Adding an I2C Temperature Sensor Using an Existing Linux Driver

Adding a new device on the I2C bus is pretty easy if a driver already already exists in the Linux kernel. I happened to have a DS1621 left over from a previous project and was going to use wiringpi to connect it, but I started looking through the kernel source and found many devices already have drivers, so why not use one of those. The list of drivers here is pretty big, so if you can find any of the sensors with a driver you can still follow this tutorial. This is a rough draft so send me comments by email: ryan at bitflippersanonymous.

The basic steps:

- Build the Linux Kernel
- Install the module
- Read the temperature from /sys

Build It

Pick up the packages you'll need to configure and build the kernel.

```
sudo apt-get install git gcc  make ncurses-dev vim
```

The I2C device is disabled by default because it uses two of the GPIO pins. You can enable it for now.

```
sudo modprobe i2c-bcm2708
sudo modprobe i2c-dev
```

To load the modules on boot, modify these two files.

```
/etc/modprobe.d/raspi-blacklist.conf
#blacklist i2c-bcm2708 # Comment this out

/etc/modules:
snd-bcm2835
spi-bcm2708
i2c-bcm2708
i2c-dev
```

Cone the Raspberry Pi Linux kernel. This will take a long time and about 1.5G of space total.
Make sure you're up-to-date so your kernel version matches the head version in the repo. A simple uname -a will show the version on your Pi. List the branches and checkout the one that matches. You can check the top Makefile for the version string.

```
git clone git://github.com/raspberrypi/linux.git
cd linux

# Then if you need a different branch
git fetch git://github.com/raspberrypi/linux.git rpi-3.6.y:refs/remotes/origin/rpi-3.6.y git
checkout rpi-3.6.y
# Or just do this
git clone -b rpi-3.6.y git://github.com/raspberrypi/linux.git

head Makefile
VERSION = 3
PATCHLEVEL = 6
SUBLEVEL = 11
```

Next you need to configure your kernel. There are 5 million options more or less, but you can just use the existing config of your running kernel and then enable the few extra drivers you need. Go to device drivers, enable Hardware Monitoring Support <*>, then the driver for the DS1621 as a Module <M>. Then start the build and go get a sandwich for the next 8 hours.

```
cd ~/linux
zcat /proc/config.gz > .config
make menuconfig

grep DS1621 .config
CONFIG_SENSORS_DS1621=m

make -j2
make modules
```

The eLinux.org wiki has some directions on how to build the kernel for your RPi using a different, faster machine by cross compiling. http://elinux.org/RPi_Kernel_Compilation#Ubuntu Setup the build the same and then tell make which compiler to use. You can find a pre-built toolchain from the official repo.

```
cd ~/linux
scp pi@raspberrypi.local:/proc/config.gz .; zcat config.gz > .config
git clone git://github.com/raspberrypi/tools.git ~/tools

CCPREFIX=~/tools/arm-bcm2708/arm-bcm2708hardfp-linux-gnueabi/bin/arm-bcm2708hardfp-linux-gnueabi-
make ARCH=arm CROSS_COMPILE=$CCPREFIX menuconfig
make ARCH=arm CROSS_COMPILE=$CCPREFIX -j8
# copy ./driver/hwmon/*ko back to your RPi
```

Install It

Done? OK. Find your module and insert it into your running kernel. lsmod and dmesg will show you if your driver is loaded.

```
find -name ds1621.ko
sudo insmod ./drivers/hwmon/ds1621.ko
lsmod ; dmesg | tail
```

Details on connecting the DS1621
Connect SDA, SCL, 3.3V power and ground to the RPi. Here's the DS1621 Datasheet. I used super glue to hold it to a little PCB and soldered the tiny pins to a header. No external pullup resisters are needed as they're included internal on RPi SoC and enabled by the i2c driver. Here's the P2_header Pinout for reference.

Install i2cdetect

```
sudo apt-get install i2c-tools
```

Look for address of sensor on the i2c bus. The address you see can vary a little depending on how you connect your sensor. The extra pins on the DS1621 select the i2c address so you can connect multiple sensors on the bus without external enables. I connected all of mine to ground so I see it at address 4f.

```
i2cdetect 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 4f
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Now, we need to tell the i2c controller that there is a new device on the bus and what driver to use to communicate with it. The /sys directory is a filesystem the kernel uses to expose control and status of hardware to userspace.

```
sudo su
echo ds1621 0x4f > /sys/class/i2c-adapter/i2c-1/new_device
```

By default your new device won't be setup next time you reboot. To do that you'll just need to move your modules where the kernel can find them, and tell it to load. Copy the modules to the directory that matches your kernel version. (Use uname -a). This should load your module on boot, but you'll still need to add the device. You could do that with an init script in /etc/init.d. Take a look at how /etc/init.d /urandom works for an example. Your init script could even use modprobe to load the driver.

```
sudo cp ds1621.ko hwmon.ko /lib/modules/3.6.11+/kernel/drivers/hwmon/
sudo echo ds1621 >> /etc/modules
```

Use It

Now the /sys directory is a tree organized by bus, device, driver, etc so your device will show up in multiple places with lots of symlinks. Here are a couple paths all to the same device and the current temperature in mili-degrees C. Your app can just read this file whenever you

want the temperature.

```
/sys/bus/i2c/devices/1-004f
/sys/bus/i2c/drivers/ds1621/1-004f
/sys/devices/platform/bcm2708_i2c.1/i2c-1/1-00
/sys/class/hwmon/hwmon1/device

cat /sys/class/hwmon/hwmon1/device/temp1_input
24000
```

You can also use lm-sensors to read the temperature. Install it with apt-get and run 'sensors'. Note, you don't need to be root. lm-sensors does a file read of the file in /sys/...hwmon and includes a C library API over sysfs if you want a more generic interface.

```
sensors
ds1621-i2c-1-4f
Adapter: bcm2708_i2c.1
temp1:        +24.0°C  (low  = +10.0°C, high = +15.0°C)  ALARM (HIGH)
```

The libudev library http://www.signal11.us/oss/udev/ also wraps up sysfs and looks like you can create callbacks for events like when the temperature changes.