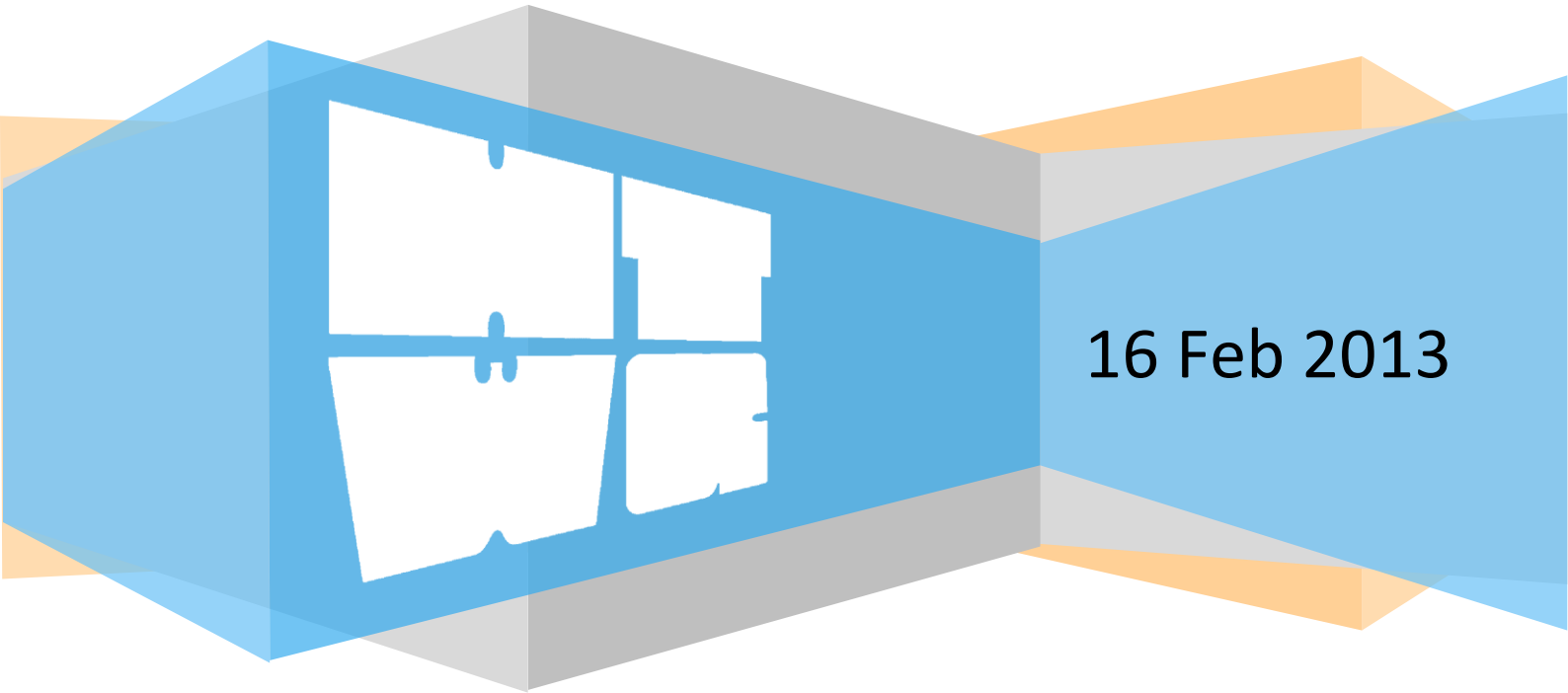
HTWG-Konstanz

PBF – Plants Best Friend

Der digitale, grüne Daumen

Sebastian Elben, Ulrich Raab



16 Feb 2013

Inhalt

1. Einleitung

Das Überleben einer regulären Hauspflanze ist stark von den Fähigkeiten und dem Engagement des Besitzers abhängig. Oft kann es, aufgrund mangelnder Beachtung zu Austrocknung, Übergießung, etc. kommen.

Aus diesem Grund wird im Rahmen der Vorlesung „Lange Nächte“ bei Prof. Dr. Christian Johner, ein System entwickelt, mit dem es möglich sein soll, die erfolgsentscheidenden Faktoren der Umgebung von Pflanzen für ihre Entwicklung zu überwachen um ggf. Interventionsmöglichkeiten zu erkennen.

1.1 Projektsetup

Für die Umsetzung des Systems steht ein Versionsverwaltungssystem unter der URL: https://github.com/BauerMitFackel/pbf.git des Anbieters GitHub zur Verfügung.

Das Projekt steht „Open Source“ für jeden zur Verfügung.

Als Hardware wird ein Arduino UNO Mikrocontroller-Board mit verschiedenen Sensoren verwendet.

Die Sensoren sind:

- Feuchtigkeitssensor
- Lichtsensor
- Temperatursensor

1.2 Projektmanagement

Als Vorgehen im Projekt wird eine abgewandelte Form der Scrum-Methode angewandt. Dieses Agile Vorgehen ermöglicht es mit sog. Backlogs -Arbeitsrückständen- zu arbeiten. Das Produktbacklog beinhaltet alle Nutzungsanforderungen aus Abschnitt 2.1. Eine Nutzungsanforderung wird in diesem Kontext, je nach Komplexität, „Story“ bzw. „Epic“ genannt. Jeder Story und jedem Epic sind Elemente untergeordnet welche die Systemanforderungen aus Abschnitt 2.2 in Tasks oder Arbeitspakete abbilden.

Aufgrund der zuvor definierten Prioritäten der Nutzungsanforderungen ergeben sich die Prioritäten der Systemanforderungen.

Aus dem Produktbacklog werden zunächst die Tasks der hoch priorisierten Storys zyklisch, in sogenannten Sprints, abgearbeitete. Je nach verbleibender Zeit, wird nach Abschluss eines Sprints ein neuer begonnen mit den nächsten Storys.

Durch diese Methode sind nach Ablauf der Projektzeit die hoch priorisierten Storys umgesetzt, die „must-have“ Features wurden implementiert.

Bei einer Projektverlängerung bzw. in Folgeprojekten können dann die übrig gebliebenen Features umgesetzt werden.

1.3 Aufbau des Dokuments

Zu Beginn des Dokuments steht die Analyse mit der Nutzungsanforderung und der Systemanforderung. Auf die Analyse folgt die Konzeption, welche die Topologie, die Software Architektur sowie die Anwendungsdokumentation enthält. Die Anwendungsdokumentation wurde nachträglich mit Screenshots der fertigen Anwendung illustriert.

Im Kapitel der Umsetzung wird kurz auf spezifische Probleme bei der Realisierung eingegangen. Es wird zwischen Hard- und Software unterschieden. Der Abschnitt Tests befasst sich mit der Ausführung und den Ergebnissen unterschiedlicher Tests.

Zuletzt wird auf mögliche folgende Projekte und Funktionalitäten, sowie auf den Verlauf des Projektes eingegangen.

2. Analyse

In diesem Kapitel soll geklärt werden, welche (Nutzungs-)Anforderungen der Anwender, an das System stellt. Aus diesen Anforderungen leiten sich die Systemanforderungen ab, welche für das folge Kapitel Konzeption die relevantesten Informationen liefern.

2.1 Nutzungsanforderung

Die Anforderungen beziehen sich auf den Anwender als alleinigen Stakeholder.

ID	Nutzungsanforderung	Prio
NA1	Der Anwender muss am System die Umgebungstemperatur einer Pflanze erkennen können.	1
NA2	Der Anwender muss am System die Erdfeuchtigkeit einer Pflanze erkennen können.	1
NA3	Der Anwender muss am System die Lichtintensität der Umgebung einer Pflanze erkennen können.	1
NA4	Der Anwender muss am System die Dauer der Sonnenbestrahlung einer Pflanze erkennen können.	2
NA5	Der Anwender muss mobil in der Lage sein, das System zu nutzen.	1
NA6	Der Anwender muss in der Lage sein, im System mehrere Pflanzen zu verwalten.	4
NA7	Der Anwender muss in der Anwendung schnell kritische Pflanzenumgebungen erkennen.	5
NA8	Der Anwender muss am System die Zustände der Umgebung der letzten Tage überblicken können.	2
NA9	Der Anwender muss am System eindeutig erkennen, um welche Pflanze es sich handelt.	4
NA10	Der Anwender muss über kritische Zustände und Interventionsmöglichkeiten informiert werden	6
NA11	Der Anwender muss sich intuitiv in der Anwendung bewegen können.	1
NA12	Der Anwender muss neue Sensorstationen verwalten können	2

2.2 Systemanforderung

Systemanforderungen werden aus den Nutzungsanforderungen Abgeleitet und versuchen diese technisch abzubilden.

ID	NA ID	Systemanforderung	Prio
SA1	NA1	Die Anwendung zeigt die aktuelle Umgebungstemperatur auf der Detailseite einer einzelnen Pflanze an	1
SA2	NA1,	Die Anwendungen greift die Temperaturinformation	1

	NA8	von einem Sensor je Mikrokontroller-board mit wiederum mehreren Pflanzen ab.	
SA3	NA1, NA2, NA3, NA9, NA4	Die Anwendung besitzt je Pflanze eine Detailansicht, welche Umgebungsinformationen, einen beschreibenden Namen und ein Bild der Pflanze bereitstellt.	4
SA4	NA2	Die Anwendung zeigt die aktuelle Erdfeuchtigkeit je Sensor an	1
SA5	NA2, NA8	Die Anwendungen greift die Erdfeuchtigkeit von einem Sensor je Pflanzen ab.	1
SA6	NA3	Die Anwendung zeigt die aktuelle Lichtintensität auf der Detailseite der Sensorstation an	1
SA7	NA3, NA8	Die Anwendungen greift die Lichtintensität von einem Sensor je Mikrocontroller-board ab.	1
SA8	NA8	Hinter jeden Sensor liegt ein Diagramm, welches den zeitlichen Verlauf der Umgebungsinformationen darstellt.	2
SA9	NA1, NA2, NA3, NA8, NA4	Die Anwendung muss Umgebungsinformationen mit ihrem zeitlichen Auftreten und der einzelnen Pflanze speichern und verwalten können.	3
SA10	NA4	Die Anwendung zeigt die Dauer der Leichteinstrahlung der letzten Periode auf der Detailseite der Sensorstation an.	2
SA11	NA5	Die Anwendung läuft auf jeden aktuelle Android Smartphone und verbindet sich über das lokale Netzwerk mit dem Mikrocontroller und den Sensoren.	1
SA12	NA6, NA11, NA7	Die Anwendung wird eine Ansicht allen Sensorstation(Mikrocontroller) mit aggregierten Informationen der unterliegenden Sensorik haben.	1
SA13	NA6, NA11	Die Übersicht aller Sensorstationen leitet beim Selektieren einer Station auf eine Übersicht aller Pflanzen der Station weiter.	4
SA14	NA6, NA11	Die Anwendung zeigt die Detailansicht einer Pflanze bei der Selektion der Pflanze in der Pflanzenübersicht je Sensorstation an.	4
SA15	NA7	Die Anwendung verdichtet alle	5

		Umgebungsinformationen und stellt sie in Form einer Gesamtstatus-Kennzahl da.	
SA16	NA7, NA11	Die Anwendung errechnet den Durchschnitt der Kennzahl des Gesamtstatus um den durchschnittlichen Status aller Pflanzen einer Sensorstation abzubilden	5
SA17	NA11	Die Anwendung berücksichtigt die Android User Interface Guidelines von Google	1
SA18	NA10	Die Anwendung erstellt im Fall eines kritischen Umgebungszustandes einen Eintrag im Google Kalender des Smartphone-Benutzers	6
SA19	NA12	Der Anwender muss mehrere Sensorstationen verwalten können (Anzeigen, Bearbeiten, Löschen, neu erstellen)	2

2.3 Use-Case Diagramm

Nach den Aufgenommen Anforderungen ergibt sich folgendes Use-Case Diagramm.

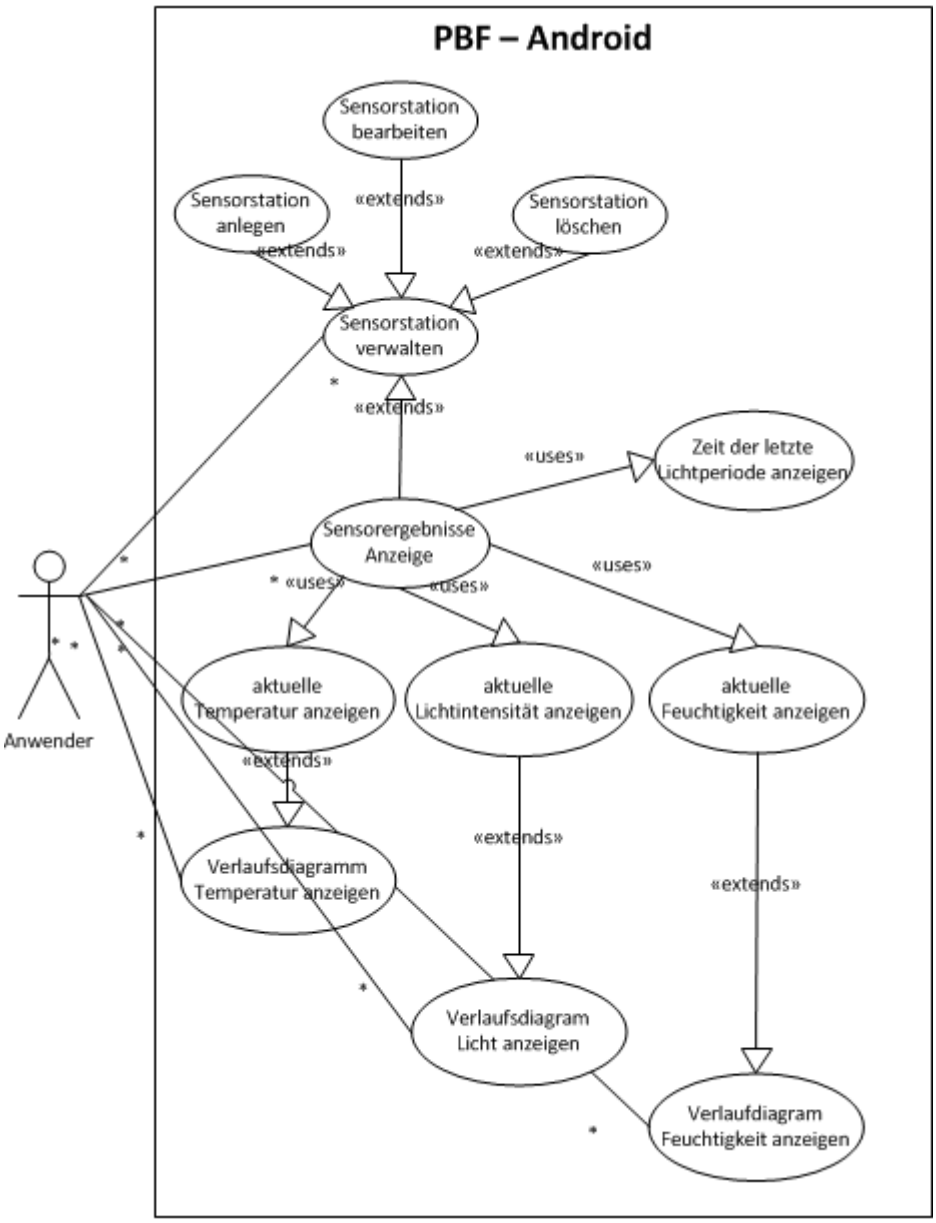


Abbildung 1: Use-Case Diagramm

3. Konzeption

Die Konzeption beinhaltet die System Topologie, die Software Architektur sowie die Anwenderdokumentation basierend auf Screenshots der Android Anwendung. Dem Dokument beigefügt, wird ein Video welches den Praxiseinsatz dokumentiert bereitgestellt.

3.1 Topologie

Das System ist wie in Abbildung 2 zu sehen, aufgebaut. Die Werte der einzelnen Sensoren werden von einem Arduino Ethernet Board abgegriffen und über einen Webservice im lokalen Netzwerk veröffentlicht. Das Gateway ist ein Wlan-Rooter über dem das Android Smartphone mit der PBF-Applikation auf den Webserver zugreifen kann und die Sensor Daten ausliest. Die Daten werden auf dem Smartphone für den Benutzer bereitgestellt.

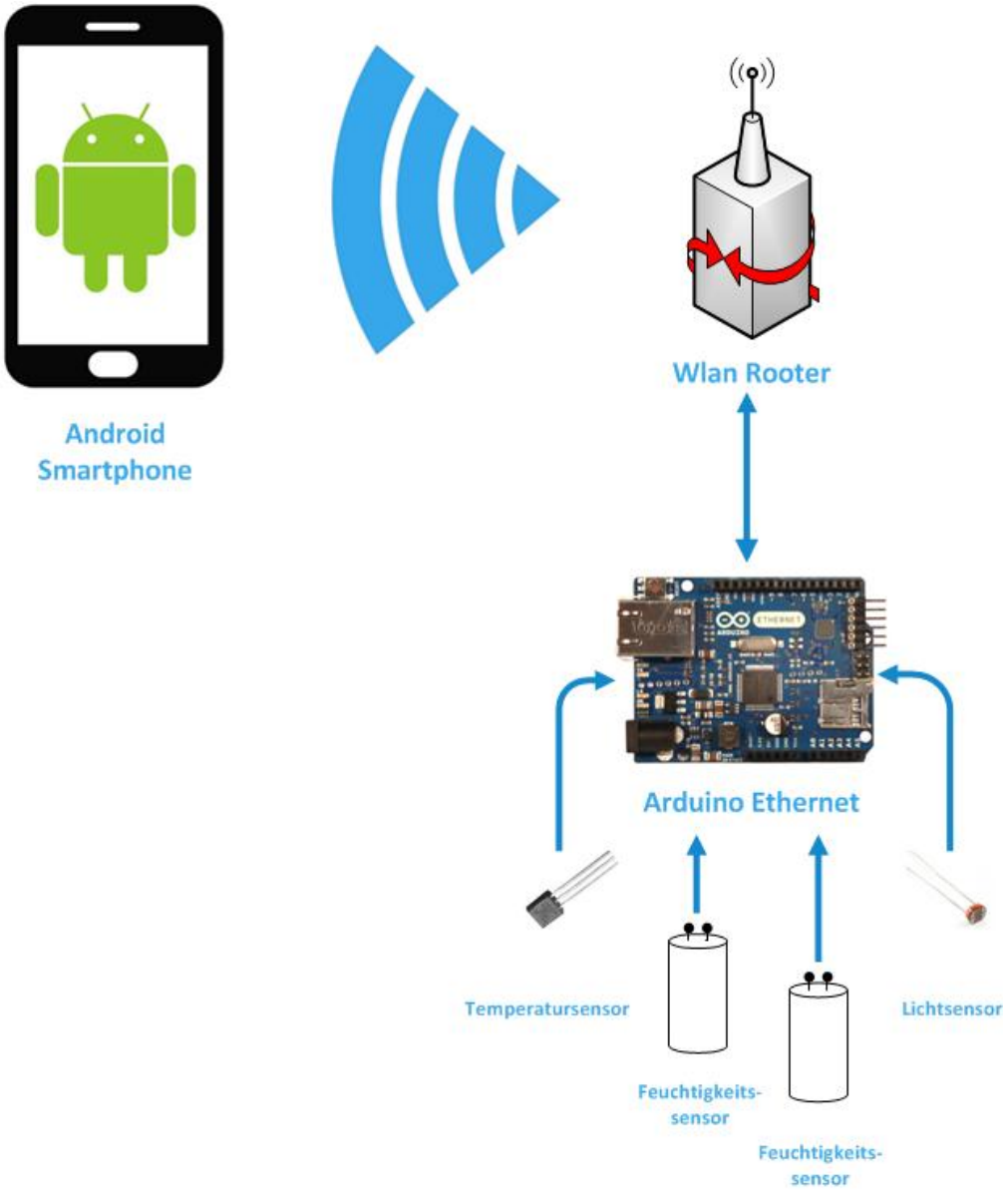


Abbildung 2: PBF System Topologie

3.2 Software Architektur

Die Softwarearchitektur beschreibt die grundlegenden Komponenten und deren Zusammenspiel innerhalb einer Anwendung. Die Qualität der Architektur ist dementsprechend wichtig für viele Aspekte wie beispielsweise die grundlegende Erweiterbarkeit der Software. Ausgangspunkt beim Entwurf ist das Architekturmuster MVC. Demzufolge ist die App logisch in die Bereich Model, View und Controller unterteilt.

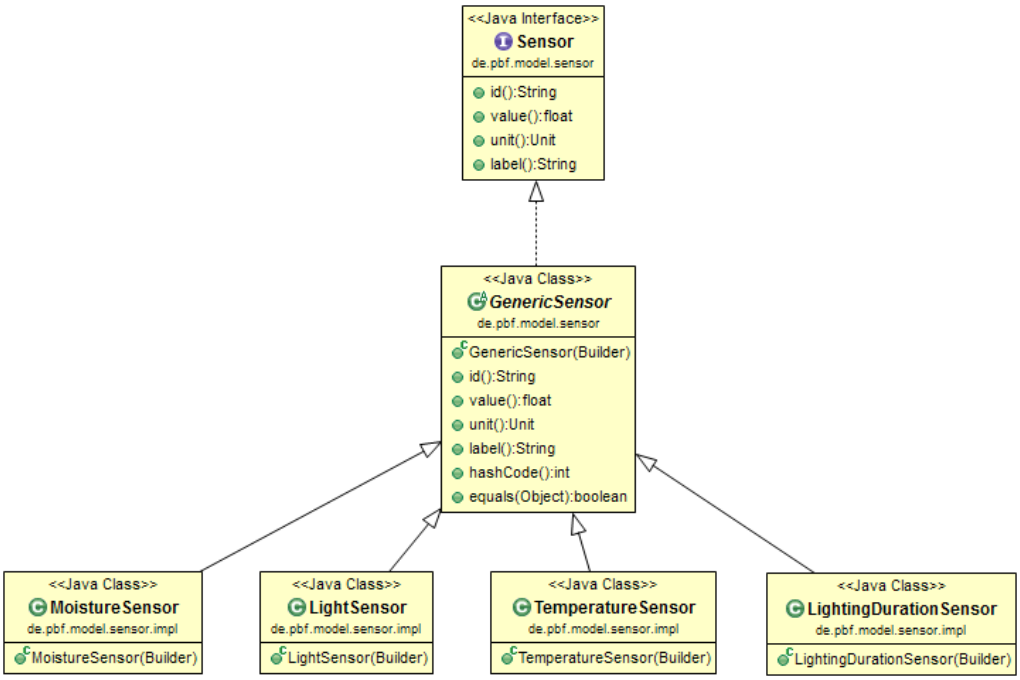


Abbildung 3: Klassendiagramm des Model

Wie man in Abbildung 3 sehen kann wurde ein Sensor Interface erstellt, um den Zugriff auf Sensoren zu vereinheitlichen. Die abstrakte Klasse GenericSensor implementiert das zuvor genannte Interface rudimentär und wird von den konkreten Sensor Klassen erweitert. Der im GenericSensor enthaltene Code wird folglich von allen Sensoren wiederverwendet und muss nicht je Sensor neu implementiert werden.

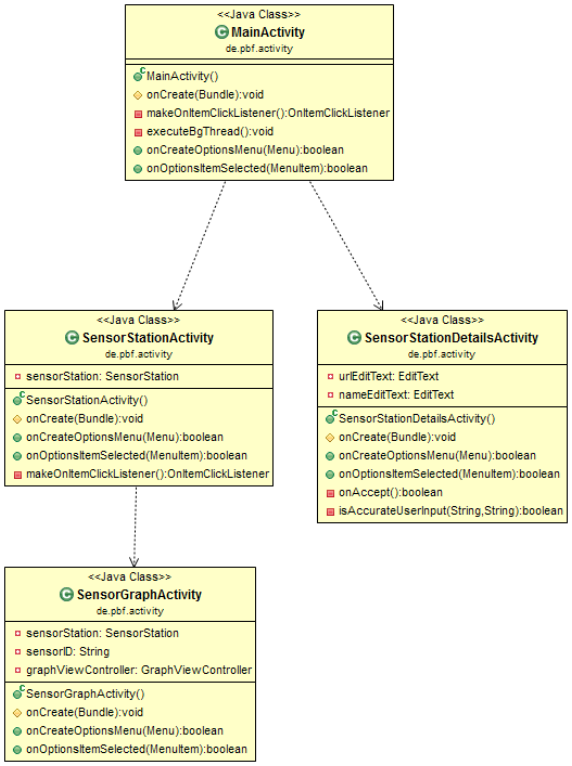


Abbildung 4: Klassendiagramm der Activities

Activities können im Kontext von Android den Controller Klassen zugewiesen werden. Die Verantwortung dieser liegt vor allem in der Bereitstellung eines Fensters, im dem die UI Elements platziert werden können. Zudem bieten Activities Methoden zur Verwaltung und Management des Lifecycles dieser.

Eine weitere Einarbeitung in die Architektur ist mit der Entwicklungsumgebung Eclipse und dem Plugin ObjectAid eigenständig möglich.

3.3 Anwendungsdokumentation

Beim ersten Start der Anwendung muss der Anwender die Adresse des Arduino Webservers und den Namen unter den die Sensorstation gelistet werden soll eingeben.

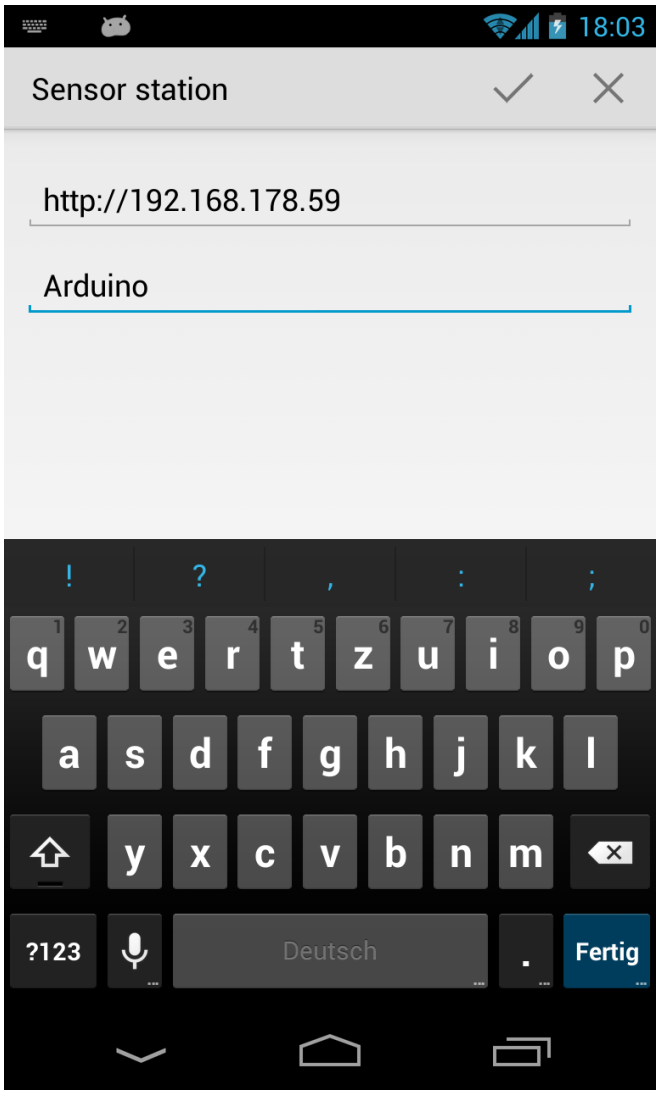


Abbildung 5: Adresse und Name von Sensorstation

Ist dies geschehen, so wird die Sensorstation in einer Liste angezeigt. In dieser Ansicht werden alle registrierten Sensorstationen angezeigt.

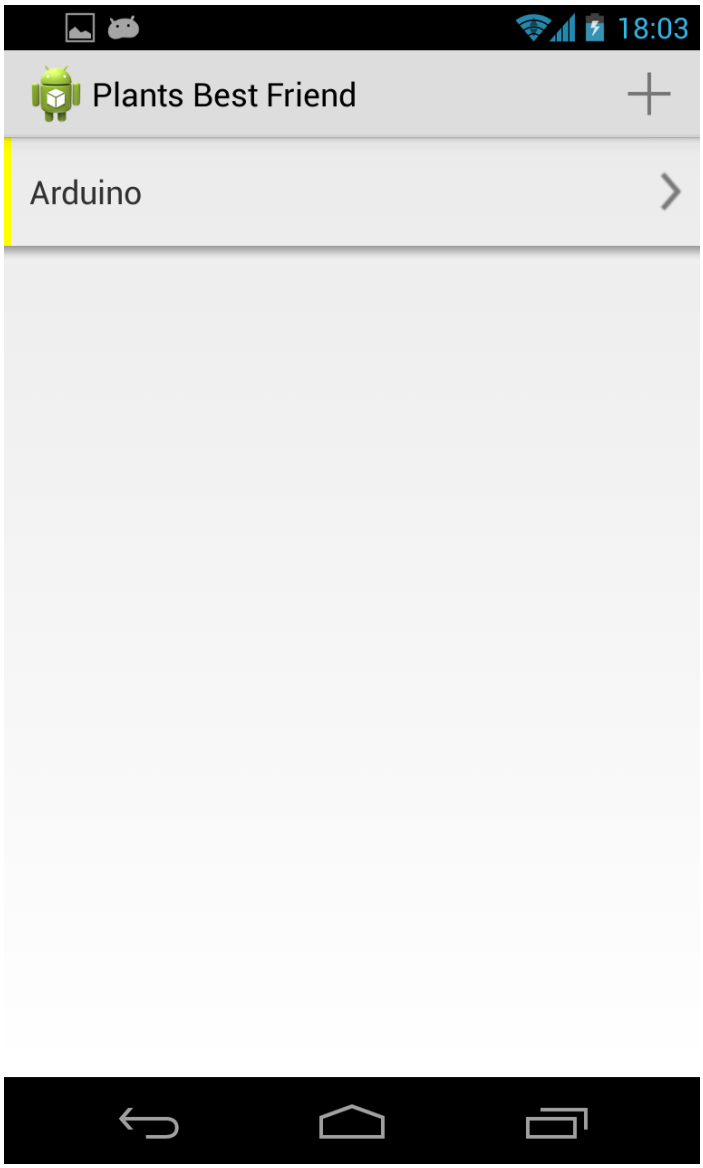


Abbildung 6: Liste registrierter Sensorstationen

Mit Klick auf eine Sensorstation gelangt der Anwender zu den Sensoren der Station. Diese werden auch als Liste angezeigt.

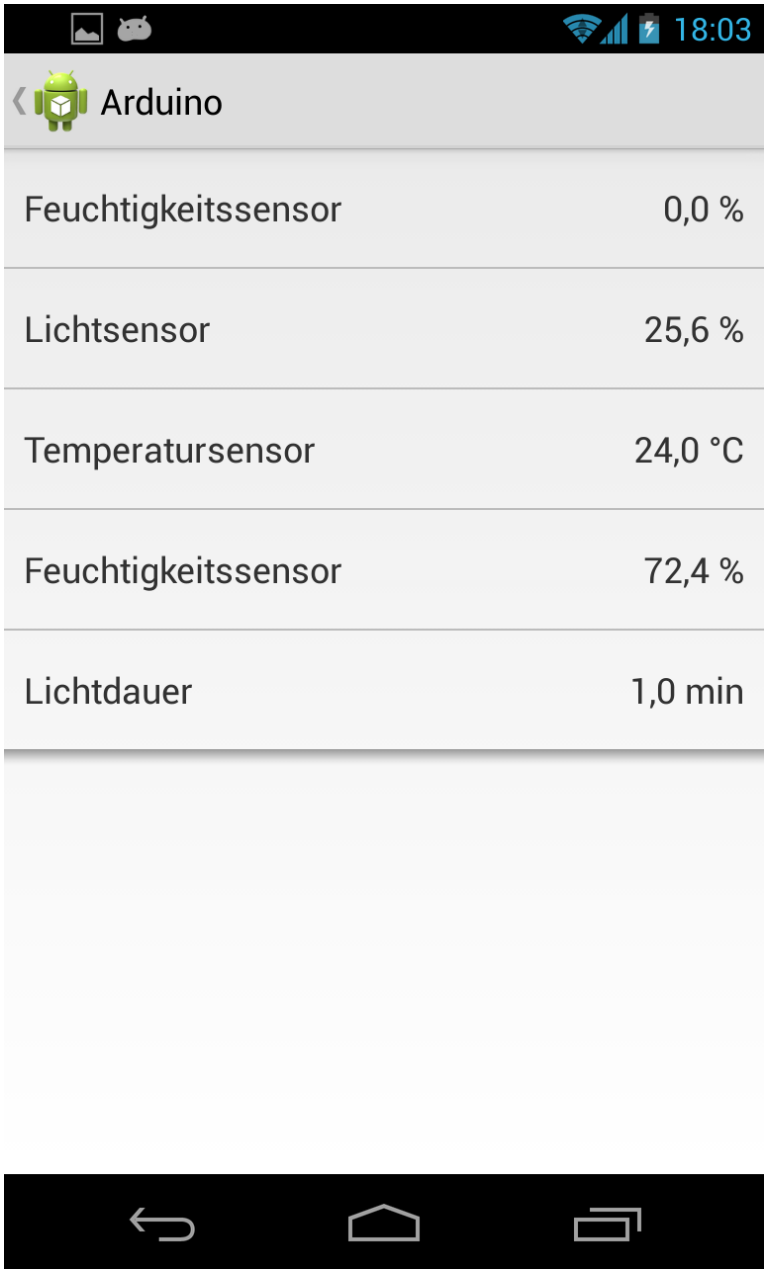


Abbildung 7: Sensorliste

Die Sensorwerte in Abbildung 7 repräsentieren reale Werte des Testaufbaus aus Abbildung 10. Da einer der Feuchtigkeitssensoren komplett trocken ist, wird auch in der Liste der Wert 0.0% angezeigt.

Mit Klick auf einen Sensor gelangt der Anwender auf das Verlaufsdiagramm dieses Sensors.

Der Listeneintrag „Lichtdauer“ ist allerdings kein richtiger Sensor sondern eine Berechnung anhand der Lichtintensität und dessen Verlauf. Genauer beschrieben in Abschnitt 18 unter „Berechnung der Lichtdauer“. Klickt der Anwender auf diesen Eintrag, so gelangt er zu dem Verlaufsdiagramm der Lichtintensität. Wie zu erwarten ist, wurde dieser Listeneintrag als Ausnahme in Android implementiert.

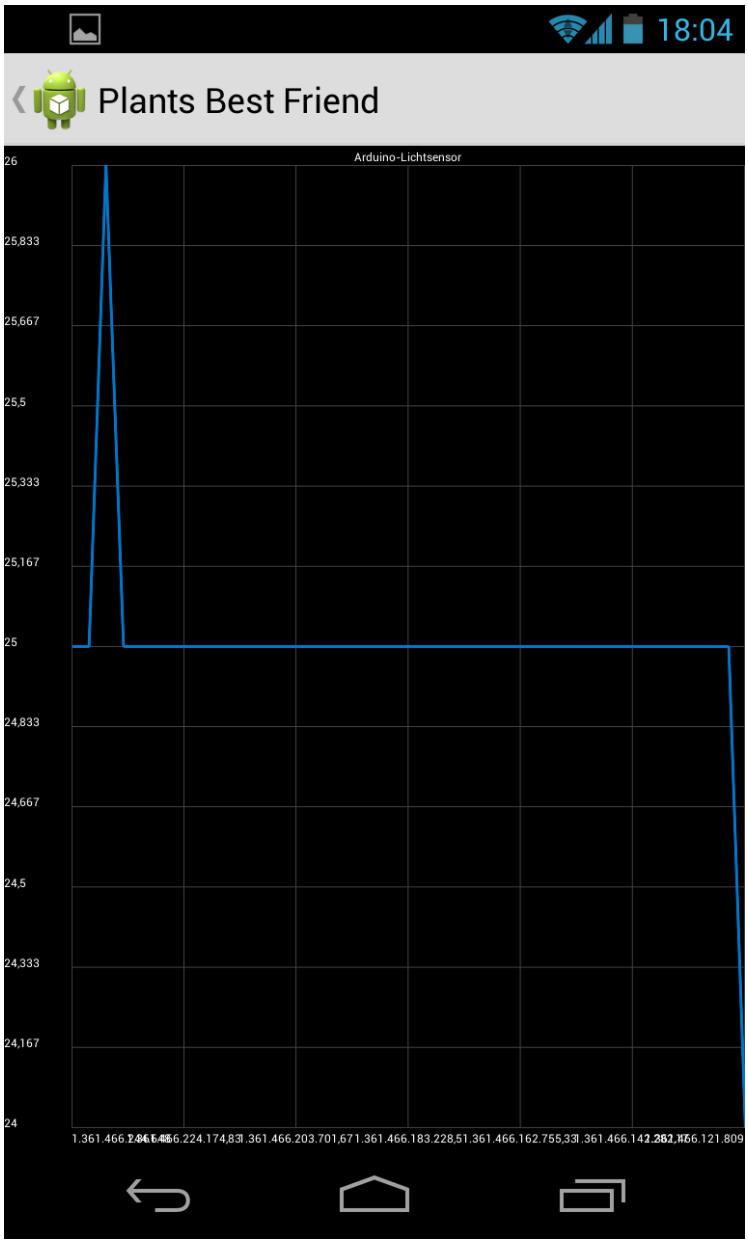


Abbildung 8: Verlaufsdiagramm des Lichtsensors

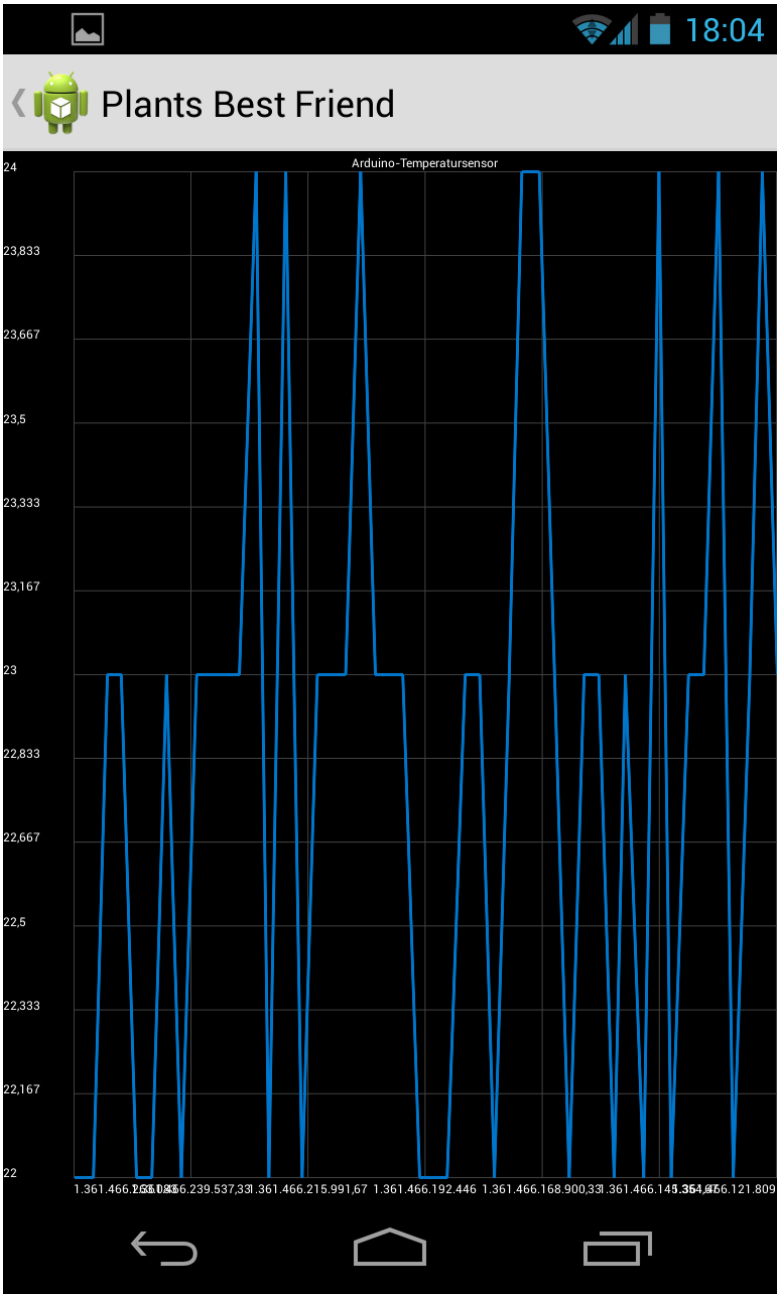


Abbildung 9: Verlaufsdiagramm des Temperatursensors

Wie in Abbildung 9 zu sehen ist, schwank die Temperatur bzw. der Messwert des Sensors. Diese Ungenauigkeit geht vom physischen Sensor aus und kann durch Softwaretests belegt werden.

4. Umsetzung

In diesem Kapitel wird auf spezielle Probleme und Sachverhalten während der Realisierung eingegangen.

4.1 Hardware

Für den direkten Umgang mit den Sensoren wird ein Arduino Ethernet Board verwendet. Dieses kontrolliert die Sensoren wie in Abschnitt 1.1 aufgelistet.

In Abbildung 10 ist der Testaufbau einer Sensorstation zu sehen. Einer der Feuchtigkeitssensoren befindet sich aus Testzwecken in einem Wasserglas um eine maximale Gießmenge zu simulieren.

Die Sensorstation ist über den Ethernet Anschuss direkt mit einem Rooter im lokalen Netz verbunden.

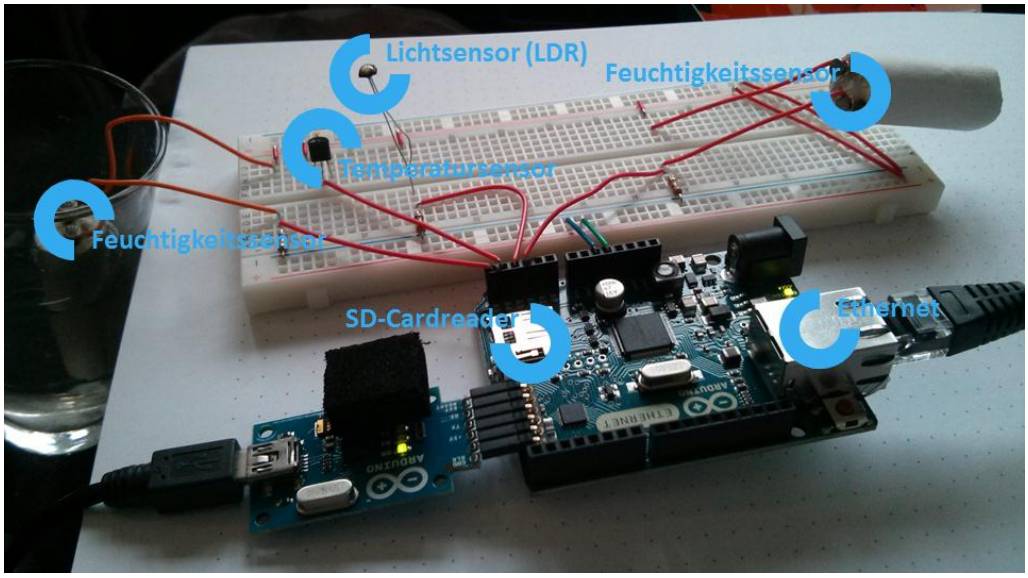


Abbildung 10: Sensorstation-Testaufbau

Feuchtigkeitssensor

Die Feuchtigkeitssensoren wurden selber hergestellt. Sie bestehen hauptsächlich aus Gips und nutzen die Leitfähigkeit von Wasser.

In Abbildung 11 sind alle verbauten Materialien zu sehen. Als Schale wird ein gerolltes Papier verwendet. Zwei verzinkte Nägel pro Sensor für den Plus und Minus pol und Fertigspachtelmasse für den Körper des Sensors.

Abbildung 11: Feuchtigkeitssensor - einzelne Komponenten

Die Papierröhren werden mit der Spachtelmaße gefüllt und die beiden Nägel werden parallel mit circa gleichem Abstand in die Spachtelmaße gesteckt. Nachdem die Spachtelmaße getrocknet ist, kann der Sensor verwendet werden.

Abbildung 12: Feuchtigkeitssensoren – beim trocknen

Der Gibs bzw. die Spachtelmaße saugt sich mit umliegendem Wasser voll und gewährleistet dadurch einen mehr oder minder guten Stromfluss. Je mehr Wasser die Maße aufnimmt, je stärker fließt das Wasser von einem Nagel zum anderen. Die Stromstärke wird verwendet um das Wassergehalt abzuschätzen.

Abbildung 13: Fertige Feuchtigkeitssensoren

4.2 Software

In diesem Teil des Dokuments wir auf die Umsetzung der Software eingegangen.

4.2.1 Android

In dieser Sektion wird nur ein kleiner Teil aus der Umsetzungsphase erläutert.

GUI

Die Gestaltung der Benutzeroberfläche resultiert aus den Richtlinien der Google User Interface Guidlines, welche unter folgender URL zu finden ist: http://developer.android.com/guide/practices/ui_guidelines/index.html

Das bedeutet unteranderem, die Navigation der App Android konform zu realisieren. So funktioniert der Back-Button wie der Button neben dem App-Logo um auf die vorrangehende Activity zurück zu springen.

Json und GSON

Zum Austausch von Daten zwischen Arduino und Android wird Json verwendet. Json hat den Vorteil kompakt zu sein und infolge dessen wenig Speicherplatz zu verbrauchen. Dies ist besonders dann ein Vorteil wenn unterwegs auf den Webservice zugegriffen wird.

Um die vom Arduino Webservice erhaltenen Daten in Form von Json Zeichenketten in Java Objekte umzuwandeln wird das Framework Gson verwendet. Dieses ist einfach zu benutzen und gleichzeitig flexibel genug um bei Bedarf angepasst zu werden. Ein weiterer Positiver Aspekt ist die Tatsache dass es Open-Source ist und kontinuierlich weiterentwickelt wird. Die URL des Projektes lautet: http://code.google.com/p/google-gson/

GraphView

Um den Verlauf eines Sensorwertes zu visualisieren wurde die externe Bibliothek des Open Source Projektes GrapView verwendet. Zu finden unter der URL:

https://github.com/jjoe64/GraphView

Durch diese Bibliothek ist es möglich die Zeitabhängigen Sensorwerte als Liniendiagramm oder als Blockdiagramm anzuzeigen. Leider stellte sich während der Implementierung heraus, dass die Aktualität der Bibliothek und die der Dokumentation und Beispiele nicht dieselbe ist, was den Implementierungsstil auf eine sehr experimentelle Art Veränderte.

Nachträglich betrachtet würden andere Projekt für die Anzeigelogik der Diagramme verwendet werden.

Berechnung der Lichtdauer

Neben der Lichtintensität welcher die Pflanze ausgesetzt ist, kommt es auch auf die Dauer der Lichteinstrahlung an. Hierfür soll den Anwender die Zeit der letzten Lichtperiode angezeigt werden. So kann er feststellen ob für eine Pflanze der richtige Tag/Nacht-Zyklus besteht.

Um die Zeitliche Dimension zu ermöglichen wird den Sensorwerten, bei der Abfrage vom Webserver, die aktuelle Zeit hinzugefügt.

Anhand der vergangen Werte wird überprüft wann das letzte Mal Licht für die Pflanze herrschte und wie lange. Die Berechnung kann wie folgt dargestellt werden.

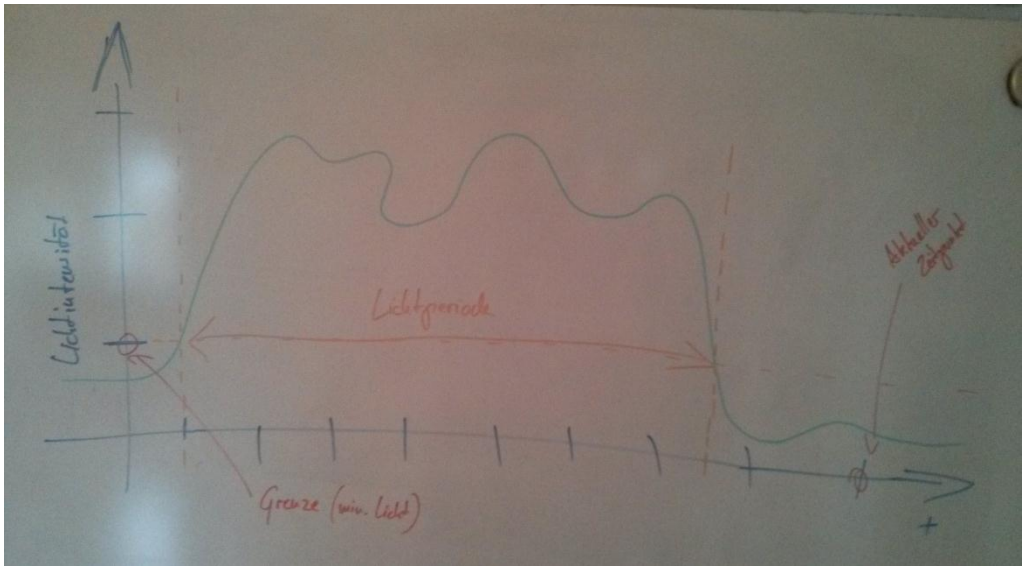


Abbildung 14: Berechnung von Lichtperiode

4.2.2 Arduino

Die Sensorstation fungiert auch als Webserver. Der Webserver stellt per JSON Dateiformat die aktuellen Werte der angeschlossenen Sensoren im lokalen Netz zur Verfügung.

Eine mögliche Ausgabe könnte wie folgt aussehen.


```
[{"type":"moisture_sensor","id":"S2","value":0},{"type":"light_sensor","id":
"S3","value":230},{"type":"temperature_sensor","id":"S4","value":25},{"typ
e":"moisture_sensor","id":"S5","value":727}]
```


Jeder Sensor hat als Variablen: den Typ des Sensors, eine eindeutige ID und den aktuellen Wert.

Die Ausgabe des Webserver wird jede 5 Sekunden erneuert.

5. Qualität & Test

In diesem Kapitel soll der Nachweis zur Qualitätssicherung erbracht werden.

Probleme wie Bugs können über den Bugtracker von GitHub eingereicht werden. Dieser ist unter https://github.com/BauerMitFackel/pbf/issues zu finden.

5.1 Codedokumentation

Für die Dokumentation des Programmcodes wurden Javadocs für Klassen sowie öffentliche Methoden geschrieben. Auf das Schreiben von Javadocs für private Methoden und Variablen wurde aufgrund der Übersichtlichkeit verzichtet.

5.2 Softwaretests

Die implementierten Testfälle sollen zeigen, dass die Android Applikation und deren einzelne Komponenten, soweit möglich fehlerfrei funktionieren. Unter anderem werden die verwendeten Algorithmen, der Datenaustausch sowie die asynchron bzw. nebenläufig ablaufenden Programmteile getestet, da dies

die kritischen Bereiche der Anwendung sind. Abbildung 15 zeigt die Ergebnisse eines Testlaufs in der Entwicklungsumgebung. Wie man sehen kann sind alle Testfälle positiv, d.h. die in den Testfällen abgebildeten Kriterien werden vollständig erfüllt.

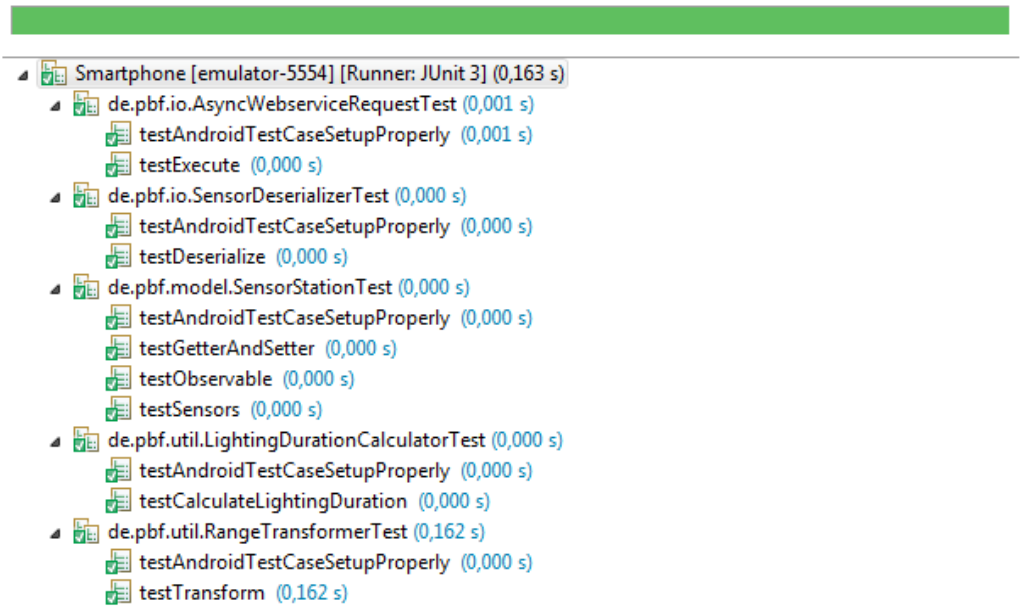


Abbildung 15: Testfälle

5.3 Systemtests

Der Systemtest soll prüfen und bewerten, in wieweit das System die definierten Anforderungen erfüllt.

Dafür wird zunächst eine Verifikation durgeführt. Diese prüft anhand der Systemanforderungen ob das System richtig entwickelt wurde.

Bei der Validierung wird mit Hilfe der Nutzungsanforderungen geprüft, ob das richtige System entwickelt wurde und die Anforderungen des Nutzers erfüllt.

5.3.1 Verifizierung

Zu jeder Systemanforderung wird die Erfüllung dokumentiert sowie eine grobe Einschätzung zum Erfüllungsgrad abgegeben. Der Erfüllungsgrad schließt die Komplexität und den Zeitaufwand der fehlenden Funktionen mit ein.

ID	Prio	Verifizierung	Erf. G
SA1	1	Die Umgebungstemperatur wird auf der Detailseite der Sensorstation angezeigt → **gefordert**: auf Detailseite der Pflanze Die Aktualität des Ergebnisses wird durch die Update-routine und die Observer sichergestellt.	80%
SA2	1	Temperatur wird auf Detailseite einer Sensorstation angezeigt	100%
SA3	4	Es wurden keine Ansichten noch Modelobjekte für Pflanzen realisiert.	0 %
SA4	1	Die Erdfeuchtigkeit wird je Sensor auf der Detailseite der jeweiligen Sensorstation angezeigt.	100%
SA5	1	Die Erdfeuchtigkeit je Sensor wird angezeigt → **gefordert**: je Pflanze (Anwender kann Sensor keiner Pflanze zuordnen) Die Aktualität des Ergebnisses wird durch die Update-routine und die Observer sichergestellt.	90%
SA6	1	Die Lichtintensität wird auf der Detailseite der Sensorstation angezeigt Die Aktualität des Ergebnisses wird durch die Update-routine und die Observer sichergestellt.	100%
SA7	1	Die Lichtintensität wird auf der Detailseite der Sensorstation angezeigt	100%
SA8	2	Hinter jeden Sensor liegt ein Diagramm, welches den zeitlichen Verlauf der Umgebungsinformationen darstellt.	100%
SA9	3	Die Umgebungsinformationen werden mit Zeit je Sensorstation gespeichert → **gefordert**: je Pflanze und Möglichkeit zum Verwalten	60%
SA10	2	Die Dauer der Leichteinstrahlung der letzten Periode wird auf der Detailseite der Sensorstation angezeigt.	100%
SA11	1	Die Anwendung läuft auf jeden aktuelle Android Smartphone und verbindet sich über das lokale Netzwerk mit dem Mikrocontroller und den Sensoren. → Ab Android Version 4.2 (API 17)	100%

SA12	1	Die Anwendung wird eine Ansicht allen Sensorstation(Mikrocontroller) mit aggregierten Informationen der unterliegenden Sensorik haben. → **gefordert**: Aggregierte Informationen	40%
SA13	4	Die Übersicht aller Sensorstationen leitet beim Selektieren einer Station auf eine Übersicht aller Pflanzen der Station weiter.	0%
SA14	4	Die Anwendung zeigt die Detailansicht einer Pflanze bei der Selektion der Pflanze in der Pflanzenübersicht je Sensorstation an.	0%
SA15	5	Die Anwendung verdichtet alle Umgebungsinformationen und stellt sie in Form einer Gesamtstatus-Kennzahl da.	0%
SA17	1	Die Anwendung berücksichtigt die Android User Interface Guidelines von Google	100%
SA18	6	Die Anwendung erstellt im Fall eines kritischen Umgebungszustandes einen Eintrag im Google Kalender des Smartphone-Benutzers	0%
SA19	2	Der Anwender muss mehrere Sensorstationen verwalten können (Anzeigen, Bearbeiten, Löschen, neu erstellen)	100%

Anhand der Priorität wird der Erfüllungsgrad bewertet. Damit ergibt sich ein **Gesamterfüllungsgrad der Systemanforderungen** von **77,65 %**. Aufgrund des angewendeten Projektvorgehens und der üppigen Anforderungsanalyse war ein höherer Erfüllungsgrad in der benötigten Zeit nur schwer möglich.

5.3.2 Validierung

ID	Prio	Validierung	Erf. G
NA1	1	Der Anwender muss am System die Umgebungs-temperatur einer Pflanze erkennen können. → Die Umgebungstemperatur wird auf Ebene der Sensorstation angezeigt, diese gilt für alle überwachten Pflanzen an der Sensorstation.	95%
NA2	1	Der Anwender muss am System die Erdfeuchtigkeit einer Pflanze erkennen können.	90%

		→ Der Anwender muss die Pflanze anhand des Sensors erkennen, da der Wert auf Ebene der Sensorstation angezeigt wird.	
NA3	1	Der Anwender muss am System die Lichtintensität der Umgebung einer Pflanze erkennen können. → Die Umgebungstemperatur wird auf Ebene der Sensorstation angezeigt, diese gilt für alle überwachten Pflanzen an der Sensorstation.	95%
NA4	2	Der Anwender muss am System die Dauer der Sonnenbestrahlung einer Pflanze erkennen können.	100%
NA5	1	Der Anwender muss mobil in der Lage sein, das System zu nutzen.	100%
NA6	3	Der Anwender muss in der Lage sein, im System mehrere Pflanzen zu verwalten. → physisch können mehrere Pflanzen verwaltet werden, in der Anwendung muss der Anwender allerdings wissen welcher Feuchtigkeitssensor welcher Pflanze gehört.	25%
NA7	4	Der Anwender muss in der Anwendung schnell kritische Pflanzenumgebungen erkennen. → Durch das Ablesen der Sensorwerte kann eine kritische Situation erkannt werden. Eine Aggregierte Kennzahl wäre jedoch effizienter.	75%
NA8	2	Der Anwender muss am System die Zustände der Umgebung der letzten Tage überblicken können.	100%
NA9	4	Der Anwender muss am System eindeutig erkennen, um welche Pflanze es sich handelt. → Pflanzen wurden nicht implementiert	0%
NA10	6	Der Anwender muss über kritische Zustände und Interventionsmöglichkeiten informiert werden	0%
NA11	1	Der Anwender muss sich intuitiv in der Anwendung bewegen können. → subjektive Einschätzung eines langjährigen Androidnutzers.	100%
NA12	2	Der Anwender muss neue Sensorstationen verwalten können	100%

Da die Sensoren Pflanzenübergreifend und nur Raumspezifisch sind, ist die Erfüllung von NA1 und NA2 fast gegeben. Es wird davon ausgegangen, dass alle Pflanzen einer Sensorstation im selben Raum stehen.

Der Gesamterfüllungsgrad beinhaltet den Erfüllungsgrad der einzelnen Nutzungsanforderungen sowie die jeweilige Priorisierung. Damit ergibt sich ein **Gesamterfüllungsgrad der Nutzungsanforderungen** von **84%**.

6. Ausblick

Wie schon an den Anforderungen und ihrer Erfüllung zu sehen ist, bestehen schon viele Features die noch in der Anwendung aufgenommen werden können. So wäre beispielsweise eine Anbindung an den Google Kalender des Android Nutzers möglich. Hiermit könnt der Nutzer über Interventionsmöglichkeiten zu einem bestimmten Zeitpunkt aufgeklärt werden.

Schaut man noch weiter in die mögliche Zukunft, so könnte die Sensorikpalette erweitert werden. Ein PH-Wert bzw. IC-Wert Sensor kann bei Nutzpflanzen die Düngesituation abbilden und bei Über- bzw. Unterdüngung einen Alarm auf dem Smartphone auslösen.

Eine wichtige Erweiterung des Systems, wäre die Adaptierbarkeit von Sensoren auf dem Arduino-Board zu steigern. So können Botaniker das System mit weiteren Sensoren ausbauen.

7. Fazit

Das Projekt ist in den Augen der Mitglieder durchweg positiv verlaufen. Es gab viele Herausforderungen die (fast) alle lösbar waren. So stellte die Verwendung der GraphView Bibliothek das Team auf Probe. Die angeboten Bibliothek entsprach nicht dem Beispielen noch der sehr rudimentären Dokumentation. Alle die Feststellung dieser Tatsache beschäftigte das Team für eine lange Zeit.

Auch der Webserver von Arduino stellte eine große Herausforderung dar. Die Verbindung zum Rooter brauchte viele Anläufe da MAC und IP Adresse nicht erkannt wurden.

Leider ist die Zeit für das Projekt verstrichen obwohl noch so viele Ideen vorhanden sind, die nach einer Umsetzung schreien. Jedoch hat sich das Team auf ein folge Projekt geeinigt, um eine Vielzahl der Ideen in kommender Zeit zu realisieren.