

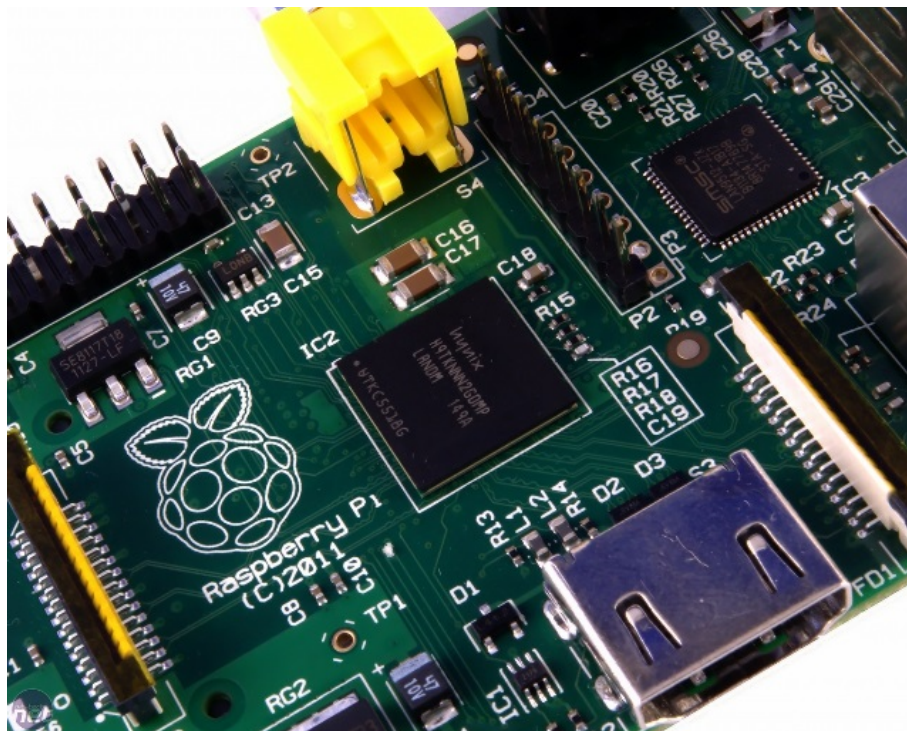
Raspberry Pi Learning Resources

Temperature Log > Worksheet

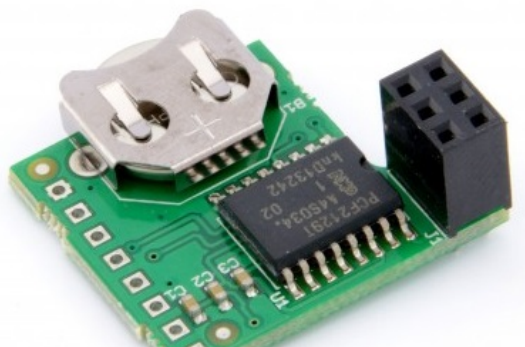
[Help](#) / [Teach](#) / [Learn](#) / [Make](#)

TEMPERATURE LOG

The system on a chip (SoC) of the Raspberry Pi has a temperature sensor that can be used to measure its temperature from the command line. It can provide information on how much heat the chip has generated during operation and also report on the temperature of the environment. This project's aim is to create a simple shell script that can run automatically as you boot up your Raspberry Pi, take measurements from the temperature sensor at given intervals, and write them into log files that can be viewed later.



If you would like accurate timestamps for your temperature logs without network access, you will also need to set up a Real Time Clock. If you are using RasClock, for example, see [Afterthought Software - RasClock](#) for information on setting it up.





CREATE A SHELL SCRIPT TO RECORD THE TEMPERATURE

The chip's temperature can be read from the command line and can also be written into a file from there. A shell script, however, can run a sequence of commands and with a bash interpreter it allows the repeated execution of this procedure. We will therefore use a shell script to record the temperature into a log file at given intervals.

VIEWING THE TEMPERATURE

To view the current temperature of your Raspberry Pi, you can just type the following into the command line: `/opt/vc/bin/vcgencmd measure_temp`.

```
pi@raspberrypi:~$ /opt/vc/bin/vcgencmd measure_temp
temp=37.9'C
```

CREATING A SHELL SCRIPT

When writing a shell script, the first line contains information on what interpreter should be used to run the script. In this case we will use a bash script, since that allows the usage of loops which will be useful for our repeated measurements. To specify this, the first line of the script will be:

```
#!/bin/bash
```

If you would like to edit the script directly in the command line, you can use the built-in basic text editor, nano:

1. Type `nano temperature_log.sh` to create a file named `temperature_log.sh` and edit it. If the file already exists, nano will just let you edit it.
2. Use the cursor to move and select where you want to write the code into the file.
3. When you have finished editing the file, press `Ctrl` + `O` to attempt to save the file.
4. Press `Enter` to save the file (nano allows you to change the file name before this, should you wish to).
5. Press `Ctrl` + `X` to exit nano.

For more information on how to use the nano editor, please type `man nano` or see [nano's online documentation](#).

The following lines can contain a sequence of commands, so we can just write the command for viewing the temperature after the first line to test it:

```
#!/bin/bash

/opt/vc/bin/vcgencmd measure_temp
```

In order to be able to actually run the script, you will need to set its permission accordingly. Type

`chmod +x temperature_log.sh` to give the file permission to be executed. After that, you can run the script by typing `./temperature_log.sh`, provided that the script is in the directory you are currently in. For example, if it is in your home directory, you can just type `~/temperature_log.sh` to run it from anywhere.

REPEATEDLY VIEWING THE TEMPERATURE

In order to record the temperature continuously, we will need to use a loop in our shell script. There are two options: you can either create a loop that repeats indefinitely, or one that repeats a certain number of times, depending on what you would like to use it for.

The following will execute the code between `do` and `done` indefinitely:

```
while :
do
    /opt/vc/bin/vcgencmd measure_temp
done
```

Whereas this will repeatedly execute the code between `do` and `done` 30 times:

```
for i in {1..30}
do
    /opt/vc/bin/vcgencmd measure_temp
done
```

This loop now repeats the measurement, but it does not wait in between the measurements; it just takes the measurements immediately, one after the other. To take measurements only at specified intervals, we will need to wait a certain amount of time after each measurement. This can be done by inserting the line `sleep 10` just after the line for viewing the temperature; this will cause the script to wait for 10 seconds each time.

Here is an example script:

```
#!/bin/bash

while :
do
    /opt/vc/bin/vcgencmd measure_temp
    sleep 10
done
```

WRITING INTO A FILE

The `echo` command allows you to write something into the command line, and by setting a file as its output you can also write into text files with it. The following example first prints "test", then writes "test" into the file `test.txt`, and finally shows the contents of `test.txt`:

```
pi@raspberrypi:~$ echo test
test
pi@raspberrypi:~$ echo test >test.txt
pi@raspberrypi:~$ cat test.txt
test
```

In order to record multiple temperatures in a single file, we will need to append a new line to the file instead of rewriting it. This can be achieved by typing `echo test >>test.txt`, for example.

CREATING A TIMESTAMP

When creating log files, it is a good idea to take a timestamp that will be included in the log file's name, so that we can identify the different logs easily. If you don't have a Real Time Clock or network access the time will not be correct, but it will still maintain the log files in the correct order.

If you type the command `date +%F_%H-%M-%S`, it will give you the current date and time as known to the Raspberry Pi in a format that is readable. This can be used as part of a filename and will also be ordered correctly. For example: `2014-07-30_10-59-56`. In this command, `%F` stands for the full date in a format such as `2014-07-30`, `%H` stands for the hour in the range `00..23`, `%M` stands for the minute in the range `00..59`, and `%S` stands for the second in the range `00..59`.

First, we will tell the interpreter to execute the `date` command, and then use the command's output by putting it between backticks: ``date +%F_%H-%M-%S``. Then we can create a variable which we will call `timestamp`, and store the result in it: `timestamp=`date +%F_%H-%M-%S``.

ADDING THE TIMESTAMP TO THE LOG

At the beginning of the script we will create the log file, initialised with a small header line.

Type `echo "Temperature Log - $(date)" >/home/pi/logs`
`/temperature_log_$timestamp.txt` to create a log file called `temperature_log_<timestamp>` in the directory `/home/pi/logs/`. Note that we cannot reference the home directory as `~` if we choose to run the script before logging in. You will have to create the directory beforehand, for example with `mkdir ~/logs`.

In the above command, `$(date)` returns the date in the default format and `$timestamp` returns the value of the variable called `timestamp`.

We can use a similar technique to store the temperature inside the loop in a variable, by writing `temp=`/opt/vc/bin/vcgenclmd measure_temp``. After that, optionally, we can get rid of the `temp=` part of the temperature measurement's result by typing `temp=${temp:5:16}`; this will take the variable's value, starting from the 5th character, for up to 16 characters. This value can then be appended to the file by writing: `echo $temp >>/home/pi/logs`
`/temperature_log_$timestamp.txt`.

Here is an example script:

```
#!/bin/bash

timestamp=`date +%F_%H-%M-%S`
echo "Temperature Log - $(date)" >/home/pi/logs/temperature_log_$timestamp.txt
while :
do
    temp=`/opt/vc/bin/vcgenclmd measure_temp`
    temp=${temp:5:16}
    echo $temp >>/home/pi/logs/temperature_log_$timestamp.txt
    sleep 10
done
```

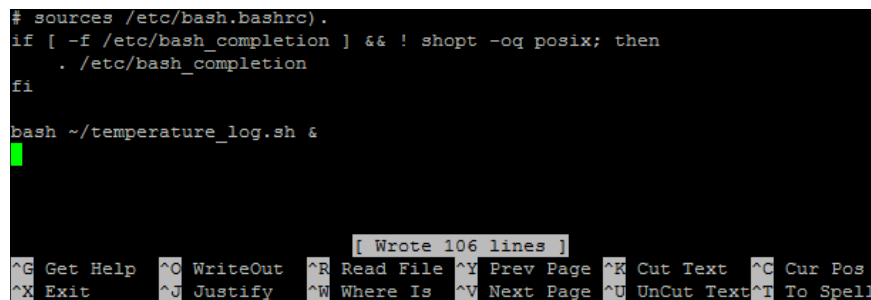


AUTOMATE YOUR SHELL SCRIPT AT STARTUP

If you would like this script to be run automatically when you boot up your Raspberry Pi, you will need to run it from one of the scripts that get executed at startup. You have two options: you can either add it to your `~/.bashrc` and then it will run when you log in, or you can add it to `/etc/rc.local` to make it run automatically while booting, before you have to log in. **Please be careful when editing those files (especially in the second case), as your Raspberry Pi will not boot up properly if the scripts are modified incorrectly!**

OPTION 1: RUN AUTOMATICALLY WHEN YOU LOG IN USING .BASHRC

Type `nano ~/.bashrc` to open the file for editing (you can use any text editor you like), and add the following line to the end of the file: `bash ~/temperature_log.sh &`. `bash` will execute the shell script provided as its argument and the `&` character at the end ensures that the script will run in the background.



```
# sources /etc/bash.bashrc).
if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
  . /etc/bash_completion
fi

bash ~/temperature_log.sh &
```

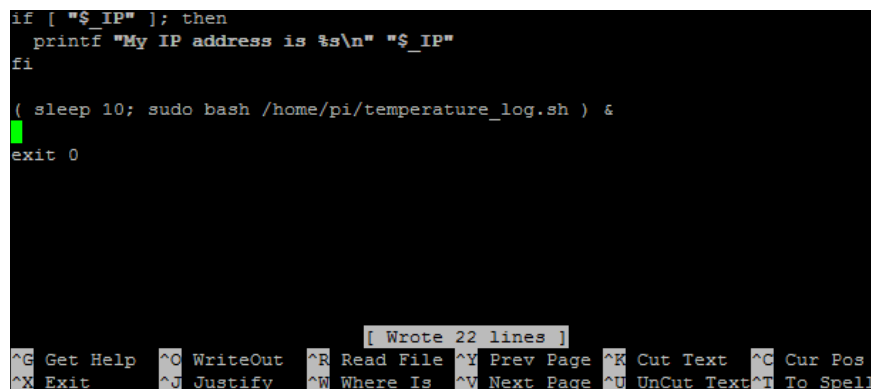
[Wrote 106 lines]

^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

Make sure not to miss the `&` character at the end, otherwise the script will not run in the background and will make the booting process get stuck in a loop!

OPTION 2: RUN AUTOMATICALLY WHILE BOOTING USING RC.LOCAL

Type `sudo nano /etc/rc.local` to open the file for editing (you can use any text editor you like), and add the following line just before the `exit 0` at the end of the file: `(sleep 10; sudo bash /home/pi/temperature_log.sh) &`. `bash` will execute the shell script provided as its argument and the `&` character at the end ensures that the script will run in the background. The parentheses group the `sleep` and `bash` commands together, so that execution begins with a 10 second delay.



```
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

( sleep 10; sudo bash /home/pi/temperature_log.sh ) &

exit 0
```

[Wrote 22 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

Make sure not to miss the `&` character at the end, otherwise the script will not run in the background and will make the booting process get stuck in a loop!



FINAL SHELL SCRIPT

Your final shell script should look something like this:

```
#!/bin/bash

timestamp=`date +%F_%H-%M-%S`
echo "Temperature Log - $(date)" >/home/pi/logs/temperature_log_${timestamp}.txt
while :
do
    temp=`/opt/vc/bin/vcgencmd measure_temp`
    temp=${temp:5:16}
    echo $temp >>/home/pi/logs/temperature_log_${timestamp}.txt
    sleep 10
done
```



WHAT NEXT?

Try another shell script using the Raspberry Pi for temperature measurements in intervals of 5 minutes.

The following example script can be used to take a temperature measurement every 10 seconds for 5 minutes and then shut down, while also printing out information about the measurements:

```
#!/bin/bash

echo "Starting to record the temperature"
timestamp=`date +%F_%H-%M-%S`
echo "Writing into /home/pi/logs/temperature_log_${timestamp}.txt"
echo "Temperature Log - $(date)" >/home/pi/logs/temperature_log_${timestamp}.txt
for i in {1..30}
do
    temp=`/opt/vc/bin/vcgencmd measure_temp`
    temp=${temp:5:16}
    echo $temp >>/home/pi/logs/temperature_log_${timestamp}.txt
    echo "Recorded temperature #${i}:"
    tail -1 /home/pi/logs/temperature_log_${timestamp}.txt
    sleep 10
done
echo "Finished recording the temperature, shutting down"
sudo shutdown -h now
```

This learning resource is provided for free by the [Raspberry Pi Foundation](https://www.raspberrypi.org/) under a [Creative Commons](https://creativecommons.org/licenses/by/4.0/) licence. Find more at [raspberrypi.org/resources](https://www.raspberrypi.org/resources) and github.com/raspberrypilearning.