|  |  |  |
| --- | --- | --- |
| **Übung Nr. 3**  Jahrgang: BHME20  Gruppe: 3D |  | **Protokollabgabe**  Solldatum: 16.11.2023  Istdatum:  Note: |

PROTOKOLL

**Thema: Serielle Schnittstelle & Optimierung der Aufgabe 10**

Tag: 9.11.2023

Zeit: 10:45-13:15 Uhr

Ort: HTBLA Kaindorf | PRR Labor

Anwesend: Traußnigg Jan

Abwesend: Ursnik Iwana

Schriftführer\*in: Traußnigg Jan

Betreuer: Dipl.-Ing. Steiner Walter

**Aufgabenstellung**

Die Serielle Schnittstelle soll gemeinsam mit dem Professor besprochen werden. Dann soll die Aufgabe 10 fertig programmiert & getestet werden und wenn die Zeit reicht, die Aufgabe 11 angefangen werden.

**Resümee**

Aus der heutigen Einheit konnten wir mitnehmen, wie serielle Schnittstellen funktionieren. Dadurch, dass wir die Aufgabe 10 gemeinsam nochmal genau durchgegangen sind, konnte tieferes Verständnis, wie man mehrere Prozesse auf einem Arduino gleichzeitig ablaufen lässt, gefunden werden.

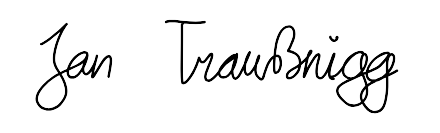
**Unterschriften**

Ein Bild, das Schwarz, Dunkelheit enthält.

Automatisch generierte Beschreibung

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Iwana Ursnik



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Jan Traußnigg

**Inhaltsverzeichnis**

[1 Zeitplan 2](#_Toc150666814)

[2 Thema 2](#_Toc150666815)

[2.1 Aufgabenstellung 2](#_Toc150666816)

[2.2 Verwendete Geräte und Hilfsmittel 2](#_Toc150666817)

[2.3 Vorgangsweise 3](#_Toc150666818)

[Feedback zum letzten Protokoll: 3](#_Toc150666819)

[Aufgabe 10) 1) Programmieren der main.cpp 3](#_Toc150666820)

[main.cpp 3](#_Toc150666821)

[3 Messergebnisse 4](#_Toc150666822)

# Zeitplan

10:45 – 11:45 Besprechung der seriellen Schnittstelle

11:45 – 12:10 gemeinsam mit den Kollegen Christopher & Peter nach Fehlern in deren Programm gesucht & Code verglichen

11:40 – 13:15 gemeinsam die Aufgabe 10 durchgedacht & kommentiert

# Thema

## Aufgabenstellung

* Besprechen der seriellen Schnittstelle mit dem Lehrer
* Prüfen des Codes, welcher in der letzten Einheit geschrieben wurde, auf dem Arduino
* Optimierung, Besprechung & Kommentierung des Codes
* Wenn die Zeit reicht, mit Aufgabe 11 anfangen

## Verwendete Geräte und Hilfsmittel

* Visual Studio Code (mit Plugin PlatformIO)
* Arduino IDE (privat-Laptop oder Schul-PC)
* Arduino Mega Shield

## Vorgangsweise

### Serielle Schnittstellen

Eine **serielle Schnittstelle** ist eine Verbindung zwischen zwei Geräten, bei der Daten nacheinander übertragen werden. Im Gegensatz zur parallelen Schnittstelle, bei der mehrere Bits gleichzeitig übertragen werden, erfolgt die **Datenübertragung** bei der seriellen Schnittstelle **bitweise**. Hier sind einige wichtige Punkte dazu:

**RS-232** (Recommended Standard 232): Dieser Standard wurde in den 1960er Jahren von der Electronic Industries Association (EIA) entwickelt und war lange Zeit in Computern weit verbreitet.

**Weitere Varianten:** Neben RS-232 gibt es viele andere serielle Schnittstellen, darunter RS-422, RS-485, Ethernet und CAN-Bus, die sich in Datenraten, Übertragungsdistanzen und anderen Eigenschaften unterscheiden.

**Vorteile**: Serielle Schnittstellen benötigen weniger Stromkreise und haben daher einen geringeren Verkabelungsaufwand, was besonders bei größeren Distanzen wichtig ist.

**Datenübertragung:** Bei der seriellen Datenübertragung werden einzelne Bits nacheinander gesendet. Die Übertragung kann in eine Richtung (simplex) oder in beide Richtungen (duplex) erfolgen.

**Übertragungsparameter:** Die Parameter für eine serielle Schnittstelle werden oft in der Form Baudrate-Datenbits-Parität-Stoppbits angegeben. Zum Beispiel bedeutet 9600, N, 8, 1:

* Baudrate: Die Baudrate gibt an, wie viele Bits pro Sekunde übertragen werden. In diesem Fall sind es 9600 Bits pro Sekunde.
* Datenbits: Es werden 8 Datenbits übertragen.
* Parität: Das “N” steht für “keine Parität”. Parität ist ein Fehlererkennungsmechanismus, der optional verwendet werden kann.
* Stoppbits: Es wird 1 Stoppbit verwendet, um das Ende eines Datenframes zu kennzeichnen.

Ein weiteres Beispiel könnte sein: 19200, E, 8, 2

1. Baudrate (19200): Die Baudrate gibt an, wie viele Bits pro Sekunde übertragen werden. In diesem Fall sind es 19200 Bits pro Sekunde.
2. Datenbits (8): Es werden 8 Datenbits übertragen. Diese repräsentieren die tatsächlichen Nutzdaten.
3. Parität (E): Das “E” steht für “Even Parity” (gerade Parität). Parität ist ein Fehlererkennungsmechanismus. Bei gerader Parität wird ein zusätzliches Paritätsbit hinzugefügt, sodass die Gesamtzahl der Einsen (1-Bits) im Datenframe (einschließlich Paritätsbit) immer gerade ist.
4. Stoppbits (2): Es werden 2 Stoppbits verwendet, um das Ende eines Datenframes zu kennzeichnen. Nach den 8 Datenbits und dem Paritätsbit folgen die 2 Stoppbits.

Zusammengefasst sieht ein 12-Bit-**Datenframe** mit den oben genannten Parametern folgendermaßen aus:

* Startbit: Ein logisches 0-Bit signalisiert den Beginn des Frames.
* Datenbits: 8 Bits für die tatsächlichen Daten.
* Paritätsbit: Ein zusätzliches Bit für die Paritätsprüfung (gerade Parität).
* Stoppbits: Zwei logische 1-Bits signalisieren das Ende des Frames.

# Messergebnisse

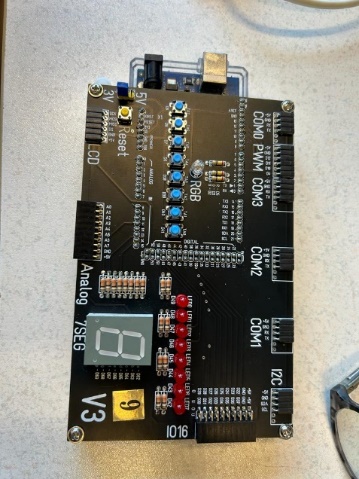


Abbildung 1: Arduino Mega Shield   
(Foto von Kollegin Anna-Aurora Schreiner)

Nach Testung des Programmes auf dem Arduino, konnten folgende Kenntnisse gezogen werden:

* Bei mehrmaligem Betätigen der „Anzeige aus, bis Button wieder gedrückt wird“-Taste wird der Counter rückgesetzt. Die Vermutung liegt, dass das mit dem default() in der switch-case zu tun haben könnte.
* Bei langem Halten der „Anzeige aus, bis Button wieder gedrückt wird“, wird die Anzeige nicht gestoppt. Dies muss genauer behandelt werden.

Wenn man jedoch keine dieser Benutzer Eingaben tätigt, funktioniert das Programm.