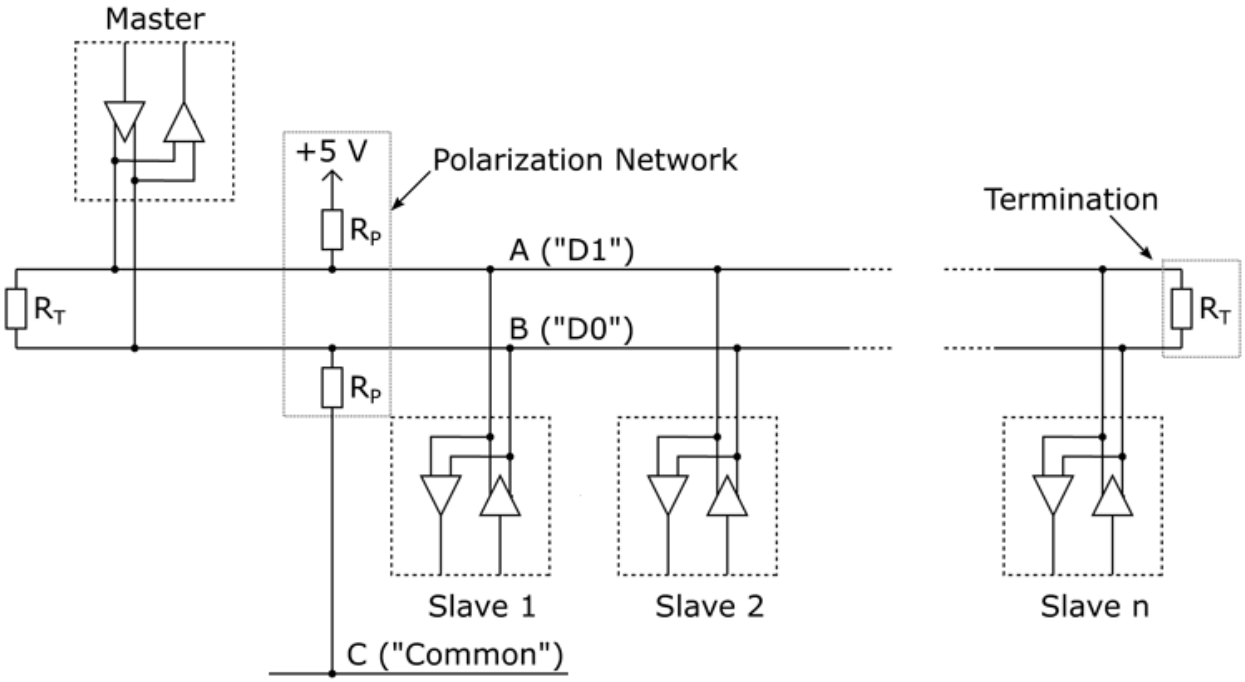
MODBUS Teil 1

Inhaltsverzeichnis

1. RS485

Der RS-485 Bus ist ein zusätzliches Interface auf Basis des bekannten UART-Bussystems. Aufgrund der Änderungen am Aufbau erhält der UART-Bus eine Linienbusstruktur, die es ermöglicht, mehrer Busteilnehmer über längerer Distanz miteinander zu verbinden.

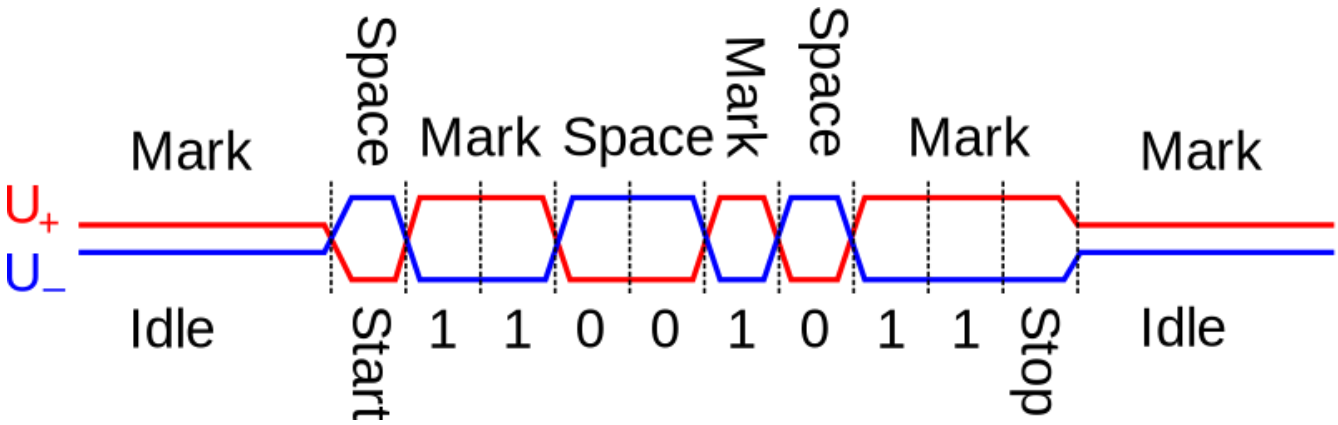


Bildquelle (vgl.): https://www.drago-automation.de/tl_files/images/modbus-rs485.png

Bei der RS-485-Schnittstelle sollten die Leitungsenden (zumindest bei größeren Leitungslängen bzw. größeren Übertragungsraten) abgeschlossen werden. In der Regel wird ein passiver Abschluss R_T durch Verbinden der Signalleitungen über jeweils einen 120 Ω Widerstand an den beiden Leitungsenden verwendet. Ein optionales Bias-Netzwerk R_P verbessert für den Fall inaktiver Leitungstreiber den Störabstand, der sonst lediglich durch die Hysterese des Empfängers gegeben ist. Die Schnittstelle benutzt in der Regel nur ein Adernpaar und wird halbduplex betrieben. Nachfolgend werden die Spezifikationen nach IEA-485 aufgeführt.

Parameter	**Wert**
Anzahl Empfänger	32 Empfänger
Maximale Leitungslänge	1200 m
Maximale Datenübertragungsrate	12 Mbps
Gleichtakt-Eingangsspannung	−7 V bis +12 V
Eingangswiderstand des Empfängers	12 kΩ (1 Unit Load)
Eingangsempfindlichkeit des Empfängers	±200 mV

RS-485 benutzt ein Leitungspaar, um den invertierten und einen nichtinvertierten Pegel eines 1-Bit-Datensignals zu übertragen. Am Empfänger wird aus der Differenz der beiden Spannungspegel das ursprüngliche Datensignal rekonstruiert. Das hat den Vorteil, dass sich Gleichtaktstörungen nicht auf die Übertragung auswirken und somit die Störsicherheit vergrößert wird. Im Gegensatz zu RS-232 sind so wesentlich längere Übertragungsstrecken und höhere Taktraten möglich.



Bildquelle: https://upload.wikimedia.org/wikipedia/commons/f/f2/RS-485_waveform.svg

Im Gegensatz zu anderen Bussystemen sind bei RS-485 nur die elektrischen Schnittstellenbedingungen definiert. Das Protokoll kann anwendungsspezifisch gewählt werden (im Praktikum MODBUS). Deshalb werden sich RS-485-Geräte unterschiedlicher Applikationen oder Hersteller i. Allg. nicht verstehen. Sollen Daten transportiert werden, bedient man sich zur Zeichenübertragung oft des Universal Asynchronous Receiver Transmitter-Protokolls (UART), bekannt von RS-232-Schnittstellen. Meist werden hier acht gleichwertige Bits pro Rahmen übertragen.

2. MODBUS Richtlinien

Im Versuch "MODBUS Teil 1" sollen Sie ein eigenes Protokoll auf Basis von *MODBUS RTU* oder *MODBUS ASCII* oder auch einer Kombination aus beiden Standards entwerfen.

Übertragungsmodus der seriellen Verbindung

Aufgrund der Limitierung durch die Klasse *SoftwareSerial* wird nicht das Standardzeichenformat von MODBUS RTU bzw. MODBUS ASCII verwendet. Zu verwenden ist folgendes Zeichenformat und wahlweise eines der gegebenen Baudraten:

a) Zeichenformat: 1 Start-, 8 Daten-, 1 Stopbit, keine Parität
b) Baudrate [bps]: 1200, 2400, 4800, 9600, 14400, 19200, 31250, 38400, 57600, 11520

Telegramm Spezifikation

In den folgenden Abschnitten werden die wichtigsten Standardrichtlinien für *MODBUS RTU* und *MODBUS ASCII* beschrieben.

Aufbau des Telegramms

Telegramm MODBUS RTU			
Geräte-Adresse	*Funktion*	*Daten*	*CR-Check*
1 Byte	1 Byte	n * Bytes	2 Bytes

Telegramm MODBUS ASCII			
Geräte-Adresse	*Funktion*	*Daten*	*LR-Check*
2 Zeichen	2 Zeichen	n * Zeichen	2 Zeichen

Geräte-Adresse: Gibt an, welches Gerät angesprochen werden soll. Erlaubt sind bei MODBUS die Adressen: 1 bis 247, wobei bei MODBUS ASCII die Adresse als ASCII Zeichen in hexadezimaler Schreibweise übertragen werden. Die Adresse 0 kann für Mitteilungen an alle Geräte (Broadcast) verwendet werden, sofern die gewählte Funktion dies unterstützt. **Sollte das adressierte Gerät im Bus nicht reagieren, wird nach einem Timeout eine definierte Anzahl an Neuversuchen gestartet. (Timeout festlegen)!**

Funktion: Gibt den Zweck der Datenübertragung an. Der Wertebereich für Funktionen liegt bei: 0 bis 127. Werte größer oder gleich 128 signalisieren einen Fehler für die entsprechende Funktion. **Im Praktikum sollen Sie die benötigten Funktionen selbst definieren!**

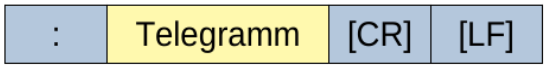
Daten: Enthält die zu übertragenden Informationen. Dieses Feld wird in der Regel unterteilt in Registern, Anzahl der zu übertragenden Registern und gegebenenfalls ausgelesene oder abzuspeichernde Informationen. Die Daten werden normalerweise als vielfaches von 16-Bit Registern übertragen. Die Telegrammgröße, inklusive der Start- und Endezeichen bei MODBUS ASCII, beträgt max. 256

Bytes/Zeichen. **Im Praktikum sollen Sie den Aufbau bzw. die Struktur der Daten und die daraus resultierende Telegrammgröße selbst festlegen (Empfehlung: ASCII-Darstellung der Daten)!**

CRC/LRC-Check: Je nach verwendeten MODBUS System (RTU oder ASCII), wird die Prüfsumme mittels CRC16 (**C**yclical **R**edundancy **C**hecking) für RTU oder LRC (**L**ongitudinal **R**edundancy **C**hecking) für ASCII ermittelt. Die Prüfsumme wird über alle Bytes eines Telegramms berechnet um Übertragungsfehler festzustellen. Ausgenommen von der Prüfsummenberechnung sind bei MODBUS ASCII die Start- und Endezeichen.

Start- und Endebedingungen

Gemäß MODBUS-RTU muss zwischen den Telegrammen eine Pause von mind. 3,5 Zeichen eingehalten werden. Ab einer Baudrate größer 19200 bps wird eine definierte Pause von 1750 µs festgelegt. Die Telegramme werden mithilfe der Pausen voneinander abgegrenzt.



Bei MODBUS ASCII werden die Telegramme mithilfe der Start- und Endezeichen voneinander abgegrenzt. Das Startzeichen ist ein Doppelpunkt ":" bzw. der Wert `0x3A`. Die Endezeichen sind CR (**C**arriage **R**eturn) mit dem Wert `0x0D` und LF (**L**ine **F**eed) mit dem Wert `0x0A`.

CRC- & LRC-Berechnung

Die CRC-Prüfsumme wird mithilfe einer CRC16-Maske (Standard = 0xA001) und einem Startwert (Standard = 0xFFFF) ermittelt. **Die CRC16-Maske und Startwert sind im Praktikum selbst festzulegen!**

Beispielfunktion für eine CRC16-Prüfsummenberechnung:

```
#define CRC16MASK 0xA001        // CRC16 Maske
#define CRC16VALUE 0xFFFF       // CRC16 Startvalue

short berechneCRC16(unsigned char* buffer, unsigned char size) {

    // Lokale Variablen
    short lsb, crc16 = CRC16VALUE;

    // Funktionsanweisungen
    for(unsigned char i = 0; i < size; i++) {
        crc16 = ((crc16^buffer[i]) | 0xFF00) & (crc16 | 0x00FF);
        for (unsigned char j = 0; j < 8; j++) {
            lsb=(crc16 & 0x0001);
            crc16=crc16/2;
            if(lsb) crc16 = crc16^CRC16MASK;
        }
```

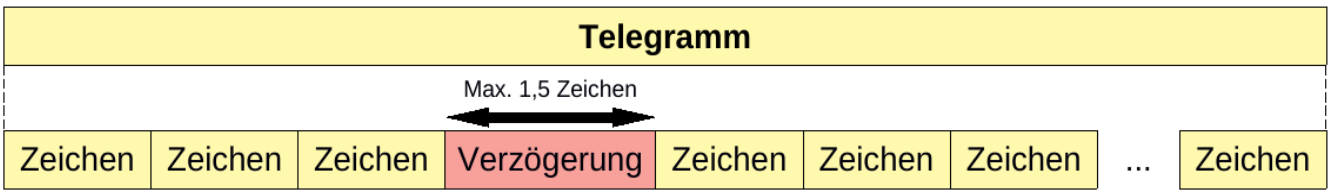
```
    }

    return crc16;
}
```

Bei der LRC-Prüfsummenberechnung werden die Hexadezimalwerte der einzelnen Zeichen aufsummiert. Im Anschluss erfolgt die Bildung des Zweierkomplemets. Das Ergebnis (1 Byte große Hexadezimalzahl) wird dann in zwei ASCII-Zeichen umgewandelt.

Beispielfunktion für eine LRC-Prüfsummenberechnung:

```
#define LRCVALUE 0x00       // CRC16 Startvalue

short berechneLRC(unsigned char* buffer, unsigned char size) {

    // Lokale Variablen
    unsigned char hByte, lByte, lrc = LRCVALUE;

    // Funktionsanweisungen
    for(unsigned char i = 0; i < size; i++) lrc += buffer[i];
    lrc = ~lrc+1;

    // Hex-Wert in ASCII umwandeln
    hByte = lrc >> 0x04;
    lByte = lrc & 0x0F;

    if(hByte >= 10) hByte += 0x37;
    else hByte += 0x30;

    if(lByte >= 10) lByte += 0x37;
    else lByte += 0x30;

    return (hByte << 8) | lByte;
}
```

Verzögerungszeiten und Timeout



Innerhalb eines Telegramms dürfen bei MODBUS die einzelnen Zeichen nicht mehr als 1,5 Zeichen Abstand aufweisen. Ab einer Baudrate größer 19200 bps wird eine definierte maximale Verzögerung von 750 µs festgelegt. **Wird die maximal festgelegte Verzögerung nicht eingehalten, wird das**

Datenpaket Verworfen! Legen Sie daher einen sinnvollen Wert für den Timeout fest, damit der Sender des Pakets auf das Verwerfen des Pakets beim Empfänger entsprechend reagieren kann.

Byte-Reihenfolge

Telegramm MODBUS RTU			
Geräte-Adresse	*Funktion*	*Daten*	*CR-Check*
1 Byte	1 Byte	n * Bytes	2 Bytes
			High-Byte / Low-Byte

Telegramm MODBUS ASCII					
Geräte-Adresse		*Funktion*		*Daten*	*LR-Check*
2 Zeichen		2 Zeichen		n * Zeichen	2 Zeichen
High-Byte	Low-Byte	High-Byte	Low-Byte		High-Byte / Low-Byte

Die Reihenfolge der Bytes erfolgt von High → Low oder von Low → High. **Im Praktikum sollen Sie selbst festlegen, in welcher Reihenfolge die Daten übertragen werden!**

Fehlerbehandlung

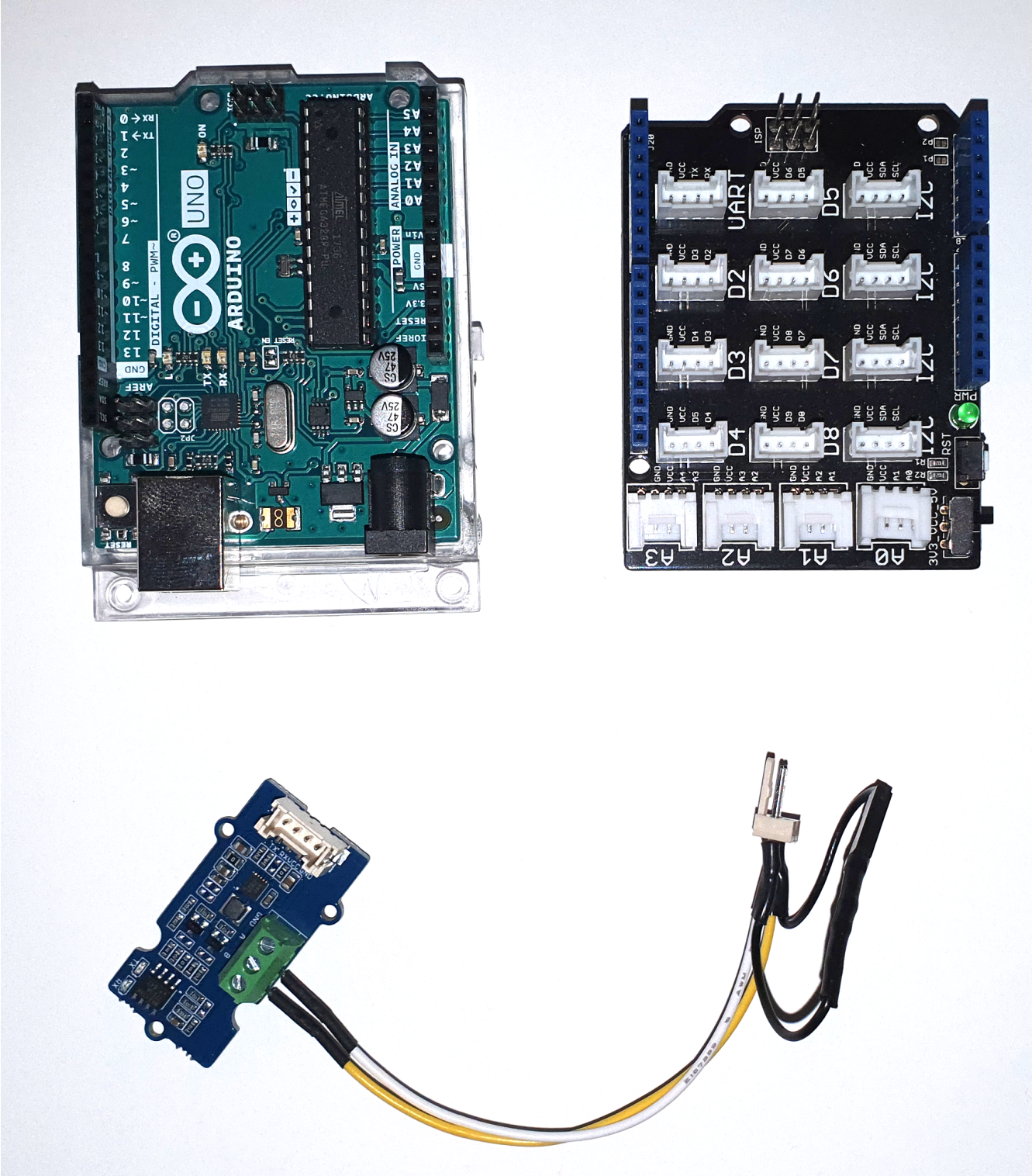
Wenn der Empfänger einer Nachricht einen Fehler feststellt, so sendet er eine entsprechende Fehlermeldung zurück an den Sender. Diese Fehlerrückmeldung kann auf verschiedene Art und Weise erfolgen. Im folgenden Beispiel wird bei MODBUS RTU das MSB im Funktionswert auf 1 gesetzt, was bei der Rückantwort dem Empfänger signalisiert, dass ein Fehler bei der entsprechenden Funktion aufgetreten ist. Im Datenfeld ist ein entsprechender Fehlercode für die Fehlerbeschreibung hinterlegt.

Telegramm			
Geräte-Adresse	*Funktion*	*Daten*	*CR-Check*
1 Byte	**Wert + 0x80** (1 Byte)	**Fehlercode**	2 Bytes

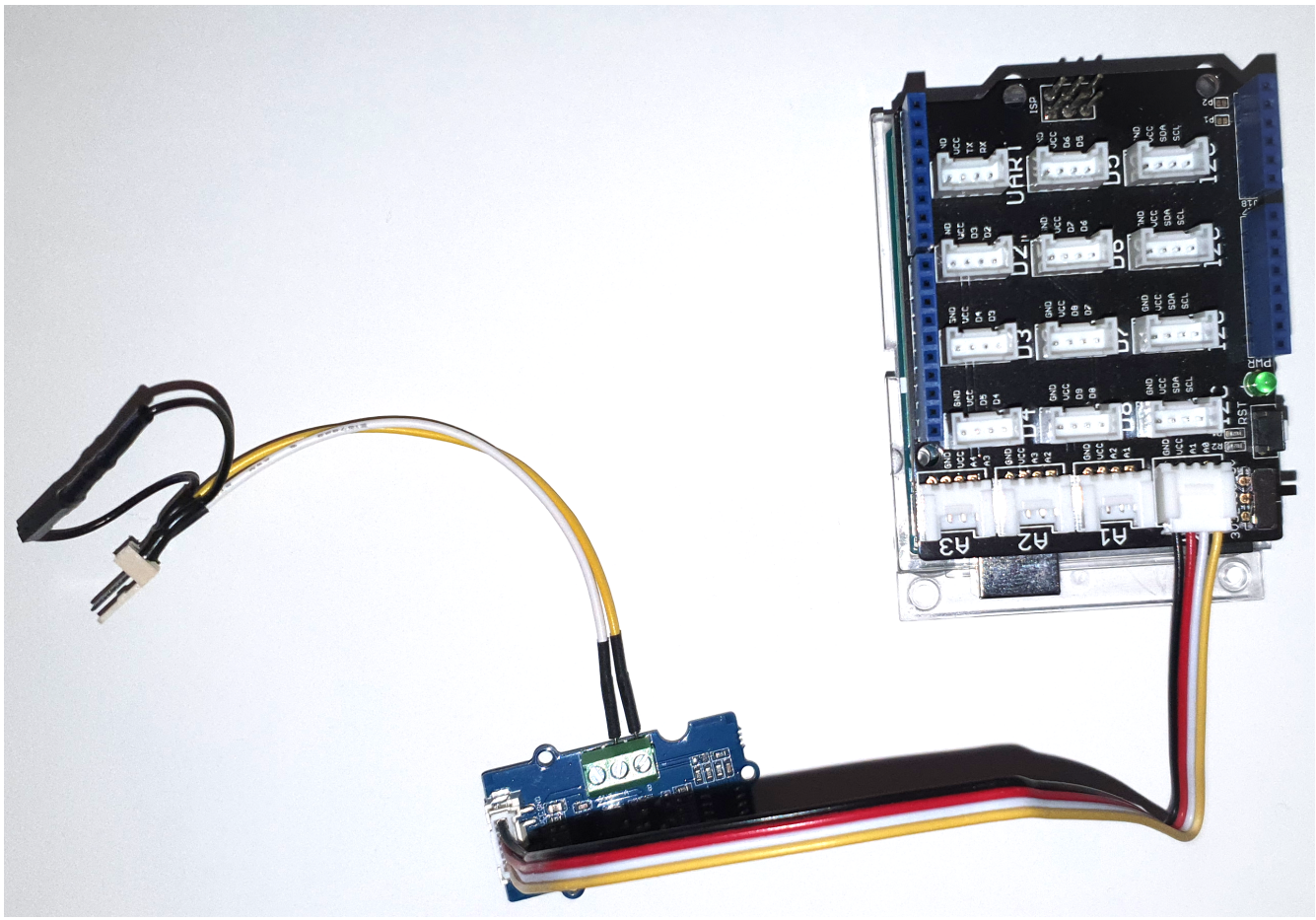
Im Praktikum sind eventuelle Fehlercodes bzw. das Verhalten des Protokolls bei einem Fehler, z. B. wenn eine unbekannte Funktion übertragen wird, festzulegen!

3. Versuchsaufbau

Für den Versuchsaufbau benötigen Sie folgende Module:



- Arduino Uno,
- Base-Shield,
- RS485-Empfänger.



Stecken Sie das Base-Shield auf den Arduino. Welchen Port Sie für den RS-485 Adaper verwenden, hängt von Ihrer Spezifiktion ab, die Sie im Laufe dieses Praktikums definieren. Beispielhaft wurde hier der Port *A0* gewählt.

Zum Schluss brauchen Sie nur noch das Buskabel mit dem RS-485 Adapter verbinden. Vergessen Sie dabei nicht, den Abschlusswiderstand (schwarzes Kabel) zu setzen, je nachdem, ob sie sich am Anfang oder am Ende des Bussystems befinden.

4. Aufgabenstellung

In diesem Praktikumsversuch wird eine RS-485 Verbindung zwischen zwei oder mehreren Arduino Uno aufgebaut. Im Anschluss sollen Sie ein eigenes Bus-Protokoll auf Basis von MODBUS-RTU oder MODBUS ASCII für einen Messenger entwerfen.

Das Protokoll ist ein Regelwerk für die Kommunikation auf dem RS485 Bus. Es bereitet die entsprechenden Textnachrichten auf und legt fest, wie und wohin die Telegramme gesendet werden. Die Kommunikation soll nach dem Master/Slave Prinzip erfolgen. Dabei soll es möglich sein, dass jeder Busteilnehmer sowohl im Master- als auch Slavebetrieb arbeiten kann.

Im Dokument MODBUS-Protokollrichtlinien sollen Ihre Richtlinien für das Protokoll definiert werden. Das Dokument ist mit den anderen Gruppen auszutauschen.

Folgende Punkte müssen in Ihrer Protokollrichtlinie berücksichtigt werden:

- Pin Konfiguraion für die RS485 Verbindung,
- Telegrammaufbau (MODBUS RTU oder MODBUS ASCII),
- Baudrate,
- Verzögerungszeiten und Timeouts
- maximale Telegrammgröße
- Prüfsummenwerte
- Geräteadressen
- Broadcastbefehl zur Weitergabe des Master-Tokens
- Fehlercodes für die Fehlerbehandlung
- Funktionen für die Übertragung
- Aufbau des Datenpakets
- Sonstige Festlegungen
- Verhalten des Bussystems in verschiedenen Situationen

5. Literaturverzeichnis

[1] Microchip
ATmega328 (Mikrocontroller) Datenblatt
http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega328PB-Automotive-Data-Sheet-40001980B.pdf
Abfragedatum: 15.04.2019

[2] Arduino
Language Reference
https://www.arduino.cc/en/Reference/HomePage
Abfragedatum: 23.10.2017

[3] MODBUS
MODBUS Application Protocol Specification
http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf
Abfragedatum: 06.04.2018

[4] MODBUS
MODBUS over Serial Line Specification and Implementation
http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf
Abfragedatum: 06.04.2018

[5] SeeedStudio
Grove - RS485
http://wiki.seeedstudio.com/Grove-RS485/
Abfragedatum: 06.04.2020

[6] Wikipedia
EIA-485
https://de.wikipedia.org/wiki/EIA-485
Abfragedatum: 02.04.2020

Hochschule Anhalt | Anhalt University of Applied Sciences | Fachbereich 6 EMW
Praktikum Mikrocomputertechnik für EIT, MT und BMT des 3. & 4. Semesters

Prof. Dr.-Ing. Ingo Chmielewski
Ingo.Chmielewski@HS-Anhalt.de

Tobias Müller, M. Eng.
Tobias.Mueller@HS-Anhalt.de;

Dipl. Ing. Harald Prütting
Harald.Pruetting@HS-Anhalt.de