

Raspberry PI 'How-To' Series

Real Time Clock - DS3231 Implementation Guide

Written by: Sopwith
Revision 1.0
February 7, 2019
sopwith@ismellsmoke.net

"If it works out of the box – what fun is that?"

Introduction

In order to keep the cost of the Raspberry Pi as low as possible, the designers had to leave out some features. One of those missing items is a real-time hardware clock. For the majority of Pi freaks, this is just fine because their Pi's have access to a network where they can obtain the time from a time server via the NTP protocol.

In fact, the Linux kernel implements a pseudo or 'fake' hardware clock driver that makes the hardware think there is a real clock on board.

There are plenty of times when Makers will use a Pi in some project full of awesomeness, and they need an accurate and permanent time keeping capability. This implementation 'How-To' shows how to attach a real time clock to a Pi.

Clock Hardware

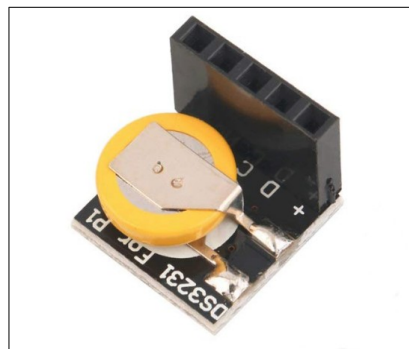
When it comes to real time clock (RTC) hardware for the Pi, there are plenty of choices. The first decision you have to make is whether you want to mount a RTC device directly on the Pi's GPIO pins, or use a separate breakout board. Your decision will be based on your specific needs and how much room you have available to house your project.

Probably the most popular RTC chip in use for single board computers (SBC) is the DS1307. They are amazingly inexpensive (@ \$1 USD). This chip will work in most applications that do not require an accurate timekeeper. The drawback of the DS1307 is the variance in accuracy. This chip can gain or lose seconds of accuracy every day. [In some cases](#), the device is known to lose \pm minutes per day. The variance is due to the accuracy of the crystals used and temperature variances.

If you are looking for something with much better accuracy, then I recommend you choose a RTC based on the DS3231 chip. This device has a temperature sensor and will adjust its timing based on temperature. It is accurate to \pm .437 seconds per day. For me, this is something I can live with.

The device I will use in this document is shown below in Figure-1.

Figure-1



The above device is small enough to sit within the confines of a US nickel. Tiny, cheap, and accurate. A winning combination. You can buy them online at [Amazon](#) who sells a set of five for \$10.99 USD. You can also get them on E-Bay or buy them direct from China even cheaper. [Adafruit](#) also has an excellent collection of RTC devices. Most of them do not ship with batteries because of post restrictions.

Step-by-Step

In this guide, I will install the DS3231 RTC on a Raspberry Pi Zero V.1.2. The device uses the I2C interface of the Pi and mounts right on the GPIO pins. Many of the steps I outline here are described in an excellent [document](#) published by Adafruit.

There are 7 steps to installing the RTC.

1. Get your Pi up and running
2. Enable the I2C interface
3. Install required support software
4. Install the RTC
5. Test the RTC
6. Configure the Pi for a hardware clock.
7. Set the RTC to the correct time.

Step-1 - Get your Pi up and running with Raspian Stretch.

The first thing you need to do is get your Pi running. Head out to raspberrypi.org and download *Raspian Stretch* or *Stretch Lite*. The former has a Window GUI while the latter is a command-line only OS. Either one works.

Once the download completes, burn the image to an SD card and fire up your Pi. If you need help with this step, there is a ton of resources on the web to help you. When you are up and running and connected to the Internet, be sure to update your Pi with the latest patches.

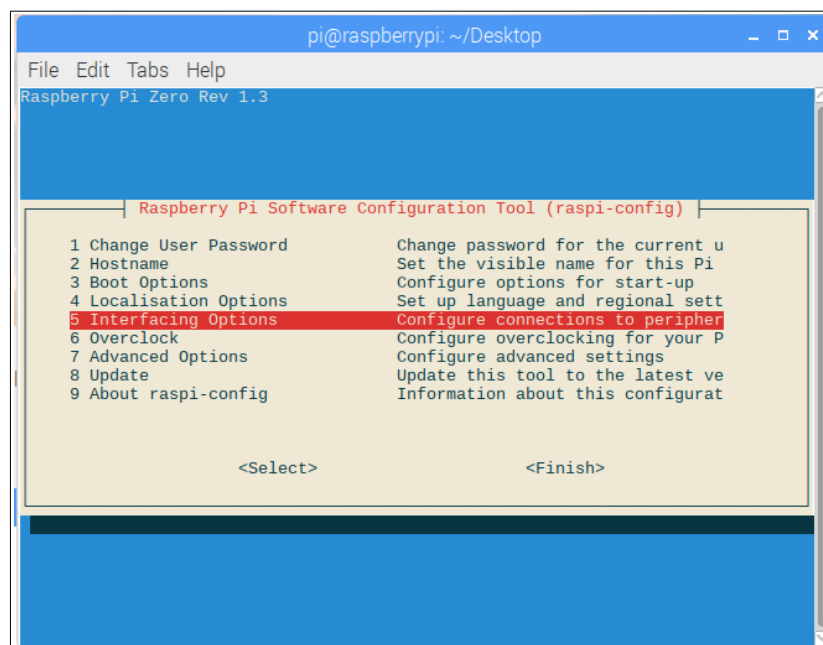
```
$sudo apt-get update  
$sudo apt-get upgrade  
$sudo reboot
```

Step-2 - Enable the I2C Interface

Enable the I2C interface using the *raspi-config* utility. From the command window run the command:
`$sudo raspi-config`.

You will see the window shown in Figure-2.

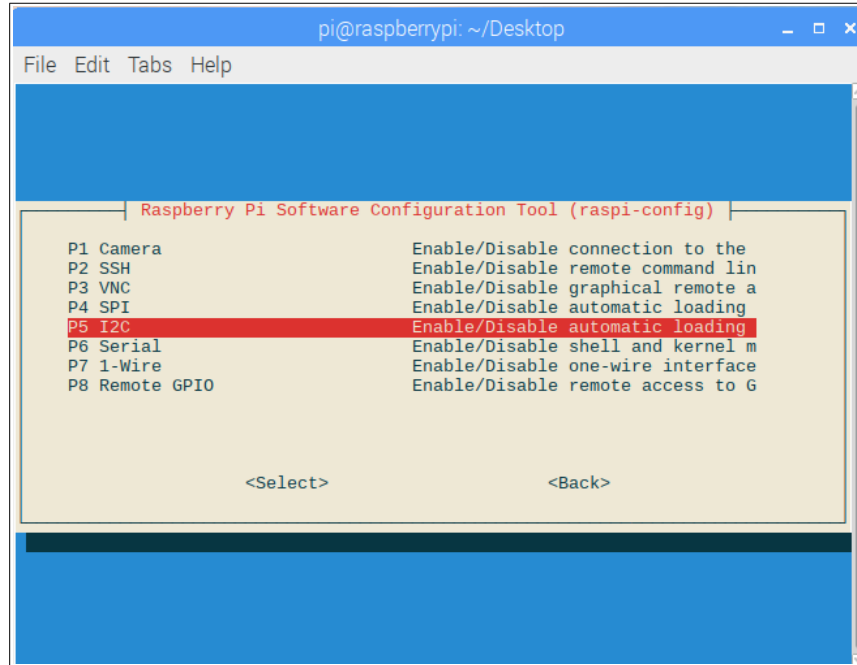
Figure-2



"If it works out of the box – what fun is that?"

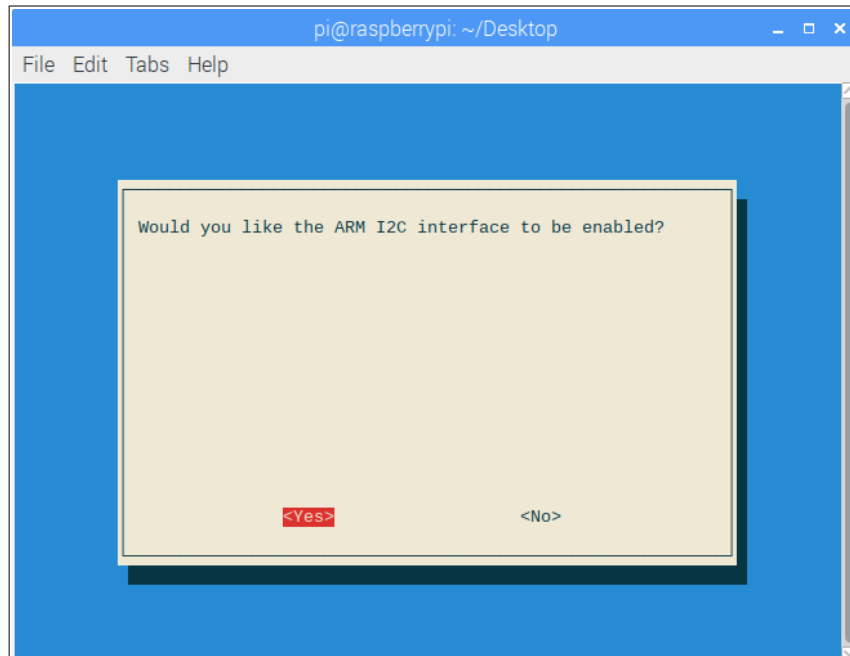
Press <Enter> on Line-5 and you will see the window shown in Figure-3.

Figure-3



Press <Enter> and you will see the window shown in Figure-4.

Figure-4



Select <Yes> and press <Enter> and you will see the window shown in Figure-5.

Figure-5



Exit raspi-config. Just to be safe, reboot your Pi:
`$sudo reboot.`

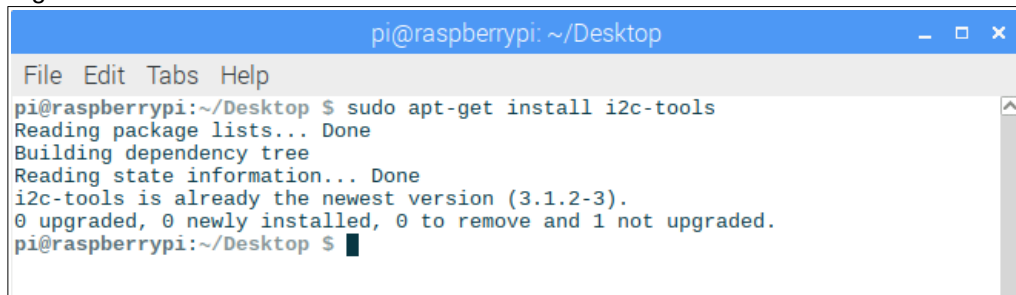
Step-3 - Install required support software.

There are two (2) software packages that must be installed for the I2C interface to work correctly. Follow the below steps to get them installed.

Open a command window and type in the command:
`$sudo apt-get install i2c-tools.`

You will probably get the output shown in Figure-6 below.

Figure-6



Next, run the command:
`$sudo apt-get install python-smbus.`

You will probably get the output shown in Figure-7 below.

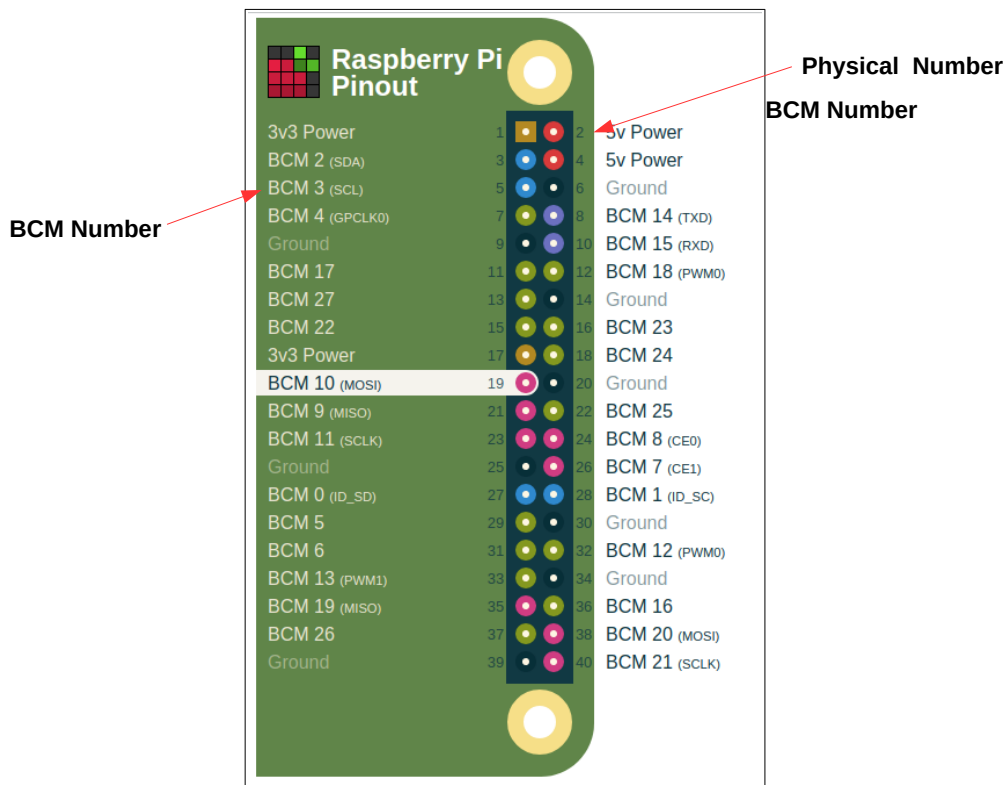
Figure-7

```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo raspi-config  
pi@raspberrypi:~ $ sudo apt-get install python-smbus  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
python-smbus is already the newest version (3.1.2-3).  
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
pi@raspberrypi:~ $ _
```

Step-4 - Install the RTC

Installation of the RTC is a snap. You simply slide it on the GPIO pins of your Pi with your Pi powered off. Which pins you ask? Well, that can be a bit confusing because the Pi GPIO pins have a pair of reference sets. If you look at the “Raspberry Pi Pinout” in Figure-8, the GPIO pins are labeled by their physical location from 1-40, and also by their ‘BCM’ number.

Figure-8¹



The physical numbers start with 1 and alternate across the board and then down. This puts all of the even pins on one side, and the odd pins on the other. Pin-1 on most Pi’s is on the side of the board closest to the SD card. Be sure to know where Pin-1 is on your particular Pi board.

The BCM numbers are assigned by *Broadcom*, the maker of the Pi’s CPU. In most documentation, code, and discussions around the Pi universe, when describing a GPIO pin number, they are usually talking about the BCM pin number, *not* the physical pin number.

1 https://pinout.xyz/pinout/pin19_gpio10

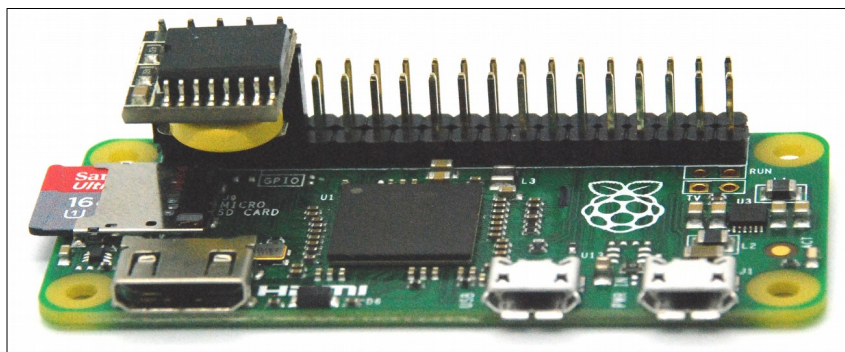
An interesting side-note here. The Raspian Desktop image has a cool utility named *pinout* that you can run from a command prompt. An image will appear as shown in Figure-9. This is really handy when you need it.

Figure-9



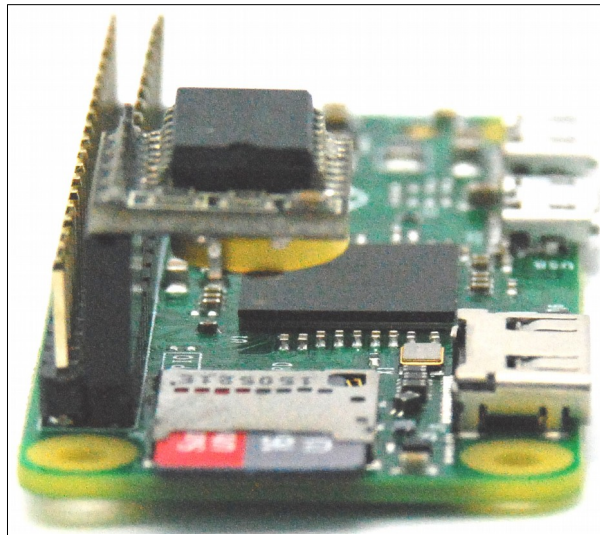
The DS3231 RTC I am using slides over physical pins 1,3,5,7 and 9, or in BCM terminology the 3v pin, BCM pins 2,3 and 4, and the ground pin. Figure-10 and Figure-11 below shows the RTC installed on my Pi Zero.

Figure-10



"If it works out of the box – what fun is that?"

Figure-11



Step-5 - Test the RTC

Once the RTC is installed on the correct GPIO pins, boot up your Pi and open a command window. From the command line run the command:

```
$sudo i2cdetect -y 1
```

This is shown below in Figure-12.

Figure-12

```
pi@raspberrypi: ~
File Edit Tabs Help
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: -- -- -- -- -- -- -- -- 68 -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
pi@raspberrypi:~ $ _
```

If your Pi recognized the RTC, you should see its I2C hardware address somewhere in the grid. In my case, the RTC has an address of 68. The address of your device may be different. Also note there may be more than one address shown in the grid if you have other I2C devices attached to your Pi.

Now that we know the hardware all works, we need to make some more configuration changes.

Step-6 – Configure the Pi for a hardware clock

There are a few things we need to do to tell the Pi it has its own clock on board. First, add a line to the `/boot/config.txt` file.

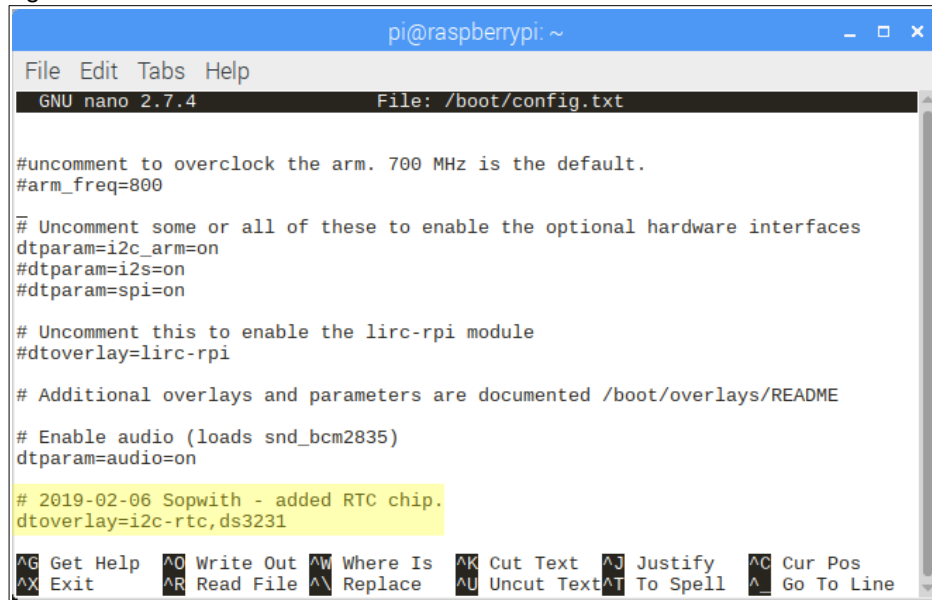
Open the config.txt file for editing:

```
$sudo nano /boot/config.txt.
```


At the bottom of the file add a line:
`dtoverlay=i2c-rtc,ds3231`

The line you need to add is shown below in Figure-13.

Figure-13



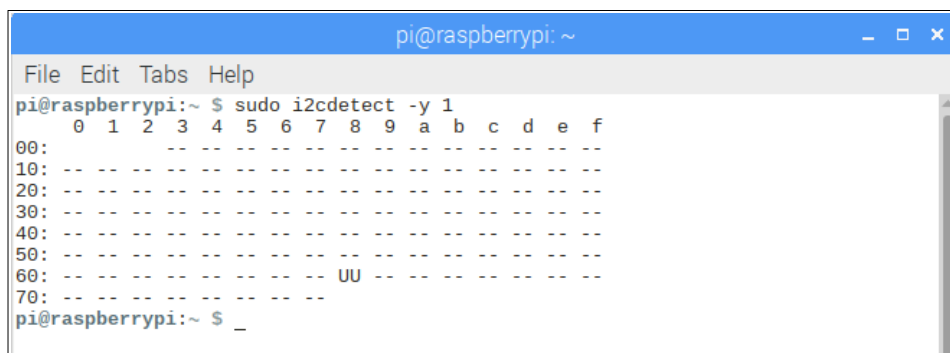
Save the file: `<Ctrl> <o> <Enter> <Ctrl><x>`.

Reboot your Pi and run the command:

`$sudo i2cdetect -y 1`

This is shown below in Figure-14.

Figure-14

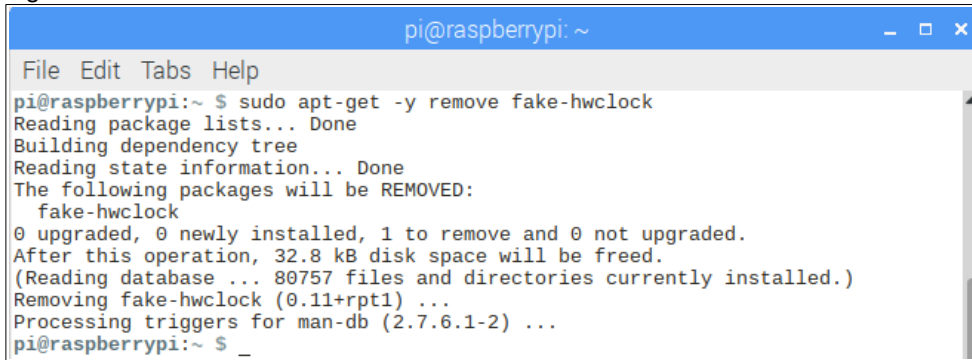


Notice that the I2C address of the RTC changed from 68 to UU. This indicates that the kernel has taken over control of the device. This is exactly what we want.

Next, we need to remove the fake-hwclock package from the Pi. To so do, enter the following command:
`$sudo apt-get -y remove fake-hwclock`.

The output of this command is shown in Figure-15.

Figure-15



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo apt-get -y 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, 32.8 kB disk space will be freed.  
(Reading database ... 80757 files and directories currently installed.)  
Removing fake-hwclock (0.11+rpt1) ...  
Processing triggers for man-db (2.7.6.1-2) ...  
pi@raspberrypi:~ $ _
```

Remove the fake-hwclock from the run command daemon by running the command:
`$sudo update-rc.d -f fake-hwclock remove.`

This is shown in Figure-16.

Figure-16

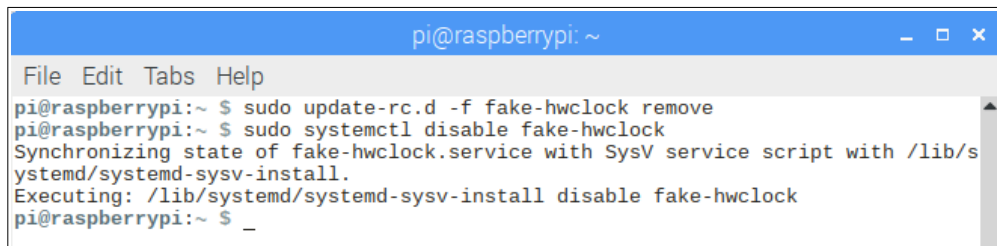


```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo update-rc.d -f fake-hwclock remove  
pi@raspberrypi:~ $ _
```

Remove the fake-hwclock for systemd with the command:
`$sudo systemctl disable fake-hwclock remove.`

This is shown in Figure-17.

Figure-17



```
pi@raspberrypi: ~  
File Edit Tabs Help  
pi@raspberrypi:~ $ sudo update-rc.d -f fake-hwclock remove  
pi@raspberrypi:~ $ sudo systemctl disable fake-hwclock  
Synchronizing state of fake-hwclock.service with SysV service script with /lib/s  
ystemd/systemd-sysv-install.  
Executing: /lib/systemd/systemd-sysv-install disable fake-hwclock  
pi@raspberrypi:~ $ _
```

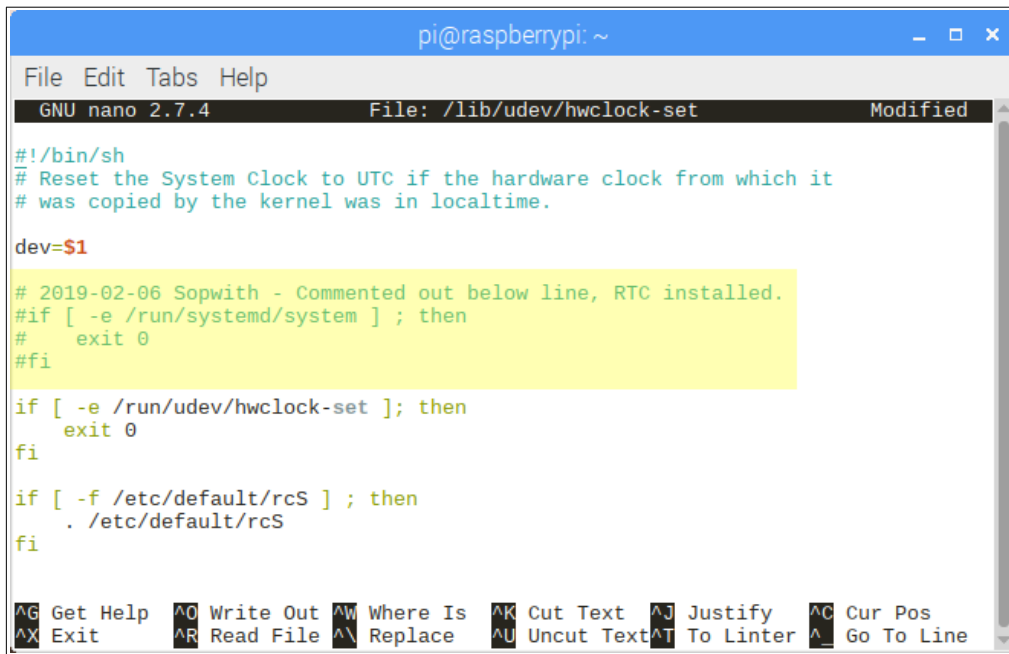
Edit the `/lib/udev/hwclock-set` file using the below command:
`$sudo nano /lib/udev/hwclock-set`

Look for the below three lines and comment them out:

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

These lines should be near or at the top of the file. You can see the required changes in Figure-18.

Figure-18



```
pi@raspberrypi: ~
File Edit Tabs Help
GNU nano 2.7.4 File: /lib/udev/hwclock-set Modified

#!/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

# 2019-02-06 Sopwith - Commented out below line, RTC installed.
#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

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Linter ^_ Go To Line
```

Finally, reboot your Pi so the changes will take effect.

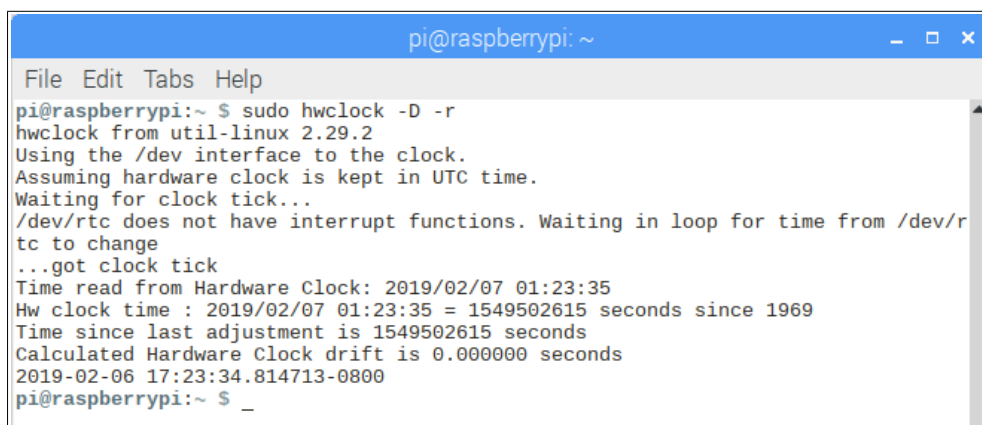
Step-6 - Set the RTC to the correct time

We are done with all the configuration stuff. Now we need to make sure the RTC has the correct time. From a command prompt enter the following command:

```
$sudo hwclock -D r
```

This is shown in Figure-19.

Figure-19



```
pi@raspberrypi:~ $ sudo hwclock -D -r
hwclock from util-linux 2.29.2
Using the /dev interface to the clock.
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: 2019/02/07 01:23:35
Hw clock time : 2019/02/07 01:23:35 = 1549502615 seconds since 1969
Time since last adjustment is 1549502615 seconds
Calculated Hardware Clock drift is 0.000000 seconds
2019-02-06 17:23:34.814713-0800
pi@raspberrypi:~ $ _
```

As you can see, my hardware clock is set to the right time. In this case, since I have a WiFi USB device attached to a powered hub, my Pi Zero has access to the Internet. During the last reboot, an NTP time request was made from the Internet. The time returned was used to set the hardware clock. Thus, I do not have to do anything.

If your Pi is not connected to the Internet, you may have to set the RTC yourself. This can be done a couple of ways, including the use of the `hwclock` command from above. A Google search will show how to do this if you are unsure.

Summary

In order to save on costs and because the board space on a Raspberry Pi is so limited, no version of the Pi comes with a hardware clock. This means the device is completely dependent on an Internet connection to set the “fake” hardware clock to the right time. This is problematic for those Makers who do not have or want an Internet connection but still need an accurate time clock.

This “How-To” walks through the process of installing and configuring a tiny, low-cost and highly accurate hardware clock to a Raspberry Pi. Although these steps highlight a small clock that slides over the Pi GPIO pins, these instructions can be used to install any hardware that used the I2C bus.

Send corrections, comments, complaints, ideas, or any other feedback to: sopwith@ismellsmoke.net.