A Software-Defined Radio Implementation of Maritime AIS

Kgabo Frans Mathapo

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Electronic Engineering at the University of Stellenbosch

Supervisor: Dr. G-J van Rooyen

December 2007

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work, unless otherwise stated and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature

Date

Abstract

SumbandilaSat is the second South African satellite, and is scheduled to be launched in April/May 2007. A software defined radio (SDR) automatic identification system (AIS) receiver is proposed as a possible experimental payload for this satellite. The AIS receiver can be used to track and store movement of ships at sea, and then forward this information to the ground station upon request. This thesis demonstrates the design of a SDR AIS receiver for Sumbandila satellite. The design of a GMSK/FM modem as used in AIS is presented. Models are developed and simulated in Matlab. Digital signal processing algorithms developed for the AIS receiver are highlighted. Algorithms are developed to decode and translate the AIS encapsulated binary messages. The models are transferred to C++ and the AIS receiver is implemented on the SDR architecture. Finally the real time performance of the AIS receiver is presented along with some test results and performance analysis.

Opsomming

SumbandilaSat is die tweede Suid-Afrikaanse satelliet, en sy lansering is geskeduleer vir April of Mei 2007. Hierdie tesis stel 'n sagteware-gedefinieerde radio-implementering van 'n Outomatiese Identifikasie-Stelsel (AIS) se ontvanger voor as 'n moontlike eksperimentele loonvrag vir die satelliet. So 'n AIS-ontvanger kan gebruik word om AIS-seine vanaf die grond te onderskep (om sodoende skepe se beweging te monitor en te stoor), waarna die inligting na 'n grondstasie afgelaai kan word. Die tesis demonstreer die ontwerp van 'n SDR-AIS-ontvanger, insluitende die ontwerp van 'n GMSK/FM-modem (soos gebruik in AIS). SDR-komponente word ontwikkel en in Matlab gesimuleer. Die ontwikkeling van syferseinverwerkingsalgoritmes word beklemtoon, en algoritmes word ontwikkel waarmee die binêr-geënkapsuleerde AIS-boodskappe gedekodeer kan word. Hierdie komponente word vervolgens na C++ vertaal, en die AIS-ontvanger word binne Stellenbosch Universiteit se SDR-argitektuur geïmplementeer. Ter afsluiting word die intydse werking van die ontvanger gedemonstreer, tesame met toetsresultate en 'n analise van die werkverrigting van die stelsel.

Acknowledgements

I thank my Heavenly Father for everything and for all the people He has privileged me to work with.

I express my appreciation and sincere gratitude to the following people:

- My study leader, Dr. G-J van Rooyen for his guidance excellent motivation: it's been a great pleasure working under your guidance and supervision.
- Richard Brady for his willingness to help with Linux
- SunSpace and information systems for their support and the invaluable knowledge I gained about SumbandilaSat satellite.
- Prof Jan du Plessis of SunSpace and information systems for his contribution.
- Dr Riaan Wolhuter for his contribution.
- Prof Herman Steyn for his satellite knowledge contribution.
- Marine Data Solutions for their assistance with the AIS equipment.
- Department of Communications and ISSA for their financial support.
- My wife Makgabo who sacrificed and supported me and allowed me to go on this adventure.
- My princess and prince, Phenyo and Caleb, for their love and their not understanding why the thesis is keeping me away from them at times.

Contents

List of Figures

List of Tables

Nomenclature

Acronyms

ASK	Amplitude Shift Keying
ADC	Analogue-to-Digital Converter
AGC	Automatic Gain Control
AIS	Automatic Identification System
AWGN	Additive White Gaussian Noise
BT	Bandwidth Time
BPF	Band Pass Filter
COG	Course Over Ground
CPFSK	Continuous-Phase Frequency Shift Keying
CRC	Cyclic Redundancy Check
CCITT	Consultative Comittee International Telephone and Telegraph
DC	Direct Current
DDS	Direct Digital Synthesis
DSC	Digital Selective Calling
DLS	Data Link Service
DECT	Digital European Cordless Telephone
EIRP	Effective Isotropic Radiated Power
ESL	Electronic Systems Laboratory
ETA	Expected Time of Arrival
FIR	Finite Impulse Response
FIFO	First In First Out
FCS	Frame Check Sequence
FM	Frequency Modulation
FSK	Frequency-Shift Keying
GMDSS	Global Maritime Distress and Safety System
GMSK	Gaussian Minimum Shift Keying
GMSK/FM	Frequency-Modulated, Gaussian-Filtered Minimum-Shift Keying
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
HDLC	High Level Data Link Control
HF	High Frequency

$IALA$	International Association of Marine Aids to Navigation and Lighthouse Authorities
IEC	International Electrotechnical Commission
IIR	Infinite Impulse Response
ISO	International Organization for Standardization
$ITDMA$	Incremental Time Division Multiple Access
ITU	International Telcommunication Union
IF	Intermediate Frequency
IMO	International Maritime Organisation
ISI	Intersymbol Interference
$KBPS$	Kilobits Per Second
LED	Light Emitting Diode
LEO	Low Earth Orbit
LME	Link Management Entity
LPF	Low Pass Filter
MAC	Medium Access Control
MSK	Minimum Shift Keying
MSC	Maritime Safety Committee
$MMSI$	Maritime Mobile Service Identities
F	Noise figure
$NRZI$	Non Return To Zero Inverted
$MSMI$	Multi Sensor Micro Satellite Imager
OBC	On-board Computer
PAM	Pulse Amplitude Modulation
PC	Personal Computer
PSK	Phase-Shift Keying
RF	Radio Frequency
$RATDMA$	Random Access Time Division Multiple Access
SNR	Signal To Noise Ratio
SDR	Software Defined Radio
SOG	Speed Over Ground
$SOTDMA$	Self-Organising Time Division Multiple Access
$SOLAS$	International Convention on the Safety of Life at Sea
$TDMA$	Time Division Multiple Access
UHF	Ultra High Frequency
UTC	Universal Time Coordinated
VCO	Voltage Controlled Oscillator
VHF	Very High Frequency
VDM	VHF Data-Link Message
VDL	Very High Frequency Data Link
VTS	Vehicle Tracking System

Chapter 1

Introduction

1.1 Automatic Identification System

The International Association of Marine Aids to Navigation and Lighthouse Authorities (IALA) made a proposal during the early 1990s for the carriage requirement of Automatic Identification System (AIS) by ships. The aim of the proposed system was to identify ships and their position in Vehicle Tracking System (VTS) area of coverage and in areas of restricted waters [11]. AIS operates in the very high frequency (VHF) maritime band and it allows ships to easily track, identify and exchange navigational information from one another or with the shore. This assists in collision avoidance, security and Vehicle Tracking System (VTS) reporting [11]. The information is continuously updated in near real-time, and is received by all AIS-equipped vessels and stations in the vicinity. The carriage of AIS has been made mandatory by the International Maritime Organisation (IMO) [11]. AIS uses a frequency-modulated, Gaussian-filtered minimum-shift keying (GMSK/FM) modulation scheme.

1.2 Software-defined radio and space applications

SumbandilaSat is the second satellite developed fully in South Africa. The satellite's primary mission is earth observation and it will be used inter alia for disaster management, water resource management and agricultural monitoring. Provision has been made for 2 x 1-kg experimental payloads to be included as the secondary mission of the satellite. One possible experimental payload is a software-defined radio (SDR) that would be capable of implementing an Automatic Identification System (AIS) receiver [18].

The primary purpose of the SDR AIS experimental payload is to monitor marine traffic on the South African coastline. The secondary purpose is to carry out a scientific experiment in SDR that will demonstrate the possibility of reconfiguring a radio system on a satellite through software updates, and to serve as a proof-of-concept of SDR for satellite communications systems. Software-defined radio is a technology that is currently being researched at Stellenbosch University, because of its potential to help realise reconfigurable radio sys-

tems and networks, which use the same hardware for different applications [28]. SDR allows reconfigurable radios and system upgrades can be done through software updates. Generic hardware can then be used for a variety of applications [33].

Traditional analogue radios are built from analogue components that are subject to factors such as temperature and their parameters may degrade with time. Because of this potential instability, analogue radios may need to be re-calibrated often. Digital systems on the other hand, perform signal processing consistently and are not sensitive to temperature. Their performance generally does not degrade with time. SDR is particularly suitable to space applications because communications systems on mobile platforms such as satellites can then be maintained and reconfigured using software updates, thus increasing the satellite lifetime. The satellite mission can then be changed by reconfiguring and upgrading the radio through software updates - - capability that is not possible with conventional hardware-defined radios. It is from this background that a SDR AIS receiver is designed and implemented for the Sumbandila satellite.

1.3 Scope of the thesis

SunSpace has interest in demonstrating a proof-of-concept system for passive interception of AIS messages for surveillance applications. This gave birth to the idea of an SDR AIS receiver. The receiver will be used to demodulate, decode and interpret AIS messages and then store those messages in a file or data base. The aim of this study is to design and implement an SDR AIS receiver. The following aspects are involved:

- An in-depth literature study on AIS and its functionality.
- An in-depth literature study on the Gaussian minimum shift keying (GMSK) modulation scheme.
- Design of a GMSK modulator, and simulating it in Matlab.
- Design of a GMSK demodulator, and simulating it in Matlab.
- Implementation of the GMSK modulator and demodulator in C++ on the Stellenbosch University SDR architecture.
- Design and implementation of the algorithms used to decode and interpret the AIS encapsulated messages, both in Matlab and in C++.

1.4 Thesis outline

Chapter 2 gives the literature background needed for the development of further chapters. The chapter describes the origin of AIS, the various types of AIS, the modulation scheme used in AIS, time division multiple access (TDMA), AIS messages and the process involved

in encapsulating and interpreting the AIS sentences. SDR is highlighted in this chapter. The chapter concludes by examining the Gaussian minimum shift keying (GMSK) modulation scheme and its predecessor, minimum shift keying (MSK).

Chapter 3 describes the way in which the AIS receiver will be used on the satellite. A link analysis and budget is done on the uplink and the system performance is investigated.

Chapter 4 aims to provide theoretical background on the building blocks of the AIS (GMSK modulator, filtering and demodulator) including decoding and interpretation of AIS messages. Models are developed in Matlab and the performance of the algorithms are investigated. Finally, ways to port the Matlab algorithms to C++ to finally implement the AIS are investigated. Before a system is implemented, it has to be simulated first in order to verify its performance. Chapter 5 demonstrates the simulation results of the AIS developed in Chapter 4. The Bit error probability of the demodulator is compared to the theoretical one.

Finally, the performance of the AIS receiver is investigated with actual AIS signals. Chapter 6 gives a brief overview of the Stellenbosch University SDR architecture. The way in which components are linked and how attributes are changed is described. Finally, the performance of the C++ modulator and demodulator is evaluated in terms of a test class. The test results are discussed in Chapter 7. Chapter 8 concludes the thesis and provides recommendations for further system improvement.

Chapter 2

Theoretical background

2.1 SumbandilaSat

Sumbandila is a Low Earth Orbit (LEO) satellite with multispectral imaging capability. The main payload on the satellite is a multi-sensor micro satellite imager (MSMI) which will be used to take high-resolution images of the earth [32]. Collected images are downloaded to the ground station using an S-band downlink. Low-bandwidth VHF and ultra high frequency (UHF) channels are used as telemetry, tracking and command links. The satellite is three-axis stabilised and will be launched into a sun-synchronous orbit at a nominal altitude of 500 km with a nominal local time (at the equator) of 10:00 AM. SumbandilaSat is a 80-kg satellite and the launch is scheduled for April or May 2007. The entrance pupil of the imager will normally be pointed at nadir [32]. Two South African based ground stations will be used to download data and command the satellite, one at the Stellenbosch University Electronic Systems Laboratory (ESL) and one at Hartebeeshoek near Pretoria. The remote sensing capability of the satellite will be used to monitor land, water and vegetation [32], [18].

2.2 Background on AIS

The main interest of the International Maritime Organisation (IMO) is safer shipping and cleaner oceans [11]. The proposal to make the carriage of AIS as International Convention on the Safety of life at Sea, 1974 (SOLAS) requirement was made by the International Association of Marine Aids to Navigation and Lighthouse Authorities (IALA) during the early 1990s. This was done using the Global Maritime Distress and Safety System (GMDSS) that was approved and was being implemented. The aim of proposed system was to identify ships and their position in VTS area of coverage and in areas of restricted waters. The system used the maritime VHF channel 70 which had been assigned for Digital Selective Calling (DSC) [11]. After the DSC-based system was considered, IMO received further proposals from Scandinavia Authorities to introduce a more robust transponder system. The system was to be automatic in operation, suitable for ship-to-shore and ship-to-ship operation. It would use the maritime VHF band and must cope with the density and operational intensity

of shipping in congested areas. The proposal was considered by IMO and the AIS was born. The IMO has now mandated the use of an Automatic Identification to improve safety at sea by enabling the tracking of vessels by shore based stations and other vessels (SOLAS Chapter V, regulation 19) [11]. [34] says:

> On the 1st July 2003, the Coast Guard published regulations, as required by Maritime Transportation Security Act of 2002, requiring all vessels subject to SOLAS convention and certain domestic vessels operating in VTS areas to install a ship-board AIS.

The performance standards for AIS was prepared and named as a Recommendation on the performance standards for ship-borne AIS. These standards were approved by the IMO Maritime Safety Committee (MSC) at its sixty-ninth session (May 1998) under resolution MSC.74 (69). The IMO requested in Geneva during October 1997 that two maritime VHF channels be assigned for AIS. The channels allocated were: AIS 1(161.975 MHz) and AIS 2(162.025 MHz) and it was stated that these frequencies will be used for an AIS and surveillance system for providing worldwide operation on high seas, unless other frequencies are designated on a regional basis for this purpose [11].

2.2.1 Types of shipborne AIS mobile stations

There are two types of shipborne AIS mobile stations namely; Class A mobile stations which must be fully compliant with the IMO performance standard and the IEC 61993-2 standard, Class B stations which have a different functionality on the very high frequency data link (VDL) message level and a different reporting rate [10]. For example, long-range and DSC polling are compulsory on Class A stations whereas they are optional on Class B stations.

2.2.2 Layer model of an AIS station

Table 2.1 illustrates the layer model of an AIS station. The physical layer is responsible for transmitting bits from the transmitter, out on to the data link [10]. where R_x is the receiver and T_x is the transmitter.

The thesis focuses on the design and implementation of the physical layer of the AIS. The link layer is responsible for data link activation, data packaging and error detection and control.

2.3 AIS functional overview

AIS is a shipboard broadcast system that continuously and autonomously broadcasts a ships identification, position, call sign, Maritime Mobile Service Identities (MMSI), its Global Positioning System (GPS) position, speed, course, heading and rate-of-turn, its destination, cargo and other maritime navigational messages [11]. AIS operates in the VHF maritime

Table 2.1: *Illustration of the layer model of an AIS (adapted from [13]).*

	Application layer	
	Presentation layer	
	Session layer	
	Transport layer	
CHANNEL 1	Network layer	CHANNEL 2
Link management entity (LME) layer		Link layer LME
Data link service (DLS) layer		Link layer DLS
Medium access control (MAC) layer		Link layer MAC
Physical layer		Physical layer
R_x A	T_x A/B	R_x B

band and it allows ships to easily track, identify and exchange navigation information between one another or the shore. This information is continuously updated near real-time, and is received by all AIS equipped vessels and stations in its vicinity. Each AIS system consists of one VHF transmitter and two VHF Time Division Multiple Access (TDMA) receivers [11].

Position and timing information is usually obtained from an integral or external global navigation satellite system (GNSS) such as a GPS and a differential GNSS receiver for precise position. Each AIS system is capable of handling over 4500 reports per minute and updates as often as every two seconds. It uses a Self-Organising Time Division Multiple Access (SOTDMA) scheme to meet this high broadcast rate and to ensure reliable ship-to-ship operation [11]. AIS uses 9.6 kbps FM/GMSK modulation over 25 or 12.5 kHz channels for transmissions. AIS uses High Level Data Link Control (HDLC) packet protocols. The AIS stations are divided into mobile AIS and Fixed AIS stations. In this study we focus on the development of a mobile AIS receiver. Figure 2.1 shows a block diagram of an AIS mobile station. AIS uses 161.975 MHz and 162.025 MHz as carrier frequencies. The AIS receiver should be capable of operating on 25 kHz or 12.5 kHz channels. Bandwidth-adapted frequency modulated Gaussian-filtered minimum shift keying (GMSK/FM) is used in the AIS physical layer. A non-return-to-zero inverted (NRZI) waveform is used for data encoding. The NRZI encoded data are then GMSK encoded before frequency modulating the carrier. AIS uses a GMSK Bandwidth time (BT) product of between 0.3 and 0.5 [9].

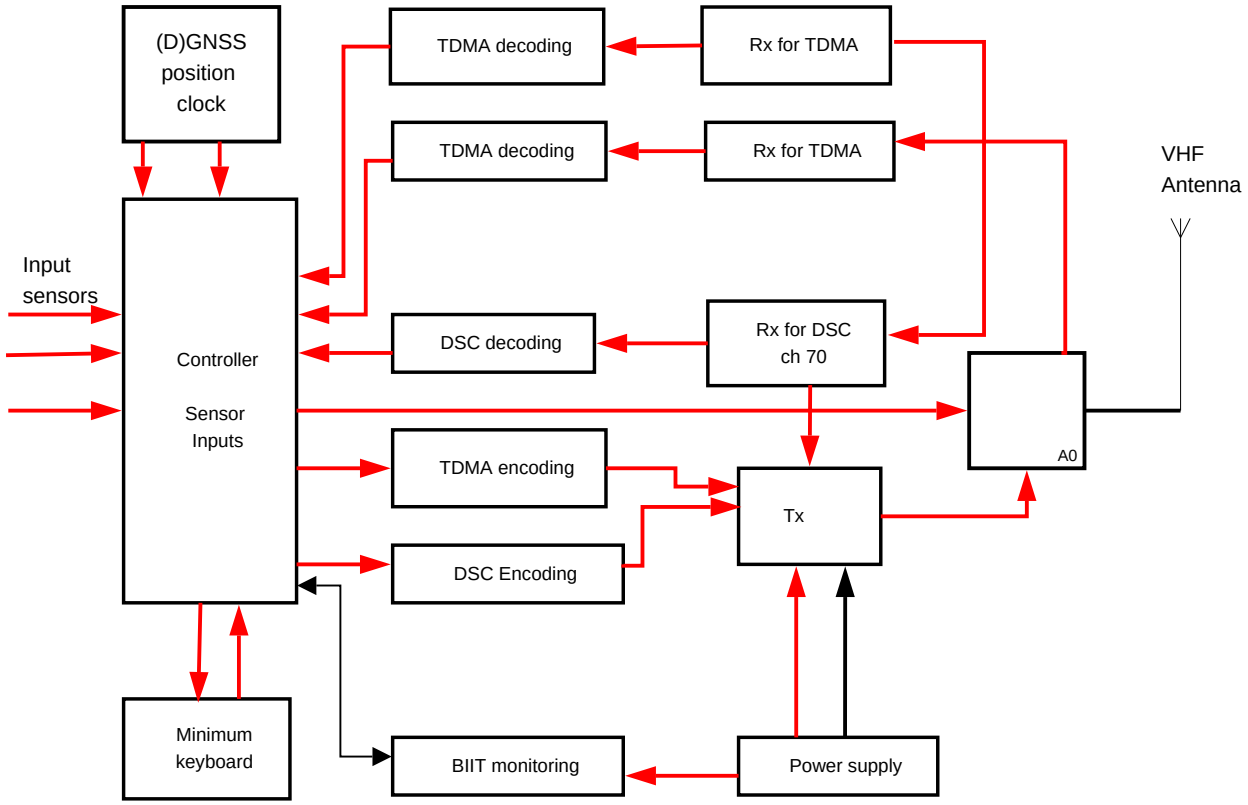


Figure 2.1: *Overview of a ship-borne AIS mobile station.*

2.4 Channel allocation

Very High Frequency (VHF) is the radio frequency range between 30 MHz (wavelength 10m) and 300 MHz (wavelength 1 m). These frequencies are commonly used for FM radio broadcasting at $88 - 108$ MHz, television broadcasting, terrestrial navigation systems and aircraft communications. VHF signals have a range that is further than the line of sight from the transmitter [14], [29], [6].

VHF signals are not usually reflected by the ionosphere, therefore transmissions do not interfere with transmissions from thousands of kilometers away [14]. Frequencies at the lower end of the VHF band are capable of overcoming the shielding effects of hills and structures to some degree [29]. While VHF signals are more easily blocked by land features such as hills or large structures than High Frequency (HF) signals, they are less easily blocked by buildings and other less substantial objects than HF. VHF can propagate further than their normal range through tropospheric enhancement e.g. travelling along inside a tropospheric duct [16], [29].

2.4.1 Very high frequency data link (VDL) capacity

AIS uses maritime frequencies 156-162.5 MHz and it uses both 25 kHz and 12.5 kHz simplex channel bandwidths. Although only one radio channel is necessary, each AIS station

transmits and receives over two radio frequencies to avoid interference problems [11]. When AIS is operating in half-duplex mode, the resulting capacity is 2250 slots per minute at a transmission rate of 9.6 kilobits per second (kbps). When both of the AIS channels (AIS 1 and AIS 2) are used, the reporting capacity is doubled to 4500 slots per minute or 4500 reports per minute because each position report fits into one slot [11].

When the number of AIS stations within the line-of-sight range of a receiving station is more than the frame capacity in terms of reports per minute, the SOTDMA algorithm and the GMSK/FM modulation effectively reduces the radio cell range or size for each AIS station. This is done by prioritising AIS stations closer to the receiving stations and dropping or suppressing transmissions from far away. The result is that, as a channel approaches an overloaded state, the SOTDMA algorithm produces a continuous reduction in the radio cell [11].

2.5 Time division multiple access (TDMA)

Time division multiplexing is a method whereby each channel is dedicated during a specific time slot. TDMA systems work by assigning multiple users or stations to one frequency channel. One carrier frequency is assigned to the channel and digital signals sent out from different stations are transmitted on that frequency channel in specified time slots in a repetitive frame structure [27]. Each station is allocated one or more of these time slots per frame and keeps them until a call is completed. There must be provision to distinguish the beginning and the end of the frame [27]. Figure 2.2 shows an example of TDMA repetitive frame structure with N slots per frame.

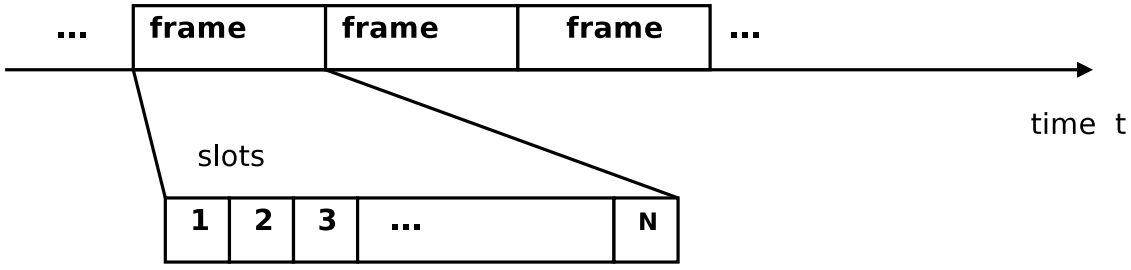


Figure 2.2: *TDMA slot structure (adapted from [27]).*

2.5.1 Self organising time division multiple Access (SOTDMA)

SOTDMA is handled by the medium access control (MAC) link sub-layer. The MAC provides a method for granting access to the data transfer medium. Self Organising Time Division Multiple Access (SOTDMA) is a special form of TDMA. In SOTDMA each AIS station announces in advance in which time slots it will make its future transmissions [11], [9]. Because future transmissions are known by every station that heard the broadcast - the

other stations are able to remove those transmission possibilities and thus allocate their transmissions to unused slots. Thus, as [35] says:

> Each station determines its own transmission schedule (slot allocation) based on the data link traffic history and knowledge of future actions by other stations. AIS stations continuously synchronise themselves to each other to avoid overlap of slot transmissions. Slot selection by an AIS station is randomised within a defined interval and tagged with a random timeout of between 0 and 8 frames. When a station changes its slot assignment, it pre-announces both the new location and the timeout for that location. In this way new stations, including those stations which suddenly come within radio range close to other vessels will always be received by those vessels.

To be able to keep a track record of future transmissions, each AIS station contains a "slot map". The slot map contains at least one minute which is equal to one frame. Each time slot is 26.67 ms long and contains 256 bits [35]. Figure 2.3 shows a graphical explanation of principles of SOTDMA. Each station is required to transmit several times in each frame. If

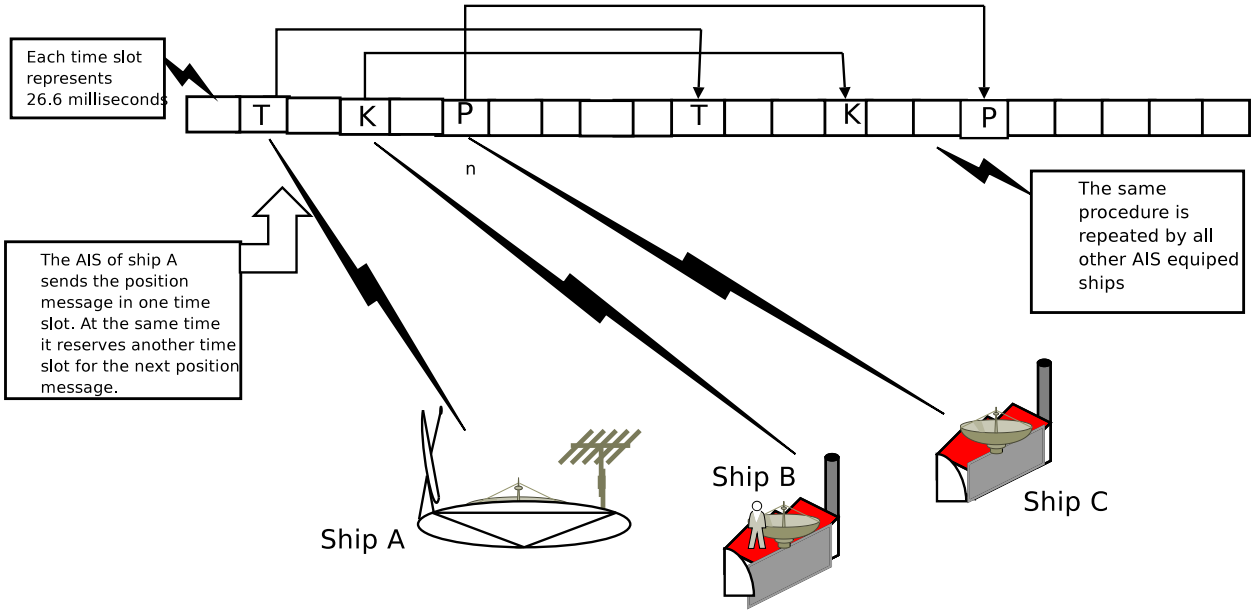


Figure 2.3: *Principles of SOTDMA (adapted from [35]).*

the system gets overloaded by the number of transmissions required, the SOTDMA method ensures continuous reduction in the radio cell. This is done by stealing slots from transponders that are far away and allowing transmissions from nearby stations. This ensures that each station will always hear vessels that are closer to it, which are potential threats in terms of collision [11].

2.6 AIS data transfer

In AIS, the two VHF frequencies AIS 1 and AIS 2 are used in parallel. The modulation used in AIS is FM/GMSK (Frequency Modulation/Gaussian Minimum Shift Keying) because of its robustness, discrimination possibilities, bandwidth efficiency and its application in mobile digital communications [9]. The output power of an AIS station is between 2 and 12.5 W.

An AIS station can operate by using one channel for transmitting and the other for receiving, or it can communicate on both channel frequencies at the same time in half-duplex mode. Each minute of time on each channel is divided into 2250 time slots. Those 2250 time slots make up a frame and each frame is repeated every minute. These frames are synchronised using GNSS time information as the first timing mechanism. Access by any AIS station to the data link is given at the start of a slot [9]. The beginning and end of a frame coincide with the Universal Time Co-ordinated (UTC) minute, when UTC is available. If UTC is not available, the AIS station uses messages from other stations or base stations to re-synchronise itself [9]. An AIS position report message fits into one time slot. When using the 12.5 kHz channels the communication range is reduced [11]. The size of the radio cell when using the 12.5 kHz channel in overload condition is reduced to approximately half the size compared to that in the 25 kHz channel [11].

2.6.1 Required update rates

The AIS must be able to handle a minimum of 2000 reports per minute. The reporting intervals of each AIS station depend on the speed of the vessel and whether it is changing course or not. Class A ship-borne mobile station reporting rates are shown in Table 2.2. In

Table 2.2: *Update intervals for a Class A Mobile AIS station (adapted from [11]).*

Ship's dynamic conditions	**Nominal reporting interval**
Ship at anchor and not moving faster than 3 knots	3 minutes
Ship at anchor or moored and moving faster than 3 knots	10 seconds
Ship 0-14 knots	10 seconds
Ship 0-14 knots and changing course	3 minutes
Ship 14-23 knots	6 seconds
Ship 14-23 knots and changing course	2 seconds
Ship > 23 knots	2 seconds
Ship > 23 knots and changing course	2 seconds

order to predict the turning rate and track when ships are changing their course, a reporting rate that is three times higher than the normal rate has been chosen based on the required

position accuracy [11].

2.6.2 AIS message types and formats

AIS messages can be either binary messages that are addressed to a particular mobile or shore station, or they can be binary messages that are broadcast to all stations. The different types of AIS information messages are classified as static, dynamic or voyage related. These messages require different update rates. Static information is entered into the AIS on installation and need only be changed if the ship changes its name or is converted to a different type of ship. Dynamic information is automatically updated from the ship sensors connected to the AIS. Voyage-related information is entered manually and updated during the voyage [11].

Static information include the MMSI, the call sign and name, the IMO number, length and beam, type of ship, location of position fixing antenna and height over keel. This information is reported every 6 minutes, when data has been amended, and on request. Dynamic information include ships position with accuracy, position time stamp in UTC, course over ground (COG), speed over ground (SOG), heading and rate of turn. Voyage-related information include ships draught, hazardous cargo (type), destination and expected time of arrival (ETA), route plan and number of persons on board [10]. This information is reported every 6 min, when data has been amended, and on request. There are also short safety related messages which are sent as required and are independent of timing. They are manually entered and addressed or broadcasted [11].

2.6.3 Standard message formats

The information transmitted by AIS is grouped into a series of standard formatted messages. The AIS Technical standard ITU-R M.1371-1 lists 22 different types of these messages. Table 2.3 shows some of the message grouping and the modes of operation associated with each message. An example of how the AIS messages are structured is demonstrated by

Table 2.3: *Primary Message groups and operational mode (AU = Autonomous, AS = Assigned, IN = polling)(adapted from [11]).*

Message Identifier	Description	Mode of operation
1, 2, 3	Position report	AU, AS
5	Static and voyage related Class A	AU, AS
6, 7, 8	Binary Messages-Addressed, acknowledged or broadcast.	AU, AS, IN
12, 13, 14	Safety related	AS, IN
17	DGNSS Broadcast Binary	AS

studying the Position Report, which is message group 1, 2 or 3. Appendix A, Table A.1

shows how the position report message is structured. The structure for static and voyage related data message five "5" is also shown in Appendix A. The message ID identifies the message. The user ID should be the MMSI. The MMSI is 30 bits long and the first 9 digits (most significant bits) should be used only [9]. Parameter fields containing sub-fields (e.g. communication state) are defined in separate tables.

AIS messages can use either SOTDMA, random access time division multiple access (RATDMA) or incremental time division multiple access (ITDMA) communication state. Message one "1" which is a position report uses the SOTDMA communication state [9]. The communication state gives the following information:

1. It contains information used by the slot allocation algorithm in the SOTDMA concept.
2. It also indicates the synchronisation state

The SOTDMA communication state is structured as shown in appendix B, Table B.1. Other communication states are discussed in [9].

2.6.4 VHF Data-Link Message (VDM) encapsulation

The AIS message consists of a preamble or Training sequence (24 bits), Start flag (8 bits), Message data (maximum of 168 bits for a single slot message), Frame check sequence (FCS) (16 bits) and end flag (8 bits). The messages transmitted by AIS are encapsulated binary data which need to be translated into meaningful information after reception at the receiver. The sentences used to provide information assembled for broadcast by the AIS uses the six-bit binary field [13]. In order to encapsulate, the number of binary bits must be a multiple of six. If it is not, one to five *fill bits* are added. The IEC 61162-1 standard supports the transport of encapsulated binary coded data [13].

The structure of the sentence provides for the transfer of long binary messages by using multiple sentences. This is done because the maximum number of message data bits that can be contained in an AIS radio message is 1008 bits [13]. Thus the 1008 bits requires 168 6-bit symbols. According to [13], this number of characters is too many for a single sentence. The AIS standard sentence contains up to 82 characters. When coding and decoding the encapsulation string, the string may require more than one sentence to transfer. When this is the case, there are characters on the encapsulated sentence which specifies the number of sentences used and the order of the sentence in the message. An example of the encapsulated AIS sentence is shown below [13]:

```
!AIVDM,1,1,,A,1P000Oh1IT1svTP2r:43grwb0Eq4,0*01<CR><LF>
```

The first character "!" in the encapsulation sentence is the start of a sentence. It is then followed by alphanumeric characters identifying the type of talker. In this case the talker is an AIS and the received message is a VHF Data-link message. The first one (1) represent the total number of sentences used in transferring the message. Encapsulated information

sometimes require more than one sentence [12]. The second one "1" represent the sentence number field. This value identifies which sentence of the total sentences in a message. The value between the two commas represent the sequential message identifier. This field distinguishes one encapsulated message that required more than one sentence to transfer the message, from another message that required the same number of sentences. This field is incremented every time there is a new message that requires the same formatter as the previous message [12].

The *A* indicate the AIS channel on which the message was received. It is then followed by the encapsulated radio message. The zero after the coma represent the number of fill bits. This is the number of fill bits that were added to complete the last six bit coded character on the message. This field is then followed by the Checksum delimiter. It shows that the next field represent the hexadecimal value of the Checksum. It is followed by the checksum field. The Checksum is the absolute value obtained by exclusive-OR'ing the eight data bits of each character in the sentence between but excluding ″!″ and ″∗″ [12]. The last characters terminates the VDM sentence.

2.7 Software-Defined Radio (SDR)

According to [33]

> An SDR is defined as radio in which the receive digitisation is performed at some stage downstream from the antenna, typically after wideband filtering, low noise amplification, and down conversion to a lower frequency in subsequent stages with a reverse process occurring for the transmit digitisation.

A SDR is a radio that performs signal processing using software techniques. The radio still incorporate analog components like antennas, wideband band pass filters (BPF), low noise amplifiers and mixers for down converting the higher radio frequencies to a lower intermediate frequency (IF) in the heterodyne receivers or to baseband in the direct conversion receivers. Figure 2.4 shows a simplified functional block diagram of the SDR. In SDR, digitisation is

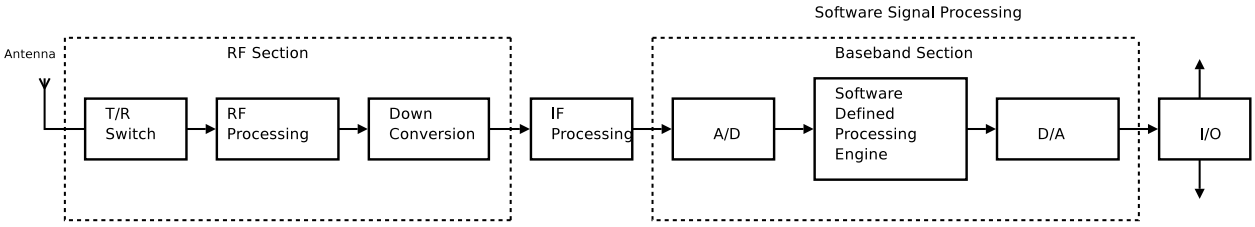


Figure 2.4: *The functional block diagram of the software defined radio (SDR) (reproduced from [13]).*

performed after down conversion process. A SDR is characterised by digital signal processing accomplished in flexible and reconfigurable functional blocks [33].

The aim in SDR is to be able to transmit and receive signals of any frequency, power level, bandwidth and modulation scheme [33]. The goal is to move towards multiband and multimode radios.

2.8 Modulation

Modulation is a process by which information is encoded onto a signal [20]. During the modulation process, a part of the carrier signal is varied in accordance with the message signal. The message can be carried in the amplitude, frequency or phase of the carrier signal. There are two categories in which a modulation scheme can fall into. In Analog modulation, the information is carried in a continuous waveform and in digital modulation; the message is contained in a set of discrete values. In digital modulation, the result of modulating the carrier is referred to as amplitude shift keying (ASK), phase-shift keying (PSK) or frequency-shift keying (FSK) if it is the amplitude, phase or frequency that is varied in accordance with the baseband signal respectively [39]. Demodulation is a process by which the message signal is recovered from the modulated carrier signal.

2.8.1 GMSK modulation

Minimum shift keying (MSK) is a form of continuous-phase frequency shift keying (CPFSK) binary modulation where the modulated carrier has no phase discontinuities, and frequency changes occur at the carrier zero crossings. In MSK modulation, the frequency spacing between the two frequencies transmitted during the FSK is $1/2Tb$, where Tb is the bit interval [8], [31]. This minimum frequency spacing allows the two FSK signal waveforms to be orthogonal to each other and to be correctly detected using a coherent detector. The FSK signal subsequently has phase continuity. MSK is a form of FSK modulation with modulation index h = 0.5 (where h is the modulation index defined as Δf per bit rate, and Δf is the tone spacing) which results in minimum frequency separation for orthogonal signalling over symbol interval. The MSK signal can be written as:

$$s_i(t) = \cos(\omega_0 t + \Delta\omega t + \Theta_0) \tag{2.1}$$

where $\Delta\omega = \frac{2\pi}{4T_b}$, $\Delta\omega$ is the frequency deviation and Θ_0 is the initial phase at time t = 0..

MSK is popular in wireless communications because of its desirable characteristics, namely, high immunity to noise and interference and ease of implementation [20]. A disadvantage with MSK is that the spectrum is not compact enough for narrowband applications [23]. The plot of the spectrum reveals sidelobes extending well above data rates (power spectrum density does not fall fast enough to completely reduce co-channel interference) as shown in Figure 2.5. The power spectrum of MSK can be made more compact by using a pre-modulation low-pass filter at the cost of intersymbol interference.

Another alternative method for generating MSK modulation is to directly inject the NRZ data into the frequency modulator with the modulation index set at 0.5.

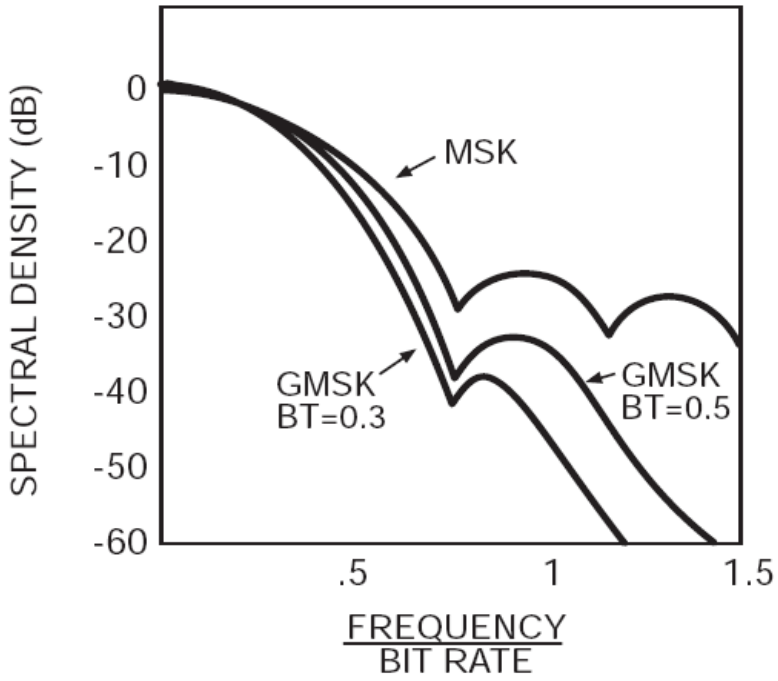


Figure 2.5: *Power spectrum density of the MSK versus GMSK (reproduced from [17]).*

Gaussian minimum shift keying (GMSK) is MSK modulation with a Gaussian premodulation filter [27]. The Fourier Transform of the Gaussian shaping function is also Gaussian shaped. The advantage of the Gaussian filter is that it has very little overshoot or ringing in its impulse response and the magnitude response does not change abruptly at the cutoff frequency [1]. From Fourier analysis, the width of a pulse and its bandwidth are inversely proportional to each other [27]. The information bit input or pulse train is passed through the Gaussian low-pass filter (LPF), and the output of the filter is frequency modulated. Figure 2.6 shows a block diagram of GMSK modulator. This results in GMSK achieving smooth

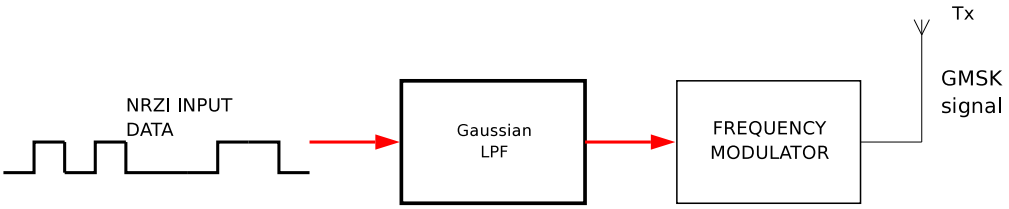


Figure 2.6: *Block diagram of a GMSK modulator implemented using a Gaussian premodulation filter and FM.*

phase transitions between signal states, thereby reducing the bandwidth requirements. In most cases, the Gaussian pulse is made longer than one bit interval, which results in pulses overlapping, giving rise to intersymbol interference (ISI). The extent of the intersymbol interference is controlled by the product of the Gaussian filter bandwidth and bit duration

[20]. The smooth phase transitions of GMSK signals make demodulation at the receiver more complex: For the receiver to achieve a specified bit error rate, more power must be generated at the transmitter in order to overcome the receiver noise in the presence of ISI [4]. GMSK requires more bandwidth than ordinary MSK at the receiver to effectively recover the carrier.

GMSK can be considered to be a combination of frequency and phase modulation, where the phase of the carrier is increased or decreased by $\pi/2$ over a bit period, depending on the data pattern. The rate of phase change is limited by the Gaussian filter response. This assists in improving the bandwidth efficiency [4].

2.8.2 GMSK as frequency modulation

Because a GMSK modulator can be considered to be a simple Gaussian filter in cascaded with a voltage controlled oscillator (VCO), this is equivalent to frequency modulating the carrier frequency with the Gaussian lowpass filtered data. We now consider angle modulation.

An angle modulated signal can be represented as [23]:

$$s_t = A\cos[\omega_c t + \phi(t)] = \mathcal{Re}(Ae^{j[\omega_c t+\phi(t)]}) \tag{2.2}$$

The instantaneous phase is represented as:

$$\theta_i(t) = \omega_c t + \phi(t) \tag{2.3}$$

Frequency modulation (FM) is the form of angle modulation in which the instantaneous frequency $\omega_i(t)$ of the carrier is varied linearly with the message signal m(t). For FM signals, the frequency deviation of the carrier is proportional to the message signal [23]. That is:

$$\frac{d\phi}{dt} = k_f m(t) \tag{2.4}$$

$$\phi(t) = k_f \int_{t_0}^{t} m(\lambda)d\lambda + \phi(t_0) \tag{2.5}$$

where k_f is the frequency deviation constant (radians/sec/volt). Thus the FM waveform equation can be expressed as:

$$s(t) = A\cos\left[\omega_c t + 2\pi k_f \int_0^t m(t)dt\right] \tag{2.6}$$

Chapter 3

System-level design

3.1 Link analysis

The link analysis of any system involves the calculation and tabulation of the available signal power and the interfering noise power at the receiver [30]. The link budget of the system gives us more information about the overall system. From the link margin, one can learn whether it will be possible to have reliable communication between the transmitter and the receiver. For satellite links, the propagating medium or channel is mostly occupied by empty space [30].

The signal to noise ratio (SNR) is used to measure the performance of the system at various points in the link. The SNR is degraded either through decrease in signal power or increase in noise power. Sources of noise and factors that cause signal degradation have been studied and can be found in textbooks such as [30], [19]. The path loss between any pair of antennas is ratio between the transmitted power and the received power. The power received at the receiving antenna is given by:

$$P_r = \frac{P_t G_t A_{er}}{4\pi d^2} \tag{3.1}$$

where P_t is the transmitter power, A_{er} is the effective area of the receiving antenna, d is the distance between the two antennas and G_t is the transmitting antenna gain. G_t is defined as the ratio of the power radiated to the centre of the coverage area to the power radiated by an isotropic (radiating equally in all directions) antenna [30].

The signal power will be a function of frequency. When we include all the antenna gains and the path loss (loss suffered by the signal due to the distance between the transmitter and the receiver), the received power is expressed as [30]:

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi d)^2} \tag{3.2}$$

The effective isotropic radiated power (EIRP) is given by

$$EIRP = \frac{P_t G_t}{L_t} \tag{3.3}$$

where L_t is the loss incurred by the signal between the transmitter and the antenna (antenna line loss). The carrier-to-noise power spectral density is given by

$$\frac{P_r}{N_o} = \frac{EIRP\frac{G_r}{T^o}}{\kappa L_s L_o} \tag{3.4}$$

where N_o is the noise power spectral density, T^o is the system effective temperature, L_s is the path loss, L_o represents all other losses and κ is Boltzmann's constant $= 1.38 \times 10^{-23}$ joule/kelvin. Assuming all the received power to be in the modulated signal, the above equation can be written as

$$\frac{P_r}{N_o} = \frac{S}{N_o} = \frac{E_b}{N_o}\, R \tag{3.5}$$

where S is the average modulating signal power, E_b/N_o the bit energy per noise power spectral density, and R the bit rate. The safety margin or link margin is given as the difference between the actually received $(E_b/N_o)_r$ and the required $(E_b/N_o)_{reqd}$ in dB [30]. The link margin M parameter is given by

$$M[\text{dB}] = \left(\frac{E_b}{N_o}\right)_r [\text{dB}] - \left(\frac{E_b}{N_o}\right)_{reqd} [\text{dB}] \tag{3.6}$$

The link budget is calculated using the formula in [30] in decibels:

$$\begin{aligned} M &= EIRP\ [\text{dBW}] + G_r[\text{dBi}] - \left(\frac{E_b}{N_o}\right)_{reqd} [\text{dB}] - \\ &\quad R[\text{dB-bit/s}] - \kappa T^o[\text{dBW/Hz}] - L_s[\text{dB}] - L_o[\text{dB}] \end{aligned} \tag{3.7}$$

3.1.1 Link budget

A link budget for the uplink was computed. The AIS Antenna height used is 20m. The transmission power of the AIS station is 12.5 W maximum. A 600 km orbit is used to compute the slant range of 2328 km with 5 degrees minimum ground station elevation. The satellite receiver noise temperature T^o is 864K, satellite receiver noise figure F is 6 dB and the satellite receiver antenna gain varies according to the orientation of the satellite between 2 dBi and -18 dBi [32]. (Simulation show it to be better than -7dBi for more than 95% of satellite orientations [32]). The noise spectral density N_o is calculated from [19]:

$$N_o = \kappa T^o \tag{3.8}$$

In order to demodulate a digital bit reliably, the energy-per-bit E_b must exceed the noise spectral density, N_o, by a specified amount [19]. The required bit error rate (the probability that a data bit is incorrectly received) using the GMSK modulation scheme was determined in Appendix E to be 10^{-4}. The $\left(\frac{E_b}{N_o}\right)_{reqd}$ needed to achieve this required bit error rate was read from the graph of the bit error probability for GMSK in [23, page 330] to be 16 dB.

The received $\left(\frac{E_b}{N_o}\right)_r$ was calculated using [19]:

$$\left(\frac{E_b}{N_o}\right)_r = EIRP + L_s + L_o + G_r + 228.6 - 10\log T^o - 10\log R \tag{3.9}$$

where $10 \log \kappa = -228.60\ dBW/(Hz \cdot K)$. The $\left(\frac{E_b}{N_o}\right)_r$ was found to be 23.48 dB. The carrier-to-noise density ratio was calculated using Equation 3.5 (where R is the data rate and is equal to 9600 bps) to be equal to $23.48 + 39.82 = 63.30\ dB$. Table 3.1 gives the link budget. The link budget results shows a link margin 7.48 dB, which is more than sufficient

Table 3.1: *Link budget, uplink performance*

Item	**Symbol**	**Value**	**Unit**
Frequency	f	161.975	MHz
Transmitter power	P_t	10.97	dBW
Bit rate	R	9600	bps
Transmit Antenna Gain	G_t	5	dBi
Minimum elevation	ϵ	5	deg
Transmitter line loss	L_t	1	dB
Effective isotropic radiated power	EIRP	14.97	dBW
Propagation path length	S	2328	km
Space path loss	L_s	143.9	dB
Satellite Rx Antenna Gain	G_r	-7	dBi
Receiver noise figure	F	6	dB
Receiver noise temperature	T_r	864	K
Receiver IF bandwidth	B	25	kHz
Noise spectral density	N_o	-199.2	dBW/Hz
Carrier-to-noise density ratio	C/N_o	63.30	dB-Hz
Required $\frac{E_b}{N_o}$	Reg $\frac{E_b}{N_o}$	16	dB
Other Losses	L_o	1	dB
Received signal power	P_r	-135.94	dBW
Received $\frac{E_b}{N_o}$	$\frac{E_b}{N_o}$	23.48	dB
Link margin	M	7.48	dB

to receive the AIS signals on the satellite from ships at sea.

Figure 3.1 shows how the satellite will be used to monitor marine traffic. The satellite is equipped with an SDR AIS receiver for ship surveillance purposes. Each ship is fitted with an AIS transponder, and continuously broadcasts AIS information.

3.2 AIS system

AIS uses a bit-oriented protocol for data transfer, which is based on the high-level data link control (HDLC) standard, as specified by ISO/IEC 3309: 1993 "Definition of packet structure" [9]. Data is transmitted using a transmission packet as shown in Figure 3.2.

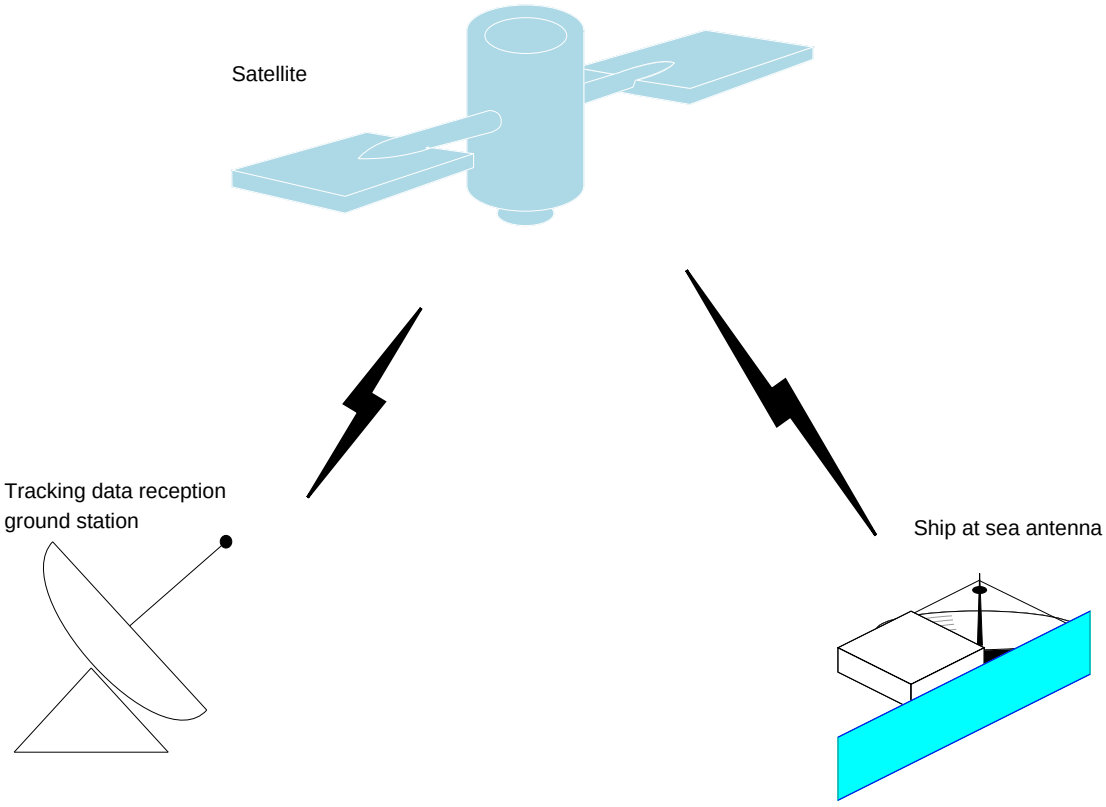


Figure 3.1: *Diagram of the satellite monitoring marine traffic and relaying it to a ground station.*

Packets are sent from left to right in the diagram. Data transmission begins with a 24-bit

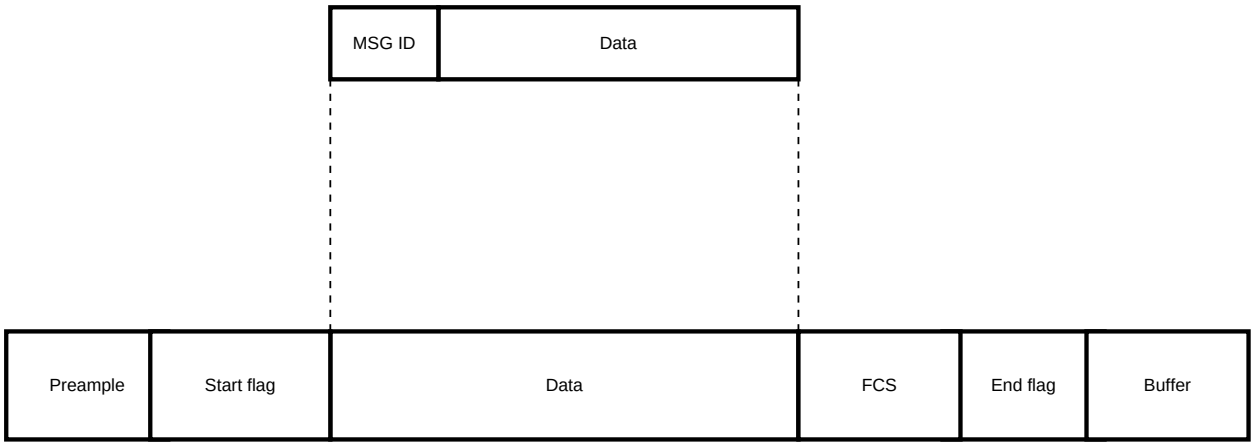


Figure 3.2: *Structure of the AIS data transmission packet.(adapted from [9]).*

training sequence (the preamble) consisting of one synchronisation segment. This segment consists of alternating ones and zeros (0101 ...) and the sequence can either start with a "1" or a "0" since NRZI encoding is used [9]. The NRZI encoder gives a change in the signal level when it encounters a zero "0" in the bit stream and no change when a one "1" is encountered. The training sequence is used to synchronise the receiver. The start flag is 8 bits long and consists of a standard HDLC flag (01111110), marking the start of each transmission packet. The data portion is 168 bits long in the default transmission packet.

Bit stuffing is applied on the transmitter side. This means that if five consecutive ones are encountered on the output bit stream to be transmitted, a zero should be inserted. The main reason for bit-stuffing is to prevent the transmitter from transmitting a data sequence that is identical to the start or stop flag. Bit stuffing applies to all bits except the start flag, stop flag and the training sequence [9]. When data is transmitted on the AIS VHF link, the message is grouped into bytes first and the least significant bits are output first. During the output process, data should be subject to bit-stuffing and NRZI coding [9]. Unused bits in the last byte should be set to zero in order to preserve the boundary of the bytes.

The frame check sequence (FCS) uses a 16-bit cyclic redundancy check (CRC) polynomial to calculate the checksum as defined in ISO/IEC 3309: 1993 [9]. The CRC bits are preset to "1" at the beginning of the CRC calculation. Only the data portion is included in the checksum calculation. The end flag is 8 bits long and identical to the start flag. The buffer is normally 24 bits long and is used to compensate for bit stuffing, distance delays, repeater delay and synchronisation jitter [9]. The total length of the normal packet is 256 bits, which is equivalent to one TDMA time slot. The satellite is to receive the AIS signals on the VHF uplink. The AIS information can then be stored and forwarded to the ground station using either the S-band channel or the UHF downlink provided on the satellite (the S-band channel allows high-speed data transmission). Both downlink channels were developed separately from the project described in this paper, and are not considered further here.

3.3 SDR subsystems on the satellite

Figure 3.3 shows an overview of the satellite subsystems used to receive the AIS signals. The

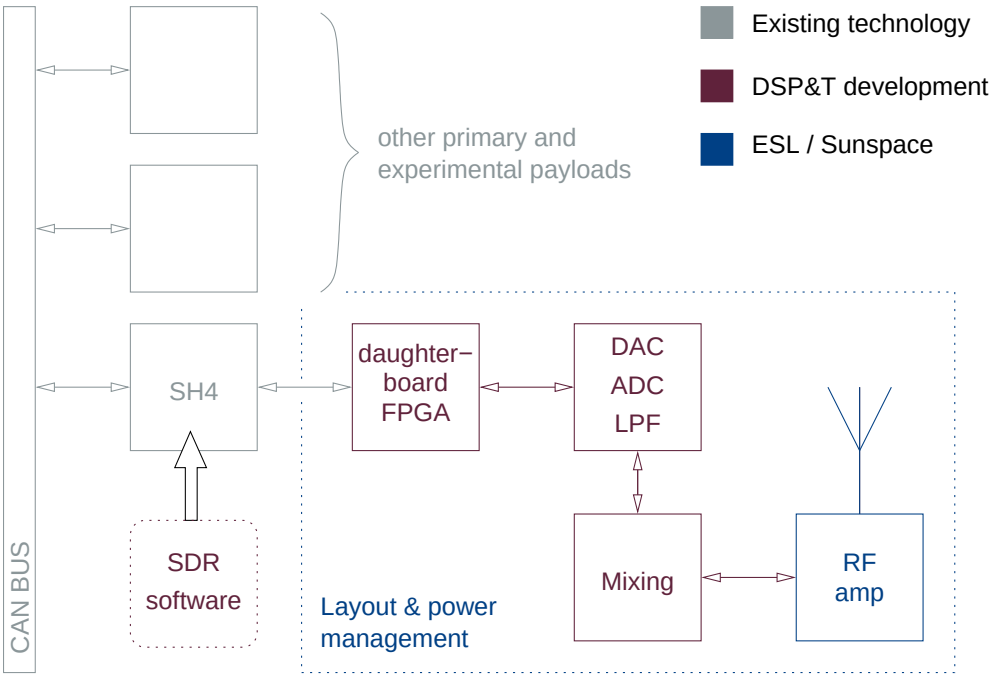


Figure 3.3: *Block diagram of the SDR subsystem on SumbandilaSat (reproduced from [37]).*

on-board computer (OBC) used for satellite control is a Sun Space and Information Systems design, and it contains a Hitachi SH4 as primary CPU. The RF front end consists of the RF amplifier, a mixer for downconversion of the RF signal to the required intermediate frequency (IF), a bandpass filter or lowpass filter to remove the other mixer products and allow only the IF to pass through, and an analogue-to-digital converter (ADC) to digitise the analogue IF signal. The digitised IF signal is then routed to the SDR processor. Demodulation, filtering, synchronisation and signal recovery are all performed in software by the digital processor.

Chapter 4

Detailed design

4.1 Introduction

The aim of Chapter 4 is to present the background theory for each component of the AIS receiver. Each building block of the AIS receiver was modeled and implemented in Matlab first in order to verify the working of the system before it was implemented in C++. But in order to develop the proposed AIS receiver, it was necessary to first develop the modulator for the purpose of providing the receiver with the reference modulated signal. The GMSK modulator was first developed in Matlab to provide the demodulator with the GMSK modulated signal before it was implemented in C++. The models which are used to build the modulator and the AIS receiver are shown in Figure 4.1. The models developed in Matlab

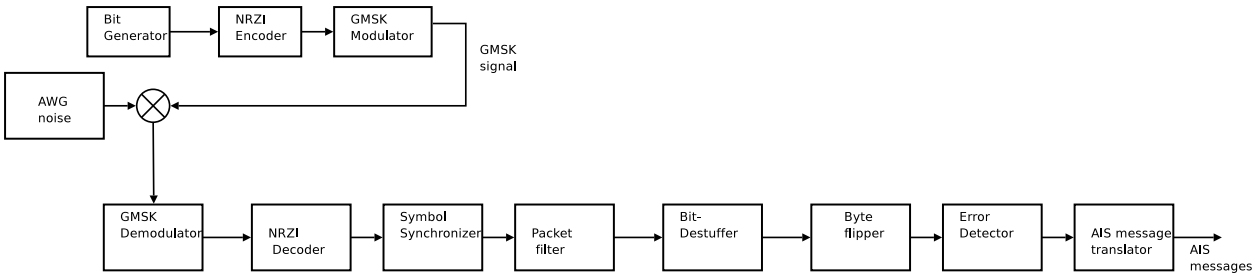


Figure 4.1: *Block diagram of a GMSK modem.*

and then on the SDR architecture of the University of Stellenbosch are:

- Non-return-to-zero inverted (NRZI) encoder
- GMSK modulator
- GMSK demodulator
- NRZI decoder
- Symbol synchroniser

- Packet filter
- Bit-destuffer
- Byte flipper
- Error detector
- AIS message translator

The next sections give the background theory of the components used to build the AIS receiver. The sections gives the Matlab development and the C++ model implementation. The C++ implemetation is done on the SDR architecture of the University of Stellenbosch.

4.2 Mathematical development

GMSK signals can be generated either by using a quadrature baseband process followed by quadrature phase modulation or by frequency modulation. The FM technique was used to develop the GMSK modulator in this paper. The demodulation process was implemented using a simple FM discriminator. The GMSK modulator was implemented by first differentially encoding the data, representing it as a non-return-to-zero inverted (NRZI) signal and then passing it through a Gaussian low-pass filter. The signal is then frequency modulated using a direct digital synthesis (DDS) system. The Gaussian-filtered signal was then re-sampled to the carrier sampling frequency, before frequency modulating the carrier. The GMSK modulator was designed to provide the GMSK signal to be demodulated by the proposed GMSK demodulator. The block diagram of the GMSK modulator is shown in Figure 4.2.

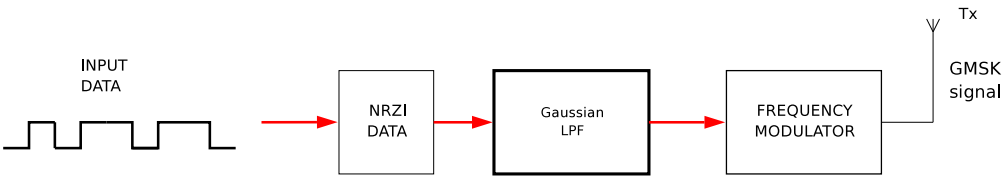


Figure 4.2: *Block diagram of a GMSK modulator.*

4.2.1 Data encoding and Gaussian filter design

The data encoding used in AIS is NRZI. The first step in the modulator is the NRZI encoder. The NRZI encoder gives a change in the signal level when it encounters a zero “0” in the bit stream and no change when a one “1” is encountered. The Gaussian filter component is designed using the coefficients of the Gaussian filter for a given bandwidth-time product. A Gaussian-shaped impulse response filter produces a signal spectrum with low side lobes and narrower main lobe than the rectangular pulse [7].

The impulse response of the Gaussian LPF is given by [20]:

$$h(t) = \frac{1}{\sqrt{2}\pi\sigma T} \exp\left(\frac{-t^2}{2\sigma^2 T^2}\right) \tag{4.1}$$

where

$$\sigma = \frac{\sqrt{\ln(2)}}{2\pi BT} \tag{4.2}$$

and B is the 3-dB bandwidth of the filter and T is the symbol period. Figure 4.3 shows Gaussian filters with BT = 0.3 and 0.5. The response of the Gaussian LPF to the square

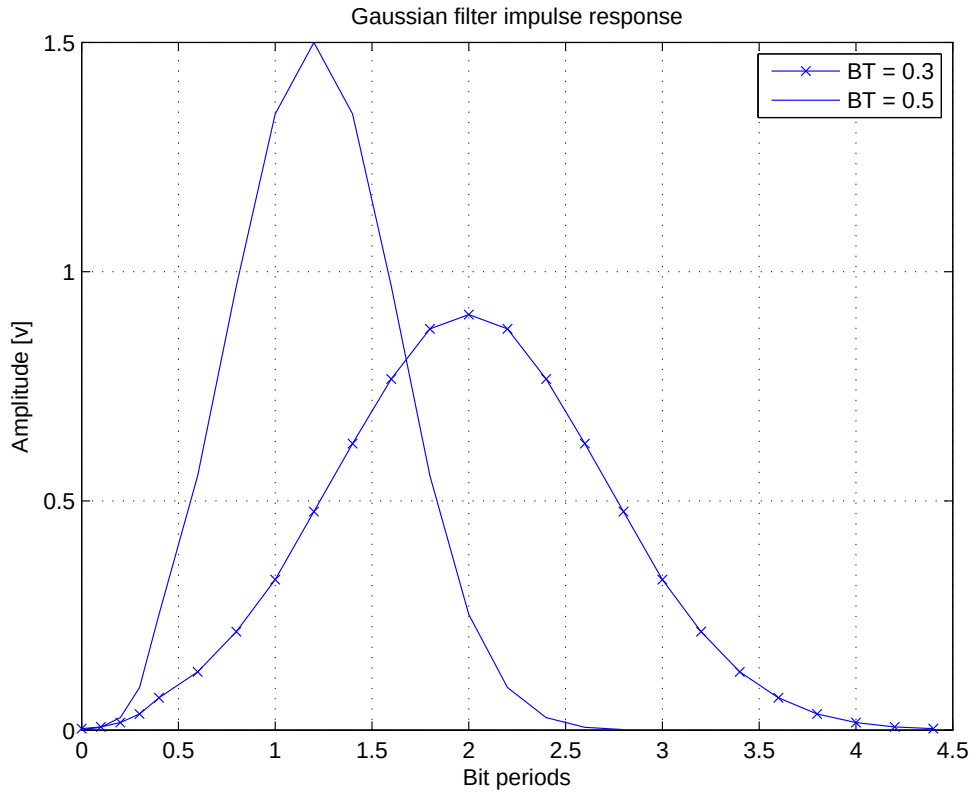


Figure 4.3: *Gaussian filter impulse response with BT of 0.5 and 0.3 respectively.*

pulses is equivalent to convolving the filter impulse response with the square pulses and is given by:

$$g(t) = h(t) * \prod\left(\frac{t}{T}\right) \tag{4.3}$$

The pulse response $g(t)$ (which is equivalent in the frequency domain to multiplying the Fourier Transform of $h(t)$ and the Fourier Transform of $\prod(t/T)$, can be written as:

$$g(t) = \frac{1}{2T}\, Q\left(2\pi BT \frac{t - T/2}{T\sqrt{\ln(2)}}\right) - Q\left(2\pi BT \frac{t + T/2}{T\sqrt{\ln(2)}}\right) \tag{4.4}$$

where $Q(t)$ is the function

$$Q(t) = \int_t^\infty \frac{\exp(-y^2/2)}{\sqrt{2\pi}}\, dy \tag{4.5}$$

4.3 Filters

Filters are frequency-selective circuits. An electronic filter can be described as a device that changes the frequency spectrum of a signal. Different filters are classified according to their frequency selectivity, e.g. low-pass, high-pass, bandpass and bandstop. Filters can be analysed in the frequency domain or in the time domain. Another way of classifying filters is by their frequency selectivity characteristics with respect to the filter's magnitude and phase response [24]. Some of the filter responses are Butterworth, Chebyshev and elliptic responses. Filters are usually characterised by their transfer function $H(s)$ which can be expressed as [38]:

$$H(s) = K\frac{s^m + a_{m-1}s^{m-1} + a_{m-2}s^{m-2} + \cdots + a_1 s + a_0}{s^n + b_{n-1}s^{n-1} + b_{n-2}s^{n-2} + \cdots + b_1 s + b_0} \tag{4.6}$$

From Equation 4.6 it is clear that the general s-domain transfer function of a filter can be described by the ratio of two polynomials, where the m and n specify the order of the numerator and denominator polynomials respectively. The constant K represents the overall gain constant of the system. The filter roll-off rate is determined by the value of n [38].

The zeros of the filter are obtained by factorising the numerator polynomial into first order-factors and the poles are obtained by factorising the denominator polynomial into first-order factors. The zeros and poles describe the characteristics of the filter [24].

4.3.1 Digital filtering

One way of describing a digital filter is by using its difference equation which is generally written as [24]:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k) \tag{4.7}$$

where the coefficients a_k and b_k determine the frequency response characteristics of the system. Using the z-transform, the resulting equation yields [24]:

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{4.8}$$

where z^{-k} represents the delay of the k^{th} sample. This equation is the ratio of two polynomials in z^{-1}. From this we can obtain the poles and zeros of the system function.

4.3.2 Digital filter implementation

Digital filters can be implemented directly from either their difference equation or the z-transform. One filter implementation is called the direct form I structure for both Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. Equation 4.8 shows a

system that characterises an IIR system which can be viewed as two systems in cascade, that is:

$$H(z) = H_1(z)H_2(z) \tag{4.9}$$

Where $H_1(z)$ consists of the zeros of $H(z)$, and $H_2(z)$ consists of the poles of $H(z)$. The direct form realisation is shown in Figure 4.4. The disadvantage of this structure is that

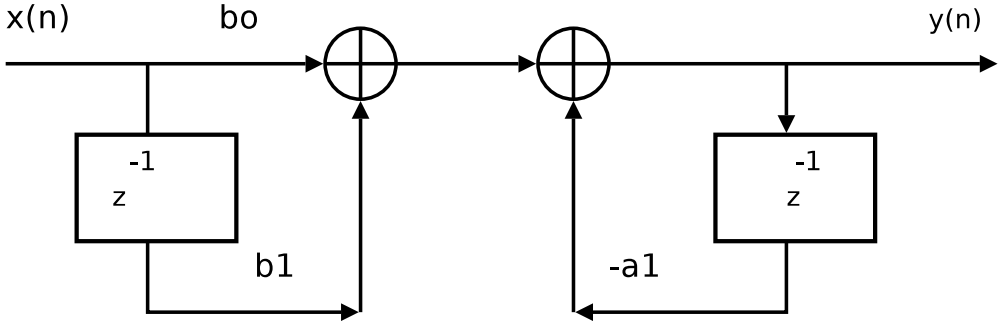


Figure 4.4: *Direct form I filter implementation (adapted from [24]).*

when large-order filters are required, the realisation requires $M + N + 1$ memory locations [24]. It is not memory-efficient since it requires two sets of delays, one for the input and one for the output. If the all-pole filter $H_2(z)$ is placed before the all-zero filter $H_1(z)$, a modified structure called the direct form II realisation is obtained [24]. This form yields two difference equations:

$$y(n) = \sum_{k=0}^{M} b_k w(n-k) \tag{4.10}$$

$$w(n) = -\sum_{k=1}^{N} a_k w(n-k) + x(n) \tag{4.11}$$

From these two equations, we see that only a single delay element array is needed. One disadvantage with this implementation is that a new structure is needed for different filter orders. One way of overcoming this problem is by factorising the system into a cascade of second order subsystems. The transfer function $H(z)$ for the second-order basic two-pole, two-zero IIR system has second order polynomials represented by [24]:

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{4.12}$$

An example of such structure is shown in Figure 4.5. Here, the coefficients $\{a_k\}$ and $\{b_k\}$ in the second order subsystems are real-valued. This means that, in forming the second-order subsystems, any pair of complex conjugate zeros or poles must be grouped. Any two real-valued zeros or poles can be paired together to form a quadratic factor. The procedure described above for digital filter implementation will be used to implement a Gaussian filter and $4th$ order low-pass filters. The Gaussian filter implementation is described in the next section.

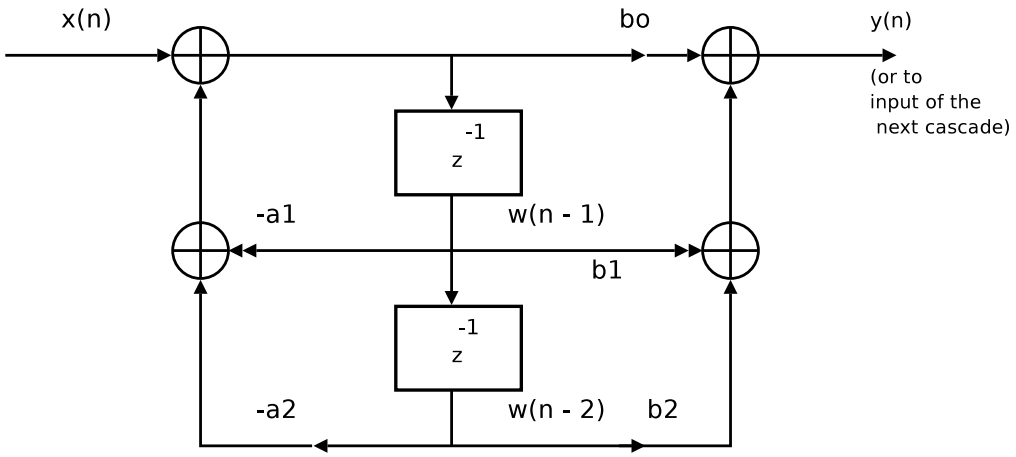


Figure 4.5: *Block diagram of an example of second order filter structure (adapted from [23]).*

4.4 Gaussian filtering

The Gaussian filter component takes each sample from the differential encoder and filters it using the Gaussian filter coefficients. The Gaussian filter with $BT = 0.3$ is a 25-tap FIR filter with coefficients given in [2]. The coefficients for the Gaussian filter with $BT = 0.5$ are also given in [2]. Appendix C shows the coefficients of the Gaussian filter for $BT = 0.5$ and 0.3 and how they are calculated. One way of calculating the Gaussian filter coefficients is by calculating the probability density function (*pdf*)

$$y(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{4.13}$$

with mean μ and variance σ^2. The function yields the coefficients of the Gaussian filter for a specified BT value. Appendix C shows the Gaussian filter impulse response plots from the calculated coefficients using Equation 4.13

4.4.1 Matlab implementation of Gaussian filter

The Matlab function `filter` was used to filter the NRZI waveform using the Gaussian filter coefficients. The syntax used in Matlab to achieve Gaussian filtering is:

```
GLPF = filter(B,1,MG)./8;
```

The `B` represent the coefficients for non-recursive part and `MG` is the input bit stream. The filtered signal was divided by eight in order to normalise the signal amplitude.

4.4.2 C++ implementation of Gaussian filter

The Gaussian filter function in C++ was implemented using a cascade of second order IIR filters as discussed in Section 4.3. The roots of the coefficients of the Gaussian filter (polynomial) were obtained using the suntax:

```
Z = roots(B)
```

where B represent the coefficients of the Gaussian filter. Any pair of complex conjugates roots were grouped together using the syntax:

```
Z1 = [Z(1) Z(2)];
Z2 = [Z(3) Z(4)];
Z3 = [Z(5) Z(6)];
```

where $Z1$,$Z2$,$\cdots$ are the roots of B. A polynomial was obtained for each pair of roots using the Matlab function poly. The syntax used is:

```
B1 = poly(Z1)
B2 = poly(Z2)
B3 = poly(Z3)
```

A second order filter structure is abtained for each polynomial $B1, B2, \cdots$. The following piece of syntax shows how a second order filter is implemented in C++.

```
double w0 = -a1 * w1 - a2 * w2 + read_input_port(port_in)->read_double();
write_output_port(port_out, (b0*w0 + b1*w1 + b2*w2)*gain);
w2 = w1;
w1 = w0;
```

where an is the filter coefficients for the recursive part and the bn are the coefficients for the non-recursive part. wn represent the memory elements. The second order filters are then connected in cascade to yield a Gaussian filter. The gain of the filter is obtained from the first coefficient of the Gaussian filter B.

The frequency modulator component then takes the incoming Gaussian-filtered samples and frequency modulates the carrier frequency using the direct digital synthesis technique. This technique is discussed next.

4.5 Direct digital synthesis

Direct digital synthesis (DDS) is used to digitally synthesise a desired analogue signal, usually a sine wave of which the frequency must be variable. DDS uses a fixed sampling frequency

and different output frequencies are obtained by sampling a sine wave using variable phase increments. If a higher frequency is desired, the sine wave is sampled at larger phase increments [36]. Figure 4.6 shows a block diagram of a DDS system. The heart of the DDS

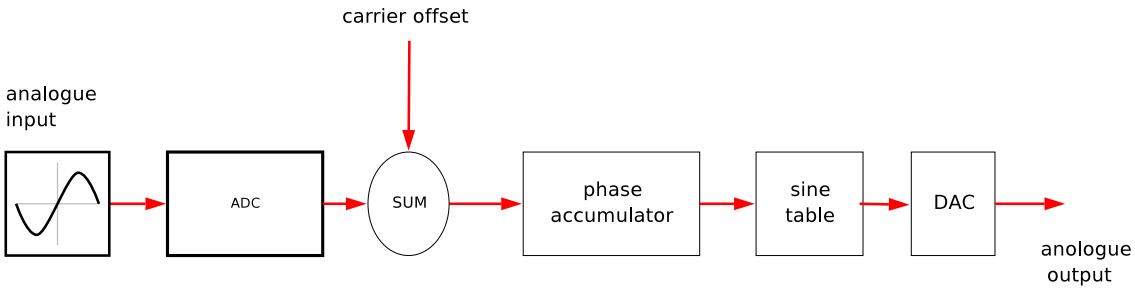


Figure 4.6: *Block diagram of a DDS system.*

system shown in Figure 4.6 is the phase accumulator. The value in the accumulator represents the current phase ($0 \leq \phi \leq 2\pi$) of the output signal wave. Every clock cycle, the phase of the accumulator is incremented by $\Delta\phi$. The size of this increment is directly proportional to the output frequency. Phase wrap from 2π to 0 radians is done by ignoring the phase accumulator overflow. The phase value in the accumulator can then be matched to a value in the sine or cosine lookup table, thus producing a sampled sine wave. The resulting sampled sine wave can be reconstructed to produce a desired output sine wave [36]. [21] shows that an output frequency of f_o can be produced if the phase increment is:

$$\Delta\phi = 2\pi \left(\frac{f_0}{f_s} \right) \text{ radians} \tag{4.14}$$

By dynamically changing the phase increment, it is possible to implement different modulation schemes. The instantaneous phase of an analogue frequency-modulated signal is given by:

$$\theta(t) = \omega_c t + k_f \int_0^t m(\tau) d\tau + \theta_0 \tag{4.15}$$

where ω_c is the carrier frequency in radians, k_f is the frequency deviation constant, $m(t)$ is the modulating signal, and θ_0 is the initial phase angle at $t = 0$.

To perform the frequency modulation (FM) in discrete time using DDS, the instantaneous phase of the FM signal can be calculated to be:

$$\theta(nT) = nTF_c + k_f T \sum_{k=1}^{n} m(kT) + \theta_0 \tag{4.16}$$

where T is the sampling interval, n is the sample number, F_c is the constant that determines the carrier frequency of the discrete-time output signal and θ_0 is the initial phase in radians. Sampling must be done below the half Nyquist frequency.

The DDS FM modulator can synthesise an FM signal directly at the desired broadcast frequency or at an intermediate frequency which can then be up-mixed to a desired broadcast frequency [36].

4.5.1 Matlab implementation of the modulator

The Matlab GMSK modulator was developed in Matlab using the procedure discussed in Section 4.5. The Gaussian-filtered data was re-sampled to the carrier frequency's sampling frequency before frequency modulating the carrier. The Matlab function `resample` was used to achieve this. The syntax used to calculate the instantaneous phase of the GMSK signal is:

```
for n = 1:length(MSG);
phi = phi + (2*pi*fd/fs).*MSG(n) + (fc/fs)*2*pi;
if (phi > (2*pi))
phi = phi-(2*pi);
end
module = [ module sin(phi)];
end
```

where `fd` is the frequency deviation, `fs` is the sampling frequency, `fc` is the carrier frequency, `MSG` is the input modulating signal, `phi` is the instantaneous phase and `module` is the modulated signal.

4.5.2 C++ implementation of the modulator

The GMSK modulator was implemented in C++ using the DDS algorithms used in Matlab. The only difference was that the sampling frequency was normalised and so was the carrier frequency and the frequency deviation. The syntax used to implement FM is:

```
double input_sample = read_input_port(port_INPUT)->read_double();
phase = phase + (2*M_PI*FD*input_sample) + delta_phi;
if (phase > 2*M_PI)
phase -= 2*M_PI;
write_output_port( port_Y, amplitude*Sin(phase) );
```

4.6 Noise

Noise is a random process. All practical electronic systems are affected by noise. The effect of noise on the signal can introduce distortions that can result in errors on the receiver side. Different types of internal noise to the system that affect electronic systems are thermal noise, shot noise and flicker noise [8]. Thermal noise is caused by the random motion of electrons in conducting media. Thermal noise is modeled by a noise generator with an open-circuit mean-square voltage of $4kTBR$, where k is Boltzmann's constant, T is the temperature in Kelvin, R is the resistance and B is the system bandwidth. The equation that describes

additive white Gaussian noise (AWGN) power (whose power spectral density is independent of frequency) in watts is [8]:

$$P_n = kTB \tag{4.17}$$

Thermal noise is considered to be AWGN process since it has a uniform power spectral density [30].

In practical systems, the modulated signal has to pass through a channel that adds noise to the modulated signal. In this thesis, noise was added to the modulated signal in order to demonstrate the demodulator performance in the presence of noise. In Matlab, the function `randn` was used to generate a length sequence of Gaussian random variables with zero mean. This sequence was then added to the modulated signal.

4.7 Automatic gain control system

Before demodulating the modulated signal, in most cases it is necessary to amplify the incoming radio frequency (RF) signal to a reasonable voltage level. This is because of the loss that the RF signal suffers along the channel. Losses occur when a part of the signal is reflected, absorbed, or diverted along the way. Many factors contribute to the signal degradation, e.g. path loss, and noise. A linear amplifier could be used to amplifier the signal before detection, but if the input signal suddenly increases, this results in the amplifier saturating. The results of saturating an amplifier introduces distortion on the output signal. An automatic gain control (AGC) system is used to vary the gain of the amplifier [15]. The incoming signal into the AGC is amplified by a variable gain. The variable gain is controlled by an error signal which is proportional to the difference between the input signal and the reference signal [25] If the input signal is very low, the AGC increases the gain and if the input signal is high, it decreases the gain of the system.

4.7.1 Matlab implementation of the AGC

A Matlab model of the AGC was developed. The desired reference level for the output signal is first set. The AGC operates like a voltage controlled amplifier. An error signal is fed back to determine the increase or decrease of the overall gain of the system. The syntax used to implement the AGC model function in Matlab is:

```
for i = 1:  length(Input)
Output(i) = Input(i) * vgain;
error = Ref - abs(Output(i));
imgain = vgain;
vgain = imgain + gain.*error;
end
```

where `INPUT` is the input signal, `Output` is the output of the AGC (output signal), `gain` is the fixed gain (initial gain of the system), `vgain` is the variable gain of the system, `Ref` is the desired reference level of the output signal and `error` is the error signal. The same algorithms used in Matlab were ported to C++ and no modifications were made in the development of the AGC model.

4.8 GMSK demodulator

The GMSK demodulator uses an FM discriminator detector. The ideal FM discriminator consists of a differentiator and a lowpass filter. Since GMSK is a special form of FM, the message is contained in the instantaneous frequency of the modulated signal. The demodulation process is performed as follows [26]: Let

$$x(t) = A_c \cos\left[\omega_c t + k_f \int_0^t m(\tau)d\tau\right] \tag{4.18}$$

be the GMSK signal, with k_f equal to $2\pi f_d$, where f_d is the frequency deviation, assuming a normalised $m(t)$. The differentiated signal is given by:

$$e(t) = -A_c\left[\omega_c t + k_f m(t)\right] \cdot \sin\left[\omega_c + k_f \int_0^t m(\tau)d\tau\right] \tag{4.19}$$

Rectifying the differentiated signal and lowpass filtering results in:

$$\begin{aligned} y(t) &= |-A_c\{\omega_c + k_f m(t)\}| \\ &= A_c\{\omega_c + k_f m(t)\} \end{aligned} \tag{4.20}$$

Removing the DC component yields

$$y(t) = A_c k_f m(t). \tag{4.21}$$

However, in the presence of additive Gaussian noise, the carrier amplitude variations cause distortion at the output of the discriminator. The receiver uses a bandpass filter to remove the out-of-band interference.

4.8.1 Software design of the FM discriminator

Figure 4.7 shows a block diagram of the SDR GMSK discriminator. The modulated signal is

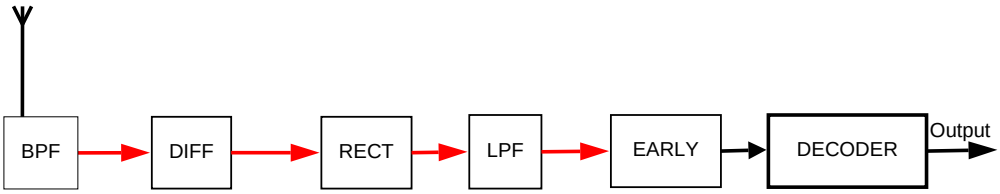


Figure 4.7: *Block diagram of the SDR GMSK demodulator.*

passed through a bandpass filter with sufficient bandwidth to let through the lower and upper sidebands and to remove out-of-band interference. The first step in the GMSK demodulation process is differentiation. Note that differentiation in the continuous-time domain can be approximated in the digital domain by calculating

$$\frac{d}{dt}\, x(t) \approx \frac{x[nT] - x[(n-1)T]}{T} \tag{4.22}$$

where $x(t)$ is the continuous modulated signal, T is the sampling period and n is a positive integer representing the sample index. The original message signal is proportional to the phase difference of the consecutive data samples. The differentiator computes a new phase difference vector for each new sample. The differentiated signal is then passed through the rectifier component. The rectification process is accomplished in software by taking the absolute value of the differentiated signal. The resulting signal is then divided by $2A/\pi$ (where A is the carrier amplitude). This is because the Fourier series representation of a full-wave rectified signal is [5]:

$$v(t) = \frac{2A_m}{\pi}\left(\cdots + \frac{1}{37}\, e^{-j3\omega_0 t} + \frac{1}{17}\, e^{-j2\omega_0 t} + \frac{1}{5}\, e^{-j\omega_0 t} + \cdots\right) \tag{4.23}$$

where the A_m is the signal amplitude. The $2A_m/\pi$ is the DC component which needed to be removed from the rectified signal.

The rectified signal is lowpass filtered using an 4th order Butterworth low pass filter with a cutoff frequency slightly higher than half of the symbol frequency. A filter with a high cutoff frequency is desired to avoid introducing intersymbol interference at the receiver. The DC component is removed by subtracting the $2\pi f_c$ from Equation 4.20. The baseband signal is recovered as follows:

$$y(t) = A_c\{\omega_c + 2\pi f_d m(t)\} - A_c\omega_c = A_c 2\pi f_d m(t) \tag{4.24}$$

and then dividing by the frequency deviation term to recover the baseband signal:

$$y(t) = \frac{\{A_c 2\pi f_d m(t)\}}{\{2\pi f_d\}} = A_c m(t) \tag{4.25}$$

The syntax used to implement differentiation in Matlab is:

```
for n = 1:length(MOULATED)
DIFFEREN(n) = (MODULATED(n) - last_sample)fs;
last_sample = MODULATED(n);
end
```

where `fs` is the sampling frequency. The syntax used to rectify the differentiated signal in Matlab is:

```
RECTIFIED = abs(DIFSIGNAL);
```

The Matlab function `butter` is used to implement a Butterworth filter.The syntax used to calculate the coefficients of the 4th order Butterworth low pass filter in Matlab is:

```
[B,A] = butter(4,1/(0.8*Ts*fs))
LOWPASS_SIG = filter(B,A,RECTIFIED)
```

where `Ts` is the symbol interval and `fs` is the sampling frequency. The term `1/(0.8*Ts*fs)` represent the cut-off frequency of 5 kHz.

4.8.2 C++ implementation of demodulator

The same algorithms used in Matlab were ported to C++ and no modifications were made in the development of the differentiator and the rectifier model.

Modifications were necessary during the development of the lowpass filter model in C++. Because there is no `filter` function in C++ programming language, a general purpose second order infinite impulse response (IIR) filter model was developed as discussed in Section 4.3. The filter model consist of programmable filter coefficients as attributes which the user can change at will and memory elements or delay taps which are initialised to zero. If a forth order or any higher order filter is required, a cascade of the second order filter model can be formed and this cascade will function just like the required fourth order filter or higher order filter desired as explained in Section 4.3. The filter coefficients are calculated from Matlab using the function `butter`. The syntax used is:

```
[B,A] = butter(4,1/(0.8*Ts*fs))
```

The piece of syntax used to implement the C++ second order filter is:

```
double w0 = -a1 * w1 - a2 * w2 + read_input_port(port_in)->read_double();
write_output_port(port_out, (b0*w0 + b1*w1 + b2*w2)*gain);
w2 = w1;
w1 = w0;
```

The low-pass filtered baseband signal must then be passed through the symbol synchroniser in order to extract the digital bits. The symbol synchroniser is responsible for making correct symbol decisions.

4.8.3 Symbol synchronisation

After successfully demodulating the modulated carrier, the receiver must have an accurate knowledge of the beginning of the symbol and the end of the symbol. This is necessary for the receiver to make correct symbol decisions [30]. If the receiver integrates over an

interval of inappropriate length, the ability to make accurate symbol decisions will be degraded. The early-late gate is the method chosen to achieve symbol synchronisation in this implementation [30].

The generic early-late gate synchroniser operates by performing two separate integrations of the incoming signal energy over two different (T - d) portions of the symbol interval, where T is the symbol interval and d = T/2 [30]. For the AIS demodulator, the symbol synchroniser was modified by performing three separate integrations instead of two. This was done in order to use the same synchroniser when demodulating M-ary pulse amplitude modulated (PAM) signals, resulting in a more widely reusable component. Figure 4.8 shows a block diagram of the early-late gate symbol synchroniser. The first integration, I_1, (the early

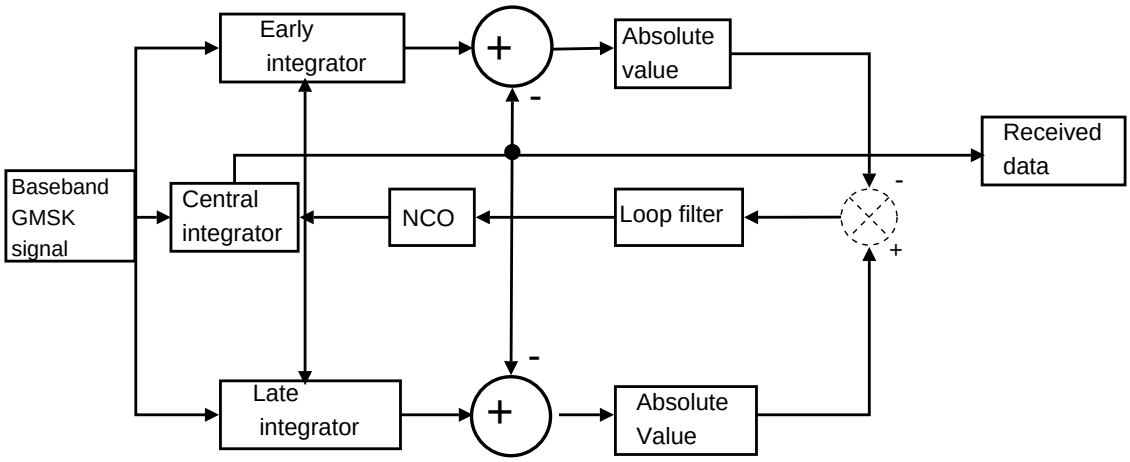


Figure 4.8: *Block diagram of an early-late gate symbol synchroniser.*

interval) starts the integration at the loop's estimation of the beginning of a symbol period (the nominal time zero) and integrates to T/3, where T is the nominal symbol period. The second integral, I_2, (the central interval) delays the start of its integration for T/3 seconds, and then integrates to $2T/3$. The last integral, I_3, (the late interval) delays the start of its integration for 2T/3 seconds and then integrates to the end of the symbol period (the nominal time T). The measure of the receiver's symbol timing error is now given by:

$$\epsilon = |I_2 - I_1| - |I_2 - I_3| \tag{4.26}$$

which can be fed back to the loop's timing reference to correct loop timing. When the synchroniser achieves lock, the (early and late intervals) integrators will accumulate the same signal energy and $\epsilon \approx 0$. Thus, when this device is synchronised, it is stable and there is no tendency to drive itself away from synchronisation [30].

4.8.4 Matlab early-late gate symbol synchroniser

The demodulated signal is then passed through the early-late gate synchroniser for bit synchronisation, as described in Section 4.8.3. The Matlab early-late gate synchroniser was developed. The following piece of syntax taken from the Matlab m-file used to implement the early-late gate symbol synchroniser shows how the early, central and late intervals integrators are calculated:

```
while n <= length(INPUT)
if SICS <= SPSD
Early = (SICS-1) * EARLY_INTERVAL + INPUT(n);
EARLY_INTERVAL = (Early)/SICS;
elseif (SICS > SPSD) & (SICS <= (SPS-SPSD))
Central = ((SICS-SPSD-1)) * CENTRAL_INTERVAL + INPUT(n);
CENTRAL_INTERVAL = (Central)/(SICS-SPSD);
elseif (SICS > (SPS-SPSD)) & (SICS < SPS)
Late = (SICS-SPS + SPSD-1) * LATE_INTERVAL + INPUT(n);
LATE_INTERVAL = (Late)/(SICS-SPS+SPSD);
```

where `SICS` is the sample index in the current symbol, `SPSD` is the number of samples in each interval and `SPS` is the number of samples per symbol.

The central integration result I_2, is used to decide whether the symbol represents a "1" or a "0". The recovered symbols are finally passed through the differential decoder component. The Matlab early-late gate symbol synchroniser model was migrated over to C++ and no modifications were needed. The following piece of syntax is taken from the C++ early-late gate symbol synchroniser:

```
Late = (sample_index - SPS + NB_samples - 1) * ALI + new_sample;
ALI = Late/(double)(sample_index - SPS + NB_samples);
if (ACI > 0)
symbol = 1;
else
symbol = 0;
```

4.8.5 NRZI decoder

The recovered symbols from the early-late gate symbol synchroniser are finally passed through the differential decoder shown in Figure 4.9. The decoding process is performed by comparing the received symbol and its delayed symbol version to reverse the encoding process, thus recovering the transmitted symbols. The Matlab NRZI decoder was migrated over to C++ and no modifications were made.

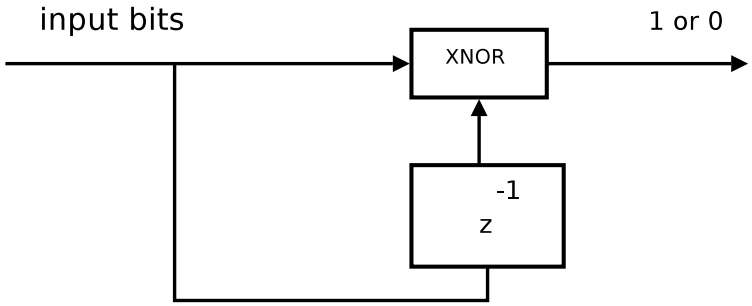


Figure 4.9: *NRZI decoder Block diagram.*

After decoding, the start flag and stop flag must be extracted from the AIS message data. The next section describes the extraction of the flags.

4.9 Start flag and stop flag detection

AIS packets consist of a 24-bit training sequence which is used to synchronise the receiver. The preamble is followed by an 8 bit start flag (01111110) which marks the start of each transmission packet. The data portion follows the start flag and then the FCS bits which are followed by the stop flag. It is necessary to extract the message data bits from the entire packet that is received in order to effectively decode and interpret the encapsulated binary data. The Matlab packet-filter model was developed. This component monitors the incoming bit stream and whenever it detects a start flag, it discards the previously received bits and then stores all the bits received after the start flag. It then monitors the recorded message for a stop flag. When a stop flag is detected, the entire recorded message is then passed over to the next component for bit-destuffing. The following piece of syntax taken from the Matlab packet filter model shows the detection of the start flag:

```
while m <= length(decoded);
if (isequal([0 1 1 1 1 1 1 0],[buffer(1:end)]))
disp('start bit reached');
break
end
m = m + 1;
buffer = [buffer(2:end) decoded(m)];
end
```

After detecting the start flag, the packet-filter model searches for the stop flag. When the stop flag is detected, the extracted data bits are transferred to the next component for bit-destuffing.

4.9.1 C++ packet-filter model

Modifications were necessary for the implementation of packet-filter model in C++. Practical issues had to be taken into consideration when developing this model. It has to be taken into consideration that this model will function in real time. In AIS, messages are transmitted every 26.6ms (which is equivalent to one time slot). In Matlab, it was assumed that the first flag to be detected by this component on the received message will be a start flag.

However, in real-time system, when the system is switched on, the first flag to be detected can either be a start flag or a stop flag, because other AIS stations might have been transmitting all along. It was necessary to consider packet synchronisation. If packet synchronisation is not achieved and the stop flag is mistaken for a start flag, the entire packet and the preceding packets will be discarded because they will always fail the CRC check. Again it should be remembered that if data in the transmitted packet is corrupted by noise, this will result in the packet failing the CRC check.

The modifications introduced was the condition that if the packet failed the CRC check, then the stop flag of that packet must be treated as a start flag of the next packet. This required the knowledge of the CRC check results before processing the next packet. But the CRC check is performed by another model component after the packet synchronisation. This necessitated a feedback loop where the model that does the CRC check had to inform the packet-filter model about the status of the CRC check. Another condition that needed to be implemented on this model was that if the stop flag appears right after the start flag, the next model in the system (bit-desuff) must inform the packet- filter model. This too necessitated a feedback loop. If this is not taken into consideration when designing the system, the whole system goes into a wait mode because no samples are processed by the preceding modules. On the other hand, the packet-filter module never receives a feedback about the CRC check results. The packet-filter model flow chart is shown in Figure 4.10.

4.10 Bit de-stuffing

After the symbols are decoded, the start flag and end flag are removed. The decoded message is then bit de-stuffed. This is the reverse process of bit stuffing. This means that if five ones are found in the decoded message, the next bit, a zero must be removed from the bit stream. The Matlab bit de-stuff model was developed. This component traces five consecutive bits in the decoded message and if it encounters five ones, it removes the next bit (a zero). The syntax used to implement bit-destuffing in Matlab is:

```
if isequal(message(n-5:n),[1 1 1 1 1 0]),
messag(n) = [];
```

where, `message` is the incoming data bits and `messag` is the output of the bit-destuff component.

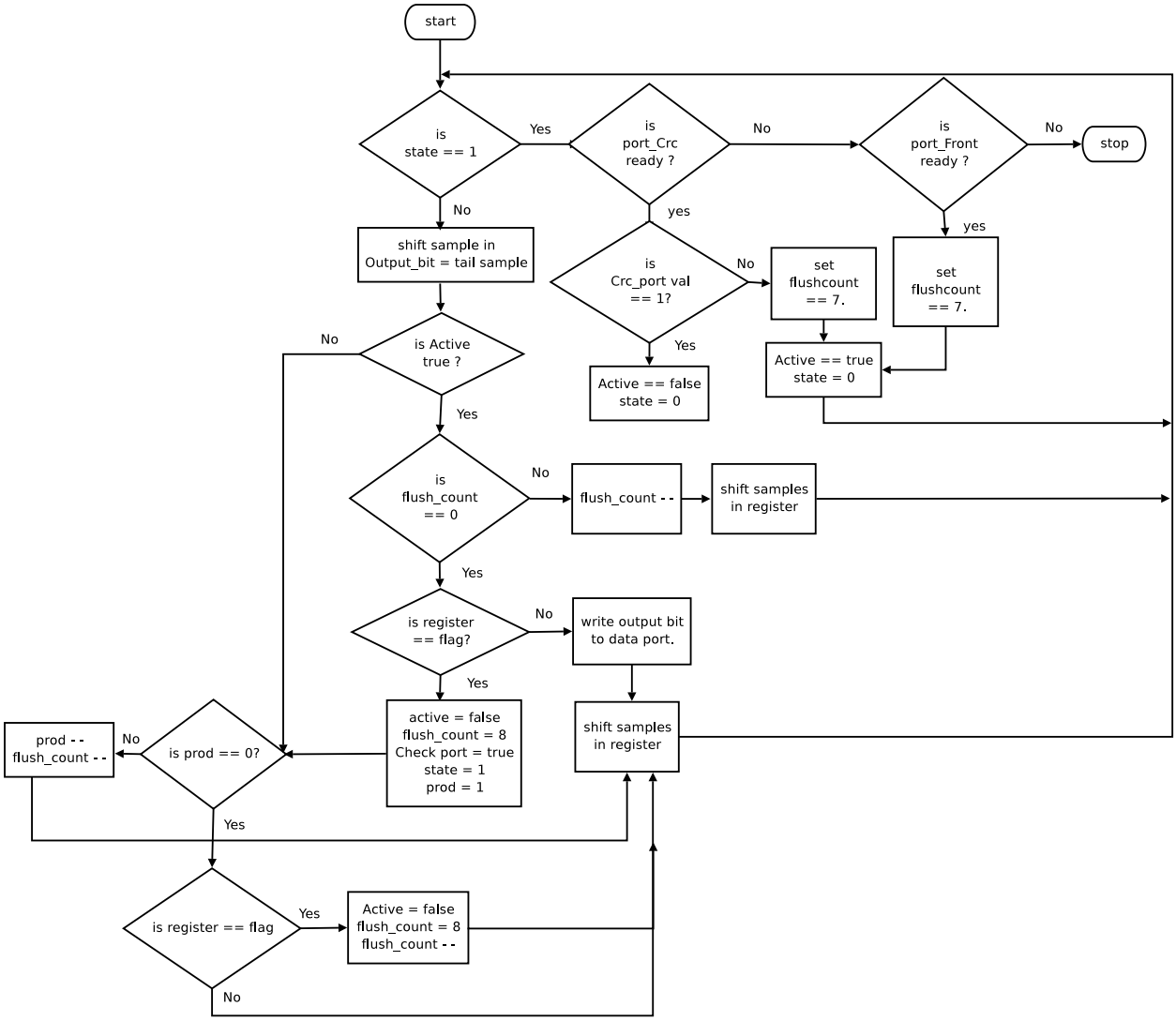


Figure 4.10: *The flow diagram for the packet filter.*

The Matlab bit de-stuff model was ported to C++ and certain modifications were made in order for the system to operate effectively in real time. The modifications which were made are:

1. If the model is informed that a stop flag has been detected by the previous component, but no samples are available on the input port, this component must feedback a one "1" to the previous component. Figure 4.11 shows the flow diagram of the bit-destuff component.

2. The maximum number of characters that an AIS standard sentence can contain is 82 x 6-bit characters [13] and this is equal to 492 bits. Adding the 16 CRC bits, the maximum number of data bits that can be contained in one AIS message is 508 bits. So, the modification was,if there are more than 520 bits on the input port, the samples must be discarted.

4.11 Byte grouping

After the bit de-stuffing process, the message is grouped into bytes. But each byte must be reversed (flipped) in order to accurately represent the recovered message. This is done because on the transmitter side, each byte is transmitted the least significant bit first. In Matlab, this process is accomplished by storing every eight symbols and reversing them, with the first bit occupying position eight and the eighth bit occupying position one in every byte. The syntax used to group the message into bytes is:

```
while i <= (len/8)
bytes = [bytes MSG(i*8:-1:1+(8*(i-1)))];
i = i + 1;
end
```

The same algorithms were ported to C++.

4.12 Error detection

Error detection and control is performed by using the CRC polynomial. The system works in such a way that if an error occurs in the received message, the entire message is discarded and no further action is taken by the AIS. The FCS uses the cyclic redundancy check (CRC) 16-bit polynomial to calculate the checksum. At the beginning of the CRC calculation, the CRC bits are pre-set to one. Only the data bits are included in the CRC calculation. AIS uses CRC-CCITT-16 for a FCS. At the receiver side, the FCS is performed on the data bits and the appended 16 bit CRC checksum.

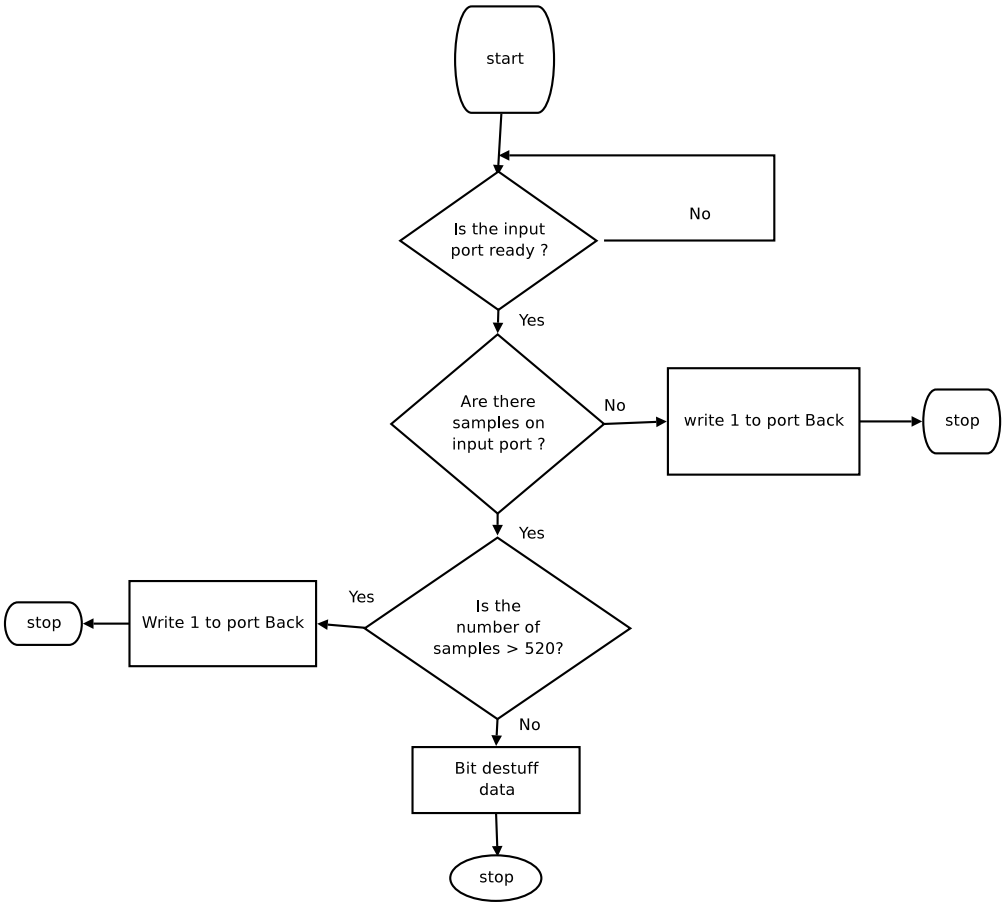


Figure 4.11: *The flow diagram for the bit-destuff model.*

The Matlab model was developed to calculate the checksum on the receiver side. The process uses lookup table. The CRC bits are shifted to the right eight times. This shifting process is equivalent to dividing the 65535 by 256. The result is exclusive-ORed with the relevant value in the lookup table. The syntax used to calculate the CRC checksum is:

```
CRC = bitxor[fix(CRC/2^8), LOOKTABLE(bitand(bitxor(CRC, msg), 255) +1)];
```

where `msg` is the incoming message byte.

If there are no errors in the received message, the calculated checksum on every message is F0B8 hexadecimal or 61624 decimal.

4.12.1 C++ Error detection model

The same algorithms were ported over to C++ for the development of the FCS model and certain modifications were made. Three output ports were necessary for this model, one for the data message excluding the CRC bits, one for the feedback path to send information about the status of the CRC check to the packet-filter and one for the status of the CRC check information to the next component. The flow-chart of the error detection model component is shown in Figure 4.12.

4.13 Message translation

After the received message has passed the FCS test, it is first organised into six-bit strings. This is done in order to convert the 6-bit strings into their representative valid characters [12, Table 7]. The valid characters are then assembled into an encapsulation string. But in Matlab the ASCII characters are eight bits long instead of six bit long. In order to be able to convert the six bit binary strings into valid characters, a mathematical relationship was established between the six bit binary representing the characters and the eight bit binary representing the same character. The relationship is that:

If the six bit binary number is less than forty, add 48 to that value and the resulting decimal number is equal to the corresponding eight bit ASCII character. But if the binary number is greater than 39, then add 56 to the six bit binary number [13]. The syntax used to implement this is:

```
if (number <= 39)
bnumber = number + 48;
elseif (number > 39)
bnumber = number + 56;
end
```

The six bit binary strings are again organised using the rules contained in ITU-R M. 1371-1 [10, Table 15]. Appendix D shows the example of how binary bits are organised for Message

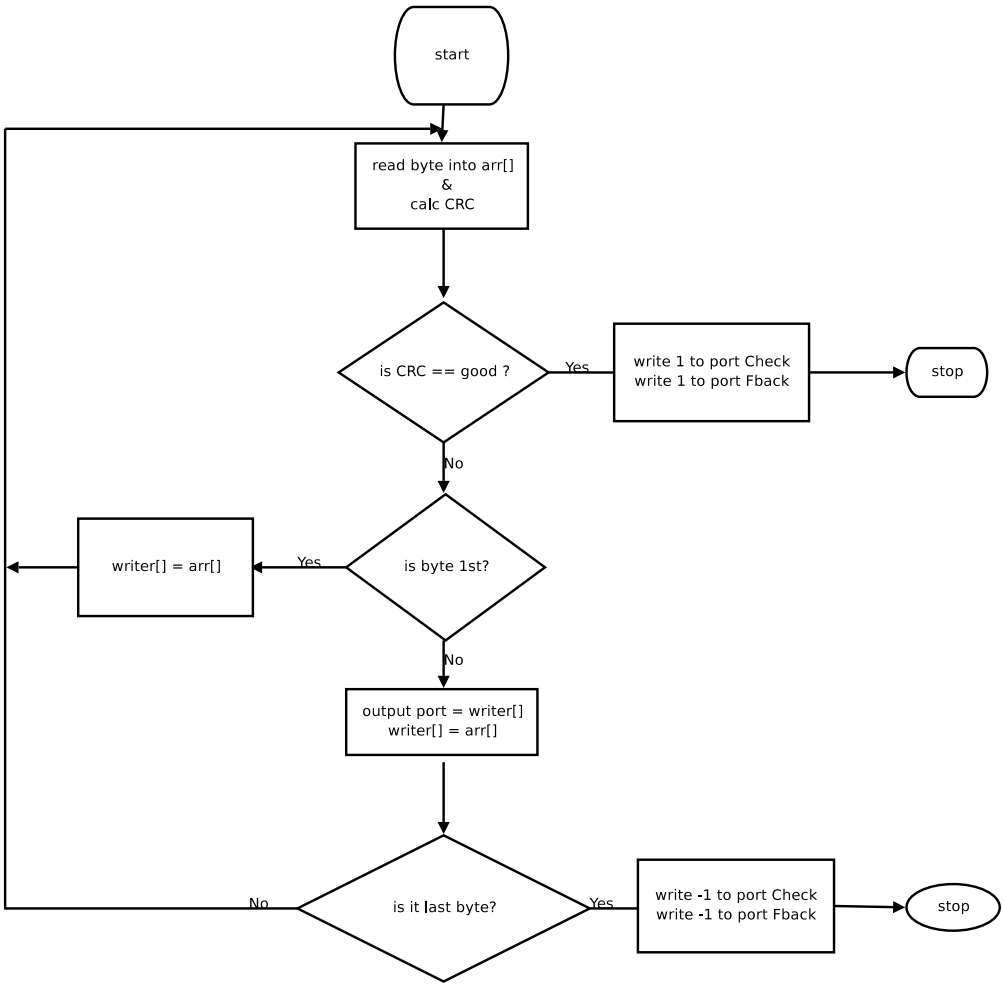


Figure 4.12: *The flow diagram for the CRC byte filter model.*

one “1” and according to ITU-R M. 1371-1. The rules in the referenced document are used to convert the binary message into relevant information. The first step in the process of organising the decoded binary message data into relevant information is the identification of the message number. The first 6 bits of the received message represent the message identifier. Once the message is identified, the rest of the binary bits are organised following the appropriate message table [13].

The syntax used to convert the binary encapsulated message into meaningful information such as message identifier is:

```
MSG_ID = bi2de(bytes_flipped_msg(1:6),'left-msb');
Repeat_indicator = bi2de(bytes_flipped_msg(7:8),'left-msb');
MMSI_NUMBER = bi2de(bytes_flipped_msg(9:38),'left-msb');
```

The same Matlab algorithms were ported over to C++ for the implementation of the model that change the 6-bit binary strings into an encapsulated string and the model that translate the binary strings into meaningful information. The syntax used to convert binary numbers to a decimal numbers is:

```
for (int i = 0; i < count; i++)
parameter = 2*x + buf[i];
x = parameter;
```

The following piece of syntax shows how binary strings are translated into meaningful information in C++:

```
if (word_len == 6)
write_output_port(port_Output, parameter);
cout << "message ID is" << " " << parameter << endl;
count = 2;
MSG_ID = parameter;
else if ( word_len == 8 )
write_output_port(port_Output, parameter);
cout << "Repeat indicator is" << " " << parameter << endl;
count = 30;
```

Chapter 5

Physical layer simulation

5.1 Introduction

This chapter consists of two experimental parts. The first part involve the simulation of the GMSK modulator and demodulator using a random number of bits generated in Matlab. The binary bits were modulated to provide the modulated signal for the proposed GMSK demodulator. The Matlab modem was evaluated in a noise-free environment and then in a noisy environment. The second experiment involved demodulating recorded actual AIS signals.

5.2 GMSK modem simulation

In this chapter the performance of the proposed GMSK modem is evaluated, first in ideal noise-free conditions, then in the presence of additive white Gaussian noise (AWGN). The GMSK modem was simulated using Matlab. The simulation was performed using a data rate of 9600 bits-per-second (bps), a carrier frequency of 100 kHz, frequency deviation of 12.5 kHz, a carrier sampling frequency of 1 536 000 samples/s and both $BT = 0.5$ and BT = 0.3. An input frame length of 1 000 bits was used in the bit error rate (BER) performance simulation. Figure 5.1 shows the Gaussian-filtered bits. Because the information bits are sampled at 76800 samples/s and the carrier frequency is sampled at 1536000 samples/s, the Gaussian-filtered bit stream must be up-sampled to the carrier frequency's sampling frequency. Figure 5.2 and Figure 5.3 show the differentiated and rectified signals respectively.

The results show the message signal contained in the envelope of both the differentiated and rectified signal. The DC component at 6.28×10^5 corresponds to carrier frequency ω_c in radians per second.

5.3 Low-pass filtering and signal recovery

A fourth order Butterworth low-pass filter was used to filter-out the high-frequency components. The cut-off frequency of the low-pass filter was set at 5 kHz. Figure 5.4 shows that the

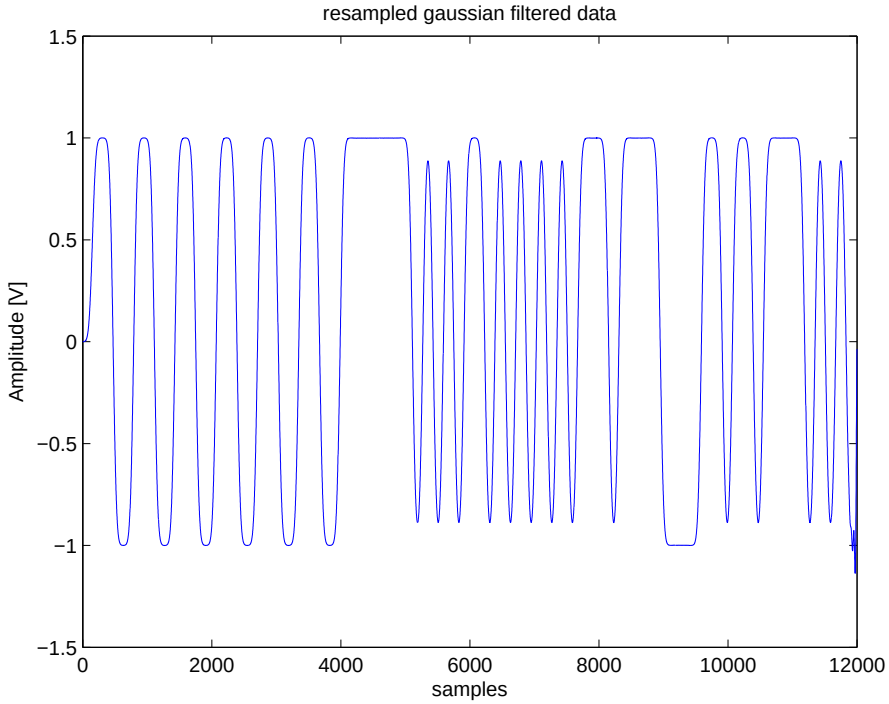


Figure 5.1: *Re-sampled Gaussian-filtered data. A Gaussian filter with BT= 0.5 was used in this experiment.*

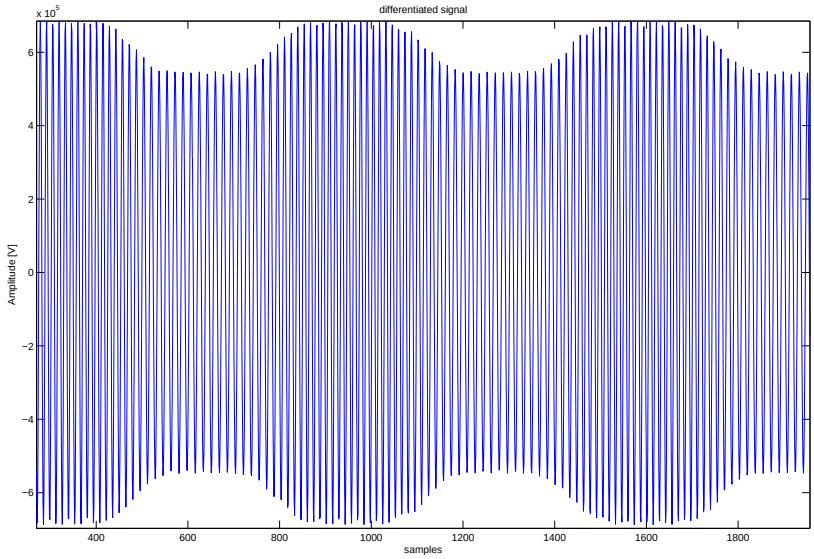


Figure 5.2: *The differentiated FM signal.*

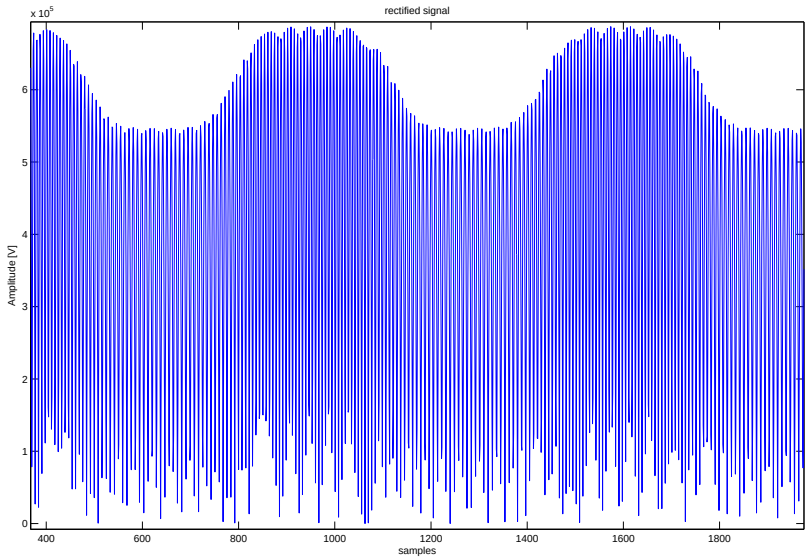


Figure 5.3: *After differentiation, the received FM signal is rectified in order to extract its envelope.*

low-pass filter has removed the high-frequency components of the original modulated signal. The low-pass filtered baseband signal in Figure 5.4 shows glitches or ringing. The glitches are caused by the low order low-pass filter. The recovered baseband signal still contains an unwanted DC component proportional to the carrier frequency. The DC component is removed by subtracting the carrier frequency in radians, and then dividing by the frequency deviation factor. Figure 5.5 shows the low-pass filtered signal after passing through the early-late gate symbol synchroniser. We see that at the beginning, synchronisation has not yet been achieved and this is indicated by diamonds displayed on the graph. As the synchroniser continues to feed back the symbol timing error to the loops timing reference to correct loop timing as in Section 4.8.3, there comes a time when the early and late integrators accumulate the same amount of signal energy and $\epsilon \approx 0$. When this happens, the synchroniser displays circles on the graph to show that synchronisation is achieved.

The simulation was performed using the 24-bit training sequence specified for AIS, an 8-bit start flag, and then randomly generated bits representing the message signal. The results show the training sequence of alternating ones and zeros, followed by start bits and then the message data. Figure 5.6 shows the demodulated signal when $BT = 0.3$ is used. The effect of the intersymbol-interference is displayed on the demodulated signal.

5.4 The effect of noise on demodulation process

Figure 5.7 and Figure 5.8 shows the rectified signal and the recovered signal after the early-late gate synchroniser in the presence of additive Gaussian noise. The degradation of the

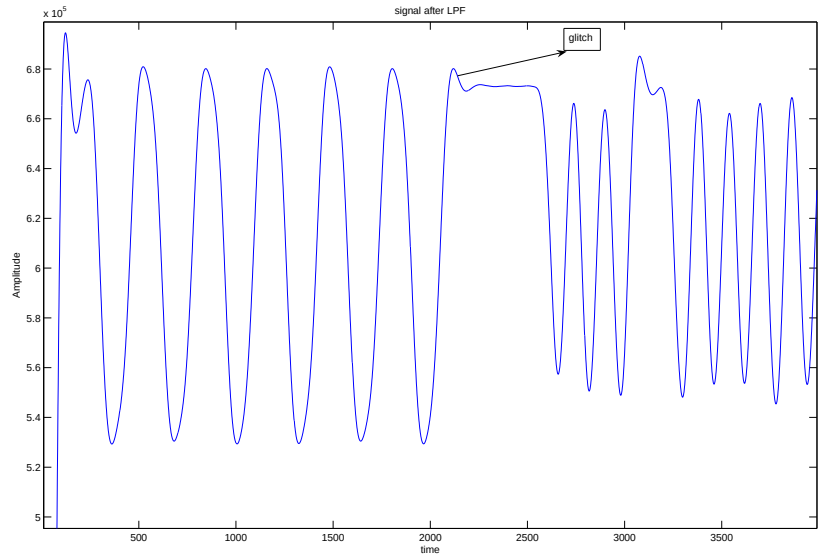


Figure 5.4: *The rectified FM signal after low pass filtering. The modulating signal has now been recovered.*

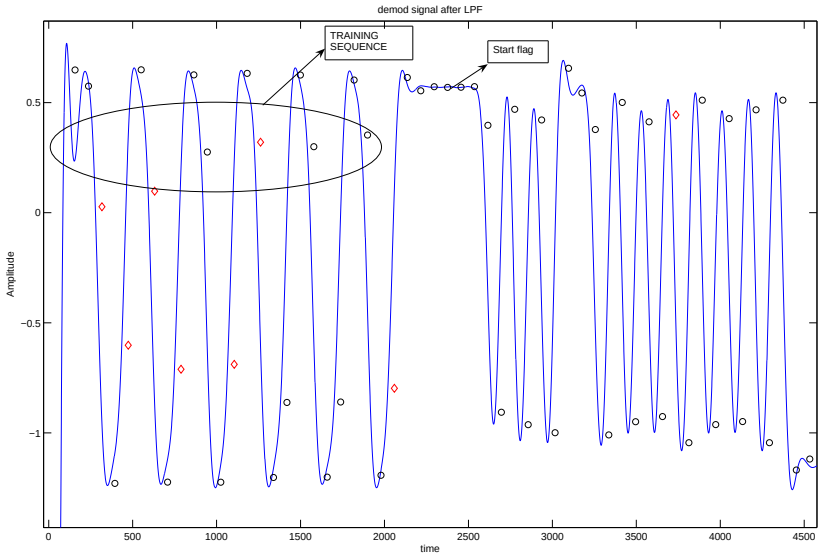


Figure 5.5: *The demodulated GMSK signal after symbol synchronisation*

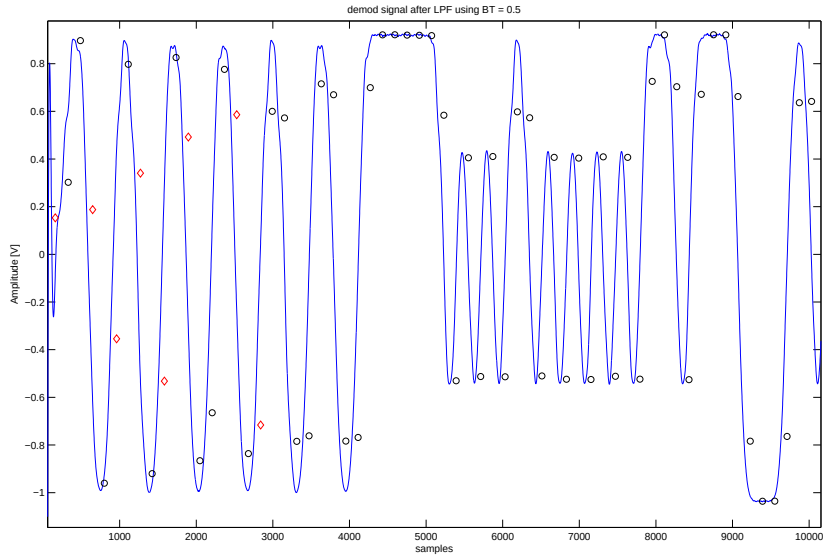


Figure 5.6: *The demodulated GMSK signal after symbol synchronisation using $BT = 0.3$.*

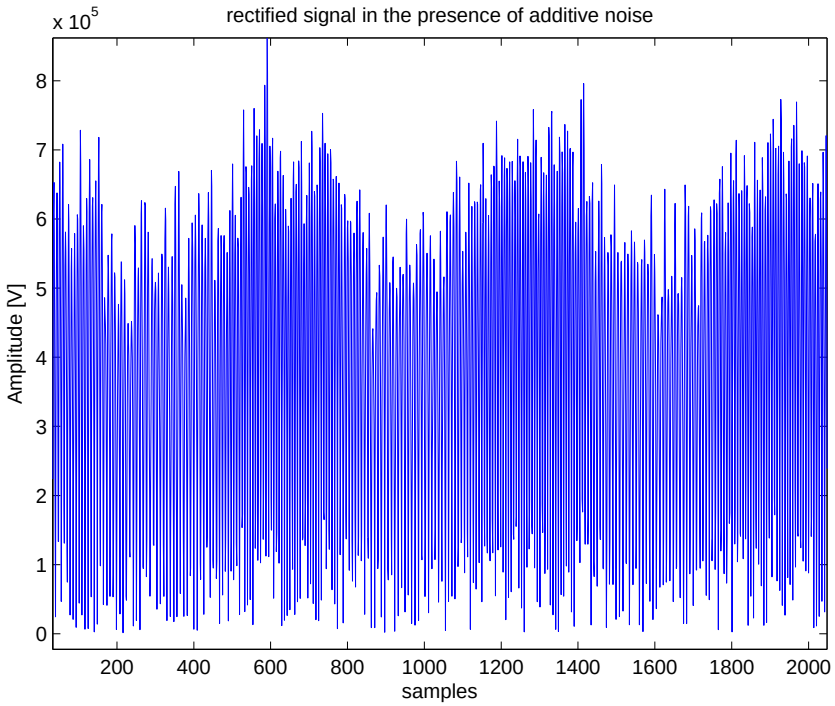


Figure 5.7: *The rectified signal with AWGN at 10 dB SNR.*

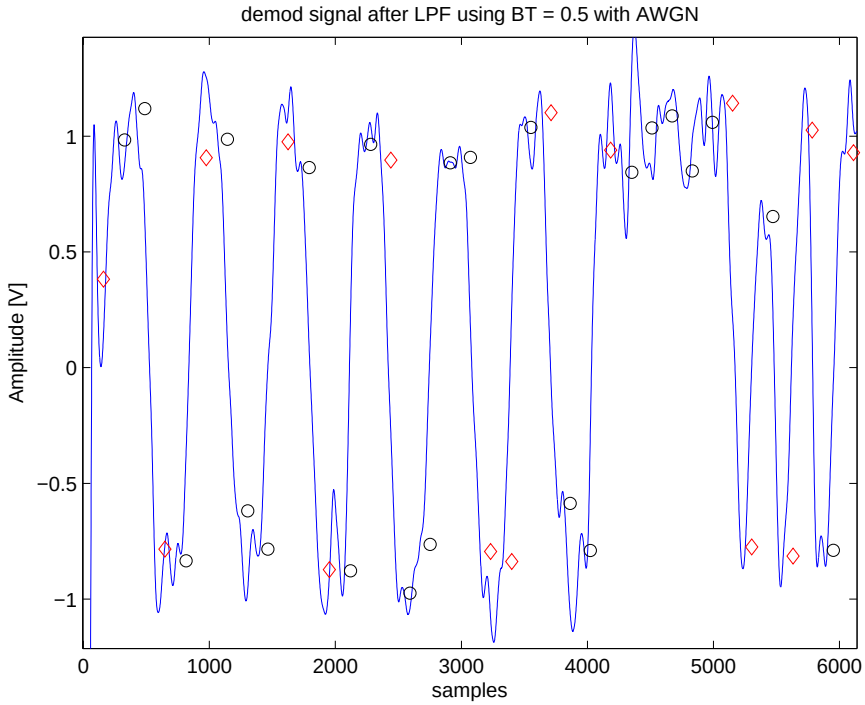


Figure 5.8: *Demodulated baseband signal after symbol synchronisation in the presence of AWGN.*

signal by the noise is evident in the rectified signal. This is shown by the amplitude variations of the rectified signal. This results in errors being introduced in the recovered bits.

5.5 Frequency spectrum of the GMSK signal

Figure 5.9 shows the power density spectrum of the GMSK signal with frequency deviation of 12.5 kHz and BT = 0.5. The spectrum in Figure 5.9 was generated by windowing the GMSK signal by the Hamming window. The results show that the Gaussian filter helps to prevent the GMSK signal power from spilling into adjacent channels. The spectrum has lower sidelobes outside of the 25-kHz bandwidth, about 30dB below the main lobe. The maximum permissible adjacent-channel interference limit for radio transmissions is -60 dB [23]. According to [23], both GSM and Digital European Cordless Telephone (DECT) may not meet the recommended value of -60 dB. For DECT, an adjacent-channel interference of only -40 dB is guaranteed [23].

5.6 Bit error probability

Figure 5.10 shows the bit error probability performance of the discriminator receiver for GMSK in the presence of additive Gaussian noise using BT = 0.5. The simulated results

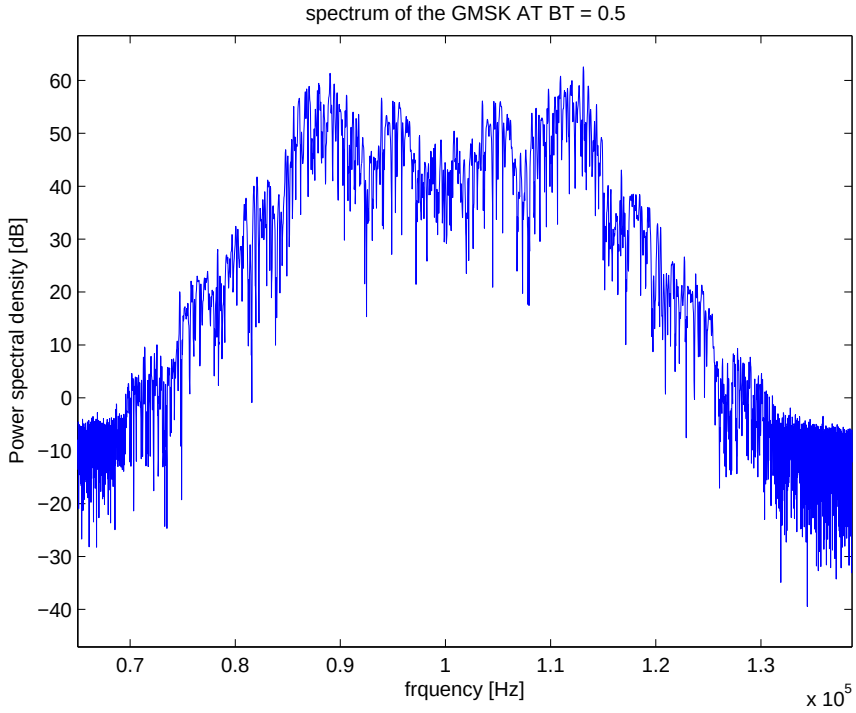


Figure 5.9: *Frequency spectrum of a GMSK signal with a frequency deviation of 12.5 kHz and BT = 0.5.*

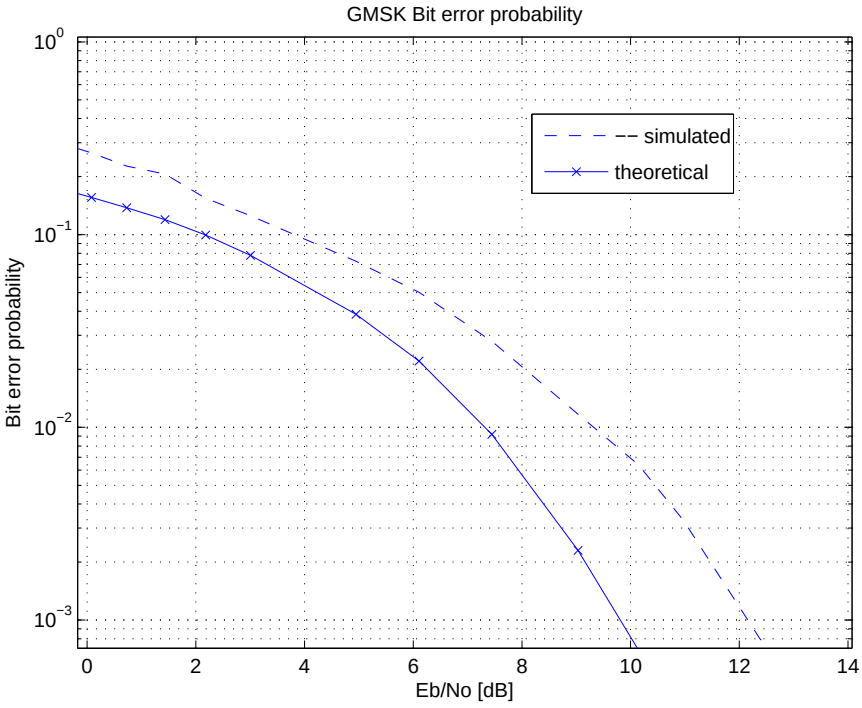


Figure 5.10: *Bit error probability curve for GMSK.*

differ from the theoretical results by about 2.5 dB. The simulated bit error probability is 2.5 dB worse than the theoretical bit error probability because of the low-order filter used. A low-order filter does not give a sharp cut-off frequency as compared to a higher-order filter which gives a sharp cutoff frequency. The low-order low-pass filter allows some of the out-of-band noise components to pass through, resulting in poor bit error probability than the theoretical one.

5.7 Simulation using actual AIS signals

The second part of the experiment involved simulation using actual AIS signals. Actual AIS signals were transmitted using the Marine Data Solution AIS instrument. The AIS signals were demodulated using a Hewlett Packard (HP) 8920A RF Communications Test Set at a frequency of 162.025 MHz. The demodulated signals were recorded using a 14-bit, 4-channel, 2MS/s data acquisition card. The rx_logger.cpp file was used to record the signals to a file. The recorded samples were passed through the Matlab early-late gate symbol synchroniser for symbol synchronisation. The symbols were then decoded using the NRZI decoder. After decoding, the decoded message was passed through the packet filter component for the extraction of the start and stop flag. The data packet was then bit de-stuffed before being organised into bytes. The symbols were passed through the FCS and if the message passed the CRC test, the message was finally translated to meaningful AIS information. Figure 5.11

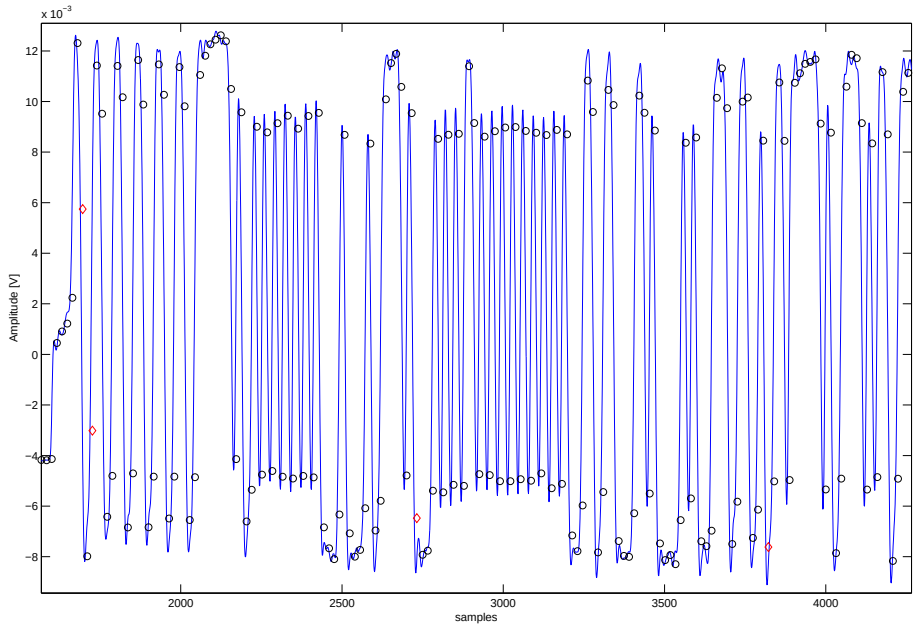


Figure 5.11: *Demodulated AIS baseband signal after symbol synchronisation.*

shows the actual AIS signal at the early-late gate symbol synchroniser. The encapsulated binary coded data obtained from the message in Figure 5.11 is found to be:

```
!AIVDM,1,1,,B,103>rCPP001FGCidUO;aD?vb0<00
```

The encapsulated binary coded data was translated into meaningful information using the AIS translator developed in Matlab.

5.8 Summary

In this chapter the performance of the GMSK demodulator was evaluated on Matlab generated signals and then on real AIS signals. The demodulator performed well in a noise-free environment. The modulating signal was recovered successfully. The effect of the low-order lowpass filter resulted in glitches on the demodulated signal. The early-late gate symbol synchroniser performed as expected and achieved lock within the training sequence. The addition of noise on the modulated signal resulted in errors during the detection process.

The bit error probability performance of the demodulator in the presence of noise was evaluated. The simulated bit error probability was 2.5 dB worse than the theoretical one due to the low-order filter being used. The performance of the demodulator on actual AIS signals was evaluated. The real AIS signals were symbol-synchronised and then translated correctly.

Chapter 6

Implementation in C++

6.1 SDR Architecture

Stellenbosch University (SU) has been involved in SDR research since 1998. The university is developing a software architecture for rapid SDR development. The goal is to develop a library of reusable radio building blocks that are portable to many platforms. The lowest level of the SDR architecture is the converter layer, which is equivalent to analogue systems (mixers, filters etc.) of the hardware defined radio [3]. The converter layer is responsible for all the signal processing functionality of the system. [3] says:

> The converter layer is made up of a collection of converters that derive from an abstract base class called sdr-converter.

The middle layer is called the subcontroller layer. The subcontroller's function is to manage all the converter components under its control and to communicate information from the top-level application to the converters. The subcontroller controls information flow between converters too. Data flow between converters operate on the push principle, which means that only the input buffers of converters need buffering. The control application provides the user interface for the system [3].

6.1.1 SDR Functionality

The SU SDR architecture allows the development of a library of components that are used to build a radio system [3]. Figure 6.1 shows a generic SDR component. A typical SDR component models a hardware component. The SDR components operate on the principle that all components receive, process and output streams of samples as illustrated in Figure 6.1, regardless of the number of input and output ports. The exact process function of each component is defined by the type of component being modeled [3].

6.1.2 Component's attributes

Attributes are parameters of each converter component. Examples of attributes are amplifier gain and filter coefficients. In practice this parameters will need to be changed in a system.

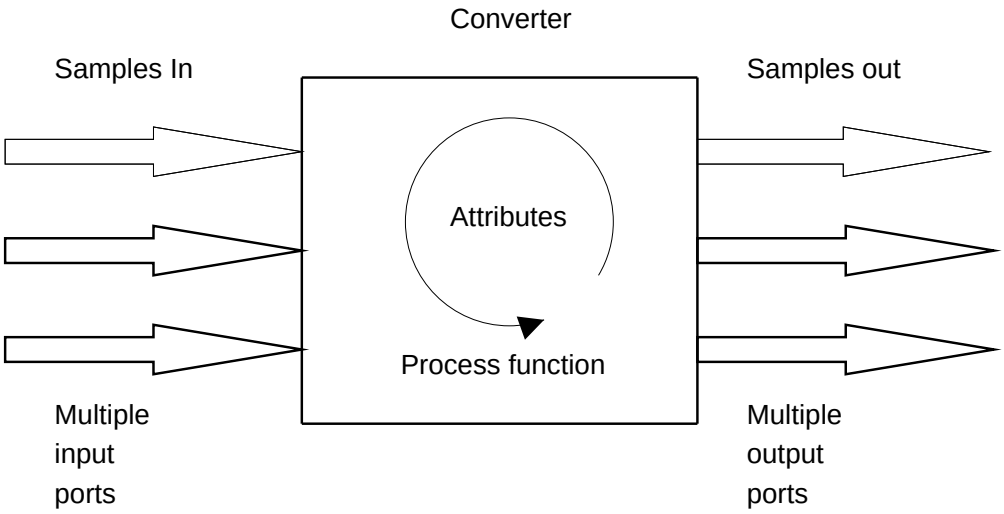


Figure 6.1: *A typical SDR converter component (adapted from [3]).*

The architecture allows the user to modify the values of the attributes of a converter [3].

6.1.3 Component methods

SU SDR components are designed in such a way that they are reusable. Three functions are used in each component to receive, process and dispatch samples between components. These functions are:

- A "read" method that acquires samples from the compo-nents input buffer.
- A "virtual process" function that specifies what process gets performed.
- A "write" method that dispatches samples to the input buffer of the next component.

6.1.4 System topology and linking of components

The way in which different modeled components transfer data samples amongst each other depends on the topology used. Since data is streamed on a sample-by-sample basis in the SDR architecture, a control-based topology is employed. In SDR, each component has its own input buffers and it is the responsibility of a director called a subcontroller to monitor the status of each buffer and to schedule processing. The scheduler activates process routines of each component in a round-robin schedule, allowing each component to fully deplete its input buffer [3]. The SDR components are linked by their input and output ports.

6.2 Test Class

The test class was developed in order to verify the performance of the SDR modulator and demodulator. A randomly generated binary bits were used to modulate a carrier frequency

and then the modulated signal was then demodulated using the demodulator developed in C++.

6.2.1 GMSK modulator and demodulator implementation in C++

The GMSK modulator and demodulator were implemented in C++ on the University of Stellenbosch SDR architecture. The bitgenerator component developed was used to generate a random zeros and ones. The randomly generated binary numbers were encoded using the NRZI encoder component. After the encoding process, the encoded data was re-sampled using the resampler component that regenerate each bit the required number of times. The re-sampled data was then fed to the Gaussian filter component. The Gaussian filtered data was then used to frequency modulate the carrier. A high level implementation of the modulator and demodulator is shown in Figure 6.2. The GMSK modulated signal was

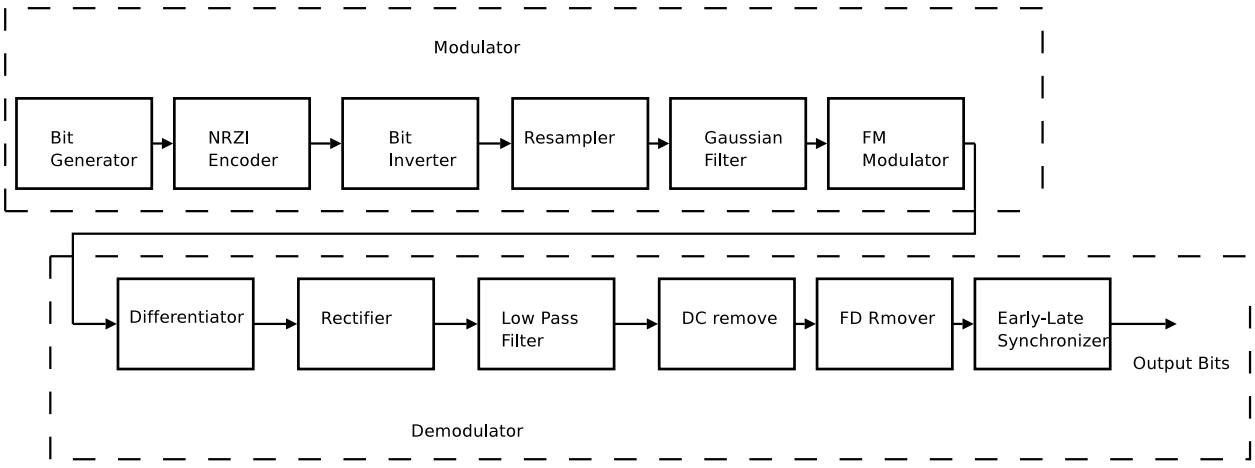


Figure 6.2: *A block diagram of the Modulator and Demodulator test class.*

demodulated using the discriminator (which consists of the differentiator, the rectifier, lowpas filter, DC component extractor and the frequency deviation remover). Figure 6.3 shows the Gaussian filtered signal using a BT value of 0.5. Figure 6.4 shows the differentiated GMSK signal. Figure 6.5 shows the rectified signal and Figure 6.6 shows the rectified signal after low pass filtering. A fourth order Butterworth low pass filter with a cut-off frequency of 5 kHz was used to filter the rectified signal. The filter was implemented as discussed in Section 4.3. The coefficients of the 4th order Butterworth low-pass filter were obtained in Matlab using this syntax:

```
[B,A] = butter(4,1/(0.8*Ts*fs))
```

The numerator polynomial B and the denominator polynomial A were found to be:

```
B = 1.0e-05 * [0.0341  0.1362  0.2044  0.1362  0.0341]
A = [1.0000  -3.8717  5.6234  -3.6312  0.8796];
```

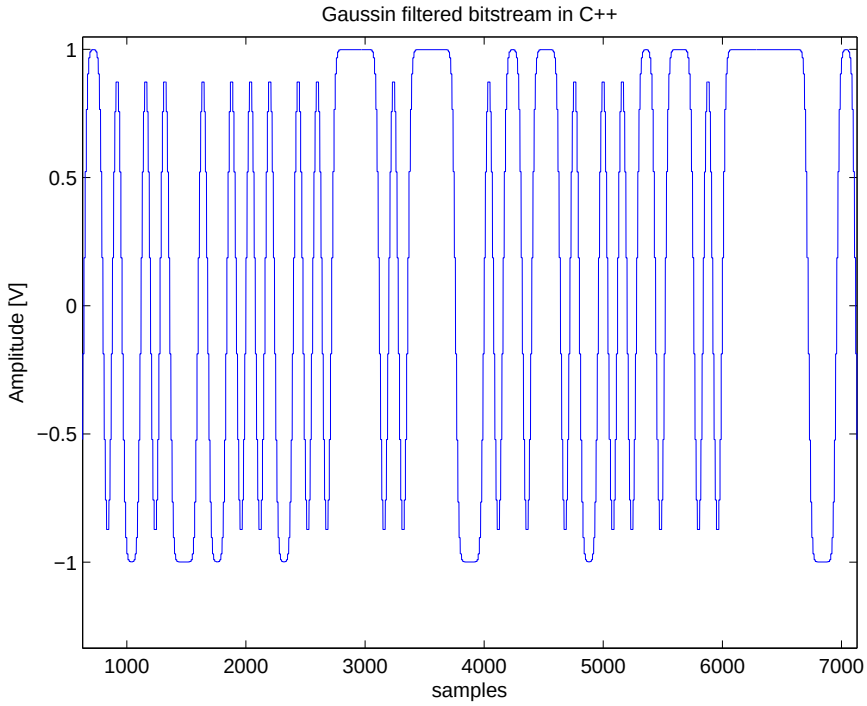


Figure 6.3: *Re-sampled Gaussian-filtered data in C++ using BT = 0.5.*

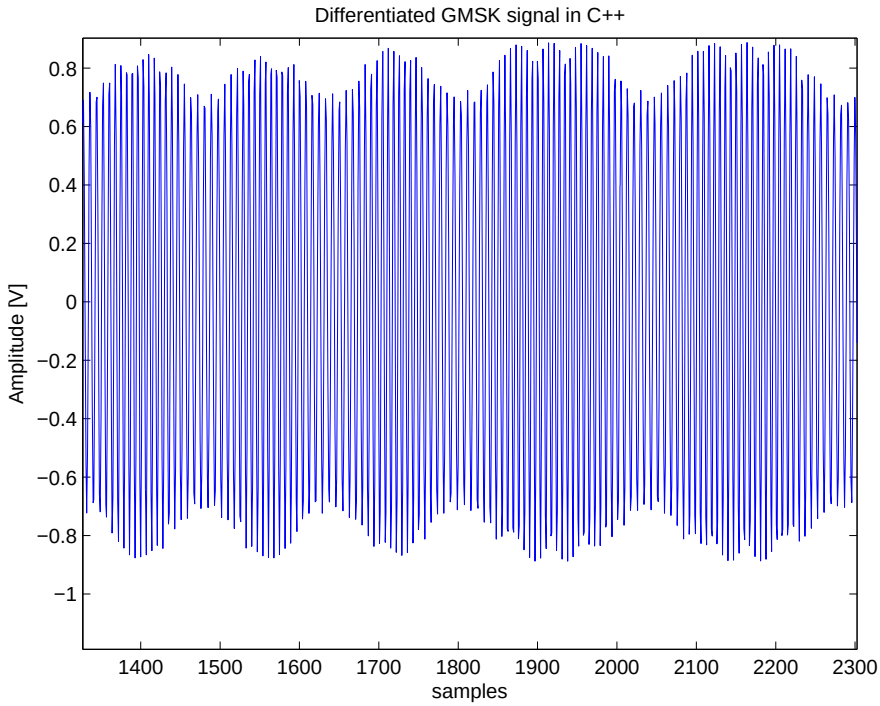


Figure 6.4: *The differentiated GMSK signal in C++.*

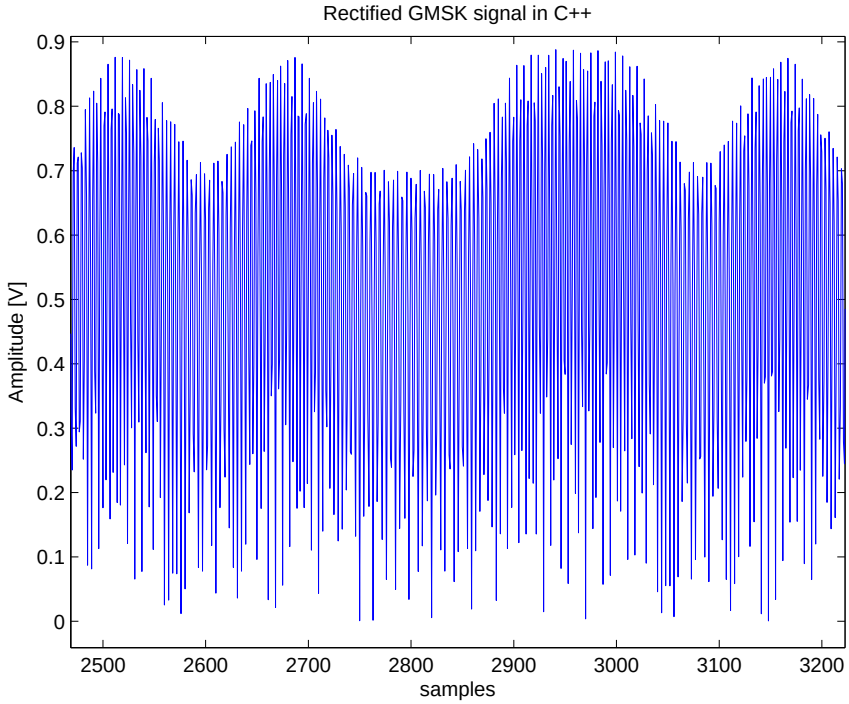


Figure 6.5: *The rectified GMSK signal in C++.*

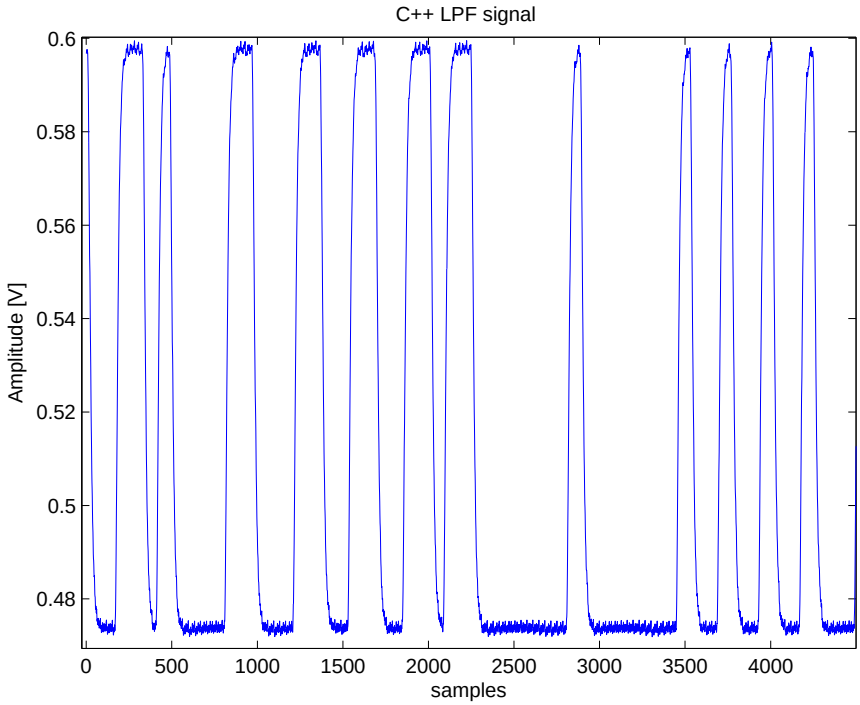


Figure 6.6: *The rectified GMSK signal after low pass filtering in C++. The modulating signal has been recovered.*

The roots of the B polynomial were obtained using Matlab `roots` function and the complex conjugate pairs were grouped together as shown below:

```
Z = roots(B)
= [-1.1885 + 0.2394i  -1.1885 - 0.2394i]
and [-0.8086 + 0.1629i  -0.8086 - 0.1629i]
```

Then the Matlab `poly` function was used to obtain the new polynomials like this.

```
Z1 = poly([-1.1885 + 0.2394i
-1.1885 - 0.2394i])
Z2 = poly([-0.8086 + 0.1629i
-0.8086 - 0.1629i])
```

where $Z1$ and $Z2$ are the new polynomials found from grouping the complex conjugate poles. The new coefficients of the new polynomials $Z1$ and $Z2$ were found to be

```
Z1 = [1.0000  2.3769  1.4698 ]
Z2 = [1.0000  1.6172  0.6804 ]
```

The above coefficients of the polynomials were used as attributes in the direct form IIR filter model developed in C++. The following piece of syntax shows how the second order filter subsystem is implemented in C++:

```
double w0 = -a1 * w1 - a2 * w2 + read_input_port(port_in)->read_double();
write_output_port(port_out, (b0*w0 + b1*w1 + b2*w2)*gain);
w2 = w1;
w1 = w0;
```

where $b0$ in the formula is 1.0000, $b1$ is 2.3769 and $b2$ is 1.4698. The same procedure was followed for the denominator polynomial A in order to obtain $a1$ and $a2$. The two second order filters were connected together to give a 4th order low-pass filter. Figure 6.7 shows the recovered GMSK baseband signal.

6.3 Actual AIS demodulation and message recovery

A high level implementation of the AIS system is shown in Figure 6.8. Actual AIS signals were transmitted using the Marine Data Solutions AIS. A detailed explanation of the AIS receiver system is demonstrated in 7 parts: the symbol synchronisation, decoding, start and stop flag detection, bit de-stuffing, byte flipping, CRC checksum calculation and message translation. These processes were implemented by linking each component in Figure 6.8.

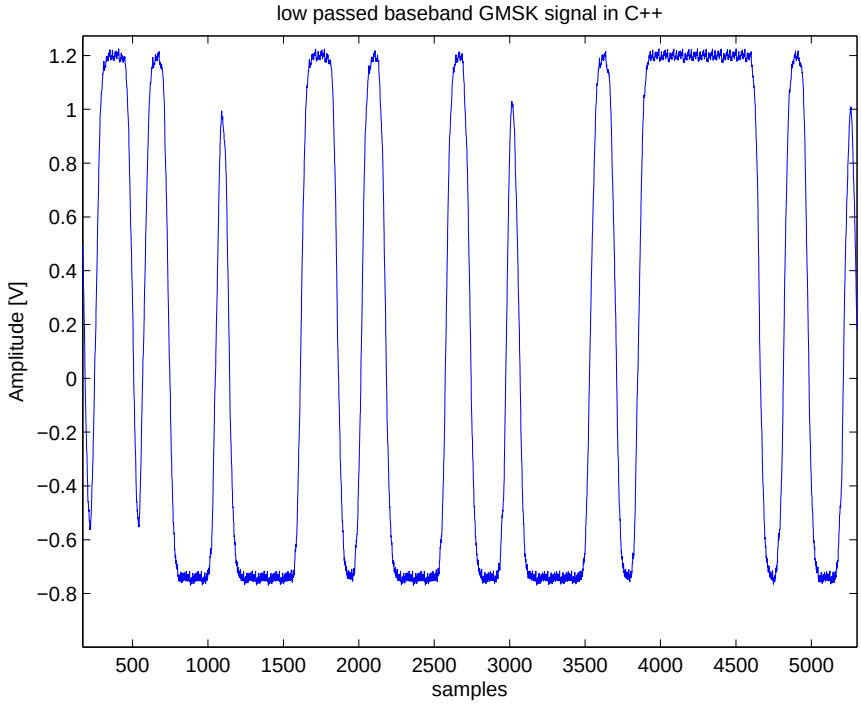


Figure 6.7: *The demodulated GMSK signal after DC offset is removed in C++.*

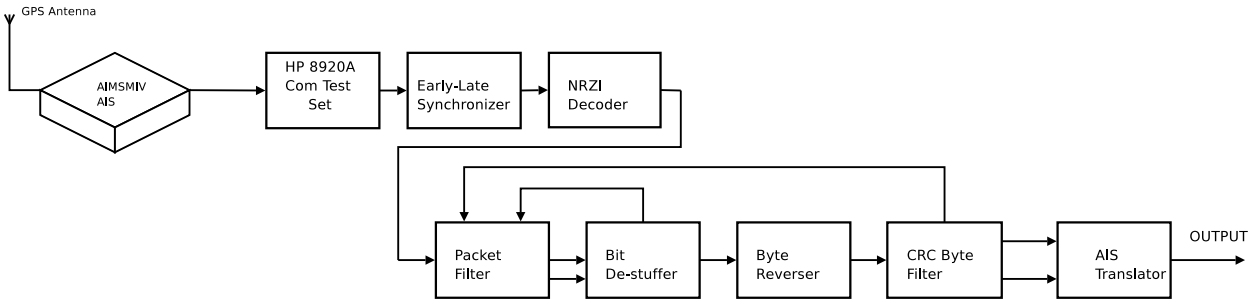


Figure 6.8: *Diagram of components implemented for the AIS system.*

In the SDR architecture, a component is instantiated from each of the converter subclasses by the subcontroller. An example of how a NRZI decoder component is instantiated is shown below:

```
Long decoder = sc.add_component(sd_basic_NRZI_decoder, );
```

Usually components' attributes are initialised to a certain value. But it is always desirable to change an attribute of a component when the the application of the component is changed. To change an attribute of each component to a new value is accomplished by using change_component_attribute function. E.g. changing the carrier frequency of a modulator component called `modd` is accomplished the following instruction:

```
sc.change_component_attribute(modd, frequency, 0.0163);
```

where "`sc`" is the sub-controller object and the value 0.0163 is a normalised frequency, normalised to the sampling frequency.

Components of the system are linked by their output and input ports. An example of linking the modulator component `modd` to the differentiator component called `diff` is done using the instruction:

```
sc.link_components(modd ,0, diff ,0);
```

where the second and fourth parameters indicate the port number. zero indicate the first input or output port and one indicate the second input or output port. To activate the process functions of each component so that the system should start running, the `sc.run()` function is used.

6.4 Analogue to digital conversion of the AIS signals

An analog signal must be sampled first before it is analysed and processed in the digital domain. An analogue-to-digital converter (ADC) is used to convert an analogue signal (which is continuous in both time and value), into a digital signal, which is discrete in both time and value. In order to translate signals from analogue to digital form, an ADC performs two fundamental steps, namely, sampling and quantisation [25]

If $x(t)$ is a continuous time signal, a discrete time (sampled) signal of $x(t)$ can be represented as the original signal $x(t)$, multiplied by a periodic impulse train $p(t)$ [25]. The pulses are spaced by a period T_s, which is equal to $1/F_s$ (where F_s is the sampling frequency).

$$x_{sampled}(t) = x(t)p(t) = \sum_{n=-\infty}^{\infty} x(t)\delta(t - nT_s) \tag{6.1}$$

The frequency spectrum of the sampled signal consist of the scaled replicas of the continuous time signal spaced at integral multiples of the sampling frequency and centred at multiples of F_s. This is represented by [25]:

$$X_{sampled}(f) = F_s \sum_{n=-\infty}^{\infty} X(f - kF_s) \tag{6.2}$$

When sampling a band limited signal, the Nyquist criterion must be carefully followed in order to avoid aliasing. The signal needs to be sampled at Nyquist rate (twice the highest bandwidth of the signal) or higher [25]. The Nyquist criterion allows an analogue signal to be reconstructed completely from a set of uniformly spaced discrete-time samples. If the Nyquist criteria is not obeyed, aliasing (if the sampling signal's bandwidth is greater than half the sampling frequency, the sampling process will result in the image spectrums overlapping) is the result. [25] says:

> Quantisation is a process of mapping a continuous valued signal onto a discrete set of levels

Quantisation process in data converters is characterised by the number of available quantisation levels, the range of quantisable input voltages and the width of a quantisation level or step size [25]. The number of quantisation levels is determined by the number of bits used. An ADC can be described as an anti-aliasing filter, followed by a sample and hold circuit which is followed by a quantiser circuit [25].

6.4.1 DAQ-2010 ADC

The DAQ-2010 used provides 4 different analogue input channels. Only one channel was used in this experiment. The ADC acquisition is initiated by a trigger source. The data acquisition begins when a trigger condition is matched. After the analogue-to-digital conversion process, the ADC data is buffered in a data first in first out (FIFO) buffer. After that the buffered data is transferred into the computer's memory for further processing. The Data acquisition card was used to sample the demodulated signal from the Communication test set using one channel.

Chapter 7

Practical results

7.1 Introduction

Chapter 7 contains the experimental results of the AIS receiver developed in Chapters 4, 5 and 6. Actual AIS signals were transmitted using Marine Data Solutions AIS. The experiment was performed by transmitting actual AIS signals and then using the Hewlett Packard (HP) 8920A RF Communications Test Set as VHF frequency demodulator. The HP Test Set was used because the high frequency used by AIS (162.025 and 161.975 MHz) precludes the use of digitisation at the radio frequency. Alternatively, if we wanted to use the SDR discriminator developed to demodulate the AIS signal, we would have had to use external mixer in order to down-covert the RF signal to a reasonable intermediate frequency (a few kHz). The data acquisition card that we used is limited to a sampling rate of 2 MS/s. Two experiments were conducted. The first experiment involved processing the signals offline, and the second experiment involved real time system processing.

7.2 Experimental setup

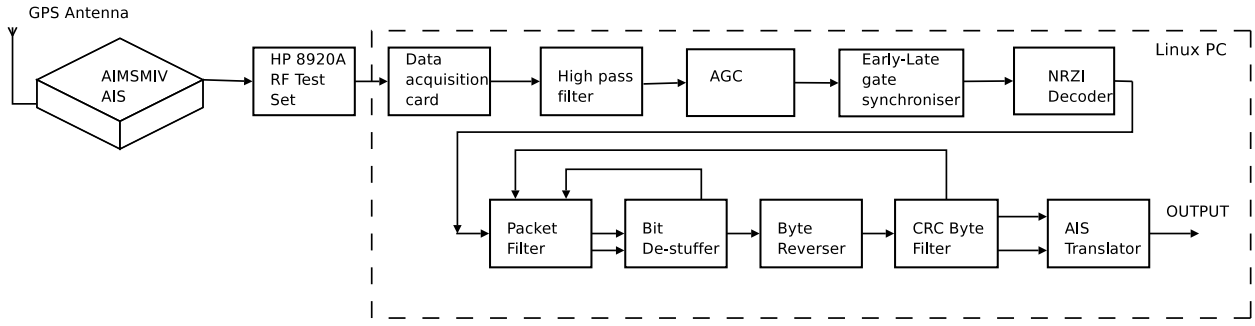


Figure 7.1: *Block diagram of the experimental setup conducted with actual AIS signals.*

Figure 7.1 shows a block diagram of the experimental setup. The system consists of three stages. The AIS transmitter, the HP 8920A radio frequency (RF) Communications test set and the Linux personal computer (PC). The digitisation is performed by the data

acquisition card in the Linux PC. All the stages from the automatic gain control (AGC) up to the AIS translator are performed in software.

7.3 Offline processing

The experiment was conducted as in Figure 7.2: The AIS transmitter uses a 24 V power

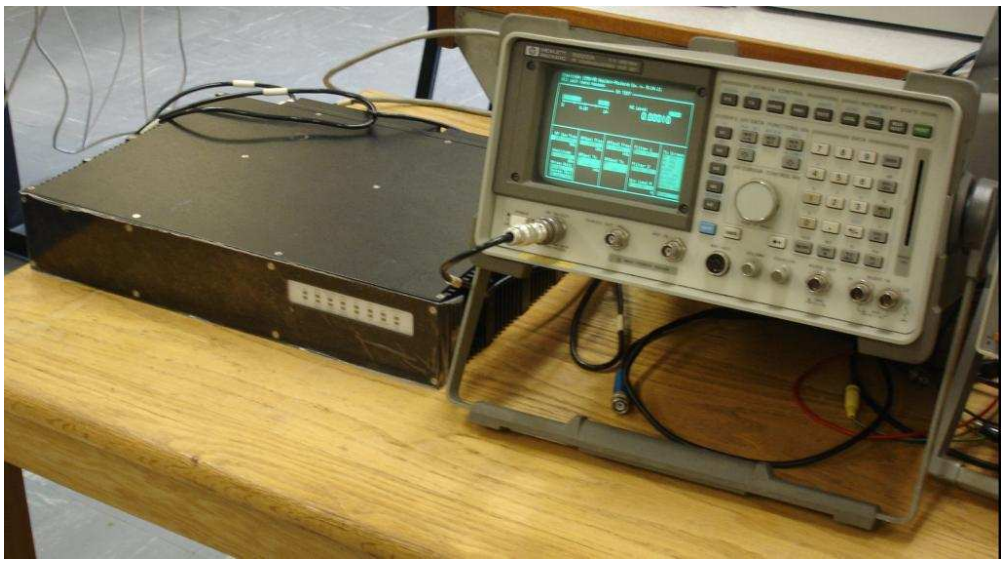


Figure 7.2: *The experimental setup conducted with actual AIS signals (showing the AIS transmitter and HP Communications Test Set).*

supply and draws a maximum of 6A current. A GPS antenna was connected on the GPS antenna connector port using a 50Ω TNT connector. A 50 Ω cable was then connected to the VHF antenna connector port and this cable was then connected to the RF input of the HP Communications test set as shown in Figure 7.2. The HP test set was set to the FM demodulating mode at the AIS centre frequency of 161.975 MHz. The output of the HP test set was connected to the input channel of the interface which connect to the data acquisition card. The data acquisition card used is the NuDAQ, DAQ-2010/2006, 4-channel, simultaneous, high performance multifunction card. The data acquisition card could then stream received samples into a file. A sampling rate of 153.6 kHz was used to sample the output of the HP communications test set. The setup with the interface to the data acquisition card is shown in Figure 7.3. The RS232 port of the AIS was connected to another computer to monitor the encapsulated sentences of the transmitted AIS messages. This information was displayed on the PC terminal for comparison with the demodulated and interpreted AIS messages from the developed AIS receiver. Because the reporting interval of the AIS is known, it was possible to use a timer to estimate the time for the next transmission. In such a way the data acquisition card was activated to begin recording at the estimated time of transmission. Several position report messages were captured including static and voyage related data message.

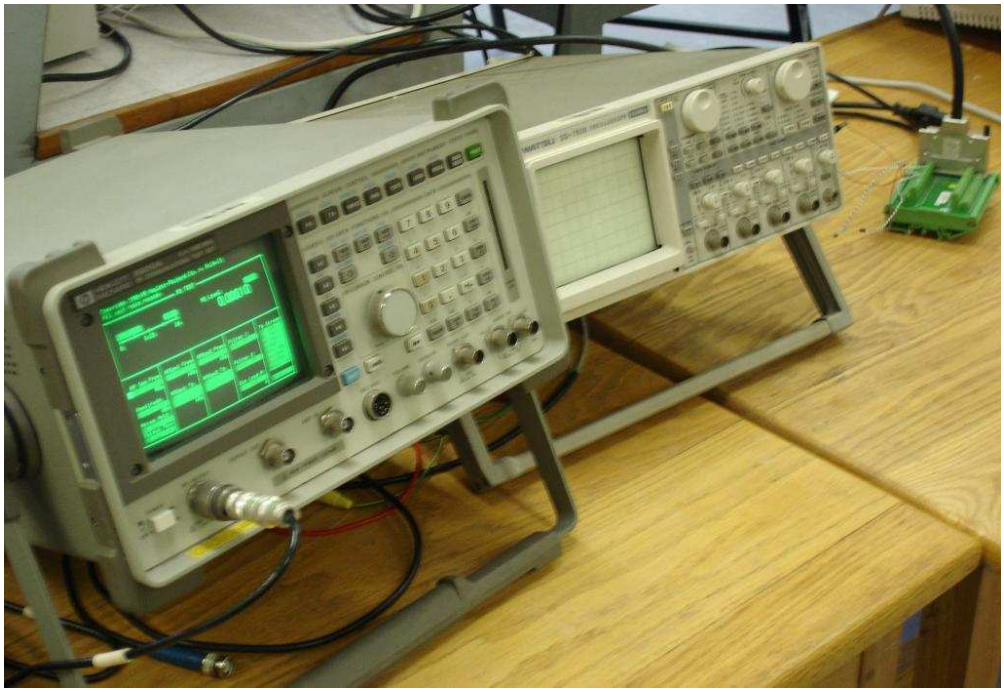


Figure 7.3: *The experimental setup indicating the position of the interface to the data acquisition card and the HP Communications Test Set.*

Figure 7.4 and Figure 7.5 show some of the demodulated AIS signals captured using the data acquisition card.

7.4 DC offset and noise

Analysis of the demodulated AIS baseband signals captured (shown in Figure 7.5) reveals the DC offset that needed to be removed before symbol synchronisation. The DC offset is a result of the difference between the AIS transmitter's centre frequency (161.975 Mhz) and the HP Communications Test Set's local oscillator (161.975 Mhz) frequency. This is because of the oscillators being made of components that have tolerance limits. A Butterworth high pass filter with a cut-off frequency of 150 Hz was used to remove the DC offset. The received AIS signal in Figure 7.5 is very weak. This is demonstrated by the signal's amplitude of 0.02V.

7.5 AIS receiver performance using Matlab model

The Matlab model of the AIS receiver was used first to synchronise and then extract the message bits from the rest of the data bits. First the DC offset was removed by subtracting the mean of the demodulated signal. A low pass filter was used to filter the noise and the signal was amplified to a maximum of unity amplitude using a linear amplifier. The

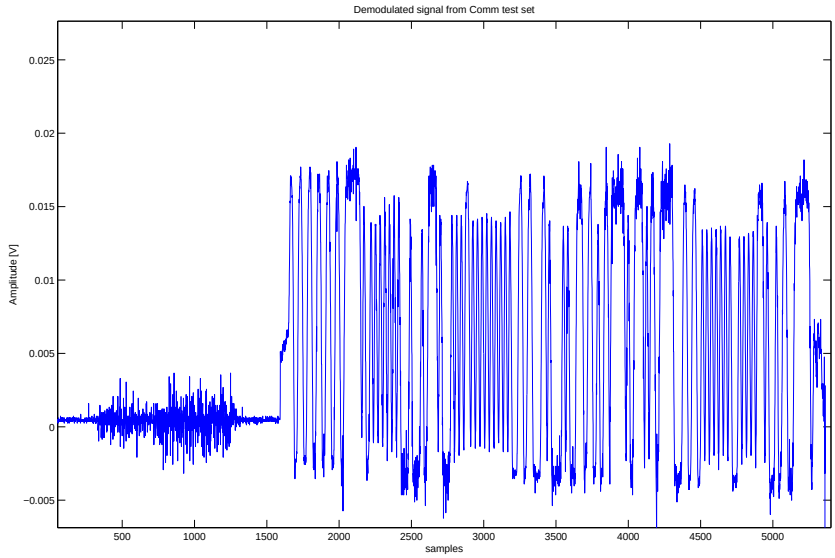


Figure 7.4: *AIS received message captured.*

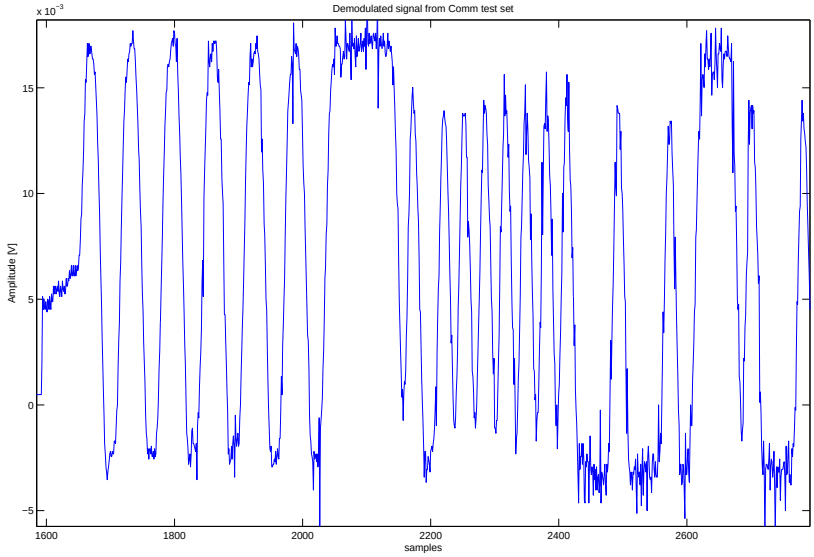


Figure 7.5: *AIS received message captured zoomed in.*

resulting signal was fed to the early-late gate symbol synchroniser and the resulting signal is shown in Figure 7.6. This experiment was performed on a series of messages captured. Note the synchroniser achieving synchronisation within the preamble or training sequence

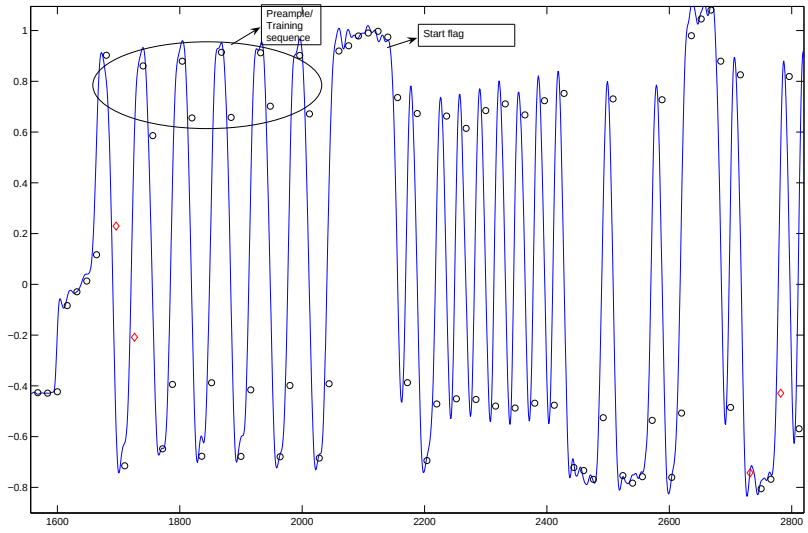


Figure 7.6: *The demodulated AIS signal after symbol synchronisation.*

in Figure 7.6. The resulting symbols from the synchroniser were decoded using the NRZI decoder developed. After decoding, the start and stop flags were extracted by the packet filter model and then bit-destuffing was applied to the data bits. The resulting data bits are passed through the byte flipper component before passing through the CRC byte filter model that performs the CRC checksum. The results obtained from the model component that finally translate the data bits into meaningful AIS message is shown in Table 7.1. The encapsulated string of the received message was found to be:

```
!AIVDM,1,1,,B,103>rCPP001FGCidUO;aD?vb0<00
```

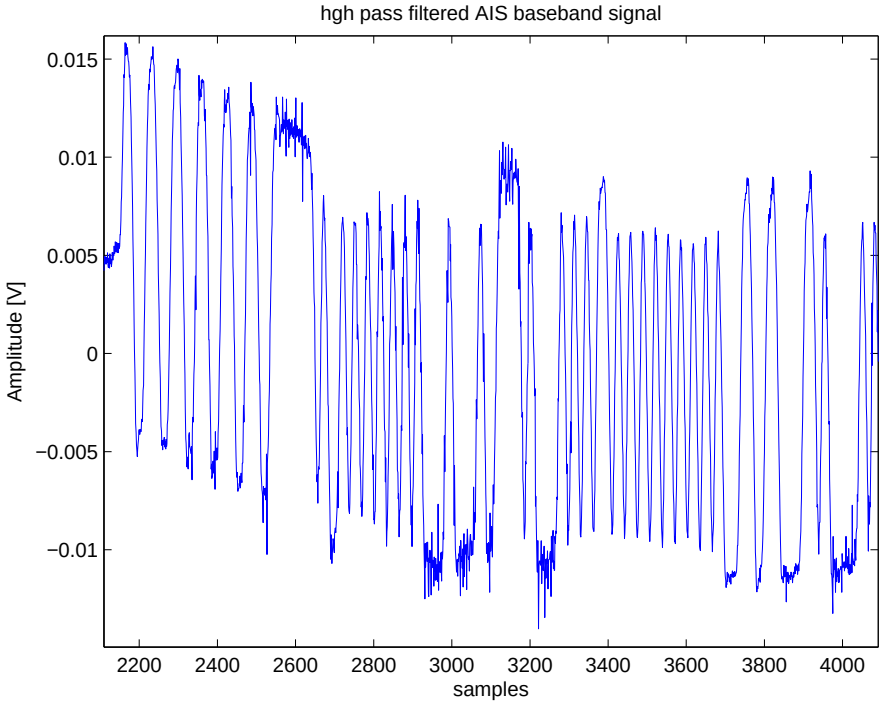
This message is a position report which is message ID number one “1” of an AIS. The checksum reveals that it is a valid AIS message.

7.5.1 Amplification using AGC

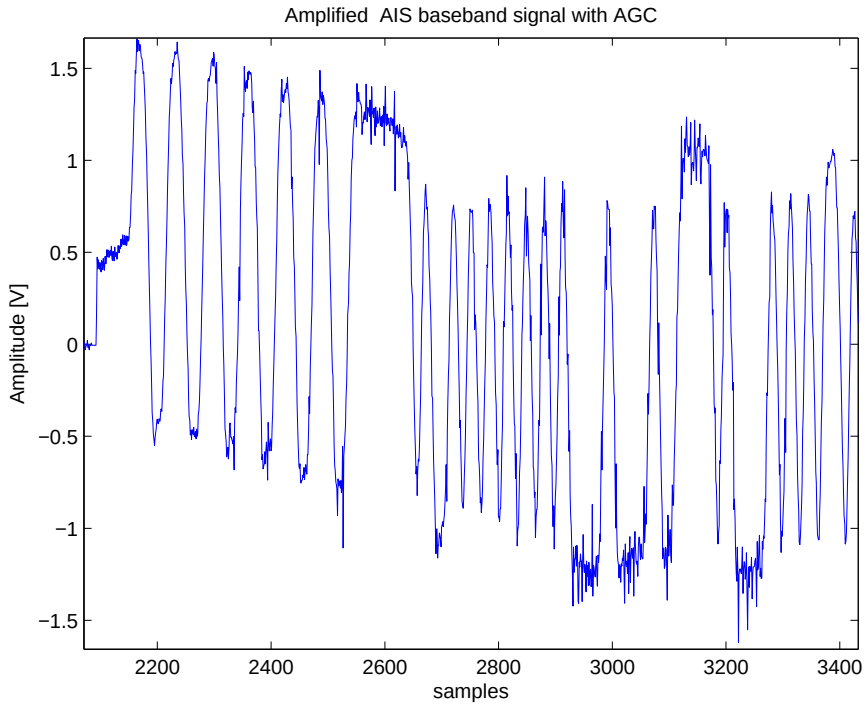
The received AIS signal in Figure 7.4 was amplified using the AGC model that was developed in Matlab. This GMSK baseband signal was highpass filtered first using a 4th order butterworth filter with a cut-off frequency of 150 Hz in order to remove the unwanted DC component. The resulting signal was then amplified using the AGC. This was done because in real time systems, the received signal strength can vary due to various factors such as; path loss and multipath effects. The high pass filtered signal and the amplified signal are shown in Figure 7.7. The resulting amplified signal was low pass filtered using a 4th order

Table 7.1: *Results from the AIS message with a position report*

Parameter	**Value**
Message ID	1
Repeat indicator	0
MMSI	3390030
Navigational status	0
Rate of turn	128 degrees/minute
SOG	0 knots
Position accuracy	0
Longitude	18 degrees 52 minutes East
Latitude	33 degrees 55.7 minutes South
COG	238.4 degrees
True heading	511 Degrees
UTC second when report generated	21 seconds past minute
Regional application	0
Spare	0
RAIM	0
Communication state	49152
CRC-CCITT checksum	Hex F0B8



(a)



(b)

Figure 7.7: *(a) The Output of the high pass filter. A varying DC component appears at the beginning due to the filter's impulse response (a transient effect).(b) The output of the AGC after AGC achieved stability*

butterworth filter with a cut-off frequency of 5 kHz to remove the noise component. This signal was synchronised and decoded using the Matlab receiver model and the output of the symbol synchroniser is shown in Figure 7.8. The performance of the AGC was evaluated

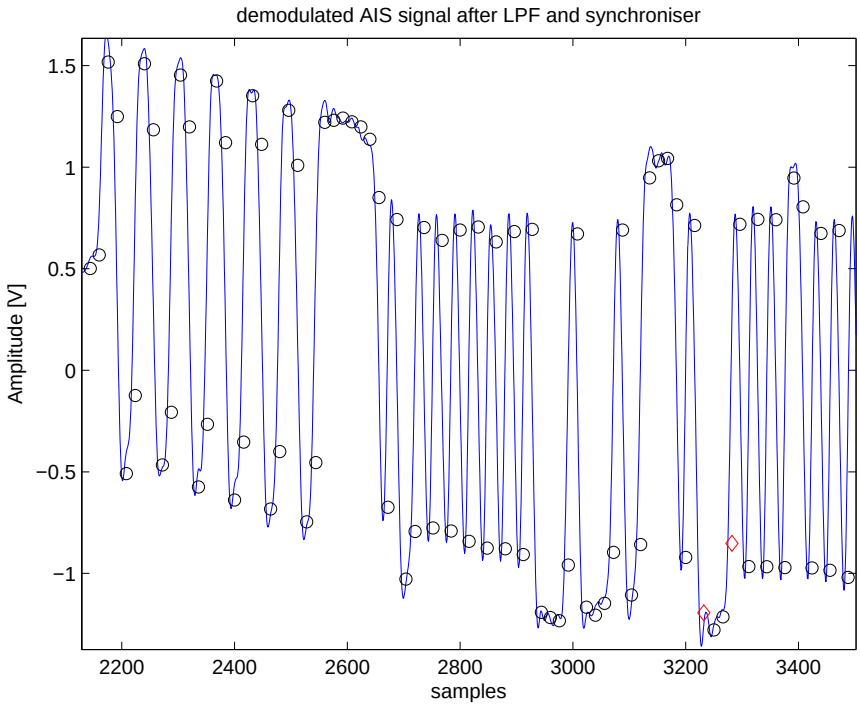


Figure 7.8: *The output of the early-late gate synchroniser*

under different conditions. One interesting occurrence is shown in Figure 7.9. If the information part of the signal arrives before the AGC has reached stability, the first part of the received information signal is amplified less than the latter part of the signal. This results in errors at the decision device unless the symbol synchroniser threshold is adjusted. But if the AGC received the noise before the information part of the signal appears, the AGC achieves stability and the signal is amplified as shown in Figure 7.7.

The AGC was modified in such a way that if the signal strength is high, the system should reduce the gain drastically and if the signal strength is very low, the system should increase the gain drastically. With this modifications, errors at the decision device were eliminated. The same signal that was amplified in Fig 7.9 was amplified using the modified AGC and the results are shown in Figure 7.10. This signal was decoded and symbol synchronised correctly and gave the correct checksum. The translated sentence is:

```
!AIVDM,1,1,,B, 103>rCPP011FGCGdUONt?OwDODOO
```

7.5.2 Static and voyage related data message

One of the messages captured by the data acquisition card was a type of message five which communicate static and voyage related data. This message is shown in Figure 7.11. Note

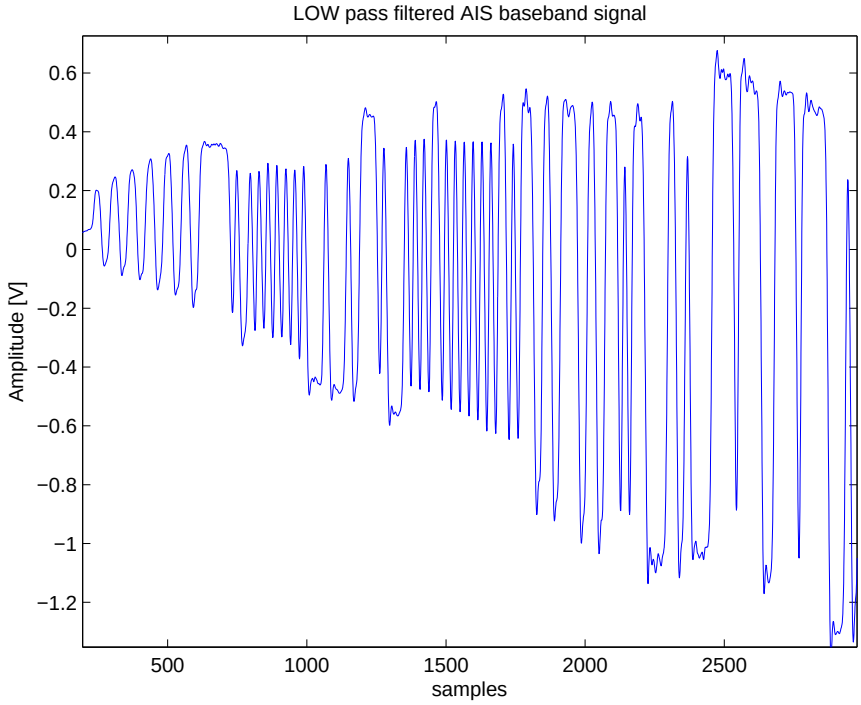


Figure 7.9: *Output of the AGC before AGC is stabilised*

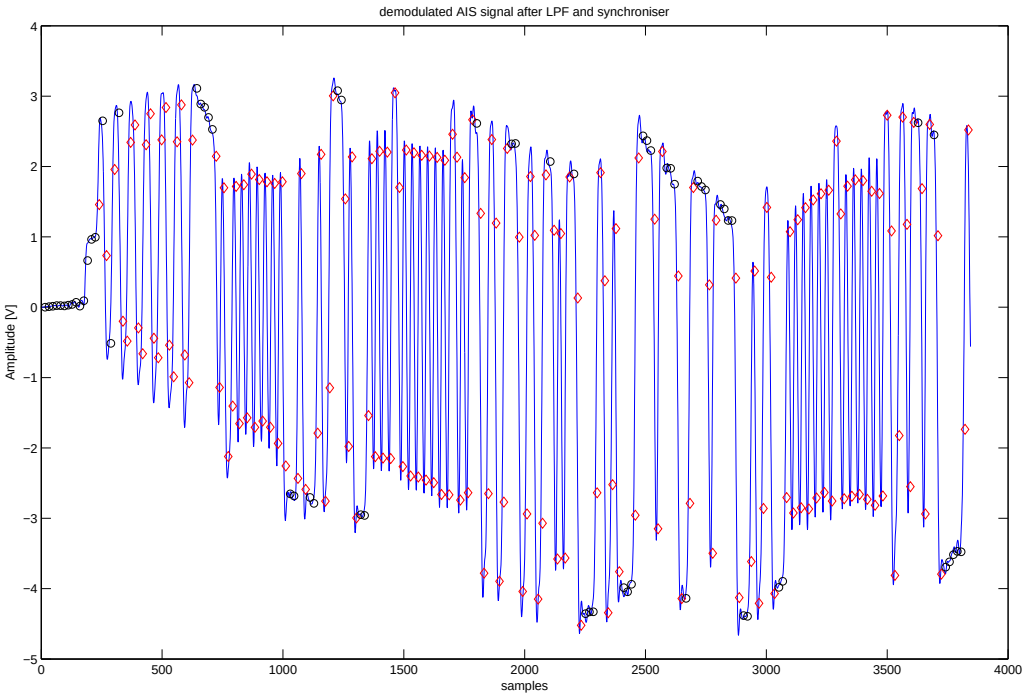


Figure 7.10: *Output of the AGC after modifications.*

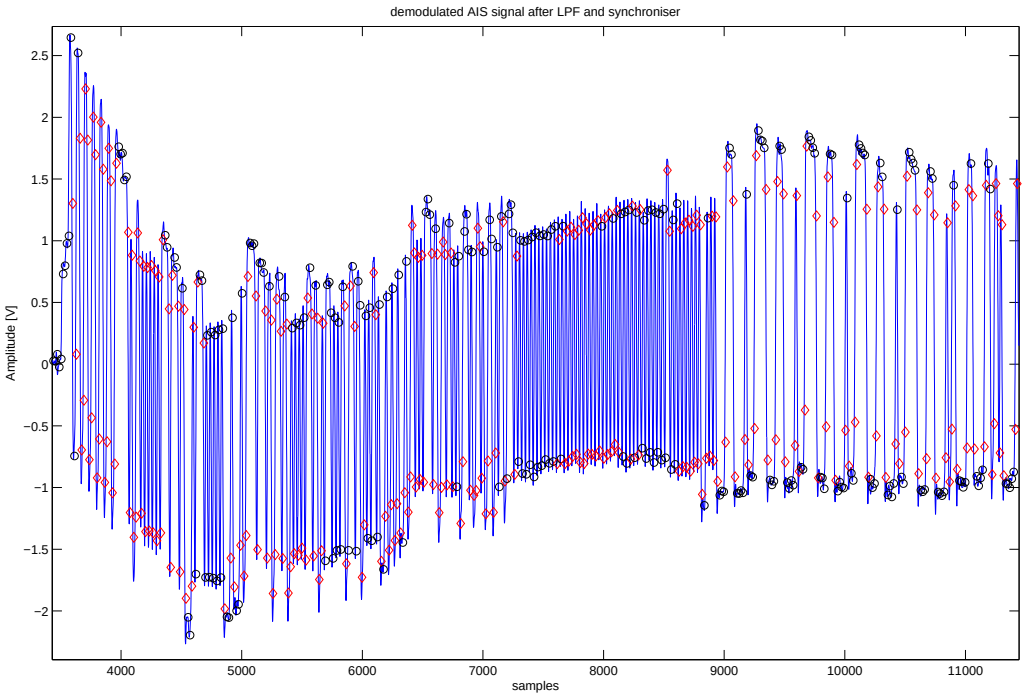


Figure 7.11: *Plot of static and voyage related AIS baseband signal after symbol synchronisation.*

that this message is long compared to the previous message signals received. The diamonds demonstrate that the synchroniser has not yet achieved lock. As soon as it synchronises, circles are shown in the plot. This signal was decoded and translated by the Matlab model to be a message ID 5. Its encapsulated string was constructed to be:

```
!AIVDM,1,1,7,B,503>rCP0kfTq=@DhiD1=@Dhj1DpUJ1@E=@
0000000000040000?ggggggggggggggggggg
```

The results of the translation of this message are shown in Table 7.2. The results shows that this is a valid AIS message number five “5”. The checksum confirms that it is a valid AIS message and that there are no errors in the data.

7.6 SDR AIS receiver performance

The same demodulated AIS baseband signals from the HP Communications Test Set were applied to the C++ AIS receiver. It should be noted that the C++ receiver is designed to function in real time. In real time it will not be viable to calculate the mean of a signal, so a high pass filter was used to remove the DC offset before symbol synchronisation. The high pass filter was implemented using the cascades of the second order structure that was developed. The high pass filtered signal was amplified using the AGC system developed in

Table 7.2: *Results from the AIS message with a static and voyage related data*

Parameter	**Value**
Message ID	5
Repeat indicator	0
MMSI	3390030
AIS version number	0
IMO number	3390030
Call sign	CD5<<E0
Name of ship	CD5<<PE>9FPD5CD00000
Type of ship and cargo	0
Dimension/Reference for position	0
Electronic Position Fixing Device	1
Estimated Time of Arrival	0
Maximum Present Static Draught	0
Destination	vvvvvvvvvvvvvvvvvvv
Data Terminal Ready	1
Spare	0
CRC-CCITT checksum	Hex F0B8

C++. The resulting signal was low pass filtered before being synchronised, decoded and then translated to meaningful information. The demodulated, C++ high pass filtered, amplified and low pass filtered signals are shown in Figure 7.12. The results in (b) show that the high

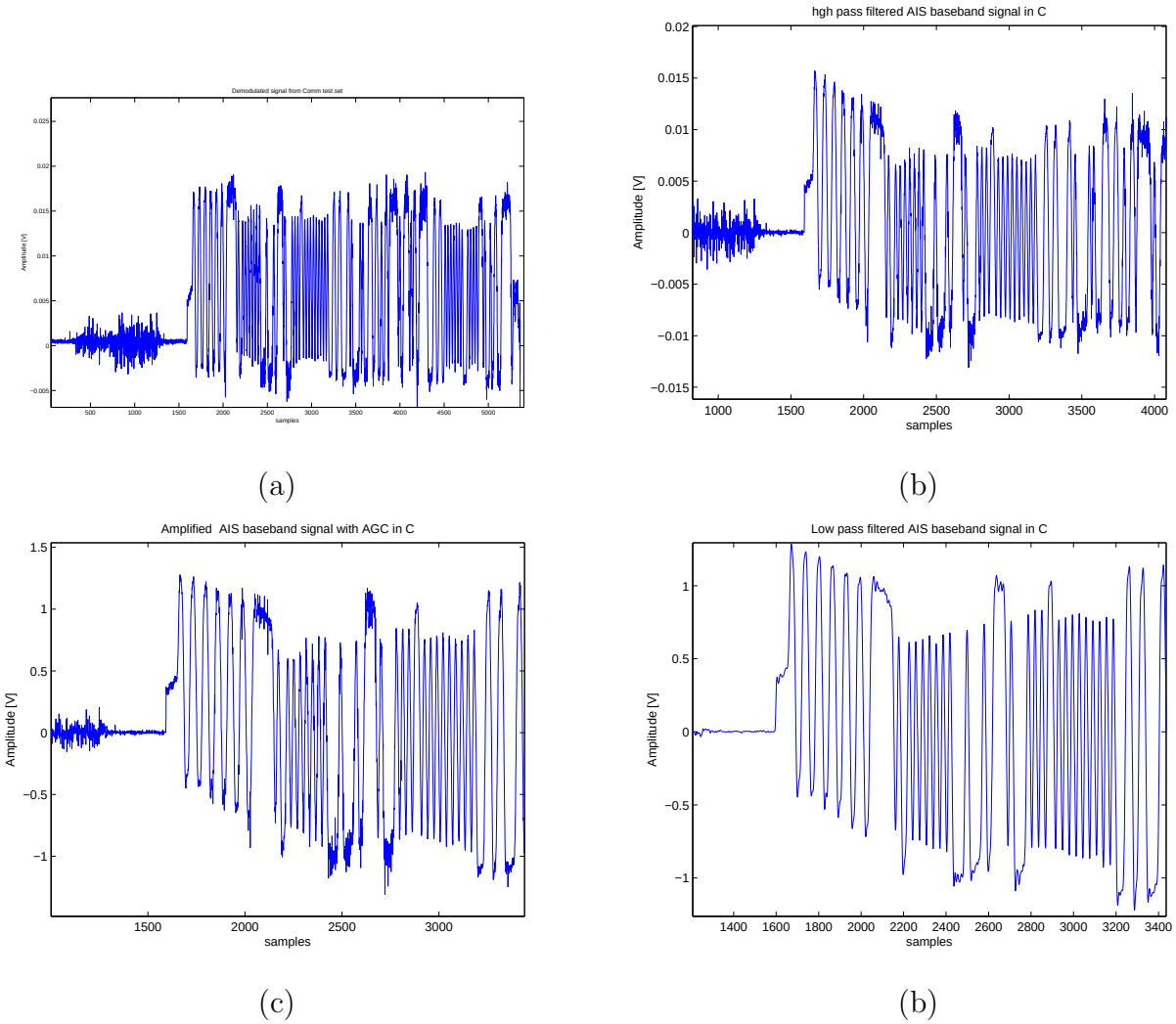


Figure 7.12: *The SDR receiver performance: (a) is the demodulated baseband signal from Comm test set; (b) is the high pass filtered signal; (c) is the amplified signal using AGC, and (d) is the resulting signal after low pass filtering.*

pass filter filtered out the unwanted DC component. A varying low frequency component appears at the start of recording. This is because of the impulse response of the high pass filter and is a transient effect that dies out quickly. A cut off frequency of 150 Hz was found to give the best performance.

Tests were performed on a series of AIS messages and the results of one of the messages captured are shown in Table 7.3. The encapsulated string of the received message was found to be:

```
103>rCPP001FGCidUO;aD?vb0<00
```

Table 7.3: *Results from the AIS message with a position report using C++ receiver*

Parameter	**Value**
Message ID	1
Repeat indicator	0
MMSI	3390030
Navigational status	0
Rate of turn	128 degrees/minute
SOG	0 knots
Position accuracy	0
Longitude	18 degrees 52 minutes East
Latitude	33 degrees 55.7 minutes South
COG	238.4 degrees
True heading	511degrees
UTC second when report generated	21 seconds past minute
Regional application	0
Spare	0
RAIM	0
Communication state	49152
CRC-CCITT checksum	Hex F0B8

Which is the same position report message obtained using the Matlab model. The longitude and latitude co-ordinates are correct compared to the known longitude and latitude co-ordinates of Stellenbosch which are:

1. Longitude = $18^\circ\ 51'\ 0E$
2. Latitude = $33^\circ\ 55'\ 60S$

The static and voyage related message number five "5" was synchronised, decoded and translated by the C++ receiver and the results gave the encapsulated string of the message as:

```
503>rCP0kfTq=@DhiD1=@Dhj1DpUJ1@E=@0000000000040000?
gggggggggggggggggggg
```

The encapsulated string contains the valid AIS 6 bit characters. The translated message is shown in Table 7.4. It must be remembered that in real time systems operation, messages

Table 7.4: *Results from the AIS message with a static and voyage related data using C++ receiver*

Parameter	**Value**
Message ID	5
Repeat indicator	0
MMSI	3390030
AIS version number	0
IMO number	3390030
Call sign	CD5<<E0
Name of ship	CD5<<PE>9FPD5CD00000
Type of ship and cargo	0
Dimension/Reference for position	0
Electronic Position Fixing Device	1
Estimated Time of Arrival	0
Maximum Present Static Draught	0
Destination	vvvvvvvvvvvvvvvvvvv
Data Terminal Ready	1
Spare	0
CRC-CCITT checksum	61624

will be received continuously one after another. The ultimate optimum performance of the C++ receiver is in its ability to process one message after another and output the results if the message passed the CRC check test. In real time, errors do occur and the receiver must

be able to detect errors in a corrupt message, discard that message, and then go ahead and process the next message. The C++ receiver was fed a series of messages that followed one another. One such message which was captured from the AIS is shown in Figure 7.13 and Figure 7.14. In Figure 7.14 we see that the second message has been amplified more than

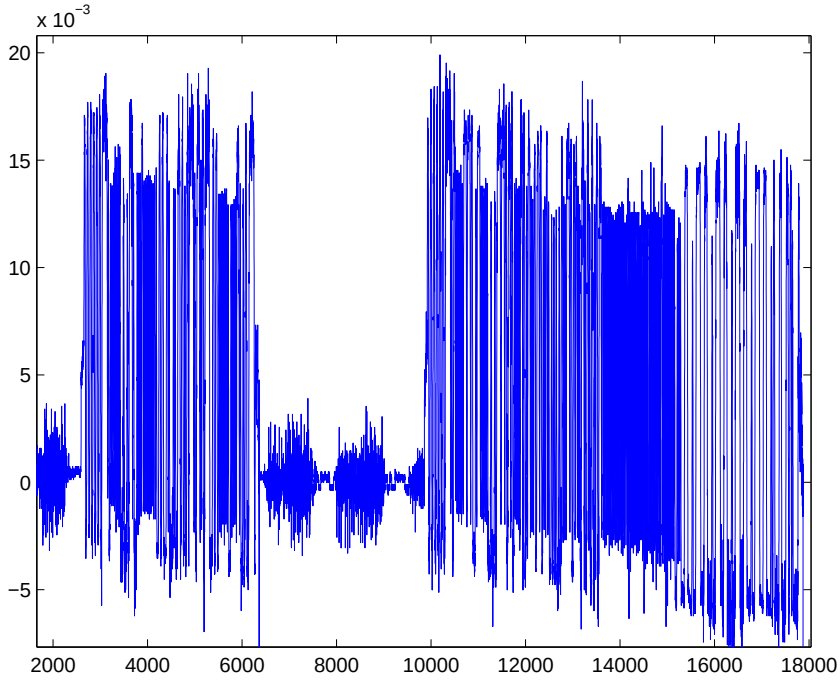


Figure 7.13: *Demodulated AIS messages arriving in succession*

the first message. This is because after amplifying the first message signal, the AGC had to amplify only the noise component, which is very low compared to the information signal. So by the time the second message signal arrives, the gain of the system was quite high and the system had to start reducing the gain. This problem can be solved by implementing a `squelch`, where the AGC has a threshold and it only amplify anything above the threshold. This will allow the AGC not to amplify the noise component. The receiver translated the two messages correctly consecutively. Other messages which were demodulated and translated are:

```
!AIVDM,1,1,,B, 103>rCPP011FGBodUOaJ3wv605SD
!AIVDM,1,1,,B, 103>rCPP011FGCSdUO6rm?vf00Rs
!AIVDM,1,1,,B, 103>rCPP021FGC;dUOCargwB0@?W
!AIVDM,1,1,,B, 103>rCPP041FGC=dUO?:?Ovf0D00
!AIVDM,1,1,,B, 103>rCPP001FGBidUO2HcOv60D00
!AIVDM,1,1,,B, 103>rCPP011FGCKdUOGaKwwD0H?i
!AIVDM,1,1,,B, 103>rCPP021FGCmdUOLr7?wD0<00
```

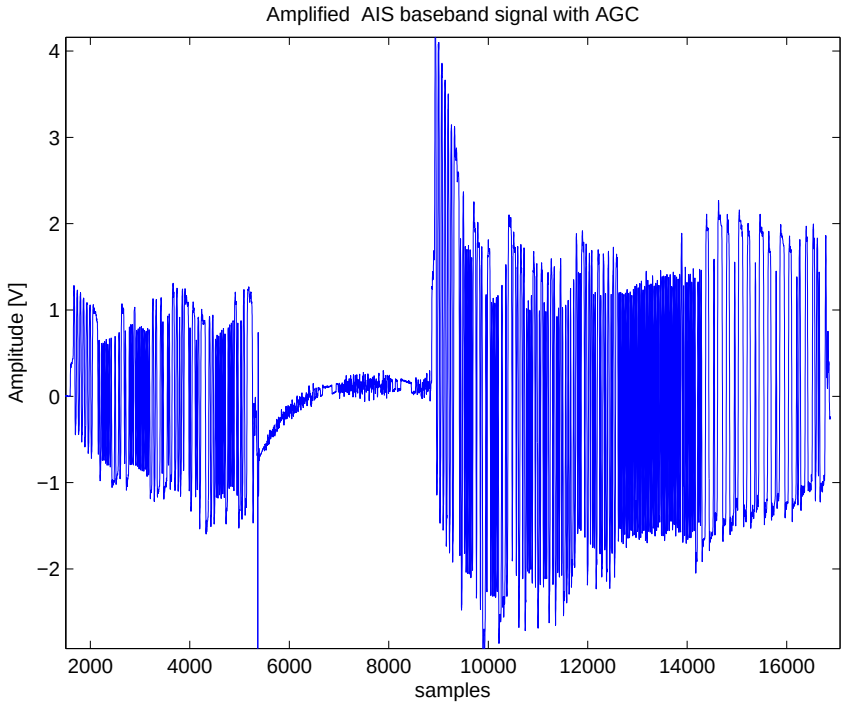


Figure 7.14: *Effects of HP and the AGC on messages arriving in succession*

7.6.1 Practical bit error probability

Figure 7.15 shows the error probability performance of the SDR receiver in the presence of AWGN using BT = 0.5. The GMSK signal which was corrupted with noise was bandpass filtered in order to filter out the out-of-band noise components before demodulation. The first cut-off frequency of the bandpass filter was half the carrier frequency (which is equal to 50 kHz, with a carrier frequency of 100 kHz) and the second cut-off frequency was 150 kHz. As seen from this curve in Figure 7.15 and Figure 5.10, the bit error performance of the SDR receiver is about 4.0 dB poorer in the presence of AWGN than the theoretical one, and about 1.5 dB poorer than the simulated one. This is because of the low-order lowpass filter used. A low-order filter does not give a sharp cut-off frequency as compared to a higher-order filter which gives a sharp cutoff frequency. The low-order low-pass filter allows some of the out-of-band noise components to pass through, resulting in poor bit error probability than the theoretical one.

7.7 Real time system performance of the SDR AIS receiver

The SDR receiver was implemented in real time. The AIS transmitted AIS signals in real time and the signals were demodulated using the HP communication test set. The data

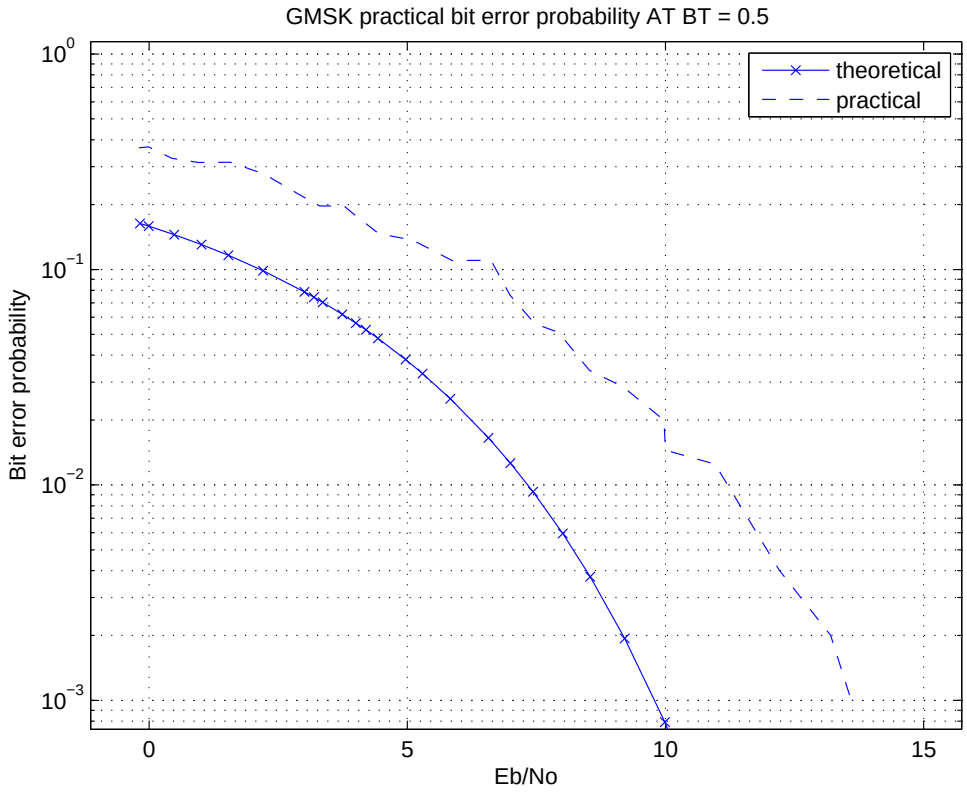


Figure 7.15: *Practical bit error probability curve for GMSK*

acquisition card was used to digitise the demodulated baseband signals in real time. The SDR receiver was linked to the data acquisition card component called DAQ. The whole SDR receiver system is stored in the *Tx_test.cpp* file. The *Tx_test.cpp* file was activated to start processing.

The sampling rate on the data acquisition card was reduced to 76.8 kbps because of the speed of processing on the computer. The system output several position report messages in real time. Some of the messages which where translated are shown in Table 7.5 and Table 7.6. The results shows the correct MMSI number of 3390030. This MMSI number correspond to the MMSI number that was obtained from the computer that was connected to the RS232 port of the AIS. The messages show the correct CRC-CCITT checksum.

Table 7.5: *Realtime SDR receiver results*

Parameter	**Value**	**Value**
Message ID	1	1
Repeat indicator	0	0
MMSI	3390030	3390030
Navigational status	0	0
Rate of turn	128 degrees/minute	128 degrees/minute
SOG	2 knots	102.3 knots
Position accuracy	0	0
Longitude	18 degrees 51 minutes	181 degrees 0 minutes
Latitude	33 degrees 55 minutes	91 degrees 0 minutes
COG	238.4 degrees	360 degrees
True heading	167 degrees	511 degrees
UTC second when report generated	23 seconds past minute	28 seconds past minute
Regional application	0	0
Spare	0	0
RAIM	0	0
Communication state	49152	395428
CRC-CCITT checksum	61624	61624

The messages in Table 7.5 and Table 7.6 are position reports, which are messages 1, 2 or 3 of an AIS station. From Table A.1 in Section 2.6.3 it was stated that the value of 181° for latitude means the latitude is not available. We can see from Table 7.5 that the latitude value is available on the first message. The second message shows the latitude value of 181°, which means that the latitude was not available. This happened because the AIS was transmitting before the position fix was obtained from the GPS antenna. When this is the case, the position fix on the AIS instrument shows a red light. As soon as the position fix is obtained, the light emitting diode (LED) changes to green.

Table 7.6: *Output of the SDR receiver in real time*

Encapsulated strings
!AIVDM,1,1,,A,1 0 3 > r C P P ? w < t S F 0 l 4 Q @ > 4 ? w j 1 P S 7
!AIVDM,1,1,,A,1 0 3 > r C P P ? w < t S F 0 l 4 Q @ > 4 ? w n 1 ‘ R L
!AIVDM,1,1,,A,1 0 3 > r C P P ? v < t S F 0 l 4 Q @ > 4 ? w > 0 8 I I
!AIVDM,1,1,,A,1 0 3 > r C P P 0 2 1 F G B q d U O H D T ? w > 0 5 H p
!AIVDM,1,1,,A,1 0 3 > r C P P 0 2 1 F G F a d U O q j t w v T 0 0 S b
!AIVDM,1,1,,A,1 0 3 > r C P P 0 1 1 F G B I d U O P Q q w w l 0 @ D n
!AIVDM,1,1,,A,1 0 3 > r C P P 0 2 1 F G A i d U N k 1 I O v V 0 5 I @
!AIVDM,1,1,,A,1 0 3 > r C P P 0 1 1 F G C c d U N f A H ? w > 0 D 0 0
!AIVDM,1,1,,A,1 0 3 > r C P P 0 4 1 F G @ q d U N e n N w v V 0 < 0 0
!AIVDM,1,1,,A,1 0 3 > r C P P 0 1 1 F G ? U d U N l o E ? w l 0 5 I l
!AIVDM,1,1,,A,1 0 3 > r C P P ? w < t S F 0 l 4 Q @ > 4 ? v N 0 @ D E
!AIVDM,1,1,,A,1 0 3 > r C P P 0 1 1 F G A 1 d U N i F M w v R 0 L 0 0
!AIVDM,1,1,,A,1 0 3 > r C P P 0 0 1 F G A S d U N l W 1 w w < 0 ¡ 0 0
!AIVDM,1,1,,A,1 0 3 > r C P P ? w < t S F 0 l 4 Q @ > 4 ? v f 1 P S >
!AIVDM,1,1,,A,1 0 3 > r C P P ? w < t S F 0 l 4 Q @ > 4 ? v f 1 d 0 0
!AIVDM,1,1,,A,1 0 3 > r C P P ? w < t S F 0 l 4 Q @ > 4 ? v f 1 h 8 U
!AIVDM,1,1,,A,1 0 3 > r C P P 0 4 1 F G B A d U O G D C w v V 0 0 S 9

Chapter 8

Conclusions

8.1 Conclusions

The aim of this study was to design and implement a SDR AIS receiver. In AIS, different types of messages are broadcast over the data link. AIS uses a FM/GMSK modulation scheme. The thesis has focused on the development of various components of the AIS receiver.

8.1.1 GMSK modem development

The AIS receiver was designed and simulated in Matlab first in order to evaluate its functionality and its performance in practise. The first part involved the development of the modulator and demodulator Matlab prototype models in order to verify the algorithms used. The DDS technique was used in the implementation of the GMSK modulator. The FM discriminator was used to demodulate the GMSK signal. The GMSK modem was then implemented in C++. The early-late gate symbol synchroniser was developed in Matlab and the algorithms used to implement this component were ported over to C++ for the implementation of the C++ symbol synchroniser.

8.1.2 Packet filter model development

During the AIS system development, it became apparent that for the system to perform in real time, certain modifications will be needed on the development of certain components of the system in C++. The packet filter model component was designed and implemented in C++ to effectively detect the start flag and stop flag in real time.

8.1.3 CRC bytefilter model development

Algorithms were developed to calculate the CRC checksum of the received AIS messages. Feedback loops were implemented successfully between the various components of the AIS

receiver in C++. Algorithms were developed to interpret the AIS messages that passed the CRC checksum test.

8.1.4 System performance and results

Actual AIS signals were demodulated and then, high pass filtered, amplified, symbol synchronised, decoded, packet filtered, bit-destuffed and translated after passing the CRC checksum. The results obtained in Chapter 5 and Chapter 7 indicate the performance of the Matlab AIS receiver. The receiver high pass filtered, amplified, symbol synchronised the demodulated GMSK baseband signal successfully and then decoded, packetfiltered, bit-destuffed, interpreted and translated the encapsulated AIS message after successfully checking the CRC checksum. However, the Matlab AIS receiver was limited in that it could not function in real time.

The results in Chapter 7 indicate the performance of the C++ receiver that was set to be designed and implemented. The results in Figure 7.12, Table 7.3, Table 7.5 and Table 7.6 show that the SDR AIS receiver is capable of high pass filtering, amplifying, symbol synchronising, decoding, bit-destuffing, error checking, translating and interpreting the the AIS messages in real time. The messages were identified correctly and the correct MMSI of the AIS is shown in the results. The messages in the above tables shows the correct CRC checksum and the encapsulated strings indicates the valid AIS characters.

8.1.5 Future work

The work done in this thesis has gone into great length to develop a SDR AIS receiver. There are some shortcomings on the current system and some suggestions for potential further work are outlined below:

- Currently the AGC amplify both the noise component and the signal during AIS receiver operation. The AGC can be modified to operate like `squelch`, where it start amplification when it detects the presence of the AIS signal. This can be done by setting a threshold level above the noise floor.

- Currently the algorithm used to translate the AIS messages gives the communication state as a number. Further work can be done to interpret the communication state to show the value of the slot-timeout and other parameters as stated in Table B.1 and Table B.2.

- The system does not output AIS messages every time the AIS transmits. This is because of the processing load that is imposed on the computer used. The sampling rate used by the ADC (data acquisition card) is 78600 samples per second. The computer used to implement the SDR receiver need to match this processing speed in order to avoid errors. (The computer implementing the SDR AIS receiver does not process samples fast enough to be synchronised to the ADC.) The first point discussed about

the AGC operating like `squelch` can solve this problem by reducing the processing load on the PC.

- The current system was designed and developed to only recognise messages “1” and “5” of an AIS. But, The system currently detects some messages that pass the CRC checksum and those messages need to be verified. This can be done by further developing the system to recognise all the 22 AIS messages.

8.2 Contributions

The AIS receiver was designed and then implemented in SDR. The SDR AIS receiver operated successfully by receiving and translating AIS position report messages in real-time. The real-time implementation of the SDR AIS receiver was achieved, which was the aim of the thesis. The translated messages were verified by comparing them to the messages from the AIS RS232 port.

Bibliography

[1] BLINCHIKOFF, J. H. and ZVEREV, I. A., *Filtering in the Time and Frequency Domains*. John Wiley & Sons, 1976.

[2] COMBLOCK, "FSK/MSK/GFSK/GMSK DIGITAL MODULATOR." http://www.comblock.com/download/com1028.pdf. 2006.

[3] CRONJE, J. J., "Software Architecture Design of a Software Defined Radio System." Master's thesis, University of Stellenbosch, 2004.

[4] DIANA, R., JOHNSON, M., and NGUYEN, M. T., "Bandwidth-Efficient Modulation Through Gaussian Minimum Shift Keying." *Space Communications*, Winter 2001/2002, Vol. 3, No. 1.

[5] DIRECTOR, W. S., *CIRCUIT THEORY A Computational Approach*. John Wiley & Sons, 1975.

[6] DXFM, "The Basics of VHF and UHF Signal Propagation." http://www.dxfm.com/Content/propagation.htm. November 2006.

[7] ERA TECHNOLOGY LTD, "Electromagnetic Compatibility Aspects of Radio-based Mobile Telecommunications Systems." tech. rep., LINK Personal Communications Programme, January 2006.

[8] HAYKIN, S., *An Introduction to Analog and Digital Communications*. New York: John Wiley & Sons, 1980.

[9] IALA, "Technical Clarification on Recommendation ITU-R.1371-1." in *IALA Guidelines on The Automatic Identification System (AIS)*, vol. 1 of *Edition 1.3*, IALA, December 2002.

[10] IALA, "Technical issues." in *IALA Guidelines on The Automatic Identification System (AIS)*, vol. 1 of *Part II, Edition 1.1*, IALA, December 2002.

[11] IALA, "Operational issues." in *IALA Guideline No. 1028 on The Automatic Identification System (AIS)*, vol. 1 of *Part I, Edition 1.3*, IALA, December 2004.

[12] IEC, "Maritime navigational and radiocommunication equipment and systems – digital interfaces." in *IEC 61162-1*, vol. 1 of *Part 1*, IEC, December 2000.

[13] IEC, "Maritime navigational and radiocommunication equipment and systems – automatic identification systems (AIS) IEC 61993-2." in *Class A AIS – Operational and performance requirements, methods of test and required test results*, Part 2, IEC, 2001.

[14] INMARSAT, "VHF (Very High Frequency)." tech. rep., December 2006.

[15] ISAAC, M. G., "Automatic Gain Control (AGC) circuits, Theory and Design." tech. rep., University of Toronto, 2001.

[16] JONES, C. E., "The Basics of Radio Wave Propagation." tech. rep., University of Tennessee, December 2006.

[17] KOSTEDT, F. and KEMELING, J. C., "Practical GMSK Data Transmission." tech. rep., MX.COM,INC, August 2003.

[18] KRAMER, H. J., "Zasat-002 (south african satellite) / sumbandilasat." November 2006.

[19] LARSON, J. W. and WERTS, J. R., *Space Mission Analysis and Design*. Third edition. Space Technology Library, 2005.

[20] LASTER, J. D., *Robust GMSK Demodulation Using Demodulation Diversity and BER Estimation*. PhD thesis, Virginia Polytechnic Institute and State University, March 1997.

[21] MANASSEWITSCH, V., *Frequency Synthesizers—Theory and Design*. Second edition. New York: Wiley, 1980.

[22] MARINEWAYPOINTS.COM, "Nautical Glossary." tech. rep., MarineWaypoints.com, December 2006.

[23] MEHROTTRA, A., *Cellular Radio Performance Engineering*. Boston. London: Artech House, 1994.

[24] PROAKIS, J. G. and MANOLAKIS, D. G., *Digital Signal Processing, Principles, Algorithms, and Applications*. Prentice Hall International Editions, 1996.

[25] REED, H. J., *Software Radio: A Modern Approach to Radio Engineering*. Prentice Hall, 2002.

[26] RICE, M. and LONG, D. G., *Introduction to Analog and Digital Communication Theory*. Brigham Young University, 1994.

[27] SCHWARTZ, M., *Mobile Wireless communications*. Cambridge, United Kingdom: Cambridge University Press, 2005.

[28] SDR FORUM, "SDR Forum FAQs." http://www.sdrforum.org/faq.html. April 2006.

[29] SERIES, E. E. T., "Very-High-Frequency and Above Communications." tech. rep., Integrated Publishing, December 2006.

[30] SKLAR, B., *Digital Communications—Fundamentals and Applications*. Englewood Cliffs: Prentice Hall, 1988.

[31] STREMLER, F. G., *Introduction to Communication Systems*. Third edition. Reading: Addison-Wesley, 1990.

[32] SUNSPACE, "Programmes." http://www.sunspace.co.za/programmes/ZA002.htm. August 2006.

[33] TUTTLEBEE, W., *Software Defined Radio: Enabling Technologies*. First edition. New York: Wiley & Sons, 2002.

[34] U.S. COAST GUARD, "AUtomatic Identification System (AIS)." tech. rep., Coast Guard Fact File, April 2006.

[35] U.S. COAST GUARD, "How AIS Works." tech. rep., The Navigation Center of Excellence, April 2006.

[36] VAN ROOYEN, G.-J., *The theory of frequency synthesis*. University of Stellenbosch, 2002.

[37] VAN ROOYEN, G.-J., "Software-Defined Radio, An Intecnet Presentation." tech. rep., University of Stellenbosch, 2006.

[38] WITKOWSKY, J., "A hardware emulator testbed for a software defined radio." Master's thesis, University of Stellenbosch, 2003.

[39] ZIEMER, R. E. and TRANTER, W. H., *Principles of Communications: Sytems, Modulation and Noise*. Fourth edition. New York: John Wiley & Sons, 1995.

Appendix A

Position report and Static and voyage related data

A.1 Position report Message structure

A Position Report is an AIS message 1, 2 or 3. Table A.1 shows how the position report message is structured.
NB: A knot is speed used by ships and it is equal to one nautical mile per hour. A speed of one nautical mile (6,076 feet or or 1,852 meters) per hour [22]. Distance at sea is measured in nautical miles. A nautical mile is about 6076 feet, 1.15 statute miles or 1852 meters. [22] says that Nautical miles have the unique property that a minute of latitude is equal to one nautical mile. Measurement of speed is done in knots where one knot equals one nautical mile per hour.

A.2 Static and voyage related data message structure

The static and voyage related data, which is AIS message five "5", is only used by a Class A ship-borne mobile equipment. This message is transmitted routinely every 6 minutes. The static and voyage related data message will also be send immediately after any parameter value has been changed or in response to an interrogation by a competent authority [11]. Table A.1 shows how the static and voyage related data message is structured.

Table A.1: *Message 1, 2 and 3: Position report (adapted from [12]).*

Parameter	**Bits**	**Description**
Message ID	6	Identifier for this message [1, 2 or 3]
Repeat indicator	2	Used by the repeater to show how many times the message was repeated; [0-3; and 3 = do not repeat anymore]
User ID	30	MMSI number
Navigational status	4	0 = under way using engine, 1 = at anchor, 2 = not under command
Rate of turn	8	127 (-128 (80h) indicates not available, which should be the default).
Speed over ground (SOG)	10	Speed over ground in 1 / 10 knot steps (0-102.2 knots). 1023 = not available, 1022 = 102.2 knots or higher
Position accuracy	1	1 = high, 0 = low, 0 = default
Longitude	28	Longitude in 1 / 10000 min (180 ,East = positive, West = negative. 181 = not available = default
Latitude	27	Latitude in 1 / 10000 min (90 , North = positive, South = negative. 91 = not available = default
Course over ground (COG)	12	Course over ground in 1 / 10 (0-3599). 3600 = not available = default. 3 601-4 095 should not be used.
True heading	9	Degrees (0-359). (511 indicates not available = default.)
Time stamp	6	UTC second when report was generated (0-59), or 60 if time stamp is not available, which should also be the default value.
Reserved for regional application	4	Reserved for definition by a competent regional authority.Should be set to zero, if not used for any regional application. Regional application should not use zero.
Spare	1	Not used. Should be set to zero.
RAIM flag	1	RAIM (Receiver autonomous integrity monitoring) flag of electronic position fixing device; [0 = default; 1 = RAIM in use]
Communication State	19	Defined in separate Table B.1 (see below).

Table A.2: *Ship static and voyage related data report (content and format) (adapted from [11]).*

Parameter	**No. of bits**	**Description**
Message ID	6	Identifier for this message (5)
Repeat indicator	2	Used by the repeater to indicate how many times a message has been repeated. 0 3; default = 0; 3 = do not repeat again.
User ID	30	MMSI number
AIS Version indicator	2	0 = Station compliant with AIS Edition 0 (Rec. ITU-R M. 1371 1); 1 3 = Station compliant with future AIS Editions 1, 2 and 3
IMO number	30	1 999999999 ; 0 = not available = default
Call sign	42	7 x 6 bit ASCII characters, @@@@@@@ = not available = default
Name of Ship	120	Maximum 20 characters 6 bit ASCII, @@@@@@@@@@@@@@@@@@@@ = not available = default
Type of ship and cargo type	8	0 = not available or no ship = default; 1 - 99 = as defined in Table 11 [11]; 100 - 199 = reserved, for regional use; 200 255 = reserved for future use.
Dimension/Reference for Position	30	Reference point for reported position; Also indicates the dimension of ship in metres.
Type of Electronic Position Fixing Device	4	0 = Undefined (default); 1 = GPS, 2 = GLONASS, 3 = Combined GPS/GLONASS, 4 = Loran-C, 5 = Chayka, 6 = Integrated Navigational System, 7 = surveyed, 8 15 = not used
ETA	19	Estimated Time of Arrival;MMDDHHMM UTC month; 1 12; 0 = not available = default; day; 1 31; 0 = not available = default; hour; 0 23; 24 = not available = default; minute; 0 59; 60 = not available = default
Maximum Present Static Draught	8	in 1 / 10 m; 255 = draught 25.5 m or greater, 0 = not available = default; in accordance with IMO Resolution A.851
Destination	120	Maximum 20 characters using 6-bit ASCII, @@@@@@@@@@@@@@@@@@@@ = not available = default
DTE	1	Data terminal ready (0 = available 1 = not available = default)
Spare	1	

Appendix B

SOTDMA communication state

Table B.1: *SOTDMA communication state table (adapted from [9]).*

Parameter	**Number of bits**	**Description**
Synchronisation state	2	0 indicates UTC direct 1 indicates UTC indirect. 2 Station is synchronised to a base station. 3 Station is synchronised to another station based on the highest number of received stations.
Sot time out	3	Specifies frames remaining until a new slot is selected. 0 means that this was the last transmission in this slot. 1-7 means that 1 to 7 frames respectively are left until slot change.
Sub message	14	The sub message depends on the current value in slot time-out as described in Table B.2.

The SOTDMA communication state is applicable only to the slot in the channel where the relevant transmission occurs [9].

Table B.2: *The sub message of the SOTDMA communication state (adapted from [9]).*

Slot time-out	**Sub message**	**Description**
3, 5, 7	Received stations	number of other stations (not own station) which the station currently is receiving (between 0 and 16 383)
2, 4, 6	Slot number	Slot number used for this transmission (between 0 and 2249)
1	UTC hour and minute	If the station has access to UTC, the hour and minute should be indicated in this sub message. Hour (0-23) should be coded in bits 13 to 9 of the sub message (bit 13 is MSB). Minute (0-59) should be coded in bit 8 to 2 (bit 8 is MSB)
0	Slot offset	If the slot time-out value is 0 (zero) then the slot offset should indicate the relative jump to the slot in which transmission will occur during the next frame. If slot offset is zero, the slot should be de-allocated after transmission

Appendix C

Gaussian filter coefficients

A random variable X is said to be normally distributed with mean μ and variance σ^2 if its probability density function (*pdf*) is:

$$y(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{C.1}$$

This equation generates a gaussian distribution for a given average and standard deviation. The Gaussian filter coefficients of the Gaussin filter with BT = 0.5 were calculated from Equation C.1 using the mean of $\mu = 0$ and $\sigma = 0.07$. [2] gives the gaussian filter with BT = 0.5 as a 17-tap FIR filter with the following impulse response (coefficients):

```
Coeff(0) = 1/1024 Coeff(1) = 6/1024 Coeff(2) = 28/1024 Coeff(3) =
95/1024 Coeff(4) = 258/1024 Coeff(5) = 568/1024 Coeff(6) =
992/1024 Coeff(7) = 1376/1024 Coeff(8) = 1536/1024 Coeff(j = 9:16)
= coeff(16-j);
```

Where j is equal to 16. Figure C.1 shows the Gaussian filter impulse response of the calculated coefficients using mean $\mu = 0$ and $\sigma = 0.07$, and the coefficients from [2]. The Gaussian filter coefficients of the Gaussin filter with BT = 0.3 were calculated from Equation C.1 using the mean of $\mu = 0$ and $\sigma = 0.192$. [2] gives the gaussian filter with BT = 0.3 as a 17-tap FIR filter with the following impulse response (coefficients):

```
Coeff(0) = 3/1024 Coeff(1) = 7/1024 Coeff(2) = 17/1024 Coeff(3) =
36/1024 Coeff(4) = 72/1024 Coeff(5) = 130/1024 Coeff(6) = 220/1024
Coeff(7) = 336/1024 Coeff(8) = 488/1024 Coeff(9) = 640/1024
Coeff(10) = 784/1024 Coeff(11) = 896/1024 Coeff(12) = 928/1024
Coeff(j = 13:24) = coeff(24-j);
```

Where j is equal to 24. Figure C.2 shows the Gaussian filter impulse response of the calculated coefficients using mean $\mu = 0$ and $\sigma = 0.192$, and the coefficients from [2].

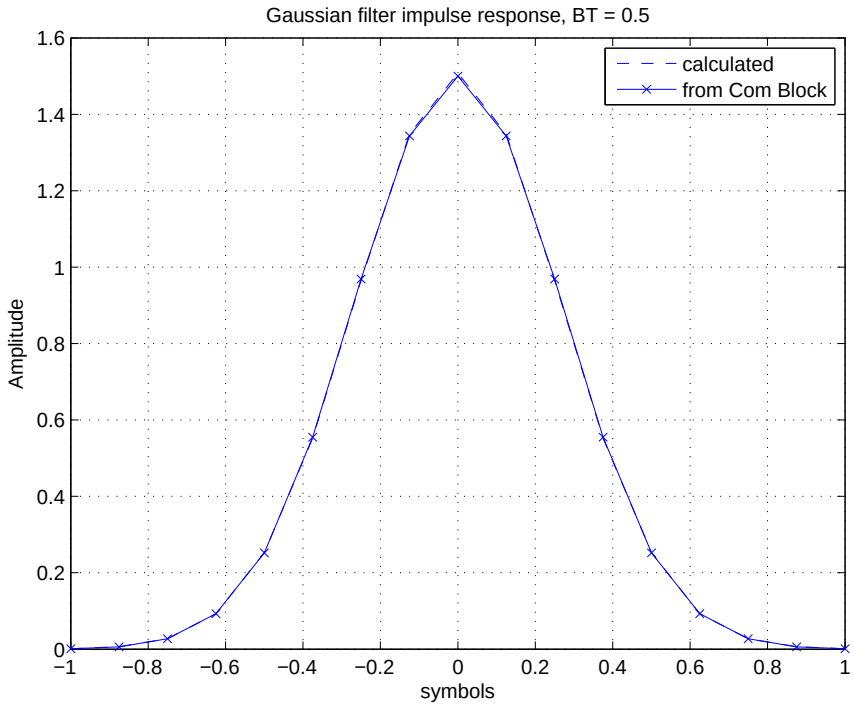


Figure C.1: *Gaussian filter impulse response. Using 8 samples/symbol.*

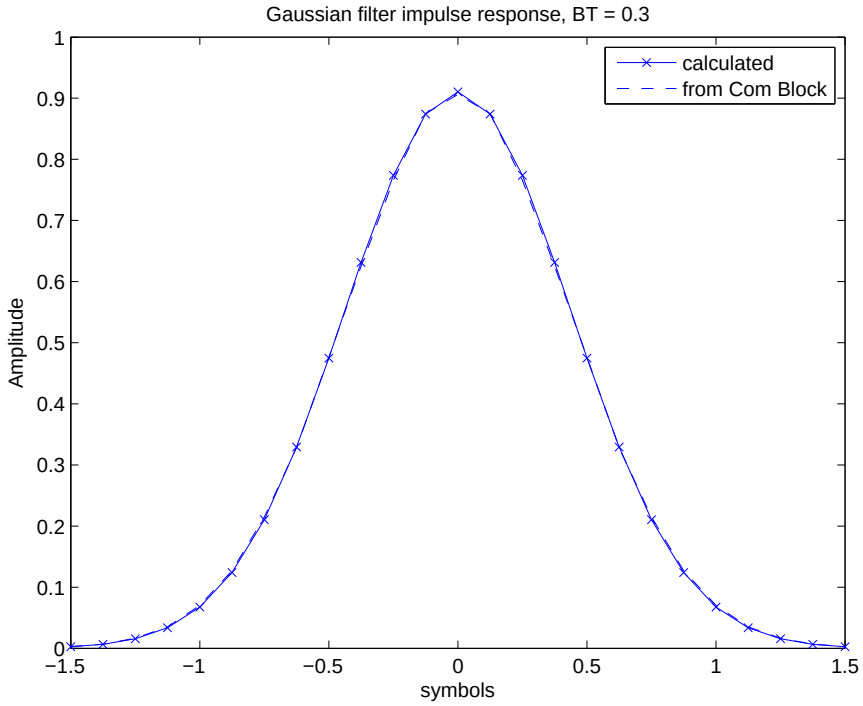


Figure C.2: *Gaussian filter impulse response. Using 8 samples/symbol.*

Appendix D

Organising binary bits into message one

D.0.1 An example of decoding 'AIS Message 1".

Identification of the AIS message is done from the first six bits of the binary message data [13]. The message number is given by the decimal eqiuvalent of the binary number represented by those first six bits. If the first six binary bits are "000001", then the message number is = message 1. After this number is known, the remaining blocks of the message can be filled using the information in table 15 found in ITU-R M.1371-1 [13]. The ordering of bits for "message 1" are as shown in Table D.1: The way in which binary bits (represented by encapsulation symbol), are converted to Encapsulation string and then to meaningful information, is demonstrated by decoding and interpreting the message shown in Table D.2:

Table D.1: *Table for organising AIS message "1" (adapted from [13]).*

Parameter	Bits position
Message ID	Bits 1-6
Repeat Indicator	Bits 7-8
User ID	Bits 9-38
Navigational status	Bits 39-42
Rate of turn	Bits 43-50
SOG	Bits 51-60
Position Accuracy	Bits 61
Longitude	Bits 62-89
Latitude	Bits 90-116
COG	Bits 117-128
True Heading	Bits 129-137
UTC second when report generated	Bits 138-143
Regional Application	Bits 144-147
Spare	Bit 148
RAIM	Bit 149
Communications State	Bits 150-168

Table D.2: *VDM decoding example showing Binary conversion of symbols (adapted from [13]).*

VDM bit position						Encapsulation Symbol string	Bits represented by symbol					
1	2	3	4	5	6	1	0	0	0	0	0	1
7	8	9	10	11	12	P	1	0	0	0	0	0
13	14	15	16	17	18	0	0	0	0	0	0	0
19	20	21	22	23	24	0	0	0	0	0	0	0
25	26	27	28	29	30	0	0	0	0	0	0	0
31	32	33	34	35	36	O	0	1	1	1	1	1
37	38	39	40	41	42	h	1	1	0	0	0	0
43	44	45	46	47	48	1	0	0	0	0	0	1
49	50	51	52	53	54	I	0	1	1	0	0	1
55	56	57	58	59	60	T	1	0	0	1	0	0
61	62	63	64	65	66	1	0	0	0	0	0	1
67	68	69	70	71	72	s	1	1	1	0	1	1
73	74	75	76	77	78	v	1	1	1	1	1	0
79	80	81	82	83	84	T	1	0	0	1	0	0
85	86	87	88	89	90	P	1	0	0	0	0	0
91	92	93	94	95	96	2	0	0	0	0	1	0
97	98	99	100	101	102	r	1	1	1	0	1	0
103	104	105	106	107	108	:	0	0	1	0	1	0
109	110	111	112	113	114	4	0	0	0	1	0	0
115	116	117	118	119	120	3	0	0	0	0	1	1
121	122	123	124	125	126	g	1	0	1	1	1	1
127	128	129	130	131	132	r	1	1	1	0	1	0
133	134	135	136	137	138	w	1	1	1	1	1	1
139	140	141	142	143	144	b	1	0	1	0	1	0
145	146	147	148	149	150	0	0	0	0	0	0	0
151	152	153	154	155	156	E	0	1	0	1	0	1
157	158	159	160	161	162	q	1	1	1	0	0	1
163	164	165	166	167	168	4	0	0	0	1	0	0

Appendix E

Calculating the required bit error rate

The probability of the detector making an incorrect decision is called probability of bit error (P_e) [30]. The probability of bit error for one bit GMSK modulation scheme is the same as that of one bit MSK and it is given as [23]:

$$P_e(GMSK) = \frac{1}{2}\, erf_c\left(\sqrt{E_b/2N_0}\right) \tag{E.1}$$

where erf_c is the error function. Because a position report in AIS consist of 168 bits, the probability of bit error $P_e = 10^{-2}$ is used. The probability of correctly detecting a bit P_c is given as:

$$P_c = 1 - P_e \tag{E.2}$$

The probability of correctly detecting a whole AIS packet (each packet for a position report message consistss of 168 bits) is given by:

$$P(correctpacket) = P_c(bit\ 1) \cap P_c(bit\ 2) \cap P_c(bit\ 3) \cap \cdots \cap P_c(bit\ 168) \tag{E.3}$$

which is equal to:

$$P(correctpacket) = {P_c}^{168} = 0.99 \tag{E.4}$$

$$P_c = \sqrt[168]{0.99} = 0.9999 \tag{E.5}$$

The required bit error rate (the probability that a data bit is incorrectly received) using the GMSK modulation is:

$$P_e = 1 - P_c = 10^{-4} \tag{E.6}$$

Appendix F

Paper presented at AIAA/USU Small Satellite Conference 2006

A Software Defined Radio AIS for the ZA-002 Satellite

Kgabo Frans Mathapo

Advisor: Dr G-J van Rooyen
Digital Signal Processing & Telecommunications Research Group
University of Stellenbosch, Stellenbosch,Western Cape, South Africa
{kgabom,gvrooyen}@sun.ac.za

***Abstract*—ZA-002 is the second South African satellite, and is scheduled to be launched in December 2006. A software defined radio (SDR) automatic identification system (AIS) receiver is proposed as a possible experimental payload for this satellite. The AIS receiver can be used to track and store movement of ships at sea, and then forward this information to the ground station upon request.**

This paper demonstrates the design of a SDR AIS receiver for a satellite. The design of a GMSK/FM modem as used in AIS is presented, followed by simulation results.

1. Introduction

The ZA-002 Pathfinder satellite is the second satellite developed fully in South Africa. The satellite's primary mission is earth observation, and it will be used *inter alia* for disaster management, water resource management agricultural monitoring. Provision has been made for two 1-kg experimental payloads to be included for the secondary mission of the satellite. One possible experimental payload is a software defined radio (SDR) automatic identification system (AIS) receiver.

The primary purpose of the SDR AIS experimental payload is to monitor marine traffic on the South African coastline. The secondary purpose is to carry out a scientific experiment in SDR that will demonstrate the possibility of reconfiguring a radio system on a satellite through software updates, and to serve as a proof-of-concept of SDR for satellite communications systems. Software defined radio is a technology that is currently being researched at Stellenbosch University because of its potential to help realize reconfigurable radio systems and networks, that use the same hardware for different applications [1]. SDR allows reconfigurable radios, and system upgrades can be done through software updates. Generic hardware can then be used for a variety of applications [13].

Traditional analog radios consist of analog components that are subject to factors such as temperature, and their parameters may degrade with time. Because of this potential instability, analog radios may need to be recalibrated often. Digital systems, on the other hand, perform signal processing consistently, are not sensitive to temperature and their performance generally does not degrade with time.

SDR is particularly suitable to space applications, because communications systems on mobile platforms such as satellites can then be maintained and reconfigured using software updates, thus increasing the satellite lifetime. The satellite mission can then be changed by reconfiguring and upgrading the radio through software updates – capability that is not possible with conventional hardware-defined radios. It is from this background that a SDR AIS receiver is designed and implemented for the ZA-002 satellite.

2. ZA-002 Overview

The ZA-002 is a Low Earth Orbit (LEO) satellite with multispectral imaging capability. The main payload on the satellite is a multi-sensor microsatellite imager (MSMI) which will be used to take high-resolution images of the earth. Collected images are downloaded to the ground station using an S-band downlink. Low-bandwidth VHF and UHF channels are used as telemetry, tracking and command links. The satellite is three-axis stabilized, and will be launched into a sun-synchronous orbit at a nominal altitude of 500 km, with a nominal local time (at the equator) of 10:00 AM. The ZA-002 is a 70-kg satellite and the launch is tentatively scheduled for late 2006 or early 2007. The entrance pupil of the imager will normally be pointed at nadir. Two South African based ground stations will be used to download data and command the satellite, one at the Stellenbosch University Electronic Systems Laboratory (ESL), and one at Hartebeeshoek, near Pretoria. The remote sensing capability of the satellite will be used to monitor land, water and vegetation.

3. AIS Functional overview

Automatic Identification (AIS) is a shipboard broadcast system that continuously and autonomously broadcasts a ship's identification, position and other maritime navigational messages [2]. AIS operates in the VHF maritime band and it allows ships to easily track, identify and exchange navigation information between one another or the shore. Each AIS system consists of one VHF transmitter and two VHF Time Division Multiple Access (TDMA) receivers [2]. Time division multiplexing is a method whereby each channel is dedicated to a single line during a specific time slot. TDMA works by assigning multiple users or stations to a single frequency channel. A carrier frequency is assigned to the channel, and digital signals sent out from different stations are transmitted on that frequency in specified time slots in a repetitive frame structure.

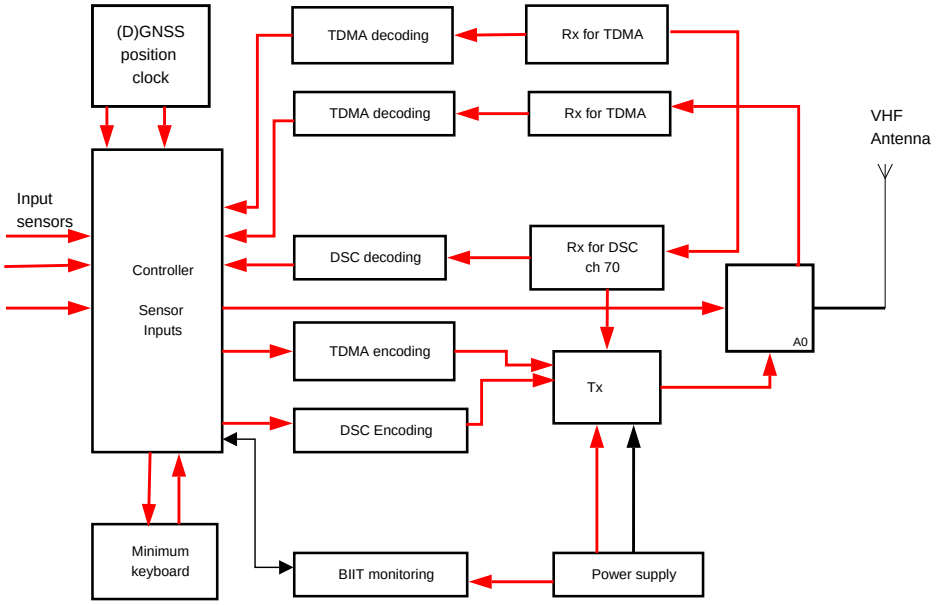


Figure 1: Overview of a ship-borne AIS mobile station.

Figure 1 shows a block diagram of an AIS mobile station. AIS uses 161.975 MHz and 162.025 MHz as carrier frequencies. The AIS receiver should be capable of operating on 25-kHz or 12.5-kHz channels. Bandwidth-adapted frequency-modulated Gaussian-filtered minimum shift keying (GMSK/FM) is used in the AIS physical layer.

A non-return-to-zero inverted (NRZI) waveform is used for data encoding. The NRZI encoded data are then GMSK encoded before frequency modulating the carrier. AIS uses a GMSK bandwidth-time (BT) product of between 0.3 and 0.5.

4. System-Level Design

Figure 2 shows how the satellite will be used to monitor marine traffic. The satellite is equipped with an SDR AIS receiver for ship surveillance purposes. Each ship is fitted with an AIS transponder, and continuously broadcasts AIS information.

AIS uses a bit-oriented protocol for data transfer, which is based on the high-level data link control (HDLC) standard, as specified by ISO/IEC 3309: 1993 – *Definition of packet structure* [4]. Data is transmitted using a transmission packet as shown in Figure 3.

Packets are sent from left to right in the diagram. Data transmission begins with a 24-bit training sequence (the preamble) consisting of one synchronization segment. This segment consists of alternating ones and zeros (0101…) and the sequence can either start with a 1 or a 0 since NRZI encoding is used [4]. The training sequence is used to synchronize the receiver. The start flag is 8 bits long and consists of a standard HDLC flag, marking the start of each transmission packet. The data portion is 168 bits long in the default transmission packet.

The FCS uses a 16-bit cyclic redundancy check (CRC) polynomial to calculate the checksum as defined in ISO/IEC 3309: 1993 [4]. The CRC bits are preset to one (1) at the beginning of the CRC calculation. Only the data portion is included in the checksum calculation. The end flag is 8 bits long and identical to the start flag. The buffer is normally 24 bits long and is used to compensate for bit stuffing distance delays, repeater delay and synchronization jitter. The total length of the normal packet is 256 bits, which is equivalent to one TDMA time slot.

The satellite receives the AIS signals on the VHF uplink. The AIS information is then stored and forwarded to the ground station using either the S-band channel or the UHF downlink provided on the satellite (the S-band channel allows high-speed data transmission). Both downlink channels were developed separately from the project described in this paper, and are not considered further here.

Proposed system

Figure 4 shows an overview of the satellite subsystems used to receive the AIS signals. The on-board computer (OBC) used

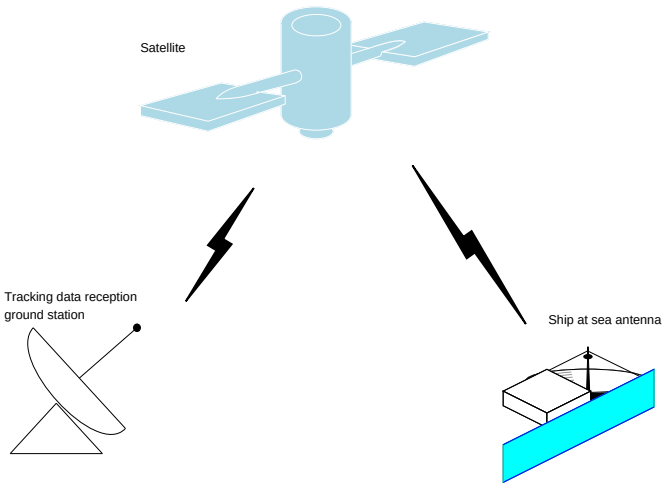


Figure 2: Diagram of the satellite monitoring marine traffic and relaying it to a ground station.

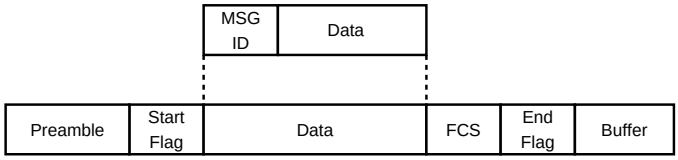


Figure 3: Structure of the AIS data transmission packet [4].

for satellite control is a Sun Space and Information Systems design, and it contains a Hitachi SH4 as primary CPU.

The RF front end consists of the RF amplifier, a mixer for downconversion of the RF signal to the required intermediate frequency (IF), a bandpass filter or lowpass filter to remove the other mixer products and allow only the IF to pass through, and an ADC to digitize the analog IF signal. The digitized IF signal is then routed to the SDR processor. Demodulation, filtering, synchronization and signal recovery are all performed in software by the digital processor.

The next few sections will detail the design of the signal processing software, starting with an overview of the SDR architecture.

5. SDR

Stellenbosch University (SU) has been involved in SDR research since 1998. The university is developing a software architecture for rapid SDR development. The goal is to develop a library of reusable radio building blocks that are portable to many platforms.

A. SDR Functionality

The SU SDR architecture allows the development of a library of components that are used to build a radio system [8]. Figure 5 shows a generic SDR component.

A typical SDR component models a hardware component. The SDR components operate on the principle that all components receive, process and output streams of samples as illustrated in Figure 5, regardless of the number of input and output ports. The exact process function of each component is defined by the type of component being modeled.

B. Component methods

SU SDR components are designed in such a way that they are reusable. Three functions are used in each component to receive, process and dispatch samples between components. These functions are:

- A "read' method that acquires samples from the component's input buffer.
- A virtual "process" function that specifies what process gets performed.
- A "write" method that dispatches samples to the input buffer of the next component.

C. System topology and linking of components

The way in which different modeled components transfer data samples amongst each other depends on the topology used. Since data is streamed on a sample-by-sample basis in the SDR architecture, a control-based topology is employed. In

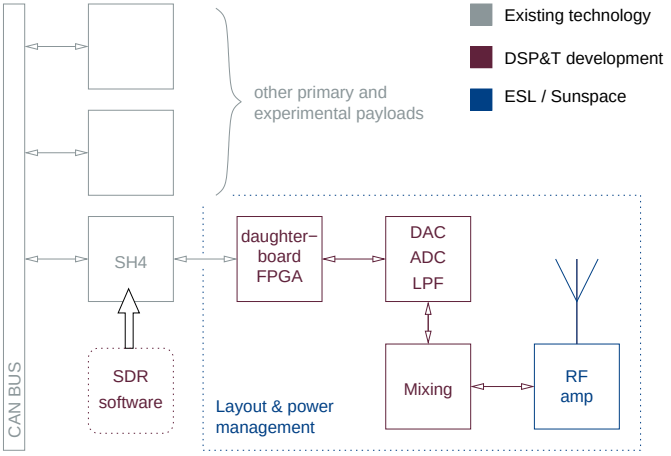


Figure 4: Block diagram of the SDR subsystem on ZA-002.

SDR, each component has its own input buffers and it is the responsibility of a director called a *subcontroller* to monitor the status of each buffer and to schedule processing. The scheduler activates process routines of each component in a round-robin schedule, allowing each component to fully deplete its input buffer [8]. The SDR components are linked by their input and output ports.

D. GMSK modulation

Minimum shift keying (MSK) is a form of continuous-phase frequency shift keying (CPFSK) binary modulation where the modulated carrier has no phase discontinuities, and frequency changes occur at the carrier zero crossings. In MSK modulation, the frequency spacing between the two frequencies transmitted during the FSK is $1/2T_b$, where T_b is the bit interval [19], [20]. This minimum frequency spacing allows the two FSK signal waveforms to be orthogonal to each other and to be correctly detected using a coherent detector. The FSK signal subsequently has phase continuity.

Gaussian minimum shift keying (GMSK) is MSK modulation with a Gaussian premodulation filter [10]. The information bit input or pulse train is passed through the Gaussian LPF, and the output of the filter is MSK modulated. This results in GMSK achieving smooth phase transitions between signal states, thereby reducing the bandwidth requirements. In most cases, the Gaussian pulse is made longer than one bit interval, which results in pulses overlapping, giving rise to intersymbol interference (ISI). The extent of the intersymbol interference is controlled by the product of the Gaussian filter bandwidth and bit duration [5]. The smooth phase transitions of GMSK signals make demodulation at the receiver more complex. For the receiver to achieve a specified bit error rate, more power must be generated at the transmitter in order to overcome the receiver noise in the presence of ISI [14]. GMSK requires more bandwidth than ordinary MSK at the receiver to effectively recover the carrier.

GMSK can be considered to be a combination of frequency and phase modulation, where the phase of the carrier is increased or decreased by $\pi/2$ over a bit period, depending on the data pattern. The rate of phase change is limited by the

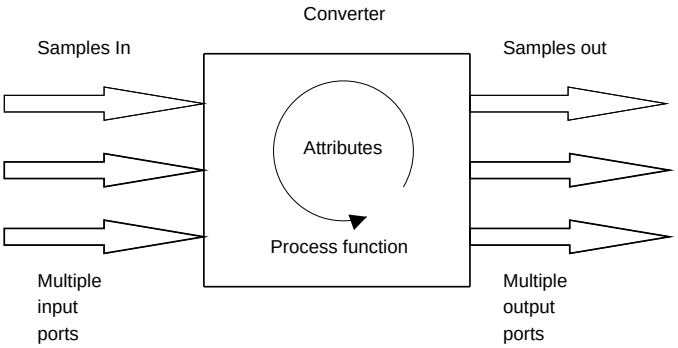


Figure 5: A typical SDR component.

Gaussian filter response. This assists in improving the bandwidth efficiency [14].

E. Detail design

1) Mathematical development

GMSK signals can be generated either by using a quadrature baseband process followed by quadrature phase modulation, or by frequency modulation. The FM technique was used to develop the GMSK modulator in this paper. This method was chosen because it is simple to implement using direct digital synthesis (DDS), and the demodulation process can be implemented using a simple FM discriminator.

The GMSK modulator was implemented by first differentially encoding the data, representing it as a non-return-to-zero inverted (NRZI) signal and then passing it through a Gaussian low-pass filter. The signal is then frequency modulated using a DDS system. Each NRZI symbol consists of eight samples. The Gaussian filtered signal was then resampled to the carrier sampling frequency, before frequency modulating the carrier.

The GMSK modem was designed to provide the GMSK signal to be demodulated by the proposed GMSK demodulator. The block diagram of the GMSK modulator is shown in Figure 7.

The first step in the modulator is the differential encoder. The differential encoder calculates an exclusive-NOR between the incoming bit and the reference bit, and outputs the differentially encoded bit.

The Gaussian filter component is designed using the coefficients of the Gaussian filter for a given bandwidth-time product. A Gaussian-shaped impulse response filter produces a signal with low side lobes and narrower main lobe than the rectangular pulse [18]. Figure 6 shows Gaussian filters with BT = 0.3 and 0.5.

The impulse response of the Gaussian LPF is given by [5]:

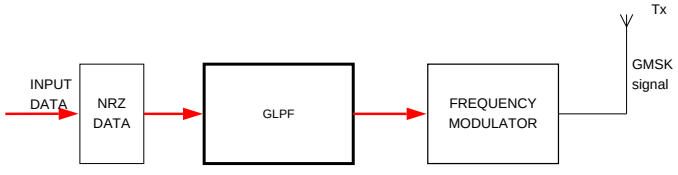


Figure 7: Block diagram of a GMSK modulator.

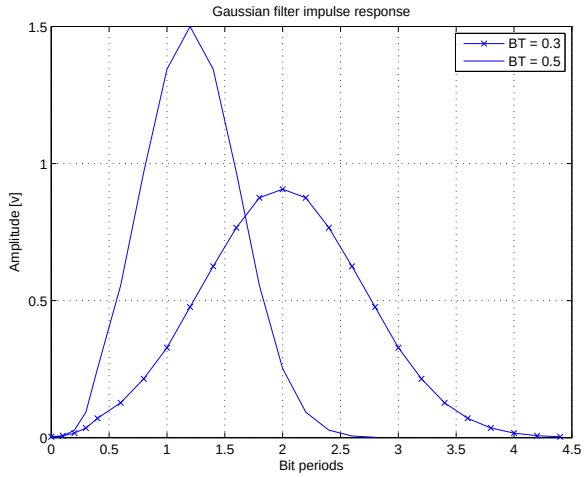


Figure 6: Gaussian filter impulse responses with BT products of 0.5 and 0.3 respectively.

$$h(t) = \frac{1}{\sqrt{2}\,\pi\,\sigma\,T} \exp\left(\frac{-t^2}{2\,\sigma^2\,T^2}\right), \tag{1}$$

where

$$\sigma = \frac{\sqrt{\ln(2)}}{2\,\pi\,BT} \tag{2}$$

and B is the 3-dB bandwidth of the filter and T is the symbol period.

The response of the Gaussian LPF to the square pulses is equivalent to convolving the filter with the square pulses and is given by:

$$g(t) = h(t) * \Pi\left(\frac{t}{T}\right). \tag{3}$$

The pulse response $g(t)$ (which is equivalent to multiplying the Fourier Transform of $h(t)$ and the Fourier Transform of $\Pi(t/T)$, can be written as:

$$g(t) = \frac{1}{2\mathrm{T}} Q\left(2\,\pi\,BT\,\frac{t - T/2}{T\sqrt{\ln(2)}}\right) - Q\left(2\,\pi\,BT\,\frac{t + T/2}{T\sqrt{\ln(2)}}\right), \tag{4}$$

where $Q(t)$ is the function

$$Q(t) = \int_t^{\infty} \frac{\exp\left(-y^2/2\right)}{\sqrt{2\,\pi}}\,dy\,. \tag{5}$$

The GMSK signal is given by:

$$s(t) = \sqrt{2\,E\,T_b}\cos\left(2\,\pi\,f_c t + \theta + \theta_0\right), \tag{6}$$

where θ_0 is the initial phase at $t = 0$.

The Gaussian filter component takes each sample from the differential encoder and filters it using the Gaussian filter coefficients. The frequency modulator component then takes the incoming Gaussian-filtered samples and frequency modulates the carrier frequency using the direct digital synthesis technique. This technique is discussed next.

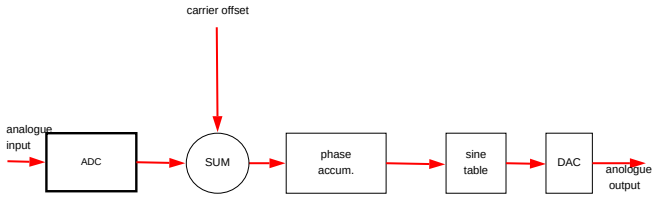


Figure 8: Block diagram of a DDS system.

2) Direct Digital Synthesis

Direct digital synthesis (DDS) is used to digitally synthesize a desired analog signal, usually a sine wave of which the frequency must be variable. DDS uses a fixed sampling frequency, and different output frequencies are obtained by sampling a sine wave using variable phase increments. If a higher frequency is desired, the sine wave is sampled at larger phase increments [15]. Figure 8 shows a block diagram of a DDS system.

The heart of the DDS system shown in Figure 8 is the phase accumulator. The value in the accumulator represents the current phase ($0 \le \phi \le 2\pi$) of the output signal wave. Every clock cycle, the phase of the accumulator is incremented by $\Delta\phi$. The size of this increment is directly proportional to the output frequency. Phase wrap from 2π to 0 radians is achieved by simply ignoring the phase accumulator overflow. The phase value in the accumulator can then be matched to a value in the sine or cosine lookup table, thus producing a sampled sine wave. The resulting sampled sine wave can be reconstructed to produce a desired output sine wave.

[7] shows that an output frequency of f_0 can be produced if the phase increment is:

$$\Delta\phi = 2\pi\left(\frac{f_0}{f_s}\right) \text{ radians.} \tag{7}$$

By dynamically changing the phase increment, it is possible to implement different modulation schemes.

The instantaneous phase of an analogue frequency modulated signal is given by:

$$\theta(t) = \omega_c t + k_f \int_0^t m(\tau)\, d\tau + \theta_0\ , \tag{8}$$

where ω_C is the carrier frequency in radians, k_f is the frequency deviation constant, $m(t)$ is the modulating signal, and θ_0 is the initial phase angle at $t = 0$.

To perform the frequency modulation (FM) in discrete form using DDS, the instantaneous phase of the FM signal can be calculated to be:

$$\theta(nT) = nTF_c + k_f T \sum_{k=1}^{n} m(kT) + \theta_0\ , \tag{9}$$

where T is the sampling interval, n is the sample number, F_C is the constant that determines the carrier frequency of the discrete-time output signal, and θ_0 is the initial phase in radians. Sampling must be performed below the half Nyquist frequency.

The DDS FM modulator can synthesize an FM signal directly at the desired broadcast frequency or at an intermediate frequency which can then be up-mixed to a desired broadcast frequency.

3) GMSK demodulator

The GMSK demodulator uses an FM discriminator. The ideal FM discriminator consists of a differentiator and a low-pass filter and a decision device.

Since GMSK is a special form of FM, the message is contained in the instantaneous frequency of the modulated signal. The demodulation process is performed as follows [16]: Let

$$x(t) = A_c \cos\left[\omega_c t + k_f \int_0^t m(\tau)\, d\tau\right], \tag{10}$$

be the GMSK signal, with k_f equal to $2\pi f_d$, where f_d is the frequency deviation. The differentiated signal is given by:

$$\begin{aligned} e(t) = &-A_c\left[2\pi f_c + 2\pi f_d m(t)\right]\cdot \\ &\sin\left[2\pi f_c t + 2\pi f_d \int_0^t m(\tau)\, d\tau\right]. \end{aligned} \tag{11}$$

Rectifying the differentiated signal and lowpass filtering results in:

$$\begin{aligned} y(t) &= \left|A_c\left[\omega_c t + k_f m(t)\right]\right| \\ &= A_c\left[\omega_c + k_f m(t)\right]. \end{aligned} \tag{12}$$

Removing the DC component yields

$$y(t) = A_c k_f m(t). \tag{13}$$

However, in the presence of additive Gaussian noise, the carrier amplitude variations cause distortion at the output of the discriminator. The receiver uses a bandpass filter to remove the out-of-band interference.

After successfully demodulating the modulated carrier, the receiver must have an accurate knowledge of the beginning of the symbol and the end of the symbol. This is necessary for the receiver to make correct symbol decisions [9]. If the receiver integrates over an interval of inappropriate length, the ability to make accurate symbol decisions will be degraded. The early-late gate is the method chosen to achieve symbol synchronization in this implementation.

The generic early-late gate synchronizer operates by performing two separate integrations of the incoming signal energy over two different ($T - d$) portions of the symbol interval, where T is the symbol interval and $d = T/2$ [9]. For the AIS demodulator the symbol synchronizer was modified by performing three separate integrations instead of two. This was done in order to use the same synchronizer when demodulating M-ary pulse amplitude modulated (PAM) signals, resulting in a more widely reusable component. Figure 9 shows a block diagram of the early-late gate symbol synchronizer.

The first integration, I_1, (the early interval) starts the integration at the loop's estimation of the beginning of a symbol period (the nominal time zero) and integrates to $T/3$, where T is the nominal symbol period. The second integral, I_2, (the central interval) delays the start of its integration for $T/3$ seconds, and then integrates to $2T/3$. The last integral, I_3, (the late interval) delays the start of its integration for $2T/3$ seconds,

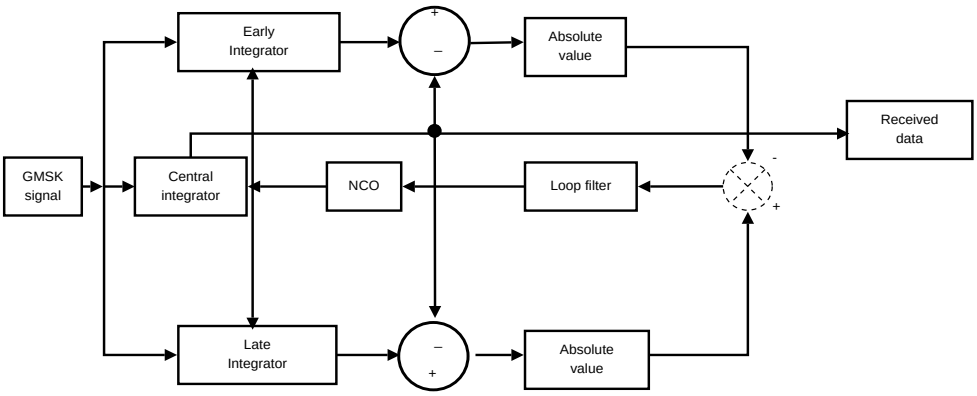


Figure 9: Block diagram of an early-late gate symbol synchronizer.

and then integrates to the end of the symbol period (the nominal time T).

The measure of the receiver's symbol timing error is now given by

$$\epsilon = \left| I_2 - I_1 \right| - \left| I_2 - I_3 \right| \tag{14}$$

which can be fed back to the loop's timing reference to correct loop timing. When the synchronizer is achieves lock, all the integrators will accumulate the same signal energy, and $\epsilon \approx 0$. Thus, when this device is synchronized, it is stable and there is no tendency to drive itself away from synchronization [9].

4) Software Design of the GMSK demodulator

Figure 10 shows a block diagram of the SDR GMSK discriminator. The modulated signal is passed through a bandpass filter with sufficient bandwidth to let through the lower and upper sidebands and to remove out-of-band interference.

The first step in the GMSK demodulation process is differentiation. Note that differentiation in the continuous-time domain can be approximated in the digital domain by calculating

$$\frac{d}{dt} x(t) \approx \frac{x[nT] - x[(n-1)T]}{T}, \tag{15}$$

where $x(t)$ is the continuous modulated signal, T is the sampling period and n is a positive integer representing the sample index. The original message signal is proportional to the phase difference of the consecutive data samples. The differentiator computes a new phase difference vector for each new sample.

The differentiated signal is then passed through the rectifier component. The rectification process is accomplished in software by taking the absolute value of the differentiated signal. The resulting signal is then divided by $2A/\pi$ (where A is the carrier amplitude) which are the coefficients for full-rectified sine wave.

The rectified signal is lowpass filtered using an 8th order butterworth low pass filter with a cutoff frequency slightly higher than half of the symbol frequency. A filter with a high cutoff frequency is desired to avoid introducing intersymbol interference at the receiver. The DC component is removed by subtracting the $2\pi f_c$ from (12). The baseband signal is recovered as follows:

$$\begin{aligned} y(t) &= A_c \left\{ \omega_c + 2\pi f_d m(t) \right\} - A_c \omega_c \\ &= A_c 2\pi f_d m(t) \end{aligned} \tag{16}$$

and then dividing by the frequency deviation term to recover the baseband signal:

$$\begin{aligned} y(t) &= \frac{\{ A_c 2\pi f_d m(t) \}}{\{ 2\pi f_d \}} \\ &= A_c m(t). \end{aligned} \tag{17}$$

The demodulated signal is then passed through the early-late gate synchronizer for bit synchronization, as described in the previous section. The central integration result, I_2, is used to decide whether the symbol represents a 1 or a 0.

The recovered symbols are finally passed through the differential decoder component. The decoding process is performed by comparing the received symbol and its delayed symbol version, to reverse the encoding process, thus recovering the transmitted symbols.

6. Physical-Layer Simulation

In this section the performance of the proposed GMSK modem is evaluated, first in ideal noise-free conditions, then in the presence of additive white gaussian noise (AWGN). The GMSK modem was simulated using Matlab. The simulation was performed using a data rate of 9600 b/s, a carrier frequency of 100 kHz, frequency deviation of 12.5 kHz, a carrier sampling frequency of 1 536 000 samples/s, and both $BT = 0.5$ and $BT = 0.3$. An input frame length of 1 000 bits was used in the bit error rate (BER) performance simulation.

Figure 11 shows the Gaussian-filtered bits. Because the information bits are sampled at 76800 samples/s and the

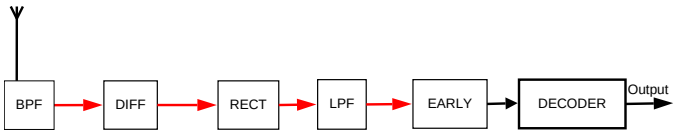


Figure 10: Block diagram of the SDR GMSK demodulator.

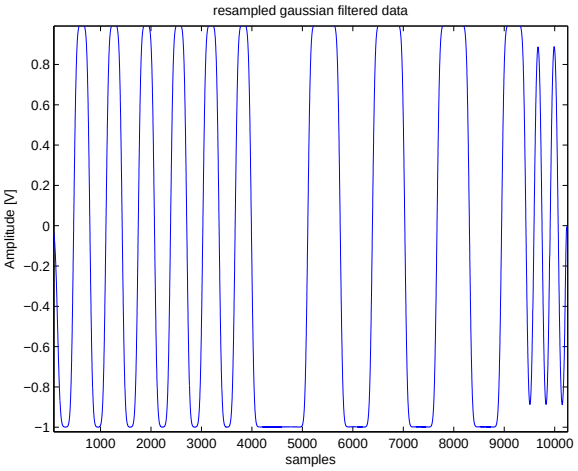


Figure 11: Resampled Gaussian-filtered data. A Gaussian filter with BT = 0.5 was used in this experiment.

carrier frequency is sampled at 1536000 samples/s, the Gaussian-filtered bit stream must be upsampled to the carrier frequency's sampling frequency.

Figure 13 and Figure 15 show the differentiated and rectified signals respectively. The results show the message signal contained in the envelope of both the differentiated and rectified signal. The DC component at 6.2832e5 corresponds with to carrier frequency ω_c in radians per second.

In Figure 12 the low-pass filter has removed the high-frequency components of the original modulated signal. The recovered baseband signal still contains an unwanted DC component proportional to the carrier frequency.

Figure 14 shows the low-pass filtered signal after passing through the early-late gate symbol synchronizer. We see that at the beginning, synchronization has not yet been achieved, and this is indicated by diamonds displayed on the graph. As the synchronizer continues to feed back the symbol timing error to the loop's timing reference to correct loop timing as in (14), there comes a time when all the integrators accumulate the same amount of signal energy, and $\epsilon \approx 0$. When this happens,

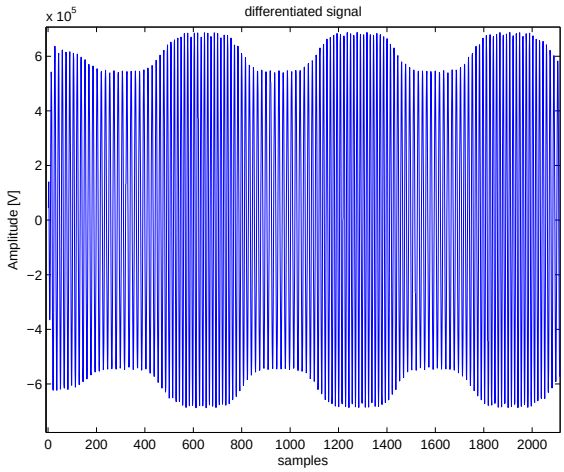


Figure 13: The differentiated FM signal.

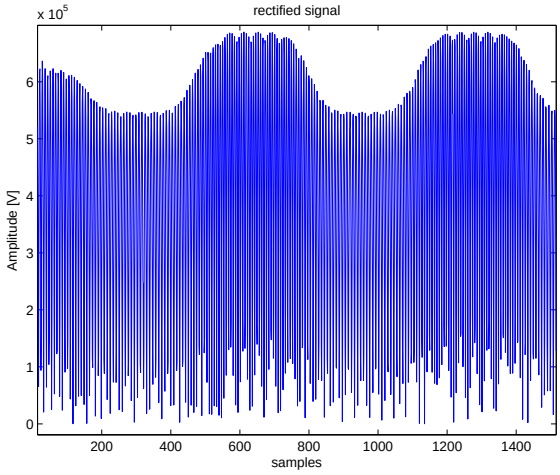


Figure 15: After differentiation, the received FM signal is rectified in order to extract its envelope.

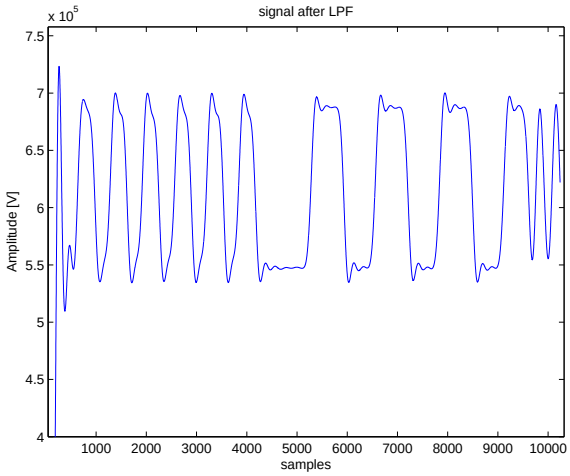


Figure 12: The rectified FM signal after lowpass filtering. The modulating signal has now been recovered.

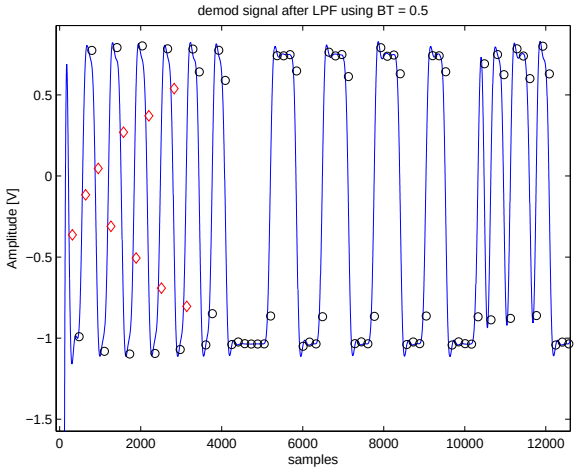


Figure 14: The demodulated GMSK signal after symbol synchronization.

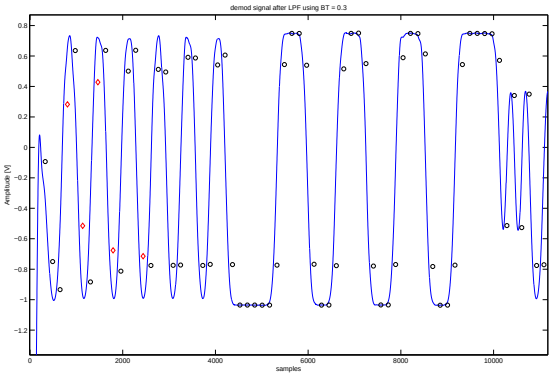


Figure 16: Demodulated signal after symbol synchronization using BT = 0.3.

the synchronizer displays circles on the graph to show that synchronization is achieved.

The simulation was performed using the 24-bit training sequence specified for AIS, an 8-bit start flag, and then randomly generated bits representing the message signal. The results shows the training sequence of alternating ones and zeros, followed by start bits and then the message data.
Figure 16 shows the demodulated signal when BT = 0.3 is used. The effect of the intersymbol-interference is displayed on the demodulated signal.

Figure 17 and Figure 18 shows the rectified signal and the recovered signal after early-late gate synchronizer in the presence of additive Gaussian noise. The degradation of the signal by the noise is evident in the rectified signal. This is shown by the amplitude variations of the rectified signal. This results in errors being introduced in the recovered bits.

Figure 19 shows the power density spectrum of the GMSK signal with frequency deviation of 12.5 kHz and BT = 0.5. The spectrum in Figure 19 was generated by windowing the GMSK signal by the Hamming window. The results show that the Gaussian filter helps to prevent the GMSK signal power from spilling into adjacent channels. The spectrum has lower side-lobes outside of the 25-kHz bandwidth, about 30dB below the

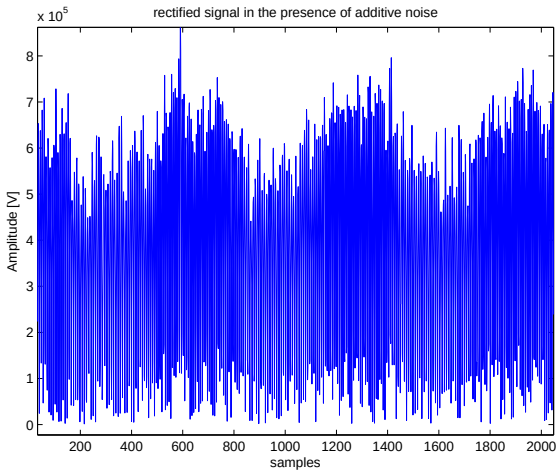


Figure 17: The rectified signal with AWGN at 10 dB SNR.

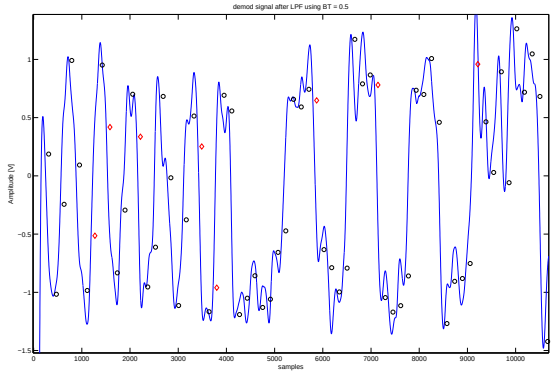


Figure 18: Demodulated baseband signal after symbol synchronization in the presence of AWGN.

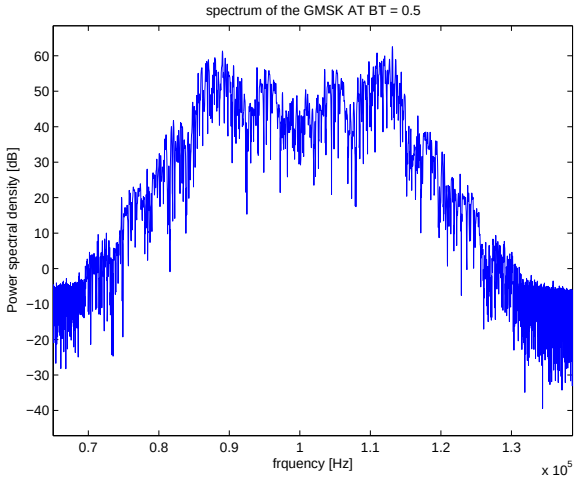


Figure 19: Frequency spectrum of a GMSK signal with a frequency deviation of 12.5 kHz and BT = 0.5.

main lobe. The maximum permissible adjacent-channel interference limit for radio transmissions is -60 dB [12]. According to [12], both GSM and Digital European Cordless Telephone (DECT) may not meet the recommended value of -60 dB, and for DECT, adjacent-channel interference of only -40 dB can be guaranteed.

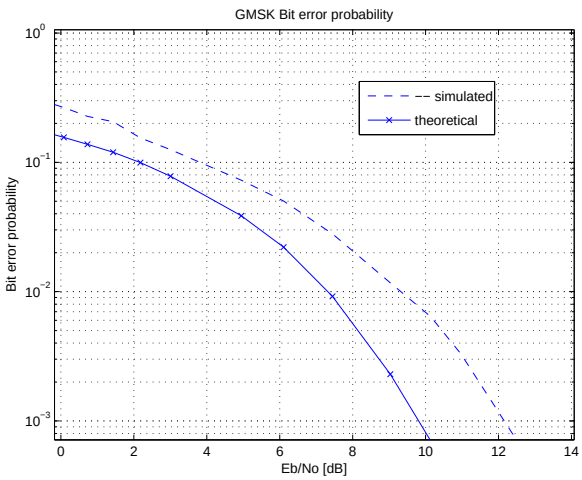


Figure 20: Bit error probability curve for GMSK.

Figure 20 shows the bit error probability performance of the discriminator receiver for GMSK in the presence of additive Gaussian noise using BT = 0.5. The simulated results differ from the theoretical results by about 2.5 dB.

7. Link Budget

A link budget for the uplink was calculated to verify that it is possible for a LEO satellite to reliably receive AIS transmissions. An AIS antenna height of 20 m was assumed. The transmission power of the AIS station is 12.5 kW at maximum. A 600 km orbit is used to compute the slant range of 2328 km with 5 degrees minimum ground station elevation [17]. Table 1 gives the link budget.

Table 1 Link budget, uplink performance

Item	*Symbol*	*Value*	*Unit*
Frequency	f	161.975	MHz
Transmitter power 12.5W	P_t	10.97	dB
Bit rate	R	9600	bps
Transmit Antenna Gain	G_t	5	dBi
Minimum elevation	$\in$	5	deg
Transmitter Line loss	L_t	-1.0	dB
Effective isotropic radiated power	EIRP	14.97	dB
Propagation path length	S	2328	km
Space path loss	Ls	143.9	dB
Satellite Rx Antenna Gain	G_r	-6.0	dBi
Receiver noise figure	F	6.0	dB
Receiver noise temperature	T_r	1000	K
Receiver IF bandwidth	B	20	kHz
Noise spectral density	N_O	-198.6	dBW/Hz
Minimum required RX SNR	SNR	10	dB
Other Losses	L_O	6	dB
Received signal power	P_r	-110.9	dBm
Received signal to noise ratio	SNR_r	14.69	dB
Link margin	***M***	**4.69**	**dB**

The link budget results shows a link margin of 4.69 dB, which is more than sufficient to receive the AIS signals on the satellite from ships at sea. The link budget was computed for the worst-case scenario.

8. Further Work on the SDR Implementation

Further development of the GMSK demodulator is scheduled to be implemented in SDR environment as from May 2006. The GMSK modulator and demodulator must be implemented using the C++ programming language. To verify the design, actual AIS signals will be transmitted and then demodulated using the software defined radio AIS receiver. An AIMSmIV AIS mobile station from Marine Data Solutions was acquired and will be used to transmit reference AIS signals.

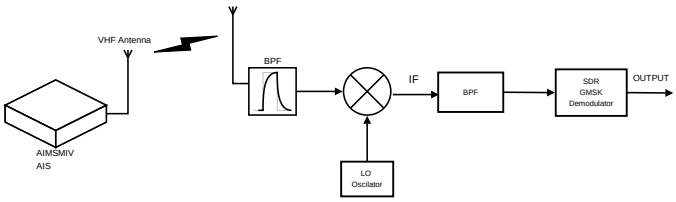


Figure 21: Block diagram of the experimental implementation of the SDR GMSK demodulator.

The proposed SDR implementation is shown in Figure 21.

9. Conclusions

In this paper we have presented the design and simulation of the proposed SDR GMSK modulator and demodulator, which is used in AIS. The GMSK demodulator is proposed to be used on the ZA-002 satellite to monitor maritime AIS signals. The GMSK demodulator was implemented as a software-defined radio, which allowed us to test much of its functionality by simulation. The simulation results showed that both the differentiation and rectification processes can successfully be performed in the digital domain by calculating a backward difference, and by taking the absolute value of the signal. The early late gate synchronizer performed as expected and achieved symbol synchronization on the demodulated signal within the training sequence of a received AIS signal.

In the presence of AWGN, the bit error probability performance of the GMSK discriminator showed 2.5 dB loss compared to theoretical results. The link budget for the uplink was calculated and a link margin of 4.69 dB was obtained, which demonstrated that sufficient AIS signal power will be received at the satellite's altitude.

There are several advantages to the demodulation taking place in the digital domain. First, the modulation scheme can be updated as new software techniques become available. It also means that there is a possibility for a radio system on mobile platforms such as satellites to be reconfigured to entirely new applications. This demonstrates the concept that it is possible to change the satellite mission completely while the satellite is already in orbit by performing a simple software update. The disadvantage may be the speed of processing that is needed in processing signals, especially at high intermediate frequencies, due to the limited sampling frequency of the ADCs.

The GMSK modem presented here is scheduled to be implemented on the SDR environment of the University of Stellenbosch, which will allow hardware tests to be performed. The actual AIS signals will be transmitted and then be demodulated by the proposed SDR demodulator.

Acknowledgments

The research in this paper was partially sponsored by Sun Space and Information Systems, and its submission was made possible by Prof. Jan du Plessis of the same company. The author wishes to extend his thanks to Marine Data Solutions for assisting with the AIS equipment for testing.

References

[1] SDR Forum, *SDR forum FAQs*, April 2006, http://www.sdrforum.org/faq.html

[2] IALA, "Operational issues", *IALA Guidelines No. 1028 on The Automatic Identification System (AIS),* vol. 1, part 1, edition 1.3, IALA, December 2004.

[3] U.S. Coastguard, *AIS Schematic*, April 2006, http://navcenter.org/enav/ais/IALA_AIS_ClassA_Schematic.pdf

[4] IALA, *Technical Clarification on Recommendation ITU-R.1371-1,* Edition 1.3, December 2002, http://navcenter.org/enav/ais/ITU-R_M1371-1_IALA_Tech_Note1.3.pdf

[5] Jeffrey D. Laster, *Robust GMSK Demodulation Using Demodulation Diversity and BER Estimation*. PhD dissertation, Virginia Polytechnic Institute and State University, March 1997.

[6] Rodger E. Ziemer and William H. Tranter*, Principles of Communications, Systems, Modulations and Noise*, Fifth Edition. John Wiley & Sons, Inc, 2002.

[7] Vadim Manassewitsch, *Frequency Synthesizers – Theory and Design*, Second Edition. New York: Wiley, 1980

[8] Johan J. Cronje, *Software Architecture Design of a Software Defined Radio System*, Master's thesis, University of Stellenbosch, 2004.

[9] Bernard Sklar, *Digital Communications, Fundamentals and Applications*, Prentice Hall International Editions, pp. 430-460, 1988.

[10] Mischa Schwartz, *Mobile Wireless communications*, Cambridge University Press, 2005.

[11] K. Sam Shanmugam, *Digital and Analog Communication Systems*, First edition, New York: John Wiley and Sons, 1979.

[12] Ashar Mehrottra, *Cellular Radio Performance Engineering*, Artech House, pp. 327-333, 1994.

[13] Walter Tuttlebee, *Software Defined Radio: Enabling Technologies,* First edition, New York: Wiley and Sons, 2002.

[14] R. Diana, M. Johnson and Tien M. Nguyen, "Bandwidth-Efficient Modulation Through Gaussian Minimum Shift Keying", *Space Communications*, vol. 3, no. 1, Winter 2001/2002.

[15] G-J van Rooyen, *The theory of frequency synthesis*, Lecture notes. University of Stellenbosch, 2002.

[16] Mike Rice and David G. Long, *Introduction to Analog and Digital Communication Theory*, Brigham Young University, 1994.

[17] Riaan Wolhuter, "*Link budget analysis notes*", University of Stellenbosch, 2006.

[18] *Digital Modulation and GMSK*, Hull University, January 2006, http://www.emc.york.ac.uk/reports/linkpcp/appD.pdf

[19] Simon Haykin, *An Introduction to Analog and Digital Communications*, New York: John Wiley and Sons, 1989, pp. 378-381.

[20] Ferrel G. Stremler, *Introduction to Communication Systems*, Third edition, Addison-Wesley, 1990, pp 641-650.

[21] G-J van Rooyen, *"Software-Defined Radio: An Intecnet Presentation",* Presentation slides. Stellenbosch University, 19 March 2006. http://www.amsatsa.org.za/intecnet_sdr.pdf.