Gesamtpunkte:

10.5/15

DatKom	Übungsblatt 3	Kaan Giray Buzluk	405099
SS 2021		Su Ada Yıldırım	410949
31. Mai 2021		Ozan Ege Şap	411851

Aufgabe 3.1 1/1.5

PCM384 versenden Frequenzen bis 384 kHz, also ist 384000 Hz ist unsere Grenzfrequenz. Nach Abtasttheorem ergibt sich eine Abtastrate von 768000 Hz. Wegen Störfaktoren und Imperfektionen der Hardware wird dies auf 800000 Hz geeinigt. Mit einer 32 Bit-Integer Codierung erhalten wir damit 800000 Hz $\cdot 32$ Bit$=25600000 Bit/s$. Also das macht 25.6MBit/s.

Stereo hat 2 Kanäle, also 2·25,6MBits −0.5

Von der Idee her richtig, passt aber auf solche Werte nicht wahllos zu wählen. In der Klausur einfach direkt mit den Werten rechnen.

Aufgabe 3.2 2/2.5

1/1 (a) Das erste Gerät sendet die Datenfolge 1,1,1. Also kriegt man die Signalfolge: -1,1,-1,1,-1,1,-1,1,-1,1,-1,1

Das zweite Gerät sendet die Datenfolge 1,0,0. Also kriegt man die Signalfolge: -1,1,1,-1,1,-1,-1,1,1,-1,-1,1

0.5/0.5 (b) Dann erhält die Basisstation die Signalfolge: -2, 2, 0, 0, 0, 0, -2, 2, 0, 0, -2, 2

0.5/1 (c) Dann erhält die Basisstation eine falshe additive Signalfolge und bei der Kodierung kriegt es nicht die originale Datenfolge, sondern eine fehlerhafte Information.

Und was ist mit Dämpfung? −0.5

Aufgabe 3.3 2/4

(a) $\underbrace{01111110}_{Flag}\ \underbrace{0111110'10}_{A}\ \underbrace{01000111}_{B}\ \underbrace{110'000111}_{C}\ \underbrace{110'100000}_{D}\ \underbrace{01111110}_{Flag}$

1/1 wobei die $0'$ die eingefügte Bits sind.

(b) $\underbrace{11100000}_{DLE}\ \underbrace{01111110}_{STX}\ \underbrace{01111110}_{A}\ \underbrace{01000111}_{B}\ \underbrace{11000111}_{C}\ \underbrace{11100000}_{DLE}\ \underbrace{11100000}_{D}\ \underbrace{11100000}_{DLE}\ \underbrace{01111110}_{STX}$

1/1

(c) 0/2

(i) Bei Bit-Stuffing tritt einen Overhead von mindestens 16 Bits auf, da wir vor und nach der BitFolge jeweils 1 Byte dranhängen. Hier haben wir noch 3 0-Bits eingefügt, also haben wir hier einen maximalen Overhead von 19 Bits.

(ii) Bei Character-Stuffing tritt einen Overhead von mindestens 32 Bits auf, weil wir vor und nach der Bitfolge jeweils 2 Bytes dranhängen. Bei unserem Beispiel haben wir einen maximalen Overhead von 40 Bits, da wir vor dem Zeichen D noch ein Byte hinzufügt haben.

Hier ging es um den i.A. maximal erreichbaren Overhead, z.B. bei Char-Stuffing wenn nur 0s versandt werden 100%.

Aufgabe 3.4 1.5/1.5

Aus der Beziehung zwischen Bitfehlerrate und Paketfehlerrate erstellen wir die folgende Formel:
$PER = 1 - (1 - BER)^n$
Wobei $(1 - BER)$ die Wahrscheinlichkeit für eine korrekt übertragene Bit darstellt, dann hat man mit $(1 - BER)^n$ die Wahrscheinlichkeit für n Bits. Dann subtrahieren wir diese Zahl von 1 und damit kriegen wir die Wahrscheinlichkeit dafür, dass ein Paket nicht komplett korrekt übertragen wird.

Mit einer Bitfehlerrate von 10^{-5} erhält man für BER 0.00001.
Dann ist für eine Länge von 40 $Byte = 320$ $Bits$:
$PER = 1 - (1 - 0.00001)^{320} = 0.0031949$ ✓

Und für eine Länge von 1500 $Byte = 12000$ $Bits$:
$PER = 1 - (1 - 0.00001)^{12000} = 0.11308$ ✓ Top!

Aufgabe 3.5 1.5/2

(a) Mit diesem Technik kann man nicht so viel machen. Wenn ein Fehler an einer geraden oder ungeraden Position auftritt, kann man dieses 1-Bit mit dem jeweiligen Parity-Bits erkennen aber man kann nicht bestimmen an welcher Position es passiert. Also kann man es nicht korrigieren. ✓ Wenn aber sowohl in geraden als auch ungeraden Positionen ein Fehler auftritt, dann kann man erkennen, dass es 2-Bits fehlerhaft sind. Wenn 2 Fehler an einer ungeraden Stell vorkommen lässt sich dies auch nicht erkennen. -0.5

(b) Die Bitfolge ist 11101000101101011001100010110 1. Wir bilden damit einen Block mit Größe 6-Bit. 111

1	1	1	0	1	0	0
0	0	1	0	1	1	1 ✓
0	1	0	1	1	0	1 Längsparität
0	1	1	0	0	0	0
1	0	1	1	0	1	0
0	1	0	0	1	0	

Querparität ✓

Aufgabe 3.6 2.5/3.5

(a) Aus dem Generatorpolynom erhalten wir die Bitsequenz: 10101
Dann führen wir die Division auf die Bitsequenz 101000110000, denn wir 0000 zum Ende der Bitsequenz einfügen, da der maximale Grad des Generatorpolynomes 4 ist: 1.5/1.5

```
101000110000 : 10101
10101
-----
 10110
 10101
-----
  11000
  10101
-----
   1101 (Rest)
```

Also ist die Prüfsumme 1101.

(b) Es gibt doch einen Fehler, da der Rest der Rechnung 011111000101:10101 nicht 0 ergibt. 1/1

```
011111000101 : 10101
10101
-----
 11010
 10101
-----
  11111
  10101
-----
   10100
   10101
-----
    10010
    10101
-----
     1111
```

(c) fehlt -1