GME - Dokumentation

Sgns by Günther Grund
Habbinss by Didi2004
Repaint by Daniel Grasegger

Kurzzusammenfassung:

Diese Dokumentation soll einen Einblick in das Maturaprojekt GME geben.
Das Maturaprojekt GME befasst sich mit der Geschwindigkeitsmessung von Waggons, insbesonders jene, welche im ZVB Kledering vom Abrollberg kommen und in die entsprechenden Gleise rollen. Ziel ist es, jene Waggons zu erfassen, welche trotz des vorhandenen Bremssystems noch immer zu schnell sind.
Die Kapitel sind entsprechend der Aufteilung der einzelnen Teilprojekte und beschreiben die Arbeiten der einzelnen Teammitglieder. Die Kapitel umfassen die Arbeitsschritte und die Funktionserklärungen der einzelnen Teilprojekte, aber auch die Zusammenarbeit dieser.
Da die Kapitelreihenfolge dem Funktionsweg der Hardware entspricht, soll eine schrittweise Erläuterung der Funktion des gesamten Systems gegeben werden, ohne jedoch einzelne Kapitel von vorangegangenen abhängig zu machen.

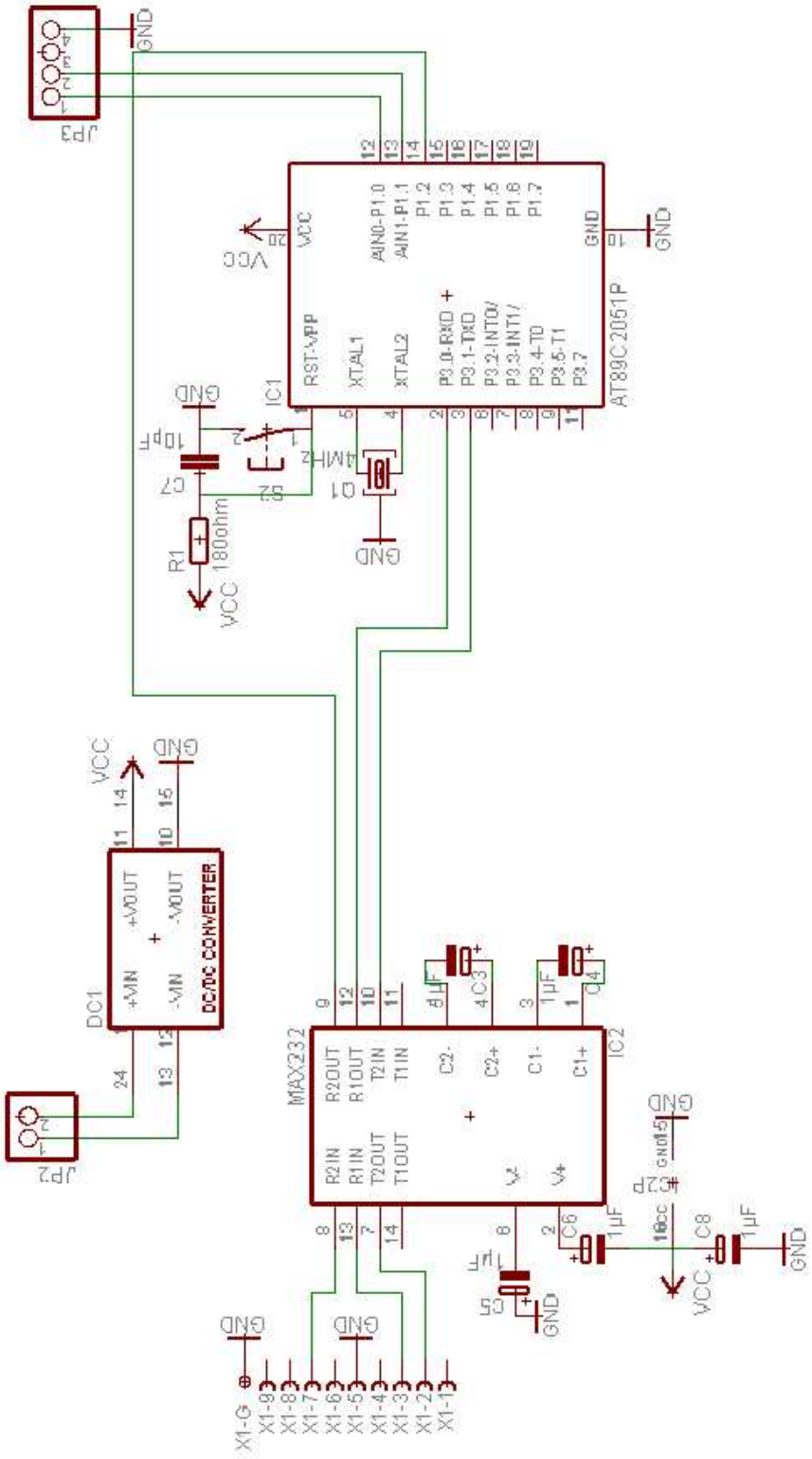
Short summary:

This documentation is to give an idea of the maturaproject GME.
The maturaproject GME is concerned with the speed measurement of railroad cars, in particular those, which in the ZVB Kledering of the unreeling mountain come and roll into the appropriate tracks. A goal is it to seize those railroad cars which are still too fast despite the existing brake system.
The chapters are according to the allocation of the individual subprojects and describe the work of the individual team members. The chapters cover the work procedures and the function explanations of the individual subprojects, in addition, the co-operation of these.
Since the chapter sequence corresponds to the function way of the hardware, a gradual explanation is to be given to the function of the entire system, without making however individual chapters dependent on preceding.

Inhaltsverzeichnis

1 Hardware

1.1 Schaltplan



1.2 Mikrokontroller- Programmierung

Hier werde ich allgemeine Informationen und die Funktion des Programms für den Mikrokotroller des Projektes GME (Geschwindigkeitsmesseinrichtung) näher erläutern.

Programmiert wurde in Franklin.

Verwendet wurde der Mikrokontroller 89C2051 aus der 8051 Familie.

Prinzipielle Funktion des Mikrokontrollers:

Der Mikrokontroller besitzt zwei 16-Bit Timer/Counter. Die 16 Bits bestehen aus den Registern TL 0/1 und TH 0/1. Der Timer 0 wird dazu verwendet, um die Anzahl der Takte zwischen den Impulsen der Lichtschranken zu zählen. Der Timer 1 wird als Baudrate- Generator verwendet. Der Mikrokontroller wird durch einen Quarz mit bestimmter Frequenz getaktet. In unserem Projekt beträgt der Keramikresonator 4 MHz. Jedoch kann man sich nicht auf die angegebene Frequenz verlassen, da Keramikresonatoren ungenau und temperaturabhängig sind. Und da unser Projekt im Freien Verwendung findet, ist dieses Kriterium nicht zu vernachlässigen. Deswegen ist eine Time-check Leitung vorgesehen, um die genaue Taktfrequenz festzustellen, die aber später noch genauer erklärt wird.
Um die Geschwindigkeit der Waggons zu erfassen, muss die Zeit zwischen Erfassen eines Waggons des 1. Lichtschrankens und des 2. Lichtschrankens gemessen d.h. die Anzahl der Takte des Mikrokontrollers gezählt werden. Dies geschieht durch den Timer 0. Der Mikrokontroller zählt immer nur die Anzahl der Vergangenen Takte, egal ob Lichtschranke A oder Lichtschranke B zuerst anspricht. Damit können Waggons von beiden Richtungen erfasst werden. Um festzustellen in welche Richtung der Waggon rollt, wird ein „Richtungsbit“ an der letzten Stelle des 2. Bytes mitgeschickt. Wenn das Richtungsbit 0 ist, kommt das Signal vom Lichtschranken A, wenn das Richtungsbit gesetzt ist, vom Lichtschranken B. Das 1. Mal wenn ein Lichtschranke auslöst, werden nur Nullen und das Richtungsbit gesendet und der Timer 0 wird gestartet. Bei jedem weiteren Mal wird die Anzahl der Takte und auch das Richtungsbit nach Bild 1 gesendet. Wenn der Timer *MaxOV* Sekunden überschreitet, wird er gestoppt, da angenommen wird dass der Waggon bzw. die Waggonkette bereits vorbeigerollt ist. Die gezählten Takte werden in zwei Bytes über eine serielle Schnittstelle an das embedded System geschickt:

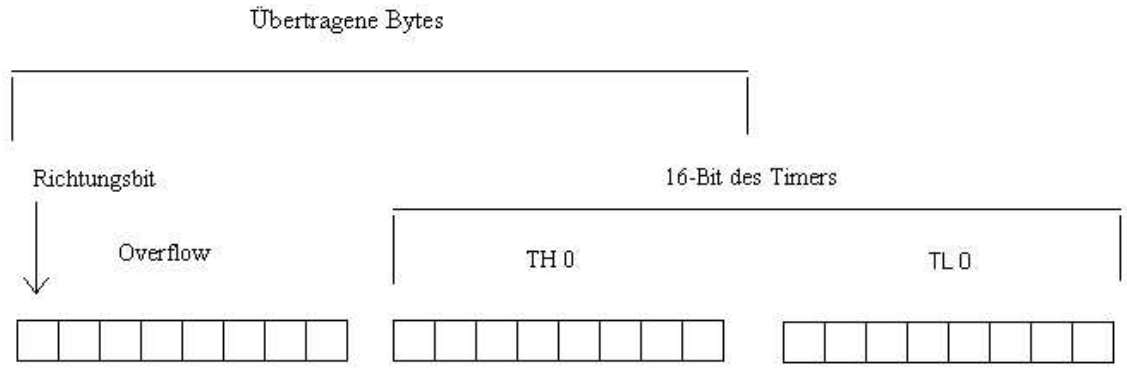


Bild 1: Bitaufteilung des Timers 0 und der seriellen Übertragung

```
Interrupt
void timer0 (void)  interrupt 1 using 2
{
  overflow++;
}
```

Mit diesem Befehl wird der Interrupt des Timer 0 definiert. Bei Erreichen des Wertes 65536 des 16-Bit Speichers(2^{16}), wird der Interrupt ausgelöst. Dabei wird die im Programm definierte Variable *overflow* um 1 inkrementiert. In unserem Fall findet dies alle $12*65536/(4*10^6)$ 200 ms

statt. Somit brauchen wir uns keine Sorgen machen, dass das 7. Bit des Overflow besetzt wird, da es ja durch das Richtungsbit besetzt ist.

MaxOV
das ist die Zeit die den Abbruch des Zählvorganges vornimmt, d.h. der Timer0 wird gestoppt. Achtung: die Zeit darf maximal 25 Sekunden betragen, sonst wird das 8. Bit des overflow gebraucht, das aber durch das Richtungsbit überschrieben wird, damit wäre das Ergebnis nicht sinnvoll.

1.2.1 Timer

C/T (TMOD.2/TMOD.6) Auswahl der Timer- oder Counter (externer Takt) Funktion.
M0 und *M1* (TMOD.0 und TMOD.1 für Timer 0 und TMOD.4 und TMOD.5 für Timer 1) legen den Mode des jeweiligen Timer/Counters fest. In diesem Fall arbeitet Timer 0 im Mode 1 für einen vollständigen 16-Bit Zähler, und Timer 1 im Mode 2 mit einer automatischen Reload-Funktion für die Generierung der Baudrate.

TR0/TR1 Startbit des Timers0/1, durch diese Parameter wird der Timer gesteuert. TR1 ist immer HI, d.h. der Timer läuft ununterbrochen weil für die Erzeugung der Baudrate keine Ein- und Ausschaltvorgänge nötig sind.

EA=1; allgemeine Freigabe der Interrupts
ET0=1; Freigabe des Timer-0-Überlauf-Interrupt

1.2.2 Time-check

Diese Funktion wird beim Start ausgeführt. Aber auch mehrmals am Tag, da sich durch Temperaturänderung die Taktfrequenz ändert. Damit kann man jederzeit die Frequenz des Resonators überpüfen. Beim Time-check Abgleich wird 1 Sekunde lang die vom Rechner kommende Time-check Leitung auf Ground gelegt. Der Mikrocontroller zählt die Takte des Timers 0, wie durch Auslösung des Lichtschrankens, solang die Leitung auf Ground liegt. Wenn die Leitung wieder auf 1 liegt, wird die wiederum die Anzahl der Takte über die Serielle Verbindung an den Rechner weitergegeben, der die Frequenz ausrechnen und die Änderung berücksichtigen kann, um wirklich möglichst genaue Zeiten zu messen. Falls in dieser Zeit ein Lichtschranke ansprechen sollte, muss dieser ignoriert werden.

1.2.3 Hauptprogramm

Das Hauptprogramm enthält eine Endlosschleife (while(1)), in der abgefragt wird, ob einer der Lichtschranken anspricht, die Time-check Leitung auf Ground gezogen wird oder die *MaxOV* -Zeit überschritten ist. Bei einer Änderung der angegebenen Fälle, geht man in das dementsprechende Unterprogramm über.
oldpina und *oldpinb* sind da, um das entsprechende Unterprogramm nur 1 Mal auszuführen, da das Signal des Lichtschranken länger anhalten wird, als das Unterprogramm auszuführen ist. Somit wird nur die Änderung vom Passiven ins Aktive festgestellt.

1.2.4 Serielle Schnittstelle bzw. Baudrate-Generierung: sm0

Die Betriebsweise des Timer1 arbeitet in Mode 2. Den Mode bestimmt man durch M0 und M1 im TMOD- Register. Dadurch kann man eine variable Baudrate mit Hilfe des Timers generieren. Dieser Mode wird mit Auto-Reload-Mode betrieben. Berechnet wird der Wert *xTH1* durch Befehl *#define xTH1* zu Beginn des Programms. 0.5 wird zu dieser Formel addiert, da eine Kommazahl entsteht. Und um diese Kommazahl auf einen ganzzahligen Wert zu runden, wird 0.5 addiert da die Kommastellen beim Schreiben in *TH1* wegfallen. *xTH1* ist nur die Hilfsvariable für *TH1*, die Zuweisung erfolgt erst später, da man *TH1* nicht durch *#define* zuweisen kann.
Der Timer-Overflow wird als Eingangssignal verwendet. Das bedeutet in das *TL1* Byte wird der Wert *xTH1* also *TH1* geschrieben, und der Timer-Überlauf ist der Taktgeber. Danach wird automatisch der Inhalt von *TH1* wieder in *TL1* geladen, und der Timer zählt wieder bis zum Überlauf usw. Der Inhalt von *TH1* bleibt unverändert.

Gestartet wird der Sendevorgang, indem SBUF (Datenpuffer) als Zieladresse eines Schreibbefehls angegeben wird. In diesem Programm *SBUF=“Byte1/2“;* Nach diesem Startbefehl wird Bit für Bit bei jedem Timer-Überlauf gesendet.

SMOD Wird für die Berechnung der Baudrate verwendet. Damit kann man unterschiedliche Baudraten erzeugen. Dieses Bit wird durch Schreiben des PCON- Registers definiert. SMOD kann also 0 oder 1 betragen.

1.2.5 LED Beschreibung

Nach dem Einschaltvorgang blinken beide LEDs dreimal kurz auf.
Bei jedem Sendevorgang wechseln die beiden LEDs hin und her.
Bei Ausführung des time- check Vorgangs, der auch zu Beginn vorgenommen wird, leuchten beide LEDs solang die time- check Leitung auf 0 liegt.
Bei Auslösung des Lichtschrankes A leuchtet die LED am Pin1.5 (rot), solang bis ein anderes Ereignis eintrifft.
Bei Auslösung des Lichtschrankes B leuchtet die LED am PIN1.6(gelb), solang bis ein anderes Ereignis eintrifft.
Nach Erreichen der Zeit *MaxOV* erlöschen beide LEDs.

2 GME – Linux

Das GME-Linux wurde auf einem Computer mit SuSE Linux 8.1 Professional aufgebaut. Als Vorlage für GME-Linux wurde die Embedded Linux Version auf der dem Buch *Messen, Steuern, Regeln mit Linux* von *Klaus-Dieter Walter* Ausgabe 2001 beiliegenden CD verwendet. Auf der CD im Ordner linux/konfig1 (Linux-Verzeichnisname) bzw. \Linux\Konfig1 (Windows-Verzeichnisname) sind Kernel und Ramdisk zu finden.

Da jedoch der Kernel zu alt für bestimmte Anwendungen ist, wurde ein neuer Kernel (Quellcodeversion 2.4.19) kompiliert.

Von einer Ramdisk wurde aufgrund der Größe der Verzeichnisstruktur abgesehen, trotzdem finden sich in dieser Dokumentation die erforderlichen Befehle für die Erstellung einer Ramdisk (siehe 2.2). Bei GME-Linux wurde, ähnlich der Vorlage, eine Busybox (Quellcodeversion busybox-unstable-20021219 nicht vom Namen täuschen lassen!) und ein Tinylogin (Quellcodeversion tinylogin-1.2) verwendet.

Die Ordner `/bin`, `/etc`, `/sbin`, `/usr/sbin`, `/var` wurden von der Vorlage übernommen und im nachhinein abgeändert.

GME-Linux ist so konfiguriert, dass über Minicom (Linux) oder Hyperterminal (Windows) auf das System zugegriffen werden kann. Dazu muss man die beiden Computer (GME-Linux mit anderem Computer) seriell verbinden. Es ist hierbei die COM1 (Windows) bzw. ttyS0 (Linux) zu verwenden.

Die Einstellungen für Minicom bzw. Hyperterminal sind:

Geschwindigkeit:	115200 Bit/s
Parität:	ungerade (odd)
Stoppbits:	1
Datenbits:	8

2.1 Howto – Hilfe

Im nachfolgenden Text sind viele Schritte kurz aufgeführt die notwendig sind um ein funktionierendes Linux-System zu erstellen. Genaue beschreibung der Schritte findet sich in Howto's im Internet und auch lokal auf Linux-Systemen.

Linux – Kernel kompilieren:
http://www.oreilly.de/german/freebooks/rlinux3ger/ch075.html#36862
http://www.lugs.ch/lib/doc/kernel-compiling.phtml.de
http://www.kernelnotes.de/dlhp/DE-Kernel-HOWTO.html
http://www-aix.gsi.de/~kay/Linux-Kurs/kernel.html
http://www.tldp.org/linuxfocus/Deutsch/May1998/article37.html (+Ramdisk)

Ramdisk:
http://www.informatik.fh-trier.de/~weber/misc/linux

Lokal (SuSE-Linux 8.1 Pfade)
file:/usr/share/doc/howto/de/html/DE-Kernel-HOWTO-1.html
file:/usr/src/linux/Documentation/ramdisk.txt (Kernel-Sources müssen installiert sein!)

2.2 Ramdisk erstellen

Die Ramdisk ist eine Datei die die gesamte Ordnerstruktur des Betriebssystems enthält und beim Starten in den RAM geladen wird, d.h. nach jedem neuen Bootvorgang wird die Ramdisk in den Arbeitsspeicher geladen und geht wird beim Ausschalten aus dem Arbeitsspeicher entfernt. Mit den folgenden Befehlen kann man eine Ramdiskdatei erstellen. Siehe 2.10.1 um nachzulesen wie eine Ramdisk beim Booten verwendet wird.

```
dd if=/dev/zero of=/zielverzeichnis/zieldatei bs=1k count=1024
```

Der obige Befehl erstellt eine leere 1 MB (1024 kB) große Datei. Der Dateiname ist frei wählbar und statt `zieldatei` einzutragen.

Als nächstes muss die Ramdisk mit einem Dateisystem versehen werden. Dies geschieht mit dem Befehl `mkfs.dateisystem dateiname`, anstatt dateisystem ist eines der Verfügbaren Dateisysteme einzutragen, z. B. für ein Minix-Dateisystem `mkfs.minix dateiname`.

Nach erstellen des Dateisystems muss die noch leere Ramdisk mit dem Befehl `mount -o loop ramdiskname mountpoint` gemountet werden. Als Standardmountpoint empfiehlt sich `/mnt`.

Sobald die Ramdisk gemountet wurde kann die vorbereitete Ordnerstruktur mit `cp -av quelle ziel` (zB.: /mnt) kopiert werden. Der Parameter -a stellt sicher das sich die Dateiattribute beim Kopieren nicht ändern.

Wenn das Kopieren abgeschlossen ist muss die Ramdisk mit `umount mountpoint` wieder ausgehängt werden. Danach muss die Ramdisk mit `gzip -9 dateiname` komprimiert werden. Um die Ramdisk später wieder mounten zu können um die Ordnerstruktur zu ändern muss sie zuerrst mit `gunzip dateiname` entpackt werden!

2.3 Kernel kompilieren (http://www.kernel.org)

Kernelquellcodeversion 2.4.19 wurde als Grundlage für den GME-Linux Kernel verwendet. Um Kerneloptionen ein- oder auszuschalten muss die Datei `.config` editiert werden. Dies kann mit Hilfe verschiedener Hilfsprogramme ermöglicht werden. Standardmäßig sollten die Kernelquell-dateien installiert sein, falls dies nicht der Fall ist, so kann man sich die Dateien von http://www.kernel.org runterladen. Um die Datei `.config` zu editieren, kann man unter der KDE in Linux das Kontrollzentrum öffnen den Eintrag System auswählen und dann auf **Einrichtung des Linux-Kernels** klicken. Daraufhin sollte folgendes Fenster erscheinen.

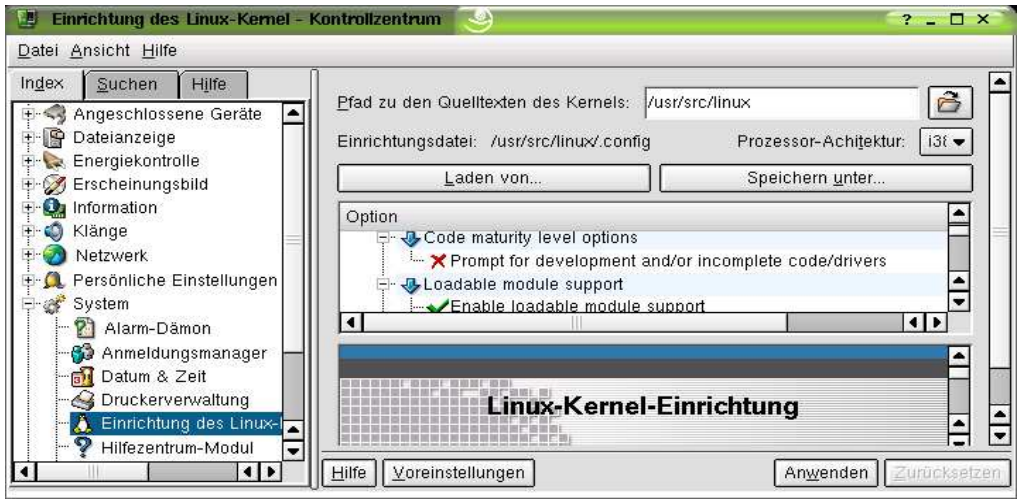


Abb. 2.1: Einrichtung des Linux-Kernels

Eine andere (unkomfortable) Möglichkeit die Konfigurationsdatei zu editieren erhält man durch die Eingabe von `make config` in der Textkonsole (muss vom Pfad ausgeführt werden in dem sich die Kernelsources befinden – Standard: */usr/src/linux*). Hier wird jedoch Punkt für Punkt der Kernelkonfigurationsdatei abgefragt. Dies führt zu einer zeitaufwendigen Einrichtung, da verhältnismäßig wenige Einstellungen notwendig sind um einen funktionierenden Kernel zu erhalten.

Nächste Möglichkeit wäre `make menuconfig` in der Textkonsole einzugeben. Dies ermöglicht uns die Konfigurationsdatei komfortabel in der Konsole einzurichten.

Eine weitere Möglichkeit ist ebenfalls in der Textconsole `make xconfig` einzugeben. Dazu muss jedoch der Xserver aktiv sein und man muss Schreibrechte auf diesem besitzen. Falls der Xserver aktiv ist und man jedoch keine Schreibrechte besitzt, so kann man sich durch den Befehl `ssh -X root@localhost` als Systemadministrator Schreibrechte auf einem von einem anderen User gestarteten Xserver verschaffen.

Anmeldung als Systemadministrator wird empfohlen, da standardmäßig nur der Benutzer root Schreibrechte auf die Konfigurationsdatei hat.

Nachdem man alle Einstellungen getroffen hat ist die Eingabe des Befehls `make dep` erforderlich. Dieser Befehl führt zur verlinkung aller benötigten Dateien. Wenn der Computer diesen Befehl abgearbeitet hat muss `make bzImage` eingegeben werden, da der Kernel erwartungsgemäß zu groß für ein zImage wird. Durch den Befehl make bzImage wird der Kernel kompiliert. Wenn der Kernel für die Rechnerarchitektur `i386` erstellt wurde ist nach der Kompilierung der fertige Kernel im Unterverzeichnis `arch/i386/boot` zu finden. Der Kernel heißt dann bzImage.

2.4 Busybox kompilieren (http://www.busybox.net)

Busybox ist ein Programm dass viele kleine Unixanwendungen in sich vereint. Dieses Programm ist jedoch so programmiert, dass es so wenig Speicherplatz wie möglich belegt.

Anhand des Befehls `make menuconfig` lässt sich ein Textkonsolen basiertes Konfigurations-menü öffnen.

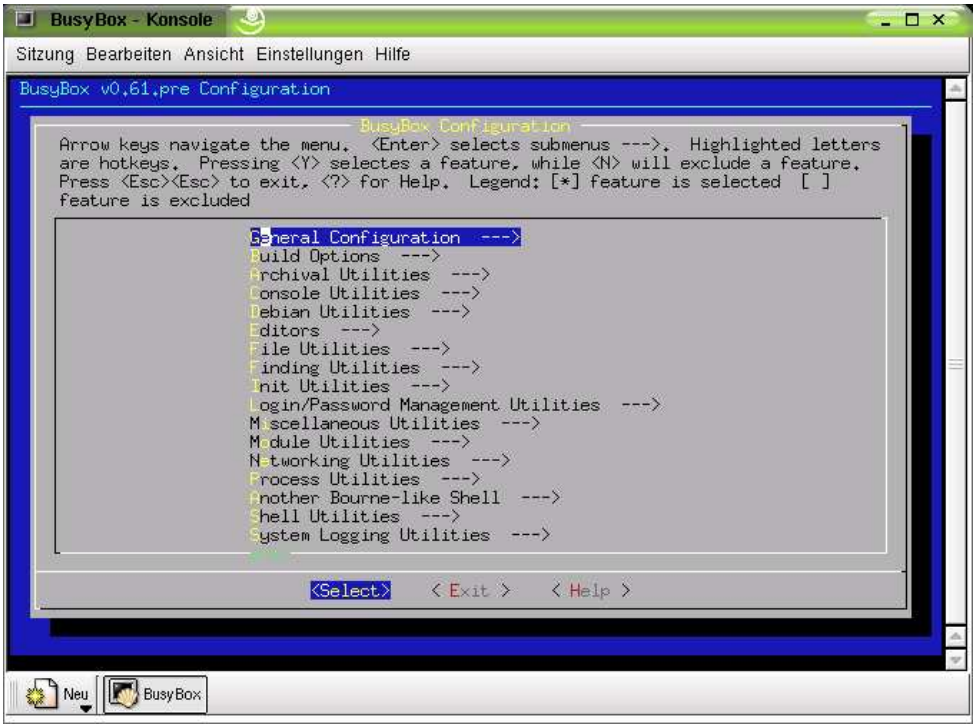


Abb. 2.2: Busybox Konfigurationsprogramm

Eine der wichtigsten Optionen ist die, dass Busybox sich über den Befehl `busybox --install` auf dem GME-Linux installieren lässt. Diese Option ist im Untermenü *General Configuration* zu finden und heißt *Support –install [-s] to install applet links at runtime.*

Nachdem man alle gewüschten Optionen eingestellt hat muss man den Befehl `make dep`, dieser Befehl führt zur verlinkung benötigter Dateien. Nach Abschluss des vorigen Befehls kann die Busybox durch den Befehl `make` kompiliert werden. Die ausführbare Datei heißt busybox und muss nun in den Ordner */bin* auf GME-Linux kopiert werden. Die Installation erfolgt über den Befehl `busybox --install`.

2.4.1 Busybox einstellungen

Alle hier nicht aufgelisteten Einstellungen bleiben unverändert.

General Configuration	->	Support --install [-s] to install applet links at runtime
Build Options	->	Build BusyBox as a static binary (no shared libs)
Console Utilities	->	dumpkmap; loadkmap
Miscellaneous Utilities	->	update
Networking Utilities	->	hostname; ifconfig; ping

2.5 Tinylogin kompilieren (http://tinylogin.busybox.net)

Tinylogin dient dazu die Loginprogramme zu ersetzen (login, su, ...). Tinylogin ist ähnlich wie Busybox aufgebaut. Die Konfiguration erfolgt jedoch durch direkte änderung der Datei Config.h im Tinylogin Quellcodeordner. Die Datei muss mit einem Texteditor geöffnet werden, dort befolgt man die Anleitung und editiert nur die Punkte die man benötigt.

Durch den Befehl `make` wird Tinylogin kompiliert. Die bei der GME-Linux bestehenden verweise auf Tinylogin müssen nicht gelöscht werden und können weiterhin verwendet werden. Es ist lediglich die ausführbare Datei tinylogin in den Ordner */bin* auf GME-Linux zu kopieren.

2.6 Setzen eines Passwortes

Um ein Passwort für einen Benutzer zu setzen, muss die Datei passwd die sich im Ordner /etc befindet editiert werden. Zuerst muss das gewünschte Passwort codiert werden. Dies kann entweder mit `grub-md5-crypt` (grub muss installiert sein!) oder mit dem folgenden Befehl durchgeführt werden (Perl muss dazu installiert sein!):

```
perl -e 'print crypt(passwort, Code)' > passwort.txt
```

Anstatt Passwort ist natürlich das gewünschte Passwort einzutragen (Achtung auf die Groß- und Kleinschreibung). Der Ausdruck > passwort.txt veranlässt die Erstellung einer Textdatei Namens passwort.txt. Der Name der Textdatei ist frei wählbar. Wenn der Ausdruck zur Erstellung der Textdatei weggelassen wird, so erscheint das codierte Passwort auf dem Bildschirm.

2.6.1 Setzen des Rootpasswortes

In der Datei passwd muss die Zeile `root::0:0:root:/root:/bin/ash` editiert werden. Das codierte Passwort wird nun folgendermaßen eingefügt:
`root:`*`codiertesPasswort`*`:0:0:root:/root:/bin/ash`.
Beispiel für das Format eines codierten Passwortes:
`CoM04niiEUU9U` oder `$1$s19fe/$RXaH9tKB97GZr0mvGoTqK1`

2.7 Vorbereitung der Textkonsole

Der nachfolgende Text beschreibt verschiedene Versuche die Textkonsole anzupassen. Diese Versuche waren jedoch nicht 100% erfolgreich.

ACHTUNG: Jeder Versuch dient als Grundlage für jeden weiteren.

2.7.1 Erster Versuch

Bei der verwendeten Version der Busybox muss eine Veränderung an der ash vorgenommen werden, damit das Einloggen richtig funktioniert. Es muss die Standardversion der ash die bei der Embedded Linux Vorlage verwendet wird vorerst beibehalten. Diese Datei in ashold umbennen.

Als nächstes muss ein Skript geschrieben werden. Dazu einen Texteditor öffnen und folgendes eintragen:

```
#!/bin/ashold
busybox ash
```

Der Name der Textdatei muss `ash` sein! Um sicherzustellen dass dieses Skript richtig ausgeführt werden kann sollte folgender Befehl ausgeführt werden: `chmod +x ash`

2.7.2 Zweiter Versuch

Aufgrund der zuvor beschriebenen Änderungen lassen sich die Skripten die automatisch starten sollen nicht ausführen. Um dieses Problem zu lösen und gleichzeitig den möglichen Gebrauch der neuen und komfortablen ash dem Benutzer zu ermöglichen wurde das obige Skript folgendermaßen verändert.

```
#!/bin/ash
busybox ash
```

Das Skript ist unter dem Namen userash abzuspeichern. Die alte ash muss statt dem Namen ashold den Namen ash beibehalten. Danach muss in der Datei */etc/passwd* folgende Änderung vorgenommen werden (Bsp. für User root):

`root:`*`codiertesPasswort`*`:0:0:root:/root:/bin/ash` auf

`root:codiertesPasswort:0:0:root:/root:/bin/`*`userash`* ändern.

2.7.3 Dritter Versuch (aktuelle Konfiguration)

Die im zweiten Versuch beschriebenen Änderungen haben die automatisch startenden Skripten wieder ausführen lassen, jedoch konnte nichts mehr mit scp (siehe 2.7) zu GME-Linux kopiert werden. Es war jedoch möglich von GME-Linux scp auszuführen, da dies jedoch nicht das Ziel ist mussten Änderungen durchgeführt werden.

Der Inhalt der Datei /bin/userash sollte folgenden Eintrag beinhalten:

```
#!/bin/ash
busybox ash
echo You returned to the parent ash!
```

In der Datei */etc/passwd* muss sichergestellt sein, dass bei den Usern als Konsole ash eingetragen ist. Beispiel für Benutzer root:

`root:codiertesPasswort:0:0:root:/root:`*`/bin/ash`*

Um nach dem Einloggen die komfortable Ash zu erhalten muss man `ua` oder `userash` eintippen. Damit der Befehl ua später funktioniert muss man vom Ordner `/bin` aus `ln -s userash ua` eingeben. Dadurch wird ein Link namens ua erstellt der auf die Datei userash verweist.

2.7.4 Begrüßung beim Einloggen

Beim Einloggen soll eine Nachricht erscheinen die den Benutzer auf die Userash aufmerksam macht. Dazu muss in der Datei */etc/login.defs* die Zeile

```
MOTD_FILE       /etc/motd
```

hinzugefügt werden. Danach kann die Datei */etc/motd* erstellt werden. In dieser Datei steht der Begrüßungstext. Momentaner Begrüßungstext lautet:

```
Type ua or userash to get the better Shell!
```

In der Datei `/etc/issue` bzw. `/etc/issue.net` kann ein Informationstext für das Einloggen über die serielle Schnittstelle eingegeben werden, jedoch muss zuerst in der Datei `/etc/init.d/bootmisc.sh` der folgende Teil (ähnlich folgendem Abschnitt) mit dem Rautesymbol auskomentiert werden:

```
...
if [ "$EDITMOTD" != no ]
then
#  echo "Embedded Linux Version $(cat /etc/config/version)" >/etc/issue
#  cp /etc/issue /etc/issue.net
fi
```

2.8 Einbindung des SSH-Deamons (sshd)

Um den sshd in GME-Linux einzubinden müssen Bibliotheken, Verzeichnisse und Konfigurationsdateien in die Verzeichnisstruktur des GME-Linux eingebunden werden.

Folgende Dateien bzw. Verzeichnisse müssen in das Verzeichnis */lib* kopiert werden um die Funktionsfähigkeit des sshd sicherzustellen:

```
ld-linux.so.2 -> ld-2.2.5.so              ld-2.2.5.so
libasn1.so.5 -> libasn1.so.5.0.0          libasn1.so.5.0.0
libcom_err.so.1 -> libcom_err.so.1.1.0    libcom_err.so.1.1.0
libgssapi.so.1 -> libgssapi.so.1.2.4      libgssapi.so.1.2.4
libkrb5.so.17 -> libkrb5.so.17.1.2        libkrb5.so.17.1.2
libpam.so.0 -> libpam.so.0.76             libpam.so.0.76
libpam_misc.so.0 -> libpam_misc.so.0.76   libpam_misc.so.0.76
libroken.so.9 -> libroken.so.9.5.3        libroken.so.9.5.3
libz.so.1 -> libz.so.1.1.4                libz.so.1.1.4

libc.so.6                                 libcrack.so.2
libcrack.so.2.7                           libcrypt.so.1
libcrypto.so.0.9.6                        libdb-4.0.so
libdl.so.2                                libgcc_s.so.1
libm.so.6                                 libnsl.so.1
libnss_compat.so.2                        libnss_dns.so.2
libnss_files.so.2                         libresolv.so.2
libutil.so.1                              libxcrypt.so.1
libxcrypt.so.1.1.0                        security/
```

Alle Einträge mit einem Pfeil (->) bedeuten folgendes: linkname -> zieldatei. Der gegenüberstehende Dateiname steht ebenfalls für die Zieldatei.

Unabhängig davon wo die Bibliotheken auf dem Quellsystem zu finden sind (*/lib* oder auch */usr/lib*) können sie auf GME-Linux alle in den Ordner */lib* kopiert werden. Dies erleichtert spätere Änderungen.

Folgende Dateien bzw. Verzeichnisse müssen in das Verzeichnis */etc* kopiert werden:

```
pam.d/                                    ssh/
security/                                 krb5.conf
services                                  protocols
nsswitch.conf                             host.conf
hosts                                     hosts.allow
hosts.deny                                hosts.equiv
hosts.lpd
```

Alle aufgelisteten Dateien und Verzeichnisse befinden sich im Ordner /etc auf dem Entwicklersystem.

An der Datei nsswitch.conf müssen gegebenenfalls Änderungen vorgenommen werden:

```
passwd: compat    und   group: compat     durch
passwd: files     und   group: files      ersetzen.
```

Folgende Dateien bzw. Verzeichnisse müssen auf GME-Linux kopiert werden:

```
/usr/bin/ssh-keygen                       /usr/lib/ssh/
/usr/sbin/sshd/                           /var/lib/empty/
/var/lib/sshd/                            /usr/bin/scp
```

Um von einem entfernten Computer mittels scp Dateien zu/von GME-Linux zu kopieren, muss unbedingt die Datei /usr/bin/scp in GME-Linux integriert werden.

Damit das Einloggen über ssh funktioniert, sollte der Benutzer root ein Passwort haben.

Grundsätzlich ist es möglich ssh so einzurichten dass die Anmeldung eines Benutzers ohne Passwort möglich ist. Davon ist jedoch aus Sicherheitsgründen abzuraten.

2.8.1 Fish

Fish ist ein Programm mit dem man vom Konqueror aus auf einen anderen Computer über das SSH-Protokoll zugreifen kann (SSH-Client). Befehlszeile im Konqueror Adressfeld:

```
fish://benutzername@IPadresse
```

Auf dem Computer auf den zugegriffen werden möchte muss Perl installiert sein, da Perl jedoch zu viel Speicherplatz auf benötigt, kann es nicht in GME-Linux integriert werden.

2.9 Konfiguration des SSHD und des Netzwerkes

Bevor der sshd zum ersten mal gestartet werden kann müssen sshkeys generiert werden. Dies geschieht mit folgenden Befehlen (auf GME-Linux ausführen!).

```
ssh-keygen -t rsa1 -b 1024 -f /etc/ssh/ssh_host_key -N ''
ssh-keygen -t dsa -b 1024 -f /etc/ssh/ssh_host_dsa_key -N ''
ssh-keygen -t rsa -b 1024 -f /etc/ssh/ssh_host_rsa_key -N ''
```

Der nächste Schritt besteht darin die Netzwerkverbindung einzurichten. Dazu müssen loopback device (lo) und Netzwerkkarte (eth0) konfiguriert werden.

```
ifconfig lo 127.0.0.1
ifconfig eth0 172.16.116.200        172.16.116.200 ist nur ein Beispiel!
```

Nun kann der sshd durch den Befehl *sshd* gestartet werden.
Um die Konfiguration der Netzwerkkarte und den Start des SSH-Deamons beim Bootvorgang automatisch zu initiieren mussen folgende Einstellungen vorgenommen werden.

Im Ordner */etc/init.d* muss jeweils ein Skript für sshd und Netzwerkkarte erstellt werden. Das Skript für die Konfiguration der Netzwerkkarte heißt *network* und das Skript für den sshd heißt *sshstart*.

In der Datei network müssen folgende Einträge vorhanden sein:

```
#!/bin/sh
#Netzwerkkonfiguration
ifconfig lo 127.0.0.1
ifconfig eth0 $(cat /etc/config/ip)
```

In der Textdatei `/etc/config/ip` befindet sich die IP-Adresse die das GME-Linux der ersten Netzwerkkarte (eth0) zuweisen soll. Im Ordner `/etc/config` lässt sich auch der Hostname ändern, indem man die Datei `hostname` editiert.

In der Datei sshstart müssen folgende Einträge vorhanden sein:

```
#!/bin/sh
#SSH-Deamon Startskript
sshd
```

Wenn diese Skripten erstellt wurden muss der Befehl *chmod +x network sshstart* ausgeführt werden.

Nun muss der Befehl `ln -s ../init.d/network S10network` und `ln -s ../init.d/sshstart S11sshstart` eingegeben werden. Diese Befehle müssen vom Ordner */etc/rc2.d* ausgeführt werden (dort sollen sich auch die 2 Skripten befinden).

S10 und S11 heißt: S=Start 10/11=Nummer der Abfolge

2.10 Ersten Bootvorgang vorbereiten

Das Embedded System wird durch einen PC simuliert. Linux wird von Zipdiskette gestartet. Auf der Zipdiskette müssen min. 2 Partitionen mit `cfdisk` oder einem ähnlichen Programm erstellt werden. Um den Linuxkernel booten zu können, wird zuerst DOS gestartet und der Linuxkernel mit dem Programm `LOADLIN` gestartet. Daher muss die 1. Partition eine DOS-Partition sein. Die 2. Partition ist eine Linux-Partition. Nach erstellen der Partitionen ist das Dateisystem zu erstellen. Mit dem Befehl `mkfs.dateisystem /dev/hdd1` kann ein Dateisystem auf einer Partition erstellt werden.

Bsp.: In diesem Beispiel wird auf einer Zipdiskette ein Dateisystem erzeugt. Die Zipdiskette ist hier am Secondary IDE Port als Slave angeschlossen (hdd). Der verwendete Parameter gibt die Position der Partition an.

`mkfs.msdos /dev/hdd1`	hdd1: Auf der 1. Partition wird ein DOS Dateisystem erzeugt.
`mkfs.minix /dev/hdd2`	hdd2: Auf der 2. Partition wird ein Minix Dateisystem erzeugt.

Sollte das Ziplaufwerk wo anders angeschlossen sein so ist dies statt `/dev/hdd` einzutragen.

Primary IDE Master	*/dev/hda*
Primary IDE Slave	*/dev/hdb*
Secondary IDE Master	*/dev/hdc*
Secondary IDE Slave	*/dev/hdd*

Um bei einer Unterbrechung der Spannungsversorgung das Dateisystem zu Schützen wurde eine 3. Partition erstellt. Diese Partition dient als Arbeitspartition, auf dieser kommt die Datenerfassungssoftware zum Einsatz. Dies hat zur Folge, dass im Falle eines Fehlers nicht alle Partitionen neu erstellt werden müssen und dadurch auch nicht die ganze Software neu auf den Datenträger gespielt werden muss.

Die Arbeitspartition (`/dev/hdd3`) wird im Ordner /mnt gemountet. Dazu muss die Datei `/etc/fstab` auf GME-Linux editieren und um folgenden Eintrag ergänzen.

```
/dev/hdd3       /mnt    minix   defaults        2       2
```

2.10.1 Loadlin – Einstellungen

Loadlin benötigt einige Startparameter, diese können mit Hilfe einer Batchdatei automatisch eingegeben werden. Wenn man die Batchdatei autoexec.bat nennt, so wird automatisch nach dem DOS gestartet wurde Loadlin ausgeführt.

Batchdatei für System mit Ramdisk:

```
ECHO Start Embedded Linux for GME ...
loadlin kernelname console=ttyS0,115200 initrd=ramdiskname root=/dev/x
ECHO Embedded Linux stopped or failed!
```

Batchdatei für System ohne Ramdisk:

```
ECHO Start Embedded Linux for GME ...
loadlin kernelname console=ttyS0,115200 root=/dev/x
ECHO Embedded Linux stopped or failed!
```

Anstatt *x* muss das entsprechende Gerät eingetragen werden.

2.11 Texteditior einbinden

Um Fehler in Skripten zu korrigieren oder Änderungen an diesen vorzunehmen, ist ein Texteditor auf GME-Linux praktisch. Mit der Busybox wurde auch der Texteditor *vi* kompiliert, der Umgang mit diesem Editor ist jedoch nicht leicht. Deswegen wurde der Texteditor *joe* eingebunden.

Zusätzliche Bibliotheken und Dateien die *joe* benötigt sind:

/usr/bin/joe */etc/joerc*
/lib/libncurses.so.5 */lib/libncurses.so.5.2*

Alle Dateien sind auf GME-Linux in die gleichen Ordner zu kopieren in denen sie sich auf dem Entwicklersystem befinden.

Anmerkung: JOE ist nicht zwingend notwendig für die fertige GME-Linux Version, da der Texteditor vorrangig für Konfigurationstätigkeiten genutzt wird.

2.12 Erstellen einer blockorientierten Datei

Eine blockorientierte Datei wird zum Beispiel zum einbinden eines Gerätes benötigt. Bevor man ein Gerät mounten kann, muss zuerst eine Datei mit dem Gerätenamen im Ordner `/dev` vorhanden sein.

Beispiel:

Um eine Zipdiskette mounten zu können muss im Verzeichnis `/dev` eine blockorientierte Datei namens hdd vorhanden sein. Wenn diese Datei auf dem GME-Linux nicht vorhanden sein sollte so muss man wissen welche Parameter diese Datei hat. Dies schaut man am besten am Entwicklersystem mittels `ls -al /dev/hdd` nach. Die Ausgabe sollte folgendermaßen aussehen:

b*rw-rw-rw- 1 root disk **22, 64** 2002-09-09 22:24 /dev/hdd*

Um nun eine blockorientierte Datei zu erstellen die den Zugriff auf dieses Gerät ermöglicht, zu erstellen muss man `mknod /dev/hdd b 22 64` eingeben.

2.13 Befehle

`which programmname`	Lokalisiert den Pfad zum angegebenen Programm
`ldd /pfad/programm`	Spürt mit dem Programm verlinkte Bibliotheken auf
`cp -av quelle ziel`	Kopiert Dateien/Verzeichnisse + gesetzte Attribute
`strace -f -e trace=open -o tracefile.txt programm`	Zeichnet Programmaufrufe auf und schreibt diese in die Datei tracefile.txt (ist frei wählbar).
`ifconfig device ip`	Konfiguriert ein device (z. B. Eth0) und weist diesem device eine gewünschte IP-Adresse zu.
`ln -s quelle ziel`	Macht eine symbolische Verknüpfung.
`chmod +x datei`	Ändert die Zugriffsrechte auf die Datei so dass jeder sie ausführen kann.
`mknod /dev/geraet parameter1 parameter2 parameter3`	Erstellet eine blockorientierte Datei. Verwendung, siehe 2.12.

2.14 Dateisystemfehler

2.14.1 Einbinden von fsck

Fsck bedeutet file system check. Wie der Name schon sagt überprüft und repariert (sofern möglich) Dateisysteme. Da die verwendeten Linuxpartitionen alle ein Minixdateisystem besitzen müssen nur die Dateien kopiert werden die zur Überprüfung einer Minixpartition benötigt werden.

Dateien die auf GME-Linux kopiert werden müssen:

/sbin/fsck
/sbin/fsck.minix
/lib/libext2fs.so.2
/lib/libext2fs.so.2.4
/lib/libcom_err.so.2
/lib/libcom_err.so.2.0

Auf GME-Linux sollten die Dateien *libcom_err.so.1* und *libcom_err.so.1.1.0* bereits vorhanden sein, da der sshd diese benötigt. Da aber eine neuere Version auf GME Linux kopiert wierd (libcom_err.so.2 und libcom_err.so.2.0) werden die zuvor genannten nicht mehr benötigt, aber es muss eine Datei namens *libcom_err.so.1* existieren die auf die Datei *libcom_err.so.2.0* verweist!

2.15 Autostart der Datenerfassungssoftware

Die Datenerfassungssoftware befindet sich standardmäßig im Ordner */mnt*. Um den Konfigurationsdateien den Ordnernamen der Software mitzuteilen, müss die Dateien */etc/config/gmesoftdir* und */etc/fstab* verändert werden.

Inhalt von */etc/config/gmesoftdir* ist der Ordnername: mnt

Ergänzung von */etc/fstab* – dort muss der mountpoint definiert sein:

```
/dev/hdd3       /mnt    minix   defaults        2       2
```

Zum automatischen starten nach dem Booten muss ein Startskript im Ordner */etc/init.d/* und ein link auf diese Datei im Ordner */etc/rc2.d/* vorhanden sein.

Inhalt vom Startskript in */etc/init.d/* (Name der Datei z.B. startgme):

```
#!/bin/ash
#2 Sekunden warten, dann GME starten; Wenn GME tot, 2 Sekunden warten,
#dann GME starten
while sleep 2
 do
  echo "Starting GME"
  cd /$(cat /etc/config/gmesoftdir) && ./startgme
 done &
```

2.16 Loginmöglichkeit über tty2

Um die Loginmöglichkeit anzubieten muss das Programm */sbin/mingetty* auf GME-Linux kopiert werden. In der Datei */etc/inittab* auf GME-Linux muss folgender Eintrag hinzugefügt werden:

```
2:123:respawn:/sbin/mingetty tty2
```

Um die deutsche Tastatur auf GME-Linux nutzen zu können muss zuerst mittels des Programms `dumpkmap > de_keymap.bin` auf einem Computer ausgeführt werden auf dem die deutsche Tastatur installiert ist die Datei *de_keymap.bin* erzeugt werden. Um diese Datei nutzen zu können muss sie auf GME-Linux kopiert und der Befehl `loadkmap < de_keymap.bin` ausgeführt werden.

3 Datenerfassungssoftware

3.1 Allgemein

Die GME-Software dient zur Aufbereitung, Filterung und Archivierung der vom Mikrokontroller erhaltenen Messdaten zur weiteren Bearbeitung. Die archivierten Daten werden in einem datumsspezifischen Datenfile gesichert, welches sich im gleichen Verzeichnis wie die Software befindet.

3.2 Programmstart

Die Software wird über das Shell-Script „startgme“ in der Textkonsole aufgerufen.
Die Parametrierung für diverse Einstellmöglichkeiten erfolgt über die Textdatei „config.dat“.

Auszug aus config.dat

```
#Konfigurationsfile der GME

#Bitte nachstehend die verwendeten Parameter eintragen:
#Abstand der beiden Sensoren zueinander (in cm)
ABSTAND=100.0

#Zeitdifferenz bevor die Messung unterbrochen wird (in Sek)
TIMEOUT=5.0

#Anzahl der Zeitabgleiche mit dem Mikrokontroller pro Tag
ABGLEICH=1

#Nachstehende Parameter nur verändern bei geänderter Hardware
#Eingangsgerät der Sensoren (Gerät aus /dev/) (Standard=ttyS1)
SENSORINPUT=ttyS1

#Baudrate der Seriellen Kommunikation mit dem Mikrocontroller
#Nur genormte Baudraten für serielle Übertragung benutzen (Bx)
BAUDRATE=B9600
```

Die Parameter sind:

Softwarebeeinflussend:

- ABSTAND=x.x
  Abstand der beiden Sensoren zueinander; Dieser Wert sollte nach Möglichkeit zwischen 100 und 120 cm liegen, kann jedoch diese auch über- bzw. unterschreiten. Die Eingabe erfolgt in cm.

- TIMEOUT=x.x
  Mit dem Timeout gibt man an, nach wieviel Sekunden eine Laufende Messung abgebrochen werden soll, wenn kein weiteres Signal erfolgt. Die Eingabe erfolgt in Sekunden.

- ABGLEICH=x
  Da die Frequenz des Mikrokontrollers nicht definiert ist, bzw. im Betrieb schwanken kann, ist es sinnvoll einen Zeitabgleich zwischen Mikrokontroller und Rechner durchzuführen. Es sollte wenigstens 1 Abgleich pro Tag erfolgen. Die Eingabe erfolgt in Abgleiche pro Tag.

Hardwarebeeinflussend:

- SENSORINPUT=ttyx
  Der Mikrokontroller ist über eine Kommunikationsschnittstelle mit dem Rechner verbunden. Die Software ist derzeit für eine Kommunikation über die Serielle Schnittstelle ausgelegt. Die Eingabe erfolgt über ein Gerät im Verzeichnis /dev/

ttyS0	Serielle Schnittstelle 1
ttyS1	Serielle Schnittstelle 2
usw.	

- BAUDRATE=x
  Kommunikationsgeschwindigkeit zwischen Rechner und Mikrokontroller in Form der Baudrate. Die Eingabe erfolgt über einen der folgenden Parameter:

300	300bps
600	600bps
1200	1200bps
1800	1800bps
2400	2400bps
4800	4800bps
9600	9600bps
19200	19200bps
38400	38400bps
57600	57600bps
115200	115200bps
230400	230400bps

3.3 Ausgabedateien

3.3.1 Datenfile

Das Datenfile befindet sich im gleichen Verzeichnis wie die Software, und hat den Dateinamen „data_Tag-Monat-Jahr.txt“, wobei sich Tag, Monat und Jahr auf das numerische Datum zum Zeitpunkt des Programmstartes bezieht.
Z.B. die GME-Software wurde am 7 März 2003 gestartet, so lautet der Dateiname des Datenfiles „data_07-03-2003.txt“.

Auszug aus einem möglichen Datenfile:

```
14    02    2003  12    07    40    3637  3874  4146  >
14    02    2003  12    17    17    2945  2955  2965  >
14    02    2003  12    23    47    2983  3217  3463  >
14    02    2003  12    35    26    4484  4486  4486  >
14    02    2003  12    50    17    5216  5715  6284  >
14    02    2003  13    03    04    2942  2944  2947  >
```

Jede Zeile enthält die Daten einer abgeschlossenen Messung , und enthält die Daten in Reihenfolge wie oben angeführt:
Tag, Monat, Jahr, Stunde, Minute, Sekunde, langsamste gemessene Geschwindigkeit, durchschnittliche Geschwindichkeit, schnellste gemessene Geschwindichkeit und Richtung.

Tag, Monat, Jahr, Stunde, Minute und Sekunde beziehen sich auf den Zeitpunkt der Messung. Die Geschwindichkeiten sind in mm/s angegeben, die Richtung wird als Pfeil (< bzw. >) angegeben.

3.3.2 Logfile

Das Logfile befindet sich im gleichen Verzeichnis wie die Software, und hat den Dateinamen „log_Tag-Monat-Jahr.txt“, wobei sich Tag, Monat und Jahr auf das numerische Datum zum Zeitpunkt des Programmstartes bezieht.
Z.B. die GME-Software wurde am 7 März 2003 gestartet, so lautet der Dateiname des Logfiles „log_07-03-2003.txt“.

Auszug aus einem möglichen Logfile:

```
Abgeglichen mit mC; 1Sek entspricht 1015 Takten
1027  1.011823    0
2451  2.414778    1
1024  1.008867    0
2482  2.445320    1
1021  1.005911    0
2760  2.719212    1
1027  1.011823    0
32767 32.282757   1
14.02.2003  13:41:48  :1. und letzter Sensor ungleich; Kein Sensor
ausgefallen; Richtung A->B
```

1. Zeile:
Nach jedem Zeitabgleich (siehe 3.2) erfolgt eine Meldung die angibt wieviel Takte des Mikrokontrollers einer Sekunde entsprechen.
2. - 9. Zeile:
Jeder Messwert wir wiedergegeben durch den vom Mikrokontroller erhaltenen Wert, den sich daraus ergebenden Wer in Sekunden und dem angesprochenen Sensor.
10. Zeile:
Mit Datum und Zeit des ersten Messwertes wird der Filterungsprozess für die Richtung dargestellt.

3.4 Programmaufbau

Die Software ist in mehrere Unterprogramme aufgeteilt die nachstehend ausführlich erklärt werden. Die Software ist in C++ geschrieben und für einen Linux-Compiler ausgelegt.
Die Software umfasst folgende Programmteile:

3.4.1 Halte Signal 1 Sekunde

Syntax: `int hs1s(int serialpointer);`
Das Programm erhält über `serialpointer` die Adresse der Kommunikationsschnittstelle und setzt die Ready-to-Send Leitung für eine Sekunde auf High. Diese Funktion ist notwendig um den Zeitabgleich mit dem Mikrokontroller zu ermöglichen.
Rückgabewert: -1 bei Erfolg.

3.4.2 Empfange Zaehler vom Microcontroller

Syntax: `int getticks(int serialpointer);`
Das Programm erhält über `serialpointer` die Adresse der Kommunikationsschnittstelle und ließt 2 Zeichen von dieser, um nach hs1s (siehe 3.4.1) die vom Mikrokontroller gesendeten Takte für den Zeitabgleich zu erhalten. Die Wertigkeit der Zeichen ist 256 für das 1. Zeichen und 1 für das 2. Zeichen. Die Werte der einzelnen Zeichen richten sich nach der ASCII-Tabelle;
Rückgabewert: Anzahl der Takte.

3.4.3 Neues File erstellen

Syntax: `int getfilename(char *dataname, char *logname);`
Das Programm fragt das aktuelle Datum ab und erstellt dann ein Datenfile und ein Logfile mit den Dateinamen: „data_Tag-Monat-Jahr.txt“ bzw log_Tag-Monat-Jahr.txt (siehe auch 3.3).
Diese Dateinamen werden auf die Adressen von `dataname` bzw. `logname` geschrieben.
Rückgabewert: -1 bei Erfolg.

3.4.4 Richtungsfilter

Syntax: `int Filter(int sensorfield[100], int counter, int &vzr0, int &vzr1, struct tm *b, char *logname);`
`sensorfield` enthält entsprechd den aufgenommenen Messwerten die angesprochenen Sensoren in chronologischer Reihenfolge.
`counter` gibt die anzahl der empfangenen Messdaten an.
`vzr0` und `vzr1` geben an wie oft eine Richtung als sicher angenommen wurde um daraus eine Vorzugsrichtung abzulesen. Diese Werte sind mit Adressen übergeben um sie nach verlassen des Programmes weiter zu erhalten.
`tm` enthält Zeit und Datum des ersten empfangenen Messwertes für die Ausgabe im Logfile.
`logname` Zeiger zum Dateinamens des Logfiles.
Zuerst stellt das Programm fest ob ein Sensor ganz ausgefallen ist. Ist dies nicht der Fall, wird als nächstes Festgestell ob der 1. und der letzte Sensor ungleich sind. Weiters wird noch festgestellt ob weitere Sensoren ausfallen sind.
Wenn eine Richtung sicher ist, wird die dementsprechende Vorzugsrichtung in ihrer Wertigkeit erhöht. Eine Richtung ist dann sicher wenn entweder kein Sensor ausgefallen ist, oder wenn zwischen 1. und letztem Sensor ein oder mehrere Sensoren ausgefallen sind, jedoch aufgrund des 1. und letzten Sensors gesagt werden kann dass er sich in Vorzugsrichtung befindet.
Sollte die Richtung aus dem Messwerten nicht klar ersichtlich sein (z.B. 1. und letzter Sensor sind gleich) wird die Vorzugsrichtung angenommen. Ist die Vorzugsrichtung noch nicht sicher werden beide Richtungen angenommen.
Rückgabewert: -1 wenn ein Sensor tot ist.
0 wenn Richtung 0 ist
1 wenn Richtung 1 ist
2 wenn Richtung unsicher ist

3.4.5 Auswertung

Syntax: *int Auswerte(float timefield[100], int sensorfield[100], int counter, int Richtung, float Abstand, struct tm *b, char *dataname, int kt);*
timefield enthält entsprechd den aufgenommenen Messwerten die gezählte Zeit in chronologischer Reihenfolge.
sensorfield enthält entsprechd den aufgenommenen Messwerten die angesprochenen Sensoren in chronologischer Reihenfolge.
counter gibt die anzahl der empfangenen Messdaten an.
Richtung gibt die Richtung für die Auswertung an.
Abstand gibt den Abstand zwischen den beiden Sensoren an
tm enthält Zeit und Datum des ersten empfangenen Messwertes für die Ausgabe im Datenfile.
dataname Zeiger zum Dateinamens des Datenfiles.
kt Kommentarwert. Bei 0 und 1 kein Kommentar bei 2 Kommentar „Richtung unbekannt“ sonst Kommentar „Fehler“.
Zunächst wird jeder Wert bei dem die Richtung stimmt (Vergleich *Richtung sensorfield*) genommen und in eine Geschwindigkeit umgerechnet. Dies geschieht mit der Formel: (*Abstand/*100)/*timefield[x]*.
Mit *Abstand*/100 erhält man den Abstand in Metern (cm -> m) und mit der Division durch die entsprechende Zeit (*timefied[x]*) erhält man die Geschwindigkeit in m/s.
Jeder errechnete Geschwindigkeitswert wird gezählt und mit den anderen Geschwindigkeitswerten aufsummiert. Nach der Berechnung des letzten Geschwindigkeitswertes wird die Summe aller Geschwindichkeitswerte durch die Anzahl der Geschwindichkeitswerte dividiert um die durchschnittliche Geschwindichkeit zu ermitteln.
Zudem Wird nach jeder errechneten Geschwindichkeit kontrolliert ob diese eine maximale oder minimale Geschwindigkeit dieser Messung ist.
Nach Abschluss sämtlicher Berechnungen wird das Datum und die Zeit in Zeichenfolgen umgewandelt und zusammen mit den 3 Geschwindigkeitswerten (minimale, durchschnittliche und maximale Geschwindichkeit) im mm/s und der Richtung ins Datenfile geschrieben.
Rückgabewert: -1 bei Erfolg.

3.4.6 Hauptprogramm

Syntax: *int main(int argc, char *argv[]);*
Das Programm kontrilliert zuerst ob alle 5 Übergabeparameter (siehe 3.2) vorhanden sind und speichert diese in geeigneten Formaten. Sind mehr oder weniger als diese Parameter vorhanden gibt das Programm eine Fehlermeldung aus und beendet sich.
Danach werden die Dateinamen für das Datenfile und das Logfile über *int getfilename* (siehe 3.4.3) festgestellt
In Folge wird die Kommunikationsschnittstelle geöffnet und die benötigten Parameter für diese eingestellt.
Erst jetzt beginnt das eigendliche Programm.
Bei Programmstart wird der erste Zeitabgleich mit dem Mikrokontroller vorgenommen, später erst im Zyklus der Abfragezeit. Dieser Zeitabgleich erfolgt durch *int hs1s* (siehe 3.4.1) und *int getticks*(siehe 3.4.2).
Danach erwartet das Programm auf die Daten vom Mikrokontroller.
Diese Daten werden werden erfasst und gesammelt bis ein Timeout eintritt, oder aber der Mikrokontroller bekanntgibt das die Messung beendet ist (Taktwert = 2^15).
Danach findet die Auswertung über *int Filter* (siehe 3.4.4) und *int Auswerte* (siehe 3.4.5) statt.
Dieser Zyklus wiederholt sich bis das System abgeschaltet wird.

3.5 Serielle Schnittstelle

3.5.1 Initialisierung

Die Initialisierung der Seriellen Schnittstelle erfolgt mit mehreren Schritten. Zuerst wird mit
```
int serialpointer = open("/dev/ttyS1", O_RDWR );
```
die 2. serielle Schnittstelle „/dev/ttyS1“ für schreiben und lesen „0_RDWE“ geöffnet und der Pointer in die Variable „serialpointer“ geschrieben.
Dann wird mit
```
struct termios tio;
```
eine Struktur für die Einstellungen der Schnittstelle erstellt und mit
```
bzero(&tio, sizeof(tio));
```
gelöscht. Dann erst werden die Einstellungen vorgenommen welche vorerst in „tio“ stehen. Mit
```
tio.c_cflag = BAUDRATE | CLOCAL | CS8 | CREAD;
tio.c_iflag = 0;
tio.c_oflag = 0;
tio.c_lflag = 0;
```
werden die diese festgelegt, wobei „BAUDRATE“ nur eine Variable ist (integer) der ein Baudratenwert zugewiesen werden kann. (z.B. BAUDRATE=B9600 für eine Baudrate von 9600; mögliche Baudraten siehe 3.2). Mit den Befehlen
```
tio.c_cc[VINTR]    = 0;
tio.c_cc[VQUIT]    = 0;
tio.c_cc[VERASE]   = 0;
tio.c_cc[VKILL]    = 0;
tio.c_cc[VEOF]     = 0;
tio.c_cc[VTIME]    = 0;
tio.c_cc[VSWTC]    = 0;
tio.c_cc[VSTART]   = 0;
tio.c_cc[VSTOP]    = 0;
tio.c_cc[VSUSP]    = 0;
tio.c_cc[VEOL]     = 0;
tio.c_cc[VREPRINT] = 0;
tio.c_cc[VDISCARD] = 0;
tio.c_cc[VWERASE]  = 0;
tio.c_cc[VLNEXT]   = 0;
tio.c_cc[VEOL2]    = 0;
```
werden die Funktionen von einzelnen Bytes abgeschalten und mit
```
tio.c_cc[VMIN]     = 2;
```
wird die Mindestanzahl von 2 Bytes für des Lesen von der Schnittstelle eingestellt. Anschließend wird mit
```
tcflush(serialpointer, TCIFLUSH);
```
die Schnittstelle (serialpointer=>/dev/ttyS1) gelöscht, sprich die Einstellungen und alle anstehenden Daten auf dieser Schnittstelle. Dann werden mit
```
tcsetattr(serialpointer,TCSANOW,&tio);
```
die zuvor vorgenommen und in „tio“ gespeicherten Einstellungen auf die Schnittstelle übertragen.

3.5.2 Lesen

Um von der Schnittstelle zu lesen wird der Befehl
```
end=read(serialpointer,buffer,2);
```
ausgeführt. „read“ liest bis zu „2“ Zeichen von der Schnittstelle (serialpointer) und schreibt diese nach „buffer“, sowie die Anzahl der gelesenen Zeichen nach „end“
„buffer“ ist ein Array (int buffer[5];)

3.5.3 RTS-Leitung beeinflussen

Diese Funktion wird benötigt um die RTS-Leitung auf Spannung, bzw. auf Masse zu legen. Dazu wird mit

```
int flag = TIOCM_RTS;
```

ein Code für die RTS-Leitung nach „flag“ kopiert. Mit diesem Code kann man nun über

```
ioctl(serialpointer, TIOCMBIS, &flag);
```

die RTS-Leitung setzen, bzw. mit

```
ioctl(serialpointer, TIOCMBIC, &flag);
```

löschen.

4 Auswertung und Visualisierung

4.1 Aufgabenstellung

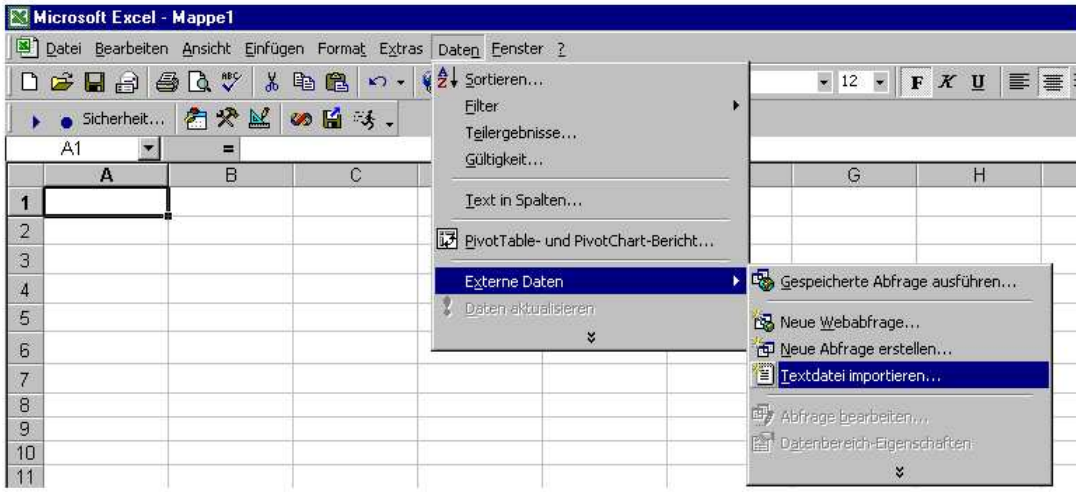
Meine Aufgabe war die Software für die Auswertung und Visualisierung der Messwerte der GME (Geschwindigkeitsmesseinrichtung) im Microsoft Excel zu erstellen. Das Programm sollte leicht zu bedienen sein und die Messwerte automatisch einlesen, auswerten und darstellen. Das Excel hatte alle diese Möglichkeiten und mittels Makros liesen sich die gewünschten Anforderungen auch automatisieren und verknüpfen.

4.2 Realisierung

Die auf einer Flashdisk gespeicherten Messergebnisse der GME werden auf den gewünschten Computer kopiert und mittels meiner Software automatisch eingelesen. Mit dem Befehl Textdatei importieren ist es dem Excel möglich die Daten in die durch Tabstops getrennten Spalten einzufügen. Vorher ist nur der gewünschte Tabellenkopf zu erstellen und der Cursor in die zweite Zeile an den linken Rand zu stellen.

Der Tabellenkopf sieht folgendermaßen aus:

Textdatei importieren:



Bei dem Befehl „Textdatei importieren“ muss nur die einzulesende Datei ausgewählt werden und mit „Fertig stellen“ bestätigt werden. Bei dem darauf erscheinenden Feld „Datei importieren“ muss unter „Eigenschaften“ das „Hakerl“ bei „Spaltenbreite einstellen“ weggenommen werden da sonst die Tabelle sehr zusammengedrückt wird.

Ein Problem das aufgetreten ist waren die unterschiedlichen Namen der Verzeichnisse und Ordner in der die Messwerte gespeichert wurden. Der Befehl Textdatei importieren kopiert nämlich die Messwerte immer von dem Verzeichnis das man einmal genannt hat. Auf meiner Festplatte waren die Messwerte unter R:/sbranjue/GME/Daten gespeichert, wurde daran etwas geändert trat ein Fehler auf und alle Makros wurden unterbrochen! Da aber auf jedem PC die Laufwerke und Ordner anders bezeichnet werden können entstand das oben genannte Problem. Die Lösung war eine kleine Erweiterung im Visual Basic.
Am Beginn des Programms musste die „GetOpenFilename-Methode“ eingefügt werden. Dieser Befehl öffnet einen Datei-Öffnen Dialog. Damit kann man problemlos zu den abgespeicherten gelangen (wo auch immer sie sind) und sie auswählen.
Bei dieser Methode wird das Dialogfeld Öffnen mit einem Dateifilter für Textdateien angezeigt. Wenn der Benutzer einen Dateinamen wählt, wird dieser in einem Meldungsfeld angezeigt.

```
fileToOpen = Application _
        .GetOpenFilename(„Text Files (*.txt), *.txt“)
If fileToOpen <> False Then
        MsgBox “Open “ & fileToOpen
End If
```

Den Rest übernimmt wieder das Makro. Die Messwerte werden in folgende 10 Spalten importiert:

1. Tag
2. Monat
3. Jahr
4. Stunde
5. Minute
6. Sekunde
7. kleinste Geschwindigkeit
8. mittlere Geschwindigkeit
9. größte Geschwindigkeit
10. Richtung

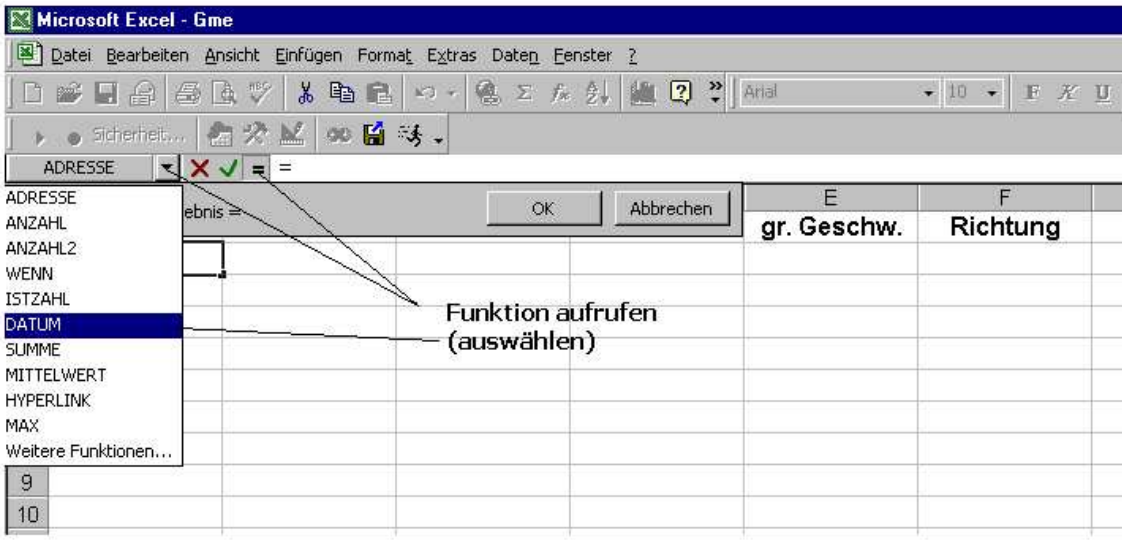
Nach diesen erfolgreich ausgeführten Schritten ist die Tabelle 1 fertig.
Das Makro das diese Schritte automatisiert hat den Namen „einfügen“

Da die Anzahl der Messwerte variabel ist entstand mein nächstes kleines Problem. Diese Tabelle 1 musste nämlich in der Tabelle 2 zusammengefasst werden. Teils wegen der Übersichtlichkeit, teils mussten die Messwerte weiterbearbeitet werden. In der ersten Tabelle wurde die Geschwindigkeit mit z.B.: 1234 angegeben. Dieser Wert entsprach in Wirklichkeit 1,234 ms.
Das Datum und die Uhrzeit mussten ebenfalls von drei Spalten auf eine zusammengefasst werden (sonst können keine Diagramme erstellt werden).
Die Tabelle 1 wurde mit den Funktionen „Datum“ und „Zeit“ zusammengefasst.

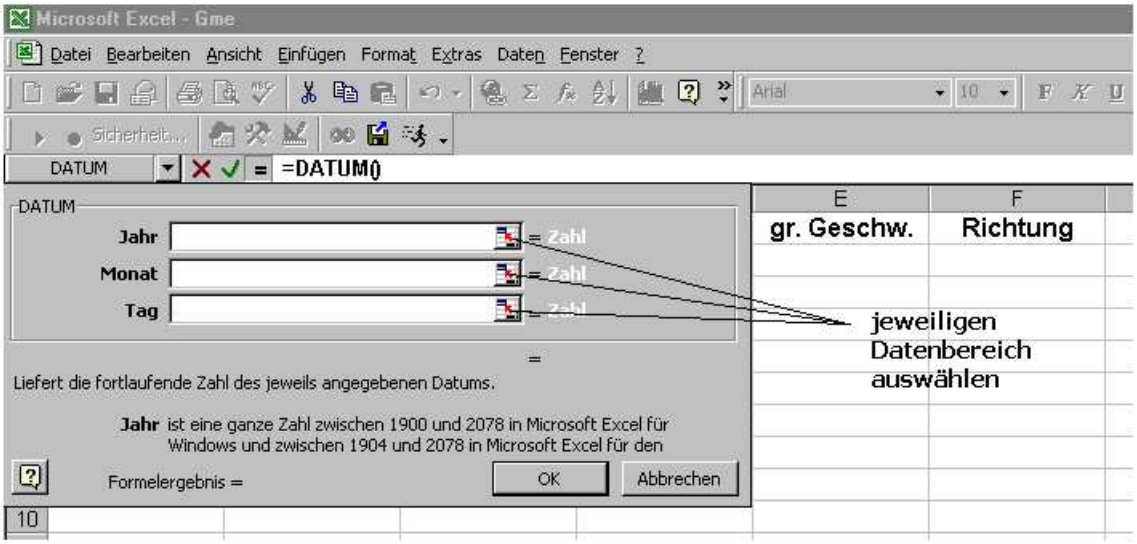
In der Tabelle 2 habe ich folgenden Schriftkopf erstellt:

Die jeweiligen Funktionen „Datum“ und „Zeit“ fassen die Werte von alleine zusammen, man muss nur angeben in welcher Spalte der Tag, das Monat und das Jahr steht (bzw. Stunde, Minute, Sekunde).

Funktion aufrufen:

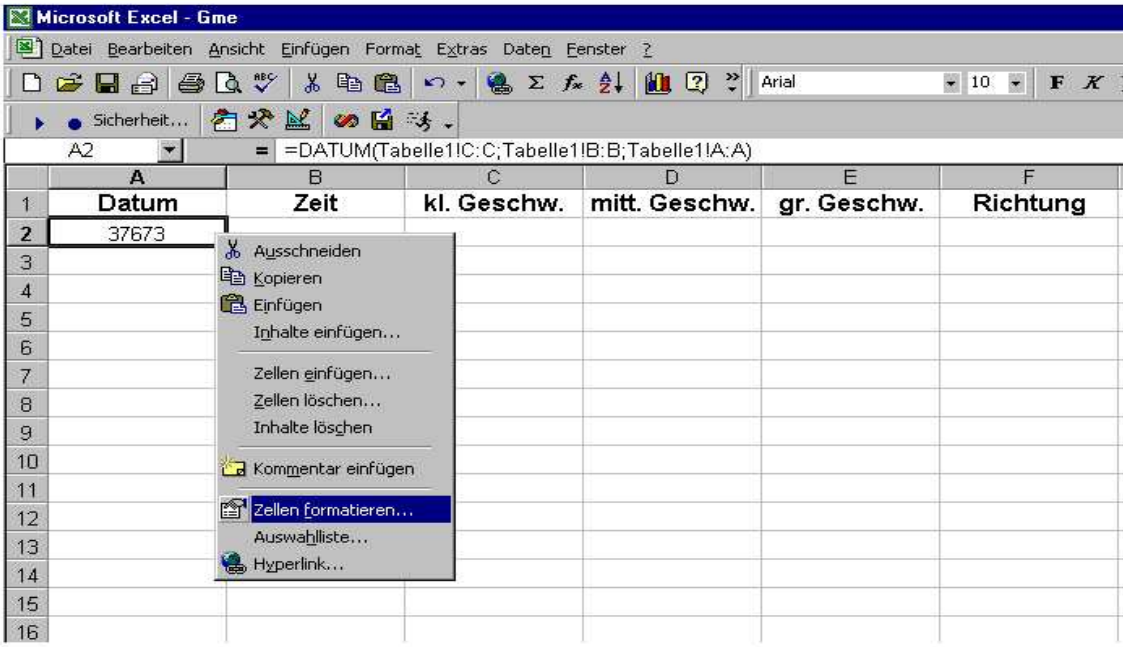


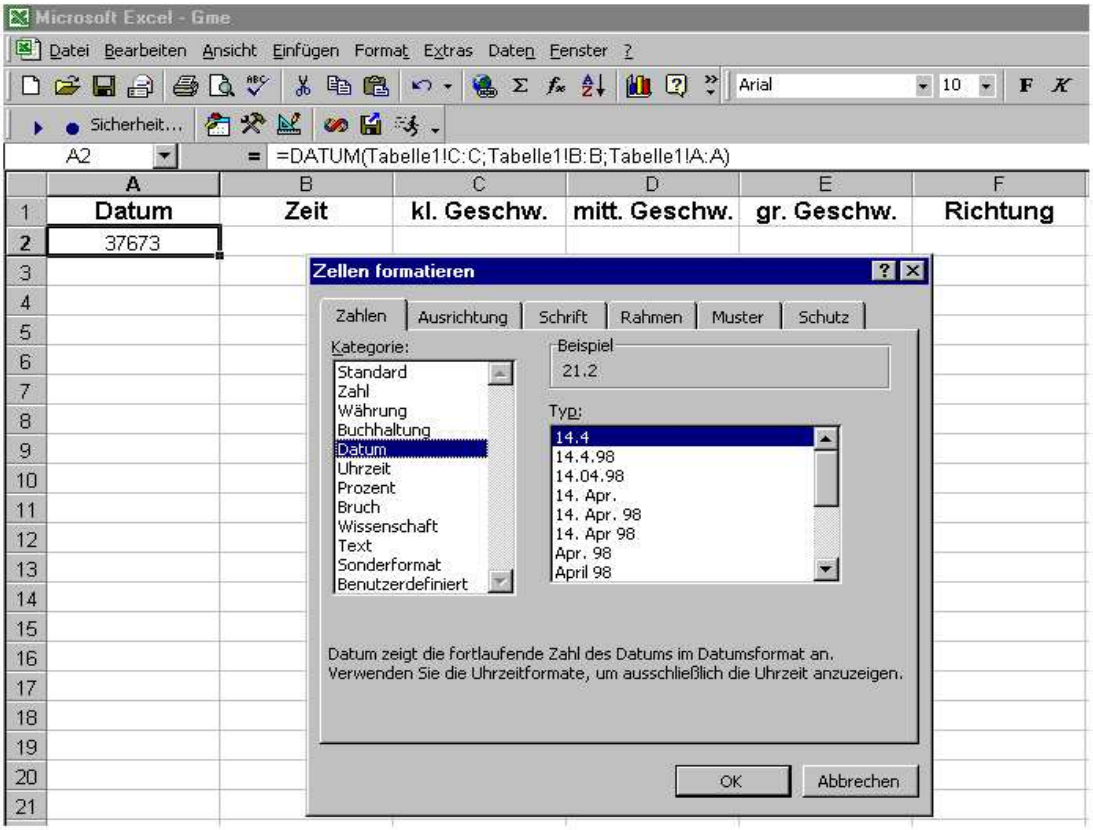
Datum – Funktion:



Nachdem die richtigen Datenbereiche (Tag, Monat, Jahr...) in Tabelle 1 ausgewählt wurden, mit „OK“ bestätigen und das zusammengefasste Datum (oder die Zeit) erscheint in Tabelle 2.

Falls in der Zelle eine Zahl angegeben wird muss die Zelle formatiert werden:





Unter der Kategorie „Datum“ und „Zeit“ kann das gewünschte Datumsformat oder Zeitformat gewählt werden.
Die Geschwindigkeiten kopierte ich einfach nur von der ersten Tabelle in die zweite und dividierte sie durch 1000 um auf m/s zu kommen. Die Spalte „Richtung“ wurde eins zu eins in die Tabelle 2 kopiert.
Das Makro das die Tabelle 1 in der Tabelle 2 zusammenfasst hat den Namen „zusammenfassen2“.

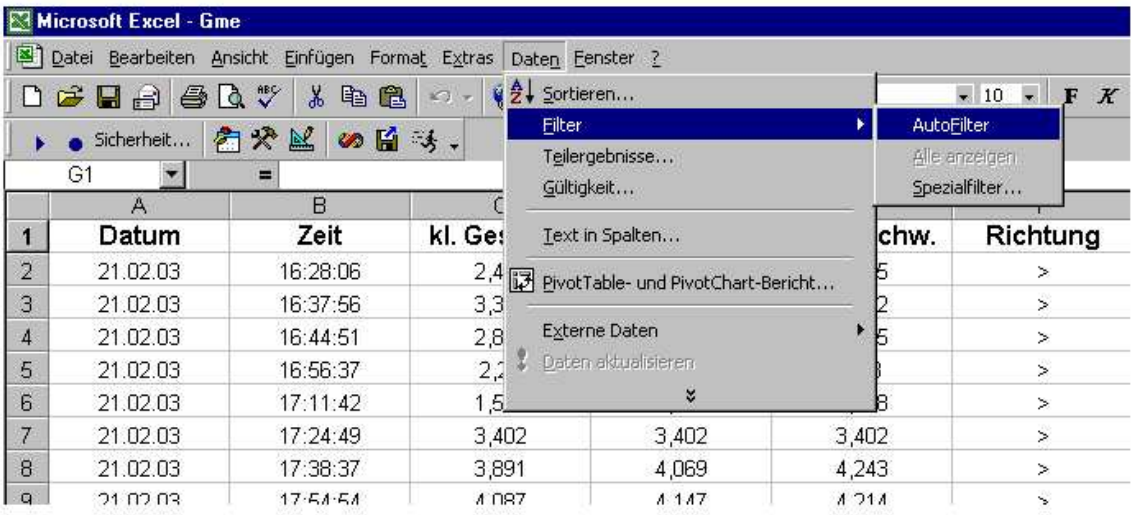
Das zusammenfassen und bearbeiten war aber nicht das Problem, sondern wie die zusammengefasste und bearbeitet Zeile der zweiten Tabelle so weit hinunterkopiert wird, so viele Messwerte es gibt. Bei Makros ist es ja nicht möglich das man den Datenbereich soweit mit der Maus hinunterzieht wie man ihn benötigt, da das Makro jedes Mal neue und auch eine andere Anzahl von Messwerten einlesen muss. Wird der ausgewählte Datenbereich entweder von den Messwerten unter- oder überschritten dann tritt wieder ein Fehler auf und das Makro kann nicht ausgeführt werden. Leider habe ich dieses Problem nur halb gelöst, denn ich habe nur eine Lösung gefunden wie man das Problem löst wenn weniger Messwerte vorkommen als der Datenbereich groß oder lang ist. Die eingegebene Formel in einer Zelle verursacht nämlich einen Fehler sobald ein Wert der Formel nicht gegeben ist. Das Excel schreibt dann sofort in die Zelle #Wert, was soviel bedeutet wie ungültiger oder nichtdefinierter Wert. Sobald auch nur eine Zelle im gesamten Datenbereich dieses #Wert enthält tritt die Fehlermeldung und somit auch die Unterbrechung des Makros ein. Dieses #Wert konnte ich durch eine Wenn-Funktion beseitigen. Diese Funktion wird vor jeder verwendeten Formel eingefügt und bewirkt das wenn ein Wert der Formel nicht vorhanden ist, die Zelle leergelassen und nicht #Wert hineingeschrieben wird.

Wenn – Funktion: =WENN(ISTLEER(Tabelle1!A2);"";

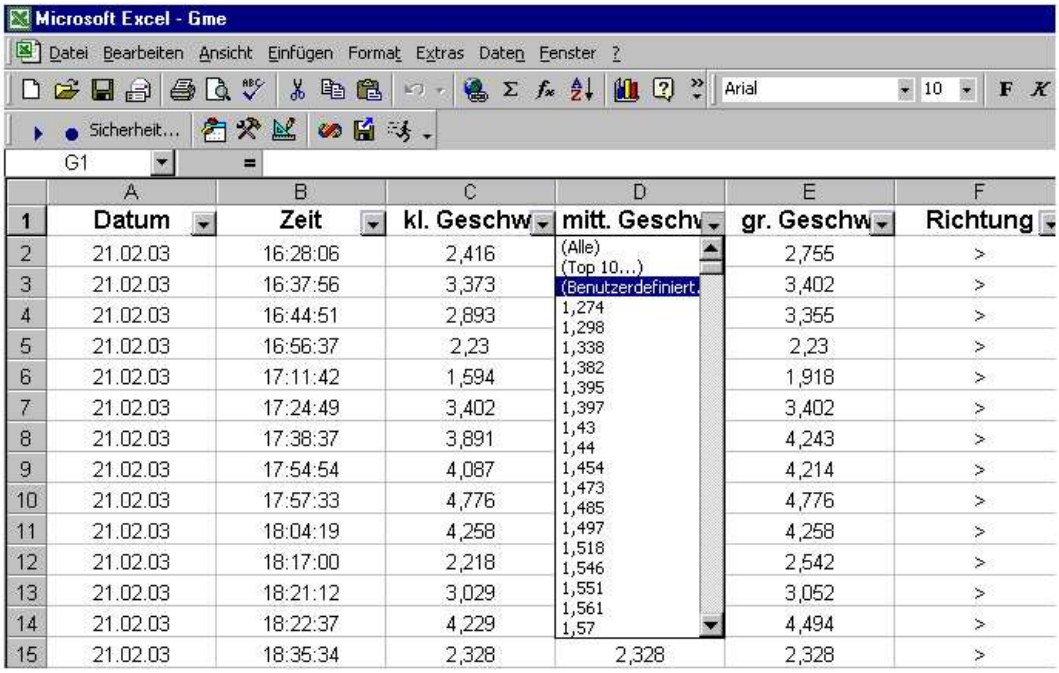
Um zu verhindern das die Anzahl der Messwerte größer ist als der Datenbereich ist, habe ich ihn bis zur Zelle 10000 definiert und somit auch die verwendeten Formeln soweit und in jede Zelle kopiert. Durch die Wenn-Funktion entsteht hier kein Problem mehr und es ist auszuschließen das mehr als 10000 Waggons in einer Woche den Abrollberg hinunterrollen. Damit wäre auch dieses Problem zwar nicht perfekt gelöst, aber immerhin gelöst. Zur Visualisierung benötigte ich außerdem noch das Datum und die Zeit zusammengefasst in einer Zelle, da diese Zelle dann meine Werte für die X-Achse des Diagramms beinhaltet.
Dieses Problem habe ich auf Etappen gelöst. Als erstes habe ich die Tabelle 1 zusammengefasst (Makro: „zusammenfassen2“), anschließend fügte ich mit einem Makro die Wenn – Funktion hinzu (Makro: „mitWenn“), kopierte dies
bis zur Zeile 10000 (Makro: „hinunterziehen“) und fasste als letztes die Zeit und das Datum in eine Zelle zusammen (Makro: „Zeittabelle“).

Als nächste fügte ich einen Filter in die Tabelle ein (Daten, Filter, Autofilter). Damit ist es möglich die Messwerte nach allen möglichen Kriterien zu filtern.

<u>Filter:</u>

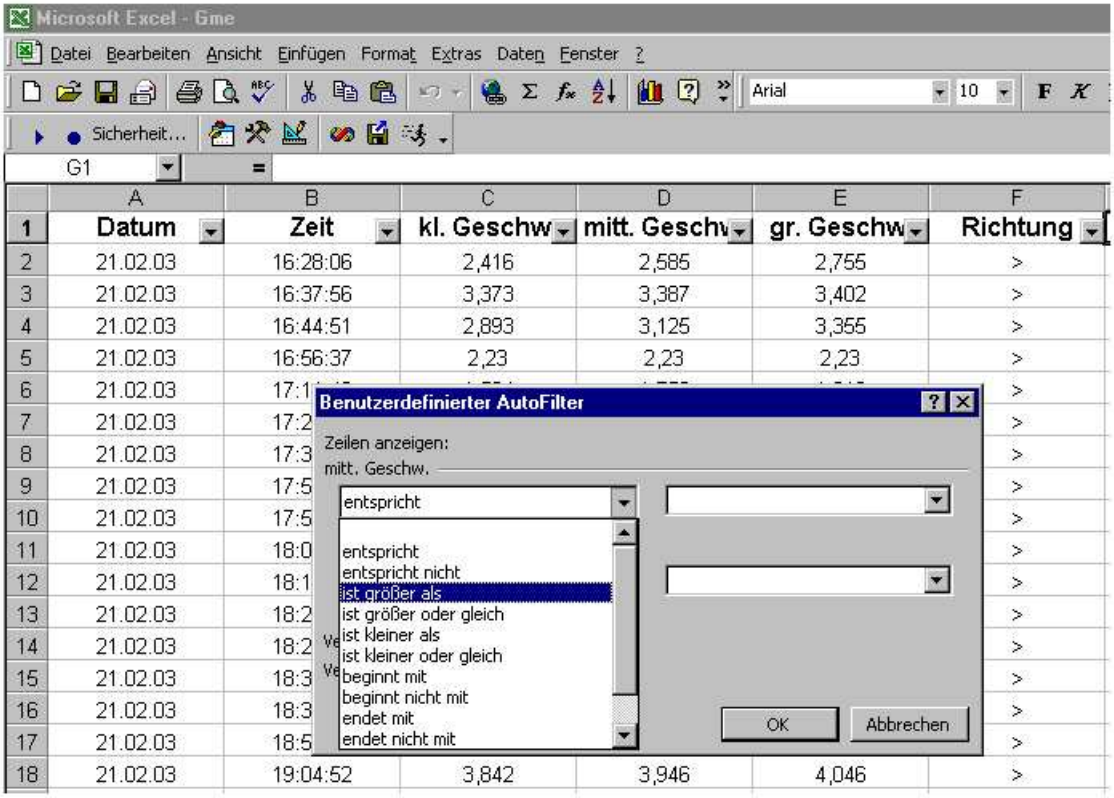


Nachdem der Autofilter in den Tabellenkopf eingefügt wurde konnten die Messwerte nach allen möglichen Kriterien gefiltert werden.



	A	B	C	D	E	F
1	Datum	Zeit	kl. Geschw	mitt. Geschw	gr. Geschw	Richtung
2	21.02.03	16:28:06	2,416		2,755	>
3	21.02.03	16:37:56	3,373		3,402	>
4	21.02.03	16:44:51	2,893		3,355	>
5	21.02.03	16:56:37	2,23		2,23	>
6	21.02.03	17:11:42	1,594		1,918	>
7	21.02.03	17:24:49	3,402		3,402	>
8	21.02.03	17:38:37	3,891		4,243	>
9	21.02.03	17:54:54	4,087		4,214	>
10	21.02.03	17:57:33	4,776		4,776	>
11	21.02.03	18:04:19	4,258		4,258	>
12	21.02.03	18:17:00	2,218		2,542	>
13	21.02.03	18:21:12	3,029		3,052	>
14	21.02.03	18:22:37	4,229		4,494	>
15	21.02.03	18:35:34	2,328	2,328	2,328	>

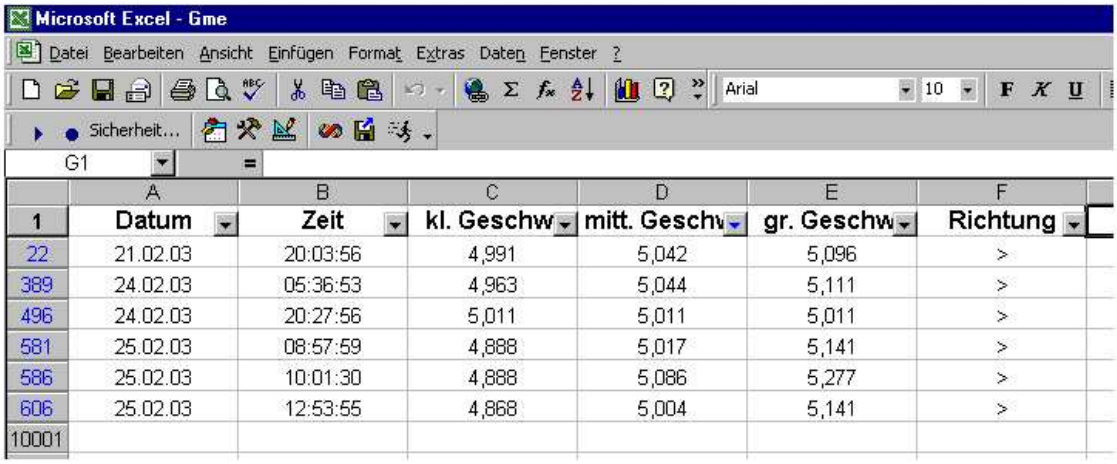
Mit dem „Benutzerdefinierten Filter“ ist es möglich die Messwerte nach bestimmten, frei wählbaren Punkten einzugrenzen.



	A	B	C	D	E	F
1	Datum	Zeit	kl. Geschw	mitt. Geschw	gr. Geschw	Richtung
2	21.02.03	16:28:06	2,416	2,585	2,755	>
3	21.02.03	16:37:56	3,373	3,387	3,402	>
4	21.02.03	16:44:51	2,893	3,125	3,355	>
5	21.02.03	16:56:37	2,23	2,23	2,23	>
6	21.02.03	17:1				>
7	21.02.03	17:2				>
8	21.02.03	17:3				>
9	21.02.03	17:5				>
10	21.02.03	17:5				>
11	21.02.03	18:0				>
12	21.02.03	18:1				>
13	21.02.03	18:2				>
14	21.02.03	18:2				>
15	21.02.03	18:3				>
16	21.02.03	18:3				>
17	21.02.03	18:5				>
18	21.02.03	19:04:52	3,842	3,946	4,046	>

Das Makro das den Filter in die Tabelle 2 einfügt hat den Namen „filtern2“.

Gefilterte Tabelle:



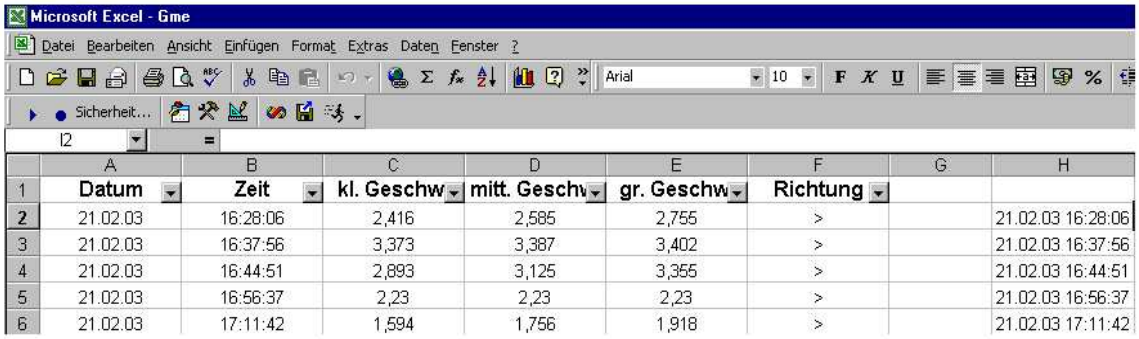
	A	B	C	D	E	F
1	Datum	Zeit	kl. Geschw	mitt. Geschw	gr. Geschw	Richtung
22	21.02.03	20:03:56	4,991	5,042	5,096	>
389	24.02.03	05:36:53	4,963	5,044	5,111	>
496	24.02.03	20:27:56	5,011	5,011	5,011	>
581	25.02.03	08:57:59	4,888	5,017	5,141	>
586	25.02.03	10:01:30	4,888	5,086	5,277	>
606	25.02.03	12:53:55	4,868	5,004	5,141	>
10001						

Bei der obigen Tabelle ist mit dem „Benutzerdefinierten Filter“ die mittlere Geschwindigkeit mit „größer als“ 5m/s gefiltert.

Aus allen oben genannten Makros (einfügen, zusammenfassen2, mitWenn, hinunterziehen, Zeittabelle, filtern2) machte ich ein Makro mit dem Titel **„GMEAuswertung1“**.

GME Auswertung:

Das fertige Makro GME Auswertung sieht folgendermaßen aus:

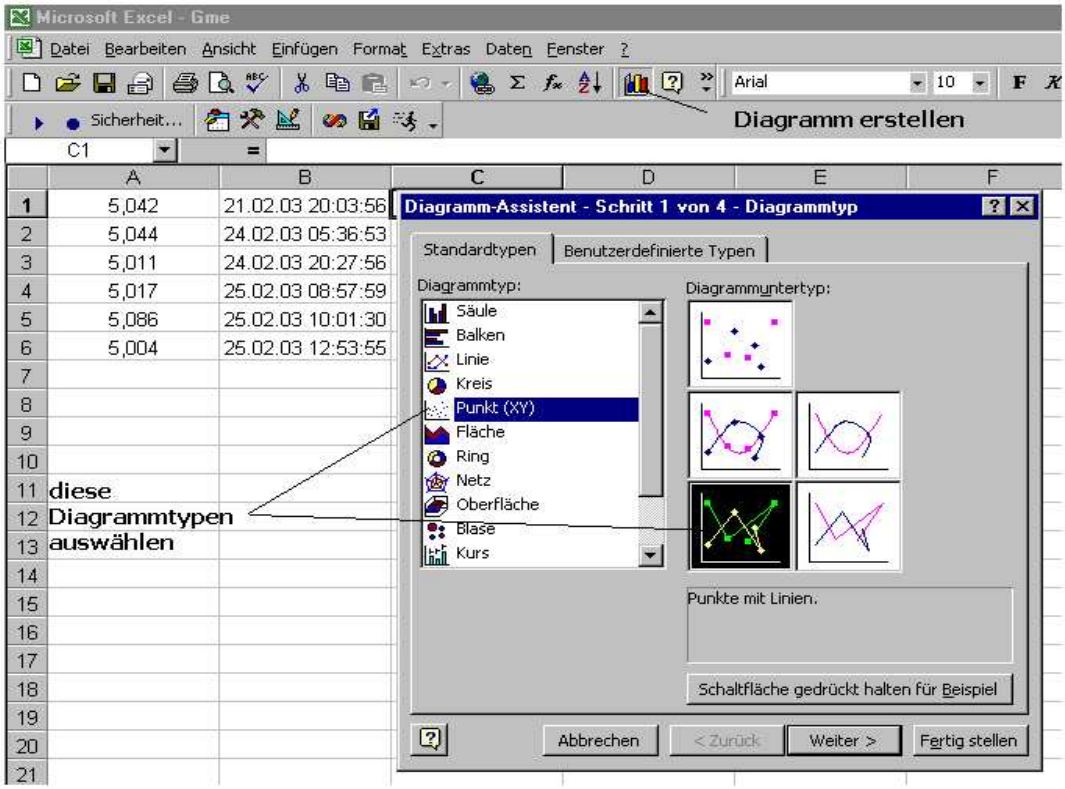


	A	B	C	D	E	F	G	H
1	Datum	Zeit	kl. Geschw	mitt. Geschw	gr. Geschw	Richtung		
2	21.02.03	16:28:06	2,416	2,585	2,755	>		21.02.03 16:28:06
3	21.02.03	16:37:56	3,373	3,387	3,402	>		21.02.03 16:37:56
4	21.02.03	16:44:51	2,893	3,125	3,355	>		21.02.03 16:44:51
5	21.02.03	16:56:37	2,23	2,23	2,23	>		21.02.03 16:56:37
6	21.02.03	17:11:42	1,594	1,756	1,918	>		21.02.03 17:11:42

GME Visualisierung:

Die Visualisierung realisierte ich so, dass ich die mittlere Geschwindigkeit nach meinen Vorstellungen mit dem Benutzerdefinierten Filter filterte (z.B: schneller als 5m/s). Da der Filter die Daten die nicht den Filterkriterien entsprechen in der Hintergrund verschiebt entstand mein nächstes Problem. Das Diagramm das ich versuchte mit den gefilterten Werten zu zeichnen verwendete auch die in den Hintergrund verschobenen Werte. Die Lösung war recht einfach, man musste nur die gefilterten Werte in eine neue Tabelle (Tabelle 3) kopieren, die nicht erforderlichen Spalten und Zeilen löschen (Datum, Zeit, kl. Geschw., gr. Geschw., Richtung) und das Diagramm mittels Diagrammassistenten zeichnen.

Diagramm erstellen:



Das fertige Diagramm könnte (ganz nach ihren Geschmack) in etwa so ausschauen: