UNIVERSITAS INDONESIA

RANCANG BANGUN USB ON-THE-GO ENABLER DENGAN MIKROKONTROLER AT90USB1287

SKRIPSI

ADITYA PRAYOGA
0806365412

FAKULTAS TEKNIK
PROGRAM SARJANA TEKNIK ELEKTRO
DEPOK
DESEMBER 2010

UNIVERSITAS INDONESIA

RANCANG BANGUN USB ON-THE-GO ENABLER DENGAN MIKROKONTROLER AT90USB1287

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana

ADITYA PRAYOGA
0806365412

FAKULTAS TEKNIK
PROGRAM SARJANA TEKNIK ELEKTRO
DEPOK
DESEMBER 2010

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama : Aditya Prayoga
NPM : 0806365412
Tanda Tangan : .....
Tanggal : .....

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Aditya Prayoga
NPM : 0806365412
Program Studi : Teknik Elektro
Judul Skripsi : Rancang Bangun USB On-The-Go Enabler dengan Mikrokontroler AT90USB1287

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia

DEWAN PENGUJI

Pembimbing : Prima Dewi Purnamasari, ST., M.Sc., MT. ( )

Penguji : Ir. A. Endang Sriningsih M.T., Si ( )

Penguji : Dr. Ir. Anak Agung Putri Ratna M.Eng. ( )

Ditetapkan di : Depok

Tanggal : .....

KATA PENGANTAR

Puji syukur saya panjatkan kepada Tuhan Yang Maha Esa, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada pembuatan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

(1) Prima Dewi Purnamasari, ST., M.Sc., MT., selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
(2) orang tua dan keluarga saya yang telah memberikan bantuan dukungan material dan moral; dan
(3) sahabat yang telah banyak membantu saya dalam menyelesaikan skripsi ini

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 12 Desember 2010

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Aditya Prayoga
NPM	: 0806365412
Program Studi	: Teknik Elektro
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

Rancang Bangun USB On-The-Go Enabler dengan Mikrokontroler AT90USB1287

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada tanggal : .....

Yang menyatakan

(.....)

ABSTRAK

Nama : Aditya Prayoga
Program Studi : Teknik Elektro
Judul : Rancang Bangun USB On-The-Go Enabler dengan Mikrokontroler AT90USB1287

Dewasa ini perkembangan teknologi sangat pesat. Banyaknya penemuan akan teknologi baru mempengaruhi pola pertukaran informasi. Dengan sistem digital transfer data menjadi cepat dan ringkas. Saat ini banyak perangkat portabel digital yang mempunyai port USB. Dengan adanya koneksi USB, transfer data menjadi cepat dan secara ukuran fisik menjadi ringkas. Sistem USB memindahkan beban kerja pada sisi PC sebagai *host* dan menyederhanakan rangkaian perangkat USB *peripheral*. Dengan terbitnya suplemen USB OTG, perangkat USB yang mempunyai komputasi tinggi dapat berperan menggantikan sebagian fungsi PC sebagai *host*. Skripsi ini membahas tentang rancang bangun sistem USB On-The-Go dengan menggunakan mikrokontroler AT90USB1287. Sistem ini menghubungkan perangkat USB yang tidak memenuhi standard USB OTG sehingga antar perangkat tersebut dapat saling berkomunikasi. USB OTG Enabler ini mempunyai kecepatan akses 3,76 kBps untuk telepon genggam ke *flash drive* dan 2,24 kBps untuk *flash drive* ke telepon genggam. *Bottlencek* terjadi karena keterbatasan kecepatan serial pada platform Java.

Kata kunci:
USB, On-The-Go, Atmel, AT90USB, *host controller*, USB Host

ABSTRACT

Name : Aditya Prayoga
Study Program : Electrical Engineering
Title : The Design And Development Of USB On-The-Go Enabler Using Microcontroller AT90USB1287

Nowadays technologies development is very fast. Many invention of new technologies effect information exchange pattern. Using digital system, data transfer would be fast and simple. Currently many digital portable device have USB port. With USB connectivity, the data transfer would be fast and the physical dimension is simple. USB system were move the heavy work to PC as host and made USB peripheral circuitary simpler. As the USB OTG supplement issued, USB devices that have good computing resources can replace some of PC function as host. This paper explain about USB On-The-Go system design using AT90USB1287 microcontroller. This system can connect several USB devices that doesn't compliant with OTG standard so those device could communicate each other. This USB OTG Enabler device have transfer rate at 3,76 kBps for handphone to flash drive and 2,24 kBps for flash drive to handphone. Bottleneck occur because serial speed are limited at Java platform.

Key words:
USB, On-The-Go, Atmel, AT90USB, *host controller*, USB Host

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Dewasa ini perkembangan teknologi sangat pesat. Banyaknya penemuan akan teknologi baru mempengaruhi pola pertukaran informasi. Salah satu bidang teknologi yang berkembang pesat adalah teknologi informasi. Perangkat portabel adalah salah satu hasil dari teknologi informasi. Saat ini tidak sulit untuk menemukan perangkat portabel. Perangkat-perangkat portabel ini misalnya *handphone*, kamera digital, dan MP3 player penggunaannya sudah meluas di masyarakat.

Dengan penemuan USB yang ringkas dan cepat, perlahan-lahan USB mulai menggantikan jenis konektivitas lain seperti serial RS232 maupun paralel. Rancangan awal USB adalah memindahkan beban kerja pada PC yang mempunyai sumber daya besar dan menyederhanakan desain perangkat USB. Namun sejalan dengan kemampuan komputasi perangkat portabel yang semakin tinggi, beban kerja sistem USB dapat dilakukan oleh perangkat portabel sehingga dapat menggantikan peran PC. Standard baru ini dikenal dengan nama USB On-The-Go (OTG).

Pada skripsi ini akan disampaikan perancangan USB OTG Enabler. USB OTG memungkinkan dua buah piranti USB untuk saling berkomunikasi tanpa perlu bantuan PC. Sepintas ini seperti mengubah konsep USB, yang bila dianalogikan seperti jaringan komputer, berbasis dengan *slave* dan *master* menjadi *peer to peer*. Namun spesifikasi OTG merupakan spesifikasi tambahan (supplemen) dari spesifikasi USB rev 2.0, sehingga yang dilakukan dari standard USB OTG adalah salah satu piranti bertindak sebagai *master/host* dan dapat bertukar peran(role) menjadi *slave/peripheral* dengan protokol tertentu.

1.2 Permasalahan

Masalah yang diangkat pada skripsi ini adalah bagaimana membuat sebuah piranti USB OTG Enabler agar kompatibel dengan dua piranti USB sehingga dapat terhubung satu sama lain.

1.3 Tujuan

Tujuan skripsi ini adalah merancang dan mengimplementasikan perangkat keras dan perangkat lunak USB OTG Enabler menggunakan mikrokontroler AT90USB1287. Unjuk kerja sistem akan diamati dan diuji dengan cara mengukur kecepatan transfer data dari telepon genggam menuju *flash drive* dan sebaliknya, dari *flash drive* menuju telepon genggam. Sistem ini akan dirancang modular sehingga dapat dikembangkan untuk mendukung lebih banyak perangkat USB dan beragam fungsi.

1.4 Batasan Masalah

1. Piranti USB yang akan diuji menggunakan USB OTG Enabler hanya Telepon Genggam dan perangkat yang termasuk *Mass Storage Class* seperti USB *Flash drive*,
2. Telepon genggam yang digunakan telah didukung oleh JAVA yaitu, Siemens CX75,
3. Kabel data yang digunakan berisi IC PL2303 USB-to-Serial,
4. Kapasitas *flash drive* kurang dari 2GB dengan format FAT16,
5. Jumlah partisi pada *drive* hanya satu,
6. Konsumsi daya tidak diperhitungkan dan tidak diamati,
7. Mode kecepatan yang dibahas adalah mode *full speed* dan *low speed*.

1.5 Sistematika Penulisan

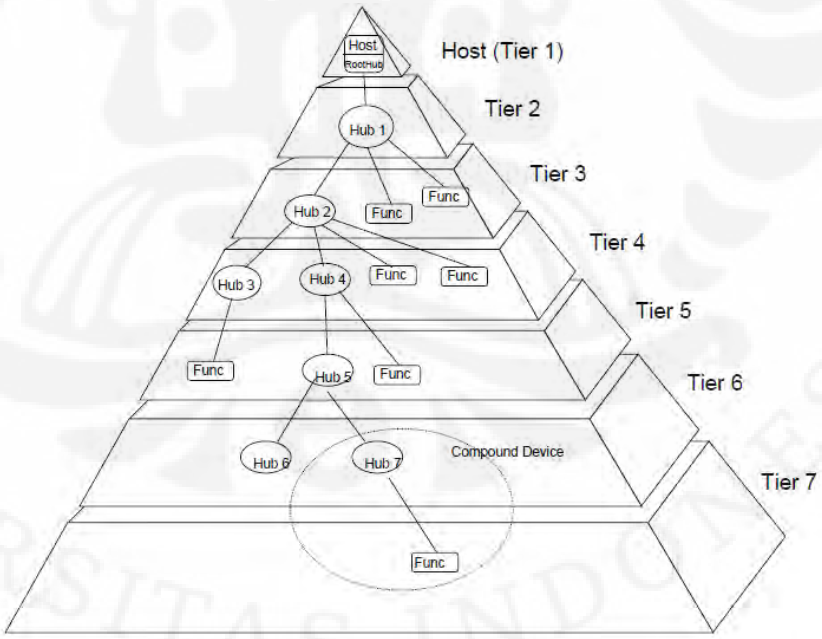
Sistematika penulisan skripsi ini dibagi atas beberapa bab. Pembagian babnya adalah sebagai berikut, bab satu berisi pendahuluan yang membahas tentang latar belakang, tujuan, batasan masalah, dan sistematika penulisan. Bab dua menjelaskan dasar teori USB, sistem file FAT, spesifikasi AT90USB1287 serta bahasa pemrograman. Bab tiga berisi perancangan perangkat keras dan perancangan perangkat lunak USB OTG Enabler. Bab empat berisi data yang didapat dari percobaan yang dilakukan dan analisa dari data yang diperoleh. Bab lima merupakan kesimpulan dari keseluruhan skripsi ini.

BAB 2
USB ON-THE-GO, MIKROKONTROLER AT90USB1287, FILE ALLOCATION TABLE, DAN JAVA

2.1 Universal Serial Bus

Universal Serial Bus (USB) adalah standar bus serial untuk perangkat tambahan dengan komputer yang mulai dikembangkan pada tahun 1996 oleh beberapa perusahaan besar industri komputer. Sistem USB mempunyai desain yang asimetris, yang terdiri dari *host controller* dan perangkat tambahan yang terhubung membentuk hirarki piramid dengan menggunakan *hub*.[1]

Jumlah perangkat yang dapat dihubungkan maksimal 127 buah dengan jumlah tingkat yang didukung adalah 7 tingkat termasuk *host controller*. Untuk menambah tingkatan digunakan *hub* yang terhubung secara *daisy chain* seperti terlihat pada Gambar 2.1. Perangkat dengan tipe *compound* menempati dua tingkat sehingga tidak dapat dihubungkan pada tingkat ke-tujuh. [1]



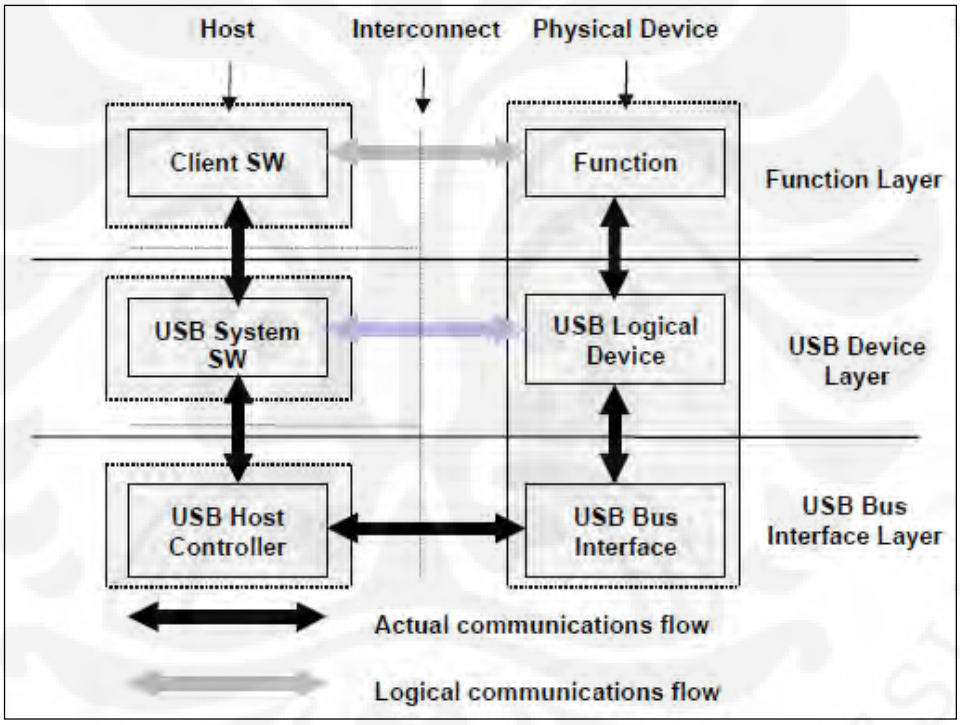
Gambar 2.1 Hirarki Arsitektur USB [1]

Desain USB ditujukan untuk menghilangkan perlunya penambahan *expansion card* ke bus PCI, dan memperbaiki kemampuan *plug-and-play* dengan memperbolehkan peralatan-peralatan ditukar atau ditambah ke sistem tanpa perlu

me-*reboot* komputer. Ketika perangkat USB dipasang, sistem komputer langsung mengenali dan memproses *device driver* yang diperlukan untuk menjalankannya.

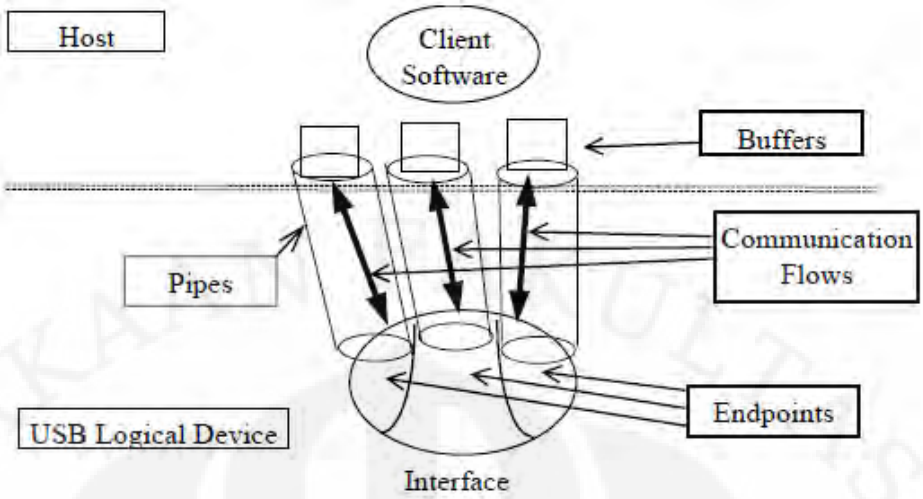
USB dapat menghubungkan perangkat tambahan komputer seperti mouse, keyboard, pemindai gambar, kamera digital, printer, hard disk, dan komponen jaringan. USB kini telah menjadi standar bagi perangkat multimedia seperti pemindai gambar dan kamera digital.[2]

USB adalah sistem bus yang *host-centric* yaitu semua kegiatan bus diatur oleh *host* dengan sistem *pooling*. Pada mode *full speed* dan *low speed pooling* dilakukan tiap 1 ms sedangkan pada mode *Hi-speed pooling* dilakukan setiap 125 µs.[3]



Gambar 2.2 Alur Komunikasi USB [1]

Gambar 2.2 menunjukkan alur komunikasi pada sistem USB. Lapisan pada bagian bawah mengabstrakkan lapisan diatasnya sehingga komunikasi yang terjadi hanya pada komponen yang mempunyai lapisan yang sama contohnya pada *client SW* berkomunikasi dengan *Function*. Pada Gambar 2.3 terlihat hubungan logikal antara *client software* dengan *Function*.



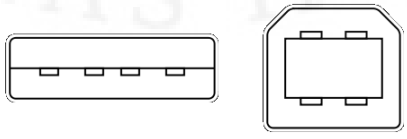
Gambar 2.3 Hubungan Pada Lapisan Atas [1]

Endpoint dapat diartikan sebagai sumber data atau titik tujuan data yang merupakan ujung saluran komunikasi USB. *Endpoint* juga dilihat sebagai antarmuka fungsional perangkat keras dan *firmware* yang berjalan di fungsi tersebut.

Karena perangkat USB mengirim dan menerima data melalui beberapa *endpoint* yang terpasang secara seri, perangkat lunak di sisi klien akan mentransfer data melalui *pipe*. *Pipe* adalah hubungan logika antara *host* dan *endpoint*. *Pipe* menentukan jenis transfer yang digunakan yaitu control, bulk atau interupsi, serta arah dari data yang mengalir padanya,

Spesifikasi USB terbaru saat ini adalah versi 3.0. Namun masih berupa *draft* dengan prototipe yang diproduksi kuartal pertama 2010. Pada tulisan ini akan dibahas pada spesifikasi versi 2.0 yang sudah matang.

Ada dua macam konektor USB, yaitu konektor A (Gambar 2.4a) untuk hubungan ke *host* dan konektor B (Gambar 2.4b) untuk hubungan ke perangkat USB. Kedua jenis konektor ini dapat dibedakan dengan mudah untuk menghindari kesalahan pemasangan.



(a) (b)

Gambar 2.4 Jenis Konektor USB [3]

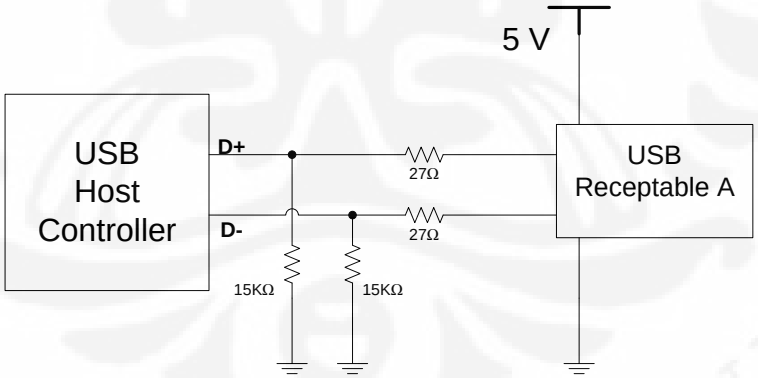
Untuk hubungan lebih luas ke perangkat berukuran kecil seperti *handphone*, PDA, dan sebagainya dibuatlah konektor mini-A dan mini-AB. Semua jenis konektor dihubungkan dengan empat kabel seperti pada Tabel 2.1.

Tabel 2.1 Koneksi Kabel USB [4]

Pin	Warna Kabel	Fungsi
1	Merah	V_{BUS} (5 Volt)
2	Putih	D-
3	Hijau	D+
4	Hitam	GND

2.1.1 USB Host

Hanya ada satu *host* dalam sistem USB. Antarmuka USB pada sistem komputer disebut sebagai *Host Controller*. *Host Controller* dapat diimplementasikan dalam kombinasi perangkat keras, *firmware*, atau perangkat lunak. Sebuah *root hub* menyatu pada sistem *host* untuk menyediakan beberapa titik pemasangan.



Gambar 2.5 Koneksi Resistor dan Kabel pada *Host* [5]

Pada Gambar 2.5 terlihat bahwa terdapat dua buah resistor *pull-down* sebesar 15KΩ. Hal ini ditujukan agar ketika tidak ada perangkat yang terpasang nilai logika pada sistem menjadi D+ = 0 dan D- = 0. Spesifikasi USB mensyaratkan bahwa impedansi driver harus bernilai antara 28Ω hingga 44Ω sehingga resistor 27Ω dipasang seri untuk memenuhi persyaratan tersebut. [2]

Sistem *host* mempunyai beberapa tugas[5], yaitu:

- Mendeteksi pemasangan (attachment) dan pelepasan (removal) USB
- Mengatur aliran kontrol antara *host* dan perangkat
- Mengatur aliran data antara *host* dan perangkat
- Mengumpulkan statistik aktivitas dan status
- Menyediakan daya untuk perangkat yang terpasang

Software sistem USB pada *host* mengatur interaksi antara perangkat dan *host driver*. Ada lima area interaksi[5], yaitu:

- Enumerasi perangkat dan konfigurasi
- Transfer data Isochronous
- Transfer data Asynchronous
- Manajemen daya
- Informasi perangkat dan manajemen bus

Pada inisialisasi perangkat oleh *host*, terdapat beberapa tahap dasar[5] yaitu:

- Set Address : menetapkan alamat perangkat dari alamat default 0.
- Get Device Descriptor : membaca informasi tentang perangkat seperti manufaktur, versi *firmware* dan sebagainya.
- Get Configuration Descriptor : membaca konfigurasi *endpoint* yang digunakan.
- Get Interface Descriptor : membaca macam-macam konfigurasi *interface* yang mungkin digunakan.
- Get String Descriptor : membaca data string yang berisi nama manufaktur dan nama produk dalam format Unicode.

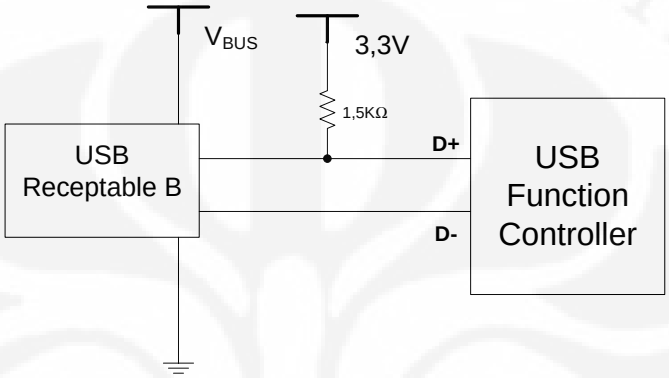
Setelah tahap-tahap ini dilewati maka software system USB akan mencari *driver* yang dibutuhkan sehingga perangkat siap digunakan oleh *client software.* Syarat-syarat menjadi *host* [1]:

- Resistor pull down 15kΩ pada D+ dan D-
- Sebagai sumber tegangan pada Vbus

Selain syarat secara elektrik, dibutuhkan juga kemampuan untuk:

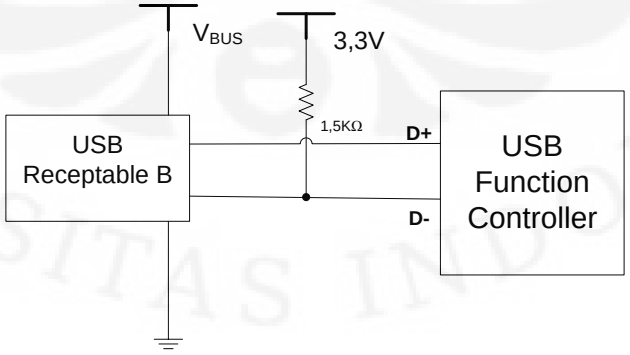
- Mengirim paket SOF(Start Of Frame)
- Mengirim paket SETUP, In dan OUT
- Mengirim sinyal USB Reset
- Menyediakan manajemen daya

2.1.2 USB Function



Gambar 2.6 Koneksi Resistor dan Kabel USB Full Speed[4]

Pada rangkaian USB Function terdapat resistor pull up sebesar 1,5KΩ yang terpasang pada 3,3V. Letak atau hubungan resistor pull up ini menentukan mode kecepatan yang didukung oleh perangkat tersebut. Pada Gambar 2.6 terlihat bahwa resistor pull up terpasang pada pin D+, ini menandakan bahwa perangkat tersebut beroperasi pada mode kecepatan *Full Speed* sedangkan pada Gambar 2.7, perangkat beroperasi pada mode kecepatan *Low Speed* karena resistor pull up yang terpasang adalah pada pin D-.



Gambar 2.7 Koneksi Resistor dan Kabel USB Low Speed[4]

2.1.3 USB On-The-Go

USB OTG memungkinkan dua buah piranti USB untuk saling berkomunikasi tanpa perlu bantuan PC. Sepintas ini seperti mengubah konsep USB, bila dianalogikan dengan jaringan komputer, dari slave dan master menjadi *peer to peer*. Namun spesifikasi OTG merupakan spesifikasi tambahan (supplemen) dari spesifikasi USB rev 2.0, sehingga yang dilakukan dari standard USB OTG adalah salah satu piranti bertindak sebagai *master/host* dan dapat bertukar peran(role) menjadi *slave/peripheral* dengan protokol tertentu. [6]

Gambar 2.8 Skema Koneksi USB OTG [6]

Aplikasi-aplikasi yang dapat dilakukan OTG dapat dilihat pada Tabel 2.2 dan Gambar 2.8.

Tabel 2.2 Macam-macam Aplikasi USB OTG [6]

OTG HOST	PERIPHERAL	TASK
PDA	PDA	Pertukaran file
	Telepon genggam	Menjelajah web dan mengirim e-mail
	Kamera digital	Pertukaran gambar
	Keyboard, Mouse	Antarmuka pengguna
	Printer	Mencetak file dan gambar
	Portable storage	Menyimpan / mengambil file data
	Portable audio player	Menyimpan / mengambil file musik
	Scanner	Memindai gambar
	GPS	Mendapatkan arah dan peta
Telepon genggam	Telepon genggam	Pertukaran info kontak, pesan, dan lagu
	PDA	Pertukaran file dan menjelajah web
	Kamera digital	Mengunggah gambar ke web

Tabel 2.2 Macam-macam Aplikasi USB OTG [6]

OTG HOST	PERIPHERAL	TASK
	Digital audio player	Pertukaran lagu
	Card scanner	Memindai kartu nama
Kamera digital	Kamera digital	Pertukaran gambar
	Telepon genggam	Mengunggah gambar ke web dan mengirim e-mail
	Printer	Mencetak gambar
	Mass storage	Menyimpan / Mengarsipkan gambar
Digital audio player	Digital audio player	Pertukaran lagu
	Speakers	Memainkan lagu
	Storage	Menyimpan / mengambil lagu
Portable storage	Digital audio player	Menyimpan / mengambil lagu
	Kamera digital	Menyimpan gambar
	Kamera video digital	Menyimpan video
Printer	Kamera digital	Mencetak gambar
	Scanner	Mencetak gambar yang dipindai
	Mass storage	Mencetak file yang tersimpan

Pada spesifikasi OTG diperkenalkan dua protokol baru yaitu HNP (Host Negotiation Protocol) dan SRP (Session Request Protocol). HNP digunakan untuk bernegosiasi piranti mana yang bertindak sebagai *host* tanpa harus mengubah secara fisik pemasangan kabel. SRP digunakan bila piranti peripheral ingin melakukan transaksi, dengan mengirim sinyal khusus maka *host* akan merespon, bila diijinkan maka akan dilakukan *setup* koneksi.

Spesifikasi OTG menuntut beberapa perubahan pada desain hardware, yaitu USB transceiver yang digunakan harus dapat memberikan daya 5V dengan arus minimal 8mA, selain itu resistor pull up dan pull down harus dapat diatur waktu pemasangannya.[5]

Selain pada sisi hardware, kompleksitas pun terdapat pada sisi software. Piranti yang bertindak sebagai *host* harus mempunyai driver piranti peripheral yang terhubung dengannya sehingga dari awal harus sudah ditentukan piranti peripheral apa saja yang didukung oleh *host* tersebut. Tidak tersedianya driver bukan hanya

karena keterbatasan storage piranti *host* tapi juga dari pihak pengembang piranti peripheral yang hanya merilis driver untuk PC. [5]

2.1.4 Protokol USB

Setiap transaksi USB terdiri dari:

- Paket token (header yang mendefinikan data selanjutnya),
- Paket data yang opsional,
- Paket status (sebagai control kesalahan)

Karena USB adalah bus yang terpusat di *host*, tiap transaksi akan dimulai oleh *host*. Paket pertama yang dimunculkan oleh *host* adalah token yang berisi informasi tentang bagian yang harus mengikutinya, informasi tentang sifat baca atau tulis yang dimiliki transaksi, dan juga informasi bentuk *endpoint* dan alamat untuknya. Paket berikutnya adalah data packet yang berisi data yang ditransaksikan, sekaligus diikuti paket jabat tangan (*handshake*) yang melaporkan jika token atau data sudah diterima atau malah melaporkan kegagalan *endpoint* menerima data. [4]

2.1.5 Tipe Paket USB

Gambar 2.9 Format Paket USB [3]

Tipe Paket	Format Field
Token	Sync | PID | ADDR | ENDP | CRC5 | EOP
Data	Sync | PID | Data | CRC16 | EOP
Handshake	Sync | PID | EOP
Start Of Frame (SOF)	Sync | PID | Frame Number | CRC5 | EOP

Gambar 2.9 menunjukkan format paket USB. Setiap paket harus diawali dengan Sync yang penggunaannya hanya ditujukan untuk mekanisme sinkronisasi clock saja. 2 bit terakhir menandai awal field PID. Sync bernilai 0000 0001. Pada implementasi di lapangan, format-format ini cenderung untuk tidak diperhatikan

oleh desainer umum karena format-format ini sudah dilaksanakan pada level IC yang digunakan. Desainer hanya diberikan field Data, ADDR, dan ENDP saja.

2.1.6 Jenis-Jenis Field pada Paket USB

Setiap paket USB tersusun dari berbagai macam field[3], yaitu:

a. PID

Kependekan dari Packet IDentifier.

b. ADDR

Field ADDRess ini menandakan kemana paket ini ditujukan.

c. ENDP

Field endpoint ini mempunyai lebar 4 bit sehingga jumlah endpoint maksimum adalah sebanyak 16 buah. Namun pada low speed mode hanya diperbolehkan mempunyai maksimum sebanyak 3 buah.

d. Data

Field data ini dapat mempunyai isi hingga 1024 byte namun masih dapat diatur sesuai kemampuan device yang rata-rata hanya mampu 8-64 byte saja.

e. CRC

Field ini berisi hasil perhitungan algoritma Cyclic Redudancy Check yang dilakukan pada data yang dilindungi

2.2 Mikrokontroler AT90USB1287

Atmel sebagai salah satu vendor yang mengembangkan dan memasarkan produk mikroelektronika telah menjadi suatu teknologi standard bagi para desainer sistem elektronika masa kini. Dengan perkembangan terakhir, yaitu generasi AVR (*Alf and Vegard Risc processor*), para desainer sistem elektronika telah diberi suatu teknologi yang memiliki kapabilitas yang amat maju, tetapi dengan biaya ekonomis yang cukup minimal.[7]

Mikrokontroler AVR memiliki arsitektur RISC 8 bit dimana semua instruksi dikemas dalam kode 16-bit (word) dan sebagian besar instruksi dieksekusi dalam 1 (satu) siklus clock, berbeda dengan intruksi MCS51 yang membutuhkan 12

siklus clock. Tentu saja ini terjadi karena kedua jenis mikrokontroler tersebut memiliki arsitektur berbeda. AVR seperti sudah dikemukakan sebelumnya, berteknologi RISC (*Reduced Instruction Set Computing*) sedangkan seri MCS51 berteknologi CISC (*Complex Instruction Set Computing*). Secara umum, AVR dapat dikelompokkan menjadi 4 kelas, yaitu ATtiny, AT90xx, ATmega dan AT86RFxx. Pada dasarnya yang membedakan masing-masing kelas adalah memori, periperal, dan fungsinya. Dari segi arsitektur dan instruksi yang digunakan dapat dikatakan sama.[8]

2.2.1 Fitur

- CPU : AVR dengan 32 buah register
- Memori : 128 KB *Flash*, 8 KB SRAM, 4 KB EEPROM, 64KB eksternal.
- Saluran I/O sebanyak 48 buah yang dibagi menjadi Port A, Port B, Port C, Port D, Port E, Port F.
- Empat buah *Timer/Counter* dengan kemampuan pembandingan.
- ADC 10 bit sebanyak 8 saluran.
- *Watchdog Timer* dengan osilator internal.
- Unit interupsi internal dan eksternal
- Port antarmuka SPI
- Antarmuka komparator analog
- Port antarmuka USART untuk komunikasi serial.
- Enam pilihan mode *sleep* menghemat penggunaan daya listrik.
- USB 2.0 Full-speed/Low-speed Device and On-The-Go Module
- USB Full-speed/Low Speed Device Module with Interrupt on Transfer Completion
- 48 MHz PLL for Full-speed Bus Operation
- 6 buah endpoint yang dapat diprogram dengan arah IN atau OUT dan modus Bulk, Interrupt, atau Isochronous.

2.3 *File system* FAT (File Allocation Table)

File system FAT (File Allocation Table) dirancang pada akhir 1970 untuk mendukung sistem operasi Microsoft® MS-DOS®. Awalnya dikembangkan

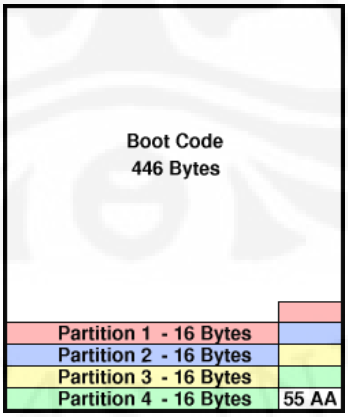
untuk arsitektur IBM PC sebagai sistem file sederhana bagi *floppy disk* yang berukuran kurang dari 500KB. Karena arsitektur IBM PC menggunakan prosesor Intel, format penyimpanan yang digunakan adalah little endian yaitu LSB diletakkan pada byte pertama.

Seiring dengan perkembangan kapasitas media penyimpanan yang makin besar, FAT dikembangkan agar mendukung media-media tersebut dengan tetap menjaga kompabilitas. Saat ini ada 3 (tiga) tipe FAT, yaitu FAT12, FAT16, dan FAT32. Perbedaan antara ketiga tipe tersebut adalah dari segi ukuran, seperti terlihat pada namanya. Ukuran FAT entry pada FAT12 adalah 12 bit, pada FAT16 adalah 16 bit, dan pada FAT32 adalah 32 bit.[8]

Pada skripsi ini yang akan dibahas adalah sistem file FAT16 yang umum digunakan pada banyak perangkat portabel dengan kapasitas maksimal yang didukung adalah kurang dari 2GB.

2.3.1 Struktur MBR

Sektor pertama dari sebuah *drive* disebut dengan Master Boot Record (MBR). Setiap drive difabrikasi dengan ukuran sektor 512 byte. MBR tersusun dari boot code yang berisi program untuk memulai sistem operasi dan tabel partisi dan diakhiri dengan kode penanda (*signature*) 0xAA55 seperti terlihat pada Gambar 2.10.



Gambar 2.10 Tataletak MBR [9]

Pada struktur tabel partisi (Gambar 2.11), terdapat beberapa field namun cukup dua field saja yang perlu diperhatikan untuk penggunaan sederhana. Field pertama yaitu kode tipe yang berisi tipe partisi (Tabel 2.3) dan field ke-dua yaitu LBA

Begin yang berisi pada alamat relatif awal partisi terhadap MBR dalam ukuran sektor.

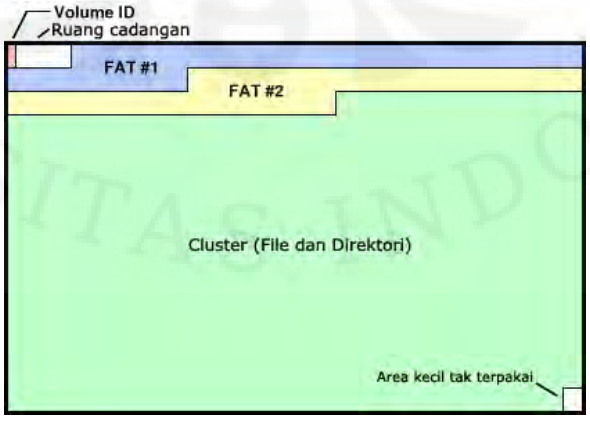
Gambar 2.11 Struktur Tabel Partisi [9]

Tabel 2.3 Beberapa kode tipe partisi [10]

Nilai	Deskripsi
00h	Tidak diketahui atau tidak ada
01h	FAT12
04h	FAT16 (Partisi lebih kecil dari 32MB)
05h	Partisi Extended MS-DOS
06h	FAT16 (Partisi lebih besar dari 32MB)
0Bh	FAT32 (Partisi sampai dengan 2048GB)
0Ch	FAT32 LBA
0Eh	FAT16 LBA
0Fh	FAT12 LBA

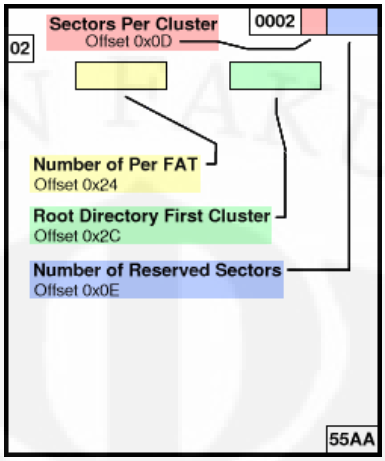
2.3.2 Struktur Data FAT

Tataletak dari sebuah sistem file FAT sederhana. Sector pertama selalu Volume ID, lalu diikuti dengan beberapa ruang cadangan yang disebut reserved sector. Setelah reserved sector adalah dua salinan dari FAT (File Allocation Table). Bagian sisa dari sistem file adalah data yang disusun dalam cluster seperti ditunjukan pada Gambar 2.12



Gambar 2.12 Tataletak Keseluruhan Sistem File FAT [9]

Gambar 2.13 tidak menggambarkan seluruh struktur Volume ID. Beberapa field penting yang dibutuhkan untuk mengakses data pada cluster ditunjukkan pada Tabel 2.4.



Gambar 2.13 Struktur Volume ID [9]

Untuk mendapatkan alamat awal FAT dalam satuan sector diperlukan perhitungan menggunakan Persamaan 2.1[9] dimana nRS menunjukkan jumlah sektor tercadang.

$$Awal\_FAT = Awal\_Partisi + nRS \quad (2.1)$$

Dimana:

Awal_Cluster : Alamat sektor awal cluster data & direktori

Awal_Partisi : Alamat sektor awal partisi yang didapat dari tabel partisi

nRS : Jumlah sektor tercadang

Sedangkan untuk mendapatkan alamat awal cluster yang berisi data, diperlukan perhitungan menggunakan Persamaan 2.2[9] dimana nRS adalah jumlah sektor tercadang, nFAT adalah Jumlah FAT, dan spf adalah sektor per FAT.

$$Awal\_Cluster = Awal\_Partisi + nRS + (nFAT * spf) \quad (2.2)$$

Dimana:

Awal_Cluster : Alamat sektor awal cluster data & direktori

Awal_Partisi : Alamat sektor awal partisi yang didapat dari tabel partisi

nRS : Jumlah sektor tercadang

nFAT : Jumlah FAT pada partisi

spf : Jumlah sektor per FAT

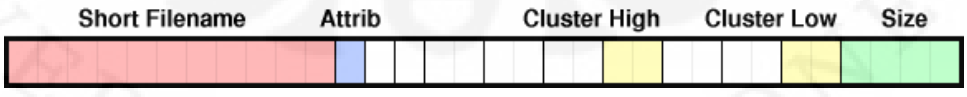
Variabel-variabel ini didapatkan dari tabel partisi dan volume ID.

Tabel 2.4 Field Penting pada Volume ID [9]

Field	Offset	Ukuran	Nilai
Byte per Sector	0x0B	16 bit	Selalu 512 byte
Sector per Cluster	0x0D	8 bit	1, 2, 4, 8, 16, 32, 64, 128
Jumlah sector tercadang	0x0E	16 bit	Umumnya 0x0020
Jumlah FAT	0x10	8 bit	Selalu 2
Sektor per FAT	0x24	32 bit	Tergantung ukuran partisi
Cluster pertama Root Direktori	0x2C	32 bit	Umumnya 0x00000002
Penanda	0x1FE	16 bit	Selalu 0xAA55

2.3.3 Struktur Direktori FAT

Entri direktori FAT mempunyai ukuran sebesar 32 byte yang mempunyai susunan yaitu nama file 11 byte, atribut direktori 1 byte, 1 byte tercadang untuk Windows NT, waktu pembuatan direktori dalam satuan milisekon 1 byte, waktu pembuatan direktori 2 byte, tanggal pembuatan direktori 2 byte, tanggal akses terakhir 2 byte, alamat *high* cluster pertama direktori 2 byte, waktu terakhir penulisan 2 byte, tanggal terakhir penulisan 2 byte, alamat *low* cluster pertama direktori 2 byte serta 4 byte untuk ukuran dalam satuan byte. Gambar 2.14 tidak menampilkan semua field tersebut karena banyak field yang dapat diacuhkan untuk penggunaan sederhana.



Gambar 2.14 Struktur Entri Direktori [9]

2.4 Bahasa pemrograman Java

Java adalah bahasa pemrograman yang disusun oleh James Gosling yang dibantu oleh rekan-rekannya seperti Patrick Naughton, Chris Warth, Ed Frank, dan Mike Sheridan di suatu perusahaan perangkat luinak bernama Sun Microsystems, pada

tahun 1991. Bahasa pemrograman ini mula-mula diinisialisasi dengan nama "Oak", namun pada tahun 1995 diganti namanya menjadi "Java".

Alasan utama pembentukan bahasa Java adalah untuk membuat aplikasi-aplkikasi yang dapat diletakkan diberbagai macam perangkat elektronik, seperti *microwave oven* dan *remote control* , sehingga Java harus bersifat *portable* atau yang lebih sering disebut dengan *platform-independent* (tidak bergantung pada *platform*). Itulah yang menyebabkan dalam dunia pemrograman Java, dikenal adanya istilah "*write once, run everywhere*", yang berarti kode program hanya ditulis sekali, namun dapat dijalankan di bawah *platform* manapun, tanpa harus melakukan perubahan kode program.[11]

2.4.1 Arsitektur Java

Secara arsitektur, Java tidak berubah sedikit pun semenjak awal mula bahasa tersebut dirilis. Kompiler Java (yang disebut dengan **javac** atau *Java Compiler*) akan mentransformasikan kode-kode dalam bahasa Java ke dalam suatu *bytecode*. *Bytecode* adalah sekumpulan perintah hasil kompilasi yang kemudian dapat dieksekusi melalui sebauh mesin komputer abstrak, yang disebut dengan JVM (*Java Virtual Machine*). JVM juga sering dinamakan sebagai *interpreter*, karena sifatnya yang selalu menerjemahkan kode-kode yang tersimpan dalam *bytecode* secara baris demi baris.[11]

2.4.2 Macam-macam edisi Java

Sun Microsystems telah mendefinisikan tiga macam edisi dari Java versi 2, yaitu sebagai berikut:

- ***Java 2 Standard Edition (*J2SE)**, yang digunakan untuk mengembangkan aplikasi-aplikasi *desktop* dan *applet* (aplikasi Java yang dapat dijalankan di dalam *web browser*).

- ***Java 2 Enterprise Edition (*J2EE)**, merupakan *superset* dari J2SE yang memperbolehkan kita untuk mengembangkan aplikasi-aplikasi berskala besar (*enterprise*), yaitu dengan melakukan pembuatan aplikasi-aplikasi di sisi *server* dengan menggunakan EJBs (*Enterprise JavaBeans*), aplikasi *web* dengan menggunakan *servlet* dan JSP (*Java Server Pages*) dan teknologi

lainnya seperti CORBA (*Common Object Request Broker Architechture*) dan XML (*Extensible Markup Language*).

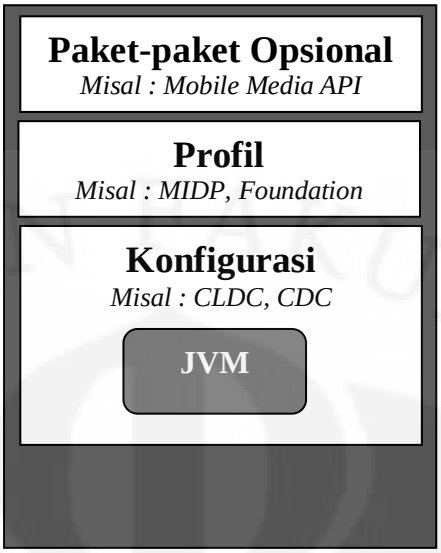
- ***Java 2 Micro Edition (*J2ME)**, merupakan *subset* dari J2SE yang digunakan untuk menangani pemrograman di dalam perangkat-perangkat kecil, yang tidak memungkinkan untuk mendukung implementasi J2SE secara penuh.

2.4.3 J2ME

J2ME merupakan sebuah kombinasi yang terbentuk antara kumpulan interface Java yang sering disebut dengan Java API (Application Programming Interface) dengan JVM (Java Virtual Machine) yang didesain khusus untuk alat, yaitu JVM dengan ruang yang terbatas. Kombinasi tersebut kemudian digunakan untuk melakukan pembuatan aplikasi-aplikasi yang dapat berjalan di atas alat (dalam hal ini mobile device).[11]

Konfigurasi merupakan bagian yang berisi JVM dan beberapa *library* kelas lainnya. Perlu diperhatikan bahwa JVM yang dimaksud di sini bukanlah JVM tradisional seperti yang terdapat pada J2SE, melainkan JVM yang sudah didesain secara khusus untuk alat. Terdapat dua buah konfigurasi yang disediakan oleh Sun Microsystems, yaitu CLDC (*Connected Limited Device Configuration*) dan CDC (*Connected Device Configuration*). Target alat dari konfigurasi CLDC adalah alat-alat kecil, seperti telepon selular, PDA, dan *pager*.[11]

Profil merupakan bagian perluasan dari konfigurasi. Artinya, selain sekumpulan kelas yang terdapat pada konfigurasi, terdapat juga kelas-kelas spesifik yang didefinisikan lagi di dalam profil. Dengan kata lain, profil akan membantu secara fungsional yaitu dengan menyediakan kelas-kelas yang tidak terdapat di level konfigurasi. Adapun profil yang sangat populer penggunaannya adalah profil yang disediakan oleh Sun Microsystems, yaitu MIDP (*Mobile Information Device Profile*).[11]



Gambar 2.15 Lapisan pada J2ME [11]

2.4.4 Konfigurasi CLDC

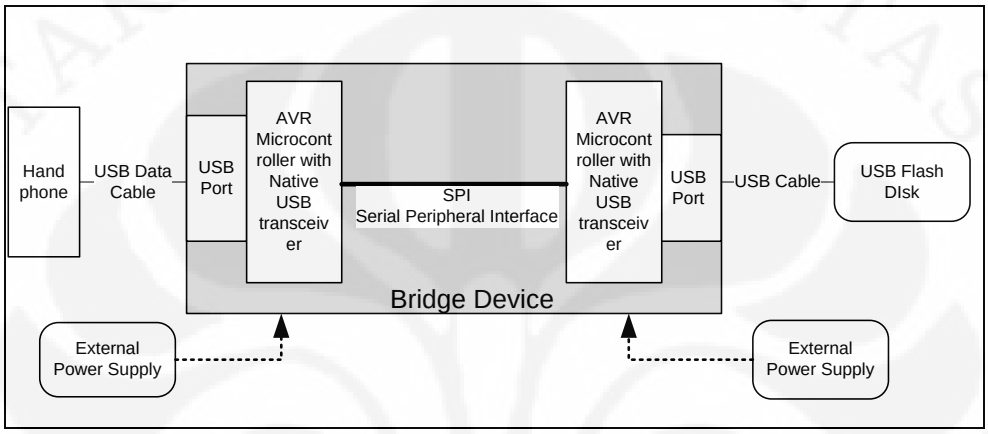
CLDC adalah sebuah konfigurasi yang terdapat di dalam J2ME untuk alat-alat yang memiliki keterbatasan ruang memori atau RAM (kurang dari 512 KB) dan pada umumnya dioperasikan dengan menggunakan baterai, serta memiliki bandwidth kecil. Terdapat tiga buah paket dari J2SE yang didukung oleh CLDC, yaitu sebagai berikut [11]:

- java.lang
- java.io
- java.util

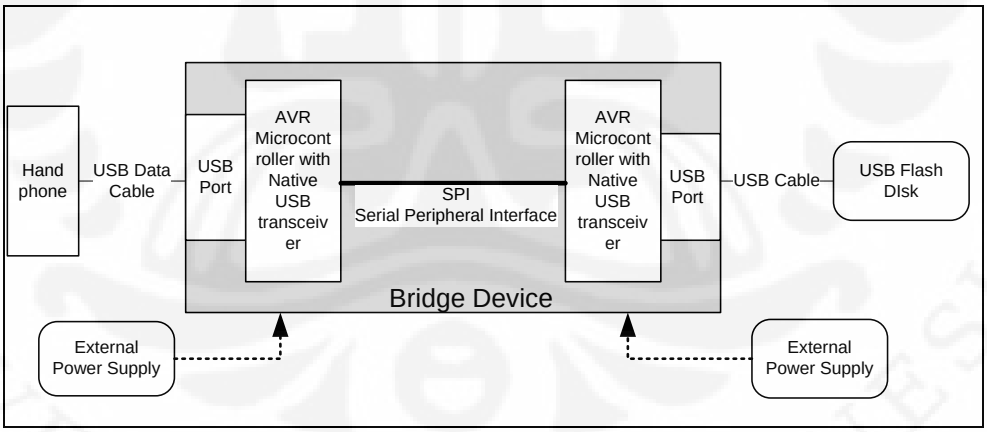
BAB 3
PERANCANGAN USB OTG ENABLER DENGAN AT90USB1287

3.1 Rancangan Perangkat Keras

Rancangan menggunakan minimum sistem mikrokontroler AT90USB1287 sehingga sistem perangkat keras menjadi sederhana.

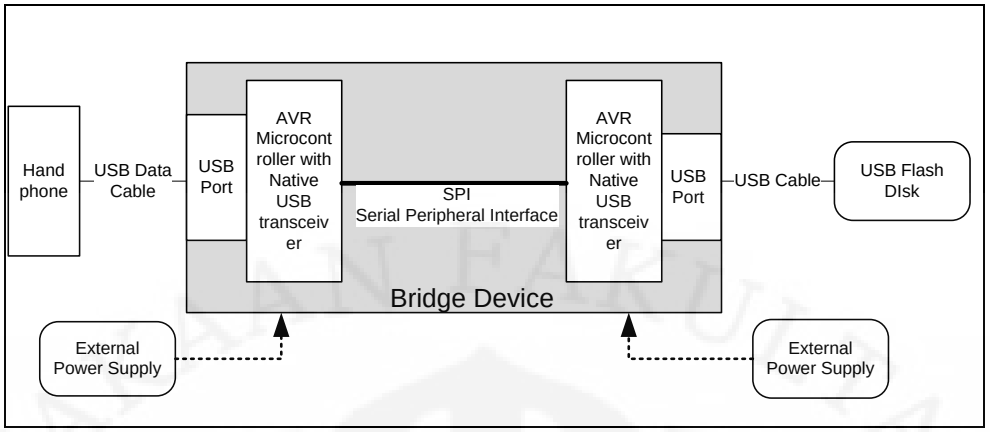


Gambar 3.1 menunjukkan skema perancangan sistem. Perangkat ini akan menggunakan sumber daya dari sumber tersendiri yaitu, menggunakan baterai 9V ataupun adaptor dengan tegangan 7,5-12 V dan arus 800mA.



Gambar 3.1 Skema Perancangan

Perangkat keras akan dibuat menjadi dua buah modul yang identik dan saling terhubung melalui antarmuka SPI (Serial Peripheral Interface) dengan menggunakan kabel. Masing-masing modul dapat terhubung dengan satu buah piranti USB, baik kabel data telepon genggam ataupun *flash drive*. Sehingga, dengan mengacu pada

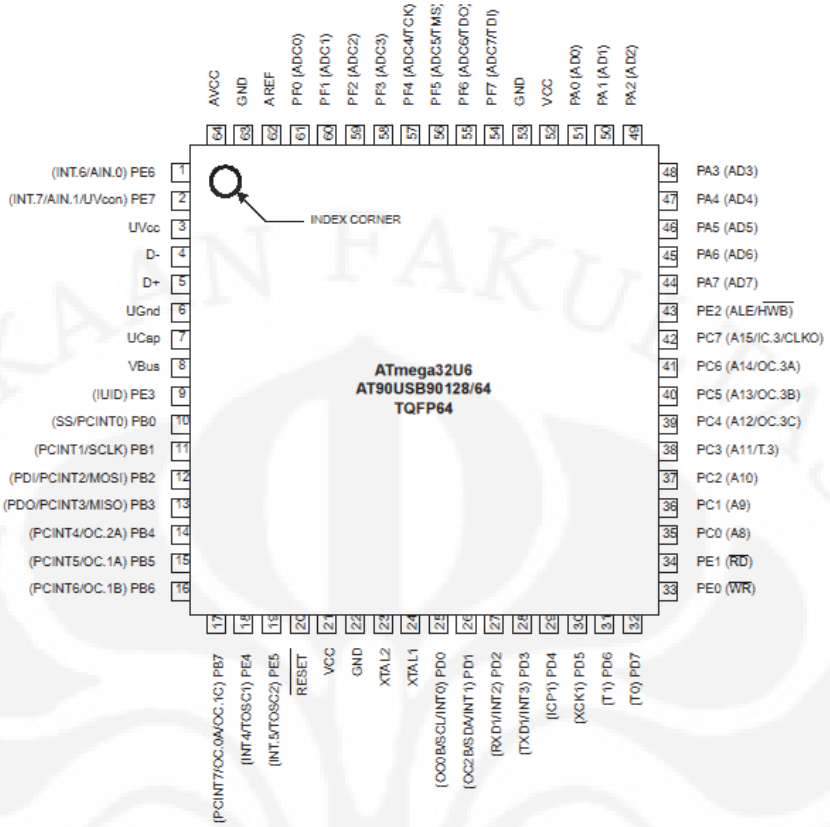


Gambar 3.1, kabel data beserta telepon genggam dapat terhubung dengan modul pada sisi kanan dan *flash drive* terhubung dengan modul pada sisi kiri.

Telepon genggam berfungsi sebagai *master* yang memberikan perintah pada modul sehingga pada salah satu modul harus terhubung dengan telepon genggam dan modul lainnya terhubung dengan *flash drive*. Bila kedua modul terhubung dengan *flash drive* saja maka USB OTG Enabler ini tidak akan berfungsi karena tidak ada yang berperan sebagai *master*. Kedua modul dapat terhubung dengan telepon genggam dengan syarat salah satu telepon genggam menggunakan mode *Mass Storage* sehingga oleh sistem akan dianggap sebagai *flash drive*.

3.2.1 Mikrokontroler

Mikrokontroler yang digunakan harus mempunyai fitur USB Serial Interface Engine sehingga tidak diperlukan lagi emulasi yang menghabiskan sumberdaya prosesor dan lebih fokus terhadap software USB *Host*. Selain itu mikrokontroler tersebut harus mempunyai SRAM > 2 KB karena untuk mengakses FAT dibutuhkan cukup banyak memori minimal 2 KB sedangkan program yang akan dibuat tidak hanya berkutat pada FAT saja.



Gambar 3.2 *Pinout* mikrokontroler AT90USB1287

Mikrokontroler AVR AT90USB647 atau AT90USB1287 sudah mempunyai fitur-fitur yang disyaratkan. Mikrokontroler yang akan digunakan berjumlah dua buah karena jumlah USB *transceiver* yang dimiliki masing-masing unit mikrokontroler hanya satu sedangkan yang dibutuhkan dua buah.

Pada Gambar 3.2 konektor USB terhubung dengan pin 3,4,5 dan 6 AT90USB1287 yang merupakan pin keluaran dari USB *controller* internal. Untuk hubungan antar mikrokontroler digunakan antarmuka SPI (pin 10,11,12, dan 13).

SPI merupakan antarmuka pengiriman data secara serial dan sinkron dengan sistem *master* dan *slave*. Mempunyai kecepatan hingga setengah kecepatan *clock master*. SPI terdiri dari minimal empat pin yaitu, pin SCLK sebagai *clock* yang dikeluarkan oleh *master* sebagai sarana sinkronisasi pengiriman, pin MOSI sebagai keluaran data yang dikirim oleh *master*, pin MISO sebagai masukan dari data yang dikirim oleh *slave*, dan pin SS sebagai masukan bahwa *master* akan melakukan pengiriman data.

3.2.2 Telepon Genggam

Telepon genggam yang digunakan harus sudah mendukung Java MIDP 2.0, mempunyai dokumentasi yang terbuka, mempunyai dukungan terhadap Java API JSR75 atau kompatibel, dan mengizinkan akses terhadap serial port. Sebagai perangkat yang diuji dipilih Siemens CX75. Java harus didukung karena akan dibuat piranti lunak sebagai pengelola berkas (*File Manager*).

3.2.3 USB Data Cable

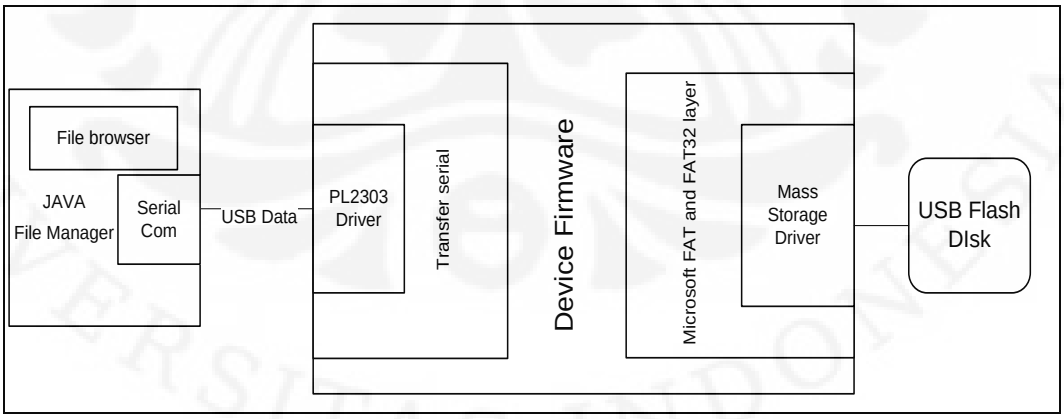
Kabel data USB yang digunakan adalah model DCA-510 USB to Serial Data Cable produksi Siemens Mobile ataupun kabel data jenis imitasi yang kompatibel. Kabel data ini menggunakan IC Prolific PL2303 USB to RS232 Converter.

3.2.4 Flash Disk

Flash disk yang didukung adalah dengan format sistem berkas (*filesystem*) FAT dengan kapasitas <=2GB. Pembatasan kapasitas ini lebih kepada keterbatasan ukuran maksimal partisi FAT pada 2048MB, sedangkan untuk ukuran lebih besar dibutuhkan format sistem berkas FAT32.

3.2 Rancangan Perangkat Lunak

Perangkat lunak akan dibagi menjadi dua blok utama yaitu *firmware* pada USB OTG Enabler dan Java pada handphone.



Gambar 3.3 Skema Arsitektur Perangkat Lunak

Firmware mikrokontroler akan dibuat menggunakan bahasa C agar modular dan fleksibel. Seperti terlihat pada Gambar 3.3, bagian *firmware* yang berinteraksi langsung dengan hardware luar adalah pada level driver. Kemudian pada level

lebih dalam adalah implementasi dari perintah logikal seperti serial/AT command pada kabel data dan FAT pada USB flash disk.

Software JAVA pada handphone akan berupa *file manager* sehingga fungsi-fungsi yang ada harus berupa browser sistem file internal serta komunikasi serial dengan USB OTG Enabler yang dibuat.

3.2.1 Library LUFA

LUFA (The Lightweight USB Framework for AVRs) adalah sebuah *library* USB terbuka untuk mikrokontroler AVR yang mempunyai kontroler USB. Didesain untuk menyediakan kemudahan penggunaan, rangka kerja yang kaya fitur untuk pengembangan piranti USB baik sebagai *peripheral* maupun sebagai *host*. Ditulis oleh Dean Camera dan dirilis dibawah lisensi MIT dengan target *compiler* AVR-GCC dan menggunakan beberapa ekstensi khusus GCC.[13]

3.2.2 Library LUFA

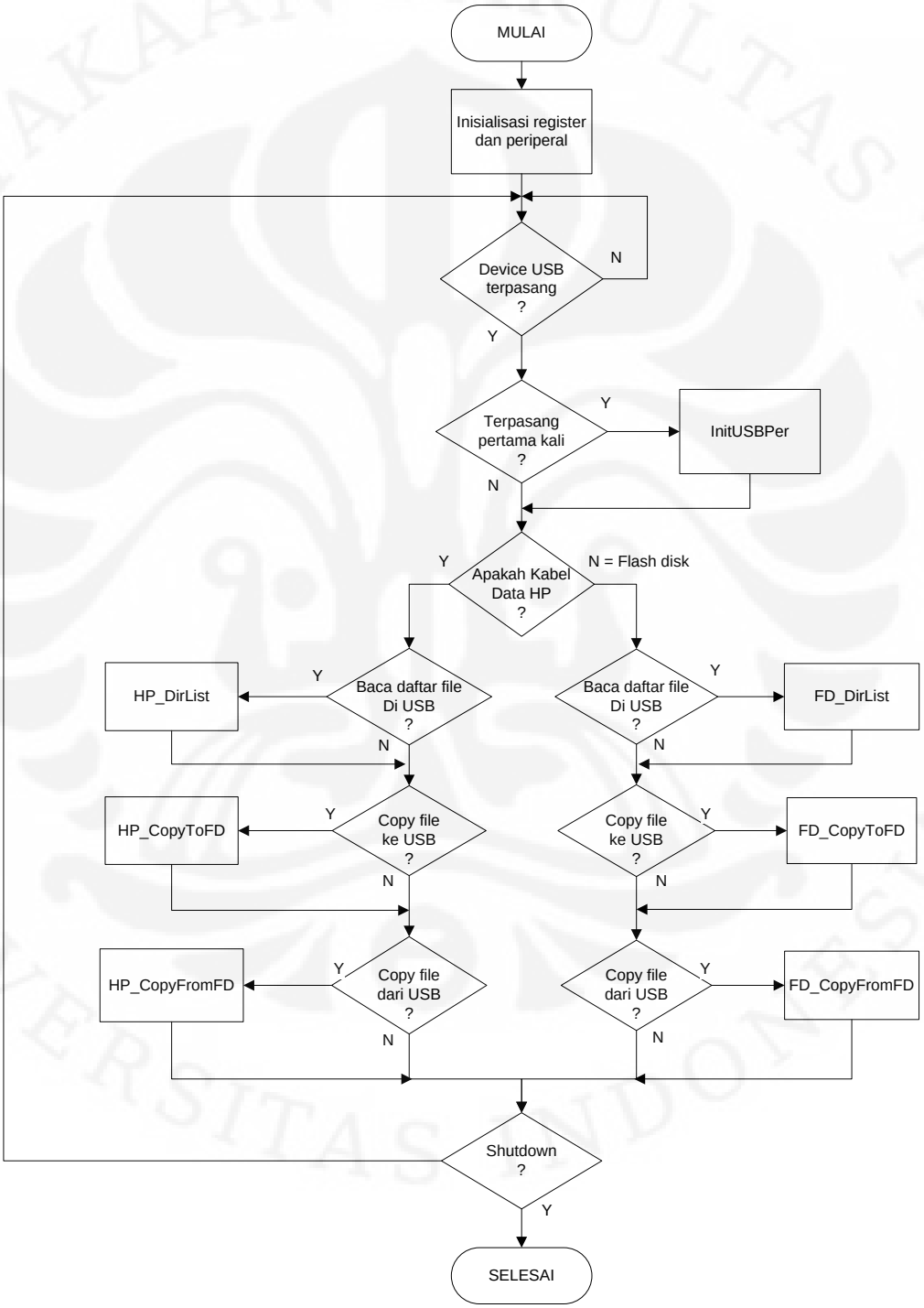
FatFS adalah modul generik sistem file FAT untuk sistem *small embedded*. FatFS ditulis dengan standard ANSI C dan benar-benar terpisah dari lapisan akses I/O disk sehingga independen terhadap arsitektur perangkat keras. Dengan kata lain FatFS berfungsi sebagai pustaka *middleware* yang mengabstraksi akses FAT sehingga dapat dipergunakan pada berbagai macam mikrokontroler seperti AVR, 8051, PIC, Z80 dan sebagainya tanpa harus mengubah kode sumber FatFS.

FatFS menyediakan fungsi-fungsi standard akses file pada C seperti f_open(), f_close(), f_write(), f_read() dan sebagainya. Karena berupa *middleware* yang terpisah dari lapisan akses I/O disk, FatFS masih membutuhkan akses level bawah untuk membaca atau menulis fisik disk dan untuk mendapatkan waktu penulisan. Akses level bawah ini lah yang dibuat pada modul *Mass Storage Driver* seperti terlihat pada Gambar 3.3. [14]

3.2.3 Algoritma Perangkat Lunak

Gambar 3.4 memperlihatkan alur program utama. Setelah inisialisasi, program akan terus mengecek apakah ada perangkat USB yang terhubung. Bila ada perangkat USB yang terhubung, program akan melakukan *setup* perangkat

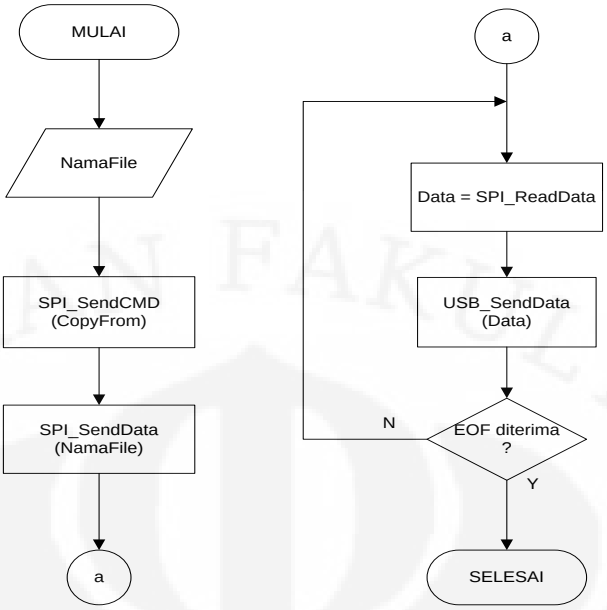
tersebut untuk mengetahui apakah perangkat tersebut didukung. Bila perangkat tersebut merupakan perangkat yang didukung, program akan menunggu perintah dari pengguna untuk menyalin data ataupun menampilkan daftar file. Algoritma dirancang universal sehingga setiap modul dapat terhubung dengan kabel data ataupun *flash drive*.



Gambar 3.4 Algoritma Program Utama

Pada Gambar 3.4 terdapat beberapa sub fungsi yang dipanggil apabila terjadi kondisi tertentu, berikut adalah fungsi-fungsi tersebut:

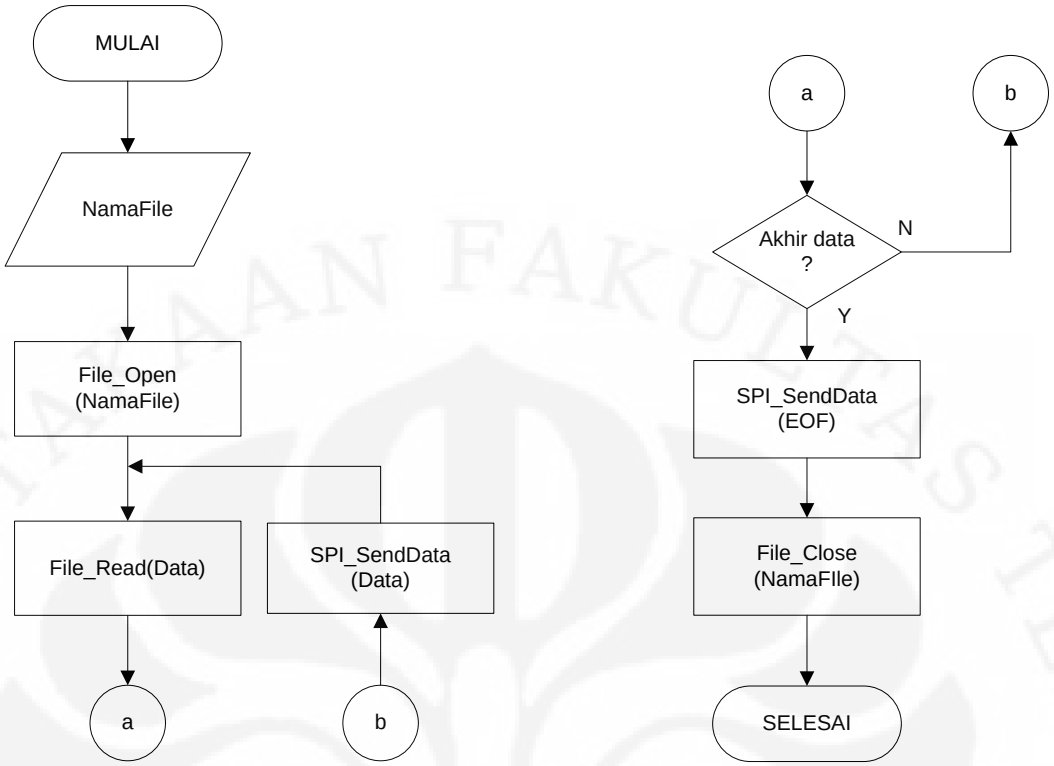
- **InitUSBPer**, adalah fungsi inisialisasi perangkat USB yang terpasang sesuai protokol USB dan memberi tanda (*flag*) bahwa perangkat yang terpasang adalah kabel data atau USB *flash drive.*
- **HP_DirList**, adalah fungsi yang dijalankan oleh mikrokontroler pada sisi telepon genggam untuk memberikan perintah kepada mikrokontroler *flash drive* untuk membaca daftar file yang tersedia.
- **FD_DirList**, adalah fungsi yang dijalankan oleh mikrokontroler sisi *flash drive* untuk membaca daftar file yang tersedia pada *flash drive*.
- **HP_CopyToFD**, adalah fungsi yang dijalankan oleh mikrokontroler sisi telepon genggam untuk memberikan perintah kepada mikrokontroler sisi *flash drive* untuk menyalin data dari telepon genggam ke *flash drive*
- **FD_CopyToFD**, adalah fungsi yang dijalankan oleh mikrokontroler sisi *flash drive* untuk menyalin data dari telepon genggam ke *flash drive*.
- **HP_CopyFromFD**, adalah fungsi yang dijalankan oleh mikrokontroler sisi telepon genggam untuk memberikan perintah kepada mikrokontroler sisi *flash drive* untuk menyalin data dari *flash drive* ke telepon genggam.
- **FD_CopyFromFD**, adalah fungsi yang dijalankan oleh mikrokontroler sisi *flash drive* untuk menyalin data dari *flash drive* ke telepon genggam.



Gambar 3.5 Algoritma Salin Data ke HP pada Sisi HP

Gambar 3.5 menunjukan alur fungsi penyalinan data ke telepon genggam. Fungsi akan mengirimkan perintah copy dan nama file yang akan disalin melalui SPI. Bila tidak ada masalah maka pada jalur SPI akan terdapat data file yang akan disalin, data ini kemudian langsung dikirimkan ke telepon genggam. Fungsi-fungsi yang digunakan:

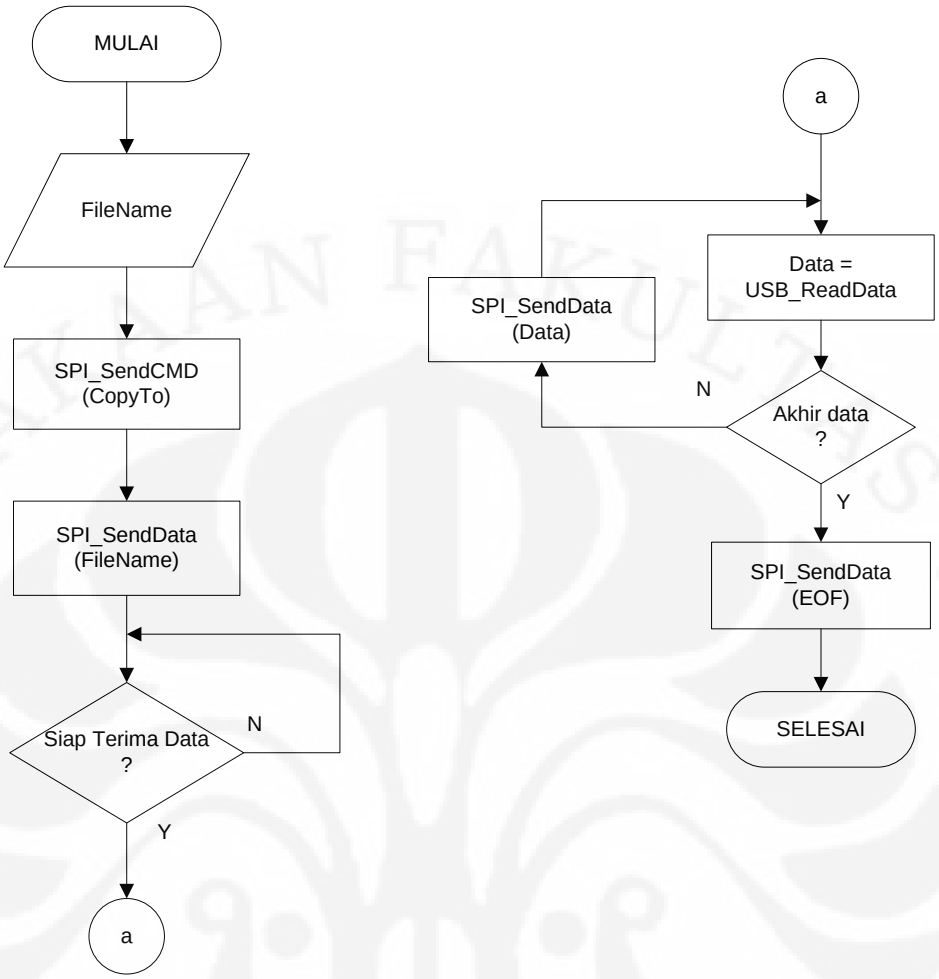
- **SPI_SendCMD**, adalah fungsi untuk mengirim perintah melalui jalur komunikasi SPI.
- **SPI_SendData**, adalah fungsi untuk mengirim data, baik data file maupun data parameter perintah, melalui jalur komunikasi SPI.
- **SPI_ReadData**, adalah fungsi untuk membaca data yang dikirm melalui jalur komunikasi SPI.
- **USB_SendData**, adalah fungsi untuk mengirimkan data melalui kabel data USB.



Gambar 3.6 Algoritma Salin Data ke HP pada Sisi Flash Disk

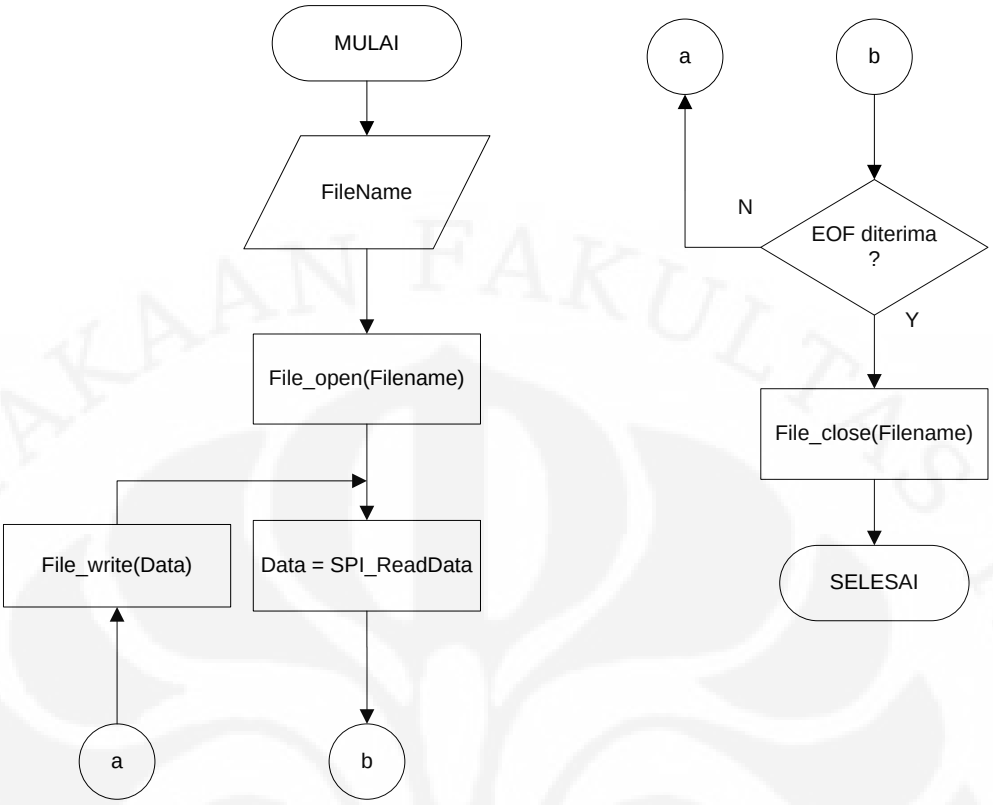
Program menerima data nama file yang akan disalin kemudian membaca data file tersebut dari *flash drive*. Data ini kemudian dikirim melalui SPI untuk disalin menuju telepon genggam. Fungsi-fungsi yang digunakan:

- **File_Open**, adalah fungsi untuk membuka file yang akan disalin sehingga siap untuk dibaca datanya.
- **File_Read**, adalah fungsi untuk membaca data dari file yang sudah dibuka. Data yang dibaca diletakan pada variabel Data yang diberikan.
- **File_Close**, adalah fungsi untuk menutup file yang telah selesai digunakan dan memastikan semua data telah ditulis sempurna. Waktu penulisan yang ada pada tabel FAT ditulis pada fungsi ini.



Gambar 3.7 Algoritma Salin Data ke *Flash Drive* pada Sisi HP

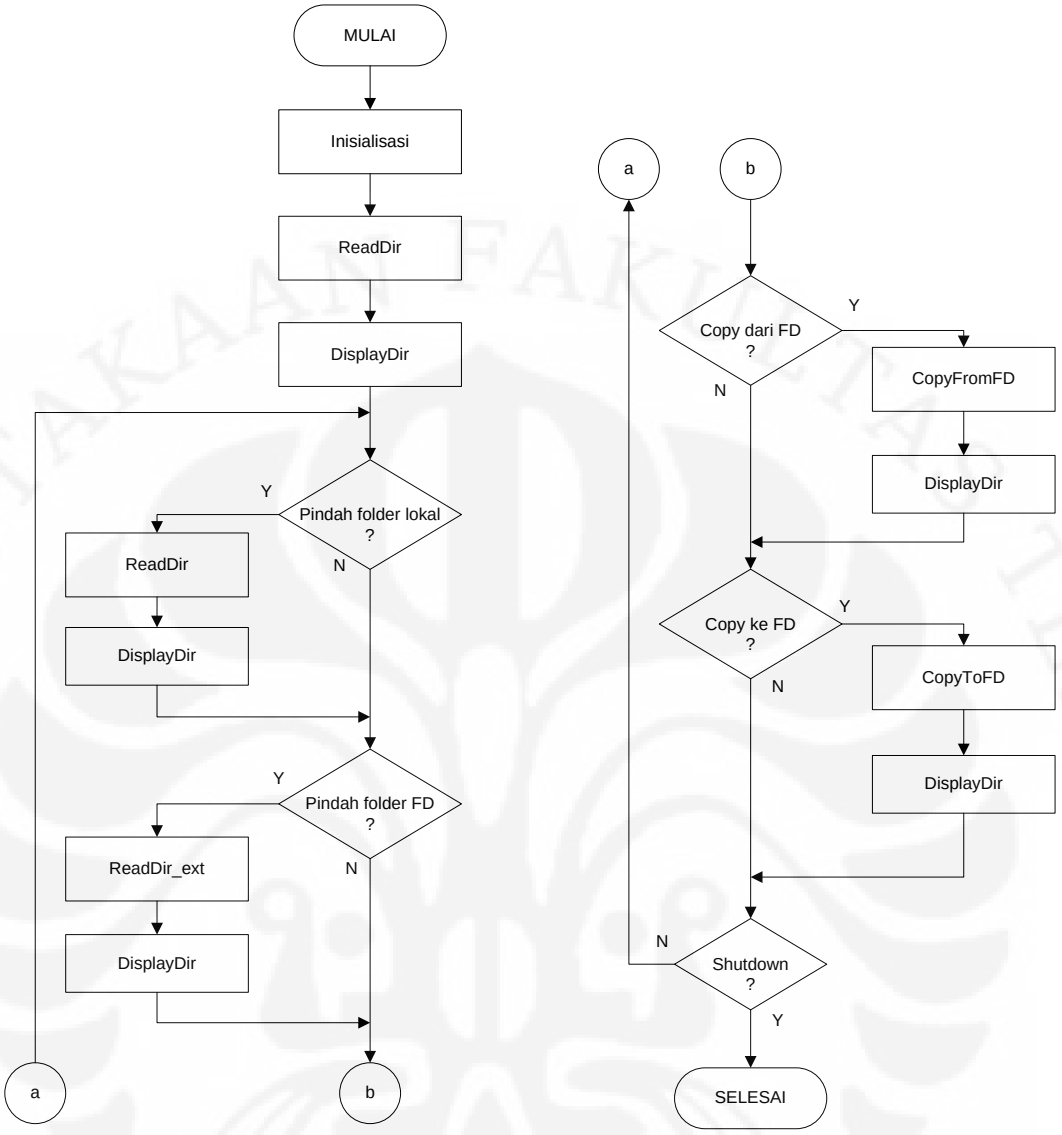
Alur fungsi salin data ke *flash drive* seperti pada Gambar 3.7 dimulai dengan membaca nama file dan dikirimkan melalui SPI. Bila *flash drive* telah siap menulis data maka data file akan diminta dari telepon genggam kemudian dikirim melalui SPI. Sinyal EOF (*End Of File*) dikirim untuk menyatakan bahwa data terakhir telah dikirim.



Gambar 3.8 Algoritma Salin Data ke *Flash Drive* pada sisi *Flash Drive*

Pada Gambar 3.8 proses penyalinan data ke *flash drive* akan dilaksanakan dengan alur pembacaan nama file dan membuka file tersebut. Data dibaca pada jalur SPI untuk ditulis pada file. Bila sinyal EOF diterima maka data file terakhir sudah dikirim dan file siap ditutup kembali. Fungsi yang digunakan:

- **File_write**, adalah fungsi untuk menuliskan data yang terdapat pada variabel yang diberikan, secara fisik. File yang digunakan harus sudah dibuka terlebih dahulu dan saat telah selesai harus dilakukan penutupan file.



Gambar 3.9 Algoritma Aplikasi *File Manager* Java

Gambar 3.10 menunjukan algoritma aplikasi *file manager*. Aplikasi akan menginisialisasi variabel kemudian membaca daftar direktori dan file lalu menampilkannya pada layar LCD telepon genggam. Aplikasi akan menunggu masukan perintah dari pengguna yaitu pindah direktori lokal, pindah direktori *flash drive*, salin data dari telepon genggam, salin data dari *flash drive* atau pun mematikan aplikasi. Fungsi-fungsi yang digunakan:

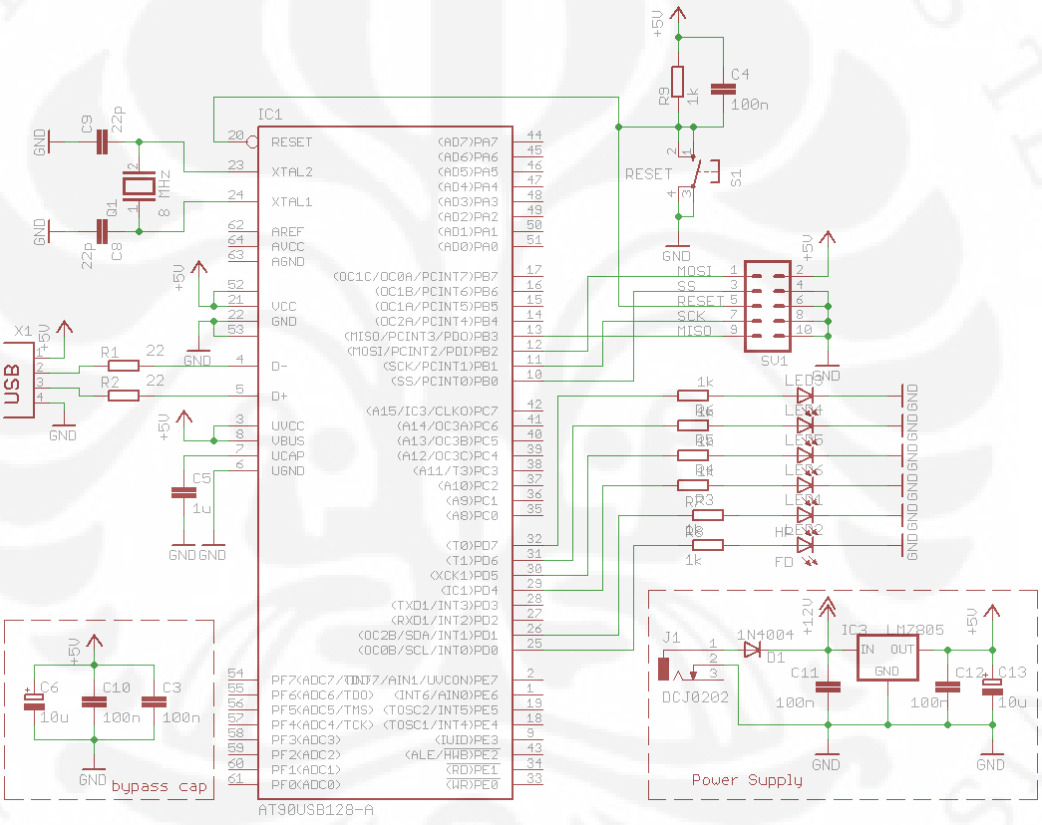
- **ReadDir**, adalah fungsi untuk membaca daftar direktori dan file lokal pada telepon genggam untuk kemudian ditampilkan oleh fungsi DisplayDir.

- **ReadDir_ext**, adalah fungsi untuk membaca direktori dan file pada *flash drive* untuk kemudian ditampilkan oleh fungsi DisplayDir.
- **DisplayDir**, adalah fungsi untuk menampilkan daftar direktori, baik lokal maupun pada *flash drive,* pada layar telepon genggam.
- **CopyFromFD**, adalah fungsi untuk memberikan perintah ke perangkat USB OTG Enabler berupa perintah salin data dari *flash drive* ke telepon genggam.
- **CopyToFD**, adalah fungsi untuk memberikan perintah ke perangkat USB OTG Enabler berupa perintah salin data dari telepon genggam ke *flash drive*.

BAB 4
UJI COBA DAN ANALISA USB OTG ENABLER DENGAN MIKROKONTROLER AT90USB1287

4.1 Implementasi Perangkat Keras

Perangkat keras dibuat menjadi 2 buah modul yang identik dan saling dihubungkan melalui jalur SPI dengan menggunakan kabel pita. Jalur UART RS232 dihubungkan dengan PC sebagai sarana *debugging*.



Gambar 4.1 Rangkaian sistem minimum AT90USB1287

Rangkaian minimum sistem yang digunakan dapat dilihat pada Gambar 4.1. Pin SPI yang digunakan untuk menghubungkan antar mikrokontroler dihubungkan dengan soket 10pin yang berfungsi juga sebagai soket ISP (In-System Programming).

Gambar 4.2 Rangkaian perangkat keras OTG Enabler

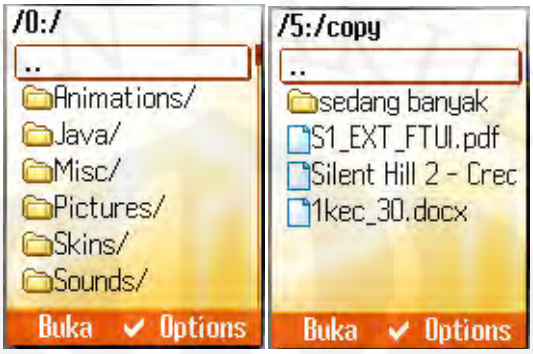
Pada Gambar 4.2, terlihat modul tidak identik namun secara rangkaian identik. Perbedaan terdapat pada salah satu modul menggunakan 2 lapisan PCB yaitu lapisan atas dan lapisan bawah sedangkan modul lain menggunakan hanya jalur bawah saja. Selain itu beberapa LED indikator dan tombol yang terpisah pada modul versi awal, digabungkan pada modul baru.

4.2 Implementasi Perangkat Lunak

Aplikasi *file manager* yang dikembangkan dengan bahasa Java ini, mempunyai tampilan sederhana. Pengembangan lebih diarahkan untuk fungsionalitas dasar sebagai *file manager* saja.

Gambar 4.3 Tampilan awal *file manager*

Komponen *propietary* com.siemens.mp.io.File dari Siemens digunakan pada aplikasi ini untuk mengakses memori internal telepon genggam. Komponen ini merupakan komponen yang setara dengan javax.microedition.io.File yang terdapat pada JSR75 Java MIDP 2.0.



(a) (b)

Gambar 4.4 Tampilan daftar file dan direktori

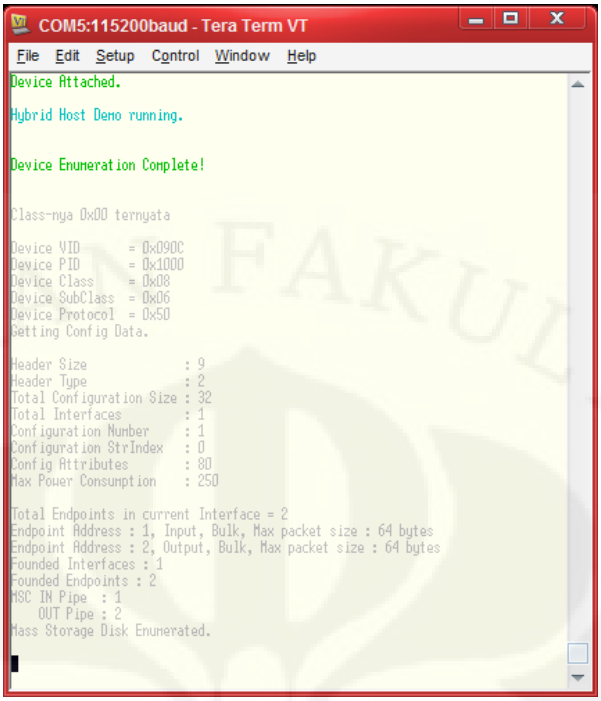
Gambar 4.3 merupakan tampilan awal aplikasi berisi daftar *drive* yang dapat diakses. *Drive* "5:/" adalah *drive* yang dipakai sebagai akses terhadap USB *flash drive* yang terhubung dengan OTG Enabler ini. Ketika "0:/" dipilih maka akan ditampilkan daftar file pada memori internal seperti terlihat pada Gambar 4.4a. Seangkan ketika "5:/" dipilih maka akan ditampilkan daftar file pada USB *flash drive* seperti terlihat pada Gambar 4.4b. Penghitungan waktu transfer data ditampilkan dalam bentuk *pop-up* seperti pada Gambar 4.5.

Gambar 4.5 Tampilan penghitung waktu transfer

Driver PL2303 yang terdapat perangkat lunak (*firmware*) pada OTG Enabler menggunakan sebagian kode sumber dari kernel Linux dengan modifikasi dibeberapa bagian menyesuaikan arsitektur AVR yang digunakan.

Gambar 4.6 Tampilan keluaran deteksi kabel data PL2303

Perangkat lunak menampilkan keluaran melalui jalur komunikasi UART RS232 yang terhubung dengan PC. Keluaran ini dimaksudkan untuk mengamati proses kerja dan deteksi perangkat sehingga bila terjadi kesalahan dapat diketahui letak kesalahan tersebut. Tampilan saat mendeteksi USB *flash drive* terlihat pada Gambar 4.7 dan saat mendeteksi kabel data PL2303 terlihat pada Gambar 4.6.



Gambar 4.7 Tampilan keluaran deteksi *flash drive*

4.3 Pengambilan Data

Untuk mengetahui performa USB OTG Enabler ini, maka dibutuhkan pengujian dengan cara pengambilan beberapa data. Pengujian yang dilakukan adalah:

1. **Kompabilitas dengan berbagai perangkat USB *Flash disk***

   Perangkat USB OTG Enabler akan dihubungkan dengan beberapa merk dan kapasitas flash disk yang berbeda. Dalam pengujian digunakan tiga buah *flash drive* dengan spesifikasi sebagai berikut:

Tabel 4.1 Ringkasan spesifikasi perangkat uji

Nama	Kapasitas	Versi USB	VID & PID	Endpoint			
				Jumlah	arah	tipe	Ukuran paket
Kabel Data PL2303	-	1.1	067B - 2303	3	2 In 1 Out	2 Bulk 1 Int	64
Easy Disk	1 GB	2.0	090C - 1000	2	1 In 1 Out	2 Bulk	512
A-DATA C003	2 GB	2.0	0419 - 0100	3	2 In 1 Out	2 Bulk 1 Int	512
Easy MP3 EM120X	SD 256 MB	1.1	125F – C03A	3	2 In 1 Out	2 Bulk 1 Int	64

2. **Kecepatan transfer data berukuran besar**

   Pengujian dilakukan dengan cara menyalin data berukuran besar dari telepon genggam ke flash disk dan sebaliknya kemudian dicatat waktu yang dibutuhkan. Data yang digunakan dalam percobaan adalah file audio dengan format MP3 berukuran 1498929 byte.

3. **Kecepatan transfer data berukuran kecil**

   Pengujian dilakukan dengan cara menyalin data berukuran kecil dari telepon genggam ke flash disk dan sebaliknya kemudian dicatat waktu yang dibutuhkan. Data yang dipergunakan dalam percobaan adalah file teks dengan format DOCX berukuran 54715 byte.

4. **Kecepatan transfer data berukuran kecil namun berjumlah banyak**

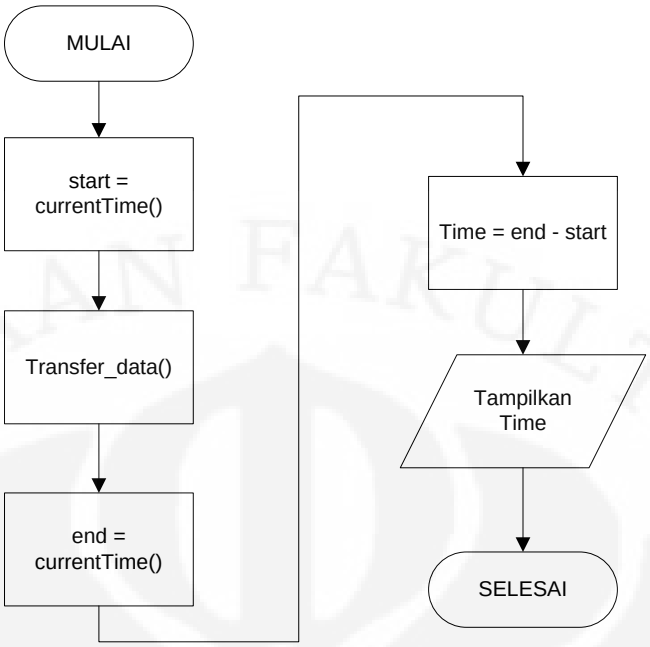
   Pengujian dilakukan dengan cara menyalin file sejumlah 30 buah, dengan total kapasitas mendekati dengan pengujian poin kedua, dari telepon genggam ke flash disk dan sebaliknya kemudian dicatat waktu yang dibutuhkan. Data yang dipergunakan adalah file teks dan gambar berukuran antara 10240 – 97323 byte dengan total kapasitas 1432499 byte.

5. **Kecepatan transfer data dengan variasi kecepatan SPI dan PL2303**

   Pengujian dilakukan dengan menyalin file-file seperti pada poin 2 - 4 namun kecepatan transfer SPI dan PL2303 diubah-ubah. Kombinasi kecepatan SPI dan PL2303 yang diujikan adalah:

   - Kecepatan SPI 500 kHz dan kecepatan PL2303 115200 bps,
   - Kecepatan SPI 125 kHz dan kecepatan PL2303 115200 bps,
   - Kecepatan SPI 500 kHz dan kecepatan PL2303 57600 bps.

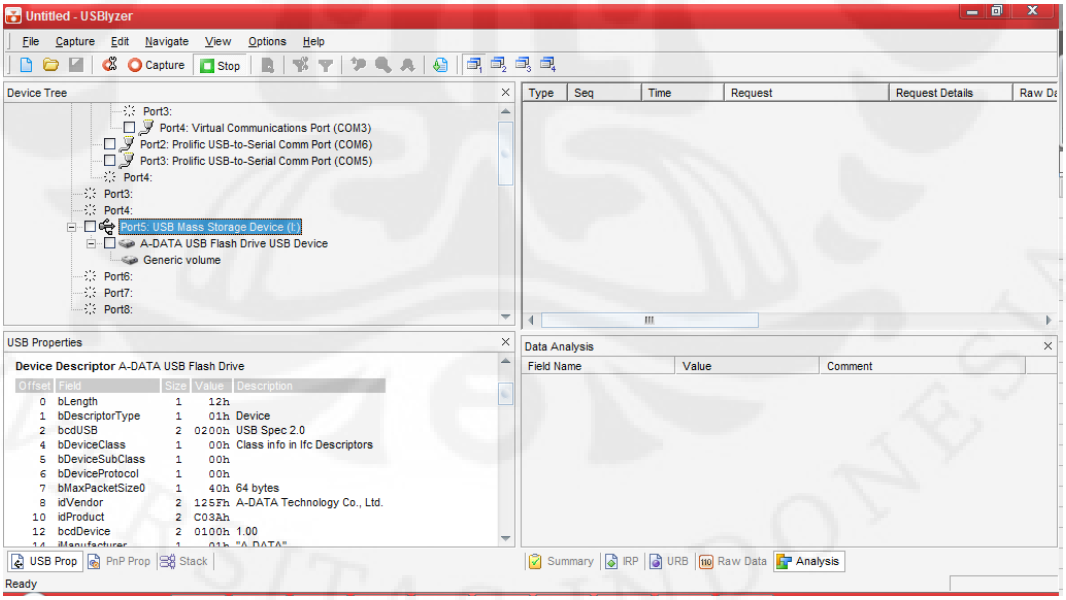
Pada setiap poin pengujian dilakukan pengambilan data sebanyak lima kali kemudian hasil catatan waktu akan dirata-ratakan. Penghitungan waktu dilakukan oleh apliasi *file manager* yang dibuat dengan resolusi hingga 1 ms. Algoritma penghitungan waktu dapat dilihat pada Gambar 4.8.



Gambar 4.8 Algoritma penghitungan waktu transfer data

4.4 Data Percobaan

Spesifikasi perangkat yang diuji didapatkan dengan menggunakan perangkat lunak USBlyzer (Gambar 4.9) yang berjalan pada sistem operasi Windows.



Gambar 4.9 Perangkat lunak USBlyzer

4.4.1 Kompabilitas Perangkat

Pengujian dilakukan dengan melihat apakah perangkat USB *flash drive* dapat dideteksi dan sesuai dengan spesifikasi yang diperoleh dengan USBlyzer.

Kemudian dilakukan pengetesan menampilkan daftar file. Dilakukan pula pengujian akses tulis dengan cara menyalin file dari telepon genggam ke USB *flash drive* dan pengujian akses baca dengan cara menyalin file dari USB *flash drive* ke telepon genggam. Hasil pengujian ditampilkan pada Tabel 4.2.

Tabel 4.2 Hasil pengujian kompabilitas

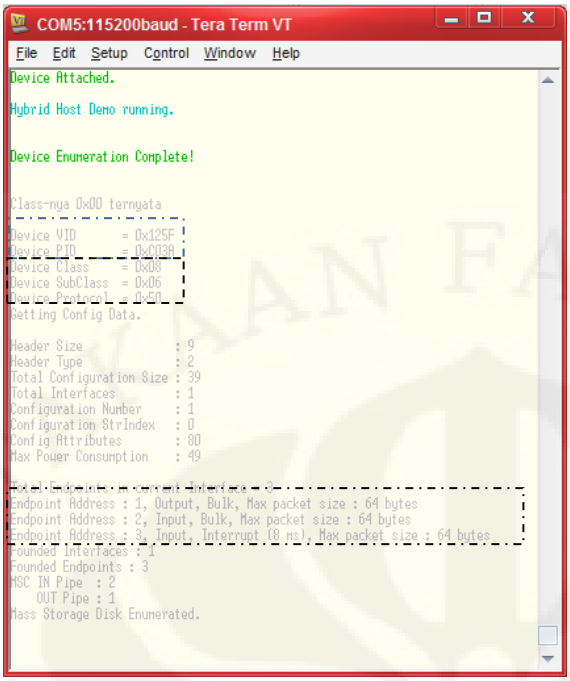
Nama	Deteksi (s)	Tampilkan file (s)	Akses Tulis (s)	Akses Baca (s)
Easy Disk	4,10	0,78	12,69	19,13
Easy MP3 EM120X	4,13	0,78	13,86	20,43
A-DATA C003	4,38	1,50	13,91	19,88

Tampilan saat mendeteksi *storage media* Easy Disk dapat dilihat pada Gambar 4.10. VID & PID yang terdeteksi sama seperti yang tercantum pada Tabel 4.1 yaitu 0x090C & 0x1000.

Easy Disk	
Vendor ID	090C
Product ID	1000
Class	08
Sub Class	06
Protocol	50
Endpoint 1	IN, Bulk, 512 byte
Endpoint 2	OUT, Bulk, 512 byte

Gambar 4.10 Tampilan deteksi Easy Disk

Tampilan saat mendeteksi *storage media* A-Data dapat dilihat pada Gambar 4.11. VID & PID yang terdeteksi sama seperti yang tercantum pada Tabel 4.1 yaitu 0x125F & 0xC03A.



A-Data C003	
Vendor ID	125F
Product ID	C03A
Class	08
Sub Class	06
Protocol	50
Endpoint 1	OUT, Bulk, 512 byte
Endpoint 2	IN, Bulk, 512 byte
Endpoint 3	IN, Interrupt, 64 byte

Gambar 4.11 Tampilan saat deteksi A-Data

Tampilan saat mendeteksi *storage media* Easy Disk dapat dilihat pada Gambar 4.12. VID & PID yang terdeteksi sama seperti yang tercantum pada Tabel 4.1 yaitu 0x0419 & 0x0100.

Easy MP3 EM120X	
Vendor ID	0419
Product ID	0100
Class	08
Sub Class	06
Protocol	50
Endpoint 1	IN, Interrupt, 8 byte
Endpoint 2	IN, Bulk, 64 byte
Endpoint 3	OUT, Bulk, 64 byte

Gambar 4.12 Tampilan saat deteksi Easy MP3

4.4.2 Kecepatan Transfer Data

Pengujian dilakukan dengan 3 kombinasi kecepatan dan 2 arah pengiriman data, yaitu dari telepon genggam (HP) ke *flash drive* (FD) dan dari *flash drive* (FD) ke telepon genggam (HP). Setiap kombinasi kecepatan dan arah diambil data sebanyak masing-masing lima kali dan hasilnya dirata-ratakan. Hasil pengujian kecepatan transfer pada perangkat Easy Disk dapat dilihat pada Tabel 4.3. Sedangkan waktu transfer pada perangkat Easy Disk dapat dilihat pada Tabel 4.4. Data waktu dan kecepatan transfer yang didapat sudah termasuk data overhead yang dikirim yang berupa nama file. Hal ini dikarenakan perhitungan waktu sudah dimulai sejak perintah dikirim sedangkan setiap perintah yang dikirim harus mempunyai parameter lain yaitu nama file dengan panjang maksimal 255 byte.

Tabel 4.3 Kecepatan transfer rata-rata Easy Disk

	PL2303 = 115200 SPI = 500000		**PL2303 = 115200 SPI = 125000**		**PL2303 = 57600 SPI = 500000**	
	Bps	**kBps**	**Bps**	**kBps**	**Bps**	**kBps**
(HP ke FD)						
Kecil (54715 byte)	4315,75	4,21	3706,12	3,62	3190,32	3,12
Kecil banyak (1432499)	4164,00	4,07	3942,25	3,85	3390,47	3,31
Besar (1498929)	4608,88	4,50	3935,48	3,84	3399,49	3,32
(FD ke HP)						
Kecil (54715 byte)	2860,49	2,79	2187,75	2,14	1827,04	1,78
Kecil banyak (1432499)	2848,79	2,78	2194,85	2,14	1831,57	1,79
Besar (1498929)	2902,27	2,83	2212,37	2,16	1841,47	1,80

Tabel 4.4 Waktu transfer rata-rata Easy Disk

	PL2303 = 115200 SPI = 500000 (s)	PL2303 = 115200 SPI = 125000 (s)	PL2303 = 57600 SPI = 500000 (s)
(HP ke FD)			
Kecil (54715 byte)	12,69	14,77	17,16
Kecil banyak (1432499)	344,35	363,41	422,55
Besar (1498929)	325,24	380,91	441,02
(FD ke HP)			
Kecil (54715 byte)	19,13	25,01	29,99
Kecil banyak (1432499)	502,86	652,66	782,11
Besar (1498929)	516,47	677,52	813,99

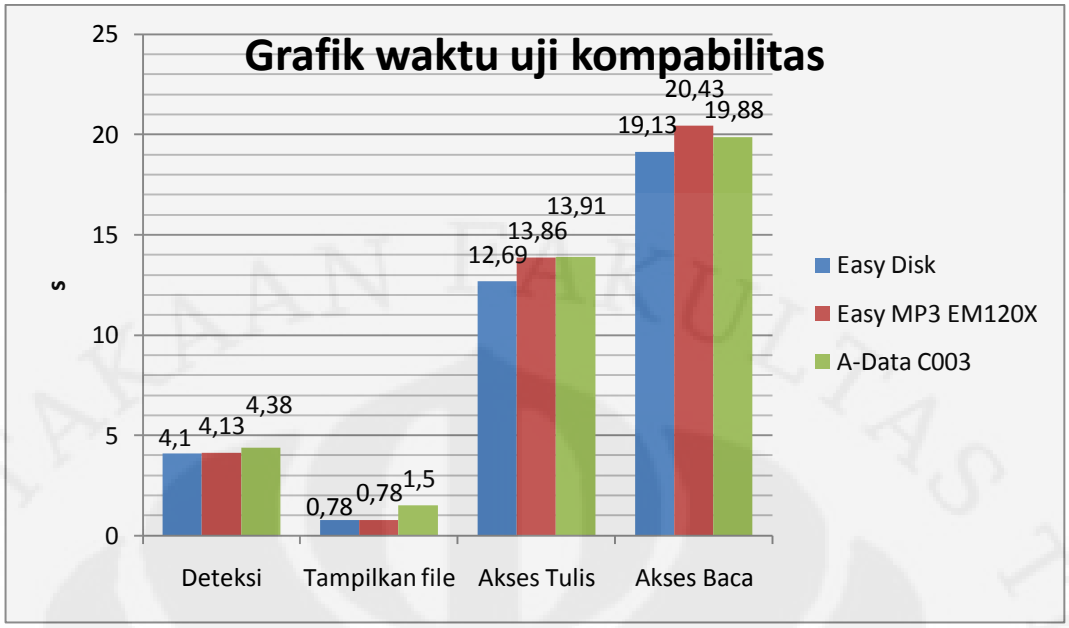
Sebagai data tambahan disertakan juga pengujian terhadap kecepatan internal USB *flash drive* maupun kecepatan memori internal telepon genggam. Hasil pengujian terdapat pada Tabel 4.5. Metode pengujian dilakukan dengan cara menyalin data dengan tujuan masih diperangkat yang sama namun berbeda folder.

Tabel 4.5 Data waktu kecepatan transfer internal rata-rata

Storage Media	Jenis file	Ukuran (byte)	waktu (s)	kecepatan (Bps)	kecepatan (kBps)
Siemens CX75 (memori internal)	kecil	54715	0,809	68028,63	66,43
	kecil banyak	1432499	18,835	76108,49	74,32
	besar	1498929	13,919	107702,63	105,18
Easy Disk	kecil	54715	0,613	89281,12	87,19
	kecil banyak	1432499	20,184	71007,55	69,34
	besar	1498929	8,905	168372,80	164,43

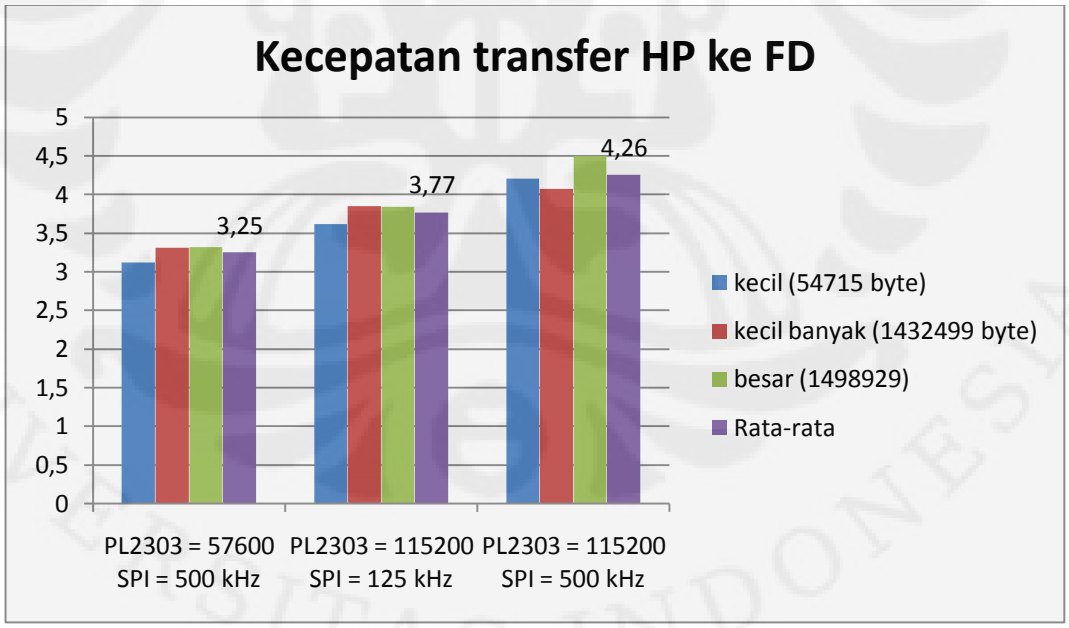
4.5 Analisa

Berdasarkan data percobaan, USB OTG Enabler ini mempunyai kompabilitas yang baik. Dari berbagai perangkat USB *Mass Storage Class* yang diuji cobakan semua dideteksi dengan baik, dengan waktu akses seperti terlihat pada Gambar 4.13.



Gambar 4.13 Grafik waktu uji kompabilitas

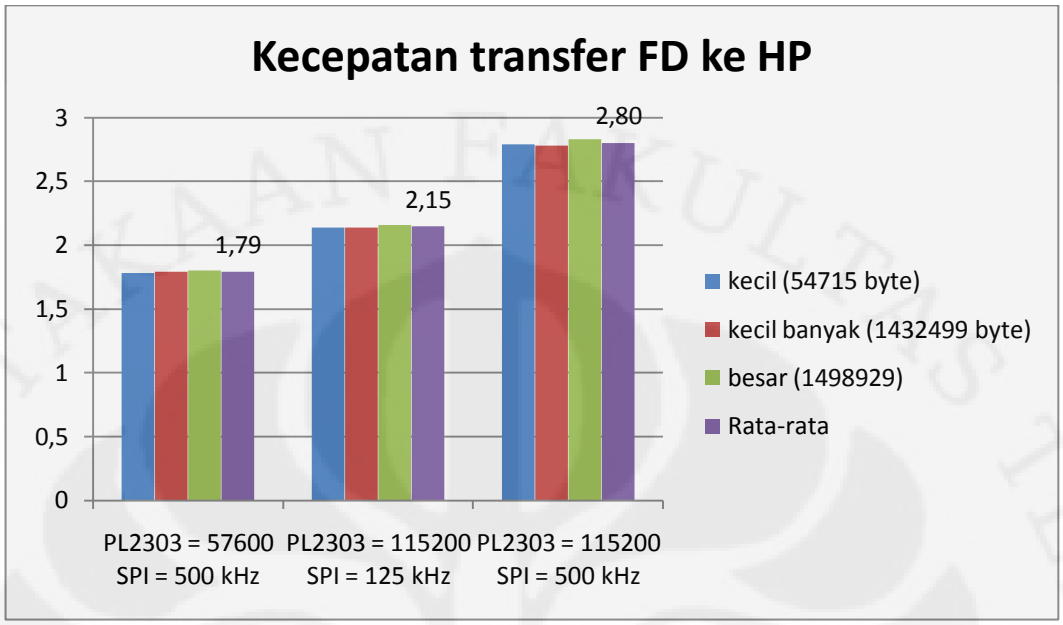
Untuk setiap kombinasi kecepatan transfer HP ke FD pada Tabel 4.3, dihitung nilai rata-ratanya sehingga didapatkan hasil seperti pada Gambar 4.14. Sedangkan untuk kecepatan transfer dari FD ke HP dapat dilihat pada Gambar 4.15.



Gambar 4.14 Grafik kecepatan transfer HP ke FD

Dengan menggunakan data rata-rata pada Gambar 4.14 dan Gambar 4.15 kemudian dirata-ratakan kembali didapatkan bahwa kecepatan akses USB OTG

Enabler adalah 3,76 kBps untuk telepon genggam ke *flash drive* dan 2,24 kBps untuk *flash drive* ke telepon genggam.



Gambar 4.15 Grafik kecepatan transfer FD ke HP

Bila dibandingkan dengan kecepatan internal Easy Disk pada Tabel 4.5, kecepatan transfer data sistem USB OTG Enabler ini relatif lambat. Hal ini menandakan terjadi *bottleneck* pada sistem.

Menurut *datasheet* PL2303, kecepatan kabel data ini maksimal pada 1Mbps. Penggunaan baudrate 115200 bps dan 57600 bps karena keterbatasan pada platform Java yang digunakan. Dengan menggunakan aplikasi bantuan didapatkan baudrate yang didukung oleh platform Java adalah 1200, 2400, 4800, 9600,19200, 38400, 57600, 115200, 128000, dan 256000. Namun kedua kecepatan terakhir ini tidak didukung oleh PL2303.

Untuk mengetahui kecepatan transfer PL2303 dalam satuan byte per sekon (Bps) perlu diketahui terlebih dahulu konfigurasi bit per simbol yang digunakan. Dalam aplikasi ini konfigurasi yang digunakan adalah,

- Start bit : 1 bit
- Data bit : 8 bit
- Parity bit : 0 bit
- Stop bit : 1 bit

Sehingga total bit per simbol adalah 10 bit.

$$v = \frac{baudrate}{n} \tag{4.1}$$

Dimana,

v : kecepatan (Bps)

n : jumlah bit per simbol

Dengan menggunakan Persamaan 4.1 didapatkan kecepatan transfer sebesar:

$$v = \frac{115200}{10}$$

$$= 11520 \text{ Bps}$$

Sedangkan untuk kecepatan SPI 500 kHz mempunyai kecepatan transfer sebesar:

$$v = \frac{500000}{8}$$

$$= 62500 \text{ Bps}$$

Secara teoritis dengan menggunakan asumsi kecepatan 11520 Bps dan tidak terjadi *bottleneck*, file "kecil" yang berukuran 54715 byte dapat dikirim dalam waktu:

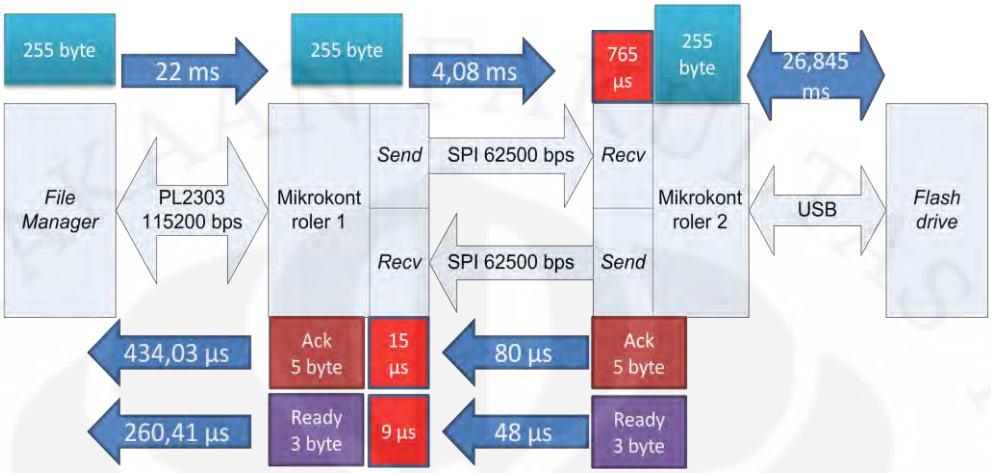
$$t = \frac{54715}{11520}$$

$$= 4{,}75 \text{ s}$$

Namun pada kenyataannya proses yang terjadi tidak secepat itu. Karena memori mikrokontroler yang terbatas maka dalam proses pengiriman data, file "kecil" tersebut dipecah-pecah. Panjang maksimum 1 paket adalah 255 byte dengan 2 byte telah digunakan sebagai header paket sehingga untuk data file hanya tersisa 253 byte. Karena pengiriman dibagi menjadi per 253 byte maka file "kecil" akan terjadi pengiriman sebanyak, 54715 / 253 = 216 kali paket 253 byte dan 1 kali paket 67 byte.

Pada proses pengiriman data terdapat waktu tunda karena operasi yang dilakukan oleh sistem. Dengan melihat kode assembly yang dihasilkan *compiler*, pada rutin penerimaan data SPI membutuhkan waktu 24 cycle untuk menyimpan data yang diterima. Dengan menggunakan *clock* sistem sebesar 8MHz maka dibutuhkan

waktu tunda sebesar 3 µs untuk setiap 1 byte data yang dikirim. Dengan maksimum besar paket adalah 255 byte untuk 1 kali pengiriman paket maka waktu tunda yang terjadi sebesar 3 x 255 = 765 µs.



Gambar 4.16 Proses transfer data

Ilustrasi proses transfer data yang terjadi dapat dilihat pada Gambar 4.16. Untuk satu sesi pengiriman paket 255 byte dibutuhkan waktu minimal sebesar:

- 22 ms, didapat dari waktu transfer melalui kabel data PL2303 dengan kecepatan 11520 Bps. (255 / 11520 = 22 ms)
- 4,08 ms, didapat dari waktu transfer melalui SPI dengan kecepatan 62500 Bps. (255/62500 = 4,08 ms)
- 765 µs, seperti telah disebutkan sebelumnya bahwa pengiriman melalui SPI membutuhkan jeda waktu 3 µs/byte untuk pemrosesan.
- 26,845 ms, didapat dari waktu tulis *flash drive* dengan asumsi kecepatan 89112,38 Bps. (255 / 89112,38 = 26,845 ms)
- 152 µs, didapat dari waktu pengiriman sinyal Ack dan Ready yang berjumlah 8 byte melalui SPI.
- 694,44 µs, didapat dari waktu pengiriman sinyal Ack dan Ready yang berjumlah 8 byte melalui PL2303.

Sehingga total waktu yang dibutuhkan adalah 22 ms + 4,08 ms + 765 µs + 26,845 ms + 152 µs + 694,44 µs = 54,536 ms.

Jadi waktu minimal yang dibutuhkan untuk mengirim file "kecil" adalah:

(216 paket x 54,536 ms) + (67/255 * 54,536 ms) = 11,794 s.

Dari hasil perhitungan ini bila dibandingkan dengan hasil percobaan sudah mendekati nilai sebenarnya. Sehingga dengan perhitungan yang sama namun dengan mengasumsikan kecepatan serial PL2303 dapat mencapai kecepatan maksimal dan SPI mencapai 2 MHz akan didapat hasil seperti pada Tabel 4.6.

Tabel 4.6 Hasil perhitungan waktu transfer dengan kecepatan PL2303 912600 bps

	Waktu transfer (s)	Kecepatan transfer	
		(Bps)	(kBps)
PL2303 = 921600 SPI = 500 kHz	7,509	7286,59	7,12
PL2303 = 921600 SPI = 2 MHz	6,827	8014,50	7,83

Dari data yang ada didapatkan bahwa kecepatan serial PL2303 berpengaruh besar terhadap kecepatan sistem. Kecepatan yang dibatasi pada 115200 bps menjadi sumber *bottleneck* yang terjadi.

Waktu sebesar 54,536 ms yang dihitung pada proses transfer data (Gambar 4.16) belum termasuk waktu transfer dari data overhead yang dikirim. Proses pengiriman data overhead hampir sama dengan proses pengiriman data pada Gambar 4.16, hanya saja data overhead hanya sampai pada mikrokontroler 2 tidak dilanjutkan hingga USB *flash drive*. Sehingga waktu penulisan sebesar 26,845 ms tidak dihitung. Jadi waktu yang dibutuhkan untuk mengirim data overhead dengan panjang data 255 byte adalah :

54,536 ms – 26,845 ms = 27,691 ms

Pada percobaan yang dilakukan, nama file yang digunakan dan waktu yang dibutuhkan untuk mentransfer data overhead dapat dilihat pada Tabel 4.7. Data ini hanya untuk kombinasi kecepatan PL2303 115200 bps dan SPI 500 kHz.

Tabel 4.7 Nama file pada percobaan

	Nama file	Ukuran (byte)	Waktu transfer (ms)
Kecil	kec_30.docx	11	1,195
Kecil banyak	*Lihat lampiran untuk daftar lengkap*	293	31,818
Besar	Silent Hill 2 - Credit.mp3	26	2,823

Dengan menggunakan data Tabel 4.4 untuk kombinasi kecepatan PL2303 115200 bps dan SPI 500 kHz dengan arah HP ke FD dan membandingkan dengan data pada Tabel 4.7 akan didapat rasio waktu transfer data overhead dengan waktu transfer total yang hasilnya dapat dilihat pada Tabel 4.8. Terlihat bahwa data overhead yang dikirim tidak berpengaruh signifikan terhadap waktu transfer keseluruhan dengan nilai persentase kurang dari 1%.

Tabel 4.8 Rasio waktu transfer data overhead terhadap waktu transfer total

	Waktu transfer (s)		Overhead (%)
	Total	Overhead	
Kecil	12,69	0,001195	0,00942
Kecil banyak	344,35	0,031818	0,00924
Besar	325,24	0,002823	0,00087

Dengan asumsi bahwa ukuran overhead 255 byte adalah 5% dari ukuran file, maka overhead akan berpengaruh secara signifikan jika ukuran file kurang dari 5100 byte. Atau dengan kata lain, overhead dinyatakan tidak berpengaruh signifikan untuk ukuran file lebih besar dari 5100 byte.

BAB 5
KESIMPULAN

Dari percobaan dan analisa yang didapat, dapat ditarik beberapa kesimpulan yaitu :

1. USB On-The-Go Enabler telah berhasil dibuat menggunakan sistem minimum mikrokontroler AT90USB1287 dan berhasil mendeteksi serta melakukan transfer data untuk ketiga buah *storage media* yang diujicobakan,
2. Kecepatan akses data USB OTG Enabler yang dibuat rata-rata adalah 3,76 kBps untuk telepon genggam ke *flash drive* dan 2,24 kBps untuk *flash drive* ke telepon genggam,
3. Kecepatan transfer USB OTG Enabler dibatasi oleh adanya *bottleneck* pada keterbatasan kecepatan serial pada platform Java,
4. Data overhead yang dikirim tidak berpengaruh signifikan terhadap proses transfer data keseluruhan bila ukuran file lebih besar dari 5100 byte.

DAFTAR ACUAN

[1] USB Implementers Forum, Inc. *Universal Serial Bus Revision 2.0 specification.* USB Developer – Document. [Online] 27 April 2000, dikutip November 2009. http://www.usb.org/developers/docs/

[2] Brey, Barry B. *Mikroprosesor Intel 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro Prosesor, Pentium II, Pentium III, Pentium 4 : Arsitektur, Pemrograman, dan Antarmuka*. [penerj.] Y. A. Amrulloh, M. A. Setiawan, & S. Adinandra. Edisi 6. Yogyakarta: Penerbit ANDI, 2005. Vol. 1.

[3] Peacock, C. *USB in a NutShell.* [Online] 6 April 2007, dikutip November 2009. http://www.beyondlogic.org/usbnutshell/usb1.htm

[4] Budiharto, W. *Interfacing Komputer dan Mikrokontroler.* Jakarta: PT. Elex Media Komputindo, 2004.

[5] Maxim Integrated Products. *USB On-The-Go Basic.* Maxim Application Note 1822. [Online] 20 Desember 2002, dikutip 28 Agustus 2009. http://www.maxim-ic.com/an1822

[6] Philips Semiconductor. *USB On-The-Go: A Tutorial.* TechOnline. [Online] Januari 2002, dikutip 28 Agustus 2009. http://www.techonline.com/learning/techpaper/193101730?pop_up=http%3A//www.techonline.com/article/pdf/showPDF.jhtml%3Fid%3D1931017301

[7] Heryanto, M. A., & P, W. A. *Pemrograman Bahasa C untuk Mikrokontroler ATMega8535.* Yogyakarta: Penerbit ANDI, 2008.

[8] Wardhana, L. *Belajar Sendiri Mikrokontroler AVR Seri ATMega8535 Simulasi, Hardware, dan Aplikasi.* Yogyakarta: Penerbit ANDI, 2006.

[9] Microsoft Corp. *Microsoft Extensible Firmware Initiative FAT32 File System Specification.* Windows Hardware Developer Central. [Online] 6 Desember 2000, dikutip Oktober 2009. http://www.microsoft.com/whdc/system/platform/firmware/fatgendown.mspx

[10] Stoffregen, P. *Understanding FAT32 Filesystems.* Paul's 8051 Code Library. [Online] 24 Februari 2005, dikutip 19 November 2009. http://www.pjrc.com/tech/8051/ide/fat32.html

[11] Dobiash, J. *FAT16 Structure Information*. Dobiash Realms [Online] 17 Juni 1999, dikutip 19 November 2009. http://home.teleport.com/~brainy/fat16.htm

[12] Raharjo, B., Heryanto, I., & Haryono, A. *Tuntunan Pemrograman Java Untuk Handphone.* Bandung: Informatika, 2007.

[13] Camera, Dean. *LUFA: The Lightweight USB Framework for AVRs*. LUFA Blogs. [Online] 19 Februari 2010, dikutip 25 Februari 2010. http://www.fourwalledcubicle.com/LUFA.php

[14] Chan. *FAT16 Structure Information*. ELM. [Online] 28 Januari 2010, dikutip 12 Juni 2010. http://elm-chan.org/fsw/ff/ 00index_e.html

Easy Disk, PL2303 = 115200 bps, SPI = 500 kHz

Nama flash disk	Arah	Nama file	test #	ukuran	waktu (s)	kecepatan (bps)	kecepatan (kbps)
Easy Disk	HP to FD	kecil	1	54715	13,274	4121,97	4,03
			2	54715	12,448	4395,49	4,29
			3	54715	12,420	4405,39	4,30
			4	54715	12,770	4284,65	4,18
			5	54715	12,517	4371,26	4,27
		kecil banyak	1	1432499	356,057	4023,23	3,93
			2	1432499	350,034	4092,46	4,00
			3	1432499	330,193	4338,37	4,24
			4	1432499	352,200	4067,29	3,97
			5	1432499	333,245	4298,64	4,20
		besar	1	1498929	325,393	4606,52	4,50
			2	1498929	327,728	4573,70	4,47
			3	1498929	326,699	4588,10	4,48
			4	1498929	323,200	4637,78	4,53
			5	1498929	323,164	4638,29	4,53
	FD to HP	kecil	1	54715	19,445	2813,83	2,75
			2	54715	18,987	2881,71	2,81
			3	54715	19,232	2845,00	2,78
			4	54715	18,955	2886,57	2,82
			5	54715	19,029	2875,35	2,81
		kecil banyak	1	1432499	502,590	2850,23	2,78
			2	1432499	508,500	2817,11	2,75
			3	1432499	501,829	2854,56	2,79
			4	1432499	500,860	2860,08	2,79
			5	1432499	500,527	2861,98	2,79
		besar	1	1498929	517,706	2895,33	2,83
			2	1498929	515,634	2906,96	2,84
			3	1498929	515,768	2906,21	2,84
			4	1498929	517,794	2894,84	2,83
			5	1498929	515,449	2908,01	2,84

Easy Disk, PL2303 = 57600 bps, SPI = 500 kHz

Nama flash disk	Arah	Nama file	test #	ukuran	waktu (s)	kecepatan (bps)	kecepatan (kbps)
Easy Disk	HP to FD	kecil	1	54715	17,317	3159,61	3,09
			2	54715	16,879	3241,60	3,17
			3	54715	16,782	3260,34	3,18
			4	54715	17,441	3137,15	3,06
			5	54715	17,354	3152,88	3,08
		kecil banyak	1	1432499	416,943	3435,72	3,36
			2	1432499	430,167	3330,10	3,25
			3	1432499	423,686	3381,04	3,30
			4	1432499	420,687	3405,14	3,33
			5	1432499	421,282	3400,33	3,32
		besar	1	1498929	444,977	3368,55	3,29
			2	1498929	449,902	3331,68	3,25
			3	1498929	441,303	3396,60	3,32
			4	1498929	430,918	3478,46	3,40
			5	1498929	438,008	3422,15	3,34
	FD to HP	kecil	1	54715	30,608	1787,60	1,75
			2	54715	28,555	1916,13	1,87
			3	54715	29,790	1836,69	1,79
			4	54715	31,872	1716,71	1,68
			5	54715	29,134	1878,05	1,83
		kecil banyak	1	1432499	781,314	1833,45	1,79
			2	1432499	782,024	1831,78	1,79
			3	1432499	781,628	1832,71	1,79
			4	1432499	782,104	1831,60	1,79
			5	1432499	783,517	1828,29	1,79
		besar	1	1498929	812,706	1844,37	1,80
			2	1498929	813,244	1843,15	1,80
			3	1498929	815,678	1837,65	1,79
			4	1498929	814,256	1840,86	1,80
			5	1498929	814,049	1841,33	1,80

Easy Disk, PL2303 = 115200 bps, SPI = 125 kHz

Nama flash disk	Arah	Nama file	test #	ukuran	waktu (s)	kecepatan (bps)	kecepatan (kbps)
Easy Disk	HP to FD	kecil	1	54715	15,378	3558,00	3,47
			2	54715	14,797	3697,71	3,61
			3	54715	15,078	3628,80	3,54
			4	54715	14,123	3874,18	3,78
			5	54715	14,506	3771,89	3,68
		kecil banyak	1	1432499	370,139	3870,16	3,78
			2	1432499	359,750	3981,93	3,89
			3	1432499	362,635	3950,25	3,86
			4	1432499	360,384	3974,92	3,88
			5	1432499	364,133	3934,00	3,84
		besar	1	1498929	376,153	3984,89	3,89
			2	1498929	379,970	3944,86	3,85
			3	1498929	387,304	3870,16	3,78
			4	1498929	381,650	3927,50	3,84
			5	1498929	379,476	3950,00	3,86
	FD to HP	kecil	1	54715	25,093	2180,49	2,13
			2	54715	25,555	2141,07	2,09
			3	54715	24,729	2212,58	2,16
			4	54715	24,872	2199,86	2,15
			5	54715	24,817	2204,74	2,15
		kecil banyak	1	1432499	652,739	2194,60	2,14
			2	1432499	652,982	2193,78	2,14
			3	1432499	653,210	2193,01	2,14
			4	1432499	651,987	2197,13	2,15
			5	1432499	652,402	2195,73	2,14
		besar	1	1498929	678,981	2207,62	2,16
			2	1498929	678,603	2208,85	2,16
			3	1498929	676,669	2215,16	2,16
			4	1498929	677,111	2213,71	2,16
			5	1498929	676,250	2216,53	2,16

Kecepatan Internal

Nama flash disk	Arah	Jenis file	test #	ukuran	waktu (s)	kecepatan (bps)	kecepatan (kbps)
	HP to HP	kecil	1	54715	0,756	72374,34	70,68
			2	54715	0,770	71058,44	69,39
			3	54715	0,937	58393,81	57,03
			4	54715	0,775	70600,00	68,95
			5	54715	0,808	67716,58	66,13
		kecil banyak	1	1432499	19,306	74199,68	72,46
			2	1432499	18,457	77612,78	75,79
			3	1432499	19,588	73131,46	71,42
			4	1432499	18,346	78082,36	76,25
			5	1432499	18,480	77516,18	75,70
		besar	1	1498929	13,745	109052,67	106,50
			2	1498929	14,081	106450,47	103,96
			3	1498929	13,731	109163,86	106,61
			4	1498929	14,077	106480,71	103,99
			5	1498929	13,961	107365,45	104,85
Easy Disk	FD to FD	kecil	1	54715	0,614	89112,38	87,02
			2	54715	0,614	89112,38	87,02
			3	54715	0,627	87264,75	85,22
			4	54715	0,614	89112,38	87,02
			5	54715	0,596	91803,69	89,65
		kecil banyak	1	1432499	20,487	69922,34	68,28
			2	1432499	20,285	70618,63	68,96
			3	1432499	19,315	74165,10	72,43
			4	1432499	20,372	70317,05	68,67
			5	1432499	20,460	70014,61	68,37
		besar	1	1498929	9,023	166123,13	162,23
			2	1498929	8,894	168532,61	164,58
			3	1498929	8,968	167141,95	163,22
			4	1498929	9,019	166196,81	162,30
			5	1498929	8,621	173869,50	169,79
Easy MP3 EM120X	FD to FD	kecil	1	54715	3,263	16768,31	16,38
			2	54715	3,180	17205,97	16,80
			3	54715	3,175	17233,07	16,83
			4	54715	3,313	16515,24	16,13
			5	54715	3,226	16960,63	16,56
		kecil banyak	1	1432499	87,272	16414,19	16,03
			2	1432499	87,069	16452,46	16,07
			3	1432499	87,558	16360,57	15,98
			4	1432499	87,083	16449,81	16,06

Nama flash disk	Arah	Jenis file	test #	ukuran	waktu (s)	kecepatan (bps)	kecepatan (kbps)
			5	1432499	86,358	16587,91	16,20
		besar	1	1498929	76,592	19570,31	19,11
			2	1498929	77,547	19329,30	18,88
			3	1498929	77,575	19322,32	18,87
			4	1498929	77,016	19462,57	19,01
			5	1498929	76,758	19527,98	19,07
ADATA C003	FD to FD	kecil	1	54715	1,186	46134,06	45,05
			2	54715	1,297	42185,81	41,20
			3	54715	1,269	43116,63	42,11
			4	54715	1,333	41046,51	40,08
			5	54715	1,357	40320,56	39,38
		kecil banyak	1	1432499	53,344	26853,99	26,22
			2	1432499	53,340	26856,00	26,23
			3	1432499	53,681	26685,40	26,06
			4	1432499	53,423	26814,27	26,19
			5	1432499	51,157	28002,01	27,35
		besar	1	1498929	29,622	50601,88	49,42
			2	1498929	29,829	50250,73	49,07
			3	1498929	29,672	50516,61	49,33
			4	1498929	29,875	50173,36	49,00
			5	1498929	29,838	50235,57	49,06

Nama file yang digunakan pada percobaan

	Nama file	ukuran (byte)
kecil	kec_30.docx	54.715
kecil banyak	kec_01.db	10.240
	kec_02.docx	14.129
	kec_03.docx	34.857
	kec_04.jpg	64.103
	kec_05.vsd	77.312
	kec_06.doc	26.112
	kec_07.rar	46.865
	kec_08.vsd	28.160
	kec_09.xls	24.064
	kec_10.docx	84.378
	kec_11.gz	83.924
	kec_12.pdf	45.293
	kec_13.pdf	18.581
	kec_14.pdf	79.215
	kec_15.vsd	36.352
	kec_16.vsd	35.840
	kec_17.doc	30.208
	kec_18.mht	13.915
	kec_19.mht	12.536
	kec_20.mht	11.781
	kec_21.zip	45.502
	kec_22.zip	46.264
	kec_23.pdf	77.728
	kec_24.pdf	86.605
	kec_25.pdf	76.052
	kec_26.pdf	94.446
	kec_27.pdf	58.201
	kec_28.PDF	97.323
	kec_29.rar	17.798
	kec_30.docx	54.715
besar	Silent Hill 2 - Credit.mp3	1.498.929