UNIVERSITAS INDONESIA

SISTEM PELACAK BUS KAMPUS DENGAN MENGGUNAKAN MODUL DT-51 LCMS DAN *WIRELESS* YS 1020 RF DATA TRANSCEIVER

SKRIPSI

ROFANAHARTO ANUGRAH
0404030733

FAKULTAS TEKNIK
DEPARTEMEN ELEKTRO
DEPOK
DESEMBER 2008

UNIVERSITAS INDONESIA

SISTEM PELACAK BUS KAMPUS DENGAN MENGGUNAKAN MODUL DT-51 LCMS DAN *WIRELESS* YS 1020 RF DATA TRANSCEIVER

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

ROFANAHARTO ANUGRAH
0404030733

FAKULTAS TEKNIK
DEPARTEMEN ELEKTRO
DEPOK
DESEMBER 2008

HALAMAN PERNYATAAN ORISINALITAS

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

Nama : Rofanaharto Anugrah

NPM : 0404030733

Tanda Tangan :

Tanggal : 12 Desember 2008

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh	:
Nama	: Rofanaharto Anugrah
NPM	: 0404030733
Program Studi	: Elektro
Judul Skripsi	: Sistem Pelacak Bus Kampus Dengan Menggunakan Modul DT-51 LCMS dan *Wireless* YS 1020 RF Data Transceiver

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

DEWAN PENGUJI

Pembimbing	: Dr. Ir. Riri Fitri Sari, M.Sc, MM	(.....)
Penguji	: Prof. Dr. Ir. Bagio Budiarjo, M.Sc	(.....)
Penguji	: Dr.-Ing. Ir. Kalamullah Ramli, M.Eng	(.....)

Ditetapkan di : Ruang Rapat Lantai 1 Departemen Teknik Elektro Depok

Tanggal : 19 Desember 2008

KATA PENGANTAR/UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Allah SWT, 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 Program Studi Elektro pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih kepada:

(1) Dr. Ir. Riri Fitri Sari, M.Sc, MM, selaku dosen pembimbing yang telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
(2) Dr. Abdul Muis ST. MT., yang telah meminjamkan peralatan yang digunakan dalam skripsi ini;
(3) orang tua dan keluarga saya yang telah memberikan dukungan moral dan material; dan
(4) teman-teman saya di Teknik Elektro angkatan 2004 khususnya Anggi, Hartanto, Arie, Yani, dan Ulfa atas ide-ide serta saran dan juga bantuan berupa peralatan yang digunakan dalam skripsi ini.

Akhir kata, saya berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 12 Desember 2008

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

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

Nama : Rofanaharto Anugrah
NPM : 0404030733
Program Studi : Elektro
Departemen : 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:

SISTEM PELACAK BUS KAMPUS DENGAN MENGGUNAKAN MODUL DT-51 LCMS DAN *WIRELESS* YS 1020 RF DATA TRANSCEIVER

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan 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 : 12 Desember 2008
Yang menyatakan

( )

ABSTRAK

Nama : Rofanaharto Anugrah
Program Studi : Teknik Elektro
Judul : Sistem Pelacak Bus Kampus Dengan Menggunakan Modul DT-51 LCMS dan *Wireless* YS 1020 RF Data Tranceiver

Sistem pelacak bus kampus merupakan sebuah kesatuan dari perangkat-perangkat lunak maupun keras tertentu yang dapat melacak posisi dari bus kampus dan menampilkannya agar dapat dilihat oleh pengguna. Skripsi ini akan membahas sistem pelacak bus kampus di Universitas Indonesia secara keseluruhan mulai dari landasan teori tentang teknologi yang digunakan, perancangan sistem, dan hasil pengujian serta analisa sistem. Selain itu juga diadakan kuisioner yang memberikan gambaran mengenai tanggapan pengguna terhadap sistem pelacak bus kampus Universitas Indonesia ini. Perangkat lunak yang digunakan adalah cygwin sebagai *compiler* bahasa C. Sedangkan perangkat-perangkat keras utama yang digunakan adalah komputer, modul DT-51 LCMS 2.0 dan modul *wireless* YS 1020 RF Data Transceiver.

Modul DT-51 LCMS memberikan ID kepada bus berupa 8 bit biner. Oleh modul *wireless* YS 1020 RF Data Transceiver di sisi pengirim diteruskan dan diterima oleh modul *wireless* YS 1020 RF Data Transceiver di sisi penerima. Modul *wireless* YS 1020 RF Data Transceiver penerima yang terpasang pada komputer yang diasumsikan ada di setiap halte meneruskannya ke komputer halte tersebut agar dapat diolah dengan menambahkan ID halte dan dikirimkan ke *server* melalui protokol TCP/IP. Di *server*, data terakhir direpresentasikan dalam bentuk GUI sederhana. Langkah-langkah yang dilakukan adalah memprogram modul DT-51 LCMS 2.0, memasang dan menggabungkan modul DT-51 LCMS 2.0 dengan modul *wireless* YS 1020 RF Data Transceiver serta komputer halte/fakultas, pembuatan program menggunakan bahasa C untuk komputer halte/fakultas dan untuk komputer *server*.

Hasil pengujian menunjukkan bahwa prototipe sistem pelacak bus kampus Universitas Indonesia ini dapat menampilkan posisi bus kampus dengan cukup akurat, namun masih mempunyai kekurangan yaitu GUI masih sangat sederhana, hanya bisa menampilkan satu bus per satu waktu, dan aplikasinya hanya bisa dijalankan sekali setiap dieksekusi. Dari tanggapan pengguna terhadap sistem ini diketahui bahwa sistem pelacak bus kampus berguna, memiliki GUI yang kurang memenuhi ekspektasi, dan setuju bahwa posisi bus yang ditampilkan cukup akurat.

Kata kunci:
context-aware, sistem pelacak bus kampus, GUI

ABSTRACT

Name : Rofanaharto Anugrah
Study Program: Electrical Engineering
Title : Campus Bus Tracking System Using DT-51 LCMS and *Wireless* YS 1020 RF Data Tranceiver Modules

Campus bus tracking system is an integration of certain softwares and hardwares which can track the position of campus buses and display it to user. This final project will focus on University of Indonesia campus bus, starting from the basic theory about technology used in this system, design of the system, and the testing result with analysis about the system. Besides that, there is a questioner form that will give an abstraction about users' feedback of University of Indonesia campus bus tracking system. Software used in this final project is cygwin as a C compiler. Hardwares used in this system are DT-51 LCMS 2.0 module and YS 1020 RF Data Transceiver wireless module.

DT-51 LCMS module gives an ID for a bus in 8 bit binary. YS 1020 RF Data Transceiver wireless module on the transmitting side forward it and YS 1020 RF Data Transceiver wireless module on the receiving side receive it. YS 1020 RF Data Transceiver wireless module on receiving side installed on a computer is assumed installed on each shelter in University of Indonesia. This computer processes the data and inserts each shelter's ID. The next process is to forward the final data via TCP/IP to server. In the server, the final data contains bus' ID and shelter's ID represented in a simple GUI. The steps taken are programming DT-51 LCMS module, attaching and integrating DT-51 LCMS 2.0 module and YS 1020 RF Data Transceiver wireless module with computer in the shelter, making program using C for computer in the shelter and for server.

The testing results shows that the prototype of University of Indonesia campus bus tracking system can display the position of campus bus quite accurately but still have some weaknesses. The weaknesses are the GUI is very simple, the GUI can only show a bus in a time, and the application can only run once when executed. Users' feedback about this campus bus tracking system shows that this system is helpful, the design of GUI isn't up to expectation. However, users agree that the position shows in GUI is accurate.

Key words:
context-aware, campus bus tracking system, GUI

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PERNYATAAN ORISINALITAS ii
LEMBAR PENGESAHAN iii
KATA PENGANTAR/UCAPAN TERIMA KASIH iv
LEMBAR PERSETUJUAN PUBLIKASI KARYA ILMIAH v
ABSTRAK vi
ABSTRACT vii
DAFTAR ISI viii
DAFTAR GAMBAR x
DAFTAR TABEL xii
DAFTAR GRAFIK xiii
DAFTAR SINGKATAN xiv
DAFTAR LAMPIRAN xv
1. PENDAHULUAN 1
1.1. Latar Belakang 1
1.2. Tujuan 2
1.3. Batasan Masalah 2
1.4. Sistematika Penulisan 2

2. LANDASAN TEORI 4
2.1. Sekilas *Context-Aware* 4
2.2. Sekilas Komunikasi Serial 7
2.2.1. Sekilas Mengenai RS-232 7
2.2.2. *Port* Serial 8
2.2.2. *In-System Programming* (ISP) 10
2.3. Perangkat-Perangkat yang Digunakan 11
2.3.1. Modul DT-51™ Low Cost Micro System (LCMS) versi 2.0 dengan Mikrokontroler Atmel AT89S51 11
2.3.2. Modul *Wireless* YS-1020UA RF Data Transceiver 16

3. SISTEM PELACAK BUS KAMPUS MENGGUNAKAN MODUL DT-51™ LCMS 2.0 DAN *WIRELESS* YS-1020UA RF 19
3.1. Gambaran Umum Sistem Pelacak Bus Kampus 19
3.2. Pemrograman Modul DT-51 LCMS 2.0 20
3.3. Pembuatan Program Berbahasa C untuk Komputer Halte/Fakultas 23
3.4. Pembuatan Program Berbahasa C untuk Komputer *Server* 27

4. ANALISA PELACAK BUS KAMPUS MENGGUNAKAN MODUL DT-51™ LCMS 2.0 DAN *WIRELESS* YS-1020UA RF 30
4.1. Pengujian Prototipe Sistem Pelacak Bus Kampus 30
4.2. Tanggapan Pengguna Terhadap Sistem Pelacak Bus Kampus 35

5. KESIMPULAN..... 37
DAFTAR REFERENSI ..... .38
LAMPIRAN. ..... .40

DAFTAR GAMBAR

Gambar 2.1. Arsitektur abstrak dari sebuah sistem *context-aware*.....6

Gambar 2.2. Blok diagram dan gambar asli modul DT-51™ LCMS versi 2.0.....11

Gambar 2.3. Alokasi RX, TX, dan ground pada pin J1 modul DT-51 LCMS versi 2.0 dan *port* COM komputer.....12

Gambar 2.4. Blok diagram alokasi pin-pin J3 sampai J6 modul DT-51 LCMS versi 2.0.....13

Gambar 2.5. Blok diagram alokasi pin J7 modul DT-51 LCMS versi 2.0.....15

Gambar 2.6. Blok diagram alokasi pin-pin J8 dan J9 modul DT-51 LCMS versi 2.0.....16

Gambar 2.7. Modul *wireless* YS-1020UA RF Data Transceiver.....16

Gambar 2.8. Dimensi modul *wireless* YS-1020UA RF Data Transceiver.....18

Gambar 3.1. Gambaran umum sistem pelacak bus kampus untuk tiap-tiap fakultas atau halte dimana ada bus yang terdeteksi.....19

Gambar 3.2. Skema sistem pelacak bus kampus yang akan dibuat.....20

Gambar 3.3. Diagram alir pada bagian bus kampus.....20

Gambar 3.4. Diagram alir pada bagian halte/fakultas.....23

Gambar 3.5. Tampilan di cygwin C *compiler* agar data yang diterima dari modul DT-51 LCMS 2.0 dapat dicek.....26

Gambar 3.6. Diagram alir pada bagian *server*.....27

Gambar 3.7. Tampilan di cygwin C *compiler* agar data yang diterima dari program komputer halte/fakultas dapat dicek.....28

Gambar 3.8. GUI sederhana yang menggambarkan jalur yang dilewati bus kampus.....29

Gambar 4.1. Foto keseluruhan sistem yang diujikan.....30

Gambar 4.2. GUI pada *server* yang menunjukkan ada bus label merah di halte Fakultas Psikologi.....31

Gambar 4.3. GUI pada *server* yang menunjukkan ada bus label merah di halte Fakultas Hukum.....32

Gambar 4.4. GUI pada *server* yang menunjukkan ada bus label biru di halte Fakultas Teknik.....33

Gambar 4.5. GUI pada *server* yang menunjukkan ada bus label biru di halte Fakultas Ekonomi.....34

DAFTAR TABEL

Tabel 2.1. Pin 1.5-1.7 pada *port* 1 serta salah satu alternatif fungsinya.....14

Tabel 2.2. Pin 3.0-3.7 pada *port* 3 serta salah satu alternatif fungsinya.....15

Tabel 2.3. Deskripsi dari pin-pin yang ada pada modul *wireless* YS-1020UA RF Data Transceiver.....18

Tabel 3.1. Pengalokasian bit-bit untuk masing-masing halte yang ada di UI.....25

Tabel 4.1. Tanggapan pengguna terhadap sistem pelacak bus kampus UI....35

DAFTAR GRAFIK

Grafik 4.1. Grafik hasil tanggapan pengguna terhadap sistem pelacak bus kampus UI.....36

DAFTAR SINGKATAN

GUI	: Graphical User Interface
UI	: Universitas Indonesia
USB	: Universal Serial Bus
GPS	: Global Positioning System
RF	: Radio Frequency
ISM	: Industrial, Scientific, and Medical
I/O	: Input Output

DAFTAR LAMPIRAN

Lampiran 1. Program mikrokontroler untuk bus kampus merah nomor 5

Lampiran 2. Program berbahasa C yang terinstal di komputer halte/fakultas

Lampiran 3. Program berbahasa C yang terinstal di komputer *server*

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi saat ini yang sangat pesat di berbagai bidang, termasuk di bidang informasi dan transportasi memberi sebuah pemikiran baru untuk dapat mencari solusi bagaimana cara mengintegrasikan teknologi informasi dengan transportasi.

Sebagaimana kita tahu, manusia sebagai makhluk hidup tentu tidak dapat berdiam diri saja, melainkan manusia mempunyai banyak kegiatan. Dalam melakukan kegiatan tersebut, manusia memerlukan perpindahan dari satu posisi ke posisi lain. Perpindahan itu dapat dalam jarak pendek yang dapat ditempuh hanya dalam berjalan kaki saja, atau dalam jarak yang lebih jauh yang sulit ditempuh dengan berjalan kaki. Oleh karena itu dibutuhkan alat transportasi. Alat transportasi itu dapat berupa kendaraan darat, laut dan udara.

Salah satu alat transportasi yang ada misalnya bus kampus. Para pengguna bus kampus, baik mahasiswa atau civitas akademika lainnya tentu berharap agar dapat menggunakan alat transportasi ini secara efektif tanpa membuang-buang waktu.

Untuk dapat mempermudah mahasiswa atau pengguna bus kampus lainnya, dapat dikembangkan sebuah sistem yang mampu melacak posisi dari bus kampus. Posisi dari bus kampus dilacak dengan menggunakan mikrokontroler AT89S51 modul DT-51 LCMS versi 2.0, lalu dengan menggunakan modul *wireless* YS-1020UA RF Data Transceiver datanya dikirimkan ke komputer fakultas yang akan mengirimkannya lagi ke *server* pusat. *Server* ini tentu akan menjadi sebuah *server* yang akan memuat informasi-informasi yang berhubungan dengan bus kampus tersebut, khususnya posisi dari bus kampus tersebut yang akan ditampilkan dalam bentuk GUI sederhana bukan teks biasa. Maka dengan demikian pengguna bus kampus dapat mengakses *server* tersebut selama *server* tersebut tetap hidup dan berfungsi serta tidak ada masalah dalam jaringan komputer yang berkaitan dengan sistem ini.

1.2 Tujuan

Tujuan dari penyusunan skripsi ini adalah pengaplikasian mikrokontroler AT89S51 modul DT-51 LCMS versi 2.0 untuk melacak posisi dari bus kampus UI, lalu dapat diakses di *server* kapanpun sehingga dapat membuat sebuah sistem yang *context-aware*. Pengguna dapat mengaksesnya jika ingin menggunakan bus kampus dapat melakukan aktivitas lain terlebih dahulu tanpa perlu membuang waktu lama menunggu bus kampus, atau dapat segera mengejar bus kampus apabila bus kampus akan segera tiba.

1.3 Batasan Masalah

Pada skripsi ini, beberapa masalah yang akan dibatasi adalah sebagai berikut:

1. Perancangan GUI untuk sistem ini hanya terbatas pada kampus UI Depok, sehingga tampilan GUI yang akan memuat posisi dari bus kampus merupakan peta yang menggambarkan lingkungan fisik kampus UI Depok saja.
2. Jumlah halte pemberhentian atau tempat pemberhentian lain yang umum untuk bus kampus UI berjumlah 15 tempat, maka untuk desain dan analisa sistem akan dikhususkan di 15 titik tersebut saja. Sementara untuk perancangan akhir nantinya hanya diambil contoh satu titik saja.
3. Sistem ini masih merupakan prototipe dan belum dapat diterapkan dalam skala UI, hanya pada skala kecil saja.

1.4 Sistematika Penulisan

Sistematika penulisan pada skripsi ini adalah sebagai berikut :

BAB 1 PENDAHULUAN

Bab ini terdiri dari latar belakang, tujuan, batasan masalah, dan sistematika penulisan.

BAB 2 LANDASAN TEORI

Bab ini menjelaskan tentang teori-teori dasar mengenai pengertian *context-aware*, komunikasi serial, serta alat-alat yang digunakan dalam sistem pelacak bus kampus ini.

BAB 3 RANCANGAN SISTEM PELACAK BUS KAMPUS

Bab ini menjelaskan mengenai tahapan-tahapan yang dilakukan dalam merancang sistem pelacak bus kampus.

BAB 4 ANALISA

Bab ini berisi pengujian dan analisa dari sistem serta tanggapan pengguna terhadap sistem pelacak bus kampus UI

BAB 5 KESIMPULAN

Babi ini berisi kesimpulan keseluruhan skripsi

BAB 2
LANDASAN TEORI

2.1. Sekilas *Context-Aware*

Gagasan tentang *context* telah banyak diobservasi di berbagai macam lingkup, termasuk dalam tata bahasa, filosofi, representasi tentang pengetahuan dan pemecahan masalah dalam bidang kecerdasan buatan, serta teori komunikasi [1], [3]. Pada banyak hal di dunia ini, *context* adalah gagasan kunci dan logika telah dikembangkan untuk dapat menonjolkan tentang *context* dan memungkinkan *context* dipertimbangkan secara eksplisit di sistem berbasis pengetahuan [2].

Langkah pertama dalam mempelajari *context-aware computing* adalah dengan mendefinisikan komponen utamanya; "*context*". Sebenarnya, apa yang dimaksud dengan *context*? Sejak awal hingga sekarang banyak *workshop* atau seminar yang berfokus pada pertanyaan ini. Ada sangat banyak konsepsi tentang penggunaan istilah "*context*". Dengan sasaran untuk menemukan definisi yang konkrit, kita dapat memulainya dengan definisi yang dikemukakan oleh kamus definisi umum dan berlanjut dengan pemilihan yang paling sesuai dengan bidang *engineering* yang diurutkan secara kronologis. Mengacu pada kamus Merriam-Webster [4], istilah "*context*" biasanya mempunyai dua arti utama, yaitu:

1. Bagian-bagian dari sebuah wacana yang melingkupi sebuah kata atau bagian dan menghasilkan keterangan dalam artinya.
2. Kondisi-kondisi yang saling berkaitan dimana sesuatu ada atau terjadi.

Arti yang pertama sangat berkaitan dengan tata bahasa dan juga merupakan definisi yang sering digunakan, sedangkan arti yang kedua lebih umum. Sinonim lain dari kata yang ada seperti: keadaan, situasi, kondisi, posisi, postur, letak, sekitar, dan lingkungan [5]. Dari perspektif *engineering*, banyak definisi yang telah diajukan dan dimasukkan di antara dua sisi definisi di atas.

Istilah *context-aware computing* pertama kali diperkenalkan oleh Shilit dan Theimer di [6] dimana mereka merujuk *context* sebagai "lokasi dari penggunaan, kumpulan dari orang-orang dan objek-objek yang berdekatan, serupa perubahan-perubahan dari objek-objek tersebut seiring dengan waktu". Definisi serupa juga diberikan oleh P. G. Brown, J. D. Bovey, dan X. Chen [7]: "Kami

mendefinisikan *context* sebagai informasi apapun yang bisa digunakan untuk dapat mengkarakterisasi situasi pada sebuah entitas, dimana sebuah entitas tersebut dapat berupa seseorang, tempat, atau objek komputasional fisik". Brézillon dan Pomerol [8], mendefinisikan *context* sebagai "semua pengetahuan yang mendesak sebuah pemecahan masalah pada langkah tertentu tanpa ikut campur di dalamnya secara eksplisit". Dey di [9] mengajukan definisi yang lebih umum yang menyatakan: "*Context* adalah informasi apapun yang bisa digunakan untuk dapat mengkarakterisasi situasi pada sebuah entitas. Entitas tersebut adalah seseorang, atau objek yang dipertimbangkan relevan pada interaksi antara pengguna (*user*) dengan aplikasi itu sendiri". Definisi dari Dey ini memasukkan baik *input* eksplisit (informasi yang secara eksplisit tersedia untuk *user* melalui *user interface*) dan *input* implisit yang membutuhkan proses komputasi sebelum penggunaan efektif (informasi tentang keadaan jaringan lokal dengan mengambil jumlah dari *user* yang terkoneksi). Walaupun ada kesamaan dalam pendefinisian-pendefinisian tersebut, kita dapat membayangkan betapa sulit untuk menemukan dasar yang umum. Definisi-definisi ini jauh dari presisi matematika dan terkait erat dengan bidang dan fokus dari penggagas definisi-definisi tersebut.

Context telah menjadi sebuah bagian dalam berbagai macam bidang dari *computer science*, termasuk di dalamnya *pervasive computing* dan *computer security* pada saat ini. Jika dianalogikan dengan akal pikiran manusia, sasaran dari mempertimbangkan *context* adalah untuk menambah adaptibilitas dan pengambilan keputusan yang efektif. Di bawah ini akan sedikit dibahas tentang bidang-bidang utama yang memasukkan *context* sebagai *domain* dari *pervasive computing* secara umum dan keamanan pada lingkup ini secara khusus.

Pada dasarnya, *pervasive computing* sangat erat kaitannya dengan *context*. Alasan mendasar dari korelasi ini datang dari tingginya tingkat heterogenitas dan *ubiquity* dari komunikasi antar entitas pada lingkup ini. Dua aspek ini membutuhkan adaptasi *run-time* dari layanan yang disediakan dan dari divais yang dimiliki oleh *user* tergantung pada lokasi, peranan, dan tugas yang akan dikerjakan. *Context aware* pada sebuah sistem pervasif menurut [2] memiliki tiga fungsionalitas dasar, yaitu:

1. *Sensing*

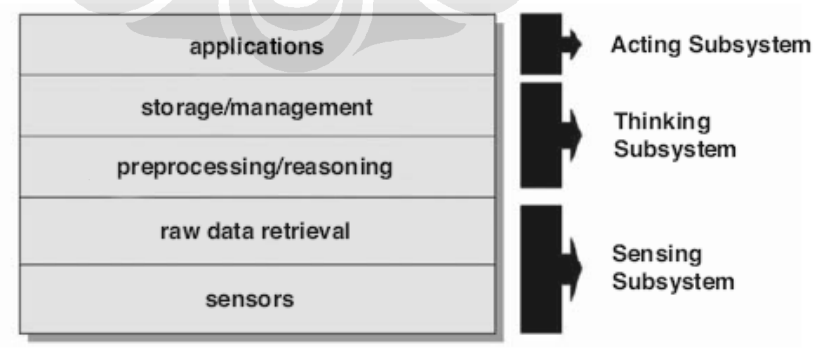
Sensor, baik yang bersifat biologis atau tidak, memberikan data atau informasi tentang lingkungan fisik atau beberapa aspek dari lingkungan fisik tersebut. Pengetahuan tentang lingkungan fisik tersebut dapat digunakan oleh sistem komputer untuk menentukan tindakan yang paling tepat atas situasi yang terjadi. Kombinasi dari beberapa sensor dapat memberikan informasi yang lebih untuk sistem komputer. Informasi apa sajakah yang bisa didapat dengan menggunakan sensor? Beragam sensor telah dikembangkan seperti sensor cahaya, sensor suhu, sensor asap, sensor gerakan, atau sensor sentuhan. Oleh karena itu sensor dapat menerima banyak informasi seperti contoh-contoh tersebut.

2. *Thinking*

Setelah data didapatkan menggunakan sebuah atau banyak sensor, yang harus dilakukan adalah mengutilisasi data tersebut dan bagaimana caranya agar dapat membuat data-data tersebut menjadi bernilai informasi.

3. *Acting*

Setelah konteks dari informasi telah dikumpulkan atau situasi telah dapat dikenali, maka *action* (tindakan) pun dilakukan. Tindakan mungkin harus dilakukan tepat waktu sebelum situasi berubah yang mengakibatkan harus mengambil tindakan yang lain. Secara ideal, *user* seharusnya dapat memegang kendali dan dapat meng-*override*, membatalkan, memberhentikan, atau membalikkan efek dari dari tindakan.



Gambar 2.1. Arsitektur abstrak dari sebuah sistem *context-aware* [2]

2.2. Sekilas Komunikasi Serial

Mengacu pada [10], dalam ilmu komputer dan ilmu telekomunikasi, komunikasi serial adalah proses dari pengiriman data satu bit dalam satu waktu secara sekuensial melalui kanal komunikasi atau *bus* pada komputer. Hal ini tentunya sangat kontras dengan komunikasi paralel, dimana beberapa bit dikirimkan secara bersamaan pada sebuah jalur yang terdiri dari beberapa kanal berkabel yang tersusun paralel. Komunikasi serial digunakan untuk semua komunikasi *long-haul* dan kebanyakan jaringan komputer dimana biaya untuk kabel dan kesulitan sinkronisasi membuat komunikasi paralel tidak praktis. Pada jarak yang lebih dekat, *bus* serial pada komputer menjadi lebih umum digunakan karena kekurangan dari *bus* paralel, misalnya ketidakcocokan *clock* dan densitas interkoneksi, mengalahkan kelebihannya, yaitu tidak perlu ada proses serialisasi dan deserialisasi (*serializer and deserializer*/SERDES). Teknologi yang berkembang untuk memastikan integritas sinyal dan untuk mengirim dan menerima pada kecepatan per jalur yang cukup tinggi telah membuat jalur-jalur serial menjadi kompetitif. Migrasi dari PCI ke PCI-Express contohnya.

2.2.1. Sekilas Mengenai RS-232

Pada [11] disebutkan bahwa RS-232 merupakan standar untuk pengkoneksian sinyal data biner serial antara sebuah DTE (*Data Terminal Equipment*) dan sebuah DCE (*Data Circuit-terminating Equipment*). Hal ini umumnya digunakan pada *port* serial pada komputer. Standar RS-232 yang ditetapkan oleh EIA (Electronics Industries Association) pada 1969 mendefinisikan hal-hal berikut:

- Karakteristik sinyal elektrik seperti level voltase; *signaling rate*; pewaktuan dan *slew-rate* dari sinyal; level ketahanan voltase; kelakuan pada *short-circuit*; dan kapasitansi beban maksimumnya
- Karakteristik mekanikal antarmuka, konektor yang dapat dipasang, dan identifikasi pin
- Fungsi-fungsi dari setiap sirkuit di konektor antarmuka

- Subset standar dari sirkuit antarmuka untuk aplikasi telekomunikasi yang dipilih

Sedangkan elemen-elemen berikut ini tidak didefinisikan pada standar RS-232:

- *Encoding* untuk karakter (misalnya ASCII, Baudot code, atau EBCDIC)
- *Framing* dari karakter-karakter dalam aliran data (*bits per character, start/stop bits,* dan *parity bits*)
- Protokol untuk pendeteksian *error* atau algoritma untuk kompresi data
- *Bit rates* untuk transmisi, walaupun dikatakan dalam standar ini ditujukan untuk *bit rates* kurang dari 20.000 bps padahal banyak modem mendukung kecepatan 115.200 bps atau lebih
- *Power supply* untuk divais-divais eksternal

2.2.2. *Port* Serial

Port serial merupakan antarmuka fisik untuk komunikasi serial melalui yang melewatkan informasi mana yang masuk atau keluar satu bit dalam satu waktu tertentu seperti yang telah dijelaskan sebelumnya dan juga tentunya kontras dengan *port* paralel [12]. Ethernet, FireWire, dan USB memang mengalirkan data secara serial, namun isitilah *port* serial ini sebenarnya merujuk pada sebuah perangkat keras yang memenuhi standar RS-232, dimana perangkat tersebut berantarmuka dengan modem atau divais-divais lain misalnya modul *wireless* yang akan dipakai dalam sistem pelacak bus kampus ini. *Port* serial ini pada saat sekarang sudah semakin jarang ditemukan pada komputer *desktop* atau laptop *low-end* karena mulai tergantikan oleh USB. *Port* serial banyak terdapat pada perangkat jaringan komputer semacam *router* atau *switch* sebagai jalur untuk konfigurasi. *Port* serial masih digunakan pada perangkat-perangkat ini karena sederhana, murah, dan memungkinkan adanya interoperabilitas antar divais. Sistem operasi biasanya menggunakan nama simbolik untuk menunjukkan *port* serial dari sebuah komputer. Sistem Unix, tentunya termasuk Linux melabelkan divais *port* serial sebagai /dev/tty* (tty kependekan dari *teletype*) dimana * adalah string yang mengidentifikasi divais terminal. Sintaks untuk string tersebut tergantung pada sistem operasi dan divais itu sendiri. Misalnya /dev/ttyS1

merepresentasikan divais *port* serial di sistem operasi Linux. Sedangkan di lingkungan sistem operasi Microsoft Windows, biasanya *port* serial dilabelkan dengan COM1, COM2, dan seterusnya tergantung jumlah *port* serial yang ada di komputer tersebut. Aplikasi atau divais yang sering digunakan melalui *port* serial diantaranya adalah mikrokontroler atau EPROM, GPS *receiver*, modem *dial-up*, dan *bar code scanner*.

Ada beberapa hal yang perlu diperhatikan dalam mengatur koneksi serial yang digunakan untuk komunikasi asinkronus *start-stop*, diantaranya:

- Kecepatan (*speed*)

Port serial menggunakan pensinyalan biner, jadi *data rate* yang digunakan lebih merupakan *symbol rate* dalam *baudrate.* Kecepatan dari *port* dengan kecepatan dari divais yang memakai *port* tersebut sebagai jalur komunikasi dengan komputer harus sesuai.

- *Data bits*

Jumlah dari *data bits* dalam setiap karakter dapat 5 (untuk kode Baudot), 6 (jarang digunakan), 7 (untuk ASCII sebenarnya), 8 (untuk segala macam data, karena sesuai dengan ukuran dari satu byte), atau 9 (jarang digunakan). *Data bits* dengan jumlah 8 bits biasanya lebih banyak digunakan secara universal dan untuk berbagai macam aplikasi terbaru. *Data bits* dengan jumlah 5 bits atau 7 bits biasanya hanya untuk perangkat yang lebih tua seperti *teleprinter*.

- *Parity bit*

Parity adalah sebuah metode untuk mendeteksi terjadinya kesalahan dalam proses transmisi. Dalam penggunaannya pada *port* serial, satu *data bit* tambahan dikirim dengan setiap data karakter, diatur sedemikian rupa sehingga jumlah bit 1 di setiap karakter termasuk *parity bit* dapat selalu genap atau ganjil. Jika satu byte diterima dengan jumlah bit 1 yang salah, maka data tersebut pasti sudah rusak. Jika *parity* benar maka telah ada kesalahan dalam jumlah genap. *Parity bit* dalam setiap karakter dapat diset menjadi *none* (N), *odd* (O), *even* (E), *mark* (M), atau *space* (S). *None* berarti tidak ada *parity bit* yang dikirimkan sama sekali. *Mark* berarti *parity bit* selalu diset untuk kondisi sinyal *mark* (logika 1) dan begitu juga dengan *space* yang selalu mengirimkan *parity bit* di kondisi sinyal *space*.

Walaupun demikian dua macam bentuk *parity bit* tersebut tidak menambahkan informasi kesalahan. *Odd parity* lebih umum dibandingkan dengan *even parity*, karena lebih menjamin paling tidak satu transisi keadaan terjadi di setiap karakter, yang membuat *parity bit* ini lebih dapat diandalkan. Akan tetapi *parity bit* yang paling banyak digunakan adalah *none*, dimana deteksi kesalahan ditangani oleh protokol komunikasi.

- *Stop bits*

Stop bits yang dikirimkan pada akhir setiap karakter memungkinkan perangkat keras penerima sinyal untuk mendeteksi akhir dari sebuah karakter dan untuk meresinkronisasi dengan aliran karakter. Divais-divais elektronik umumnya menggunakan satu *stop bit*.

Notasi umum yang menspesifikasikan hubungan serial biasanya yaitu D/P/S, dimana D adalah *data bits*, P adalah *parity bit*, dan S adalah *stop bit(s)*. Yang paling umum penggunaannya adalah 8/N/1 (8N1) yang menyatakan ada 8 *data bits*, tidak ada *parity bit*, dan ada 1 *stop bit*.

2.2.3. *In-System Programming* (ISP)

In-System Programming (disingkat ISP) adalah kemampuan dari beberapa divais *programmable logic*, mikrokontroler, dan *programmable chip* elektronik lain untuk dapat diprogram selama diinstalasikan dalam sebuah sistem yang lengkap [13].

Kelebihan utama dari fitur ini adalah memungkinkan pabrikan divais elektronik untuk mengintegrasikan pemrograman dan pengetesan dalam sebuah fase produksi, daripada memerlukan tahap pemrograman terpisah sebelum melakukan proses *assembly* sistem. Hal ini juga memungkinkan banyak pabrikan untuk memprogram *chip* atau perangkat *programmable logic* lainnya sesuai dengan tujuan produksi daripada membeli *chip* yang sudah terprogram. Tentunya hal ini memungkinkan untuk mengubah atau memasukkan kode program atau perubahan desain di tengah-tengah proses produksi.

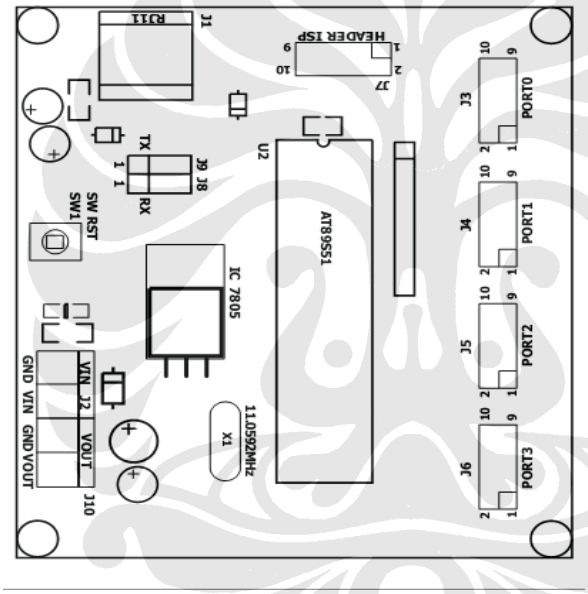
Biasanya *chip* yang mendukung ISP mempunyai sirkuit internal untuk menghasilkan segala tegangan untuk pemrograman dari sumber tegangan sistem.

Selain itu *chip* ini biasanya berkomunikasi dengan pemrogramnya dengan menggunakan protokol serial.

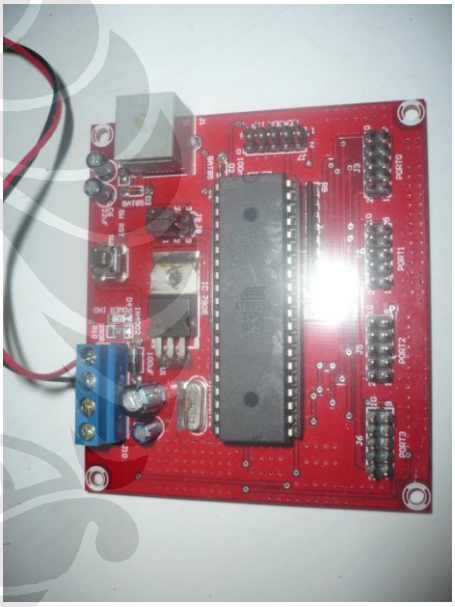
2.3. Perangkat-Perangkat yang Digunakan

Dalam merancang sistem pelacak bus kampus, ada dua perangkat keras khusus yang akan dipakai, yaitu:

2.3.1. Modul DT-51™ Low Cost Micro System (LCMS) versi 2.0 dengan Mikrokontroler Atmel AT89S51



Gambar 2.2(a). Blok diagram modul DT-51™ LCMS versi 2.0 [14]



Gambar 2.2(b). Gambar asli modul DT-51™ LCMS versi 2.0

Spesifikasi dari modul DT-51™ LCMS versi 2.0 yang adalah sebagai berikut [14]:

1. Mikrokontroler AT89S51 dengan 4 kB *flash memory*
2. Mendukung varian MCS-51® 40 pin antara lain: AT89S51, AT89S52, AT89S53, AT89S51, AT89S8252, AT89LS53, dan AT89LS8252
3. Memiliki hingga 32 pin jalur input/output dengan *pull-up*

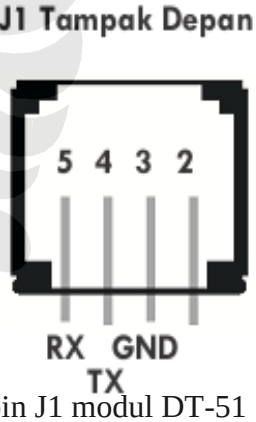
4. Rangkaian RC reset, tombol reset, serta *brown-out detector*
5. Frekuensi osilator sebesar 11,0592 MHz
6. Tersedia jalur komunikasi serial UART RS-232 yang telah disempurnakan, dengan konektor RJ11
7. Tersedia *port* untuk pemrograman secara ISP
8. Tegangan input 9-12 Volt DC pada Vin dan tegangan output 5 Volt DC pada Vout

Penjelasan dan gambar alokasi dari pin-pin yang dipakai modul adalah sebagai berikut:

1. Pin J1

Pin ini berfungsi sebagai jalur komunikasi serial RS-232, memakai konektor RJ11, dengan perangkat lain atau komputer. Untuk menghubungkan modul DT-51™ LCMS versi 2.0 dengan komputer secara serial dengan menggunakan pin J1 ini, maka diperlukan kabel serial dengan ujung konektor RJ11 untuk modul ini dan ujung lainnya konektor DB9 untuk *port* COM (*port* serial pada komputer).

COM port Komputer DB9	DT-51 Low Cost Micro System v2.0 J1
RX (pin 2)	RX (pin 5)
TX (pin 3)	TX (pin 4)
GND (pin 5)	GND (pin 3)



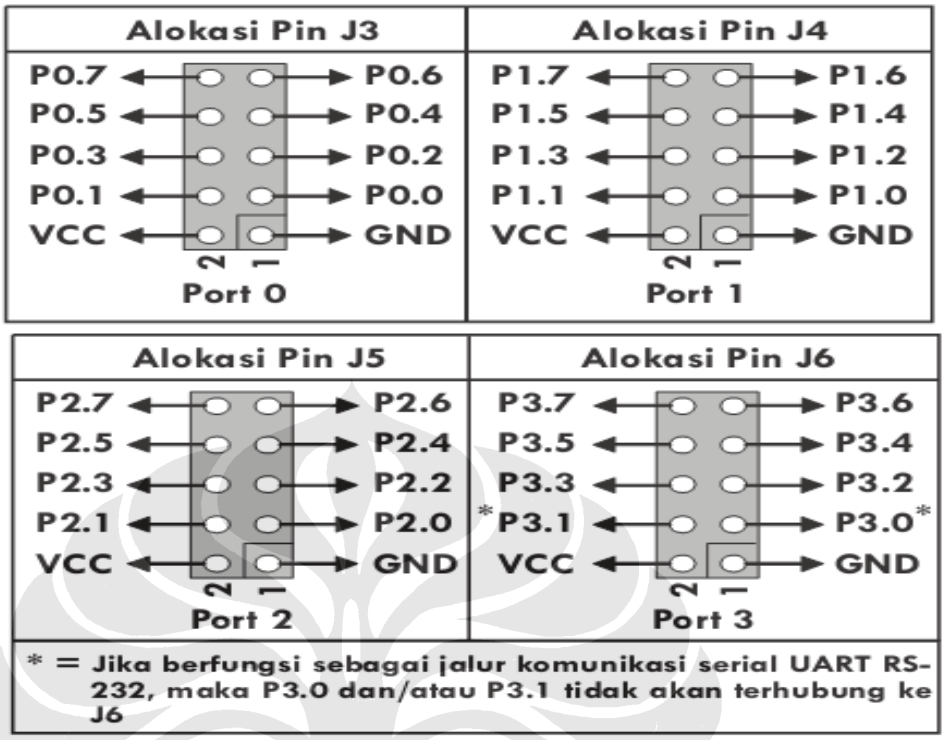
Gambar 2.3. Alokasi RX, TX, dan ground pada pin J1 modul DT-51 LCMS versi 2.0 dan *port* COM komputer
[14]

2. Pin J2

Pin yang terdiri dari Vcc_{in} dan Gnd_{in} ini sebagai jalur untuk sumber tegangan 9-12 Volt DC.

3. Pin J3-J6 (*port* 0-*port* 3)

Pin-pin ini merupakan persambungan dari pin-pin mikrokontroler Atmel AT89S51 yang merupakan *port* dua arah input dan output.



Gambar 2.4. Blok diagram alokasi pin-pin J3 sampai J6 modul DT-51 LCMS versi 2.0
[14]

Berikut penjelasan masing-masing *port* [14]:

- *Port* 0

Merupakan 8-bit *open drain bidirectional* I/O *port*. Sebagai *port* output, setiap pin dapat melakukan *sink* pada 8 input TTL. Saat bit-bit 1 ditulis ke pin-pin *port* 0, pin-pin tersebut dapat digunakan sebagai input berimpedansi tinggi. *Port* 0 ini juga dapat dikonfigurasikan menjadi *low-order address/data bus* yang termultipleks selama akses ke program eksternal dan memori data. Dalam mode ini, *port* 0 mempunyai *internal pull-ups*. *Port* 0 juga menerima kode dalam bentuk byte selama *flash programming* dan mengeluarkan kode berbentuk byte tersebut selama verifikasi program. *External pull-ups* dibutuhkan selama verifikasi program.

- *Port* 1

Merupakan 8-bit *bidirectional* I/O *port* dengan *internal pull-ups*. *Output buffers* dari *port* 1 dapat melakukan *sink/source* 4 input TTL. Saat bit-bit 1 ditulis

ke pin-pin *port* 1, bit-bit tersebut mengalami proses *pulled high* oleh *internal pull-ups* dan dapat digunakan sebagai input. Sebagai input, pin-pin *port* 1 yang secara eksternal mengalami proses *pulled low* akan melakukan *source* arus (I_{IL}) karena proses *internal pull-ups*. *Port* 1 juga menerima *low-order address bytes* selama *flash programming* dan verifikasi program.

Tabel 2.1. Pin 1.5-1.7 pada *port* 1 serta salah satu alternatif fungsinya

Port Pin	Alternate Functions
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

[15]

- *Port* 2

Merupakan 8-bit *bidirectional* I/O *port* dengan *internal pull-ups*. *Output buffers* dari *port* 2 dapat melakukan *sink/source* 4 input TTL. Saat bit-bit 1 ditulis ke pin-pin *port* 2, bit-bit tersebut mengalami proses *pulled high* oleh *internal pull-ups* dan dapat digunakan sebagai input. Sebagai input, pin-pin *port* 2 yang secara eksternal mengalami proses *pulled low* akan melakukan *source* arus (I_{IL}) karena proses *internal pull-ups*. *Port* 2 juga menerima *low-order address bytes* selama *flash programming* dan verifikasi program.

- *Port* 3

Merupakan 8-bit *bidirectional* I/O *port* dengan *internal pull-ups*. *Output buffers* dari *port* 3 dapat melakukan *sink/source* 4 input TTL. Saat bit-bit 1 ditulis ke pin-pin *port* 3, bit-bit tersebut mengalami proses *pulled high* oleh *internal pull-ups* dan dapat digunakan sebagai input. Sebagai input, pin-pin *port* 3 yang secara eksternal mengalami proses *pulled low* akan melakukan *source* arus (I_{IL}) karena proses *internal pull-ups*. *Port* 3 juga menerima *low-order address bytes* selama *flash programming* dan verifikasi program. *Port* 3 juga melayani fungsi dari beragam fitur spesial dari AT89S51, seperti ditunjukkan oleh tabel berikut:

Tabel 2.2. Pin 3.0-3.7 pada *port* 3 serta salah satu alternatif fungsinya

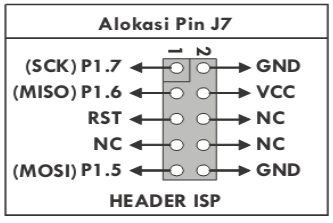
Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

[15]

Port 3 ini pada sistem pelacak bus kampus akan dipakai sebagai jalur komunikasi serial dengan modul *wireless*. Penjelasan lebih lanjut akan dibahas di bab selanjutnya.

4. Pin J7

Pin ini merupakan *header* ISP yang berfungsi sebagai jalur untuk memprogram mikrokontroler. Pemrograman dapat dilakukan dengan menghubungkan *header* ISP ini ke *port* paralel komputer menggunakan kabel yang disediakan bersama modul DT-51™ LCMS versi 2.0. Bahasa pemrograman yang dipakai adalah bahasa tingkat rendah, misalnya bascom (basic compiler) atau assembly. Untuk "menginjeksikan" program ke dalam mikrokontroler Atmel AT89S51 diperlukan *software* ISP yang memerlukan program yang akan di-*load* dalam format .HEX (format Intel). *Software* ini, yaitu *software* ISP dari Atmel sendiri, juga disertakan bersama modul DT-51™ LCMS versi 2.0. Untuk penjelasan lebih lanjut akan dibahas di bab selanjutnya.



Gambar 2.5. Blok diagram alokasi pin J7 modul DT-51 LCMS versi 2.0

[14]

4. Pin J8 dan J9

Pin-pin ini cenderung berfungsi sebagai *jumper* jika kita ingin memakai pin P3.0 dan P3.1 sebagai jalur komunikasi serial (tidak menggunakan pin J1) baik UART RS-232 atau UART TTL. Berikut ini blok diagram tata letak *jumper* jika ingin memakai pin P3.0 dan P3.1 sebagai jalur komunikasi serial:

Gambar 2.6. Blok diagram alokasi pin-pin J8 dan J9 modul DT-51 LCMS versi 2.0

[14]

Untuk pengaturan tata letak *jumper* pada pin-pin ini untuk sistem pelacak bus kampus akan dibahas di bab selanjutnya.

2.3.2. Modul *Wireless* YS-1020UA RF Data Transceiver

Gambar 2.7. Modul *wireless* YS-1020UA RF Data Transceiver

[16]

Modul *wireless* YS-1020UA RF Data Transceiver didesain untuk sistem pengiriman data secara *wireless* dalam jarak yang dekat. Modul ini bekerja pada pita frekuensi ISM, pengiriman dan penerimaan terintegrasi secara *half duplex*. Modul ini dapat terhubung langsung dengan prosesor-prosesor monolitik, komputer, divais-divais RS-485 dan komponen-komponen UART lainnya dengan

port antarmuka RS-232, RS-485, dan UART/TTL [16]. Beberapa parameter penting modul *wireless* ini antara lain adalah [16]:

1. Fitur-fitur utama

- Frekuensi *carrier*: 433 MHz, 450 MHz, dan 868 MHz (opsional) atau opsional ISM lainnya
- Antarmuka: RS-232, RS-485, TTL (opsional)
- Multikanal: 8 kanal, dapat diperluas untuk 16 atau 32 kanal
- *Baud rate* di udara: 1200 bps, 2400 bps, 4800 bps, 9600 bps, 19200 bps, atau 38400 bps
- Transmisi data *transparent*, dimana data yang diterima sesuai dengan data yang telah dikirim
- Format antarmuka: 8N1, 8E1, atau 801 dan dapat dikustomisasi untuk format antarmuka lainnya
- Modulasi: GFSK (*Gaussian Frequency Shift Keying*) sehingga sangat anti-terinterferensi dan BER (*Bit Error Rate*) yang rendah
- *Half duplex*: Intregasi dari *receiver* dan *transmitter*
- Komsumsi daya yang rendah dan ada fungsi *sleep*
- *Range* temperature kerja: -35^{o}C sampai 75^{o}C (-31 F sampai 167 F)
- Kelembaban kerja: 10%-90% kelembaban relatif tanpa kondensasi
- Impedansi: 50 Ω
- Mengikuti standar EN 300220 dan ARIB STD-T67

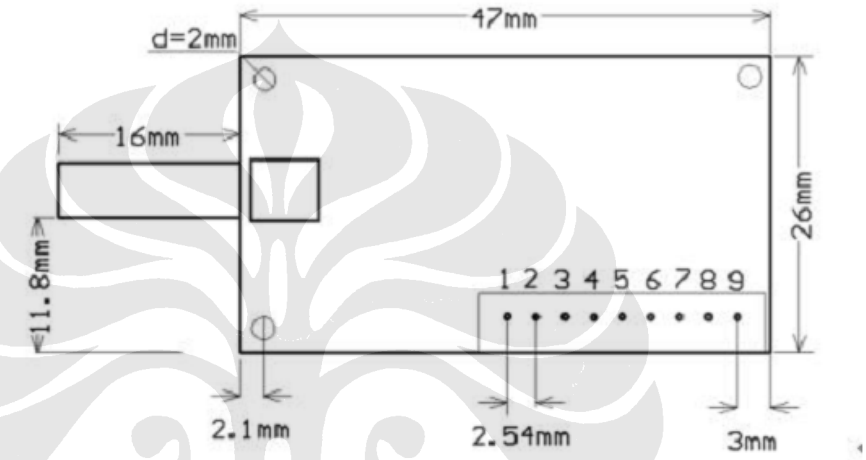
2. Spesifikasi

- Daya RF: ≤10 mW/ 10 dBm
- Arus saat menerima: ≤25 mA
- Arus saat mengirim: ≤40 mA
- Arus saat dalam keadaan *sleep*: ≤20 μA
- *Power supply*: 5 Volt atau 3,3 Volt DC
- Sensitivitas penerimaan: -115 dBm pada 9600bps dan -120 dBm pada 1200bps
- Ukuran: 47 mm x 26 mm x 10 mm (tanpa *port* antena)

- Jangkauan: ≤0,5 m, BER=10^{-3}, *baudrate* 9600bps, saat antena 2 m diatas permukaan tanah di area terbuka; dan ≤0,8 m, BER=10^{-3}, *baudrate* 1200bps, saat antena 2 m diatas permukaan tanah di area terbuka

3. Dimensi dan definisi antarmuka

   Dimensi dari modul *wireless* YS-1020UA RF Data Transceiver dapat dilihat sebagai berikut:



Gambar 2.8. Dimensi modul *wireless* YS-1020UA RF Data Transceiver [16]

Sedangkan untuk penjelasan antarmukanya dapat dilihat dari tabel berikut:

Tabel 2.3. Deskripsi dari pin-pin yang ada pada modul *wireless* YS-1020UA RF Data Transceiver

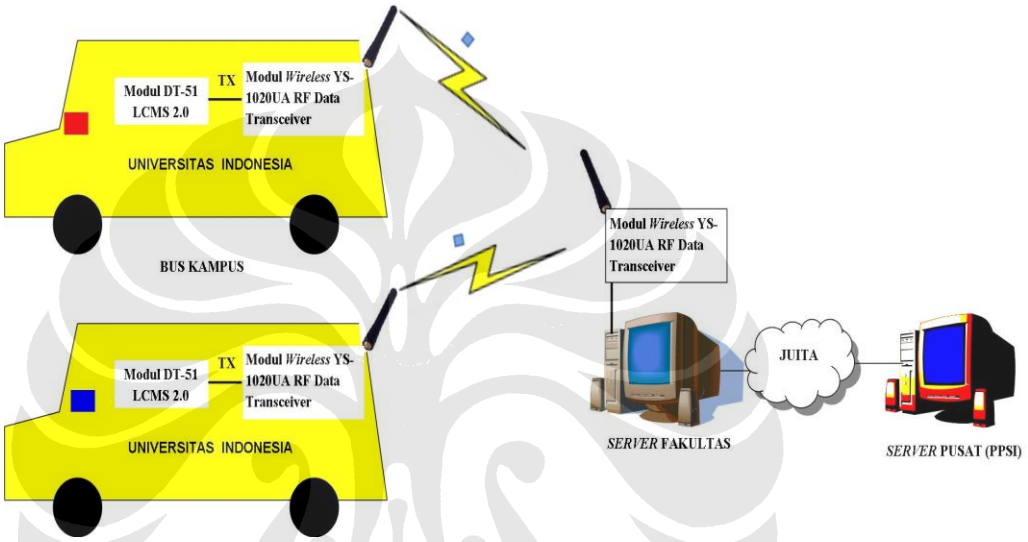
Pin No.	Pin name	Description	Level	Connection with terminal	Remands
1	GND	Grounding of power supply		Ground	
2	Vcc	Power supply DC	+3.3~5.5V		
3	RXD/TTL	Serial data receiving end	TTL	TxD	
4	TXD/TTL	Serial data transmitting end	TTL	RxD	
5	DGND	Digital grounding			
6	A(TXD)	A of RS-485 or TXD of RS-232		A(RxD)	
7	B(RXD)	B of RS-485 or RXD of RS-232		B(TxD)	
8	Sleep	Sleep control (input)	TTL	Sleep signal	Low level sleep
9	Test	Ex-factory testing			

[16]

BAB 3
SISTEM PELACAK BUS KAMPUS MENGGUNAKANMODUL DT-51™ LCMS 2.0 DAN *WIRELESS* YS-1020UA RF

3.1. Gambaran Umum Sistem Pelacak Bus Kampus

Secara garis besar, keadaan sebenarnya dari sistem pelacak bus kampus yang dirancang untuk tiap-tiap fakultas/halte adalah sebagai berikut:



Gambar 3.1. Gambaran umum sistem pelacak bus kampus untuk tiap-tiap fakultas atau halte dimana ada bus yang terdeteksi

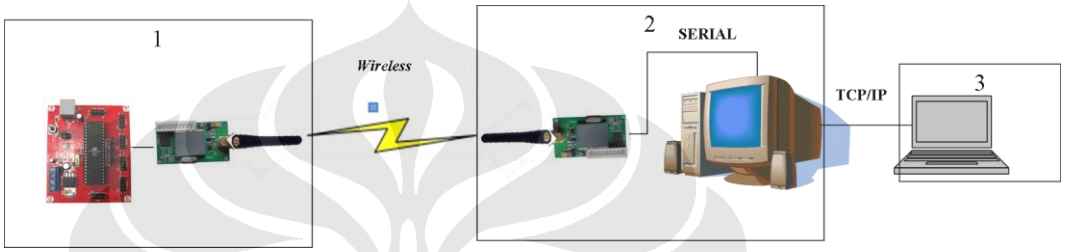
Dari skema tersebut, maka perangkat-perangkat yang dibutuhkan masing-masing untuk bus (bagian 1) dan fakultas/halte (bagian 2), yaitu:

1. Bagian bus kampus (bagian 1):
   - Modul DT-5 LCMS mikrokontroler AT89S51 yang berfungsi memberikan ID berupa data berbentuk biner (0 dan 1).
   - Modul *wireless* YS 1020 RF Data Tranceiver dan antena yang berfungsi sebagai penghubung dengan bagian 2.
2. Bagian fakultas/halte (bagian 2):
   - Modul *wireless* YS 1020 RF Data Tranceiver dan antena yang berfungsi sebagai penghubung dengan bagian 1.
   - Sebuah komputer yang berfungsi sebagai pusat penerima data, yang sudah terinstal program pengolah data dari *port* serial yang selanjutnya akan

dikirimkan ke *server* pusat di PPSI melalui Jaringan UI Terpadu (JUITA). Masing-masing fakultas mempunyai komputer dengan fungsi tersebut.

3. Bagian *server* di PPSI selanjutnya dapat menampilkan data dari fakultas menjadi bentuk *web* yang menampilan GUI sederhana dari peta UI Depok.

Gambaran umum diatas merupakan keadaan sistem yang sebenarnya, namun karena keterbatasan perangkat, sumber daya manusia, dan waktu maka sistem yang akan dibuat adalah sebagai berikut:

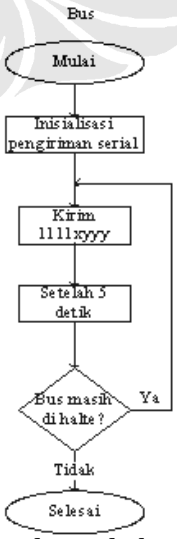


Gambar 3.2. Skema sistem pelacak bus kampus yang akan dibuat

Dari gambar tersebut, fungsi dari masing-masing bagian sama seperti yang telah dijelaskan sebelumnya. Untuk tahapan perancangan dan pengintegrasian sistem pelacak bus kampus ini akan dibahas di sub bab selanjutnya.

3.2. Pemrograman Modul DT-51™ LCMS 2.0

Seperti sudah dibahas sebelumnya, modul DT-51 LCMS 2.0 ini berfungsi untuk memberikan ID untuk bus kampus. Untuk lebih jelas dapat dilihat dari diagram alir berikut:



Gambar 3.3. Diagram alir pada bagian bus kampus

Dari diagram alir tersebut, dapat dilihat ada tiga proses atau algoritma penting yang terjadi pada bagian bus kampus ini, yaitu:

1. Inisialisasi serial

   Proses ini dimaksudkan untuk mengatur pengiriman data secara serial yang dilakukan oleh modul DT-51 LCMS 2.0. Parameter-parameter yang perlu diperhatikan untuk dapat diprogramkan pada modul DT-51 LCMS 2.0 tersebut adalah:

   Baud rate: 9600bps

   Data bits: 8 bit

   Parity bit: tidak ada (*none*)

   Stop bit: 1 bit

   Pengaturan tersebut merupakan pengaturan yang cukup umum dalam komunikasi serial. Di sisi penerima, yaitu komputer halte/fakultas, juga harus mempunyai parameter-parameter yang sama dengan di sisi pengirim, yaitu modul DT-51 LCMS 2.0, agar komunikasi serial dapat berjalan.

2. Pengiriman data sebanyak 8 bit

   Data yang dikirimkan, sesuai dengan yang sudah diinisialisasikan sebelumnya, adalah sebanyak 8 bit. Data tersebut, misalnya 1111xyyy, akan dikirimkan ke komputer halte/fakultas. Empat bit awal, yaitu 1111 adalah penanda yang mengirimkan data adalah modul DT-51 LCMS pada bus kampus, bukan perangkat lain sehingga dapat dikatakan empat bit awal ini akan lolos verifikasi di komputer halte/fakultas nanti untuk diproses lebih lanjut. Bit x menandakan warna label dari bus kampus. Untuk konvensi seterusnya, bit 0 menandakan bus berlabel merah dan bit 1 menandakan bus berlabel biru. Tiga bit terakhir, yaitu yyy, menandakan nomor dari bus. Maka jumlah maksimal dari bus adalah masing-masing delapan buah bus untuk tiap label.

3. Pengulangan pengiriman data setiap lima detik sekali

   Data bus sebanyak 8 bit tersebut akan dikirimkan ke komputer fakultas setiap lima detik sekali. Hal tersebut dimaksudkan agar dapat memberi jeda dari bus kampus berhenti di satu halte sampai berangkat menuju halte berikutnya. Jeda

selama lima detik tersebut juga dimaksudkan agar dapat mengatasi kasus ada dua buah atau lebih bus kampus pada satu halte. Tentunya jika masing-masing bus mengirimkan data secara terus-menerus dengan jeda yang hampir nol, akan terjadi tabrakan data yang terkirim ke komputer fakultas sehingga tidak akan terbaca sempurna di komputer halte/fakultas.

4. Pengiriman data melalui modul *wireless*

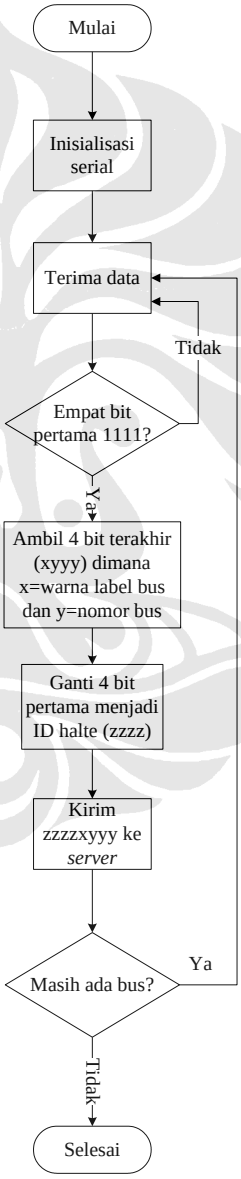
   Data sebanyak 8 bit yang akan dikirimkan setiap lima detik sekali tersebut akan melalui modul *wireless* YS 1020 RF Data Transceiver, yang dalam hal ini berfungsi sebagai *transmitter*, untuk diteruskan agar dapat ditangkap oleh modul *wireless* yang berfungsi sebagai receiver di sisi penerima, yaitu komputer halte/fakultas. Komunikasi yang terjadi antara modul DT-51 LCMS dan dengan modul *wireless* YS 1020 RF Data Transceiver juga secara serial. Pin J9 pada modul DT-51 LCMS sebagai *jumper* dipasangkan ke mode TX. Konfigurasi parameter-parameter komunikasi serial pada modul *wireless* YS 1020 RF Data Transceiver ini sudah sama dengan modul DT-51 LCMS sehingga langsung dapat berkomunikasi secara serial. *Plug* TX dan RX pada modul *wireless* masing-masing disambungkan pada pin 3.0 dan pin 3.1 *port* 3 pada modul DT-51 LCMS. Sedangkan untuk sumber tegangan yaitu *plug* Vcc dan Gnd dapat disambungkan pada pin Vcc dan Gnd pada *port* mana saja modul DT-51 LCMS. Tegangan keluaran yang dapat dihasilkan oleh modul DT-51 LCMS adalah sebesar 5 Volt DC.

Untuk poin 1 sampai 3 pada penjelasan di atas masuk dalam program yang dituliskan ke dalam modul DT-51 LCMS. Untuk memprogram modul DT-51 LCMS, modul dipasang dalam keadaan siap diprogram yaitu dengan menghubungkan *header* ISP (pin J7) dari modul ke *port* paralel komputer menggunakan kabel yang telah disediakan bersama modul ini yang memang berfungsi sebagai kabel untuk ISP. Selanjutnya dengan menjalankan ISP *software* dari Atmel, lalu me-*load file* berbentuk .HEX yang yang berisi program yang sudah dibuat. ISP *software* ini lalu "menuliskan" program yang telah dibuat ke dalam mikrokontroler. Rincian program dapat dilihat pada Lampiran 1 dan 2. Ada dua program untuk modul DT-51 LCMS dimaksudkan agar ada contoh kasus dua

bus dengan label dan nomor yang berbeda berhenti di suatu halte. Bus pertama berlabel merah dan bernomor 5 sedangkan bus kedua berlabel biru dan bernomor 3.

3.3. Pembuatan Program Berbahasa C untuk Komputer Halte/Fakultas

Untuk dapat memproses dan mengolah data yang dikirimkan oleh modul DT-51 LCMS secara serial serta mengirimkan data baru ke *server* melalui protokol TCP/IP dapat dilihat dari diagram alir berikut:



Gambar 3.4. Diagram alir pada bagian halte/fakultas

Dari diagram alir tersebut, dapat dilihat ada enam proses atau algoritma penting yang terjadi pada bagian komputer halte/fakultas ini, yaitu:

1. Inisialisasi serial

   Proses ini dimaksudkan untuk mengatur penerimaan data secara serial yang dilakukan oleh modul DT-51 LCMS 2.0 melalui modul *wireless* YS 1020 RF Data Transceiver yang berperan sebagai *transmitter* yang diterima oleh modul *wireless* YS 1020 RF Data Transceiver yang berperan sebagai *receiver* yang terhubung ke *port* serial komputer halte/fakultas. Parameter-parameter yang harus diperhatikan untuk komunikasi serial di komputer halte/fakultas ini sama dengan parameter-parameter pada modul c, yaitu:

   Baud rate: 9600bps

   Data bits: 8 bit

   Parity bit: tidak ada (*none*)

   Stop bit: 1 bit

2. Terima data

   Komputer halte/fakultas yang terhubung dengan modul *wireless* YS 1020 RF Data Transceiver yang berperan sebagai *receiver* menerima semua data yang dapat ditangkap oleh modul *wireless* tersebut.

3. Pengecekan data

   Data yang masuk akan diperiksa oleh program di komputer halte/fakultas apakah data tersebut valid (merupakan data dari bus kampus). Caranya adalah dengan memeriksa empat bit pertama data. Jika empat bit awalnya adalah 1111, maka akan dilanjutkan ke proses selanjutnya. Namun jika data yang diterima tidak mempunyai empat bit awal 1111, maka proses akan dikembalikan pada penerimaan data hingga data yang diterima valid.

4. Pengekstrakkan warna label bus dan nomor bus

   Jika data yang diterima sudah valid, yaitu data sejumlah 8 bit dengan empat bit awalnya adalah 1111, maka empat bit setelahnya misalkan xyyy akan diambil. Huruf x menandakan warna label bus dan huruf y menyatakan nomor bus. Jadi misalkan empat bit terakhirnya adalah 0101 menandakan bahwa bus berlabel merah dan bernomor 5.

5. Penambahan nomor halte

   Setelah warna label dan nomor bus diketahui, maka untuk proses selanjutnya adalah menambahkan empat bit lagi di depan empat bit xyyy, yang merupakan warna label dan nomor bus, untuk menjadi nomor halte. Empat bit tersebut misalkan zzzz. Alokasi empat bit untuk nomor halte ini disebabkan jumlah halte bus kampus yang sesungguhnya adalah 15 halte sehingga butuh empat bit untuk dapat menomori semua halte yang dilewati oleh bus kampus. Berikut ini alokasi bit-bit untuk penomoran halte yang ada di kampus UI Depok:

Tabel 3.1. Pengalokasian bit-bit untuk masing-masing halte yang ada di UI

No.	Nama fakultas/halte	ID (zzzz)
1	Asrama UI	0000
2	Gerbatama	0001
3	Stasiun UI	0010
4	Fakultas Psikologi	0011
5	Fakultas Ilmu Sosial dan Politik	0100
6	Fakultas Ilmu Budaya	0101
7	Fakultas Ekonomi	0110
8	Fakultas Teknik	0111
9	Stadion UI	1000
10	Politeknik	1001
11	Fakultas Matematika dan Ilmu Pengetahuan Alam	1010
12	Fakultas Kesehatan Masyarakat	1011
13	Pondok Cina	1100
14	Masjid UI	1101
15	Fakultas Hukum	1110

6. Pengiriman data yang sudah diolah ke *server*

   Setelah data dari bus diolah oleh program di komputer halte/fakultas sehingga menjadi delapan bit yang empat bit awalnya mewakili nomor halte, bit kelima mewakili warna label bus, dan tiga bit sisanya mewakili nomor bus, maka data hasil olahan tersebut dikirimkan ke *server*. Komunikasi yang dilakukan oleh komputer halte/fakultas dan *server* melalui protokol TCP/IP. Data

tersebut juga dikirim secara terus-menerus hingga tidak ada lagi bus yang ada di halte.

7. Pengecekan bus

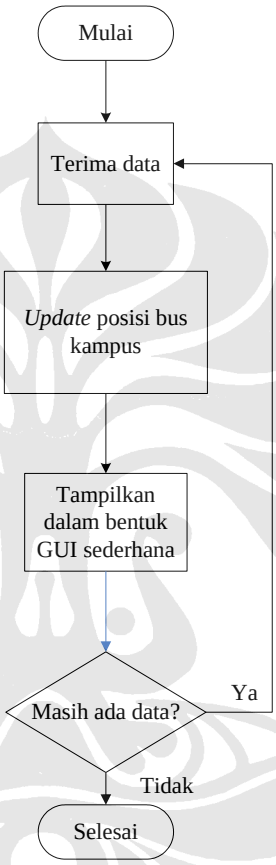
   Setelah semua proses di atas selesai, maka program di komputer halte/fakultas akan memeriksa apakah ada bus kampus lain yang sampai di halte. Jika tidak, maka program di komputer halte/fakultas akan terus menunggu hingga ada bus di halte tersebut.

Di komputer halte/fakultas ini sudah terinstal program yang dibuat dengan menggunakan bahasa C. Di program ini ada beberapa bagian yang penting, yaitu membuka *port* serial, membaca data dari *port* serial, mengolahnya seperti yang telah dijelaskan sebelumnya, dan mengirimkan data lengkap yang memuat nomor halte, warna label bus, dan nomor bus ke *server* melalui protokol TCP/IP. Untuk rincian program lebih jelas, ada pada lampiran 2. Untuk mengecek apakah data yang diterima dari modul DT-51 LCMS 2.0 benar, maka pada program diberikan fungsi yang dapat memperlihatkan nomor bus dan warna label bus yang sedang ada di halte. Berikut ini contohnya masing-masing untuk bus label merah nomor 5 dan bus label biru nomor 3:

Gambar 3.5. Tampilan di cygwin C *compiler* agar data yang diterima dari modul DT-51 LCMS 2.0 dapat dicek.

3.4. Pembuatan Program Berbahasa C untuk Komputer *Server*

Pada sisi *server* yang merupakan sisi yang bisa langsung dapat berinteraksi dengan pengguna, dibutuhkan sebuah program yang dibuat menggunakan bahasa C. Diagram alir yang menunjukkan proses atau algoritma yang terjadi di *server* adalah sebagai berikut:



Gambar 3.6. Diagram alir pada *server*

Dari diagram alir tersebut, dapat dilihat ada tiga proses atau algoritma penting yang terjadi pada bagian komputer halte/fakultas ini, yaitu:

1. Terima data

   Komputer *server* yang sudah terinstal program yang dibuat menggunakan bahasa C ini menerima data melalui protokol TCP/IP. Data dari komputer halte/fakultas ini diterima secara terus-menerus oleh *server* hingga data yang dikirim oleh komputer halte/fakultas berhenti yang berarti tidak ada bus di halte.

2. Menampilkan dalam bentuk GUI sederhana

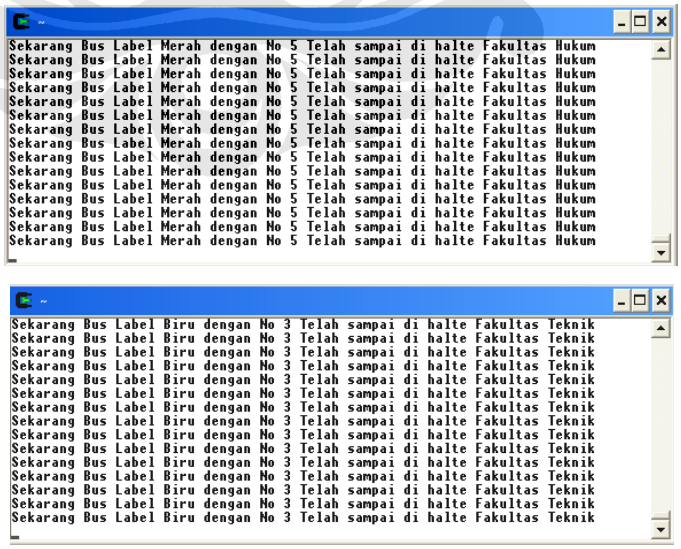
   Setelah data diterima oleh *server*, maka data tersebut akan diolah sehingga dapat direpresentasikan dalam bentuk GUI sederhana yang akan ditampilkan dan dapat dibaca oleh pengguna dengan mudah.

3. Pengecekan data yang diterima

   Pada proses ini, *server* akan terus menerima data dari komputer halte/fakultas hingga data yang dikirim oleh komputer halte/fakultas berhenti yang berarti tidak ada bus di halte.

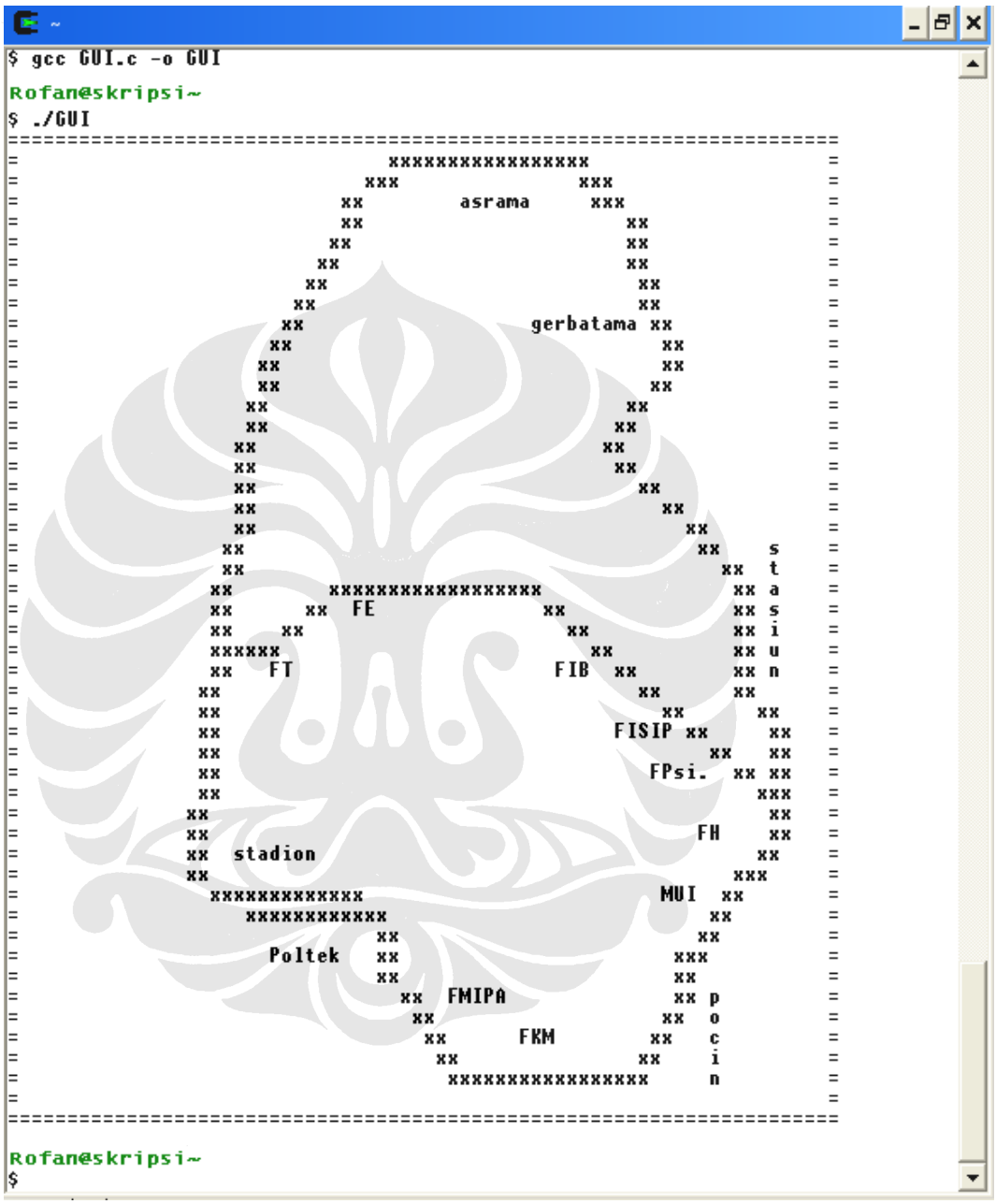
Dalam program yang dibuat menggunakan bahasa C untuk *server* ini, ada tiga bagian program yang penting, yaitu menerima data dari *socket* TCP/IP, memecahnya kembali menjadi tiga bagian (nomor fakultas, warna label bus, dan nomor bus), dan menampilkan data hasil olahan menjadi bentuk GUI sederhana. Rincian program untuk *server* dapat dilihat di lampiran 3.

Untuk mengecek apakah data yang diterima dari program di komputer halte/fakultas benar, maka pada program di *server* diberikan fungsi yang dapat memperlihatkan nomor halte/fakultas, warna label bus, dan nomor bus sehingga dapat terlihat kebenarannya untuk data masukan dari program di komputer halte/fakultas masing-masing adalah bus label merah nomor 5 dari Fakultas Hukum dan bus label biru nomor 3 dari Fakultas Teknik:



Gambar 3.7. Tampilan di cygwin C *compiler* agar data yang diterima dari program komputer halte/fakultas dapat dicek.

Sedangkan untuk GUI sederhana yang dibuat pada program di komputer *server* ini adalah sebagai berikut:

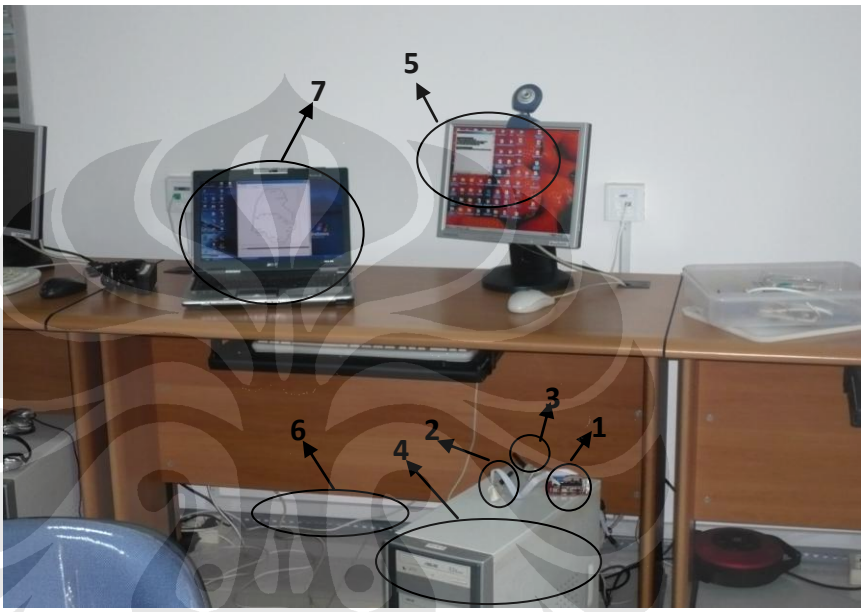


Gambar 3.8. GUI sederhana yang menggambarkan jalur yang dilewati bus kampus

BAB 4
ANALISA SISTEM PELACAK BUS KAMPUS MENGGUNAKAN MODUL DT-51™ LCMS 2.0 DAN *WIRELESS* YS-1020UA RF

4.1. Pengujian Prototipe Sistem Pelacak Bus Kampus

Untuk pengujian sistem, berikut ini adalah foto yang menunjukkan keseluruhan sistem yang diujikan.



Gambar 4.1. Foto sistem keseluruhan yang diujikan

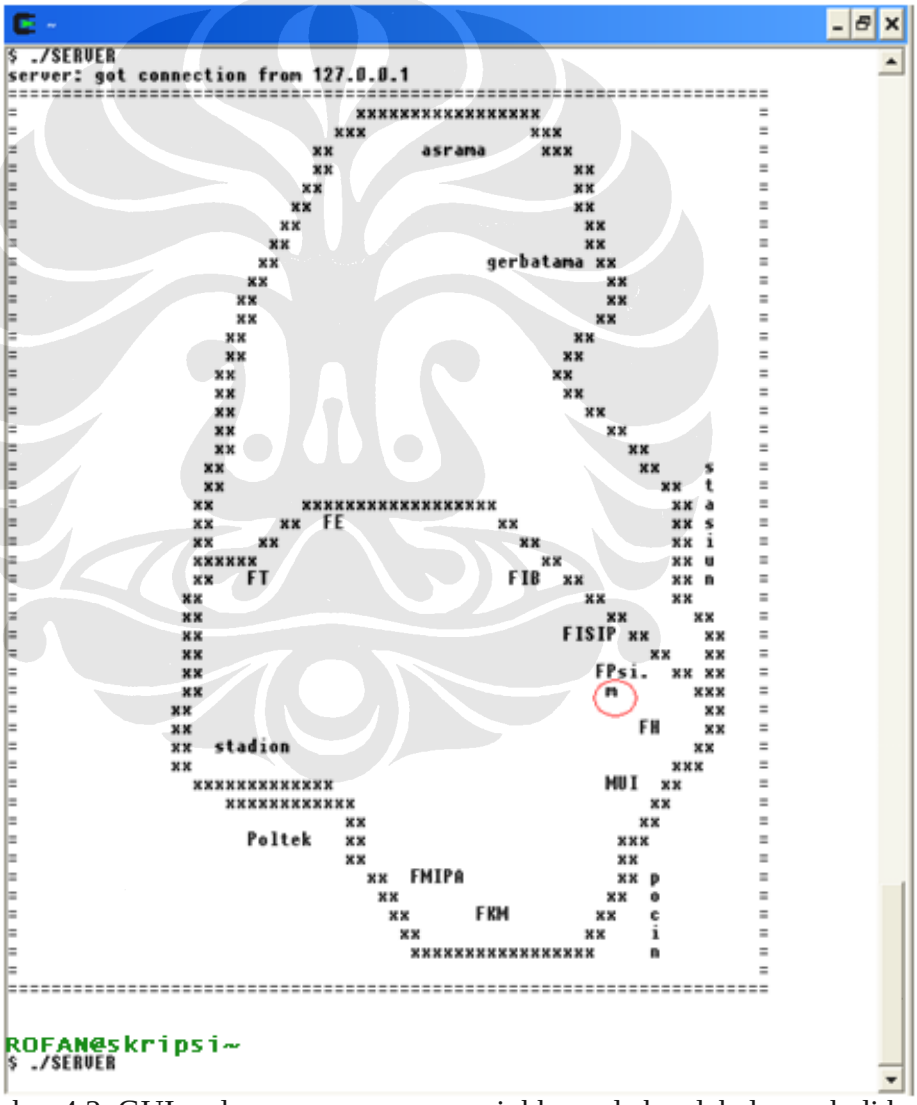
Dari foto tersebut, berikut ini keterangan untuk masing-masing angka:

1. Modul DT-51 LCMS 2.0 sebagai pemberi ID bus kampus
2. Modul *wireless* YS 1020UA RF Data Tranceiver sebagai pengirim ID bus kampus
3. Modul *wireless* YS 1020UA RF Data Tranceiver sebagai penerima ID bus kampus
4. Komputer halte/fakultas sebagai pengolah ID bus dan ditambahkan dengan ID halte untuk dikirim ke *server* melalui TCP/IP
5. Tampilan di komputer halte/fakultas
6. Kabel UTP yang menghubungkan antara komputer halte/ fakultas dengan *server*

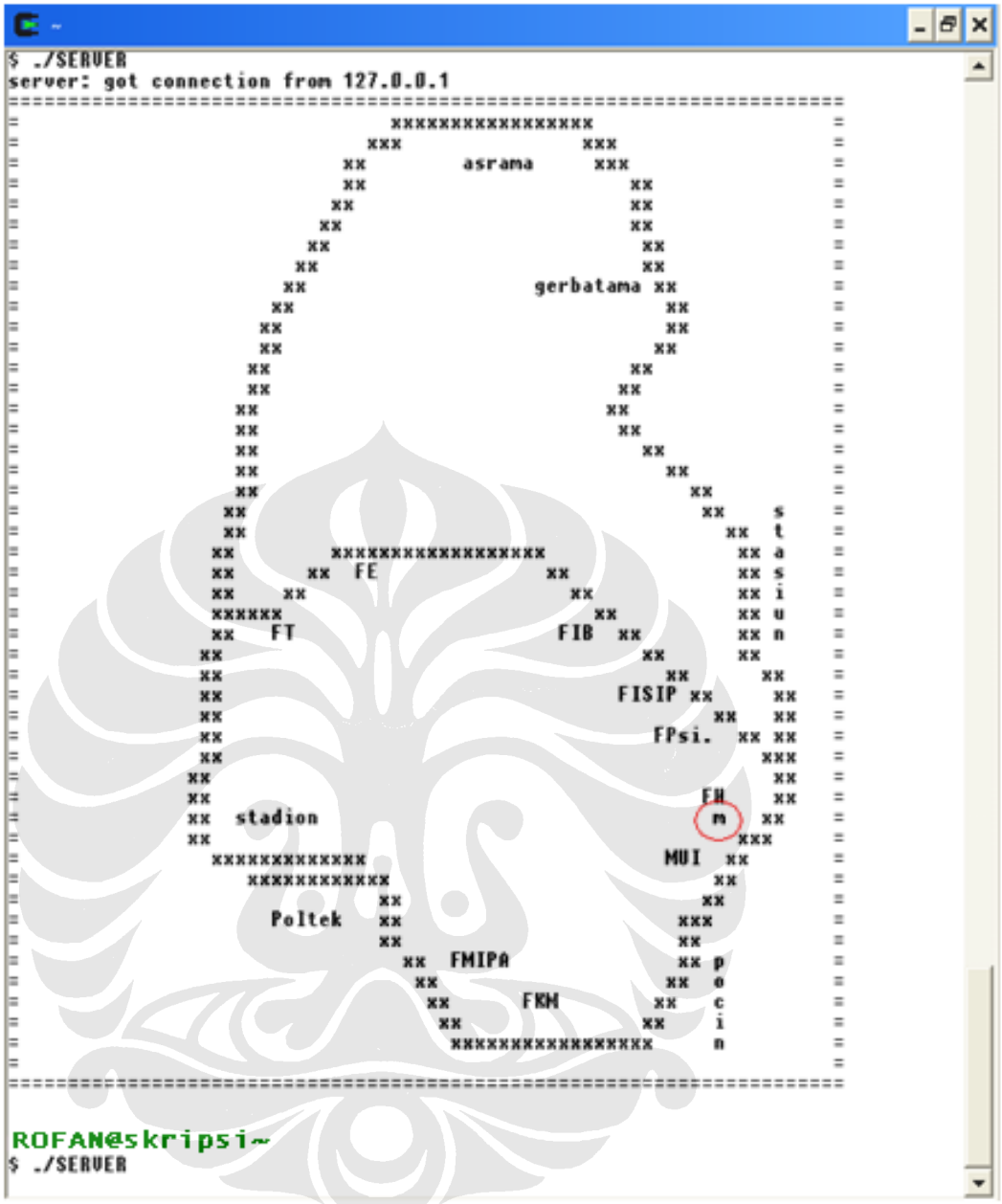
7. *Server* sebagai pengolah data terakhir dan ditampilkan dalam bentuk GUI sederhana

Proses pengujian dari sistem pelacak bus kampus ini dilakukan sebanyak empat kali, dimana diujikan empat sampel kasus, yaitu:

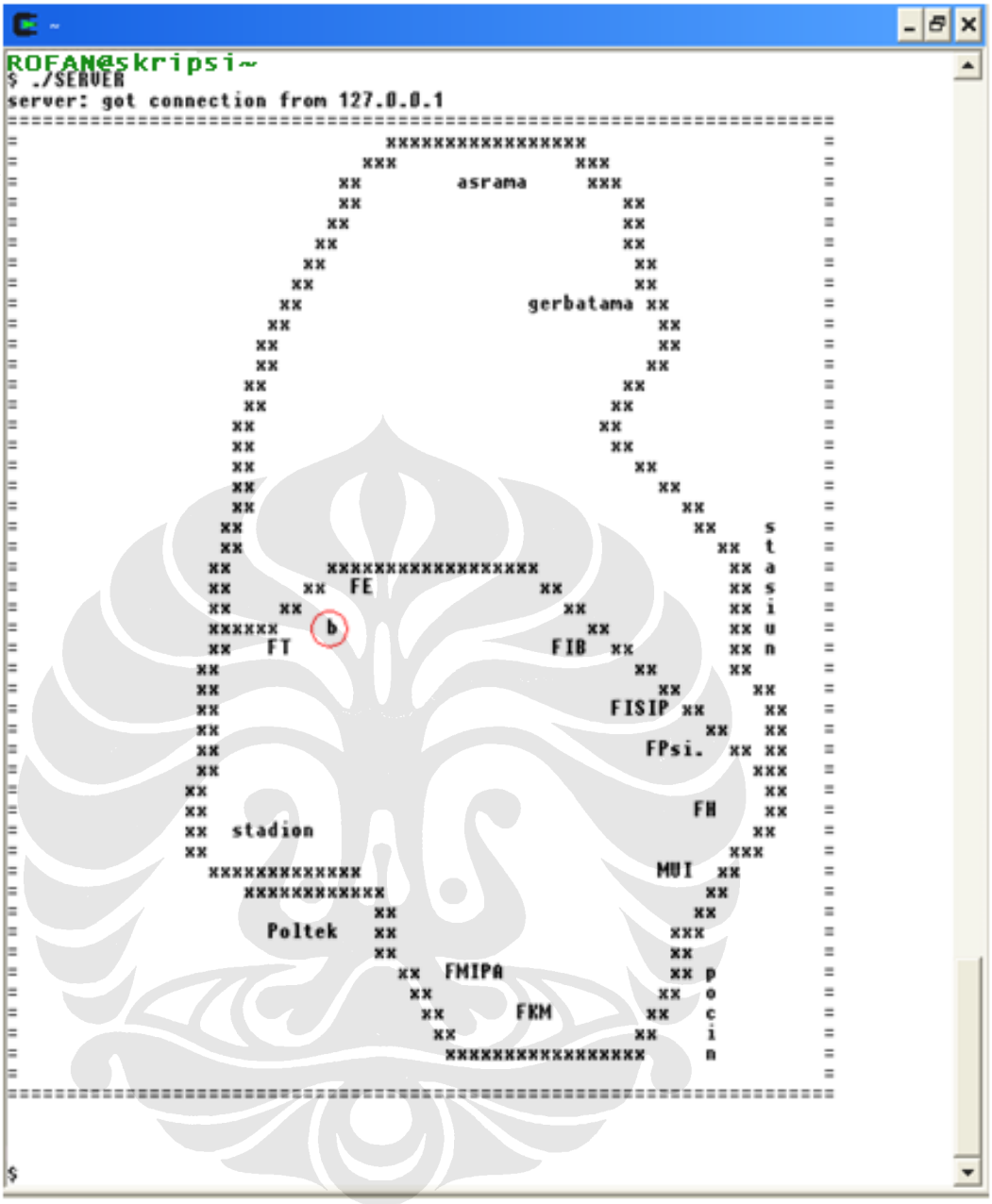
1. Bus label merah berhenti di halte Fakultas Psikologi
2. Bus label merah berhenti di halte Fakultas Hukum
3. Bus label biru berhenti di halte Fakultas Teknik
4. Bus label biru berhenti di halte Fakultas Ekonomi



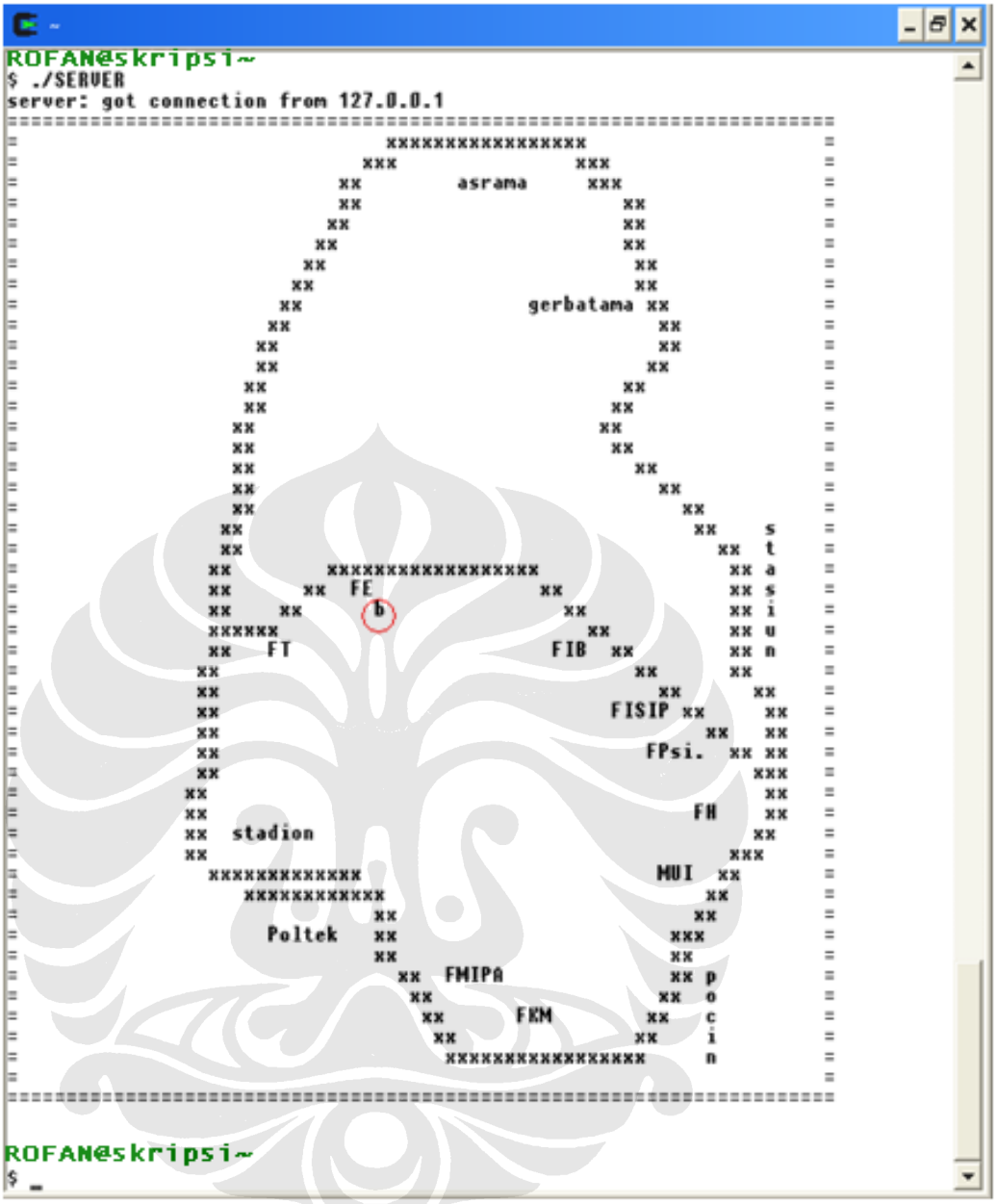
Gambar 4.2. GUI pada *server* yang menunjukkan ada bus label merah di halte Fakultas Psikologi



Gambar 4.3. GUI pada *server* yang menunjukkan ada bus label merah di halte Fakultas Hukum



Gambar 4.4. GUI pada *server* yang menunjukkan ada bus label biru di halte Fakultas Teknik



Gambar 4.5. GUI pada *server* yang menunjukkan ada bus label biru di halte Fakultas Ekonomi

Dari hasil pengujian sistem, dapat dilihat bahwa sistem cukup akurat dalam menampilkan posisi bus kampus untuk empat sampel, namun masih terdapat kelemahan yaitu:

1. GUI masih sangat sederhana.
2. Peta jalur bus kampus UI hanya dapat menampilkan satu bus saja per waktu.
3. Program dari sisi *server* dan komputer halte/fakultas hanya bisa dijalankan sekali, jika berulang maka GUI-nya akan menjadi tidak rapi.

4.2. Tanggapan Pengguna Terhadap Sistem Pelacak Bus Kampus

Untuk mengetahui tanggapan pengguna tentang sistem pelacak bus kampus, dilakukan kuisioner dengan lima buah pertanyaan. Responden berjumlah sepuluh orang mahasiswa Teknik Elektro pengguna bus kampus dan kuisioner dilakukan di Laboratorium Mercator UI. Berikut tabel yang memuat pertanyaan dan hasil pengolahan datanya.

Tabel 4.1. Tanggapan pengguna terhadap sistem pelacak bus kampus UI

No	Parameter	Jumlah responden terhadap nilai					StdDev	Rata-rata	Optimis	Pesimis
		1	2	3	4	5				
1	Sistem pelacak bus kampus UI diperlukan	1	0	1	4	4	1,870829	4	4,41833	3,58167
2	GUI yang dirancang menarik	5	2	2	0	1	1,870829	2	2,41833	1,58167
3	Anda mengerti teknologi yang digunakan	3	1	4	1	1	1,414214	2,6	2,916228	2,283772
4	Posisi bus yang ditampilkan akurat	0	0	3	5	2	2,12132	3,9	4,374342	3,425658
5	Sistem perlu dikembangkan lebih lanjut	0	0	0	3	7	3,082207	4,7	5,389202	4,010798

Dari tabel tersebut, untuk pertanyaan pertama yaitu apakah sistem pelacak bus kampus UI diperlukan terdapat empat responden yang menjawab sangat setuju, empat responden menjawab cukup setuju, satu responden yang menjawab biasa saja dan satu responden menjawab tidak setuju.

Untuk pertanyaan kedua yaitu apakah GUI yang dirancang menarik terdapat satu responden yang menjawab sangat setuju, dua responden menjawab biasa saja, dua responden menjawab kurang setuju, dan lima responden menyatakan tidak setuju.

Mengenai pertanyaan apakah pengguna mengerti tentang teknologi yang digunakan dalam sistem pelacak bus kampus UI ini, terdapat satu orang menjawab sangat setuju, satu orang menjawab cukup setuju, empat orang menjawab biasa saja, satu orang menjawab kurang setuju, dan tiga responden menyatakan tidak setuju.

Pada pertanyaan apakah posisi bus yang ditampilkan akurat terdapat dua responden menjawab sangat setuju, lima responden menjawab setuju, dan tiga responden menjawab biasa saja.

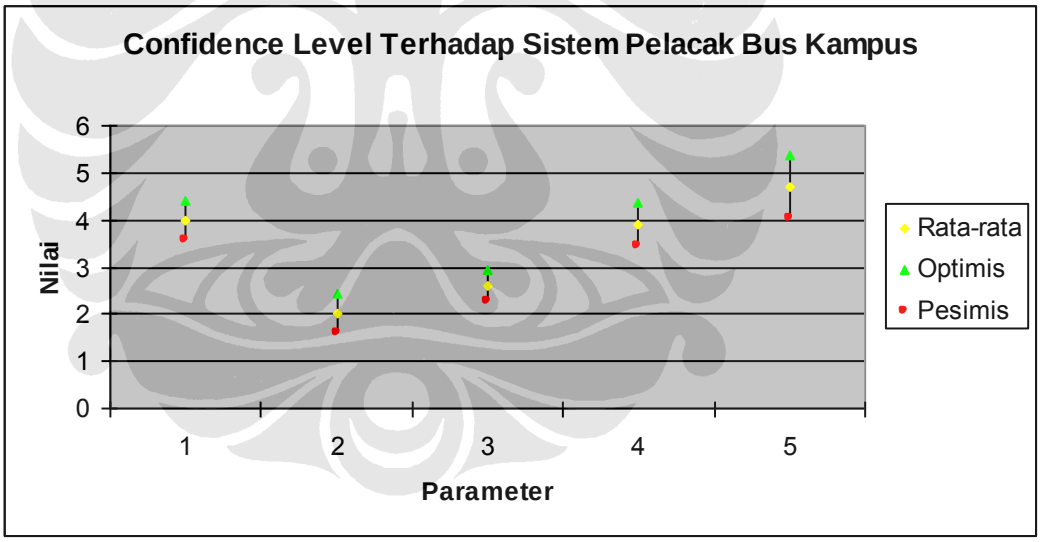
Untuk pertanyaan apakah sistem pelacak bus kampus UI ini perlu dikembangkan lebih lanjut terdapat tujuh orang menjawab sangat setuju dan tiga orang menjawab cukup setuju.

Dari keseluruhan pertanyaan, dapat disimpulkan bahwa pengguna berpendapat bahwa sistem pelacak bus kampus UI diperlukan, GUI yang telah dirancang kurang menarik, pengguna tidak terlalu mengerti tentang teknologi yang digunakan pada perancangan sistem pelacak bus kampus UI ini, posisi bus yang ditampilkan cukup akurat, dan sistem ini perlu dikembangkan lebih lanjut.

Pada tabel terdapat nilai *confidence interval* yang dihitung menggunakan rumus berikut:

$$\text{Confidence Interval} = \text{average} \pm 1.96 \left(\frac{\text{stdev}}{\sqrt{10}} \right)$$

sehingga jika dijadikan dalam bentuk grafik adalah sebagai berikut:



Grafik 4.1. Grafik hasil tanggapan pengguna terhadap sistem pelacak bus kampus UI

BAB 5
KESIMPULAN

1. Sistem pelacak bus kampus memungkinkan penggunanya dapat mengetahui keberadaan terakhir bus di halte.
2. Langkah-langkah yang dilakukan dalam merancang sistem pelacak bus kampus ini adalah memprogram modul DT-51 LCMS 2.0, memasang dan menggabungan modul DT-51 LCMS 2.0 dengan modul *wireless* YS 1020 RF Data Tranceiver serta komputer halte/fakultas, pembuatan program menggunakan bahasa C pada komputer halte/fakultas, dan pembuatan program menggunakan bahasa C pada komputer *server*.
3. Dalam memprogram modul DT-51 LCMS 2.0 ada empat hal yang perlu diperhatikan yaitu inisialisasi serial, memasukkan data yang diinginkan, pengiriman secara serial, dan pewaktuan data terkirim berulang setiap 5 detik agar data tidak selalu terkirim yang dapat mengakibatkan tabrakan dengan data lain.
4. Dalam pembuatan program menggunakan bahasa C pada komputer halte/fakultas ada empat hal yang perlu diperhatikan, yaitu membuka *port* serial, membaca data dari *port* serial, mengolahnya sebelum dikirim ke komputer *server*, dan mengirimkannya melalui jaringan komputer protokol TCP/IP ke *server*.
5. Di *server* data tersebut ditampilan dalam bentuk GUI sederhana yang menampilkan jalur yang dilewati oleh bus kampus.
6. Hasil pengujian memperlihatkan bahwa data berbentuk GUI cukup akurat dengan keadaan aslinya namun hanya dapat menampilkan satu bus saja per waktu dan tampilannya masih sangat sederhana.
7. Dari hasil kuisioner, pengguna berpendapat bahwa sistem pelacak bus kampus berguna, memiliki GUI yang tidak terlalu menarik, pengguna tidak terlalu mengerti teknologi yang digunakan, namun setuju bahwa posisi bus yang ditampilkan akurat. Pengguna juga berpendapat bahwa sistem ini perlu dikembangkan lebih lanjut.

DAFTAR REFERENSI

[1] Akman, V., *Context in artificial intelligence: A fleeting overview*. in Penco, C., Ed., La Svolta Contestuale, McGraw-Hill, Milan, 2002.

[2] Loke, S., *Context-Aware Pervasive Systems: Architectures for a New Breed Applications*. Auerbach Publications, New York, 2007.

[3] McCarthy, J., *Notes on formalizing contexts, in Ruzena Bajcs.* Ed., Proceedings of the 13th International Joint Conference on Artificial Intelligence, pp. 555–560, Morgan Kaufmann, San Mateo, 1993.

[4] *"Context"*. Kamus *Online* Merriam-Webster, http://www.m-w.com/home.htm, terakhir diakses 29 November 2007.

[5] *"Context"*. thesaurus.reference.com, http://thesaurus.reference.com, terakhir diakses 29 November 2007.

[6] Schilit, B., Theimer, *M. Disseminating active map information to mobile hosts*. 8(5):22-32, IEEE Network, 1994.

[7] Brown, P. G., Bovey, J. D., Chen, X.**,** *Context-aware applications: From the laboratory to the marketplace.* 4(5):58-64, IEEE Personal Communications, 1997.

[8] Brézillon, P., Pomerol, J. C. *Contextual knowledge sharing and cooperation in intelligent assistant systems, in Le Travail Humain.* 62(3):223-246, 1999.

[9] Dey, A. K.. *Providing architectural support for building context-aware applications.* PhD thesis, College of Computing, Georgia Institute of Technology, 2000.

[10] *"Serial Communication"*. Wikipedia, http://en.wikipedia.org/wiki/serial_communication, terakhir diakses 1 Oktober 2008.

[11] "RS-232". Wikipedia, http://en.wikipedia.org/wiki/RS-232, terakhir diakses 1 Oktober 2008.

[12] *"Serial Port".* Wikipedia, http://en.wikipedia.org/wiki/serial_port, terakhir diakses 1 Oktober 2008.

[13] *"In-System Programming".* Wikipedia,

http://en.wikipedia.org/wiki/in-system_programming, terakhir diakses 1 Oktober 2008.

[14] "DT-51 LC Micro2". *Manual DT-51 LCMS v. 2.0*. CD-ROM. Innovative Electronics. 2006.

[15] "DT-51 LC Micro2". *AT89S51*. CD-ROM. Innovative Electronics. 2006.

[16] "YS-1020UA RF Data Transceiver *Manual"*. ShenZhen Yishi Electronic Technology Development Co., Ltd, http://www.yishi.net.cn/YS-1020UA, terakhir diakses 5 Oktober 2008.

Lampiran 1. Program mikrokontroler untuk bus kampus merah nomor 5

```
ROM EQU 0000H

 Org ROM ; Reset Vector
 Ajmp Start ;
 Org ROM+3H ; External Interrupt 0 Vector
 Reti ;
 Org ROM+0BH ; Timer 0 Interrupt Vector
 Reti ;
 Org ROM+13H ; External Interrupt 1 Vector
 Reti ;
 Org ROM+1BH ; Timer 1 Interrupt Vector
 Reti ;
 Org ROM+23H ; Serial Interrupt Vector
 Reti ;

Start:
 Acall Init_Serial       ; Initial Serial
 mov b, #11110101b       ; kode bus merah no 5

Loop:
  Acall Serial_Out ; Send serial data to shelter
  Acall wait5sec
  Ajmp Loop

;SUBROUTINE
Init_Serial:
 MOV SCON,#52H ; Mode 1 Ren
 MOV TMOD,#20H ; T1 Mode 2
 MOV TH1,#0FDH ; 9600 Baudrate
 MOV TCON,#040H ; T1 On, T0 Off
 MOV PCON,#00H ;
 RET
Serial_Out:
 Clr TI
 Mov SBUF,b
WaitSend:
 Jnb TI,WaitSend
 ret
Wait5sec:                ; program wait 5 detik
 Mov a, 0
 Wait:
  inc a
  mov R1, #0
  wait2:
   inc R1
   mov R2, #0
   wait3:
    inc R2
    cjne R2, #21, wait3
   cjne R1, #255, wait2
  cjne a, #255, wait
 Ret

END
```

Lampiran 2. Program berbahasa C yang terinstal di komputer halte/fakultas

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>      /* File control definitions */
#include <errno.h>      /* Error number definitions */
#include <termios.h>    /* POSIX terminal control definitions */
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <signal.h>
#include <sys/wait.h>
#include <sys/ipc.h>
#include <sys/shm.h>

#define ONE_BYTE 1
#define MYPORT 5678

int fd, sock_fd;
int OK;
int variable_ToSend;
struct sigaction sa;
struct sockaddr_in my_addr;
struct sockaddr_in their_addr;
struct hostent *he;
int numbytes_send;

int open_port(void);
void init_port(int);
int readport(int, unsigned char *, int);
void writeport(int, unsigned char, int);
void sigchld_handler(int s);

unsigned char bus_char;
unsigned int bus_value;
unsigned int warna;
unsigned int nomor;
unsigned int fak_value = 7;
unsigned int bus_fak_val;
unsigned char bus_fak_char;
unsigned int data_bef=0;
unsigned int data_rght;
int i;

int main (int argc, char *argv[]) {

      fd = open_port(); // Buka Port Serial
      init_port(fd);          // Inisialisasi Port
      while (1) {
            // terima
            while(!readport(fd, &bus_char, ONE_BYTE));
```

```
            bus_value = bus_char;
            nomor = bus_value & 0x7;
            warna = (bus_value & 0x8)>>3;
            // cek
            data_rght = bus_value & 0xF0;
            if (data_rght == 0xF0 && bus_value != data_bef)
            {
                  // kirim
                  bus_fak_val = (fak_value<<4)+(bus_value & 0x0F);
                  bus_fak_char = bus_fak_val;
                  if (warna == 0)
                  printf("Nomor bus = %d dan warna bus merah\n",
nomor);
                  if (warna == 1)
                  printf("Nomor bus = %d dan warna bus biru\n",
nomor);
                  data_bef = bus_value;
                  printf("Data bus dikirim ke server....\n");
                  if(argc != 2) {
                  fprintf(stderr, "usage:client hostname\n");
                  exit(1);
                  }
                  if((he=gethostbyname(argv[1])) == NULL) {
                  perror("gethostbyname");
                  exit(1);
                  };


                  if((sock_fd = socket(AF_INET, SOCK_STREAM, 0))
== -1) {
                        perror("socket");
                        exit(1);
                  }

                  if ((he=gethostbyname(argv[1])) == NULL) { //
get the host info
                        perror("gethostbyname");
                        exit(1);
                  }

                  OK = 1;

                  if(setsockopt(sock_fd, SOL_SOCKET, SO_REUSEADDR,
&OK, sizeof(int)) == -1) {
                        perror("setsockopt ERROR");
                        exit(1);
                  }

                  sa.sa_handler = sigchld_handler;    // Reap all
dead processes
                  sigemptyset(&sa.sa_mask);
                  sa.sa_flags = SA_RESTART;
                  if(sigaction(SIGCHLD, &sa, NULL) == -1) {
                        perror("sigaction ERROR");
                        exit(1);
                  }
```

```
                  my_addr.sin_family = AF_INET;
                  my_addr.sin_port = htons(MYPORT);
                  my_addr.sin_addr.s_addr = htons(INADDR_ANY);
                  // automatically fill with my IP
                  memset(my_addr.sin_zero, '\0', sizeof
my_addr.sin_zero);      // zero the rest of the struct

                  their_addr.sin_family = AF_INET;
                  their_addr.sin_port = htons(MYPORT);
                  their_addr.sin_addr = * ((struct in_addr *)he-
>h_addr);
                  memset(&(their_addr.sin_zero), '\0', 8);

                  if(connect(sock_fd, (struct sockaddr
*)&their_addr, \
                              sizeof(struct sockaddr)) == -1) {
                        perror("connect");
                        exit(1);
                  }
                  while(1) {
                        variable_ToSend = bus_fak_val;
                        printf("variable is sent = %d\n",
variable_ToSend);
                        if((numbytes_send=send(sock_fd,
&variable_ToSend, sizeof(variable_ToSend), 0)) == -1) {
                              perror("send");
                              exit(1);
                        }
                        i++;
                  }
                  printf("variable is sent = %d",
variable_ToSend);
                  close(sock_fd);
                  return 0;
                  }
}
return 0;
}
int open_port(void) {
      int fd;

      fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY | O_NDELAY);
      if(fd == -1) {
            perror("-- Unable to open /dev/ttyS0 --");
            exit(1);
      }
      else {
            fcntl(fd, F_SETFL, 0);
      }

      return(fd);
}
void init_port(int fd) {
      struct termios configs;

      // Ambil konfigurasi sekarang dari ttyS0 (COM1)
      tcgetattr(fd, &configs);
```

```
	//Set baud rate ke 9600
	cfsetispeed(&configs, B9600);
	cfsetospeed(&configs, B9600);

	// Setting Control Mode
	configs.c_cflag &= ~PARENB;		// Parity disable
	configs.c_cflag &= ~CSTOPB;		// 1 stop bit
	configs.c_cflag &= ~CSIZE;
	configs.c_cflag |= CS8;			// 8-bit data
	configs.c_cflag &= ~CRTSCTS;  // Disable hardware flow
control

	// Setting Local Mode
	configs.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG);   //RAW
input

	// Setting Input mode
	configs.c_iflag &= ~INPCK;			// Disabling
Parity Check
	configs.c_iflag &= ~ISTRIP;			// Disable strip
off 8th bit when received
	configs.c_iflag &= ~(IXON | IXOFF | IXANY);     // Disable
software flow control

	// Setting Output Mode
	configs.c_oflag &= ~OPOST;		// RAW output

	// Setting Control chars
	configs.c_cc[VTIME] = 0;		// Timeout in ds, 0 (no
timeout)
	configs.c_cc[VMIN] = 1;

	// Write konfigurasi yang baru ke ttyS0
	tcsetattr(fd, TCSANOW, &configs);
}
int readport(int fd, unsigned char* receive_data, int n) {
	int size_read;

	fcntl(fd, F_SETFL, FNDELAY);
	size_read = read(fd, receive_data, n);
	//printf("size: %d; receive data: %d\n", size_read,
*receive_data);
	fcntl(fd, F_SETFL, 0);
	if(size_read < 0)
		return 0;
	return 1;
}
void writeport(int fd, unsigned char transmit_data, int n) {
	int size_write;
	int write_true = 0;

	while(write_true == 0) {
		size_write = write(fd, &transmit_data, n);
		if(size_write >= 0)
			write_true = 1;
	}
```

```
}
void sigchld_handler(int s) {
      while(wait(NULL) > 0);
}
```

Lampiran 3. Program berbahasa C yang terinstal di komputer *server*

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>      /* File control definitions */
#include <errno.h>      /* Error number definitions */
#include <termios.h>    /* POSIX terminal control definitions */
#include <time.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <netdb.h>
#include <signal.h>
#include <sys/wait.h>

#define BACKLOG 10
#define MYPORT 5678

int sock_fd;
void sigchld_handler(int);
char* halteMerah[15] = {"tidak dilampirkan karena terlalu
panjang"};

char* halteBiru[15] =  {"tidak dilampirkan karena terlalu
panjang"};

int main(void) {
      int sin_size, OK;
      int variable_ToGet;
      int noHalte, labelBus;

      struct sigaction sa;

      struct sockaddr_in my_addr;
      struct sockaddr_in their_addr;
      int numbytes_recv;

//    system("clear");

      if((sock_fd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
            perror("socket");
            exit(1);
      }
      OK = 1;
      if(setsockopt(sock_fd, SOL_SOCKET, SO_REUSEADDR, &OK,
sizeof(int)) == -1) {
            perror("setsockopt ERROR");
            exit(1);
      }
      sa.sa_handler = sigchld_handler;
      sigemptyset(&sa.sa_mask);
      sa.sa_flags = SA_RESTART;
      if(sigaction(SIGCHLD, &sa, NULL) == -1) {
            perror("sigaction ERROR");
```

```
            exit(1);
      }
      my_addr.sin_family = AF_INET;
      my_addr.sin_port = htons(MYPORT);
      my_addr.sin_addr.s_addr = htons(INADDR_ANY);

      memset(my_addr.sin_zero, '\0', sizeof my_addr.sin_zero);
      if(bind(sock_fd, (struct sockaddr *)&my_addr, sizeof(struct
sockaddr)) == -1) {
            perror("bind");
            exit(1);
      }
      if(listen(sock_fd, BACKLOG) == -1) {
            perror("listen");
            exit(1);
      }
      sin_size = sizeof(struct sockaddr_in);
      //while(1){
            if((sock_fd = accept(sock_fd, (struct sockaddr
*)&their_addr, &sin_size)) == 1) {
            perror("accept");
            }
            printf("server: got connection from %s\n",
inet_ntoa(their_addr.sin_addr));
            if((numbytes_recv = recv(sock_fd, &variable_ToGet,
sizeof(variable_ToGet),0))== -1) {
            perror("recv");
                  }
                  noHalte = variable_ToGet >> 4;
                  labelBus = (variable_ToGet >> 3) % 2;
            //    noBus = variable_ToGet % 8;
                  printf("%d no halte \t %d label\n",noHalte,
labelBus);
            if(labelBus==0)
                  printf("%s\n", halteMerah[noHalte]);
            if(labelBus==1)
                  printf("%s\n", halteBiru[noHalte]);
            //}
                  return 0;
}
void sigchld_handler(int s) {
      while(wait(NULL) > 0);
}
```