B.Sc.Eng. Thesis
Bachelor of Science in Engineering

DTU Compute
Department of Applied Mathematics and Computer Science

Protocol analysis in a GSM network

using software defined radio with HackRF One

Martin Jesper Low Madsen
(s124320)

Kongens Lyngby 2015

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Matematiktorvet
Building 303B
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

The evolution of mobile communication technologies continue to expand to fulfill the demand for better network coverage and increased capacity, which in turn introduce many new ideas, designs and methods to learn and comprehend.

This thesis investigates the Um interface of the *global system for mobile communication*, GSM, technology by describing the design and implementation of a *software-defined radio*, SDR-driven protocol analyzer by processing digital complex samples from the radio link between a *base transceiver station*, BTS, and a *mobile station*, MS. These samples are provided by a HackRF One SDR and are transferred to a computer over a *universal serial bus*, USB, interface.

By demodulating the radio signal and estimating the digital information on the air interface, a synchronous connection can be established and any traffic within a desired GSM cell can be monitored. This enables the underlying protocols for call-, mobility- and radio control to be examined and analyzed.

The software implementation described in this thesis proposes a use in an educational aspect for learning and comprehension of protocols used in mobile communications.

Preface

This bachelor thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfillment of the requirements for acquiring a Bachelor of Science in Engineering degree in IT and Communications Technology. This thesis corresponds to 20 ECTS points.

The project has been supervised by Ph.D. student Maxwell Walter and Associate Professor Sven Karlsson during the period of February 2015 through June 2015.

Kongens Lyngby, July 1, 2015

[illegible]

Martin Jesper Low Madsen (s124320)

Acknowledgements

I would like to thank my two supervisors Maxwell Walter and Sven Karlsson for their feedback and ideas during our weekly Skype meetings, and again Maxwell for proofreading my thesis in the weeks up to the project deadline.

Contents

List of Figures

Acronyms

3GPP	3rd Generation Partnership Project
ADC	Analog-to-digital Converter
AGCH	Access Grant Channel
ARFCN	Absolute Radio Frequency Channel Number
BCC	Base Station Color Code
BCCH	Broadcast Common Channel
BPSK	Binary Phase Shift Keying
BSC	Base Station Controller
BSIC	Base Station Identity Code
BSS	Base Station Subsystem
BTS	Base Transceiver Station
CC	Call Control
CCCH	Common Control Channel
CR	Command/response
CRC	Cyclic Redundancy Check
DCCH	Dedicated Control Channel
EA	Extend Address
FACCH	Fast Associated Control Channel
FCCH	Frequency Correction Channel

FDMA	Frequency Division Multiple Access
FSM	Finite State Machine
GMSK	Gaussian Minimum Shift Keying
GPRS	General Packet Radio Service
GSM	Global System For Mobile Communication
GUI	Graphical User Interface
HLR	Home Location Register
HSPA	High Speed Packet Access
I	In-phase
ISDN	Integrated Services For Digital Network
ISI	Inter Symbol Interference
LAPD	Link Access Protocol, D Channel
LAPD$_m$	Link Access Protocol, Dm Channel (D-modified)
LPD	Link Protocol Discriminator
LTE	Long-term Evolution
MLSE	Maximum Likelihood Sequence Estimation
MM	Mobility Management
MS	Mobile Station
MSC	Mobile Switching Center
MSK	Minimum Shift Keying
NCC	Network Color Code
NSS	Network Switching Subsystem
OSI	Open Systems Interconnection
OSR	Oversampling Ratio

PCH	Paging Channel
PD	Protocol Discriminator
PDS	Packet Data Signalling
PSTN	Public Switched Telephone Network
Q	Quadrature
QoS	Quality Of Service
RACH	Random Access Channel
RFN	Reduced Frame Number
RR	Radio Resource
SACCH	Slow Associated Control Channel
SAPI	Service Access Point Identifier
SCH	Synchronization Channel
SDCCH	Standalone Dedicated Control Channel
SDR	Software-defined Radio
SIP	Session Initiation Protocol
SMS	Short Message Service
SS	Supplementary Service
TCH	Traffic Channel
TDMA	Time Division Multiple Access
TI	Transaction Identifier
TRX	Transceiver
UMTS	Universal Mobile Telecommunications System
USB	Universal Serial Bus
VLR	Visitor Location Register
VoIP	Voice Over Internet Protocol

CHAPTER 1

Introduction

The *global system for mobile communication*, GSM, has grown to be the leading mobile communication standard in the world since first introduced in 1982. GSM was initially developed as a wireless alternative to the *public switched telephone network*, PSTN, which allowed intercommunication in both technologies. GSM was later developed to support packet switching as a *general packet radio service*, GPRS, meaning it had access to the Internet. Circuit- and packet switched networks share a stack of protocols used on the Um air interface between a *base transceiver station*, BTS, and the mobile phone. One of such protocols is the *link access protocol, Dm channel (D-modified)*, LAPD$_m$, used in the data link layer.

Since the air interface is an open medium, it is possible to listen in on communication between a mobile phone and the base station. Using a *software-defined radio*, SDR, some GSM communication can be decoded by implementing the freely available GSM standards through software as specified by the *3rd generation partnership project*, 3GPP. For security reasons, dedicated user data that are exchanged are encrypted with an A/5.1 stream cipher, but unfortunately, such data can be decrypted quickly [Bar+03]. Newer, faster and more secure technologies exist today to counter this vulnerability, for example *universal mobile telecommunications system*, UMTS, *high speed packet access*, HSPA, and *long-term evolution*, LTE. While LTE and newer technologies are slowly getting deployed globally to fulfill customer needs of *quality of service*, QoS, GSM remains the leading standard in mobile coverage for now.

An SDR is built with the purpose of handling radio signals in software. This is done by having SDR-hardware sample radio signals and forward those samples to host memory of a computer. HackRF One is such a peripheral that forwards samples over a USB. The USB interface also allows the HackRF to be configured by software. Examples of such configurable parameters are tune frequency, sample rate and gain. Its USB interface is operated using the HackRF C library, libhackrf.

The prime motivation of this thesis, is to contribute a tool which can be used for education purposes in understanding of- and learning about mobile communications. SDR provides the means to make this possible on a low budget and the software behind SDR, makes it a flexible tool for many applications. GSM is one example.

In this report, I examine the GSM technology and describe my design and implementation of a mobile network protocol analyzer piece of software, in terms of displaying mobile communication in real time, using a HackRF One, an SDR. The protocol analyzer software is developed with several mobile communications technologies in mind for easy extension, but currently only supports GSM. It synchronizes with a near base station and decodes data sent over the air in the downlink direction.

1.1 Thesis outline

The thesis is outlined as briefly described in the following.

» Chapter 2 dives into alternative solutions and related work that accomplish the same functionality as the design and implementation for parsing GSM traffic proposed in this thesis.

» The infrastructure behind GSM is walked through in Chapter 3 starting with a wide perspective of the infrastructure that then digs into the Um interface between the BTS and the mobile entity, its design and its LAPD$_m$ protocols.

» By further describing the Um interface, Chapter 4 looks into the theory behind the processing of signals on this interface. The chapter briefly describes sampling of a signal and modulation schemes used on the Um interface.

» Chapter 5 takes the two previous chapters and describes SDR in general in respect to mobile communications with focus on GSM.

» Chapter 6 describes my ideas for- and design of the mobile network protocol analyzer with the SDR interaction and each operation following the flow of samples towards the parsing element in the application.

» The specifics of the implementation are introduced in Chapter 7 which follows a similar flow of Chapter 6.

» Chapter 8 evaluates my implementation of the protocol analyzer and a few examples of GSM broadcast traffic content are tested on a 1800MHz GSM band.

» Some improvements and ideas that didn't make it to the current implementation, but may be implemented in the future are discussed in Chapter 9.

» Chapter 10 evaluates and discusses my process throughout the project.

» Finally, Chapter 11 concludes on the project.

CHAPTER 2

Related Work

In this chapter, we take a brief look at other existing software defined radio solutions and projects that have inspired my process, design and implemention of this project.

» GNU Radio is a free and open-source software development toolkit that provides signal processing blocks to implement software radios. It can be used with readily-available low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment. It is widely used in hobbyist, academic and commercial environments to support both wireless communications research and real-world radio systems [org].

» AirProbe is a GSM-sniffer project that provides tools to parse GSM communication using SDR. Its main module interfaces with GNU Radio from which the tuning frequency and gain can be changed through a *graphical user interface*, GUI [KSW].

» gr-gsm is an actively developed project based on the GSM receiver module from AirProbe. It provides the same functionality as AirProbe, thus decodes GSM communication using GNU Radio data structures and graphical interface [Kry].

» Kalibrate is a utility, originally developed by Joshua Lackey, that scans GSM bands for base transceiver stations by searching for a frequency correction burst that is transmitted periodically by the BTS. It uses the algorithm as described in the paper *Robust Frequency Burst Detection Algorithm for GSM / GPRS* by G. Narendra Varma, Usha Sahu and G. Prabhu Charan to find a pure sinusoid of all zero bits within a GSM burst [Lac].

» OpenBSC is an open source Linux software application that implements GSM telecommunication protocols and functions as a software-based mobile backhaul entity using a software defined radio as transceiver peripheral [gro].

» OpenBTS is similar to OpenBSC, but is not limited to GSM protocols and it supports mobile devices presented as *session initiation protocol*, SIP, endpoints which makes *voice over internet protocol*, VoIP, possible [Net].

» gr-LTE provides a similar approach of gr-gsm by demodulating a signal and then synchronizing to it, but for LTE networks [DM].

CHAPTER 3

Global System for Mobile Communications

This chapter describes the fundamental parts of GSM with prime focus on explaining communication over the Um interface, which we will look closer at in Section 3.2. The infrastructure of GSM is usually pictured as a tree as shown in Figure 3.1 where the *mobile switching center*, MSC, with connection to the PSTN, is the trunk and the *base station controllers*, BSCs, and BTSs, make branches. The MSC is a backhaul part of the core of GSM known as the *network switching subsystem*, NSS, which links to the PSTN and serves functionality such as mobile registration, mobility management, call control and more.

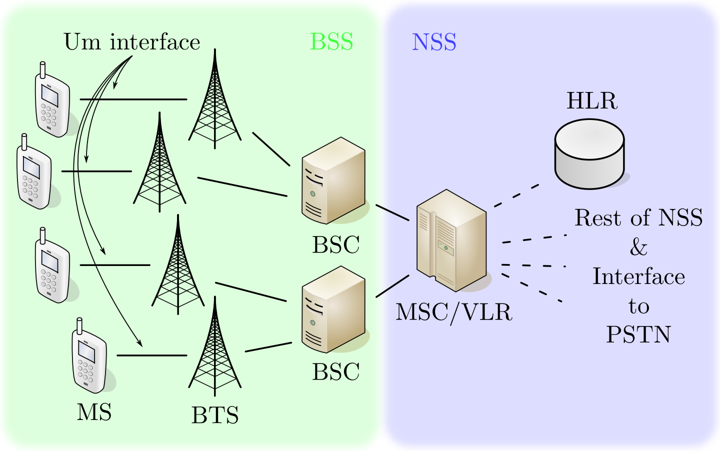


Figure 3.1: The infrastructure of the outermost part of GSM combined by the BSS and NSS with *mobile stations*, MSs, connected to the BTSs [Sau11, p. 11, 21–44].

The *home location register*, HLR, and *visitor location register*, VLR, are databases storing records of authenticated subscribers e.g. mobile phones where the HLR stores each subscriber and VLR stores only subscribers in coverage of the corresponding MSC. When a subscriber moves into a different coverage area, the new MSC in this area fetches the record from the HLR and stores it locally in its VLR. The record in the old MSC is then deleted, and by storing these records locally in the VLR, the load is reduced on the shared HLR database [Sau11, p. 13-17]. Subscriber registration and mobility management is done by the MSC which also handles call establishment and call routing between two users [Sau11, p. 11-13] and by digging further into the *base station subsystem*, BSS, there is the BSC that controls medium allocation on the Um interface, smooth handover to a different base station while roaming, time advance in transmission and signal power control and thus by combining these two entities a link between two subscribers can be established and maintained [Sau11, p. 30-34]. A BSC controls a set of BTSs which are directly linked to the Um interface.

3.1 Base Transceiver Station

The BTS, links directly to the Um interface and it ensures a radio link from the mobile phone to the backhaul network. The BTS covers an area wirelessly known as a cell in which mobile devices must remain to ensure coverage and access to mobile services. As stated above, the BSC enables extra flexibility and hands over mobile devices to a different BTS if one were to move out of coverage, however, this means that the new area must be covered by another BTS. By measuring signal strength, the BSC can determine when the best time is to perform a handover to a different BTS.

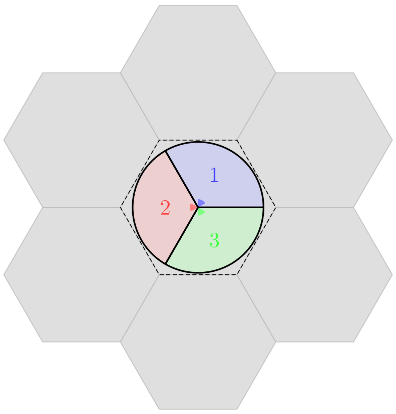


Figure 3.2: A GSM cell sectorized into three areas that make a part of a cellular structure.

BTSs are positioned such that they cover a sector within a cell, commonly these sectors make a 120° degree angle thus a cell is divided into two or three sectors as seen in Figure 3.2, and to avoid interference, each sector operates at different frequencies. Dividing a cell into three sectors enables a better reuse of frequencies without interferring with neighbor sectors. These areas vary much in size because capacity is limited, but generally they can vary from $35km$ down to $100m$ in distance. $35km$ is only theoretical and due to the required signal power these areas are much smaller in practice, 3 to $4km$ is common in residential and business areas [Sau11, p. 23].

3.2 Um air interface

Within a sector, the BTS transmits in a frequency spectrum licensed to commercial operators and depending on what country, these spectra differs. The GSM standard specifies a couple of different frequency bands; e.g. 850MHz, 900MHz primary, 900MHz extended, 900MHz railway-specific, 1800MHz and 1900MHz. In Denmark, 900MHz and 1800MHz are reserved for GSM [Erh]. In each of these bands there are channels with a width of 200kHz that can be assigned to a specific sector. For example, this leaves 124 channels in the 900MHz band and each of these channels are identified by a number, the *absolute radio frequency channel number*, ARFCN [Sau11, p. 21–22]. The channel number can be calculated to- and from the corresponding carrier frequency for 900MHz primary with the equations given in (3.1) [3GPg, p. 10].

$$\begin{aligned} f_{ul} &= 900\text{MHz} + 0.2\text{MHz} \cdot n \qquad \text{for } 1 \leq n \leq 124 \\ f_{dl} &= f_{ul} + 45\text{MHz} \end{aligned} \tag{3.1}$$

f_{ul} and f_{dl} refer to carrier frequencies for uplink and downlink, respectively, in a given channel n. The equation is similar for other frequency bands where 900MHz in the equation refers to the band, 0.2MHz remains the same for all bands, but n may need to be offset in certain cases and ARFCN, values are abnormal thus different in all bands. The 45MHz offset for the downlink expression vary with the number of channels in a specific band. The specifics are defined in the standard [3GPg, p. 10].

3.2.1 Multiplexing

As just introduced, GSM defines a set of channels given by the ARFCN number, each of these numbers correspond to a carrier frequency with a bandwidth of 200kHz. A BTS may communicate with multiple users either at different frequencies or by taking turn rapidly for each user. The first method is called *frequency division multiple access*, FDMA, which is described briefly next and the other method is called *time division multiple access*, TDMA.

3.2.1.1 Frequency Division Multiple Access

Given the 124 channels in the 900MHz frequency band, a BTS is not limited to just three channels for each of its sectors. By installing multiple *transceivers*, TRXs, a single sector can serve an increased number of users at the same time and each of these TRXs operate at a uniquely assigned frequency within the band. Communication is thus multiplexed using FDMA both within a single sector and across the whole cell.

3.2.1.2 Time Division Multiple Access

Apart from frequency multiplexing, GSM also divides communication in time. Such a division results in a TDMA frame that consists of eight timeslots each of which can transfer a so-called burst with a duration of 577μs. Figure 3.3 shows the structure of a TDMA frame with two timeslots carrying data. Timeslot 0 and 1 on the main frequency is mostly used for BTS signalling and by listening to this timeslot a mobile device can determine if the corresponding frequency is currently used by a BTS.

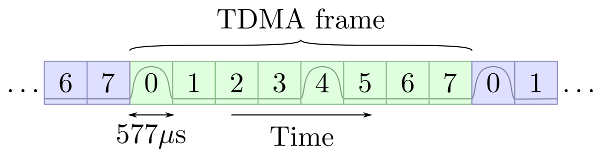


Figure 3.3: A TDMA frame consisting of eight timeslots numbered from 0 to 7 with timeslot 0 and 4 carrying a burst.

A subscriber is allocated a single timeslot in the up- and downlink and it is only allowed to send at that point in time. Such a timeslot allocation can be thought as a physical channel. Transmission and reception therefore occurs at an interval of 4.6ms, the width of a TDMA frame.

The information carried in each burst depends on what type of information is transmitted. The burst bit configuration differs for broadcast signals, traffic communication and direct signalling to mobile devices. The different types of bit configurations are; a normal burst, frequency correction, synchronization, dummy and an access burst as seen in Figure 3.4. The normal burst is used for voice traffic, both transmission and reception and it carries encrypted sectionized bits in a pair. Inbetween this pair, a training sequence bit pattern is transmitted to compensate for interference caused. Two stealing bits are transmitted before and after the training sequence and they are used to indicate if either of the sections of encrypted bits are stolen for urgent signalling. At the beginning and end of a burst, a pattern of bits, tail bits, are transmitted to help detect these transitions and before and after these patterns a brief guard time period is present to avoid overlapping of neighbor timeslots. The frequency correction burst is transmitted by the BTS and is used by the mobile device to synchronize its local oscillator to the clock frequency of the BTS. The BTS also transmits a synchronization burst which is used to time synchronize to a TDMA frame number (see Section 3.2.2). A dummy burst is transmitted in all timeslots by the BTS if they aren't already occupied and it is used to measure signal strength at the receiver which then transmits the result back to the BTS and the BSC. The BSC then determines if a handover would be sufficient. The access burst is used to correctly control burst timing for extra ensurance of collision-free bursts. Timing advance depends on the distance between the subscriber and the BTS thus this must be adjusted for bursts to arrive at either end at the right moment in time [Sau11, p. 25–26].

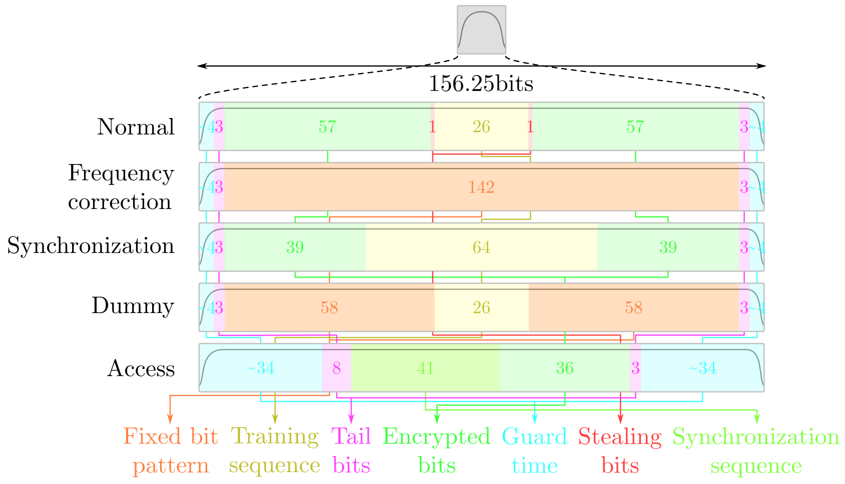


Figure 3.4: Five different types of GSM bursts each with a different bit configuration [3GPf, p. 19–31].

All these timeslots, or physical channels as they can also be thought as, are arranged into logical channels, which accomplish some of the functions as mentioned above for each type of burst. Not all data is necessary to transmit every TDMA frame, say a frequency correction burst or a synchronization burst thus these types of different data are transmitted at a greater interval. Depending on whether this data is a broadcast message, some exchanged information to an idle mobile device or a speech channel, the interval can vary differently. The GSM standard defines two types of groups of TDMA frames for this purpose, the 26 multiframe and the 51 multiframe. These two groups of frames are repeated infinitely and their content is described below [Sau11, p. 27].

3.2.2 26- and 51 multiframe

The synchronization burst enables synchronization to a specific TDMA frame number which is the number of a TDMA frame in either the 26 frame or the 51 frame. The synchronization burst is repeated every tenth TDMA frame in timeslot 0 on the main carrier frequency and is part of the 51 frame. The synchronization burst is mapped onto the logical *synchronization channel*, SCH, which makes five timeslots out of 51. Apart from the SCH, frequency correction also takes place in the 51 frame together with BTS broadcast information, paging of a mobile device, timeslot allocation etc. and all these are mapped in a specific order and to a specific frame number in the 51 frame.

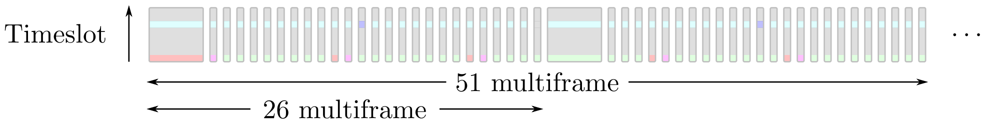


Figure 3.5: A bundle of frames represented as the 26- and 51 multiframe where light blue color represents subscriber traffic, dark blue represents signal measurements, red represents frequency correction, purple represents synchronization and the light green represents other types of information such as paging information, location updates, timeslot allocation requests etc.

The 26 frame is used for traffic and the timeslot allocation remains to an interval of one TDMA frame until the timeslot in frame number 12 which, instead of data, is used to transmit signal quality measurements and timing advance control information. For both the 26- and 51 frame some frames are idle and no information is transfered in this time period. Frame number 25 is idle in the 26 frame thus an on-going call suffers two very brief moments of interruptions twice in every 26 frame, but the interruptions are not audible to the user. Similar situations occur in the 51 multiframe as seen on Figure 3.5 [Sau11, p. 26–28]. These categorizations of channels are explained next.

3.2.3 Channels

As with the SCH, other types of information are mapped onto a logical channel. The logical signalling channels are described as follows[Sau11, p. 26–29]:

» **FCCH**: The frequency correction burst is transmitted every tenth TDMA frame on timeslot 0 called *frequency correction channel*, FCCH and is used by the mobile device to detect and correctly tune in to the frequency of the BTS. Its local oscillator becomes synchronized with the BTS clock frequency by searching for a unmodulated sine wave carrier at an offset of 67.7kHz [3GPf].

» **SCH**: During synchronization, synchronization bursts on the SCH channel are used to identify the BTS and synchronize to its TDMA frame clock frequency.

» **BCCH**: The *broadcast common channel*, BCCH, carries the main information about the current cell and neighbor cells and mobile devices monitor this channel while idle.

» **RACH**: *random access channel*, RACH is a common channel used by all mobile devices for requests of mobile originated calls and sending and fetching of *short message service*, SMS, messages.

» **AGCH**: During call establishment, the mobile device must request a channel which the BTS grants access to by either allocating a *standalone dedicated control channel*, SDCCH, or in exceptional cases, a *traffic channel*, TCH. The *access grant channel*, AGCH, is used for this purpose and is carried by the RACH.

» **PCH**: When an incoming call is attempted, mobile devices are paged on the *paging channel*, PCH, in the location area on all BTSs belonging to the BSC controlling this area. The mobile device is aware of its own identity and the paging message contains the specific mobile device's identity thus all devices know if the paging message belong to them individually.

» **SDCCH**: The dedicated control channel is a pure signalling channel used during call establishment before being assigned a TCH, transmission and reception of SMS messages and location area update procedures.

» **SACCH**: The *slow associated control channel*, SACCH, is used to transmit radio signal measurements and timing advance in the 26 frame as mentioned above. The network may perform handovers to other cells based on the mobile device's feedback if necessary.

» **FACCH**: In contrast to SACCH, this channel is a fast alternative, used in urgent signalling and it steals a data section in a normal burst. The abbreviation comes from *fast associated control channel*, FACCH.

All these different channels are further categorized into one of two types, a broadcast, a *common control channel*, CCCH or a *dedicated control channel*, DCCH. Thus, a channel can either be dedicated to one user or be broadcast to a common group of users. In this sense, a broadcast- and a common channel are similar.

3.2.4 Channel coding

Since the air interface is prone to transmission errors because of sudden frequent changes in the radio environment, some steps are taken to counter these errors, namely the methods: block coding, convolution coding and interleaving, in hope for a successful recreation of data due to lost bits.

3.2.4.1 Block coding

The block coding part, e.g. for speech frames, extracts a sequence of 260 bits into two chunks referred to as class 1, 182 bits, and class 2, 78. The prior chunk is further split into two subclasses 1a of 50 bits and 1b of 132 bits. The 1a subclass adds three bits of *cyclic redundancy check*, CRC, and this new chunk is concatenated with 1b and four filler bits. The concatenation is then protected by the convolutional code and the result is prepended with the unprotected chunk, class 2. The filler bits appended to class 1b ensures the final size of the convolved bits to fit in four bursts, 456 bits [3GPa, p. 17]. Frames sent on broadcast channels are convolved for the whole sequence of bits. Their input is 184 bits and a CRC parity check of 40 bits and 4 filler bits are added. After convolution, the final size is also stretched over four bursts.

3.2.4.2 Convolution coding

The class 1 data is convolved with a convolution code with a code rate of 1/2 such that two coded bits correspond to a single bit of data. Lowering the code rate gives better correction potential in turn for less actual data being sent. GSM specifies eight different polynomials used for convolving the data bits. For the full-rate TCH, data bits at index i are convolved using the two polynomials

$$\begin{aligned} C_0\left(D\right) &= 1 + D^3 + D^4 \\ &\Downarrow \\ C_0\left(D_i\right) &= D_i + D_{i-3} + D_{i-4} \mod 2, \\ &\text{and} \\ C_1\left(D\right) &= 1 + D + D^3 + D^4 \\ &\Downarrow \\ C_1\left(D_i\right) &= D_i + D_{i-1} + D_{i-3} + D_{i-4} \mod 2. \end{aligned} \tag{3.2}$$

A sequence of encoded bits are then made of code words $W_c = (C_0, C_1)$ where each word represents a single data bit. To decode this again, a common way is to perform the Viterbi algorithm based on the Hamming distance as path metrics.

3.2.4.3 Viterbi algorithm

The Viterbi algorithm is an algorithm to find the most likely sequence of hidden states. In the case of decoding, the hidden states are data bits that are derived from the code words, W_c. The number of states correspond to $2^{|W_c|}$ where each state represent the bit pattern of a code word. Figure 3.6 illustrates possible transition decisions and the Hamming distance is used to calculate the branch metric. The Hamming distance is the number of changing bits of the input compared to the possible transitions in each state. For example, starting in state S_0 with an input of a $(1, 1)$ code word, the goal is to minimize the total Hamming distance from t_0 to t_n, where t_n is the time of the last data bit. The smallest metric from t_0 to t_1 is zero since there are no bits that differ between the code word $(1, 1)$ and the $(1, 1)$ transition. In contrast, the code word, $(1, 1)$, has a metric distance to $(0, 0)$ of length two [VO79, p. 235–239].

Figure 3.7 illustrates the finite state machine of Figure 3.6 over time. For each change of state, a binary decision is made and once the algorithm terminates, the state tree is traced back and the binary sequence with the smallest total metric from t_0 to t_n is the estimated decoded data.

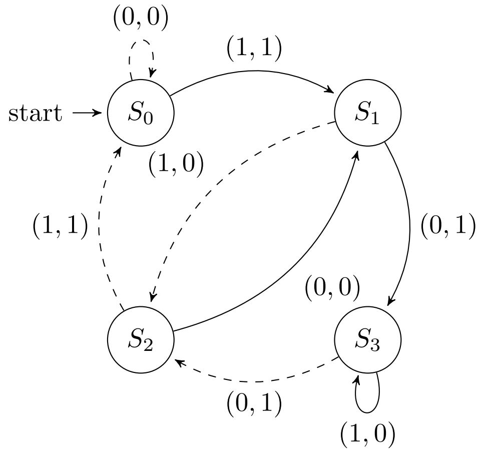


Figure 3.6: A *finite state machine*, FSM, of the Viterbi decode algorithm. The look of line transitions determine the output where dashed lines represent an output of 0 and normal lines an output of 1 at time t.

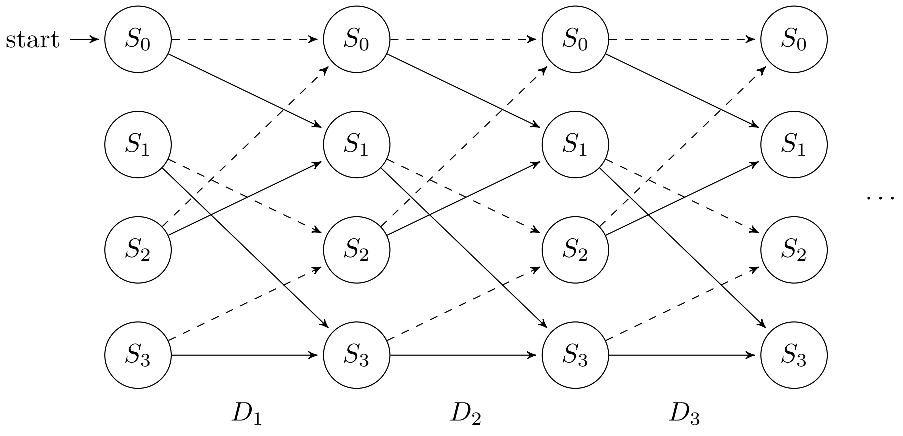


Figure 3.7: A trellis graph of the Viterbi finite state machine algorithm over time, t. D_1, D_2, D_3 and onwards are estimations of decoded data bits based on the total Hamming distance metric from t_0 to t_n. A dashed line represent a binary 0 and a normal line a binary 1.

3.2.4.4 Interleaving

As an extra step to reduce the impacts of the air interface, most data, with a few exceptions, is further processed and bits are interleaved. Data corruption often happen within a short time interval thus by interleaving the bits, just a few bits are lost in a grouped sequence and the chance of recoverable data more becomes probable.

In the interleaving process, 57 bits, a half burst, are paired with the forth following group of the same size into a single timeslot such that sequential data of 57 bits are split into multiple timeslots [Hei98, p. 100–101, 358].

3.3 Protocol stack

The TDMA and FDMA multiplexing covers the bottom of the GSM protocol stack, the physical layer and exactly what data bursts carry in layers above is described in this section.

The LAPD$_m$ procotol derives from *link access protocol, D channel*, LAPD in *integrated services for digital network*, ISDN, with the same purpose of ensuring error free messaging between two points and that these messages are executed in the right sequence. LAPD$_m$ is slightly modified to deal with the limited resources and peculiarities of the radio link [Hei98, p. 101–102] and it makes the data link layer in reference to the *open systems interconnection*, OSI, model's second layer. The third OSI network layer is provided by *radio resource*, RR, management, *mobility management*, MM, and *call control*, CC, GSM protocols as presented in Figure 3.8.

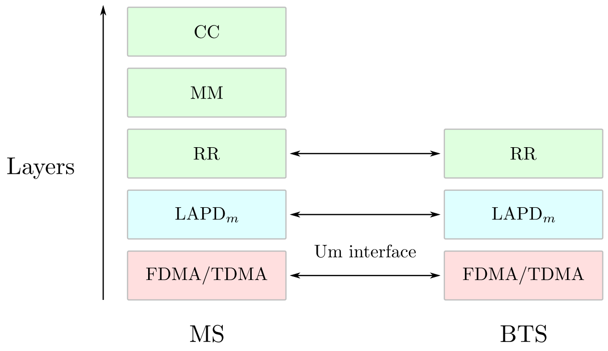


Figure 3.8: The GSM protocol stack of the mobile device and the BTS. Data works it way up towards the CC layer from the physical FDMA/TDMA layer, through the LAPD$_m$ data link layer, RR management and MM network layers [Hei98, p. 108].

The network layers manage individual procedures when it comes to establishment, maintenance and releasement of radio resources, support for mobility and identity and establishment, clearing and information exchanging of call state procedures [3GPb,

p. 37–38, 91–92, 176]. These network protocols are described more in detail in Section 3.3.2 and details of LAPD$_m$ is presented next.

3.3.1 Link Access Protocol, Dm channel

LAPD$_m$ ensures sequential order of frames across data link connections and provides flow control of such frames. It also detects errors in frames and notifies these to the network layer entity which it recognizes from the frame type.

LAPD$_m$ exchanges conform to six different types of formats:

» *Format A*: the A-format is used for frames without an information field and it is sent on DCCHs, similar to the following formats;

» *Format B/Bter/B4*: data dedicated to a single user uses the B-format for its frame and it is used for actual signalling data. Bter and B4 are used for unacknowledged data transmission.

» *Format Bbis*: a frame in the Bbis-format is used on BCCH, PCH, and AGCH. It has no address information since the sent information is broadcast to all mobile devices in which case these are referred to as CCCHs.

» *Format C*: A sixth C-format exists and it is used for transmission of random access signals [3GPd, p. 9–12].

3.3.1.1 Frame structure

The LAPD$_m$ frame formats consists of an address-, control- and length indicator field except for the Bbis-format which only carries network layer information. The B-format also includes an information field used by third layer protocols. Both the B- and A-format come with extra fill bits indicated by the length field as seen on Figure 3.9.

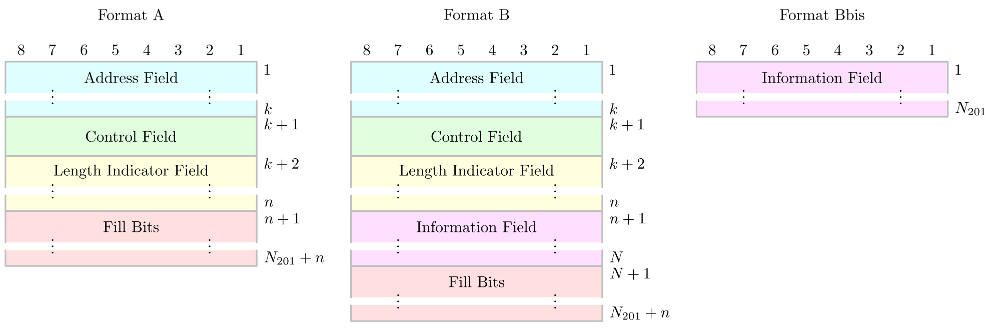


Figure 3.9: Three examples of LAPDm frame formats consisting of a number of fields with varying size of k, n and N octets, up to a channel specific size parameter, N_{201}.

Each field, if present, works as described in the following [3GPd]:

» Address field: The address field consists of an *extend address*, EA, bit, a *command/response*, CR, bit, a *service access point identifier*, SAPI, field and a *link protocol discriminator*, LPD. The EA bit determines if the address field is extended to one more octet which may hold additional address info.

  The CR bit tells the recipient whether the data is a command or a response seen from the BTS's perspective. Data sent from the mobile device use the inverted value to inform if it's a command or a response.

  The SAPI field is used to determine the service of the payload such as RR, MM, CC or even SMS data.

  The LPD field is reserved and should not be set.

» Control field: This field conform to three different formats, the *information transfer*, I-format, the *supervisory*, S-format and the *unnumbered*, U-format. They control which action to take based on the CR bit from the address field such as acknowledge or unacknowledge certain received information and even request retransmission of information. The S-format supervises and is used to control the next move such as acknowledge an I-frame, request retransmission etc., while the I-format is used to transfer information between network layer entities. The U-format may be used to provide additional control functions.

» Information field: this field is the payload of the network layer protocols and is not processed.

3.3.2 Network layer

The third layer consists of three different protocols for network management and these are individually identified by a *protocol discriminator*, PD, and their transaction origin by a *transaction identifier*, TI, value. The frame format is mostly identical for all three protocols and they all contain the identification, message type and the parameters assigned to that specific message type.

Apart from origin, the TI value also distinguishes messages within a transaction, however only the CC protocol allow several simultaneously transactions. The mobile device and the NSS may use the same TI values set by the three least significant bits since their origin differ and the TI's most significant bit identifies the origin [3GPc, p. 87].

3.3.2.1 Radio resource management

Procedures of radio resource management are necessary to establish, maintain and release RR data link connections between the mobile device and the network. These common transmission resources are partly managed and processed on both ends on the Um interface, but also further in the network e.g. at the MSC. This allows further procedures like cell selection and handover procedures.

Radio resource management is assigned the PD, 06 [Hei98, p. 107].

The RR procedures manage the logical and physical channels which are then processed in the remaining network layer protocols [3GPb, p. 37–38].

3.3.2.2 Mobility management

Unlike RR management, generally, neither MM nor CC, with a few exceptions, are involved at the BTS and both of these protocols are managed by the NSS and the mobile device. This allows transparent information exchange on logical channels provided by the RR protocol.

The functions of mobility management procedures are mainly location area updates and providing user identity confidentiality [3GPb, p. 91].

Mobility management is assigned the PD, 05 [Hei98, p. 107].

3.3.2.3 Call control

The CC protocol is an application that consists of entities that make large communicating finite state machines which controls functions such as call establishment procedures, call clearing procedures and call information phase procedures etc. [3GPb, p. 176].

Call control and supplementary services are assigned the PD, 03 [Hei98, p. 107].

CHAPTER 4

Signal processing

In this chapter, we look at the basics of signal processing and how it relates to signal modulation, demodulation and decoding of information in GSM communication. Such signals on the Um air interface in GSM are quadrature modulated using *gaussian minimum shift keying*, GMSK. In this type of modulation, bits are mapped to symbols, pulse shaped by a gaussian filter and carried pairwise by two orthogonal signals, 90° degrees apart. These signals are combined into a final signal, $v(t)$, ready to be transmitted. Modulation schemes can carry symbol information by shifts in frequency, phase and amplitude. The GMSK modulation scheme changes its phase in a continuous matter which reduces the required bandwidth compared to an abrupt change as in the phase shift keying modulation scheme.

It follows from Euler's equation that a complex number in polar form is dependent on the phase angle, ϕ, and can be represented as the sum of cosine- and sine functions of phase angle, where cosine is the real part and sine is the imaginary part,

$$e^{i\phi} = \cos(\phi) + i\sin(\phi), \quad \phi \in \mathbb{R}. \tag{4.1}$$

Figure 4.1 illustrates a so-called harmonic phasor of a signal, a complex number in polar form, and its real and imaginary parts which are made of two real orthogonal signals as a cosine- and a sine function. The amplitude of these two individual signals are referred to as a complex signal's I-, and Q-, components and they are important quantities in quadrature modulation. In fact, the modulated final signal, $v(t)$, can be identified as a complex signal made of its two orthogonal signal components. Hence the name quadrature modulation.

In modulation schemes, a number of bits are mapped to one symbol e.g. *binary phase shift keying*, BPSK, and G/MSK map just a single bit and an element in such a binary data sequence of length k at index i can be described as

$$d_i \in \{0,1\}^k. \tag{4.2}$$

In the same sense, symbols D_i can be mapped directly to elements in the binary data sequence as

$$D_i = \begin{cases} +1 & \text{if } d_i = 1, \\ -1 & \text{if } d_i = 0, \end{cases} \quad i = 1, \ldots, k. \tag{4.3}$$

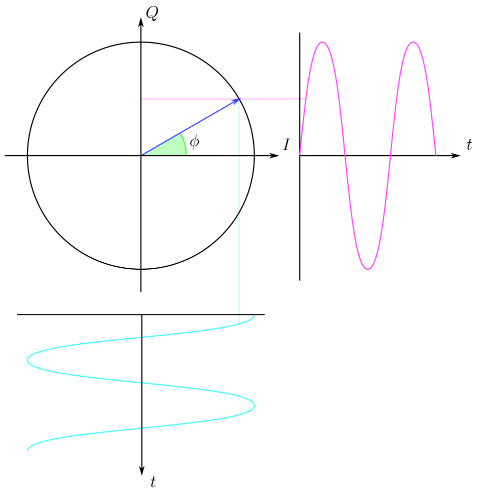


Figure 4.1: I/Q phasor plot of a complex signal made of two real signals, 90° degrees apart.

Symbols are used to better recognize zero values, at which a regular digital square wave has no zero crossings. It is difficult in the analysis of an analog signal to distinguish between a sequence of digital zeros and no transmission at all [CS03, p. 157].

The symbol mapping is one step towards the modulation of a signal.

4.1 Modulation

Modulation comes in many shapes and its purpose is typically to carry information in an analog signal by varying the signal's phase, frequency or amplitude. To fulfill these requirements a radio system must convert the information to a format that can be carried by radio waves. In GSM, the information is given by a binary sequence, mapped to symbols, which must be modulated such that the receiver is able to translate and receive the original message.

A square wave that crosses zero is still not optimal, since such a signal requires high frequencies to shape its steep transitions. Its Fourier transform results in a sinc function in the frequency domain that is unlimited in bandwidth. The binary data must therefore be shaped into a different signal to limit the bandwidth. This process is called *pulse shaping* where a digital signal is convolved into an analog signal through a pulse shaping filter. The sinc can be applied to such convolution of a digital signal which results in a very narrow and sharp bandwidth of low frequencies. In communication systems, this phenomena is ideal since it increases the amount of carriers which equals more channels in an FDMA aspect.

Another important situation that must be considered when choosing the pulse shape filter, is how a sequence of symbols will be transmitted continuously. The sinc function convolved with the symbols over a time interval cause adjacent symbols to overlap. This situation is called *inter symbol interference*, ISI, and it makes symbols difficult to restore. For this reason, the type of filter is chosen based on the specific application.

With two streams of pulse-shaped symbols in quadrature modulation, the modulated signal, $v(t)$, can be expressed as

$$v(t) = m_1(t)\sqrt{2}\cos(2\pi f_c t) - m_2(t)\sqrt{2}\sin(2\pi f_c t) \tag{4.4}$$

where

m_1, m_2	are the message streams of pulse-shaped symbols and
f_c	is the carrier frequency [CS03, p. 102].

(4.4) is illustrated in Figure 4.2.

4.1.1 Gaussian minimum shift keying

GSM uses the GMSK modulation scheme that is a combination of *minimum shift keying*, MSK modulation and a gaussian filter used for shaping the symbol pulse. Both GMSK and MSK avoid abrupt changes in the signal by smoothly changing the phase over time for each transmitted symbol. This behavior is known as a continuous phase scheme and it gives no discontinuity in the signal and reduces the required bandwidth. Since the information is carried in phase, the symbols are redefined and further mapped to constellation points, as illustrated in Figure 4.3, in a complex plane

$$S_i = j \cdot D_i \cdot D_{i-1} \cdot S_{i-1}. \tag{4.5}$$

S_0 is initially set to 1 unless specified otherwise and the new mappings are later used for estimation of a symbol sequence.

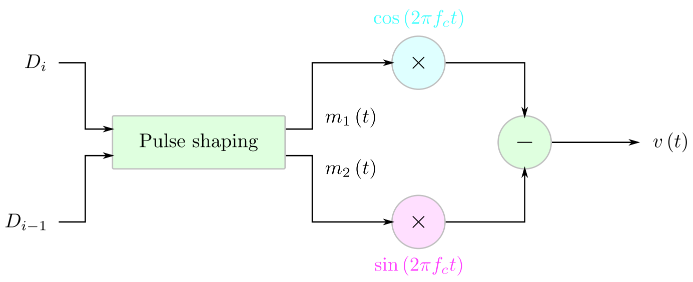


Figure 4.2: Digital to analog signal conversion with quadrature modulation.

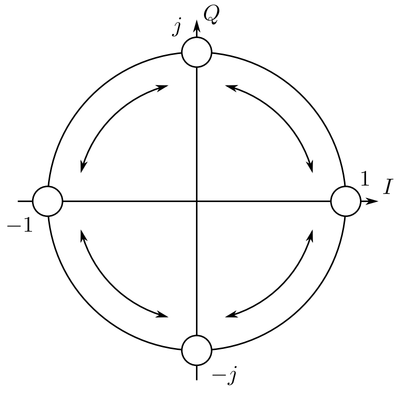


Figure 4.3: Symbols mapped to $\pi/2$ rotations over time as expressed by (4.5).

The gaussian filter shapes the symbols to stretch slightly longer in time, but it further reduces the signal's bandwidth. In fact, its time response is infinite and it is therefore truncated.

The GSM standard specifies a pulse shape of the gaussian function, $h(t)$, in (4.6) with $BT = 0.3$. T is the symbol period at 3.69μs and B refers to the bandwidth of gaussian filter. By increasing the bandwidth, B, with T declared constant, the shape begins to relate to MSK [3GPe, p. 5–7].

$$\begin{aligned} \delta &= \frac{\sqrt{\ln(2)}}{2\pi BT} \\ h(t) &= \frac{1}{\sqrt{2\pi}\delta T} e^{-t^2/\left(2\delta^2 T^2\right)}. \end{aligned} \tag{4.6}$$

As Figure 4.4 illustrates, the gaussian function's growth varies with the value of the BT product. By adding a sequence of this growth with a symbol duration width of T, it is easily thought to cause trouble if the shape is too wide and that its frequency response widens as the shape gets thinner.

With a fully modulated signal, the process must also be invertable at the receiver which is part of the sampling process.

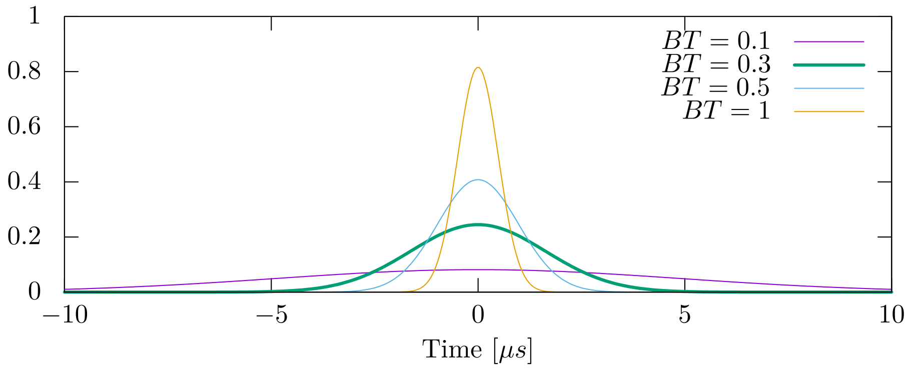


Figure 4.4: Gaussian filter pulse shapes of different filter bandwidths. The $BT = 0.3$ ratio is specified for GSM communications.

4.2 Sampling

The combined quadrature components in $v(t)$ are extracted at the receiver with the same orthogonal carriers of harmonic functions. The extraction in (4.8) uses the trigonometric identities,

$$\begin{gathered}\cos(x)^2 = \frac{1}{2}(1+\cos(x)) \\ \text{and} \\ \sin(x)\cos(x) = \frac{1}{2}(\sin(x-y)+\sin(x+y))\end{gathered} \tag{4.7}$$

to perform the third step. The last step reveals the original value of $m_1(t)$ after the high carrier frequency has been filtered away by a low pass filter.

$$\begin{aligned} v(t)\sqrt{2}\cos(2\pi f_c t) &= \Big(m_1(t)\sqrt{2}\cos(2\pi f_c t) - \\ &\qquad m_2(t)\sqrt{2}\sin(2\pi f_c t)\Big)\sqrt{2}\cos(2\pi f_c t) \\ &= m_1(t)\,2\cos(2\pi f_c t)^2 - m_2(t)\,2\sin(2\pi f_c t)\cos(2\pi f_c t) \\ &= m_1(t)(1+\cos(4\pi f_c t)) - m_2(t)\sin(4\pi f_c t) \\ &= m_1(t) \end{aligned} \tag{4.8}$$

A similar extraction is done for $m_2(t)$ and with both of these original messages their analog nature is converted into a discrete digital signal of I- and Q sample components by an *analog-to-digital converter*, ADC. The Nyquist sampling theorem states, that to be able to reconstruct a real signal, the sampling frequency must be greater than the double of the signal's bandwidth [CS03, p. 111]. For a complex quadrature modulated signal, the sampling frequency can be reduced to the width of the bandwidth since two I- and Q components come from simultaneous sampling. Due to unavoidable noise in the sampled signal and reflections from multipath propagated signals, the received signal must be filtered, estimated and equalized.

4.3 Estimation

The noisy signal is estimated by a filter at the receiver which cross-correlates with an impulse response that, reversed in time, matches the original shape of the source pulse. The impulse response in this matched filter reverts the noise that was applied over the air interface and provides an estimation of the original noise-free signal. Training sequences in GSM help approximate the impulse response coefficients used in this filter and time synchronize to the beginning of a burst. Since the training sequences are fixed bit patterns, these coefficients can be predicted for every burst in respect to the *oversampling ratio*, OSR, the ratio between the bandwidth and the sampling frequency.

The estimated output of the matched filter must then be equalized to mitigate the remaining interference on the channel, such that the sequence of data symbols can be extracted from the phase change in the I/Q samples. The autocorrelation function is useful in this situation as it describes the general dependence of the values of the samples at one time on the values of the samples at another time. A *maximum likelihood sequence estimation*, MLSE, algorithm is often used to decide the binary values by the use of an autocorrelation of the impulse response.

CHAPTER 5 Software defined radio

In this chapter, we take a brief look at the general concepts of software defined radio and how these concepts compare to a regular radio implemented in hardware. As the name implies, some parts of a radio are implemented in software which provides flexibility and efficiency in a variety of different radio applications. The flexibility relates to radios that allow the software to be redefined and therefore can, at any time, be reprogrammed to perform a specific task. Software-updates to accomodate operating needs and improve security are easy and they make the SDR a cost efficient solution, not only for development but also in deployed systems.

Some of the signal processing concepts presented in the previous chapter are concepts that define the functionality of a radio. Mixers, filters, amplifiers and modulators/demodulators are some of the components that enable these functionalities in a radio. For a SDR, these components can be defined in software to accomodate certain needs in different environments.

5.1 General architecture

The general characteristics of SDR imply that the hardware components must be configurable in software and that these components handle the analog signal generation and sampling.

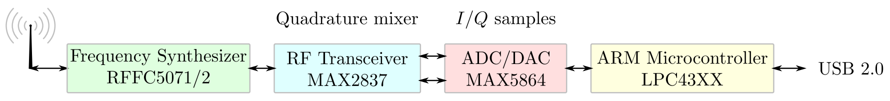


Figure 5.1: A roughly sketched example of HackRF One's architecture and its hardware components [Rya]. A received signal is converted to a supported frequency range, separated into I/Q signal components, then sampled and passed to a USB, interface at the microcontroller. The USB interface allows the device to be configured and both receive and transmit data.

Figure 5.1 roughly illustrates an example of how a SDR data pipeline could look, in this case the HackRF One. HackRF One is an open source SDR platform developed by the Great Scott Gadgets team [Gad].

The USB standard provides a serial interface that is supported by a wide range of machines and microcontrollers. With a microcontroller, the configuration of hardware can be mapped by low-level firmware and this leaves a simple protocol on the USB interface to communicate with the device. For signal generation and sampling, parameters such as tuning frequency and signal gain may be of interest to configure. These opportunities are what makes a SDR attractive and the device can be used for any application as long as the hardware support the requirements e.g. frequency spectrum, bandwidth and sufficient sampling rate.

CHAPTER 6

Telecommunication Analyzer Design

Analysis of the GSM Um air interface requires knowledge of nearby GSM cells and their corresponding frequencies. In Denmark, both 900MHz and 1800MHz are common frequency bands used for mobile phone communication in GSM and the occupied channels within these bands can be scanned for data activity. This functionality defines the first set of requirements for the design of the telecommunication analyzer using an SDR to tune in to an occupied GSM channel and analyze its data traffic. The GSM standard defines the two logical channels, FCCH and SCH, with the purpose of synchronizing to the data traffic. The FCCH burst is transmitted periodically which informs about the occupation of a channel from a nearby BTS and GSM cell. The SCH is similar, but it is used to synchronize to a specific point in time and it carries information regarding current frame number and BTS identity.

Using these two synchronization channels, all traffic carried on a single frequency channel can be monitored and parsed accordingly. The telecommunication analyzer should be able to use these channels to synchronize itself to the traffic and extract information therein.

This chapter walks through the architecture of my design for a telecommunication analyzer which uses an SDR to analyze GSM data traffic and extract information about the network. Figure 6.1 illustrates the structure of the analyzer and a HackRF One is used to serve as an SDR. The colored blocks in the diagram are described in detail throughout this chapter from the bottom and up.

6.1 Radio device interaction

The bottom part of the telecommunication analyzer architecture controls the interaction with the HackRF One. Frequency tuning, amplifier control and data transfer are three examples of parameters that can be controlled through software over a USB 2.0 interface. The host node enables the device and controls these parameters whenever the analyzer application is alive and active. When active, the root of the application ensures a flow of I/Q samples between the SDR and the application. By communicating with the upper layers of the analyzer structure, the analyzer node, the actual

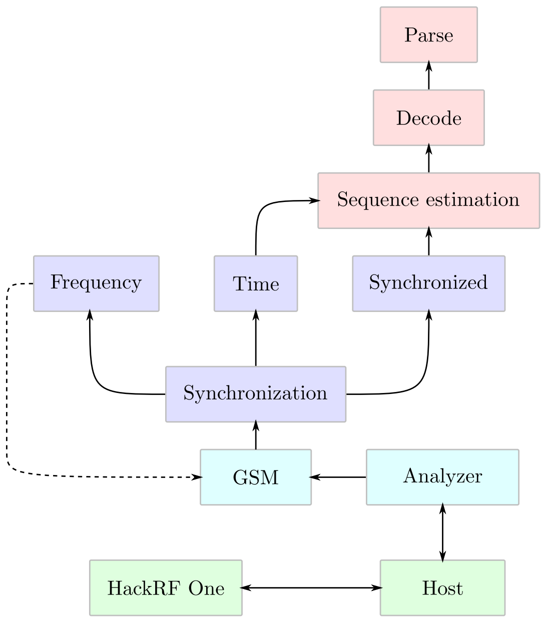


Figure 6.1: The structure of the telecommunication analyzer. A bidirectional path exists between the SDR and the analyzer object for configuration purposes. The SDR samples work their way upwards and are translated to digital information at the sequence estimation block.

behavior is determined by the mobile technology of interest. Figure 6.2 illustrates the two main operations on the link between the host and the SDR, namely frequency tuning and the flow of quadrature samples to the application. These samples are then processed by the analyzer.

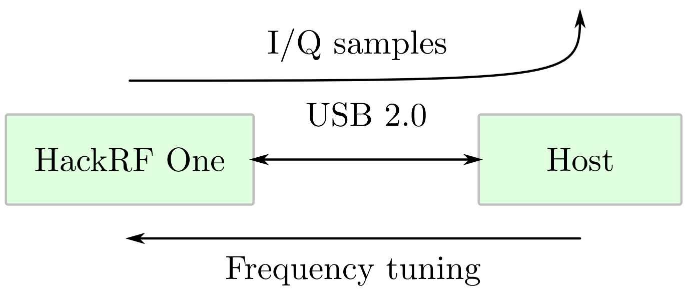


Figure 6.2: A bidirectional link between the host and the HackRF One used to configure the device and move samples of a signal between the two.

6.2 Analyzer

The analyzer is designed such that support for multiple technologies like GPRS, UMTS, HSPA etc. is relatively easy to implement. The core analyzer provides a common interface with the radio device and leaves technology specific functionality to the inheritors. In the situation of GSM, the application may, as an example, perform a GSM cell search and iterate all possible channels given by a band's ARFCN identity and attempt to find FCCH traffic. The analyzer node keeps track of its findings during this search. It is up to the user to decide exactly which channel to monitor. The upper layers process the I/Q samples further, towards the extraction of the digital information in the timeslots. Before leaving the analyzer, the I/Q samples are passed through a low pass filter to remove high frequencies and to further improve the signal, the frequency synchronization state provides feedback on a possible offset in frequency.

The action-interface establishment between the GSM specific implementation and the core analyzer is illustrated on Figure 6.3 which results in two different actions, scan and analyze.

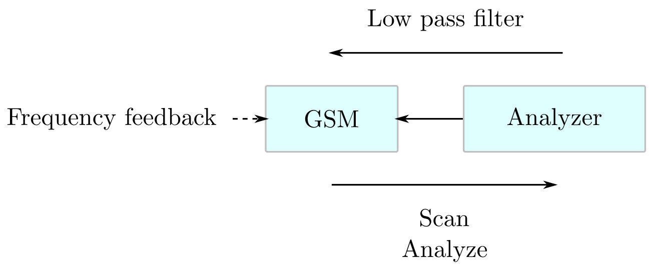


Figure 6.3: The GSM-analyzer inheritor informs the core analyzer of its scan and analyze behaviors and builds an interface between the two. Before passing the samples on to the inheritor, the core analyzer filters them such that only low frequency samples are passed through.

These two actions are independent of one another, but scanning prior to analyzing allows the user to choose a known channel. Otherwise, the frequency or channel is chosen blindly and isn't necessarily occupied. Both of these actions require some sort of synchronization which leads to the synchronization layer.

6.3 Synchronization

Since any knowledge about channels initially is non-existant, the HackRF One must synchronize to a specific frequency and then to the beginning of a burst on this channel. The scan action solely attempts to find these unknown channels by performing a frequency search and returns its findings to the user, whom may then, specifically, choose to analyze one of these channels. The scan action may be performed prior to analyzing, but the user should not be forced to scan everytime, thus he or she may specify a channel to analyze directly. There are three different outtakes during synchronization. Either the application attempts to synchronize to a frequency, otherwise to the beginning of a burst or finally it is synchronized. Since synchronization may be lost, the application has to adapt to this situation and return to a synchronization state.

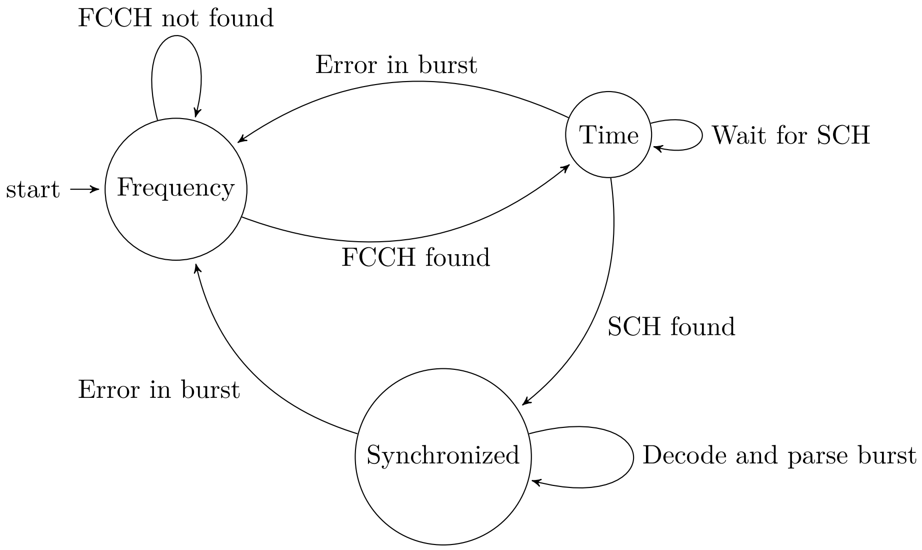
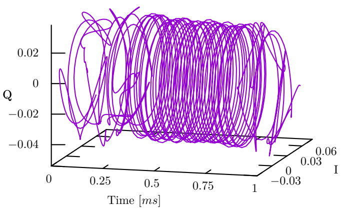


Figure 6.4: A rough sketch of the synchronization finite state machine when analyzing data.

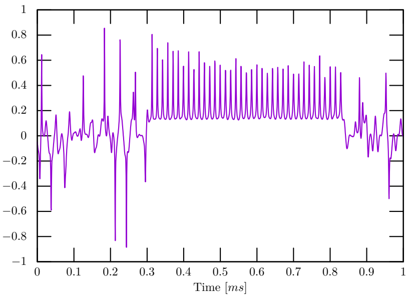
Figure 6.4 roughly illustrates the transitions during synchronization in a finite state machine and the three possible outcomes. The application initially starts in the frequency state and transitions to the time state if and when it finds the FCCH channel. The scan action is a special case and it doesn't advance into the time state, instead it returns and notifies the user of the result. In either case, the frequency state must find the unique frequency correction burst to satisfy the condition.

6.3.1 Frequency state

The frequency correction burst consists of only zero valued bits which, in continuous phase modulation, is a pure tone as illustrated in Figure 6.5a, plotted from collected FCCH data. Inspecting this burst further, reveals a positive phase difference of two samples over time as seen in Figure 6.5b.



(a) Three dimensional phasor plot of a complex signal during a pure FCCH tone at $67.7kHz$.



(b) Phase difference behavior of a pure tone in a continuous phase modulation scheme.

Figure 6.5: FCCH burst of length $576.92\mu s$ (6.5a) and phase difference (6.5b) of pairs of complex samples over time.

Thus, the frequency state condition can be satisfied by collecting a sequence of samples of the 158 bits in a burst, times the OSR and then confirm that all these hits, pairwise, result in a positive phase difference.

To find the phase difference between a pair of samples, let S_n and S_{n-1} be two consecutive complex samples, in polar form $r_n e^{i\theta_n}$ and $r_{n-1}e^{i\theta_{n-1}}$, and their subtraction

$$\begin{aligned} S_n - S_{n-1} &= r_n e^{i\theta_n} - r_{n-1}e^{i\theta_{n-1}} \\ &= r_n r_{n-1} e^{i(\theta_n - \theta_{n-1})} \\ &= S_n \overline{S_{n-1}}, \end{aligned} \tag{6.1}$$

then their difference in angle is given by

$$\angle\left(S_n, S_{n-1}\right) = \text{arctan2}\left(\Im\left(S_n\overline{S_{n-1}}\right), \Re\left(S_n\overline{S_{n-1}}\right)\right) \tag{6.2}$$

where arctan2 is two-argument software implementation of the arctangent function whose value range is $[-\pi;\pi[$.

The transition to the time state is then satisfied if a sufficient amount of complex samples with a phase difference, as given above, are positive.

6.3.2 Time- and synchronized state

Once the frequency correction burst is found, the state machine advances to the time state where the beginning of the frequency correction burst is known. According to the 51 TDMA frame, a synchronization burst follows one TDMA frame later, thus by waiting 4.6ms, the duration of a TDMA frame, the sample at this point in time is a synchronization burst. Figure 6.6 illustrates how the synchronization burst is sent one frame later.

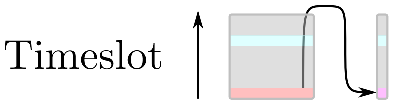


Figure 6.6: A synchronization burst (violet) is transmitted one TDMA frame after the frequency correction burst (red).

All bursts, except for frequency correction, carry a fixed training sequence midway, and knowing that part of the transmitted data, much of the noise in the signal is countered by matching the pulse shape and thus finding the impulse response of the matched filter. Since information is carried in phase, the known training sequence is first mapped to a rotated sequence, $T^* [n]$ and the received, sampled signal $r(t)$ is $v(t)$ with noise

$$r(t) = v(t) + n(t) . \tag{6.3}$$

Recall from the signal processing chapter, that symbols were mapped to complex values, $\{1, j, -1, -j\}$.

A symbol rate sampled version of $r(t)$, denoted as a discrete signal $r_T[n]$, is cross-correlated with the known, rotated training sequence $T^*[n]$, at the training sequence offset

$$R_m[n] \approx r_T[n] \star T^*[n+m] . \tag{6.4}$$

Due to the nature of the GSM training sequences, their ideal autocorrelation with lag m gives a highly peaked function,

$$R_m[n] = \begin{cases} 16 & m = 0 \\ 0 & m \in \{\pm5, \pm4, \pm3, \pm2, \pm1\} . \end{cases} \tag{6.5}$$

The training sequence is therefore easy to distinguish in a signal with noise. By examining the energy of the cross-correlation with different lag, m, over a desired impulse length, L_h of at most 5, the estimated energy then gives an idea of the beginning of the estimated impulse response. The impulse length is limited due to the peak nature in (6.5). The energy is summarized by

$$P(m) = \sum_{i=m}^{m+L_h} (R_i[n])^2 . \tag{6.6}$$

The maximum energy at lag m_{max}, then refers to the start index of the impulse response in $R_m[n]$ and it may be estimated as

$$h[n] = R_{m_{max}}[n] \tag{6.7}$$

The time and synchronized state are similar in that, both states attempt to detect and decode bursts, thus by estimating the impulse response in the matched filter, bursts are now detectable in each case. The time state proceeds to the synchronized state, if the burst is successfully decoded, which leads to the estimation of the actual symbol sequence based on the filtered signal and rotation mapped training sequences.

6.4 Sequence estimation

In both the time and synchronized state an attempt is made to estimate the sequence of symbols within a burst. The impulse response of the matched filter is used for this purpose, and the fact that the exact time between samples is found by the correlation of training sequences, means that both of these can be used to determine the actual digital information. Similar to the channel decoding, the symbol sequence estimation also uses the Viterbi algorithm for the MLSE. This design is based on the design and MATLAB implementation documented in the technical report for GSMsim [EM97].

The probable shifts in phase during a change of state is estimated by using the autocorrelation of the impulse response to examine the gain between every possible outcome. The autocorrelation holds information of the following samples and thus can be used to help determine the probable sequence of rotations based on the gain of previous decisions.

The most probable sequence of rotations are estimated based on the maximum path metric, given by the gain expressed in (6.8). Looking at the multiplication between the sum of products of the previous decisions, $I[m]$, and the autocorrelation of h, R_{hh}, and the complex conjugate of the two possible next state decisions, gives two candidates for a positive and a negative transition.

$$\text{Gain}\left(Y[n], S_p, S_c\right) = \Re\left(\overline{I[n]}Y[n]\right) - \Re\left(\overline{I[n]} \sum_{m=n-L_h}^{n-1} I[m] R_{hh}[n-m]\right) \tag{6.8}$$

Figure 6.7 illustrates two possible transitions for each state, either a negative or a positive, and the nth state is defined as a L_h-long set

$$\begin{aligned} \sigma[n] &= \{I[n], I[n-1], \ldots, I[n-(L_h-1)]\} \\ &\Updownarrow \\ \sigma[n] &\in \{S_0, S_1, \ldots, S_M\} \text{ for } M = 2^{L_h+1}. \end{aligned} \tag{6.9}$$

Just half the states, M, are necessary since only half of the transitions are possible, shifting either by a complex or a real value.

The final sequence of rotated symbols are estimated by the total gain throughout the trellis of 148 bits by traversing the tree and noting each rotation. These values may then be demapped directly to the original binary sequence.

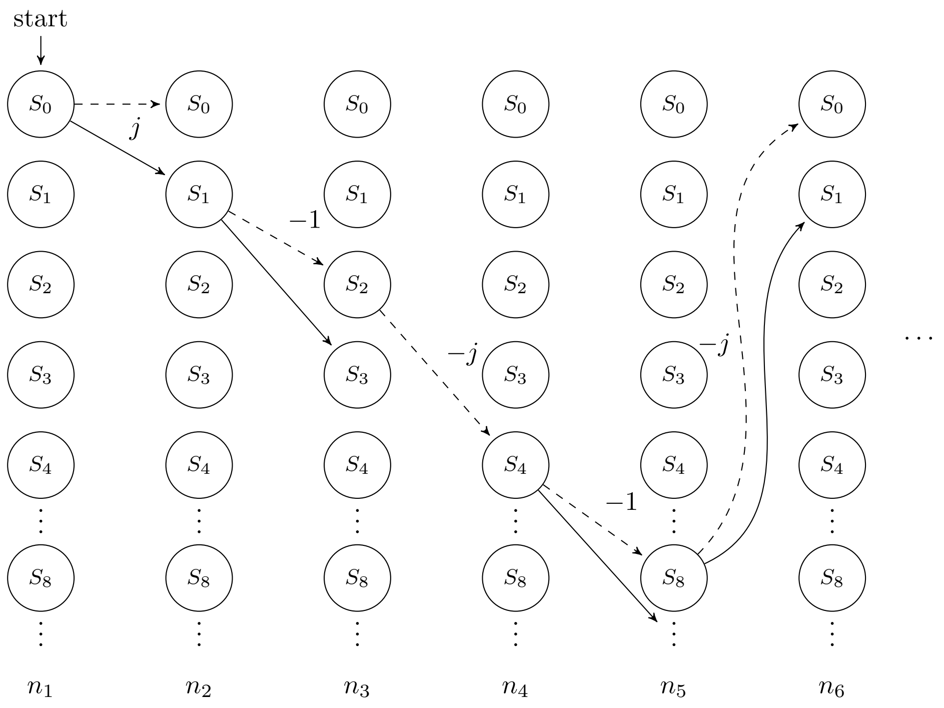


Figure 6.7: A trellis is built to estimate the probable values of the 148 bits within a burst. Due to the way symbols are split across I/Q samples every transition shifts between a real- or a complex rotation value, either positive or negative represented by a dashed line. The path metric is given by (6.8) and this example has previous shifts in state S_0 of $\sigma[6] = \{-1, -j, -1, -j\}$.

6.5 Burst analysis

With an estimated binary sequence, recall that data is interleaved over multiple bursts. It is therefore necessary to fetch three succeeding bursts, in total four, to deinterleave data and restore the original information. Also recall that some of the data is protected by a convolutional code that doubles the number of transferred bits. By reverting these coding schemes, and fixing eventual errors, the actual payload can be parsed. This process, as seen in Figure 6.8, leaves the last part of the structure of the application and GSM protocol specific messages can be read. For speech traffic, this sequence of data must be decrypted prior to decoding.

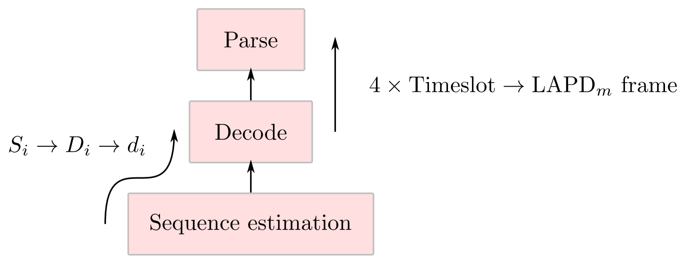

Parse
Decode
Sequence estimation
$4 \times$ Timeslot $\rightarrow$ LAPD$_m$ frame
$S_i \rightarrow D_i \rightarrow d_i$

Figure 6.8: The sequence estimator passes the demapped binary data to the decode block, which reverts the interleaving and convolution of the data and reconstructs the LAPDm frame ready for parsing.

6.5.1 Decode

The GSM standards define the channel coding as presented in Section 3.2.4, which specifies the function of this part of the application by reverting the encoding. Given the estimated binar sequence, the data payload must be extracted and deinterleaved before being deconvolved to the original structure.

6.5.1.1 Deinterleave

The tail bits and the training sequence are not interesting once the system is synchronized. Thus, before reverting the interleaving process, four bursts are gathered and their 114 payload bits are extracted. The extracted payload is then concatenated into a sequence of 456 bits. E.g. for a normal burst, the interleaved bit sequence, with the latest burst n, is found by

$$S_{Ib}\left[n\right] = \mathop{\Big\Vert}_{i=-3}^{0} \left(\left(B_{n+i}\left[k+3\right]\right)_{k=0}^{56} \parallel \left(B_{n+i}\left[k+88\right]\right)_{k=0}^{56} \right). \qquad (6.10)$$

Both data fields in a normal burst are 57 long and their position is at offset 3 and 88, respectively. The concatenated and interleaved bit sequence, S_{Ib}, may then be deinterleaved using the translation sequence, S_T,

$$S_T\left[k\right] = \left(k \mod 4\right) \cdot 114 + \left(\left(49 \cdot k\right) \mod 57\right) \cdot 2 + \left(\left(k \mod 8\right)/4\right). \qquad (6.11)$$

Looking up an index in this sequence, will return a value that corresponds to the actual index in the interleaved bit sequence. The deinterleaved bit sequence, now only convolved, is then given by

$$S_{Cb} = S_{Ib}\left[S_T\left[k\right]\right]. \qquad (6.12)$$

6.5.1.2 Deconvolution

The block coding of control channels adds a CRC check to the bits and four filler bits. In order to decode S_{Cb} in this case, the sequence is first deconvolved using the Viterbi algorithm with a code rate of 1/2, and then divided up into its parts of data and CRC sequence. The estimated burst sequence is found by feeding the convolved sequence to the Viterbi algorithm,

$$S_b = Viterbi\left(S_{Cb}\right). \tag{6.13}$$

Figure 6.9 gives an example of a convolved burst sequence, S_{Cb}, that is passed to the Viterbi deconvolution algorithm and in return results in an estimation of the original bit sequence. Should the estimation result in a wrong decision, then the CRC bits may be used to correct the sequence otherwise the data should be dropped.

6.5.2 Parse

The parsing stage translates the received binary sequence, S_b into recognized data structures within the LAPD$_m$ frame and formats the data for easy readability. Recall the layer three protocols of the GSM stack from Section 3.3, these are categorized and the types of messages sent on the Um interface can be translated into meaningful information.

The LAPD$_m$ frame can take shape of six different formats, where each format is tied to GSM channel types. In that case the B, Bter, B4 and A formats are used on

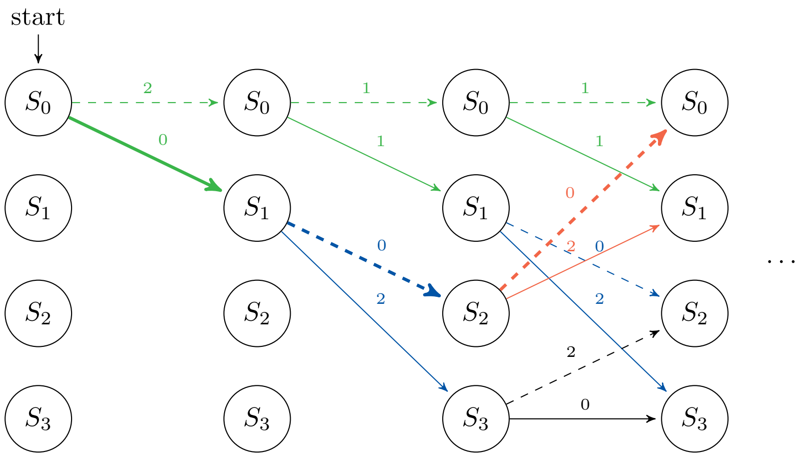


Figure 6.9: As an example, given the sequence $(11, 10, 11)$, the most likely output sequence becomes $(1, 0, 0)$ since it has the smallest path metric of 0 across the trellis. The input sequence is compared to the transition constraints as illustrated in Figure 3.6, $(00/11, 10/01, 11/00)$

DCCHs, where the latter has no information field. The Bbis format is tied to the remaining types that are CCCHs, except for the SCH channel that does not follow the basic formatting. The parser is designed around these three cases of format differences, where the SCH uses the synchronization burst and is identified before the parser. A normal burst is used for both the common- and dedicated channels and the parser must examine the LAPD$_m$ header to determine the type of these.

The layer three type is identified by the PD value as the first octet of the information field. The different types are listed in Table 6.1. This knowledge provides a way to construct a branch for each type and categorize the payload.

Layer three messages further identify themselves by a message type, which follows the PD value and the specifics of these are described in more detail in the 3GPP technical specification; Mobile radio interface layer 3 specification [3GPb].

bits 4321	
0000	group call control
0001	broadcast call control
0010	*packet data signalling*, PDSS1
0011	call control; call related *supplementary service*, SS messages
0100	PDSS2
0101	mobility management messages
0110	radio resources management messages
1000	GPRS mobility management messages
1001	SMS messages
1010	GPRS session management messages
1011	non call related SS messages
1100	location services
1110	reserved for extension of the PD to one octet length
1111	reserved for tests procedures

Table 6.1: A list of possible PDs used in the first octet of the LAPD$_m$ B/Bbis-format information field [3GPc, p. 87].

CHAPTER 7

Implementation

My implementation of the design presented before is a piece of software written in C++ that depends on a few elements from GNU Radio, gr-gsm and the official libhackrf C library [al]. libhackrf provides a configurable interface with the HackRF One and that leaves a simple data structure to communicate with the device. GNU Radio and gr-gsm brings signal processing software implementations of things like low pass filters, frequency- measurements and feedback information for frequency correction.

The project is hosted online and is located at `https://github.com/martinjlowm/telecan`. The implementation uses CMake [tea] as its build- and project management system in conjunction with Google Test [Goob] and Google Mock [Gooa] C++ testing frameworks. The structure of the project is as follows.

» **src/** — The root of all the source code which further divides the code into multiple subdirectories. The directory ttself contains three source code files; the base analyzer, **base_analyzer.cc**, some helper functions, **helper_functions.cc** and the entry point, **telecom_analyzer.cc**. The associated header files, **.h**, are located in the same directory which also applies for the following subdirectories.

  » **device/** — Source code for supported devices and their local data structures are placed here. The HackRF One interface and a circular buffer implementation reside in this directory.

  » **dsp/** — This directory holds digital signal processing source code and it, at the time of this writing, only has a refractional resampler implementation from the GNU Radio project.

  » **gsm/** — All GSM related source code is placed in this directory and its subdirectory, **state_machine**. It contains files similar to the blocks in Figure 6.1, in Chapter 6, apart from a burst counter and a Viterbi detector. The Viterbi detector is a slightly modified version of the symbol sequence estimator of the same name found in gr-gsm [Kry] by Piotr Krysik and is based on GSMsim [EM97]. The burst counter keeps track of the time synchronization feedback provided by the SCH.

    - » **state_machine/** — The implementation consists of two *finite state machines*, FSMs, a synchronization- and a FCCH search state machine and they are both placed in this directory.
  - » **umts/**: This directory is reserved for UMTS specific code.
  - » **lte/**: Similar to UMTS, this directory is reserved for an LTE implementation.
- » **lib/**: CMake is configured to look for Google Test and Google Mock in this directory, which must be downloaded and extracted manually if testing is desired. The directory is otherwise unused.
- » **test/**: A few unit tests for some of the source code reside in this directory.

This chapter introduces the data structures and methods that are used in the implementation of the design to accomplish the functionality of parsing GSM mobile traffic on a specific channel. Similar to the design chapter, this chapter also describes the implementation from the bottom and up as the samples are processed.

7.1 HackRF One interface

The ADC in the HackRF One, MAX5864, provides dual 8-bit I/Q samples which are transferred over the USB interface and written to a circular buffer. The transfer is configured with a callback function that ensures samples to be locally available. Upon filling the local circular buffer, these samples are converted to a complex type with each term as a float data type. The main loop of the application iterates this buffer to continuously process the available samples and pass them along to the specified analyzer. In this situation, with just one inheritor, the GSM analyzer is the receiver of these samples.

7.2 GSM analyzer

The GSM analyzer inherits the properties of the base analyzer and interfaces directly with the SDR source. It acts upon frequency feedback from the synchronization state machine and ensures that samples are filtered by a low pass filter and downsampled from two million samples per second to a multiple of the GSM bit rate, 270.83kb/s. The analyzer is defined to use an oversampling ratio of 4, which implies that every fourth sample may be a symbol. The oversampling ratio helps determine the exact gap between symbols later on, during the synchronization stage.

Generally, the analyzer assures that all the receiver properties are in order, such that the synchronization state machine can proceed to its synchronous state.

7.3 Synchronization state machine

The synchronization state machine consists of three different states, one state that searches for a frequency correction burst, one that searches for the following synchronization burst and then a last state that, when synchronized, estimates, decodes and parses every burst. If the analyzer should lose synchronization at any time, it switches back to an appropriate state from where it can recover synchronization. In order to remain synchronous, the state machine counts all bursts to determine what the next burst type and channel will be. This is based on the GSM standard that specifies that channels repeat every 26th and 51st TDMA frame in a known pattern. Listing 1 lists the three different states of the synchronization state machine and the search for SCH establishes synchronization to the repetitive frame pattern. But first, the occupied frequency is found by the FCCH search state, which is described next.

Listing 1 States of the synchronization state machine.

```
class FCCHSearch : public State {};

class SCHSearch : public State {};

class Synchronized : public State {};
```

7.4 FCCH state machine

One of the synchronization states triggers a new state machine, the FCCH state machine. GSM specifies a frequency correction burst sent on the FCCH channel which is used to correct the tuned frequency to get the best possible signal.

The frequency correction burst is transmitted at 67.7kHz above the carrier frequency and the FCCH state machine supplies the exact frequency to the analyzer which then retunes to the appropriate carrier frequency, such that the burst is found at the correct frequency offset. The state machine consists of five different states listed in Listing 2 and in the best possible scenario, it transitions each state in chronological order as they are listed. The Search state transitions directly to FoundSomething state, whenever there is a pair of samples with a positive phase difference. The FoundSomething state attempts to find the a number of positive phase differences approximately of the length of a burst. Once the phase difference begins to drop, it is assumed that the burst must be over, which Figure 6.5b, in Section 6.3.1, also illustrates. If there are sufficient hits of positive phase differences, then the state machine advances to the FoundFCCH state, which tells the synchronization state machine to continue with the SCH state.

Listing 2 States of the FCCH state machine.

```
class Initial : public State {};

class Search : public State {};

class FoundSomething : public State {};

class FoundFCCH : public State {};

class SearchFailed : public State {};
```

7.5 SCH- and synchronized state

The SCH state is a very brief state and it waits only for a desired number of samples until it reaches the synchronization burst. If it fails to detect the synchronization burst, it transitions back to the FCCH search to correct the frequency properties. If the FCCH state was correct, the burst on the SCH channel should be detectable and its content estimated correctly. The SCH state only fails if the burst sequence estimation is wrong. The synchronization burst is not interleaved across multiple bursts, thus the decoder gives instant feedback if the decoded result is valid. The decoded content is passed onto the parser which then extracts synchronization information and informs the synchronization state machine of the frame pattern.

Both the SCH- and the synchronization state rely on the nature of the training sequence to estimate the channel's impulse response and beginning of the current burst. This leads to the sequence estimator that attempts to find a proper impulse response to first estimate the sequence of symbols and then a binary sequence.

7.6 Sequence estimator

By matching the pulse shape filter of the BTS with extra noise on the radio link, the matching filter output is an almost identical signal to what was generated at the BTS. The symbol sequence hidden in this signal is estimated by the sequence estimator by using the Viterbi algorithm. At this moment, the time between symbols are known by the oversampling ratio, and the analyzer is correctly downsampling the signal to ensure that this gap is correct.

The design of the sequence estimator builds a burst-long trellis for all the samples at a known symbol rate and it finds the most likely sequence of rotations hidden in the signal based on the autocorrelation of the estimated impulse response. The best sequence is calculated by finding the total path metric for the 148 bits in the burst. The combination with the highest gain, is assumed to be the correct sequence and an increment table is precalculated based on the autocorrelation of the impulse response as part of the metric calculation. The table consists of the possible combination for the second term in (6.8).

The last step of the estimator is the demapping part which iterates each decision in the best path and denotes the sign of the current decision and compares it with the previous decision. The way symbols are mapped to the rotated counterparts, requires a similar approach for the demapping process which is the reason for the comparison with the previous decision. Regardless of the correctness of the sequence, it passed on to the decoder which then determines if the sequence is valid.

7.7 Sequence decoder

With an estimated binary sequence, the decoder stores four normal bursts in its local parallel buffers, as illustrated on Figure 7.1, and then reverts the interleaving process followed by deconvolving the convolution. It should be noted that the parallel buffers only apply for normal bursts and not synchronization bursts.

The Viterbi deconvolution process builds a trellis tree of the bit sequence, estimates the most likely sequence of encoded bits, based on the Hamming distance, and produces a bit sequence of half length due to the code rate of 1/2.

In case of a wrong decision, the CRC parity bits are checked and an attempt is made to correct the errors, otherwise the data is discarded. If all is well, the decoded data is passed on to the parser.

7.8 Parser

During time synchronization, the payload of the synchronization burst is examined after decoding and its data reveals a *reduced frame number*, RFN and a *base station*

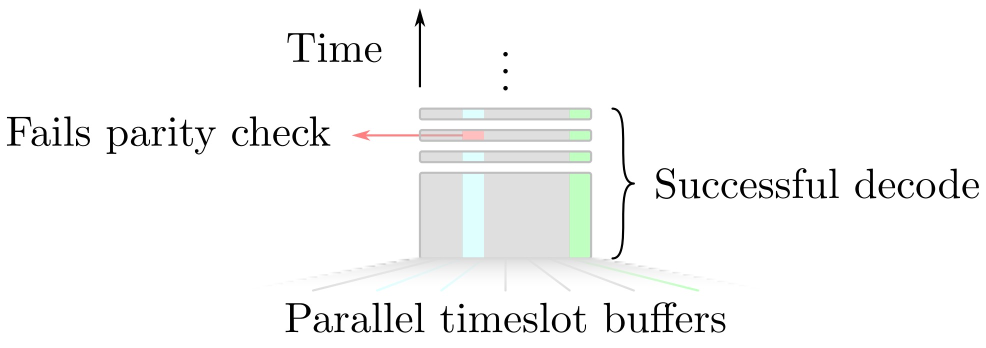


Figure 7.1: Parallel buffers used by the decoder to temporarily store bursts based on their frame number.

identity code, BSIC. The RFN value provides information regarding the current frame number, such that the following burst types can be predicted. In additional to the time information, the BSIC is made of two identifiers, a *network color code*, NCC and a *base station color code*, BCC. The NCC is an identifier unique to the vendor operating the BTS and the BCC identifier specifies which training sequence, out of eight, that is used for bursts of a normal type.

The parser extracts these properties of a synchronization burst and updates the synchronization state machine with the knowledge of frame synchronization. The repeating frame pattern is predictable and the burst type following the synchronization burst is therefore known.

The real function of the parser is to visually present the properties and content of a burst. For that reason, the parser is split over multiple functions for each type of burst. The normal burst parser function is the most complex, since it must determine the LAPD$_m$ format and from there distinguish between layer three protocols. Common control channels are the most simple format to parse as it uses the Bbis format, that in a general sense defines no LAPD$_m$ header. It has just an information field, which holds the layer three payload. Recall the protocol discriminator values in Table 6.1, in Chapter 6, which define the layer three specific payload type. The parser's implementation, as of this writing, is limited and it only writes some of a burst's content.

CHAPTER 8

Evaluation

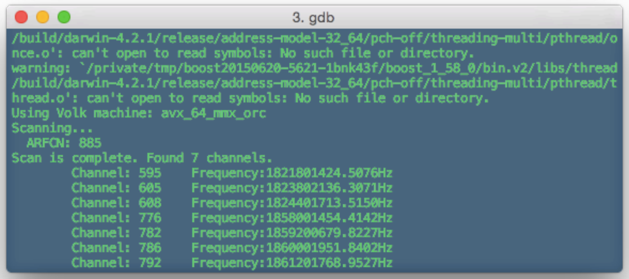
This chapter evaluates my implementation of the telecommunication analyzer and its functionality in terms of a scan and an analyze process. These operations are performed by executing the compiled binary, **telecan**, with a set of options as listed in Figure 8.1. As of this writing, the binary currently depend on GNU Radio, Boost C++ library and libhackrf to be available on the system and the build process is only tested on OS X. A refractional resampler and low pass filter is linked from GNU Radio as part of the downsampling process. These dependencies are meant as a temporary test environment and is planned to be dropped in turn for filter implementations specific to this application.

```
Usage:
    Base Station scan:
          telecan -s <band indicator>

    Tune to channel frequency:
          telecan -b <band indicator> -c <channel number>

Where options are:
    -s    band to scan (GSM850, GSM-R, GSM900, EGSM, DCS, PCS)
    -c    channel of nearby GSM base station
    -b    band indicator (GSM850, GSM-R, GSM900, EGSM, DCS, PCS)
```

Figure 8.1: A set of options supported by the **telecan** binary.

With the libraries available, the scan and analyze can be executed. The scan process is solely meant for BTS searches which, after completion, informs about nearby GSM channels and their corresponding frequencies and calculated frequency offset. Such a scan is performed and illustrated in Figure 8.2 on the GSM 1800MHz band in Nærum, Denmark and the channel number refers to the ARFCN identity.

Three attempts are made for each channel with a new set of samples to improve the certainty that the result is in fact a FCCH channel. An uncertainty exists for some sequences of samples with an oversampling ratio of four and this causes trouble during synchronization. Feedback about phase offset is required which the application currently does not provide.

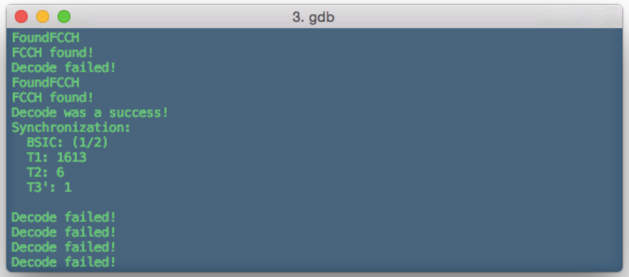


```
/build/darwin-4.2.1/release/address-model-32_64/pch-off/threading-multi/pthread/o
nce.o': can't open to read symbols: No such file or directory.
warning: `/private/tmp/boost20150620-5621-1bnk43f/boost_1_58_0/bin.v2/libs/thread
/build/darwin-4.2.1/release/address-model-32_64/pch-off/threading-multi/pthread/t
hread.o': can't open to read symbols: No such file or directory.
Using Volk machine: avx_64_mmx_orc
Scanning...
  ARFCN: 885
Scan is complete. Found 7 channels.
        Channel: 595    Frequency:1821801424.5076Hz
        Channel: 605    Frequency:1823802136.3071Hz
        Channel: 608    Frequency:1824401713.5150Hz
        Channel: 776    Frequency:1858001454.4142Hz
        Channel: 782    Frequency:1859200679.8227Hz
        Channel: 786    Frequency:1860001951.8402Hz
        Channel: 792    Frequency:1861201768.9527Hz
```

Figure 8.2: A successful scan action that finds nearby channels on the GSM 1800MHz band.

With a list of known nearby channels, the full synchronization process can be initiated with the band- and channel options and the application attempts to find the frequency correction channel again, correct the frequency offset and search for the synchronization channel. Figure 8.3 illustrates a succesful synchronization process which then continues into a failing stage for the following estimations of normal bursts. The application attempts to resynchronize with new samples when ten failures occur.

After the SCH search the application attempts to decode the following bursts in the remaining timeslots before returning to timeslot 0 and these are stored over a sequence of four consecutive TDMA frames. By examining this situation, an array of four bursts is attempted to be decoded. Figure 8.4 shows the midamble of these bursts and how they are incorrectly estimated compared to the known training sequence at the bottom.



```
FoundFCCH
FCCH found!
Decode failed!
FoundFCCH
FCCH found!
Decode was a success!
Synchronization:
  BSIC: (1/2)
  T1: 1613
  T2: 6
  T3': 1

Decode failed!
Decode failed!
Decode failed!
Decode failed!
```

Figure 8.3: A succesful synchronization and parsed burst information.

The first two bursts are completely off, which could be caused by an incorrectly estimation of the impulse response. The exact reason is yet to be discovered however.

```
... 000000000000000000000000000000000000000000000000000000000000 ...
... 000000000000000000000000000000000000000000000000000000000000 ...
... 110111111000011110111101111100010110011010000010110001000110 ...
... 101010001001100110011000010100010111011010111100101011110010 ...
                     01000011101110100100001110
```

Figure 8.4: A midamble sequence of four bursts over four frames in timeslot 0 that shows a misestimation of bits. The training sequence at the bottom should have been visible at the same offset in all four sequences.

With just a partly working application, as of this writing, there is much work to be done in the future. This leads to the next chapter which walks through a list of examples and ideas of what to work on in the future.

CHAPTER 9
Future work

This chapter walks through some features and ideas that didn't make it to the final implementation of the protocol analyzer.

» A bridge design that has a proxy on the SDR interface, such that functions can be mapped to many different types of SDR and not solely a HackRF One.

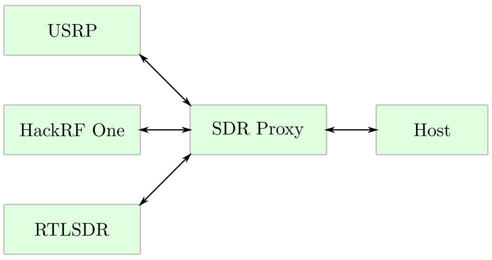


Figure 9.1: Support for multiple types of SDR devices.

» A better pipeline for incoming samples.

Samples are currently fetched in chunks that forces the analyzer to lose synchronization when all samples in a chunk is completely processed. A better pipeline could be based on the approach GNU Radio uses, by having a scheduler that keeps track of ingoing and outgoing samples. Each module must inform the scheduler if it consumes any samples and if that is the case, the supplied pointer of the sample buffer is incremented.

» Support for decoding and decryption of speech frames.

Only dedicated user traffic is encrypted, and such a feature can be added by supporting the verified implementation of A5/1, reverse engineered by Smartcard Developer Association [BGW].

» Decoding of SMS messages, as they are fetched by a mobile phone on the current monitored channel.

» GSM specifies frequency hopping as an optional feature and it might be a necessity for the procotol analyzer in certain areas to remain synchronized.

» Fully implement all extractable information of GSM layer three protocols.

» The analyzer was designed for multiple mobile technologies in mind, thus these were intended to be easy to implement in the future. This project could be a base for future bachelor projects.

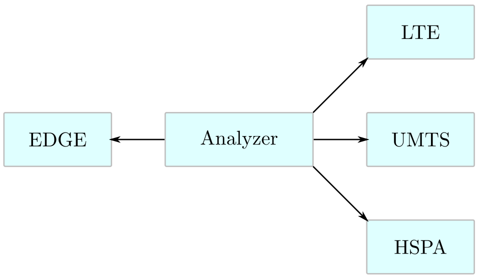


Figure 9.2: An evolved version of the analyzer with support for the upgraded GSM technology, EDGE, and the third and fourth generation mobile technologies.

CHAPTER 10

Project process

This chapter walks through my point of view of the overall process during the project and it reflects thereon. The inception phase of the project started in February with an initial interest in the understanding of the lower level protocols in mobile communications. I chose GSM as the main series of technologies to examine because there is a safe and sufficient amount of documentation available. The preliminary planning of the project was divided into multiple milestones and a risk analysis was performed to predict possible drawbacks.

10.1 Preliminary planning

The initial plan for the project was to divide the project into the following milestones over the project period of twentyone weeks.

» Preparation of the project

» Initial GSM work using a HackRF One

» Implement first wave of GSM test cases

» Implement second wave of GSM test cases

» Implement third wave of GSM test cases

» GUI display of GSM monitored information

» Finalize project

The preparation of the project included a risk analysis and a search for litterature that describes parts on the subject of communication on the GSM Um interface. With a collection of litterature at hand, the initial design and implementation stage began. This led to a series of waves to implement test cases relevant for parsing of GSM traffic. Lastly, the monitored traffic was planned to be presented in a GUI in time for the last milestone, such that the focus could be on the thesis then.

For this process to be as smooth as possible the following risk analysis was performed.

10.1.1 Risk analysis

Table 10.1 lists my assumptions of possible drawbacks that could occur during the project process.

Description	**Likelihood**	**Impact**	**Result**	**Prevention**
Writing the report could take longer than expected.	Medium	Medium	The report will be handed in incomplete.	Multiple draft reports should be handed in during the last milestone. Some sections of the report can be written in parallel to the coding in previous milestones.
Time consumption of each milestones may be wrongly estimated.	High	Medium	What was meant to be done in a specific milestone may not be complete.	Continue to the next milestone and if time allows it return back to the incomplete one and finish it.
Illness	Low	Medium	Tasks may be more time consuming.	None.
Physical damage to- or theft of laptop.	Low	High	Loss of work that in turn complicates the time plan schedule.	Have a cloud backup service active at all times.
Delay in HackRF One delivery.	High	High	Unable to implement software.	Look for an alternative device to use temporarily until the HackRF One arrives.

Table 10.1: Risk analysis made at project inception.

10.2 Evaluation

The initial planning turned out to be unrealistic and the scope of the project was much different than what was predicted. The project process made a turn towards a more digital signal processing-based focus rather than the intended analysis of GSM traffic. This implied that the goal of a GUI and a fully implementation of GSM traffic parsing were not reached. The synchronization, sequence estimation and decoding introduced many more elements behind the scene than what were expected, and in turn a lot more theory to be examined and learned. These factors were the main causes for the imperfect result and there is, as listed in Chapter 9, plenty more work to be done in the future.

To that end, the project has not been managed well and the aim of the project could have been clearer if more time and energy had been spent on researching the topic. The planned deadlines throughout the process were not met and work kept getting postponed which led to a huge amount of work the last month. If the scope of the project had been clearer, it might have been easier to comply with revised deadlines.

The planned milestones ended up being meaningless as the specifics of the GSM air interface were learned, and time allocations of “test case waves” were greatly misestimated. These allocations assumed that everything prior to the parser would have been done and that was not the case. To that end, the HackRF One arrived eight weeks late and this time should have been better valued in terms of research. The misestimated time allocations might have been discovered earlier and then work could have been distributed better.

A lot of work was done the last month, but there were some moments that consumed more time than others. Since the software implementation was developed as a standalone application and without any useful sample debugging utility, a lot of time was spent on debugging the code. If GNU Radio had been used as a platform, then a great deal of this time would have been saved, since it provides efficient tools to visualize and debug input samples. In the end, a more complete implementation could have been accomplished if GNU Radio had been used as a platform.

CHAPTER 11

Conclusion

This thesis proposes a protocol analysation solution for mobile communication technologies that analyses the underlying protocols used on the air interface to control and communicate with mobile entities wirelessly. This solution supports parts of the GSM standard as specified by the 3GPP and allows a synchronous link between the BTS and connected MSs in a specific cell of interest. The Um interface consists of protocols which enable a solid wireless connection that introduces multiplexing principles of TDMA and FDMA to allow multiple parallel connections of several bursts carried in timeslots across multiple frequencies.

Through analysis of these bursts, the underlying protocols of GSM can be examined and this examination shows the ISDN-derived LAPD$_m$ structured data that controls the layer three protocols of GSM for call-, mobility- and radio control. The radio link introduces modulation techniques for transmission of these bursts and their digital information through phase manipulation of radio waves. This enables the use of SDR and its ability to adapt to modulation schemes as specified by the GSM standard. SDR provides a flexible platform for development of software-based wireless solutions and mobile technologies are no exception.

By defining an SDR through software for telecommunication purposes, mobile technologies become more accessible and their underlying use of protocols easier to learn and comprehend.

The protocol analyzer design and implementation described in this thesis is the beginning of a contributed tool that is freely available and is proposed a use for education purposes in comprehension of protocols in mobile communications.

Bibliography

[3GPa] 3GPP. *Digital cellular telecommunications system (Phase 2+); Channel coding.* URL: `http://www.3gpp.org/DynaReport/0503.htm` (visited on June 15, 2015).

[3GPb] 3GPP. *Digital cellular telecommunications system (Phase 2+). Mobile radio interface layer 3 specification.* URL: `http://www.3gpp.org/DynaReport/0408.htm` (visited on June 1, 2015).

[3GPc] 3GPP. *Digital cellular telecommunications system (Phase 2+). Mobile radio interface signalling layer 3. General aspects.* URL: `http://www.3gpp.org/DynaReport/0407.htm` (visited on June 2, 2015).

[3GPd] 3GPP. *Digital cellular telecommunications system (Phase 2+). Mobile Station — Base Stations System (MS — BSS) Interface Data Link (DL) Layer Specification.* URL: `http://www.3gpp.org/DynaReport/0406.htm` (visited on June 1, 2015).

[3GPe] 3GPP. *Digital cellular telecommunications system (Phase 2+); Modulation.* URL: `http://www.3gpp.org/DynaReport/0504.htm` (visited on June 9, 2015).

[3GPf] 3GPP. *Digital cellular telecommunications system (Phase 2+). Multiplexing and multiple access on the radio path.* URL: `http://www.3gpp.org/DynaReport/0502.htm` (visited on May 29, 2015).

[3GPg] 3GPP. *Digital cellular telecommunications system (Phase 2+). Radio Transmission and Reception.* URL: `http://www.3gpp.org/DynaReport/0505.htm` (visited on May 28, 2015).

[al] The Great Scott Gadgets team et al. *HackRF One C library.* URL: `https://github.com/mossmann/hackrf/tree/master/host/libhackrf` (visited on June 28, 2015).

[Bar+03] E. Barkan et al. "Instant ciphertext-only cryptanalysis of GSM encrypted communication". English. In: (2003).

[BGW] Marc Briceno, Ian Goldberg, and David Wagner. *A pedagogical implementation of A5/1.* URL: `http://www.scard.org/gsm/a51.html` (visited on June 22, 2015).

[CS03] Jr. C. Richard Johnson and William A. Sethares. *Telecommunication Breakdown. Concepts of communication transmitted via software-defined radio.* Prentice Hall, 2003.

[DM] Johannes Demel and Kristian Maier. *gr-lte.* URL: `https://github.com/kit-cel/gr-lte` (visited on May 28, 2015).

[EM97] Arne Norre Ekstrøm and Jan Hvolgaard Mikkelsen. *GSMsim. A MATLAB Implementation of a GSM Simulation Platform.* Technical report. Institute of Electronic Systems, Division of Telecommunications, Aalborg University, 1997.

[Erh] Erhversstyrelsen. *Frekvensplan.* URL: `http://dif.erst.dk/Pages/Default.aspx` (visited on May 28, 2015).

[Gad] Great Scott Gadgets. *Great Scott Gadgets; open source hardware for innovative people.* URL: `https://greatscottgadgets.com` (visited on June 16, 2015).

[Gooa] Google. *GoogleMock.* URL: `https://code.google.com/p/googlemock/` (visited on June 28, 2015).

[Goob] Google. *GoogleTest.* URL: `https://code.google.com/p/googletest/` (visited on June 28, 2015).

[gro] Osmocom group. *OpenBSC.* URL: `http://openbsc.osmocom.org/trac/` (visited on May 28, 2015).

[Hei98] Gunnar Heine. *GSM Networks: Protocols, Terminology and Implementation.* Artech House Publishers, 1998.

[Kry] Piotr Krysik. *gr-gsm.* URL: `https://github.com/ptrkrysik/gr-gsm` (visited on May 28, 2015).

[KSW] Piotr Krysik, Dieter Spaar, and Harald Welte. *AirProbe.* URL: `https://svn.berlin.ccc.de/projects/airprobe/` (visited on May 28, 2015).

[Lac] Joshua Lackey. *Kalibrate. Website down.* Archived at `https://archive.org/`. URL: `http://thre.at/kalibrate` (visited on May 28, 2015).

[Net] Range Networks. *OpenBTS.* URL: `http://openbts.org` (visited on May 28, 2015).

[org] GNU Radio organization. *GNU Radio.* URL: `http://gnuradio.org/redmine/projects/gnuradio/wiki` (visited on May 28, 2015).

[Rya] Mike Ryan. *HackRF block diagram.* URL: `https://github.com/mossmann/hackrf/tree/master/doc/wiki/images` (visited on June 16, 2015).

[Sau11] Martin Sauter. *From GSM to LTE. An introduction to mobile networks and mobile broadband.* Volume 1. John Wiley & Sons, Ltd, 2011.

[tea] The CMake team. *CMake.* URL: `http://www.cmake.org` (visited on June 28, 2015).

[VO79] Andrew J. Viterbi and Jim K. Omura. *Principles of digital communication and coding*. McGraw-Hill, 1979.