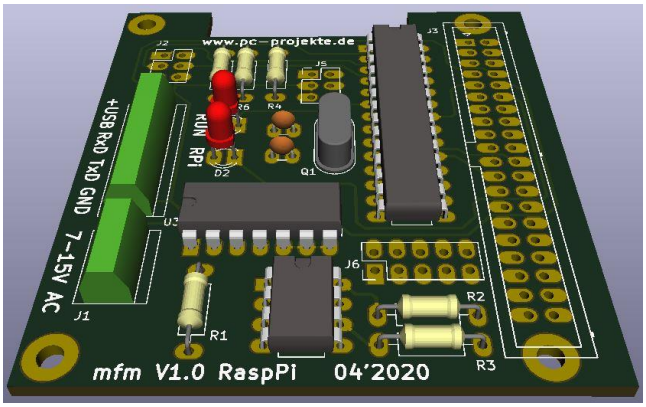
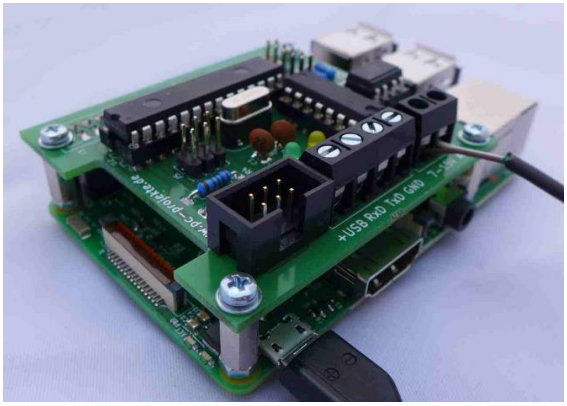
MAINS FREQUENCY MONITORING UNIT
Netzfrequenzmessung- und Überwachung für den RaspberryPi (und USB-TTL) erweiterte Dokumentation (Extendend Version)

Der Blick in unser Stromnetz





Projektbeschreibung MAINS FREQUENCY MONITORING V2.0
Extended Version

Messen und beobachten Sie Ihre Netzfrequenz. Die Abweichung vom Sollwert stellt einen direkten Qualitätsindikator über die angebotene Momentanleistung der Kraftwerke und die Abnahme der Verbraucher dar. Die Netzfrequenz ist somit ein Garant der Netzstabilität und zeigt Effekte des Intraday- Stromhandels

09.02.2021

Mueller, Jens, www.pc-projekte.de
pc_projekte.de@arcor.de

Inhaltsverzeichnis

Lizenz

Dokumentation

Creative-Commons-Lizenz

OpenHardware (OSHWA)

Die Hardware ist unter einer CERN-Lizenz veröffentlicht als OpenHardware (Open-Source-Hardware)

Das Projekt ist OSHWA- zertifiziert unter Nr. DE000086.

Gesamte Projektdokumentation und alle Lizenzen unter

https://gitlab.com/jm_wtal/mains-frequency-measure

Infos zu Oshwa: https://www.oshwa.org/definition/german

Software / Quellcode

Der Quellcode steht unter der GNU GPL 3-Lizenz:

- Freiheit das Programm für jeden Zweck auszuführen
- Freiheit den Quellcode zu studieren und anzupassen
- Freiheit, das Programm zu kopieren
- Freiheit, das veränderte Programm zu kopieren

Diese Freiheiten sollten auch langfristig sichergestellt werden. Zu diesem Zweck enthält die GPL zwei wesentliche Klauseln:

- Jedes Derivat muss ebenfalls vollständig unter der GPL lizenziert werden.
- Bei Weitergabe des Programmes in Binärform muss der Quellcode des gesamten Programms mitgeliefert oder auf Anfrage ausgehändigt werden.

Hintergrund

Unser Stromnetz

Die Netzfrequenz in Europa beträgt 50Hz. Eine Abweichung vom Nennwert ist ein direktes Fenster der elektrischen Erzeugung und dem Verbrauch. Der abgegebenen Leistung muss zu jedem Zeitpunkt eine gleich große Leistungsaufnahme gegenüberstehen.

Kommt es zu Abweichungen, führt das zu einer Veränderung der Netzfrequenz: Bei einem Überangebot von elektrischer Leistung kommt es zu einer Steigerung der Netzfrequenz, bei einem Unterangebot zu einer Absenkung. Normalerweise sind diese Abweichungen im westeuropäischen Verbundnetz minimal und bewegen sich im Bereich von 49,80Hz - 50,20Hz. Die Aufgabe der so genannten Leistungsregelung im Stromnetz ist es, die Lastschwankungen auszugleichen und so die Netzfrequenz auf konstantem Nennwert zu halten.

In Europa, Asien, Australien, dem Großteil von Afrika und Teilen von Südamerika wird für das allgemeine Stromnetz, in sogenannten Verbundnetzen, eine Netzfrequenz von 50 Hz verwendet. In Nordamerika verwendet man im öffentlichen Stromnetz eine Netzfrequenz von 60 Hz. Die Unterschiede sind durch die historische Entwicklungsgeschichte der ersten Stromnetze in den 1880er und 1890er Jahren bedingt und haben heute keinen technischen Grund.

Einige Eisenbahnen wie die ÖBB, SBB und die Deutsche Bahn nutzen für ihre Bahnstromversorgung eine nominale Frequenz von 16,7 Hz. Früher betrug die nominale Bahnnetzfrequenz 16 $^{2}/_{3}$ Hz, was genau einem Drittel der im Verbundnetz verwendeten 50 Hz entspricht. Einige Eisenbahnen und auch industrielle Abnehmer in Nordamerika werden aus historischen Gründen mit einer Netzfrequenz von 25 Hz versorgt. Die vergleichsweise niedrigen Netzfrequenzen resultieren aus der technologischen Entwicklung der ersten elektrischen Maschinen: Anfang des 20. Jahrhunderts konnte man elektrische Maschinen größerer Leistung nur mit diesen niedrigen Frequenzen bauen. Wegen des großen Umstellungsaufwandes werden jedoch die damals eingeführten niedrigen Netzfrequenzen auch noch heute beibehalten.

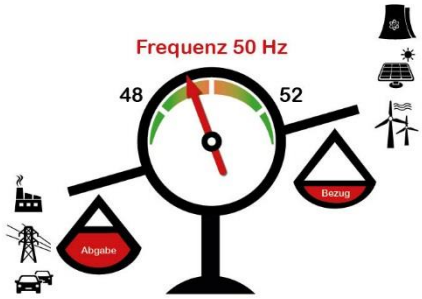
In speziellen Bereichen, z. B. im Bordnetz von Flugzeugen, sind höhere Netzfrequenzen üblich, z. B. 400 Hz, da sich dafür kleinere und leichtere Transformatoren bauen lassen und die Leitungslängen kurz sind.

Netzfrequenz als Qualitätsindikator

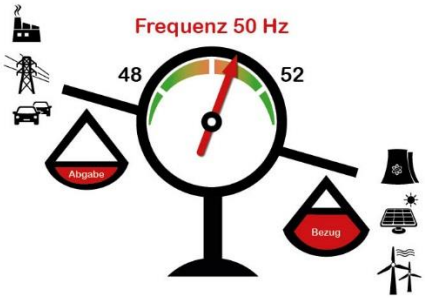
Die Netzfrequenz und deren Abweichung vom Nennwert ist ein direkter Qualitätsindikator über die Relation der über Erzeuger wie Kraftwerke angebotenen elektrischen Momentanleistung und der Abnahme der elektrischen Momentanleistung durch Verbraucher. Elektrische Energie kann in Verbundnetzen kaum gespeichert, sondern nur zwischen Erzeuger und Verbraucher verteilt werden. Der abgegebenen Leistung muss, bis auf die Blindleistung bei Wechselstrom, zu jedem Zeitpunkt eine gleich große Leistungsaufnahme gegenüberstehen.

Kommt es zu Abweichungen, führt das in Wechselspannungsnetzen zu einer Veränderung der Netzfrequenz: Bei einem Überangebot von elektrischer Leistung kommt es zu einer Steigerung der Netzfrequenz, bei einem Unterangebot zu einer Absenkung. Im Normalfall sind diese Abweichungen im westeuropäischen Verbundnetz minimal und bewegen sich unter 0,2 Hz. Eine Frequenzabweichung von 0,2 Hz entspricht im europäischen Verbundsystem einer Leistungsdifferenz von ca. 3 GW. Diese 3 GW Leistungspuffer sind als Referenzausfall (dem Continental Europe Operation Handbook) als Richtwert vorgegeben und beschreibt den ungeplanten Ausfall von zwei größeren Kraftwerksblöcken im europäischen Verbundnetz RG-CE (vormals UCTE). Die Aufgabe der Leistungsregelung in Verbundnetzen

ist es, die zeitlichen Schwankungen auszugleichen und so die Netzfrequenz auf konstantem Nennwert zu halten. Je kleiner ein Stromversorgungsnetz ist und je schlechter die Netzregelung funktioniert, desto stärkere Schwankungen treten bei der Netzfrequenz auf.



Unterangebot der Energie



Überangebot der Energie

Diese Schwankung kann man messen:

Ich möchte mit meinem Projekt die Dynamik des Stromnetzes sichtbar machen. Die Elektronik erfasst die Netzfrequenz und ermöglicht, abgeleitet über eine mathematische Funktion, die Anzeige der Netzlastdifferenz in MW (Megawatt). Diese Netzlastdifferenz entspricht der **primären Regelleistung**.

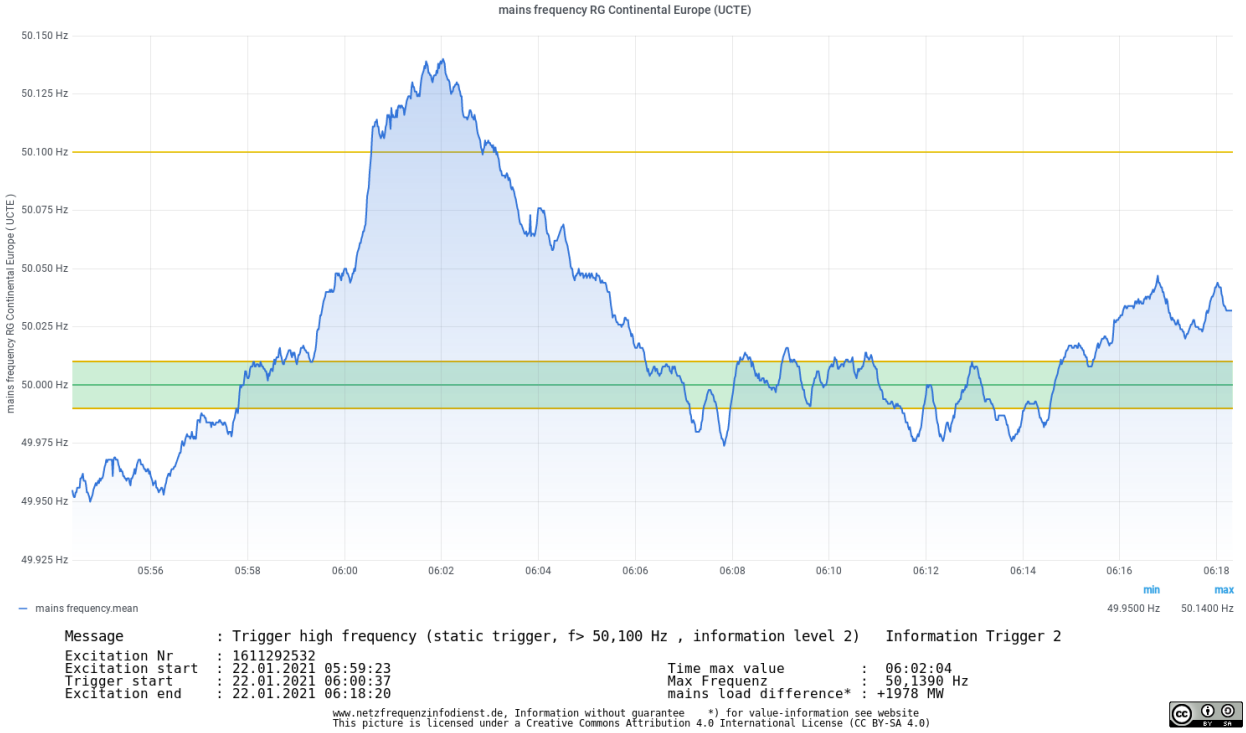


Bild: Netzfrequenzmessung (Beispiel grafische Auswertung mit Grafana®)

Es handelt sich um ein privates, nicht kommerzielles Projekt. Die Elektronik, µController- und Projektsoftware ist von mir entwickelt (siehe auch Creativ-Commons-Lizenz).

Platinen, Programmierservice und Bauteile siehe [Bestellmöglichkeit](#).

Einsatz der Netzfrequenzplatine

Messplatine zur Aufzeichnung, Informations-, Meldungs-, Alarmierungs-und Beobachtungszwecken der Netzfrequenz.

Das Gerät kann flexibel an die Kenndaten eines Stromnetzes angepasst werden (z.B. 60Hz-Netz). Eine Berechnung und Ausgabe der Netzlastdifferenz (primäre Regelleistung) ist vorbereitet (ENTSO-E RG-CE).

Die Ausgabe erfolgt über die serielle Schnittstelle. Die Platine ist für den RaspberryPi konzipiert, kann aber alternativ auch problemlso mit einen USB-TTL-Wandler verwendet werden.

Außerdem sind Output-Ausgaben für Alarmierungs- und Meldesysteme vorbereitet.

Eine Bereitstellung der Messwerte über eine Modbus-TCP-Software steht in Kürze zur Verfügung, weitere Protokolle z.B. MQTT (IoT), WebAPI (Influx-DB), IEC60870-4-104 FW-Protokoll oder als Logic-Device einer IEC61850-Anlage sind geplant.

Bauteile (Info)

Es ist keine smd-Technik verwendet, um die Nachbaubarkeit so einfach wie möglich zu gestalten. Die entsprechenden Nachbauanleitungen und Bauteillisten sind ab Seite 34 zu finden (Bestellmöglichkeit der Platine und der Bauteile siehe Seite 44). Eine teilbestückte Platine mit SMD ist in Vorbereitung.

Software

(µController ATMEGA328p-16PU DIP-28 8-Bit 10 MHz (Arduino - Kompatibel)

Die aktuelle Software steht auf der Gitlab-Seite als Download bereit.

Weitere Infos unter www.pc-projekte.de bzw. http://www.netzfrequenzanzeige.de.

Messspannung

Warnung: Verwenden Sie NIEMALS direkte 230V Netzspannung zur Netzfrequenzmessung!

Messspannung: Netzteil (oder Trafo) mit **Wechselspannungsausgang**

Bild: Beispiele für Klingeltrafos

Bild: Steckernetzteil (AC-AC)

Bild: Printtrafo

Siehe auch Hinweis / Gewährleistungsausschluss.

Frequenzmessung

Kurzbeschreibung Messablauf

Die Frequenzmessung erfolgt über eine Flankenauswertung des Wechselspannungssignals. Das Signal wird über zwei Optokoppler und einem RS-FlipFlop (4093-Nand-Gatter als RS-FlipFlop) dem µController zugeführt und mit dem 10MHz-Quarz-Takt und des Timer1 und Timer2 (als Counter) des µControllers ausgewertet.

Genauigkeit

Die Messgenauigkeit der Netzfrequenzmessung ist auch ohne Kalibrierung sehr hoch. Ausschlaggebend ist die Genauigkeit des Quarz Q_1 (und die korrekte Größe C_1 und C_2 , abgestimmt auf den Q_1). Der Messfehler der Frequenzmessung (ohne Kalibrierung oder Netzfrequenztrigger-Vorschaltplatine) liegt unter ± 1 mHz.

Eine Kalibrierung der Messung ist über eine Quellcode-Einstellung möglich.

Temperaturbereich

Die Netzfrequenzanzeige sollte bei Raumtemperatur verwendet werden. Direkte Sonneneinstrahlung auf das Gehäuse oder die Platine und die damit verbundene Temperatureinwirkung können das Messergebnis verfälschen.

Netzrückwirkungen

Die Netzfrequenzanzeige ist im Normalfall unempfindlich gegenüber in der EN50160 beschriebenen Netzrückwirkungen.

Sehr starke elektrische Impulse in der Umgebungen, ungewöhnlich hohe Oberwellenanteile oder Rundsteuersignale in der Messspannung (Netzrückwirkung) können das Messergebnis verfälschen.

Siehe auch Messdatenaufbereitung

Messbereichserkennung bei Programmstart

Beim Programmstart wird eine automatische Netzfrequenzerkennung durchgeführt.

Nach der erfolgreichen Erkennung werden die entsprechenden Grenzbereiche der erkannten Netzfrequenz in den Filterstufen verwendet.

Min/Max	50Hz-Netz	60Hz-Netz	16,7 Hz*
Max f	45 Hz	55 Hz	22 Hz
Min f	55 Hz	65 Hz	12 Hz

*) 16,7 Hz-Bahnnetzfrequenz z.B. Deutschland, Österreich und Schweiz (Normalspur)

16,7 Hz-Messung ist grundsätzlich möglich, ist aber momentan nicht integriert

Durch die manuelle Einstellung kann die entsprechende Netzfrequenz vorgegeben werden

Automatische Erkennung	50 Hz / 60 Hz (16,6 Hz)	**FA = 0**
50 Hz-Netz	45,000 Hz – 55,000 Hz	**FA = 50**
60 Hz-Netz	55,000 Hz – 65,000 Hz	**FA = 60**
Eine Version für 16,7 Hz ist vorbereitet	12,000 Hz – 22,000 Hz	**FA = 16**

Einstellungen für das Prescale-Messverfahren

Für das Messverfahren können einige Einstellungen vorgenommen werden, um Messdauer und Messperioden an den µController anzupassen bzw. zu optimieren.

Dazu können im µController-Programm einige Parameter einstellt oder geändert werden.

Grundeinstellungen im Arduino-Quellcode:

Einstellung	**Voreinstellung** (Empfehlung)	**Funktion**
Prescaler	64	Interner Teiler des µC-Taktes (ATMega328-Takt)
Messperiode	18	Maximale Periodenzählung (Flankenzähler bis nächste Frequenzberechnung)
Dividend	2812500	Konstanter Wert für Frequenzumrechnung (ergibt sich aus dem Prescaler und der Messperiode)
Sperr	2400	Sperrzeit nach erkannter Eingangsflanke zur Unterdrückung von Signalprellung
UBRR	0X40	Korrekturwert für serielle Schnittstelle (Anpassung der Baud-Rate an den 10MHz-Takt)
DiffMAX	60	Maximalwert der zulässigen Frequenzänderung (siehe Differenz-Komperator-Filter)
FA	0	Automatische oder manuelle Frequenzbereichseinstellung (Automatische Netzfrequenzerkennung bei Programmstart)
Calibration	0	Messwert-Kalibrierung, falls benötigt

Auf den nachfolgenden Seiten sind die Einstellungen detailliert beschrieben.

Deployment	µC Oscillating Quartz	Prescaler	Period	Dividend	Lock-Counts	UBRR (U2X=1)	Approxinate measuring time at 50Hz
USB-TTL A dapter (5V)	20 MHz	256	37	2890625	1200	0x81	0,74 s
	20 MHz	64	9	2812500	4800	0x81	0,18 s
	20 MHz	8	1	2500000	38402	0x81	0,02 s
	16 MHz	256	47	2937500	960	0x67	0,94 s
	16 MHz	64	11	2750000	3840	0x67	0,22 s
	16 MHz	8	1	2000000	30721	0x67	0,02 s
	12 MHz	256	62	2906250	720	0x4d	1,24 s
	12 MHz	64	15	2812500	2880	0x4d	0,3 s
	12 MHz	8	1	1500000	23041	0x4d	0,02 s
Rspberry Pi (3,3V)	10 MHz	256	75	2929688	600	0x40	1,5 s
	10 MHz	**64**	**18**	**2812500**	**2400**	**0x40**	**0,36 s**
	10 MHz	8	2	2500000	19201	0x40	0,04 s
	8 MHz	256	94	2937500	480	0x33	1,88 s
	8 MHz	64	23	2875000	1920	0x33	0,46 s
	8 MHz	8	2	2000000	15360	0x33	0,04 s
	6 MHz	256	125	2929688	360	0x26	2,5 s
	6 MHz	64	31	2906250	1440	0x26	0,62 s
	6 MHz	8	3	2250000	11520	0x26	0,06 s
	4 MHz	256	188	2937500	240	0x19	3,76 s
	4 MHz	64	47	2937500	960	0x19	0,94 s
	4 MHz	8	5	2500000	7680	0x19	0,1 s
	2 MHz	256	377	2945313	120	0xc	7,54 s
	2 MHz	64	94	2937500	480	0xc	1,88 s
	2 MHz	8	11	2750000	3840	0xc	0,22 s

<== Empfohlene Einstellung Raspberry Pi

- Quarz **10MHz**
- Prescaler **64**
- Period **18**
- Dividend **2812500**
- Lock-Counts **2400**
- UBRR **0x40**

(Programmierung über ISP-Port)

Tabelle: Übersicht optimierte Einstelldaten für die Netzfrequenzmessung

Einstellungen im Arduino-Programmcode (Basisquellcode):

```
// Basic-Settings / Grundeinstellungen
// ***********************************************************************************************
#define F_CPU 10000000
const word Prescaler = 64;                    // Prescaler (0,8,64,256,1024)
const byte Messperiode = 18;                  // Period
const unsigned long Dividend = 2812500;       // DIVIDEND Calculation constant
const word Sperr = 2400;                      // Lock-Counts
const int UBRR = 0x40;                        // Correction serial baud/clock
const int DiffMAX = 60;                       // DifferenzMAX
byte FA = 0;                                  // Selection 0 = Auto, 50=50Hz, 60=60Hz
const short Calibration = 0;                  // MeasureCalibration
//  ***********************************************************************************************
```

Alle Programmeinstellungen sind in den Zeilen 10 – 80 im Arduino-Quellcode (Basiscode und Extenden Version) zu finden.

Messung der Netzfrequenz

Grundprinzip sind die Timer T1 und T2 (T2 als virtueller 16-Bit-Counter) des µControllers, die ohne weitere Verarbeitung mit dem Start des µControllers auf den festen Takt 10MHz DIV **64** eingestellt sind. Die Timer werden mit dieser Einstellung der Referenztakt, aus dem die Netzfrequenz nun abgeleitet wird. Die zeitliche Gesamtlänge eines durchlaufenden Timers (0… 65535) beträgt ca 360ms (bei 50Hz). Die dadurch optimierte Periodenanzahl der Netzfrequenzmessung (für 45 Hz) beträgt **18**. Durch die alternierende Auswertung wird eine serielle Ausgabe der Netzfrequenz von ca. 180ms erreicht.

Berechnung der Netzfrequenz

Die Timer laufen im eingestellten Prescale-Verfahren mit dem (geteilten) Takt des µControllers als Counter.

Die Netzfrequenz wird über den Optokoppler und dem Schmitt-Trigger dem INT0 und INT1 des µController zugeführt. Ein erkanntes Triggerereignis führt zum internen Hochzählen des zugehörigen Periodenzählers.

Die Trigger-Sperrzeit (Sperr/ LockCounts **2400**) verhindert dabei eine zu schnelle Mehrfachauswertung der Flanke (z.B. durch ein Störsignal).

Ist der maximale Periodenzählwert erreicht, erfolgt die Übernahme des Zählwerts aus Timer1 oder Timer2 (T2 als virtueller 16Bit-Counter)

Der Counterwert der Timer wird als Divisor mit dem konstanten Dividend (**2812500**) die Berechnungsgrundlage für die Netzfrequenz. Zusätzlich wird noch der **Faktor 1000** multipliziert. Die Ausgabe der Netzfrequenz erfolgt dann in **mHz**. Das hat den Vorteil, dass sämtliche Netzfrequenzberechnungen bis zu diesem Zeitpunkt *ausschließlich in **Festkomma-Arithmetik*** ausgeführt werden. Dieser Vorgang wird daher im µC sehr schnell ausgeführt (dadurch ist das Grundprogramm sogar lauffähig auf einem ATTiny).

Berechnung:

$$\frac{50Hz}{56250\ Counts} = \frac{fMess}{Counts} \quad \longrightarrow \quad \frac{2812500*1000}{Counts} = fMess$$

fMess (Variablenbezeichnung im Quellcode) **in mHz** (Bsp. 50021 = 50,021 Hz)

Der vorläufige Frequenzmesswert fMess (MessA / MessB) wird nun über einige Filterstufen geführt.

Die Ausgabe (Frequenz) erfolgt als serielles Protokoll auf dem TxD-Port des µControllers als alternierende Ausgabe der Ergebnisse T1 und T2.

UBRR (Anpassung an 10MHz)

Um die 19200Bd-Ausgabe der seriellen Schnittstelle auf den 10MHz-Quarz anzupassen, muss über die Eingabe der UBRR-Registerwert angepasst werden.

Für einen 10MHz-Quarz muss **UBRR = 0x40** eingestellt werden.

Detaillierte Beschreibung der Messung

Eingangsbeschaltung Schmitt-Trigger

Die Eingangsbeschaltung (Netzfrequenz-Messeingang) ist über 2 Optokoppler galvanisch getrennt. Die Messspannung wird über ein Trafo zugeführt und sollte zwischen **7V- 15 V ~ AC** liegen. Der Strom der verwendeten Messspannung beträgt nicht mehr als 20mA. Zur Messung sollte ein kurzschlussfester Trafo oder ein Klingeltrafo verwendet werden. Die Eingangsbeschaltung der Optokoppler U1 (U1a/U1b) wird durch R_1 **strombegrenzt** (siehe auch Informationen Bauteile Seite 34).

Die Optokoppler steuern über ein NAND-Gatter aufgebauten Schmitt-Trigger an (CMOS 4093, IC U3), der dann das Messsignal als Rechtecksignal dem INT0- und INT1- Eingang des µControllers weiterleitet.

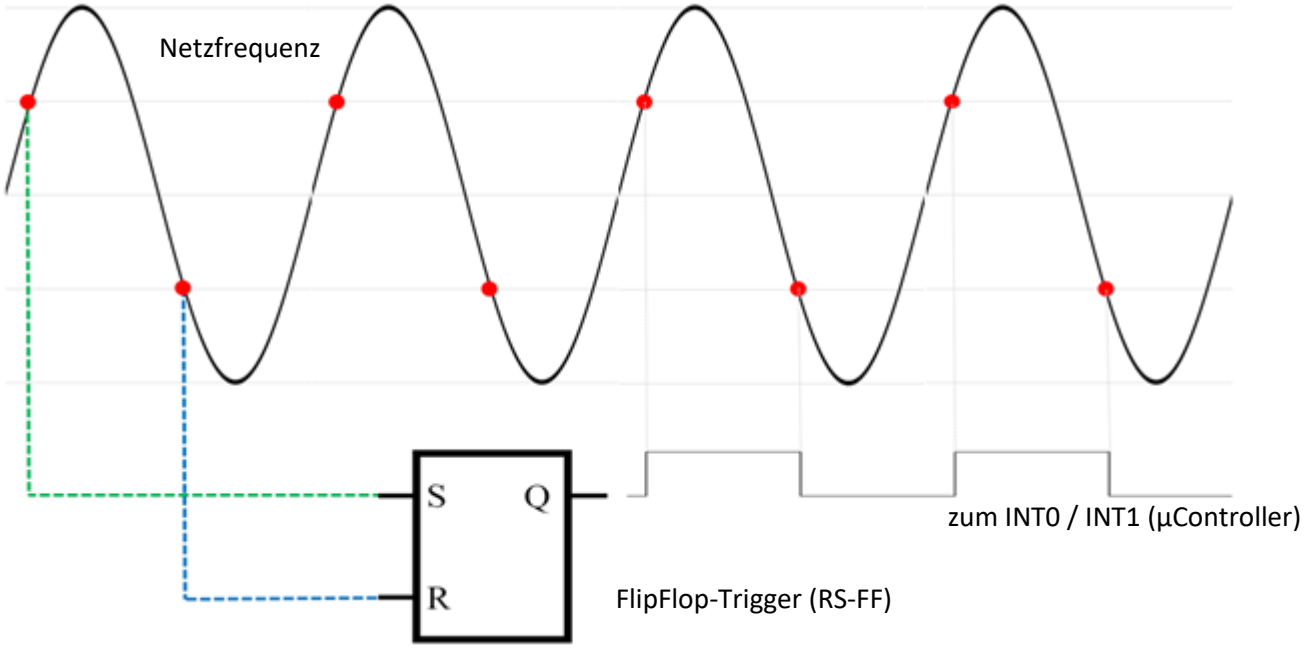


Bild: Schematische Darstellung der Flankenauswertung (Schmitt-Trigger)

Vorteil dieser Messdatenaufbereitung ist, dass auch **ein sehr verrauschtes Eingangssignal sicher erfasst wird.**

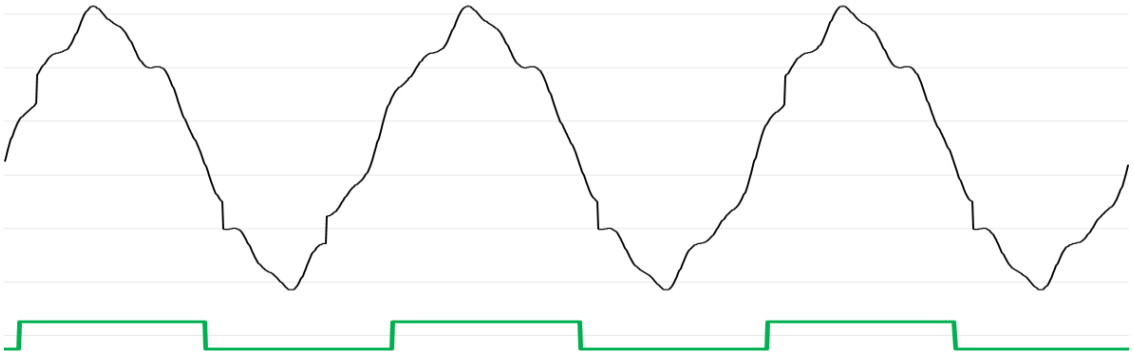
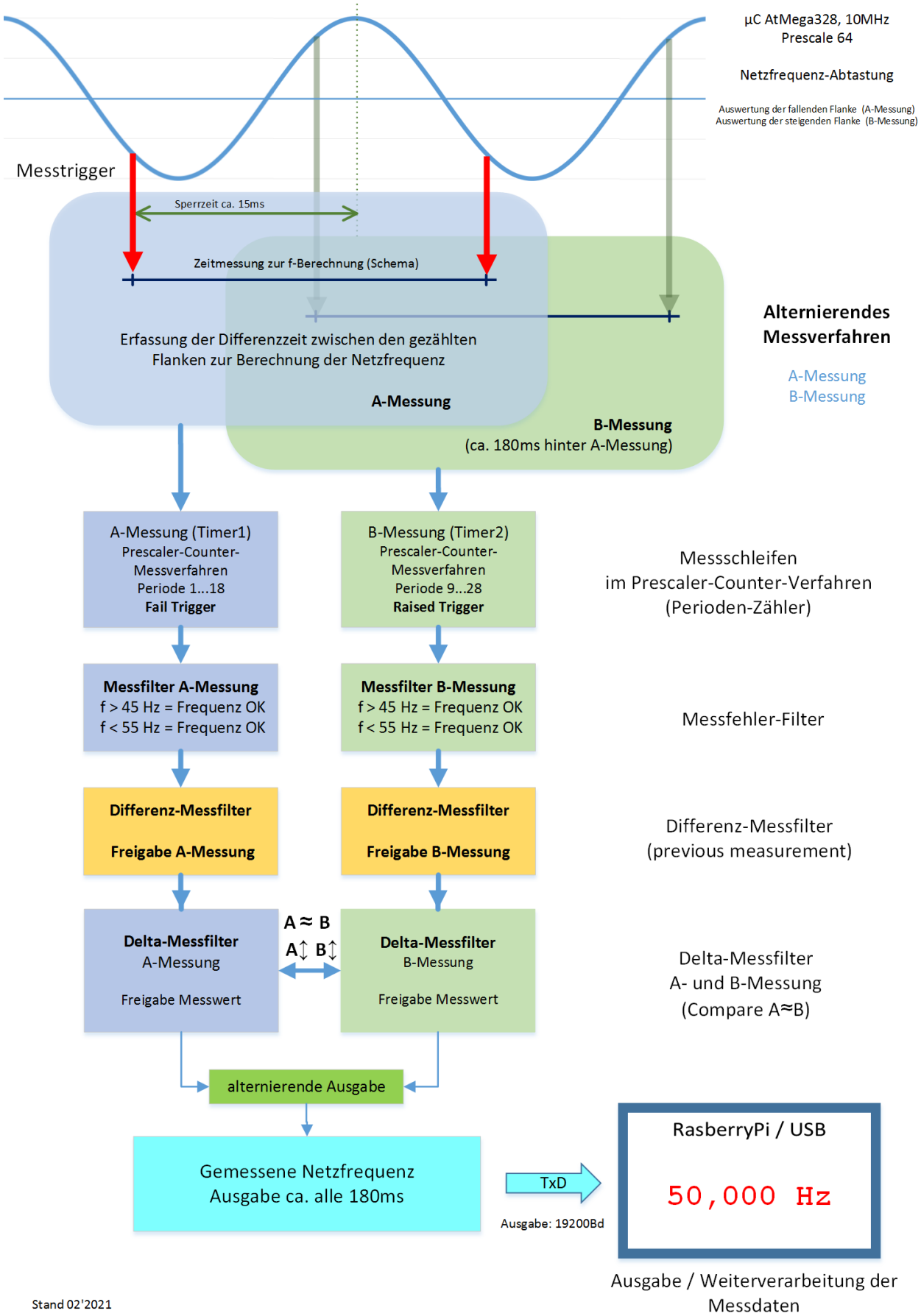


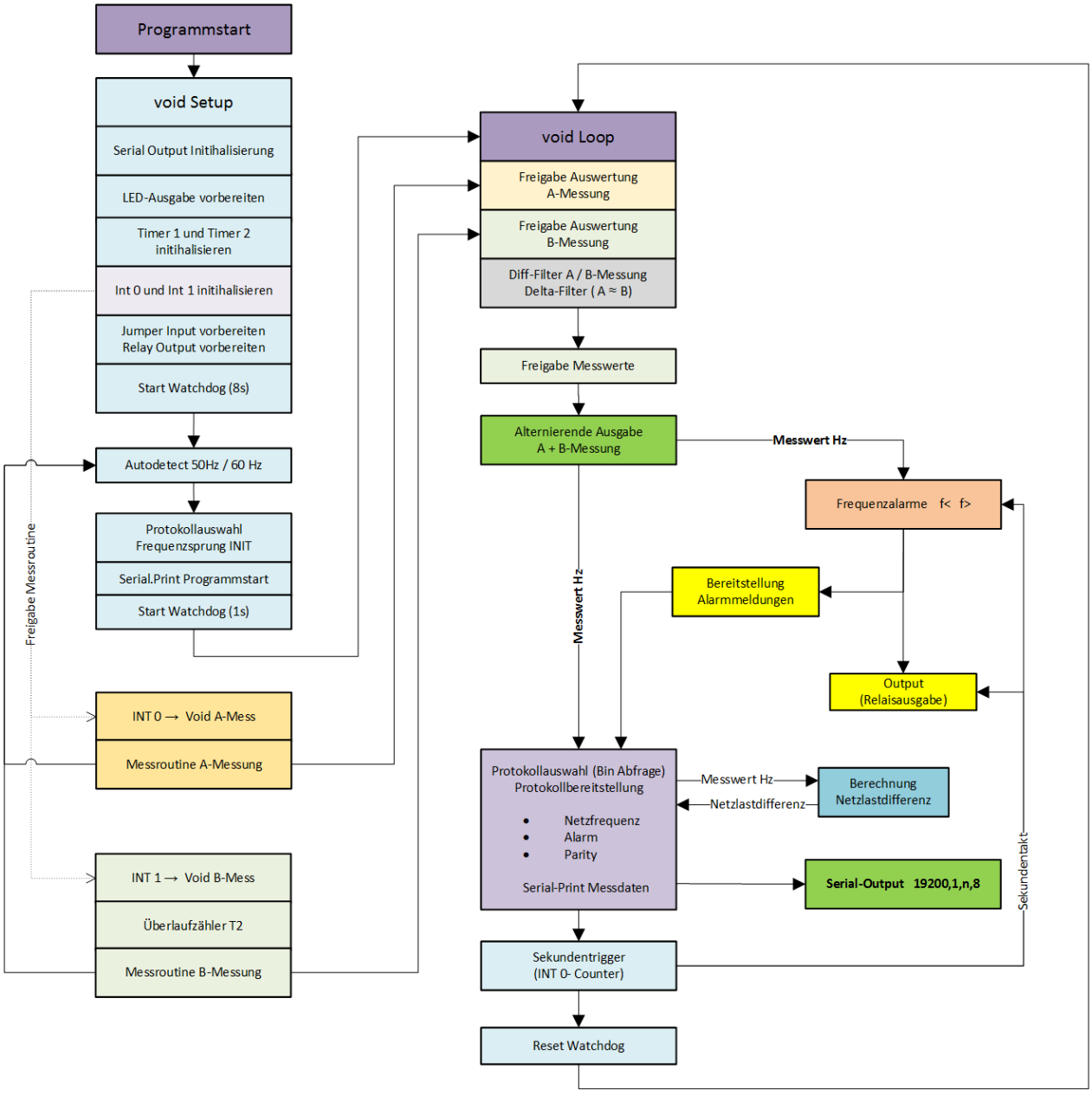
Bild: Schematische Darstellung eines verrauschtes Eingangssignal und mit Ausgangssignal (hinter dem Schmitt-Trigger)

Schematischer Programmablauf Messauswertung



Stand 02'2021
www.pc-projekte.de , www.netzfrequenzanzeige.de

Arduino-Programm Ablauf (extended Version)



Auswertung mit dem µController INT0 und INT1

Flankenerkennung und Triggersignal-Sperre eines Timers:

Erkennt der µController eine Flanke, wird sofort eine Sperrzeit (ca. 15ms) gestartet, um nachfolgende Messspitzen zu unterdrücken. Eine mögliche zu frühe Flanke wird durch die nachfolgende Stufe sofort kompensiert, da nun eine Mittelwertbildung folgt.

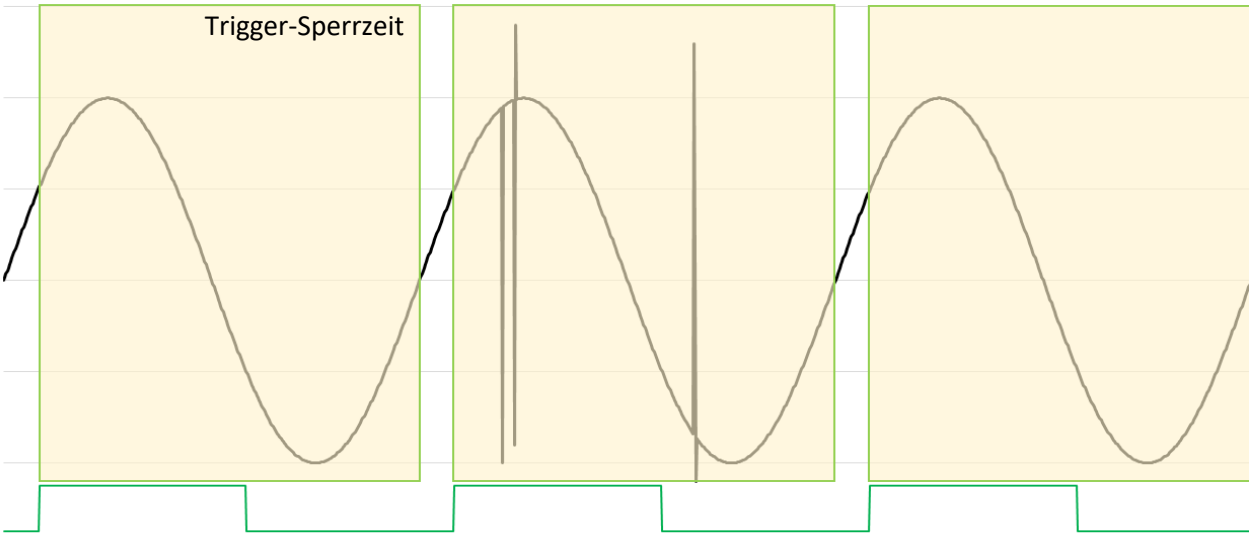


Bild: Schematische Darstellung der Trigger-Sperrzeit (T1)

Sperrzeit / Lock-Counts

Die Sperrzeit nach einem Impuls ist so gewählt, dass das nächste Triggersignal nur mit einer Frequenz unter 65 Hz passen würde.

Counter T2 als virtueller 16Bit-Counter

Der Counter T2 des ATMega328p ist ein 8-Bit-Counter. Über eine erweiterte Zählfunktion, bei dem die Überlaufereignisse des Counters T2 gezählt und in einem virtuellen T2-Zählerspeicher addiert werden, steht ein weiterer 16Bit-Counter zur Verfügung.

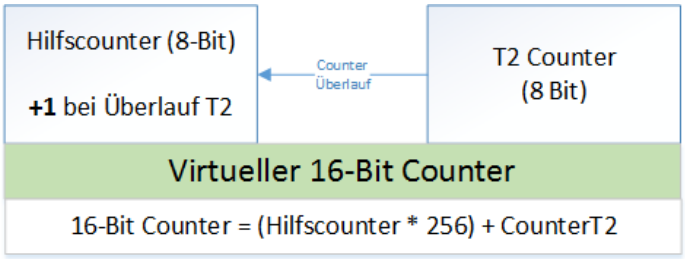
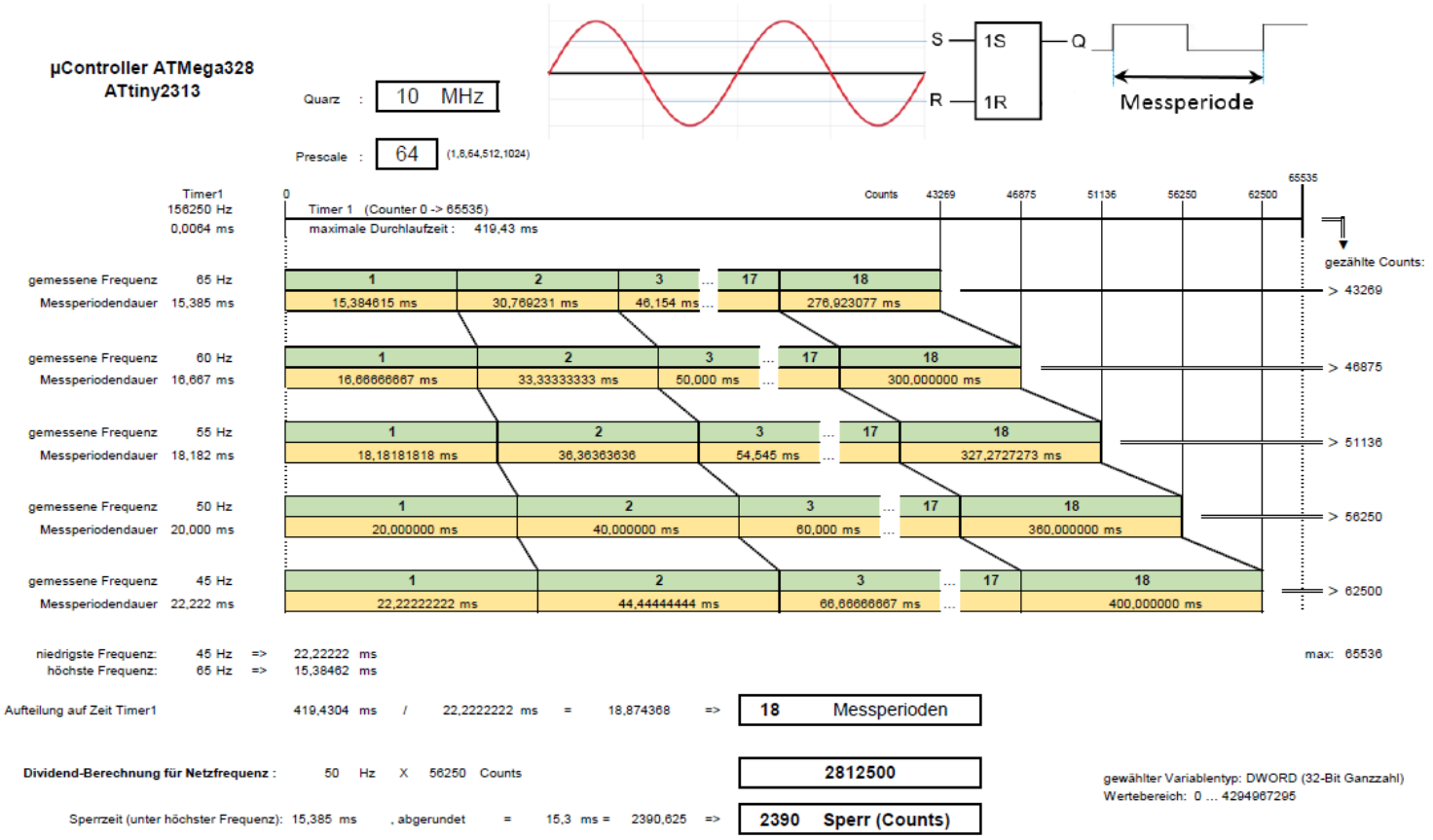


Bild: Schematische Darstellung „virtueller 16-Bit-Counter T2“

Auswertung der Messcounts T1 und T2 (virtuell)



Plausibilitätsfilter

Bevor der gemessene Netzfrequenzwert weiterverarbeitet wird, durchläuft der vorläufige neue Messwert noch eine zweistufige Prüfung: Dabei wird festgestellt, ob der gemessene Netzfrequenzwert grundsätzlich im zulässigen Bereich liegt:

Messung	50 Hz-Messung	60Hz-Messung	16,7 Hz*
Prüfung f>	f > 45 Hz	f > 55 Hz	22 Hz
Prüfung f<	f < 55 Hz	f < 65 Hz	12 Hz

Nach dieser ersten Überprüfung, der unplausible Messwerte ausschließt, geht es weiter zum Differenz-Filter

Differenz-Komparator-Filter

Funktionsweise des Differenz-Komparator-Filters:

Der Differenz-Komparator-Filter verwendet prinzipiell die **Vorgabe der maximalen Netzgradiente**. Eine Netzfrequenzänderung über der maximalen Netzgradiente wird nur bedingt berücksichtig.
Sollten aber zwei aufeinanderfolgende Messungen an der Gültigkeitsprüfung auflaufen, wird auch eine sehr schnelle Änderung sofort übernommen. Die Voreinstellung DiffMAX (**100mHz/s**) ist zugleich auch die maximale Frequenzausgabeänderung.
Beim letzten großen Frequenzeinbruch am 8.1.2021 lag die Frequenzänderung von 50,020 Hz auf 49.739 Hz innerhalb 13s bei ca. **22mHz/s**.

Ablauf: Der vorläufige (neue) Messwert wird mit dem letzten gültigen Messwert verglichen.
Durch die Differenzbildung der beiden Messwerte wird die Differenzgröße ermittelt.

$$Diff = ABS\,(f_{preliminary} - f_{Mess})$$

Ist die Differenz (Absolutwert) kleiner der Voreinstellung DiffMAX, wird der neue Wert (Preliminary Value) zum gültigen Messwert f_{Mess}.

Ist die Differenz größer, wird der aktuelle Messwert f_{Mess} in Höhe der Voreinstellung DiffMAX dem neue Wert (Preliminary Value) nachgeführt.

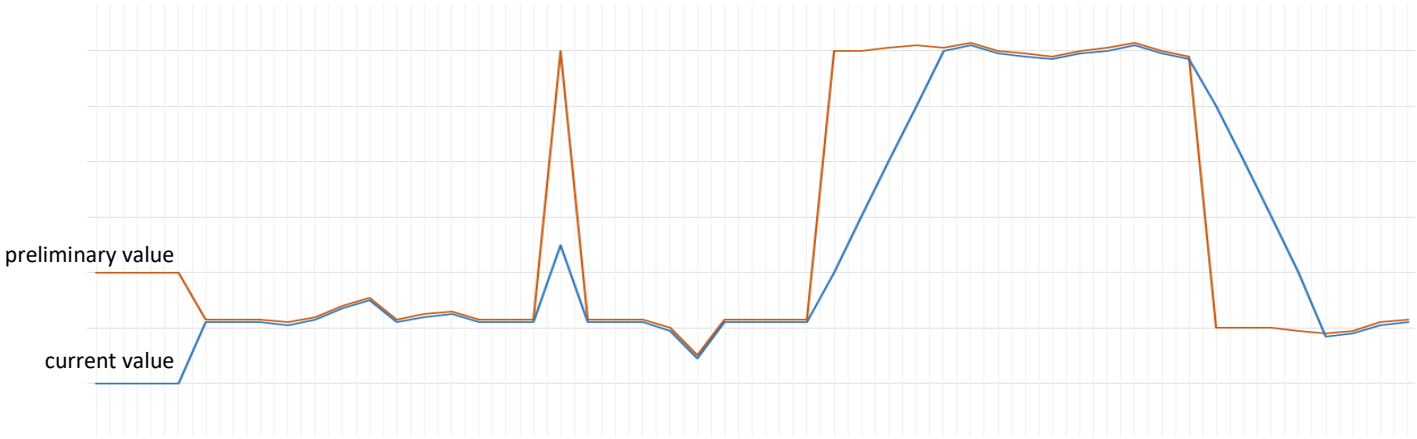


Bild: schematische Darstellung Verlauf Messdaten Differenz-Komparator-Filter

Die Netzgradiente hängt von vielen Netz-Schaltungszuständen, Einspeisungen, Netzlastsituationen und des Messortes ab. Deshalb ist der DiffMAX-Wert vom 100mHz/s eine Einstellungsempfehlung für das RG-CE (UCTE)-Netz.

Die Voreinstellung des DiffMAX-Filters beträgt **100mHz/s** (Empfehlung).

Programm Diff-Filter (Arduino-Programm)

```
word Diff_filter(word filter_wert)              // Messdaten in mHz im WORD-Format
{
        int Diff = filter_wert – Cachewert;
        if ( abs(Diff) > DiffMAX )              // bei Überschreitung DiffMAX
        {
                if ( Diff > 0) filter_wert = Cachewert + DiffMAX;       // Änderung (positive Steigung) max
                if ( Diff < 0) filter_wert = Cachewert – DiffMAX;       // Änderung (negative Steigung) max
        }
        Cachewert = (int)filter_wert;
 return filter_wert;
}
```

Kalibrierung

Normalerweise braucht die Netzfrequenzplatine keine Kalibrierung.

Sollte aber eine Kalibrierung doch erforderlich sein, kann in der Einstellung ein Kalibrierungswert beigefügt werden. In Kürze können verschiedene Einstellungen ein Softwaretool (über eine EEP-Datei) geänderte Einstellung ins EEProm des ATMega328 übertragen werden. Nach einen Neustart werden die geänderten Einstellungen verwendet.

Frequenz + Calibration → Ausgabe zur seriellen Schnittstelle

Einstrellung serielle Schnittstelle

Einstellung der seriellen Schnittstelle			
Baudrate: **19200 Baud**	Parity: **None**	Stoppbit: **1**	Handshake: **None**

UBRR = 0x40

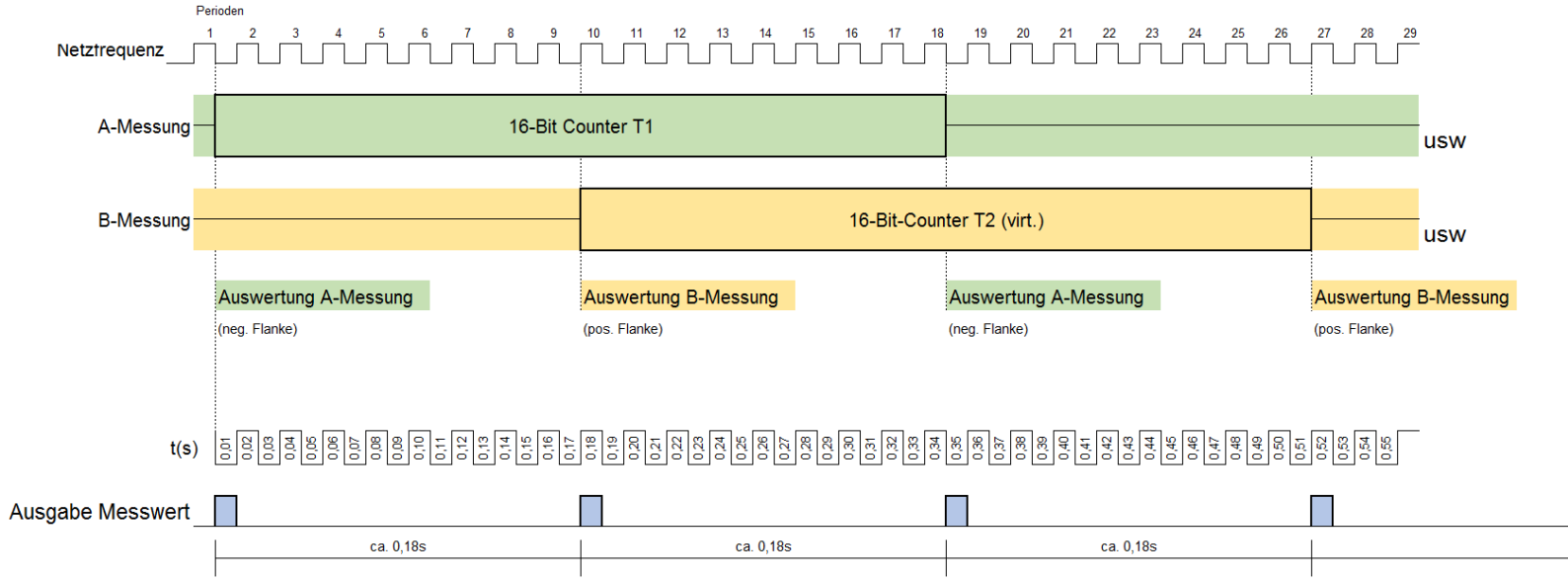
Um die Ausgabe der seriellen Schnittstelle auf den 10MHz-Quarz anzupassen, muss über die Eingabe der UBRR-Registerwert angepasst werden. Für einen 10MHz-Quarz muss UBRR = 0x40 eingestellt werden.

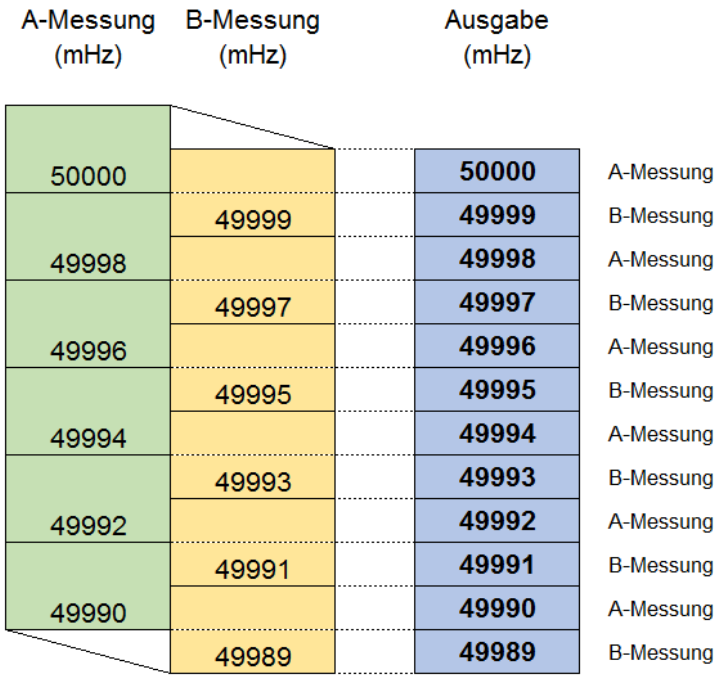
Netzfrequenzmessung Extended-Version

Alternierende Messdatenausgabe A- und B-Messung

Die Ausgabe der Messdaten der A- und B-Messung erfolgt alternierend. Während die A-Messung die Berechnung und Ausgabe ausführt, misst die B-Messung bereits im Hintergrund den nächsten Messwert.

Beide Messungen kontrollieren sich gegenseitig und überwachen somit die messfehlerfreie Ausgabe. Zu große Abweichungen im Messdatenvergleich A-Messung/B-Messung blockiert die Ausgabe zur weiteren Verarbeitung.





Erweiterte Einstellungen im Arduino-Quellcode

Einstellung	**Variable (Name)**	**Voreinstellung** (Empfehlung)	**Funktion**
Kennlinientyp	**Typ**	**1**	Auswahl Kennlinientyp für die Ausgabe der Netzlastdifferenz
Netzlastdifferenz **λ**	**Lamda**	**15,333**	Steigung Netzlastdiffernz-Kennlinie
Offset (in mHz)	**Offset**	**10 (±)**	Offset Netzlastdifferenz-Kennlinie
Negierung Kennlinie	**Neg**	**false**	Vorzeichennegierung Ausgabe Netzlastdifferenz
f1 > (in mHz)	**f1g**	**50100**	Frequenzalarm 1 (=> 50,100 Hz)
f1 < (in mHz)	**f1k**	**49900**	Frequenzalarm 1 (=> 49,900 Hz)
f1 Haltezeit nach Rückfall (in s)	**tf1**	**10**	Haltezeit Alarmausgabe nach Anrege-Rückfall Alarm f1
f2 > (in mHz)	**f2g**	**50200**	Frequenzalarm 2
f2 < (in mHz)	**f2k**	**49800**	Frequenzalarm 2
f2 Haltezeit nach Rückfall (in s)	**tf2**	**10**	Haltezeit Alarmausgabe nach Anrege-Rückfall Alarm f2
f3 > (in mHz)	**f3g**	**51000**	Frequenzalarm 3
f3 < (in mHz)	**f3k**	**49000**	Frequenzalarm 3
f3 Haltezeit nach Rückfall (in s)	**tf3**	**10**	Haltezeit Alarmausgabe nach Anrege-Rückfall Alarm f3
f4 > (in mHz)	**f4g**	**52000**	Frequenzalarm 4
f4 < (in mHz)	**f4k**	**48000**	Frequenzalarm 4
f4 Haltezeit nach Rückfall (in s)	**tf4**	**10**	Haltezeit Alarmausgabe nach Anrege-Rückfall Alarm f4
LowAktiv	**LowAktiv**	**true**	Änderungsmöglichkeit für eine HighAkive Meldungsausgabe (Binärausgabe J6)
BlinkON	**BlinkON**	**false**	Ausgangssignale J6 werden als Blinksignale ausgeführt (z.B. für eine Warnmeldung /Warnlampe usw.)

Auf den nachfolgenden Seiten sind die Einstellungen detailliert beschrieben.

Ausgabe der Netzlastdifferenz

MW-Funktion

Funktionsgleichung MW / mHz mit oder

Unempfindlichkeitsbereich (Offset)

$$\lambda = \frac{\Delta P(MW)}{\Delta \mathrm{f} \ \pm Offset\ (mHz)}$$

Die Funktion entspricht einer linearen Gleichung, allerdings mit einem Unempfindlichkeitsbereich (Offset / Totbereich), der ebenfalls eingestellt werden kann. Zur besseren Übersicht und Einstellung ist die Kennlinie als Grafik sichtbar.

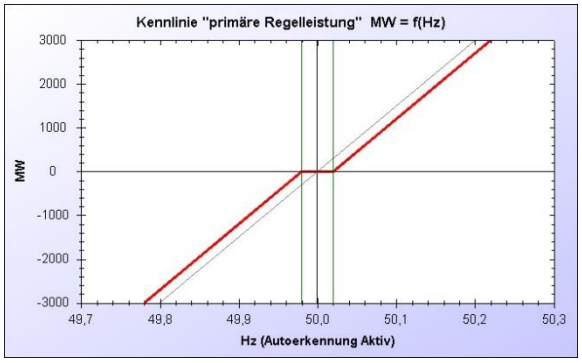
Kennlinientyp

Die Voreinstellung ist **Kennlinientyp 1, λ = 15,333** (Europa)

1. **Unempfindlichkeitsbereich I / offset range I:**

Die lineare Funktion startet außerhalb des Unempfindlichkeitsbereichs (Tot-Bereich) und verläuft parallel zur

$$\boldsymbol{\lambda} = \frac{\Delta\, P(MW)}{\Delta f\ (mHz) \pm Offset(mHz)}$$

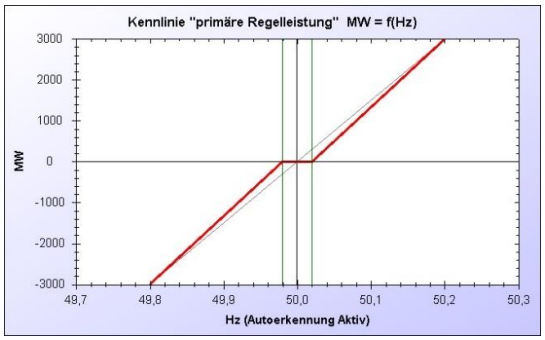


in Vorbereitung, noch nicht integriert:

2. **Unempfindlichkeitsbereich II / offset range II:**

Die lineare Funktion startet außerhalb des Unempfindlichkeitsbereichs (Tot-Bereich)

$$\boldsymbol{\lambda} = \frac{\Delta\, P(MW)}{\Delta f\ (mHz) \pm Offset(mHz)}$$

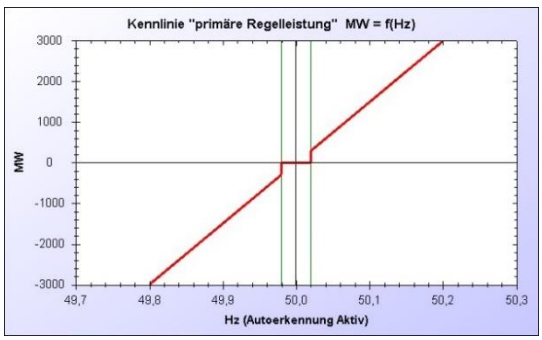


(frequency offset / sensitivity range or step behave).

3. **Rampenfunktion / step behave:**

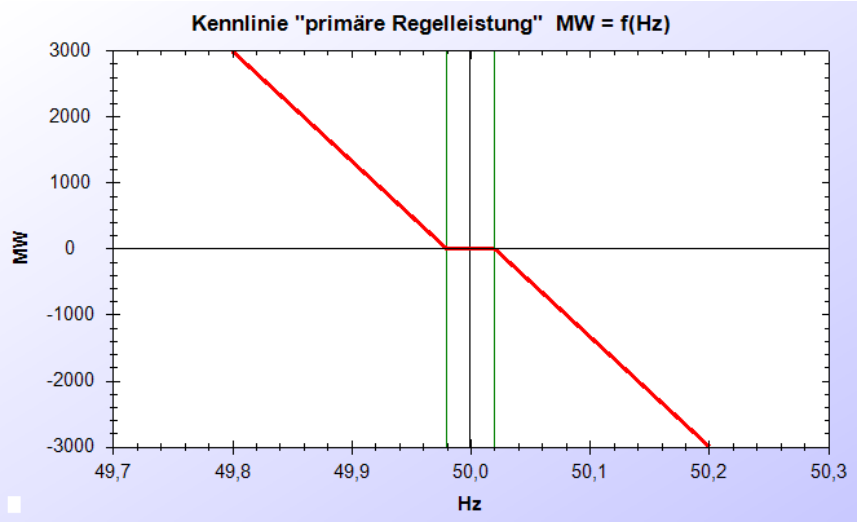
Die Lineare Funktion enthält eine Rampe. Der Offset-Bereich wird nicht zur Berechnung verwendet.

$\boldsymbol{\lambda} = \frac{\Delta P\ (MW)}{\Delta f(mHz)}$, Rampe im Offset-Bereich



Kennlinieninvertierung / Invert characteristic

Bei der Aktivierung wird die Leistungskennlinie invertiert, auf der Anzeige wird das Vorzeichen invertiert. Leistungen unter 50 Hz werden negativ dargestellt, über 50Hz positiv.



Diese Art der Darstellung zeigt die Erzeugersicht bzw. die Anforderung an die Erzeugung.

- Kennlinientyp **1** Auswahl Kennlinientyp für die Ausgabe der Netzlastdifferenz
- Netzlastdifferenz λ **15,333** Steigung Netzlastdiffernz-Kennlinie
- Offset (in mHz) ± **10** Offset Netzlastdifferenz-Kennlinie
- Negierung Kennlinie **false** Vorzeichennegierung Ausgabe Netzlastdifferenz

Einstellungen im Arduino-Programm:

```
//          mains load differenz / Netzlastdifferenz
byte  Typ = 1;
float Lamda = 15.33333333;
float Offset = 10;
bool Neg = false;
```

Tabelle Warnstufen Netzfrequenz

Reaktionen / Meldungen: Netzfrequenz (UCTE Continental Europe, 50Hz)

Hz	Messbereichsende Netzfrequenzanzeige	Stabilitätsgrenze überschritten	Warnmeldung Netzfrequenzanzeige Stufe 3	Warnmeldung Netzfrequenzanzeige Stufe 2	Warnmeldung Netzfrequenzanzeige Stufe 1	Beginn Einspeiseabwurf EEG	positive primäre Regelleistung (Netzsicht)	Nennfrequenz	negative primäre Regelleistung (Netzsicht)	Warnmeldung Netzfrequenzanzeige Stufe 1	Warnmeldung Stromnetze (ÜNBs / VNBs)	Warnmeldung Netzfrequenzanzeige Stufe 2	Lastabwurf nach Frequenzabwurfplan	Stabilitätsgrenze unterschritten	Warnmeldung Netzfrequenzanzeige Stufe 3	Messbereichsende Netzfrequenzanzeige	Alarmausgabe Netzfrequenzmessung
55,00	X																
52,00		X	X														f4
51,00				X													f3
50,20					X												f2
50,10						X											f1
50,02							X										
50,00								X									
49,98									X								
49,90										X							f1
49,80											X						f2
49,00												X	X				f3
48,00														X	X		f4
45,00																X	

Einstellung Frequenzmeldungen

Alarm	Einstellung *)	Alarm-Byte	Relaisausgabe
f1 >	50,100 Hz	Bit 0	Relais 1
f1 <	49,900 Hz		
f2 >	50,200 Hz	Bit 1	Relais 2
f2 <	49,800 Hz		
f3 >	51,000 Hz	Bit 2	Relais 3
f3 <	49,000 Hz		
f4 >	52,000 Hz	Bit 3	Relais 4
f4 <	48,000 Hz		
Res.	Vorbereitet für Alarm Netzfrequenzsprung	Bit 4	
Res.	Vorbereitet für Alarm Netzfrequenzpendelung	Bit 5	
Res.		Bit 6	
Sammelalarm	Sammelalarm (alle Meldungen)	Bit 7	Relais 5

```
// Alert mains frequency / Frequenzalarme          *)
word f1g = 50020;       // fmax Alert
word f1k = 49980;       // fmin Alter
word tf1 = 10;          // Alert RelapseTime
```

…

Alarmausgabe im Protokoll

Byte	Sammelalarm	Res.	Res.	Res.	Alarm f4	Alarm f3	Alarm f2	Alarm f1
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Wert	2^7 = 128	64	32	16	8	4	2	1
0	-		-		-	-	-	-
129	**X**							**X**
147	**X**			**X**			**X**	**X**
135	**X**					**X**	**X**	**X**
143	**X**				**X**	**X**	**X**	**X**
161	**X**		**X**					**X**

Tabelle: Berechnungsbeispiele für Alarmmeldungen

***) 60 Hz Trigger :**

Die Anpassungen der Trigger-Einstellungen für 60Hz-Netze erfolgt im Unterprogramm „**Settings_60Hz()“**.

Alarmausgabe J6

J6 vorbereitet als universelle Ausgabe-Schnittstelle (Low-Aktiv, z.B. Relaisausgabe)

Einsatz als Melde/Alarmierungssystem
Die Ausgabe sollte ausschließlich zu Meldungs- oder Alarmierungszwecken dienen. Netzschutzfunktionalitäten (z.B. Unterfrequenzabwurf) sollte nicht über die Netzfrequenzmessung ausgeführt werden.

Funktion der Schnittstelle J6:

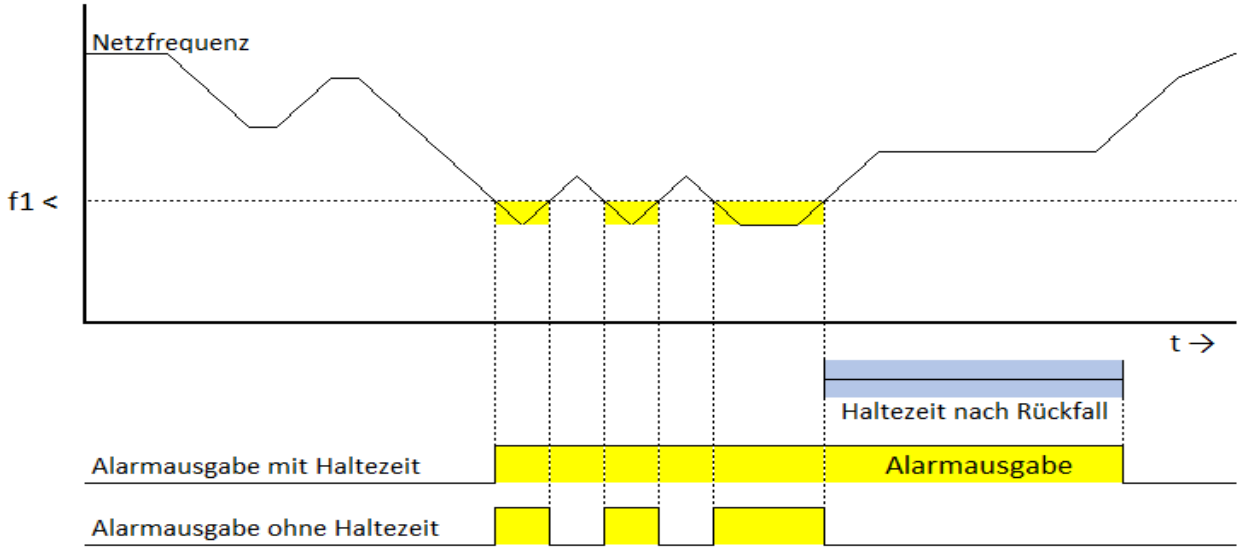
Alarm	J6	Pin		
f1	Rel1 Pin14 (Port PB.0)	**2**	**1**	GND
f2	Rel2 Pin13 (Port PD.7)	**4**	**3**	GND
f3	Rel3 Pin23 (Port PC.0)	**6**	**5**	GND
f4	Rel4 Pin15 (Port PB.1)	**8**	**7**	GND
SA	Rel5 Pin16 (Port PB.2)	**10**	**9**	GND

Ausgänge:

SP1 als Ausgang.

Im Einstellprogramm sind die Alarmausgänge aktiviert. Alarme werden als Binärinformation an den Ausgängen bereitgestellt. Angeschlossene Relais, LED's oder Optokoppler können die Alarme an Meldesysteme übergeben.

f <> Alarmausgabe mit Rückfallverhalten (J6 und Protokoll)



Skizze: Beispielhafte Alarmausgabe mit Rückfallverhalten

Die Haltezeit (das Rückfallverhalten) verhindert ein „Meldungsflattern" im Anregebereich z.B. bei angeschlossenen Relais.

J6 Ausgabeverhalten

Ausgabe High/Low-Aktiv

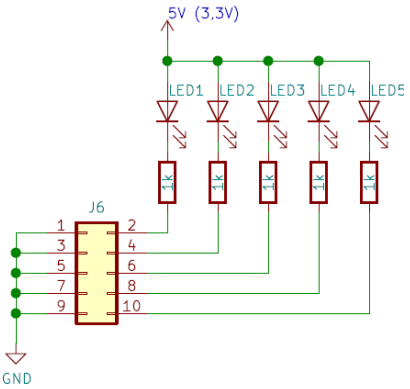
Die Voreinstellung *LowAktiv = **false*** negiert die Ausgabe, die Pins sind dann im Alarmzustand HighAktiv;

Blinkende Ausgabe

BlinkON = true erzeugt bei einer aktiven Meldung ein Blinksignal am Ausgangsport JP1 (für alle Ausgänge)

```
//Alert Output // Alarmausgaben
bool LowAktiv = true;   // Output Alarmausgabe LowAktiv
bool BlinkON = true;    // Blinkende Ausgabe z. B. LEDs
```

Ausgang mit LED-Beschaltung

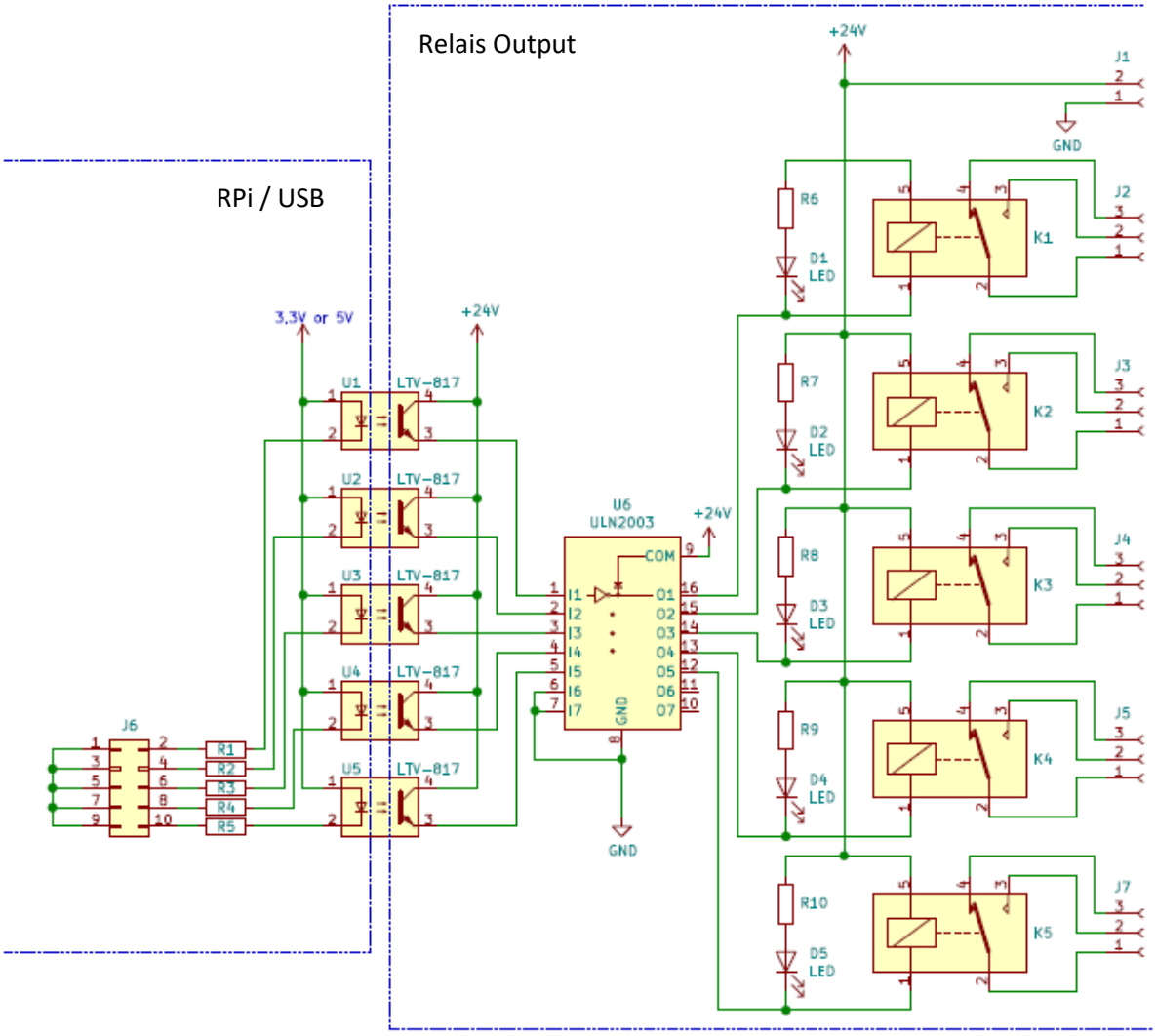


Beispiel für den Anschluss von LowCurrent-LED's an der Schnittstelle J6

Die Ausgabe der LED's in diesem Beispiel erfolgt LOW-Aktiv.

- ⇨ **„0" => 0V Meldung Aktiv,**
- ⇨ **„1" => 5V Meldung nicht Aktiv**

Beispiel: Ausgang mit Relais-Beschaltung



Die Ausgabe der Relais in diesem Beispiel erfolgt LOW-Aktiv (Voreinstellung).

- ⇨ **„0" => 0V Meldung Aktiv,**
- ⇨ **„1" => 5V Meldung nicht Aktiv**

Hinweis:

Ich empfehle, eine angeschlossene Relaisplatine über eine externe Stromversorgung zu versorgen.

LEDs

LED 1: RUN-LED

LED1 (RUN) Zustandswechsel (toggle) bei jedem erfolgreichen Filterdurchlauf.

(toggle= wechselt den binären Zustand)

Arduino-Ausgabe : 4 (=Pin6, PD4)

LED 2: RPi-Led

Die LED 2 kann durch ein Programm auf dem RaspberryPi angesteuert werden.

RaspberryPi Ausgabe : GPIO5 (=Pin12)

Verwendung mit einem USB-TTL-Konverter

Die Netzfrequenzplatine kann auch ohne einem RaspberryPi zur Netzfrequenzmessung verwendet werden. Die Datenausgabe und die Stromversorgung ist dann über die die USB-Schnittstelle (USB-TTL-Konverter) realisiert.

Für diese Betriebsart ist der **Anschluss J4** vorbereitet.

JP4: Dieser Port stellt das serielle Telegramm bereit (Details siehe „serielles Protokoll“)

Pin	Bezeichnung	Funktion	Hinweis
1	**GND**	Masse / GND	
2	**RxD**	Empfangsrichtung (zur USB), TTL-Pegel*	Vom µC (TxD)
3	**TxD**	Empfangsrichtung (von USB), TTL-Pegel*	Zum µC (RxD)
4	**+5V**	Versorgungsspannung 5V DC	

*) siehe TTL-Pegel

Auswahl der Protokolle

J5.1 und J5.2 vorbereitet für Protokollauswahl (serielle Schnittstelle)

Für die Protokollverarbeitung stehen in kürze einige Beispielprogramme bereit.

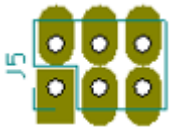
Protokoll	Frequenz	Netzlastdifferenz*	f <> Alarm*	Hardware	Netzfrequenz-Server-Tool*
1 : 7-Byte	Ja	Nein	Nein	RPi	Ja
2 : 17-Byte	Ja	Ja	Ja	RPi	Ja
3 : 24-Byte	Ja	Ja	Ja	RPi + NFA	Ja
4 : 29.Byte	Ja	Ja	Ja	RPi	Nein

*) Eine softwareseite Auswertung der Netzlastdifferenz, der Frequenzalarme und der Netzpendelerkennung ist mit dem gewählten Netzfrequenz-Protokoll auch auf dem Modbus/MQTT-Server möglich.

Basisprotokoll 7-Byte-Protokoll (5+2)

Protokollauswahl 1

Jumperstellung J5 :



Protokoll

Byte

1	2	3	4	5	6	7
5	0	0	0	0	\r	\n

=> 50,000 Hz

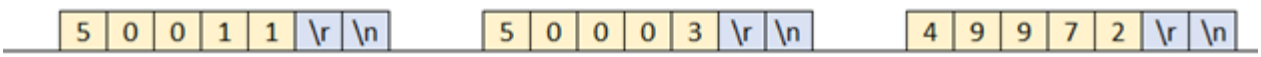


Bild: Aufbau und Telegrammverlauf 6-Byte-Protokoll

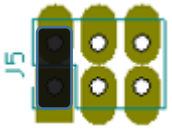
(Byte 6 und 7 sind ASCII- bzw. C0-Steuerzeichen: \n = LineFeed \r = Carriage Return)

Aus Ausgabe des Basisprotokolls auf der seriellen Schnittstelle erfolgt in **mHz**.

Protokoll 17-Byte-Protokoll

Protokollauswahl 2

Jumperstellung J5:



Protokoll

Byte

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
5	0	0	0	0	;	±	9	9	9	9	;	2	5	5	\r	\n

f (mHz) MW Alarm

Bild: Aufbau und Telegrammverlauf 17-Byte-Protokoll

Byte 7 :

f < 50000 mHz	f = 50000 mHz	f > 50000 mHz
Byte 7 = „-“	Byte 7 = „0“	Byte 7 = „+“

(Byte 16 und 17 sind ASCII- bzw. C0-Steuerzeichen: \n = LineFeed \r = Carriage Return)

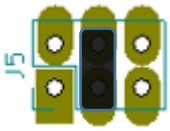
Das 17-Byte-Protokoll benötigt die Angabe des Netzlastkoeffizienten, Energierichtung (Kennlinieninvertierung), Ptot und der Grenzwerte der Alarmierungsfunktion.

Das Protokoll ermöglicht die einfache Weiterverarbeitung (z.B. mit einem Zeitstempel) zur Speicherung in eine CSV-Datei und späteren Auswertung z.B. mit Excel.

Protokoll 24-Byte-Protokoll

Protokollauswahl 3

Jumperstellung J5:



Protokoll

Byte

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
N	F	A	5	0	0	0	0	f	±	9	9	9	9	P	2	5	5	M	2	5	C	\r	\n
Header			Hz					f	± MW					P	Alarm			M	Parity		C	\r	\n
			Parity Prüfsummenberechnung																Partity				

Bild: Aufbau und Telegrammverlauf 24-Byte-Protokoll

Byte 10 :

f < 50000 mHz	f = 50000 mHz	f > 50000 mHz
Byte 10 = „-“	Byte 10 = „0“	Byte 10 = „+“

(Byte 23 und 24 sind ASCII- bzw. C0-Steuerzeichen: \n = LineFeed \r = Carriage Return)

Das 24-Byte-Protokoll benötigt die Angabe des Netzlastkoeffizienten, Energierichtung (Kennlinieninvertierung), Ptot und der Grenzwerte der Alarmierungsfunktion.

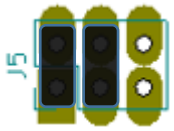
LED- 7-Seg Netzfrequenzanzeige V4

Das 24-Byte-Protokoll wird auch von der Netzfrequenzanzeige V4 Ausgabe unterstützt.

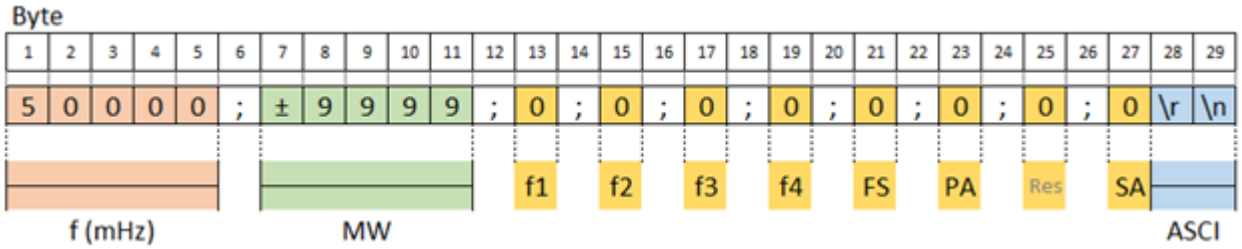
Protokoll 29-Byte-Protokoll

Protokollauswahl 4

Jumperstellung J5:



Protokoll



Byte 7 :

f < 50000 mHz	f = 50000 mHz	f > 50000 mHz
Byte 10 = „-“	Byte 10 = „0“	Byte 10 = „+“

f1 = Frequenzalarm 1 (Zusätzlich auch Ausgabe Bin/Rel. 1)

f2 = Frequenzalarm 2 (Zusätzlich auch Ausgabe Bin/Rel. 2)

f3 = Frequenzalarm 3 (Zusätzlich auch Ausgabe Bin/Rel. 3)

f4 = Frequenzalarm 4

FS = Frequenzsprung (vorbereitet)

PA = Pendelalarm (vorbereitet)

SA = Sammelalarm

(Byte 28 und 29 sind ASCII- bzw. C0-Steuerzeichen: \n = LineFeed \r = Carriage Return)

Die Ausgabe der Alarmmeldungen im Protokoll erfolgt „High-Aktiv“

=> „1“ = Alarm aktiv, „0“ = kein Alarm

Dieses Protokoll ist für eine Aufzeichnung der Messdaten z.B. durch ein Terminalprogramm vorgesehen. Die Auswertung ist als CSV-Datei möglich.

HINWEIS: Zur weiteren genauen Verarbeitung muss ein zusätzlicher Zeitstempel eingefügt werden.

Nachbau

DIL-Version V1.0

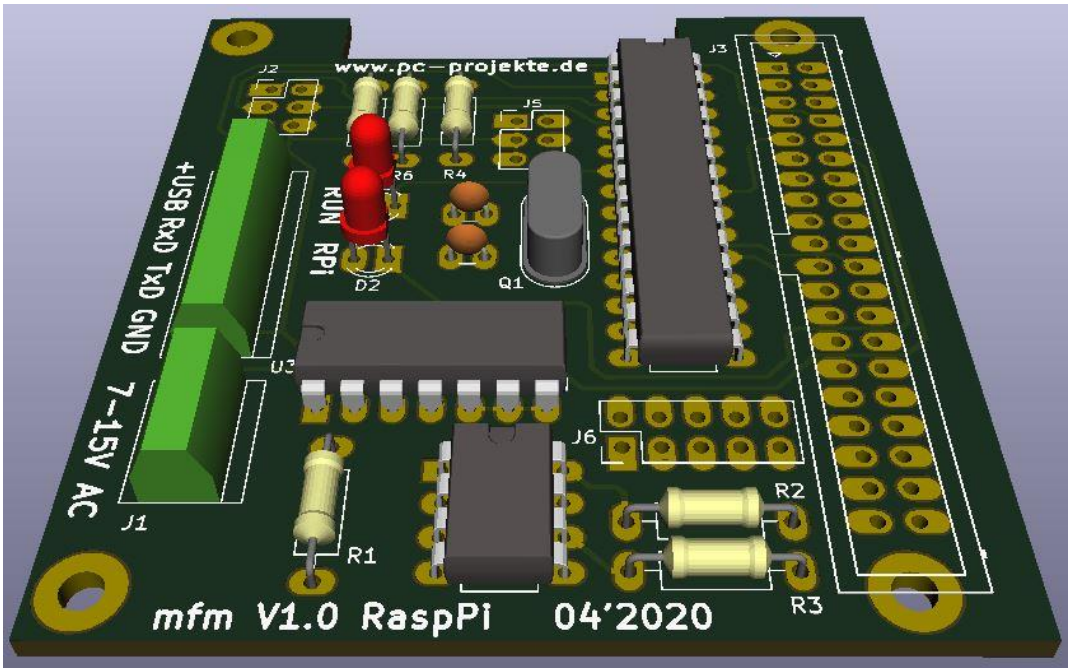
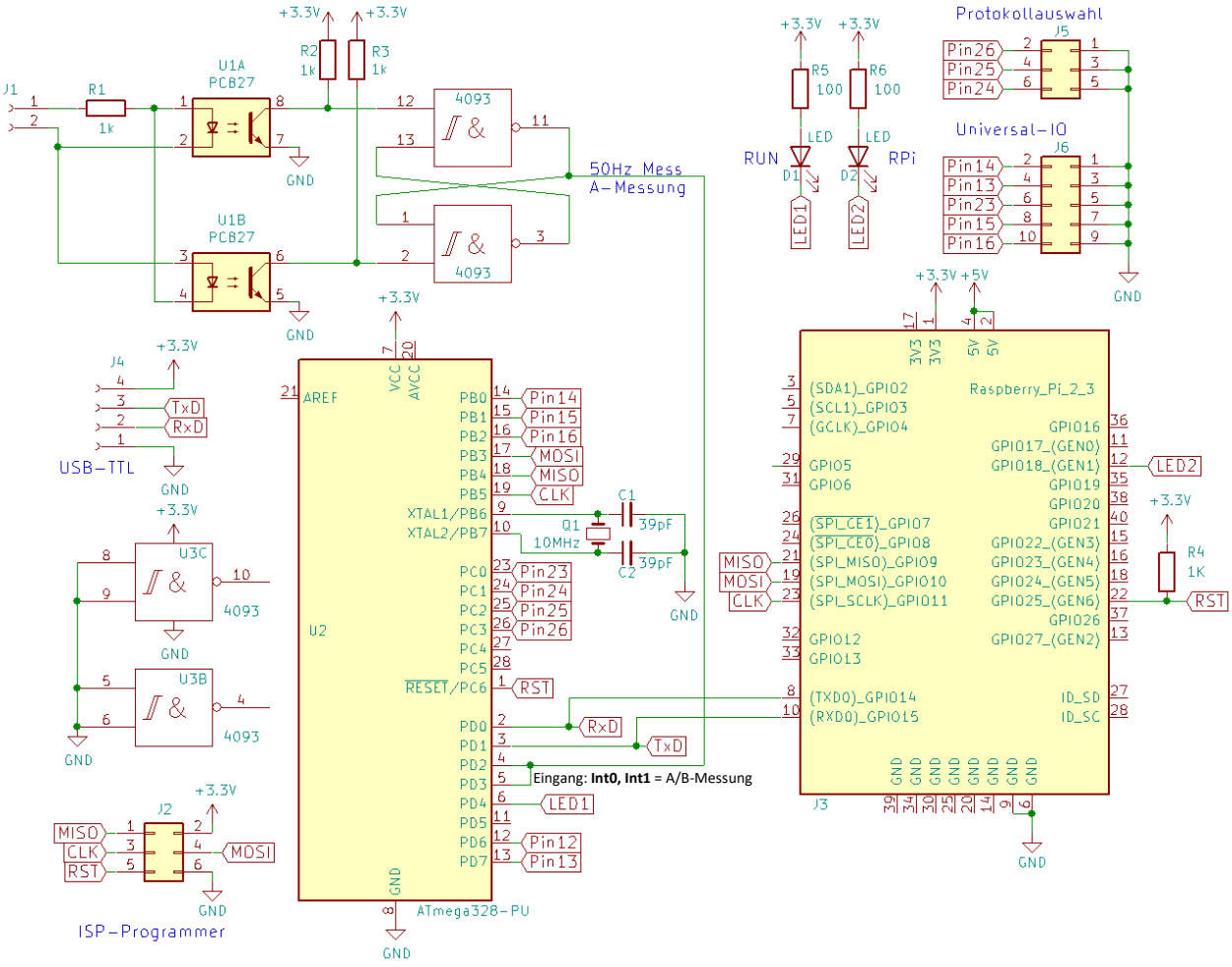


Bild: Platine Netzfrequenzmessung mfm V1.0 RaspPi



Aufbauskizze

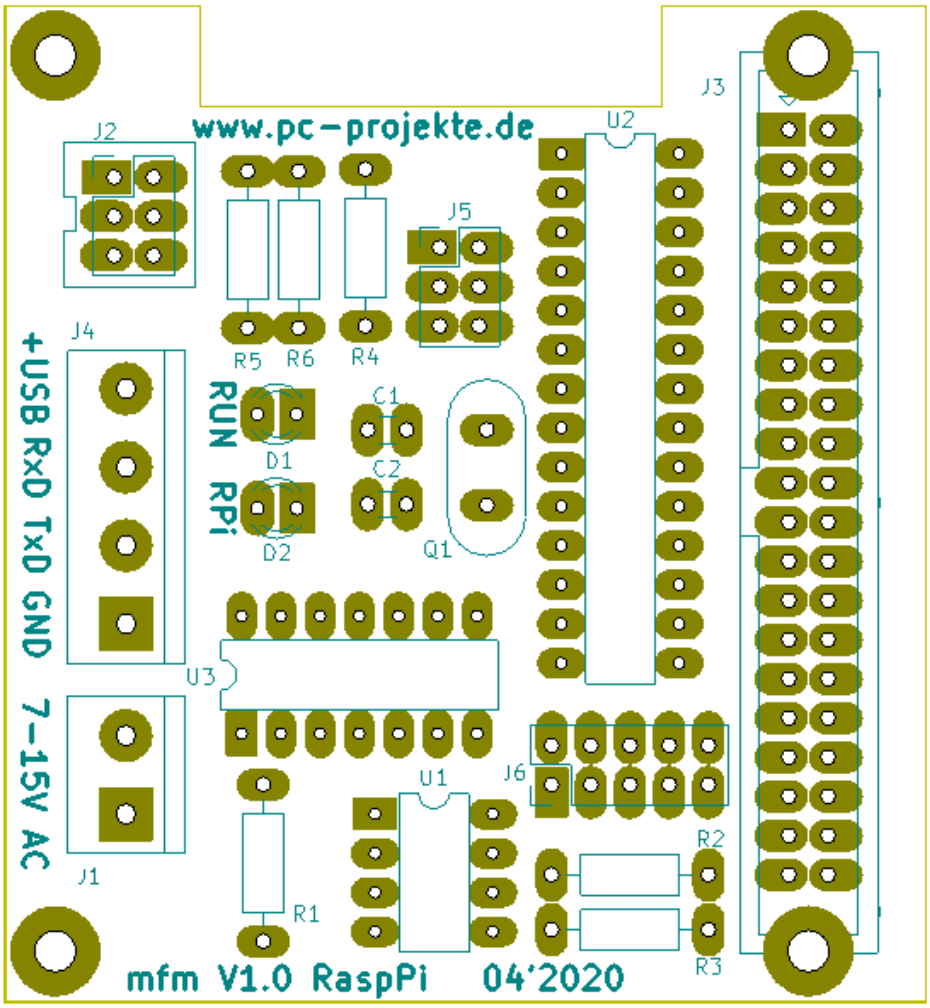


Bild: Aufbau Bauteile

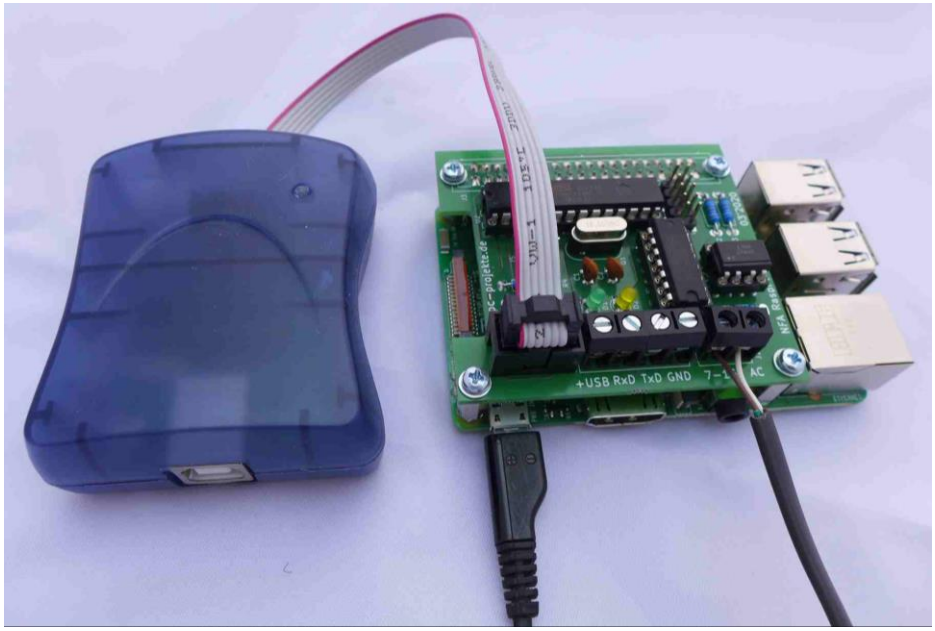
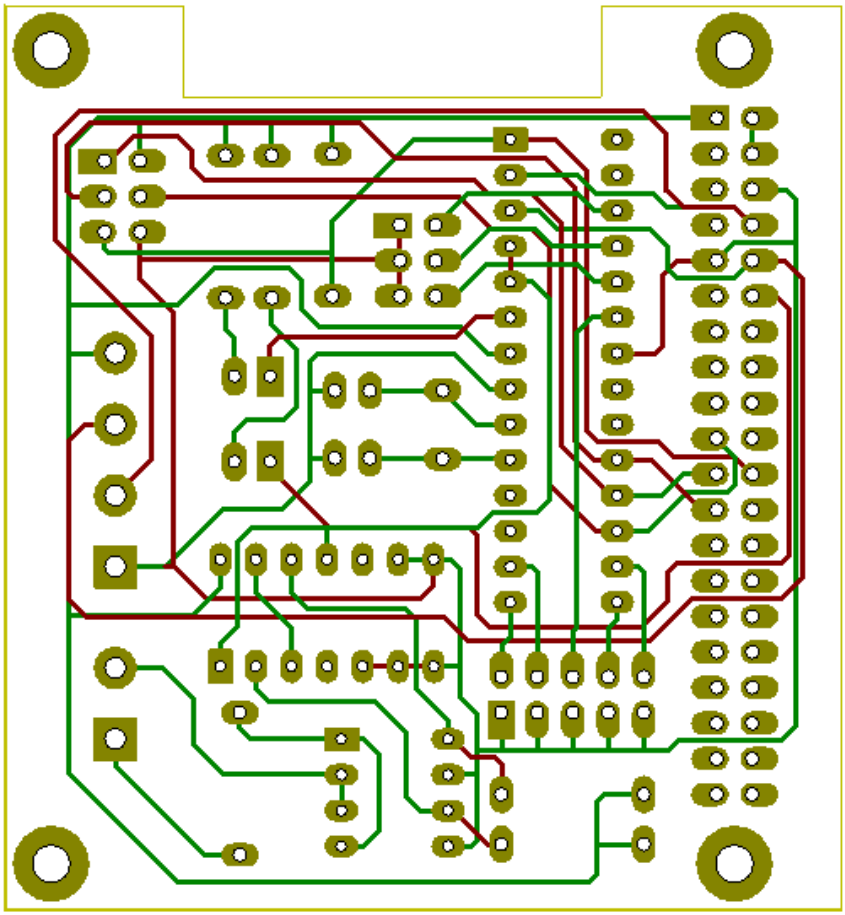


Foto: bestückte Platine auf dem RaspberryPi mit angeschlossenem ISP-Programmer

Platinen-Layout



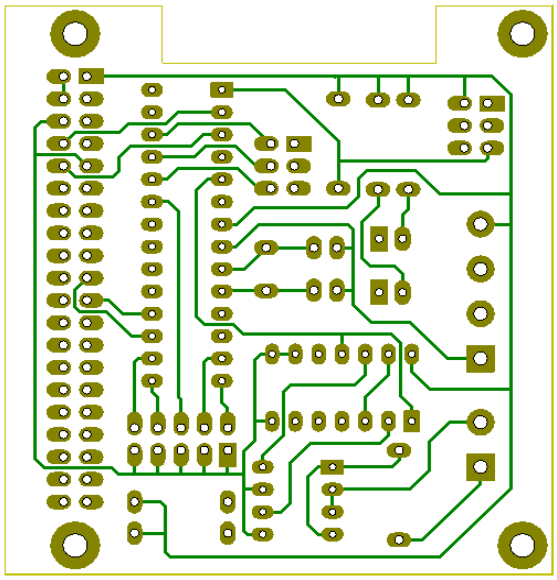


Bild: Platinenlayout Rückseite

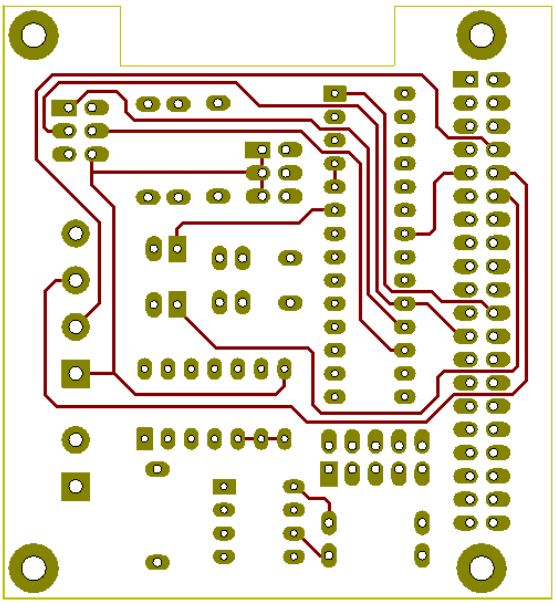


Bild: Platinenlayout Vorderseite

Bauteile:

Bauteile	Wert	Funktion	Info
C1,C2	22pF	Kondensator	siehe auch *Quarz und C1 / C2*
D1,D2	LED	LowCurrent-LED	
R1*	1k (1000 Ω)	Widerstand siehe Unten	Anpassung an Eingangsspannung
R2,R3,R4	1k (1000 Ω)	Widerstand	
R5,R6	R100 (100 Ω)	Widerstand	
U1	PC827	Optokoppler	Alternativ: 2 Stk. PC817
U2	ATMega328P	Atmel µController	(Arduino - Kompatibel)
U3	CMOS4093	SchmittTrigger (NAND-Gatter)	
Q1	10MHz	Quarz (HC49U)	**Q-Lastkapazität: 30pF**
J1	2-Pol Schraubklemme	Klemme	Lötbare Schraubklemme 5mm
J2	Wannenstecker 6-Pol	ISP-Port für ISP-Programmer	WSL 6G gerade (2X3-Pol)
J3 *	Buchsenleiste 2X40 Pol	RaspberrPi Buchsenleiste	RaspPi Connector Pin 1-40
(U1)	IC-Sockel 8-polig	IC-Sockel für U1	Empfohlenes Zubehör für U1
(U2)	IC-Sockel 28-polig	IC-Sockel für U2	Empfohlenes Zubehör für U2
(U3)	IC-Sockel 14-polig	IC-Sockel für U3	Empfohlenes Zubehör für U3
Trafo	7 – 15 V AC ~	Netzteil / Trafo	Messsignal-Trafo
J4	4-Pol Schraubklemme	Klemme (2X2 Schraubklemme)	Bei USB-TTL-Verwendung
J5	Stiftleiste 2X3-Pol	Jumper	Universelle Eingabe-Erweiterung
J5.1, J5.2	Jumper	Jumper (Kurzschlussbrücke)	Für die Auswahl Protokollausgabe
J6	Stiftleiste 2X5Pol	Relais-OUT	Universelle Ausgabe-Erweiterung

*) bei der Anwendung für den RaspberryPi

Außerdem

- RaspberryPi 3 oder 4 ***oder*** USB-TTL-Konverter
- Abstandshalter für die Netzfrequenzplatine: M2,5 X 12mm, Female-Female + Schrauben 2,5mm X 5
- Gehäuse

*) Empfehlung / Berechnung Eingangswiederstand R_1

Eingangsspannung (AC)	Empfohlene Größe: R_1
5V ~ - 7V ~	470 Ω
7V ~ - 15V ~	1k Ω
15V ~ - 25V ~	1,5k Ω

$$R_1 \approx \frac{(U\sim) - U_{Led}}{I_{Led}} = \frac{U_{Mess} - 2V}{10mA}$$

*Bei Verwendung mit USB (**ohne RPi**)*

- USB zu TTL serieller UART-Wandler mit 5V (oder 3,3V)

Bild: USB-TTL-Konverter

Bild: Anschlussbeispiel USB-TTL-Konverter (ohne RPi)

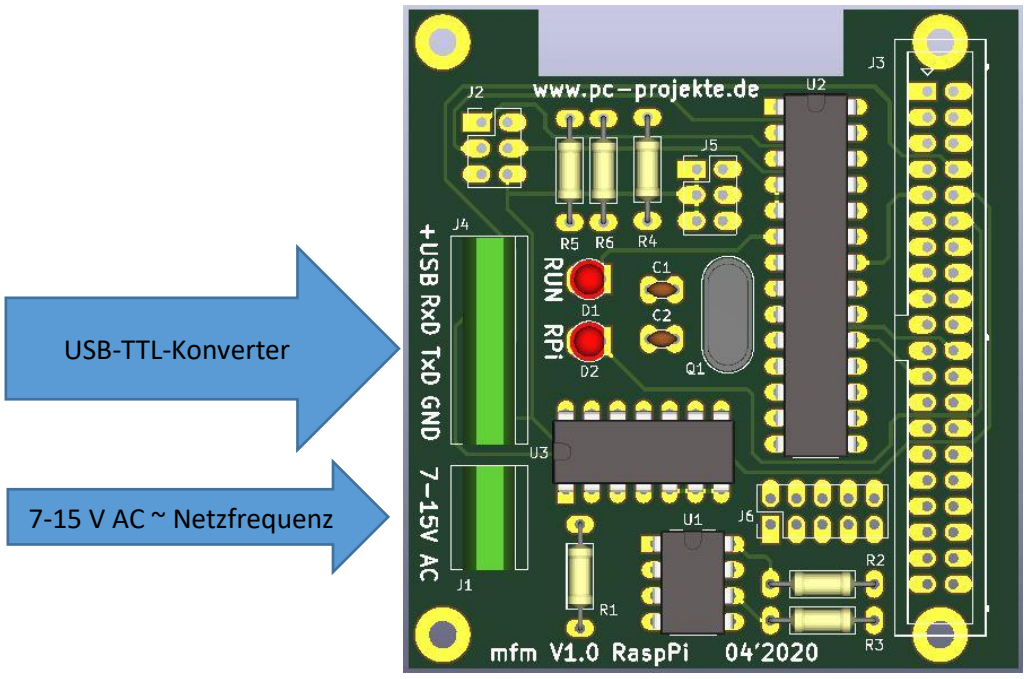


Bild: Anschlussschema USB-TTL-Konverter

Einbauhinweise:

U1 Optokoppler

Statt eines PC827 (U1) können auch zwei PC817 (U1a, U1b) verwendet werden.

Die Bestückung muss dann erfolgen wie in auf dem Bild:

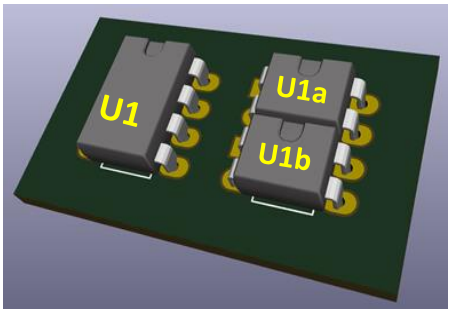
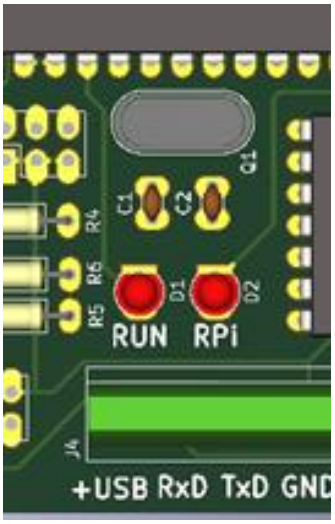


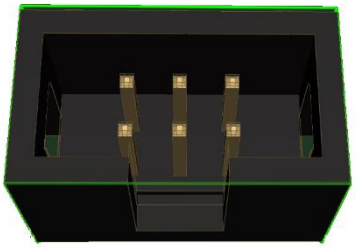
Bild: alternative Bestückung 2 Stk. PC817 statt 1 Stk. PC827

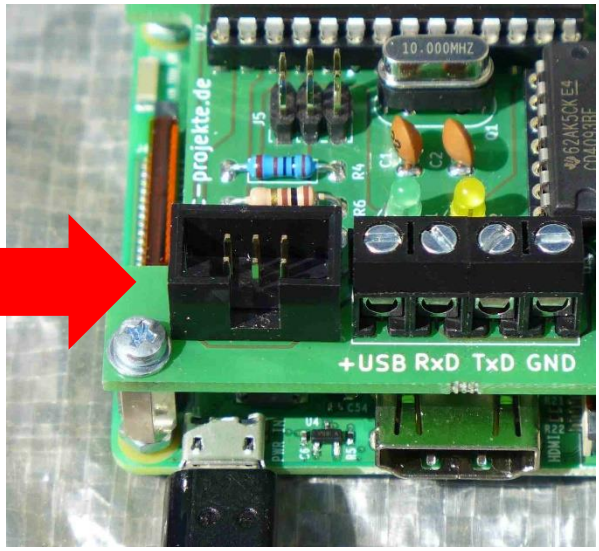
Einbaulage LED 1 und LED 2

Die Bestückung / Einbaulage der LEDs muss wie im Bild rechts angegeben erfolgen.



Einbaulage J2





Infos zum Quarz

Abstand des Quarz zur Platine

Beim Bestücken unbedingt Abstand zur Platine halten, um Kurzschlüsse zu vermeiden

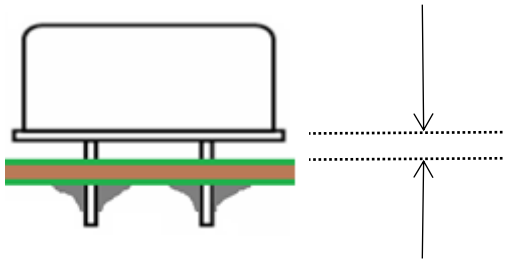


Bild: Quarzabstand zur Platine

Bauteilinformation: Quarz und C1 / C2

10,0000-HC49U-S, Standardquarz im Gehäuse HC49/U-S.

- Frequenz: 10,0000 MHz
- Cl: 32 pF ==> **C_1 / C_2 = 22pF** Lastkapazität im Datenblatt beachten!
- Rsmax: 50 Ohm
- Temperaturkoeffizient: ± 30 ppm
- Frequenztoleranz: ± 30 ppm

Gehäuseempfehlung

Die Netzfrequenz-Platine ist für den Einsatz in einem Hutschienengehäuse vorbereitet

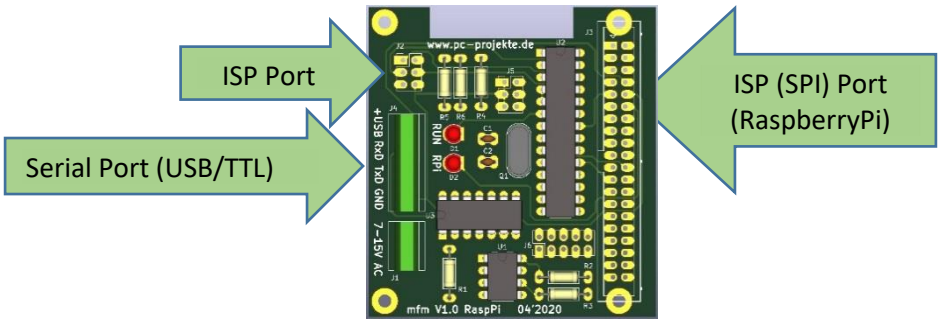
Bild: Hutschienengehäuse für RaspberryPi 4 und RaspberryPi 3

Software / Quellcode

Das Basisprogramm und die extended-Version stehen in der Arduino-IDE –Variante (C / C++) und als HEX-Code zum Download bereit:

Wichtiger Hinweis zur Arduino-Kompatiblität:

Die Netzfrequenzplatine V1.0 arbeitet mit 3,3V (RaspberryPi). Durch die 3,3V kann der ATMega328p nur ein maximaler Quarz von 10MHz verwenden. Deshalb müssen einige **Controller-Fuses** umgestellt werden. Eine direkte Arduino-Programmierung über die serielle Schnittstelle (USB) ist deshalb in dieser Hardware-Version nicht möglich. Eine Programmierung des ATMega328p ist über den vorbereiteten ISP-Port oder über den RaspberryPi (ISP-Schnittstelle) möglich.



Ein 100% kompatibles Arduino-*„Netzfrequenzboard“* (mit serieller Übertragung) ist in Vorbereitung.

µController-Fusebits

Die *„System Clock“* und *„Clock Otions“* - Fusebits des **ATMega328** müssen folgende Einstellungen haben:

LOW Fuses = 0xFF	HIGH Fuses = 0xDA	EXTENDED Fuses = 0xFF

Die korrekten Fusebits können mit *AVRDude* oder *Atmel AVR-Studio eingestellt und* übertragen werden. Ohne die korrekte Einstellung läuft das Netzfrequenz-Programm mit 3,3V Betreibspannung nicht.

Übertragung / Programmierung

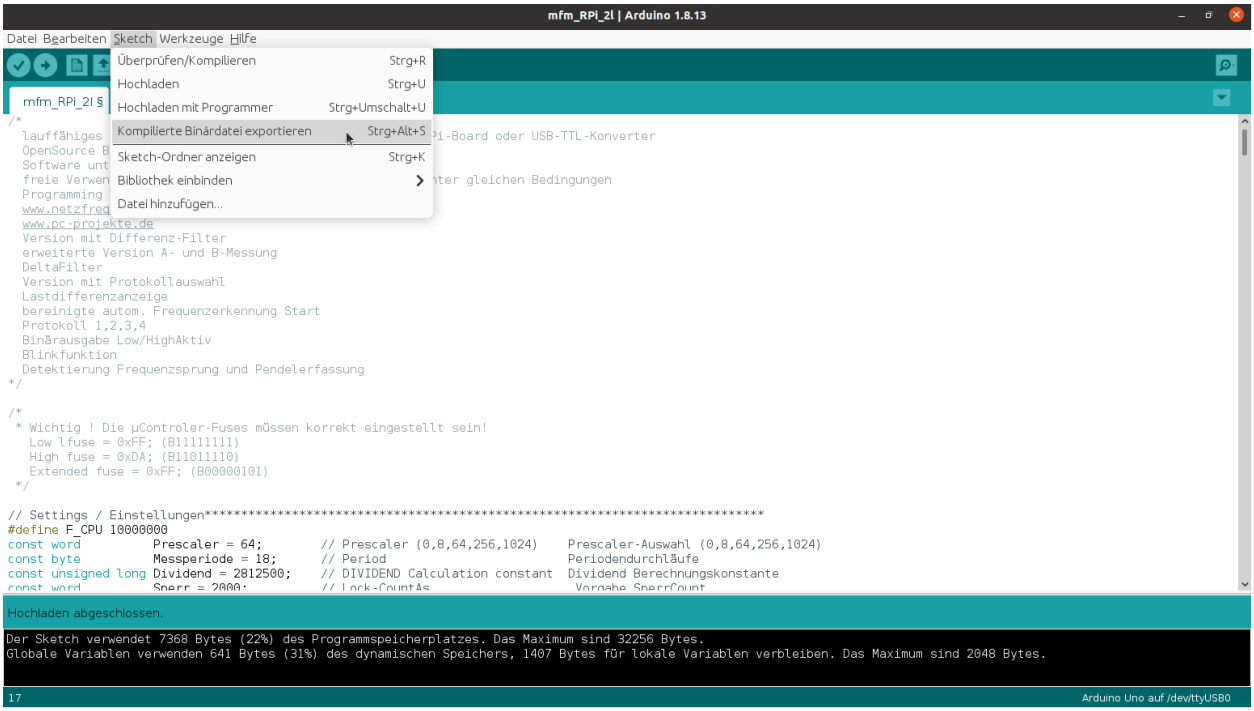
Die Übertragung des Quellcodes kann über mehrere Wege erfolgen. Ich beschreibe auf der nächsten Seite eine beispielhafte Möglichkeit. Weitere Programmiermöglichkeiten siehe Projektseite.

Programmierservice

Wer nicht selbst programmieren möchte, für den biete ich einen Programmierservice an.

Arduino Quellcode compilieren

Das aktuelle oder geänderte Arduino-Projekt (Quellcode) kann einfach als Hex-Datei kompiliert werden.



Im Arduino-Projektordner stehen nun (nach der fehlerfreien Comilierung) zwei kompilierte Hex-Files zur Verfügung.

Bitte verwenden Sie **nicht** das Hexfile mit dem vorbereiteten Arduino-Bootloader, sondern das File mit der Bezeichung „*projektname*.**ino.standard.hex**“.

Dieses Hexfile kann nun über die ISP-Schnittstelle auf den µController übertragen werden.

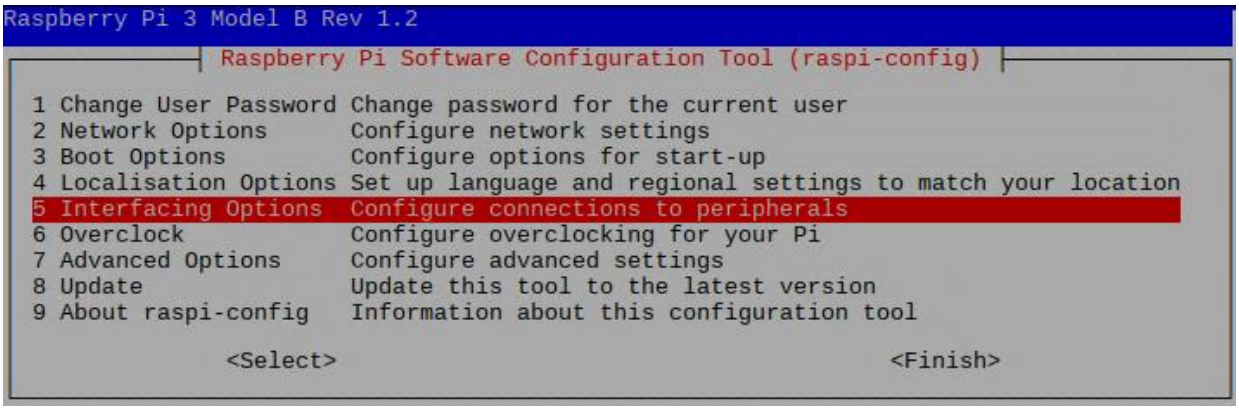
Zur Vereinfachung sollten sie das neue Arduino-Hexfile umbenennen z.B. „mfm_2.hex“

Die nächste Seite beschreibt die Vorbereitung der ISP(SPI)-Schnittstelle, Installation und Übertagung von AVRDude und dem RaspberryPi

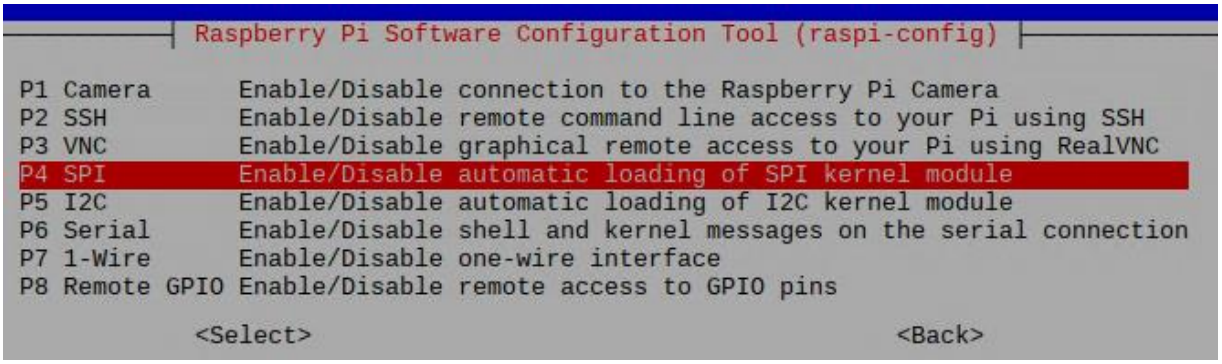
SPI - Vorbereitung des RaspberryPi

1. Freigabe SPI-Schnittstelle (RaspberryPi)
   Raspi-config öffnen

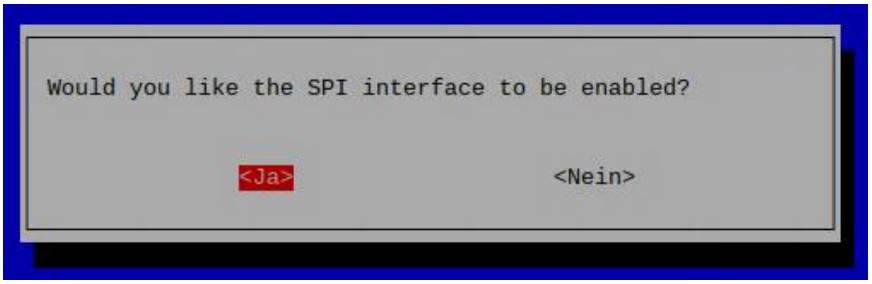
```
sudo raspi-config
```



2. Freigabe SPI-Schnittstelle



3. Auswahl bestätigen



4. „raspi-config“ schliessen

Der RaspberryPi ist nun für die SPI-Kommunikation vorbereitet.

Zur Übertragung der Software auf den ATMega328p muss noch das Tool „**avrdude**“ installiert werden.

Übertragung über den RaspberryPi mit AVRDude (Hex-File über Raspi ISP-Port)

1. **Installation AVRDude auf dem RaspberryPi**

```
sudo apt-get install avrdude
```

2. **Anpassung der Einstellungen AVRDude an benötigte Hardwarepins**

Öffne AVRDude Konfiguration:

```
sudo nano /etc/avrdude.conf
```

Suche Zeile:

```
#
# PROGRAMMER DEFINITION
#
```

Füge unterhalb folgende Zeilen ein:

```
#------------------------------------------------------------------
programmer
  id    = "spi_mfm";
  desc  = "Use RPi SPI for mains frequency monitor unit";
  type  = "linuxspi";
  reset = 25;
  baudrate = 100000;
;
```

Speichere die geänderten Einstellungen

3. **Test AVRDude / Auslesen der Fuses**

```
sudo avrdude -c spi_mfm -p m328p –P /dev/spidev0.0  -U lfuse:r:-:b
```

4. **Setzen der benötigten Fuses mit folgenden Kommandozeilen:**

```
sudo avrdude -c spi_mfm -p m328p –P /dev/spidev0.0  -U lfuse:w:0b11111111:m
```

```
sudo avrdude –c spi_mfm -p m328p –P /dev/spidev0.0  -U hfuse:w:0b11011001:m
```

```
sudo avrdude -c spi_mfm -p m328p –P /dev/spidev0.0  -U efuse:w:0b11111111:m
```

5. **Übertrage das aktuelle HEX-Programm**
(aktualisiert mit Arduino-IDE oder als Download von der Projektseite)

```
sudo avrdude –p m328p –c spi_mfm –P /dev/spidev0.0 –U flash:w:“mfm_2.hex“
```

Hexprogramm *„mfm_2.hex“* , oder eigener/neuer Quellcode

Getestet mit dem RaspberryPi 3 und einem ATMega328p

weitere Übertragungsmöglichkeiten finden Sie in Kürze auf der Projektseite…

Messung / Ausgabetest

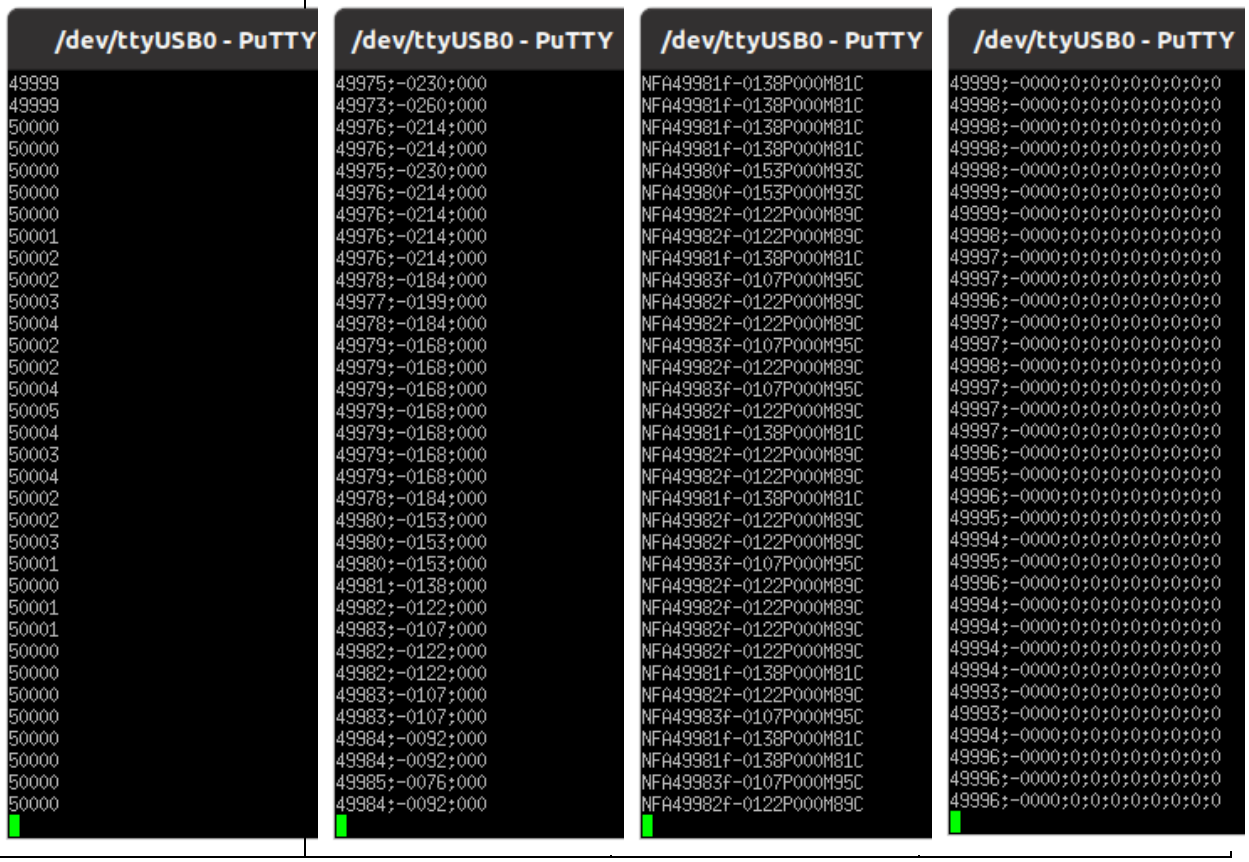
Überprüfung der Funktion mit einem Terminalprogramm:

- Öffnen sie ein Terminalprogramm z.B. PUTTY
- Überprüfen Sie die Einstellungen der Serial-Schnittstelle
- Serial Line: /dev/ttyS0 (oder /dev/ttyAMA0)

Baud: 19200 Bd **Databits: 8** **Stopbits: 1** **Parity: NONE** **Flow Control: NONE**

Das Terminalprogramm zeigt nach dem Öffnen den Datenfluss der Netzfrequenzmessung.

Protokollausgabe 1	Protokollausgabe 2	Protokollausgabe 3	Protokollausgabe 4



Die Aufzeichnung habe ich an meinem Linux-Laptop und einem USB-TTL-Adapter mit der universellen Netzfrequenzmessung RPi/USB gemacht.

Kommunikation der Netzfrequenzplatine

Das Konsolenprogramm mfmServerTool ermöglicht die Aufzeichnung und die Bereitstellung der Netzfrequenzmessdaten für leittechnische Anbindung z.B. OPC-Server.
Das Konsolenprogramm ist lauffähig auf Windows und Linux (RaspberryPi).

Die Daten können auf dem Web-Server des ioBrokers, dem OpenHAB (Open Home Automation Bus) oder dem OpenMUC (Web-Interface zur Systemkonfiguration und Visualisierung) problemlos visualisiert werden.

Das ServerTool-Programm steht in Kürze zur Verfügung.

Details zur Projekterweiterung finden Sie auf der Webseite www.netzfrequenzanzeige.de.

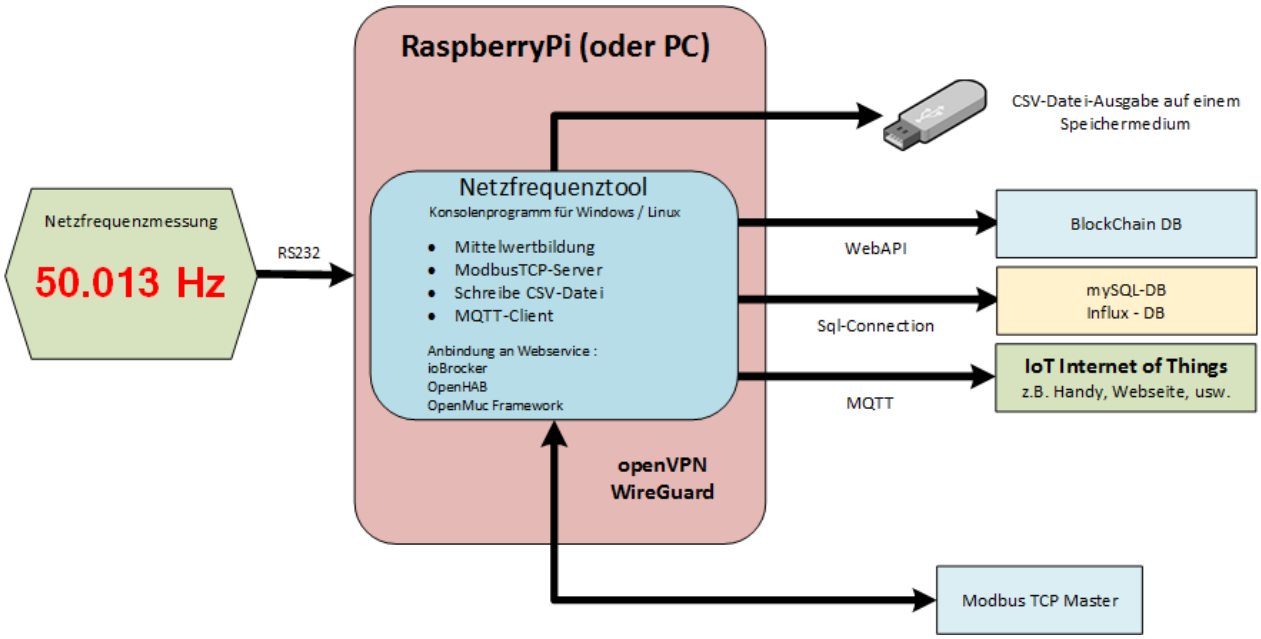
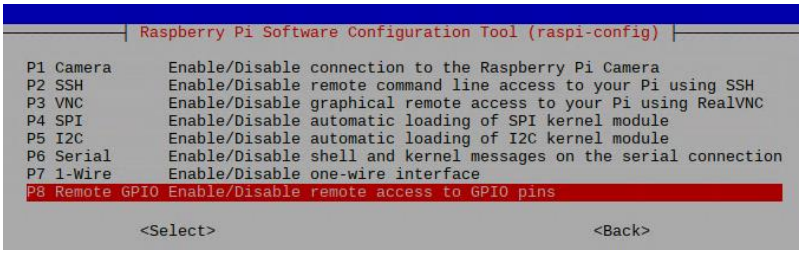
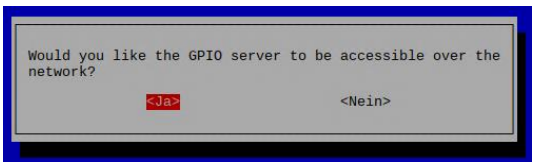


Bild: Schema Netzfrequenzmessung und Anbindung an eine Leittechnik

Sonstiges : Freigabe GPIOs





Infos / Bestellmöglichkeit

- weitere Informationen unter www.netzfrequenzanzeige.de
- Anfrage privater Verkauf Platinen und Wechselspannungsnetzteile (Restposten) pc_projekte.de@arcor.de
- Privater Programmierservice für µController
- Link für aktuelle Bauteileliste (bei Reichelt) https://www.reichelt.de/my/1689769

Hinweis / Gewährleistungsausschluss

Projekthinweis: Das gesamte Projekt ist ein privates, nicht kommerzielles Projekt.

Der Autor übernimmt keinerlei Gewähr für Funktionalität der vorgestellten Hard- und Software. Haftungsansprüche gegen den Autor, welche sich auf Schäden materieller oder ideeller Art beziehen, sind grundsätzlich ausgeschlossen. Die Inhalte der Homepage, der Projektbeschreibung oder der Software werden ohne Anspruch auf Richtigkeit veröffentlicht. Alle Angaben sind ohne Gewähr. Die Verwendung der Texte, Grafiken, Bilder und Programme sind im Rahmen der zum Projekt gehörenden OpenSource-Lizenzen erlaubt. Der Autor behält es sich ausdrücklich vor, Teile der Seiten ohne gesonderte Ankündigung zu verändern, zu ergänzen oder zu löschen.

- Verwendung der Soft- und Hardware auf eigene Verantwortung
- Jegliche Produkthaftung durch die Verwendung der Soft- und Hardware wird ausgeschlossen

Haftungsausschluss nach §1 Abs.2 Z.3 ProdHaftG , (BGBl. I S. 1474 m.W.v. 08.09.2015 der Bundesrepublik Deutschland)

Information zur Spannungsversorgung

Die Spannungsversorgung muss der Beschreibung entsprechen, da es sonst zu Zerstörung von Bauteilen kommen kann.

Jens Müller, 02‘2021 www.pc-projekte.de