How to write udev rules

posted Sep 18th 2009 10:00am by Mike Szczys
filed under: linux hacks

Since the adoption of Kernel 2.6, Linux has used the udev system to handle devices such as USB connected peripherals. If you want to change the behavior when you plug something into a USB port, this section is for you. As an example, we will use a USB thumb drive but these methods should translate to any device handled by udev. As a goal for this exercise we decided to create a symlink and execute a script when a specific thumb drive was loaded. The operating system we used for this exercise is Ubuntu 9.04 Jaunty Jackalope.

Background Investigation

Before writing rules, we have to gather all the information needed to identify our device. We found the block device node that the drive was assigned to by plugging it in and checking `/var/log/messages`. We then pass that location (/dev/sdd1) to two commands that we run at the same time. Some distributions use the “udevinfo”

command but with Ubuntu 9.04 the command has changed to “udevadm info”:

```
udevadm info -a -p $(udevadm info -q path -n /dev/sdd1)
```

The output of this is pretty meaty. We need to find the top of the chain that provides the block node which is used for mounting removable storage (in our case, /dev/sdd1). Using this KERNEL as identification will ensure that our symlink points to a mountable block device and not some part of the USB controller. We are also looking for device specific identifiers that differentiate this particular thumbdrive from all others:

```
looking at device '/devices/pci0000:00/0000:00:02.1/usb1/1-2/1-2:1.0/host29/target29:0:0/29:0:0:0/block/sdd/sdd1':
 KERNEL=="sdd1"
 SUBSYSTEM=="block"
 DRIVER==""
 ATTR{partition}=="1"
 ATTR{start}=="63"
 ATTR{size}=="31310622"
 ATTR{stat}==" 208 15448 16282 776 2 0 2 12 0 508 788"

looking at parent device '/devices/pci0000:00/0000:00:02.1/usb1/1-2':
KERNELS=="1-2"
 SUBSYSTEMS=="usb"
 DRIVERS=="usb"
 ATTRS{configuration}==""
 ATTRS{bNumInterfaces}==" 1"
 ATTRS{bConfigurationValue}=="1"
 ATTRS{bmAttributes}=="80"
 ATTRS{bMaxPower}=="200mA"
 ATTRS{urbnum}=="1858"
 ATTRS{idVendor}=="13fe"
 ATTRS{idProduct}=="1f00"
 ATTRS{bcdDevice}=="0110"
 ATTRS{bDeviceClass}=="00"
 ATTRS{bDeviceSubClass}=="00"
 ATTRS{bDeviceProtocol}=="00"
 ATTRS{bNumConfigurations}=="1"
 ATTRS{bMaxPacketSize0}=="64"
 ATTRS{speed}=="480"
 ATTRS{busnum}=="1"
 ATTRS{devnum}=="69"
 ATTRS{version}==" 2.00"
 ATTRS{maxchild}=="0"
 ATTRS{quirks}=="0x0"
 ATTRS{authorized}=="1"
 ATTRS{manufacturer}=="OCZ"
 ATTRS{product}=="DIESEL"
 ATTRS{serial}=="50E6920B000AE8"
```

In writing a udev rule, any of these characteristics can be used as conditions for the rule’s execution. That being said, only properties from **one** parent of the device and from the device itself can be match. Trying to match values from more than one parent in the chain will be invalid and will not work.

The Rule

Rule files are stored in the `/etc/udev/rules.d/` directory. We got some advice from the README in that directory on how to name rule files:

> Files should be named xx-descriptive-name.rules, the xx should be chosen first according to the following sequence points:

< 60 most user rules; if you want to prevent an assignment being overriden by default rules, use the := operator.

these cannot access persistent information such as that from vol_id

< 70 rules that run helpers such as vol_id to populate the udev db

< 90 rules that run other programs (often using information in the udev db)

>=90 rules that should run last

We plan to run a script with this rule so we gave it a name that started with a higher number than our other rules but lower than 90. We used the filename:

```
81-thumbdrive.rules
```

The first part of a udev rule is the matching keys. We will use the KERNEL entry from the very top of the chain as well as the idVendor, idProduct, and serial attributes from the device specific information. This will positively identify this particular thumb drive and ignore all others. The kernel argument uses a question mark as a wild card so that if our drive were mounted on a different node (ie: sda1, sdb1, sdc1, etc.) it could still be identified.

```
1 KERNEL=="sd?1", ATTRS{idVendor}=="13fe", ATTRS{idProduct}=="1f00",
  ATTRS{serial}=="50E6920B000AE8"
```

Now that we have the keys necessary to identify the particular hardware we're looking for we can add assignment arguments. In our case we added two. The first creates a symlink to this device inside of the /dev/ directory. The second executes a script in our home directory:

```
1 SYMLINK+="hackaday", RUN+="/home/mike/notify-plugin.sh 'HackaDay Thumbdrive:' 'Connected as:
  $KERNEL'"
```

Here is the final rule assembled into one line:

```
1 KERNEL=="sd?1", ATTRS{idVendor}=="13fe", ATTRS{idProduct}=="1f00",
  ATTRS{serial}=="50E6920B000AE8", SYMLINK+="hackaday", RUN+="/home/mike/notify-plugin.sh
  'HackaDay Thumbdrive:' 'Connected as: $KERNEL'"
```

We added this as the only line in our rule file and then restarted udev using these commands:

```
sudo nano /etc/udev/rules.d/81-thumbdrive.rules
sudo /etc/init.d/udev restart
```

The Script (and the bug workaround)

We wanted to use the pop-up notification we covered a while back but couldn't get it to work. After a bit of frustration we found out that the notify-send package has trouble putting notifications on a user's screen when called from a script run by root. There is a workaround for this bug. We altered the script just a bit for our purposes and pasted it to a new file named: /usr/local/bin/alt-notify-send

```
01 #!/bin/sh
02 user=`whoami`
03 pids=`pgrep -u $user gnome-panel`
```

```
title=$1
text=$2
timeout=$3
icon=$4

if [ -z "$title" ]; then
 echo You need to give me a title >&2
 exit 1
fi
if [ -z "$text" ]; then
 text=$title
fi
if [ -z "$timeout" ]; then
 timeout=60000
fi

for pid in $pids; do
 # find DBUS session bus for this session
 DBUS_SESSION_BUS_ADDRESS=`grep -z DBUS_SESSION_BUS_ADDRESS \
 /proc/$pid/environ | sed -e 's/DBUS_SESSION_BUS_ADDRESS=//'`
 # use it

 #icon hack:
 if [ -z $icon ]; then
 DBUS_SESSION_BUS_ADDRESS=$DBUS_SESSION_BUS_ADDRESS \
 notify-send -u low -t $timeout "$title" "$text"
 else
 DBUS_SESSION_BUS_ADDRESS=$DBUS_SESSION_BUS_ADDRESS \
 notify-send -u low -t $timeout -i "$icon" "$title" "$text"
 fi
done
```

We then created the script that the udev rule calls. This is placed in our home directory at /home/mike/notify-plugin.sh

```
#!/bin/bash

su mike alt-notify-send "$1" "$2" 6000 "/home/mike/hackaday_icon.png"
```

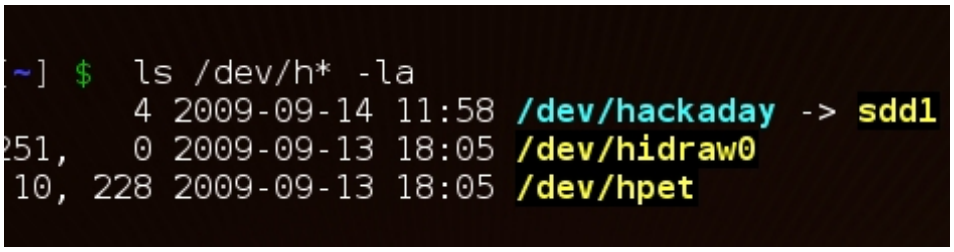
The script can do just about anything we want it to. In this case it calls the notification workaround script passing two strings from the udev rule, a delay time, and an icon to display with the pop-up.

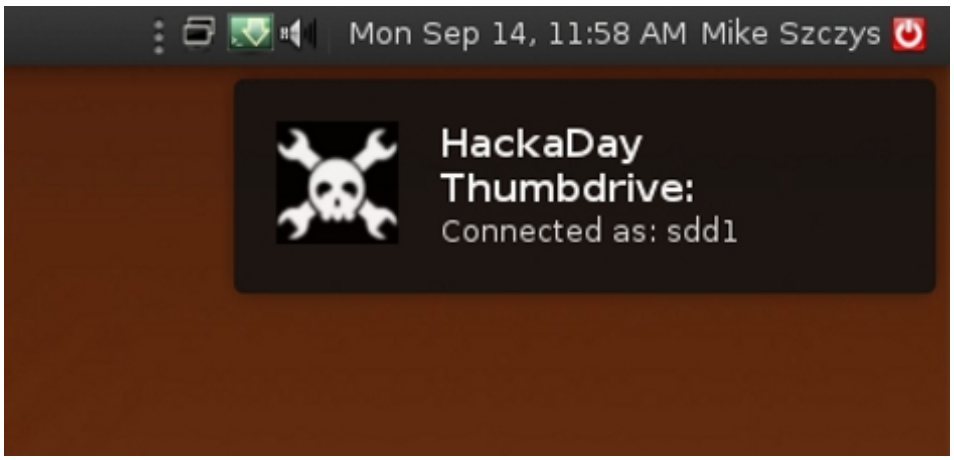
Order of events:

Now that everything's in place, let's take a look at what happens when our drive is plugged in.

1. -USB drive is plugged into the computer
2. -Udev checks the /etc/udev/rules.d/ directory and starts using the rule files in order
3. -Udev gets to our file: 81-thumbdrive.rules and matches the "sd?1″ kernel, idVendor, idProduct, and serial number of the thumbdrive.
4. -If udev confirms a match on our four conditions, a symlink is created at /dev/hackaday and the /home/mike/notify-plugin.sh script is executed, passing it a message that includes the kernel information.
5. -Our script executes creating a pop-up notification using the alt-notify-send workaround.
6. -HAL takes over, automatically mounting our drive (this is part of Ubuntu's removable storage handling and unrelated to our udev rule).

Here we see the symlink pointing to our block device and the pop-up notification:





Other uses:

Udev rules give you control over the hardware attached to your machine. If you are working on a USB connected project, these rules will allow you to set permissions for access, execute scripts when added or removed, and provide a persistent symlink for accessing this hardware without relying on the same node name each time the device is connected. We use a udev rule to allow avrdude to access our AVR Dragon programmer without root permission. In this case our rule sets read/write permissions for owner and group, then assigns the device to the "plugdev" group. As long as the user trying to run avrdude is a member of the plugdev group the program will be able to access the dragon. Here's the rule:

```
1 SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2107", MODE="0660", GROUP="plugdev"
```

We hope this helps clarify how the udev system works. Give it a try with your next project.

Resources:

- Writing udev rules by Daniel Drake: http://reactivated.net/writing_udev_rules.html
- notify-send bug: https://bugs.launchpad.net/ubuntu/+source/libnotify/+bug/160598
- notify-send workaround: http://ubuntuforums.org/showthread.php?p=6889270#post6889270

- Comments [31]

tagged: AVR, block, dragon, hal, linux, notify-send, plugdev, programmer, root, rules, thumb drive, ubuntu, udev, udev rules, usb

31 Responses to How to write udev rules

-
- *medix* says:
September 18, 2009 at 11:00 am

Excellent work Mike! This will come in handy very soon..

Reply Report comment
- *nave.notnilc* says:
September 18, 2009 at 11:18 am

now /this/ is a neat post. what sort of practical stuff would one do with this?
backup script triggered by plugging in the backup medium?

Reply Report comment
- *Mike Szczys* says:
September 18, 2009 at 11:45 am

@nave.notnilc: Indeed, a lot of folks use programs like rsync to do this. Setup a script to run rsync with the settings you want and then execute the script with a udev rule.

Reply Report comment
- *anon* says:
September 18, 2009 at 11:49 am

@nave: that's what I was thinking, plug and play backup.

Reply Report comment
- *Marl* says:
September 18, 2009 at 1:33 pm

Meanwhile on every other operating system you just pick the option you want from the window that pops up when you plug the device in.

WTG Linux.

Reply Report comment

- *Adam Ziegler* says:

September 18, 2009 at 2:29 pm

@Marl… fact is many linux distos work that way also… but if you want more control now you have it.

Reply Report comment

- *Scott* says:

September 18, 2009 at 2:34 pm

Anyone know if there's a way to leverage this with serial-to-USB converters (such as the PL-2303)? I have a couple old serial devices that I want to identify and handle differently, but I have 2 identical PL-2303′s so they "look" like the same device? Is there some way I could communicate with it and see what it returns to differentiate that the serial device is that is behind the USB-to-serial converter??

Reply Report comment

- *Ragnar* says:

September 18, 2009 at 3:13 pm

@Marl: and if you have an idea yourself what should happen to your not so common USB-Device (e.g. Multimeter, Datalogger, Microcontroller, Homeautomation), then you have a neat way to do this under Linux, or you wait till someone writes something for you under windows, which may never happen. You decide.

Reply Report comment

- *OgreProgrammer* says:

September 18, 2009 at 4:02 pm

Uh, Marl?

Every other operating system lets you select one action for ALL like devices as far as I recall. For example, two identical canon DSLR cameras would detect the same.

That was exactly how my two rocketfish web cams work anyway.

In this system you could set it that way, or they could act uniquely based on serial number.

WTG paying attention.

Reply Report comment

- *Joshua* says:

September 18, 2009 at 4:16 pm

Nice idea! You can use a specific USB-Stick as a Crypto-Key or save storing device with a kind of automount triggered by this stuff. Like a encoded ignition key.
Right USB-stick and right key = more safety!

Or if you connect a WiFi-USB to start it as a "Kismet drone".

Reply Report comment

- *Jimmy* says:
September 18, 2009 at 4:30 pm

@Scott,

You can. Try using the USB serial number.

Reply Report comment
- *bobdole* says:
September 18, 2009 at 4:34 pm

Nice writeup!

I was looking into this a couple months ago, I have a sata dock that I can just drop harddrives into, and I was hoping to make it so instead of mounting the drive it would dd /dev/zero to it, making into a hard drive nuking cradle, but I couldn't get it to work..

Reply Report comment
- *pRoFiT* says:
September 18, 2009 at 5:07 pm

bobdole, haha you running for press on the next election?

Anyways, why would you want to nuke a drive automaticly! that could be dangerous if you didn't want to kill a drive. Are you trying to have something incase the FBI break your door down looking for ilegal MP3′s!

If so i always wanted a super magnet in the doorway so all drives coming out of the room would fry :)

Reply Report comment
- *medix* says:
September 18, 2009 at 5:32 pm

@Jimmy: This can be spoofed with a microcontroller such as the 18F4550 using Microchip's mass storage firmware. Get the serial number from the thumb drive and you're in. This seems like a far more versatile system, since you can specify a set of credentials rather than just one for recognition.

I believe there was a USB Serial Key entry system posted on hackaday awhile back.

Reply Report comment
- *bobdole* says:
September 18, 2009 at 6:42 pm

@pRoFiT:

I work at a computer store and we occasionally use harddrives for cloning or testing, so when we're done we like to at least zero the drive once before reselling it. It'd be nice to just drop it into a cradle and walk away rather than grabbing an unused computer, putting in a DBAN boot disk, booting it up and clicking through menu options. Nothing top secret or anything, a single pass of zeroes should at least make the data unrecoverable by amateur efforts.

Reply Report comment

- *miked* says:
September 18, 2009 at 8:41 pm

I LOLed at the idea of a magnetic door way that destroys electrical devices….I pictured the FBI raiding me and trying to get all my electronics through the window.

Reply Report comment
- *Chad Oliver* says:
September 18, 2009 at 8:57 pm

Hmm … no-one seems to be saying what I'm thinking:

Thank you hack-a-day, that was the type of article I like to read … meaty, useful, original. This is what everyone wants. Keep it up!
You know how one-upon-a-time you did articles on cpld's and etc? …. give us some more in that vein.

Way to go, H-A-D, you're getting it right!

Reply Report comment
- *Noobius* says:
September 19, 2009 at 1:50 am

Finally a decent post. Very usefull and well written. Good job and please post more like this one.

Reply Report comment
- *Greycode* says:
September 19, 2009 at 3:10 pm

Mike I had a hard time reading all of your write up because of the tears I had in my eyes because it makes me proud I kept this link on my homepage. THIS is what I am looking for, you mad skill b*st*rd you! Grats man, grats! Now I need to go get some tissues to wipe my face and blow my nose.

Reply Report comment
- *Camille Goudeseune* says:
September 19, 2009 at 3:57 pm

Yay! Now my prs-505 ebook reader gets fresh copies of the .txt's and .pdf's that organize my life, without doing the manual plug in, mount, run ruby script, unmount, unplug dance.

Reply Report comment
- *tedxrochester* says:
September 21, 2009 at 11:50 am

Excellent! I had seen information of this type years ago, but couldn't find it when I went looking. I had wanted to setup my laptop to automatically launch gpsd & gpsdrive whenever I plugged in my GPS receiver into the USB port.

Of course, I've long since given up on that now that I have an iPhone, but I have no doubt this kind of info will come in handy again! Thanks very much for the research and detail you put into it.

Reply Report comment

- *Frederik* says:
October 9, 2009 at 3:26 pm

Very powerful. How would one go about executing a script this way after the drive is mounted (done automatically)? With the above rules, it's not yet possible to automatically copy something from the disk for instance, as it's only being mounted in step 6 according to the order of events.

Reply Report comment

- *Camille Goudeseune* says:
October 9, 2009 at 4:07 pm

The script that's called at the end can mount the disk and then copy files in either direction. For example, my (ruby) script, invoked as …RUN+="/usr/local/bin/doit.rb", does among other things:

DirInstall = "/media/disk/database/media/books"
FAlreadyMounted = File.exists? DirInstall
if !FAlreadyMounted
`mount /media/disk 2>/dev/null`
sleep 2
end
… copy files to/from DirInstall

Reply Report comment

- *speedlight* says:
February 21, 2010 at 5:44 am

Hi,

I have 2 serial-usb identical device. How can I differentiate them. I can't find any unique attributes. Attached is the diff result of both udevinfo:

diff ttyUSB7 ttyUSB8
8,9c8,9
< looking at device '/devices/pci0000:00/0000:00:1d.7/usb1/1-6/1-6.4/1-6.4.4/1-6.4.4:1.0/ttyUSB7/tty/ttyUSB7':
looking at device '/devices/pci0000:00/0000:00:1d.7/usb1/1-6/1-6.4/1-6.4.3/1-6.4.3:1.0/ttyUSB8/tty/ttyUSB8′:
> KERNEL=="ttyUSB8″
12c12
ATTR{dev}=="188:8″
14c14
looking at parent device '/devices/pci0000:00/0000:00:1d.7/usb1/1-6/1-6.4/1-6.4.3/1-6.4.3:1.0/ttyUSB8/tty':
19,20c19,20
< looking at parent device '/devices/pci0000:00/0000:00:1d.7/usb1/1-6/1-6.4/1-6.4.4/1-6.4.4:1.0/ttyUSB7':
looking at parent device '/devices/pci0000:00/0000:00:1d.7/usb1/1-6/1-6.4/1-6.4.3/1-6.4.3:1.0/ttyUSB8′:
> KERNELS=="ttyUSB8″
25,26c25,26
< looking at parent device '/devices/pci0000:00/0000:00:1d.7/usb1/1-6/1-6.4/1-6.4.4/1-6.4.4:1.0':
looking at parent device '/devices/pci0000:00/0000:00:1d.7/usb1/1-6/1-6.4/1-6.4.3/1-6.4.3:1.0′:
> KERNELS=="1-6.4.3:1.0″
37,38c37,38
< looking at parent device '/devices/pci0000:00/0000:00:1d.7/usb1/1-6/1-6.4/1-6.4.4':

looking at parent device ‘/devices/pci0000:00/0000:00:1d.7/usb1/1-6/1-6.4/1-6.4.3′:
> KERNELS==”1-6.4.3″
41c41
ATTRS{dev}==”189:15″
47c47
ATTRS{urbnum}==”36″
58c58
ATTRS{devnum}==”16″
76c76
ATTRS{urbnum}==”77″

Thanks,

speedlight

Reply Report comment

- *Eric* says:

November 8, 2010 at 6:45 am

Thanks for the post !

Reply Report comment

- *graf* says:

June 3, 2011 at 6:46 am

Thanks for the great article, though I’ve not yet succeeded in getting my automatic updates done – my udev rules are matched 13 times, and i dont know how to get it matched only a single time.
Anyways I’ve found an cleaner workaround for the notify-send problem, just do it like this

DISPLAY=:0 /bin/su username -c ‘notify-send Oink!’

works for me (although I’m using e-notify-send, but it has the same issues).

Reply Report comment

- *LinuxNoob* says:

June 25, 2011 at 3:58 am

I was trying your method.
This is the udev rules file
ATTRS{bDeviceClass}==”00″,ATTRS{bInterfaceClass}==”08″,SYMLINK+=”USB MASS STORAGE”,RUN+=”/home/hello/notification.sh”

#!/bin/sh
notify-send “Drive Detected” -t 10000
When I insert my flash drive,the script doesnt run.

Reply Report comment

- *Mark Watson* says:
November 27, 2011 at 11:49 am

I was tearing my hair out with udev. This article was very helpful. I could not get udev to write any debug into to syslog and found another way to get the debug messages was (as root):
i) /etc/init.d/udev stop
ii) udevd –debug
(this runs in the foreground and logs lots of useful stuff on connect/disconnect on rules run/rule files edited etc)
Once debugged stop this and switch it back on:
iii) /etc/init.d/udev start

Reply Report comment

- *Nik* says:
December 6, 2011 at 8:46 am

This is just what I needed. I have a few USB gizmos that I want to attach, that Windows recognizes as thumb drives or serial devices, but my debian box is clueless about. Most of the udev articles I;ve read seem to written with the assumption that you already know how to write udev rules. This article answered my two main questions: how to find the info to identify the device, and how udev fits into the scheme of things.

Reply Report comment

- *Justin M. Bennett* says:
May 9, 2012 at 2:29 am

Very useful post, including the referenced notify-send workaround.

I had to make a slight change to the /usr/local/bin/alt-notify-send script, in using XFCE4 instead of Gnome3:

```
pids=`pgrep -u $user gnome-panel`
```

Becomes

```
pids=`pgrep -u $user xfce4-panel`
```

Now to code some rsync for synchonizing my Android's pictures to my computer, and pushing a music folder to the Android when plugged in.

Reply Report comment

- *dejot* says:
October 3, 2012 at 2:44 am

thanks a lot for the tutorial. one mistake i found cost me some nerves: string substitutions in udev must not be capitalized – use $kernel or %k instead of $KERNEL, as described.

Reply Report comment

Leave a Reply

Name (required)

Mail (will not be published) (required)

Website

XHTML: You can use these tags: <a href="" title="" rel=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <pre> <del datetime=""> <i> <q cite=""> <strike>

Submit Comment

Notify me of follow-up comments via email.

Notify me of new posts via email.

Hack a Day serves up fresh hacks each day, every day from around the web as well as hacking related news.

Send us your hacks

Search

Hacks

- 3d Printer hacks (57)
- android hacks (216)
- arduino hacks (959)
- ARM (27)
- Ask Hackaday (20)
- ATtiny hacks (39)
- beer hacks (53)
- blackberry hacks (8)
- cellphones hacks (328)
- chemistry hacks (87)
- classic hacks (818)
- clock hacks (166)
- cnc hacks (316)
- computer hacks (361)
- cons (188)
- contests (128)
- cooking hacks (57)
- digital audio hacks (394)
- digital cameras hacks (389)
- downloads hacks (101)
- drone hacks (3)
- Engine Hacks (25)
- Featured (41)
- firefox hacks (23)
- g1 hacks (26)
- google hacks (43)
- gps hacks (101)
- green hacks (135)
- Hackaday links (125)
- Hackerspaces (82)
- HackIt (101)
- handhelds hacks (186)
- hardware (205)

- Holiday Hacks (76)
- home entertainment hacks (496)
- home hacks (658)
- how-to (131)
- Interviews (9)
- iphone hacks (167)
- ipod hacks (142)
- kickstarter (24)
- Kindle hacks (13)
- Kinect hacks (79)
- laptops hacks (107)
- laser hacks (156)
- led hacks (629)
- lifehacks (35)
- linux hacks (112)
- lockpicking hacks (11)
- macs hacks (130)
- Medical hacks (90)
- Microcontrollers (450)
- misc hacks (1708)
- multitouch hacks (88)
- musical hacks (251)
- netbook hacks (50)
- news (1049)
- nintendo ds hacks (33)
- nintendo gameboy hacks (84)
- nintendo hacks (246)
- nintendo wii hacks (76)
- Nook Hacks (1)
- palm pre hacks (6)
- parts (63)
- peripherals hacks (631)
- phone hacks (16)
- playstation hacks (119)
- podcasts (8)
- portable audio hacks (66)
- portable video hacks (57)
- psp hacks (47)
- radio hacks (111)
- rants (25)
- Raspberry Pi (79)
- repair hacks (90)
- reviews (17)
- robots hacks (836)
- roundup (35)
- security hacks (422)
- Software Development (82)
- software hacks (34)
- solar hacks (41)

- tablet pcs hacks (31)
- teardown (25)
- tool hacks (585)
- toy hacks (326)
- transportation hacks (444)
- Uncategorized (332)
- video hacks (190)
- Virtual Reality (4)
- weapons hacks (44)
- wearable hacks (168)
- Weekly roundup (18)
- wireless hacks (310)
- xbox hacks (129)

Resources

- Send us news tips
- Contact us

Most commented on (30 days)

- New Hackaday template coming soon. Here's your chance to make it better! (326)
- Hackaday's official Kickstarter policy (216)
- Tell us what development board you love (188)
- Traffic camera countermeasure (185)
- Bringing Java to the world of microcontrollers (101)
- Find a way to stop robocalls to grab this $50k prize (100)
- The Arduino Due is finally here (81)
- µJ, a Java virtual machine for microcontrollers (75)
- Turning a 600 mil chip to 300 mil (73)
- Gravity bike (72)

Recent comments

- **SuperNurd:** Never going to start with
- **freax:** Am I the only one
- **mur1010:** "allow the wearer to update
- **asdf:** You can defeat automatic plate
- **S:** It's not a digital circuit
- **Standard Mischief:** How about bringing back the
- **Rob:** though this particular project likely
- **Extrano:** http://icydockuk.com/goods.php?id=135 Even hardware based versions exist
- **Rob:** Which is why I defillibrate
- **fonz:** when sampling an async input