Pengembangan Sistem Kendali Waktu Nyata dengan *Embedded System* berbasis Embedded Linux

Lintang Dwi F, Briliant Adhi Prabowo, Dianadewi Riswantini, Sandra Yuwana
Pusat Penelitian Informatika
Lembaga Ilmu Pengetahuan Indonesia
Komplek LIPI Gd. 20 Lt.3 Jalan Sangkuriang 154D
Bandung, Jawa Barat
Telp:(022) 2504711 Fax:(022) 2504712
Email: lintang; briliant; diana; sandra@informatika.lipi.go.id

Abstrak

Embedded System merupakan sistem komputer yang berorientasi pada aplikasi spesifik dalam skala yang bervariasi baik pada perangkat lunak maupun perangkat kerasnya. Sistem ini harus memenuhi kebutuhan akan kegunaan, kehandalan, biaya, kapasitas dan sumber daya dari suatu aplikasi. Salah satu distro yang cukup fleksible untuk mengembangkan Embedded system untuk sistem waktu nyata, adalah uClinux yang merupakan varian dari sistem operasi linux yang menyatu pada perangkat keras (embedded linux). Dalam paper ini akan dipaparkan Embedded system dalam contoh kasus pada perancangan stasiun cuaca.

Kata kunci : Embedded system, embedded linux, sistem kendali waktu nyata, uClinux

Pengembangan Sistem Kendali Waktu Nyata dengan *Embedded System* berbasis Embedded Linux

Abstrak

Embedded System merupakan sistem komputer yang berorientasi pada aplikasi spesifik dalam skala yang bervariasi baik pada perangkat lunak maupun perangkat kerasnya. Sistem ini harus memenuhi kebutuhan akan kegunaan, kehandalan, biaya, kapasitas dan sumber daya dari suatu aplikasi. Salah satu distro yang cukup fleksible untuk mengembangkan Embedded system untuk sistem waktu nyata, adalah uClinux yang merupakan varian dari sistem operasi linux yang menyatu pada perangkat keras (embedded linux). Dalam paper ini akan dipaparkan Embedded system dalam contoh kasus pada perancangan stasiun cuaca.

Kata kunci : Embedded system, embedded linux, sistem kendali waktu nyata, uClinux

1. Pendahuluan

Perkembangan yang cepat di bidang IC, menyebabkan harga CPU menjadi semakin murah. Perkembangan ini menyebabkan kebutuhan *Embedded system* semakin meningkat, bahkan diperkirakan lebih dari enam milyar microprocessor baru digunakan untuk keperluan *Embedded system* setiap tahunnya, dan hanya sekitar dua persennya yang digunakan untuk *general purposed computer*[1].

Aplikasi *Embedded system* semakin kompleks, maka sistem operasi yang mendukungnya pun menjadi suatu isu yang sangat penting. Dalam perkembangannya sebuah desain *Embedded system* juga dituntut memiliki reliabilitas yang sangat handal dan dapat bekerja secara waktu nyata, karena berbagai kebutuhan dari fungsi spesifik yang telah bergantung pada *Embedded system* tersebut.

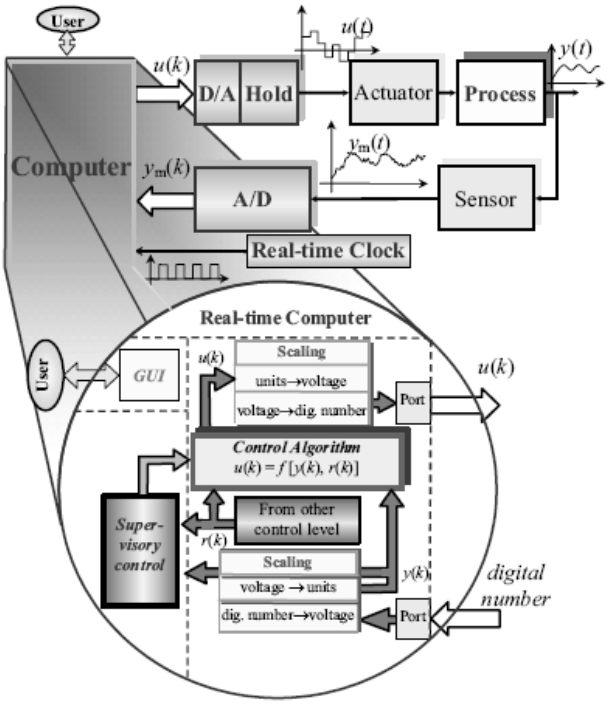
Tujuan dari penulisan ini adalah untuk mengkaji pengembangan sistem kendali waktu nyata pada embedded system yang menggunakan *framework* uClinux.

2. *Embedded System*

Embedded System merupakan sistem komputer yang berorientasi pada aplikasi spesifik dalam skala yang bervariasi baik pada perangkat lunak maupun perangkat kerasnya. Sistem ini harus memenuhi kebutuhan akan kegunaan, kehandalan, biaya, kapasitas dan sumber daya dari suatu aplikasi.

Single Board Computer (SBC) merupakan sebuah komputer lengkap yang dibangun pada sebuah *Printed Circuit Board* (PCB). Seperti halnya sebuah komputer pada umumnya dalam sebuah SBC sudah terdapat *microprocessor* dengan *Random Access Memory* (RAM), IO, ADC/DAC dan semua fitur lain yang diperlukan untuk berfungsi sebagai komputer dalam sebuah *board.*

Sebuah SBC dapat diimplementasikan dalam berbagai *Embedded system*, seperti pada sistem kendali waktu nyata, *monitoring*, ataupun *interfacing* IO. Sebuah *overview* sistem kendali komputer dapat diilustrasikan pada Gambar 1.



Gambar 1. Overview Sistem Kendali Komputer[1]

3. Sistem Kendali Waktu Nyata

Tujuan dari kendali komputer adalah menggunakan komputer untuk memanipulasi masukan-masukan pada sistem dinamik sehingga

dapat membuat sistem berperilaku sesuai dengan yang diharapkan. Kendali komputer digunakan di banyak area aplikasi seperti, otomasi pabrik, proses kendali, robotika, sistem otomotif dan lainnya. Pada aplikasi-aplikasi tersebut, komputer digunakan untuk pengontrol proses dan diharapkan untuk dapat bereaksi dalam konstrain waktu yang akurat sesuai dengan perubahan eksternal berdasarkan kebutuhan aplikasi.

Menurut standar dari IEEE POSIX (*Portable Operation System Interface for Computer Environments*), sistem waktu nyata dapat diasumsikan sebagai sebuah sistem yang ketepatan hasilnya tidak hanya tergantung dari ketepatan komputasi logikal, namun juga dapat memenuhi rentang waktu yang diharapkan[1].

Dari pengertian tersebut dapat dicermati, bahwa komputasi waktu nyata tidak sama dengan komputasi cepat *(fast computing)* yang bertujuan untuk memperoleh hasil secepat mungkin. Komputasi waktu nyata bertujuan untuk memperoleh hasil dalam rentang waktu yang sesuai dengan batas toleransi.

Berdasarkan *deadline* rentang waktu hasil yang diharapkan, sebuah sistem waktu nyata dapat dikategorikan dalam *hard real time system* dan *soft real time system*. Pada *hard real time system,* tenggat waktu hasil harus terpenuhi untuk menghindari kesalahan sistem yang berbahaya. Sedangkan *soft real time system*, keterlambatan waktu saat hasil dipenuhi tidak menyebabkan bahaya yang sangat fatal.

Sistem waktu nyata harus memenuhi karakteristik sebagai berikut[1]:

- *preemptive multitasking* untuk memenuhi *hard real time task*
- memenuhi POSIX (*Portable Operating System Interface*) yaitu standar IEEE utnuk sistem operasi. Kompatibilitas dan kebutuhan untuk sistem waktu nyata adalah norm 1003.1b, 1003.1d, 1003.1j
- mendukung penjadualan untuk waktu nyata. Beberapa algoritma yang mendukung *task* ini misalnya RMS, EDF, MLF dan MUF
- *latency* yang kecil. Untuk 1 ms waktu *sampling* harus bisa dipenuhi untuk beberapa *control loops*

Selain itu karakteristik tersebut, integrasi dengan *Labview,* yaitu lingkungan pemrograman grafis dari National Instrumen Inc yang menggabungkan pengembangan dengan bahasa pemrograman yang mendukung presentasi data dan visualisasi dalam 2-D dan 3-D. Pada implementasi *software* waktu nyata penggunaan Matlab/Simulink/RTW tidak menjadi *tool* khusus, melainkan hanya merupakan fasilitas tambahan. Karena itu SOWN dengan lingkungan pengembangan untuk menulis *software* juga diperlukan.

Dengan menggunakan SBC sebagai sistem kendali, dapat dirancang sebuah sistem kendali waktu nyata pada implementasinya. SBC sebagai *Embedded system* untuk keperluan sistem waktu nyata dapat dikembangkan dengan menggunakan 2 pendekatan yang bisa dilakukan untuk mendapatkan *multitasking* yaitu dengan menggunakan bahasa waktu nyata yang *concurrent* atau dengan menggunakan bahasa *sequensial* pada *framework* sistem operasi waktu nyata.

4. Embedded Linux

Linux merupakan sistem operasi yang bersifat modular, sehingga mudah untuk dilakukan penyesuaian (*configurable*) terhadap lingkungan pengoperasian dengan cara mengurangi program *utility*, *tools,* dan *services* lainnya yang tidak diperlukan dalam *Embedded system*. Linux telah berhasil dijalankan pada prosesor dengan berbagai arsitektur dan berbagai tipe CPU.

Keuntungan menggunakan Linux diantaranya, Linux merupakan OS *open source* yang telah banyak mendukung divais, *system file*, protokol jaringan, memperbaiki *bugs*, serta terus menambahkan layanan baru secara konstan dan teruji karena di dukung oleh komunitas pemrogram dan pengguna yang sangat besar.

Linux dapat dikonversikan menjadi sistem operasi yang bersifat waktu nyata dengan memodifikasi kernel agar dapat menjalankan proses di bawah penjadualan waktu nyata. Sistem operasi waktu nyata menyediakan layanan yang berkaitan dengan waktu respon pada tingkat seperti yang disyaratkan pada sistem waktu nyata. Program aplikasi pada lingkungan ini dibuat berbasis *interrupt*.

Salah satu sistem operasi yang banyak digunakan dalam *Embedded system* adalah *embedded linux*, sebuah distro linux yang menggunakan kernel 2.x tertanam pada mikro-kontroler[2]. Keunggulan utama dari *embedded linux* adalah bahwa sistem operasi ini mempunyai fungsi yang lengkap. Dengan dukungan layanan yang baik terhadap jaringan komunikasi menyebabkan sistem operasi ini menjadi pilihan yang tepat dalam mengembangkan *Embedded system.*

Embedded linux memiliki sistem platform *cross-compilation*, *Embedded System* umumnya mempunyai *resources* yang terbatas sehingga kita tidak dapat menginstal *tool chain* dan *source code* diatasnya. Sebagai gantinya kita gunakan PC

sebagai *computer host,* dan menggunakan *cross-compiler* untuk membuat program yang jalan di atas *Embedded system* (juga disebut sebagai *target system*).

uClinux (***m****icro* ***C****ontroller linux)* adalah salah satu *embedded linux* open source yang digunakan pada perangkat *embedded* yang berfungsi sebagai penghubung antara mikrokontroler dan SBC. Sistem operasi ini merupakan *clone* dari linux untuk mikrokontroler yang tidak mempunyai MMU (*Memory Management Unit*) dan membutuhkan sumber daya serta memori yang kecil.

Komponen ukuran menjadi hal utama yang menjadi konsentrasi dari *Embedded system*. Mikrokontroler dengan unit manajemen memori (MMU) tertanam pada perangkat keras akan cenderung menjadi kompleks dan mahal, dan bukan tipe yang sesuai untuk ukuran kecil sebuah *Embedded system* [3].

Produk dari *open source project* ini dirancang dengan memanfaatkan fitur utama dari kernel linux antara lain *process control, file systems, networking* dan *device driver.*

Process control dilengkapi dengan *scheduler* baik untuk *single process* ataupun *multi processes.* Manajemen memori pada uClinux dilakukan secara manual. uClinux didukung oleh pustaka yang cukup lengkap (uClib, glibc, openssl, libpcap, libm, libdes, zlib). Pustaka uClib banyak digunakan karena bersifat "*lightweight*" dan kompatibel dengan bahasa C.

Embedded Linux sebenarnya bukanlah sebuah sistem operasi waktu nyata (SOWN), namun dapat dioptimalkan sehingga mampu menjadi SOWN dengan cara mereduksi beberapa fungsi dan menyesuaikan dengan fungsi spesifik yang dibutuhan. Reduksi tersebut meliputi pengurangan *device drivers* dan perangkat lunak aplikasi, juga dengan memodifikasi kernel sehingga dapat berjalan sebagai *task* pada proses penjadualan waktu nyata (*wikipedia.org*).

Penggunaan SOWN pada SBC memungkinkan sistem operasi melakukan *multitasking*. *Multitasking* adalah kemampuan sebuah sistem operasi untuk mengeksekusi beberapa *task* yang terpisah namun nampak dapat bekerja secara simultan[4]. Pada dasarnya sebuah processor tidak bisa benar-benar mengeksekusi beberapa *task* secara bersama, namun sebuah sistem dapat dirancang dengan *scheduling task* yang ketat sehingga processor akan mengerjakan *task-task* tersebut secara bergantian dengan cepat, sehingga nampak dieksekusi secara bersama-sama.

Pada sebuah sistem *multitasking*, reabilitas komunikasi antara berbagai *task* dalam sistem tersebut juga diperlukan. Komunikasi yang gagal antara *task-task* dalam sistem *multitasking* akan membuat sistem berjalan tidak simultan. Komunikasi tersebut diantaranya terjadi pada *task* yang sedang di eksekusi dengan *task* yang menunggu perintah melanjutkan eksekusi. Ketepatan pengiriman informasi *brekpoint* dari *task* yang sedang dieksekusi dengan informasi untuk melanjutkan eksekusi bagi *task* selanjutnya harus diatur dengan tepat.

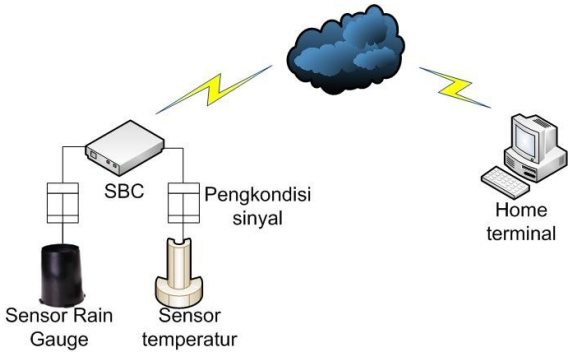
Kebutuhan dasar lainnya pada sistem *multitasking* adalah pengaturan prioritas. Pengatur prioritas akan bekerja dengan persetujuan *context switcher,* dan akan menentukan *task* mana yang harus dieksekusi selanjutnya. Pengaturan prioritas tersebut bisa dilakukan oleh *task secheduler* maupun oleh interupsi pada sistem yang sedang berjalan.

Dalam sebuah *multitasking* waktu nyata dibutuhkan sebuah pengontrol waktu (*timing control)*. Bagian ini akan berinteraksi dengan pengatur prioritas, jika pengatur prioritas akan menentukan *task* mana yang selanjutnya akan dieksekusi, maka t*iming control* akan bertanggungjawab pada kapan dan berapa lama *task* selanjutnya akan dieksekusi.

5. Studi kasus Aplikasi SKWN pada stasiun cuaca

Aplikasi SKWN salah satunya adalah pada Stasiun Cuaca. Stasiun Cuaca dirancang untuk memantau banyaknya curah hujan dan temperatur di suatu daerah. Kedua nilai tersebut adalah unsur cuaca yang merupakan dampak dari perubahan iklim. Data-data mengenai curah hujan diantaranya diperlukan untuk pertanian, perkebunan, maupun untuk sistem peringatan dini bencana. Sedangkan pemantauan temperatur pada jangka waktu singkat dapat digunakan untuk kenyamanan pengguna, Dan untuk jangka waktu panjang dapat dimanfaatkan untuk pemetaan iklim.

Nilai hasil pemantauan dari sensor-sensor pada stasiun cuaca yang diletakkan di suatu daerah akan langsung dapat terbaca pada *home terminal,* sehingga pengguna dapat segera mengambil keputusan yang berkaitan dengan penggunaan data-data tersebut. Diagram sistem untuk stasiun pemantau cuaca adalah seperti pada gambar 3 berikut:

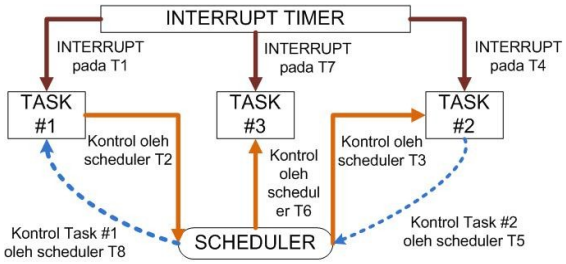


Gambar 3. Diagram sistem stasiun cuaca

Input dari stasiun cuaca berasal dari sensor *rain gauge* yang akan mencacah banyaknya curah hujan dan sensor temperatur. Data-data yang diterima dari sensor berupa sinyal analog yang harus dikondisikan dengan pengkondisi sinyal sebelum dibaca oleh SBC. Kemudian SBC akan merubah sinyal analog yang sudah dikondisikan menjadi sinyal digital, dan mengirimkan data-data tersebut kepada *home terminal* setiap 1 menit melalui jaringan publik.

Rancangan stasiun cuaca ini adalah sistem dengan *soft real-time task*, yang tidak memiliki akibat fatal jika sistem tidak memenuhi batas waktu yang ditentukan. Walaupun begitu, sistem tetap dirancang untuk bisa memberikan data secara akurat. Karena itu, dipilih penggunaan *interrupt timer* ketimbang *looping,* untuk menjalankan program terus menerus. *Interrupt timer* akan men-*trigger* masing-masing task untuk dijalankan, sehingga data dipastikan dapat di-*sampling* sesuai dengan waktu yang dijadualkan tanpa harus terpengaruh oleh proses dari *task* lain.

Pada sistem stasiun cuaca tersebut, ada 3 *task* yang dijalankan oleh SBC sebagai otak dari sistem, *task* pertama yaitu untuk membaca data dari *rain gauge*, *task* kedua membaca data dari sensor temperatur dan *task* ketiga untuk mengirimkan kepada *home terminal*. Ketiga *task* tersebut dijalankan secara *pre-emptive multitasking* yaitu sebuah desain *multitasking* pada sebuah sistem operasi di mana eksekusi sebuah *task* dapat diinterupsi setiap saat oleh perintah internal atau pun eksternal dari sistem operasi tersebut [5]. Diagram prosesnya adalah seperti pada gambar 4 berikut:



Gambar 4. Diagram proses *pre-emptive multitasking* pada stasiun cuaca

6. Kesimpulan

Dalam studi kasus perancangan stasiun cuaca sistem yang merupakan *embedded system* dengan berbasis uClinux, yang memenuhi karakteristik untuk sistem kendali waktu nyata diantaranya adalah : menggunakan *pre-emptive multitasking*, memenuhi standar POSIX, mendukung penjadualan untuk waktu nyata dan memiliki *latency* yang kecil.

Daftar Pustaka

[1] Gambier, A, 2004. *Real Time Control System: A Tutorial*. Proceedings Control Conference 5th Asian.

[2] Wang, Catherine Lingxia, Bo Yao, Yang Yang, Zhengyong Zhu, *A Survey of Embedded Operating System.*

[3] Durrant, Michael and Michael Leslie Of Lineo, *How uClinux provides MMU-less processor with an alternative*, http://uclinux.home.at.

[4] Curtis, Keith E.. 2006. *Embedded Multitasking with Small Microcontrollers*. Newnes

[5] Intel Corporation. 2009. Intel® EP80579 *Integrated Processor Product Line Single-Board Computer*. Application note

[6] Labib, Gamal Ali. 2007. *The basics of embedded multitasking on a PIC: Part 1 – Reentrant multitasking.* Emmbedded.com article.