



h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

fbi

FACHBEREICH INFORMATIK

Abschlussarbeit zum Seminar Wissenschaftliches Arbeiten in der Informatik I

Zeichenkodierung

von ASCII zu UTF-8

Sommersemester 2016

Betreuung

Inge Schestag

Autoren

Florian Zimmermann

Dennis Müssig

Ort, Abgabetermin

Darmstadt, 15.06.2016

Zusammenfassung

Die vorliegende Seminararbeit gibt einen Überblick über den zeitlichen Verlauf und Entwicklung der Zeichenkodierung. Dabei werden zunächst die wichtigsten Grundbegriffe erläutert. Danach folgt ein zeitlicher Überblick, von der Antike bis zur heutigen Zeit, über die Entwicklung der Zeichenkodierung. Am Ende wird der aktuelle technische Fortschritt genauer betrachtet und die Funktionsweise erklärt. Die Seminararbeit soll dem Leser einen Einstieg in die Thematik Zeichenkodierung geben.

Inhaltsverzeichnis

Zusammenfassung.....	II
1 Einleitung.....	1
2 Grundlegende Begriffe.....	2
2.1 Zeichensatz	2
2.2 Codepoint.....	3
2.3 Zeichenkodierung.....	3
3 Historie der Zeichenkodierung.....	4
3.1 Morse-Code.....	4
3.2 ASCII.....	5
3.3 Unicode.....	5
4 Funktionsweise von UTF-8.....	7
5 Schlusswort.....	9
Literaturverzeichnis.....	10
A Erklärung	11

1 Einleitung

Schon seit Beginn der Sprache und Schrift konnte man diese nicht immer direkt übertragen. Man war schon damals auf andere Übertragungen angewiesen, welche eine Kodierung erforderten. Damals waren es Rauchzeichen heute sind es elektrische Impulse. Aber um diese Informationen zu vermitteln brauchte man eine Kodierung, die sowohl Empfänger als auch Sender verstehen konnten. Im Laufe der Jahre wurde und musste diese Technik immer wieder an die neuen Begebenheiten angepasst werden. Die Kodierungen reichen von Rauchzeichen zu Signalflaggen bis hin zu neuesten technischen Methoden, um die Zeichen dieser Welt zu kodieren und dekodieren.

In Kapitel 2 wird zunächst eine kurze Einführung in die wichtigsten Begrifflichkeiten gegeben. Das dritte Kapitel befasst sich mit der Historie der Zeichenkodierung und der Veränderungen der Technologie. Zuletzt dient Kapitel 4 dazu, die Funktionsweise der aktuellen Kodierung zu erläutern sowie den aktuellen Stand der Technik näher zu bringen.

2 Grundlegende Begriffe

Im folgenden Kapitel werden die grundlegenden Begriffe erläutert, die zum Verständnis der Historie und der Funktionsweise von UTF-8 wichtig sind.

2.1 Zeichensatz

Der *Zeichensatz* (*character set*) enthält alle zur Verfügung stehenden Zeichen. Er beschreibt die Zeichen selbst sowie auch die Reihenfolge der Zeichen. Die Nummerierung legt die Reihenfolge fest, welche z.B. für die Sortierreihenfolge verantwortlich ist. Für die Abbildung von Zeichen in Byte-Werte ist der Zeichensatz nicht verantwortlich. Dafür ist die Zeichenkodierung, wie ASCII (*American Standard Code for Information Interchange*) oder UTF-8 (*Unicode Transformation Format*), zuständig.

The ASCII code

American Standard Code for Information Interchange

ASCII control characters			ASCII printable characters						Extended ASCII characters																					
DEC	HEX	Simbolo ASCII	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	DEC	HEX	Simbolo	
00	00h	NULL (carácter nulo)	32	20h	espacio	64	40h	@	96	60h	`	128	80h	Ç	160	A0h	á	192	C0h	À	224	E0h	Ó							
01	01h	SOH (inicio encabezado)	33	21h	!	65	41h	A	97	61h	a	129	81h	ü	161	A1h	í	193	C1h	Á	225	E1h	Ô							
02	02h	STX (inicio texto)	34	22h	"	66	42h	B	98	62h	b	130	82h	é	162	A2h	ó	194	C2h	Â	226	E2h	Ö							
03	03h	ETX (fin de texto)	35	23h	#	67	43h	C	99	63h	c	131	83h	â	163	A3h	ú	195	C3h	Ï	227	E3h	Ø							
04	04h	EOT (fin transmisión)	36	24h	\$	68	44h	D	100	64h	d	132	84h	ä	164	A4h	ñ	196	C4h	Ï	228	E4h	ø							
05	05h	ENQ (enquiry)	37	25h	%	69	45h	E	101	65h	e	133	85h	å	165	A5h	Ñ	197	C5h	Ï	229	E5h	Õ							
06	06h	ACK (acknowledgement)	38	26h	&	70	46h	F	102	66h	f	134	86h	ä	166	A6h	ª	198	C6h	Ï	230	E6h	µ							
07	07h	BEL (timbre)	39	27h	'	71	47h	G	103	67h	g	135	87h	ç	167	A7h	º	199	C7h	Ï	231	E7h	þ							
08	08h	BS (retroceso)	40	28h	(72	48h	H	104	68h	h	136	88h	ê	168	A8h	¿	200	C8h	Ï	232	E8h	Û							
09	09h	HT (tab horizontal)	41	29h)	73	49h	I	105	69h	i	137	89h	ë	169	A9h	®	201	C9h	Ï	233	E9h	Ü							
10	0Ah	LF (salto de línea)	42	2Ah	*	74	4Ah	J	106	6Ah	j	138	8Ah	è	170	AAh	¬	202	CAh	Ï	234	EAh	Ý							
11	0Bh	VT (tab vertical)	43	2Bh	+	75	4Bh	K	107	6Bh	k	139	8Bh	ï	171	ABh	½	203	CBh	Ï	235	EBh	Ú							
12	0Ch	FF (form feed)	44	2Ch	,	76	4Ch	L	108	6Ch	l	140	8Ch	î	172	ACH	¼	204	CCh	Ï	236	ECh	Ý							
13	0Dh	CR (retorno de carro)	45	2Dh	-	77	4Dh	M	109	6Dh	m	141	8Dh	ï	173	ADh	»	205	CDh	Ï	237	EDh	ÿ							
14	0Eh	SO (shift Out)	46	2Eh	.	78	4Eh	N	110	6Eh	n	142	8Eh	Ë	174	AEd	«	206	CEh	Ï	238	EEh	ÿ							
15	0Fh	SI (shift in)	47	2Fh	/	79	4Fh	O	111	6Fh	o	143	8Fh	Ä	175	AFh	»	207	CFh	Ï	239	EFh	ÿ							
16	10h	DLE (data link escape)	48	30h	0	80	50h	P	112	70h	p	144	90h	É	176	B0h	»	208	D0h	Ï	240	F0h	ÿ							
17	11h	DC1 (device control 1)	49	31h	1	81	51h	Q	113	71h	q	145	91h	æ	177	B1h	»	209	D1h	Ï	241	F1h	±							
18	12h	DC2 (device control 2)	50	32h	2	82	52h	R	114	72h	r	146	92h	Æ	178	B2h	»	210	D2h	Ï	242	F2h	—							
19	13h	DC3 (device control 3)	51	33h	3	83	53h	S	115	73h	s	147	93h	ø	179	B3h	»	211	D3h	Ï	243	F3h	¾							
20	14h	DC4 (device control 4)	52	34h	4	84	54h	T	116	74h	t	148	94h	ò	180	B4h	»	212	D4h	Ï	244	F4h	¶							
21	15h	NAK (negative acknowle.)	53	35h	5	85	55h	U	117	75h	u	149	95h	ó	181	B5h	»	213	D5h	Ï	245	F5h	§							
22	16h	SYN (synchronous idle)	54	36h	6	86	56h	V	118	76h	v	150	96h	û	182	B6h	»	214	D6h	Ï	246	F6h	÷							
23	17h	ETB (end of trans. block)	55	37h	7	87	57h	W	119	77h	w	151	97h	ü	183	B7h	»	215	D7h	Ï	247	F7h	°							
24	18h	CAN (cancel)	56	38h	8	88	58h	X	120	78h	x	152	98h	ÿ	184	B8h	»	216	D8h	Ï	248	F8h	ˆ							
25	19h	EM (end of medium)	57	39h	9	89	59h	Y	121	79h	y	153	99h	ÿ	185	B9h	»	217	D9h	Ï	249	F9h	˙							
26	1Ah	SUB (substitute)	58	3Ah	:	90	5Ah	Z	122	7Ah	z	154	9Ah	Ü	186	BAh	»	218	DAh	Ï	250	FAh	˘							
27	1Bh	ESC (escape)	59	3Bh	;	91	5Bh	[123	7Bh	{	155	9Bh	ø	187	BBh	»	219	DBh	Ï	251	FBh	˚							
28	1Ch	FS (file separator)	60	3Ch	<	92	5Ch	\	124	7Ch		156	9Ch	£	188	BCh	»	220	DCh	Ï	252	FCh	³							
29	1Dh	GS (group separator)	61	3Dh	=	93	5Dh]	125	7Dh	}	157	9Dh	Ø	189	BDh	»	221	DDh	Ï	253	FDh	²							
30	1Eh	RS (record separator)	62	3Eh	>	94	5Eh	^	126	7Eh	~	158	9Eh	x	190	BEh	»	222	DEh	Ï	254	FEh	■							
31	1Fh	US (unit separator)	63	3Fh	?	95	5Fh	-				159	9Fh	f	191	BFh	»	223	DFh	Ï	255	FFh	■							
127	20h	DEL (delete)																												
									theASCIIcode.com.ar																					

<https://commons.wikimedia.org/wiki/File:Ascii-codes-table.png>

2.2 Codepoint

Jedes Zeichen besitzt eine bestimmte Position in einem Zeichensatz. Diese Position nennt man den *Codepoint*. Einen Zeichensatz mit Codepoints nennt man einen *codierten Zeichensatz* (*coded character set*).

Ein Unicode-Zeichen wird in der Form **U+XXXX** beschrieben. Das **XXXX** steht hier für den Codepoint in dem Zeichensatz Unicode. Üblicherweise wird dieser in hexadezimaler Form angegeben. Zum Beispiel hat das Zeichen **€** den Codepoint **U+20AC**.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20A0	₠	¢	£	₣	₤	₥	₦	₧	₨	₩	₪	₫	€	₭	₮	₯
20B0	₱	₲	₳	₴	₵	₶	₷	₸	₹	₺	₻	₼	₽	₾	₿	
20C0																

<http://unicode-table.com/de/#currency-symbols>

2.3 Zeichenkodierung

Die Zeichenkodierung beschreibt die konkrete Zuweisung eines Codepoints zu einer Bytesequenz. So beschreibt die Kodierung UTF-8 den Zeichensatz Unicode, welcher in Kapitel 3.3 genauer betrachtet wird.

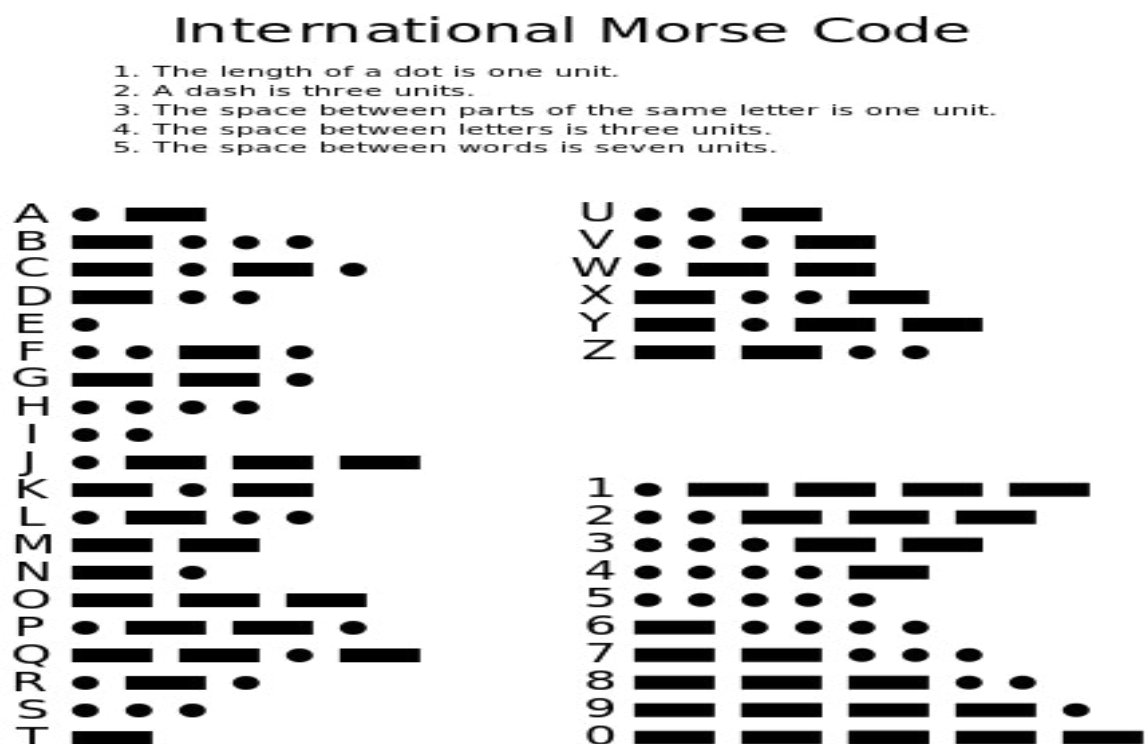
Noch immer existiert das Synonym *Zeichentabelle* (*code page*) in der Informatik. Die Zeichenkodierung UTF-8 wird, zum Beispiel in einem Windows-Betriebssystem, als *Codepage* bezeichnet.

3 Historie der Zeichenkodierung

Dieses Kapitel betrachtet den zeitlichen Verlauf der Zeichenkodierung. Der Ursprung der Zeichenkodierung liegt in der Antike. Damals wurden Truppen über Angriffe informiert, mithilfe von *Feuer- und Lichtzeichen*. Bekannt sind auch die *Rauchzeichen* der Indianer oder *Trommelsignale* in Afrika. Im Laufe der Jahre entwickelte sich diese Technik weiter zu Kodierungen mit Flaggen bis hin zu Kodierungen mit Byte im digitalen Zeitalter.

3.1 Morse-Code

Mit der Erfindung der *elektrischen Telegrafie* von *Samuel Morse* im Jahre 1837, löste man die *optischen Balkentelegrafen* ab. Der Balkentelegraf arbeitete mit 4096 Stellung mit 6 Flügeln, die am oberen Ende eines Mastes montiert waren. Die elektrischen Telegrafen waren zuverlässiger und weit überlegen im Bereich der Geschwindigkeit. Allerdings musste man nun alle Zeichen, Zahlen und Buchstaben in elektrische Impulse umwandeln. 1844 erfand Morses Assistent *Alfred Vail* das erste *Morse-Alphabet*. Es ist für Menschen leicht einzugeben und ist heute immer noch im Einsatz. Erst 2004 wurde das Alphabet um das *@-Zeichen* erweitert.



Das wohl bekannteste abbildbare Wort in Morse ist das Hilfezeichen S.O.S. Mit der obigen Grafik lässt sich das Wort leicht bilden.

Mit dem Bau von Zeigetelegraphen und Fernschreibern musste man aber auf eine neue Kodierung zurückgreifen, da der Morse-Code für Maschinen zu langsam ist. Der *Baudot-Code*, erfunden von *Émile Baudot*, ließ sich einfach von Maschinen verarbeiten. Der Code wurde über eine Tastatur mit 5 Tasten eingegeben. Jedes Zeichen war gleich lang, anders als beim Morsecode. So entstand ein 5-Bit-Wort, welches die maschinelle Entschlüsselung einfacher zu realisieren machte. Dieser wurde bis 1967 ständig weiterentwickelt und selbst von den ersten Computer genutzt.

3.2 ASCII

Im Jahre 1967 wurde von der US-amerikanischen Standardisierungsorganisation *ASA (American Standards Association)*, die 7-Bit-Zeichenkodierung *ASCII (American Standard Code for Information Interchange)* ins Leben gerufen. Die Mitarbeit der *Internationalen Standardorganisation (ISO)* und der europäischen Computer-Hersteller *ECMA (European Computer Manufactures Association)* machte dies möglich und löste den Vorgänger ASCII-1963 ab. Mit der Einräumung genügender Steuerzeichen konnte ASCII für Computer sowie Fernschreiber funktionieren. Die Zeichenkodierung bestand anfangs aus 127 Zeichen (33 Steuerzeichen und 95 Druckzeichen). Später erweiterte man diese noch durch das achte Bit für Fehlerkorrekturzwecke. Heutzutage wird es als Zeichenerweiterung genutzt.

3.3 Unicode

Mit der Zeit merkte man, dass viele *ASCII-Codepages* eine erhebliche Einschränkung in der Kommunikation mit anderen Nationen darstellt. Die Zeichentabelle *Unicode* wurde Ende der 80er Jahren erfunden. Sie dient als universelle Zeichentabelle für alle Sprachen der Welt. Über die Jahre nahm man für alle *lebendige Sprachen*, sowie auch für viele *tote Sprachen (z.B. Gotische Sprache)*, die Zeichen auf. Sind diese einmal aufgenommen, so bleiben diese auch erhalten. Nachträgliche Änderungen sind nicht möglich, um die Eindeutigkeit der Codepoints zu gewährleisten. Der Unicode definiert nicht nur Zeichen sondern auch *Leserichtung*, *Sortierreihenfolge* und *Kombinationsregeln*. Zurzeit enthält die Zeichentabelle mehr als 1 Millionen Zeichen. Betriebssysteme beschränken sich aber nur auf sinnvolle Teilmengen. Das Windows-

Betriebssystem beschränkt sich auf 652 Zeichen. So kann ein deutsch- oder englischsprachiges System immerhin lateinische, griechische, kyrillische, arabische und hebräische Texte darstellen. Weitere Zeichen können dann über Sprachpakete hinzugefügt werden.

Das *Unicode Transformation Format (UTF)* ist eine Kodierungsmethode, um Zeichen des Unicodes abzubilden. Mit UTF lassen sich alle vorhandenen Zeichen im Unicode abbilden, weshalb dies auch der legitime Nachfolger der ASCII-Zeichenkodierung ist. Die UTF-Zeichenkodierung kann zwölf unterschiedliche Formate zur Codierung der Zeichen verwenden, z.B UTF-8, UTF-16, UTF-32 und UTF-EBCDIC. Diese Formate unterscheiden sich hinsichtlich ihre Speichereffizienz, Kodierungs- und Dekodierungsaufwand sowie in ihrer Kompatibilität zu anderen (älteren) Kodierungsarten wie zum Beispiel ASCII. Im Laufe des nächsten Abschnittes wird UTF-8 genauer betrachtet, weil es die häufigste und bekannteste Kodierung ist. UTF-8 ist aber, trotz seiner häufigen Nutzung, noch kein anerkannter Standard.

In Anlehnung an: *Braun, Herbert: Kleine Geschichte der Zeichensätze (09.2006)*,
<http://www.heise.de/ct/Redaktion/heb/zeichensaetze.html>

4 Funktionsweise von UTF-8

In diesem Abschnitt wird die funktionale Vorgehensweise der *Kodierung von UTF-8* anhand eines Unicode-Zeichens genauer erläutert.

Die *Unicode-Zeichen* mit den Werten aus dem Bereich von 0 bis 127 (0 bis 7F) werden in der UTF-8 Kodierung als ein Byte mit dem gleichen Wert weitergegeben. Dadurch, dass es UTF-8 möglich ist, Zeichen mit einer variablen Länge von Bytes zu kodieren, unterscheidet es sich bei der Kodierung der ersten 127 Zeichen nicht von ASCII. UTF-8 benutzt für die Kodierung der Zeichen eine maximale Länge von 4 Byte großen *Byteketten*. Der Unicode in seiner Verwendung als UTF-Kodierung ist auf den gemeinsamen Coderaum aller Unicode-Kodierung beschränkt, also von 0 bis 0010 FFFF. Das erste Byte eines UTF-8 kodierten Zeichens nennt man *Startbyte*, weitere Bytes heißen *Folgebytes*. Die Startbytes beginnen immer mit 0 oder 11, die Folgebytes immer mit 10.

Unicode-Bereich	UTF-8 Kodierung	Bemerkung
0000 0000 – 0000 007F	0xxxxxxx	ASCII-Zeichen
0000 0080 – 0000 07FF	110xxxxx 10xxxxxx	Mehrbytezeichen
0000 0800 – 0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx	
0001 0000 – 0010 FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	

Die obige Tabelle zeigt, dass bei UTF-8 Kodierungen, bei denen das höchste Bit des ersten Bytes mit einer 0 beginnt, es sich um ASCII-Zeichen handelt. Damit sind alle ASCII-Zeichenketten dementsprechend aufwärtskompatible zu UTF-8. Würde das erste Bit des ersten Bytes eine 1 enthalten, würde dies indizieren, dass es ein Unicode Zeichen ist. Bei genauerem Betrachten der UTF-8 Kodierung der zweiten Zeile erkennt man sofort, dass es sich um *Mehrbytezeichen* handelt. Dies erkennt man daran, dass das höchste Bit des ersten Bytes durch ein 11 angeführt wird und somit als Startbyte dient. Aufgrund dessen, dass es sich nun um ein Mehrbytezeichen handelt, muss man das zweite Byte nun als Folgebyte mit 10 kennzeichnen. Die Anzahl der Einsen im Startbyte geben an, ob es sich um eine *2-Byte*, *3-Byte* oder *4-Byte* lange Bytekette handelt. Dafür werden immer die *Einsen* im Startbyte bis zum höchsten Null-Bit gezählt.

Um dies besser zu veranschaulichen, dient das *Währungszeichen* € als Beispiel. Das €-Zeichen hat den *Codepoint* U+20AC. Der Hexadezimal-Wert zeigt, dass es sich um ein Mehrbytezeichen handelt. Nach der Kodierung erhalten wir eine 3-Byte lange Kette. Die Maske sieht dementsprechend so aus: **1110xxxx 10xxxxxx 10xxxxxx**. Nun muss der Codepoint noch als Binärzahl umgerechnet werden. Somit ergibt sich die Binärzahl: **00100000 10101100**. Als letzten Schritt muss die Binärzahl in die Maske eingetragen werden. Dies sieht folgendermaßen aus:

Maske: **1110xxxx 10xxxxxx 10xxxxxx**

Binärzahl: **00100000 10101100**

Ergebnis: **11100010 10000010 10101100**

5 Schlusswort

Die Kommunikation zwischen den Menschen forderte schon vor vielen Jahren eine Kodierung, vor allem wenn die Kommunikation nicht direkt und unmittelbar erfolgen konnte. Die Art der Zeichenkodierung passte sich immer den Gegebenheiten der Zeit an. Im Laufe der Zeit wurde diese immer komplexer. Heutzutage ist dieses System der Zeichenkodierung mit dem Unicode und der Kodierung UTF-8, dank der Geschichte, eine zuverlässige Technologie, die aber immer noch weiterentwickelt wird. Man lernte aus den Fehlern der Vergangenheit und verbesserte somit die Technologie der Zeichenkodierung.

Literaturverzeichnis

1. <https://wiki.selfhtml.org/wiki/Zeichenkodierung> (*Stand: 07.06.2016*)
2. <https://de.wikipedia.org/wiki/Zeichenkodierung#Geschichte>
(*Stand:08.06.2016*)
3. https://de.wikipedia.org/wiki/UTF-8#Byte_Order_Mark (*Stand: 11.06.2016*)
4. Braun, Herbert: Kleine Geschichte der Zeichensätze (09.2006), <http://www.heise.de/ct/Redaktion/heb/zeichensaetze.html> (*Stand: 08.06.2016*)
5. Herrmann, Ralf: Die Entwicklung der Zeichenkodierung, Teil 1 (18.02.2011),
<http://www.typografie.info/3/artikel.htm/wissen/zeichenkodierung-teil1/>
(*Stand: 13.06.2016*)
6. https://de.wikipedia.org/wiki/Optische_Telegrafie (*Stand: 13.06.2016*)
7. <https://de.wikipedia.org/wiki/Baudot-Code> (*Stand: 13.06.2016*)

A Erklärung

Wir versichern hiermit, dass wir die vorliegende Arbeit selbstständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von uns selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, den 15.06.2016