

Set RTC Time | Adding a Real Time Clock to Raspberry Pi | Adafruit Learning System



🌐 Web Clip

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

- Don't forget to set up I2C in the previous step!

Raspbian Jessie (Systemd)

[Thanks to kd8twg for the hints!](#)

You can add support for the RTC by adding a device tree overlay.
Run

sudo nano /boot/config.txt

to edit the pi configuration and add whichever matches your RTC chip:

dtoverlay=i2c-rtc,ds1307

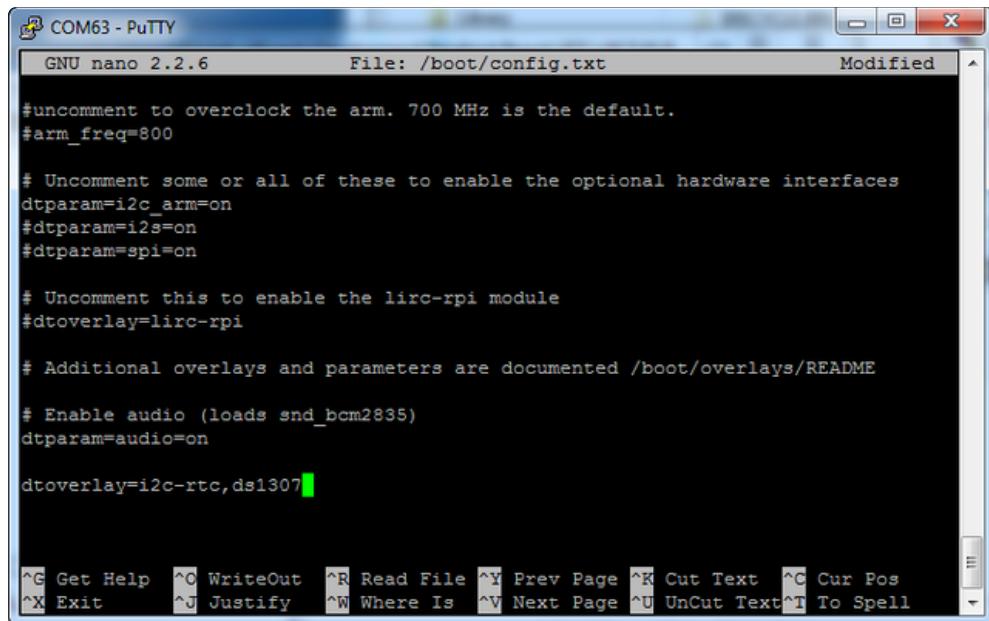
or

dtoverlay=i2c-rtc,pcf8523

or

dtoverlay=i2c-rtc,ds3231

to the end of the file



```
COM63 - PuTTY
GNU nano 2.2.6          File: /boot/config.txt          Modified

#uncomment to overclock the arm. 700 MHz is the default.
#arm_freq=800

# Uncomment some or all of these to enable the optional hardware interfaces
dtparam=i2c_arm=on
#dtparam=i2s=on
#dtparam=spi=on

# Uncomment this to enable the lirc-rpi module
#dtoverlay=lirc-rpi

# Additional overlays and parameters are documented /boot/overlays/README

# Enable audio (loads snd_bcm2835)
dtparam=audio=on

dtoverlay=i2c-rtc,ds1307

^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
```

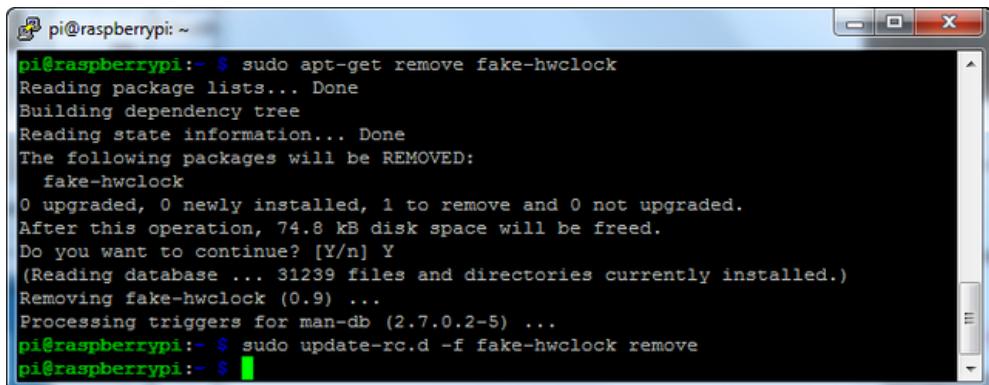
Save it and run **sudo reboot** to start again. Log in and run **sudo i2cdetect -y 1** to see the UU show up where 0x68 should be



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

Disable the "fake hwclock" which interferes with the 'real' hwclock

- **sudo apt-get -y remove fake-hwclock**
- **sudo update-rc.d -f fake-hwclock remove**
- **sudo systemctl disable fake-hwclock**



```
pi@raspberrypi: ~
pi@raspberrypi:~$ sudo apt-get remove fake-hwclock
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  fake-hwclock
0 upgraded, 0 newly installed, 1 to remove and 0 not upgraded.
After this operation, 74.8 kB disk space will be freed.
Do you want to continue? [Y/n] Y
(Reading database ... 31239 files and directories currently installed.)
Removing fake-hwclock (0.9) ...
Processing triggers for man-db (2.7.0.2-5) ...
pi@raspberrypi:~$ sudo update-rc.d -f fake-hwclock remove
pi@raspberrypi:~$
```

Now with the fake-hw clock off, you can start the original 'hardware clock' script.

Run **sudo nano /lib/udev/hwclock-set** and comment out these three lines:

```
#if [ -e /run/systemd/system ] ; then
# exit 0
#fi
```

```
GNU nano 2.2.6      File: /lib/udev/hwclock-set

#!/bin/sh
# Reset the System Clock to UTC if the hardware clock from which it
# was copied by the kernel was in localtime.

dev=$1

#if [ -e /run/systemd/system ] ; then
#    exit 0
#fi

if [ -e /run/udev/hwclock-set ]; then
    exit 0
fi

if [ -f /etc/default/rcS ] ; then
    . /etc/default/rcS
fi

# These defaults are user-overridable in /etc/default/hwclock
[ Read 37 lines ]
^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
```

Also comment out the two lines

`/sbin/hwclock --rtc=$dev --systz --badyear`

and

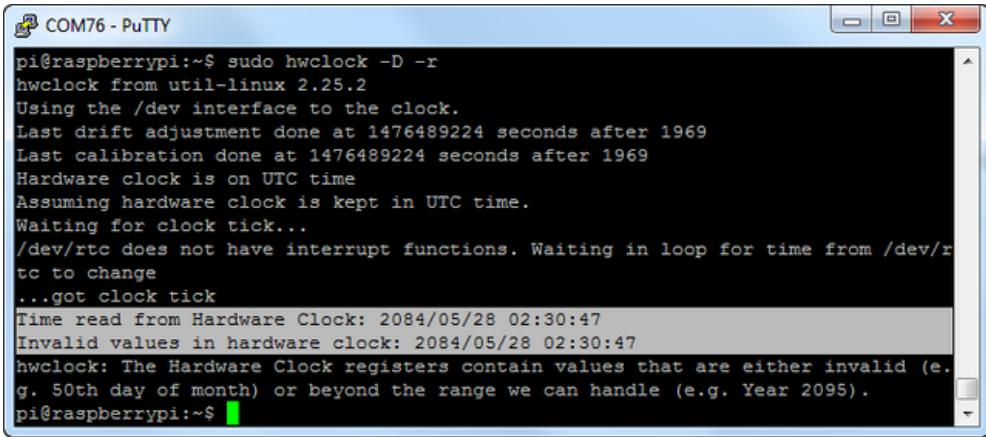
`/sbin/hwclock --rtc=$dev --systz`

```
if [ -e /run/systemd/system ] ; then
#    /sbin/hwclock --rtc=$dev --systz --badyear
#    /sbin/hwclock --rtc=$dev --hctosys --badyear
else
#    /sbin/hwclock --rtc=$dev --systz
#    /sbin/hwclock --rtc=$dev --hctosys
fi

# Note 'touch' may not be available in initramfs
> /run/udev/hwclock-set
```

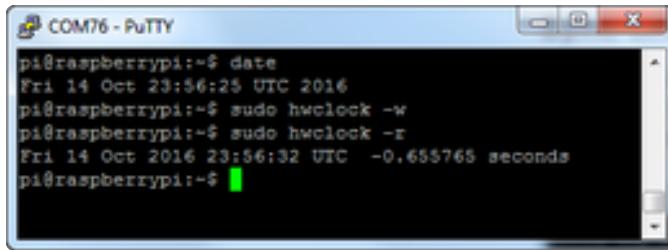
Sync time from Pi to RTC

When you first plug in the RTC module, it's going to have the wrong time because it has to be set once. You can always read the time directly from the RTC with `sudo hwclock -D -r`



```
pi@raspberrypi:~$ sudo hwclock -r
hwclock from util-linux 2.25.2
Using the /dev interface to the clock.
Last drift adjustment done at 1476489224 seconds after 1969
Last calibration done at 1476489224 seconds after 1969
Hardware clock is on UTC time
Assuming hardware clock is kept in UTC time.
Waiting for clock tick...
/dev/rtc does not have interrupt functions. Waiting in loop for time from /dev/rtc to change
...got clock tick
Time read from Hardware Clock: 2084/05/28 02:30:47
Invalid values in hardware clock: 2084/05/28 02:30:47
hwclock: The Hardware Clock registers contain values that are either invalid (e.g. 50th day of month) or beyond the range we can handle (e.g. Year 2095).
pi@raspberrypi:~$
```

You can see, the date at first is invalid! You can set the correct time easily. First run `date` to verify the time is correct. Plug in Ethernet or WiFi to let the Pi sync the right time from the Internet. Once that's done, run `sudo hwclock -w` to write the time, and another `sudo hwclock -r` to read the time



```
pi@raspberrypi:~$ date
Fri 14 Oct 23:56:25 UTC 2016
pi@raspberrypi:~$ sudo hwclock -w
pi@raspberrypi:~$ sudo hwclock -r
Fri 14 Oct 2016 23:56:32 UTC -0.655765 seconds
pi@raspberrypi:~$
```

Once the time is set, make sure the coin cell battery is inserted so that the time is saved. You only have to set the time *once*

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

Raspbian Wheezy or other pre-systemd Linux

First, load up the RTC module by running

```
sudo modprobe i2c-bcm2708  
sudo modprobe i2c-dev  
sudo modprobe rtc-ds1307
```

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

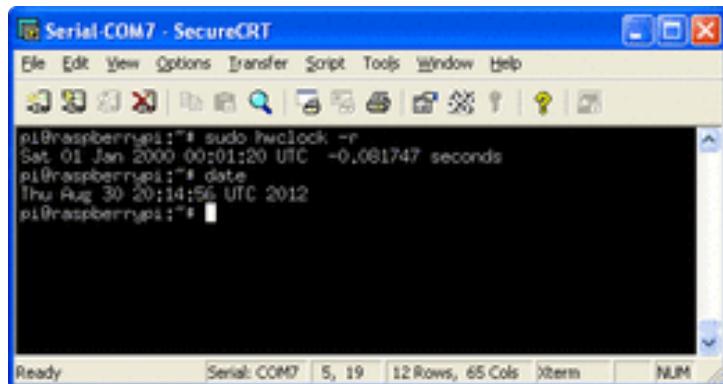
```
echo ds1307 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

If you happen to have an old Rev 1 Pi, type in

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

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



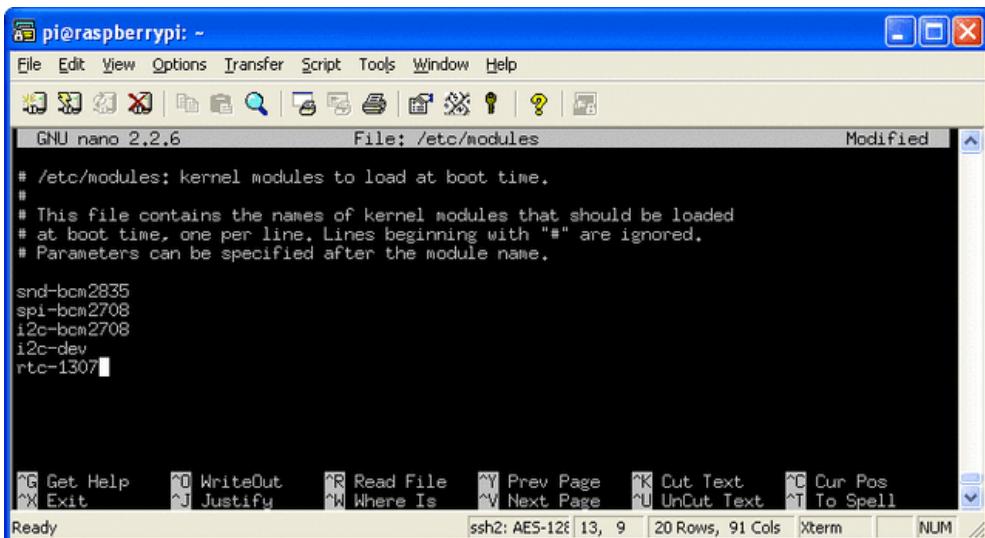
First you'll need to get the right time set on the Pi, the easiest way is to connect it up to Ethernet or Wifi – it will automatically set the time from the network. Once the time is correct (check with the **date** command), run **sudo hwclock -w** to write the system time to the RTC

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



```
pi@raspberrypi:~$ sudo hwclock --set
pi@raspberrypi:~$ sudo hwclock --read
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 below says rtc-1307 but its a typo)



```
pi@raspberrypi: ~
File Edit View Options Transfer Script Tools Window Help
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
```

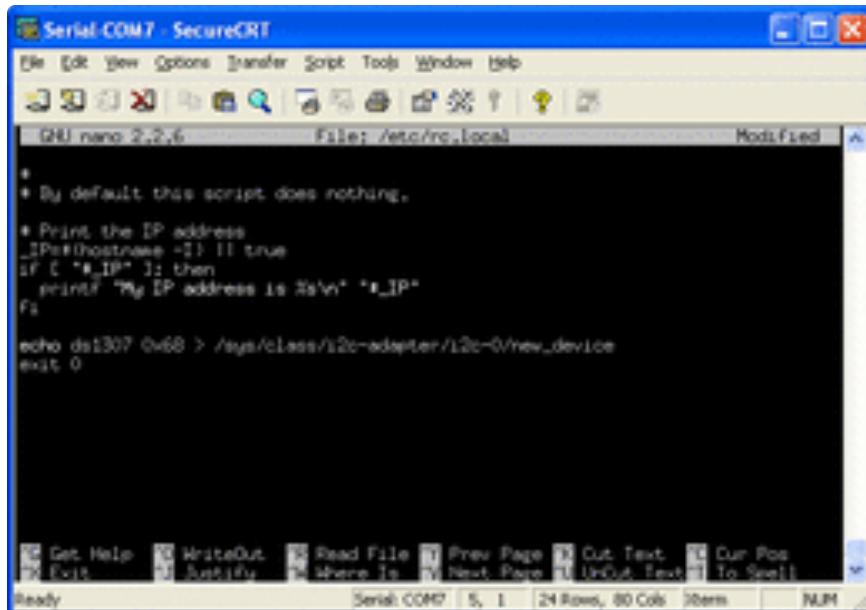
Older pre-Jessie raspbian is a little different. First up, you'll want to create the DS1307 device creation at boot, edit /etc/rc.local by running

```
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 (for v2 raspberry pi)
sudo hwclock -s (both versions)
```

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



```
Serial-Com7 - SecureCRT
File Edit View Options Transfer Script Tools Window Help
GNU nano 2.2.6          File: /etc/rc.local      Modified A
+
+ By default, this script does nothing.
+
+ Print the IP address
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

Ready           Serial-Com7 | 5, 1 | 24 Rows, 80 Cols | 0 terms | NUM
```

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

