Digole Display • Wall Monitor • U-Boot • Linux Gaming • Home Assistant

ODROID Magazine

Year Four
Issue #45
Sep 2017

Stackable Power

ODROID HC-1 *Home Cloud*

Docker Swarm:

Take the ODROID-C2 to the next level

ODROID-HC1 and ODROID-MC1: Affordable High-Performance And Cloud Computing At Home

September 1, 2017 By Rob Roy ODROID-HC1

Many people have been using the ODROID-XU4 for server, NAS, cluster, mining and build-farm applications, thanks to its high computing performance and connectivities. They kept requesting easier and cheaper solutions for scalability with a stripped-down version of XU4, so Hardkernel has introduced a new product that is intended to be used for building an affordable and powerful Home Cloud server, called the ODROID-HC1, which is now available for USD$49 at http://bit.ly/2wjNToV.

ODROID-HC1

The ODROID-HC1 is a single board computer (SBC) which is an affordable solution for a network attached storage (NAS) server. This home cloud server centralizes data and enables users to share and stream multimedia files to phones, tablets and other devices on a network, which is ideal for a single user on many devices, for sharing files between family members, and for developers or a group. You can tailor the ODROID-HC1 to your specific needs, and there is plenty of software available with minimal configuration. The storage of the server can be customized by using a large capacity hard drive or SSD. Depending on your needs, the frame is made to be stackable. The engineered metal frame body is designed to store a 2.5 inch HDD/SSD while offering excellent heat dissipation.

The ODROID-HC1 is an elegant, affordable solution to home cloud computing

The integrated SATA interface will give you the best performance

The ODROID-HC1 is an amazing stackable board for building your personal cluster

The ODROID-HC1 is based on the very powerful ODROID-XU4 platform, and can run Samba, FTP, NFS, SSH, NGINX, Apache, SQL, Docker, WordPress and other server software smoothly using full Linux distributions like Ubuntu, Debian, Arch and OMV. Available and ready-to-go operating system (OS) distributions are on the Hardkernel Wiki at http://bit.ly/2wjNsuI. Any OS that runs on the XU4 is fully compatible with the HC1. We guarantee the production of ODROID-HC1 to the middle of 2020, but expect to continue production long after.

Key features

- Samsung Exynos5422 Cortex-A15 2Ghz and Cortex-A7 Octa core CPUs
- 2Gbyte LPDDR3 RAM PoP stacked

- SATA port for 2.5inch HDD/SSD storage
- Gigabit Ethernet port
- USB 2.0 Host
- UHS-1 capable micro-SD card slot for boot media
- Size : approximately 147 x 85 x 29 mm (including aluminum cooling frame)
- Linux server OS images based on modern Kernel 4.9 LTS

Since bad USB cables and faulty USB-to-SATA bridge chipsets make users struggling due to physical/electrical tolerance issues as well as driver compatibility issues, the ODROID-HC1 has a built-in SATA connector on the PCB with a fully tested SATA bridge controller. To lower the cost, the board size was minimized due to the cost of the 10-layers PCB, which resulted in the removal of some features such as the HDMI output, eMMC connector, USB 3.0 hub, power button, and slide switch.

Any type of 2.5-inch SATA HDD/SSD storage may be installed, including 7mm, 9.5mm, 12mm and 15mm thick units. The Seagate Barracuda 2TB/5TB HDDs, Samsung 500GB HDD and 256GB SSD, Western Digital 500GB and 1TB HDD, HGST 1TB HDD and other storages were fully tested with UAS and S.M.A.R.T. functions.

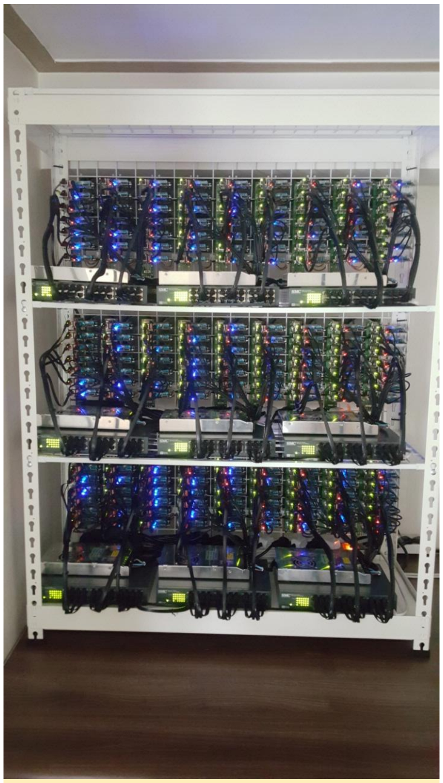
The ODROID-HC1 supports many different types of hard drives

Your distributed server can be modular, compact, and stylish

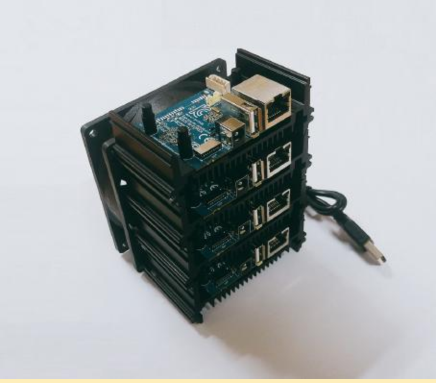
For further details and a demonstration of the ODROID-HC1 functionality, please watch the video at https://youtu.be/t-L99pUANaA.

ODROID-MC1

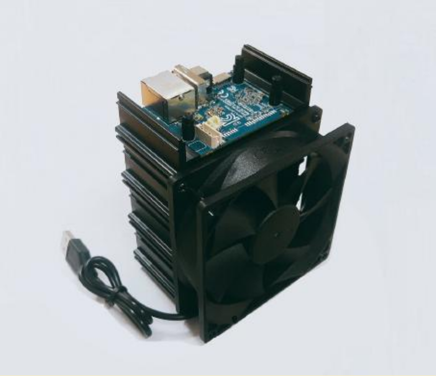
The ODROID-MC1, which stands for My Cluster, is a simple solution for those who need an affordable and powerful personal cluster. It is similar to the ODROID-HC1, but excludes the SATA interface and adds a large cooling fan for heavy computing loads. It is anticipated to be available in September 2017 for USD$200.



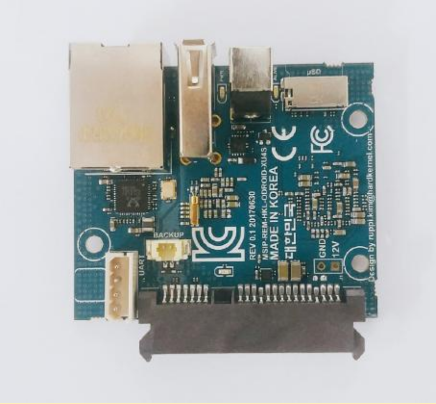
Hardkernel spent several days building this big cluster computer using 200 ODROID-MC1 units with 1600 CPU cores and 400GB RAM



A stack of 4 ODROID-MC1 units, creating a cluster with 32 CPU cores and 8GB RAM.



Cooling 4 ODROIDs with a single fan is a great cost-saving feature



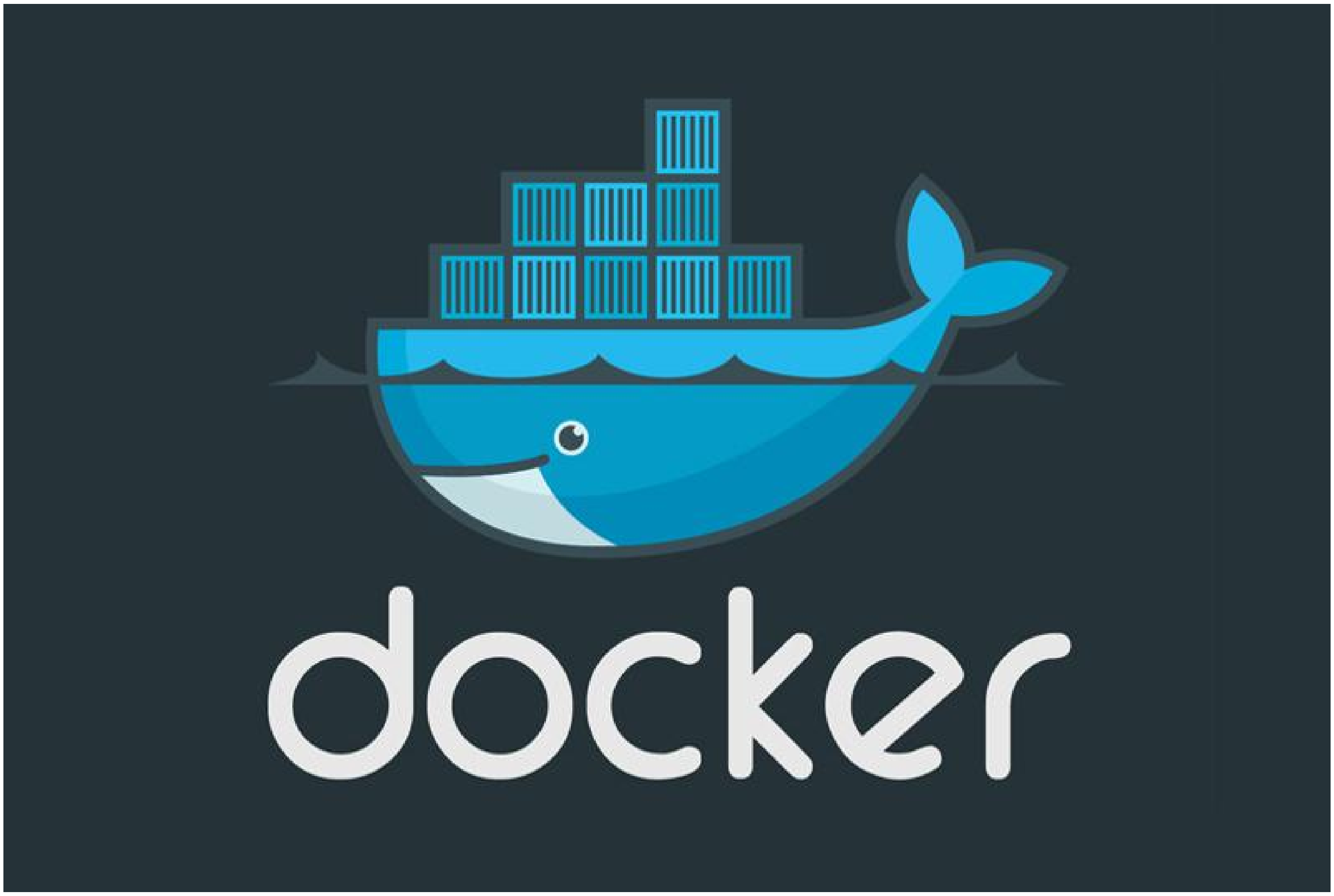
The SoC and PMIC are glued with epoxy resin to increase their reliability

Technical specifications

Schematics http://bit.ly/2vLy7zH
PCB mechanical drawings (AutoCAD format) http://bit.ly/2el0VZm
Product details http://bit.ly/2xzBXho

My ODROID-C2 Docker Swarm: Part 1 – Swarm Mode Features

September 1, 2017 By Andy Yuen Docker, Linux, ODROID-C2



Docker introduced swarm mode in version 1.12.x to enable the deployment of containers on multiple docker hosts. Swarm mode provides cluster management and service orchestration capabilities including service discovery and service scaling, among other things, using overlay networks and an in-built load balancer respectively. These are mandatory features for the enterprise as there is a limit to the number of containers one can deploy on a single docker host. For a high level architectural description of swarm mode, please read my previous article published in the November 2016 issue of ODROID Magazine at http://bit.ly/2wiTVXM.

Several months ago, I was experimenting with Docker's swarm mode on my five board ODROID-C2 cluster. I was able to start multiple Docker containers on multiple docker hosts but neither overlay network, routing mesh, nor load balancing were working in swarm mode. I tried using different versions of docker (1.12.x and 1.13.x) compiled on my ODROID-C2 to no avail. I also tried running Kubernetes on my ODROID-C2 cluster. Again the networking part of Kubernetes did not work. I suspected that the kernel was missing certain modules needed for Docker and Kubernetes networking. Due to this, I stopped my experimentation until now. What rekindled my passion to get Docker swarm mode working was seeing my hardware not being used: an ODROID VU7 multi-touch screen and a VuShell for VU7.

I assembled the VU7 screen and an ODROID-C1+ with the VuShell enclosure. Then I thought to myself, why not put my ODROID-C2 cluster there as well? You can see the screen displaying a soft keyboard in the Figure 1. All ODROID single board computers are connected together with an 8-port gigabit Ethernet switch, and an SSD is also put inside the VuShell enclosure. The ODROID cardboard box houses the power supply. The tiny wireless router uses Wireless Distribution System, WDS, to connect to my main router to provide Ethernet Internet access for all the ODROIDs housed in the VuShell, because they don't have built-in WiFi.

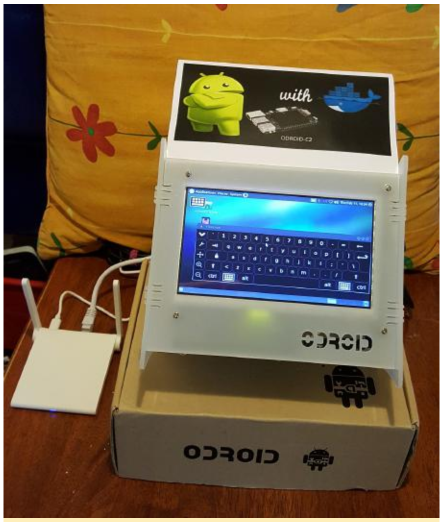


Figure 1 – A Docker swam cluster using the ODROID-VU shell as a case

Hardkernel's Ubuntu 16.04 OS

I had the suspicion that the cause for Docker's swarm mode not working in previous attempts was due to some missing or incompatible kernel modules in the OS. So, I decided to switch to another OS. I noticed that Hardkernel recently released Ubuntu 16.04 (v2.3) for the ODROID-C2 so I gave it a try. The earlier version of Hardkernel's Ubuntu OS that I tried months earlier was unstable, but the current release worked without any issues. I was happy and told myself that this time it might work!

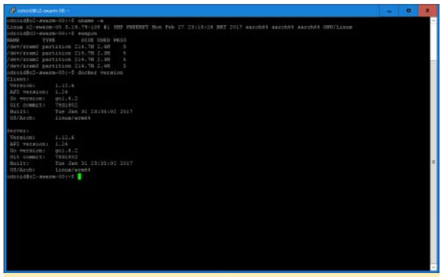
To make things easier, I installed and configured the following packages:

- parallel-ssh on the docker manager to allow me to issue commands once from the docker manager to be executed on all nodes
- nfs-kernel-server on the manager and nfs-common on all nodes
- curl on the manager for testing
- dnsutils on all nodes

I also generated SSH keys for the "odroid" and "root" users on all members of the cluster, so that they can SSH into each other without a password.

Docker Swarm Mode Reboot

I installed docker.io using apt-get and did a quick "docker run" test using my httpd image, and it worked. I wanted to try out the swarm mode next to see if it will work with the new OS. Here is a screenshot of the versions of software being used. It is interesting to note that Hardkernel's Ubuntu distribution came with zram pre-installed for swap, which is handy.



```
odroid@c2-swarm-00:~$ uname -a
Linux c2-swarm-00 3.14.79-108 #1 SMP PREEMPT Mon Feb 27 23:18:28 BRT 2017 aarch64 aarch64 aarch64 GNU/Linux
odroid@c2-swarm-00:~$ swapon
NAME       TYPE        SIZE USED PRIO
/dev/zram0 partition 214.7M 2.4M    5
/dev/zram1 partition 214.7M 2.3M    5
/dev/zram2 partition 214.7M 2.3M    5
/dev/zram3 partition 214.7M 2.4M    5
odroid@c2-swarm-00:~$ docker version
Client:
 Version:      1.12.6
 API version:  1.24
 Go version:   go1.6.2
 Git commit:   78d1802
 Built:        Tue Jan 31 23:35:02 2017
 OS/Arch:      linux/arm64

Server:
 Version:      1.12.6
 API version:  1.24
 Go version:   go1.6.2
 Git commit:   78d1802
 Built:        Tue Jan 31 23:35:02 2017
 OS/Arch:      linux/arm64
odroid@c2-swarm-00:~$
```

Figure 2 – Docker showing the versions of all of the current software

Creating a Swarm

The host names and static IP addresses for my swarm hosts are:

- c2-swarm-00 – 192.168.1.100 (manager)
- c2-swarm-01 – 192.168.1.101 (node 1)
- c2-swarm-02 – 192.168.1.102 (node 2)
- c2-swarm-03 – 192.168.1.103 (node 3)
- c2-swarm-04 – 192.168.1.104 (node 4)

Only c2-swarm-00 has a SSD drive connected, but the file system is shared using NFS.

A node is a docker host participating in a swarm. A manager node is where you submit a service definition and it schedules the service to run as tasks on worker nodes. Worker nodes receive and execute tasks scheduled by a manager node. A manager node, by default, is also a worker node unless explicitly configured not to execute tasks. Multiple master and worker nodes can be set up in a swarm to provide High Availability (HA). To bring up swarm mode, issue the following command on the manager:

```
$ docker swarm init --advertise-addr
192.168.1.100
```

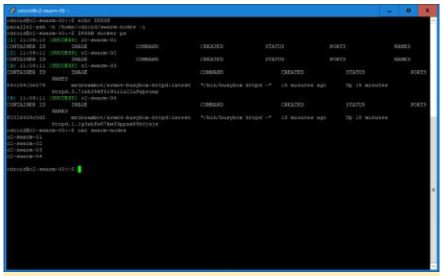
which returns:

```
swarm initialized: current mode
(8jw6y313hmt3vfa1me1dinro) is now a manager
```

To add a worker to this swarm, run the following command on each node:

```
$ docker swarm join --token SWMTKN-1-
2gvqzfx48uw8zcokwl5033iwdel2rl9n96lc0wjlqso7
lrztub-aokk5xcm5v7c4usmeswsgg1k
192.168.1.100:2377
```

To make the other nodes join the cluster, issue the previous "docker swarm join" command on each node. This can be done using parallel-ssh to issue the command once from the manager, which is then executed on each node. The image below shows a screenshot after running the "docker ps" command using parallel-ssh, which signifies that the Docker swarm is up and running.



```
odroid@c2-swarm-00:~$ echo $PSSH
parallel-ssh -h /home/odroid/swarm-nodes -i
odroid@c2-swarm-00:~$ $PSSH docker ps
[1] 11:09:10 [SUCCESS] c2-swarm-01
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[2] 11:09:11 [SUCCESS] c2-swarm-02
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS               NAMES
[3] 11:09:11 [SUCCESS] c2-swarm-03
CONTAINER ID        IMAGE                                   COMMAND                  CREATED             STATUS              PORTS
          NAMES
84c184156479        mrdreambot/arm64-busybox-httpd:latest   "/bin/busybox httpd -"   18 minutes ago      Up 18 minutes
          httpd.3.71eh39kf919iils22ufwproep
[4] 11:09:11 [SUCCESS] c2-swarm-04
CONTAINER ID        IMAGE                                   COMMAND                  CREATED             STATUS              PORTS
          NAMES
f2026459c0d5        mrdreambot/arm64-busybox-httpd:latest   "/bin/busybox httpd -"   18 minutes ago      Up 18 minutes
          httpd.1.1g3ahfw07kw33ppzx69h0jeje
odroid@c2-swarm-00:~$ cat swarm-nodes
c2-swarm-01
c2-swarm-02
c2-swarm-03
c2-swarm-04

odroid@c2-swarm-00:~$
```

Figure 3 – The output of the "docker ps" command showing all of the nodes

One annoyance of Docker swarm I found is that after you shut-down all nodes and power them up again, all nodes will be "Active" but "Down". This is seen when you use the "docker node ls" to find out the status of your nodes. Since the nodes are down, all services will be running on the manager. The fix is to run "systemctl restart docker" on every node. This will change their status from "Down" to "Ready", and everything is fine again. The tool parallel-ssh is a convenient way to do this, since all you have to do is issue the command once from your manager.

Running Docker Swarm Visualizer and HTTPD Services

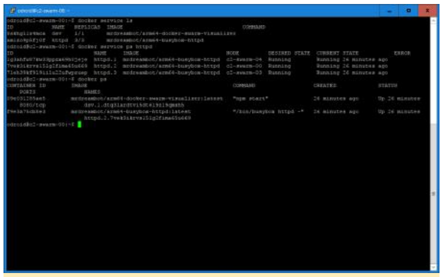
To help visualize what is going on in the swarm, I built the "Docker Swarm Visualizer" image based on Docker Samples on Github. I've pushed it to docker hub at http://dockr.ly/2ipXzcL, so that anyone can use it. The image's name is "mrdreambot/arm64-docker-swarm-visualizer", available at http://bit.ly/2xqSaV4. I then deployed it as a service by issuing the following command from the manager:

```
$ docker service create --name=dsv --
publish=8080:8080/tcp --
constraint=node.role==manager --
mount=type=bind,src=/var/run/docker.sock,dst
=/var/run/docker.sock mrdreambot/arm64-
docker-swarm-visualizer
```

I then pointed the browser at the master node at http://192.168.1.100:8080, but it also works when you point your browser to any of the nodes in the swarm. The changes reported by the visualizer when deploying the httpd service can then be observed:

```
$ docker network create --driver overlay
home-net
$ docker service create --replicas 3 --
network home-net --name httpd -p 80:80
mrdreambot/arm64-busy-box-httpd
```

The command line output for listing the services is shown in Figure 4. Figure 5 is a Docker Swarm Visualizer screenshot showing the nodes on which the service replicas are run, which illustrates the declarative service model used by swarm mode.



```
odroid@c2-swarm-00:~$ docker service ls
ID            NAME   REPLICAS  IMAGE                                     COMMAND
8s4ngiis4mca  dsv    1/1       mrdreambot/arm64-docker-swarm-visualizer
amizo9pkfj0f  httpd  3/3       mrdreambot/arm64-busybox-httpd
odroid@c2-swarm-00:~$ docker service ps httpd
ID                         NAME     IMAGE                           NODE         DESIRED STATE  CURRENT STATE           ERROR
1g3ahfw07kw33ppzx69h0jeje  httpd.1  mrdreambot/arm64-busybox-httpd  c2-swarm-04  Running        Running 26 minutes ago
7vek3ikrvs151g2fima65u669  httpd.2  mrdreambot/arm64-busybox-httpd  c2-swarm-00  Running        Running 26 minutes ago
71eh39kf919iils22ufwproep  httpd.3  mrdreambot/arm64-busybox-httpd  c2-swarm-03  Running        Running 26 minutes ago
odroid@c2-swarm-00:~$ docker ps
CONTAINER ID        IMAGE                                             COMMAND                  CREATED             STATUS
    PORTS               NAMES
89e031285ee5        mrdreambot/arm64-docker-swarm-visualizer:latest   "npm start"              26 minutes ago      Up 26 minutes
    8080/tcp            dsv.1.dtq3lardtvi5d0619z19gmshh
f9e3a75cb8e3        mrdreambot/arm64-busybox-httpd:latest             "/bin/busybox httpd -"   26 minutes ago      Up 26 minutes
                        httpd.2.7vek3ikrvs151g2fima65u669
odroid@c2-swarm-00:~$
```

Figure 4 – Command line output for listing the services

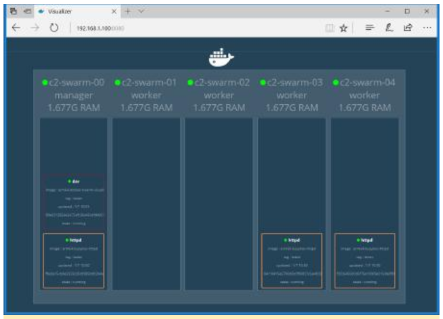


Figure 5 – Docker Swarm Visualizer

Routing Mesh, Load Balancing and Self-healing

The routing mesh in the swarm allows a request to reach a service even when the service is not running on the node where the request has been received. This means that although the httpd service is running on c2-swarm-00, c2-swarm-03 and c2-swarm-04, one can point the browser at any one of the 5 nodes and still get a response with the ODROID-Docker image. This was the behaviour that I observed.

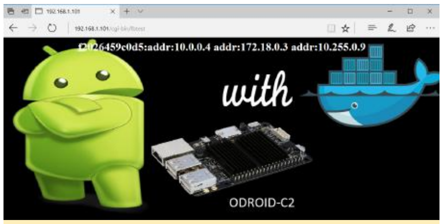

192.168.1.101
192.168.1.101/cgi-bin/lbtest
f2026459c0d5:addr:10.0.0.4 addr:172.18.0.3 addr:10.255.0.9
with
ODROID-C2

Figure 6 – Load balancing example using 10.255.0.9

In addition to providing a routing mesh, the swarm also performs load balancing. To test the load balancing feature, I connected to the manager multiple time using my browser, at the httpd service using the address http://192.168.1.100/cgi-bin/lbtest. Notice that the hostnames (container Id) and IP addresses are different in the two screenshots.

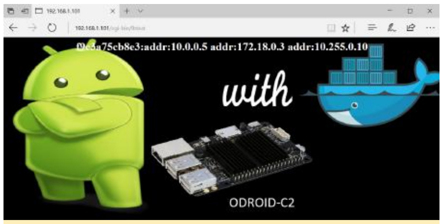
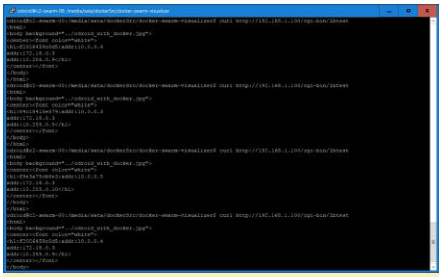

192.168.1.101
192.168.1.101/cgi-bin/lbtest
f9e3a75cb8e3:addr:10.0.0.5 addr:172.18.0.3 addr:10.255.0.10
with
ODROID-C2

Figure 7 – Load balancing example using 10.255.0.10

The tests were repeated using the curl command:

```
$ curl http://192.168.1.100/cgi-bin/lbtest
```

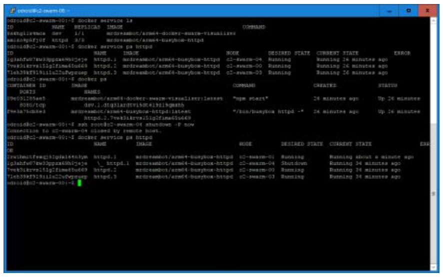
Here is a screenshot of the curl commands output which confirmed, again, that each request has been directed to a different node:



```
odroid@c2-swarm-00:/media/sata/dockerSrc/docker-swarm-visualizer$ curl http://192.168.1.100/cgi-bin/lbtest
<html>
<body background="../odroid_with_docker.jpg">
<center><font color="white">
<h1>f2026459c0d5:addr:10.0.0.4
addr:172.18.0.3
addr:10.255.0.9</h1>
</center></font>
</body>
</html>
odroid@c2-swarm-00:/media/sata/dockerSrc/docker-swarm-visualizer$ curl http://192.168.1.100/cgi-bin/lbtest
<html>
<body background="../odroid_with_docker.jpg">
<center><font color="white">
<h1>84c184156479:addr:10.0.0.3
addr:172.18.0.3
addr:10.255.0.5</h1>
</center></font>
</body>
</html>
odroid@c2-swarm-00:/media/sata/dockerSrc/docker-swarm-visualizer$ curl http://192.168.1.100/cgi-bin/lbtest
<html>
<body background="../odroid_with_docker.jpg">
<center><font color="white">
<h1>f9e3a75cb8e3:addr:10.0.0.5
addr:172.18.0.3
addr:10.255.0.10</h1>
</center></font>
</body>
</html>
odroid@c2-swarm-00:/media/sata/dockerSrc/docker-swarm-visualizer$ curl http://192.168.1.100/cgi-bin/lbtest
<html>
<body background="../odroid_with_docker.jpg">
<center><font color="white">
<h1>f2026459c0d5:addr:10.0.0.4
addr:172.18.0.3
addr:10.255.0.9</h1>
</center></font>
</body>
```

Figure 8 – Load balancing across nodes

As for a demo on self-healing, I shut down c2-swarm-04, and you can see from the visualizer as well as the command line that another httpd container was spun up on c2-swarm-02 to replace the one on c2-swarm-04. This is because when we started the service, we specified "replica=3". This means the Docker swarm will maintain the desired number of replicas, here it is 3. This is called desired state reconciliation.



```
odroid@c2-swarm-00:~$ docker service ls
ID            NAME   REPLICAS  IMAGE                                     COMMAND
8s4ngiis4mca  dsv    1/1       mrdreambot/arm64-docker-swarm-visualizer
amizo9pkfj0f  httpd  3/3       mrdreambot/arm64-busybox-httpd
odroid@c2-swarm-00:~$ docker service ps httpd
ID                         NAME     IMAGE                           NODE         DESIRED STATE  CURRENT STATE           ERROR
1g3ahfw07kw33ppzx69h0jeje  httpd.1  mrdreambot/arm64-busybox-httpd  c2-swarm-04  Running        Running 26 minutes ago
7vek3ikrvs151g2fima65u669  httpd.2  mrdreambot/arm64-busybox-httpd  c2-swarm-00  Running        Running 26 minutes ago
71eh39kf919iils22ufwproep  httpd.3  mrdreambot/arm64-busybox-httpd  c2-swarm-03  Running        Running 26 minutes ago
odroid@c2-swarm-00:~$ docker ps
CONTAINER ID        IMAGE                                             COMMAND                  CREATED             STATUS
    PORTS               NAMES
89e031285ee5        mrdreambot/arm64-docker-swarm-visualizer:latest   "npm start"              26 minutes ago      Up 26 minutes
    8080/tcp            dsv.1.dtq3lardtvi5d0619z19gmshh
f9e3a75cb8e3        mrdreambot/arm64-busybox-httpd:latest             "/bin/busybox httpd -"   26 minutes ago      Up 26 minutes
                        httpd.2.7vek3ikrvs151g2fima65u669
odroid@c2-swarm-00:~$ ssh root@c2-swarm-04 shutdown -P now
Connection to c2-swarm-04 closed by remote host.
odroid@c2-swarm-00:~$ docker service ps httpd
ID                         NAME         IMAGE                           NODE         DESIRED STATE  CURRENT STATE               ERR
OR
2rwihmutfemgi5lgdsl44ntym  httpd.1      mrdreambot/arm64-busybox-httpd  c2-swarm-01  Running        Running about a minute ago
1g3ahfw07kw33ppzx69h0jeje   \_ httpd.1  mrdreambot/arm64-busybox-httpd  c2-swarm-04  Shutdown       Running 34 minutes ago
7vek3ikrvs151g2fima65u669  httpd.2      mrdreambot/arm64-busybox-httpd  c2-swarm-00  Running        Running 34 minutes ago
71eh39kf919iils22ufwproep  httpd.3      mrdreambot/arm64-busybox-httpd  c2-swarm-03  Running        Running 34 minutes ago
odroid@c2-swarm-00:~$
```

Figure 9 – Service recovery

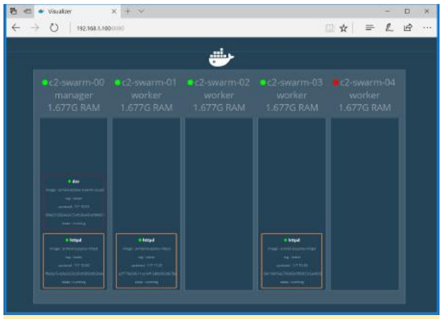


Figure 10 – Service recovery httpd

I then shut down the rest of the nodes and left only the manager running and the Visualizer showed the output in Figure 11.

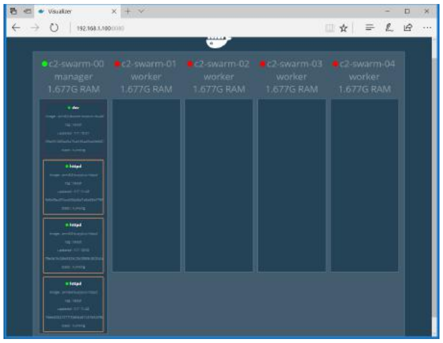


Figure 11 – Service recovery httpd – 1 node

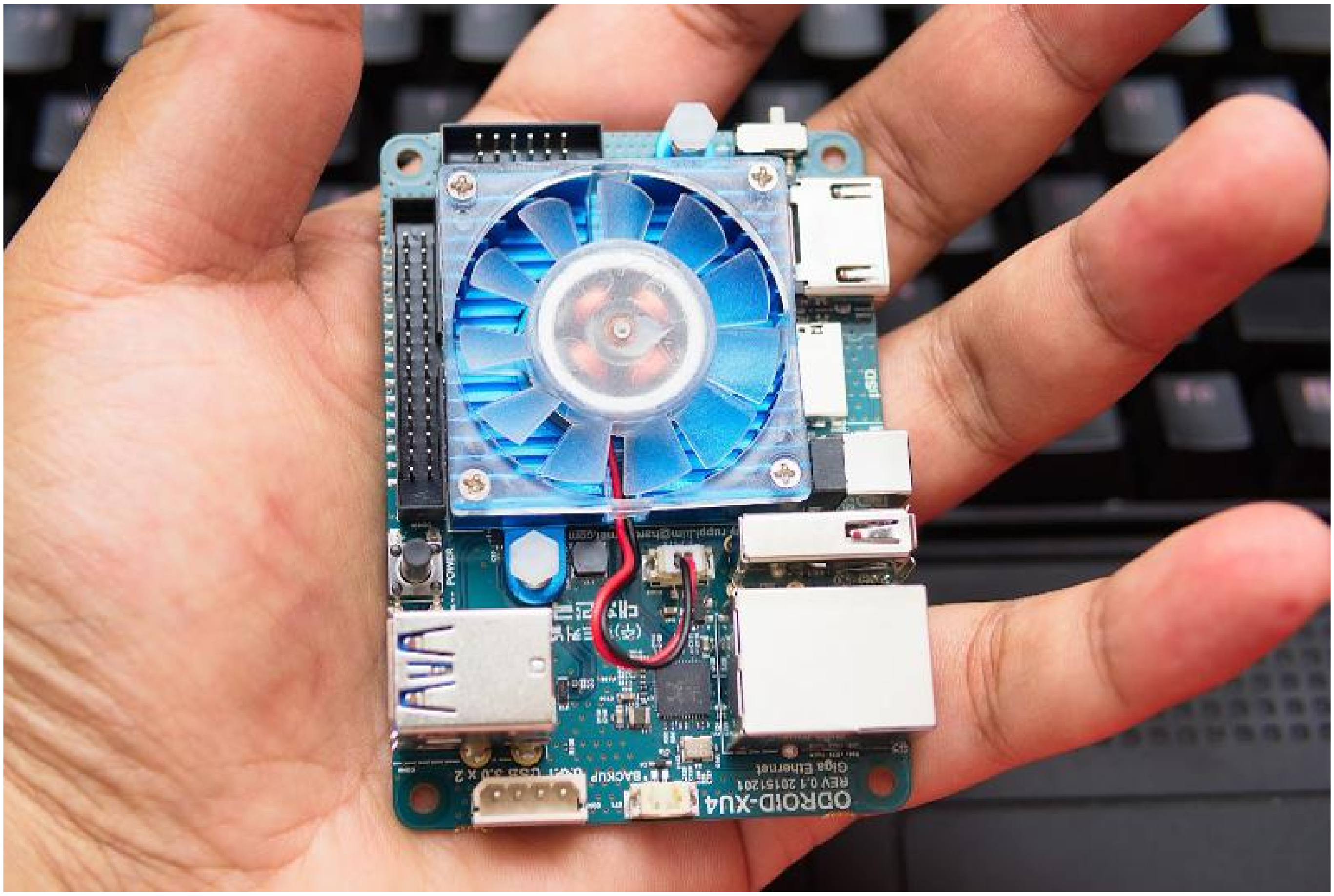
Everything worked as expected!

Conclusion

Hardkernel's new Ubuntu 16.04 release really made a difference. The Docker swarm mode is now fully working on my ODROID-C2 cluster. In the next installment, I will upgrade Docker to 1.13.x to experiment with the "docker stack deploy" feature new to v.1.13.x. A stack is a collection of services that make up an application. It automatically deploys multiple services that are linked to each other, removing the need to define each one separately. In other words, it is docker-compose in swarm mode which manages service orchestration. The use of an overlay network for service discovery will also be described.

ODROID-XU4 Mainline U-Boot

September 1, 2017 By Rob Roy Linux, ODROID-XU4



Hardkernel is working on a new version of U-Boot for the ODROID-XU4, with the following capabilities:

- Enables the HYP mode for the KVM virtualization with Kernel 4.9 LTS
- Enables the Ethernet device to support the TFTP/PXE remote booting
- Boots from various new eMMC chipsets.
- All fatload and ext4load commands are available natively
- Many other new features

The new version of U-Boot is available on the latest update package for Linux, and Android 4.4 and Android 7.1 users will receive an update soon, which will include the fastboot protocol. The source code is available at http://bit.ly/2xrM7R3.

A sample boot log is shown below:

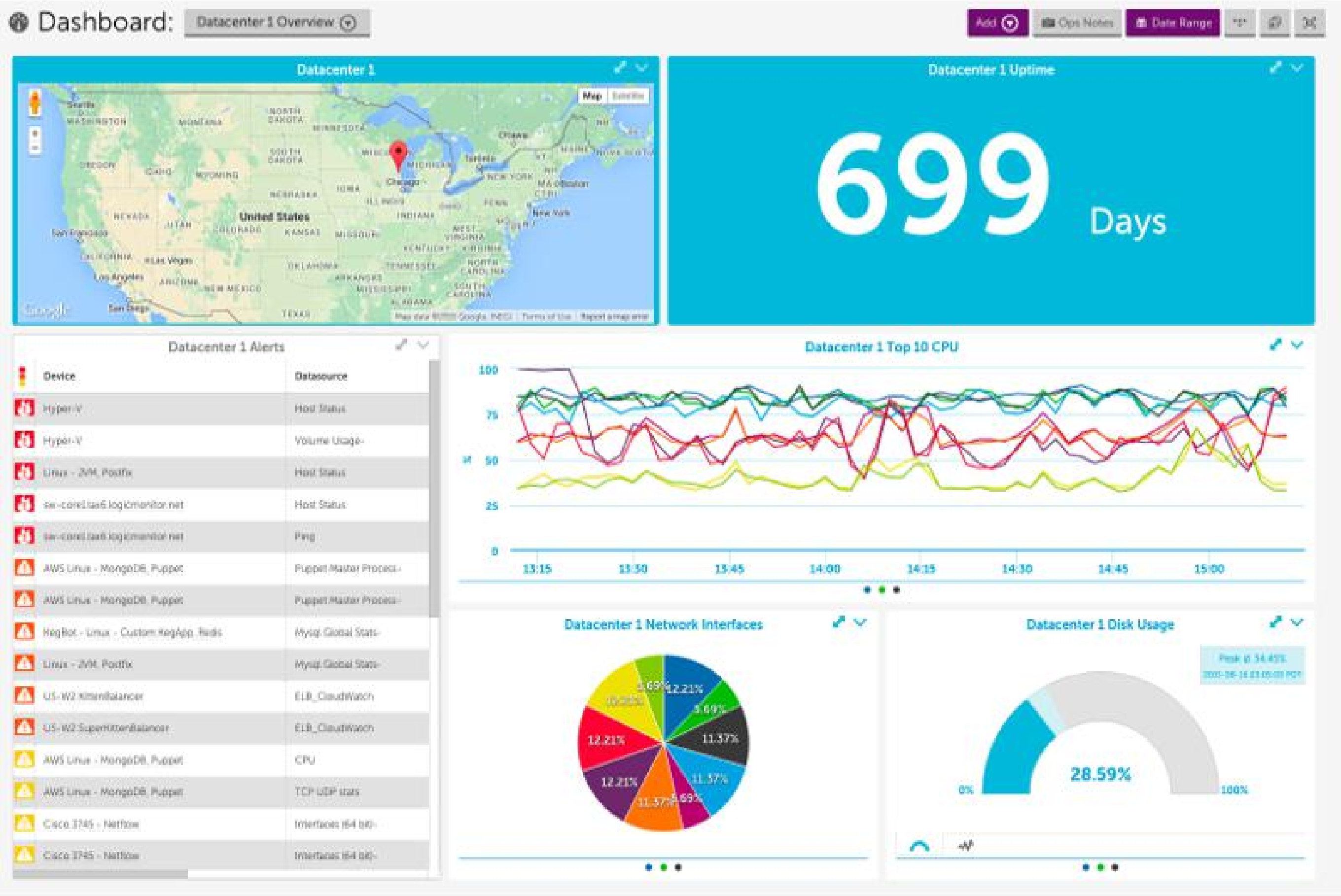
```
u-boot booting log from the serial console
output.
U-Boot 2017.05-12186-gf98cc91-dirty (Aug 08
2017 - 12:16:58 +0900) for ODROID XU4

CPU:   Exynos5422 @ 800 MHz
Model: Odroid XU4 based on EXYNOS5422
Board: Odroid XU4 based on EXYNOS5422
Type:  xu4
DRAM:  2 GiB
MMC:   EXYNOS DWMMC: 0, EXYNOS DWMMC: 1
MMC Device 0 (eMMC): 14.7 GiB
Info eMMC rst_n_func status = enabled
Card did not respond to voltage select!
mmc_init: -95, time 11
*** Warning - bad CRC, using default
environment

In:    serial
Out:   serial
Err:   serial
Net:   No ethernet found.
Press quickly 'Enter' twice to stop
autoboot:  0
reading boot.ini
9088 bytes read in 4 ms (2.2 MiB/s)
cfgload: applying boot.ini...
cfgload: setenv initrd_high "0xffffffff"
cfgload: setenv fdt_high "0xffffffff"
cfgload: setenv macaddr "00:1e:06:61:7a:39"
cfgload: setenv vout "hdmi"
cfgload: setenv cecenable "false" # false or
true
cfgload: setenv disable_vu7 "false" # false
cfgload: setenv governor "performance"
cfgload: setenv ddr_freq 825
cfgload: setenv external_watchdog "false"
cfgload: setenv external_watchdog_debounce
"3"
cfgload: setenv HPD "true"
cfgload: setenv bootrootfs "console=tty1
console=ttySAC2,115200n8 root=UUID=e139ce78-
9841-40fe-8823-96a304a09859 rootwait ro
fsck.repair=yes net.ifnames=0"
cfgload: fatload mmc 0:1 0x40008000 zImage
reading zImage
4793144 bytes read in 135 ms (33.9 MiB/s)
cfgload: fatload mmc 0:1 0x42000000 uInitrd
reading uInitrd
5327028 bytes read in 143 ms (35.5 MiB/s)
cfgload: if test "${board_name}" = "xu4";
then fatload mmc 0:1 0x44000000 exynos5422-
odroidxu4.dtb; setenv fdtloaded "true"; fi
reading exynos5422-odroidxu4.dtb
61570 bytes read in 9 ms (6.5 MiB/s)
cfgload: if test "${board_name}" = "xu3";
then fatload mmc 0:1 0x44000000 exynos5422-
odroidxu3.dtb; setenv fdtloaded "true"; fi
cfgload: if test "${board_name}" = "xu3l";
then fatload mmc 0:1 0x44000000 exynos5422-
odroidxu3-lite.dtb; setenv fdtloaded "true";
fi
cfgload: if test "${fdtloaded}" != "true";
then fatload mmc 0:1 0x44000000 exynos5422-
odroidxu4.dtb; fi
cfgload: fdt addr 0x44000000
cfgload: setenv hdmi_phy_control "HPD=${HPD}
vout=${vout}"
cfgload: if test "${cecenable}" = "false";
then fdt rm /cec@101B0000; fi
cfgload: if test "${disable_vu7}" = "false";
then setenv hid_quirks
"usbhid.quirks=0x0eef:0x0005:0x0004"; fi
cfgload: if test "${external_watchdog}" =
"true"; then setenv external_watchdog
"external_watchdog=${external_watchdog}
external_watchdog_debounce=${external_watchd
og_debounce}"; fi
cfgload: setenv bootargs "${bootrootfs}
${videoconfig} ${hdmi_phy_control}
${hid_quirks} smsc95xx.macaddr=${macaddr}
${external_watchdog} governor=${governor}"
```

```
cfgload: bootz 0x40008000 0x42000000
0x44000000
Kernel image @ 0x40008000 [ 0x000000 -
0x492338 ]
## Loading init Ramdisk from Legacy Image at
42000000 ...
   Image Name:   uInitrd
   Image Type:   ARM Linux RAMDisk Image
(uncompressed)
   Data Size:    5326964 Bytes = 5.1 MiB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
## Flattened Device Tree blob at 44000000
   Booting using the fdt blob at 0x44000000
   Using Device Tree in place at 44000000,
end 44012081

Starting kernel ...
```

For comments, questions and suggestions, please visit the original post at **http://bit.ly/2wNfnVu**.

ODROID Wall Display: Using An LCD Monitor And An ODROID To Show Helpful Information

September 1, 2017 By William Green ODROID-C0



Datacenter 1 Alerts

Device	Datasource
Hyper-V	Host Status
Hyper-V	Volume Usage-
Linux - JVM, Postfix	Host Status
sw-core1.lax6.logicmonitor.net	Host Status
sw-core1.lax6.logicmonitor.net	Ping
AWS Linux - MongoDB, Puppet	Puppet Master Process-
AWS Linux - MongoDB, Puppet	Puppet Master Process-
KegBot - Linux - Custom KegApp, Redis	Mysql Global Stats-
Linux - JVM, Postfix	Mysql Global Stats-
US-W2 KittenBalancer	ELB_CloudWatch
US-W2 SuperKittenBalancer	ELB_CloudWatch
AWS Linux - MongoDB, Puppet	CPU
AWS Linux - MongoDB, Puppet	TCP UDP stats
Cisco 3745 - Netflow	Interfaces (64 bit)-
Cisco 3745 - Netflow	Interfaces (64 bit)-

I am a technical intern at ameriDroid.com, and I helped with answering technical questions and got to work on fun and interesting projects involving ODROIDs and related electronics. This is a writeup of a recent project that I just finished, which involved creating a wall monitor.

A wall monitor is an effective method of passively delivering a constant stream of information. Rather than purchasing a digital picture frame that is not only expensive, but is also small with limited functionality, why not use a computer monitor or TV screen with an ODROID to display photos, weather information, RSS feeds, and other media?

Before assembling the project, you'll need the following things:

- an ODROID (we used an ODROID-C0, but others can work fine)
- USB WiFi module or an Ethernet cable, if the board allows
- computer monitor
- power adapter
- video output cable
- mouse and keyboard, for setup
- a method of mounting the ODROID on or near the monitor

Double sided mounting tape can be a handy way to mount the ODROID behind the monitor. The ODROID-C0 is a good choice due to its compact size and minimal cost. A Linux distribution is required, and Hardkernel's Lubuntu is recommended for the C0. The keyboard and mouse are necessary for creating the project, but the wall monitor can run without a keyboard or mouse after the project is setup. The ODROID can be controlled at any time using SSH if necessary.

After the required materials have been collected, follow the steps below to assemble the monitor. Since ODROIDs, like most other single board computers, lack a BIOS interface, it instead uses a special file where various settings, such as screen resolution, are stored. To allow the ODROID to output a signal to the monitor, first find out the monitor's screen resolution. Edit the boot.ini file on the boot partition of your media by uncommenting the matching monitor specification resolution. Power on the ODROID and connect to an available wireless network. Then, run the software updater using the following command:

```
$ sudo apt update
```

After restarting the ODROID, the next step is to install the required software. Navigate to the terminal and install the unclutter program:

```
$ sudo apt install unclutter
```

Unclutter will hide the cursor when not in use. DAKboard is a website that will provide the monitor with its content. Navigate to dakboard.com, register an account, and customize the page with RSS feeds, photos, weather, and other information. Under DAKboard Options, Account Settings, there is a private URL, which you should copy. Open the terminal again and type the following:

```
$ sudo vi
~/.config/xlsession/Lubuntu/autostart
```

Press the o key and type "firefox –url", then paste your DAKboard private URL. In vi, you can paste by clicking the left and right mouse buttons at the same time. Press the escape key, then type ":wq". Open Firefox and navigate to https://mzl.la/2wpotqS in order to install the R-kiosk extension for Firefox. After the extension is installed, disconnect the power from the ODROID. Mount the ODROID near the monitor, and position and organize the power and video cables. Power on the ODROID and verify that everything is working properly before mounting the monitor to the wall.

The ODROID should now display the information that you configured in DAKboard. Log in to DAKboard on another computer to edit the page at any time. The page should now provide live information for weather, photos, and RSS feeds.

Linux Gaming on ODROID: Fanboy Part 2 – I am a Sega Fanboy!

September 1, 2017 By Tobias Schaaf Gaming, Linux

In last month's issue, I talked about how I am not particularly a Nintendo fanboy. By definition, this must mean I'm a Sega fanboy, right? I mean, that's the typical assumption: you're either a Nintendo or a Sega fanboy. Since I already said I'm not Nintendo, that must mean I'm Sega, right? I guess that's right, but let's take a deeper look into it.

Sega Hardware

In the 1980s and 1990s, Nintendo and Sega fought to dominate the console market. While both had similar products, there were still some major differences. If you compare their earlier products, the Nintendo Entertainment System (NES) and Sega Master System (SMS), and Game Boy/Game Boy Color (GB/GBC) and Game Gear (GG), it always seemed like Sega was in the lead when it came to technology.

The SMS had a faster processor, more RAM, more colors, and could display more of those colors at the same time. Overall it seemed to be the more powerful machine. Still, if you compare sales, the NES outshone the SMS.

When I was still in school, a friend of mine owned an SMS. He often brought it to school so we could play after classes until our parents picked us up. Back then, we mostly played Alex Kidd in Miracle World, which was built into the system. This also meant if you bought a SMS you already got your first game with it and didn't needed to buy one.

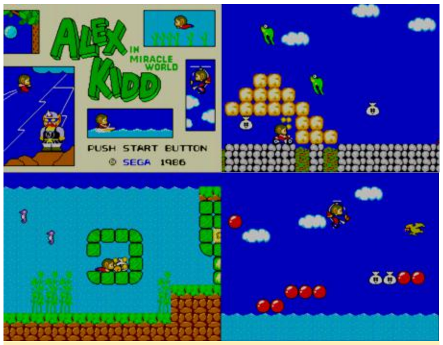


Figure 1 – Alex Kidd in Miracle World on the ODROID with 2xsal-level2-crt shader

The graphics and gameplay blew my mind back then. It was so much better than what I've seen on the NES at my uncle's place, and there were more games we liked to play, like Space Harrier 3D.

Later, I got my own SMS. I thought I would finally be able to finish Alex the Kidd, but it turned out that not every console came with the same built-in game. Mine came with Hang-On, which was a nice motorcycle racing game, but still not what I wanted at the time.

Someone else at school had a Game Gear, and wow, was that a huge thing. This was a portable game machine, unlike the Game boy, this was the real deal, with color and everything. You could even play your SMS games on the Game Gear if you had an adapter, and there was even an add-on which allowed you to watch TV with a built-in antenna. How cool was that?

The Game Gear had nearly the same specs as the SMS. The CPU was the same, but was just clocked slightly lower, and it was able to use even more colors than the SMS. Overall, it was an awesome handheld device, but which came with a hefty price. The device itself wasn't cheap and it used batteries very quickly. You only had a play time of 3-5 hours on 6 AA batteries. Still, seeing what the GG was capable of is what prevented me from ever seriously wanting a Game Boy. So yes, even in my early years I preferred Sega consoles over Nintendo. Perhaps for the wrong reasons, but I was a kid and didn't know any better.

Looking back today, it's clear why Nintendo had the lead at the time. The number of high quality games on the NES was much higher than on the SMS. This was partly due to restrictions Nintendo that put on their third-party software developers, which prevented them from producing games for Nintendo's competitors if they wanted if they wanted their games on Nintendo's consoles.

I missed most of the fourth and fifth generation Sega consoles due to the fact I was not really into console gaming at that time. I missed all the goodness and badness of the Genesis/Mega Drive, Sega CD, Sega 32X, and Sega Saturn. I got the Sega Dreamcast much later, which was the last console that Sega built. I still have it to this day, and I must say it was one of the most fun consoles I've ever played. I was really impressed, and having Soul Calibur as one of my first titles for the console really blew my mind!

Figures 2 and 3 – Soul Calibur for the Sega Dreamcast running in Reicast on ODROID

Sega obviously made some big mistakes when it came to hardware. The Genesis/Mega Drive was a good device, but Sega tried to expand the lifetime of the system with the 32X, and Sega CD add-ons were expensive and caused a lot of frustration for third-party game developers due to the rapid release cycle of hardware and add-ons, as well as the short lifetime of these add-ons.

Because of this, Sega developed nearly all of the games for the Dreamcast on their own. Sega may not have been the best hardware developer at the time, but they knew how to make good games. They still make games today, and have even developed games for their former rival Nintendo and their consoles.

Sega Emulation on ODROID

Since Sega stopped producing hardware after the Sega Dreamcast, it is probably safe to say that all of Sega's consoles run on ODROID, although some may run better than others. For example, the Sega 32X, which is an addon for the Sega Genesis, was difficult to emulate at first, but thanks to the dynamic recompiler on the PicoDrive emulator we can now play these games, even on the ODROID-C1.

With Reicast, the Sega Dreamcast got a speedy emulator that allows users to play many, but not all, of the Sega Dreamcast games on ODROID. Even the Sega Saturn, which is known to have a very complicated architecture, runs at the very least on the XU3 and XU4 at a decent speed, with many working titles.

Thanks to emulation, I found a lot of new games for the Sega Genesis/Mega Drive and other Sega systems to enjoy. Animaniacs, Beyond Oasis, Comix Zone, Donald in Maui Mallard, or Monster World IV are just a few of the Sega Genesis games I enjoy. With Keio Flying Squadron, Lunar: The Silver Star, Popful Mail, and especially Snatcher, Sega CD has in its library some really awesome games. I like the fact that many games for the Sega CD are still cartoon- or anime-based and, as such, often have cartoon or anime cut-scene videos which, along with some great music tracks, greatly enhance the experience. Maybe the Sega CD was not a commercial success, but it had some nice games which I enjoy playing to this day.

Although only really playable on the ODROID-XU3/XU4, the Sega Saturn has some great games, including The Legend of Oasis, Elevator Action Returns, Keio Flying Squadron 2, or Radiant Silvergun. Personally, I think it was a failure as a 3D console. Games like Radiant Silvergun or Wipeout show pretty well that 3D was not one of the Sega Saturn's strengths. You can see a lot of dithering, and the overall 3D quality is much worse than on a Playstation or an N64. However, the 2D capabilities were rather good. This could have been a great 2D 32-bit console game with lots of video cut-scenes and improved 2D graphics, rather than trying to push "pseudo 3D" on the console. Some games obviously went this route, but many tried too hard to be 3D console games.

Of course, I still love Dreamcast emulation on ODROID. Crazy Taxi 2, Dead or Alive 2, Evolution 1 and 2, Giga Wing 1 and 2, Grandia II, Ikaruga, Incoming, Kidou Senshi Gundam – Renpou vs. Zeon DX, Phantasy Star Online Ver. 2, Power Stone 2, Rez, Skies of Arcadia, Sonic Adventure 2, Soul Calibur, Star Wars Episode I:Racer, Virtual Tennis 2, and Zero Gunner 2. There are so many awesome games with excellent 3D graphics that I really can't get enough of it. Next to the PSP, I find it's the console with the second most impressive graphics on the ODROID.

Final Thoughts

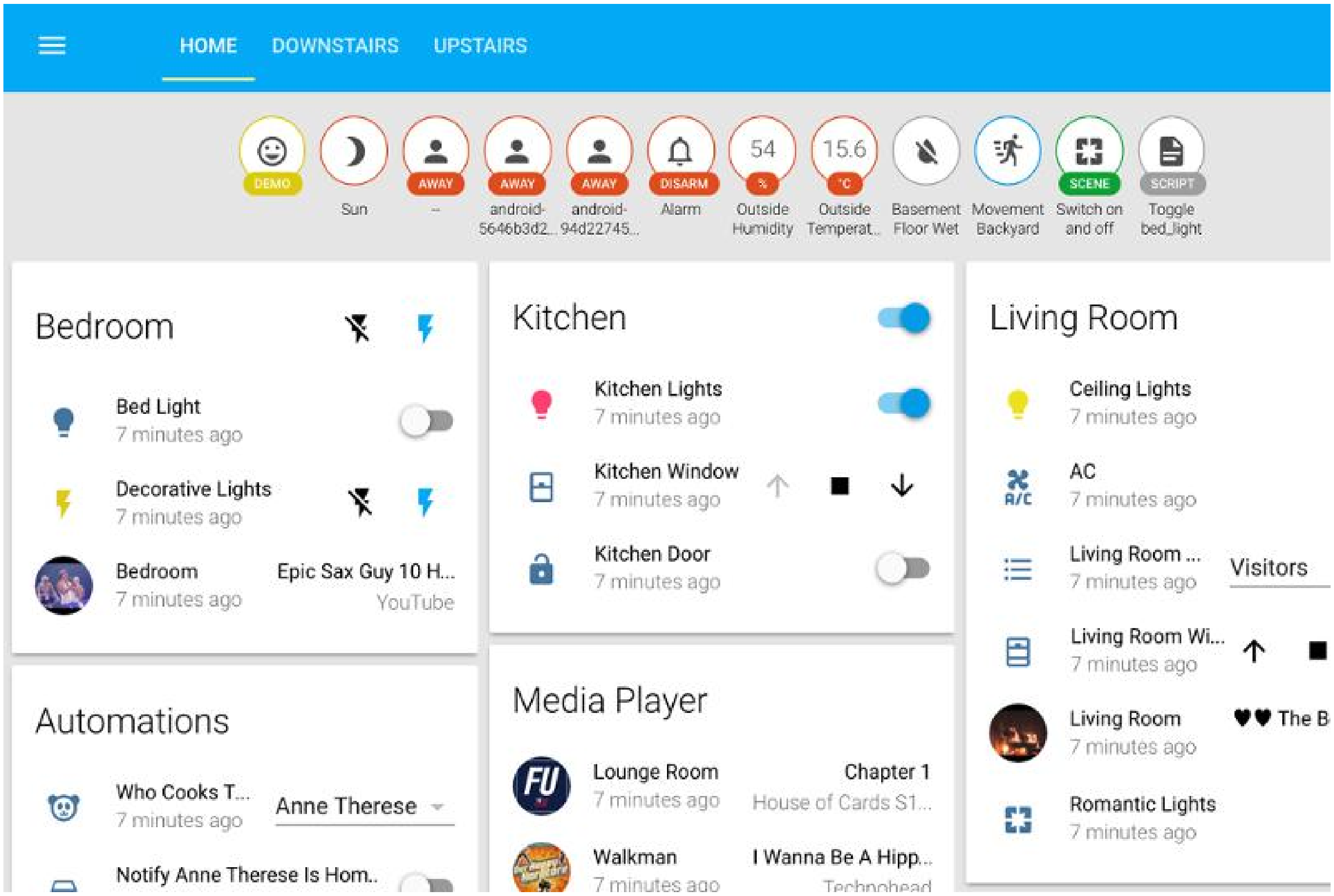
So am I a Sega fanboy? I guess I am, due to my good experiences with Sega, first as a child with the Master System and Game Gear (compared to the NES and Game Boy), and later through the impressive Dreamcast, which I still love to this day and of which I have the most fond memories. Today, we can pick the best games for every Sega console. It may not be as many as for Nintendo, but that's fine with me. I continue to enjoy Sega consoles.

Still, I realize that the gaming libraries of Sega offered less Triple A series compared to the Nintendo consoles, especially when it came to great series such as Super Mario Bros., Pokémon, Zelda, Earthbound, F-Zero, Metroid, and Dragon Quest. However, Sega had his shining stars as well, with Sonic the Hedgehog, Phantasy Star, Golden Axe, Outrun, Virtual Fighter, and Wonderboy.

Sega might not have been the most successful console company, but it created some very impressive consoles for its time, even if some may have been ill-fated. Sega also provided numerous games for arcade systems, and they continue to produce great games today.

Home Assistant: Customization and Automations

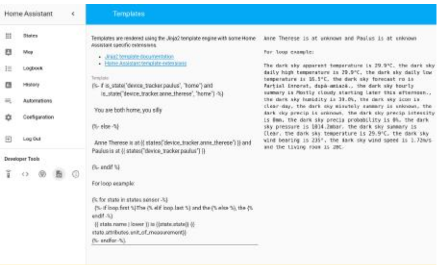
September 1, 2017 By Adrian Popa Linux, Tinkering, Tutorial



In the July 2017 issue of ODROID Magazine, I introduced you to Home Assistant (http://bit.ly/2hlOPOE), which is an open-source home automation platform. Based on the examples listed in that article, this article will discuss advanced topics related to Home Assistant (HA) using in-depth steps. This will allow you to maximize the use of HA, and also help with experimentation.

Working with HA Developer Tools

When accessed, the HA webui looks similar to the one shown in Figure 1a. In the previous article, I mentioned the Developer Tools inside HA. You can access them by going to the left panel (bottom) in the web interface and hovering over the buttons.



Templates are rendered using the Jinja2 template engine with some Home Assistant specific extensions.

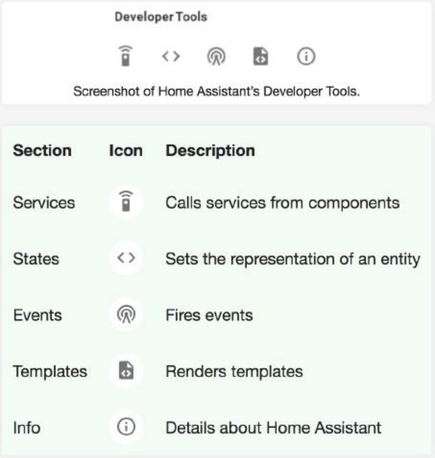
- Jinja2 template documentation
- Home Assistant template extensions

Template

```
{%- if is_state('device_tracker.paulus', 'home') and
    is_state('device_tracker.anne_therese', 'home') -%}

  You are both home, you silly

{%- else -%}

  Anne Therese is at {{ states('device_tracker.anne_therese') }} and
Paulus is at {{ states('device_tracker.paulus') }}

{%- endif %}

For loop example:

{% for state in states.sensor -%}
  {%- if loop.first %}The {% elif loop.last %} and the {% else %}, the {%
endif -%}
  {{ state.name | lower }} is {{state.state}} {{
state.attributes.unit_of_measurement}}
{%- endfor -%}.
```

```
Anne Therese is at unknown and Paulus is at unknown

For loop example:

The dark sky apparent temperature is 29.9°C, the dark sky
daily high temperature is 29.9°C, the dark sky daily low
temperature is 16.5°C, the dark sky forecast ro is
Partial înnorat, după-amiază., the dark sky hourly
summary is Mostly cloudy starting later this afternoon.,
the dark sky humidity is 39.0%, the dark sky icon is
clear-day, the dark sky minutely summary is unknown, the
dark sky precip is unknown, the dark sky precip intensity
is 0mm, the dark sky precip probability is 0%, the dark
sky pressure is 1014.2mbar, the dark sky summary is
Clear, the dark sky temperature is 29.9°C, the dark sky
wind bearing is 235°, the dark sky wind speed is 1.72m/s
and the living room is 28C.
```

Figure 1a – Webui



Screenshot of Home Assistant's Developer Tools.

Section	Icon	Description
Services	🛈	Calls services from components
States	<>	Sets the representation of an entity
Events	◉	Fires events
Templates	▣	Renders templates
Info	ⓘ	Details about Home Assistant

Figure 1b – Working with Developer Tools and Templates

The following tools are available to the developer:

Services: This lets you make calls to a variety of services exposed by your components. You can do things like trigger an automation, hide or show groups, reload HA configuration, control a media player object (play/pause/load playlist), and so on. The available services can change based on what components you have active in your configuration. It is a good place to test some action before adding it to an automation.

States: This lets you override the state of any entity. It also lists all entities, with their current state and attributes. You can use this list to find out an entity name, either to know how to reference it in the configuration (e.g., the entities visible in a view) or to use it in a template.

Events: This lets you generate an event on the event bus. There are several events available, but in practice you may not need to generate some of these events.

Templates: The HA templating engine uses Jinja2 templating syntax (http://bit.ly/2vd497l) with the addition of exposing some internal variables. The templating syntax is more like a programming language, so take your time to read the documentation (http://bit.ly/2vOK7no). The point of templates is to process input or output data, and to format it in a different way. The Templates view gives you a workspace where you can experiment and test the syntax before writing it to the configuration file. When you first load the page, it will have a sample syntax, which among other things, iterates through all your sensors and shows you their current values. For example, this can teach you that you can access a sensor state by calling {{ states.sensor.sensor_name.state }}.

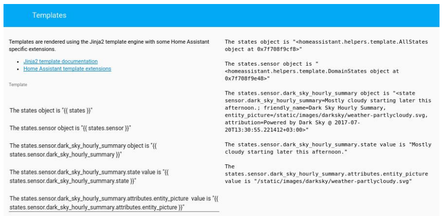
Info: Shows you the current version as well as any errors that have been logged.

In order to better understand the relationship between an entity name and how to use it in a template, let us try an experiment. Let us assume that we need to get the icon of Dark Sky (https://darksky.net) weather forecast. First of all, we need to use the States tool to get the correct entity name. If you search there for the name shown in the web interface "Dark Sky Hourly Summary", then you will find an entity called "sensor.dark_sky_hourly_summary". Most HA entities have a state and may have one or more attributes and those should already be visible in the States view. Now we can switch to the Templates tool and add our own template at the end of the template dialog.

Let us try the following templates and let's see what the output is:

```
The states object is "{{ states }}"

The states.sensor object is "{{
states.sensor }}"

The states.sensor.dark_sky_hourly_summary
object is "{{
states.sensor.dark_sky_hourly_summary }}"

The
states.sensor.dark_sky_hourly_summary.state
value is "{{
states.sensor.dark_sky_hourly_summary.state
}}"

The
states.sensor.dark_sky_hourly_summary.attrib
utes.entity_picture  value is "{{
states.sensor.dark_sky_hourly_summary.attrib
utes.entity_picture }}"
```

The output you receive can be viewed in Figure 2. Some of the data points to Python objects, and some others (like, state and attributes) return string values which you can use. With this information, you are prepared to start writing templates and automations.



Templates

Templates are rendered using the Jinja2 template engine with some Home Assistant specific extensions.

- Jinja2 template documentation
- Home Assistant template extensions

Template

The states object is "{{ states }}"

The states.sensor object is "{{ states.sensor }}"

The states.sensor.dark_sky_hourly_summary object is "{{ states.sensor.dark_sky_hourly_summary }}"

The states.sensor.dark_sky_hourly_summary.state value is "{{ states.sensor.dark_sky_hourly_summary.state }}"

The states.sensor.dark_sky_hourly_summary.attributes.entity_picture value is "{{ states.sensor.dark_sky_hourly_summary.attributes.entity_picture }}"

```
The states object is "<homeassistant.helpers.template.AllStates
object at 0x7f708f9cf8>"

The states.sensor object is "
<homeassistant.helpers.template.DomainStates object at
0x7f708f9e48>"

The states.sensor.dark_sky_hourly_summary object is "<state
sensor.dark_sky_hourly_summary=Mostly cloudy starting later this
afternoon.; friendly_name=Dark Sky Hourly Summary,
entity_picture=/static/images/darksky/weather-partlycloudy.svg,
attribution=Powered by Dark Sky @ 2017-07-
20T13:30:55.221412+03:00>"

The states.sensor.dark_sky_hourly_summary.state value is "Mostly
cloudy starting later this afternoon."

The
states.sensor.dark_sky_hourly_summary.attributes.entity_picture
value is "/static/images/darksky/weather-partlycloudy.svg"
```

Figure 2 – Templates in action

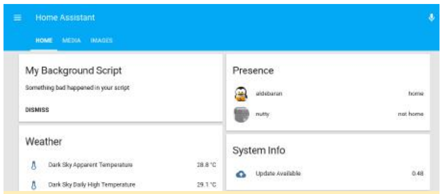
Notification interface and HA API

If you have scripts which run in the background (started by say, cron) you may want to be notified when things go wrong and the script fails for whatever reason. Most tutorials online will show you how to send a notification email or SMS, but for problems which are not too critical, maybe you would not like checking email or being woken up at 3 AM. For this, you can push messages to Home Assistant using curl and its API, so that you can get notifications from your scripts whenever you log into Home Assistant. This way, you get to know what happened if you regularly log into the web interface. A similar approach can be taken to change the states of Home Assistant entities by using external triggers, or you can use the API to query entities from external scripts.

To set this up, you only need to run a shell command from your script when handling an error:

```
$ /usr/bin/curl -X POST -H "x-ha-access:
api_password" -H "Content-Type:
application/json" --data "{"message":
"Something bad happened in your script",
"title": "My background script"}"
http://odroid-
ip:8123/api/services/persistent_notification
/create
```

The command above uses the persistent notification action (**http://bit.ly/2wkVRiW**) called via Home Assistant API. To use it, you will need to provide the "api_password" value and send a json (**http://www.json.org/**) object containing the message and title. Note that JSON mandates that you use the quote mark ("), and not apostrophe (') for quoting. The nice thing is that the notification will be displayed on all views/tabs, so you would not miss it. The result will look like Figure 3.



Home Assistant

HOME MEDIA IMAGES

My Background Script

Something bad happened in your script

DISMISS

Presence

aldebaran	home
nutty	not home

Weather

Dark Sky Apparent Temperature	28.8 °C
Dark Sky Daily High Temperature	29.1 °C

System Info

Update Available	0.48

Figure 3 – Persistent notification

Running external scripts on state change

Suppose, for example, that I wish to get the weather forecast from Dark Sky in Romanian so that it is useful for some non-English speakers. Since Dark Sky does not support Romanian yet, I need to do it myself, which is not a problem, since we can do it with Home Assistant using the technique outlined below.

1. Install a translation program on the ODROID system that can use various online translation services and output the desired language. I used trans (**http://bit.ly/2vcLJDU**):

```
$ sudo wget -O /usr/local/bin/trans
git.io/trans
$ sudo chmod a+x /usr/local/bin/trans
```

Test the program to make sure it works as desired:

```
$ trans -b :ro "My name is my password"
```

2. Set up a new shell command component in Home Assistant (**http://bit.ly/2vOFnhe**) to call the command-line script. The shell component can execute a command and take the output of a template as parameter for the command. When a template is used as a parameter, the command execution is more strict and you are not allowed to use pipes or redirection to file. Should you need to use more complex command lines with pipes and templates, you could add them to a shell script and call the script instead. Fortunately, the trans command supports writing output directly to a file. Make the following changes to configuration.yaml:

```
shell_command:
  translate_weather: '/usr/local/bin/trans -
b :ro "{{
states.sensor.dark_sky_hourly_summary.state
}}" -o /tmp/ha-weather-forecast.txt'
```

The command takes the state of the Dark Sky Hourly Summary sensor and passes it to trans for translation. Practice getting the right state by playing in the Template tool, as we have done before. It outputs the translated text into /tmp/ha-weather-forecast.txt. To run this command manually, log into the Home Assistant web interface, navigate to Developer Tools in the left panel and click on the Services icon. You can call the "shell_command" domain with the translate_weather service and without other parameters. If you check the temporary file, you should see your translated weather forecast.

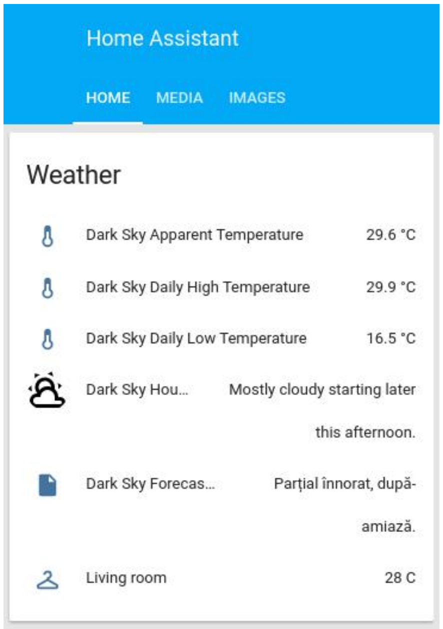
3. Import the translation back into Home Assistant by configuring a file sensor (**http://bit.ly/2x2nmuw**). The file sensor monitors a file for changes and imports the last line into Home Assistant. Make the following changes to your configuration.yaml:

```
sensor:
…
 - platform: file
    file_path: /tmp/ha-weather-forecast.txt
    name: Dark Sky Forecast Ro
```

You should also import this new entity in any views where you wish to use it:

```
group:
…
  weather:
    entities:
…
       - sensor.dark_sky_forecast_ro
```

If you restart Home Assistant, you should see the new item in the Weather group. However, there is still a problem: this entity will never update. We still need to add a trigger so that when the English forecast changes, the translated forecast should change as well. For this, we need to use automation.



Home Assistant

HOME MEDIA IMAGES

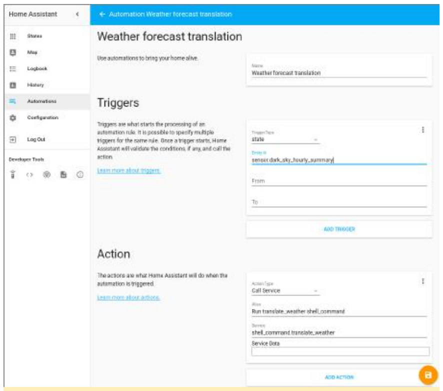
Weather

Dark Sky Apparent Temperature	29.6 °C
Dark Sky Daily High Temperature	29.9 °C
Dark Sky Daily Low Temperature	16.5 °C
Dark Sky Hou...	Mostly cloudy starting later this afternoon.
Dark Sky Forecas...	Parțial înnorat, după-amiază.
Living room	28 C

Figure 4 – The translated forecast next to the original one

4. Create an automation by going to the Automation link in the side panel. Note that you currently need to use the Chrome browser for this step, since other browsers are not supported. Use the "+" button to add an automation, and give it a suggestive name like "Weather forecast translation". The trigger should be "state" and the entity id should match the desired "source" entity, which is, in our case, called "sensor.dark_sky_hourly_summary". Note that we are using the sensor name as it can be found in the States tool. You can leave the "From" and "To" fields blank, since we want it to trigger on any value change.

Next, we need to specify an action or a sequence of actions to be performed when triggered. We need "Call Service" as an action type. The Alias is just a descriptive name for our action and we can call it "Run translate_weather shell_command". The Service field is composed from the whole service call, meaning domain and service name are the same as those used in the Services tool, so in our case it will be "shell_command.translate_weather". The Service Data field can be left blank in our case, since the component does not need additional parameters. You can now click the Save icon and save your automation.



Weather forecast translation

Use automations to bring your home alive.

Name: Weather forecast translation

Triggers

Triggers are what starts the processing of an automation rule. It is possible to specify multiple triggers for the same rule. Once a trigger starts, Home Assistant will validate the conditions, if any, and call the action.

Learn more about triggers.

Trigger Type: state

Entity Id: sensor.dark_sky_hourly_summary

From

To

Action

The actions are what Home Assistant will do when the automation is triggered.

Learn more about actions.

Action Type: Call Service

Alias: Run translate_weather shell_command

Service: shell_command.translate_weather

Service Data

Figure 5 – Create a new automation

Now, when the weather forecast changes, it will trigger the automation and cause the forecast to be translated and saved in a different entity, except there is still a problem. When you restart Home Assistant, the weather state might not change for a long time and your translation might return “unknown” until the first transition. To fix this, we will run a second automation on Home Assistant startup to update the translation. This time the trigger will be platform homeassistant with event start. The action will be the same as the previous automation. Unfortunately, the web UI does not yet support this platform, so we will have to edit the file manually.

All automations are saved in ~/homeassistant/.homeassistant/automations.yaml. In this case, you would need to add the following to it:

```
- action:
  - alias: Run translate_weather
shell_command
    service: shell_command.translate_weather
  alias: Update weather translation on
startup
  id: '1502097058891'
  trigger:
    platform: homeassistant
    event: start
```

If you look at the automation you’ve already added through the user interface, you’ll see a very similar syntax. The only thing new is the id. This is simply the current UNIX timestamp, and needs to be unique for your system (you can get a new one with date +%s). Once this is configured, after you restart Home Assistant, you will get the translated state shortly. Starting with Home Assistant version 0.51, there is a simpler, but less efficient, way of doing the same thing. You could have a command-line sensor (**http://bit.ly/2uUIQw3**) with a templated parameter, like this:

```
sensor:
…
  - platform: command_line
    command:  /usr/local/bin/trans -b :ro "
{{
states.sensor.dark_sky_hourly_summary.state
}}"
    friendly_name: Dark Sky Forecast
Romanian
```

The reason why this is less efficient than the first solution is that the sensor is polled frequently, and is translated every time. So, in this particular case, you may run into quota problems with the translation providers.

Toggling a system service from Home Assistant

Let’s explore a new use-case. Suppose you have a system service running on your ODROID that you want to turn on/off from Home Assistant. In my case, such a service would be Mycroft (**https://mycroft.ai**), because it uses some resources when idle and can get confused by ambient sounds when I am watching a movie. There are more details about Mycroft at **http://bit.ly/2tt3crC**. The point is that you can use commands such as service mycroft start to control the service. You are not limited to services; you could toggle anything on or off.

To control it from Home Assistant we can use the command line switch component (**http://bit.ly/2wdVGW5**). Add the following to your configuration.yaml:

```
switch:
  - platform: command_line
    switches:
      mycroft:
        command_on: sudo /usr/sbin/service
mycroft start
        command_off: sudo /usr/sbin/service
mycroft stop
        command_state: /usr/sbin/service
mycroft status >/dev/null 2>&1
        friendly_name: "Mycroft status"
```

You can also add it to a separate view:

```
group:
…
 switches:
    name: Switches
    view: yes
    entities:
    - switch.mycroft
```

There is one more thing you need to add for this to work. The sudo command will ask for a password by default, so we need to tell sudo that the user homeassistant can run the service command as root without a password. We can do this by running sudo visudo and adding the following line at the end of the file:

```
homeassistant ALL=NOPASSWD:
/usr/sbin/service
```

Let us expand this example a little. Suppose you want to be able to toggle a service running on a different device. The most secure way to do this would be through ssh. In order to do this, we will need to setup keys for ssh, so that homeassistant user can run commands through ssh without being prompted for a password (note that for security’s sake you will need to protect your keys). You will need to run the following steps with the homeassistant user:

1. Create a new ssh key for homeassistant with no password:

```
$ sudo su -s /bin/bash homeassistant
$ cd ~homeassistant
$ ssh-keygen -t rsa
```

Accept the default values (key stored in /home/homeassistant/.ssh/id_rsa, and no passphrase). You can use this key to control many devices (including using it to login on routers for presence detection), so there is no need to create multiple keys.

2. Copy the key to the remote system. Make sure that you input the correct password for the account you are connecting as (I am using root on the remote device):

```
$ ssh-copy-id root@other-device-ip
```

3. Test the connection manually:

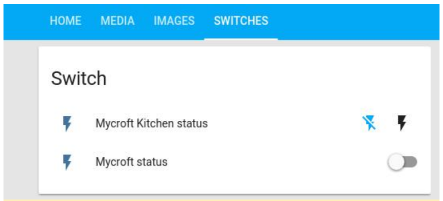
```
$ ssh root@other-device-ip hostname
```

You should receive one line with the other device’s hostname, without being prompted for a password. If you get this, it means it is working. If not, you can find an awesome troubleshooting guide at **http://bit.ly/2vTQYdA**.

4. Configure it in Home Assistant by adding a new switch entry in configuration.yaml:

```
switch:
  - platform: command_line
    switches:
…
      mycroft_kitchen:
        command_on: ssh root@kitchen
/usr/sbin/service mycroft start
        command_off: ssh root@kitchen
/usr/sbin/service mycroft stop
        friendly_name: "Mycroft Kitchen
status"
```

If you do not want the constant polling from Home Assistant for the state, you can omit the “command_state” line and in this case Home Assistant will assume it is off and will keep track only of the changes you make in the user interface. Also, the interface will change from a slider to two icons to activate/deactivate.



Switch

Mycroft Kitchen status

Mycroft status

Figure 6 – Switches for system processes

Toggling a switch based on media playback or presence information

Now that we can manually turn Mycroft on/off (or any switch for that matter), let us make things interesting. I would like to have Mycroft running when I am at home (my phone is connected to the router and detected by the presence detection we have implemented in the previous article) and Kodi is not playing. However, that can be ambiguous. So, let us define what we really want:

- User transitions from “not_home” to “home” and Kodi is idle => turn on Mycroft
- User transitions from “home” to “not_home” => turn off Mycroft
- User is “home” and Kodi transitions from anything to playing => turn off Mycroft
- User is “home” and Kodi transitions from anything to idle => turn on Mycroft

For this, we will create a few automations. What is different from the previous automations will be the use of conditions (**http://bit.ly/2x2FDYz**). Triggers indicate when an action should happen, while conditions are used as filters and say if that action should happen.

So, let us take the first step. My user is tracked by the device called “nutty”. Since the web interface does not support conditions (note: conditions are only supported starting with version 0.51), we will have to do it manually, in the config file automations.yaml:

```
- action:
  - alias: Turn on Mycroft
    service: switch.turn_on
    entity_id:
      - switch.mycroft
  alias: Turn on Mycroft when Nutty arrives
home and Kodi is idle
  id: '1502097058892'
  trigger:
    platform: state
    entity_id: device_tracker.nutty
    to: 'home'
  condition:
    condition: and
    conditions:
      - condition: state
        entity_id:
'media_player.kodi_livingroom'
        state: 'idle'
```

The automation is triggered and evaluated each time the entity "device_tracker.nutty" changes state. When it is triggered, the condition is evaluated as well and if "media_player.kodi_livingroom" is idle at that time, then the action is executed and the switch is turned on. I could have also tested that Mycroft is off, but turning on an already-on switch has no negative side effects.

If that is difficult to follow, here is the pseudo-code:

```
onStateChange(device_tracker.nutty):
  if states.device_tracker.nutty.state ==
'home':
    if
states.media_player.kodi_livingroom.state ==
'idle':
      switch.turn_on(switch.mycroft)
```

The off automation looks similar, but is simpler since it does not have an extra condition:

```
- action:
  - alias: Turn off Mycroft
    service: switch.turn_off
    entity_id:
      - switch.mycroft
  alias: Turn off Mycroft when Nutty leaves
home
  id: '1502097058893'
  trigger:
    platform: state
    entity_id: device_tracker.nutty
    to: 'not_home'
```

The last two automations should be triggered by Kodi state changes and use conditions to test if the user is home or not.

```
- action:
  - alias: Turn off Mycroft
    service: switch.turn_off
    entity_id:
      - switch.mycroft
  alias: Turn off Mycroft when Kodi is
playing and Nutty is home
```

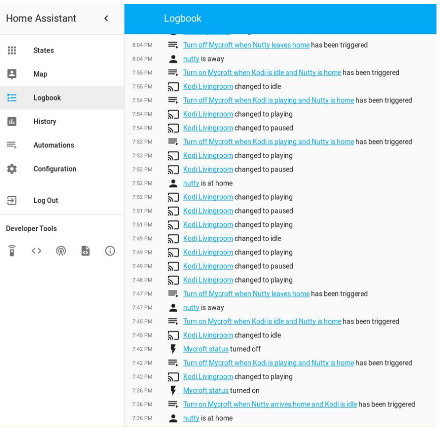
```
  id: '1502097058894'
  trigger:
    platform: state
    entity_id: media_player.kodi_livingroom
    to: 'playing'
  condition:
    condition: and
    conditions:
      - condition: state
        entity_id: 'device_tracker.nutty'
        state: 'home'
```

And the last one should be:

```
- action:
  - alias: Turn on Mycroft
    service: switch.turn_on
    entity_id:
      - switch.mycroft
  alias: Turn on Mycroft when Kodi is idle
and Nutty is home
  id: '1502097058895'
  trigger:
    platform: state
    entity_id: media_player.kodi_livingroom
    to: 'idle'
  condition:
    condition: and
    conditions:
      - condition: state
        entity_id: 'device_tracker.nutty'
        state: 'home'
```

Once you are done editing automations.yaml, you can reload the automations directly from Home Assistant by going to the "Configuration" view and selecting "Reload Automation".

You should now test the automations by triggering them and checking the result in all the cases to rule out any bugs. You can use the Logbook view to see when automations have been triggered.



Logbook

```
8:04 PM  Turn off Mycroft when Nutty leaves home has been triggered
8:04 PM  nutty is away
7:55 PM  Turn on Mycroft when Kodi is idle and Nutty is home has been triggered
7:55 PM  Kodi Livingroom changed to idle
7:54 PM  Turn off Mycroft when Kodi is playing and Nutty is home has been triggered
7:54 PM  Kodi Livingroom changed to playing
7:54 PM  Kodi Livingroom changed to paused
7:53 PM  Turn off Mycroft when Kodi is playing and Nutty is home has been triggered
7:53 PM  Kodi Livingroom changed to playing
7:53 PM  Kodi Livingroom changed to paused
7:52 PM  nutty is at home
7:52 PM  Kodi Livingroom changed to playing
7:51 PM  Kodi Livingroom changed to paused
7:51 PM  Kodi Livingroom changed to playing
7:49 PM  Kodi Livingroom changed to idle
7:49 PM  Kodi Livingroom changed to playing
7:49 PM  Kodi Livingroom changed to paused
7:48 PM  Kodi Livingroom changed to playing
7:47 PM  Turn off Mycroft when Nutty leaves home has been triggered
7:47 PM  nutty is away
7:45 PM  Turn on Mycroft when Kodi is idle and Nutty is home has been triggered
7:45 PM  Kodi Livingroom changed to idle
7:42 PM  Mycroft status turned off
7:42 PM  Turn off Mycroft when Kodi is playing and Nutty is home has been triggered
7:42 PM  Kodi Livingroom changed to playing
7:36 PM  Mycroft status turned on
7:36 PM  Turn on Mycroft when Nutty arrives home and Kodi is idle has been triggered
7:36 PM  nutty is at home
```

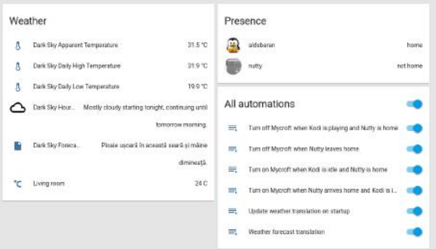
Fig 7 – Logbook viewer

Customize the names and icons

Let us address one more issue. By default, all switches have the "lightning" icon, and maybe you want to use something more appropriate. Also, you may later want to change the friendly name of an entity, and that would change its id and break the automations it is in. There are also some built-in groups – like all the devices managed by a "device_tracker" or all the automations which allows you to enable/disable/manually trigger an automation, but they are hidden by default. In order to make all these changes, we will need to add a Customize section in the beginning of the configuration file, under the homeassistant label, indented by two spaces (http://bit.ly/2x2q6bv).

Let us do the following: display the automations group and change the icons for the switches with something more appropriate. You can use icons from Material Design (http://bit.ly/2wleenC) or your own images. We will make changes to configuration.yaml:

```
homeassistant:
…
  customize:
    group.all_automations:
      hidden: false
      friendly_name: All automations
    switch.mycroft:
      friendly_name: Mycroft living room
      icon: mdi:assistant
    switch.mycroft_kitchen:
      icon: mdi:assistant
    sensor.living_room:
      icon: mdi:temperature-celsius
…
group:
  default_view:
    entities:
…
    - group.all_automations
```



Dark Sky Apparent Temperature	31.5 °C
Dark Sky Daily High Temperature	31.9 °C
Dark Sky Daily Low Temperature	19.9 °C
Dark Sky Hour...	Mostly cloudy starting tonight, continuing until tomorrow morning.
Dark Sky Foreca...	Ploaie ușoară în această seară și mâine dimineață.
Living room	24 C

aldebaran	home
nutty	not home

- Turn off Mycroft when Kodi is playing and Nutty is home
- Turn off Mycroft when Nutty leaves home
- Turn on Mycroft when Kodi is idle and Nutty is home
- Turn on Mycroft when Nutty arrives home and Kodi is i...
- Update weather translation on startup
- Weather forecast translation

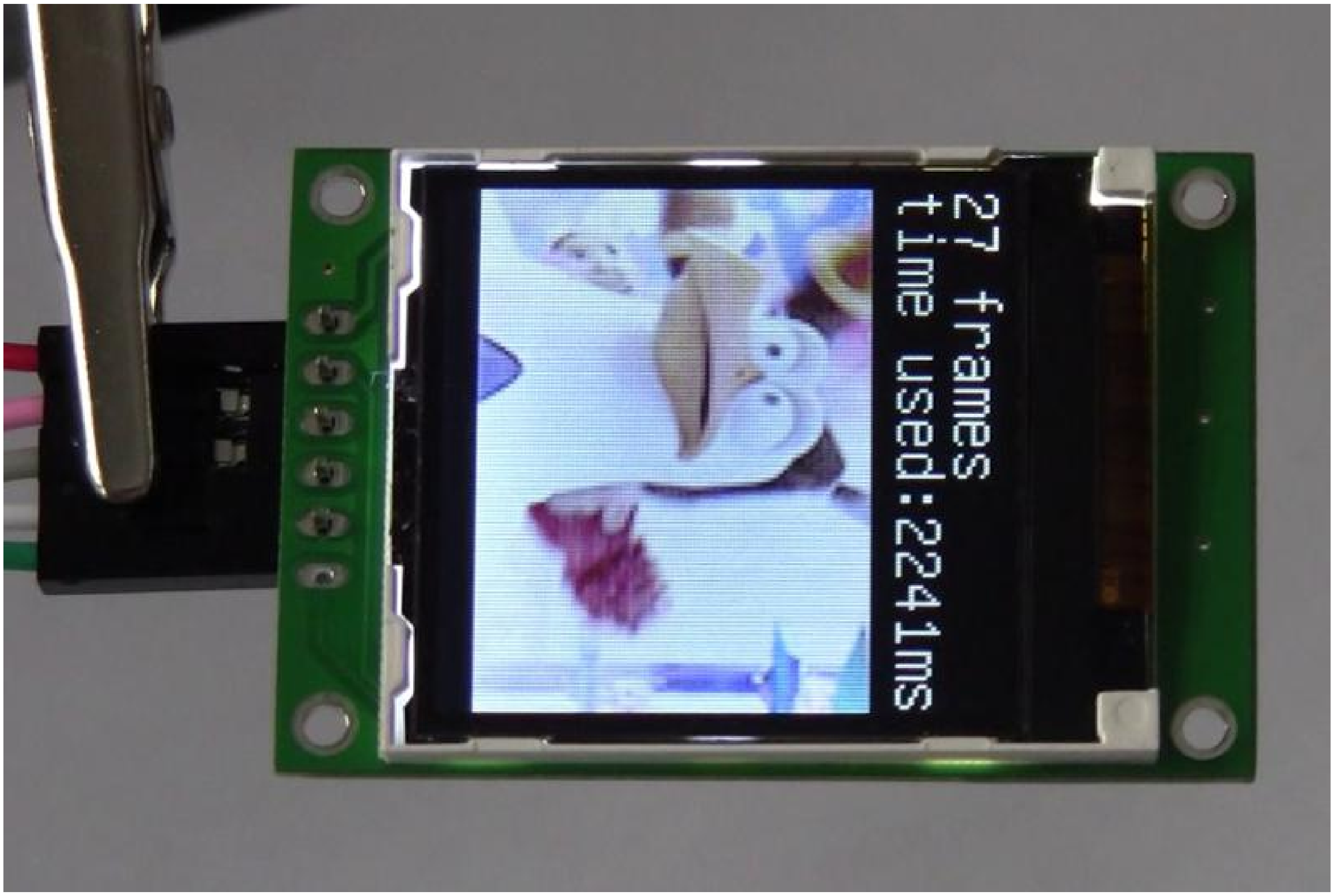
Fig 8 – Customizations for automations and icons

More examples

The Home Assistant community has numerous examples in their cookbook at http://bit.ly/2xfAr2V. There are additional great examples in their forums at http://bit.ly/2v34WbL. For example, an alarm clock sample is available at http://bit.ly/2vOCv48. For further discussions, consult the original thread http://bit.ly/2fVogVu.

Digole Serial Displays: Driving Digole’s Serial Display in UART, I2C, and SPI Modes with an ODROID-C1+

September 1, 2017 By Dennis Chang Linux, ODROID-C1+, Tutorial



Digole.com offers several intelligent serial displays that are controlled through a complete set of high-level proprietary commands. These commands make drawing complex graphics and displaying images and video much easier, offering a offer a layer of abstraction that makes it easy to port their displays to a number of different platforms. Perhaps the most useful is that all of Digole’s models of serial displays are controlled in the same manner, with the same high level command set, and are firmware-upgradeable. The user manual at http://bit.ly/2fXiD9y provides complete documentation of all available commands.

For this article, I used a Digole 1.8 inch Serial UART/I2C/SPI True Color 160×128 OLED Module with 2MB Flash, model number DS160128COLED-46F. This model does not have a backlight or touchscreen like some of the other thin film transistor (TFT) displays. If you buy a different model, you may have to modify the source code in order to change the screen resolution.

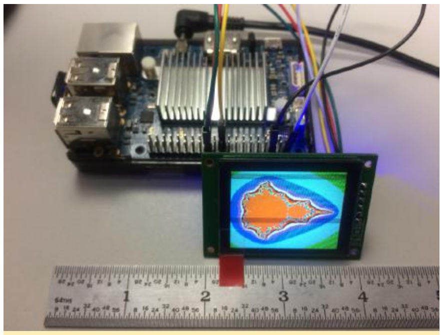


Figure 1 – Dougherty’s mandel.c test in SPI 3-wire mode. Scan lines in photos are not visible to the human eye.

All of the following was done on an ODROID-C1+ running the official Ubuntu 16.04 minimal image and logged in as root.

Initial Test via UART Serial Connection

The Digole serial display ships in UART mode (both SPI and I2C jumpers open). It’s always 8 bits, no parity bit, 1 stop bit. The initial, user-configurable baud rate is 9600.

With the ODROID turned off, wire up the following:

Digole VCC = 5V GPIO pin 2
Digole GND = Ground GPIO pin 6
Digole DATA = TXD1 GPIO pin 8

Power on the ODROID. The Digole should immediately go through its own boot process, which involves an RGB test and ends in a line of text. My display showed “UART baud:9600 V4.1V+2MB Flash”. V4.1V is the firmware version, 2MB Flash is how much flash memory is available on this particular display model. Not all models have flash memory.

Set the UART device’s baud rate:

```
$ stty -F /dev/ttyS2 9600
```

Clear the screen with:

```
$ echo “CL” > /dev/ttyS2
```

For this next command, use single-quotes–not smart-quotes or backticks–so that the terminator is handled correctly:

```
$ echo -n -e ‘TTHello ODROID’ > /dev/ttyS2
$ echo “CL” > /dev/ttyS2
```

Draw a 45px x 45px square:

```
$ echo -n -e ‘DR--’ > /dev/ttyS2
```

Since this is a test, we don’t need to learn Digole’s coordinate syntax right now.

At this point, the display is considered fully functional. It is possible to use Digole’s proprietary commands to completely control the display just by echoing to the UART device. This means one could write an app or game entirely in Bash script or any programming language that can pipe directly to the UART device, including PHP, Perl, Ruby, and Python, although probably with a higher baud rate. With this approach, one can avoid coding in C and using the Digole C library.

We will be testing other methods of serial connection, so shut down the ODROID and remove the power plug to cut power to the Digole display, then remove the Digole's connections to the GPIO pins.

I2C Serial Connection

Using a soldering iron with a tiny conical tip, carefully jump the I2C jumper while leaving the SPI jumper open. It is important not to solder both jumpers by bridging all three pads. This requires a sharp eye or microscope, and a steady hand. With the ODROID turned off, wire up the following:

Digole VCC = 5V GPIO pin 2
Digole GND = Ground GPIO pin 6
Digole DATA = I2CA_SDA GPIO pin 3
Digole CLK = I2CA_SCL GPIO pin 5

Note that the User Manual has diagrams with resistors of 10K or greater between VCC and DATA and VCC and CLK, but the the sample code diagrams on the webpage do not have any resistors. I found that it worked well enough without the resistors, so I did not test to see if the resistors worked.

Next, power up the display. If you soldered the I2C jumper correctly, the startup test will say "I2C address:0x27..." The startup test doesn't appear to know if DATA and CLK are wired up correctly.

Enable I2C on the ODROID by running:

```
$ modprobe aml_i2c
```

To test I2C, we will use the Digole sample C code provided at http://bit.ly/2xh29MJ.

The sample code was written by Javier Sagrera for the Raspberry Pi. We can modify it for the ODROID with a few small changes; nothing critical, just re-naming a few Raspberry Pi references and correcting misspellings, as shown below:

```
// Pin-out using I2C
// ODROID – Digole LCD
// 1: 5v = 5: VCC
// 3: SDA0 = 4: DATA
// 5: SCL0 = 3: CLK
// 6: GND = 1: GND
/*

// Communication set up command
* "SB":Baud (ascII bytes end with
0x00/0x0A/0x0D) -- set UART Baud Rate
* "SI2CA":Address(1 byte <127) -- Set I2C
address, default address is:0x27
* "DC":1/0(1byte) -- set config display
on/off, if set to 1, displayer will display
current commucation setting when power on

// Text Function command
* "CL": -- Clear screen--OK
* "CS":1/0 (1 byte)-- Cursor on/off
* "TP":x(1 byte) y(1 byte) -- set text
position
* "TT":string(bytes) end with 0x00/0x0A/0x0D
-- display string under regular mode

// Graphic function command
* "GP":x(1byte) y(1byte) -- set current
graphic position
* "DM":"C/!/~/&/|/^"(ASCII 1byte) -- set
drawing mode--C="Copy",! and ~ = "Not", & =
"And", | = "Or", ^ = "Xor"
* "SC":1/0 (1byte) -- set draw color--only 1
and 0
* "LN":x0(1byte) y0(1byte) x1(1byte)
y2(1byte)--draw line from x0,y0 to x1,y1,set
new pot to x1,y1
* "LT":x(1byte) y(1byte) -- draw line from
current pos to x,y
* "CC":x(1byte) y(1byte) ratio(byte) -- draw
circle at x,y with ratio
* "DP":x(1byte) y(1byte) Color(1byte) --
draw a pixel--OK
* "DR":x0(1byte) y0(1byte) x1(1byte)
y2(1byte)--draw rectangle, top-left:x0,y0;
right-bottom:x1,y1
* "FR":x0(1byte) y0(1byte) x1(1byte)
y2(1byte)--draw filled rectangle, top-
left:x0,y0; right-bottom:x1,y1
*/

#include < stdlib.h >
#include < linux/i2c-dev.h >
#include < fcntl.h >
#include < string.h >
#include < sys/ioctl.h >
#include < sys/types.h >
#include < sys/stat.h >
#include < unistd.h >

int main(int argc, char **argv)
{
   int fd;
   char *fileName = "/dev/i2c-1";      //
Name of the port we will be using
   int  address = 0x27;                //
Address of I2C device
   char buf[100];

   if ((fd = open (fileName, O_RDWR)) < 0) {
// Open port for reading and writing
       printf("Failed to open i2c port
");
       exit(1);
   }

   if (ioctl(fd, I2C_SLAVE, address) < 0) {
// Set the port options and set the address
of the device printf("Unable to get bus
access to talk to slave
"); exit(1); } if (argc>1) {
       sprintf(buf,argv[1]);
       //printf("%s %d %s
",buf,strlen(buf),buf[strlen(buf)]);
       if ((write(fd, buf, strlen(buf)+1))
!= strlen(buf)+1) {
           printf("Error writing to i2c
slave
");
           exit(1);
       }
   } else {
       printf(" Simple tool to send commands
to Digole graphic adapter
examples:
");
       printf(" digolei2ctest "CLTTHello
ODROID" - Clear the screen (CL) and prints
"Hello ODROID" (TT)
");
       printf(" digolei2ctest "CC002" -
Draws a circle at x=30 (0), y=30 (0) with a
radius of 32 (2)
");    //not for Character LCD
   }

   return 0;
}
```

Save the above source code as digolei2ctest.c, then compile it:

```
$ gcc -o digolei2ctest digolei2ctest.c
```

You can then run it to send commands (several are provided in the comments):

```
$ ./digolei2ctest "CLTTHello ODROID"
$ ./digolei2ctest "CC002"
```

Again, you can use every high-level command available in the User Manual.

Note: I2C is the only means of communicating with the Digole serial display that is capable of two-way communication. Considering that we are only drawing on the display, receive capability is not necessary, but I2C is probably required for touchscreen access.

Next, we will try the SPI method of communication. This is the fastest, but most complicated, of available serial methods. Once again, shut down the ODROID and remove the power plug to turn off the Digole display, then disconnect the connections between the ODROID and the Digole display.

SPI 3-Wire Serial Connection

Using a soldering iron with tiny conical tip, carefully desolder the I2C jumper and replace it by soldering the SPI jumper instead. Again, make sure not to solder both jumpers by bridging all three pads. With the ODROID turned off, wire up the following:

Digole VCC = 5V GPIO pin 2
Digole GND = Ground GPIO pin 6
Digole DATA = MOSI_PWM1 GPIO pin 19
Digole CLK = SPI_SCLK GPIO pin 23
Digole SS = SPI_CEN0 GPIO (#117) pin 24

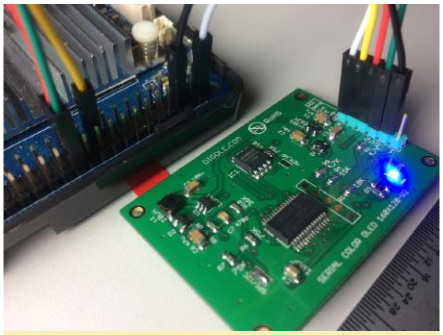


Figure 2 – SPI wiring up close

Power on the ODROID. The Digole startup text should start with "SPI Mode:0..." if you soldered the SPI jumper correctly. It does not seem to know whether DATA, CLK, or SS are wired correctly.

Note that Digole states in the manual that SPI mode has the additional requirement of a "special handshake" to "clock out data." Check the "SPI transceiver data flow chart" at the end of the Port Connection section of the Digole serial display User Manual for details. For SPI testing, we will be using James F. Dougherty's driver and sample code at http://bit.ly/2wmyPIi.

This script is also written for the Raspberry Pi, but works without modification on the ODROID-C1+. The only difference is the pinout: connect the Digole SS pin to GPIO pin 24 on the ODROID-C1+ instead of GPIO pin 26 on a Raspberry Pi.

Enable SPI on the ODROID:

```
$ modprobe spicc
```

Then obtain and build Dougherty's SPI driver:

```
$ git clone
https://github.com/jafrado/digole.git
$ cd digole
$ make
```

Run the included test code:

```
$ ./oledtest /dev/spidev0.0
```

You should see the display start with an image of a compass followed by many test screens. Don't worry about the slow drawing speed, since a way to increase it will be described in the next section. Try the other sample program to display a Mandelbrot fractal:

```
$ ./mandel /dev/spidev0.0
```

There are other sample programs, but they appear to be slightly buggy and tend to draw their graphics in unexpected positions. At this point, between the oledtest.c and mandel.c programs, you should have all you need to start building your own apps that use the Digole serial displays.

Performance Considerations

In Dougherty's code, change the spi_speed value at rpi_spi.c line 41 from 200,000 to 1,000,000 (1MHz) to increase the speed at which the images are displayed on the screen. Going faster than 1MHz breaks the drawing commands badly on my ODROID-C1+. Dougherty commented in the code that he was not able to go faster than 200KHz, but he was using a very slow Raspberry Pi Zero for testing.

Curious about the limitation, I used a simple while loop with a "sleep x" command and varying values of x in order to overwhelm the Digole display by sending sentences of commands too quickly, causing mis-drawn graphics or corrupt images, which is exactly what happened when "spi_speed" value was increased above 1MHz in Dougherty's sample programs. Theoretically, the SPI bus and the Digole display can go much faster than 1MHz, but I suspect that the aforementioned "special handshake" and precise management of SPI communications at the byte and word level will be necessary to achieve maximum performance.

We do know, however, that these displays are capable of performing very well. Digole links to a YouTube video at http://bit.ly/2wfwPRJ showing a fast, smooth video sequence of 27 frames in about 2 seconds, which is approximately 14fps.

Unfortunately, they do not detail in the video how to achieve this speed. The title of the video indicates they are using the relatively new Video Box feature (as of firmware V4.0V) which allows one to write raw image data directly to the display. The User Manual says Video Box runs at "maximum speed: UART mode-460800bps, I2C->400K bps, SPI-10MHz." That's ten times faster than our current best performance using Dougherty's samples in SPI mode. It will probably require contacting Digole tech support to find out how to pull it off.

Regarding the while loop tests, a "sleep 0.05" command appears to be the shortest delay between "TP00TTHello ODROID" sends, resulting in a barely-perceptible flicker of the words "Hello ODROID" being redrawn in place with no errors. For many projects, especially those that update text periodically, 0.05 seconds is plenty fast enough, and one will not have to wrestle with performance tuning of the serial communications.

Conclusion

I'm quite impressed with the Digole serial displays for their multiple connection methods and easy, but powerful, commands. There are many advanced features including stored fonts, stored command sequences, and integrated touchscreen that other, less intelligent, displays simply don't have. Most other touchscreens are a separate device from the display, but the Digole touchscreen is controlled through the same serial interface as the display. The simple fact is that there aren't that many full-color, high-resolution displays in this small of a size, especially in OLED.

I expect that these tiny full-color displays will find their way into many ODROID projects, especially portable, battery-powered projects. This is especially true of the models with resistive touchscreens and the handful of TFT models with dimmable backlights. OLED models do not have a backlight to dim, but dimming can be accomplished by changing the colors to darker shades.

The performance of the Digole serial displays is good enough for most uses without any performance tuning. For games and video where frame rate matters, it is certainly possible to achieve decent performance through managing the serial communications management and by taking advantage of the Digole display's advanced features.

Meet An ODROIDian: Ted Jack-Philippe Nivan (@TedJack)

September 1, 2017 By Ted Jack-Philippe Nivan Meet an ODROIDian

Please tell us a little about yourself.
I'm 25 years old, and was born and raised in Martinique (French West Indies). I'm the Lead Developer of Adok (www.getadok.com) and I operate in both Software and Electrical areas. I currently live in the south of Paris, and hold a Certificate in Electrical and Electronics Engineering from École de Technologie Supérieure (Canada), a Master's degree in Signal and Image Processing from Université Claude Bernard Lyon 1 (France) and a Master's degree in Electrical and Electronics Engineering from Institut national des Sciences appliquées de Lyon (France). I've also done my end-of-study internship at Harvard Medical School in MRI Artifact Quantification of the brain.

End of study at Harvard Medical School

How did you get started with computers?
Well, I've started with computers at the age of 13 and it quickly became a need to play with this incredible piece of hardware. I felt at this time that I had some much power in my hands and I could extend my thoughts through the machine. I was fascinated by both software and hardware materials that the computer was made of. I've been working on various projects ever since and one of them that I'm particularly proud of was turning a bicycle into a motorcycle. It was back in 2012, in my hometown.

Turning a bicycle to a motorcycle

What attracted you to the ODROID platform?
It is mainly for the product's quality and the community. It is easy to start out with ODROIDs and the platform is well documented.

How do you use your ODROIDs?
I use my ODROIDs mainly for developing Android Apps, Embedded Systems and Learning Linux Kernel Development.

Which ODROID is your favorite and why?
My favorite ODROID is the XU4 because of its power and the community behind it. People like @voodik have been doing such a great job on keeping the platform up-to-date.

What innovations would you like to see in future Hardkernel products?
I would like to see a next-generation board with a system on module design in order to decrease the time to market. In addition, a board with Windows support would be great as well.

What hobbies and interests do you have apart from computers?
I'm an independent music producer/artist who goes by the name of "runthecode". I'm also involved in sports such as tennis and football.

What advice do you have for someone wanting to learn more about programming?
Just get your hands on! The community is so big and still growing up tremendously day after day. The Internet is the place to be. I don't think there's a need to buy books while starting off. Therefore, start as soon as you can, even if you're young. It will pay off in the long term. Do with what you have, and more importantly, if you want to be successful, do not let anyone tell you what to do in life. Cultivate the seed within you, market yourself and get along with the right people. You will get what you deserve . Remember that successful people are not the ones who were talented at the beginning, but the ones who stuck to their beliefs.

Ted is always cooking up something