Konzept eines kostengünstigen GPS-synchronisierten Spannungsphasenwinkelmesssystems für Smart-Grid-Applikationen

Beitrag zur Konferenz Zukünftige Stromnetze 2023, Berlin
Christian Hotz, Tim Schäfer, Ashraf Ishag, Ahmed Omer, Sergej Baum, Ingo Stadler, Eberhard Waffenschmidt
TH-Köln, 16.12.2022

Kurzfassung

In dieser Arbeit wird ein PMU-System vorgestellt, das auf Basis von Standardkomponenten (einem handelsüblichen GPS-Modul, einem Einplatinen-PC wie dem Raspberry Pi) sowie einer eigens entworfenen Stoppuhr-Platine basiert. Ziel ist es, Spannungsphasenwinkelmessungen auf der Niederspannungsebene zu ermöglichen, um Smart-Grid-Applikationen, etwa im Bereich privater Elektromobilität, zu unterstützen.

1. Übersicht

Eine zunehmende messtechnische Durchdringung elektrischer Netze, auch im Niederspannungsbereich, ist notwendige Bedingung zur Implementierung von Smart-Grid-Applikationen. Herausforderungen wie der Elektrifizierung des Personenverkehrs und von Gebäudeheizung sowie dem Zubau privater Energieerzeugungsanlagen kann so effektiv begegnet werden. Insbesondere die Berechnung von Knoten- und Leitungsleistungen gemäß Gl. 1, 2 erfordert Kenntnis der komplexen Knotenspannungen im Netz, also auch der Spannungsphasenwinkel. [1]

$$\underline{S_i} = \underline{U_i} \cdot \sum_{j=1}^{n} \underline{y_{ij}^*}\,\underline{U_j^*} \qquad (1.1)$$

$$\underline{S_{ij}^*} = \underline{U_i^*}\left(\underline{U_i} - \underline{U_j}\right)\underline{Y_{ij}} + U_i^2\,\underline{Y_{i0}} \qquad (1.2)$$

Da die Messung von Spannungsphasenwinkeln synchronisierte Messungen an räumlich verteilten Standorten erfordert, ist sie aufwendig und damit im Allgemeinen teuer. Die Kosten für professionelle Spannungsphasenwinkelmessgeräte (engl.: Phasor Measurment Units, PMUs), liegen laut [2] im günstigsten Fall bei 35000 €.
Dieses Paper beschreibt den Aufbau einer kostengünstigen PMU in Hard- und Software mit Standard-Hardware-Komponenten und einer PCB-Schaltung.

2. Stand der Technik

Im Forschungsumfeld sind verschiedene Ansätze zu dem Thema zu finden. In [3] wird das Projekt OpenPMU vorgestellt. OpenPMU ist eine Plattform für PMU-Entwicklung. Es werden keine Details zur Hardware-Entwicklung angegeben, nur ein Preis von ca. 940 €. Paper [4] stellt einen Ansatz vor, der, wie der hier präsentierte, mit einem Raspberry Pi arbeitet. Leider werden nur vorläufige, wenig aussagekräftige Ergebnisse vorgestellt. [5] stellt einen Ansatz vor, dessen Ergebnisse durchaus brauchbar wirken. Eine detaillierte Analyse der Performance bleibt aus. Die Kosten belaufen sich auf 330 € pro Messsystem. Ziel ist es hier, diesen Kostenpunkt noch deutlich zu unterbieten.

Um den gesamten Entwurfsprozess abzudecken, werden folgende Unterpunkte beschrieben:

- die entwickelte und verwendete Hardware
- die Datenverarbeitung an einem Knoten
- die Kommunikation der Messknoten untereinander
- die Zusammenführung und Filterung von Messreihen
- eine Analyse der Performance des Systems

Es wird außerdem eine Abschätzung der Systemkosten vorgenommen.

3. Allgemeine Beschreibung

Folgende wesentliche Hard- und Softwarekomponenten wurden im Rahmen des Projekts entwickelt und werden im Folgenden beschrieben:

- Hardware:
  - Eine Komparatorbasierte Phasennulldurchgangsmessplatine
  - Eine Zeit-Stopp-Platine basierend auf Counter-, Register- und Quarz-Chips
- Softwaremodule für:
  - die Zeitsynchronisierung über GPS
  - das Auslesen der Stoppuhrregister
  - die Zusammenführung der Daten verschiedener Module auf einem Server
  - die Verrechnung der Daten aller Messmodule zu Spannungsphasenwinkeln inkl. geeigneter Filtermethoden, Lastsprungdetektion und Netzfrequenzmessung.

Um den Spannunsphasenwinkel zwischen zwei Messknoten zu bestimmen, muss der Gangunterschied zwischen den Sinusverläufen der Spannungen bestimmt werden. Wegen der räumlichen Distanz (es können sich beispielsweise zwei Messknoten in je einem Haushalt eines Straßenzuges befinden) kann in aller Regel keine Messung beider Spannungsverläufe in einem galvanisch zusammenhängenden Messsystem vereint werden. Stattdessen muss je eine galvanisch unabhängige Messeinheit pro Messknoten mit einem an allen Knoten zeitlich synchronen Referenzsignal verglichen werden.
Nach Formeln 3.1 und 3.2 lassen sich Spannungsphasenwinkeldifferenzen $\Delta\delta$ aus lokal gemessenen Zeitabständen Δt_1 und Δt_2 zwischen einem Referenzsignal und lokalen Spannungsverläufen berechnen:

$$\Delta t_{1,2} = \Delta t_2 - \Delta t_1 \tag{3.1}$$

$$\Delta\delta = \Delta t_{1,2} \cdot \omega_{Netz} \tag{3.2}$$

Das Funktionsprinzip ist in Abbildung 1 schematisch dargestellt.

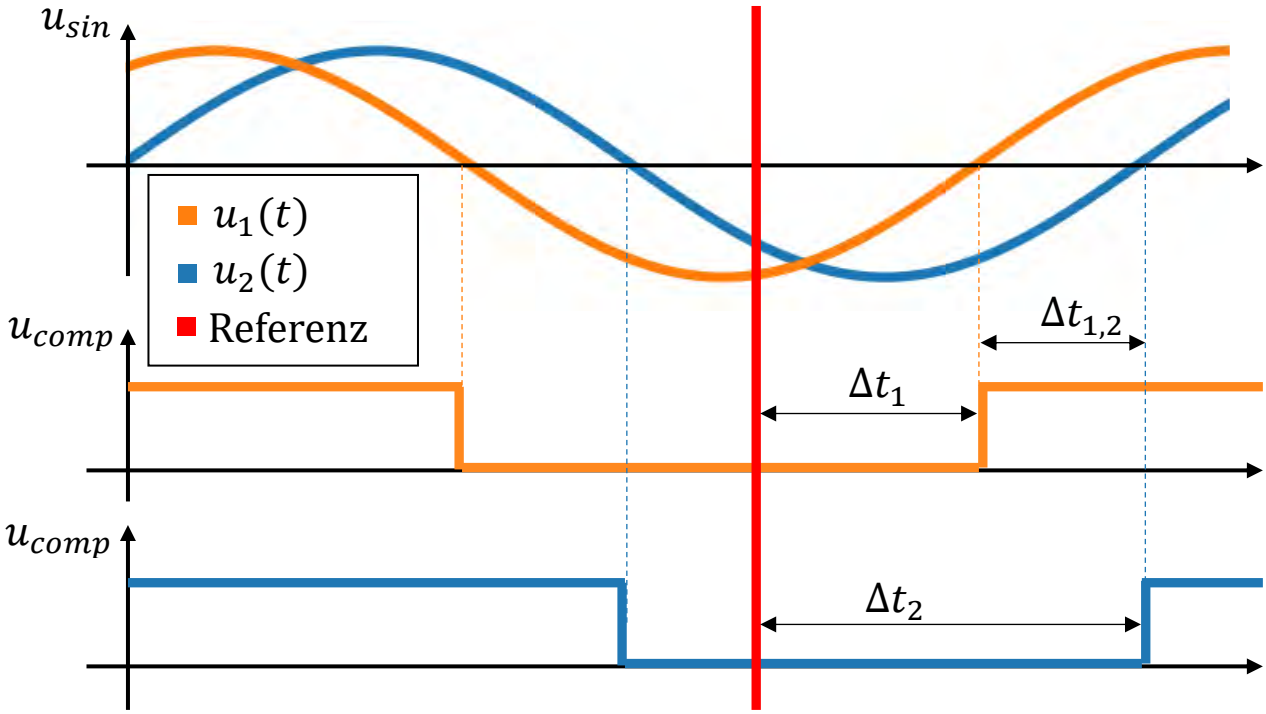


Abbildung 1: Schematische Darstellung der Signale im Messsystem

3.1. Anforderungen an die zeitliche Auflösung

Um praktisch nutzbare Messungen zu erreichen, ist eine Präzision von deutlich unter 1° Phasenwinkel notwendig. 1° Messtoleranz entspricht einer zeitlichen Toleranz nach Gleichung 3.3:

$$1° \cdot \frac{1}{50\ Hz \cdot 360\ °} = 56\ \mu s \qquad (3.3)$$

Da sich Rauschen und Latenzen einzelner Systemkomponenten kumulieren, wird hier eine zeitliche Toleranz jeder einzelnen kritischen Messkomponente von nicht mehr als 5 µs gefordert. Zur Messung der lokalen Zeitwerte Δt_1 und Δt_2 wird eine Quarzgesteuerte Stoppuhr entworfen, die Zeitabstände zwischen Signalflanken an zwei verschiedenen Anschlüssen misst (vgl. Δt_1 und Δt_2 in Abbildung 1) und über eine serielle Schnittstelle ausgibt. Für das räumlich verteilte Referenzsignal ist eine Synchronisation beispielsweise per TCP/IP deutlich zu unpräzise, ebenso ist eine direkte elektrische Signalverbindung in den meisten Fällen unpraktikabel. Verwendet wird daher ein Pulse-per-Second oder PPS-Signal, das Teil des GPS-Standards ist. Laut [6] weist PPS einen Jitter von max. 50 ns auf. Das kostengünstige GPS-Modul GNSS 5 Click bietet sowohl eine PPS- als auch eine uart-Schnittstelle. Letztere ermöglicht das Auslesen der GPS-Datetime (Datum und Uhrzeit), die der Synchronisation lokal aufgenommener Daten an zentraler Stelle dient. Das PPS-Signal wird an allen Messknoten mit dem Phasennulldurchgang (bzw. u_{comp}) verglichen.

4. Lokales Messverfahren

Zunächst wird erläutert, wie die Komponenten eines lokalen Messpunktes auf Hard- und Software-Ebene funktionieren. (Vgl. Abb. 2)

4.1. Übersicht der Hardware an einem Messknoten

Die einphasige Knotenspannung (230 V) ist an die Komparatorplatine angeschlossen, die daraus das 5-V-Rechtecksignal u_{comp} erzeugt. Dieses Signal wird an die Stoppuhr und den Raspberry Pi gesendet. Die Stoppuhr, die mit jedem PPS-Signal resettet wird, hält in jeder Netzperiode den Messpunkt (Zeit zwischen PPS- und Komparatorflanke) vor und der Raspberry Pi liest ihn aus. Die Systemzeit des Raspberry Pi wird per uart und PPS µs-genau eingestellt. In einem letzten Schritt bündelt der Raspberry Pi einen Sekundenmessvektor, versieht ihn mit einem Zeitstempel und überträgt ihn über eine Internet- oder LAN-Schnittstelle (z.B. PLC) an einen zentralen Server.

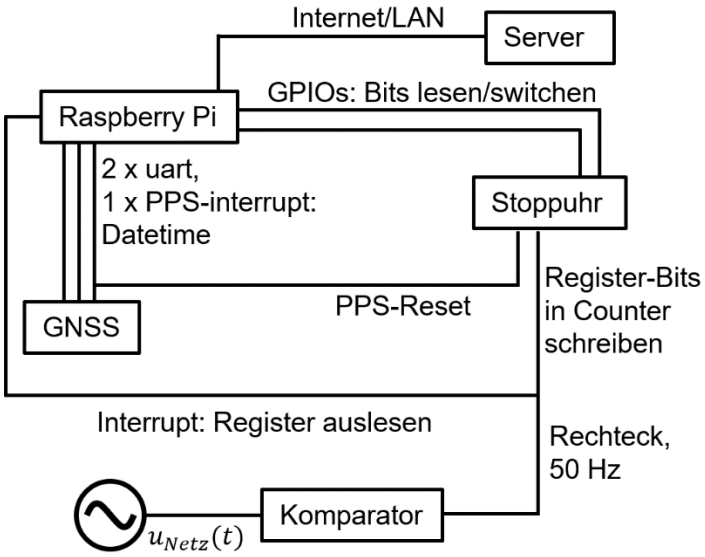


Abbildung 2: Signalflussschema des Messsystems

4.2. Komparatorschaltung

Die Schaltung zur Transformation von u_{sin} in u_{comp} (vgl. Abbildung 1) besteht aus einem Spannungsteiler ($5{,}6\ \mathrm{k\Omega}$ bzw. $330\ \mathrm{k\Omega}$), einem Komparator und einem Optokoppler zum Schutz der nachgeschalteten Komponenten. Das Rechtecksignal am Ausgang hat eine Amplitude von 5V und es gilt Gl. 4.1:

$$u_{comp} = \begin{cases} 5\ V, wenn\ u_{sin} > 0 \\ 0\ V, wenn\ u_{sin} \leq 0 \end{cases} \qquad (4.1)$$

Ein Prototyp der Stoppuhr-Platine ist in Abbildung 3 zu sehen.

Abbildung 3: Prototyp der Komparatorschaltung

4.3. Stoppuhr

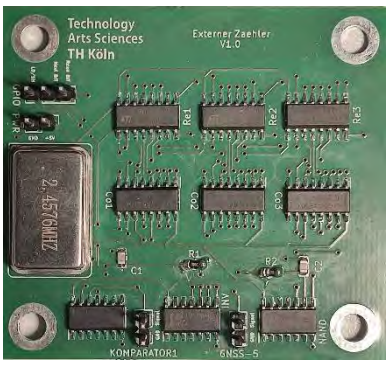


Abbildung 4: Prototyp der Stoppuhr-Platine

Die Stoppuhr-Platine zum Erfassen der Zeitabstände $\Delta t_1, \Delta t_2$ zwischen PPS- und Komparatorflanke besteht aus Countern, Schieberegistern und einem hochpräzisen Quarz. Die steigende Flanke des PPS-Signals wird mittels einer Schaltlogik aus PT1-Glied und NAND-Gatter in einen schmalen Puls (<1 µs) umgewandelt, der die Counter resettet. Da der Puls an allen Messknoten die gleiche Breite hat, subtrahiert die Verzögerung sich später aus dem Gesamtergebnis heraus und beeinflusst die Präzision des Ergebnisses kaum. Nach jedem Reset des Counters durch das PPS-Signal zählt der Counter mit der Taktfrequenz des Quarzes (f_{Quarz} =2,4576 MHz, also 407-ns-Schritte) hoch. Um die Sekunde zwischen zwei Resets durch das PPS-Signal numerisch abdecken zu können, werden drei Counter je 8 Bit zusammengeschaltet (siehe Formel 4.2).

$$2^{24} \cdot \frac{1}{2{,}4576\ MHz} = 6{,}827\ s > T_{PPS} = 1\ s \qquad (4.2)$$

Der 24-Bit-Vektor entspricht dann einer natürlichen Zahl, die entsprechend f_{Quarz} bis $2{,}4576 \cdot 10^6$ zählt und durch Division durch f_{Quarz} wieder zum Zeitwert der Stoppuhr umgerechnet wird.
Ein Prototyp der Platine ist in Abbildung 4 dargestellt. Der Quarz (links), die drei Schieberegister (oben), die drei Counter (in der Mitte) und NAND-Gatter inkl. PT1-Glieder (unten) sind seine wesentlichen Komponenten.

4.4. Raspberry Pi

Die in den Registern der Stoppuhr vorgehaltenen Werte werden durch den Raspberry Pi seriell über dessen GPIOs eingelesen: ein Output-Pin schaltet die 24 Register durch, ein Input-Pin liest sie ein. Diese Routine wird über einen Hardware-Interrupt ausgelöst, der auf Flanken des Komparators reagiert. Die fünfzig sekündlich aufgenommenen Werte werden in einem Array zusammengefasst und mit einem Zeitstempel versehen. Der gesamte Algorithmus ist in Python implementiert und in Tabelle 1 schematisiert.

Tabelle 1: In der Raspberry-Pi-Firmware implementierter Algorithmus

Zu Beginn: einmalig System-Datetime der RPis mit GPS synchronisieren
Jede Sekunde:
Counter-Reset durch PPS-Flanke
50-mal pro Sekunde (bei jeder Komparatorflanke):
Daten von Counter an Register senden **Interrupt-Routine RPi:**
Register auslesen:
Für die 24 Register-Bits:
Bit an Input-GPIO lesen **Output-GPIO: Register-Bit weiterschalten**
Bit-Folge in Zeitwert umwandeln **Zeitwert in Array loggen** **Counter-Wert auf PPS-Reset überprüfen**
Falls Counter-Reset detektiert wurde:
Zeitstempel erstellen **Messwerte mit Zeitstempel an Server übertragen** **„Volle Minute“ überprüfen**
Falls „volle Minute“ detektiert wurde:
Datetime mit GPS synchronisieren

Tabelle 2 zeigt einen beispielhaften, so erzeugten Sekundendatensatz:

Tabelle 2: An den Server übertragener Datensatz mit Zeitstempel und Nulldurchgangsvektor

Typ:	**Nulldurchgangsvektor** (von RPi an Server)
Zeitstempel:	1670961402
Daten [ms]:	[19.4059, 39.4027, 59.399, 79.3929, 99.3876, 119.384, 139.3807, 159.3801…, 999.2712]

Der Unix-Zeitstempel („epoch-time“) entspricht dem 13.12.2022, 19:56:42, GMT bzw. den Sekunden, die seit dem 01.01.1970, 00:00 vergangen sind. Die Zahlenwerte im Data-Array sind die gemessenen Zeiten zwischen PPS- und Komparatorflanken in ms (Δt_1 oder Δt_2 in Abbildung 1). Die Inkrementierung mit Netzfrequenz ist an den Zeitdifferenzen von ca. 20 ms zwischen den Datenpunkten zu erkennen. Da sie je etwas weniger als 20 ms beträgt, lässt sich auf eine Netzfrequenz $f_{Netz} > 50\ Hz$ schließen. Das Versenden dieses Datensatzes konkludiert den lokal auf dem Raspberry Pi durchgeführten Teil des Algorithmus.

4.5. Erläuterung: zeitkritische Prozesse

Wichtig ist die Unterscheidung von zeitkritischen und weniger zeitkritischen Prozessen. Die GPS-, Komparator- und Stoppuhr-Systeme müssen, wie in 3.1 beschrieben, Präzisionen im µs-Bereich aufweisen. Das Einlesen, Verarbeiten und Übertragen von Daten des Raspberry Pi geschieht immer zwischen zwei Komparatorflanken und hat damit mit 20 ms deutlich mehr Zeit.
Vor Umsetzung der Stoppuhr-Schaltung ist eine Zeitmessung per Hardware-Interrupt des Raspberry Pi untersucht worden. Die Präzision genügt allerdings nicht, um zufriedenstellende Messergebnisse zu erzeugen. Eine umfassende Untersuchung zur zeitlichen Präzision von Raspberry-Pi-Hardware-Interrupts gibt es in [7].

5. Mehrpunktmessungen

Im Folgenden wird beschrieben, wie die am Server gesammelten, sekündlichen Datensätze weiterverarbeitet werden. Dies umfasst das Filtern fehlerhafter Werte, die Verrechnung zu Spannungsphasenwinkeln und verschiedene Filterungsverfahren zur Glättung des Messrauschens.

5.1. Fehleranalyse

Das Verfahren analysiert Datensätze wie den aus Tabelle 2 auf Fehler und Rauschen und eignet sich um

- Fehler durch den Quarz,
- Bit-Fehler in Counter und Register,
- Bit-Fehler bei der Datenspeicherung und -übertragung zwischen Counter, Register, Raspberry Pi und Server,
- und Fehlerhafte Detektion des Spannungsnulldurchgangs

zu identifizieren.
Nicht detektiert werden können Fehler, die auf ein fehlerhaftes PPS-Signal oder fehlgeschlagene Synchronisierung der Systemzeit zurückzuführen sind. Solche Fehler sind innerhalb einer Sekunden-Messreihe nicht erkennbar. Sie können später bei der Zusammenführung der Messreihen aller Messknoten identifiziert und gegebenenfalls korrigiert werden.

5.1.1. Reduktion des Messrauschens und Elimination von Übertragungsfehlern

Ein Analyse- und Filteransatz, der sich als robust und effektiv herausgestellt hat, basiert auf der Erzeugung eines Periodenvektors: er enthält die Differenzen von jedem Messpunkt des Nulldurchgangsvektors zum Nächsten und hat somit 49 Einträge. Aus dem Datensatz wie er in Tabelle 2 abgebildet ist, ergibt sich folgender Periodenvektor (Tabelle 3):

Tabelle 3: Periodenvektor zur Veranschaulichung des Filterprinzips

Typ:	**Periodenvektor** (Differenzen zwischen Nulldurchgangsvektor-Elementen)
Zeitstempel:	1670961402
Daten [ms]:	[19.9967, 19.9963, 19.9939, 19.9947, 19.9963, 19.9967, 19.9931, 19.9939, 19.9967, 19.998, 19.9919, 19.9939, 19.9976, 19.9963, 19.9951, 19.9931 , …]

Der Median des Periodenvektors T_{med} wird nun als Periodendauer der Messreihe angenommen. Im Gegensatz zum arithmetischen Mittel ist der Median sehr robust gegen Rauschen und Datenübertragungfehler. Hier ist der Median 19,9951 ms.
In einem zweiten Schritt werden die Differenzen zwischen den Einträgen des Periodenvektors und T_{med} gebildet – man erhält einen „Rauschvektor". Das Rauschen setzt sich aus Netzfrequenzschwankungen, Messrauschen und möglichen Datenübertragungsfehlern zusammen. Der Rauschvektor ist in Tabelle 4 abgebildet (Achtung: die Einheit ist hier µs!):

Tabelle 4: Rauschvektor eines Nulldurchgangsvektors

Typ:	**Rauschvektor** (Periodenvektor abzüglich Periodendauer T_{med})
Zeitstempel:	1670961402
Daten [µs]:	[…, 2.8483, -3.2552, -1.2207, 2.4414, 1.2207, 0.0, -2.0345, 1.2207, **104.9805**, -3.6621, **5.069**, 1.6276, -3.2552, 0.4069, 2.0345, 2.0345, -2.4414, …]

Im Rauschvektor ist schon die zeitliche Quantelung der Zeitwerte mit den 407 ns Periodendauer des Quarz-Chips zu erkennen. Die maximale Rauschamplitude beträgt (bis auf einen „Ausreißer" von über 100 µs) etwas über 5 µs.
Es wird ein Filter implementiert: Alle Werte aus dem Nulldurchgangsvektor, die mit einem Periodenrauschen von mehr als 5 µs behaftet sind, werden eliminiert.
In einer 10-minütigen Demo-Messreihe waren 11,7 % aller Nulldurchgangswerte der Messreihe betroffen, es bleiben also im Schnitt 44,2 Nulldurchgangswerte pro Sekundenvektor übrig.
Bezüglich des „Ausreißers" folgt eine weitere Analyse.

5.1.2. Analyse signifikanter Ausreißer

Das beschriebene Systemrauschen bis zu einer Amplitude von 5 µs wird als „normal" betrachtet. Diejenigen Nulldurchgangsvektoren, die in mindestens einem Messpunkt ein Rauschen aufweisen, das 20 µs übersteigt, können in drei Kategorien eingeteilt werden, die in den 5.1.2.1 bis 5.1.2.3 erläutert werden.

5.1.2.1. Wert aus vorangegangener Periode eingelesen

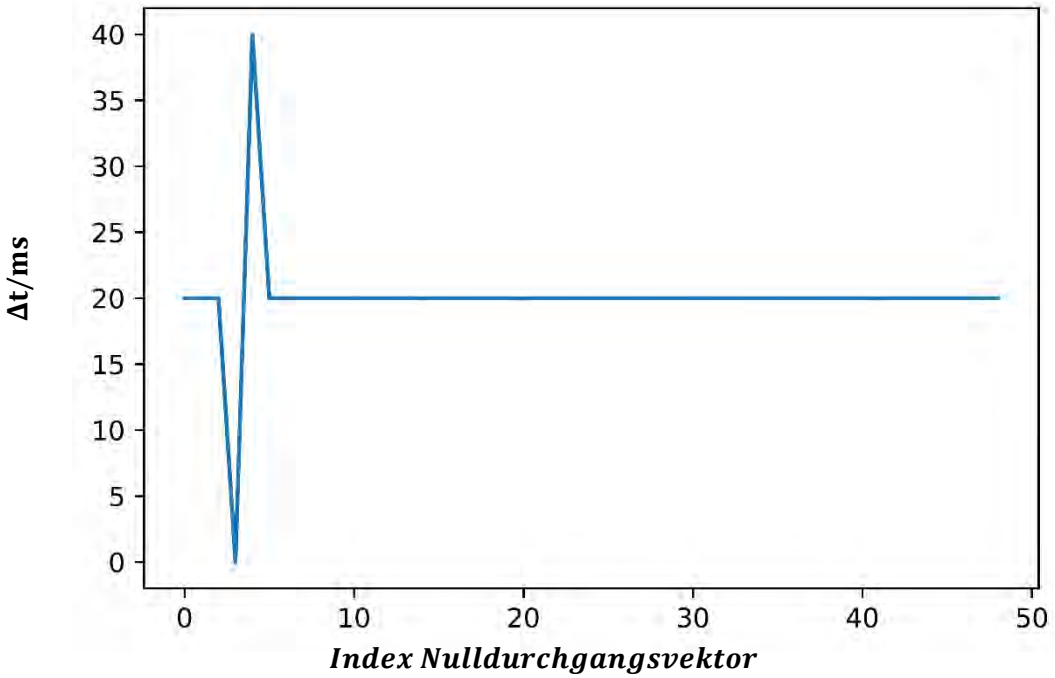
In einigen Fällen kommt es vor, dass ein Nulldurchgangszeitwert doppelt eingelesen wird und der darauffolgende Wert in einem Abstand von ca. 40 ms liegt (in Tabelle 5 ist ein solcher Fall abgebildet):

Tabelle 5: zwei identische Zeitstempel in Sekundenarray

Typ:	**Nulldurchgangsvektor** (von RPi an Server)
Zeitstempel:	1670961024
Daten [ms]:	[9.6745, 29.6623, 49.6554, **69.6444, 69.6444, 109.6269**, 129.6163, 149.6048, …]

Da zum einen der Rasperry Pi jeweils in korrekter Taktung Werte aufgezeichnet hat und zum anderen alle Werte vor und nach der „Störstelle" einer Geraden mit einer Steigung von ca. 20 ms folgen, wird davon ausgegangen, dass die Übertragung der Werte vom Counter an den Register in diesem Fall nicht ordnungsgemäß funktioniert hat. In der Demo-Messreihe (600 Werte) ist dieser Fehler in drei Nulldurchgangsvektoren aufgetreten. Abbildung 5 zeigt einen Plot des Periodenvektors für einen solchen Fall: ein Wert der Messreihe beträgt 0 ms, der darauffolgende Wert ca. 40 ms, der Rest aller Werte ist im Bereich 20 ms angesiedelt.

Abbildung 5: Periodenvektor mit einem Nulldurchgangswert aus der Vorperiode



5.1.2.2. Einzelne fehlerhafte Werte

In anderen Fällen weichen einzelne Perioden von der Netzperiodendauer ab, in der nächsten Messung tritt die Differenz dann negativ auf. Es ist einfach eine Zeitmessung korrumpiert.
Der Wert der Verschiebung fällt dabei sehr verschieden aus. In 9 von 600 Fällen liegt er im Bereich zwischen 20 und 250 µs und lässt auf einen Fehlerhaften Zeitpunkt der Komparatorflanke schließen, welcher wiederum durch Netzrauschen bedingt sein könnte.
In 2 der 600 Fälle ist der Verschiebungswert wesentlich größer, es wird ein Bit-Fehler bei der Datenübertragung vermutet.
Dieser Fehler ist dem in 5.1.2.1. beschriebenen Phänomen verwandt, die „Ausreißer“ im Periodenvektor haben aber nicht die Werte 0 ms und 40 ms, sondern können beliebige Werte annehmen, die um die Netzperiodendauer von ca. 20 ms gespiegelt sind (im Beispiel in Tabelle 6: 19,764 ms bzw. 20,2494 ms).

Tabelle 6: Periodenvektor eines Nulldurchgangsvektors mit korrumpiertem Wert

Typ:	**Periodenvektor** (Differenzen zwischen Nulldurchgangsvektor-Elementen)
Zeitstempel:	1670961201
Daten [ms]:	[…, 20.002, 20.0057, 20.0024, 20.0008, 20.0061, 19.9967, **19.764, 20.2494**, 20.0016, 20.0012, 20.0053, …]

5.1.2.3. Abschnittsverschiebung

Zuletzt sind Nulldurchgangsvektoren aufgefallen, bei denen zu einem bestimmten Zeitpunkt ein Offset gegenüber der durch alle vorherigen Werte beschriebenen Trajektorie zu verzeichnen war. Tabelle 7 stellt einen solchen Fall dar.

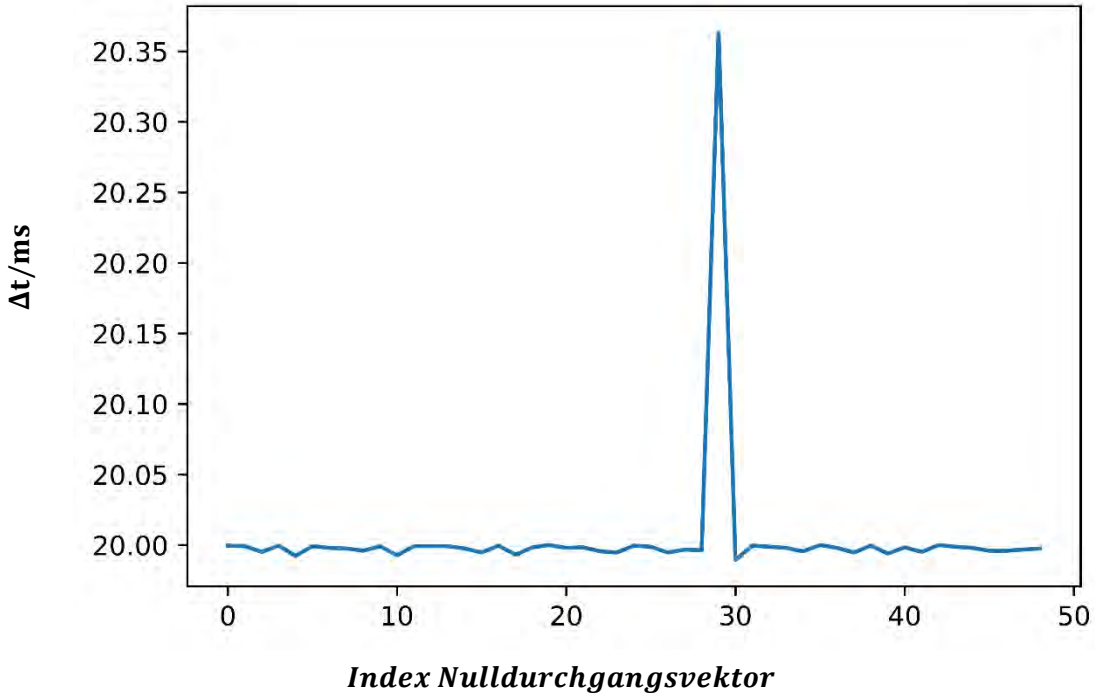
Tabelle 7: Nulldurchgangsvektor mit Abschnitts-Offset

Typ:	**Nulldurchgangsvektor** (von RPi an Server)
Zeitstempel:	1670961317
Daten [ms]:	[…, 499.9992, 519.998, 539.9927, 559.9894, 579.9858, 600.3491, 620.3385, 640.3381, 660.3369, 680.3349, …]

Zwischen den Messpunkten 579 ms und 600 ms ist zu beobachten, dass sich ein Offset in der Größenordnung von 350 µs einstellt. Da die Werte auf beiden Seiten des Zeitpunktes, zu dem der

Offset sich einstellt, präzise einer Trajektorie folgen, wird ausgeschlossen, dass es sich um einen Messfehler handelt und das Phänomen auf einen Phasensprung der gemessenen Netzspannung zurückgeführt. Es handelt sich um einen Lastsprung. Abbildung 6 zeigt den Periodenvektor einer solchen Sekundenmessreihe. Im Gegensatz zu den in Abschnitten 5.1.2.1. und 5.1.2.2. vorgestellten Anomalitäten hat dieser Periodenvektor nur einen abweichenden Wert und zwar zum Zeitpunkt des Lastsprungs.

Abbildung 6: Periodenvektor mit Lastsprung



Gemäß Formeln 5.1 – 5.3 kann der Lastsprung quantifiziert werden.

$$\delta_0 = \left((t_0 + 10\,ms)\%T_{med} - 10\,ms\right) \cdot \frac{360°}{20\,ms} \tag{5.1}$$

$$\delta_n = \left((t_n + 10\,ms)\%T_{med} - 10\,ms\right) \cdot \frac{360°}{20\,ms} \tag{5.2}$$

$$\Delta\delta = \delta_n - \delta_0 \tag{5.3}$$

Hier ist T_{med} die geschätzte Periodendauer (Median des Periodenvektors) der Netzspannung. Die Winkel δ_0 und δ_n sind theoretische Phasenwinkel der gemessenen Spannung gegenüber dem PPS-Signal. Die Messwerte t_0 bzw. t_n sind der erste und der letzte Eintrag des Nulldurchgangsvektors. Es werden vor der Modulo-Operation 10 ms addiert, die später wieder abgezogen werden, damit der Ergebniswinkel im Bereich zwischen $-180°$ und $180°$ liegt (negative Winkel werden also nicht um 360° verschoben). Die Differenz $\Delta\delta$ ist dann der geschätzte tatsächliche Winkel des Phasensprungs. Es ergibt sich für das Beispiel aus Tabelle 7 das Ergebnis in Tabelle 8:

Tabelle 8: Ergebniss der Lastsprunganalyse zur Beispielmessreihe aus Tabelle 7.

T_{med}	$= 19{,}997965\,ms$
t_0	$= 0{,}0639\,ms$
t_n	$= 980{,}293\,ms$
δ_0	$= 7{,}07°$
δ_n	$= 1{,}15°$
$\Delta\delta$	$= -5{,}92°$

[Anmerkung: Im Ergebnisteil wird sich zeigen, dass zu diesem Zeitpunkt tatsächlich ein Lastsprung stattgefunden hat, der einen Phasenwinkelsprung von -14,1° auf -20,4° bewirkt. Der hier abgeschätzte Wert $\Delta\delta$ entspricht damit in erster Näherung dem erwarteten Wert (-6,3°). Eine systematische Analyse dieses Phänomens würde den Rahmen dieser Arbeit sprengen.]

5.1.3. Regression

Als letzter Schritt zur Errechnung der Spannungsphasenwinkel zwischen mehreren Messpunkten wird ein Regressionsverfahren verwendet, das eine möglichst präzise Abschätzung von Netzfrequenz und insbesondere dem Δt zwischen PPS-Signal und Komparatorflanke (vgl. Δt_1 und Δt_2 in Abbildung 1) gewährleistet und den Umgang mit fehlenden Werten eines Nulldurchgangsvektors erleichtert. Als Beispiel wird ein unvollständiger Nulldurchgangsvektor herangezogen, der in Tabelle 9 abgebildet ist (einige Werte sind gemäß dem in 5.1.1. vorgestellten Verfahren wegen starkem Rauschen eliminiert worden).

Tabelle 9: Nulldurchgangsvektor mit fehlendem Eintrag

Typ:	**Nulldurchgangsvektor** (von RPi an Server)
Zeitstempel:	1670961003
Daten [ms]:	[36.9214, 56.9242, 76.9206, **96.9259, 136.9316**, 156.934, …]

Zur Berechnung der Regressionsgerade wird die Funktion *polyfit* aus der Python-Bibliothek *numpy* verwendet: *numpy.polyfit(x,y)* errechnet Steigung und Ordinatenabschnitt einer Regressionsgeraden, wobei x und y die Vektoren der x- und y-Koordinaten von Messpunkten sind. Setzt man für x Indizes von Messperioden und für y Nulldurchgangszeitpunkte ein, dann entspricht die Steigung der Periodendauer und der y-Achsen-Abschnitt dem gesuchten Δt.
Den Einträgen t des in Tabelle 9 abgebildeten Nulldurchgangsvektors werden gemäß Formel 5.1 die Indizes i zugeordnet, die der *polyfit*-Algorithmus benötigt:

$$i = \left\lfloor \frac{t}{t_{med}} \right\rceil \tag{5.4}$$

Nach Zuordnung der Indizes entsteht folgender Vektor, dessen Einträge Tupels aus dem Index und dem Zeitwert sind (Tabelle 10):

Tabelle 10: Nulldurchgangsvektor nach Zuordnung der Periodenidizes für den polyfit-Algorithmus

Typ:	**Nulldurchgangsvektor mit Indizes**
Zeitstempel:	1670961003
Daten [ms]:	[(1, 36.9214), (2, 56.9242), (3, 76.9206), **(4, 96.9259)**, **(6, 136.9316)**, (7, 156.934), …]

Die Ergebnisse des *polyfit* für den Nulldurchgangsvektor aus Tabelle 10 sind in Tabelle 11 abgebildet.

Tabelle 11: Ergebnisse des polyfit-Algorithmus

$$T = 20{,}0038\,ms$$
$$\Delta t = 16{,}9129\,ms$$

Aus den so errechneten Δt_i kann mit Formel 5.5 der Spannungsphasenwinkel eines Messknotens i gegenüber einem Referenzknoten r errechnet werden.

$$\delta_i = (\Delta t_r - \Delta t_i) \cdot \frac{360°}{T_{med,i}} \tag{5.5}$$

Dabei ist δ_i der Phasenwinkel am Messknoten, Δt_i bzw. Δt_r sind die *polyfit*-Werte des Mess- und des Referenzknotens. Die so errechneten Winkel werden nochmal über die Formel 5.6 in den

Bereich von -30° bis +30° projiziert – so kann sichergestellt werden, dass auch bei Anschluss der Messsysteme an verschiedene Phasen oder dem Vertauschen von Phasen- und Neutralleiter sinnvolle Ergebnisse erzielt werden.

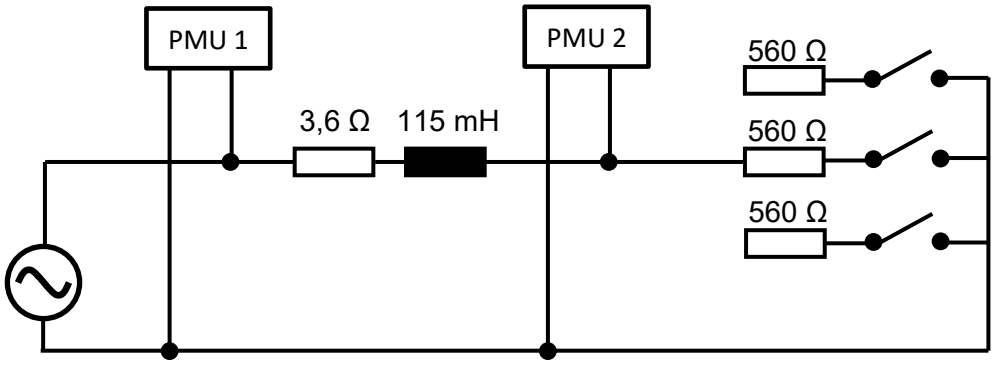
$$\delta_{i,bereinigt} = (\delta_i + 30°)\%(60°) - 30° \quad (5.6)$$

Außerdem wird so ein Winkel-Offset von 360° vermieden, wenn die Zeitdifferenz Δt am PPS-Signal „vorbeiwandert“, also von 0 ms auf 20 ms springt, weil der Nulldurchgang nun plötzlich kurz vor statt kurz nach dem PPS-Signal stattfindet. Andererseits ist die Winkelmessung damit auf $\pm$30° begrenzt, was für reale Energiesysteme aber bei Weitem ausreicht.

6. Ergebnisse

Es wurde eine Messreihe aufgezeichnet, bei der zwei Spannungsphasenwinkelmesseinheiten über ein Leitungsmodell und eine variable, automatisch schaltbare Last zusammengeschaltet wurden. Der Aufbau ergibt je nach Lastsituation einen Spannungsphasenwinkel zwischen den Messknoten. Über einen Zeitraum von zehn Minuten wurde die Lastsituation in regelmäßigen Abständen geändert und der Spannungsphasenwinkel kontinuierlich gemessen. Der Aufbau ist in Abbildung 7 dargestellt.

Abbildung 7: Messaufbau zur Realisierung dynamisch veränderlicher Spannungsphasenwinkel



Gemäß Formeln 6.1 - 6.4 kann der zu erwartende Spannungsphasenwinkel zwischen den beiden Messpunkten PMU 1 und PMU 2 bestimmt werden.
Der Eingangswiderstand des PMU beträgt 335,6 kΩ und wird bei zugeschalteten Widerständen vernachlässigt. Folgende Phasenwinkel werden erwartet (Gleichungen 6.1-6.4, Tabelle 12):

$$\underline{U_2} = \underline{U_1} \cdot \frac{R}{R + \underline{Z_L}} \quad (6.1)$$

$$\underline{Z_L} = 3{,}6\,\Omega + 115\,mH \cdot \omega_{Netz} \quad (6.2)$$

$$R \in \{335{,}6\,k\Omega, 560\,\Omega, 280\,\Omega, 186{,}7\Omega\} \quad (6.3)$$

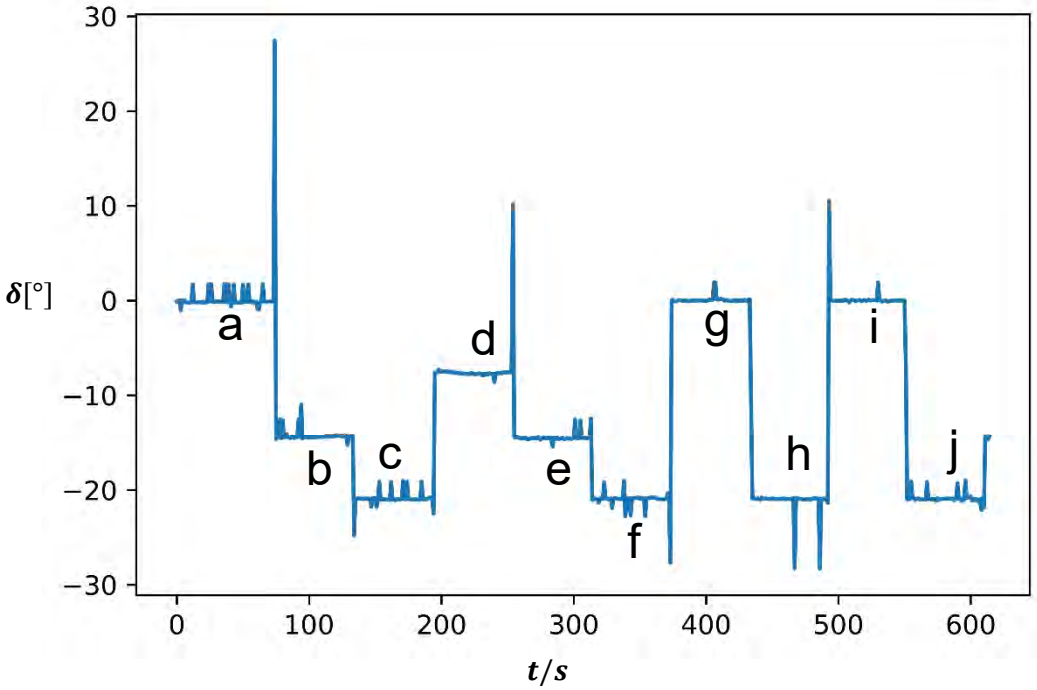
$$\delta_2 = \arg\left(\frac{R}{R + \underline{Z_L}}\right) \quad (6.4)$$

Tabelle 12:Spannungsphasenwinkel in Abhängigkeit vom Lastzustand des Modells

Schaltzustand	$\boldsymbol{R}$	$\boldsymbol{\delta_2}$
Kein Widerstand zugeschaltet	$335{,}6\,k\Omega$	$-0{,}012°$
Ein Widerstand zugeschaltet	$560\,\Omega$	$-7{,}3°$
Zwei Widerstände zugeschaltet	$280\,\Omega$	$-14{,}1°$
Drei Widerstand zugeschaltet	$186{,}7\,\Omega$	$-20{,}4°$

Die Ergebnisse der Messreihe sind in Abbildung 8 dargestellt.

Abbildung 8: Phasenwinkelmessreihe vor Median-Filterung



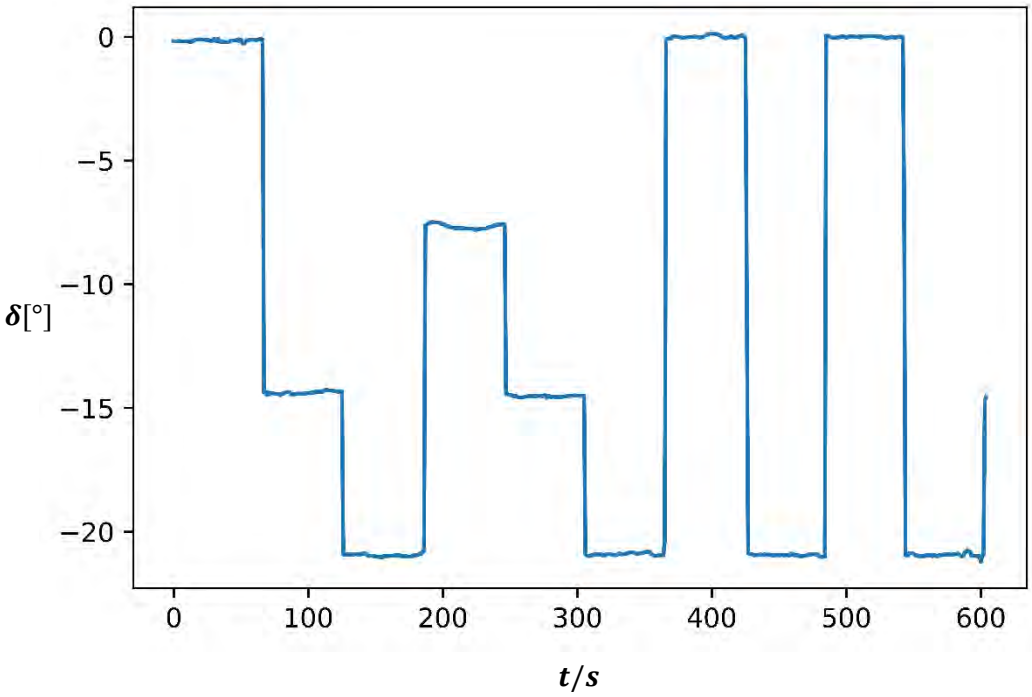
Zu sehen ist, dass die Plateaus der Messreihe prinzipiell erfasst werden, es aber ein deutliches Rauschen in den Messungen gibt. Gemessen wurden für die Abschnitte $a - j$ (Abbildung 8) folgende Durchschnittswinkel (Tabelle 13):

Tabelle 13: Vergleich der Messwerte mit errechneten Werten

Abschnitt	**Erwarteter Winkel**	**Messwinkel**
a	$-0{,}012°$	$0{,}04°$
b	$-14{,}1°$	$-14{,}25°$
c	$-20{,}4°$	$-20{,}86°$
d	$-7{,}3°$	$-7{,}67°$
e	$-14{,}1°$	$-14{,}45°$
f	$-20{,}4°$	$-20{,}96°$
g	$-0{,}012°$	$0{,}07°$
h	$-20{,}4°$	$-20{,}93°$
i	$-0{,}012°$	$0{,}02°$
j	$-20{,}4°$	$-20{,}97°$

Wegen des deutlichen Messrauschens (Abbildung 8) wird die Messreihe gefiltert: die letzten fünf Sekundenwerte für den Spannungsphasenwinkel werden gepuffert und deren Median abgebildet. Dies bewirkt eine Messlatenz von ca. drei Sekunden. Das Ergebnis ist in Abbildung 9 dargestellt. Es ist nun deutlich glatter. Ob der Trade-Off zwischen Messlatenz und -rauschfreiheit im Einzelfall sinnvoll ist, hängt von der Anwendung ab. Die Ergebnisse weisen für niedrige Winkel eine gute Präzision auf. Im Bereich höherer Winkel reichen die Abweichungen der Messungen von den erwarteten Werten um bis zu 5 % oder 0,6°, was ebenfalls unter dem geforderten Ziel (deutlich unter 1°) liegt.

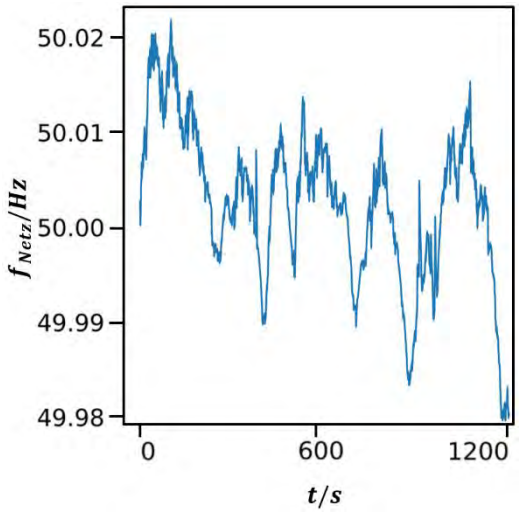
Abbildung 9: Phasenwinkelmessreihe nach Median-Filterung



7. Netzfrequenzmessung

Aus den kontinuierlichen Regressionsanalysen für jeden der sekündlichen Nulldurchgangsvektoren lässt sich auch eine Netzfrequenzmessung ableiten. Das Ergebnis ist in Abbildung 10 abgedruckt.

Abbildung 10: Netzfrequenzmessreihe



7.1. Kostenüberblick

Ziel des Projektes war es, ein möglichst kostengünstiges PMU-System zu entwerfen und umzusetzen. In Tabelle 14 werden die Positionen inklusive der Preise zum Zeitpunkt des Verfassens des Papiers (Ende 2022) aufgeschlüsselt.

Tabelle 14: Kostenübersicht für die Fertigung eines Messgerätes für einen Messknoten

Komponente	**Kostenpunkt (ca.)**
PCB	17 €
Bauteile (Counter, Register, Komparatoren, Optokoppler, Widerstände und Kondensatoren, Buck-Converter etc.)	37 €
GPS-Chip: GNSS 5 Click mit NEO-M8N	61 €
Raspberry Pi 3B	35 €
Summe	**150 €**

Mit kumulierten Materialkosten von 150 € sind die Kosten für das günstigste in der Literatur zu findende PMU-Messsystem, das in [5] vorgestellt wurde, deutlich unterboten worden.

8. Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Spannungsphasenwinkelmesssystem vorgestellt, das auf Standard-komponenten basiert und sehr kostengünstig herzustellen ist. Neben der Konzeption und Fertigung einer Stoppuhr-Platine wurde ein umfangreiches Softwarepaket zur Rauschfilterung und diverse Ergebnisanalysen vorgestellt. Es wurden verschiedene Verfahren zur Aufbereitung der

Messergebnisse und zur Zusammenführung der Daten von räumlich verteilten Messeinheiten implementiert und ausgewertet. Möglichkeiten der weiteren Nachschärfung liegen in der elektrischen Filterung des Eingangssignals aus der Netzspannung. Des Weiteren könnte es lohnen, die Untersuchung bezüglich der an einem einzelnen Knoten detektierten Spannungsphasenwinkelsprünge weiterzuführen, die in Abschnitt 5.1.2.3. skizziert wurde.

9. Danksagung

Die Arbeit entstand im Rahmen des Förderprojektes Progressus. Dieses Projekt wurde gefördert durch das Electronic Components and Systems for European Leadership Joint Undertaking unter dem Grant Agreement No. 876868. Dieses Gemeinschaftsprojekt wird durch das Forschungs- und Innovationsprogramm Horizon 2020 der Europäischen Union sowie durch Deutschland, die Niederlande, Spanien, Italien und die Slowakei unterstützt.

10. Quellen

[1] Handschin, E. (1987): Elektrische Energieübertragungssysteme. 2. Aufl. Heidelberg: Hüthig
[2] D. Schofield, F. Gonzalez-Longatt and D. Bogdanov, "Design and Implementation of a Low-Cost Phasor Measurement Unit: A Comprehensive Review," *2018 Seventh Balkan Conference on Lighting (BalkanLight)*, 2018, pp. 1-6, doi: 10.1109/BalkanLight.2018.8546936
[3] D. M. Laverty, R. J. Best, P. Brogan, I. Al Khatib, L. Vanfretti and D. J. Morrow, "The OpenPMU Platform for Open-Source Phasor Measurements," in IEEE Transactions on Instrumentation and Measurement, vol. 62, no. 4, pp. 701-709, April 2013
[4] D. M. Laverty, R. J. Best, P. Brogan, I. Al Khatib, L. Vanfretti and D. A. Mingotti, L. Peretto, R. Tinarelli, A. Angioni, A. Monti and F.
Ponci, "Calibration of Synchronized Measurement System: from the Instrument Transformer to the PMU," 2018 IEEE 9th International Workshop on Applied Measurements for Power Systems (AMPS), Bologna, 2018, pp. 1-5.
[5] B. Pinte, M. Quinlan and K. Reinhard, "Low voltage micro-phasor measurement unit (μPMU)," *2015 IEEE Power and Energy Conference at Illinois (PECI)*, 2015, pp. 1-4, doi: 10.1109/PECI.2015.7064888
[6] Internet-Quelle: Foo, A. https://www.fmad.io/blog/accuracy-of-pps-pulse-per-second Abgerufen: 16.12.2022. FMADIO GPS PULSE PER SECOND ACCURACY.
[7] Tomaž Šolc, “Interrupt response times on Arduino and Raspberry Pi” Jožef Stefan International Graduate School. Source code and raw data at https://github.com/avian2/interrupt-response-times.