



ABSCHLUSSARBEIT LIXIE-UHR

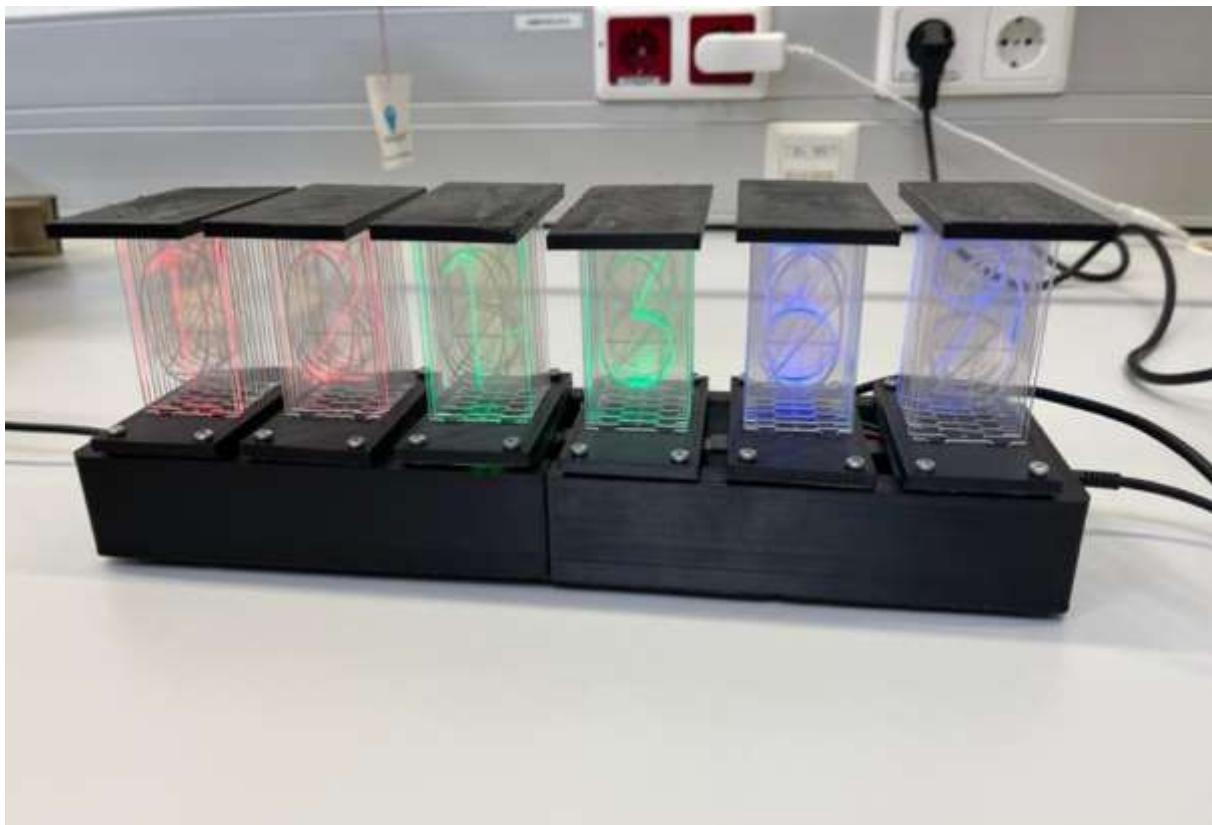


Abbildung 1: Lixie-Uhr

Ausgeführt im Schuljahr 2022/23 von:

Eren Akpinar

4AFEL

Betreuer/in:

Ing. Raffael Gächter

Samir El-Farfar

4AFEL

Ken Simon Höner

4AFEL

Rankweil, am 18.03.2023

Abgabevermerk:

AA original, am 28.03.2023

Ing. Raffael Gächter

AA digital, am 28.03.2023

AV Dipl.-Ing. Leopold Moosbrugger



1. Eidesstattliche Erklärung

Ich versichere an Eides statt, dass ich die entsprechend gekennzeichneten Teile der vorliegenden Abschlussarbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie alle wörtlich oder sinngemäß übernommenen Stellen in der Arbeit gekennzeichnet habe.

Ferner gestatte ich der Höheren Technischen Lehr- und Versuchsanstalt Rankweil (HTL), die vorliegende Abschlussarbeit unter Beachtung Datenschutz- und wettbewerbsrechtlicher Vorschriften für Lehre und Forschung zu benutzen.

1.1. Declaration of Oath

I declare by oath that all accordingly indicated parts of my final paper were independently written by myself, no other than the indicated sources and aids have been used and that all parts of the final paper which have been taken over, either literally or in a general manner, have been accordingly indicated. Furthermore I permit the Higher Technical College and Laboratory (Rankweil) (Höhere Technische Bundeslehr- und Versuchsanstalt Rankweil-HTL) to use the final paper for teaching and research, paying attention to data security and competition protection regulations.

Rankweil, am 18.03.2023

.....
Eren Akpinar

.....
Samir El-Farfar

.....
Ken Simon Höner



Inhalt

ABSCHLUSSARBEIT LIXIE-UHR	1
1. Eidesstattliche Erklärung	2
1.1. Declaration of Oath.....	2
1.2. VORWORT.....	7
1.3. ZIELSETZUNG	7
1.4. GEPLANTES ERGEBNIS.....	7
1.5. DANKSAGUNG	7
1.6. RECHTLICHE REGELUNG.....	8
2. ZUSAMMENFASSUNG.....	9
2.1. ABSTRACT.....	9
3. ABSCHLUSSARBEIT DOKUMENTATION	10
4. Individuelle Themenstellung im Rahmen des Gesamtprojekt.....	11
5. Meilensteine	12
6. Zeitplan.....	13
7. Der Besuch bei BLUM	18
8. Verwendete Programme (Hardware).....	20
9. Segmentplatine / Slave.....	21
9.1. Schaltplan.....	21
9.2. Schaltungsbeschreibung	21
9.3. Layout (Top)	22
9.4. Layout (Bottom)	22
9.5. Bohrplan	23
9.6. Bestückungsplan (Top).....	24
9.7. Bestückungsplan (Bottom)	24
10. Mainboard / Master.....	25
10.1. Schaltplan	25
10.2. Schaltungsbeschreibung:	26
10.2.1. Mikrocontroller:	26
10.2.2. USB-/UART Wandler:	26
10.2.3. Stützkondensatoren:.....	26
10.2.4. Zusätzliche Stifteleisten:	26
10.2.5. Berechnungen:	26



10.3	Layout (Top).....	27
10.4	Layout (Bottom).....	27
10.5	Bohrplan.....	28
10.6	Bestückungsplan	29
10.7	Stückliste.....	30
11	Gehäuse-Pläne.....	31
11.1	Unterseite:.....	31
11.2	Einsteckplatte:.....	33
11.3	Deckel:	34
11.4	3D-Ansicht von der Uhr	35
12	3D-Druck (Gehäuse).....	36
12.1	Verwendeter 3D-Drucker.....	36
12.2	Gedruckte 3D-Werkstücke	36
13	Gehäuse (Plexigläser)	37
14	Bestückung der Platinen.....	47
15	DCF-77-Empfangsmodul	49
15.1	Funktionsweise	49
15.2	DCF77-Dekoder	49
16	Messprotokoll.....	50
16.1	Mainboard	50
16.2	Segmentplatine	50
16.2.1	Betrieb einer LED mit niedrigster Intensität	50
16.2.2	Eine LED mit höchster Intensität	51
16.2.3	Ein Segment mit niedrigster Intensität	51
16.2.4	Ein Segment mit höchster Intensität	51
16.2.5	Alle Segmente im Betrieb mit niedrigster Intensität	52
16.2.6	Alle Segmente im Betrieb mit höchster Intensität	52
16.3	DCF77-Empfängermodul	53
17	APA102c-LED Funktionsanalyse.....	54
17.1	Demoprogramm-Quelltext	55
17.2	Aufbau mit APA102c	56
17.2.1	Blockschaltbild:	56
18	Software.....	57



18.1	Projektverwaltung / Verwendete Programme	57
18.2	Projektkonzept / Programmkonzept	59
18.3	Programmbeschreibung	60
18.4	Entwicklungsablauf / Codebeschreibung.....	61
18.5	Hauptprogramm	62
18.5.1	Struktogramm.....	62
18.5.2	Code.....	63
18.5.3	Beschreibung.....	64
18.6	Interrupt Service Routine.....	65
18.6.1	Struktogramm	65
18.6.2	Code	66
18.6.3	Beschreibung.....	66
18.7	Timer Setup / Initialisierung.....	67
18.7.1	Struktogramm	67
18.7.2	Code	68
18.7.3	Beschreibung.....	68
18.8	Zeitverarbeitung	69
18.8.1	Struktogramm	69
18.8.2	Code	69
18.8.3	Beschreibung.....	69
18.9	Datentransfer	70
18.9.1	Struktogramm	70
18.9.2	Code	71
18.9.3	Beschreibung.....	72
18.10	LED-Zeit.....	73
18.10.1	Struktogramm	73
18.10.2	Code	73
18.10.3	Beschreibung.....	73
18.11	SPI-Initialisierung	74
18.11.1	Struktogramm	74
18.11.2	Code	74
18.11.3	Beschreibung.....	74
18.12	SPI-Übertragung	75



18.12.1	Struktogramm	75
18.12.2	Code	75
18.12.3	Beschreibung.....	75
19	Testaufbauten:.....	76
19.1	Testaufbau (01.02.2023)	76
19.2	Testaufbau (03.02.2023)	76
19.3	Testaufbau/Inbetriebnahme der ersten Segmentplatine (22.02.2023)	77
19.4	Testaufbau (22.03.2023)	77
19.5	Testaufbau (03.03.2023)	78
19.6	Zwischenstand (08.03.2023)	78
19.7	Der endgültige Aufbau (17.03.2023)	79
20	Datenblätter	80
20.1	Apa102c:	80
20.2	FT232RL (USB-/UART Wandler):	85
20.3	Attiny1606:	90
20.4.2.	Allgemein.....	90
20.4.3.	SPI-Datasheets.....	95
20.4.4.	USART-Datasheets	104
20.5.	DCF-77-Empfangsmodul:.....	110
21.	Abbildungsverzeichnis	114
22.	Quellenverzeichnis	115



1.2. VORWORT

Ziel war es, eine Nixie-Uhr umzusetzen. Die hohen Spannungen, die zum Betrieb benötigt werden, stellen jedoch ein großes Problem dar. Ebenso der große Energieverbrauch. Daher soll eine energiesparende Alternative mit niedriger Spannung entwickelt werden.

1.3. ZIELSETZUNG

Ziel des Projekts ist die Entwicklung einer voll funktionsfähigen Lixie-Uhr, die mit LEDs betrieben wird. Die Ziffern der Uhr werden dabei über Plexigläser in die Ziffern eingefräst sind dargestellt. Die Steuerung der Uhr wird über ein Tastenfeld bewerkstelligt. Optional lässt sich auch die Farbe der LEDs anpassen sowie die Uhr mithilfe eines DCF77-Empfängers einstellen.

1.4. GEPLANTES ERGEBNIS

Eine kostengünstige Alternative zur konventionellen Nixie-Uhr, die zugleich günstig und energiesparend ist und mit niedriger Betriebsspannung und LEDs funktioniert.

1.5. DANKSAGUNG

Zuallererst möchten wir uns bei unserem Projektbetreuer Raffael Gächter herzlich bedanken, dafür, dass er uns während der Entwicklungsphase der Lixie-Uhr unterstützt und bei Problemen immer ein offenes Ohr hatte und uns bei der Lösungsfindung unterstützt hat. Zudem möchten wir uns bei dem Unternehmen Blum GmbH beim Herrn Michael Baumann für das Fräsen der Plexigläser und bei Stefan Zudrell-Koch, der uns ermöglicht hat, weitere Platinen zu bestellen bedanken.



1.6. RECHTLICHE REGELUNG



Erklärung

Die unterfertigten Kandidaten/Kandidatinnen haben gemäß § 34 Abs. 3 Z 1 und § 37 Abs. 2 Z 2 des Schulunterrichtsgesetzes in Verbindung mit den Bestimmungen der „Prüfungsordnung BMHS, Bildungsanstalten“, BGBl. II Nr. 177/2012 i.d.g.F. die Ausarbeitung einer Abschlussarbeit mit folgender Aufgabenstellung gewählt:

Lixieuhr - LED Nixie Uhr (Gesamtprojekt)

Individuelle Aufgabenstellungen im Rahmen des Gesamtprojektes:

- Eren AKPINAR (4AFEL): **Leiterplattenentwicklung, Protokollierung, Mechanik**
- Ken Simon HÖNER (4AFEL): **Schaltplan, Dimensionierung, Bestückung, Protokollierung**
- Samir EL-FARFAR (4AFEL): **Programmierung, Dokumentation, Protokollierung**

Die Kandidaten/Kandidatinnen nehmen zur Kenntnis, dass die Abschlussarbeit in eigenständiger Weise und außerhalb des Unterrichtes zu bearbeiten und anzufertigen ist, wobei Ergebnisse des Unterrichtes mit einbezogen werden können, die jedenfalls als solche entsprechend kenntlich zu machen sind.

Die Abgabe der vollständigen Abschlussarbeit hat in digitaler und in zweifach ausgedruckter Form bis spätestens **31.03.2023** beim zuständigen Betreuer/der zuständigen Betreuerin zu erfolgen.

Die Kandidaten/Kandidatinnen nehmen weiters zur Kenntnis, dass ein Abbruch der Abschlussarbeit nicht möglich ist.

Kandidaten/Kandidatinnen:

Eren AKPINAR (4AFEL)

Ken Simon HÖNER (4AFEL)

Samir EL-FARFAR (4AFEL)

Datum und Unterschrift bzw. Handysignatur:

28.09.2022

Eren A.

28.09.2022

Ken Simon Höner

28.09.2022

Samir El-Farfarr



2. ZUSAMMENFASSUNG

Das Ziel des Abschlussprojekts ist, eine Lixie-Uhr zu entwickeln. Die Uhr, bestehend aus jeweils sechs Segmenten, die Zahlen von 0 bis 9 anzeigen soll. Die Segmente werden in drei verschiedene Untergruppen eingeteilt. Die erste Gruppe zeigt die Stunden, die zweite Gruppe die Minuten und die dritte Gruppe die Sekunden an. Es gibt für jedes Segment eine eigene Platine, die mit 20 LEDs bestückt ist. Die LEDs sorgen dafür, dass die Zahl, die ins Plexiglas eingraviert wurde, angezeigt wird. Für die Uhr ist ein entsprechendes Programm zu erstellen.

Für die Umsetzung wurden auf die fachtheoretischen und praktischen Grundlagen, die in der Fachschule erarbeitet werden, zurückgegriffen.

2.1. ABSTRACT

The goal of the project is to develop a Lixie-Watch. The clock consists of six segments, each intended to display the numbers from 0 to 9. The segments are divided into three subgroups. The groups consist of hours, minutes and seconds. Each segment has its own board, which is equipped with 20 LEDs. The LEDs ensure that the digit engraved in the acrylic glass is displayed. A corresponding program must then be created for the clock.

Things who were taught in the school were implemented in the project.



3. ABSCHLUSSARBEIT DOKUMENTATION

Name der Verfasser	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
Jahrgang Schuljahr	4AFEL 2022/2023
THEMA der Abschlussarbeit	Lixie-Uhr
Kooperationspartner	Julius BLUM GmbH



4. Individuelle Themenstellung im Rahmen des Gesamtprojekt

Projektleiter

Akpınar Eren



Individuelle Themenstellung:

Leiterplattenentwicklung, Protokollierung, Konzeption, Schaltplan/Layout, Gehäuse, Materialmanagement, Dokumentation, Planung

Projektmitglied

El-Farfar Samir



Projektmitglied

Höner Ken



Individuelle Themenstellung:

Schaltplan/Layout, Dimensionierung, Bestückung, Protokollierung, Konzeption, Gehäuse, Materialmanagement, Dokumentation

Projektbetreuer: Raffael Gächter



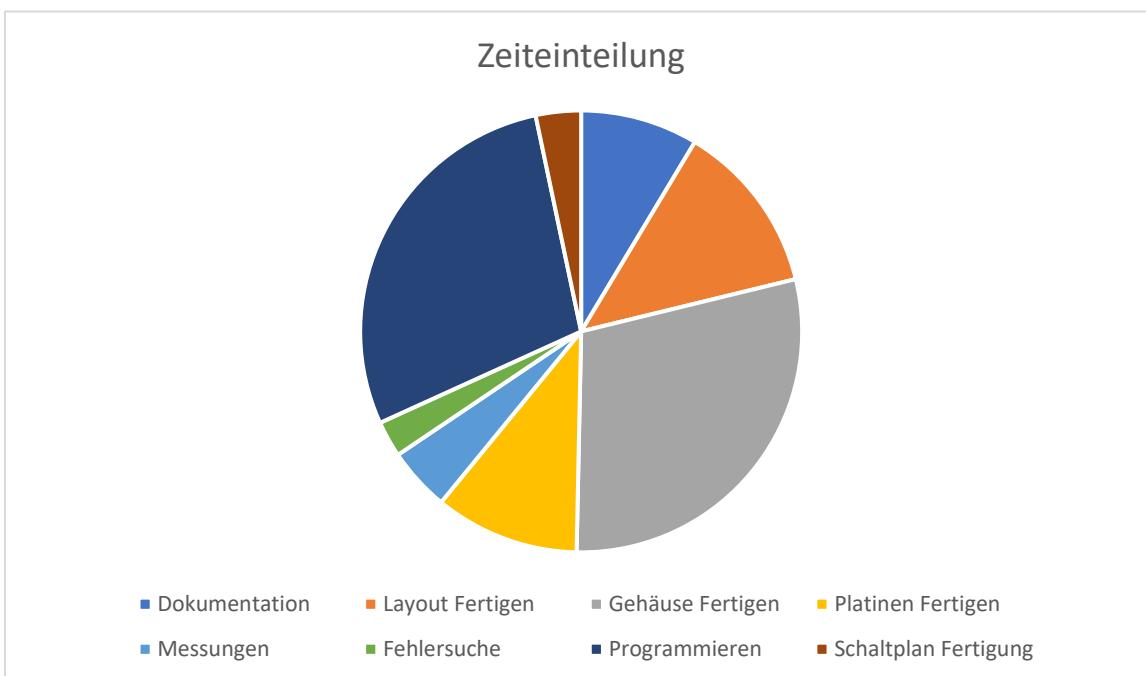


5. Meilensteine

Konzeption:	09.11.2022	Funktion/Protokoll:	19.03.2023
	<ul style="list-style-type: none">- Spezifikation- Design- Kosten		<ul style="list-style-type: none">- Funktionstest aller Komponenten- Durchführen von Messungen- Protokollierung
Schaltplan/Layout:	04.12.2022	Dokumentation:	02.04.2023
	<ul style="list-style-type: none">- Schema- ERC-Check- Layout- DRC-Check- Inspektion- Gerber Files- Bohrplan- Stückliste		<ul style="list-style-type: none">- Übersicht- Mechanik- Leiterplatte- Messtechnik- Protokollierung
Gehäuse:	04.12.2022		
	<ul style="list-style-type: none">- Planung- Fertigung		
Materialbestellung:	25.12.2022		
	<ul style="list-style-type: none">- Leiterplatte- Bauteile- Gehäuse- Sonstiges		
Vorentwicklung Software:	01.01.2023		
	<ul style="list-style-type: none">- Softwarekonzept- Struktogramm/ Programmablaufplan- Bibliotheken- Vorentwurf (Alpha)		
Fertigen der Uhr:	26.02.2023		
	<ul style="list-style-type: none">- Leiterplatte- Gehäuse- Sonstiges		
Fertigstellen der Software:	15.03.2023		
	<ul style="list-style-type: none">- Testprogramm- Beta- Release		



6. Zeitplan





Abschlussprojekt: Lixie-Uhr Zeitplan

Datum	Dauer	Tätigkeit	Student
10.09.2022	5h	Informationsbeschaffung für die Abschlussarbeit und Ideenaustausch	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
12.09.2022	4h	Erstellung der Zusammenfassung für die Abschlussarbeit	Eren Akpinar, Ken Simon Höner, Samir El-Farfar
12.09.2022	3h	Erstellung Schaltplan Segmentplatine	Ken Simon Höner, Eren Akpinar
12.09.2022	3h	Erstellung Layout Segmentplatine	Ken Simon Höner, Eren Akpinar
27.10.2022	1h	Programmvorbereitung	Samir El-Farfar
27.10.2022	1h	Erstellung Zeitplan	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
31.10.2022	3h	Gehäusekonzept erstellen	Eren Akpinar
31.10.2022	2h	Programm-Test	Samir El-Farfar
08.11.2022	1h	Kostenliste / Stückliste erstellt	Eren Akpinar, Samir El-Farfar
08.11.2022	2h	"Entwicklung eines Buck-Converter (Layout / Schaltplan)"	Eren Akpinar, Ken Simon Höner
09.11.2022	3h	Überarbeitung des Konzepts	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
09.11.2022	2h	Erstellung eines neuen Schaltplans/Layout, weil die ws2812-LED durch die APA102c ersetzt wurde (Folgen: andere Bauform -> neues Layout)	Eren Akpinar, Ken Simon Höner
09.11.2022	5h	Überarbeitung des ganzen Projekts und Erstellung einer neuen Stückliste	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
23.11.2022	2h	Das Segment Layout überarbeitet	Eren Akpinar
23.11.2022	2h	Gehäuse bearbeitet	Eren Akpinar
30.11.2022	4h	Erstellung einer Library und eines Programms für die Ansteuerung von LEDs (Apa102c)	Eren Akpinar
30.11.2022	1h	Erstellung einer Zwischenplatte für die Befestigung von Plexigläsern	Eren Akpinar
02.12.2022	1h 30min	Master Layout erstellt und Segment-Layout--> Fehler ausgebessert	Eren Akpinar
04.12.2022	2h	Master Layout fertiggestellt	Ken Simon Höner
09.12.2022	2h	Die ersten Libraries wurden programmiert	Samir El-Farfar



10.12.2022	1h	Konzept eines Frames wurde erstellt (nicht funktionsfähig)	Samir El-Farfar
12.12.2022	1h 30min	Gehäuse (Unterseite) für die Platinen angepasst	Eren Akpinar
12.12.2022	2h	Layouts überarbeiten --> Fehler ausbessern	Eren Akpinar, Ken Simon Höner
13.12.2022	3h	Bauteile für die Bestellliste raussuchen und inklusiv Links hinzufügen	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
13.12.2022	2h	Weitere Versuche für den Datenframe (nicht funktionsfähig)	Samir El-Farfar
13.12.2022	1h	SPI (Atmega16) und LED Library hinzugefügt	Samir El-Farfar
14.12.2022	30min	Gerber-Files erstellen	Eren Akpinar, Ken Simon Höner
14.12.2022	4h	Das erste Programmkonzept für die Megacard wurde entwickelt	Samir El-Farfar
14.12.2022	2h	Struktogramme wurden erstellt	Samir El-Farfar
14.12.2022	1h	UART-Interface für das Einstellen der Uhr wurde erstellt	Samir El-Farfar
19.12.2022	30min	Master (Kondensatoren mit Elkos ersetzen)	Eren Akpinar
08.01.2023	2h	Gehäuse (Unterseite) für den 3D-Druck vorbereiten und das erste Konzept gedruckt	Eren Akpinar
10.01.2023	30min	Gehäuse (Das Loch für USB-Port vergrößern und Fehler beheben)	Eren Akpinar
31.01.2023	4h	Erste Portierung für den Attiny1606 wurde durchgeführt	Samir El-Farfar
01.02.2023	5h	An der Portierung vom Atmega16 Programm für den Attiny1606 wurde weitergearbeitet	Samir El-Farfar
01.02.2023	4h	SPI wurde für den Attiny1606 funktionsfähig programmiert	Samir El-Farfar
01.02.2023	2h	Struktogramme wurden für das Programm des Attiny1606 erstellt	Samir El-Farfar
01.02.2023	2h	Mit der Dokumentation angefangen	Eren Akpinar, Ken Simon Höner, Samir El-Farfar
01.02.2023	3h	Test (Plexiglas für den Test besorgt und zugeschnitten + Muster eingraviert, um nachzuschauen wie das Licht die Muster anzeigt)	Eren Akpinar, Ken Simon Höner
03.02.2023	2h	Testversuch 2 --> Plexiglas fräsen lassen und geschaut, ob die Muster angezeigt werden	Eren Akpinar, Ken Simon Höner



05.02.2023	3h	Erstellung einer Zwischenplatte / Einstechplatte / Deckel mit den richtigen Maßen für die Plexigläser	Eren Akpinar
05.02.2023	4h	Das Design für jedes Plexiglas erstellt + eine endgültige Baugruppe mit allen Komponenten erstellt	Eren Akpinar
05.02.2023	3h	Gehäuse Optimierung (Größen geändert)	Eren Akpinar
07.02.2023	2h	Dokumentation weiter gearbeitet	Eren Akpinar, Ken Simon Höner, Samir El-Farfar
08.02.2023	3h	An der Softwaredokumentation wurde weitergearbeitet	Samir El-Farfar
16.02.2023	2h	Blum Planung der Vorgehensweise für Plexigläser	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
17.02.2023	4h	Endgültige Fräsumgebung von Plexigläsern bei BLUM	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
22.02.2023	2h	Materialbeschaffung für die Leiterplatten	Eren Akpinar, Ken Simon Höner
22.02.2023	6h	Bestückung von 2 Segmentplatinen und Prüfung	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
22.02.2023	1h	Das Programm wurde durchkommentiert	Samir El-Farfar
23.02.2023	2h	Das erste Segment wurden mithilfe einer Testsoftware erstmals getestet	Samir El-Farfar
24.02.2023	4h	Bestückung der restlichen Segmentplatinen	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
24.02.2023	1h 30min	Funktionstest von Segmentplatinen mit Beta-Programm	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
24.02.2023	4h	Firmware wurde aktualisiert	Samir El-Farfar
01.03.2023	4h	Bestückung der Masterplatine und Funktionstest	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
01.03.2023	3h	Die Uhrenverarbeitung wurde verbessert	Samir El-Farfar
03.03.2023	3h	Masterplatine mit den Segmentplatinen getestet und mit SPI zum Laufen gebracht	Eren Akpinar, Samir El-Farfar, Ken Simon Höner



08.03.2023	3h	Fehlersuche Masterplatine	Eren Akpinar, Ken Simon Höner
08.03.2023	2h	Fehlerbehebung im Layout von Master und Durchführung der Bestellung	Eren Akpinar, Samir El-Farfar, Ken Simon Höner
15.03.2023	7h	Dokumentation weiter gearbeitet	Eren Akpinar, Ken Simon Höner
15.03.2023	1h	Gehäuse bearbeitet	Eren Akpinar, Ken Simon Höner
15.03.2023	1h	Photoshop Bilder bearbeitet	Ken Simon Höner
15.03.2023	3h	UART wurde getestet und die Firmware aktualisiert	Samir El-Farfar
17.03.2023	1.5h	Master fertigstellen	Ken Simon Höner
17.03.2023	0.5h	Gehäuse bearbeitet	Eren Akpinar, Ken Simon Höner
17.03.2023	4h	Dokumentation fortsetzen	Eren Akpinar, Ken Simon Höner, Samir El-Farfar
18.03.2023	3h	Dokumentation fertiggestellt	Eren Akpinar, Ken Simon Höner, Samir El-Farfar

Schüler	Stunden
Eren Akpinar	118,5h
Samir El-Farfar	106.5h
Ken Simon Höner	94h
Gesamte Stunden	319h



7. Der Besuch bei BLUM



Abbildung 2: Gruppenbild bei Blum

Für das Zuschneiden und Fräsen des Plexiglases wurden wir von der Firma Blum unterstützt. Michael Baumann hat uns bei der Umsetzung sehr unterstützt. Er hat uns gezeigt, wie man eine SolidWorks-Zeichnung in ein Programm umschreiben kann, damit es für die Maschine lesbar ist und verarbeitet werden kann.

Die CNC-Fräse hat 3 Achsen, die sich in X, Y und Z-Achsen bewegen können. Beim Fräsen wird das Plexiglas mithilfe eines Vakuums festgehalten, um es auf Position zu halten. Während dem Fräsvorgang wird durch Düsen ein Wasser-Öl Gemisch zugeführt, damit der Fräskopf nicht zu heiß wird und das Plexiglas schmilzt.

Nach dem Fräsvorgang werden unscharfe Kanten entgratet.

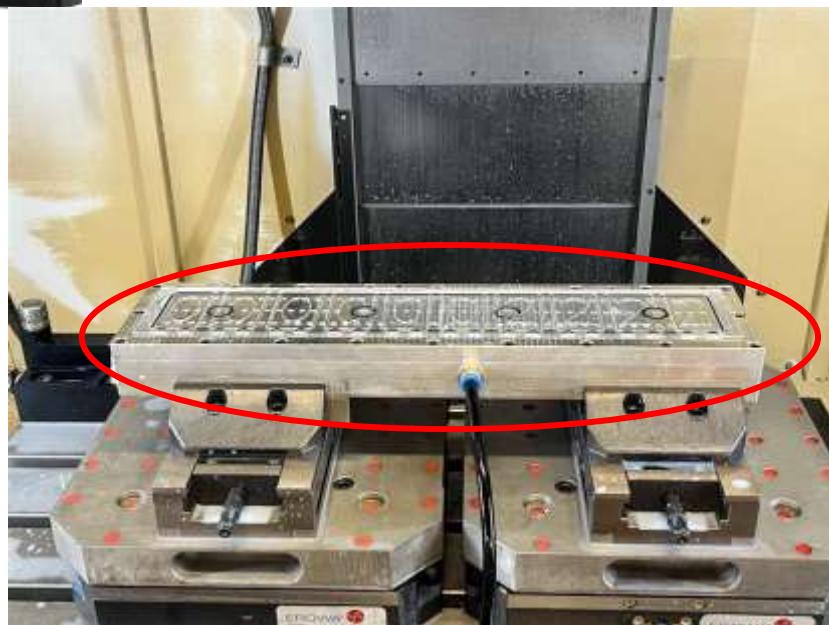


Abbildung 3: Fräsmaschine

Das Plexiglas wird mit Vakuum befestigt. Es war nicht möglich, die Platte mithilfe eines Schraubstockes zu spannen, da dadurch zu viel Last auf der Platte lasten würde und sie sich so verformen würde.

Da wegen dem Vakuum die einzelnen Gläser nicht komplett durchgefräst werden können, wurden sie am Schluss ausgebrochen.



Abbildung 4: Fräsmaschine

Während dem Fräsvorgang wird ein Gemisch aus Öl und Wasser von Düsen auf den Bohrer gespritzt, um ihn zu kühlen. Das Gemisch wird unten in der Maschine gesammelt, abgepumpt, gefiltert und wiederverwertet.



8. Verwendete Programme (Hardware)



Abbildung 5: EAGLE Logo (EAGLE, 2023)

Für das Erstellen von Schaltplänen und Layouts wurde das Programm EAGLE verwendet.



Abbildung 6: KiCad Logo (KiCad, 2023)

Das Programm KiCad hat die gleiche Funktion wie EAGLE. Nur mit dem Unterschied, dass das Programm Open Source ist. Ein großer Vorteil gegenüber EAGLE ist die 3D-Ansicht von Leiterplatten. Dadurch lässt sich die Bauteilplatzierung bereits vor dem Fertigen der Leiterplatte darstellen.

SOLIDWORKS wurde für das Erstellen der Gehäusepläne verwendet. Die Software verwendet parametrische Konstruktionsprinzipien und erstellt drei miteinander verbundene Dateien: Objekte, Baugruppen und Zeichnungen.



Abbildung 7: SOLIDWORKS Logo (SolidWorks, 2023)

Das Dremel 3D Digilab Slicer ist ein Programm, indem CAD-Dateien ohne Internetverbindung gesliced werden können. Um das Werkstück zu slicen, muss man die SolidWorks-Datei auf *.stl umändern. Anschließend kann man mit dem Slicer die GCODE-Datei erstellen und mit dem 3D-Druck starten.



Abbildung 8: Dremel Digilab 3D Slicer (Dremel, 2023)



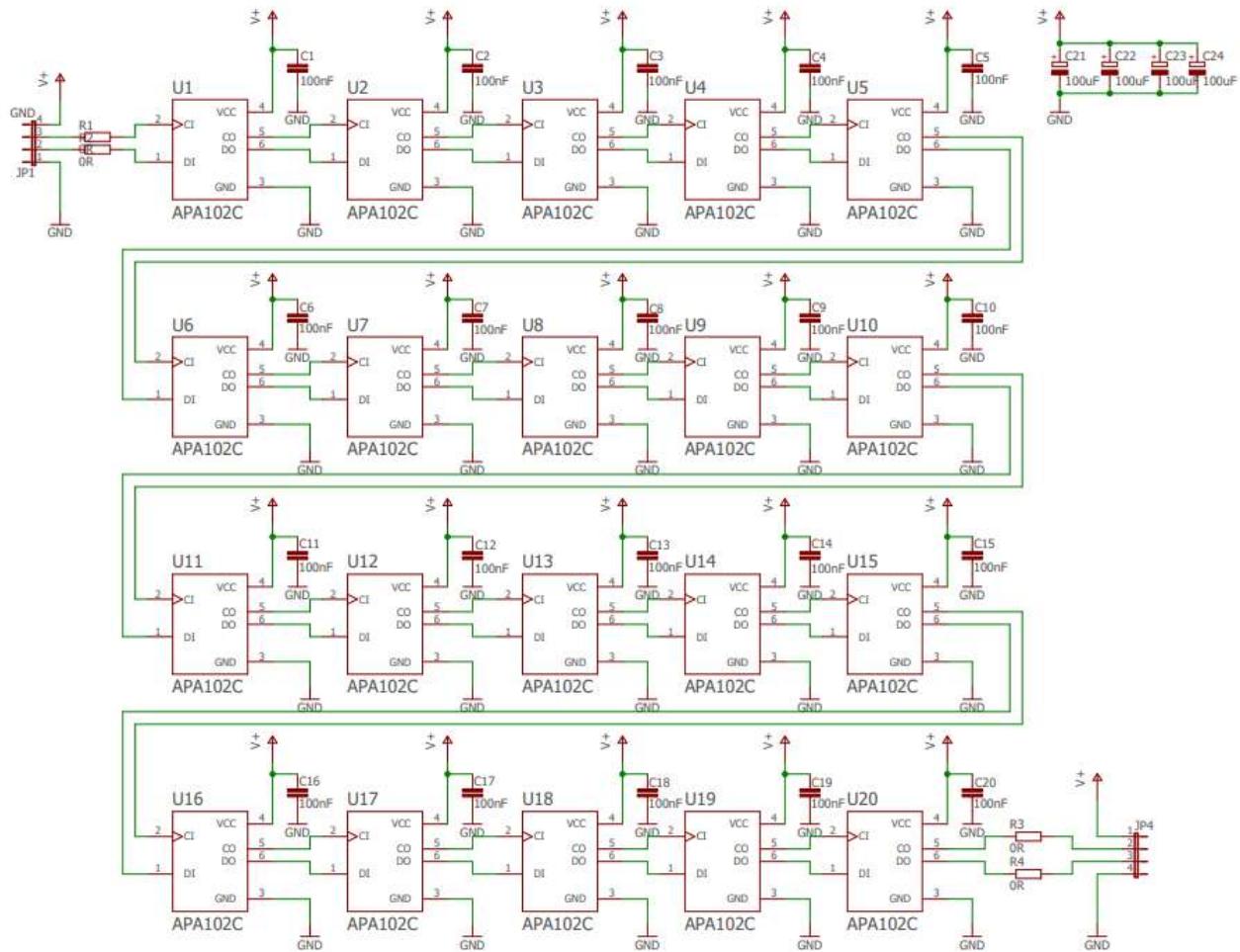
Mit Fritzing lassen sich einfache Schaltungen und Prototypen aufbauen. Dabei gibt es sehr viele Bauteile, die für den Aufbau von Schaltungen verwendet werden können.

Abbildung 9: Fritzing Logo (Fritzing, 2023)



9. Segmentplatine / Slave

9.1. Schaltplan



9.2. Schaltungsbeschreibung

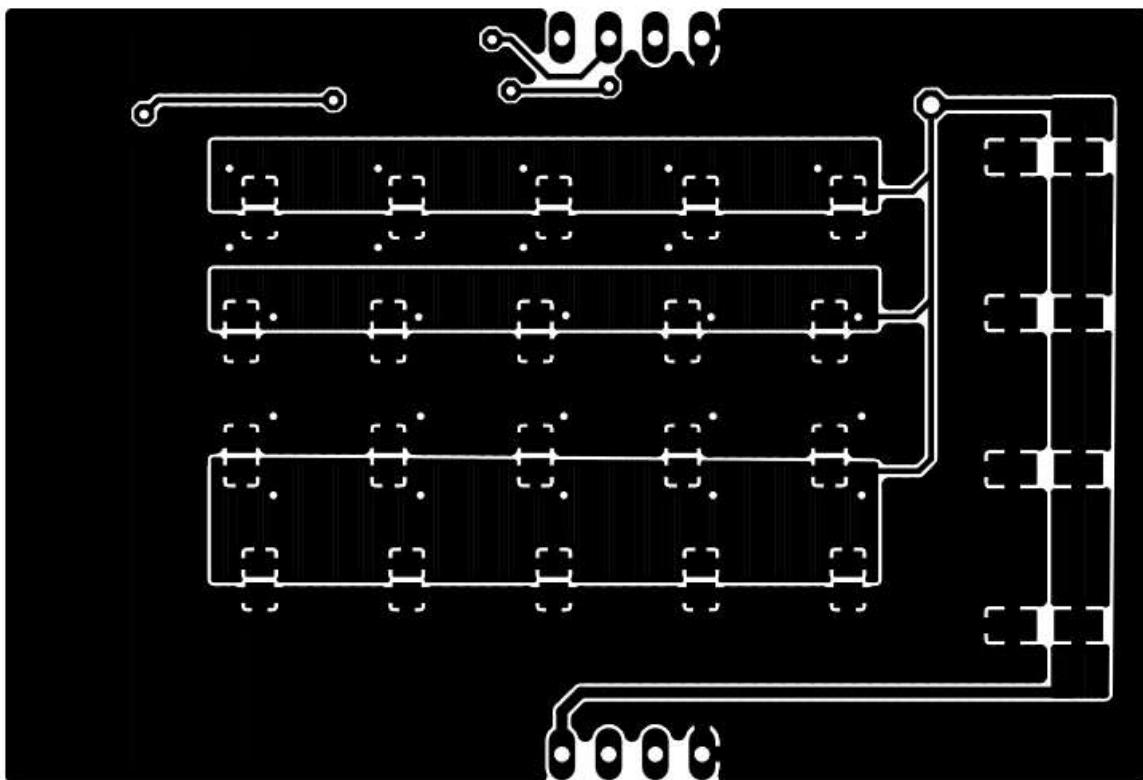
Die Schaltung beinhaltet 20 LEDs, die in Reihe geschaltet sind.

Die 0 Ohm Widerstände ermöglichen eine elektrische Trennung zwischen den Teilschaltungen. Im Falle eines Kurzschlusses wird dadurch die Zerstörung der Leiterplatte verhindert.

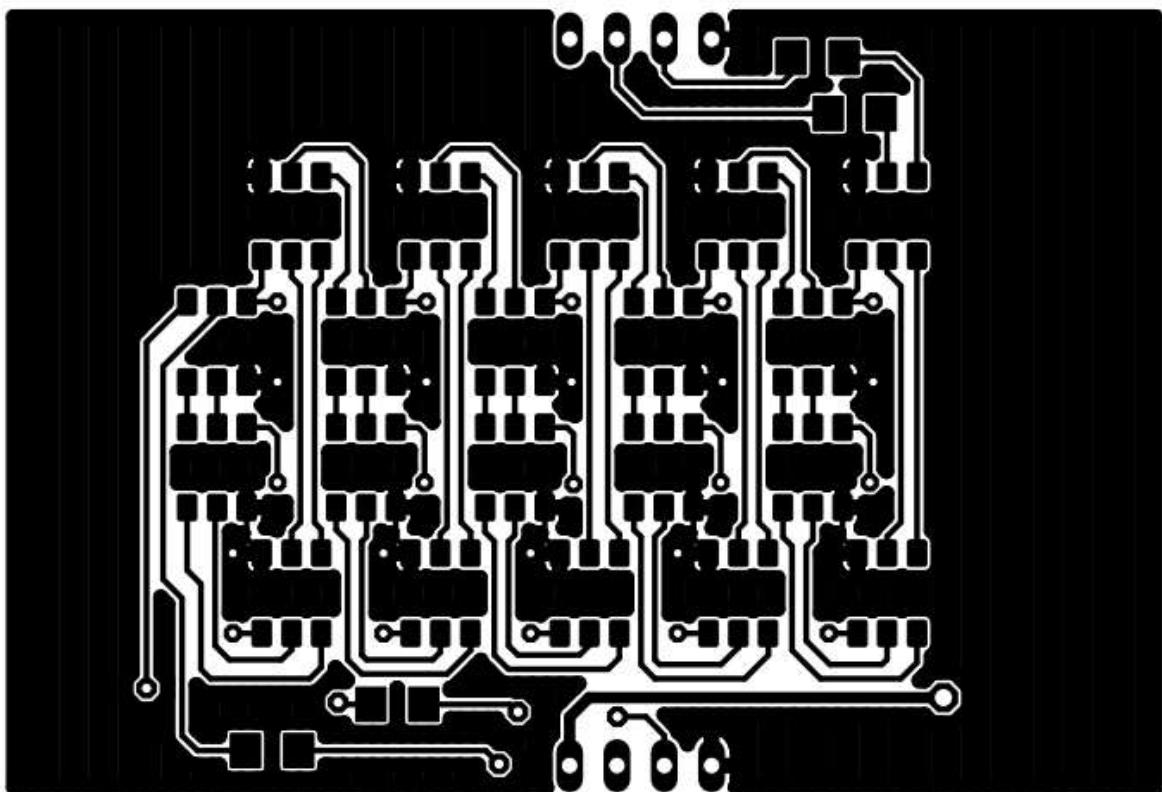
Zur Versorgungsspannung werden mehrere Elektrolyt-Kondensatoren parallelgeschaltet. Diese verhindern den Einbruch der Spannung bei hoher Leistungsaufnahme durch die LEDs und verhindern so einen Absturz der SPI-Schnittstelle.



9.3. Layout (Top)

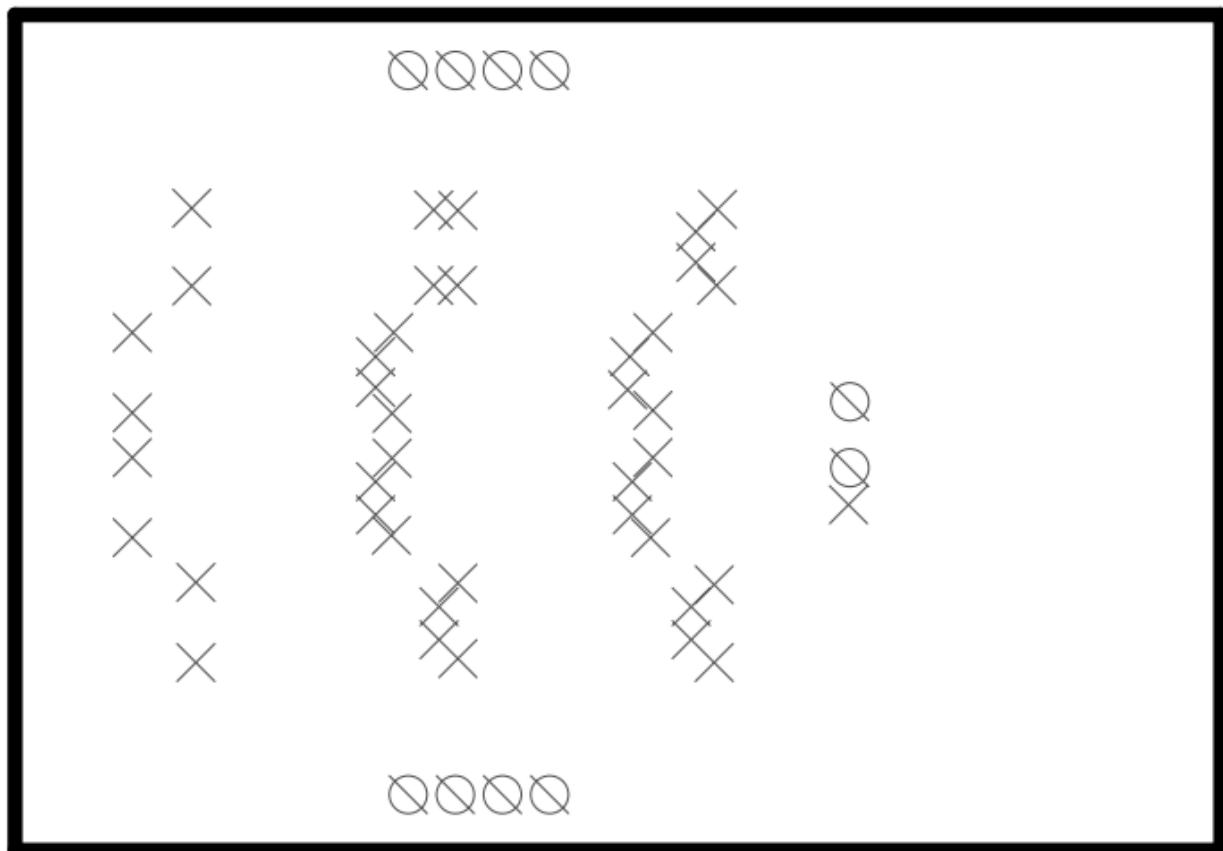


9.4. Layout (Bottom)





9.5. Bohrplan



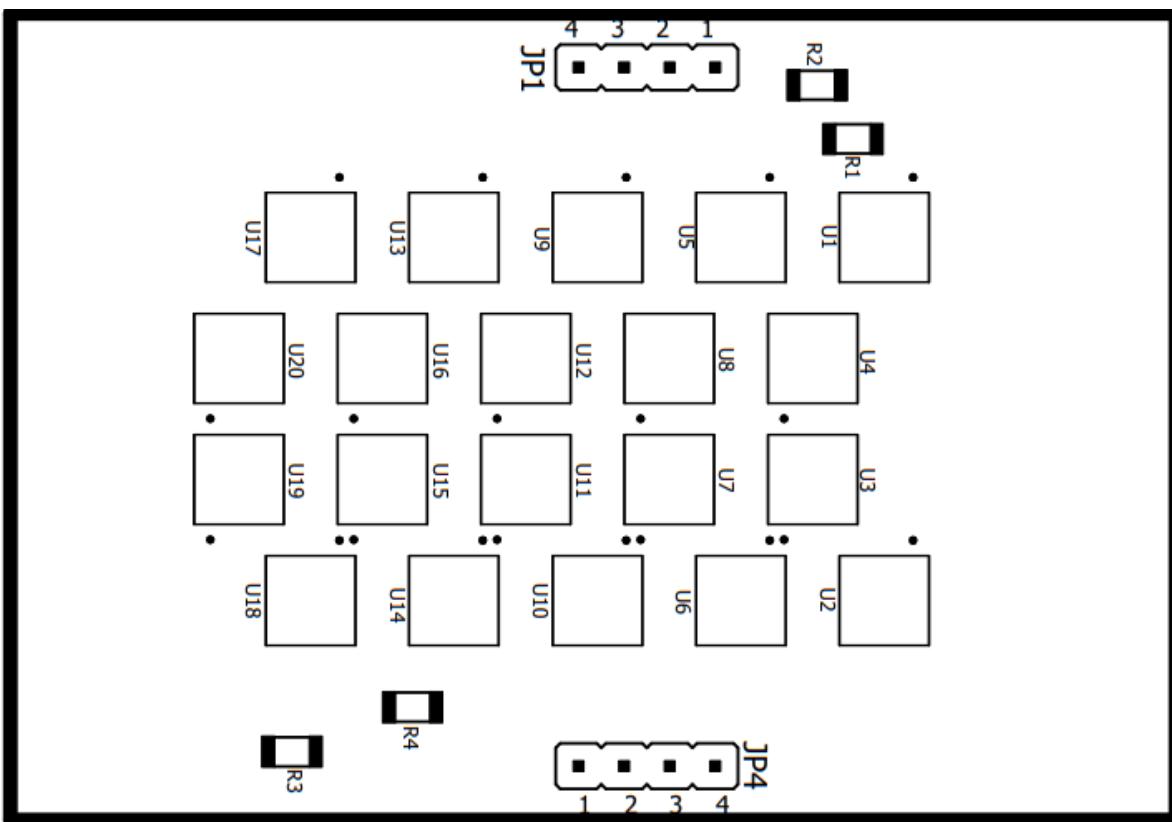
Material: FR4 1,5mm 35u Cu

Toleranz: Aussenmass -0,5mm, sonst +/_0,2mm

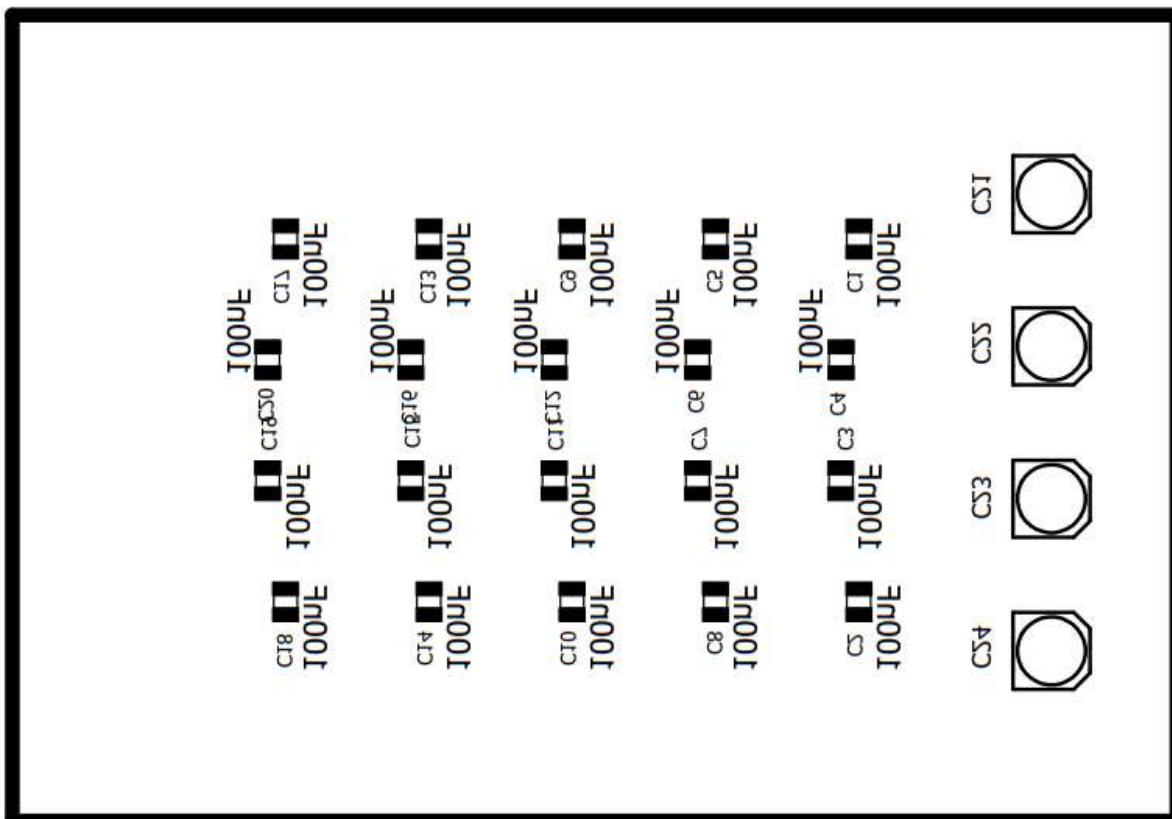
Bohrplan		
SYM	SIZE	CNT
×	0.60 mm	41
Ø	0.80 mm	2
Ø	0.90 mm	8
TOTAL		51



9.6. Bestückungsplan (Top)



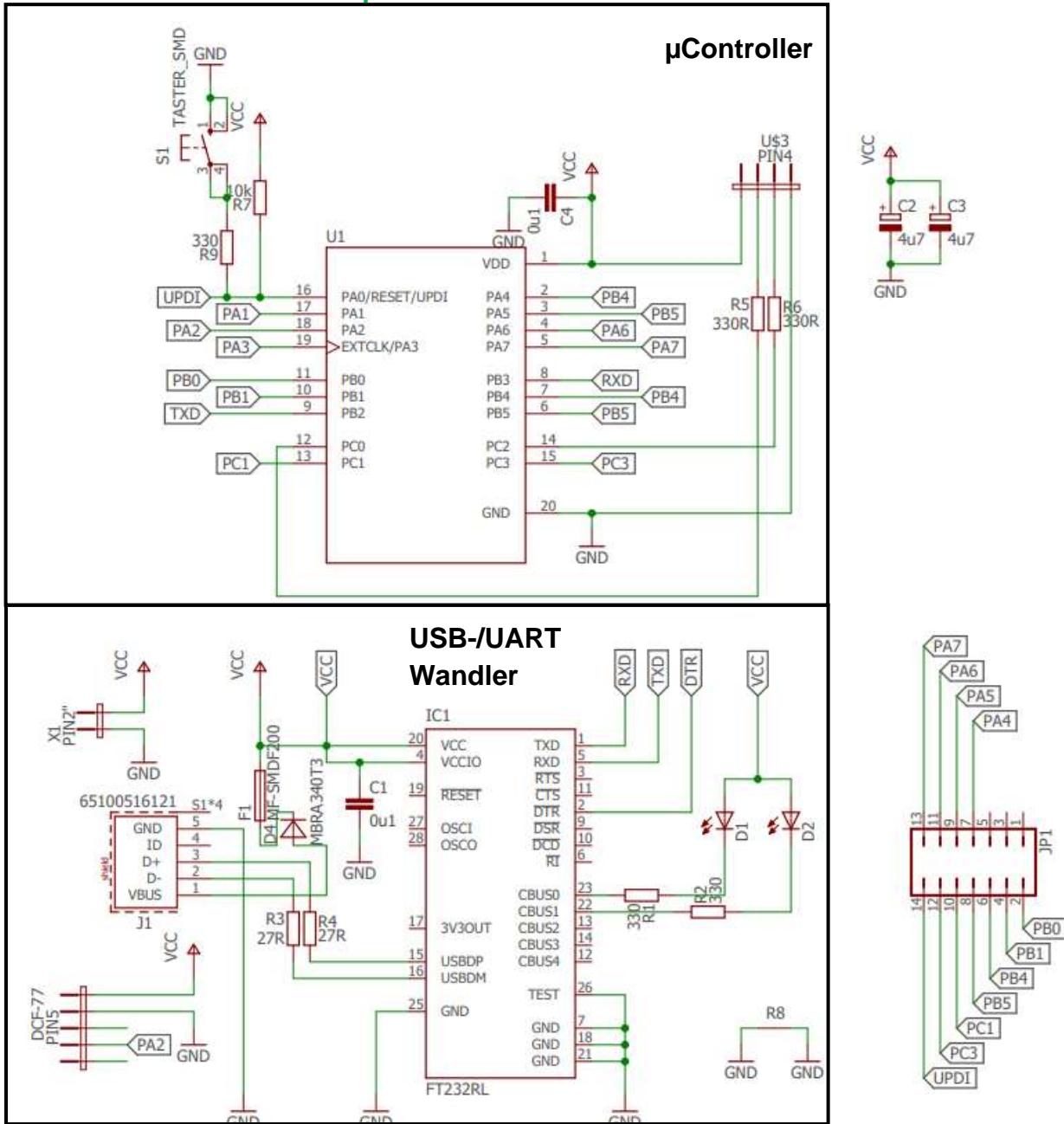
9.7. Bestückungsplan (Bottom)





10. Mainboard / Master

10.1. Schaltplan





10.2. Schaltungsbeschreibung:

10.2.1. Mikrocontroller:

Als Mikrocontroller wurde der Attiny-1606 verwendet. Der Attiny1606 ist ein Mikrocontroller, der den 8-Bit-AVR-Prozessor mit einem Hardware-Multiplikator verwendet. Er ist mit einer Taktfrequenz von 20 MHz und 16KB Flash ausgestattet.

Weitere Informationen zum Mikrocontroller können dem Datenblatt entnommen werden.

10.2.2. USB-/UART Wandler:

Ein UART-Wandler dient zur Schnittstellenwandlung von USB auf konventionelles UART. Der Attiny1606 ist werkseitig bereits mit einem synchronen/asynchronen Receiver/Transmitter ausgerüstet.

Der Mikrocontroller verfügt über keinen USB-Anschluss. Eine solche Schnittstelle wird installiert, damit eine serielle Verbindung mit einem PC hergestellt werden kann. Das benutzte Bauteil ist ein sogenannter USB/UART – Converter (FT232RL).

Die USB-Empfänger-Zelle unterstützt den USB 1.1 & USB 2.0 Schnittstelle.

Der Wandler bietet einige Funktionen mehr → siehe Datenblatt

10.2.3. Stützkondensatoren:

Zur Versorgungsspannung werden zwei Elektrolyt-Kondensatoren parallelgeschaltet. Diese dienen dem Lastausgleich bei hohen Leistungen und verhindern somit das Unterschreiten der min. zulässigen Spannung mit der der uController versorgt werden kann.

10.2.4. Zusätzliche Stifteleisten:

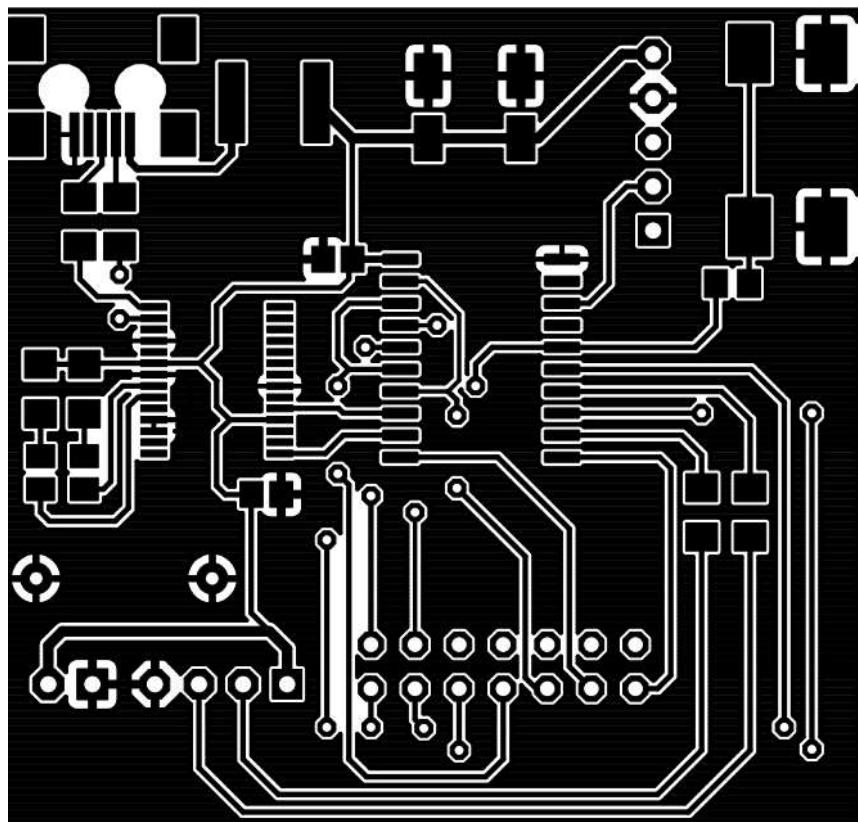
Die Verwendung der Stifteleisten dient dem Zugriff auf die nicht verwendeten Pins.

10.2.5. Berechnungen:

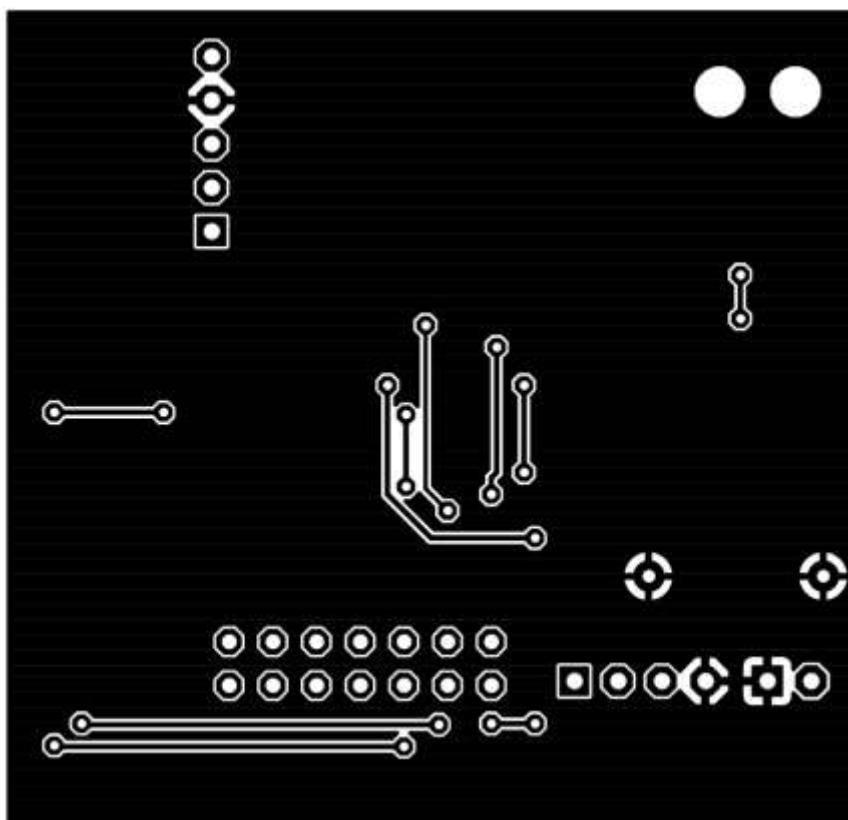
Die Bauteilwerte wurden dem Datenblatt entnommen. Dementsprechend wurde keine Berechnung der Bauteile durchgeführt.



10.3 Layout (Top)

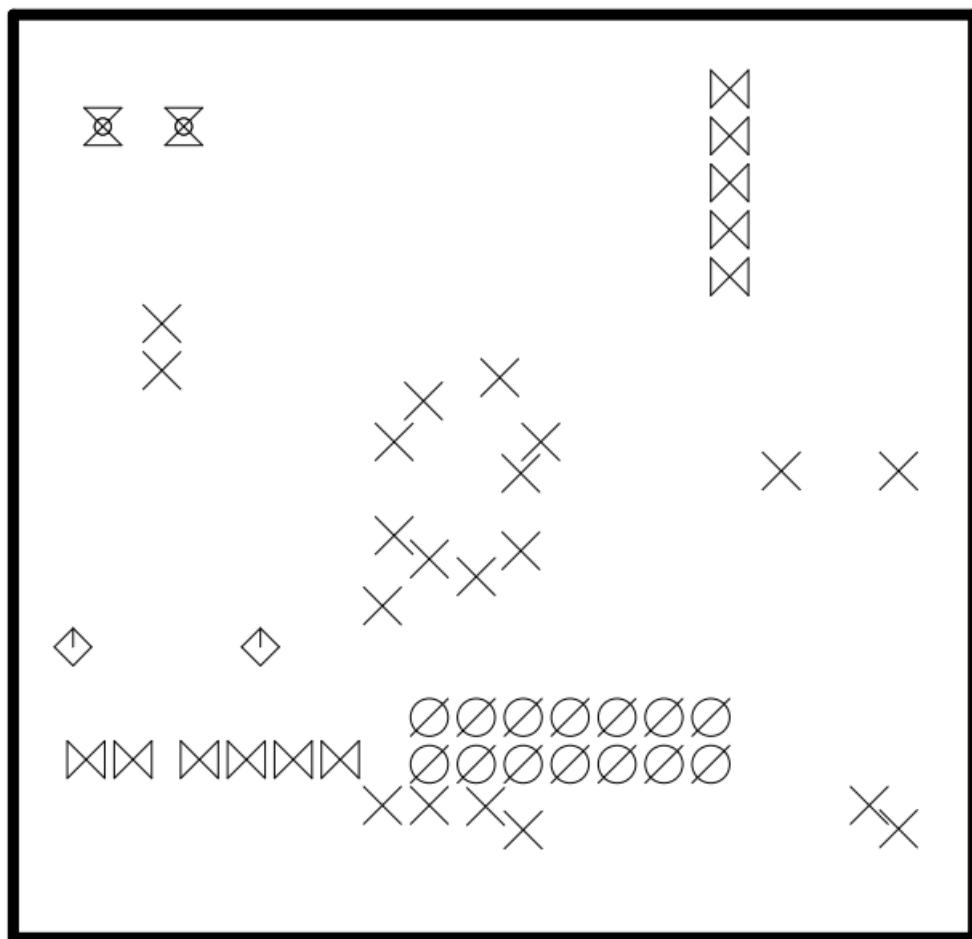


10.4 Layout (Bottom)





10.5 Bohrplan



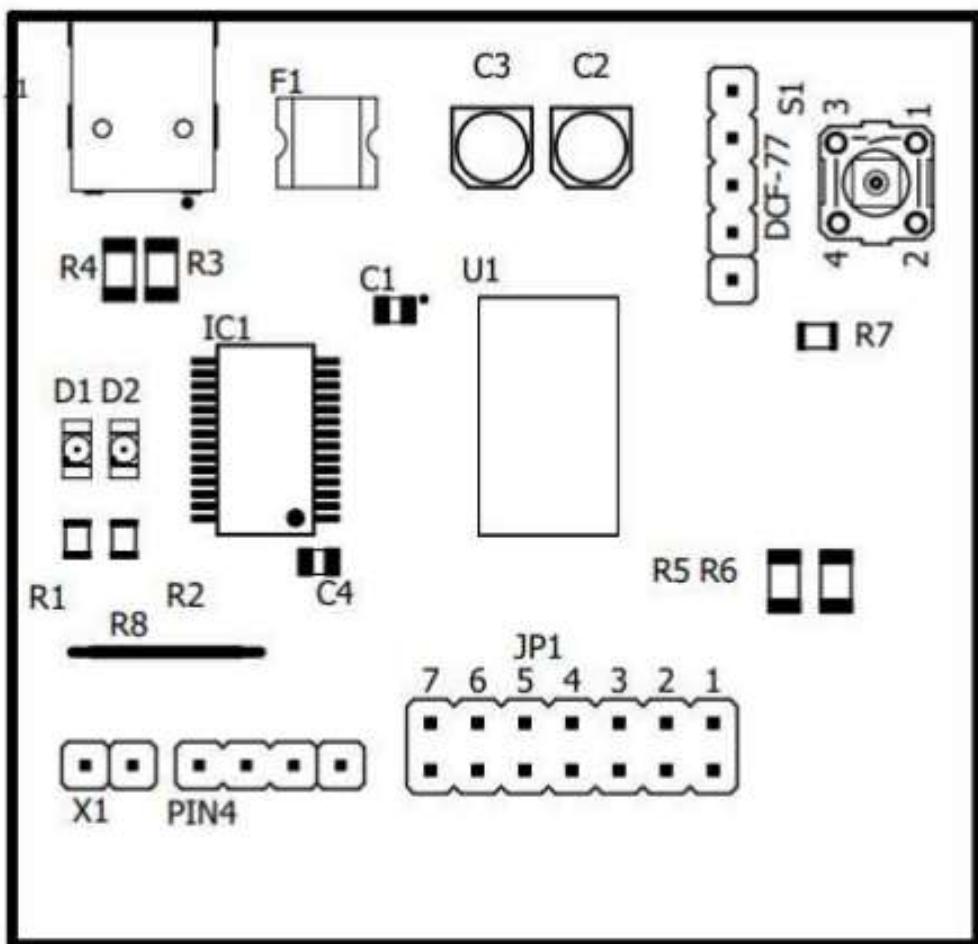
Material: FR4 1,5mm 35u Cu

Toleranz: Aussenmass -0,5mm, sonst +/_0,2mm

Bohrplan		
SYM	SIZE	CNT
×	0.60 mm	20
◇	0.80 mm	2
☒	0.90 mm	2
∅	0.90 mm	14
▣	1.00 mm	11
TOTAL		49



10.6 Bestückungsplan





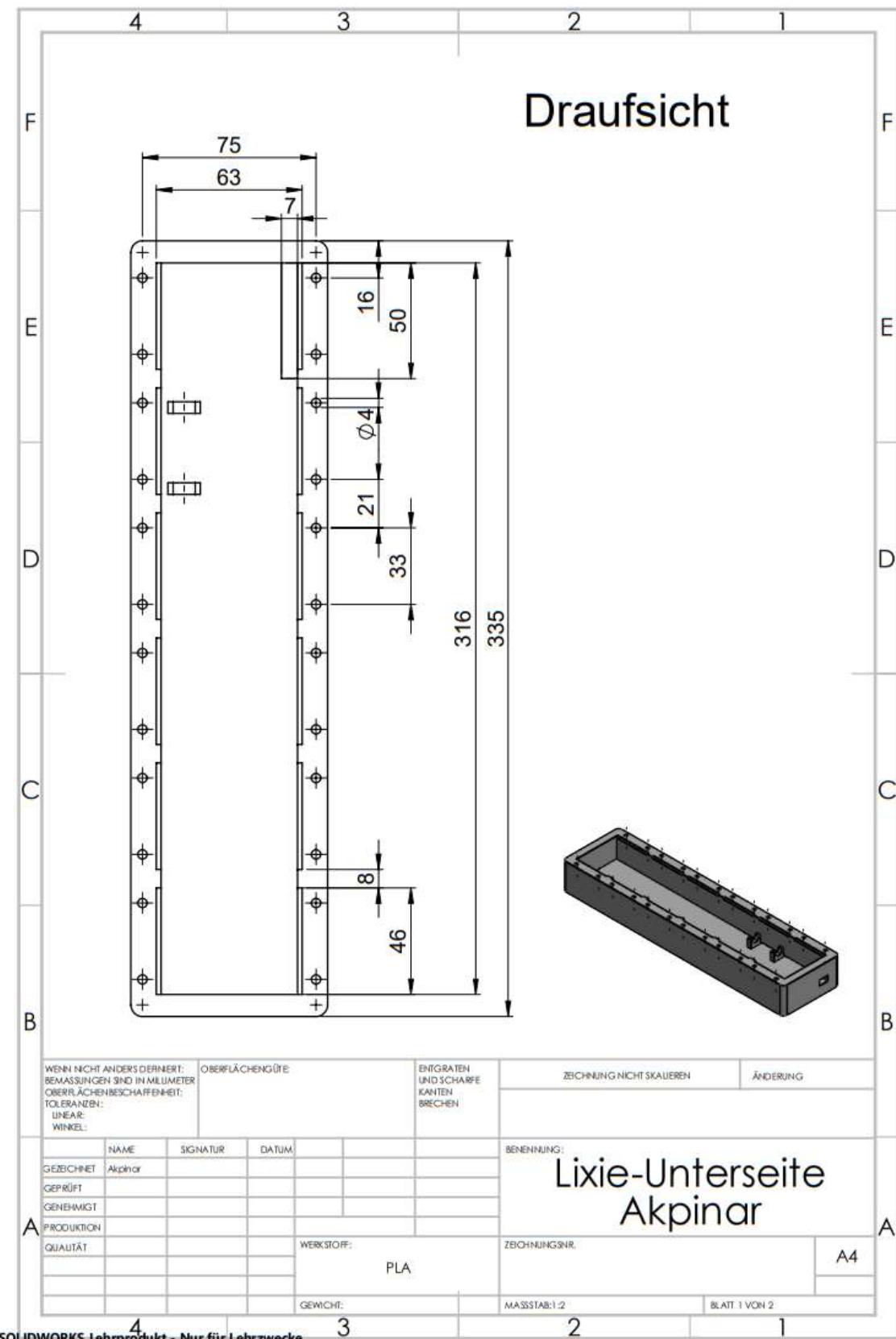
10.7 Stückliste

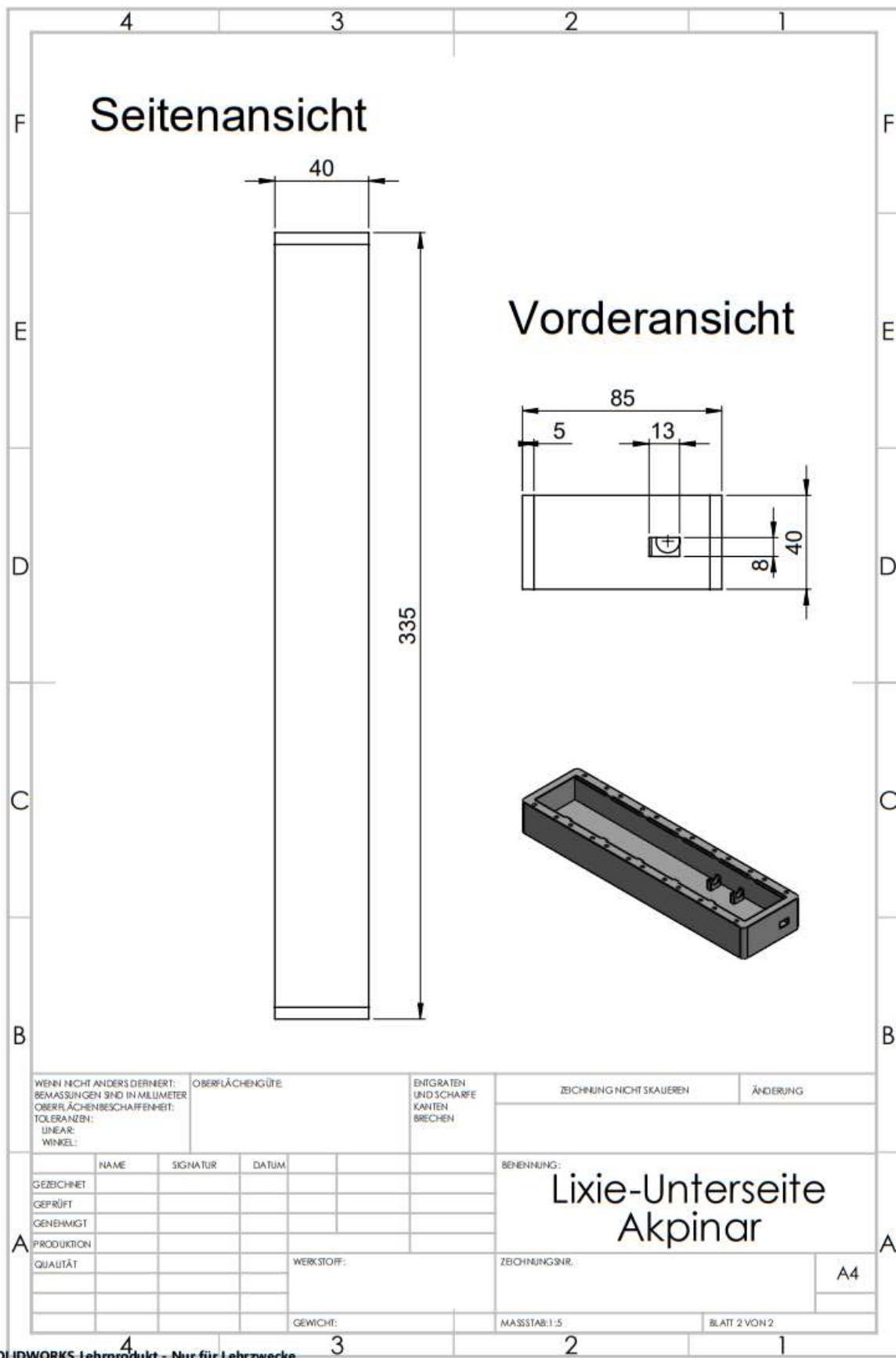
	Menge	Wert	Bauform	Bauteil	Bestellort	Stückpreis / €
Segmentplatine	200	APA102C	SON160P500X500X140-6N	LED RGB	DIGIKEY	0,580
	200	100n	C0805	C1 ... C20	Reichelt	0,009
	40	100uF	C0805	C21 ... C24	DIGIKEY	0,392
	50	OR	R1206	R1 ... R4	HTL Rankweil	0,150
Mainboard	1	ATTiny1606	SOIC127P1032X265-20N	U1	DIGIKEY	1,010
	1	FT232RL	SSOP28	IC1	HTL Rankweil	2,900
	5	0u1	C0805	C1	HTL Rankweil	0,009
	100	27R	R1206	R3, R4	HTL Rankweil	0,099
	50	330R	R0805	R1, R2, R7	HTL Rankweil	0,060
	50	330R	R1206	R5, R6	HTL Rankweil	0,010
	25	4u7	153CLV-0405	C2, C3	DIGIKEY	0,345
	1	MF-SMDF200	MF-SMDF050	F1	DIGIKEY	0,480
	1	PIN4	PIN4_P	U\$3	HTL Rankweil	0,400
	1	TASTER_SMD	TASTER_SMD	S1	HTL Rankweil	0,730
	1	mini USB_B	65100516121	J1	HTL Rankweil	1,410
Gesamtpreis:						207,12



11 Gehäuse-Pläne

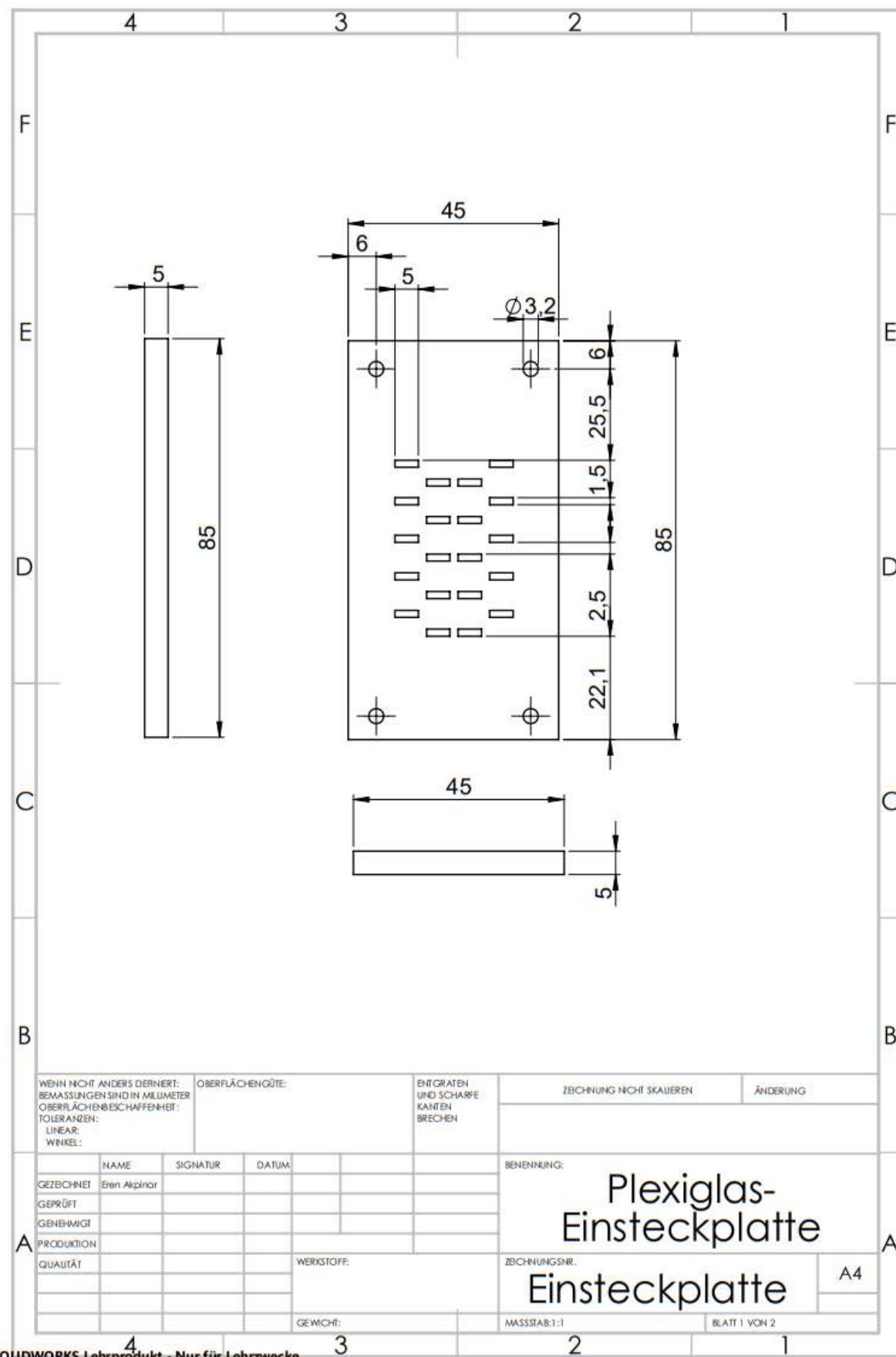
11.1 Unterseite:





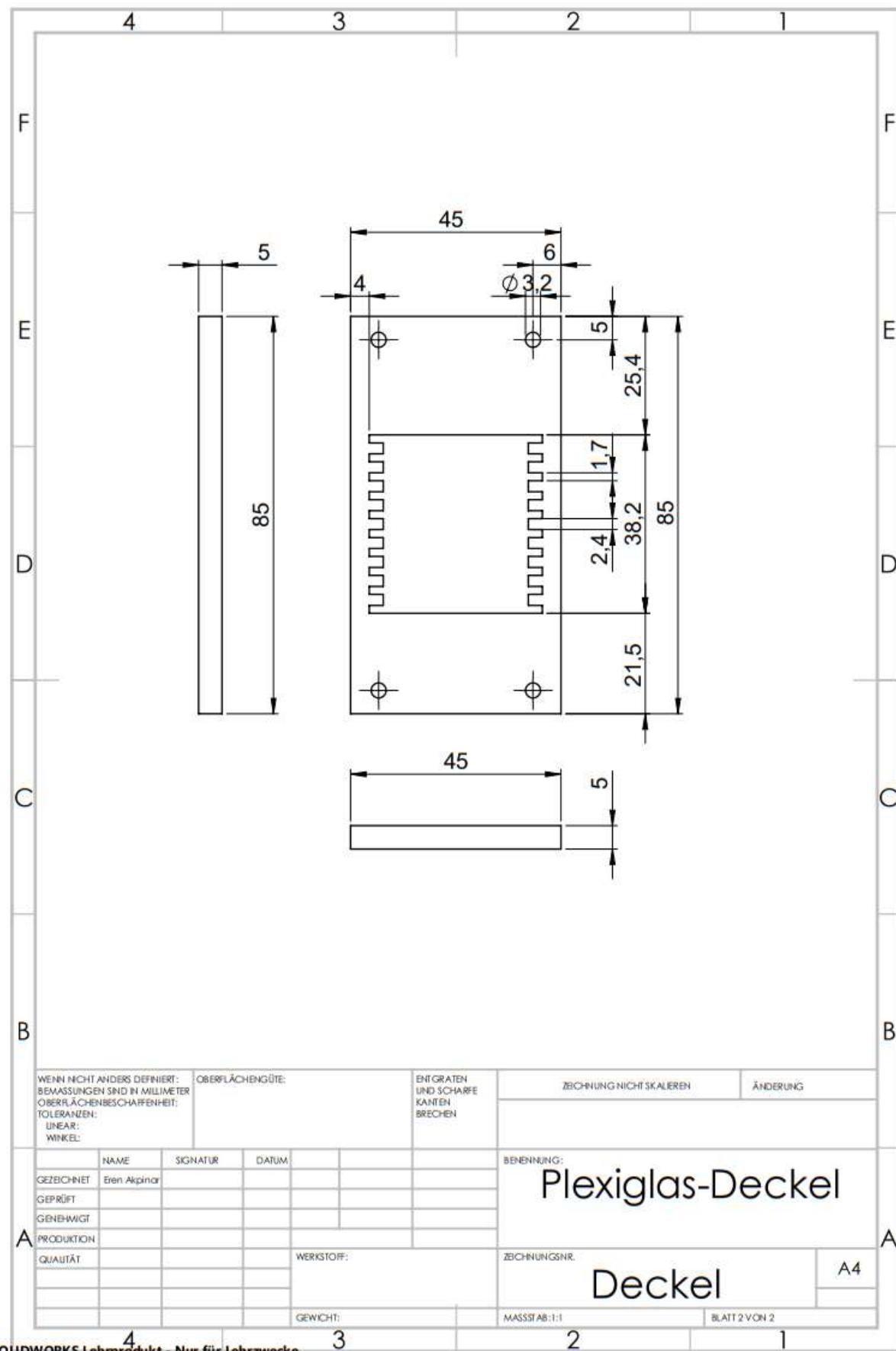


11.2 Einstechplatte:



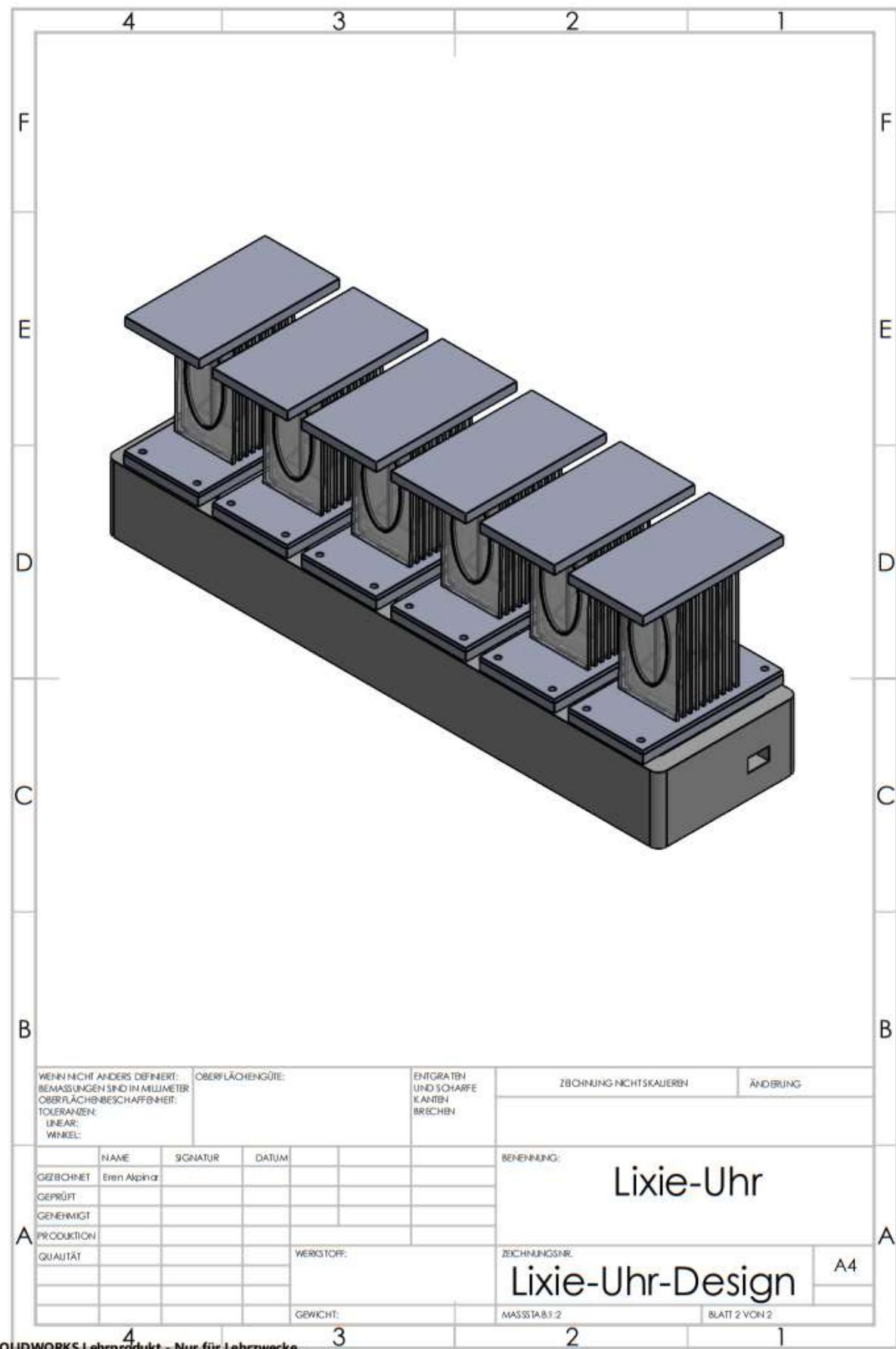


11.3 Deckel:





11.4 3D-Ansicht von der Uhr





12 3D-Druck (Gehäuse)

12.1 Verwendeter 3D-Drucker

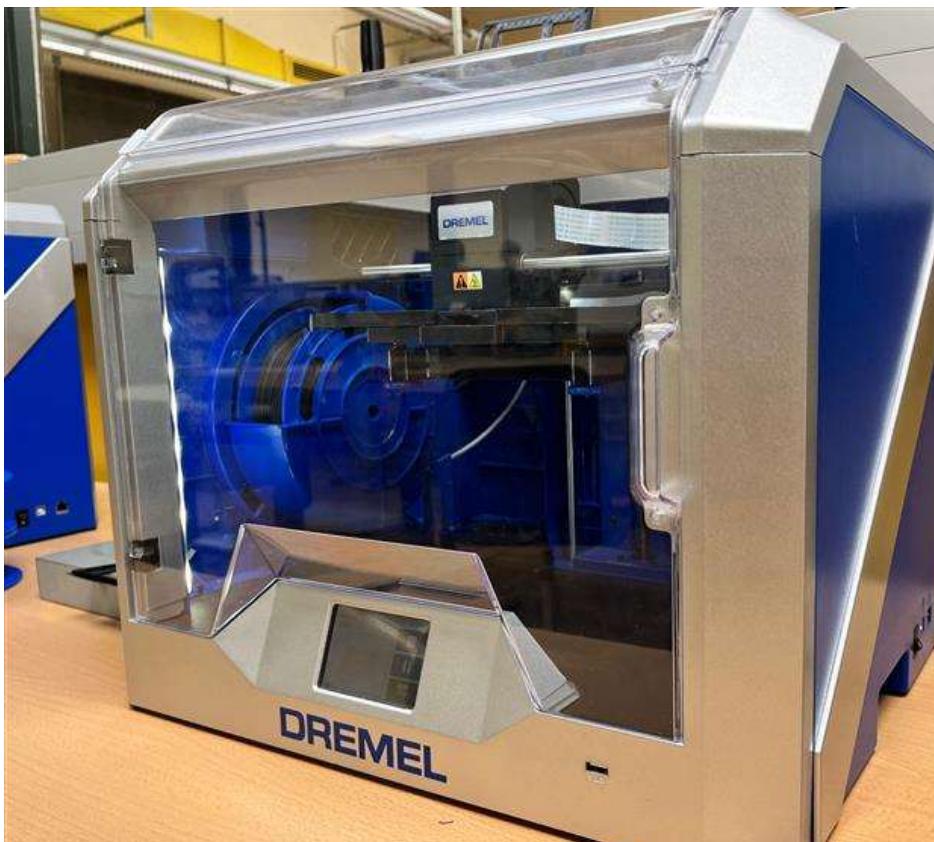


Abbildung 10: Dremel 3D-Drucker

Für den 3D-Druck wurde der Dremel 3D40 verwendet. Die SolidWorks-Dateien müssen zuerst in STL-Dateien umgewandelt werden, damit sie anschließend mit der Software "Dremel-DigiLab-3D-Slicer" geslicht werden kann.

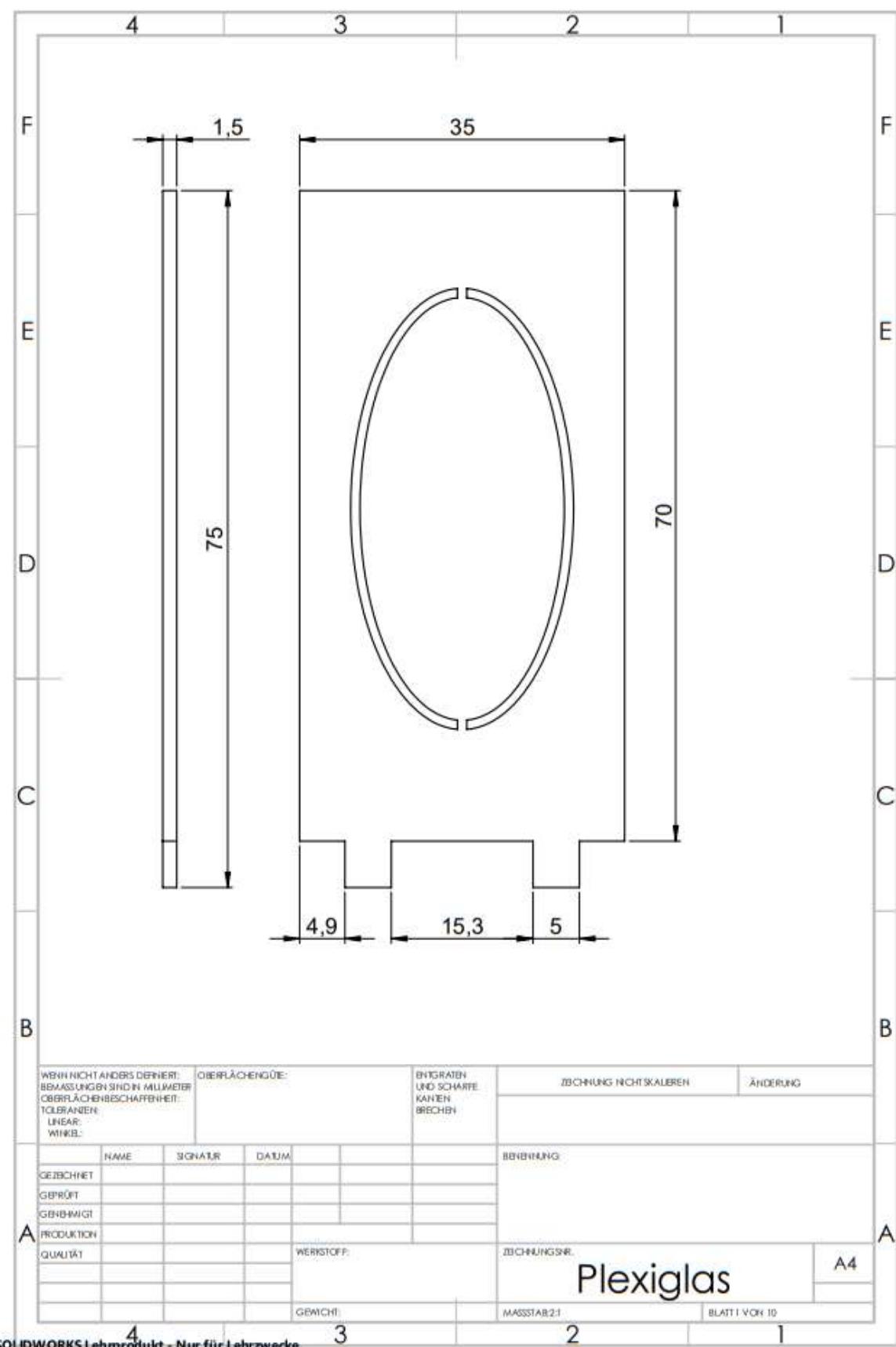
12.2 Gedruckte 3D-Werkstücke

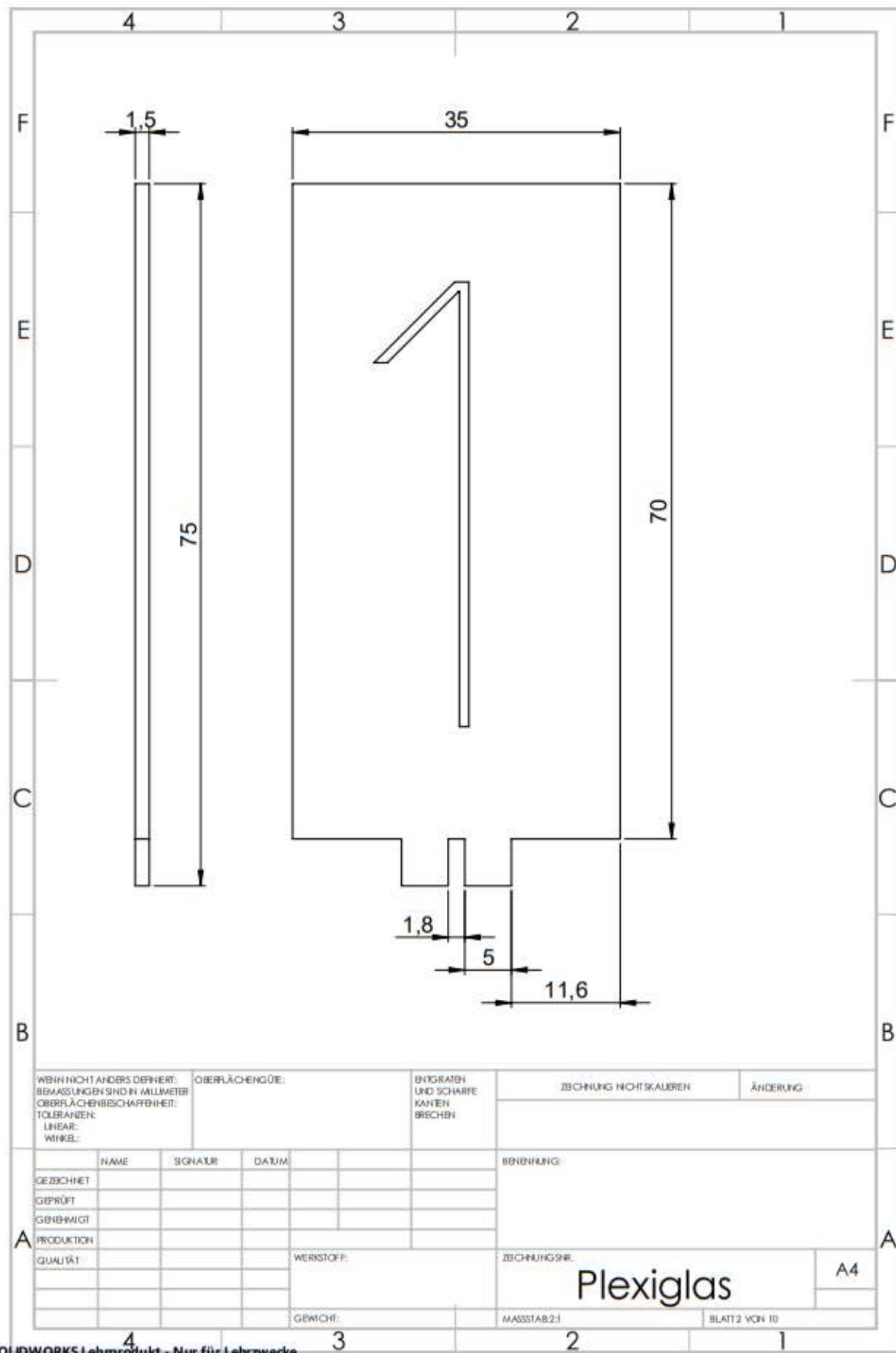


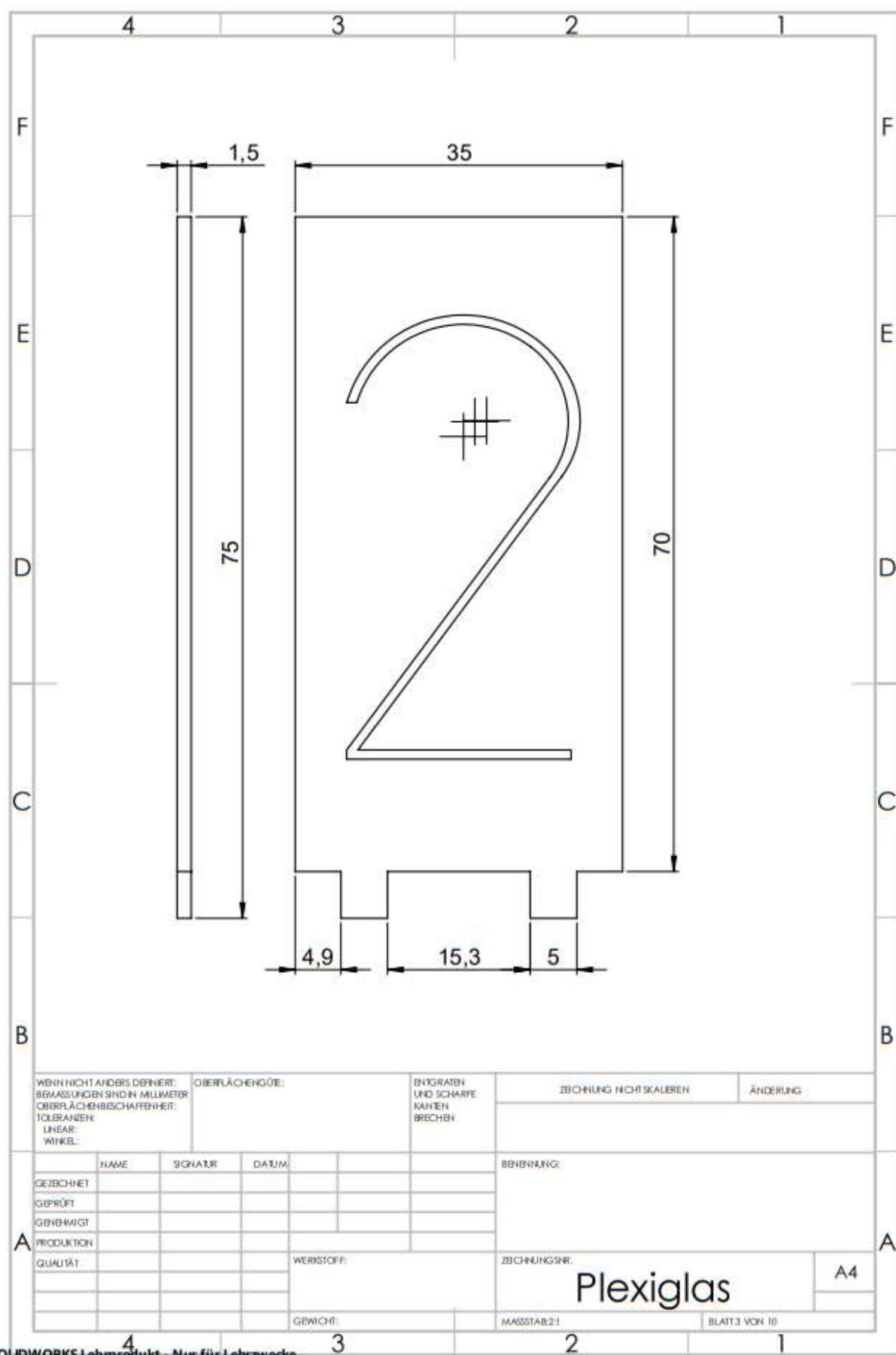
Abbildung 11: Werkstücke

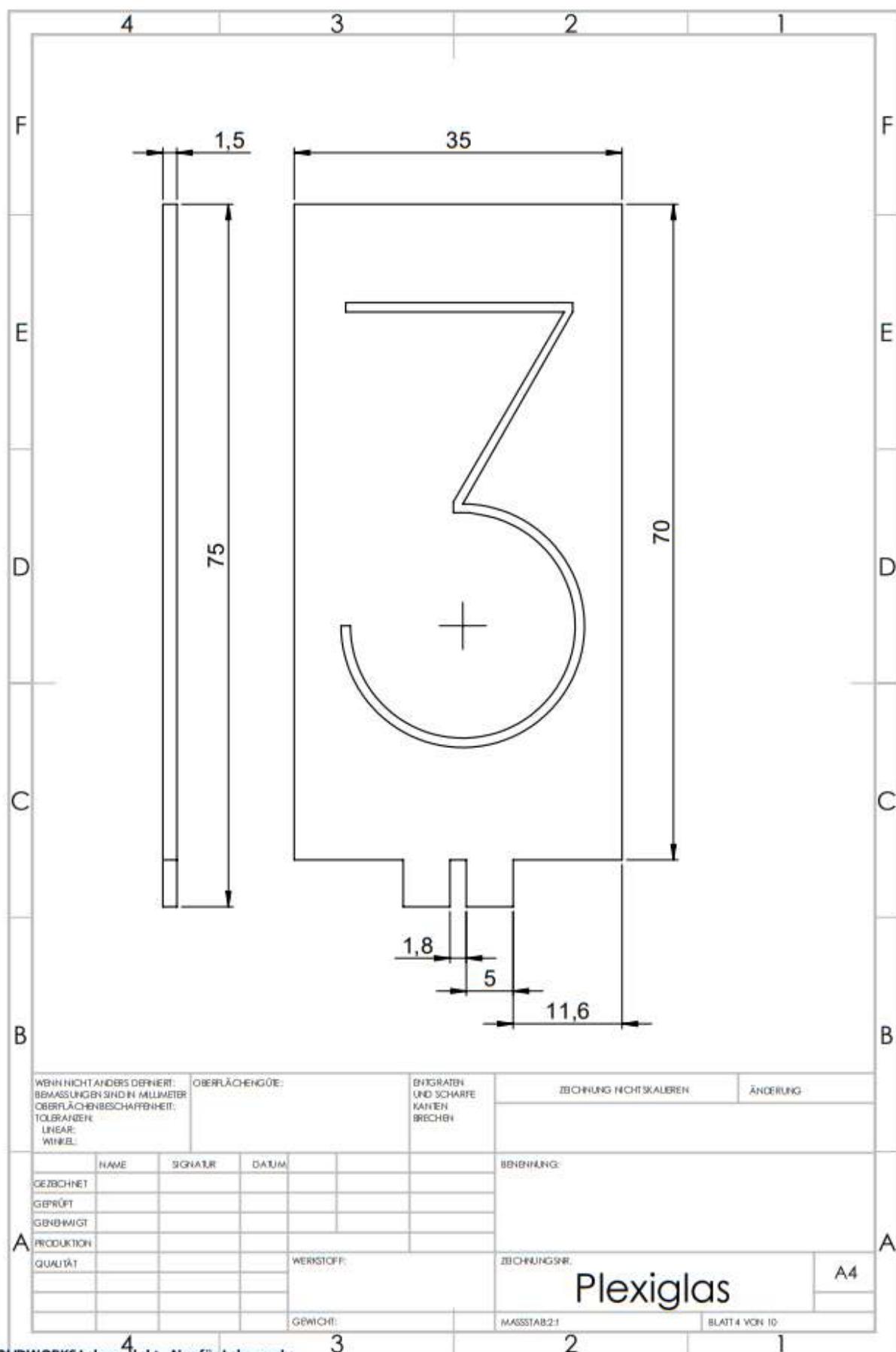


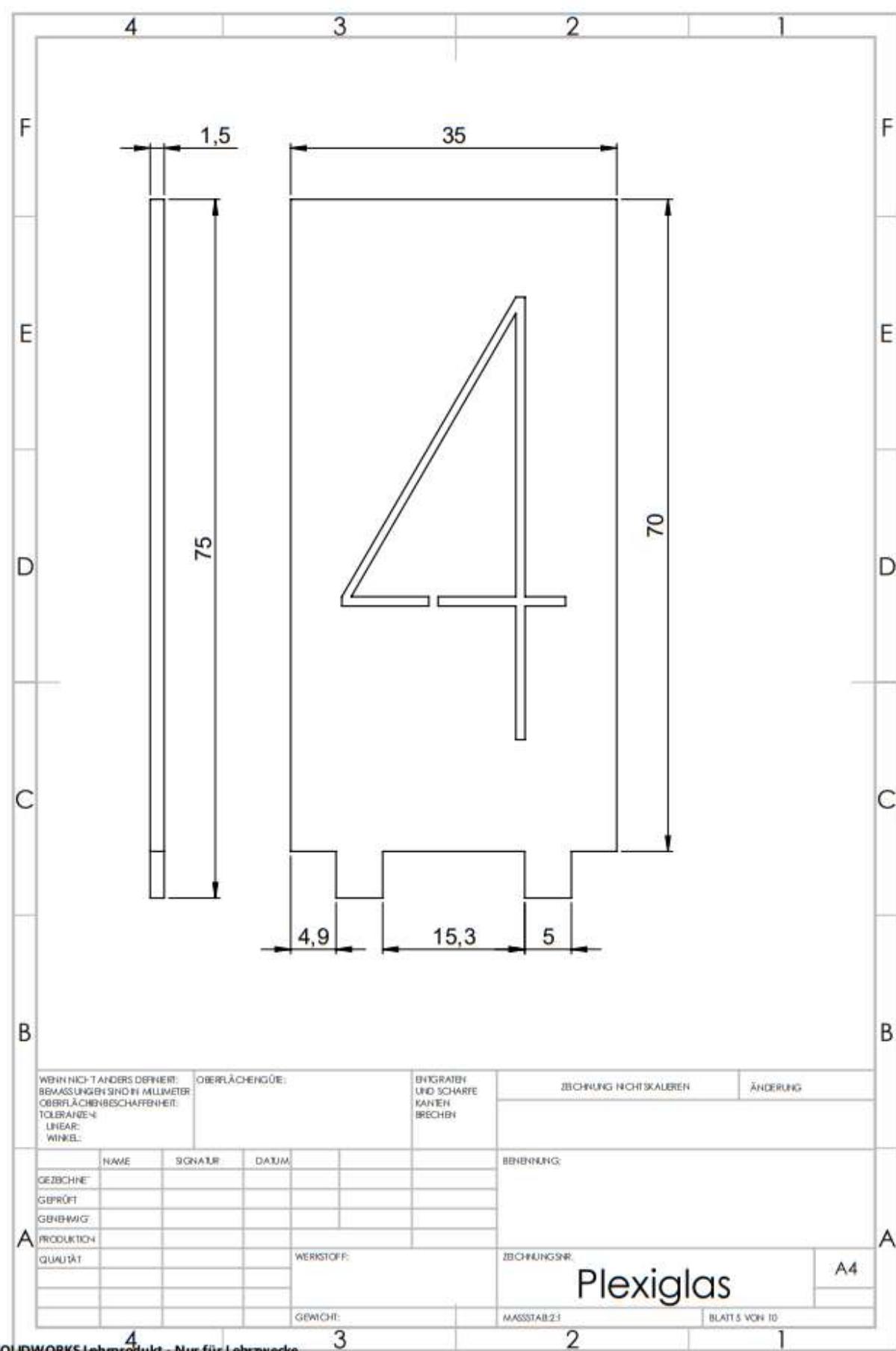
13 Gehäuse (Plexigläser)

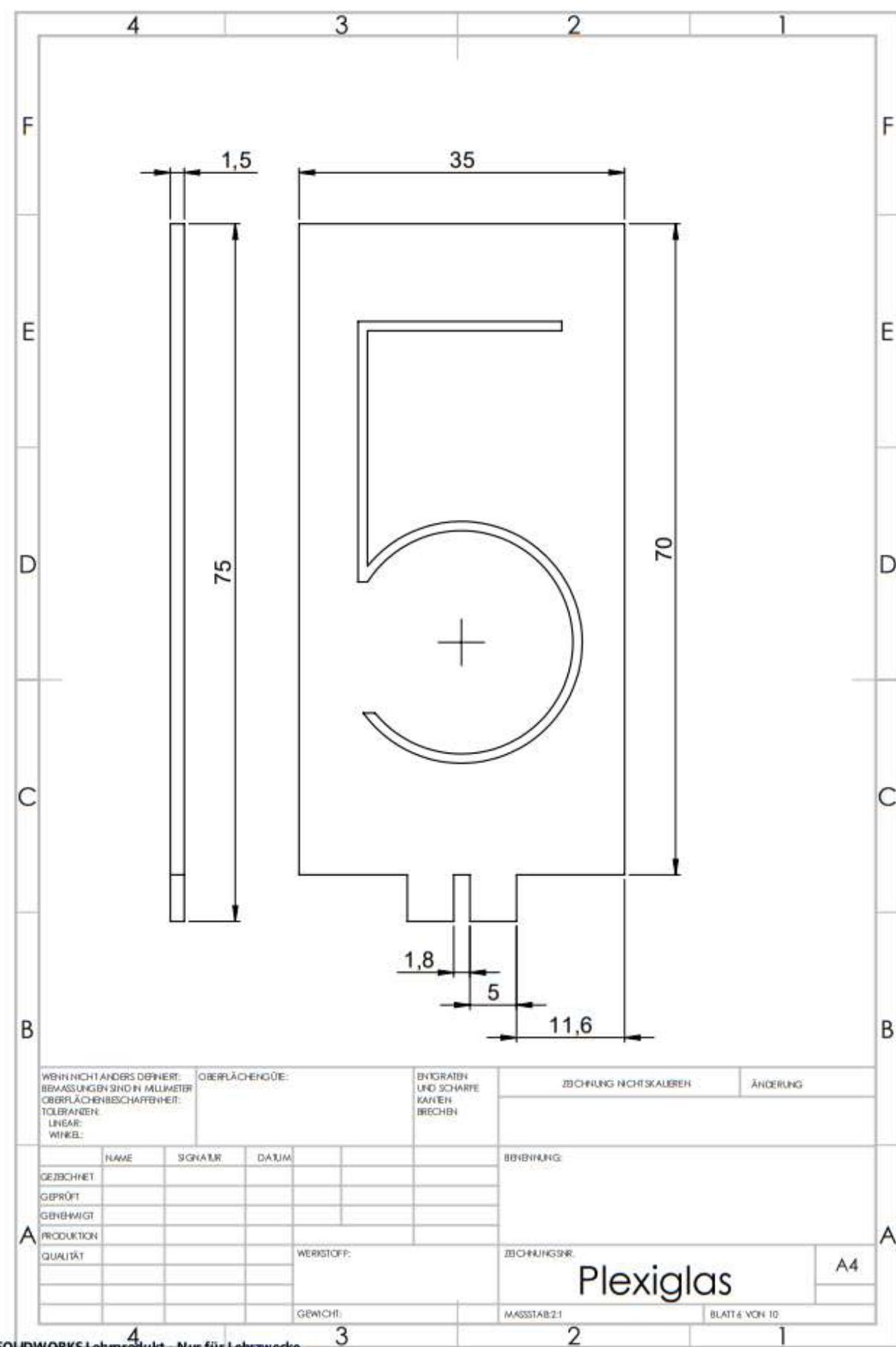


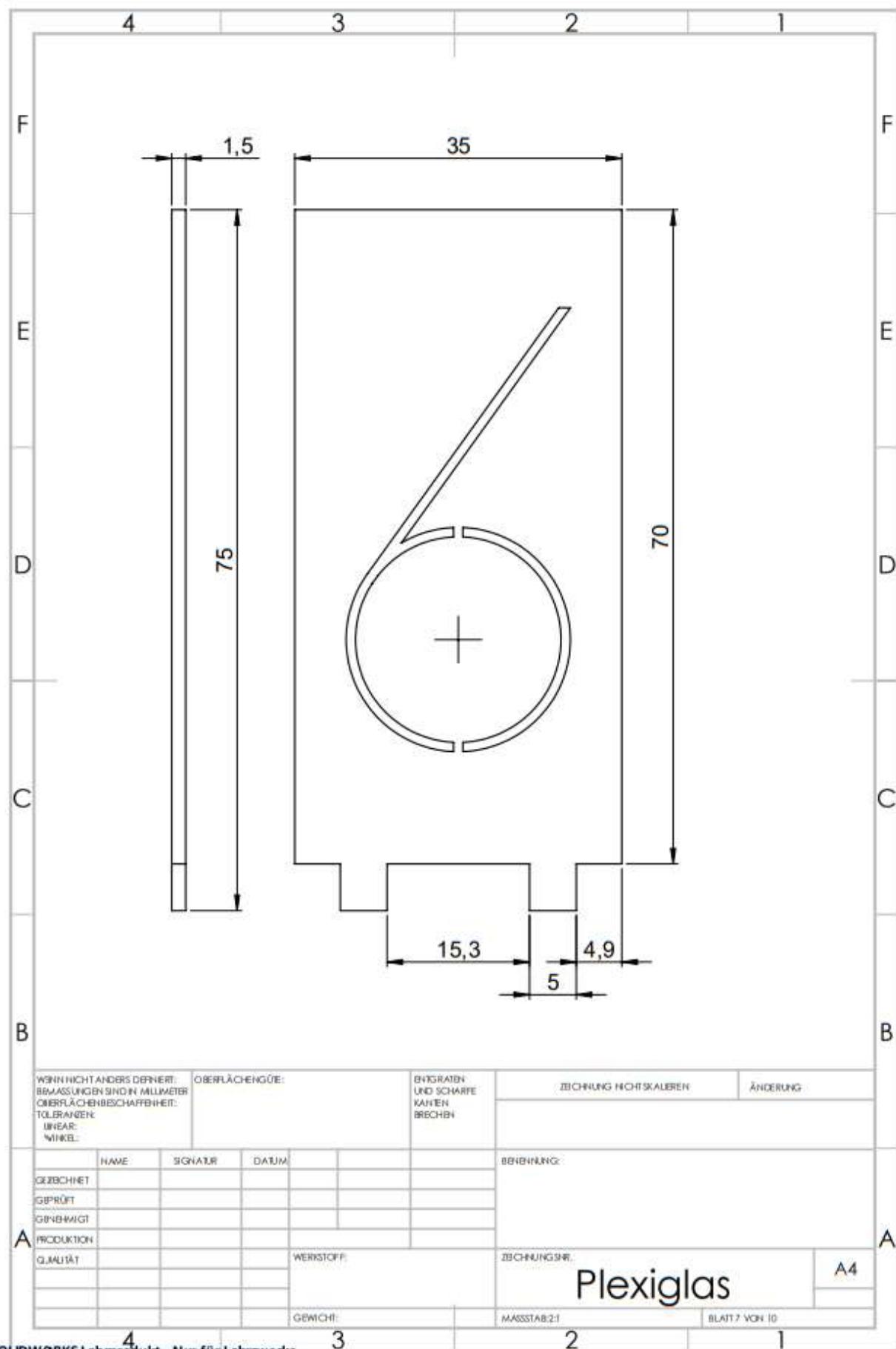




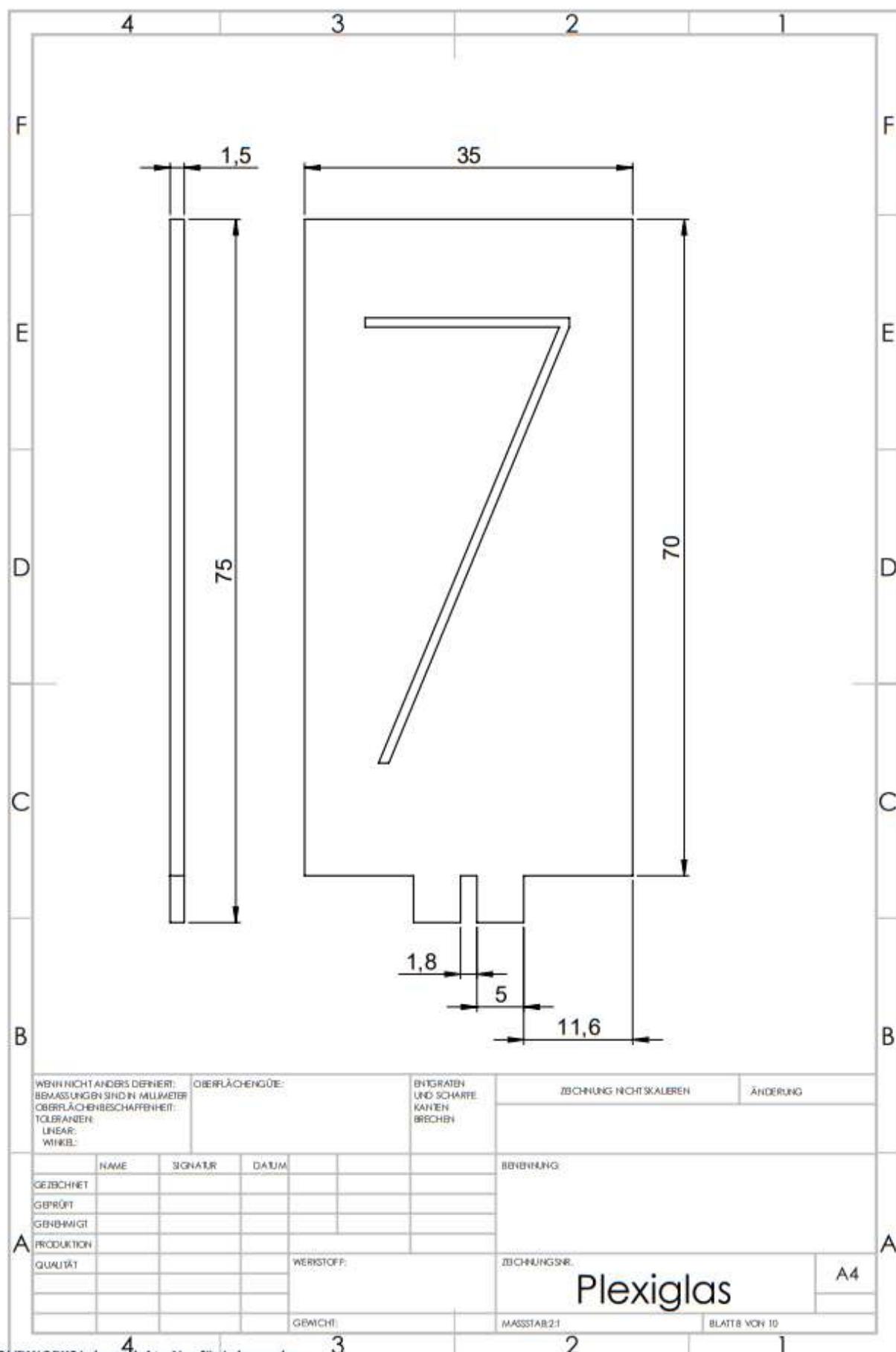


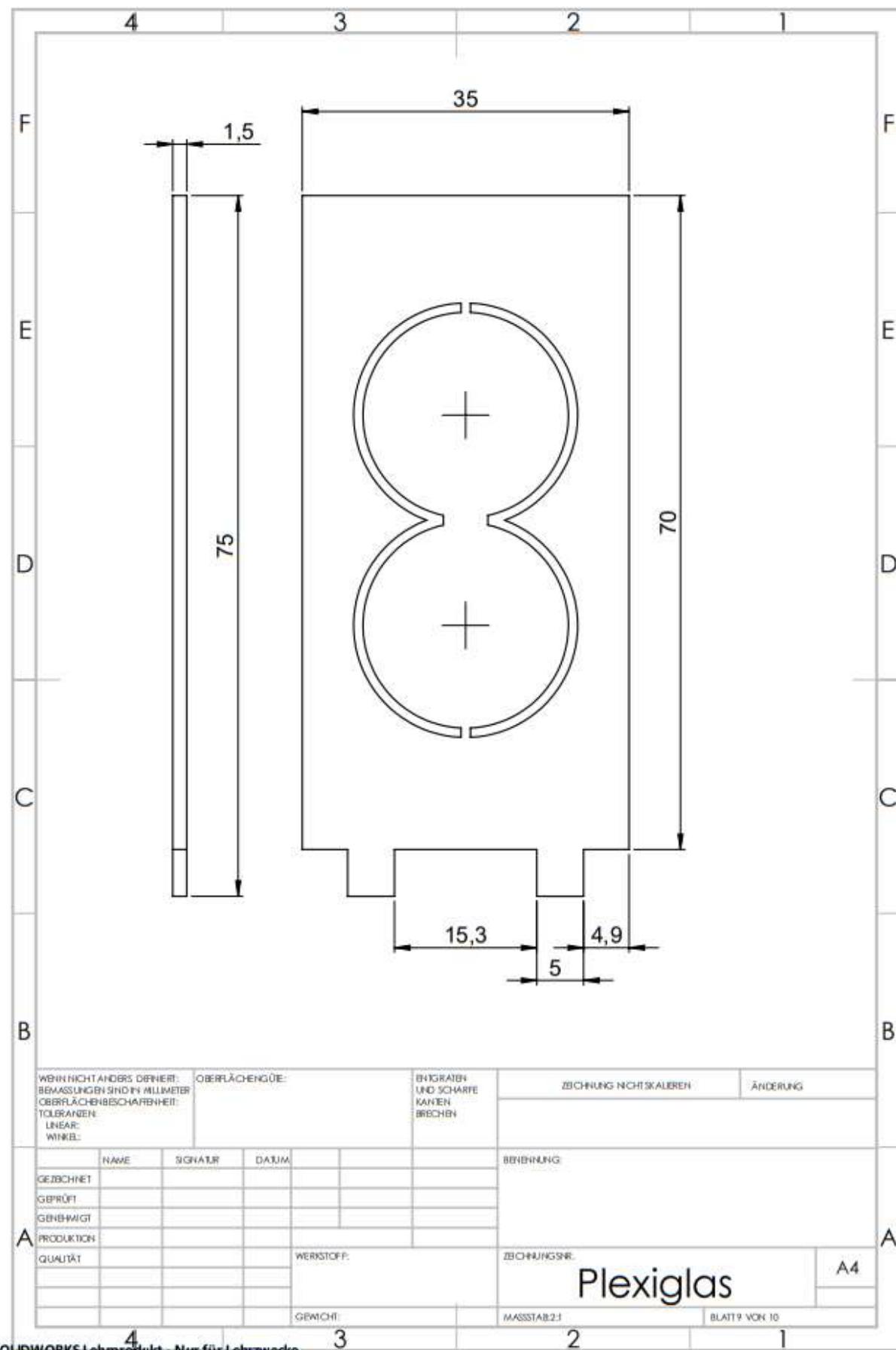


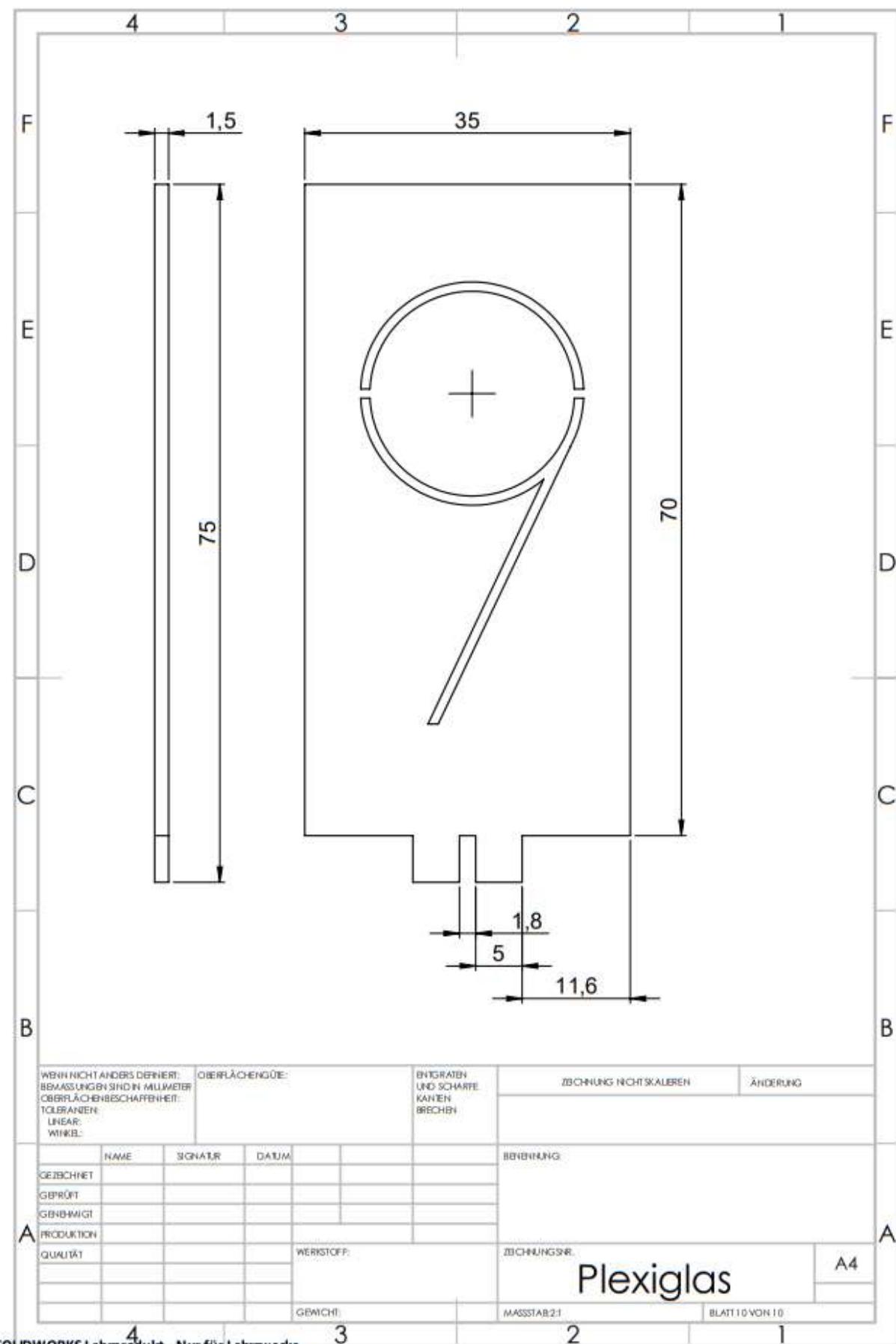




SOLIDWORKS Lehrprodukt – Nur für Lehrzwecke.

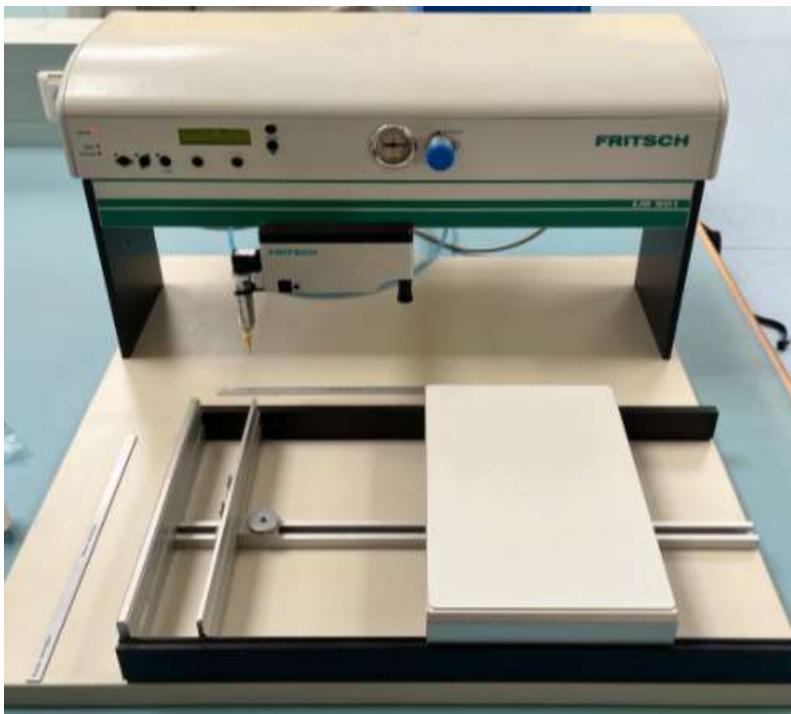








14 Bestückung der Platinen



Für das Auftragen von Lötpaste wurde die Dosiermaschine FRITSCH LM901 verwendet. Dabei kann man die Intensität der Lötpaste per Knopfdruck ändern.

Abbildung 12: Dosiermaschine

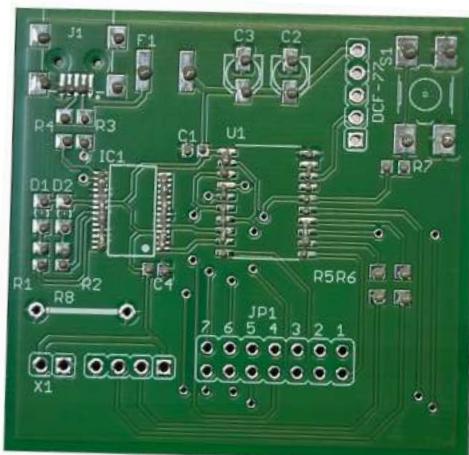


Abbildung 14: Master nach Dispensen

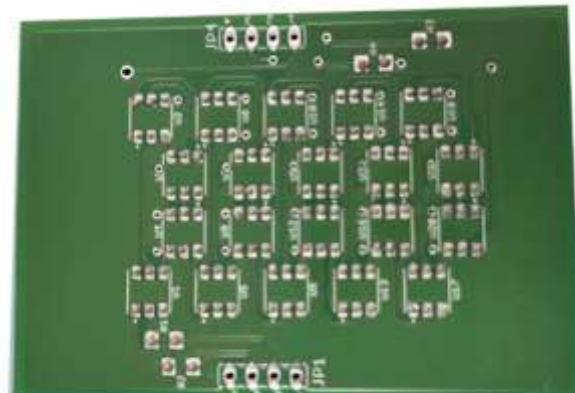
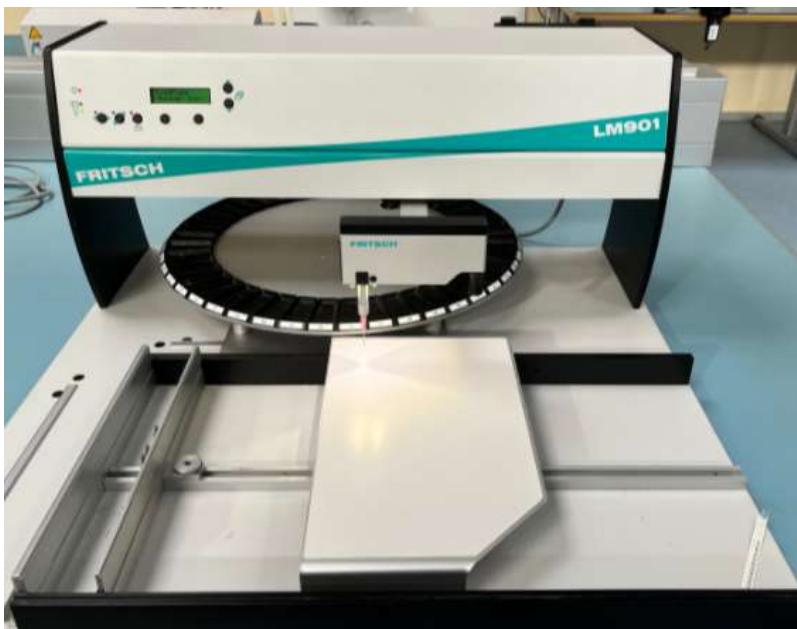
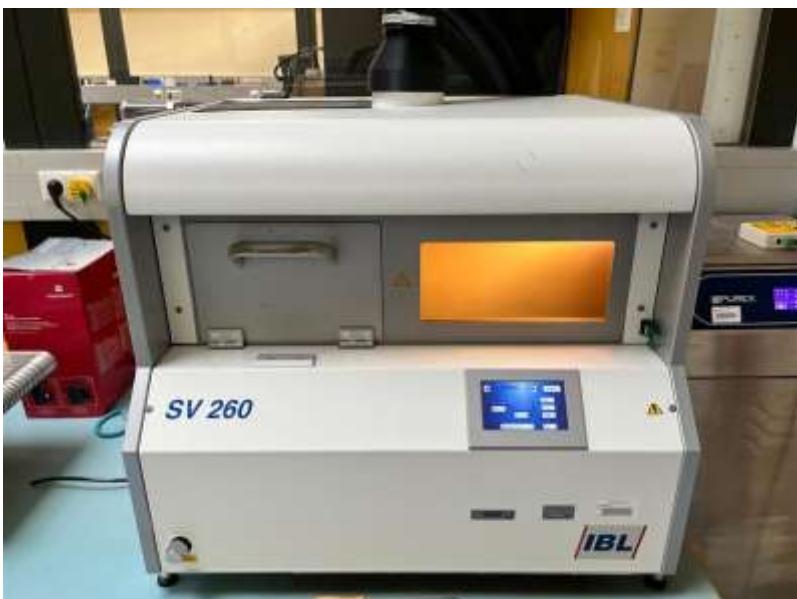


Abbildung 13: Segmentplatine nach Dispensen



Für das Bestücken der Bauteile wurde die Handbestückungsmaschine FRITSCH LM901 verwendet. Die Bestückung erfolgt mithilfe eines Vakuums. Die Bauteile werden durch das Vakuum festgehalten und können danach auf der Platine platziert werden.

Abbildung 15: Handbestückungsmaschine



Die SV260 ist eine Dampfphasenlötanlage und ermöglicht ein sauerstofffreies Löten ohne eine Überhitzung der Bauteile. Der Lötprozess erfolgt mit Hilfe von heißem Dampf. Die Wärmeableitung beim Kondensationslöten ist bis zu zehnmal größer als beim Konvektionslöten. Besonders große oder massive Leiterkarten können so problemlos in einer stabilen

Abbildung 16: Dampfphasenlötanlage

Prozessumgebung bearbeitet werden.



15 DCF-77-Empfangsmodul

15.1 Funktionsweise

Der DCF77 ist als Zeitzeichensender bekannt, der das aktuelle Datum und die Uhrzeit in digitaler Form über Langwelle mit einer Frequenz von 77,5 kHz überträgt. MEZ oder MESZ. Der DCF77-Sender steht in Mainflingen in Frankfurt. Es hat eine Reichweite von etwa 2000 km und steuert die meisten Funkuhren in Westeuropa.

DCF77 steht für:



Abbildung 17: DCF-77 Empfänger Modul

- D steht für Deutschland
- C ist ein Kürzel für Langwelle
- F steht für Frankfurt
- 77 ist die Übertragungsfrequenz

Das DCF77-Empfängermodul ist so aufgebaut, dass es während der Absenkung ein HIGH-Signal liefert, ansonsten ist das Signal LOW.

15.2 DCF77-Dekoder

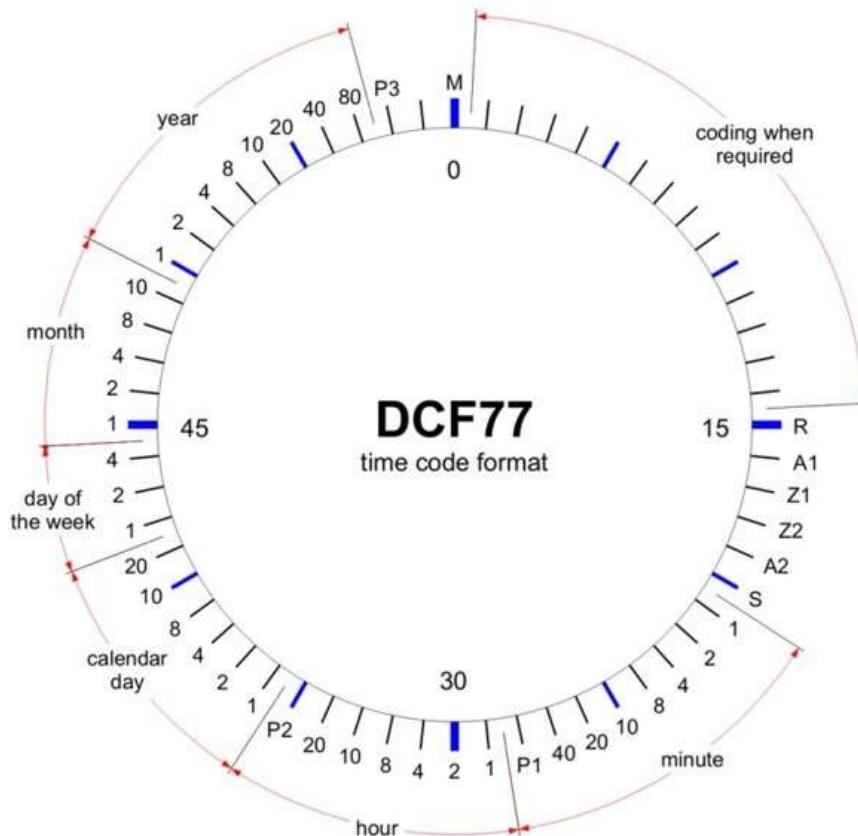
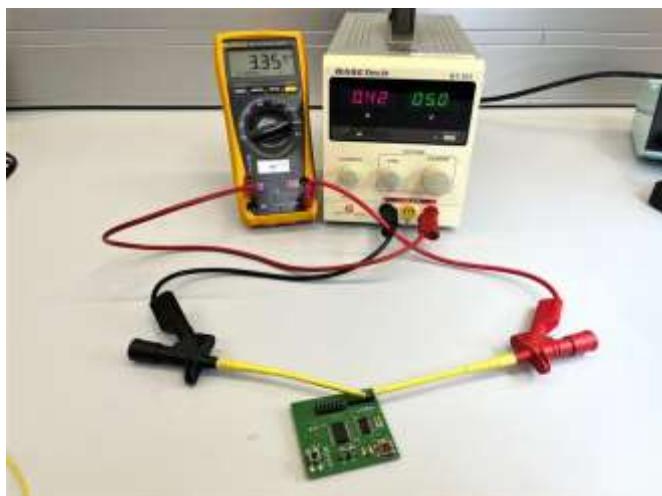


Abbildung 18: DCF77-Dekoder (DCF77-Dekoder, 2023)



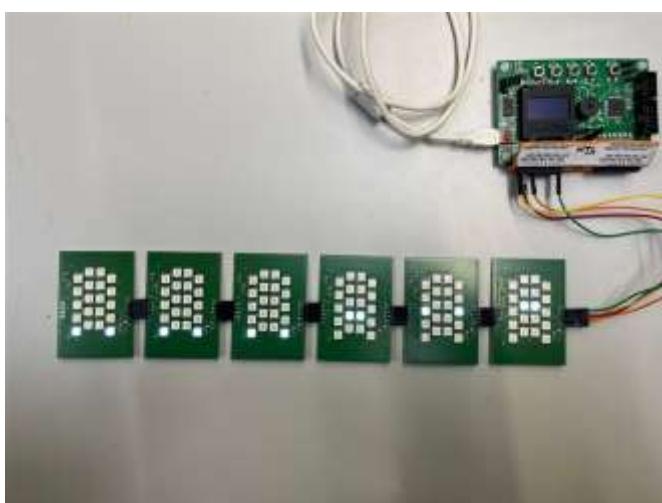
16 Messprotokoll

16.1 Mainboard



Der Ruhestrom der Masterplatine beträgt 3.35 mA und der Gesamtwiderstand der Schaltung vor der Inbetriebnahme beträgt $687\text{ k}\Omega$.

16.2 Segmentplatine



Der Ruhestrom der Segmentplatten beträgt 130 mA, ohne dass die LEDs in Betrieb sind.

16.2.1 Betrieb einer LED mit niedrigster Intensität

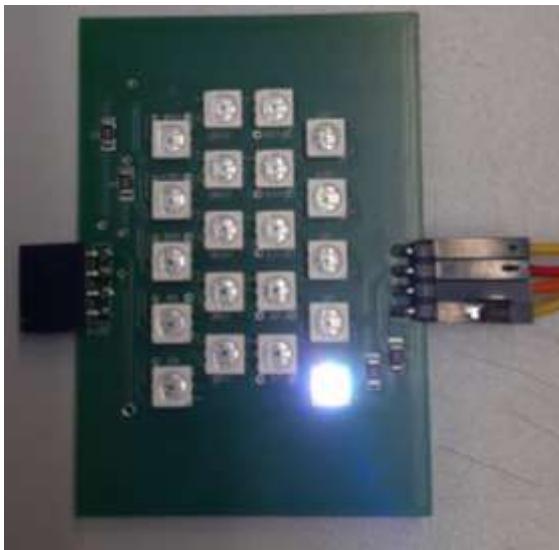


Um festzustellen wie viel Strom eine LED mit der niedrigsten Intensität verbraucht. Wurde eine LED eines Segments aktiviert. Der Strom beträgt dabei 12,31mA.





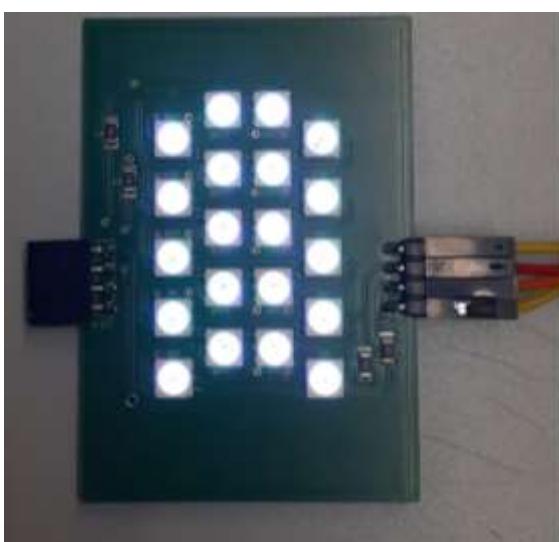
16.2.2 Eine LED mit höchster Intensität



Im Anschluss wurde eine LED mit höchster Intensität betrieben. Der Strom, der die LED dabei verbraucht beträgt $38,45mA$.



16.2.3 Ein Segment mit niedrigster Intensität



Die Gesamtstromaufnahme eines Segments wurde ebenso auf der niedrigsten Intensität gemessen. Der Strom, der dabei fließt, beträgt $53,4mA$.



16.2.4 Ein Segment mit höchster Intensität

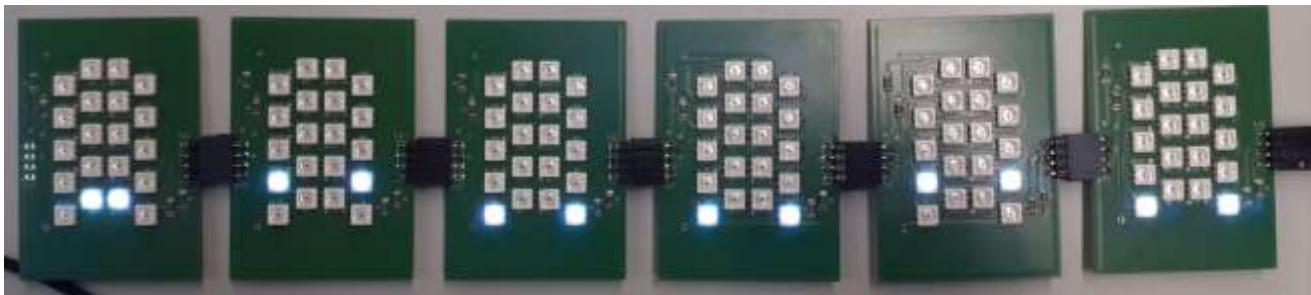


Anschließend wurden die LEDs noch mit der höchsten Intensität gemessen. Der Strom, der durch die Segmentplatine fließt, beträgt $337,8mA$.





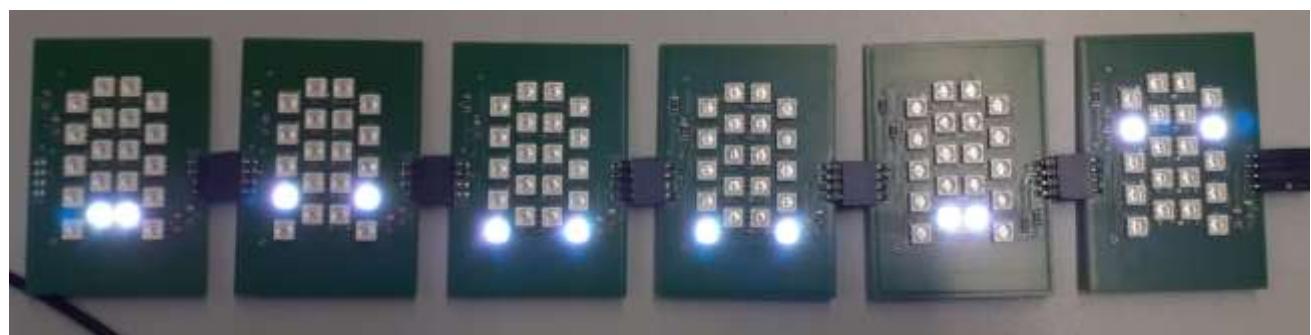
16.2.5 Alle Segmente im Betrieb mit niedrigster Intensität



Um nachzuvollziehen, wie viel die Lixie-Uhr im Betrieb verbraucht, wurden die Segmentplatinen miteinander verbunden. Der Gesamtstrom mit niedrigster Intensität beträgt $82.3mA$.



16.2.6 Alle Segmente im Betrieb mit höchster Intensität



Bei höchster Intensität beläuft sich die Gesamtstromaufnahme auf $291,3mA$.





16.3 DCF77-Empfängermodul

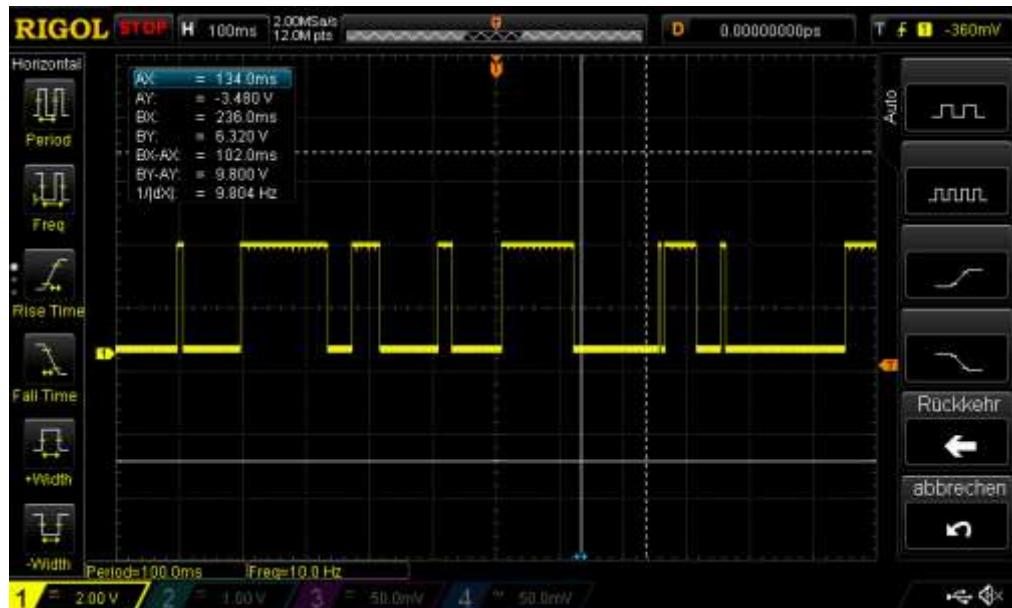


Abbildung 19: DCF-77-Signal

Aus Abbildung 20 lässt sich das DCF-77-Signal erkennen. Das DCF77-Signal beinhaltet die Uhrzeit, das Datum und das Wetter. Die Information wird übertragen, indem das Funksignal einmal pro Sekunde für entweder 100 oder 200 Millisekunden auf den Pegel LOW abgesenkt wird. Eine Absenkung für 100 ms bedeutet eine „logisch 0“, 200 ms bedeutet eine „logische 1“. Die Sequenz dauert eine Minute. Somit werden 60 Bit übertragen, aber das letzte Bit dient zur Synchronisierung

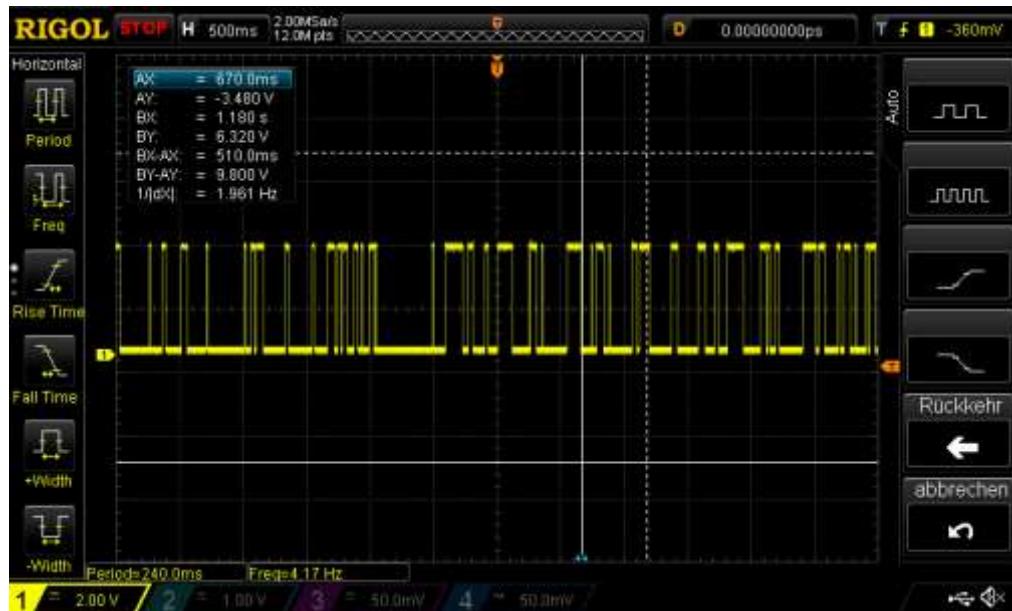


Abbildung 20: DCF77-Signal



17 APA102c-LED Funktionsanalyse

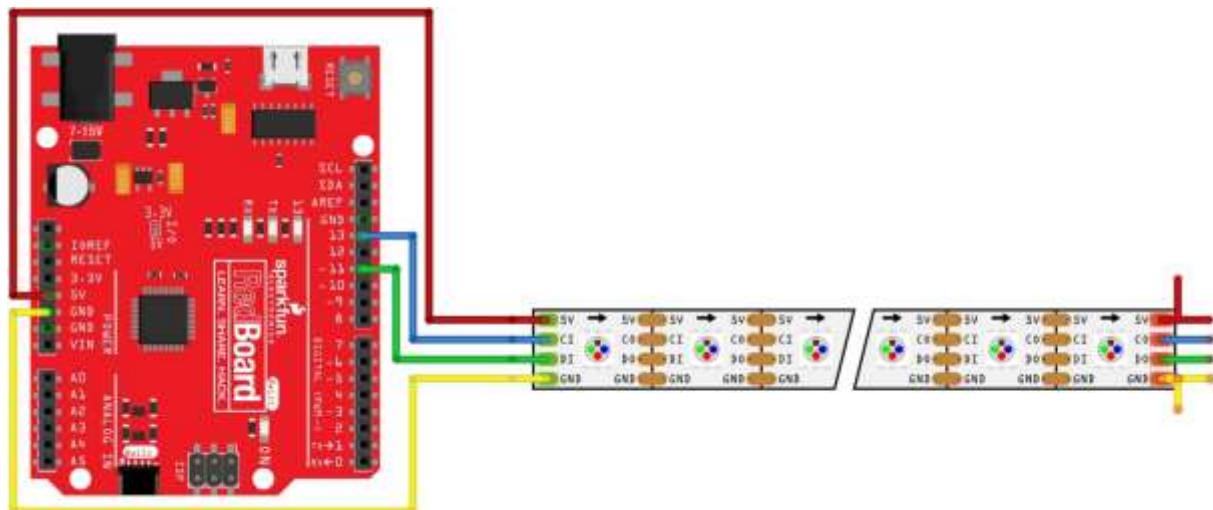


Abbildung 21: Fritzing APA102c

Wie jede LED, hat auch der APA102c ein Pin für die Versorgung und ein Pin für GND. Die Daten werden über den Data-In (DI) Pin empfangen. Im Vergleich zu WS2812 ist bei APA102c die Timing-Toleranz besser, sodass die APA102c LED PWM-Frequenzen von bis zu 20kHz erreichen können.



17.1 Demoprogramm-Quelltext

```
/*
 * 10_Apa102c.c
 *
 * Created: 30.11.2022 09:19:45
 * Author : 170334
 */

#define F_CPU 12000000UL

#include <avr/io.h>      //Libraries aufrufen
#include <util/delay.h>
#include "spi/spi.h"

void init(void)
{
    //PORT Setup
    DDRA = 0x00;
    PORTA = 0x0F; //Enable Pullup

    DDRC = 0xFF;
}

int main(void)
{
    init();                //Aufruf der Grundinitialisierung

    // SPI Bus
    //           Mode: Master
    //           Clock: F_CPI/128
    // Direction: MSB First
    //           Polarity: Rising
    //           Phase: Rising
    spi_init(SPI_Master, SPI_MSB, SPI_Rising, SPI_Rising);

    while (1)
    {
        _delay_ms(10);
        spi_transfer(0x00);      //Start Time
        spi_transfer(0x00);      //Start Time
        spi_transfer(0x00);      //Start Time
        spi_transfer(0x00);      //Start Time

        spi_transfer(0xFF);      //LED FRAME-Helligkeit
        spi_transfer(0x00);      //LED FRAME-Farbe (BLAU)
        spi_transfer(0x00);      //LED FRAME-Farbe (GRUEN)
        spi_transfer(0xFF);      //LED FRAME-Farbe (ROT)

        spi_transfer(0xFF);      //LED FRAME
        spi_transfer(0xFF);      //LED FRAME
        spi_transfer(0xFF);      //LED FRAME
        spi_transfer(0xFF);      //LED FRAME

        _delay_ms(10);           //Zeitverzögerung von 10ms
    }
}
```



17.2 Aufbau mit APA102c

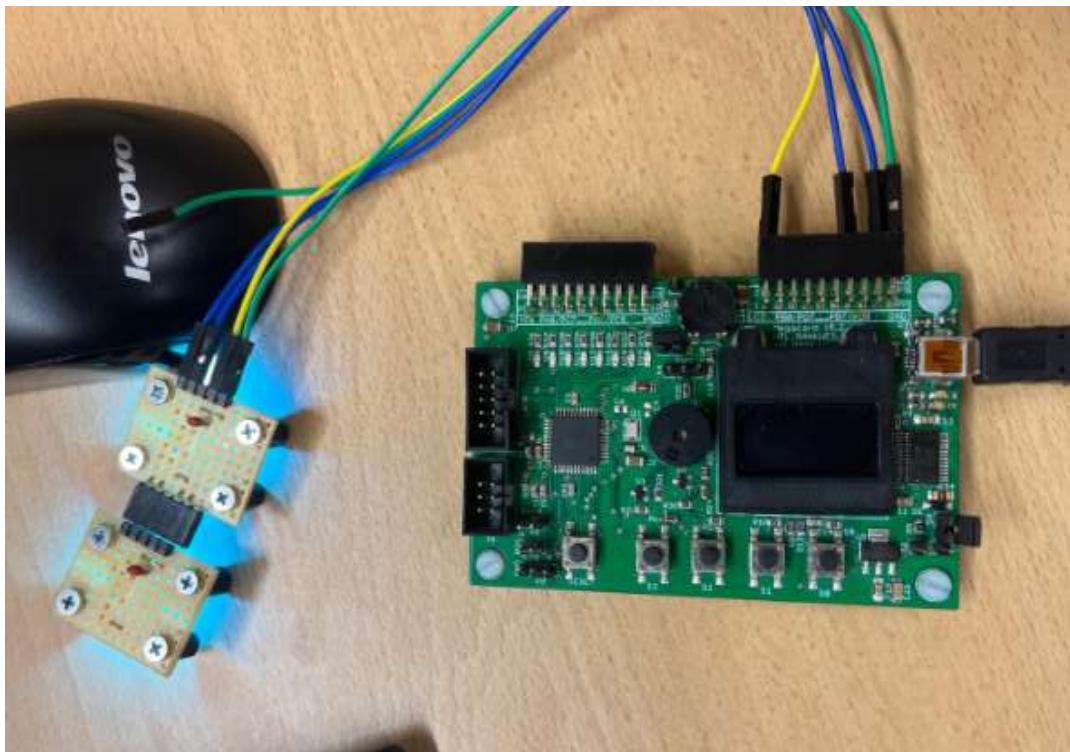
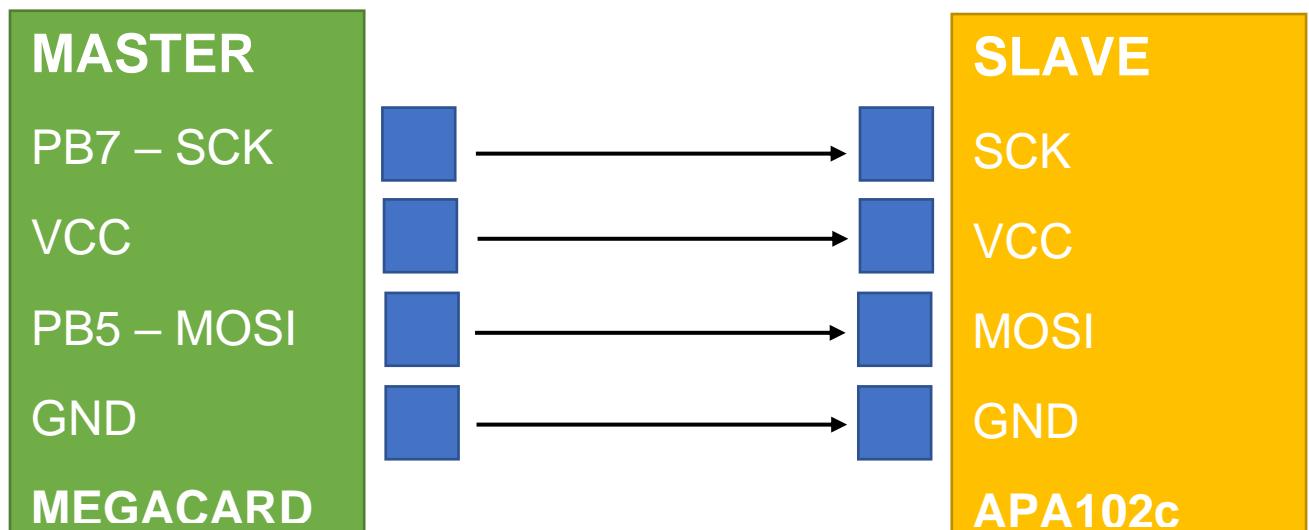


Abbildung 22: Funktionstest APA102c

Der Aufbau besteht aus der Megacard (ATMEGA16) und 2 Stück der APA102c LED. Die Verbindung erfolgt durch 4 Steckverbindungen.

17.2.1 Blockschaltbild:





18 Software

18.1 Projektverwaltung / Verwendete Programme

Für die Verwaltung des Projektes und der Projektumsetzung wurden folgende Programme verwendet:



Abbildung 23: Github Logo (Github, 2023)

GitHub / GitHub Desktop

GitHub wurde verwendet, um die Programmcodes, die Gehäusedaten, die PCB-Daten, die Stücklisten und die Datenblätter für alle Teammitglieder einfach zugänglich zu machen und ebenso sicher zu verwalten. Änderungen konnten so jederzeit nachvollzogen und im Fehlerfall rückgängig gemacht werden



Abbildung 24: OneDrive Logo (OneDrive, 2023)

OneDrive

OneDrive wurde hauptsächlich für die Verwaltung der Dokumentation verwendet. OneDrive ermöglicht den gleichzeitigen Zugriff auf Office Dateien (beispielsweise, die Dokumentation).

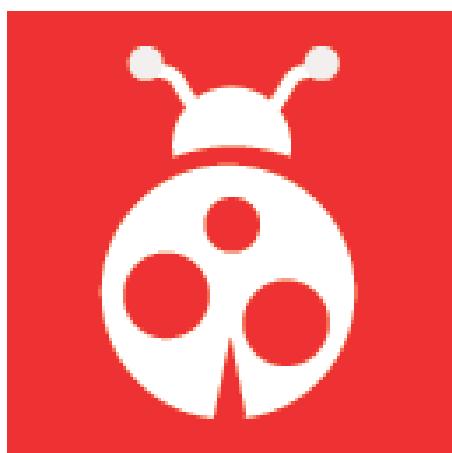


Abbildung 25: Microchip Studio Logo (Studio, 2023)

Microchip Studio

Microchip Studio ist die Programmierumgebung, die für dieses Projekt verwendet wurde. Mithilfe von Microchip Studio ist es möglich, den ATMEL ICE zu verwenden, um den Attiny1606 In-System (über UPDI) zu programmieren.

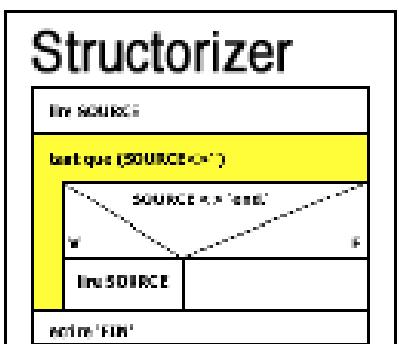


Abbildung 26: Structorizer Logo (Structorizer, 2023)

Structorizer

Der Structorizer wurde verwendet, um Struktogramme erstellen zu können. Struktogramme sind eine vereinfachte bzw. übersichtlichere Darstellung von Programmcode.

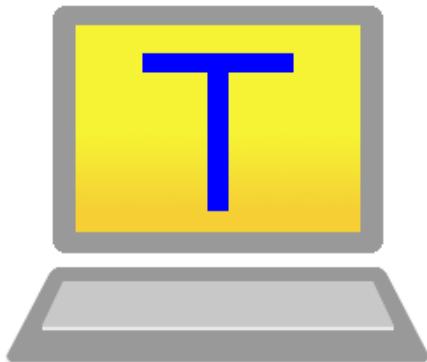


Abbildung 27: TeraTerm Logo (TeraTerm, 2023)

Teraterm

Teraterm wurde verwendet, um die Kommunikation zwischen der Lixie-Uhr und dem Rechner zu ermöglichen. Mithilfe von Teraterm lässt sich die Uhrzeit einstellen.



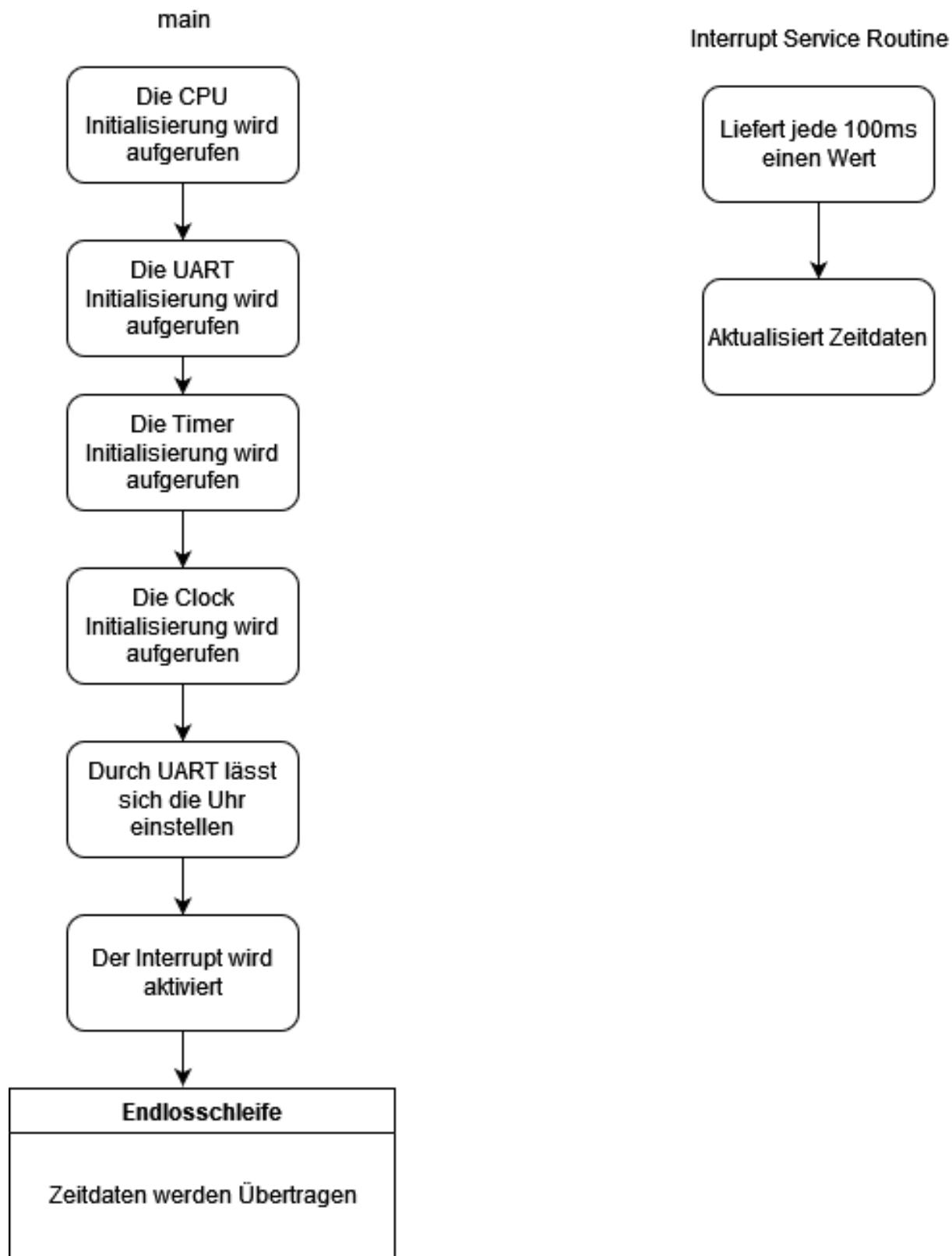
Abbildung 28: FTDI Chip Logo (Chip, 2023)

FT_Prog

Der FTDI Programmer dient der Programmierung der FTDI ICs.

18.2 Projektkonzept / Programmkonzept

Ein Flussdiagramm, dass den Ablauf des Programmes übersichtlich veranschaulichen soll:





18.3 Programmbeschreibung

Um die Lixie-Uhr in Betrieb nehmen zu können, wird ein Programm benötigt. Dieser entstandene Code wurde in der Sprache C für den Attiny1606 (tinyAVR® 0-series) geschrieben.

Das Programm erlaubt dem Benutzer, die Lixie-Uhr an einem Rechner mit einem Mini-USB zu USB-Kabel anzuschließen und über eine Konsole die Uhrzeit, die Farben und die Intensität des Lichts einzustellen.

Primär ist das Programm eine Uhrenansteuerung.

Der Mikrocontroller sendet die Byteframes kontinuierlich an die LEDs.

Sekundär erlaubt die Software eine individuelle Anpassung der Uhr.

Man kann beispielsweise das aktuelle Datum anzeigen oder ein Farbmuster generieren.



18.4 Entwicklungsablauf / Codebeschreibung

Das Programm wurde zu Testzwecken für den Atmega16 geschrieben und anschließend für die AttinyAVR® 0-serie umgeschrieben. Zu Beginn wurde der Attiny406 in Betracht gezogen, doch aufgrund der geringen Speicherkapazität und die Notwendigkeit der Benutzung des UART-Interfaces, wurde der Attiny406 durch den Attiny1606 der statt 4 Kilobyte, 16 Kilobyte an Speicher bietet, ersetzt.

Insgesamt wurde mit 4 Teilprojekten gearbeitet:

- ▲ **01_Lixie_Clock_ATmega16**
 - ▷ Dependencies
 - ▷ Output Files
 - ▷ Libraries
 - ▷ clock
 - ▷ led
 - ▷ spi
 - ▷ uart
 - ▷ main.c

Dieses Projekt wurde für den ATmega16 erstellt. Nach Fertigstellung der Segmente wurde der Code mithilfe der MEGACARD V5.5 getestet. Da dieses Projekt nach ein paar Fehlerbehebungen funktioniert hatte, wurde es zum Grundprojekt für den Attiny1606.

- ▲ **02_Lixie_Clock_ATtiny1606**
 - ▷ Dependencies
 - ▷ Output Files
 - ▷ Libraries
 - ▷ clock
 - ▷ led
 - ▷ spi
 - ▷ uart
 - ▷ main.c

Dieses Projekt stellt das Hauptprojekt der Abschlussarbeit dar und wurde mit den Attiny1606. verwendet. Von der Funktionsweise unterscheidet sich dieses Projekt nicht zum ersten Projekt, jedoch musste der Code von der ATmega auf die Attiny Plattform angepasst werden.

- ▲ **03_Platinentest**
 - ▷ Dependencies
 - ▷ Output Files
 - ▷ Libraries
 - ▷ led
 - ▷ spi
 - ▷ main.c

Dieses Projekt wurde für das erstmalige Testen der Segmente verwendet. Es ist hauptsächlich dafür gedacht, die LED's auf jeder Platine zu Testen und zu messen.

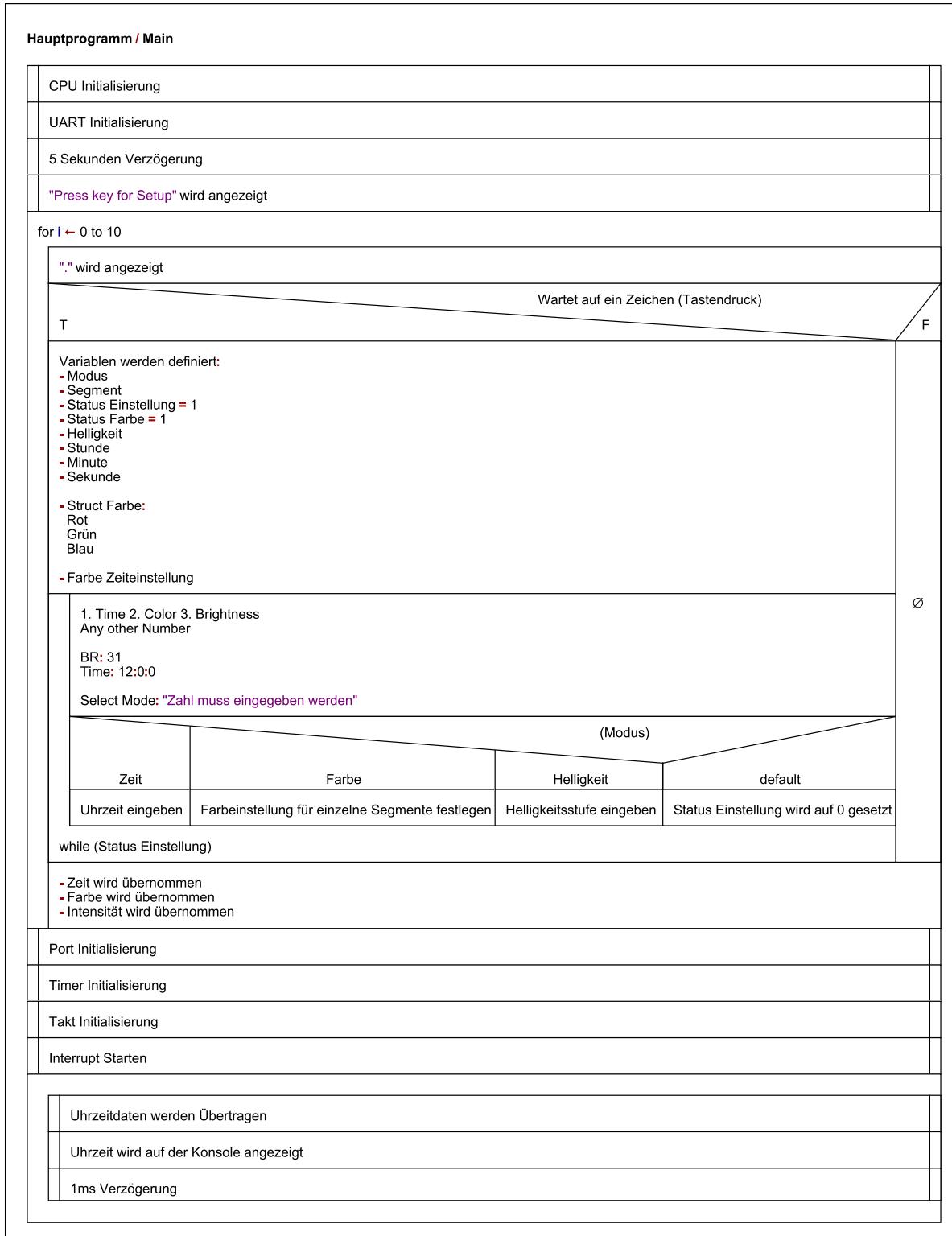
- ▲ **Attiny1606_UART_Test**
 - ▷ Dependencies
 - ▷ Output Files
 - ▷ Libraries
 - ▷ main.c

Dieses Projekt wurde für das Testen der UART-Funktion des Attiny1606 verwendet. Da es Schwierigkeiten mit der UART-Funktion gab, wurde das Demoprogramm für den Attiny1606 verwendet, um die UART-Funktion zu testen.



18.5 Hauptprogramm

18.5.1 Struktogramm





18.5.2 Code

```
int main(void)
{
    cpu_init();
    uart_init();

    _delay_ms(5000);

    printf("\n\n\n\rPress key for setup\n\r");

    char data = 0;

    for(unsigned char i=0; i < 10; i++)
    {
        printf(".");
        if(uart_scanchar_nonblocking(&data) == UART_Received)
        {

            unsigned int mode = 0;
            unsigned int segment = 0;
            unsigned int status_settings = 1;
            unsigned int status_color = 1;
            unsigned int brightness = 31;

            unsigned int hour = 9;
            unsigned int minute = 11;
            unsigned int second = 0;

            typedef struct
            {
                unsigned int R;
                unsigned int G;
                unsigned int B;
            } Clock_Color_t;

            Clock_Color_t hour_setting = {255,255,255};
            Clock_Color_t minute_setting = {255,255,255};
            Clock_Color_t second_setting = {255,255,255};

            // Run setup:

            do
            {
                printf("\n\n\r1.Time      2.Color     3.Brightness\n\rAny other Number: Exit\n\r\n\rBR: %u\n\rTime: %u:%u:%u\n\rSelect Mode: ",
                    brightness, hour, minute, second);

                scanf("%2u", &mode);

                switch(mode)
                {
                    case 1:
                    do
                    {
                        printf("\n\n\rSet Time [Hours:Minutes:Seconds]: ");
                        scanf("%2u:%2u:%2u", &hour, &minute, &second);
                        printf("\r");
                    } while ((hour > 23) || (minute > 59) || (second > 59));

                    break;

                    case 2:
                    do
                    {
                        status_color = 1;
                        printf("\n\n\r1. Hours      2. Minutes      3. Seconds      4. Exit\n\rSelect Segment: ");
                        scanf("%2u", &segment);
                        switch(segment)
                        {
                            case 1:
                                printf("\n\nSet Color for hours (max. 255) [Red:Green:Blue]: ");
                                scanf("%3u:%3u:%3u", &hour_setting.R, &hour_setting.G, &hour_setting.B);
                                break;

                            case 2:
                                printf("\n\nSet Color for minutes (max. 255) [Red:Green:Blue]: ");
                                scanf("%3u:%3u:%3u", &minute_setting.R, &minute_setting.G, &minute_setting.B);
                                break;

                            case 3:
                                printf("\n\nSet Color for seconds (max. 255) [Red:Green:Blue]: ");
                                scanf("%3u:%3u:%3u", &second_setting.R, &second_setting.G, &second_setting.B);
                                break;

                            case 4:
                                status_color = 0;
                                break;
                        }
                    } while (status_color);

                    break;

                    case 3:
                    do
                    {
                }
```



```
printf("\n\n\rBrightness (max. 31): ");
scanf("%2u", &brightness);

} while (brightness > 31);
break;

default:
status_settings = 0;
break;
}

} while (status_settings);

// Parameter übernehmen
hours.time = (unsigned char)hour;
minutes.time = (unsigned char)minute;
seconds.time = (unsigned char)second;

hours.R = (unsigned char)hour_setting.R;
hours.G = (unsigned char)hour_setting.G;
hours.B = (unsigned char)hour_setting.B;

minutes.R = (unsigned char)minute_setting.R;
minutes.G = (unsigned char)minute_setting.G;
minutes.B = (unsigned char)minute_setting.B;

seconds.R = (unsigned char)second_setting.R;
seconds.G = (unsigned char)second_setting.G;
seconds.B = (unsigned char)second_setting.B;

hours.intensity = (unsigned char)brightness;
minutes.intensity = (unsigned char)brightness;
seconds.intensity = (unsigned char)brightness;
break;
}
_delay_ms(1000);
}

port_init();           // Ports initialisieren
timer_init();          // Timer initialisieren
clock_init();
sei();                 // Interrupt starten
printf("\n\n");
while (1)
{
    clock_data(&hours, &minutes, &seconds);
    printf("\rHours: %2u Minutes: %2u Seconds: %2u", hours.time, minutes.time, seconds.time);
    _delay_ms(1);
}
}
```

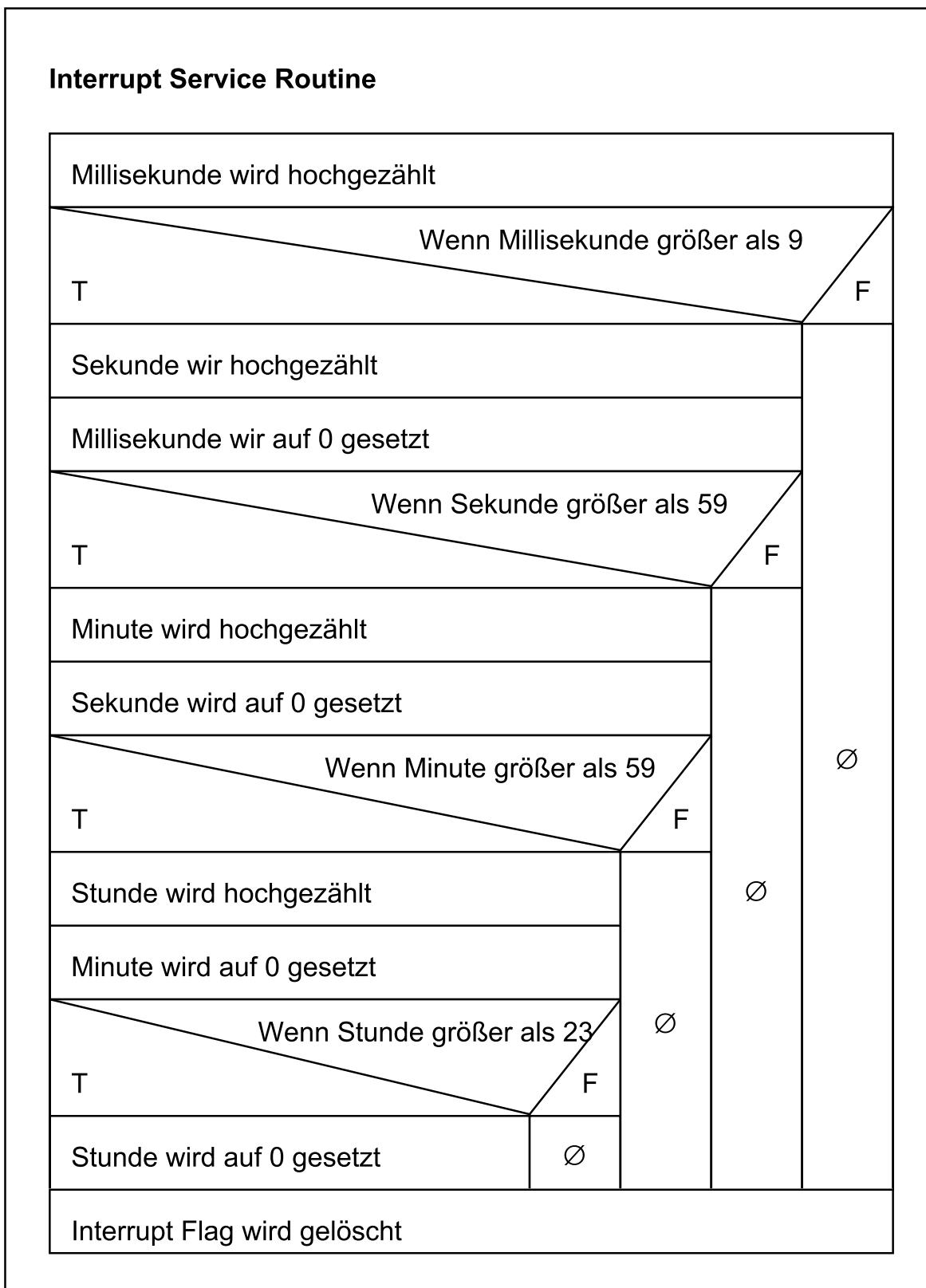
18.5.3 Beschreibung

Die Hauptfunktion ist für die Initialisierung der CPU, des Timers, der UART, der SPI und dem Takt zuständig. Man hat zudem die Möglichkeit, mithilfe einer Konsole (TeraTerm), ein Setup Menü aufzurufen, indem man innerhalb 10 Sekunden auf eine Taste drückt. Mit diesem Menü lässt sich die Zeit, die Farbe und die Helligkeit einstellen. Wenn die Einstellung festgelegt ist, wird sie von der Uhr übernommen und die Uhrzeit wird sowohl auf der Uhr, als auch auf der Konsole angezeigt.



18.6 Interrupt Service Routine

18.6.1 Struktogramm





18.6.2 Code

```
ISR(TCA0_OVF_vect)
{
    if((++miliseconds) > 9)          // Wenn die Millisekunde größer als 9 ist:
    {
        seconds.time++;              // Die Sekunde wird hochgezählt
        miliseconds = 0;             // Die Millisekunde wird auf 0 gesetzt

        if(seconds.time > 59)         // Wenn die Sekunde größer als 59 ist:
        {
            minutes.time++;          // Die Minute wird hochgezählt
            seconds.time = 0;         // Die Sekunde wird auf 0 gesetzt

            if(minutes.time > 59)     // Wenn die Minute größer als 59 ist:
            {
                hours.time++;        // Die Stunde wird hochgezählt
                minutes.time = 0;      // Die Minute wird auf 0 gesetzt

                if(hours.time > 23);   // Wenn die Stunde größer als 23 ist:
                {
                    hours.time=0;
                }
            }
        }
    }

    TCA0.SINGLE.INTFLAGS = TCA_SINGLE_OVF_bm; // Das Interrupt Flag wird am Ende jedes
                                                // Interrupts gelöscht
}
```

18.6.3 Beschreibung

Dieses Timer Overflow Interrupt wurde so konfiguriert, dass die ISR jede 100ms ausgelöst wird. Um die Zeit zählen zu können, muss bei jedem Aufruf die Variable „miliseconds“ hochgezählt werden. Wenn die Variable den Wert 9 übersteigt, wird er zurückgesetzt und die Variable „seconds.time“ wird um 1 hochgezählt. Sollte „seconds.time“ den Wert 59 übersteigen, wird die Variable auf 0 gesetzt und die Variable „hours.time“ wird um 1 hochgezählt. Wenn zu guter Letzt „hours.time“ den Wert 23 übersteigt, wird „hours.time“ auf 0 gesetzt.

Mit diesem Prinzip kann jede Uhr verarbeitet werden, egal ob sie mit LEDs angesteuert wird oder über eine 7 Segment Anzeige läuft.

Am Ende der ISR ist es notwendig das Interrupt Flag manuell zurückzusetzen.



18.7 Timer Setup / Initialisierung

18.7.1 Struktogramm

Timer Initialisierung

Die Periode wird Eingestellt

Timer Interrupt Control wird Konfiguriert

- Overflow Interrupt wird aktiviert

Timer Control B wird Konfiguriert

- Der Normale Modus wird Eingestellt

Timer Control A wird Konfiguriert

- Die Taktquelle wird eingestellt (sysclk / 64)
- Starte Timer



18.7.2 Code

```
void timer_init()
{
    TCA0.SINGLE.PER = 31250;
    TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm;
    TCA0.SINGLE.CTRLB = TCA_SINGLE_WGMODE_NORMAL_gc;
    TCA0.SINGLE.CTRLA = TCA_SINGLE_CLKSEL_DIV64_gc | TCA_SINGLE_ENABLE_bm;
}
```

18.7.3 Beschreibung

Um die Interrupt Service Routine für die Uhr nutzen zu können, muss der Timer richtig eingestellt werden.

Um das Aufrufen der ISR richtig einstellen zu können, muss der OCR-Wert berechnet werden.

Dies ist mit folgender Formel möglich:

$$TCA_{period} = \frac{time_{TCAlrq}(s)}{TCA_{prescaler}} - 1$$

Zuerst wurde versucht, die ISR jede Sekunde aufzurufen.

Weil sich das Ergebnis unter 2^{16} (65536) befinden muss, war dies leider nicht möglich. Das sind die Ergebnisse:

$$TCA_{period} = \frac{1(s) * 20MHz}{64} - 1 = 312499$$

$$TCA_{period} = \frac{1(s) * 20MHz}{256} - 1 = 78124$$

$$TCA_{period} = \frac{1(s) * 20MHz}{1024} - 1 = 19540,25$$

Letzen Endes haben wir uns für jede 100ms entschieden, da dieser Wert sich unter 2^{16} befindet und keine Kommastellen lieferte.

Das Ergebnis lautet wie folgt:

$$TCA_{period} = \frac{1 * 10^{-1}(s) * 20MHz}{64} - 1 = 31249$$

Damit die ISR auslöst, ist der Overflow Interrupt zu aktivieren.

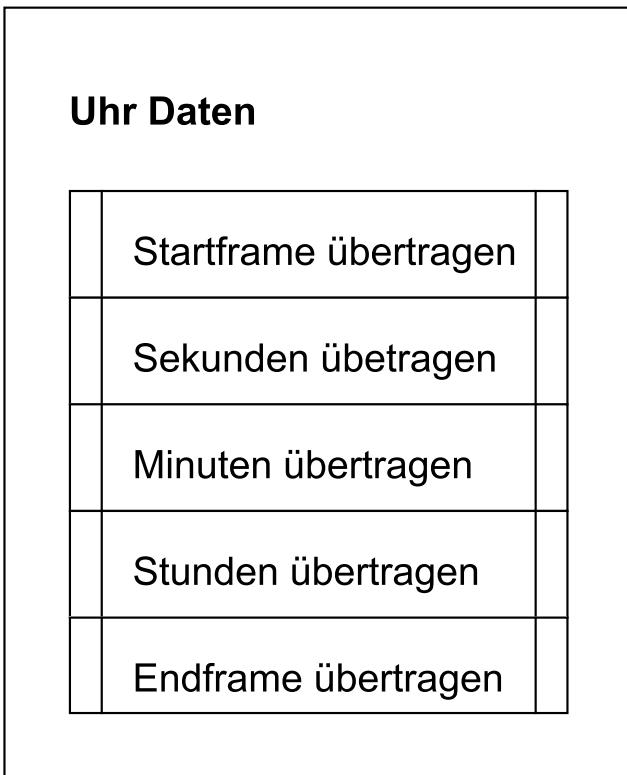
Der Timer wird im Normalen Modus betrieben.

Am Schluss wird der Vorteiler auf 64 eingestellt und der Timer startet.



18.8 Zeitverarbeitung

18.8.1 Struktogramm



18.8.2 Code

```
void clock_data(volatile Clock_time_t *hours, volatile Clock_time_t *minutes, volatile Clock_time_t *seconds)
{
    led_sof();

    // Sekunden
    clock_transmit(seconds);

    // Minuten
    clock_transmit(minutes);

    // Stunden
    clock_transmit(hours);

    led_eof();
}
```

18.8.3 Beschreibung

In diesem Unterprogramm findet Datentransfer der Uhr statt. Die Funktion übernimmt die Sekunden, Minuten und Stunden und übergibt sie der Funktion „clock_transmit“. Dies Überträgt die Uhrzeit auf das jeweilige Segment. Vor und nach den Sekunden-, Minuten- und Stundentransfer befindet sich das Start- und Endframe, damit die LEDs wissen, dass sie angesprochen werden.



18.9 Datentransfer

18.9.1 Struktogramm

Uhr Datentransfer

Zehnerziffer ergibt Zeit/10

Bsp. $16/10 = 1,6$

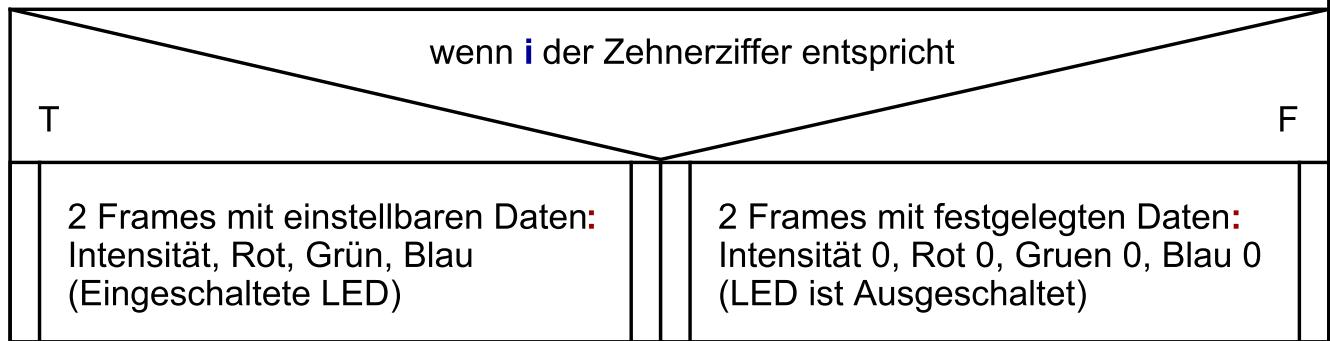
Die Kommzahl wird weggestrichen, die 1 bleibt übrig

Einserziffer ergibt Zeit **mod** 10

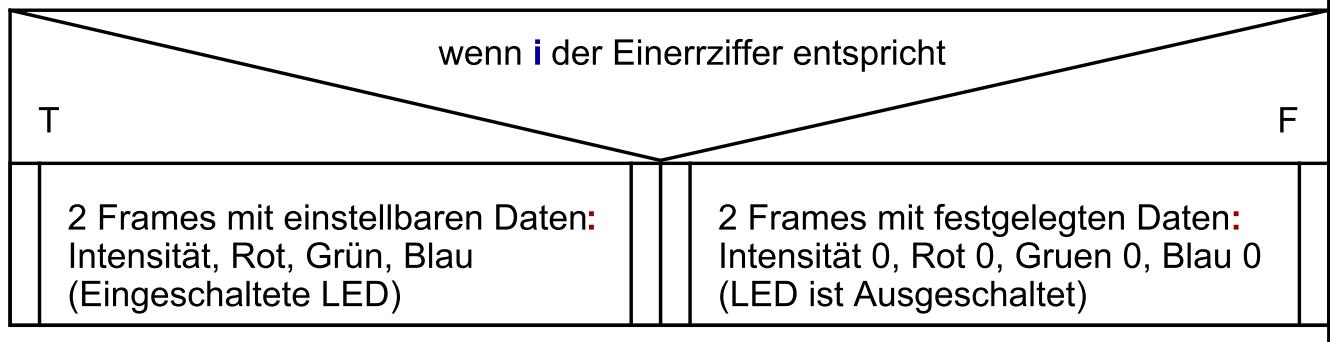
Bsp. $26 \bmod 10 = 6$

$26/10 = 2$ (Rest 6)

for **i** \leftarrow 9 to 0



for **i** \leftarrow 9 to 0





18.9.2 Code

```
void clock_transmit(volatile Clock_time_t *data)
{
    unsigned char upper_time= ((*data).time)/10;
    unsigned char lower_time = ((*data).time)%10;

    for(signed char i=0; i <= 9; i++)
    {
        if(i == lower_time)
        {
            led_time((*data).intensity, (*data).R, (*data).G, (*data).B);
            led_time((*data).intensity, (*data).R, (*data).G, (*data).B);
        }
        else
        {
            led_time(0x00, 0, 0, 0);
            led_time(0x00, 0, 0, 0);
        }
    }

    for(signed char i=0; i <= 9; i++)
    {
        if(i == upper_time)
        {
            led_time((*data).intensity, (*data).R, (*data).G, (*data).B);
            led_time((*data).intensity, (*data).R, (*data).G, (*data).B);
        }
        else
        {
            led_time(0x00, 0, 0, 0);
            led_time(0x00, 0, 0, 0);
        }
    }
}
```



18.9.3 Beschreibung

Dieser Teil des Codes ist für die Aktivierung der jeweiligen Ziffer auf dem Segment verantwortlich.

Um die Zehnerziffer beleuchten zu können, wird die Zeit durch 10 geteilt.

Für die Beleuchtung der Einser Ziffer wird die Zeit mit 10 Modulo genommen. Das bedeutet, dass die Zahl durch 10 dividiert wird und die daraus resultierende Kommazahl das Ergebnis darstellt.

Das Prinzip der Verarbeitung lässt sich mit folgender Rechnung verdeutlichen:

Einser Ziffer ergibt Zeit Modulo 10

Bsp. 26 Modulo 10 = 6

$26/10 = 2$ (Rest 6)

Da wir die Zahl 26 anzeigen möchten, muss bei der Einser Ziffer die 6 beleuchtet werden.

Zehnerziffer ergibt Zeit/10

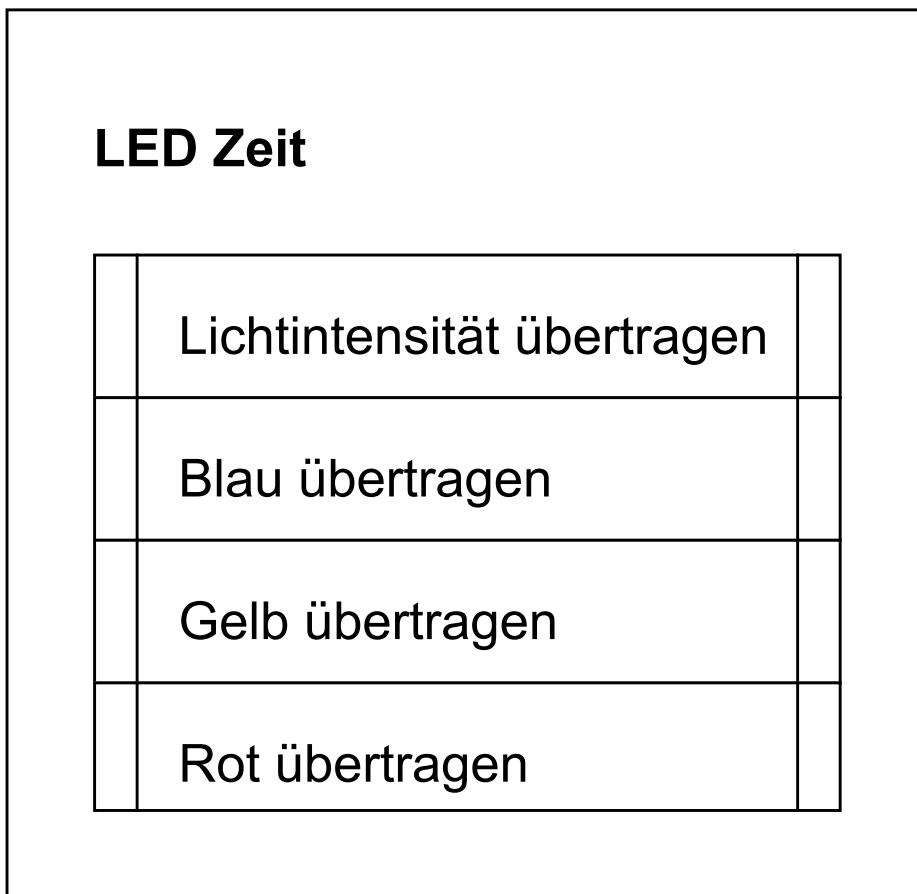
Bsp. $16/10 = 1,6$

Da die Zahl 16 angezeigt werden soll, muss bei der Zehner Ziffer die Zahl 1 beleuchtet werden. Da Integers benutzt werden, wird die Kommazahl weggestrichen.



18.10 LED-Zeit

18.10.1 Struktogramm



18.10.2 Code

```
void led_time(unsigned char intensity, unsigned char r, unsigned char g, unsigned char b)
{
    spi_transfer(0xE0 | intensity);
    spi_transfer(b);
    spi_transfer(g);
    spi_transfer(r);
}
```

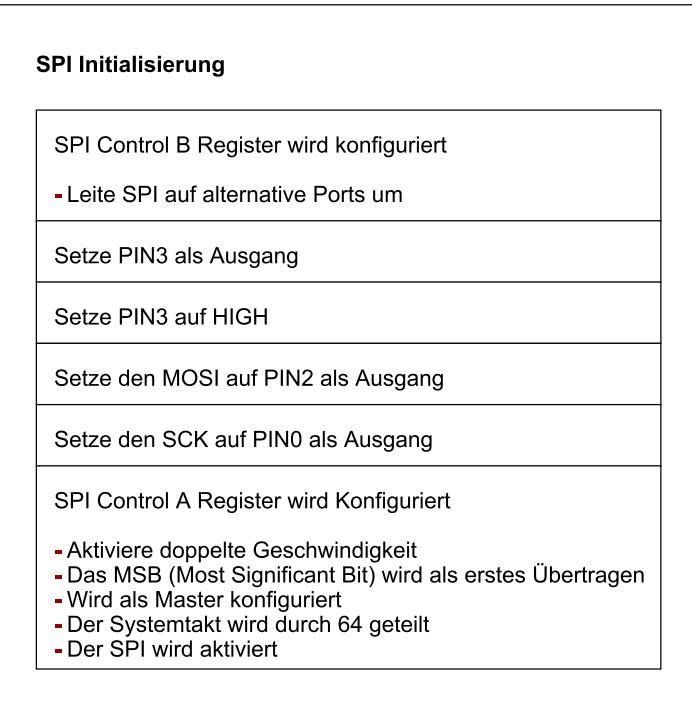
18.10.3 Beschreibung

In dieser Funktion werden die Datenframes an die LEDs übertragen. Diese sind notwendig, um den LEDs mitzuteilen mit welcher Intensität und Farbe sie leuchten sollen.



18.11 SPI-Initialisierung

18.11.1 Struktogramm



18.11.2 Code

```
void spi_init()
{
    PORTC.DIR |= PIN3_bm;
    PORTC.OUT |= 0x08;
    PORTMUX.CTRLB |= PORTMUX_SPI0_ALTERNATE_gc;
    PORTC.DIR |= PIN2_bm;           /* Set MOSI pin direction to output */
    PORTC.DIR |= PIN0_bm;          /* Set SCK pin direction to output */

    SPI0.CTRLA = SPI_CLK2X_bm
    | SPI_ENABLE_bm;              /* Enable module */
    // | SPI_DORD_bm;
    | SPI_MASTER_bm;             /* SPI module in Master mode */
    | SPI_PRESC_DIV64_gc;         /* System Clock divided by 64 */
}
```

18.11.3 Beschreibung

Da PC2 (PORTC / Pin 2) und PC0 (PORTC / Pin 0) alternative SPI-Pins sind, müssen diese manuell eingestellt werden. PC3 muss als Ausgang auf High gesetzt sein, damit der Slave Select nicht im Floating-Zustand verweilt und den SPI-Master Mode unterbricht.

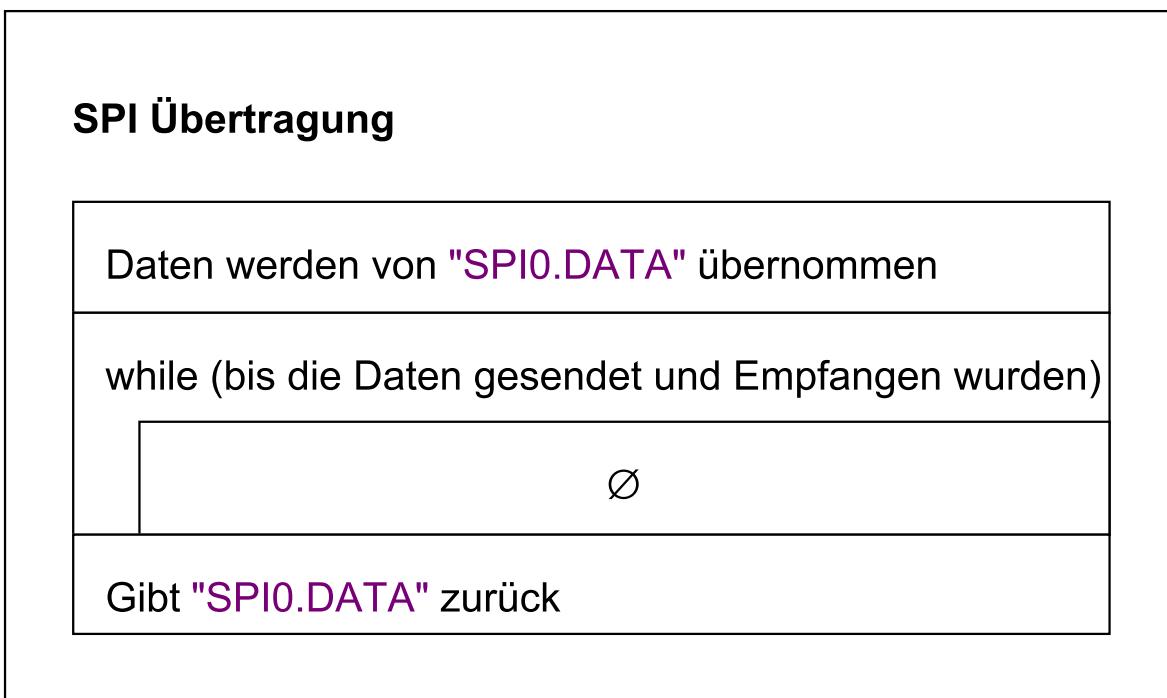
Für die Uhr werden nur MOSI (Master Output / Slave Input) und SCK (Taktleitung) verwendet.

Die Geschwindigkeit des Taktes vom SPI wird verdoppelt, der SPI wird als Master und der Prescaler wird auf 64 eingestellt. Am Schluss wird das SPI-Modul aktiviert.



18.12 SPI-Übertragung

18.12.1 Struktogramm



18.12.2 Code

```
unsigned char spi_transfer(unsigned char data)
{
    SPI0.DATA = data;
    while (!(SPI0.INTFLAGS & SPI_IF_bm)); /* waits until data is exchanged*/

    SPI0.INTFLAGS = SPI_IF_bm;

    return SPI0.DATA;
}
```

18.12.3 Beschreibung

Der Code für die SPI-Übertragung wurde vom Datenblatt „TB3215-Getting-Started-with-SPI-90003215A.pdf“ übernommen und befindet sich auf Seite 17.

$$\frac{\text{Systemtakt}}{\text{Prescaler}} = \text{SPI Takt}$$

$$\frac{20\text{MHz}}{64} = 312,5\text{KHz}$$



19 Testaufbauten:

19.1 Testaufbau (01.02.2023)

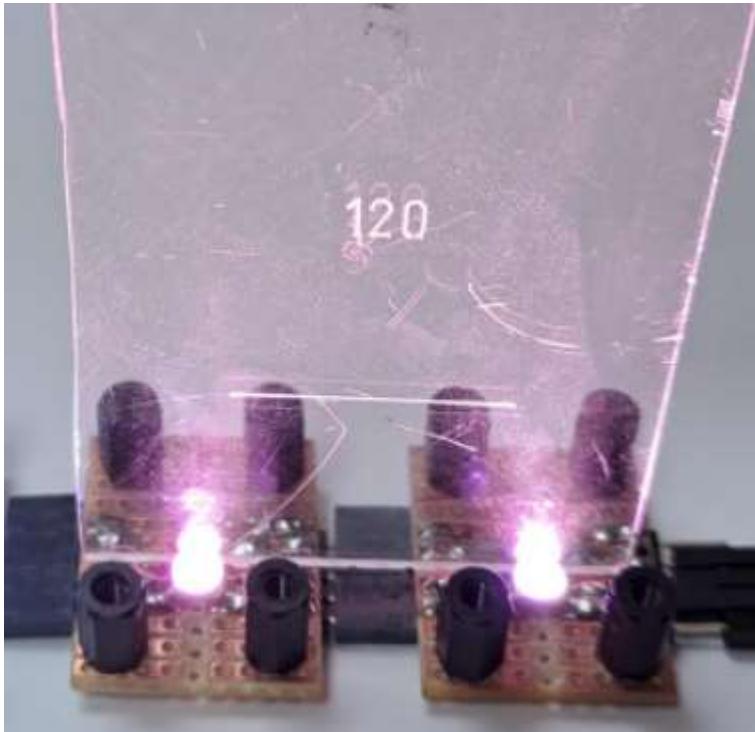


Abbildung 30: Testaufbau 01.02.2023

Am 01. Februar 2023 wurde eine Zahl ins Plexiglas eingraviert, um zu ermitteln, ob sich die Zahl erkennen lässt. Die Lichtausbreitung hat dafür gesorgt, dass sich die Zahl problemlos erkennen lässt. Allerdings muss man die Zahl durch 9 Plexigläser erkennen zu können.

19.2 Testaufbau (03.02.2023)

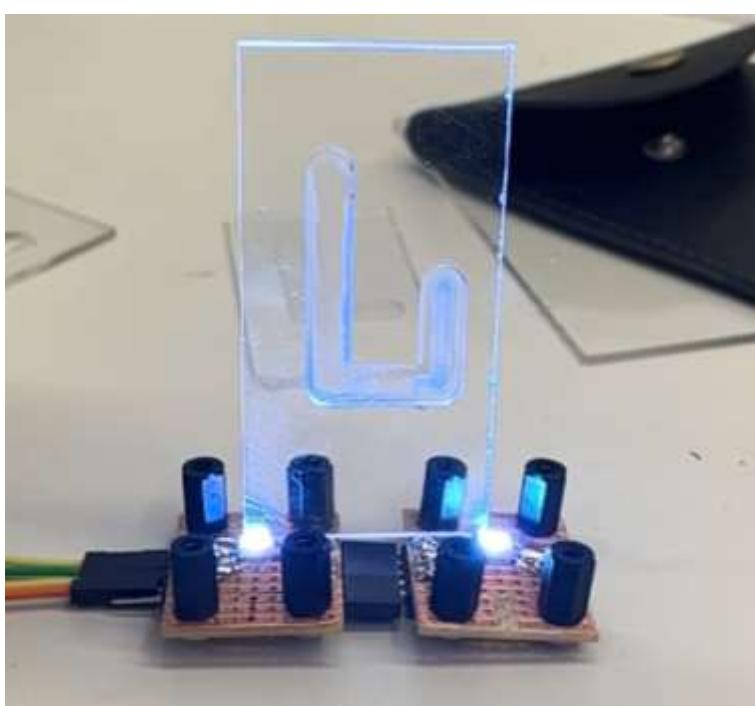


Abbildung 31: Testaufbau 03.02.2023

Am 03. Februar 2023 wurde die Lichtausbreitung von der APA102c getestet.

Das Plexiglas wurde nur provisorisch mit vertikalen und horizontalen Bohrungen gefräst.

Man sieht, dass sich das Licht gut ausbreitet, jedoch lässt sich die Fräse rung nicht gut genug erkennen.

Zusätzlich ist das Zeichen nicht transparent, sondern milchig. Das liegt daran, dass nicht genug Öl während des Fräsvorganges verwendet wurde.



19.3 Testaufbau/Inbetriebnahme der ersten Segmentplatine (22.02.2023)

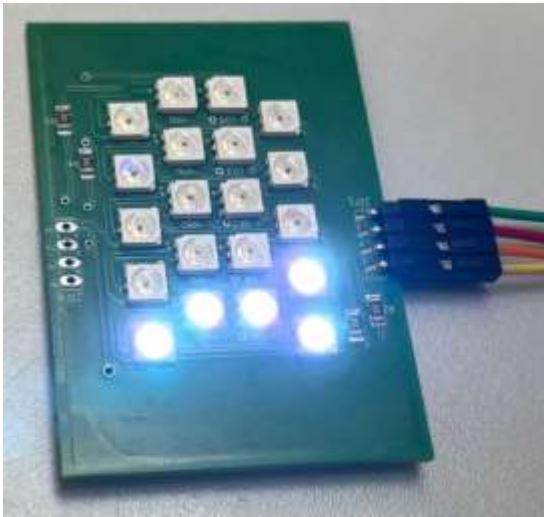


Abbildung 32: Testaufbau 22.03.2023

Die erste Segmentplatine wurde nach Prüfung in Betrieb genommen. Mithilfe eines Testprogramms wurde jede einzelne LED eingeschalten, um deren Funktion zu testen. Die Inbetriebnahme war erfolgreich und die Platine funktionierte einwandfrei.

19.4 Testaufbau (22.03.2023)



Auf dem Bild lässt sich das erste Segment, das aufgebaut wurde, erkennen. Die Passgenauigkeit von der Einstechplatte und den Plexigläsern wurde geprüft. Anschließend wurde der Segmentteil auf der Segmentplatine provisorisch positioniert und begutachtet, ob die Zahlen angezeigt werden.

Da die Plexigläser noch nicht transparent sind, weil das Öl die Transparenz von den Plexigläsern verschwinden lässt, wurden diese entsprechend gereinigt.

Abbildung 33: Testaufbau 22.03.2023



19.5 Testaufbau (03.03.2023)

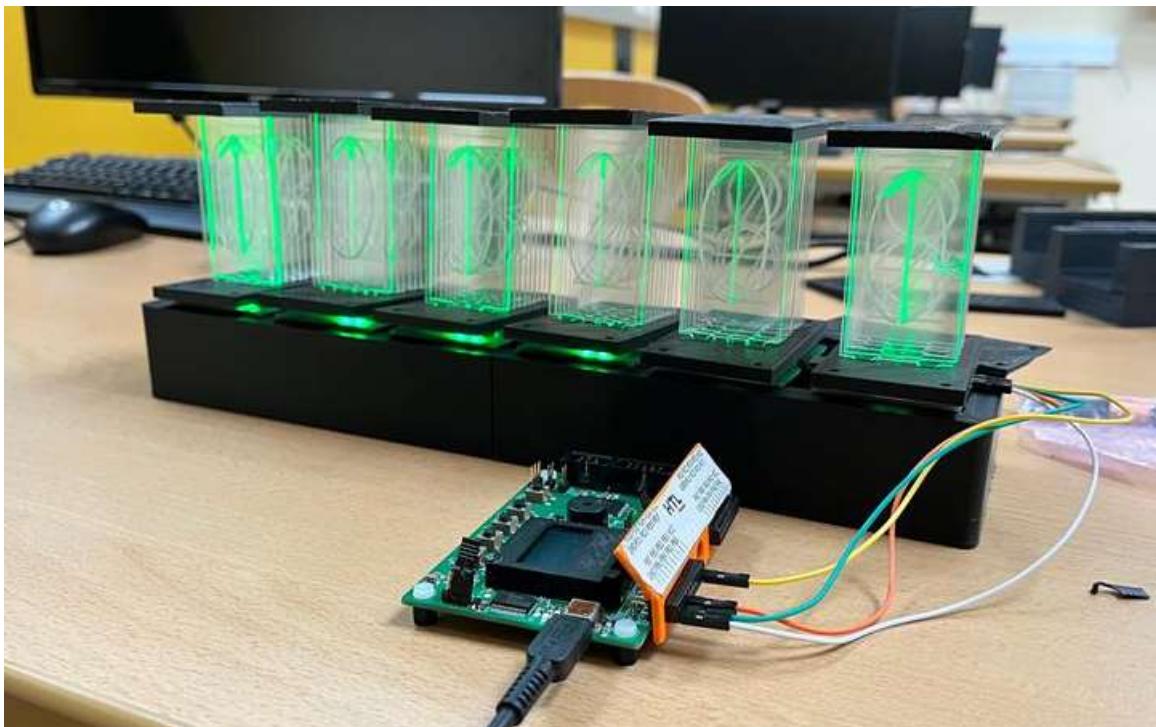


Abbildung 34: Testaufbau 03.03.2023

Die Lixie-Uhr wurde das erste Mal am 03.03.2023 mit allen Segmenten getestet. Die Segmente wurden noch nicht fixiert, da einige Änderungen am Gehäuse noch durchzuführen sind. Die Uhr lässt sich über SPI steuern. Allerdings funktioniert die Ansteuerung über UART nicht, da die Verbindung zwischen dem USB-/UART Wandler und dem Mikrocontroller fehlerhaft ist.

19.6 Zwischenstand (08.03.2023)

Es wurde in der Gruppe entschieden, dass die Masterplatine erneut bestellt wird. Grund dafür ist, dass TxD und RxD des USB/UART Wandlers nicht richtig mit dem Mikrocontroller verbunden sind. Am Anfang wurde versucht die Verbindungen mithilfe von 2 Drähten umzuverdrahten. Allerdings haben sich die Lötstellen aufgelöst und ein IC wurde dabei zerstört. Beim neuen Layout wurden Verbesserungen, wie die RESET-Bedingung und die richtigen Verbindungen von TxD und RxD, durchgeführt.



19.7 Der endgültige Aufbau (17.03.2023)



Abbildung 35: Testaufbau: Lixie-Uhr

Am 17.03.2023 erfolgte der Endgültige Aufbau der Lixie-Uhr. Nachdem die Masterplatine nachgeliefert und bestückt wurde, funktioniert die Verbindung zwischen uC und USB/UART Wandler. Ebenso der RESET-Knopf, allerdings ist, um den Knopf zu verwenden, die UPDI-Funktion vom Mikrocontroller zu deaktivieren. Beim Aufbau wurde zuallererst die Masterplatine festgemacht und anschließend die Segmentplatinen. Die Masterplatine wurde mit Hilfe von Male-to-Female-Steckern mit der Segmentplatine verbunden.



20 Datenblätter

20.1 Apa102c:



iPixel LED
Shiji Lighting

APA102C

RGB Full Color LED control IC

● (General Description)

APA102C for the three-color RGB LED dimming control string Then IC , using the CMOS process , providing three-color RGB LED output driver to adjust the output with 256 gray-scale and 32 brightness adjustment APA102 with two-output WAY ,the CLK signal by synchronization , so that the crystal cascadePiece of output movements synchronized.

● (application)

LED lamps
Large LED screen
LED billboards

● (Features)

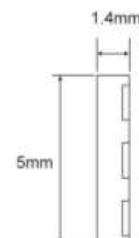
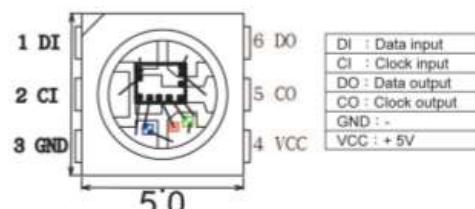
CMOS process , low voltage , low power consumption
Synchronous of two-lane
Choose positive output or negative output RGB tri-color LED output, 8 Bit (256 level) color Set, 5Bit (32 level) brightness adjustment
Built-20mA constant current output

With self-detection signal
Built-in support for continuous oscillation PWM output can be maintained Static Screen

PRODUCT SPECIFICATIONS

Model number	Color	Millicandela	refresh rate	Applied voltage	Power consumption	View angle	weight (g)	Dimensions(mm)	Operating temperature
SUPER LED	Full Color 16777216	R 500~650 mcd G 370~530 mcd B 120~165 mcd	400 cycle	5VDC	0.2W (MAX:1W)	H:160	0.1	5x5x1.4	-40°C~70°C

PHYSICAL DIMENSIONS



Add:5F A2 Huafeng Zhenbao Industrial Park,BeiHuan Rd ,Shiyan,Baoan,Shenzhen,China. 518108

Tel: 0086-755-23125058 Fax:0086-755-23125658 Web:www.shiji-led.com

E-mail:contact@shiji-led.com

Web:www.ipixelled.cn

1



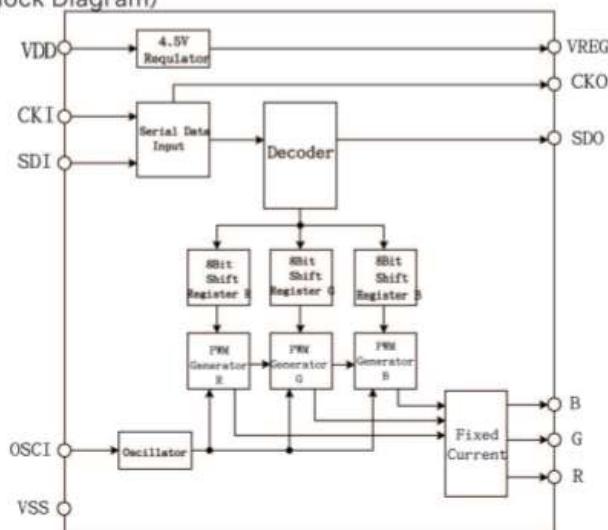
iPixel LED
Shiji Lighting

APA102C

● 腳位說明 (Pin Description)

NO.	PIN NAME	I/O	FUNCTION
1	VDD	P	Power is terminal
2	VREG	O	4.5V regulator output
3	CKO	O	Series with the output clock signal
4	SDO	O	Series with the output data
5	VEN	I	Self-test function selection
6	CSEL	I	Invert the clock signal cascade
7	POLAR	I	Positive and negative output options
8	OSCI	I	Oscillator input
9	SDI	I	Series with the input data
10	CKI	I	Series with the input clock signal
11	REXT	I	Constant current source to adjust side
12	VSS	P	Power supply negative terminal
13	G	O	Green LED output
14	R	O	Red LED output
15	B	O	Blue LED output

● 功能方塊圖 (Block Diagram)



Add:5F A2 Huafeng Zhenbao Industrial Park,BeiHuan Rd ,Shiyan,Baoan,Shenzhen,China. 518108

Tel: 0086-755-23125058 Fax:0086-755-23125658 Web:www.shiji-led.com

E-mail: contact@shiji-led.com

2

Web:www.ipixelled.cn



● 最大額定範圍 (Absolute Maximum Ratings)

Supply Voltage ——————0.3V to 6.0V

Input Voltage ——————VSS-0.3 to VDD+0.3

Operating Temperature ——————40°C to 70°C

Storage Temperature ——————50°C to 125°C

Note: Stress above those listed may cause permanent damage to the devices

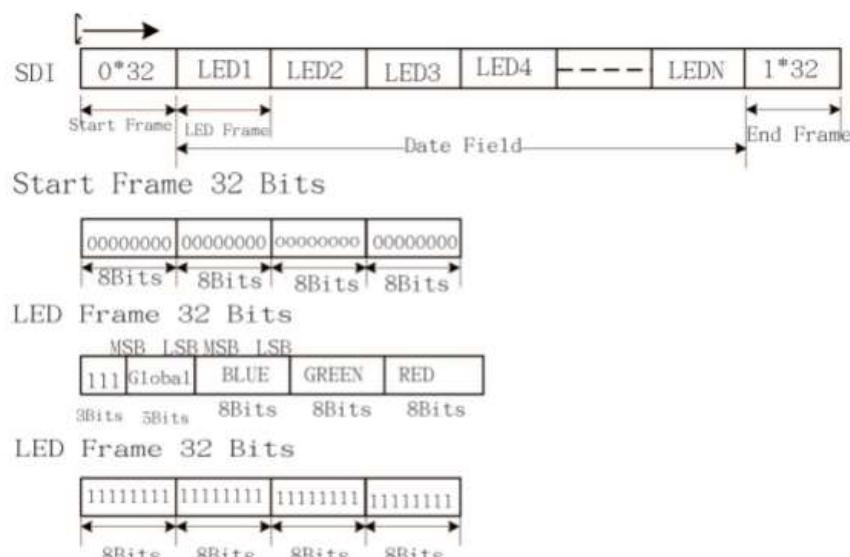
● 電氣特性 (Electrical Characteristics)

Symbol	Parameter	Condition	Min.	Typ.	Max	Units
VDD	Supply Voltage			5.0	5.5	V
VIH	Input High Voltage		0.7VDD		VDD+0.3	V
VH	Input Low Voltage		Vss-0.3		0.3VDD	V
LOL	Sink Current Voltage (RGB)	@VDD=5V, VOL>1V	22.5	24.5	26.5	mA
RIN	Pull High	@VDD=5V		570		kΩ
VREG	Regulator Voltage (VREG)	@VDD>5V	4.4	4.5	4.7	V
FOSC	Oscillator Frequency		800		1200	KHz

● 功能說明 (function description)

(1) .cascading data structure

Tabdem N-LED



Add:5F A2 Huafeng Zhenbao Industrial Park,BeiHuan Rd ,Shiyan,Baoan,Shenzhen,China. 518108

Tel: 0086-755-23125058 Fax:0086-755-23125658 Web:www.shiji-led.com

E-mail: contact@shiji-led.com

3

Web:www.ipixelled.cn



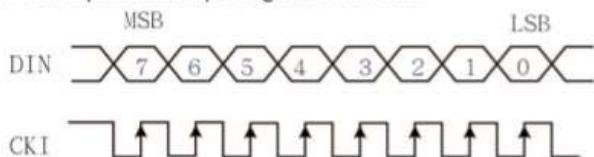
iPixel LED
Shiji Lighting

APA102C

Global bit : 5-bit (32 level) brightness setting , while controlling R, G, B three-color constant current output value , if set the Global bit for the 10000 (16/31) is the output current is half again the original PWM settings.

DATA MSB↔LSB	Driving Current
00000	0/31
00001	1/31
00010	2/31
...	
11110	30/31
11111	31/31(max)

PWM input and output signals Relations



Data MSB—	Duty Cycle
00000000	0/256(min)
00000001	1/256
00000010	2/256
...	
11111101	253/256
11111110	254/256
11111111	255/256(max)

2).The number of pixels per second sent to CKI frequency (FCKI) minus the Start Frame bit divided by the number 40 the number of LED Frame bit 32, if CKI frequency (FCKI) to 512KHz, the pixel number (512000–40) /32=15998, if the 50 second update Views can be connected in series LED number 15998/50=319.To increase the number of cascaded IC CKI frequency to be increased.

(3).POLAR to empty , R, G, B for the negative output; POLAR access VSS, R, G, B is positive output.

(4).VEN: Self-detection

Data Field to the middle of 3bit were B, G, R in the MSB of the opposite phase , otherwise regarded as invalid data. VEN close to empty when the self-detection: when VEN VSS then activated self-detection.

(5).CSEL to empty when the CKO and CKI RP :CSEL connected with VSS when the CKO compared with CKI.

Add:5F A2 Huafeng Zhenbao Industrial Park,BeiHuan Rd ,Shiyan,Baoan,Shenzhen,China. 518108

Tel: 0086-755-23125058 Fax:0086-755-23125658 Web:www.shiji-led.com

E-mail: contact@shiji-led.com

Web:www.ipixelled.cn

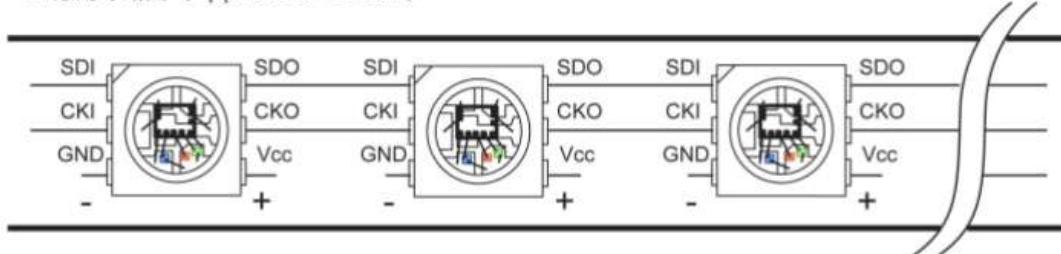
4



iPixel LED
Shiji Lighting

APA102

● 應用線路圖 (Application Circuit)



Add:5F A2 Huafeng Zhenbao Industrial Park,BeiHuan Rd ,Shiyan,Baoan,Shenzhen,China. 518108

Tel: 0086-755-23125058 Fax:0086-755-23125658 Web:www.shiji-led.com

E-mail: contact@shiji-led.com

Web:www.ipixelled.cn

5



20.2 FT232RL (USB-/UART Wandler):



FT232R USB UART IC Datasheet Version 2.16

Document No.: FT_000053 Clearance No.: FTDI# 38

1 Typical Applications

- USB to RS232/RS422/RS485 Converters
- Upgrading Legacy Peripherals to USB
- Cellular and Cordless Phone USB data transfer cables and interfaces
- Interfacing MCU/PLD/FPGA based designs to USB
- USB Audio and Low Bandwidth Video data transfer
- PDA to USB data transfer
- USB Smart Card Readers
- USB Instrumentation
- USB Industrial Control
- USB MP3 Player Interface
- USB FLASH Card Reader and Writers
- Set Top Box PC - USB interface
- USB Digital Camera Interface
- USB Hardware Modems
- USB Wireless Modems
- USB Bar Code Readers
- USB Software and Hardware Encryption Dongles

1.1 Driver Support

Royalty free VIRTUAL COM PORT (VCP) DRIVERS for...

- Windows 10 32,64-bit
- Windows 8/8.1 32,64-bit
- Windows 7 32,64-bit
- Windows Vista and Vista 64-bit
- Windows XP and XP 64-bit
- Windows 98, 98SE, ME, 2000, Server 2003, XP, Server 2008 and server 2012 R2
- Windows XP Embedded
- Windows CE 4.2, 5.0 and 6.0
- Mac OS 8/9, OS-X
- Linux 2.4 and greater

Royalty free D2XX Direct Drivers (USB Drivers + DLL S/W Interface)

- Windows 10 32,64-bit
- Windows 8/8.1 32,64-bit
- Windows 7 32,64-bit
- Windows Vista and Vista 64-bit
- Windows XP and XP 64-bit
- Windows 98, 98SE, ME, 2000, Server 2003, XP, Server 2008 and server 2012 R2
- Windows XP Embedded
- Windows CE 4.2, 5.0 and 6.0
- Linux 2.4 and greater
- Android(J2xx)

The drivers listed above are all available to download for free from FTDI website (www.ftdichip.com). Various 3rd party drivers are also available for other operating systems - see FTDI website (www.ftdichip.com) for details.

For driver installation, please refer to <http://www.ftdichip.com/Documents/InstallGuides.htm>



2 FT232R Block Diagram

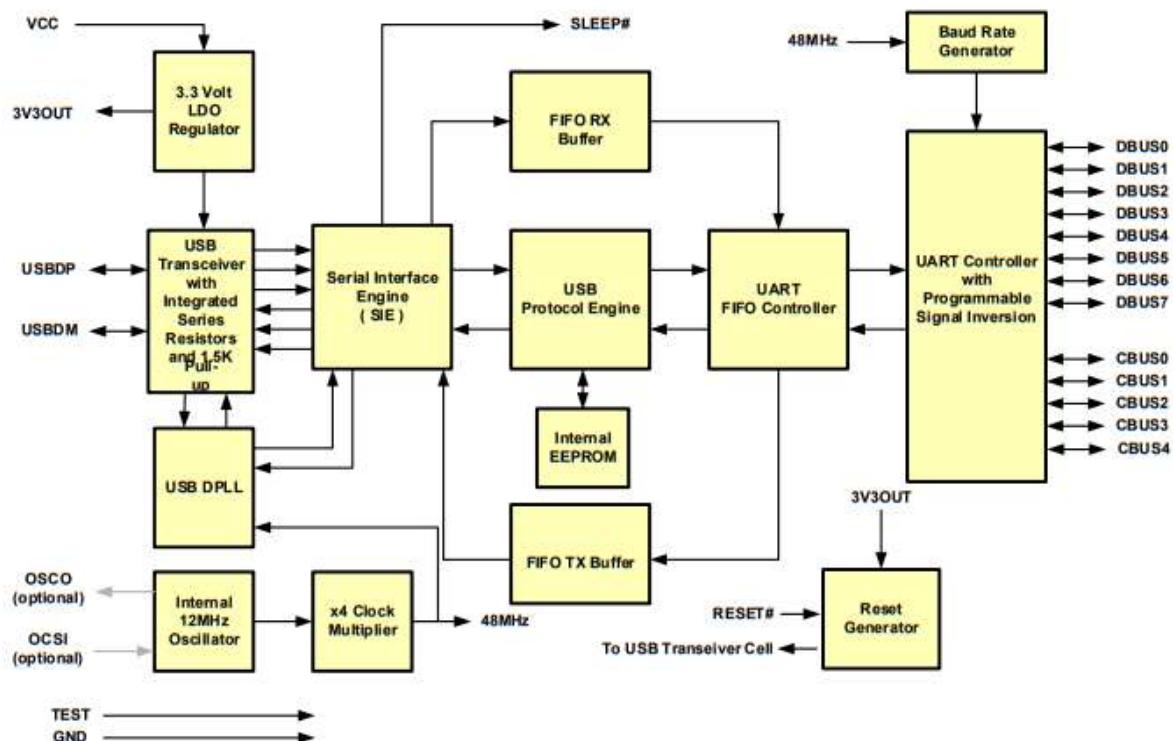


Figure 2.1 FT232R Block Diagram

For a description of each function please refer to Section 4.



3 Device Pin Out and Signal Description

3.1 28-LD SSOP Package

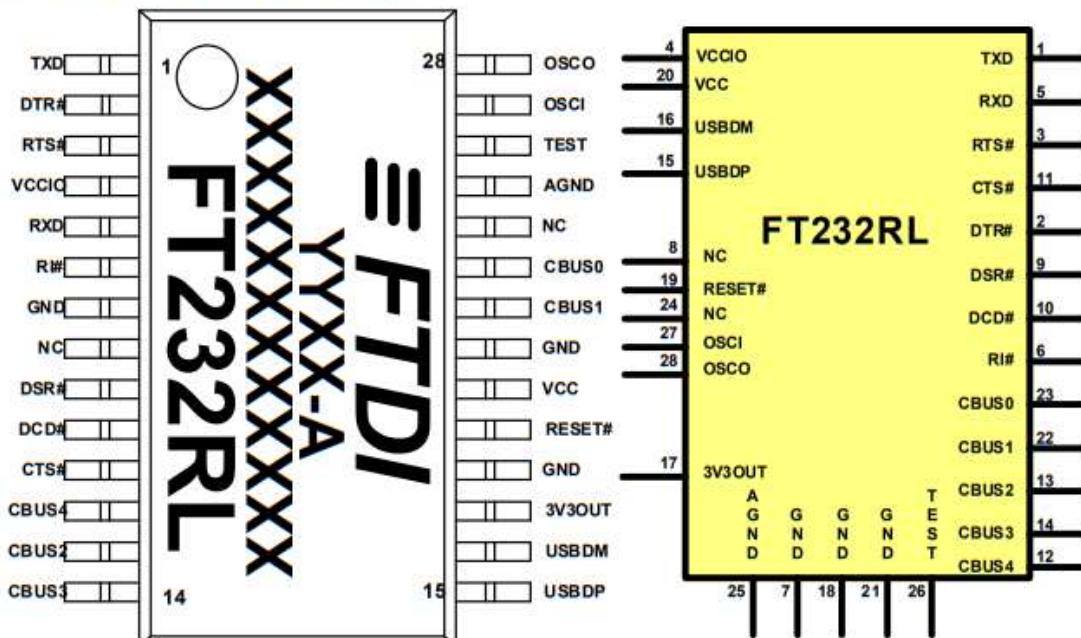


Figure 3.1 SSOP Package Pin Out and Schematic Symbol

3.2 SSOP Package Pin Out Description

Note: The convention used throughout this document for active low signals is the signal name followed by #.

Pin No.	Name	Type	Description
15	USBDP	I/O	USB Data Signal Plus, incorporating internal series resistor and 1.5kΩ pull up resistor to 3.3V.
16	USBDM	I/O	USB Data Signal Minus, incorporating internal series resistor.

Table 3.1 USB Interface Group

Pin No.	Name	Type	Description
4	VCCIO	PWR	+1.8V to +5.25V supply to the UART Interface and CBUS group pins (1...3, 5, 6, 9...14, 22, 23). In USB bus powered designs connect this pin to 3V3OUT pin to drive out at +3.3V levels, or connect to VCC to drive out at 5V CMOS level. This pin can also be supplied with an external +1.8V to +2.8V supply in order to drive outputs at lower levels. It should be noted that in this case this supply should originate from the same source as the supply to VCC. This means that in bus powered designs a regulator which is supplied by the +5V on the USB bus should be used.
7, 18, 21	GND	PWR	Device ground supply pins
17	3V3OUT	Output	+3.3V output from integrated LDO regulator. This pin should be decoupled to ground using a 100nF capacitor. The main use of this pin is to provide the internal +3.3V supply to the USB transceiver cell and the internal 1.5kΩ pull up resistor on USBDP. Up to 50mA can be drawn from



7.4 USB to MCU UART Interface

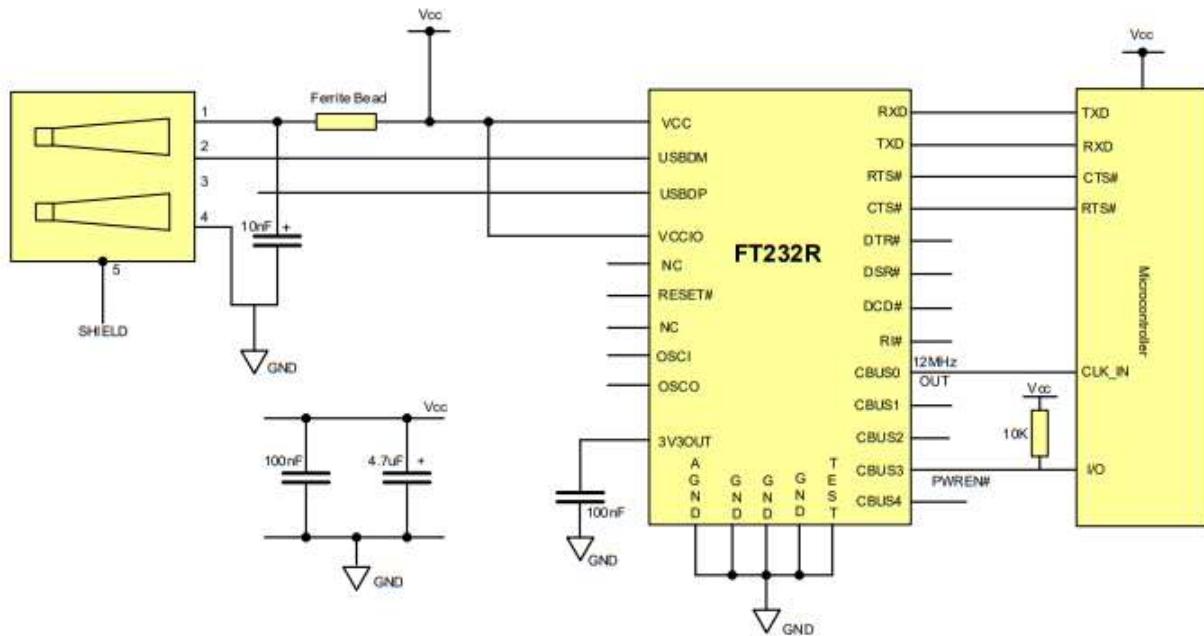


Figure 7.4 USB to MCU UART Interface

An example of using the FT232R as a USB to Microcontroller (MCU) UART interface is shown in Figure 7.4. In this application the FT232R uses TXD and RXD for transmission and reception of data, and RTS# / CTS# signals for hardware handshaking. Also in this example CBUS0 has been configured as a 12MHz output to clock the MCU.

Optionally, RI# could be connected to another I/O pin on the MCU and used to wake up the USB host controller from suspend mode. If the MCU is handling power management functions, then a CBUS pin can be configured as PWREN# and would also be connected to an I/O pin of the MCU.



9 Package Parameters

The FT232R is available in two different packages. The FT232RL is the SSOP-28 option and the FT232RQ is the QFN-32 package option. The solder reflow profile for both packages is described in Section 9.3.

9.1 SSOP-28 Package Dimensions

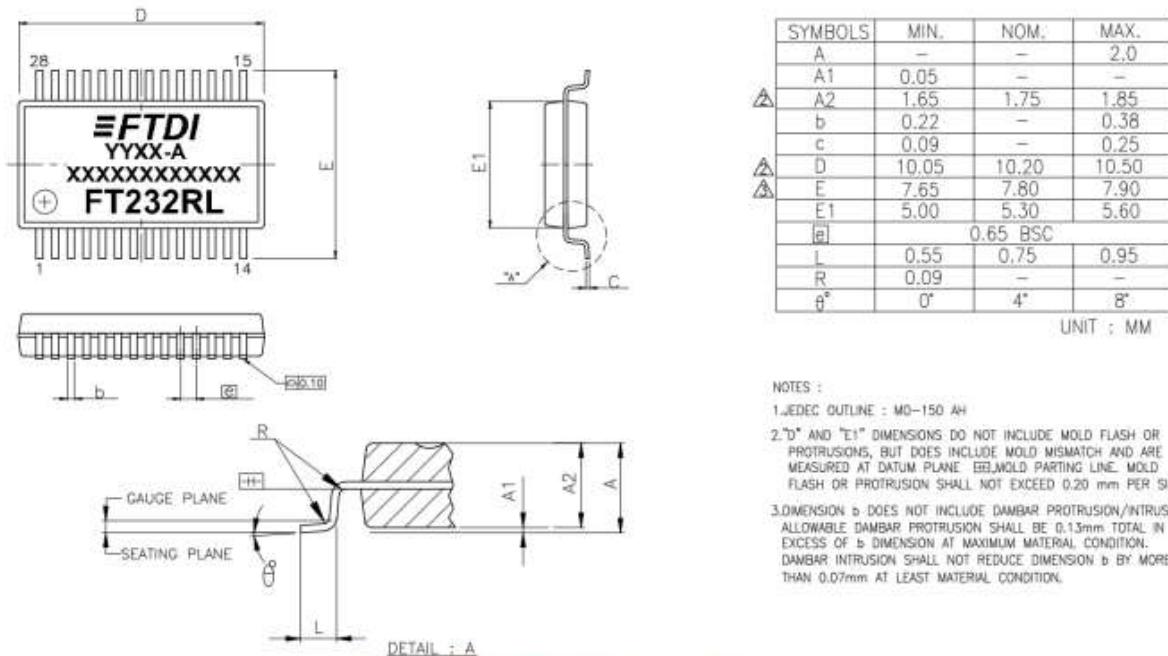


Figure 9.1 SSOP-28 Package Dimensions

The FT232RL is supplied in a RoHS compliant 28 pin SSOP package. The package is lead (Pb) free and uses a 'green' compound. The package is fully compliant with European Union directive 2002/95/EC.

This package is nominally 5.30mm x 10.20mm body (7.80mm x 10.20mm including pins). The pins are on a 0.65 mm pitch. The above mechanical drawing shows the SSOP-28 package.

All dimensions are in millimetres.

The date code format is **YYXX-A** where XX = 2 digit week number, YY = 2 digit year number, A = single letter corresponding to the revision of the device (e.g. A or B or C).

The code **XXXXXXXXXXXX** is the manufacturing LOT code. This only applies to devices manufactured after April 2009.



20.3 Attiny1606:

20.4.2. Allgemein



Introduction

The ATtiny804/806/807/1604/1606/1607 are members of the tinyAVR® 0-series of microcontrollers, using the AVR® processor with hardware multiplier, running at up to 20 MHz, with 8/16 KB Flash, 512/1024 bytes of SRAM, and 128/256 bytes of EEPROM in a 14-, 20-, or 24-pin package. The tinyAVR® 0-series uses the latest technologies with a flexible, low-power architecture, including Event System, accurate analog features, and Core Independent Peripherals (CIPs).



Attention: This data sheet is valid for industrial qualified devices.

Features

- CPU
 - AVR® CPU
 - Running at up to 20 MHz
 - Single-cycle I/O access
 - Two-level interrupt controller
 - Two-cycle hardware multiplier
- Memories
 - 8/16 KB In-system self-programmable Flash memory
 - 128/256 bytes EEPROM
 - 512/1024 bytes SRAM
 - Write/erase endurance:
 - Flash 10,000 cycles
 - EEPROM 100,000 cycles
 - Data retention:
 - 40 years at 55°C
- System
 - Power-on Reset (POR)
 - Brown-out Detector (BOD)
 - Clock options:
 - 16/20 MHz low-power internal RC oscillator
 - 32.768 kHz Ultra Low-Power (ULP) internal RC oscillator
 - External clock input
 - Single-Pin Unified Program and Debug Interface (UPDI)
 - Three sleep modes:
 - Idle with all peripherals running for immediate wake-up
 - Standby



ATtiny804/806/807/1604/1606/1607

- Configurable operation of selected peripherals
- Power-Down with full data retention
- Peripherals
 - One 16-bit Timer/Counter type A (TCA) with a dedicated period register and three compare channels
 - One 16-bit Timer/Counter type B (TCB) with input capture
 - One 16-bit Real-Time Counter (RTC) running from an external clock or internal RC oscillator
 - Watchdog Timer (WDT) with Window mode, with a separate on-chip oscillator
 - One USART with fractional baud rate generator, auto-baud, and start-of-frame detection
 - One host/client Serial Peripheral Interface (SPI)
 - One Two-Wire Interface (TWI) with dual address match
 - Philips I²C compatible
 - Standard mode (Sm, 100 kHz)
 - Fast mode (Fm, 400 kHz)
 - Fast mode plus (Fm+, 1 MHz)
 - One Analog Comparator (AC) with a low propagation delay
 - One 10-bit 115 kspS Analog-to-Digital Converter (ADC)
 - Multiple voltage references (V_{REF}):
 - 0.55V
 - 1.1V
 - 1.5V
 - 2.5V
 - 4.3V
 - Event System (EVSYS) for CPU independent and predictable inter-peripheral signaling
 - Configurable Custom Logic (CCL) with two programmable look-up tables
 - Automated CRC memory scan
 - External interrupt on all general purpose pins
- I/O and Packages:
 - Up to 22 programmable I/O lines
 - 14-pin SOIC150
 - 20-pin SOIC300
 - 20-pin VQFN 3x3 mm
 - 24-pin VQFN 4x4 mm
- Temperature Ranges:
 - -40°C to 105°C
 - -40°C to 125°C
- Speed Grades:
 - 0-5 MHz @ 1.8V – 5.5V
 - 0-10 MHz @ 2.7V – 5.5V
 - 0-20 MHz @ 4.5V – 5.5V



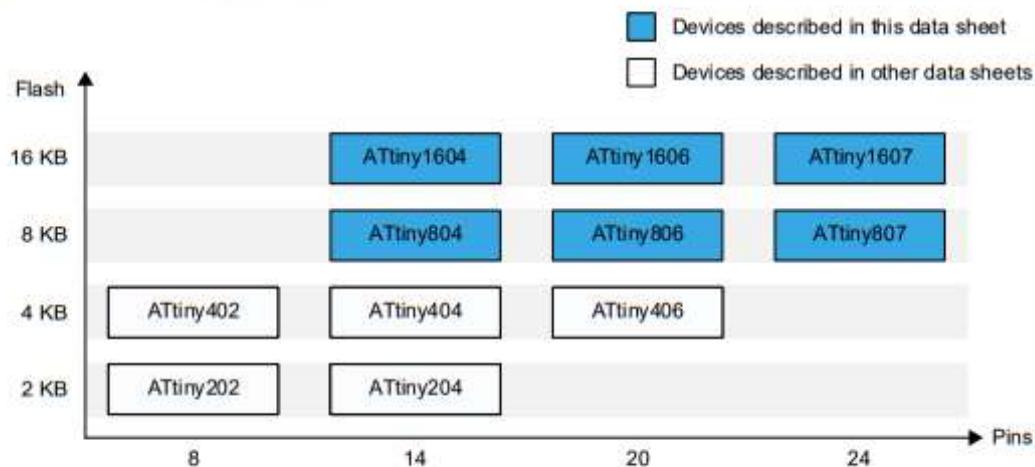
ATtiny804/806/807/1604/1606/1607 tinyAVR® 0-series Overview

tinyAVR® 0-series Overview

The following figure shows the tinyAVR 0-series devices, laying out pin count variants and memory sizes:

- Vertical migration upwards is possible without code modification, as these devices are pin-compatible and provide the same or more features. Downward migration may require code modification due to fewer available instances of some peripherals.
- Horizontal migration to the left reduces the pin count and, therefore, the available features

Figure 2-1. tinyAVR® 0-series Overview



Devices with different Flash memory sizes typically also have different SRAM and EEPROM.

Configuration Summary

Peripheral Summary

Table 2-1. Peripheral Summary

	ATTiny804	ATTiny806	ATTiny807	ATTiny1604	ATTiny1606	ATTiny1607
Pins	14	20	24	14	20	24
SRAM	512B	512B	512B	1024B	1024B	1024B
Flash	8 KB	8 KB	8 KB	16 KB	16 KB	16 KB
EEPROM	128B	128B	128B	256B	256B	256B
Max. frequency (MHz)	20	20	20	20	20	20
16-bit Timer/Counter type A (TCA)	1	1	1	1	1	1
16-bit Timer/Counter type B (TCB)	1	1	1	1	1	1
12-bit Timer/Counter type D (TCD)	No	No	No	No	No	No
Real-Time Counter (RTC)	1	1	1	1	1	1

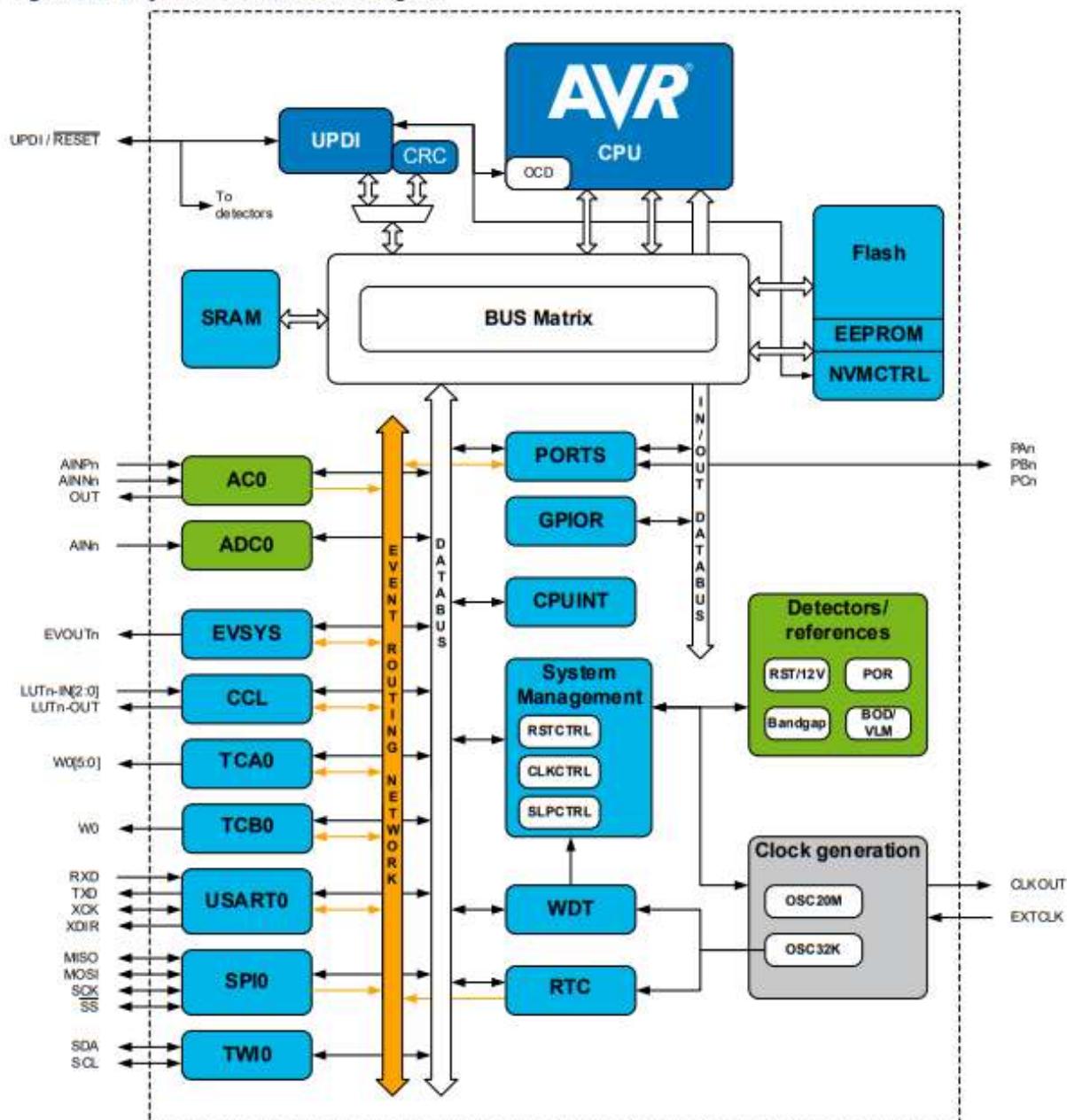


ATtiny804/806/807/1604/1606/1607

Block Diagram

Block Diagram

Figure 3-1. tinyAVR® 0-series Block Diagram

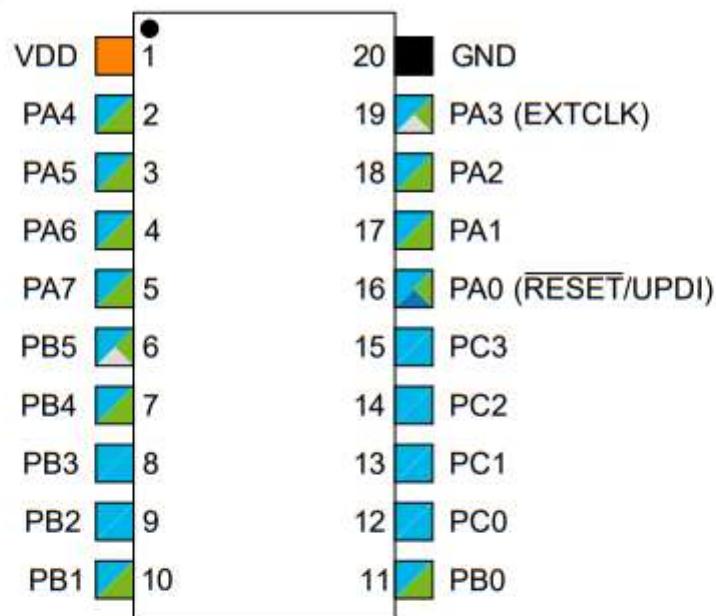


Note: The block diagram represents the largest device of the tinyAVR® 0-series, both in terms of pin count and Flash size. See sections [2.1 Configuration Summary](#) and [5. I/O Multiplexing and Considerations](#) for an overview of the features of the specific devices in this data sheet.



ATtiny804/806/807/1604/1606/1607 Pinout

20-Pin SOIC



Power

Power Supply

Ground

Pin on VDD Power Domain

Functionality

Programming/Debug

Clock/Crystal

Digital Function Only

Analog Function



20.4.3. SPI-Datasheets

TB3215 Overview

Overview

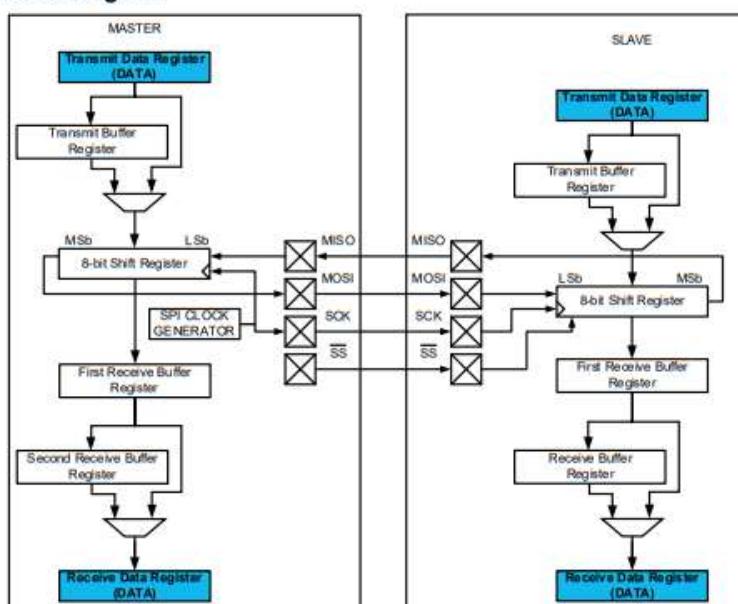
The SPI bus is a synchronous serial communication interface based on four types of logic signals:

- SCK: Serial Clock (output from master)
- MOSI: Master Output Slave Input (data output from master)
- MISO: Master Input Slave Output (data output from slave)
- SS: Slave Select (active-low, output from master)

This peripheral is used for short distance and high-speed communication, primarily in embedded systems. The SPI devices communicate in Full-Duplex mode, using a channel for transmitting and one for receiving data. The SPI is based on a master-slave architecture with a single master at a time and one or more slaves. The master device is the only one that can generate a clock, thus it is the initiator of the data exchange. The SPI master device uses the same SCK, MOSI and MISO channels for all the slaves, but usually individual lines of SS for each of the slaves. The master device selects the desired slave by pulling the SS signal low.

The data to be sent will be stored in either a data register or, if a transmission is already in progress and the Buffer mode was activated, in a buffer register. The data are sent out serially on the MOSI channel, using a shift register, and every bit is being synchronized using the SPI clock generator. While every bit is shifted out, new data are received on the MISO channel from the slave and are shifted in a receiver buffer and further in the receive DATA register. If the receiver is busy, meaning there are already data in the receive DATA register and the Buffer mode was activated, the data will be temporarily stored in a second receiver buffer. The Buffer mode is activated by setting high the BUFEN bit of the CTRLB register.

Figure 2-1. SPI Block Diagram



The SPI module has five registers. One register is used for data transfer and storage, two registers are used for Interrupt flags, and the other two registers (CTRLA and CTRLB) are for initializations. All the configurations required to make the peripheral work correctly are reduced to changing some bits in the CTRLA register, while the CTRLB register is focused on different modes of operation that are optional.



Sending Data as a Master SPI Device

The master is the device that decides when to trigger communication and which slave is the recipient of the message. SPI master devices are generally used in high-speed communication and the focus is to exchange data with other devices acting as slaves (e.g. sensors, memories or other MCUs).

This use case follows the steps:

- Configure the location of the SPI pins
- Initialize the peripheral
- Configure the direction of the pins
- Control slave devices
- Send data as a master device

How to Configure the Location of the SPI Pins

The way to configure the location of the pins is independent of the application purpose and the SPI mode. Each microcontroller has its own default physical pin position for peripherals. These locations can be found on PORTMUX peripheral chapter from the family data sheet of the megaAVR 0-series. For ATmega4809, the SPI pins are located on PA[7:4] and can be changed on PC[3:0] or PE[3:0] using the TWISPIROUTEA register of the PORTMUX module.

Figure 3-1. TWISPIROUTEA Register

Bit	7	6	5	4	3	2	1	0
			TWI0[1:0]				SPI0[1:0]	
Access		R/W	R/W			R/W	R/W	

The order of the pins is the following: MOSI, MISO, SCK, SS; MOSI representing the lowest pin number from the group. This is how a user can change the location of the SPI pins for option 1 with port C:

Value	Name	Description
0x0	DEFAULT	SPI on PA[7:4]
0x1	ALT1	SPI on PC[3:0]
0x2	ALT2	SPI on PE[3:0]
0x3	NONE	Not connected to any pins

This translates into the following code:

```
PORTRMUX.TWISPIROUTEA |= PORTRMUX_SPI00_bm;
```

Or option 2 with port E:



TB3215

Sending Data as a Master SPI Device

Value	Name	Description
0x0	DEFAULT	SPI on PA[7:4]
0x1	ALT1	SPI on PC[3:0]
0x2	ALT2	SPI on PE[3:0]
0x3	NONE	Not connected to any pins

This translates into the following code:

```
PORTMUX.TWISPIROUTEA |= PORTMUX_SPI01_bm;
```

How to Initialize the Peripheral

The clock frequency is derived from the main clock of the microcontroller and is reduced using a prescaler or divider circuit present in the SPI hardware. By default, the source of the main clock is a 20 MHz internal oscillator, which is divided by a prescaler whose default value is 6. Thus, resulting in a main clock frequency of approximately 3.33 MHz. More information about the clock can be found in Clock Controller chapter of the family data sheet of the megaAVR 0-series.

The clock frequency of the SPI can also be increased using the Double Clock mode, which works only in Master mode. The Data Order bit represents the endianness (Most Significant bit or Least Significant bit) of the data, the order in which the bits are transmitted on the channel (starting with the last or the first bit from a register). All the configurations are related to CTRLA register.

Figure 3-2. CTRLA Register

Bit	7	6	5	4	3	2	1	0
Access		DORD	MASTER	CLK2X		PRESC[1:0]		ENABLE
Reset	0	0	0		0	0	0	0

Next is an example on how to configure a master SPI device with the default main clock source and with the default pin location presented in the previous topic. A 416 kHz frequency will result by configuring the device in Double-Speed mode and with a 16 times divider. The data will be shifted out starting with the Most Significant bit (MSb):

Value	Name	Description
0x0	DIV4	CLK_PER/4
0x1	DIV16	CLK_PER/16
0x2	DIV64	CLK_PER/64
0x3	DIV128	CLK_PER/128

This translates into the following code:

```
SPI0.CTRLA = SPI_CLK2X_bm  
| SPI_DORD_bm  
| SPI_ENABLE_bm  
| SPI_MASTER_bm  
| SPI_PRESC_DIV16_gc;
```



TB3215

Sending Data as a Master SPI Device

How to Configure the Direction of the Pins

Since the master devices control and initiate transmissions, the MOSI, SCK and \overline{SS} pins must be configured as output, while the MISO channel will keep its default direction as input. The default values, directions and configurations explained above are still applicable here. The following example is based on the default position of the SPI pins:

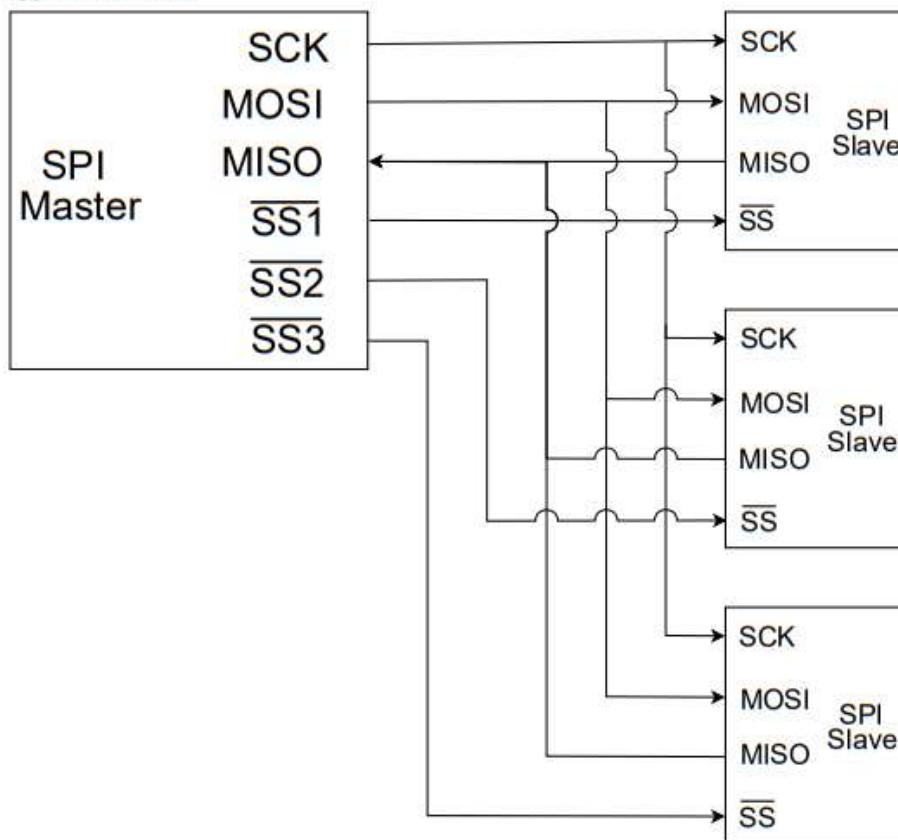
```
PORTE.DIR |= PIN4_bm; // MOSI channel
PORTE.DIR &= ~PIN5_bm; // MISO channel
PORTE.DIR |= PIN6_bm; // SCK channel
PORTE.DIR |= PIN7_bm; // SS channel
```

An SPI master device can control more than one slave, thus requiring more \overline{SS} pins. The additional \overline{SS} channels can be configured just like the one in the example above. The user must choose an unused pin and configure its direction as output.

How to Control Slave Devices

A master will control a slave by pulling low the \overline{SS} pin. If the slave has set the direction of the MISO pin to output, when the \overline{SS} pin is low, the SPI driver of the slave will take control of the MISO pin, shifting data out from its transmit DATA register. All slave devices can receive a message, but only those with \overline{SS} pin pulled low can send data back. Though, it is not recommended to enable more than one slave in a typical connection (like the one below) because all of them will try to respond to the message and there is only one MISO channel, thus the transmission will result in a write collision. The user can check the appearance of collisions by reading the value of WRCOL bit in INTFLAGS register.

Figure 3-3. Typical SPI Bus





TB3215

Sending Data as a Master SPI Device

How to Send Data as a Master Device

All the settings configured before are considered in the following example and the polling method is used for flag checking. Before sending data, the user must pull low an SS signal to let the slave device know it is the recipient of the message.

```
PORTA.OUT &= ~PIN7_bm;
```

Once the user writes new data into the DATA register the hardware starts a new transfer, generating the clock on the line and shifting out the bits.

Figure 3-4. DATA Register

Bit	7	6	5	4	3	2	1	0
DATA[7:0]								
Access	R/W							
Reset	0	0	0	0	0	0	0	0

```
SPI0.DATA = data;
```

When the hardware finishes shifting all the bits, it activates a receive Interrupt flag, which can be found in the INTFLAGS register.

Figure 3-5. INTFLAGS Register

Bit	7	6	5	4	3	2	1	0
	IF	WRCOL						
Access	R/W	R/W						

The user must check the state of the flag, before writing new data in the register, by either activating the interrupts or by constantly reading the value of the flag (method called polling), else a write collision interrupt will occur.

```
while (!(SPI0.INTFLAGS & SPI_IF_bm))
{
    ;
}
```

The user can pull the SS channel high if there is nothing left to transmit.

```
PORTA.OUT |= PIN7_bm;
```

Full Code Example



View Code Example on GitHub
Click to browse repository



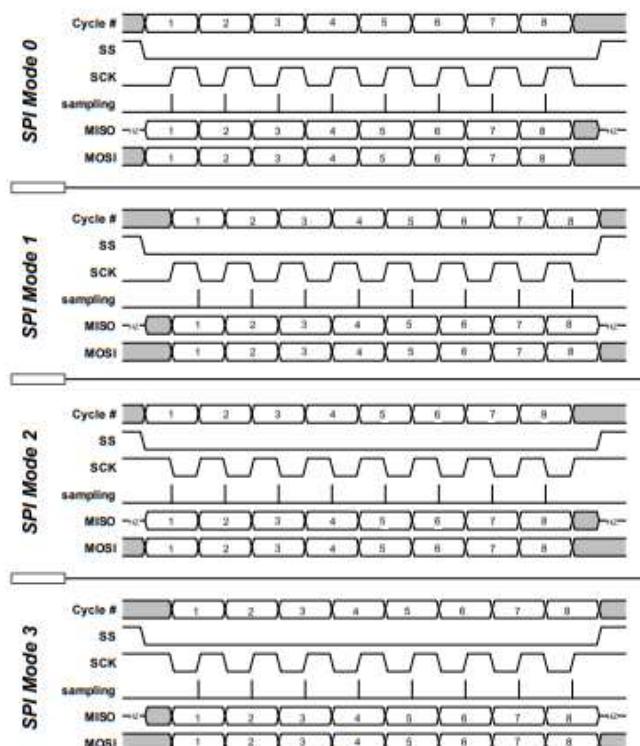
Tip: The full code example is also available in the [Appendix](#) section.



Changing Data Transfer Type

It represents the way in which data is transmitted with respect to the clock generation. The clock polarity and the clock phase are the ones important for data modes. By clock polarity, one can understand the level of the signal which can be low while in Idle state and will start with a rising edge when transmitting data, or it can be high while in Idle state and will start with a falling when exchanging data. Depending on the phase, the data are generated or sampled with respect to the clock on the channel: on a rising or a falling edge. See figure:

Figure 5-1. SPI Data Transfer Modes



Both the master and the slave devices must be configured in the same way, so one can decode correctly what the other encoded. Data modes can be selected by changing the value of MODE[1:0] bit field from CTRLB register.

Figure 5-2. CTRLB Register

Bit	7	6	5	4	3	2	1	0
Access	BUFEN	BUFWR				SSD	MODE[1:0]	
Reset	0	0				0	0	0

Until now, the examples were based on SPI Mode 0 because there was no change made to these bits and that is the default value of the bits.



TB3215
Changing Data Transfer Type

Here is an example of how to configure the SPI in Data Mode 3, and is based on the normal/basic master SPI Initialization mode presented in the [Sending Data as a Master SPI Device](#) section, the only difference being the change of the data transmission type:

```
SPI0.CTRLB |= SPI_MODE_3_gc;
```

Value	Name	Description
0x0	0	Leading edge: Rising, sample Trailing edge: Falling, setup
0x1	1	Leading edge: Rising, setup Trailing edge: Falling, sample
0x2	2	Leading edge: Falling, sample Trailing edge: Rising, setup
0x3	3	Leading edge: Falling, setup Trailing edge: Rising, sample

Full Code Example



[View Code Example on GitHub](#)
Click to browse repository



Tip: The full code example is also available in the [Appendix](#) section.



Appendix

Example 7-1. Sending Data as a Master SPI Device Full Code Example

```
#include <avr/io.h>

void SPI0_init(void);
void slaveSelect(void);
void slaveDeselect(void);
uint8_t SPI0_exchangeData(uint8_t data);

void SPI0_init(void)
{
    PORTA.DIR |= PIN4_bm; /* Set MOSI pin direction to output */
    PORTA.DIR &= ~PIN5_bm; /* Set MISO pin direction to input */
    PORTA.DIR |= PIN6_bm; /* Set SCK pin direction to output */
    PORTA.DIR |= PIN7_bm; /* Set SS pin direction to output */

    SPI0.CTRLA = SPI_CLK2X_bm           /* Enable double-speed */
                | SPI_DORD_bm          /* LSB is transmitted first */
                | SPI_ENABLE_bm        /* Enable module */
                | SPI_MASTER_bm        /* SPI module in Master mode */
                | SPI_PRESC_DIV16_gc;   /* System Clock divided by 16 */
}

uint8_t SPI0_exchangeData(uint8_t data)
{
    SPI0.DATA = data;

    while (!(SPI0.INTFLAGS & SPI_IF_bm)) /* waits until data is exchanged*/
    {
        ;
    }

    return SPI0.DATA;
}

void slaveSelect(void)
{
    PORTA.OUT &= ~PIN7_bm; // Set SS pin value to LOW
}

void slaveDeselect(void)
{
    PORTA.OUT |= PIN7_bm; // Set SS pin value to HIGH
}

int main(void)
{
    uint8_t data = 0;

    SPI0_init();

    while (1)
    {
        slaveSelect();
        SPI0_exchangeData(data);
        slaveDeselect();
    }
}
```

Example 7-2. Receiving Data as a Slave SPI Device Full Code Example

```
#include <avr/io.h>
#include <avr/interrupt.h>

void SPI0_init(void);
volatile uint8_t receiveData = 0;
```



```
volatile uint8_t writeData = 0;

void SPI0_init(void)
{
    PORTA.DIR &= ~PIN4_bm; /* Set MOSI pin direction to input */
    PORTA.DIR |= PIN5_bm; /* Set MISO pin direction to output */
    PORTA.DIR &= ~PIN6_bm; /* Set SCK pin direction to input */
    PORTA.DIR &= ~PIN7_bm; /* Set SS pin direction to input */

    SPI0.CTRLA = SPI_DORD_bm           /* LSB is transmitted first */
                | SPI_ENABLE_bm        /* Enable module */
                & (~SPI_MASTER_bm); /* SPI module in Slave mode */

    SPI0.INTCTRL = SPI_IE_bm;          /* SPI Interrupt enable */
}

ISR(SPI0_INT_vect)
{
    receiveData = SPI0.DATA;
    SPI0.DATA = writeData;

    SPI0.INTFLAGS = SPI_IF_bm; /* Clear the Interrupt flag by writing 1 */
}

int main(void)
{
    SPI0_init();

    sei(); /* Enable Global Interrupts */

    while (1)
    {
        ;
    }
}
```



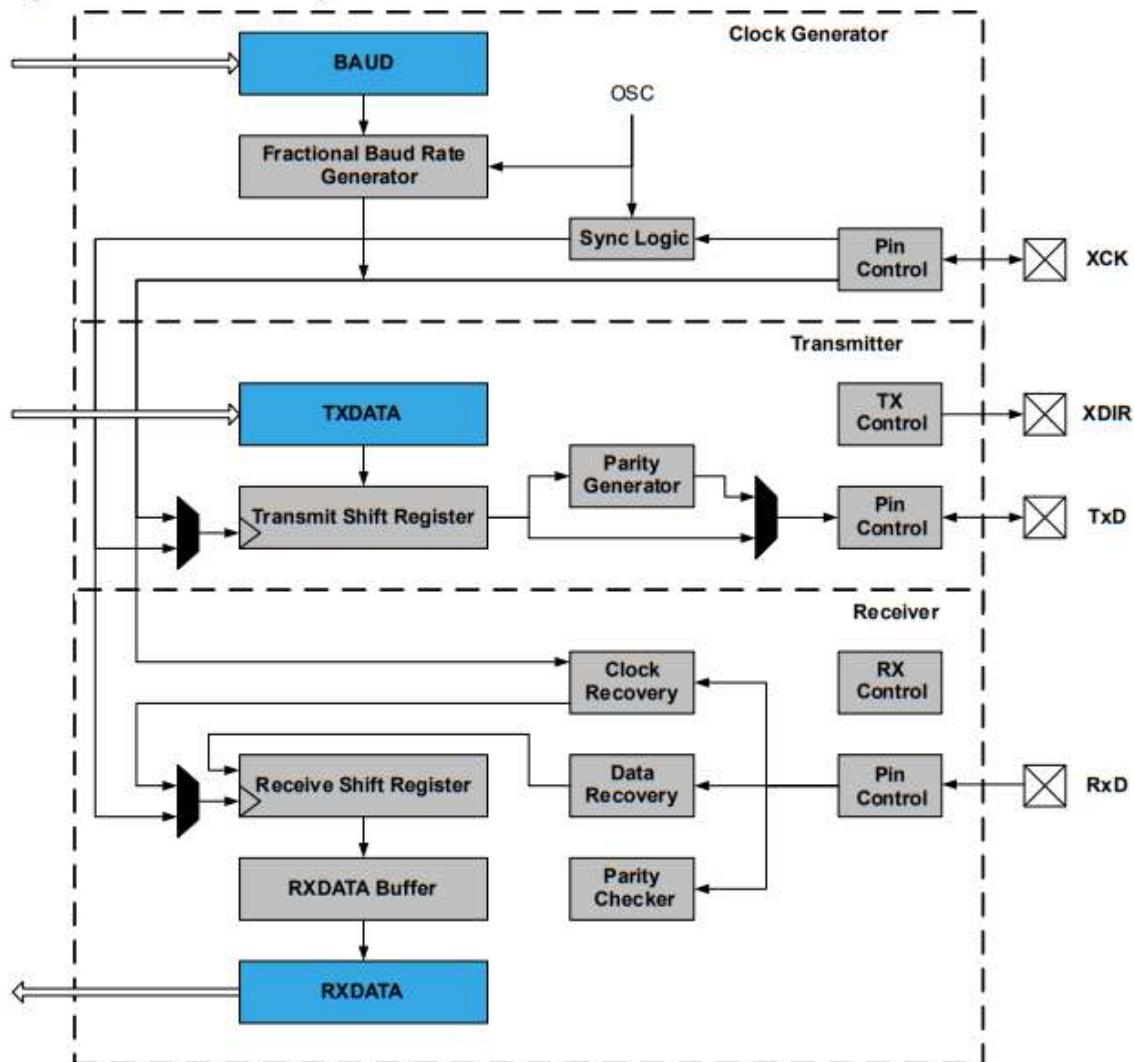
20.4.4. USART-Datasheets

TB3216 Overview

Overview

The USART module has four pins, named RX (receive), TX (transmit), XCK (clock) and XDIR (direction). In One-Wire mode only, the TX pin is used for both transmitting and receiving. The downside of this mode is that it only provides half-duplex communication. In Asynchronous mode, both RX and TX pins are used, thus achieving full-duplex communication. The XCK pin is used for clock signal in Synchronous mode, and the XDIR pin is used for RS485 mode.

Figure 2-1. USART Block Diagram



The most common USART configuration is referred to as "9600 8N1", meaning 9600 baud rate, eight data bits, no parity, and one Stop bit. Therefore, a typical USART frame will have 10 bits (one Start bit, eight data bits, and one Stop bit) and will be able to represent one ASCII character, which means an "8N1" configuration will transmit $\text{BAUD_RATE}/10$ ASCII characters per second.

Note: All examples described in this document will use a 9600 baud rate and "8N1" frame format. The serial terminal must be set for this configuration.

Moreover, the USART is a complex peripheral and can be used to achieve a handful of other protocols such as:



How to Enable the Transmitter and Send Data

Depending on the application needs, the user may choose to enable only the receiver or the USART module transmitter. Since in this use case only the microcontroller sends messages, only the transmitter needs to be enabled.

```
USART1.CTRLB |= USART_TXEN_bm;
```

Before sending data, the user needs to check if the previous transmission is completed by checking the USARTn.STATUS register. The following code example waits until the transmit DATA register is empty and then writes a character to the USARTn.TXDATA register:

```
void USART1_sendChar(char c)
{
    while (!(USART1.STATUS & USART_DREIF_bm))
    {
        ;
    }
    USART1.TXDATAL = c;
}
```

The Send register is nine bits long. Therefore, it was split into two parts: The lower part that holds the first eight bits, called TXDATAL, and the higher part that holds the remaining one bit, called TXDATAH. TXDATAH is used only when the USART is configured to use nine data bits. When used, this ninth bit must be written before writing to USARTn.TXDATAL, except if CHSIZE in USARTn.CTRLC is set to '9-bit - Low byte first', where USARTn.TXDATAL should be written first.

How to Configure Pins

The TX pin must be configured as an output. By default, each peripheral has some associated pin positions. The pins are described in the *Multiplexed Signals* section in the device-specific data sheet. Each USART has two sets of pin positions. The default and alternate pin positions for USART1 are shown below.

Table 3-1. Multiplexed Signals

Pin name ^(1,2)	USARTn
PC0	1,TxD
PC1	1,RxD
PC2	1,XCK
PC3	1,XDIR
VDD	
GND	
PC4	1,TxD ⁽³⁾
PC5	1,RxD ⁽³⁾
PC6	1,XCK ⁽³⁾
PC7	1,XDIR ⁽³⁾

Notes:

1. Pin names are of type Px_n, with x being the PORT instance (A,B,C,...) and n the pin number. Notation for signals is PORTx_PINn. All pins can be used as event input.
2. All pins can be used for external interrupt, where pins Px2 and Px6 of each port have full asynchronous detection.
3. Alternate pin positions. For selecting the alternate positions, refer to the PORTMUX documentation.



Send Formatted Strings/Send String Templates Using `printf`

It is a common use case for an application to send a string with variable fields, for example, when the application reports its state or a counter value. Using formatted strings is a very flexible approach and reduces the number of code lines. This can be accomplished by changing the output stream of the '`printf`' function.

This use case follows these steps:

- Configure the USART peripheral the same as for the first use case
- Create a user defined stream
- Replace the standard output stream with the user-defined stream

Usually, when using '`printf`', the characters are sent to a stream of data, called standard output stream. On a PC, the standard output stream is handled by the function to display characters on the screen. But streams can be created so that another function handles their data.

The following code creates a user-defined stream that will be handled by the `USART1_printChar` function. This function is a wrapper of the `USART1_sendChar` function but has a slightly different signature to match what `FDEV_SETUP_STREAM` expects as a parameter.

```
static void USART1_sendChar(char c)
{
    while (!(USART1.STATUS & USART_DREIF_bm))
    {
        ;
    }
    USART1.TXDATAL = c;
}

static int USART1_printChar(char c, FILE *stream)
{
    USART1_sendChar(c);
    return 0;
}

static FILE USART_stream = FDEV_SETUP_STREAM(USART1_printChar, NULL, _FDEV_SETUP_WRITE);
```

Then replace the standard output stream with the user-defined stream, handled by the USART send function.

```
stdout = &USART_stream;
```

The application can now use '`printf`' instead of writing to USART registers directly.

```
uint8_t count = 0;
while (1)
{
    printf("Counter value is: %d\r\n", count++);
    delay_ms(500);
}
```

Note: The '`printf`' function uses specifiers to mark where to insert variables in the string template. Some of the available specifiers are in the table below:

Table 4-1. `printf` Specifiers

Specifier	Description
%d	Insert a signed integer
%s	Insert a sequence of characters
%c	Insert a character
%x	Insert integer unsigned in hex format



Receive Control Commands

One important usage of the USART represents the implementation of a command-line interface. This way, the microcontroller can receive control commands via USART. It is convenient to use the line terminator as a command delimiter, so, for this use case, the USART will read full lines.

This use case follows the steps:

- Configure the USART peripheral same as for the first use case
- Enable the receiver
- Read and store the incoming data until the end of line
- Check if the received data are a valid command; if so, execute it

How to Enable the Receiver and Receive Data

For USART1, the default pin position for RX is Port C pin 1 (PC1). The following line sets the PC1 direction to input.

```
PORTE.DIR |= -PIN1_bm;
```

Same as the transmitter, the receiver is enabled by writing to the USARTn.CTRLB register.

```
USART1.CTRLB |= USART_RXEN_bm;
```

Before reading the data, the user must wait for the data to be available by polling the Receive Complete Interrupt Flag, RXCIF.

```
uint8_t USART1_read()
{
    while (!(USART1.STATUS & USART_RXCIF_bm))
    {
        ;
    }
    return USART1.RXDATAL;
}
```

How to Read a Line

The following code snippet reads one line of data and stores it in an array. It assumes that a valid line is shorter than the array length.

The array index is reset to zero when reaching the array end to avoid a buffer overflow error in case of longer lines received. The characters '\n' (line feed) and '\r' (carriage return) are ignored because they are part of the line terminator. When '\n' is found, the string end (NULL) is added to the command, and the function 'executeCommand' will call a function based on the value of the command string.

```
char command[MAX_COMMAND_LEN];
uint8_t index = 0;
char c;

/* This delay invalidates the initial noise on the TX line, after device reset. */
_delay_ms(10);

while (1)
{
    c = USART1.readChar();
    if(c != '\n' && c != '\r')
    {
        command[index++] = c;
        if(index > MAX_COMMAND_LEN)
        {
            index = 0;
        }
    }
    if(c == '\n')
    {
```



Other Implementation Modes

The applications described above demonstrate the basic USART functionalities. This section describes the USART configured in Synchronous mode and One-Wire mode.

Synchronous Mode

Figure 6-1. USART Communication Mode (CMODE) Bit Field in Control C Register

Bit	7	6	5	4	3	2	1	0
	CMODE[1:0]		PMODE[1:0]		SBMODE	CHSIZE[2:0]		
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The CMODE bit field in the CTRLC register controls the communication modes.

The disadvantage of the Asynchronous mode is that the receiver chip and the transmitter chip need to use the same baud rate, and exact timing is required. The asynchronous protocols use a separate line for the clock signal, so the chip that generates the clock dictates the communication speed, which is much more flexible in terms of exact timings and creates two roles in the communication: The server that generates the clock and the client that receives the clock.

In the Synchronous USART mode, an additional clock pin, XCK, is used. Same as the RX and TX pins, XCK has a default pin, and changing the PORTMUX register will also change XCK. Configuring the XCK direction decides if the device is a server (generates clock) or a client (receives clock).

To activate the Synchronous mode:

- Configure the XCK pin (PC2) direction as output;
`PORTC.DIR |= ~PIN2_bm;`
- Write 0x01 to the CMODE bit field in the USARTn.CTRLC register.

Figure 6-2. USART Communication Mode

Value	Name	Description
0x0	ASYNCHRONOUS	Asynchronous USART
0x1	SYNCHRONOUS	Synchronous USART
0x2	IRCOM	Infrared Communication
0x3	MSPI	Host SPI

`USART1.CTRLC = USART_CMODE_SYNCHRONOUS_gc;`



View the ATmega4809 Code Example on GitHub
Click to browse repository

An MPLAB MCC generated code example for AVR128DA48 with the same functionality as the one described in this section can be found here:



View the AVR128DA48 Code Example on GitHub
Click to browse repository



One-Wire Mode

Using only one wire effectively reduces the number of pins used for USART communication to one. RX and TX are internally connected, and only TX is used, which means that both incoming and outgoing data will share the same wire, so transmission and reception cannot happen simultaneously. This is called half-duplex communication.

Figure 6-3. Loop-back Mode Enable (LBME) Bit in Control A Register

Bit	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	DREIE	RXSIE	LBME	ABEIE		RS485[1:0]
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Use the LBME bit in the CTRLA register to enable an internal loopback connection between RX and TX. An internal connection between RX and TX can be created by writing to USARTn.CTRLA.

```
USART1.CTRLA |= USART_LBME_bm;
```

This will internally connect the RX and TX pins, but only the TX pin is used. As the TX pin is used for both transmit and receive, the pin direction needs to be configured as an output before each transmission and switched back to input when the transmission ends.

Since RX is connected internally to TX during transmission, it will receive the data sent, which can be used as a collision detection mechanism. If there is another transmission occurring, the received data will not match the transmitted data. An advanced one-wire driver could take advantage of this strategy.



[View the ATmega4809 Code Example on GitHub](#)

Click to browse repository



[View the AVR128DA48 Code Example on GitHub](#)

Click to browse repository

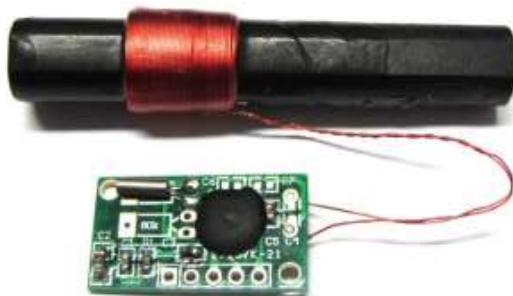


20.5. DCF-77-Empfangsmodul:



DAEV6180B1COB.005
3 April, 2012

TIME SIGNAL RECEIVER MODULE



- Tuned ferrite antenna
- AM receiver IC board
- Reception of:
 - German DCF77
 - US WWVB
 - British MSF
 - Japanese JJY60

INTRODUCTION

The time signal receiver module comprises of a ferrite antenna and an AM receiver IC printed circuit board. The board includes a MAS6180B1 AM receiver IC accompanied with necessary filter crystal and capacitor components. The circuitry includes also an RC-filter for the supply voltage. The EB6180B1COB77K5A1 module is tuned for 77.5 kHz and suitable for receiving German DCF77 time signal transmission whereas the EB6180B1COB60K0A1 and A2 modules are tuned for 60 kHz suitable for receiving US WWVB, British

MSF and Japanese JJY60 time signal transmissions. The A2 module with 100mm antenna bar is especially suited for WWVB to cover the weak signal areas.

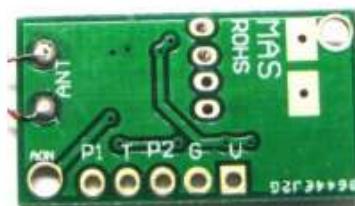
The MAS6180B1 AM receiver IC includes amplifier, demodulator and comparator blocks that transforms the received AM transmission into series of pulse width coded digital pulses which can be directly processed by an appropriate digital circuitry such as a micro controller unit (MCU).

PIN DESCRIPTION

Pin ID	Type	Function	Note
P1	DI	PDN (power down) control pin	HIGH = receiver off LOW = receiver on Do not leave this pin floating
T	DO	Time pulse output	
P2	NC	-	Leave unconnected
G	G	Supply ground	
V	P	Supply voltage	
AON	DI	AGC on/off control (optional)	Leave unconnected when not used

D = Digital, P = Power, G = Ground, I = Input, O = Output, NC = Not Connected

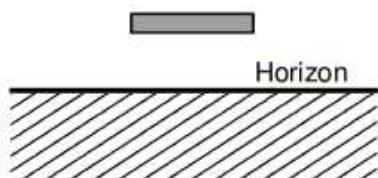
PCB backside
pin marking





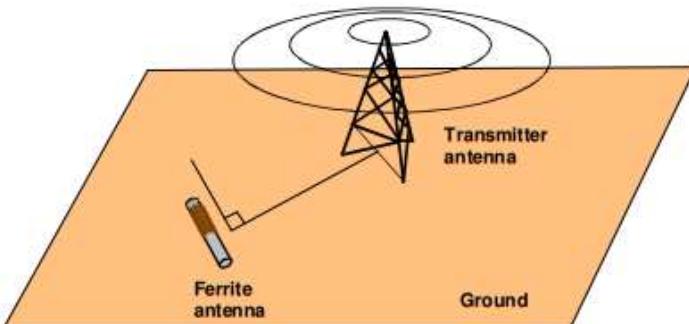
APPLICATION INFORMATION

Antenna orientation



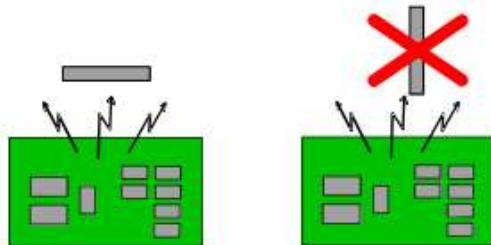
The magnetic field component of the propagating long wave time signal transmission has a horizontal polarization thus the ferrite antenna should be oriented horizontally to maximize the signal (see figure 1).

Figure 1. Antenna orientation relative to ground



The ferrite antenna should also be pointing orthogonally relative to the transmitter (see figure 2).

Figure 2. Antenna orientation relative to transmitter station



The ferrite bar antenna should be located as far as possible from conductive metal walls, PCB ground plane or ferromagnetic objects (speakers). All those objects affect the antenna tuning and can attenuate the received signal. To avoid noise coupling the ferrite antenna should also not be pointing towards noisy electronic circuits (figure 3). It is a good practice to turn off all unnecessary electronic circuits when receiving the weak radio transmission.

Figure 3. Antenna orientation relative to noisy electric circuits

Getting a signal

The antenna is sensitive for magnetic and electric disturbances. As an example, in digital radio controlled clocks it is known that LCD displays, refreshed using a 32Hz signal, has a 1875th odd harmonic hitting exactly at 60kHz and its amplitude can be strong enough (μV_{rms} level) to reduce the sensitivity. The antenna and module placement is critical and one should maximize distance to other disturbing electronics and metal/ferrous parts which might affect the antenna and the reception.

A good place to start is to put the module close to a window and turn the antenna to an optimal position relative the transmitter (see the figure 2 above). As the second step trigger the fast startup by moving PDN control from power down (PDN=VDD) to power up (PDN=VSS) which will make the AGC find its level within a few seconds if the receiving conditions are sufficient. Initially the OUT signal should be high but soon after finding a signal (or disturbance in case of poor SNR) the output goes low and after a few seconds it should start receiving pulses. If the output stays low all the time there is probably some disturbance stronger than the signal. If the signal is bad, change location and repeat the fast startup by setting PDN=VDD (power down) to PDN=VSS (power up).

Please note that if PDN control is not used but the P1 pin (PDN) is permanently tied to GND (receiver on), the start-up time before the receiver finds the signal can take a few minutes.



ELECTRICAL CHARACTERISTICS

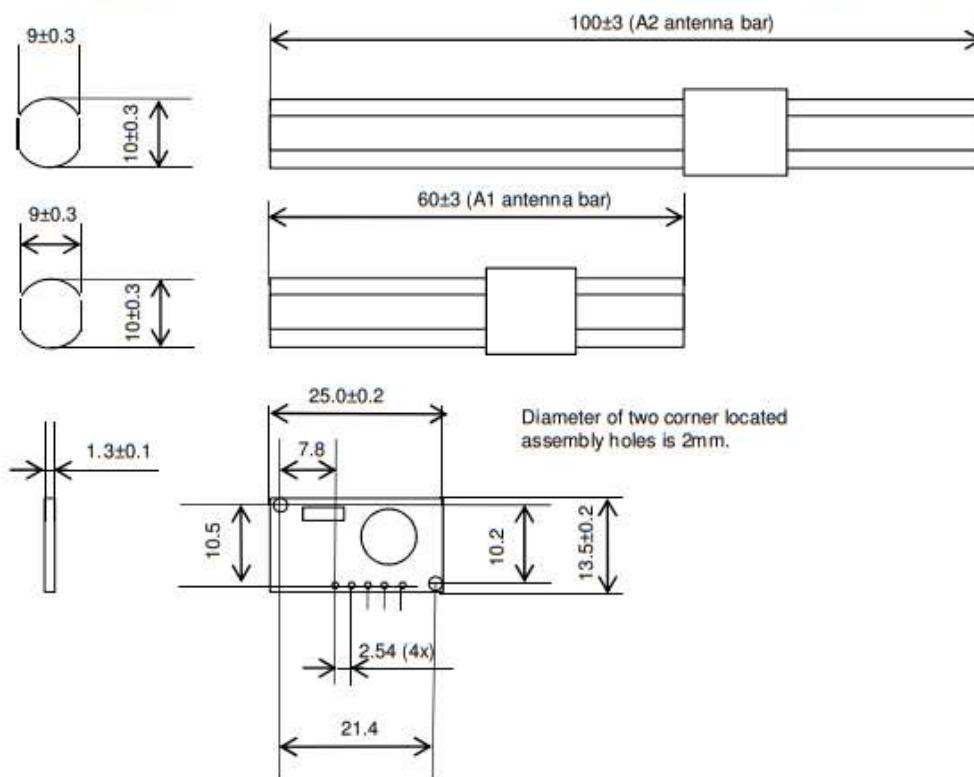
Operating Conditions: VDD = 1.5V, Temperature = 27°C, unless otherwise specified.

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Operating Voltage	V _{DD}		1.1	1.5	3.6	V
Current Consumption	I _{DD}	VDD=1.5 V, weak signal VDD=1.5 V, strong signal VDD=3.6 V, weak signal VDD=3.6 V, strong signal		66 40 68 42	85 65	µA
Stand-By Current	I _{DDoff}				0.1	µA
Receiving Frequency	f _{IN}	module EB6180B1COB77K5A1 module EB6180B1COB60K0A1 module EB6180B1COB60K0A2 See ordering information below.		77.5 60 60		kHz
Sensitivity	E _{MIN}			25		µV/m

Note: For more detailed electrical characteristics see MAS6180B1 AM receiver IC datasheet

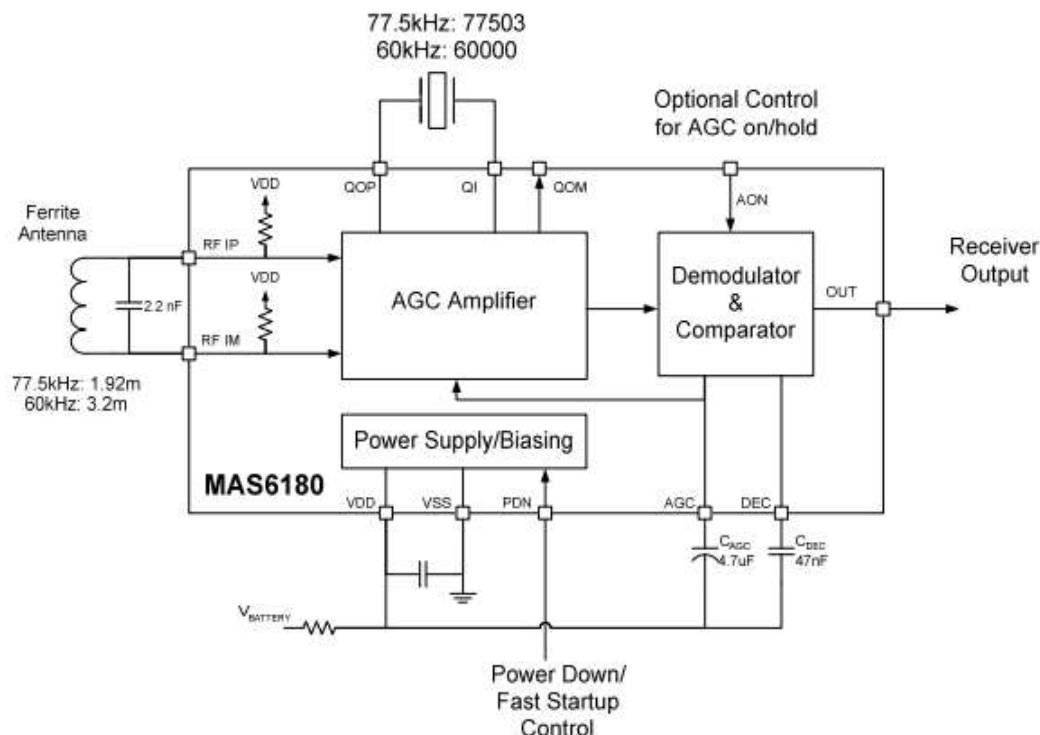
MECHANICAL DIMENSIONS

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Antenna						
length	L _A	A1 antenna bar	-3	60	+3	mm
width	W _A	A2 antenna bar	-3	100	+3	
height	H _A		-0.3	10	+0.3	
			-0.3	9	+0.3	
PCB						
length	L _{PCB}		-0.2	25.0	+0.2	mm
width	W _{PCB}		-0.2	13.5	+0.2	
thickness	T _{PCB}		-0.1	1.3	+0.1	





CIRCUIT SCHEMATIC



Note: The two attachment holes on the PCB corners have electrical connection to AON and GND. Ensure proper isolation when attaching to conductive enclosure.

ORDERING INFORMATION

Product Code	Product	Antenna
EB6180B1COB77K5A1	77.5kHz DCF77 receiver module	A1: 60x10x9 mm
EB6180B1COB60K0A1	60kHz WWVB/MSF/JJY60 receiver module	A1: 60x10x9 mm
EB6180B1COB60K0A2	60kHz WWVB/MSF/JJY60 receiver module	A2: 100x10x9 mm

Note: Modules are RoHS compliant.

MICRO ANALOG SYSTEMS OY CONTACTS

Micro Analog Systems Oy
Kutomotie 16
FI-00380 Helsinki, FINLAND

Tel. +358 10 835 1100
Fax +358 10 835 1119
<http://www.mas-oy.com>

NOTICE

Micro Analog Systems Oy reserves the right to make changes to the products contained in this data sheet in order to improve the design or performance and to supply the best possible products. Micro Analog Systems Oy assumes no responsibility for the use of any circuits shown in this data sheet, conveys no license under any patent or other rights unless otherwise specified in this data sheet, and makes no claim that the circuits are free from patent infringement. Applications for any devices shown in this data sheet are for illustration only and Micro Analog Systems Oy makes no claim or warranty that such applications will be suitable for the use specified without further testing or modification.



21. Abbildungsverzeichnis

Abbildung 1: Lixie-Uhr	1
Abbildung 2: Gruppenbild bei Blum	18
Abbildung 3: Fräsmaschine	19
Abbildung 4: Fräsmaschine	19
Abbildung 5: EAGLE Logo (EAGLE, 2023).....	20
Abbildung 6: KiCad Logo (KiCad, 2023).....	20
Abbildung 7: SOLIDWORKS Logo (SolidWorks, 2023)	20
Abbildung 8: Dremel Digilab 3D Slicer (Dremel, 2023).....	20
Abbildung 9: Fritzing Logo (Fritzing, 2023)	20
Abbildung 10: Dremel 3D-Drucker	36
Abbildung 11: Werkstücke	36
Abbildung 12: Dosiermaschine	47
Abbildung 13: Segmentplatine nach Dispensen	47
Abbildung 14: Master nach Dispensen	47
Abbildung 15: Handbestückungsmaschine.....	48
Abbildung 16: Dampfphasenlötanlage.....	48
Abbildung 17: DCF-77 Empfänger Modul	49
Abbildung 18: DCF77-Dekoder (DCF77-Dekoder, 2023)	49
Abbildung 19: DCF-77-Signal	53
Abbildung 20: DCF77-Signal	53
Abbildung 21: Fritzing APA102c	54
Abbildung 22: Funktionstest APA102c	56
Abbildung 23: Github Logo (Github, 2023)	57
Abbildung 24: OneDrive Logo (OneDrive, 2023)	57
Abbildung 25: Microchip Studio Logo (Studio, 2023)	57
Abbildung 26: Structorizer Logo (Structorizer, 2023)	58
Abbildung 27: TeraTerm Logo (TeraTerm, 2023)	58
Abbildung 28: FTI Chip Logo (Chip, 2023).....	58
Abbildung 29: Flussdiagram	59
Abbildung 30: Testaufbau 01.02.2023	76
Abbildung 31: Testaufbau 03.02.2023	76
Abbildung 32: Testaufbau 22.03.2023	77
Abbildung 33: Testaufbau 22.03.2023	77
Abbildung 34: Testaufbau 03.03.2023	78
Abbildung 35: Testaufbau: Lixie-Uhr	79



22. Quellenverzeichnis

Chip, F. (22. März 2023). *FTDI Chip*. Von

https://upload.wikimedia.org/wikipedia/en/c/c8/Logo_of_FTDI.png abgerufen

DCF77-Dekoder. (22. März 2023). *DCF77-Dekoder*. Von

https://www.hopf.com/about-dcf77_de.php abgerufen

Dremel. (22. März 2023). *Dremel*. Von <https://marketplace.createeducation.com/wp-content/uploads/2022/10/dremel-digilab-3d-slicer-169cc03bbf1342d9a79d92a7df568db4-1024x576.jpeg> abgerufen

EAGLE. (22. März 2023). *EAGLE*. Von <https://www.eurocircuits.de/wp-content/uploads/blog/Neues-groes-EAGLE-Software-Update/Eagle.png> abgerufen

Fritzing. (22. März 2023). *Fritzing*. Von

https://upload.wikimedia.org/wikipedia/commons/4/44/Fritzing_logo_%28new%29.png abgerufen

Github. (22. März 2023). *Github*. Von

https://github.githubusercontent.com/images/modules/logos_page/GitHub-Mark.png abgerufen

KiCad. (22. März 2023). *KiCad*. Von

<https://upload.wikimedia.org/wikipedia/commons/thumb/5/59/KiCad-Logo.svg/2560px-KiCad-Logo.svg.png> abgerufen

OneDrive. (22. März 2023). *OneDrive*. Von

https://upload.wikimedia.org/wikipedia/commons/thumb/3/3c/Microsoft_Office_OneDrive_%282019%E2%80%93present%29.svg/2560px-Microsoft_Office_OneDrive_%282019%E2%80%93present%29.svg.png abgerufen

SolidWorks. (22. März 2023). *SolidWorks*. Von <https://1000logos.net/wp-content/uploads/2020/08/SolidWorks-Logo.png> abgerufen

Structorizer. (22. März 2023). *Structorizer*. Von

<https://structorizer.fisch.lu/Pictures/structorizer.png> abgerufen

Studio, M. (22. März 2023). *Microchip Studio*. Von

https://www.it.unlv.edu/sites/default/files/styles/250_width/public/sites/default/files/assets/software/logos/atmel_studio.png?itok=bO_6oTM6 abgerufen

TeraTerm. (22. März 2023). *TeraTerm*. Von https://static-cdn.osdn.net/thumb/g/4/899/800x600_0.png abgerufen