University of Ulster

School of Electrical & Mechanical Engineering

MEng Electronic Systems

Final Project Dissertation

Implementation of Comprehensive Data Compression in an embedded GPS Black Box System

Author:	Michael Bachner
1st Supervisor:	Dr Ian McCrum
2nd Supervisor:	Dr Hubert Hoeggl
Year:	2005/2006

Affirmation

I affirm, that the Master of Engineering Final Year Project is my own work, and that it has never been submitted for examination purposes before. All sources and citations used have been quoted as such, and all utilized tools have been mentioned.

.....

Michael Bachner

Acknowledgement

My sincere gratitude goes to:

Professor Dr Ian McCrum (University of Ulster, Newtownabbey), for proposing the topic of this thesis, for his encouragement and his support.

Dr. Hubert Hoeggl (University of Applied Sciences, Augsburg), for his support

Dr. Frank Owens (University of Ulster, Newtownabbey), for his support and endeavour during the exchange program

Abstract

The *GPS Black Box* is designed as a ground based tracking application that integrates GPS, portable storage and micro-controller technology in an embedded system. The unit will record position, speed and direction on a regular basis together with other logged data. Features of the device are, low cost, reliability and small size.

Comprehensive Data Compression is introduced as a method using absolute and relative positions to increase the storage efficiency of the system. The accuracy of recorded positions used to trace the Black Box is verified by several measurements in distinct environments.

Executive Summary

The aim of this master thesis is to design and construct a *GPS Black Box*. It is considered an example application for the combined use of microcontroller and GPS technology in a low cost and mobile device. To improve the performance of such a system, it has to be optimised in the areas of power consumption and storage efficiency.

A literature research that addresses several state of the art methods to reduce power consumption of digital systems has been carried out. Techniques that are applicable to the *Black Box* system are identified, namely operating on low supply voltage, sleep modes and clock frequency adjustment. These results were used to chose an appropriate microcontroller.

An interrupt driven software that extracts RMC messages, that contains GPS position data, is implemented in an ARM7 microcontroller on an Olimex LCD-E2294 development board.

The issue of accuracy of GPS positions is discussed in detail. Several measurements on different routes were evaluated to identify parameters that affect GPS accuracy. It is shown, that the main influence comes from the speed of the vehicle, the *Black Box* is mounted on. The velocity is also used, to characterise a number of distinct environments, namely highway, country and city traffic.

To lower memory requirements, a compression routine using relative positions is introduced. This method is termed *Comprehensive Data Compression*. The compression algorithm is implemented in C-code running on a laptop PC.

Automated offline analysis of different compression settings is carried out using Matlab scripts. It is shown, that the position error resulting from data compression is directly linked to velocity and therefore to a distinct environment.

Evidence is given, that memory requirements could be reduced to 40 % by applying *Comprehensive Data Compression*. The mean position error introduced by the copmpression algorithm is below 5m.

The realization of these concepts in a practical device is demonstrated on the *GPS Black Box*, which is considered an example for further applications.

Contents

1. Introduction

1.1. Background and Motivation

Since its inception in 1978, GPS has fast become the most popular satellite aided navigation system used worldwide. Many industries, such as civil aviation, shipping and agriculture, have become quite dependent on this service.

The combination of GPS technology and the computational power of a modern micro-controller, to build up a mobile device has great potential. The development of such small, low cost devices, that are able to record position, speed, direction as well as other data has many applications in industry and research, e.g. navigation and tracking of all kinds of traffic on ground, sea and air, measuring topological distribution of environmental data etc.

In order, to give such a device the ability to work autonomous for a sufficient long time, it is necessary to make use of low power technology and efficient data storage methods.

Although a number of research projects on this field has been carried out, a complete summary on the various fields of interest, which are related to such a project, is hard to find.

Therefore the motivation for the project arises, to conduct research on data compression and optimisation of power consumption with microprocessors.

1.2. Aims and Objectives

The aim of this master thesis is to design the *GPS Black Box* as an example application that combines microcontroller, GPS- and portable storage technology. As the device is a mobile application technologies and methods such as data compression and optimisation of power consumption are of interest and used when applicable.

To stay within time and budget restrictions, the device is composed of currently available standard components. The GPS module is an already existing closed system connected to and interacting with the microcontroller. The microcontroller is the core element which has significant influence on the overall system performance. An appropriate one has to be chosen to meet the system requirements. Main criteria are low power, internal storage, number and type of interfaces and programming support.

Today, there are a number of useful methods to reduce power consumption in digital designs. Appropriate techniques that have the potential to effectively reduce overall power consumption have to be identified.

The main task, the *GPS Black Box* has to manage, is the storage of the acquired data. Since the storage on a local microprocessor is limited, it is necessary to investigate into methods of efficient data storage. This is accomplished, by implementing a compression routine, termed *Comprehensive Data Compression*, that uses absolute and relative positions to trace the Black Box. The effectiveness of this compression method has to be evaluated.

GPS measurements have to be conducted on different routes, namely highway, country and city traffic. The data samples recorded are analysed offline to determine accuracy and the error resulting from data compression.

1.3. Potential Applications

The GPS Black Box is designed for ground based tracking of an automotive. In a preliminary design, it is powered by the batteries of the car and simply stores geographical positions, time, speed and course along the route to demonstrate the basic functionality in a low cost and low power system. The data can be uploaded to a PC and used to display and recorded data along the route.
The implementation of a rechargeable battery system and the integration of additional sensors (e.g. for the strength of radio signals, meteorological data, surface temperature etc.) makes the Black Box design useful for numerous research projects where surveys based on the geological distribution are of interest.
An even more powerful extension is the integration of a GSM unit to give the Black Box the ability to communicate with a remote control station[1]. This functionality can be used to continuously monitor the system in remote sensing applications,to send periodical SMS containing the system status and data samples or generally to be part of the Amateur Position Reporting System (APRS[2])
Furthermore the GSM unit can be used to set up an alarm message in case of a specified event or an emergency. This makes the design interesting for security and rescue services. [3] describes a tracking system that starts tracking and sends a message containing the actual position to the owner of a vehicle, if an unauthorized movement of the vehicle is detected.
From a commercial point of view, the system can be used to monitor the position of a fleet of delivery, police or rescue vehicles, to guide ships navigating in narrow harbors[3], to introduce an automated toll collection system and many other applications.

1 Section 7.2.3 gives a short note on the integration of a GSM modem

2 APRS is a free web based tracking system for all kind of data. See [1] and [2] for further information on APRS

3 [4] describes such an application in detail

Especially the emerging field of Location Based Services (LBS) makes full use of the combination of GPS and GSM. LBS is a collective term for a great range of mobile services personalised based on location, e.g. finding the closest hotel, Italian restaurant, gas station etc. Further information on LBS can be found in [5] and [6].

1.4. Technology used

The GPS Black Box system utilises a range of different technologies which are briefly described here. To simplify the design process, the physical design is divided into several subsystems that can be dealt with separately. This approach is illustrated in figure 1.1. Chapter *3 Data Logger* gives a detailed description of all components.

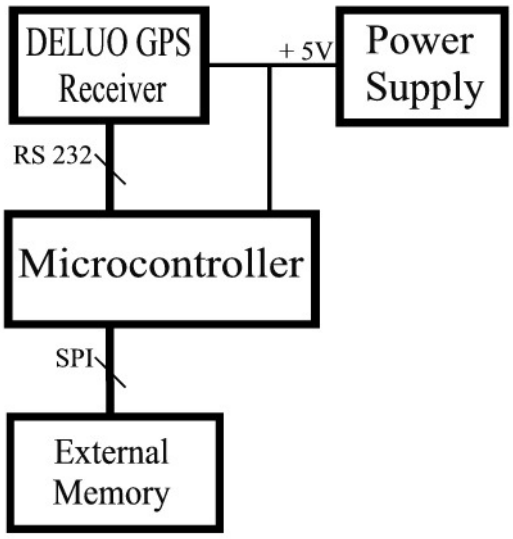


Figure 1.1.: Block diagram of the GPS Black Box system

Power Supply

The whole system is powered by a 12V car plug. The power supply unit (PSU) is used to provide a 5V supply voltage for both, the micro-controller and the GPS receiver.

Microcontroller

The core element in the design is the microcontroller used for data processing. It provides all functionalities to controll and interface the GPS receiver and external memory.

External Memory

The external memory is used to store the recorded data. To enable further computation an interface to a desktop PC or laptop has to be provided. The use of a MultiMedia card (MMC) which is interfaced via a serial peripheral interface (SPI) is a straightforward way to accomplish both tasks.

GPS Receiver

A low cost DELUO GPS receiver that outputs standadisised NMEA data is used to obtain position data. Its characteristics and important functionalities are briefly summarised in section 3.1.2.

The chip on-board the receiver uses the time information sent by satellites to calculate position, velocity, and course. Together with the actual time which is used to uniquely identify each NMEA sentence, these are transmitted through the serial RS232 interface using the standard NMEA-0183 protocol.

In section 2.1 the data that is available from the NMEA protocol is described. A detailed discussion on the accuracy of the obtained data and related issues is found in section 2.2.

1.5. Strategy

At the beginning of the project work an internet based literature research concerning similar projects and low power design is conducted. These issues have been subject of research for the recent years, therefore literature provides sufficient and reliable information to aid further decisions.
Similar projects, that have been carried out by other universities or commercial organisations, such as [7], [8] and [4], are identified to gain a broad overview about the different strategies used to solve the various problems concerned.
To achieve an understanding of the different ways to reduce power dissipation in digital systems and to make statements about the utility of these in the *GPS Black Box* design, a quantitative research about the methods and techniques that are currently used is carried out. The results obtained from this inquiry assist the selection of a suitable microcontroller and feasible methods to reduce power consumption.
A number of measurements are taken on different routes to analyse the accuracy of the used GPS receiver. The data samples recorded are analysed offline using C-code and automated Matlab scripts. Different settings of compression parameters are used to estimate the position error introduced by *Comprehensive Data Compression*.
The Black Box prototype is constructed and the approved compression algorithm is implemented. Final tests verify correct functionality.

1.5.1. Project Plan

The project starts at 15.03.05 in Augsburg, continues on 05.09.05 in Belfast and will be finished on 19.05.06. The period before February 2006 is intended for introduction and orientation. Major project work is done only during the last

semester.

To support the development of the project, the work load is divided in several consecutive parts (See table 1.1). A milestone, which is bound to special requirements and a fixed date, stands at the end of each part. The missing of a fixed date is an early warning, that the progress of the project might be in danger. In that way actions to avoid further delay can be attempted. The requirements of each milestone are explained below.

Milestone	Date	Description
1	August 2005	Background-Research
2	December 2005	Hardware-Design
3	March 2006	Construction of Black Box
4	April 2006	Software-Implementation
5	May 2006	Testing and Documentation

Table 1.1.: Milestones

The first Milestone *Research of Background* is the end of the investigation period. Insights in the above described fields of interest should be available.

At the second milestone *Hardware-Design*, all hardware components should be chosen according to the previous research and already available modules. The components can be ordered and should be available before the beginning of the spring semester at University of Ulster.

At the milestone *Construction of Black Box* all components should be linked to each other. The complete hardware should be working, so that the software design is about to start.

At the end of *Implementation* the software should be complete and running correctly. The system will be able to receive, compute and store data. It should be possible to upload the gathered information to a PC for further processing.

Until the end of May the phase of *Testing and Documentation* should be concluded. Major bugs should be fixed and the dissertation should be completed.

1.6. Document Structure

For clarity, this thesis is structured in several chapters that are briefly described here.

Chapter *2 Position Fixing* briefly describes the basic principles behind the GPS system. GPS accuracy in general, different sources for position deviation and means to increase accuracy are discussed. The chapter concludes with the analysis regarding variations of data recorded on different routes.

The overall architecture of the *GPS Black Box system* is described throughout chapter *3 Data Logger*. All hardware components used in the design is referred to. As an implementation guide, the results of the conducted literature research on low power design techniques are stated here.

In *Chapter 4 Data Processing* the software design and its implementation is discussed in detail. In section *4.2.3 Navigation* important navigation formulas and unit conversions are given.

The compression algorithm implemented here is discussed in chapter **??** *Data Compression*. To increase storage efficiency, *Comprehensive Data Compression* and the optimisation of internal data representation are introduced. The nature of position error resulting from data compression is described.

The total amount of position error due to the implemented data compression routine is described and analysed throughout chapter *6 Real World Testing*. Different settings of compression parameters are applied on GPS data taken beforehand.

The resulting position deviation is analysed to show their relation to the position error.

Chapter *7 Conclusion and Further Work* summarises the results obtained throughout this project. In section *7.2 Further Work* the remaining steps to finish the *Black Box* design are described. The integration of a GSM modem is described in short as a useful extension.

The Appendix contains all documents related to this project. The complete source code that was created is included.

2. Position Fixing

Throughout this chapter the basic principles of GPS and the process of obtaining a fixed position will be discussed. Section 2.1 gives a short introduction in GPS and positioning on a spherical surface like the earth. The kind of data that is available and how these data is processed is described.
Issues of data accuracy and means to increase it are addressed in section 2.2. Data obtained by this receiver on different routes in Germany, Ireland and Northern Ireland are analysed to make statements about the accuracy of the DELUO receiver[1]. The results and its consequences are discussed in section 2.3.

2.1. The Global Positioning System

The global positioning system (GPS) is a satellite navigation system with 24 satellites orbiting the earth in 12-h orbits. An onboard atomic clock is used to synchronise the satellites. This synchronised time, the position of every satellite and other information are continuously transmitted using two spread-spectrum carriers(L1 and L2).
For civilian purposes the free and unencrypted carrier L1 is provided. L2 is encrypted and only the US military is able to use it.

1 The specification and a detailed description of the DELUO receiver is found in section 3.1.2.

A GPS receiver listens for the satellite signals and measures the time of arrival which is compared to the GPS time that is sent in the data. This time difference is used to calculate a pseudo-range to each satellite that is received, and that range is used to compute the position. Three satellite signals are necessary to be able to obtain a two-dimensional fix (latitude and longitude) and four signals to get a three-dimensional fix (latitude, longitude + altitude).
The satellites are evenly distributed such that, on average, there are 12 satellites visible in each hemisphere at any time. However, not all 12 satellites can be used to receive position data, since some of them are located close to the horizon, appear on an acute angle or are otherwise obstructed. This results in a weak or inaccurate signal.
Typically GPS receivers are scanning for all 12 satellites that should be in the hemisphere where the receiver is located, but only the strongest ones are used to obtain a fix. Accuracy increases with the number of satellite signals available.
[9], [10], [11] and [12] provide further general information about the GPS system.

2.1.1. Available Data

The NMEA-0183 standard[2] is the most common output format for GPS data. Nearly all GPS receivers provide a serial RS-232 port that outputs NMEA messages at 4800 bps in intervals of one second. Because NMEA messages are one-way, an application program's control of the receiver's output is limited. The data rate and burst time can not be changed.
Receivers, that support advanced functionalities like faster position update, data logging or the use of an internal buffer, provide a binary protocol (e.g. SiRF) that allows to send commands to the receiver and offers more sophisticated control over output rate and output format. For most standard applications this is

2 NMEA stands for the National Marine Electronics Association.

not necessary and the use of NMEA is recommended to give maximum compatibility with all GPS receivers.
Since the NMEA is a consistent and well defined standard, the messages send by the receiver are easy to parse. The general NMEA message format is:

$<Address>,<Data>*<Checksum><CR><LF>

The data field of each sentence starts with a "$" and ends with a "*" followed by a 8bit checksum and a terminating "CR/LF". The checksum is used to verify that the received sentence is valid and complete. In table 2.1 NMEA sentences that

RMC	Contains recommended minimum specific GPS/transit data
GLL	Provides latitude, longitude, and UTC (Universal Time Coordinated) data
GGA	Contains UTC, fix, and position data
GSA	Provides GPS DOP and active satellite information

Table 2.1.: Useful NMEAsentences

are useful for GPS applications are listed[3]. Out of the various GPS sentences only the RMC (Recommended Minimum Specific GPS Data) sentence is used, since it contains all the basic information necessary for navigation. Table 2.2 displays the comma delimited fields of a RMC message as well as their format.

Latitude and longitude are the two fundamental angles, that describe a position on the spherical surface of the earth[4]. Latitude describes the height of the position compared to the equator. It goes from 0° (the equator) to 90°S (south pole) and 90°N(north pole). The longitude is the angle between the actual position and the longitude that goes through Greenwich, England (0°). It ranges up to 180° in both directions (East and West).

3 See [13] for further information about NMEA messages and related topics.

4 In reality, the ellipsoidal nature of the earth has to be taken into account. This is done by the quasi standard WGS-84 coordinate system[14]. The corresponding British standard is the British Ordinance Survey Grid OSGB36. For conversions see [15].

Field	Data format
Sentence	$GPRMC
UTC Time	hhmmss.sss
Status	A = Valid, V = Invalid
Latitude	ddmm.mmmm
N/S Indicator	N = North, S = South
Longitude	dddmm.mmmm
E/W Indicator	E = East, W = West
Speed over ground	Knots
Course over ground	Degrees
UTC Date	DDMMYY
Magnetic variation	Degrees
Magnetic variation	E = East, W = West

Table 2.2.: Data available from a RMC message

The <magnetic variation> field indicates the actual divergence between the geographical and the magnetic north pole and is required only, when using a GPS receiver in combination with a magnetic compass. The course is given as true heading based on the geographic north. Since this is the standard setting for mapping software applications, the <magnetic variation> field is skipped.

A typical example of RMC data taken on university campus is shown below:

```
$GPRMC,101137.878,A,5441.1338,N,00553.0118,W,000.0,225.1,310306,005.7,W*6D
```

The location is 54°41.1338' north and 5°53.0118' west. The message was received on March 31st, 2006 at 10:11:37 UTC and although the receiver is not moving, it has a true (geographic) course of 225.1° north. Such illogical effects are due to the GPS inaccuracy and is filtered out by the software.

2.2. Accuracy of NMEA Positions

When dealing with GPS, one of the most important facts to consider is the accuracy of the positional data that is achievable. Accuracy matters and has a direct impact on the usability of GPS for certain applications.
Regarding GPS accuracy, two major effects are important. First are the inaccuracies due to the GPS system itself. The reasons and means to prevent them are discussed in section 2.2.1. Second is the limited resolution due to the number of digits used to express the angles of position. Throughout this paper, this is referred to as numerical accuracy and is described in detail in section 2.2.2.
Measurements on different environments, namely highway, cross country, city traffic and stationary, have been taken to analyse the accuracy of positions obtained by the DELUO GPS receiver. The results of this analysis are stated in section 2.3.

2.2.1. GPS Accuracy

Since turning off the selective availability (SA[5]) the accuracy of GPS is reported to be within 10 to 30 m in the horizontal plane and slightly more in the vertical plane[17]. This is the accuracy over a longer period of time, taking daily changing atmospheric conditions into account. For short periods a repeatability of around 1m can be achieved.
Nowadays, the gross inaccuracies in the system come primarily from the changes of the speed of satellite signals traveling through the Ionosphere, but also environmental effects like surrounding buildings or canyons that reflect or block signals

5 The US government used SA to introduce a pseudo random error to the free and nonmilitary Standard Positioning Service (SPS). The accuracy level was decreased to about 100m horizontal and 160 m vertical. [16] contains a comparison of horizontal errors with and without SA

as well as the kind and position of the receiver have to be considered.
For a detailed discussion on various GPS accuracy related issues, [18] is recommended. [19] contains an in depth discussion of how GPS errors of different GPS receivers can be modeled to predict GPS accuracy on different conditions.
In the following, three means to increase the accuracy of position data are described in more detail. They all require additional hardware and are therefore not utilised in the current design, but can be seen as potential methods for optimisation.

Position of Antenna and GPS Receiver

To get the highest accuracy out of a GPS receiver, the position of the receiver or its antenna is of great importance and has to be considered carefully. In general, it should be placed as high and open as possible.
The GPS receivers available can be divided into two main groups. The first group has an antenna on-board the receiver while the second one uses an external active antenna.
The latter group is usually more expensive, but provide better reception due to the fact that the external antenna is more powerful and can be placed at an exposed position, such as the roof of a vehicle to boost the strength of received satellite signals. For further reading, [20], [21] and [22] provide a detailed summary over the quality of various GPS receivers, different antenna designs and related issues.

The DELUO GPS receiver, which is used in the GPS Black box design, has an on-board antenna. To reduce shielding effects by the car as much as possible and obtain high accuracy data, it is highly recommended to place the GPS Black Box right behind the front or back windscreen.

Differential GPS (DGPS)

The original intention of differential GPS (DGPS) was to cancel out the errors introduced by SA, but even now it provides an significant improvement to the

accuracy of the GPS system[6].

In short, DGPS works as follows. A fixed, accurately positioned, reference station calculates an error value based on long term variations of the measured GPS signals. This error is the same for all local GPS receivers surrounding the reference station. It is broad casted via satellite or FM radio and used to correct the position.

Major applications are found in the guidance of coastal shipping traffic and farming industry for guidance of tractors working very large fields.

Wide Area Augmentation Signal (WAAS), EGNOS and MSAS

This is a signal transmitted from additional geo-stationary satellites that mimics the GPS satellites. These satellites transmit corrections for errors due to variations in the speed of radio signals traveling through the ionosphere. These corrections are measured and updated on a regular basis by correction stations around the world.

WAAS covers North America, but similar systems are provided by Europe and Russia (EGNOS) and by Japan and Australia (MSAS). Together these form a complete global correction system[24].

2.2.2. Numerical Accuracy

The numerical accuracy of a GPS position is determined by the resolution or number of digits of the position data, that comes with an NMEA message. The values for latitude and longitude come as ASCII data in the format DDDMM.mmmm for latitude and DDMM.mmmm for longitude. DD equals degrees(°), MM minutes(') and .mmmm is the decimal fraction of a minute.

For means of accuracy and navigation the angular values need to be expressed

6 [23] contains a number of measurements, that give evidence to this fact.

as distances. A nautical mile (nM) is defined as the great circle distance of one minute. For the conversion to kilometers formula 2.1 is used.

$$1' = 1nM = \frac{40003km}{360^o \cdot 60'} = 1.8520km \tag{2.1}$$

Calculating course and distance is easier when the angles are given as radians. Table 2.3 gives conversions between radians, degrees and distance in meters. Since NMEA data is accurate up to 4 decimal places, the highest numerical

Minutes	Distance	Radians
1'	1852 m	0.0002909 rad
0.0001'	0.1852 m	$2.909 \cdot 10^{-8}$ rad
0.0057296'	6.366 m	$1 \cdot 10^{-6}$ rad

Table 2.3.: Numerical accuracy of NMEA data

accuracy achievable with NMEA data (DDMM.mmmm) is 0.1852 m (see table 2.3). However, it is important to note, that although using less decimal places will reduce accuracy, this may has benefits in the internal representation of the positional data.

E.g. using an integer value (16 bit) instead of an long integer (32 bit) will increase storage efficiency by as much as 50%! This issue is dealt with in more detail in section *5 Data Compression* on page 42.

2.3. Position Accuracy of the DELUO GPS Receiver

To make specific statements about the accuracy of positions obtained with the DELUO GPS receiver, a number of data samples have been taken while driving

on different routes including highway, major and smaller cross country roads and city traffic in Germany, Ireland and Northern Ireland. A sample taken while holding a stationary position on University campus is included as well.
These samples are analysed and discussed regarding variations of positional data. The figures 2.1 - 2.5 show the variation of latitude, longitude, speed and course of the actual RMC sentence compared to the previous one in different environments. This is referred to as differential latitude, differential longitude, differential speed and differential course and is an indicator of the accuracy over short periods.
Obviously, changes are expected to be more pronounced in city traffic or small cross country roads, than on a highway. The results and consequences of the analysis of above described samples are summarised in section 2.3.4.

2.3.1. Stationary Position

To estimate the stationary accuracy of the receiver under open sky conditions it was placed for two hours outside on university campus[7]. The results of the stationary observation are displayed in figure 2.1. As expected, the data showed very little variation in latitude and longitude. In fact it is around $\pm$ 10m.
Although the receiver was not moving, it showed a quite small speed value ($<$ 5 km/h) and a considerable variation in course. This can be explained as effects of small, quasi random changes of latitude and longitude around the actual position. The effect is displayed in figure 2.2, which shows the variation of latitude and longitude over the period of two hours.
Latitude and longitude are given in radians. Thus, the stationary position is varying by $\pm\ 0.08 \cdot 10^{-4}$ radians ($\pm$ 5 m). These variations are caused by distortions of the satellite signal in the ionosphere.

7 It has to be noted, that the receiver often had problems to locate enough satellites for a fixed position when it was located inside a building or the view was otherwise obstructed.

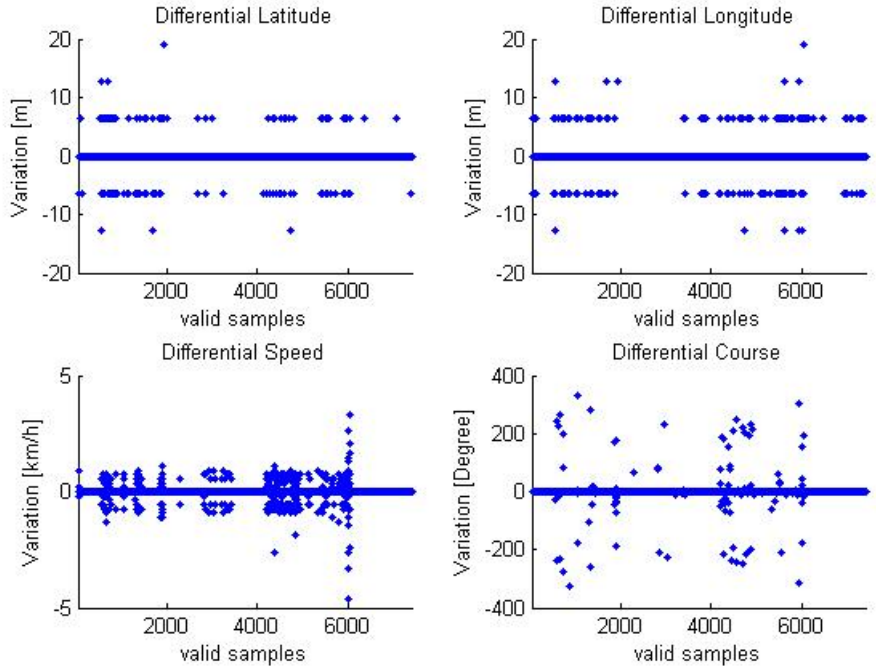


Figure 2.1.: Position accuracy of stationary measurement on University campus

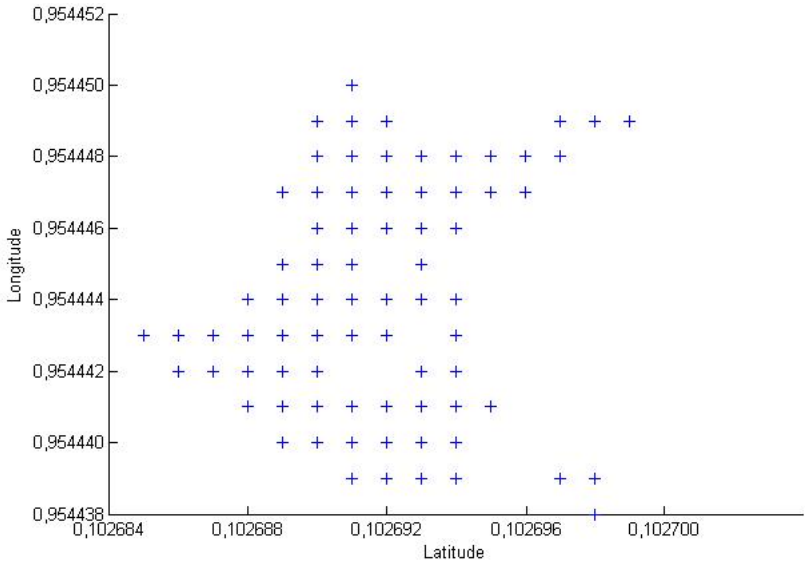


Figure 2.2.: Stationary Dilution of Position

2.3.2. Highway and Major Roads

Due to the higher speed on big straight roads, the samples taken on the highway and major cross country roads are supposed to show more but rather constant changes in latitude and longitude, and less changes in course.
The first sample which is shown in figure 2.3 was taken on a highway in Germany, the second one (2.4) while driving on a highway and major cross country roads in Northern Ireland. As expected, latitude and longitude are well inside boundaries

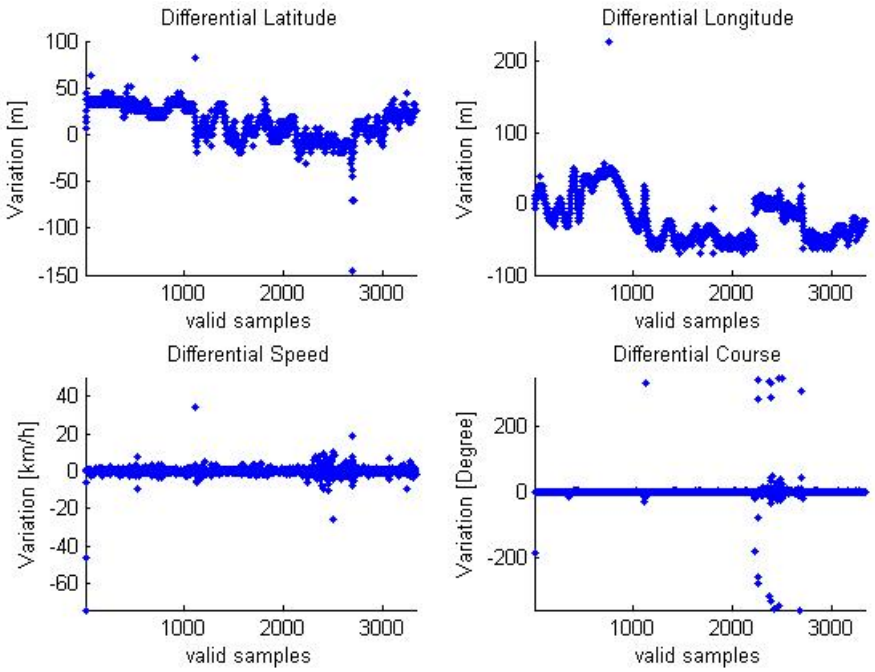


Figure 2.3.: Position accuracy on a highway in Germany

of ± 50 m. The values for speed differ by ± 10 km/h. Since the plots show the difference between one position and the previous one, *dif_lat* and *dif_lon* can be seen as velocities and *dif_speed* represents an acceleration.

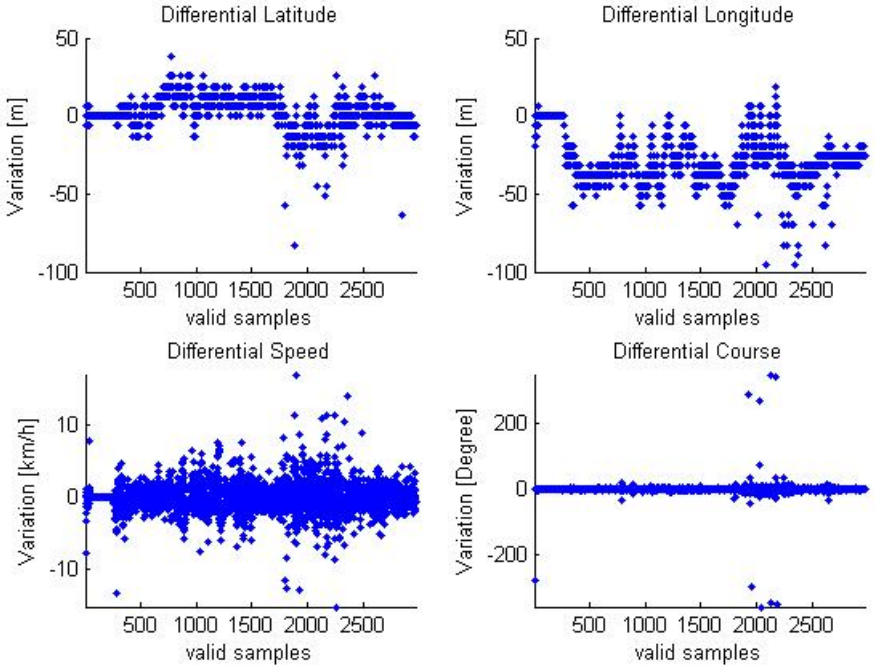


Figure 2.4.: Position accuracy on highway and major cross country roads

2.3.3. City Traffic and Small Cross Country Roads

Because the mean speed is usually lower than on major roads, the values for latitude and longitude are expected to change less. However, variations in speed and course are likely to be more distinct, since steady driving is more often prevented by poor road conditions, traffic lights, turns or other traffic. As illustrated in figure 2.5, latitude and longitude show variations less than ± 20 m. Although the main *dif_speed* values are below 10 km/h/s, there are some peaks up to 40km/h/s. This can be interpreted as a short period of massive acceleration e.g. after traffic lights turning green etc.

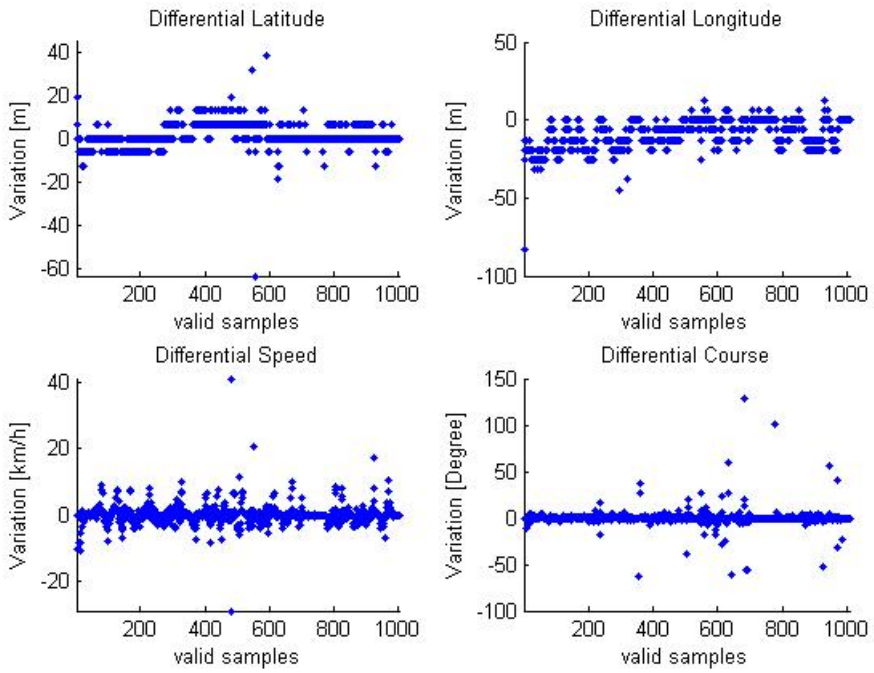


Figure 2.5.: Position accuracy in city traffic

2.3.4. Results and Consequences

The variations in latitude, longitude, speed and course shown in the figures 2.1 - 2.5 can be seen as characteristics of a particular environment. Summarised in table 2.4 are the characteristic values for highway, cross country and city traffic. The main parameter that distincts the different types of environments is the

Environment	max. variation lat. / lon.	max. variation speed
Stationary	± 10 m	< 5 km/h
Highway / Major roads	± 50 m	± 10 km/h
City traffic / Small roads	± 20 m	± 10 km/h

Table 2.4.: Summary of analysed data samples

speed at which vehicles are moving. This fact is important, because it is directly related to the feasibility and effectiveness of data compression as well as to the

resulting position error.

These issues will be considered in more detail throughout the sections *5 Data Compression* and *6 Real World Testing*.

3. Data Logger

This chapter gives a description of the architecture and the basic components in the design of the GPS Black Box. The different technologies introduced briefly in section 1.4 are described in more detail. The system is considered a standalone application used for logging of position and other data inside a car.
However, due to time restrictions and a number of hardware problems it was not possible to build up a Black Box system. Only a demonstration unit showing the basic functionalities could be implemented. Therefore, this chapter is considered as a guide for further work.
An important point to consider, when designing a mobile application is the reduction of power consumption. This is essential to enable the system to work autonomous as long as possible. Section 3.2 contains the summary of a conducted literature research covering various low power issues.

3.1. Architecture

The architecture and physical design of the whole system including all components, their interfaces and interconnections is described in this section. As shown in figure 3.1, the design process is split into several functional subsystems interacting with each other. Each subsystem is discussed separately throughout the following subsections.

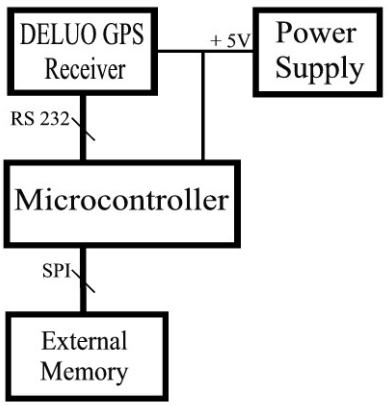


Figure 3.1.: Block diagram of the GPS Black Box system

3.1.1. Microcontroller

The microcontroller running the control software is the core element in the design. It has controll over all system activities and is the central component at which all of the information is processed. Out of the great variety of microcontrollers available, an Atmel ATmega128 was the final choice.

ATmega128

The main reasons for this decision were its two hardware UARTs and the advanced support of a number of low power functionalities[1]. Other advantageous features are free integrated development environments (IDE) for C and Assembler and a good programming support through a large programmers community[26]. The memory available is 128 kB of programmable flash for program space, 4 kB of SRAM and 4 kB of EEPROM which is more than enough to meet the requirements of this project and leaves space for experimentation and future extensions. One of the two UARTs will be used for communication with the GPS receiver while the second is used to set up a debug connection with a PC. In a further stage, this second UART can be used to communicate with the GSM modem.

1 It supports up to six different sleep modes as well as internal control over clock frequency[25]

All communication is realised via the serial RS 232 standard at 4800 baud. Further, the ATmega128 comes with a development board with an integrated USB bootloader for programming[2].
Unfortunately, it was not possible to create a communication with the ATmega on the board and the manufacturer did not respond to several direct inquiries.

PICF877

As a result, it was considered to use a PICF877 with a Millennium Board which was ready at hand. Since the PIC microcontroller is very limited in memory (8kB Flash, 368 Byte RAM and 256 Byte EEPROM) and provides only one UART this proved to be no alternative.

Olimex LPC-E2294 Board

To build up a demonstration unit, that implements basic functionality a ARM7 on a Olimex LPC-E2294 developmentboard was used. Important features useful

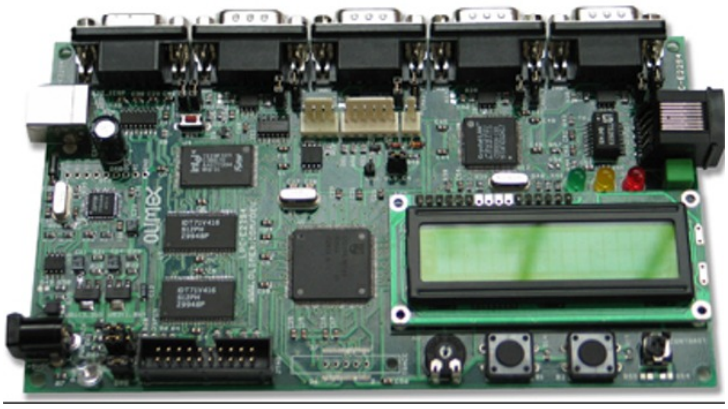


Figure 3.2.: Olimex LPC-E2294 Board

for the set up of the *GPS Black Box* are:

- MCU: LPC2294 16/32 bit ARM7TDMI-S with 256K Bytes Program Flash, 16K Bytes RAM, EXTERNAL MEMORY BUS, 2x UART

2 The USB ATMEGA Controller Board is a new product from Fox4You Ltd. & Co KEG

- 1MB (256Kx32bit) 12 ns 71V416 SRAM
- 4MB (512Kx16bit) 70ns TE28F320C3BD70 C3 INTEL FLASH
- standard JTAG connector with ARM 2x10 pin layout for programming/debugging with ARM-JTAG
- USB to RS232 convertor, board can take power only from USB port
- RESET circuit with exterman control of Philips ISP utility via USB-RS232 virtual port
- Jumpers for ISP/RUN mode
- LCD 16x2 DISPLAY with BACKLIGHT
- SPI connector
- RS232 driver and connector
- two on board voltage regulators 1.8V and 3.3V with up to 800mA current

The Olimex board is used in another masters project by Michael Labus to implement a *eCos* real time system[27]. Due to the kind support from Mr Labus a demonstration program could be implemented. A description of the functionality is given in section 4.1.

3.1.2. The Deluo GPS Receiver

The receiver utilised in this project is a standard integrated receiver manufactured by DELUO. It outputs standard NMEA sentences to acquire the positional data. Furthermore, it is able to track up to twelve satellites simultaneously and provides accurate satellite positioning data with fast time-to-first-fix (TTFF). The receiver with integrated antenna minimises board size and power consumption. Table 3.1

summaries the specification of the DELUO GPS device. The according data sheet is found in appendix A.1.

FEATURES	DESCRIPTION
General	L1 frequency, C/A code, 12-channel
Sensitivity	-165 dBW minimum
Update Rate	1Hz
Accuracy	Position: 25m CEP without S/A
	Velocity : 0.1/sec without S/A
Acquisition	Cold start: < 90sec (typical)
	Warm start: < 40sec (typical)
	Hot start: < 15sec
	Reacquisition <100msec
Primary Power	+3.8V - 8V DC
Current Consumption	105mA @ 3.8V
Serial Interface	RS-232
Protocol	NMEA-0183 v2.20 @ 4800/9600 baud, 8-None-1
Datum	219 standard datum, default WGS-84
Antenna	On-Board Patch Antenna
NMEA Messages	GGA, GLL, GSA, GSV, RMC, and VTG
Dimension	55mm x 40mm x 20mm
Weight	115g

Table 3.1.: DELUO GPS Receiver Specification

The support of standard NMEA messages ensures compatibility with nearly all GPS receivers available today. This is important, because it simplifies further improvements of the design.

For example: The design of the GPS Black Box can be optimised for higher accuracy by simply using a receiver with an active antenna. There is no need to touch another part of the system or to adjust any part of the software, as long as no binary protocol is used to add advanced functionality[3].

3 More sophisticated GPS receivers support the use of a binary protocol to implement advanced functionalities like internal data logging. These protocols are different for each manufacturer and therefore prevent compatibility.

Start-up Behaviour

Scanning for available satellites begins after the initial self-test is completed. On the very first power-up or when the receiver was off over a considerable amount of time, a cold start has to be performed. This results in an increased acquisition time, because the receiver has to cycle through all possible satellite codes until a satellite with sufficient signal-to-noise ratio is found. The satellite is used to download the current satellite constellation, which enables the receiver to tune in the visible satellites. Since the orbital information is stored inside the receiver, further power-ups require less time. This is referred to as warm or hot start.

Internal Data Processing

The chip on-board the receiver uses the time information sent by the satellites to calculate position, velocity, and course. Together with the actual time which is used to uniquely identify each NMEA sentence and some other data, these are transmitted through the serial RS232 interface.

3.1.3. Data Storage

To store the recorded data the GPS Black Box needs to be equipped with some kind of external memory. As the system is a mobile application, an interface to a PC application has to be provided for further computation.
The most common storage type in microcontroller systems is the flash storage. It is low cost and provides large amounts of memory. Another common technology particularly used in mobile applications are portable memory cards with an integrated controller.
Portable memory cards like SD card, MMC card and Compact Flash cards provide an easy and straightforward way to transfer data from the Black Box unit to a desktop PC. Currently available MMCs offer storage capacities of up to 4 GB

and a transfer rate of 2,5 MB/s. MMCs are about the size of a postage stamp (24 mm x 32 mm x 1.5 mm) and used in almost every context in which memory cards are used. The only hardware needed is a card reader, which is low cost and already included in many modern desktop and Laptop PCs.
Communication with a MMC socket can be established via a serial peripheral interface (SPI) using seven pins of one of the I/O-ports of the microcontroller. Since the socket is powered by the microcontroller, no further hardware components are necessary.
The data sheet of a MMC compatible DataFlash card, considered and ordered for the use in the Black Box system, is found in Appendix A.2. Due to the problems with the microcontroller mentioned above the external storage could not be implemented and remains further work.

3.1.4. Power Supply Unit (PSU)

In the first attempt, the GPS Black Box is designed as a vehicle based application powered from the car's 12V battery. The GPS receiver accepts voltages from 3.8V - 8V. Since the Olimex board housing the ARM7 controller is powered with 5V a single voltage level is sufficient and all the PSU needs to do is to convert the 12V down to 5V.

Requirements	Voltage	Current
Olimex Board	5V	800mA
DELUO GPS receiver	3.8V - 8V	200mA (105mA[4])
PSU specification	5V	800mA

Table 3.2.: PSU specification

4 according to data sheet [28]

Although the receiver specification states typical currents of 105mA, measurements on the operating receiver showed currents around 200mA. The PIC will consume another 100mA. The requirements are summarised in table 3.2.
Two switching 7805 regulators with a maximum output current of 500mA are used in parallel to generate the 5V level for both devices and provide sufficient current. Compared to a linear regulator, a switching regulator has an increased efficiency which means that less energy is wasted as heat. This means that there is no need for any heat sinks with their additional space and mounting costs. A possible circuit is shown in figure 3.3. The power supply also has several capaci-

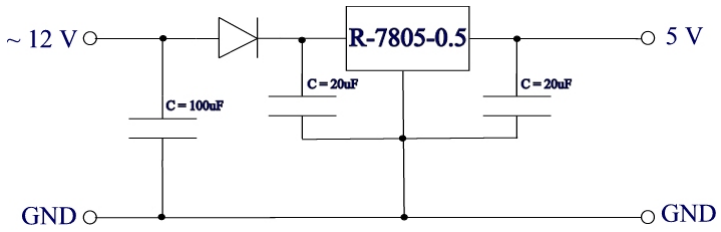


Figure 3.3.: Power Supply Circuit

tors and diodes to protect it against excessive positive spikes and load dumps[5].
As a future extension, a rechargeable battery system would be a useful thing to implement. This would give the Black Box system the ability to operate as a standalone device independent from a vehicle.
However, due to time restrictions the current design is still in the phase of development and no PSU is implemented.

3.2. Power Consumption

Since the *GPS Black Box* is a mobile device, power consumption is a critical issue. The design goal is to minimise power consumption as much as possible,

5 both negative and positive spikes can reach voltages over 80V in a car![29]

but guarantee proper functionality.

To achieve an understanding of the different ways to reduce power dissipation in digital systems and to make statements about the utility of these in the *GPS Black Box* design, a quantitative research about the methods and techniques that are currently available has been carried out. The results obtained from this inquiry can be found in appendix *B Low Power Design*. They are used to assist the selection of a suitable micro-controller and to identify feasible methods that have a high potential to reduce power consumption.

Different types of power consumption have been analysed and several methods to reduce power dissipation have been discussed. The most effective techniques in regard to the *GPS Black Box* system are evaluated here.

In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. An additional effect is

Minimisation Method	**required hard-/ software**	**expected power savings**
Low Voltage Operation	-	significant
Sleep Modes	wakeup logic	depending on idle time
Clock Frequency control	frequency divider low speed clock source	depending on system utilisation
Reduce external accesses	internal memory	depending on access frequency
Reduce logic state transitions	programming style (i.e. Gray Code)	in loops or frequently used sequences only

Table 3.3.: Power Minimising Techniques

provided by lowering the clock frequency to a minimum level, without affecting timing requirements. The alignment of the clock frequency to processor utilisa-

tion provides minimal power consumption for a given task.
Table 3.3 summarises the power minimisation methods that seem to have the greatest potential, the required hard- or software and the expected power savings. The GPS module controlled by the micro-controller is a closed system. The power consumed by it is fixed to 125mA @ 3.3V.
Using a low power micro-controller that runs at 3V is a straightforward way to reduce power dissipation. The micro-controller of choice should also support advanced power management features. Especially clock frequency control (section B.3.1) is important, because it enables the system to work at a minimal operating frequency, but ensures correct functionality.
To achieve further power savings external memory which is optimised for minimal power dissipation is required. In general this results in increased access time, but this can be tolerated since the update rate for new data is only 1Hz. The internal memory must be able to hold most of the data that is necessary to compute the received data. Data transfer to the external memory is done burst like.

Introducing these techniques to the Black Box System should provide significant savings of power.

4. Data Processing

Besides interfacing other system components, the main task of the micro-controller is the processing of data. To clarify the design, this can be split up into four main components (see figure 4.1).
The first component is the GPS receiver front end. Since the GPS receiver is a standalone device that only sends data, the front end only catches RMC messages out of the various messages sent by the receiver. Extraction of data is done by

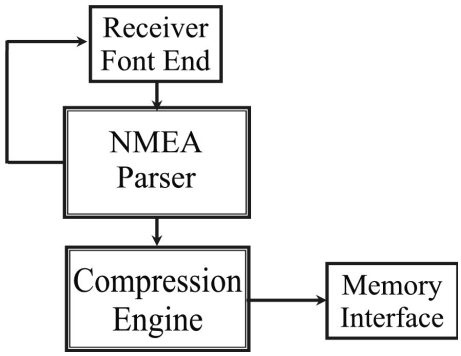


Figure 4.1.: Main Software Components

the NMEA parser which is the core element and has to perform several other equally important functionalities. First, it has to ensure that the computed data is correct. Second, the parser has to perform a number of unit conversions to aid further computation and finally it passes the computed data to the Compression Engine.

The compression function uses relative positions and an appropriate internal representation to reduce the total amount of data. The decision whether to use an absolute or a relative position depends on different threshold values, like the distance traveled, changing speed and course or a maximum number of relative positions in a row. The compression algorithm and related issues are discussed in chapter *5 Data Compression*.

To prove the effectiveness of the introduced compression routine a number of tests in different environments has been accomplished. GPS data was recorded on different routes in Northern Ireland, Ireland and Germany to make statements on the overall data reduction in different environments. In chapter *6 Real World Testing* the effects of *Comprehensive Data Compression* on major roads, small country roads and in city traffic are compared and evaluated.

The memory interface controls the external memory and is currently not implemented. Its structure is dependent on the kind of memory that is used. In case of a MMC card, the communication can be established via SPI.

4.1. The GPS receiver front-end

The GPS receiver sends its data once every second at 4800 bps. Communication between the GPS receiver and the microcontroller is established over an interrupt triggered asynchronous serial interface (UART).

If a RMC message is received, a software flag which enables the data transfer from the receive buffer to a holding buffer is set. Processing of data from the holding buffer is done, while new data enters the receiver buffer. Two buffers are necessary, because otherwise data from other NMEA sentences would overwrite the received RMC message. The receiver front end is implemented in the ARM7 controller on the Olimex board.

4.2. NMEA Parser

The processes performed by the NMEA parser are shown in the flow diagram on figure 4.2. Relevant data is extracted from valid RMC sentences and the received data is converted to units appropriate for navigational computation. To reduce the total amount of data and therefore increase storage efficiency a combination of absolute and relative positions is stored. The NMEA parser and the compression algorithm are currently implemented in a program running on a laptop PC to perform offline analysis. Adjusting the code for an embedded system remains further work.

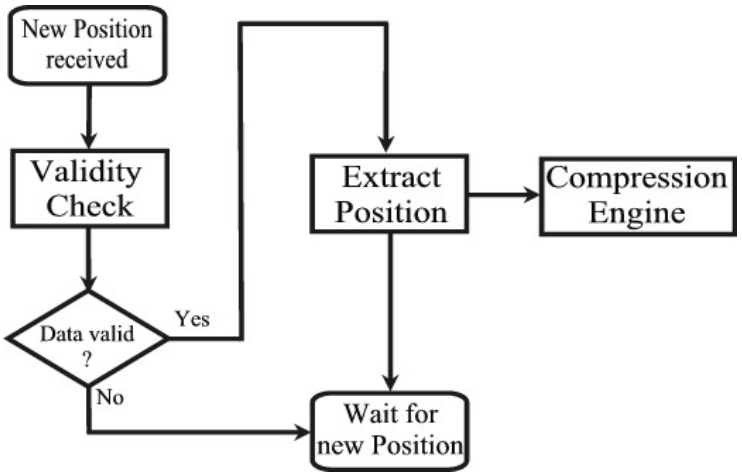


Figure 4.2.: Data processing in the micro-controller

The following sections give a detailed description of the software written to perform the above specified tasks.

4.2.1. Validity Check

A validity check is performed to ensure that only valid and complete RMC messages are accepted. To detect an incomplete message, a checksum is calculated

and compared to the checksum at the end of the message. The checksum is calculated as the XOR of bytes between, but not including the "$" and "*" (see listing 4.1). If the checksums do not match, the sentence is discarded.

```
int checksum(char *pmessage){
  char t1[3] = "\0", t2[3] = "\0", c, check = 0;

  pmessage++;                          // Skip the '\$'
  do{                                  // Loop through all chars
    c = *pmessage;                     // to get a checksum
    check = check ^ c;
  }while(*++pmessage != '*');          // exit loop on '*'

  sprintf(t1, "%X", check);            // convert to hex
  sprintf(t2, "%s", ++pmessage);
  return ( strncmp(t1, t2, 2) );       // return 0 when checksum
}                                      // is correct
```

Listing 4.1: Function to verify checksum

The GPS receiver itself also performs a validity check and sets a status byte when a locked position was obtained(A = valid, V = Invalid). Again, the whole message is rejected, if an invalid position is indicated.
This double validity check guarantees, that only accurate positions are used to keep track of the GPS Black Box.

4.2.2. Extracting data

After passing the validity check, the data from the RMC message is split into single words. Those parts that are necessary for further computation are stored in the structure shown in listing 4.2.

```
struct RMC    {          // Structure that holds RMC position data
              char time[7];
              double lat;
              char nsi;
              double lon;
              char ewi;
              double speed;
              double course;
              char date[7];
              };
```

Listing 4.2: Structure for parsed RMC data

Using a structure to hold all relevant data has the advantage that it can be easily expanded to hold data provided by other NMEA messages or additional sensors. E.g. the HDOP/VDOP/PDOP value[1] given by a $GPGSA message can be used to add advanced functionality. In the same way, data obtained from a sensor(temperature, radio signal strength etc.) attached to the Black Box can be appended for further computation.

The next step is to convert latitude and longitude to radians. This simplifies the calculation of the distance between two positions. The conversion algorithm and a short introduction into the basic theories behind navigation are given in the following section. Conversions between the different popular measurement systems and formulas to calculate distance and course on a spherical surface are introduced.

1 HDOP/VDOP/PDOP indicates a certain level of accuracy depending on the position and number of satellites in view

4.2.3. Navigation

On a spherical surface, like the world, we live on, the shortest distance between two points is not a straight line, but an arc with its centre at the centre of the earth[2]. In navigation, this arc is called the great-circle route[17].
In the following, important equations for calculation of distance and course from latitude and longitude (and reverse) are explained. Useful conversions between the different units used in modern navigation are given.

Unit Conversion

In order to simplify calculations and avoid confusion all angles should be converted to radians. Table 4.1 summarises conversions between radians, degrees and distance in kilometers (km) and nautical miles (nM)[3].

1 nM	$= \frac{40003km}{21600}$	= 1.8520 km
1 rad	$= \frac{180^o}{\pi}$	= 57.29577°
1 rad	$= \frac{40003km}{2 \cdot \pi}$	= 6366.71 km (3437.7 nM)
1°	$= \frac{\pi}{180}$	= 0.017453° rad
1°	$= \frac{40003km}{360}$	= 111.120 km (60 nM)
1 km	$= \frac{2 \cdot \pi}{40003}$	= 0.00015707 rad
1 km	$= \frac{360^o}{40003}$	= 0.00899927° (0.53996 nM)

Table 4.1.: conversion of angles and distances

2 For simplicity the earth is considered to be a perfect sphere. In reality it is flattened at the poles and a bit fatter at the equator. Therefore, for a conversion from spherical coordinates (geocentric) to ellipsoidal coordinates (geodetic) is performed. [10] gives a good introduction on transformations of different coordinate systems.

3 The nautical mile is currently defined to be 1852 meters, which implies the earth's radius to be 1.852 * (180*60/pi) = 6366.71 km

Latitude (DDMM.mmmm) and longitude (DDDMM.mmmm) are converted into radians using the algorithm displayed in listing 4.3 below. According to table 4.1, the global constant DEG_TO_RAD is 0.017453. Since the conversion from float to integer will cut off remaining decimal places, the accuracy of the position data is determined by the constant ACC. The default value for highest precision is ACC = 10^6. In order, to store the position information in a 16 bit integer, instead of an 32 bit long integer variable, it is reduced to ACC = 10^4, giving an position accuracy of 18.52 m.

```
// converts latitude / longitude into radians
int convert(double DDMM)
{
  double DD, MM;

  MM = modf( (DDMM/100), &DD );          // split in degree and minutes
  DD += MM / 0.6;                        // convert to DD.dddd

  return int( DD * DEG_TO_RAD * ACC ); // convert to radians
}                                        // ACC determines accuracy
```

Listing 4.3: Function to convert latitude / longitude to radians

Distance and Course

The distance between two points can be calculated from the angle of the great-circle route. This is straightforward, when both points lie on the same latitude or longitude, but for more complex routes the general formula 4.1 is used. The formula is derived from polar coordinate transformation[9].

$$dist = \arccos(\sin(lat1) \cdot \sin(lat2) + \cos(lat1) \cdot \cos(lat2) \cdot \cos(lon1 - lon2)) \quad (4.1)$$

Next is the calculation of the bearing from one point to another. It is defined as the angle measured horizontally from north to the current direction of travel. As outlined before, in navigation always true north is used, rather than magnetic north. With the distance d calculated beforehand, the true course is obtained according to formula 4.2[17]. Since on a sphere two solutions are possible, the result must be qualified by $\sin(lon2 - lon1) > 0$. If negative, the true course of the shortest distance is $2 \cdot \Pi - course$.

$$course = \arccos(\frac{\sin(lat2) - \sin(lat1) \cdot \cos(d)}{\cos(lat1) \cdot \sin(d)}) \tag{4.2}$$

Lat/lon from radial position, distance and course

Going the other way around, a radial position is calculated from a known position *lat1/lon1* and the distance d on a true course of c. First, the latitude *lat2* of the new position is calculated using formula 4.3 which is derived from formula 4.2.

$$lat2 = \arcsin(\sin(lat1) \cdot \cos(d) + \cos(lat1) \cdot \sin(d) \cdot \cos(c)) \tag{4.3}$$

Unless the new position is one of the poles[4] the longitude *lon2* is calculated as shown in formula 4.4[5].

$$lon2 = modulo(lon1 - \arcsin(\frac{\sin(c) \cdot \sin(d)}{\cos(lat2)} + \pi, 2\pi) - \pi \tag{4.4}$$

Equation 4.3 and 4.4 are taken from [30] which provides and explains a huge variety of formulas used in navigation.

4 This is the case if $\cos(lat2) = 0$

5 This algorithm is limited to distances such that $dlon = (lon1 - lon2) < pi/2$, i.e those that extend around less than one quarter of the circumference of the earth in longitude.

5. Data Compression

The route the Black Box has traveled is stored as a comma separated list in the external memory. A straightforward way to increase storage efficiency, is to omit all data not strictly necessary to identify a position. Therefore the approach was taken, to store relative positions, rather than absolute ones whenever possible. This is referred to as *Comprehensive Data Compression* and discussed in detail throughout section 5.1.
In order to maximise the data reduction, the internal representation of data (i.e. how the data is stored) is optimised. Section 5.2 describes this additional effect. The amount of position error introduced by the compression algorithm is analysed and evaluated respectively.

5.1. Comprehensive Data Compression

The data processed in the GPS Black Box system mainly consists of numerical positions in a fixed structure extracted from the appropriate NMEA sentence.

E.g. Latitude 4250.5589 S Longitude 14718.5084 E

As discussed in section 2.2, the accuracy of GPS positions is about ± 10 m. This implies that positions which are less than twice this distance apart from each other provide no new information.

Therefore, these positions need not be stored and can be omitted[1].
An additional reduction effect can be achieved, when only relative positions are stored rather than absolute ones. An absolute position is only recorded if speed and heading are changing. If both remain constant[2] the vehicle is considered to move at constant speed and heading on a straight line between the actual and the last position. Therefore the distance to the absolute position is the only information worth to be stored. Throughout this paper this is referred to as a relative position.

Position	Absolute	Relative
Time Stamp	X	X
Lat / Lon	X	-
Speed / Course	X	-
Distance	-	X
Additional sensor data	currently not implemented	currently not implemented

Table 5.1.: Data of absolute and relative positions

An absolute position consists of a time stamp used to uniquely identify each position, latitude, longitude, speed and course, whereas a relative position uses simply a time stamp and the distance to the last absolute position(see table 5.1). Additional sensors are currently not implemented, but may be used in further extensions of the design.
The method of data reduction by using absolute and relative positions is termed *comprehensive data compression*. Shown below is the realisation in C code (see listing 5.1).

1 In future extensions, where information obtained by additional sensors is processed, data obtained on positions within the GPS accuracy limits can be averaged to a mean value to increase the reliability of data.

2 The variations due to normal GPS inaccuracy have to be taken into account

```
...
dist = distance( rmc, dat );
if ( dist < DIST )       // skip entries nearer than DIST
{
  rmc = rmc->next;
  continue;
}
if (rel_count < LIM )    // More than LIM rel. pos. in a row?
{
  s_max = int( last->pos[2] * MAX );  // set threshold for
  s_min = int( last->pos[2] * MIN );  // speed and course
  c_max = int( last->pos[3] * MAX );
  c_min = int( last->pos[3] * MIN );

  dat = dat->next;
                    // check wether speed or course have changed
  if( (rmc->speed < s_max) && (rmc->speed > s_min) ...
      && (rmc->course < c_max) && (rmc->course > c_min) )
  {
    rmc = store_rel(rmc,dat,last,int(dist)); // store rel. pos.
    rel_count++;
    continue;
  }
}
rmc = store_abs(rmc, dat, last);       // store abs. pos.
rel_count = 0;
...
```

Listing 5.1: Comprehensive Data Compression

5.1.1. Consideration of Errors

One has to be aware that the use of a relative position instead of an absolute one will introduce an additional position error. This error can be tolerated if it stays within the limits of the normal GPS accuracy (±10 m). The positional error due to comprehensive data compression is depicted in figure 5.1.

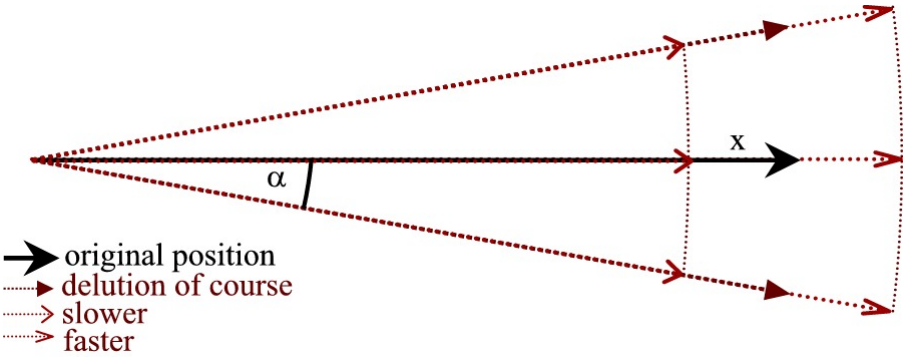


Figure 5.1.: Position error due to comprehensive data compression

The amount of error is depending on the threshold values used to detect a change in speed and course. The error due to a variation in course can be calculated according to formula 5.1[3]. For a distance of 20m and a dilution of 1° this yields an position error of $\sin 1^o \cdot 20m = 0.35m$.

$$e = \cos\alpha \cdot \sin\alpha \cdot x \approx \sin\alpha \cdot x \tag{5.1}$$

Formula 5.2 is used to calculate the error resulting from a variation in speed. Moving at a speed 1% higher than the reference speed, measured at the last absolute position, will result in a distance 1% smaller than estimated by the relative position.

$$e = \frac{v - v_r}{v_r} \cdot x = (v/v_r - 1) \cdot x \tag{5.2}$$

3 for small angles $\cos\alpha \approx 1$

Obviously, the distance x is a critical variable that amplifies both kinds of error. This is, why the number of relative positions in a row has to be limited as well. A range of different settings for the compression parameters described here are applied throughout the real world tests that are described in chapter 6

5.2. Optimise Internal representation

The internal representation of a processed value is quite important, since it determines the amount of memory that is necessary to hold a certain value. To reduce memory requirements as much as possible, an optimal representation has to be found. The optimal notation is different for each kind of data and is depending on the nature and range of values, as well as on the accuracy, that has to be achieved. The following sections describe the representations that were found to minimise memory requirements for Latitude / Longitude, Time Stamp and for Speed / Course.

Latitude and Longitude

Consider the following as an example for a position obtained from a GPS receiver. For demonstration purposes an extreme position close to the south pole is chosen.

```
Original Format:  Latitude 8959.5555 S    Longitude 17955.6666 E
```

As outlined in section 2.2.2 these values are in the format DDMM.mmmm and need to be converted to DD.dddddd to obtain meaningful values[4].

```
Degree(DD.ddddd): Latitude 89.992592 S    Longitude 179.927777 E
```

4 DD equals degrees(°), MM minutes(') and .mmmm and .dddddd are decimal fractions of a minute / degree

By now, latitude and longitude come as a floating point value, that will require 32 bit each when stored as a *float* variable. Converting the values to integer by multiplying it with 10^6 leads to:

Integer [°]:	Latitude 89992592 S	Longitude 179927777 E

This conversion has no reduction effect, because an 16 bit *integer* variable is limited to values up to 2^{16} (65536). Instead a *longinteger* variable is needed which again requires 32 bit.

The only way to achieve a reduction of memory requirements is to reduce the resolution, and therefore the accuracy of the position data. Since NMEA data is accurate up to 4 decimal places (DDMM.mmmm), the highest numerical accuracy achievable with NMEA data is 0.1852 m[5]. Cutting off the last two decimal places will increase the numerical inaccuracy up to 18.52 m, which could just be tolerated. However, this still gives no reduction in memory requirement.

As shown in the conversion table[6], 1° equals a distance of 111.12 km. Supposed, that the distance covered by the Black Box during a single measurement is smaller than 1111.2 km, the digits representing 100° and 10° can be stored as a global reference position and only the inner digits D.dddd need to be computed for every position but the very first. In order to simplify the calculation and avoid rounding errors the position angle is stored in the format MMM.mm. Thus, only one digit needs to be converted from degree to minutes.

Global position:	Latitude	80°	S	Longitude	170° E
Local position:	Latitude	599.55'	S	Longitude	595.66' E
as integer:		59955'			59566'

Now the position angles with an accuracy of ±18.52 m can be stored as 16 bit *integer* variables, thus yielding a reduction in size by as much as 50%.

5 see the conversion table 2.3 in section 2.2.2

6 table 4.1 on page 39

Time Stamp

Date and time information available from a RMC message comes in the format DDMMYYYY, and hhmmss.ss respectively. Since, the update rate of the NMEA data is once per second, the decimal fractions of a second can be thrown away. Like in the Windows and Unix time format a method using a global time base is used to reduce the amount of data. In this case, the time base is obtained from the first valid RMC message and stored as a global value.
Using a 16 bit *integer* variable to hold the relative time information gives a maximum period of measurement of

$$\frac{2^{16}}{60 * 60} = 18.2h$$

before a timer overflow occurs. This should be sufficient for most applications. Where continuous tracking over a longer period is necessary, a function to detect the overflow and increment a counter has to be implemented.

Speed and Course

Speed and course are easy to deal with. They both come in the format XXX.x, where speed is in knots (nM/h) and course in degree. Conversion of the speed value is done using the conversion factor from table 4.1 on page 39:

$$1nM/h = 1.852km/h$$

As mentioned in section 2.1.1, most GPS receivers indicate a movement at small speed and varying course during a stationary measurement. This can be explained by the GPS inaccuracy and the nature of the internal receiver algorithm to calculate speed and heading.

To minimize such unwanted ”stationary movement” effects, speeds smaller than 3 km/h are set to zero. This implies, that the decimal fraction of the speed value need not be considered. Thus speed can be stored as a 16 bit *integer* without further computation.
The course value is multiplied by 10 and stored as 16 bit *integer* as well.

5.3. Summary

The compression algorithm that is considered throughout this project uses relative positions to reduce the total amount of data. The current implementation is illustrated by the flow diagram in figure 5.2.
In section 5.1.1 a range of parameters that determine the magnitude of the position error has been identified. Namely, the threshold values to detect a changed speed or course and the distance that lies between the relative position and the absolute position used as reference.
Different settings of threshold values have been applied to analyse their effects on the overall position error. The results are discussed in chapter *6 Real World Test*.

To achieve an optimal representation of the positional data, a global reference position and a global time base are introduced. Both are derived from the first valid RMC message that is received and stored in a 64 bit header as shown in table 5.2.

Data field	global position	time base
number of bits	32	32

Table 5.2.: Memory requirements for global header

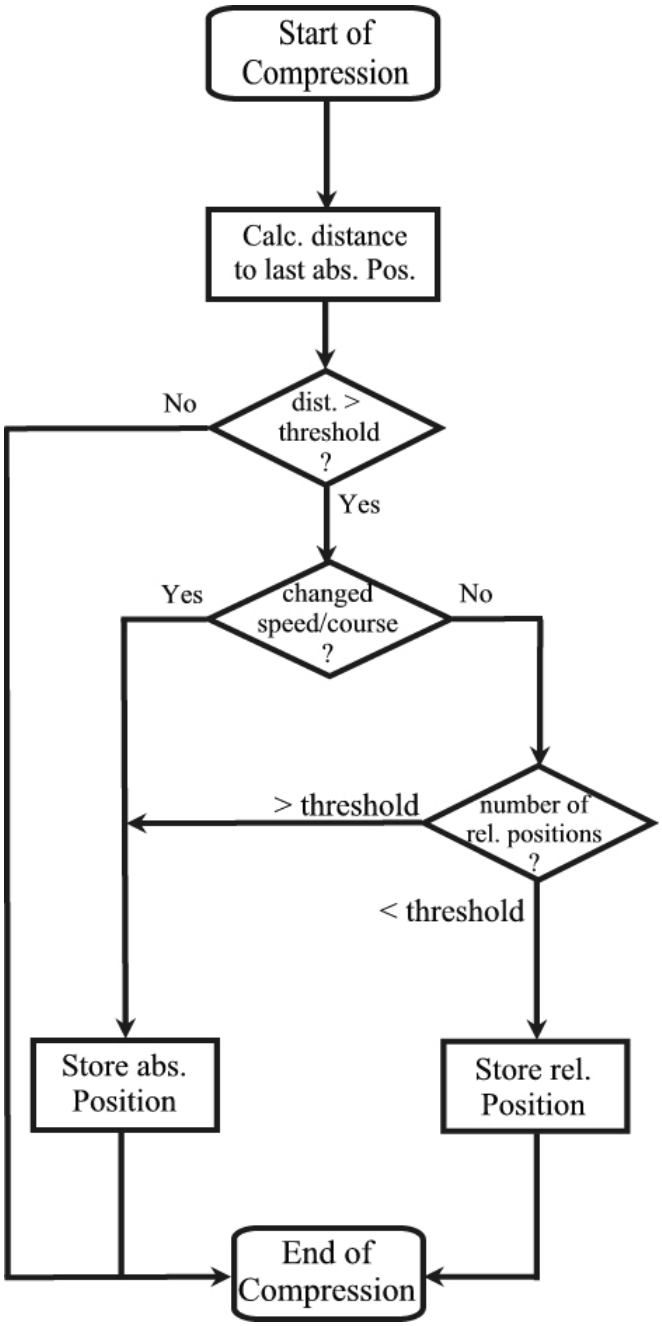


Figure 5.2.: Integration of comprehensive data compression

The memory requirements for absolute and relative positions are summarised in table 5.3. It turns out, that a relative position requires less than half the memory of an absolute one.

Position	Absolute	Relative
Status	8 bit	-
Lat / Lon	32 bit	-
Speed / Course	32 bit	-
Time Stamp	16 bit	16 bit
Distance	-	16 bit
Total	88 bit	32 bit

Table 5.3.: Memory requirements for absolute and relative positions

Since the compression algorithm is currently implemented on a Windows PC that uses 32 bit integer values by default, a reduction effect due to the optimisation of internal representation could not be observed. It will become obvious when implementing the code into an embedded system.
However, the overall reduction effect is greatly dependent on environmental conditions, like quality of roads frequency of sharp turns etc. These matters will be considered throughout the next section.

6. Real World Testing

To analyse the reduction rate that can be achieved with comprehensive data compression, GPS measurements on different environments have been made. GPS data has been recorded while driving on a highway in Germany and a number of other routes in Ireland and Northern Ireland. The different samples taken are summed up in table 6.1 The amount of data variation on these tracks will

Country	Route	Road Type[1]
GER	Würzburg - Augsburg	H
IRE	Westport - Achill Island	R, S
NI	Belfast - Bundoran	P, A
IRE	Dublin City	C
IRE	Galway - Cliffs of Moher	R, S
IRE	Galway - Clifden	R, S

Table 6.1.: Routes covered with measurements

determine the utility of the compression routines described above. While the effect is expected to be rather small when driving in a city with lots of turns and stop and go traffic, the *Comprehensive Data Compression* method has great potential to reduce the amount of data when driving on straight overland roads.

1 H = Highway, R = Regional road, S = Secondary road, P = Primary Route, A = A road

6.1. Data Analysis

The data was recorded and analysed offline on a laptop PC. The NMEA messages sent by the receiver are stored in a text file, which is fed to a C-program that implements the compression routine as described in section 5. The program writes the parsed and compressed data in a comma delimited **.csv* file to aid further analysis. This approach has the advantage, that different settings of compression parameters can be readily applied and analysed[2].

As outlined in section 5.1.1, the applied thresholds for distance, speed and course can be directly related to the position error. Therefore these parameters are implemented as global constants, that can be adjusted as required. To consider their effects on the absolute position error a number of different settings are analysed. The settings applied are summarised in table 6.2. The initial setting is

setting	distance threshold	speed threshold	course threshold	limited # of rel. positions
Initial	20 m	1 %	1 %	-
Setting 1	20 m	1 %	1 %	10
Setting 2	20 m	0.5 %	0.5 %	10
Setting 3	20 m	0.5 %	0.5 %	-

Table 6.2.: Compression settings

used as reference to make statements on the inaccuracy due to data compression. Setting 1 introduces a limit of maximal 10 relative positions in a row and leaves the threshold values untouched. To increase the accuracy of the position data, setting 2 halves these thresholds. Setting 3 is used to verify the results obtained by the former two.

Of course, this is only a basic approach that shows the functionality of the software

2 The source code is found in Appendix D.1. It is also provided on the project CD attached to the report.

used to analyse the data. The Matlab functions used here are designed to make the evaluation of different settings as comfortable as possible. As shown in listing 6.1 two vectors with filenames are required. The first one contains the compressed files, while the second holds their uncompressed counterparts.

```
% specify file name(s)
% compressed files
file_c={'gps1.csv' 'gps2.csv' 'gps_achill.csv' 'gps_bundoran.csv'...
        'gps_dublin.csv' 'gps_mayo.csv' 'gps_moher.csv'};

% uncompressed reference files
file ={'gps1.xls' 'gps2.xls' 'gps_achill.xls' 'gps_bundoran.xls'...
       'gps_dublin.xls' 'gps_mayo.xls' 'gps_moher.xls'};

    % call read_in function
    data_c = read_in( file_c , 1);

    ds = size(files);              % dynamic scaling factor
    for t = 1:ds(2)                % loop through all filenames
        data{t} = xlsread( file{t} );
    end
    disp('reading of files complete!')
    ...
```

Listing 6.1: Specification of files for analysis

The Matlab script reads in the generated files to automatically carry out further analysis. It uses dynamic three dimensional cell arrays to scale automatically with the size of the samples that are analysed.
According to the formulas for the reverse calculation, stated in section *4.2.3 Navigation*, absolute positions consisting of latitude and longitude are calculated from a relative position and an absolute reference position(see listing 6.2).

```
%..............................
function[] = calc()
%..............................
% nested inside var() (level 2)
    d = double( sample{3} )/RAD;    % distance to last abs. pos.
    % calculate absolute positions
    lat2=asin(cos(tc)*sin(d)*cos(lat1)+sin(lat1)*cos(d));
    if( cos( lat2 ) == 0 )
        lon2 = lon1;                % endpoint is a pole
    else
        lon2=mod(lon1-asin(sin(tc)*sin(d)/cos(lat2))+pi,2*pi)-pi;
    end
end
% end of nested function calc (level 2)
```

Listing 6.2: Matlab function to restore an absolute position

These 'restored' values are compared to the original data to obtain the position error due to data compression. To accomplish this, the relative time information of a relative position is used to calculate the index pointing to the corresponding absolute position.

The resulting absolute error values are plotted against their time stamp to show the accumulation of position error under different conditions.

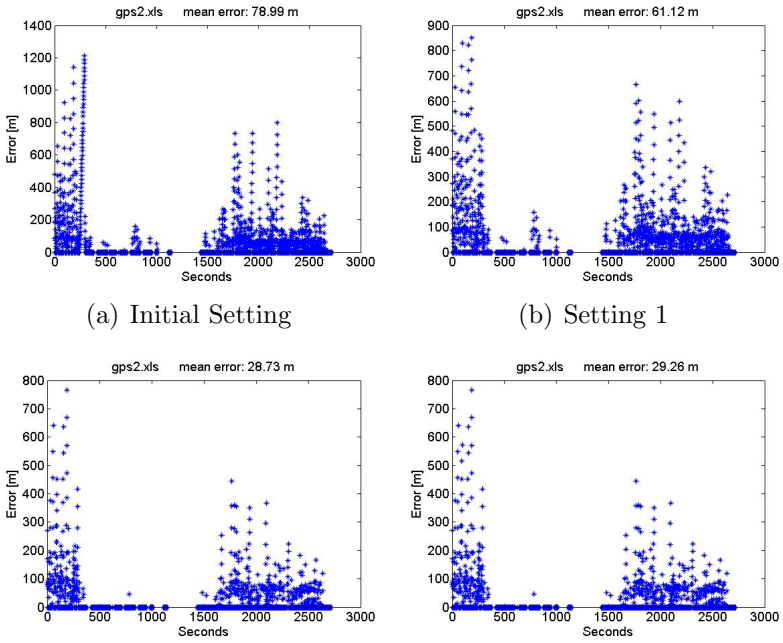
All Matlab code generated throughout this project is found in Appendix D.2. The source code for the implementation of the compression algorithm is found in Appendix D.1.

6.2. Results

The following figures display the accumulated position error due to *Comprehensive Data Compression*. Each figure contains four subfigures, each representing a different setting of parameters. Only the most expressive ones are stated here. A complete set is found in Appendix C.

Judging the usability of comprehensive data compression, two facts are of interest. The first is the absolute magnitude of the error value, which represents the greatest error that has to be expected.

Rather than these sporadic high error values which might be ignored, the frequency of high errors is interesting too. This reveals information on the averaged accuracy of the system which is included on top of each figure.



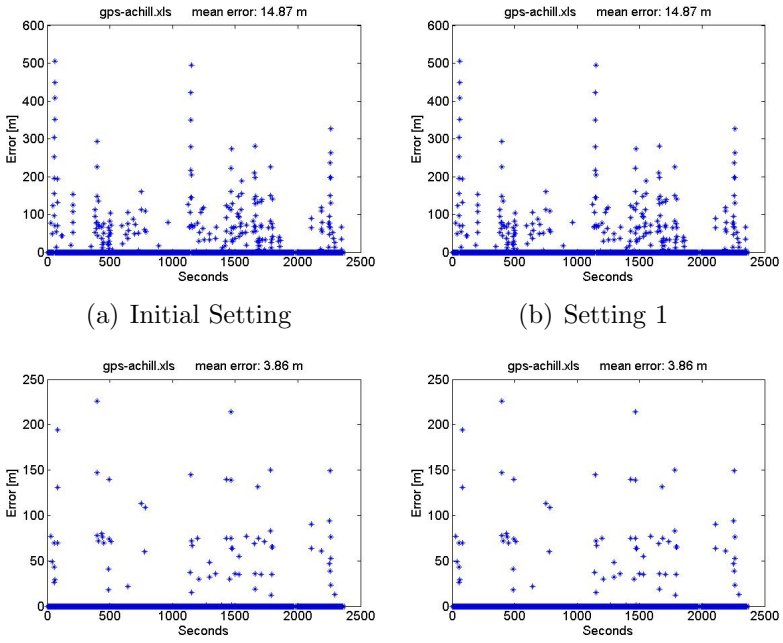
(a) Initial Setting (b) Setting 1

(c) Setting 2 (d) Setting 3

Figure 6.1.: Position error on German highway (Augsburg - Bad Tölz)

The considerably high position error of the initial setting (6.1(a)) is reduced by 30%, by limiting the number of relative positions (b). However, the frequency of high errors is not reduced. Applying stricter bounds to the speed and course values seems to be more effective(c). (d) inicates that the effect of the limitation is vanishing compared to the threshold values.

A similar relation can be seen at the figures 6.2 and 6.3. The effect of the limitation proves to be even less pronounced. This can be explained by the fact, that the velocity of the vehicle is directly proportional to the position error.

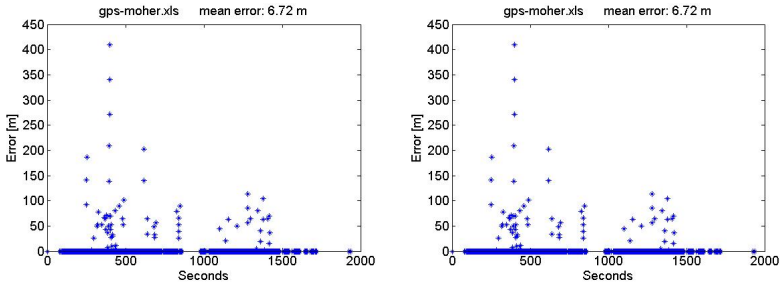


(a) Initial Setting (b) Setting 1

(c) Setting 2 (d) Setting 3

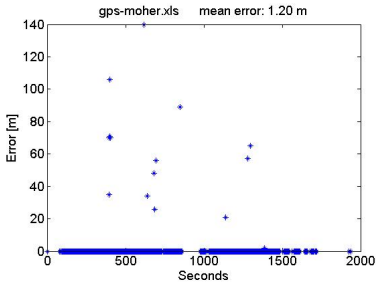
Figure 6.2.: Position error on a regional road on Achill Island (IRE)

Considering figure 6.4 which was recorded in Dublin's city traffic clearly confirms this observation. The speed, the Black Box moves with seems to be a substantial source of error. This applies to the magnitude of error as well as on its frequency.

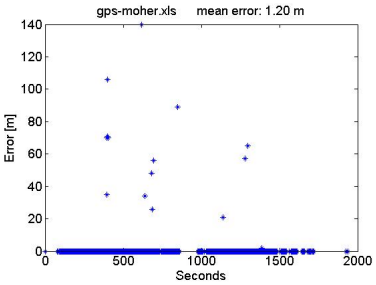


(a) Initial Setting

(b) Setting 1

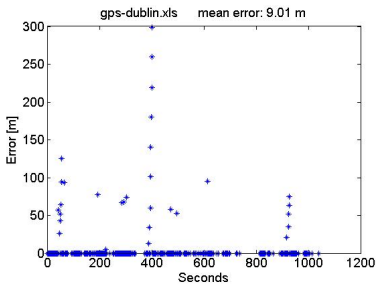


(c) Setting 2

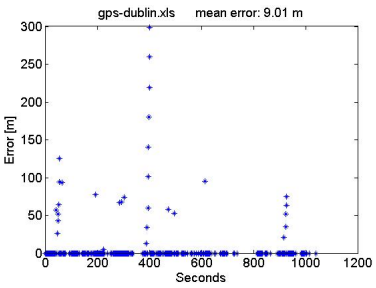


(d) Setting 3

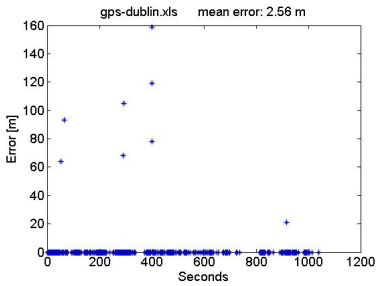
Figure 6.3.: Position error on regional and secondary road Moher (IRE)



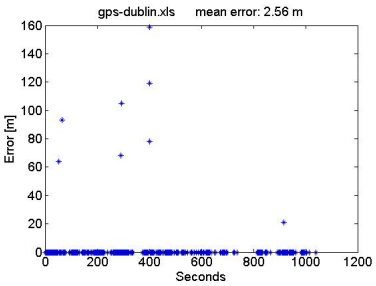
(a) Initial Setting



(b) Setting 1



(c) Setting 2



(d) Setting 3

Figure 6.4.: Position error in city traffic in Dublin (IRE)

6.3. Consequences

The analysis discussed above clearly shows a relation between the threshold values for speed and course and the overall position error. Except for highways, the limitation of consecutive relative positions proves to have no, or only marginal effects on the maximum error value (see table 6.3). The resulting variation of position is proportional to the speed of the vehicle, the Black Box is mounted on[3].

	Initial max./mean	Setting 1 max./mean	Setting 2/3 max./mean
Highway 2 (D)	1200/79	<800/61	<750/29
Achill Island	500/15	<450/15	<250/3.9
Dublin (City)	300/6.7	<300/6.7	<160/1.2
Cliffs of Moher	420/9.0	<420/9.0	<140/2.6

Table 6.3.: resulting position error [m]

While a limitation of relative positions only cuts off extreme error values, a reduction of all errors regardless their size is achieved by applying tighter boundaries to the threshold values. Table 6.3 clearly shows, that although the max. error is quite high, the average inaccuracy of compressed positions is reasonable low.
To decide, which levels of position deviation can be justified is not within the scope of this project and depends highly on the actual GPS application.
Further analysis of this subject is necessary and can be conducted using the software created while working on this project.

3 This was already stated in section 5.1.1

Table 6.4 shows the file sizes before and after compression has been accomplished. The routes in Germany and the overland routes in Ireland / Northern Ireland show a similar reduction rate of roughly about 44 %. The reduction rate for

Route	uncompressed size [kB]	compressed size [kB]	reduction rate [%]
Highway 1 (D)	140	67	47.8
Highway 2 (D)	100	41	41.0
Achill Island	100	41	41.0
Bundoran	125	58	46.4
Dublin	43	9	20.9
Connemarra / Mayo	76	28	36.8
Burren / Cliffs of Moher	69	24	34.8
average red. rate			38.4

Table 6.4.: Compression results on different routes

the data taken on small regional roads and in city traffic is even better, which is surprisingly on the first look. It can be explained with the lower speed in such environments. Due to the distance threshold applied, more positions are discarded which results in a smaller file size.

Since no operating Black Box prototype was constructed, the results stated here are preliminary and require verification by a GPS device with embedded components. Especially the file sizes are likely to differ considerably in a microcontroller system.

7. Conclusion and Further Work

7.1. Conclusion

To obtain in depth knowledge on low power design techniques, a literature research has been conducted. It summarises currently approved methods to reduce power dissipation in embedded systems and is considered a guideline for the design of the *GPS Black Box* and future designs.

GPS data on several distinct routes in Germany, Ireland and Northern Ireland have been recorded. These samples have been analysed to determine the accuracy of position that is possible using the DELUO GPS receiver.
The stationary GPS inaccuracy proved to be around ± 10 m. It is shown, that the velocity of the traced vehicle is a parameter, that can be used to characterise a specific environment, namely highway, country and city traffic. It also determines the accuracy that is achievable.

Several unforeseen hardware problems prevented the construction of a prototype of the *Black Box* system. Communication with the initially chosen ATmega128 development board could not be established. However, all external components are available and this reports provides sufficient information to complete this task.

Section *3.1 Architecture* describes the overall system architecture and all subsystems used in the design. The receiver front end, that provides an interrupt driven interface to the GPS receiver is implemented in an ARM7 microcontroller. The Olimex LCD-E2294 development board provides all hardware necessary to carry out further implementation. Circuitry to design a sufficient power supply is provided.

C-Code implementing *Comprehensive Data Compression* has been written and verified on a laptop PC. The data samples taken to determine GPS accuracy were used to apply different settings of compression parameters. Evaluation of these experiments was carried out using automated Matlab scripts.

Evidence is given, that memory requirements can be reduced to 40 % by applying the compression algorithm. Except on highways[1], the mean position error due to *Comprehensive Data Compression* is below 5m.

The C- and Matlab functions created throughout this project to analyse compressed and uncompressed GPS data can be used to carry out further optimisation of the compression algorithm.

7.2. Further Work

This report summarises all design steps to build up a GPS Black Box as an embedded system implementing *Comprehensive Data Compression*. Due to time restrictions, the construction could not be concluded. The following sections give a brief description of the steps to complete the construction process.

1 Due to the higher velocity on highways the position error increases as well, but stays below 30m.

7.2.1. Construction of Black Box Prototype

As outlined above, the construction of a Black Box has been left unfinished. All external components are available and need only be assembled and connected. The ARM7 Olimex LCD-E2294 development board provides sufficient memory and I/O ports to build up a prototype that demonstrates the correct functionality of the design.

7.2.2. Implementation of Software into Embedded System

The GPS receiver interface is already implemented into the ARM7 microcontroller. The NMEA parser and the compression engine are realised in C-code running on a Windows laptop PC. The code needs to adjusted to the special requirements of an embedded system. The complete source code is available in Appendix D.1 as well as on the attached project CD.

7.2.3. Integration of GSM

As already outlined in section *1.3 Potential Applications*, the integration of a GSM modem would make the Black Box design useful for a whole range of applications. It gives the system the ability to establish a bidirectional communication which makes additional features like remote control possible.

Cellular Communication

Using the cellular modem technology has several advantages. Most notably, most modern computers already come with an integrated modem that can be used to talk directly to the cellular modem in the device simply by dialing up the phone number. The only draw back is that the user will probably have to pay a cellular service charge every month they want to operate the Black Box.

In order to keep the charges for operating the system as low as possible, it has to be evaluated whether the SMS or the GPRS service[2] suits the communication requirements best. This is highly dependent on the actual application and the local cellular service charges. Table 7.1 gives a general outline on both services.

	SMS	GPRS
Charges	based on connected time	based on amount of data
Transmission	Message center using store and forward mechanism	TCPIP
Connection	creates new connection every time	always connected
# of connections	every connection requires an own channel	multiple channels simultaneously

Table 7.1.: Overview on cellular services

Technical Aspects

When integrating a GSM modem, a few important points need to be considered. First of all is the question of interfacing the new subsystem. Since most GSM modules come with a serial RS232 interface, a microcontroller that provides two USARTs is advantageous. Otherwise either a second USART can be implemented by software, or a multiplexer can be used to control whether the GPS- or the GSM-module have access to the serial port.

The second task is to provide sufficient power for the GSM modem. Especially when setting up a connection, the modem is likely to demand currents up to 2 Ampere. Although this is only for a few microseconds, the PSU has to be designed accordingly[31].

2 SMS: Short Messaging Service, GPRS: General Packet Radio Service

List of Figures

List of Tables

A. Hardware

A.1. DELUO GPS receiver Specification

1-877-885-9090 http://www.deluo.com support@deluo.com

INTEGRATED GPS RECEIVER

The GPS receiver is a miniature, 12-channel, fully integrated, low-power GPS receiver. It includes on-board patch antenna, RS-232/PS2/USB driver, and is intended for fast integration into GPS-enabled products. The receiver is high performance and low cost, designed specifically for high-volume GPS applications.

Due to its reduced size, fast time-to-first-fix and low power consumption, the GPS receiver may be used as the location sensor in GPS-enabled products.

Features

1. **12 parallel channel, fast satellite acquisition**
2. **Low power consumption, miniature size**
3. **Integrated antenna and RS-232 /PS/2 or USB interface**
4. **On board RTC and SRAM data sustained by rechargeable battery to provide fast time to first fix under normal operation**

1-877-885-9090 http://www.deluo.com support@deluo.com

Applications

- Land / Marine Navigation
- Telematics
- Fleet Management
- Asset Management

Specifications:

Features	Description
General	L1 frequency, C/A code, 12-channel
Sensitivity	-165 dBW minimum
Update Rate	1Hz
Accuracy	Position: 25m CEP without S/A
	Velocity : 0.1/sec without S/A
Acquisition	Cold start: < 90sec
	Warm start: < 40sec
	Hot start: < 15sec
Reacquisition	<100msec
Dynamics	Altitude: -1000m to 18000m
	Velocity: 500m/sec
	Acceleration: ±4g
Operation Temperature	-20 to +75
Storage Temperature	-55 to +90
Operating Humidity	5% to 95%
Primary Power	+3.8V~8V DC
Power Consumption	105mA
PC Interface	RS232
	PS/2
	USB
Protocol	NMEA 0183 v2.20@4800/9600 baud, 8-None-1
Datumns	218 standard datumn, default WGS-84
Antenna	Built-in patch
NMEA Message	GGA, GLL, GSA, GSV, RMC, and VTG
Dimension	55mm x 40mm x 20mm
Weight	115g

Specifications are preliminary and subject to change without notice.

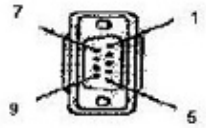
ELECTRONICS 1-877-885-9090 http://www.deluo.com support@deluo.com

GLOBAL POSITIONING SYSTEM

INTEGRATED GPS RECEIVER

Serial/PS2 Version

Connectors



DB9 RS232 Connector

Pin	FUNCTION	Pin	FUNCTION
1	NC	2	RX
3	TX	4	NC
5	GND	6	NC
7	NC	8	NC
9	NC		

PS/2 Connector

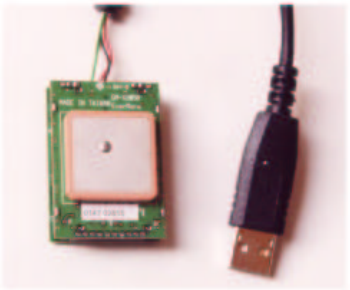
Pin	FUNCTION
1	NC
2	NC
3	GND
4	VCC (+5V)
5	NC
6	NC

ELECTRONICS 1-877-885-9090 http://www.deluo.com support@deluo.com

GLOBAL POSITIONING SYSTEM

INTEGRATED GPS RECEIVER

USB Version



A.2. DataFlash Card

Features

- **MultiMediaCard (MMC) Form Factor**
- **Single 2.7V to 3.6V Supply**
- **20 MHz Max Clock Frequency**
- **Serial Peripheral Interface (SPI) Compatible**
- **Low Power Dissipation**
  - **4 mA Active Read Current Typical**
  - **2 µA CMOS Standby Current Typical**
- **Industrial Temperature Range**

Description

The AT45DCB008, AT45DCB004 and AT45DCB002 are 2.7-volt only SPI compatible serial interface DataFlash® Cards. They are offered in 8-megabyte (64-Mbit), 4-megabyte (32-Mbit) and 2-megabyte (16-Mbit) densities. These 7-pin cards are form factor compatible to the MultiMediaCard standard leveraging on a wide range of low cost connectors that are commercially available. Additionally, the DataFlash Card is pinout compatible to the SPI version of the MMC card.

The small form factor and simple interface make the DataFlash Card ideal for a wide variety of data- and program code-storage applications where a removable storage medium is required. Applications range from voice/audio, text or image storage to software applications, system upgrades and software patches.

For detailed device specifications, please refer to the corresponding DataFlash datasheets:

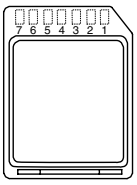
- AT45DCB008 reference AT45DB642
- AT45DCB004 reference AT45DB321B
- AT45DCB002 reference AT45DB161B

Pin Configuration

Pin #	Pin Name	Function
1	$\overline{CS}$	Chip Select
2	SI	Serial Input
3	GND	Ground
4	VCC	Supply
5	SCK	Serial Clock
6	NC	No Connect
7	SO	Serial Output

DataFlash Card

Top View



8-megabyte, 4-megabyte, and 2-megabyte 2.7-volt Only DataFlash® Cards

AT45DCB008
AT45DCB004
AT45DCB002

Rev. 3257A–11/01

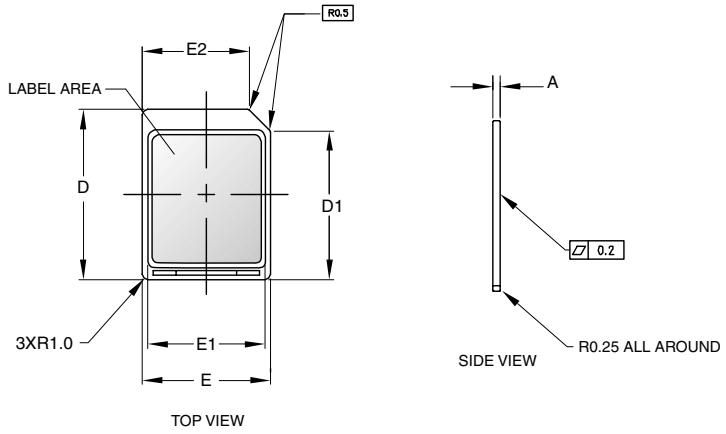
Ordering Information

f_{SCK} (MHz)	I_{CC} (mA)		Ordering Code	Package	Operation Range
	Active	Standby			
20	10	0.01	AT45DCB008 AT45DCB004 AT45DCB002	7DF1	Industrial (-40°C to 85°C)

Package Type	
7DF1	7-lead, 2.5 mm Pitch 24 x 32 x 1.4 mm Body DataFlash Card

Packaging Information

7DF1 – DataFlash Card



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	1.30	1.40	1.50	
B		2.40 TYP		
C		4.00 TYP		
D	31.90	32.00	32.10	
D1		28.00 REF		
E	23.90	24.00	24.10	
E1		22.00 REF		
E2		20.00 REF		
E3		15.00 TYP		
F		20.00 REF		
G		15.00 TYP		
e		2.50 TYP		

10/09/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**7DF1**, 7-pad, 2.5 mm Pitch 24 x 32 x 1.4 mm Body DataFlash Card	7DF1	A

Atmel Headquarters

Corporate Headquarters
2325 Orchard Parkway
San Jose, CA 95131
TEL (408) 441-0311
FAX (408) 487-2600

Europe
Atmel SarL
Route des Arsenaux 41
Casa Postale 80
CH-1705 Fribourg
Switzerland
TEL (41) 26-426-5555
FAX (41) 26-426-5500

Asia
Atmel Asia, Ltd.
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimhatsui
East Kowloon
Hong Kong
TEL (852) 2721-9778
FAX (852) 2722-1369

Japan
Atmel Japan K.K.
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
TEL (81) 3-3523-3551
FAX (81) 3-3523-7581

Atmel Product Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

Atmel Heilbronn
Theresienstrasse 2
POB 3535
D-74025 Heilbronn, Germany
TEL (49) 71 31 67 25 94
FAX (49) 71 31 67 24 23

Atmel Nantes
La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 0 2 40 18 18 18
FAX (33) 0 2 40 18 19 60

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Atmel Smart Card ICs
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-357-000
FAX (44) 1355-242-743

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

3257A–11/01/0M

B. Low Power Design

In the past, improving circuit speed and reducing circuit area have been the main objectives for advances of digital technology. The emerging market for portable computing and communication devices and increasing difficulties in providing cooling for high performance CPUs has meant that reducing power consumption has become a major factor for the design of digital systems. This is perhaps no more apparent than in the case of medical implantable devices, but it holds for any battery operated application. [33]

In addition, as transistors get smaller and smaller, there is a technological need for circuits operating on low currents. While size is proportional to the propagation delay, this is true for the maximum current it can withstand without getting damaged, too. This fact forces vendors of highly integrated systems to optimise their devices for operating at low power.

Designers of modern digital devices can obtain several benefits from low power design:

- Improve the lifetime of batteries
- Reduce the risk of overheating high integrated devices
- Increased reliability
- Next generation circuits can be adopted with lesser effort

In the following sections different types of power dissipation are identified, in order to gain an understanding of where power is consumed and how this can be optimised. Based on this knowledge, various methods to reduce power consumption are described and evaluated. A summary of methods to reduce power consumption which are applicable to the black box system will conclude the chapter.

B.1. Power Consumption

Most micro-controllers are fabricated using CMOS technology. To a first order, the power consumption of CMOS circuitry is given by the formula:

$$P = ACV^2F \tag{B.1}$$

If a capacitance of C is charged and discharged by a clock signal of frequency F and supply voltage V, then the charge moved per cycle is CV and the charge moved per second is CVF. Since the charge packet is delivered at voltage V, the energy dissipated per cycle, or the power, is CV^2F. The power for a clocked flip-flop, which can toggle at most once per cycle, will be $\frac{1}{2}CV^2F$. When capacitances are clock gated or when flip-flops do not toggle every cycle, their power consumption will be lower. Therefore, a constant called the activity factor ($0 \leq A \leq 1$) is used to model the average switching activity in the circuit.[34]

Equation B.1 suggests that there are essentially four ways to reduce power:

- Reduce the supply voltage, V
- Reduce the capacitive load, C
- Reduce the switching frequency, f
- Reduce the activity rate A

Based on this equation a number of methods to reduce power dissipation are obvious. The ones that seems to be most effective are outlined in chapter B.2 on page 84.

B.1.1. Classification

To achieve a more detailed understanding of how power dissipation in digital systems can be minimized, it is important to identify parts of the system which are responsible for increased power dissipation. Based on this knowledge attempts can be made to cut down energy dissipation to a minimum level. The following classifications will give an overview on the different types of power.

[33] contains a valuable classification of energy in digital systems because it is based on the way each kind of energy can be optimised. The three types are:

- **Decision Energy** is consumed while performing logical operations. Most of the energy is consumed turning transistors ON and OFF. It is highly dependent on the activity of a system. This is also referred to as dynamic power.
- **Storage Energy:** is used to store and retrieve data from memory elements. Often a considerably amount of power is dissipated in sense amplifiers in order to reduce access times. To minimize this special attention is paid to the selection of low power memory (See Chapter *Storage* on page !! To Do !!
- **Communication Energy:** is required to transmit data from one place in the system to another. Main components are wires, drivers and receivers. Power dissipation could be minimised if data that is required more frequently is kept in the internal storage.

Apart from the above definitions [32] contains another classification. According to Ronald J. Landry there are three types of power consumption relevant to digital circuits(See Table B.1).

Dynamic Power		Static Power
Internal Power	Switching Power	Leakage

Table B.1.: Classification according to [32]

Internal power and switching power are referred to as dynamic power, because consumption of energy is highly dependent on system activities, like toggling signals, switching outputs etc. Leakage on the other hand can be regarded as static power, because it is only dependent on system layout and operating conditions. The different types are described in detail in the following subsections.

The following sections will give an explanation of the above introduced types of power consumption(table B.1). In section *Power Consumption in the Black Box System* on page 90 the major power consumers in the BlackBox-System are identified and discussed.

B.1.2. Switching Power

Switching power is perhaps the easiest to understand. As the output of a cell changes (switches) the parasitic capacitance associated with that node must either charge or discharge. The currents used to deliver this charge come from the power supply and therefore translate directly to power consumption. If the outputs do not change there is no charging or discharging and therefore no associated switching current.

An example of this is the approach taken for a simple counter. A Gray code counter will by definition switch fewer outputs and therefor consume less power than a binary counter of the same bit width.[32] Another example to reduce switching power on address lines is given in section B.3.2.

B.1.3. Internal Power

Internal power is quite similar, but it requires a closer look into the cell. Even if the outputs of a cell do not change their state, energy is consumed every clock cycle. This is because the clock signal which is provided throughout the whole system toggles on every clock cycle. These nodes have a capacitance so current flows in the same way as described for switching power.

Power savings can be made by working on a low clock frequency or using clock gating.[32]

B.1.4. Leakage

Leakage occurs because transistors are never perfect. Even when they are in the OFF state, the conductance of the channel does not go to zero. Leakage is highly dependent on the geometry of the transistor. This is why leakage is a major factor in nanometer designs. As the channel gets shorter the conductance from drain to source gets larger. One can think of this as one resistor rather than several in series.

It is important to be aware that faster designs tend to have more leakage as well. This is because speed is achieved by increasing the drive of the cell. The higher drive is achieved by increasing the width of the channel. One can think of this as putting several resistors in parallel. While this approach decreases the ON resistance it also decreases the OFF resistance, which allows more leakage current to flow in the OFF state.[32] The importance of minimising leakage depends highly on the activity of the system.

Voltage thresholds also have a drastic effect on leakage. This relationship can be seen in figure B.1. A lower Vt corresponds to higher leakage. In many processes

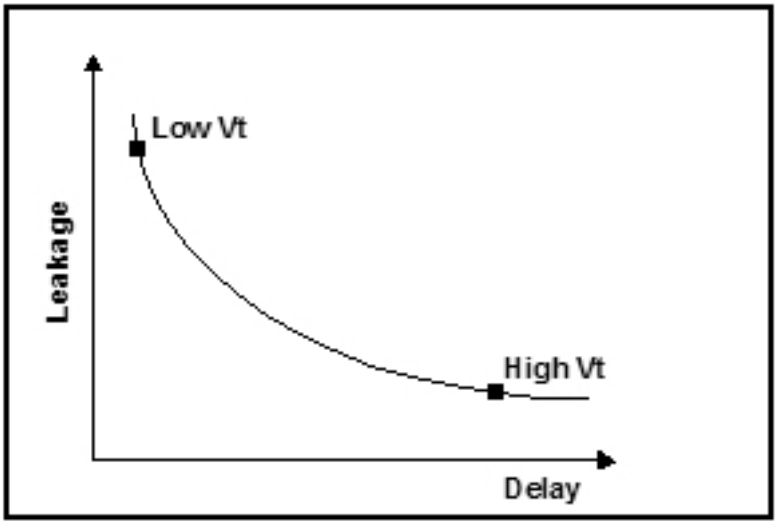


Figure B.1.: Scaling V_t

Vt is minimised to increase speed or lower the minimum supply voltage requirements. For low power designs where leakage is a major factor, this needs to be taken into account.

B.2. Methods of Power Minimisation

The previous sections have given an overview on the different types of power in digital systems. Based on the classification of power consumption a few hints to reduce power dissipation are given.

An important point to consider are the inter-workings of static and dynamic power consumption. There might be decisions that trade off one for the other. E.g. if there is not much computation or communication, that means system activity is low, then minimising of leakage becomes important. If, on the other hand, the system is busy most of the time, larger leakage currents can be tolerated and may lead to a reduction in overall power consumption.

An assembly of methods that are frequently used in modern system designs is shown below.

- Operating on low supply voltage
- Using built in power management features
- System design and programming techniques

Especially the built in power management features seem to have the greatest potential to reduce power consumption in the black box system. The various methods are described in detail in the following sections.

In section *Power Consumption in the Black Box System* on page 90 the different techniques are evaluated depending on their utility in the black box system.

B.3. Low Voltage Operation

As seen from equation B.1, power consumption is proportional to the square of supply voltage, which means that reducing a processor's supply voltage can result in dramatic savings in power. As a result, most vendors now have three volt versions of their processors. In some cases, these are actually five volt parts that happen to run at the lower voltage. In such cases the system clock must typically be reduced to ensure correct operation. In other cases, vendors have introduced "'true"' three volt versions of their processors that run at the same speed as their five volt counterparts.[35]

Voltage Scaling

A more advanced approach is the technique of *Voltage scaling*. It requires a sophisticated voltage regulator but can provide a significant extension to battery life. Temperature, process and voltage all act together to affect the speed of a

device. As temperature increases, speed decreases, and cells slow down as voltage decreases. A process may vary the speed of a cell by as much as ±50%. These relationships can be used to compensate for one another. If a temperature sensor is on-die, the device can read it's own temperature and adjust its voltage, to meet the application's timing requirements. With the proper voltage regulator, this lower voltage will translate into noticeable power savings[32].

This method is hardly used in general purpose micro-controllers, because it requires extra hardware(temperature sensor, voltage regulator) and detailed libraries containing curves for voltage, temperature and process variations. Also, estimating battery life can be very difficult when using voltage scaling.

B.3.1. Power Management Features

Most micro-controllers today have a number of built-in hardware or software features, designed to give the programmer better control over the processor's power consumption. These are referred to as power management features. The most common power management features are described below.

Sleep and Idle modes Sleep or idle modes typically turn off the clock to all but certain sections of the processor to reduce power consumption. Because CMOS power consumption is proportional to signal toggling frequency, turning off the clock to the processor can greatly reduce power consumption. While asleep, the processor does no work. A wake-up event (usually an unmasked interrupt) wakes the processor from the sleep mode.

Of course, if the interrupt can come from on-chip peripherals, this implies that these peripherals are active and receiving a clock signal, resulting in increased power consumption. As a result, some processors have two, but usually more

sleep modes: one in which peripherals are enabled, and one in which they aren't. In the former, any unmasked interrupt wakes the processor. In the latter, only an unmasked interrupt occurring on an external I/O pin wakes the processor. [35]

Terms, functionality and return events for sleep modes are different depending on the manufacturer of the micro-controller. Below is a summary of common functionalities that can be halted to reduce power consumption. Because of the huge variety of available micro-controllers this may not be complete.

- associated on-board oscillator circuitry
- watchdog logic
- clock monitor
- timer/counter functions
- analog comparator
- internal voltage reference
- pins of I/O port
- clock distribution to peripherals

One thing to consider when bringing a core out of sleep mode is the oscillator startup time. Normal oscillator architectures can take several milliseconds to stabilize. In many cases this may be longer than the core will stay active before going back to sleep. Increasing power consumption due to this fact can be minimised by using a quick-start oscillator.

Brownout Protection Brownout protection is usually an on-board protection circuit that resets the device when the operating voltage *Vcc* is lower than the brownout voltage. The device is held in reset and will remain in reset when *Vcc*

stays below the Brownout voltage. The device will resume execution from reset after *Vcc* has risen above the brownout voltage.

Clock Frequency Control Although many applications require a processor's maximum processing speed from time to time, few applications require maximum speed at all times. Instead, there may be periods during which the processor requires a fraction of its normal execution speed to perform some tasks. Due to this fact many micro-controllers offer control over clock frequency by software. The two most common clock control mechanisms are clock dividers and internal low-speed clock sources.

In that way clock frequency can be aligned to meet application specific timing targets without waste of energy by performing a task faster than required. This represents a compromise between full speed operation and a sleep mode. [35]

Control over unused Outputs and Peripherals A number of micro-controllers provide output pins that make the processor's clock available to other components in the system. Some chips allow the user to disable such outputs if the output signals are not needed. This can be particularly important for clock output signals: even though the capacitive load on the pin may be small, the clock frequency is usually quite high, resulting in significant increased power consumption. Turning off the system clock to unused peripherals is a straightforward way to reduce power consumption. This is also know as *clock gating* on system level (additional information about *clock gating* can be found in source [36]).

B.3.2. System and Programming Techniques

There are a number of system design or programing techniques that can be used to reduce power consumption. Usually these require the use of assembler based programs. The most important ones are outlined below.

Avoid External Accesses As stated above (equation B.1), power consumption in CMOS circuitry is proportional to capacitance. Connections to external components, such as external memory, typically have much greater capacitance than connections to on-chip resources. As a result, accessing external memory can increase power consumption. If a program frequently accesses external memory for certain values, those values should be moved to on-chip memory (if available) to reduce power consumption.

Avoid Unnecessary Logic State Transitions Consumption of power increases proportional with the frequency at which signals change (equation B.1). This is true for every signal path in a system whether it is a clock signal, a data pin, or an address line. Therefor writing code that reduces changing of states will noticeable reduce power dissipation. As previously pointed out, a counter implemented in Gray Code will consume less switching current. Another example, is given by the two short programs below:

```
          Program A
0x0fff  loop:  add  x0,a
0x1000           jmp  loop

          Program B
0x0000  loop:  add  x0,a
0x0001           jmp  loop
```

Although the programs are identical, (A) will consume more power when executing than (B). This is because it changes the address bus from 0x0fff to 0x1000 and back again each time through the loop, which requires toggling 13 bits from *0* to *1* and back. In contrast, the code in (B) changes only one bit from *0* to *1* and back again each time through the loop. [35] Especially in loops with many iterations or frequently used address sequences this will lead to significant power savings.

B.4. Power Consumption in the Black Box System

In the previous sections various methods to reduce power dissipation in digital designs have been described. The next step is to analyze the Black Box System in order to be able to choose the most effective ones. The design goal is to minimise power consumption as much as possible, but guarantee proper functionality.

The Black Box System mainly consists of three subsystems:

- the GPS module, which delivers standard NMEA sentences at an update rate of 1 Hz
- the GSM module, which sends periodical status SMS and is used as a sensor for the signal strength of nearby mobile masts
- the micro-controller, which computes and stores received data and controls the whole system

The GPS and GSM modules are closed systems interacting with the micro-controller. Therefore the power consumed by these modules is fixed. One way to reduce dissipation is to shut them down after data has been received. To determine the utility of this approach the exact timing behavior is of interest and has

to be evaluated. In a further approach these modules might be replaced by ones that are optimised for low power consumption.

Table B.2 summarises the power minimisation methods that seem to have the greatest potential, the required hard- or software and the expected power savings.

Minimisation Method	**required hard-/ software**	**expected power savings**
Low Voltage Operation	-	significant
Sleep Modes	wakeup logic	depending on idle time
Clock Frequency control	frequency divider low speed clock source	depending on system utilisation
Reduce external accesses	internal memory	depending on access frequency
Reduce logic state transitions	programming style (i.e. Gray Code)	in loops or frequently used sequences only

Table B.2.: Summary of Power Minimising Techniques

Using a low power micro-controller that runs at 3V or lower is a straightforward way to reduce power dissipation. The micro-controller of choice should also support advanced power management features. Especially clock frequency control (section B.3.1) is important, because it enables the system to work at a minimal operating frequency, that ensures correct functionality. A global sleep mode that halts all modules should be introduced as well. This could be done using the brownout protection feature (sectionB.3.1).

To achieve further power savings external memory which is optimised for minimal power dissipation is required. In general it results in increased access time, but this can be tolerated since the update rate for new data is only 1Hz. The internal memory must be able to hold most of the data that is necessary to compute the received data.

Special attention has to be paid, when addressing external data or subsystem interfaces. As outlined in section B.3.2, a decrease in power dissipation can be achieved if frequently logic state transitions are avoided when ever possible.

Introducing these techniques to the Black Box System should provide significant saving of power.

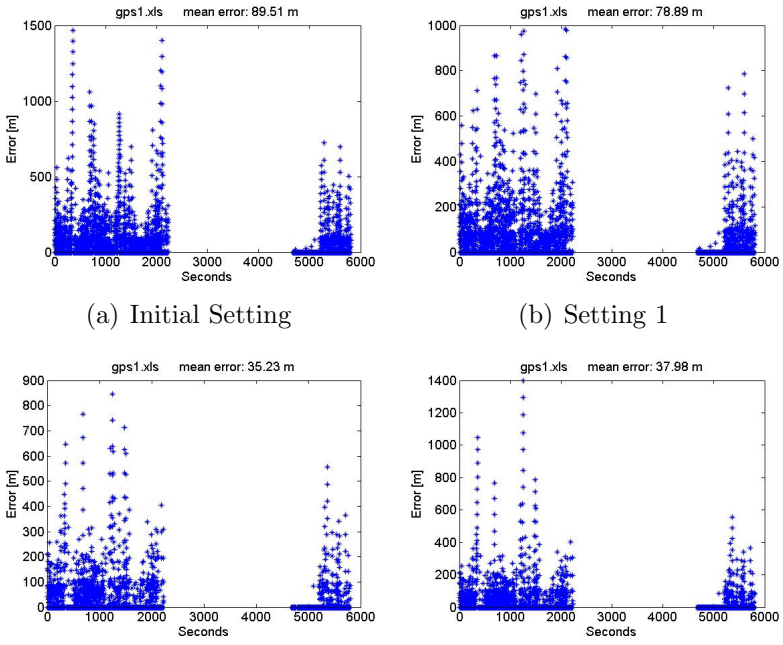
B.5. Summary Low Power Design

Throughout this chapter, different types of power consumption have been analyzed and several methods to reduce power dissipation have been discussed. The most effective techniques in regard to the Black Box System are evaluated.

In general, sleep modes should be used as much as possible, and the sleep mode should be selected so that as few as possible of the device's functions are operating. All functions not needed should be disabled. An additional effect is provided by lowering the clock frequency to a minimum level, without affecting timing requirements. The alignment of the clock frequency to processor utilisation provides minimal power consumption for a given task.

Special attention is spend on the selection of memory. This point is discussed in chapter

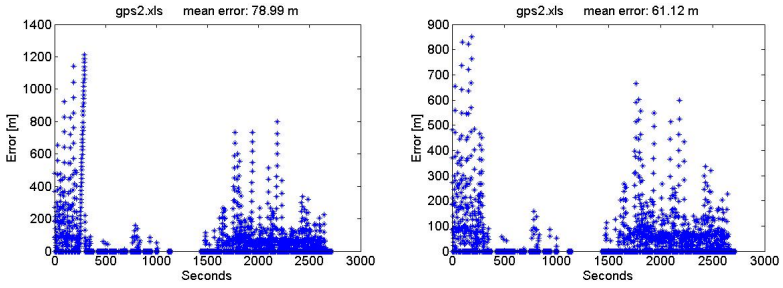
C. Complete Results of Realworld Test



(a) Initial Setting (b) Setting 1

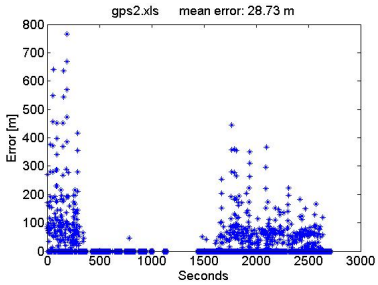
(c) Setting 2 (d) Setting 3

Figure C.1.: Position error on German highway (Würzburg - Augsburg)

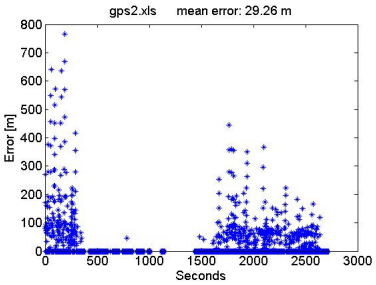


(a) Initial Setting

(b) Setting 1

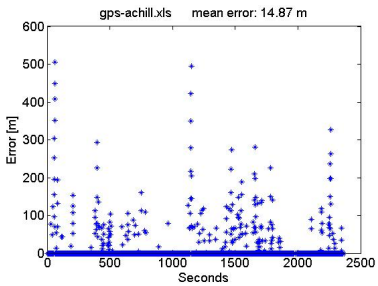


(c) Setting 2

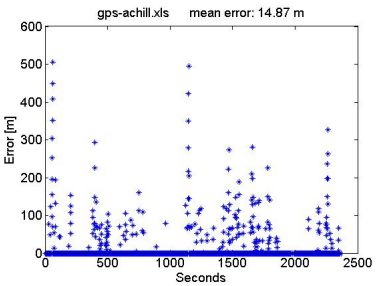


(d) Setting 3

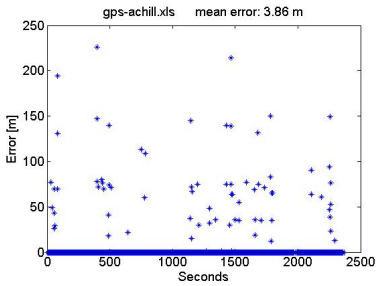
Figure C.2.: Position error on German highway (Augsburg - Bad Tölz)



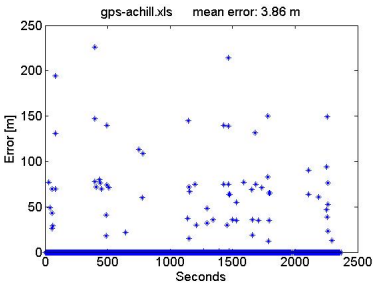
(a) Initial Setting



(b) Setting 1

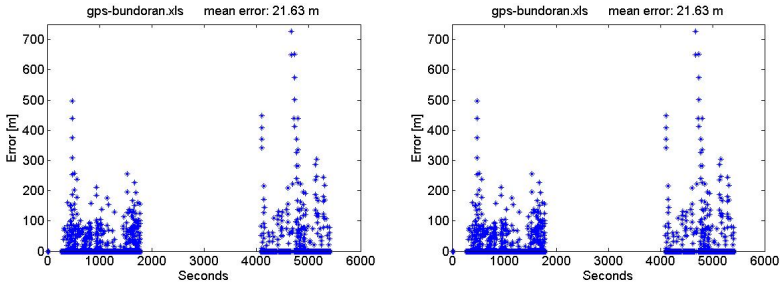


(c) Setting 2



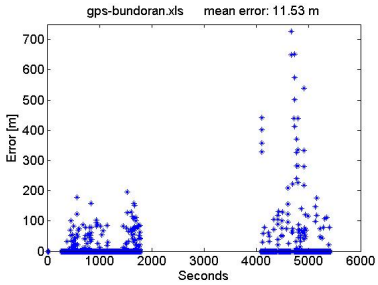
(d) Setting 3

Figure C.3.: Position error on a regional road on Achill Island (IRE)

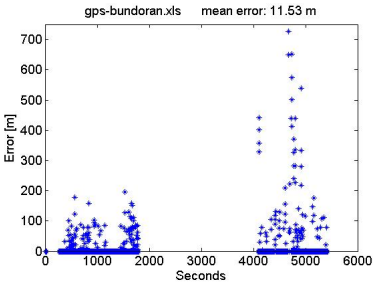


(a) Initial Setting

(b) Setting 1

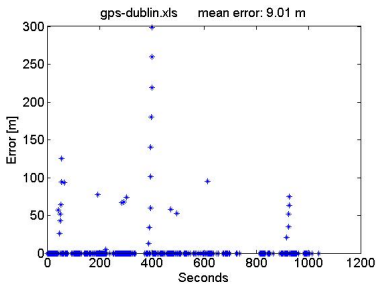


(c) Setting 2

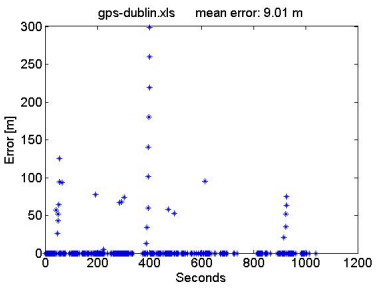


(d) Setting 3

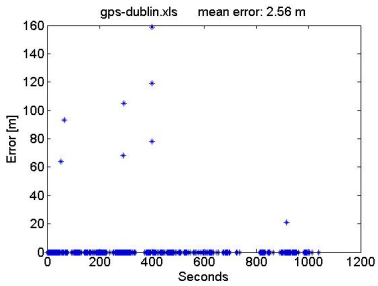
Figure C.4.: Position error on primary route and A-roads to Bundoran (NI)



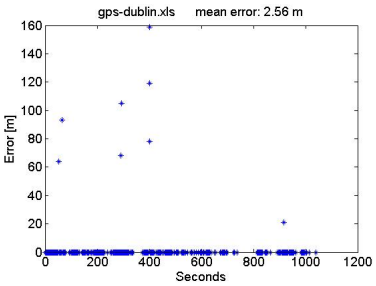
(a) Initial Setting



(b) Setting 1

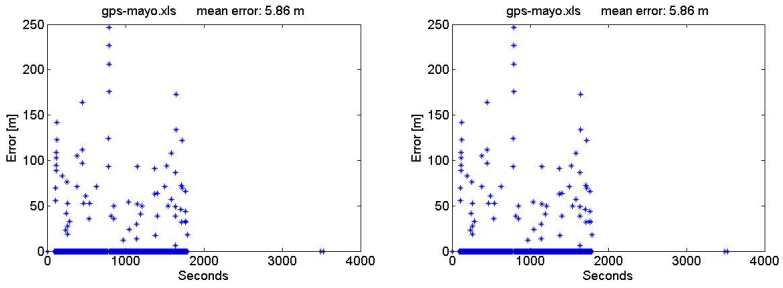


(c) Setting 2



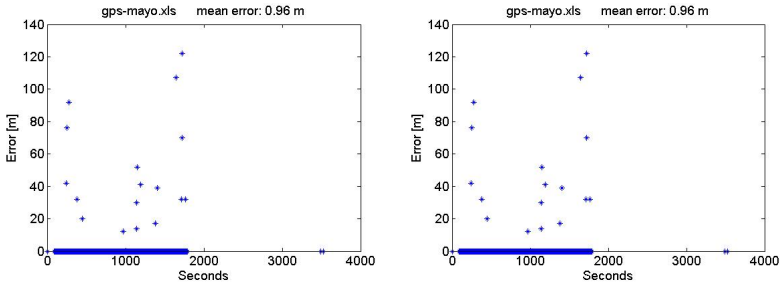
(d) Setting 3

Figure C.5.: Position error in city traffic in Dublin (IRE)



(a) Initial Setting

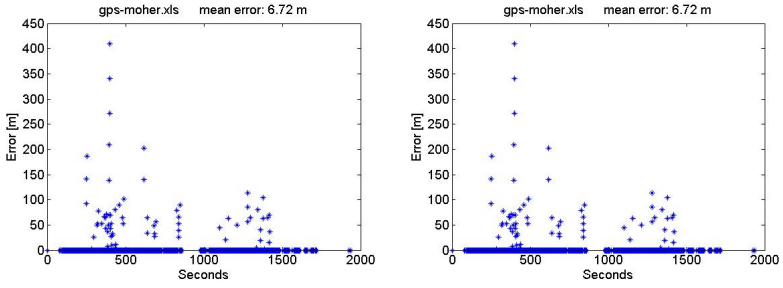
(b) Setting 1



(c) Setting 2

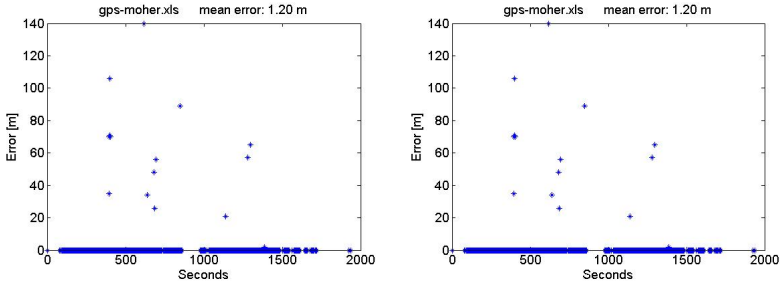
(d) Setting 3

Figure C.6.: Position error on regional and secondary road Connemara (IRE)



(a) Initial Setting

(b) Setting 1



(c) Setting 2

(d) Setting 3

Figure C.7.: Position error on regional and secondary road Burren (IRE)

D. Software

D.1. Implementation of NMEA Parser and Compression Engine

D.1.1. C Header

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
/* Constants */
#define DEL ","                    // define delimiter
#define ACC 1000000                // Determines accuracy
#define DEG_TO_RAD 0.0174532925 // conversion factor

/* Structure Definition */
struct gprmc  {           // Structure holding raw RMC data
        char nsi, ewi;
        char utc_t[7], utc_d[7];
        int lat, lon, speed, course;
        struct gprmc *next,*prev,*last,*first;
        };                // fully linked list
```

```
struct entry  {           // Structure holding compressed data
        int time, a_r;    // (a_r==1): absolute / (a_r==0): relative
        int pos[4];       // 4 data fields
        struct entry *next, *abs; // linked list
        };

struct header {           // Global Header
        char date[7], time[7];
        int lat_glob, lon_glob, timebase;
        };
typedef struct gprmc tgprmc;
typedef struct entry tentry;
typedef struct header theader;

/* Function prototypes */
FILE *input(int, char *[]);     // open file with raw NMEA data
tgprmc *parse(FILE *, tgprmc *);// parses file for valid RMC
                                   sentences
tentry *compress(tgprmc *);     // compress position data
void rmc_out(tgprmc *);         // write raw RMC data to .xls file
void compress_out(tentry *);    // write compressed data to .csv
                                   file
/*
parse() subfunctions:
********************/
int convert(double);            // converts lat / lon to radians
int checksum(char *);           // calculates and compares checksum
/*
compress() subfunctions:
************************/
// store absolute position
tgprmc *store_abs(tgprmc *, tentry *, tentry *);
// store relative position
```

```
tgprmc *stroe_rel(tgprmc *, tentry *, tentry *, int dist);
// calculates distance between two positions
int distance(tgprmc *, tentry *);
/*
store_abs() subsubfunctions:
**************************/
int ctime(char *, char *);          // converts date/time to seconds
                                       since TIMEBASE (from global
                                       header)
```

D.1.2. C Sourcecode

```
/***********************************/
/*  Michael Bachner                */
/*  Final Year Project 06          */
/*  GPS Black Box                  */
/*  Main file                      */
/***********************************
Functional Description:
0: open .txt file containing NMEA data
1: extract RMC sentences out of NMEA data
2: carry out validity check
3: extract position data from RMC message
4: store raw position data in *.xls file
5: compress position data
6: store compressed data in *.csv file
***********************************/
#include "gps.h"
/* Constants for Data Compression */
#define DIST  20                  // Min. distance between two stored
                                     positions
#define MAX   1.01                // Speed & course are allowed to
                                     change by +/-0.5%
#define MIN   0.99                // before an absolute position is
                                     used
#define LIM   10                  // Max. number of relative positions
                                      in a row
/* Global variables */
double dist;
theader Header;                   // Global Header(holds initial pos,
                                     time, date and Timebase)
/* Main Funtion */
int main (int argc, char *argv[])
```

```
{
  FILE *fp = NULL;
  tgprmc *Liste = NULL;
  tentry *data = NULL;
  char mode, new;
  do{
    fp = input(argc, argv);      // call input() function
    Liste = parse (fp, Liste);  // call parse() function
    if( Liste )
    {
      printf("Specify output\n-> n: RMC data\n-> c: compressed ");
      scanf("%c", &mode);
      switch (mode)
      {
        case 'n':
          printf("Generate output file (.xls)...\n");
          rmc_out(Liste);        // call rmc_out function
          break;
        case 'c':
          printf("Generate compressed output file (.csv)...\n");
          data = compress(Liste); // call compress function
          compress_out(data);
          break;
        default:
          printf("\nInvalid command!\nNo Output file generated!\n");
      }
    }
    else
      printf("\nNo valid data found!\n");

    printf("\n\nExit program? (e)\n");  // Run Again?
    scanf("%c", &new);
  }while(new != 'e');
```

```
    printf ("\n\n\nProgrammende!\n\n");
    return 0;
}
```

```
/*
input function
opens file with raw NMEA data
*/
FILE *input(int argc, char *argv[])
{
  char filename[20] = "\0", dummy;
  FILE *fp = NULL;
  if (argc > 1)                    // when called from command line
  {
    if ((fp = fopen (argv[1], "r")) == NULL)     // open file
          printf ("\nError while accessing file!\n\n");
  }
  if(fp == NULL)
  {
    while (1)
    {
      printf ("Specify file name:\t");
      scanf ("%s", filename);
      scanf ("%c", &dummy);
      strcat (filename, ".txt");     // Append file ending '.txt'

      if ((fp = fopen (filename, "r")) == NULL)    // open file
      {
        printf ("\nError while accessing file!\n\n");
        continue;
      }
      break;
    }
  }
  return fp;
}
```

```
/*
parse function
extracts valid RMC sentences from raw NMEA data
stores position data in a fully linked list (struct gprmc)
parse() subfunctions:
*********************/
int convert(double);          // converts lat/lon to radians
int checksum(char *);         // calculates and compares checksum
tgprmc *parse (FILE *fp, tgprmc *Liste)
{
  char zeile[100] = "\0";
  int i, t=0;
  tgprmc *p;
  if ( (Liste = (tgprmc *) malloc(sizeof(tgprmc))) == NULL)
    printf("ERROR: Memory allocation failed !!!");

  p = Liste;
  p->first = p;
  for (i = 0; fgets( zeile, 100, fp ) != NULL; i++) // Read data
  {
    if ( strncmp( "$GPRMC", zeile, 6) == 0 )    // GPRMC sentence
    {
      if ( 'A' == (zeile[18]) )        // check GPS status
      {
        if (checksum(zeile) == 0)      // test checksum
        {
          strtok(zeile,DEL);           // skip sentence ID
          strcpy( p->utc_t, strtok(NULL,DEL));      // UTC time
          strtok(NULL,DEL);            // skip status flag
          p->lat = convert( atof(strtok(NULL,DEL)));  // Latitude
          p->nsi = zeile[30];          // North/South indicator
          strtok(NULL,DEL);
```

```
        p->lon = convert( atof(strtok(NULL,DEL)));  // Longitude
        p->ewi = zeile[43];          // East/West indicator
        strtok(NULL,DEL);
        p->speed = int(atof(strtok(NULL,DEL))*10);  // Speed
        p->course = int(atof(strtok(NULL,DEL))*10); // Course
        strcpy( p->utc_d, strtok(NULL,DEL));       // UTC date
        if ( (p->next=(tgprmc*)malloc(sizeof(tgprmc)))== NULL)
        {                            // Dynamic allocation of
                                            memory
          printf("ERROR: Memory allocation failed !!!");
          break;
        }
        p->next->prev = p;           // set pointer to previous
                                            entry
        p = p->next;                 // set pointer to next entry
        p->first = Liste;            // set pointer to first entry
        t++;
      }
    }
  }
}
fclose(fp);                          // close file
printf("\nFound %d valid RMC entries in %d NMEA messages",t,i);
if ( t > 0 )                         // set return pointer
{
  p = p->prev;                       // set pointer to previous
                                       entry
  p->next = NULL;
  while ( Liste != NULL )            // set pointer to last entry
  {
    Liste->last = p;
    if (Liste == Liste->last)
      break;
```

```
      Liste = Liste->next;
    }
    Liste->next = NULL;                    // set last pointer to NULL
    return p->first;                       // return pointer on first
                                             entry
  }
  else
    return NULL;
}
```

```
/*
checksum function
calculates and compares checksum
*/
int checksum(char *pmessage)
{
  char t1[3] = "\0", t2[3] = "\0", c, check = 0;
  int i=0;
  pmessage++;              // Skip the '$'
  do                       // Loop through all chars to get a checksum
  {
    if(((c=*pmessage)<0)||(i==100))    // message too long ??
      return -1;           // return -1 if invalid character occurs
    check = check ^ c;     // computing checksum (XOR)
    i++;
  }while(*++pmessage != '*');          // exit loop on '*'

  if((*(pmessage+1)<0)||(*(pmessage+2)<0))
    return -1;             // return -1 if invalid character occurs

  sprintf(t1, "%X", check);            // convert to hex
  sprintf(t2, "%s", ++pmessage);
  return(strncmp(t1,t2,2));    // return 0 when checksum is correct
}
```

```
/*
convert function
converts latitude / longitude into radians
*/
int convert(double DDMM)
{
  double DD, MM;
```

```
  MM = modf( (DDMM/100), &DD ); // split in degree & minutes
  DD += MM / 0.6;               // convert to DD.dddd

  return int(DD*DEG_TO_RAD*ACC);// convert to radians
                                // ACC determines accuracy
}
```

```
/*
compress function
uses absolute and relative positions to reduce data

compress() subfunctions:
************************/
// creates header that holds global data
void create_header(const tgprmc *);
// store absolute position
tgprmc *store_abs(tgprmc *, tentry *, tentry *);
// store relative position
tgprmc *store_rel(tgprmc *, tentry *, tentry *, int dist);
// calculates the distance between two positions
int distance(const tgprmc *, const tentry *);

tentry *compress(tgprmc *rmc)
{
  tentry *dat = NULL;
  tentry *last = NULL;
  int rel_count = 0, s_max, s_min, c_max, c_min;

  create_header(rmc->first);       // create global header

  if ( (dat = (tentry *) malloc(sizeof(tentry))) == NULL) // dynamic
                                        memoryallocation
```

```
  printf("ERROR: Memory allocation failed !!!");

rmc = rmc->first;                  // goto first entry
rmc = store_abs(rmc, dat, dat); // store first pos as abs pos
last = dat;

while (rmc->next != rmc->last)  // loop through all entries but
                                   the last one
{
  dist = distance( rmc, dat );
  if ( dist < DIST )               // skip entries nearer than DIST
  {
    rmc = rmc->next;
    continue;
  }
  if (rel_count < LIM )            // limitation of rel. pos.
  {
    // set threshold for speed & course
    s_max = int( last->pos[2] * MAX );
    s_min = int( last->pos[2] * MIN );
    c_max = int( last->pos[3] * MAX );
    c_min = int( last->pos[3] * MIN );
    dat = dat->next;

    // check wether speed or course have changed
    if( (rmc->speed<s_max)&&(rmc->speed>s_min)&&...
        (rmc->course<c_max)&&(rmc->course>c_min) )
    {
      // if no: store relative position
      rmc = store_rel(rmc, dat, last, int(dist));
      rel_count++;
      continue;                    // continue loop
    }
```

```
    }
    rmc = store_abs(rmc, dat, last);  // store abs. pos.
    last = dat;
    rel_count = 0;
  }
  dat = dat->next;
  rmc = store_abs(rmc->next, dat, last);  // last one is abs. pos.
  dat->next = NULL;
  while(dat != dat->abs)           // search for first entry
    dat = dat->abs;
  return dat;                      // return pointer to first entry
}
```

```
/*
create_header
creates header that holds global data (Timebase & First position)
*/
void create_header(const tgprmc *global)
{
  if(FIRST)                 // function is executed only once
  {
    strcpy(Header.date, global->utc_d); // Global Date
    strcpy(Header.time, global->utc_t); // Global Time
    Header.lat_glob = global->lat;      // Global Latitude
    Header.lon_glob = global->lon;      // Global Longitude
    Header.timebase = 0;                // Reset Timebase
    Header.timebase = ctime(global->utc_t); // calculate Global
                                            Timebase
    FIRST = false;          // lock further initialisations
  }
}
```

```
/*
```

```
store_abs function
stores an abs. position
consisting of latitude, longitude, speed, course
*/
tgprmc *store_abs(tgprmc *rmc, tentry *dat, tentry *last)
{
  dat->a_r = true;          // set abs. flag
  // assign full RMC data
  dat->time = ctime(rmc->utc_t);
  dat->pos[0] = rmc->lat;
  dat->pos[1] = rmc->lon;
  dat->pos[2] = rmc->speed;
  dat->pos[3] = rmc->course;

  dist = 0;                 // reset dist
  dat->abs = last;          // set pointer to last abs. pos.
  last = dat;
  // dynamic allocation of memory
  if ( (dat->next = (tentry *) malloc(sizeof(tentry))) == NULL)
    printf("ERROR: Memory allocation failed !!!");
  return rmc->next;
}
```

```
/*
store_rel function
stores an relative position consisting of the
distance to last position (relative or absolute)
*/
tgprmc *store_rel(tgprmc *rmc, tentry *dat, tentry *last, int dist)
{
  dat->a_r = false;         // clear abs. flag
  dat->time = ctime(rmc->utc_t);
  dat->pos[0] = dist;
```

```
  // dynamic allocation of memory
  if ( (dat->next = (tentry *) malloc(sizeof(tentry))) == NULL)
    printf("ERROR: Memory allocation failed !!!");

  dat->abs = last;
  return rmc->next;
}
```

```
// calculates the distance between the actual and the last absolute
                                    position
int distance(const tgprmc *rmc, const tentry *dat)
{
  double lat1;
  double lat2 = float(rmc->lat)/ACC;  // get RMC lat
  double lon1;
  double lon2 = float(rmc->lon)/ACC;  // get RMC lon

  if( dat->a_r )                      // get lat/lon from abs. pos.
  {
    lat1 = float( dat->pos[0] ) / ACC;
    lon1 = float( dat->pos[1] ) / ACC;
  }
  else
  {
    lat1 = float( dat->abs->pos[0] ) / ACC;
    lon1 = float( dat->abs->pos[1] ) / ACC;
  }
  dist = acos(  sin(lat1)*sin(lat2)+cos(lat1)*cos(lat2)...
              * cos(lon1-lon2) )*6366710;
  return int( dist );
}
```

```
/*
```

```
ctime function

converts date and time information into seconds since 01.01.06
*/
int ctime(const char *time)
{
  char hh[2], mm[2], ss[2];
  int h, m, s, out=0;

  strncpy(hh, time, 2);    // parse time string
  time += 2;
  strncpy(mm, time, 2);
  time += 2;
  strncpy(ss, time, 2);
  time += 2;

  h = atoi(hh);            // convert to integer
  m = atoi(mm);
  s = atoi(ss);

  out = out * 24 + h;      // sum everything up
  out = out * 60 + m;
  out = out * 60 + s;

  return (abs(Header.timebase-out));  return dif to TIMEBASE
}
```

```
/*
output function
outputs extracted valid, but uncompressed RMC data to .xls file
*/
void rmc_out (tgprmc *p)
{
```

```
int i;
char filename[40] = "\0", YN, dummy, buffer[42] = "\0";
FILE *fp = NULL;

while (1)
{
  printf ("\nSpecify filename for output file:\t");
  scanf ("%s", filename);
  scanf ("%c", &dummy);
  strcat (filename, ".xls");          // append file ending '.xls'

  if ((fp=fopen(filename,"r"))!=NULL) // file already existing?
  {
    printf ("\nFile already exists!\nDo you want...
             to overwrite it(Y/N)?\n\n");
    scanf ("%c", &YN);
    scanf ("%c", &dummy);
    if (YN == 'N' || YN == 'n')
      continue;
  }
  if ((fp = fopen (filename, "w")) == NULL) // try to open file
  {
    printf ("\nError while accessing file!\n\n");
    continue;
  }
  break;
}
// write header
sprintf (buffer, "Date\tTime\tLat\tLon\tSpeed\tCourse");
buffer[41]='\n';
if (fwrite(&buffer, sizeof(zeile), 1, fp) != 1)
  fprintf(stderr, "... Error on fwrite\n");
```

```
  for(i=0;i<41;i++)                    // clear buffer
    buffer[i]=' ';

  while (p != p->last)                 // write all data
  {
        sprintf (buffer, "%s\t%.6s\t%d\t%d\t%d\t%d", ...
                         p->utc_d, p->utc_t, p->lat, ...
                         p->lon, p->speed, p->course);
    if (fwrite(&buffer, sizeof(buffer), 1, fp) != 1)
      fprintf(stderr, "... Error on fwrite\n");
    p = p->next;
  }

  if (!ferror(fp))
    printf("\n\nSucessfully created data file!\n");
  fclose(fp);                          // close file

}
```

```
/*
output function
outputs compressed position data to .csv file
*/
void compress_out(tentry *data)
{
  char filename[80] = "\0", YN, dummy, buffer[32], rel[13];
  int i;
  FILE *fp = NULL;
  tentry *dd;

  while (1)
  {
    printf ("\nSpecify filename for output file:\t");
```

```
    scanf ("%s", filename);
    scanf ("%c", &dummy);
    strcat (filename, ".csv");          // append file ending '.csv'

    if ((fp=fopen(filename,"r"))!=NULL) // file already exiting?
    {
      printf ("\nFile already exists!\nDo you want...
                to overwrite it(Y/N)?\n\n");
      scanf ("%c", &YN);
      scanf ("%c", &dummy);
      if (YN == 'N' || YN == 'n')
        continue;
    }
    if ((fp = fopen (filename, "w")) == NULL)      // open file
    {
      printf ("\nError while accessing file!\n\n");
      continue;
    }
    break;
  }
  // write header
  sprintf (buffer, "Compressed;GPS;Data;Header     ");
  buffer[31]='\n';
  if (fwrite(&buffer, sizeof(buffer), 1, fp) != 1)
    fprintf(stderr, "... Error on fwrite\n");

  sprintf (buffer, "Date;Time;Lat_glob;Lon_glob    ");
  if (fwrite(&buffer, sizeof(buffer), 1, fp) != 1)
    fprintf(stderr, "... Error on fwrite\n");

  sprintf (rel, "%s;%.6s", Header.date, Header.time);
  if (fwrite(&rel, sizeof(rel), 1, fp) != 1)
    fprintf(stderr, "... Error on fwrite\n");
```

```
sprintf (buffer, ";%d;%d", Header.lat_glob, Header.lon_glob);
if (fwrite(&buffer, sizeof(buffer), 1, fp) != 1)
  fprintf(stderr, "... Error on fwrite\n");

sprintf (buffer, "A/R;Time;Lat;Lon;Speed;Course ");
if (fwrite(&buffer, sizeof(buffer), 1, fp) != 1)
  fprintf(stderr, "... Error on fwrite\n");

for(i=0;i<31;i++)                    // clear buffer
  buffer[i]=' ';

for(i=0;i<12;i++)
  rel[i]=' ';
rel[12] = '\n';

dd = data;
while (dd != NULL)                   // write data
{
  if ( dd->a_r )
  {
    sprintf(buffer, "A;%d;%d;%d;%d;%d", dd->time, dd->pos[0], ...
                    dd->pos[1], dd->pos[2], dd->pos[3]);

    if (fwrite(&buffer, sizeof(buffer), 1, fp) != 1)
      fprintf(stderr, "... Error on fwrite\n");
  }
  else
  {
    sprintf(rel, "R;%d;%d", dd->time, dd->pos[0]);
    if (fwrite(&rel, sizeof(rel), 1, fp) != 1)
      fprintf(stderr, "... Error on fwrite\n");
  }
```

```
    dd = dd->next;
  }
  if (!ferror(fp))
    printf("\n\nSucessfully created data file!\n");
  fclose(fp);                    // close file
}
```

D.2. Matlab Sourcecode

```
%.....................................................
%       Michael Bachner
%       University of Ulster
%       Final Year Project
%       GPS Black Box
%       read and analyse GPS
%       data from *.csv files generated
%       by compression rouine
%
%       Main Program
%.....................................................
%       Functions used:
%
%       read_in( structure with *.csv / *.xls filenames, filetype(
                                    compressed/uncompressed) )
%           --> cell array {files}{samples}{data} with extracted data
%               --> uncompressed:   data: time, lat/lon, speed,
                                    course
%               --> compressed:     abs. data: rel. time, lat/lon,
                                    speed, course
%                                   rel. data: rel. time, distance to
                                     abs. pos
```

```
%
%      decompress( cell array with compressed data )
%          --> cell array with absolute positions calculated from
                                    relative ones
%              according to navigation formulary (See section
                                    4.1.2.1 Navigation)
%              --> structured:      rel. time, dist lat/lon
%
%      variation( cell array with original data, cell array with
                                    compressed data )
%          --> cell array with data variations
 %.....................................................

% specify file name(s)
files_c = { 'gps1.csv' 'gps2.csv' 'gps_achill.csv' ...
            'gps_bundoran.csv' 'gps_dublin.csv' ...
            'gps_mayo.csv' 'gps_moher.csv'}; % 'gps_home.csv'
files   = { 'gps1.xls' 'gps2.xls' 'gps-achill.xls' ...
            'gps-bundoran.xls' 'gps-dublin.xls'
            'gps-mayo.xls' 'gps-moher.xls'}; % 'gps_home.xls'

data_c = read_in( files_c , 1);      % call read_in function

ds = size(files);                    % dynamic scaling factor
for t = 1:ds(2)                      % loop through all filenames
    data{t} = xlsread( files{t} );
end
disp('reading of files complete!')

 %.....................................................
% Start of analysis section
 %.....................................................
% analyse variations between restored and original data
```

```
data_dif = variation(data, data_c);

save data_dif data_dif;
disp('Analysis completed!')

% plot error values
for t = 1:ds(2)                    % loop through all filenames
    dat = data_dif{1, t};
    ds2 = size(dat);
    datx = dat{1};
    for s = 2:ds2(2)               % loop through all samples
        datx = [datx; dat{s}];     % write error values into vector
    end
    datx = datx';
    figure
    plot(datx(1,:), datx(2,:), 'b*');
    xlabel('Seconds')
    ylabel('Error [m]')
    m = mean(datx(2,:));
    str = sprintf('%s mean error: %.2f m', files{t}, m);
    title(str);
end
```

Listing D.1: Main script

```
%.....................................................
%       Michael Bachner
%       University of Ulster
%       Final Year Project 2006
%       GPS Black Box
%       Function stores data from an array
%       of files that hold GPS data
%.....................................................
% Description:
```

```
%       Inputs:
%           files:      array with filenames specified for analysis
%                       --> Files must exist in same directory!
%           filetype:   compressed (*.csv) or uncompressed (*.xls)
%
%       Output:
%           data:       output cell array containing all data
%
%       Processing:     set parameter according to filetype
%                       del:    delimiter (*.csv: ';'/ *.xls: '\t')
%                       header: # header lines (*.csv: 4 / *.xls: 1)
%.....................................................

function[data, stat] = read_in(files, compressed)

MAX=intmax('int16');                    % Maximum number of samples
                                        per file

if ( compressed )                       % set delimiter and # of
                                        headerlines
    del = ';';                          % depending on file
                                        structure
    header = 3;
else
    del = '\t';
    header = 1;
end

ds = size(files);                       % dynamic scaling factor

for t = 1:ds(2)                         % loop through all filenames
    c=0; abs=0; rel=0;                  % statistical information
    file = files{t};
```

```
fid=fopen( file );                    % open file
if ( fid < 0 )
    disp(file)
    disp('File failed to open!')
    disp('Exit')
    break
end

textscan(fid,'%*s',1,'headerlines',header ); % dump header lines

for n = 1:MAX                         % read data

    data{t, n-c} = textscan(fid,'%s%d%d%d%d%d%s',1,'delimiter',
                                          del);
    data{t, n-c}(7)=[];
    if ( feof(fid) )                  % exit loop when EOF reached
        break;
    end
    if ( compressed )                 % increment counters
        switch(data{t,n-c}{1}{1})
            case{'A'}
                abs = abs + 1;
            case{'R'}
                rel = rel + 1;
            otherwise
                c = c + 1;
        end
    else
        if(n == 3300)
        end
    end
end
fclose(fid);                          % close file
```

```
    disp(file)                          % report summary
    disp('succesfully analysed')
    disp('number of lines: ')
    disp(n-c)
    if( compressed )                    % statistical information
        stat = [abs rel];
        disp('Absolute Positions: ')
        disp(abs)
        disp('Relative Positions: ')
        disp(rel)
    else
        stat = [0 0];
    end
end
```

```
%.....................................
%      Michael Bachner
%      University of Ulster
%      Final Year Project 2006
%      GPS Black Box
%      Function analyses variations between
%      original data and compressed data
%.....................................
% Interface:
%      Inputs:
%          data:   cell array with original data
%              --> structured: [date] time, lat/lon, speed, course
%          data_c: cell array with restored compressed data
%              --> structured:
%                      abs. data:  rel. time, lat, lon, speed,
                                    course
%                      rel. data:  rel. time, distance, lat_r, lon_r
```

```
%                       ( *_r are restored compressed values! )
%       Output:
%           data_d: cell array with data variations
%                   --> structured:  rel. time, lat_v lon_v
%
%       Processing: 1.  Loops through all tracks & samples of data_c
%                   2.  The rel. time is used as index to obtain the
%                       corresponding original data samples
%                   3.  Restored pos. is compared to original pos.
%                   4.  Variation is stored in output array (data_d)
%.....................................................
function[data_d] = variation( data, data_c )
%.....................................................

    ds = size(data_c);                          % dynamic scaling factor

    for t = 1:ds(1)                             % looop through all
                                                tracks
        for s = 1:ds(2)                         % and samples
            sample = data_c{t, s};
            if( isempty( sample{1} ) )          % reached last sample?
                break;
            end
            if (sample{2}== 4694)
            end

            data_d{t}{s} = var(sample);     % call var to calculate
                                                abs. position
        end
    end

    %.............................
    function[dat] = var(sample)
```

```
 %.............................
% nested inside variation() (level 1)

    persistent lat1 lon1 tc          % define static variables
                                       lat1, lon1, tc
    lat2 = 0; lon2 = 0;              % nested function may
                                       access lat2, lon2
    ACC = 10^6;                      % ACC determines
                                       precision
    RAD = 6366710;                   % Earth radius according
                                       to definition of nautical
                                       Miles (nM)

    switch (sample{1}{1})
        % sample is an absolute position
        case{'A'}
            set_ref();
            dat = [sample{2} 0];     % assign output
        % sample is a relative position
        case{'R'}
            calc();                  % calulate 'restored'
                                           position
            dat = [ sample{2} get_error() ];% build output cell
                                           array
        otherwise
            disp('invalid data')
            dat = [ -1 0 ];          % default value
    end
    %.............................
    function[] = set_ref()
    %.............................
    % nested inside var() (level 2)
```

```
    lat1 = double( sample{3} )/ACC; % set lat / lon of abs
                                        position
    lon1 = double( sample{4} )/ACC;
    tc = double( sample{6} );       % set true course
end
% end of nested function set_ref (level 2)

%..............................
function[] = calc()
%..............................
% nested inside var() (level 2)
    d = double( sample{3} )/RAD;    % set distance(in rad)
                                        to last abs. pos.
    % calculate absolute positions
    lat2 = asin( cos(tc) * sin(d) * cos(lat1) + sin(lat1) *
                                        cos(d) );
    if( cos( lat2 ) == 0 )
        lon2 = lon1;                % endpoint is a pole
    else
        lon2 = mod( lon1 - asin( sin(tc) * sin(d) / cos(lat2
                                        ) ) + pi, 2*pi )
                                        - pi;
    end
end
% end of nested function calc (level 2)

%..............................
function[d] = get_error()
%..............................
% nested inside var() (level 2)
    pos = sample{2};
    index = get_pos( data{1, t}, pos );
    lat_o = ( data{1, t}(index, 3) )/ACC;
```

```
            lon_o = ( data{1, t}(index, 4) )/ACC;
            d = sqrt( ( lat2 - lat_o )^2 + ( lon2 - lon_o )^2 ) *
                                                    RAD;
        end
        % end of nested function dist (level 2)

    end
    % end of nested function var (level 1)

end
% end of function variation
```

```
%.................................................
%       Michael Bachner
%       University of Ulster
%       Final Year Project 2006
%       GPS Black Box
%       Function converts time information
%       from hhmmss to seconds
%.................................................
function[sec] = to_sec(time)
% convert from hhmmss to seconds
  h = fix(time/10000);
  ms = rem(time, 10000);
  m = fix(ms/100);
  s = rem(ms, 100);

  hm = m + (h*60);

  sec = s + (hm*60);
end
```

```
function[index] = get_pos(data, pos)
```

```
timebase = to_sec( data(1, 2) );

time = pos + timebase;

ds=size(data);
for i = 1:ds(1)
    if( time == to_sec( data(i, 2) ) )
        index = i;
        break;
    else
        index = 0;
    end
end
```

Bibliography

[1] http://www.aprs.net/vm/index.htm.

[2] http://www.kd4rdb.com/bbs/.

[3] H. McDonald, H. Park, and M. Lee, Technical report, Mississippi State University, (unpublished), http://www.ece.msstate.edu/courses/design/ece4532/2003_spring/car_alarm/.

[4] M. Chao and L. Ming, Circuit Cellar **Issue 151**, (2003).

[5] http://www.gisdevelopment.net/technology/lbs/index.htm.

[6] http://www.lbszone.com/.

[7] L. A. Clarke and J. Skobla, IEEE Aerospace Conference Proceedings (2004).

[8] G. A. McFarlane and J. Skobla, IEEE Aerospace Conference Proceedings (2004).

[9] I. Cyliax, Circuit Cellar **Issue 78**, (1999).

[10] I. Cyliax, Circuit Cellar **Issue 79**, (1999).

[11] P. Bachmann, Handbuch der Satellitennavigation.

[12] D. Görrisch, GPS im Selbstbau.

[13] http://www.gpsinformation.org/dale/nmea.htm.

[14] The Earth according to WGS 84.

[15] A. Etchells, GPS- Deriving British Ordinance Survey Grid Reference from NMEA data, http://www.developerfusion.co.uk/show/5507/.

[16] http://users.erols.com/dlwilson/gpscmpsa.htm.

[17] J. Stefan, Circuit Cellar **Issue 123**, (2000).

[18] http://users.erols.com/dlwilson/gps.htm.

[19] http://users.erols.com/dlwilson/gpsacc.htm.

[20] J. Mehaffey, A Comparison of four popular AMPLIFIED GPS ANTENNAS, http://gpsinformation.net/main/gpsantrev1.htm.

[21] Overview of Low-Cost GPS Antennas, http://home.tiscali.nl/~samsvl/anttable.htm.

[22] Test Results of some Low Cost GPS Antennas, http://home.tiscali.nl/~samsvl/anttest.htm.

[23] http://users.erols.com/dlwilson/gpscomp.htm.

[24] http://gpsinformation.net/exe/waas.html.

[25] Datasheet ATMEL ATmega128, 2005.

[26] http://www.avrfreaks.net.

[27] M. Labus, Technical report, University of Ulster (unpublished).

[28] Datasheet DELUO GPS receiver, 2003.

[29] http://www.nomad.ee/micros/etrex.shtml.

[30] E. Williams, Aviation Formulary V1.42, http://williams.best.vwh.net/avform.htm.

[31] V. K. Garg and J. E. Wilkes, Prentice Hall (1999).

[32] R. J. Landry, IEE Circuit and Systems **Series 8**, , AMI Semiconductor Inc. Dallas, Texas USA).

[33] G. A. S. Machando, Low-Power HF Microelectronics a unified approach.

[34] B. K. Mathew, The Perception Processor, `http://www.stanford.edu/~mbinu/perception/`.

[35] I. B. Berkeley Design Technology, Low Power Programmable DSP Chips: Features And System Design Strategies.

[36] H. Li *et al.*, Deterministic Clock Gating for Microprocessor Power Reduction, `http://www.ece.purdue.edu/~vijay/papers/2003/dcg.pdf`.

[37] `http://creativecommons.org`.

[38] `http://www.eej.ulst.ac.uk/~michael/`.

[39] `http://www.us.design-reuse.com/articles/article9822.html`.

[40] `http://www.us.design-reuse.com/articles/article9953.html`.

[41] `http://www.geocities.com/mapref/mapref.html`.

[42] `http://www.kh-gps.de/trackbox.ht`.

[43] `http://www.laipac.com/personal_locator_gps.htm`.

[44] `http://www.esitrack.com/products/index.html`.

[45] J. Bachiochi, Circuit Cellar **Issue 126**, (2001).

[46] J. Person, How to Write a GPS Application, `http://www.developerfusion.co.uk/show/4634/1/`.

[47] `http://www.gpsinformation.org/dale/dgps.htm`.

[48] .

[49]

[50]

[51] (unpublished).

[52] (PUBLISHER, ADDRESS, YEAR).

[53] Ph.D. thesis, .

[54] Technical report (unpublished).