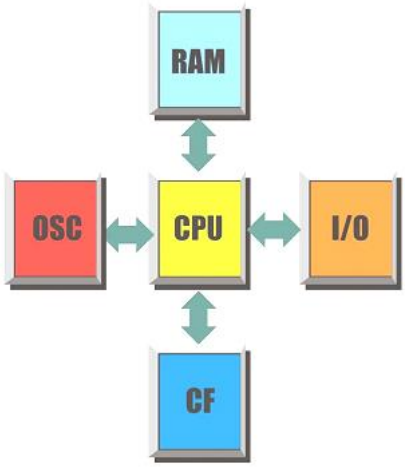
II. TINJAUAN PUSTAKA

A. *Embedded System*

Embedded system merupakan komputer yang didesain untuk melakukan satu atau beberapa fungsi tertentu. *Embedded system* biasanya dibenamkan sebagai bagian dalam perangkat lengkap disamping perangkat keras dan bagian mekanik. *Embedded system* umumnya digunakan untuk mengontrol berbagai perangkat umum yang banyak digunakan sekarang ini [4].



Gambar 1. Blok Diagram *Embedded PC*

Sejak *embedded system* didedikasikan sebagai fungsi khusus, para *design engineer* dapat mengoptimalisasinya, mereduksi dimensi dan biaya produksi atau

meningkatkan kemampuan dan performa. Beberapa *embedded system* diproduksi secara masa karena dinilai lebih menguntungkan dari pertimbangan ekonomi [4].

1. Karakteristik

Embedded system memiliki beberapa karakteristik, antara lain :

a. *Embedded system* didesain untuk melakukan beberapa fungsi khusus berbeda dengan komputer pada umumnya yang digunakan untuk multifungsi. Beberapa juga memiliki kemampuan *real-time* bahkan yang lain hanya membutuhkan peralatan dengan performa rendah, untuk mereduksi biaya produksi,
b. *Embedded system* tidak selalu merupakan perangkat yang berdiri sendiri. Banyak *embedded system* yang terdisi dari bagian komputer yang berukuran kecil di dalam perangkat lebih besar yang melakukan tujuan umum,
c. Instruksi program yang ditulis untuk *embedded system* disebut sebagai *firmware* dan disimpan di dalam ROM atau memori *flash* [4].

2. *User Interface*

Embedded system terkadang tidak dilengkapi perangkat antar muka dengan pengguna karena hanya didedikasikan untuk satu fungsi. Akan tetapi ada pula yang dilengkapi dengan perangkat antar muka yang kompleks menyerupai komputer *desktop* modern [4].

3. Kompleksitas

Perangkat *embedded system* yang sederhana menggunakan tombol, led dan karakter atau digit kecil sebagai penampil, bahkan ada yang menggunakan sistem menu yang sederhana. Perangkat *embedded system* yang kompleks dilengkapi layar grafis dengan sensor sentuh bahkan juga dilengkapi dengan antar muka *world wide web* [4].

4. *CPU Platforms*

Prosesor yang digunakan dalam *embedded system* dibedakan menjadi dua kategori yaitu mikroprosesor dan mikrokontroler dengan dilengkapi berbagai periferal pada *chip*-nya sehingga dapat mereduksi biaya dan dimensi. Berbeda halnya dengan *personal computer* maupun *server*, yang secara umum menggunakan standar arsitektur CPU, diantaranya baik *Von Neumann* maupun *Havard architectures*, baik RISC maupun non-RISC [4].

5. Periferal

Embedded system berkomunikasi dengan dunia luar melalui perangkat-perangkat periferal, antara lain :

a. Antar muka komunikasi serial (RS-232, RS-422, RS-485),
b. Antar muka komunikasi serial sinkron (I2C, SPI, ESSI),
c. *Universal Serial Bus* (USB),
d. Kartu multimedia (*SD cards*, *Compact flash*),
e. Jaringan (*Ethernet*, *Controller area network*),
f. *Analog to Digital/Digital to Analog* (ADC/DAC) [4].

6. *Tools*

Umumnya *embedded system* didesain dengan menggunakan *compilers*, *assemblers* dan *debugger* untuk mengembangkan *software embedded system* tersebut. Meskipun terkadang juga menggunakan beberapa *tools* yang lebih spesifik untuk tujuan khusus [4].

Seiring dengan perkembangan teknologi yang semakin maju, *embedded system* pun mengalami perkembangan yang signifikan dari segi *hardware* maupun *software*. Dari sisi *hardware*, perkembangan yang paling menonjol adalah pada *board embedded* yang semakin beragam dengan didukung oleh berbagai variasi artistektur prosesor yang digunakan. Tabel 1 berikut ini merepresentasikan perkembangan *board embedded* hingga menjadi *mainstream* untuk 10 tahun terakhir [13].

Tabel 1. *Mainstream board embedded* 10 tahun terakhir

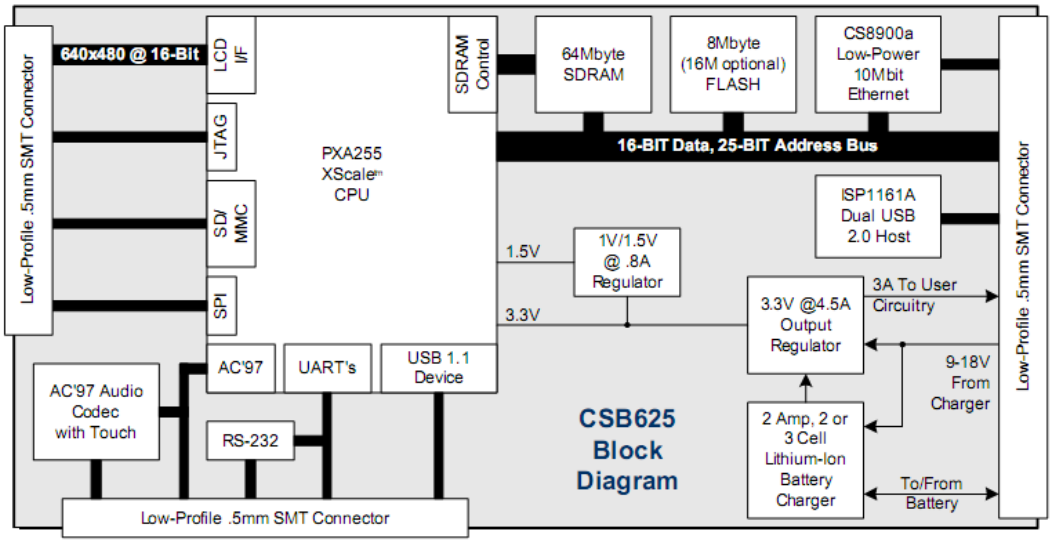
Tahun	Jenis Board	Prosesor yang digunakan
2000	Aplio/TRIO	ARM7TDMI ASIC
2001	Altera Excaliber	ARM922T
2002	Freescale DragonBall MX1	ARM 9/11
2003	Atmel AT75Cxxx	ARM 70
2004	Atmel AT91RM9200	ARM 920T
2005	Alchemy Au1200	MIPS
2006	IEI WAFER LX 800 3.5"	AMD Geode GX3
2007	Atmel AT91SAM7S-EK	ARM7TDMI
2008	TS-7800 SBC	ARM 9
2009	Beagle Board OMAP3530	ARM 926
2010	PC/104 SBC	AMD LX800 & Intel Atom N270

B. *Embedded PC* Cogent CSB625

Embedded PC CSB625 didesain untuk daya rendah, bersifat *portable*, serta menggambarkan sebuah terobosan secara fungsional dalam ukuran dan daya. *Embedded PC* CSB625 diimplementasikan dengan menggunakan perangkat yang merupakan komponen penyusun dari sebuah sistem komputer. Komponen penyusun beserta fitur CSB625 antara lain :

1. Prosesor Intel PXA255 400MHz 32-bit,
2. Memori SDRAM 64MB 32-bit,
3. Memori *Flash* 8MB 16-bit,
4. *Ethernet Controller* CS8900a 10MBit,
5. *Port* Serial RS-232,
6. USB 2.0 *Controller* Philips ISI1161A 12MBit,
7. Konektor Internal SD/MMC,
8. Antarmuka *PCMCIA/Compact Flash*,
9. Konektor Antarmuka JTAG,
10. Regulator 3,3V *on board* dengan range *input* 9-20V,
11. Berdaya rendah, secara umum <1W dan maksimum 5W [2].

Blok diagram dari CSB625 ditunjukkan oleh gambar 2 berikut ini.



Gambar 2. Blok Diagram Cogent CSB625

1. Prosesor Intel PXA255

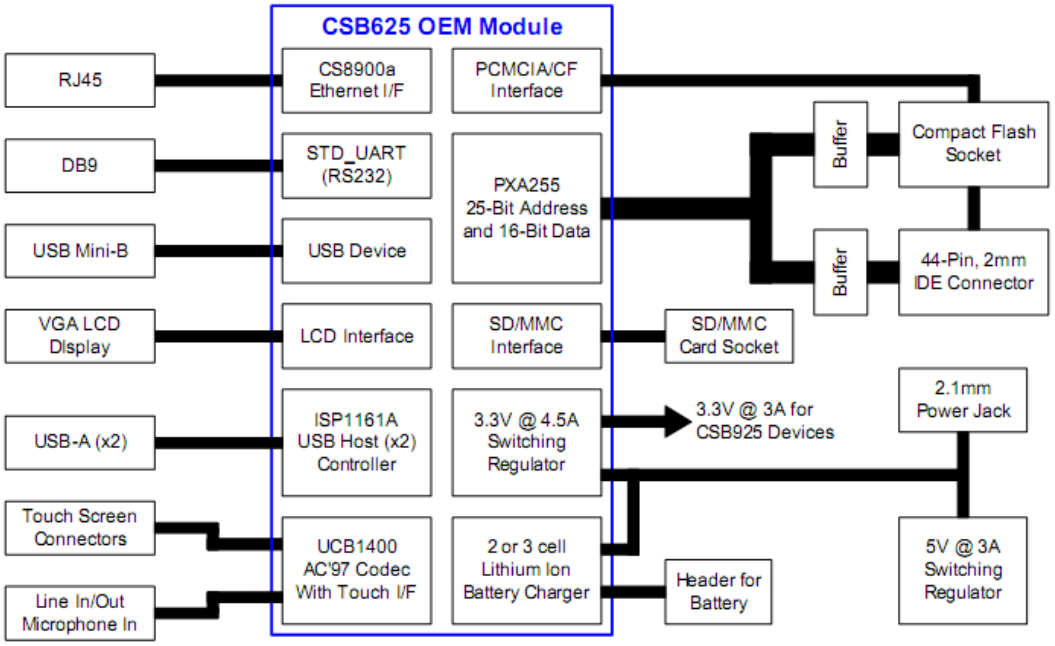
CSB625 menggunakan mikroprosesor Intel PXA255 berbasis Intel® XScaleTM *Microarchitecture*, prosesor ini mengkombinasikan konsumsi daya yang rendah dengan performa yang tinggi. Kemampuan dari prosesor jenis ini dikombinasi dengan adanya *chip* LCD, kartu memori SD, *audio* AC'97, SSP dan antarmuka *Compact Flash* yang memberikan PXA255 kemampuan *power low-cost*, produk dengan performa tinggi untuk telematika, industri otomasi, aplikasi medis dan sebagainya [2].

Intel PXA255 merupakan prosesor 32-bit berarsitektur ARM (*Advanced RISC Machine*) V5TE. ARM pada mulanya merupakan jenis prosesore desktop yang sekarang ini didominasi oleh keluarga x86. Desain arsitektur ARM yang sederhana menjadikannya cocok untuk aplikasi berdaya rendah. Hal ini yang membuat prosesor ARM mendominasi pasar *mobile electronics* dan *embedded system* dimana membutuhkan daya dan harga yang rendah [2].

Gambar 3. Prosesor Intel PXA255

2. CSB925 *Development Board*

CSB925 berfungsi untuk mengkonversi CSB625 *Micro Single Board Computer* (uSBC) ke dalam sebuah *platform* pengembangan produk lebih lanjut. Karena CSB625 dioptimalkan untuk penggunaan OEM, sehingga tidak memiliki *port* serial DB9, RJ45, sakelar dan sebagainya. CSB925 menyediakan konektor-konektor standar tersebut, selain itu juga dilengkapi dengan akses ke GPIO dan fungsi periferal dari CSB625 yang dibutuhkan selama produk berada dalam tahap pengembangan [9].



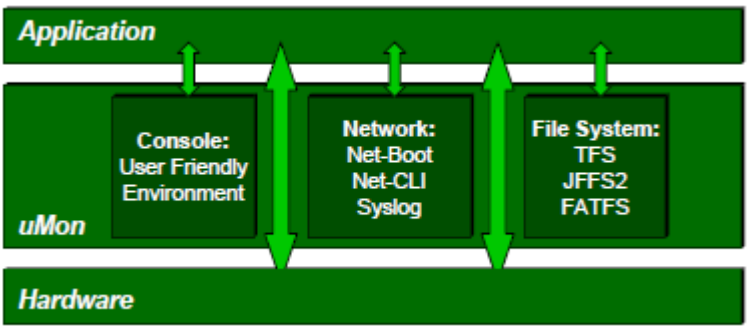
Gambar 4. Blok Diagram Cogent CSB925

3. *Bootloader Micromonitor* (uMon)

Micromonitor adalah aplikasi *bootloader* untuk sistem *embedded* yang bersifat *open source*. *Micromonitor* memiliki kemampuan yang bersifat fleksibel dalam proses pengembangan sistem *embedded* maupun dalam pengembangan aplikasi di dalamnya. *Bootloader* ini digunakan untuk pengembangan sistem *embedded* dengan mekanisme penyimpanan data maupun program serta antar muka komunikasi yang sesuai dengan standar industri. Beberapa fitur dari *bootloader* ini antara lain :

a. Mendukung untuk *file system* JFFS2 dan FAT
b. Memiliki TFTP *client/server* untuk transfer data dalam jaringan
c. Memiliki Xmodem untuk transfer data serial

d. Mendukung berbagai macam jenis arsitektur prosesor seperti PowerPC, MIPS, ARM/Xscale, 68K/ColdFire, Blackfin, SH2, MicroBlaze, Nios [12].



Gambar 5. Hubungan Umon dengan aplikasi dan *hardware*

C. Sistem Operasi Linux

Linux merupakan sebuah sistem operasi alternatif yang saat ini banyak digunakan selain sistem operasi Microsoft Windows. Linux saat ini sudah banyak digunakan pada komputer perumahan maupun komputer perkantoran. Linux dirancang dengan menggunakan bahasa pemrograman C yang juga digunakan untuk merancang sistem operasi lain seperti FreeBSD, NetBSD dan OpenBSD. Linux merupakan suatu aplikasi istimewa yang berfungsi sebagai penghubung antara perangkat keras yang ada pada sistem komputer dengan aplikasi yang berjalan pada komputer tersebut, yang disebut sebagai *kernel* [1].

1. Komposisi Sistem Operasi Linux

Sistem operasi linux secara garis besar dapat dibagi menjadi empat bagian :

a. Aplikasi Pengguna

Aplikasi pengguna merupakan kumpulan dari berbagai aplikasi yang berjalan pada sistem operasi linux, misalnya Open Office yang merupakan aplikasi untuk membuat dokumen.

b. Layanan Sistem Operasi

Layanan sistem operasi merupakan bagian dari aplikasi yang tersedia pada sistem operasi, misalnya *command shell, compiler*.

c. *Kernel* Linux

Kernel linux merupakan mediator antara *resource* perangkat keras dengan aplikasi yang membutuhkan layanan perangkat keras tersebut.

d. Pengontrol Perangkat Keras (*Driver*)

Pengontrol perangkat keras merupakan sebuah subsistem yang merupakan bagian dari *kernel* linux. Bagian ini berfungsi untuk mengatur layanan yang diberikan oleh sistem operasi terhadap *resource* perangkat keras yang tersedia pada sistem computer [1].

Gambar 6. Bagian dari Sistem Operasi Linux

2. Struktur *Kernel* Linux

Kernel linux terdiri dari lima bagian sub sistem utama, yaitu :

a. *Process Scheduler* (SCHED)

Process Scheduler bertanggung jawab untuk mengontrol proses yang mengakses CPU. *Scheduler* akan memberikan kesempatan yang sama untuk setiap proses dalam menggunakan CPU dan juga memastikan aksi yang dilakukan oleh perangkat keras dilakukan oleh *kernel* secara tepat waktu.

b. *Memory Manager* (MM)

Memory manager berfungsi untuk memberikan kesempatan pada setiap proses untuk mengakses memori secara aman. *Memory manager* juga melayani *virtual memory* yang memberikan kemampuan kepada sistem untuk memiliki memori yang melebihi dari memori sistem yang dimiliki.

c. *Virtual File System* (VFS)

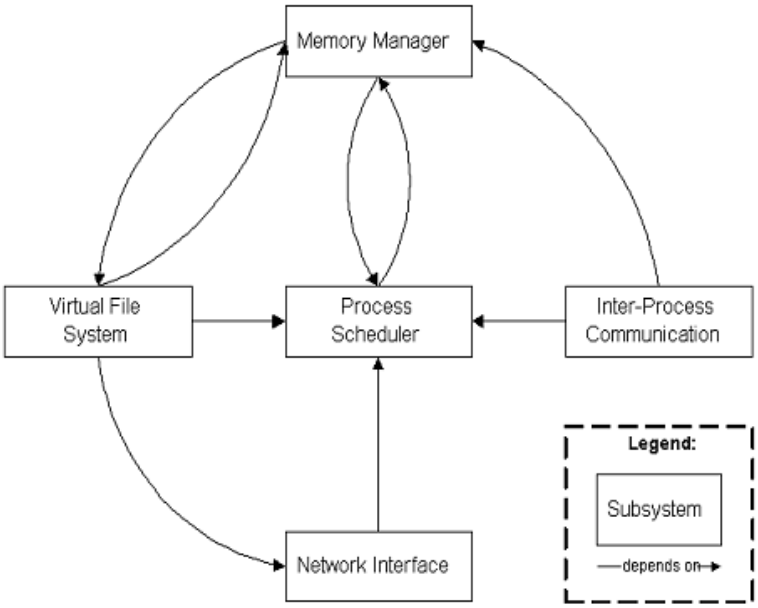
Virtual file system memberikan abstraksi terhadap pengaksesan *file*. VFS memberikan layanan untuk berbagai kebutuan *filesystem* yang sesuai dengan sistem operasi lain misalnya FAT, NTFS, UFS dan *filesystem* lainnya.

d. *Network Interface* (NET)

Bagian ini memberikan akses untuk melakukan komunikasi jaringan dengan komputer lainnya. Bagian ini juga memberikan layanan untuk mengakses berbagai perangkat jaringan.

e. *Inter-Process Communication* (IPC)

IPC merupakan sebuah mekanisme dalam *kernel* linux sehingga memungkinkan adanya komunikasi antar proses dalam sebuah sistem linux [1].



Gambar 7. Struktur *Kernel* Linux

3. Struktur Direktori dari Kode Program *Kernel* Linux

Struktur direktori dari kode program *kernel* linux terdiri dari beberapa bagian yang setiap bagian tersebut biasanya memiliki fungsi tersendiri. Direktori-direktori tersebut biasanya diberi nama dengan fungsi dari kode program yang terapat dalam direktori tersebut, misalnya direktori “mm” atau “mmnommu” adalah direktori untuk fungsi manajemen memori [1].

Isi dari direktori pada kode program *kernel* linux adalah sebagai berikut :

a. *Arch*

Direktori ini berisi kode program yang spesifik untuk arsitektur tertentu, arsitektur yang didukung oleh sistem operasi linux antara lain ALPHA, ARM, INTEL, MOTOROLA, SPARC dan beberapa jenis prosesor lainnya.

b. *Crypto*

Direktori ini berisi kode program untuk fungsi kriptografi yang diimplementasikan dalam *kernel* linux.

c. *Documentation*

Direktori ini berisi dokumentasi dari *kernel* linux yang menjelaskan mengenai pemakaian *driver*, opsi dan konfigurasi dari sistem operasi.

d. *Drivers*

Direktori ini berisi kode program untuk pengendali dari perangkat keras yang akan menggunakan sistem operasi ini, misalnya driver untuk CD-ROM, *keyboard*, *mouse*.

e. *Fs*

Direktori ini berisi kode program untuk pengendali dari *filesystem* yang akan digunakan dalam sistem operasi ini, pengendali ini bergunsi untuk membaca dan mungkin untuk menulis dalam *filesystem* yang di dukung oleh sistem operasi ini.

f. *Include*

Direktori ini berisi *file-file header* untuk sistem operasi linux seriap arsitektur yang didukung oleh sistem operasi ini. *File header* merupakan definisi-definisi dari kemampuan sistem operasi yang akan digunakan ataupun definisi dari kemampuan perangkat keras yang akan menggunakan sistem operasi ini.

g. *Init*

Direktori ini berisi kode program untuk inisialisasi dari sistem operasi linux seperti misalnya menangkap parameter yang diberikan oleh *bootloader*.

h. *Ipc*

Direktori ini berisi kode program untuk fungsi *Inter Process Communication* (IPC). IPC adalah suatu fungsi untuk membagi memori sehingga memori tersebut dapat digunakan secara bersamaan oleh setiap aplikasi yang membutuhkan data dari memori tersebut.

i. *Kernel*

Direktori ini berisi kode program untuk *kernel* linux yang berisi fungsi-fungsi yang berhubungan dengan *kernel* linux misalnya *scheduler, sysctl, dma, process accounting*.

j. *Lib*

Direktori ini berisi kode program untuk fungsi *library* dari *kernel* linux. Fungsi yang didukung misalnya *checksum*, vprintf, string.

k. *Mm atau mmnommu*

Direktori ini berisi kode program untuk fungsi manajemen memori, direktori “mm” berisi kode program untuk arsitektur yang menukung manajemen memori seperti Intel sedangkan direktori “mmnommu” berisi kode program untuk arsitektur yang tidak menggunakan manajemen memori.

l. *Net*

Direktori ini berisi kode program untuk fungsi *networking*, misalnya untuk fungsi serial *networking*, IPv4, IPv6 dan fungsi *network* lainnya.

m. *Scripts*

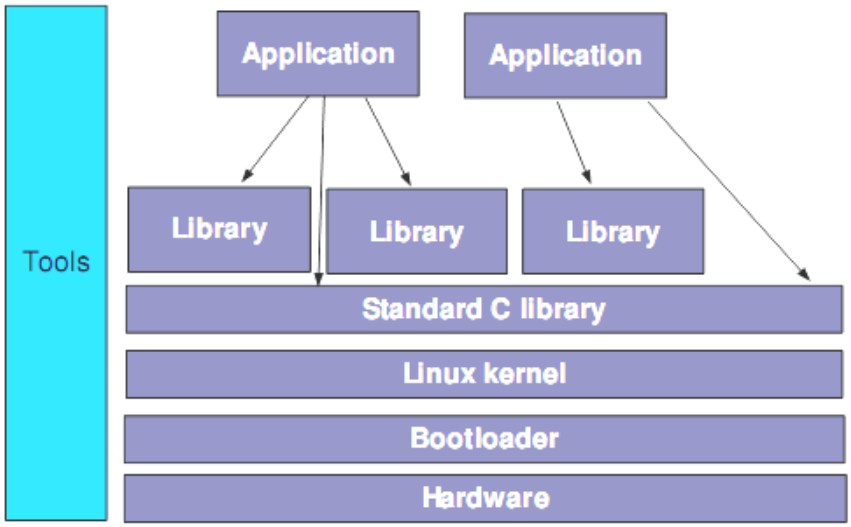
Direktori ini berisi kumpulan *scripts* yang berguna untuk proses kompilasi *kernel* linux. *Scripts* yang digunakan misalnya mkversion untuk memperbaharui versi dari *kernel* linux pada saat kompilasi [1].

4. *Embedded Linux*

Embedded linux merupakan sistem operasi linux yang digunakan dalam sistem komputer *embedded* seperti Handphone, PDA, perlengkapan jaringan, mesin pengendali, industri otomasi, perlengkapan navigasi dan instrumentasi kesehatan. Berdasarkan survei yang dilakukan oleh Venture Development Corporation, linux telah digunakan oleh 18% *embedded engineers* [7].

Berbeda dengan dengan linux versi *desktop* dan *server*, versi *embedded* di desain untuk peralatan dengan sumber daya yang dibatasi. Perangkat *embedded* umumnya menggunakan RAM yang minimal dan media

penyimpanan sekunder seperti memori *flash*. Sejak perangkat *embedded* digunakan untuk kebutuhan spesifik dibandingkan kebutuhan umum, para pengembang mengoptimisasi distribusi *embedded* linux untuk target konfigurasi perangkat keras dan penggunaan pada situasi yang spesifik. Optimisasi yang dilakukan termasuk mereduksi jumlah *device driver* dan *software* aplikasi serta memodifikasi *kernel* linux untuk menjadi sistem operasi yang bersifat *real-time*. Dalam membangun sebuah sistem *embedded* linux, umumnya digunakan beberapa aplikasi utilititas yang bersifat *freeware* seperti *busybox, glibc C standard library* [7].



Gambar 8. Arsitektur Dasar E*mbedded Linux*

Keuntungan *embedded* linux dibandingkan sistem operasi *embedded* lainnya meliputi tidak adanya biaya lisensi, *kernel* yang stabil, mendukung berbagai macam arsitektur perangkat keras dan dapat memodifikasi dan meredistribusi *source code*. Sedangkan kerugian *embedded* linux meliputi kompleksitas akses memori mode *user* dan mode *kernel* serta kompleksitas kerangka *device drivers* [7].

Dalam membangun sistem operasi *embedded linux* untuk target, ada empat langkah utama yang harus dilakukan yaitu :

a. Mendeskripsikan komponen sistem operasi

   Komponen sistem merupakan daftar kelengkapan yang dibutuhkan dalam membangun sistem operasi *embedded* linux. Daftar komponen ini lebih cenderung kepada fitur-fitur yang akan dikembangkan nantinya. Fitur-fitur yang akan dikembangkan tentunya berkaitan dengan *hardware* yang dimiliki. Dengan perpaduan tersebut maka dapat ditentukan *tools* yang akan digunakan untuk membangun sistem operasi tersebut.

b. Mengkonfigurasi dan membangun kernel

   Kernel yang dipilih berkaitan dengan fitur-fitur yang telah dideskripsikan sebelumnya, apakah kernel yang akan digunakan mendukung atau tidak. Setiap versi kernel memiliki fitur tersendiri yang bisa dipilih sesuai dengan kebutuhan sistem. Kernel yang akan digunakan perlu dilakukan konfigurasi untuk memilih jenis fitur yang akan diaktifkan, tidak mesti semua fitur yang ada harus diaktifkan karena hal ini akan berpengaruh terhadap kapasitas file *image* dari kernel yang dihasilkan.

c. Membangun *root filesystem*

   Root filesystem pada *embedded linux* sama halnya seperti pada linux *workstation*. Akan tetapi pada *embedded linux, root filesystem* berisi beberapa aplikasi, *library* dan beberapa *file* yang dibutuhkan sistem dengan kondisi minimal. Semakin variatif aplikasi yang dikembangkan untuk target, maka hal ini akan memperbesar ukuran *root filesystem*.

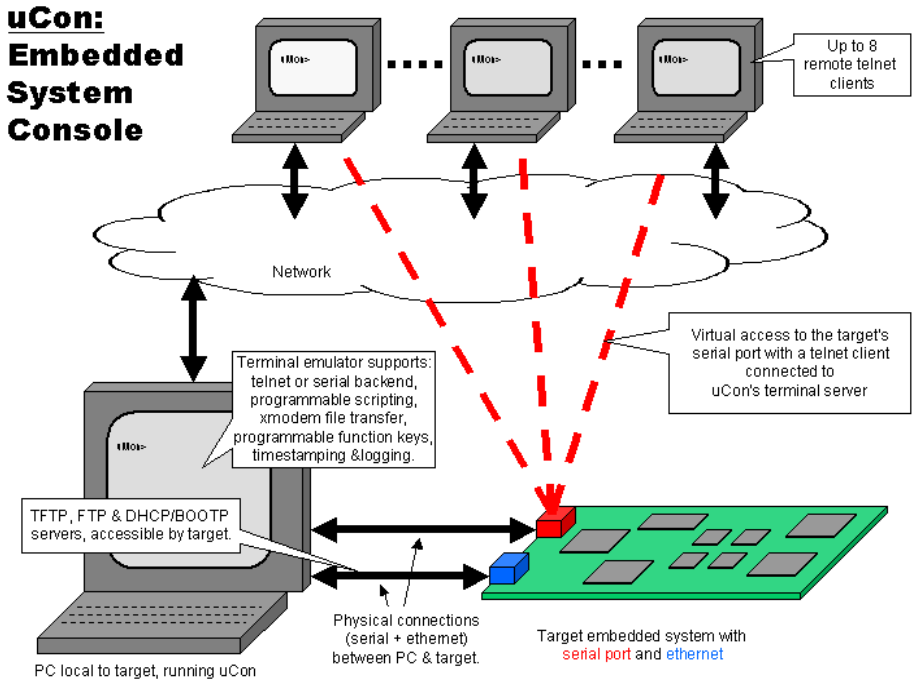
d. Melakukan instalasi *bootup software* dan konfigurasinya

Bootloader software merupakan bagian terpenting dari sistem *embedded* yang bertugas untuk menjalankan sistem pertama kali sebelum *file* sistem operasi *embedded* dieksekusi. *Bootloader* yang digunakan menyesuaikan dengan arsitektur yang digunakan dalam sistem. Proses *booting* adakalanya melalui *storage device, disk on chip* bahkan melalui jaringan [6].

D. Ucon (*The Embedded System Console)*

Ucon adalah *tools* yang didesain untuk pengembangan *firmware* dari sistem *embedded* tetapi tidak hanya terbatas pada hal tersebut. Ucon memiliki kemampuan sebagai *terminal emulator*, terminal *server* dan beberapa jenis *server* yang dapat digunakan untuk pengembangan sistem *embedded*. Motivasi utama dalam pengembangan Ucon adalah untuk menyediakan akses *multi-user* secara *remote* kepada sebuah perangkat komputer melalui *port* komunikasi serial dan tidak menghilangkan kemampuan *local-user* untuk menjalankan Ucon untuk mengakses *port* yang sama [18].

Ucon berjalan pada PC *host* melalui *port* komunikasi serial (COM) dengan berbasiskan CLI (*Command Line Interface*) untuk beberapa target sistem dan terlihat seperti program terminal emulasi yang sederhana. [18]



Gambar 9. Ucon *Embedded System Console*

E. GNU *Cross-Platform Development Toolchain*

Toolchain diperlukan sebagai aplikasi *cross-develop* untuk berbagai target arsitektur yang mencakup *binary utilities* seperti ld, gas dan ar, C *compiler* gcc dan C *library* glibc. Langkah awal dalam membangun toolchain adalah memilih versi dari komponen-komponen yang akan digunakan. Dalam hal ini adalah memilih versi dari binutils, gcc dan glibc. Karena paket-paket yang telah dirilis bersifat independen satu sama lain, tidak semua versi dari paket yang ada akan bisa dikombinasikan dengan versi yang berbeda dengan paket lainnya. Penggunaan versi terbaru juga tidak akan menjamin bahwa kombinasi tersebut akan bekerja dengan baik tanpa masalah [6].

Setidaknya ada lima langkah utama dalam membangun *toolchain* yaitu :

1. Instalasi *kernel headers*,
2. Instalasi *binary utilities*,
3. Instalasi *bootstrap compiler*,
4. Instalasi C *library*,
5. Instalasi *full compiler*.

Kernel merupakan *central software* untuk semua sistem linux, kemampuan ini akan berpengaruh terhadap kemampuan seluruh sistem. Jika kernel yang digunakan gagal untuk mendukung salah satu perangkat keras dari target, maka perangkat tersebut tidak akan dapat digunakan. Instalasi *kernel header* ini akan menghasilkan *file-file* yang mendefinisikan kemampuan sistem sehingga perlu dilakukan pada tahap awal [6].

Binary utilities (*binutils*) adalah paket komponen *toolchain* yang digunakan untuk memanipulasi *file binary*. Dua *utilities* paling penting yang harus ada dalam paket komponen ini adalah GNU *assembler* as dan *linker* ld [6].

Berbeda dengan paket *binutils*, paket komponen gcc hanya berisi satu *utility* yaitu GNU *compiler*, berhubungan dengan komponen pendukung seperti *runtime libraries*. *Bootstrap compiler* merupakan *compiler* yang hanya akan mendukung untuk kompilasi aplikasi yang ditulis dalam bahasa C. Akan tetapi setelah C *library* terinstal pada sistem, maka selanjutnya dapat dilakukan kompilasi ulang gcc dengan dukungan C++ secara penuh [6].

F. *Crosstool*

Crosstool dikembangkan oleh seorang *software engineer* bernama Dan Kegel. *Crosstool* merupakan kumpulan *script* yang digunakan untuk membangun dan melakukan tes terhadap beberapa versi gcc dan glibc untuk berbagai arsitektur yang didukung oleh glibc. *Crosstool* dikembangkan untuk para pengembang sistem *embedded*, tetapi juga sangat membantu para pengembang yang ingin melakukan kompilasi secara cepat atau yang ingin membangun program yang akan berjalan pada versi linux yang telah lama tetapi tidak ingin mengembangkan pada sistem yang kuno tersebut [10].

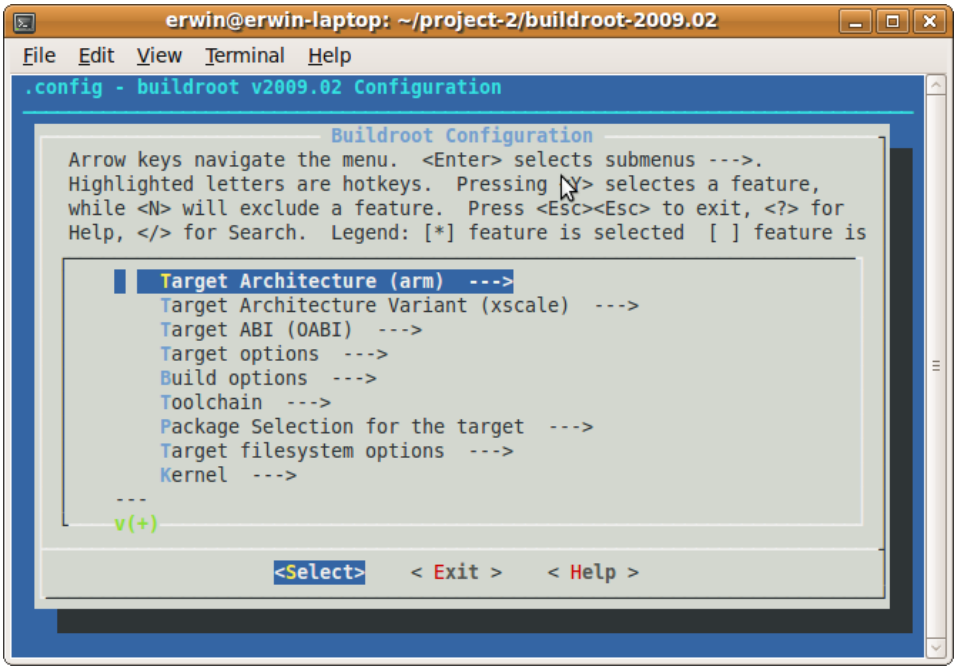
Crosstool memuat minimal *patches* untuk gcc dan glibc yang diperlukan untuk membangun beberapa kombinasi sepeti (alpha, arm, i686, ia64, mips, powerpc, powerpc64, sh4, sparc, sparc64, s390, x86_64) x (gcc-2.95.3 ... gcc-4.0.0) x (glibc-2.1.3 ... glibc-2.3.5). *Crosstool* merupakan *script* yang bersifat *portable*. *Compiler* yang dibangun untuk pengembangan linux pada sistem target dapat berjalan pada Linux, Mac OS X, Solaris dan Cygwin [10].

G. *BuildRoot*

Buildroot merupakan kumpulan *Makefiles* dan *patches* yang menyediakan fasilitas untuk memudahkan dalam membangun *cross-compilation toolchain*, *root filesystem* dan *linux kernel image* untuk target. *Buildroot* dapat digunakan sendiri maupun secara berkelompok karena bersifat independen [11].

Buildroot sangat cocok digunakan untuk orang-orang yang bekerja dengan *embedded system*. *Embedded system* seringkali menggunakan prosesor yang tidak familiar digunakan seperti x86 pada PC. Prosesor yang biasanya digunakan adalah PowerPC, MIPS, ARM dan sebagainya [11].

Buildroot secara otomatis menangani permasalahan instalasi *toolchain*, *compiler*, *binary utilities* dan *library* dengan memberikan kombinasi yang tepat untuk semua paket tersebut sehingga dapat berjalan dengan baik pada berbagai jenis arsitektur. Lebih dari itu, *buildroot* juga menyediakan fasilitas untuk mengulang proses pembangunan kernel, *cross toolchain* dan *embedded root filesystem*. Proses tersebut diperlukan ketika ada komponen yang perlu untuk dilakukan patch maupun update atau ketika orang lain mengambil alih *project* yang sedang dikerjakan [11].



Gambar 10. *BuildRoot*