PEMROGRAMAN ANTARMUKA MODEM GSM DENGAN PENGENDALI MIKRO AVR MENGGUNAKAN BAHASA C

Daniel Kartawiguna

Information Systems Department, School of Information Systems, Binus University
Jl. K.H. Syahdan No. 9, Palmerah, Jakarta Barat 11480
daniel.kartawiguna@gmail.com; daniel@binus.ac.id

ABSTRACT

Cell phones are now a cheap means of wide-ranged wireless communication. One feature widely used is the short message service (SMS). The cellular communication system is more appropriate for long-distanced applications than high speed data transfer. This study aims to develop a GSM modem interface with AVR microcontroller using C programming language. The tools controlled by the microcontroller system can gain more benefit if it is linked with GSM mobile communication system. The results of this study can be applied whether in a remote monitoring system, remote control system, or communication between the microcontroller via SMS. Many additional benefits can be obtained for the tools controlled by the microcontroller when connected to a GSM system.

Keywords: *interface, GSM modem, AVR microcontroller, C language.*

ABSTRAK

Telepon seluler sudah menjadi sarana komunikasi nirkabel yang murah dengan jangkauan yang luas. Salah satu fitur yang banyak dimanfaatkan adalah layanan pesan singkat atau SMS (short message service). Sistem komunikasi seluler ini sangat tepat untuk aplikasi yang lebih mengutamakan sambungan jarak jauh dari pada kecepatan transfer data yang tinggi. Penelitian ini bertujuan untuk mengembangkan antarmuka modem GSM dengan pengendali mikro (microcontroller) AVR yang menggunakan bahasa pemrograman C. Peralatan yang dikendalikan oleh sistem microcontroller dapat memperoleh manfaat lebih banyak bila dihubungkan dengan sistem komunikasi bergerak GSM. Hasil penelitian ini dapat diaplikasikan dalam sistem pemantauan jarak jauh, sistem pengendalian jarak jauh, atau komunikasi antar microcontroller melalui SMS. Banyak manfaat tambahan yang diperoleh bagi peralatan yang dikendalikan oleh microcontroller apabila terhubung ke sistem GSM.

Kata kunci*: antarmuka, modem GSM, microcontroller AVR, bahasa C.*

PENDAHULUAN

Jaringan GSM yang digunakan oleh telepon seluler saat ini sudah menjadi sarana komunikasi tanpa kabel yang murah dengan jangkauan yang luas. Sistem komunikasi seluler dengan teknologi GSM ini sangat tepat untuk aplikasi yang lebih mengutamakan sambungan jarak jauh dari pada kecepatan transfer data yang tinggi. Peralatan yang dikendalikan oleh sistem *microcontroller* (pengendali mikro) dapat memperoleh manfaat yang lebih banyak bila dihubungkan dengan sistem komunikasi bergerak GSM. Peralatan seperti mesin pendingin industri dan mesin pembeku, sistem HVAC (*Heating, Ventilating, and Air Conditioning*), mesin vending, layanan kendaraan, sistem pemantauan dan pengendalian jarak jauh, dan lain-lain dapat memperoleh manfaat tambahan apabila terhubung ke sistem GSM. Bahkan ada kemungkinan antar sistem yang dikendalikan oleh *microcontroller* dapat melakukan komunikasi langsung dari mesin ke mesin malalui fasilitas SMS.

Penelitian ini akan mengembangkan antarmuka pengendali mikro AVR dengan modem GSM telepon seluler menggunakan bahasa pemrograman C. Terealisasinya antarmuka ini maka akan memungkinkan dilakukannya komunikasi dua arah antara pengendali mikro AVR dengan telepon seluler atau modem GSM melalui saluran komunikasi serial RS-232C. Sebagian besar telepon seluler dan modem GSM dapat digunakan, kecuali ponsel Nokia® yang menggunakan F-bus atau M-bus. Dengan antarmuka ini, sistem yang dikendalikan oleh pengendali mikro AVR dapat mengirimkan pesan SMS kepada telepon seluler dan sebaliknya telepon seluler dapat mengirimkan pesan atau perintah pada sistem pengendali mikro AVR.

Protokol yang digunakan untuk pengaturan dan pengendalian modem GSM didasarkan pada kumpulan perintah-AT Hayes (*Hayes AT-Command set*) yang biasa digunakan untuk pengaturan modem telepon biasa. *Hayes AT-Commands* adalah protokol tekstual yang hampir dikenali oleh seluruh telepon seluler atau modem GSM. Perintah-AT khusus modem GSM dilengkapi dengan layanan tambahan yang dikenal pada sistem komunikasi GSM seperti: komunikasi pesan teks (SMS), pemanggilan nomor telepon yang diberikan, menghapus lokasi memori, pengaturan telepon, dan lain-lain. Karena tujuan utama dari penelitian ini adalah untuk mengembangkan antarmuka untuk mengirim dan menerima pesan teks singkat, maka hanya sebagian dari kumpulan perintah-AT yang dijelaskan pada landasan teori ini yang akan diimplementasikan.

Institut Standar Telekomunikasi Eropa (ETSI) GSM 07.05 mendefinisikan antarmuka *AT-Command* untuk modem yang GSM kompatibel. Dari dokumen ini beberapa perintah yang terpilih akan dipilih, dan dijelaskan secara singkat dalam bagian ini. Sebagian perintah ini akan memungkinkan modem untuk mengirim dan menerima pesan SMS. Untuk penjelasan yang lebih terinci, dapat langsung mengacu kepada standar GSM 07.05.

Kumpulan Perintah-AT (AT-*Command*)

Tabel 1 memberikan gambaran tentang perintah AT yang diimplementasikan dalam aplikasi antarmuka ini. Penggunaan perintah ini dijelaskan dalam bagian berikutnya.

Tabel 1
Ikhtisar Kumpulan Perintah AT

Perintah	Penjelasan
AT	Memeriksa apakah komunikasi antarmuka serial dengan modem GSM bekerja.
ATE0	Menonaktifkan *echo*, mengurangi trafik pada saluran serial.
AT+CNMI	Menampilkan pesan SMS yang baru masuk.
AT+CPMS	Pemilihan memori SMS.
AT+CMGF	SMS format string, bagaimana mereka terkompresi.
AT+CMGR	Baca pesan baru dari lokasi memori yang diberikan.

AT+CMGS	Kirim pesan ke penerima yang telah ditentukan.
AT+CMGD	Menghapus pesan.

Format berikut ini akan digunakan dalam Tabel 2 – 9:
Karakter *string* dalam tanda kutip adalah teks sesungguhnya yang harus dikirim ke modem.
Perintah tambahan dan parameter tanggapan ditulis dalam kurung siku.
Untuk mengeksekusi perintah, pada akhir perintah perlu ditambahkan karakter CR + LF (*Carriage return* + *line feed* = \ r \ n).

Status (AT)

Perintah "AT" (Tabel 2) adalah permintaan status. Digunakan untuk menguji apakah modem yang kompatibel tersambung dan bahwa antarmuka serial bekerja dengan benar.

Tabel 2
Perintah AT dan Tanggapan yang Mungkin

Perintah	Tanggapan	Keterangan
"AT"	"OK"	Terhubung dan berfungsi
	"ERROR"	Saluran serial baik, modem error

Echo off (ATE0)

Perintah "ATE0" digunakan untuk konfigurasi komunikasi. Secara *default*, modem GSM diatur untuk mengirimkan kembali perintah yang diberikan sebagai konfirmasi bahwa perintah telah diterima (*echo*). Contoh pengeksekusian perintah AT adalah seperti contoh berikut ini:

AT\r\n // Perintah yang dikirim ke modem
AT\r\nOK\r\n // Respon dari modem dengan echo diaktifkan

Setelah mengirim perintah "AT", modem akan menjawab dengan "AT\r\nOK\r\n". Apabila diberikan perintah "ATE0" (Tabel 3), modem dalam keadaan *echo off*, sehingga apabila diberikan perintah "AT" maka modem akan menjawab "\r\nOK\r\n". Perintah *echo off* akan mengurangi lalu lintas data pada jalur komunikasi serial. Untuk mengaktifkan kondisi *echo on* kembali maka digunakan perintah "ATE1".

Tabel 3
ATE0 Perintah dan Tanggapan Mungkin

Perintah	Tanggapan (*echo off*)	Keterangan
"ATE0"	"OK"	*Echho off*
	"ERROR"	Tidak dapat mengaktifkan echo off

Indikasi Pesan Baru (AT + CNMI)

Perintah "AT + CNMI" (Tabel 4) berfungsi untuk mengkonfigurasi cara modem memberikan sinyal tanda diterimanya pesan baru ke perangkat terminal yang terhubung, dan bagaimana pesan tersebut disimpan dalam modem. Fitur ini berguna ketika akan membaca pesan yang baru datang. Dengan perintah "AT + CNMI" modem akan memberikan pesan bahwa sebuah pesan baru telah tiba sehingga tidak perlu melakukan pemungutan suara (*pooling*) pada modem secara berkala untuk memantau kedatangan pesan baru. Pengendali mikro AVR akan menangkap indikasi tersebut, dan

menetapkan bendera (*flag*). Hal ini memastikan bahwa modem menggunakan sumber daya prosesor hanya bila diperlukan.

Tabel 4
Perintah AT + CNMI dan Tanggapan Mungkin

Perintah	Tanggapan	Keterangan
"AT + CNMI = [mode][1], [mt][2], [bm][3], [ds][4], [bfr][5]"	"OK"	Mode set
	"ERROR"	Error, tidak bisa mengatur mode tersebut.

Catatan:
1. [mode] tipe integer: bagaimana pesan akan di-*buffer*.
2. [mt] tipe integer: indikasi SMS baru, diset ke 1.
3. [bm] tipe integer: Tidak dipakai.
4. [ds] tipe integer: Tidak dipakai.
5. [bfr] tipe integer: Tidak dipakai.

Nilai-nilai "[mode]", "[mf]", "[bm]", "[ds]" dan "[bfr]" yang didapat akan berbeda dari modem ke modem. Ini harus diuji secara *off line* menggunakan modem yang terhubung ke komputer atau pengendali mikro. Sebuah contoh diberikan di bawah ini:
AT+CNMI=?\r\n //permintaan nilai kemungkinan
+CNMI: (0,1),(0,1),(0,2),(0,2),(1) //Kemungkinan nilai parameter
OK //Perintah dilaksanakan OK

Penyimpanan pesan yang dipilih (AT + CPMS)

Perintah "AT + CPMS" (Tabel 5) berfungsi untuk menentukan lokasi memori tujuan yang digunakan untuk menyimpan pesan SMS yang dikirim, dibaca, dihapus, dan diterima. Kebanyakan modem memiliki beberapa jenis lokasi momori penyimpanan pesan:
"SM": memori kartu SIM.
"ME": penyimpanan *Mobile Equipment*. Lokasi penyimpanan dalam modem hanya untuk pesan teks saja.
"MT": kumpulan semua lokasi penyimpanan yang terhubung ke modem: SM, ME atau yang lain. Telepon akan memilih salah satu yang sesuai jika pilihan ini diaktifkan.

Tabel 5
Perintah AT + CPM dan Tanggapan Mungkin

Perintah	Tanggapan	Keterangan
"AT + CPMS = [M1][1], [M2][2], [M3][3]"	"+ CPM: [used1], [total1], [used2], [total2], [used3], [total3] \ R \ ROK \ r \ n "	Memori terkonfigurasi OK
	"+ CMS ERROR"	Kesalahan.

Catatan:
[M1] tipe string: memori dari pesan yang dibaca dan dihapus.
[M2] tipe string: memori untuk pesan mana yang ditulis dan dikirim.
[M3] tipe string: memori yang menerima pesan akan disimpan, jika meneruskan ke PC tidak diatur.
[used] tipe integer: jumlah pesan yang saat ini dalam x.
[total] tipe integer: jumlah lokasi pesan dalam x.

Format Pesan (AT + CMGF)

Perintah "AT + CMGF" (Tabel 6) ini digunakan untuk mengatur format masukan dan keluaran dari pesan SMS. Ada dua mode yang tersedia: (1) mode PDU – membaca dan mengirim SMS dilakukan dalam format disandikan khusus; (2) mode teks – membaca dan mengirim SMS dilakukan dalam teks biasa. Format terkompresi ini akan menghemat muatan pesan dan merupakan kondisi standar (*default*) pada sebagian besar tipe modem. Mode PDU (dijelaskan pada sub bab berikutnya) diimplementasikan dalam kode sumber dalam tulisan ini. Dimungkinkan juga untuk menggunakan mode teks jika modem terhubung mendukung mode ini. Penggunaan mode teks akan mengurangi panjang kode program.

Dalam mode teks, *header field* seperti alamat pengirim, panjang pesan, periode validasi, dan lain-lain dapat dibaca dalam teks biasa bersama-sama dengan pesan yang dikirim. Untuk keterangan lebih lanjut tentang membaca pesan dalam mode teks dapat mengacu pada dokumen GSM 07.05. Mode teks tidak akan dibahas secara rinci di sini.

Tabel 6
Perintah AT + CMGF dan Tanggapan Mungkin

Perintah	Tanggapan (*echo off*)	Keterangan
"AT + CMGF = [mode][1]"	"OK"	Mode terpilih
	"ERROR"	Terjadi kesalahan

Catatan:
[mode] tipe integer: 0 adalah mode PDU, 1 adalah mode teks.

Membaca Pesan (AT + CMGR)

Perintah "AT + CMGR" (Tabel 7) ini digunakan untuk membaca pesan dari lokasi memori yang diberikan. Pelaksanaan perintah "AT + CMGR" akan mengembalikan pesan di [index] dari lokasi memori yang dipilih [M1] (lihat bagian 2.1.4 untuk pengaturan memori). Status dari pesan dan seluruh pesan yang dikompresi (dalam format PDU) dikembalikan. Untuk mendapatkan informasi yang berguna dari pesan terkompresi itu maka pesan tersebut harus di-dekompresi.

Tabel 7
Perintah AT + CMGR dan Tanggapan Mungkin

Perintah	Tanggapan	Keterangan
"AT + CMGR = [index][1]"	"+CMGR:[stat][2],[alpha] [3],	Pesan terbaca
	[length][4] \r\n [pdu][5]"	OK
	"+CMS ERROR"	Kesalahan, tidak ada indeks tersebut.

Catatan:
[index] tipe integer: Membaca pesan dari lokasi [index].
[stat]: tipe integer: Status pesan dalam memori: READ (terbaca), UNREAD (belum dibaca), SENT (terkirim), dan UNSENT (belum terkirim).
[alpha] tipe integer: Isian khusus dari pabrik. Tidak digunakan.
[length] tipe integer: Panjang dari pesan terkompresi.
[pdu] tipe string ketik: Pesan terkompresi.

Kirim Pesan (AT + CMGS)

Perintah ini memungkinkan pengguna untuk mengirim SMS. Setelah isian sudah ditentukan oleh pengguna, pesan dapat dikompresi dan dikirim menggunakan perintah "AT + CMGS" (Tabel 8).

Tabel 8
Perintah AT + CMGS dan Tanggapan yang Mungkin

Perintah	Tanggapan	Keterangan
"AT+CMGS = [length][1] **CR**[2]	OK	Pesan terkirim
[pdu][3] **ctrl-Z**[4]"	"+CMS ERROR"	Perintah salah

Catatan:
[length] tipe integer: Panjang pesan.
CR = *Carriage return*, penekanan tombol ”Enter” atau karakter CR pada kode ASCII
[pdu] tipe string: Pesan yang terkompresi
Ctrl-Z: Perintah terminasi. Karakter ASCII 26 (desimal).

Hapus Pesan (AT + CMGD)

Perintah ini digunakan untuk menghapus pesan diterima yang tersimpan pada [M1] (Tabel 5).

Tabel 9
Perintah AT + CMGD dan Tanggapan yang Mungkin

Perintah	Tanggapan	Keterangan
"AT + CMGD = [index] [1]"	OK	Pesan terhapus
	"+CMS ERROR"	Perintah salah

Catatan:
[index] tipe integer: Indeks dari pesan yang akan dihapus.

Hingga di sini telah dijelaskan kumpulan perintah-AT yang diimplementasikan dalam penelitian ini. Perintah-perintah AT lainnya dapat dipelajari dalam standar ETSI GSM 07.05, dan dokumen ini bersama dengan lembar data teknis pabrik disarankan sebagai referensi dalam proses mengembangkan aplikasi antarmuka modem GSM yang kompatibel.

Kode Kesalahan

Banyak dari bagian perintah yang diimplementasikan dapat terhenti dengan sebuah pesan kesalahan yang berhubungan dengan modem atau jaringan. Kesalahan-kesalahan ini dapat seperti: (1) kegagalan memori; (2) nomor penerima yang tidak valid; (3) tidak ada tanggapan dari jaringan (*Network timeout*); (4) SIM sibuk atau salah; (5) operasi tidak diijinkan; (6) tidak ada layanan jaringan.

Pesan-pesan kesalahan dapat bermanfaat, dan dapat diterapkan sebagai bagian dari aplikasi. Hal ini dimungkinkan untuk memperluas penanganan kode kesalahan, tapi ini di luar lingkup dari penelitian ini.

Penjelasan Format PDU

Ada dua cara untuk mengirim dan menerima pesan SMS: dengan modes teks dan mode PDU (*protocol description unit*). Secara *default* kebanyakan telepon dan modem di-*set up* untuk mengirim pesan SMS dengan menggunakan format kompresi khusus (mode PDU).

Beberapa modem mendukung mode teks, di mana setiap informasi dan pesan itu sendiri dapat dibaca sebagai teks biasa. Namun perlu dicatat bahwa tidak semua telepon dan modem GSM mendukung mode teks.

Mode PDU menggunakan tiga jenis data khusus: (1) oktet – kelompok 8 bit dalam pengkodean heksa-desimal (0x00→0xFF). Contoh: E8; (2) semi-oktet – kelompok 8 bit dalam pengkodean desimal (0→153). Contoh: 11; (3) septet – kelompok 7 bit dalam pengkodean integer (0→127). Contoh: 126.

Alfabet GSM *default* menggunakan 7 bit untuk mewakili karakter-karakter. Pesan "hello" terdiri dari lima karakter yang disebut septet, ketika diwakili dengan masing-masing tujuh bit. String septet harus dikodekan menjadi aliran oktet untuk transfer SMS (Tabel 10).

Tabel 10
Mengompresi Septet String ke Stream Oktet

Nilai	**h**	**e**	**l**	**l**	**o**
Desimal	104	101	108	108	111
Heksadesimal	0x68	0x 65	0x 6C	0x 6C	0x 6F
Septet	1101000	1100101	1101100	1101100	1101111
8-bit	11101000	00110010	10011011	11111101	00000110
Oktet	E8	32	9B	FD	06

Catatan:
Septet pertama (h) diubah menjadi suatu octet dengan menambahkan bit paling kanan dari septet kedua (tercetak tebal). Disisipkan pada sebelah kiri hingga menjadi: 1 + 1101000 = 11101000 ("E8"). Kemudian karakter kedua (septet) menerima dua bit (tercetak tebal) dari septet ketiga, maka karakter kedua (e) menjadi sebuah oktet: 00 + 110010 = 00110010 ("32"). Lima bit pertama karakter terakhir ini (o) diisi dengan angka nol (tercetak tebal).

Pesan yang disandikan dengan cara ini kemudian dapat digunakan sebagai muatan dalam perintah "AT + CMGS" yang dijelaskan pada bagian berikutnya.

Ketika menerima pesan baru, perintah "AT + CMGR" dapat digunakan untuk membaca dari lokasi memori di mana pesan itu berada. Aliran oktet akan dikembalikan dari modem sebagai tanggapan dari perintah ini.

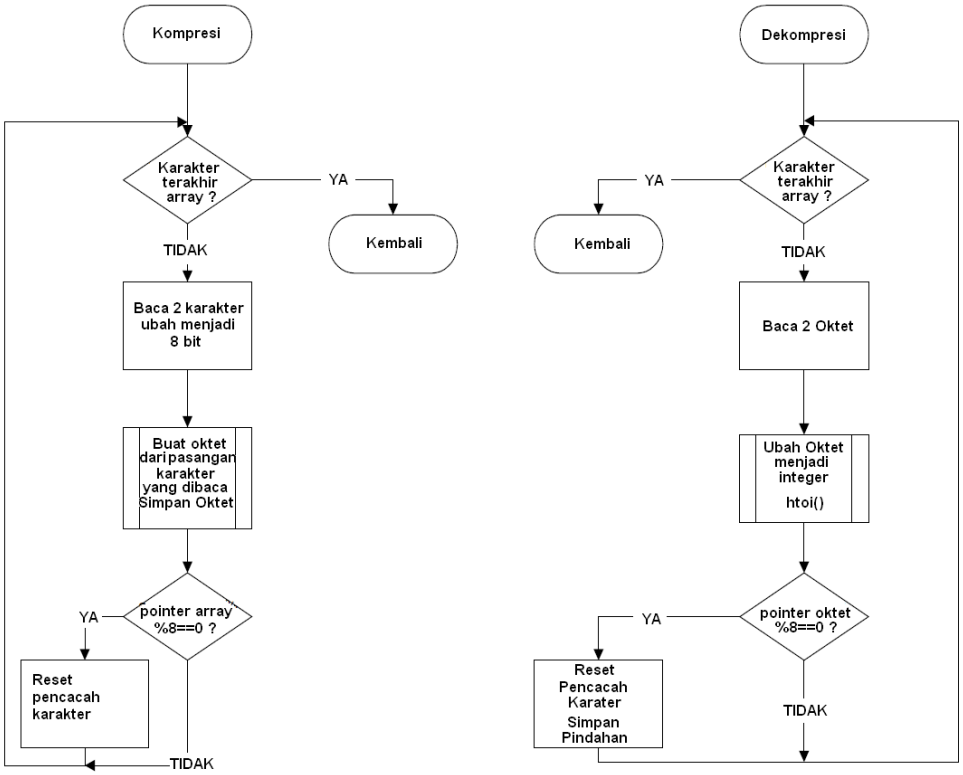
Untuk mengekstrak informasi yang berguna dari aliran data ini maka diperlukan metode dekompresi. Tabel 11 menunjukkan contoh decoding representasi oktet dari pesan teks "hello" kembali ke septet.

Tabel 11
Dekompresi Aliran Oktet ke Septet

Oktet	**E8**	**32**	**9B**	**FD**	**06**
8-bit	11101000	00110010	10011011	11111101	00000110
Septet	1101000	1100101	1101100	1101100	1101111
Desimal	104	101	108	108	111
Nilai	h	e	l	l	o

Catatan:
Untuk menjadi sebuah septet, oktet pertama akan memberikan bit 1 (tercetak tebal) yang paling depan ke oktet kedua. Bit 1 tersebut ini akan ditambahkan di bagian belakang oktet kedua, sedangkan dua buah bit nol paling depan dibuang. Oktet terakhir (06) akan membuang semua bit 0 yang mendahuluinya dan menyalin (1111) yang diterima dari oktet ke empat dibagian belakangnya.

Gambar 1 menunjukkan diagram alir untuk kedua rutin seperti yang diterapkan dalam kode program. Algoritma didasarkan pada metode heuristik, karena algoritma tidak ditentukan dalam standar GSM 03.38 atau GSM 03.40.



Gambar 1. Diagram alir (*flowchart*) proses kompresi dan dekompresi.

Sebuah string SMS terdiri dari tiga bagian utama: panjang *header*, *header*, dan string PDU. Ketika membaca pesan dari modem menggunakan perintah "AT + CMGR", sebuah string SMS akan diterima:

```
AT+CMGR=42          //Baca SMS dari lokasi penyimpanan pesan 42

+CMGR: 0,,42  //Menerima dari telepon
                    //string SMS sebagai berikut:

07914467429499400 40ED0C5BAFC2D0ED3CB00005040623194914019E8329BFD06B540A06B10
EA2A56A54F61905A740D9F4D

OK                  //Pengakuan dari telepon.
                    //AT+CMGR mengembalikan pesan OK.
```

Bagi kebanyakan orang potongan kode di atas tidak mengandung informasi yang dapat dipahami sama sekali. Tabel 12 menunjukkan cara mengekstrak rincian dari string SMS yang dikembalikan. Untuk presentasi penjelasan yang menyeluruh dari semua field, skema kode, huruf, dan lain-lain, dapat mengacu pada standar GSM 03.40.

Tabel 12

Rincian dalam String SMS

	Oktet	Deskripsi
Header	07	Jumlah oktet dalam header, 0x07 = 7
	91	Format penomoran, 91 adalah internasional
	446742949940	Nomor pusat layanan dalan semi-oktet dan terbalik. Nomor sebenarnya adalah: +447624499904
String PDU	04	Oktet pertama SMS-Deliver. Pesan yang dikirim dari pusat layanan ke modem GSM.
	0E	Panjang alamat 0x0E = 14
	D0	Jenis Alamat
	C5BAFC2D0ED3CB	Pengirim: "Eurobate", string oktet
	00	Pengenal Protokol
	00	Skema pengkodean
	50406231949140	Tanda waktu, semi-oktet: 26.04.05 13:49:19 GMT +1,00
	19	Panjang data pengguna. 0x19 = 25 septets
	E8329BFD06B540A06B10EA2A56A54F61905A740D9F4D	Teks pesan yang ditentukan pengguna: "halo-WAP.EUROBATE.COM".

Pesan SMS, sebagaimana ditentukan oleh organisasi ETSI, bisa sampai 160 septet panjangnya. Muatan maksimal pengguna kemudian dibatasi sampai 140 oktet, bersama dengan *field* tambahan dalam protokol PDU. *Field* tambahan ini sangat penting karena berisi informasi tentang alamat-penerima, panjang alamat, periode validitas, jenis alamat, skema pengkodean data, pengenal protokol, dan lain-lain.

Sebuah pesan yang dikirim dari modem ke pusat layanan yang disebut sebuah pesan SMS-SUBMIT. Tabel 13 menunjukkan bagaimana membangun pesan tersebut. Untuk menghindari masalah dengan meta data produsen tertentu-, modem diatur untuk melakukan ini dengan menggunakan pilihan "00". Protokol stack sisanya didefinisikan menurut GSM 03.40.

Tabel 13
Field Pesan SMS-SUBMIT

	Oktet	Deskripsi
Header	00	Jumlah oktet di meta data, 0 berarti bahwa modem harus menggunakan meta tersimpan.
String PDU	11	Pertama oktet dalam SMS-SUBMIT
	00	Pesan referensi, 00 berarti bahwa modem set nomor referensi.
	0A	Panjang alamat: 0x0A = 10
	91	Jenis nomor, tipe internasional
	7421436587	Nomor penerima: +4712345678
	00	Pengenal protokol
	00	Skema pengkodean data
	AA	Waktu kedaluarsa pesan: 4 hari
	05	Jumlah septet: 0x05 = 5 septet.
	E8329BFD06	Teks yang ditentukan pengguna: hello

Sebelum string dikirimkan, kita perlu untuk menghitung panjangnya. Menghitung jumlah oktet, tidak termasuk awalan informasi-meta, memberikan panjang keseluruhan 17 oktet bagi string SMS pada Tabel 13.

Untuk menggunakan aplikasi terminal, perintah berikut ini bisa dikirim ke modem.

```
AT+CMGS=18          // Mengirim pesan yang berisi 18 oktet,
                    // tidak termasuk dua nol awal yang mendahului.
```

Setelah perintah di atas diberikan, modem sekarang akan menunda sesaat dan bersiap-siap untuk menerima oktet string SMS dengan panjang 18. Sebuah "\r\n>" akan ditampilkan pada layar ketika modem sudah siap untuk menerima muatan pesan yang akan dikirimkan. Lihat kode di bawah ini.

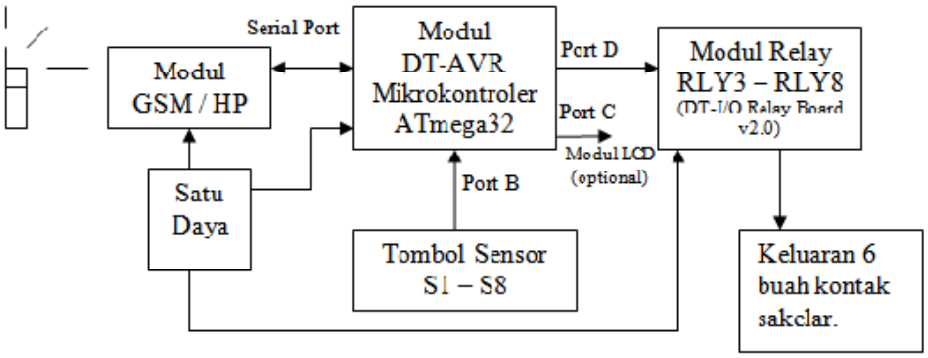
```
> //Prompt ditampilkan oleh ponsel ketika siap untuk mengirim SMS
0011000A9174214365870000AA05E8329BFD06<ctrl-z> // SMS string untuk dikirimkan
```

Modem setelah itu harus menampilkan pesan "OK", yang berarti bahwa pesan tersebut telah dikirim. Dengan menggunakan telepon seluler, proses ini bisa diverifikasi sebagai entri baru di folder terkirim. Kebanyakan pesan kesalahan untuk perintah "AT+CMGS" berasal dari akibat menggunakan panjang pesan yang salah, jadi ini harus diperiksa dua kali. Ingatlah jangan menghitung setiap nol di awal *header*.

METODE

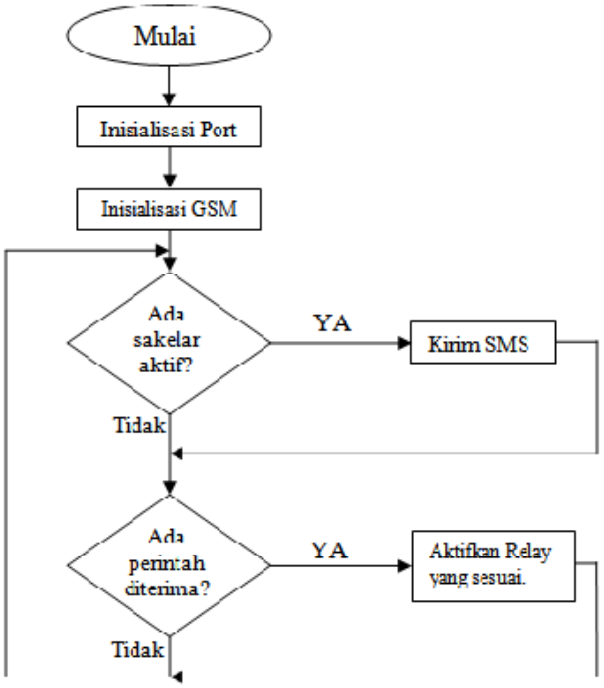
Pengembangkan antarmuka dalam bahasa C untuk penelitian ini akan menggunakan konfigurasi perangkat keras (Gambar 3). Blok diagram alat menggunakan sebuah modul DT-AVR sebagai sistem kontrol utama yang mengatur dan mengolah data dari modul GSM dan modul masukan/keluaran. DT-AVR *Low Cost Mirco System* adalah sebuah modul *microcontroller* AVR tipe ATmega32 yang memiliki kemampuan untuk melakukan komunikasi data serial secara UART RS-232 serta pemrograman memori melalui ISP (*In-System Programming*). Modul DT-AVR akan dihubungkan dengan telepon seluler Siemens M65® menggunakan kabel data berbasis komunikasi serial RS-232C. Sebagai masukannya digunakan delapan buah sakelar yang dihubungkan pada port B dan keluarannya menggunakan modul DT-I/O Relay Board v2.0 yang dihubungkan pada port D. Dengan konfigurasi seperti ini, keluaran tersebut dapat menggerakan enam buah relay RLY3 – RLY8.

Untuk memudahkan proses pelacakan kesalahan dan penelusuran eksekusi alur program pada modul CT-AVR, modul LCD ditambahkan. Dengan tampilan LCD, proses pengeksekusian perintah dalam modul DT-AVR dapat dengan mudah ditelusuri. Apabila terjadi kesalahan, proses pencarian kesalahan dan perbaikan dapat dilakukan dengan cepat. Modul LCD yang digunakan adalah EMS LCD Display merupakan modul LCD 16 karakter x 2 baris. Dalam tampilan LCD ini akan ditampilkan proses inisialisasi, tombol yang aktif, dan pesan SMS yang diterima.



Gambar 3. Blok diagram perangkat keras.

Perangkat keras yang digunakan tersebut akan diberikan program berdasarkan diagram alir yang ditunjukkan pada Gambar 4. Pada diagram alir tersebut digambarkan alat ini pada awalnya akan melakukan inisialisasi port-port masukan dan keluaran unit *microcontroller*.



Gambar 4. Diagram alir perangkat lunak utama.

Setelah unit *microcontroller* dan *port* masukan keluaran yang akan digunakan terinisialisasi, kemudian dilanjutkan tahap inisialisasi modem untuk menguji komunikasi dengan modem dan mempersiapkan modul untuk dapat menerima SMS yang masuk. Untuk menginisialisasi modem tersambung, aplikasi yang berjalan di DT-AVR harus melalui langkah yang disebutkan dalam Tabel 14.

Tabel 14
Penjelasan Pengaturan Telepon (Modem GSM)

Perintah Terkirim	Pesan dari Modem	Komentar
”AT”	”AT” ”OK”	Modem tersedia dan kompatibel dengan perintah-AT.

	”ERROR”	Telepon tidak terhubung.
”ATE0”	”ATE0” ”OK”	Echo off
	”ERROR”	Telepon tidak terhubung atau kesalahan perintah.
"AT + CPM =" ME "," ME "," ME ""	"+ CPM:" "OK"	Pemilihan lokasi memori berhasil.
	"ERROR"	Satu atau lebih lokasi memori tidak tersedia
"AT + CNMI = 1,1,0,0,1"	"OK"	Indikasi pesan baru diaktifkan.
	"ERROR"	Tidak dapat mengaktifkan mode ini.

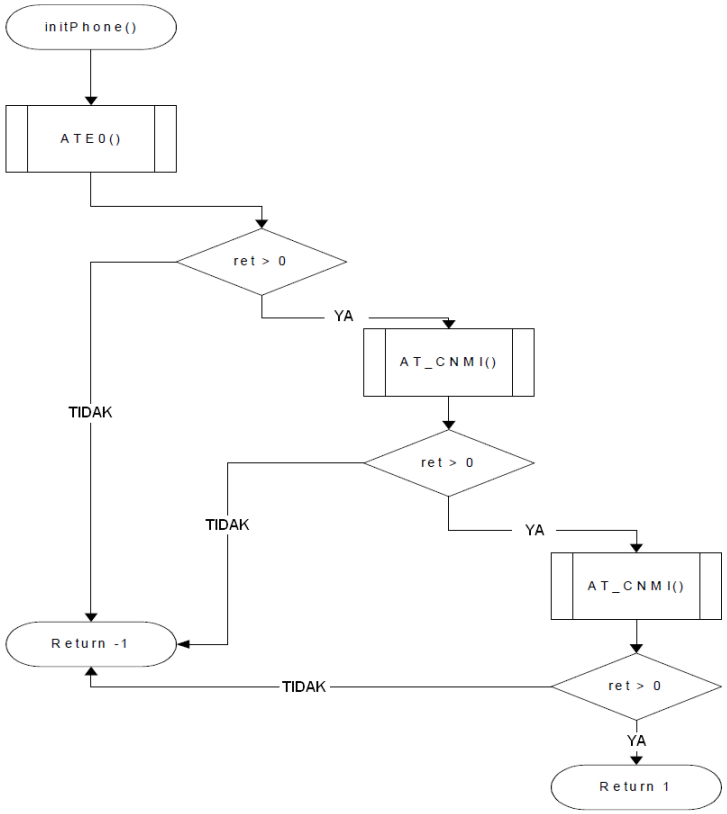
Tidak ada cara bagi aplikasi untuk mengetahui apakah sambungan ke modem terputus atau tidak. Tidak ada jabat tangan diterapkan sehingga pemberian perintah kemodem bisa saja terjadi tidak ada tanggapan balik sebagai pengakuan bahwa perintah sudah diterima. Solusinya adalah dengan memulai pencacah ketika perintah-AT baru dikirim. Jika tidak ada string "OK" diterima dalam periode *timeout* ini, maka terjadi kondisi error untuk mencegah peristiwa *loop* tak terbatas yang dihasilkan akibat sambungan ke modem yang terputus. Solusi lain adalah dengan melaksanakan jabat tangan atau memiliki telepon yang menyediakan pin sinyal khusus untuk mendeteksi sambungan komunikasi serial ke telepon.

Kode pseudo berikut menunjukkan bagaimana perintah-AT dikirim. Kirim AT-Command dengan menggunakan printf(<perintah-AT>);

```
Pewaktu dimulai;
    TUNGGU untuk pewaktu selesai atau pesan "OK" diterima dari modem
        JIKA(Pewaktu kedaluwarsa || "Error" diterima)
                Kembalikan nilai -1;
        JIKA("OK")
                Kembalikan nilai 1
Selesai
```

Setelah mengirim perintah-AT dengan cara ini, kode aplikasi dapat menentukan status modem dengan aman.

Melanjutkan proses inisalisasi modem, perlu diberikan perintah "ATE0" untuk mengaktifkan *echo off*. Jika modem mengakui dengan mengembalikan pesan "OK" bertanda semuanya baik-baik saja, dan perintah berikutnya dari Tabel 15 dapat dilaksanakan. Jika terjadi kesalahan pengaturan kembali rutin kode kesalahan. Lihat Gambar 5 untuk diagram alur lengkap untuk prosedur "API_modem_init ()".



Gambar 5. Diagram alir prosedur API_modem_init ().

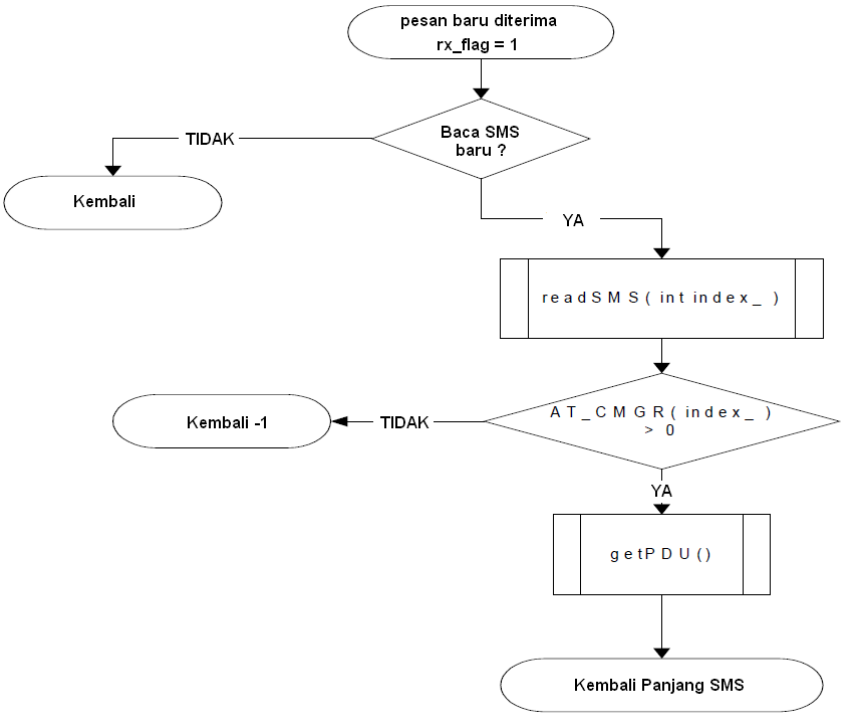
Selanjutnya, modul akan selalu memantau adanya tombol sensor yang tertekan/aktif atau adanya SMS yang masuk. Bila ada tombol sensor yang tertekan maka sistem akan langsung mengirimkan SMS kepada sebuah nomor telepon genggam yang ditetapkan dengan format PDU yang dikompresi berdasarkan diagram alir pada Gambar 2-1.

Tabel 15
Status Tombol dan Pesan SMS yang dikirim

Status Tombol	Port Masukan	Pesan SMS yang dikirim
S1 aktif	Port C Pin 0	”S1 ON”
S2 aktif	Port C Pin 1	”S2 ON”
S3 aktif	Port C Pin 2	”S3 ON”
S4 aktif	Port C Pin 3	”S4 ON”
S5 aktif	Port C Pin 4	”S5 ON”
S6 aktif	Port C Pin 5	”S6 ON”
S7 aktif	Port C Pin 6	”S7 ON”
S8 aktif	Port C Pin 7	”S8 ON”

Untuk menghindari pengiriman pesan SMS secara terus-menerus selama tombol tertekan terus, pada sistem ini diatur pesan SMS hanya dikirimkan sekali dalam interval sepuluh menit selama sebuah tombol dalam keadaan aktif. Agar pesan tidak dikirim dalam selang sepuluh menit secara terus-menerus selama tombol aktif maka dapat diberikan perintah ”ACK” melalui SMS untuk menonaktifkan pendeteksian tombol tersebut. Untuk menonaktifkan waktu tunggu dan membatalkan perintah ”ACK” ini dapat dilakukan dengan perintah ”RSTALL”.

Selain itu, modul juga akan menanggapi pesan SMS yang masuk untuk mengaktifkan atau menonaktifkan relay-relay pada Relay Board.



Gambar 6. Diagram alur proses penerimaan pesan SMS masuk.

Pesan SMS yang dapat dieksekusi oleh modul *microcontroller* adalah sebagai berikut (Tabel 16).

Tabel 16.
Perintah SMS yang Dapat Diberikan

Perintah SMS	Tindakan	Perintah SMS	Tindakan
SET1	Relay RLY3 menutup (ON).	CLR1	Relay RLY3 membuka (OFF).
SET2	Relay RLY4 menutup (ON).	CLR2	Relay RLY4 membuka (OFF).
SET3	Relay RLY5 menutup (ON).	CLR3	Relay RLY5 membuka (OFF).
SET4	Relay RLY6 menutup (ON).	CLR4	Relay RLY6 membuka (OFF).
SET5	Relay RLY7 menutup (ON).	CLR5	Relay RLY7 membuka (OFF).
SET6	Relay RLY8 menutup (ON).	CLR6	Relay RLY8 membuka (OFF).
SETALL	Semua relay menutup (ON).	CLRALL	Semua relay membuka (OFF).
ACK1	Menonaktifkan sensor S1.	ACKALL	Menonaktifkan semua tombol sensor.
ACK2	Menonaktifkan sensor S2.	RSTALL	Membatalkan waktu tunggu dan mengaktifkan kembali semua tombol sensor.
ACK3	Menonaktifkan sensor S3.	PLS1	Relay RLY3 menutup selama 1 detik.
ACK4	Menonaktifkan sensor S4.	PLS2	Relay RLY4 menutup selama 1 detik.
ACK5	Menonaktifkan sensor S5.	PLS3	Relay RLY5 menutup selama 1 detik.
ACK6	Menonaktifkan sensor S6.	PLS4	Relay RLY6 menutup selama 1 detik.

ACK7	Menonaktifkan sensor S7.	PLS5	Relay RLY7 menutup selama 1 detik.
ACK8	Menonaktifkan sensor S8.	PLS6	Relay RLY8 menutup selama 1 detik.

HASIL DAN PEMBAHASAN

Pengujian antarmuka modul *microcontroller* AVR dengan Modem GSM ini dilakukan dengan menuliskan kode program bahasa C yang telah dibuat pada modul DT-AVR melalui antarmuka ISP (*In-System Programming*). Kemudian port B modul DT-AVR dihubungkan pada delapan buah sakelar S1 hingga S8 sebagai piranti masukan. Modul DT-I/O *Relay Board* v2.0 sebagai piranti keluaran dihubungkan pada port D modul DT-AVR. Hubungan antara modem GSM dengan unit DT-AVR dilakukan dengan menggunakan jalur komunikasi RS-232. Pesawat telepon genggam Siemens M65® digunakan sebagai modem GSM dalam pengujian ini. Provider GSM yang digunakan adalah IM3 dari Indosat. Pesawat GSM tujuan juga menggunakan provider yang sama.

Tujuan pengujian ini adalah untuk mengetahui apakah program antar muka modem GSM dengan *microcontroller* dapat bekerja dengan baik atau tidak. Proses pengujian dilakukan sebanyak 100 kali pengiriman dan penerimaan SMS. Dari hasil pengujian, didapatkan bahwa pesan yang dikirimkan oleh *microcontroller* ke pesawat telepon genggam tujuan dapat diterima dengan baik dengan tingkat keberhasilan 95%. Sedangkan perintah SMS dari pesawat telepon genggam yang dikirimkan ke modul *microcontroller* dapat diterima dengan baik dengan tingkat keberhasilan 97%. Dari hasil pengujian yang telah dilakukan sebanyak 100 kali, terlihat bahwa tingkat terjadinya kegagalan sangat kecil sekali, dan hal ini sebenarnya bukan gagal sepenuhnya tetapi dapat juga disebabkan karena padat lalulintas komunikasi SMS sehingga pengiriman SMS tertunda (*pending*). Penyebab lain tidak diterimanya pesan yang dikirimkan adalah karena kualitas sinyal GSM yang kurang baik atau tidak stabil.

Pengujian komunikasi antara modul DT-AVR dengan Modul Relay Board v2.0 dilakukan dengan perngiriman perintah pengendali melalui telepon genggam pengendali. Setiap kali pengiriman pesan singkat atau SMS, modul DT-AVR akan mengaktifkan salah satu relay pada kondisi sesuai dengan apa yang telah di programkan seperti pada Tabel 16. Perintah SMS dikirimkan dari telepon genggam ke *microcontroller* sesuai dengan perintah pada Tabel 16 kemudian tanggapan dari *relay board* diamati. Hasil pengamatan yang diperoleh, semua perintah yang diberikan dari daftar Tabel 16 menghasilkan tanggapan yang sesuai dengan yang diharapkan. Dengan demikian, dapat dinyatakan bahwa antarmuka Modem GSM dengan *microcontroller* AVR yang ditulis dalam bahasa C tersebut sudah berfungsi dengan baik.

PENUTUP

Simpulan

Sesudah melakukan perancangan antarmuka modem GSM dengan *microcontroller* AVR yang ditulis dalam Bahasa C serta melakukan beberapa percobaan untuk menguji fungsionalitas modul anatarmuka tersebut, kesimpulan yang diperoleh melalui penelitian ini adalah sebagai berikut: (1) Antarmuka Modem GSM dengan *microcontroller* AVR yang ditulis dalam Bahasa C tersebut sudah berfungsi dengan baik. Modul *microcontroller* dapat mengirimkan pesan ke telepon genggam melalui jaringan GSM dan sebaliknya modul *microcontroller* juga dapat menerima pesan yang diterima melalui jaringan GSM; (2) Prosedur kompresi dan dekompresi dari format teks ke PDU dan sebaliknya berfungsi dengan baik dan memudahkan pemrogram dalam membuat aplikasi pengiriman atau penenerimaan pesan dalam bentuk teks SMS. Tidak perlu proses pengubahan format teks ke PDU

dan sebaliknya secara manual; (3) Pesan yang dikirimkan oleh *microcontroller* ke pesawat telepon genggam tujuan dapat diterima dengan baik dengan tingkat keberhasilan 95%. Sedangkan perintah SMS dari pesawat telepon genggam yang dikirimkan ke modul *microcontroller* dapat diterima dengan baik dengan tingkat keberhasilan 97%; (4) Modul yang dikembangkan dalam penelitian ini dapat digunakan sebagai pengendali jarak jauh melalui SMS dengan jaringan telepon GSM; (5) Penggunakan bahasa C dalam pemrograman *microcontroller* memudahkan dan mempercepat pengembangan aplikasi. Pemrogram tidak perlu memahami secara mendalam perangkat keras *microcontroller*.

Saran

Sementara itu berikut adalah beberapa saran berdasarkan temuan yang ada. Pertama, aplikasi antarmuka *microcontroller* AVR dengan modem GSM yang berhasil dirancang merupakan gabungan dari beberapa subrutin dalan satu file, sehingga file program terlihat besar dan panjang. Peneliti menyarankan agar beberapa prosedur utama dapat dijadikan prosedur standar yang dapat dijadikan header file. Dengan demikian aplikasi yang dikembangkan dengan memanfaatkan antarmuka ini dapat lebih kompak.

Kemudian, belum ada prosedur yang berfungsi untuk mendapatkan informasi nomor telepon pengirim dari pesan SMS yang diterima oleh modem GSM. Prosedur ini perlu ditambahkan sehingga dengan mengenali nomor telepon pengirim pesan dapat ditambahkan fitur sekuriti. Fitur ini penting agar tidak semua nomor telepon dapat mengendalikan *microcontroller* tersebut.

Dapat ditambahkan pula fasilitas untuk pengiriman pesan pada beberapa nomor telepon tujuan sekaligus. Pada antarmuka yang dirancang disini hanya dapat mengirimkan pesan pada satu nomor telepon tujuan saja. Agar lebih fleksibel dapat juga diatur pengguaan nomor telepon dari buku telepon modem GSM.

Dalam hal penanganan pesan kesalahan, implementasi dalam aplikasi ini hanya akan menangkap pesan ERROR, dan mengulangi perintah kembali. Jika diinginkan penanganan pesan kesalahan yang lebih maju maka pengembang harus mengacu pada lembaran data teknis modem.

DAFTAR PUSTAKA

Atmel Corporation. (2006). *Application Note AVR323: Interfacing GSM Modems*. San Jose, USA: Atmel Corporation.

Atmel Corporation. (2006). *Datasheet 8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash ATmega8535(L)*. San Jose, USA: Atmel Corporation,.

Siemens. (2000). *Manual Reference AT Command Set (GSM 07.07, GSM 07.05, Siemens specific commands) for the SIEMENS Mobile Phones S3i, C35i, M35i*. Germany: Siemens AG.