Raspberry Pi –
Adding a Real Time Clock (RTC)

Level of difficulty: Beginner
Hardware: Raspberry Pi Model B, RTC Module, wires, optional connectors
Tools required: Wire cutters, soldering iron
Project cost: Under $10

Document Version: September 2013 v1.0.0

Written By: David Osborne raspberry_pi@hotmail.com

The point of this series of documents is to provide a very simple set of directions to add hardware or install software to the Raspberry Pi. These represent my findings. I’m not a ‘professional’, not an engineer and don’t come close to being an expert on the Raspberry Pi (RPI). I am however familiar with a lot of the fundamentals of electronics, pretty good at soldering and have done a lot of programming in many different languages. You follow my advice at your own risk. Much of the information here is taken from other documents and from others who are in some cases much more knowledgeable than I am. One of the issues I found is that many articles are out of date. These documents are specifically for the Raspberry Pi Model B 512MB and based on what I could find in the Fall of 2013. For the most part I expect this information will apply to the original RPi model and may even carry forward. I hope you find them useful.

Hardware

About “Clocks”. Once the Raspberry Pi (RPI), or any computer really, is running, it starts keeping track of the date/time. Without a starting point many systems default to 01/01/1980 00:00 or some equally useless date/time. This isn’t very good for obvious reasons. Many systems allow you to manually set the date/time, for example in Linux you can issue the “date” command and tell the RPI what day and time it is. From then on it should be fairly accurate as it does have timer capabilities. This is known as a “System Clock” or “Software Clock”. Without a Hardware Clock, after a reboot, you have to set the date/time again as the value is not “saved” anywhere.

A “Hardware Clock”, also known as a Real Time Clock or RTC, is a small module or set of components that are powered by a battery. The chip is a timer and will continue to run even if the RPi or computer is turned off and unpowereed. Once set it will keep fairly accurate time on its own.

The ideal situation is for the RPi or computer to boot up and read the RTC Hardware Clock as a starting point for its own System Clock.
The RPi does not include a Real Time Clock (RTC) to keep costs and size at a minimum and for other reasons explained on the official website. Raspbian, and some other releases, use a pseudo-clock resolution to keep the time up to date or at least close to it if no internet connection is available. When performing a shutdown (using the appropriate commands), the RPi will automatically write the date/time that it thinks is current to a small file. This file is read on the next startup as a starting point for System Clock. Obviously if the RPi is unpowered for more than a few minutes then the System Clock won’t be anywhere near the actual time, but at least it doesn’t default to a date in 1980. Fortunately, there is a standard on the internet to read the current date/time from specific “Time Servers”, for free. The RPi uses the Network Time Protocol (NTP) to get the current time, assuming it is connected to the internet) and update the Software Clock. This is a pretty good solution without having to add hardware or increase the size of the RPi.

Confused about “Software Clock”, “Hardware Clock” and “Internet Clock” yet? I hope not. Just remember all three can be different but ultimately you want all three to be as in sync as possible.

The first and obvious question is: Do I need a Real Time Clock. The answer in most cases is “No”. It can be a fun and simple starting project though for learning and experimenting. If your RPi is not connected to the internet, or specifically a NTP Server, then it can’t update the clock to the actual time. In these cases it may be appropriate to add a RTC. If your RPi is connected to the internet and you do not run time sensitive functions or do not want to play with some simple hardware, then the default pseudo-hardware clock may work just fine for you.

Let’s assume that if you are still reading and you want to install a RTC on your RPi. Fortunately, there are MANY options for a RTC due to similarities of the Arduino and other project boards. Unfortunately, this may cause some unwanted problems too, more on this shortly.

The first thing you must do is get a RTC module. These range from $2 to $40 depending on where you buy them. You can even buy just the chip, crystal and battery holder and build your own “module” should you wish to do so. This document will show how to add a standard pre-packaged module.

When searching for a RTC you will discover a variety of chip sets available to you. You will see the most common references currently are DS1302, DS1307 and DS3231. The basic instructions here apply to all 3 options. I’m not a super-technical-electronics-guru and I’m not writing a super-technical document so to keep things simple, the DS1302 can be used but uses a different (older - SPI) interface. The DS1307 and DS3231 can use the same software (I2C) and is essentially ready to go as some of the scripts are already in your distribution. If you really want to make it simple or do not have basic soldering equipment or skills, then you can even buy (at a premium) a RTC that has the appropriate connector already in place and simply fit it onto the GPIO port of the RPi. Then all you have to do is skip to the software section to make it work, or follow the directions, if any, that it came with. The RTC modules will look something similar to the following as they all contain the same basic components.

Example of a DS1307 RTC module with connector:
Example of a DS1307 RTC without a connector:

Example of a DS3231 RTC

Without getting too technical, for a RTC to work you need to provide it power and connect the data wires. The battery on the RTC is used, in most cases, only to run the clock when external power is not available. This is the case when your RPi is turned off or unplugged. The battery keeps the clock running so when you turn your RPi back on the date/time are correct. This is much like the RTC is most computers, laptops, tablets, etc. Some have batteries you need to periodically replace and some recharge. If you find the RTC is not keeping proper time when the RPi is off then you may need to check and replace the battery. Be sure to replace it with an appropriate size, voltage and type. Many coin cell batteries look similar. Most RTCs use the CR2032 which is a 3V battery and should last 1-3 years.

The power connections for the RTC may be marked in a variety of ways. The most common references are (+) 5V or Vcc for the positive voltage and GND or (-) for the negative/ground connection.
The data connections for the RTC should be marked SDA and SDL.

The 5V/Vcc connection on the RTC connects to the 5.0V pin (#2) on the RPi.

The GND connection on the RTC connects to the GND pin (#6) on the RPi. (although you can use any ground point)

The Serial **Data** line [SDA] connection on the RTC connects to the SDA pin (#3) on the RPi.

The Serial **Clock** line [SCL] connection on the RTC connects to the SCL pin (#5) on the RPi.

Your module should have similar markings.

Here are images showing the GPIO connector (top right of the RPi) and the function of the various pins. Pin #1 is at the top left.

Now the unfortunate part I referred to earlier in this document. Many of the RTC modules that you can buy include something called “pull-up resistors”. These resistors change the voltage on the data connections from a safe 3.3V to an unsafe 5V. On the RTC these are often 2.2K Ohm resistors. If your RTC comes “Raspberry Pi – friendly” then these resistors should not already be installed. If the RTC was designed for an Arduino or other development board, then it is likely they are installed. The RTC image posted earlier shows an example.
Repost of the DS1307 RTC which has pull-up resistors in place:

![DS1307 RTC Breakout Module](image)

In the example above, the resistors marked as R1 and R2 are 2.2K Ohm pull-up resistors and they should be removed. Note that the R1 and R2 designations are specific to this module and may or may not be the same markings on another module. It is also potentially possible to identify the appropriate resistors by locating them by value (2.2K Ohm reflected by the 3 red bands) although they are the only resistors mounted on this particular module. De-soldering them is the ideal approach and allows you to keep them for future use although they are cheap and easy to find in an electronics store if required. However it is often easier to simply cut the leads to the resistors with a pair of cutters and remove them. This also avoids potential damage to the rest of the module when de-soldering. If you are not great at it I suggest just cutting the resistors off.

Here is another example of the DS1307 that has pull-up resistors (332) that must be removed. This is actually the RTC module that I installed:

![Tiny RTC I2C Modules](image)

In this example, the pull-up resistors are marked R2 and R3. Although I suspected them to be the actual pull-up resistors I was not sure until I Googled the device and found another post verifying it. In this example there are many resistors mounted to the board. They are SMT/SMD or Surface Mount Resistors that are VERY tiny. They are essentially a tiny black spec to the left of the R2 and R3 markings. Removing them is a lot trickier since they cannot be cut easily. De-soldering is the only real approach. There was also a post questioning if this changed with the revision of the specific TinyRTC version as someone found R2/R3 connected to the battery and R1 was the pull-up resistor. I checked my board with a multimeter and verified in my case that R2/R3 were the ones I needed to remove.
Why is it necessary to remove the pull-up resistors? The RPi is designed for 3.3V on its GPIO port. Running 5V through it can cause actual physical damage to the RPi or provide irregular results. It may not seem like much of a difference but these are “logic levels” and the chips have little to no protection. That being said, depending on your power source and the number of type of devices you are running, it may be possible, BUT UNADVISABLE to run with the pull-up resistors in place. If you do it, you do it at your own risk. I wouldn’t devote this much space to something if it wasn’t important.

Now that we have identified everything, let’s make the connections.

If you are lucky, the easiest way to make the connections, at least for testing, is to use four jumper cables with single female connectors on each end. If your RTC has a male pin header then it is just a matter of plugging in the wires.

It is up to you to decide how best to make the four required connections. You could solder 4 wires from the RPi to the RTC. I don’t recommend doing that, at least not at the RPi end. The RTC is essentially a throw-away device, assuming you didn’t pay much for it. I picked up a few for about $2-$3 (U.S.) so don’t really care if they get damaged and thrown away later. I plan on using them for the RPi only. I soldered 4 wires directly to the connection points on the RPi. This particular module allows for other devices to be daisy chained to it and has a second set of connections. Colour-coding your wiring is nice, but obviously not essential. Red is typically used for (+)5V/Vcc and Black for GND. Data wires can be any colour you want, the same for SDA and SCL or different. The only thing that really matters is that you connect them to the proper connection points. Four separate colours make it easy to see what you are doing though.

Here is an image of the one I installed:

![Image of RTC](image)

I soldered the wires to the connection points on the RTC but I installed a 2x3 female connector on the RPi end. I simply inserted the wires into the corresponding holes in the plug connector to match up with the GPIO. I could also have installed a 4 pin male header on the RTC and used jumper wires with connectors already on them, but decided not to.

If you do not have the option of building a connector yourself, one of the easiest things to do is to buy a connection cable that is already made and modify it to what we need. Ideally, you want a 6-wire flat cable with a 2x3 connector on the end.
Example connection cable:

If you purchase something like the cable above, then it is simply a matter of cutting the cable at an appropriate point to remove one connector and expose the wires. Don’t cut it too close to one of the connectors as you could potentially use it for two RTC installs or have a spare in case you need it for some reason. You also don’t want too much wire as the signalling voltages are fairly low and longer wires cause more resistance. They are also susceptible to interference as they are usually unshielded. Don’t cut it too short or it will be difficult to work with. You want a few inches to work with at least, let’s say 4. Cut straight across the wires as you want them all the same length. Separate the individual strands and keep only the four wires you need. Remember the wire connector in this case is attaching to the GPIO port on the RPi so you will want the wires that connect to pins 2, 3, 5 and 6. Pin #1 is usually marked with a triangle on the connector. Corresponding wire #1 is usually marked with a red stripe. Once you have located the correct 4 wires and removed the rest (cut them shorter), strip a bit of the ends off the wires you do want. About 1/8” to ¼” should be fine. Insert the wires through the connection points on the RTC, matching the appropriate functions as indicated previously and solder them in place. Cut off any extending ends after soldering.

If your RTC already has a connector mounted then you may have to get inventive and figure out what parts/skills you have and what you are going to do. If you are extremely lucky the RTC will come with a connection cable already designed for the RPi, but this is pretty unlikely. If your RTC has a connector rather than solder points and you can make ribbon cable connectors then go for it!

As mentioned the RTC is basically the inexpensive item in this scenario so if you are going to solder anything, then solder to the RTC and not the RPi. You may decide later you want to remove the RTC or want to attach other devices so you really do not want to solder directly to the RPi.

Warning: I read this warning on one site, I’m not sure of the validity but I will repost it here just to be safe. You should not connect a RTC to your RPi and power it up without a battery in place on the RTC. It may damage the RTC chip. I hope not as I’m not sure what would happen when the RTC battery dies.

You should not have the battery installed when soldering as you could damage the battery with the heat and cause leakage. This is obviously not a good thing. Once the connections have been made, install the battery in your RTC.
Never install or remove hardware with the power connected. Always pull the power plug or switch off all devices when making hardware changes.

This concludes the hardware section.

Software Installation

In order to actually use the RTC you must install some software. At this point I will assume that you have a working RPi and you can access it either directly with a monitor / keyboard / mouse or are accessing it remotely through SSH, VNC or some other package. These instructions assume you have root (admin) user access or the password, and that the RTC is now connected.

You should also be familiar with a basic text editor that is used to modify the files as required below. There are many tutorials on Vi, ViM, Nano and other text editors that come with Raspbian. Other distribution packages of Linux for the RPi will be similar, but I will document the steps that I specifically followed for Raspbian.

The Raspbian release "2012-10-28-wheezy-raspbian" or later should already include some of the files that you need that are referenced in older installation documents. Specifically, there should already be a script "/etc/init.d/hwclock.sh" on your system. If you are running and older release or do not have this file then you should use a different installation document, or update your software. The commands issued below represent what is required on a “clean” system. You may have already performed some of the steps installing other devices or software and if so, then you can skip doing them again.

Logging into your RPi as the root user simplifies the commands required but if necessary use the “sudo” command to preface the statements. Google search “sudo” or type “man sudo” for more information. This command allows you to run specific actions as if you were logged in as the root user which is necessary to make certain system level changes. For example, if you need to run the command “apt-get update” and are not logged in as the root user, then you can issue the command “sudo apt-get update” and the system will run the command with root privileges.

Although not technically required, I strongly suggest that you consider updating all your packages to the most current level. If for some reason you are unable to do so, you should at least consider updating the required packages. The commands and methods here reflect using “2013-09-10-wheezy-raspbian.img”.

In Raspbian, and other Debian Linux-like distributions, you can use the “apt-get” command to install or update the software on your system. To update all your installed software packages, you can issue the following two commands:

apt-get update

apt-get upgrade
These commands will go to a predetermined web site and search for and install any updates that are available. It may take a few minutes to run and obviously requires a connection to the internet.

By default, the Raspbian distribution disables and blocks certain modules from being loaded as a safety measure. You will need to unblock the I2C module in order to use the I2C interface as that is where the RTC is connected. Edit the file /etc/modprobe.d/raspi-blacklist.conf and comment out the line:

blacklist i2c-bcm2708

by adding a # (hash tag / pound sign) at the beginning of the line. This symbolizes a comment line which will be ignored by the program but allows you to easily re-enable the block at a future date should you wish to do so. You can of course just delete the line if you wish. The line should now look like:

#blacklist i2c-bcm2708

Save the file and exit the file editor. Should you wish to verify that the file was modified correctly, you can type:

cat /etc/modprobe.d/raspi-blacklist.conf

and the file contents will be displayed.

The next step is to install the I2C software which will allow you to “see” the I2C interface and any connected devices. Use the “apt-get” command again:

apt-get install i2c-tools

apt-get install libi2c-dev

This will install the I2C tool software and the development libraries that are also required.

The next step is to modify the list of modules that will be loaded automatically. Use your text editor again and edit the file “/etc/modules”. By default you will see a couple of lines already in the file. Add the following new line to the end of the file if you have not already done so for another project or. The same line of text works for the DS1302, DS1307 and DS3231 RTC Modules as they all use the same code. Add:

rtc-ds1307

The contents of the file should now look something like:

i2c-bcm2708

i2c-dev

rtc-ds1307 <- You just added this line

Save the file and exit from the text editor.
At this point you should be able to query the RPi and determine what it can “see” on the I2C interface, assuming the RTC module is connected at this point.

Use the “i2cdetect” command to perform this task. From the command prompt type:

i2cdetect -y 1

(This command is specific to the RPi Model B, The Model A uses “-y 0”)

You should see something like the following image.

![Image showing i2cdetect output]

This is a representation of all the devices you currently have connected to your RPi. In the example above, only a simple RTC is connected. If you do have other devices then they should also show up in the list. If you run into any problems with using the RTC you could try removing (after powering off the RPi of course) all other devices until you have sorted out the issue. Most, if not all, RTC modules will show up at grid point 0x68. The “0x” refers to the hexadecimal address. If the command does not detect your RTC and shows all blanks, then it is a hardware issue. Go back and check all the connections.

Your RTC module may include other devices that will also show up in this table. For example some RTC modules also contain an eeprom chip to store the date/time values and/or a temperature sensor.

We will run the rest of the steps interactively for now. This means that if you reboot, you will have to type them in again to make your RTC work. Later, we will edit a few more files to make the RTC run automatically on a reboot.

Once you can successfully detect the RTC, you need to load a software module in order to access it. Use the “modprobe” command to perform this task. Type:

modprobe rtc-ds1307

At this point there should be a new directory that was created by the system. You should be able to find the directory “/sys/class/i2c-adapter/i2c-1” and it will contain a number of files. This directory is once again specific to the RPi Model B, the Model A will show a directory “/sys/class/i2c-adapter/i2c-0” instead.

You need to replace one of the files completely with a value that represents your RTC. The simplest way to accomplish this is to use the “echo” command and redirect the output to the intended file, since it is a very simple file.
Type:

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

This step will replace the file “new_device” with the content “ds1307 0x68”. This tells the system that your RTC is of type “DS1307” (or compatible) and can be found at hexadecimal address” 0x68 “which is where you detected it earlier. Substitute a different value if required for your specific RTC.

The majority of the work is now done. You should be able to “read” the clock and see what time it thinks it is. Keep in mind that the time may never have been set on this clock or the battery may have been removed, so initially you may get an unexpected value. This can easily be corrected.

To see what date/time the RPi thinks it is currently, at a command prompt type:

date

You should see something like:

```
root@raspberrypi:~# date
Thu Sep 26 12:49:20 EDT 2013
```

Use the “hwclock” command to read the time that the RTC thinks it is currently. Type:

```
hwclock -r
```

You should (hopefully) see something like:

```
root@raspberrypi:~# hwclock -r
Thu Sep 26 2013 12:52:05 PM EDT -0.277485 seconds
```

If you do, then congratulations! Everything is working great.

The “hwclock” command has a number of parameters that you can use. The most useful commands are to set or read the RTC clock. Here is a list of the available commands:
Let’s assume that the RTC has an incorrect date/time and we want to update it. Use the “date” command to read the RPi System Clock. Set the date/time on the RPi if necessary by issuing the “date” command followed by the date/time in the format:

```
    date [MMDDhhmm]
```

If you need more information on the “date” command, type:

```
    date --help
```

Once you have the RPi date set correctly, you can tell the Hardware Clock to be set to the System Clock values using the “hwclock –w” command. Type:

```
    hwclock -w
```

Verify that the RTC clock was updated correctly by reading it again by typing:

```
    hwclock –r
```

You should get the current date/time.

Keep in mind that the RPi is likely still using the pseudo-clock and NTP combination at this point so it should have the correct value.
Let’s look now at making these commands work after a reboot of your RPi.

You want the Hardware Clock to update your Software Clock as early as possible in your boot cycle so that the various processes start with the correct date/time. This makes log files much easier to read as well since you know when things are happening.

There are quite a few different ways that you can manage processes, their start order, dependencies, etc. Fortunately, most Linux distributions provide a similar system to handle booting up.

The easiest way to add the commands is to modify /etc/rc.local. This is where many other documents say to install the RTC commands and where you can easily add other commands for other devices. Unfortunately, rc.local is processed at the end of the boot cycle. You can do so if you wish. In fact, if you do not remove the original pseudo-hardware clock implementation or NTP, then it is fairly reasonable to make the changes in rc.local.

It isn’t too hard though to use the “rc.d” boot sequence manager to get the Hardware Clock to load and update the Software Clock earlier so let’s do that.

You could replace most of the code in the existing file “/etc/init.d/hwclock.sh” since it isn’t really useful to you or you can just add a new script and tell the system to run it when booting. Use your text editor to create/edit a file “/etc/init.d/load_RTC.sh”, or whatever name you want to give it. Copy and paste or type the following code into the script:
The script contains some commands that aren’t absolutely necessary but they will show in the log what the System Clock was set to on startup, what the Hardware Clock is set to and finally what the System Clock has been updated to. You can leave them out if you wish. The “INIT INFO” block at the top is used by the “update-rc.d” command to determine when to load this program in the startup sequence.

To tell the system to load this file as part of the “rc.d” startup process, type:

```
update-rc.d load_RTC.sh enable
```

If all went well, then the system should respond with:

“update-rc.d: using dependency based boot sequencing”

You should now be able to issue the “reboot” command and the time will be set correctly when the RPi starts up again. As previously mentioned, you can leave the default mechanisms in place relating to
setting the date/time or you can remove them. If you want to “prove” that everything is working properly then proceed with the following.

To disable the pseudo-hardware clock implementation, type:

```
update-rc.d -f fake-hwclock remove
```

To remove the pseudo-hardware clock COMPLETELY from your system rather than just disable it, type:

```
dpkg --purge fake-hwclock
```

Network Time Protocol (NTP) is a good thing to have running even with a hardware clock as it can correct minor time variances due to “drift” in the clock value. For the purposes of testing though you may want to disable it until your testing is done and then enable it again.

To disable NTP, type:

```
update-rc.d ntp remove
```

## Testing your Real Time Clock

If everything has gone well up to this point then you can put your clock to the test. Make sure to disable “fake-hwclock” and “ntp”. If you leave them running you won’t know for sure that your clock is working.

With the RPi powered off, disconnect the network connection by unplugging the cable or removing your wireless adapter if you have one. Unplug your RTC module.

Start your RPi and type “date” from a command prompt to see what date/time the RPi thinks it is. Mine shows a date/time in December of 1969. This is good. It means the previous implementations have been correctly disabled.

Power down the RPi again using the “shutdown” command of course, and connect your RTC module but leave the network unconnected.

Start your RPi and type “date” from a command prompt to see what date/time the RPi thinks it is now. If you had previously set the time on your RTC or if it came already set, then you should see the correct date/time. If it is not correct, then use the “date” command (type “date –help” for the syntax) to set the System Clock. After the System Clock is set, type “hwclock –w” or “hwclock –systohc” to update the Hardware Clock with the System Clock values. Type “hwclock –r” or “hwclock –show” to verify that the Hardware Clock has been updated. Reboot and check the date/time again. It should now be correct.

The true ultimate test though is to shutdown the RPi for a few minutes or hours. Start it back up and check the date/time. If it is still correct, then you have 100% success and your RTC is working as it should.
I suggest starting the NTP process again to ensure that the proper time is maintained.

Should you find that once in a while you need to update your Hardware Clock value then follow the directions above. If your RTC does not keep time very well and your RPi is turned off a lot, then you may need to replace the battery.

Project complete! I hope you found this information helpful.