III. METODE PENELITIAN

A. Waktu dan Tempat Penelitian

Penelitian dilakukan dari bulan Oktober 2009 sampai Februari 2010, bertempat di Laboratorium Teknik Digital yang merupakan bagian dari Laboratorium Terpadu Teknik Elektro, Jurusan Teknik Elektro, Universitas Lampung.

B. Alat dan Bahan

Alat dan bahan penelitian mencakup berbagai instrumen, komponen, perangkat kerja serta bahan-bahan yang digunakan dalam proses penelitian, di antaranya:

Peralatan:

1. Komputer pribadi (PC),
2. Kabel penghubung serial dan *local area network* (LAN) dengan modul *embedded pc*,
3. Koneksi internet.

Bahan-bahan:

1. Modul *Embedded PC* CSB 625 dan CSB 925,
2. Linux Ubuntu 9.04,
3. Memori *Compact Flash*.

C. Prosedur Kerja

Dalam penyelesaian tugas akhir ini ada beberapa langkah kerja yang dilakukan untuk mencapai hasil akhir yang diinginkan, di antaranya :

1. Studi Literatur

Dalam studi literatur dilakukan pencarian informasi mengenai segala sesuatu yang berkaitan dengan *embedded system* dan *open source* linux diantaranya adalah:

a. Manual Embedded PC CSB 625 dan CSB 925
b. Konsep *embedded system*
c. Tutorial linux
d. *Kernel* sistem operasi linux
e. Modifikasi sistem operasi linux untuk *embedded system*

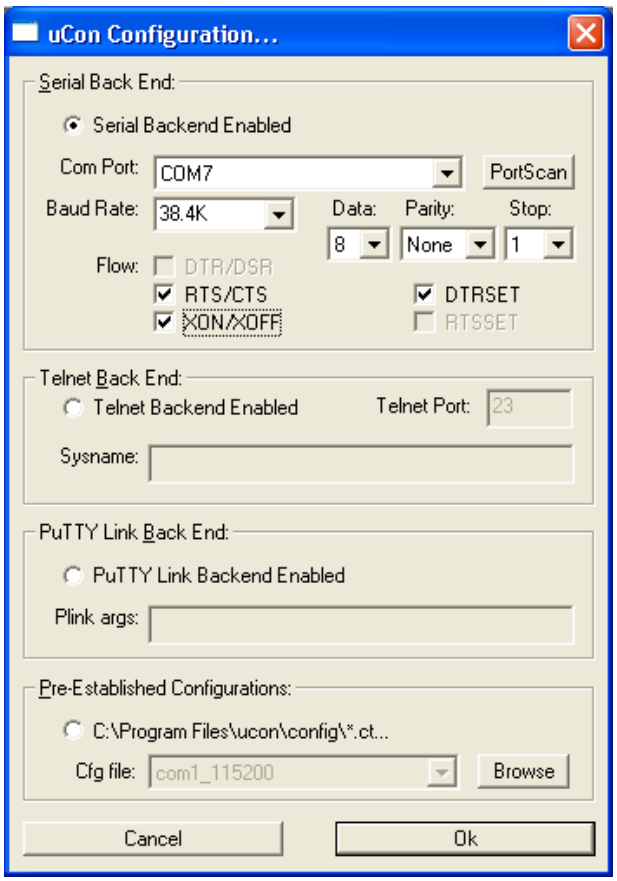
Studi literatur dilakukan dengan cara mencari dan mempelajari bahan-bahan ajar dari buku-buku dan internet.

2. Pengujian Awal Sistem Operasi GX-Linux Pada Modul CSB625

a. Pengaturan konfigurasi komunikasi serial

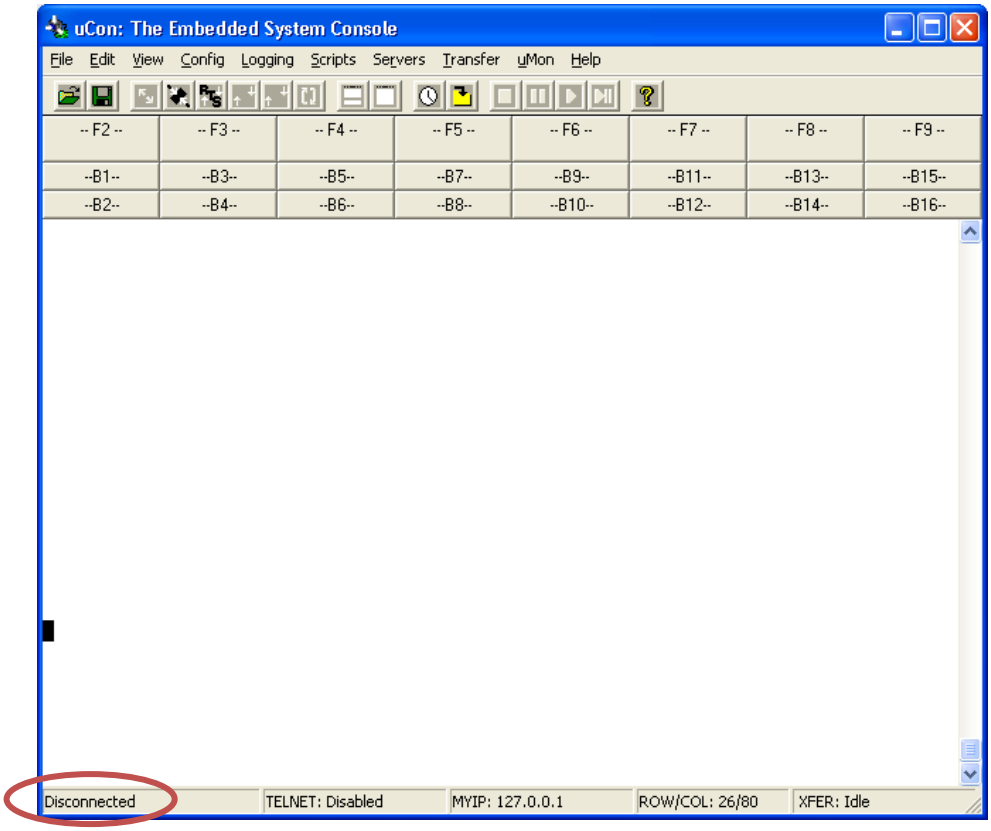
Pengujian dilakukan secara emulasi dengan menggunakan aplikasi *terminal emulator* Ucon (*The Embedded System Console*). Standar komunikasi serial yang digunakan oleh modul CSB625 adalah RS232 dengan konfigurasi *baudrate* 38400, 8-N-1 serta tidak mendukung untuk *handshaking.* Konfigurasi ini harus diatur pertama kali ketika akan

menggunakan aplikasi Ucon untuk melakukan komunikasi antara modul CSB625 dengan PC. *Port* komunikasi yang aktif dapat diketahui dengan mudah hanya dengan menekan tombol *PortScan* pada jendela konfigurasi.



Gambar 11. Konfigurasi komunikasi serial Ucon

Setelah konfigurasi ini diatur pada aplikasi Ucon, akan ditampilkan jendela utama Ucon seperti ditunjukkan pada gambar 12 berikut.



Gambar 12. Jendela utama Ucon

Pada bagian bawah jendela Ucon ini, dapat diketahui status komunikasi antara target dengan PC *host*. Dengan kondisi status *disconnected*, komunikasi antara target dengan *host* belum dapat dilakukan, hal ini karena kondisi target yang belum aktif atau belum dapat melakukan respon terhadap komunikasi tersebut. Kita dapat melakukan proses *restarting* modul CSB625 dengan menekan tombol *reset* yang ada pada *development board* CSB925.

```
TFS Scanning //FLASH/...
MICRO MONITOR 1.14.5
Platform: Cogent CSB625
CPU: PXA255 XScale
Built: Jan 15 2008 @ 09:34:51
Monitor RAM: 0xa0000000-0xa001ee00
Application RAM Base: 0xa008d000
MAC: 00:30:23:25:03:89
IP: 192.168.254.157
uMON>
```

Pada memori *flash* sistem target telah tertanam *embedded linux system*, GX-Linux. Untuk memulai menjalankan *embedded linux* pada target, kita harus melakukannya secara manual terlebih dahulu karena belum dilakukan konfigurasi untuk membuatnya berjalan secara otomatis ketika target dinyalakan. *File-file* apa saja yang sudah berada dalam memori dapat diketahui dengan mengetikkan perintah sebagai berikut pada UMon console.

```
uMON>tfs ls
 Name                      Size   Location   Flags  Info
 /$APPRAMBASE               169  0x003dda6c
 \$APPRAMBASE               167  0x003de29c
 logfile1                    53  0x003dd3dc
 monrc                       85  0x003de46c  e      envsetup
 my_file                    166  0x003de52c
 romfs.img              2222080  0x001be65c
 startlinux                7321  0x0004014c  e
 zImage                 1558452  0x00041e4c

Total: 8 items listed (3788493 bytes).
```

Informasi yang ditampilkan menunjukkan beberapa *file* yang sudah berada dalam memori *flash* termasuk *file embedded linux* diantaranya, *romfs.img* merupakan *file image root filesystem* dari *embedded linux*, *zImage* merupakan *file image* dari *kernel linux* yang jalankan pada target dan *startlinux* yang merupakan *file* untuk memulai menjalankan *embedded linux* pada target. *Flags* "e" menunjukkan bahwa *file* tersebut merupakan *executable script*.

b. Aktifasi sistem operasi GX-Linux

Sebelum melakukan aktifasi sistem operasi GX-Linux yang ada pada modul CSB625, kita harus mengetahui *file* mana yang akan dieksekusi untuk menjalankan sistem operasi yang sudah diimplementasikan pada

modul. Untuk mengetahui *file* mana yang harus dijalankan pertama kali untuk mengaktifkan sistem operasi *embedded* pada modul CSB625, pada *Umon console* masukkan perintah berikut.

```
Umon> tfs ls
```

File yang dimaksud bersifat *executable* yang ditandai dengan *flags* `e` sehingga dapat dengan mudah dilakukan eksekusi.

c. Pengujian perangkat *input*/*output* pada modul CSB625

Beberapa perangkat *input*/*output* yang dilakukan pengujian adalah *port Ethernet* dan USB. Pengujian perangkat jaringan dalam penelitian ini dilakukan dengan melakukan uji koneksi antara *target* dengan *host* melalui perintah ping pada *Command Prompt* sistem operasi Windows maupun pada *Terminal* pada sistem operasi Linux.

Sedangkan untuk pengujian *port* USB dengan menggunakan sebuah USB *flashdisk* untuk mengetahui proses identifikasi perangkat baru yang dipasangkan apakah dapat dideteksi secara langsung oleh sistem atau tidak serta untuk mengetahui proses transfer data yang terjadi antara *device* dengan sistem apakah bisa dilakukan atau tidak setelah dilakukan proses *mounting device* ke dalam sistem.

Mount merupakan proses untuk mengaitkan sebuah sistem berkas yang baru ditemukan pada sebuah piranti ke struktur direktori utama yang sedang dipakai. Tiap-tiap sistem berkas yang di-*mount* akan diberikan *mount point* atau sebuah direktori dalam pohon direktori sistem yang sedang diakses. *Mount point* adalah direktori tempat dimana akan

meletakkan sistem berkas tesebut. Kalau kita ingin me-*mount* sistem berkas berupa direktori, maka *mount point*-nya harus berupa direktori. Sebaliknya, jika yang hendak kita *mount* adalah file, maka *mount point*-nya juga harus berupa *file*.

d. Pengujian aplikasi

Fitur-fitur serta aplikasi yang telah diimplementasikan pada sistem berada pada direktori `/bin` *root filesystem*. Semua aplikasi yang ada pada direktori tersebut dapat diketahui dengan mengetikkan perintah berikut.

```
# cd bin/
# ls
```

Dari aplikasi-aplikasi yang telah diimplementasikan, pengujian dilakukan untuk mengetahui apakah aplikasi tersebut dapat berjalan dengan baik atau tidak pada modul CSB625.

3. Implementasi Sistem Operasi Baru Berbasis *Open Source* Linux

a. Deskripsi sistem

Sistem operasi *embedded* yang akan dibangun dengan menggunakan *open source* linux memiliki fitur-fitur secara umum sebagai berikut :

1. Memiliki kemampuan untuk melakukan identifikasi terhadap *device* baru yang dipasangkan,
2. Memiliki *driver-driver* untuk beberapa perangkat input/output yang sudah tersedia,

3. Memiliki *compiler* dalam sistem sehingga mendukung untuk perancangan aplikasi lebih lanjut,
4. Memiliki aplikasi standar sistem linux serta aplikasi yang didedikasikan untuk tujuan tertentu,

b. Metode implementasi secara manual

Metode implementasi secara manual dilakukan tahap demi tahap dalam konfigurasi, kompilasi dan instalasi masing-masing *tools* yang akan digunakan termasuk kernel sebagai inti sistem operasi yang akan dibangun. Beberapa tahapan yang harus dilakukan dalam perancangan ini adalah sebagai berikut.

1. Melakukan konfigurasi, kompilasi dan instalasi *Cross-platform development toolchain,*
2. Melakukan konfigurasi, kompilasi dan instalasi kernel sistem operasi yang akan digunakan pada target sistem,
3. Membuat *root filesystem* untuk target sistem,
4. Melakukan konfigurasi dan instalasi *boot software.*

Cross-platform development toolchain meliputi beberapa *tools*, yaitu *Binary utilities*, *Compiler C* dan *Library C*. Proses instalasi toolchain sendiri meliputi beberapa tahapan berikut.

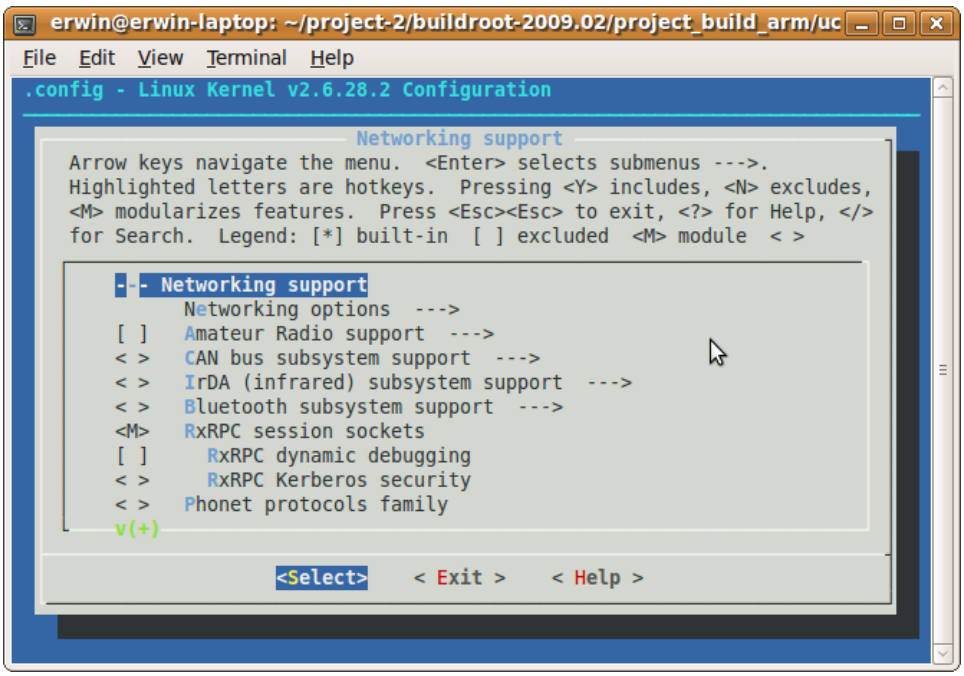
1. Instalasi *kernel header,*
2. Instalasi *binary utilities,*
3. Instalasi *bootstrap compiler,*
4. Instalasi *C library,*

5. Instalasi *full compiler.*

Instalasi *kernel header* diawali dengan melakukan konfigurasi kernel sesuai dengan arsitektur target. Modul CSB625 menggunakan prosesor Intel PXA255 yang merupakan keluarga arsitektur ARM versi 5TE, sehingga untuk konfigurasi kernel perintah yang digunakan adalah

```
make ARCH=arm CROSS_COMPILE=i386-linux- menuconfig
```

Konfigurasi kernel dapat dilakukan dengan mudah karena jendela konfigurasi yang sudah berbasis grafis (GUI).



Gambar 13. Jendela konfigurasi kernel

Instalasi *binary utilities* dilakukan dengan melakukan ekstraksi *file binutils* dengan menggunakan perintah

```
tar xvzf binutils-2.16.1.tar.gz        (ekstensi .gz)
tar xvjf binutils-2.16.1.tar.bz2       (ekstensi .bz2)
```

Selanjutnya lakukan proses konfigurasi dengan menggunakan perintah berikut dengan beberapa tambahan parameter

```
./configure –target=$TARGET –prefix=${PREFIX}
```

Untuk melakukan kompilasi, perintah yang digunakan adalah `make` sedangkan untuk proses instalasi digunakan perintah `make install`.

Cara instalasi *bootstrap compiler*, *C library* dan *full compiler* secara umum sama seperti proses instalasi *binary utilities*. Perbedaannya terletak pada parameter-parameter yang digunakan dalam proses konfigurasi. Untuk *bootstrap compiler*, perintah yang digunakan adalah

```
./configure --target=$TARGET --prefix={PREFIX} --
without-headers --with-newlib –enable-languages=c
```

Untuk *C library*, perintah yang digunakan adalah

```
./configure --host=$TARGET --prefix=”/usr” –enable-
add-ons --with-headers=${TARGET_PREFIX}/include
```

Untuk *full compiler*, perintah yang digunakan adalah

```
./configure --target=$TARGET --prefix={PREFIX} --
enable-languages=c,c++
```

Setelah proses instalasi *toolchain* selesai dilakukan, proses selanjutnya adalah kompilasi kernel. Kompilasi ini dilakukan untuk membentuk *file image* kernel yang akan digunakan oleh target. Format *file image* standar yang digunakan dalam arsitektur ARM adalah zImage, sehingga perintah yang digunakan adalah

```
make ARCH=arm CROSS_COMPILE=i386-linux- zImage
```

Sedangkan untuk kompilasi modul-modul kernel yang akan dibangun digunakan perintah

```
make ARCH=arm CROSS_COMPILE=i386-linux- modules
```

Proses selanjutnya adalah membuat *root filesystem* untuk target. *Root filesystem* merupakan direktori-direktori yang berisi *image kernel*, modul kernel, sistem *library*, aplikasi utama sistem dan aplikasi tambahan bagi pengguna. Perintah yang digunakan untuk membuat direktori berikut direktori yang harus dibuat untuk *root filesystem* adalah sebagai berikut.

```
mkdir bin dev etc lib proc sbin tmp usr var
chmod 1777 tmp

mkdir usr/bin usr/lib usr/sbin

mkdir var/lib var/lock var/log var/run var/tmp

chmod 1777 var/tmp
```

Tahapan terakhir adalah instalasi *bootloader* untuk target. *Bootloader* inilah yang bertugas pertama kali ketika sistem aktif dan akan melakukan respon untuk menjalankan kernel. Ada beberapa jenis *bootloader* yang umumnya digunakan dalam sistem linux. Akan tetapi untuk arsitektur ARM, *bootloader* standar yang digunakan adalah U-boot [6].

c. Metode implementasi secara otomatis

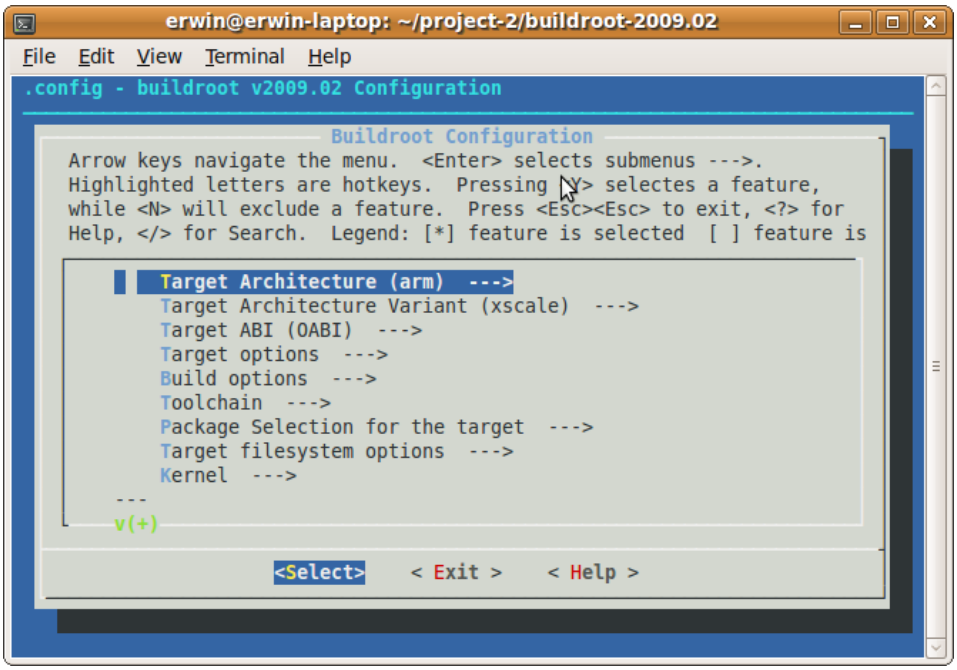
Metode implementasi secara otomatis dilakukan dengan menggunakan *tools* yang telah dirancang khusus oleh para *engineer* yang telah lama berkecimpung dalam *embedded system development* seperti *crosstool* dan *buildroot*. Proses yang dilakukan bisa menjadi lebih singkat dan sederhana karena proses konfigurasi hanya untuk mengatur spesifikasi dan kebutuhan sistem secara umum. Konfigurasi-konfigurasi lebih lanjut telah ditentukan

dengan menggunakan *script-script* yang akan dieksekusi secara otomatis. Metode ini sangat membantu dalam pengembangan sistem operasi *embedded* terutama bagi para pemula.

Dalam metode kedua ini, penulis lebih cenderung menggunakan *tools buildroot* karena lebih mudah dalam penggunaannya dan telah berbasis grafis (GUI). Langkah awal yang dilakukan adalah melakukan konfigurasi buildroot dengan mengetikkan perintah

```
make menuconfig
```

Sehingga akan muncul jendela konfigurasi *buildroot* seperti ditunjukkan pada gambar 14 berikut.



Gambar 14. Jendela konfigurasi *Buildroot*

Konfigurasi yang harus dilakukan adalah pemilihan jenis arsitektur target dan variannya, pemilihan fitur untuk *toolchain*, pemilihan paket-paket

aplikasi yang akan diinstal untuk target, jenis *filesystem* target beserta *bootloader*, dan versi kernel beserta format yang akan digunakan untuk target. Setelah proses konfigurasi selesai dilakukan, simpan konfigurasi tersebut, keluar dari jendela konfigurasi dan lakukan instalasi dengan mengetikkan perintah `make`. Selanjutnya secara otomatis proses konfigurasi, kompilasi dan instalasi akan berjalan, yang diawali dengan proses pengambilan *file* (*download*) yang diperlukan dari *server software source buildroot* dan beberapa situs yang lain [11].

4. Pengujian Kedua Pada Sistem Operasi Baru

Pengujian ini dilakukan untuk mengetahui tingkat keberhasilan sistem yang telah dibuat. Pengujian yang akan dilakukan meliputi beberapa tahap, yaitu :

a. Pengujian perangkat komunikasi, dengan menghubungkannya ke PC *host* melalui *port* komunikasi yang ada,
b. Pengujian fungsional sistem operasi *embedded linux*, dengan mengeksekusi *file image* yang ditempatkan pada memori,
c. Pengujian *software* aplikasi, dengan menjalankan program aplikasi sederhana yang telah dikembangkan.

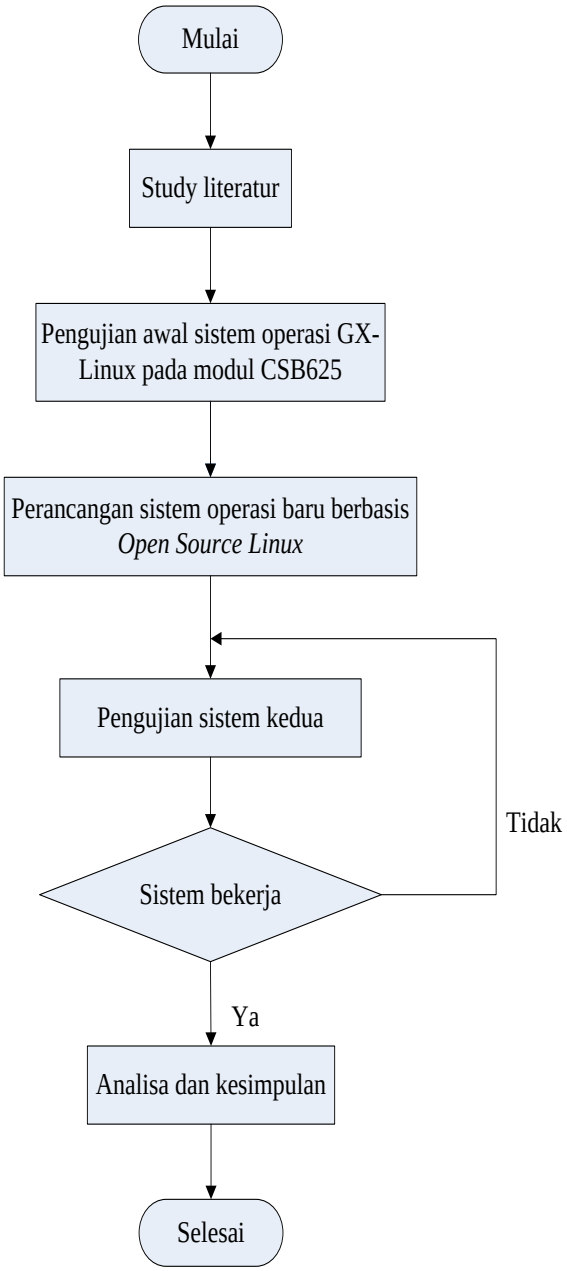
5. Analisa dan Simpulan

Analisa dilakukan dalam dua tahap. Pertama setelah proses pengujian kemampuan sistem operasi *embedded* GX-Linux pada modul CSB625. Dari pengujian tersebut akan dapat diketahui bagaimana kemampuan sistem operasi embedded yang sudah tertanam. Kedua setelah sistem operasi baru yang

diimplementasikan pada modul. Dari proses tersebut akan diketahui apakah sistem operasi memang sudah berjalan sesuai dengan yang diharapkan atau masih ada beberapa masalah yang perlu diperbaiki. Hasil dari proses analisa tersebut selanjutnya akan digunakan dalam pengambilan kesimpulan.

6. Penulisan Laporan

Akhir dari tahap penelitian ini adalah penulisan laporan dari semua kegiatan penelitian yang telah dilakukan. Secara umum urutan pekerjaan yang dilakukan dalam penelitian ini dapat dijelaskan secara sistematis dalam diagram alir pada gambar 15.



Gambar 15. Diagram Alir Penelitian