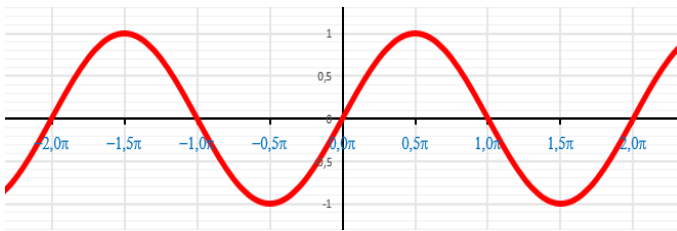
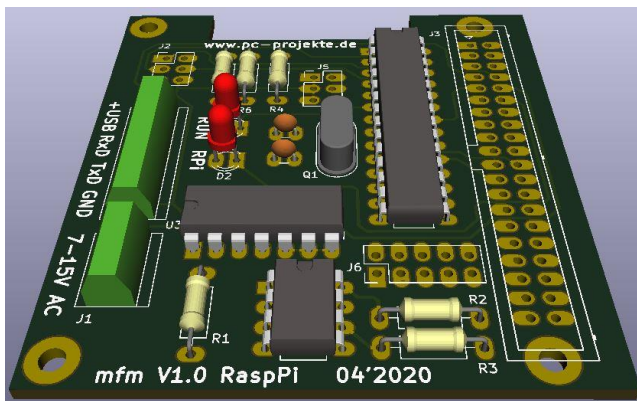


Netzfrequenzmessung für den RaspberryPi (und USB-TTL) Basisdokumentation



Der Blick in unser Stromnetz



Projektbeschreibung MAINS FREQUENCY MONITORING V1.0

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

Lizenz	2
Dokumentation	2
<i>Creative-Commons-Lizenz</i>	<i>2</i>
OpenHardware (OSHWÄ).....	2
Software / Quellcode	2
Hintergrund	3
Unser Stromnetz	3
Netzfrequenz als Qualitätsindikator	3
Einsatz der Netzfrequenzplatine.....	5
Software	5
Messspannung.....	5
Frequenzmessung.....	6
Messbereichserkennung bei Programmstart	6
Einstellungen für das Prescale-Messverfahren.....	7
Messung der Netzfrequenz	9
Berechnung	9
Detaillierte Beschreibung der Messung	10
Eingangsbeschaltung Schmitt-Trigger	10
Schematischer Programmablauf Messauswertung	11
Differenz-Komperator-Filter	14
Kalibrierung	15
Verwendung mit einem USB-TTL-Konverter.....	16
Nachbau.....	17
Schaltbild Netzfrequenzplatine	17
Aufbauskizze	18
Bauteile:	20
Einbauhinweise:	22
Software / Quellcode.....	24
µController-Fusebits	24
Übertragung / Programmierung	24
Programmierservice	24
Übertragung über den RaspberryPi mit AVRDUde (Hex-File über Raspi ISP-Port)	25
Messung / Ausgabetest	26
Kommunikation der Netzfrequenzplatine.....	27
Infos / Bestellmöglichkeit	28
Hinweis / Gewährleistungsausschluss	28

Lizenz

Dokumentation

Creative-Commons-Lizenz

Diese Projektdokumentation steht unter der Creative-Commons-Lizenz Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-sa/4.0/>



Dieses Dokument ist ein Basis-Dokument und gehört zur die Veröffentlichung OSHWA Openhardware

Eine Projekt-Extended-Version mit aktuellen Features und Erweiterungen steht in Kürze zum Download bereit.

OpenHardware (OSHW)

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 $16 \frac{2}{3}$ 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.

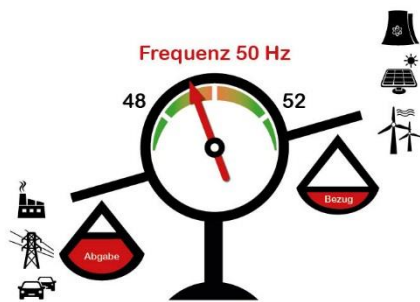
In speziellen Bereichen, z. B. im Bordnetz von Flugzeugen, sind höhere Netzfrequenzen üblich, z. B. 400 Hz, da sich dafür kleinere und leichtere Transformatoren bauen lassen und die Leitungslängen kurz sind.

Netzfrequenz als Qualitätsindikator

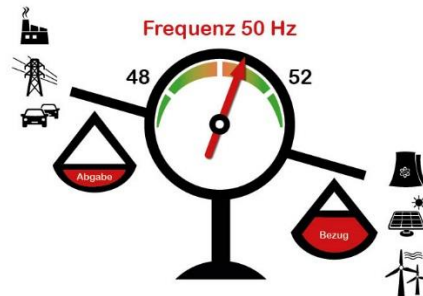
Die Netzfrequenz und deren Abweichung vom Nennwert ist ein direkter Qualitätsindikator über die Relation der über Erzeuger wie Kraftwerke angebotenen elektrischen Momentanleistung und der Abnahme der elektrischen Momentanleistung durch Verbraucher. Elektrische Energie kann in Verbundnetzen kaum gespeichert, 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, welche auch gleich dem sogenannten Referenzausfall aus dem Continental Europe Operation Handbook entspricht und ca. dem ungeplanten Ausfall von zwei größeren Kraftwerksblöcken entspricht. 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**.

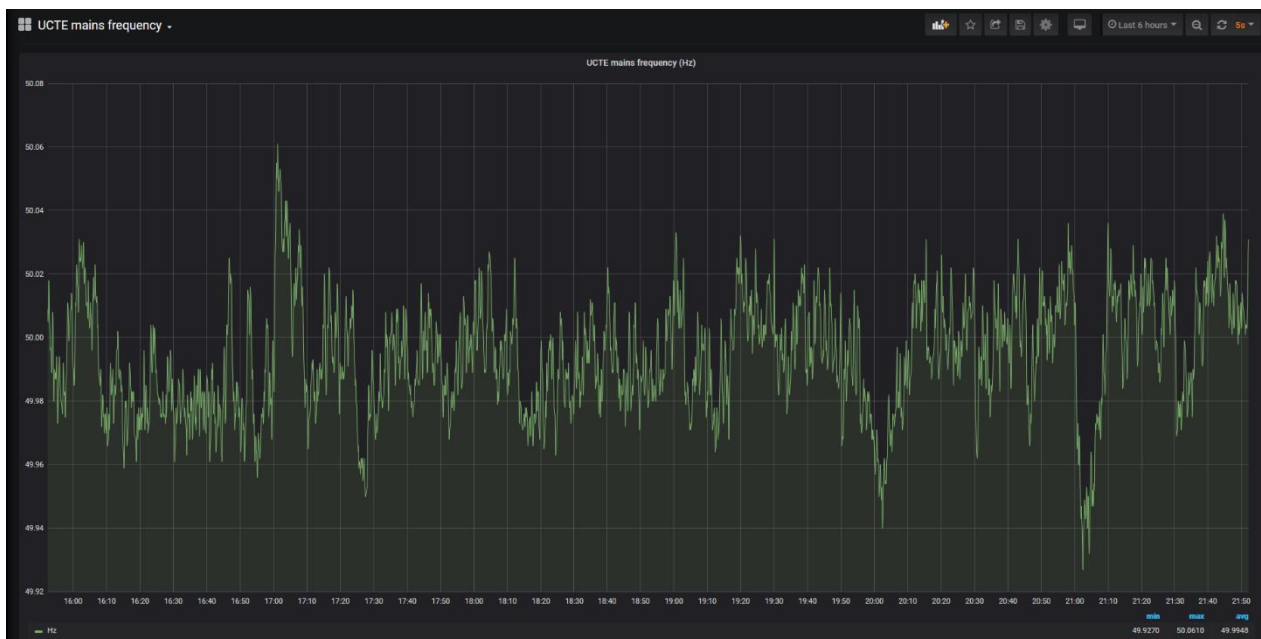


Bild: Netzfrequenzmessung (Beispiel grafische Auswertung mit Grafana®)

Es handelt sich um ein privates, nicht kommerzielles Projekt. Die Elektronik, μ Controller- 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 in Vorbereitung.

Die Ausgabe erfolgt über die serielle Schnittstelle. Die Platine ist für den RaspberryPi konzipiert, kann aber alternativ auch mit einem 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), 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 20 zu finden (Bestellmöglichkeit der Platine und der Bauteile siehe Seite 28). Eine teilbestückte Platine mit SMD ist in Vorbereitung.

Software

(µController ATMEGA328-16PU DIP-28 8-Bit 10 MHz (Arduino - Kompatibel))

Die aktuelle Software kann auf der Gitlab-Seite oder unter www.pc-projekte.de bzw. <http://www.netzfrequenzanzeige.de> heruntergeladen werden.

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 Wechsellspannungssignals. Das Signal wird über zwei Optokoppler und einem RS-FlipFlop (4093-Nand-Gatter als RS-FlipFlop) dem μ Controller zugeführt und mit dem 10MHz-Quarz-Takt und des Timer1 (als Counter) des μ Controllers 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.

Netzurückwirkungen

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

Sehr starke elektrische Impulse in der Umgebungen, ungewöhnlich hohe Oberwellenanteile oder Rundsteuersignale in der Messspannung (Netzurückwirkung) 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-Netze

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 μ Controller anzupassen bzw. zu optimieren.

Dazu können im μ Controller-Programm einige Parameter eingestellt oder geändert werden.

Einstellungen im Arduino-Quellcode:

Einstellung	Voreinstellung (Empfehlung)	Funktion
Prescaler	64	Interner Teiler des μ C-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 Frequenzbereicheinstellung (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)	Approximate measuring time at 50Hz
USB-TTL A dapter (5V) Raspberry Pi (3,3V)	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
	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 (Basis Quellcode):

```
// Settings / Einstellungen
// *****
#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 – 20 im Arduino-Quellcode (Basiscode) zu finden.

Messung der Netzfrequenz

Grundprinzip ist der Timer1 des µControllers, der ohne weitere Verarbeitung mit dem Start des µControllers den festen Takt 10MHz DIV **64** eingestellt ist. Der Timer1 wird mit dieser Einstellung der Referenztakt, aus dem die Netzfrequenz nun abgeleitet wird. Die zeitliche Gesamtlänge des durchlaufenden Timer1 (0... 65535) beträgt 423ms. Die dadurch optimierte Periodenanzahl der Netzfrequenzmessung (für 45 Hz) beträgt **18**.

Berechnung

Der Timer 1 läuft im eingestellten Prescaleverfahren mit dem (geteilten) Takt des µControllers als Counter.

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

Der Counterwert des Timer1 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 sogar lauffähig auf einem ATTiny).

Berechnung:

$$\frac{50\text{Hz}}{56250 \text{ Counts}} = \frac{f_{\text{Mess}}}{\text{Counts}} \longrightarrow \frac{2812500 \cdot 1000}{\text{Counts}} = f_{\text{Mess}}$$

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

Der vorläufige Frequenzwert (fMess) wird nun über einige Filterstufen geführt.

Die Ausgabe (Frequenz) erfolgt als serielles Protokoll auf dem TxD-Port des µControllers

UBRR

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 20).

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

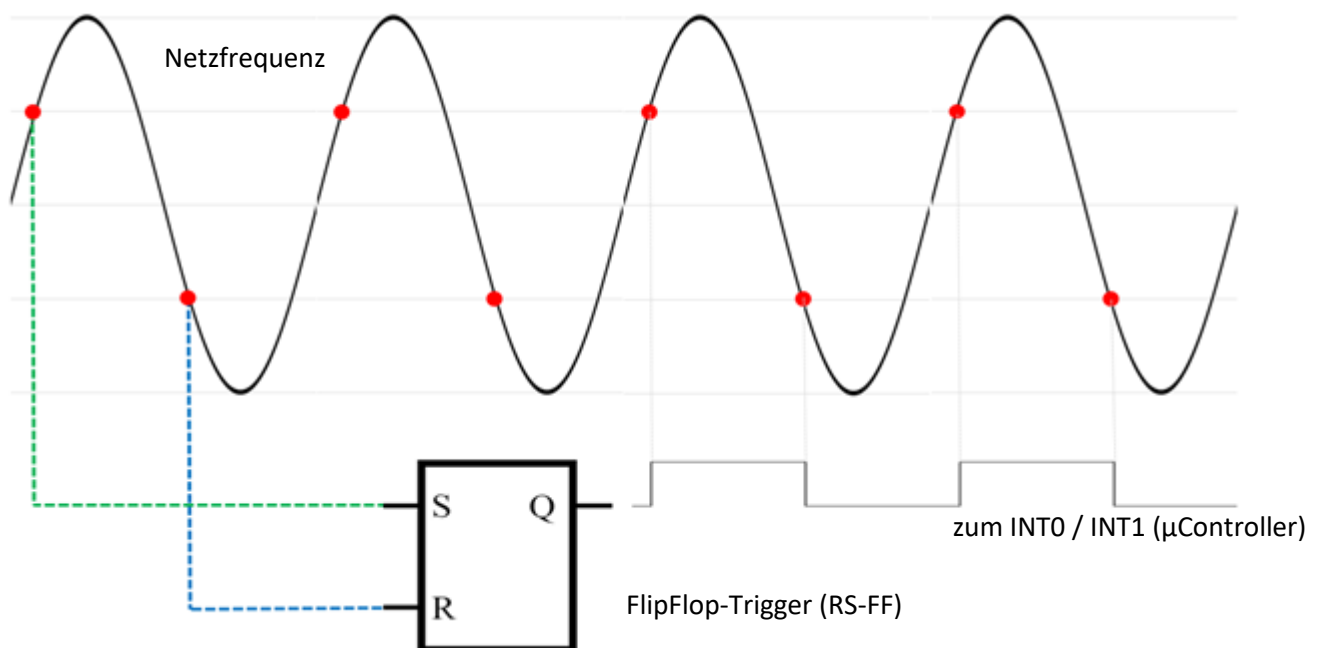


Bild: Schematische Darstellung der Flankenauswertung (Schmitt-Trigger)

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

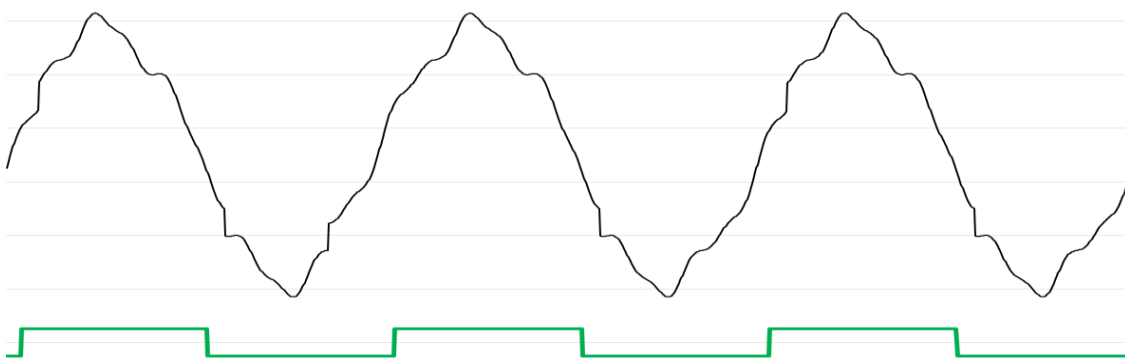
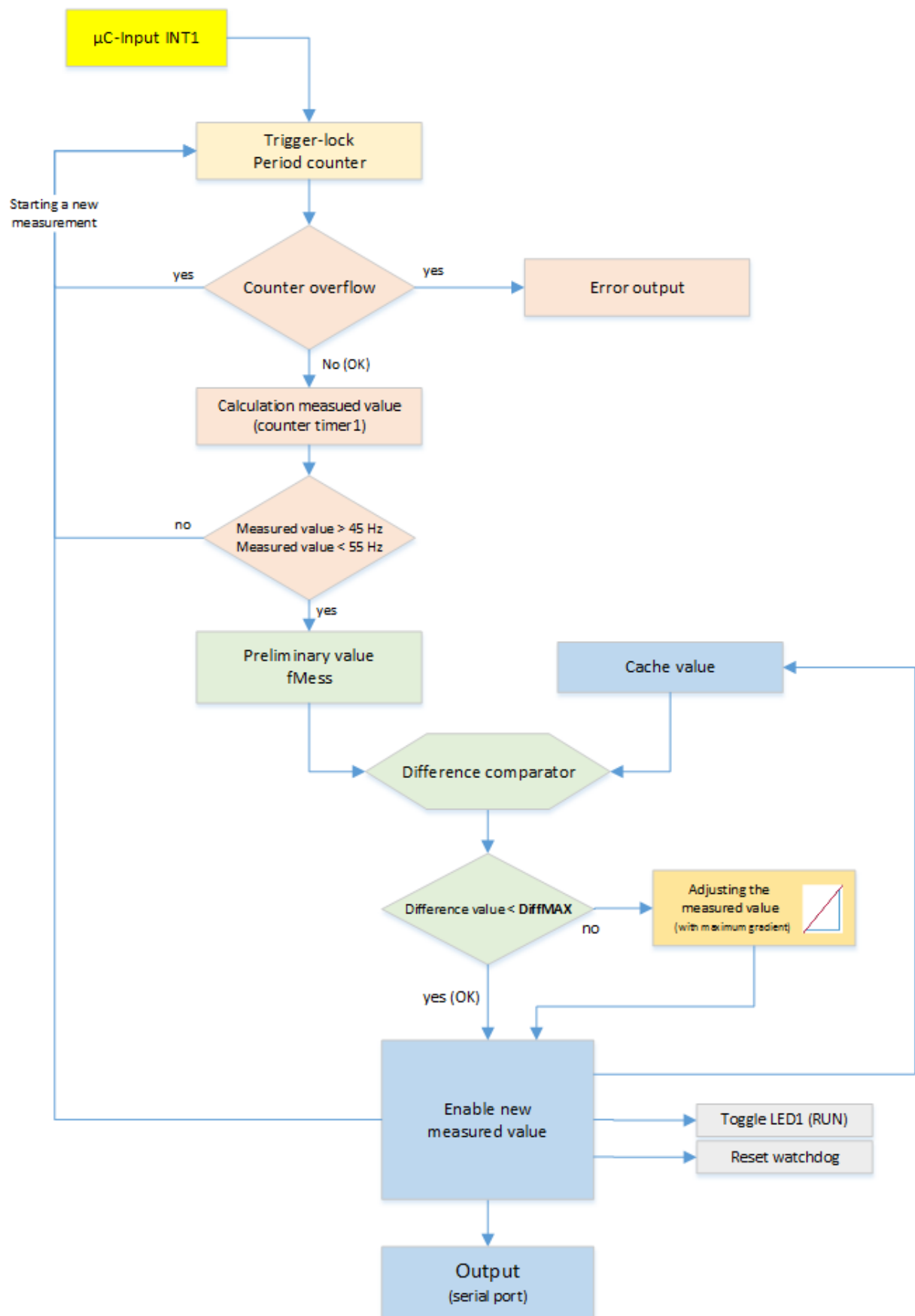


Bild: Schematische Darstellung eines verrauschten Eingangssignal und mit Ausgangssignal (hinter dem Schmitt-Trigger)

Schematischer Programmablauf Messauswertung



Flankenerkennung und Triggersignal-Sperre:

Erkennt der μ Controller 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.

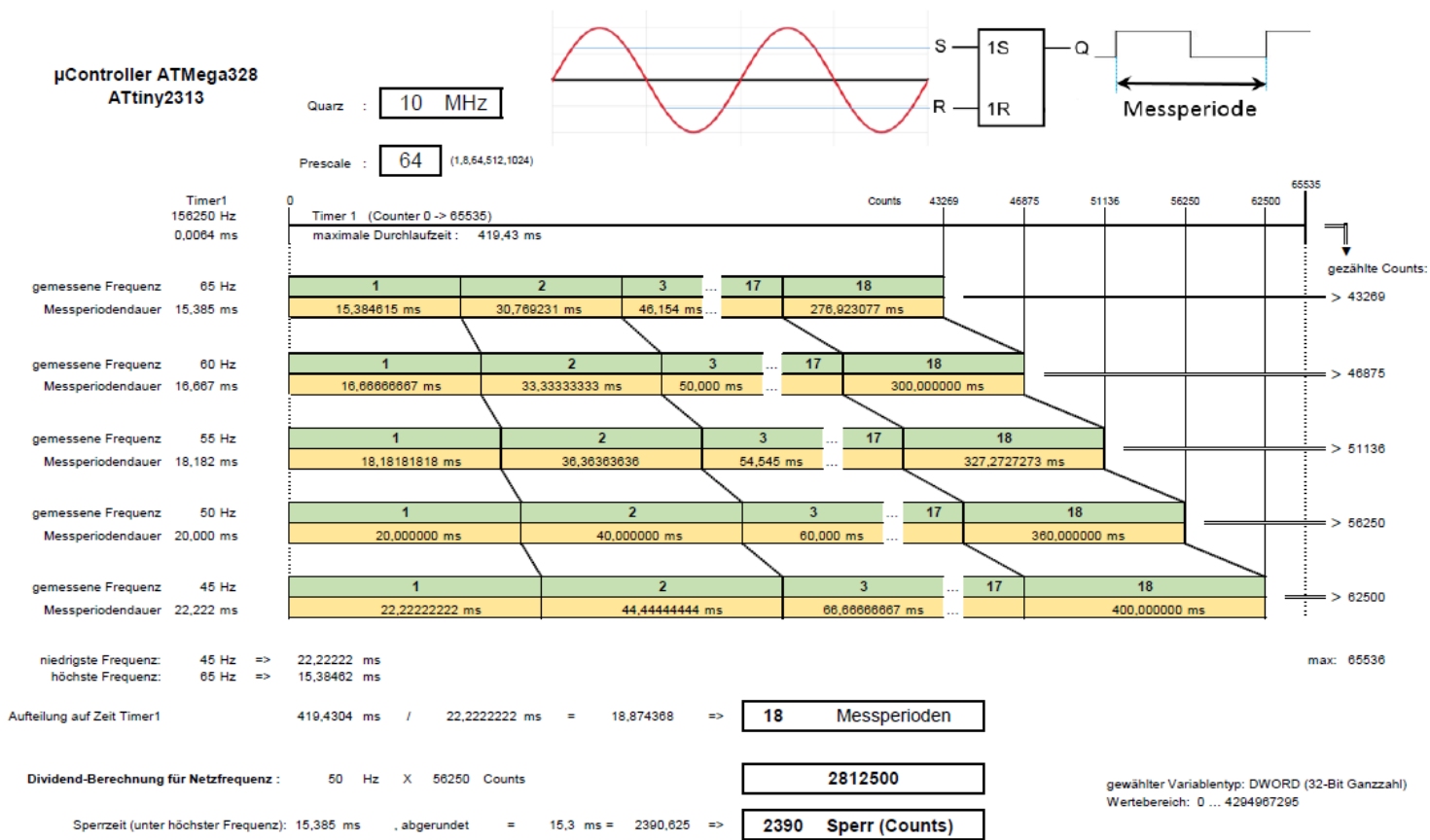


Bild: Schematische Darstellung der Trigger-Sperrzeit

Sperrzeit / Lock-Counts

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

Auswertung der Messcounts



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

Funktionsweise des Differenz-Komperator-Filters:

Der Differenz-Komperator-Filter verwendet prinzipiell die Vorgabe der maximalen Netzgradienten. Eine Netzfrequenzänderung über der maximalen Netzgradienten wird nur bedingt berücksichtigt. Sollten aber zwei aufeinanderfolgende Messungen die Gültigkeitsprüfung passieren, wird auch eine sehr schnelle Änderung sofort übernommen.

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 neuen Wert (Preliminary Value) nachgeführt.

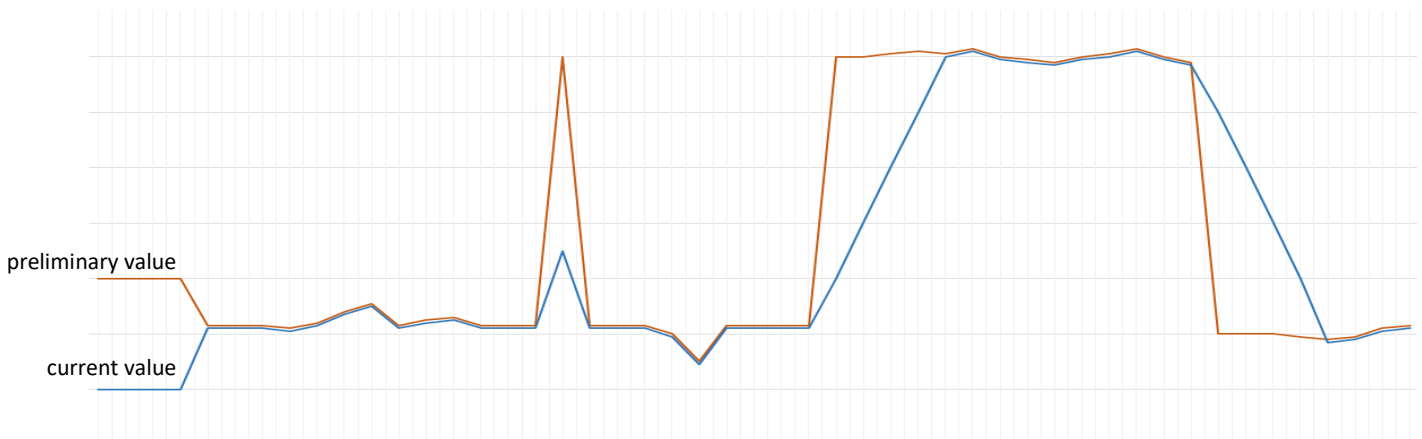


Bild: schematische Darstellung Verlauf Messdaten Differenz-Komperator-Filter

Die Netzgradienten hängen von vielen Netz-Schaltungszuständen, Einspeisungen, Netzlastsituationen und des Messortes ab. Deshalb ist der nachfolgende DiffMAX-Wert eine Einstellungsempfehlung für das UCTE-Netz.

Die Voreinstellung des DiffMAX-Filters beträgt **60mHz/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 einem Neustart werden die geänderten Einstellungen verwendet.

Frequenz + Calibration → Ausgabe zur seriellen Schnittstelle

Seriellles Protokoll

Einstellung der seriellen Schnittstelle

Baudrate: **19200 Baud** Parity: **None** Stopbit: **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.

Auswahl der Protokolle

In der Basisversion ist diese Möglichkeit noch nicht realisiert. In der extendend Version können über die Jumperstellung der J5-Schnittstelle die Ausgabeprotokolle ausgewählt werden.

Basisprotokoll 6-Byte-Protokoll (5+1)

Byte

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

=> 50,000 Hz

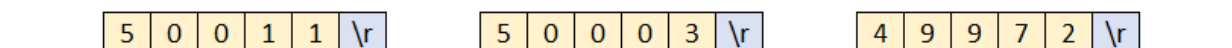


Bild: Aufbau und Telegrammverlauf 6-Byte-Protokoll

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

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

Sonstiges / Erweiterungen:

Stiftleiste J5

J5 vorbereitet für Protokollauswahl (serielle Schnittstelle)

Stiftleiste J6

J6 vorbereitet als universelle IO-Schnittstelle (z.B. Relaisausgabe)

Nachbau

DIL-Version V1.0

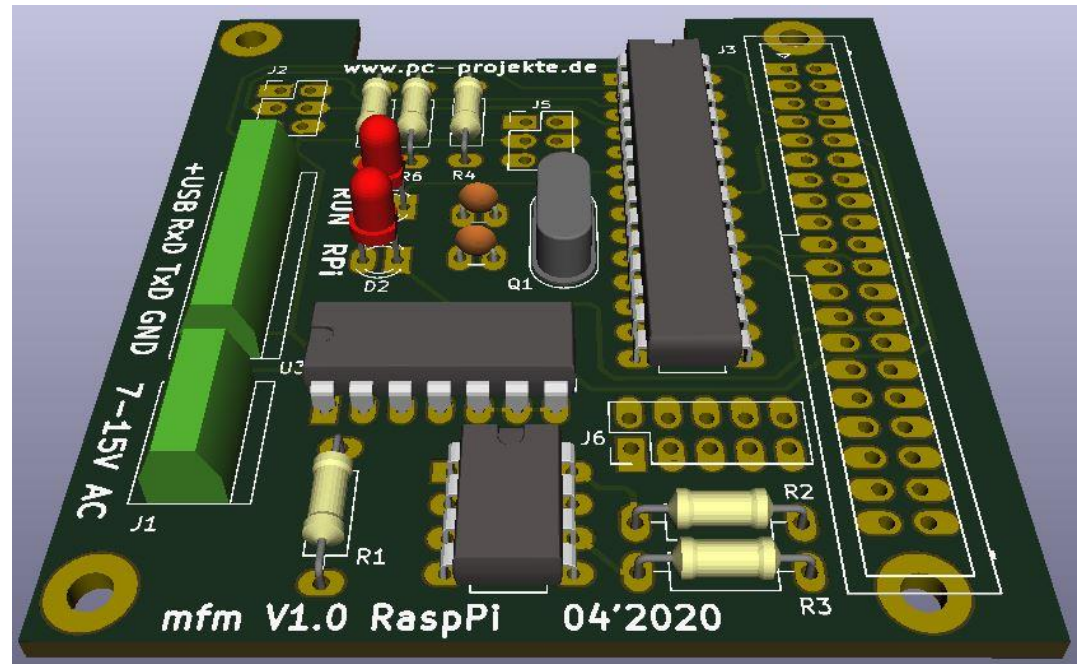
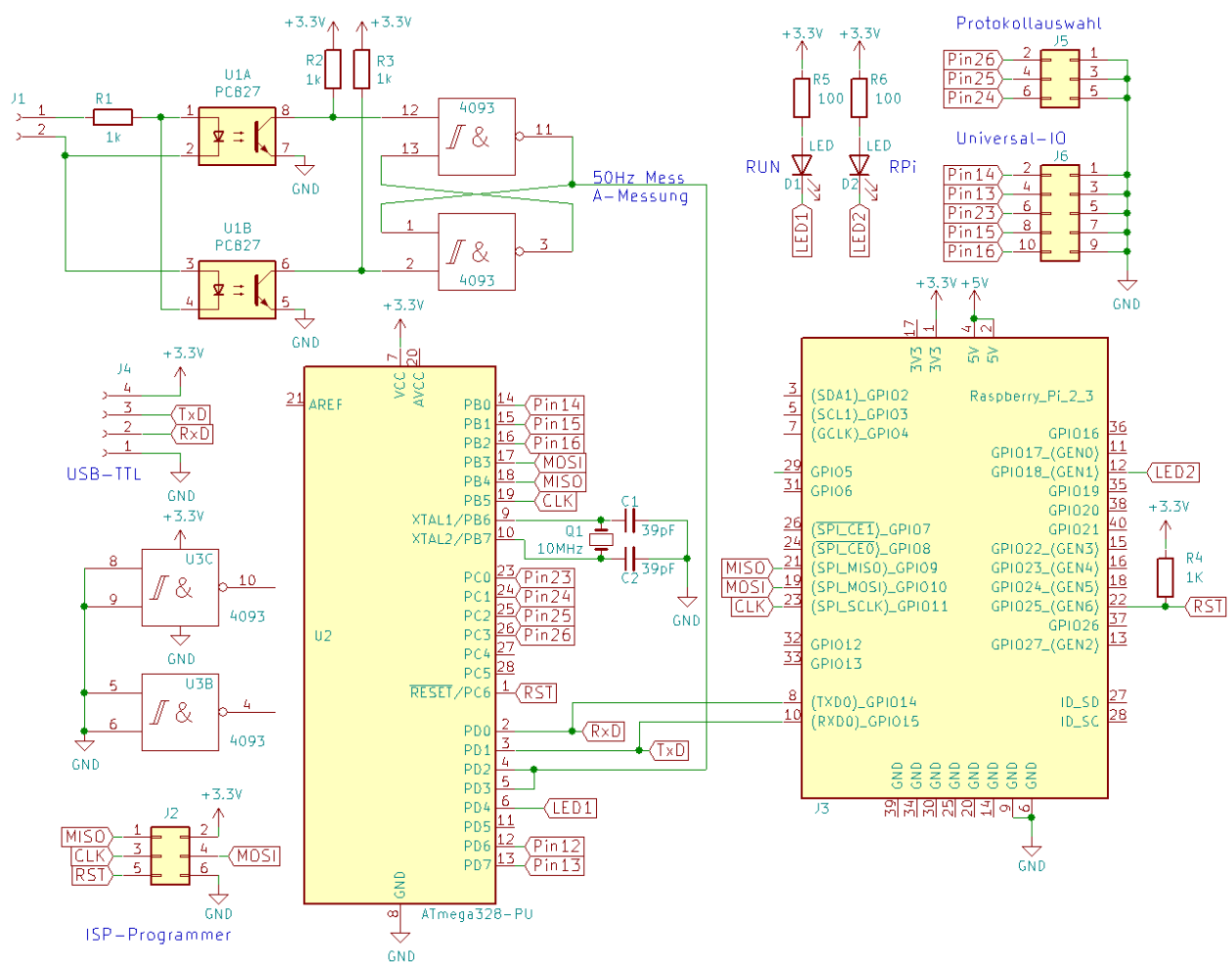


Bild: Platine Netzfrequenzmessung mfm V1.0 RaspPi



Schaltbild Netzfrequenzplatine mfm V1.0

Aufbauskizze

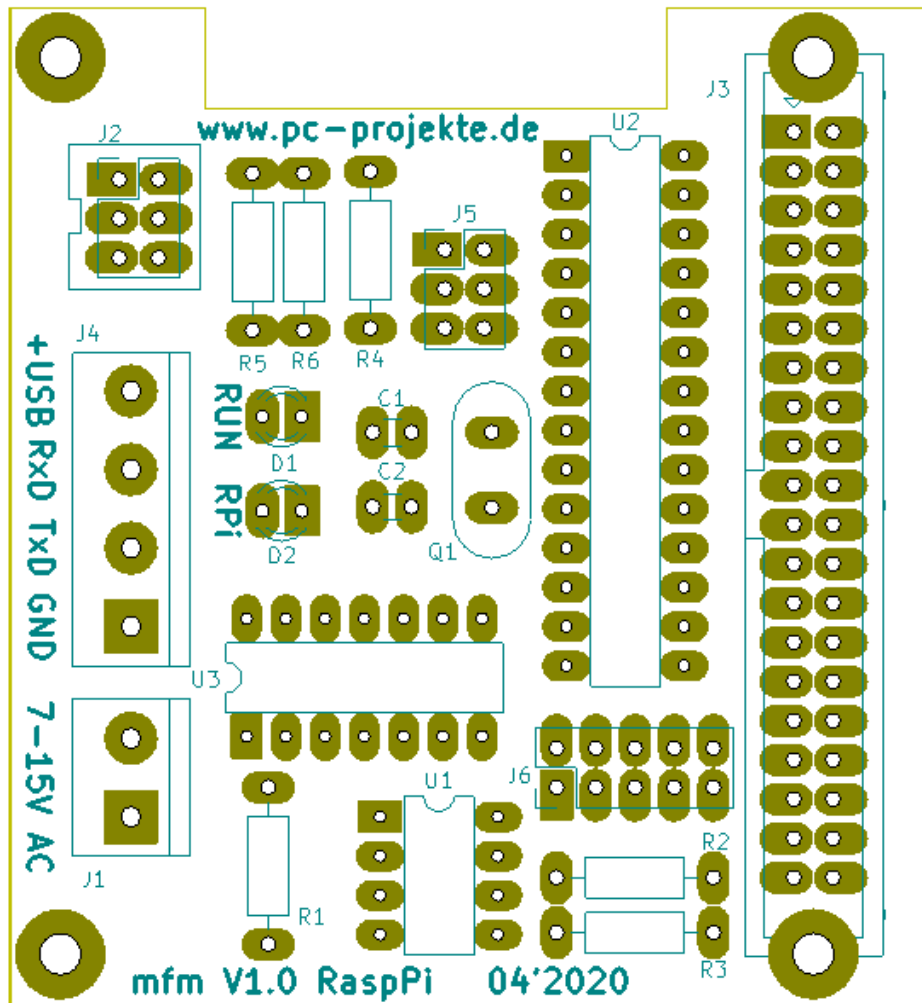


Bild: Aufbau Bauteile



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

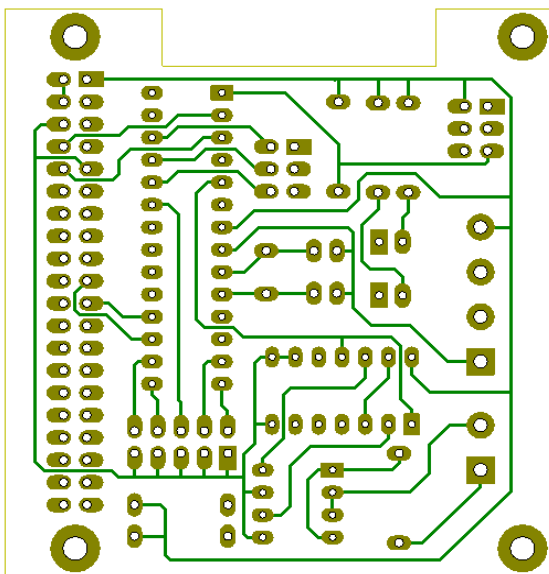
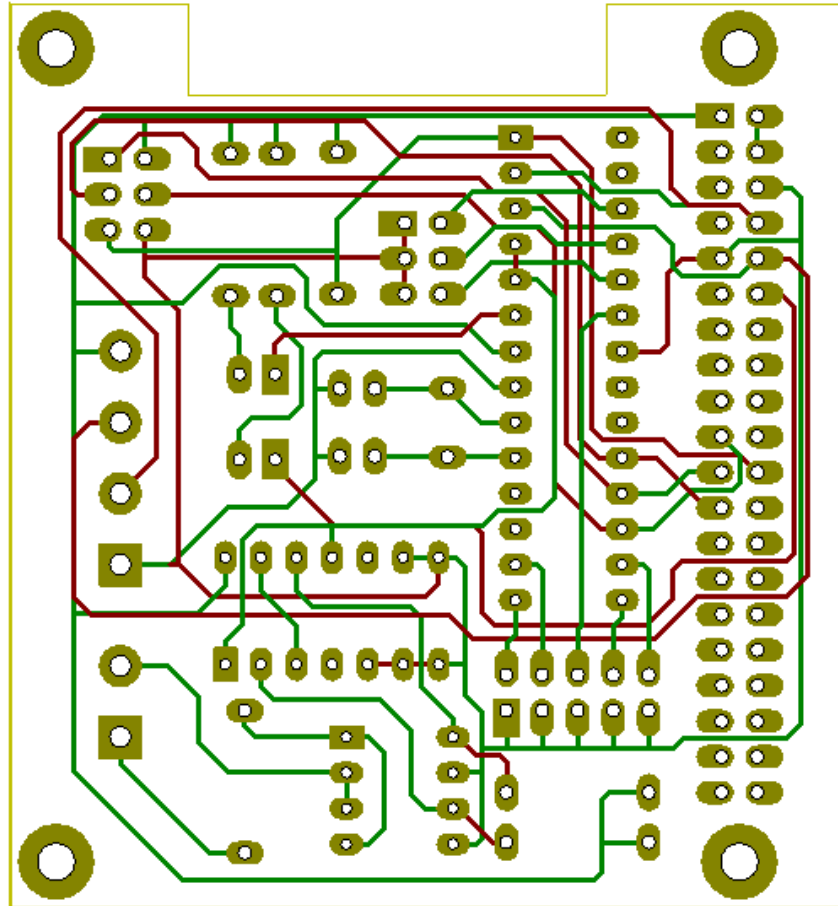


Bild: Platinenlayout Rückseite

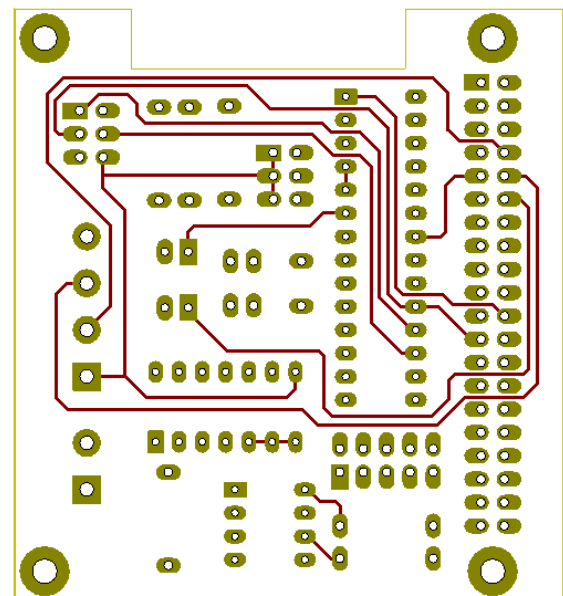


Bild: Platinenlayout Vorderseite

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	ATMega328	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 2X12 Pol	RaspberrPi Buchsenleiste	RaspPi Connector Pin 1-24
(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	
J4	4-Pol Schraubklemme	Klemme (2X2 Schraubklemme)	Bei USB-TTL-Verwendung
J5	Stiftleiste 2X3-Pol	Jumper	Universelle Eingabe-Erweiterung
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 Eingangswiderstand 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)

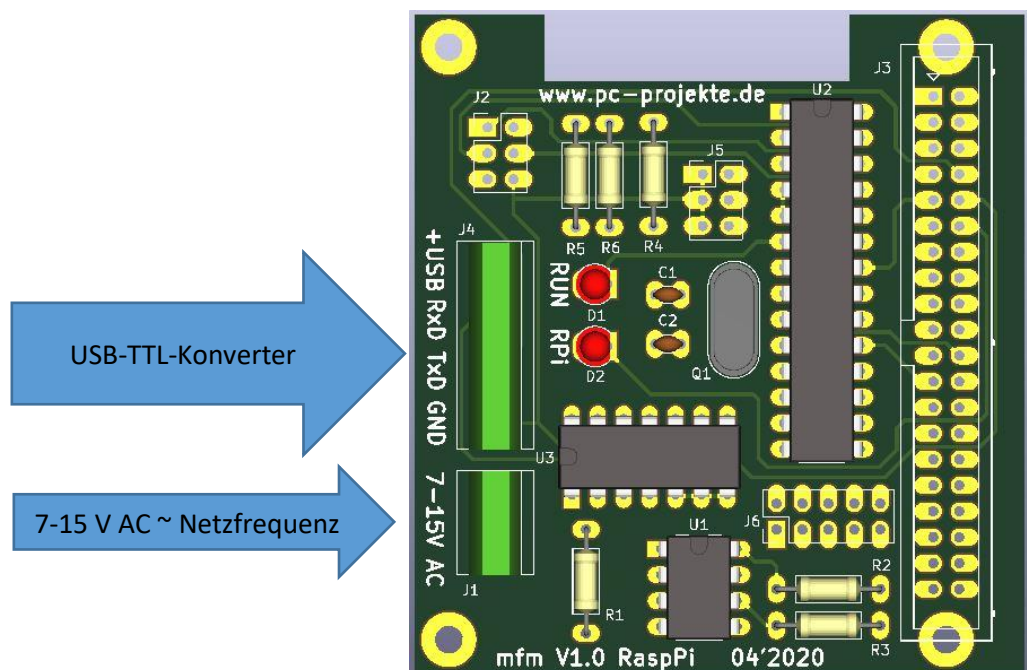


Bild: Anschlussschema USB-TTL-Konverter

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:

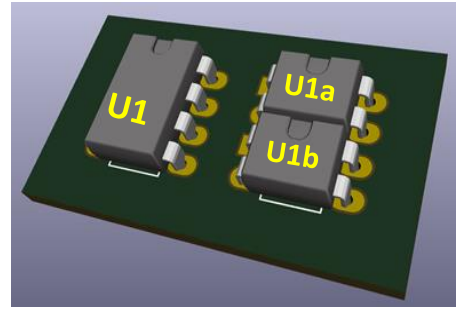
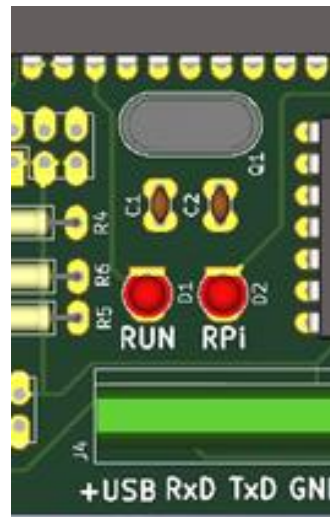


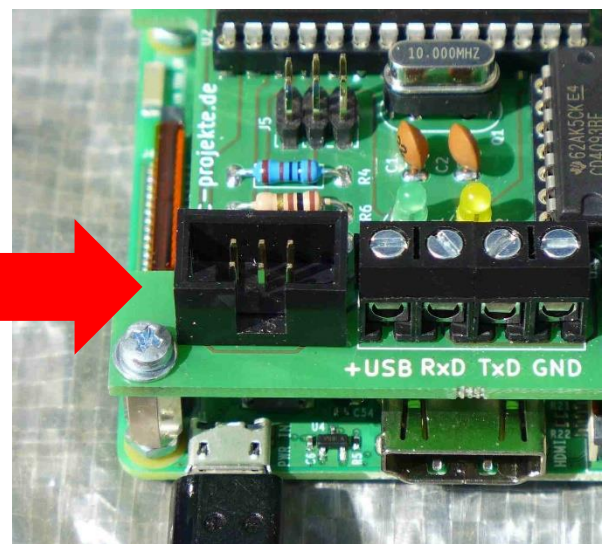
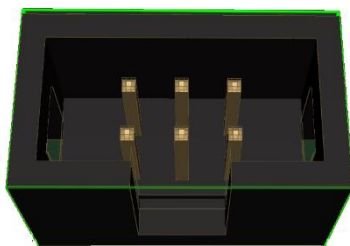
Bild: alternative Bestückung 2 Stk. PC817 statt 1 Stk. PC827

Einbaulage LED 1 und LED 2

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



Einbaulage J2



Abstand des Quarz zur Platine

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

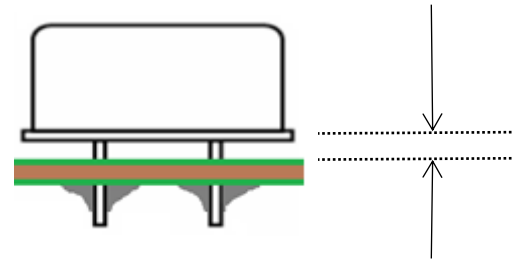


Bild: Quarzabstand zur Platine

Bauteilinformation: Quarz und C1 / C2

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

- Frequenz: 10,0000 MHz
- Cl: 32 pF ==> **$C_1 / C_2 = 22\text{pF}$** Lastkapazität im Datenblatt beachten!
- R_{smax}: 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

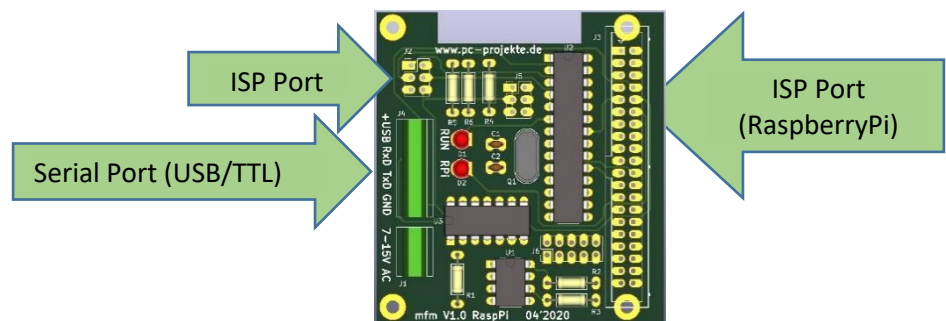
Software / Quellcode

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

Die Basisversion beinhaltet das einfache Grundprogramm der Netzfrequenzmessung und kann für eigene Projekte verwendet bzw. angepasst werden. Die Erweiterungen sind in Kürze erhältlich (ebenfalls unter den GNU GLP3 Lizenzen).

Wichtiger Hinweis zur Arduino-Kompatibilität:

Die Netzfrequenzplatine arbeitet mit 3,3V (RaspberryPi). Durch die 3,3V kann nur ein maximaler Quarz von 10MHz verwendet werden. Deshalb müssen einige Controller-Fuses umgestellt werden. Eine direkte Arduino-Programmierung über die serielle Schnittstelle ist deshalb in dieser Hardware-Version nicht 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 Betriebsspannung 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.

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   = "mfm";  
  desc = "Use the Linux sysfs interface to bitbang GPIO lines for mains frequency monitor";  
  type = "linuxgpio";  
  reset = 25;  
  sck   = 11;  
  mosi  = 10;  
  miso  = 9;  
;
```

Speichere die geänderten Einstellungen

3. Test AVRdude / Auslesen der Fuses

```
sudo avrdude -c mfm -p atmega328p -U lfuse:r:-:b
```

4. Setzen der benötigten Fuses mit folgenden Kommandozeilen:

```
sudo avrdude -c mfu -p atmega328p -U lfuse:w:0b11111111:m
```

```
sudo avrdude -c mfu -p atmega328p -U hfuse:w:0b11011001:m
```

```
sudo avrdude -c mfu -p atmega328p -U efuse:w:0b11111111:m
```

5. Übertrage das aktuelle HEX-Programm

(aktualisiert mit Arduino-IDE oder als Download von der Projektseite...)

```
avrdude -p mfm -c atmega328p -P flash:w:mfm_1.hex
```

weitere Übertragungsmöglichkeiten finden Sie in Kürze auf der Projektseite...

Messung / Ausgabetest

Überprüfung der Funktion mit einem Terminalprogramm:

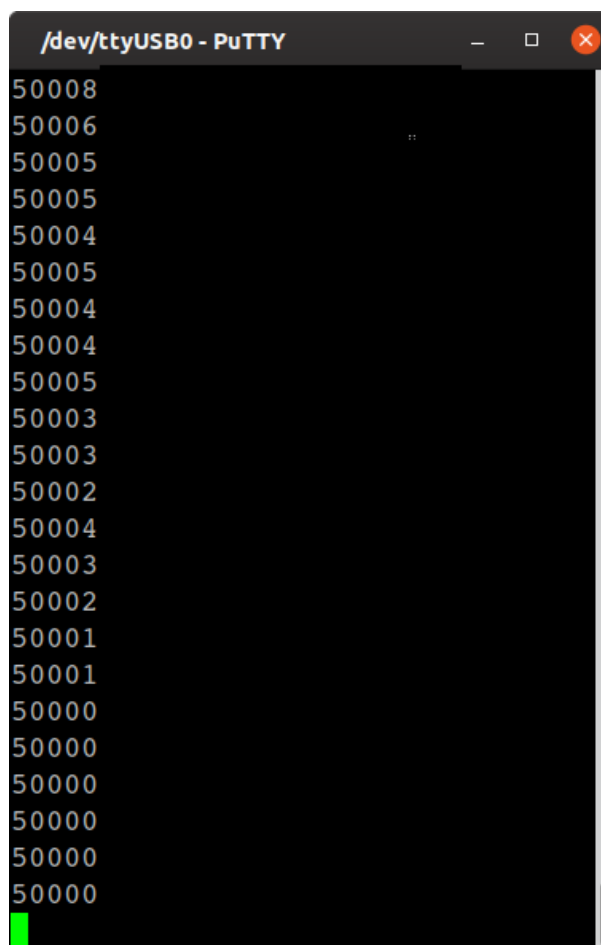
Öffnen sie ein Terminalprogramm z.B. PUTTY

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.



The image shows a screenshot of a PuTTY terminal window titled "/dev/ttyUSB0 - PuTTY". The window displays a list of frequency measurements in Hz, starting from 50008 and ending at 50000. The values are: 50008, 50006, 50005, 50005, 50004, 50005, 50004, 50004, 50005, 50003, 50003, 50002, 50004, 50003, 50002, 50001, 50001, 50000, 50000, 50000, 50000, 50000. A green cursor is visible at the bottom left of the terminal window.

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 ioBrockers, dem OpenHAB (Open Home Automation Bus) oder dem OpenMUC (Web-Interface zur Systemkonfiguration und Visualisierung) problemlos visualisiert werden.

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

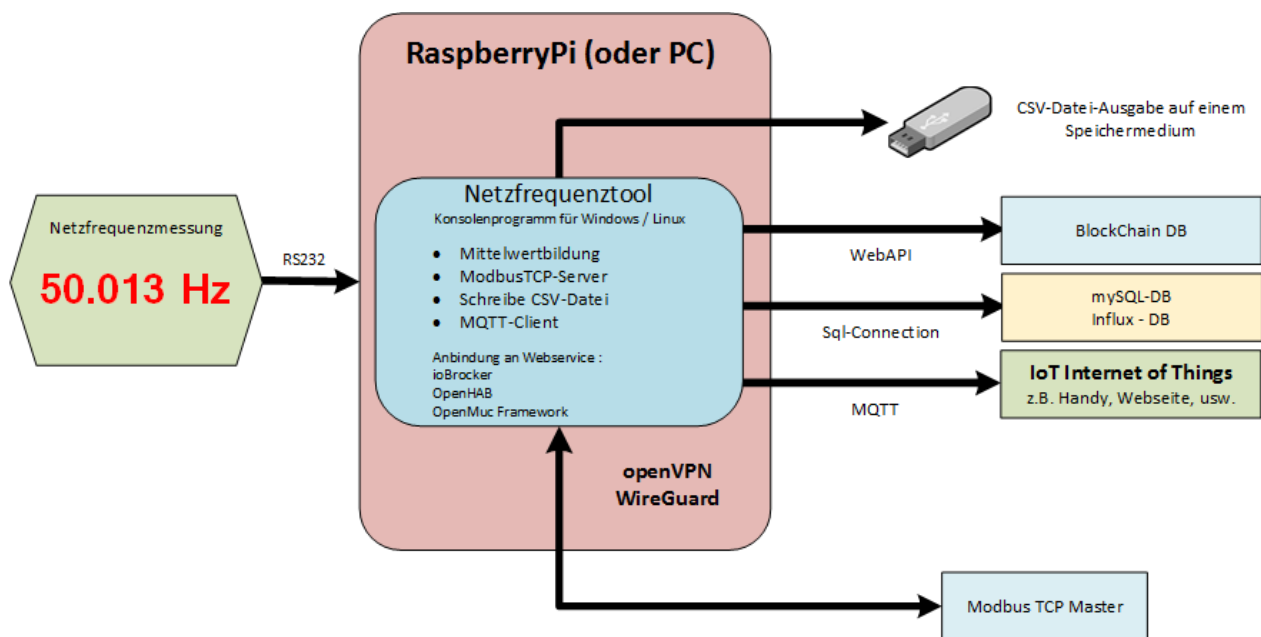


Bild: Schema Netzfrequenzmessung und Anbindung an eine Leittechnik

- 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>

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.