Perancangan Sistem Embedded Linux Berbasis ARM

Andika Gunawan [1)], Henry Hermawan [2)]
Teknik Elektro Universitas Surabaya [1,2)]
ndikz91@gmail.com [1)]
henryhermawan@staff.ubaya.ac.id [2)]

Abstrak – Perancangan sistem *embedded* Linux ini bertujuan mempelajari *embedded system* dengan membuat *single board computer* AT91RM9200A berbasis mikrokontroler ARM AT91RM9200. *Board* ini bersifat *general purpose* sehingga diharapkan dapat diaplikasikan dalam berbagai macam fungsi yang bermanfaat. Pembuatan *board* ini dilatarbelakangi oleh perkembangan dalam bidang elektronika di mana kini semakin banyak penggunaan *embedded system* untuk berbagai macam keperluan. Pembelajaran yang telah dilakukan meliputi pembuatan *hardware* berupa rangkaian *single board computer* dan rangkaian pendukungnya, sedangkan *software* meliputi konfigurasi *bootloader*, *kernel* dan *root filesystem* untuk *board* AT91RM9200A beserta *cross compiler* sebagai *compiler software*. Hasil akhir perancangan berupa sebuah *embedded system* yang telah diuji hingga tahap implementasi *kernel* pada *board* AT91RM9200A tanpa *hardware* EMAC. Dengan memanfaatkan *board* ini sebagai *board prototype* untuk pembelajaran sistem dan uji coba aplikasi maka proses pembuatan *embedded system* dapat dipercepat.
Kata kunci : sistem *embedded* Linux, *embedded system*, *single board computer*, ARM, AT91RM9200*, prototyping board*.

Abstract – *This Linux embedded system design aims to study embedded system by building a single board computer AT91RM9200A based on ARM microcontroller AT91RM9200. This board is a general purpose board so that it can be used on many useful application. This project was prompted by the development on electrical fields which direction is on using more embedded system for any needs at this moment. The purpose of this project is to learn about building single board computer and its peripheral circuit and to learn about porting bootloader, Linux kernel and root filesystem so that it can be used on AT91RM9200A board with an appropriate cross compiler. The result is an embedded system which have been tested to the stage of implementing kernel to AT91RM9200A board without EMAC hardware. By using this board as a prototyping board for learning and testing an application, the development process of an embedded system can be accelerated.*
Keywords**:** : Linux *embedded system*, *embedded systems*, *single board computer*, ARM, AT91RM9200*, prototyping board*.

PENDAHULUAN

Kebutuhan manusia akan *embedded system* semakin meningkat seiring perkembangan teknologi. Meningkatnya kebutuhan ini harus diiringi dengan peningkatan kemampuan sumber daya manusia untuk memahami dan mengembangkan teknologi ini. Untuk memenuhi kebutuhan tersebut maka muncul sebuah gagasan untuk mempelajari *embedded* system dengan membuat *portable single board computer* yang berfungsi sebagai *prototyping board* untuk digunakan sebagai *board* pembelajaran. *Single board computer* yang sudah ada memiliki berbagai macam variasi sesuai dengan tujuan pembuatan *board* tersebut, sebagai contoh terdapat beberapa proyek pembuatan *board embedded* Linux yang digunakan sebagai referensi untuk desain ini, umumnya *board embedded* Linux yang ada sudah memiliki fungsi khusus sehingga kurang sesuai untuk digunakan dengan tujuan pembelajaran.

Tujuan utama proyek ini adalah untuk mempelajari *embedded system* dengan membuat *single board computer* dengan fitur minimal supaya *board* dapat ditanamkan OS *custom* Linux. Pembelajaran ini dilakukan untuk mendalami ilmu mengenai *embedded system* dengan harapan *single board computer* yang dirancang dapat digunakan sebagai media belajar pada perkuliahan *embedded system* pada Teknik Elektro Universitas Surabaya.

Pembuatan *single board computer* difokuskan pada pembuatan *hardware* dengan mikrokontroler ARM. Pengerjaan proyek ini dimulai dengan mempelajari 3 desain referensi sebagai referensi dalam mendesain *board* AT91RM9200A. Pada proyek ini digunakan 3 desain untuk referensi, masing-masing desain tersebut memiliki fokus desain tersendiri sehingga setiap desain yang ada berbeda dengan desain lainnya. Tiga desain referensi untuk proyek ini :

1. ECB AT91 V1 untuk aplikasi umum dengan memori kecil, selanjutnya akan disebut desain A.[1]
2. Darrell Harmon's *Single Board Computer* [2] untuk aplikasi dengan FPGA dan memori besar, selanjutnya akan disebut desain B.
3. *Open* ARM9 SBC [3] untuk aplikasi dengan sensor CMOS dan FPGA, selanjutya akan disebut desain C.

Board A merupakan *board* sederhana tanpa FPGA yang mengutamakan fitur IC ARM AT91RM9200. Fitur yang tersedia cukup banyak namun masih belum seluruh fitur yang tersedia pada IC AT91RM9200, *board* ini digunakan pada bidang robotika dan masih dapat digunakan untuk aplikasi umum.

Board B menggunakan IC FPGA dan IC ARM sekaligus. Dari desain yang dibuat, *board* ini cocok untuk aplikasi yang membutuhkan ukuran data besar karena memori yang disediakan cukup banyak. Pada desain ini konektor untuk expansion lebih banyak terpasang pada FPGA sehingga dapat diketahui bahwa desain B lebih memfokuskan pada fleksibilitas penggunaan FPGA tersebut.

Board C memiliki IC FPGA dan IC ARM seperti referensi B, perbedaannya ada pada ukuran memori yang lebih kecil dan adanya komponen CMOS sensor sehingga penggunaannya lebih terbatas pada aplikasi dengan sensor tersebut.

Ketiga desain referensi ini menggunakan IC AT91RM9200, IC ini sudah dikenal luas sebagai *entry point* pembelajaran sistem *embedded* Linux sehingga sudah banyak referensi dan *development tools* yang tersedia. *Board* ini dibuat dengan tujuan pembelajaran sehingga AT91RM9200A dibuat sebagai *board general purpose* dengan menghubungkan seluruh *pin* I/O dari IC AT91RM9200 pada *pin header* untuk modifikasi lebih lanjut.

Tahap berikutnya adalah merancang *hardware* PCB yang akan dibuat dengan membandingkan fitur dari referensi dan kemudian memilih komponen yang digunakan. Setelah PCB direalisasikan maka langkah berikutnya adalah *debugging*. Proses *debugging board* AT91RM9200A dapat dilakukan menggunakan fitur JTAG[4] dan ICE atau menggunakan *debug port* yang tersedia pada *cross development environment*[4].

Setelah komponen dipilih maka tahap berikutnya adalah pembuatan PCB dan mem-*porting software* yang dibutuhkan seperti *bootloader, kernel* Linux, *root filesystem* [4] [5] dan mem-*build cross compiler* pada *development host*.

PERANCANGAN *SINGLE BOARD COMPUTER* AT91RM9200A

Perancangan *single board computer* AT91RM9200A dilakukan secara bertahap diawali dari pembelajaran pustaka kemudian pembuatan *hardware* dan

konfigurasi *software*. Data pembelajaran pustaka yang digunakan berupa data pustaka untuk *single board computer* dengan rangkaian yang berbeda sehingga perancangan dilakukan dengan mempelajari ketiga desain tersebut dan kemudian merancang *single board computer* AT91RM9200A.

Pembelajaran ketiga desain tersebut diawali dengan menganalisa rangkaian ketiga desain tersebut dan dilanjutkan dengan pembuatan rangkaian *single board computer* AT91RM9200A, pemilihan komponen dan realisasi desain dengan mencetak rangkaian pada PCB. Dari analisa rangkaian ketiga desain referensi diperoleh perbandingan fitur yang digunakan, hasil perbandingan fitur dapat dilihat pada Tabel 1 sedangkan perbandingan komponen ketiga desain referensi dapat dilihat pada Tabel 2. Pada kedua tabel tersebut juga dapat dilihat fitur dan komponen yang digunakan untuk *board* AT91RM9200A yang dibuat.

Tabel 1 menunjukkan fitur-fitur yang digunakan pada 3 desain referensi dan board yang dibuat. *Board* yang dibuat adalah *general purpose board* sederhana sehingga tidak menggunakan FPGA dan memiliki banyak fitur yang masih dapat digunakan karena pin I/O IC AT91RM9200 pada board ini terhubung dengan *header* sehingga dapat diberi rangkaian tambahan untuk modifikasi *board*.

Tabel 1 : Tabel perbandingan fitur 3 desain referensi dan desain AT91RM9200A

Desain referensi	A	B	C	Board
On-board FPGA	×	✓	✓	×
Oscillator	2	2	2	2
Phase Locked Loop (PLL)	2	2	2	2
Supervisory Circuit	✓	✓	✓	✓
External Bus Interface	SDRAM (16MB)	SDRAM (32MB) NAND Flash (64MB) Compact Flash	SDRAM (32MB)	SDRAM (32MB)
USB Host User Interface	✓	✓	✓	✓
Multimedia Card Interface	✓	×	×	×
Two-wire Interface	✓	×	×	✓
Ethernet MAC	✓	✓	✓	✓
USART	1	1	1	1
SPI	3	3	3	3
Flash Memory on SPI0	✓ (2MB)	✓ (2MB)	✓ (4MB)	✓ (4MB)
Debug Port	✓	✓	✓	✓
JTAG Port	✓	×	×	✓
Expansion Port (I/O)	×	×	×	✓
Expansion Port (Memory)	✓	✓	×	✓
Buffer on Expansion Port	×	✓	×	✓
Board Size	8.5cm x 7.7cm	10cm x 10cm	16.5cm x 7.5cm	11.5cm x 10.3cm
Board Layer	2 layer	4 layer	2 layer	2 layer
CMOS Sensor	×	×	✓	×

Tabel 2 menunjukkan perbandingan komponen yang digunakan, kebanyakan fitur *board* AT91RM9200A yang digunakan hanya memiliki *header port* sehingga untuk menggunakannya diperlukan rangkaian tambahan. Komponen-komponen ini dipilih berdasarkan kompatibilitas, ketersediaan komponen, ketersediaan *development kit* komponen dan juga kejelasan dokumen atau *datasheet* dari komponen. Secara umum komponen yang digunakan pada *board* yang dibuat sudah pernah digunakan pada desain *board* lainnya sehingga dapat dipastikan bahwa komponen-komponen ini dapat digunakan dengan baik.

Setelah menentukan fitur dan komponen yang digunakan maka desain sudah dapat dibuat dan kemudian direalisasikan dalam sebuah PCB. Dari Tabel 1 fitur yang digunakan dapat dianalisa pengalamatan memori yang dapat digunakan. *Memory mapping* ketiga desain referensi dan *board* AT91RM9200A dapat dilihat pada Tabel 3, 4, 5 dan 6.

Tabel 2 : Tabel perbandingan komponen 3 desain referensi dan desain AT91RM9200A

Desain Referensi	A	B	C	Board
Supervisory	MCP130	LP3470	LP3470	MCP130
SDRAM	MT48LC8M16A2	HYB39S256	MT48LC16M16A2	MT48LC16M16A2
NAND Flash	-	TC58DVM92A1FT00	-	-
Compact Flash	-	Port	-	-
USB Host Filter	USBDF01W5	RC Circuit	USBDF01W5	RC Circuit
Multimedia Card Interface	Port	-	-	-
Two-Wire Interface	Header port	-	-	Header port
Ethernet MAC	KS8721BL	DP83846A	LXT972A	KS8721BL
USART	MAX3223ECAP	MAX3232	MAX3232	FT232RL
SPI	Header port	Header port	Header port	Header port
Flash Memory on SPI0	AT45DB161	AT45DB161B	AT45DB321C	AT45DB321C
Debug Port	Header port	Header port	Header port	Header port
JTAG Port	Header port	-	-	Header port
Expansion Port (Memory)	Header port	Buffer	-	Buffer
Buffer on Expansion Port	-	74ALVCH16245	-	74ALVCH16245
CMOS Sensor	-	-	Micron MT9T001	-

Tabel 3 : *Memory mapping* desain A

Start Address	End Address	Used For	Size
0x0000 0000	0x0FFF FFFF	Internal Memories	256MB
0x2000 0000	0x2FFF FFFF	SDRAMC	256MB
0xC000 0000	0xCFFF FFFF	Dataflash on SPI0	256MB
0xFFFB 4000	0xFFFB 8000	MCI	16KB
0xFFFB 8000	0xFFFB C000	TWI	16KB
0xFFFB C000	0xFFFC 0000	EMAC	16KB
0xFFFC 0000	0xFFFC 4000	USART0	16KB
0xFFFE 4000	0xFFFF FFFF	System Peripheral Mapping	112KB

Tabel 4 : *Memory mapping* desain B

Start Address	End Address	Used For	Size
0x0000 0000	0x0FFF FFFF	Internal Memories	256MB
0x2000 0000	0x2FFF FFFF	SDRAMC	256MB
0x4000 0000	0x4FFF FFFF	NAND Flash	256MB
0x5000 0000	0x5FFF FFFF	Compact Flash	256MB
0xC000 0000	0xCFFF FFFF	Dataflash on SPI0	256MB
0xFFFB C000	0xFFFC 0000	EMAC	16KB
0xFFFC 0000	0xFFFC 4000	USART0	16KB
0xFFFC 8000	0xFFFC C000	USART2	16KB
0xFFFE 0000	0xFFFE 4000	SPI	16KB
0xFFFE 4000	0xFFFF FFFF	System Peripheral Mapping	112KB

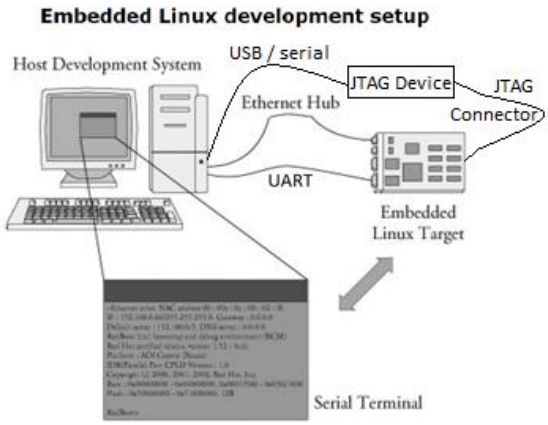
Tabel 5 : *Memory mapping* desain C

Start Address	End Address	Used For	Size
0x0000 0000	0x0FFF FFFF	Internal Memories	256MB
0x2000 0000	0x2FFF FFFF	SDRAMC	256MB
0xC000 0000	0xCFFF FFFF	Dataflash on SPI0	256MB
0xFFFB 8000	0xFFFB C000	TWI	16KB
0xFFFB C000	0xFFFC 0000	EMAC	16KB
0xFFFC 0000	0xFFFC 4000	USART0	16KB
0xFFFE 4000	0xFFFF FFFF	System Peripheral Mapping	112KB

Tabel 6 : *Memory mapping* desain AT91RM9200A

Start Address	End Address	Used For	Size
0x0000 0000	0x0FFF FFFF	Internal Memories	256MB
0x2000 0000	0x2FFF FFFF	EBI (Chip Select 1 / SDRAMC)	256MB
0xC000 0000	0xCFFF FFFF	Dataflash on SPI0	256MB
0xFFFB 8000	0xFFFB C000	TWI	16KB
0xFFFB C000	0xFFFC 0000	EMAC	16KB
0xFFFC 0000	0xFFFC 4000	USART0	16KB
0xFFFE 0000	0xFFFE 4000	SPI	16KB
0xFFFE 4000	0xFFFF FFFF	System Peripheral Mapping	112KB

Setelah PCB sudah dibuat dan komponen terpasang maka untuk mengimplementasikan *software* pada *board* AT91RM9200A *board* harus dihubungkan dengan *development host* Linux menggunakan *environment* seperti pada Gambar 1. Untuk melakukan *debugging* awal maka JTAG *device* dan *ethernet hub* tidak digunakan.



Gambar 1 : *Cross development environment*.

Langkah berikutnya adalah implementasi *software*, *software* yang pertama kali di-*build* pada *development host* adalah *cross compiler*, instalasi 2 macam *cross compiler* ini berbeda-beda, *cross compiler* `arm-none-eabi` di-*build* menggunakan *script* secara manual sedangkan *cross compiler* `arm-linux` di-*build* menggunakan Buildroot versi 2011.11[6].

Setelah kedua *cross compiler* selesai di-*build* dan variabel `$PATH` telah diedit maka langkah berikutnya adalah *build bootloader*, untuk menyimpan *bootloader* pada *dataflash* diperlukan beberapa langkah yang dimulai dengan mengaktifkan ROM *boot* untuk mengupload *file* pada *internal* SRAM. Karena *internal* SRAM hanya berukuran 16Kbyte sedangkan *file bootloader* U-boot berukuran sekitar 600Kbyte maka dibutuhkan *file preloader* yang dapat disimpan pada *internal* SRAM dan dapat dijalankan untuk menerima *file* melalui protokol XMODEM dan menyimpan *file* tersebut langsung pada *dataflash*. Karena ukuran *internal* SRAM ini maka untuk menyimpan U-boot dan *software* lainnya pada *dataflash* dibutuhkan 3 macam *bootloader*, yang pertama adalah ROM *boot* yang tersedia pada *internal* ROM, yang kedua adalah *preloader* untuk menyimpan *file* langsung pada *dataflash* dan yang ketiga adalah U-boot sebagai *bootloader* akhir.

Setelah *preloader* tersimpan pada *dataflash* maka *file* U-boot, *kernel* dan *root filesystem* dapat disimpan pada *dataflash* juga. Program *preloader* yang digunakan adalah *preloader* dari desain B yang dibuat oleh Darrell Harmon yang

telah diedit [7] kemudian diedit lagi untuk disesuaikan dengan *board* AT91RM9200A.

U-boot yang digunakan pada proyek ini adalah U-boot versi 2010.06[8]. U-boot digunakan untuk mengatur *environment variable,* melakukan inisialisasi awal dan kemudian menjalankan *kernel*. Setelah *kernel* dijalankan maka *kernel* akan mencari *root filesystem* yang kemudian akan digunakan sebagai *directory root* atau / pada OS Linux yang dijalankan. *Kernel* adalah bagian *software* yang mampu mengatur penggunaan *hardware*, *memory device* dan *scheduling* penggunaan prosesor sehingga pengguna tidak perlu mengetahui apa sebenarnya kerja *board* pada layer abstraksi ketika pengguna menggunakan aplikasi pada *user space context*.

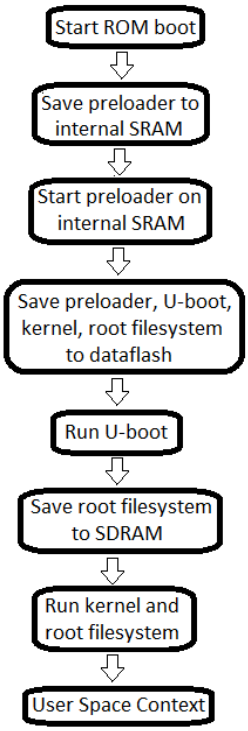
Kernel Linux yang digunakan adalah *kernel* versi 2.6.38[9] dengan *patch* dari Maxim[10] sedangkan *root filesystem* yang digunakan telah dicoba menggunakan 2 macam *filesystem* yaitu *ramdisk* dan *ramfs*[11]. Pada *software-software* ini terutama pada *kernel* dan U-boot dilakukan penambahan dan pengaturan beberapa variabel penting seperti :

- Pembagian *memory* :
  - *Bootstrap* : 0x00000000 – 0x000041FF
  - *Enviroment variable* : 0x00004200 – 0x000083FF
  - U-boot : 0x00008400 – 0x00041FFF
  - *Kernel* : 0x00042000 – 0x00251FFF
  - *Filesystem* : 0x00252000 – 0x0041FFFF
- Penambahan nomor registrasi *board* AT91RM9200A dengan `MACH_TYPE`=8888
- Aktivasi fitur SPI, TWI, I2C dan USART dengan menambahkan kode program pada file konfigurasi U-boot dan mengubah *setting* konfigurasi *kernel*.

Setting software pada AT91RM9200A dilakukan dengan *boot sequence* seperti pada Gambar 2. Setelah seluruh *software* telah dikonfigurasi maka langkah-langkah pada gambar tersebut digunakan untuk menyimpan seluruh *file*

software tersebut pada *dataflash* dan kemudian menjalankan system dengan OS Linux telah terinstal.

Langkah pertama seperti yang telah dijelaskan, mengaktifkan ROM *boot* kemudian *upload preloader* pada *internal* SRAM dan setelah *preloader* dijalankan maka *file preloader*, U-boot, *kernel* dan *root filesystem* dapat disimpan dalam *dataflash*. Setelah file *software* tersimpan pada *dataflash*, untuk menjalankan *kernel* maka *root filesystem* yang digunakan harus disimpan dalam SDRAM karena *root filesystem* yang digunakan berupa *compressed file* dan ketika digunakan membutuhkan ukuran memori sekitar 4MByte sehingga jika alamat memori *root filesystem* yang digunakan menunjuk pada alamat *dataflash* maka *root filesystem* tidak akan dapat digunakan. Untuk itu *root filesystem* harus disimpan pada SDRAM menggunakan *command copy* dari U-boot. Setelah *root filesystem* tersimpan pada *dataflash* maka *kernel* dapat dieksekusi pada *board* dan kemudian pengguna dapat menggunakan aplikasi pada *user space context* dengan *filesystem* sebagai *directory root* atau /.



Gambar 2 : *Boot sequence* AT91RM9200A

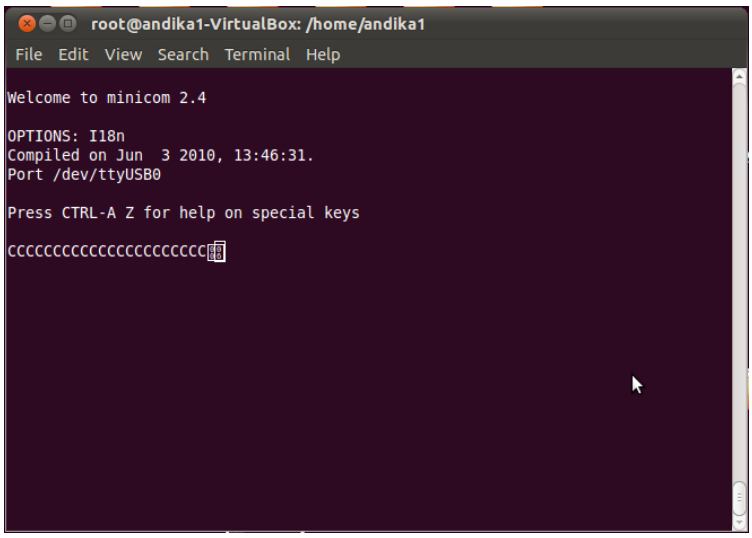
PENGUJIAN

Pengujian untuk *board* AT91RM9200A dilakukan dengan menggunakan *board* hingga pada tahap implementasi *kernel* yang mana karena *ramdisk* atau *ramfs* yang dibuat masih mengalami *error* maka tahap akhir yang dapat dikerjakan ada pada tahap konfigurasi *kernel* dan *root filesystem*. Pengujian yang dilakukan adalah pengujian untuk *hardware* dan *software* yang telah dibuat, pengujian *hardware* dilakukan beriringan dengan pengujian *software* karena jika *hardware* masih mengalami masalah maka *software* yang telah diimplementasikan juga akan bermasalah sehingga pengecekan *hardware* dapat dilihat pada kerja *software* pada *board*.

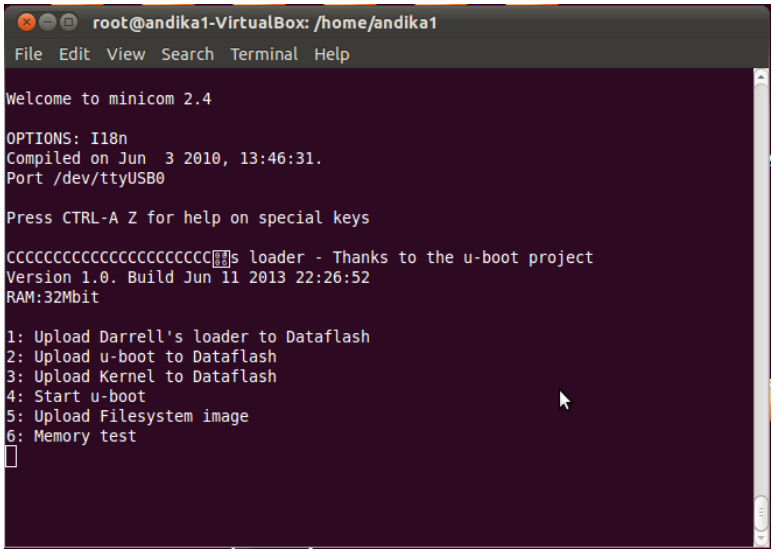
Pengujian dimulai dengan menghubungkan *board* AT91RM9200A sesuai dengan *development environment* untuk *debugging* awal seperti pada Gambar 3 yang mana dapat terlihat *board* terhubung dengan *development host* menggunakan konektor USB saja. *Board* dinyalakan dengan kondisi *flash disabled* supaya ketika menyala *board* AT91RM9200A tidak mendeteksi adanya *flash memory* dan bersiap untuk menerima *file preloader* dari *development host*.

Gambar 3 : *Development environment board* AT91RM9200A

Gambar 4 menunjukkan kondisi *board* baru dinyalakan dan tampilan pada minicom yang menampilkan huruf CCC menandakan *board* sudah siap menerima *file* untuk disimpan pada *internal* SRAM AT91RM9200. Gambar ini menunjukkan tampilan minicom ketika *board* dinyalakan, tampilnya karakter CCC ini menunjukkan bahwa *clock system* dan *debug port* yang terhubung dengan FT232RL sudah bekerja dengan benar. Setelah ini pengujian dilanjutkan dengan meng-*upload preloader* dari *development host*.



Gambar 4 : Tampilan minicom ketika *board* dinyalakan.

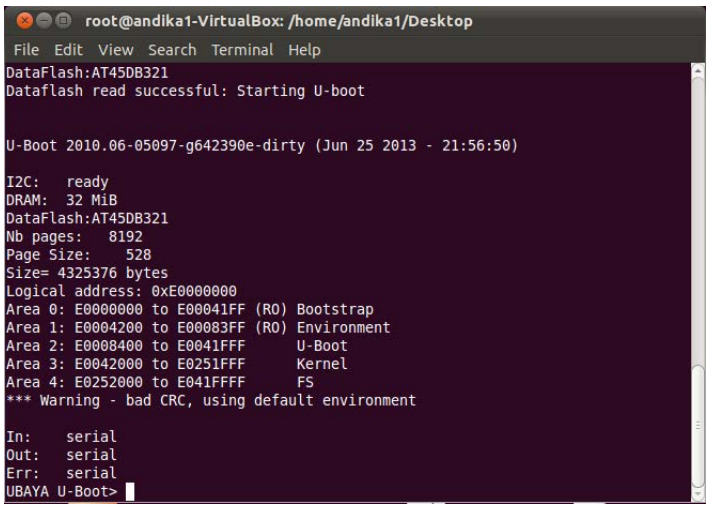


Gambar 5 : *Menu preloader* setelah di-*upload*

Setelah proses *upload* selesai maka *preloader* akan tersimpan pada *internal* SRAM IC AT91RM9200 dan kemudian akan langsung dijalankan seperti pada Gambar 5. Pada gambar tersebut terlihat *menu preloader* untuk meng-*upload preloader*, *U-boot*, *kernel* dan *filesystem* pada *dataflash*. Jika *preloader* sudah berhasil di-*upload* maka dapat diartikan bahwa rangkaian *supervisory*, *regulator* dan *power supply* yang digunakan dapat bekerja dengan baik. Ketika proses *upload* sering dijumpai permasalahan tegangan yang tidak stabil sehingga mengganggu kerja *board*, jika tidak terjadi *reset* atau kegagalan proses *upload* maka dapat diartikan ketiga komponen tersebut bekerja dengan baik. Selanjutnya untuk menyimpan *file preloader* dalam *dataflash* maka *header dataflash* harus diubah menjadi *dataflash enabled* seperti pada.

Setelah *flash* di-*enable*-kan langkah selanjutnya adalah *upload preloader* pada *dataflash* dengan *menu* nomor 1 *preloader*. Setelah proses *upload* selesai, akan tampil *menu preloader* sama seperti pada Gambar 5, sama seperti ketika *preloader* tersimpan pada *internal* SRAM. Pengujian *dataflash* dinilai berhasil jika proses *upload* berhasil dan *menu preloader* tampil pada layar *minicom*.

Pengujian selanjutnya adalah dengan meng-*upload file* U-boot, *kernel* dan *filesystem* menggunakan *preloader,* setelah proses *upload* berhasil maka berikutnya adalah menggunakan *menu* nomor 4 dari *preloader* untuk menjalankan U-boot seperti pada Gambar 6.

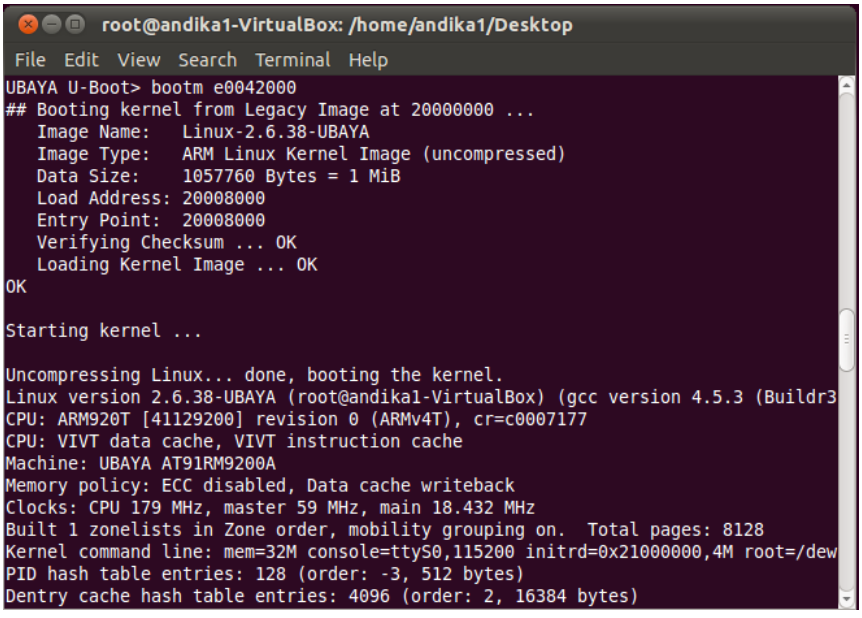


Gambar 6 : Tampilan *menu* U-boot

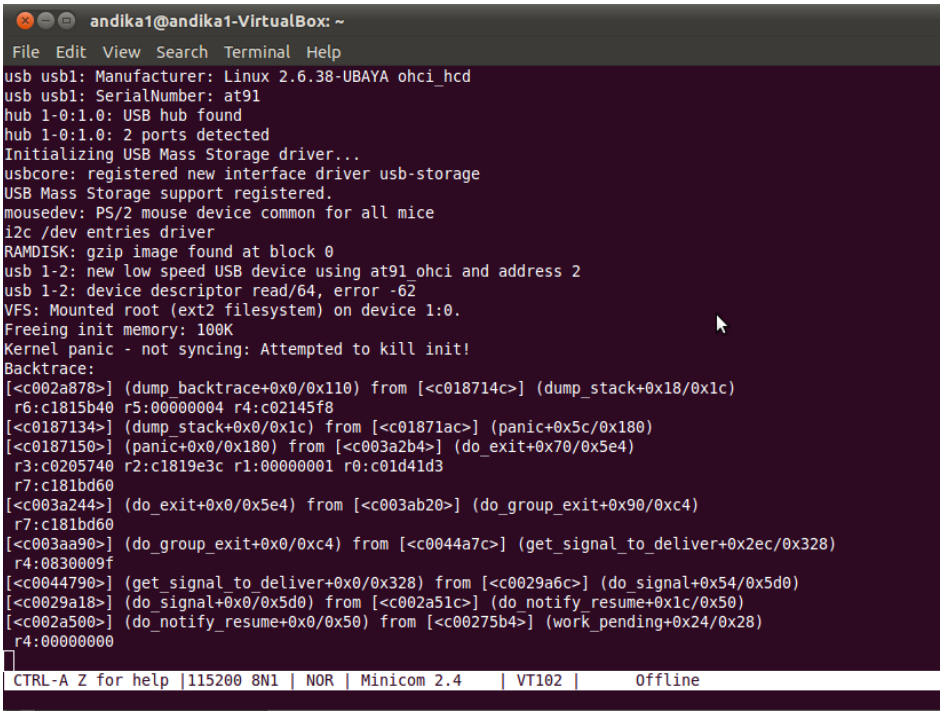
Tampilan *menu* U-boot membuktikan bahwa kerja *preloader* sudah benar, dapat dilihat pada tampilan *menu* U-boot pada Gambar 6 bahwa SDRAM dan *dataflash* telah terdeteksi, ini membuktikan bahwa *hardware dataflash* dan SDRAM dan juga *software preloader* dan U-boot berhasil diimplementasikan.

Pengujian selanjutnya adalah menggunakan *command* U-boot untuk menjalankan *kernel*. Seperti yang telah dijelaskan mengenai *kernel*, *kernel* membutuhkan sebuah *root filesystem* sebagai *directory* kerja *kernel* sedangkan *setting* yang disimpan konfigurasi *kernel* yang digunakan adalah alamat *filesystem* (*ramdisk*) ada pada lokasi memori 0x21000000 sehingga untuk menggunakan *ramdisk* ini pertama-tama dimulai dengan meng-*copy filesystem* dari *dataflash* pada SDRAM dengan alamat 0x21000000 dan kemudian menggunakan *command boot* untuk mengeksekusi *kernel* pada alamat memori e0042000.

Pada Gambar 7 dapat dilihat *command* U-boot untuk melakukan *boot kernel* pada alamat e0042000 (alamat *dataflash*), dari gambar tersebut dapat dilihat bahwa *image kernel* dapat dibaca, ini membuktikan bahwa konfigurasi *kernel* Linux sudah benar. Proses eksekusi *kernel* ini diikuti dengan *error message kernel panic* seperti pada Gambar 8. Kedua gambar ini menunjukkan kerja *kernel* dengan konfigurasi untuk penggunaan *ramdisk*.



Gambar 7 : *Command boot* untuk mengeksekusi *kernel* (*ramdisk*)



Gambar 8 : *Kernel panic* (*ramdisk*)

Untuk menghindari penggunaan *ramdisk* yang menurut konfigurasi dan cara pembuatan *ramdisk* yang sudah sesuai dengan referensi [11] namun masih mengalami *error* maka dibuat sebuah *ramfs* untuk menggantikan *ramdisk* ini. Untuk mencoba menggunakan *ramfs* maka konfigurasi pada *kernel* harus diubah dan kemudian di-*upload* ulang pada *dataflash* tanpa meng-*upload filesystem ramfs* yang dibuat karena *ramfs* telah di-*compile* bersama dengan *file kernel* sehingga *kernel* yang digunakan memiliki ukuran lebih besar disbanding ketika menggunakan *ramdisk* karena dalam *kernel* ini sudah terdapat *filesystem ramfs*. Setelah *kernel* di-*upload* dan kemudian dieksekusi, hasil yang diperoleh menggunakan *ramfs* tidak jauh berbeda dengan ketika menggunakan *ramdisk* yang mengalami *kernel panic*.

Hasil yang diperoleh menggunakan dua macam *filesystem* yang berbeda masih sama yaitu mengalami *error* dengan *error message* `Kernel panic - not syncing: attempted to kill init!` Sehingga hasil konfigurasi untuk menggunakan SPI, USART, I2C dan fitur lainnya masih belum dapat dibuktikan dan diuji.

Konfigurasi untuk beberapa fitur sudah diaktifkan, dapat dilihat pada Gambar 8 sudah terlihat *message* bahwa *driver* I2C telah di-*load* pada *kernel* yang dibuat. Sedangkan untuk fitur SPI dapat dilihat bahwa *device memory* yang digunakan untuk menyimpan seluruh *file bootloader, kernel* dan *filesystem* adalah *dataflash* yang diakses menggunakan komunikasi SPI dan d*ebug port* yang digunakan berkomunikasi menggunakan komunikasi UART. Fitur lain seperi *oscillator*, PLL dan memori (SDRAM) juga dapat digunakan dengan baik, SDRAM sudah dapat digunakan sebagai tempat penyimpanan memori dan *oscillator* dan PLL sudah dapat menghasilkan frekuensi *clock* yang benar sehingga tampilan pada *debug port* dan kerja prosesor yang menggunakan *clock* tersebut bekerja dengan baik.

Pengujian lain yang dilakukan adalah pengukuran daya yang digunakan *board* secara keseluruhan, pengukuran dilakukan dengan kondisi *board* menyala dan menjalankan program seperti *preloader* untuk *upload file* kemudian menjalankan U-boot untuk melakukan *command copy* dan *boot* kemudian menjalankan *kernel* meskipun masih mengalami *error*. Dari hasil pengukuran diperoleh bahwa nilai maksimal arus yang dibutuhkan ketika *board* bekerja adalah sekitar 208 mA dengan nilai tegangan dari *power supply* berkisar antara 11,2 V hingga 13,2 V. Hasil pengukuran ini menunjukkan bahwa *power supply* yang digunakan masih dalam batas normal sebagai tegangan *input regulator* dan *switching supply* yang digunakan untuk menghasilkan tegangan 1,8V; 5V dan 3,3V.

Hasil pengukuran menunjukkan arus yang digunakan *board* maksimal sebesar 208 mA. Jika *board* ini akan digunakan untuk dibawa-bawa (*portable*) maka dapat diperkirakan jika menggunakan baterai Li-Po 3 sel dengan daya *output* 1100 mAh maka *board* ini dapat digunakan sekitar 5 jam.

PENUTUP

Dari hasil perancangan *single board computer* AT91RM9200A diperoleh simpulan bahwa dari sisi *hardware*, hasil akhir *board* ini dapat bekerja sesuai spesifikasi awal hingga tahap implementasi *kernel* dengan pengecualian fitur

EMAC ditiadakan dan bagian *user space context* yang belum dapat diimplementasikan. *Hardware* sudah sesuai dengan spesifikasi yang dibutuhkan dan telah diuji sehingga kesalahan yang terjadi ada pada sisi *software* yang mana *software* telah dapat di-*compile* dan di-*build* dengan baik namun karena kendala pada bagian *software* maka banyak fitur dari sisi *software* yang tidak dapat dibuktikan dan hanya dapat diperlihatkan keberadaannya saja.

Tugas akhir ini banyak memberi pengetahuan yang baru mengenai perancangan *embedded system* berbasis ARM. Melalui perancangan *board* AT91RM9200A terdapat banyak ilmu berharga yang diperoleh, antara lain :

- Ketika melakukan *tracing* PCB lebih baik jika komponen yang berhubungan diletakkan berdekatan (sesuai dengan kebutuhan) sehingga lebih mudah untuk melakukan *debugging hardware*.
- Dari permasalahan yang dihadapi ketika *debugging hardware board*, salah satu kriteria pemilihan *regulator* atau *power supply* adalah kemampuan *supply* arus dan kestabilan yang dihasilkan, masalah ini dapat diatasi dengan menggunakan *regulator* atau *power supply* yang memiliki kemampuan *supply* arus melebihi kebutuhan sistem dan memiliki tingkat kestabilan yang bergantung pada kebutuhan sistem.
- Dalam proses desain *single board computer* harus diperhitungkan bagian sistem yang dapat digunakan untuk *debugging* seperti dengan memberi *header* tambahan supaya dapat menggunakan *logic analyzer* untuk *debugging*.
- Salah satu permasalahan besar yang dihadapi ketika pengerjaan *hardware* adalah PCB yang tidak *reliable*, PCB dengan desain yang rumit harus dibuat pada *manufacturer* yang memang mampu membuat PCB sesuai spesifikasi desain tersebut karena jika terjadi kesalahan maka pengecekan yang dapat dilakukan akan sangat sulit jika dilakukan secara manual.
- Masalah mengenai EMAC yang ditemui adalah kesulitan untuk melakukan pengecekan terhadap rangkaian EMAC sehingga dokumentasi ketika menggunakan EMAC sangat penting dilakukan karena pada tahap awal

EMAC belum terdeteksi dan tidak ada *interface* yang dapat membuktikan bahwa EMAC terpasang dan bekerja dengan baik.

- Untuk mengimplementasikan OS pada sebuah sistem dibutuhkan beberapa *software* seperti *bootloader*, *kernel* dan *filesystem* dan *toolchain* pada *development host*..
- Media *debug* yang digunakan pada tugas akhir ini menggunakan sebuah kabel USB saja, dengan menambahkan rangkaian FT232RL untuk mengkonversi komunikasi *debug port* UART menjadi USB maka cara *debugging board* menjadi lebih mudah dan dapat dilakukan menggunakan laptop yang tidak memiliki konektor UART.
- *Bootloader* digunakan sebagai *file* untuk inisialisasi awal pada implementasi OS, *bootloader* adalah program untuk mikrokontroler tanpa OS (*baremetal* OS) yang digunakan untuk *setting environment* awal sebelum mengeksekusi *kernel*.
- *Bootloader* yang digunakan pada proyek ini ada 3 yaitu ROM *boot* pada *internal* ROM IC AT91RM9200 lalu *preloader* dan U-boot yang dikonfigurasikan melalui *development host*. ROM *boot* digunakan untuk meng-*upload preloader* pada *internal* SRAM AT91RM9200, *preloader* digunakan untuk menyimpan program *preloader* itu sendiri pada *dataflash* beserta U-boot, *kernel* dan *filesystem* sedangkan U-boot digunakan untuk melakukan *boot kernel*.
- *Kernel* mengatur komunikasi antara *user space context* dengan *hardware*, mengatur *scheduling*, inisialisasi *hardware* dan mengatur *memory device*.
- Untuk membuat program yang dapat dijalankan pada *board* AT91RM9200A tanpa OS maka program yang dibuat pada *development host* harus di-*compile* dengan *cross compiler* `arm-none-eabi-gcc` sedangkan untuk *board* AT91RM9002A dengan OS maka *cross compiler* yang digunakan adalah `arm-linux-gcc`.
- Salah satu keuntungan menggunakan OS *open source* adalah OS tersebut mudah diperoleh dan dapat dengan mudah dikonfigurasi untuk banyak sistem, OS Linux yang digunakan pada tugas akhir ini memiliki *kernel*

yang mendukung penggunaan *kernel* Linux pada *board* dengan IC AT91RM9200 sehingga konfigurasi *kernel* menjadi lebih mudah.

- Proses *debugging embedded system* pada tahap awal sangat sulit karena banyak faktor kesalahan yang dapat terjadi seperti spesifikasi komponen kurang sesuai, PCB yang terlalu lama dipanaskan hingga kerusakan *hardware* yang tidak terlihat secara fisik seperti permasalahan fitur EMAC, *regulator* dan *power supply*.
- Proses *debugging software embedded system* dapat meliputi lingkup yang luas, lingkup luas ini dikarenakan penggunaan OS *open source* yang mana dapat dikembangkan secara bebas oleh individu sehingga banyak cara yang dapat dilakukan untuk mengimplementasikan OS tersebut.

Saran untuk pengembangan dan perbaikan pembuatan *single board computer* berbasis IC ARM antara lain :

- Referensi yang digunakan sebagai dasar pembuatan *board* sebaiknya menggunakan referensi yang dapat digunakan sebagai referensi *hardware* dan *software* sehingga proses pembuatan *board* akan lebih mudah sehingga pembuatan *software* sudah memiliki petunjuk pembuatan.
- IC yang digunakan untuk membuat *single board computer* lebih baik jika menggunakan IC baru karena IC baru umumnya memiliki *development tools* yang lebih mudah digunakan dan memiliki *community support* yang masih aktif seperti yang dibahas pada subbab 3.2 dan masalah *software* yang dihadapi pada subbab 3.8.
- Pembelian komponen untuk membuat *single board computer* sebaiknya berjumlah lebih banyak dari yang dibutuhkan. Kelebihan komponen ini digunakan sebagai komponen cadangan ketika ada komponen yang rusak.

DAFTAR PUSTAKA

[1] ECB AT91 Developer. (2011). *ecb-at91-v1.6_schematics* [pdf]. Tersedia: http://wiki.emqbit.com/free-ecb-at91 diakses 14 Maret 2012

[2] Darrell Harmon. (2004). *Darrell Harmon's Single Board Project* [Online]. Tersedia: http://dlharmon.com/sbc.html diakses 12 April 2012

[3] Flavio Ribeiro. (2005). *Open ARM9* SBC [Online]. Tersedia: http://www.lps.usp.br/~fr/sbc/ diakses 12 April 2012

[4] C. Hallinan, *Embedded Linux Primer : a practical, real-world approach*. Indiana: Prentice Hall, 2006.

[5] K. Yaghmour, *Building Embedded Linux Systems*. CA : O'Reilly & Associates, 2003.

[6] Buildroot developers (2013). *Buildroot: making Embedded Linux easy* [Online]. Tersedia: http://buildroot.uclibc.org/downloads/ diakses 23 Januari 2013

[7] Linuxstamp contributor (2011). *Need source to Darrell's Loader* [Online]. Tersedia: https://groups.google.com/forum/?fromgroups=#!msg/linuxstamp/eaIauV6GSfU/96_RxaDkhuAJ diakses 1 Mei 2013

[8] DENX Software Engineering (2012). *Das U-Boot – the Universal Boot Loader* [Online]. Tersedia: http://www.denx.de/wiki/U-Boot diakses 24 April 2013

[9] Linux Kernel Organization Inc. (2011). *Index of /pub/linux/kernel/v2.6* [Online]. Tersedia: https://www.*kernel*.org/pub/linux/*kernel*/v2.6/ diakses 16 Januari 2013

[10] Maxim.org contributor (2011). *AT91 Linux 2.6 Patches* [Online]. Tersedia: http://maxim.org.za/at91_26.html diakses 16 Januari 2013

[11] Texas Instruments Inc. (2010). Texas Instruments Wiki [Online]. Tersedia: http://processors.wiki.ti.com/index.php/Initrd diakses 19 Juni 2013