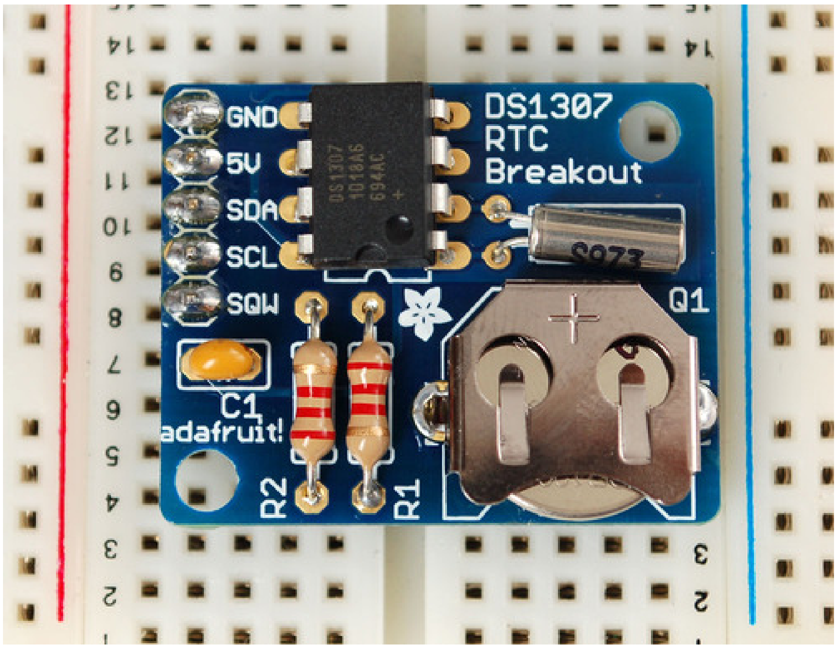
adafruit learning system

Adding a Real Time Clock to Raspberry Pi

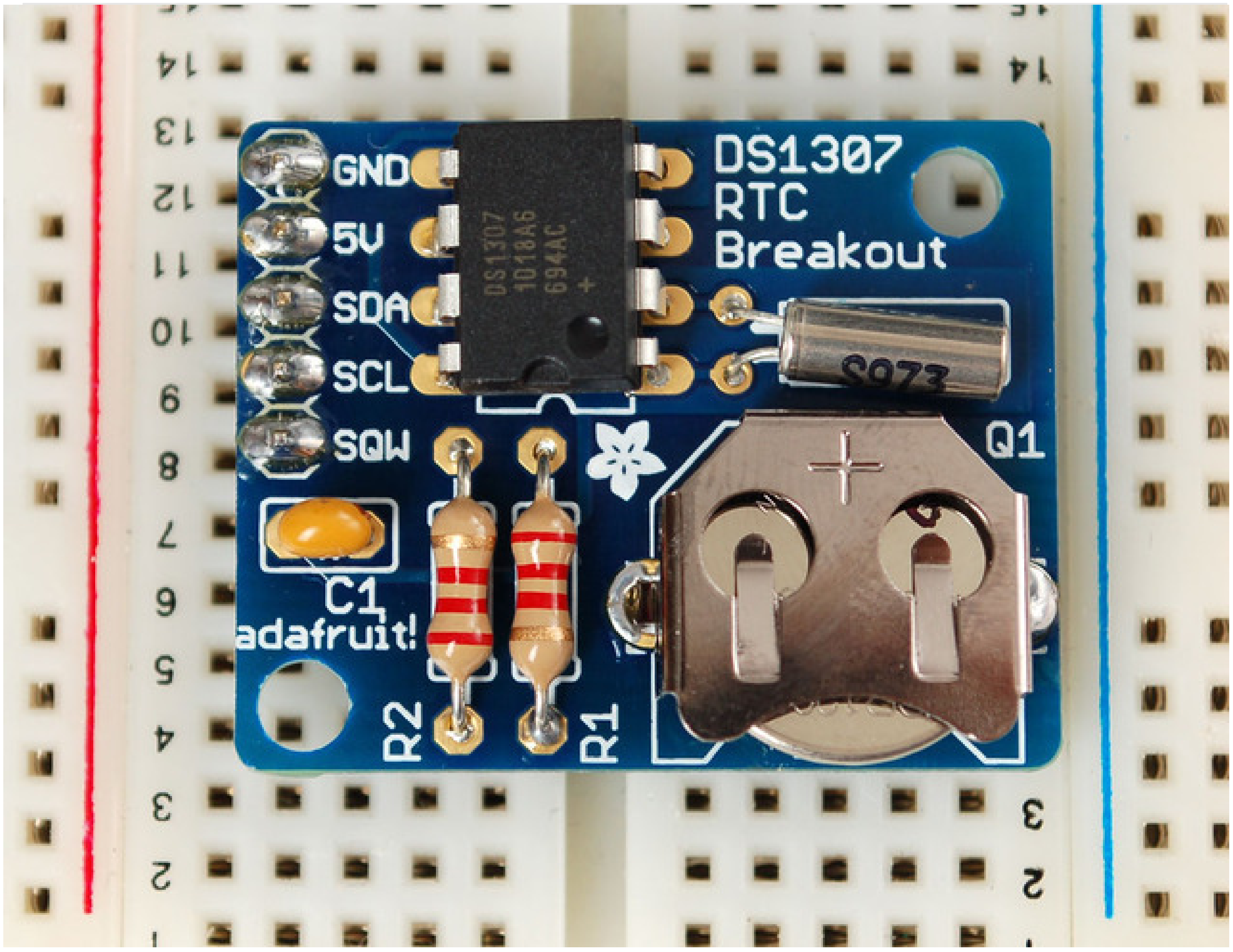
Created by lady ada



Last updated on 2014-06-28 08:30:50 PM EDT

Guide Contents

Overview



This tutorial requires a Raspberry Pi running a kernel with the RTC module and DS1307 module included. Current Raspbian distros have this, but others may not!

The Raspberry Pi is designed to be an ultra-low cost computer, so a lot of things we are used to on a computer have been left out. For example, your laptop and computer have a little coin-battery-powered 'Real Time Clock' (RTC) module, which keeps time even when the power is off, or the battery removed. To keep costs low and the size small, an RTC is not included with the Raspberry Pi. Instead, the Pi is intended to be connected to the Internet via Ethernet or WiFi, updating the time automatically from the global **ntp** (nework time protocol) servers

For stand-alone projects with no network connection, you will not be able to keep the time when the power goes out. So in this project we will show you how to add a low cost battery-backed RTC to your Pi to keep time!

Wiring the RTC

To keep costs low, the Raspberry Pi does not include a Real Time Clock module. Instead, users are expected to have it always connected to WiFi or Ethernet and keep time by checking the network. Since we want to include an external module, we'll have to wire one up. We'll go with the easy-to-use and low-cost DS1307. To make the job really easy, we'll use the the Adafruit DS1307 RTC Breakout Board Kit (http://adafru.it/264)- it comes with all the parts you need, even a coin battery!

The Kit does require a little light soldering. In theory you could use all the parts and build them onto a breadboard, but the coin holder is a little difficult since its not breadboard-friendly, so please go ahead and build the kit (http://adafru.it/clq).

When building the kit, leave out the 2.2KΩ resistors - by leaving them out, we force the RTC to communicate at 3.3V instead of 5V, which is better for the Pi!

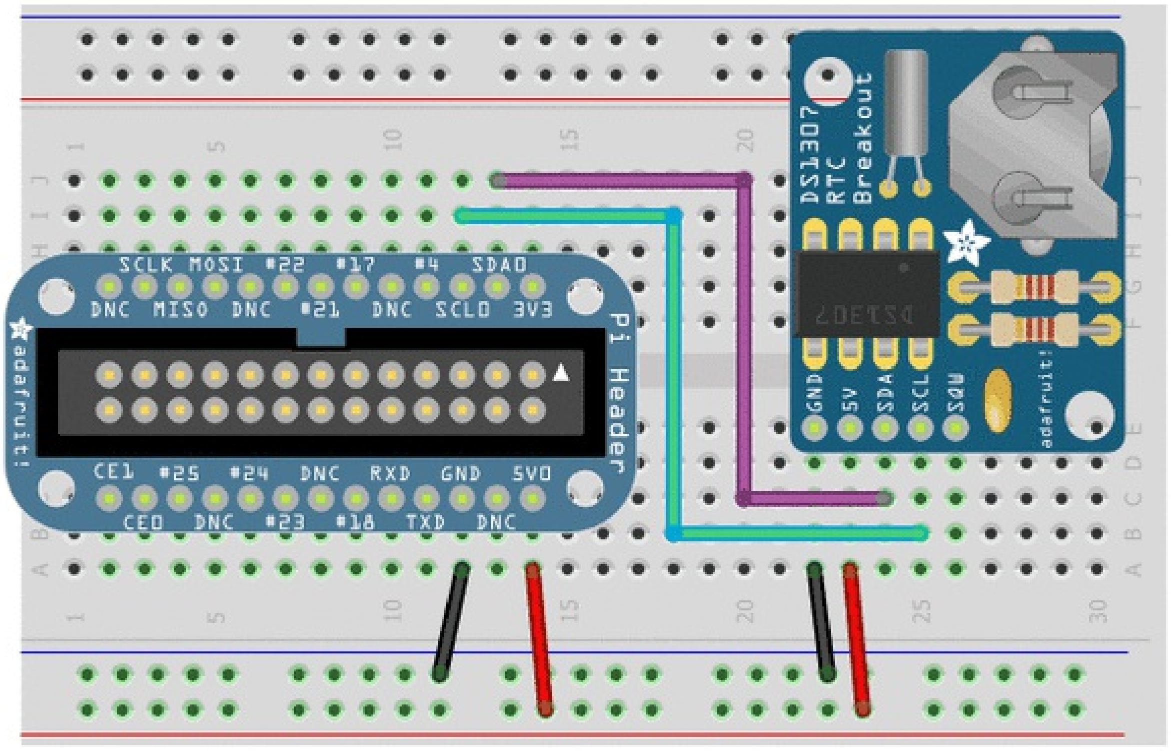
The diagrams below show the 2.2KΩ resistors in place - but please remove them either by not soldering them in or clipping them out if you did solder them in! This way, we'll use the Pi's 1.8K pull-up resistors to 3.3V

Wiring is simple:

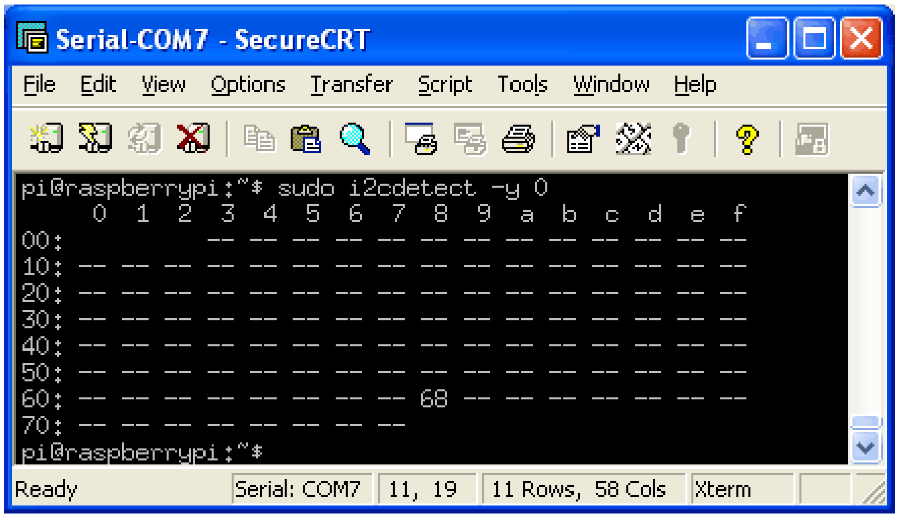
1. Connect **VCC** on the breakout board to the **5.0V** pin of the Pi
2. Connect **GND** on the breakout board to the **GND** pin of the Pi
3. Connect **SDA** on the breakout board to the **SDA0** pin of the Pi
4. Connect **SCL** on the breakout board to the **SCL0** pin of the Pi

DONT FORGET TO POWER IT FROM 5V - THIS IS FINE AND SAFE FOR THE PI - JUST MAKE SURE TO REMOVE THE 2.2K PULLUPS AS MENTIONED IN THE LARGE RED BOX UP ABOVE TO KEEP THE I2C PINS AT 3.3V LOGIC LEVELS

You'll also need to set up i2c on your Pi, to do so, check out our tutorial on Raspberry Pi i2c setup and testing at http://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c (http://adafru.it/aTI)
(http://adafru.it/aTI)



Verify your wiring by running **sudo i2cdetect -y 0** at the command line, you should see ID #68 show up - that's the address of the DS1307! If you have a rev 2 Pi, you will have to run **sudo i2cdetect -y 1** as the I2C bus address changed from 0 to 1



```
pi@raspberrypi:~$ sudo i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi:~$
```

Once you have the Kernel driver running, i2cdetect will skip over 0x68 and display UU instead, this means its working!

Set RTC Time

Now that we have the module wired up and verified that you can see the module with i2cdetect, we can set up the module.

First, load up the RTC module by running

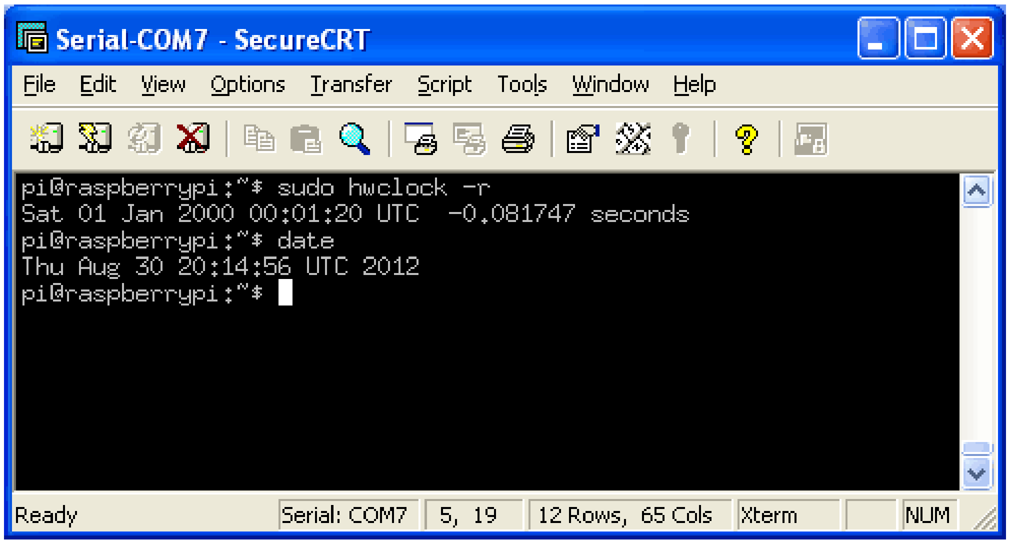
- **sudo modprobe rtc-ds1307**

Then, as root (type in **sudo bash**) run

- **echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device** (if you have a rev 1 Pi)
- **echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device** (if you have a rev 2 Pi)

You can then type in **exit** to drop out of the root shell.

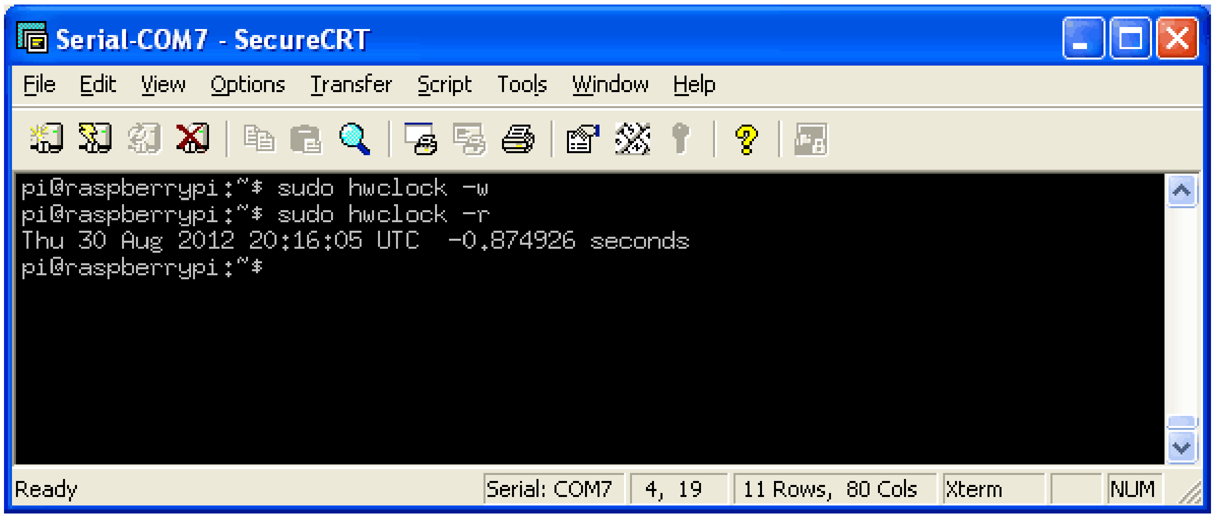
Then check the time with **sudo hwclock -r** which will read the time from the DS1307 module. If this is the first time the module has been used, it will report back Jan 1 2000, and you'll need to set the time



```
pi@raspberrypi:~$ sudo hwclock -r
Sat 01 Jan 2000 00:01:20 UTC  -0.081747 seconds
pi@raspberrypi:~$ date
Thu Aug 30 20:14:56 UTC 2012
pi@raspberrypi:~$
```

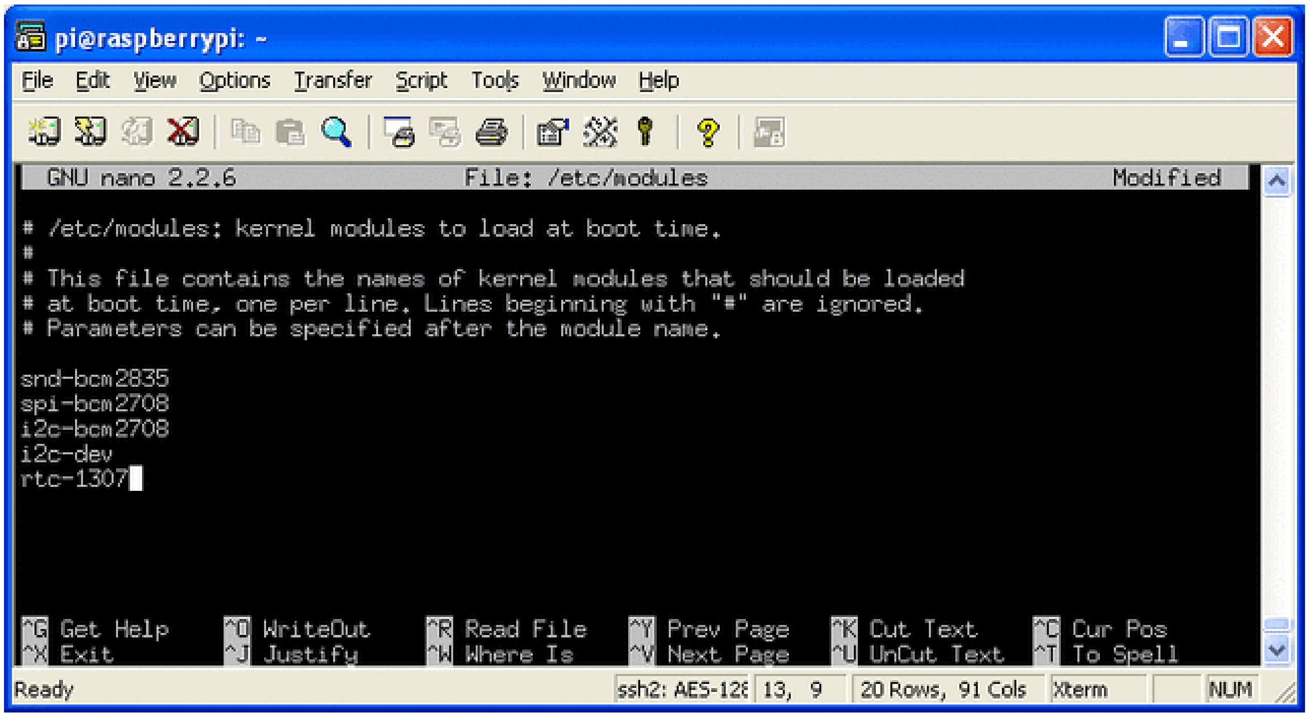
First you'll need to get the right time set on the Pi, the easiest way is to connect it up to Ethernet or Wifi - it will automatically set the time from the network. Once the time is correct (check with the **date** command), run **sudo hwclock -w** to write the system time to the RTC

You can then verify it with **sudo hwclock -r**



```
pi@raspberrypi:~$ sudo hwclock -w
pi@raspberrypi:~$ sudo hwclock -r
Thu 30 Aug 2012 20:16:05 UTC  -0.874926 seconds
pi@raspberrypi:~$
```

Next, you'll want to add the RTC kernel module to the /etc/modules list, so its loaded when the machine boots. Run **sudo nano /etc/modules** and add **rtc-ds1307** at the end of the file (the image blow says rtc-1307 but its a typo)



```
  GNU nano 2.2.6              File: /etc/modules                            Modified

# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.

snd-bcm2835
spi-bcm2708
i2c-bcm2708
i2c-dev
rtc-1307



^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

Then you'll want to create the DS1307 device creation at boot, edit /etc/rc.local by running

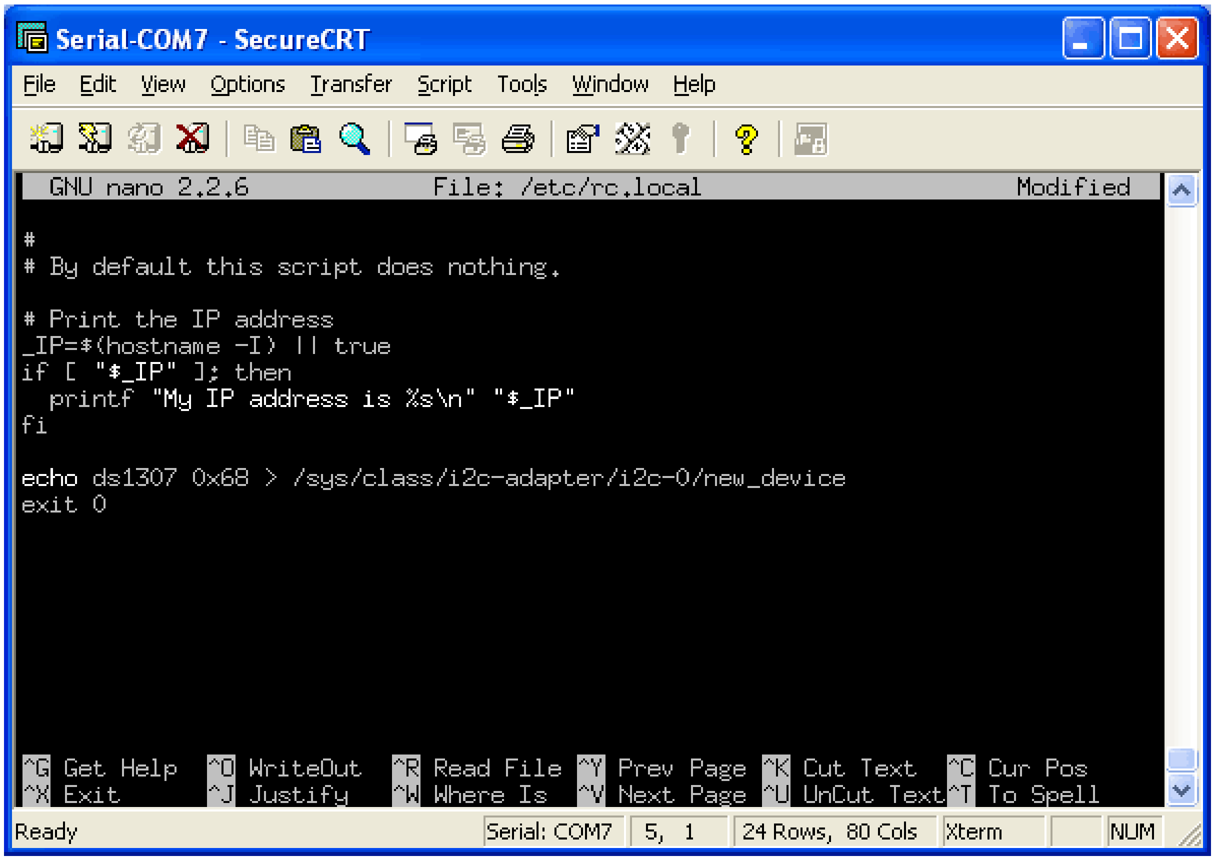
- **sudo nano /etc/rc.local**

and add:

> **echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device** *(for v1 raspberry pi)*
> **echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device** *(v2*

> *raspberry pi)*
> **sudo hwclock -s** *(both versions)*

before **exit 0** (we forgot the hwclock -s part in the screenshot below)



```
  GNU nano 2.2.6              File: /etc/rc.local                      Modified

#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-0/new_device
exit 0
```

```
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

That's it! Next time you boot the time will automatically be synced from the RTC module