

Projektplan

Inhaltsverzeichnis

| | | |
|----------|----------------------------------------------|-----------|
| 1 | EINFÜHRUNG | 4 |
| 1.1 | ZWECK DES DOKUMENTS | 4 |
| 1.2 | REFERENZEN | 4 |
| 2 | PROJEKTÜBERSICHT | 5 |
| 3 | PROJEKTORGANISATION | 6 |
| 3.1 | ORGANISATIONSSTRUKTUR | 6 |
| 3.1.1 | STUDIERENDE | 6 |
| 3.1.2 | BETREUER HSR | 6 |
| 3.1.3 | ANSPRECHPARTNER AUFTRAGGEBER | 6 |
| 4 | MANAGEMENT ABLÄUFE | 7 |
| 4.1 | TERMINE | 7 |
| 4.2 | ZEITBUDGET | 7 |
| 4.3 | VORGEHEN | 7 |
| 4.4 | MEILENSTEINE | 8 |
| 4.5 | BESPRECHUNGEN | 9 |
| 4.5.1 | PROTOKOLLFÜHRUNG | 9 |
| 5 | RISIKOMANAGEMENT | 10 |
| 5.1 | RISIKEN | 10 |
| 5.2 | UMGANG MIT RISIKEN | 10 |
| 6 | ARBEITSPAKETE | 11 |
| 7 | INFRASTRUKTUR | 12 |
| 7.1 | PROJEKTMANAGEMENT TOOL | 12 |
| 7.2 | VERSION CONTROL SYSTEM | 12 |
| 7.3 | HOSTING DER SERVER-API | 12 |
| 7.4 | CONTINUOUS INTEGRATION & CONTINUOUS DELIVERY | 12 |
| 7.5 | LICHTMESSGERÄTE | 12 |
| 7.6 | POSITIONIERUNGSSYSTEM | 12 |
| 7.7 | WEITERE INFRASTRUKTUR | 13 |
| 8 | QUALITÄTSMASSNAHMEN | 14 |
| 8.1 | ENTWICKLUNGS-WORKFLOW | 14 |
| 8.2 | CODE STYLE | 15 |
| 8.3 | TESTING | 15 |
| 8.4 | DEFINITION OF DONE | 15 |

| | | |
|------------------|-------------------------------------|------------------|
| <u>9</u> | <u>ABBILDUNGSVERZEICHNIS</u> | <u>16</u> |
| <u>10</u> | <u>TABELLENVERZEICHNIS</u> | <u>17</u> |

1 Einführung

1.1 Zweck des Dokuments

Ziel des Dokuments ist es, dem Leser einen Überblick über die Bachelorarbeit zu geben und so einen schnellen Einstieg in den Projektablauf zu gewähren. Dabei wird der Projektablauf definiert, die möglichen Risiken analysiert und ein Überblick über die Arbeitsweise und Infrastruktur gegeben.

1.2 Referenzen

Dieses Dokument dient als Ergänzung zum technischen Bericht der Bachelorarbeit. Alle projektspezifischen Informationen, die im Bericht keinen Platz finden, sind hier aufgeführt.

2 Projektübersicht

Motivation, Zweck und Ziel, Lieferumfang sowie Annahmen und Einschränkungen werden im technischen Bericht bereits detailliert ausgeführt.

Um allen Beteiligten einen möglichst einfachen Zugriff zum aktuellen Projektstand zu ermöglichen, wurde eine Website erstellt. Dort sind die wichtigsten Dokumente und weiterführende Links aufgelistet.

Website: <http://flux-coordinator.com>

3 Projektorganisation

Diese Arbeit wird als Bachelorarbeit an der HSR Hochschule für Technik Rapperswil im Frühjahrssemester 2018 durchgeführt.

3.1 Organisationsstruktur

Diese Bachelorarbeit findet im Auftrag der Firma *HSi Elektronik AG* statt. Zur Betreuung und Bewertung der Arbeit wird ein Dozent der HSR zugeteilt.

3.1.1 Studierende

- Esteban Luchsinger, esteban.luchsinger@hsr.ch
- Patrick Scherler, patrick.scherler@hsr.ch

3.1.2 Betreuer HSR

- Prof. Dr. Farhad Mehta, *Institut für Software*, farhad.mehta@hsr.ch

3.1.3 Ansprechpartner Auftraggeber

- Tobias Hofer, *HSi Elektronik AG*, tobias.hofer@hsi-ag.ch

4 Management Abläufe

4.1 Termine

Die zeitliche Planung ist grösstenteils durch die Organisation als Bachelorarbeit gegeben. Folgende Termine sind definiert:

| Termin | Beschreibung | Betrifft |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------|------------------|
| 19.02.18 | Beginn der Bachelorarbeit | Studierende |
| 23.02.18 | Kickoff Meeting Auftraggeber | Stud. & Auft. |
| 26.02.18 | Kickoff Meeting Betreuer | Stud. & Betreuer |
| 08.06.18 | Abgabe von Abstract und Poster an Betreuer | Studierende |
| 13.06.18 | Der Betreuer gibt das Dokument mit dem korrekten und vollständigen Abstract zur Weiterverarbeitung an das Studiengangsekretariat frei. | Betreuer |
| 15.06.18 | Präsentation und Ausstellung der Bachelorarbeiten, 16 bis 20 Uhr | Alle |
| 15.06.18 | Abgabe des Berichts an den Betreuer bis 12:00 Uhr | Studierende |
| Ab 15.06.18 | Mündliche BA-Prüfung | Studierende |

Tabelle 1 Terminplan

Das Projekt startet also mit Beginn des Semesters am 19.02.18 und endet mit der Abgabe der Arbeit am 15.06.18.

4.2 Zeitbudget

Für die Arbeit stehen pro Student ca. 360 Arbeitsstunden über einen Zeitraum von 17 Wochen zur Verfügung. Das Semester dauert allerdings nur 14 Wochen, da eine Woche als Kompensation der Feiertag hinzugefügt wurde und die beiden letzten Wochen in der unterrichtsfreien Zeit liegen.

Das Zeitbudget pro Student wird auf 14 Wochen zu je 20 Stunden und anschliessenden 2 Wochen zu je 40 Stunden aufgeteilt. Die Osterwoche (02.04.18 – 08.04.18) dient der Kompensation der Feiertage.

4.3 Vorgehen

Das Projekt wird grundsätzlich agil in zeitlich abgegrenzten Iterationen (Sprints) durchgeführt. Bei Bedarf werden Elemente von Scrum¹ verwendet, eine zwingende Einhaltung aller Scrum-Richtlinien ist allerdings nicht gefordert.

¹ <https://www.scrumalliance.org/>

4.4 Meilensteine

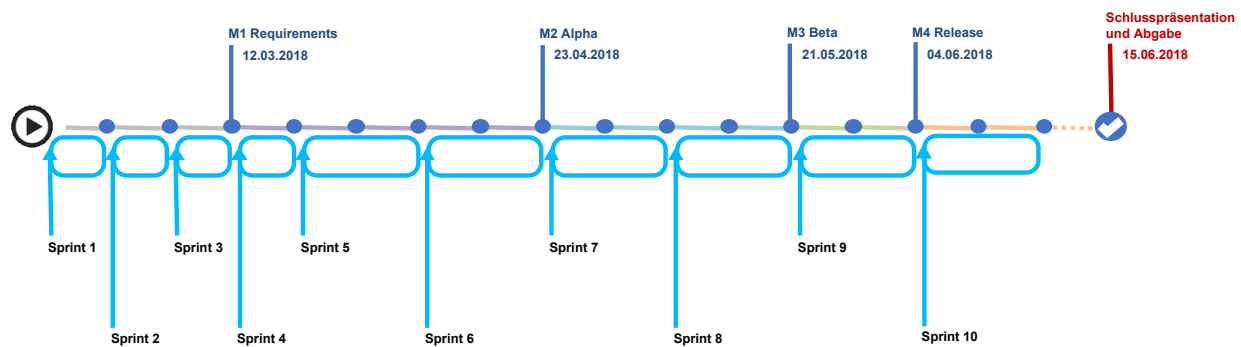


Abbildung 1 Zeitplan mit Meilensteinen

M1 Requirements

Die Anforderungen sind definiert und mit allen Beteiligten abgesprochen.

12.03.18

Work Products: Design Constraints, User Stories, nicht-funktionale Anforderungen, Prototypen

M2 Alpha

Die Architektur des Systems steht fest und ist implementiert. Es gibt einen «Durchstich» durch alle Komponenten des Systems.

23.04.18

Work Products: Executable Architecture

M3 Beta

Dieser Meilenstein markiert den Übergang in die Stabilisationsphase des Projekts. Alle Features sind zu diesem Zeitpunkt implementiert. Ab hier liegt der Fokus auf Bugfixing und Analyse (Performance, UX, etc.).

21.05.18

Feature Freeze: Ab diesem Meilenstein werden keine grösseren Features mehr eingeführt.

Work Products: Executables für Server und Clients

M4 Release

Die Applikation kann an den Auftraggeber ausgeliefert werden.

Code Freeze: Ab diesem Meilenstein gibt es keine Änderungen mehr am Quellcode. Die einzige Ausnahme sind Änderungen zur Behebung von ernsten Fehlern.

04.06.18

Work Products: Executables für Server und Clients

MP Abgabe und Präsentation

Der letzte Meilenstein ist die Abgabe und Präsentation der Arbeit.

15.06.18

Work Products: Bericht, Poster, Präsentation und Demo

4.5 Besprechungen

Es finden regelmässige Besprechungen mit dem Betreuer in ein- oder zweiwöchigen Abständen statt. Dem Auftraggeber wird regelmässig per E-Mail, Telefonkonferenz oder bei einem persönlichen Treffen Bericht erstattet.

Beim Auftauchen von organisatorischen oder technischen Herausforderungen sind diese grundsätzlich mit dem Auftraggeber zu lösen und anschliessend dem Betreuer zu berichten.

4.5.1 Protokollführung

Alle Besprechungen werden mit einer Traktandenliste vorbereitet und den Teilnehmern wenn möglich 24 Stunden vorher zugestellt.

Zu allen Besprechungen werden Protokolle geführt und auf der Projektwebsite passwortgeschützt veröffentlicht.

5 Risikomanagement

Die Risikobeurteilung führt alle projektspezifischen Risiken als Produkt aus der geschätzten Eintrittswahrscheinlichkeit und dem erwarteten Schadensausmass in Stunden auf.

5.1 Risiken

| # | Titel | Beschreibung | Massnahme | Ausmass [h] | Wahrscheinlichkeit [%] | Risiko [h] |
|---|-------------------------------------------------------------|------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------|-------------|------------------------|------------|
| 1 | Geringe Leistung des Raspberry Pi | Die Hardware des Raspberry Pi Zero (RAM, CPU, Schnittstellen) reicht für das Projekt nicht aus. | - Mit Architektur-Prototyp testen - Erneute Evaluation der Hardware | 16 | 40 | 6.4 |
| 2 | Lux-Sensor Schnittstelle nicht vorhanden oder dokumentiert | Die Lux-Sensor Werte lassen sich nicht über eine dokumentierte Schnittstelle auslesen. | - Funktionstests zu Projektbeginn - Risiko mit Kunde besprechen | 40 | 60 | 24 |
| 3 | Pozyx entspricht nicht den Anforderungen | Stabilität, Genauigkeit oder Performance von Pozyx reichen nicht aus. | - Funktionstests zu Projektbeginn - Risiko mit Kunde besprechen | 40 | 40 | 16 |
| 4 | Ausfall / Verlust der Projekthardware | Die für die Umsetzung benötigte Hardware (Raspberry Pi, Pozyx, Lux-Sensor) steht nicht mehr zur Verfügung. | - Hardware-Lieferzeiten beachten - Hardware einschliessen - ESD-Massnahmen beachten | 16 | 30 | 4.8 |
| 5 | Probleme mit Projektinfrastruktur | Datenverlust oder fehlende Verfügbarkeit der Management Tools | - Self-hosting - automatisches Backup | 16 | 20 | 3.2 |
| 6 | Verzögerungen bei der Entwicklung der Benutzerschnittstelle | Verzögerungen durch fehlende Praxiserfahrung | - Zeit für eine detaillierte Einarbeitung in die Technologie einplanen | 16 | 30 | 4.8 |

Tabelle 2 Risikoanalyse

5.2 Umgang mit Risiken

Die grössten Risiken sollen bereits im ersten Drittel des Projekts minimiert werden. Dazu werden Funktionstests durchgeführt und Prototypen erstellt, um bei Problemen möglichst schnell reagieren zu können.

Mit dem Auftraggeber wurde ausserdem besprochen, dass der Fokus dieser Arbeit in der Software-Entwicklung und Visualisierung der Sensordaten liegt. Die Ansteuerung der Sensor-Hardware oder andere nicht Software-spezifische Aufgaben können notfalls dem Auftraggeber übertragen werden.

6 Arbeitspakete

Die Arbeitspakete, Meilensteine und Sprints werden auf Jira verwaltet:

<http://jira.flux-coordinator.com>

Der mögliche Workflow der Arbeitspakete wurde vereinfacht im Zustandsdiagramm unten abgebildet.

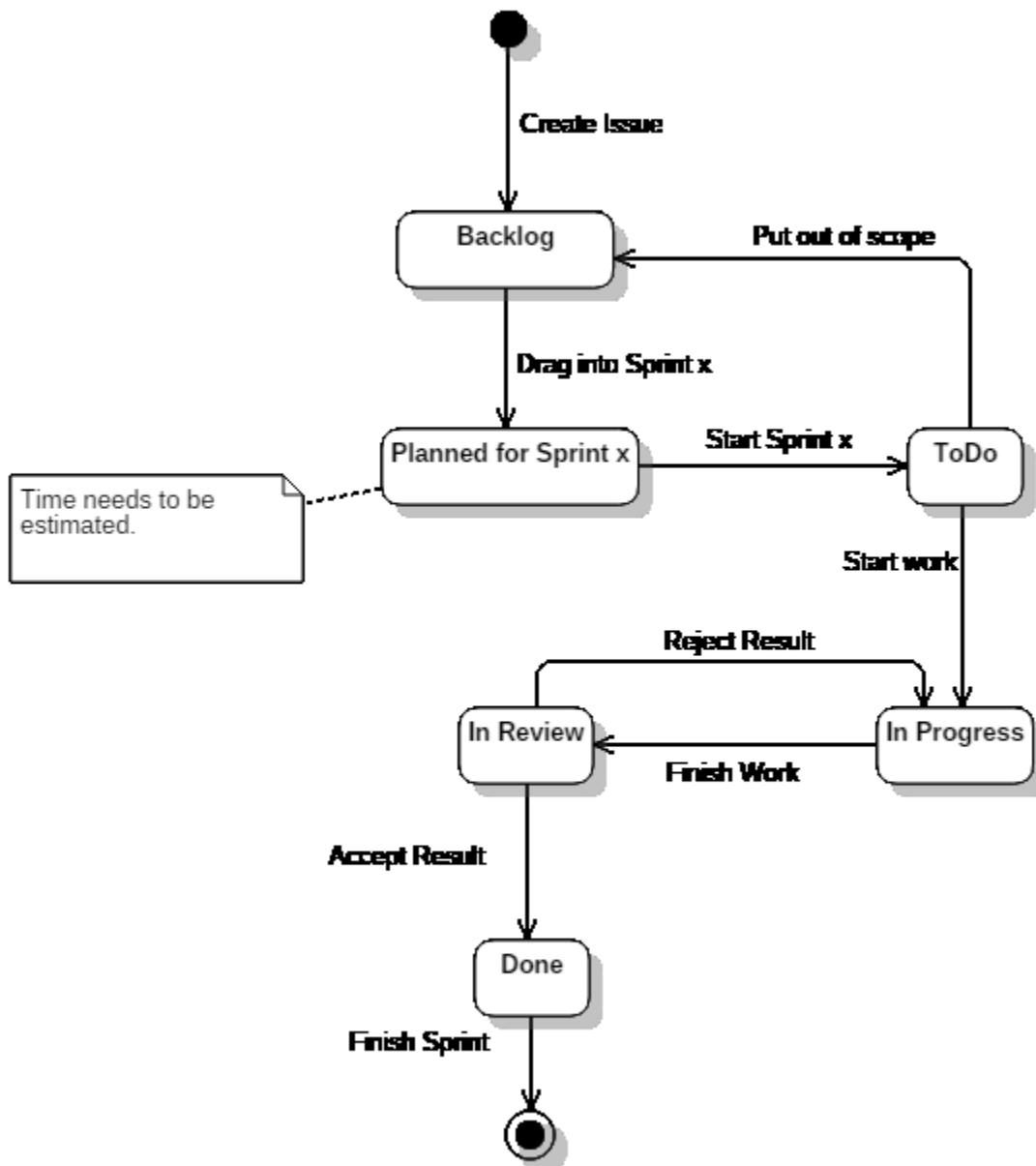


Abbildung 2 Zustandsdiagramm für Arbeitspakete

7 Infrastruktur

In diesem Kapitel wird die für diese Arbeit verwendete Infrastruktur aufgelistet und erklärt.

7.1 Projektmanagement Tool

Für das Hosten der Projektmanagement Software Jira² wird ein virtueller Server mit Ubuntu verwendet. Dieser wird von der HSR bereitgestellt und inhouse gehostet.

Hinweis: Die Konfiguration und Dokumentation der Infrastruktur ist in folgendem Repository verfügbar:

<https://github.com/Flux-Coordinator/infrastructure>

7.2 Version Control System

Der Code der verschiedenen Projekte wird auf GitHub gehostet. GitHub übernimmt auch das Betreiben der benötigten Infrastruktur.

7.3 Hosting der Server-API

Das Projekt wird eine API für die Kommunikation zwischen den verschiedenen Clients zur Verfügung stellen. Diese soll plattformunabhängig sein und auch in der «Cloud» gehostet werden können. Zum Hosting der Anwendung wird daher auch ein Server verwendet. Der Server wird vom gewählten Cloud Anbieter gehostet.

7.4 Continuous Integration & Continuous Delivery

Für das Testing der Anwendung muss ein CI/CD Dienst aufgesetzt werden. Das Hosting der Testinfrastruktur wird vom CI/CD Anbieter übernommen.

7.5 Lichtmessgeräte

Die Lichtmessungen erfordern die Verwendung von spezialisierten Messgeräten, sogenannten Luxmetern. Für die Anwendung wird ein vom Auftraggeber zur Verfügung gestellter TCS3430-EVM Sensor³ des Herstellers AMS mit einer USB/I2C Schnittstelle verwendet. Dieser Sensor kann abgesehen von den Luxwerten auch die Farbtemperatur und Farbe messen.

Im Verlaufe der Arbeit kann es nötig sein, einen zweiten Sensor oder ein dediziertes Luxmeter anzuschaffen, um den TCS3430-EVM zu kalibrieren. Dieser kann beim Auftraggeber ausgeliehen oder verwendet werden.

7.6 Positionierungssystem

Für die Positionsbestimmung wurde das «Ready to Localize Kit⁴» der Firma Pozyx ausgewählt. Dieses System ermöglicht laut Hersteller eine genaue Positionsbestimmung im Zentimeterbereich.

Das Kit umfasst folgende Komponenten:

² <https://www.atlassian.com/software/jira>

³ <http://ams.com/eng/Products/Light-Sensors/Color-Sensors/TCS3430>

⁴ <https://www.pozyx.io/store/detail/2>

- 1x Arduino kompatibler Pozyx-Tag
- 4x Pozyx-Anchor (mit Gehäuse)
- 4x Micro USB Netzteil

Trotz kleiner Hardware-Unterschiede der Pozyx-Tags und Anchors kann jedes Gerät in beiden Modi eingesetzt werden⁵. Die Auswahl des Modus geschieht per Jumper.

7.7 Weitere Infrastruktur

Zusätzlich zur oben erwähnten Infrastruktur wurde ein Raspberry Pi Zero und ein Arduino UNO besorgt. Diese sollen die Kommunikation mit den Sensoren bewerkstelligen.

Ebenfalls kann ein Hilffsystem zur Bewegung der Sensoren verwendet werden. Ein geeignetes Hilffsystem wird im Laufe der Arbeit evaluiert werden.

⁵ https://www.pozyx.io/Documentation/Tutorials/troubleshoot_basics

8 Qualitätsmassnahmen

Durch die in diesem Kapitel behandelten Massnahmen wird die Qualität der abgelieferten Software sichergestellt.

8.1 Entwicklungs-Workflow

Es wird nach dem *Gitflow Workflow* entwickelt. Anstatt einen einzigen master-Branch zu verwenden, werden feature-Branche in einen development-Branch zusammengelegt. Die dadurch entstehende Stabilisierung des master-Branche wirkt sich positiv auf eine mögliche Continuous Delivery Pipeline aus. So ändert sich eine angebotene API nicht bei jedem Commit in den development-Branch und die Releases können besser geplant werden.

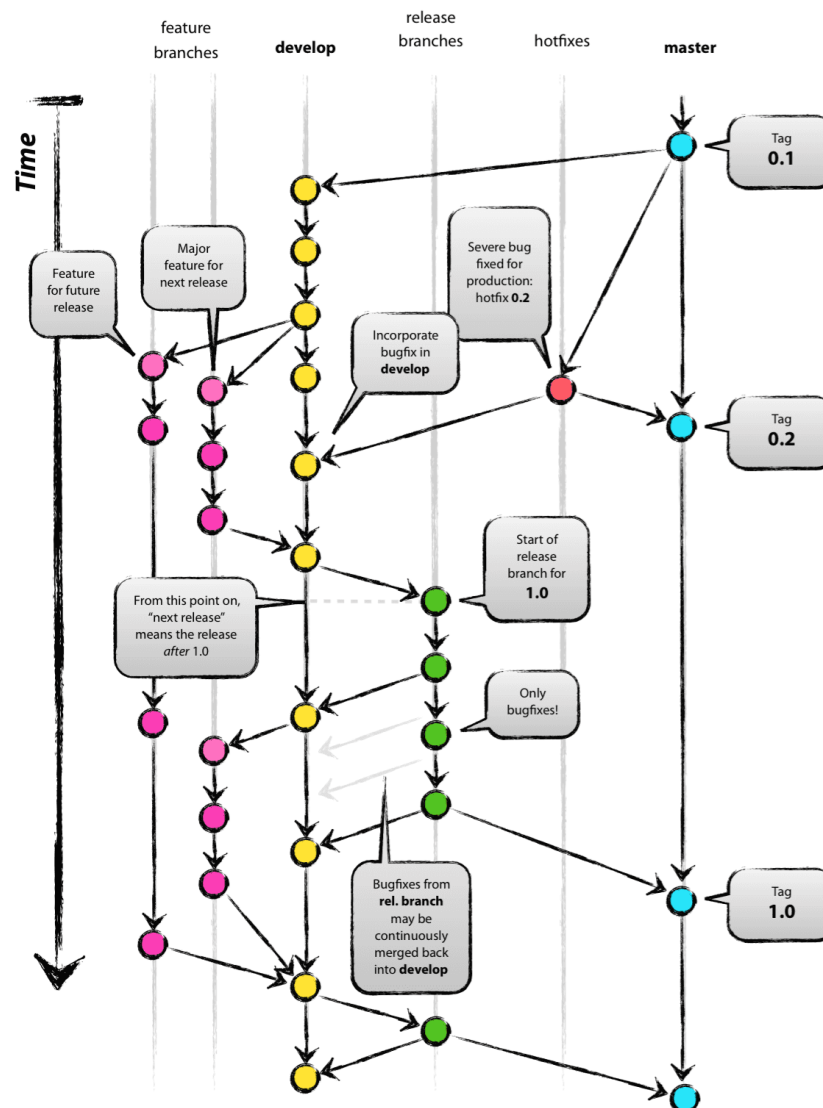


Abbildung 3 Gitflow Workflow (Quelle: Vincent Driessen, Creative Commons BY-SA)

8.2 Code Style

Da in dieser Arbeit mehrere Programmiersprachen und Werkzeuge verwendet werden, kann nicht ein einzelner Coding Style Guide verwendet werden. Wichtig zu beachten ist jedoch, dass der Programmierstil innerhalb einer Technologie einheitlich bleibt. Bei jeder neu verwendeten Programmiersprache muss deshalb beim Aufsetzen des Projektes eine Coding Style Guide ausgewählt werden und durchgehend verwendet werden. Die verwendeten Coding Style Guides müssen pro Programmiersprache sauber vermerkt werden.

8.3 Testing

Grundsätzlich sollten alle Komponenten des Systems automatisiert getestet werden können. Für Webserver gibt es eine komplette CI/CD Pipeline, welche an das VCS angeschlossen wird. Beim Pushen von Commits und vor dem Zusammenlegen von Branches wird der CI-Server die Tests automatisch durchführen und das Ergebnis anzeigen. Ein Zusammenlegen von Branches ohne die Tests zu bestehen ist nicht erwünscht und wird durch Massnahmen des VCS verhindert.

Eine Ausnahme vom automatisierten Testing bilden die hardwarenahen Komponenten, die direkt mit den Sensoren kommunizieren. Diese können nicht automatisiert getestet werden, da sie auf Werte der Sensoren angewiesen sind. Vielleicht ist es jedoch möglich, wenigstens einen Teil dieser Komponenten mit Hilfe von Sensor-Mocking zu testen.

8.4 Definition of Done

Die Definition of Done (DoD) ist eine Liste von Kriterien, die erfüllt sein müssen, damit eine Arbeit als fertig gilt. Wenn die Kriterien der DoD nicht erfüllt sind, ist das Produkt noch nicht fertig. Wichtig dabei ist, dass die Kriterien nicht schwarz-weiss, sondern im Kontext verstanden werden müssen. Auch die verschiedenen Interpretationsmöglichkeiten lassen oft keine klare Linie zwischen Einhaltung und Nichteinhaltung des DoD ziehen.

Folgende Kriterien sind einzuhalten:

| | |
|---------------------------|--------------------------------------------------------------------------------------------------------|
| Akzeptanzkriterien | Die Akzeptanzkriterien, welche in den Stories im Jira definiert wurden, sind erfüllt. |
| Code Styling | Die Richtlinien der Coding Style Guide für den Code sind erfüllt. (Siehe 8.2) |
| CI Tests | Die Tests, die auf dem Continuous Integration Dienst ausgeführt werden sind erfüllt. (Siehe 8.3) |
| Review | Falls eine Story im Voraus für einen Code-Review markiert wurde, muss dieser durchgeführt worden sein. |

Falls die Kriterien für ein Feature nur mit ungerechtfertigtem Aufwand vollständig erfüllt werden können und deshalb ausgelassen werden, müssen die Auswirkungen dokumentiert werden. Der Auftraggeber wird über den regelmässigen Statusreport informiert. Dabei sollten aber auch die Auswirkungen von technische Schulden (technical debts) im Team diskutiert und eine klare Schuldengrenze gezogen werden.

9 Abbildungsverzeichnis

| | |
|---------------------------------------------------------------------------------------|----|
| Abbildung 1 Zeitplan mit Meilensteinen | 8 |
| Abbildung 2 Zustandsdiagramm für Arbeitspakete | 11 |
| Abbildung 3 Gitflow Workflow (Quelle: Vincent Driessen, Creative Commons BY-SA) | 14 |

10 Tabellenverzeichnis

| | |
|------------------------------|----|
| Tabelle 1 Terminplan..... | 7 |
| Tabelle 2 Risikoanalyse..... | 10 |