GSM-MODUULI SULAUTETUSSA JÄRJESTELMÄSSÄ

Timo Rantanen

Opinnäytetyö
Joulukuu 2011
Tietotekniikka
Sulautetut järjestelmät ja elektroniikka
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Sulautettujen järjestelmien ja elektroniikan suuntautumisvaihtoehto

RANTANEN, TIMO: GSM-moduuli sulautetussa järjestelmässä

Opinnäytetyö 39 s., liitteet 10 s.
Joulukuu 2011

Tässä opinnäytetyössä tutustuttiin GSM-moduulin ohjaukseen ja käyttöön sulautetussa järjestelmässä. Työssä käytettiin apuna Olimexin AVR-GSM-kehitysalustaa. Kehitysalustan tärkeimmät komponentit ovat ATMega32-mikrokontrolleri sekä Simcomin SIM300D GSM-moduuli.

Työssä esitellään AVR-GSM-kehitysalusta ja SIM300D GSM-moduulin ohjaus AT-komennoilla. Työssä tehtiin myös esimerkkisovellus, joka havainnollistaa kehitysalustan ja GSM-moduulin käyttöä ja ohjausta. Opinnäytetyön ohjelmistokehitys tehtiin C-kielellä AVR-Studio-kehitysympäristössä.

Työn tuloksena saadussa esimerkkisovelluksessa toteutettiin peruskommunikointi GSM-moduulin, ATMega32-mikrokontrollerin sekä käyttäjän matkapuhelimen välillä. Toimintoja luotiin käyttäen kehityskortin mahdollisuuksia. Pidemmälle viety käytännön sovellus, jossa on ulkoisia komponentteja esimerkiksi erilaisia antureita, on helppo toteuttaa tämän työn pohjalta.

Avainsanat: GSM-moduuli, AVR-GSM-kehitysalusta, AT-komennot

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Computer Systems Engineering
Option of Embedded Systems and Electronics

RANTANEN, TIMO: GSM-module in embedded systems

Bachelor's thesis 39 pages, appendices 10 pages
December 2011

The purpose of this thesis was to study the control and usage of GSM-module in embedded systems. Development board used in this thesis was Olimex's AVR-GSM board. The most essential components of the development board are ATMega32-microcontroller and Simcom's SIM300D GSM-module.

Thesis introduces AVR-GSM-development board and usage of SIM300D GSM-module with AT-commands. Thesis also contains example application that demonstrates the usage of GSM-module and the development board. Software development was done with C-language in AVR-Studio development environment.

This work resulted in example application that will provide basic communications between GMS-module, ATMega32-microcontroller and the users mobile phone. More advanced application that includes external components such as different sensors, is easy to develop based on the work done in this thesis.

Keywords: GSM-module, AVR-GSM-development board, AT-commands

ALKUSANAT

Tämä työ on tehty Tampereen ammattikorkeakoulun tietotekniikan koulutusohjelman opinnäytetyönä syksyllä 2011. Työn tarkoituksena oli tutustua GSM-moduulin ohjaukseen ja käyttöön sulautetussa järjestelmässä. Haluan kiittää Tampereen ammattikorkeakoulua sekä työn ohjaajaa työssä käytetyn kehitysalustan hankkimisesta.

Tampereella 5.12.2011

Timo Rantanen

SISÄLLYSLUETTELO

LYHENTEET JA TERMISTÖ

GSM	Global System for Mobile Communications, matkapuhelinjärjestelmä, jota käytetään maailmanlaajuisesti.
SMS	Short message service, matkapuhelinten käyttämä tekstiviestijärjestelmä.
AT	Tulee sanasta “attention”, tunnetaan myös nimellä Hayes-komentosarja, on komentokieli, joka luotiin alunperin Hayes-smartmodemin ohjaukseen.
SIM	Subscriber Identity Module, on älykortti, jota käytetään matkapuhelinliittymän tilaajan yksilöllisen IMSI-avaimen tallentamiseen.
PIN	Personal Identification Number, on salasanana käytetetty luku, jolla SIM-kortti avataan.

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on tutustua Olimexin AVR-GSM-kehitysalustaan sekä GSM-moduulin käyttöön sulautetussa järjestelmässä. GSM-moduulin käyttö sulautetussa järjestelmässä mahdollistaa monia eri matkapuhelimella etäohjattavia järjestelmiä. Myös erilaiset tiedonkeruujärjestelmät, joista informaatio tulee tekstiviestinä, ovat mahdollisia. Esimerkiksi etäkäynnistettävä sähkökiuas tai muu laite voisi olla eräs sovelluskohde. Informaationkeruuna lämpötilan tai muiden tietojen haku GSM-verkon kautta matkapuhelimeen on mahdollista.

Työssä esitellään kehitysalustan sekä GSM-moduulin käyttö ja ominaisuudet pääpiirteittäin sekä tutustutaan GSM-moduulin ohjaukseen AT-komennoilla. Työssä luodaan myös esimerkkisovellus kehitysalustalle. Sovellus keskittyy peruskommunnikoinnin luomiseen mikrokontrollerin ja GSM-moduulin välille sekä tärkeimpien toimintojen ohjelmointiin. SMS-viestin kautta vastaanotettavat komennot, raporttiviestin lähetys käyttäjälle sekä puheyhteyden muodostaminen ovat eräitä ominaisuuksia joita sovelluksessa luodaan. Ohjelmistokehitys tehdään C-kielellä AVR-Studio-kehitysympäristössä.

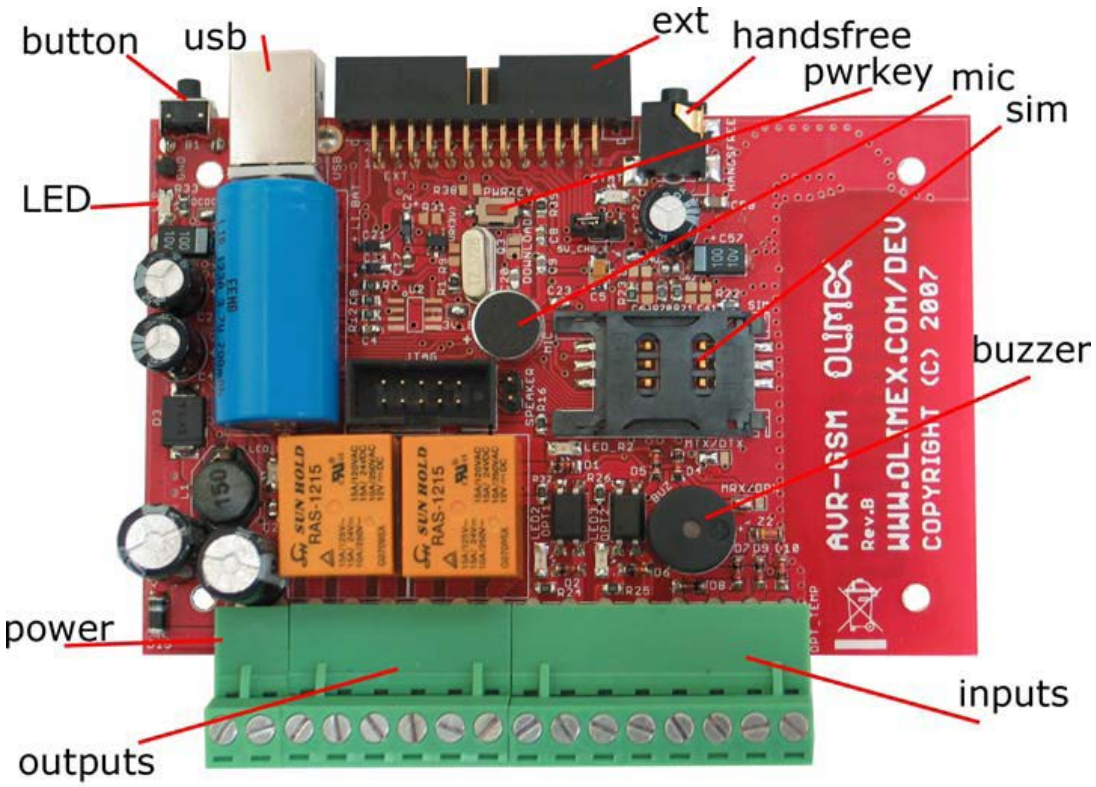
2 AVR-GSM-KEHITYSALUSTA

Tässä luvussa esitellään opinnäytetyössä käytetyn kehitysalustan ominaisuuksia. GSM-moduulin ominaisuuksia ja AT-käskyjä käsitellään tarkemmin myöhemmissä luvuissa.

2.1 Kehitysalustan ominaisuudet

Työssä käytettiin Olimexin valmistamaa AVR-GSM-kehitysalustaa. Käytetty kehitysalusta on erinomainen työkalu GSM-verkkoa tiedonsiirtoon käyttävien sovellusten kehitykseen. Tällaisia ovat erilaiset matkapuhelimella etäohjattavat laitteet tai tiedonkeruulaitteet, joista informaatio tulee käyttäjälle tekstiviestinä.

Kehitysalusta pohjautuu Atmelin ATMega32-mikrokontrolleriin sekä Simcomin SIM300D GSM-moduuliin. Kuvassa 1 on esitetty kehitysalustan muita oleellisia komponentteja.



Kuva 1. Olimex AVR-GSM kehitysalusta [1, s. 9]

AVR-GSM-kehitysalustan tärkeimmät ominaisuudet:

- ATMega32-mikrokontrolleri, jossa on 32 kB flash-, 2 kB RAM- ja 1 kB EEPROM-muistia
- AVR-JTAG-USB-ohjelmointi- ja debuggausliitäntä
- SIM300D GSM/GPRS-moduuli, toimintataajuudet 900/1800/1900 MHz
- Kehitysalustaan sisäänrakennettu antenni
- Li-ion-vara-akku, 3,7 V / 700 mAh
- SIM-kortin pidike
- Kaksi 240 VAC / 10 A relettä
- Kaksi optoerotettua sisääntuloa
- USB-liitäntä
- 32 ohm kaiutinliitäntä
- 2,5 mm Handsfree-liitin
- Summeri
- GSM-moduulin status-LED
- Digitaalinen lämpötila-anturi (TCN75A)
- 26-pinninen liitin ATMega32-mikrokontrollerin ja GSM-moduulin vapaille nastoille
- Kehitysalustan mitat: 130x82x34 mm
  [1, s. 2]

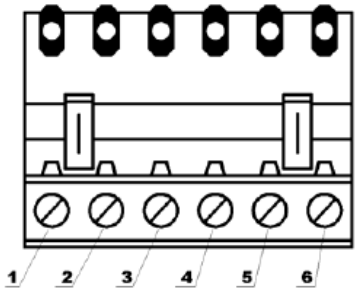
Kehityalustalle syötetään 12 VDC käyttöjännite erillisestä liitännästä ja jumpperien avulla voidaan valita, onko vara-akku käytössä vai ei. Tarkempi ohjeistus jumpperikytkentöihin on esitetty kehitysalustan käyttöohjeessa.

Alustalla olevan ATMega32-mikrokontrollerin käyttöjännitte on 3 V ja se toimii 7,37 MHz kiteellä. Kehitysalustan kytkentäkaavio on nähtävissä liitteistä.

2.2 Kehitysalustan tulo- ja lähtöliitännät

AVR-GSM kehitysalusta sisältää kaksi 240 VAC / 10 A relettä, joita voidaan käyttää ulkoisen laitteen ohjaukseen. Releet kytkeytyvät lähtöliittimeen kuvan 2 mukaisesti.

Pin #	Signal Name
1	NO-REL1
2	COMMON-REL1
3	NC-REL1
4	NO-REL2
5	COMMON-REL2
6	NC-REL2



Kuva 2. Releiden liitin [1, s. 12]

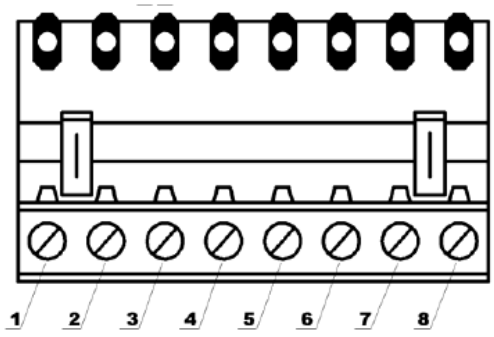
Releitä voidaan käyttää ulkoisen kuorman kytkemiseen. Seuraavia arvoja ei tule ylittää.

- 15 A / 125 VAC
- 10 A / 250 VAC
- 15 A / 24 VDC

Relettä 1 ohjataan ATMega32-mikrokontrollerin nastasta PC7 ja releellä on merkkivalona alustalla oleva LED_R1. Vastaavasti relettä 2 ohjataan nastasta PC6 ja sillä on merkkivalona alustalla oleva LED_R2.

Kehitysalusta sisältää myös kaksi optoerotinta sekä I2C-liitännän, jotka kytkeytyvät tuloliittimeen kuvan 3 mukaisesti.

Pin #	Signal Name
1	DIGITAL IN1 +
2	DIGITAL IN1 -
3	DIGITAL IN2 +
4	DIGITAL IN2 -
5	GND
6	SCL
7	SDA
8	3V

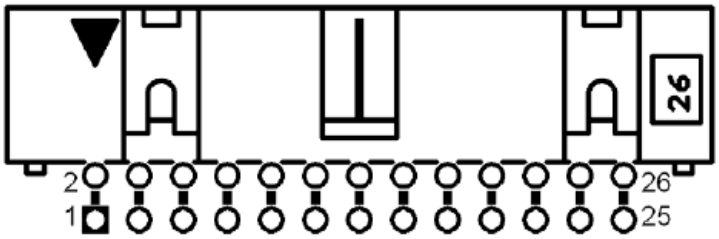


Kuva 3. Optoerottimien ja I2C:n liitin [1, s. 13]

Optoerottimien tulotaso on 5 – 12 VDC. Jännite kytkettynä optoerottimeen 1 (positiivinen jännite nastaan 1 ja negatiivinen jännite tai maa nastaan 2) aiheuttaa loogisen tason 0 mikrokontrollerin tulonastaan PB0. Vastaavasti jännite kytkettynä optoerottimeen 2 aiheuttaa loogisen tason 0 tulonastaan PD2. Merkkivaloina optoerottimille toimivat alustalla olevat LED2 ja LED3.

I2C-signaaleja SCL (Serial Clock) ja SDA (Serial Data) voidaan käyttää ulkoisen lämpötilasensorin tai muun sopivan I2C-laitteen kytkemiseen.

Alustalla on lisäksi liitin ATMega32-mikrokontrollerin sekä GSM-moduulin käyttämättömille nastoille. Liittimen pinnijärjestys on esitetty kuvassa 4.



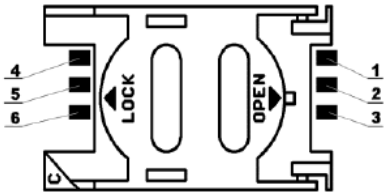
Pin #	Signal Name	Pin #	Signal Name
1	BACKUP	2	AREF
3	GND	4	3VA
5	3V	6	AGND
7	VBAT	8	(ADC3)/PA3
9	+5V	10	PWRKEY - (ADC2)/PA2
11	POWERKEY-pin12 of GSM module	12	(ADC1)/PA1
13	AUXADC	14	(ADC0)/PA0
15	GPO1	16	(SCK)PB7
17	SPI_DATA	18	(MISO)PB6
19	SPI_CLK	20	(MOSI)PB5
21	SPI_CS	22	(SS)PB4
23	SPI_D/C	24	(T1)PB1
25	KBROW0	26	RST

Kuva 4. Mikrokontrollerin ja GSM-moduulin ylimääräiset nastat [1, s. 14]

Liitin sisältää mm. GSM-moduulin SPI-liitännän, ulkoisen näppäimistöliitännän sekä ATMega32-mikrokontrollerin A/D-muunninkanavat ADC0, ADC1 ja ADC3. Liittimestä päästään käsiksi myös powerkey- ja reset-nastoihin.

Kehitysalusta sisältää standardin 6-pinnisen SIM-kortin liittimen. Liittimen pinnijärjestys on esitetty kuvassa 5.

Pin #	Signal Name
1	VSIM
2	SIMRST
3	SIMCLK
4	GND
5	NC
6	SIMDATA



Kuva 5. SIM-kortin liitin [1, s 11]

Toimiakseen GSM-verkossa alusta tarvitsee tavallisen matkapuhelinoperaattorin SIM-kortin. Käynnistettäessä GSM-moduulia se kysyy SIM-kortin PIN-koodia. Koodi voidaan antaa AT-komennoilla, mutta kortti voidaan asettaa siten, että se ei kysy PIN-koodia. Tämä voidaan tehdä asettamalla kortti ensin tavalliseen matkapuhelimeen, jonka avulla PIN-koodin kysely poistetaan käytöstä.

Alusta sisältää merkkiledin STAT, joka kertoo GSM-moduulin tilasta.

- Led on pois päältä – GSM-moduuli ei ole käynnissä
- 64 ms On / 800 ms Off – GSM-moduuli ei löydä verkkoa
- 64 ms On/ 3000 ms Off – GSM-moduuli on kytkeytynyt verkkoon
- 54 ms On / 300 ms Off – GPRS-kommunikointi
  [2, s. 43]

AVR-GSM-kehitysalusta sisältää lisäksi muutamia muita liitäntöjä, joiden toimintaa kuvataan seuraavassa:

Kehitysalusta sisältää FT232RL USB to UART-muuntopiirin. Jos alusta on kytketty kaapelilla PC:n USB-porttiin, kommunikointi tapahtuu PC:n ja GSM-moduulin välillä. Kun alusta ei ole kytkettynä USB-kaapelilla, kommunikointi tapahtuu ATMega32-mikrokontrollerin ja GSM-moduulin välillä.

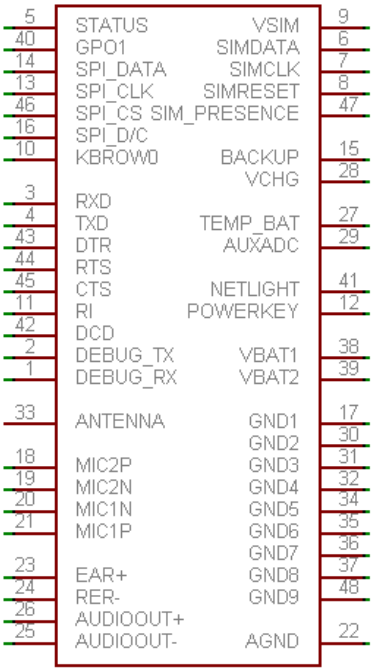
Alustalla on 2,5 mm handsfree-liitin audiolähtö ja audiotulo sekä erillinen alustalla oleva mikrofoni. AT-käskyillä voidaan valita, otetaanko äänitulo handsfree-liitännästä vai alustalla olevasta mikrofonista. Kehitysalustalla on myös erillinen 32 ohm kaiutinliitin ja AT-käskyillä valitaan, ohjataanko äänisignaali erilliseen kaiutinliitäntään vai handsfree-liitäntään. Alustalla on myös summeri, jota voidaan käyttää vaikkapa tulevan soiton ilmaisemiseen, sekä painonappi, joka kytkeytyy mikrokontrollerin nastaan PD3.

Lisäksi on myös GSM-moduulin powerkey-painike. Jos painiketta painetaan noin kolme sekuntia moduulin ollessa ON-tilassa, menee moduuli power-down tilaan. Jos nappia painetaan kauemmin kuin yksi sekunti moduulin ollessa power-down tilassa, se menee ON-tilaan. Moduulin käynnistäminen voidaan suorittaa myös ohjelmallisesti asettamalla pinni PA2 alatilaan 2-3 sekunnin ajaksi.

3 GSM-MODUULIN OMINAISUUDET

Kehitysalustalla oleva GSM-moduuli on Simcomin valmistama SIM300D (Kuva 6). Integroidun latauspiirin ansiosta se soveltuu hyvin akulla toimiviin sovelluksiin. Moduuli sisältää tietenkin GSM-puhelimelle tavanomaisia ominaisuuksia. Moduulia ohjataan AT-komennoilla sarjaliikenneliitännän kautta. AT-komentoja käsitellään tarkemmin seuraavassa luvussa. Moduulin pinnien nimet ja vastaavat numerot on esitetty kuvassa 7.

Kuva 6. SIM300D GMS-moduuli



Kuva 7. SIM300D pinnit

Moduulin keskeisiä ominaisuuksia:

- Toimintataajuudet 900/1800/1900 MHz GSM/GPRS
- Puhelinluettelo
- SMS-lähetys ja -vastaanotto
- SMS-viestin tallennus SIM-kortille.
- Kaksi sarjaporttia
- Kaksi audiokanavaa sisältää kaksi mikrofonituloa ja kaksi kaiutinlähtöä
- Käyttöjännite 3,4 V – 4,5 V
- Toimintalämpötila -20°C - +55°C
- Pieni virrankulutus, SLEEP-moodissa 3 mA
- Integroitu latauspiiri Li-Ion-akulle
- Moduulin mitat 33x33x3 mm, paino 8 g
  [2, s. 11]

3.1 GSM-moduulin toimintatilat

SIM300D-moduulilla on useita eri toimintatiloja. Eri tilat on esitetty taulukossa 1. Eri toimintatilojen virrankulutus riippuu GMS/GPRS-asetuksista. Siirtyminen eri tilojen välillä on esitetty liitteissä.

Taulukko 1. SIM300D-moduulin toimintatilat [2, s. 15]

MOODI	TOIMINTA	
NORMAALITILA	GSM/GPRS SLEEP	Moduuli menee SLEEP-tilaan, kun DTR pinni asetetaan ylätilaan. Virrankulutus menee minimiin.
	GSM IDLE	Ohjelmisto on toiminnassa, moduuli on rekisteröitynyt GSM-verkkoon ja on valmis vastaanottamaan ja lähettämään.
	GSM TALK	CDS-yhteys on käynnissä.
	GPRS IDLE	Moduuli on valmiina datan siirtoon, mutta dataa ei tällä hetkellä lähetetä tai vastaanoteta.
	GPRS DATA	GPRS-dataa lähetetään / vastaanotetaan.
POWER DOWN	Virta katkeaa moduulin baseband-puolelta. Ainoastaan reaaliaikakellolla on virta. Ohjelmisto ei ole toiminnassa ja sarjaliikenne ei ole käytettävissä.	
Minimitoiminta	RF-osio ei ole toiminassa eikä SIM-kortti ei ole käytössä. Sarjaliitäntä on toiminnassa.	
Alarm-tila	Rajoitettu toiminta ja vain osa AT-käskyistä on käytössä. Moduuli ei ole kytkeytynyt GSM-verkkoon.	
GHOST-tila	Vain latausmoodi.	
Normaalitilan lataus	Lataus, kun moduuli on normaalitilassa. Sisältää tilat SLEEP, IDLE, TALK, GPRS IDLE ja GPRS DATA.	

3.2 GSM-moduulin virransyöttö

GSM-moduulille syötetään käyttöjännite VBAT-pinnien kautta. Käyttöjännite moduulille on 3,4 – 4,5 V ja virrankulutus voi nousta aina 2 A asti, joten virtalähteen on kyettävä tähän. Mikäli käyttöjännite putoaa alle 3,4 V:n moduuli voi sammua. Moduulin käyttöjännite syötetään tyypillisesti akulta. Tätä varten GSM-moduuli sisältää integroidun latauspiirin. [2, s. 17]

Integroitu latauspiiri

Moduuli sisältää integroidun latauspiirin Li-ion akulle. Latausalgoritmi on optimaalinen akulle, joka täyttää seuraavat ominaisuudet.

- Jännite 3,7 V
- Kapasiteetti 580 mAh
- Latausjännite 4,2 V
- Suurin latausvirta 1,5 C
- Suurin purkausvirta 1,5 C

Lataus tapahtyy yksinkertaisesti kytkemällä ulkoinen latausjännite 5 V GSM-moduulin VCHG-pinniin, jolloin moduuli syöttää latausvirran ja jännitteen VBAT-pinnin kautta akulle. Moduuli tunnistaa, onko latausjännite kytkettynä ja onko akku kytkettynä. Moduulissa on TEMP_BAT-pinni akun lämpötilan seurantaa varten. [2, s.24]

Moduulin käyttöjännitettä sekä akun varausta voidaan seurata AT-komennolla ”AT+CBC”. Tällä komennolla moduuli palauttaa kolme parametria: latauksen tilan, akun varauksen prosenttilukuna 1-100 sekä käyttöjännitteen arvon millivoltteina.

3.3 Moduulin käynnistäminen

SIM300D-moduuli voidaan käynnistää muutamalla eri tavalla, jotka on lueteltu tässä kappaleessa. Moduulin tilaa voidaan seurata STATUS-pinnin avulla.

PWRKEY-pinnin kautta:
Asettamalla PWRKEY-pinni alatilaan kolmen sekunnin ajaksi moduuli käynnistyy normaalitilaan. Kun moduuli on käynnistynyt, se palauttaa viestin ”RDY”. Tämä ilmaisee, että moduuli on toimintavalmiina. [2, s. 18]

VCHG-pinnin kautta:
Jos latausjännite (5 V) kytketään VCHG-pinniin POWER DOWN-tilassa, moduuli menee GHOST-tilaan eli vain lataustilaan. Tässä tilassa moduuli vastaa vain muutamiin AT-komentoihin. Jos moduuli käynnistetään näin, se palauttaa viestit ”RDY” ja ”GHOST MODE”. Asettamalla PWRKEY-pinni alatilaan edellisen kohdan tapaan, moduuli menee normaalitilan latausmoodiin. Tällöin kaikki toiminnot ja AT-komennot ovat käytettävissä. [2, s. 19]

Reaaliaikakellon keskeytyksen kautta:
Reaaliaikakellon ”alert”-toiminto herättää moduulin POWER DOWN-tilasta, jolloin moduuli menee alarm-tilaan. Tällöin moduuli ei rekisteröidy GSM-verkkoon ja vain osa AT-komennoista on käytettävissä. [2, s. 20]

3.4 Moduulin sammuttaminen

Myöskin moduulin sammuttaminen voidaan hoitaa muutamalla eri tavalla, jotka on lueteltu seuraavassa:

PWRKEY-pinnin kautta:
Moduuli voidaan sammuttaa asettamalla PWRKEY alatilaan noin yhden sekunnin ajaksi. Tämä antaa moduulin uloskirjautua verkosta. Se myös mahdollistaa ohjelmiston siirtymisen turvalliseen tilaan sekä tallentaa tiedot, ennenkuin virta katkeaa. Moduuli palauttaa viestin POWER DOWN. [2, s.21]

AT-komennolla:

Moduuli voidaan sammuttaa AT-komennolla ”AT+CPOWD=1”. Tämä antaa moduulin kirjautua ulos verkosta sekä mahdollistaa ohjelman siirtymisen turvalliseen tilaan. Moduuli palauttaa viestin POWER DOWN. [2, s. 21]

Automaattinen sammutus alijännitteessä:

Moduulin ohjelmisto tutkii jatkuvasti VBAT-pinniin kytkettyä käyttöjänitettä. Jos jännite putoaa alle 3,5 V:n, moduuli palauttaa viestin POWER LOW WARNING. Jos VBAT-jännite putoaa alle 3,4 V:n moduuli palauttaa viestin UNDERVOLTAGE POWER DOWN. Tämän jälkeen AT-komennot eivät ole käytössä ja moduuli kirjautuu ulos verkosta sekä menee POWER DOWN-tilaan. [2, s. 22]

Automaattinen sammutus ylikuumenemistilanteessa:

Moduulin ohjelmisto vahtii jatkuvasti moduulin lämpötilaa. Jos mitattu lämpötila on suurempi kuin +85°C tai pienempi kuin -35°C, moduuli kirjautuu ulos verkosta ja menee POWER DOWN-tilaan.

3.5 Sarjaliitäntä

GSM-moduuli sisältää kaksi sarjaliikenneporttia. GSM-moduulin ohjaus tapahtuu AT-komennoilla sarjaliikenneportin 1 kautta. Toinen sarjaliikenneportti on debuggaustarkoitusta varten. Sarjaportti 1 toimii nopeuksilla 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600 ja 115200 bit/s.

Moduuli sisältää autobaud-toiminnon joka tunnistaa käytetyn nopeuden.
Tämä ominaisuus on käytössä oletuksena ja se toimii nopeuksilla 1200 – 115200 bit/s.
Jos käytetään autobaud-toimintoa, on käytettävä seuraavia sarjaliikenteen asetuksia:
8 databittiä, 1 stop-bitti, ei pariteettia. Nämä ovat moduulissa oletusasetuksina.

4 AT-KOMENNOT

Tässä luvussa esitellään AT-komentojen käyttöä ja niiden syntaksia GSM-moduulin ohjauksessa.

Hayes-komennot, joita kutsutaan myös nimellä AT-komennot (attention), ovat komentoja, jotka kehitettiin vuonna 1981 alunperin Hayes Smartmodemia varten. Komennot koostuvat lyhyistä merkkijonoista ja niiden parametreista.

4.1 SIM300D AT-komennot

SIM300D GSM-moduulin käyttämät AT-komennot ovat yhdistelmä GSM 07.07-, GSM 07.05- ja ITU-T V.250-suositusten mukaisa komentoja. Lisäksi moduulilla on joitakin valmistajakohtaisia komentoja. AT-komennoilla kommunikointi GSM-moduulin ja mikrokontrollerin välillä tapahtuu sarjaportin eli pinnien RXD ja TXD kautta. [3, s. 5]

Kaikki AT-komennot alkavat etuliitteellä ”AT” ja komento lopetetaan rivinvaihtoon <CR>. Komentoja seuraa yleensä vastaus muodossa <CR><LF><vastaus><CR><LF>. AT-komennot voidaan jakaa syntaksin perusteella kolmeen eri kategoriaan, jotka ovat ”basic”, ”S parameter” ja ”extended”. [3, s. 5]

Basic syntax

Näillä komennoilla on muoto AT<x><n> tai AT&<x><n>, jossa <x> on komento ja <n> on komennolle ominainen parametri. Esimerkkinä ATE<n>, joka kertoo moduulille, kaiutetaanko vastaanotetut merkit takaisin vai ei. Parametri <n> voi olla joko 0 (kaiutus pois) tai 1 (kaiutus päällä). Jos parametri jätetään antamatta, on käytössä komennon oletusarvo. [3, s. 5]

S parameter syntax

Näiden AT-komentojen muoto on ATS<n>=<m>, jossa <n> on S-rekisterin indeksi ja <m> on arvo, joka siihen asetetaan. Parametri <m> on valinnainen. Jos se jätetään antamatta, on käytössä parametrin oletusarvo. Esimerkiksi komennossa ATS0=<n>, joka asettaa automaattisen puheluun vastaamisen, parametri <n> on odotettavien hälytyskertojen määrä [3, s.5].

Extended syntax

Näillä komennoilla on useita eri toimintoja. Toiminnot on esitetty seuraavassa taulukossa.

Taulukko 2. Extended syntax-komentojen muoto [3, s. 6].

Testikomento	AT+<x>=?	Komento palauttaa listan parametreista ja parametrien arvoalueista, jotka voidaan asettaa kirjoitus- tai suorituskomennoilla.
Lukukomento	AT+<x>?	Palauttaa sen hetkisen tilan, eli parametrin/parametrien arvot.
Kirjoituskomento	AT+<x>=<...>	Suorittaa komennon käyttäjän määrittelemillä parametreilla.
Suorituskomento	AT+<x>	Suorittaa komennon parametreilla, jotka haetaan GMS-moduulin sisäisistä prosesseista. Esimerkiksi numeron haku SIM-kortin muistista tietystä indeksistä.

Moduuli hyväksyy maksimissaan 256 merkkiä pitkän komennon. Jos maksimipituus ylitetään, mitään osaa komennosta ei suoriteta ja moduuli palauttaa viestin ERROR. Kun komentoja syötetään peräjälkeen, jokaisen komennon jälkeen tulee odottaa vastausta OK tai virheilmoitusta, ennenkuin seuraava komento voidaan syöttää.

4.2 Esimerkkejä GSM-moduulin ohjauksesta

GMS-moduulille syötetään komentoja sarjaliikenneliitännän kautta. Moduuli vastaa aina syötettyyn komentoon. Myös esimerkiksi GSM-moduulin vastaanottaessa tekstiviesti tai puhelu, moduuli automaattisesti lähettää viestin sarjaportista. Näitä viestejä lukemalla ja käskyjä lähettämällä voidaan moduulia ohjata mikrokontrollerilla.

Monet GSM-moduulin antamat paluuviestit ja niiden muoto riippuvat moduulin asetuksista. Esimerkiksi onko SMS-formaatti PDU-, vai tekstimuodossa, onko soittajan numeronnäyttö asetettu päälle vai ei, mikä on paluuviestien muoto tai onko viestien näyttö ylipäätään päällä. Tässä luvussa näytetyt paluuviestit ovat moduulin oletusasetusten muotoisia lukuunottamatta seuraavia oletusasetuksista poikkeavia muutoksia, jotka on asetettu käytön helpottamiseksi.

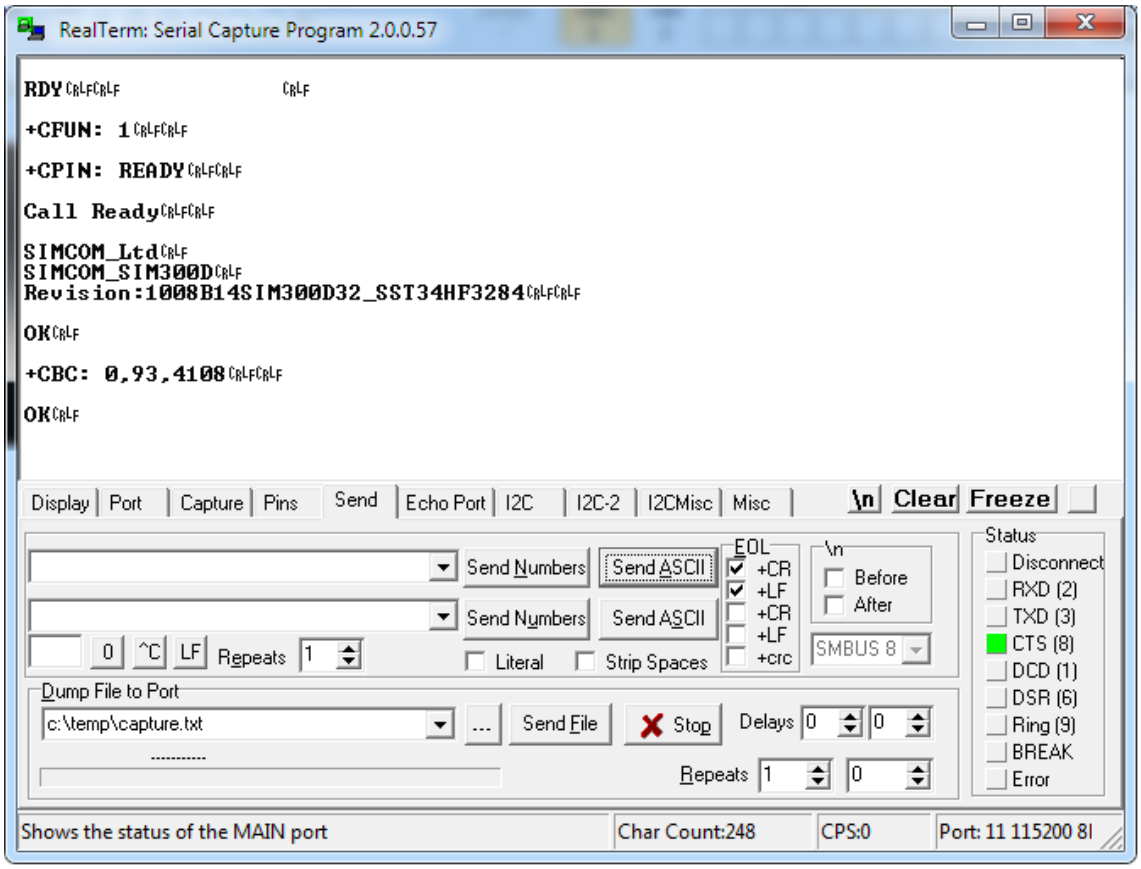
```
ATE0        // Merkkien kaiutus pois päältä.
AT+CMGF     // SMS-formaatti tekstimuotoon
AT+CLIP=1   // Soittajan numeronnäyttö päälle.
```

Kun moduulille syötetään komento, jolla halutaan muuttaa jotakin moduulin asetuksista, tulee paluuviestinä joko OK tai ERROR riippuen siitä onko komennon suorittaminen onnistunut. Kun moduulille syötetään komento, jonka avulla halutaan saada joitakin tietoja, kuten tekstiviestin sisältö tai akun varaus, on paluuviestit muotoa +<komennon tunnus>: <parametrit pilkuilla erotettuina>. Esimerkiksi akun varausta kysyvä komento AT+CBC antaa paluuviestin: +CBC: 0,95,4100
Näillä tiedoilla ja tutkimalla lähteissä mainitun pdf-dokumentin [3] ”SIM300D AT Command set” komentoja ja niiden paluuviestien muotoa, voidaan sarjaportista tulevasta merkkivirrasta hakea haluttuja tietoja.

Tässä työssä käytetty AVR-GSM-kehitysalusta sisältää myös USB-liitännän, jonka kautta moduulia voidaan testata ja ohjata ilman, että on vielä kirjoitettu mitään varsinaista toimivaa ohjelmakoodia. Seuraavissa alaluvuissa esitellään joitakin moduulin peruskomentoja ja moduulin antamia paluuviestejä käyttäen ilmaista RealTerm-terminaaliohjelmaa.

4.2.1 Peruskomentoja

GSM-moduuli antaa paluuviestejä, kun sille syötetään komentoja. Kuvassa 8 näkyy moduulin antamia viestejä, kun se käynnistetään sekä moduulin vastauksia joihinkin peruskomentoihin.



Kuva 8. SIM300D-moduulin testausta RealTerm-ohjelmalla

Osa viesteistä tulee automaattisesti moduulin käynnistyttyä, ja osa viesteistä on vastauksia tiettyihin komentoihin. Viestien merkitykset on selitetty seuraavassa.

Kun moduuli käynnistetään, se ilmoittaa joitakin viestejä tilastaan.

```
RDY             // Moduuli on käynnissä ja valmiina vastaanottamaan
                // komentoja.
+CFUN: 1        // CFUN ilmoittaa moduulin tilan, 1 = Full Functionality.
```

```
+CPIN: READY      // CPIN ilmoittaa PIN-koodin tilan. READY tarkoittaa, ettei
                  // PIN-koodia odoteta syötettäväksi. Tässä tapauksessa on
                  // näin, sillä PIN-koodin kysely on poistettu käytöstä.

Call Ready        // Moduuli on kirjautunut GSM-verkkoon ja on valmis
                  // vastaanottamaan viestejä ja puheluita.
```

Seuraavat paluuviestit ovat moduulin vastauksia komentoon ”ATI”, jolla moduulilta voidaan hakea sen tuotetiedot.

```
SIMCOM_Ltd                                  // Valmistaja
SIMCOM_SIM300D                              // Tuotenimike
Revision: 1008B14SIM300D32_SST43HF3284      // Versio
```

Viimeisenä kuvassa 8 näkyvä paluuviesti on moduulin vastaus komentoon ”AT+CBC”, jolla moduulilta voidaan kysyä siihen kytketyn akun varausta ja latauksen tilaa.

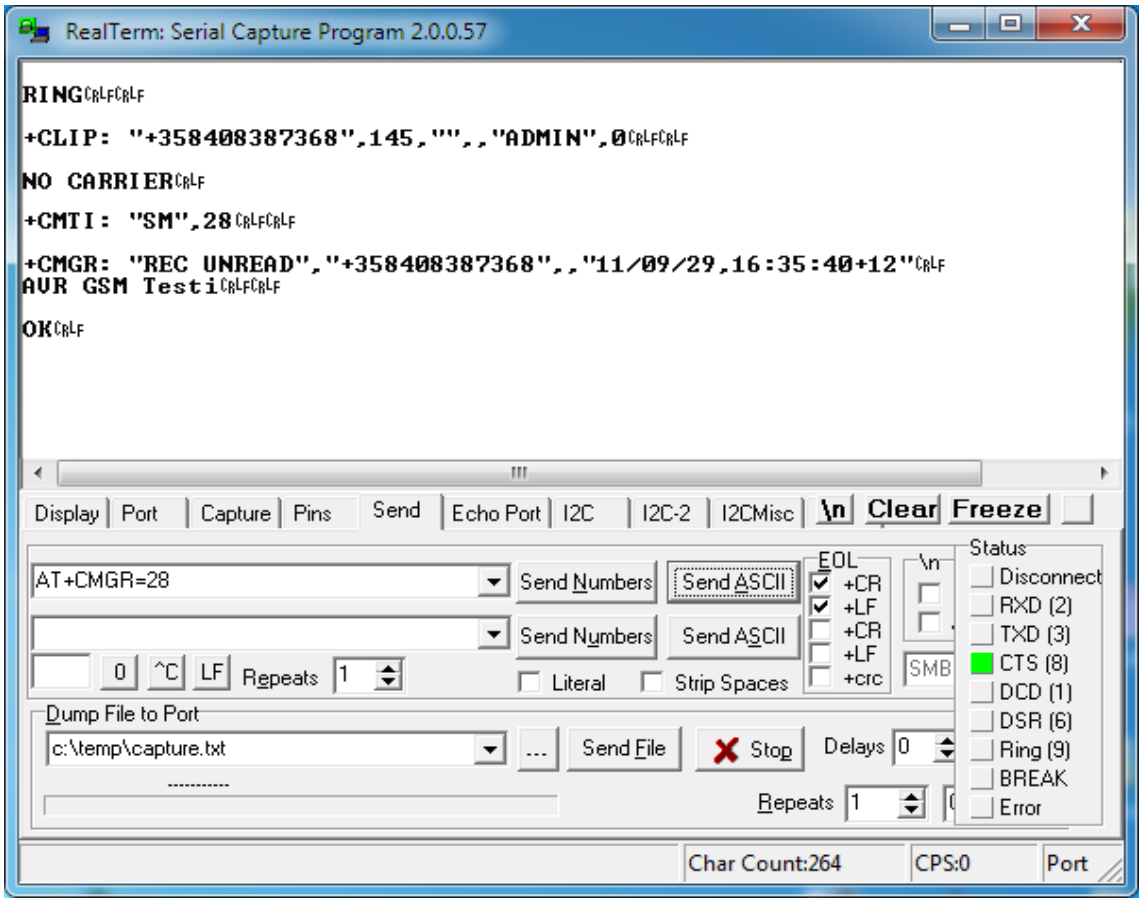
```
+CBC: 0,93,4100   // Komento ”AT+CBC” palauttaa vastauksen, jossa on kolme
                  // parametria: latauksen tila, (0=akkua ei ladata, 1=akkua
                  // ladataan), akun varaus prosentteina ja akun jännite
                  // millivoltteina.
```

Tässä tapauksessa akku ei ole latauksessa, varaus on 93 % ja jännite 4,1 V.

Moduulin ymmärtämiä komentoja on erittäin paljon. Kaikki komennot sekä niiden tarkemmat kuvaukset löytyvät lähdeluettelossa mainitusta dokumentista [3]. Seuraavissa alaluvuissa on kuitenkin esitetty vielä joitakin tärkeimpiä AT-komentoja ja moduulin paluuviestejä, joita tämän työn esimerkkisovelluksessa käytetään.

4.2.2 Puhelun vastaanotto

Puheluun vastaaminen sekä soittajan tietojen hakeminen on yksi tärkeimpiä toimintoja. Kuvassa 9 näkyy GSM-moduulin antamia viestejä, kun vastaanotetaan puhelu tai tekstiviesti.



Kuva 9. GSM-moduulin paluuviestejä vastaanotettaessa puhelu tai tekstiviesti

Moduulin antamat viestit on selitetty alla.

```
RING            // Ilmaisee, että vastaanotetaan puhelu ja soittajan päässä
                // puhelin hälyttää. Paluuviesti RING tulee jokaisella
                // hälytysäänellä. Komennolla ATA
                // voidaan vastata puheluun, jolloin yhteyden onnistuessa
                // moduuli antaa paluuviestin OK.
```

Jos soittajan puhelinnumero halutaan saada selville, on moduulin ”Calling line indentification presentation” oltava ennalta asetettu päälle. Tämä onnistuu komennolla AT+CLIP=1. Tällöin moduuli antaa puhelun tullessa alla olevan viestin mukaiset tiedot. Soittajan tiedot tulevat jokaisen RING-paluuviestin jälkeen eli jokaisella hälytyskerralla.

+CLIP: ”+358408387368”,145,””, ,”ADMIN”,0

Moduulin antama vastaus sisältää ensin tiedon +CLIP, josta tiedetään, että paluuviesti kertoo tulevan puhelun soittajan tiedot. Ensimmäisenä tietona on soittajan puhelinnumero, tämän jälkeen on numeron formaatti, tässä tapauksessa 145, joka tarkoittaa, että numero on kansainvälistä muotoa. Seuraavaksi tulee kaksi muuta parametria, joiden selitykset löytyvät lähteissä mainitusta dokumentista [3]. Viidennessä kentässä on puhelinnumeroa vastaava nimi puhelinluettelossa, jos sellainen on tallennettuna.

NO CARRIER // Ilmaisee, että puhelu on katkennut.

4.2.3 Tekstiviestin vastaanotto

Kuvassa 9 näkyvät loput paluuviestit liittyvät SMS-viestin vastaanottoon. Paluuviestien merkitykset on selitetty alla.

+CMTI: ”SM”,28 // +CMTI ilmaisee saapuneesta SMS-viestistä. ”SM” kertoo,
// että viesti on SIM-kortin muistissa ja numero 28 ilmaisee
// viestin muistipaikan, josta se voidaan lukea.

Seuraavaksi viesti luetaan komennolla AT+CMGR=<n>, jossa <n> on viestin muistipaikan numero. Luettaessa viestin sisältöä, moduuli voi palauttaa viestin joko PDU- tai tekstimuodossa. Tekstimuodossa viestin sisältö ja lähettäjän numero sekä muut tiedot tulevat helppolukuisina tekstijonoina, kun taas PDU-muodossa vastaus on vaikeampilukuisena hexakoodina. Viestin lukemisen helpottamiseksi SMS-formaatti voidaan ennalta asettaa tekstimuotoon komennolla AT+CMGF=1.

Kun moduulille lähetetään viestin lukemiskomento AT+CMGR=28, moduuli antaa seuraavan vastauksen: +CMGR: <viestin tila>, <lähettäjän numero>,<numeroa vastaava nimi puhelinluettelossa>, <vastaanotto pvm, vastaanottokellonaika>. Tämän jälkeen tulee rivinvaihto ja itse tekstiviestin sisältö. Kuvan 9 esimerkkitapauksessa moduulin antama viesti on siis:

```
+CMGR: ”REC UNREAD”,”+358408387368”, ,”11/09/29,16:35:40+12”
AVR GSM Testi
```

GMS-moduuli ilmaisee tulevasta puhelusta tai tekstiviestistä myös asettamalla RI-pinnin (Ring indicator) 0-tilaan. Tästä ei kuitenkaan voida vielä erottaa, onko kyseessä puhelu vai tekstiviesti, vaan se on luettava moduulin antamista paluuviesteistä sarjaportin kautta.

4.2.4 Tekstiviestin lähetys

Tekstiviestin lähetys on yksi keskeisimpiä toimintoja moduulia käytettäessä. Tämän työn esimerkkisovelluksessakin lähetetään käyttäjälle SMS-viestissä erilaisia tietoja. Viestin lähetys tapahtuu komennolla AT+CMGS. Kuten aiemmin on mainittu, on SMS-formaatti asetettu tekstimuotoon, jolloin viestin lähetys on helppoa. Viestin muodostaminen tapahtuu seuraavasti:

Ensin syötetään komento AT+CMGS=<vastaanottajan puhelinnumero>. Esimerkiksi:

```
AT+CMGS=”0408387368”
```

Tähän moduuli vastaa merkillä ”>”, joka tarkoittaa, että moduuli odottaa itse tesktiviestin sisällön syöttämistä. Kaikki seuraavat syötetyt merkit ovat viestin sisältöä, kunnes moduulille syötetään viestin loppua ilmaiseva ASCII-taulukon mukainen merkki SUB, joka on desimaalimuodossa 26. Terminaaliohjelmaa käytettäessä tämä voidaan tehdä myös näppäinyhdistelmällä CTRL+Z. Suoraan tämän jälkeen viesti lähetetään eteenpäin.

5 ESIMERKKISOVELLUS

Tässä luvussa on esitetty AVR-GSM-kehitysalustalle luotu esimerkkisovellus, jonka tarkoituksena on havainnollistaa GMS-moduulin ohjausta. Sovellus keskittyy lähinnä GSM-moduulin ja ATMega32-mikrokontrollerin väliseen kommunikointiin sarjaliikenteen välityksellä sekä käyttäjän matkapuhelimen ja GSM-moduulin välille luotavan SMS-viestiyhteyden muodostamiseen. Sovellukseen luodut ominaisuudet on tehty kehitysalustan mahdollisuuksien mukaan. Pidemmälle viety käytännön sovellus on helppoa toteuttaa tämän esimerkin pohjalta.

5.1 Sovelluksen toiminnot ja sen käyttö

Tämän esimerkkisovelluksen käyttöönotossa on kehitysalustaan asetettava SIM-kortti, jonka PIN-koodin kysely on poistettu käytöstä. Tämä onnistuu helpoiten asettamalla SIM-kortti tavalliseen matkapuhelimeen, jonka avulla koodin kysely asetetaan pois päältä. Tässä sovelluksessa on käytetty kuukausimaksutonta prepaid-liittymää, johon on helppo ladata puheaikaa internetistä. Lisäksi SIM-kortille tallennetaan pääkäyttäjän puhelinnumero nimellä ADMIN sekä mahdollisten muiden käyttäjien numerot, joiden nimien alku on USER. Esimerkiksi USER1, USER2 ja niin edelleen. Kun SIM-kortti on asetettu, voidaan moduulille kytkeä käyttöjännite. Tämä tapahtuu kytkemällä jumpperi BAT_E, jolloin moduuli saa virran akulta. Ruuviliittimeen voi myös kytkeä ulkoisen käyttöjänitteen 12 V, jolloin akku latautuu sekä releet ova käytössä. GSM-moduuli lähtee käyntiin painamalla PWRKEY-painiketta noin kolmen sekunnin ajan.

Tähän esimerkkisovellukseen on luotu seuraavat ominaisuudet GSM-moduulin ohjauksen ja kehitysalustan ominaisuuksien testaamista ja havainnollistamista varten:

1. Puheluun vastaaminen automaattisesti, mikäli soitto tulee pääkäyttäjän numerosta sekä audioinputin ja -outputin kytkeytyminen kortin handsfree-liitäntään.
2. Automaattinen puheluiden hylkääminen, mikäli puhelu on muusta kuin pääkäyttäjän numerosta.

3. Erilaisten komentojen vastaanotto SMS-viesteillä ja komennon perusteella suoritettava toiminto kortilla. Komentoja voi lähettää pääkäyttäjä ja muut käyttäjät, joiden numerot on lisätty SIM-kortille.
4. Käyttäjälle paluuviestinä lähetettävä SMS-viesti, joka sisältää erilaisia tietoja sovelluksen tilasta.

Toiminnon 3 ymmärtämät komennot ja niitä vastaavat toiminnot on lueteltu seuraavassa:

Komento	Toiminto
Rele1 On	Asettaa releen 1, joka on mikrokontrollerin pinnissä PC7.
Rele1 Off	Nollaa releen 1.

Vastaavat toiminnot releelle 2 komennoilla *Rele2 On* ja *Rele2 Off*.

Tila	Hakee releiden tilat sekä kortilla olevan akun varauksen tilan.

Jos moduulille lähetetään SMS-viesti, joka sisältää edellisistä komennoista poikkeavan viestin, saa käyttäjä paluuviestinä ilmoituksen ”Tuntematon komento”.
Tila-komennolla erikseen haettava tai muiden komentojen seurauksena automaattisesti paluuna tuleva SMS-viesti on esimerkiksi seuraavanlainen:

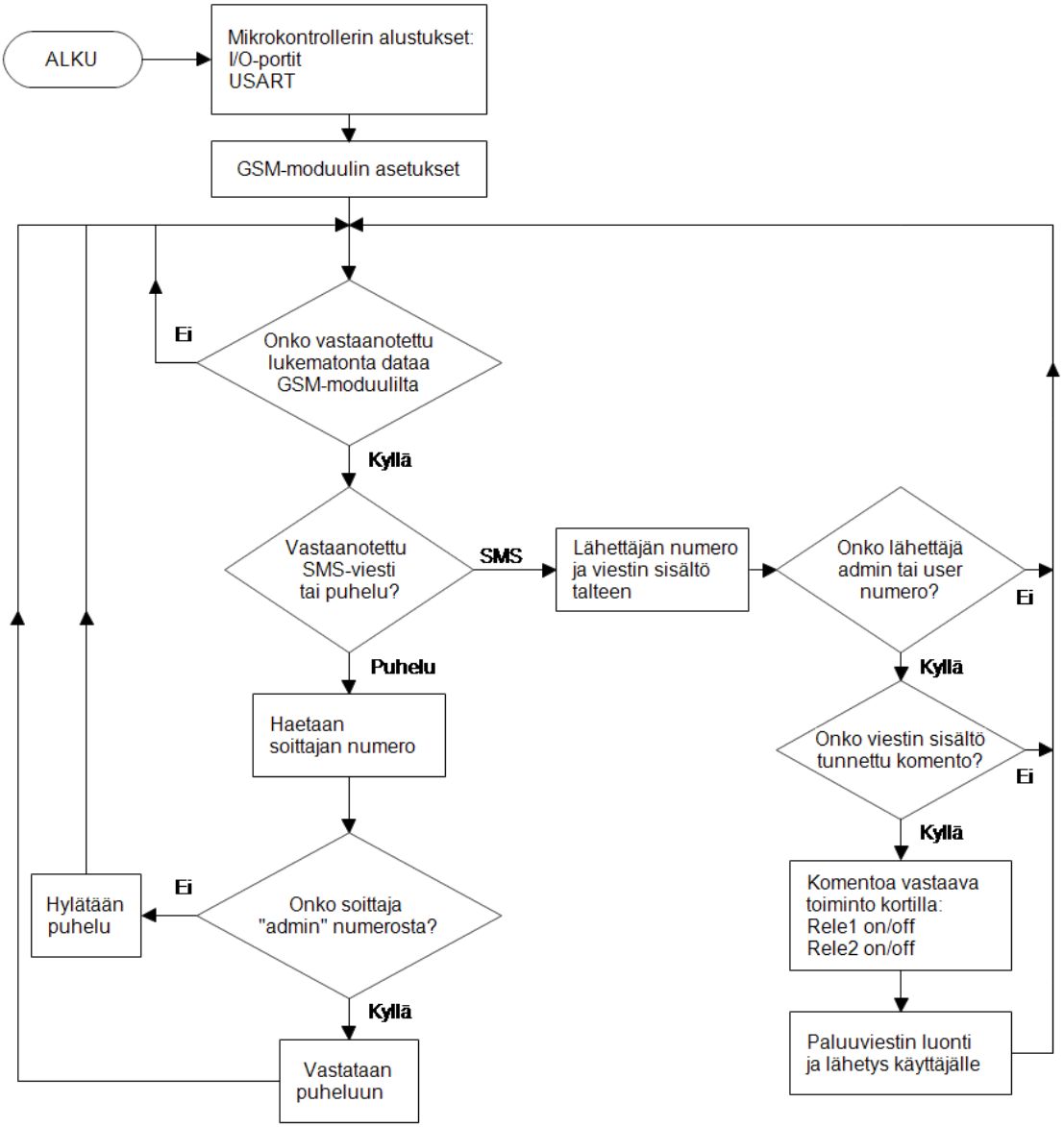
```
RELAY1=1
RELAY2=0

*Battery info*
Charge mode: 1
Capacity: 96%
Voltage: 4150mV
```

Releiden perään ei tässä sovelluksessa ole kytketty mitään, mutta niihin olisi helppo kytkeä esimerkiksi jokin laite, joka halutaan kytkeä päälle/pois. Samoin mikrokontrollerin muita I/O-linjoja voisi ohjata lisäämällä sovelluksen lähdekoodiin komento ja luoda sitä vastaavan pinnin asetus/nollaus. Myös erilaisten anturien lisääminen kytkentään voisi olla eräs pidemmälle viety käytännön sovellus, jossa tietoja haetaan ja vastaanotetaan SMS-viestiyhteyden välityksellä.

5.2 Sovelluksen lähdekoodi

Tässä luvussa käsitellään sovelluksen lähdekoodin tärkeimpiä toimintoja ja perusperiaatteita. Koko lähdekoodin listaus kommentointeineen on luettavissa liitteistä. Koodi on C-kieltä ja se on kirjoitettu käyttäen AVR Studio 4-kehitysympäristöä.

Ohjelman toimintaa kuvaava kaavio on esitetty seuraavassa:



Kuva 10. Esimerkkisovelluksen toiminta

Kuvassa 10 esitettyjen lohkojen lähdekoodin toimintaa kuvataan seuraavissa alaluvuissa.

5.3 Alustukset

Tarvittavat alustukset kuten I/O-porttien ja USART:n alustus suoritetaan funktiossa Init(), joka ajetaan pääohjelman alussa. Sarjaliikenteen asetuksiksi asetetaan 8 databittiä, 1 stop-bitti, ei pariteettia sekä nopeudeksi 115200 bit/s.

```
/** IO-porttien ja USART alustus **/
void Init()
{
PORTA=0x00;
DDRA=0x00;   //PORTA input
PORTB=0x00;
DDRB=0x00;   //PORTB input
PORTC=0x00;
DDRC=0xC0;   //PORTC 7,6 (Relay1, Relay2) output, 5-0 input
PORTD=0x00;
DDRD=0x92;   //PORTD 7,4,1 (Led, Buzz, TX) output 6,5,3,2 input

/** UART init **/
/** 115200baud, 8bit, 1stop bit, no parity **/
UCSRA=0x00;
UCSRB |= (1 << RXCIE) | (1 << RXEN) | (1 << TXEN);
UCSRC |= (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1);
UBRRH = (BAUD_PRESCALE >> 8);
UBRRL = BAUD_PRESCALE;
rx_counter = wr_index = rd_index = 0;

PORTB &=0b11111011;   //Pidä FT232RL resetissä.
DDRB |= 0x04;

Send_UART("AT+CPOWD=1");      //Moduuli power down
asm volatile("sei");          //Sallitaan keskeytykset.
}
```

USART:n alustukseen tarvittavat vakiot on asetettu seuraavasti:

```
#define F_CPU 7372800UL
#define USART_BAUDRATE 115200
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)
```

5.3.1 USART-lähetys ja -vastaanotto

Sarjaliikenteen lähetys tapahtuu aliohjelmalla Send_UART(), jolle annetaan parametrina lähetettävä merkkijono. Tämä aliohjelma lähettää merkin kerrallaan eteenpäin putchr()-funktiolla aina, kun mikrokontrollerin UDR-rekisteriin voidaan kirjoittaa. Jokaisen merkkijonon perään lähetetään ”carriage return” CR, jonka GSM-moduuli tulkitsee komennon lopuksi.

```
/** USART Tx, yhden merkin lähetys **/
static void putchr(char c)
{
  while((UCSRA & (1 << UDRE)) == 0){};  //Odotetaan kunnes UDR rekisteriin voidaan
  UDR = c;                              //kirjoittaa.
}
/** USART merkkijonon lähetys **/
void Send_UART(unsigned char *byte)
{
  for(;*byte;++byte){
    putchr(*byte);
  }
  putchr(13);               //Merkkijonon loppuun CR.
}

/*** USART Rx, Rx-keskeytyspalvelu. ***/
ISR(USART_RXC_vect)
{
  char data = UDR;                      //Luetaan vastaanotettu merkki

  if((UCSRA & (FE | PE | DOR)) == 0){   //Frame error, parity error, data overrun
    rx_buffer[rx_wr_index] = data;      //lisätään merkki char taulukkoon rx_buffer

    if(++rx_wr_index == RX_BUFFER_SIZE) //Kasvatetaan indeksiä, jos se on sama kuin
      rx_wr_index = 0;                  //bufferin koko nollataan index.

    if(++rx_counter == RX_BUFFER_SIZE){ //Kasvatetaan rx_counteria.
      rx_counter = 0;                   //rx_counter pitää kirjaa onko lukemattomia
      rx_buffer_overflow = 1;           //merkkejä bufferissa.
    }
  }
}
```

Moduulin lähettämää dataa vastaanotetaan keskeytyspalvelussa, jossa se tallennetaan taulukkoon rx_buffer[] myöhempää käsittelyä varten. Kaikki moduulin lähettämä data siis tallentuu suoraan puskuriin, jota voidaan myöhemmin lukea merkki kerallaan.

Merkin hakeminen puskurista tapahtuu funktiossa getchar_UART(), jolle annetaan parametrina aika, jonka verran funktio odottaa merkkejä, jos yhtään merkkiä ei ole vielä vastaanotettu. Funktio palauttaa puskurista seuraavan lukemattoman merkin. Apumuuttujat pitävät kirjaa, missä kohtaa puskuria ollaan lukemassa. Seuraavalla sivulla on esitetty funktion getchar_UART() koodi.

Yhden merkin hakeminen puskurista:

```
char getchar_UART(unsigned long t)
{
  char data;
  while(t > 0){               //Pyöritään annettun timeout ajan verran odottelemassa.
    asm volatile("cli");      //Jos rx_counter on eri kuin 0 on bufferissa
    if(rx_counter != 0) break;//lukemattomia merkkejä.
    asm volatile("sei");
    t--;
  }
  asm volatile("sei");
  if(t == 0){        //Jos timeout pääsee nollaan, ei ole vastaanotettu mitään.
    timeout = 1;     //Poistutaan aliohjelmasta palauttaen 0xFF
    return 0xFF;
  }
  data = rx_buffer[rx_rd_index];     //Luetaan merkki rx_bufferista.
  if(++rx_rd_index == RX_BUFFER_SIZE)
    rx_rd_index = 0;

  asm volatile("cli");          //Vähennetään rx_counteria koska on luettu merkki.
  --rx_counter;
  asm volatile("sei");
  return data;       //Palautetaan luettu merkki.
}
```

5.3.2 Main-silmukka

Ohjelman pääsilmukka on alla esitetyn mukainen. Pääsilmukassa alustetaan funktiossa Init() mikrokontrollerin I/O-portit sekä USART. Ohjelman loputtomassa silmukassa tutkitaan, onko vastaanotettu dataa GSM-moduulilta ja tiedetäänkö, ollaanko vastaanotettu SMS-viesti tai puhelu. Tämän perusteella ajetaan aliohjelma checkSMS() tai checkCaller().

```
/*** PÄÄOHJELMA ***/
int main(void)
{
  Init();    // Alustetaan I/O-portit sekä sarjaportti
  while(checkResponse(callready)==0){
    merkki = getchar_UART(100000);
  }
  /** Moduulin alkuasetukset **/
  Send_UART("ATE0");           //Merkkien kaiutus pois päältä.
  Send_UART("AT+CMGF=1");      //Asetaan SMS-viestille tekstimuoto.
  Send_UART("AT+CLIP=1");      //Soittajan numero näkyviin.
while(1){
  int i;
  for(i=0; i<64; i++){
    temp_array[i] = 0;}
  if(rx_counter != 0){               //Onko rx-bufferissa lukemattomia merkkejä.
    merkki = getchar_UART(100000);   //Haetaan merkki rx-bufferista.
    if(merkki == '+'){               //Jos ensimmäinen merkki oli '+',
      checkSMS();                    //tutkitaan vataanotettu viesti. (+CMTI....)
    }
    else{
      if(checkResponse(ring))        //Muutoin tutkitaan onko puhelu tulossa. "RING"
        checkCaller();
  }}}}
```

5.3.3 Puhelun vastaanoton koodi

Mikäli pääohjelmasilmukassa havaitaan moduulin lähettämän datan perusteella, että vastaanotetaan puhelua, ajetaan aliohjelma checkCaller(), jonka koodi on esitetty alla. Moduuli on alkuasetuksissa asetettu siten, että se ilmoittaa automaattisesti soiton tulessa paluuviestissä soittajan tiedot. Puhelun vastaanottoon liittyviä moduulin paluuviestejä on käsitelty luvussa 4.2.2. Alla esitetty koodi hakee soittajan numeroa vastaavan nimen, jos sellainen on tallennettu puhelinluetteloon. Jos soittajan numeroa vastaava nimi löytyy luettelosta nimellä ADMIN tai USER1, USER2 ja niin edelleen, puheluun vastataan. Muista numeroista tulevat puhelut katkaistaan.

```
void checkCaller(void){
  int colcount = 0;
  int i = 0;

  do{
    merkki = getchar_UART(100000);
    if(merkki == '"')                 //"-merkin perusteella siirrytään oikeaan
      colcount++;                     //tietokenttään viestissä
    if(colcount == 5 && merkki!='"') //Jos oli viides "-merkki ollaan soittajan nimi-
      temp_array[i++] = merkki;       //kentässä. Otetaan soittajan nimi talteen.
  }while(merkki != 10 || colcount <=5);

  if(Compare(admin, sizeof(admin)) ||
    Compare(user,  sizeof(user)))      //Jos soittajan numero vastaa tallennettua ADMIN
    Send_UART("ATA");                  //tai USER numeroa, vastataan puheluun
  else
    Send_UART("ATH");                  //muutoin katkaistaan puhelu
}
```

5.3.4 Tekstiviestin vastaanoton koodi

Jos pääohjelmasilmukassa havaitaan moduulin lähettämän datan perusteella, että ollaan mahdollisesti vastaanotettu SMS-viesti, suoritetaan aliohjelma checkSMS(). Tämä aliohjelma tutkii, onko moduulin paluuviesti ilmoitus saapuneesta SMS-viestistä. Aliohjelma hakee myös paluuviestistä SMS-viestin muistipaikan, syöttää moduulille tekstiviestin lukemiskomennon ja hakee viestin sisällön talteen. Seuraavaksi aliohjelma tutkii, onko viestin lähettäjän numero puhelinluettelossa nimellä ADMIN tai USER ja onko viestin sisältö hyväksyttävä komento. Jos komento on tunnettu, aliohjelma suorittaa komentoa vastaavan toiminnon (releiden ohjaus). Moduulin antamista paluuviesteistä SMS-viestin saapumisen yhteydessä on kerrottu tarkemin luvussa 4.2.3.

Aliohjelman checkSMS() oleellisimpia kohtia GSM-moduulin ohjauksen kannalta on esitetty seuraavassa. Koko aliohjelman listaus löytyy liitteistä. Aliohjelma sisältää useita do-while-silmukoita, joissa moduulin antamista paluuviesteistä haetaan haluttuja tietoja.

Alla esitetyssä ensimmäisessä do-while-silmukassa ohjelma hakee paluuviestin merkkejä taulukkoon temp_array[], ja jos merkki on numero, se asetetaan myös taulukkoon smsind[]. Taulukkoon smsind[] on tarkoitus hakea SMS-viestin muistipaikkaa ilmaiseva numero. Kun vastaanotetaan ”line feed”, tiedetään, että koko paluuviesti on käyty läpi. Seuraavaksi ohjelma muodostaa valmiiksi AT-komennon, jolla SMS-viestin sisältö voidaan lukea.

```
do{
  merkki = getchar_UART(100000);     //Haetaan seuraavia merkkejä rx-bufferista.
  if(merkki != 13 && merkki != 10){  //Merkkejä taulukkoon mutta hylätään CR ja LF.
    temp_array[i++] = merkki;
    if(isdigit(merkki))              //Jos bufferista haettu merkki on numero,
      smsind[j++] = merkki;          //se asetetaan smsind taulukkoon
  }
}while(merkki != 10);                //Loopissa pyöritään kunnes vastaanotetaan LF.

smsind[j++] = 0;
ch = atoi(smsind);                   //atoi() muuntaa merkkijonon numeroksi.
sprintf(atCommand,"AT+CMGR=%d",ch); //Asetetaan taulukkoon viestin lukemis komento.
```

Ensimmäisessä do-while-silmukassa talteen otettua paluuviestiä verrataan ennalta tiedettyyn merkkijonoon, joka on ilmoitus saapuneesta SMS-viestistä. Jos ehto on tosi, lähetetään moduulille SMS-viestin lukemiskomento. Nyt moduuli palauttaa lähettäjän tiedot ja itse tekstiviestin sisällön. Seuraavassa do-while-silmukassa haetaan lähettäjän numero ja numeroa vastaava nimi talteen, jotta voidaan tutkia, onko viesti numerosta, josta SMS-viestin välityksellä lähetetyt komennot sallitaan. Numero otetaan talteen myös, jotta käyttäjälle voidaan lähettää paluuna raporttiviesti sovelluksen tilasta.

```
if(Compare(sms_rec, sizeof(sms_rec))){ //Onko paluuviesti, ilmoitus saapuneesta SMS-
  Send_UART(atCommand);                //viestistä. Jos on, lähetetään moduulille SMS
  while(merkki != '+')
  merkki = getchar_UART(100000);       //Odotetaan moduulin vastausta.
  i=j=colcount=0;                      //Apumuuttujien nollaus
  do{
    merkki = getchar_UART(100000);  //Haetaan merkkejä bufferista.
    if(merkki == '"')               //siirrytään oikeaan tietokenttään.
      colcount++;
    if(colcount == 3){              //Kolmas "-merkki, ollaan puhnro kentässä.
      if(isdigit(merkki))           //Numerot asetetaan sms_number taulukkoon.
        sms_number[i++] = merkki;
    }
    if(colcount==5 && merkki!='"'){ //Viides "-merkki ollaan nimi-kentässä.
    temp_array[j++] = merkki;       //Numeroa vastaava nimi talteen.
    }
  }while(merkki != 10);             //Loopissa pyöritään kunnes vastaanotetaan LF
```

Edellisen sivun do-while-silmukassa on haettu siis talteen lähettäjän puhelinnumero ja sitä vastaava nimi. Kun vastaanotetaan rivinvaihto tiedetään, että seuraavat merkit ovat itse tekstiviestin sisältöä.

Alla esitetyssä koodissa lähettäjän nimeä verrataan sallittuihin numeroihin ja asetetaan apumuuttuja sen mukaisesti. Aliohjelman checkSMS() viimeisessä do-while-silmukassa haetaan itse viestin sisältö taulukkoon temp_array[]. Kun viestin sisältö on haettu talteen, voidaan viesti poistaa SIM-kortin muistista samantien, jottei muisti pääse milloinkaan täyttymään. Viimeisenä talteen haettua viestin sisältöä verrataan tunnettuihin komentoihin ja suoritetaan toiminto sen perusteella.

```
if(Compare(admin, sizeof(admin))||
  Compare(user, sizeof(user)))  //Jos lähettäjä vastaa tallennettua admin/user numeroa,
  pass = 1;                     //sallitaan releiden ohjaus
else
  pass = 0;

i=0;
do{
  merkki = getchar_UART(100000);     //Haetaan merkkejä bufferista.
  if(merkki != 13 && merkki != 10)
    temp_array[i++] = merkki;
}while(merkki != 10);

sprintf(atCommand,"AT+CMGD=%d",ch); //Luodaan viestin poistokomento.
Send_UART(atCommand);               //Halutut tiedot on tallessa poistetaan sms-viesti.

if(pass == 1){
  if(Compare(relay1on,              //Jos sisältö vastaa komentoa,
    sizeof(relay1on))){             //toimitaan sen mukaan.
    bit_set(PORTC, BIT(7));         //aseta Rele1
  }
```

5.3.5 Tekstiviestin lähetyksen koodi

Ohjelmassa käyttäjälle lähetetään SMS-viestinä releiden tilat sekä akun varaus ja jännite, kun käyttäjä lähettää releitä ohjaavan komennon tai ”Tila”-komennon. Viestin lähetys tapahtuu sendSMS()-aliohjelmassa. Aluksi aliohjelmassa muodostetaan AT-komento, jolla viestin kirjoitus aloitetaan. Tämä tapahtuu yhdistämällä viestin lähetyskomennon etuliite ja puhelinnumero, johon viesti lähetetään. Puhelinnumero on tallennettuna taulukkoon sms_number[], johon se on haettu, kun edellistä saapunutta SMS-viestiä on käsitelty. AT-komento lähetetään GSM-moduulille ja ohjelma jää odottamaan moduulin vastausta ”>”, jonka perään syötetään itse viesti.

Lähetettävä viesti on tallennettu taulukkoon message[]. Viesti, jonka message[]-taulukko sisältää, on muodostettu getStatus()-aliohjelmassa. Lopuksi jäädään vielä odottamaan moduulin vastausta onnistuneesta viestin lähetyksestä, jonka jälkeen voidaan siirtyä ohjelmassa eteenpäin. Viestin lähetykseen liittyviä AT-komentoja on käsitelty luvussa 4.2.4.

```
void sendSMS(){

  int k = sizeof(sms_send);              //SMS lähetyskomennon pituus

  memcpy(sms_send_number, sms_send, k); //Kopioidaan SMS lähetyskomennon alku
  memcpy(sms_send_number + k,            //Kopioidaan edellisen perään puh. nro.
  sms_number, (strlen(sms_number)));     //Numeron pituus talteen.
  k = k + strlen(sms_number);            //Siirretään indeksiä oikeaan kohtaan.
  sms_send_number[k++]='"';
  sms_send_number[k]='\0';

  Send_UART(sms_send_number);            //Viestin lähetyskomento moduulille.

  while(!(merkki == '>'))                //Odotetaan '>'-merkkiä.
    merkki=getchar_UART(100000);

  Send_UART(message);                    //Syötetään haluttu tekstiviesti.
  putchr(26);                            //Viestin kirjoitus lopetetaan SUB-merkillä.

  while(merkki != '+')
  merkki = getchar_UART(100000);         //Moduuli vastaa +CMGS: x
}
```

Esimerkkisovellus sisältää lisäksi muutaman muun aliohjelman, joita käytetään ohjelman suorituksen aikana. Näiden aliohjelmien toimintaa ei kuvata tarkemmin, mutta niiden tarkoitus on selitetty lyhyesti alla. Koko sovelluksen lähdekoodin listaus on luettavissa liitteistä.

Taulukko 3. Esimerkkisovelluksen muut funktiot.

Aliohjelma	Toiminto
`int checkResponse(char*)`	Hakee parametrina annettua merkkijonoa moduulin paluuviesteistä. Esimerkiksi ”Call Ready” tai ”RING”.
`unsigned char Compare(char*,int)`	Vertaa SMS-viestistä haettua tekstiä ennalta määrättyihin komentoihin.
`void checkStatus(void)`	Muodostaa message[]-taulukon sisälle sovelluksen tilasta kertovan viestin, joka lähetetään paluuna käyttäjälle aliohjelmassa sendSMS().

5.4 Ohjelmiston testaus

Ohjelmiston testaus on melko hankalaa, sillä jokaisen järkevän testitilanteen luomiseksi, on moduulille lähetettävä tekstiviesti tai soitettava puhelu. Myös eri numeroista samaan aikaan tehtäviä testitilanteita olisi hyvä olla. Joitakin tilanteita voi testata RealTerm-terminaaliohjelmaa käyttäen ja tutkien moduulin antamia paluuviestejä.

GSM-moduulin ja mikrokontrollerin välinen keskustelu sarjaliikenteen välillä voidaan tallentaa omaan taulukkoon lisäämällä siihen jokainen vastaanotettu ja jokainen lähetetty merkki. Tämä informaatio voidaan sitten lähettää vaikka viestin sisällä testaajalle, jotta voidaan havaita, että liikennöinti tapahtuu juuri siten, kuten on haluttu.

6 YHTEENVETO

Työn tavoitteena oli perehtyä GSM-moduulin käyttöön ja ohjaukseen. Käytetty AVR-GSM-kehitysalusta on erinomainen työkalu matkapuhelinverkkoa käyttävien järjestelmien kehittämiseen ja GSM-moduulin ohjaukseen tutustumiseen. Moduulin ohjaus vaatii hieman tutustumista AT-komentojen käyttöön sekä perustason laitteistoläheisen ohjelmoinnin hallitsemista, jonka jälkeen mitä erilaisempien sovellusten kehitys on mahdollista.

Työn tuloksena saatiin toimiva esimerkkisovellus, joka havainnollistaa GSM-moduulin tärkeimpien toimintojen ohjelmointia. SMS-viestiyhteyden luominen sekä puheluun vastaaminen ja käyttäjän numeron tarkistaminen olivat eräitä luotuja ominaisuuksia. Usean testin ja kattavan C-koodin tutkimisen jälkeen ohjelma voitiin todeta varmatoimiseksi. Tämän työn pohjalta on helppoa lähteä kehittämään pidemmälle vietyjä käytönnön tarkoituksiin tehtyjä sovelluksia.

LÄHTEET

[1] Olimex AVR-GSM User manual
http://www.olimex.com/dev/pdf/AVR/AVR-GSM/AVR-GSM.pdf
(17.11.2011)

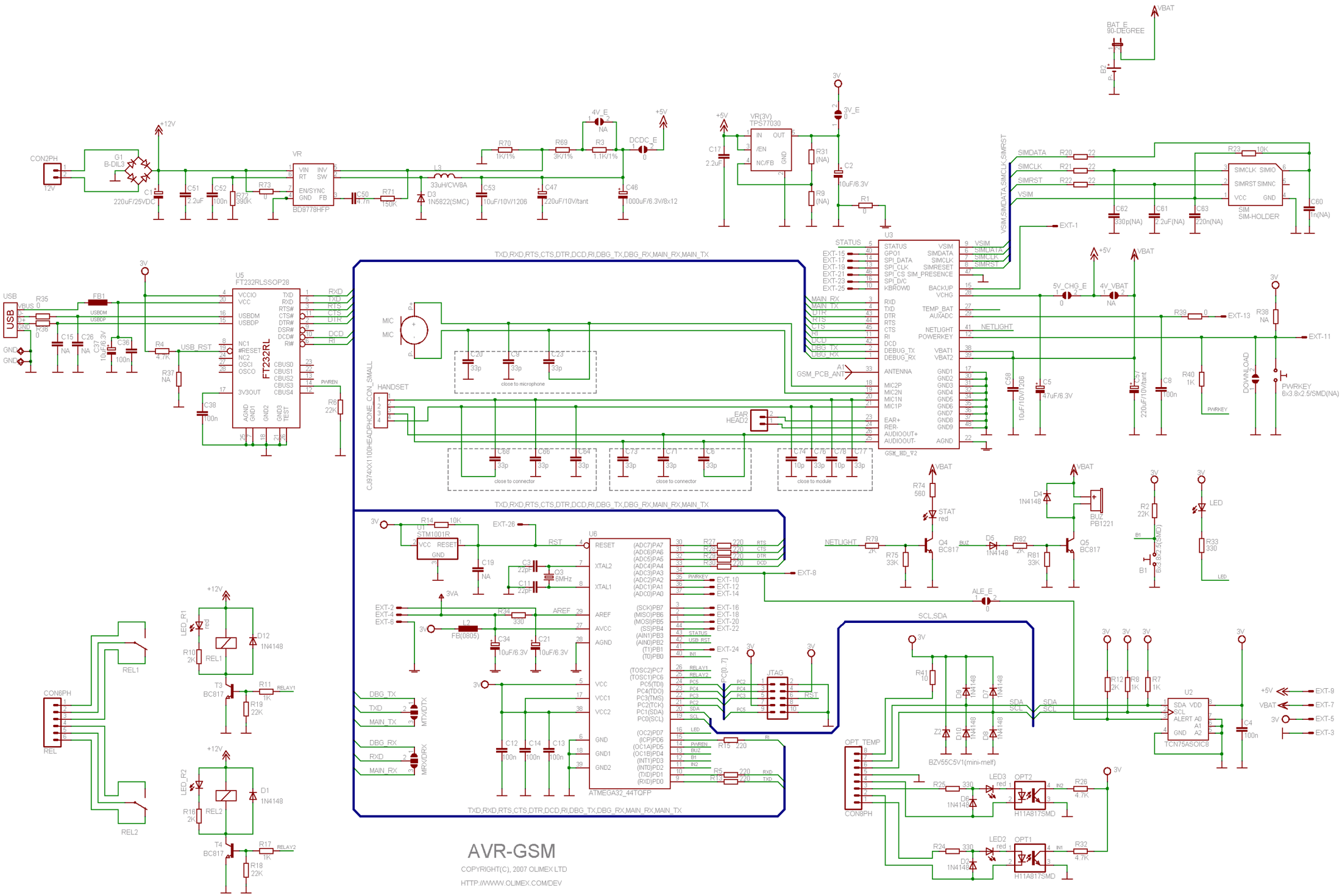
[2] SIM300D Hardware Specification
http://mdfly.com/Download/Wireless/sim300D_HD_V2.01.pdf
(17.11.2011)

[3] SIM300D AT Command set
http://www.olimex.com/dev/pdf/AVR/AVR-GSM/SIM300DATC.pdf
(17.11.2011)

[4] ATMega32 Datasheet
http://atmel.com/dyn/resources/prod_documents/doc2503.pdf
(17.11.2011)

LIITTEET

AVR-GSM-KEHITYSALUSTAN KYTKENTÄKAAVIO [1, s. 8] LIITE 1



GSM-MODUULIN SIIRTYMINEN ERI TILOJEN VÄLILLÄ. [2, s. 28] LIITE 2

Further mode Current mode	POWER DOWN	Normal mode	Ghost mode (Charge-only mode)	Charging in normal	Alarm mode
POWER DOWN		Use PWRKEY	Connect charger to VCHG and connect battery to VBAT	No direct transition, but via "Ghost mode" or "Normal mode"	Switch on from POWER DOWN mode bye RTC
Normal mode	AT+CPOWD or use PWRKEY pin		Connect charger to VCHG and connect battery to VBAT, then switch off module by AT+CPOWD or using PWRKEY	Connect charger to VCHG pin of module and connect battery to VBAT pin of module	Set alarm by "AT+CALARM", and then switch off the module. When the timer expire, the module turn on and enter Alarm mode
Ghost mode (Charge-only mode)	Disconnect charger	No direct transition, but via "Charging in normal" mode		Turn on the module using PWRKEY OR SET AT Command "AT+CFUN=1"	Set alarm by "AT+CALARM", when the timer expire, module will enter Alarm mode
Charging in normal	AT+CPOWD → "Ghost mode", then disconnect charger	Disconnect the charger	Switch off module by AT+CPOWD or using PWRKEY		No direct transition
Alarm mode	Use PWRKEY pin or wait module switch off automatically	Use AT+CFUN	No transition	Use AT+CFUN let module enter Normal mode, then connect the charger to VCHG pin of module	

ESIMERKKISOVELLUKSEN LÄHDEKOODI LIITE 3: 1 (8)

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

//Makrot bittien käsittelyyn.
#define bit_set(p,m) ((p) |= (m))
#define bit_clear(p,m) ((p) &= ~(m))
#define bit_get(p,m) ((p) & (m))
#define BIT(x) (0x01 << (x))

//Vakioiden määrittelyt
#define F_CPU 7372800UL
#define USART_BAUDRATE 115200
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)
#define RX_BUFFER_SIZE 256

//Käytettyjen AT-komentojen alkuosat ja verrattavat merkkijonot.
char sms_read[] = "AT+CMGR=";
char sms_rec[]  = "CMTI: \"SM";
char sms_send[] = "AT+CMGS=\"+";
char callready[]= "Call Ready";
char ring[]  = "RING";
char admin[] = "ADMIN";
char user[]  = "USER";

//Komennot joita tekstiviesteistä luetaan.
char relay1on[]  = "RELE1 ON";
char relay1off[] = "RELE1 OFF";
char relay2on[]  = "RELE2 ON";
char relay2off[] = "RELE2 OFF";
char status[]   = "TILA";

//Globaalit muuttujat
unsigned long a;
unsigned char merkki;
unsigned char temp_array [64];
char sms_send_number[30];
char message[160];
char smsind[6];
char sms_number[16];
char atCommand[13];
int timeout=0;
char rx_buffer[RX_BUFFER_SIZE];
int wr_index, rd_index;
volatile int rx_counter;
unsigned char rx_buffer_overflow;

/** Funktioiden esittelyt **/
void checkSMS(void);
void checkCaller(void);
void checkStatus(void);
void sendSMS(void);
void Send_UART(unsigned char *);
void Delay (unsigned long a) { while (--a!=0);}
void delay_ms(uint16_t ms){while(ms--) _delay_ms(1.0);}
```

LIITE 3: 2 (8)

```
/** IO-porttien ja USART alustus **/
void Init()
{
PORTA=0x00;
DDRA=0x00;   //PORTA input
PORTB=0x00;
DDRB=0x00;   //PORTB input
PORTC=0x00;
DDRC=0xC0;   //PORTC 7,6 (Relay1, Relay2) output, 5-0 input
PORTD=0x00;
DDRD=0x92;   //PORTD 7,4,1 (Led, Buzz, TX) output 6,5,3,2 input

/** UART init **/
/** 115200baud, 8bit, 1stop bit, no parity **/
UCSRA=0x00;
UCSRB |= (1 << RXCIE) | (1 << RXEN) | (1 << TXEN);
UCSRC |= (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1);
UBRRH = (BAUD_PRESCALE >> 8);
UBRRL = BAUD_PRESCALE;
rx_counter = wr_index = rd_index = 0;

PORTB &=0b11111011;   //Pidä FT232RL resetissä.
DDRB |= 0x04;

Send_UART("AT+CPOWD=1");      //Moduuli power down
asm volatile("sei");          //Sallitaan keskeytykset.
}


/** USART Tx, yhden merkin lähetys **/
static void putchr(char c)
{
  while((UCSRA & (1 << UDRE)) == 0){};  //Odotetaan kunnes UDR rekisteriin
  UDR = c;                              // voidaan kirjoittaa.
}
/** USART merkkijonon lähetys **/
void Send_UART(unsigned char *byte)
{
  for(;*byte;++byte){
    putchr(*byte);
  }
  putchr(13);                           //Merkkijonon loppuun CR.
}


/*** USART Rx, Rx-keskeytyspalvelu. ***/
ISR(USART_RXC_vect)
{
  char data = UDR;                      //Luetaan vastaanotettu merkki.

  if((UCSRA & (FE | PE | DOR)) == 0){   //Frame error, parity error, data overrun
tarkistus.
    rx_buffer[wr_index] = data;         //Lisätään merkki rx-bufferiin.
    if(++wr_index == RX_BUFFER_SIZE)
      wr_index = 0;

    if(++rx_counter == RX_BUFFER_SIZE){ //Kasvatetaan rx_counteria.
      rx_counter = 0;
      rx_buffer_overflow = 1;
    }
  }
}
```

LIITE 3: 3 (8)

```
/** Haetaan merkki vastaanotetusta datasta rx_buffer[] taulukosta   **/
/** Parametrina annetaan timeout aika jonka verran odotellaan merkkejä **/
char getchar_UART(unsigned long t)
{
  char data;

  while(t > 0){           //Pyöritään annettun timeout ajan verran odottelemassa.
    asm volatile("cli");  //Jos rx_counter on eri kuin 0, on bufferissa
    if(rx_counter != 0)   //lukemattomia merkkejä.
      break;
    asm volatile("sei");
    t--;
  }
  asm volatile("sei");

  if(t == 0){             //Jos timeout pääsee nollaan, ei ole vastaanotettu mitään.
    timeout = 1;          //Poistutaan aliohjelmasta palauttaen 0xFF
    return 0xFF;
  }

  data = rx_buffer[rd_index];     //Luetaan merkki rx_bufferista.
  if(++rd_index == RX_BUFFER_SIZE)
    rd_index = 0;

  asm volatile("cli");            //Vähennetään rx_counteria koska on luettu merkki.
  --rx_counter;
  asm volatile("sei");

  return data;                    //Palautetaan luettu merkki.
}


/** Aliohjelma joka etsii parametrina annettua**/
/** merkkijonoa moduulin paluuviesteistä.    **/
int checkResponse(char* s){

  int i;

  for(i=0; s[i] != '\0'
    && s[i] == merkki; i++)
  {
    merkki = getchar_UART(10000);
  }
  return ((i == strlen(s)) ? 1 : 0);  //Jos jokainen merkki täsmäsi palautetaan 1.

}


/** Aliohjelma vertaa moduulin antamaa viestiä ennalta tiedettyihin komentoihin. **/
/** Parametreina annetaan ennalta määrätty komento jota verrataan sekä sen pituus **/
unsigned char Compare(char* s, int l){

  unsigned char a, b=0;
  int length = l;

  for(a=0; a != length; a++){           //Verrataan temp-taulukkoon luettua komentoa.
    if(toupper(temp_array[b++]) != s[a])//Jos se ei vastaa ennalta määrättyjä komentoja,
      return 0;                         //palautetaan nolla.
  }
  return 1;
}
```

LIITE 3: 4 (8)

```
/*** PÄÄOHJELMA ***/
int main(void)
{

  Init();    // Alustetaan I/O-portit sekä sarjaportti


  /** Odotetaan moduulin vastausta "Call Ready",        **/
  /** jolloin tiedetään että ohjelmaa voidaan ajaa eteenpäin **/
  while(checkResponse(callready)==0){
    merkki = getchar_UART(100000);
  }

  /** Moduulin alkuasetukset **/
  Send_UART("ATE0");                //Merkkien kaiutus pois päältä.
  Send_UART("AT+CMGF=1");           //Asetaan SMS-viestille tekstimuoto.
  Send_UART("AT+CLIP=1");           //Soittajan numero näkyviin.


  /** Ohjelman pääsilmukka **/
  while(1){

  if(!(bit_get(PIND, BIT(6))))      //Tutkitaan moduulin RI-pinniä.
    bit_clear(PORTD, BIT(7));       //Sytytetään merkkiledi, jos
  else                              //vastaanotetaan puhelua tai tekstiviestiä.
    bit_set(PORTD, BIT(7));

  int i;
  for(i=0; i<64; i++){
    temp_array[i] = 0;

  }

  /** Tutkitaan moduulin paluuviestejä **/
  if(rx_counter != 0){              //Onko rx-bufferissa lukemattomia merkkejä.


    merkki = getchar_UART(100000);  //Haetaan merkki rx-bufferista.

    if(merkki == '+'){              //Jos ensimmäinen merkki oli '+',

      checkSMS();                   //tutkitaan vataanotettu viesti. (+CMTI....)
    }
    else{
      if(checkResponse(ring))       //Muutoin tutkitaan onko puhelu tulossa. "RING"
      checkCaller();
    }
  }
  }
}
```

LIITE 3: 5 (8)

```
/** Aliohjelma tutkii kuka soittaa, jos soittaja on ennalta sallittu **/
/** numero, niin vastataan puheluun, muutoin se hylätään.      **/
void checkCaller(void){

  int colcount = 0;
  int i = 0;

  /** Moduuli ilmaisee soittajan numeron ja vastaavan nimen **/
  /** jos sellainen on tallennettu puhelinluetteloon.    **/
  /** +CLIP: "+35840*******",145,"",,"ADMIN",0 **/
  do{
    merkki = getchar_UART(100000);
    if(merkki == '"')                  //"-merkin perusteella siirrytään
      colcount++;                      //oikeaan tietokenttään viestissä
    if(colcount == 5 && merkki!='"')   //Jos oli viides "-merkki ollaan soittajan nimi-
      temp_array[i++] = merkki;        //kentässä. Otetaan soittajan nimi talteen.
  }while(merkki != 10 || colcount <=5);

  if(Compare(admin, strlen(admin)) ||
    Compare(user,  strlen(user)))      //Jos soittajan numero vastaa tallennettua
    Send_UART("ATA");                  //ADMIN- tai  USER-numeroa,vastataan puheluun.
  else
    Send_UART("ATH");                  //Muutoin katkaistaan puhelu
}


/** Aliohjelma tutkii onko vastaanotettu SMS-viesti.          **/
/** Lähettäjän numero otetaan talteen sekä viestin sisältämä  **/
/** komento, jonka perusteella tehdään toiminto               **/
void checkSMS(void){

  int i=0,j=0;
  int colcount = 0;
  int ch = 0;
  int pass = 0;

  while(j != 6){smsind[j++] = 0;}    //Tyhjennetään edellisen viestin muistipaikka.
  i=j=0;                             //Apumuuttujien nollaus.


/** Etsitään moduulin paluuviestistä **/
/** tekstiviestin muistipaikka.     **/
/** +CMTI: "SM",28          **/
    do{
      merkki = getchar_UART(100000);     //Haetaan merkkejä rx-bufferista.
      if(merkki != 13 && merkki != 10){  //Merkit taulukkoon mutta hylätään CR ja LF.
        temp_array[i++] = merkki;
        if(isdigit(merkki))              //Jos haettu merkki on numero, se asetetaan
          smsind[j++] = merkki;          //smsind-taulukkoon (viestin muistipaikka).
      }
    }while(merkki != 10);                //Loopissa pyöritään kunnes vastaanotetaan LF.

    smsind[j++] = 0;                     //Asetetaan merkkijonon loppu.
    ch = atoi(smsind);                   //atoi() muuntaa merkkijonon numeroksi.
    sprintf(atCommand,"AT+CMGR=%d",ch); //Asetetaan taulukkoon viestin lukemiskomento.

    if(Compare(sms_rec, strlen(sms_rec))){ //Verrataan onko moduulin antama viesti,
      Send_UART(atCommand);                //ilmoitus uudesta viestistä. Jos on,
                                           //lähetetään moduulille lukemiskomento.
```

LIITE 3: 6 (8)

```
/** Viestin lukemiskomennon jälkeen moduuli lähettää viestin tiedot. **/
/** Haetaan lähettäjän numero, ja vastaava nimi puhelinluettelossa   **/
/** +CMGR: "REC READ","+358408387368","ADMIN","11/08/30,14:12:23+12" **/
/** AVR GSM testi                                **/
  while(merkki != '+')
  merkki = getchar_UART(100000);   //Odotetaan '+'-merkkiä.

  i=j=colcount=0;                  //Apumuuttujien nollaus
  do{
    merkki = getchar_UART(100000); //Haetaan merkkejä bufferista.
    if(merkki == '"')              //"-merkin perusteella
      colcount++;                  //siirrytään oikeaan tietokenttään.
    if(colcount == 3){             //Kolmas "-merkki, ollaan puhnro kentässä.
      if(isdigit(merkki))          //Jos merkki on numero, se asetetaan
        sms_number[i++] = merkki;  //sms_number-taulukkoon.
    }
    if(colcount==5 && merkki!='"'){//Viides "-merkki ollaan nimi-kentässä.
    temp_array[j++] = merkki;      //Mahdollinen numeroa vastaava nimi
    }                              //puhelinluettelossa talteen.
  }while(merkki != 10);

  sms_number[i] = '\0';            //Asetetaan merkkijonon loppu.

  if(Compare(admin, strlen(admin))||
    Compare(user, strlen(user)))   //Jos viestin lähettäjä vastaa tallennettua ADMIN-
    pass = 1;                      //tai USER-numeroa, sallitaan releiden ohjaus.
  else
    pass = 0;

/** Seuraavana haetaan itse viestin sisältö **/
  i=0;
  do{
    merkki = getchar_UART(100000);      //Haetaan merkkejä bufferista.
    if(merkki != 13 && merkki != 10)    //Merkit temp-taulukkoon, CR ja LF hylätään.
      temp_array[i++] = merkki;
  }while(merkki != 10);

   sprintf(atCommand,"AT+CMGD=%d",ch);  //Luodaan viestin poistokomento.
  Send_UART(atCommand);                 //Halutut tiedot on tallessa, poistetaan SMS.


/** Viestin sisältöä verrataan ennalta **/
/** määrättyihin komentoihin.      **/
  if(pass == 1){
    if(Compare(relay1on,       //Jos sisältö vastaa komentoa,
      strlen(relay1on))){      //toimitaan sen mukaan.
      bit_set(PORTC, BIT(7));  //aseta Rele1
      checkStatus();
      sendSMS();
    }
    else if(Compare(relay2on,
      strlen(relay2on))){
      bit_set(PORTC, BIT(6));    //Aseta Rele2.
      checkStatus();
      sendSMS();
    }
    else if(Compare(relay1off,
      strlen(relay1off))){
      bit_clear(PORTC, BIT(7));  //Nollaa Rele1.
      checkStatus();
      sendSMS();
    }
```

LIITE 3: 7 (8)

```
      else if(Compare(relay2off,
        strlen(relay2off))){
        bit_clear(PORTC, BIT(6));  //Nollaa Rele2.
        checkStatus();
        sendSMS();
      }
      else if(Compare(status,      //Lähetä tila.
        strlen(status))){
        checkStatus();
        sendSMS();
      }
      else{
        int f=0;
        for(f = 0; f<160; f++){
        message[i]= '\0';}
        sprintf(message, "Tuntematon komento!");
        sendSMS();
      }
    }
  }
}


/** Aliohjelma joka lähettää tekstiviestin.      **/
/** Viesti lähetetään samaan numeroon josta edellinen **/
/** komento on tullut.                 **/
void sendSMS(){

  int k = strlen(sms_send);             //SMS lähetyskomennon pituus


  /** Muodostaa komennon:   **/
  /** CMGS="+358*********" **/
  memcpy(sms_send_number, sms_send, k); //Kopioidaan SMS lähetyskomennon alku
  memcpy(sms_send_number + k,           //Kopioidaan edellisen perään puh. nro.
  sms_number, (strlen(sms_number)));    //Numeron pituus talteen.
  k = k + strlen(sms_number);           //Siirretään indeksiä oikeaan kohtaan.
  sms_send_number[k++]='"';
  sms_send_number[k]='\0';

  Send_UART(sms_send_number);           //Viestin lähetyskomento moduulille.

  /** Moduuli vastaa merkillä '>' **/
  /** Jonka perään syötetään viesti**/
  while(!(merkki == '>'))               //Odotetaan '>'-merkkiä.
    merkki=getchar_UART(100000);


  Send_UART(message);                   //Syötetään haluttu tekstiviesti.
  putchr(26);                           //Viestin kirjoitus lopetetaan SUB-merkillä.

  while(merkki != '+')
  merkki = getchar_UART(100000);        // Moduuli vastaa +CMGS: x

}
```

LIITE 3: 8 (8)

```
/** Hakee releiden tilat tekstiviestin lähetystä varten **/
/** Sekä muita tietoja joita mahdollisesti halutaan lähettää **/
void checkStatus(void)
{
  int i, j, colcount;
  i = j = colcount = 0;
  for(i = 0; i<160; i++)
    message[i]= '\0';

  i=0;
  char mode[4];
  char capacity[4];
  char voltage[4];
  for(i=0; i<4; i++)
  mode[i]=capacity[i]=voltage[i]=0;
  i=0;

  if(PORTC & 0b10000000){
    sprintf(message +i,"RELAY1=1\n");
    i=i+9;
  }else{
    sprintf(message +i,"RELAY1=0\n");
    i=i+9;
  }
  if(PORTC & 0b01000000){
    sprintf(message +i,"RELAY2=1\n");
    i=i+9;
  }else{
    sprintf(message +i,"RELAY2=0\n");
    i=i+9;
  }

  /** Haetaan akun tila **/
  Send_UART("AT+CBC");

  while(merkki != '+')
  merkki = getchar_UART(100000);

  /** Muodostetaan akun tilaa kertova viesti **/
  do{
    merkki = getchar_UART(100000);
    if(merkki == ','){
      colcount++;
      j=0;
    }
    if(isdigit(merkki)){
      switch(colcount){
        case 0:
        mode[j++] = merkki;
        break;
        case 1:
        capacity[j++] = merkki;
        break;
        case 2:
        voltage[j++] = merkki;
        break;
      }
    }
  }while(merkki != 10);

  sprintf(message +i,"\n*Battery info* \nCharge mode: %s\nCapacity: %s%%\nVoltage: %smV",
  mode, capacity, voltage);
}
```