Testbericht der Qualitätssicherungsphase

Definition und Durchführung von Messwertverarbeitung für den Physikunterricht auf Basis eines Raspberry Pis

Version 0.5.0

David Gawron Stefan Geretschläger Leon Huck
Jan Küblbeck Linus Ruhnke

1. September 2019

Inhaltsverzeichnis

1 Ziel des Testberichts

Das Ziel des Testberichtes ist es dem Leser einen Überblick über die verwendeten Testverfahren zu geben und die während der Qualitätssicherungsphase entdeckten Fehler zu dokumentieren. Die Qualitätssicherungsphase hat das Ziel, möglichst viele Fehler aufzudecken, diese zu korregieren und zu dokumentieren. Zusätzlich soll das unbemerkte Wiederauftreten bereits gefundener Fehler durch Regressionstests verhindert werden. Dabei werden die Funktionalitäten und deren Qualitäten getestet.

1.1 Bedingungsüberdeckung

Wir streben eine mehrfache Bedingungsüberdeckung an. Dadurch werden Zweig- ,Anweisungs-, einfache und minimal-mehrfache Bedingungsüberdeckung subsumiert. Eine einfache Bedingungsüberdeckung ist subsumiert nicht einmal die Anweisungsüberdeckung und ist somit ungeeignet. Eine minimal-mehrfache Bedingungsüberdeckung wäre ein guter Kompromiss zwischen Aufwand und Nutzen, allerdings verwendet unser Plug-In *EclEmma* für *JaCoCo* standardmäßig mehrfache Bedingungsüberdeckung. Außerdem ist die Anzahl an Bedingungen in unserer Anwendung noch überschaubar. Eine Pfadüberdeckung streben wir nicht an, da dessen Aufwand mit 2 hoch k skaliert, wobei k die Anzahl an Anweisungen ist.

2 Planung der Qualitätssicherungsphase

Die Qualitätssicherungsphase wird in drei Meilensteine aufgeteilt, siehe dazu Abbildung 1. Der erste Meilenstein wird erfüllt, wenn das Modul Model der Anwendung eine hohe Testüberdeckung erreicht. Dabei sollen alle Tests automatisch mit J-Unit ablaufen. Das Model ist die Basis, die alle anderen Module benutzen und auch diese verbindet. Deshalb ist die erste Priorität eine getestetes Modul, um komplexe Folgefehler für die anderen Module zu verhindern.

Im zweiten Meilenstein werden alle anderen Module, außer der GUI, getestet. Auch hier erfolgt das Testen über automatische J-Unit Tests.

Die GUI ist ein Sonderfall beim Testen, da diese nur sehr begrenzt mit automatischen Tests getestet werden kann. Deshalb wird diese im dritten Meilenstein getestet. Der Dritte Meilenstein umfasst die GUI und auch das Testen der gesamten Anwendung. Die GUI wird hauptsächlich über Klickstrecken getestet. Die gesamte Anwendung wird durch Testszenarien aus dem Pflichtenheft geprüft. Weiter werden Qualitätsanforderungen der Anwendung durch verschiedene Tests geprüft. Schließlich wird die Leistung und auch die Hardware für die Anwendung getestet.

In Abbildung 2 ist der tatsächliche Verlauf der Qualitätssicherungsphase abgebildet. Man sieht sofort, dass sich der Schwerpunkt der Arbeiten auf den 31.8. verschoben hat. Dies liegt daran, dass das Testen und Beheben von Fehlern der anderen Module auch weiterhin im dritten Meilenstein stark präsent war. Weiter sieht man, dass die planmäßige Testabdeckung am Ende des zweiten geplanten Meilenstein nicht erfüllt war.

TODO Meilenstein Drei Testabdeckung

Testplan für die Qualitätssicherung	11. bis 16. August	16.08.19	20.08.19	21.08.19	22.08.19	bis	25.08.19	26.08.19	27.08.19	28.08.19	29.08.19	30.08.19	31.08.19
Spalteninformationen	MS 0: Altlasten	**Meilenstein 1**: Model 90 % Abdeckung mit J-Unit			**Meilenstein 2:** Backend, Cache, Controller und Fileservice 90% Abdeckung mit J-Unit			Puffer	**Meilenstein 3**: GUI- Abdeckung, Belastungs- und Integrationstests				Puffer
		Modul-Abdeckung Sollwert in %			**Modul-Abdeckung Sollwert in %**				**Modul-Abdeckung Sollwert in %**				
Controller				0	0-30	30-60	95					95	
FileService & Main				0	0-30	30-60	95					95	
GUI				0					0-20	20-40	40-65	85	
Klickstrecken													
Monkey Testing & Hallway Testing													
Laufzeit & Speicherverbrauch													
Qualitätsanforderungen													
TestSzenarien													
Model			65	90			90					95	
Backend				0	0-30	30-60	90					95	
Cache				0	0-30	30-60	90					95	
Hardwaretest Sensoren													

Abbildung 1: Der Sollplan für die Qualtätssicherungsphase.

Testplan für die Qualitätssicherung	11. bis 16. August	16.08.19	20.08.19	21.08.19	22.08.19	bis	25.08.19	26.08.19	27.08.19	28.08.19	29.08.19	30.08.19	31.08.19
Spalteninformationen	MS 0: Altlasten	**Meilenstein 1**: Model 90 % Abdeckung mit J-Unit			**Meilenstein 2**: Backend, Cache, Controller und Fileservice 90% Abdeckung mit J-Unit			Puffer	**Meilenstein 3**: GUI- Abdeckung, Belastungs- und Integrationstests				Puffer
Controller		Modul-Abdeckung Istwert in %		0			0	Modul-Abdeckung Istwert in %					TODO
FileService & Main				0			25						TODO
GUI				0									TODO
Klickstrecken													
Monkey Testing & Hallway Testing													
Laufzeit & Speicherverbrauch													
Qualitätsanforderungen													
TestSzenarien													
Model				82			75						TODO
Backend				0			50						TODO
Cache				0			75						TODO
Hardwaretest Sensoren													

Abbildung 2: Der Istplan für die Qualtätssicherungsphase.

3 Gefundene Fehler und deren Regressionstests

Dieses Kapitel umfasst die Regressionstests für gefundene und behobene Fehler. Die Tests sind nach Modul und Klassen strukturiert. Jeder Regressionstest verweist auf ein Issue der verwendeten Bugtracking-Software (hier GitHub).

3.1 Übersicht aller Issues

In der Tabelle 1 wird angezeigt, wo ein Issue aufgetreten ist, und was für eine Kategorie es hat. Das Issue wird dabei durch seine Nummer repräsentiert. Zu den roten Issues gibt es keine Regressionstests, das diese nicht behoben wurden. Bei den Issues in Klammern handelt es sich um keine eigentlichen Fehler, sondern um Verbesserungen, Löschen von Ungenutztem, Fragen usw... .

Art des Issue vs Fundort	**Null Pointer**	**Index out Of Bounds**	**Path related**	**fehlerhafte Funktion**	**Sonstige**
Backend					(29), (34), 36, (38)
Cache	22, 65				
Controller					(37)
Gui	27		25	15, 58, 62	
Model	7, 8, 9, 10, 12, 13, 19, 51, 59, 60	11, 18		14, 21, 23, 35	(32),(33), (39), 53
Fileservice und Main	47		57	50, 54	(30),(31),(41),(42),(45
Gesamtzahl	13	2	2	9	4 + (11)

Tabelle 1: Übersicht über alle Issues.

Falls ein Problem hauptsächlich mit der Gui zu tun hat und ein automatischer Regressionstest nur schwer oder gar nicht realisierbar ist, wird für ein entsprechendes Issue eine Klickstrecke mit erwartetem Ergebnis geschrieben.

3.2 Model

3.2.1 Measurement Configuration

Issue Nr.7 in der Methode getInChan

Fehlersymptom: Unbehandelte NullPointer Exception bei Eingabe einer ungültigen Id.

Fehlerursache: Prüfen nach NullPointer Exception fehlt.

Fehlerbehebung: Eine Null Prüfung wurde implementiert.

Verantwortlicher: David Gawron

Issue Nr.8 in der Methode getOutChan

Fehlersymptom: Unbehandelte NullPointer Exception bei Eingabe einer ungültigen Id.

Fehlerursache: Prüfen nach NullPointer Exception fehlt.

Fehlerbehebung: Eine Null Prüfung wurde implementiert.

Verantwortlicher: David Gawron

Issue Nr.9 in der Methode addConnection

Fehlersymptom: Unbehandelte NullPointer Exception bei Eingabe einer ungültigen Id.

Fehlerursache: Prüfen nach NullPointer Exception fehlt.

Fehlerbehebung: Eine Null Prüfung wurde implementiert.

Verantwortlicher: David Gawron

Issue Nr.10 in der Methode removeConnection

Fehlersymptom: Unbehandelte NullPointer Exception bei Eingabe einer ungültigen Id.

Fehlerursache: Prüfen nach NullPointer Exception fehlt.

Fehlerbehebung: Eine Null Prüfung wurde implementiert.

Verantwortlicher: David Gawron

Issue Nr.11 in der Methode createInChannelList

Fehlersymptom: Auftreten einer Index Out Of Bounds Exception.

Fehlerursache: Eine Prüfung, ob der Index groß genug ist, fehlt.

Fehlerbehebung: Der Fehler wird abgefangen durch einen Vergleich der Anzahl der InChannel zwischen yaml-File und Prototypblock.

Verantwortlicher: David Gawron

Issue Nr.12 in der Methode getOutChanPosi

Fehlersymptom: NullPointer Exception beim Laden einer Messkonfiguration mit ungültigen Block Id.

Fehlerursache: Prüfen nach NullPointer Exception fehlt.

Fehlerbehebung: Es wird nach Null geprüft. Dann ergab sich eine Folgefehler, der sich in der Methode createLoadedConnections als eine Index Out Of Bounds Exception äußerte. Durch das Implementieren einer Methode checkBlockInitId, die prüft, ob eine geladene Id auch gültig ist, wurde der Folgefehler behoben.

Verantwortlicher: David Gawron

Issue Nr.13 in der Methode createInChannelList

Fehlersymptom: NullPointer Exception bei ungültiger Messkonfiguration mit einer fehlenden BlockChannelliste.

Fehlerursache: Prüfen nach NullPointer Exception fehlt.

Fehlerbehebung: Eine Prüfung nach Null wurde hinzugefügt.

Verantwortlicher: David Gawron

Issue Nr.14 in der Methode loadConfig

Fehlersymptom: Fehlerhaftes Verhalten beim Laden der fehlerhaften Messkonfiguration 4, die einen Block mit sich selber verbindet.

Fehlerursache: Überprüfen der Verbindung fehlt.

Fehlerbehebung: Die Überprüfung der Verbindung wurde implementiert.

Verantwortlicher: David Gawron

Issue Nr.18 in der Methode removeBlock

Fehlersymptom: Der Versuch einen nicht existierenden Block zu entfernen, resultiert in eines Index Out Of Bounds Exception.

Fehlerursache: Der Index wurde nicht geprüft.

Fehlerbehebung: Eine Prüfung des Indexes wurde hinzugefügt. Außerdem wurde der Rückgabewert der Methode von void zu boolean geändert.

Verantwortlicher: David Gawron

Issue Nr.19 in der Methode removeBlock

Fehlersymptom: Der Versuch eine Konfiguration ohne eine Liste von Block Ids zu laden, führt zu einer Null Pointer Exception.

Fehlerursache: Es wurde nicht nach Null geprüft.

Fehlerbehebung: Die betreffende Zeile wurde in einen schon existierenden Null-Check verschoben.

Verantwortlicher: David Gawron

Issue Nr.23 in der Methode loadConfig

Fehlersymptom: Der Versuch eine Konfiguration mit einem Splitter mit vertauschten Verbindungstupel führt zu einem ungewollten Fehlschlag des Ladens.

Fehlerursache: Die private Methode createInChannelList, die von loadConfig benutzt wird, funktionierte nicht korrekt.

Fehlerbehebung: Die private Methode createInChannelList wurde reimplementiert.

Verantwortlicher: David Gawron

Fehler Nr.35 in der Methode getInitId

Fehlersymptom: Die Methode funktionierte nicht richtig und gab immer NULL zurück.

Fehlerursache: Der Zugriff auf die Blöcke in der Hasmap der Konfigurationsblöcke schlägt fehl.

Fehlerbehebung: Die KonfigurationsId wird nun über die Blockliste der Messkonfiguration geholt.

Verantwortlicher: David Gawron

3.2.2 Sensor

Issue Nr.51 in der Methode constructBuildingBlocks

Fehlersymptom: Die Methode liefert bei Übergabe von null oder einer leeren Map eine NullPointerException.

Fehlerursache: Fehlende Überprüfung der Übergabewerte auf die notwendigen Argumente

Fehlerbehebung: constructBuildingBlock() überprüft, ob die Map null oder leer ist.

Verantwortlicher: Linus Ruhnke

Issue Nr.59 in der Methode processKvPair

Fehlersymptom: Die Methode liefert bei falschen Übergabewerten eine NullPointerException.

Fehlerursache: Fehlende Überprüfung der Übergabewerte auf die notwendigen Argumente

Fehlerbehebung: processKvPair() überprüft, ob die Map der Bausteine die notwendigen Argumente zum Aufbauen des Bausteins enthält.

Verantwortlicher: Linus Ruhnke

Issue Nr.60 in der Methode initialiseModel

Fehlersymptom: Die Methode überprüft Bausteine nicht auf null und kann somit eine NullPointerException liefern.

Fehlerursache: Fehlende Überprüfung der Bausteine auf die null.

Fehlerbehebung: Die Methode überprüft, ob die Bausteine null sind.

Verantwortlicher: Linus Ruhnke

3.2.3 BuildingBlockDirectory

Issue Nr.21 in der Methode addConfigBlock

Fehlersymptom: Konfigurationsbausteine konnten mehrfach mit dem gleichen Key zu der Liste hinzugefügt werden. Alte Einträge mit dem gleichen Key wurden überschrieben.

Fehlerursache: Standart-Funktion einer HashMap.

Fehlerbehebung: Konfigurationsbausteine können nicht mehrfach mit dem gleichen Key hinzugefügt werden. Alte Einträge werden nicht mehr überschrieben.

Verantwortlicher: Linus Ruhnke

3.3 Cache

Issue Nr.65 in der Klasse Cache

Fehlersymptom: NullPointerException, wenn ein Messlauf mit inkorrekter Messkonfiguration gestartet wird.

Fehlerursache: Keine Überprüfung auf null in der Methode latestMSetFromCollection(...).

Fehlerbehebung: Methode gibt nun anstatt null eine leer initialisierte Instanz von MSet zurück.

Verantwortlicher: Jan Küblbeck

3.4 Backend

Issue Nr.36 in der Klasse PickupPointForBackendAgents

Fehlersymptom: Es war nicht möglich, gleichzeitig eine simulierte und eine tatsächliche Backend-Verbindung aufzubauen.

Fehlerursache: Durch PickupPointForBackendAgents wurde immer nur entweder eine echte oder eine simulierte Verbindung initialisiert.

Fehlerbehebung: Durch separate Methoden können nun eine echte und eine simulierte Verbindung separat verwendet werden.

Verantwortlicher: Jan Küblbeck

3.5 Controller

3.6 Fileservice und Main

Issue Nr.54 in der Methode saveIntoYaml

Fehlersymptom: Beim Speichern einer Konfiguration als .yaml-Datei, werden einzelne ungewollte Anführungszeichen hinuzugefügt.

Fehlerursache: SnakeYaml hat wohl Schwierigkeiten bei unseren Einstellungen mit eckigen Klammern umzugehen. Daraus resultieren die einzelnen Anführungszeichen.

Fehlerbehebung: Es wurde eine private Methode im FileService angelegt, die die neu gespeicherte Datei erneut öffnet und alle einzelnen Anführungszeichen entfernt.

Verantwortlicher: David Gawron

Issue Nr.47 in der Methode readOutAllYamll

Fehlersymptom: Bei einem Fehler beim Lesen des Inhalt einer Yaml-Datei wird null zurückgegeben.

Fehlerursache: Implementierung der Methode.

Fehlerbehebung: Bei einem Fehler beim Lesen wird eine leere Map zurückgegeben.

Verantwortlicher: Linus Ruhnke

Issue Nr.50 in der Methode testWriteIntoYaml

Fehlersymptom: Die Methode löschte eine durch den Test erzeugte Methode bei einem erfolgreichen Durchlauf nicht, was zur Folge hatte, dass beim nächsten Test der Test fehl schlug.

Fehlerursache: Unbekannt.

Fehlerbehebung: Die Methode löscht vor dem Test die Datei aus dem Verzeichniss.

Verantwortlicher: Linus Ruhnke

Issue Nr.57 in der Methode readFromYaml

Fehlersymptom: Falls ein angegebener Pfad nicht existiert wird eine Exception von SnakeYaml übergeben.

Fehlerursache: Übergebener Pfad existiert nicht, aber wird nicht abgefangen.

Fehlerbehebung: Vor dem Lesen wird überprüft, ob eine Ressource in dem Verzeichnis mit dem Pfad existiert.

Verantwortlicher: Linus Ruhnke

3.7 GUI

Issue Nr.27 in der Methode pushData

Fehlersymptom: NullPointerException tritt auf, wenn ein Messlauf begonnen wird ohne zuvor die GUI zu starten.

Fehlerursache: Es wurde in der Klasse DataVisualisation auf ein Textfeld zugegriffen, welches nicht initialisiert wurde.

Fehlerbehebung: Das Textfeld wird nun zuerst auf null überprüft.

Verantwortlicher: Jan Küblbeck

Issue Nr.62 in der Klasse MeasurementButtonField

Fehlersymptom: Durch das klicken des "Reset"-Knopfs wurden die angezeigten Daten des vorhergegangenen Messlaufs nicht aus der Anwendung entfernt.

Fehlerursache: Fehlender Code in der Klasse MeasurementButtonField.

Fehlerbehebung: Code hinzugefügt, um die Daten zu entfernen.

Verantwortlicher: Jan Küblbeck

Issue Nr.66 in der Gui

Fehlersymptom: Lädt man eine gültige Konfiguration, macht sie ungültig und checkt sie dann ohne Erfolg, bleibt die alte, gültige Konfiguration im Model geladen und startbar, obwohl in der Gui die ungültige zu sehen ist.

Fehlerursache: Die Methode deleteMeasurementConfiguration in der MeasurementConfiguration funktionierte nicht richtig.

Fehlerbehebung: Die Methode wurde entsprechend angepasst.

Klickstrecke: Anwendung wird geöffnet → lade gültige Konfiguration → mache sie ungültig → drücke den Check-Knopf → klicke OK → klicke run

erwartetes Ergebnis: Die Messung wird nicht gestartet und es werden keine Daten angezeigt.

Verantwortlicher: David Gawron

4 Testen der GUI

4.1 Testen der GUI durch Klickstrecken

4.1.1 Öffnen der Systemmenüs und dessen Funktionen

In dieser Klickstrecke werden die Funktionen des Systemleistenmenüs getestet unter der Vorbedinung, dass die Anwendung geöffnet ist. In Klickstrecke Nr.1 werden mehrere Bausteine bei der Initialisierung der Anwendung geladen und bei Klickstrecke Nr.2 sind keine Bausteine bei dem angegeben Pfad initialisiert worden.

4.1.2 Erstellen, Speichern, Checken und Laden einer Messkonfiguration

In dieser Klickstrecke wird die Anwendung anhand ihrer Funktion rund um das Erstellen, Speichern und Laden einer Messkonfiguration getestet. Als Vorbedingung ist hier die geöffnete Anwendung mit einer leeren Messkonfiguration gegeben. Die Klickstrecke und deren Ergebnisse sind in Tabelle 3 zu sehen. Dabei besteht Konfiguration A aus einem BMP180 Sensor-Baustein, einer textuellen Repräsentation für einen Kanal und der korrekten Verbindung dazwischen. Konfiguration B besteht aus dem selben Bausteinen wie Konfiguration A, aber die Verbindung fehlt.

4.1.3 Kontrolle des Messlaufs

Durch diese Klickstrecke (Tabelle 4) wird überprüft, ob das Starten, Pausieren, Fortsetzen und Beenden eines Messlaufs und das Speichern von Messdaten korrekt funktioniert. Vorbedingung ist, dass eine korrekte Messkonfiguration bereits geladen ist (siehe vorherige Klickstrecke)

4.2 Monkey Testing

Monkey Testing bedeutet die Durchführung von zufälligen oder pseudozufälligen Eingaben mit dem Ziel, Fehler in einer Anwendung zu provozieren.

Auch in dieser Qualitätssicherungsphase wurde Monkey Testing durchgeführt. Dabei wurden mehrere Fehler gefunden und im Issue-Tracker dokumentiert.

Nr.	Aktions- und Klickstrecke	Erwartetes Ergbenis	Bewertung tatsächliches Ergebnis
1	Anwendung wird geöffnet → Systemmenü "Bausteine"wird gedrückt.	"Prototyp-Bausteine Fenster wird geöffnet. Geladene Bausteine werden im dem Fenster angezeigt.	Das erwartete Ergebnis stimmt mit dem dem tatsächlichen Ergebnis überein.
2	Anwendung wird geöffnet → Systemmenü "Bausteine"wird gedrückt.	"Prototyp-Bausteine Fenster wird geöffnet. Es werden keine Bausteine im Fenster angezeigt	Das erwartete Ergebnis stimmt mit dem dem tatsächlichen Ergebnis überein.
3	Systemmenü "Bausteine"wird gedrückt → Sensoren-Untermenü wird geöffnet → Transformation-Untermenü wird geöffnet → Repräsentation-Untermenü wird geöffnet.	Beim Öffnen der Untermenüs werden die einzelnen Bausteine der unterschiedlichen Typen angezeigt.	Das erwartete Ergebnis stimmt mit dem dem tatsächlichen Ergebnis überein.
4	Klicke auf "Bearbeiten"Knopf unter den Namen der Bausteine → Bearbeiten der Baustein-Informationen durch Editieren des Textfeldes der Wert-Spalte.	Beim Drücken des Knopfes öffnet sich das Ëigenschaften"Fenster mit Baustein-Spezifischen Informationen über den Baustein. Eigenschaften lassen sich bearbeiten und der dadurch neu entstandene Baustein soll gespeichert oder weiterverwendet werden können.	Es öffnet sich das Ëinstellungen"Fenster im Hintergrund hinter dem "Prototyp-Bausteine"Fenster. Es werden nicht alle Eigenschaften, welche in der Tabelle dargestellt werden sollen dargestellt. Das Wert-Textfeld lässt sich editieren. Das Editieren des Textfeld erfüllt jedoch keine Funktionalität und lässt keine weiteren Funktionalitäten zu.
5	Systemmenü Ëinstellungen"wird gedrückt.	Es öffnet sich das Ëinstellungen"Fenster im Vordergrund der Anwendung, bestehend aus mehreren Untermenüs zu verschiedenen Einstellungsmöglichkeiten.	Das erwartete Ergebnis stimmt mit dem dem tatsächlichen Ergebnis überein. (Weitere Klickstrecken zum Ëinstellungen"Menü im Unterpunkt 4.1.todo)

Nr.	Aktions- und Klickstrecke	Erwartetes Ergebnis	Bewertung tatsächliches Ergebnis
6	Systemmenü "Hilfe"wird gedrückt.	Es öffnet sich das "Hilfe"Fenster im Vordergrund der Anwendung. In diesem Fenster sind Informationen über die Anwendung, wie auch über die Entwicklung zu finden. Zu den wichtigesten Funktionen der Anwendung gibt es ebenfalls kleinere Tutorials.	Das erwartete Ergebnis stimmt mit dem dem tatsächlichen Ergebnis überein.

Tabelle 2: Testen der Systemmenüs und dessen Funktionen.

Nr.	Aktions- und Klickstrecke	Erwartetes Ergebnis	Bewertung tatsächliches Ergebnis
1	Erstelle Konfiguration A → klicke auf Check-Knopf → klicke auf Ok → klicke auf Speichern-Knopf → wähle Namen und Pfad aus und klicke auf Speichern	Die Datei mit dem entsprechenden Namen ist am entsprechenden Ort zu finden. Die Datei enthält die Konfiguration A.	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.
2	Erstelle Konfiguration B → klicke auf Check-Knopf	Eine Meldung öffnet sich, dass die Konfiguration nicht gültig ist.	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.
3	klicke auf Speichern-Knopf → klicke auf Abbrechen	Das Hauptfenster ist geöffnet und es hat sich nichts verändert.	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.
4	klicke auf Laden-Knopf → klicke auf Abbrechen	Das Hauptfenster ist geöffnet und es hat sich nichts verändert.	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.
5	klicke auf Laden-Knopf → klicke auf laden, wobei Konfiguration A aus Nr. 1 ausgewählt ist	Eine Fenster öffnet sich und zeigt an, dass die Konfiguration erfolgreich geladen wurde.	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.
6	klicke auf Laden-Knopf → klicke auf laden, wobei Konfiguration B aus Nr. 2 ausgewählt ist	Eine Fenster öffnet sich und zeigt an, dass die Konfiguration nicht gültig, und somit nicht startbar ist.	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.

Tabelle 3: Klickstrecke um das Erstellen, Checken, Laden und Speichern einer Messkonfiguration mit der Gui zu testen.

Nr.	Aktions- und Klickstrecke	Erwartetes Ergebnis	Bewertung tatsächliches Ergebnis
1	klicke “Run Configuration”	Auslesen und Anzeigen der Messdaten (Messlauf) beginnt	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.
2	klicke “Pause”	Es werden keine neuen Daten verarbeitet und angezeigt.	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.
3	klicke “Resume”	Der Messlauf wird fortgesetzt.	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.
4	klicke “Pause” → “Save Measurement Data” → wähle Zielpfad aus → klicke “Save”	bisher gemessene Daten werden in einer Datei gespeichert	Das tatsächliche Ergebnis stimmt mit dem erwarteten Ergebnis überein.
5	klicke “Reset”	Der Messlauf wird auf den ursprünglichen Zustand zurückgesetzt.	Die angezeigten Messdaten werden nicht aus dem Anzeigefeld entfernt.

Tabelle 4: Kontrolle des Messlaufs

5 Testen der Qualität

5.1 Hallway Usability Testing

5.2 Testen der Qualität der Funktionalitäten

6 Durchführen der Testfälle aus dem Pflichtenheft

6.1 T010 Starten der Anwendung und Hilfe

DISCLAIMER: Der Testfall wurde so nicht wirklich durchgeführt, da der Pfad zur Textdatei noch nicht richtig funktioniert. Siehe Issue Nr. 15 in Git-Hub. Der Testfall wurde so angelegt, wie er später aussehen könnte. Er dient lediglich dazu, frühzeitig Feedback zu erhalten.

6.2 T020 Starten der Demo

Der Demomodus funktioniert anders als im Pflichtenheft vorgesehen. Es muss dazu in den Einstellungen der Anwendung der Demomodus eingestellt werden. Außerdem muss der Pfad angegeben werden, wo die Dateien mit Demo-Messdaten zu finden sind. Dann kann einfach eine gewöhnliche Konfiguration geladen (siehe Klickstrecke 4.1.2) und eine Messung gestartet werden (siehe Klickstrecke 4.1.3), als wäre der *Raspberry Pi* angeschlossen.

6.3 T030 Lehrer erstellt und speichert eine Messkonfiguration

Die Durchführung dieses Testfalles ist in Tabelle 6 zu sehen. Der Testfall aus dem Pflichtenheft schlägt fehl, weil viele Features nicht oder nur teilweise implementiert sind.

6.4 T040 Schüler bearbeitet Aufgabe

Die Durchführung dieses Testfalles ist in Tabelle 7 zu sehen. Dabei wird bei dem Informieren über die Bausteine das Untermenü aller Bausteine beliebig genutzt. Eine explizite Klickstrecke wird nicht angegeben, da eine solche über alle Bausteine viel Schreibarbeit und wenig Erkenntnis bringt

6.5 T050 Schüler startet Messung und speichert Ergebnisse

Die Durchführung dieses Testfalles ist in Tabelle 8 zu sehen.

6.6 T200 Laden einer ungültigen Datei als Messkonfiguration

Die Durchführung dieses Testfalles ist in Tabelle 9 zu sehen.

6.7 T210 Starten einer ungültigen Messkonfiguration

Die Durchführung dieses Testfalles ist in Tabelle 10 zu sehen.

6.8 T220 Entfernen eines Sensors bei laufender Messung

Die Durchführung dieses Testfalles ist in Tabelle 11 zu sehen.

Strukturelement	**Beschreibung**
Testfallnummer (Pflichtenheft)	T010
Testfallverweis	Zum Testen dieses Szenarios werden ausschließlich Klickstrecken benutzt. JUnit Tests sind zur Überprüfung von Gui-Verhalten ungeeignet und werden deswegen nicht benutzt.
Verantwortlicher Tester	Linus
Vorbedingung	Die Anwendung ist als fat-Jar-Datei auf dem Rechner vorhanden. Es läuft keine Instanz dieser Anwendung. Die Anwendung wurde zuvor auf dem Rechner ausgeführt.
Testziel	Zu Testen ist das Verhalten des Anwendung, wenn sie gestartet wird. Außerdem soll die Hilfe-Funktion der Anwendung getestet werden.
Klickstrecke	Doppelklick auf Fat-jar. Einzelklick auf Hilfe-Systemmenü.
Beschreibung	Die Anwendung öffnet sich bei dem Öffnen der Fat-Jar-Datei. Dabei öffnet sich das Hauptfenster, in dem keine Messkonfiguration zu sehen ist. Während des Startvorganges werden Bausteine und Benutzereigenschaften geladen. Drückt man den Knopf für die Hilfe, öffnet sich das Hilfefenster mit Informationen über die Benutzung der Anwendung, kurze Tutorials und Informationen über die Entwicklung.
Erwartetes Ergebnis	Das Hauptfenster und das Hilfefenster öffnen sich wie gewollt. Das Hilfefenster steht nach Öffnen im Vordergrund der Anwendung.
Verhalten im Fehlerfall	Eine Fehlermeldung wird angezeigt, falls beim Pfad zur Textdatei für das Hilfefenster keine Datei gefunden wurde. Falls in den Einstellungen falsche Pfade zur Initialisierung angegeben werden erscheinen ebenfalls aussagekräftige Fehlermeldungen.
Nachbedingung	Das Hauptfenster der Anwendung ist geöffnet. Es wird von dem geöffneten Hilfe-Fenster teilweise überdeckt. Das Hauptfenster ist leer, Bausteine und Eigenschaften wurden automatisch geladen. Der Hilfstext wird im Hilfe-Fenster angezeigt.
Getestete Anforderungen	**F010** erreiche GUI nach Start, **F140** leere Darstellung nach Anwendungsstart, **F480** Hilfe zu Anwendung, **F490** Texte der Anwendung auf Deutsch

Tabelle 5: Testfall T010 aus dem Plfichtenheft: Öffnen der Anwendung und Hilfe.

Strukturelement	Beschreibung
Testfallnummer (Pflichtenheft)	T30
Verantwortlicher Tester	David
Vorbedingung	Die Anwendung ist geöffnet, das Konfigurationsfeld ist leer.
Testziel	Zu Testen ist das Verhalten des Anwendung, wenn eine Konfiguration über den Editor teilweise erstellt wird, und als Zwischenergebnis gespeichert wird.
Klickstrecke	erstelle Konfiguration (siehe Beschreibung) → klicke auf Speichern-Knopf → wähle Pfad und Namen aus und klicke auf speichern
Beschreibung	Der Benutzer erstellt eine Konfiguration textuell über den Editor. Dabei gibt er die Blöcke als Liste von BlockIds an. Hier enthält die Liste zwei Sensoren(BMP180, MMA8451) und eine Transformation(Transformation-Add-2-Channel). Zu den drei Bausteinen erstellt der Benutzer eine jeweilige List ihrer Channel. Die Liste aller Verbindungen bleibt hier leer.
Abweichungen vom Pflichtenheft	Der Ablauf dieses Testfalles unterscheidet sich massiv von dem Testfall aus dem Pflichtenheft. Die Anwendung unterstützt kein Drag-and-Drop. Deshalb kann auch kein Baustein im Konfigurationsfeld als Icon sichtbar werden. Hier erfolgt die Erstellung der Konfiguration ausschließlich textuell über den Editor. Außerdem prüft die Anwendung nicht, ob der entsprechende Sensor angeschlossen ist, wenn ein Sensorbaustein hinzugefügt wird. Weiter überprüft die Anwendung nicht explizit beim Speichern, ob die Messkonfiguration gültig oder vollständig ist. Dies geschieht entweder über den Check-Knopf oder beim Laden der Konfiguration. Es könne keine expliziten Einstellung für nur eine Messkonfiguration eingestellt und mit ihr gespeichert werden.
Erwartetes Ergebnis	Die Anwendung ist geöffnet, die Konfiguration ist im Konfigurationsfeld zu sehen. Außerdem ist ein Fenster geöffnet, mit der Meldung, dass die Konfiguration erfolgreich gespeichert wurde. Die unfertige Konfiguration ist als Datei am entsprechenden Ort als Datei gespeichert.
Verhalten im Fehlerfall	Die Anwendung gibt nur bei Benutzung des Check-Knopfes an, ob die Konfiguration gültig ist.
Nachbedingung	Das Hauptfenster der Anwendung ist geöffnet. Es wird von dem geöffneten Hilfe-Fenster teilweise überdeckt.
Getestete Anforderungen	**F180** füge Sensor hinzu, **F210** füge Transformation hinzu, **F250** speichere Messkonfiguration
fehlende zu testete Anforderungen	**F190** prüfe ob Sensor angeschlossen, **F290** Einstellungen Messkonfiguration

Tabelle 6: Testfall T030 aus dem Plfichtenheft: Öffnen der Anwendung und Hilfe.

Strukturelement	Beschreibung
Testfallnummer (Pflichtenheft)	T40
Verantwortlicher Tester	David
Vorbedingung	Die Anwendung ist geöffnet, das Konfigurationsfeld ist leer.
Testziel	Zu Testen ist das Laden einer unvollständigen Konfiguration, deren Vervollständigung und das speichern der kompletten Konfiguration.
Klickstrecke	klicke auf Laden-Knopf → wähle richtige Datei aus und klicke auf laden → klicke auf OK → klicke auf Bausteine in der Systemleiste und informiere dich → klicke auf X → klicke auf Hilfe und informiere dich klicke auf X → vervollständige Konfiguration und klicke auf Check-Knopf → klicke Meldung(Konfiguration gültig) weg → klicke auf speichern und führe den Dialog korrekt aus
Beschreibung	Die ungültige Konfiguration wird geladen. Die Angebotenen Bausteine sind im Untermenü der Bausteine zu finden. Der Benutzer bearbeitet die Konfiguration, in dem er der Liste eine textuelle Repräsentation mit drei Kanälen hinzufügt. Außerdem erstellt er die Liste der Verbindungen. Eine Verbindung ist dabei ein Tupel zweier Kanäle. Ist der Benutzer der Meinung, dass die Konfiguration fertig ist, kann er den Check-Knopf benutzten, um zu prüfen, ob die Konfiguration gültig ist. Die Konfiguration kann jederzeit gespeichert und geladen werden.
Abweichungen vom Pflichtenheft	Die Anwendung prüft nicht beim Laden der Konfiguration, ob für die benutzten Sensorbausteine entsprechende Sensoren angeschlossen sind. Außerdem kann die Anwendung die graphische Darstellung der Konfiguration nicht aktualisieren, dass es keine solche gibt.
Erwartetes Ergebnis	Die vollständige Konfiguration ist als Datei am entsprechenden Pfad zu finden.
Verhalten im Fehlerfall	Beim Laden gibt die Anwendung an, dass die Konfiguration ungültig ist und bearbeitet werden sollte. Außerdem gibt sie bei Benutzung des Check-Knopfes an, ob die Konfiguration gültig ist.
Nachbedingung	Das Hauptfenster der Anwendung ist geöffnet. Die vollständige Konfiguration ist im Feld zu sehen. Sie ist auch als Datei gespeichert.
Getestete Anforderungen	**F230** füge Darstellung hinzu, **F470** Hilfe zu Bausteinen
fehlende zu testete Anforderungen	

Tabelle 7: Testfall T040 aus dem Plfichtenheft: Öffnen der Anwendung und Hilfe.

Strukturelement	Beschreibung
Testfallnummer (Pflichtenheft)	T050
Verantwortlicher Tester	Jan
Vorbedingung	Die Anwendung ist gestartet und es ist die in T040 erstelle Messkonfiguration geladen worden.
Testziel	Teste die Anwendung aus Sicht eines Schülers, der Messung starten und deren Ergebnisse speichern soll.
Klickstrecke	klicke auf "Run Configuration" → warte, bis Daten angezeigt werden → klicke auf "Pause" → klicke auf "Save Measurement Data" → definiere, wo die Daten gespeichert werden sollen → klicke auf "Save"
Beschreibung	Der Messlauf, der durch die bereits vorbereitete Konfiguration definiert ist, wird gestartet. Nachdem einige Daten erfolgreich ausgelesen wurden, wird der Messlauf angehalten und die Daten gespeichert.
Abweichungen vom Pflichtenheft	Da die Daten in der umgesetzten Anwendung nicht grafisch dargestellt werden können, kann auch keine grafische Darstellung gespeichert werden.
Erwartetes Ergebnis	Die ausgelesenen Daten sind in einer Datei am angegebenen Pfad gespeichert.
Verhalten im Fehlerfall	Falls die Zieldatei bereits existiert, werden die Daten nicht gespeichert.
Nachbedingung	Das Hauptfenster der Anwendung ist geöffnet. Die Konfiguration ist weiterhin geladen. Der Messlauf ist im pausierten Zustand. Die bislang ausgelesenen Daten sind in einer Datei gespeichert.
Getestete Anforderungen	**F130, F300** Messung starten, **F150, F320** Messdaten werden angezeigt, **F370** Messung pausieren, **400** Messdaten speichern
fehlende zu testete Anforderungen	**F410** da kein Graph erzeugt wird, kann auch kein Graph gespeichert werden

Tabelle 8: Testfall T050 aus dem Plfichtenheft: Öffnen der Anwendung und Hilfe.

Strukturelement	**Beschreibung**
Testfallnummer (Pflichtenheft)	T200
Verantwortlicher Tester	Leon
Vorbedingung	Die Anwendung ist geöffnet, das Konfigurationsfeld ist leer.
Testziel	Zu Testen ist das Verhalten der Anwendung beim Laden von ungültigen Messkonfigurationsdateien.
Klickstrecke	Klicke "Lade Konfiguration" → Wähle im Datei-Management-System die bereits existierende, fehlerhafte, Konfiguration aus → Klicke auf "Öffnen"
Beschreibung	Der Benutzer versucht eine Konfiguration über den vorgesehenen Weg zu laden. Nachdem die Konfigurationsauswahl getroffen wurde überprüft FreeJDAQ das Format der Konfiguration.
Abweichungen vom Pflichtenheft	Keine
Erwartetes Ergebnis	FreeJDAQ zeigt eine Fehlermeldung an, die auf die fehlerhafte Konfiguration hinweist. Anschließend wird der Ladevorgang der Konfiguration abgebrochen.
Nachbedingung	FreeJDAQ ist geöffnet und es ist keine Konfiguration geladen. FreeJDAQ wird von dem geöffneten Warnungs-Fenster teilweise überdeckt.
Getestete Anforderungen	**F430** überprüfe Format bei Laden
fehlende zu testete Anforderungen	Keine

Tabelle 9: Testfall T200 aus dem Plfichtenheft: Laden einer ungültigen Datei als Messkonfiguration

Strukturelement	**Beschreibung**
Testfallnummer (Pflichtenheft)	T200
Verantwortlicher Tester	Leon
Vorbedingung	Die Anwendung ist geöffnet, das Konfigurationsfeld ist leer.
Testziel	Zu Testen ist das Verhalten der Anwendung beim Laden von ungültigen Messkonfigurationsdateien.
Klickstrecke	Klicke "Lade Konfiguration" → Wähle im Datei-Management-System die bereits existierende, fehlerhafte, Konfiguration aus → Klicke auf "Öffnen"
Beschreibung	Der Benutzer versucht eine Konfiguration über den vorgesehenen Weg zu laden. Nachdem die Konfigurationsauswahl getroffen wurde überprüft FreeJDAQ das Format der Konfiguration.
Abweichungen vom Pflichtenheft	Keine
Erwartetes Ergebnis	FreeJDAQ zeigt eine Fehlermeldung an, die auf die fehlerhafte Konfiguration hinweist. Anschließend wird der Ladevorgang der Konfiguration abgebrochen.
Nachbedingung	FreeJDAQ ist geöffnet und es ist keine Konfiguration geladen. FreeJDAQ wird von dem geöffneten Warnungs-Fenster teilweise überdeckt.
Getestete Anforderungen	**F430** überprüfe Format bei Laden
fehlende zu testete Anforderungen	Keine

Tabelle 10: Testfall T210 aus dem Plfichtenheft: Starten einer ungültigen Messkonfiguration

Strukturelement	Beschreibung
Testfallnummer (Pflichtenheft)	T220
Verantwortlicher Tester	Jan
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.
Testziel	Teste das Verhalten der Anwendung, wenn ein Sensor ausfällt und dessen Datenstrom abbricht.
Klickstrecke	klicke auf "Run Configuration" → trenne Verbindung zum Sensor
Beschreibung	Der Benutzer startet die Messung. Die Verbindung zu einem Sensor wird getrennt.
Abweichungen vom Pflichtenheft	
Erwartetes Ergebnis	Die Anwendung erkennt, dass ein Sensor keine Daten mehr sendet. Die Messung stoppt. Eine aussagekräftige Fehlermeldung wird ausgegeben.
Verhalten im Fehlerfall	Der Messlauf bricht ab, ohne eine Fehlermeldung anzuzeigen.
Nachbedingung	Das Hauptfenster der Anwendung ist geöffnet. Die Konfiguration ist weiterhin geladen. Der Messlauf ist im pausierten Zustand. Die bislang ausgelesenen Daten sind in einer Datei gespeichert.
Getestete Anforderungen	**F450**
fehlende zu testete Anforderungen	

Tabelle 11: Testfall T220 aus dem Plfichtenheft: Öffnen der Anwendung und Hilfe.

7 Hardware Tests und sonstige Tests

7.1 Leistung und Speicherverbrauch

7.2 Hardware Test der Sensoren

7.3 Testen auf verschiedenen Systemen

8 Glossar

EclEmma EclEmma ist ein Plug-In für Eclipse für Code-Überdeckungsanalysen. Es basiert auf JaCoCo. Die hier verwendete Version ist 3.1.2.

JaCoCo JaCoCo ist eine freie Code-Überdeckungs Bibliothek für Java. Hier verwendete Version: 0.8.4.

Raspberry Pi Der Raspberry Pi ist ein Einplatinencomputer. In diesem Projekt dient der Raspberry Pi als Hardwareplattform, um Messwerte aus angeschlossenen Sensoren auszulesen.