Linux Kernel Mainline Real-Time History, Support and Experience Based on Robotic and Automotive Projects

Pavel Pisa
pisa@fel.cvut.cz , ppisa@pikron.com

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering
https://fel.cvut.cz/en

PiKRON s.r.o
https://www.pikron.com/

2025-02-02
FOSDEM 2025

Content of Presentation

Outline

Author's Point of View Base and Disclaimer

- 1988 – practice at Development Wokshop of Czech Academy of Sciences, obtained Unix root account on Philips Microcontroller Development System (PMDS 85)
- 1990 – became responsible for HPLC laboratory instruments firmware for my father's projects
- 1990 – started study at Czech Technical University in Prague and with Petr Porazil (schoolmate) overtaken all electrical HW and SW development
- 1992 – PiKRON s.r.o. foundation, bought own development and continued projects there due to collapse of the state owned Laboratory Instruments Prague
- around 1993 – introduced by schoolmates to set of diskettes (Slackware GNU/Linux, kernel 1.1.18)
- interest to use Linux for control application
  - our own RS-485 based protocol to control instruments by PC UART and own Intel 80510 based card (AA_SIO)
  - SJA1000 controller integrated to AA_SIO as well
  - parallel port GPIO and stepper and DC motors applications

RS-385, uLAN, CAN Bus, LinCAN and AA_SIO ISA Addon Card (1997)

- 1991 RS-485 uLAN
  `https://ulan.sourceforge.net/`
  - MS DOS and Intel 8051
  - 1994 Linux till today, but out of mainline
  - later Windows NT 3.5, WMD Windows 2000 ... Windows 11
  - UART, ISA, PCI, USB, embedded systems NXP LPC, NuttX
- CAN bus experiments on AA_SIO
- 2003 LinCAN driver
  `https://ortcan.sourceforge.net/lincan/`
  - used by more companies and in the wild even in 2020
  - but focus and help to SocketCAN (drivers bittiming)

- 2025 LinCAN based CAN FD stack mainlined into RTEMS
  `https://www.rtems.org/`

Need for Real-Time

- The ISO/IEC 2382 standard defines "Real-Time" as the capability of a system to respond to inputs or events within a specified time frame, known as deadline.
  - Hard-Realtime systems – guarantee deterministic behavior, violation of the deadline → catastrophic consequences (defined by ISO 26262, IEC 61508, SIL, ASIL, etc.)
  - Soft-Realtime systems – can violate deadline occasionally → quality of service degradation
- Hard Real-Time – used in control systems, avionics, automotive, industrial production, robotics, medical, robotic surgery, etc.
- Soft Real-Time – on-line video capture, processing, delivery, audio including on stage audio mixing, etc.

The First Real-Time Linux Workshop – RTLWS (Year 1999)

- Need for data acquisition and IO cards control in real-time
- Initiated by Peter Wurmsdobler (Centre de Transfert des Microtechniques) developing micro-piezo-electric testing rig (MicroCoupleMetre) for torque testing
- Nicholas Mc Guire, Peter Wurmsdobler, Stefan Jakubek
- FSMLab's RTlinux (Victor Yodaiken) – absorbed by WindRiver
- DIAPM Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano
- DIAPM's RTAI $\rightarrow$ RTAI, Xenomai, ADEOS
- KURT: The Kansas University Real-Time Linux

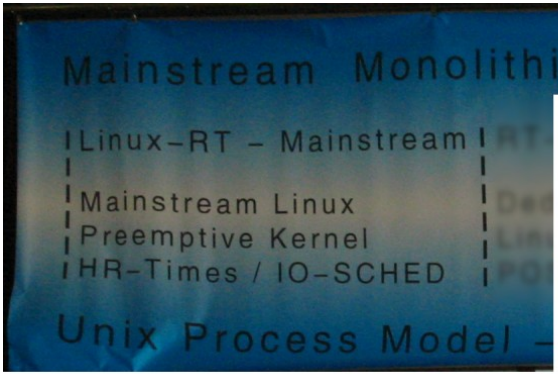
https://www.osadl.org/RTLWS-1999.rtlws-1999.0.html

Outline

8-th Real Time Linux Workshop – Lanzhow University (Year 2006)

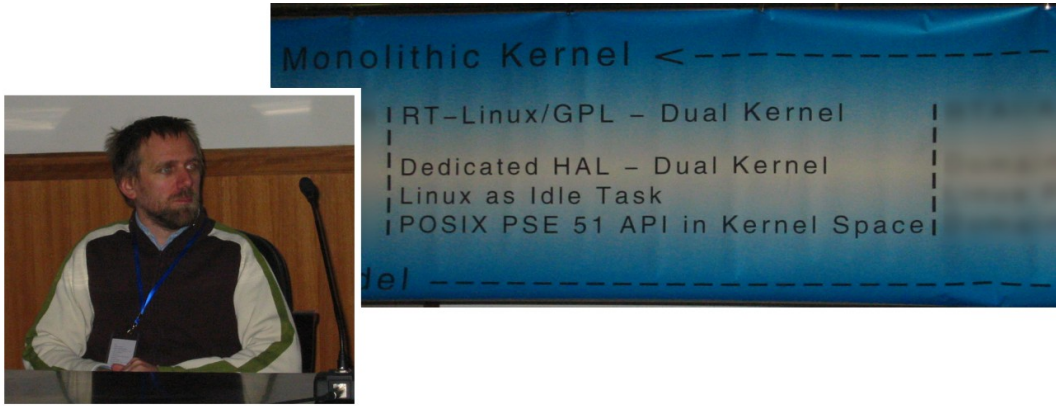
8-th RTLWS – RT Alternatives Debate (Year 2006)

Alternatives – Mainline Linux Kernel Change to RTOS (Year 2006)



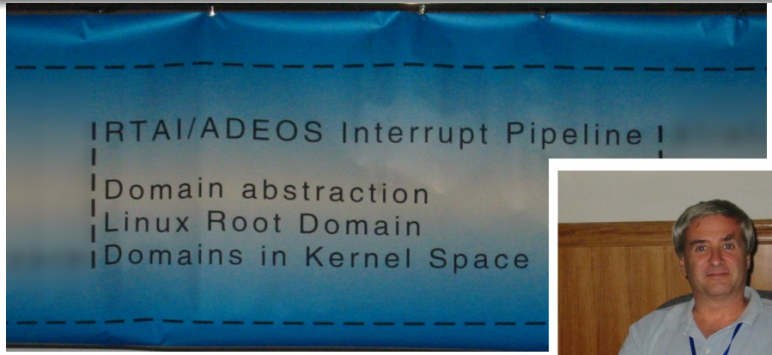
Fully-Preemptive Mainline Kernnel – Thomas Gleixner (Linutronix)

Alternatives – Dual Kernel – RT Linux (Year 2006)



LT Linux – Nicholas Mc Guire (OpenTech)

Alternatives – RTAI/ADEOS (Year 2006)

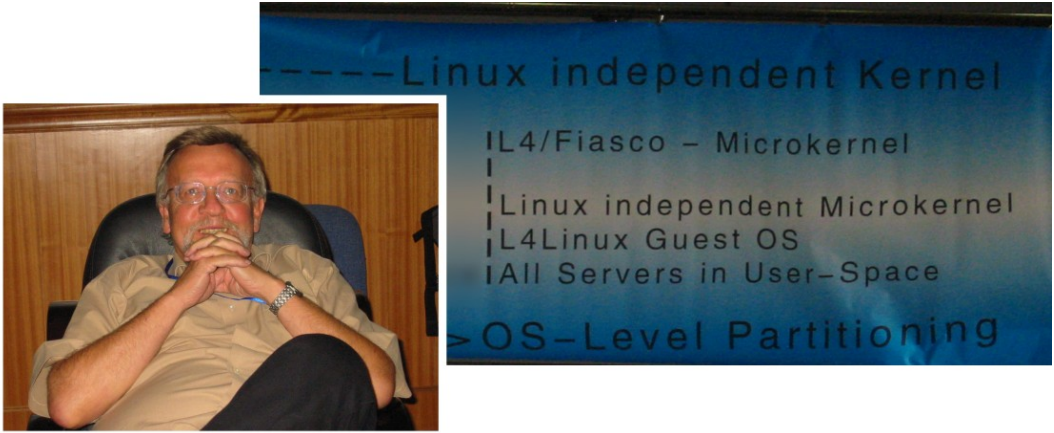


RTAI/ADEOS – professor Roberto Bucher (University of Applied Sciences of Southern Switzerland – SUPSI)

Alternatives – Hypervisor Xtratum (Year 2006)

Xtratum – Ismael Ripoll (Universidad Politecnica de Valencia – UPVLC)

Alternatives – Hypervisor L4 and L4 RTOS Domain (Year 2006)



L4 Fiasco – professor Herman Haertig (Technical University Dresden)

Outline

Fully-Preemptive Linux Kernel

Realtime is not as fast as possible - realtime is as fast as specified – Doug Niehaus, Summer 2001

- More attempts to run RT task parallel to Linux base on same CPU (RT-Linux, RTAI) existed. But around 2001 and 2006 KURT/KUPS project tries to make whole kernel real-time. Work followed by Timesys, Thomas Gleixner, Ingo Molnar and OSADL.org.
- The main idea behind changing Linux kernel to RTOS is to use already present support for multiple cores SMP and provide to system as many virtual CPUs as there are running threads/task.
- Realized by replacement of spin-lock synchronization by RT mutexes. redefinition of spin_lock/spin_unlock, spin_lock_irqsave/spin_unlock_irqrestore to use struct rt_mutex instead of atomic variables based lock

Linux Kernel Development

- 1991-01-05 Linus Torvalds bought IBM PC
- Linus informs about intent to write a kernel for fun
- 1991-08-25 – version 0.01 published on Internet
- 1994 – v1.0 – only single i386 CPU, no kernel preemption
- 1996 – v2.0 – SMP for applications, BKL (Big Kernel Lock) for kernel
- 1999 – v2.2 – spinlocked critical sections, m68k a PowerPC
- 2001 – v2.4 – ISA PnP, USB, PC Cards, PA-RISC, LVM, RAID, ext3, Bluetooth
- 2003 – v2.5.2 – ultra-scalable O(1) SMP and UP scheduler
- 2003 – v2.5.4 – PREEMPT kernel option, preemptible outside critical sections
- 2003 – v2.5.37-mm1 – Read-Copy Update infrastructure, Paul E. McKenney
- 2003 – v2.6 – mainline µClinux, ARM and more, PAE, ALSA, preemption, Native POSIX Thread Library, Futex, latter FUSE, JFS, XFS, ext4, robust mutex, priority inheritance mutex, high resolution timers

The First Mainline Accepted Patches from the RT Project

- CONFIG_PREEMPT_RT separate patches, spinlock → RT-mutex, removal BKL, IRQ → threads, preemptible RCU
- 2004 – v2.6.9 – Sven-Thorsten Dietrich (MontaVista) announces Real Time Kernel
- 2004 – Ingo Molnar and Thomas Gleixner joined on realtime preempt patches
- 2005 – v2.6.11 – Generic Interrupt subsystem
- 2006 – v2.6.16 – RT-Mutex (Thomas Gleixner), Priority Inherintance, PI-Futex, Mutexes, RT-mutex implementation design documentation (Steven Rostedt), Lockdep,
  the first alternative production ready Preempt-RT release
- 2007 – v2.6.21 – Generic timekeeping, High resolution timers, Tickless idle
- 2007 – v2.6.23 – mainline switch to Completely Fair Scheduler (CFS)
- 2008 – v2.6.27 – Tracing
- 2009 – v2.6.32 – Preemptible RCU, Threaded interrupts, Raw Spinlocks
- 2010 – v2.6.37 – RT maintenance mode

Linux Kernel Development

- 2011 – v2.6.39 – "BKL: That's all, folks" in mainline kernel
- 2016 – v4.9 – LF Realtime Linux project, Timer wheel rework
- 2017 – v4.14 – CPU hotplug rework
- 2018 – v4.19 – Tree wide cleanup of locking constructs
- 2019 – v5.4 – FPU, stacktrace, timers support for RT, Introduction of CONFIG_PREEMPT_RT into mainline
- 2020 – v5.10 – BPF support for RT, migration control and high-mem cleanup, seqcount rework, in_interrupt() rework
- 2021 – v5.15 – First batch of printk() related work, RT locking primitives
- 2022 – v6.1 – Network consolidation, Further prink() work
- 2023 – v6.6 – Continue prink() rework, Preparation of serial drivers
- 2024 – v6.11 – Again printk() waits for the final bits

Linux Real-Time Thread Attributes Preparation

```
pthread_attr_t attr;
struct sched_param schparam;

/* Initialize thread attributes by default parameters */
pthread_attr_init(&attr);

/* The scheduling policy is applied to the started thread */
pthread_attr_setinheritsched(&attr, PTHREAD_EXPLICIT_SCHED);

/* Choice of the desired scheduling policy */
pthread_attr_setschedpolicy(&attr, SCHED_FIFO);

/* Specify the thread priority in the given policy range */
schparam.sched_priority = sched_get_priority_max(SCHED_FIFO) - 10;

/* Setup scheduling policy in the thread create attributes */
pthread_attr_setschedparam(&attr, &schparam);
```

Linux Real-Time Thread Start

Lock whole program in the memory

```
mlockall(MCL_FUTURE | MCL_CURRENT);
```

Start RT thread – start_routine()

```
/* Create thread with parameters specified */
pthread_create(thread, &attr, start_routine, arg);

/* Release resources used to build parameters */
pthread_attr_destroy(&attr);
```

List individual threads with ascending priority

```
ps Hxa --sort rtprio -o pid,policy,rtprio,state,tname,time,command
```

Linux Real-Time Sampling Period Implementation

```
sample_period_nsec = 20*1000*1000; /* period in nanoseconds */
clock_gettime(CLOCK_MONOTONIC, &sample_period_time);

do {
  /* Compute time for next period invocation */
  sample_period_time.tv_nsec += sample_period_nsec;
  if (sample_period_time.tv_nsec > 1000*1000*1000) {
    sample_period_time.tv_nsec -= 1000*1000*1000;
    sample_period_time.tv_sec += 1;
  }

  clock_nanosleep(CLOCK_MONOTONIC, TIMER_ABSTIME,
                  &sample_period_time, NULL);

  /* The place to insert code to execute periodically  */
  ...
} while(1);
```

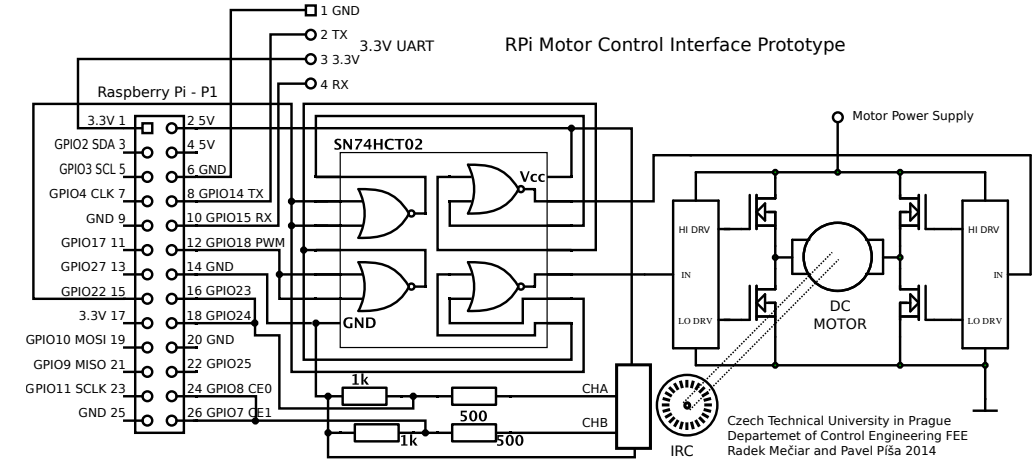
Ensure Transition to the Safe State in Case of User Break or Error

```
/* Stop actuators in the case of error */
void stop_motor(void)
{
  /* Place to put code for technology stop */
}

/* Signal handler and program termination in case of error */
void sig_handler(int sig)
{
  stop_motor();
  exit(1);
}

...
  struct sigaction sigact;
  memset(&sigact, 0, sizeof(sigact));
  sigact.sa_handler = sig_handler;
  sigaction(SIGINT, &sigact, NULL);
  sigaction(SIGTERM, &sigact, NULL);
```

PWM and GPIO Only Based DC Motor PRi Interfacing



As simple as possible

- Four NOR gates (SN74HCT02) and H-bridge (L6203)

Example of PSD (PID) Controller for DC Motor Control

```
/* Control error, difference between requested and measured state, computation */
err = (pos_req - actual_pos);

/* Accumulator of control error */
ctrl_i_sum += err * ctrl_i;

/* Control action computation */
action = ctrl_p * err + /* proportional component */
         ctrl_i_sum +   /* "integration" component */
              /* differential/"derivative" component */
         ctrl_d * (err - ctrl_err_last);

/* Remember the current error for next differential component computation */
ctrl_err_last = err;

/* Scale adjustment for computation in the fixed point arithmetic */
rpi_bidirpwm_set(action >> 8);
```

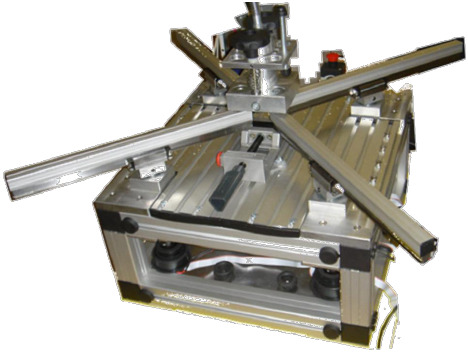
https://github.com/ppisa/rpi-rt-control/tree/master/appl/rpi_simple_dc_servo

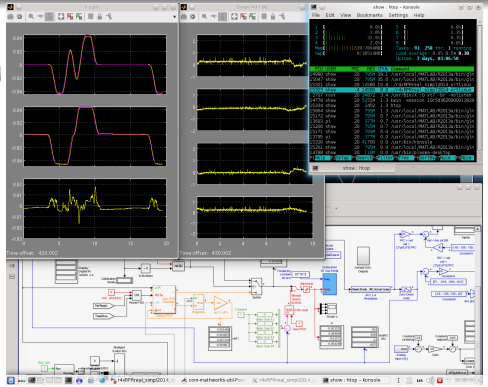
Or Use Some Rapping Control Applications Development System

- pysimCoder (started by Roberto Bucher)
  `https://github.com/robertobucher/pysimCoder`
  `https://github.com/robertobucher/pysimCoder-examples`

- Matlab® Simulink®
  - Simulink Embedded Coder target for Linux
    `http://lintarget.sourceforge.net/`
    `https://github.com/aa4cc/ert_linux`
  - Matlab/Simulink model for Xilinx Zynq and MZ_APO DC and PMSM peripherals
    `https://github.com/aa4cc/zynq-rt-control/`

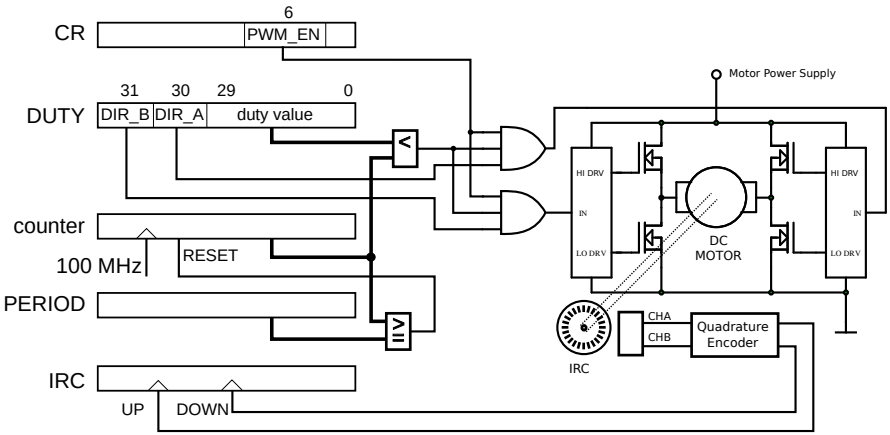
x86 Linux ERT and Parallel Kinematic Robot Control





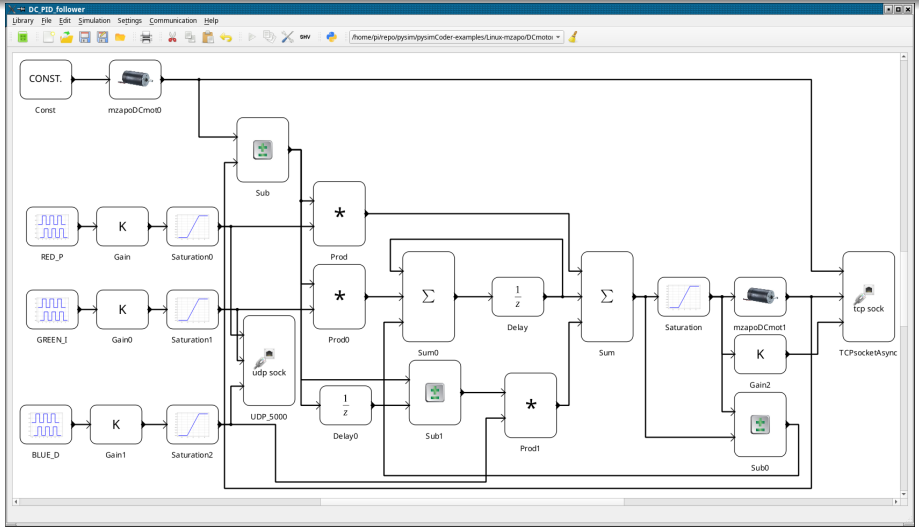
- 4 DC motors, 4 incremental encoders, other I/Os
- Presented at Embedded world 2014
- Sampling period 1 ms but complex computations
- More reliable that previously used Windows target

DC Motor Control for Education, This Time on Zynq FPGA



https://gitlab.fel.cvut.cz/canbus/zynq/zynq-can-sja1000-top/-/tree/master/system/ip/dcsimpledrv_1.0

pysimCoder Servo Control ("Steer by Wire") on Xylinx Zynq MZ_APO Kit



https://github.com/robertobucher/pysimCoder-examples

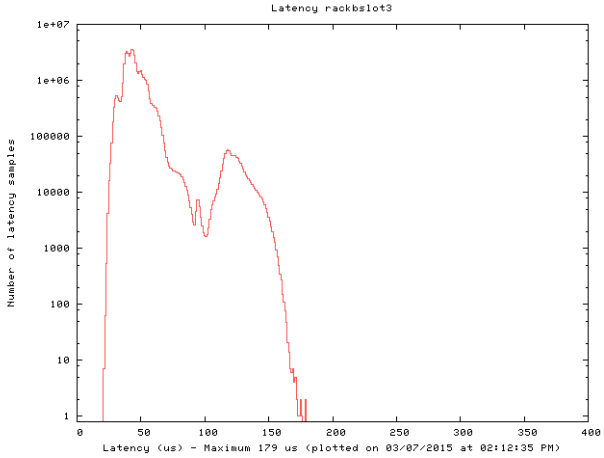
Outline

OSADL QA Farm Real-Time

- Open Source Automation Development Lab – long term testing and Quality Assurance Realtime Farm
  `https://www.osadl.org/OSADL-QA-Farm-Real-time.linux-real-time.0.html`
- Latest available RT-Preempt
  `https://www.kernel.org/pub/linux/kernel/projects/rt/`
- Maximal under about 40 µsec on powerful SMP x86 systems
- But even on less powerfull ARM 32-bit systems usually 200 µsec

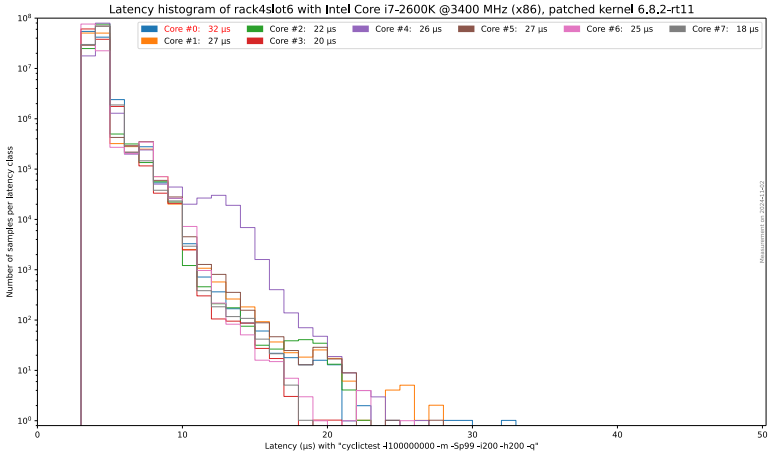
RPi 3.18.7-rt2 Latency Plot (2015)

OSADL.org – OSADL.org QA Farm Realtime – BCM2835 rack-b-slot-3
cyclictest -l50000000 -m -n -a0 -t1 -p99 -i400 -h400 -q

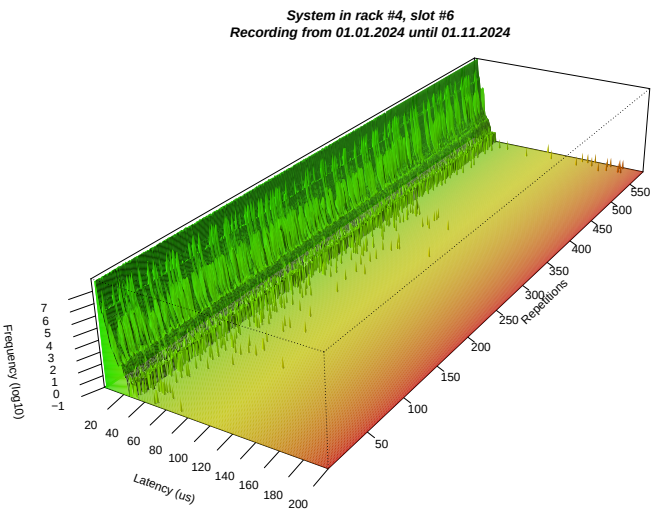


Intel Core i7-2600K @3400 MHz, kernel 6.2.8-rt11 Latency Plot (2024)

OSADL.org – OSADL.org QA Farm Realtime – rack-4-slot-6
cyclictest -l00000000 -m -Sp99 -i200 -h200 -q

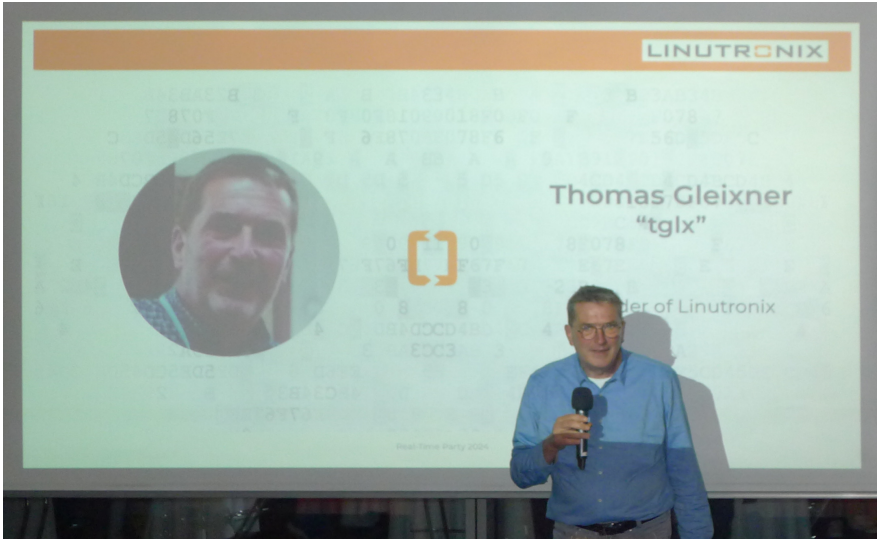


Intel Core i7-2600K @3400 MHz, kernel 6.2.8-rt11 Long Term (2024)



Outline

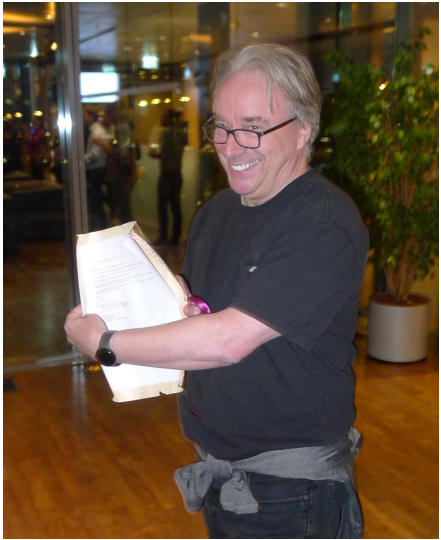
RT Event – Thomas Gleixner (Year 2024)



RT Event – RT Enablement Patch Passed to Linus Torvalds (Year 2024)

RT Event – RT Enablement Patch Passed to Linus Torvalds (Year 2024)

RT Event – The Golden Patch for 6.12 Kernel (Year 2024)



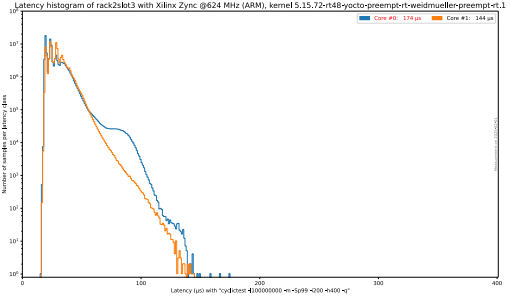
RT Event – Linus Torvalds and Thomas Gleixner (Year 2024)

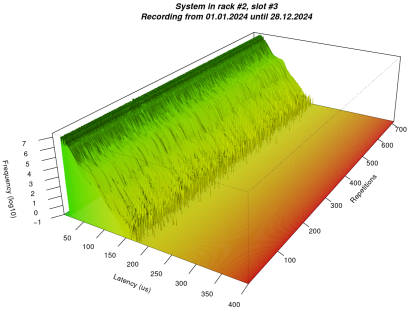
Outline

PREEMPT_RT and OSADL QA Farm on Real-time

ARM Xilinx Zync @624 MHz, 5.15.72-rt48, rack 2, slot 3

```
cyclictest -l100000000 -m -Sp99 -i200 -h400 -q
```





100 million samples per plot, performance governor, duration 5 hours, 33 minutes

`https://www.osadl.org/OSADL-QA-Farm-Real-time.linux-real-time.0.html`

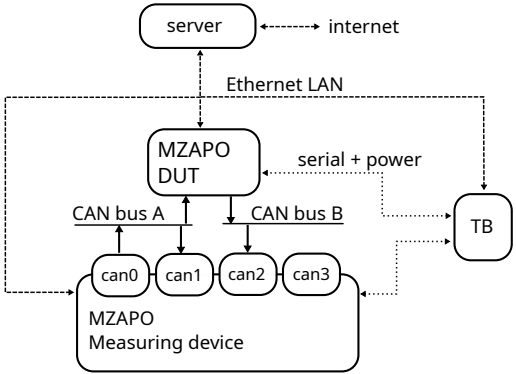
Communication Latency is Critical Often Too

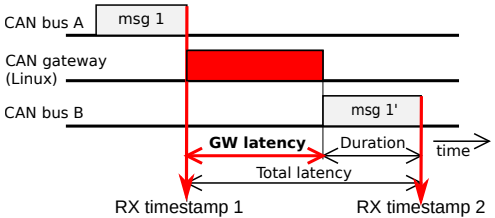
- OSADL.org runs networking latency benchmarks as well
- CTU developed on Volkswagen contract multiple systems to evaluate CAN bus latencies
- CAN drivers on x86 (LinCAN, SocketCAN), MPC5200 (LinCAN, SocketCAN, RTEMS) evaluation
- Linux kernel CAN gateway evaluation for PREEMPT_RT and mainline kernels under different loads and built conditions
- Linux kernel CAN frames processing overhead evaluation when different system calls are used (rtems kernel, read-write, readnb-write, readbusy-write, mmap-mmap, mmap-write, mmapbusy-mmap, mmapbusy-write, readnb-mmap, readbusynoirq-write, mmsg-mmsg)
  for complete report see Performance evaluation of Linux CAN-related system calls by M. Sojka and P. Pisa Czech Technical University in Prague (2014) report

Current CAN Latency Testing Initiative

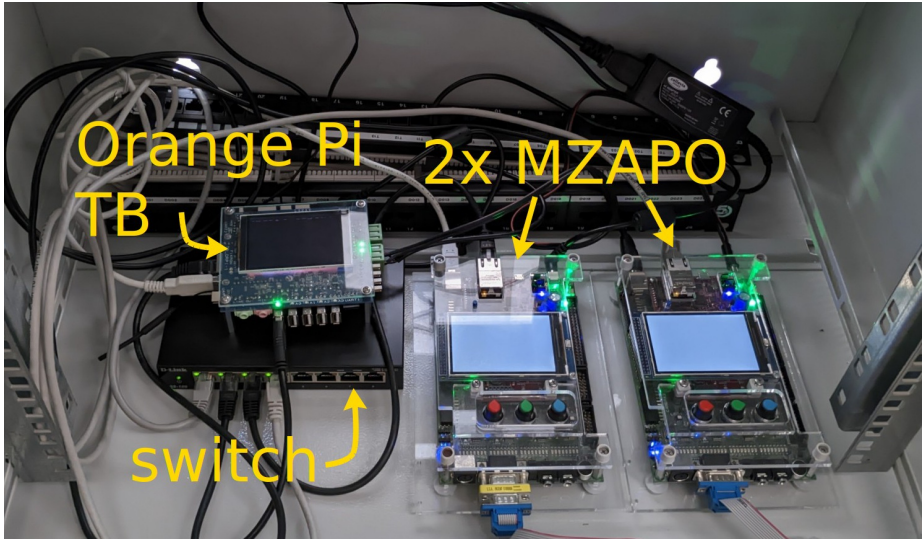
- Use CTU CAN FD IP Core (10 nsec timestamping sychronized over 4 channels on Zynq MZ_APO)
- Prepare system to run daily test on mainlne and RT_PREEMPT development kernels
- Timestamping code implemented by Matej Vasilevski in frame of his thesis
  `https://dspace.cvut.cz/bitstream/handle/10467/101450/F3-DP-2022-Vasilevski-Matej-vasilmat.pdf`
- Work on automation and presentation of results on web in a frame of Pavel Hronek's thesis
  `https://dspace.cvut.cz/bitstream/handle/10467/109308/F3-BP-2023-Hronek-Pavel-CAN-Latester-Automation.pdf`
- All sources, drivers and documentation for CTU OTREES CAN related projects and testing on Linux and RTEMS is available at
  `https://canbus.pages.fel.cvut.cz/`

CAN Gateway Latency Definition





CAN Gateway Latency Tester Cabinet



CAN Latency Tester – Daily Results

Overview Inspect Compare

Time series of maximum latency measured in selected configurations

CAN Latency Tester – Inspection

RT, Under Load, RT priority set, Flood, CAN FD

Overview Inspect Compare

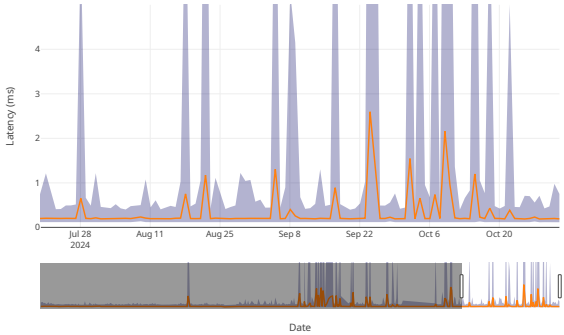
○master ●RT

☑ Under load ☑ RT priority set ☑ Kernel GW ☑ Flood ☑ CAN FD

Gateway latency

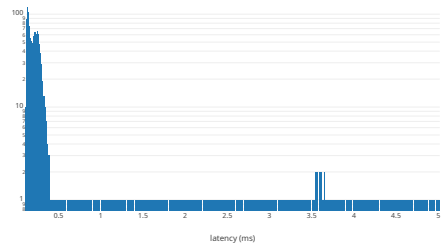
Line plot | Heatmap view | Surface view

Click into graph to show individual histogram below



CAN Latency Tester – Inspection – RT Bad, RT OK

run-241022-045232-hist+6.12.0-rc2-rt4-g7bc6f8add0ae+flood-kern-prio-fd-load.json



Toggle cumulative

Single run statistics

Lost: 0 (0.00 %)

Best: 0.115 ms

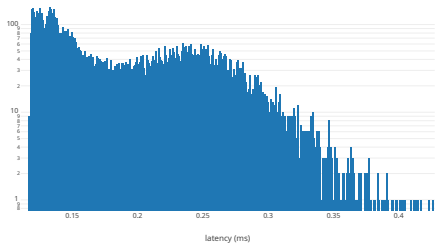
5th percentile: 0.127 ms

Median: 0.212 ms

95th percentile: 1.779 ms

Worst: 5.015 ms

run-241029-045308-hist+6.12.0-rc4-rt6-ga4680e452f4f+flood-kern-prio-fd-load.json



Toggle cumulative

Single run statistics

Lost: 0 (0.00 %)

Best: 0.117 ms

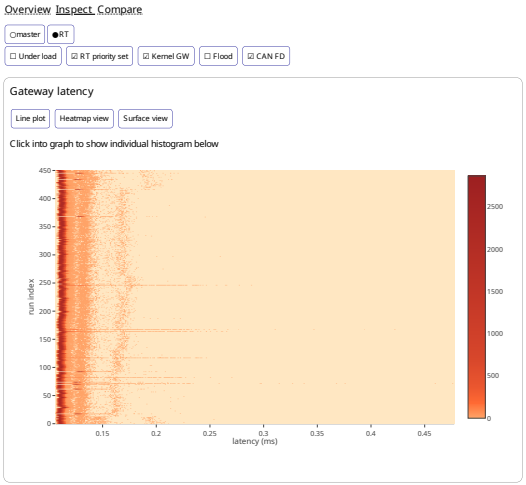
5th percentile: 0.121 ms

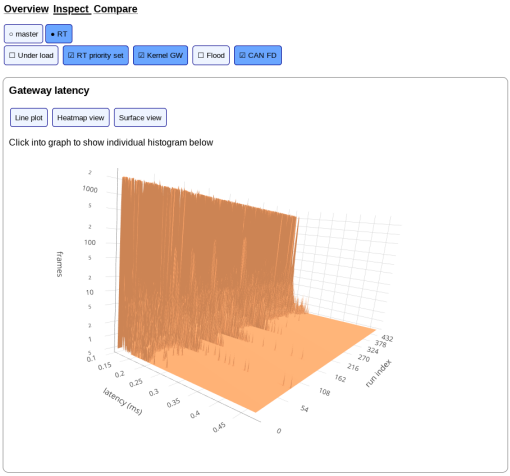
Median: 0.18 ms

95th percentile: 0.297 ms

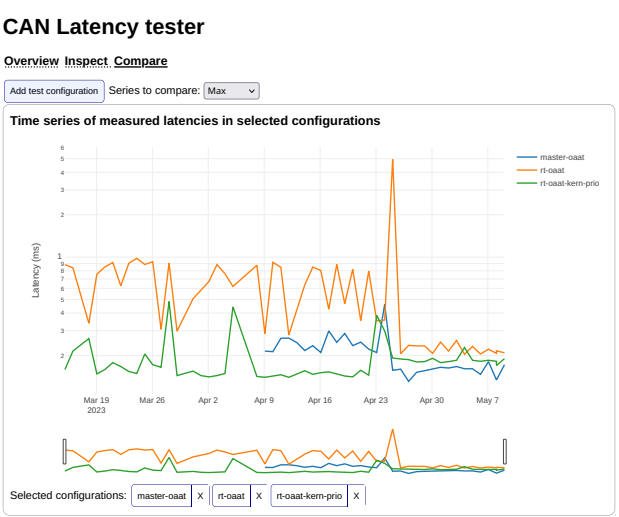
Worst: 0.426 ms

CAN Gateway Latency – Heatmap and Surface





CAN Gateway Latency – Compare



Outline

Previous Presentations

- InstallFest 2015

Is Raspberry Pi Usable for Industrial and Robotic Applications?
http://installfest.cz/if15/slides/pisa_rpi.pdf

- LinuxDays 2015

Linux, RPi and other HW for DC and Brushless/PMSM Motor Control
https://www.linuxdays.cz/2015/video/Pavel_Pisa-Rizeni_stejnosmernych_motoru.pdf

- LinuxDays 2016

Processor Systems, GNU/Linux and Control Applications
https://www.linuxdays.cz/2016/video/Pavel_Pisa-Procesorove_systemy_a_nejen_GNU_Linux_v_ridicich_aplikacich.pdf

- InstallFest 2017

GNU/Linux and FPGA in Real-time Control Applications
https://installfest.cz/if17/slides/so_t2_pisa_realtime.pdf

Related Articles

- Píša, P., Vacek, F.:Open Source Components for the CAN Bus, 5-th RTLWS, 2003
- Píša, P., Lisový, R.:COMEDI and UIO drivers for PCI Multifunction Data Acquisition and Generic I/O Cards and Their QEMU Virtual Hardware Equivalents, 13-th RTLWS, 2011, drivers in mainline UIO, COMEDI
- Píša, P., Smolík, P., Fanda, F., Boháček, M., Štefan, J., Němeček, P.: Process Data Connection Channels in uLan Network for Home Automation and Other Distributed Applications, 13-th RTLWS, 2011, related to uLAN project
- Lenc, M., Píša, P., and Bucher, R.: pysimCoder – Open-Source Rapid Control Prototyping for GNU/Linux and NuttX, In: 2023 24th International Conference on Process Control (PC), Strbske Pleso, Slovakia, 2023, pp. 102-107 DOI: 10.1109/PC58330.2023.10217596
- Píša, P.; Hronek, P.; Vasilevski, M.; Novák, J.: Continuous CAN Bus Subsystem Latency Evaluation and Stress Testing on GNU/Linux-Based Systems, In: embedded world Conference 2024. Haar: WEKA FACHMEDIEN GmbH, 2024. p. 77-82. ISBN 978-3-645-50199-6
- Lenc, M.; Píša, P.: Scheduling of CAN frame transmission when multiple FIFOs with assigned priorities are used in RTOS drivers, international CAN Conference, CAN in Automation, 2024

Information Sources

- It is really time to celebrate!
  25th anniversary of RTLWS
  20th anniversary of Preempt RT
  Thomas Gleixner and Heinz Egger
  `https://www.linutronix.de`
- `https://wiki.linuxfoundation.org/realtime/start`
- `https://www.osadl.org`
- `https://canbus.pages.fel.cvut.cz/`
- `https://gitlab.fel.cvut.cz/otrees/org/-/wikis/home` – theses, talks, references
- `https://social.kernel.org/ppisa`
- Photo © Pavel Pisa