Perancangan *Embedded Linux Operating System* Dan *Upgrading Software* Untuk *Voice Terminal Intercom Naval System*

1st Luthfi Febriansyah
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
Luthfifebriansyah12@student.telkomuniversity.co.id

2nd Aris Hartaman
Fakultas Ilmu Terapan
Universitas Telkom
Bandung, Indonesia
arishartaman@telkomuniversity.ac.id

3rd Wisnu Adji Kharisma
PT. Len Industri
Bandung, Indonesia
wisnu.adji@len.co.id

Abstrak **´ Kemajuan teknologi menyebabkan kebutuhan manusia semakin bertambah, terdapat keterkaitan antara kemajuan teknologi dengan bertambahnya kebutuhan manusia, yaitu kebutuhan manusia menjadi lebih mudah untuk dipenuhi. Seperti penggunaan sistem operasi atau *Operating System* merupakan sebuah perangkat lunak yang mampu mengelola sumber daya (*resources*) dari *software* dan *hardware*, sehingga dapat berfungsi dengan baik serta memudahkan interaksi dengan pengguna. Pada penelitian ini telah dilakukan perancangan *embedded linux operating system* dan *upgrading software* untuk *voice terminal intercom naval system* yang bertujuan untuk mengembangkan sistem operasi yang terintegrasi dengan perangkat keras dan perangkat lunak *voice terminal* interkom untuk sistem perkapalan. Sistem operasi ini dirancang menggunakan teknologi *embedded linux* dan memiliki kemampuan untuk mengelola *input* dan *output* suara serta tampilan untuk terminal interkom. Hasil akhir dari perancangan ini adalah sistem operasi Linux yang dikustomisasi dan perangkat lunak terminal interkom yang ditingkatkan dapat beroperasi dengan lebih efisien dan efektif dalam lingkungan perkapalan. Sistem operasi yang terintegrasi dengan perangkat keras dan perangkat lunak *voice terminal* interkom juga memungkinkan pengguna untuk melakukan upgrade *software* secara mudah dan cepat, sehingga sistem ini dapat terus berkembang seiring dengan perkembangan teknologi.**

Kata kunci´ **Sistem operasi, Linux, Interkom, Peningkatan perangkat lunak.**

I. PENDAHULUAN

Operating System atau sistem operasi merupakan sebuah perangkat lunak yang mampu mengelola sumber daya (resources) dari software dan hardware, serta memiliki beberapa contoh sistem operasi seperti Windows, Mac OS dan Linux sehingga dapat berfungsi dengan baik dan memudahkan interaksi dengan pengguna [1]. Seiring dengan berkembangnya teknologi di Indonesia pada bidang telekomunikasi, intercom ethernet yang berbasis pada IP networkable sebagai salah satu bukti berkembangnya teknologi dibidang telekomunikasi, intercom ini memberikan manfaat dalam komunikasi antar pangkalan-pangkalan militer mulai dari skala besar komunikasi dengan pusat kendali hingga pos-pos jaga.

Keunggulan yang dimiliki intercom ini adalah berbasis VoIP (Voice Over Internet Protocol) yang merupakan teknologi yang bisa digunakan untuk melakukan percakapan suara jarak jauh dengan media yang digunakan adalah intranet dengan secara umum kerja VoIP mengubah data suara menjadi kode digital [2], dan intercom juga menggunakan backbone Poe (Power-over-Ethernet) sehingga untuk sistem suplay catu daya sudah terintegrasi dengan sistem komunikasi data, sehingga mempermudah instalasi di dalam kapal perang hanya butuh menarik 1 kabel UTP ke setiap node voice terminal [3], kemudian intercom dapat terintegrasi juga dengan sistem komunikasi existing pada kapal seperti public addressors (sistem pengeras suara di kapal), radio transceiver (sistem komunikasi radio kapal), telephony kapal (telephone existing di masing-masing ruang kapal) dan sistem alarm kapal (alarm tanda bahaya untuk para awak kapal). Dengan semua keunggulan ini intercom dapat memberikan kenyamanan dan keamanan untuk setiap awak kapal ketika berlayar.

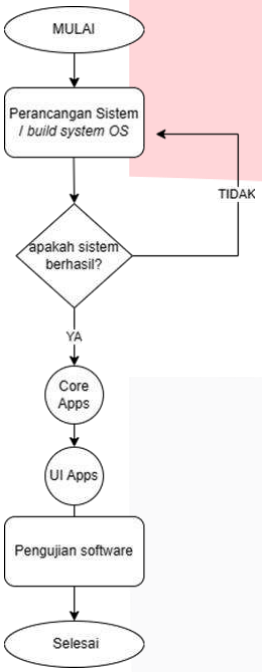
Intercom naval system merupakan teknologi audio digital dengan ethernet network backbone yang sangat fleksibel untuk mengkonfigurasi dan mengintegrasikan dengan sub-sistem lainnya. Teknologi ini merupakan perancangan dan implementasi untuk naval communication system guna menunjang komunikasi antar awak kapal. Pada perangkat yang didapati telah terpasang interkom yang masih memiliki beberapa kekurangan seperti bug dan penggunaan RAM yang digunakan terlalu besar dimana platform yang digunakan pada perangkat sebelumnya adalah platform NanoPi dengan prosesor berbasis ARM Cortex A7. Oleh karena itu pada penelitian ini, melihat dan menganalisis dari beberapa kekurangan yang ada pada software sebelumnya, maka telah dilakukan perancangan ulang pada sistem operasi yang digunakan pada voice terminal intercom naval system, perancangan ini sangat membantu dalam menutupi kekurangan dan memaksimalkan kinerja pada software sebelumnya dengan acuan pada booting time, resources dan besaran RAM dengan berapa persen RAM yang digunakan

sebelumnya, dengan upgrading software dan embedded Linux operating system untuk voice terminal intercom naval system.

II. KAJIAN TEORI

A. Deskripsi Penelitian

Pada penelitian ini dilakukan perancangan *embedded Linux operating system* pada *software intercom* yang akan diintegrasikan pada sistem komunikasi kapal laut dengan mengacu pada sistem sebelumnya yang telah digunakan dimana *embedded Linux operating system* yang digunakan sebelumnya menggunakan *platform* NanoPI dan yang akan dirancang pada penelitian ini menggunakan *platform beaglebone*. Diagram alur tahapan dapat dilihat pada Gambar 1.



GAMBAR 1
Diagram alur tahap pengerjaan penelitian

Pada bagian ini dijelaskan proses tahapan pengerjaan perancangan pada penelitian *upgrading* pada *software intercom* melalui beberapa tahap.

Tahap pertama, perancangan pada sistem operasi yang digunakan pada penelitian ini berbasis Linux. Pada tahap ini dilakukan pembangunan sistem operasi dengan menggunakan *build root* untuk mengkonfigurasi, membangun kernel Linux dan mampu menghasilkan *toolchain*, *root filesystem*, *kernel image* dan *bootloader* yang sesuai dengan kebutuhan.

Tahap kedua, melakukan pengujian untuk sistem operasi yang telah dibangun apakah sistem operasi yang telah dibangun sudah memenuhi target yang diinginkan seperti penambahan *dependencies* yang diperlukan dan optimasi pada sistem operasi dengan menguji lama kecepatan dari awal OS dinyalakan hingga masuk pada tampilan *login*, setelah OS dinyatakan siap maka akan diintegrasikan dengan appcore dan *user interface*.

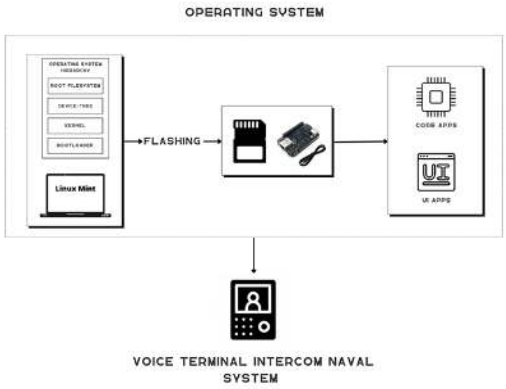
Tahap ketiga, pembuatan fungsi-fungsi aplikasi pada *software intercom* seperti fungsi *call* dan fungsi-fungsi yang digunakan akan di *cross compile* dari bahasa C ke *make file* agar dapat terintegrasi dengan sistem operasi yang digunakan.

Tahap keempat, pembuatan *User Interface* untuk *software intercom* dengan melakukan proses *slicing* dan keluaran dari proses itu akan diintegrasikan dengan fungsi-fungsi yang sebelumnya telah dibuat.

Tahap kelima, pengujian yang ditujukan khusus pada penelitian ini mengacu pada pembuatan *operating system* yang telah dibuat dengan memperhatikan *booting time* dan RAM yang digunakan ketika proses pengujian berlangsung. *Operating system* yang dikembangkan adalah mengikuti basis dari *operating system* yang digunakan pada *Intercom* KRI yang berbasis ARMv7 (NanoPi Neo Core) menjadi *intercom* yang berbasis menggunakan ARMv8 Beaglebone. Dengan menggunakan beberapa *dependencies* yang dibutuhkan, Adapun poin penting yang perlu diperhatikan adalah versi dari Linphone yang digunakan yakni versi 4 ke atas dengan menggunakan *buildroot* versi terbaru dalam konfigurasi perancangan *operating system*.

B. Proses Pengerjaan Penelitian

Pada Proyek Akhir ini akan dilakukan pengerjaan untuk upgrading software terminal voice intercom naval system pada pengerjaan ini terbagi menjadi tiga pengerjaan perancangan model sistem pada upgrading software dapat dilihat pada Gambar 3. 2



GAMBAR 2
Blok diagram model sistem pengerjaan

Pada Gambar 2 adalah blok diagram model sistem untuk *upgrading software* untuk *voice terminal intercom* pada *naval system* dengan 3 indikator utama yaitu: Core Apps, UI Apps dan *Embedded Linux Operating System* yang diintegrasikan untuk *upgrading* pada interkom. Pada penelitian ini ditujukan khusus untuk perancangan *embedded Linux operating system* yang mencakup pada *make a basic running embedded Linux operating system, configuring the Linux OS for auto-start to voice terminal apps, create a bootable embedded Linux OS with all dependencies needed by the apps dan optimizing the embedded Linux OS for the best configured device*.

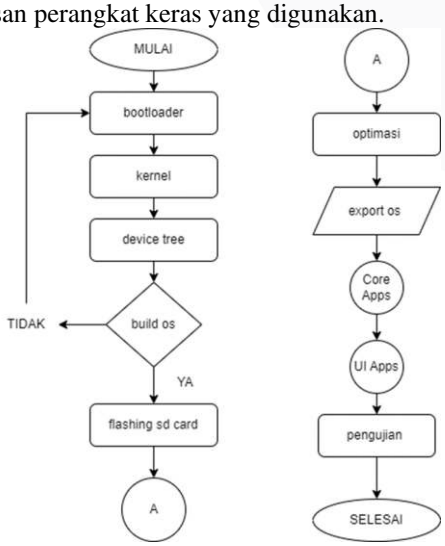
Dengan mengacu pada proses menciptakan sistem operasi berbasis Linux yang minimal dan fungsional yang disesuaikan untuk berjalan pada *system embedded*. *System embedded* adalah perangkat komputasi khusus yang dirancang untuk tugas atau aplikasi tertentu dan biasanya memiliki sumber daya terbatas, seperti daya pemrosesan, memori dan penyimpanan serta pada proses *make a basic*

running embedded Linux operating system ini melibatkan pada langkah-langkah berikut:

1. Memilih *platform* perangkat keras pada perancangan ini menggunakan platform beaglebone dengan berarsitektur kan prosesor ARM.
2. Konfigurasi kernel mendukung komponen dan fitur perangkat keras yang diperlukan untuk *platform* target pada perancangan penelitian ini menggunakan kernel Linux versi 4.19.
3. *Root filesystem* mencakup komponen-komponen penting, seperti pustaka, utilitas sistem dan node pada perancangan penelitian ini menggunakan MySQL sebagai *database* nya dan NodeJS sebagai *interpreter languages and scripting* yang telah disesuaikan dengan persyaratan spesifik dengan aplikasi target nya.
4. Konfigurasi *bootloader* atur *bootloader* untuk memuat kernel Linux dan menginisialisasi sistem dengan menggunakan *bootloader default* beaglebone pada perancangan penelitian ini.
5. *Device driver* penting untuk mengaktifkan komunikasi antar perangkat keras dan sistem operasi pada penelitian ini. Perangkat keras yang digunakan adalah beaglebone dan layar LCD dengan mengkonfigurasi *graphics support*, *display panels* dan *frame buffer device* DA8xx/OMAP-L1xx/AM335x.

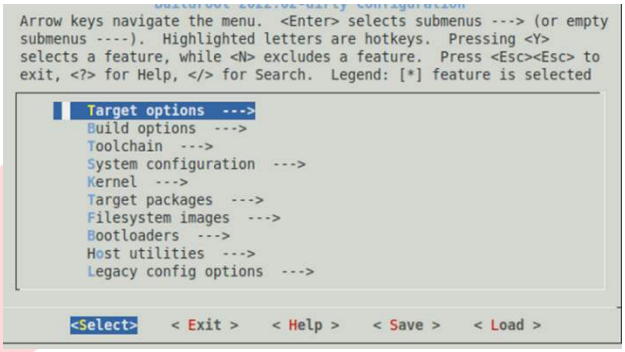
Create a bootable embedded Linux OS with all dependencies needed by the apps menciptakan sebuah sistem operasi Linux yang tertanam yang dapat di-*boot* dari perangkat keras tertentu seperti sebuah perangkat *embedded* dan sudah menyertakan semua dependensi atau komponen perangkat lunak yang diperlukan untuk menjalankan aplikasi-aplikasi tertentu sehingga sistem operasi dapat berjalan pada perangkat keras.

Kemudian membuat konfigurasi *auto-start* pada sistem operasi yang telah dirancang pada *voice terminal apps* dengan mengkonfigurasi *source* aplikasi yang disimpan di etc/init.d yang akan membuat aplikasi berjalan secara otomatis pada sistem. Setelah melakukan konfigurasi pada perancangan sistem operasi, maka dilakukan optimasi minimal pada sistem operasi yang telah dirancang untuk mencapai kinerja yang maksimal dan penggunaan sumber daya yang efisien untuk mencocokkan kemampuan dan keterbatasan perangkat keras yang digunakan.



GAMBAR 3
Flowchart Pengerjaan *Operating System*

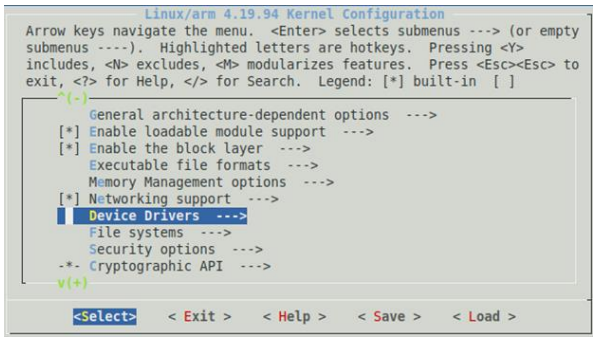
Pada Gambar 3 menjelaskan mengenai perancangan *embedded Linux operating system* untuk *voice terminal intercom naval system*. Pada perancangan *embedded Linux operating system* terdapat beberapa indikator utama diantaranya yaitu: bootloader, kernel dan *device-tree*.
Berikut tahapan pengerjaan konfigurasi sistem pengerjaan *embedded Linux operating system*. Tahap pertama menggunakan *default beaglebone_defconfig* kemudian konfigurasi *buildroot* dengan perintah make *menuconfig* kemudian konfigurasi pada bagian berikut:



GAMBAR 4
Tampilan *menuconfig*

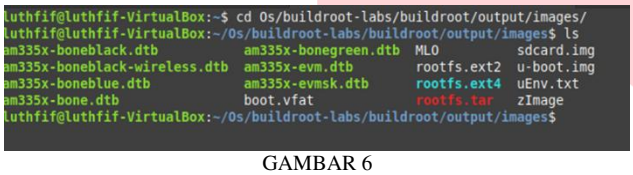
Main Item	Sub Item	Value	Commands
Target Options	Floating point strategi	Neon	
	ARM instruction	Thumb2	
Toolchain	C library	Glibc	
	Kernel headers	4.19.x custom kernel headers series	
	Install glibc utilities		Checked
	Enable C++ support		Checked
System Configuration	Init system	System V	
	System hostname	Intercom	
	System banner	Intercom KRI	
Kernel	Kernel version	Custom tarball	$(call github,beagleboard,linux,4.19.94-ti-r72/linux-4.19.94-ti-r72.tar.gz

Setelah melakukan konfigurasi *menuconfig*, selanjutnya melakukan konfigurasi dengan perintah Linux-*menuconfig* untuk dapat mengakses LCD yang digunakan.



GAMBAR 5
Tampilan dari konfigurasi kernel

Kemudian hasil konfigurasi disimpan dan dijalankan menggunakan perintah *make* untuk membangun sistem operasi yang telah dikonfigurasikan. Setelah perancangan pada sistem bootloader, kernel, *device-tree dan root file system* dengan melakukan konfigurasi menggunakan Buildroot 2021.11 berhasil kemudian.

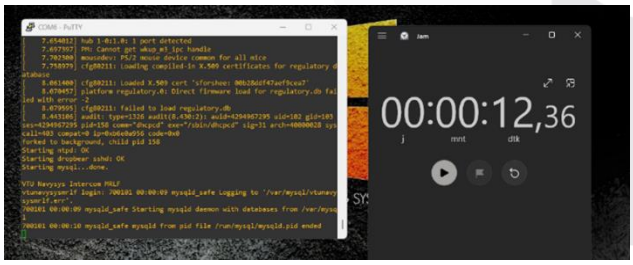


GAMBAR 6
Output File

Proses selanjutnya *flashing data* yang telah di *build* ke dalam SD Card dengan mengosongkan SD Card.

GAMBAR 7
Proses *Flashing*

Kemudian dilakukan *booting* ke beaglebone dan dilakukan optimasi pada beaglebone dengan mengukur waktu yang dibutuhkan pada saat beaglebone dinyalakan hingga masuk pada tampilan *login*.



GAMBAR 8
Booting Time

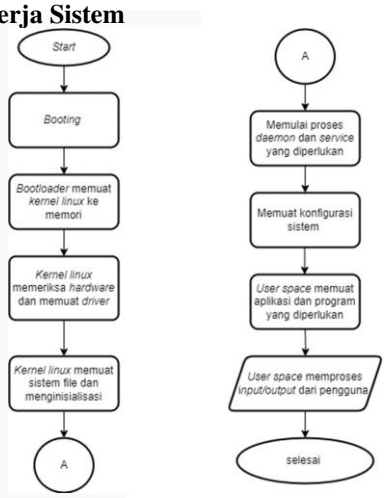
Setelah melakukan optimasi *file* OS yang telah dibangun, kemudian di *export* untuk diintegrasikan dengan perancangan *application core* dan *user interface* yang dikerjakan secara berurutan dan dilakukan pengujian yang mengacu pada *booting time*, *load* CPU dan RAM yang digunakan.

2.3 Analisa Kebutuhan Sistem

Dalam pengerjaan penelitian ini, perangkat yang digunakan terdiri dari perangkat keras (*hardware*) dan perangkat lunak (*software*) dengan spesifikasi sebagai berikut:

1. Beaglebone Black
a. 1GHz ARM Cortex-a8
b. SGX 3D Graphics Engine
c. NEON floating-point accelerator
d. 2x32-bit 200-MHz programmable real-time units
e. Memory SDRAM: 512MB DDR3L 800MHZ, Onboard Flash: 4GB, 8bit Embedded MMC
f. SD/MMC Connector for microSD
g. Debug Support: Optional Onboard 20-pin CTI JTAG, Serial Header
h. Power Source
¿ miniUSB
¿ 5VDC external via expansion header
2. SD Card
a. 16GB
3. LCD Cape
a. 5V DC Supply suitable for the BeagleBone Black, recommended 2A @ 5V.
b. Resistive Touch Display
c. 7 push buttons including LEFT, RIGHT, UP, DOWN, ENTER, RESET and POWER.
d. 4x 3.5mm Mounting holes
e. Module dimensions with BeagleBone Black connected: Approx 179.9 x 114.
4. Buildroot versi 2021.11
5. Sistem operasi Linux mint.

2.4 Cara Kerja Sistem



GAMBAR 9
Flowchart pengerjaan sistem

Berikut merupakan penjelasan *flowchart* seperti pada Gambar 9:

1. Beaglebone dihidupkan proses dimulai saat beaglebone dinyalakan.
2. *Booting* sistem operasi melakukan *booting* untuk memuat sistem *file* dan kernel.
3. *Bootloader* memuat kernel Linux ke dalam memori: *bootloader* memuat proses kernel Linux ke dalam memori dan menginisialisasi perangkat keras sehingga sistem operasi atau perangkat lunak tersebut dapat berjalan dengan benar. Kernel Linux adalah inti dari sistem operasi Linux yang bertanggung jawab untuk mengatur dan mengelola sumber daya perangkat keras, seperti memori, prosesor dan perangkat *input/output*.

4. Kernel Linux memeriksa *hardware* dan memuat *driver* yang sesuai: setelah kernel Linux dimuat, kernel akan memeriksa perangkat keras yang terpasang dan memuat *driver* yang sesuai untuk perangkat-perangkat tersebut. *Driver* adalah perangkat lunak yang memungkinkan sistem operasi berkomunikasi dengan perangkat keras.
5. Kernel Linux memuat sistem *file* dan menginisialisasi sistem: kernel Linux memuat sistem *file* yang diperlukan untuk menjalankan sistem operasi.
6. *Unit System* memulai proses daemon dan *service* yang diperlukan: setelah kernel Linux selesai menginisialisasi sistem, *unit system* menggunakan *systemV* untuk memulai proses daemon dan *service* yang diperlukan untuk menjalankan sistem operasi. Daemon adalah program yang berjalan di latar belakang dan tidak memerlukan interaksi langsung dengan pengguna.
7. *Unit System* memuat konfigurasi sistem: *unit system* memuat konfigurasi sistem yang telah ditentukan, seperti pengaturan jaringan, pengaturan waktu dan pengaturan lainnya.
8. *User Space* memuat aplikasi dan program yang diperlukan: setelah proses inisialisasi selesai, *user space* memuat aplikasi dan program yang diperlukan oleh pengguna. Ini termasuk antarmuka pengguna (GUI) dan aplikasi-aplikasi lainnya.
9. *User Space* memproses *input* dan *output* dari pengguna: *user space* bertanggung jawab untuk memproses *input* dan *output* dari pengguna, seperti menerima *input* dari tombol, menampilkan *output* ke layar dan berinteraksi dengan perangkat keras lainnya.

III. METODE

A. Pengujian Sistem

Pada bagian ini dilakukan pengujian sistem pada perancangan *embedded Linux operating system*. Optimasi adalah upaya untuk mengoptimalkan sistem operasi dengan tujuan untuk mengurangi penggunaan sumber daya yang diperlukan dan meningkatkan kinerja sistem, sehingga dapat berjalan dengan baik pada perangkat keras yang memiliki spesifikasi yang terbatas. Pengujian ini dilakukan di lingkungan PT. Len Industri dengan bertujuan dapat melihat hasil pengujian optimasi pada sistem operasi yang telah dibangun pada pengerjaan penelitian ini. Pengujian dilakukan berdasarkan 3 skenario *dataset* yang diambil secara langsung di PT. Len Industri yaitu;

1. Skenario 1 yaitu pengujian dengan menghitung *booting time* yang diperlukan.
2. Skenario 2 yaitu pengujian dengan melihat RAM yang digunakan.
3. Skenario 3 yaitu pengujian dengan melihat *load* CPU yang digunakan ketika sistem sedang berjalan.

B. Skenario 1

Berdasarkan skenario 1 yang telah dilakukan pengujian kinerja sistem dengan menghitung *booting time* yang dilakukan ketika sistem operasi dinyalakan hingga masuk pada tampilan *User Interface*.

TABEL 1
Hasil Booting Time

Aspek Pengujian	Pengujian Booting Time				Rata - rata						
	1	2	3	4	5	6	7	8	9	10	
Durasi (s)	23,28	21,08	21,85	22,71	23,32	22,50	22,83	22,60	22,47	22,23	22,49

Berdasarkan Tabel 1 dari hasil pengujian skenario 1, didapatkan rata-rata sebesar 22,49/s untuk memulai *booting* dari nol sampai siap digunakan.

C. Skenario 2

Berdasarkan skenario 2 yang telah dilakukan pengujian dengan melihat RAM yang digunakan pada saat sistem menyala, adapun hasil dari setiap pengujian yang dapat dilihat pada Tabel 2 ketika melakukan panggilan. Pengujian dilakukan ketika tampilan *screen*.

TABEL 2
Hasil pengujian RAM tampilan UI

Aspek Pengujian	Konsumsi RAM					Rata - rata
	1	2	3	4	5	
Home	29 MB	29 MB	29 MB	29 MB	29 MB	29MB
Personal	30 MB	30 MB	30 MB	30 MB	30 MB	30MB
Setting	29 MB	29 MB	29 MB	29 MB	29 MB	29MB

Pada Tabel 2 merupakan hasil pengujian skenario 2 pada pengujian RAM yang digunakan ketika sistem menampilkan beberapa tampilan UI pada *software* interkom dengan durasi 5 menit. Dari hasil pengujian tersebut mendapatkan nilai rata-rata 29 MB, 30 MB dan 29 MB yang masing-masing menampilkan pada tampilan *home*, *personal* dan *setting*.

TABEL 3
Hasil pengujian RAM pada panggilan

Aspek Pengujian	Konsumsi RAM					Rata - rata
	1	2	3	4	5	
Pemanggilan	27 MB	27 MB	27 MB	27 MB	27 MB	27 MB
Menjawab panggilan	27 MB	27MB	27 MB	27 MB	27 MB	27 MB

Pada Tabel 3 merupakan hasil dari pengujian skenario 2 pada pengujian RAM yang digunakan ketika sistem sedang melakukan pemanggilan dan menjawab panggilan. Dari hasil pengujian tersebut mendapatkan hasil pengujian dengan nilai rata-rata RAM yang digunakan, pemanggilan durasi lima menit 27 MB dan ketika menjawab pemanggilan dengan durasi lima menit nilai rata-rata RAM yang digunakan 28 MB.

D. Skenario 3

Berdasarkan skenario 3 yang telah dilakukan pengujian dengan melihat *load* CPU yang digunakan ketika sistem sedang berjalan pada saat pemanggilan dan pada saat menampilkan *screen* UI. Pengujian dilakukan dengan

REFERENSI

[1] K. S. Agus Aan Jiwa Permana, Sistem Operasi, Depok: Raja Grafindo Persada, 2022.
[2] R. T. I. W. Muntahanah, "Implementasi Voice Over Internet Protocol (VOIP) Berbasis Linux (Studi Kasus SMK Negri 03 Bengkulu)," pseudocode, pp. 41-50 , 2020.
[3] S. Robinson, "The Rapid Deployment of Light Fidelity Networking ON U.S. NAVY SHIPS," pp. 34-36, 2020.
[4] T. Pera, "Comparison of custom embedded Linux build systems : Yocto and Buildroot," p. 11, 2022.
[5] A. r. R. Sk Golam Muhammad Hasnain, "Windows, Linux, Mac Operating System and Decision Making," p. 12, 2019.
[6] X. X. X. Xia Jia, "Design of ERM Based on Embedded Linux Operating System," International Conference on Computer and Communications, p. 1278, 2019.
[7] C. S. Frank Vasquez, "Mastering Embedded Linux Programming: Create fast and reliable embedded solutions with Linux 5.4 and the Yocto Project 3.1," dalam Computers, Brimingham, Packt Publishing, 2021, p. 758.
[8] Bootlin, "Embedded Linux training," December 2022. [Online]. Available: https://bootlin.com/training/embedded-linux/. [Diakses 4 March 2023].
[9] J. Cabral, "Creating tailored OS images for embedded systems using buildroot," no. Atribuição-NãoComercial-CompartilhaIgual , p. 11, 2019.
[10] D. O. F. D. G. Q. T. Jan Van den Herrewegen, "Fill your Boots: Enhanced Embedded Bootloader Exploits via Fault Injection and Binary Analysis," Transactions on Cryptographic Hardware and Embedded Systems, no. IACR, pp. 56-58, 2020.
[11] G. Likely, "The Linux Kernel," The kernel development community, November 2021. [Online]. Available: https://www.kernel.org/doc/html/latest/devicetree/usage-model.html#id2. [Diakses 13 March 2023].
[12] L. T. Keith Cooper, "Engineering a compiler," Amsterdam, Elsevier, 2022.
[13] R. S. A. G. M. F. H. S. K. M. P. Mahendra Swain, "A reliable approach to customizing linux kernel using custom build tool-chain for ARM architecture and application to agriculture," International Journal of Electrical and Computer Engineering (IJECE) , no. Institute of Advanced Engineering and Science, p. 4924, 2019.
[14] F. H. P. S. Siddique Abubakr Muntaka, "Implementation of an IP Telephony System Based on Asterisk PBX; A," Case Study of Garden City University College, Ghana, no. ResearchGate, p. 16, 2019.
[15] Beagleboard, "Beaglebone Black," Beagleboard, [Online]. Available: https://www.beagleboard.org/boards/beaglebone-black. [Diakses 13 March 2023].