Lidar/Radar-User-Unit-Software-Entwicklungs-dokumentation

Projektbezeichnung	Unmanned Surface Vehicle
Projektleiter	Prof. Dr.-Ing. Habil. Jörg Grabow
Verantwortlich	Dang Hoang Ha, Thach
Erstellt am	23.07.2023
Neuste Änderungen am	01.08.2023
Bearbeitungsstand	In Bearbeitung
Version/Revision	1.2/1.01
Dokumentenablage	https://github.com/hathach23/USV/tree/Develop/01%20Hardware/01%20Lidar/03%20Software/00%20doc

Änderungsverzeichnis

Änderungsinformation				Beschreibung	Autor	Neuer Zustand
Nr.	Datum	Version	Kapitel			
1	23.07.2023	1.0	1	Erste Version	Thach	Fertig gestellt
2	27.07.2023	1.1	2	Beschreibung für das Testen der USART-Einheit des Mikrocontrollers hinzugefügt	Thach	Fertig gestellt
3	27.07.2023	1.1	2	Korrektur des Rechtschreibungsfehlers	Thach	Fertig gestellt
4	31.07.2023	1.1	2.1.2.	Korrektur der Beschreibung der Kommunikation Methode von User-Unit-Modul	Thach	Fertig gestellt
5	01.08.2023	1.2	3	Beschreibung der Implementierung der Kommunikation zwischen dem User-Unit- und dem Interface-Modul	Thach	In Bearbeitung
6	03.08.2023	1.2	3.3.	Beschreibung des Aufbaus des Moduls	Thach	In Bearbeitung

1. Einleitung:

Die User-Unit-Einheit übernimmt die Aufgabe, rohe Daten aus dem Sensor zu lesen, zu verarbeiten und die verarbeiteten Daten in Interface-Einheit der Slave-Baugruppe weiter zu schicken und Konfiguration aus Interface-Einheit der Slave-Baugruppe zu empfangen, zu verarbeiten und ins Sensor zu schreiben. Sie dient als der Interprator in Slave-Baugruppe.

Diese Dokument wird erstellt zur Beschreibung der Entwicklungsprozess des Treibers der User-Unit-Einheit

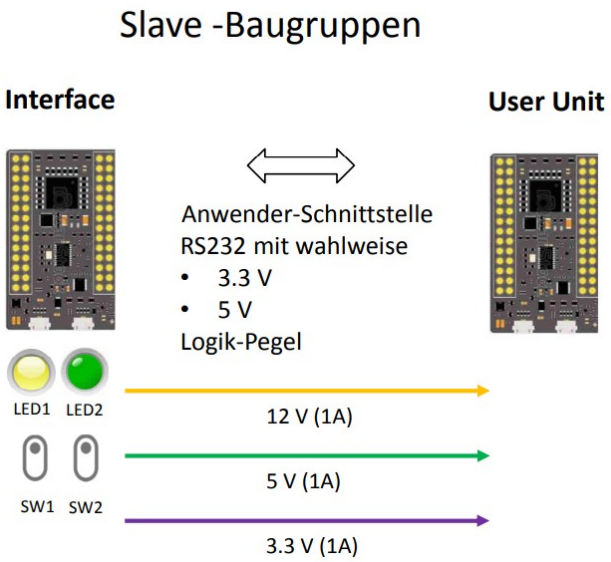

Abbildung 1: Aufbau der Slave-Baugruppe[1]

2. Funktionalitätstesten der USART-Einheit des Mikrocontrollers:

2.1. Vorkenntnisse:

2.1.1. USART:

USART vom Mikrocontroller ist ein Modul, das die Kommunikation von Mikrocontroller miteinander oder mit anderen Peripheriegeräten unter Anwendung des Prinzips der Serial Kommunikation und der Schnittstelle von UART realisiert.

Die Pinlbelegung von UART (mit Flow-Control) wird in Abbildung 2 dargestellt

Für das Logik-Pegel für die Daten des USART-Moduls ist es vom Mikrocontroller abhängig. Der für das User-Unit-Gerät verwendete Mikrocontroller, nämlich ATMega4808, verwendet folgenden Logik-Pegel (siehe Tabelle 1)

Tabelle 1: Logik Pegel für ATMega4808 [2]

Logik-Wert	Logik-Pegel
0	Max. 0,3*$V_{dd:}$ 1,5V bei V_{dd} = 5,0V und 1,05V bei V_{dd} = 3,5V
1	Min. 0,7*Vdd: 3,5V bei V_{dd} = 5,0V und 2,45V bei V_{dd} = 3,5V

2.1.2. RS-232:

RS-232 ist ein Standard für Übertragungsprotokoll der seriellen Kommunikation. Es legt die Parameter für eine problemlose Kommunikation zwischen Geräten fest, nämlich den Spannugnspegel (siehe Tabelle 2), Steckerverbindung und Pinbelegung(siehe Abbildung 3) zwischen DTE(Data Terminal Equipment) und DCE (Data Circuit-Terminating Equipment)sowie Steuerinformation zwischen DTE und DCE[4].

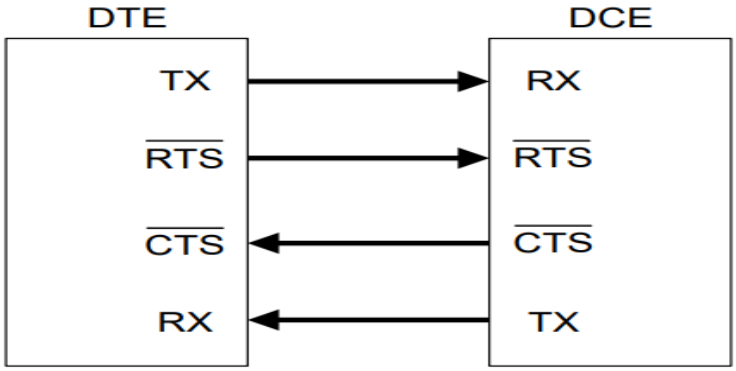

Abbildung 2: Pinbelegung bei UART mit Flow-Control bei Hardware[3]

Tabelle 2: Logik Pegel für RS232[5]

Logik-Wert	Logik-Pegel
0 (Space)	3V bis 15V
1 (Mark)	-3V bis -15V

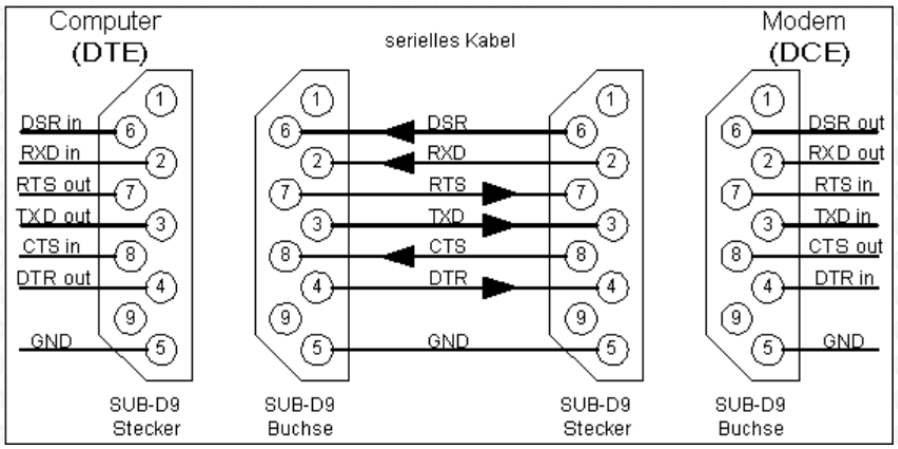

Abbildung 3: Pinbelegung bei RS232[6]

Die Hardware-Bestandteile der UART mit Flow-Control und RS232 ist ziemlich ähnlich zueinander. Der Unterschied ist der Logikspannungspegel. Das Modul von User-Unit verwendet die USART-Einheit vom Mikrocontroller zur Implementieren der Kommunikation über die RS232-Bus.

Aus diesem Grund ist eine gute funktionierende UART-Einheit eine der notwendigen Voraussetzung für eine problemlose Kommunikation zwischen Interface- und User-Unit-Einheit
Um die USART-Einheit in Betrieb zu nehmen und ihre Funktionalität zu tesen wird ein klein Testprogramm durchgeführt.

2.2. Angewendete Hardware und Software:

Die Hardwares sind der User-Unit-Mikrocontroller, nämlich ATMega4808, eine USART-zu-USB-Treiber, ein Computer (siehe Abbildung 4 für die Anordnung)
Die Softwares sind das Programm Hterm im Computer zum Lesen und Schreiben über serieller Schnittstelle
Zusätzlich wird ein Hardware-Abstract-Layer (HAL) Bibliothek für den Mikrocontroller angegeben

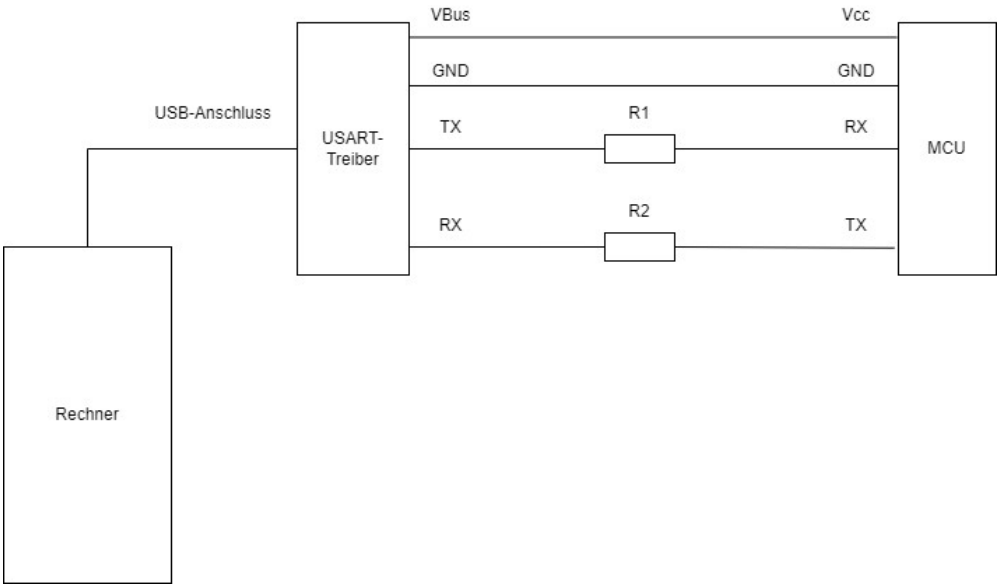

Abbildung 4: Physikalische Anordnung des Versuches

2.3. Beschreibung des Funktionalitätstesten

In diesem Test wird die Kommunikation zwischen einem Computer und dem Mikrocontroller über USART-Einheit durchgeführt. Der Computer dient als Master und Mikrocontroller als Slave. Der Master sendet eine Zeichenfolge zum Slave und der Slave muss das Zeichenfolge zusammen mit ihrer Länge zum Master zurücksenden. Die Anforderung ist, dass die User-Unit-Einheit (ab hier als Slave genannt) die Zeichenfolge unterscheiden muss, z.B: zwei separate Zeichenfolge: AAAAA und BBBB müssen richtig getrennt anstatt gemischt zueinander werden

2.3.1. Implementationsmöglichkeiten:

Es gibt zwei Möglichkeiten zur Realisierung der Anforderung. Sie sind wie folgt beschrieben:

1. Möglichkeit: Das Zähler/Timer – Modul vom Slave wird als USART-Wächter festgelegt. Bekommt der Slave nach einem bestimmten Zeit (Timeout-Zeit) kein Zeichen mehr, werden alle bisherige empfangene Zeichen als eine Zeichenfolge festgestellt und in ein FIFO gespeichert und der Empfangvorgang beginnt am Vorne mit neuer Zeichenfolge. Die Method wird als „Text-Beendungsmethode mit Timeout-Mechanismus“ oder 1.Methode genannt.

2. Möglichkeit: Ein Sonderzeichen wird als Beendungssymbol der Zeichenfolge (ab hier als Endsymbol genannt) festgestellt. Der Inhalt jedes kommenden Zeichen wird betrachtet. Wenn das Zeichen gleich dem Endsymbol ist, werden alle bisherige empfangene Zeichen als eine Zeichenfolge festgestellt und in ein FIFO gespeichert und der Empfangvorgang beginnt am Vorne mit neuer Zeichenfolge. Die Method wird als „Text-Beendungsmethode mit Sonderzeichen“ oder 2. Methode genannt.

Das Ablaufdiagramm beider Methoden wird in Abbildung 5 und 6 dargestellt.

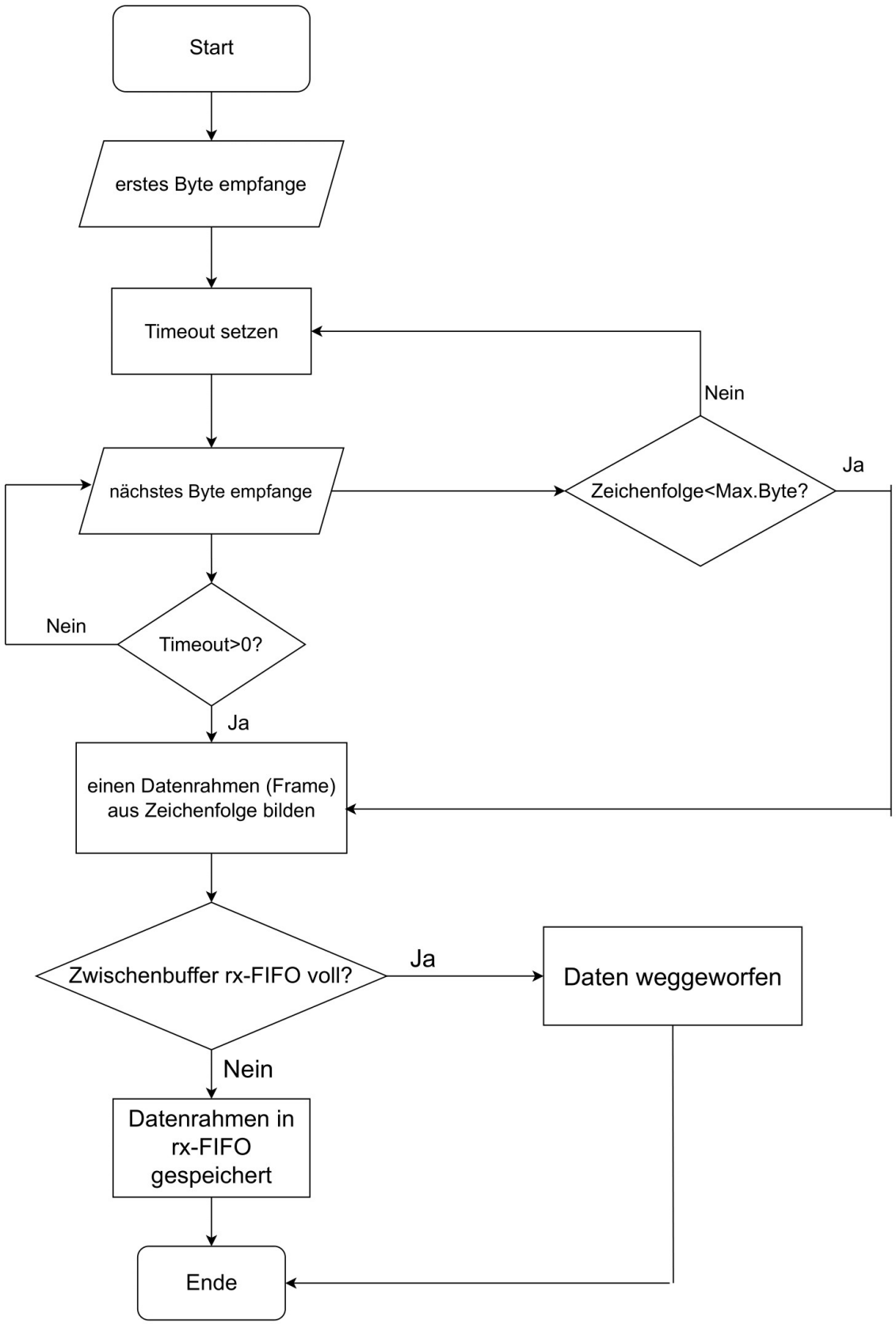

Abbildung 5: Ablaufdiagramm von 1. Methode

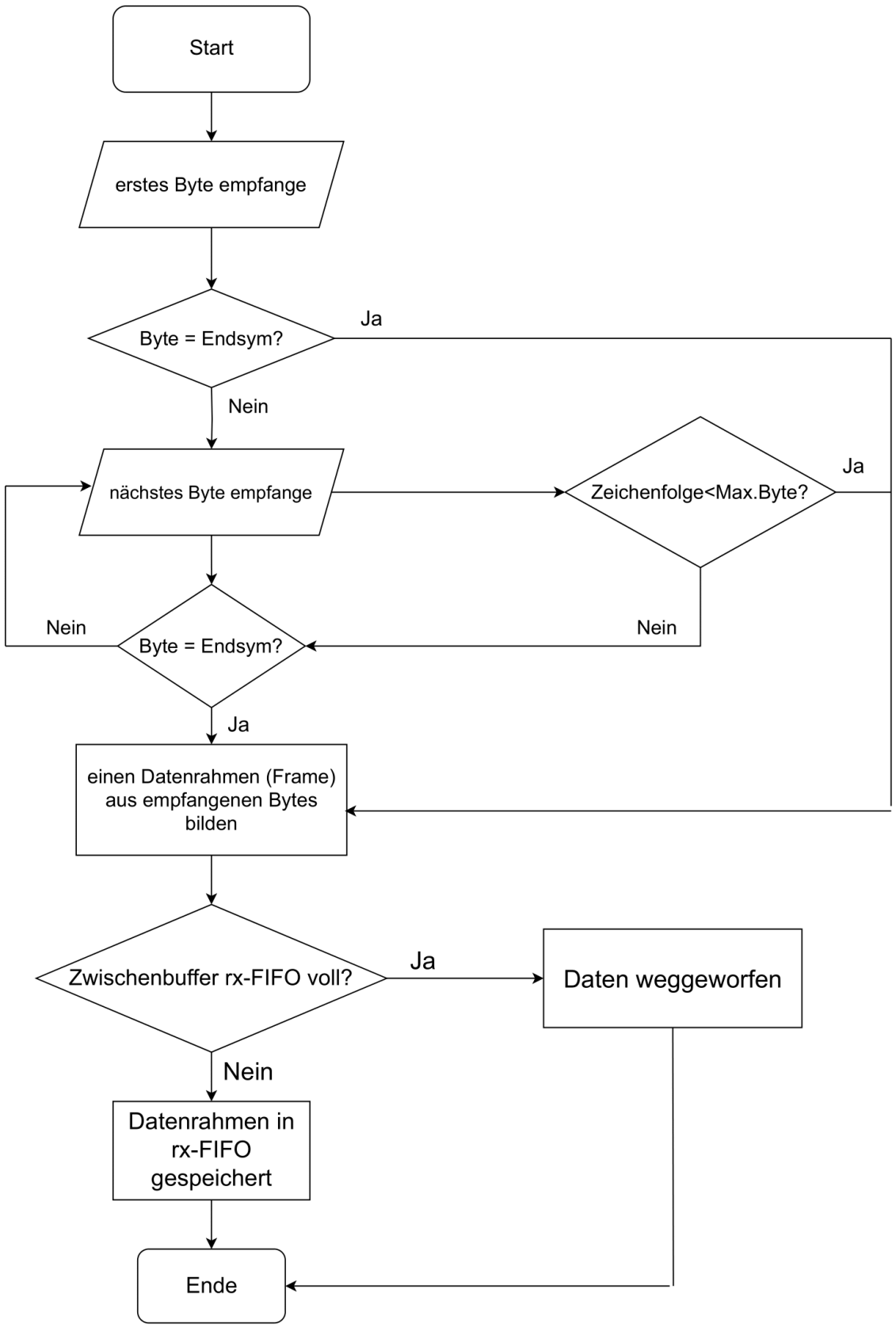

Abbildung 6: Ablaufdiagramm von 2. Methode

Anmerkung für beide Abbildung: Alle empfangene Bytes werden vorläufig in einem Zwischenbuffer gespeichert (deren Länge vordefiniert ist) bevor sie in Zwischenbuffer rx-FIFO gespeichert ist.

Beide Methode werden in Quellcode umgesetzt und die Leistungsversuch wird durchgeführt, um die beiden zu vergleichen und dadurch um eine passende Methode auszuwählen. Das Ergebnis wird in Tabelle 3 dargestellt

Tabelle 3: Bewertung der Leistung und der Fähigkeit beider Methoden

Methode / Kriterien	Text-Beendung mit timeout-Zeit (1. Methode)	Text-Beendung mit Sonder-zeichen (2. Methode)
Notwendiges Modul	2 (Timer/Counter, USART)	1 (USART)
Max. Baudrate	76800 Baud	250000 Baud
Verwandte Interrupts	1-2 (ein von USART und ein von Timer)	1 (ein von USART)
Aufwand	Aufwendig, da mehr Schritte	Leicht, weil wenige Schritte
Belastung auf Mikrocontroller	Hoch, da zwei Moduls arbeiten müssen	Mittel, da nur ein Modul arbeiten muss
Nuztbyte (Nutzzeichen)	n: die Anzahl der zu empfangenden Daten	n-1: die Anzahl der zu empfangenden Daten

Nach der Bewertung beider Methoden wird die Methode der Text-Beendung mit Sonderzeichen ausgwählt, da sie bei hoher Baudrate arbeiten kann und ihre Belastung auf Mikrocontroller weniger als die der 1. Methode. Dazu ist der Aufwand leichter als die erste. Obwohl deine Nutzbytes etwa kürzer als die der 1. Methode ist, ist der Unterschied bei sehr langer Zeichenfolge sehr klein.

Im Programm wird der Slave als eine Struktur objektiviert. Die Struktur wird in Abbildung 7 dargestellt und die Beschreibung jedes Bereiches der Struktur wird in der Tabelle 4 geschrieben

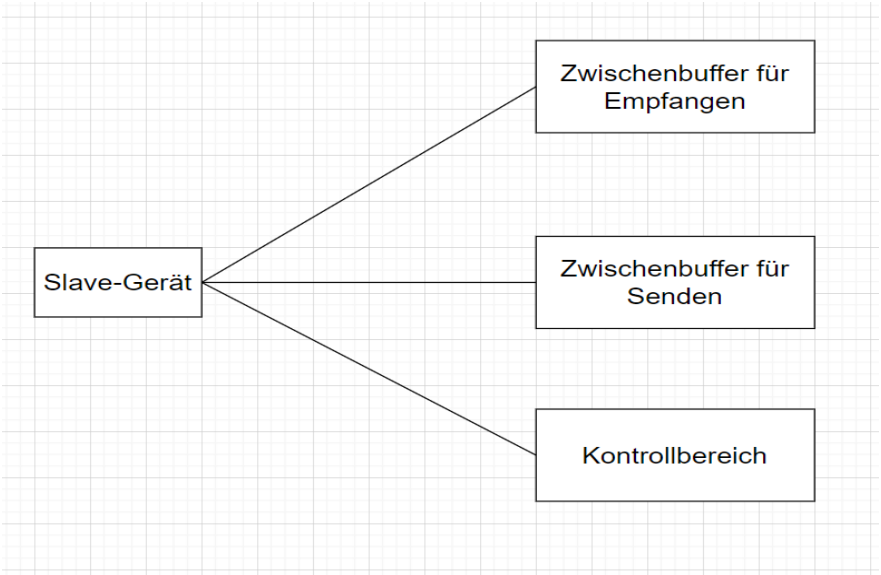

Abbildung 7: Strukturierung des Slaves im Programm

Tabelle 4: Name und Aufgabe der Bereiche der Slave-Gerät

Name des Bereiches	Beschreibung
Zwischenbuffer für Empfangen	Buffer, wobei die empfangenen Zeichenfolgen gespeichert werden, bevor sie gelesen und bearbeitet werden. Als Ring-FIFO organisiert
Zwischenbuffer für Senden	Buffer, wobei die zu sendenen Zeichenfolgen gespeichert werden. Es ist nur ein normales Array,
Kontrollbereich	Alle Informationen bezüglich des Slave-Gerätes werden hier gespeichert

Das gesamte Programm arbeitet mit der Polling-Methode. Es wartet auf die Daten aus der Seite des Masters. Wenn es die Daten aus der Masterseite bekommt, berechnet es die Länge und schicke zusammen mit der Originaldaten zum Master zurück. Der gesamte Programmablauf wird in Abbildung 8 dargestellt:

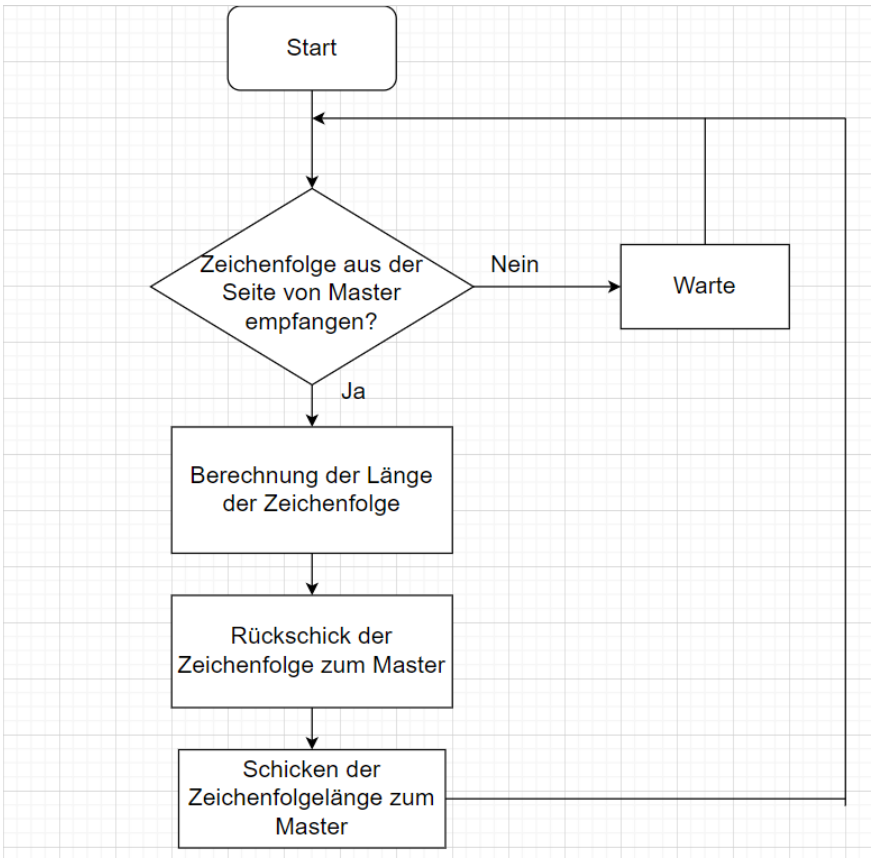

Abbildung 8: Ablaufdiagramm des gesamten Programm

2.3.2. Ausführungsergebnis und Verbesserungsmöglichkeiten

Nach der Ausführung des Programms mit beider Methoden lässt sich feststellen, dass die USART-Einheit des Mikrocontrollers zusammen mit der HAL-Bibliothek gut funktionieren
Die Implemetation der ersten Methode ist bei Hardware besser als bei Software.
Für die 2. Methode kann man unter Anwendung des Flow-Control Mechanismuses verbessern. Mit Implementation des Flow-Control Mechanismuses ist die maximale Baudrate, in dem das Programm noch gut funktioniert, erhöht. Jedoch ist es auch schwierig, da die Master-Seite auch den Flow-Control Mechanismus unterstützen muss

3. Implementierung der Kommunikation zwischen dem User-Unit- und dem Interface-Modul
3.1. Beschreibung:

In diesem Abschnitt wird die Programm zur Behandlung der Kommunikation zwischen dem User-Unit- und Interface-Modul bei der Seite vom User-Unit implementiert.

3.2.Durchführung:

Wie die Beschreibung in der Abschnitt 2.1.2 und in der Abbildung 1 wird die Kommunikation über RS232-Bus mit Hilfe der UART-Schnittstelle des Mikrocontrollers realisiert.

Die Implementation erfolgt wie folgt:
1. Schritt: Das Programm wird erst über UART-Schnittstelle ohne Flow-Control implementiert.
2. Schritt: Das Test wird durchgeführt zwischen dem echten User-Unit-Modul und dem Simulator des Interface-Moduls.
3. Schritt: Nach dem erfolgreichen Test wird die Hardware in RS232-Bus umgebaut und wird das Programm refaktorisiert, damit es mit dem RS232-Bus passt.
4. Schritt: Das Test wird durchgeführt zwischen dem echten und User-Unit- und Interface-Modul.

3.3. Aufbau des Moduls:

- Das Modul wird in zwei Teile geteilt: ein Teil ist API (Application Program Interface) und ein der Treiber
- Das API übernimmt die Aufgabe zur Behandlung der Kommunikation mit dem Interface-Modul und der Treiber definiert wie die Grundfunktionen des APIs implementiert

- Der Treiber wird geschrieben unter Anwendung der gegebenen HAL-Bibliothek

- Das Struktur vom API wird in Abbildung 9 dargestellt

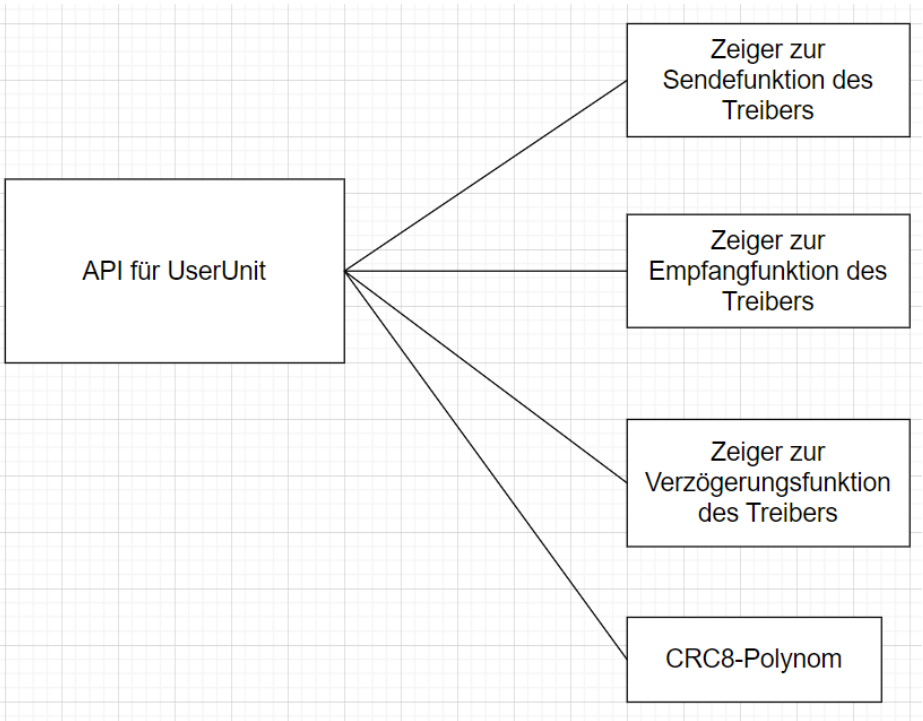

Abbildung 9: Struktur des APIs

- Um das API zu verwenden muss man die Funktionen zu ansprechenden Zeigern verbinden. Die Funktionen im API definieren den Ablauf der Kommunikation zwischen User-Unit-Modul und Interface-Modul

Implementation des APIs ist ähnlich wie die vom Shared-Library[7]. Diese Anordnung hat den Vorteil, dass man nur den Treiber wieder schreiben muss und danach die Verbindung zwischen API und Treiber wieder verbindet, wenn die Hardware wechselt wird. Es spart die Entwicklungszeit

Quellen und Literaturen

[1]: Interne Dokument „Buskonzept_Übersicht.pdf“
Link: https://github.com/Joe-Grabow/USV/tree/main/00%20doc/00%20Bussystem
Zugriff am 23.07.2023
[2]: Microchip: ATmega4808/4809 Data Sheet DS40002173A: Abschnitt 32.11: I/O Pin Characteristics
[3]: Michael Wyman and Christopher Best, Microchip Technology Inc. : Basic Operation of UART with Protocol Support TB3208
[4],[5]: Dallas Semiconductor: Application Note 83 Fundamentals of RS232 Serial Communication
[6]: Link: http://www.sprut.de/electronic/interfaces/rs232/rs232.htm
Zugriff am 27.07.2023
[7]: Shared Libraries on UNIX System V : James Q. Arnold, Herausgeber: USENIX (1986) Seite 395-404
Links: https://archive.computerhistory.org/resources/access/text/2020/12/102713986-05-01-acc.pdf
Zugriff am 03.08.2023