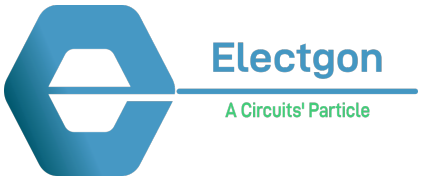

Raspberry Pi

With RealTime Patch

ELECTGON
www.electgon.com
ma_ext@gmx.net

01.05.2018

Contents

Abstract

Realtime applications are used widely in many industrial fields. It is connected more with embedded devices which are built mainly on ARM processors. Raspberry Pi is using ARM processor as its core CPU. Thus, using Raspberry Pi for developing or prototyping Realtime applications is reasonable approach specially with the low cost of the Raspberry Pi. This document illustrates then how to use Realtime patch in Raspberry Pi and issues that can be found during development.

1 Introduction

Raspbian is built for Raspberry Pi as an operating system based on native Linux. This means, user has the feasibility to tweak Linux kernel to customize it according to his final system preferences. Normally, Raspbian is provided on Raspberry Pi website in two versions; Lite version with only console interface and another one with GUI interface. Both versions are based in Linux without Realtime patch. Although this will perform actually with high efficiency in terms of latency and multitasking, However more harness sometimes is needed for critical Realtime applications. Applying Realtime patch to kernel of Raspbian will adopt and give more accuracy for the performance of Raspbian as will be shown in this document. Therefore, steps needed for applying Realtime patch will be discussed here. As a prerequisite, steps mentioned here assume that reader has already known about Raspbian and how to install it as discussed in previous tutorial.

Realtime patch used here is PREEMPT_RT which is a patch that can be applied to a Linux kernel so that the kernel will have more hard time requirements. Linux kernel without this patch can be considered also as a Realtime kernel. The PREEMPT_RT adds more restrictions on the latency of performed process as will be shown later.

2 Install Evaluation Packages

In order to be able to evaluate performance of Raspbian before and after applying Realtime Patch, we need to install some packages that can be used to measure the performance. We will use Two packages Cyclictest and LTP.

2.1 Cyclictest

Cyclictest is used to measure latency of the operating system. Download and build it by

```
$git clone git://git.kernel.org/pub/scm/linux/kernel/git/clrkwllms/rt-tests.git
$cd rt-tests make all
$sudo cp ./cyclictest /usr/bin/
```

Cyclictest can be used as

```
$sudo cyclictest -t5 -p80 -n -i10000 -l10000
```

Which means that cyclictest will generate 5 threads with priority 80. Each thread will be iterated 10000, each iteration has 10000 loop. So Cyclic test is calculating minimum, average and maximum latency of each iteration.

2.2 LTP

LTP has various scripts that can be used to test specific behavior of the operating system. So we can measure Scheduling Overhead, Context Switching, Time to create and start a thread, interprocess communication, etc. LTP can be installed using the following steps

```
$wget http://downloads.sourceforge.net/ltp/ltp-full-20150420.tar.bz2
$tar xvf ltp-full-20150420.tar.bz2
$cd ltp-full-20150420
$./configure --with-realtime-testsuite
$make
```

Note

You may face the following warning while installing any package

```
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
    LANGUAGE = (unset),
    LC_ALL = (unset),
    LC_TIME = "de_DE.UTF-8",
    LC_MONETARY = "de_DE.UTF-8",
    LC_ADDRESS = "de_DE.UTF-8",
    LC_TELEPHONE = "de_DE.UTF-8",
    LC_NAME = "de_DE.UTF-8",
    LC_MEASUREMENT = "de_DE.UTF-8",
    LC_IDENTIFICATION = "de_DE.UTF-8",
    LC_NUMERIC = "de_DE.UTF-8",
    LC_PAPER = "de_DE.UTF-8",
    LANG = "en_GB.UTF-8"
```

To get rid of this warning open the following file

```
$sudo nano /etc/default/locale
```

then add this line

```
LC_ALL=en_GB.UTF-8
```

3 Adding Realtime Features

As mentioned before, although Raspbian version is highly optimized, however it can miss some realtime requirements. Figure 1 shows latency of Raspbian without Realtime patch.

```
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -i10000 -l10000 -n
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.22 0.25 0.15 1/116 15509

T: 0 (15505) P:80 I:10000 C:  10000 Min:     15 Act:   18 Avg:   21 Max:      96
T: 1 (15506) P:80 I:10500 C:   9524 Min:     16 Act:   28 Avg:   24 Max:     257
T: 2 (15507) P:80 I:11000 C:   9091 Min:     16 Act:   28 Avg:   21 Max:     129
T: 3 (15508) P:80 I:11500 C:   8696 Min:     16 Act:   23 Avg:   21 Max:     103
T: 4 (15509) P:80 I:12000 C:   8333 Min:     15 Act:   26 Avg:   20 Max:      90
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -i10000 -l10000 -n
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.05 0.18 0.14 1/116 15523

T: 0 (15519) P:80 I:10000 C:  10000 Min:     16 Act:   42 Avg:   21 Max:     124
T: 1 (15520) P:80 I:10500 C:   9524 Min:     14 Act:   18 Avg:   20 Max:     102
T: 2 (15521) P:80 I:11000 C:   9091 Min:     15 Act:   31 Avg:   19 Max:     105
T: 3 (15522) P:80 I:11500 C:   8696 Min:     16 Act:   28 Avg:   30 Max:     583
T: 4 (15523) P:80 I:12000 C:   8334 Min:     14 Act:   18 Avg:   19 Max:      92
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -i10000 -l10000 -n
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.02 0.12 0.13 1/117 15539

T: 0 (15532) P:80 I:10000 C:  10000 Min:     11 Act:   22 Avg:   25 Max:     162
T: 1 (15533) P:80 I:10500 C:   9523 Min:     11 Act:   96 Avg:   23 Max:     132
T: 2 (15534) P:80 I:11000 C:   9091 Min:     11 Act:   29 Avg:   25 Max:     192
T: 3 (15535) P:80 I:11500 C:   8695 Min:     10 Act:   24 Avg:   21 Max:     108
T: 4 (15536) P:80 I:12000 C:   8333 Min:     10 Act:   18 Avg:   20 Max:     146
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -i10000 -l10000 -n
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.12 0.14 0.14 1/116 15552

T: 0 (15548) P:80 I:10000 C:  10000 Min:     16 Act:   23 Avg:   21 Max:      88
T: 1 (15549) P:80 I:10500 C:   9524 Min:     14 Act:   19 Avg:   21 Max:      91
T: 2 (15550) P:80 I:11000 C:   9091 Min:     16 Act:   29 Avg:   20 Max:     127
T: 3 (15551) P:80 I:11500 C:   8696 Min:     18 Act:   23 Avg:   27 Max:     245
T: 4 (15552) P:80 I:12000 C:   8333 Min:     14 Act:   17 Avg:   20 Max:      82
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -i10000 -l10000 -n
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.11 0.15 0.14 1/117 15566

T: 0 (15561) P:80 I:10000 C:  10000 Min:     18 Act:   29 Avg:   26 Max:     259
T: 1 (15562) P:80 I:10500 C:   9524 Min:     15 Act:   17 Avg:   21 Max:      92
T: 2 (15563) P:80 I:11000 C:   9091 Min:     15 Act:   29 Avg:   20 Max:      77
T: 3 (15564) P:80 I:11500 C:   8696 Min:     14 Act:   17 Avg:   20 Max:      95
T: 4 (15565) P:80 I:12000 C:   8333 Min:     14 Act:   24 Avg:   20 Max:     111
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -i10000 -l10000 -n
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.16 0.16 0.14 1/117 15580

T: 0 (15576) P:80 I:10000 C:  10000 Min:     16 Act:   50 Avg:   29 Max:     180
T: 1 (15577) P:80 I:10500 C:   9524 Min:     15 Act:   17 Avg:   19 Max:     162
T: 2 (15578) P:80 I:11000 C:   9091 Min:     15 Act:   26 Avg:   20 Max:      95
T: 3 (15579) P:80 I:11500 C:   8696 Min:     14 Act:   17 Avg:   19 Max:      71
T: 4 (15580) P:80 I:12000 C:   8334 Min:     15 Act:   17 Avg:   19 Max:      61
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -i10000 -l10000 -n
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.03 0.10 0.13 1/117 15594

T: 0 (15589) P:80 I:10000 C:  10000 Min:     15 Act:   20 Avg:   20 Max:      69
T: 1 (15590) P:80 I:10500 C:   9524 Min:     15 Act:   17 Avg:   20 Max:      88
T: 2 (15591) P:80 I:11000 C:   9091 Min:     15 Act:   29 Avg:   20 Max:      78
T: 3 (15592) P:80 I:11500 C:   8696 Min:     16 Act:   28 Avg:   29 Max:     227
T: 4 (15593) P:80 I:12000 C:   8333 Min:     15 Act:   22 Avg:   20 Max:      86
```

Figure 1: Raspbian Latency Without Realtime Patch

You can notice that there are some over latency recorded. This is what we meant previously that Linux is a Realtime system but it can miss some timing. Therefore you may need to control this latency. This can be done by configuring the kernel. First we have to define which requirement we need for our Realtime performance. In this tutorial, performance added by PREEMPT_RT is sufficient. So all what we need is to apply this patch to a Linux kernel and

use this kernel instead of current Raspbian kernel.

Not any kernel version can work in Raspberry Pi, e, only ported kernel can work with Raspberry Pi board. That is because processor architecture should be specified in the kernel machines. Fortunately, you can find Linux kernel ported for Raspberry Pi provided at their website. Use the following command to get this kernel

```
$git clone --depth=1 https://github.com/raspberrypi/linux
```

Open the downloaded folder, we need now to apply PREEMPT_RT patch to that kernel. To do that, we need to know first which version of the kernel we have downloaded. You can know that by opening Makefile and see VERSION, PATCHLEVEL, SUBLEVEL at the beginning of the file. Those indicate version of the kernel. For example, I had downloaded this when writing this document

VERSION = 4

PATCHLEVEL = 1

SUBLEVEL = 15

EXTRAVERSION =

NAME = Series 4800

which means that this Linux kernel is 4.1.15. knowing that we can download matching PREEMPT_RT patch from the following link

https://www.kernel.org/pub/Linux/kernel/projects/rt/

Extract the downloaded patch

```
$gunzip patch-4.1.13-rt15.patch.gz
```

Apply the patch using

```
$cd linux
$patch -p1 -b < ../path/to/patch-4.1.13-rt15.patch
```

I am assuming now that you are preparing this realtime version in another host machine (not in Raspberry Pi). Which means that we need cross compiler in order to be able to compile the new kernel.

Next is we need to make sure that we have cross compiler to compile this kernel for ARM processor. For that purpose, you can use Codesourcery provided by MentorGraphics which is available at the following link

https://sourcery.mentor.com/sgpp/lite/arm/portal/kbentry62

However it is advised to use linaro cross compiler which can be installed by getting source files

```
$git clone https://github.com/raspberrypi/tools
```

Then add */path/to/downloadDirectory/gcc-linaro-arm-linux-gnueabihf-raspbian/bin* to your $PATH in the .bashrc in your home directory (or .profile or in etc/profile or etc/environment.).

Kernel is ready now for compilation, type the following commands

Create a folder to install compiled modules in it

```
$mkdir my_modules_folder
```

Then go to downloaded Linux folder

```
$cd linux
$export INSTALL_MOD_PATH=../my_modules_folder
$export KERNEL=kernel7 ##(with small 'k' in kernel7)
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bcm2709_defconfig
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig
```

note that bcm2709_defconfig is the configuration file for the Raspberry Pi2.

Most important at this step is to enable PREEMPT_RT feature. This can be found in kernel feature section. You can then modify other features in the kernel as you need. After finishing modifications in the kernel, start the compile process

```
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- zImage
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- dtbs
$make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- modules_install
```

Then you have to generate the compiled kernel. While you are in Linux directory execute the following command

```
$./scripts/mkknlimg ./arch/arm/boot/zImage $INSTALL_MOD_PATH/$KERNEL.img
```

Prepare now files and folder that will be transferred to the Raspbian image

```
$mkdir $INSTALL_MOD_PATH/boot
$cp ./arch/arm/boot/dts/*.dtb $INSTALL_MOD_PATH/boot/
$cp -r ./arch/arm/boot/dts/overlays $INSTALL_MOD_PATH/boot
```

This compilation process should go fine. If successfully finished, start to transfer this compiled kernel to the raspbian installed image. Insert SD card which has this installed Raspbian into your hosting machine. open your SD card to execute the following commands

```
###Make Backup of the Original Kernel Files#######
$cd /boot
$sudo mkdir OriginalKernelFiles
$sudo cp kernel7.img OriginalKernelFiles/
$sudo cp *.dtb OriginalKernelFiles/
$sudo cp -r overlays OriginalKernelFiles/

###Make Backup of the Original Modules Files#######
$cd /lib sudo mkdir OriginalKernelFiles
$sudo cp -r modules OriginalKernelFiles/
$sudo cp -r firmware OriginalKernelFiles/

####Transfer Compiled Kernel####
$cd
$mkdir CompiledKernel
```

now in the host machine where we did cross compilation, we should transfer these compiled kernel

```
$scp -rp $INSTALL_MOD_PATH/* pi@192.168.147.xxx:/home/pi/CompiledKernel
```

In Raspberry Pi, compiled kernel shall be transferred now. Install it

```
####Install Compiled Kernel####
$sudo cp /home/pi/CompiledKernel/my_modules_folder/kernel7.img /boot/kernel7.
    img
$sudo cp /home/pi/CompiledKernel/my_modules_folder/boot/*.dtb /boot/.
$sudo cp -r /home/pi/CompiledKernel/my_modules_folder/boot/overlays/*  /boot/
    overlays/.
$sudo cp -r /home/pi/CompiledKernel/my_modules_folder/lib/modules/*  /lib/
    modules/.
$sudo cp -r /home/pi/CompiledKernel/my_modules_folder/lib/firmware/*  /lib/
    firmware/.
```

Note that:

- To copy kernel from host machine to Raspberry Pi we used -rp option.
- It is advised to run “sync” command before removing the SD card from host machine.

You should have Raspbian with PREEMPT_RT kernel now. To make sure that everything has been setup correctly, you can restart the RaspberryPi and run the following command

```
$dmesg | grep error
```

You shouldn’t get any recorded error messages. Further you can now check performance of your realtime Raspbian by running Cyclictest to test latency of the system. The following Results in figure 2 are expected.

```
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -n -i10000 -l10000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.23 0.17 0.15 1/151 29737

T: 0 (29352) P:80 I:10000 C:  10000 Min:     13 Act:   21 Avg:   19 Max:      94
T: 1 (29353) P:80 I:10500 C:   9524 Min:     13 Act:   16 Avg:   20 Max:      81
T: 2 (29354) P:80 I:11000 C:   9091 Min:     13 Act:   19 Avg:   20 Max:     128
T: 3 (29355) P:80 I:11500 C:   8696 Min:     13 Act:   14 Avg:   20 Max:      87
T: 4 (29356) P:80 I:12000 C:   8334 Min:     13 Act:   14 Avg:   20 Max:      85
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -n -i10000 -l10000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.18 0.15 0.14 1/154 30149

T: 0 (29758) P:80 I:10000 C:  10000 Min:     14 Act:   21 Avg:   22 Max:      91
T: 1 (29759) P:80 I:10500 C:   9524 Min:     13 Act:   21 Avg:   19 Max:      95
T: 2 (29760) P:80 I:11000 C:   9091 Min:     13 Act:   19 Avg:   20 Max:      85
T: 3 (29761) P:80 I:11500 C:   8696 Min:     12 Act:   16 Avg:   20 Max:     117
T: 4 (29762) P:80 I:12000 C:   8334 Min:     13 Act:   15 Avg:   19 Max:      80
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -n -i10000 -l10000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.39 0.21 0.16 1/153 30594

T: 0 (30206) P:80 I:10000 C:  10000 Min:     13 Act:   16 Avg:   19 Max:     103
T: 1 (30207) P:80 I:10500 C:   9524 Min:     13 Act:   19 Avg:   20 Max:     122
T: 2 (30208) P:80 I:11000 C:   9091 Min:     13 Act:   21 Avg:   20 Max:     110
T: 3 (30209) P:80 I:11500 C:   8695 Min:     13 Act:   23 Avg:   20 Max:     126
T: 4 (30210) P:80 I:12000 C:   8333 Min:     13 Act:   20 Avg:   20 Max:      79
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -n -i10000 -l10000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.36 0.22 0.17 1/153 31059

T: 0 (30675) P:80 I:10000 C:  10000 Min:     13 Act:   17 Avg:   19 Max:      99
T: 1 (30676) P:80 I:10500 C:   9524 Min:     11 Act:   15 Avg:   20 Max:      88
T: 2 (30677) P:80 I:11000 C:   9091 Min:     13 Act:   22 Avg:   20 Max:     102
T: 3 (30678) P:80 I:11500 C:   8695 Min:     13 Act:   20 Avg:   21 Max:     112
T: 4 (30679) P:80 I:12000 C:   8333 Min:     13 Act:   17 Avg:   20 Max:      92
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -n -i10000 -l10000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.39 0.27 0.19 1/151 31477

T: 0 (31092) P:80 I:10000 C:  10000 Min:     13 Act:   14 Avg:   20 Max:      85
T: 1 (31093) P:80 I:10500 C:   9524 Min:     13 Act:   15 Avg:   21 Max:     120
T: 2 (31094) P:80 I:11000 C:   9091 Min:     13 Act:   23 Avg:   20 Max:      92
T: 3 (31095) P:80 I:11500 C:   8695 Min:     13 Act:   13 Avg:   20 Max:      87
T: 4 (31096) P:80 I:12000 C:   8333 Min:     13 Act:   20 Avg:   20 Max:     130
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -n -i10000 -l10000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.10 0.20 0.17 1/152 31900

T: 0 (31510) P:80 I:10000 C:  10000 Min:     13 Act:   25 Avg:   19 Max:     112
T: 1 (31511) P:80 I:10500 C:   9523 Min:     13 Act:   26 Avg:   19 Max:      89
T: 2 (31512) P:80 I:11000 C:   9091 Min:     13 Act:   32 Avg:   21 Max:      80
T: 3 (31513) P:80 I:11500 C:   8695 Min:     13 Act:   24 Avg:   20 Max:      85
T: 4 (31514) P:80 I:12000 C:   8333 Min:     13 Act:   26 Avg:   20 Max:      86
pi@raspberrypi:~ $ sudo cyclictest -p80 -t5 -n -i10000 -l10000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.18 0.21 0.18 1/153 32346

T: 0 (31957) P:80 I:10000 C:  10000 Min:     13 Act:   18 Avg:   22 Max:     101
T: 1 (31958) P:80 I:10500 C:   9524 Min:     13 Act:   15 Avg:   21 Max:     118
T: 2 (31959) P:80 I:11000 C:   9091 Min:     13 Act:   23 Avg:   21 Max:     118
T: 3 (31960) P:80 I:11500 C:   8696 Min:     12 Act:   14 Avg:   21 Max:     137
T: 4 (31961) P:80 I:12000 C:   8333 Min:     13 Act:   21 Avg:   27 Max:     139
pi@raspberrypi:~ $
```

Figure 2: Raspbian Latency With Realtime Patch

There is also Ubuntu Mate that can be used with Raspberry Pi. The following results were obtained after applying Preempt patch for Ubuntu Mate.

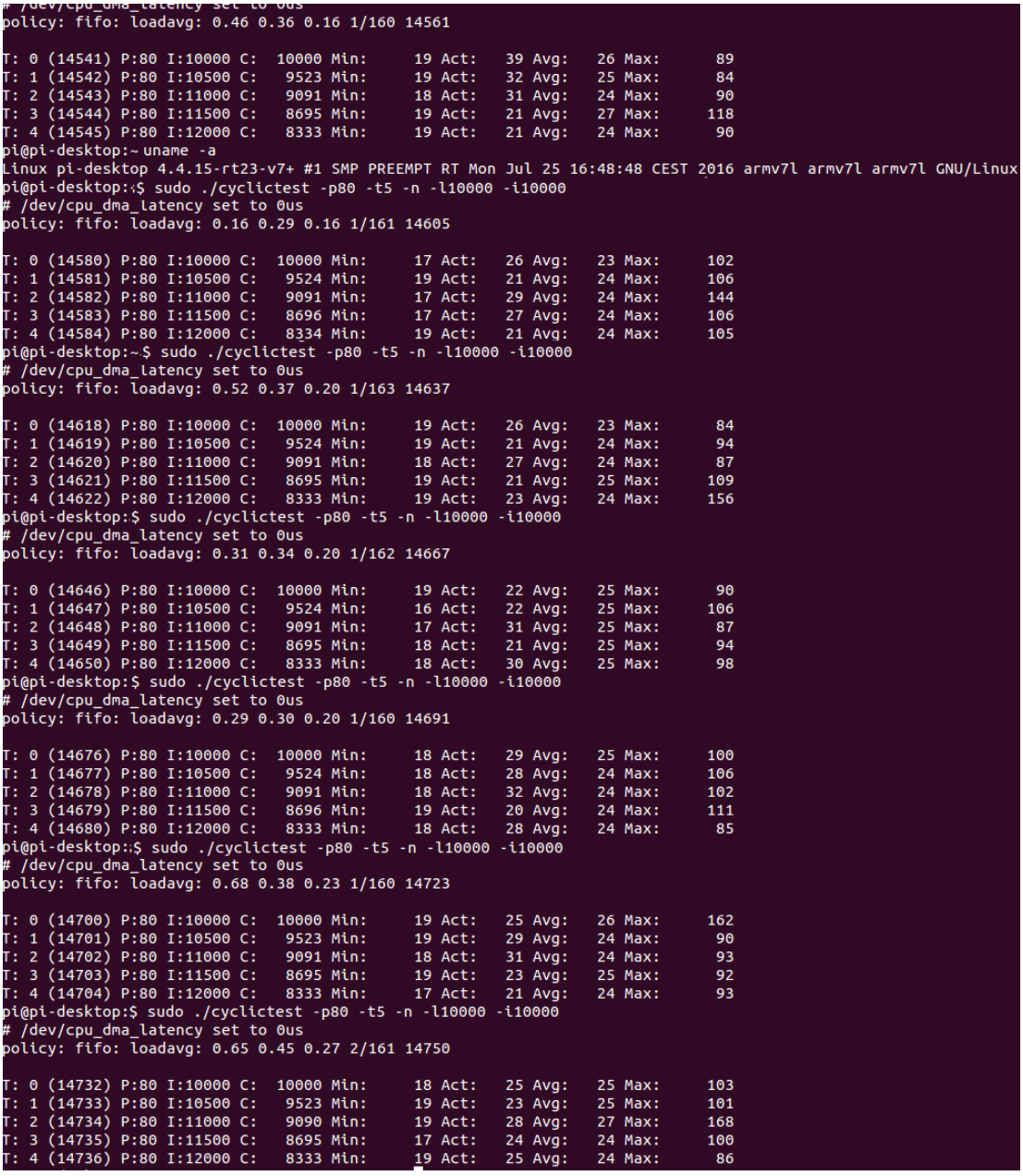

Figure 3: Ubuntu Mate Latency With Realtime Patch

Installing Ubuntu Mate is quite simple like Raspbian. Download it from Raspberry Pi website. Then install it in your SD Card

```
$sudo dd bs=4M if=ubuntu-source-file.img of=/dev/sdb
##(note it is sdb or whatever the driver of the sd Card)
```

however Ubuntu mate doesn't have default user account, so after installation, we have to connect Raspberry Pi with a Monitor to create a user account and prepare needed settings (Time, Keyboard, etc).

After that we need to expand the image on the full disk size

```
$sudo fdisk /dev/mmcblk0
```

This fdisk utility is used to reformat partitions of the SD Card, so run the following commands

```
d
2
n
p
2
ENTER
ENTER
w
```

then reboot the system and issue the following command

```
$sudo resize2fs /dev/mmcblk0p2
```

It is advised also to disable the graphical interface of ubuntu mate and work with a console interface. This can be done using this command

```
$graphical disable
```

4 RealTime Patch Issues

Adding this realtime patch wasn't that easy task. One problem has been found that took a lot of time to be figured out. This problem is Raspberry Pi board is "freezing" i.e. it is hanging and stops working. By doing more investigation about this problem, it has been found that it happens only when the realtime patch is applied to the kernel. Which can lead to a hint that there is a mis-compatibility problem between the operating system (Raspbian) and the compiled kernel (Linux kernel) and the applied patch (PREEMPT-RT). This incompatibility may be due to incompatibility in used packages and libraries, or it is incompatibility between hardware modules and firmware.

Another suggestion was pointing out that this freezing behavior is because of lack of power resources in Raspberry Pi since its allowed voltage supply is 2 volt. The 4 USB Interfaces and Ethernet Interface are transmitting/receiving data on the same bus (i.e. transmission rate is distributed over these interfaces). These interfaces are driven by one driver which makes this drive is a power consuming one. Not only but also the USB interface are driving also attached devices. This powering issue was thought as a cause of the problem because inappropriate power levels in the Raspberry Pi makes it stop working.

A Third suggestion is referring this freezing behavior to the configuration of the PREEMPT-RT patch. This patch is using Spinlock for sharing resource between processes instead of using mutexes. i.e. normal Linux kernel is using mutexes, by applying PREEMPT-RT patch, it will replace this mechanism by using Spinlock. Not only but also other configurations in the kernel like 'panic' behavior of the kernel. The kernel can be set in case hang-up or panic situation to reboot automatically, or just wait, or reboot after some time. Other setting like detect if there is a panic or not...etc. All these configurations may cause disturbance in the system after applying the PREEMPT- RT patch.

To sum up causes of this freezing problem, it might be because of

- Mis-compatibility.

- Lack of Power Resources.
- PREEMPT-RT Configuration.

Other causes might be possible also.

Bibliography

[1] https://www.raspberrypi.org/

[2] http://www.frank-durr.de/?p=203