

Pflichtenheft

# **Visuelle Programmiersprache für den Physikunterricht zur Datenerfassung auf einem Raspberry Pi**

**Version 0.3.0**

David Gawron      Stefan Geretschläger      Leon Huck  
Jan Küblbeck      Linus Ruhnke

31. Mai 2019

# Inhaltsverzeichnis

<b>1</b>	<b>Produktübersicht</b>	<b>4</b>
<b>2</b>	<b>Zielbestimmung</b>	<b>4</b>
2.1	Eignung für den Unterricht . . . . .	5
2.2	Einige hilfreiche Begriffsdefinitionen . . . . .	6
2.3	Handhabung von Bausteinprototypen . . . . .	6
2.4	Entwurf von Messkonfigurationen . . . . .	7
2.5	Gewährleisten von Persistenz . . . . .	7
2.6	Bereitstellung vorgefertigter Teile . . . . .	8
2.7	Handhabung von Messläufen . . . . .	8
2.8	Benutzbarkeit der GUI . . . . .	9
2.9	Abgrenzungskriterien . . . . .	10
<b>3</b>	<b>Produkteinsatz</b>	<b>10</b>
3.1	Anwendungsbereiche . . . . .	10
3.2	Zielgruppe . . . . .	11
3.3	Betriebsbedingungen . . . . .	11
<b>4</b>	<b>Produktumgebung</b>	<b>11</b>
4.1	Software . . . . .	11
4.2	Hardware . . . . .	12
4.3	PhyPiDAQ . . . . .	12
<b>5</b>	<b>Funktionale Anforderungen</b>	<b>12</b>
5.1	GUI . . . . .	12
5.1.1	Menüfeld . . . . .	13
5.1.2	Optional: Zusätzliche Funktionen im Menüfeld . . . . .	13
5.1.3	Konfigurationsfeld . . . . .	13
5.1.4	Darstellungsfenster . . . . .	13
5.2	Konfigurationserstellung . . . . .	14
5.2.1	Optional: Weiterentwicklung der Konfigurationserstellung . . . . .	14
5.3	Messablauf . . . . .	15
5.4	Fehlermeldungen . . . . .	15
5.5	Bedienungshilfen . . . . .	16
5.6	Sprache . . . . .	16
5.6.1	Optional: Internationalisierung . . . . .	16
5.7	Sonstiges . . . . .	16
5.7.1	Optional: Sonstiges . . . . .	17
<b>6</b>	<b>Produktdaten</b>	<b>17</b>

<b>7</b>	<b>Nichtfunktionale Anforderungen</b>	<b>17</b>
7.1	Produktleistungen . . . . .	17
7.2	Benutzbarkeit . . . . .	18
7.3	Zuverlässigkeit . . . . .	18
7.4	Sonstige . . . . .	19
7.5	Qualitätsanforderungen . . . . .	19
<b>8</b>	<b>Globale Testfälle und Testszenarien</b>	<b>19</b>
8.1	Einführung . . . . .	19
8.2	Testfälle zur Nutzung der Anwendung . . . . .	20
8.3	Testfälle zur Handhabung von Fehlern . . . . .	23
<b>9</b>	<b>Systemmodelle</b>	<b>25</b>
<b>10</b>	<b>Benutzungsoberfläche</b>	<b>27</b>
10.1	Ziel der Benutzeroberfläche . . . . .	27
10.2	Generell . . . . .	27
10.3	Eingabebegeäte . . . . .	27
10.4	Überblick . . . . .	27
10.5	Die einzelnen Teile . . . . .	28
10.5.1	Systemmenüleiste . . . . .	28
10.5.2	Auswahl . . . . .	31
10.5.3	Konfigurationsfeld . . . . .	32
10.6	Erweiterungsmöglichkeiten . . . . .	33
10.6.1	Startbildschirm . . . . .	33
<b>11</b>	<b>Spezielle Anforderungen an die Entwicklungsumgebung</b>	<b>33</b>
<b>12</b>	<b>Zeit- und Ressourcenplanung</b>	<b>34</b>
12.1	Projektphasen . . . . .	34
12.2	Risikomanagement . . . . .	34
<b>13</b>	<b>Glossar</b>	<b>34</b>

# 1 Produktübersicht

Die im Rahmen des *OSL*<sup>2</sup> zu entwickelnde Anwendung soll es Lehrern ermöglichen, Schülern die Grundlagen von physikalischen Messtechniken zu vermitteln (siehe Kapitel 2). Demzufolge kommt die Anwendung in Schulen zum Einsatz. Die Zielgruppe sind Schüler in der siebten Klasse aufwärts (siehe Kapitel 3). Die Anwendung läuft in einer *Java Virtual Machine* und erhält Messwerte von *Sensoren* (siehe Kapitel 4). Die hardwarenahe Ansteuerung der Sensoren soll durch das bereits vorhandene Softwarepaket *PhyPiDAQ* erfolgen. (siehe Kapitel 9). Detaillierte funktionale und nicht funktionale Anforderungen sind in Kapitel 5 bzw. 7 aufgelistet. Die Anwendung soll über eine *Grafische Benutzeroberfläche*, mit Maus und Tastatur, bedient werden. (siehe Kapitel 10).

## 2 Zielbestimmung

Die Anwendung soll es Lehrern ermöglichen, Schülern ab der siebten Klasse Grundkenntnisse der digitalen Messwerterfassung in einer für Schüler interessante und motivierenden Weise näher zu bringen.

In einem nicht etwas komplexeren Versuchsaufbau sind die wesentlichen Schritte einer messtechnischen Verarbeitung die Folgenden:

1. Abtastung der physikalischen Größen
2. Transformation der Messwerte durch mathematische Funktionen oder programmtechnisch definierte Verarbeitung
3. die Darstellung der transformierten Werte in einer zur Beantwortung der Fragestellung geeigneten Form.

Das projektierte Softwareprodukt soll die Schüler dabei unterstützen, die gängigen Schritte einer messtechnischen Verarbeitung zu definieren und zueinander in Beziehung zu setzen. Hierzu soll das Produkt *Konfigurationsbausteine* unterschiedlichen Typs anbieten, aus denen ein Schüler dann eine Messkonfiguration entwerfen kann. Die Konfigurationsbausteine sollen anpassbar sein, nicht nur um die Software für möglichst viele physikalische Fragestellungen einsetzen zu können sondern auch um den Bedürfnissen unterschiedlicher Altersstufen gerecht zu werden.

Natürlich soll dann auch der Betrieb der Messkonfiguration gesteuert werden können und das Produkt sollte eventuell auch Einblicke in Abläufe während des Betriebs unterstützen, beispielsweise zur Analyse unerwünschten Verhaltens.

## 2.1 Eignung für den Unterricht

Aus didaktischer Sicht überflüssige technische Details wie z. B. die zum Auslesen der Sensoren notwendigen Protokolle sollen vor dem Schüler in jedem Fall verborgen bleiben. Der Schüler wird so nicht überfordert, sondern soll ermutigt werden, selbstständig mit der Software umzugehen. Dabei kann er im spielerischen Umgang mit Versuchsaufbauten Prinzipien der digitalen Messtechnik wie z. B. Kaskadierung und natürlich auch das Grundprinzip von Ursache und Wirkung erfahren.

Es wird eine graphische Oberfläche angeboten, die es dem Schüler ermöglicht, allein per Drag and Drop eine Messanordnung zu entwerfen. Überflüssige Details, die dahinter stecken, bleiben vor dem Schüler verborgen. Die Anwendung motiviert den Schüler dazu, mit Sensoren, Transformationen und Darstellungen zu spielen und Dinge auszuprobieren. Dabei wird ihm durch eine intuitive Status- und Fehleranzeige gezeigt, ob sein Entwurf funktioniert. Wenn nicht, dann hilft sie ihm das Problem einzugrenzen und gibt Hinweise zur Problembehebung. Eine lästige Fehlersuche soll dem Schüler weitestgehend erspart bleiben.

Außerdem liefert die Anwendung dem Schüler zu den vorhandenen Bausteinen und zu der Anwendung allgemein die nötigen Informationen, die er für die Nutzung braucht. Dabei wird auf ein ausführliches Tutorial am Anfang verzichtet. Die Anwendung liefert die Informationen häppchenweise durch Informationsanzeigen an den jeweilig relevanten Stellen. Damit findet der Schüler die Hilfe, die er sucht, an der Stelle, an der er sie braucht.

Die Anwendung ermöglicht es dem Lehrer vor dem Unterricht, eine Reihe von Messversuchen teilweise oder ganz zu konfigurieren und zu speichern. Dabei kann er genau bestimmen, was er den Schülern zeigen will. Diese Konfigurationen kann er dann schnell und einfach im Unterricht zum Einsatz bringen.

Weiter ermöglicht es die Anwendung, dass Schüler aus höheren Klassen mit mehr Details der digitalen Messwerterfassung zusammen gebracht werden. Diese nutzen keine oder nur eine abstrakt bzw. lückenhaft vorgefertigte Konfiguration. Sie können dann auch selbst einen Baustein modifizieren oder erstellen. Trotz des höheren Detailgrads bleibt die Anwendung übersichtlich und strukturiert. Damit bleiben auch komplexere Messversuche für den Schüler motivierend.

Die Anwendung ist als Bindeglied zwischen PhyPiDAQ und dem Nutzer zu verstehen. Sie ermöglicht dem Nutzer, über eine übersichtliche, strukturierte und intuitive GUI sowie über eine einfache Bedienung die Nutzung eines *Raspberry Pi* mit PhyPiDAQ. Dadurch soll dem Schüler digitale Messwerterfassung, Physik und auch Informatik in einer motivierenden Weise näher gebracht werden. Womöglich kann die Anwendung den Schüler auch für diese Themen begeistern.

## 2.2 Einige hilfreiche Begriffsdefinitionen

Im Folgenden werden *Musskriterien*, *Sollkriterien* und *Wunschkriterien* zur Abnahme des projektierten Softwareprodukts definiert. Während der Entwicklung wird ihre Umsetzung mit hoher, mittler und niedriger Priorität betrieben. Musskriterien werden mit MK, Sollkriterien werden mit SK, Wunschkriterien werden mit WK abgekürzt.

Für die Beschreibung der Kriterien ist es hilfreich, ein paar grundsätzliche Überlegungen vorzunehmen, wie Bausteine geartet sein müssen, um eine (Teil-)Aufgabe erfüllen zu können. Denn es macht Sinn, gänzlich unterschiedliche Teilaufgaben dann auch von Bausteinen unterschiedlichen Typs erledigen zu lassen bzw. im Umkehrschluss ähnliche Teilaufgaben jeweils einen bestimmten Bausteinotyp vorzusehen.

- Bausteine vom Typ Sensor haben nur Ausgänge. Ein solcher (logischer) Sensorbaustein muss alle Informationen referenzieren können, die zum Ansprechen eines Messgeräts bzw. physikalischen Sensors benötigt werden. Da ein Messgerät Ausgänge bzw. Messkanäle haben kann, kann ein Sensorbaustein mehrere Ausgänge haben. Eingänge besitzt ein Sensorbaustein nicht.
- Bausteine vom Typ Transformation haben einen oder mehrere Eingänge sowie einen oder mehrere Ausgänge. Für jeden Ausgang kann ein Transformationsbaustein eine Vorschrift zur Berechnung eines Ausgangswertes aus einem Satz von Eingangswerten beinhalten. Eine Berechnungsvorschrift soll durch eine mathematische Funktionen oder durch eine programmtechnisch definierte Verarbeitung definiert werden können.
- Bausteine vom Typ Darstellung haben einen oder mehrere Eingänge. Ein Darstellungsbaustein soll definieren können, wie ein Satz von Eingangswerten die Erstellung bzw. Aktualisierung einer Darstellung beeinflusst. Ausgänge besitzt ein Darstellungsbaustein nicht.

## 2.3 Handhabung von Bausteinprototypen

Möchte ein Benutzer einem Entwurf ein neues Bausteinexemplar hinzufügen, so wird einem *Bausteinprototypen* eine Kopie angelegt.

- **MK 1** Der Benutzer kann die Eigenschaften eines Bausteinprototyps einsehen.
- Der Benutzer soll von allen Bausteintypen neue Prototypen erstellen können. Dies kann auf zweierlei Art und Weise geschen:

- **MK 2** durch Anpassung bereits vorhandener (evtl. auch generischer) Bausteintypen,
- **SK 1** durch Erweiterung um neue Softwarekomponenten, die definierte programmtechnische Schnittstellen nutzen.
- **MK 3** Der Benutzer soll die Eigenschaften benutzererstellter Bausteinprototypen verändern können.
- **MK 4** Der Benutzer soll benutzererstellte Bausteinprototypen löschen können.

## 2.4 Entwurf von Messkonfigurationen

- **MK 5** Hinzufügen eines Bausteins zur Messkonfiguration, d. h. Auswahl eines Prototypen und Platzieren des Bausteinexemplars auf der Entwurfsfläche
- **MK 6** Bearbeiten der Baustein-Eigenschaften oder Verändern der Position des Bausteins auf der Entwurfsfläche
- **MK 7** Löschen eines Bausteins aus der Konfiguration
- **MK 8** Erstellen einer Weiterleitungsregel bzw. eines „Drahtes“ vom Ausgang eines Bausteins zum Eingang eines anderen Bausteins.
- **MK 9** Löschen einer Weiterleitungsregel bzw. eines „Drahtes“
- **SK 2** Anbieten einer Undo- und einer Redo-Funktion
- **WK 1** Hinzufügen, Darstellen, Bearbeiten und Löschen ergänzender Text- und evtl. auch Bildinformation, beispielsweise zur ausführlichen Beschreibung der physikalische Hintergründe oder für Memos, To-dos, etc

## 2.5 Gewährleisten von Persistenz

- **MK 10** Eine Messkonfiguration, einschließlich der grafischen Anordnung der Bausteine, lässt sich in einer Datei speichern.

- **MK 11** Eine zuvor gespeicherte Messkonfiguration lässt sich zum weiteren Gebrauch durch Laden aus der entsprechenden Datei wiederherstellen.
- **SK 3** Vom Benutzer erstellte Bausteinprototypen lassen sich auswählen und in einer Datei speichern.
- **SK 4** Zuvor gespeicherte Bausteinprototypen lassen sich zum weiteren Gebrauch durch Laden aus der entsprechenden Datei wiederherstellen.

## 2.6 Bereitstellung vorgefertigter Teile

- Die Anwendung enthält eine Reihe von vorgefertigten Bausteinprototypen:
  - **MK 12** Sensorbausteine generischer Natur, beispielsweise für einen Analog/Digital-Wandler, sowie bereits spezialisierte Sensorbausteine, beispielsweise für Licht, Temperatur oder Lage,
  - **MK 13** Transformationsbausteine generischer Natur, beispielsweise für lineare Abbildungen oder für bedingte Ausgaben, sowie speziellere Transformationsbausteine, beispielsweise ein Baustein zur Datenstromduplizierung oder ein Multiplexer als beispielhafte Spezialisierung eines „bedingten“ Bausteins.
  - **MK 14** für die meisten Zwecke gut anpassbare Darstellungsbausteine, beispielsweise zur Generierung von Tabellen oder Koordinatensystemen.
- **MK 15** Die Anwendung enthält mindestens drei vorgefertigte Messkonfigurationen unterschiedliche Komplexität. Die Konfigurationen sollten möglichst ansprechend und motivierend sein. Denkbar wäre beispielsweise auch Spiele, die das Replizieren bzw. „Nachmachen“ von vorgegebenen Messergebnissen vorsehen.

## 2.7 Handhabung von Messläufen

Die folgenden Features bzw. Kriterien sollen den Benutzer dabei unterstützen, zufriedenstellende *Messläufe* zu erhalten.

- **MK 16** Der Benutzer kann einen Messlauf mit adäquater Eingabe starten, pausieren, fortführen und beenden.
- **MK 17** Der Benutzer kann vor Start eines Messlaufs die Zeitpunkte definieren, zu denen an den Ausgängen der Sensorbausteine die zuletzt abgetasteten Mess-



werte registriert und von den nachgelagerten Bausteinen gemäß ihrer Definition sukzessive weiterverarbeitet werden.

- **SK 5** Während des Entwurfs bzw. der Bearbeitung einer Messkonfiguration kann der Nutzer Checks vornehmen, um überprüfen, ob die gesamte Messkonfiguration oder Teile in ihrem derzeitigen Zustand funktionieren. Bei negativem Ergebnis erhält der Benutzer möglichst aussagekräftige Hinweise.
  - Ist beispielsweise ein physikalischer Sensor nicht funktionsfähig, beispielsweise weil er nicht oder fehlerhaft angeschlossen ist, so kann der Benutzer dies frühzeitig erkennen.
  - Sind Ein- oder Ausgänge von Bausteinen nicht verdrahtet, obwohl dies für eine ordnungsgemäße Funktion des Bausteins erforderlich ist, so kann der Benutzer dies ebenso frühzeitig erkennen.
- **WK 2** Bausteine vom Typ Sensor sollen als Alternative zu abgetasteten Messwerten auch vorgehaltene Testdaten bereithalten können, so dass eine Messkonfiguration auch ohne Messgeräte oder physikalische Sensoren getestet oder vorgeführt werden kann.
- **WK 3** Der Benutzer soll bei jedem Baustein den lokalen Datenfluss während eines Messlaufs aufzeichnen, einsehen und speichern können.

**WK 4** Für einen Darstellungsbaustein wäre eine Druckfunktion für die von ihr generierte Darstellung wünschenswert.

## 2.8 Benutzbarkeit der GUI

- **WK 5** Die Benutzungsoberfläche bietet als Sprache mindestens Deutsch an und muss leicht um weitere Sprachen ergänzt werden können. Sobald die Anwendung mehr als eine Sprache anbietet, kann der Benutzer die GUI von einer Sprache auf eine andere umstellen.
- **SK 6** Die Anwendung reagiert auf jede Benutzereingabe innerhalb von zwei Sekunden. Benötigt die Anwendung zur Reaktion auf eine Benutzereingabe länger als zwei Sekunden, so präsentiert sie dem Benutzer fortlaufend mindestens ein Mal pro Sekunde ein möglichst aussagekräftiges Zwischenergebnis, mindestens in Form eines Fortschrittsbalkens. Nach Ablauf eines einstellbaren Timeouts präsentiert die Anwendung eine möglichst aussagekräftige Fehlermeldung. Benötigt die Be-

arbeitung einer Benutzereingabe länger als 2 Sekunden, so hat der Benutzer die Möglichkeit, den Bearbeitungsvorgang abubrechen.

- **SK 7** Beim Entwurf einer Messkonfiguration soll der Benutzer die von ihm beabsichtigten Aktionen per Drag and Drop anstoßen können. Dies betrifft beispielsweise das Hinzufügen eines Konfigurationsbausteins oder das Erstellen einer Kante vom Ausgang eines Bausteins mit dem Ausgang Baustein eines anderen Bausteins.
- **WK 6** Da nicht immer alle Eingänge oder Ausgänge Verwendung finden müssen, sollen nicht benötigte Eingänge oder Ausgänge bei einem Baustein bzw. Bausteinprototyp ausgeblendet bzw. unsichtbar gemacht werden können.
- **SK 8** Die Anwendung hält interaktiv zugängliche Erklärungen und Informationen zu den einzelnen GUI-Elementen bereit.
- **WK 7** Die Farben der GUI-Elemente sollen für den Benutzer anpassbar sein. Bei Benutzung der vorgeschlagener Farbschemata soll die grafische Benutzungsschnittstelle möglichst barrierefrei sein, d. h. die Semantik sollte auch für Benutzer mit Farberkennungsschwächen ohne Probleme zu erkennen sein.
- **WK 8** Zur weiteren Barrierefreiheit sollte das Produkt beispielsweise für Schüler mit Sehbehinderung variable Darstellungsgrößen anbieten.

## 2.9 Abgrenzungskriterien

- Unterschiedliche Benutzerkonten bzw. personalisierte Arbeitsumgebungen für die einzelnen Benutzer sieht das Produkt nicht vor.
- Die zu erstellende Anwendung sieht keinerlei Erfassung oder Speicherung personenbezogener Nutzerdaten vor. Der Einsatz der Software erfordert dementsprechend keine Einverständniserklärung der Nutzer gemäß europäischer *Datenschutz-Grundverordnung*.

# 3 Produkteinsatz

## 3.1 Anwendungsbereiche

Die Anwendung ist für die Verwendung in Schulklassen ab der 7. bis zu 10. Klasse für Schüler und Lehrer im Physikunterricht konzipiert. Ebenfalls soll die Anwendung in

Physikprojekten, sowie *Science Labs* verwendet werden können. Als *Open Source* Projekt im Rahmen des OSL<sup>2</sup> steht die Anwendung jedoch jedem Interessierten zur Verwendung und Weiterentwicklung zur Verfügung.

### 3.2 Zielgruppe

Die Zielgruppen sind hauptsächlich Schülerinnen und Schüler im Physikunterricht, welche die Anwendung verwenden, um erste eigene Messungen durchzuführen. Das Ziel ist es den Schülern erste Einblicke in Messtechniken und Zusammenhänge zwischen Ursache und Wirkung zu zeigen. Die Anwendung soll ebenfalls nicht nur von physikbegeisterten Schülern verwendet werden können, sondern auch für Schüler ohne ein großes Vorwissen in der Physik und Messtechniken. Diesen Schülern soll ebenfalls eine interessante und einfach zu bedienende Plattform geboten werden.

### 3.3 Betriebsbedingungen

Die Anwendung läuft auf Computern, welche über eine USB-Schnittstelle mit einem Raspberry Pi verbunden sind, auf welchem das Programm PhyPiDAQ läuft. Über das PhyPiDAQ werden die an das Raspberry Pi angeschlossenen Messsensoren adressiert, welche die Messdaten wiederum über das Raspberry Pi an die am Computer laufende Anwendung schicken.

## 4 Produktumgebung

Die Anwendung läuft auf einem Computer, die Messdaten werden über Sensoren an einem Raspberry Pi erfasst.

### 4.1 Software

Die Anwendung läuft auf Computern mit den Betriebssystemen Linux ab Kernel 4.15 und Microsoft Windows ab Windows 10. Die Anwendung muss auf dem Computer vollständig installiert sein.

## 4.2 Hardware

Die Anwendung läuft auf Computern, welche für den generellen Einsatz in der Schule gedacht sind. Die enthaltenen Komponenten sollten beispielsweise ein aktueller 4-Kern-Prozessor, 8 GB RAM, mindestens 256 GB Speicherplatz sein.<sup>1</sup> Schnittstellen für USB und Netzwerk sind ebenfalls erforderlich. Auf dem per USB-Schnittstelle verbundenen Raspberry Pi der Variante 3 Mod. B+ muss PhyPiDAQ installiert und verwendungsfähig sein. Die an den Raspberry Pi angeschlossenen Sensoren müssen richtig und sinnvoll angeschlossen sein.

## 4.3 PhyPiDAQ

Bei PhyPiDAQ<sup>2</sup> handelt es sich um eine Anwendung zur Datenerfassung und Analyse mit einem Raspberry Pi. Diese ist nicht Bestandteil des Produktes, wird jedoch zur Datenerfassung und Datenverarbeitung und somit zur Funktionalität der Anwendung benötigt. Das Programm, welches in der Programmiersprache *Python 3* geschrieben ist bietet einheitliche Schnittstellen zur Verwendung der Sensoren mit dem Raspberry Pi. Die Verwendung setzt jedoch Grundkenntnisse im Umgang mit Python 3 und dem Raspberry Pi voraus, weshalb es ungeeignet für den Einsatz in Schulen ist.

# 5 Funktionale Anforderungen

## 5.1 GUI

F010 Die Benutzer erreichen nach Öffnung der Anwendung direkt die *GUI*.

F020 Die Anwendung ist konfigurierbar.

F030 Die Anwendung besitzt eine Dateiverwaltung.

F040 Die Anwendung bietet eine Hilfe-Oberfläche.

---

<sup>1</sup>Beispielkonfiguration eines Schulcomputers nach dem Beraterkreis zur IT-Ausstattung von Schulen des Bayrischen Staatsministerium für Bildung und Kultus, Wissenschaft und Kunst, Votum 2016 unter <https://www.mebis.bayern.de/infoportal/konzepte/it-ausstattung/votum/>

<sup>2</sup><https://github.com/GuenterQuast/PhyPiDAQ>

### 5.1.1 Menüfeld

F050 Es gibt eine graphische Auswahl an vordefinierten Sensoren.

F060 Es gibt eine graphische Auswahl an vordefinierten Transformationen.

F070 Es gibt eine graphische Auswahl an vordefinierten Darstellungen.

F080 Der Benutzer erhält durch visuelle Repräsentationen der Konfigurationsbausteine eine Darstellung ihrer Komplexität.

### 5.1.2 Optional: Zusätzliche Funktionen im Menüfeld

F090 Der Benutzer soll in F050 weitere Sensoren hinzufügen können.

F100 Der Benutzer soll in F060 weitere Transformationen hinzufügen können.

F110 Die in Transformationen in F100 sollen in eigenen Python-Skripten geschrieben werden können.

### 5.1.3 Konfigurationsfeld

F120 Durch *Drag-and-Drop* kann der Benutzer Konfigurationsbausteine im Konfigurationsfeld platzieren.

F130 Der Benutzer kann eine Messung starten.

### 5.1.4 Darstellungsfenster

F140 Das Darstellungsfenster ist bei der Initialisierung der Anwendung leer.

F150 Durch die *Messkonfiguration* im Konfigurationsfeld wird durch F130 automatisch die Darstellungsart im Darstellungsfenster geöffnet.

## 5.2 Konfigurationserstellung

- F160 Über F030 kann der Benutzer eine gespeicherte Messkonfiguration öffnen oder alte Messwerte verwenden.
- F170 Durch F160 geladene Konfigurationen und Messdaten werden automatisch nach Format überprüft.
- F180 Aus F050 kann der Benutzer Sensoren durch Drag-and-Drop in das Konfigurationsfeld ziehen.
- F190 Die Anwendung sollte automatisch überprüfen, ob der ausgewählte Sensor richtig angeschlossen ist.
- F200 Durch F190 wird dem Benutzer automatisch eine visuelle Rückmeldung gegeben.
- F210 Aus F060 kann der Benutzer eine oder mehrere Transformationen durch Drag-and-Drop in das Konfigurationsfeld ziehen.
- F220 Die ausgewählten Transformationen kann der Benutzer mit einem ausgewählten Sensor verknüpfen.
- F230 Aus F070 kann der Benutzer eine Darstellungsart per Drag-and-Drop in das Konfigurationsfeld zu ziehen.
- F240 Die Ausgewählte Darstellung kann der Benutzer mit einer Transformation verknüpfen.
- F250 Über F030 kann der Benutzer seine eigene Messkonfiguration speichern.
- F260 Bausteine können durch Drag-and-Drop in die Menüleiste wieder aus dem Konfigurationsfeld entfernt werden.

### 5.2.1 Optional: Weiterentwicklung der Konfigurationserstellung

- F270 Die Anwendung besitzt einen Check-Knopf, welcher die Messkonfiguration auf Vollständigkeit und Richtigkeit kontrolliert
- F280 Durch F270 wird durch visuelle Rückmeldung dargestellt, ob die Konfiguration verwendet werden kann.

### 5.3 Messablauf

- F290 Der Nutzer kann Messlänge und die Wertebereiche einer Messung festlegen.
- F300 Der Nutzer kann eine Messung nach der Messkonfiguration starten.
- F310 Bei F300 wird automatisch überprüft ob die Messkonfiguration eine sinnvolle Kombination von Sensoren, Transformationen und Darstellungen ist.
- F320 Durch F300 wird automatisch mit der visuellen Darstellung der Messdaten begonnen.
- F330 Der Benutzer kann eine Messung löschen.
- F340 Bei F330 wird die Messung gestoppt und die visuelle Darstellung auf den Ausgangszustand bringen.
- F350 Durch F330 wird nicht die Messkonfiguration gelöscht.
- F360 Durch F330 muss der Benutzer erst die Messung starten um weiter zu messen.
- F370 Der Benutzer kann eine Messung pausieren.
- F380 Der Benutzer kann eine Messung fortsetzen.
- F390 Der Benutzer kann eine Messung durch F380 nur fortsetzen, wenn sie zuvor durch F370 pausiert wurde.
- F400 Der Benutzer kann die Messdaten speichern.
- F410 Der Benutzer kann den durch die Messung erzeugten Graphen speichern.
- F420 Bei F400 und F410 öffnet sich automatisch das Verzeichnis und der Nutzer muss einen eigenen Dateinamen eingeben und speichern.

### 5.4 Fehlermeldungen

- F430 Durch F170 wird bei einem falschen Format eine aussagekräftige Fehlermeldung zurückgegeben.

- F440 Die Anwendung sollte dem Benutzer automatisch bei Betätigen des Messung löschen-Button oder bei Schließen der Anwendung darauf hinweisen, dass die Messdaten ohne Speichern der Messdaten verloren gehen.
- F450 Die Anwendung sollte dem Benutzer eine aussagekräftige Fehlermeldung zurückgeben, falls es zu einem Datenabbruch der Messdaten kommt.
- F460 Durch F310 wird bei einer Benutzerkonfiguration, welche nicht verwendet werden kann eine aussagekräftige Fehlermeldung zurückgegeben.

## **5.5 Bedienungshilfen**

- F470 Bei enthaltenen Konfigurationsbausteine erhalten Informationen, über welche der Benutzer Informationen und Hilfestellung bereitgestellt bekommt.
- F480 Über F040 erhält der Nutzer eine kurze Beschreibung zur Funktionalität der Anwendung.

## **5.6 Sprache**

- F490 Die Anwendung ist in deutscher Sprache.

### **5.6.1 Optional: Internationalisierung**

- F500 Die Anwendung stellt dem Benutzer weitere Sprachpakete für die GUI zur Verfügung.
- F510 Durch F020 ist die Sprache für den Nutzer änderbar.

## **5.7 Sonstiges**

- F520 Die in der Anwendung enthaltenen Farben sollten mit Rücksicht auf Benutzer mit Rot-Grün Schwäche oder Farbenblindheit ein barrierefreies Verwendung der Anwendung ermöglichen.



### 5.7.1 Optional: Sonstiges

F530 Durch F020 sollte der Benutzer die in der Anwendung verwendeten Farben ändern können.

F540 Durch F020 sollte der Benutzer die in der Anwendung verwenden Buchstabengröße verändern können.

## 6 Produktdaten

Zu speichern sind ausschließlich:

D010 Messkonfiguration(en)

D020 Messdaten

D030 Sensorkonfigurationsdaten

## 7 Nichtfunktionale Anforderungen

### 7.1 Produktleistungen

NF010 Auslesen von *Messdaten* von einem einzelnen Sensor innerhalb von 50 ms.

NF020 Alle 20 ms können neue Messdaten angefordert werden.

NF030 Bis zu drei Sensoren können zeitgleich verwendet werden, ohne die Leistung zu beeinträchtigen.

NF040 Mehrere *Transformationen* können hintereinander eingestellt werden.

NF045 Verarbeitung von Daten benötigt maximal 10 ms pro Transformation.

NF050 Mehrere *Darstellungen* können zeitgleich verwendet werden.

NF055 Darstellung von Daten benötigt maximal 30 ms pro Darstellungsbaustein.

NF060 Die Anwendung reagiert auf Benutzereingaben innerhalb von zwei Sekunden; mindestens mit einem Zwischenergebnis, Ladebalken oder ähnlichem.

## **7.2 Benutzbarkeit**

NF110 Die *Benutzeroberfläche* ist ergonomisch gestaltet, orientiert an ISO 9241.

NF115 Nichttriviale Funktionen werden dem Benutzer erklärt.

NF120 Ungespeicherte Daten werden nicht ohne Warnung verworfen.

NF130 Über Programmfehler wird der Benutzer gegenüber durch aussagekräftige Fehlermeldungen informiert.

NF140 Schriftgröße von Text kann verändert werden.

NF150 Farbschema von farbigen *UI*-Elementen ist veränderbar.

NF160 Die Sprache der Benutzeroberfläche kann verändert werden.

## **7.3 Zuverlässigkeit**

NF210 Konsistenz: Bei gleichen Vorbedingungen führen identische Eingaben auch zum gleichen Ergebnis.

NF220 Korrektheit: Bei korrekten Eingaben werden die erwarteten funktionalen Anforderungen erfüllt.

NF230 Unerwartete und fehlerhafte Eingaben führen nicht zum Absturz der Anwendung.

NF240 Verbindungsverlust von Sensoren führt nicht zum Absturz der Anwendung.

NF250 Unerwartete Messdaten (z.B. außerhalb eines eingestellten Wertebereichs) führen nicht zum Absturz der Anwendung.

## 7.4 Sonstige

- NF310 Die Software erfüllt die Definition von Open Source- und freier Software.<sup>3</sup>
- NF320 Die Software ist erweiterbar und anpassbar an veränderte Umstände, auch durch Außenstehende.
- NF330 Die Datenschutz-Grundverordnung wird nicht verletzt; d.h. personenbezogene Daten werden ausschließlich dann verarbeitet, wenn sie vom Benutzer freiwillig eingespeist werden.
- NF340 Daten werden ausschließlich lokal gespeichert.

## 7.5 Qualitätsanforderungen

Auf folgende Qualitätsmerkmale wird entsprechend besonders Wert gelegt:

Merkmal	sehr wichtig	wichtig	weniger wichtig	unwichtig
Robustheit	X			
Zuverlässigkeit		X		
Benutzerfreundlichkeit	X			
Effizienz		X		
Portierbarkeit			X	
Kompatibilität			X	

## 8 Globale Testfälle und Testszenarien

### 8.1 Einführung

In diesem Kapitel werden einige Testfälle zur Benutzung der Anwendung und zur Handhabung von Fehlern dargestellt. Zum besseren Verständnis werden noch einmal einige verwendete Begriffe und der generelle Ablauf einer Nutzung erläutert. Die Anwendung bezeichnet die gesamte in diesem Projekt erstellte Software. Sie benutzt Messkonfigurationen um einen Versuchsaufbau zu modellieren. Eine Messkonfiguration besteht aus Bausteinen(Sensor, Transformation, Darstellung) und deren Verbindungen untereinander. Die Bausteine können selbst wieder durch eine *Konfigurationsdatei* eingestellt werden. Die Messkonfigurationen werden in der Konfigurationsfeld erstellt. Außerdem

---

<sup>3</sup><https://www.gnu.org/philosophy/free-sw.de.html>

können sie geladen und gespeichert werden. Beim Starten überprüft die Anwendung, ob eine gültige Messkonfiguration vorliegt. Um die resultierenden Daten und Graphen zu speichern, muss die Messung gestoppt werden.

## 8.2 Testfälle zur Nutzung der Anwendung

### T010 Starten der Anwendung und Hilfe

**Testziel:** Teste das Verhalten der Anwendung aus der Sicht eines Benutzers, der diese zum ersten Mal verwendet.

**Vorbedingung:** Die Anwendung ist installiert. Zu Sehen ist der Desktop des Benutzers mit einer Verknüpfung zur Anwendung.

**Aktion:** Der Benutzer öffnet die Anwendung über die Verknüpfung. Danach informiert er sich über die Anwendung über die Hilfe in der Systemleiste.

**Reaktion:** Die Anwendung öffnet sich. Zu Sehen ist das Hauptfenster mit leerer Konfigurationsfeld und leeren Darstellungen. Nach dem Öffnen der Hilfe, sieht der Benutzer die nötigen Informationen zur Bedienung der Anwendung.

**Nachbedingung:** Nach dem öffnen der Anwendung, überprüft diese, ob ein Raspberry Pi angeschlossen ist. Jeder dargestellte Text ist auf Deutsch.

**Ergebnis:** Der Benutzer kann die Anwendung starten. Die Bedienung der Anwendung ist komplett auf Deutsch möglich.

**Wichtige abgedeckte Funktionale Anforderungen, die noch nicht abgedeckt wurden:** F010 erreiche GUI nach Start, F140 leere Darstellung nach Anwendungsstart, F480 Hilfe zu Anwendung, F490 Texte der Anwendung auf Deutsch

### T020 Starten der Demo

**Testziel:** Teste das Verhalten der Anwendung beim Ausführen der Demo. Die Demo nutzt Daten aus einer Datei und benötigt keinen angeschlossenen Raspberry Pi.

**Vorbedingung:** Die Anwendung ist gestartet worden. Zu Sehen ist das offene Hauptfenster mit leerer Konfigurationsfeld.

**Aktionen:** Der Benutzer öffnet die Messkonfigurationsdatei der Demo mit Hilfe der Ladefunktion der Systemleiste. Sobald die Messkonfiguration geladen ist, erstellt der Benutzer den Messversuch und startet ihn danach.

**Reaktionen:** Nach dem Laden ist die Messkonfiguration in der Konfigurationsfeld sichtbar. Nach dem Starten der Messung sieht der Benutzer wie die Daten im Graph dargestellt werden.

**Nachbedingung:** Die Anwendung ist offen. Eine gültige Messkonfiguration ist in der Konfigurationsfeld geladen. Die Messung läuft ohne Fehler und die Ergebnisse werden richtig dargestellt.

**Ergebnis:** Die Anwendung kann eine Messkonfiguration, die nur Daten aus einer Datei benötigt, problemlos ohne angeschlossenen Raspberry Pi durchführen.

**Wichtige abgedeckte Funktionale Anforderungen, die noch nicht abgedeckt wurden:** F130 Starten einer Messung, F160 Laden einer Messkonfiguration

T030 Lehrer erstellt und speichert eine Messkonfiguration

**Testziel:** Teste eine typische Verwendung der Software anhand des gegebenen Szenarios. Dabei wird das Erstellen, Verändern und Speichern einer Messkonfiguration getestet.

**Vorbedingung:** Die Anwendung ist gestartet und es liegt keine Messkonfiguration vor. Die Anwendung hat Zugriff auf einen laufenden Raspberry Pi mit PhyPiDAQ. Die Sensoren die verwendet werden sollen sind ordnungsgemäß angeschlossen.

**Aktionen:** Der Lehrer zieht zwei angeschlossene Sensoren und eine Transformation in die Konfigurationsfeld. Danach ändert er einige Einstellungen an der Messkonfiguration. Das Ergebnis speichert er über die Systemleiste als Messkonfigurationsdatei.

**Reaktion:** Nach jedem Hinzufügen eines Sensors oder einer Transformation, wird an der entsprechenden Stelle das jeweilige Icon sichtbar. Nach dem Speichern der Messkonfiguration sieht der Lehrer die erstellte Datei.

**Nachbedingung:** Die Anwendung prüft beim Hinzufügen eines Sensor-Icons, ob der Sensor angeschlossen ist. Die Anwendung speichert die angepassten Einstellungen, sofern sie gültig sind. Die Messkonfigurationsdatei wird erstellt, unabhängig ob die Messkonfiguration vollständig oder gültig ist.

**Ergebnis:** Die Anwendung kann eine Messkonfiguration als Datei speichern und prüfen, ob ein Sensor angeschlossen ist.

**Wichtige abgedeckte Funktionale Anforderungen, die noch nicht abgedeckt wurden:** F180 füge Sensor hinzu, F190 prüfe ob Sensor angeschlossen, F210 füge Transformation hinzu, F250 speichere Messkonfiguration, F290 Einstellungen Messkonfiguration

T040 Schüler bearbeitet Aufgabe

**Testziel:** Teste die Anwendung aus Sicht eines Schülers, der eine Aufgabe bearbeiten soll.

**Vorbedingung:** Die Anwendung ist gestartet und es liegt keine Messkonfiguration vor. Die Anwendung hat Zugriff auf einen laufenden Raspberry Pi mit PhyPiDAQ. Die Sensoren die verwendet werden sollen sind ordnungsgemäß angeschlossen.

**Aktionen:** Der Schüler lädt über die Systemleiste die Messkonfiguration der Aufgabe. Danach informiert er sich über die Sensoren, Transformationen und Darstellungen über die angebotene Hilfestellungen. Als Nächstens vollendet er die Messkonfiguration.

**Reaktionen:** Nach dem Laden sieht der Schüler die Messkonfiguration in der Konfigurationsfeld. Wenn der Schüler die Hilfe aufruft, bekommt er die wesentlichen Informationen die er zur Verwendung der Anwendung braucht. Weiter sieht er, wie sich die graphische Darstellung der Messkonfiguration durch seine Aktionen verändert.

**Nachbedingung:** Die Anwendung prüft beim Laden der Messkonfiguration, ob die Sensoren angeschlossen sind.

**Ergebnis:** Der Benutzer kann Messkonfigurationen laden, diese verändern und durch das Erstellen auf Gültigkeit prüfen.

**Wichtige abgedeckte Funktionale Anforderungen, die noch nicht abgedeckt wurden:** F230 füge Darstellung, F470 Hilfe Bausteine

T050 Schüler startet Messung und speichert Ergebnisse

**Testziel:** Teste die Anwendung aus Sicht eines Schülers, der Messung starten und deren Ergebnisse speichern soll.

**Vorbedingung:** Die Anwendung ist gestartet und es ist die in T040 erstellte Messkonfiguration geladen worden.

**Aktionen:** Der Schüler startet die Messung. Während der Messung beeinflusst er die Daten durch einen Sensor. Am Ende der Messung speichert der Schüler den resultierenden Graphen und die Werte als Dateien ab.

**Reaktionen:** Nach dem Starten, sieht er, wie die Daten als Graphen dargestellt und diese durch seine Aktionen verändert werden. Die Ergebnisse der Messung sieht er nach dem Speichern als erstellte Dateien.

**Nachbedingung:** Die Anwendung prüft vor dem Starten der Messung, ob die Messkonfiguration gültig ist. Während die Messung läuft, werden die Daten verarbeitet und dargestellt unabhängig davon, ob der Schüler sie durch den Sensor beeinflusst. Die Ergebnisse sind zum Speichern verfügbar, sobald die Messung angehalten wurde.

**Ergebnis:** Der Benutzer die Messung einer gültigen Messkonfiguration starten. Die Ergebnisse kann er als Graphen oder als Werte speichern.

**Wichtige abgedeckte Funktionale Anforderungen, die noch nicht abgedeckt wurden:** F300 Starte Messung, F320 stelle Ergebnisse dar, F400 speichere Messdaten, F410 speichere Messgraph

### 8.3 Testfälle zur Handhabung von Fehlern

T200 Laden einer ungültigen Datei als Messkonfiguration

**Testziel:** Teste das Verhalten der Anwendung beim Laden von ungültigen Messkonfigurationsdateien.

**Vorbedingung:** Geöffnete Anwendung.

**Aktionen:** Der Benutzer öffnet den Dialog zum Laden einer Datei über die Systemleiste und wählt eine Datei mit ungültigem Format zum Laden aus.

**Reaktionen:** Eine Fehlermeldung wird ausgegeben.

**Nachbedingung:** Die Anwendung überprüft, ob die Datei ein gültiges Format hat und verhindert das Laden von ungültigen Formaten.

**Ergebnis:** Die Anwendung kann beim Laden von Messkonfigurationen zwischen gültigen und ungültigen Dateiformaten unterscheiden und entsprechend reagieren.

**Wichtige abgedeckte Funktionale Anforderungen, die noch nicht abgedeckt wurden:** F430 überprüfe Format bei Laden

#### T210 Erstellen einer ungültigen Messkonfiguration

**Testziel:**

**Vorbedingung:** Das Hauptfenster der Anwendung ist geöffnet. Die verwendeten Sensoren sind angeschlossen.

**Aktionen:** Der Benutzer konstruiert eine Messkonfiguration aus zwei Sensoren, einer Transformation und einer Darstellung. Allerdings verbindet er die Darstellung nicht mit dem Rest der Konstruktion und startet die Messkonfiguration.

**Reaktionen:** Die Messung wird nicht gestartet, sondern eine Fehlermeldung wird ausgegeben.

**Nachbedingung:** Die Anwendung überprüft vor dem Starten der Messkonfiguration, ob diese gültig ist.

**Ergebnis:** Die Anwendung kann beim Starten zwischen gültigen und ungültigen Messkonfigurationen unterscheiden und entsprechend reagieren.

**Wichtige abgedeckte Funktionale Anforderungen, die noch nicht abgedeckt wurden:** F310 überprüfe Messkonfiguration beim Start

#### T220 Entfernen eines Sensors bei laufender Messung

**Testziel:** Teste das Verhalten der Anwendung, wenn ein Sensor ausfällt und dessen Datenstrom abbricht.

**Vorbedingung:** Die Anwendung ist geöffnet. Alle verwendeten Sensoren sind angeschlossen und betriebsbereit. Eine gültige Messkonfiguration aus zwei Sensoren, einer Transformation und einer Darstellung wurde geladen.

**Aktionen:** Der Benutzer startet die Messung. Aus einem unbekannten Grund wird die Verbindung zu einem Sensor getrennt.

**Reaktionen:** Die Messung stoppt. Eine Fehlermeldung wird ausgegeben.



**Nachbedingung:** PhyPiDAQ verliert die Verbindung zu einem Sensor und sendet keine Daten mehr an die Anwendung. Die Anwendung stürzt nicht ab, sondern gibt eine Fehlermeldung aus.

**Ergebnis:** Die Anwendung kann mit dem ungewollten Verlust eines Datenstroms umgehen ohne abzustürzen.

**Wichtige abgedeckte Funktionale Anforderungen, die noch nicht abgedeckt wurden:** F450 Melde Abbruch des Datenstrom

## 9 Systemmodelle

Das projektierte Gesamtsystem wird aller Voraussicht nach drei Systemprozesse benötigen, die wie in Abbildung 1 dargestellt, miteinander Daten austauschen.

Einer dieser drei Systemprozesse hostet die Java Virtual Machine, auf der die zu erstellende Hauptkomponente läuft. Diese enthält die grafische Benutzeroberfläche, die im Wesentlichen die Erstellung der Werte verarbeitenden Messkonfiguration ermöglicht, und die auch deren Betrieb darstellen kann.

Zum Ansteuern und Auslesen der Sensoren auf dem Raspberry Pi wird das auf Python basierende Framework PhyPiDAQ bereitgestellt. Dieses unterstützt bereits eine ganze Reihe von Sensoren zur Messung diverser physikalischer Größen der Mechanik, Thermodynamik und Elektrodynamik. Da der Python-Interpreter im Allgemeinen nicht im selben Systemprozess wie die Java Virtual Machine ausgeführt werden kann, wird für diesen ein zweiter Systemprozess benötigt.

Um Ressourcenknappheit auf dem Raspberry Pi aus dem Weg zu gehen, aber auch zur Vereinfachung von Entwicklung und Demonstration, soll die Hauptkomponente nicht zwingend ebenso auf dem Raspberry Pi ausgeführt werden müssen.

Falls denn nun die Hauptkomponente tatsächlich auf einem anderen Rechner ausgeführt werden sollte, so muss sie mit dem Python-Interpreter hardware- bzw. plattformübergreifend Daten austauschen können. Bisher stellt PhyPiDAQ hierfür jedoch keine Funktionalität zur Verfügung.

Zur Realisierung der hardwareübergreifenden Kommunikation sind die folgenden zwei Lösungsvarianten denkbar:

- PhyPiDAQ wird um eine Kommunikationsschnittstelle erweitert, beispielsweise zur Kommunikation über eine Netzwerkschnittstelle wie z. B. WLAN oder zur Kommunikation über eine Peripheralschnittstelle wie z. B. USB.

Da bei letzterer Variante die Utility-Softwarekomponente zusätzlich auch noch eine Überwachungsfunktion bezüglich PhyPiDAQ übernehmen und dieses im Falle von Abstürzen und Nicht-Responsivität erneut ausführen kann, ist letztere Lösungsvariante die bevorzugte Lösungsvariante zur Realisierung der optionalen hardwareübergreifenden Kommunikation.

Abbildung 1 stellt die Verteilung der Systemkomponenten auf zwei Rechnern mittels der gewählten zweiten Lösungsvariante dar, und benennt die notwendige Utility-Komponente mit „Measurement-Server“.

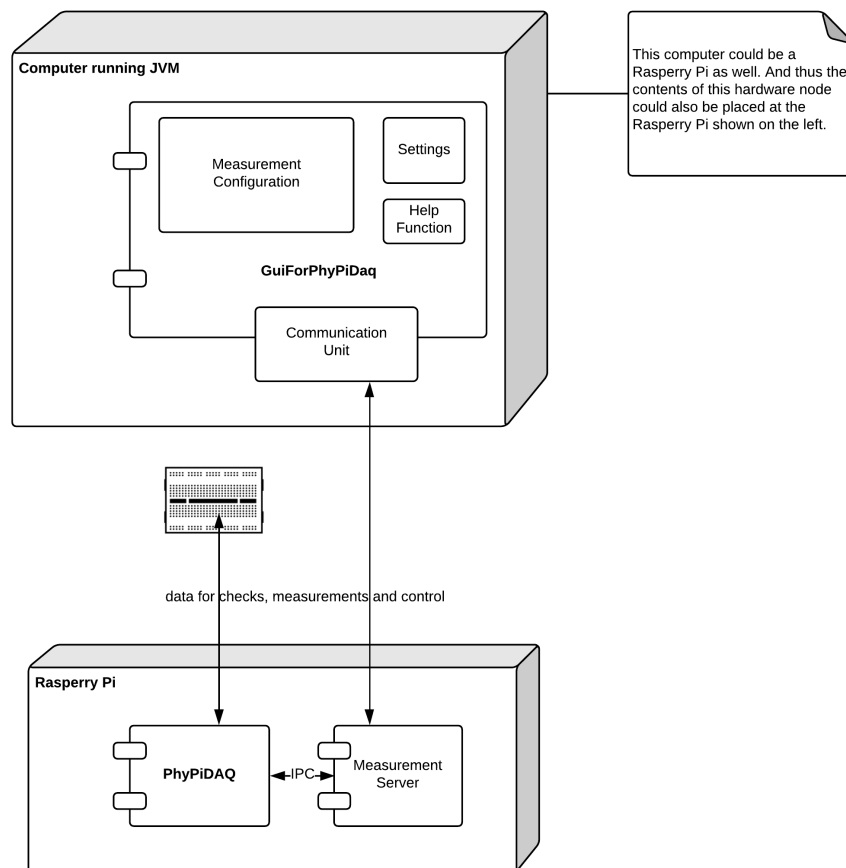


Abbildung 1: UML-Deployment-Diagramm

## **10 Benutzungsoberfläche**

### **10.1 Ziel der Benutzeroberfläche**

Das Ziel der Benutzeroberfläche ist es dem Anwender eine intuitive Benutzung des Programmes zu ermöglichen. Da das Programm im Schulbetrieb eingesetzt werden soll ist eine gute Verständlichkeit wichtig. So soll auch eine lange Einarbeitungszeit für den Anwender vermieden werden. Dabei soll keine Funktionalität verloren gehen.

### **10.2 Generell**

Um die Ziele zu erfüllen wird das Programm über eine Grafische Benutzeroberfläche (kurz „GUI“) bedient. Der Anwender soll in der Lage sein bereits vorhandene Computer und Mobilgerät Kenntnisse zu nutzen.

### **10.3 Eingabegeräte**

Die Grafische Benutzeroberfläche soll Maus und Tastatureingaben unterstützen. Außerdem soll eine Drag-and-Drop Eingabe möglich sein. Die Verwendung anderer Eingabegeräte ist nicht vorgesehen.

### **10.4 Überblick**

Abbildung 2 zeigt den Aufbau einer möglichen Grafische Benutzeroberfläche. Dabei geben die Zahlen in den roten Kästchen, rechts von dem Programmfenster, die logische Unterteilung an.

Das Design der Einzelnen Elemente kann sich im Verlauf der Anwendungsentwicklung ändern. Die Grundlegenden Bausteine der Grafische Benutzeroberfläche werden sich in ihrer Aufgabe nicht ändern. Zu ihnen gehören:

- Sensoren
- Transformationen
- Darstellungen

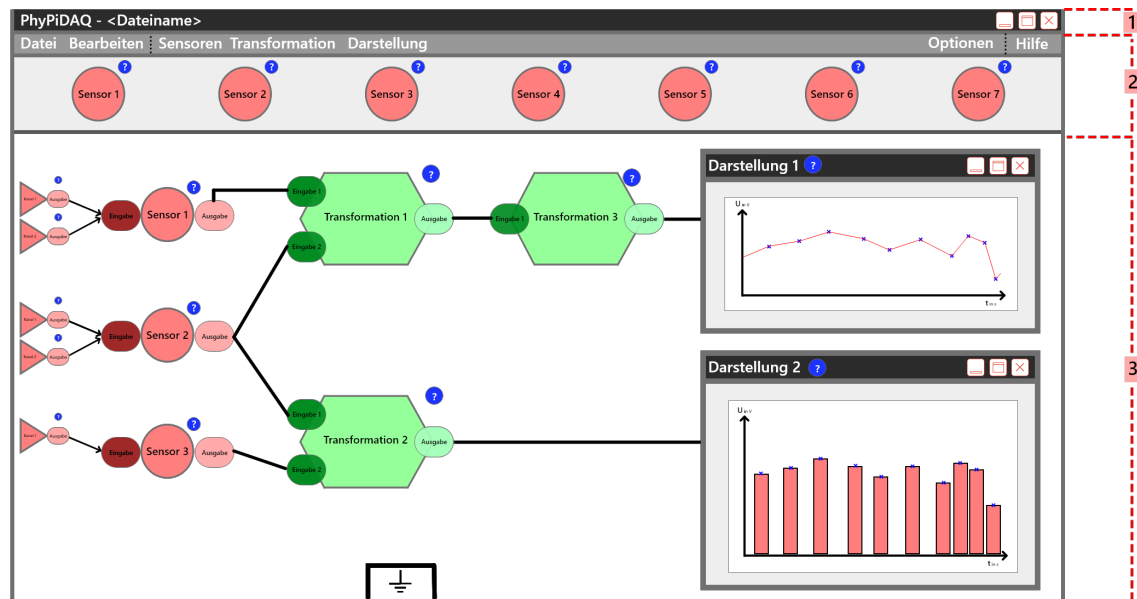


Abbildung 2: Der Grundlegende Aufbau der Hauptbenutzeroberfläche

Die zu dem jeweiligen logischen Teil gehörenden Elemente sollen im folgenden erklärt werden.

## 10.5 Die einzelnen Teile

### 10.5.1 Systemmenüleiste

In der Systemmenüleiste befinden sich Funktionen, die Einfluss auf die gesamte Grafische Benutzeroberfläche haben.



Der Name des Programmes wird hier angezeigt. Rechts daneben steht, sofern vorhanden, der Name der Datei, die gerade geöffnet ist.



Das "Maximieren"-Symbol vergrößert das Programmfenster auf die maximale Größe. Die Größe ist von der Benutzungs-umgebung abhängig.



Das "Minimieren"-Symbol blendet das Programmfenster aus. Es ist weiterhin geöffnet und wieder aufrufbar.



Das "Schließen"-Symbol beendet die Anwendung. Vor dem Beenden findet eine Abfrage statt, ob der Anwender eventuell vorgenommene Änderungen speichern möchte.



## 10.5.2 Auswahl

### Datei

Der Benutzer hat die Möglichkeit Dateien zu bearbeiten. Die wichtigsten Funktionen sind:

- Das anlegen einer neuen Datei
- Das Speichern der aktuellen Datei
- Das Öffnen einer bereits erstellten Datei

### Bearbeiten

Der Benutzer hat die Möglichkeit den Inhalt der aktuell geöffneten Datei zu bearbeiten. Die wichtigsten Funktionen sind:

- Das Kopieren eines ausgewählten Objektes
- Das Einfügen eines gespeicherten Objektes
- Das Anpassen eines ausgewählten Elements. Diese Einstellungsmöglichkeiten sind von dem Objekt abhängig.

### Sensoren

Der Benutzer kann sich eine Auswahl von Sensoren anzeigen lassen. Die Sensoren können anschließend in das Konfigurationsfeld eingefügt werden. Die angezeigte Auswahl ist unabhängig von den angeschlossenen Sensoren

### Transformation

Der Benutzer kann sich eine Auswahl von Transformationen anzeigen lassen. Die Transformationen können anschließend per Drag-and-Drop in das Konfigurationsfeld gezogen werden. Die Auswahl kann unter Optionen erweitert werden.

### Darstellung

Der Benutzer kann sich eine Auswahl von Darstellungen anzeigen lassen. Die Darstellungen können anschließend in das Konfigurationsfeld eingefügt werden. Die Auswahl kann erweitert werden.

### Optionen

Der Benutzer kann sich eine Auswahl an Optionen anzeigen lassen. Beispiele hierfür sind

- Das Ändern von angezeigten Farben
- Das Erweitern der Transformations- und Darstellungsauswahl

### Hilfe

Der Benutzer kann sich eine allgemeine Hilfe anzeigen lassen. In der Hilfe werden die Funktionen der Anwendung erklärt.

Im Laufe der Entwicklung kann es sich als sinnvoll herausstellen weitere Optionen einzufügen bzw. bestehende Optionen zusammenzufügen.

### 10.5.3 Konfigurationsfeld



Das "Informations"-Symbol zeigt nach einem Click weiterführende Informationen zu dem Element an, zu dem es gehört. So würde beispielsweise bei einer Transformation die Funktion angezeigt werden, die sie realisiert.



Repräsentation eines Sensors, der in PhyPiDAQ erkannt werden kann. Voraussetzung hierfür ist die Existenz einer Konfigurationsdatei.



Die zu einem Sensor gehörende Eingabe. Die Eingabe wird nur angezeigt, wenn auch die Kanäle angezeigt werden.



Die zu einem Sensor gehörende Ausgabe. Jeder Sensor hat genau eine Ausgabe.



Der zu einem Sensor gehörende Kanal. Die Anzahl an angezeigten Kanälen hängt von dem Sensor ab. In den Optionen kann eingestellt werden, ob die Kanäle angezeigt werden.



Repräsentation einer Transformation. Abhängig von der Anzahl an Eingängen der Transformation besitzt die GUI dementsprechend viele Eingänge. Jedem Eingang können durch Einfügen von einer Verbindung Daten übergeben werden.



Die Eingabe einer Transformation. Die Anzahl der verfügbaren Eingaben hängt von der Transformation ab.



Die Ausgabe einer Transformation. Jede Transformation hat genau eine Ausgabe.



Eine Verbindung zwischen zwei Elementen. Sie besitzt eine Richtung. Daten "fließen" also nicht in beide Richtungen. Sie wird durch cklicken auf zwei Elemente erstellt. Dabei ist das Element welches zuerst geclickt wurde der Anfang der Verbindung.



## 10.6 Erweiterungsmöglichkeiten

### 10.6.1 Startbildschirm

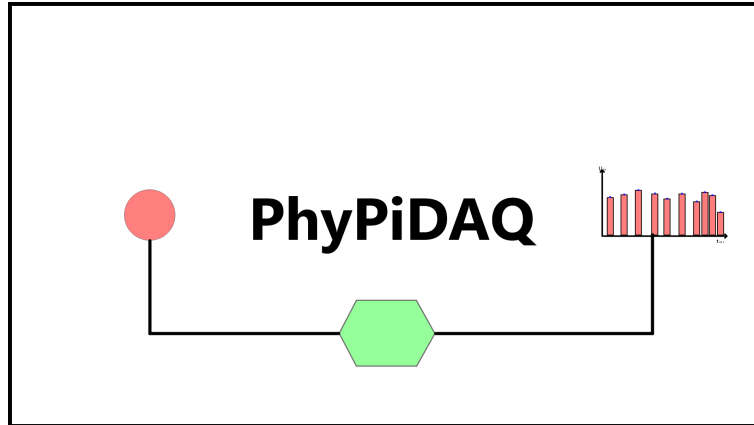


Abbildung 3: Ein Beispiel für einen Startbildschirm

Eine Erweiterungsmöglichkeit wäre das Einfügen eines Startbildschirmes vor dem Öffnen der Anwendung. Dieser ist in Abbildung 3 angedeutet.

## 11 Spezielle Anforderungen an die Entwicklungsumgebung

Für die Umsetzung des Projekts ist Zugang zu funktionierender Hardware (konkret: mindestens ein Raspberry Pi, möglichst mehrere Sensoren, Peripheriegeräte) notwendig, um die Funktionalität des Produkts sicherzustellen.

Ansonsten muss gegebenenfalls auf simulierte Messdaten zurückgegriffen werden.

Das Projekt ist von PhyPiDAQ abhängig, das heißt Funktionen welche in PhyPiDAQ fehlerhaft sind können möglicherweise nicht korrekt implementiert werden. PhyPiDAQ sollte beispielsweise im Fall eines Programmfehlers eindeutige Fehlercodes produzieren, damit der Benutzer aussagekräftig über das Problem informiert werden kann.

## 12 Zeit- und Ressourcenplanung

### 12.1 Projektphasen

Phase	Verantwortlicher	Zeitraum	Kolloquium
Pflichtenheft	Jan Küblbeck	KW 20–22	04.06.2019
Entwurf	Leon Huck	KW 23–26	02.07.2019
Implementierung	Stefan Geretschläger	KW 27–29, 31	(tbd)
Klausurenphase	—	KW 30, 32	—
Qualitätssicherung	David Gawron	KW 33–35	03.09.2019
Abnahme	—	KW 36	—
Abschlussprüfung	Linus Ruhnke	KW 37/38	(tbd)

### 12.2 Risikomanagement

In jeder Phase treten unterschiedliche Risiken auf.

Das Ausfallen eines Teammitglieds durch Krankheit oder anderen Gründen ist ein bedeutendes Risiko. In diesem Fall muss die zusätzliche Arbeit gerecht auf die anderen Teilnehmer verteilt werden. Eventuell können dadurch einzelne Kriterien und Anforderungen nicht mehr zufriedenstellend erfüllt werden.

## 13 Glossar

**Bausteinprototyp** Baustein, von dem eine Kopie angelegt wird, wenn der Benutzer ein neues Baustein-Exemplar einem Entwurf hinzufügen möchte.

**Benutzeroberfläche** Steht für die Oberfläche, die der Benutzer verwendet um die Anwendung zu bedienen.

**Darstellung** Gibt Daten aus eingehendem Datenstrom graphisch wieder.

**Datenschutz-Grundverordnung** kurz DSGVO, Verordnung der Europäischen Union vom 25. Mai 2018.

**Drag-and-Drop** Methode, um mit graphischen Benutzeroberflächen zu interagieren. Dabei wird ein Objekt erst mit der Maus festgehalten und an einen anderen Ort gezogen. Durch das Lösen der Maustaste wird das Objekt platziert.

**Grafische Benutzeroberfläche** Eine Oberfläche, die eine Anwendung durch grafische Symbole bedienbar macht.

**GUI** engl. Graphical User Interface; siehe: Grafische Benutzeroberfläche.

**Java Virtual Machine** Die Java Virtual Machine (JVM) ist eine Plattform für die Ausführung von Java-Software, die von der Firma Oracle für alle gängigen Betriebssysteme bereitgestellt wird.

**Konfigurationsbaustein** Teil einer Messkonfiguration, der eine Teilaufgabe bestimmten Typs erfüllen kann. Es gibt Sensorbausteine, Konfigurationsbausteine und Darstellungsbausteine. Liegt am Ausgang eines Bausteins ein Wert an, so kann dieser an den Eingang eines nachgelagerten Bausteins weitergeleitet werden.

**Konfigurationsdatei** Können das Messverhalten anpassen, beispielsweise die Anzahl der Messungen pro Zeiteinheit. Für jeden Sensor gibt es eine eigene Konfigurationsdatei.

**Messdaten** Daten, welche die Anwendung von einem Sensor (über PhyPiDAQ-Schnittstelle) oder direkt aus einer Datei erhält.

**Messkonfiguration** Gerichteter zyklenfreier Graph mit Knoten vom Typ Sensor, Transformation oder Darstellung. Hierbei ist zu beachten, dass Sensoren keine Eingangskanten und Darstellungen keine Ausgangskanten haben dürfen.

**Messlauf** Zeitabschnitt, in dem zu definierten Zeitpunkten an allen Bausteinen eines Entwurfs sukzessive die Werte an allen Ausgängen und Eingängen bestimmt werden.

**Musskriterien** Werden zusammen mit Soll- und Wunschkriterien bei der Abnahme eines Softwareprodukts überprüft und haben während der Entwicklung höchste Priorität. Dass ein Musskriterium in den nachfolgenden Projektphasen nicht umgesetzt wird, ist nur dann zulässig, falls unerwartet unausweichliche Probleme bei der Umsetzung auftreten. In diesem Fall ist es erforderlich, dass diese Probleme sehr genau dokumentiert werden.

**Open Source** Software, deren Quelltext öffentlich eingesehen werden kann, wird als „Open Source“ bzw. „quelloffen“ bezeichnet.

**OSL<sup>2</sup>** Open-Source-Lehrsoftware-Labor, <https://formal.iti.kit.edu/projects/osls1/?lang=de>.

**PhyPiDAQ** Siehe Abschnitt 4.3 „PhyPiDAQ“.

**Python 3** Python ist eine universelle Programmiersprache. Sie ist die offizielle Programmiersprache des Raspberry Pi und wurde deswegen zur Programmierung von PhyPiDAQ verwendet.

**Raspberry Pi** Der Raspberry Pi ist ein Einplatinencomputer. In diesem Projekt dient der Raspberry Pi als Hardwareplattform, um unterschiedliche Sensoren zu verbinden und ihre Daten auszulesen.

**Science Labs** Ein Science Lab ist ein Arbeitsplatz, welcher Schülern ermöglicht wissenschaftliche Forschungen unter kontrollierten Bedingungen durchzuführen.

**Sensor** Ein Sensor erfasst Messwerte physikalischer Größen. In einer Messkonfiguration stellen sie die Eingangspunkte dar.

**Sollkriterien** Werden zusammen mit Muss- und Wunschkriterien bei der Abnahme eines Softwareprodukts überprüft und haben während der Entwicklung mittlere Priorität. Falls ein Sollkriterium umgesetzt werden kann, dann muss es nach Möglichkeit auch realisiert werden. Falls ein Sollkriterium in den nachfolgenden Projektphasen nicht umgesetzt werden kann, so muss dies dokumentiert und begründet werden.

**Transformation** Erzeugen eines Datenstroms unter Verwendung von einem oder mehreren bereits vorhandenen Datenströmen. Eine Transformation kann durch mathematische Funktion realisiert werden.

**UI** engl. User Interface; siehe: Benutzeroberfläche.

**Wunschkriterien** Werden zusammen mit Muss- und Sollkriterien bei der Abnahme eines Softwareprodukts überprüft und haben während der Entwicklung niedrige Priorität. Je nach Ressourcenlage können sie nach Bearbeitung aller Muss- und Kannkriterien umgesetzt werden. Falls ein Wunschkriterium nicht umgesetzt werden kann, so muss dies nicht begründet werden.