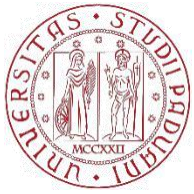

University of Padova

Department of Information Engineering

Master Thesis in ICT for Internet and Multimedia

MIoTy Overview

a mathematical description of the physical layer

Supervisor

Prof. Lorenzo Vangelista

Student

Naga Divya Sunkara
1216017

Academic Year 2021-2022
11 July 2022

Table of Contents

List of Figures

List of Tables

Abstract

MIoTy is one of the emerging technology for the Massive Internet of Things Connectivity and is a low-power, wide-area network (LPWAN) protocol that was designed to have the best-in-class reliability and scalability to support massive IoT i.e., MIoTy is a standardized massive IoT connectivity solution operating in the license-free spectrum. A single MIoTy gateway can handle up to 100,000 sensor nodes and about 1.5 million messages per day with interference immunity and transmits data with ultra-low power consumption. It is the first and only technology to comply with the ETSI telegram splitting ultra-narrow band (TS-UNB) technical specification for low throughput networks.

The contribution of this thesis is to provide a mathematical description of the Physical Layer at the transmitter side (since the related standard is describing the transmitter side only, leaving the implementation of the receiver to the user).

Chapter 1

Introduction

The Internet of Things paradigm brings us diverse information together and provides the language for the devices and applications to communicate with each other the process starts with the devices themselves which securely communicate with an Internet of Things platform. This platform integrates the data from many devices and applies analytics to share the most valuable data with applications that address industry-specific needs with hardware, sensors, and actuators that empower these devices to send and get information over a system without the necessity of human-to-human or human-to-PC communication. IoT applications have some specific requirements such as reliability, performance, quality, and long-term availability [1].

1.1 Thesis Contribution

1.1.1 LPWAN of IoT

LPWAN solutions fall broadly into two groups based on their operating spectrum: licensed vs. license-free. While all licensed spectrum LPWANs are derived from 3GPP cellular standards, the license-free landscape is much more diverse [2]. These solutions cluster around three major technologies: Ultra-Narrowband, Spread Spectrum, and Telegram Splitting.

Low power wide area network (LPWAN) technology provides the low-cost, low power, and wide-area coverage needed for robust IoT sensor networks to secure data transportation. Figure 1 depicts the comparison of LPWAN technologies.

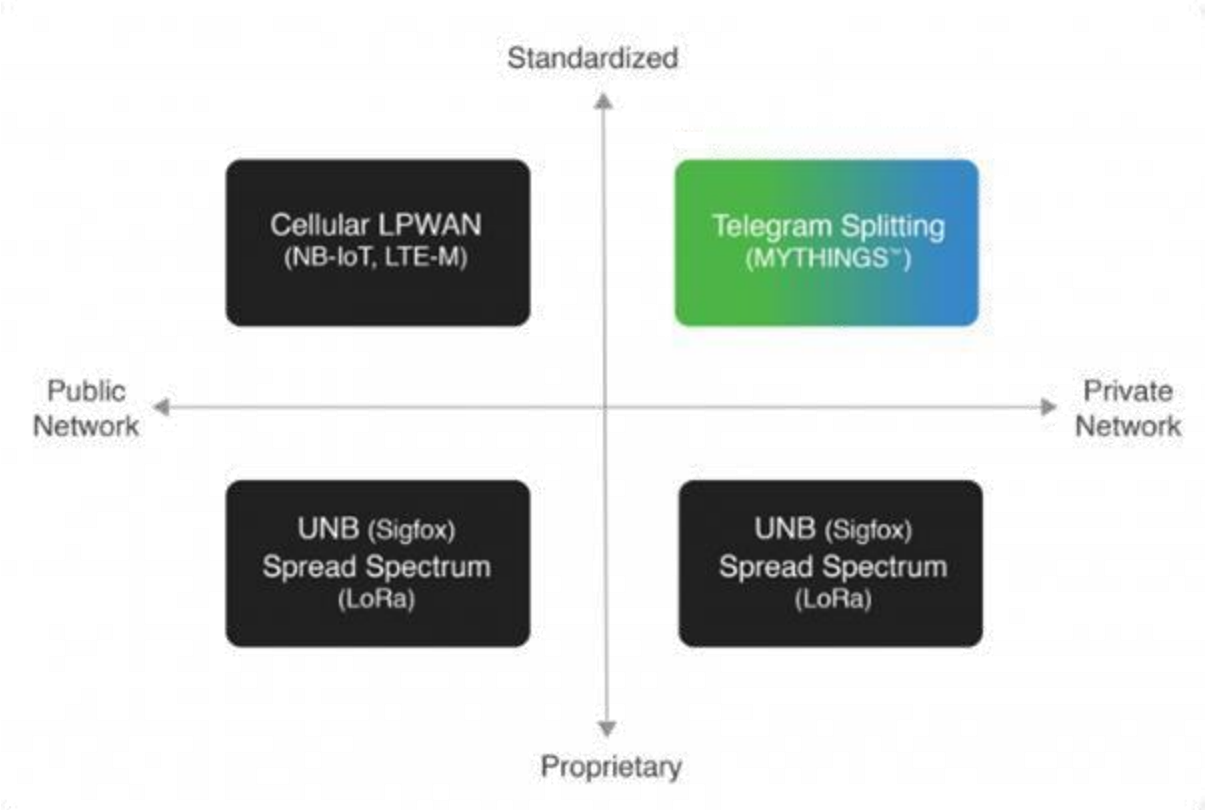

Figure 1: Comparison of LPWAN technologies [Web 1]

There are **four** main LPWAN technologies on the market that fall into two groups based on their operating spectrum, **licensed** and **license-free**, here are the main differences:

1. **Cellular LPWAN:** This technology uses the licensed spectrum and existing cellular infrastructure for data transmission since there is little co-channel interference this ensures reliable data transmission however the operation requires more complex protocols since the nodes must first get permission from the base station to send a message, this could take several attempts to get approved which can significantly increase power consumption.

2. **Traditional Ultra Narrowband:** This technology uses very little bandwidth to send messages proving to be very spectrum efficient, the low data rate enables the receiver to detect and decode messages at farther distances thereby improving range. on the other hand, this

lengthens the transmission time which increases power consumption and interference vulnerability from other systems operating in the same frequency band [3][4].

3. **Spread Spectrum:** This technology transmits a narrow signal over a wider frequency band that is hard to detect and intercept, coding is added to compensate for the high noise floor and improve receiver sensitivity, therefore, achieving long-range. However, spreading a narrowband signal over wideband results in less efficient use of the spectrum, and the risk of self-interference is high which limits network capacity [3][5].

4. **Telegram Splitting:** This technology splits an ultra-narrowband signal into multiple smaller sub-packets; these sub-packets are divided into small radio bursts and sent at different time periods and frequencies for transmission. Short airtime and pseudo-randomness minimize the likelihood of collisions with other subpackets, therefore, improving interference, resilience, and scalability [6].

1.1.2 MIoTy With Other Technologies

Low-power wide-area network (LPWAN) technologies play a pivotal role in massive IoT applications, owing to their capability to meet the key massive IoT requirements (e.g., long-range, low cost, small data volumes, massive device number, and low energy consumption). LPWAN is a wireless wide area network that is optimized for applications in machine-to-machine (M2M) and Internet of Things (IoT) communications [Web 2]. LPWAN deployments benefit M2M and IoT applications in terms of extended coverage, low transmission rate, longer battery life, cost savings in operations, and lower pricing for network services. Leading LPWAN technologies include LoRaWAN, Sigfox, RPMA, and Wi-SUN.

Proprietary LPWANs typically operate within the license-free industrial, scientific, and medical (ISM) bands and are hence more cost-effective on a per-connection basis than current mobile cellular and other licensed radio options. Latency is not the first concern; hence long-range operation is possible. However, they face strong competition from licensed LPWAN cellular-based technologies, since, at the same time, the cellular industry is additionally developing standards to support M2M and IoT deployments in an efficient manner. A number of these technologies include narrowband (NB)-IoT and long-term Evolution for Machine (LTE-M) [Web 3].

The Global LPWAN market continues to be at a growing stage; with network operators expanding public and private LPWAN networks across the world. In certain cases, operators are conducting trials of unlicensed LPWAN integrated with alternative technologies like satellites for extended coverage or roaming capabilities.

LoRaWAN

LoRa: Proprietary radio modulation to connect between end devices and gateways.

LoRaWAN: An access control Protocol working at the MAC layer (MAC-media access control) to transmit and manage messages between the LoRaWAN Network Server and the end device. LoRa is a physical layer technology that modulates the signals in the sub-GHz ISM band and deploys a proprietary spread spectrum technique. Like Sigfox, LoRa uses unlicensed ISM bands. The communication is provided via a chirp spread spectra (CSS) modulation that spreads a narrow-band signal over a wider channel bandwidth [7]. Figure 2 illustrates the architecture of the LoRaWAN network.

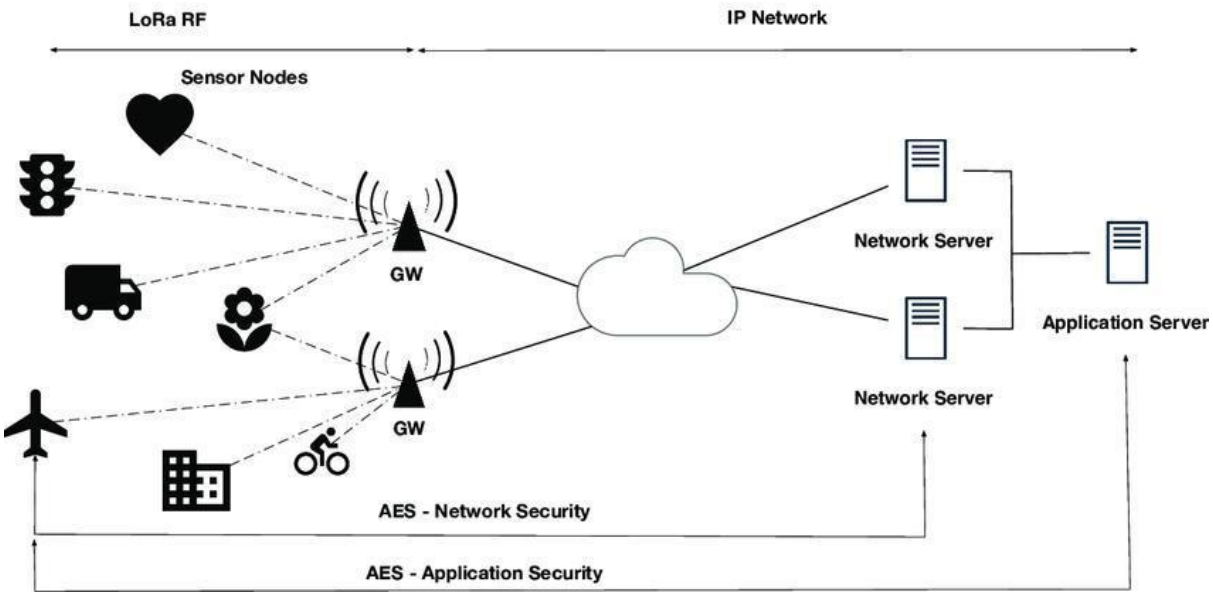

Figure 2: Architecture of LoRaWAN network [8]

Sigfox:

Sigfox is an LPWAN network operator that offers an end-to-end IoT connectivity solution supported by its patented technologies. Sigfox deploys its proprietary base stations equipped with cognitive software-defined radios and connects them to the backend servers using an IP-based network. Sigfox uses unlicensed ISM bands, for example, 868 MHz in Europe, 915 MHz in North America, and 433 MHz in Asia. The number of messages over the uplink is restricted to 140 messages per day. The longest payload length for each uplink message is 12 bytes. However, the number of messages over the downlink is limited to four messages per day, which implies that the acknowledgment of each uplink message is basically not supported [9]. Figure 3 illustrates the architecture of Sigfox network.

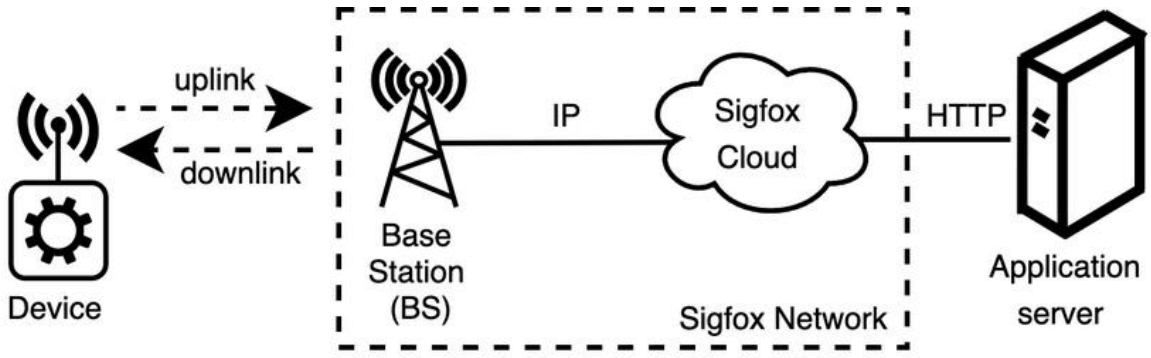

Figure 3: Architecture of Sigfox Network [10]

MIoTy:

MIoTy is a low-power, wide-area network [LPWAN] communications solution that is the first and only technology to comply with the just-released ETSI telegram splitting ultra-narrow band (TS-UNB) technical specification for low throughput networks (TS 103 357). MIoTy powers the IIoT by facilitating the last mile of communications with wireless data sources in delivering a real-world solution based on the ETSI (TS 103 357) specifications.

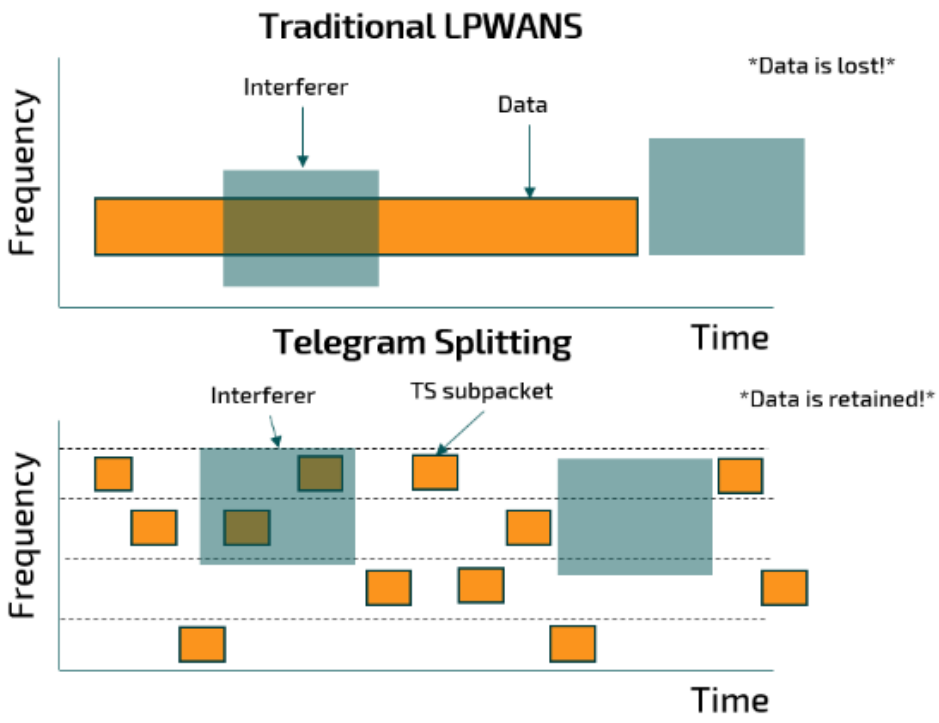

Figure 4: Framework of MIoTy technology [Web 4]

1.2 Thesis Organization

The organization of this thesis is as follows. Chapter 2 introduces the MIoTy Overview and in section 2.2 we will study the specifications of the MIoTy physical layer functions in uplink and downlink transmission. In the last section, we study the features of MIoTy technology.

In Chapter 3 we introduces the Error Detection techniques and mainly focus on one of the error detection technique uses in MIoTy physical layer and the generation of an CRC codes using Polynomials of CRC-4 bit and CRC-8 bit. By taking an input example, we will discuss how CRC bits are working in the MIoTy physical layer.

In Chapter 4 we extend our CRC code using the Forward Error Correction (FEC) in MIoTy and we will see in detail the Convolutional Codes specified in the recent standard (ETSI TS 103 357) and the general construction of Convolutional Codes by having the (2,1,2) convolutional code as an example, we will construct the 1/3 convolutional codes which is used in MIoTy physical layer.

Chapter 5 provides the information on Gaussian minimum-shift keying modulation which is used in the MIoTy physical layer. The last section in Chapter 5 is focusing on the description of a mathematical function of the physical layer.

Chapter 2

2.1 MIoTy Overview

Telegram Splitting Multiple Access (TSMA) is the main invention within the MIoTy technology. The European Telecommunications Standards Institute (ETSI) standard for telegram splitting (ETSI TS 103 357), uses an algorithm to split the data packets to be transmitted into small sub-packets at the transmission source. MIoTy is based on the protocol family telegram splitting ultra-narrow band (TS-UNB). These TSMA systems have a data rate of 512 bit/s. The ultra-narrow band (UNB) telegram is split at the physical layer into multiple sub-packets, each equal in size. Each of which is randomly sent on a unique carrier frequency and at a unique time. The sub-packets are much smaller than the initial telegram and only require an on-air time of 16 ms. The overall airtime of all the sub-packets at transmission time for a 10-byte telegram is about 390 ms [11]. The network capacity of over one million telegrams per base station (within a 200 kHz spectrum) per day is supported because of the short transmission time of the UNB telegrams as well as the extremely high interference resilience of the system.

The risk of suffering data loss resulting from interference is substantially reduced due to the virtually random distribution of sub-packet transmissions through time and varying frequencies. And, because of the use of sophisticated forward error correction (FEC) techniques, the receiver needs only about 50% of the packets to reconstruct the original telegram completely. The result is a much lower impact of corrupted or lost packets (and therefore, the whole telegram), due to collisions and greatly increased immunity to interference than existing LPWAN technologies. Packet error rates (PER) over 10% [12] are not unusual with traditional LPWAN solutions. With MIoTy, Packet Error Rate is typically under 1%. MIoTy uses 100kHz to 1.5MHz in the worldwide license-free spectrum.

The telegram splitting ultra-narrow band (TS-UNB) protocol offers several modes of operation in uplink and downlink. If a low-throughput network (LTN) System is using the TS-UNB protocol, it shall implement one subset of the different modes of operation according to table 1.

TSMA Mode	Channel Bandwidth	Carrier Spacing step size B_c	Occupied bandwidth per frame
Narrow	25 kHz	396,729 Hz	12,616 kHz
Standard	100 kHz	2 380,371 Hz	60,223 kHz
Wide	725 kHz	28 564,453 Hz	688,641 kHz

Table 1: Different TSMA Modes of operations[ETSI TS 103 357]

2.2 MIoTy Physical Layer

2.2.1 Overview of a Physical layer

In a telegram splitting ultra-narrow band (TS-UNB), the radio transmission of a packet is split into several radio bursts, which are distributed over time and frequency within the radio frame. For a generation of the physical layer output, the following processing blocks according to Figure 5 are necessary for the end-point (uplink transmission) and base station (downlink transmission). In this thesis, we will study some blocks of physical waveform illustrate the figure 6.

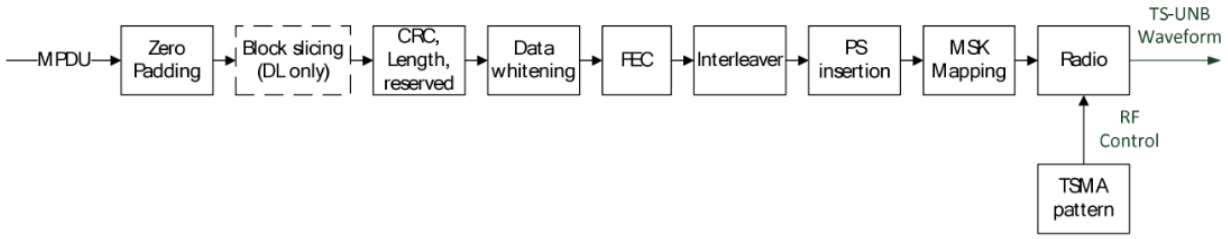

Figure 5: Block diagram of Physical waveform generation [ETSI TS 103 357]

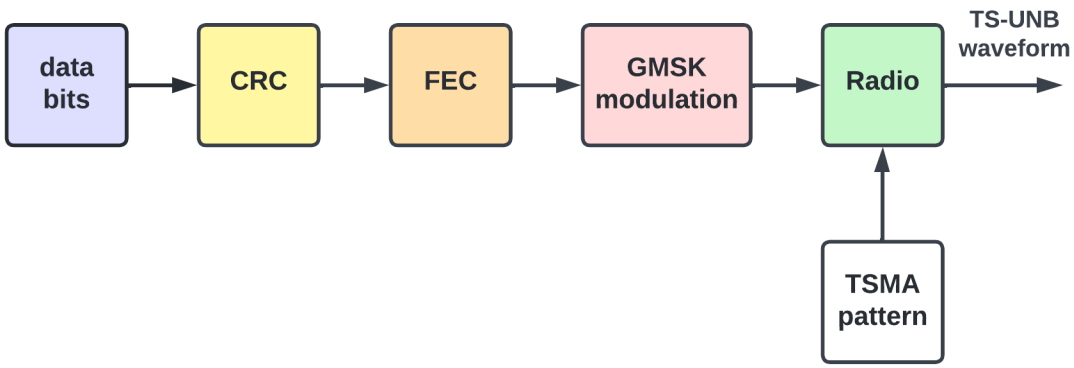

Figure 6: MIoTy physical layer transmitter waveform

The TSMA protocol supports Class Z (uplink only) and Class A (bidirectional) end-points. The downlink communication is triggered by an uplink transmission. After the reception of uplink transmission, the base station may send a downlink transmission after a defined period. To further keep transmission duration short, especially for battery or energy harvesting operated end-points, channel coding together with coherent MSK or GMSK modulation is used, which give the ability to detect signal in very low signal to noise ratio. Further, we will discuss in detail of physical waveform generation (processing of blocks) in later chapters.

2.3 Features of MIoTy

1. ***Scalability***: MIoTy operates without interference, and a single network can have hundreds of thousands of devices, potentially over one million devices, transmitting up to 1.5 million telegrams per day. Networks can start small and grow over time.

2. ***Robustness:*** The robustness of MIoTy supports nodes in hard-to-reach areas with poor propagation properties and physical obstacles.

3. ***Battery Efficiency:*** Thanks to the Telegram Splitting technology, the sub-packets sent from the sensors to the gateway have very short airtime. This allows for very low power battery operation. MIoTy has the longest battery life of any other existing LPWAN technology which makes it possible to have massive networks in hard-to-reach, noisy environments.

Chapter 3

Error Detection Codes

In the physical layer wave form generation, the second block performs the error detection code on the received information data bits from the previous layer (MAC layer) illustrate in figure 7. In this chapter we will study the definition of the error detection codes and different types of detection codes. After, that we study about the CRC code, which is used in MIoTy physical layer. CRC 4-bit and CRC 8-bit are the specifications of MIoTy physical layer transmitter.

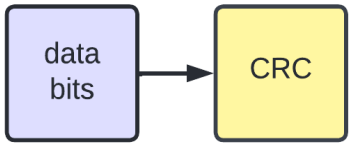

Figure 7: Block generation of CRC data bits

Error detection is the detection of errors caused by noise or other impairments during data transmission from the transmitter to the receiver. In digital communication system errors are transferred from one communication system to another, along with the data. If these errors are not detected and corrected, then the data will be lost. For effective communication, data should be transferred with high accuracy [13]. This can be achieved by first detecting the errors and then correcting them. We use some redundancy codes to detect the errors, by adding to the data while it is transmitted from the source (transmitter). These codes are called “Error detecting codes" [14].

3.1 Error Detection Methods

- Vertical Redundancy Check (VRC)
- Longitudinal Redundancy Check (LRC)
- Check Sum
- Cyclic Redundancy Check (CRC)

Vertical Redundancy Check (VRC):

The Vertical Redundancy Check (VRC) is also known as Parity Checking. A way of trying to establish if binary data has changed during transmission. Parity bits are generally applied to the smallest units of a communication protocol, typically 8-bit octets (bytes), although they can also be applied separately to an entire message string of bits. Before adding the parity bit, several 1's or 0's is calculated in the data. Based on this calculation of data an extra bit is added to the actual information or data. The addition of a parity bit to the data will result in a change in data string size [16].

There are two types of parity bits in error detection, they are

1. Even bit parity
2. Odd bit parity

Even bit parity:

- If the information has an even number of 1's, the parity check bit is **0.**

  Ex: data is 10000001 -> parity check bit 0

- If the information has an odd number of 1's, the parity check bit is **1**.

  Ex: data is 10010001 -> parity check bit 1

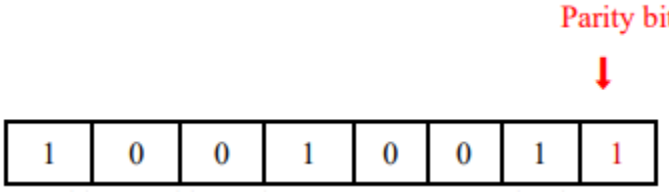

Table 2: Adding a bit to get even parity bit data

Odd bit parity:

- If the information has an even number of 1's, the parity check bit is **0.**

  Ex: data is 10011101 -> parity check bit 0

- If the information has an odd number of 1's, the parity check bit is **1**.

  Ex: data is 10010101 -> parity check bit 1

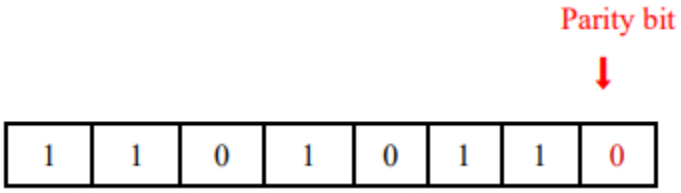

Table 3: Adding a bit to get odd parity bit data

The circuit which adds a parity bit to the data at the transmitter is called a "Parity generator". The parity bits are transmitted, and they are checked at the receiver. If the parity bits sent at the transmitter and the parity bits received at the receiver are not equal, then an error is detected. The circuit which checks the parity at the receiver is called the "Parity checker"

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Even parity	Odd parity
1	0	1	1	0	0	1	0	0	1
1	1	0	0	1	0	0	0	1	0
1	1	1	1	1	0	1	1	1	0
1	0	1	1	1	1	1	0	0	1
0	0	1	0	1	0	1	0	1	0
0	1	1	1	0	1	0	1	1	0
0	1	0	1	0	0	1	1	0	1

Table 4: Messages with even bit and odd bit parity

Longitude Redundancy Check (LRC):

Parity check bits are computed for every row and then computed for all columns. Afterward, both are transmitted together with the information. These computed odd-even check bits for both rows and columns are compared with the parity bits computed on the received data. It is also known as two-dimensional parity. This method can easily detect burst and single-bit errors and fails to detect the 2-bit errors within the same vertical slice.

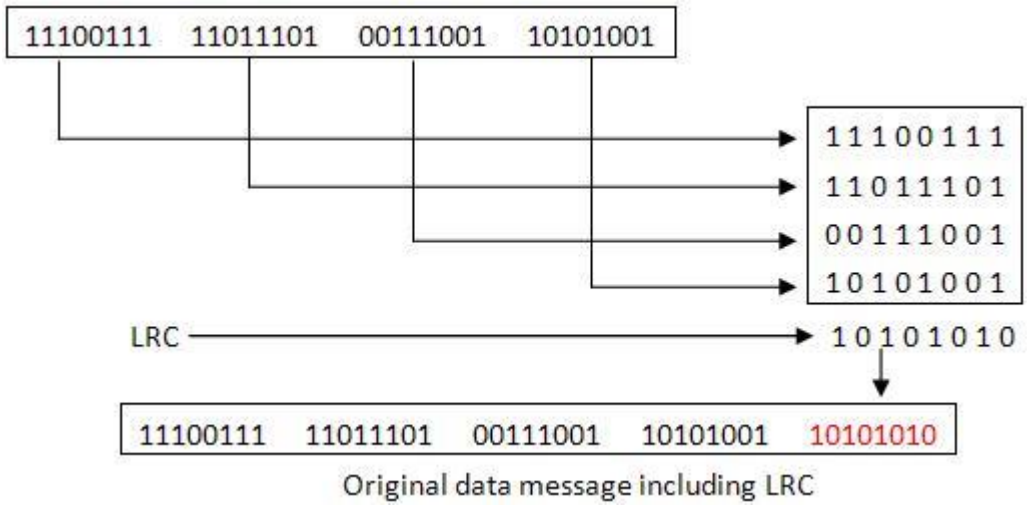

Figure 8: LRC Error Detection data transmission [Web 5]

Checksum:

The name itself says that checksum means, it is going to check the summation of the data. The transmitter is to create the message and with the message, it is going to calculate the checksum and attach that checksum to the original message and on the receiver side, the checksum is validated. So, after validation, if the receiver finds out that there are no errors, then the packet is accepted, or the data is accepted. Otherwise, the data is rejected.

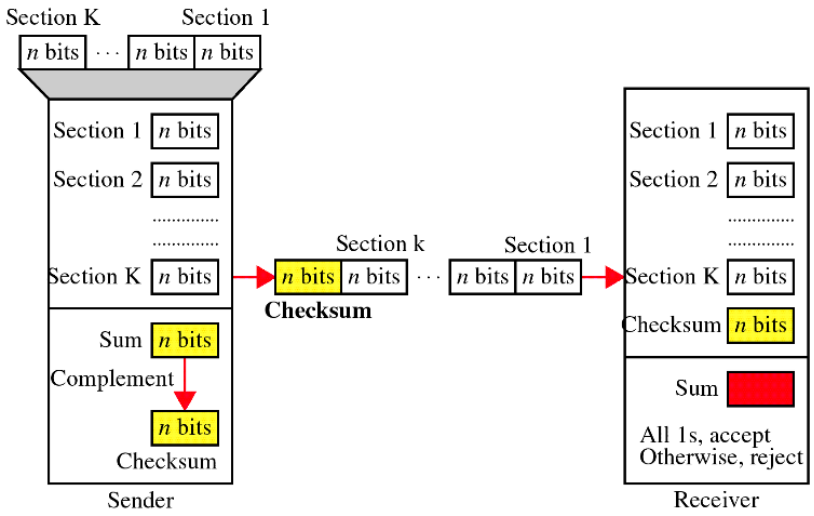

Figure 9: Operation of checksum error [Web 6]

Cyclic Redundancy Check (CRC):

Cyclic Redundancy Codes (CRCs) provide a first line of defense against data corruption in many networks. A CRC can be thought of as a non-secure function for a data word that can be used to detect data corruption. These types of codes are used for error detection and encoding. CRC codes will provide an effective and high level of protection. CRC is widely used in data communications, data storage, and data compression as a powerful method for detecting errors in the data [17].

The General CRC Generator block generates cyclic redundancy check (CRC) code bits for each input data frame and appends them to the frame. In MIoTy physical layer the CRC codes are used by generating the CRC 4-bit [15] and CRC 8-bit. In uplink the TSMA protocol uses an 8-bit cyclic redundancy check code. In downlink core frame, the TSMA protocol uses a 4-bit cyclic redundancy check and in extension frame, the same which is used in the uplink function an 8-bit cyclic redundancy check is used. We will see the generation process of CRC codes .

3.2 Generation of CRC codes

Based on the desired number of bit checks, an arrangement of repetitious bits added some zeros (0) to the original message data. This new binary data sequence is divided by a new word of length n + 1, where n is the number of check bits to be added. The reminder obtained because of this modulo-2 division is added to the dividend bit sequence to form the cyclic code. The generated code word is completely divisible by the divisor that is used in the generation of code. This is transmitted through the transmitter [18].

Example

Generator Polynomial **(G)** = 10011,

message **(M)** after 4 zero bits are appended = 11010110110000

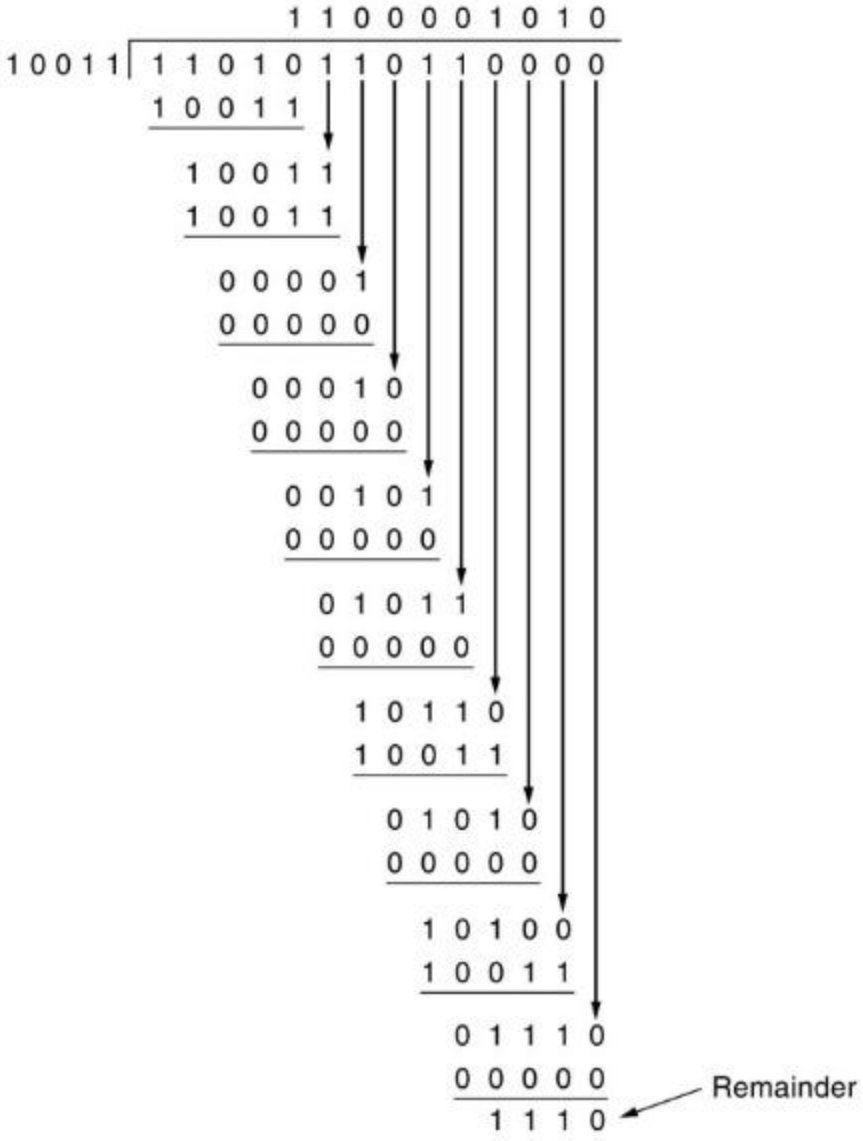

Figure 10: CRC calculation using polynomial long division [Web 7]

The remainder (r) and a CRC code generator = **1110**

CRC generated data is **1110**

Original message is **1101011011**

After generator of a CRC code, we will transmit the original message along with the generated CRC code or data i.e., the transmitter, transmits a message is **11010110111110**

3.2.1 Polynomial representation of CRC codes

Mathematically, a CRC can be described as treating a binary data word as a polynomial over GF (2) [Web 8] (i.e., with each polynomial coefficient being zero or one) and performing polynomial division by a generator polynomial G(x), which is commonly called a CRC polynomial (CRC polynomials are also known as feedback polynomials). Can be easily implemented with a small amount of hardware with Shift registers and XOR (for addition and subtraction). CRCs are so-called because the data verification (check) value expands the message without adding information (redundancy) and the algorithm is based on cyclic codes [19].

Generation of CRC 4-bit polynomial:

To start, think of an (n+1) bit message as being represented by a degree of polynomial, that is, a polynomial whose highest-order term is x^n . The message is represented by a polynomial by using the value of each bit in the message as the coefficient for each term in the polynomial, starting with the most significant bit to represent the highest-order term. For example, a 4-bit message consist of the bits M(x) is 110101111 corresponds to the generated polynomial for the CRC-4-ITU, 0x3 =$x^4 + x + 1$.

$$G(x) = x^4 + x + 1$$

By expanding the generated polynomial.

$$G(x) = x^4 + x^3 + x^2 + x^1 + x^0$$

$$G(x) = (1 \times x^4) + (0 \times x^3) + (0 \times x^2) + (1 \times x^1) + (1 \times x^0)$$

$$G(x) = x^4 + x + 1$$

For suppose the calculating of CRC, the transmitter and the receiver should have a common divisor called $C(x)$ or G(x) . G(x) is a polynomial of degree k. Here, the generation of CRC 4-bit is G(x) = $x^4 + x + 1$, in this case, the degree value is k = 4 [20].

The transmitter transmits a message $M(x)$ that is (n + 1) bits long, along with the message of (n+1)-bit plus k bits is transmitted. The complete transmitted message also includes the redundant bits $P(x)$. The polynomial represents $P(x)$ exactly divisible by $C(x)$. If the $P(x)$ is transmitted over a link and there are no errors introduced during the transmission, then the receiver should be able to

divide $P(x)$ by $C(x)$ exactly, leaving a remainder of zero. On other hand, if some error is introduced $P(x)$ during the transmission, then the receiver polynomial will no longer be exactly divisible by $C(x)$, and thus the receiver will obtain a non-zero remainder implying that an error occurred. Figure 11 shows the binary division of generated polynomial for CRC 4-bit length [21].

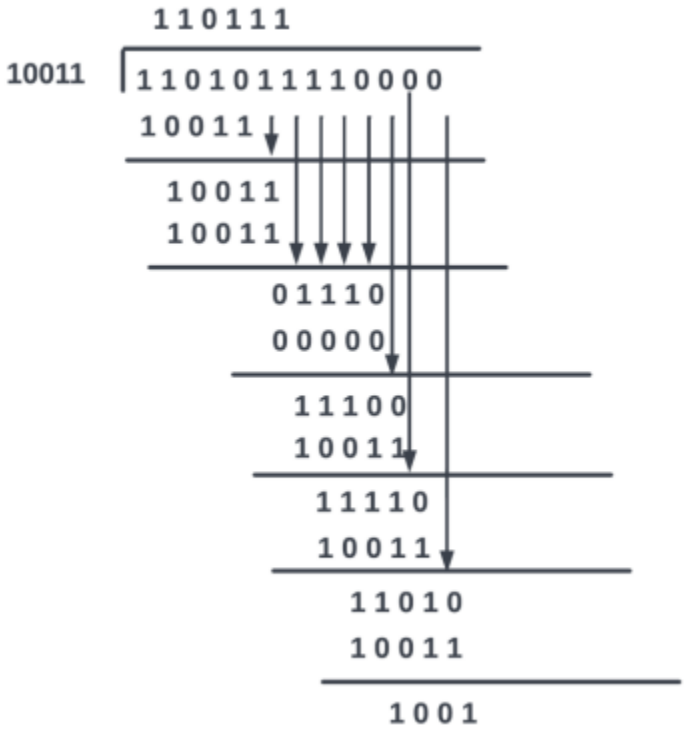

Figure 11: CRC 4-bit transmitter generator

From here, CRC = **1001**

Now,

- The code word to be transmitted is obtained by replacing the last 4 zeroes of 1101011110000 with the CRC.
- Thus, the code word transmitted to the receiver = **1101011111001**.

The CRC algorithm, while seemingly complex, is easily implemented in hardware using a k-bit shift registers and XOR gates. The number of bits in the shift register equals the degree of the generator polynomial (k). Figure 12 shows the hardware implementation of the CRC 4-bit, by using a generator polynomial $x^4 + x + 1$. The message is shifted in from the left to right, beginning with the most significant bit and ending with the string of k zeros that is attached to the message. Likewise, other CRC bits are performed based on their degree polynomial and generator polynomial [22].

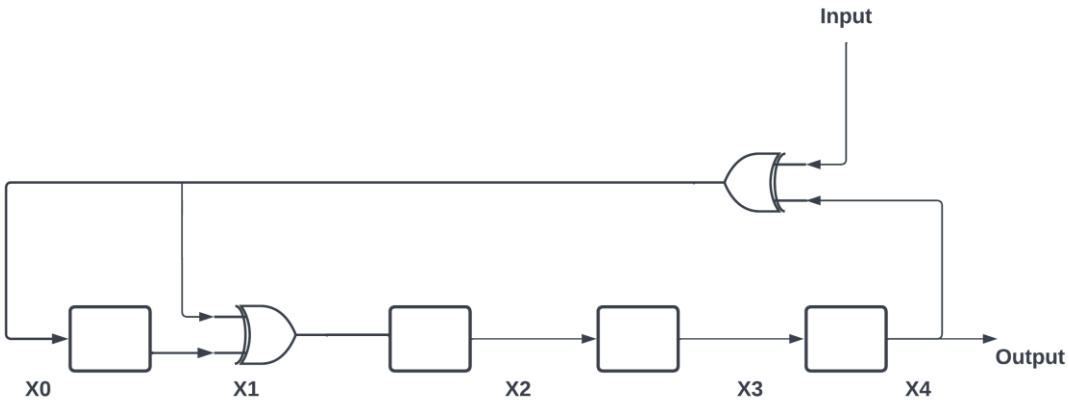

Figure 12: Hardware implementation of CRC 4-bit

Generation of CRC 8-bit polynomial:

The parity bits (defined in section 3.1) are generated using a CRC encoder using the cyclic generator polynomials. The general form for the encoder is shown in figure 13, which includes the generator for the different CRC code lengths to be used in the WCDMA system [24]. The CRC decoder operates by dividing the received codewords by the same polynomial as used in the transmitter, using a shift register structure like that used in the transmitter.

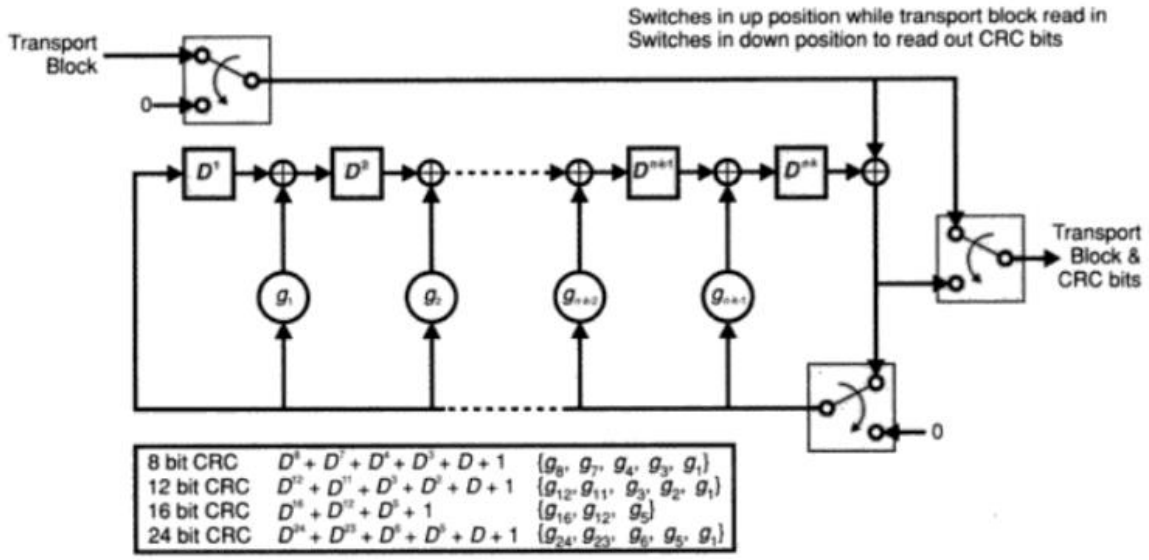

Figure 13: General form for the encoder

The CRC values are based on an 8-bit polynomial, which is defined as $x^8 + x^7 + x^4 + x^3 + x + 1$. To place this polynomial in the table, arrange the polynomial as $1 + x + x^3 + x^4 + x^7 + x^8$.

A value can be obtained from the polynomial if the coefficients of each power of x are arranged in the order shown in **table 6** [23]. That coefficient only provides the size of the CRC result. The value for the CRC 8-bit turns out to be 8C (hexadecimal). For the remainder of this document, when referring to the polynomial, this is the value used [25].

x^0	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8
1	1	0	1	1	0	0	1	1

Table 5: Coefficients for CRC 8-bit polynomial

The message bits for 8-bits are 110101111 and the generator polynomial is 110011011 and the degree value $\boldsymbol{k = 8}$.

The constraints for CRC 8-bits are

$$G(x) = x^8 + x^7 + x^4 + x^3 + x + 1$$

$$M(x) = 110101111$$

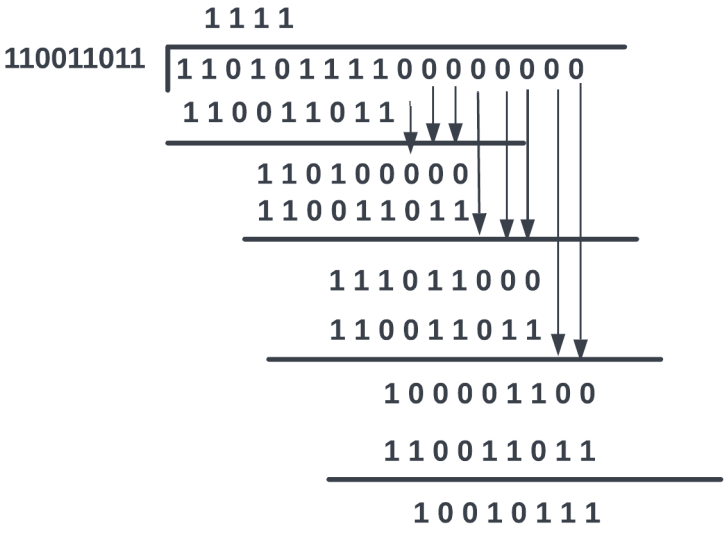

Figure 14: CRC 8-bit generator transmitter

From this the CRC = **10010111** Now,

- The code word to be transmitted is obtained by replacing the last 8 zeroes of 11010111100000000 with the CRC.
- Thus, the code word transmitted to the receiver is as follows, T(x) = 11010111110010111.

Figure 15 shows the hardware implementation of the CRC 8-bit, by using a generator polynomial $x^8 + x^7 + x^4 + x^3 + x + 1$.

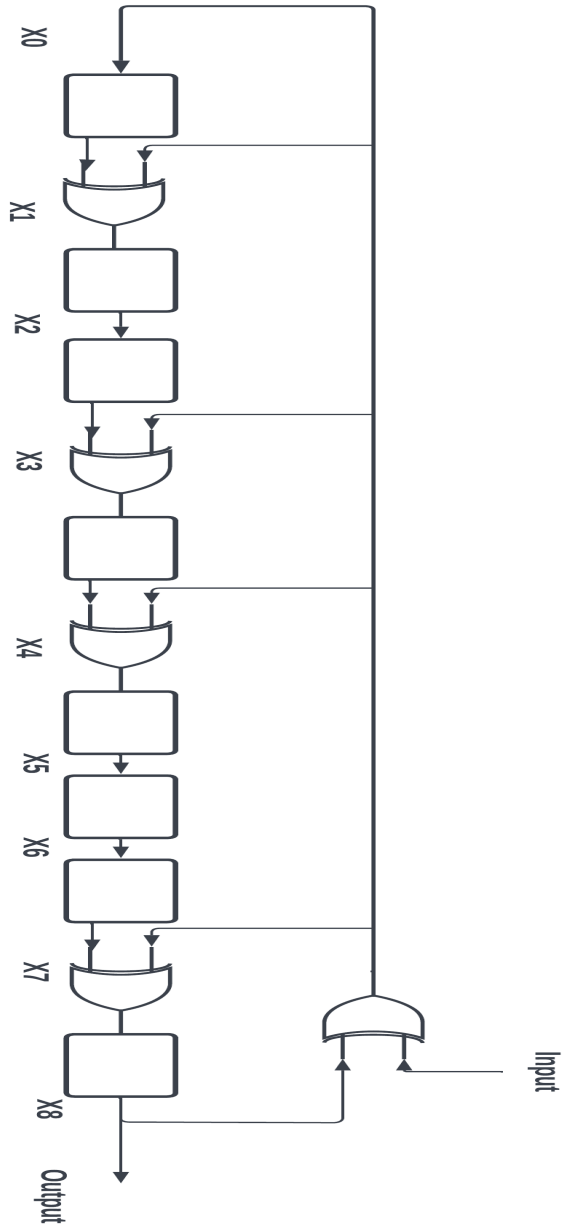

Figure 15: Hardware implementation of CRC 8-bit

Chapter 4

Error Correction codes

The third block of physical layer waveform performs the error correction on the received information data bits from the previous second block of error detection (CRC) code. In this chapter we study about the Forward Error Correction (FEC) codes which is used in MIoTy physical layer and the specification of MIoTy physical layer in this specific block of the physical waveform is to use 1/3 convolutional code. We will construct the hardware implementation of the FEC 1/3 convolutional code using trellis codes. In MIoTy physical layer by using this 1/3 convolutional code , 50% of the data packets will be secured.

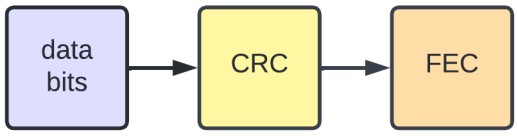

Figure 16: FEC generation block of the waveform

4.1 Forward Error Correction

FEC (Forward Error Correction) or channel coding is a technique for obtaining error control in data transmission over noisy communication channels, in which, the source (transmitter) sends redundant data to the destination (receiver) [26].

The Forward Error Correction code (FEC) is calculated over the complete physical layer payload after block code (CRC). The FEC extends the original series of 186 - 826 uncoded bits to a series of 576 - 2 496 encoded bits in the uplink and in the downlink of the core frame-block after FEC the sequence of 78 bits results in 252 encoded bits. In the downlink extension frame, one block can transmit 66 to 202 uncoded bits, which results in 216 to 624 encoded bits after the FEC. For example, in block codes, the transmitted bit stream is divided into blocks of k bits. Each block is then appended with r parity bits to form an n-bit codeword. This is called a (n, k) code. The MIoTy physical layer uses a ⅓ convolutional code rate as we mentioned above and (3,1,7) convolutional code is implemented in section 4.3 as stated in the ETSI (TS 103 357) standard.

4.1.1 Types of Forward Error Correction

Error-correcting codes for forwarding error corrections can be broadly categorized into two types, namely, block codes and convolution codes [27].

- **Block codes**: The message is divided into fixed-sized blocks of bits to which redundant bits are added for error correction.
- **Convolutional codes**: The message comprises data streams of arbitrary length and parity symbols are generated by the sliding application of a Boolean function to the data stream [28].

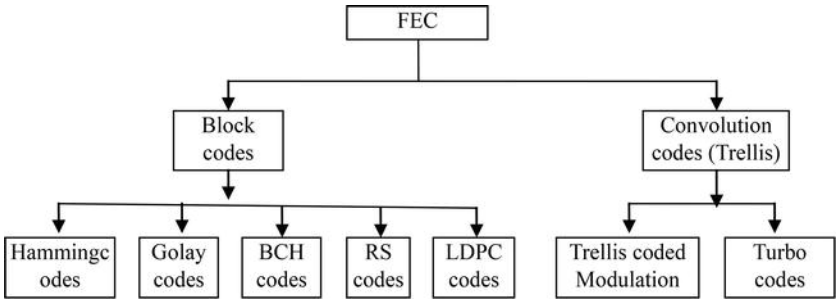

Figure 17: Classification of Forward Error Correction[Web 15]

Hamming Encoder:

This is an example of hamming encoder. The codes that Hamming devised, the single-error-correcting binary Hamming codes and their single-error-correcting, double-error-detecting extended versions marked the starting of coding theory. These codes remain important to the present day, for theoretical and practical reasons still as historical. It satisfies three conditions, several check bits should be always greater than three, and the block length is equal to $\mathbf{2^{q-1}}$, the final condition is the minimum distance between the codes is 3. The channel-coded hamming code is used for 64-bit single error detection and correction. This system provides perfect synchronization along with degradable BER (bit error rate) values [29], for different SNR (sound noise ratio) values [30].

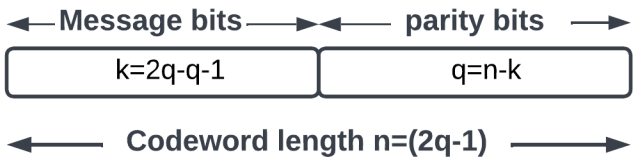

Figure 18: Hamming code word with a message and check bits

When the minimum distance i.e., d_{min}= 3 the code is said to be hamming code [Web 9]. The detection of two or more errors within the block, hamming code is one of the commonest codes used in the protection of information from error [31]. The error detection and correction capabilities of a coding technique depend on the minimum distance d_{min}

$$d_{min} \geq (j + 1) \qquad (1)$$

$$d_{min} \geq (2z + 1) \qquad (2)$$

$$d_{min} \geq (j + z + 1) \qquad (3)$$

The encoder takes n bits input data and produces a k-bit codeword. The encoder was designed through the usual generator matrix multiplication while in the decoder design the computation of

the syndrome vector were ignored. Meanwhile, the various states that may represent a specific input were calculated and therefore the decoder was designed to identify every codeword representing a specific input. Results have shown that the method is additionally reliable, Equation (1) can detect errors up to j errors per word, and equation (2**)** can correct errors up to z errors per word. Equation (3) can correct up to z errors and detect $j > z$ errors per word.

4.2 Convolutional Codes

In this section we will study the construction of the convolutional code and as mentioned earlier, in MIoTy physical layer we use only 1/3 convolutional code and it is the one of the specification in MIoTy. In this section, first we will study the general structure of convolutional codes with (2,1,2) code and will construct a (3,1,7) code using 1/3 code rate.

4.2.1 Encoder structure

Convolutional codes are widely used as channel codes in practical communication systems for error correction. The convolutional encoder processes the data sequence continuously [32]. whereas the n-bit encoder output at a specific time depends not only on the k-bit information sequence but also on K previous input blocks, i.e., a convolutional encoder has a memory order of m. The set of sequences produced by a k-input and n-output encoder of memory order m is termed an (n, k, m) convolutional code [33]. The values n, k are much smaller for convolutional codes compared to the block codes. The general structure of encoder convolutional code is shown below in figure 19.

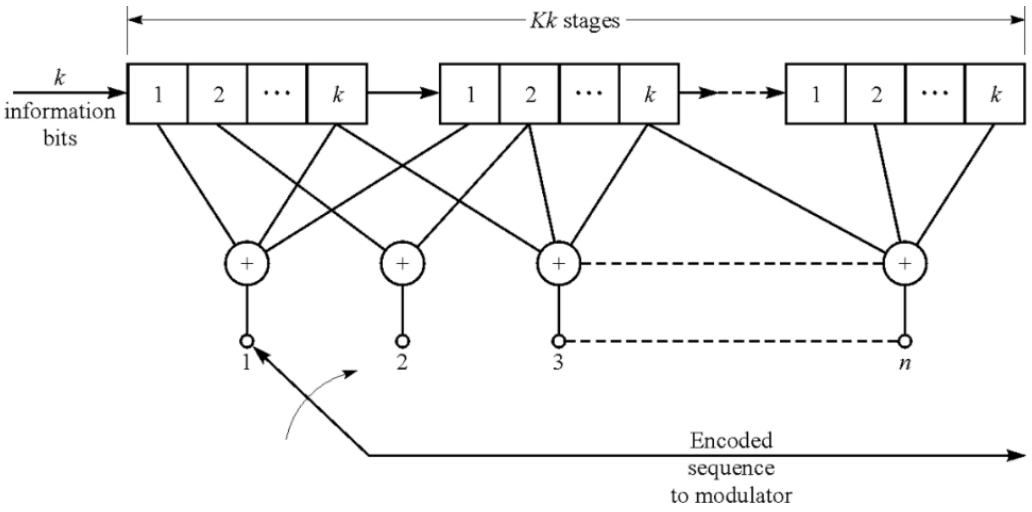

Figure 19: General structure of Convolutional codes [Web 10]

Here,

$k = number\ of\ message$ or information $bits$

$n = number\ of\ encoded\ output\ bits$

$m = number\ of\ flip - flops\ /\ size\ of\ shift\ register$

We calculate the $constraint\ length = m \times n$

$rate\ efficiency =$ K

4.2.2 Encoder Representation

4.2.2.1 Generator representation

In section 4.3.1 we see (n, k, m) convolutional code, which can be described by the generator sequences, $g^{(1)}$, $g^{(2)}$, $g^{(3)}$, etc., [34] these are the impulse responses for each output branch. The generator sequences specify Convolutional code completely by the associated generator matrix

'G'. The encoded output of the convolutional code is obtained by matrix multiplication of the input 'u' and the generator matrix 'G' [35], i.e., V=uG. The generator matrix is obtained by interlacing the generator sequence $g^{(1)}$, $g^{(2)}$as illustrated in Figure 20 and $(n, k, m) = (2,1, m)$.

$$G = \begin{bmatrix} g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & g_2^{(1)} g_2^{(2)} & \cdots & g_m^{(1)} g_m^{(2)} & 0\ 0 & 0\ 0 \\ 0\ 0 & g_0^{(1)} g_0^{(2)} & g_1^{(1)} g_1^{(2)} & \cdots & g_{m-1}^{(1)} g_{m-1}^{(2)} & g_m^{(1)} g_m^{(2)} & 0\ 0 \\ 0\ 0 & 0\ 0 & g_0^{(1)} g_0^{(2)} & \cdots & g_{m-2}^{(1)} g_{m-2}^{(2)} & g_{m-1}^{(1)} g_{m-1}^{(2)} & g_m^{(1)} g_m^{(2)} \\ & & \cdots & & & & \end{bmatrix}$$

Figure 20: Generator sequence of (2,1,m) []

The first row of G is obtained by interlacing the generator sequence g (1) and g (2) . The next row is obtained by shifting the previous row by n spaces. Here n = 2 is considered and G is the semi-infinite matrix, as the input is a continuous stream. For a finite length input (say 'x' bits), G will have 'x' rows and $n(m + x)$ columns. where 'm' is number of memory stages, 'n' is number of input bit.

4.2.2.2 Tree representation

Each tree branch represents an input symbol with the corresponding pair of output binary symbols on the branch. It is also called a code tree. Let's see an example of a (2,1,2) convolution encoder. To represent a code tree of (2,1,2) encoder we must perform the following three steps.

Step:1

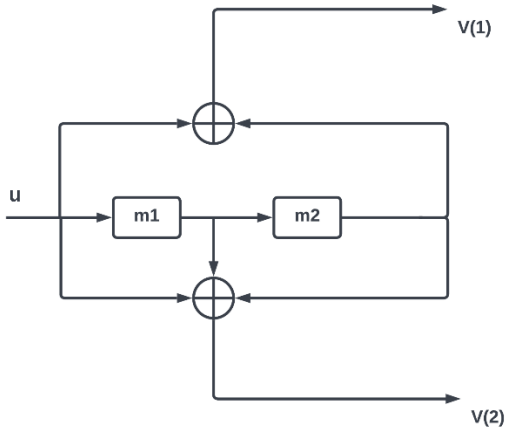

Figure 21: (2,1,2) Convolutional code representation

As there are two memory stages in the encoder, at any given instant time, the encoder will be in any of the 2^m states i.e., the encoder will be in any of the 4 states for input ('0' or '1') illustrate in the table.

Here $k = 1, n = 2, m = 2$

i.e., $2^m = 2^2 = 4$ states.

m_1	m_2	state
0	0	a
0	1	b
1	0	c
1	1	d

Table 6: Encoder possible state

Step 2: Now, we construct a response table for the (2,1,2) encoder illustrated in the following table.

Input bits u (0 & 1)	Current State of encode m_1 m_2	Next State of encoder m_1 m_2	Output of encoder for particular state: $V1\ V2$	
			$V^{(1)} = u + m_1$	$V^{(2)} = u + m_2$
0	0 0	0 0	0	0
1	0 0	1 0	1	1
0	0 1	0 0	1	1
1	0 1	1 0	0	0
0	1 0	0 1	0	1
1	1 0	1 1	1	0
0	1 1	0 1	1	0
1	1 1	1 1	0	1

Table 7: Response table of encoder (2,1,2) convolutional code

Step 3: Based on the response table for the (2,1,2) convolutional code encoder, illustrate a code tree or tree representation shown in the figure 22. For the (2,1,2) convolutional encoder there are 4 stages, and they depend on the input bits in the code generator.

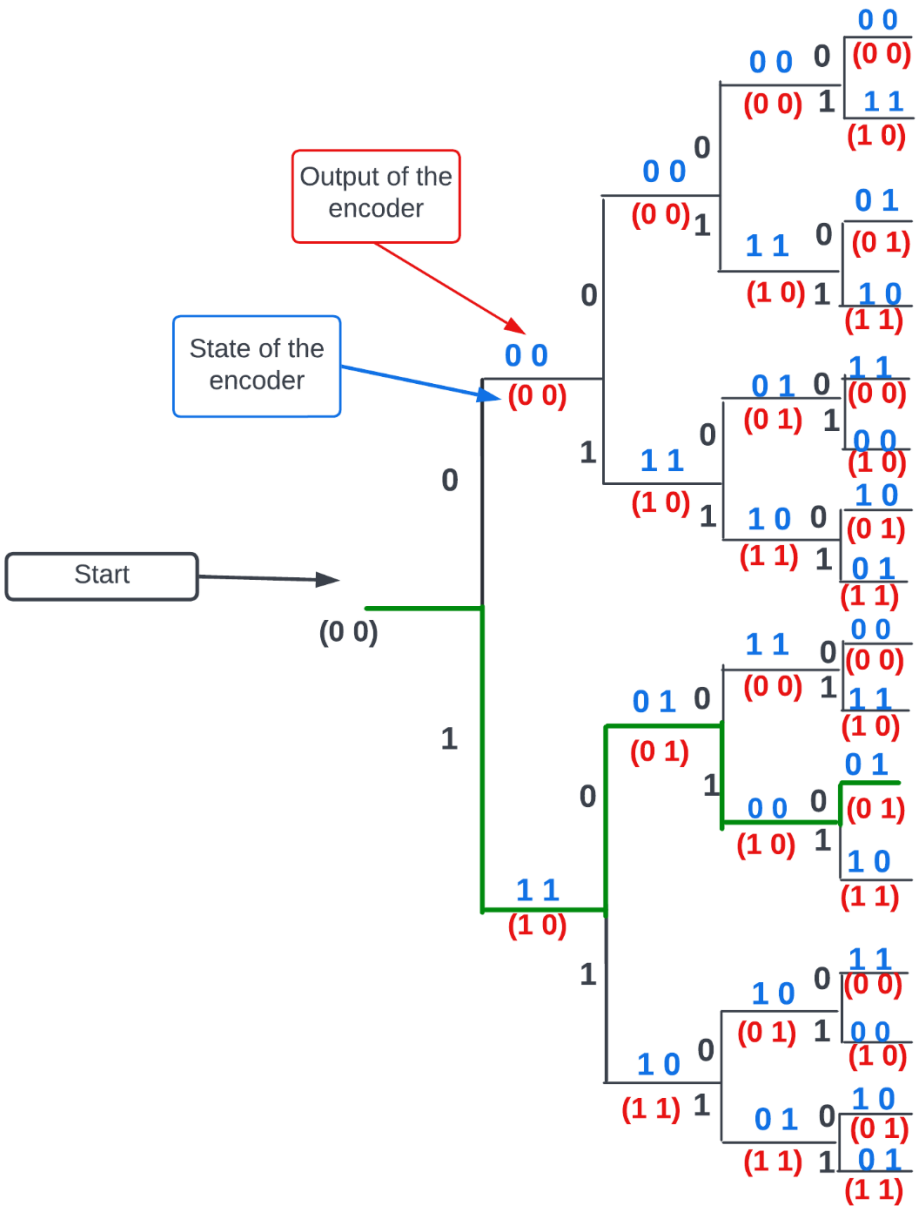

Figure 22: Tree representation of (2,1,2) convolutional encoder

The output of the encoder for the first input bit 0 is 0 0 and the states are {(0 0) (0 0)} (current state) and (next state). Likewise, the remaining bits are represented.

Example: Based on the code tree, let's see for an input bit string u = 1010 and will find out the encoded output for u = 1010 input string.

The first input bit 1 is inserted in the code tree, therefore first we start with (0 0) states and then insert the first input bit 1 and the encoded output for the first input bit is 11 refers to the code tree representation. Next, continue the path from the previous input bit 1 and insert the second input bit 0 according to the code tree representation the encoder output is 01. After that insert, the third input bit 1, and the output encoder is 00.

For the last input bit 0, the encoded output is 01 referring to the figure. The code tree for the input bit **u = 1010** of the output path, shown in the figure 22 and it plotted separated below in the figure 23.

For the input bit, **u = 1010**,

The encoded output is **11010001**

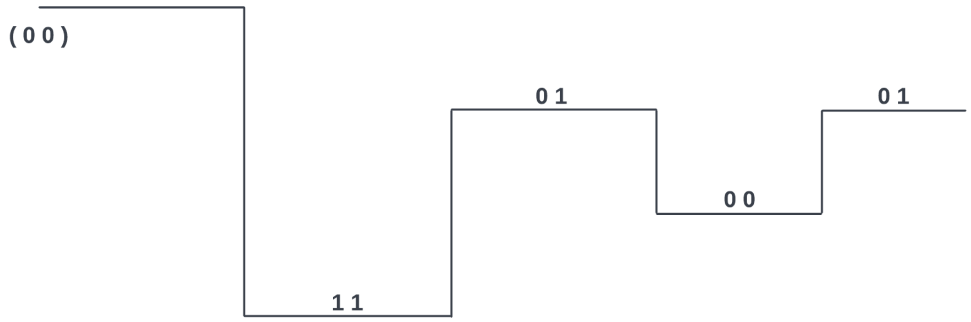

Figure 23: Tree representation of input data u=1010

After, the code tree representation for the input bit **u = 1010**, we will represent the state diagram for the same input bit **u = 1010** (section 4.4.2 in step 3 example).

4.2.2.3 State representation

The convolutional encoders are Finite State Machines (FSM). The entire behavior of the convolutional encoder can be represented using a state diagram. At any time, step, the encoder will be at any of the 2^m finite states (m is memory depth). The state of the encoder is represented by a circle ○ or box □. whenever the input changes, there will be a shifting of bits in the encoder, which also changes the state of the encoder. This transition of the state (from one to another) is represented by the lines (arrow pointing towards the next state) for input 0 it denotes the dark line and for input 1 it denotes the dotted arrow lines. Initially, the encoder will be in a ZERO state. Any input sequence is encoded by using a state diagram. The state diagram can be drawn from the response table or Code tree diagram.

From referring table 6 and the figure 22 (code tree), the state diagram will be represented by having the same example of the input bit from section 4.4.2. For the Input bit u = 1010, the state diagram is illustrated in figure 24.

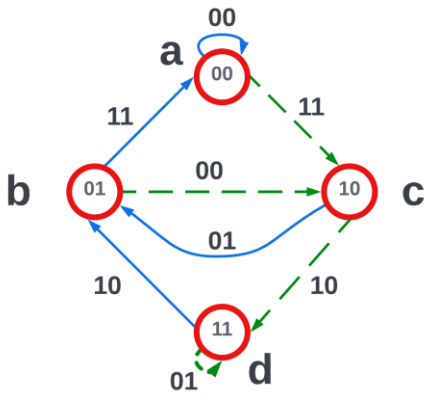

Figure 24: State diagram for (2,1,2) convolutional code

Using the state data corresponding to the above figure 24. we can encode the input bit u = 1010. We always start with (0 0) state and enter the first input bit that is 1 and follow the dotted line

indicating the next state as 10 which is the c state with the encoded output being 11. At the c state, enter the second input bit 0 it follows to the next state 01 or b state with the dark line and the encoded output is 01. At the b state, enter the third input bit 1 and it follows to the next state 10 or c state with dotted lines and the encoded output is 00. At c state, enter the last input bit 0 and it follows the next state to b state or 01 state indicating the dark line arrow, the encoded output is 01. From the state diagram of (2,1,2) convolutional code, for the input bit u = 1010 the encoded output bit is **11010001**. Notice that the Encoded output for the input bit u = 1010 is the same for the code tree or state diagram representation.

4.2.2.4 Trellis representation

Trellis diagrams represent linear time sequencing of events like a code tree (tree diagram). The horizontal axis in the trellis diagram represents discrete time steps and the vertical axis in the trellis diagram represents all possible states (table 6) of the encoder, the horizontal and vertical axis depicts the figure 24. The trellis diagram is drawn from a code tree (figure 22) or encoder response table 7, the same as a state diagram. The trellis diagram is more compact and used to trace the output of the encoder. The advantage of the trellis diagram is that it avoids the redundancies of the code tree [36].

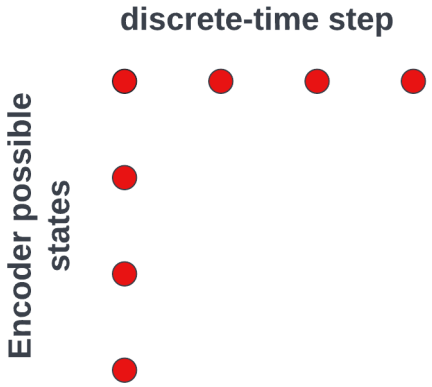

Figure 25: Axial representation for trellis diagram

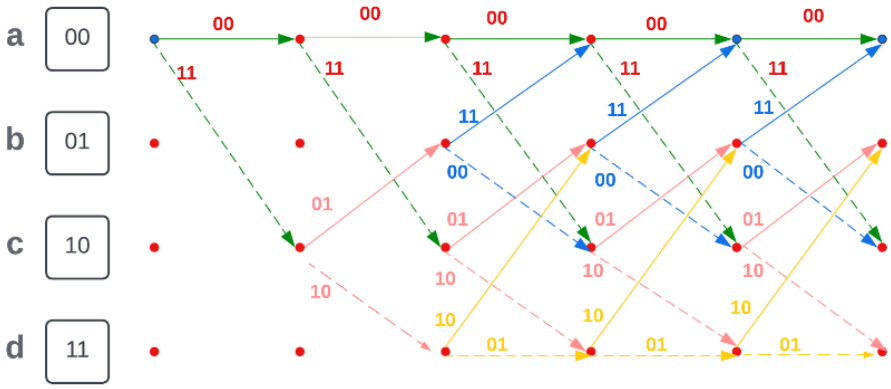

Figure 26: (2,1,2) convolutional encoder of trellis representation

Look at the example in section 4.4.2, by using the trellis representation let's encode the output for the input bits u = 1010

Insert the first input bit 1 in the trellis diagram from the starting state 00 and the encoded output is 11 which is represented in the dotted lines because the input bit is 1. For the input bit 0, it represents in the dark line and for input 1 it represents in dotted lines. At state 10, insert second input bit 0 is

represented to the 01 state with the encoded output is 01. At the 01 state, insert the third input bit 1 represented in dotted line to the state 10, the encoded output is 00. For the final input bit 0, enter the bit 0 at the state 10 represented in the dark line with the encoded output bit stream is 01. The encode output for the input bits u = 1010 is **11010001**.

For the convolutional encoder representation, we can perform any one of the representations to encode the input bits. All three representations can be used to implement the convolutional code with different constraint lengths of different input bit streams.

The MIoTy physical layer uses the 1/3 convolutional code and as we seen the example construction of (2,1,2) convolutional codes and by referring to that example we will construct the hardware implementation of the (3,1,7) convolutional code and the constraints for the (3,1,7) codes as stated in the ETSI (TS 103 357) standard.

Figure 27 shows the hardware implementation of the (3,1,7) convolutional code with the shift registers and XoR operations. The output for the (3,1,7) convolutional code is generated by performing the XoR operations with the input bits u = 01101101.

Here the constraint length K = 7 and the rated efficiency r = ⅓. The generator polynomials are $g^{(1)}$= 135 , $g^{(2)}$ = 165 , $g^{(3)}$ = 171 [37], and the possible states be 2^{K-1} states i.e., 2^{7-1}= 2^{6}= 64 possible states. The encoded output for the (3,1,7) convolutional code is based on the input bit(s) stream and the generator polynomial i.e., $V^{(1)}= u \oplus g^{(1)}$, $V^{(2)}= u \oplus g^{(2)}$, $V^{(3)}= u \oplus g^{(3)}$. The values of the generator polynomials are $g^{(1)}$= 1011101 , $g^{(2)}$= 1110101 , $g^{(3)}$= 1111001. Hence the reference from the section 4.3.1 the generated polynomial for the given values is represented as

$$g^{(1)} = 1 + x^2 + x^3 + x^4 + x^6$$

$$g^{(2)} = 1 + x + x^2 + x^4 + x^6$$

$$g^{(3)} = 1 + x + x^2 + x^3 + x^6$$

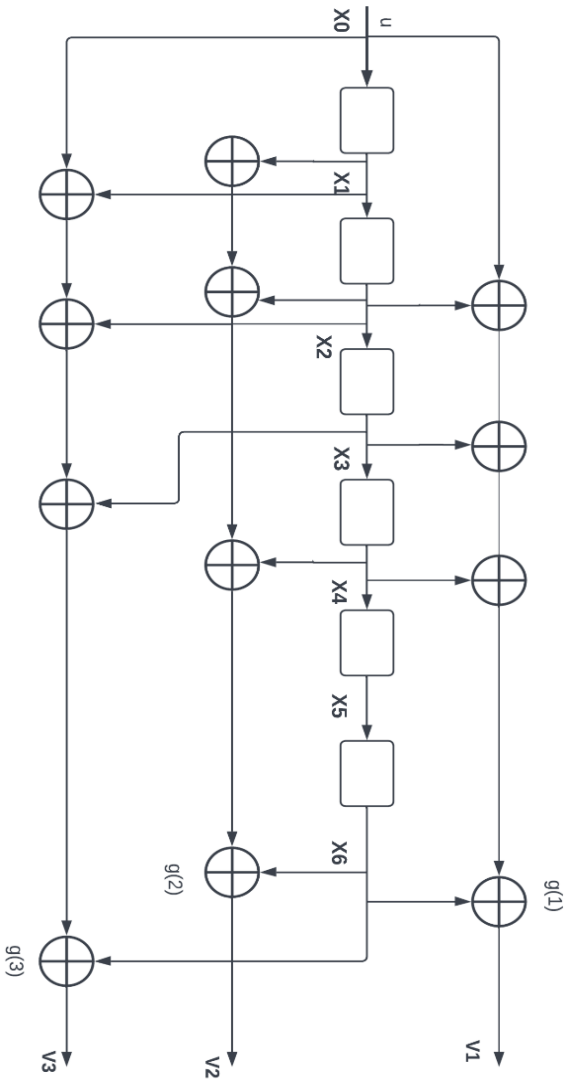

Figure 27: Hardware implementation of (3,1,7) Convolutional encoder

Input bit **u = 01101101**

$$\boldsymbol{V^{(1)} = u \oplus g^{(1)}},$$

$$\boldsymbol{V^{(2)} = u \oplus g^{(2)}},$$

$$\boldsymbol{V^{(3)} = u \oplus g^{(3)}}$$

The encoded output for the (3,1,7) convolutional encoder by applying the XoR operation and the shift registers. Tables 8,9,10 shows the encoded output process. To generate the output, first we will insert the input bits into the shift register and corresponding generated polynomials performs XoR operation. After, performing the operations the encoded output as follows

$$\boldsymbol{V^{(1)} = 10\ 01\ 10\ 10\ 01\ 00\ 00}$$

$$\boldsymbol{V^{(2)} = 11\ 11\ 10\ 11\ 00\ 10\ 00}$$

$$\boldsymbol{V^{(3)} = 11\ 10\ 10\ 10\ 00\ 11\ 00}$$

$g^{(1)}$	**u**							
	0	**1**	**1**	**0**	**1**	**1**	**0**	**1**
1	1	0	0	1	0	0	1	0
0	0	1	1	0	1	1	0	1
1	1	0	0	1	0	0	1	0
1	1	0	0	1	0	0	1	0
1	1	0	0	1	0	0	1	0
0	0	1	1	0	1	1	0	1
1	1	0	0	1	0	0	1	0

Table 8: Generation of encoded output bits of $V^{(1)}$

$g^{(2)}$	u							
	0	**1**	**1**	**0**	**1**	**1**	**0**	**1**
1	1	0	0	1	0	0	1	0
1	1	0	0	1	0	0	1	0
1	1	0	0	1	0	0	1	0
0	0	1	1	0	1	1	0	1
1	1	0	0	1	0	0	1	0
0	0	1	1	0	1	1	0	1
1	1	0	0	1	0	0	1	0

Table 9: Generation of encoded output bits of $V^{(2)}$

$g^{(3)}$	u							
	0	**1**	**1**	**0**	**1**	**1**	**0**	**1**
1	1	0	0	1	0	0	1	0
1	1	0	0	1	0	0	1	0
1	1	0	0	1	0	0	1	0
1	1	0	0	1	0	0	1	0
0	0	1	1	0	1	1	0	1
0	0	1	1	0	1	1	0	1
1	1	0	0	1	0	0	1	0

Table 10: Generation of encoded output bits of $V^{(3)}$

Chapter 5

Modulation

After the completion of correcting the errors in the data bits of the physical waveform depicts the figure 28, the FEC block forward the data bits to the fourth block of the modulation (MSK or GMSK). In this block, the modulation effects change the property of sound over time waveform using GMSK I and Q phase representation. In MIoTy physical layer, we use the MSK or GMSK modulation with specific bandwidth-time product. In this thesis, we are focusing only GMSK modulation, because the Gaussian pulse shaping smooths the pulse trajectory of the MSK signals and hence stabilizes the instantaneous frequency variations over time. We will study the overview of GMSK modulation with bandwidth-time product and ideally this thesis is to describe a mathematical description of the physical layer, we will discuss in section 5.2.

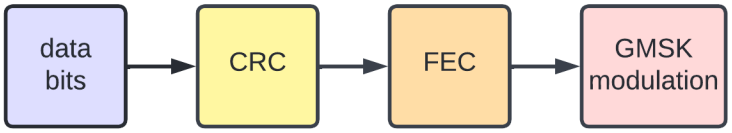

Figure 28: GMSK generation block of the waveform

5.1 GMSK Modulation

The proliferation of computers today has increased the demand for transmission of data over wireless links. Binary data, composed of sharp "one to zero" and "zero to one" transitions, results in a spectrum rich in harmonic content that is not well suited to radio frequency (RF) transmission. Hence, the field of digital modulation has been flourishing. Recent standard such as ETSI (TS 103 357) specify Minimum Shift Keying (MSK), or Gaussian filtered Minimum Shift Keying (GMSK) for their modulation method in the MIoTy physical layer.

The pre-modulation filtering is an effective method for reducing the occupied spectrum for wireless data transmission. In addition to a compact spectrum, a wireless data modulation scheme must have good bit error rate (BER) performance under noisy conditions. Its performance should also be independent of power amplifier linearity to allow the use of class C power amplifiers. GMSK is a simple yet effective approach to digital modulation for wireless data transmission . To provide a good understanding of GMSK, we will review the basics of GMSK.

Gaussian minimum-shift keying (GMSK) is a form of continuous-phase Frequency shift keying (CPFSK) and derived from Offset Quadrature phase-shift keying (OQPSK) modulation**.** That mean the phase is change between the symbols to provide a constant envelope. Gaussian Minimum Shift Keying (GMSK) is a modified MSK modulation technique [38], where the spectrum of MSK is manipulated by passing the rectangular-shaped information pulses through a Gaussian low-pass filter (LPF) prior to the frequency modulation of the carrier. A typical Gaussian LPF, used in GMSK modulation standards, is defined by the zero-mean Gaussian (bell-shaped) impulse response. In this way, the basic MSK signal is converted to GMSK modulation [39].

Bandwidth is a frequency range over which a signal spans itself such a type of bandwidth is measured in a unit known as Hertz. It can also be defined as the data transmitted per unit of time which is measured in bits per second or megabits per second one important thing to be noted about bandwidth is that it increases the speed of the data connections. Bandwidth is not just directly related to the speed; it is also inversely related to the range. Hence as the bandwidth increases the range decreases.

The BT product is the bandwidth-symbol time product. where B is the 3-dB (half-power) bandwidth of the pulse/filter and T is the symbol duration. For different applications we will have a different bandwidth values. For example, in GSM telephony for instance, a BT=0.3 is recommended and in satellite communications with GMSK, for near-earth missions a BT=0.25 is recommended. For MIoTy physical layer applications, the MSK or GMSK modulation with

BT=1.0 (see in figure) is recommended in the uplink and downlink transmission, for more about the uplink and downlink transmission refer to [ETSI TS 103 357].

For a BT=1, the pulse shaping the symbol spreads over one-bit period duration. For BT=0.25, the spread is over 4-bit periods, for BT=0.3 the spread is over approximately 3-bit periods, and for BT=0.5 the spread is over 2-bit periods. Figure 29 shows the pulse shapes for different values of the bandwidth-symbol time product (BT). This means that a smaller BT product results into higher intersymbol interference (ISI) and a compact spectrum [40]. Measures need to be taken for the introduced ISI in this case much more than in the case of a higher BT product where less ISI is introduced, and we have much less compact spectrum.

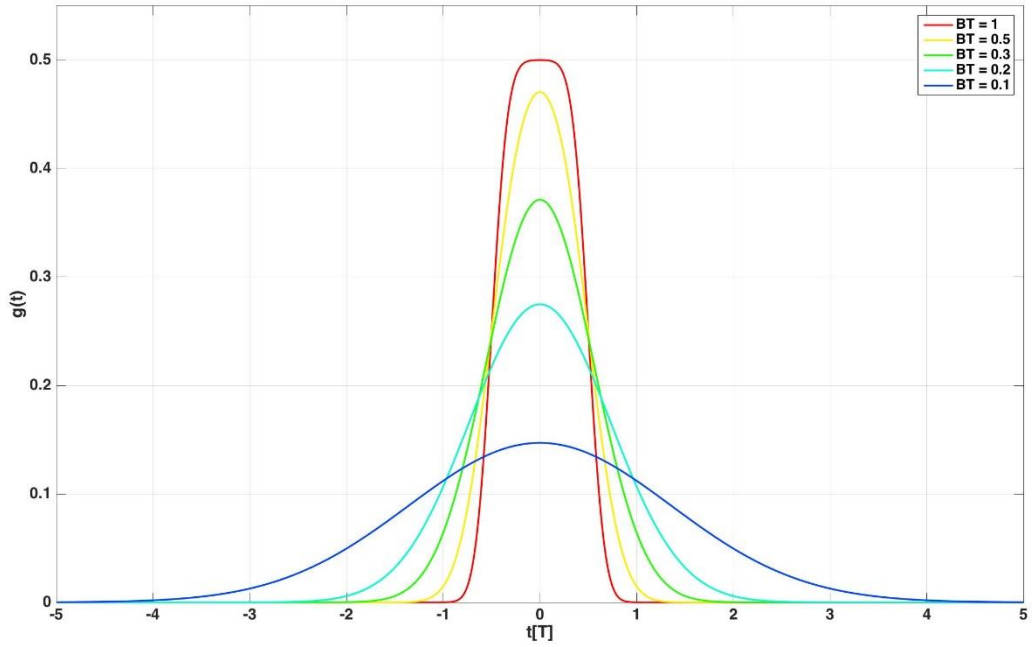

Figure 29: Pulse shape for different BT product [Web 11]

In GMSK, one of its properties is, it maintains a constant envelope and that's because of the Gaussian pulse applied prior to modulation. The GMSK frequency pulse is the difference of two

time-displaced (by T_b seconds) Gaussian probability integrals (Q-functions) i.e., we assume here a frequency pulse shape, **g(t)**, that results from excitation of the Gaussian filter (arbitrarily assumed to have zero group delay) with the unit rectangular pulse **p(t) = 1, 0 ≤ t ≤ T_b** . The GMSK pulse can be defined as in equation (4) below

$$g(t) = \frac{1}{2T_b}\left[Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{t}{T_b} - 1\right)\right) - Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\frac{t}{T_b}\right)\right],$$

$$Q(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{y^2}{2}\right) dy, \quad -\infty \le t \le \infty \tag{4}$$

where B is the 3-dB bandwidth of the lowpass Gaussian filter [41] and is related to the noise bandwidth, B_N is,

$$\frac{B}{B_N} = 2\sqrt{\frac{\ln 2}{\pi}} = 0.93944 \tag{5}$$

Thus, for a given application, the value of , BT_b is selected as a compromise between spectral efficiency and bit-error probability (BEP) (6) performance.

$$P_b(E) = \frac{1}{2}\operatorname{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right), \quad E_b = PT_b \tag{6}$$

$$g(t) = \begin{cases} \frac{1}{2T_b}\left[Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{t}{T_b}-1\right)\right) - Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\frac{t}{T_b}\right)\right], & -(L-1)\,T_b/2 \le t \\ & \le (L+1)\,T_b/2 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where L is chosen as above in accordance with the value of BT_b. Technically speaking, g (t) of (7) should be scaled by a constant C to satisfy a condition namely,

$$q(t) = \int_{-\infty}^{t} g(\tau)d\tau = \begin{cases} 0, & t \le -(L-1)\,T_b/2 \\ 1/2, & t \ge (L+1)\,T_b/2 \end{cases} \tag{8}$$

Also, although g (t) of (8) appears to have a "Gaussian-looking" shape, we emphasize that the word Gaussian in GMSK refers to the impulse response of the filter through which the input rectangular pulse train is passed and not the shape of the resulting frequency pulse. We can say that the BT somehow determines the degree of filtering. For more information on BT [42] discussions in relation to equation (7)

5.2 Mathematical description of Physical layer (GMSK modulation)

As we above mentioned in section 5.1, the GMSK modulation is an extension of MSK modulation by applying the Gaussian Filters to the basic Filter of MSK modulation. A mathematical description of GMSK modulation is described from the MSK modulation. In this thesis, we below provide the GMSK mathematical description from the reference [Web 12],[47][43]

$$s_{GMSK}(t) = \sqrt{\frac{2E_b}{T_b}}\cos\left(2\pi f_c t + \frac{\pi}{2T_b}\sum_i \alpha_i \int\left[Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{T}{T_b} - (i+1)\right)\right) - Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{T}{T_b} - i\right)\right)\right]dT\right),$$

$$nT_b \le t \le (n+1)T_b$$

(9)

In figure 30, if the input is represented by its equivalent NRZ data stream (i.e., the frequency pulse stream that would ordinarily be inputted to the (frequency modulation) FM modulator in MSK), then the filter impulse response, h (t), becomes Gaussian, as implied by the GMSK acronym, i.e.,

$$h(t) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{t^2}{2\sigma^2}\right), \quad \sigma^2 = \frac{\ln 2}{(2\pi B)^2} \qquad (2.8\text{-}56)$$

(10)

the implementation of the gaussian filter h(t) in figure 30 is represented as in figure 31

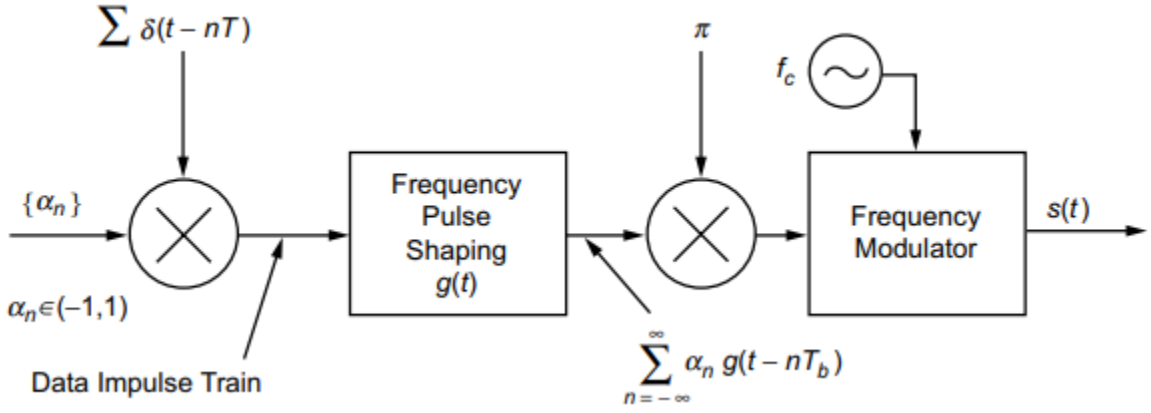

Figure 30: GMSK transmitter (CPM representation)[47]

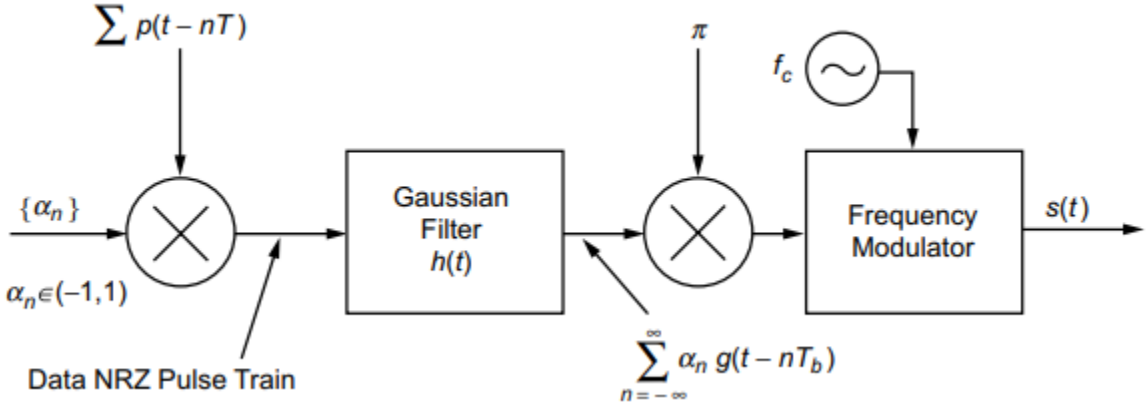

Figure 31: Equivalent GMSK transmitter(CPM transmitter)[47]

From expression (7) and (10), we compare a different BT_b values as shown in figure 32,

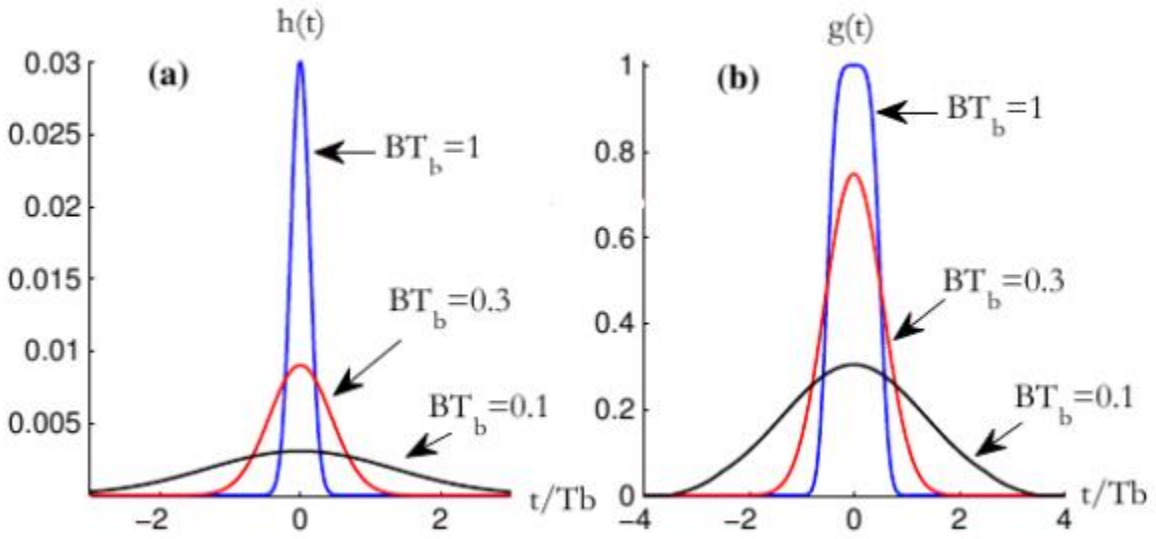

Figure 32: Gaussian LPF: (a) impulse response and

(b) response to isolated rectangular pulse[Web 13]

The frequency modulator in Fig 30 or Fig 31, is typically implemented with a phase-locked loop synthesizer whose voltage-controlled oscillator (VCO) input is the point at which the modulation is injected. When long strings of zeros or ones are present in the data, the spectrum of the modulation extends to direct-circuit (dc), which presents a problem, since phase-locked loop frequency [Web 14] synthesizers implemented as above do not respond to this low-frequency signal due to their inherent high pass filter characteristic. As such, the VCO output (the location of the modulated signal) would not contain the low-frequency content of the information (modulating) signal. By contrast, if the modulation were to be injected at the input of the master oscillator preceding the phase-lock loop (the oscillator must be capable of being modulated by a voltage signal), then since this oscillator is not in the loop, the VCO output would contain the low-frequency content of the modulation (i.e., that within the loop filter bandwidth) but not its high-frequency content. Clearly then, a combination of these two approaches would yield the desirable result of constant modulation sensitivity, irrespective of the loop bandwidth. Such an FM scheme is referred to as two-point modulation and corresponds to a dc-coupled GMSK modulator wherein the Gaussian filtered input signal is split sending one portion to the VCO modulation input and the other to the phase-lock loop master oscillator input.

Equivalent I-Q Representations:

For high carrier frequencies, direct synthesis of the GMSK signal, using a digital approach is impractical since maintaining an adequate sampling rate requires an extremely high operating frequency. Instead, one can resort to a quadrature implementation where lowpass I and Q signals containing the phase information are generated that vary much slower than the phase of the modulated carrier [44], thus making it feasible to implement them digitally. Applying the simple trigonometric rule for the cosine of the sum of two angles to (9) we obtain

$$s_{GMSK}(t) = \sqrt{\frac{2E_b}{T_b}}\left[cos\phi(t,\alpha)\, cos2\pi f_c t - sin\phi(t,\alpha)\, sin2\pi f_c t\right] \tag{11}$$

Where,

$$\phi(t,\alpha) =$$

$$\frac{\pi}{2T_b}\sum_i \alpha_i \int \left\{ Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{\tau}{T_b} - (i+1)\right)\right) - Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{\tau}{T_b} - i\right)\right)\right\} d\tau \tag{12}$$

Conceptually then, an I-Q transmitter for GMSK is one that performs the following sequence of steps: first, the Gaussian-filtered NRZ data stream is generated. Next, integration is performed to produce the instantaneous phase of equation (11). Finally, the integrator output is passed through sine and cosine read-only memories (ROMs) whose outputs are applied to I and Q carriers (see Figure 32). Such a scheme has also been referred to as quadrature cross-correlated GMSK [45] for an illustration like Figure 32. In these implementations, the block labeled "Gaussian filter" is either an actual filter that approximates the Gaussian impulse response as per (7) or, more efficiently, a ROM table lookup, whereas the block labeled "integrator" is typically performed by a phase accumulator (i.e., Without loss in generality, the Gaussian filter and integrator blocks can be switched as is the case in some of the implementations).

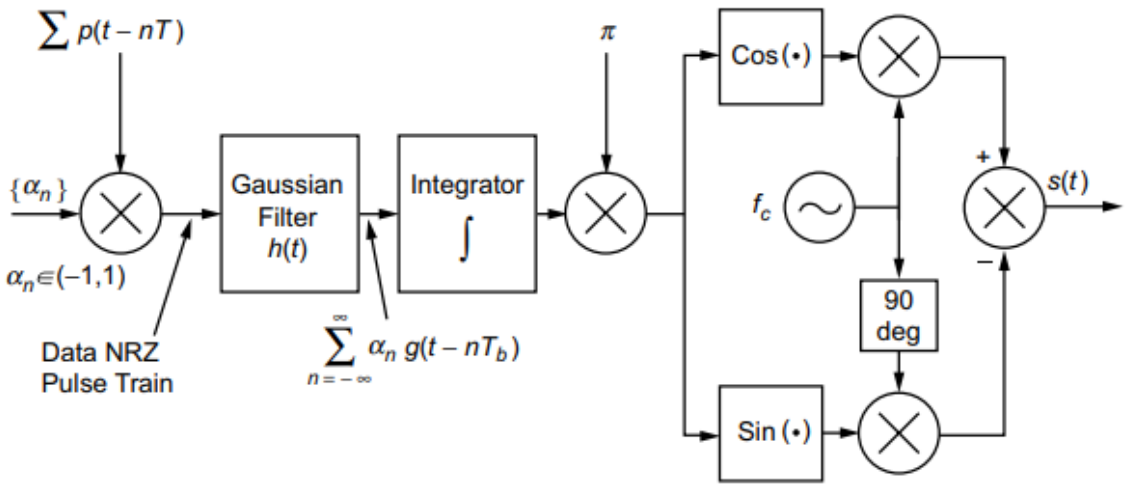

Figure 33: GMSK transmitter (I-Q representation)

An efficient I-Q implementation of a GMSK modulator is presented that skips the above sequence of steps and instead generates the I and Q baseband signals directly from the binary data, thereby eliminating the errors in filtering, phase truncation, and sine/cosine computation inherent in the conventional architecture.

Consider the GMSK frequency response (pulse train) that generates the phase of (12). If we impose the condition that this response in the mth bit interval, mTb ≤ |t| ≤ (m + 1) Tb, be dependent only on the bit of interest, α_m, and its two nearest neighbors, α_{m-1} and α_{m+1}.

$$\left.\begin{aligned} Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\right) &\cong 0 \\ Q\left(-\frac{2\pi BT_b}{\sqrt{\ln 2}}\right) &\cong 1 \end{aligned}\right\} \tag{13}$$

Assuming (13) is true, then since by superposition the response to a train of NRZ pulses varying from −1 to 1 is the equivalent to the response to a rectangular pulse train varying from 0 to 2 minus a constant of value 1, the normalized frequency response in the above interval can be expressed as

$$\begin{aligned} g_m(t) \triangleq & \sum_{i=m-1,m,m+1} (\alpha_i + 1)\left[Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{t}{T_b} - (i+1)\right)\right)\right. \\ & \left. - Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{t}{T_b} - i\right)\right)\right] dt - 1 \\ \cong & (\alpha_{m-1} + 1) Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{t}{T_b} - m\right)\right) \\ & + (\alpha_m + 1)\left[Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{t}{T_b} - (m+1)\right)\right) - Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{t}{T_b} - m\right)\right)\right] \\ & + (\alpha_{m+1} + 1)\left[1 - Q\left(\frac{2\pi BT_b}{\sqrt{\ln 2}}\left(\frac{t}{T_b} - (m+1)\right)\right)\right] - 1 \end{aligned} \tag{14}$$

Alternatively, since the Gaussian Q-function can be expressed in terms of the error function by $Q(x) = (1/2)\left[1 + erf\left(x/\sqrt{2}\right)\right]$, then letting $\alpha_i' = (1/2)(\alpha_i + 1)$ denote the (0,1) equivalent of

the (−1, 1) αi's, and introducing the constant $\beta \triangleq \pi B\sqrt{2/ln\ 2}$, as in Eq. (19) of [46], (14) can be rewritten as

$$\begin{aligned} g_m(t) \cong\ & \alpha'_{m-1}\left[1 - \operatorname{erf}\left(\beta\left(t - mT_b\right)\right)\right] \\ & + \alpha'_m\left[\operatorname{erf}\left(\beta\left(t - mT_b\right)\right) - \operatorname{erf}\left(\beta\left(t - (m+1)T_b\right)\right)\right] \\ & + \alpha'_{m+1}\left[1 + \operatorname{erf}\left(\beta\left(t - (m+1)T_b\right)\right)\right] - 1 \end{aligned} \tag{15}$$

Corresponding to the values (0,1) for each of the three $\alpha'_i s$ in (15), there are eight possible waveforms $f_i(t - mT_b)$, i = 0, 1, 2, ⋯ , 7 that characterize the frequency response in the mth bit interval. These are given in Table 11 assuming m = 0 for simplicity.

$\alpha'_{-1}, \alpha'_0, \alpha'_1$	i	$f_i(t)$
000	0	-1
001	1	$\operatorname{erf}\left(\beta(t - T_b)\right)$
010	2	$\operatorname{erf}(\beta t) - \operatorname{erf}\left(\beta(t - T_b)\right) - 1$
011	3	$\operatorname{erf}(\beta t)$
100	4	$-\operatorname{erf}(\beta t)$
101	5	$-\operatorname{erf}(\beta t) + \operatorname{erf}\left(\beta(t - T_b)\right) + 1$
110	6	$-\operatorname{erf}\left(\beta(t - T_b)\right)$
111	7	1

Table 11: Possible frequency responses in the interval $0 \le t \le T_b$

We observe from the table 11 that there are only three independent frequency response waveforms, i.e., $f_2(t), f_3(t), f_7(t)$ in that the remaining five can be obtained from these three by means of simple operations, namely,

$$\begin{aligned} f_0(t) &= -f_7(t) \\ f_1(t) &= f_3(t) - f_2(t) - f_7(t) = f_3(t - T_b) \\ f_4(t) &= -f_3(t) \\ f_5(t) &= -f_2(t) \\ f_6(t) &= -f_1(t) \end{aligned} \tag{16}$$

In view of the above, the frequency modulating signal corresponding to the phase modulating signal of (12) can be expressed in the form of a data-dependent pulse train as

$$f(t, \alpha) = \frac{1}{2\pi}\frac{d}{dt}\phi(t, \alpha) = \frac{1}{4T_b}\sum_i f_{l(i)}(t - iT_b)\, p(t - iT_b) \tag{17}$$

whereas before, $p(t)$ is a unit amplitude rectangular pulse in the interval $0 \leq t \leq T_b$ and the index $l(i) = 4a_{i-1} + 2a_i + a_{i+1}$ is the decimal equivalent of the 3-bit binary sequence influencing the ith bit interval and determines the frequency waveform for that interval in accordance with Table 11. The corresponding complex phase modulating signal can be written in the form

$$\exp\{\phi(t,\alpha)\} = \exp\left\{\sum_i \phi_{l(i)}\,(t-iT_b)\,p\,(t-iT_b)\right\},$$

$$\phi_i(t) = \frac{\pi}{2T_b}\int_0^t f_i\,(\tau)\,d\tau + \phi_i(0) \tag{18}$$

where $\phi_i(0)$ is the initial phase value that depends on the previous history of the data sequence. Analogous to Table 11, there are eight possible phase responses in any given bit interval. These are evaluated in [46], using the approximation of (13) (reformulated in terms of the error function as $(\beta T_b) \cong 1, erf\ (-\beta T_b) \cong -1$, along with appropriate asymptotic expansions of the error function. Once again, there are only three independent phase response waveforms, e.g., $\phi_2(t), \phi_3(t), \phi_7(t)$, and the remaining five can be obtained from these three by means of simple operations.

Conclusion

The main objective of this thesis is to provide a mathematical description of the physical layer using GMSK modulation in MIoTy physical layer transmitter waveform using the TSMA pattern. In this thesis, we focused only on some blocks of the physical layer transmitter and studied, how the information bits perform step by step by applying different techniques (schemes). In the first block of the MIoTy physical layer transmitter, we represent the information data bits, later in the second block is to identify the errors in the data bits. In this block we studied, how it determines the errors in the data bits by using different CRC code bits. After completion of identifying an errors in data bits, that block sends the data bits to the third block which is the FEC block. In this FEC block, it performs to correct the errors in data bits, which is identified in the previous block. We studied how it is correcting the errors by using the 1/3 convolutional trellis codes and constructed the (3,1,7) convolutional code. After completion of correcting errors, it forwards the data bits to the next block which is the modulational block. In this block, we studied the GMSK modulation with a bandwidth-time product and we described a mathematical description of the MIoTy physical layer in detail. By using GMSK modulation in this block, we can maintain the bandwidth symbol time product BT=1 with modulation index h=1.

Finally, after performing all the blocks (step by step), the GMSK block forwards the information bits to radio channels. In this last block, the telegram-splitting multiple access (TSMA) pattern is applied, and it split the data bits into small sub-packets, these sub-packets of data has sent through the telegram-splitting ultra-narrow band (TS-UNB) waveform. This waveform is then transmitted from the transmitter to the receiver. The further research work of this thesis is to perform the remaining blocks of the physical layer transmitter and a depth understanding of the MIoTy physical layer.

Bibliography

1. *Monjur, Mohammad Mezanur, "Internet-of-Things (IoT) Security Threats: Attacks on Communication Interface" (2020). Master's Theses and Capstones. 1388. https://scholars.unh.edu/thesis/1388*
2. *LPWA network technologies and low-power standards. (n.d.). Retrieved June 21, 2021, from i-SCOOP: https://www.i-scoop.eu/internet-of-things-iot/lpwan/*
3. *Nitin Naik. (n.d.). LPWAN Technologies for IoT Systems: Ultra Narrow Band and Spread Spectrum. LPWAN Technologies for IoT Systems: Choice Between Ultra Narrow Band and Spread Spectrum, 8. Retrieved from LPWAN_IoT_UNB_SS_Naik.pdf*
4. *RealWireless. (2015) A comparison of UNB and spread spectrum wireless technologies as used in LPWA M2M applications. [Online]. Available: https://www.thethingsnetwork.org/forum/uploads/default/original/1X/3b1c1ae4a925e9aa897110ccde10ec61f3106b87.pdf*
5. *Comtechefdata.com. (2012) Spread Spectrum in the SLM5650A: Features, performance, and applications. [Online]. Available: http://www.comtechefdata.com/files/articles papers/WPSpread-Spectrum-in-SLM-5650A.pdf*
6. *G. Kilian et al., "Increasing Transmission Reliability for Telemetry Systems Using Telegram Splitting," in IEEE Transactions on Communications, vol. 63, no. 3, pp. 949-961, March 2015, doi: 10.1109/TCOMM.2014.2386859.*
7. *Milarokostas, Christos; Tsolkas, Dimitris; Passas, Nikos; Merakos, Lazaros (2021): A Comprehensive Study on LPWANs With a Focus on the Potential of LoRa/LoRaWAN Systems. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.16853893.v1*
8. *Ertürk, M.A., Aydin, M.A., Buyukakkaslar, M.T., & Evirgen, H. (2019). A Survey on LoRaWAN Architecture, Protocol and Technologies. Future Internet, 11, 216.*
9. *Mekki, Kais & Bajic, Eddy & Chaxel, Frédéric & Meyer, Fernand. (2019). A comparative study of LPWAN technologies for large-scale IoT deployment. 5. 1-7. 10.1016/j.icte.2017.12.005.*
10. *Aguilar, Sergio & Wistuba, Diego & Platis, Antonis & Vidal Ferré, Rafael & Gomez, Carles & Céspedes, Sandra & Zuniga, Juan-Carlos. (2021). Packet Fragmentation over Sigfox: Implementation and Performance Evaluation of SCHC ACK-on-Error. IEEE Internet of Things Journal. 10.1109/JIOT.2021.3127170.*

11. *Kisseleff, Steven & Kneissl, Jakob & Kilian, Gerd & Gerstacker, Wolfgang. (2020). Efficient Detectors for Telegram Splitting based Transmission in Low Power Wide Area Networks with Bursty Interference.*
12. *J. S. E, A. Sikora, M. Schappacher, and Z. Amjad, "Test and Measurement of LPWAN and Cellular IoT Networks in a Unified Testbed," 2019 IEEE 17th International Conference on Industrial Informatics (INDIN), 2019, pp. 1521-1527, doi: 10.1109/INDIN41052.2019.8972256.*
13. *Lawrence Williams. (2022, April 23). Error Detection and Correction with Examples. Retrieved from https://www.guru99.com/hamming-code-error-correctionexample.html/*
14. *R. W. Hamming, "Error detecting and error correcting codes," in The Bell System Technical Journal, vol. 29, no. 2, pp. 147-160, April 1950, doi: 10.1002/j.1538-7305.1950.tb00463.x.*
15. *M. S. Santos and R. d'Amore, "Error detection method for the ARINC429 communication," 2018 IEEE 19th Latin-American Test Symposium (LATS), 2018, pp. 1-6, doi: 10.1109/LATW.2018.8349687.*
16. *F. Zulfira, H. H. Nuha, D. Wisaksono Sudiharto and R. G. Utomo, "Modified Bit Parity Technique for Error Detection of 8 Bit Data," 2021 9th International Conference on Information and Communication Technology (ICoICT), 2021, pp. 517-521, doi: 10.1109/ICoICT52021.2021.9527522.*
17. *Singh, Varinder and Sharma, Narinder. (2018). A Review on Various Error Detection and Correction Methods Used in Communication.*
18. *Aphasana Mulla, Shital Deshmukh, & Wrushali Deshmukh. (2021, May). A Review on Generations of Various Error Detection Codes Using C Programming in Computer Network. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), Volume 5(Issue 1), 2-6. Retrieved from https://ijarsct.co.in/Paper1139.pdf*
19. *P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," International Conference on Dependable Systems and Networks, 2004, 2004, pp. 145-154, doi: 10.1109/DSN.2004.1311885.*
20. *M. Ghosh and F. LaSita, "Puncturing of CRC Codes for IEEE 802.11ah," 2013 IEEE 78th Vehicular Technology Conference (VTC Fall), 2013, pp. 1-5, doi: 10.1109/VTCFall.2013.6692382.*
21. *Michael Vega. (2005, October). Calculating CRC With TI Battery Management Products. Texas Instruments(Application Report SLUA363), 2-5. Retrieved from https://www.ti.com/lit/an/slua363/slua363.pdf*
22. *David Culler. (n.d.). Components and Design Techniques for Digital Systems. Retrieved from https://profile.iiita.ac.in/bibhas.ghoshal/lecture_slides_coa/LFSR_CRC.pdf*

23. *Texas Instruments. (2020, August). Implementation of CRC for ADS7066. 1-4. Retrieved from https://www.ti.com/lit/an/sbaa456/sbaa456.pdf*
24. *M. Sundelin, W. Granzow and H. Olofsson, "A test system for evaluation of the WCDMA technology," VTC '98. 48th IEEE Vehicular Technology Conference. Pathway to Global Wireless Revolution (Cat. No.98CH36151), 1998, pp. 983-987 vol.2, doi: 10.1109/VETEC.1998.686387.*
25. *Henriksson, Tomas, and Liu, Dake. (2003). Implementation of fast CRC calculation. Proceedings of the Asia and South Pacific, Design Automatation Conference. 563- 564. 10.1109/ASPDAC.2003.1195079.*
26. *Martin P Clark. (2000). Forward Error Correction Codes. Retrieved from https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470841516.app6*
27. *Ting Chen, "Analysis of forward error correcting codes," 2011 International Conference on System science, Engineering design and Manufacturing informatization, 2011, pp. 329-332, doi: 10.1109/ICSSEM.2011.6081220.*
28. *A. C. Vaz, C. G. Nayak, D. Nayak and N. T. Hegde, "Performance analysis of Forward Error Correcting Codes in a Visible Light Communication System," 2021 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2021, pp. 1-5, doi: 10.1109/CONECCT52877.2021.9622633.*
29. *Zhiyong Chen. (2017, October 16). Computing and Communications 2. Channel Coding and Modulation. Institute of Wireless Communications Technology Shanghai Jiao Tong University, 47-54. Retrieved from https://iwct.sjtu.edu.cn/Personal/zychen/ChannelCodingAndModulation.pdf*
30. *Rob Maunder. (n.d.). ELEC1011 Communications and Control (8/12) Channel Coding. Retrieved from EdShare: http://edshare.soton.ac.uk/5292/1/ChannelCoding.pdf*
31. *W. Rurik and A. Mazumdar, "Hamming codes as error-reducing codes," 2016 IEEE Information Theory Workshop (ITW), 2016, pp. 404-408, doi: 10.1109/ITW.2016.7606865.*
32. *hari. (2011, October 2). Convolutional Codes Encoding. Retrieved from CHAPTER 7 - Convolutional Codes: Construction and Encoding: http://web.mit.edu/6.02/www/f2011/handouts/7.pdf*
33. *R. J. McEliece and Wei Lin, "The trellis complexity of convolutional codes," in IEEE Transactions on Information Theory, vol. 42, no. 6, pp. 1855-1864, Nov. 1996, doi: 10.1109/18.556680.*
34. *J. Park and J. Moon, "CRC Codes Based on a Non-Primitive Generator Polynomial: A New Error Control Scheme Targeting a Prescribed Set of Error Patterns," 2006 IEEE*

International Magnetics Conference (INTERMAG), 2006, pp. 301-301, doi: 10.1109/INTMAG.2006.375883.

35. *J.-M. Brossier. (n.d.). Coding and decoding with convolutional codes. The Viterbi Algorithm. Retrieved from GIPSA-Lab: http://www.gipsa-lab.grenoble-inp.fr/~jeanmarc.brossier/printable-slides-viterbi.pdf*
36. *Jing Li and E. Kurtas, "Punctured convolutional codes revisited: the exact state diagram and its implications," Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004., 2004, pp. 2015-2019 Vol.2, doi: 10.1109/ACSSC.2004.1399518.*
37. *Lin, S., and Costello, D. J. (1983). Error Control Coding: Fundamentals and Applications (Vols. Pg. 287-313). Prentice-Hall. Retrieved from https://pg024ec.files.wordpress.com/2013/09/error-control-coding-by-shu-lin.pdf*
38. *M. A. Mobin, M. Islam, and M. S. Rahman, "Performance Analysis of GMSK and Chirp-Binary Orthogonal Keying in AWGN and Multipath Fading Channels," 2020 11th International Conference on Electrical and Computer Engineering (ICECE), 2020, pp. 443-446, doi: 10.1109/ICECE51571.2020.9393133.*
39. *K. Tsai, T. Oyang, J. Phan, Cheng Chih Yang and N. Schneier, "Gaussian minimum shift keying modulator [for satellite communication]," 2000 IEEE Aerospace Conference. Proceedings (Cat. No.00TH8484), 2000, pp. 235-241 vol.5, doi: 10.1109/AERO.2000.878495.*
40. *L. C. Galvão, C. Müller, M. C. F. De Castro, F. C. C. De Castro, and K. S. Mayer, "Bandwidth Efficient Gaussian Minimum Frequency-Shift Keying Approach for Software Defined Radio," 2018 31st Symposium on Integrated Circuits and Systems Design (SBCCI), 2018, pp. 1-4, doi: 10.1109/SBCCI.2018.8533245.*
41. *P. Varshney, J. E. Salt and S. Kumar, "BER analysis of GMSK with differential detection in a land mobile channel," in IEEE Transactions on Vehicular Technology, vol. 42, no. 4, pp. 683-689, Nov. 1993, doi: 10.1109/25.260743.*
42. *Jie Wu and G. J. Saulnier, "A two-stage MSK-type detector for low-BT GMSK signals," in IEEE Transactions on Vehicular Technology, vol. 52, no. 4, pp. 1166-1173, July 2003, doi: 10.1109/TVT.2003.814233.*
43. *P. R. Tapkir, S. B. Singh and N. N. Thune, "Analysis and implementation of minimum shift keying (MSK) modulation on FPGA platform," 2016 International Conference on*

Automatic Control and Dynamic Optimization Techniques (ICACDOT), 2016, pp. 349-354, doi: 10.1109/ICACDOT.2016.7877607.

44. *A. E. Jones and J. G. Gardiner, "Generation of GMSK using direct digital synthesis," IEE Colloquium on Implementations of Novel Hardware for Radio Systems, 1992, pp. 4/1-4/9.*
45. *K. M. Nitin Babu and K. K. Vinaymurthi, "GMSK modulator for GSM system, an economical implementation on FPGA," 2011 International Conference on Communications and Signal Processing, 2011, pp. 208-212, doi: 10.1109/ICCSP.2011.5739302.*
46. *A. Linz and A. Hendrickson, "Efficient implementation of an I-Q GMSK modulator," in IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 43, no. 1, pp. 14-23, Jan. 1996, doi: 10.1109/82.481470.*
47. *Simon, M.K. (2003). Bandwidth-Efficient Digital Modulation with Application to Deep Space Communications*.

Web Sources

1. https://behrtech.com/blog/lpwan-technologies-comparison-2/
2. https://www.microcontrollertips.com/how-does-mioty-compare-with-other-flavors-of-lpwan/
3. https://www.st.com/resource/en/application_note/an5472-lpwan-cellular-connectivity-on-stevalstwinkt1b-for-cloud-condition-monitoringbased-applications-stmicroelectronics.pdf
4. https://www.digikey.it/it/blog/mioty-emerges-as-an-option
5. https://www.kdkce.edu.in/pdf/VVC-8ETRX-SATCOM-U5-Error_Detection_correction.pdf
6. https://cppsecrets.com/users/14781151171091051164699104111117981014955641181051164610100117/Cyclic-Redundancy-check-and-Checksum.php
7. https://humphryscomputing.com/Notes/Networks/data.polynomial.html
8. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9104669
9. https://www.sciencedirect.com/topics/engineering/hamming-code
10. https://www.eee.hku.hk/~sdma/elec7073/Part2-3-Convolutional%20codes.pdf
11. https://dsp.stackexchange.com/questions/30653/what-is-a-bt-bandwidth-time-product-with-reference-to-modulation
12. https://www.inatel.br/docentes/dayan/easyfolder/Publications/40.pdf
13. https://www.gaussianwaves.com/2019/10/gaussian-minimum-shift-keying-gmsk-implementation-and-simulation-part-1/
14. https://webthesis.biblio.polito.it/14408/1/tesi.pdf
15. https://www.wikiwand.com/en/Error_correction_code