Lab4 Week1 & Week2 Report

Names: Zixiao Wang, Yue Zhang
NetID: zw579, yz2455
Date: Nov 17, 2019

Introduction:

In lab4, we explored the PreemptRT kernel patch through following steps:

1. On the current system, we implemented a square wave signal in C and measured its performance.
2. Installed a new kernel patched with PreemptRT and compiled it.
3. On the new system, run the square wave to check installation and test performance of the real time system.

Design and Test:

Week 1

1. <u>Performance measurement using cyclictest</u>
   - Download and install Cyclictest from Github:

```
pi@yz2455zw579:~/lab4/rt-tests $ sudo cyclictest --help
sudo: unable to resolve host yz2455zw579: Name or service not known
cyclictest V 0.92
Usage:
cyclictest <options>

-a [NUM] --affinity        run thread #N on processor #N, if possible
                           with NUM pin all threads to the processor NUM
-A USEC  --aligned=USEC    align thread wakeups to a specific offset
-b USEC  --breaktrace=USEC send break trace command when latency > USEC
-B       --preemptirqs     both preempt and irqsoff tracing (used with -b)
-c CLOCK --clock=CLOCK     select clock
                           0 = CLOCK_MONOTONIC (default)
                           1 = CLOCK_REALTIME
-C       --context         context switch tracing (used with -b)
-d DIST  --distance=DIST   distance of thread intervals in us default=500
-D       --duration=t      specify a length for the test run
                           default is in seconds, but 'm', 'h', or 'd' maybe ad
```

Fig1. Cyclictest installed

   - Check the latency of system by running at threads at priority = 90, using nanosleep timer, locking memory for 10000 loops:
     time sudo cyclictest -p 90 -n –m -t4 -l 10000

```
pi@yz2455zw579:~/lab4 $ time sudo cyclictest -p 90 -n -m -t4 -l 10000
sudo: unable to resolve host yz2455zw579: Name or service not known
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.00 0.00 0.00 2/196 1380

T: 0 ( 1376) P:90 I:1000 C:  10000 Min:      6 Act:   15 Avg:   14 Max:     103
T: 1 ( 1377) P:90 I:1500 C:   6655 Min:      6 Act:   18 Avg:   15 Max:     125
T: 2 ( 1378) P:90 I:2000 C:   4980 Min:      6 Act:   13 Avg:   14 Max:      66
T: 3 ( 1379) P:90 I:2500 C:   3976 Min:      6 Act:   13 Avg:   14 Max:      31

real    0m10.276s
user    0m0.115s
sys     0m0.714s
```

Fig2. 10000 loops unloaded

   - Increase number of loops by 300000, running:
     time sudo cyclictest -p 90 -n –m -h 500 -t4 -l 300000

```
sudo: unable to resolve host yz2455zw579: Name or service not known
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.23 0.20 0.08 1/195 1397

T: 0 ( 1390) P:90 I:1000 C: 300000 Min:      6 Act:   12 Avg:   14 Max:     205
T: 1 ( 1391) P:90 I:1000 C: 299979 Min:      6 Act:   17 Avg:   17 Max:     214
T: 2 ( 1392) P:90 I:1000 C: 299956 Min:      6 Act:   18 Avg:   18 Max:     330
T: 3 ( 1393) P:90 I:1000 C: 299933 Min:      6 Act:   26 Avg:   15 Max:     216
```

Fig3. 300000 loops unloaded

- Run 25 copies of sort_v1 in the background and repeat the command:
  time sudo cyclictest -p 90 -n –m -h 500 -t4 -l 300000

```
time sudo cyclictest -p 90 -n -m -h 500 -t4 -l 300000
sudo: unable to resolve host yz2455zw579: Name or service not known
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.25 0.18 0.09 1/203 2435

T: 0 ( 2431) P:90 I:1000 C:  29373 Min:      5 Act:   18 Avg:   15 Max:     122
T: 1 ( 2432) P:90 I:1000 C:  29373 Min:      6 Act:   15 Avg:   15 Max:     229
T: 2 ( 2433) P:90 I:1000 C:  29373 Min:      6 Act:   16 Avg:   13 Max:      89
T: 3 ( 2434) P:90 I:1000 C:  29373 Min:      6 Act:   15 Avg:   13 Max:      56
```

Fig4. 300000 loops loaded

- Use htop to make sure core is busy, CPU usage is 101%.

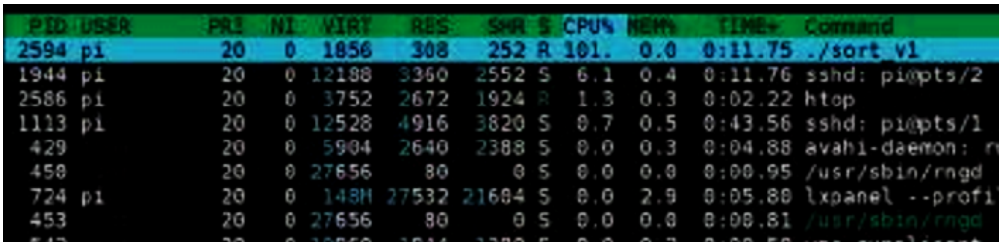

Fig5. Htop showing core busy

- Test performance of sort.c by running
  sudo perf_4.9 stat ./sort_v1

```
size of data = 131072, size of *data = 4
elapsed time = 13.94481

 Performance counter stats for './sort_v1':

      13955.080254      task-clock (msec)         #    1.000 CPUs utilized
                42      context-switches          #    0.003 K/sec
                 0      cpu-migrations            #    0.000 K/sec
                81      page-faults               #    0.006 K/sec
       16675021364      cycles                    #    1.195 GHz
       10923284054      instructions              #    0.66  insn per cycle
         992923565      branches                  #   71.151 M/sec
         329549428      branch-misses             #   33.19% of all branches

      13.959287577 seconds time elapsed

      13.947628000 seconds user
       0.010005000 seconds sys
```

Fig6. Performance of sort_v1.c without quick sort

- Modify sort_v1,c to include a sort on a large array prior to entering loop, then test performance again.

```
pi@yz2455zw579:~/lab4 $ sudo perf_4.18 stat ./sort_v1
sudo: unable to resolve host yz2455zw579: Name or service not known
size of data = 131072, size of *data = 4
elapsed time = 11.94922

 Performance counter stats for './sort_v1':

     11975.057067      task-clock (msec)         #    0.999 CPUs utilized
              113      context-switches          #    0.009 K/sec
                0      cpu-migrations            #    0.000 K/sec
              114      page-faults               #    0.010 K/sec
      14369292566      cycles                    #    1.200 GHz
      10929275715      instructions              #    0.76  insn per cycle
        993799399      branches                  #   82.989 M/sec
          1521795      branch-misses             #    0.15% of all branches

     11.985611774 seconds time elapsed

     11.966799000 seconds user
      0.010014000 seconds sys
```

Fig7. Performance of sort_v1.c with quick sort

- Use blink.py and blink_v6.c to generate stable square wave and test their frequency upper bound when system is unload and loaded.

- Frequency upper bound for blink.py with unloaded system is 821Hz.

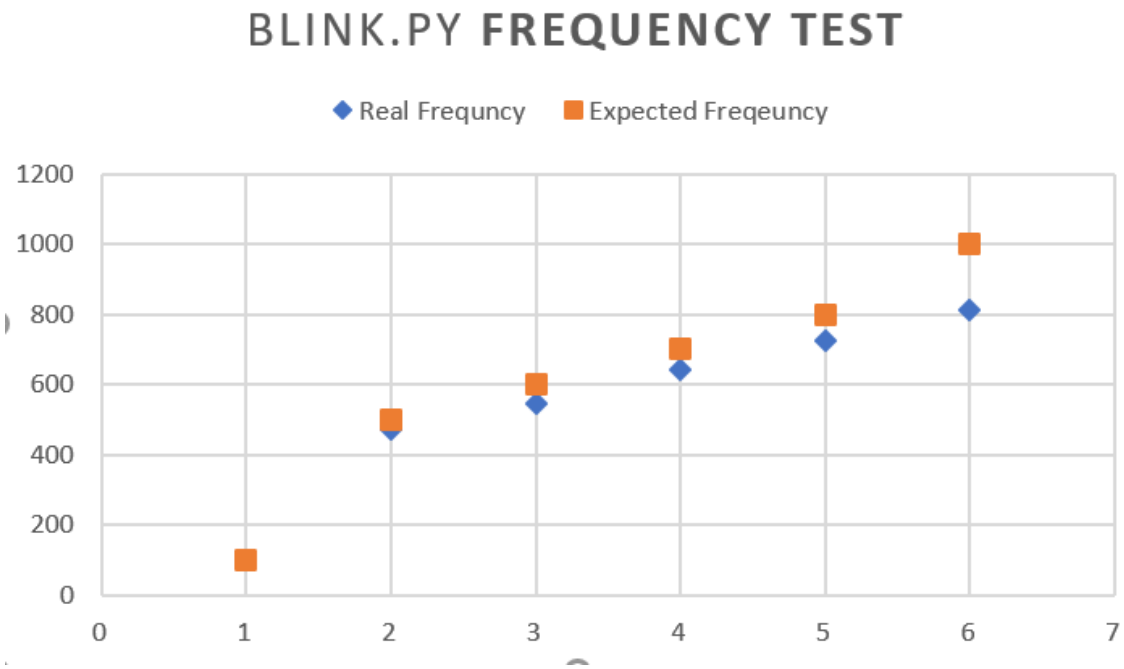

Fig8. Blink.py frequency test unloaded

- Frequency upper bound for blink.c with unloaded system is 50KHz.

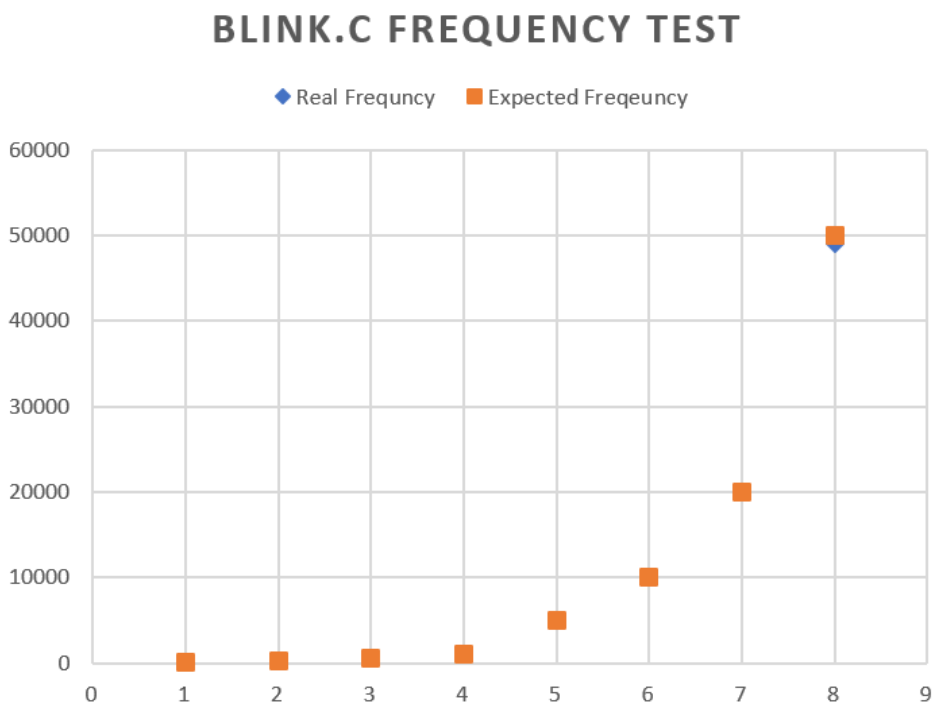

Fig9. Blink.c frequency test unloaded

- Frequency upper bound for blink.c with loaded system is 33KHz

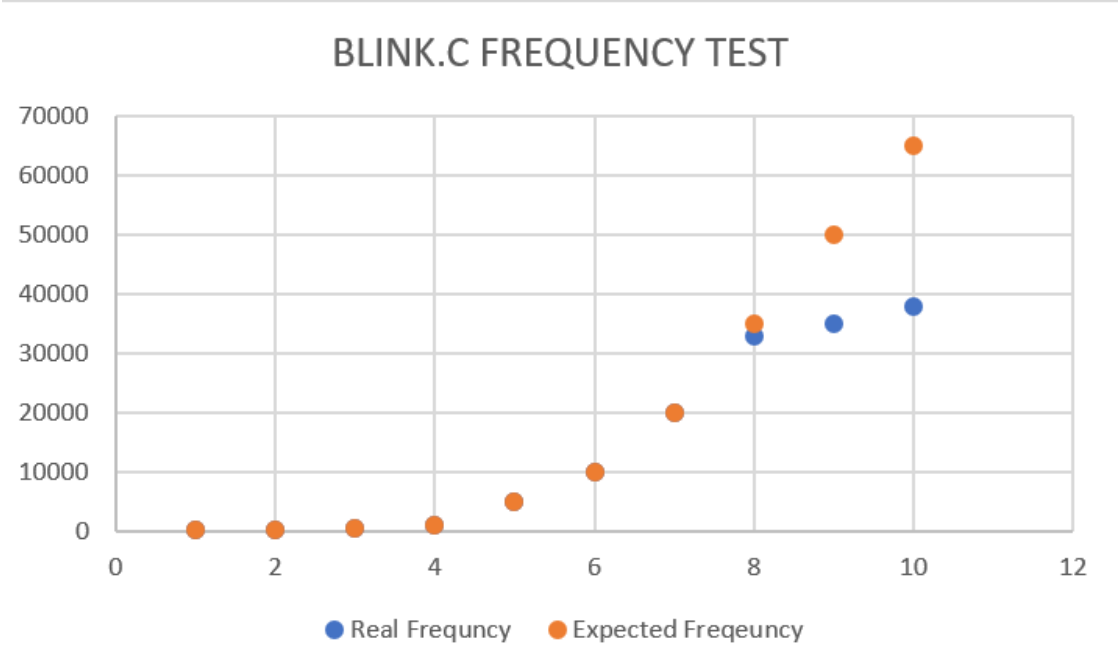

Fig10. Blink.c frequency test loaded

2. Build PreemptRT Kernel

- Free space on sd card:

  Running df -h

```
pi@yz2455zw579:~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        15G  2.8G   11G  21% /
devtmpfs        459M     0  459M   0% /dev
tmpfs           464M     0  464M   0% /dev/shm
tmpfs           464M   13M  451M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           464M     0  464M   0% /sys/fs/cgroup
/dev/mmcblk0p1  253M   41M  212M  17% /boot
tmpfs            93M     0   93M   0% /run/user/1000
```

Fig11. 11G free space on sd card

Running dpkg --get-selections

```
x11-common                                      install
x11-utils                                       install
x11-xkb-utils                                   install
x11-xserver-utils                               install
xarchiver                                       install
xauth                                           install
xcompmgr                                        install
xdg-user-dirs                                   install
xdg-utils                                       install
xfconf                                          install
xinit                                           install
xinput                                          install
xkb-data                                        install
xml-core                                        install
xsel                                            install
xserver-common                                  install
xserver-xorg                                    install
xserver-xorg-core                               install
xserver-xorg-input-all                          install
xserver-xorg-input-libinput                     install
xserver-xorg-video-fbdev                        install
xserver-xorg-video-fbturbo                      install
xxd                                             install
xz-utils                                        install
```

Fig12. All installed applications

Running du -sh *

```
pi@yz2455zw579:~ $ du -sh *
4.0K    Desktop
4.0K    Documents
4.0K    Downloads
24K     Lab1
312K    Lab2
38M     MagPi
4.0K    Music
4.0K    Pictures
4.0K    Public
4.0K    Templates
4.0K    Videos
0       bashrc
4.0K    bashrc.save
64K     lab3
4.0K    video_fifo
```

Fig13. Storage used in a particular directory

- Load appropriate source code for base Linux kernel:
  git clone --single-branch --branch 'rpi-4.14.y-rt'

  After download, check free space

```
pi@yz2455zw579:~ $ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        15G  5.5G  8.3G  40% /
devtmpfs        459M     0  459M   0% /dev
tmpfs           464M     0  464M   0% /dev/shm
tmpfs           464M   13M  451M   3% /run
tmpfs           5.0M  4.0K  5.0M   1% /run/lock
tmpfs           464M     0  464M   0% /sys/fs/cgroup
/dev/mmcblk0p1  253M   41M  212M  17% /boot
tmpfs            93M     0   93M   0% /run/user/1000
```

Fig 14. System free space

Check the version of code we have downloaded:

```
pi@yz2455zw579:~/linux $ head Makefile
# SPDX-License-Identifier: GPL-2.0
VERSION = 4
PATCHLEVEL = 14
SUBLEVEL = 91
EXTRAVERSION =
NAME = Petit Gorille

# *DOCUMENTATION*
# To see a list of typical targets execute "make help"
# More info can be located in ./README
```

Fig15. Kernel Version

- Download PreemptRT Patch and merge with kernel source, by running:
  user@host ~/linux$ wget
  https://www.kernel.org/pub/linux/kernel/projects/rt/4.14/patch4.14.87-rt49.patch.gz

- Then apply the patches to linux source:
  zcat patch-4.14.87-rt49.patch.gz | patch -p1 > /home/pi/patch.log 2>&1

- Configure the kernel by installing libncurses5-dev and giving kernel image name kernel7. Then change settings in menuconfig.

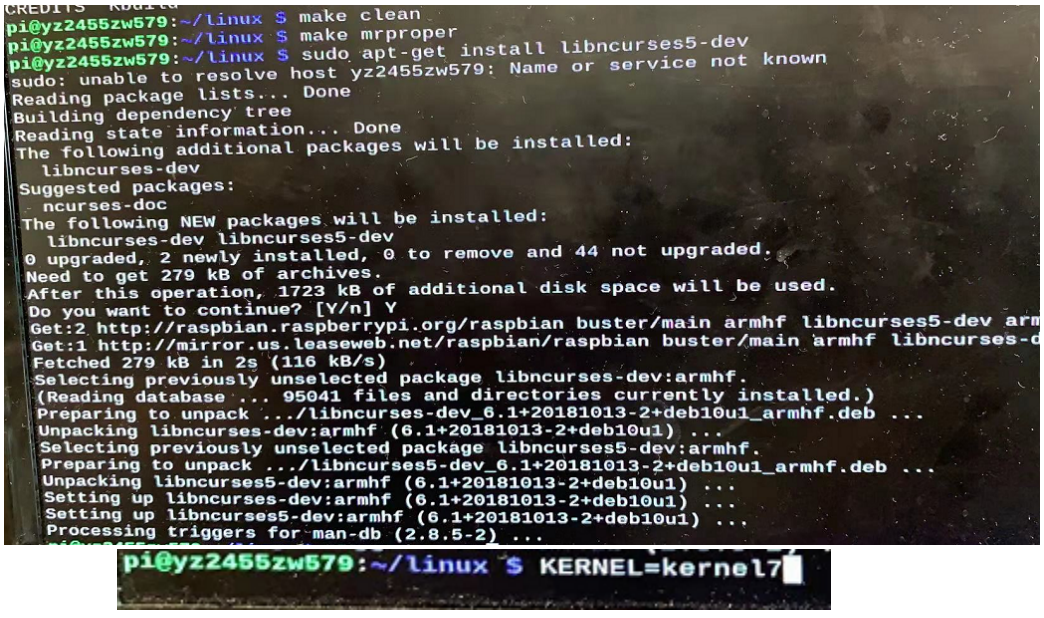

Fig16. Installing libncurses5-dev

- Compile the patched, configured kernel
  Make.log shows no error when compiling.

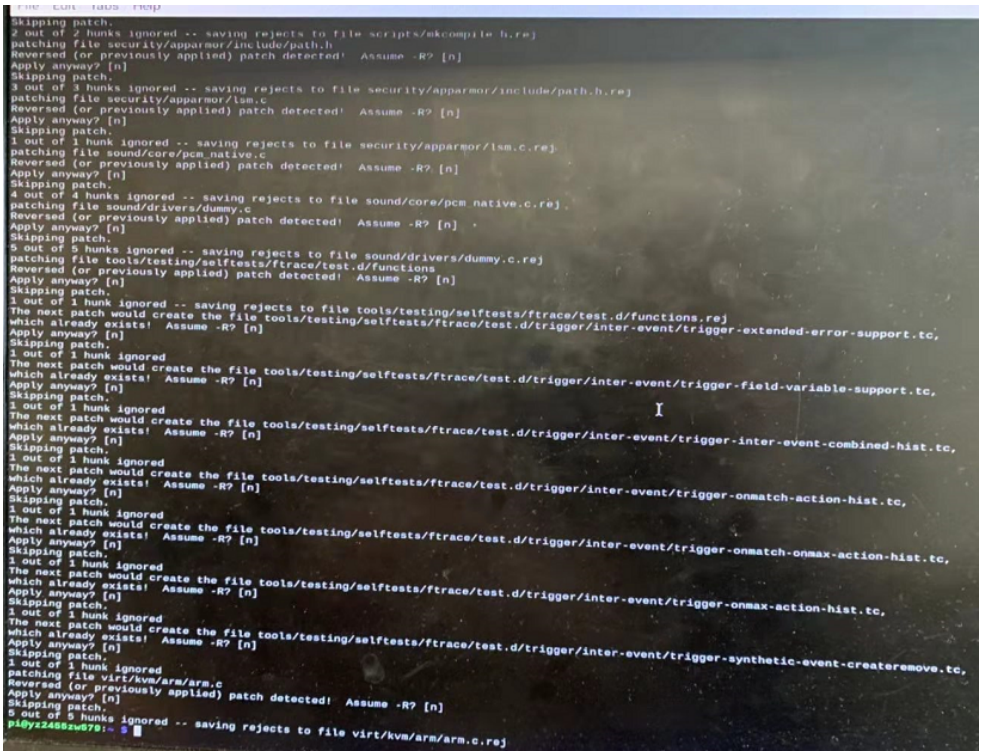

Fig17. Log information in another console

- Check free space then run

  time make –j4 zImage modules dtbs > /home/pi/make.log 2>&1

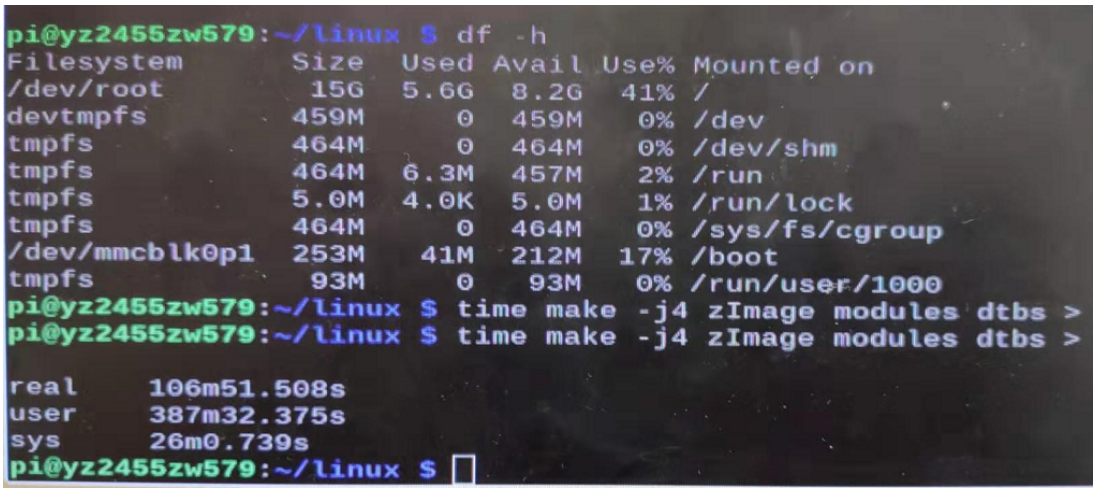

Fig18. Start 4 tasks which use 4 processors on RPi

Week 2

1. System Setup

- Place elements of new kernel in correct locations by running:

```
sudo cp arch/arm/boot/dts/*.dtb /boot/
sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/
sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/
sudo scripts/mkknlimg arch/arm/boot/zImage /boot/$KERNEL.img
```

- Add the following to the end of the file /boot/cmdline.txt
  dwc_otg.fig_enable=0 dwc_otg.fig_fsm_enable=0 dwc_otg.nak_holdoff=0
  And reboot the system.

2. Check performance of PreemptRT System

- First, try cyclic test with system unloaded:

```
sudo cyclictest -p 90 –m –n –t4 -l 10000
```

```
pi@yz2455-zw579:~/lab4/lab4 $ sudo cyclictest -p 90 -m -n -t4 -l 10000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.46 3.89 6.57 1/224 1372

T: 0 ( 1369) P:90 I:1000 C:  10000 Min:      7 Act:   15 Avg:   16 Max:      61
T: 1 ( 1370) P:90 I:1500 C:   6650 Min:      6 Act:   14 Avg:   17 Max:      65
T: 2 ( 1371) P:90 I:2000 C:   4975 Min:      7 Act:   14 Avg:   17 Max:      42
T: 3 ( 1372) P:90 I:2500 C:   3970 Min:      7 Act:   20 Avg:   17 Max:      70
```

Fig19. Cyclictest 10000 loops unloaded

- Increase the number of loops

```
sudo cyclictest -p 90 -h 500 –m –n –t4 -l 300000
```

```
pi@yz2455-zw579:~/lab4/lab4 $ sudo cyclictest -p 90 -m -n -t4 -l 300000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 0.50 1.35 4.46 1/227 1444

T: 0 ( 1394) P:90 I:1000 C: 300000 Min:      6 Act:   20 Avg:   16 Max:     165
T: 1 ( 1395) P:90 I:1500 C: 199976 Min:      6 Act:   43 Avg:   16 Max:     112
T: 2 ( 1397) P:90 I:2000 C: 149964 Min:      6 Act:   23 Avg:   17 Max:      88
T: 3 ( 1398) P:90 I:2500 C: 119961 Min:      6 Act:   18 Avg:   17 Max:      97
```

Fig20. Cyclictest 300000 loops unloaded

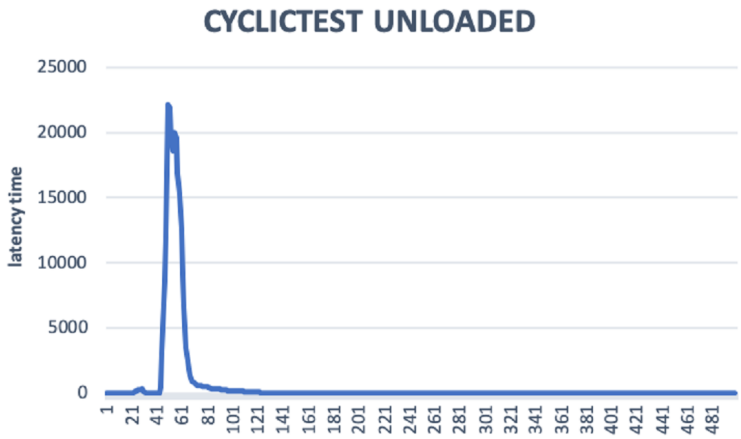

Fig20. Cyclictest 300000 loops histogram

- Try Cyclic test with system loaded:

```
pi@yz2455-zw579:~/lab4/lab4 $ sudo cyclictest -p 90 -m -n -t4 -l 10000
# /dev/cpu_dma_latency set to 0us
policy: fifo: loadavg: 5.64 1.77 3.62 26/250 1527

T: 0 ( 1522) P:90 I:1000 C:  10000 Min:      8 Act:   14 Avg:   13 Max:      52
T: 1 ( 1523) P:90 I:1500 C:   6648 Min:      9 Act:   24 Avg:   14 Max:      40
T: 2 ( 1524) P:90 I:2000 C:   4975 Min:     10 Act:   14 Avg:   15 Max:      66
T: 3 ( 1525) P:90 I:2500 C:   3971 Min:     10 Act:   17 Avg:   17 Max:      46
```

Fig21. Cyclictest 10000 loops loaded

- Increase the number of loops with system loaded:

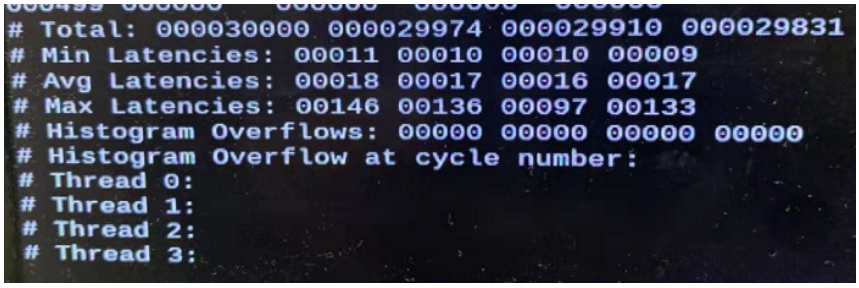

Fig22. Cyclictest 300000 loops loaded

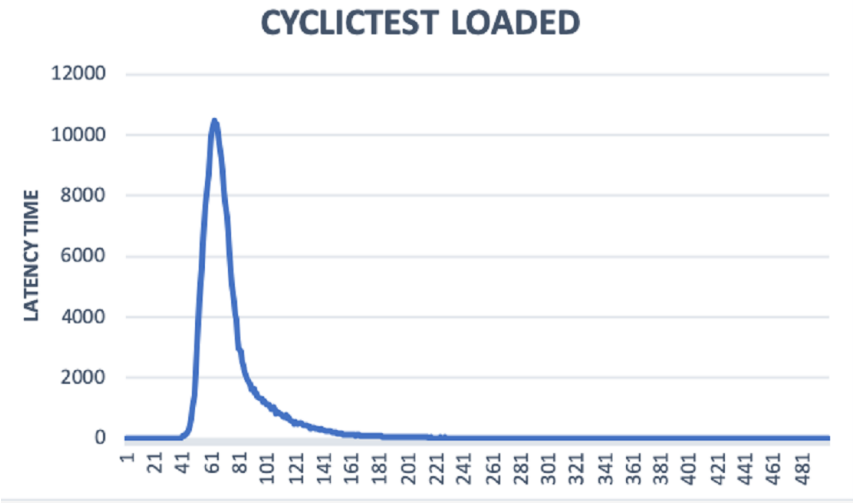

Fig23. Cyclictest 300000 loops histogram

- Re-run the perf tests on the two sort_v1 C codes

```
Performance counter stats for './sort_v1':

     13865.534591      task-clock (msec)         #    0.982 CPUs utilized
           10098      context-switches          #    0.728 K/sec
              14      cpu-migrations            #    0.001 K/sec
              81      page-faults               #    0.006 K/sec
     16597691660      cycles                    #    1.197 GHz
     10910082910      instructions              #    0.66  insn per cycle
       991193392      branches                  #   71.486 M/sec
       330464734      branch-misses             #   33.34% of all branches

    14.113805171 seconds time elapsed

    13.873744000 seconds user
     0.009846000 seconds sys
```

Fig24. Perf test on sort_v1.c with quick sort

```
Performance counter stats for './sort_v1':

     11965.940143      task-clock (msec)         #    0.984 CPUs utilized
             8278      context-switches          #    0.692 K/sec
               14      cpu-migrations            #    0.001 K/sec
              113      page-faults               #    0.009 K/sec
      14325441657      cycles                    #    1.197 GHz
      10930818941      instructions              #    0.76  insn per cycle
        994038798      branches                  #   83.072 M/sec
          1312428      branch-misses             #    0.13% of all branches

     12.166393446 seconds time elapsed

     11.976249000 seconds user
      0.000000000 seconds sys
```

Fig25. Perf Test on sort_v1.c without quick sort

Note: **Comparing with perf test on original kernel, context-switches increase largely on PreemptRT Kernel.**

3. Square wave test

- Generate a stable square wave by modifying test_rt_skel_v7.c, checking it with the oscilloscope. Below are frequency test with system unloaded.

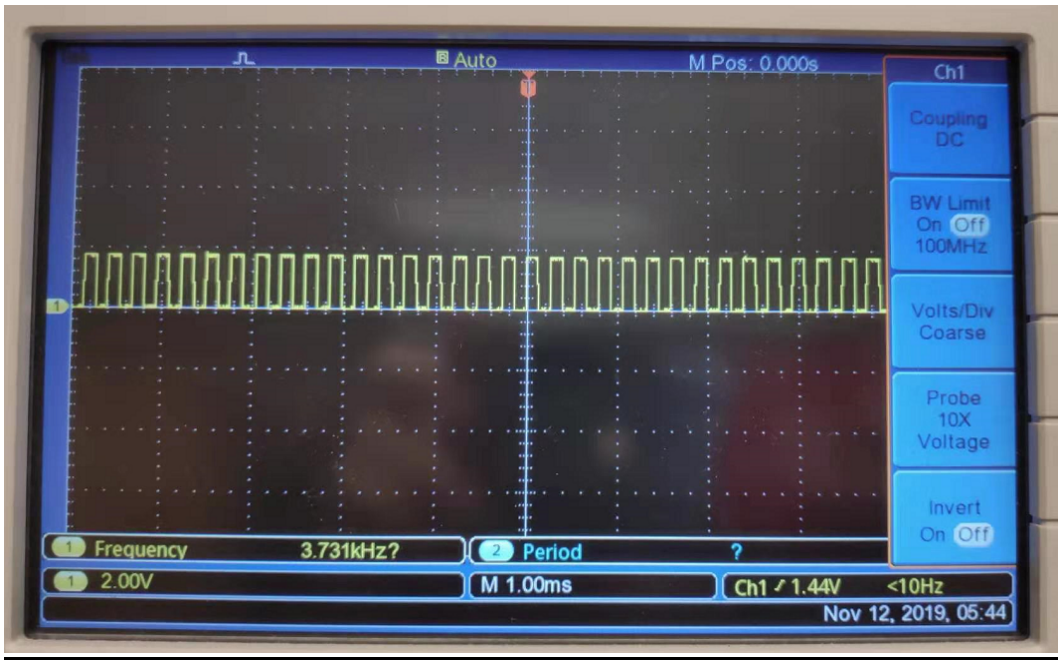

Fig26. interval=20000, frequency=3.73KHz

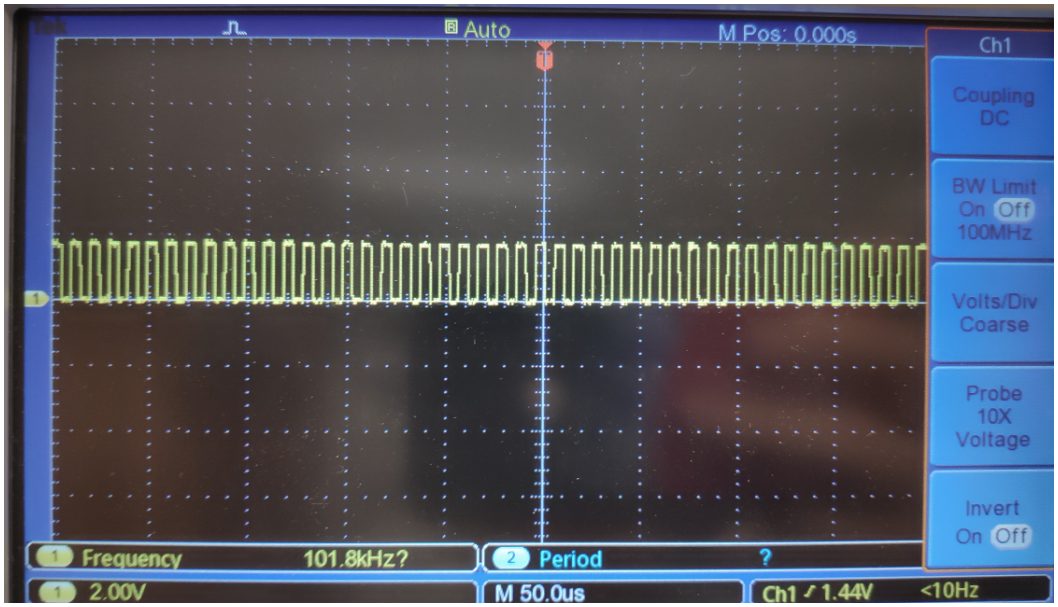

Fig27. interval=730, frequency=100Khz

- When system is loaded, frequency drops with same settings.

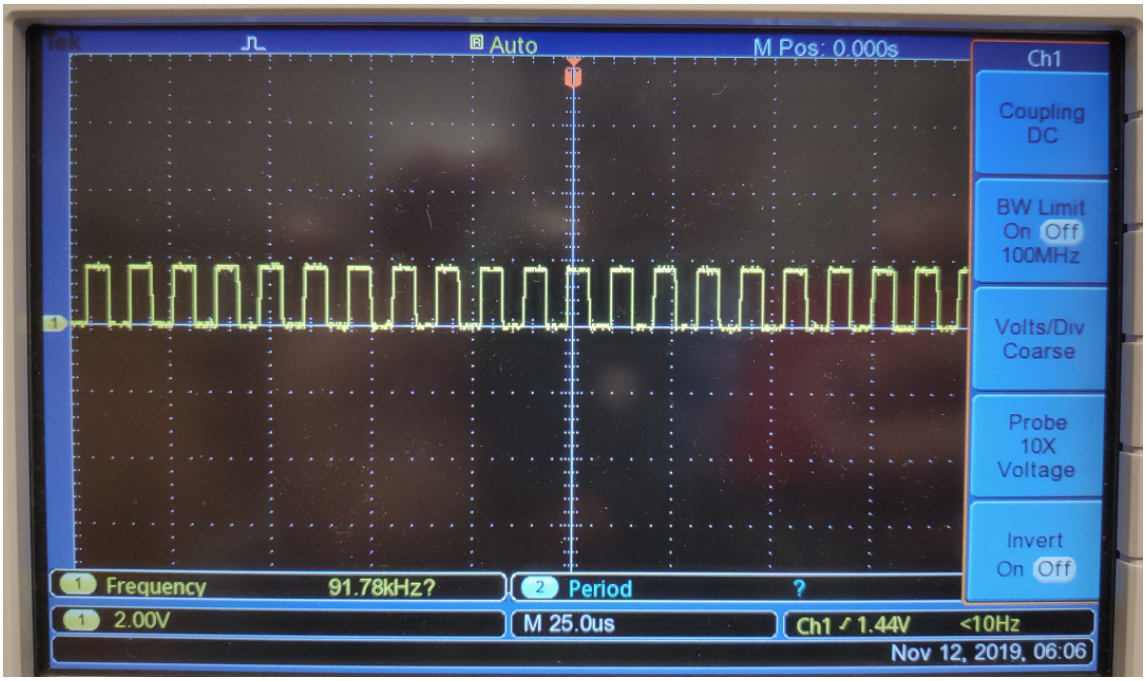

Fig28. interval=730, frequency= 91.78KHz, system loaded

- Run test_rt_v7 without any system loading and run ‘ps –alef | grep test_rt_v7’ and note the pid of the running test_rt_v7

```
pi@yz2455-zw579:~/lab4/lab4 $ ps -alef|grep test_rt_v7
4 T root      1368   707  0  80   0 -  1342 -      17:44 pts/0    00:00:00 sudo
./test_rt_v7 730
4 T root      1381  1368  3  10   - -   759 -      17:44 pts/0    00:00:26 ./tes
t_rt_v7 730
4 S root      1865   707  0  80   0 -  1342 -      17:55 pts/0    00:00:00 sudo
./test_rt_v7
4 R root      1870  1865 98  10   - -   759 -      17:55 pts/0    00:00:26 ./tes
t_rt_v7
0 S pi        1875   707  0  80   0 -   717 pipe_w 17:56 pts/0    00:00:00 grep
--color=auto test_rt_v7
```

Fig29. Processes of test_rt_v7

- Run ‘chrt -p pid’:

```
pi@yz2455-zw579:~/lab4/lab4 $ htop
pi@yz2455-zw579:~/lab4/lab4 $ chrt -p 1870
pid 1870's current scheduling policy: SCHED_FIFO
pid 1870's current scheduling priority: 49
```

Fig30. Process priority

- Run test_rt_v7 730 with system loading and run ‘ps –alef | grep sort_v1’ and note the pid of the running loading test

  Run ‘chrt -p pid’ as well

```
bash: some: command not found
pi@yz2455-zw579:~ $ ps -alef | grep sort_v1
0 R pi        1264  1103 89  80   0 -   464 -      18:27
t_v1
0 S pi        1266  1256  0  80   0 -   684 pipe_w 18:27
--color=auto sort_v1
pi@yz2455-zw579:~ $ chrt -p 1264
pid 1264's current scheduling policy: SCHED_OTHER
pid 1264's current scheduling priority: 0
pi@yz2455-zw579:~ $
```

```
pi@yz2455-zw579:~ $ chrt -p 1197
chrt: failed to get pid 1197's policy: No such process
pi@yz2455-zw579:~ $ ps -alef | grep sort_v1
0 R pi        1201  1103 99  80   0 -   464 -      18:21
0 S pi        1203  1168  0  80   0 -   684 pipe_w 18:21
pi@yz2455-zw579:~ $ chrt -p 1201
pid 1201's current scheduling policy: SCHED_OTHER
pid 1201's current scheduling priority: 0
```

Fig31. Process and its priority

- Run ‘ps –alef | grep blink’ with system unloaded and note the pid of the running blink test

```
pi@yz2455-zw579:~/lab4/lab4 $ ps -alef|grep blink
0 S pi        1250  1068  2  80   0 -   759 hrtime 18:25 pts/2    00:00:00 ./bli
nk_v6
0 S pi        1252   997  0  80   0 -   684 pipe_w 18:25 pts/1    00:00:00 grep
--color=auto blink
```

-Run ‘chrt –p pid’ where pid = pid of the blink test

```
pi@yz2455-zw579:~/lab4/lab4 $ chrt -p 1250
pid 1250's current scheduling policy: SCHED_OTHER
pid 1250's current scheduling priority: 0
```

Fig32. Process and its priority

-Run ‘blink_v6’ to generate a stable square wave and add system loading.

Fig33. Frequency dropping when loaded

System loading causes the frequency output dropping.

Conclusions:

Things worked well:

1. Download cyclitest from github and measure performance of current system. Check the latency of the current system for 10000 loops and 300000 loops in both unloaded and loaded conditions.
2. Measure performance of sort.c using perf.
3. Generate a stable square wave with blink.py and display the square wave in oscilloscope, increase the frequency and find the upper bound.
4. Installed wiringPi which will provide access to the GPIO pins in C.
5. Generate a stable square wave with blink_v6.c in unloaded condition and display the square wave in oscilloscope, increase the frequency and find the upper bound.
6. Generate a stable square wave with blink_v6.c in loaded condition and display the square wave in oscilloscope.
7. Free space on SD card to enable subsequent download and compilations.
8. Download linux source files.
9. Download the preemptRT patch and merge with kernel source.
10. Configure the kernel for the upcoming compilation.
11. Compile the patched, configured kernel.
12. Backup the SD card.
13. Use uname -a to confirm the updated kernel version.
14. Place elements of the new kernel in the correct locations.
15. Launch the newly compiled kernel.
16. Measure performance of the new system, similarly to the measurement of old system.
    a. Check the latency of the new system for 10000 loops and 300000 loops in both unloaded and loaded conditions.
    b. Measure performance of sort.c using perf.
    c. Generate a stable square wave with test_rt_v7.c and check it with the oscilloscope.increase the frequency and found that there is no upper bound.
17. Explore features of the preempt-RT kernel.

a. Run test_rt_v7 without any system loading to generate a stable square wave.
b. Run test_rt_v7 730 with system loading(sort_v1 running 20 times) and check the wave. Compared with the unloading system, frequency of the output square wave is kind of fluctuating.
c. Same measurement on blink_v6 with system loading and unloading.

Some issues we met:

In the final square wave test, each time we load the system and then run test_rt_v7 730, the system got stuck.
Solution: control it using the keyboard and monitor connected to our Pi instead of using ssh.

Improvement suggestion:

Provide more study reference on kernel compilation.

Things not clear enough:

When we use cyclictest to measure the latency, we have saved the histogram into a chart in excel, but seems it has not been checked.