Implementation of Bluetooth Baseband Behavioral
Model in C Language

Ying-Chi Kuo
Reg nr: LiTH-ISY-EX-3722-2005
Implementation of Bluetooth Baseband Behavioral Model in C Language

Ying-Chi Kuo

Reg nr: LiTH-ISY-EX-3722-2005
Title
Implementation of Bluetooth Baseband Behavioral Model in C Language

Author
Ying-Chi Kuo

Abstract
This master thesis is as a final project in the Division of Computer Engineering at the Department of Electrical of Engineering, Linköping University, Sweden. The purpose of the project is to set up a baseband behavioral model for a Bluetooth system based on standards. In the model, synchronization in demodulation part has been focused on. Simulation results are analyzed later in the report to see how the method in demodulation works. Some suggestions and future works for receiver are provided to improve the performances of the model.

Keyword
Bluetooth, physical layer, G-FSK, synchronization
Abstract

This master thesis is as a final project in the Division of Computer Engineering at the Department of Electrical of Engineering, Linköping University, Sweden. The purpose of the project is to set up a baseband behavioral model for a Bluetooth system based on standards.

A brief introduction of Bluetooth specifications is given in the report and the background knowledge for implementation of this model on a behavioral level is also presented.

In the model, synchronization in demodulation part has been focused on. Simulation results are analyzed later in the report to see how the method in demodulation works. Some suggestions and future works for receiver are provided to improve the performances of the model.
Acknowledgements

First I would like to thank my supervisor, Eric Tell, for giving me help and advices for this project. He is always available there to have a discussion with me and help me to build this model.

I would like to thank Dake Liu, Professor of the chair and the Director of Computer Engineering Division, for offering me this opportunity to work on this project.

I would like to thank my parents for all the support in these 6 months. And I would like to thank all my friends here in Linköping and from Taiwan, for your continuous and lovely encouragement. I appreciate it very much.
Contents

1. Introduction ........................................................................................................... 1

2. Bluetooth baseband standards ............................................................................. 2
   2.1. General description ......................................................................................... 2
   2.2. Overview of the standards ............................................................................. 2
   2.3. Topology ......................................................................................................... 2
   2.4. Physical channel ............................................................................................ 3
   2.5. Connections (Physical links) .......................................................................... 4
   2.6. Packets ........................................................................................................... 4
       2.6.1. Access code ............................................................................................. 5
       2.6.2. Header ..................................................................................................... 6
       2.6.3. Payload ..................................................................................................... 7
   2.7. Packet types .................................................................................................... 7
       2.7.1. Common packet types ............................................................................. 7
       2.7.2. SCO packets ............................................................................................ 8
       2.7.3. ACL packets ........................................................................................... 8

3. System Overview .................................................................................................... 10
   3.1. Bluetooth baseband behavioral model ......................................................... 10
   3.2. HEC (Header Error Check) ........................................................................... 11
   3.3. CRC (Cyclic Redundant Check) ................................................................... 12
   3.4. Header / Payload scrambling (data whitening) ............................................. 14
   3.5. FEC encoding ................................................................................................ 15
   3.6. Modulation architecture ................................................................................ 16
   3.7. AWGN channel architecture ........................................................................ 18
   3.8. Demodulation architecture (synchronization is included) ......................... 21
       3.8.1. Demodulation methods and implementation ........................................ 21
       3.8.2. Synchronization methods and implementation .................................... 22

4. Results of simulation ............................................................................................. 25
   4.1. Methods of arctangent in demodulation ....................................................... 25
   4.2. Methods of arccosine in demodulation ......................................................... 28
1. Introduction

The project in the Division of Computer Engineering plans to implement Bluetooth baseband functions on a processor. Researchers in this division try to implement several Wireless baseband functions on this processor, and Bluetooth is one of them. The projects is based on Bluetooth baseband behavioral model in C programming, and the code would later be converted to assembling code and implemented on the processor.

In this project, several techniques are used: cyclic redundant check (CRC), Forward error correction (FEC), data whitening (scrambling), and Gaussian frequency shift keying (G-FSK) for modulation. Synchronization is also included in demodulation block.

This model is implemented in C programming based on Bluetooth Specification. In order to perform this job, several theoretical (mathematical) and practical concepts should be studied.

In chapter 2, some parts of specification related would be introduced briefly. Algorithms used for this model and the implementation process would be shown in detail in chapter 3. Discussions and analysis of Simulation results are in chapter 4, which also includes future work for next step. Codes for all blocks would be presented in appendix to enable further use of this work.
2. Bluetooth baseband standards

2.1. General description
Bluetooth is an omnidirectional wireless indoor technology. It provides voice and data transmission from all directions within limited-range. It allows connections with a wide variety of fixed or portable devices that would normally have to be cabled together. One being the master and the others, up to 7, being slaves, can communicate with each other in a so-called piconet or PAN.

2.2. Overview of the standards
These are the main characteristics of the standards:

<table>
<thead>
<tr>
<th>RF</th>
<th>2.4G ISM band, 79(23) channels, 1MHz carrier spacing. G-FSK modulation,</th>
</tr>
</thead>
<tbody>
<tr>
<td>FHSS</td>
<td>1600 hops/s among 79 frequencies in a pseudo random fashion, determined by the master.</td>
</tr>
<tr>
<td>Range and Stations</td>
<td>10 meters and 8 devices per piconet</td>
</tr>
<tr>
<td>Time Division Duplex (TDD)</td>
<td>Packets sent in time slots of 625 microseconds between a master &amp; slave.</td>
</tr>
<tr>
<td>Voice channels</td>
<td>3 synchronous</td>
</tr>
<tr>
<td>Security</td>
<td>128-bit key encryption variable configuration</td>
</tr>
<tr>
<td>Address</td>
<td>48-bit MAC connection between devices</td>
</tr>
</tbody>
</table>

2.3. Topology
Bluetooth devices communicate with each other in a piconet or PAN. Devices within this piconet play one of two roles: either the master or one of the slaves. The Bluetooth topology can best be described as a multiple-piconet structure, several piconets can be established and linked together in a topology called a “scatternet”, see Fig. 2.1, a group of independent and non-synchronized piconets that share at least one common Bluetooth device.
2.4. Physical channel

Bluetooth operates in the 2.4 GHz ISM band. In the US and Europe, a band of 83.5 MHz width is available; in this band, 79 RF channels spaced 1 MHz apart are defined. In France, a smaller band is available; in this band, 23 RF channels spaced 1 MHz apart are defined.

The channel is represented by a pseudo-random hopping sequence hopping through the 79 or 23 RF channels. See Fig. 2.2.

![Fig. 2.1 Topology of Bluetooth: overlapping piconets forming a scatternet.](image1)

![Fig. 2.2 Frequency Hopping Spread Spectrum (FHSS)](image2)

The channel is divided into time slots, each 625 us in length. The time slots are numbered according to the Bluetooth clock of the piconet master.

A TDD scheme is used where master and slave alternatively transmit. The master shall start its transmission in even-numbered time slots only, and the slave shall start its
transmission in odd-numbered time slots only. The packet start shall be aligned with the slot start. See Fig. 2.3.

![Fig. 2.3 RX/TX timing in multi-slave configuration](image)

### 2.5. Connections (Physical links)

Between master and slave(s), different types of links can be established. Two link types have been defined:

**Synchronous Connection-Oriented (SCO) link:**

The SCO link is a symmetric, point-to-point link between the master and a specific slave. The SCO link reserves slots and can therefore be considered as a circuit-switched connection between the master and the slave. The SCO link typically supports time-bounded information like voice. The master can support up to three SCO links to the same slave or to different slaves. A slave can support up to three SCO links from the same master, or two SCO links if the links originate from different masters. SCO packets are never retransmitted.

**Asynchronous Connection-Less (ACL) link:**

In the slots not reserved for SCO links, the master can exchange packets with any slave on a per-slot basis. The ACL link provides a packet-switched connection between the master and all active slaves participating in the piconet. Both asynchronous and isochronous services are supported. Between a master and a slave only a single ACL link can exist. For most ACL packets, packet retransmission is applied to assure data integrity.

A slave is permitted to return an ACL packet in the slave-to-master slot if and only if it has been addressed in the preceding master-to-slave slot.

### 2.6. Packets

Each packet consists of 3 entities, the **access code** (68/72 bits), the **header** (54 bits), and the **payload** (0-2745 bits). See Fig. 2.4.
2.6.1. Access code

Access code is used for timing synchronization, offset compensation, paging and inquiry. There are three different types of Access code: Channel Access Code (CAC), Device Access Code (DAC) and Inquiry Access Code (IAC). The channel access code identifies a unique piconet while the DAC is used for paging and its responses. IAC is used for inquiry purpose. See Fig. 2.5.

The CAC consists of a preamble, sync word, and trailer and its total length is 72 bits. When used as self-contained messages without a header, the DAC and IAC do not include the trailer bits and are of length 68 bits.

Preamble:
The preamble is a fixed zero-one pattern of 4 symbols used to facilitate DC compensation. The sequence is either 1010 or 0101, depending whether the LSB of the following sync word is 1 or 0, respectively. See Fig. 2.6.

Sync Word:
The sync word is a 64-bit code word derived from a 24 bit address (LAP). The construction guarantees large Hamming distance between sync words based on different LAPs. In addition, the good auto correlation properties of the sync word improve on the timing synchronization process.
Trailer:
The trailer is a fixed zero-one pattern of four symbols. The trailer together with the three MSBs of the syncword form a 7-bit pattern of alternating ones and zeroes which may be used for extended DC compensation. The trailer sequence is either 1010 or 0101 depending on whether the MSB of the syncword is 0 or 1, respectively. See Fig. 2.7.

![Fig. 2.7 Trailer in CAC when MSB of sync word is 0 (a), and when MSB of sync word is 1 (b).](image)

2.6.2. Header

The header contains information for packet acknowledgement, packet numbering for out-of-order packet reordering, flow control, slave address and error check for header. See Fig. 2.8.

![Fig. 2.8 Header format](image)

AM_ADDR:
The AM_ADDR represents a member address and is used to distinguish between the active members participating on the piconet. To identify each slave separately, each slave is assigned a temporary 3-bit address to be used when it is active. The AM_ADDR of the slave is used in both master-to-slave packets and in the slave-to-master packets. The all-zero address is reserved for broadcasting packets from the master to the slaves.

TYPE:
Sixteen different types of packets can be distinguished. The interpretation of the TYPE code depends on the physical link type associated with the packet. The TYPE code also reveals how many slots the current packet will occupy. This allows the non-addressed receivers to refrain from listening to the channel for the duration of the remaining slots.

FLOW:
This bit is used for flow control of packets over the ACL link. When the RX buffer for
the ACL link in the recipient is full and is not emptied, a STOP indication (FLOW=0) is returned to stop the transmission of data temporarily. When the RX buffer is empty, a GO indication (FLOW=1) is returned.

ARQN:
The 1-bit acknowledgment indication ARQN is used to inform the source of a successful transfer of payload data with CRC, and can be positive acknowledge ACK or negative acknowledge NAK. If the reception was successful, an ACK (ARQN=1) is returned, otherwise a NAK (ARQN=0) is returned.

SEQN:
The SEQN bit provides a sequential numbering scheme to order the data packet stream. For each new transmitted packet that contains data with CRC, the SEQN bit is inverted. By comparing the SEQN of consecutive packets, correctly received retransmissions can be discarded.

HEC:
Each header has a header-error-check to check the header integrity. The HEC consists of an 8-bit word generated by the polynomial 647 (octal representation). After the initialization, a HEC is calculated for the 10 header bits. Before checking the HEC, the receiver must initialize the HEC check circuitry with the proper 8-bit UAP (or DCI). If the HEC does not check, the entire packet is disregarded.

2.6.3. Payload
The packet payload can contain either voice field, data field or both. It has a data field, the payload will also contain a payload header.

2.7. Packet types

2.7.1. Common packet types
ID packet: It is a very robust packet since the receiver uses a bit correlator to match the received packet to the known bit sequence of the ID packet.

NULL packet: It is used to return link information to the source regarding the success of the previous transmission (ARQN), or the status of the RX buffer (FLOW). The NULL packet itself does not have to be acknowledged.

POLL packet: In contrast to the NULL packet, it requires a confirmation from the recipient. It is not a part of the ARQ scheme. The POLL packet does not affect the
ARQN and SEQN fields. Upon reception of a POLL packet the slave must respond with a packet. This return packet is an implicit acknowledgement of the POLL packet.

**FHS packet:** It is used for frequency hop synchronization before the piconet channel has been established, or when an existing piconet changes to a new piconet. In other words, it is used in page master response, inquiry response, and in master slave switch.

Its payload contains 144 information bits plus a 16-bit CRC code, and is coded with a rate 2/3 FEC which brings the gross payload length to 240 bits. The FHS packet contains real-time clock information, and need to be updated before each retransmission when establishing connection. See Fig. 2.9.

![Fig. 2.9 Format of the FHS payload](image)

### 2.7.2. SCO packets

SCO (Synchronous Connection-Oriented) packets are used on the synchronous SCO link. The packets do not include a CRC and are never retransmitted. SCO packets are routed to the synchronous I/O (voice) port. Up to now, three pure SCO packets have been defined. In addition, an SCO packet is defined which carries an asynchronous data field in addition to a synchronous (voice) field. The SCO packets defined so far are typically used for 64 kb/s speech transmission.

![Fig. 2.10 SCO packets](image)

### 2.7.3. ACL packets

ACL (Asynchronous Connection-Less) packets are used on the asynchronous links. The information carried can be user data or control data. Including the DM1 packet, seven ACL packets have been defined. Six of the ACL packets contain a CRC code and
retransmission is applied if no acknowledgement of proper reception is received (except in case a flush operation is carried out). The 7th ACL packet, the AUX1 packet, has no CRC and is not retransmitted.

<table>
<thead>
<tr>
<th>Type</th>
<th>Payload Header (bytes)</th>
<th>User Payload (bytes)</th>
<th>FEC</th>
<th>CRC</th>
<th>Symmetric Max. Rate (kb/s)</th>
<th>Asymmetric Max. Rate (kb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DM1</td>
<td>1</td>
<td>0-17</td>
<td>2/3</td>
<td>yes</td>
<td>108.8</td>
<td>108.8</td>
</tr>
<tr>
<td>DH1</td>
<td>1</td>
<td>0-27</td>
<td>no</td>
<td>yes</td>
<td>172.8</td>
<td>172.8</td>
</tr>
<tr>
<td>DM3</td>
<td>2</td>
<td>0-121</td>
<td>2/3</td>
<td>yes</td>
<td>258.1</td>
<td>387.2</td>
</tr>
<tr>
<td>DH3</td>
<td>2</td>
<td>0-183</td>
<td>no</td>
<td>yes</td>
<td>390.4</td>
<td>565.6</td>
</tr>
<tr>
<td>DM5</td>
<td>2</td>
<td>0-224</td>
<td>2/3</td>
<td>yes</td>
<td>286.7</td>
<td>477.8</td>
</tr>
<tr>
<td>DH5</td>
<td>2</td>
<td>0-330</td>
<td>no</td>
<td>yes</td>
<td>433.9</td>
<td>723.2</td>
</tr>
<tr>
<td>AUX1</td>
<td>1</td>
<td>0-29</td>
<td>no</td>
<td>no</td>
<td>185.6</td>
<td>185.6</td>
</tr>
</tbody>
</table>

Fig. 2.11 ACL packets
3. System Overview

3.1. Bluetooth baseband behavioral model

The transceiver model is implemented in C programming and blocks are connected as shown in Fig. 3.1.

Fig. 3.1 Bluetooth baseband behavioral model

Blocks in this transceiver model are implemented separately. Following would describe functions and implementations for each block in four main parts, error check (HEC and CRC), data scrambling, forward error correction (1/3 FEC and 2/3 FEC), and
modulation/demodulation (synchronization).

3.2. HEC (Header Error Check)

HEC performs error detection on the packet's header by checking the remainder in receiver part, see Fig. 3.2. A 10-bit header is divided by some special polynomial in HEC circuitry in Tx part to get the remainder. The 8-bit remainder is appended to the 10-bit header and they are both transmitted to receiver. After being received In Rx part, the 18-bit data again pass through the same HEC circuitry to see if remainder is zero. If not, errors have been detected.

![Fig. 3.2 HEC generation and checking algorithm](image)

It is composed of two sub-blocks, one for HEC generation (transmitter) and one for HEC checking (receiver), see Fig. 3.3.

![Fig. 3.3 HEC is composed of two sub-blocks, HEC generation and HEC checking.](image)

An 8-bit shift register is used for implementation, see Fig. 3.4. The shift register is preloaded with the UAP of the master device. It participates in the detection of the address because the UAP is used. In the checking the shift register is initialized as before and HEC is performed on the header.
In HEC circuitry, the polynomial used for division is \( g(D) = D^8 + D^7 + D^5 + D^2 + D + 1 \). LSB bits of the header are entered first. The LSB of UAP goes to the left-most shifter register element. After all 10 bits have been shifted into the circuit, the left 8-bit data in shift register is remainder. HEC implementation for both transmitter and receiver is based on Fig. 3.5.

**3.3. CRC (Cyclic Redundant Check)**

The CRC and HEC blocks are very similar except that HEC must be done in all packets and it is 8 bits while CRC is done on some packets and is 16-bit CRC Generator, see Fig. 3.6.
Fig. 3.6 CRC generation and checking algorithm

This block is composed of two sub-blocks, one for CRC generation and one for CRC checking, see Fig. 3.7.

Fig. 3.7 CRC is composed of two sub-blocks, CRC generation and CRC checking

The 16-bit shift register is preloaded with the UAP of the master device. Since the UAP and DCI are 8 bit values, they are loaded into the 8 least significant (left-most) and other bits are set to zero, see Fig. 3.8. In the checking the shift register is initialized as before and CRC is performed on the header. It participates in the detection of the address because the UAP is used.

Fig. 3.8 Initial state of the CRC generating circuit
The generator polynomial for division is (CRC-CCITT) \( g(D) = D^{16} + D^{12} + D^5 + 1 \). LSB bits of the data are entered first till all data bits in payload are shifted into the LFSR. The left 16-bit data in shift register is remainder. CRC implementation for both transmitter and receiver is based on Fig. 3.9.

![LFSR circuit generating the CRC](image)

**Fig. 3.9** LFSR circuit generating the CRC

The main piece of code in CRC is similar to that in HEC.

### 3.4. Header / Payload scrambling (data whitening)

Data whitening is done in order to randomize the data from highly redundant patterns and to minimize DC bias in the packet.

This block is composed of two sub-blocks, one for scrambling and one for descrambling (data extraction), see Fig. 3.10. It is performed on the packet header and the payload (including the CRC).

![Scrambling circuit](image)

**Fig3.10** Scrambling is composed of two sub-blocks, Header/Payload scrambling and Header/Payload descrambling

The 7-bit shift register is preloaded with the master Bluetooth clock, CLK 6-1, extended with an MSB of value one. CLK1 is written to position 0, CLK2 in position 1 and so on.

The polynomial is \( g(D) = D^7 + D^4 + 1 \). The header and data are XORed with the output of the LFSR. To descramble the header and data, the scrambled stream is passed on the same circuit. Implementation for scrambling is based on Fig. 3.11. Implementation for whitening seems to be similar to it for HEC/CRC, though the difference is a bit comes
out right after a bit is input to circuit in whitening. While in HEC/CRC, it should be waited till all bits pass through the circuit and get the remainder.

**Fig. 3.11** Data whitening LFSR

### 3.5. FEC encoding

There are two types of FEC, 1/3 FEC and 2/3 FEC. Both of them are used to reduce the chances of getting corrupted information. It works by increasing the no. of transmitted bits which reduces the usable bandwidth available for the information. 1/3 FEC is used to protect the packets’ header and some types of payloads are protected by 1/3 FEC, some by 2/3 FEC and some are not protected.

1/3 FEC is implemented by repeating the bit three times, see Fig. 3.12.

**Fig. 3.12** Bit-repetition encoding scheme

2/3 FEC is implemented using the polynomial \( g(D) = (D + 1)(D^4 + D + 1) \), see Fig. 3.13. Bits in payload are divided into blocks, each block has 10 information bits and those 10 bits data would be encoded into a 15-bit codeword by appending a 5-bit remainder in tail. This encoding can correct all single errors and detect all double errors in each codeword.
Fig. 3.13 LFSR generating the (15, 10) shortened Hamming code

The code for encoding part is similar to HEC/CRC. In correction, the error bit is detected and corrected by judging the remainder in receiver. There are 10 specific cases in remainder to see which bit should be altered from 1 to 0 or from 0 to 1.

3.6. Modulation architecture
Background knowledge of G-FSK can be seen in Appendix A.

In Fig 3.14, Bits are rescaled to NRZ (1 and -1). NRZ signal passes through Gaussian filter, and each impulse in the signal is reshaped to Gaussian distribution. The purpose of using Gaussian filter is to reduce spectrum bandwidth. In FSK, reshaped pulses from Gaussian filter have been integrated to get phase curve for transmitting. The phase curve is smooth when being integrated, which means frequencies are changing gradually in FSK, due to the use of Gaussian filter. I and Q components are generated by calculating cosine and sine values from phase $\psi(t)$ from FSK. Notice that in order to make simulation easier, carrier frequency is not considered in the model. Sampling rate for bits, sam_b, is set a parameter to modulation block to determine number of samples per bit. Sampling rate for Gaussian filter, sam_g, is another parameter to determine number of samples per Gaussian distribution. According to the code, sam_g should be at least a sample larger than sam_b. Also the modulation index $h$ is the third parameter to this block, which should be 0.32 in Bluetooth.

After being rescaled to 1 or -1, bits pass through Gaussian filter. Gaussian filter is specified by its impulse response given by:

$$g(t) = \frac{1}{\sqrt{2\pi\sigma T}} e^{-\frac{-t^2}{2\sigma^2 T^2}}$$

with
\[ \sigma = \frac{\sqrt{\ln 2}}{2\pi BT} \]

and BT = 0.5 for G-FSK.

The Gaussian filter used in G-FSK is generally specified by its BT product, where B is the 3 dB bandwidth of the filter and T is the symbol duration. For the Gaussian filter used in Bluetooth, BT = 0.5 according to standards.

In Gaussian filter, each impulse is reshaped to Gaussian distribution. It smoothes the curve when accumulating signals in FSK, see Fig. 3.15.

![Fig. 3.15 Signals after pulse shaping when input data is (1, 0, 1, 1, 0, 0, 1).](image)

FSK (VCO) is the next step. The input signal from Gaussian filter is normalized before being scaled since the area under Gaussian distribution should be 1, see Fig. 3.16. Then it is multiplied by index h and 2\( \pi \), h is fixed and set to be 0.32 based on Bluetooth specification. Then phase is integrated for transmitting.
Fig. 3.16 FSK (VCO) generates phase curve for transmitting.

Black curve in Fig. 3.17 is from integrating signals which are reshaped by Gaussian filter while gray one is from integrating rectangular pulses without Gaussian filter. It can be seen that frequencies change more smoothly in black one than in gray one.

Fig. 3.17 Accumulate the phase if signal is \((1, 0, 1, 1, 0, 0, 1)\).

3.7. **AWGN channel architecture**

AWGN is considered in this channel model, see Fig. 3.18.
Fig. 3.18 Additive White Gaussian Noise

SNR is a parameter to channel block, it can be used to scale noise amplitude.

\[
SNR = 20 \log\left(\frac{E_s}{E_n}\right)
\]

\(E_s\), power of transmitted signal, is

\[
\cos^2 \theta + \sin^2 \theta = 1
\]

\(E_n\) is the power of noise.

The probability of AWGN would distribute as Gaussian distribution. In Fig. 3.19, the Gaussian distribution is based on \(\sigma = 1\).

When calculating the power of noise, only the range between \(x = \pm 3\sigma\) is considered since 99% of probability is within this range. The maximum magnitude of noise, \(n\), then is set on position \(x = 3\ (x = -3)\), which refers to \(n = 3\sigma\).

The range between \(x = 3\) and \(x = -3\) is divided into 20 blocks and each block is 0.1n, see Fig 3.19. These 21 scales are for calculating power of noise under the curve.
The method used for calculating power for noise would be given in detail as below. Probabilities under the curve should be normalized first. Noise probabilities for those 21 discrete scales should be added to be 1.

\[
P(0) + 2(P(\pm 0.1n) + P(\pm 0.2n) + P(\pm 0.3n) + P(\pm 0.4n) + P(\pm 0.5n) + P(\pm 0.6n) + P(\pm 0.7n) + P(\pm 0.8n) + P(\pm 0.9n) + P(\pm 1.0n)) = 1
\]

After normalization, probability for each noise scale would be rescaled as

\[
P(0) = 0.120 \\
p(\pm 0.1n) = 0.115 \\
p(\pm 0.2n) = 0.100 \\
p(\pm 0.3n) = 0.079 \\
p(\pm 0.4n) = 0.058 \\
p(\pm 0.5n) = 0.039 \\
p(\pm 0.6n) = 0.024 \\
p(\pm 0.7n) = 0.013 \\
p(\pm 0.8n) = 0.007 \\
p(\pm 0.9n) = 0.003 \\
p(\pm 1.0n) = 0.001
\]

To calculate the power of noise

\[
E_n = (0)P(0) + 2((0.1n)^2P(0.1n) + (0.2n)^2P(0.2n) + (0.3n)^2P(0.3n) + (0.4n)^2P(0.4n) + (0.5n)^2P(0.5n) + (0.6n)^2P(0.6n) + (0.7n)^2P(0.7n) + (0.8n)^2P(0.8n) + (0.9n)^2P(0.9n) + (1.0n)^2P(1.0n)) \\
= n^2 (0.0184)
\]

Noise can be scaled properly by adjusting \( \sigma \), and \( \sigma \) can be calculated from SNR, which is an input parameter to channel block.

To get \( \sigma \) value from input SNR
SNR = \(20 \log \left( \frac{E_s}{E_n} \right) = 20 \log \left( \frac{1}{E_n} \right)\)

\[ E_n = \frac{1}{10^{(SNR/20)}} = n^2(0.0184) \]

\[ n = 3\sigma = \sqrt{\frac{E_n}{0.0184}} = \sqrt{\frac{1}{0.0184 \times 10^{(SNR/20)}}} \]

Relation between variance \(\sigma\) and SNR is

\[ \sigma = \frac{1}{3} \sqrt{\frac{1}{0.0184 \times 10^{(SNR/20)}}} \]

Therefore, \(\sigma\) can be determined by setting SNR and used for scaling the amplitude of noise.

### 3.8. Demodulation architecture (synchronization is included)

In demodulation, both \(I(\cos \psi(t))\) and \(Q(\sin \psi(t))\) components are received from channel, see Fig. 3.20.

![Demodulation model](image)

**Fig. 3.20** Demodulation model

#### 3.8.1. Demodulation methods and implementation

In this method, binary bit value is decided by detecting frequencies in each bit period. Frequencies can be either positive or negative in each period depends on the bit when transmitted from transmitter. Frequency is slope of phase curve in Fig. 3.20. For example, in the range between 0.5T and 1.5T, slope is positive all the time, which can be used to judge the first bit as ‘1’, and so on.
To extract the phase from received curve

\[ \psi(t) = a \tan\left(\frac{\sin \psi(t)}{\cos \psi(t)}\right) \]

Frequency would be

\[ \Delta \psi(t) = \psi(t) - \psi(t - T_{\text{sample}}) \]

\( T_{\text{sample}} \) is the period between two samples. Sampling rate is set as a parameter to demodulation block, which should be the same as it in modulation block.

In stead of seeing \( \Delta \psi(t) \) in one sample, \( \Delta \psi(t) \) in half number of samples per period are accumulated to judge if frequency is either positive or negative. For example, 5 samples centered in the steepest slope are accumulated when sampling rate is 10 for bits. If accumulated frequency is positive, the bit is ‘1’, otherwise, ‘0’.

### 3.8.2. Synchronization methods and implementation

What have been discussed in demodulation is receiving a phase curve from channel. In Fig. 3.22, there is always a steepest slope at some fixed position in each bit period, like dots on the curve. The steepest slope also refers to the most frequency offset in FSK algorithm. These fixed positions can be used to synchronize each bit when receiving signals.
Way to synchronize received curve is to point out the steepest point in each bit period. It is assumed that every T the steepest slope appears again. When the slope value is not larger than both nearby ones, and when it is decreasing, shift the waveform right one sample, see Fig. 3.23(a), when increasing, shift it left, see Fig. 3.23(b), otherwise, do nothing. Therefore, all bits can be synchronized by tracing the steepest points on phase curve.

Though when comparing the steepest slope with two nearby ones, which are both only a sample away, it might cost problem when there is noise to the curve. Since value of the adjacent slope is close to the steepest (central) one, it is hard to tell the real magnitude sequence when there is some noise. Because noise makes the curve become rough and not smooth anymore, the detection of slope would not be accurate.
Instead of seeing only one sample away, slopes in a couple of samples away can be more suitable.

Square root of sampling rate, $\sqrt{s}$, is taken when choosing how many samples away in this model. For example, when sampling rate is 10, $\sqrt{s}$ is 3. The steepest slope should be compared with the slope in 3 samples left and 3 samples right in distance. If values of those 3 slopes are decreasing from left to right, shift the curve left, if increasing, shift the curve right, otherwise, do nothing. See Fig. 3.24.

![Fig 3.24](image)

**Fig 3.24** Compare with slopes $(\sqrt{s})$ samples away instead of one sample
4. Results of simulation

The results of simulation are obtained from the baseband transceiver model in Fig 3.1. When simulating, a random data has been input to the model. The input data includes: a 64-bit sync word (to generate a 72-bit access code in accesscode block), a 10-bit header (to generate a 10-bit header in header block), a 144-bit payload (to generate a 144-bit payload in payload block), a 8-bit UAP (to generate a 8-bit payload in UAP block), and a 6-bit clk (to generate a 7-bit clk in clk block).

In modulation block, there are 3 parameters should be set in advanced, sam_b (samples per bit), sam_g (samples per Gaussian distribution), and index h (0.32 when in Bluetooth). According to the code, sam_g should be at least a sample larger than sam_b. In both channel block and in demodulation block, sam_b should also be set, which is always the same as it in modulation.

Simulations have been focused on some aspects:

1. How the demodulation block works when without synchronization?
   Adjust SNR value under a fixed sampling rate to get BER curve to see the performance under different SNR, see Fig. 4.1. Also sampling rate for bit, sam_b, is adjusted under a fixed SNR value to see the performance under different sampling rate, see Fig. 4.2.

2. How the demodulation works when with synchronization?
   Adjust SNR value again to see if the curve shifts left or right too much when there is some noise (AWGN). BER and sample offset are considered in this case. See Fig. 4.3 and Fig. 4.4.

4.1. Methods of arctangent in demodulation

As mentioned above, \( \psi(t) = a \tan \left( \frac{\sin \psi(t)}{\cos \psi(t)} \right) \) is used in demodulation method.

Different SNRs under a fixed bit sampling rate: (without synchronization)
BER is almost 0% when SNR is over 23.5db, see Fig. 4.1.
Different bit sampling rates under a fixed SNR: (without synchronization)

It can be seen in Fig. 4.2, BER is not necessarily decreasing when sampling rate is increasing. It is because when bit sampling rate is increasing, distance between samples on the phase curve decreases. Frequency is calculated from subtraction of phase value between two nearby samples, when there is less distance between samples, value from subtraction can be affected by noise more.

Synchronization is now considered in simulation.

Different SNRs under a fixed bit sampling rate: (with synchronization)

When SNR is over around 28.94db, see Fig. 4.3, BER_P is almost 0%. But when noise
is increased a bit more, BER_P jumps to 25% suddenly.

**Fig. 4.3** Different SNRs under a fixed bit sampling rate

Sample offset, from shifting left or right when synchronizing bits, is used to analyze the result.
There are 5 samples per bit and 366 bits per packet, it should be totally 1830 (366x5) samples in a packet, when no shifting. When SNR is over 28.94db, see Fig. 4.4, synchronization does not affect the result since it does not shift so far when tracing the steepest slope. Number for samples in a packet is 1830 or 1831 and sample offset is 0 or 1, which is still within the range of ±5 samples (sampling rate is 5). When there is a larger noise, phase curve is not smooth anymore. It would be hard to detect the steepest slope when synchronizing bits. Synchronization makes the curve shift left or right more frequently. The sample offset is even more than 5 when SNR is less than 28db, which means it may detect over to the next bit.
To explain such imperfection, it can be the problem in demodulation when calculating

$$\psi(t) = a \tan\left(\frac{\sin \psi(t)}{\cos \psi(t)}\right)$$

to extract phase back.

The division between $\sin \psi(t)$ and $\cos \psi(t)$ can cause the extracted phase become unstable when there is some amount of noise. Slope used for synchronization would also become rough and is not growing or declining gradually anymore.

4.2. Methods of arccosine in demodulation

There is another way to improve the situation.

In stead of arctangent, arccosine is used since there is no division between sine and cosine. Phases are defined to be either positive or negative by sine value.

When calculating phase,

$$\psi(t) = a \cos(\cos \psi(t))$$

Define phase to be either positive or negative according to sine value,

If $\sin \psi(t) > 0$

$$\psi(t) = |\psi(t)|.$$

else if $\sin \psi(t) < 0$


\[ \psi(t) = -|\psi(t)|. \]

else
\[ \psi(t) = \psi(t). \]

Frequency is
\[ \Delta \psi(t) = \psi(t) - \psi(t - T_{\text{sample}}) \]

Sample offset is compared between method of arctangent and arccosine under a fixed \( \text{sam}_b \):
Since bit sampling rate is 5, total samples should be 1830. When using method of arccosine in demodulation, the offset even shifts more than when using method of arctangent, which alters the assumption above. See Fig. 4.5.

![Fig. 4.5 BER_P in arctangent and arccosine under a fixed sam_b](image)

The simulation result shows that using arccosine in demodulation is not better or sometimes even worse than using arctangent. Therefore, the method has been discarded.
5. Suggestions and future work

There is a suggestion on how to improve demodulation performance by reducing the influence from noise to synchronization part.

As can be seen in Fig 4.3 and Fig 4.4, when SNR is less than 28db, demodulation (with synchronization) does not work properly because BER is too high. Since the phase curve would become rough when affected by AWGN, synchronization makes the curve shift left or right frequently, which is not proper, and sample offset is too high to keep the curve on the right position. When the curve shifts too much, it is not able to trace the right slope (frequency) for deciding bit value, and BER increases a lot.

To solve the problem, in stead of seeing only one steepest slope each time, more slopes can be considered once to decide whether the curve should be shift left or right or do nothing.

In Fig.5.1, only when frequencies are decreasing in all 3 continuous periods, or decreasing, shift the curve left or right. Otherwise, do not shift it at all. Due to this improvement, curve would shift less frequently when being affected by noise.

![Fig 5.1 Consider 3 slopes instead of one each time](image)

Not only suggestion mentioned above, there are some more future works.

There is only a fixed set of random data in a packet to be used when simulating, which is not enough. In order to get a more precise result, more variable data can be included in simulation. Also, more data bits can be added to payload, which is at most 2075 bits.

A packet is transmitted each time when in simulation above. Instead, more packets can be transmitted as a flow. When there is a flow of packets transmitting, a feedback circuit may be needed in the model for knowing if packets should be retransmitted when there is an error.

Simulation so far only tests performance of demodulation. Next step is to see how synchronization works by adding delay to channel model.

A sliding correlator to correlate against the access code is recommended to continue,
which can be used in synchronization for each packet. When increasing bit sampling rate, the performance of demodulation is not necessary getting better since it can be affected by noise more seriously in slopes detection. It has been discussed in Fig. 4.3 and Fig. 4.4. To improve it, subtraction can be done when there are a number of samples between the two points, and the number is at most equal to the bit sampling rate. Once the distance in between has been increased when subtracting, theoretically, the influence from noise would be reduced.
6. Conclusions

This project is focused on the implementation of a behavioral model for Bluetooth baseband specifications. For this purpose, an overview of the Bluetooth Standard is needed. Since the model is implemented in C programming, studies of algorithms that are related should also be done precisely.

As a conclusion, this final year project is a rewarding experience. I have gained a good knowledge in telecommunication field and got well trained in programming. I have also learned a lot from the discussion with my supervisor and also from searching websites and reading books too seek for solutions. I believe all these can be very useful for my career in the future.
7. **References**


[6] Craig Robinson and Alan Purvis, "Demodulation of Bluetooth GFSK Signals under Carrier Frequency Error Conditions"


Appendix A:

Background knowledge of G-FSK

Bluetooth uses Gaussian Frequency Shift Keying (G-FSK) as its modulation method. G-FSK is based on Frequency Shift Keying (FSK). See Fig. 1.

![Blocks for G-FSK implementation](image)

**Fig. 1** Blocks for G-FSK implementation

In Frequency Shift Keying the carrier frequency is shifted between distinct frequencies. A common shift pattern is between two frequencies one representing a zero and the other a one. See Fig. 2.

![FSK-modulated alternates bit stream into its frequency components.](image)

**Fig. 2** FSK-modulated alternates bit stream into its frequency components.

A general form for all FSKs is

\[
s(t) = \sqrt{\frac{2E_s}{T}} \cos(\omega_0 t + \frac{a_n h \pi (t - n)}{T} + \psi_n) \quad nT \leq t \leq (n+1)T
\]

Frequency shifting offset is

\[
\frac{d}{dt} \left( \omega_0 t + \frac{a_n h \pi (t - n)}{T} \right) = \omega_0 + \frac{a_n h \pi}{T}
\]

T is the time developed to one transmission symbol, \( E_s \) is the average symbol energy, and \( a_n \) is the \( n \)th transmission symbol.

In a binary FSK, \( a_n \) takes values in \{-1, +1\}. When in G-FSK, \( a_n \) can be reshaped as a
bell curve instead of a positive or negative binary digital values. The second term in the cosine argument represents a linear shift of phase, totaling \( a_n h \pi \) radians, across \( n \)th symbol interval. The constant \( h \) is the modulation index, the constant of proportionality between the symbols and the total phase shift. When increasing \( h \), more frequency offset is shifted and cost wider bandwidth. The third factor, \( \psi_n \), is used to ensure phase continuity between intervals.

A brief introduction of phase continuity is given:

1. In discontinuous-phase narrowband FSK, \( 0 < h < 1 \), and \( \psi_n = 0 \). The phase offset begins from zero at the beginning of each internal and there is generally a discontinuity at the beginning of the next interval, see Fig. 3.16(a). This FSK is seldom used because its bandwidth can be narrowed further by removing the phase discontinuities.

2. In continuous-phase narrowband FSK, \( 0 < h < 1 \), and \( \psi_n \) is chosen so that the carrier phase is continuous at every interval boundary, see Fig. 3. The continuity is guaranteed by setting

\[
\psi_n = \pi h \sum_{i < n} a_i
\]

The signals have excellent error and bandwidth performance.

![Excess phase \( \psi(t) \) for (a) discontinuous-phase narrowband FSK and (b) continuous-phase FSK with the same data](image)

In Fig. 4, some disconnected points are shown for discontinuous-phase, which would cost a large bandwidth.
Gaussian filtering is used in G-FSK. It is one of the very standard ways for reducing the spectral width called Pulse Shaping. If using -1 for \((f_{\text{carrier}} - f_{\text{offset}})\) and 1 for \((f_{\text{carrier}} + f_{\text{offset}})\), once when it jumps from -1 to 1 or 1 to -1, the modulated waveform changes rapidly, which introduces large out-of-band spectrum. If changing the pulse going from -1 to 1 as -1, -.98, -.93 ... .96, .99, 1, and using this smoother pulse to modulate the carrier, the out-of-band spectrum will be reduced.

As mentioned in the form for FSK, \(a_n\) can be reshaped as a bell curve, Fig. 5 shows the digital input \(a_n\) and the digital input after it has passed through the Gaussian filter.
**Fig. 5** Digital input $a_n$ has passed through the Gaussian filter.

In Fig. 6 shows the waveform for GFSK signal output. Note the phase discontinuities.

**Fig. 6** Waveform for GFSK signal output
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Ying-Chi Kuo