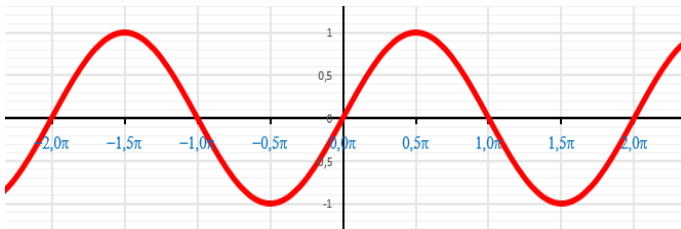


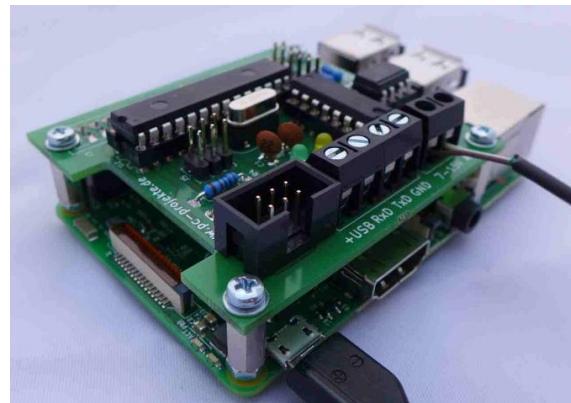
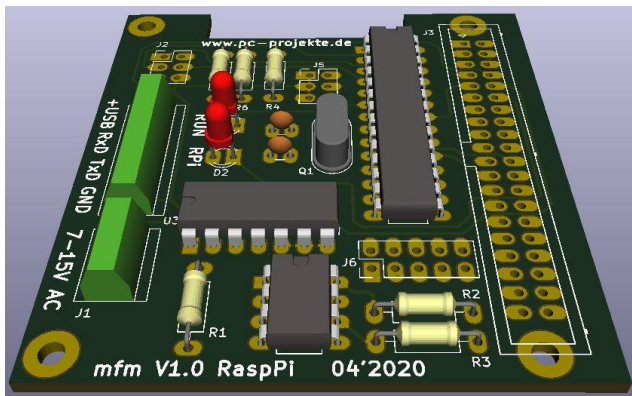
# MAINS FREQUENCY MONITORING UNIT

## Mains frequency measurement for the RaspberryPi (and USB-TTL converter)

### Basic documentation



The view into our power grid



#### Project description MAINS FREQUENCY MONITORING UNIT V1.0

Measure and monitor your mains frequency. The deviation from the Setpoint provides a direct quality indicator of the instantaneous power offered of the power plants and consumer demand. The grid frequency is therefore a Guarantor of grid stability and shows effects of intraday electricity trading

## **Inhaltsverzeichnis**

<b>License .....</b>	<b>2</b>
<b>Creative-Commons-License .....</b>	<b>2</b>
<b>OpenHardware (OSHWA).....</b>	<b>2</b>
<b>Software / source code license .....</b>	<b>2</b>
Background .....	3
Our power grid .....	3
Mains frequency as a quality indicator .....	3
Using the mains frequency board .....	5
Software .....	5
Measuring voltage .....	5
<b>Frequency measurement .....</b>	<b>6</b>
See also Measurement data preparation.....	6
Settings for the prescale measuring method .....	7
Measurement of the mains frequency .....	9
<b>Input circuit Schmitt trigger .....</b>	<b>10</b>
Evaluation with the $\mu$ Controller INTO / INT1.....	12
Evaluation of the measurement counters .....	13
Difference Comperator Filter .....	14
LEDs .....	16
Use with a USB-TTL converter .....	16
<b>Elektronic kit .....</b>	<b>17</b>
Layout sketch .....	18
Board layout .....	19
Components: .....	20
Installation instructions:.....	22
<b>Software / source code.....</b>	<b>24</b>
$\mu$ Controller-Fusebits.....	24
Transfer via the RaspberryPi with AVRDUde (hex file via Raspi ISP port) .....	25
Check the function with a terminal program .....	26
Communication of the mains frequency board .....	27
<b>Information / Order possibility .....</b>	<b>28</b>
<b>Notice / exclusion of warranty.....</b>	<b>28</b>

## License

### Documentation

#### *Creative-Commons-License*

This project documentation is licensed under the Creative Commons Attribution - Distribution under Equal Conditions 4.0 International. To view a copy of this license, visit

<http://creativecommons.org/licenses/by-sa/4.0/>



This document is a base document and is part of the OSHWA Openhardware release.

A project extended version with current features and extensions will be available for download soon.

### OpenHardware (OSHW)



### Software / source code license

The source code and this is under the GNU GPL 3 license:

- Freedom to run the program for any purpose
- Freedom to study and adapt the source code
- Freedom to copy the program
- Freedom to copy the modified program



These freedoms should also be guaranteed in the long term. To this end, the GPL contains two essential clauses:

- Every derivative must also be fully licensed under the GPL.
- If the program is distributed in binary form, the source code of the entire program must be provided or handed over on request.

## Background

### *Our power grid*

The mains frequency in Europe is 50Hz. A deviation from the nominal value is a direct window of electrical generation and consumption. The power output must be matched by an equal power consumption at all times.

If there are deviations, this leads to a change in the grid frequency: If there is an oversupply of electrical power, the grid frequency increases, if there is an undersupply, the frequency decreases. Normally, these deviations are minimal in the Western European interconnected grid and range from 49.80Hz - 50.20Hz. The task of the so-called power control in the electricity grid is to compensate for the load fluctuations and thus to keep the grid frequency at a constant nominal value.

In Europe, Asia, Australia, most of Africa and parts of South America, a grid frequency of 50 Hz is used for the general electricity grid, in so-called interconnected grids. In North America, a grid frequency of 60 Hz is used in the public power grid. The differences are due to the historical development history of the first power grids in the 1880s and 1890s and have no technical reason today.

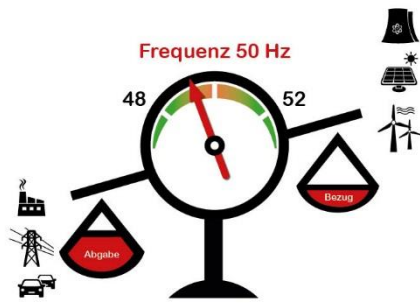
Some railways, such as the ÖBB, SBB and Deutsche Bahn use a nominal frequency of 16.7 Hz for their traction power supply. In the past, the nominal railway network frequency was  $16 \frac{2}{3}$  Hz, which is exactly one third of the 50 Hz used in the interconnected network. Some railways and also industrial customers in North America are supplied with a grid frequency of 25 Hz for historical reasons. The comparatively low grid frequencies result from the technological development of the first electrical machines: At the beginning of the 20th century, it was only possible to build electrical machines of greater power with these low frequencies. However, due to the great effort involved in the conversion, the low mains frequencies introduced at that time are still used today.

In special areas, e.g. in the on-board power supply system of aircraft, higher mains frequencies are common, e.g. 400 Hz, as smaller and lighter transformers can be built for this purpose and the cable lengths are short.

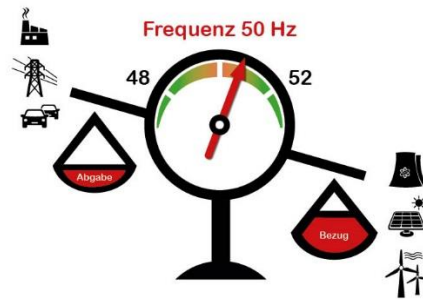
### *Mains frequency as a quality indicator*

The grid frequency and its deviation from the nominal value is a direct quality indicator of the relationship between the instantaneous electrical power offered by producers such as power plants and the decrease in instantaneous electrical power by consumers. Electrical energy can hardly be stored in interconnected networks, but only distributed between the producer and the consumer. With the exception of the reactive power of alternating current, the power output must be balanced by an equally high power consumption at all times.

If there are deviations, this leads to a change in the grid frequency in AC voltage grids: If there is an oversupply of electrical power, the grid frequency increases; if there is an undersupply, it decreases. Normally, these deviations are minimal in the Western European interconnected grid and are below 0.2 Hz. A frequency deviation of 0.2 Hz corresponds to a power difference of approx. 3 GW in the European interconnected system, which also corresponds to the so-called reference failure from the Continental Europe Operation Handbook and is approximately equivalent to the unplanned outage of two larger power plant units. The task of power control in interconnected grids is to balance out the temporal fluctuations and thus to keep the grid frequency at a constant nominal value. The smaller an electricity supply network is and the worse the network control functions, the greater the fluctuations in the network frequency.



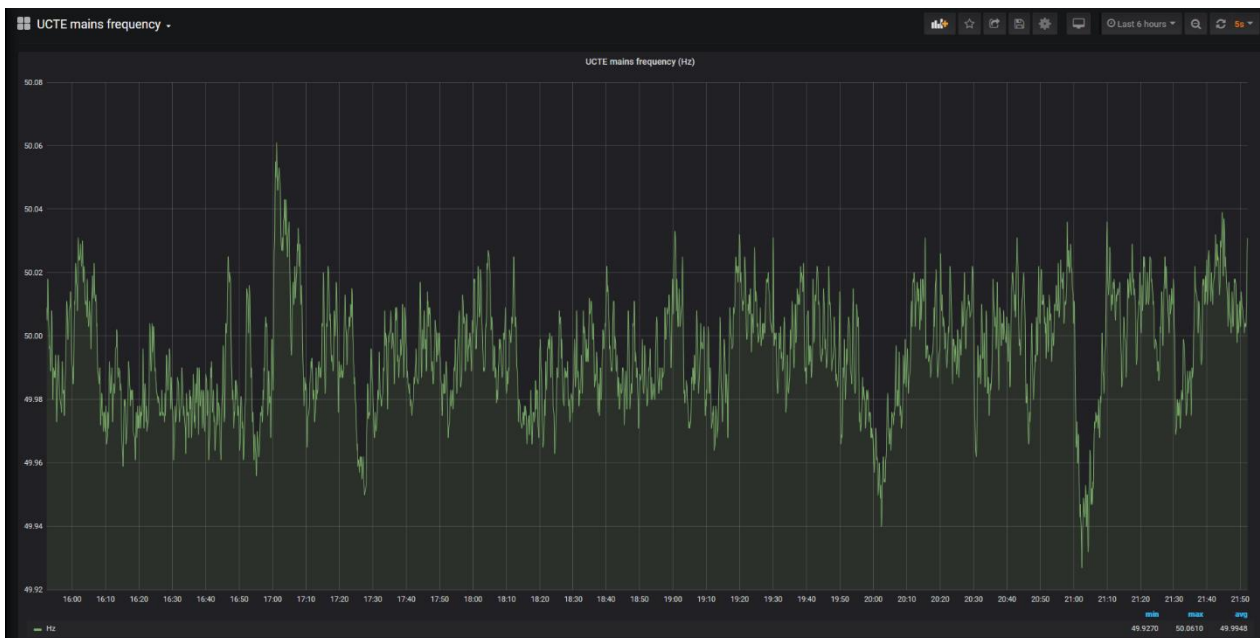
Undersupply of energy



Oversupply of energy

### This fluctuation can be measured:

With my project, I would like to make the dynamics of the power grid visible. The electronics record the grid frequency and, derived via a mathematical function, enables the grid load difference to be displayed in MW (megawatts). This grid load difference corresponds to the primary control power.



Picture: Mains frequency measurement (example graphic evaluation with Grafana®)

This is a private, non-commercial project. The electronics,  $\mu$ Controller and project software is developed by me (see also Creative Commons License).

**Boards, programming service and components see ordering option (page 28).**

## **Measuring board for recording, information, message, alarm and monitoring purposes of the mains frequency**

The device can be flexibly adapted to the characteristics of a power grid (e.g. 60Hz grid). A calculation and output of the network load difference (primary control power) is in preparation.

The output is via the serial interface. The board is designed for the RaspberryPi, but can alternatively be used with a USB-TTL converter.

Furthermore, output outputs for alarm and message systems are prepared.

A provision of the measured values via a Modbus-TCP software will be available soon, further protocols e.g. MQTT (IoT), IEC60870-4-104 FW protocol or as logic device of an IEC61850 system are planned.

### *Components (Information)*

No smd technology is used to make it as easy as possible to reproduce. The corresponding reproduction instructions and component lists can be found from page 20 onwards (for ordering the board and components, see page 26). A partially populated board with SMD is in preparation.

### *Software*

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

The latest software can be downloaded from the Gitlab site or from [www.pc-projekte.de](http://www.pc-projekte.de) or <http://www.netzfrequenzanzeige.de>.

### *Measuring voltage*

**Warning: NEVER use direct 230V mains voltage for mains frequency measurement!**

Measuring voltage: power supply unit (or transformer) with AC voltage



Picture: bell transformers



Plug-in power supply unit (AC-AC)



Print transformer

See also [note / exclusion of warranty](#).

## Frequency measurement

### Description of measuring procedure

The frequency measurement is carried out via an edge evaluation of the AC voltage signal. The signal is fed to the  $\mu$ Controller via two optocouplers and an RS-FlipFlop (4093-nand gate as RS-FlipFlop) and evaluated with the 10MHz quartz clock and the Timer1 (as counter) of the  $\mu$ Controller

### Measurement accuracy

The accuracy of the mains frequency measurement is very high even without calibration. Decisive is the accuracy of the quartz Q1 (and the correct size C1 and C2, matched to Q1 ).

The measurement error of the frequency measurement (without calibration or mains frequency trigger circuit board) is less than  $\pm 1$  mHz.

A calibration of the measurement is possible via a source code setting.

### Temperature range

The mains frequency display should be used at room temperature. Direct sunlight on the housing or the board and the associated temperature effect can falsify the measurement result.

### Powerquality

The mains frequency display is normally insensitive to mains feedback described in EN50160.

Very strong electrical pulses in the environment, unusually high harmonic content or ripple control signals in the measurement voltage (mains feedback) can falsify the measurement result.

*See also Measurement data preparation*

### Measuring range recognition at program start

An automatic mains frequency recognition is carried out when the program is started.

After successful detection, the corresponding limit ranges of the detected mains frequency are used in the filter stages.

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 networks

Rail network frequency e.g. Germany, Austria and Switzerland (standard gauge)

16.7 Hz measurement is basically possible, but is not integrated at the moment

The corresponding mains frequency can be set manually

automatic frequency detection	50 Hz / 60 Hz (16,7 Hz )	<b>FA = 0</b>
50 Hz mains frequency	45,000 Hz – 55,000 Hz	<b>FA = 50</b>
60 Hz mains frequency	55,000 Hz – 65,000 Hz	<b>FA = 60</b>
<i>A version for 16.7 Hz is prepared</i>	<i>12,000 Hz – 22,000 Hz</i>	<i><b>FA = 16</b></i>

## Settings for the prescale measuring method

Some settings can be made for the measurement procedure in order to adapt or optimize the measurement duration and measurement periods to the  $\mu$ controller.

For this purpose some parameters can be set or changed in the  $\mu$ Controller program.

### Settings in the Arduino source code:

Setting name	Default setting	Function
Prescaler	64	Internal divider of the $\mu$ C clock (ATMega328 clock)
Messperiode	18	Maximum period count
Dividend	2812500	(edge counter until next frequency calculation)
Sperr	2400	Constant value for frequency conversion
UBRR	0X40	(results from the prescaler and the measuring period)
DiffMAX	60	Blocking time after detected input edge to suppress signal bounce
FA	0	Correction value for serial interface
Calibration	0	(adaptation of the baud rate to the 10MHz clock)

The settings are described in detail on the following pages.



Deployment	$\mu$ C Oscillating Quartz	Prescaler	Period	Dividend	Lock- Counts	UBRR (U2X=1)	Approximate measuring time at 50Hz
USB-TTL A adapter (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
	<b>10 MHz</b>	<b>64</b>	<b>18</b>	<b>2812500</b>	<b>2400</b>	<b>0x40</b>	<b>0,36 s</b>
	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

<== Recommended setting Raspberry Pi

- Crystal 10MHz
- Prescaler 64
- Period 18
- Dividend 2812500
- Lock counts 2400
- UBRR 0x40

Programming via ISP port

Table: optimized setting data for mains frequency measurement

Settings in the Arduino program code (basic source code):

```
// 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
// *****
```

All program settings can be found in lines 10 - 20 in the Arduino source code (base code)

## Measurement of the mains frequency

Basic principle is the timer1 of the  $\mu$ Controller, which is set to the fixed clock 10MHz DIV 64 without further processing when the  $\mu$ Controller is started. With this setting, Timer1 becomes the reference clock from which the mains frequency is now derived. The total time length of the passing Timer1 (0 ... 65535) is 423ms.

This optimizes the number of periods for mains frequency measurement (for 45 Hz) to 18.

### Calculation

Timer 1 runs in the set prescale mode with the (divided) clock of the  $\mu$ controller as counter.

The mains frequency is fed to the INT0 of the  $\mu$ controller via the optocoupler and the Schmitt trigger. A recognized trigger event leads to an internal incrementing of the period counter.

The trigger inhibit time (inhibit/lockcounts 2400) prevents a too fast multiple evaluation of the edge (e.g. due to an interference signal).

When the maximum period counter value is reached, the counter value from Timer1 is taken over.

The counter value of Timer1 becomes the basis for calculating the grid frequency as a divisor with the constant dividend (2812500). In addition, the factor 1000 is multiplied. The mains frequency is then output in mHz. This has the advantage that all mains frequency calculations up to this point are performed exclusively in fixed-point arithmetic. This process is therefore carried out very quickly in the  $\mu$ C (thus even executable on an ATTiny).

#### Calculation:

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

fMess (variable name in source code) in mHz (Ex. 50021 = 50.021 Hz)

The preliminary frequency value (fMess) is now passed through several filter stages.

The output (frequency) is a serial protocol on the TxD port of the  $\mu$ Controller

### UBRR

To adapt the 19200Bd output of the serial interface to the 10MHz crystal, the UBRR register value must be adjusted by input.

For a 10MHz crystal UBRR = 0x40 must be set.

## Detailed description of the measurement

### Input circuit Schmitt trigger

The input circuitry (mains frequency measuring input) is electrically isolated via 2 optocouplers. The measuring voltage is supplied via a transformer and should lie between 7V- 15 V ~ AC. The current of the used measuring voltage is not more than 20mA. A short-circuit-proof transformer or a bell transformer should be used for the measurement. The input circuitry of the optocouplers U1 (U1a/U1b) is current limited by R1 (see also Information Components page 20).

The optocouplers drive Schmitt triggers (CMOS 4093, IC U3) via a NAND gate, which then forwards the measurement signal as a square wave to the INT0 input of the  $\mu$ controller.

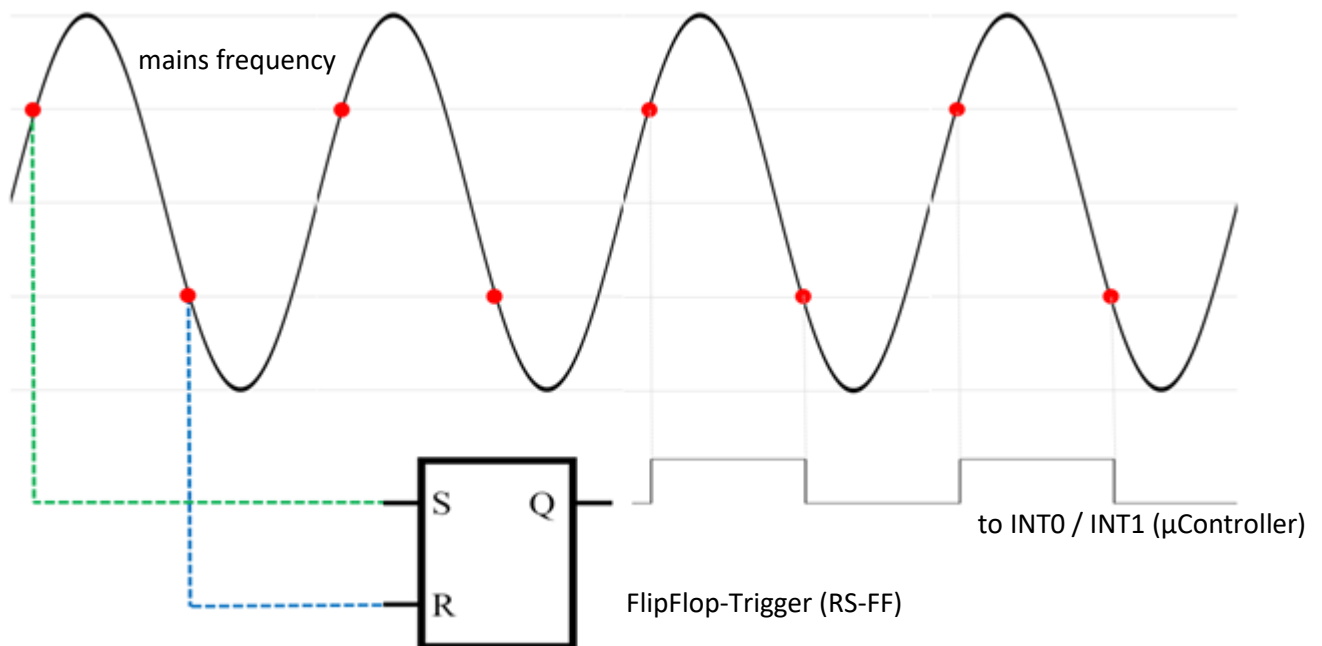


Figure: Schematic representation of edge evaluation (Schmitt trigger)

The advantage of this measurement data processing is that even a very noisy input signal is reliably detected.

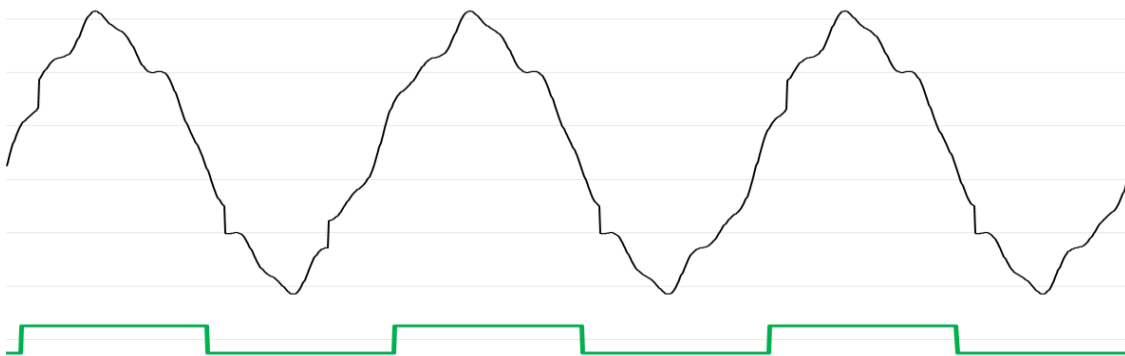
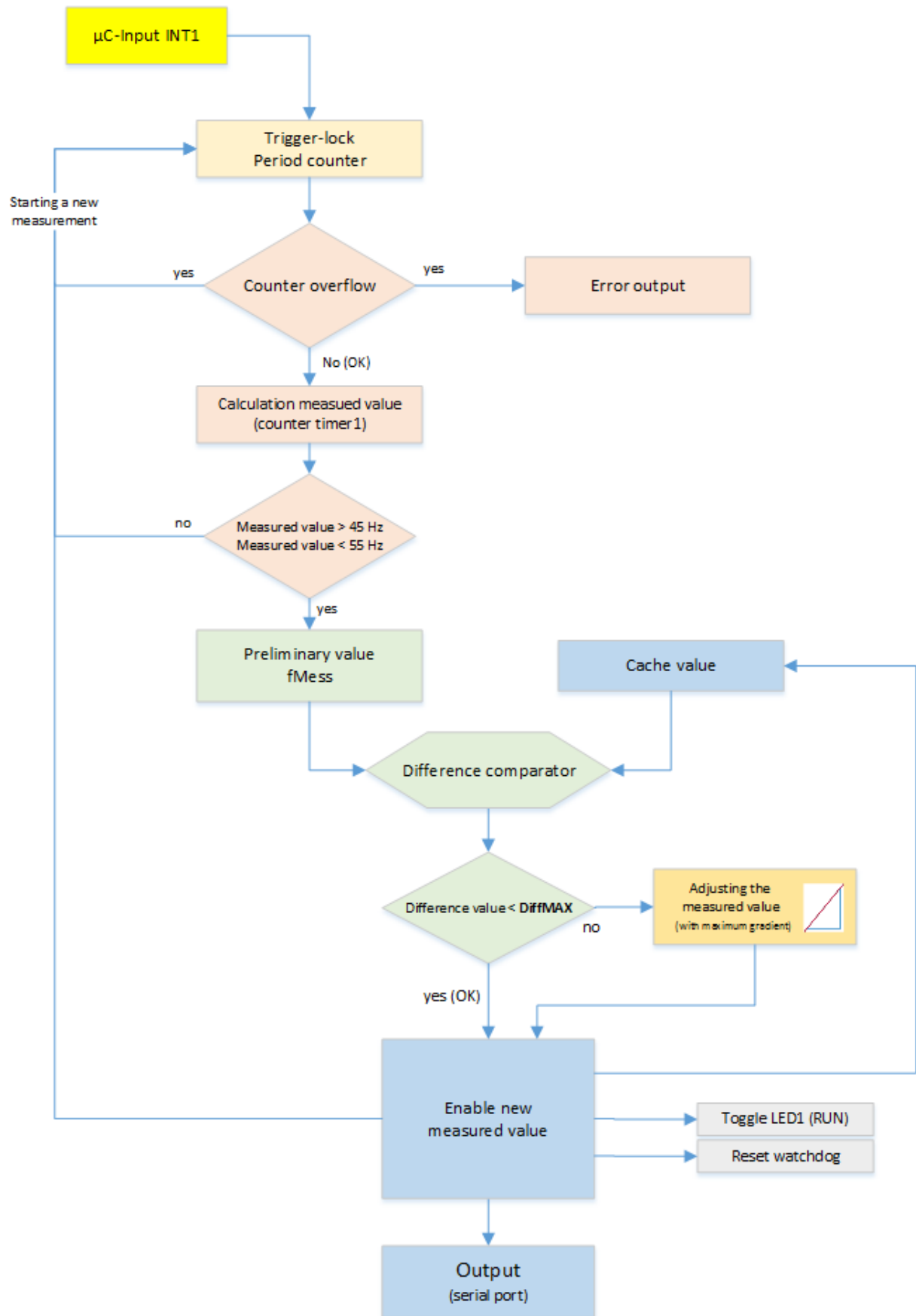


Figure: Schematic representation of a noisy input signal and with output signal (behind the Schmitt trigger)

*Schematic program sequence Measurement evaluation*



### Edge detection and trigger signal inhibit:

If the  $\mu$ controller detects an edge, a blocking time (approx. 15ms) is immediately started to suppress subsequent measurement peaks. A possible too early edge is immediately compensated by the subsequent stage, since an averaging process now follows.

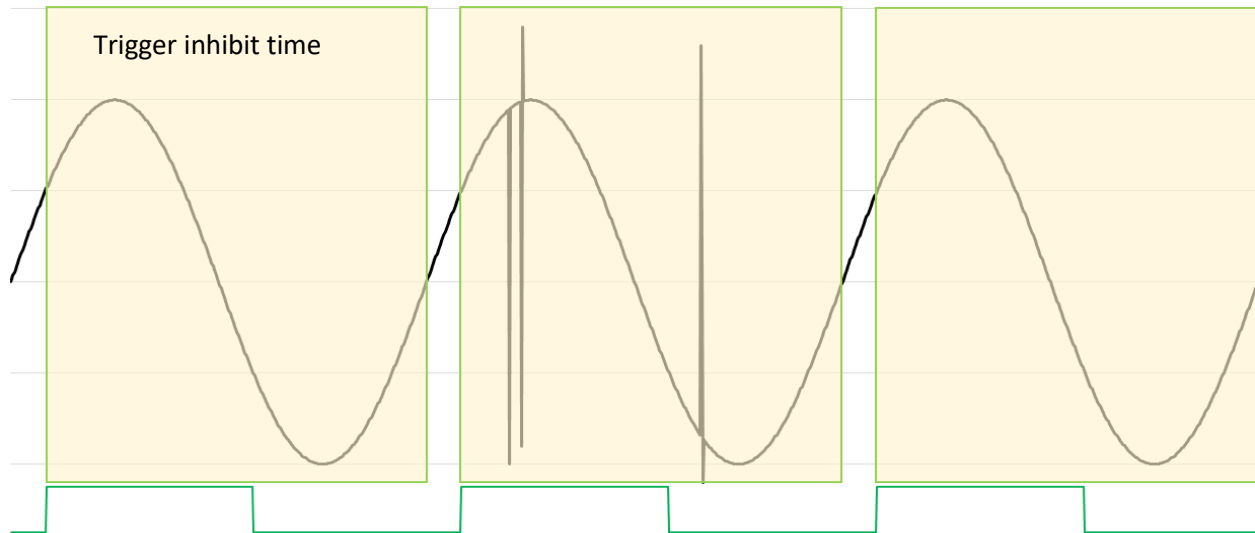
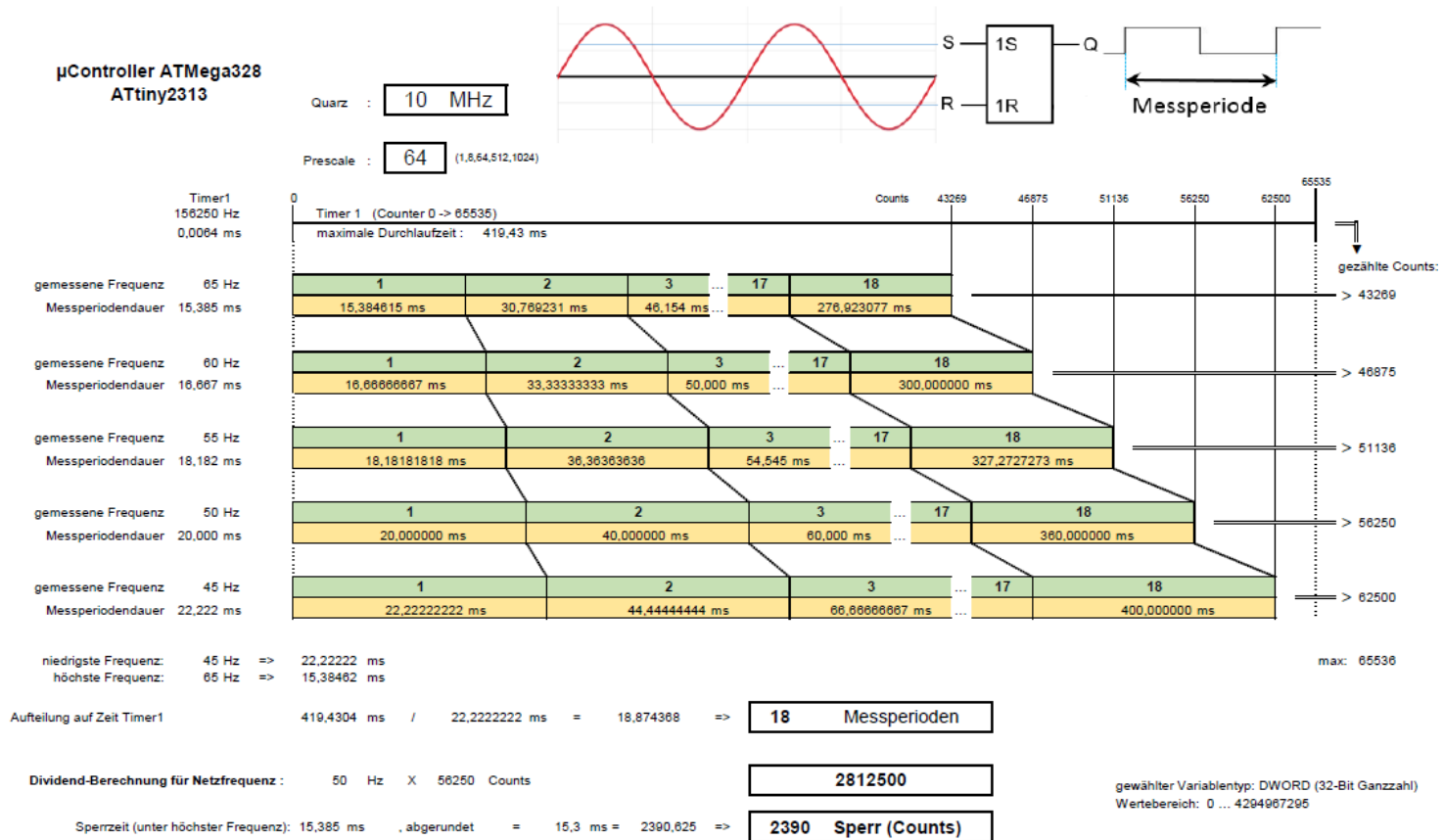


Figure: Schematic representation of the trigger inhibit time

### Lock-Counts

The blocking time after a pulse is selected so that the next trigger signal would only fit with a frequency below 65 Hz.

## Evaluation of the measurement counters



## Plausibility filter

Before the measured mains frequency value is further processed, the preliminary new measured value is subjected to a two-stage test: This determines whether the measured mains frequency value is basically within the permissible range:

Measurement	50 Hz measurement	60 Hz measurement	16.7 Hz*
Test f>	f > 45 Hz	f > 55 Hz	22 Hz
Test f<	f < 55 Hz	f < 65 Hz	12 Hz

After this first check, which excludes implausible measured values, it continues to the differential filter

### Functionality of the differential compensator filter:

The difference-comperator filter uses the default maximum mesh gradient. A mains frequency change above the maximum mains gradient is only partially taken into account.

However, if two consecutive measurements pass the validity check, even a very fast change is immediately accepted.

Procedure: The preliminary (new) measured value is compared with the last valid measured value. The difference between the two measured values is determined by calculating the difference.

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

If the difference (absolute value) is smaller than the default setting DiffMAX, the new value (Preliminary Value) becomes the valid measured value fMess.

If the difference is greater, the current measured value fMess is adjusted to the new value (Preliminary Value) at the level of the default setting DiffMAX.

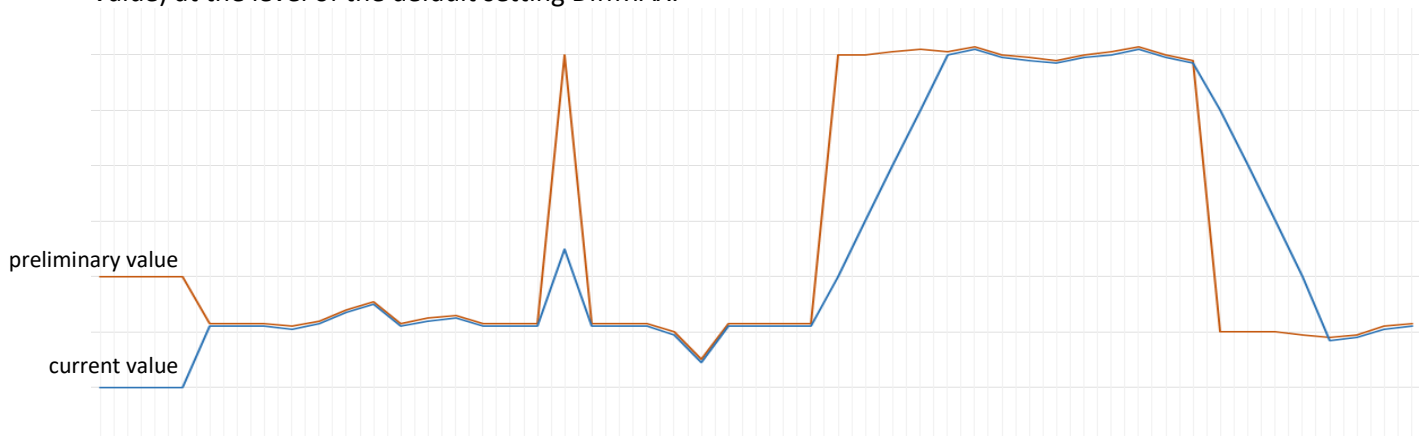


Figure: Schematic representation of the course of measurement data Difference-comparator-filter

The network gradients depend on many network circuit states, feeders, network load situations and the measurement location. Therefore, the following DiffMAX value is a recommended setting for the UCTE network.

The default setting of the DiffMAX filter is 60mHz/s (recommendation).

Programm Diff-Filter (Arduino-Programm)

```
word Diff_filter(word filter_wert)          // Measurement data in mHz in WORD format
{
    int Diff = filter_wert - Cachewert;
    if ( abs(Diff) > DiffMAX )                // if DiffMAX is exceeded
    {
        if ( Diff > 0 ) filter_wert = Cachewert + DiffMAX;    // Change (positive slope) max
        if ( Diff < 0 ) filter_wert = Cachewert - DiffMAX;    // Change (negative slope) max
    }
    Cachewert = (int)filter_wert;
    return filter_wert;
}
```

## Calibration

Normally the mains frequency board does not need calibration.

However, if calibration is required, a calibration value can be added to the setting. Shortly different settings a software tool (via an EEP file) can transfer changed settings to the EEPROM of the ATmega328. After a restart the changed settings are used.

## Frequency + Calibration → Output to serial interface

### Serial protocol

#### serial interface settings

Baudrate: **19200 Baud**      Parity: **None**      Stopbit: **1**      Handshake: **None**

#### UBRR = 0x40

In order to adapt the output of the serial interface to the 10 MHz crystal, the UBRR register value must be adapted via the input. For a 10 MHz crystal UBRR = 0x40 must be set.

### Selection of protocols

In the basic version this possibility is not yet realized. In the extendend version the output protocols can be selected via the jumper setting of the J5 interface.

#### Basic protocol 6-byte protocol (5+1)

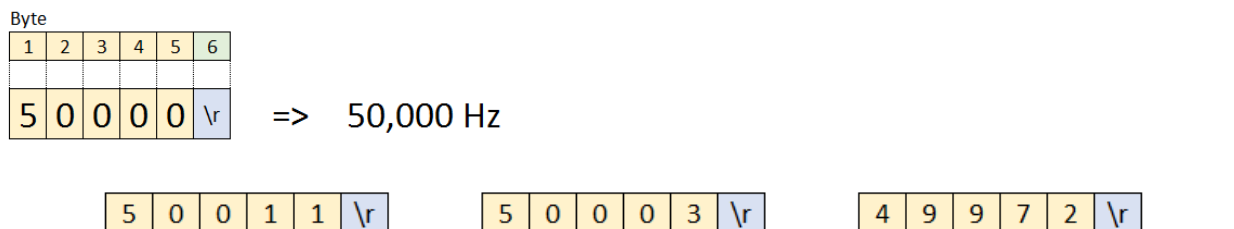


Figure: Structure and telegram sequence 6-byte protocol

From Output of the basic protocol on the serial interface is in **mHz**.



## LEDs

### LED 1: RUN LED

LED1 (RUN) Status change (toggle) at each successful filter run.

(toggle= changes the binary state)

Arduino output : 4 (=Pin6, PD4)

### LED 2: RPi-Led

LED 2 can be controlled by a program on the RaspberryPi.

RaspberryPi output : GPIO5 (=Pin12)

## Use with a USB-TTL converter

The mains frequency board can also be used without a RaspberryPi for mains frequency measurement. The data output and the power supply is then realized via the USB interface (USB-TTL converter).

For this operating mode the connector J4 is prepared.

JP4: This port provides the serial telegram (details see "serial protocol")

Pin	Bezeichnung	Funktion	Hinweis
1	<b>GND</b>	Ground / GND	
2	<b>RxD</b>	Receive direction (to USB), TTL level*	Vom $\mu$ C (TxD)
3	<b>TxD</b>	Receive direction (from USB), TTL level*	Zum $\mu$ C (RxD)
4	<b>+5V</b>	Supply voltage 5V DC	

\*) see TTL level

### Miscellaneous / Extensions:

#### Pin J5

J5 prepared for protocol selection (serial interface)

#### Pin J6

J6 prepared as universal IO interface (e.g. relay output)

## Elektronic kit

DIL-Version

mfm V1.0

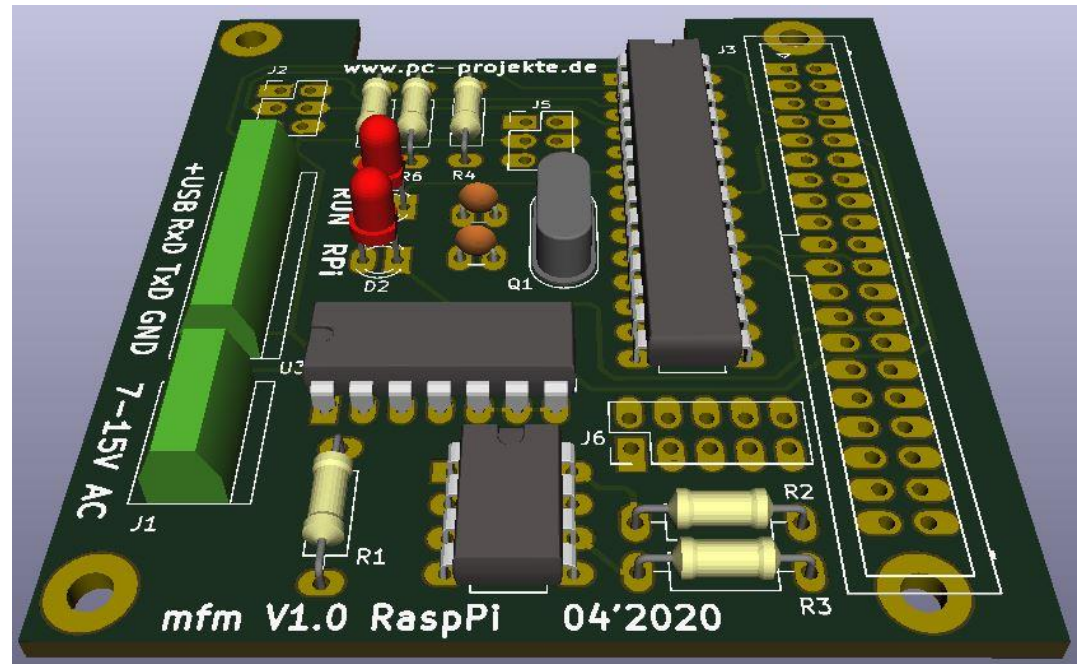
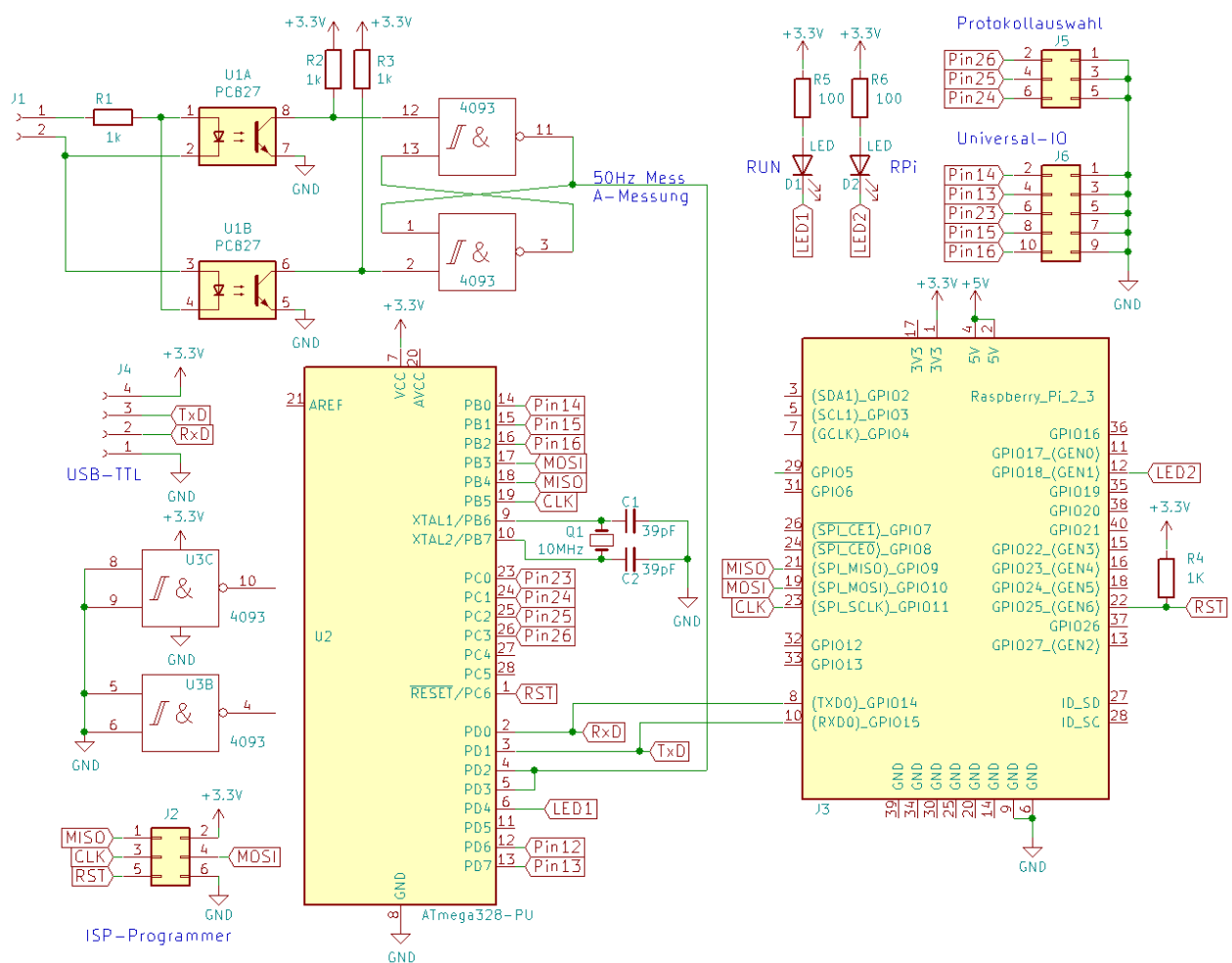


Figure: Board mains frequency measurement mfm V1.0 RaspPi



Circuit diagram mains frequency board mfm V1.0

Layout sketch

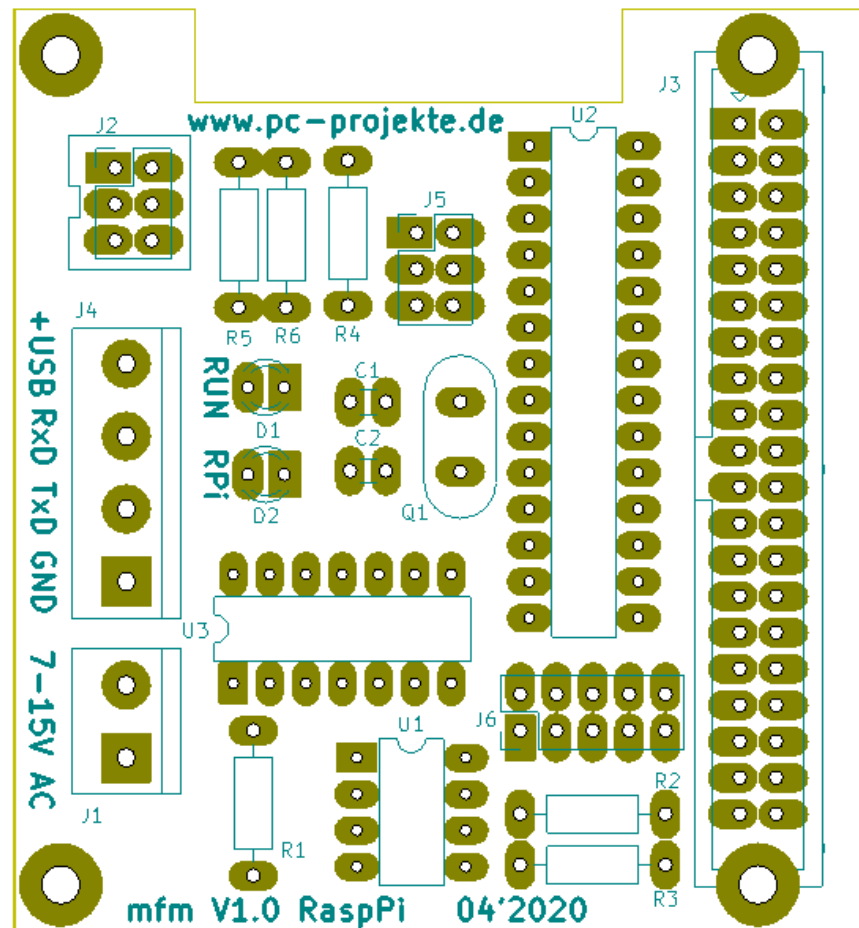
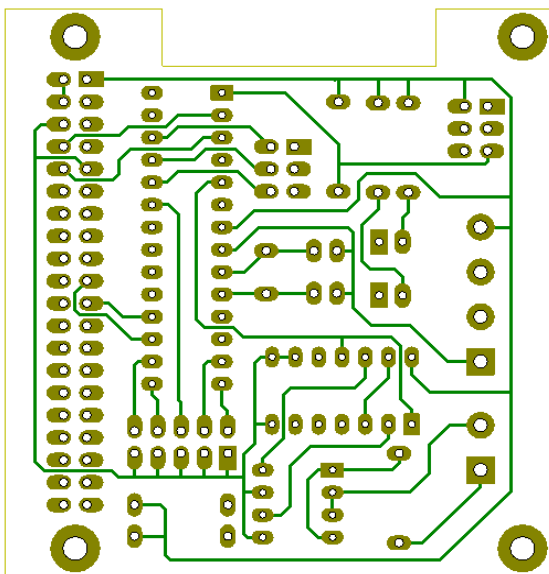
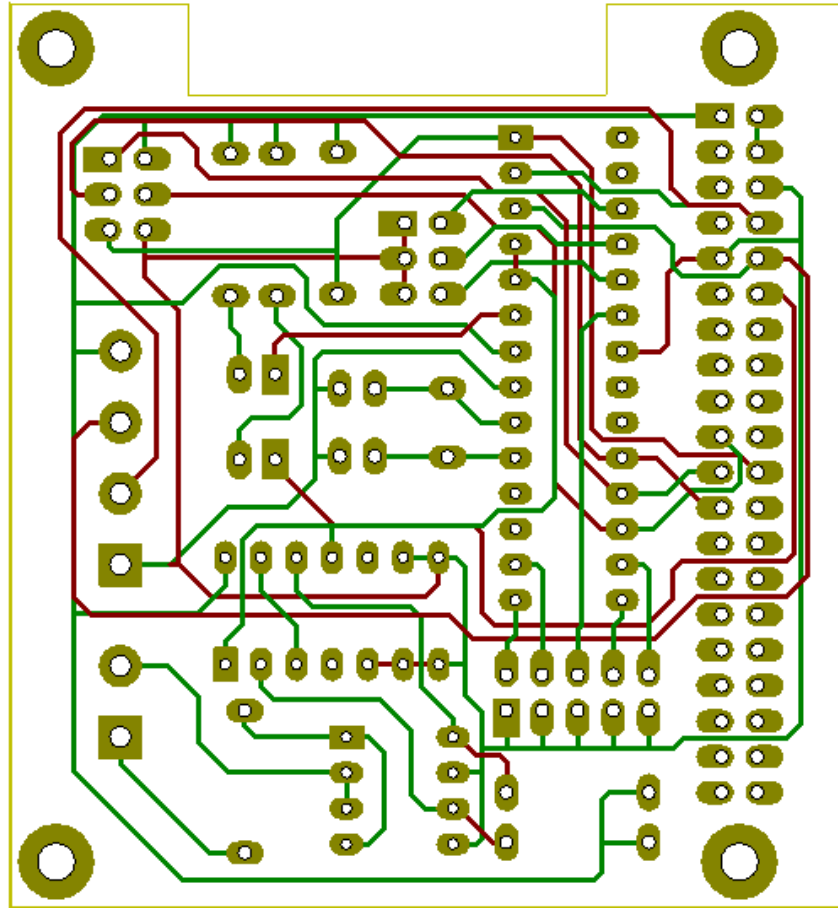


Figure: Structure of components

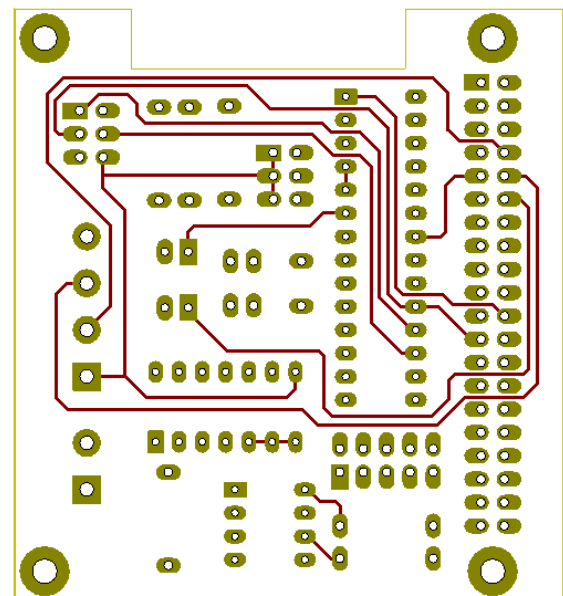


Photo: assembled circuit board on the RaspberryPi with connected ISP programmer

*Board layout*



Picture: Board layout back side



Picture: Board layout front

### Components:

Components	Value	Function	Info
C1,C2	22pF	Capacitor	<a href="#">see also quartz and C1 / C2</a>
D1,D2	LED	LowCurrent-LED	
<b>R1*</b>	1k (1000 Ω)	Resistance see below	Adaptation to input voltage
R2,R3,R4	1k (1000 Ω)	Resistance	
R5,R6	R100 (100 Ω)	Resistance	
U1	PC827	Optocoupler	Alternative: 2 pcs. PC817
U2	ATMega328	Atmel µController	(Arduino - Compatible)
U3	CMOS4093	SchmittTrigger (NAND gate)	
Q1	10MHz	Quartz (HC49U)	Q-load capacity: 30pF
J1	2-Pol Schraubklemme	Terminal	Solderable screw terminal 5mm
J2	Wannenstecker 6-Pol	ISP port for ISP programmers	WSL 6G straight (2X3-pin)
J3 *	Buchsenleiste 2X12 Pol	RaspberrPi female connector strip	RaspPi Connector Pin 1-24
(U1)	IC-Sockel 8-polig	IC Socket for U1	Recommended accessories for U1
(U2)	IC-Sockel 28-polig	IC Socket for U2	Recommended accessories for U2
(U3)	IC-Sockel 14-polig	IC Socket for U3	Recommended accessories for U3
<b>Trafo</b>	7 – 15 V AC ~	Power supply unit / transformer	
<b>J4</b>	<a href="#">4-Pol Schraubklemme</a>	Terminal (2X2 screw terminal)	When using USB-TTL
J5	Stiftleiste 2X3-Pol	Jumper	Universal input extension
<b>J6</b>	<a href="#">Stiftleiste 2X5Pol</a>	Function	Universal output extension

(\*) in the case of the application for RaspberryPi

Also

- RaspberryPi 3 or 4 or USB-TTL converter
- Spacer for the mains frequency board: M2.5 X 12mm, female-female + screws 2.5mm X 5
- Housing

\*) Recommendation / calculation Input resistance R1

Input voltage (AC)	Recommended size: R <sub>1</sub>
5V ~ - 7V ~	470 Ω
7V ~ - 15V ~	1k Ω
15V ~ - 25V ~	1,5k Ω

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

*When used with USB (without RPi)*

- USB to TTL serial UART converter with 5V (or 3.3V)



Figure: USB-TTL converter



Figure: Connection example USB-TTL converter (without RPi)

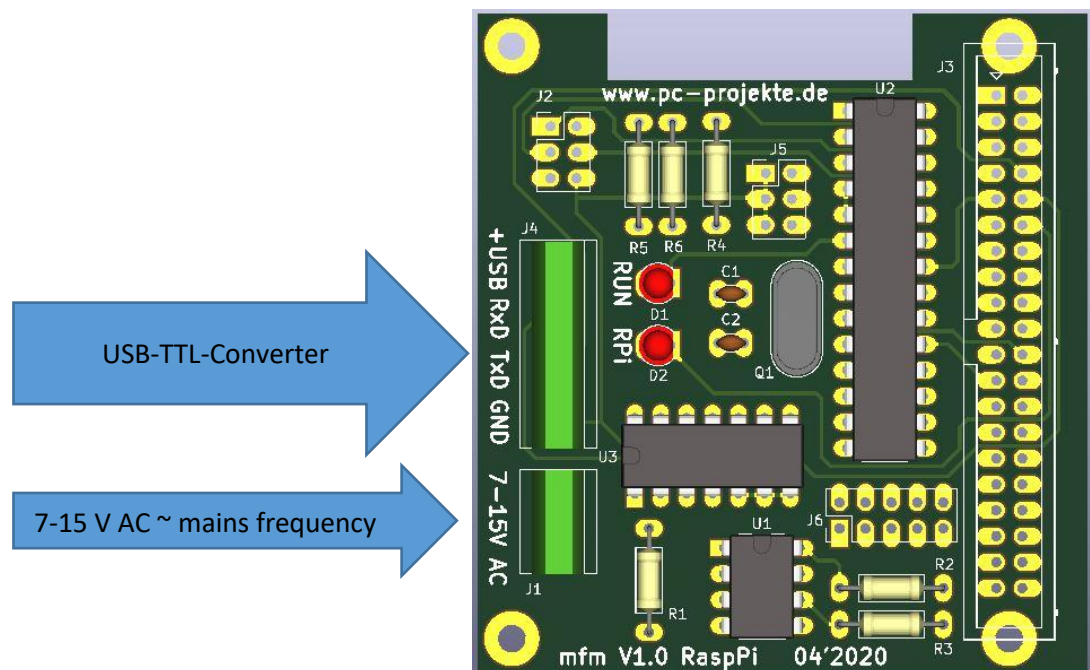


Figure: Connection diagram USB-TTL converter

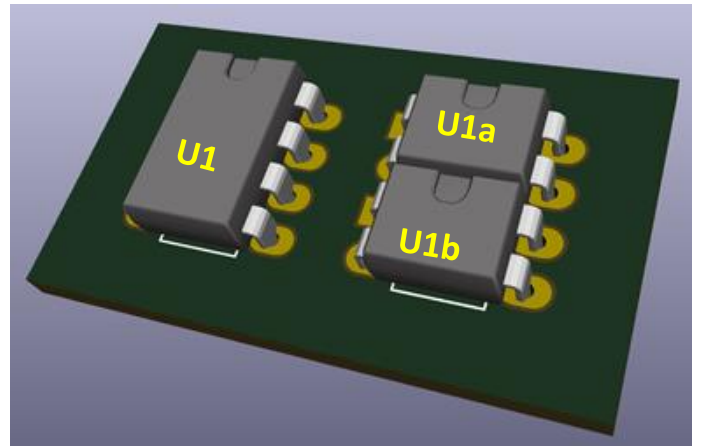


### *Installation instructions:*

#### **U1 Optocoupler**

Two PC817 (U1a, U1b) can be used instead of one PC827 (U1).

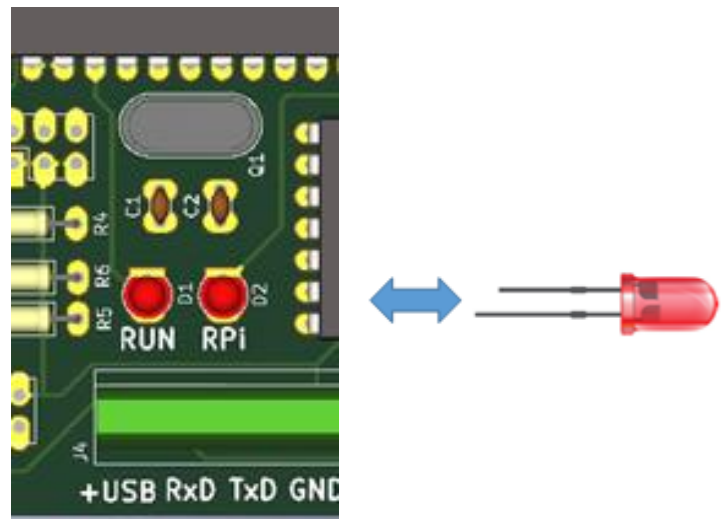
In this case the assembly must be done as shown in the following figure:



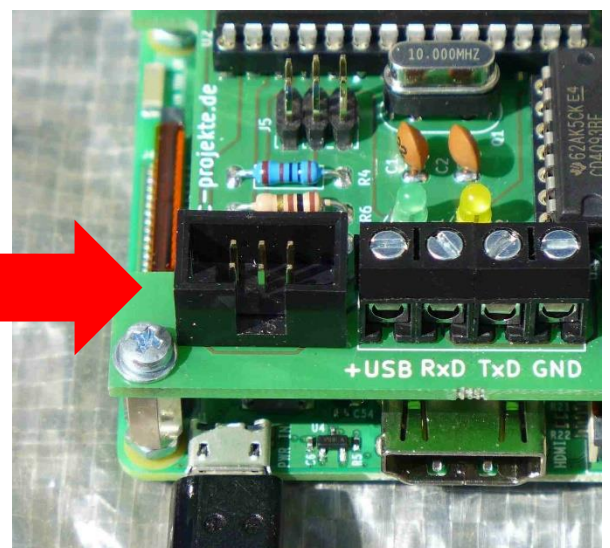
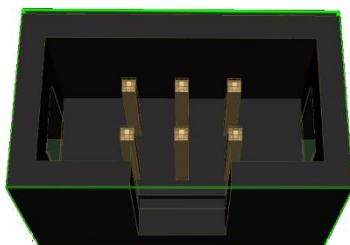
Picture: alternative assembly of 2 pcs. PC817 instead of 1 pc. PC827

#### **Mounting position LED 1 and LED 2**

The assembly/mounting position of the LEDs must be as shown in the picture on the right.

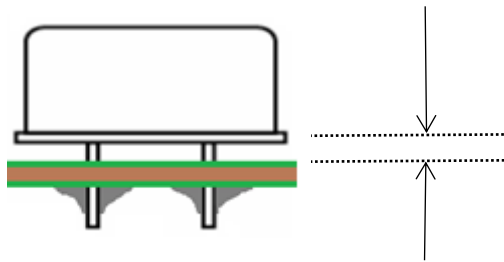


#### **Mounting position J2**



## Information about the quartz

### Distance of the quartz to the board



Always keep a distance from the board during assembly to avoid short circuits

Picture: Quartz distance to the board

### Component information: Quartz and C1 / C2

10.0000-HC49U-S, standard quartz in housing HC49/U-S.

- Frequency: 10.0000 MHz
- Cl: 32 pF                                   ==>  $C1 / C2 = 22\text{pF}$                    Note load capacitance in the data sheet!
- R<sub>smax</sub>: 50 Ohm
- Temperature coefficient:  $\pm 30$  ppm
- Frequency tolerance:  $\pm 30$  ppm

### *Housing recommendation*

The mains frequency board is prepared for use in a DIN rail housing



Picture: DIN rail housing for RaspberryPi 4 and RaspberryPi 3



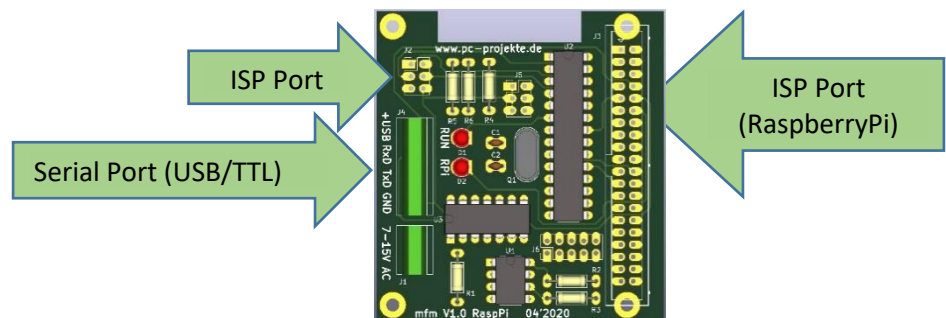
## Software / source code

The basic program is available for download in the Arduino IDE variant (C / C++) and as HEX code:

The basic version contains the simple basic program for mains frequency measurement and can be used or adapted for your own projects. The extensions will be available soon (also under GNU GLP3 licenses).

### Important note on Arduino compatibility:

The mains frequency board works with 3.3V (RaspberryPi). Because of the 3.3V only a maximum crystal of 10MHz can be used. Therefore some controller fuses must be changed. A direct Arduino programming via the serial interface is therefore not possible in this hardware version.



**A 100% compatible Arduino "mains frequency board" (with serial transmission) is in preparation.**

## *µController-Fusebits*

The "System Clock" and "Clock Options" - Fusebits of the ATmega328 must have the following settings:

<b>LOW Fuses = 0xFF</b>	<b>HIGH Fuses = 0xDA</b>	<b>EXTENDED Fuses = 0xFF</b>
-------------------------	--------------------------	------------------------------

The correct fusebits can be set and transferred with AVRdude or Atmel AVR-Studio. Without the correct setting the mains frequency program will not run with 3.3V operating voltage.

### Transfer / Programming

The transfer of the source code can be done in several ways. I describe an exemplary possibility on the next page. Further programming possibilities see project page.

## Programming Service

For those who do not want to program themselves, I offer a programming service.

### 1. installation AVRdude on the RaspberryPi

```
sudo apt-get install AVRdude
```

### 2. adjusting the settings AVRdude to required hardware pins

Open AVRdude configuration:

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

Search line:

```
#  
  
# PROGRAMMER DEFINITION  
  
#
```

Insert the following lines below:

```
#-----  
  
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;  
;
```

Save the changed settings

### 3. test AVRdude / read out the fuses

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

### 4. set the required fuses with the following command lines:

```
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. transfer the current HEX program

(updated with Arduino-IDE or as download from the project page)

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

you will find further transmission possibilities shortly on the project page...

[Measurement / Output test](#)

*Check the function with a terminal program*

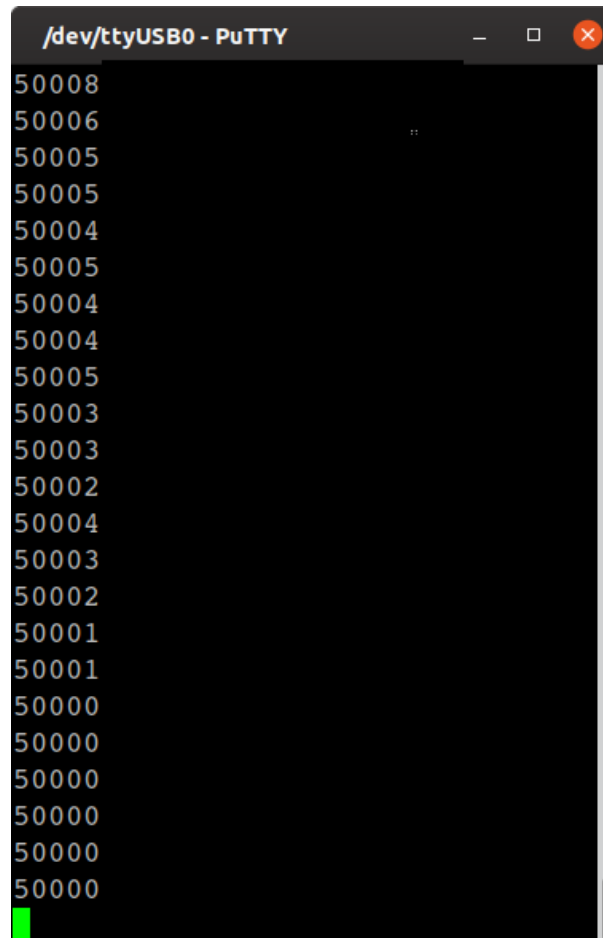
Open a terminal program e.g. PuTTY

Check the settings of the serial interface

Serial Line: /dev/ttyS0 (or /dev/ttyAMA0)

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

After opening, the terminal program shows the data flow of the mains frequency measurement.



The screenshot shows a PuTTY terminal window titled "/dev/ttyUSB0 - PuTTY". The window has a black background with white text. The text 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, 50000. A green cursor is visible at the bottom left of the terminal window.

### Communication of the mains frequency board

The console program ModbusServerTool enables the recording and provision of network frequency measurement data for control system connection, e.g. OPC server.

The console program runs on Windows and Linux (RaspberryPi).

The data can be easily visualized on the web server of the ioBrowser, the OpenHAB (Open Home Automation Bus) or the OpenMUC (web interface for system configuration and visualization).

Details about the project extension can be found on the website [www.netzfrequenzanzeige.de](http://www.netzfrequenzanzeige.de).

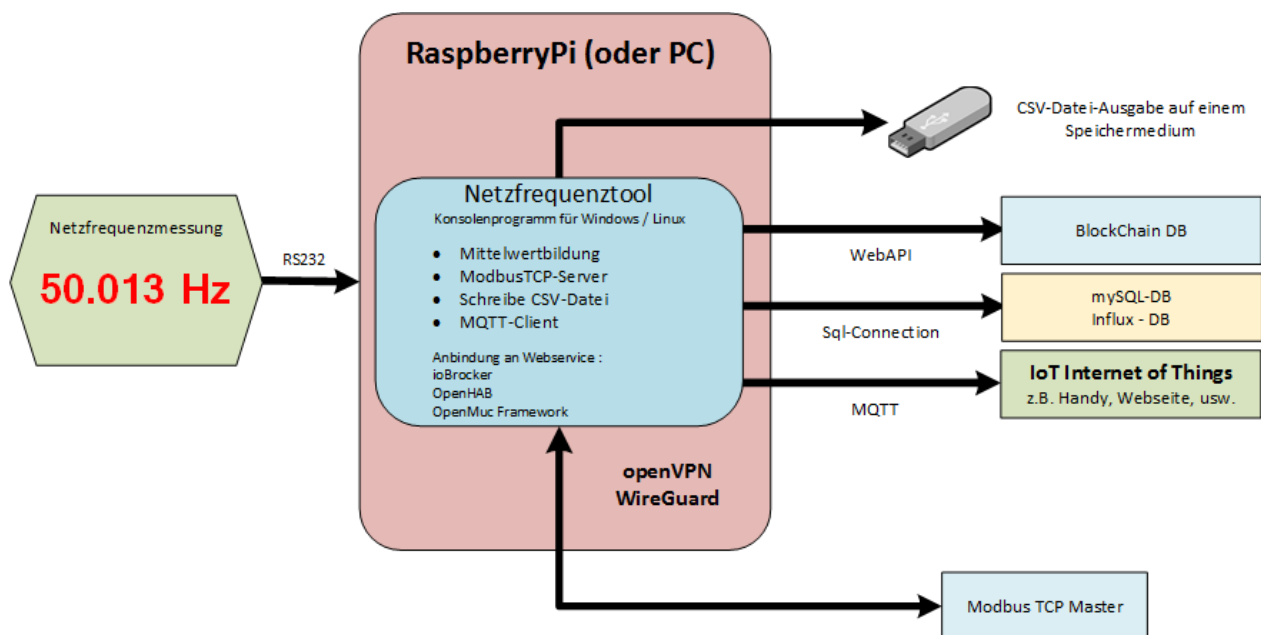


Figure: Scheme of mains frequency measurement and connection to a control system

## Information / Order possibility

- **further information at [www.netzfrequenzanzeige.de](http://www.netzfrequenzanzeige.de)**
- inquiry private sale of circuit boards and AC power supplies (remaining stock)  
[pc\\_projekte.de@arcor.de](mailto:pc_projekte.de@arcor.de)
- private programming service for  $\mu$ Controller
- Link for current parts list (at Reichelt) <https://www.reichelt.de/my/1689769>

## Notice / exclusion of warranty

Project note: The entire project is a private, non-commercial project.

The author does not take over any guarantee for functionality of the presented hard- and software. Liability claims against the author, which refer to damages of material or idealistic kind, are in principle impossible. The contents of the homepage, the project description or the software are published without any claim for correctness. All information is without guarantee. The use of the texts, graphics, pictures and programs are only allowed within the scope of the OpenSource license belonging to the project. The author expressly reserves the right to change, supplement or delete parts of the pages without prior notice.

- Use of software and hardware on your own responsibility
- Any product liability through the use of the software and hardware is excluded

Disclaimer according to §1 Abs.2 Z.3 ProdHaftG , (BGBl. I S. 1474 m.W.v. 08.09.2015 of the Federal Republic of Germany )

## Power supply information

**The power supply must correspond to the description, otherwise components may be destroyed.**

Jens Müller, 04'2020     [www.pc-projekte.de](http://www.pc-projekte.de)