ATCZ175 INTEROP PROJECT

Guidelines for Bluetooth Low Energy Matlab Codes

Version 1.1

Institute of Electrodynamics, Microwave and Circuit Engineering
TECHNISCHE UNIVERSITÄT WIEN

Advisor:
Assoc. Prof. Dipl.-Ing. Dr.techn. Holger ARTHABER

by
Proj. Ass. Dipl.-Ing. Hamid KAVOUSI GHAFI

March 5, 2019

Abstract

As a part of the ATCZ175 InterOP project, interference in the physical layer of Bluetooth Low Energy (BLE) systems is analyzed. In the first step, a basic simulation framework is considered. This framework consists of four basic functions to modulate and demodulate bit sequences. These sequences are either random (GFSK_modulator and GFSK_demodulator functions) or in the format of BLE link layer packets (BLE_modulator and BLE_demodulator functions). For BER (Bit Error Rate) calculation, the first two functions (GFSK_modulator and GFSK_demodulator) are used. For PER (Packet Error Rate) calculation, the other two functions (BLE_modulator and BLE_demodulator) are deployed. The functionality of these basic functions are described in section 1. In this section, the input and output arguments of each function are also discussed. Furthermore, these basic functions are used to calculate BER and PER for different interference scenarios. In section 2 some Matlab codes regarding to the interference effects are presented.

1 Simulation Framework for BLE

This section contains four basic functions in simulating BLE systems:

* GFSK_modulator
* GFSK_demodulator
* BLE_modulator
* BLE_demodulator

The first two functions are used to modulate and demodulate a random message bit sequence using the GFSK algorithm. The second two are implemented to simulate Bluetooth low energy packets. For BLE_modulator and BLE_demodulator the structure of a Link Layer BLE packet is considered. For more information about the BLE packet structure refer to the Bluetooth Core Specification, v5.0, pp. 2560-2570.

1.1 GFSK_modulator

modulated_signal = GFSK_modulator(tx_bits, f_sample, varargin) modulates the message data bits (tx_bits) using the GFSK modulation. The function returns a complex output signal. This function has two types of input arguments, required and optional. Required input arguments are:

- tx_bits: is an arbitrary length row vector of binary data. The example at the end of this section shows how to assign a value to this parameter.
- f_sample: represents the sampling frequency (Hz).

This function has also some optional input arguments. When an optional input argument does not include a value, the default value is assigned to that. Optional input arguments for this function are:

⋆ bitrate: specifies the data bit rate. For BLE systems, typical value is $1\,\mathrm{Mb/s}$. In the Matlab code, the value is in the unit of $\mathrm{b/s}$. The default value is set to 1e6. The example at the end of this section shows how to assign a value to this parameter.

⋆ modindex: represents the modulation index of the GFSK modulation. The default value is set to 0.5.

The example below shows how to assign values to the input arguments of this function. A few number of output signal values are also printed.

```
>> num_of_bits = 1e6;
>> tx_bits = randi(2,1,num_of_bits)-1;
>> f_sample = 10e6;
>> tx_signal = GFSK_modulator(tx_bits, f_sample,...
               'bitrate', 1e6, 'modindex', 0.5);
>> tx_signal(1:5)
ans =
  Columns 1 through 3
   0.9997 - 0.0236i   0.9958 - 0.0912i   0.9811 - 0.1934i
  Columns 4 through 5
   0.9484 - 0.3172i   0.8933 - 0.4494i
```

1.2 GFSK_demodulator

`demodulated_bits = GFSK_demodulator(rx_signal, f_sample, varargin)` demodulates the complex baseband `rx_signal` using a matched filter algorithm. The function returns a binary row vector. This function has two types of input arguments, required and optional. Required input arguments are:

- `rx_signal`: is a baseband GFSK modulated signal. It may be noisy and also suffer from frequency offset.
- `f_sample`: represents the sampling frequency (Hz).

This function has also some optional input arguments. When an optional input argument does not include a value, the default value is assigned to that. Optional input arguments for this function are:

- ⋆ `bitrate`: specifies the data bit rate. For BLE systems, typical value is $1\,\mathrm{Mb/s}$. In the Matlab code, the value is in the unit of $\mathrm{b/s}$. The default value is set to `1e6`. The example at the end of this section shows how to assign a value to this parameter.
- ⋆ `modindex`: represents the modulation index of the GFSK modulation. The default value is set to `0.5`.

The example below shows how to assign values to the input arguments of this function. In this example, the `tx_signal` is the output signal of the previous section (`GFSK_modulator`). To assess the demodulator against noise, a certain level of noise power is added to the `tx_signal`. Then, the noisy signal is fed to the `GFSK_demodulator` function. A few number of output data bits are also printed.

```
>> snr_dB = 15;
>> rx_signal = awgn(tx_signal, snr_dB);
>> f_sample = 10e6;
>> demodulated_bits = GFSK_demodulator(rx_signal, f_sample,...
                      'bitrate', 1e6, 'modindex', 0.5);
>> demodulated_bits(1:5)
ans =
     0     0     0     0     1
```

It is expected that the retrieved bits (`demodulated_bits`) are equal to modulated bits in previous section (`tx_bits`) or at least there exist low discrepancy between them. However, for this example all bits are demodulated correctly. As shown below, the number of unequal bits is zero:

```
 >> nnz(demodulated_bits-tx_bits)
ans =
     0
```

1.3 BLE_modulator

`modulated_signal = BLE_modulator(acc_adr, PDU, ch_number, CRC_init,... fsample, varargin)` modulates a BLE packet data bits using GFSK modulation. The function returns a complex output signal. This function has two types of input arguments, required and optional. Required input arguments are:

- `acc_adr`: represents the access address and is composed of 32 bits (see Figure 1). For advertising packets, the access address is a constant value (0x8E89BED6). The example at the end of this section shows how to assign a value to this parameter.

- `PDU`: stands for payload data which are attached to header and length data (see Figure 1). It is a vector of integer values. Each integer value represents a data byte. The second element of the `PDU`, length byte, determines the length of the payload data. For instance, in the example at the end of this section the second element of the `PDU` is 10 which means the `PDU` is a vector of 12 elements (10 payload bytes, one header byte, and one length byte). For BLE packets, the `PDU` is a vector of at least two (0 payload length) integer values. The maximum value for the `PDU` length is 39 (2 header and length bytes plus 37 payload data bytes). The example at the end of this section shows how to assign a value to this parameter.
- `ch_number`: represents the Linked Layer channel number in which the packet is transmitted. In the whitening procedure, the shift register initial value is set to this value. This value has the range of [0–39]. For more information about whitening procedure in the BLE systems refer to the Bluetooth Core Specification, v5.0, pp. 2600-2602.
- `CRC_init`: stands for the initial value of cyclic redundancy check (CRC). CRC consists of the last 24 bits of a BLE packet (see Figure 1). For more information about CRC in the BLE systems refer to the Bluetooth Core Specification, v5.0, pp. 2600-2602. The example at the end of this section shows how to assign a value to this parameter.
- `fsample`: represents the sampling frequency (Hz).

This function has also some optional input arguments. When an optional input argument does not include a value, the default value is assigned to that. Optional input arguments for this function are:

- ⋆ `whitening`: specifies if whitening is needed. The default value is set to `'on'`.
- ⋆ `bitrate`: specifies the data bit rate. For BLE systems, the typical value is $1\,\mathrm{Mb/s}$. In the Matlab code, the value is in the unit of $\mathrm{b/s}$. The default value is set to `1e6`. The example at the end of this section shows how to assign a value to this parameter.
- ⋆ `ramptime`: specifies the time (seconds) of the power ramping up and down at the beginning and end of a packet transmission (see Figure 1). The default value is set to `2e6`.
- ⋆ `stabletime`: specifies the time (seconds) of carrier stabilization before the first bit (see Figure 1). The default value is set to `2e6`.
- ⋆ `modindex`: represents the modulation index of the GFSK modulation. The default value is set to `0.5`.

The example below shows how to assign values to the input arguments of this function. A few number of output signal values are also printed.

```
>> acc_adr = hex2dec('8E89BED6');
>> PDU = [170 10 136 44 91 184 53 182 50 3 249 154];
>> ch_number = 37;
>> CRC_init = hex2dec('555555');
>> f_sample = 10e6;
>> tx_signal = BLE_modulator(acc_adr, PDU, ch_number, CRC_init,...
            f_sample, 'whitening', 'on', 'bitrate', 1e6,...
            'ramptime', 3e-6, 'stabletime', 1e-6, 'modindex', 0.5);
>> tx_signal(10:15)
ans =
  Columns 1 through 3
   0.2060 - 0.0051i   0.2499 - 0.0061i   0.2965 - 0.0073i
  Columns 4 through 6
   0.3454 - 0.0085i   0.3959 - 0.0097i   0.4476 - 0.0110i
```

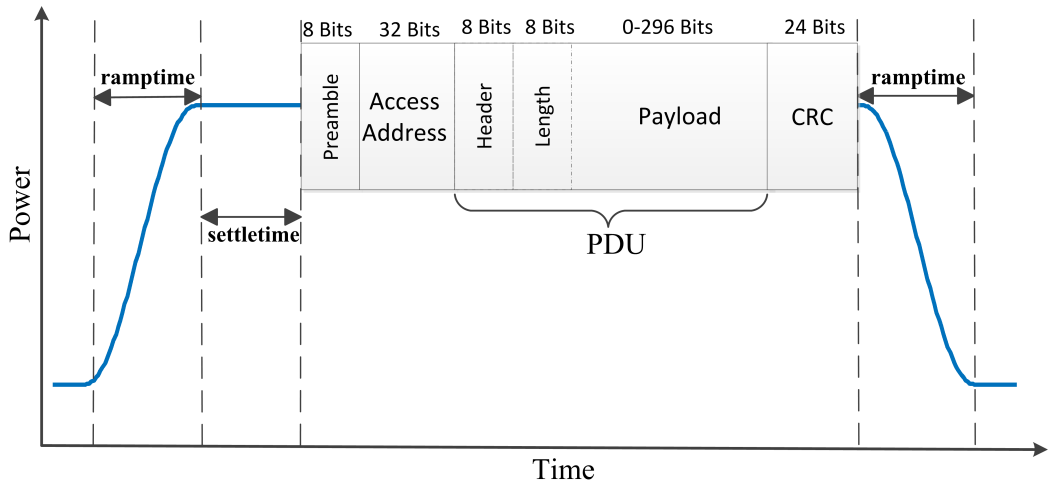

Figure 1: BLE 1M PHY packet time profile

1.4 BLE_demodulator

`demodulated_PDU_bits = BLE_demodulator(rx_signal, f_sample,... synchword_bits, ch_number, CRC_init, varargin)` demodulates the complex baseband `rx_signal` using a matched filter algorithm. The function returns a binary row vector. This function has two types of input arguments, required and optional. Required input arguments are:

- `rx_signal`: is a baseband GFSK modulated signal. It may be noisy, unsynchronized and also suffer from frequency offset.
- `f_sample`: represents the sampling frequency (Hz).
- `synchword_bits`: consists of the preamble and access address and is used to synchronize the input signal `rx_signal`. The example at the end of this section shows how to assign a value to this parameter.
- `ch_number`: represents the Linked Layer channel number in which the packet is transmitted. In the dewhitening procedure, the shift register initial value is set to this value. This value has the range of [0–39].
- `CRC_init`: stands for the initial value of cyclic redundancy check (CRC). CRC consists of the last 24 bits of a BLE packet (see Figure 1). The example at the end of this section shows how to assign a value to this parameter.

This function has also some optional input arguments. When an optional input argument does not include a value, the default value is assigned to that. Optional input arguments for this function are:

- ⋆ `dewhitening`: specifies if dewhitening is needed. The default value is set to `'on'`.
- ⋆ `bitrate`: specifies the data bit rate. For BLE systems, the typical value is 1 Mb/s. In the Matlab code, the value is in the unit of b/s. The default value is set to `1e6`. The example at the end of this section shows how to assign a value to this parameter.
- ⋆ `modindex`: represents the modulation index of the GFSK modulation. The default value is set to `0.5`.

The example below shows how to assign values to the input arguments of this function. In this example, the `tx_signal` is the output signal of the previous section (`BLE_modulator`). To assess the demodulator against noise, a certain level of noise power is added to the `tx_signal`. Then, the noisy signal

is fed to the BLE_demodulator function.

```
snr_dB = 15;
rx_signal = awgn(tx_signal, snr_dB);
f_sample = 10e6;
demodulated_PDU_bits = BLE_demodulator(rx_signal, f_sample,...
        synchword, ch_number, CRC_init, 'dewhitening', 'on',...
        'bitrate', 1e6, 'modindex', 0.5);
demodulated_PDU_bits(1:5)
ans =
     0     1     0     1     0
```

2 Interference Analysis

In this section, some interference analyzing codes are described. Each of these simulation codes uses some basic functions explained in the previous section. In the following, some of these Matlab codes are presented.

- ◇ ComputeBEROfBLEVersusSNRAndOffsetfrequency.m: analyzes the performance of a GFSK demodulator in the presence of different noise power levels and frequency offsets, using BER calculation. This m-file calls the GFSK_modulator (explained in section 1.1) and GFSK_demodulator (explained in section 1.2) functions. To change the noise level, in the Matlab code, the EbN0_dB variable is used and for tuning the frequency offset, f_offset is changed. In the Core Specification, the frequency offset is assumed not to exceed $\pm 150\,\mathrm{kHz}$. However, one may run the code for the bigger values. The results are depicted in a contour plot format.

- ◇ ComputeDelayInInterferingSignalEffectsOnBER.m: analyzes the effects of delay, between interfering GFSK signal and the desired signal, on the GFSK demodulator. The analysis are based on the BER calculation. This m-file calls the GFSK_modulator (explained in section 1.1) and GFSK_demodulator (explained in section 1.2) functions. This code computes the BER matrix over the signal to interference ratio (sir_dB_list variable in the code) and the delay over one bit (delay_sample variable in the code). The delay_sample is in the range of [0 – 1]. The zero delay means that the interfering signal bits are coincident with the bits of the desired signal. Using a fractional delay filter has lead to the capability of precise tuning of the delay parameter. Consequently, the delay variable (delay_sample) can be even a non-integer multiple of the sample period. It is also possible to run the simulation in the presence of a level of noise power (noisy desired signal). Per default, noise is ignored and snr_dB is set to inf in the Matlab code.

- ◇ ComputeFrequencyOffsetInInterferingSignalEffectsOnBER.m: evaluates effects of frequency offset (in the interfering signal) on the GFSK demodulator. The analysis are based on the BER calculation. This m-file calls the GFSK_modulator (explained in section 1.1) and GFSK_demodulator (explained in section 1.2) functions. This code computes the BER matrix over the signal to interference ratio (sir_dB_list variable in the code) and frequency offset of interfering signal (delta_f variable in the code). In the Matlab code, delta_f corresponds to the frequency offset of the interfering signal and f_offset represents the frequency offset of the desired signal. It is also possible to run the simulation in the presence of a level of noise power (noisy desired signal). Per default, noise is ignored and snr_dB is set to inf in the Matlab code.

◇ `ComputePhaseInInterferingSignalEffectsOnBER.m`: evaluates effects of phase shift in GFSK interfering signal on the GFSK demodulator. These analysis are based on the BER calculation. This m-file calls the `GFSK_modulator` (explained in section 1.1) and `GFSK_demodulator` (explained in section 1.2) functions. This code computes the BER matrix over the signal to interference ratio (`sir_dB_list` variable in the code) and phase shift of interfering signal (`delta_ph` variable in the code). It is also possible to run the simulation in the presence of a level of noise power (noisy desired signal). Per default, noise is ignored and `snr_dB` is set to `inf` in the Matlab code.

◇ `CompareNoiseAndGFSKSignalEffectsOnBER`: compares two cases. First, the interferer is a GFSK modulated signal. In the second case, the interfering signal is an AWGN signal. This comparison is based on BER.

◇ `CompareNoiseAndGFSKSignalEffectsOnPER`: compares two cases. First, the interferer is a GFSK modulated signal. In the second case, the interfering signal is an AWGN signal. This comparison is based on the PER. The simulation can be run for different packet lengths. The `payload_length` is a parameter by which the packet length can be tuned in the Matlab code.

3 Revision History

Revision	Date	Author(s)	Description
1.1	March 5, 2019	Hamid Kavousi Ghafi	Some grammatical errors and typos are corrected.
1.0	December 21, 2018	Hamid Kavousi Ghafi	First version was released.