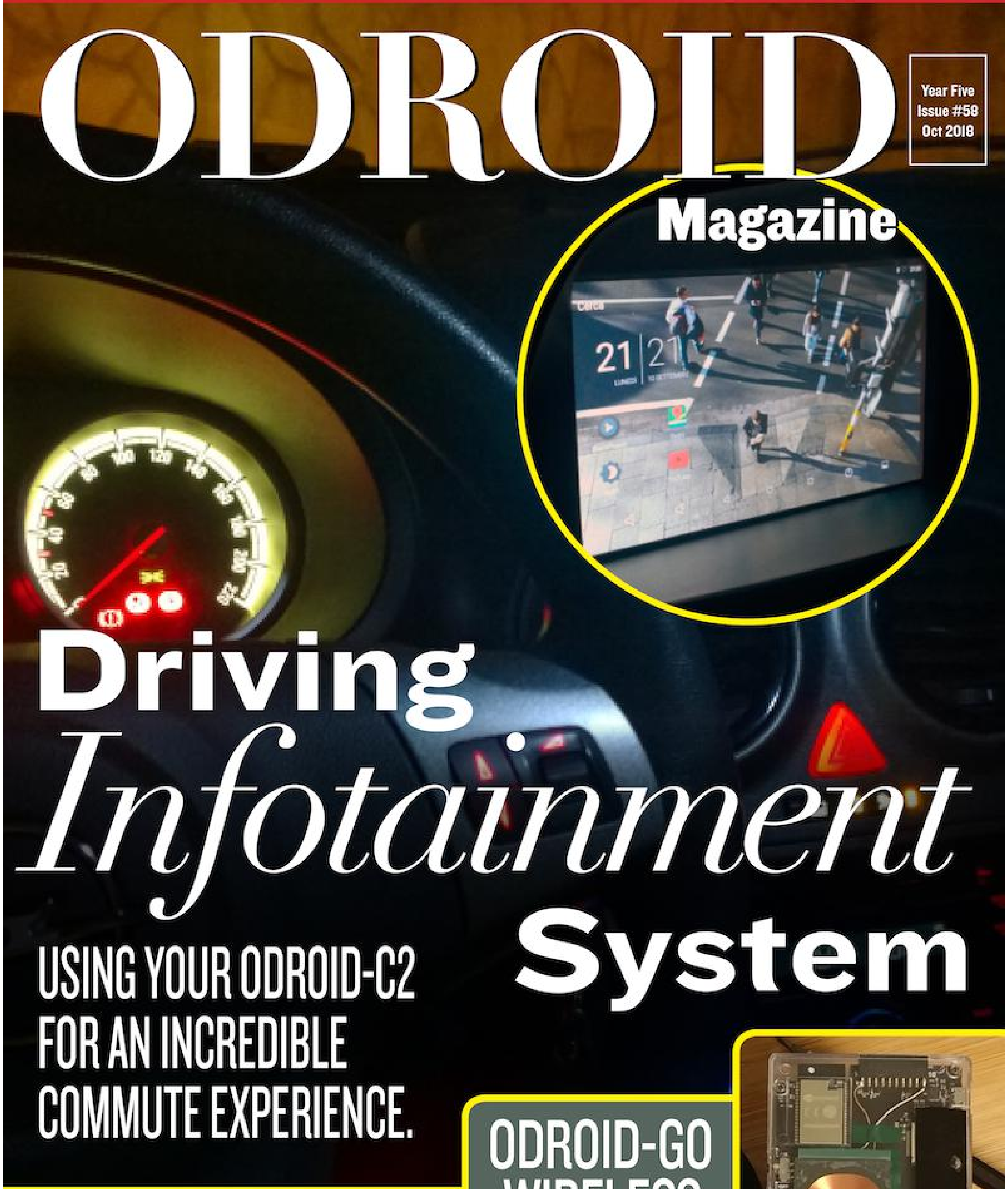
ODROID Bench • Linux Nagios • Coding Camp • Linux Gaming Classics

ODROID Magazine

Year Five
Issue #58
Oct 2018



Driving *Infotainment* System

USING YOUR ODROID-C2 FOR AN INCREDIBLE COMMUTE EXPERIENCE.

ODROID BENCH:

A REMOTE ODROID EXPERIENCE ZONE TO PREVIEW AND TEST ODROIDS

ODROID-GO WIRELESS:

ADDING WIRELESS CHARGING TO YOUR NEW HANDHELD

Linux Gaming on ODROID: Game Nostalgia Part 1 – Games I always come back to

October 1, 2018

I want to talk about games I used to play, or games I keep coming back to even today and games that I play on the ODROID

Car Infotainment System: Using an ODROID-C2

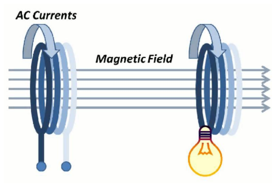
October 1, 2018

I'm proud to present my "ODROID Car Infotainment System"

ODROID Bench

October 1, 2018

We have set up a remote ODROID experience zone for someone who wants to measure, preview, and test the performance of ODROID SBCs. The ODROID SBCs in our testbed connected to Gbit ethernet and are open to the public. The experience zone offer benchmarks for performance, cloud server and more



Wireless Charging: Adding Qi Wireless Charging to the ODROID-GO

October 1, 2018

To show that this device was better than people were saying, so I made my first video about the GO: a mini-review video about the ODROID-GO. After many more videos, I'm now writing an article to be published, which I have never done before, so the goal is to add

Coding Camp – Part 5: Read the ODROID-GO built-in battery voltage

October 1, 2018

We will learn how to get the status of the battery with Arduino in this guide. The LCD will display how much battery voltage remains in volts.

Introducing NEMS Linux: Part 1 – The Nagios Enterprise Monitoring Server for ODROID Devices

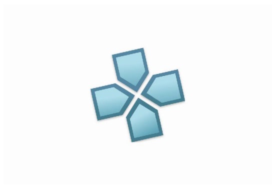
October 1, 2018

NEMS Linux has evolved to be what I feel is the best out-of-the-box Nagios experience available. As a Nagios user myself, this is the Nagios server I have longed for. As NEMS has continued to grow, I set out to find a more powerful platform than the Raspberry Pi. That's

BASH Basics: Introduction to BASH – Part 5

October 1, 2018

Our adventure into scripting continues with more tests, if statements, input and functions; we also do calculations from within BASH. First, let's look into filename manipulation, since this seems to be one of the most popular things needed in beginner scripts. As a bonus for making it through a lot



GBM Video Driver

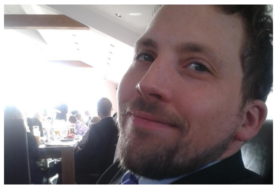
October 1, 2018

This is a guide to install GBM-enabled user space drivers and build retro gaming emulators.

Coding Camp – Part 6: Generate sound from the ODROID-GO speaker

October 1, 2018

Let us learn how to use the DAC output as a sound tone generator.



Meet An ODROIDian: David Knight

October 1, 2018

Please tell us a little about yourself. I live in Newcastle, UK, working as an optometrist working in the refractive surgery sector. My day job involves managing patients who have had cataract or laser eye surgery. I was very studious and introverted in my youth, always having my head in ▶

Linux Gaming on ODROID: Game Nostalgia Part 1 – Games I always come back to

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

Last time I talked a lot about my past, how I grew up and what systems mostly influenced my childhood. This and other events also sparked a thread in the forum about a topic that I call Game Nostalgia. I thought this was a nice topic to pick up but more related to ODROIDs and see how I can project my nostalgia in the ODROID environment. Therefore I want to talk about games I used to play, or games I keep coming back to even today and games that I play on the ODROID, for that matter.

Loom

I think I have played nearly all LucasFilm/LucasArts adventure games, especially all of the ones that were created with the SCUMM Engine, starting with Maniac Mansion, which gave the engine its name. I consider the LucasArts adventure games to the best games I have ever played and I am glad I had the chance to experience these games when I was growing up. Many “console” gamers likely missed most of these games, as Point-And-Click adventures were mainly available on “computers” and not consoles, and I can only say if you never played LucasArts Adventure games in your life, you missed a great deal of gaming history.

Monkey Island, Indiana Jones, Loom, The Dig – these games really changed me and carried me away in wonderful, bizarre and magical worlds, as a child and even as an adult these days.

Loom which is also a LucasFilm game will always hold a special place in my heart, which might be hard to understand, as commercially and also critically the game was not as successful as other LucasArts games. But for me, the game is very fascinating, and many other LucasArts games referenced this game over and over again as hidden (or not so hidden) Easter eggs.

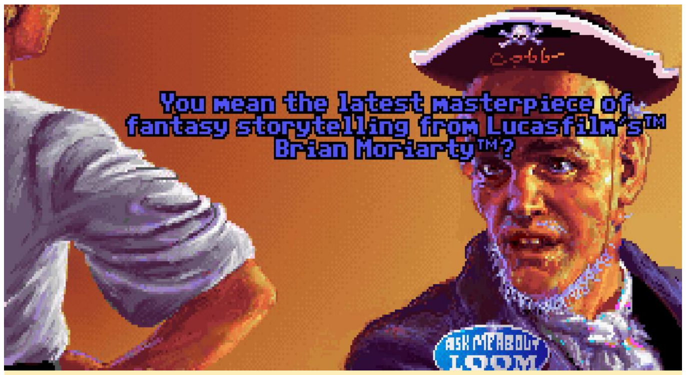

You mean the latest masterpiece of
fantasy storytelling from Lucasfilm's™
Brian Moriarty™?
ASK ME ABOUT
LOOM

Figure 01 – Loom reference in The Secret of Monkey Island (very subtly and hidden reference... not)

Loom is a game full of "guilds": the weavers, the glass makers, the blacksmiths and so on. You are part of the weavers guild. and your guild invented "The Great Loom", which allows the weavers to weave the threads of reality itself, which of course was deemed forbidden right away. The weavers also wove the great tapestries which show the history of the guild. Although the weavers do not use the Loom, they are still very capable spell weavers who can cast spells with the help of a distaff and combinations of musical notes.

The story unfolds around Bobbin, who is a child of the Loom, as he came to existence through "The Great Loom" itself, which is explained in a beautiful audio drama that came on a cassette if you bought a specific version back in the days. Nowadays, you can find the audio drama on YouTube if you want to listen to it. He ends up as the last of his guild and tries to find the rest of his kind, which is where you take over to control him. You learn magical spells on your way and have to learn how and when to use them to solve different puzzles.

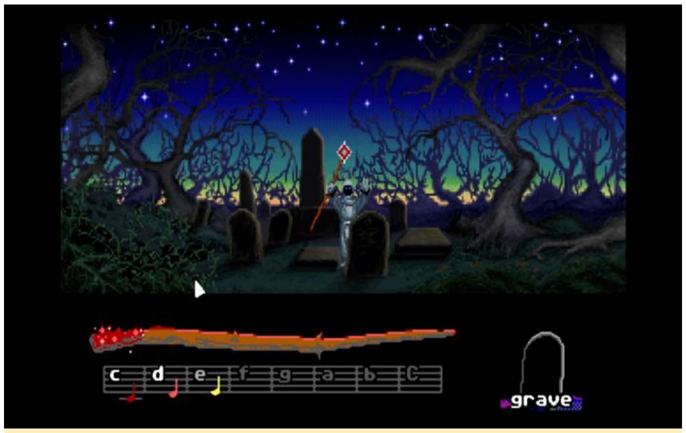


Figure 02 – Disturbing the dead is a very bad idea Bobbin. You were told not to do that, don't try it!!!

What fascinated me about this game the most is the real use of "magic", which is something you did not find in any other game of it's time. There were games such as "Simon the Sorcerer" which never casts a single spell despite it's name, or the Legend of Kyrandia series, which is about young magicians and once again you do not use any magic at all (a bunch of potions, but no "real magic"). I always found this very confusing, there are so many games about so called "wizards or magicians" but none of them really casts any spells. Even much later, these so called Harry Potter games for example are more "shooter" then that you actually use real SPELLS to solve something. A wand that shoots sparkles to kill monsters and goes "pui pui" is not "magic" for me, it is just a stupid shooter with a magic touch.

Loom was different: you had spells that could open doors, you had spells that could "refill" certain things (a cup of wine for example), you had a spell that could even turn straw to gold! You had over 15 spells (called drafts) you could learn within the game that you used to solve riddles.

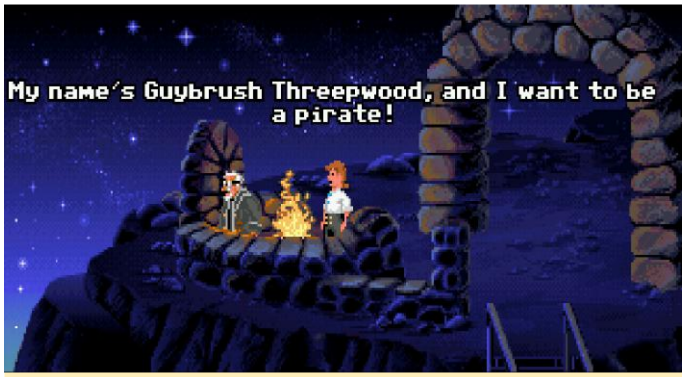
I really liked the story of this game and how it unfolds. It was originally planned as a trilogy, but was canceled after this first part. I also love its classic Swan Lake soundtrack. I used to play the game at least once a year, and it took me only something between 30 to 60 min to finish the game. If you want to play it nowadays on ODROIDs, you can use ScummVM to run it perfectly fine on ODROIDs. There are different versions of this game, including a CD "talkie" version where all text was dubbed in English.

The Curse of Monkey Island

Believe me, I love all the Monkey Island games, and I had a blast with The Secret of Monkey Island and Monkey Island 2: LeChuck's Revenge. I highly recommend these games to anyone out there. I was hooked to the story of Guybrush when he first set a foot on Mêlée Island™ and said he wanted to be a pirate.

Figure 03 – The Curse of Monkey Island

However, if you never played any Monkey Island game in your life and only have time (for whatever reason) to play a single Monkey Island game, I highly recommend playing The Curse of Monkey Island! Although The Curse of Monkey Island is not even an "official" Monkey Island title, as it is not out of the quill pen of Ron Gilbert, it is still an amazing game.



My name's Guybrush Threepwood, and I want to be a pirate!

Figure 04 – This is how everything started back in 1990

Monkey Island 1 and 2 are all time classics and were remade a couple years back in a remastered edition which improved graphics and included full voice acting that did not exist in the original game back then, but they are not compatible with ScummVM which means you cannot run them on ODROIDs. Since the original Monkey Island 1 and 2 were very pixelated, they have not aged that well, but they are still fun to play if you like old 8-bit graphics. Some new games try to replicate the look and feel of that era, but Monkey Island 3 (The Curse of Monkey Island) had already "high resolution" 640×480 graphics which are drawn in a very beautiful comic style, which even look amazing today.

Figure 05 – Monkey Island 3

Figure 06 – Monkey Island 3

ScummVM has no problem scaling these graphics, and they look like they were made for 1080p the same as they were for 640×480, which means you can enjoy the game today like it was in 1997 when it came out. This is actually something I never understood. There are certain types of graphics that do not "age" (comic style is the best example for this), but there are only a few games that really used it. They still look amazing 20 years later, and you could think it was a new game that you just bought.

The story and dialog is so funny that it is worth spending your time in the game just asking every possible question there is just to hear all the funny comments and dialog the game has. The voice acting is superb, and the music by Michel Z. Land who made all the music for the Monkey Island games is iconic and really fits the Caribbean setting. The game takes references to the previous games (Monkey Island 1 and 2), and brings back characters from these games as well, so if you want to understand every little gag in the game you should play Monkey Island 1 and 2 as well. This is one of my favorite games of all time and it runs perfectly on ODROIDs, so you can have the complete pirate Caribbean experience using ScummVM.

Dune 2

Dune 2 was my first RTS (Real Time Strategy) game. In fact, it's called the "Grandfather of all RTS games", and I played it back then on my trusty Amiga. The game was amazing: Westwood paved its way in the RTS genre with Dune 2, and classics like the famous Command and Conquer series is what followed as a result of the success of Dune 2.

Dune 2 had everything you expect from a RTS game. You collect resources (spice), you build up your base and an army, you head out to crush your enemy and his base. One of the things I remember about the Amiga version is that it already had voice samples that tell you, for example, when your buildings were completed.

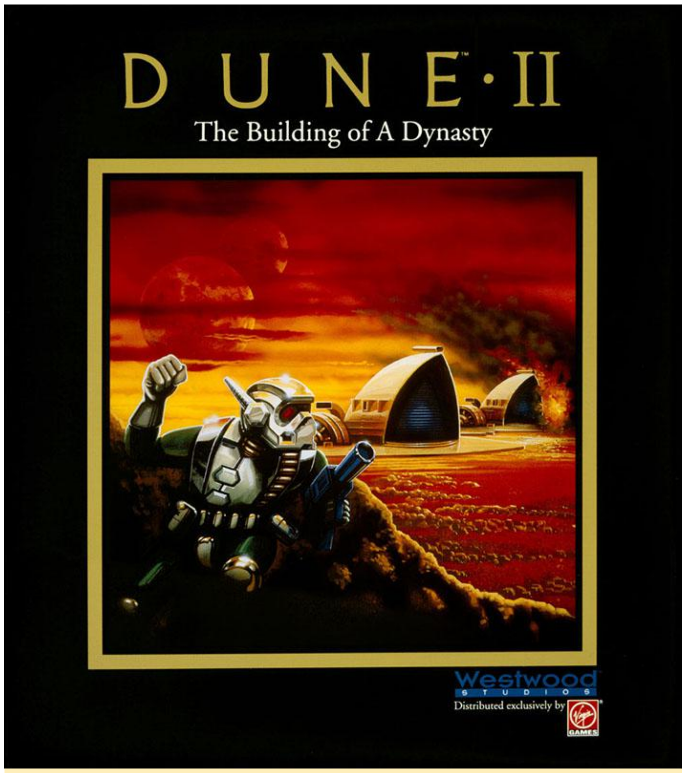
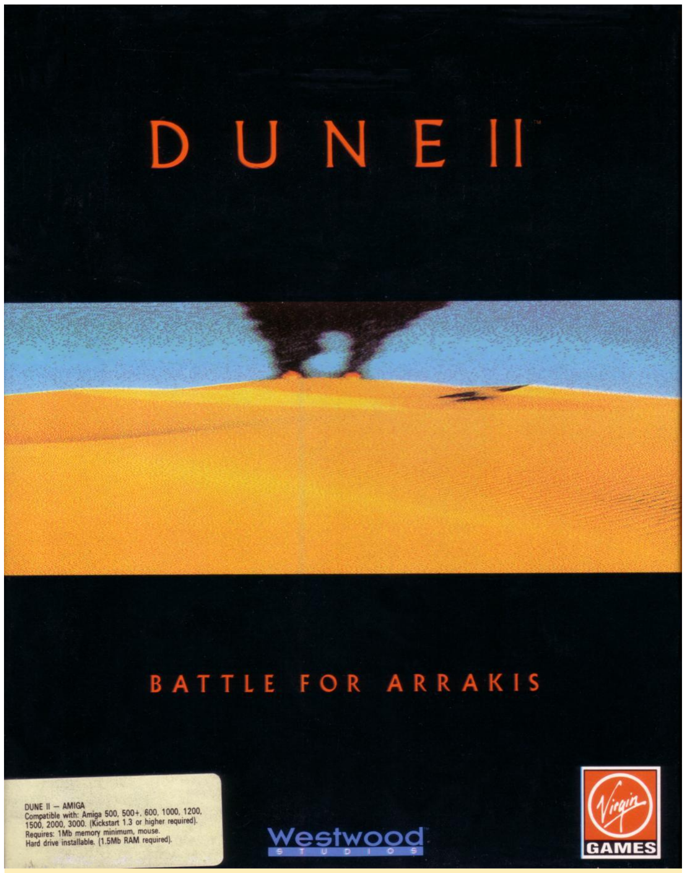


Figure 07 – Dune 2 The Building of A Dynasty


DUNE II
BATTLE FOR ARRAKIS
DUNE II — AMIGA
Compatible with: Amiga 500, 500+, 600, 1000, 1200, 1500, 2000, 3000. (Kickstart 1.3 or higher required).
Requires: 1Mb memory minimum, mouse.
Hard drive installable. (1.5Mb RAM required).
Westwood STUDIOS
Virgin GAMES
Figure 08 – Dune 2 The Building of A Dynasty

In this game, you had 3 different factions to choose from. All of them had a set of units that were identical between each faction, and some units that were unique for each faction. Each with their advantage and disadvantage, although I prefer the nuclear missile from the Harkonnen any time over the Fremen of the Atreides (special weapons from the Palace).

When it first came out, Dune 2 was revolutionary. You controlled multiple units at once over the map and so did the enemy. You had to concentrate on building, resource collection and your attacks/defense at the same time. Things that are normal today were quite new back then.

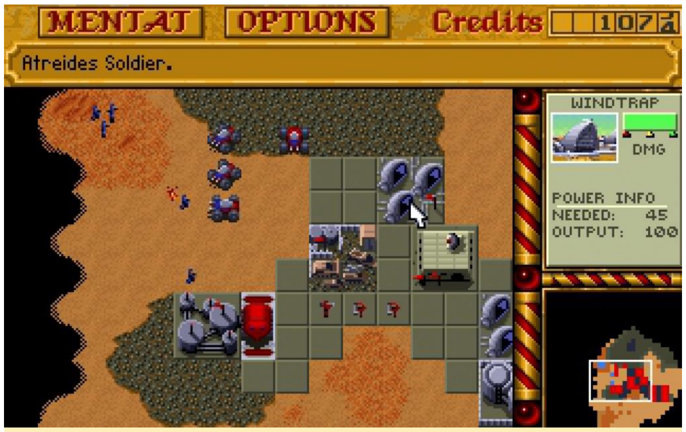


Figure 09 – Dune 2 original graphics

While Dune 2 has aged somewhat well, its controls are not very intuitive. You could not select multiple units at once to order commands, but had to give commands for each unit individually. Imagine moving an army of 30 or more units from one place to another (yes we did it the hard way “back in the old days”).

Thankfully nowadays there are several remakes of Dune 2 which include improved controls, allowing you to move entire armies at once, and have other improvements, such as build queues rally points, and often also improved graphics (higher resolutions and filters to improve the look). I personally prefer Dune Legacy for that, which is available on ODROIDs and runs very good.

Dune Legacy

Every now and then I try out some updates of Dune Legacy and end up playing the game for hours and days afterwards as the game is still very good as a strategy game, and I lose myself in its battles.

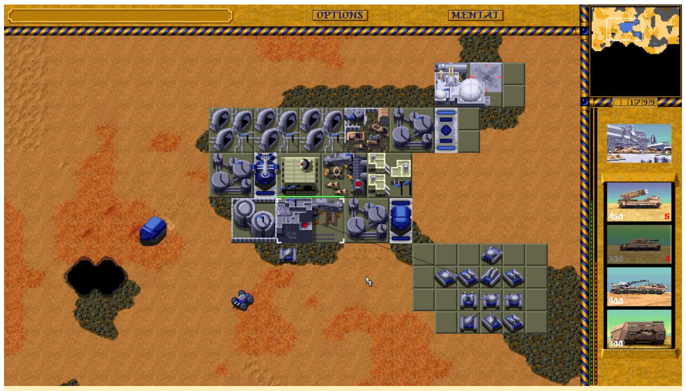


Figure 10 – Dune legacy

While talking about Dune 2 there is also a game called “Dune”. It is a mixture of Adventure and Strategy game, as you loosely follow the story of Paul Muad’dib from House Atreides as he learns about the fremen and their ways. This game is impressive as well, and has one of the best soundtrack, especially the DOS CD version with video cut-scenes and 3D renderers is a very impressive game which runs fine on DOSBox on ODROIDs, but I think I covered that game a while back already in my DOS games articles.

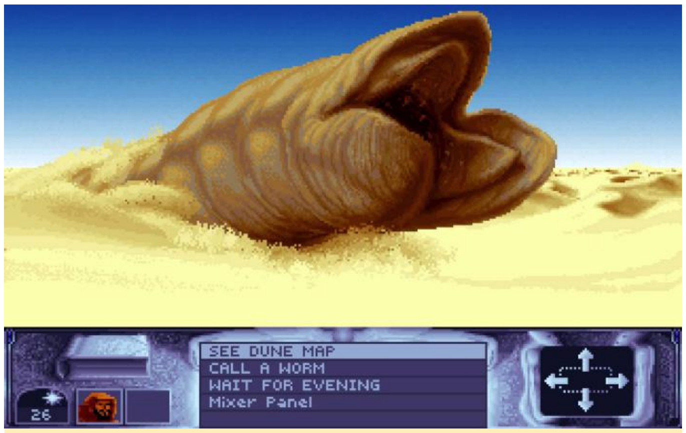


Figure 11 – Dune, epic story of gathering spice

UFO Enemy Unknown / XCOM – Ufo Defense

This one is really special to me, I cannot remember when I got into this game. I know it existed on the Amiga, but I think I played it much more intensely on the PC under DOS. I absolutely crave this genre, although the genre is really hard to define for me. Most people call it a strategy game with turned-based tactics, or call it management simulation with turn based tactics, but I think it is much more specific what fascinates me about this game genre and I know nothing that is similar to it besides the XCOM games and variations of it that came out over the years.

Figure 12 – UFO Enemy

The game is rather complex and takes a lot of time to play, master and complete, but it is well worth it. The scenario is that aliens are here, they arrived at the earth, flying around with their alien space crafts abduct humans, terrorize cities and who knows what else they have planned. The different governments of earth realize they are not fit to take on an enemy like this on their own and agreed to form a united force "XCOM" to defeat the alien intruders.

For this, each country agreed to fund the XCOM group with money and personal to achieve this goal and this is where you come into play. You are the commander of the XCOM and your choices will affect the outcome of this scenario. You have distribute your time and resources to mount a defense against the intruders. You have to explore new technologies, and means how to defeat this enemy. You have to build your own weapons, as these new technologies do not exist yet and you cannot just buy them. You have to train your soldiers, equip them with armor for their protection, weapons to mow down the enemy, and other items such as med-kits, grenades, or motion trackers.

You also have to equip your own aircrafts to shoot down enemy UFOs. Have to analyze collected items and weapons from the enemy. And of course beat the enemy on the ground, wherever you can find them.

For me, the whole mix is what makes this game so great. The fact that you research your new technologies build new weapons, armors, and aircrafts to improve your chances of surviving. The constant improvement, not only through better equipment but also from your soldiers, that get better the longer they are in the service and the more missions they participated and the more enemies they were able to take out. You build up a connection to your soldiers, especially the higher they become. Losing a rookie will not really matter as you can simply recruit a new one, but a veteran soldier is very precious and losing one of these normally makes your life a lot harder, which is also the reason why I would reload the game and try again.

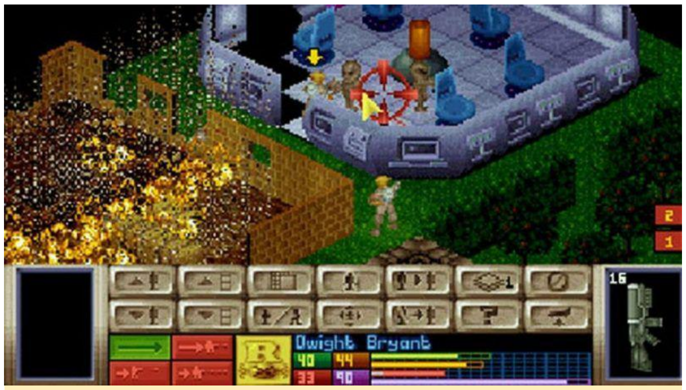
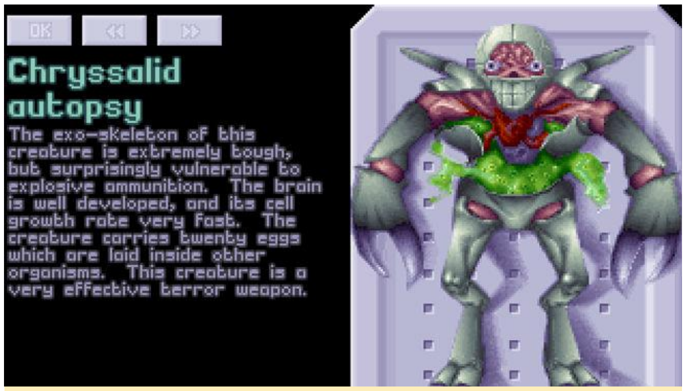


Figure 12 – UFO Enemy Unknown original graphics, fighting the "grays"

It was also one of the first game where I used a hex-editor to manipulate the save game back when I was a child, but only to create more "Elerium", which is an alien mineral which is required for high tech weapon and equipment. I constantly ran out of it as a child, so yeah I did a little bit of cheating back then.

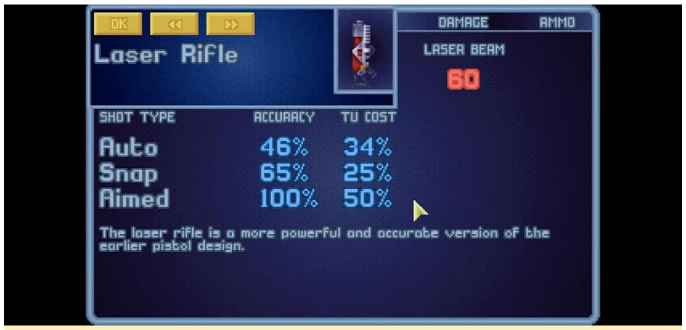
What always fascinated me the most about this game is the research. The progressing through researching and developing new technologies is what kept me playing this game.



Chryssalid autopsy

The exo-skeleton of this creature is extremely tough, but surprisingly vulnerable to explosive ammunition. The brain is well developed, and its cell growth rate very fast. The creature carries twenty eggs which are laid inside other organisms. This creature is a very effective terror weapon.

Figure 13 – Autopsy of one of the most dangerous aliens out of XCOM



Laser Rifle

DAMAGE	AMMO
LASER BEAM	
60	

SHOT TYPE	ACCURACY	TU COST
Auto	46%	34%
Snap	65%	25%
Aimed	100%	50%

The laser rifle is a more powerful and accurate version of the earlier pistol design.

Figure 14 -Autopsy of one of the most dangerous aliens out of XCOM

Research and reading all those details about the aliens and weapons or just items you found was always a highlight for me. It is what this game makes unique: power through knowledge. The more you research, the better you get the more chance you have to defeating the enemy. Nowadays, if I want to play the game I use OpenXcom on the ODROID, as it has many extra features and allows for mods, extra content, and different settings to improve the gaming experience.

I found that OpenXCom works beautifully on ODROIDs. I like the options that it gives you, such as disabling features of the original game that made the game really hard (for example fights at night), or simple improvements like right clicking the up arrow on the research screen to add all available researchers to certain research project, or the ability to "produce and sell" (produce items that are not available on market e.g. laser rifles, and sell them for a profit). OpenXcom allows you to play the original UFO Enemy Unknown, as well as the second game called "Xcom Terror from the Deep" which is pretty much the same as the first game but now you fight the aliens coming from the sea and from under water.

OpenXcom also has a lot of mods that introduce completely new scenarios in the universe, or create entirely new universes. Combined with the ability to play improved music, and sounds, add more weapons to research and so on, OpenXcom is really one of a kind, and well worth having on the ODROID.

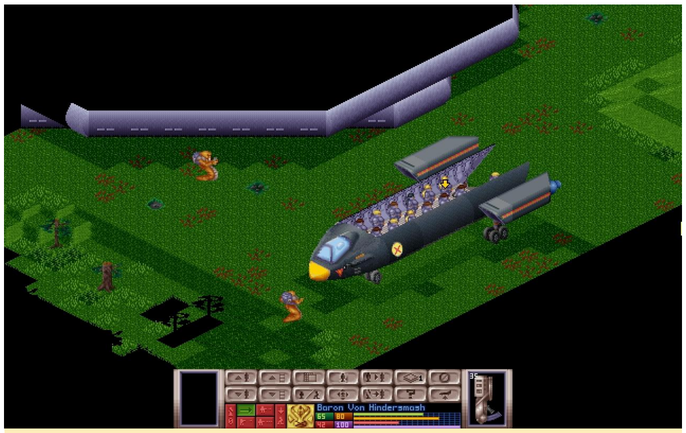


Figure 15 – Higher resolutions, just one of the many features of OpenXCom

Aside from the original UFO Enemy Unknown and Terror From the Deep games I also greatly enjoyed playing the remake also called XCOM: Enemy Unknown, which brings the genre in modern 3D.

Figure 16 – XCOM: Enemy Unknown from Firaxis Games

This game even came out for Android which technically make it compatible with ODROIDs as well, but I would really love to see a armhf Linux port for it. There is also the fan-made UFO Alien Invasion (UFOAI) which is available for ODROIDs at https://forum.odroid.com/viewtopic.php?f=91&t=2375. I have not played this game too much yet, but I want to as it is quite true to the original game mechanics.

Another great remake is Xenonauts which uses 2D graphics in an isometric view, which is a very interesting art-style. I strongly believe this game could run natively on ODROIDs if it was recompiled for armhf, as it only uses SDL2. Still, it is not currently available as such.

There is one more series based on the old UFO/XCOM series that I really like, which is also called UFO and is either called UFO trilogy or UFO Aftermath series, named after the first game of the series. UFO Aftermath, UFO Aftershock and UFO Afterlight are 3D games released between 2003 and 2007, and these games blew my mind once again.

A lot of people do not like them, as they are not as "turn-based" as the original XCOM games, but as I said earlier, for me these games are not about the turn-based tactics, but rather about the research, development, improvement and character development.

The games also had tons of bugs, and the story (especially from the first two) was not very well written (especially the ending), but the gameplay is what always counted for me, and they did a lot of improvement here compared to the other games in my opinion.

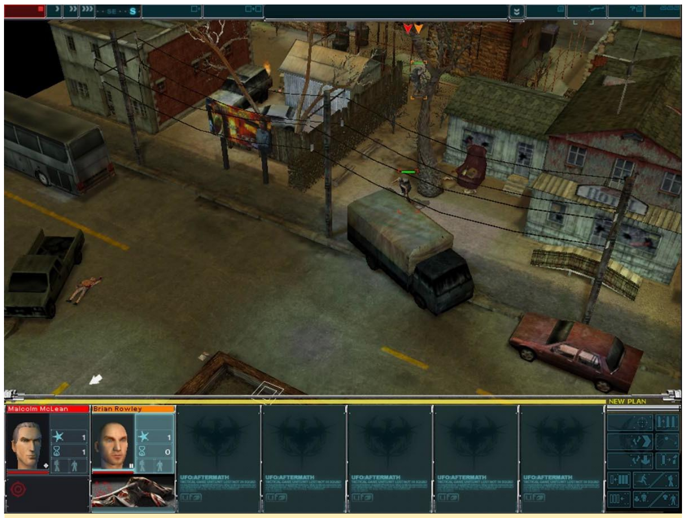


Figure 17 – UFO Aftermath has a much darker setting and is fully 3D, but sadly is fixed to 1024×768 resolution

In the XCOM games, you start off with a fully equipped base and soldiers and start research "future tech" like lasers and such right away. In UFO Aftermath, you start in a world that was surprised by aliens and humanity was nearly wiped out. Towns are crumbled and mutants roam around attacking everyone. You barely have any weapons at all, and finding an M4 machine gun is something like a treasure. It takes a while until you start developing "future tech", and until that point, you are happy with the weapons you find before you manage to replicate the alien technologies and improve a lot.

In the XCOM series, you have to satisfy different nations to fund your operations. If you do bad in a country, you lose money, and as aliens get stronger and stronger, in time you will lose a lot of money, so the game gets harder, and you have less resources to deal with it, which is slightly annoying. In UFO Aftermath, society collapsed, money has no value, and that's how it should be if you fight a global invasion force. Your "currency" is time. The harder a project is, the longer it takes to develop, and you can change this by devoting more work forces to a common goal. You do this by selecting what your different bases should do: Research, Build Technologies, or Defend with a military force.

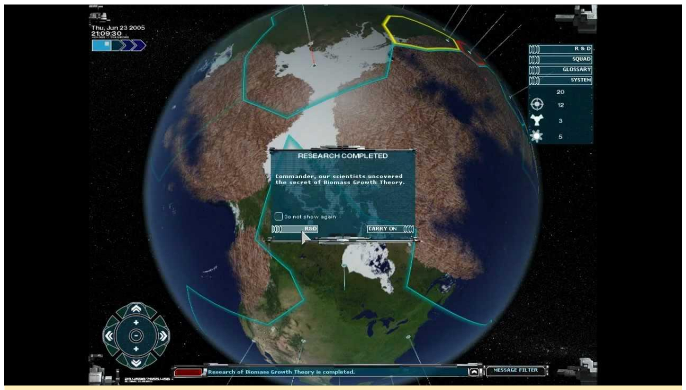


Figure 18 – The brown stuff is slowly covering the earth

The first game of the series is the most interesting story-wise and how it evolves from handguns and old rifles to alien technology. The second game in the series is the most annoying of the three. It plays after the first one and assumes that you messed up. It now has resource management back, and it is very hard and confusing. It has different sections of humans that you can recruit or make your enemy. It is not really much of an improvement over UFO Aftermath, which is my preference of the two. The third one, UFO Afterlight, is the most advanced of the three, and offers much improved 3D graphics. It is not stuck at 1024×768, and this time, you fight on Mars, our red neighbor.

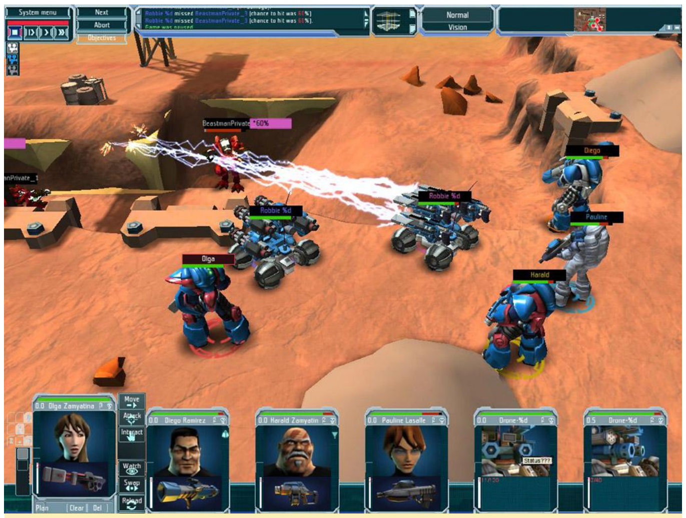


Figure 19 – Highly improved graphics in UFO Afterlight lots of new aliens

I hope one day we can play at least UFO Aftermath on the ODROID as well, but until, then I love to play OpenXcom and UFOAI. These games are really amazing, and if you have never played them, try them out.

Car Infotainment System: Using an ODROID-C2

October 1, 2018 By @poptmartone ODROID-C2, Tinkering, Tutorial

I'm proud to present my "ODROID Car Infotainment System". It is a combination of an ODROID-C2 and an Arduino platform capable of CAN communication with the car BCM module. This feature allows the steering wheel controls to "communicate" with Android operating system through a further infrared (IR) interface, as I proposed some time ago. Here is the complete hardware list:

- ODROID-C2
- VU7 Plus display assembly
- Hi-Fi Shield plus audio board
- SmartPower 2 power board
- Bluetooth USB module
- Wifi USB module
- GPS USB module
- Externally powered USB hub
- Cooling fan
- Arduino Nano
- Niren MCP2515 CANBUS transceiver
- Step down converter 12/24 V -> 5V (to power the Arduino and the CAN module)
- Many male and female DuPont connectors

The Arduino Nano works to help the C2 and provide the main operating logic, system on / off, IR interface, CAN bus communication and stereo head unit remote control services, while the ODROID board is where the fun happens. In order to minimize the build size, and to accommodate everything into my car dashboard housing, I created an electronic control board which holds all the needed electronic components as show below.

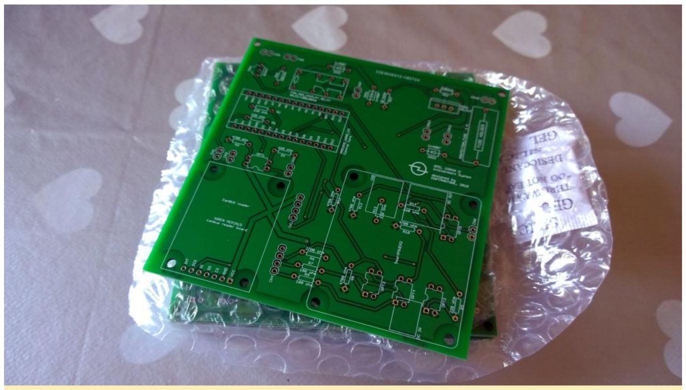


Figure 01 – Unopulated Control Board

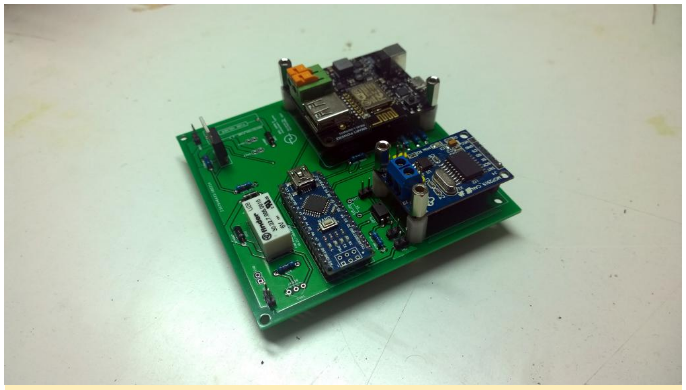


Figure 02 – Populated Control Board

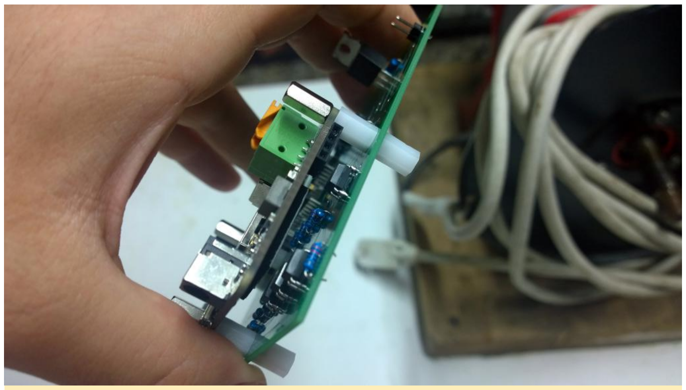


Figure 03 – Side View of Control Board

After disassembling the VU7 display kit and rearranging the components in a sort of "motherboard" this is what I got:

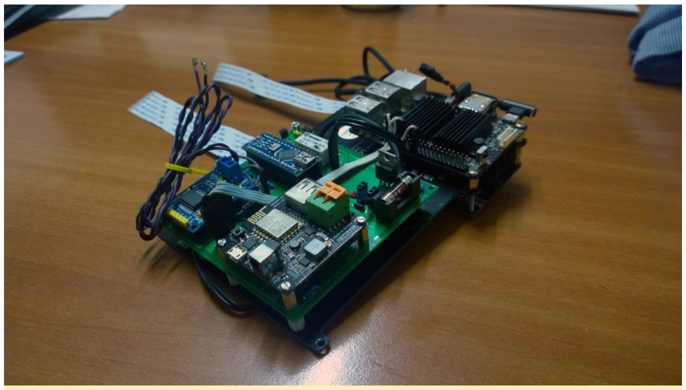


Figure 04 – All Components Connected

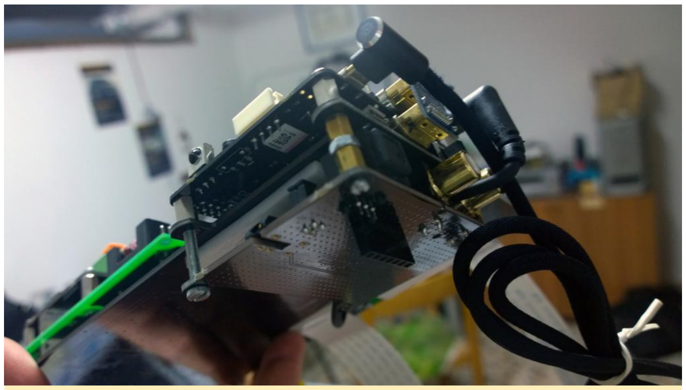


Figure 05 – Close-up View of The Board

Next, everything is, mounted into the car dashboard housing:

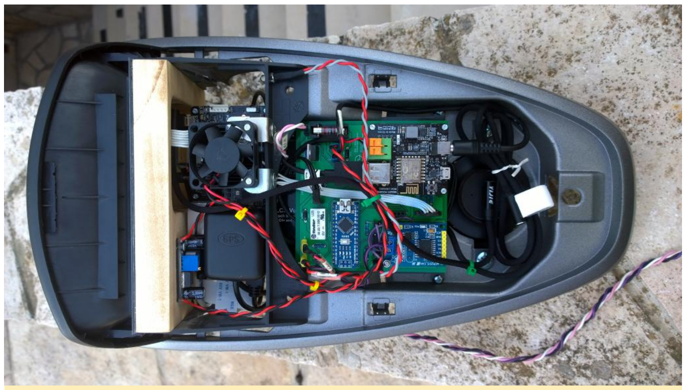


Figure 06 – Board Mounted in Car Dashboard

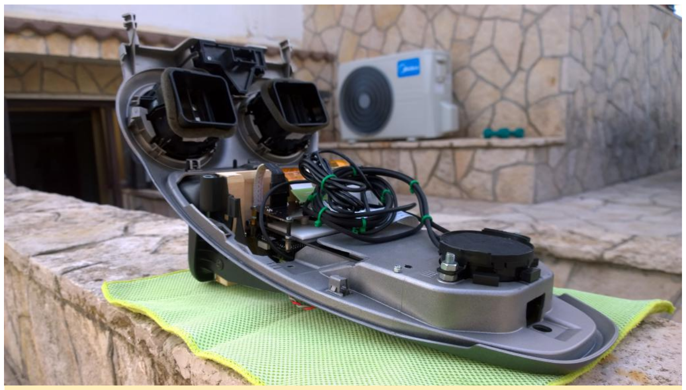


Figure 07 – Underside of Car Dashboard

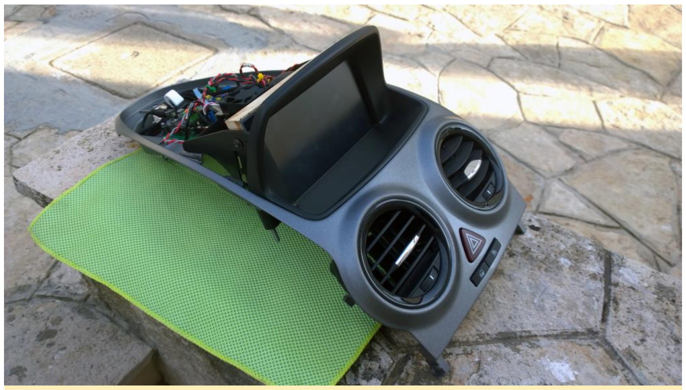


Figure 08 – Front of Car Dashboard

Note the infrared LED

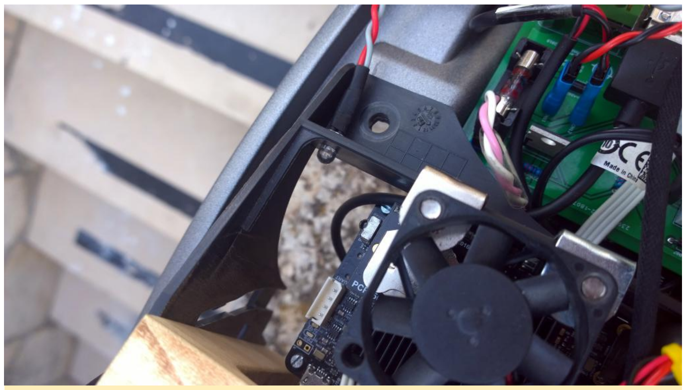


Figure 09 – Infrared LED

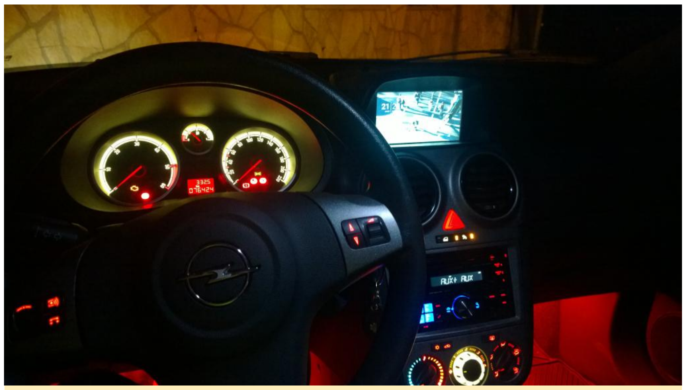


Figure 10 – Everything Installed

Figure 11 – Up and Running

Please watch a completed demo video to see the whole system in action at https://www.youtube.com/watch?time_continue=79&v=uQf7kH7wbuQ. For more detail or to ask a question, please follow the link to the ODROID forum posting at https://forum.odroid.com/viewtopic.php?f=140&t=32189.

ODROID Bench

October 1, 2018 By Dongjin Kim Linux, ODROID-C2, ODROID-XU4, ODROID-HC2

We have set up a remote ODROID experience zone for someone who wants to measure, preview, and test the performance of ODROID SBCs. The ODROID SBCs in our testbed connected to Gbit ethernet and are open to the public. The experience zone offer benchmarks for performance, cloud server and more via SSH.

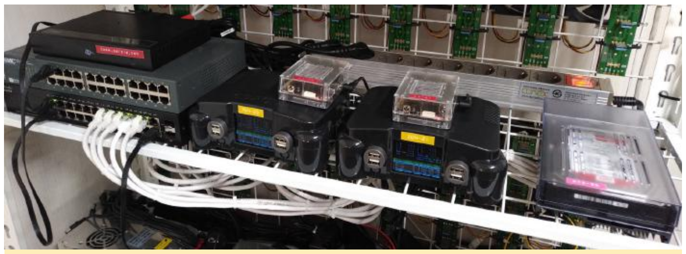


Figure 1 – ODROID SBCs on the testbed connected to maze.odroid.com

What devices are available on the testbed?

We provide a fully dedicated 1Gbps network with a domain at maze.odroid.com and different ODROID hardware setups to offer an experience to use any ODROID SBCs that you are interested in. From this environment, you can see hardware performance and computing power. Future ODROID SBCs would be accessed before launching.

Which operating system is installed on the ODROIDs?

It offers Debian Stretch in a Docker container on top of a recent Linux kernel that is maintained and updated by Hardkernel. Hardware performance and computing power are not affected by another environment with this configuration. If you are new at ODROID SBCs, all source code released for Linux kernel and U-boot are uploaded to the Github repositories at https://github.com/hardkernel.

What ODROID SBCs are accessible?

The beginnings will seem humble. You can have a command shell of ODROID after SSH-ing to a board through a dedicated port number, and you can even run or install a package. As of today, we offer 5 ODROID SBCs with a basic set up:

- 2x ODROID-XU4
- 2x ODROID-C2
- 1x ODROID-HC2 Home Cloud kit with 3.5″ 1TB HDD

How are they accessible?

Only 4 out of 5 ODROID SBC are ready to accept your commands through an SSH-ing with a dedicated port number.

```
Login account: odroid
Password: odroid
```

Once you access an ODROID SBC, you are fully granted to run any Linux commands to play with (some commands could be restricted for security reasons), and you can jump to another ODROID SBC in the same local network as well.

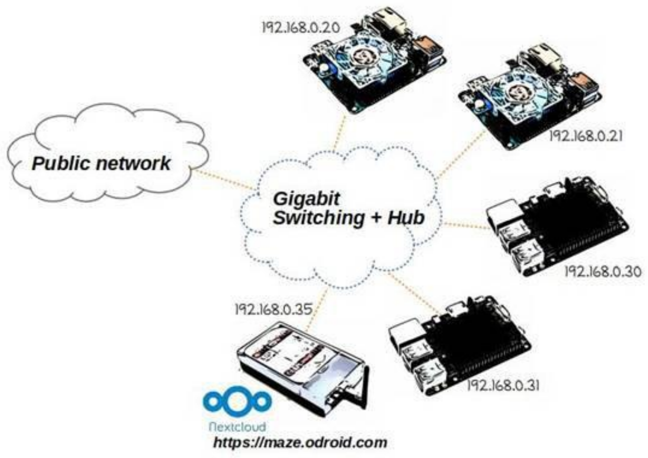


Figure 2 – Network setup

Unit	ODROID Platform	External Port #	Internal Port #	Internal IP address	Note
#1	ODROID-XU4	2220	2222	192.168.0.20	8GB uSD + 8GB eMMC
#2	ODROID-XU4	2221	2222	192.168.0.21	8GB uSD + 8GB eMMC
#3	ODROID-C2	2230	2222	192.168.0.30	8GB uSD + 8GB eMMC
#4	ODROID-C2	2231	2222	192.168.0.31	8GB uSD + 8GB eMMC
#5	ODROID-XU4	432	NONE	192.168.0.35	8GB uSD + 1TB HDD

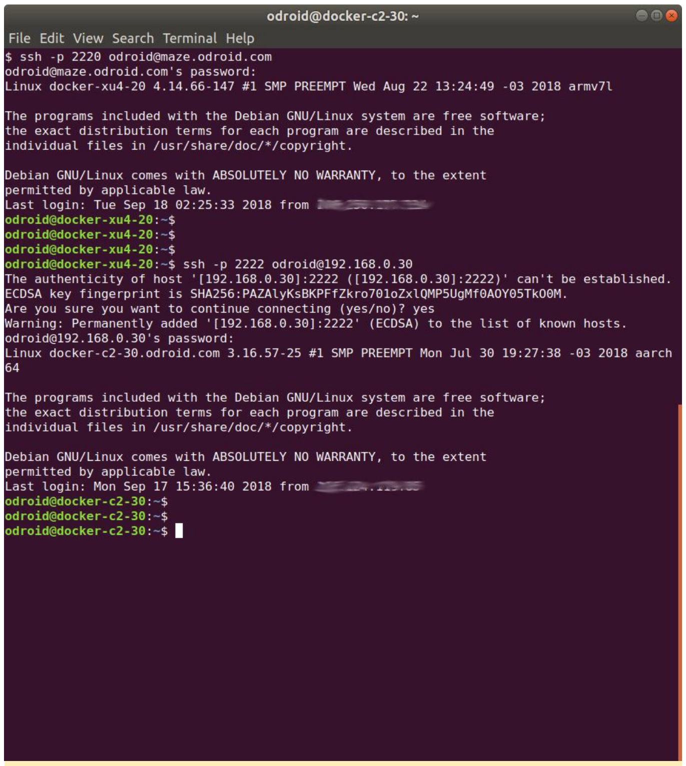
Figure 3 – Diagram of ODROIDs attached to maze.odroid.com

For instance, if you like to access the ODROID-XU4 in which port number is 2220, you can run the following command, which will let you access an ODROID-XU4 with internal IP address 192.168.0.20:

```
$ ssh -p 2220 odroid@maze.odroid.com
```

Once you have access to an ODROID, you can connect to any other ODROID in the same network with its dedicated internal IP address. For example, this command will let you access ODROID-C2 from ODROID-XU4 which you have connected with the previous command:

```
$ ssh -p 2222 odroid@192.168.0.30
```



```
$ ssh -p 2220 odroid@maze.odroid.com
odroid@maze.odroid.com's password:
Linux docker-xu4-20 4.14.66-147 #1 SMP PREEMPT Wed Aug 22 13:24:49 -03 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Sep 18 02:25:33 2018 from [redacted]
odroid@docker-xu4-20:~$
odroid@docker-xu4-20:~$
odroid@docker-xu4-20:~$
odroid@docker-xu4-20:~$ ssh -p 2222 odroid@192.168.0.30
The authenticity of host '[192.168.0.30]:2222 ([192.168.0.30]:2222)' can't be established.
ECDSA key fingerprint is SHA256:PAZAlyKsBKPFfZkro701oZxlQMP5UgMf0AOY05TkO0M.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[192.168.0.30]:2222' (ECDSA) to the list of known hosts.
odroid@192.168.0.30's password:
Linux docker-c2-30.odroid.com 3.16.57-25 #1 SMP PREEMPT Mon Jul 30 19:27:38 -03 2018 aarch
64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Sep 17 15:36:40 2018 from [redacted]
odroid@docker-c2-30:~$
odroid@docker-c2-30:~$
odroid@docker-c2-30:~$
```

Figure 4 – Connecting to the ODROIDs via SSH

What other hardware is available?

Unlike the 4 ODROID SBCs explained so far, we also provide one dedicated cloud storage running Nextcloud with 1TB HDD. This storage device is to demonstrate how you can build your own cloud device with ODROID-XU4 and offer you an opportunity to use them before you decide to build by yourself. Everyone can access this storage with the open account, so you have to aware that all files can be accessed by anyone else. Therefore, you should not upload any private files to the storage. Also, you must not share any type of non-free files for your personal purpose using the storage.

```
Login account: odroid
Password: odroidfun
```

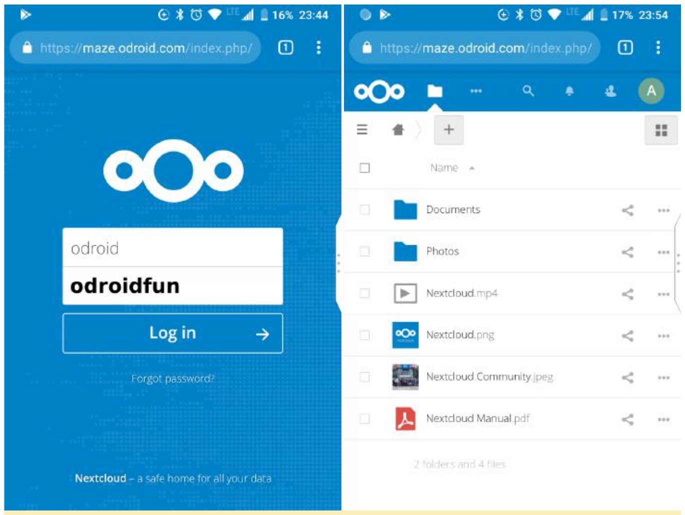
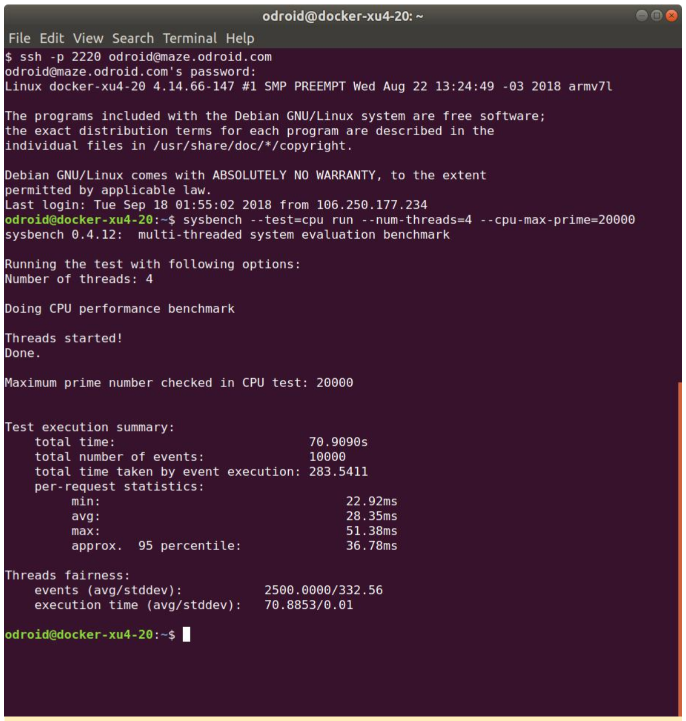


Figure 5 – Connecting to cloud storage from a mobile device

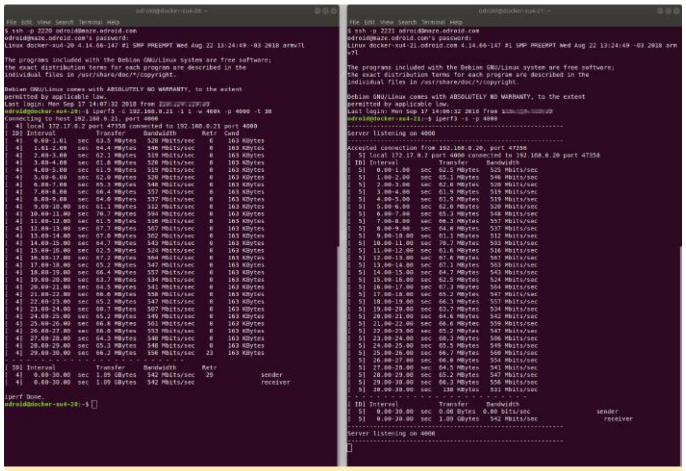
What is doable and not?

All ODROID SBCs in maze.odroid.com are publicly accessible and opened to offer sample SBC experiences. We are pleased to use them and listen to your opinion what can be improved. You can do see the performance of hardware with benchmark tools like sysbench or even simple Linux tools like dd or ping. Also, if you are willing to run a network tool with a certain port, you are able to use the ports between 4000 and 4499 in the Docker container.



```
odroid@docker-xu4-20: ~
File  Edit  View  Search  Terminal  Help
$ ssh -p 2220 odroid@maze.odroid.com
odroid@maze.odroid.com's password:
Linux docker-xu4-20 4.14.66-147 #1 SMP PREEMPT Wed Aug 22 13:24:49 -03 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Sep 18 01:55:02 2018 from 106.250.177.234
odroid@docker-xu4-20:~$ sysbench --test=cpu run --num-threads=4 --cpu-max-prime=20000
sysbench 0.4.12:  multi-threaded system evaluation benchmark

Running the test with following options:
Number of threads: 4

Doing CPU performance benchmark

Threads started!
Done.

Maximum prime number checked in CPU test: 20000


Test execution summary:
    total time:                          70.9090s
    total number of events:              10000
    total time taken by event execution: 283.5411
    per-request statistics:
         min:                                 22.92ms
         avg:                                 28.35ms
         max:                                 51.38ms
         approx.  95 percentile:              36.78ms

Threads fairness:
    events (avg/stddev):           2500.0000/332.56
    execution time (avg/stddev):   70.8853/0.01

odroid@docker-xu4-20:~$ 
```

Figure 6 – Running Sysben on ODROID-XU4

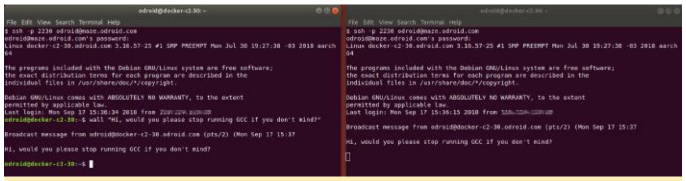


```
odroid@docker-xu4-20: ~
File  Edit  View  Search  Terminal  Help
$ ssh -p 2220 odroid@maze.odroid.com
odroid@maze.odroid.com's password:
Linux docker-xu4-20 4.14.66-147 #1 SMP PREEMPT Wed Aug 22 13:24:49 -03 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Sep 17 14:07:32 2018 from [redacted]
odroid@docker-xu4-20:~$ iperf3 -c 192.168.0.21 -i 1 -w 400k -p 4000 -t 30
Connecting to host 192.168.0.21, port 4000
[  4] local 172.17.0.2 port 47358 connected to 192.168.0.21 port 4000
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.01   sec  63.5 MBytes   528 Mbits/sec    6    163 KBytes
[  4]   1.01-2.00   sec  64.4 MBytes   546 Mbits/sec    0    163 KBytes
[  4]   2.00-3.00   sec  62.1 MBytes   519 Mbits/sec    0    163 KBytes
[  4]   3.00-4.00   sec  61.8 MBytes   520 Mbits/sec    0    163 KBytes
[  4]   4.00-5.00   sec  61.9 MBytes   519 Mbits/sec    0    163 KBytes
[  4]   5.00-6.00   sec  62.0 MBytes   520 Mbits/sec    0    163 KBytes
[  4]   6.00-7.00   sec  65.3 MBytes   548 Mbits/sec    0    163 KBytes
[  4]   7.00-8.00   sec  66.4 MBytes   557 Mbits/sec    0    163 KBytes
[  4]   8.00-9.00   sec  64.0 MBytes   537 Mbits/sec    0    163 KBytes
[  4]   9.00-10.00  sec  61.1 MBytes   512 Mbits/sec    0    163 KBytes
[  4]  10.00-11.00  sec  70.7 MBytes   594 Mbits/sec    0    163 KBytes
[  4]  11.00-12.00  sec  61.5 MBytes   516 Mbits/sec    0    163 KBytes
[  4]  12.00-13.00  sec  67.7 MBytes   567 Mbits/sec    0    163 KBytes
[  4]  13.00-14.00  sec  67.0 MBytes   562 Mbits/sec    0    163 KBytes
[  4]  14.00-15.00  sec  64.7 MBytes   543 Mbits/sec    0    163 KBytes
[  4]  15.00-16.00  sec  62.5 MBytes   524 Mbits/sec    0    163 KBytes
[  4]  16.00-17.00  sec  67.2 MBytes   564 Mbits/sec    0    163 KBytes
[  4]  17.00-18.00  sec  65.2 MBytes   547 Mbits/sec    0    163 KBytes
[  4]  18.00-19.00  sec  66.4 MBytes   557 Mbits/sec    0    163 KBytes
[  4]  19.00-20.00  sec  63.7 MBytes   534 Mbits/sec    0    163 KBytes
[  4]  20.00-21.00  sec  64.5 MBytes   541 Mbits/sec    0    163 KBytes
[  4]  21.00-22.00  sec  66.6 MBytes   558 Mbits/sec    0    163 KBytes
[  4]  22.00-23.00  sec  65.2 MBytes   547 Mbits/sec    0    163 KBytes
[  4]  23.00-24.00  sec  60.7 MBytes   507 Mbits/sec    0    163 KBytes
[  4]  24.00-25.00  sec  65.2 MBytes   549 Mbits/sec    0    163 KBytes
[  4]  25.00-26.00  sec  66.8 MBytes   561 Mbits/sec    0    163 KBytes
[  4]  26.00-27.00  sec  66.0 MBytes   553 Mbits/sec    0    163 KBytes
[  4]  27.00-28.00  sec  64.3 MBytes   540 Mbits/sec    0    163 KBytes
[  4]  28.00-29.00  sec  65.3 MBytes   548 Mbits/sec    0    163 KBytes
[  4]  29.00-30.00  sec  66.2 MBytes   556 Mbits/sec   23    163 KBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-30.00  sec  1.89 GBytes   542 Mbits/sec   29             sender
[  4]   0.00-30.00  sec  1.89 GBytes   542 Mbits/sec                  receiver

iperf Done.
odroid@docker-xu4-20:~$ 
```

```
odroid@docker-xu4-21: ~
File  Edit  View  Search  Terminal  Help
$ ssh -p 2221 odroid@maze.odroid.com
odroid@maze.odroid.com's password:
Linux docker-xu4-21.odroid.com 4.14.66-147 #1 SMP PREEMPT Wed Aug 22 13:24:49 -03 2018 arm
v7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Sep 17 14:08:32 2018 from [redacted]
odroid@docker-xu4-21:~$ iperf3 -s -p 4000
-----------------------------------------------------------
Server listening on 4000
-----------------------------------------------------------
Accepted connection from 192.168.0.20, port 47356
[  5] local 172.17.0.2 port 4000 connected to 192.168.0.20 port 47358
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  62.5 MBytes   525 Mbits/sec
[  5]   1.00-2.00   sec  65.1 MBytes   546 Mbits/sec
[  5]   2.00-3.00   sec  62.0 MBytes   520 Mbits/sec
[  5]   3.00-4.00   sec  61.9 MBytes   519 Mbits/sec
[  5]   4.00-5.00   sec  61.9 MBytes   519 Mbits/sec
[  5]   5.00-6.00   sec  62.0 MBytes   520 Mbits/sec
[  5]   6.00-7.00   sec  65.3 MBytes   548 Mbits/sec
[  5]   7.00-8.00   sec  66.3 MBytes   557 Mbits/sec
[  5]   8.00-9.00   sec  64.0 MBytes   537 Mbits/sec
[  5]   9.00-10.00  sec  61.1 MBytes   512 Mbits/sec
[  5]  10.00-11.00  sec  70.7 MBytes   593 Mbits/sec
[  5]  11.00-12.00  sec  61.6 MBytes   516 Mbits/sec
[  5]  12.00-13.00  sec  67.6 MBytes   567 Mbits/sec
[  5]  13.00-14.00  sec  67.1 MBytes   563 Mbits/sec
[  5]  14.00-15.00  sec  64.7 MBytes   543 Mbits/sec
[  5]  15.00-16.00  sec  62.5 MBytes   524 Mbits/sec
[  5]  16.00-17.00  sec  67.3 MBytes   564 Mbits/sec
[  5]  17.00-18.00  sec  65.2 MBytes   547 Mbits/sec
[  5]  18.00-19.00  sec  66.3 MBytes   557 Mbits/sec
[  5]  19.00-20.00  sec  63.7 MBytes   534 Mbits/sec
[  5]  20.00-21.00  sec  64.6 MBytes   542 Mbits/sec
[  5]  21.00-22.00  sec  66.6 MBytes   559 Mbits/sec
[  5]  22.00-23.00  sec  65.2 MBytes   547 Mbits/sec
[  5]  23.00-24.00  sec  60.3 MBytes   506 Mbits/sec
[  5]  24.00-25.00  sec  65.5 MBytes   549 Mbits/sec
[  5]  25.00-26.00  sec  66.7 MBytes   560 Mbits/sec
[  5]  26.00-27.00  sec  66.0 MBytes   554 Mbits/sec
[  5]  27.00-28.00  sec  64.5 MBytes   541 Mbits/sec
[  5]  28.00-29.00  sec  65.2 MBytes   547 Mbits/sec
[  5]  29.00-30.00  sec  66.3 MBytes   556 Mbits/sec
[  5]  30.00-30.00  sec   138 KBytes   531 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-30.00  sec  0.00 Bytes  0.00 bits/sec                  sender
[  5]   0.00-30.00  sec  1.89 GBytes   542 Mbits/sec                  receiver
-----------------------------------------------------------
Server listening on 4000
-----------------------------------------------------------
```

Figure 7 – Running iperf3 to measure the network bandwidth between two ODROID SBCs

The ODROID SBC you are accessing is running on a Docker container, so you are also able to see how the Docker container is performing enough like a native operating system on ARM hardware.



```
odroid@docker-c2-30: ~
File  Edit  View  Search  Terminal  Help
$ ssh -p 2230 odroid@maze.odroid.com
odroid@maze.odroid.com's password:
Linux docker-c2-30.odroid.com 3.16.57-25 #1 SMP PREEMPT Mon Jul 30 19:27:38 -03 2018 aarch
64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Sep 17 15:36:34 2018 from [redacted]
odroid@docker-c2-30:~$ wall "Hi, would you please stop running GCC if you don't mind?"

Broadcast message from odroid@docker-c2-30.odroid.com (pts/2) (Mon Sep 17 15:37

Hi, would you please stop running GCC if you don't mind?

odroid@docker-c2-30:~$ 
```

```
odroid@docker-c2-30: ~
File  Edit  View  Search  Terminal  Help
$ ssh -p 2230 odroid@maze.odroid.com
odroid@maze.odroid.com's password:
Linux docker-c2-30.odroid.com 3.16.57-25 #1 SMP PREEMPT Mon Jul 30 19:27:38 -03 2018 aarch
64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Sep 17 15:36:15 2018 from [redacted]

Broadcast message from odroid@docker-c2-30.odroid.com (pts/2) (Mon Sep 17 15:37

Hi, would you please stop running GCC if you don't mind?

```

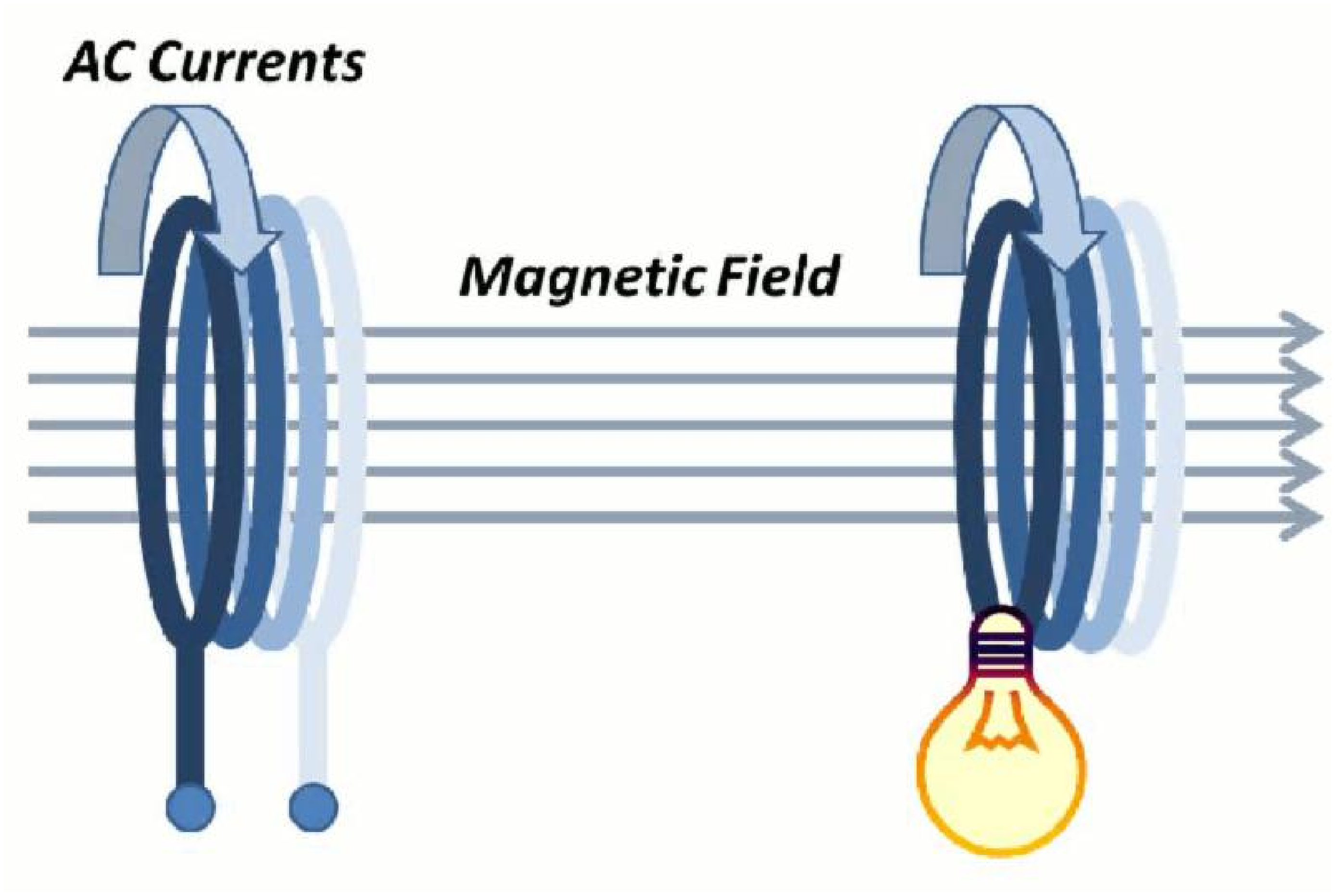
Figure 8 – Say "Hello" if you found someone else who is doing interesting work

Unfortunately, we are not providing a graphical environment access because of limited hardware capability of allowing many users at the same time. We also do not want them to be a hacking resource or used as a building machine that consumes full hardware resources. Also, the ODROID that you connect would run slowly since the devices in the maze.odroid.com can be accessed by anyone at any time. How long will maze.odroid.com run? We are expecting to offer experiences of ODROID SBCs we built, but we are also offer popular or future devices shortly after being introduced to the market. We only have 5 ODROID SBCs today, but more devices can come on demand or whenever a new ODROID SBC is published. As long as we have users who wants to have an experience, we will run more ODROID SBCs at maze.odroid.com.

We welcome hearing your suggestion or request, please visit the ODROID forum thread at https://forum.odroid.com/viewtopic.php?f=29&t=32257#p234012. For questions, comments and suggestions, please visit the original article at https://medium.com/@tobetter/odroid-bench-c5c1a10d6bec.

Wireless Charging: Adding Qi Wireless Charging to the ODROID-GO

October 1, 2018 By @Kamots ODROID-GO



I have been an electronics enthusiast ever since I was eight years old, when my grandfather used to send me small components such as LEDs, small DC motors, and 555 timers. He would include hand-drawn schematics with notes I could follow in order to build simple projects. As I got older, the projects became more complex, allowing me to learn more. I also got into computers, programming, and Amateur Radio before going to college for computer networking.

Fast forward to 2018: I now make YouTube videos about technology. I started my channel "Kamots Talks Tech" because I had purchased an ODROID-GO and got involved with the community, but I wasn't happy with the accuracy of most reviews being published about the GO. I wanted to show that this device was better than people were saying, so I made my first video about the GO: a mini-review video about the ODROID-GO. After many more videos, I'm now writing an article to be published, which I have never done before.

My project's goal is to add wireless charging capabilities to the ODROID-GO. My motivation was utilizing the Qi Wireless Charging pad I already own to make something cool for my YouTube subscribers to enjoy. I was inspired by a video from ETA PRIME where he added wireless charging to the NeoGeo Mini. It may seem like a difficult project but in reality, it's only four solder connections. If you take your time following the instructions below, you will likely be successful. However, there is risk of damaging your GO so please do not take on this project unless you are confident in your soldering and electronics skills.

Required Supplies

- ODROID-GO (https://bit.ly/2lgWvGA)

- Adafruit's Universal Qi Wireless Receiver Module (https://www.adafruit.com/product/1901)
- Very thin wire (28AWG / 0.081mm2). I took apart a flat Ethernet cable to get mine
- Electrical tape or other strong non-conductive adhesive
- Soldering iron and solder

Preparation

First, let's look at portions of the schematic for the ODROID-GO to understand where to connect the Qi coil output. Usually power for the GO is provided via the USB port but the solder pads are very small for the Micro USB connector so we need to find another way to connect on the inside of the GO.

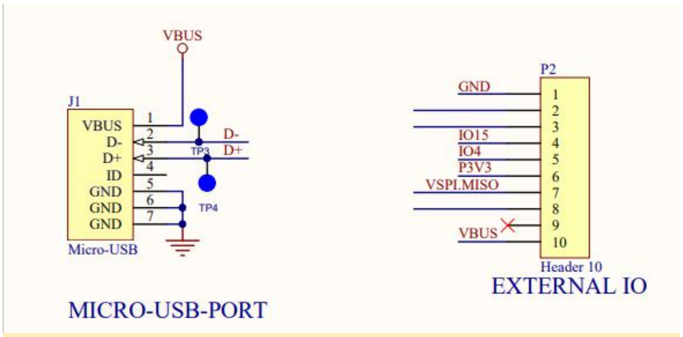


Figure 1 – Excerpts from the ODROID-GO schematic

As you can see in Figure 1, power is normally provided to the GO via the Micro USB connector to VBUS and GND. Those same power rails are available on the external IO header which has much larger pins on the GO's board. So we will solder the Qi module to pin 1 and 10 of the external IO header, which you can see in Figure 2. Standard USB power is 5 volts. The Qi module also provides 5 volts so it will charge the GO just like if you plugged it in via USB.

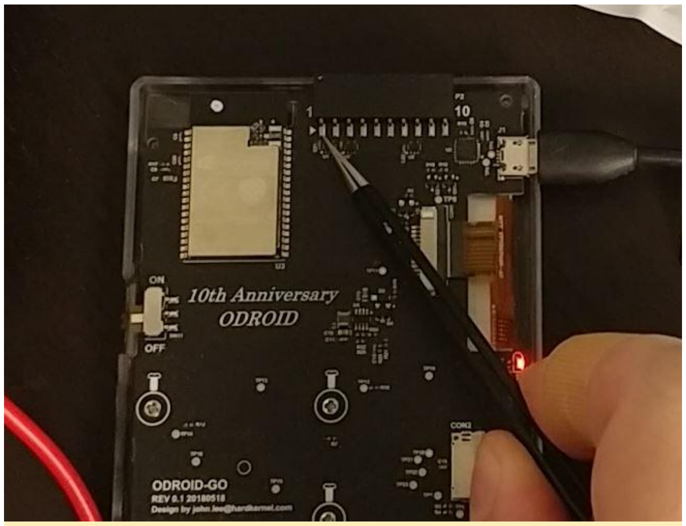


Figure 2 – External IO header

Now that we have figured out where to connect the coil, you will need to very carefully bend the two wires coming from the Qi coil into its controller board. This allows clearance for the plastic ridges on the inside of the GO's back cover where we will eventually mount the Qi module. I used tweezers to carefully bend the coil feed wires, as you can see in Figure 3. While I myself did this after I had soldered on the wires, I recommend you do it before any soldering.

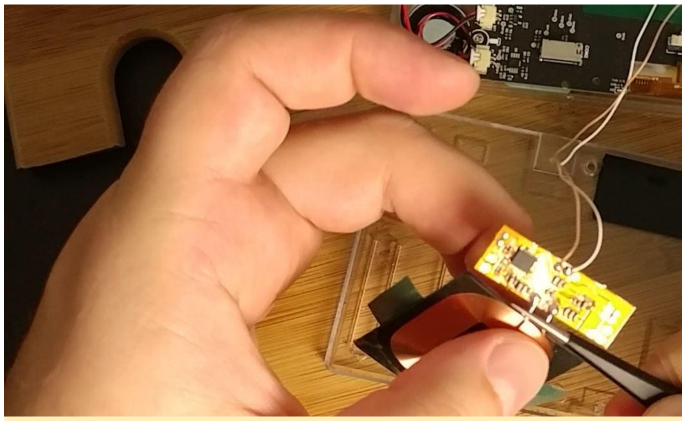


Figure 3 – Bending the Qi coil feed wires

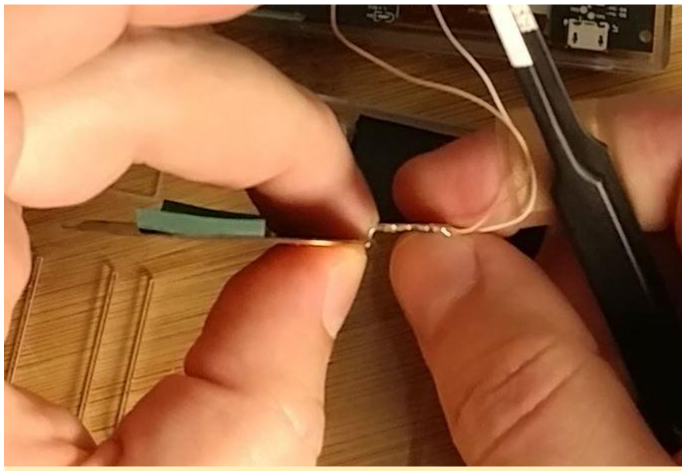


Figure 4 – Bent Qi coil wires

Making The Connections

Now begins the delicate soldering work. You will want to use the thinnest diameter solder you have. Carefully add a small amount of solder to the V4+ and GND pads on the Qi coil's controller board as seen in Figure 5. There are three pads and if you have the coil oriented the same way as in Figure 5, you will need to add solder to the center and right pad as shown.

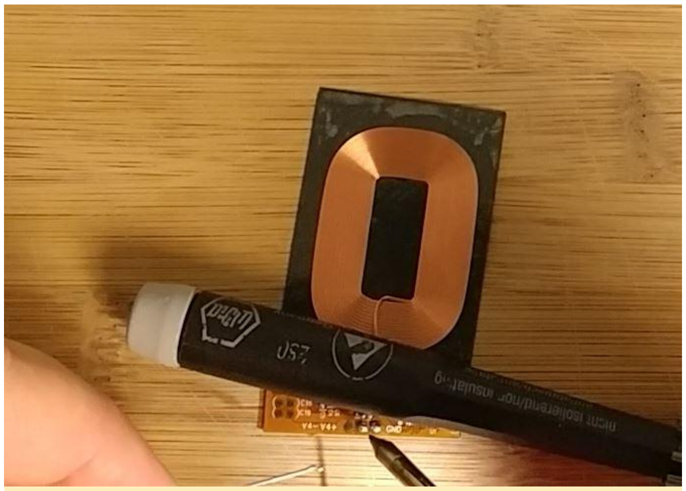


Figure 5 – Solder on the V4+ and GND pads

Once you have added solder to the pads, solder one wire to each pad by reflowing the solder you added to the pads and pressing the wire in to it. You can see me do this in Figure 6.

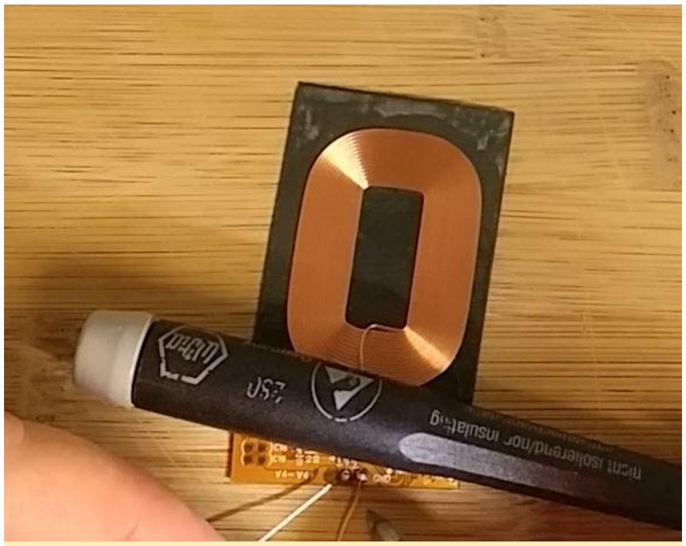


Figure 6 – Adding the wires to the Qi coil

Next, check the length of your wires to make sure they can reach the pins of the external IO header without being so long that they will be difficult to manage when you reassemble the GO. I recommend cutting where the wire reaches my fingers in Figure 7.

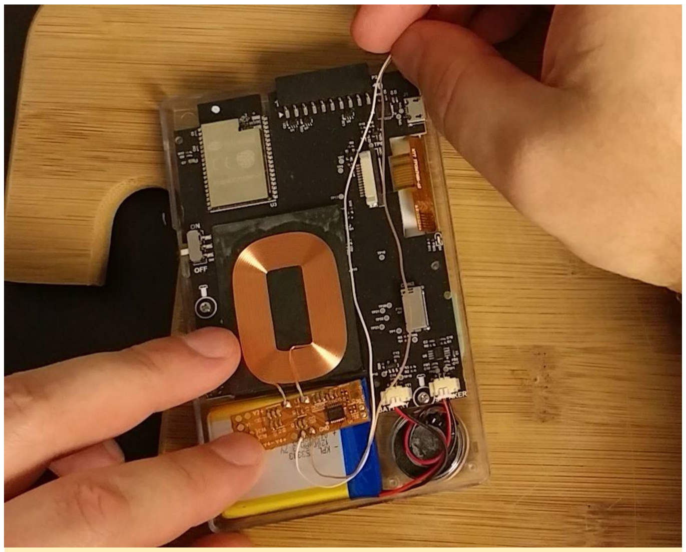


Figure 7 – Checking the wire length

For the next step you will likely need a "Helping Hands" type stand with alligator clips as seen in Figure 8, or an extra set of human hands. You will need to position the wires on the header pins, heat the header pin and wire, then add solder to make a good connection. As you may remember from the schematic in Figure 1, the wire from V4+ goes to pin 10 and the wire from GND goes to pin 1. Be very careful when soldering wire to pin 1 as the nearby component could get damaged.

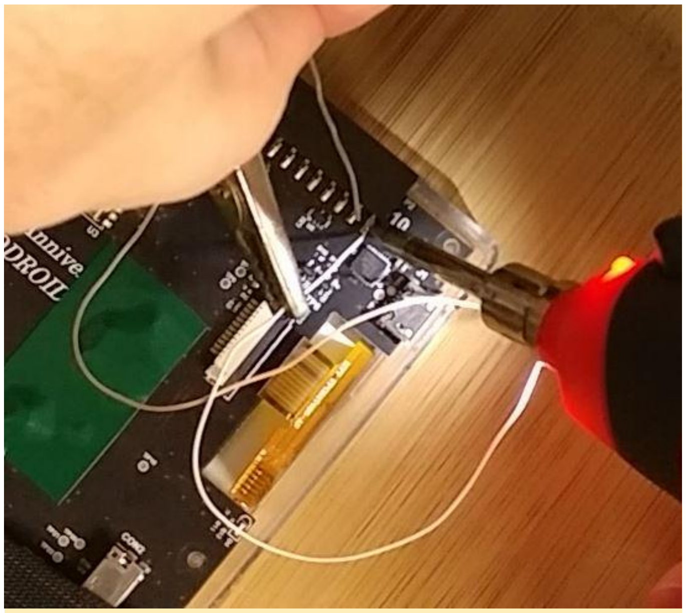


Figure 8 – Soldering to the header pins

Test your connections with a multimeter, making sure you have good connections with no resistance. Verify that you have connected the Qi coil output pins to the correct pins on the GO. Getting things backwards could damage your device and/or the Qi coil controller board.

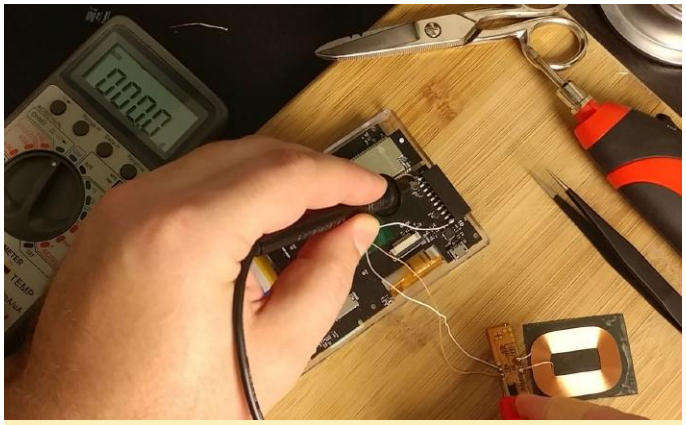


Figure 9 – Test your connections

Finishing Up

Once you are certain everything is connected properly, grab your Qi charging pad and place the Qi coil on it as seen in Figure 10. The red light on the GO should be illuminated, indicating that it is charging.

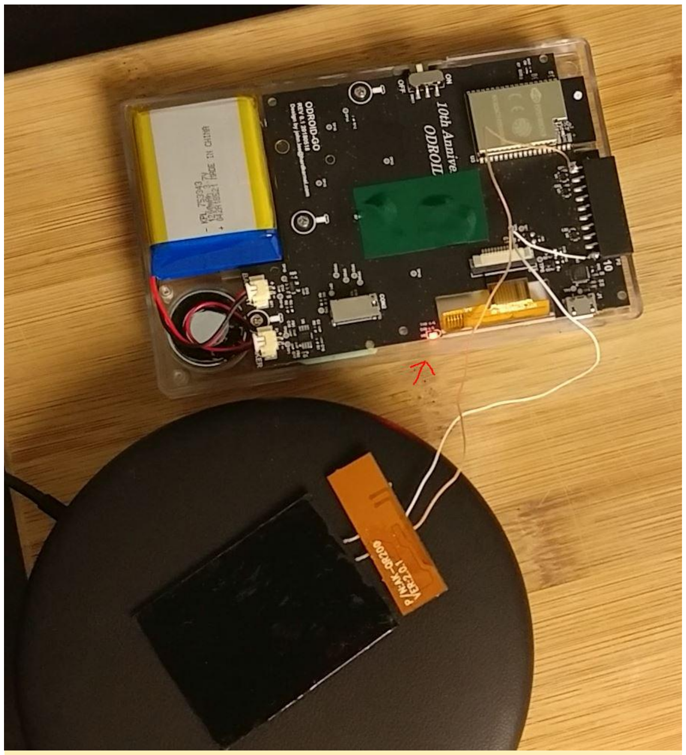


Figure 10 – Test that it charges

If it works, celebrate! The hard part is over. Now you just need to secure the Qi coil to the back case of the GO. I used electrical tape as you can see in the pictures. However, Justin Thomas left a comment on my video about this project saying I should try 3M brand VHB double-sided tape. I'll leave it up to you which one you use. The most important part is making sure that the coil lays flat against the inside of the back cover and that you don't damage it. You may also want to cut away part of the plastic ridges inside the back cover around where the Qi coil controller board sits before securing the coil. You can see how I secured mine in Figure 11.

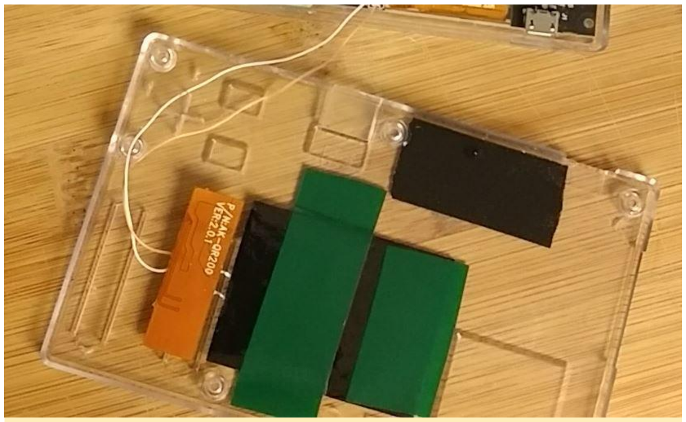


Figure 11 – Coil mounted to the back panel

Once you have the Qi coil and its board secured to the back panel, close up the GO–making sure the wires are not pinched anywhere–and put all of the screws back in place. The final result should look similar to Figure 12.

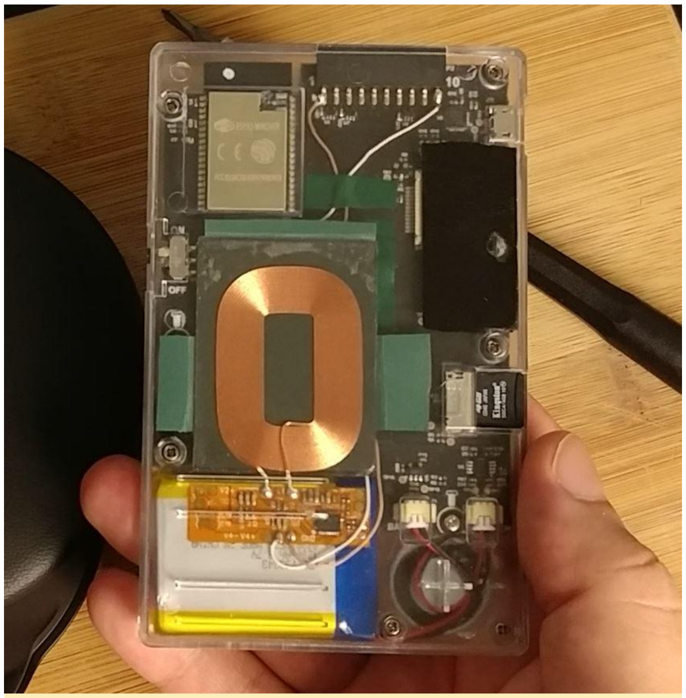


Figure 12 – ODROID-GO assembled with the Qi coil

Now you can place your GO on a Qi charging pad, making sure you center the coil on the pad, and enjoy wirelessly charging the GO. I did notice the coil gets a little warm after a while, but I don't think it is an issue. However, the coil can't tell when the GO finishes charging, so don't leave it on a charging pad overnight. You can also still use a USB cable to charge your GO if you want.

Coding Camp – Part 5: Read the ODROID-GO built-in battery voltage

October 1, 2018 By Justin Lee Tutorial, ODROID-GO

We will learn how to get the status of the battery with Arduino in this guide. The LCD will display how much battery voltage remains in volts.

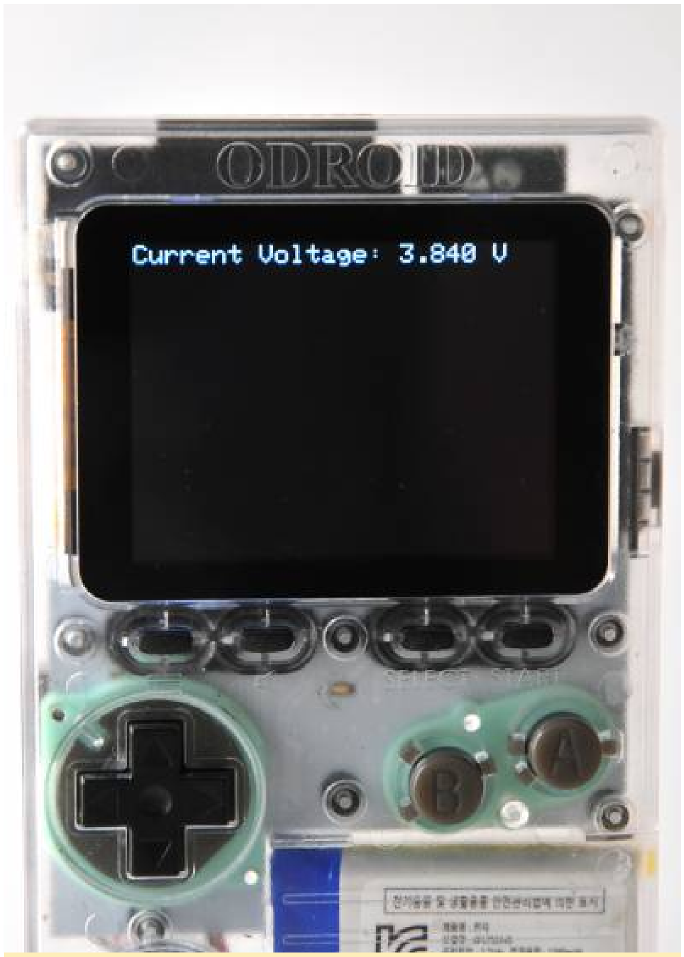


Figure 1 – The ODROID-GO has a ~3.7V battery module

We can read the battery level through one of the 12-bit SAR ADCs which are integrated in ESP32. These ADCs are:

- ADC1: 8 channels, attached to GPIOs 32-39.
- ADC2: 10 channels, attached to GPIOs 0, 2, 4, 12-15, 25-27.

There are some restrictions for an application's use of ADC2. The battery is attached to GPIO pin number 36, so we should read a value from that using ADC1. There is a library to control ADC for ESP32 and is called adc.h. In this guide, we're going to use that to read the current battery level in volts and display it on the LCD. First, prepare the code like below to display on the LCD:

```
#include

void setup() {
// put your setup code here, to run once:
GO.begin();

GO.lcd.setTextSize(2);
}

double readBatteryVoltage() {

}

void showBatteryVoltage(double voltage) {
GO.lcd.clear();
GO.lcd.setCursor(0, 0);

GO.lcd.printf("Current Voltage: %1.3lf V
", voltage);
}

void loop() {
// put your main code here, to run repeatedly:
showBatteryVoltage(readBatteryVoltage());
delay(1000);
}
```

We defined two functions in advance:

- readBatteryVoltage(): returns completely calculated voltage.
- showBatteryVoltage(): receives that voltage and print that on the screen.

Set the channel up to the proper values as we designed. Before setting it up, it's important to know that the GPIO battery voltage is divided by 2 due to the input limitation of the integrated ADC. So, if the original value coming from the battery is 3.7V, then the input value to the GPIO pin will be about 1.85V. Thus, we have to multiply the value by 2 to know the actual voltage.

We use 12 bit SAR ADC for the channel with 11 dB attenuation. We should use these rates to calculate the result as well. First of all, define that extra value as a Preprocessor macro and include necessary libraries to use ADC on ESP32:

- driver/adc.h: to get a raw ADC value about the voltage.
- esp_adc_cal.h: to calculate a correct voltage by using the AP.

Then, set the channel with the proper value by using two functions:

- adc1_config_width(): configures width of the ADC.

- adc1_config_channel_atten(): configures attenuation of the channel. GPIO pin number 36 uses channel number 1.

To get an accurate ADC voltage, calibration is needed. The ADC reference voltage is originally 1.1V by default but actually differs slightly on every ESP32 module. The manufacturer writes the calibration data in efuse:

- esp_adc_cal_characterize(): returns a characteristic of its AP as a structure.

The readBatteryVoltage() function reads an ADC value with the adc1_get_raw() function. It returns a calculated voltage as a double type value using the esp_adc_cal_raw_to_voltage() function:

```
#include
#include <driver/adc.h>
#include

#define RESISTANCE_NUM 2
#define DEFAULT_VREF 1100

static esp_adc_cal_characteristics_t
adc_chars;

void setup() {
// put your setup code here, to run once:
GO.begin();

GO.lcd.setTextSize(2);

adc1_config_width(ADC_WIDTH_BIT_12);
adc1_config_channel_atten(ADC1_CHANNEL_0,
ADC_ATTEN_DB_11);
esp_adc_cal_characterize(ADC_UNIT_1,
ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12,
DEFAULT_VREF, &adc_chars);
}

double readBatteryVoltage() {
return (double)
esp_adc_cal_raw_to_voltage(adc1_get_raw(ADC1_C
HANNEL_0), &adc_chars) * RESISTANCE_NUM /
1000;
}

void showBatteryVoltage(double voltage) {
GO.lcd.clear();
GO.lcd.setCursor(0, 0);

GO.lcd.printf("Current Voltage: %1.3lf V
", voltage);
}

void loop() {
// put your main code here, to run repeatedly:
showBatteryVoltage(readBatteryVoltage());
delay(1000);
}
Optionally, we can calculate more accurately
by multi-sampling the ADC value.
Read the value 64 times, and divide that by
the repetition count.
#include
#include <driver/adc.h>
#include

#define RESISTANCE_NUM 2
#define DEFAULT_VREF 1100
#define NO_OF_SAMPLES 64

static esp_adc_cal_characteristics_t
adc_chars;

void setup() {
// put your setup code here, to run once:
GO.begin();

GO.lcd.setTextSize(2);

adc1_config_width(ADC_WIDTH_BIT_12);
adc1_config_channel_atten(ADC1_CHANNEL_0,
ADC_ATTEN_DB_11);
esp_adc_cal_characterize(ADC_UNIT_1,
ADC_ATTEN_DB_11, ADC_WIDTH_BIT_12,
DEFAULT_VREF, &adc_chars);
}

double readBatteryVoltage() {
uint32_t adc_reading = 0;
for (int i = 0; i < NO_OF_SAMPLES; i++) {
adc_reading += adc1_get_raw((adc1_channel_t)
ADC1_CHANNEL_0);
}
adc_reading /= NO_OF_SAMPLES;

return (double)
esp_adc_cal_raw_to_voltage(adc_reading,
&adc_chars) * RESISTANCE_NUM / 1000;
}
```

```
void showBatteryVoltage(double voltage) {
GO.lcd.clear();
GO.lcd.setCursor(0, 0);

GO.lcd.printf("Current Voltage: %1.3lf V
", voltage);
}

void loop() {
// put your main code here, to run repeatedly:
showBatteryVoltage(readBatteryVoltage());
delay(1000);
}
```

Press CTRL-U to compile and upload the sketch. The current voltage of the battery will be shown on the LCD.

Completed example

The complete example is available by clicking the Files → Examples → ODROID-GO → Battery menu to import and press CTRL-U to compile/upload, as shown in Figure 2.

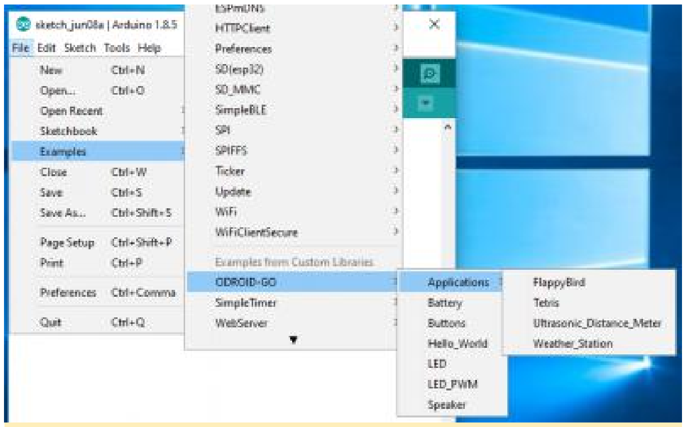


Figure 2 – Accessing the completed example

For comments, questions and suggestions, please visit the original article at https://wiki.odroid.com/odroid_go/arduino/05_battery.

Introducing NEMS Linux: Part 1 – The Nagios Enterprise Monitoring Server for ODROID Devices

October 1, 2018 By Robbie Ferguson Linux, Tutorial

Nagios® Core™–which I'll simply refer to as "Nagios" throughout this article–is a free, open source server application which monitors hosts and services that you specify, alerting you when things go bad and when they get better. I've been using Nagios for many years. If I had to hazard a guess as to when I started using it, I'd say it was around 2006.

My wife and I ran a small computer service company out of our home in those days, and in order to keep track of my customer sites and be as proactive as I could be, I had a beast of a computer in the garage keeping check on my customers' hard drive space, backup state, CPU load, and antivirus definition updates, as well as various other services.

I remember setting up that old Nagios server. The process was onerous, and all the configuration was done through the Linux terminal by opening config files in vi. One malformed syntax and Nagios would fail to start. I made it work, and if anyone has ever doubted my nerdiness, they are clearly mistaken. Ah, who am I kidding? Nobody has ever doubted my nerdiness. As the years went on and my support business customer base continued to grow, I began repurposing old hardware, installing an independent Nagios server at each client site. This worked very well.

I received my first Raspberry Pi in 2014. After it sat on the shelf for a year, I began to consider possible ways I could put it to use. I realized that the power consumption, rack space, and noise of these old Nagios servers was an incredible waste of resources. I was convinced that a single-board computer could make an excellent Nagios server, and began tinkering.

Why reinvent the wheel? Ryan Siegel's NagiosPi image was out-of-the-box ready and gaining popularity. I started using it, but quickly became dismayed by the state of the distro, which appeared to be aging and was not being updated at a pace suitable for business use. It was a wonderful starting point, but felt in some important ways like an incomplete product. I began working on my own rebuild of NagiosPi, calling it NEMS; short for "Nagios Enterprise Monitoring Server".

Figure 1: This customer Nagios server was replaced with the first-ever NEMS server in 2016

I'm a coder in my professional life. I develop server-side applications, mainly for web. Beyond building on a more current software base, the first thing I'd set out to do was build a responsive browser-based UI for NEMS, bringing all the components of NagiosPi together into a single interface. This later became the NEMS Dashboard, also known by its GitHub repository name, "nems-www."

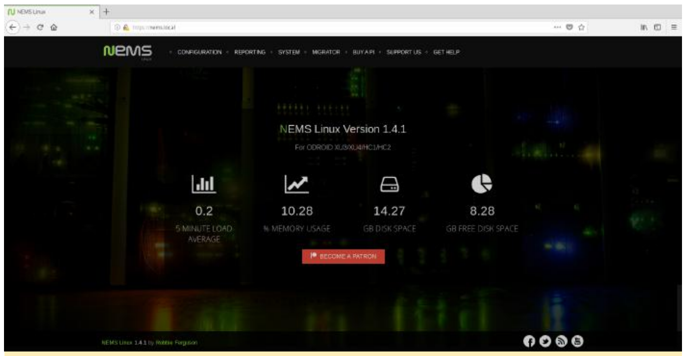


Figure 2 – NEMS Dashboard

Upon releasing NEMS to the public through my blog, Siegel himself said, "I'd love to upgrade NagiosPi, but i don't have [the] ability to make a GUI that can beat that of NEMS. I strongly feel that it has always been a necessary addition to NagiosPi and NEMS was able to deliver what is essentially an updated and improved version of NagiosPi. No reason not to start using NEMS for the time being. Nice work Robbie!" I didn't stop there, and in the wonderful spirit of community, Siegel even pitched in during the development of NEMS 1.2 in early 2017, bringing many additional Windows checks to NEMS.

With a new major release of NEMS every six months and rolling updates in between, NEMS is currently based on Debian Stretch with Nagios Core 4.3.4 at its heart. Having upgraded and maintained NEMS with a current software base, this also means things like PHP 7, current SSL certs, and a suite of customized software optimized to work on a modern server. For example, NConf (a very useful tool for configuring Nagios) stopped development years ago, so it only worked on PHP 5.3 or less. Therefore, I forked it and reworked the code to support PHP 7.0+. Of course, I made some other improvements along the way.

NEMS Linux, as it is now called (I had to find a dot-com, after all) takes the most modern network asset monitoring and does away with the old Nagios scripting requirement. The scripts are still there, it's just that you (the user) don't ever have to see them or touch them. The whole thing is controlled, configured, and monitored through your web browser, with email, Telegram, or Pushover notifications all operational out of the box. It also has a JSON API, a TV display for your server room, and more.

NEMS Linux has evolved to be what I feel is the best out-of-the-box Nagios experience available. As a Nagios user myself, this is the Nagios server I have longed for. As NEMS has continued to grow, I set out to find a more powerful platform than the Raspberry Pi. That's when I found the ODROID-XU4. Just over a year ago (September 13, 2017 to be exact) I began my quest to port NEMS Linux to the ODROID-XU4. After nearly a year of development, I am extremely proud and excited to share: NEMS Linux is now available for ODROID boards.

NEMS Features

I've already touched on the obvious interface and UX improvements that NEMS Linux brings to the Nagios experience. Those are perhaps the key points as to what makes NEMS stand out, but it's important to understand that NEMS Linux is far more than just Debian with Nagios installed. Let's look at a small selection of additional features.

NEMS Migrator

When focusing on building a distro for single board computers (SBC), I took very seriously the fact that SD cards can and likely will fail, and data can be lost. I wanted to create a way for users to be able to easily backup and restore their configuration. Out of that desire, Migrator was born.

Migrator allows you to backup your entire NEMS configuration (hosts, services, checks, system settings, etc.) via a samba share, https download, or even an optional offsite backup service. The backups can be encrypted, and only you know the decryption key. Should your device fail, you can write the image to a new SD card, restore your Migrator backup, and be up and running in under five minutes with all your settings intact. Migrator also makes it easy to transition from one platform to another. For example, having setup a NEMS Linux server on a Raspberry Pi 3, you can easily move to an ODROID-XU4 for an enormous performance boost.

Another advantage that Migrator brings to the table is an absolutely simple upgrade path: as new major releases of NEMS Linux are introduced, you can easily write the new NEMS image, import your backup, and be on the latest version of NEMS in just minutes.

UI-Based Configuration with NEMS Configurator

NEMS Configurator (NConf) is what makes browser-based Nagios configuration possible. This customized version of the old NConf configuration tool brings a sophisticated front-end to the modern architecture of NEMS. Your entire Nagios configuration is done through this interface: from adding hosts to configuring your service checks. It's all done through an intuitive browser-based system.

Now, I'll admit NConf is not the most aesthetically beautiful feature of NEMS at the moment, but it works brilliantly. And a redesign of the UI is on schedule for a future release. When that happens, the interface will be automatically updated on all existing deployments through NEMS' automatic update system. With NEMS NConf, you will never have to look at a Nagios cfg file again!

NEMS System Settings Tool (SST)

Speaking of doing away with Nagios config files, several Nagios configuration options have been moved to a tool called NEMS System Settings Tool, also referred to as NEMS SST. Items such as your SMTP server settings, domain user credentials, and other defaults are part of this interface.

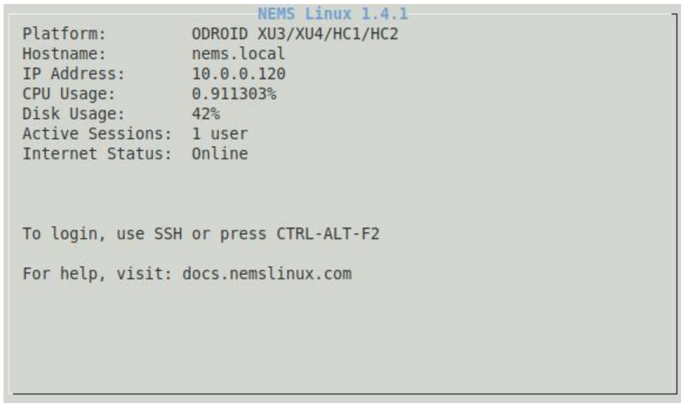
So now that you know a little about what NEMS is and how it came about, let's dive in!

Installation

All that is required in order to deploy NEMS Linux is a compatible ODROID device and a Micro SD card. At the moment, the XU3, XU4, HC1, and HC2 are all supported. More devices will be supported as soon as I have development hardware to compile and test on, so if you're reading this article several months after publication, please check the NEMS website, as your board may be supported.

Download the latest version of NEMS Linux from https://nemslinux.com and "burn" it using your favorite tool. I've really come to prefer Etcher from https://etcher.io/ but you can use whatever tool you like best. eMMC may work for you if you have a current bootloader, but it is not yet officially supported. If in doubt, please use a UHS-I or better SD card. I recommend 16GB or more, but you could get away with 8GB if that's all you have handy. You can always use NEMS Migrator to move to a bigger card down the road.

Upon booting your NEMS Linux server, your filesystem will be automatically resized to the capacity of your SD card. You can confirm NEMS is up and running by visiting https://nems. local/ in your web browser. If name resolution doesn't work, try the IP address of your NEMS device instead, which you can find in your router's DHCP table, or on a TV connected to your device's HDMI port. I do plan to introduce support for the Cloudshell 2 screen in a future update.



```
                   NEMS Linux 1.4.1
Platform:         ODROID XU3/XU4/HC1/HC2
Hostname:         nems.local
IP Address:       10.0.0.120
CPU Usage:        0.911303%
Disk Usage:       42%
Active Sessions:  1 user
Internet Status:  Online


To login, use SSH or press CTRL-ALT-F2

For help, visit: docs.nemslinux.com
```

Figure 3 – NEMS details displayed on a connected TV

Initialization

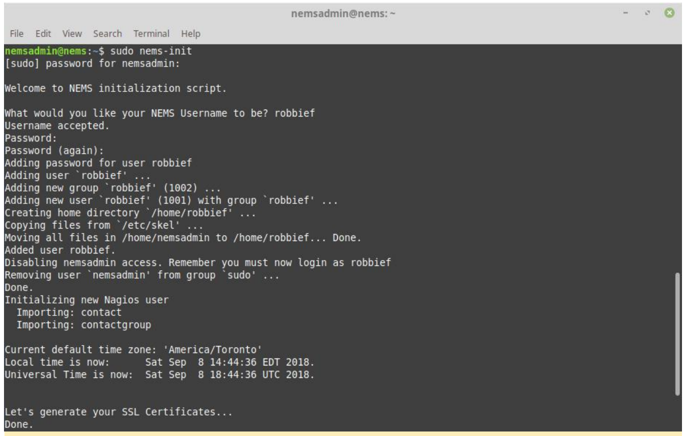
Generally speaking, the only time you'll really have to touch the Linux terminal on a NEMS server is during the initialization procedure. This task works magic in automatically configuring your entire server in just a few seconds. It generates self-signed certificates so every NEMS Linux user has a unique certificate, allows you to configure your timezone, creates your Nagios admin user, your Linux account, and so on. To initialize your NEMS Linux server, connect to your server over SSH on the default Port 22 using the following credentials:

```
Username: nemsadmin
Password: nemsadmin
```

Once connected, type:

```
$ sudo nems-init
```

You'll be asked to enter the password again. Follow the prompts. All the complicated stuff is made easy.



```
nemsadmin@nems:~$ sudo nems-init
[sudo] password for nemsadmin:

Welcome to NEMS initialization script.

What would you like your NEMS Username to be? robbief
Username accepted.
Password:
Password (again):
Adding password for user robbief
Adding user `robbief' ...
Adding new group `robbief' (1002) ...
Adding new user `robbief' (1001) with group `robbief' ...
Creating home directory `/home/robbief' ...
Copying files from `/etc/skel' ...
Moving all files in /home/nemsadmin to /home/robbief... Done.
Added user robbief.
Disabling nemsadmin access. Remember you must now login as robbief
Removing user `nemsadmin' from group `sudo' ...
Done.
Initializing new Nagios user
  Importing: contact
  Importing: contactgroup

Current default time zone: 'America/Toronto'
Local time is now:      Sat Sep  8 14:44:36 EDT 2018.
Universal Time is now:  Sat Sep  8 18:44:36 UTC 2018.


Let's generate your SSL Certificates...
Done.
```

Figure 4 – NEMS Initialization

Congratulations! Your NEMS Linux server is now online and ready to monitor your network assets.

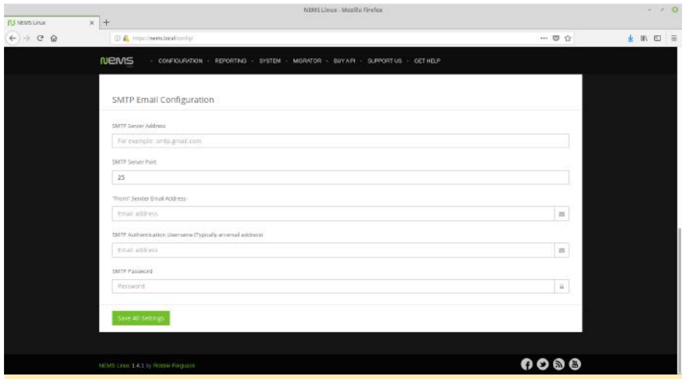
Connecting to Your NEMS Server

Now that your NEMS Linux server is up and running, you can access its entire feature set via your web browser. Simply point to https://nems.local/ which should work in most networks out of the box, but if you need to, you can alternatively use the IP address of your NEMS server.

Setting Up Email Notifications

The first thing you'll want to do on your new NEMS Linux server is configure your SMTP settings. This will allow your NEMS server to email you if a problem is detected.

Access the NEMS System Settings Tool (SST) from the Configuration menu of your NEMS dashboard. This tool does away with the need to use the traditional Nagios resource.cfg file to configure your email settings. One of the nice things about NEMS Linux is that I really don't have to go into detail about how to do this. It's so intuitive that it does not require explanation. So I'll just provide a screenshot in Figure 5.



SMTP Email Configuration

SMTP Server Address: For example: smtp.gmail.com

SMTP Server Port: 25

"From" Sender Email Address: Email address

SMTP Authentication Username (Typically an email address): Email address

SMTP Password: Password

Figure 5 – Nagios SMTP setup is easy with NEMS System Settings Tool

Configuring your SMTP server in NEMS is as simple as configuring a mail client.

Tip: If you're using Gmail as your SMTP provider, be sure to review the NEMS Documentation found at https://docs.nemslinux.com/usage/nagios-gmail-smtp for the additional steps required.

Once you are confident your SMTP settings are correctly entered, click Save All Settings and then re-connect over SSH to test your email settings with the following command (replacing youremail@yourdomain.com with your actual recipient email address):

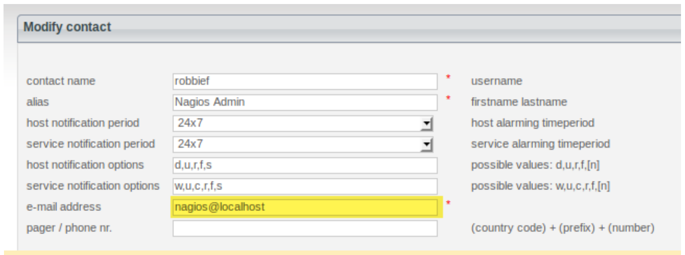
```
$ sudo nems-mailtest youremail@yourdomain.com
```

This command simply verifies your mail settings by sending a test notification. If there are any problems, it will show you on screen what the issue is, and you can make the necessary changes to your account settings in NEMS SST.

Note: NEMS currently requires your SMTP server support TLS authentication. Against my best judgement but in line with user requests, an option will be added to a future release to allow insecure authentication if required, as might be the case with an internal relay.

The Final Step to Email Notifications

The final step in setting up your email notifications is to tell NEMS where you'd like these notices to be sent. To do this, open NEMS Configurator (NConf) under Configuration, and on the left navigation press Show next to Contacts. You will see the NEMS Initialization procedure already added your user to NConf. It's robbief in my case. Press the pencil icon (Modify) to edit your contact.


Modify contact

contact name	robbief	*	username
alias	Nagios Admin	*	firstname lastname
host notification period	24x7		host alarming timeperiod
service notification period	24x7		service alarming timeperiod
host notification options	d,u,r,f,s		possible values: d,u,r,f,[n]
service notification options	w,u,c,r,f,s		possible values: w,u,c,r,f,[n]
e-mail address	nagios@localhost	*	
pager / phone nr.			(country code) + (prefix) + (number)

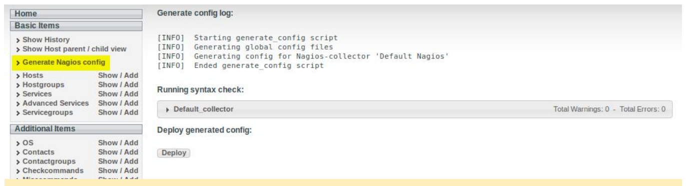
Figure 6 – Change admin email address

Under Email Address, change the value to your actual email address. Ensure your recipient email address is not the same as the one used for your SMTP settings in NEMS SST. Otherwise you may not receive the notifications as the receiving server will see you as the sender and hide the notifications from your inbox.

Once you've changed the email address, scroll down and press Submit to save the changes. Please note that your contact information will not be live until you Generate Nagios Config.

Generate Nagios Config: Make Your Changes Live

To make your changes live, press the Generate Nagios Config link on the left navigation. You should see 0 errors. If you do see errors, press the Syntax Check bar and review where you went wrong. NConf is very good at showing you where the error is so you can go back and fix it and try again.



Generate config log:

```
[INFO]  Starting generate_config script
[INFO]  Generating global config files
[INFO]  Generating config for Nagios-collector 'Default Nagios'
[INFO]  Ended generate_config script
```

Running syntax check:

Default_collector Total Warnings: 0 - Total Errors: 0

Deploy generated config:

Figure 7 – Generate Nagios Config with the NEMS Configurator

If everything checks out, press Deploy, and your admin contact email address will instantly be activated in Nagios.

Learn More

I would like to encourage you to visit the NEMS Documentation and Community Forum to learn more about how to configure and use your new NEMS Linux server, and be sure to join me again in next month's edition of ODROID Magazine as we go through our first exercise: Configuring NEMS Linux to monitor a local Linux server. NEMS has an active community forum. I check in quite regularly to provide free support to users. I also offer commercial one-on-one priority support for those needing a higher level of support.

I would like to extend thanks to mad_ady and meveric from the ODROID community for assisting me in those early days as I planned to port NEMS Linux to the ODROID platform. Of course, this release wouldn't be possible without the great work meveric has done on the Debian Stretch image, which powers NEMS Linux for ODROID-XU3/XU4/HC1/HC2. NEMS Linux is free to download and use. Its source code is available on GitHub, and the ready-to-use image can be downloaded for the ODROID at https://nemslinux.com.

About the author

Robbie Ferguson is the host of Category5 Technology TV and author of NEMS Linux. His TV show is found at https://category5.tv and his blog is https://baldnerd.com.

BASH Basics: Introduction to BASH – Part 5

October 1, 2018 By Erik Koennecke Linux

Our adventure into scripting continues with more tests, if statements, input and functions; we also do calculations from within BASH. First, let's look into filename manipulation, since this seems to be one of the most popular things needed in beginner scripts. As a bonus for making it through a lot of theory, we have another example of BASH programming in the form of a game.

Basenames

The first script people usually write are to handle lots of file manipulation. Having to do simple tasks again and again, such as conversions, gets boring real quick. It's only natural that even the most script-averse people try to automate this after a while. Usually, this involves storing the result to the same filename but with a different extension, for example converting file001.png to file001.jpg then, file002, file003 and so on.

So what can BASH do here?

If you open the bash manual, with man bash, somewhere in the nearly 6000 lines of the manual, equal to 35 parts of the BASH series you are reading now, you will find a section about 'Parameter Substitution'. I will save you from scrolling through all this and give you a helpful trick to handle long man pages:

```
$ man -P "less -p 'Parameter Expansion'" bash
```

This jumps directly to the section you are looking for, if you remember the right keywords. Let's sum up our findings here, since the man page is even drier than my article and so terse that you have to read it several times for full impact. The variable writing convention $foo is a shorthand, the full version is ${foo} which is also a short version, of ${foo:operators}. What does this mean? Let's look at some of these constructs, and then follow up with examples. There are two operators inside the curly braces, the # and the %

operator. They can be used as single characters or in pairs. The resulting combinations are

- To trim the shortest suffix from the end: ${variable%pattern}
- To trim the longest suffix from the end: ${variable%%pattern}
- To trim the shortest prefix from the beginning: ${variable#pattern}
- To trim the longest prefix from the beginning: ${variable##pattern}

The difference between "shortest" and "longest" only applies if you are using a shell wild card * in your pattern, for something like jpg, they are identical. Now we need a few examples to understand how this works. Since learning by doing is always best, you can experiment with the echo command and several shell variables yourself. If you want to follow the example, make a path /tmp/odroid with mkdir -p /tmp/odroid, followed by touch /tmp/odroid/photos.zip to generate an empty file for your experiment. It's not needed if you follow line-by-line, but for your own experiment, it's better to have something which you can manipulate. Here are the results: If we have the variable

```
foo="/tmp/odroid/photos.zip"
```

(Don't forget the quotes to make sure that you can deal with spaces and other difficult special characters) you can use

- foo=/tmp/odroid/photos.zip"; echo "${foo%/*}" to show the path /tmp/odroid. That is an easy method to get the so-called dirname of a file.
- foo=/tmp/odroid/photos.zip"; echo "${foo##*/}" yields photos.zip, the basename of the file.
- foo=photos.zip"; file="${foo%%.*}" gives photos, and foo=photos.zip"; echo "${foo#*/}" gives zip.

One other useful application of curly brackets is curly bracket expansion. For example, if you want to make a backup file of something, but are too lazy to type out the full path and filename for source and destination: cp /path/to/your/file.doc{,.bak} expands to cp /path/to/your/file.doc /path/to/your/file.doc.bak and makes a backup file of your document in the same directory. ${#foo} gives the number of characters of the variable. foo=4-letter-word; echo ${#foo} is 13.

Tests and if statements

To look into tests and if statements in more detail, we want to have an example script with all the basic blocks which can occur, as shown below:

```
#!/bin/bash  #Start of every BASH script
# Basic if statement # Comment, script purpose
if [ $1 -gt 9 ]  #test condition and if
statement
then
    echo $1, that's unusual for a lucky
number.  # if statement true
    date  # if statement true
fi
Pwd  #always executed
then
    date
else
    pwd
...
```

The snippet above could be part of such a script. There's also if – elif – else or short for 'else if' statements, case statements, or the option of multiple tests with AND or OR conditions with the operators && for AND and || for OR, but this is beyond the scope of this article. The square brackets, [ ], in the if statement above is a reference to the command test. All the operators that test allows may be used here as well. Look up the man page for test to see all the possible operators; some of the more common ones are listed below.

- ! EXPRESSION – The EXPRESSION is false.
- -n STRING – The length of STRING is greater than zero.
- -z STRING – The lengh of STRING is zero (= it is empty).
- STRING1 = STRING2 – STRING1 is equal to STRING2
- STRING1 != STRING2 – STRING1 is not equal to STRING2
- INTEGER1 -eq INTEGER2 – INTEGER1 is numerically equal to INTEGER2
- INTEGER1 -gt INTEGER2 – INTEGER1 is numerically greater than INTEGER2
- INTEGER1 -lt INTEGER2 – INTEGER1 is numerically less than INTEGER2

- -d FILE – FILE exists and is a directory.
- -e FILE – FILE exists.
- -r FILE – FILE exists and the read permission is granted.
- -s FILE – FILE exists and its size is greater than zero (= it is not empty).
- -w FILE – FILE exists and the write permission is granted.
- -x FILE – FILE exists and the execute permission is granted.

Input, more variables

For BASH input, if you want to have an interactive script, you can use 'read' to read the variable from the user input. Simple script example, greeting.sh:

```
echo Who are you?
read name
echo It's nice to meet you, $name.
```

Since the quote has a special meaning for BASH, it gets escaped via the backslash. read -p and read -sp can be used to prompt and be silent, which means not to echo the typed input:

```
read -p 'Username: ' user
read -sp 'Password: ' pass
```

stores the username in $user and the password in $pass, without showing it. If the user enters several words when prompted, read varx vary varz would store three words in the three different variables. If more than three words were given, the last variable stores the remaining input. Here are some special variables which you can use in your scripts:

- $0 – The name of the Bash script.
- $1 – $9 – The first 9 arguments to the Bash script.
- $# – Number of arguments passed to the Bash script.
- $@ – All the arguments supplied to the Bash script.
- $? – The exit status of the most recently run process.
- $$ – The process ID of the current script.
- $USER – The username of the user running the script.
- $HOSTNAME – The hostname of the machine the script is running on.
- $SECONDS – The number of seconds since the script was started.
- $RANDOM – Returns a different random number each time is it referred to.
- $LINENO – Returns the current line number in the Bash script.

These should be pretty simple, with $0 needing a little more detail: With this, you can have your script act differently depending on which name you used for it, for example, compressing and decompressing files. A full list of pre-defined variables which you can use is available if you type 'env' on the command line.

With this, there are now 3 methods of passing data to a BASH script, and which method is best depends on the situation:

- Command line arguments, like ourscript.sh foo bar baz gives $1=foo, $2=bar and $3=baz.
- Read input during script execution, see above.
- Data that has been redirected into the script via stdin. This is more for experienced scripters, basically, it is same as piping results from another script or function into our script with '|'.

We used quotes now sometimes without explaining what they are good for. Since BASH uses a space to separate items, device=Odroid XU4; echo $device gives an error message that 'XU4' cannot be found, because the variable content is just 'Odroid' and BASH tries to execute 'XU4'. If you want a variable with spaces, you need to use quotes. Text in quotes indicates to BASH that the contents should be considered as a single item. You may use single quotes ' or double quotes ". Depending on what you want to do, either one is important:

- Single quotes will treat every character literally. echo 'I am $USER' prints I am $USER.
- Double quotes will allow you to do substitution (that is, include variables which get evaluated). echo "I am $USER" prints I am odroid.

You can also use command substitution to have a variable filled with the evaluation of command(s) – foo=$(ls /etc | wc -l); echo There are $foo entries in /etc. shows how many configuration entries are in your /etc directory.

If you want to have variables available for other scripts, you need to export them. export foo makes $foo available for following scripts, but if they change

$foo, this has no impact on the exporting script. Think of it as making a copy and handing it out.

Calculations

Let’s cover calculations in BASH briefly. It’s enough to know that $((…)) evaluates the term in the double brackets. So, if you want to know quickly how many seconds a year has, type:

```
$ echo $((60*60*24*365))
```

This should return 31536000. Only integer values are allowed, though. This is true for input and output – fractional output gets dropped by BASH. echo $((4/5)) gives 0, and only echo $((5/5)) would give you 1 as a result.

Practical BASH application

Another game as a reward for making it through all the theoretical BASH aspects, in around 350 lines of BASH script, is the game 2048. You can play it with the cursor keys, link to the script in the references. download it with wget.

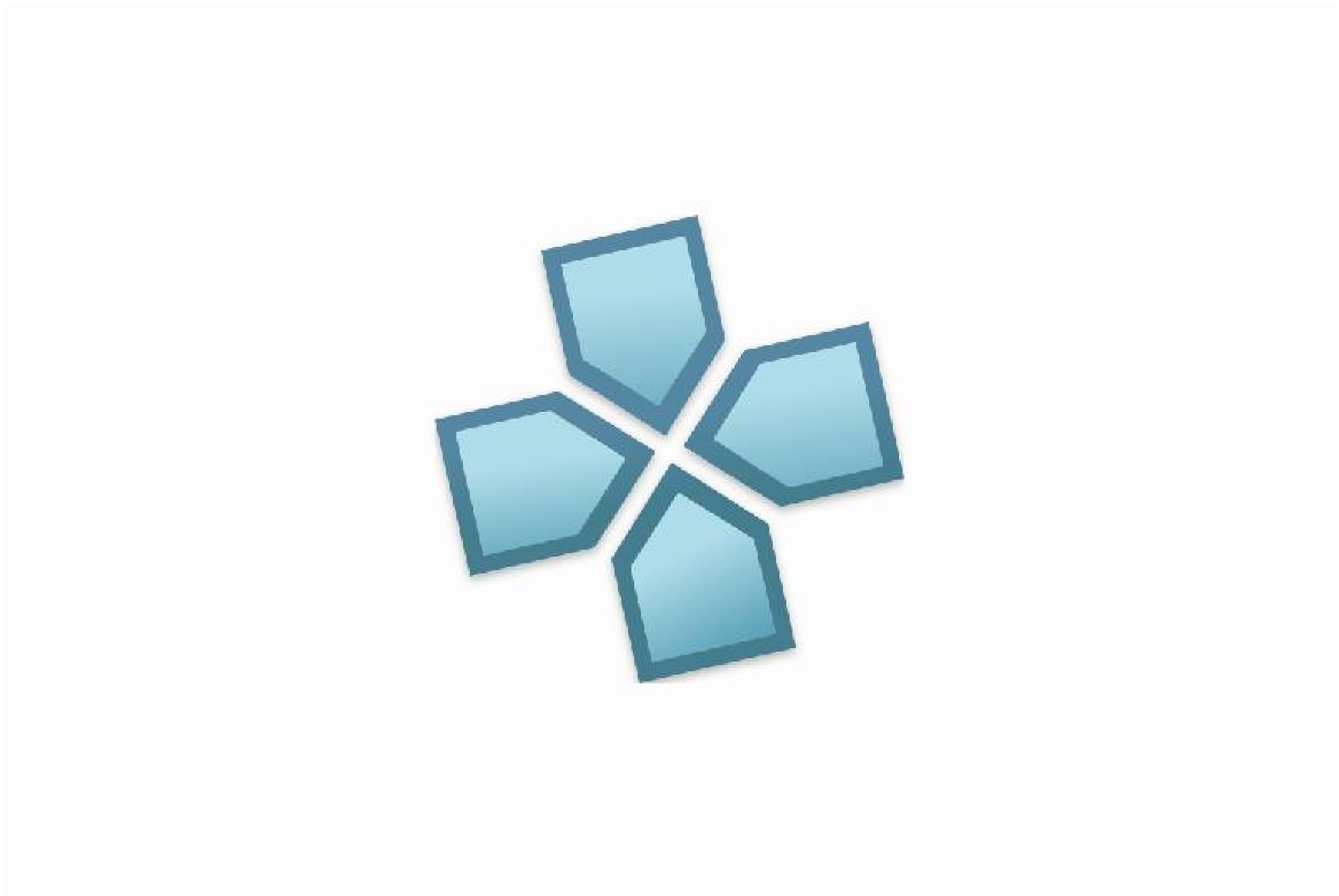
This was a lot of exciting information, in the next part we take a break from the theory and look at an interesting command line applications and useful BASH scripts. Also a little more (only a little!) about scripting with loops and functions.

References

http://linuxg.net/curly-brackets-expansion-in-bash-with-5-practical-examples/
https://raw.githubusercontent.com/mydzor/bash2048/master/bash2048.sh

GBM Video Driver

October 1, 2018 By @AreaScout Gaming, ODROID-XU4, Tutorial



This is a guide to install GBM-enabled user space drivers and build retro gaming emulators. First, we will showcase the use of the Mali GBM enabled userspace library. We will show how well it works, and for what it can be used. We will use the ODROID-VU5A display to showcase the samples.

We will install an Ubuntu 18.04 minimal image for XU4 and prepare the use of the GBM Mali Driver for it. As a result, PPSSPP will run to emulate the games well without any tearing. The video showcasing GoW – Chains of Olympus on ODROID-XU4 can be found at https://youtu.be/QegJlwflkZk.

Showcasing GoW - Chains of Olympus on ODROID XU4 / VU5A, with Mali GB...
GOD OF WAR
CHAINS OF OLYMPUS

Install the image

Download the image: ubuntu-18.04-4.14-minimal-odroid-xu4-20180531.img.xz from https://odroid.in/ubuntu_18.04lts/. Extract it and write the image to the boot media, using the the online guide described at Hardkernel's wiki.

Boot the ODROID-XU4 using the boot media and use the login credentials:

Username: root Password: odroid

Update the system software using the commands:

```
$ apt update && apt full-upgrade -y
```

Install dependent components:

```
$ apt-get install mali-x11 libx11-dev libsm-
dev libxext-dev git cmake mercurial libudev-
dev libdrm-dev zlib1g-dev pkg-config
libasound2-dev alsa-utils htop bc ifupdown2
net-tools libssl1.0-dev mlocate bluez
libfreetype6-dev libgbm-dev
```

Add the odroid user and add the userid to groups:

```
$ adduser odroid usermod -aG
Sudo,adm,audio,operator,input,video,tty,staff,
games,users, plugdev,netdev,bluetooth,disk
odroid
```

Login with the newly created user:

```
$ su - odroid
```

Enable color prompt:

```
$ sed -i '1iforce_color_prompt=yes' ~/.bashrc
$ su odroid
```

Turn the mali-x11 display driver in the GBM enabled one. This is normally relatively easy to do by only updating eglplatform.h which is multi platform (ie. fbdev x11 and GBM). However, we need to update the other headers also.

```
$ cd /usr/include/EGL
$ sudo rm eglplatform.h
$ sudo wget
https://www.khronos.org/registry/EGL/api/EGL/e
glplatform.h
$ cd /usr/include/GLES2
$ sudo rm *
$ sudo wget
https://www.khronos.org/registry/OpenGL/api/GL
ES2/gl2platform.h
$ sudo wget
https://www.khronos.org/registry/OpenGL/api/GL
ES2/gl2.h
$ sudo wget
https://www.khronos.org/registry/OpenGL/api/GL
ES2/gl2ext.h
$ cd /usr/include/GLES3
$ sudo rm *
$ sudo wget
https://www.khronos.org/registry/OpenGL/api/GL
ES3/gl3.h
$ sudo wget
https://www.khronos.org/registry/OpenGL/api/GL
ES3/gl31.h
$ sudo wget
https://www.khronos.org/registry/OpenGL/api/GL
ES3/gl32.h
$ sudo wget
https://www.khronos.org/registry/OpenGL/api/GL
ES3/gl3platform.h
```

Now get the new driver binary:

```
$ cd ~
$ wget http://deb.odroid.in/bigmali.tar
$ tar xf bigmali.tar libmali.so
$ sudo mv libmali.so /usr/lib/arm-linux-
gnueabihf/mali-egl/.
```

Add a missing symbolic link:

```
$ sudo ln -s /usr/lib/arm-linux-
gnueabihf/mali-egl/libmali.so
/usr/lib/arm-linux-gnueabihf/libGLESv1_CM.so.1
```

Add a new directory, open up vi editor for /usr/local/lib/pkgconfig/gbm.pc file:

```
$ sudo mkdir /usr/local/lib/pkgconfig
$ sudo vi /usr/local/lib/pkgconfig/gbm.pc
```

Add the following section into the new file:

```
prefix=/usr/local
exec_prefix=${prefix}
includedir=${prefix}/include
libdir=${exec_prefix}/lib
Name: libgbm
Description: A small gbm implementation
Version: 19.0.0
Cflags: -I${includedir}
Libs: -L${libdir} -lgbm
```

Save and close vi. Add symbolic links for libgbm:

```
$ cd /usr/local/lib/
$ sudo ln -s /usr/lib/arm-linux-
gnueabihf/mali-egl/libmali.so libgbm.so
```

```
$ sudo ln -s libgbm.so libgbm.so.19
$ sudo ln -s libgbm.so.19 libgbm.so.19.0.0
$ sudo ldconfig
```

Fetch gbm.h and delete the one from mesa:

```
$ cd /usr/include
$ sudo rm gbm.h
$ sudo wget -O gbm.h
https://pastebin.com/raw/5QUd011t
```

Download SDL2, build and install it:

```
$ cd ~
$ hg clone http://hg.libsdl.org/SDL SDL2
$ cd SDL2
$ ./configure --disable-video-opengl --enable-
video-kmsdrm
```

After the configure step is finished you should see this line:

```
Video drivers : dummy x11(dynamic)
kmsdrm(dynamic) opengl_es1 opengl_es2 vulkan
```

Now we have to edit SDL_config.h to permanently dlopen our libgbm.so.19 instead of the libgbm.so.1 from Mesa. Build and install it:

```
$ sed -i -e 's/libgbm.so.1/libgbm.so.19/g'
include/SDL_config.h
$ make -j7
$ sudo make install
```

After the build completes we can test:

$ cd test $ ./configure $ make -j7 $ ./testgles2

You should see a spinning cube, wait some seconds and quit with ESC key. If you are using the VU5A, you will notice something like this which indicates it is too slow:

```
INFO: 56.89 frames per second
```

It should reach 60 fps, if not it is a real problem for emulators. If they try to just hold 60 fps and they cannot, then it will really slow down things a lot. We can observe further using PPSSPP game intro videos.

Building the Kernel for >60fps on VU5A

We have to patch the kernel to get more than 56fps processing. Until we publish the needed changes in the mainline kernel, we will have to do it on our own.

The pixel clock and probably some H or V sync timing values of HDMI PHY config are not right for the VU5A Display. Details can be found at https://goo.gl/CVJYS6. Hardkernel took the closest one but they choose it to be on the lower side of 60fps. For emulation it is better to choose the higher side of 60fps, so at the end we will have a fixed frame rate of 64fps.

Ensure you have enough space on the system storage, and then Fetch the kernel source. We will select the HDMI PHI config from the next higher pixel clock, as what is really inside this HDMI PHY config is unknown to the public. We have asked several developers but no one could give me an answer, only in kernel 3.10 there is some small code to change this 32byte long config. So only some few bytes are known, but they do not change the refresh rate.

```
$ git clone
https://github.com/hardkernel/linux.git
$ cd linux
$ wget -O VU5A.patch
https://pastebin.com/raw/aWEYArWL
$ patch -p1 < VU5A.patch

$ make odroidxu4_defconfig
$ make -j7
$ sudo cp arch/arm/boot/zImage /media/boot/.
$ sudo cp arch/arm/boot/dts/exynos5422-
odroidxu4.dtb /media/boot/.
$ sudo make modules_install
```

Reboot and test the fps value again:

```
$ cd SDL2/test
$ ./testgles2
```

Now you should see something like this:

```
INFO: 64.01 frames per second
```

Now we will build PPSSPP and then RetroArch with GBM KMSDRM backend, including some libretro's.

Building PPSSPP

PPSSPP is a PSP emulator for Android.

Fetch the source and apply a needed patch:

```
$ cd ~
$ git clone --recursive
https://github.com/hrydgard/ppsspp.git
```

FFmpeg needs to be built before the PPSSPP binary is built. The pre-built binaries are all for soft floating points and we need hardfp:

```
$ ./linux_armhf.sh
$ cd ..
```

Before we can start to compile we have to turn our copy of /usr/include/GLES2/gl2ext.h into a vendor specific one by disabling the use of GL_EXT_buffer_storage. A backup file is created gl2ext.h.back. Our Mali library does not include/export that function so we cannot define it.

```
$ sudo sed -i.bak '/^#ifndef
GL_EXT_buffer_storage$/,/^$/d'
/usr/include/GLES2/gl2ext.h
```

You may want to set the use of only 4 cores in FFmpeg for tinkering and experimenting. I have observed that FFmpeg with threading does not work out very well when all cores are chosen with HMP (switching higher demanding tasks from LITTLE to the BIG CPU's). For this you can edit the file Core/HW/MediaEngine.cpp at line number 475, and change to use only 4 cores (better for switching from 4 LITTLE to 4 BIG instead using all 8 cores).

However, this was observed in Moonlight with game streaming 1080p video files. PPSSPP video files are not that big and maybe therefore not so CPU intensive so that may only impact very little to nothing:

```
av_dict_set(&opt, "threads", "4", 0);
```

Now, we need to generate the Makefile.

```
$ cmake -DUSING_EGL=OFF -DUSING_GLES2=ON -
DUSE_FFMPEG=YES -DUSE_SYSTEM_FFMPEG=NO .
```

Start compiling the binary:

```
$ make -j7
```

If you are using the VU5A you will now have touchscreen capabilities in menu and you can also enable 'On-Screen touch controls' if you want. You can watch the video at https://youtu.be/QegJlwflkZk?t=374. You can now brand your emulated PPSSPP to an unique region by generating a locale for it. In my case, I used de_AT:

```
$ sudo locale-gen de_AT.UTF-8
$ sudo update-locale LANG=de_AT.UTF-8
```

Some games may use it for In-Game Language. The download at https://goo.gl/BhHLxm contains my settings for GoW, including some texture replacements for Star Wars, The Clone Wars and Star Wars, and The Force Unleashed. These games are playable

The PPSSPP config directory would look like this:

```
odroid@odroid:~$ tree -d .config/ppsspp/
.config/ppsspp/
└── PSP
├── PPSSPP_STATE
├── SAVEDATA
│  ├── ULES01284SAVE00
│  └── ULES01376SYSDATA
├── SYSTEM
│  └── CACHE
└── TEXTURES
├── ULES00981
└── ULES01284


10 directories
```

Build and configure RetroArch

Showcasing some mame games on ODROID XU4 / VU5A, with Mali GBM en...
MARS
PRESS START
CREDIT

A video of this can be viewed at https://youtu.be/6Ewgov7_TXM. In addition to the fetching RetroArch, we need a small patch which prevents us to have the menu with only a black background:

```
$ git clone
https://github.com/libretro/RetroArch.git
$ cd RetroArch
$ wget -O retro.patch
https://pastebin.com/raw/1SCeb8EG
$ patch -p1 < retro.patch
$ ./configure --enable-opengles3 --enable-
opengles --enable-neon  --enable-floathard --
enable-freetype
$ make -j7
$ sudo make install
$ retroarch
```

Apply some useful settings. This is a suggestion to you to setup RetroArch, after it is installed you do not have to follow that tutorial anymore. Update the Assets (Icons, background pictures and stuff). You can find needed information at the menus below:

```
MainMenu -> Online Updater -> Update Assets
```

We suggest you also update these packages:

```
Core Info Files, Joypad Profiles, Database,
GLSL Shaders
```

Also, you could use the Core Updater to get some emulators. Next, enable Advanced Settings:

```
Settings -> User Interface -> Show Advanced
Settings -> ON
```

Enable Threaded Video, and it will boost up emulation a lot:

```
Settings -> Video -> Threaded Video -> ON
```

Enable FPS counter, it is helpful to see how fast the emulation runs, especially when you setup things:

```
Settings -> Onscreen Display -> Onscreen
Notifications -> Display Framerate -> ON

Settings -> Onscreen Display -> Onscreen
Notifications -> Show frame count on FPS
Display -> OFF

Settings -> Driver -> Audio Driver ->
alsathread
```

and if you are using VU5A:

```
Settings -> Onscreen Display -> Onscreen
Notifications -> Notification size -> 18

Settings -> Onscreen Display -> Onscreen
Notifications -> Notification X position ->
0.010

Settings -> Onscreen Display -> Onscreen
Notifications -> Notification Y position ->
0.010
```

If you already have games installed on your ODROID-XU4, scan for them using Import Content -> Scan Directory and select the root game folder to let RetroArch scan for your games. They will appear on the right side of the menu after some time.

Build and configure Versatile Commodore Emulator

A video of this can be viewed at: https://youtu.be/ItkppnXWd9U

First let us install Vice libretro and then mame libretro.

Install the prerequisite:

```
$ sudo apt-get install bison
```

Download the source and apply the no-border patch for VIC-II Commodore machines if you want. This will remove the border of C64 and C128 machine models, the games are way better to view without it. This is a quick way to do it. A better approach would be to add it to the libretro config.

If a game draws inside those borders it will not work and the system will probably segfault, but not a lot of games are drawing into the border.

```
$ git clone https://github.com/libretro/vice-libretro.git
$ cd vice-libretro
$ wget -O noborder.patch
https://pastebin.com/raw/VwtSDj50
$ patch -p1 < noborder.patch
```

Start to build a Commodore machine of your choice. The valid machine types are:

```
x128, x64, x64sc, x64dtv, x64scpu, xplus4,
xvic, xcbm5x0, xcbm2, xpet
```

You must add a EMUTYPE variable followed by the machine type you want to build. If you do not add this variable, then x64(C64) is the default machine type.

```
$ make EMUTYPE=x64 -f Makefile.libretro -j7
```

If you want to build more then one machine type, do not forget to run the clean command on the project, otherwise the core will not work:

```
$ make EMUTYPE=x64 -f Makefile.libretro -j7
clean
```

RetroArch config

Copy the binary into RetroArch core folder:

```
$ cp vice_x64_libretro.so
~/.config/retroarch/cores/.
```

Start RetroArch and select the vice core – either start the core without game or with it. Hit the Guide button on your game controller or F1 on the keyboard and scroll down to Options enter it and disable DriveTrueEmulaton->OFF. It will take a very long time to load a game and set Controller0Type to joystick.

You can also enable an Aspect Ratio of 16:10. It is a good compromise between 4:3 and 16:9

```
Settings -> Video -> Aspect Ratio -> 16:10
```

With the Start button you can activate the nuklear GUI settings (Select button has to pressed once to activate mouse) from there you can choose the C64 Joyport, machine cpu, sid type and more. The Onscreen keyboard is activated with the x button (Xbox layout)

Build and configure Reicast core – a Dreamcast emulator

The video link can be found at https://youtu.be/j0jEUcQx-vM. Download the source and apply a patch as usual:

```
$ cd ~
$ git clone
https://github.com/libretro/reicast-
emulator.git
$ wget -O xu4.patch
https://pastebin.com/raw/pfVjnVs3
$ patch -p1 < xu4.patch
$ platform=odroid ARCH=arm make -j7
$ strip reicast_libretro.so
```

```
$ cp reicast_libretro.so
~/.config/retroarch/cores/.
```

First you need some bios for NAOMI and Dreamcast. Some good links to obtain and use them are https://goo.gl/a5JbMT and https://docs.libretro.com/library/reicast/. If you want to know the md5 checksum of the NAOMI bios file you can take a look into the core info file:

```
/home/odroid/.config/retroarch/cores/reicast_l
ibretro.info
```

Inside a game open RetroArch menu and go to core options find the following settings and change them to the following values, which are the most important to get a decent speed:

```
reicast_framerate = "normal"
reicast_enable_rttb = "enabled"
reicast_threaded_rendering = "enabled"
```

For comments, questions, and suggestions, please visit the original post at https://forum.odroid.com/viewtopic.php?f=98&t=32173.

Coding Camp – Part 6: Generate sound from the ODROID-GO speaker

October 1, 2018 By Justin Lee Tinkering, Tutorial, ODROID-GO

Let us learn how to use the DAC output as a sound tone generator. The odroid_go.h library and its GO instance has a Speaker instance for using the speaker easily. So, you can play something with GO.Speaker. Some of the GO.Speaker functions are:

- setVolume(): to set volume level. The given parameter can be 0 to 11(mute).
- playMusic(): to play music which is written in 8 bit integers. The given parameter is a proper sample rate for playing music.
- beep(): to play a simple beep sound.
- tone(): to play a simple beep sound with two parameters of a frequency and a duration in millisecond. You can omit the duration argument.

We're going to write code that plays a sound when a button is pressed. We will use the A, B, and Start buttons and make these buttons play a sound that differs from each other. To learn about how the buttons are used, please refer to the Buttons example. We're also going to show which button is pressed on the LCD.

To learn about how the LCD is used, please refer to the Hello World example. We can write source code as shown below. Initialize the board by calling the GO.begin() function and put the code in the loop() function that activates when the button wasPressed().

```
#include

void setup() {
// put your setup code here, to run once:
GO.begin();

GO.lcd.printf("ODROID-GO speaker test:
");
GO.Speaker.setVolume(8);
GO.Speaker.playMusic(m5stack_startup_music,
25000);
}

void loop() {
// put your main code here, to run repeatedly:

if(GO.BtnA.wasPressed()) {
GO.lcd.printf("wasPressed: A
");
GO.Speaker.beep();
}

if(GO.BtnB.wasPressed()) {
GO.lcd.printf("wasPressed: B
");
GO.Speaker.tone(3000, 200);
}

if(GO.BtnStart.wasPressed()) {
GO.lcd.printf("wasPressed: Start
");
GO.Speaker.playMusic(m5stack_startup_music,
25000);
}

GO.update();
}
```

Press CTRL-U to compile and upload the sketch, and press A, B or Start button to play a sound.

Completed example

The complete example is available by clicking the Files → Examples → ODROID-GO → Speaker menu to import and press CTRL-U to compile/upload, as shown in Figure 1.

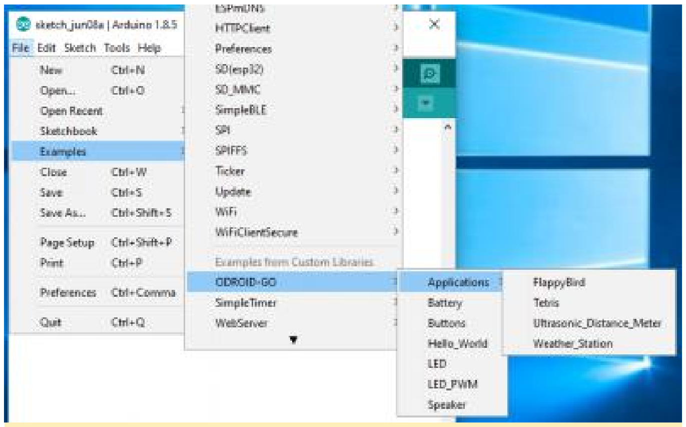
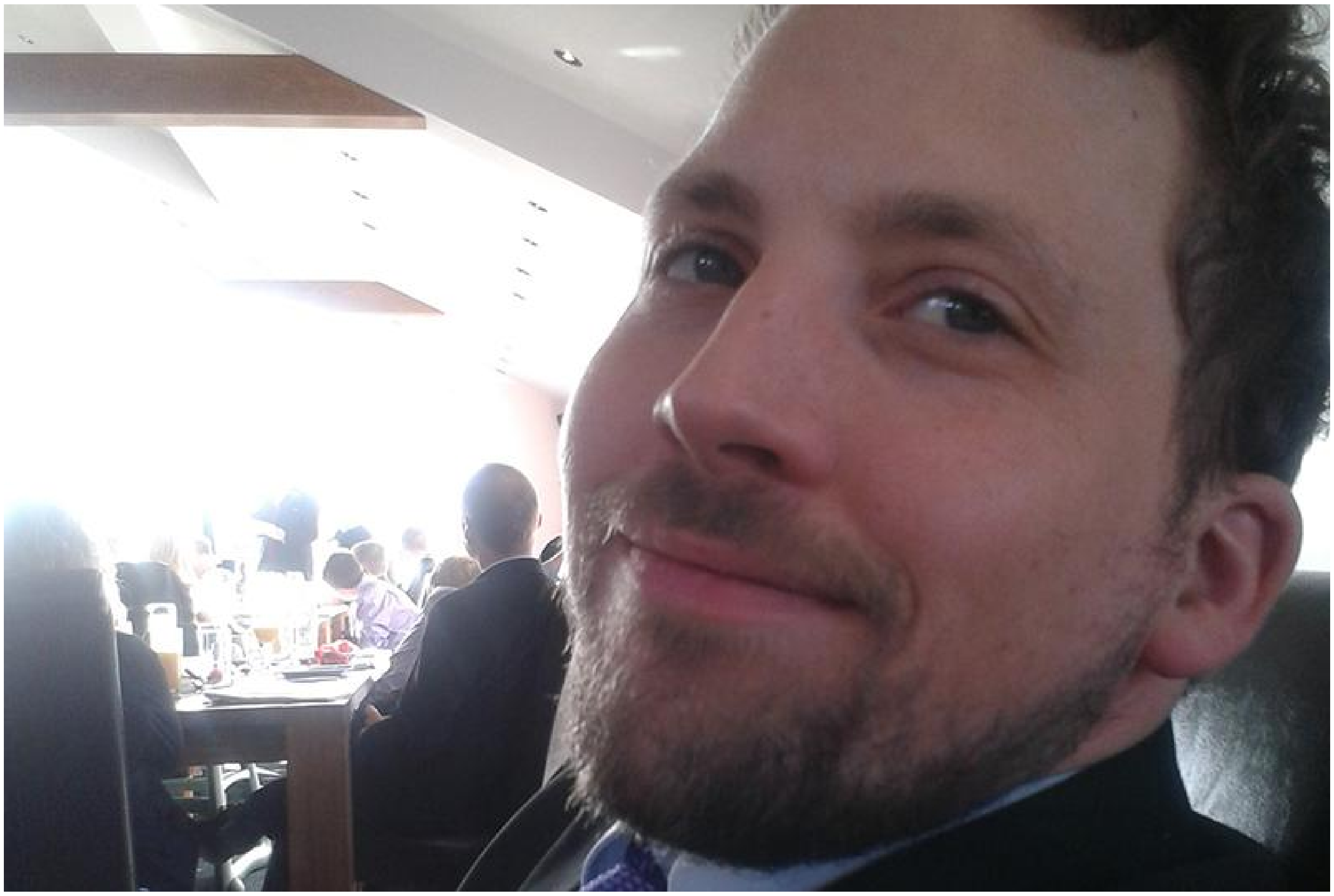


Figure 1 – Accessing the completed example

For comments, questions, and suggestions, please visit the original article at https://wiki.odroid.com/odroid_go/arduino/06_speaker.

Meet An ODROIDian: David Knight

October 1, 2018 By Rob Roy Meet an ODROIDian



Please tell us a little about yourself. I live in Newcastle, UK, working as an optometrist working in the refractive surgery sector. My day job involves managing patients who have had cataract or laser eye surgery. I was very studious and introverted in my youth, always having my head in books. I was pretty good at maths, science and chess – in the 1980's computing as a subject was still fairly new and I was more interested in biology at the time. I remember working in a spectacle glazing factory over the summer one year and shortly after decided to do a degree in Optometry. After working in private practice for a decade I needed a new challenge. That's when I turned to refractive surgery and have not looked back! When I'm not working, training or educating, I am currently completing a part-time degree in Computing and IT. Programming is my hobby and my "me" time!

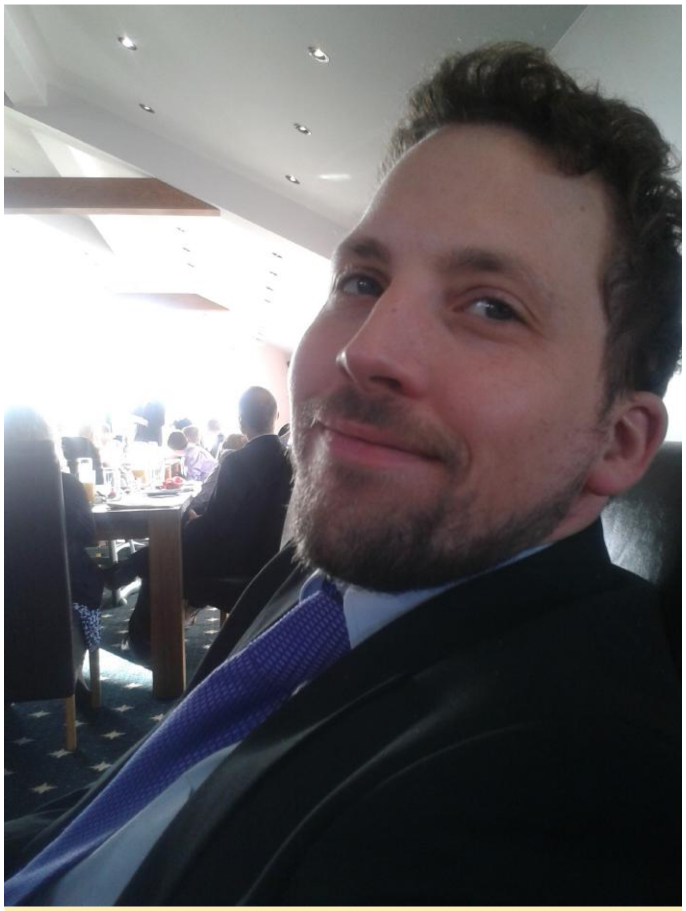


Figure 1 – David Knight

I am very happily married with four children. My wife is a fitness instructor, so I have no excuse to keep myself in shape! Three of my children have finished education and my youngest is still in primary school.

How did you get started with computers? My first computer was the ZX Spectrum 48K. I remember playing Manic Miner on it when I was 6 years old! As we couldn't afford games I remember typing a few BASIC programmes I found in magazines, but didn't really progress from there. I still have my Spectrum though sadly the keyboard has succumbed to the passage of time and no longer works. I think this is where my passion for computing came from though as I remember wanting to learn to program – at university I was the only non-computing student who dared to enter the Unix lab (though I admit it was mainly to play MUDs!)

After much procrastination around 2009, I started dabbling with Linux, learning Bash scripting in the process. I wanted to create a Linux distribution for the visually impaired and got involved with the Vinux project which modified a vanilla Ubuntu distribution to make it more accessible for visually impaired users, which is still going today. By then, I had learned a smattering of Python, but really wanted to get to grips with C and C++. As an exercise, I decided to start by porting Passage to the GCW Zero hand-held. It was certainly and exercise in frustration! After much wasted time, I managed to get it to compile and still remember the thrill of seeing the game run on my system – I was hooked! (**https://boards.dingoonity.org/gcw-releases/passage/**)

Figure 2 – David's wonderful family

Since then, I think I've ported around 30 emulators to handheld consoles, enjoying the challenges each project brings and of course learning along the way. I particularly enjoy the challenge of optimising code so it can run smoothly on slower systems. I spend way too much time improving performance by a few percent or obsessing over tiny features that nobody will notice except me, but I don't like inefficient code or using more power than is necessary. I also enjoy working on the UI so that controls are intuitive and make sense.

What attracted you to the ODROID platform? I was looking for an easy and fun introductory electronics project to make with my son Alexander over the summer holidays. It had to be easy to construct ideally without the need to solder components. After looking at a multitude of Raspberry Pi projects, I found out about the ODROID-GO. He really enjoyed constructing the kit, and his favourite games were Pac-man and Frogger. I was amazed at how much processing power was available in the ESP32 and how comfortable the controls felt. The only problem was that there was no Spectrum emulator.

How do you use your ODROIDs? Our ODROID-GO spends most of its time either in my work bag (for long train rides) or plugged into my computer when we are programming on it. We really enjoy tweaking settings and seeing how it affects the performance of the emulators. I recently made a very low power build that would run games for 19 hours.

Which ODROID is your favourite and why? I only have my ODROID-GO at the moment. I am attracted to efficient low power portable devices and love the ESP32, it has loads of power but uses so little energy! I have really enjoyed reading the Programming Guide and learning about the rich feature set available to developers.

What innovations would you like to see in future Hardkernel products? As a portable device, the ODROID-GO ideally should have a headphone socket (I'm looking at you Apple). Of course the ESP32 is capable of Bluetooth, so perhaps wireless headphones will be a possibility in future. Also currently sound volume is reduced by reducing the bit depth of sound samples, producing poor quality sound at low volume. It would be great to have a potentiometer with perhaps a scroll wheel to control volume. Similarly screen brightness control could perhaps be controlled autonomously by an LDR.

What hobbies and interests do you have apart from computers? Apart from a spot of retro gaming, I'm not really a gamer. When not spending time with my work and family I enjoy running, cycling, singing in a choir and juggling.

What advice do you have for someone wanting to learn more about programming? The reason I got into programming was because I needed to understand how computers work and I soon realised the myriad possibilities that opened up. As a beginner, I wish someone had given me "How to Think Like a Computer Scientist". It's a great introduction to computer programming. When learning, it is important to get small victories; Scratch is a great way to teach the fundamentals and retain interest, particularly with younger children. Start with fun, small projects with simple challenges to solve. Street Fighter 2 with cats? Yup, we did that with Scratch!

Links

Vinux project website **http://www.vinux.org.uk/about.html** My first compiled port **https://boards.dingoonity.org/gcw-releases/passage/** My Github and BitBucket sites **https://github.com/DavidKnight247**, **https://bitbucket.org/DavidKnight247**