Daten-/Ladekabel basteln:

Um ein Mobiltelefon dauerhaft am Rechner anzuschließen, sollte das Datenkabel auch der Stromversorgung dienen. Dazu kann ich folgende Lösungen anbieten:

- fertiges Y-Kabel
  Evtl. existieren Y-Kabel von Fremdanbietern, welche den Anschluss von Ladegerät und Datenkabel(!) ermöglichen.
- Tischladestation
  Für manche Modelle existiert eine Tischladestation, welche den Anschluss von Lade- und Datenkabel ermöglicht (Original Siemens Zubehör).
- selber Löten
  ist natürlich die günstigste und schnellste Lösung. Im Anhang gibt's die Steckerbelegung der S25-S45 Serie (Quelle: http://www.nobbi.com). Einfach von beiden Kabeln die Stecker öffnen und die 3 Kontakte des Ladekabel an das Datenkabel anklemmen. Masse natürlich gemeinsam nutzen!

  Wer das Datenkabel selber basteln will findet bei http://www.nobbi.com Infos.

Anschließen

Das Datenkabel an eine freie serielle Schnittstelle hängen. Testen der Verbindung mit einem Terminalprogramm. Unter Linux z.B. mit „minicom" unter Windows mit Hyperterminal. Im Terminalprogramm passende Schnittstelle auswählen, (z.B. /dev/ttyS0 für COM1) und die Datenrate passend einstellen (S25 mit 19200 betreiben). Keine Flusskontrolle aktiviert.
Auf „AT<enter>" sollte das Telefon mit „OK" reagieren. Auf „AT+CGMI" sollte es sich mit „+CGMI: SIEMENS" melden ;-)

Software

Die z.Z. beste SMS Software ist Kannel. Kannel ist eigentlich ein WAP Gateway, unterstützt aber auch SMS Empfang und Versand. Wird eine SMS empfangen kann sie bestimmte Services auslösen, z.B. Webseiten abrufen. Der Versand von SMS erfolgt über eine HTTP Schnittstelle. Kannel besteht aus drei Komponenten:

- Bearerbox
  Bedient die unteren Layer und wird für beide folgenden Komponenten benötigt.
- Wapbox
  Dient als WAP Gateway. Man kann z.B. die IP-Adresse im Mobiltelefon als Gateway einstellen und verwendet dann seinen eigenen WAP-Gateway
- Smsbox
  Stellt die SMS Dienste zur Verfügung

Die Developer-Version 1.1.6 in der Lage ein Mobiltelefon als GSM Modem zu verwenden.

Kannel Installation

Das Kannel Paket downloaden (http://www.kannel.3glab.org/download.shtml),
z.B. „gateway.tar.gz - Source as tarball.“ oder die aktuellste Version per CVS ziehen. Unter Win32 muss die Cygwin Umgebung genutzt werden. Kannel läuft also auch unter Windows.

Entpacken, ins Verzeichnis „gateway“ wechseln und mit ./configure das Makefile erzeugen. Ist das Makefile erfolgreich erzeugt, startet „make“ den Kompiliervorgang. „make install“ installiert die Binaries (Achtung: Manchmal mit Versionsnummer: /usr/local/sbin/bearerbox-1.1.6)

Sind die Binaries installiert fehlt nur noch die Konfigurationsdatei. Diese enthält verschiedene Sektionen:

„core“: Hier werden globale Einstellungen aller Softwarekomponenten vergenommen.
Das admin-password sollte angepasst werden. Der Zugriff auf die Komponenten kann über admin-deny-ip und admin-allow-ip geregelt werden.

```
group = core
admin-port = 13000
smsbox-port = 13001
wapbox-port = 13002
admin-password = XXXXXXXXXXXXX
log-file = "/var/log/kannel/bearer.log"
log-level = 4
box-allow-ip = "*.*.*.*"
wdp-interface-name = "*"
#box-allow-ip = "127.0.0.1"
#admin-deny-ip = ""
#admin-allow-ip = ""
unified-prefix = "0049,0"
#access-log = "access.log"
```

„wapbox“: Konfiguration der Wapbox, ist für SMS nicht notwendig

```
group = wapbox
bearerbox-host = localhost
log-file = "/var/log/kannel/wapbox.log"
log-level = 4
syslog-level = none
```

„smsc“: Hier wird’s interessant. „smsc=at2“ legt die Zugriffsmethode über AT Befehle (Version 2 in Kannel) fest. Wird das Modem „s25“ automatisch erkannt, verwendet Kannel diese Konfiguration. Schnittstellenparameter und einige weitere offensichtliche Daten. „Sim-Buffering“ heisst soviel wie: SMS werden empfangen und vom Mobiltelefon auf der SIM gespeichert, erst von dort holt sich Kannel im Zeitabstand „keepalive“ die SMS ab. (Alternativ könnte Kannel die SMS auch direkt vom Mobiltelefon empfangen.)

```
group = smsc
smsc = at2
modemtype = s25
device = /dev/ttyS0
speed = 19200
#retry = true
sms-center = +491722270000
sim-buffering = true
pin = 0000
keepalive = 10
```

„modems“: Hier wird das „Modem“ „s25“ definiert. Der detect-string „032“ ist die Antwort des s25 auf ein „ATI1“ (S35/S45 liefern andere Werte! Vorher mit Terminalprogramm abchecken).

```
group = modems
id = s25
name = s25
detect-string = "032"
message-storage = "SM"
init-string = "AT\Q3"
```

„smsbox“: global-sender anpassen. (Telefon setzt selber die Absender Rufnummer)

```
group = smsbox
bearerbox-host = localhost
sendsms-port = 13013
global-sender = +49111111111
#sendsms-chars = "0123456789 +-"
log-file = "/var/log/kannel/smsbox.log"
log-level = 4
#access-log = "access.log"
```

„sms-service“: SMS Dienste. Sendet jemand eine SMS mit dem Inhalt „www www.telehaus.net“ an das Modem, erhält er eine SMS mit der „Homepage“ (als reiner Text, nicht HTML!) des Telehauses zurück. %S steht für den Inhalt der ersten SMS hinter dem Keyword „www“.

```
group = sms-service
keyword = www
url = "http://%S"
```

Provisioning Service. Dieser Service antwortet auf Anfragen „konfiguration vodafone“ mit einem WAP Provisioning Dokument mit den Vodafone Parametern. Wird voraussichtlich erst ab S55 unterstützt. Damit lassen sich die WAP Parameter im Mobiltelefon per SMS (OTA) konfigurieren. Max-messages bedeutet, dass die Ausgabe des PHP Skriptes nicht über SMS zurückgeschickt wird (Das PHP Skript sendet selber eigene SMS ab). %p ist der Absender der eintreffenden SMS, %s das zweite Wort in der SMS (das erste ist „konfiguration“). Das dazugehörige PHP-Skript prov.php ist leider nicht frei verfügbar!

```
group = sms-service
keyword = konfiguration
url = "http://127.0.0.1/prov.php?to=%p&provider=%s"
max-messages = 0
```

SI sind „Service Indication“. Eine Meldung auf dem Display des Mobiltelefons mit einer URL hinterlegt, welche direkt aufgerufen werden kann. %r ist der Rest der eintreffenden SMS nach dem zweiten Wort. Beispiel für eine SMS: „si http://www.telehaus.net Besuch doch mal das Telehaus!“. Das S/ME45 ist das erste Mobiltelefon, welches SI unterstützt. Das dazugehörige PHP-Skript si.php ist im Anhang zu finden! SI benötigt DCS=0x15, Kannel prüft den Wert und verwirft ihn, wenn er größer als 0x02 ist. Die Sourcen vom Kannel müssen entsprechend gepatcht werden.

```
group = sms-service
keyword = si
url = "http://127.0.0.1/si.php?to=%p&url=%s&msg=%r"
max-messages = 0

group = sms-service
keyword = default
text = "No service specified"
```

„sendsms-user“: Definition des Benutzernamen/Passwort für die Webschnittstelle zum SMS Versand

```
group = sendsms-user
username = YYYYY
password = XXXXX
user-allow-ip = "127.0.0.1"
max-messages = 4
```

Kannel starten

Die 3 Softwarekomponenten von Kannel werden auf unterschiedlichen Konsolen gestartet.
1. Konsole: „bearerbox-1.1.6 /etc/kannel/kannel.conf“, auf der
2. Konsole: „smsbox-1.1.6 /etc/kannel/kannel.conf“ und auf der
3. Konsole, sofern benötigt: „wapbox-1.1.6 /etc/kannel/kannel.conf“

Die Debugausgabe von der Bearerbox enthält Informationen über die versendeten AT Befehle. Bricht die Bearerbox ab, werden automatisch auch die anderen Komponenten beendet! Die Smsbox stellt eine HTTP Schnittstelle auf Port 13013 zur Verfügung, dort kann eine URL abgerufen werden, welche den Versand einer SMS startet:
„http://127.0.0.1:13013/cgi-bin/sendsms?user=XXX&pass=YYY&to=00000&text=HALLO“

Um von einem anderen Programm eine SMS zu versenden muss nur diese URL mit den passenden Parametern aufgerufen werden. In Shell Skripts kann man zum Beispiel „lynx -source“ oder „wget –O /dev/null“ verwenden. Unter Perl und PHP existieren Funtionen, mit denen man eine URL aufruft oder z.B. extern „wget“ starten kann

Funktioniert alles sollte man Kannel (unter Linux) in /etc/inittab integrieren, dann ist sichergestellt, dass es immer läuft (Achtung: Pfade anpassen, Wapbox nur wenn notwendig!):

```
kanb:1235:respawn:/usr/local/sbin/bearerbox /etc/kannel/kannel.conf
kans:1235:respawn:/usr/local/sbin/smsbox /etc/kannel/kannel.conf
kanw:1235:respawn:/usr/local/sbin/wapbox /etc/kannel/kannel.conf
```

Ein „kill –HUP 1“ lässt den Init Prozess die Konfiguration neu einlesen, mit „ps aux“ kann man kontrollieren ob die Prozesse laufen (Achtung: Es tauchen mehrere Prozesse mit den Namen auf. Das ist normalerweise nur ein Programm das läuft, mit seinen Kind Prozessen) („pstree“ zeigt die Zusammenhänge)

Unter Windows muss man sich was anderes zum automatischen Starten der Komponenten ausdenken...

Auf der Firewall sollte man alle genutzten Ports abschotten. Die verwendeten Ports lassen sich unter Linux mit „lsof -n | grep box | grep IPv4“ rausfinden.

Anhang

Die Steckerbelegung für Siemens-Telefone x25-x45

Quelle: http://www.nobbi.com

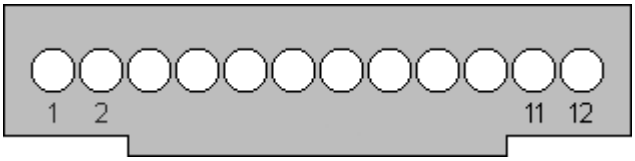

Abbildung 1

(Bild gesehen auf die Lötseite des Steckers, so, wie er im Betrieb im Handy steckt, die Tastatur ist also oben) Die Belegung trifft zu auf die folgenden Telefone: S/C25, C/M/S35, C/ME/S/SL45, Gigaset4xxx

Pin	**Name**	**Funktion**	**sonstiges**	**in / out**
1	GND	Masse		
2	SB	Erkennung/Steuerung Ladegerät	LOW = 150mA-Ladegerät HIGH = 1A-Ladegerät NC = 400mA-Lader bei 35/45	in / out
	AUD01	SL45: Audiokanal 1		out
3	CHARGE	Ladespannung/strom	U = 6,1V - 8,0V	in
4	BATT	Spannungsversorgung für Zubehör	U = 3,0V - 3,9V Umin = 2,6V Imax = 100mA	out
5	DATA OUT	Daten abgehend	PullUp im Telefon	out
6	DATA IN	Daten ankommend	PullDown im Endgerät	in
7	Z_CLK	Erkennung/Steuerung Zubehör		
8	Z_DATA	Erkennung/Steuerung Zubehör		
9	MICG	GND für Mikrofon		
10	MIC	Mikrofon	U = 1,5Vpp	in
11	AUD	Lautsprecher	U = 1,5Vpp	out
	AUD02	SL45: Audiokanal 2		
	AUDG	4xxx: GND für Lautsprecher		
12	AUDG	GND für Lautsprecher		
	AUD	4xxx: Lautsprecher/DC-Offset	Signal ist nicht ausgekoppelt!	

SI.PHP

```
<?

 include("util.inc.php");

 $kannel = array ( host => "127.0.0.1", port => "13013", user
=> "yyyy", pass => "XXXX");
 $debug=0;
 $fields['debug'] = "";
 $fields['from'] = "0111111111";
 $fields['to'] = urlencode($to);


$content=se_byte(0x01).se_byte(0x06).se_byte(0x03).se_byte(0xa
e).se_byte(0xaf).se_byte(0x82);

 $content .=
"%01%05%6A%00%45%C6%11%03%31%32%00%0B%03".se_binarystring($url
,strlen($url))."%00%08%01%03".se_binarystring($msg,strlen($msg
))."%00%01%01";

 sendpushsms($content,$size);

?>SI wird übertragen...
```

UTIL.INC.PHP

```
<?

function se_getbyte($content,$byte)
{
return substr($content,($byte-1)*3,3);
}

function se_byte($c) {
 return "%".sprintf("%02X",$c);
}

function se_file($file) {
 $cc="";
 $f=fopen($file,"rb");
 while(!feof($f))
 {
   $c=fread($f,1);
   $cc .= "%".sprintf("%02X",ord($c));
 }
 return $cc;
}
```

```
function se_binarystring($content,$size)
{
 $cc="";
 for($i=0;$i<$size;$i++){
  $cc.= "%".sprintf("%02X",ord(substr($content,$i,1)));
 }
 return $cc;
}

function sendsms (){
        global $fields, $debug, $kannel;

          $fields['mode'] = "";
          reset($fields);
          while(list($k,$v) = each($fields)) {
            if ( $v != "" ) {
              $string .= "&$k=$v";
            }
          }

          if($debug) print ($debug ? "[DEBUG]" : "" )."Getting
http://".$kannel[
'host'].":".$kannel['port']."/cgi-
bin/sendsms?user=".$kannel['user']."&pass=".$k
annel['pass'].$string."<br>\n\n";

          if ( !$debug ) { $result =
@file("http://".$kannel['host'].":".$kannel
['port']."/cgi-
bin/sendsms?user=".$kannel['user']."&pass=".$kannel['pass'].$s
tri
ng); }

          if($debug) print_r( $result);
  }

function sendpushsms($content)
{
 global $fields, $debug, $kannel;

 srand ((double) microtime() * 1000000);

 $size=floor(strlen($content)/3);
 $count=1;
 $maxsms=ceil($size / 128);
 $ID=rand(1,127);

 while($count <= $maxsms)
 {
  if($maxsms>1)
   $fields['udh'] =
"%0B%05%04%0B%84%23%F0%00%03%".sprintf("%02X",$ID)."%".sprin
```

```
tf("%02X",$maxsms)."%".sprintf("%02X",$count);
  else
   $fields['udh'] = "%06%05%04%0B%84%23%F0";
  $fields['alt-dcs'] = "21";
  $fields['text'] = substr($content,(($count-1)*128)*3,128*3);

  sendsms();
  $count++;
 }

}

?>
```

Patch

In gw/smsc_at2.c aus den Zeilen

```
       dcs = fields_to_dcs(msg,
        (msg->sms.alt_dcs ? 2 - msg->sms.alt_dcs : privdata->alt_dcs));
```

diese hier machen

```
    if(msg->sms.alt_dcs>2)
    {
            dcs=msg->sms.alt_dcs;
    }
    else
    {
       dcs = fields_to_dcs(msg,
         (msg->sms.alt_dcs ? 2 - msg->sms.alt_dcs : privdata->alt_dcs));
    }
```

Und in smsbox.c die if-Bedingung ändern:

```
    if ( alt_dcs < 0 || alt_dcs > 255 ) {
        returnerror = octstr_create("Alt-DCS field misformed, rejected");
        goto fielderror;
    }
```