password:")

# Group Vars
OUSER_GID = "2002"
```

```
def addUser():
    """"""
    try:
        ldapCon = ldap.
initialize(HOST)
        ldapCon.simple_
bind_s(ADMIN_CN, ADMIN_PASS)

        attrs = getUserAttrib-
utes()
        ldif = modlist.
addModlist(attrs)
        ldapCon.add_s("uid="
+ attrs['uid'] + "," + PEOPLE_
BASE,ldif)
        ldapCon.unbind_s()
        print "Account created."
    except ldap.LDAPError, e:
        print e
        sys.exit()
    print "Done."



def getUserAttributes():
    """"""
    attrs = {}
    attrs['objectClass'] = ['in-
etOrgPerson', 'organizationalP-
erson',
        'person', 'posixAccount',
'shadowAccount', 'top']
    attrs['cn'] = raw_input("User
Full Name: ")
    attrs['uid'] = raw_
input("User uid: ")
    attrs['uidNumber'] = raw_
input("User uidNumber: ")
    attrs['gidNumber'] = OUSER_
GID
    attrs['sn'] = attrs['cn'].
split(' ')[-1]
    attrs['userPassword'] =
generateLdapPassword(getpass.
getpass("User Password: "))
    attrs['homeDirectory'] = '/
nethome/' + attrs['uid']
    attrs['loginShell'] = '/bin/
bash'
    attrs['mail'] = attrs['uid']
+ "@rtcomputingsolutions.com"
```

```
    return attrs

def printUserAttributes(attrs):
    """"""
    for key in attrs.keys():
        if key == "objectClass":
            print attrs[key]
        else:
            print key + "= " +
attrs[key]

def
generateLdapPassword(password):
    """"""
    salt = generateSalt(6)
    m = hashlib.sha1(password)
    m.update(salt)
    return "{SSHA}" + base64.
b64encode(m.digest() + salt)

def generateSalt(size=10,
chars=string.ascii_letters +
string.digits):
    """Generate random temporary
password for web user"""
    return ''.join(random.
choice(chars) for x in
range(size))

if __name__ == "__main__":
    addUser()
```

Things got a little out of hand when we built our own ODROID cluster in the offices.



USING XBOX 360 CONTROLLERS WITH ANDROID

CONFIGURE THEM WITH RETROARCH AND PLAY THOUSANDS OF GAMES

by Rob Roy

The Xbox 360 controller works naturally with the Android operating system and can be used to navigate menus, launch applications, and most importantly, play games. I evaluated several controllers and game emulators with Android running on the ODROID, including a PS3 bluetooth controller, Wiimote, and an Xbox 360 Wired and Wireless controller. Out of all of the models that I tested, the Xbox 360 wireless model was the most convenient and widely compatible gamepad available for the Android platform.

If you already have some 360 wireless controllers from an Xbox, add an Xbox 360 USB receiver to connect them with the ODROID (http://amzn.to/1bJZv6q). The wireless interface supports up to 4 controllers per receiver.

To connect the controllers, first press the "connect" button on the USB wireless receiver until the green light flashes. Then, press the "connect" button on the controller, which is located between the left and right trigger buttons next to the charging port, directly in front of the "back" button. The 1-2-3-4 indicator will flash its lights in a circle, and eventually flash a single light slowly to indicate that a connection has been made. Although the controller normally stays solid when connected when connected to an Xbox, the 1-2-3-4 indicator light will continue to flash even when the controller is connected. Repeat the whole process until all controllers have been registered with the receiver.

To test whether the controllers are working, go to the Android desktop and use the top left joystick to navigate the main menu icons, the X button to launch applications, and the B button to go back to the previous screen. The controller also works in XBMC without any additional configuration. Once the controller is responding well, you're ready to play some games!

The most versatile and full-featured console emulator for Android is Retroarch, which is very easy to connect with the Xbox 360 controllers. It's one of the best gaming emulators available on the Play Store, and includes these consoles: Atari 2600, PlayStation 1, Super Nintendo, Nintendo Entertainment System, GameBoy, GameBoy Color, GameBoy Advance, Arcade, Neo Geo Pocket, Virtual Boy, Sega Genesis/Mega Drive, Sega Master System, Quake, Doom, Sega Game Gear and more. Retroarch also allows playing up to 4 controllers, which makes for some very fun games of multi-player Mario Kart!

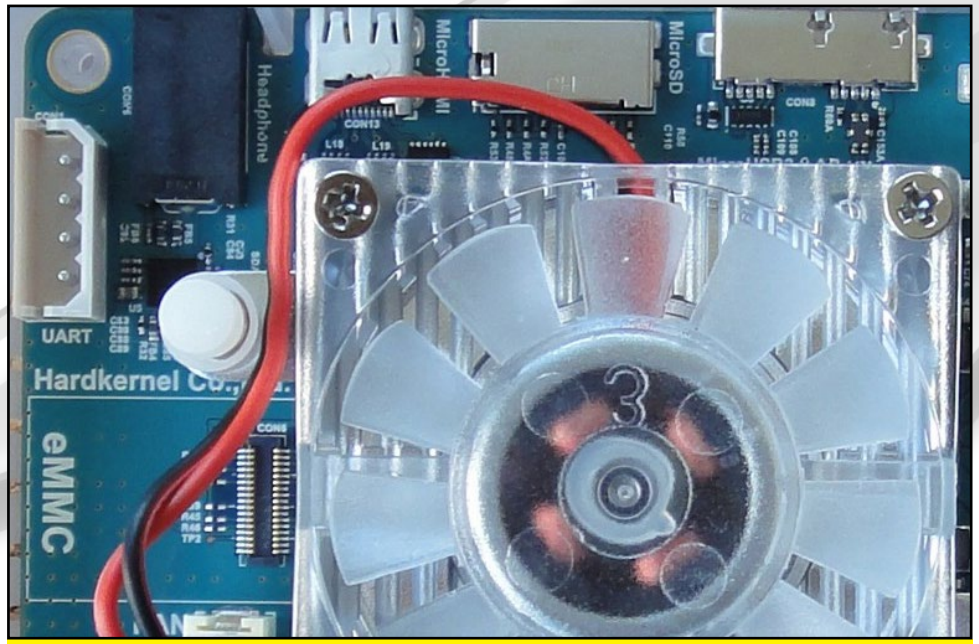
To configure Retroarch for the Xbox 360 controllers, select "Input Options" from the Retroarch main menu, turn off the "Touchscreen Overlay", then scroll down to "Player 1 Custom Binds". Select each on-screen button and press the corresponding button on the 360 controller that you wish to use to control Player 1. Repeat the process for all other controllers (to a maximum of 4). After the buttons are mapped, you can play any game system that Retroarch supports without reconfiguring the joysticks.

CONVERT AN OS IMAGE TO RUN ON AN ODROID-XU

UPGRADE YOUR LINUX SOFTWARE TO MATCH YOUR SHINY NEW HARDWARE

by Mauro Ribeiro

This article describes how to take a working X or U series OS Linux image, such as Ubuntu or Debian, and convert it to run on the XU. This can be useful when upgrading your own hardware to the XU, or for taking an image that is only available for the previous generation of ODROID hardware and upgrading it to work with the latest ODROID models such as the XU, XU-Lite, and XU+E.



The ODROID-XU is the top of the line in Hardkernel's family of SBC hardware. Most of the Linux software that runs on the X and U series hardware can be converted to run on the faster XU model.

1. Download, copy or create an Ubuntu image file from an SD card or eMMC that is designed to run on the X or U series of ODROID hardware, such as the U3 or X2.

2. Flash the image as you normally would, to an SD card

```
dd if=XXXXX.img of=/dev/sdX bs=4M
&& sync
```

3. Download a new bootloader and sample boot.ini file.

```
wget builder.mdrjr.net/tools/
uboot-xu.tar
```

4. Unpack the uboot-xu.tar file.

```
tar xf uboot-xu.tar
```

5. Fuse the sdcard with the newer bootloaders.

```
sh fusing.sh /dev/sdX
```

6. Inside the bootloader package by copying the boot.ini and u-boot.bin files to the SD card's FAT32 boot partition.

```
cp boot.ini /media/boot/
cp u-boot.bin /media/boot/
```

7. Download a kernel package for the XU.

```
wget http://builder.mdrjr.net/
kernel-3.4/00-[...]xu.tar.xz
```

8. Unpack the kernel package and copy the files to the correct directories.

```
tar -Jxf odroidxu.tar.xz
```

9. Copy boot/zImage to the fat partition of the SD card, then copy the lib directory over top of the /lib folder of the ext4 partition of the SD card.

10. Remove the old boot.scr file from the FAT32 partition.

11. Attempt a boot, which should probably work just fine.

12. Enable the HDMI driver. You may need some dependencies here, depending on your environment.

```
git clone --depth 0 https://
github.com/hardkernel/linux.git
-b odroidxu-3.4.y odroidxu-3.4.y
cd odroidxu-3.4.y/tools/hardker-
nel/exynos5-hwcomposer
sh autogen.sh
./configure --prefix=/usr
make -j5 && make install
```

13. Create an autostart script for exynos5-hwcomposer by creating the file /etc/init/exynos5-hwcomposer.conf with the following content:

```
description “Exynos5 HW Composer”

start on started lightdm

# start on runlevel [2345]
exec /usr/bin/exynos5-hwcomposer
```

14. Enable the HDMI driver. You may need some dependencies here, depending on your environment.

```
git clone --depth 0 https://github.com/hardkernel/linux.git -b odroidxu-3.4.y odroidxu-3.4.y
cd odroidxu-3.4.y/tools/hardkernel/exynos5-hwcomposer
sh autogen.sh
./configure --prefix=/usr
make -j5 && make install
```

12. Create an autostart script for exynos5-hwcomposer by creating the file /etc/init/exynos5-hwcomposer.conf with the following content:

```
description “Exynos5 HW Composer”

start on started lightdm

# start on runlevel [2345]
exec /usr/bin/exynos5-hwcomposer
```

15. Fix Xorg.conf by making sure your /etc/X11/xorg.conf look like this:

```
Section “Device”
        identifier “FBDEV”
        Driver “fbdev”
        Option “fbdev” “/dev/fb0”
EndSection

Section “Screen”
        identifier “Default Screen”
        Device “FBDEV”
        DefaultDepth 24
EndSection
```

16. Reboot. Everything should be working now!

Once the image is confirmed to be running properly on the XU, make sure to create a disk image of the working OS. If the setup doesn't run, try repeating the steps with a fresh copy of the original X or U series image. If you encounter problems and need assistance with troubleshooting, please post your question on the Hardkernel forums at http://forum.odroid.com.

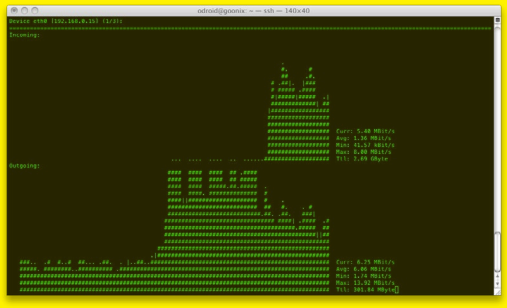
CONTROL YOUR BANDWITH

KNOW YOUR INTERFACE I/O STATUS

by Bruno Doiche

Information on your network throughput is easy as looking on your Gnome system monitor. But what about a headless ODROID that only has access via terminal? Fear not, install nload and keep your network interface load on track.

```
sudo apt-get install nload
```



SECURITY

CONTROL YOUR USER ACCESS

by Bruno Doiche

Keep your server security tight by configuring it properly. If there is no need for a certain user to access your machine via SSH, be sure to edit your /etc/ssh/sshd_config file and add the following strategy:

```
AllowUsers <users to be allowed to logon>

DenyUsers <users to be denied logon>
```

Why deny user access instead of just deleting the user? Sometimes a user only needs SMB or FTP acess to your server. This best practice gives peace of mind when publishing your system.

OS SPOTLIGHT: GAMESTATION TURBO

AN INSIDE VIEW OF THE POPULAR GAMING AND MEDIA CENTER IMAGE

by Tobias Schaaf

GameStation Turbo for the U2/U3/X/X2 may be downloaded from http://bit.ly/VolpVL. Watch the ODROID forums at http://forum.odroid.com for updated versions, including the latest release of the PPSSPP PSX emulator!

One of the biggest projects that I am working on for the ODROID community is the ODROID GameStation Turbo image, which is a Linux frontend for both games and media playback. It's intended as an total entertainment system which allows you to control your ODROID just by using a game controller in your hand, without ever having to touch the keyboard in order to watch movies, listen to music, or play your favorite games.

For a better understanding of the usefuleness of the image, I want to give you an inside view on how the image was created, what motivation I had, and how you can adapt it to meet your own needs.

Motivation

The first ARM based device that I considered was actually the Open Pandora, but by the time I was ready to to buy one, it was not available. However, even when it came available again, it was so expensive that I couldn't afford it. Finally, when I had enough money, I was already skeptical and was looking into other options.

The Pandora board is an ARM-based single core device with just 1GHz and only 512MB RAM for $700, so was it really worth it? Well, although the community was and continues to be awesome, and it's a fully portable device (like a Nintendo DS), it was way too expensive, in my opinion, for what it could do. By that time there were better devices available, including the ODROID.

After seeing the ODROID-X2 in an article on a German IT News page, I got really hooked up to it. By the time Ubuntu was announced for the ODROID, I bought myself an X2.

However, what I wanted the Pandora for was to play games, and the ODROID didn't have too many games at that time (2012). As my nephews got older, I figured that I could make something really nice for them that would grow with them as they grew. First, the ODROID could be a console for playing games, and later, it would function as a PC in order to do homework and learn Linux. That was my goal and motivation for creating the image.

Baldur's Gate is one of the PSP games that is now supported by the latest version of GameStation Turbo, thanks to the new release of PPSSPP.

Steps to Success

The first step towards achieving that goal was to generate content, so lots of games and emulators had to be ported to the ODROID. If you read the ODROID Forum's Ubuntu (All Linuxes) section, you will find many games and programs there that I ported myself. It was hard work, since I went from knowing basically nothing about porting games or compiling software on Linux to what I know now. ODROID was a great help in learning new skills and getting better at knowing that kind of Linux stuff. Now, I know how to optimize certain programs, how to set

different optimization flags, and when those flags are needed. I learned more about how ARM CPUs work and, especially the hardware differences between the Hardkernel boards.

My first project was then to port lots of games, and compile some emulators as well. If you've read my columns from the previous issues of ODROID Magazine, you will find lots of informations on what games are actually running on the ODROID, and it keeps getting bigger.

The next big step was to make it easy, even for children, to use and work with Linux and play games. I started building Gamestation Turbo from the Linaro Ubuntu 12.04 Image. I preferred that over all other operating systems because of its Unity Desktop.

Unity is easy to use and understand even for people that never used Linux before. It might not be the best desktop environment for all applications, but it's colorful, and easy to handle. For someone who has never used Linux before, it's a very nice way to get started with it.

My first approach was to get the programs easy enough to run on all system. I gave all applications and games that I created a .desktop icon file, so you can find it in Unity or just place a shortcut on the desktop. This worked fine for games, but not for emulators, since emulators normally use their own file browser interface to load the ROMs. Although adults might be able to handle starting all of the games manually, kids will have no clue what certain words mean, and it's hard to see which games are available, or what to search for on Unity. It was immediately clear that I needed some kind of front end in which to start the games.

I had already used XBMC on an old PC that functioned as a Home Theater PC (HTPC), and subsequently discovered a nice XBMC addon called Rom Collection Browser (RCB). RCB allows you to organize the emulator ROMs in the same way that you can organize your

GameStation Turbo lets you choose from a variety of emulators as well as play 1080p videos, all from within XBMC using just a gamepad controller.

video collection. It's even able to download preview images and covers and gave a short description to the games, just like video services do for movies.

Knowing this, the idea came up to use XBMC as a frontend and set it up in a way for children to play and have fun with, or better to say to set it up in a way that even a child could play with it. During that time, hardware accelerated XBMC and video playback was out of question since it development hadn't yet been completed.

The XBMC version that came with Ubuntu 12.04 was XBMC 11 (Eden), which was working, but not very fast due to software decoding. Although the Menu was working smoothly, video playback was not smooth. Still, it was working well enough that I could test out the Rom Collection Browser, and experimented with how to set everything up. When the first image of XBMC 12 (Frodo) for ODROID came out, it still did not support hardware-accelerated movie playback, but did come with OpenGL ES 2 support.

Things got a little difficult to manage around that time, since compiling Hardkernel's XBMC source code didn't work for me, and the version provided has no joystick support, which I considered very crucial to my plans I decided that, since it was planned as a gaming platform, video playback was not the most important feature, and you still could play everything that was not HD smoothly as long as it was 720p or lower. For children, it generally doesn't matter if their favorite anime or cartoon is in HD or just SD.

Well, it was about that time that a working hardware accelerated XBMC image was released, and I was able to rebuild the image with the necessary joystick support. Shortly before I released the first version of GameStation Turbo, I moved over to a fully working XBMC version.

Included Parts

After I decided how the image should work, it was important to put all the tiny pieces together into a nicely packaged image, and for this, some different kinds of programs were necessary.

The first priority was the Operating System, which had to be very stable, easy to maintain, and with an interface that many people are already familiar with. The only choice here was between Ubuntu 12.04 and Debian Wheezy. Every other image was either unstable (Debian Jessie/Sid) or wouldn't be supported for very long (Ubuntu 13.04 or newly released 13.10).

Ubuntu 12.04 is an LTS version that is supported until 2017, which is always

good, however, Debian Wheezy outperformed Ubuntu 12.04. I also found that while developing for Debian Wheezy, the programs were most likely to run on Ubuntu 12.04 and higher without any issue, but not the other way around. So, I decided to use Debian Wheezy and LXDE, which uses less than 150 MB RAM even with XBMC and a couple of other programs running. After that, it was a question of putting together the right kind of software to turn the ODROID into a gaming machine.

Rom Collection Browser

I used Rom Collection Browser as a base to install different kind of emulators such as Retoarch, Mednafen, PPSSPP and ScummVM. Once the basic setup was done, it turned out that not everything was working with a gamepad out of the box, so I added `antimicro` which is able to map certain keys to a joystick button to fill the gaps where the joystick drivers did not work.

I also maintain my own kernel builds and include the header files as well, since some parts of the kernel provided by Hardkernel did not meet my needs, and header files were not included. Besides that, there was a huge space difference between hardkernel's kernel modules and the one that I produced. The size was Hardkernel's build was over 300MB, but mine was only 16MB of my own build, which was achieve just by stripping the modules. My scripts also allows users to install or uninstall kernel packages, instead of just copying the kernel directly over the existing files.

Complications

The biggest problem for me was how to get all the parts to work with each other, and make it easy for people to use the image, even if they do not have knowledge on how to set it all up. The Rom Collection Browser was somewhat difficult to use for a beginner, since you had to choose the emulator, starting parameters and give the ROM files standard extension to set it up and get it to run. So I had to come up with a system that made it rather easy for a user to deal with that.

There was another issue. I wanted to have full Joystick (GamePad) control, but some emulators required keys as well, such as Retroarch and Mednafen which required the ESC-key to end the current game and go back to XBMC, and also MAME games which required to enter an "OK" to continue.

Configuration

One problem with preconfiguring the Rom Collection Browser was that it requires the full path of where the emulator and ROMs are located, and what file extension is used to search for ROMs. This can be rather confusing for someone that has never worked with the Rom Collection Browser. That's why I pre-selected the emulator and games, and created a folder structure where the ROMs should be placed, in order for the Rom Collection Browser to find the games.

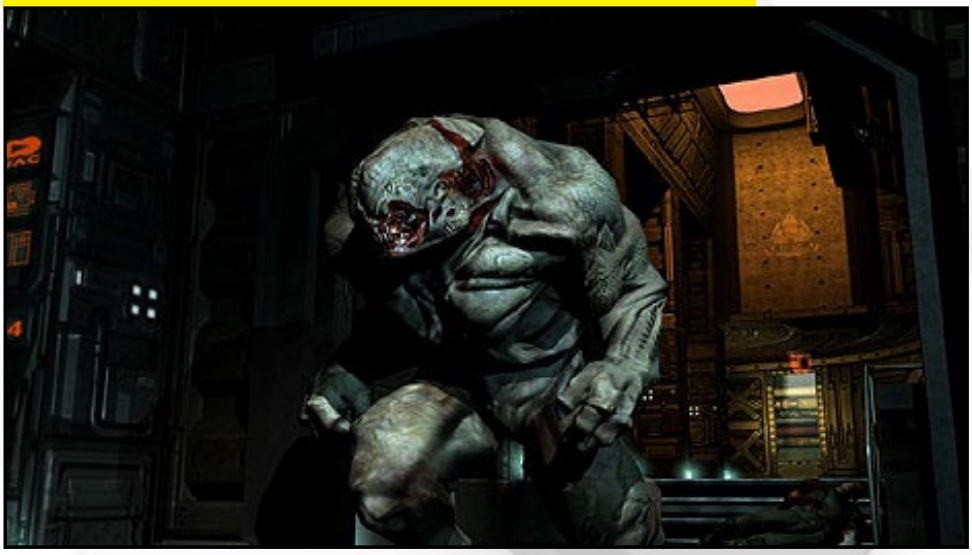
Additional emulators can be added by pressing the C key in the Rom Collection Browser and selecting "Add a new ROM collection". There you have to give the path to the emulator, the path to the ROMs, the path where it should store information and pictures and the extension of the ROMs it should look for.

The configuration file for ROM Collection Browser is stored in

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

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

DOOM 3 shows off the graphics of the ODROID by unleashing this creature inside your computer, then giving you only a crowbar to defend yourself.



Starting an Emulator

Although starting a ROM directly through the emulator will definitely work, it has a couple of disadvantages. First of all, XBMC will still be running in the background and will use some of the resources needed for a better gaming experience. Second, as mentioned before, some emulators need extra keys that are not mapped to a button. If using a joystick that is not supported, you need need `antimicro` to map the buttons for you. If so, you need to make sure that `antimicro` is started when you

need it, which might not always be the case. Directly starting `antimicro` along with the emulator didn't work either.

To solve these and other issues, I let XBMC run a small scriptiInstead of directly starting the emulator. In that script, the emulator is started, which then runs the ROM file which is given to the script as a command-line parameter from XBMC. That way, I can define different steps to make sure the emulator works the best.

Example: This script enables running an SNES game with Retroarch:

```
#!/bin/sh
 /usr/bin/killall -STOP xbmc.bin
 if [ `ps aux | grep antimicro |
grep -v grep | wc -l` -lt 1 ];
then
        antimicro --tray --pro-
file /home/odroid/joydev.xml &
 else
        /usr/bin/killall -CONT
antimicro
 fi
 /usr/local/bin/retroarch -L /
usr/local/share/retroarch/cores/
working/snes9x_next_libretro.so
"$1"
 /usr/bin/killall -CONT xbmc.bin
 /usr/bin/killall -STOP antimi-
cro
```

Reading through the above code, you can see that XBMC is set to suspend mode, which means it won't use any processing power while we run our emulator. After that, I make a check to see whether `antimicro` is running, and either load it with the required profile file, or resume it in case it's still running. Then, I call the actual emulator. Here I can pass command parameters which allows me to configure the emulator. After the emulator is terminated by exiting the emulator, XBMC is resumed and `antimicro` is suspended. Just after the script is completely done, it switches control back to XBMC. This allows for some cleanup work that may be necessary.

Homeworld: The pinnacle of 3D RTS with futuristic spaceships! This game is very fun, and I spent hours playing it when it first came out.

I wrote quite some scripts to adapt to different circumstances. For example, the ScummVM and Amiga script is a little bit more complicated; but all in all it's pretty much always the same.

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

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

FAQ

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

Where do I have to put the ROM files for my games?

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

Is there a way to load ROMs from an external storage?

Copy the contents of **/home/odroid/ROMS** to your external storage device and then auto-mount the external device to **/home/odroid/ROMS** by adding it to **/etc/fstab**, or using **/etc/rc.local** to make it permanent.

What joysticks/gamepads are supported?

I built the image for use with an Xbox 360 wireless controller and Xbox 360 wireless USB receiver. So if you have that hardware, the image should work out of the box with no modification necessary, unless I forgot something again.

Besides that, every joystick/gamepad that is supported by Linux should work as well, but you will have to adapt settings for your device. Therefore you have to change the joystick settings on the individual emulators.

Running Mednafen, you can simply press ALT+SHIFT+1 to setup controlls for your device. The setup program is easy to understand. The second player, if supported by emulator can be setup

FRACTAL GENERATORS FOR LINUX

by Rob Roy

Last month, we featured some of the mathematical applications available for the ODROID platform through the Ubuntu Software Center. Since fractals have always been one of my favorite branches of mathematics, I tried out some of the fractal generator packages available through the Synaptic Package Manager and apt-get.

To get started, type the following in a Terminal window:

```
sudo apt-get install xaos \
mandelbulber xmountains xfractint
```

Xaos

Generates classic two-dimensional fractals, and has a nice "Load Random Example" button in the file menu. You can select different formula from the menu and tweak them to see what effect it has on the overall design.

Mandelbulber

Creates beautiful 3D fractals, just by pressing the "Render" button at the top of the screen. Change the fractal type by selecting "Fractal Formula Type" from the Fractal tab. Click on any area of the fractal image to zoom in and recalculate the view at a higher resolution. Make sure that your ODROID has a cooling fan, because this one takes a lot of processing power!

Xmountains

Draws realistic mountain ranges, with snow. There's nothing to do except enjoy the mathematical mountains as they scroll by.

Xfractint

Is a keyboard-based fractal generator that can produce all kinds of different fractals. For more information on how to set up Xfractint, visit `http://bit.ly/1s00x1x`.

with ALT+SHIFT+2, and so on.

For Retroarch, it's a bit more complicated. Quit XBMC, open a terminal, and type `retroarch-joyconfig`, then follow the instructions on the screen. At the end, you will get a long list of configuration parameters in the Terminal window. Copy this list, then open the file `/home/odroid/.config/retroarch/retroarch.cfg`, where you will find the same parameters listed. Replace the already existing parameters with the ones you got from `retroach-joyconfig`, and your device should work in retroarch.

XBMC unfortunately does not support a lot of devices for joystick support. Although Xbox 360 controllers are working fine, others do nothing at all.

With PPSSPP, you can change the controller configuration within the emulator by just going into the menu. However, on PPSSPP, the way controllers are implemented is rather sluggish so only a few really work well. In worst case, you can't even use the keyboard anymore. since the controller settings won't allow you to hit certain direction keys. If that happens, delete the file

```
/home/odroid/.config/ppsspp/PSP/
SYSTEM/controls.ini
```

and start over. If all else fails, remove any mappings for the controller and keep the settings for Keyboard only, which should always work. Then, use your best buddy, `antimicro`! If you use a different joystick device and really have trouble getting PPSSPP or XBMC to work with it, `antimicro` will work. Using antimicro, you can map keyboard commands to a button on your gamepad/joystick in the same way that you could simply map the keyboard arrow keys to your gamepads.

Does the image support CEC?

Not initially. The image has libcec installed which is working on HDMI 1, but I removed CEC from the XBMC image since it was causing issues. However, you can install XBMC with CEC support if you want from my repository at http://oph.mdrjr.net/meveric/.

When I exit an emulator, the XBMC window is really small, how can I fix that?

If you exit XBMC and restart it again, it will go back to fullscreen. I would advise you to "maximize" the XBMC window anyway to make it easier for you to select options.

Is there an environment besides XBMC on the image?

Yes, running behind XBMC is a full fledged Debian distribution with LXDE. This means that you can install everything you want on the image that is available from he Debian repository, and more. You can do everything that you can do on the Ubuntu images as well, such as web browsing, document editing, and graphics design.

How's the XBMC and 3D performance?

ODROID GameStation Turbo uses the latest Mali drivers (r4p0) that are provided by Hardkernel, together with the new armsoc framebuffer drivers and Xorg patches which give very good performance. The benchmark program es2gears runs with over 250fps, and glmark2-es2 runs with over 90fps. XBMC runs full speed with 60fps and supports 1080p playback of h.264 movies. It even allows vsync on movie playback and will change the frequency of your TV to match the movies frame rate. While doing so it uses very little CPU and RAM and outperforms the original Lubuntu 13.10 image that I previously published.

If you have further questions about GameStation Turbo, or any other inquiries about the games available for the ODROID, please visit the ODROID forums at http://forum.odroid.com. You can also find lots of pre-compiled games in my personal repository at http://oph.mdrjr.net/meveric.

PORTABLE SOLAR COMPUTING

POWER YOUR ODROID USING FREE ENERGY FROM THE SUN

by Rob Roy

I came up with an easy project for running a computer from only sunlight, which is an inexpensive and environmentally friendly way to generate electricity. The power requirementsfor even the most power-efficient laptop and desktop models of a standard PC range between 50-150 watts or more. However, the ODROID U3's 5-10W power footprint makes it an ideal candidate for solar computing. Here's how I set up a wireless Apache web server for my web development business that doesn't cost me anything to run, and can be charged anywhere that has sunlight, and run for around 6 hours without needing a charge.

I already owned the U3, and bought the solar parts on Amazon for around $240. The 3-panel model has enough solar cells to collect energy for the ODROID to run for about 6 hours per day with 10 hours of sunlight collection. If you only use the computer for an hour a day, you can purchase a smaller solar panel kit, which can bring the total cost down to around $100. You will also need a standard Android tablet USB charging cable (http://amzn.to/1mOYRr4).

I used several window suction cups to attach the GoalZero portable solar panel (http://amzn.to/1z0RUIV) to the charging input of an EasyAcc portable power bank (http://amzn.to/1sQ17mM) via a USB to MicroUSB cable. Whenever the sun is out, the charging indicator lights on the power bank flash at an intensity corresponding to the amount of light being received. The on/off button of the power bank also works as a convenient remote power switch.

To expand the system, daisy-chain several of the power banks to each other with USB to MicroUSB cables, connect the last power bank to the solar array, then plug the ODROID into the first power bank. It would take about 4 of these power banks to supply enough power for a 24/7 system. Or, you can upgrade to a 12V battery and use a converter to deliver the 5V 2A USB output.

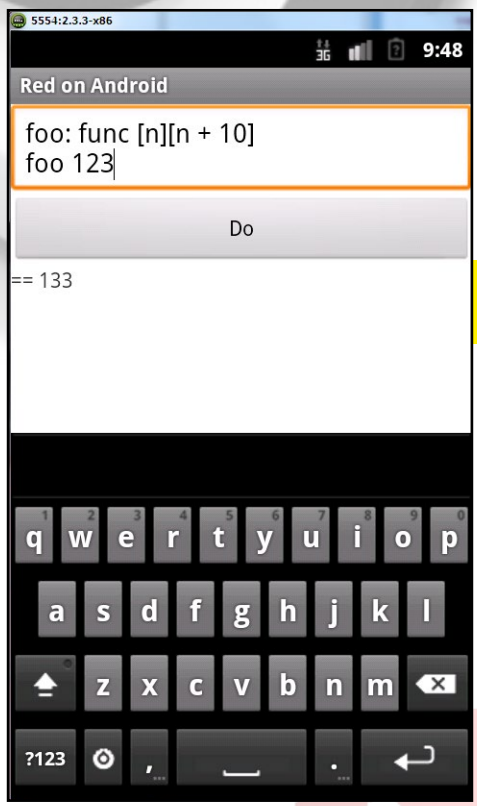
If you live in a sunny part of the world, it makes sense to use the free solar energy already available to you. Plus, in the event of a giant robot attack, you can take your ODROID camping and set up a web server in the wilderness!

Try to get a zero carbon footprint like this with an X86 computer and you are going to have to buy a huge battery!

BUILDING NATIVE ANDROID APPLICATIONS WITH RED

PART 2 - COMPILING AN ANDROID APP

By Gregory Pecheret



Red Command Line Console on Android

To build an Android application, you can download the compressed 480MB Android SDK from android.com, which requires more than 1GB of hard drive space. On the other hand, you will soon be able to download an SDK of about 1MB for the Red language instead. Even if the required disk space isn't important to you, the difference in the number of lines required to code an application is really significant, since it directly impacts the time required to develop, debug and maintain a project. All comparisons have limits, but if we simply consider source code size, the Apache server's source code is about 10MB while the Cheyenne source code (a commercial-grade Apache alternative written in Rebol) is less than 1MB.

If you're already convinced about the benefit of a high semantic level programming language like REBOL or Red, then you might wonder how to choose between REBOL and Red to develop an Android app. We already know that REBOL interprets the language while Red compiles it. In terms of user interface, REBOL has a dialect to build widgets from scratch, while Red will reuse existing native widgets.

The best solution to pick will depend on the project. Will the application use standard widgets only or will it mostly display custom graphics? Which one will be faster or more power saving? Building a user interface with Red will be very close to how it's done with REBOL's Visual Interface Dialect (VID). Because the dialect implementation in Red for Android is still a work in progress, we can only see (for now) a proof of concept in which widget instantiation is not wrapped.

To get started, download the zip archive from `https://github.com/red/red`.

Launch a REBOL console and run `build.r`, located at `bridges/android/eval/build.r`:

```
== %/home/odroid/red-master/
bridges/android/
>> do %build.r
```

```
Choose CPU target (ENTER = de-
fault):
1) ARM (default)
2) x86
3) both
=> 1

-=== Red Compiler 0.4.1 ===-

Compiling /home/odroid/red-mas-
ter/bridges/android/samples/eval/
eval.red ...
...compilation time:     637 ms

Compiling to native code...
.
.
Verification successful
...all done!
```

Once the steps have completed successfully, your very first Android application, written and compiled in Red, is located at:

home/odroid/red-master/bridges/android/builds/eval.apk

The APK can then be installed on any Android system and executed as a native Android application.

EASY ODROID-X/X2 DUAL BOOT SWITCH

BOOT FROM ANDROID OR LINUX WITH A FLICK OF A BUTTON

by Rob Roy and Venkat Bommakanti

The ODROID X and X2 devices allow for dual booting from SD card or eMMC module via a jumper on the motherboard. While the jumper is fairly accessible on the bare board, it may not be so easily reached when mounted in a case. This article describes a quick s cheme to create a specially purposed cable for setting the boot selection, even if the X/X2 is enclosed in a case.

Requirements

1. ODROID X/X2 board, with an appropriate power adapter.

2. MicroSD Card (with an SD-Card reader/writer), containing any Android or Linux image.

3. Motherboard jumper cable:
`http://www.amazon.com/gp/product/B009CWY8PA`

4. 3-way mini slide switch:
`http://www.amazon.com/gp/product/B009752DE0`

5. Heat-shrink tubing:
`http://www.amazon.com/gp/product/B005W42SW2`

6. Wire stripper (or use the small notch in a wire cutter blade)

7. Handheld flameless heat source such as a hair dryer

8. Soldering iron and solder (optional)

9. A protective case such as the one here (optional)
`http://bit.ly/1jIkp3Z`

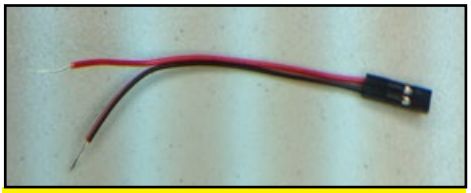
10. Superglue gel (optional)

Prepare the jumper cable

Take the motherboard jumper cable, as shown in the first photo, measure about 5-6" from the end with the jumper, and snip the cable at that point.

Motherboard jumper cable

Then, strip about ½" from the ends of the two wires, exposing the metal core.

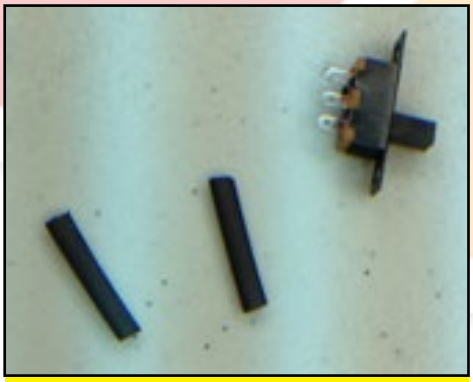


Stripped cable

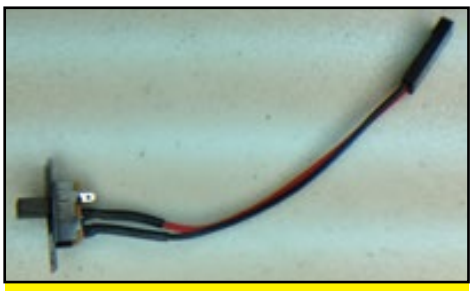
Create the custom cable

Select a small piece of heat shrink tubing of about 1" length that snugly slides onto the stripped wire. Insert one tubing into each wire of the jumper cable. Attach the 2 wires to 2 adjacent terminals on the sliding switch by looping them around the switch. You can carefully solder the connections if desired.

Then, move the tubing to completely cover the terminal and the exposed wire. Using a hair dryer on its highest setting, move the shrink wrap across the hot air and hold for 1-2 seconds. The rubber tubing will shrink and fit tightly around the connection.



Parts to attach to cable



Assembled special cable

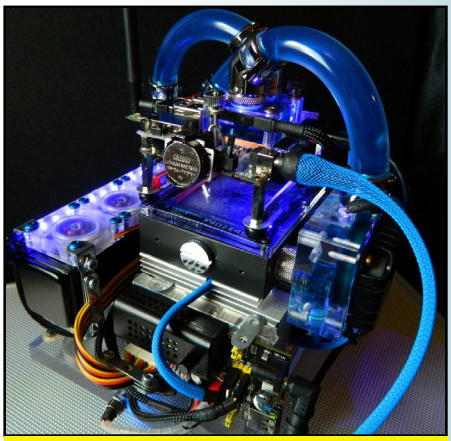
Attach new cable to X/X2

Attach the newly created jumper cable to the appropriate jumper on the de-

LIQUID COOLED ODROID-XU

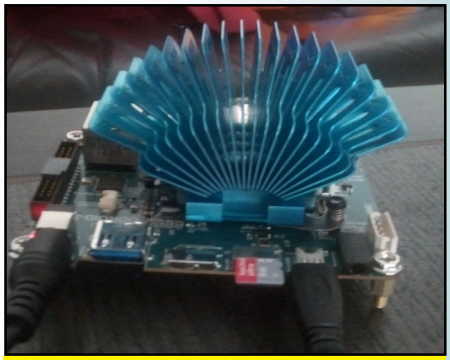
by Rob Roy

The ODROID-XU comes with a very quiet cooling fan, but several ODROID owners have taken cooling to a whole new level! Here are some pictures of the amazing systems that have been posted.



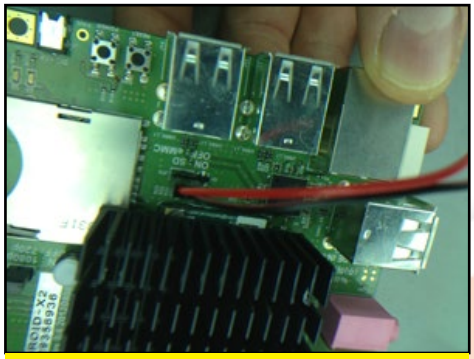
The H2ODROID, which uses water cooling and an extremely intricate heat sink

Another view of the H2ODROID

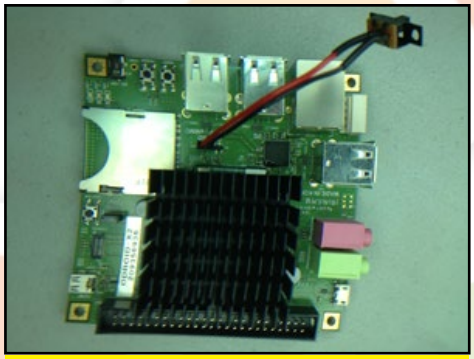


A passive cooling option using a Zalman ZM-NBF47 heat sink with an ODROID-XU

vice, as shown in figures 5 and 6. In this example, the SD/eMMC jumper is being toggled, but this could also be attached to the 720/1080p jumper as well.

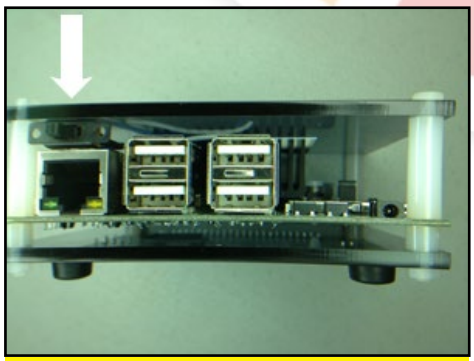


Jumper location on board



Jumper cable attached

The final two photos indicate a possible location for the newly added switch when a case is used. If you don't have a case, you can instead use a small drop of superglue gel to attach the switch to the top of the ethernet port, where it fits very neatly. You'll never worry about losing the cap to the jumper again! Moving the switch now chooses either the eMMC or SD card for booting.

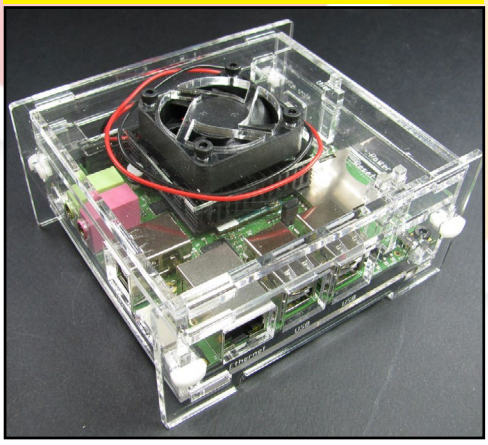


Side view of switch location on a custom case

Top view of switch location

To complete the project, you can mark an "U" (for Ubuntu) and "A" (for Android) on the case, based on the appropriate switch position, for easy OS selection. If you intend to use a different case design, the access to the switch may vary from what is shown here. For additional information or questions, please visit the original information sources at: `http://bit.ly/1iyfB69`.

Examples of custom cases for the U2 and X2



REMOTE DESKTOP FOR ODROID U3

USING VNC & XRDP PROTOCOLS

by Stangri, Edited by Venkat Bommakanti

For a variety of reasons, it may be necessary to access the desktop of a U3 remotely from another computer. A common method for access is to use the VNC (Virtual Network Computing) protocol to allow remote control of any computer's desktop. This article describes VNC-based access methods used to connect to a U3 from a variety of clients such as Windows 7+, OSX and Linux.

These steps apply to the x11vnc server on installations of Ubuntu 13.10 or 14.04, and does not apply to other VNC servers on the U3, such as TightVNCserver.

Requirements

1. An ODROID U3 board, with an appropriate power adapter.

2. Any MicroSD card or eMMC module with a bootable Ubuntu, Debian or similar image that includes an X Windows (X11) desktop.

3. A network where the device has access to the Internet and the ODROID forums.

4. ssh access to the U3 via utilities like PuTTY (MS Windows 7+) or terminal (Mac, Linux) from the remote desktop.

5. TightVNC viewer 2.7.1 for Windows 7+

6. Remmina 0.0.99.1 for Ubuntu or Debian

7. Microsoft Remote Desktop Connection for Windows 7+

Install x11vnc

First access the U3 via ssh, using any SSH tool compatible with your client machine. From a Terminal session in Linux, the command would be:

```
$ ssh root@odroid
```

On a typical Hardkernel image, the hostname of the U3 wil be odroid. Connecting through SSH via PuTTY will present a second command prompt from the remote machine. Using this prompt, install x11vnc using the commands:

```
$ sudo aptitude update
```

```
$ sudo aptitude install x11vnc
```

It is very likely that x11vnc is not already packaged into the image, by default. Validate the installation by checking the version like so:

```
$ x11vnc --version
x11vnc: 0.9.13 lastmod: 2011-08-10

Set x11vnc password
```

To secure the VNC access to the U3, set up a password using the following command:

```
$ sudo x11vnc -storepasswd {your_password_here} /etc/
x11vnc.pass
```

Use a strong password in this command, excluding the curly brackets { }. Then, update the access privilege for this newly setup password file, using the command:

```
$ sudo chmod 744 /etc/x11vnc.pass
```

Without this privilege change, the behavior of a VNC session is unpredictable. One may see issues such as a blank/gray screen in the VNC viewer, VNC server crashing with an incoming connection, or other issues.

Configure x11vnc

Prior to setting up the configuration, it's helpful to determine the display resolutions supported by the U3. Check the available resolutions using the command:

```
$ xrandr
Screen 0: minimum 320 x 200, current 1920 x 1080,
maximum 4096 x 4096
HDMI-1 connected 1920x1080+0+0 1440mm x 810mm
   1920x1080      60.0*+   30.0
   1920x1080i     60.1
   1280x1024      60.0
```

```
1280x960        60.0
1152x864        75.0
1280x720        60.0
1024x768        60.0
1440x480i       60.1
720x480         59.9
640x480         60.0
```

You may want to select a different resolution depending to the monitor that is attached to the remote desktop. The 1920x1080 resolution has been chosen for this article.

The x11vnc installation can now be configured using a config file, located at:

```
/etc/init/x11vnc.conf
```

Create this file if it's not already there, using an editor such as vi, medit, or nano. For typical VNC access, the content of the config file needs to be similar to this:

```
start on login-session-start
stop on runlevel [016]

emits vnc-server-start

script

/usr/bin/x11vnc -forever -bg -geometry 1920x1080
-usepw -auth /var/run/lightdm/root/:0 -display :0
-rfbauth /etc/x11vnc.pass -shared -noprimary -o /var/
log/x11vnc.log

initctl emit vnc-server-start

end script
```

Use the editor (with root privileges) to update the config file. Note that the x11vnc configuration line (starting with /usr/bin/x11vnc) needs to be a single line.

If your remote desktop happens to run OSX or Linux (for example, Ubuntu 14.04 LTS), you can access it by enabling the avahi services on the U3. It allows for easy discovery by avahi-aware desktops of the VNC service offered by the U3 over the local network. The x11vnc.conf file in this case would look like this:

```
start on login-session-start
stop on runlevel [016]

emits vnc-server-start

script

/usr/bin/x11vnc -forever -bg -geometry 1920x1080
-usepw -auth /var/run/lightdm/root/:0 -display :0
-rfbauth /etc/x11vnc.pass -shared -gui tray -nopri-
mary -noxdamage -zeroconf -avahi -env X11VNC_AVAHI_
NAME=odroid -o /var/log/x11vnc.log

initctl emit vnc-server-start

end script
```

The X11VNC_AVAHI_NAME option should reflect the hostname of your U3, in case it is modified from the default value of odroid. Any Screen Sharing application such as the built-in Screen Sharing application in OSX or the Remmina VNC client application for Ubuntu 14.04 can be used to view the U3's desktop.

Some users may find boot up issues with the gui config option. If so, you can remove the following option from your config file:

```
-gui tray
```

Also, ensure there no other VNC server instances using Display #0.

Run x11vnc

To start the x11vnc server on boot, type the following command on a single line in a Terminal window, then reboot the U3:

```
$ sudo /usr/bin/x11vnc -geometry 1920x1080 -usepw
-auth /var/run/lightdm/root/:0 -display :0 -forever
-bg -rfbauth /etc/x11vnc.pass -o /var/log/x11vnc.log
```

After rebooting, validate that the x11vnc server is up and running by examining the x11vnc log file at:

```
/var/log/x11vnc.log
```

It should reflect the vnc display information like so:

```
18/06/2014 14:11:27 Avahi group odroid established.
The VNC desktop is:      odroid:0
```

Note that in this case, one will have to attach to display #0 on odroid (i.e., display:0).

Install XRDP Service

If you wish to access the U3 desktop from a Windows 7+

remote system, using the Microsoft Remote Desktop Connection application, you will need to also install the xrdp service on the U3. Install it using the following command:

```
$ sudo apt-get install xrdp --no-install-recommends
```

Update the newly created xrdp config file to now reflect the following options:

```
/etc/xrdp/xrdp.ini

[globals]
bitmap_cache=yes
bitmap_compression=yes
port=3389
crypt_level=high
channel_code=1
max_bpp=24

[xrdp1]
name=local-xvnc-console
lib=libvnc.so
username=odroid
password=ask
ip=127.0.0.1
port=5900
```

The username parameter needs to reflects the default odroid user created in a clean install. If you intend to use another username, ensure the new username has the same group associations as the odroid account. The following command provides that group info:

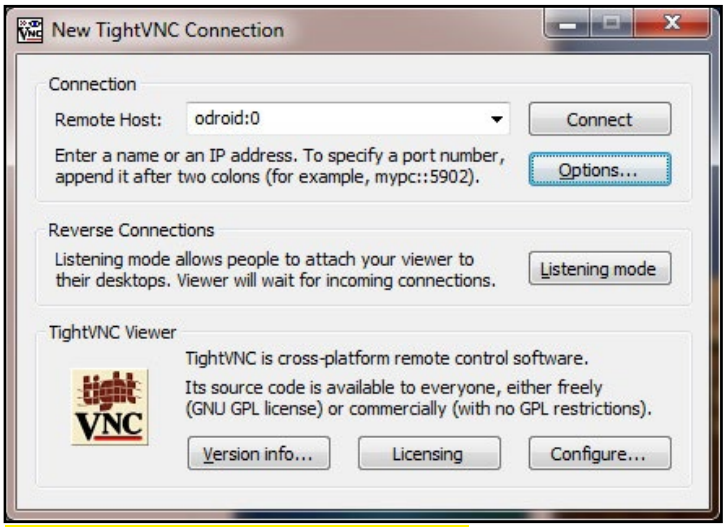
```
$ groups odroid
odroid : odroid adm dialout fax cdrom floppy tape sudo
audio dip video plugdev netdev nopasswdlogin lpadmin
scanner fuse
```

To enhance security, it is useful to force the display of client-side login dialog box. The password option value of ask enables this option. The real password will not have to used in this config file, increasing security and allowing for one to change the password at will, without altering this config file at all.

The xrdp service is a server-side add-on service, that requires a fully functional VNC server service on the U3.

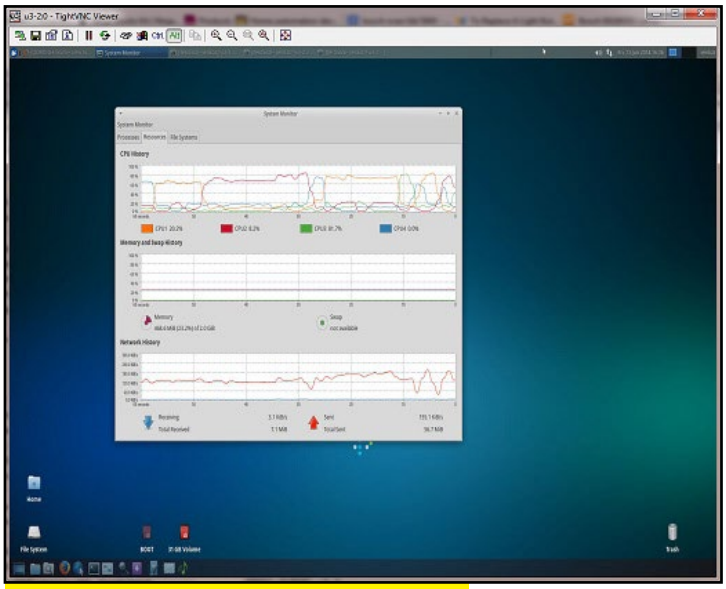
TightVNC Viewer on Windows

Launch TightVNC Viewer and enter the U3's VNC server display information as shown in the TightVNC Viewer Login image.



TightVNC viewer login

Then click the connect button to attach to the display and view the U3's desktop remotely. The U3's desktop will appear similar to what is illustrated in the U3 Desktop via TightVNC Server image.

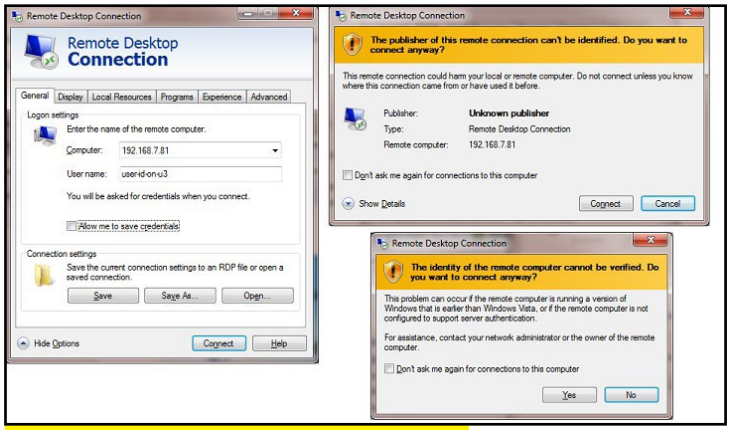


U3 desktop via TightVNC viewer

Remote Desktop Connection

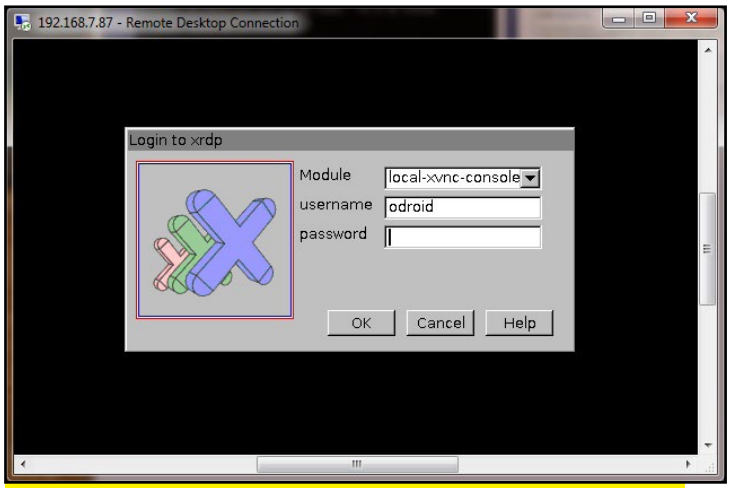
Launch the Remote Desktop Connection application. Enter the U3's VNC server information and fill out the form as shown in the Remote Desktop Connection login screen image. In the preliminary login dialog, click on Show Options to select more options. Enter all the details in the General tab, select the Display tab and select the largest (full) screen option. Select the 24bit true color option, if applicable.

On successful connection, the xrdp login screen of figure 4 is displayed. Enter the needed vnc server password and click OK to proceed. Please note that the module name



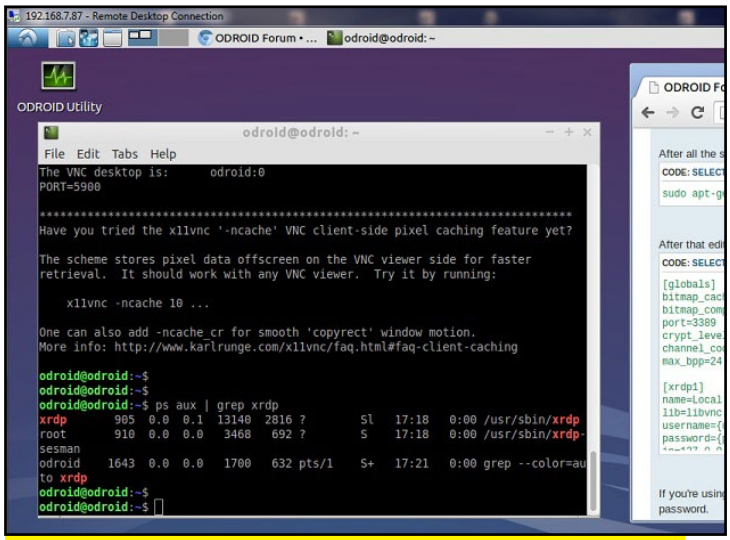
Remote Desktop Connection login screen

local-xvnc-console used in the login dialog should match the configuration entry found in xrdp.ini, as mentioned earlier.



xrdp driven login screen for Remote Desktop Connection

A successful login results in the display of the U3 desktop, similar to the one shown in the U3 Desktop via Remote Desktop Connection image.



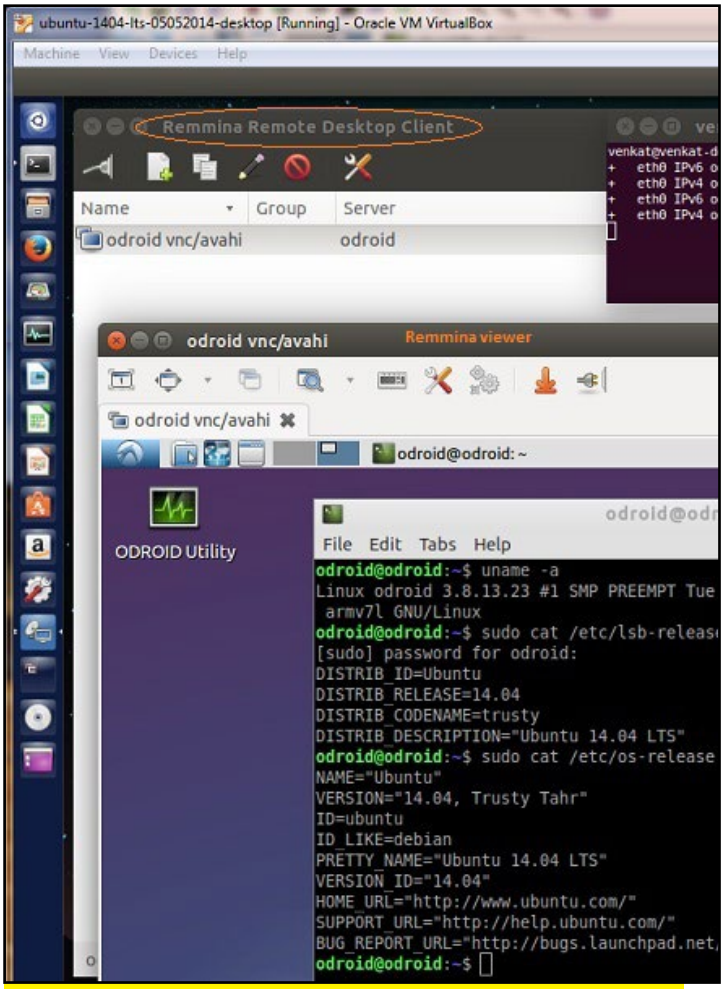
U3's desktop via Remote Desktop Connection

Remmina for Ubuntu 14.04 LTS

Install Remmina from the Ubuntu Software Center. Launch the Remmina Remote Desktop Client. Select the entry for the U3 and connect to it.

You can also scan for the available avahi-aware services by running the command:

```
$ avahi-browse -all
```



U3's desktop via Remmina

For additional information or questions regarding connecting to your ODROID via VNC and XRDP, please visit the original sources at:

http://bit.ly/1qS6AWc

http://www.tightvnc.com

http://avahi.org

http://remmina.sourceforge.net

http://bit.ly/1z93Bxd

http://bit.ly/1mVyfF2

ODROID POWERED OFF-ROAD UNMANNED GROUND VEHICLE!

PART 2: INTERFACING WITH MOTORS AND SENSORS

by Christopher D. McMurrough

This robot runs on an ODROID-XU, is made from readily available parts, and loves rough terrain.

In this article, we continue our series about building an Off-Road Unmanned Ground Vehicle (UGV) using the ODROID-XU hardware. Part 2 gives an overview of the communication between the ODROID and the platform motor controllers and sensors, which we will later use in Part 3 to autonomously navigate to GPS way-points. The controller application runs on the Ubuntu 12.04 Robotics Edition image for the ODROID-XU, which can be found for download from the Hardkernel forums.

Introduction

In Part 1, we focused mostly on the mechanical and electrical aspects of our robot. Now that we have a stable platform to work with, we will look at setting up the ODROID-XU to communicate with our motor controllers and sensors. The motor controllers we have chosen to use for this project are the Parallax HB-25, which will be controlled by a Teensy USB microcontroller [1]. The Teensy accepts serial command packets from the ODROID-XU, which will then be used to generate the necessary servo control pulses to set the motor speeds and direction.

Additionally, we will communicate with an Android device (in this case, a Nexus 7 tablet) to acquire GPS and heading data. This will allow us to control the motor speeds in order to meet our navigation goals in Part 3. The Android device communicates via USB tethering, which will allow it to talk with the ODROID as a network peer.

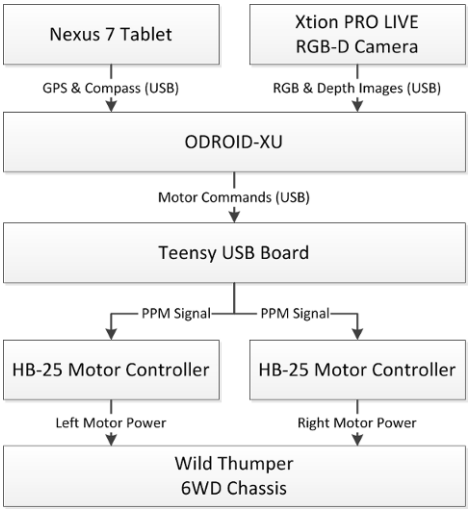
Finally, we will demonstrate the use of Robotic Operating System (ROS) to send and receive data. This modular approach will make it easier for us to extend the capabilities of our platform in the future. Example code can downloaded from http://bit.ly/1jfykOU.

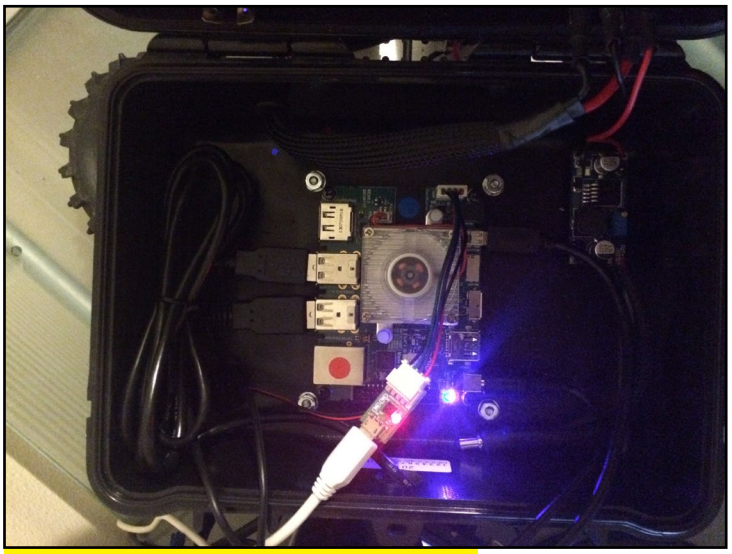
Operating System Setup

We will first set up our ODROID-XU to run Ubuntu 12.04 Robotics Edition [2]. This OS image has been preloaded with Robot Operating System (ROS) and other robotics software such as OpenCV, Point Cloud Library, and OpenNI. No extensive setup is needed, just acquire the image from the ODROID forum and copy to your SD card or EMMC module. While this series specifically uses the ODROID-XU, robotics images are available for the X2 and U2/U3 as well.

Once the OS image has been loaded onto your ODROID, boot up the device with a keyboard, mouse, and monitor attached. You will be taken directly to the LXDE desktop environment. From there, open a command window and type "roscore". Hit enter, and voila, you are now running Robot Operating System! We will be using it later in this article.

UGV System Architecture





This UGV is smart enough to have an ODROID-XU as its brains, which uses only a few watts

Motor Controller Interfacing

The Teensy microcontroller is used to interface the ODROID to the HB-25 motor controllers. The HB-25 expects servo PPM signals, which any Arduino device (such as our Teensy) can easily generate. We will connect the signal pins of the HB-25 to pins 20 and 21 on the Teensy, and set them up as servo control pins. Using the Arduino Servo library, the motors are controlled by sending an integer to the "write" command.

The value sent to this command specifies a target servo angle between 0-180 degrees, with 90 degrees being neutral. In the case of the HB-25, a value of 0 will apply no power to the motor, while a value of 0 or 180 will apply full power to the motor in either direction. As the values deviate from 90 toward 0 or 180, the motor will be supplied with more power and will begin to spin faster.

By writing an integer to either output pin using the Servo library, we can independently control the speed and direction of the motors. The next task will be to establish communication with the ODROID using serial data packets. The ODROID will transmit a single packet of data to encapsulate our two control bytes. These packets will be sent continuously at a rate of 20 Hz, so that the ODROID can smoothly control the motion of the platform. When loaded, the Teensy firmware will allow the device to be enumerated as a virtual serial port. We will then write our control packets to this port using our demo ROS node.

Command Packet Structure

Serial ports facilitate the transmission of serial data, which are just chains of bytes. For very simple applications, such as continuously sending a single byte value, an application can simply read or write only the most recent byte and ignore any alignment or data validation. Most robotics applications require transmission of several bytes of data (e.g., writing commands to multiple servos), so care must be taken to ensure that the receiver knows the meaning of each byte.

In our example, we will define a packet with a start byte, length byte, payload, and checksum. The start byte signals to the receiver that a packet is being transmitted. The length specifies the total length of the packet, including the framing data, in number of bytes (to support packets of variable size). The payload consists of 2 bytes for the motor commands, and the checksum provides a validation mechanism for the Teensy. The resulting packet structure then looks like this:

```
Packet[0]
Packet[1]
Packet[2]
Packet[3]
Packet[4]
Start byte
Packet length
Payload byte 1
Payload byte 2
Checksum
```

The start byte is arbitrarily defined as the byte 0xAA. The packet length, for this packet, is defined as 0x05 (3 framing bytes + 2 payload bytes). The checksum is computed individually for each packet based on all of the other bytes. We define the checksum to be the value obtained by "exclusive OR'ing" all of the other bytes together, so that:

```
Packet[4] = Packet[0] xOR
Packet[1] xOR Packet[2] xOR
Packet[3]
```

When the Teensy collects all of the packet bytes, it computes what the checksum should be and compares it to Packet[4]. If these two values are not equal, then either data has been corrupted or there is a conflict with the received byte count. Either way, this feature allows the Teensy to gracefully detect and recover from an error (or stop the motors if necessary). While we only need to transmit 2 payload bytes to the Teensy for this particular project, the approach can be used to send any number of bytes by adjusting the packet length appropriately. In your own application, you will want to adjust the packet length to match the length of the data that you're going to the Unmanned Ground Vehicle in order to control the motors and sensors that you're using.

Microcontroller Firmware

The Arduino microcontroller firmware is included in the "ugv" folder of the example code mentioned earlier. To use the firmware, simply compile and flash the file "teensy_firmware.ino" using the Arduino IDE by following the instructions found on the Teensy website [1]. Once the firmware has been installed, you will be ready to execute the demo. Note that the appropriate udev rules must be installed on the ODROID in order for the application to be able to open the serial connection to the Teensy without using the "sudo" command.

Motor Controller ROS Interface

Included in our example code is an ROS node that will connect to the specified serial port, and assemble properly formatted packets as specified. This node subscribes to ROS "twist" messages, which are the standard method of specifying the desired velocity of a robot. In our case, we only care about the forward linear velocity and rotation. The rotation allows us to compute the relative amount of power to supply to the left/right motors, which the linear velocity sets the overall speed. Part 3 will go into more details about calculating appropriate motor speeds, but for now, we just want to run the ROS node and verify that the USB serial port opens properly. It may be necessary to change the /dev/ttyUSB port name to match your configuration.

Navigation Data

Our system uses navigation data from an Android tablet for motion planning. To do this, we must first establish a connection between the tablet and ODROID, and link this data to ROS. Luckily, this is fairly easy to do using the "ROS Android Sensors Driver" Android app [4]. Once this app is installed on the Android device, the GPS, accelerometer, and compass data are then published to a ROS master at the specified IP address. By connecting the tablet to the ODROID via USB and enabling tethering, the Android and ODROID will then be placed on the same network and can communicate via IP. It is important to note that the android_sensors_driver ROS package must also be installed in order to receive the app's broadcast data.

Once the tablet is tethered, run the command line program roscore on the ROS host, then start the android_sensors_driver programs on the Android tablet. Enter the IP address and port of roscore into the Android app, and hit connect. The published GPS and inertial data are shown as ROS topics, which we can use for our navigation tasks.

Conclusion

We now have all of the key software components to access our motors and sensors, and have linked everything into ROS, which will make it easier for us to develop our application software. In Part 3, we will input target GPS coordinates into ROS, compute the necessary motion trajectory, and issue the appropriate motor control commands to reach the goal. We will then write additional high-level ROS nodes for command and control, and finally, test the system in an outdoor setting.

References

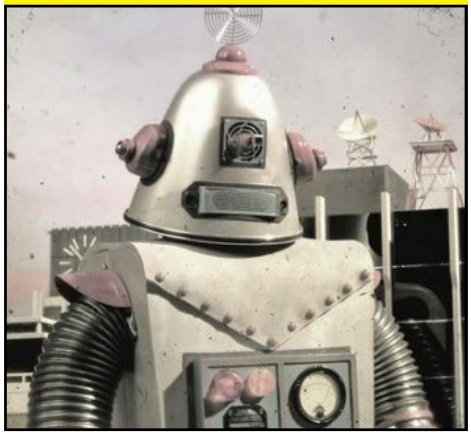
[1] P. Stroffregen, "Teensy USB Development Board," PJRC Electronics, 2014. `http://www.pjrc.com/teensy/`

[2] C. McMurrough, "Ubuntu 12.04 Robotics Edition v1 (ROS+OpenCV+OpenNI+PCL) XU," ODROID Forum, 2014. `http://forum.odroid.com/viewtopic.php?f=61&t=5216`

[3] C. McMurrough, "Odroid Code Examples and Support Files," GitHub, 2013. `https://github.com/cmcmurrough/odroid-development`

[4] C. Rockey, "Ros Android Sensors Driver," `http://wiki.ros.org/android_sensors_driver/`

We followed the instructions, but our UGV looks much different than the one Chris built



CHANGE YOUR HEARTBEAT

CONTROLLING THE ALIVE LED ON THE U3

by Mauro Ribeiro

Changing the behavior of the external blue alive LED on the ODROID-U3 is easily done with a single Linux BASH script command. Type any of the following in the Linux Terminal window:

1. Completely disable the alive led.

```
$ echo none > \
/sys/class/leds/led1/trigger
```

2. Stay solid (non-blinking):

```
$ echo default-on > \
/sys/class/leds/led1/trigger
```

3. Show the eMMC activity:

```
$ echo mmc1 > \
/sys/class/leds/led1/trigger
```

4. Show the SD activity:

```
$ echo mmc0 > \
/sys/class/leds/led1/trigger
```

5. Make it blink a custom pattern:

```
$ echo timer > \
/sys/class/leds/led1/trigger
$ echo 1000 > \
/sys/class/leds/led1/delay_on
$ echo 1000 > \
/sys/class/leds/led1/delay_off
```

The last example lets you specify how many milliseconds the light should stay in each state. You can also use the heartbeat LED to signal other activity. For example, you can track SSH activity by monitoring the IP tables by typing the following in a Linux terminal window as root:

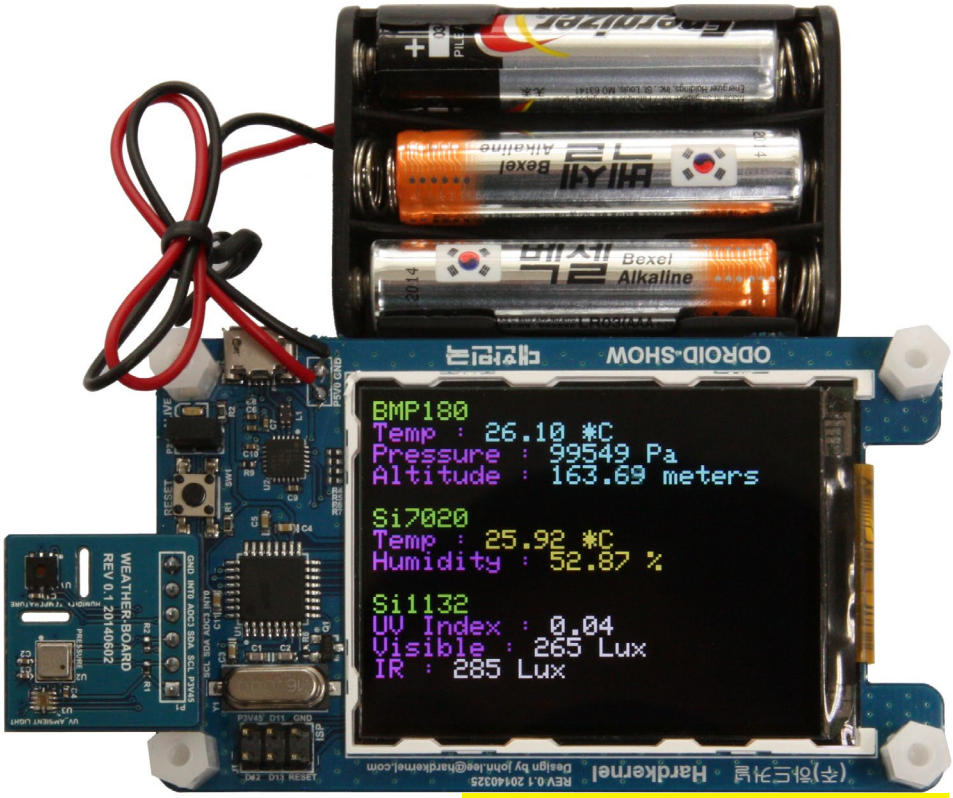
```
$ iptables -A INPUT -p tcp \
--dport 22 -j LED \
--led-trigger-id ssh \
--led-delay 1000
$ echo netfilter-ssh > \
/sys/class/leds/led1/trigger
```

To make the change permanent, add the line to the file/etc/rc.local, and the new blue LED pattern will take effect after the next reboot.

ODROID WEATHER BOARD

INTELLIGENT WEATHER MONITORING ON YOUR ODROID

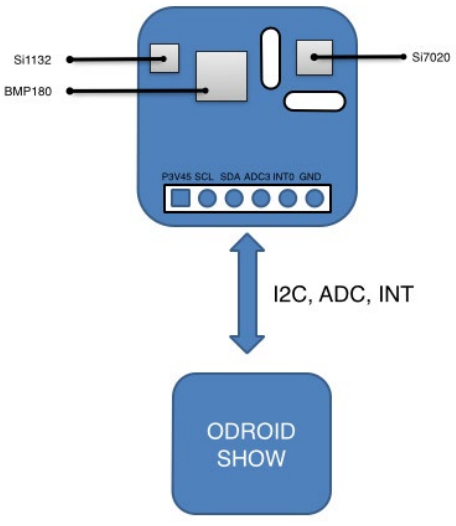
by Justin Lee



The Weather Board is your personal tool for determining whether it's safe to go outside, or just stay inside and work on an ODROID project.

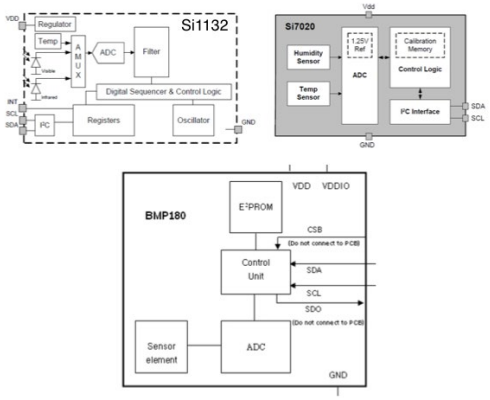
The Weather Board is an easy-to-use ODROID-SHOW addon that gives you access to UV Index, barometric pressure, altitude, relative humidity, illumination and temperature measurements. Its dimensions are only 20 x 20mm, and comes with a 6-pin connector for the ODROID-SHOW.

The board reports the Si7020 humidity, BMP180 barometric pressure, and Si1132 Ultraviolet (UV) Index with ambient light sensors, and relies on some sensor-oriented Arduino libraries. All the sensor data goes though I2C communication between sensor ICs and ATMEGA328P MCU on the ODROID-SHOW.



Weatherboard block diagram

With a few batteries, you can make a portable outdoor weather data capture system. You can also connect it to your host PC or ODROID via the ODROID-SHOW in order to log various environmental data, like a weather station does.



The Internal architecture of each sensor

Hardware setup

Connect your Weather Board to the ODROID-SHOW, then connect a micro-USB cable to your host PC or ODROID Linux board. If the connection is established, you will see a new device appear called /dev/ttyUSB0 (or something similar). You can also connect 3-4 Alkaline or NiMH/NiCD batteries if you want to build a portable device.

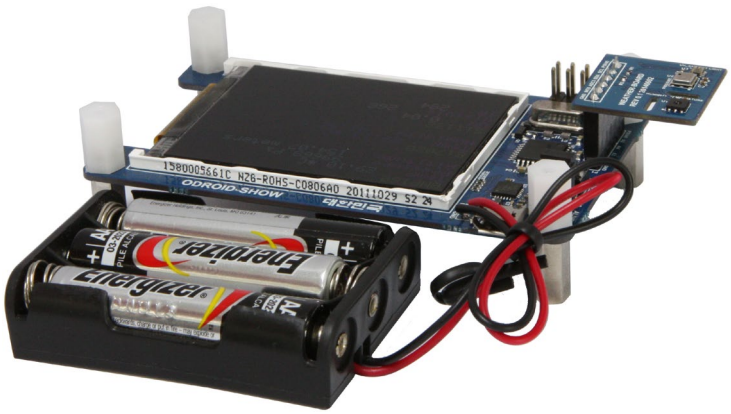
Finally, launch a copy of Arduino IDE, which is available for installation from Synaptic Package Manager or Ubuntu Software Center.

Firmware setup

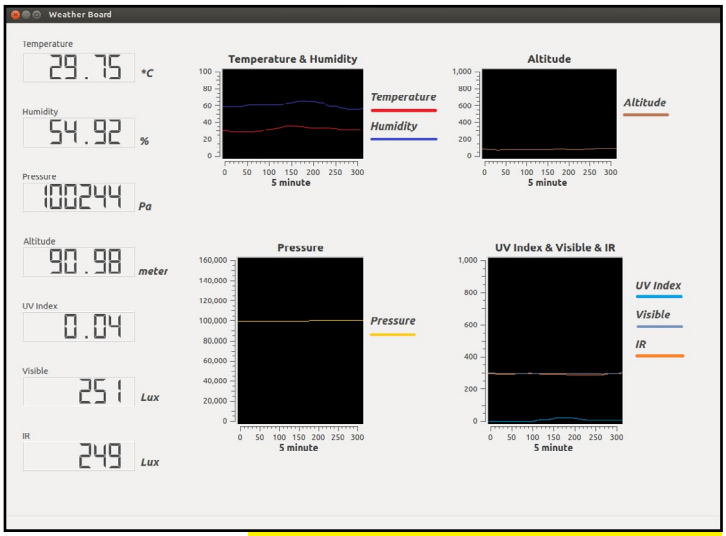
Download the sketch file and libraries from `http://bit.ly/1iuz4VK`, then open the weather_board.ino sketch file using Arduino, which is located at

```
ODROID-SHOW/firmware/
weather_board/weather_board.ino
```

Select "Preferences" in the Arduino IDE File menu and choose ODROID-



Another view of the Weather Board on a U3.



A QT-based GUI application program can monitor and show the weather data graphically.

SHOW as the sketch folder. Add the libraries by selecting Import library->Add library and choosing ODROID-SHOW/libraries. Before building and uploading the sketch file, the jumper cap, which is near the reset button, must be installed.

A host PC or ODROID can log the data as well as display it graphically in real-time. For our example, we used the QT programming language.

The ODROID Weather Board may be purchased at `http://bit.ly/1wtPdgP`.

Build the Qt Application for Windows

1. Download and install Qt 4.8.4(MinGW) version at `http://bit.ly/1nAxheV`

2. Add `C:\Qt\4.8.4\bin` to our systems path variable (qmake.exe is located here)

3. Add `C:\MinGW\bin` to your systems path variable (mingw32-make.exe. is located here)

4. Download and extract qwt-6-.1.0 at `http://bit.ly/1quAoaY`.

5. Copy the qwt-6.1.0 to `C:\qwt-6.1.0`

6. Open a command line (cmd) and navigate to "`C:\qwt-6.1.0`":

```
# cd C:\qwt-6.1.0
# qmake
# make
# make install
# qmake -set QMAKEFEATURES C:\qwt-6.1.0\features
```

7. Add `C:\qwt-6.1.0\lib` to your systems path variable (`qwtd.dll` is located here)

8. Download the PC Application source code at `http://bit.ly/1pgo0yf`.

9. Build the weather_board application using qmake:

```
# cd qt_weather
# qmake
# make –f MakeFile.Release
# qmake
# make –f MakeFile.Release
```

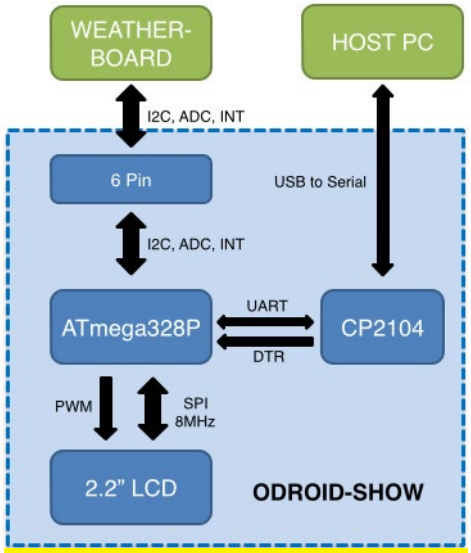
Build the Qt Application for Ubuntu (ODROID or x86)

1. Install packages

```
# sudo apt-get install qt4-default qt4-designer libqwt-dev
```

2. Build the weather_board app

```
# git clone https://github.com/hardkernel/ODROID-SHOW/qt_weather
# cd qt_weather
# qmake
# make
# uic weather_board.ui > ui_weather_board.h
# qmake
# make
# cd linux
# ./weather_board
```



Weather Board block diagram using ODROID-SHOW

Be a hero and save your village with a Weather Board and U3 portable station.

MEET AN ODROIDIAN

DENIS ODINTSOV (@OVERSUN) BLACK BELT PROGRAMMER AND XBMC EXPERT

by Rob Roy

Please tell us a little about yourself.

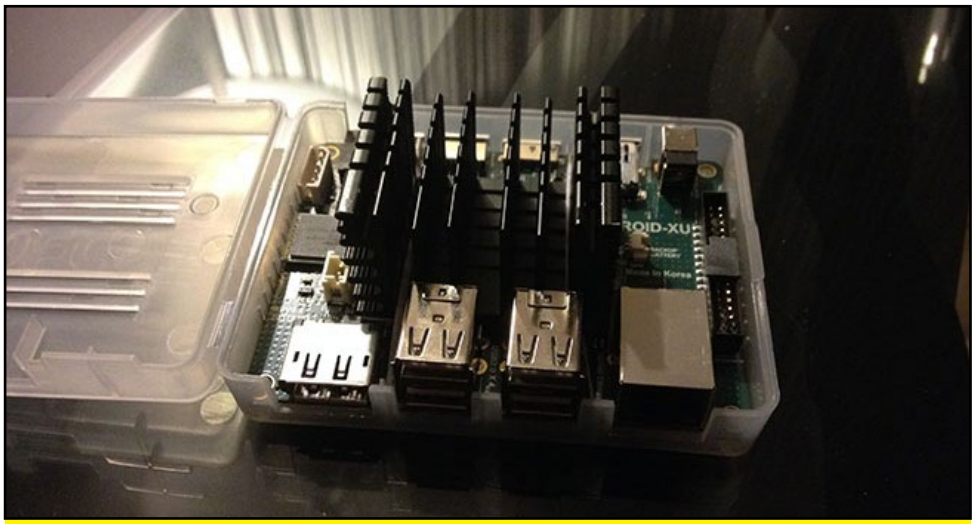
My name is Denis Odintsov. I'm from Russia and living in Poland, working for a US company called Cisco Systems. As for my interests, I love computer games, books, quiet places, clean air and good food. Although it sounds a bit autistic, these are the things that interest me at the age of 34.

How did you get started with computers?

To draw a little picture of how long I've been with computers, I can say that I remember an article that talked about the invention of a technology that should change the world called the "world wide web". Yes, I read it in a paper magazine! I didn't have a computer at that time, that was too long ago, but I was interested for sure. I received my first computer a short time afterwards which was a cutting edge 25Mhz Intel 286 from the early nineties. I can't remember the exact date right now, that's how young I was and what a long time ago.

But enough with dusty stories, the more interesting thing would be: how I started with ODROIDs. Basically I can tell that I already saw this kind of story before, since I had a Raspberry pi, and I loved it much. It was as a small media center that didn't buzz, blinks or occupy an entire drawer. That was actually so amazing, because I was into the PC world before 2012, and couldn't even imagine such a thing back then. The only problem for me was the speed of the menus. The Pi was sluggish.

So, as a long time x86 user I just went onto the great Internet and bought



Although Denis ended up with a XU case that no longer is able to close, in the purest ODROIDian spirit, he managed to custom his own board using a fanless heatsink for total silent operation!

"something faster" without any knowledge about the ARM world and all the differences. Really, I just bought something as small, but with more Mhz. And, at that point, I must say ODROID was the only and natural choice. I think that the beauty of the U2 is unbeatable still, as inside as well for the outside design. It's the most perfect piece of hardware that I have in my possession.

And then came the real truth, which is actually the most amazing part of the story: I just found out that I cannot take something that was compiled to run on the Raspberry Pi and install it on another ARM board (don't laugh!). That was like a cold shower for a guy who only knew the x86 world. Luckily, there was already a Linux distro running, and that was my turf. If something can run Linux, I can deal with it.

You've been one of the main community developers for XBMC for over a year. What type of experiences have you had while porting the popular media center to the ODROID?

All the people who have been with ODROID long enough can remember the Linux situation at that time: XBMC 11 was a stable version but was obsolete, and XBMC 12 was showing only a black screen. But the beauty of Linux is that everything has sources. The hardware supported 3D acceleration, and there were a few GL demos for that. What I did was just connect the dots. I streamlined the display initialization used in the GL demos to XBMC 12, and it started right up and ran smooth! It took probably a week of my time after work to port 500 lines of code from one program to other program. Not a challenging job at all.

And then the second part of my XBMC experience began: the whole new world of hardware that doesn't even exist in the x86 architecture. It really took days for me to understand that, in the ARM world, it's a whole different

story to be able to decode a video successfully. And so the challenge began.

Looking back right now, I can see the funny thing is that probably, the code that would eventually work was ready in a couple of weeks. However, there was an annoying Mali bug on the 3.8 kernel that was making the GPU redraw the whole screen at 6-10 frames per second (FPS). And nobody really knew what was the problem. I spent literally days understanding how things worked, inside and out, to understand why the video output was so slow, thinking it was somewhere in the hardware decoding. This activity probably contributed to my programmer knowledge building more than at any other time during the previous few years.

I'm not a real programmer, actually, since my area of knowledge is mostly troubleshooting and systems design. Programming is a hobby for me. And here is the part I want to underline most: I don't think that most my biggest contribution was the code that allowed XBMC to run on ODROIDs successfully. I honestly think that was the effect that my work had on the ODROID community.

I remember how small and enthusiastic the forum was in the beginning of 2013, and I can see how big and powerful it has become now. And I'm proud to think that, my humble efforts to run XBMC may have encouraged people to buy more hardware pieces, and also attracted people who really know how to program. I can see that the code I wrote, taken by people for whom programming is more than a hobby, evolved into amazing things from the plain and simple source. And from that point they already went far beyond, where I'm not even a mere help for those programmers anymore. And that is the real achievement for me here, and that thing I'm most proud of.

How did you become an expert in Linux development?

When can it be easier to become an expert in development than when you have the internet at your hands? Linux

Our man Denis enjoying a quiet moment at the Beach.

is open, it is amazingly documented, and gives you everything to control by your fingertips. Just choose a task, open a text editor and go ahead! At the very first difficulty that you have, even if it is the very first line of your program, just google it, and you will see hundreds of pages of answers. Create small programs that can't do anything except crash badly, create small programs that do silly things, and at some point you find yourself capable of programming anything that you want for yourself. And shortly after, you would find that some things that you create are wanted by thousands people around the world as well. It is really that simple.

What other hobbies and interests do you have?

Fitness, books, creating things… being happy is probably the most common description of the only hobby I have. Sometimes it is because of places where I am, sometimes it is because of people that I meet, and sometimes it is because of things I do and the effect that it has.

Are you involved with any other software or hardware projects?

I like programming for Apple devices lately, it gives me some fun. iOS has its blacks and whites and doesn't give you as much vast freedom as Linux can give, but it is convenient and can be profitable.

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

The last time that I went to the ODROID site, I found some absolutely amazing things - like the UPS board! Way to go, Hardkernel, this is really awesome! So, the only thing that I probably would be thinking of now is a GPS/GSM module, because I use my U2 as a GSM gate. As for the chips themselves, anything supported well in Linux would do. Hardkernel is making great boards - small, solid and beautiful.

An ODROID's job never ends! Using the U2 as a GSM gate is a THE way to do a personal cloud.