KAJIAN DAN IMPLEMENTASI REAL TIME OPERATING SYSTEM PADA SINGLE BOARD COMPUTER BERBASIS ARM

Wiedjaja A; Handi M; Jonathan L; Benyamin Christian; Luis Kristofel

Computer Engineering Department, Faculty of Engineering, BINUS University
Jln. K.H. Syahdan No. 9, Palmerah, Jakarta Barat 11480
steff@binus.edu[1], lily_fas@hotmail.com[2], jonathanlks@yahoo.com[3]

ABSTRACT

Operating System is an important software in computer system. For personal and office use the operating system is sufficient. However, to critical mission applications such as nuclear power plants and braking system on the car (auto braking system) which need a high level of reliability, it requires operating system which operates in real time. The study aims to assess the implementation of the Linux-based operating system on a Single Board Computer (SBC) ARM-based, namely Pandaboard ES with the Dual-core ARM Cortex-A9, TI OMAP 4460 type. Research was conducted by the method of implementation of the General Purpose OS Ubuntu 12:04 OMAP4-armhf-RTOS and Linux 3.4.0-rt17 + on PandaBoard ES. Then research compared the latency value of each OS on no-load and with full-load condition. The results obtained show the maximum latency value of RTOS on full load condition is at 45 uS, much smaller than the maximum value of GPOS at full-load at 17.712 uS. The lower value of latency demontrates that the RTOS has ability to run the process in a certain period of time much better than the GPOS.

Keywords: *operating system, GPOS, RTOS, latency*

ABSTRAK

Sistem Operasi merupakan peranti lunak penting dalam sebuah sistem komputer. Untuk aplikasi keperluan pribadi dan kantor sistem operasi tersebut sudah mencukupi, namun untuk aplikasi tugas vital seperti pembangkit tenaga nuklir dan sistem pengeraman pada mobil (auto braking system) yang butuh tingkat keandalan tinggi diperlukan sistem operasi yang bersifat real time. Penelitian bertujuan untuk mengkaji penerapan sistem operasi berbasis Linux pada Single Board Computer (SBC) Berbasis ARM, yaitu Pandaboard ES dengan prosesor Dual-core ARM Cortex-A9 berjenis TI OMAP 4460. Penelitian dilakukan dengan metode implementasi General Purpose OS Ubuntu 12.04-armhf-omap4 dan Linux RTOS 3.4.0-rt17+ pada PandaBoard ES, kemudian penelitian membandingkan nilai latency dari tiap OS tersebut pada kondisi tanpa bedan dan dengan beban penuh. Hasil yang didapat nilai latency maksimum dari RTOS dalam kondisi beban penuh sebesar 45 uS, jauh lebih kecil dari nilai maksimum GPOS pada beban penuh sebesar 17.712 uS. Nilai latency yang jauh lebih rendah menunjukkan RTOS memiliki kemampuan untuk menjalankan proses dalam tenggang waktu tertentu jauh lebih baik dibanding GPOS.

Kata kunci: *sistem operasi, GPOS, RTOS, latency*

PENDAHULUAN

Teknologi Informatika makin luas penggunaannya dalam membantu peningkatan taraf hidup manusia. Pekerjaan yang bersifat repetitif yang membosankan, pekerjaan yang berbahaya, pekerjaan yang membutuhkan akurasi tinggi cenderung menggunakan teknologi informatika dan robotika. Aplikasi yang membutuhkan tingkat keandalan tinggi, seperti sistem kendali pembangkit tenaga listrik, sistem kontrol pada mobil, membutuhkan perangkat keras dan lunak yang mempunyai tingkat keandalan yang tinggi. Perangkat lunak yang andal tentunya perlu didukung oleh sistem operasi yang mampu menyediakan akses ke perangkat keras dengan cepat dan efisien. Pada sistem kontrol di mobil perangkat komputer yang dipergunakan harus mempunyai tingkat deterministik yang tinggi, sehingga proses tidak terlambat dijalankan. Misalnya, perintah untuk melakukan pengereman, jika perintah ini terlambat dieksekusi, mobil akan gagal melakukan proses pengereman yang dapat berakibat tabrakan. Sistem operasi seperti ini disebut dengan *Real-time Operating System*. *Real-time Operating System* merupakan tiap proses komputasi harus mampu mengerjakan proses secara benar dan dalam jangka waktu yang telah ditentukan (Leroux, 2012).

Dewasa ini seiring dengan meningkatnya kecepatan dari prosesor, Real-time OS tetap perlu dipergunakan karena proses *blocking* dapat terjadi dari jaringan yang lambat, atau sedang menunggu proses I/O lainnya, sehingga proses tidak dapat selesai dalam kurun waktu yang ditentukan. Pemanfaatan prosesor berbasis ARM dipergunakan mengingat kebutuhan daya yang lebih rendah dan disipasi panas rendah dibanding prosesor berbasis Intel, adapun prosesor ARM menjadi prosesor utama dalam hampir seluruh perangkat *mobile* seperti telepon seluler dan komputer *tablet*.

Penelitian ini melakukan implementasi dan kajian sistem operasi *real-time* pada sistem komputer berbasis Dual-core ARM Cortex-A9 TI OMAP 4460 dari Texas Intrument. Peneliti melakukan kajian mengenai tahapan instalasi, konfigurasi, dan cara membuat aplikasi mengggunakan sistem *real-time*. Proses pembuatan aplikasi menggunakan bahasa pemrograman C/C++. Prosesor berbasis ARM dipilih karena memiliki keunggulan dalam hal performa dan konsumsi daya yang kecil. Adapun Linux dipilih karena sistem operasi *open source* dan tidak berbayar namun memiliki banyak dukungan dari komunitas maupun perusahaan. Banyaknya dukungan menjadikan Linux menjadi sebuah sistem operasi yang andal dan mempunyai dukungan yang luas terhadap berbagai jenis perangkat keras.

Sistem operasi *real time* pada umumnya mempunyai biaya lisensi yang cukup mahal, sehingga sulit bagi peneliti individu atau pemula untuk melakukan penelitian di bidang ini. Penggunaan sistem operasi *real time* berbasis Linux diharapkan dapat mengatasi hal tersebut, sehingga dosen dan mahasiswa dapat melakukan penelitian menggunakan sistem operasi *real time* dengan biaya terjangkau. Hasil dari penelitian berupa kajian dan petunjuk instalasi, konfigurasi, dan cara membuat aplikasi dengan bahasa pemrograman C/C++ dan Qt dalam sistem operasi *real-time* Qnx. Penelitian ini dilakukan selama 1 tahun dan diharapkan dapat menjadi materi pengayaan dalam mata kuliah *Embedded Linux System Development*.

Real-time Operating System

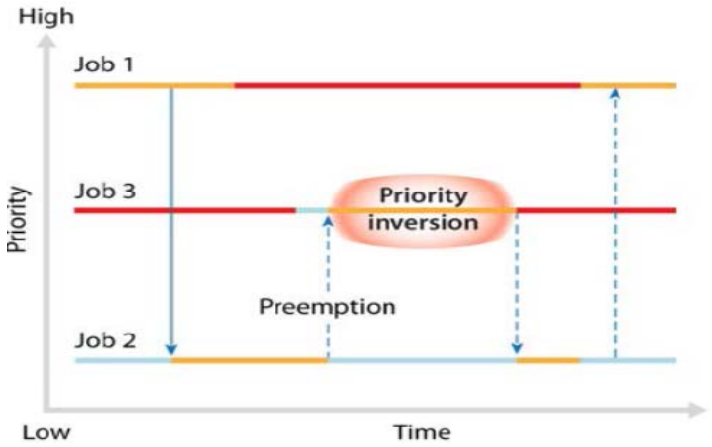
Dewasa ini dengan adanya prosesor yang memiliki kecepatan komputasi tinggi, sering memicu pertanyaan apakah Real-time OS masih diperlukan dalam sebuah sistem embedded. Real-time OS (RTOS) dirancang untuk bekerja sangat cepat dan dapat memberikan respons yang dapat diprediksi meskipun berjalan di prosesor yang memiliki komputasi rendah. Sebagai contoh, prosesor yang dipergunakan di bidang automotif merupakan 32 bit prosesor dengan *clock rate* 600MHz. Meskipun jauh lebih rendah dibanding prosesor yang terdapat pada laptop atau komputer pada umumnya, secara keseluruhan sistem komputer ini mampu memberikan respons waktu yang lebih

cepat dan terprediksi dibanding PC yang menggunakan *General Purpose Operating System* (GPOS) seperti Microsoft Windows dan Linux. Penggunaan prosesor dengan *clock rate* rendah memiliki keunggulan dari sisi harga lebih ekonomis dan konsumsi daya yang lebih rendah (Leroux, 2012).

Keunggulan RTOS dibanding dengan GPOS adalah pada sistem komputer yang dipergunakan untuk mengontrol *air bag* pada mobil. Sistem didesain agar melontarkan *air bag* dalam kurun waktu <1 detik sejak terjadi benturan keras. Sistem komputer untuk *air bag* yang menggunakan RTOS selalu memberikan jaminan bahwa *air bag* akan keluar dalam waktu <1 detik. Sedangkan *General Purpose* OS tidak dapat memberikan jaminan terhadap hal ini. (Koolwal, 2008)

Real-time OS berbeda dengan sistem operasi umum (*General Purpose* OS-GPOS), yaitu pada bagian *task scheduler*. GPOS menggunakan metode *fairness* dalam melakokasi proses ke dalam CPU (Koolwal, 2008). Teknik seperti ini berdampak baik pada aplikasi *Desktop* dan *Server*, tetapi tidak memberi jaminan pada proses yang membutuhkan prioritas tinggi atau *time-critical thread* akan dikerjakan dengan baik. Sebagai contoh, teknik *fairness scheduler*, GPOS akan menunda *thread* dengan prioritas tinggi atau secara dinamis mengatur *thread priority* agar tercapai *fairness scheduling*. Hal ini dapat mengakibatkan *thread* yang mempunyai prioritas tinggi tidak dapat melakukan proses dalam waktu yang telah ditentukan. (Vijay, 2006)

Pada RTOS, setiap *thread* akan dikerjakan sesuai dengan urutan prioritas, dan tiap *thread* akan dikerjakan selama jangka waktu tertentu yang telah ditentukan. Hal ini mencegah suatu proses menggunakan resource CPU terlalu banyak dibanding yang lain. Proses dengan urutan prioritas tinggi dapat dikerjakan sampai selesai tanpa dapat diinterupsi proses lain, atau dapat diinterupsi proses yang memiliki prioritas lebih tinggi (*priority-based preemptive scheduling*) (Leroux, 2012).



Gambar 1 Ilustrasi Priority-based Preemptive Scheduling
(Sumber: Leroux, 2012)

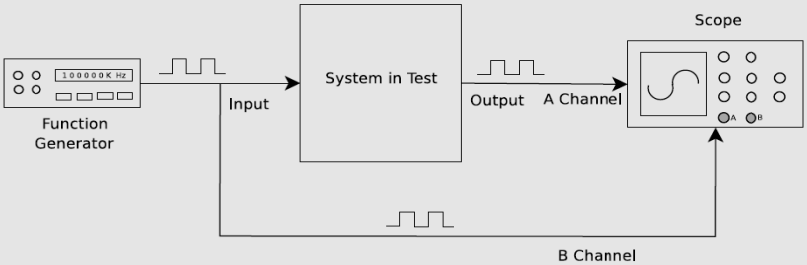
Pada Gambar 1 Job 1 menunggu Job 2 selesai, Job 3 menyela (*preempts*) Job 2 karena Job 3 mempunyai prioritas lebih tinggi dari Job 2. Hal ini membuat Job 1 menunggu lebih lama.

Real-time OS dikategorikan menjadi 2 golongan: *hard real-time* dan *soft real-time*. RTOS *hard real-time* mampu memberikan jaminan 100% respons sistem terhadap suatu proses. Jika terjadi satu kesalahan, hal tersebut akan berakibat fatal bagi sistem keseluruhan. Aplikasi RTOS *hard real-time* adalah pada sistem kontrol ABS (*Automatic Braking System*), *missile system*, *aircraft control*. Ketiga aplikasi tersebut pada umumnya menyangkut sistem keselamatan. RTOS kategori *soft real-time*

menjamin sistem mampu memberikan respons terprediksi dalam rata-rata periode waktu tertentu, misalnya pada aplikasi *audio/video broadcasting*, sistem perdagangan bursa saham, dan bagian tampilan telepon seluler. Untuk aplikasi pengontrolan robot industri disarankan menggunakan RTOS jenis *hard real-time* (Koolwal, 2008).

Performa lebih baik untuk RTOS dapat dilakukan dengan mengimplementasikan *real-time multiprocessor kernel* dalam bentuk perangkat keras atau dikenal dengan *Real-Time Unit* (RTU). Studi yang dilakukan membuktikan peningkatan performa 2.6x lebih baik dari implementasi kernel secara software (Samuelsson, et al, 2003).

Saat ini ada beberapa jenis RTOS dengan sistem lisensi dari berbayar sampai *open source* dan fitur beragam. Pada umumnya untuk mengukur performa dari suatu RTOS digunakan parameter sebagai berikut. *Latency*, adalah waktu jeda dari sistem saat menerima *input* dan melakukan proses; makin kecil latency, makin baik respons dari sistem. *Jitter*, merupakan perbedaan *latency* dari 2 pengukuran; *jitter* yang rendah menandakan makin tinggi kestabilan sebuah system. *Worst case response Time*, merupakan waktu respons terburuk dari sebuah sistem. Untuk melakukan uji tiga parameter tesebut dipergunakan metode seperti gambar berikut (Aroca & Caurin, 2009).



Gambar 2 Metode Pengujian 3 Parameter Performa RTOS

Sistem diuji kecepatannya untuk dapat menghasilkan sinyal *output* berdasarkan perubahan pada sinyal *input*. Uji coba yang dilakukan pada *platform* yang sama yaitu Pentium II 400 MHz dengan memori 256 MB memperoleh hasil sebagai berikut:

Tabel 1 Perbandingan Real Time Operating Systems (RTOS)

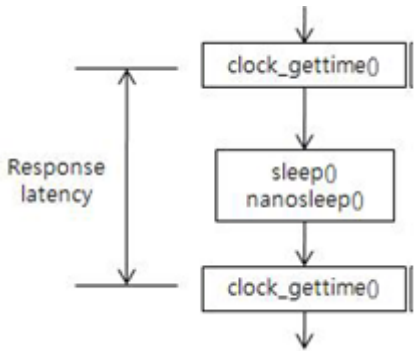
	Win XP	Win CE	Neutrino	µC/OS-II	Linux	RTAI	VxWorks
A	**200µs**	**20µs**	**20µs**	**1.92µs**	**13.89µs**	**5µs**	**3.85µs**
B	**848µs**	**99µs**	**35.2µs**	**3.2µs**	**98µs**	**11.4µs**	**13.4µs**
C	**700µs**	**88,8µs**	**32µs**	**2.32µs**	**77.6µs**	**7.01µs**	**10.4µs**

(A: *Worst case response Time;* B: *Latency;* C: *Latency Jitter)*

Perlu diperhatikan pada tabel, µC/OS-II memiliki waktu paling singkat karena tidak menggunakan *Memory Management Unit* (MMU) yang akan berakibat *crash*/*hang* ketika terjadi *overload*. Sedangkan sistem operasi lainnya tidak mengalaminya karena menggunakan MMU namun menjadi lebih lambat. Linux sebagai OS bersifat *open-source* dapat di-*patch* hingga menjadi sebuah

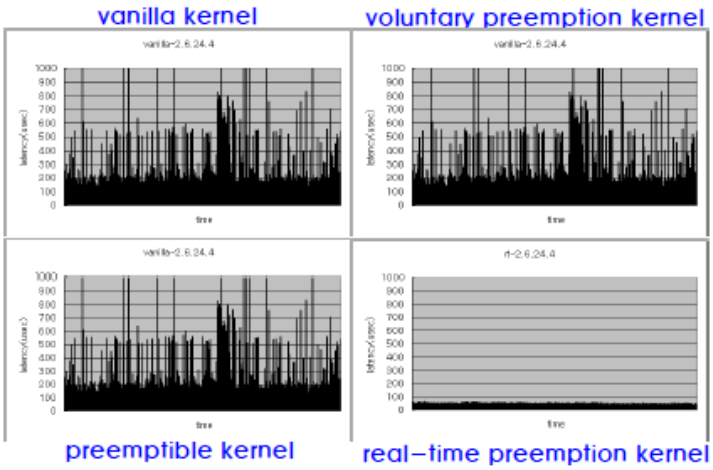
RTOS yang menggunakan MMU. Kelebihan MMU pada Linux yang sudah di-patch dengan *kernel real time* adalah kemampuan *real time* yang cukup baik tanpa perlu mengeluarkan biaya tambahan dan relatif lebih mudah dalam implementasi karena cukup banyak penggunaannya (Aroca & Caurin, 2009).

Metode lain untuk melakukan pengukuran terhadap performa suatu RTOS dapat menggunakan pendekatan secara *software*, yaitu dengan menggunakan program *cyclictest*. Prinsip dasarnya adalah dengan mengukur waktu sebenarnya dari perintah *delay* yang terdapat pada Linux API (POSIX) seperti *sleep*() atau *nanosleep*(). Untuk mengukur waktu dipergunakan *high resolution timer* atau POSIX *timer*. Sistem yang menghasilkan pengukuran *delay* paling kecil, yaitu memberikan nilai yang sama dengan parameter *sleep*() dan *nanosleep*() merupakan sistem dengan *latency* paling kecil karena dianggap mempunyai jeda waktu proses paling pendek. Sistem ini memiliki respons paling baik. Prinsip kerja pengukuran dapat dilihat pada Gambar 3 berikut ini. (Jung, et al, 2010)



Gambar 3 Prinsip Kerja Pengaturan

Dari hasil uji coba pada prosesor VIA EPIA (Nehemiah) 1 GHz, RAM: 256 MB, menggunakan Linux kernel versi 2.6.24.4 selama 10 jam, diperoleh hasil sebagai berikut:



Gambar 4 Hasil Uji Coba

Vanilla kernel adalah Linux kernel orisinal dan *real-time preemption* kernel adalah Linux kernel dengan PREEMPT_RT *patch*, terlihat *jitter* dari Linux *real time* memberikan performa terbaik dibanding yang lain (Jung, et al, 2010). Linux kernel 2.6 mempunyai kelebihan untuk mendukung RTOS, yaitu terdapat *preemptive option* (CONFIG_PREEMPT), O(1) *scheduler*, dan *optional virtual memory*. Dengan demikian, untuk penggunaan Linux *real-time* disarankan menggunakan kernel 2.6 atau lebih (Vun, et al, 2008). Hasil uji coba performa Linux Real-time Xenomai pada prosesor Intel XScale 520MHz dengan memory 128 MB memberikan respons sistem sebesar 58 uS dan 76 uS pada kondisi tanpa beban, dan 247 uS pada kondisi beban penuh (Knutsson, 2008).

METODE

Penelitian dilakukan dengan membandingkan *General Purpose Operating System* (GPOS) dan *Real Time Operating System* (RTOS). Penelitian mengukur *latency* dari tiap sistem baik dalam keadaan tanpa beban maupun dengan beban penuh. Kondisi beban diukur dari % *cpu utilization*, dengan keadaan tanpa beban <5% *cpu utilization* sedangkan pada kondisi beban penuh mencapai >90%. GPOS yang digunakan adalah Ubuntu 12.04 versi ARM dengan opsi kompilasi *hard floating* dan *patching* untuk prosesor OMAP4. Untuk RTOS yang digunakan adalah Linux dengan PREEMPT_RT *patch*, dengan kernel *source* diambil dari ti-ubuntu-3.4-1485.7 dan RT Patch versi 3-4-9-rt17, untuk selanjutnya akan disebut Linux RTOS 3.4.0-rt17+. Kedua jenis sistem operasi tersebut dimasukkan ke *SD card* yang berfungsi sebagai *boot device* pada PandaBoard ES.

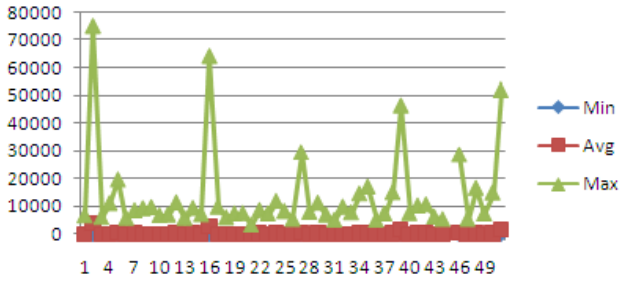
Metodologi pengukuran dilakukan dengan mengambil tingkat respons (*latency*) dari kedua jenis sistem operasi dengan menggunakan program *cyclictest.* Program *cyclictest* dikembangkan oleh tglx dan paling banyak digunakan di komunitas. Prinsip kerja program cyclictest adalah dengan mengukur waktu jeda waktu dari fungsi *sleep*() dan *nanosleep*() yang terdapat pada POSIX Linux API. Proses pengukuran dilakukan menggunakan *high resolution timer*. Jika jeda waktu tersebut kecil, sistem operasi mempunyai tingkat responsif yang tinggi atau *low latency*.

Pada percobaan dilakukan simulasi untuk membuat prosesor sibuk sampai mencapai tingkat CPU utilization >90%. Pada saat seperti ini pada umumnya sistem operasi akan sulit untuk menjaga tingkat responsif yang baik. Hasil dari kedua jenis pengukuran akan dibandingkan dan dianalisis untuk menjadi hasil dari penelitian.

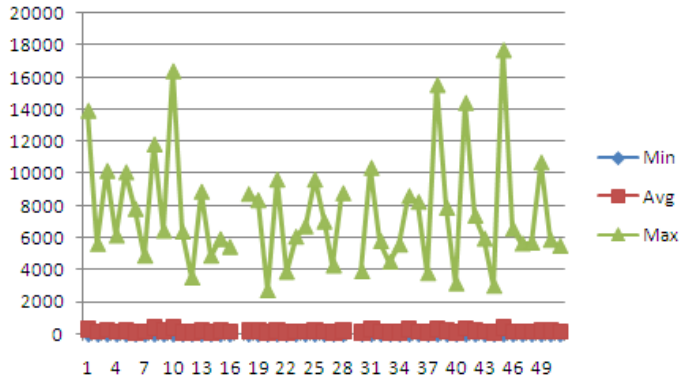
HASIL DAN PEMBAHASAN

Pengujian terhadap tingkat respons sistem operasi dilakukan dengan menjalankan program *cyclictest* dengan kondisi empat jenis kondisi beban CPU, yaitu tanpa beban, dengan beban 50%, 90%, dan 95%. Kondisi beban 90% dan 95% ditujukan untuk melihat apakah sistem operasi masih dapat memberi respons yang baik meski dalam keadaan *full load*. Data hasil pengujian ditampilkan untuk sistem operasi umum (General Purpose OS) Ubuntu 12.04-armhf-omap4 dan Linux RTOS 3.4.0-rt17+.

Berikut data hasil pengujian *cyclictest* dari GPOS Ubuntu 12.04-armhf-omap4 dan Linux RTOS 3.4.0-rt17+ dengan kondisi tanpa beban dan kondisi beban penuh, yaitu CPU *utilization* dalam kondisi >90%. Pada tiap pengujian dilakukan pengambilan data sebanyak 50 percobaan. Program dijalankan dengan prioritas tertinggi (99):



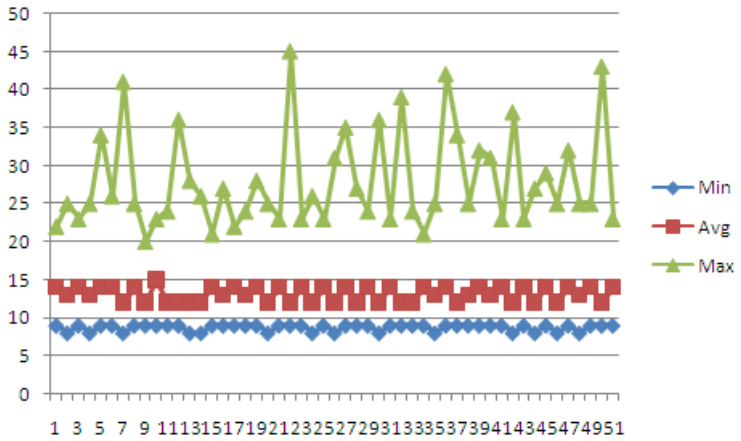
Gambar 5 Grafik Pengujian GPOS Ubuntu 12.04-armhf-omap4 dengan Tanpa Beban



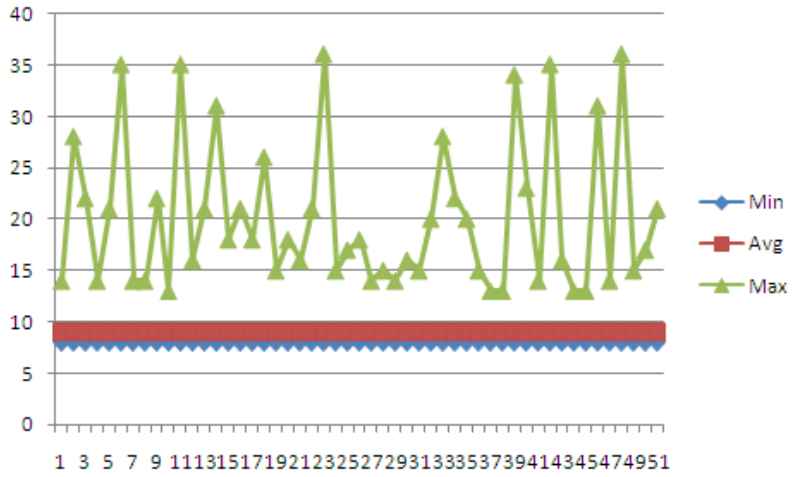
Gambar 6 Grafik Pengujian GPOS Ubuntu 12.04-armhf-omap4 dengan Beban Penuh

Dari kedua grafik tersebut, terlihat adanya beberapa nilai *latency* yang tinggi. Ini menunjukkan *response time* sistem operasi tidak dapat diprediksi (tidak stabil) dalam rentang waktu tertentu.

Pada kedua grafik data *latency* Linux RTOS 3.4.0-rt17+ baik dalam kondisi tanpa beban (Gambar 7) maupun dengan beban penuh (Gambar 8), tingkat *latency* masih dalam rentang tertentu dengan nilai maksimum *latency* 45 uS. Kondisi ini sangat jauh berbeda dengan Gambar 5 dan 6 dengan tingkat *latency* maksimum GPOS Ubuntu 12.04-armhf-omap4 sebesar 75191 uS. Terlihat bahwa RTOS mempunyai performa *latency* yang jauh lebih baik dan stabil.



Gambar 7 Grafik Pengujian Linux RTOS 3.4.0-rt17+ dengan Tanpa Beban



Gambar 8 Grafik Pengujian Linux RTOS 3.4.0-rt17+ dengan Beban Penuh

SIMPULAN

Berikut ini merupakan simpulan penelitian yang dilakukan. Pertama, pengujian terhadap *General Purpose Operating System* Ubuntu 12.04-armhf-omap4 pada PandaBoard ES dalam kondisi tanpa menghasilkan nilai maksimum *latency* 71.181 uS, nilai minimum latency 3.139 uS, rentang jarak yang sangat lebar ini menandakan proses *scheduler* yang tidak dapat diprediksi. Kedua, pengujian terhadap *General Purpose Operating System* Ubuntu 12.04-armhf-omap4 pada PandaBoard ES dalam kondisi beban penuh, yaitu saat CPU *utilization* berada pada kisaran 90-99%, menghasilkan nilai maksimum *latency* 17.712 uS, nilai minimum latency 2.697 uS, rentang jarak ini lebih baik dari kondisi tanpa beban, mengingat proses dijalankan dengan prioritas tertinggi (Priority thread = 99). Ketiga, pengujian terhadap Linux RTOS 3.4.0-rt17+ pada PandaBoard ES dalam kondisi tanpa menghasilkan nilai maksimum *latency* 45 uS, nilai minimum latency 20 uS, nilai ini jauh lebih baik dari GPOS dalam kondisi tanpa beban. Keempat, pengujian terhadap Linux RTOS 3.4.0-rt17+ pada PandaBoard ES dalam kondisi beban penuh menghasilkan nilai maksimum latency 36 uS, nilai minimum latency 13 uS, nilai ini jauh lebih baik dari GPOS dengan kondisi beban penuh. Kelima, Sistem Operasi Linux masih mampu bekerja dengan baik meskipun dalam kondisi beban penuh. Hal ini terlihat pada nilai *latency* lebih rendah dibanding kondisi tanpa beban. Keenam, secara keseluruhan terlihat RTOS memberikan tingkat respons yang sangat tinggi dibanding dengan GPOS. Hal ini terlihat dari perbandingan nilai *latency* maksimum dari GPOS sebesar 17.712 uS dan RTOS sebesar 45 uS.

Sementara saran dari penelitian ini, yaitu sebagai berikut. Pertama, dilakukan pengujian dengan menggunakan Qt Framework untuk melihat apakah PREEMPT_RT patch dapat digunakan secara transparan, tanpa harus mengetahui specific API untuk *real time system.* Kedua, melakukan pengukuran terhadap *interrupt latency* yang melakukan pengukuran terhadap kecepatan respons sistem dari *input signal* terhadap *output signal.* Ketiga, melakukan perbandingan dengan kernel *low latency* dari Ubuntu.

DAFTAR PUSTAKA

Aroca, R. V., & Caurin, G. (2009, Jul.). *A Real Time Operating Systems (RTOS) Comparison*. Brasil: Laboraorio De Mecatrinica Universidade de Sao Paulo.

Jung, Y. J., Lim, D., & Lim, C. (2010). *Measuring Responsiveness of Linux Kernel on Embedded Systems.* Embedded SW Research Department, Electronics and Telecomunications Research Institute.

Knutsson, T. (2008, Dec.). Performance evaluation of GNU/Linux for real-time application. *UPTEC IT08*. Examensarbete, Uppsala Univeritet.

Koolwal, K. (2008). *Myths and Realities of Real-time Linux Software Systems*. R&D OS Engineer VersaLogic Corp.

Leroux, P. (2012). Exacly when do you need an RTOS? *QNX Software Systems Whitepaper*.

Samuelsson, T., Akerholm, M., Nygren, P., Starner, J., & Lindh, L. (2003). *A Comparison of Multiprocessor Real-Time Operating Systems Implemented in Hardware and Software*. Sweden: ABB Robotics, Department of Computer Engineering, Malardalen University.

Vijay, S. (2006). A study of real-time embedded software systems and real-time operating systems. *M. Tech Seminar Report.* Bombay, Mumbai: Kanwal Rekhi School of Information Technology, Indian Institute of Technology.

Vun, N., Hor, H. F., & Chao, J. W. (2008). Real-time enhancements for embedded Linux. *14th IEEE International Conference of Parallel and Distributed Systems.*