DL7BJ Amateur Radio Station

BJ-Keyer
Dokumentation

ab Version 1.03 vom 10. September 2023

Tom, DL7BJ

Mail tom@dl7bj.de
Site https://isnix.de

Vorwort

Was ist ein elektronischer Morsezeichengeber? Das ist ein Gerät, welches wir Funkamateure besser unter dem Namen Morse-Keyer kennen. Kurzgesagt, ein Morse-Keyer erzeugt elektronisch Punkte, Striche und Pausen. Während dies mit der Handtaste zum Morsen manuell gemacht werden muss, wird ein Morse-Keyer in Verbindung mit Ein- oder Zweihebeltasten verwendet und erzeugt bei Betätigung die Punkte und Striche sowie die Pausen selbständig.

Ist das neu?

Nein, Morse-Keyer gibt es schon sehr lange. Als Fertiggeräte, als Bausätze und auch nur als Bauanleitungen in vielen verschiedenen Varianten. Etwas, das man quasi an jeder Straßenecke bekommt, in unterschiedlichen Preisklassen.

Warum noch ein Morse-Keyer?

Einige der erhältlichen Morse-Keyer sind in großen Gehäusen untergebracht, mit vielen Funktionen, Anschluß für eine PC Tastatur, dutzende Speicher und LC-Display und kosten viel Geld. Andere sind sehr günstig, haben aber nur einen Anschluß für eine Taste. Wer nicht gerade der Contester und DX-Jäger ist, gerne mal diverse Tasten an mehr als einem Transceiver verwendet und weder Steuerung über den PC noch Anschluß für Tastaturen benötigt, findet fast nichts am Markt.

Deswegen der BJ-Keyer, einfach, simpel, klein und trotzdem können mehrere Tasten und 2 Transceiver angeschlossen werden. Alles, was ich nicht benötige, habe ich auch weggelassen. Wer also auf der Suche nach einem Morse-Keyer mit ganz vielen Funktionen ist, dem empfehle ich, sich woanders umzuschauen.

Wer aber einen kleinen stationären Keyer mit wenigen aber praktischen Funktionen sucht, sollte hier weiterlesen.

Viel Spaß! Tom, DL7BJ

Inhalt

1 Hinweise zur Dokumentation

In dieser Dokumentation werden diverse gleichbleibende Darstellungsweisen verwendet. Dies erleichtert Dir das Verständnis der Bedeutung. Texte, die auf dem Display erscheinen, werden in der Bedienungsanleitung so `dargestellt`. Quellcode wird in einer farbigen Code-Darstellung eingebunden.

2 Funktionsübersicht

- Transceiver 1 oder 2 oder beide wählbar
- Iambic A, Iambic B und Ultimatic Mode
- Mithörton als Sinus, abschaltbar
- Anzeige der Geschwindigkeit in WpM oder BpM
- linkes/rechts Paddle vertauschen (Links- oder Rechtshänder)
- Punkt/Strich Verhältnis von 1:2 bis 1:4 einstellbar
- Punkt/Strich Gewichtung einstellbar
- Punkt/Strich Speicher abschaltbar (Iambic Modes)
- Frequenz des Mithörtons von 300-1000Hz einstellbar
- Anstieg (Attack) des Mithörtons einstellbar (kein Klicken im LS)
- Entprellzeit für Handtasten einstellbar
- Anschluß für 3 Handtasten und 3 Paddle
- Ausgänge für die Ansteuerung von 2 Transceivern
- optionales Tastenfeld für Speicher und Transceiver Umschaltung
- Speicherprogrammierung über USB
- Stromversorgung 7-15V
- Integrierter Lautsprecher für Mithörton

3 Grundlagen

3.1 Betriebsarten eines Morse-Keyers

Ein Morse-Keyer, die Tastelektronik, hat verschiedene Betriebsarten, die sich in der automatischen Erzeugung der Zeichen unterscheiden. Je nach Betriebsart gibt es einen Punkt- und Strichspeicher oder die direkte Erzeugung der Zeichen.

Bei einem zweiarmigen Paddle erzeugt ein Paddle (genauer die Tastelektronik) die Punkte und das andere Paddle die Striche. Diese Art von Paddle werden auch als Iambic oder Squeeze Paddle bezeichnet.

3.1.1 Iambic A

Im Iambic Mode A wird wird immer nur das Zeichen gegeben, dessen Paddle gerade betätigt wird. Wird das Paddle zu früh losgelassen, ergänzt die Tastelektronik das Zeichen auf die erforderliche Länge. Es wird kein weiteres Zeichen gesendet. Werden beide Paddle gleichzeitig gedrückt gehalten, ergibt sich eine Punkt-Strich-Folge. Wird der Kontakt wieder geöffnet, stoppt die Punkt-Strich-Folge. Es wird nur das Zeichen mit der erforderlichen Länge gesendet, welches beim Öffnen der Kontakte gerade gesendet wurde.

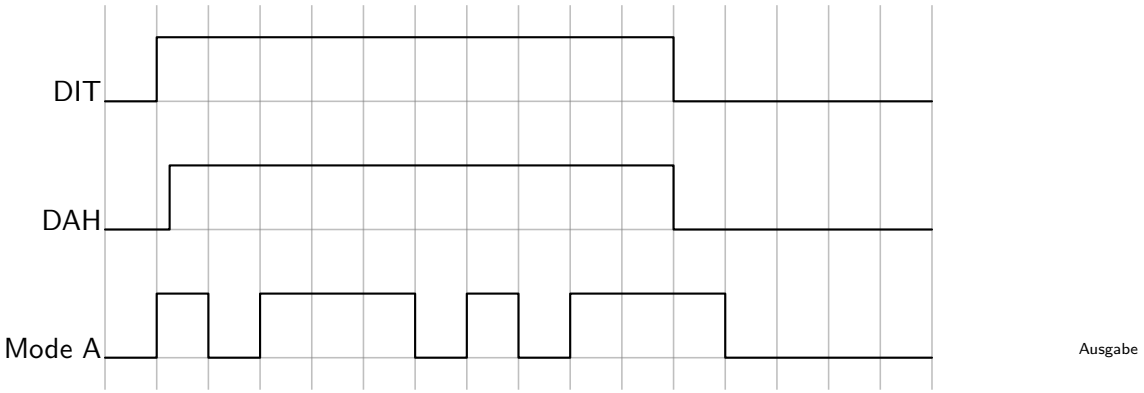

Abb. 3.1: Diagramm Iambic Mode A

3.1.2 Iambic B

Im Iambic Mode B verhält es sich im Prinzip wie im Iambic Mode A, nur dass beim gleichzeitigen Öffnen der Kontakte das entgegengesetzte Zeichen des zuletzt geöffneten Kontakts angefügt wird.

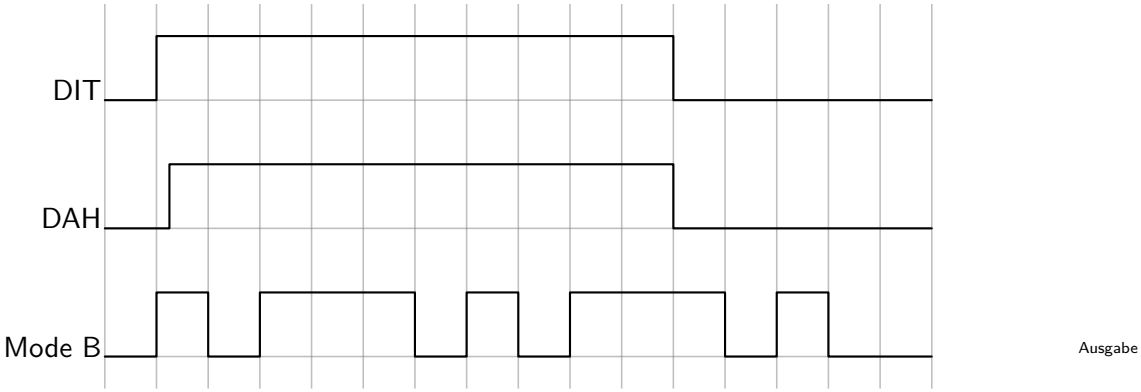

Abb. 3.2: Diagramm Iambic Mode B

Wie im Diagramm zu sehen ist, wird erst das DIT Paddle betätigt, dann das DAH Paddle, es sind also beide Paddle gleichzeitig betätigt. Es wird eine Folge von DIT und DAH, beginnend mit einem DIT gegeben, solange beide Paddle betätigt werden. Werden die Kontakte gleichzeitig gelöst, wird das entgegengesetzte Zeichen des zuletzt ausgebenen Zeichens, angefügt. Hier ist das ein DIT.

3.1.3 Ultimatic

Beim Ultimatic Mode wird im Gegensatz zu den Iambic Modes bei der gleichzeitigen Betätigung beider Paddles ein DIT oder DAH und eine Folge des jeweiligen entgegengesetzten Zeichens erzeugt. Es wird zuerst das Zeichen erzeugt, dessen Paddle zuerst betätigt wurde. Im Anschluß wird kontinuierlich das Zeichen erzeugt, dessen Paddle zuletzt betätigt wurde.

Werden die Paddle gleichzeitig losgelassen, stoppt die Aussendung der Zeichen ohne die Ausgabe eines weiteren Zeichens wie beim Iambic Mode B. Wird ein Paddle losgelassen, wird die Aussendung mit dem noch betätigten Paddle fortgesetzt. Es kann somit eine ganze Folge von dem entgegengesetzten Zeichen in den Zeichenstrom des zuerst betätigten Paddles eingefügt werden. Damit können mehr Zeichen mit der Squeeze Technik gesendet werden, als es bei den Iambic Modes möglich ist.

Im Diagramm ist dieses Verhalten am Beispiel des Buchstabens P dargestellt. Zuerst wird das DIT Paddle betätigt, es wird ein DIT erzeugt. Noch während das DIT gesendet wird, kann das

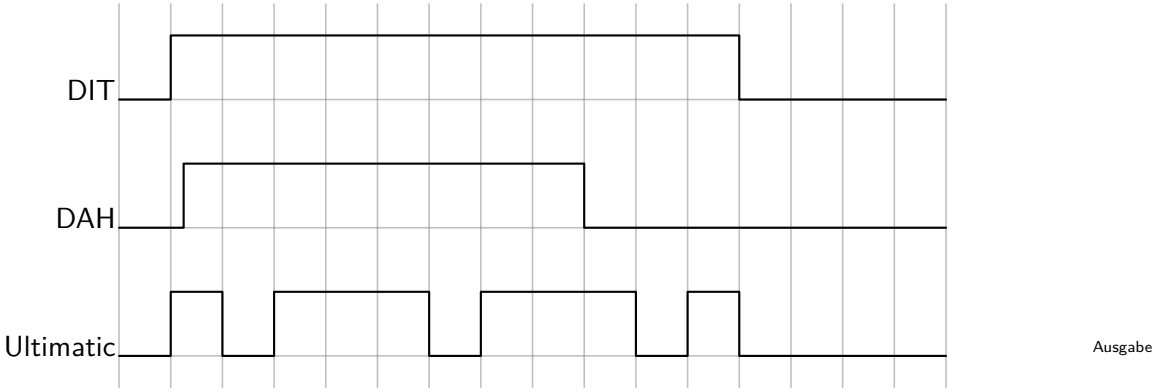

Abb. 3.3: Diagramm Ultimatic Mode

DAH Paddle betätigt werden und obwohl das DIT Paddle gehalten wird, erfolgt eine Aussendung von Strichen, so lange, wie das DAH Paddle gehalten wird. Wird der Kontakt nach zwei Strichen geöffnet, wird ein DIT gesendet, weil die DIT Taste immer noch gehalten wird. Nach dem Öffnen der Kontakte beide Paddles stoppt die Sendung sofort.

3.1.4 Gewichtung

Mit der Gewichtung der einzelnen Symbole kann die Länge von Punkten und Strichen verändert werden. Die Gewichtung verändert nicht die Gebegeschwindigkeit, weil im gleichen Maße einer Verkürzung von Punkten und Strichen, die Pausen verlängert werden.

Ein Wert kleiner 50 verringert die Gewichtung, ein Wert größer 50 erhöht die Gewichtung. Bei der Verringerung werden Punkte und Striche kürzer, die Pausen länger. Bei der Erhöhung werden Punkte und Striche länger, aber die Pausen kürzer.

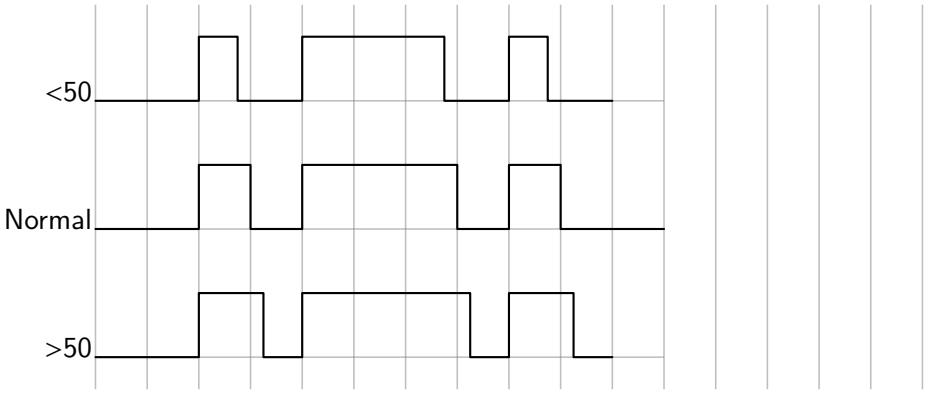

Abb. 3.4: Diagramm Gewichtung

3.1.5 Punkt/Strich Ratio

Das Ratio gibt das Verhältnis der Punktlänge zur Strichlänge an. Beim Standard Ratio von 1:3 ist ein Strich genau 3 Punkte lang. Vermindert sich das Ratio zu 1:2 wird der Strich um einen Punkt kürzer. Erhöht sich das Ratio zu 1:4, wird der Strich um einen Punkt länger. Die Länge der Pausen und der Punkte verändert sich nicht.

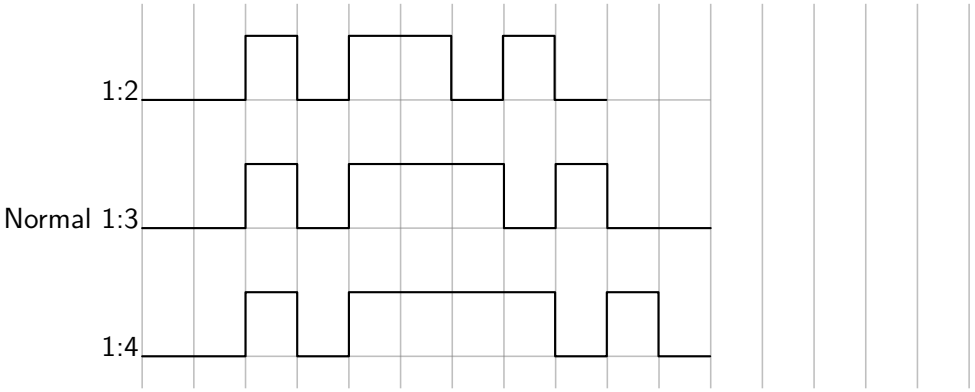

Abb. 3.5: Diagramm Punkt/Strich Ratio

3.1.6 Punkt/Strich Speicher

Der Punkt-Strich Speicher dient dazu, das der Keyer sich die Betätigung eines Paddles merkt, während ein Symbol ausgegeben wird. Nach Abschluss des Symbols wird dann das gemerkte Symbol ausgegeben.

Ein Beispiel mit dem Buchstaben Q:

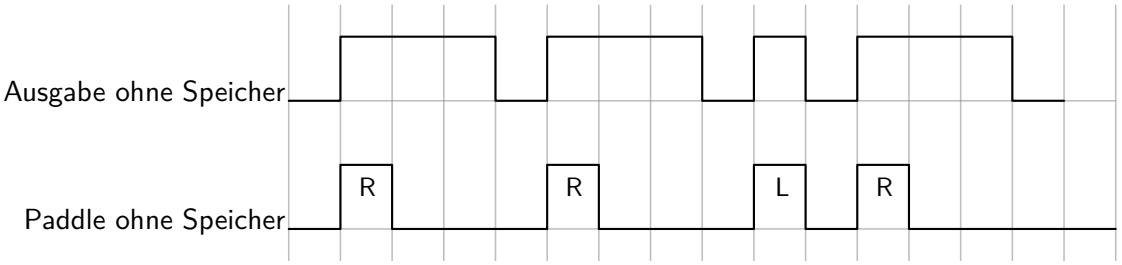

Abb. 3.6: Diagramm Punkt/Strich Speicher abgeschaltet

Das rechte Paddle wird betätigt und gehalten, bis der zweite Strich (Dah) ausgegeben wird. Am Ende des zweiten Strichs wird das linke Paddle betätigt und es wird ein Punkt ausgegeben. Das linke Paddle wird wieder losgelassen und das rechte Paddle betätigt, für den letzten Strich

vom Zeichen Q. Ohne Punkt/Strich Speicher müssen die Betätigungen der Paddles synchron zur Ausgabe der Zeichen sein.

Wird der Punkt/Strich Speicher verwendet ist das anders. Zuerst wird wieder das rechte Paddle betätigt und gehalten. Wenn nun aber während der Ausgabe des zweiten Strichs kurz das linke Paddle für einen Punkt (Dit) betätigt wird, so merkt sich der Keyer dies und gibt den Punkt nach der Ausgabe des zweiten Strichs aus, unter Beachtung des Timings, also der korrekten Punkt- und Strichlängen sowie Einhaltung der Pausen zwischen den Symbolen. Das rechte Paddle kann man während der Ausgabe des Punktes loslassen, da der Speicher aktiv ist und der Keyer sich auch den Strich gemerkt hat während der Punkt ausgegeben wird und diesen noch anfügt.

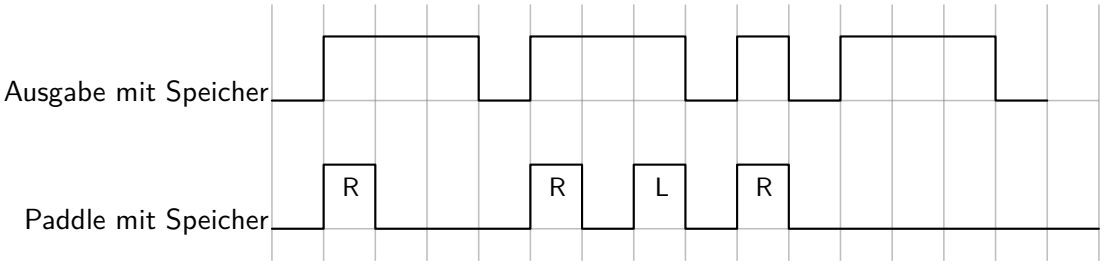

Abb. 3.7: Diagramm Punkt/Strich Speicher eingeschaltet

Mit dem Punkt/Strich Speicher können einige Zeichen einfacher gegeben werden, weil das Timing der Betätigung der Paddles unkritischer ist.

3.1.7 Handtaste

Die Handtaste wird vom Keyer nicht beeinflusst. Wird die Taste betätigt, wird sofort der Ausgang des Transceivers geschaltet. Es gibt für die Handtastenanschlüsse eine gemeinsame Entprellzeit, die in den Einstellungen bestimmt werden kann. Ich hatte beim Test Handtasten, die geprellt haben, d.h. während dem Betätigen der Taste wurden mehrere Impulse am Port registriert, weil die Taste langsamer schaltet, als der Port vom Controller geprüft wird. Um dies zu verhindern ist eine Einstellung möglich, um einen Impulswechsel am Port der Handtaste für einige Millisekunden nicht zu registrieren.

Die Anschlüsse für eine Handtaste sind nicht nur für die normale Handtaste (Straight Key, Klopftaste), sondern auch für eine Cootie oder Sideswipber und einen mechanischen Bug.

4 Die Bedienung

4.1 Bedienelemente

Als Bedienelemente stehen ein Drehencoder für diverse Einstellungen und ein Potentiometer für die Lautstärke des Mithörtons zur Verfügung. Optional kann ein Tastenfeld mit max. 5 Tastern für Zusatzfunktion wie Textspeicher angeschlossen werden. Zur Ausgabe von eingestellten Parametern wird ein 0,96" OLED Display mit 128x64 Pixeln verwendet.

4.2 Menuestruktur

4.3 Einstellungen

4.3.1 Transceiversteuerung

4.3.2 Mithörton

4.3.3 Iambic Modes

4.3.4 Punkt/Strich Speicher

4.3.5 Links- und Rechtshänder

4.3.6 Punkt/Strich Verhältnis

4.3.7 Punkt/Strich Verhältnis automatisch

4.3.8 Geschwindigkeitsanzeige

4.3.9 Anstiegszeit Mithörton

4.3.10 Entprellung für Handtasten

4.4 Einstellen der Geschwindigkeit

4.5 Lautstärke Mithörton

5 Die Schaltung

5.1 Spannungsversorgung

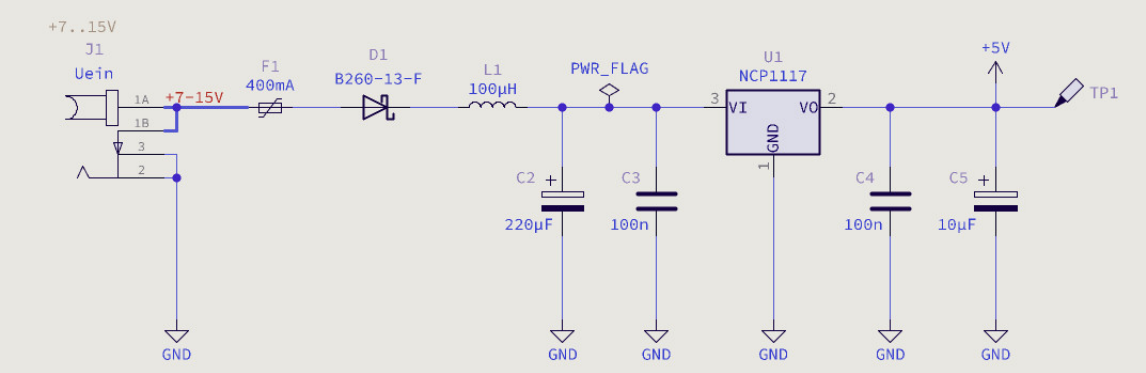

Abb. 5.1: Schaltung Spannungsversorgung

5.2 Mikrocontroller ATMega328P

5.3 USB Controller FT230

5.4 Class D NF-Verstärker

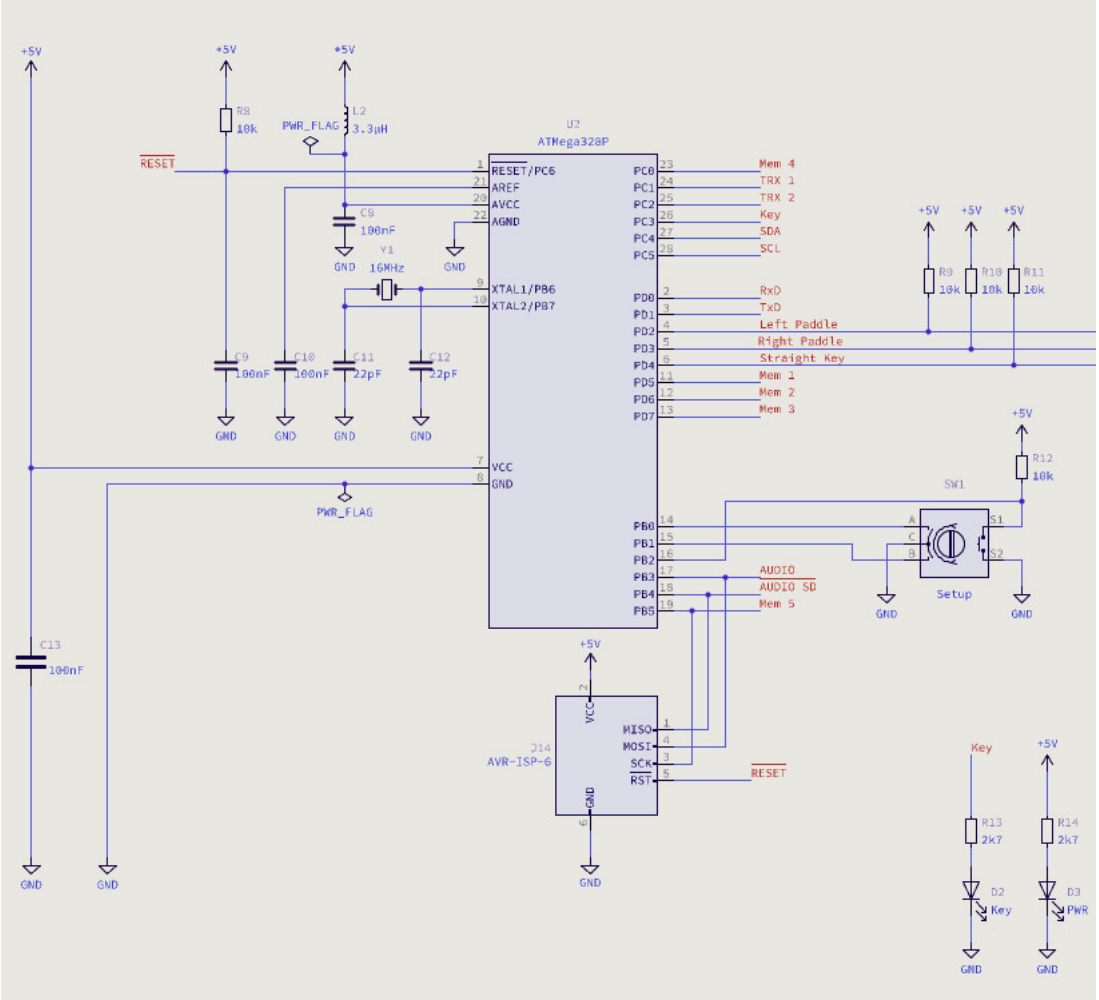

Abb. 5.2: Schaltung Mikrocontroller

5.4.1 Cauer-Filter

5.5 Class D NF-Verstärker

␣

5.6 Beschreibung

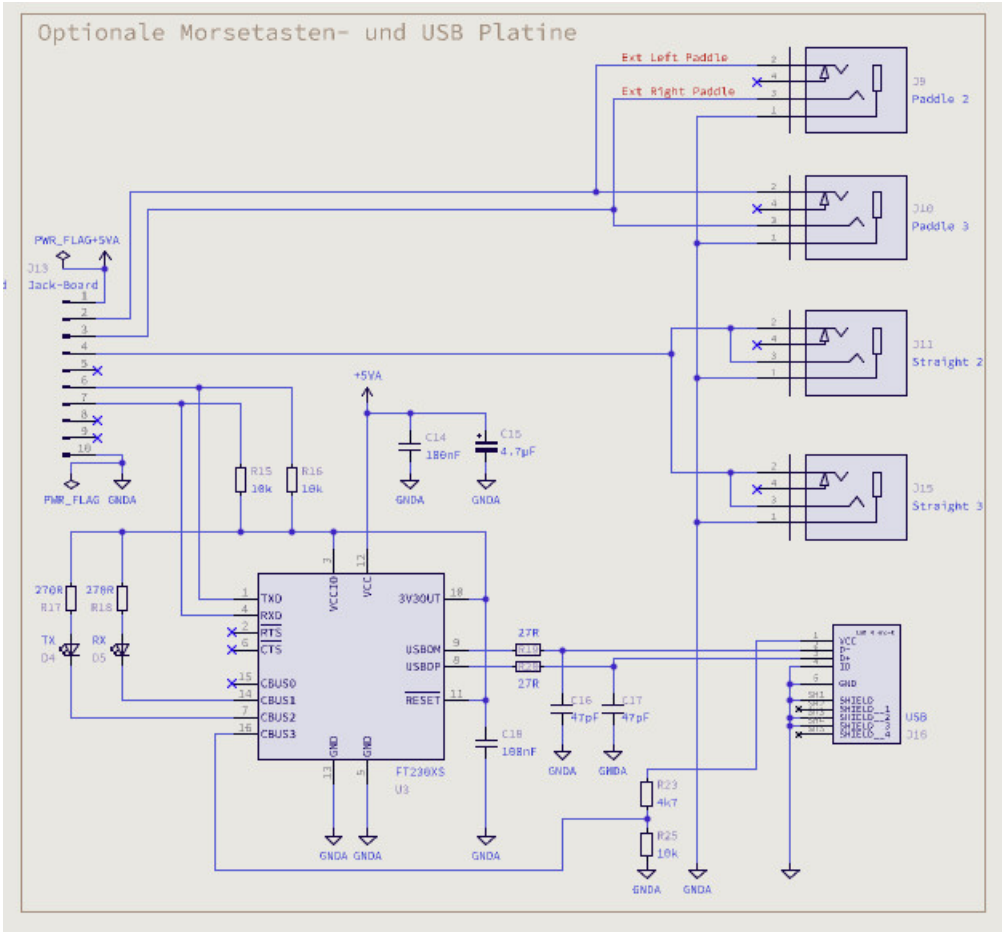

Abb. 5.3: Schaltung USB

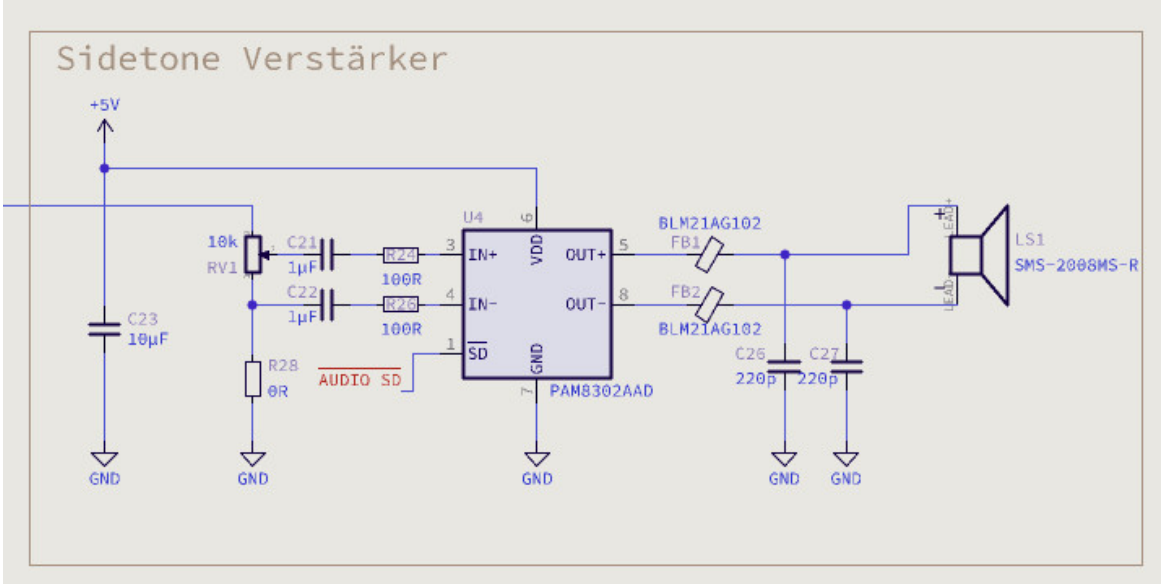

Abb. 5.4: Schaltung ClassD Verstärker

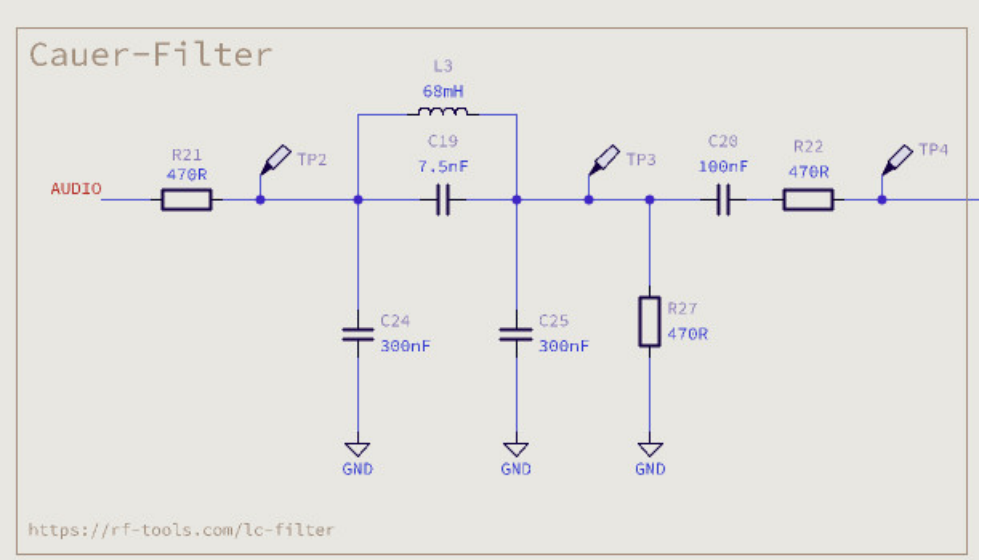

Abb. 5.5: Schaltung Cauer Filter

6 Die Software

6.1 Timer 2

Der Timer 2 läuft in der Betriebsart Pulsweitenmodulation und erzeugt das PWM Signal für den Mithörton.

6.2 Timer 1

Mit dem Timer 1 werden die Zeiten für die Nachladewerte für Timer 2 zur Erzeugung des Sinussignals für den Mithörton erzeugt.

6.3 Timer 0

Der Timer 0 läuft mit einem Takt von einer Millisekunde. Im Timer Interrupt werden 3 Zähler verwendet, so dass Zeiten von 1ms, 10ms und 20ms für diverse Abläufe zur Verfügung stehen.

6.3.1 Timer einstellen

$$f_{OCnA} = \frac{f_{clk\_I/O}}{2 \cdot \mathbb{N} \cdot (1 + OCRnA)} \tag{6.1}$$

6.4 Sinus Mithörton durch Pulsweitenmodulation

Der BJ-Keyer erzeugt einen Mithörton mit Sinuskurve, statt dem vielfach verwendeten Rechtecksignal. Der Klang eines Sinussignals ist angenehmer. Um mit dem Mikrocontroller ein Sinussignal zu erzeugen, wird die Pulsweitenmodulation verwendet.

6.4.1 Grundlagen

Die Pulsweitenmodulation, kurz PWM genannt, ist eine digitale Modulationsart, bei der eine Spannung zwischen zwei Werten wechselt.

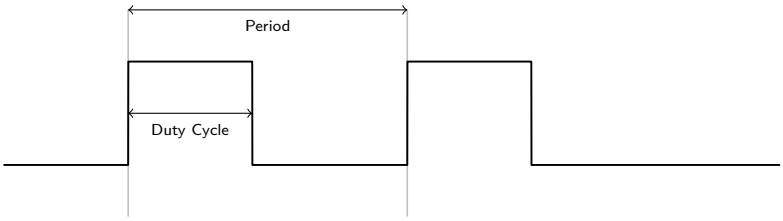

Abb. 6.1: Pulswellenmodulation

Mit einer konstanten Frequenz wird ein Rechteckimpuls moduliert, bei dem die Weite variert. Das Verhältnis zwischen Impuls und Pause wird Tastgrad (Duty Cycle) genannt.

Bei einer Rechteckschwingung gilt für den Tastgrad D:

$$D = \frac{\tau}{T} \tag{6.2}$$

mit τ als Impulsdauer und T als Periodendauer. Mit einem Tastgrad D = 0,5 = 50% würde ein symmetrischer Impuls erzeugt werden. Der Mikrocontroller schaltet den Ausgang zwischen V_{SS} und V_{DD}.

Die resultierende Ausgangsspannung berechnet sich wie folgt:

$$U_{Out} = \frac{\tau}{T} \cdot U_{In} \tag{6.3}$$

Dabei ist U_{In} gleich V_{SS}. Bei einem Tastgrad von 50% und einer Spannung V_{SS} von 5V beträgt U_{Out} = 2,5V. Je länger die Einschaltzeit ist, desto höher ist die effektive Spannung des erzeugten Rechtecksignals, bis zu V_{SS} bei einem Tastgrad von 100%.

Pulsweitenmodulation

Das PWM Signal wird mit Timer 2 des ATMega328P erzeugt. Das PWM Signal wird an PortB Pin 3, OC2A ausgegeben. Es wird der Fast PWM Mode 7 des Controllers verwendet, dabei ist der obere Wert des Timers der Wert im Register OCR2A. Der Ausgang OC2A wird auf den Ausgangswert Toggle konfiguriert, d.h. jedes Mal, wenn der Timer 2 den Wert in OCR2A erreicht, wird der Port umgeschaltet. Es wird ein Rechteck-Signal an PB3 erzeugt, dessen Tastgrad durch OCR2A eingestellt wird. Als Taktquelle wird der CPU Takt verwendet. Dies bedeutet, der Timer 2 läuft ohne einen Vorteiler.

Der maximale Wert für FastPWM berechnet sich wie folgt:

$$f = \frac{f_{Quarz}}{\mathbb{N} \cdot 256} \tag{6.4}$$

Der maximale Wert bei einem Quarz mit 16MHz und der minimalen Vorteilung von 1 beträgt somit:

$$\frac{16MHz}{1 \cdot 256} = 62,5\text{kHz} \tag{6.5}$$

Am Ausgang von PB3 liegt so bei einem Tastgrad von 50% ein symmetrisches Rechtecksignal mit 62,5kHz an. Der Effektivwert beträgt bei einer Betriebsspannung V_{SS} von 5V = 2,5V. Die 256-1 sind der maximale Wert (256 Werte von 0-255), den OCR2A haben kann (Timer 2 ist ein 8 Bit Timer). Das ist aber nicht das Ziel, da der Keyer ein sinusförmiges Signal ausgeben

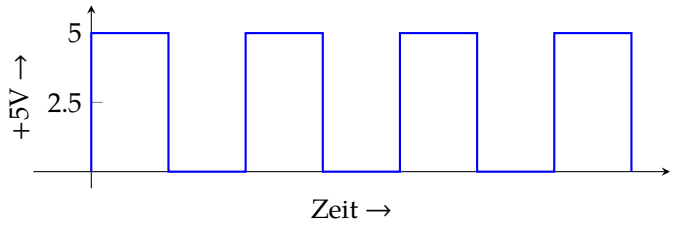

Abb. 6.2: Symmetrisches Rechtecksignal an PB3

soll. Um das zu erreichen, muss der Effektivwert der Rechteckspannung an PB3 veränderbar sein. Dies erreicht man durch eine Änderung des Tastgrades. Nun kann ein Mikrocontroller an einem digitalen Ausgangsport keinen Sinus erzeugen, einzig eine Treppe mit einer bestimmten Anzahl an Stufen, an- und absteigend ist möglich.

Wenn der obere Zählwert des Timers 2 in OCR2A verändert wird, ändert sich auch der Effektivwert der Rechteckspannung, durch die Änderung des Tastgrades. Läßt man OCR2A von 0 bis 255 zählen, ändert der Tastgrad sich von 0% bis 100%. Wenn dies über eine Zeitdauer τ durch Änderung von OCR2A passiert, dann steigt die effektive Spannung über diese Zeitdauer τ von 0V-V_{SS}.

$$V_{eff} = U_{max} \cdot \sqrt{Tastgrad} \tag{6.6}$$

Da der digitale Port nur zwischen Low und High wechseln und keine negativen Spannungen erzeugen kann, legt man eine virtuelle Nulllinie auf die Mitte, also auf 2,5V[1]. Die 2,5V werden bei einem Tastgrad von 50% erreicht, entsprechend einem Wert von 128 in OCR2A.

Erhöht man den Wert von OCR2A in Form einer Sinusfunktion von 128 auf 255 über eine Zeitdauer τ, ergibt sich eine ansteigende effektive Spannung in Form einer Sinusfunktion von 2,5V auf 5,0V. Verringert man den Wert von OCR2A von 255 auf 0 in Form einer Sinusfunktion, fällt

[1] es wird im weiteren Verlauf immer von V_{ss} = 5V ausgegangen

die effektive Spannung auf 0V. Durch die passende Änderung von OCR2A in Form einer Sinusfunktion über die Zeitdauer τ können somit Effektivspannungen mit 256 Werten dargestellt werden. Je mehr Werte es über die Zeitdauer τ sind, umso genauer ist die resultierende Hüllkurve in Form eines Sinus. Die Zeitdauer τ, mit der OCR2A mit den Werten einer Sinusfunktion

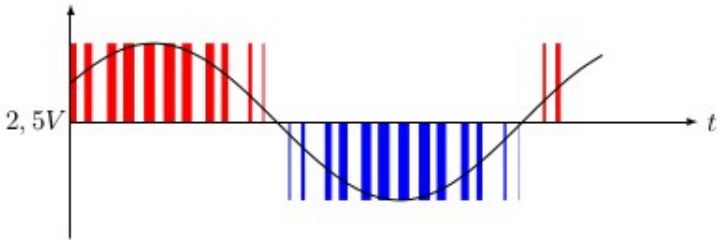

Abb. 6.3: PWM - Tastgrad - Sinus

geladen wird, legt die Frequenz des Mithörtons fest. Für die Zeitdauer τ wird ein weiterer Timer benötigt. Für einen Mithörton von 600Hz müssen 600 ·256 Werte pro Sekunde über einen Interrupt in OCR2A geladen werden. Je mehr Werte es sind, um so genauer ist die Kurvenform. Für den Timer kann die Zeit wie folgt berechnet werden:

$$600\text{Hz} \cdot 256 = 153,6\text{kHz} \tag{6.7}$$

Die Timer 1 läuft im CTC Modus und es wird ein Output Compare Match Interrupt ausgelöst. Das bedeutet, der Timer läuft bis zum Wert in OCR1A hoch und dann erfolgt der Interrupt. Der Wert für OCR1A wird wie folgt berechnet:

$$f_{OCR1A} = \frac{f\,clk_{I/O}}{\mathbb{N} \cdot (1 + OCR1A)} \tag{6.8}$$

Daraus folgt, daß sich ein Wert von

$$OCR1A = \left(\frac{16\text{MHz}}{8 \cdot 153,6\text{kHz}}\right) - 1 = 12 \tag{6.9}$$

12 für OCR1A ergibt. Allerdings bedeutet eine Frequenz von 153,6kHz für den Timer, dass alle 6,5µs ein Interrupt ausgelöst wird, bei 800Hz Mithörton sind es alle 5µs.

Der Controller läuft mit 16MHz, ein Taktzyklus dauert 62,5ns. Damit bleiben ungefähr 80 Taktzyklen für alle restlichen Aufgaben, wie Tasten abfragen, Display ansteuern, Drehgeber abfragen, CW Zeichen ausgeben. Das ist sehr wenig und die Gefahr besteht, dass Interrupts der Tasteneingänge verloren gehen[2]. Der einzige Weg zur Verlängerung der Zeit zwischen 2 Interrupts besteht in einer Verringerung der Werte für die Sinusfunktion. Mit den 256 Werten ist die Kurvenform zwar fein abgestuft, aber der Controller ist damit am Limit.

[2]was auch in der Praxis bei den Tests so war

Bei einer Verringerung auf 64 Werte für den Sinus ergibt sich dann bei 600Hz eine Zeit von 26µs, das entspricht ungefähr 416 Taktzyklen, was mehr als ausreichend ist. Das der Mithörton dadurch etwas rauher im Klang wird, muss das nachfolgende RLC Filter ausgleichen.

$$600\text{Hz} \cdot 64 = 38,4\text{kHz} \tag{6.10}$$

Mit einer Tabelle von 64 Werten ergibt sich für den Timer 1 eine Frequenz von 38,4kHz und für OCR1A ein Wert von 51 bei einem Prescaler von 8.

$$OCR1A = \left(\frac{16\text{MHz}}{8 \cdot 38,4\text{kHz}}\right) - 1 = 51 \tag{6.11}$$

Die Berechnung der Werte für OCR1A für unterschiedliche Frequenzen des Mithörtons erfolgt nach dieser Formel mit f_{Sinus} als gewünschte Frequenz des Mithörtons.

$$OCR1A = \left(\frac{f_{clk\_I/O}}{\mathbb{N} \cdot 64 \cdot f_{Sinus}}\right) - 1 \tag{6.12}$$

f_{Sinus}	OCR1A
1000Hz	30
800Hz	38
600Hz	51
400Hz	77

Tab. 6.1: OCR1A Werte für verschiedene Frequenzen des Mithörtons

6.4.2 Sinustabelle

Über den Overflow-Interrupt vom Timer 1 wird der jeweils nächste Wert einer Sinustabelle in OCR2A geladen. Die Sinustabelle wurde mit einem einfachen Python3 Script erzeugt.

Die 64 Werte vom Python3 Script ergeben sich wie folgt:

```
const unsigned char sinewave[] PROGMEM = {
0x80,0x8d,0x99,0xa5,0xb1,0xbd,0xc8,0xd2,0xdb,0xe3,0xeb,0xf1,0xf6,0xfa,0xfd,0xff, //  16
0xff,0xfe,0xfc,0xf8,0xf4,0xee,0xe7,0xdf,0xd6,0xcd,0xc2,0xb7,0xab,0x9f,0x93,0x86, //  32
0x7a,0x6d,0x61,0x55,0x49,0x3e,0x33,0x2a,0x21,0x19,0x12,0x0c,0x08,0x04,0x02,0x01, //  48
0x01,0x03,0x06,0x0a,0x0f,0x15,0x1d,0x25,0x2e,0x38,0x43,0x4f,0x5b,0x67,0x73,0x80  //  64
};
```

In dieser Grafik sind die 64 Werte als Stützpunkte eingezeichnet.

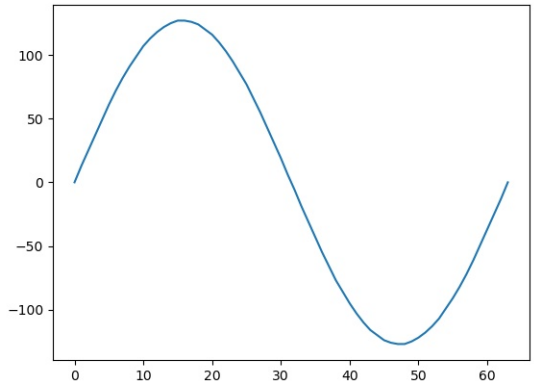

Abb. 6.4: Sinus nach Tabelle vom Python3 Script als Linie

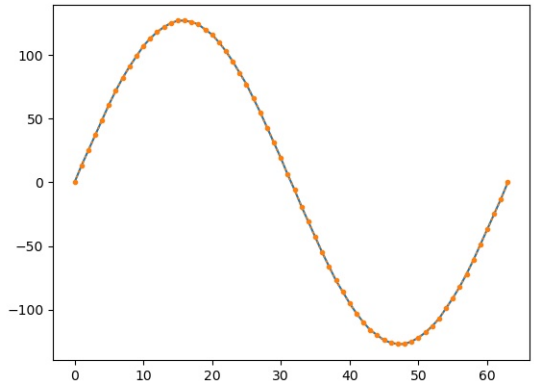

Abb. 6.5: Sinus nach der Tabelle vom Python3 Script mit Stützpunkten

7 Entwicklungsumgebung

Als Entwicklungsumgebung verwende ich mehrere, ausschließlich kostenfreie und überwiegend Open Source Programme:

- Editor neovim
- Shell bash
- Filemanager mc
- RS232 Terminal minicom
- Dokumentation lualatex
- PDF Reader zathura
- Compiler avr-gcc
- Flashprogrammer avrdude
- Layout & Schaltplan KiCad 7.xx
- Bohrschablonen FrontDesigner
- Softwareverwaltung Git
- Softwaredokumentation Doxygen
- Website Nginx & Dokuwiki
- Website Sourcecode Gitea
- Betriebssystem Entwicklung MX-Linux
- Betriebssystem Webserver Debian

Wie man sieht, sind das bis auf die CAD Anwendungen und dem PDF Reader alles Anwendungen für die Textconsole. Ich finde, richtig produktiv kann man nur mit der Textconsole arbeiten ;-)

Tabellen

Abbildungen