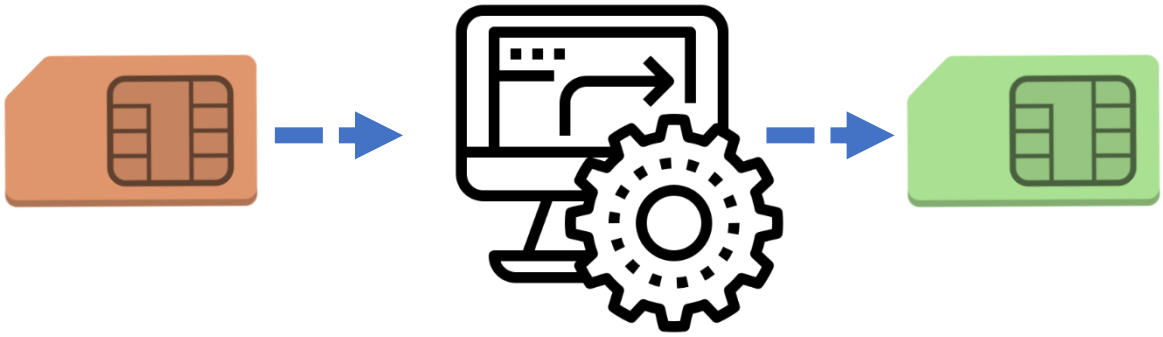


Diplomarbeit

# Automatic SIM-Card Updater- ASU

18. September 2020 - 2. November 2020

Stefano Cugis

Dipl. Techniker/in HF Informatik
Fachrichtung Systemtechnik

TEKO, höhere Fachschule

## Management Summary

### Hintergrund

Es gibts immer wieder, dass einige hundert SIM-Karten beim System Mobile Devices Team upgedatet werden müssen. Das bedeutet in der momentanen Situation, dass hier jede Karte manuell und einzeln in ein Kartenlesegerät gesteckt und anschliessend über ein Programm auf dem PC ein Update ausgeführt werden muss.

Das Problem hierbei ist, dass durch die manuelle Arbeit ein sehr hoher Zeitaufwand damit verbunden ist und ein Mitarbeiter einen halben bis einen ganzen Tag damit beschäftigt, ist diese Arbeit auszuführen. Während dieser Zeit könnte der Mitarbeiter wichtigere Aufgaben bewältigen und somit effektiver arbeiten.

### Ziele und Rahmenbedingungen:

Hierbei wurde die Aufgabe gestellt, eine Automatisierung für dieses Problem auszuführen, um eine Zeitersparnis für den Kunden zu generieren.

- Es sollte eine Zeitersparnis von mindestens 40% für den Kunden generiert werden. Zusätzlich möchte man hier auch Kosten von ca. 20% einsparen.
- Das Budget für benötigte Hardware sollte 500 CHF nicht überschreiten. Das Budgetdach liegt bei 800 CHF.
- Der Zeitaufwand sollte 250 h nicht überschreiten.
- Zeitrahmen 18. September 2020 - 2. November 2020.

### Optionenfindung

Es wurde aus den drei folgenden Optionen ausgewählt:
1. Roboter aus Lego Technics bauen.
2. Ein Roboter mit 3D gedrucktem Gehäuse bauen.
3. Schon bestehende Lösung kaufen.

Hierbei entschied man sich für den Roboter mit 3D gedruckten Gehäuse, da Lego Technics standardisierte Bauteile hat, bei welchen unsere Grössen nicht passen. Eine bestehende Lösung zu kaufen wäre zu teuer gewesen**.**

### Umsetzung

Es wurde ein 3D Gehäuse erstellt welches 25 SIM-Karten aufnehmen kann. Die unterste SIM-Karte kann über ein SIM-Karte Lesegerät ausgelesen werden und sollte per REST Anfrage an das SNOW PoS System ein Update der SIM-Karte starten. Nach Beendigung des Updates, wird die ICCID der SIM-Karte sowie der Update Status per REST an den Kunden zurückgeschickt. Diese Antwort wird dann in eine Textdatei geschrieben. Anschliessend wird die SIM-Karte ausgeworfen und macht Platz für die nächste Karte. Dieser Zyklus wird ausgeführt, bis keine SIM-Karte mehr im Ablagefach ist. Wen keine SIM-Karte mehr im Ablagefach ist, werden die abgelegten Status der SIM-Karten an den Kunden in eine E-Mail zugeschickt. Man kann den Zyklus auch per Stopp Knopf unterbrechen.

Man musste Anpassungen beim SNOW PoS System machen. Dieses war bei Beginn der Diplomarbeit noch in Entwicklung. Somit wurde anstelle des SNOW PoS Systems ein Mock-Server gesetzt, welcher REST Anfragen entgegennehmen und Antworten an den Kunden zurückgeben kann. Dieses wurde so ausgeführt, dass in einem späteren Zeitpunkt, wenn die SNOW PoS Schnittstelle zu Verfügung steht, nur kleinere Anpassungen an der Programmierung gemacht werden müssen, um diese einzubinden.

## Diplombericht

| | |
|---|---|
| Autor | Stefano Cugis |
| Erstellt | 01.11.2020 |
| Projektname | Automatic SIM-Card Updater- ASU |
| Version | 1.2 |
| Status | Abgeschlossen |

Version verlauf

| Version | Datum | Autor | Kommentar |
|---|---|---|---|
| 1.0 | 18.09.2020 | Stefano Cugis | Beginn der Dokumentation |
| 1.1 | 19.10.2020 | Stefano Cugis | Grössere Anpassung des Dokuments |
| 1.2 | 27.10.2020 | Stefano Cugis | Grammatik Korrektur |

## Inhaltsverzeichnis

## 1. Personalien


| | |
|---|---|
| Name | Cugis |
| Vorname | Stefano |
| Adresse | Winkelriedstrasse 27, 3014 Bern |
| Telefon | +41 79 723 81 85 |
| E-Mail | stefano.cugis@gmail.com |
| Geburtsdatum | 25.12.1989 |
| Heimatort | Stocken-Höfen |
| Familienstand | ledig |


*Abbildung 1 Foto Diplomand*

## 2. Beruflicher Lebenslauf

**Elektroinstallateur EFZ 2006-2011**

Nach meiner Schulzeit begann ich im Juli 2006 die Ausbildung als Elektroinstallateur EFZ. Ich arbeitete in den unterschiedlichsten Branchen wie Industrie, Militär, Wohnungsbau sowie auch in Renovationen von Denkmalgeschützen Objekten. Ich habe eine breite Palette unterschiedlichster Tätigkeiten während meiner Lehrzeit ausführen können.

**Glasfaserteam als Elektroinstallateur EFZ 2011-2014**

Nach bestandener Abschlussprüfung blieb ich im selben Betrieb und schloss mich dem Glasfaser Team an. Unsere Aufgabe bestand darin das Glasfasernetz der Aussengemeinden von Bern auszubauen.

Im Herbst 2011 erhielt ich dann die Möglichkeit die Schweizer Botschaft in Kuba zu renovieren.
Nach Beendigung dieses Auslandeinsatzes schloss ich mich wieder dem Glasfaserteam an.

**Mobile Devices als Informatiker seit 2014**

Nach 3 Jahren im Glasfaserteam brauchte ich eine neue Herausforderung und bewarb mich 2014 bei der Swisscom als Praktikant beim Mobile Devices Team. Meine Aufgaben bestanden darin, neue Software von Smartphones zu testen und diese zu evaluieren, sowie Fehleruntersuchungen zu machen und neue Features zu testen. Nach meiner Festanstellung führte ich diese Arbeiten noch weiter bis zu diesem Jahr aus.

Nun bin ich ein Technical Product Manager und bin die Technische Ansprechperson für Catepillar sowie Sony Mobile Geräte. Zusätzlich koordiniere ich alle Tests für die bei Swisscom geführten mobilen Geräten (Smartphones, Tablets).

# 3. Projektrahmen

## 3.1. Ausganslage

Im Swisscom Team „Mobile Devices“ werden diverse SIM-Karten für den Schweizer Markt getestet und konfiguriert. Pro Jahr werden schon jetzt ca. zwei Millionen SIM-Karten von verschiedenen Lieferanten vorkonfiguriert und brauchen daher keine weiteren Updates.

Trotzdem kommt es immer wieder vor, dass einige hundert SIM-Karten manuell von Hand geupdated werden müssen. Einerseits um Testkarten für neue Projekte zu konfigurieren oder eine Proof of Concept SIM-Karte zu erstellen.

Hierbei steckt der Anwender die SIM-Karte manuell in ein SIM-Karten Lesegerät, welches mit dem PC verbunden ist.

Dieser PC enthält ein Programm namens SNOW welches einem erlaubt die SIM-Karte auszulesen und ein Update auszuführen. Dies wird alles manuell per Knopfdruck gemacht. Hierbei muss jede SIM-Karte einzeln in das Lesegerät geschoben werden und das Update im SNOW gestartet werden.

SNOW ist mit einer zentralen Datenbank verbunden. Diese beinhaltet alle benötigten Daten für das Update der SIM-Karte.

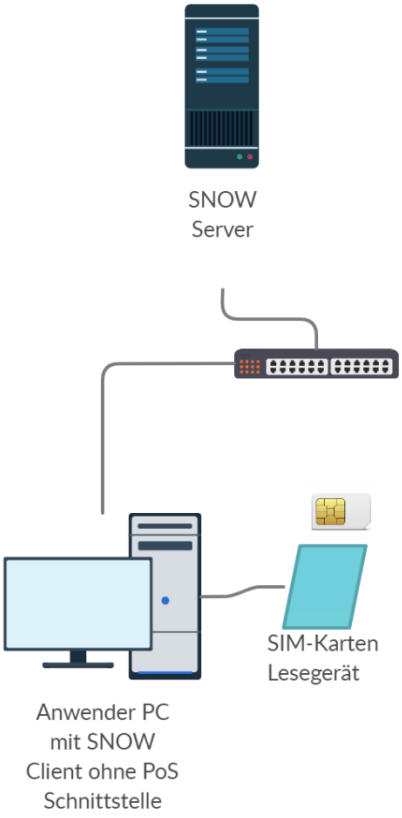



*Abbildung 2 Ausgangslage*

## 3.2. Vision

Automatic SIM-Card Updater (ASU) soll die Effizienz bei SIM-Karten Updates steigern.

Der ASU kann den Anwender dabei unterstützen Zeit einzusparen, um repetitive Arbeiten auszuführen.

Es wird ein Gehäuse gebaut, in welches man minimum 25 SIM-Karten einlegen kann.
Bei diesem wird ein SIM-Karten Leser vorhanden sein, der die unterste Karte des Stapels lesen kann. Das Lesegerät wird an einem separaten PC angeschlossen welcher mit SNOW PoS System (SIM-Karten Software) ausgestatten ist.

Der ASU wird eine REST.Post an das PoS System schicken, um ein Update zu starten.
Nach dem Update wird ein REST.Response erhalten mit folgenden Angaben ICCID und Status des Updates. Diese Informationen werden dann in eine Textdatei geschrieben.

Ein Motor treibt ein Mechanismus an, bei dem ein Schieber die eben gerade upgedatete SIM-Karte aus dem Gehäuse schiebt und somit für eine neue SIM-Karte Platz macht.
Der Zyklus fängt von vorne an.

Durch ein Lichtsensor wird die Erkennung gemacht, ob eine SIM-Karte vorhanden ist oder nicht.

Wenn keine SIM-Karte mehr vorhanden ist, stellt der Motor auf seine Nullstelle und stoppt.

Schlussendlich werden die abgefragten REST.Responses in der Textdatei in einer Benachrichtigung (E-Mail/SMS) an den Kunden gesendet.

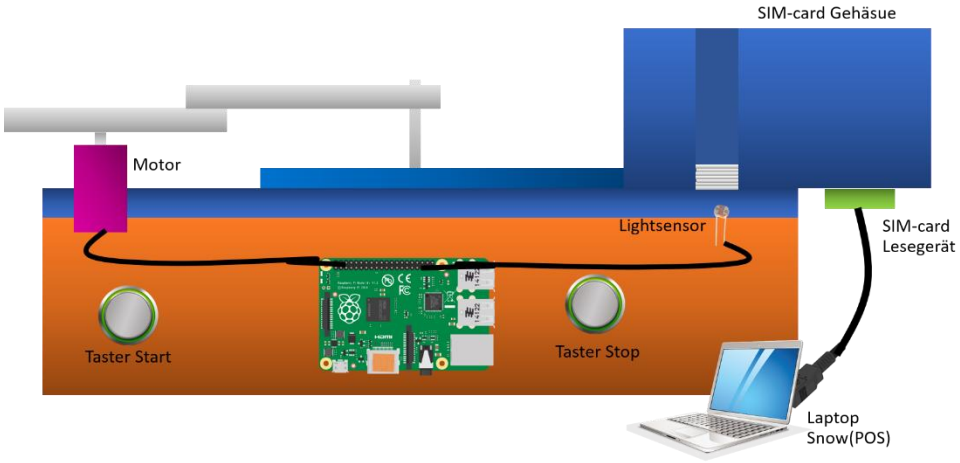



*Abbildung 3 Vision Side*

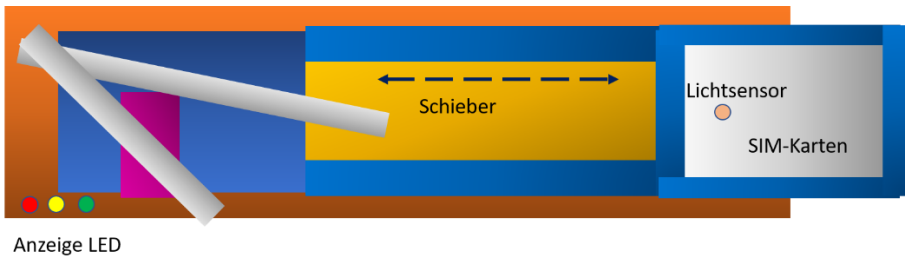



*Abbildung 4 Vision Top*

### 3.2.1. Architektur-Vision

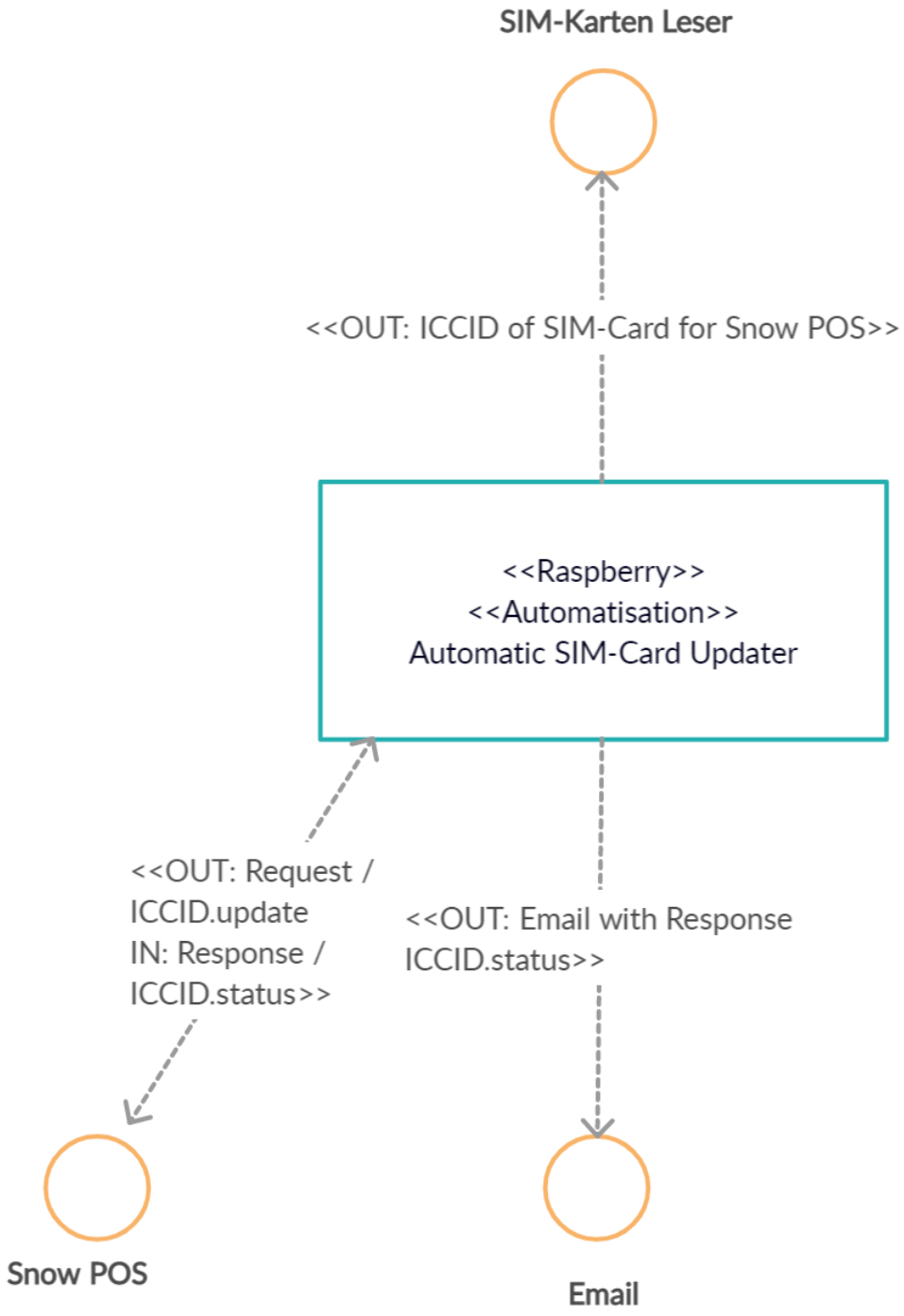



*Abbildung 5 Architektur*

### 3.2.2. Use-Case Vision

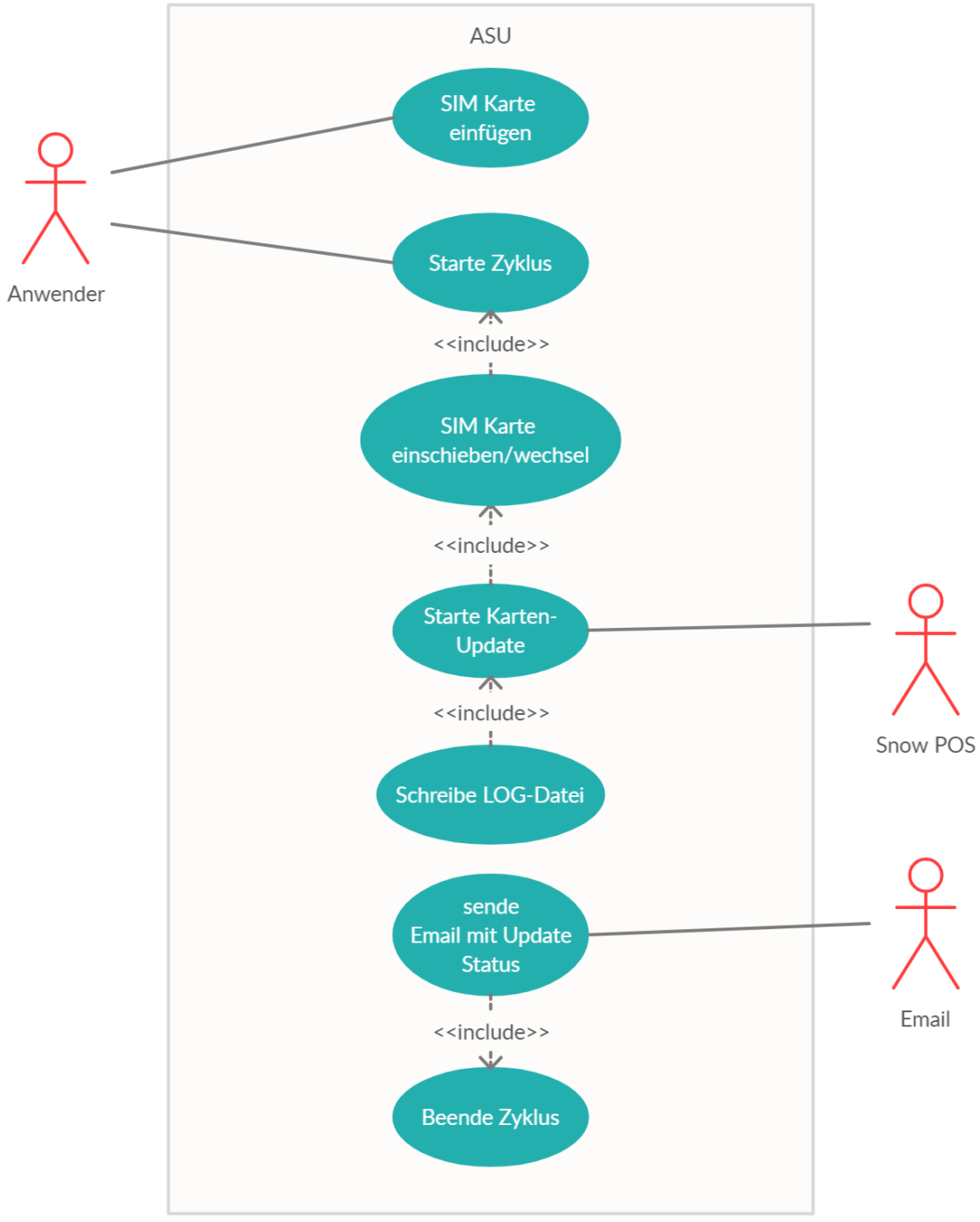



*Abbildung 6 Use-Case*

### 3.2.3. Aktivitätsdiagramm Vision

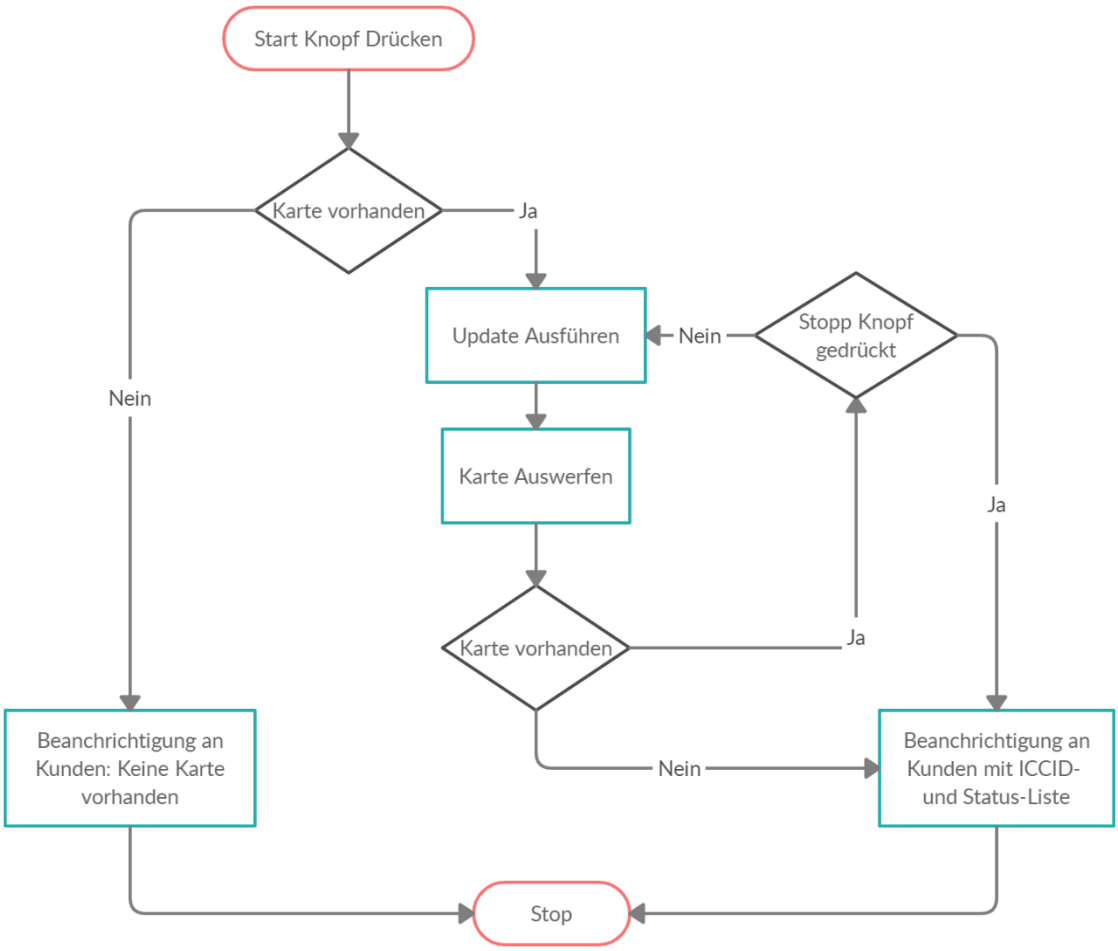



*Abbildung 7 Aktivitätsdiagramm Vision*

## 3.3. Risikoanalyse

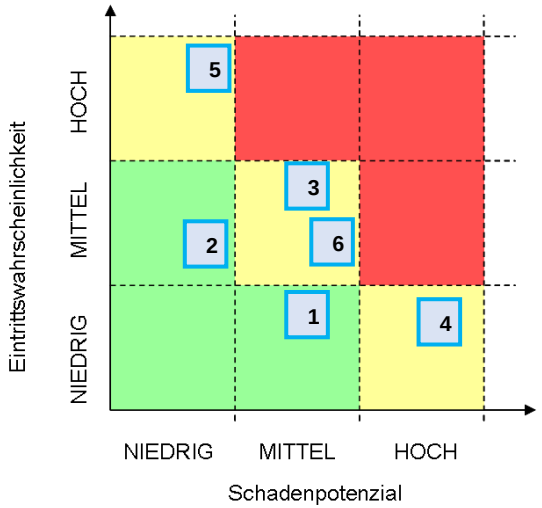



*Abbildung 8 Risikoanalyse*

| Nr. | Beschreibung | Massnahmen zur Problemlösung | Schwierigkeit zur Problemlösung |
|---|---|---|---|
| 1 | Der definierte Zeitraum reicht nicht um die Projektziele zu erfüllen | Überarbeitung der Zeitplanung<br>Zeitmängel frühzeitig erkennen und sofort handeln | Mittel |
| 2 | Ausfall durch Unfall oder Krankheit | Informieren des Expertenteams, um ggf. eine Auszeit zu nehmen oder die ausgefallene Zeit durch Überstunden nachzuholen | Leicht |
| 3 | Technische Problemstellungen, welche das Voranschreiten des Projekts verzögern und somit erschweren | Hilfe von Fachexperten einholen<br>Grösseren Aufwand in die Informationsbeschaffung investieren | Mittel |
| 4 | Fehlverhalten / Abstürze der Infrastruktur, welcher zu Verlust von Dokumenten und/oder Programmcode führt | Regelmässige Backups der jeweiligen Maschinen und deren erarbeitete Software sowie Dokumenten | Schwierig |
| 5 | Teile der gelieferten Software nicht vorhanden | Vorbereitungen treffen, dass diese auch in einem späteren Zeitpunkt eingebunden werden können. | Schwierig |
| 6 | Hardware nicht kompatibel | Hierbei müsste man neue Hardware evaluieren. | Schwierig |

*Tabelle 1 Risikoanalyse*

### 3.4. Projektabgrenzung

- Das Projekt muss innerhalb der Zeitspanne vom 18. September 2020 und dem 02. November 2020 fertiggestellt werden.
- Für das Lesen/Updaten der SIM-Karte ist nur das System SNOW PoS zuständig.
- Für den Betrieb nach dem Projektende ist Swisscom AG MOD-Team verantwortlich.
- Für das Bereitstellen des Budgets ist Swisscom AG MOD-Team verantwortlich.
- Nur Veränderungen am ASU können durchgeführt werden. Änderungen an anderen Systemen, wie zum Beispiel SNOW PoS sind nicht vorgesehen.
- Für die Umsetzung des ASU sowie der Dokumentation ist allein Stefano Cugis verantwortlich.
- Es steht ein Budget von 500 CHF für Hardware und allfällige Software zu Verfügung. Das Kostendach befindet sich bei 800 CHF.

## 3.5. Projektorganisation

Die hier präsentierte Projektorganisation wiederspiegelt die rechtliche Zuständigkeit des Projektteams.

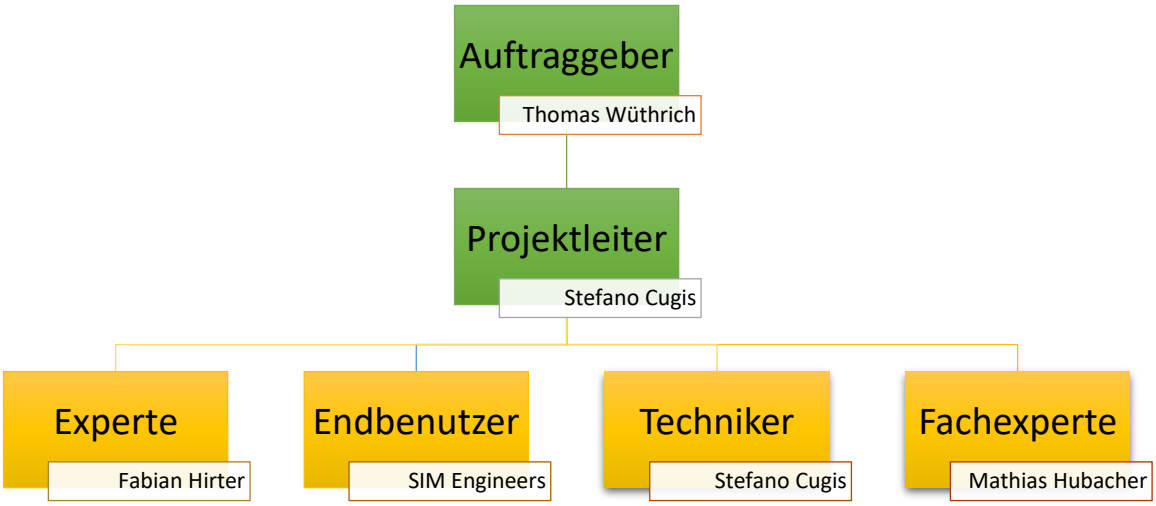



*Abbildung 9 Organisation*

| Rolle | Verantwortlichkeit |
|---|---|
| Experte | Überprüft den Stand der Arbeiten des Projekts. Bewertet die Präsentation und die Fachkenntnisse aus externer Sicht. |
| Fachexperte | Hat gute Kenntnisse mit der Materie und bewertet zusammen mit dem Experten die Präsentation und Fachkenntnisse. |
| Auftraggeber | Direkter Vorgesetzter der Projektleitung. |
| Projektleiter | Verantwortlich für die gesamte Planung, Durchführung und Präsentation des Projekts. |
| Techniker | Realisiert und dokumentiert letztendlich die technische Lösung. |
| Endbenutzer | Benutzer der finalen Lösung im Arbeitsalltag. |

*Tabelle 2 Rollen*

### 3.6. Projekttermine

In der untenstehenden Tabelle werden die wichtigsten Termine während des Verlaufes des Projektes angezeigt.

| Kalenderwoche | Projekttermine |
|---|---|
| 38/39 | Initialisierung und Unterzeichnung des Projektauftrages |
| 40 | Kick-Off-Meeting |
| 39 - 44 | Projektcontrolling |
| 39 | Start der Systemtechnikphase |
| 40 | Start der Applikationsentwicklungsphase |
| 42 | Ende der Applikationsentwicklungsphase |
| 42 | Start Testphase |
| 42 | Ende Testphase |
| 44 | Abnahme System |
| 44 | Ende der Systemtechnikphase |
| 44 | Dokumentation beendet |
| 44 | Projektabschluss Meeting |
| 45 | Projektende |

*Tabelle 3 Projekttermine*

## 3.7. Wirtschaftlichkeit

Um die Wirtschaftlichkeit des hier präsentierten Projekts noch genauer darzustellen, wurden die Berechnungen bezüglich der Zeitersparnisse und der daraus folgenden Amortisation erstellt.

### 3.7.1. Zeitersparnis

In diesem Diagramm werden die Arbeitsschritte aufgeteilt, welche manuell vom Anwender bewältigt werden müssen. Hierbei sieht man wieviel Zeit jeder Arbeitsschritt in Anspruch nimmt.

Die Zeitberechnung wurden für 100 SIM-Karten gemacht.

Durch das, dass mehr SIM-Karten in den Leser eingeschoben werden können, kann beim Einschub Zeit eingespart werden. Da man hier nicht 100-mal, sondern nur 4-mal Karten (25 Karten pro Einschub) einschieben muss.

Wie man im Diagramm sehen kann, können wir die Zeit beim Einlesen von SIM-Karten sowie Updaten von den SIM-Karten im Fall, dass man ein ASU verwendet komplett streichen, da hier der Anwender anderen Tätigkeiten nachgehen kann.

Die Kontrolle, ob die SIM-Karten korrekt geupdated wurden, entfällt hier nicht, da dies immer noch von einem Menschen überprüft werden muss.

Der Kartenleser als auch der ASU können Fehler verursachen, welche vom Anwender gelöst werden müssen. Hier wird wohl eine minimale Zeitersparnis möglich sein.

Schlussendlich kann man sehen, dass die Zeit bei der manuell etwas gemacht werden muss, von 127 auf 34 Minuten reduziert werden konnte.

Somit kann man eine Zeitersparnis von ca. **73%** erreichen.

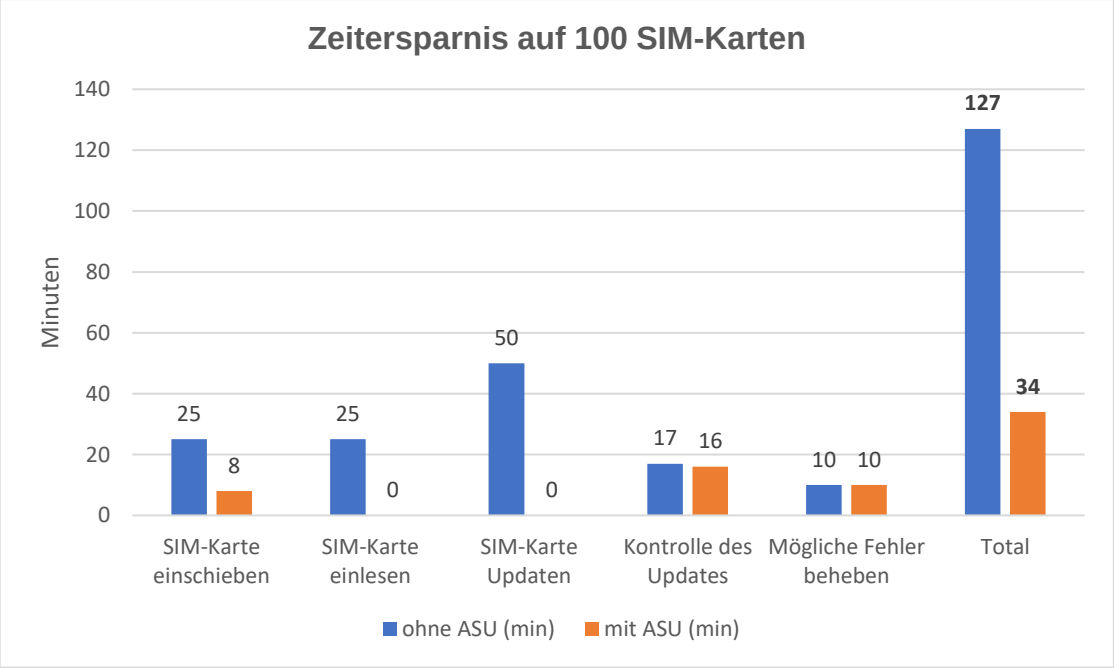



*Abbildung 10 Zeitersparnis*

### 3.7.2. Amortisation

Die Amortisation wurde wie folgt berechnet.

Es wurden 200 (50 CHF/h) Arbeitsstunden sowie 500 CHF Materialkosten für das Projekt als Ausgangslage genommen.

Man nimmt an, dass ca. sechs Mal im Jahr 600 Karten geupdated werden müssen. Das heisst, dass man ohne ASU 63.5 Stunden pro Jahr mit Updaten beschäftigt ist.

Mit ASU kann man diese Zeit reduzieren auf ca. 17 Stunden.

Somit spart man pro Jahr 2325 CHF an Ausgaben. Bis allerdings die Ausgaben vom Projekt amortisiert sind, vergehen etwas mehr als 4 Jahre.

Nach diesen vier Jahren beläuft sich die Kosteinsparung auch auf ca. **73%.**

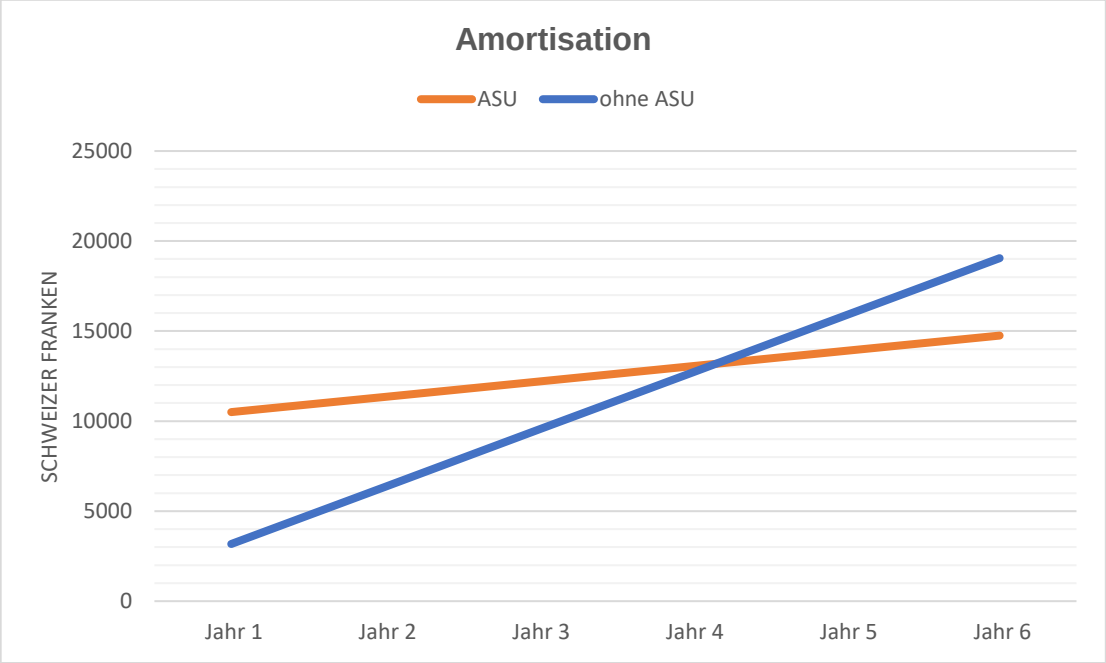



*Abbildung 11 Amortisation*

### 3.8. Projektbudget

Das Projektbudget ist in einem Minimum zu halten.

Es steht ein Budget von 500 CHF für Hardware und allfällige Software zur Verfügung. Kostendach befindet sich bei 800 CHF.

Für die Dokumentation und Arbeit stehen maximum 250 h zur Verfügung. Für detaillierte Angaben des Budgets, siehe Punkt 6, Budget Kostenaufteilung.

### 3.9. Schnittstellen

Mathias Hubacher, Experte SIM-Karten und dazugehörige Systeme.

Christoph Moser, Experte Linux Betriebssysteme.

### 3.10. Restriktionen

Dieser Automatisation Roboter muss den geltenden Datenschutzgesetzen der Schweizer Eidgenossenschaft entsprechen.

Gestützt auf Artikel 13 der schweizerischen Bundesverfassung.

Zusätzlich sollten hier die geltenden NDAs der Swisscom berücksichtigt werden, da hier zum Teil geheime Daten der Stufe C2-C4 (Sensitive; Confidential; strictly Confidential) auf dem mit SIM-Software versehenen Computer vorhanden sein werden.

Da hier ein Gmail-Account verwendet wird, müssen wir den geltenden AGBs von Google gerecht werden.

### 3.11. Information und Berichterstattung

- Der Informationsaustausch zwischen dem Auftraggeber und dem Experten, wird durch Stefano Cugis sichergestellt.
- Das Projekt ist sorgfältig zu dokumentieren, damit bei allfälligen Rechtsstreitigkeiten die Verantwortlichkeiten klar sind.
  Die sorgfältige Dokumentation dient auch dazu, ähnliche oder gleiche planerische Tätigkeiten für zukünftige Projekte zu minimieren.

## 3.12. Projektstrukturplan

Der Projektstrukturplan, ist das zentrale Planungs-, Kommunikations- und Controlling-Instrument in diesem Projekt.

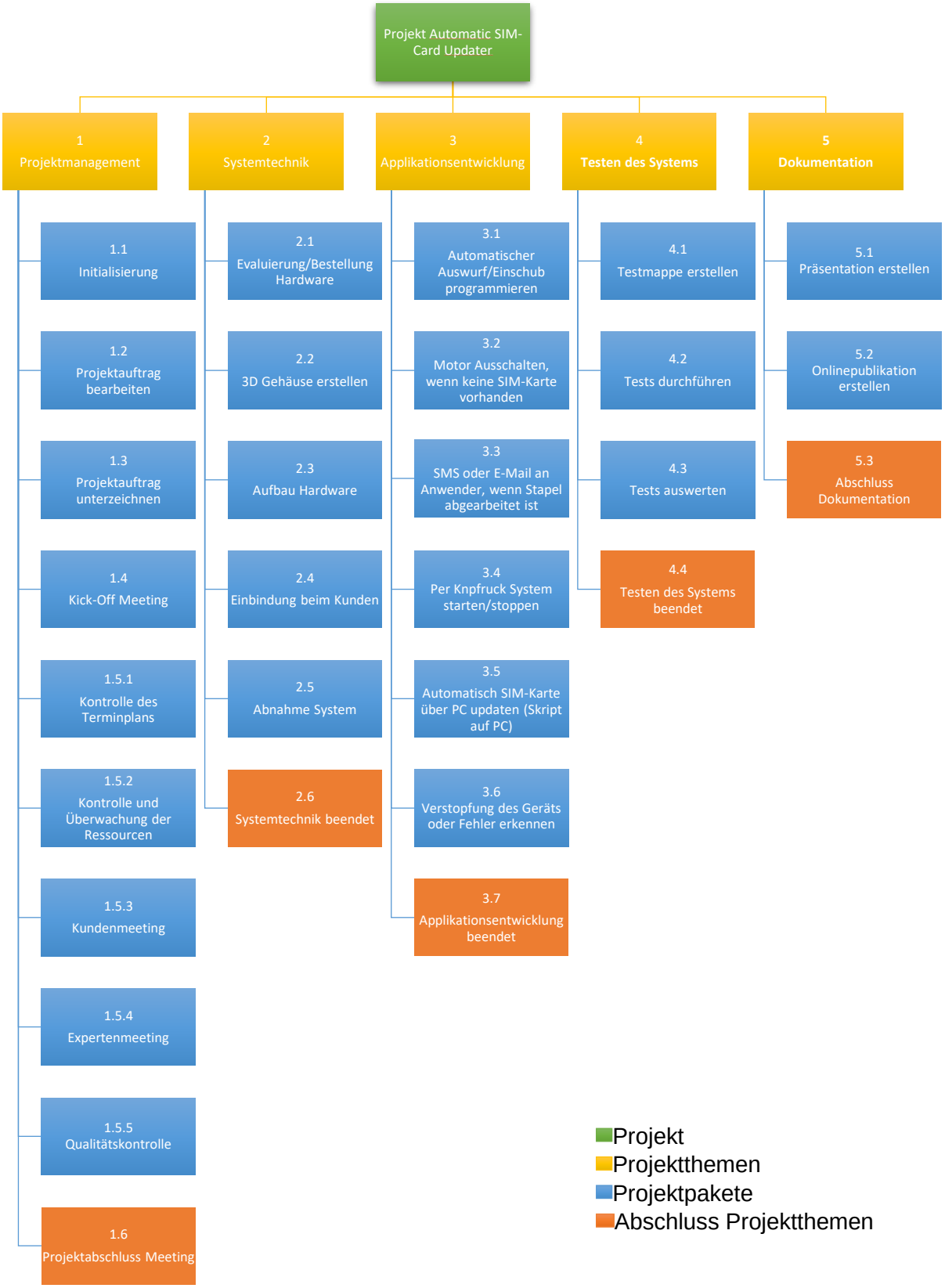



*Abbildung 12 Projektstrukturplan*

## 3.13. Vorgangsliste

| ID | Vorgang | Soll in h | Verantwortlich | **Vorgänger** | **Meilensteine** |
|---|---|---|---|---|---|
| | | | | | |
| 1 | **Projektmanagement** | | | | |
| 1.1 | Initialisierung | 2 | Stefano Cugis | | |
| 1.2 | Projektauftrag bearbeiten | 12 | Stefano Cugis | 1.1 | |
| 1.3 | Projektauftrag unterzeichnen | 0 | Stefano Cugis, Thomas Wüthrich | 1.2 | |
| 1.4 | Kick Off Meeting | 1.5 | Fabian Hirter<br>Mathias Hubacher<br>Thomas Wüthrich<br>Stefano Cugis | 1.3 | X |
| | | | | | |
| 1.5 | *Projektcontrolling* | | | | |
| 1.5.1 | Kontrolle des Terminplans | 1.5 | Stefano Cugis und Mathias Hubacher | | |
| 1.5.2 | Kontrolle und Überwachung der Ressourcen | 1.5 | Stefano Cugis und Mathias Hubacher | | |
| 1.5.3 | Kundenmeeting | 5 | Stefano Cugis<br>Mathias Hubacher<br>Thomas Wüthrich | | |
| 1.5.4 | Expertenmeeting | 1 | Stefano Cugis und Fabian Hirter | 1.3 | |
| 1.5.5 | Qualitätskontrolle | 3 | Stefano Cugis | | |
| 1.6 | Projektabschluss Meeting | 2 | Stefano Cugis<br>Mathias Hubacher<br>Thomas Wüthrich | 5.3 | X |
| | | | | | |
| 2 | **Systemtechnik** | | | | |
| 2.1 | Evaluierung/Bestellung Hardware | 8 | Stefano Cugis | | |
| 2.2 | 3D Gehäuse erstellen | 2 | Stefano Cugis | | |
| 2.3 | Aufbau Hardware | 15 | Stefano Cugis | 2.2 | |
| 2.4 | Einbindung beim Kunden | 2 | Stefano Cugis | 4.4 | |
| 2.5 | Abnahme System | 2 | Stefano Cugis und Mathias Hubacher | 2.4 | |
| 2.6 | Systemtechnik beendet | | Stefano Cugis | 2.5 | X |
| | | | | | |
| 3 | **Applikationsentwicklung** | | | | |
| 3.1 | Automatischer Auswurf/Einschub programmieren | 4 | Stefano Cugis | 2.3 | |
| 3.2 | Motor Ausschalten, wenn keine SIM-Karte vorhanden | 5 | Stefano Cugis | 3.1 | |
| 3.3 | SMS oder E-Mail an Anwender, wenn Stapel abgearbeitet ist | 4 | Stefano Cugis | 3.2 | |
| 3.4 | Per Knopfdruck System starten/stoppen | 2 | Stefano Cugis | 3.3 | |
| 3.5 | Automatisch SIM-Karte über PC updaten (Skript auf PC) | 20 | Stefano Cugis | 3.4 | |

| | | | | | |
|---|---|---|---|---|---|
| 3.6 | Verstopfung des Geräts oder Fehler erkennen | 6 | Stefano Cugis | 3.5 | |
| 3.7 | Applikationsentwicklung beendet | | Stefano Cugis | 3.6 | X |
| | | | | | |
| 4 | **Testen des Systems** | | | | |
| 4.1 | Testmappe erstellen | 7.5 | Stefano Cugis | | |
| 4.2 | Tests durchführen | 6 | Stefano Cugis | 4.1 | |
| 4.3 | Tests auswerten | 2 | Stefano Cugis, Mathias Hubacher | 4.2 | |
| 4.4 | Testen von System beendet | | | 4.3 | X |
| | | | | | |
| 5 | **Dokumentation** | 47.5 | | | |
| 5.1 | Präsentation erstellen | 10 | Stefano Cugis | 4.4 | |
| 5.2 | Onlinepublikation erstellen | 3 | Stefano Cugis | 4.4 | |
| 5.3 | Dokumentation beendet | | | 5.2 | X |
| | | | | | |
| 6 | Projekt Ende | | | 5.4 | X |

*Tabelle 4 Vorgangsliste*

Bei der Vorgangsliste wurde sich an das Beispiel vom Buch “Projektmanagement für Führungsfachleute” vom “Compendio” Verlag gehalten. Der Aufbau ist simpel und übersichtlich, die Arbeitspakete ergaben sich aus dem Projektstrukturplan.

## 3.14. Ansprechpartner

Auftraggeber: Swisscom AG
Herr T. Wüthrich
Auftraggeber
Waldeggstrasse 51
3097 Liebefeld
Mail: Thomas.Wuethrich@swisscom.com
Telefon: +41 58 221 43 36

Gesamtprojektleitung: Swisscom AG
Herr S. Cugis
Projektleiter
Winkelriedstrasse 27
3014 Bern
Mail: stefano.cugis@gmail.com
Telefon: +41 79 723 81 85

## 3.15. Unterschrift

Ort / Datum / Auftraggeber

*Liebefeld, 23.09.2020*

Swisscom AG T. Wüthrich

T. Wüthrich

*Bestätigung durch E-Mail am 23.09.2020*

Ort / Datum / Projektleitung

*Liebefeld, 23.09.2020*

Swisscom AG S. Cugis

# 4. Ziele

## 4.1. Zielsetzung (soll)

Soll-Ziele sind Ziele, die vom Auftraggeber ausdrücklich gewünscht sind. Anders als bei Kann-Zielen, besteht hier nicht die Möglichkeit, Ziele zu streichen. Nach Rücksprache mit dem Auftraggeber können Änderungen an Kann-Zielen gemacht werden.

### 4.1.1. [2.1] Evaluierung/Bestellung Hardware

Für das Projekt muss spezifische Hardware bestellt werden. Diese muss vorab evaluiert werden um den Ansprüchen, des Kunden gerecht zu werden.

### 4.1.2. [2.2] 3D Gehäuse erstellen

Um einen SIM-Karten Stapel sauber auf ein SIM-Karte Lesegerät zu bringen, muss ein Passgenaues Gehäuse vorhanden sein. Dieses muss in einer 3D Dateiformat erstellt werden, um es anschliessend bei einer Drucker Firma drucken zu lassen.

### 4.1.3. [2.3] Aufbau Hardware

Zusammenbau der bestellten Hardware.

Basierend auf einer Grundskizze sollen die bestellte Komponente zusammengebaut werden. Hierbei ist zu bemerken das dies ein Prototyp ist.

### 4.1.4. [2.4] Einbindung beim Kunden

Der ASU muss beim Kunden im Büro eingebunden werden.
Das Gerät muss an ein Netzwerk angebunden werden, sowie durch USB-Schnittstelle an den mit SIM-Karten Software versehenen PC angeschlossen werden.

Zusätzlich muss ein Funktionstest durchführen werden.

### 4.1.5. [3.1] Automatischer Auswurf/Einschub programmieren

Es muss ein Auswurf programmiert werden. Hierbei soll der Motor, welcher an einer Mechanik angeschlossen ist mit einem Schieber die Karten aus dem 3D Gehäuse auswerfen können.

### 4.1.6. [3.2] Motor Ausschalten, wenn keine SIM-Karte vorhanden

Der Motor muss in einem Zyklus des Auswerfens laufen solange eine SIM-Karte vorhanden ist.
Wird erkannt, dass keine SIM-Karte vorhanden ist, soll der Motor auf die Startposition zurückgehen und den Zyklus anhalten.

### 4.1.7. [3.3] SMS oder E-Mail an Anwender, wenn Stapel abgearbeitet ist

Nach abgearbeitetem SIM-Karten Stapel soll der ASU eine Benachrichtigung an den Kunden schicken.

Hierbei sollte man auch den Negativfall berücksichtigen. Falls von Anfang an keine Karte verfügbar ist, sollte auch hier eine Benachrichtigung an den Kunden gesendet werden.

### 4.1.8. [3.4] Per Knopfdruck System starten/stoppen

Das System muss per Knopfdruck gestartet sowie auch gestoppt werden können.

Der Start und Stopp Knopf muss nach Ausschalten und erneutem aufstarten des Systems direkt funktionieren ohne zusätzliche Einstellungen am System vorzunehmen.

#### 4.1.9. [3.5] Automatisch SIM-Karte über PC updaten (Skript auf PC)

Eine eingelesene SIM-Karte sollte über das Tool SNOW PoS automatisch ein Update ausführen können.

Folgende Schritte sollten beim Update passieren.
1. ICCID aus der SIM-Karte auslesen
2. In vorhandener Datenbank nach ICCID suchen
3. Neuste verfügbare Version auf die SIM-Karte updaten

### 4.2. Zielsetzung (kann)

#### 4.2.1. [3.6] Verstopfung des Geräts oder Fehler durch Motor erkennen

Bei einer Automatisierung besteht immer die Möglichkeit das Fehler auftreten können. Hierbei sollte das System diese erkennen können und dem Benutzer über SMS/E-Mail benachrichtigen mit dem Inhalt welcher Fehler entstanden ist.

## 5. Terminplan

Der Terminplan ist zu den Meilensteinen verknüpft mit bedeutenden Teilzielen des Projekts wie etwa die Programmierung einzelner Funktionalitäten oder der Beginn der Testphase. Das Projektmanagement erstellt ihn aufbauend auf dem Projektstrukturplan.

| |
|---|
| Kritischer Pfad |
| Nicht kritische Prozesse |
| Meilensteine |
| Laufende Prozesse |

| | | | Woche 38 | | | Woche 39 | | | | | | | Woche 40 | | | | | | | Woche 41 | | | | | | | Woche 42 | | | | | | | Woche 43 | | | | | | | Woche 44 | | | | | | | 45 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Datum | 18 - 20.09.2020 | | | 21 - 27.09.2020 | | | | | | | 28.09 - 04.10.2020 | | | | | | | 05 - 11.10.2020 | | | | | | | 12 - 18.10.2020 | | | | | | | 19 - 25.10.2020 | | | | | | | 26.10 - 01.11.2020 | | | | | | | 02.11 |
| | SOLL | IST | Fr | Sa | So | Mo | Di | Mi | Do | Fr | Sa | So | Mo | Di | Mi | Do | Fr | Sa | So | Mo | Di | Mi | Do | Fr | Sa | So | Mo | Di | Mi | Do | Fr | Sa | So | Mo | Di | Mi | Do | Fr | Sa | So | Mo | Di | Mi | Do | Fr | Sa | So | Mo |
| [1] Projektmanagement | 17.5 | 17.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.1] Initialisieurng | 2 | 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.2] Projektauftrag erstellen | 12 | 12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.3] Projektauftrag unterzeichnen | 0 | 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.4] Kick-Off Meeting | 1.5 | 0.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.6] Projektabschlussmeeting | 2 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Projektcontrolling | 12 | 11.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.5.1] Kontrolle des Terminplans | 1.5 | 1.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.5.2] Kontrolle und Überwachung der Ressourcen | 1.5 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.5.3] Kundenmeeting | 5 | 5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.5.4] Expertenmeeting | 1 | 1 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [1.5.5] Qualitätskontrolle | 3 | 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [2] Systemtechnik | 29 | 30.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [2.1] Evaluierung/Bestellung Hardware | 8 | 9.5 | Diese Aufgaben wurden aus Zeitgünden und auf Anweiseung durch den Experten schon vor dem Start der Diplomarbeit abgeschlossen. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [2.2] 3D Gehäuse erstellen | 2 | 2.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [2.3] Aufbau Hardware | 15 | 14.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [2.4] Einbindung beim Kunden | 2 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [2.5] Abnahme System | 2 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [2.6] Systemtechnik beendet | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [3] Applikationsentwicklung | 41 | 40.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [3.1] Automatischer Auswurf/Einschub programmieren | 4 | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [3.2] Motor Ausschalten, wenn keine SIM-Karte vorhanden | 5 | 5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [3.3] SMS oder E-Mail an Anwender, wenn Stapel abgearbeitet ist | 4 | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [3.4] Per Knopfdruck System starten/stoppen | 2 | 5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [3.5] Automatisch SIM-Karte über PC updaten (Skript auf PC) | 20 | 18.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [3.6] Verstopfung des Geräts oder Fehler durch Motor erkennen | 6 | 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [3.7] Applikationsentwicklung beendet | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [4]Testen des Systems | 15.5 | 14.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [4.1]Testmappe erstellen | 7.5 | 6.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [4.2]Tests durchführen | 6 | 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [4.3]Tests Auswerten | 2 | 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [4.4]Testen von Systems beendet | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [5] Dokumentation | 60.5 | 67.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [5.1] Dokumentation bearbeiten | 47.5 | 54.5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [5.2] Präsentation erstellen | 10 | 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [5.3] Onlinepublikation erstellen | 3 | 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [5.4] Dokumentation beendet | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| [6] Projekt Ende | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Total | 175.5 | 182 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

*Tabelle 5 Terminplan*

## 6. Budget Kostenaufteilung

| **Hardware** | | Preise in CHF |
|---|---|---|
| Raspberry Pi 3 Model B + (ARMv8) | | 51.30 |
| Joy-it USB-Ladekabel Switch | | 12.10 |
| Lichtsensor | | 6.60 |
| Jumperkabel 50 Stück / Verbindungskabel (10/20cm). - 24AWG<br>Jumperkabel 20 Stück / Verbindungskabel (30cm) - 26AWG | | 25.10 |
| Dsservo 20kg.cm Digital Power Servo DS3218MG | | 48.90 |
| Taster | | 7.90 |
| Gehäuse Holzbox | | 5.00 |
| 1-Kanal Relais Modul Karte 5V / 30VDC mit Kabel | | 20.90 |
| Kleinmaterial (Schrauben, Kabel, LED) | | 30.00 |
| Kartenleser | | 10.10 |
| Ladegerät | | 4.10 |
| **Hardware Total** | | **222.00** |
| | | |
| **Kosten externe Firmen** | | Preise in CHF |
| 3D Modell aus 2D Skizze erstellen | | 41.40 |
| 3D Modell drucken | | 140.60 |
| **Kosten externe Firmen Total** | | **182.00** |
| | | |
| **Effektive Arbeit /** Stefano Cugis 50. Fr/h | Stunden | Preise in CHF |
| [1] Projektmanagement | 17.5 | 875.00 |
| Projektcontrolling | 11.5 | 575.00 |
| [2] Systemtechnik | 30.5 | 1525.00 |
| [3] Applikationsentwicklung | 40.5 | 2025.00 |
| [4] Testen des Systems | 14.5 | 725.00 |
| [5] Dokumentation | 67.5 | 3375.00 |
| **Effektive Arbeit Total** | **182** | **9100.00** |
| | | |
| **Total Ausgaben** | | **9504.00** |

*Tabelle 6 Kostenaufteilung*

In der Budget-Kostenaufteilung werden ausschliesslich die Stunden des Projekt-Teams aufgezeigt.

Dies heisst, man stellt hier nur die effektiven Stunden, die ein oder mehrere Teammitglieder benötigt haben, um eine Arbeit zu bewältigen. Die effektive Zeit, die eine externe Firma für eine Arbeit benötigt, wird hier nicht dargestellt.

## 7. Dokumentation

In den folgenden Abschnitten wird beschrieben wie die Ziele umgesetzt werden. Man folgt dem Projektstrukturplan und fängt mit dem Punkt *[ID-4.1] Testmappe erstellen* an. Dies ist wichtig, um mögliche Fehler des Systems schon vor der Umsetzung der restlichen Punkte zu erkennen.

Anschliessend wird wie im Projektstrukturplan bei *[ID-2.3] Aufbau Hardware* weitergearbeitet und Punkt für Punkt nach dem Terminplan abgearbeitet.

Durchgehend werden während den Arbeiten immer wieder Qualitätskontrollen und Überwachungen der Ressourcen durchgeführt. Die Wochen, in welchen diese gemacht werden, sind im Terminplan ersichtlich. Hierbei hält man sich an das untenstehende Wasserfallmodell.

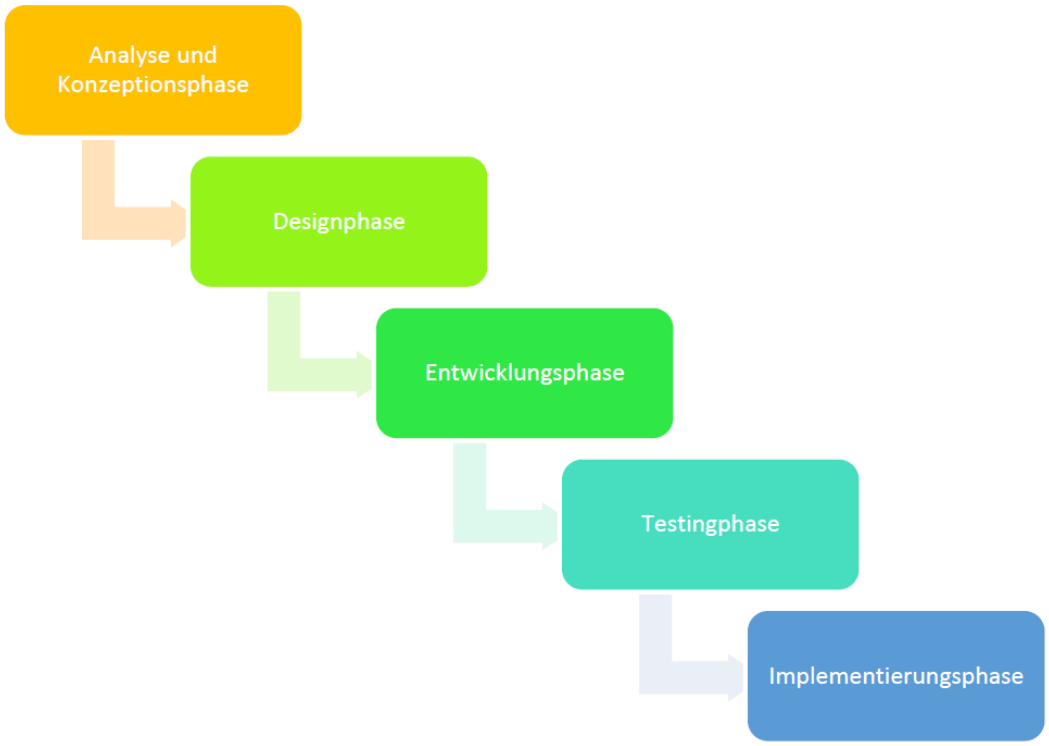



*Abbildung 13 Umsetzung der Ziele*

## 7.1. [2.1] Evaluierung/Bestellung Hardware

Für das Projekt ASU braucht es einige Spezifische Materialien, die man evaluieren muss, Hierzu gehört der Motor, Lichtsensor und das Entwicklungsboard (Raspberry Pi)

| Raspberry Pi Evaluierung | | |
|---|---|---|
| | Raspberry Pi 3 Model B | Raspberry Pi 4 4G Model B |
| **Bild** |  *Abbildung 14 Raspberry Pi 3 Model B* |  *Abbildung 15 Raspberry Pi 4 Model B* |
| **Taktrate (MHz)** | 1400 | 1500 |
| **Arbeitsspeicher** | 1000 | 4000 |
| **Anzahl I/O** | 40 | 40 |
| **HDMI** | Yes | Yes (microHDMI) |
| **Preis (CHF)** | 51.60 | 59.90 + (13.30 HDMI Adapter) |
| **Endergebnis** | Beim Entwicklerboard hat man sich für das Raspberry Pi 3 Model B entschieden. Die Gründe dafür waren die Kosten sowie der HDMI Anschluss. Da wir heutzutage nicht standartmässig microHDMI verwenden, müsste man hier noch ein Adapter dazu kaufen.<br>Der Arbeitsspeicher kann vernachlässigt werden, da wir kein leistungsintensives Programm auf der Raspberry Pi ausführen werden. | |

*Tabelle 7 Raspberry Pi Evaluierung*

| Motor Evaluierung | | |
|---|---|---|
| | Schrittmotor (MF-6402411) | Servomotor (DS3218MG) |
| **Bild** |  *Abbildung 16 Schrittmotor (MF-6402411)* |  *Abbildung 17 Servomotor (DS3218MG)* |
| **Drehmoment** | 0.6 kg-cm | 21.5 kg-cm |
| **Betriebsspannung** | 5V DC | 5V DC |
| **Rotation** | 360 Grad | 180 Grad |
| **Getriebe** | Plastik | Metall |
| **Positionserkennung** | Nein | Ja |
| **Preis (CHF)** | 6.30 | 39.90 |
| **Endergebnis** | Bei dem Motor fiel die Entscheidung auf den Servomotor. Die Gründe dafür waren das stärkere Drehmoment und was noch wichtiger war, war die Positionserkennung. Diese müsste man beim Schrittmotor noch zusätzlich bauen. Heisst zum Beispiel ein Lichtsensor, der erkennt ob der Einschub aus dem SIM-fach ist oder nicht. | |

*Tabelle 8 Motor Evaluierung*

| Lichtsensor Evaluierung | | |
|---|---|---|
| | Lichtsensor (DN1519) | Lichtsensor (TSL2591) |
| **Bild** |  *Abbildung 18Lichtsensor (DN1519)* |  *Abbildung 19 Lichtsensor (TSL2591)* |
| **Dioden** | Lux | Lux / Infrarot / Vollspektrum |
| **Output** | Digital / Analog | Digital / Analog |
| **Luxeinstellung** | Ja | Nein |
| **Preis (CHF)** | 6.55 (+10 Versand) | 8.30 (+10 Versand) |
| **Endergebnis** | Beim Lichtsensor hat man sich für den Lichtsensor (DN1519) entschieden da er erstens günstiger ist, zusätzlich kann man manuell einstellen, ab welchem LUX-Wert beim Digital-Ausgang True ausgegeben wird. | |

*Tabelle 9 Lichtsensor Evaluierung*

Um das restliche Hardware nach den Vorgaben der Vision umsetzen zu können, wurden noch folgende Materialen bestellt und in Baumärkten besorgt.

Zusätzlich online bestellte Ware

- 1x Jumperkabel 50 Stück
- 1x SD-Karte (min. 16GB)
- 1x Relais
- 1x Smartcard Lesegerät

Folgendes Material wurde in einem Baumarkt besorgt.

- 10x Schrauben M3 mit Unterlagscheiben und Mutter
- 1x 0.5 mm Blech ca 20x30 cm
- 1x Winkelstange Alu 1x0.5 cm ca. 50 cm Lang
- 1x Holzkiste ca. 30x15x10 cm
- 5x Klemmen für Drähte
- 2x Taster
- 10x Holzschrauben 3x10mm
- 10x Holzschrauben 3x40mm

Folgendes Material war schon vorhanden durch ein älteres Projekt.

- 1x Led grün/gelb/rot
- 1x 330 Ohm Wiederstand
- 2x USB-Ladegerät Minimum 5V/1A

## 7.2. [2.2] 3D Gehäuse erstellen

Um aus einem 2D Zeichnung ein 3D-Modell zu erstellen, benötigt man einen sauberen Plan mit genauen Massangaben.

Hierbei wurden die Masse der SIM-Karte als Grundstein genommen. Als erstes wurde das Gehäuse (blau) erstellt und nach den genauen Massen der SIM-Karten erstellt. Hierbei wurde auf der Länge und Breite 1 mm hinzugefügt, um genug Platz zu machen, dass die SIM-Karten nicht hängenbleiben.

Zusätzlich wurden hier ein Ausschnitt gemacht, um von der Seite ins Ablagefach greifen zu können, sowie ein Einschub, um ein Kartenbegrenzer einschieben zu können. Dieser ist dazu da die genaue Grösse der Karte noch von Hand anpassen zu können.

Beim Boden (grün) wurde noch ein Ausschnitt frei gelassen, um Platz für einen SIM-Karte Lesegerät zu machen.

Der orange und der gelbe Bereich dienen nur dazu den Schieber in einer sauberen Führungsbahn zu halten.

Es wurde eine Kerbe im gelben Bereich gemacht, so dass ein Blech Platz hat, welches nach vorne und zurück geschoben werden kann.

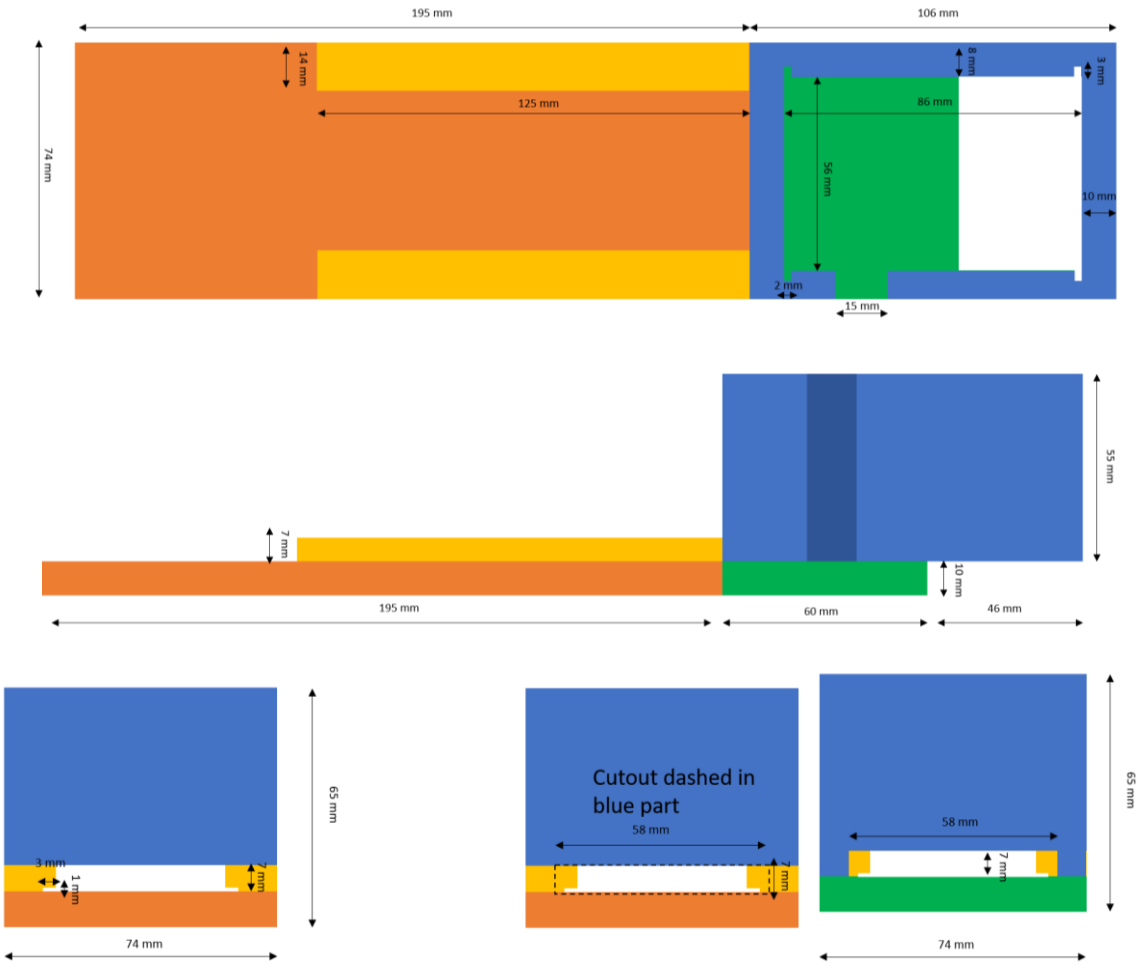



*Abbildung 12 2D Plan*

Nach Beendigung der Zeichnung wurde der Plan an einen Freelancer der 3D Modelle erstellt weitergeben. Dieser erstellte aus dem 2D Plan ein 3D Modell.

Das 3D Modell wurde im STL Format erstellt.

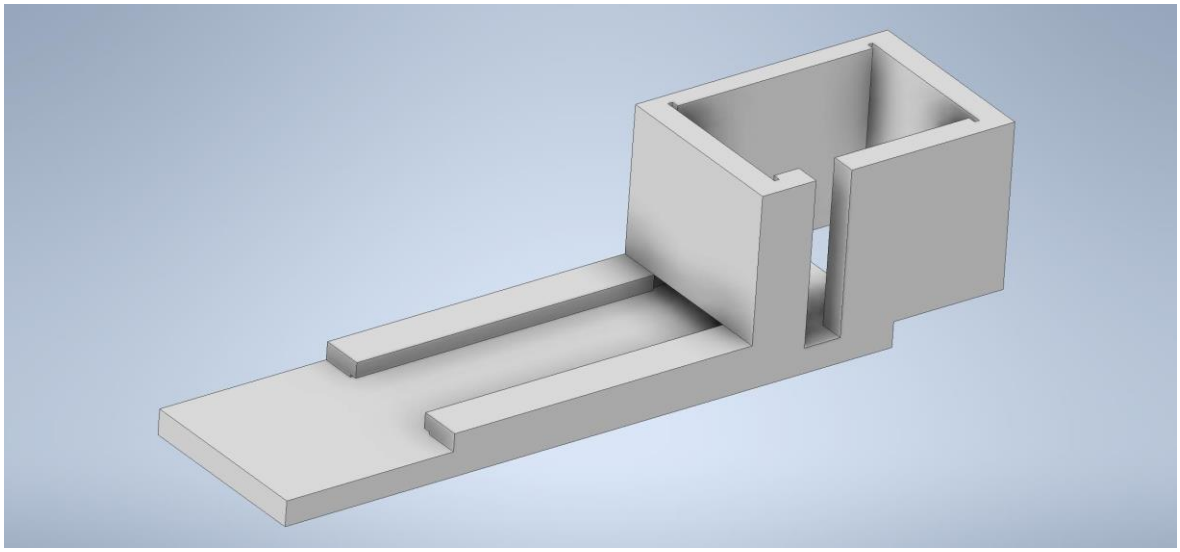


*Abbildung 20 STL Gehäuse*

Nun musste eine Druckerei gefunden werden, welche die 3D Datei in einem 3D Drucker drucken kann.

Hierbei war wichtig, dass die Druckerei in der unmittelbaren Nähe ist und dass man preislich im Budget bleiben kann.
Man hat sich für die Gaffuri Druckerei entschieden, da die Kosten moderat waren.

Man konnte über das Online Tool nach dem Upload der STL-Datei direkt die Bestellung auslösen.

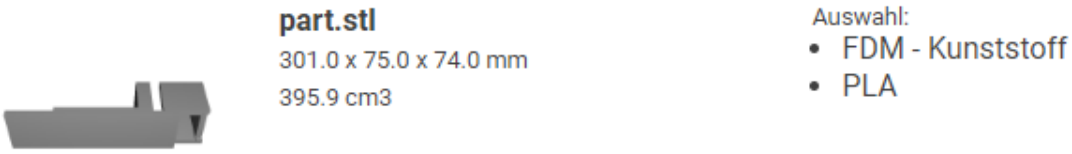



*Abbildung 21 Bestellung Gaffuri*

Das Modell wurde im Schmelzschichtungsverfahren erstellt aus Polylactide und innerhalb einer Woche geliefert.

## 7.3. [2.3] Aufbau Hardware

In den folgenden Abschnitten wird beschrieben wie die Hardware aufgebaut wird, bis man alles so vorbereitet hat, um nur noch Programmierarbeiten ausführen zu können.

### 7.3.1. Raspberry Pi mit Raspbian (Linux) Betriebssystem aufsetzen

Hierbei wurde eine schon bestehende Installationsanleitung von „elektronik-kompendium.de“. verwendet. Da der Ablauf fast eins zu eins übernommen werden konnte, mussten nur kleinere Anpassungen an der Anleitung gemacht werden.

**1.Image des Rasbian Betriebssystems herunterladen**

Zuerst muss das Image (IMG oder ISO) des Rasbian Betriebssystems heruntergeladen werden. Ein Image ist die Kopie eines Speicherabbilds eines Laufwerks. In dem Fall von einem installierten und konfigurierten Betriebssystem mit Anwendungen, die zum Zwecke der Vervielfältigung, in einer einzigen Datei gespeichert sind.

**2.Komprimierte Image-Datei entpacken**

In der Regel handelt es sich bei dem heruntergeladenen Image um eine komprimierte ZIP-Datei. Damit kann man noch nichts anfangen. Deshalb muss man nach dem Download die Datei entpacken. Allerdings kommen manche Betriebssysteme mit der Größe der Datei nicht klar, weshalb es bei der Inbetriebnahme des Raspberry Pi zu teilweise merkwürdigen Problemen kommt, deren Ursachen sich nicht eindeutig identifizieren lassen.

Um eine Image-Datei fehlerfrei zu entpacken nutzt man 7zip.

**3.SD-Speicherkarte beschreiben (flashen)**

Erst jetzt geht es darum, den Inhalt der Image-Datei (IMG oder ISO) auf einen passenden Datenträger zu schreiben. Manche bezeichnen den Vorgang als flashen.

Um eine SD-Karte lesen und beschreiben zu können, ist in jedem Fall ein Speicherkarten-Leser Voraussetzung. Entweder integriert in den PC, das Notebook oder man hat einen externen Speicherkarten-Leser für den USB-Anschluss.

Hinweis: Ein Image ist ein vollständiges Abbild, also eine 1:1-Kopie, eines oder mehrerer Partitionen mit einem Dateisystem. Durch das Aufspielen eines Images wird die SD-Karte bereits formatiert. Und zwar im Format des Images. Deshalb ist ein vorheriges Formatieren der Speicherkarte sinnlos und unnötig.

Man verwendet hier den „Win32DiskImager" für Windows.

Hierbei wählt man die vorher angeschlossene Micro SD-Karte unter Datenträger aus.
Nachher wählt man die entpackte Image-Datei über das kleine Ordnersymbol aus.
Nun kann man unten auf „Schreiben“ drücken und die SD-Karte wird beschrieben.

Das Image erzeugt auf der Speicherkarte zwei Partitionen. Eine im Windows-kompatiblen FAT32-Format, die andere im Linux-Format.

Hinweis: Wenn man nach dem Beschreiben der Speicherkarte die Partitionen unter Windows anschauen will, dann sieht man nur ein Laufwerk bzw. eine Partition, die recht klein ist (ca. 50 MByte). Das ist die Boot-Partition, auf der sich die Firmware für den Raspberry Pi und der Bootloader befinden. Die andere Partition im Linux-Format beherrscht Windows nicht. Windows kann nur diese Partition anzeigen.

Ist das Beschreiben der Speicherkarte gelungen, dann meldet man die Speicherkarte vom System ab und kann sie aus dem Kartenleser entfernen.

### 4. Erste Inbetriebnahme

SD-Karte in Raspberry Pi stecken und System mit Tastatur Maus und Bildschirm verbinden. Anschliessend Raspberry an Strom hängen somit startet das Setup auf.

### 5. Erste Schritte bei der Konfiguration (Grundkonfiguration)

Nach der ersten Inbetriebnahme des Raspberry Pi sollte man eine grundlegende Konfiguration vornehmen. Dazu gehört die Konfiguration von folgenden Einstellungen. Diese öffnet man im Terminal mit diesem Kommand „sudo raspi-config“.

- Sprache *German (Switzerland)*
- Zeitzone *Europa/Zürich*
- Netzwerk *Vebinden über WLAN mit Netzwerk „Jakal“*
- Tastatur-Layout *German (Swiss)*
- Gerätename *automaticsimcardupdater*

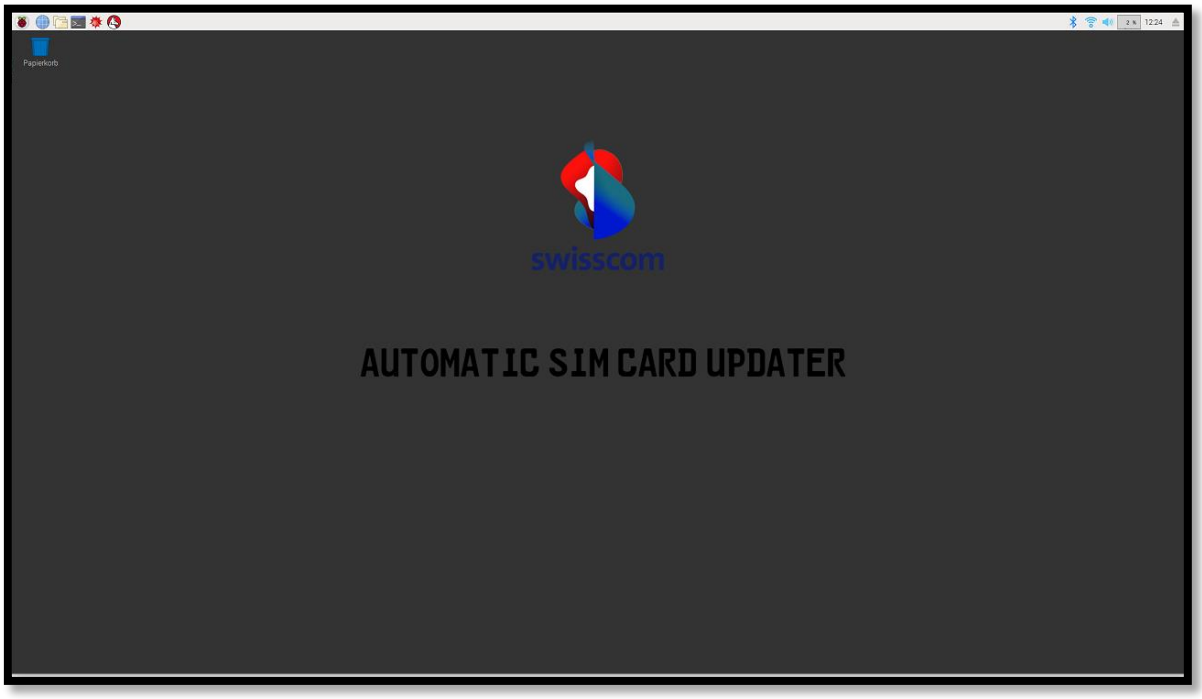



*Abbildung 22 Rasbian Desktop*

### 7.3.2. Alle elektronischen Bauteile mit Raspberry verbinden

Um die verschiedenen elektronischen Bauteile ansteuern zu können, müssen diese mit den GPIO Pins des Raspberry Pi verbunden werden.

Hierbei ist es nicht auschlaggebend mit welchem GPIO die Geräte verbunden werden.

Zusätzlich muss definiert werden, ob die Pins IN oder OUT GPIOs sind. IN wird bei einem Gerät verwendet, das dem Raspberry Pi einen Zustand mitteilen will, wie zum Beispiel ein Schalter. Out wird verwendet, um zum Beispiel einer Led-Strom zu geben, so dass sie leuchtet.

**LED** – Die 3 LED‘s werden mit den GPIO 23/24/25 verbunden. Zusätzlich müssen sie mit dem Ground verbunden werden. Zwischen Ground und den LED’s muss ein Wiederstand (330Ohm) gesetzt werden. Dieser wird verwendet, um die Strombegrenzung zu steuern. Der Wiederstand dient dazu überschüssiger Strom im Widerstand in Wärme umzuwandeln um die LED’s nicht zu überhitzen.

**Taster** – Die 2 Taster werden mit GPIO 19 und 27 verbunden. Zusätzlich werden sie an die 3.3V Stromquelle angeschlossen, da wir diese als IN GPIO’s definieren.

**Lichtsensor** – Er wird mit dem GPIO 21, 3.3V Stromquelle und Ground verbunden, da er als Input wie Output dient. Auf dem Mikroboard sind LED’s, welche anzeigen ob der Lichtsensor geschalten ist oder nicht. Daher brauchen wir den Ground.

**Relais** – Es wird ein Relais verbaut, das dazu dient, dem SIM-Karten Lesegerät vorzugeben, wann es anfangen soll die SIM-Karten zu lesen. Diese wird mit dem GPIO 11, Ground und 5V Stromquelle verbunden. Auf der SIM-Karte Lesegerät gibt es ein Taster, der erkennt wen eine Karte ins SIM-Karte Lesegerät geschoben wird. Dieser muss man entfernen da er den Auswurf der SIM-Karte stört. Er wurde dann an Drähte gelötet, die an die Anschlüsse des Relais angeschlossen werden, so kann man über das erzeugte Programm das genaue Timing vorgeben wann der Lesevorgang beginnt. Genauer beschrieben in Verzeichnispunkt 7.10.2.

**Servomotor** – Dieser wird an den GPIO 17 angeschlossen. Hier ist es wichtig den Motor an eine separate Stromquelle anzuschliessen und diesen nicht über den Raspberry Pi laufen zu lassen. Dies hat der Grund, weil der Motor zu viel Strom verbraucht ca. (1A bei 5V). Der 5V Output des Raspberry Pi ist aber nur für 0.7 A ausgelegt. Hier ist es wichtig den Ground der externen Stromquelle mit dem Ground auf dem Raspberry Pi kurzzuschliessen. Dies ist soweit nötig, dass die PWM Impulse, welche über den Raspberry ausgegeben werden, auch über den Raspberry Ground abfliessen können, sonst würde der Servomotor die Impulse nicht sauber erkennen können.

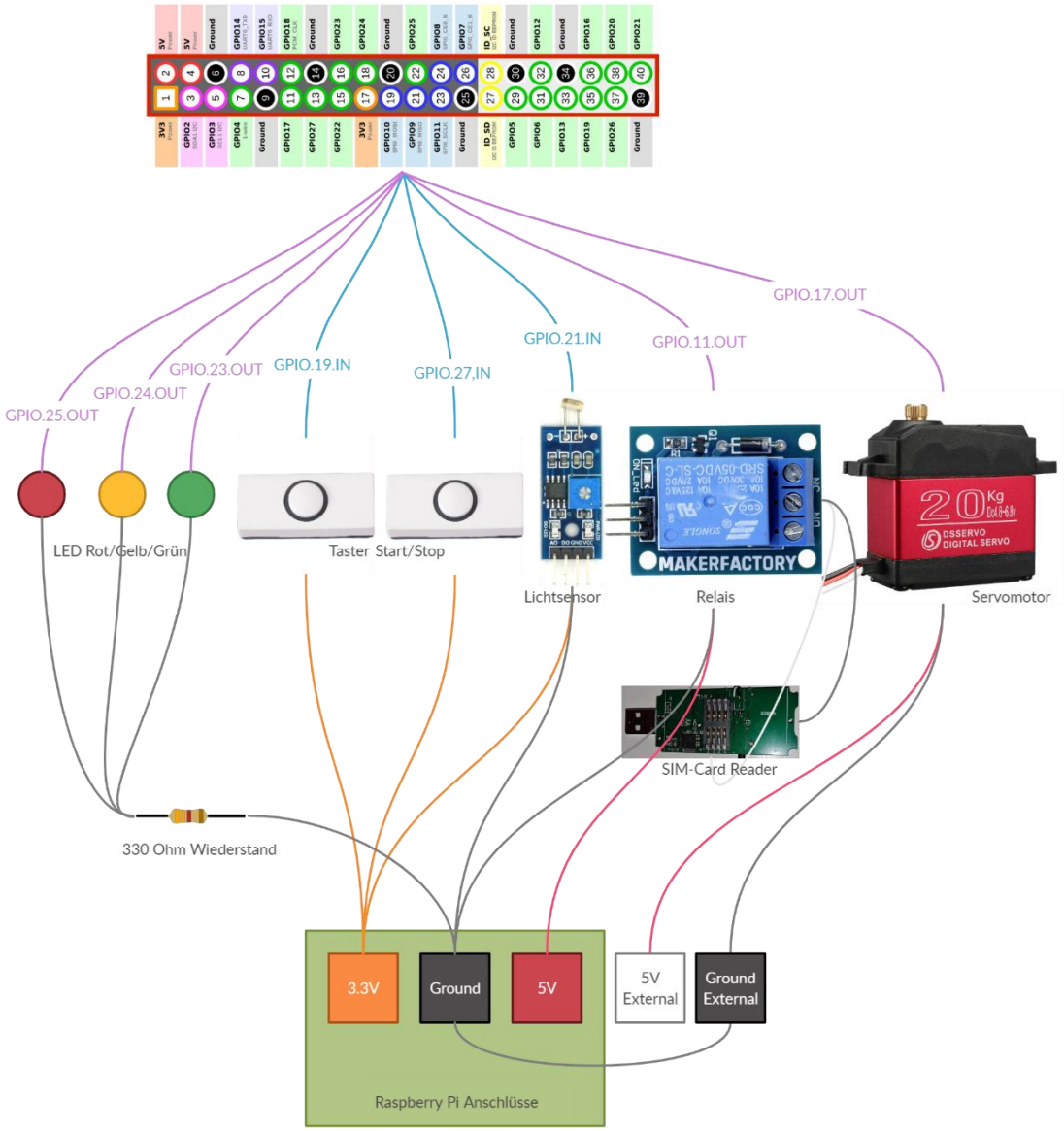



*Abbildung 23 Anschlüsse GPIO*

### 7.3.3. Zusammenbau Hardware/Mechanik

Im folgenden Abschnitt wird erklärt, welche Arbeitsschritte ausgeführt werden müssen, um den kompletten ASU zusammenzubauen.

Das 3D Gehäuse wurde mit der gekauften Holzkiste an 3 Punkten mit M-Schrauben verbunden.

Am 3D Modell musste noch einige Modifikationen durchgeführt werden:

- Loch für Lichtsensor bohren
- Kerbe für Schraube in Einschub
- SIM-Kartenfach, Länge angepasst ca. 1 mm abgefräst
- Ausschnitt für Servomotor gemacht
- Loch, um SIM-Karten Begrenzer bewegen zu können

An der Holzkiste wurden folgende Arbeiten ausgeführt:

- Loch Für Lichtsensor
- 2x Loch für Taster Start/Stopp
- 3x Loch für LED-Anzeige
- Ausschnitt für Motor
- Ausschnitt für Kabelausführung
- Loch für Kabel an SIM-Karte Lesegerät

Metallteile zugeschnitten:

- Einschub zugeschnitten und Loch gebohrt. 135x45 mm
- SIM-Karten Begrenzer zugeschnitten und Loch gebohrt. 65x60 mm
- Kolbenmechanismus Stück eins zugeschnitten 130 mm und Löcher gebohrt. Loch bei 20 mm, 40 mm, 60 mm und 115 mm
- Kolbenmechanismus Stück zwei zugeschnitten 110 mm und Löcher gebohrt. Loch bei 5 mm und 100 mm
- Abhebung für Servomotor zugeschnitten 50 mm und Löcher gebohrt bei 20 mm

Nun wurden alle fehlenden Teile am Grundgehäuse befestigt:

- SIM-Kartenlesegerät mit 3D Gehäuse verbunden
- 2x Taster angeschraubt
- Raspberry Pi in Kiste angeschraubt
- Relais in Kiste angeschraubt
- Kabel, die aus dem Gehäuse führen, wurden in der Kiste befestigt, so dass sie nicht ausreisen können
- Servomotor an 3D Gehäuse und Holzkiste verschraubt
- Schieber mit dazugehörigem Mechanismus wurde verbunden und an den Servomotor verbunden

## 7.4. [2.4] Einbindung beim Kunden

Beim Kunden muss das ASU eingebunden werden. Leider kann aus mangelnder SNOW PoS Software nur eine teilweise Einbindung gemacht werden.

Beim Kunden muss das System mit einem nicht über das Swisscom Netz laufenden Netzwerk verbunden werden. Hierbei gibt es eine private Festnetzleitung, die beim Kunden am Arbeitsplatz vorhanden ist. Dieses Netzwerk wird für Download von Dateien, die über FTP laufen verwendet, da die Swisscom kein FTP Downloads in ihrem Netz zulässt.

Dieses Netz ist aufgebaut wie ein normaler privater Router Zuhause.

Der Raspberry Pi kann ganz einfach über das User Interface in den Einstellungen von Rasbian zu dem Wi-Fi verbunden werden.

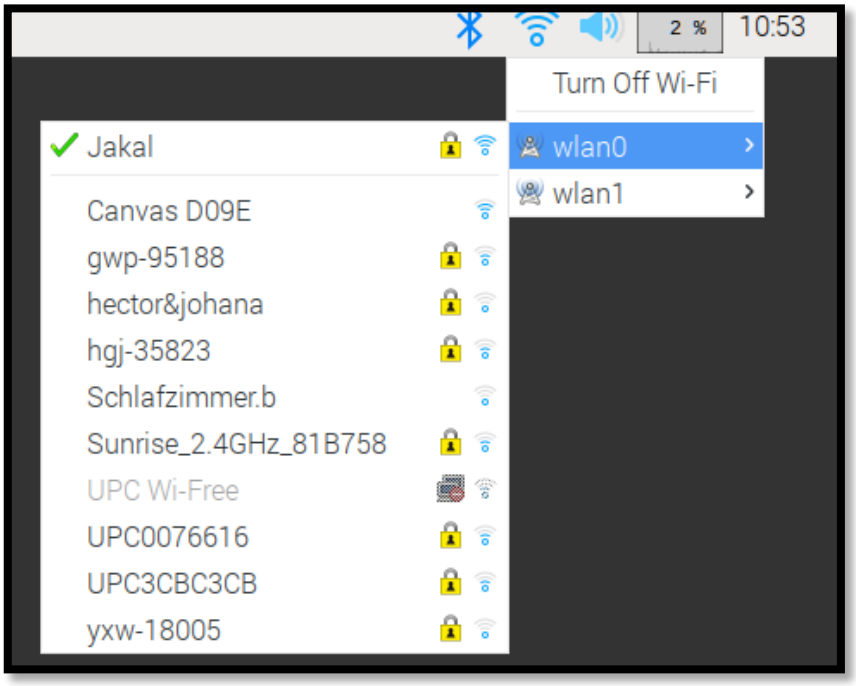



*Abbildung 24 Raspberry Pi Wi-F*

Nach Verbindung mit dem neuen Netzwerk muss die Empfänger-E-Mail-Adresse in Programm noch angepasst werden. Hierbei wurde sie an die Team-Mailbox von Mobile Devices verwiesen:

Terminal-Support.Mobile-Net@swisscom.com

Zusätzlich wurde der Kunde eingeführt in folgenden Themen:

- Aufbau Gerät
- Funktion
- Mögliche Fehler
- Behebung von möglichen Fehlern

All diese Themen wurden in einer Bedienungsanleitung festgehalten. Diese findet man im Anhang.

Um vor Datenverlust geschützt zu sein, wurde ein Backup-Image des ganzen Betriebssystems gemacht. Heisst falls hierbei der Datenträger korrupt wird, kann das ganze System ohne grösseren Aufwand wieder in Betrieb genommen werden.

1. SD-Karte aus Raspberry Pi entfernen in SD-Kartenleser an PC anschliessen.
2. Roadkil's Disk Image herunterladen und installieren. Dieses Programm ist wichtig, da man Abbilder von kompletten Datenträgern erstellen kann und nicht nur von Partitionen.
3. Nach dem Aufstarten geht man unter den Reiter Store Image, wählt dort unter Read Image die Physische SD-Karte aus. Unter dem Target File wählt man den Pfad aus, wo man das Abbild hin kopieren will und den Namen der Datei. In unserem Fall heisst die Datei ASU_V1.img. Nun kann der Backup Vorgang beginnen, indem man auf Start drückt.
4. Nach dem Abschluss sollte ein Fenster aufpoppen, welches einem bestätigt, dass der Schreibvorgang beendet wurde.

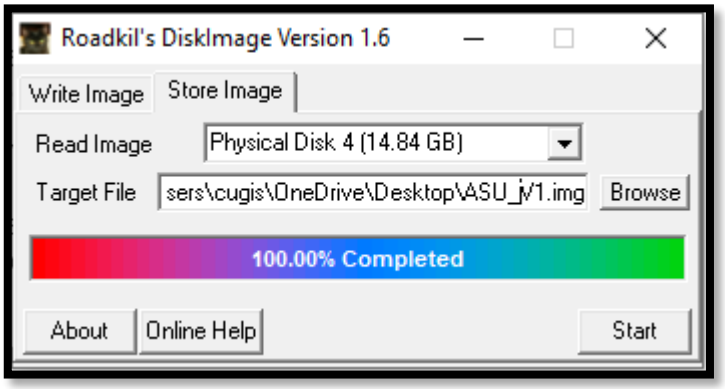



*Abbildung 25 Roadkil's Disk Image*

Das erstellte Backup wurde nun bei Swisscom im Sharepoint von System Team Mobile Devices unter folgendem Pfad abgelegt.

Dokumente\06_Projects\21_Automatisation\ASU.

## 7.5. Evaluierung Programmiersprache

In diesem Teil sollen die Varianten aufgezeigt werden, mit welcher Programmiersprache diese Diplomarbeit umgesetzt wird.

Folgende Sprachen wurden für die Evaluierung ausgewählt. Die Beschreibungen zu den Programmiersprachen wurden von „Itwissen.info“ entnommen.

**Python** - Die Programmiersprache Python wurde Anfang der 90er Jahre von Guido van Rossum in Amsterdam veröffentlicht. Die Sprache wurde im Hinblick auf die englische Komiker Truppe Monty Python benannt und ist eine dynamisch typisierte objektorientierte Programmiersprache. Python hat den Ruf, eine einfache Programmiersprache zu sein, d.h. sie ist leichter zu erlernen und zu lesen.

**C++** - Die Programmiersprache C++ ist die Weiterentwicklung der Programmiersprache C für die objektorientierte Programmierung. Es ist eine eigenständige Programmiersprache, die in den 80er-Jahren von AT&T-Mitarbeitern entwickelt und speziell in technisch-wissenschaftlichen Anwendungen eingesetzt wurde. Zwischenzeitlich kommt die von der ANSI genormte Programmiersprache C++ auch in kommerziellen Anwendungen zum Einsatz.

**Java** - Die Programmiersprache Java ist eine objektorientierte und plattformunabhängige Programmiersprache, deren Grundlagen in der ersten Hälfte der 1990er Jahre von Sun Microsystems entwickelt worden sind. Die Syntax von Java lehnt sich stark an C++ an, obwohl die Sprache selbst sich in vielen Punkten unterscheidet.

Es werden Kriterien beschrieben die in Muss/Soll/Kann unterteilt werden, diese werden verschieden gewichtet, um aufgrund des Ergebnisses eine Auswahl zu treffen.

| Beschreibung | Gewichtung | Punkte | Python | C++ | Java |
|---|---|---|---|---|---|
| Objektorientiert | Muss | 3 | Ja | Ja | Ja |
| GPIO Library vorhanden | Muss | 3 | Ja | Ja | Nein |
| In Konsole ausführbar | Soll | 2 | Ja | Nein | Nein |
| Pakete vorhanden | Kann | 1 | Ja | Ja | Ja |
| Benutzerfreundlichkeit leicht | Kann | 1 | Ja | Nein | Ja |
| **Erreichte Punktzahl** | | **Max. 10** | **10** | **7** | **5** |

*Tabelle 10 Gewichtung Programmiersprache*

Nach der Gewichtung hat man sich für Python entschieden, da es alle Kriterien abdeckt, die vorgegeben sind.

Da hier die Muss-Ziele besonders wichtig waren, hat man einerseits das objektorientierte Programmieren zur Verfügung, anderseits ist die Library für GPIO schon vorhanden.

C++ ist vor allem problematisch, da es für Anfänger eine eher schwer zu verstehende Sprache ist. Zusätzlich ist das Ausführen in der Konsole umständlich.

Java ist zwar ähnlich von der Benutzerfreundlichkeit aufgebaut wie Python, doch fehlt hier die Library für GPIO auf dem Raspberry Pi OS. Diese müsste noch hinzugefügt werden. Auch hier ist das direkte Ausführen in der Konsole schwierig.

## 7.6. [3.1] Automatischer Auswurf/Einschub programmieren

Um den Auswurf zu programmieren, muss der Servomotor angesteuert werden. Da es ein PWM Servomotor ist, kann man dieser per Impulse einfach auf Gradeinstellungen steuern.

Bei LED‘s kann man den Zustand ganz einfach steuern, indem man den PIN ein- und ausschaltet. Dies ist beim Servomotor nicht so einfach möglich, da man hier viel mehr Zustände braucht. Hierbei kommt die Pulsweitenmodulation ins Spiel. Hierbei kann man über einen Puls den Motor auf einen bestimmten Winkel setzen.

***RPi.GPIO***

Der Raspberry beinhaltet eine Library in Python die RPi.GPIO heisst, diese beinhaltet die Möglichkeit einen Puls mit 3.3 Volt über einen Pin auszugeben. Durch das schnelle Aus- und Einschalten wird ein konstanter Strom erzeugt. Hierbei wird durch längere Pulse auch die Ausgangsspannung höher. Der Servemotor erkennt dies und setzt sie auf einen bestimmten Grad je nachdem, wie hoch die Ausgangsspannung ist. Wichtig ist auch zu definieren, welche Frequenz der Motor erkennt, in diesem Fall ist dies 50 Hz.

Die Grad werden im Programm in vordefinierte Werte gesetzt. Diese sehen wie folgt aus:

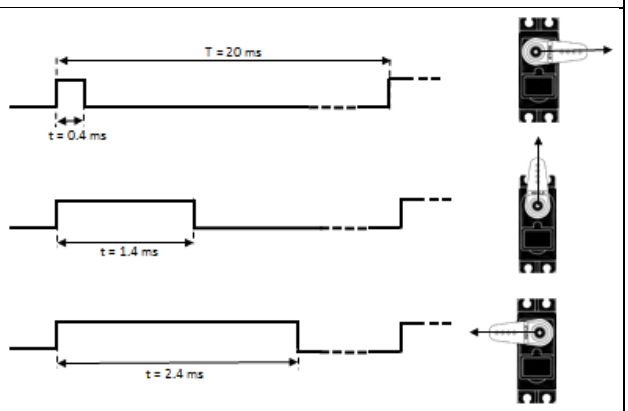
| **Zeit** | **Grad** | **PWM Ansteuerung (DutyCycle)** | |
|---|---|---|---|
| 0.4 ms | 0° | 0.4/20=2% (2.5) |  |
| 1.4 ms | 90° | 1.4/20=7% (7.5) | |
| 2.4 ms | 180° | 2.4/20=12% (12.5) | |

*Abbildung 26 PWM*

Um das Ziel umsetzen zu können, muss man verschiedene Methoden dazu schreiben, um verschiedene Aufgaben abzudecken. Hierbei erstellt man eine Klasse namens Motor und fügt dann verschiedene statische Methoden hinzu. Heisst bei einer so definierten Methode wird weder der Name der Klasse noch der Name der Instanz angegeben, von der aus die Methoden aufgerufen werden.

In diesem Fall wurden drei Methoden erstellt.

**run -** Die Run Methode wird dazu genutzt den Schieber des Einschubs in das Ablagefach zu schieben und wieder herauszuziehen. Hierbei Muss man die oben beschriebene Ansteuerung über PWM benutzen. Man startet den Prozess in der Nullstellung. Diese ist beim verwendeten Servomotor umgekehrt, heisst diese ist beim Motor 180 Grad. Danach muss eine Pause von 2.5 Sekunden erfolgen, um dem Motor Zeit zu geben in die Richtige Position zu gelangen.

Um Herauszufinden, welche Position (Gradstellung) die richtige ist, um den Einschub genau im Ablagefach zu haben, schreibt man ein kleines Programm. Dieses startet bei der Nullstellung und wartet fünf Sekunden und dreht Anschliessend um 9 Grad weiter dies wird wiederholt bis man in der richtigen Position ist.

Hierbei kann man, wenn die Position richtig ist, das Programm stoppen. Die richtige Position war bei diesem Fall 99 Grad.

**Positions_check -** Der Positionscheck ist dazu da, um die Position des Schiebers zu erkennen. Es besteht immer die Möglichkeit, dass der Motor bei einem schon gemachten Testlauf im SIM-Karten-Schacht stecken bleibt oder nicht sauber auf die Start-Position zurückgeht.

Mit diesem Skript kann man, bevor man die eigentliche Automatisierung startet den Motor auf die Ausgangsposition stellen.

**stop -** Hierbei wird der Motor gestoppt. Heisst die PWM Signale werden auf Null gestellt und anschliessend werden die GPIO Pins auf Null gestellt.

*Code https://github.com/jakal1989/ASU/blob/main/motor.py*

## 7.7. [3.2] Motor Ausschalten, wenn keine SIM-Karte vorhanden

Um dieses Ziel umzusetzen, muss man hier ein Grundprogramm schreiben. Zusätzlich wird noch eine Klasse Namens Led erstellt. Die Klasse Led wird dazu verwendet eine visuelle Anzeige bezüglich der verschiedenen Zustände anzuzeigen.

Sobald keine SIM-Karte im ASU vorhanden ist, wird das Skript für den Motorstopp ausgeführt.

Als erstes erstellt man eine if/else Funktion bei der, der Status des Lichtsensors ausgelesen wird.
Wenn der Lichtsensor nach dem Start des Programms hell erkennt, wird das Programm gestoppt, ohne das der Motor gelaufen ist.

Falls der Lichtsensor dunkel erkennt, wird eine while-Schlaufe gestartet. Hierbei wird nochmals eine if/else-Funktion geschrieben. Diese kontrolliert, ob der Lichtsensor dunkel erkennt; wenn ja wird der Zyklus für einen Auswurf ausgeführt. Dies geschieht solange bis der Lichtsensor hell erkennt. Wenn der Lichtsensor hell erkennt, wird der Motor ausgeschaltet und die GPIOs auf Null gestellt.

**if** – Diese Kondition wird verwendet, wenn der Lichtsensor dunkel erkennt.

**else** – Hier wurde keine Kondition festgelegt; falls Licht vorhanden ist, wird diese Kondition ausgeführt.

*Code https://github.com/jakal1989/ASU/blob/main/main.py*

Es wurde eine neue Klasse Namens Led erstellt. Diese wird für die Visualisierung des Status des Systems verwendet.

***RPi.GPIO***
Die Bibliothek beinhaltet die Möglichkeit, die Pins auf dem Raspberry einzuschalten und auszuschalten. Hierbei wurde dies verwendet, um die einzelnen LED‘s ein- und auszuschalten.

Folgende Methoden wurden erstellt:

**red_on** – Rotes Licht einschalten

**red_off** – Rotes Licht ausschalten

**yellow_on** – Gelbes Licht einschalten

**yellow_off** – Gelbes Licht ausschalten

**green_on** – Grünes Licht einschalten

**green_off** – Grünes Licht ausschalten

Die benutzen LED‘s stellen folgenden Status visuell dar:

| Led-Farbe | Status |
|---|---|
| Grün | SIM-Karte im Ablagefach |
| Gelb | Benachrichtigung versenden |
| Rot | Keine SIM-Karte im Ablagefach |

*Tabelle 11 Led-Status*

*Code https://github.com/jakal1989/ASU/blob/main/led.py*

## 7.8. [3.3] SMS oder E-Mail an Anwender, wenn Stapel abgearbeitet ist

Der Nutzer soll über Zustände benachrichtigt werden. Dieses wird wie folgt umgesetzt:

Man erstellt über Google eine neue E-Mail-Adresse. Hierbei hat man sich für eine Gmail Adresse entschieden, da sie gratis ist und man hier über externe SMTP Clients bis zu 100 E-Mails pro Tag versenden kann. Hierbei muss eine wichtige Einstellung in Gmail eingeschaltet werden. Diese heisst „Zugriff durch weniger sichere Apps“. Diese ermöglicht es einem über SMTP E-Mails von unsicheren Clients zu verschicken.- Die sSMTP Bibliothek, welche man verwenden wird, wird als solcher eingestuft.

Man nutzt nun die folgenden Bibliotheken, um hier eine neue Klasse zu erstellen.

***sSMTP***
Man kann sich mit diesem Programm auf den gewünschten E-Mail Server einloggen. Anschliessend wird es dazu verwendet, die E-Mails über SMTP zu senden.

***email***
Bei email verwendet man nur ein Plugin das heisst encoder. Base 64. Dieser codiert die Nutzdaten in base64-Form und setzt den Content-Transfer-Encoding-Header auf base64. Dies ist eine Codierung, die verwendet werden kann, wenn der größte Teil der Nutzdaten nicht druckbare Daten sind, da diese kompakter sind als die druckbaren. Der Nachteil der Base64-Codierung besteht darin, dass der Text nicht für Menschen lesbar ist.

***email.mime.multipart***
Der Hauptzweck dieser Klasse besteht darin, die Verwendung der Methode attach () zu verhindern, die nur für mehrteilige Nachrichten sinnvoll ist. Wenn attach () aufgerufen wird, wird eine MultipartConversionError-Ausnahme ausgelöst.

***email.mime.text***
Diese Klasse wird verwendet, um MIME-Objekte vom Haupttyp Text zu erstellen.

***email.mime.base***
Die Klasse wird hauptsächlich als praktische Basisklasse für spezifischere MIME-fähige Unterklassen bereitgestellt.

Hierzu wurde eine Klasse namens Mail und 3 Methoden dazu gebaut.

**without_sim** – Hierbei wir eine E-Mail ausgelöst, welche dem Anwender die Information zurückgibt, dass er keine SIM-Karte ins Ablagefach eingelegt hat.

**with_sim** – Bei dieser Methode wird eine E-Mail ausgelöst, welche die Update_liste.txt Datei beinhaltet und zusätzlich dem Anwender sagt, dass der Stapel abgearbeitet ist.

**hand_stop** - Bei dieser Methode wird eine E-Mail ausgelöst, welche die Update_liste.txt Datei beinhaltet und dem Anwender sagt, dass hier das Programm von Hand angehalten wurde.

*Code https://github.com/jakal1989/ASU/blob/main/mail.py*

Um immer die neuste Updateliste an den Anwender zu schicken, muss hier noch eine Methode geschrieben werden. Diese beinhaltet die Updateliste zu löschen nach dem sie per E-Mail an den Anwender geschickt wurde.

Es wurde die Klasse File erstellt. Die Klasse File benutzt die Bilbiothek os.

***os***
Das Betriebssystemmodul in Python bietet Funktionen für die Interaktion mit dem Betriebssystem. Das Modul enthält viele Funktionen für die Interaktion mit dem Dateisystem.

Folgende Methode wurde dazu geschrieben:

**delete** – Falls die Datei Update_liste.txt vorhanden ist, wird die Datei gelöscht. Wenn diese nicht vorhanden ist, gibt es eine Nachricht in der Konsole, welche sagt, dass die Datei nicht vorhanden ist.

*Code https://github.com/jakal1989/ASU/blob/main/file.py*

## 7.9. [3.4] Per Knopfdruck System starten/stoppen

Hierbei muss einen Weg gefunden werden das Programm, welches geschrieben wurde per Knopfdruck starten und stoppen zu können.

### 7.9.1. Start Knopf

Hierbei wurde im Internet nach verschiedenen Varianten gesucht, um dies umzusetzen. Dabei wurde ein Beitrag von der «Heimtli Software Ag» gefunden. Dieser beschreibt, wie man über ein Bashskript ein Pin programmieren kann, so dass er bei Betätigung des Tasters ein Command im Terminal ausführt.

Heisst man kann hier das schon geschriebene Programm einbinden. Das Bashskript wird in die Autostart Datei des Raspberry Pi eingebunden und wird daher immer bei einem Neustart wieder gestartet.

Das Skript ist so aufgebaut, dass der Zustand eines GPIO Pins ausgelesen wird, falls der Zustand ändert, erkennt dies das Skript und führt den definierten Command im Terminal aus.

Das erstellte Skript wird im selben Ordner wie das Programm für ASU abgelegt.

/home/pi/ASU/

Das Starttasterskript greift auf Ordner und verschiedenen Dateien zu. Für dieses muss eine spezielle Berechtigung gesetzt werden.

Die Datei braucht die Berechtigung auf **chmod 755**.

Diesen setzt den Eigentümer auf (Lesen Schreiben Ausführen), die Gruppe auf (Lesen Ausführen) und sonstige auf (Lesen Ausführen).

Die Berechtigung ist in Unix Systemen wie folgt aufgebaut:

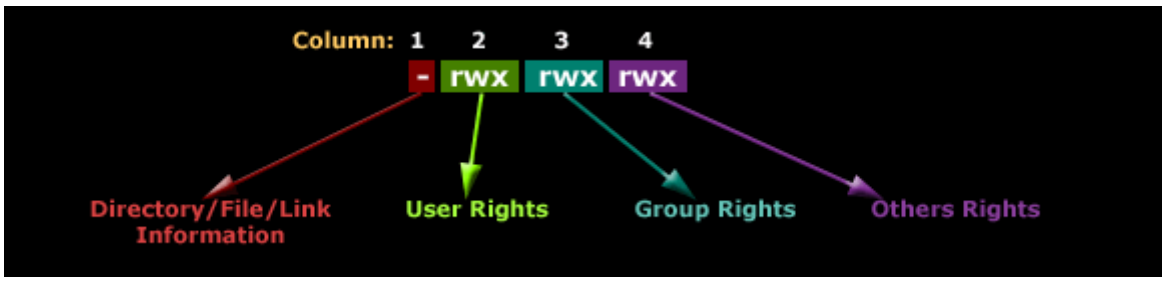



*Abbildung 27 Unix Rechte*

Die Berechtigungen werden anschliessend in Zahlenform geschrieben.

Wie diese berechnet werden, sieht man in der folgenden Tabelle:

| r, w, x Berechtigung | Berechnete Zahl |
|---|---|
| --- | 0 |
| --x | 1 |
| -w- | 2 |
| -wx | 3 (2+1) |
| r-- | 4 |
| r-x | 5 (4+1) |
| rw- | 6 (4+2) |
| rwx | 7 (4+2+1) |

*Tabelle 12 Unix Rechte in Zahlen*

| | |
|---|---|
| r | read |
| w | write (and delete) |
| x | execute (and access directory) |

Es wurde versucht, dies in Python umzusetzen. Das Programm zum Auslesen des Start Tasters konnte geschrieben werden und funktionierte, wenn man es im Terminal ausführte. Sobald man dies wie oben beschrieben über die Autostart Datei machen wollte, funktionierte dies nicht mehr. Aus diesem Grund wurde entschieden, dass schon bestehende Programm zu nutzen.

*Code https://github.com/jakal1989/ASU/blob/main/trigger*

### 7.9.2. Stopp Knopf

Um den Stopp-Knopf zu programmieren, wurde die zweite if/else-Funktion ergänzt mit einer if/elif/else Funktion, heisst man kann nun 3 Konditionen festlegen.

Konditionen:

**if** – Diese Kondition wird verwendet, wenn der Lichtsensor dunkel erkennt und der Stopp-Knopf nicht gedrückt ist.

**elif** – Diese Kondition wird verwendet, wenn der Lichtsensor dunkel erkennt und der Stopp-Knopf gedrückt ist.

**else** – Hier wurde keine Kondition festgelegt, heisst sie wird ausgeführt, falls Licht vorhanden ist und der Stopp-Knopf nicht gedrückt ist.

Das Programm funktionierte am Anfang nicht, da man hier erst später herausgefunden hatte, dass bei den Konditionen alle Werte angegeben werden müssen.

Es reicht nicht nur ein Wert wie «Licht ein» oder «Stopp-Taster aus», um eine Kondition zu starten. Es müssen beide Werte mit einer and-Funktion vorhanden sein.

*Code https://github.com/jakal1989/ASU/blob/main/main.py*

## 7.10. [3.5] Automatisch SIM-Karte über PC updaten (Skript auf PC)

Um die SIM-Karte auf dem PC upzudaten muss ein Programm vorhanden sein, welches die nötigen Lizenzen hat, um auf die SIM-Karten zu schreiben.

Die Swisscom nutzt nun schon seit einiger Zeit die SNOW Plattform, welche dazu verwendet wird SIM-Karten upzudaten. Hierbei gibt es verschiedene Varianten, um die SIM-Karten upzudaten.

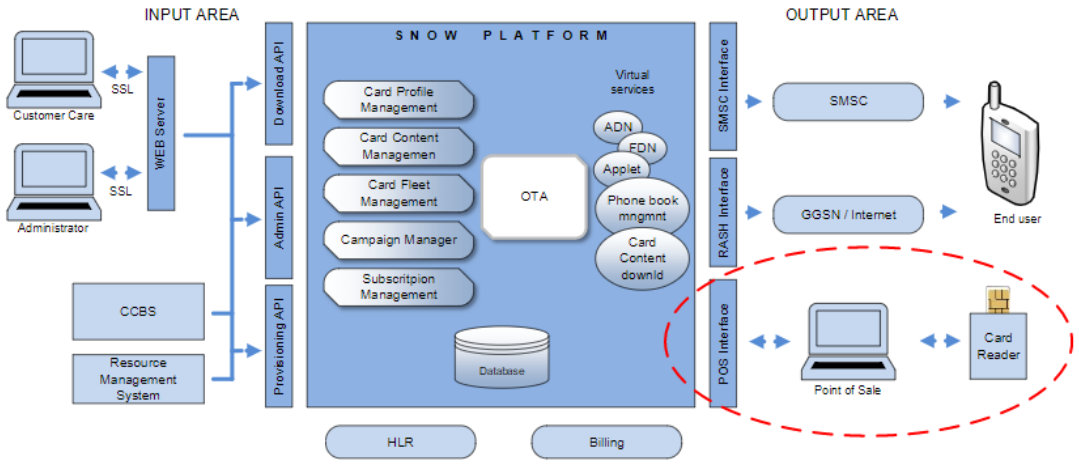



*Abbildung 28 SNOW Architektur*

SMSC - Hierbei wird ein SMS mit den nötigen Updates an das Gerät verschickt bei der die SIM-Karte upgedatet werden muss.

RASH - Mit der Rash Schnittstelle kann über das Internet ein Update an das Gerät geschickt werden.

PoS - Hierbei wird die SIM-Karte direkt über ein Kartenlesegerät mit einem PC verbunden. Auf dem PC läuft ein Web Client, welcher mit der SNOW Plattform verbunden ist.

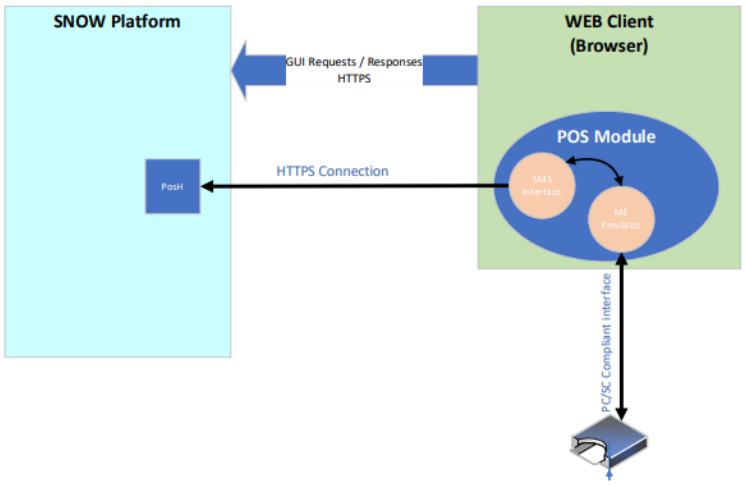



*Abbildung 29 SNOW Webclient*

Das neue Netzwerk sieht nun folgendermassen aus:

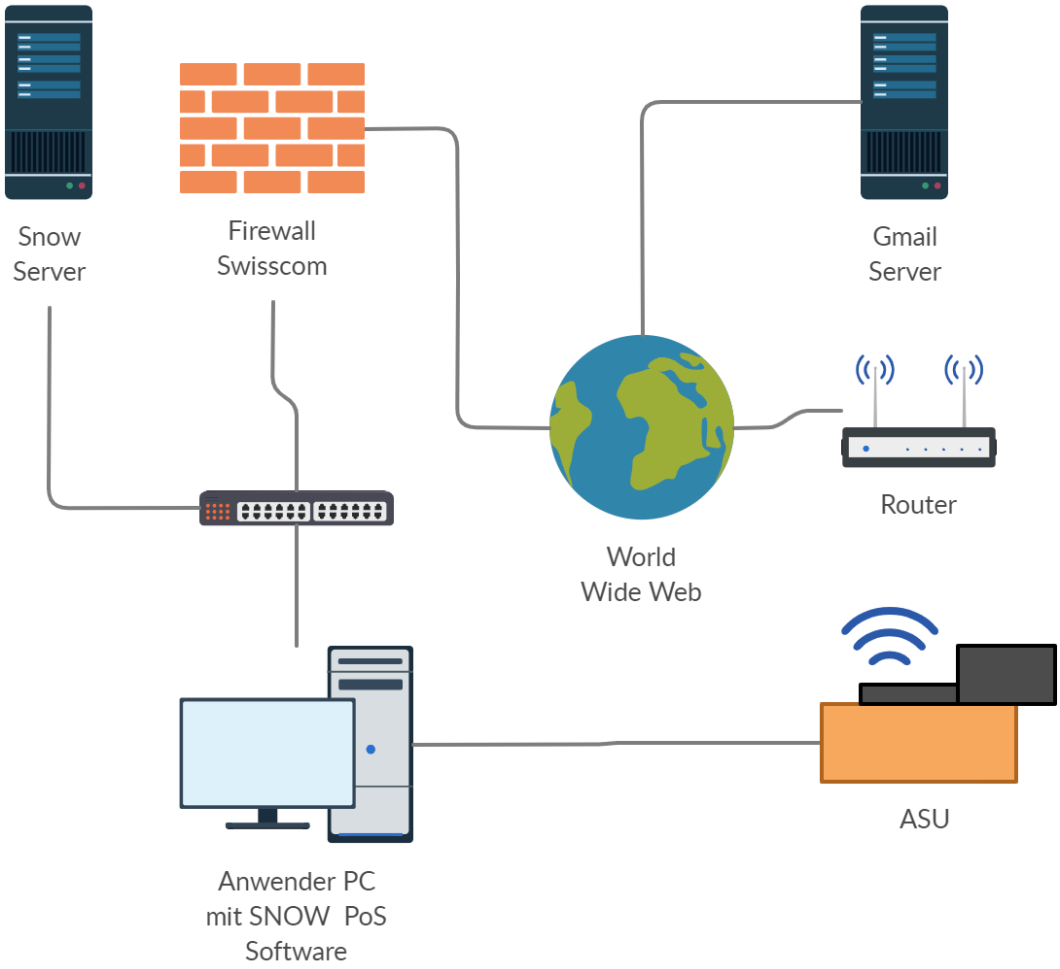



*Abbildung 30 Netzwerk ASU mit SNOW PoS*

Für das Projekt ist hier die PoS Schnittstelle wichtig. Diese wurde vor der Diplomarbeit im Frühjahr beim Hersteller bestellt, um sie dann bei der Diplomarbeit verwenden zu können.

Leider war PoS noch in Entwicklung und durch mehrere Verzögerungen wie die Covid-19 Pandemie konnte der Hersteller bis heute noch keine Schnittstelle liefern.

Somit kann das beschriebene Ziel nicht wie in der Zielsetzung umgesetzt werden.

Hierbei wurde die Entscheidung getroffen, dass an Stelle des Updates der SIM-Karte über PoS ein Testserver gebaut wird, welcher Rest-Post entgegennimmt und anschliessend eine Antwort an das ASU zurückschickt.

Dies wurde so entschieden, da man hier falls die PoS Schnittstelle zur Verfügung stehen wird nur kleine Anpassungen am Programm machen muss, um dieses Ziel korrekt umzusetzen.

Man wird die Funktion für das Auslesen der SIM-Karte trotzdem bauen, auch wenn im Moment das SIM-Karte Lesegerät keine Funktion hat.

### 7.10.1. MockServer mit Rest.Post aufsetzen

Um das neudefinierte Ziel umzusetzen braucht man einen Rest Server.
Nach kurzer Recherche wurde das Postman Tool entdeckt. Dies ist eine Gratis Software, mit welcher Rest Requests empfangen und Antworten verschickt werden können. Zusätzlich kann man mit diesem Tool API's sowie Mock Server erstellen.

Der Aufbau des Netzwerks ohne SNOW PoS sieht nur folgendermassen aus:

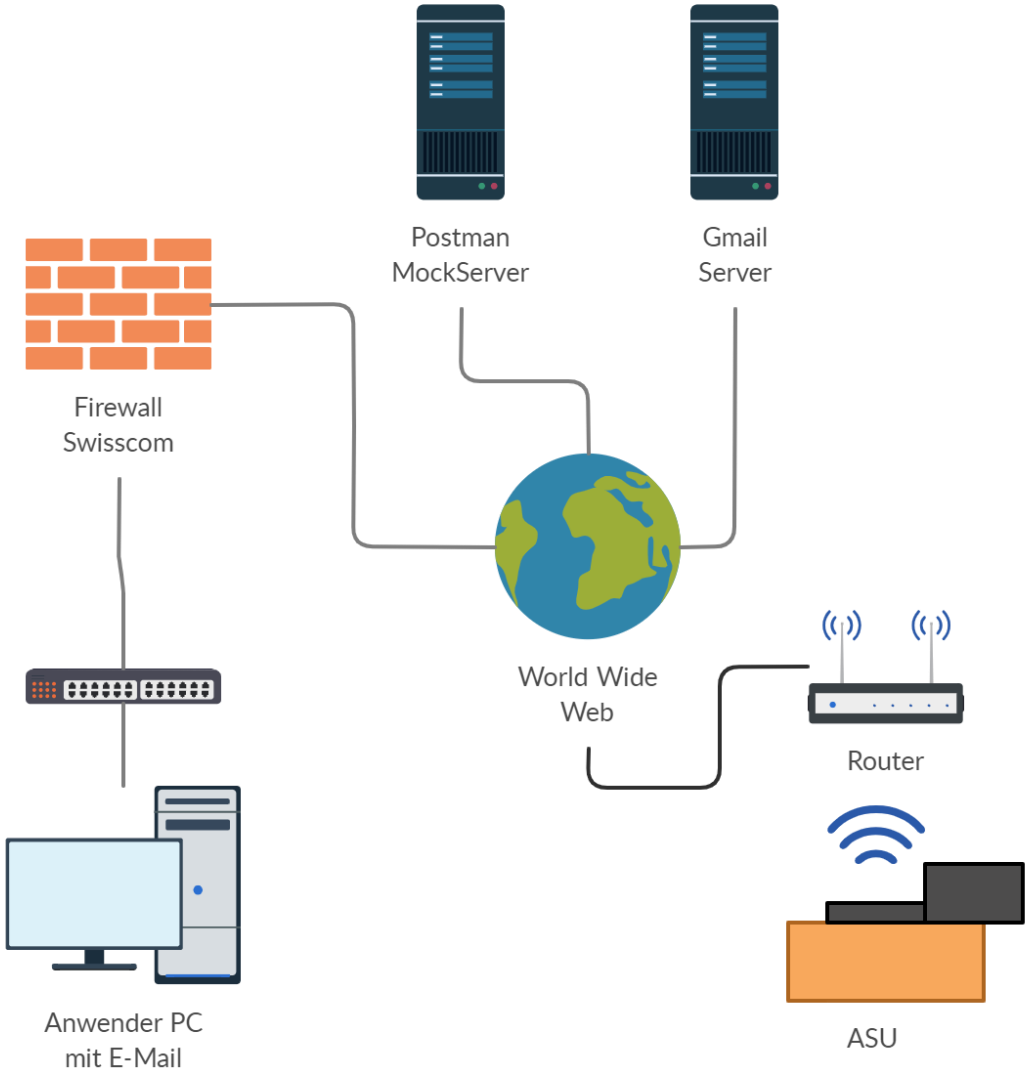



*Abbildung 31 Netzwerk ohne SNOW PoS*

Als erstes bei Postman loggt man sich mit der Google E-Mail-Adresse ein, welche man schon für die E-Mail-Adresse des ASU verwendet hat.

Nun muss man als erstes eine Rest Collection in Postman erstellen, welche man anschliessend dem Mock Server übergeben kann. In der Collection werden anschliessend alle unsere Request gespeichert.

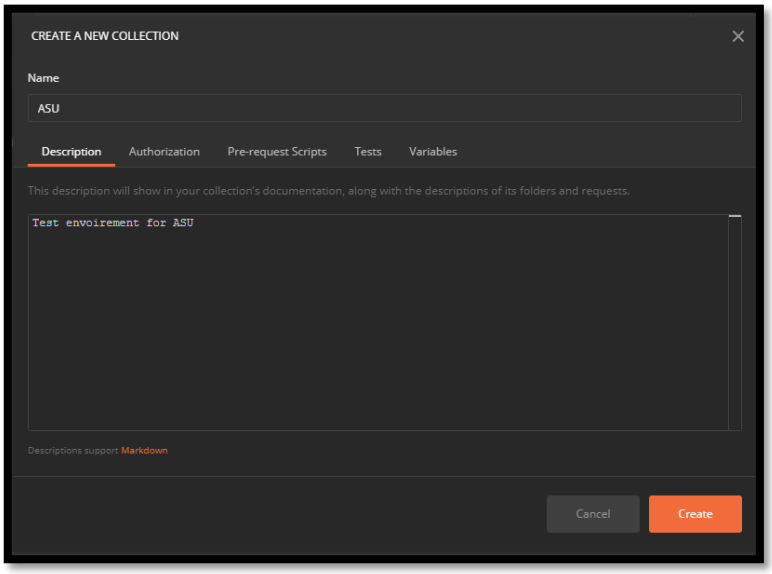



*Abbildung 32 Postman 1*

Nun kann der Mock Server erstellt werden. Da dieser angewiesen ist eine Collection zu erhalten, musste man dies im Vorfeld machen.

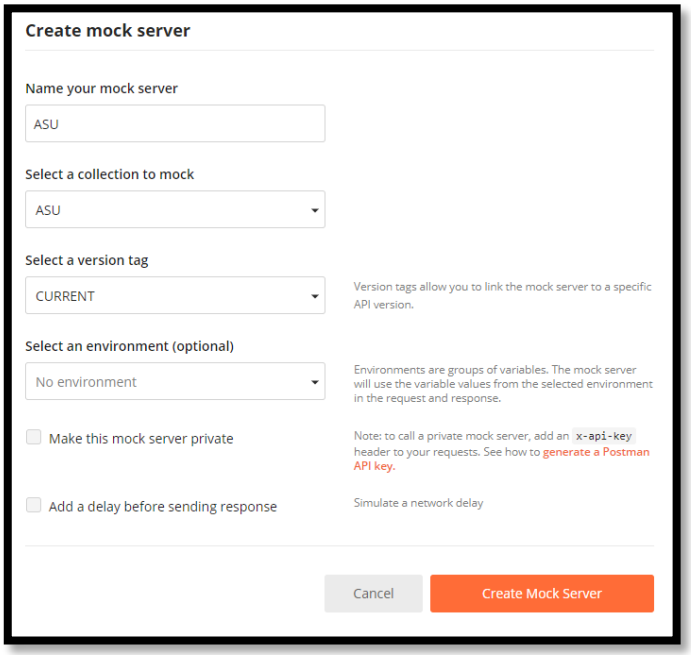



*Abbildung 33 Postman 2*

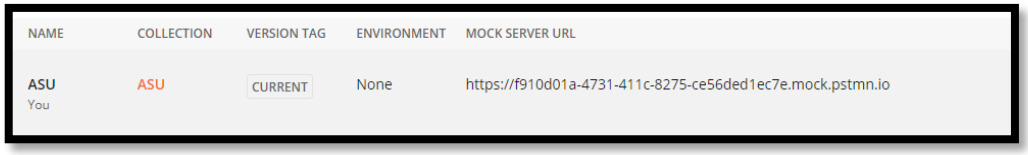



*Abbildung 34 Postman 3*

Nun wird der Link des Servers angegeben, welcher man braucht, um den Request zu bauen.

Nun kann man unter der Collection einen neuen Request erstellen. Hierbei wurde diesem Request den Name Update gegeben, da er das Update von SNOW PoS imitieren soll.

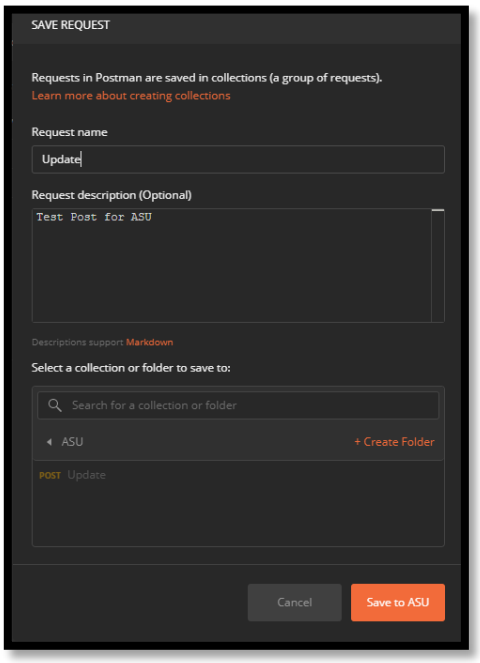



*Abbildung 35 Postman 4*

Nach dem Erstellen des Requests muss man den Link des Servers angeben und einen zusätzlichen Linkanhang machen. Hierbei wurde **/post** verwendet.

Dem Request wurde noch der Header Device mit dem value ASU hinzufgefügt. Somit weiss der Server, wenn die Abfrage vom ASU abgesetzt wird.

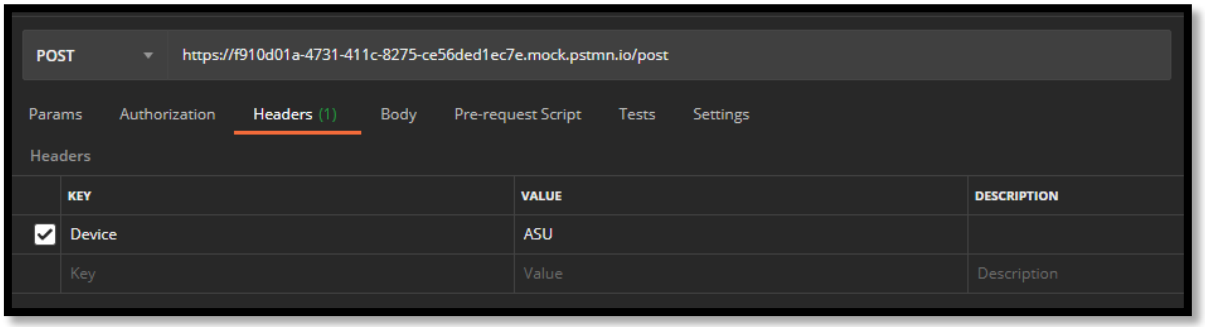



*Abbildung 36 Postman 5*

Nun muss ein Example erstellt werden. Dies ist die Antwort, welche vom Server zum ASU zurückgegeben wird, wenn man einen Post auslöst. Hierbei erstellt man ein Body, ich gebe hier eine ICCID sowie ein Status des Updates mit und ein http-Statuscode. Hierbei heisst dieser http-Statuscode 200 OK.

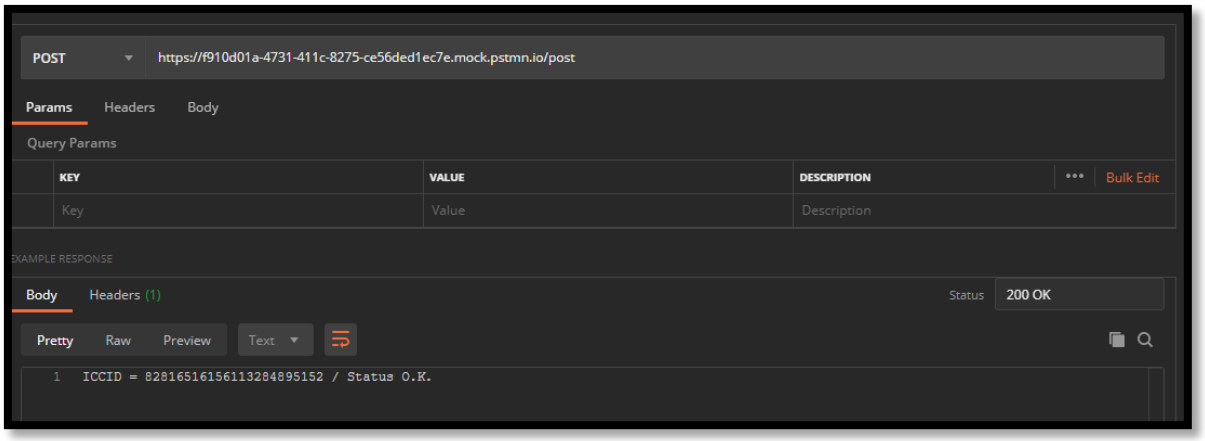



*Abbildung 37 Postman 6*

Nach dem Speichern kann man nun von Postman einen Request in allen möglichen Progrmmierssprachen erstellen lassen.

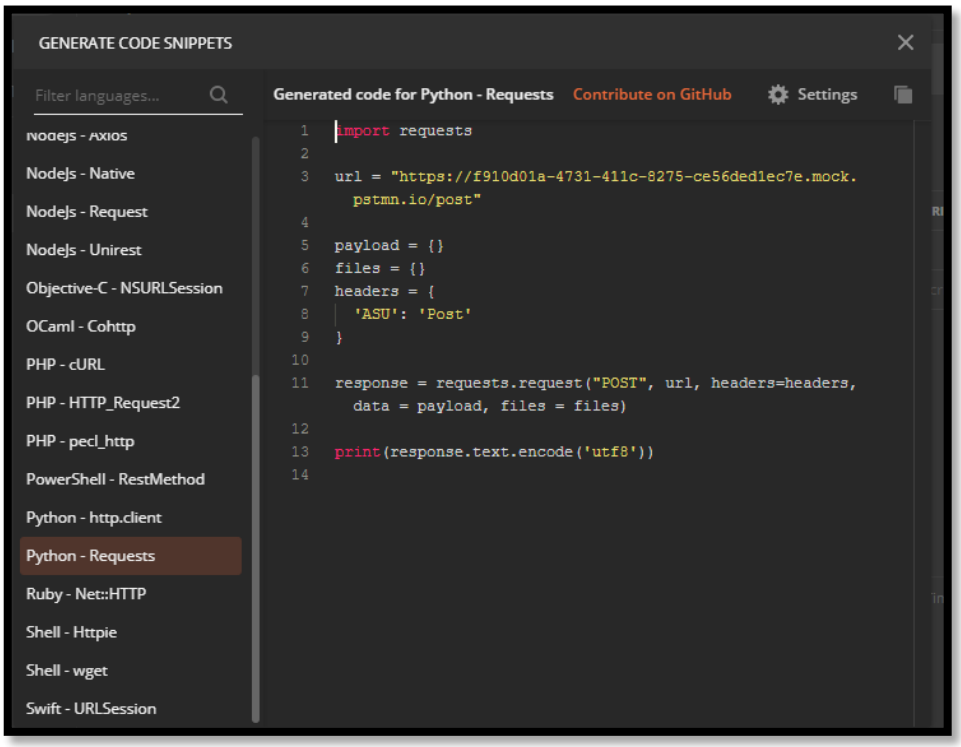



*Abbildung 38 Postman 7*

Man erstellt nun eine neue Klasse. Diese wird mit Rest benennt.

Hierbei nutzt man folgende Bibliotheken.

***requests*** - Diese Bibliothek ist dazu da, http-Anfragen zu senden. Hierzu gehört auch das Empfangen von Antworten. Also ermöglicht es einem APIs zu verwenden.

***time –*** Die Bibliothek wird verwendet, um zum Beispiel Schläfer Funktion zu nutzen also, um Pausen zu generieren mit bestimmter Zeitdauer.

***datetime –*** Sie wird vor allem dazu verwendet Datum und Uhrzeiten darzustellen.

Es wird folgende Methode erstellt:

**post** – Bei diesem fügt man den eben generierten Python code von Postman ein. Am Beginn der Methode erstellt man noch einen Zeitstempel, welcher eine Datei erstellt namens Update_liste.txt, wenn diese nicht vorhanden ist. Ansonsten fügt es den Zeitstempel in der Update_list.txt an. Zusätzlich wird die Antwort des gemachten REST.Posts auch der Update_liste.txt hinzugefügt.

Die Update Liste sieht anschliessend wie folgt aus:

```
2020-10-09 12:16:15
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:16:42
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:17:08
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:17:35
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:18:01
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:18:28
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:18:55
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:19:22
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:19:48
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:20:15
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:20:41
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:21:08
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:21:34
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:22:00
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:22:27
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:22:55
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:23:22
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:23:48
b'ICCID = 8281651615611328489515 2 / Status O.K.'
2020-10-09 12:24:14
b'ICCID = 8281651615611328489515 2 / Status O.K.'
```

*Abbildung 39 Update Liste Beispiel*

*Code* [https://github.com/jakal1989/ASU/blob/main/rest.py](https://github.com/jakal1989/ASU/blob/main/rest.py)

### 7.10.2. SIM-Erkennung für SIM-Kartenleser

Hierbei wird aufgezeigt, wie der SIM-Kartenleser erkennen kann, ob eine SIM-Karte auf dem SIM-Kartenleser vorhanden ist und wie man dieses in das Programm einbindet.

Alle üblichen USB-Smartcard Reader haben dieselbe Mechanik, um zu erkennen, ob eine SIM-Karte auf dem Lesegerät ist.

Hierbei ist immer neben den PINs des Kartenlesers ein mechanischer Taster. Wenn eine SIM-Karte ins Lesegerät eingeschoben wird, wird dieser Taster gedrückt. Falls der Taster gedrückt wird, befindet sich das Karten-Lesegerät für 10 Sekunden im Lesemodus. Wenn der Kartenleser während dieser Zeit keine Karte erkennt, geht er wieder in den Idle-Modus.

Wie schon im Verzeichnispunkt 7.3.2 beschrieben wurde hier eine Anpassung am SIM-Karten Lesegerät ausgeführt um dieses steuern zu können.

Es wurde eine neue Klasse namens Sim erstellt, in welche wieder die ***RPi.GPIO*** Bibliothek genutzt wird.

Es wurde folgende Methode erstellt:

**read_on** – Diese Methode schaltet die Überbrückung vom Relais auf das Lesegerät für 2 Sekunden ein. Anschliessend wird das Relais wieder ausgeschaltet. Diese Zeit reicht aus, um den SIM-Kartenleser für 10 Sekunden in den Lesemodus zu setzen. Somit kann bestimmt werden, wann der Lesevorgang des SIM-Karten-Lesegeräts starten soll.

*Code https://github.com/jakal1989/ASU/blob/main/sim.py*

## 7.11. [3.6] Verstopfung des Geräts oder Fehler durch Motor erkennen

Dieses Ziel wird nur zu einem Teil umgesetzt. Der Hardware- und dazugehörige Software-Teil wird nicht umgesetzt. Trotzdem wird hier eine Beschreibung gemacht, um dieses Ziel zukünftig umsetzen zu können.

Es werden einerseits Software sowie Hardware Fehlererkennungen erstellt.

Als ersten Schritt erstellt man einen Logger des Python Programms. Dieser ist dazu da allfällige Fehler im Programm festzustellen. Hierzu muss im ganzen Programm saubere Nachrichten für die Abläufe erstellt werden, welche über die Konsole ausgegeben werden können, zum Beispiel: „SIM-Karte Lesen“.

Zusätzlich ist es wichtig, wenn zum Beispiel ein If-Statement ausgeführt wird, das man einen Zeitstempel in der Konsole ausgibt. Dieser wird dementsprechend auch in das Logfile geschrieben.

Somit kann man in der Logdatei sehen wann Fehler aufgetreten sind.

Der Logger kann durch ein False oder True aus- und ein-geschaltet werden.

*Code https://github.com/jakal1989/ASU/blob/main/main.py*

*Der folgende Abschnitt wurde nicht umgesetzt:*

Als Hardwareteil nimmt man ein Volt-/Ampere-Meter, welches vom Raspberry Pi ausgelesen werden kann. Heisst, wenn der Motor hängenbleibt, entsteht auch ein höherer Stromverbrauch. Diesen Wert wird man auslesen. Nun können Definitionen getroffen werden, welcher Stromverbrauch auf einen hängenbleibenden Motor zutrifft. Hierbei wird man eine weitere elif-Kondition im Programm festlegen, wobei der Motor gestoppt wird und dem Kunden eine E-Mail-Benachrichtigung zukommen würde.


*Abbildung 40 DSN-VC288 Voltmeter Amperemeter Modul mit LED*

## 7.12. Aktivitätsdiagramm

Hier sieht man wie das Finale Programm im Aktivitätsdiagramm aussieht. Das Programm wird stark vereinfacht dargestellt, um die Grundfunktion und Ablauf des finalen Programmes darzustellen.

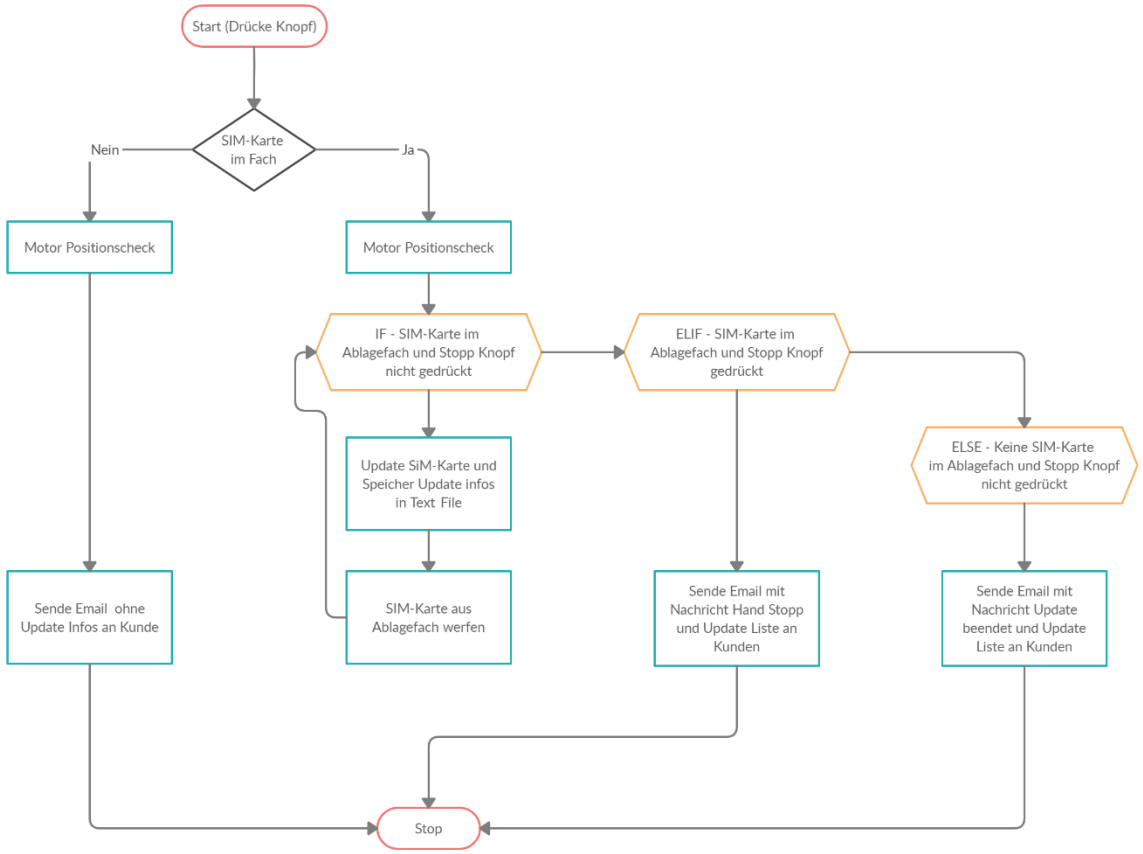



*Abbildung 41 Aktivitätsdiagramm*

## 7.13. Klassendiagramm

Hier werden die Klassen dargestellt und in welchem Bezug sie zum Programm stehen. Da man hier nur statische Methoden hat in den Klassen, haben diese grundlegend keine Bezüge zu anderen Klassen.

Alle Klassen und dazugehörigen Methoden können in dem Main Programm ausgeführt werden. Die Klassen werden im Main Programm wie Skripts abgearbeitet.

Hier soll nur dargestellt werden, wie die Klassen und dazugehörigen Programme miteinander agieren:

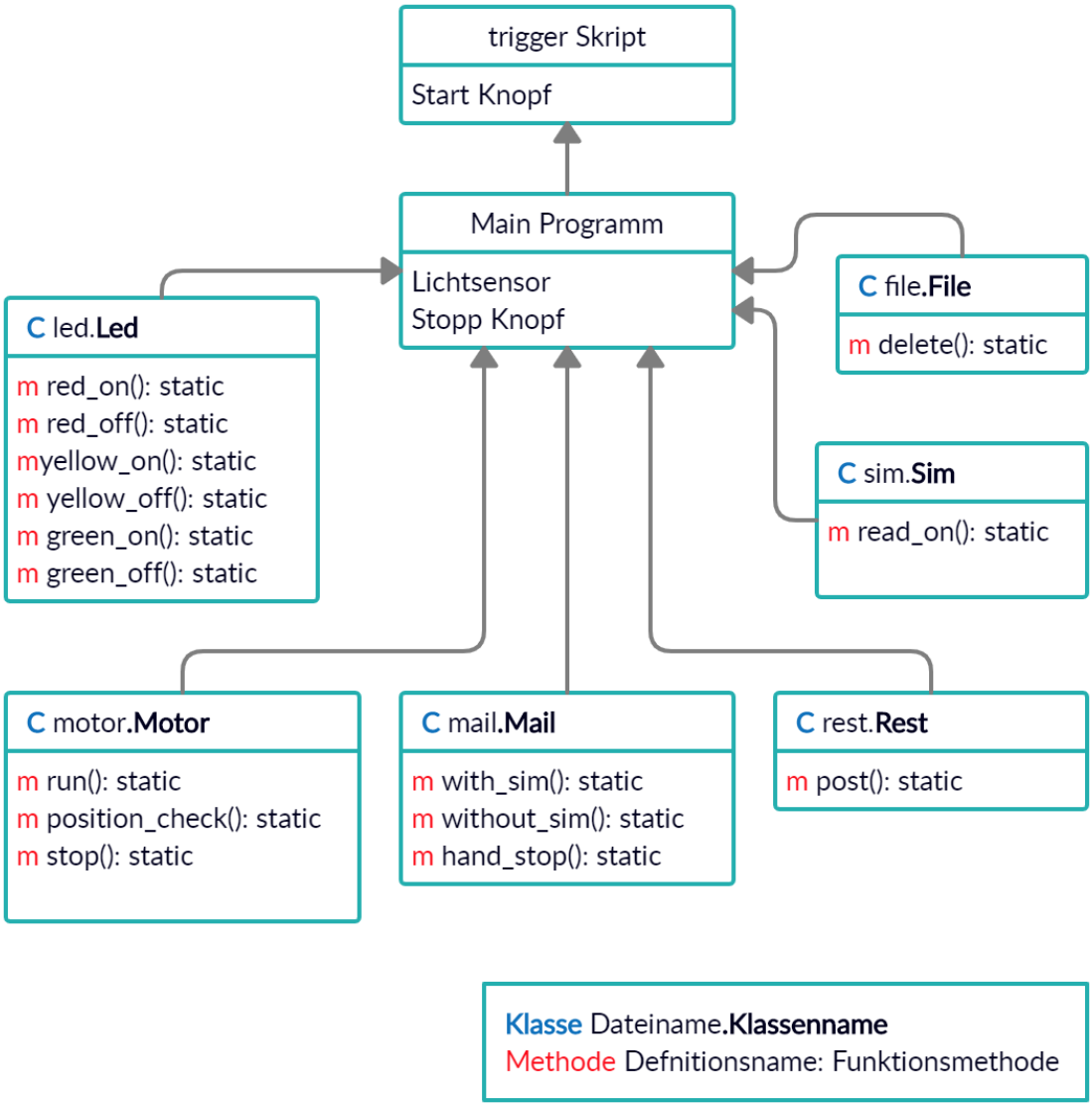



*Abbildung 42 Klassendiagramm*

## 7.14. [4.2] Testen des ASU

Nach Erreichen des Arbeitspaketes [ID-3.7] muss der komplette Automatic SIM-Card Updater nochmals auf seine Funktionsfähigkeit durchgetestet werden.

Um das fertige System zu testen, wurde schon vor Beginn des Projektes, ein Set von verschiedenen Testfällen erstellt, um möglichen Fehlern vorzubeugen.

Das Testsetup wurde in vier Kategorien unterteilt und mit wichtigen Punkten versehen:

- **SNOW PoS**: Hierbei werden Testfälle abgedeckt, welche in Zusammenhang mit dem SNOW PoS System stehen. (Testcases, welche mit SNOW PoS gekennzeichnet sind, werden hierbei mit dem Mock-Server abgedeckt, wenn möglich.)

- **Hardware Test**: Hierbei werden Testfälle abgedeckt, welche in Zusammenhang mit dem Raspberry Pi Hardware stehen.

- **Notifikation**: Hierbei werden Testfälle abgedeckt, welche in Zusammenhang mit der Kunden Notifikation stehen.

- **Stress Test**: Hierbei werden die Ausnahmefälle abgedeckt, welche bei hoher Last beim System Probleme verursachen könnten.

Es wird das Blackbox Verfahren angewendet.

Ein Black Box Verfahren ist ein spezifikationsorientiertes Testentwurfsverfahren.

Bei einem Black Box Test werden die Testfälle ausschliesslich aus der Spezifikation des zu testenden Objekts abgeleitet, ohne dabei dessen innere Struktur, den Code, zu berücksichtigen (diese werden als Black Box behandelt). Es wird also nur das von aussen sichtbaren Verhalten des Testobjektes beobachtet.

Hierbei versucht man durch positive und negative Testfälle mögliche Fehler herbeizurufen. Man erstellt dazu ein Testprotokoll.

Folgende Werte müssen enthalten sein:

- TestID
- Beschreibung
- Testvoraussetzung
- Durchführung
- Durchläufe N=?
- Testmethode
- Erwartetes Ergebnis
- Fehlerklasse
    - Fehlerfrei
    - Unwesentlicher Mangel
    - Leichter Mangel
    - Schwerer Mangel
    - Kritischer Mangel
    - Systemstillstand
- Festgestellter Fehler

### 7.14.1. Testprotokoll

| TestID | Beschreibung | Testvoraussetzung | Durchführung | Durchläufe N=? | Erwartetes Ergebnis | Testmethode | Ergebnis | Fehlerklasse | Festgestellter Fehler |
|---|---|---|---|---|---|---|---|---|---|
| | | | **SNOW PoS Test** | | | | | | |
| 001 | Keine SIM-Karte in ASU und Update starten. | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | SNOW PoS sollte nicht reagieren, da in diesem Fall kein Rest Request abgesetzt wird. | N=10 | 8/10 Update True | Black-Box | 9/10 | Unwesentlicher Mangel | Der Motor startete einmal aufgrund des Positionschecks und. ging wieder auf Nullstellung zurück |
| 002 | Über ASU einzelne SIM-Karte updaten | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | SNOW PoS sollte nach dem Start die SIM-Karte einlesen und updaten. Kontrolle auf SNOW PoS, ob dies der Fall ist. | N=10 | 8/10 Update True | Black-Box | 10/10 | | Keine Fehler festgestellt |
| 003 | Über ASU fünf SIM-Karten einlegen und dieses updaten. | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | SNOW PoS sollte nach dem Start die SIM-Karten einlesen und updaten. Kontrolle auf SNOW PoS, ob dies der Fall ist. | N=5 | 3/5 Update True | Black-Box | 5/5 | | Keine Fehler festgestellt |
| 004 | Karte verkehrt eingelegt | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | Hierbei muss die Karte verkehrt in das ASU eingelegt werden, dass die SIM-Karten Kontakte keinen Kontakt mit dem SIM-Karte-Lesegerät macht. Hierbei sollte eine Meldung von SNOW PoS erfolgen, welche einem zeigt, dass keine SIM-Karte eingelesen wurde. | N=5 | 2/5 Update keine Karte gefunden | Black-Box | x/5 | Nicht getestet | Da hier das SNOW PoS noch nicht zur Verfügung steht, wurde dieser Test nicht ausgeführt, da der Testserver diese Fehler nicht erkennen kann. |
| 005 | SNOW PoS offline | Raspberry Pi und Laptop mit SNOW PoS muss Offline sein. Netzwerk des ASU muss Online sein. | Füge eine Karte in ASU ein und starte das Update. SNOW PoS muss im Offlinemodus sein. Nach dem Start sollte eine Ausgabe, von der Rest Schnittstelle zurück kommen, welche SNOW als Offline zeigt. | N=5 | 3/5 Rest Request offline | Black-Box | x/5 | Nicht getestet | Da hier das SNOW PoS noch nicht zur Verfügung steht, wurde dieser Test nicht ausgeführt, da der Testserver diese Fehler nicht erkennen kann. |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | **Hardware Test** | | | | | | |
| 006 | Keine SIM-Karte in ASU und Update starten. | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | Durch Taster Zyklus Starten ohne SIM-Karte im Ablagefach. LED-Benachrichtigung rot (keine SIM-Karte im Fach) sowie gelb (E-Mail-Benachrichtigung) müssen aufleuchten. | N=10 | 4/5 Zyklus korrekt | Black-Box | 9/10 | Leichter Mangel | Aufgrund, dass das System schon seit einiger Zeit Online war, ist es hängen geblieben. Nach einem Neustart funktionierte es wieder. |
| 007 | Über ASU einzelne SIM-Karte updaten | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | Eine SIM-Karte einlegen. Durch Taster Zyklus starten. LED-Benachrichtigung grün (SIM-Karte im Fach) muss aufleuchten. Ein Zyklus (Motor) zum Auswerfen der SIM-Karte wird ausgeführt. Anschliessend müssen LED-Benachrichtigung rot (keine SIM-Karte im Fach) sowie gelb (E-Mail-Benachrichtigung) aufleuchten. | N=5 | 4/5 Zyklus korrekt | Black-Box | 5/5 | | Keine Fehler festgestellt |
| 008 | Über ASU fünf SIM-Karten einlegen und diese updaten. | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | 5 SIM-Karten einlegen. Durch Taster Zyklus starten. LED-Benachrichtigung grün (SIM-Karte im Fach) muss aufleuchten. Ein Zyklus (Motor) zum Auswerfen der SIM-Karte wird ausgeführt. Wiederholung bis Ablagefach leer. Anschliessend müssen LED-Benachrichtigung rot (keine SIM-Karte im Fach) sowie gelb (E-Mail-Benachrichtigung) aufleuchten. | N=5 | 4/5 Zyklus korrekt | Black-Box | 5/5 | | Keine Fehler festgestellt |
| | | | **Notifikation Test** | | | | | | |
| 009 | Keine SIM-Karte in ASU und Update starten | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online | ASU starten ohne SIM-Karte im Ablagefach. Benutzer sollte über E-Mail eine Benachrichtigung erhalten mit folgender Information: **Keine SIM in Ablagefach.** | N=10 | 8/10 E-Mail erhalten | Black-Box | 10/10 | | Keine Fehler festgestellt |
| 010 | SIM-Karten-E-Mail mit einem Eintrag | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | Eine SIM-Karte einlegen und Startknopf betätigen. Benutzer sollte über E-Mail eine Benachrichtigung erhalten mit folgender Information: **Update Status und dazugehörige ICCID.** | N=10 | 8/10 E-Mail erhalten | Black-Box | 10/10 | | Keine Fehler festgestellt |
| 011 | SIM-Karten-E-Mail mit fünf Eintragen | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | Fünf SIM-Karten einlegen und Startknopf betätigen. Nach Auswurf aller SIM-Karten sollte der Benutzer über E-Mail eine Benachrichtigung erhalten mit folgender Information: **Update Status und dazugehörige ICCID** | N=5 | 3/5 E-Mail erhalten | Black-Box | 5/5 | | Keine Fehler festgestellt |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | **Stress-Test** | | | | | | |
| 012 | Start/Stopp Test | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | Eine SIM-Karte einfügen. Starten des Updates. Direkt nach dem Start durch Betätigen der Stopp Taste, Vorgang anhalten. Gerät hält an und schickt Benachrichtigung mit folgender Information: **Update manuell angehalten.** | N=5 | 3/5 Status manuell angehalten | Black-Box | 5/5 | | Keine Fehler festgestellt |
| 013 | Starten SIM-Karte entfernen | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | Eine SIM-Karte einfügen. Update starten. Direkt nach Start des Zyklus Karte entfernen. Gerät sollte auf Status keine SIM-Karte im Fach ändern und anhalten. | N=5 | 3/5 Status keine SIM-Karte | Black-Box | 5/5 | | Keine Fehler festgestellt |
| 014 | 25 SIM-Karten Updaten. | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | 25 SIM-Karten einlegen. Durch Taster Zyklus starten. LED-Benachrichtigung grün (SIM-Karte im Fach) muss aufleuchten. Ein Zyklus (Motor) zum Auswerfen der SIM-Karte wird ausgeführt. Anschliessend müssen LED-Benachrichtigung rot (keine SIM-Karte im Fach) sowie gelb (E-Mail-Benachrichtigung) aufleuchten. Benutzer sollte über E-Mail eine Benachrichtigung erhalten mit folgenden Informationen: **Update Status und dazugehörigen ICCID aller SIM-Karten.** | N=5 | 3/5 Zyklus korrekt | Black-Box | 4/5 | Leichter Mangel | Letzte SIM-Karte eines Stapels wurde nicht sauber aus dem ASU ausgeworfen. |
| 015 | Stromausfall | Raspberry Pi und Laptop mit SNOW PoS muss eingeschaltet sein. Netzwerk Online. | 5 SIM-Karten einlegen. Durch Taster Zyklus starten, warten bis eine SIM-Karte ausgeworfen wurde. Anschliessend bei Raspberry Pi die Stromquelle entfernen. Stromquelle wieder anschliessen. Warten bis ASU aufgestartet ist. Zyklus wieder starten und checken ob der Stapel sauber abgearbeitet wird. | N=5 | 3/5 Zyklus wieder ausführbar | Black-Box | 5/5 | | Keine Fehler festgestellt |

*Tabelle 13 Testset*

### 7.14.2. [4.3] Auswertung Test

Der Test verlief ohne grössere Zwischenfälle, dennoch konnten kleinere Mängel festgestellt werden.

Diese Mängel sind einerseits auf ein zu ungenaues 3D Modell zurückzuführen, andersseits auf die Raspberry Pi Hardware.

Das Gehäuse wurde mit einer Genauigkeit von +/- 0.2 mm ausgeführt. Somit entstehen kleine Spalten auf beiden Seiten des Gehäuses. Diese Spalten führen dazu, dass die SIM-Karten zum Teil nicht sauber aus dem Gehäuse geschoben werden können. Anderseits führen die Spalten auch dazu, dass SIM-Karten teils nicht sauber vom SIM-Kartenleser gelesen werden können. Da man hier ein Prototyp erstellt, war diese Genauigkeit als erstes nicht ausschlaggebend. Man wollte nur aufzeigen, ob diese Automatisation machbar wäre. Man hat sich nicht für ein genaueres 3D Modell mit besserem Material entschieden, da dies den Rahmen des Budgets gesprengt hätte.

Der Raspberry Pi zeigte schon vor der Testphase, dass es gut ist, das Gerät ab und zu neu zu starten. Nach längerem Gebrauch kann es vorkommen, dass Befehle bei der Ausführung im Terminal hängen bleiben können und somit Probleme bei der Ausführung des Programmes verursachen.

Dieser Fehler kann ganz einfach vermieden werden, wenn man das Gerät nicht verwendet, ausschaltet.

Man muss, um den Zyklus sauber durchführen zu können, die SIM-Karten, welche im Ablagefach des Gerätes sind, noch etwas beschweren, da die 3-5 letzten Karten des Stapels sonst Schwierigkeiten haben aus dem Gerät geschoben zu werden. Dieses Problem wurde mit einigen Metall Winkel, die mit einem Kabelbinder zusammengebunden sind, gelöst.

# 8. Reflexion

## 8.1. Zielerreichung

Das Bestellen von den benutzten Waren wurde wohl meinerseits etwas unterschätzt. Da mich mein Experte darauf hingewiesen hat die Bestellung der Waren schon vor der Arbeit abzuschliessen, was hierbei sicherlich hilfreich war.

Die Muss-Ziele konnten fast alle so ausgeführt werden, wie dies vom Kunden verlangt wurde. Ich konnte meine Termine einhalten und konnte 8 der 9 Muss-Ziele nach den Zielvorgaben umsetzten.

Beim Ziel « [3.5] Automatisch SIM-Karte über PC updaten (Skript auf PC)» stiess ich auf ein Problem, welches von mir nicht gelöst werden konnte. Die Firma, bei welcher im Frühjahr die Software bestellt wurde, um die Updates der SIM-Karten auszuführen, konnte aufgrund mehrerer Verzögerungen ihren Liefertermin nicht einhalten. Somit konnte ich für den Kunden den Roboter soweit vorbereiten, dass falls die Software verfügbar ist, nur kleinere Anpassungen am ASU vorgenommen werden müssten.

Das einzelne Kann-Ziel wurde nur zum Teil umgesetzt, da man hier Zeit und Geld einsparen wollte. Trotzdem wurde eine Beschreibung für dieses Ziel gemacht, um es zukünftig umsetzen zu können.

Der Kunde gab ein Positives Feedback zurück, auch wenn er bis jetzt den ASU nicht einsetzen konnte.

Die Kosten für die Hard- und Software konnten hier unter 500 CHF gehalten werden und liegen somit im Budget.

## 8.2. Ablauf

Durch die genaue Strukturierung über den Terminplan konnte ich von Anfang an meine Abläufe einhalten.

Ich hielt mich stets an meinen Plan und konnte somit den kritischen Pfad sehr gut einhalten. Trotzdem kam es auch bei mir vor, dass Probleme entstanden sind, welche ich nicht am geplanten Tag des Arbeitspakets lösen konnte.

Da ich schon für solche Fälle Zeitschlitze für allfällige Probleme festgelegt habe, konnte dies zu einem späteren Zeitpunkt dazwischengeschoben werden und ohne Verzögerung der anderen Arbeitspakete gelöst werden.

Ich hatte häufiger Probleme beim Ablauf auch noch die Dokumentation auf dem aktuellen Stand zu halten. Hierbei wurden kleinere Notizen geschrieben und Verweise zu Links, um die Dokumentation in den letzten 2 Wochen des Projekts noch sauber zu ergänzen.

## 8.3. Lessons Learned

Während meiner Diplomarbeit konnte ich sehr viel bereits gelerntes Wissen einsetzen. Ich konnte mich vor allem im Thema Python programmieren, sowie der Steuerung von Motoren durch Microcomputer vertiefen.

Bei diesem Projekt wurde mein Verständnis über das Programmieren einige Male auf die Probe gestellt. Heisst ich kam nicht weiter, da mir unbekannte Fehler aufgetreten sind, welche auch mit Online Recherche nicht so schnell gelöst werden konnten.

Dank Arbeitskollegen, welche tiefere Programmierkenntnisse haben als ich, konnten ich einiges Wissen abholen, welches mich häufig auf die richtige Lösung lenkte.

Die Arbeit mit der Hardware fiel mir leicht und ich konnte sie auch schnell umsetzen. Trotzdem merkte ich auch hier, dass es ratsam ist für eine günstige Hardware Komponente direkt eine Reserve mitzukaufen. Dies war der Fall mit den SIM-Karten Lesegerät. Hierbei ging mir das erste in Brüche, welches ich nochmals besorgen musste.

Da ich auf einem mir noch teils unbekannten Thema arbeitete, musste ich mich in viele Themen Reinlesen und viel Recherche betreiben.

## 9. Schlusswort

Ich möchte mich hiermit bei allen Beteiligten meiner Diplomarbeit bedanken. Sehr grossen Dank an Mathias Hubacher, Iwan Kissling, Gerhard Gaudard und Christoph Moser, welche mir häufig Fragen zum Programmieren beantworten konnten.

Auch meinem Auftraggeber Thomas Wüthrich möchte ich ein grosses Dankeschön aussprechen, dass ich dieses abwechslungsreiche Projekt für ihn erarbeiten durfte.

Natürlich auch ein grosses Dankeschön dem Diplomlehrer Fabian Hirter, welcher alle meine Fragen bezüglich Projekt Dokumentation in Windeseile auch an Wochenenden und spät abends beantwortet hat.

Zum Schluss muss ich mich noch bei meiner Freundin Stefanie Mischler bedanken, da sie mir tatkräftige Unterstützung gab indem sie alle Hausarbeiten übernommen hat während dieser Zeit und natürlich meiner Mutter, welche wie schon bei den letzten Semesterarbeiten, die Grammatikkorrektur überprüft hat.

Für mich war diese Diplomarbeit ein spannendes, aber sehr stressiges Erlebnis. Ich hatte sehr viele Hochs und genauso viele Tiefs. Ich freute mich über jeden Punkt, der funktioniert hat. Doch konnte ich mich auch sehr darüber ärgern, wenn etwas nicht funktioniert hat. Ich bin froh, konnte ich etwas Nützliches für jemandem umsetzten.

Ich bin zufrieden mit meiner Arbeit und hoffe, dass das fehlende Programm bald eintrifft, um für meine Kollegen noch die finale Anpassung zu machen, so dass sie dies in ihrem Arbeitsalltag nutzen können.

## 10. Selbständigkeitserklärung

Ich bestätige, dass ich die vorliegende Diplomarbeit selbstständig verfasst und alle benutzten Quellen gekennzeichnet habe. Diese Arbeit wurde weder in gleicher noch in ähnlicher Form bereits einer Prüfungskommission vorgelegt.

Diplomarbeit

Automatic SIM-Card Updater - ASU

Cugis Stefano

Bern / 01.11.2020

# 11. Verzeichnisse

## 11.1. Quellenverzeichnis

| Bücher / Quellen / Links | Datum |
|---|---|
| Projektmanagement für Führungsfachleute - Compendio | Auflage 5. 2017 |
| SNOW_PoS_description_20e.pdf (Herausgabe auf Anfrage NDA) | Erstellt am 09.09.2020 |
| https://tutorials-raspberrypi.de/raspberry-pi-schrittmotor-steuerung-l293d-uln2003a/ | 01.09.2020 |
| https://tutorials-raspberrypi.de/raspberry-pi-servo-motor-steuerung/ | 01.09.2020 |
| https://www.makershop.de | 01.09.2020 |
| https://www.reichelt.com | 01.09.2020 |
| https://www.digitec.ch | 01.09.2020 |
| https://www.play-zone.ch | 01.09.2020 |
| https://www.conrad.ch | 01.09.2020 |
| https://www.fiverr.com/ | 02.09.2020 |
| https://www.gaffuri3d.ch | 11.09.2020 |
| https://creately.com/de/home/ | 18.09.2020 |
| https://www.raspberrypi.org | 21.09.2020 |
| https://www.elektronik-kompendium.de/sites/raspberry-pi/1905261.htm | 21.09.2020 |
| https://de.pinout.xyz/pinout/pin1_3v3_stromversorgung | 21.09.2020 |
| https://howtomechatronics.com/how-it-works/how-servo-motors-work-how-to-control-servos-using-arduino/ | 22.09.2020 |
| https://www.itwissen.info/Java-Java.html | 28.09.2020 |
| https://www.itwissen.info/Python.html | 28.09.2020 |
| https://www.itwissen.info/C-plus-plus.html | 28.09.2020 |
| https://www.python.org | 28.09.2020 |
| https://tutorials-raspberrypi.de/raspberry-pi-gpio-erklaerung-beginner-programmierung-lernen/ | 28.09.2020 |
| https://www.explainingcomputers.com/pi_servos_video.html | 28.09.2020 |
| http://www.uugear.com/portfolio/using-light-sensor-module-with-raspberry-pi/ | 29.09.2020 |
| https://tutorials-raspberrypi.de | 29.09.2020 |
| https://myaccount.google.com | 30.09.2020 |
| https://variax.wordpress.com/2017/03/07/send-mail-from-raspberry-pi-command-line-andor-python-script/ | 30.09.2020 |
| https://blog.heimetli.ch/raspberry-pi-taste-programmstart.html | 01.10.2020 |
| https://www.elektronik-kompendium.de/sites/kom/0401111.htm | 01.10.2020 |
| https://www.datacamp.com/community/tutorials/elif-statements-python | 04.10.2020 |
| https://www.postman.com | 05.10.2020 |
| https://learning.postman.com/docs/designing-and-developing-your-api/mocking-data/setting-up-mock/ | 05.10.2020 |
| https://medium.com/@elijahvalentinetroyabako/simulating-a-back-end-with-postmans-mock-servers-572e1f2ac400 | 07.10.2020 |
| https://www.askpython.com/python/built-in-methods/python-print-to-file | 08.10.2020 |
| https://stackoverflow.com/questions/2513479/redirect-prints-to-log-file | 20.10.2020 |
| https://www.amazon.de/perfk-Voltmeter-Amperemeter-Arduino-Raspberry/dp/B079Z6832V?tag=coa_de_g-21 | 21.10.2020 |

*Tabelle 14 Quellenverzeichnis*

## 11.2. Glossar

| Wörter | Erklärung |
|---|---|
| Bashskript | Ein Bashskript oder Shell-Skript ist ein Computerprogramm, das von einer Shell interpretiert und ausgeführt wird. |
| BlackBox Verfahren | Black-Box-Test bezeichnet eine Methode des Software-Tests.<br>Tests werden ohne Kenntnisse über die innere Funktionsweise / Implementierung des zu testenden Systems entwickelt. |
| Body | Der Body ist der Teil einer Rest Abfrage, in welchem man die erzeugten Ressourcen sehen kann. |
| Confidential | Geheime Daten, nicht für andere bestimmt. |
| elif-Funktion | Erlaubt mehr Optionen zum «springen» als die einfache if-Funktion. |
| else-Funktion | Else Funktion wird ausgeführt, falls die if Bedingung nicht zutrifft. |
| Example | Das Example in Postman wird als Test Rückmeldung genutzt. Hierbei gibt der Mockserver auf eine Definierte Anfrage. Eine definierte Antwort zurück |
| GPIO | general purpose input/output ist ein allgemeiner Kontaktstift an einem integrierten Schaltkreis (IC), dessen Verhalten, unabhängig, ob als Eingabe- oder Ausgabekontakt, durch logische Programmierung frei bestimmbar ist |
| Idle-Modus | Ist ein Modus in welchem ein Gerät, keine Operationen ausführt aber Trotzdem eingeschalten ist. |
| if-Funktion | If-Anweisung erlaubt es einem, im Code zu "springen" und somit den sogenannten Kontrollfluss zu verändern. |
| Kick-off-Meeting | Aus dem Englischen übersetzt - Ein Kickoff-Meeting ist das erste Meeting mit dem Projektteam und dem Kunden des Projekts. |
| Logger | Ein Logger ist eine prozessorgesteuerte Speichereinheit, welche Daten in einem bestimmten Rhythmus über eine Schnittstelle aufnimmt und auf einem Speichermedium ablegt. |
| Logger | Ein Logger ist eine prozessorgesteuerte Speichereinheit, welche Daten in einem bestimmten Rhythmus über eine Schnittstelle aufnimmt und auf einem Speichermedium ablegt. |
| Makro | Makros sind eine Variante von Unterprogrammen und können (je nach Implementierung) auch mit Parametern aufgerufen werden. Man unterscheidet Systemmakros (wie OPEN zum Dateiöffnen und PRINT zum Drucken) und von Benutzern selbst erstellte Makros – zum Beispiel um Prüf- oder Berechnungsfunktionen auszuführen wie die Gültigkeitsprüfung für die Internationale Bankkontonummer (IBAN). |
| Mockserver | MockServer ist ein Open Source-Mocking-Framework für HTTP und HTTPS, das unter der Apache-Lizenz veröffentlicht wurde. Mock Server wurde entwickelt, um Integrationstests zu vereinfachen. |
| Proof of concept | Es ist dazu bestimmt, zu entscheiden, ob eine Idee in die Realität umgesetzt werden kann, soll ein Prototyp diese Idee in eine abgespeckte Version des Endprodukts umsetzen, die auf Nutzbarkeit, Funktionalität und Design getestet und bewertet werden kann. |
| PWM | Die PWM ist eine digitale Modulationsart, bei der eine technische Größe (z. B. elektrische Spannung) zwischen zwei Werten wechselt. Dabei wird bei konstanter Frequenz ein Rechteckimpuls moduliert, dessen Weite, Breite bzw. Länge variiert. Das Verhältnis zwischen Impuls und Pause wird als Tastgrad bezeichnet. |
| Python | Die Programmiersprache Python wurde Anfang der 90er Jahre von Guido van Rossum in Amsterdam veröffentlicht. Die Sprache wurde im Hinblick auf die englische Komiker Truppe Monty Python benannt und ist eine dynamisch typisierte objektorientierte Programmiersprache. |

| | |
|---|---|
| Raspberry Pi | Der Raspberry Pi ist ein Einplatinencomputer, der von der britischen Raspberry Pi Foundation entwickelt wurde. Der Rechner enthält ein Ein-Chip-System von Broadcom mit einer ARM-CPU. Die Platine hat das Format einer Kreditkarte. |
| REST.post | Schnittstelle zum Abfragen von Informationen. Beim Post werden beim Aufruf, neue Ressourcen im System angelegt. |
| REST.request | Schnittstelle zum Abfragen von Informationen. Beim request werden angelegte Ressourcen abgefragt. |
| Router | Allgemein versteht man unter Access Routern (AR) Funktionseinheiten, die Ausseneinrichtungen mit einem grösseren Netzwerk, einem Unternehmensnetz oder dem Internet verbinden. |
| SD-Karte | Die microSD-Karte ist eine miniaturisierte SD-Karte. Die kleine Speicherkarte ist äußerst kompakt und hat Abmessungen von nur 11 mm × 15 mm. Sie ist in drei Versionen mit verschiedenen Speichergrößen und Geschwindigkeitsklassen lieferbar. |
| Sharepoint | Ist eine Webanwendung von Microsoft, die unter anderem folgende Anwendungsgebiete abdeckt: Zusammenarbeit, beispielsweise das Verwalten von Projekten oder die Koordination von Aufgaben. |
| SNOW | SunTiS New OTA – Over the Air update Service für SIM-Karten |
| SNOW PoS | SNOW ist ein Over-the-air Update Programm, welches bei Swisscom eingesetzt wird, um Updates auf SIM-Karten zu spielen.<br>POS (Point of Sale) wird ein feature von SNOW genannt, um Updates automatisiert über USB updaten zu können. |
| Textfile | Als Textdatei wird in der Informationstechnik eine Datei bezeichnet, die darstellbare Zeichen enthält. |
| UNIX | Unix ist ein von den Bell Laboratories in den 70er Jahren entwickeltes Betriebssystem für Minicomputer, inzwischen für einen weiten Bereich von Rechnern vom Personal Computer (PC) bis hin zum großen Mainframe verfügbar. |
| Update | Aktualisierte [und verbesserte] Version einer Software, einer Datei o. Ä. |

*Tabelle 15 Glossar*

## 11.3. Abbildungsverzeichnis

## 11.4. Tabellenverzeichnis

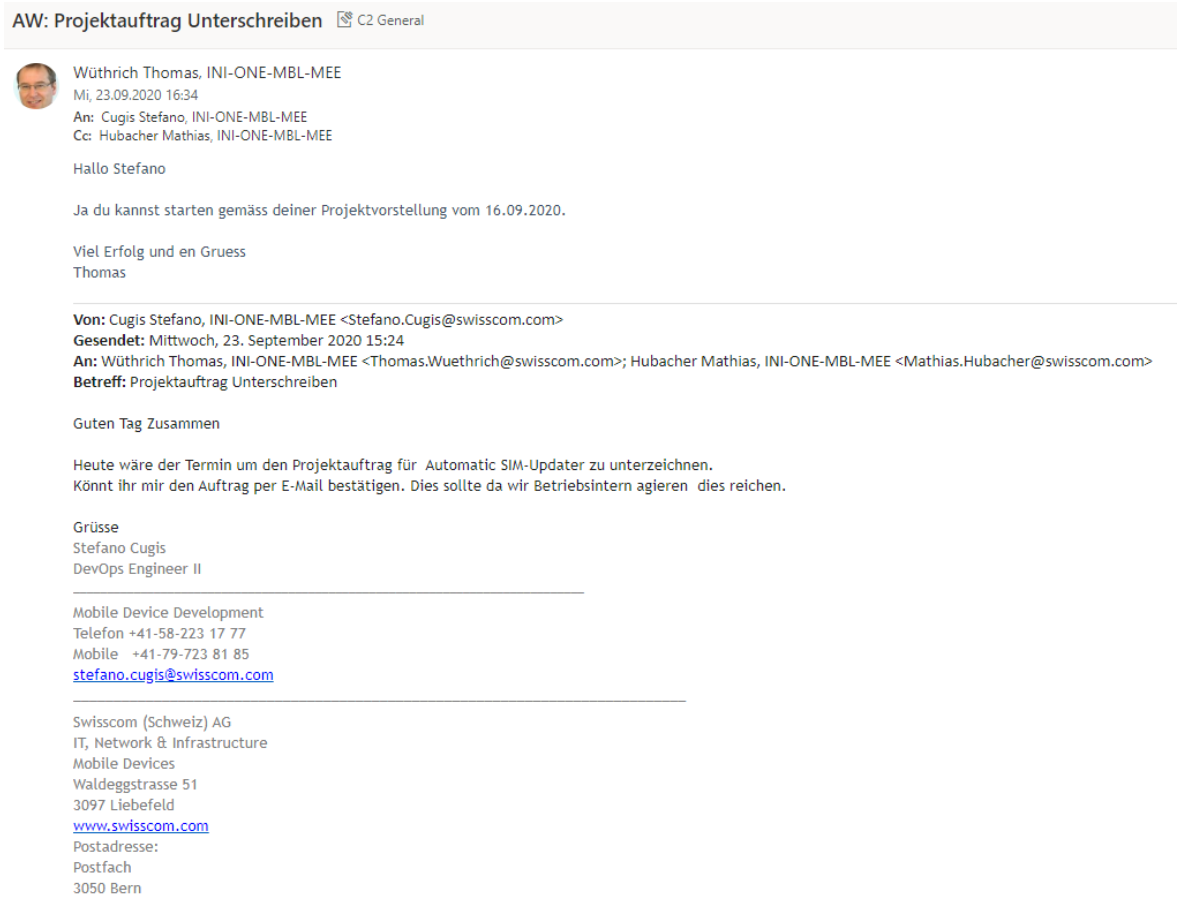
## 11.5. Abkürzungsverzeichnis

| Abkürzungen | Erklärung |
|---|---|
| A | Ampere Masseinheit für Strom |
| API | Application Programming Interface |
| CHF | Schweizer Franken |
| FTP | File Transfer Protocol |
| GPIO | general purpose input/output |
| h | Masseinheit Stunde |
| HDMI | High Definition Multimedia Interface |
| ICCID | Integrated Circuit Card Identifier |
| IMAP | Internet Message Access Protocol |
| LUX | Beleuchtungsstärke Masseinheit |
| MOD | Mobile Devices |
| NDA | non-disclosure agreement / Geheimhaltungsvertrag |
| PoS | Point of Sale |
| PWM | pulse width modulation |
| SD | Secure Digital |
| SIM | Subscriber Identification Module-Karte |
| SMSC | Short Message Service Centre |
| SMTP | Simple Mail Transfer Protocol |
| STL | Standard Triangulation/Tesselation Language |
| USB | Universal Serial Bus |
| V | Volt Masseinheit für Spannung |

*Tabelle 16 Abkürzungsverzeichnis*

# 12. Anhang

## 12.1. Projektauftrag Bestätigung



AW: Projektauftrag Unterschreiben C2 General

Wüthrich Thomas, INI-ONE-MBL-MEE
Mi, 23.09.2020 16:34
**An:** Cugis Stefano, INI-ONE-MBL-MEE
**Cc:** Hubacher Mathias, INI-ONE-MBL-MEE

Hallo Stefano

Ja du kannst starten gemäss deiner Projektvorstellung vom 16.09.2020.

Viel Erfolg und en Gruess
Thomas

---

**Von:** Cugis Stefano, INI-ONE-MBL-MEE <Stefano.Cugis@swisscom.com>
**Gesendet:** Mittwoch, 23. September 2020 15:24
**An:** Wüthrich Thomas, INI-ONE-MBL-MEE <Thomas.Wuethrich@swisscom.com>; Hubacher Mathias, INI-ONE-MBL-MEE <Mathias.Hubacher@swisscom.com>
**Betreff:** Projektauftrag Unterschreiben

Guten Tag Zusammen

Heute wäre der Termin um den Projektauftrag für Automatic SIM-Updater zu unterzeichnen.
Könnt ihr mir den Auftrag per E-Mail bestätigen. Dies sollte da wir Betriebsintern agieren dies reichen.

Grüsse
Stefano Cugis
DevOps Engineer II

---

Mobile Device Development
Telefon +41-58-223 17 77
Mobile +41-79-723 81 85
stefano.cugis@swisscom.com

---

Swisscom (Schweiz) AG
IT, Network & Infrastructure
Mobile Devices
Waldeggstrasse 51
3097 Liebefeld
www.swisscom.com
Postadresse:
Postfach
3050 Bern

*Abbildung 43 Bestätigung Projektauftrag*

## 12.2. Bedienungsanleitung

Bedienungsanleitung

**Automatic SIM-Card Updater- ASU**

*Abbildung 44 Bedienungsanleitung 1*

Inhaltsverzeichnis





*Abbildung 45 Bedienungsanleitung 2*

### 1. Inbetriebnahme ASU

Der ASU muss an den Strom angeschlossen werden. Zusätzlich muss der ASU über den Schalter am Kabel des schwarzen Ladegerätes eingeschaltet werden. Wenn die grüne Led beim Lichtsensor leuchtet, ist der ASU eingeschaltet. Gerät muss in Reichweite von Router MBBH sein. SIM-Karten Leser an PC mit SNOW PoS Software anschliessen.

Der ASU sollte an einem gut ausgeleuchteten Standort stehen.

| Schalter Aus | Schalter Ein |
|---|---|
|  |  |

Automatic SIM-Card Updater 3/8

*Abbildung 46 Bedienungsanleitung 3*

## 2. Funktion

### 2.1. Befüllen

Nach dem Aufbau kann man das ASU mit den zu updatenden SIM-Karten befüllen. Am Einfachsten macht man dies folgendermassen:

Einen kleinen Winkel nach vorne, anschliessend die SIM-Karten fallen lassen. Nach dem Befüllen muss noch das Zusatzgewicht auf die SIM-Karten gelegt werden.

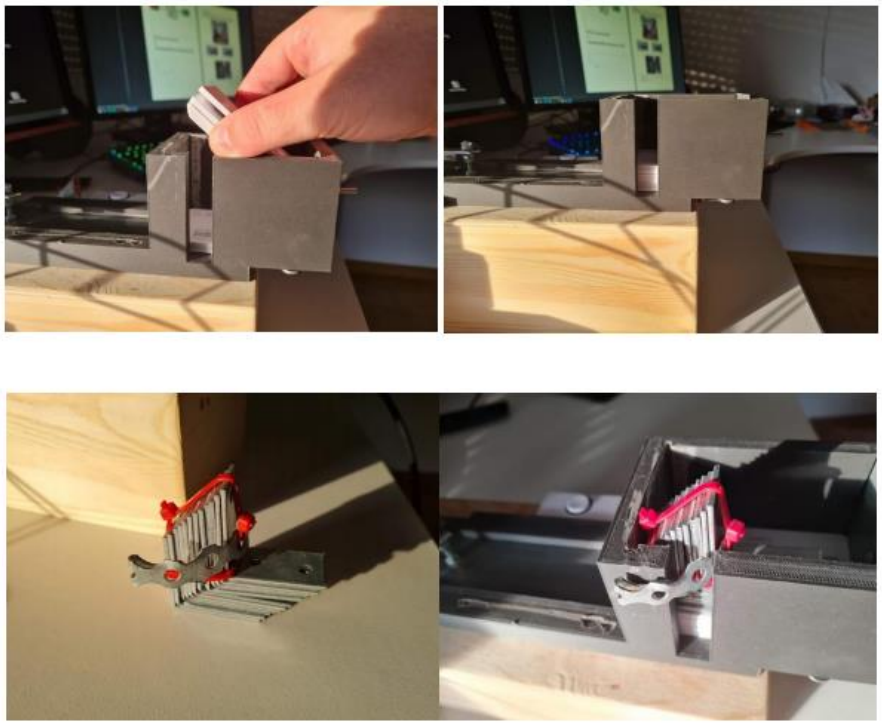




*Abbildung 47 Bedienungsanleitung 4*

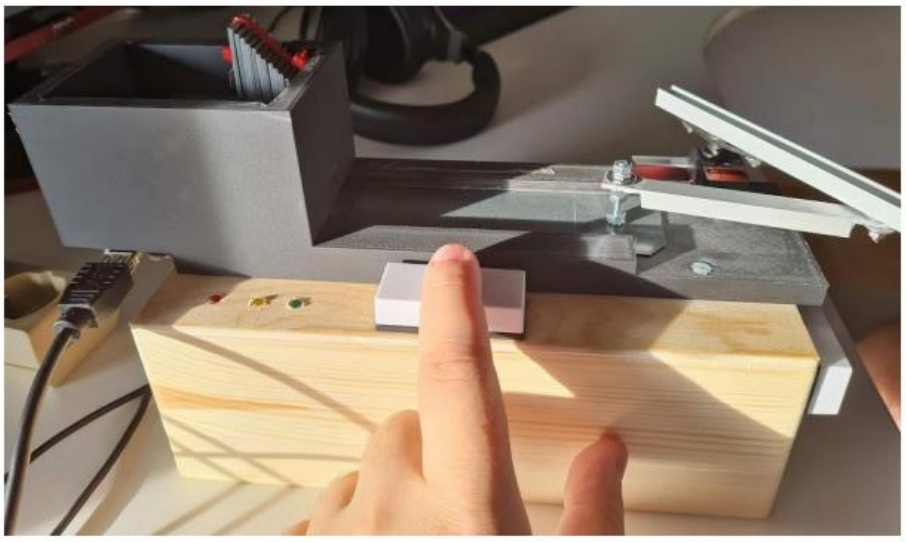
### 2.2. Kontrolle Einschub

Nach dem Befüllen ist es wichtig den Einschub zu kontrollieren, da sich das Blech verschieben könnte beim Einfügen neuer SIM-Karten. Wen dieses die SIM-Karten nicht aus dem Gehäuse lässt, muss man das Blech noch etwas in die richtige Position bewegen. Hierbei sollte eine SIM-Karte komplett von aussen sichtbar sein.

| Korrekt | Falsch |
| --- | --- |
|  |  |

### 2.3. Update starten

Nach der Vorbereitung kann nun das Update gestartet werden. Hierbei muss nur der Start Knopf gedrückt werden, Nach ca. 3 Sekunden sollte die grüne Led angehen und der Zyklus sollte starten.



Automatic SIM-Card Updater 5/8

*Abbildung 48 Bedienungsanleitung 5*

### 2.4. Update von Hand stoppen

Es besteht die Möglichkeit, das Gerät anzuhalten, indem man auf den Stopptaster drückt. Hierbei wurde das Programm so geschrieben, dass man den Taster gedrückt halten muss bis der Stopp Vorgang beginnt, um nicht versehentlich einen Stopp zu generieren. Hierbei ist es am besten, direkt nachdem der Schieber aus dem Ablagefach ist, die Stopptaste zu drücken, da man so die kürzeste Wartezeit hat.

Wenn der Stopp Vorgang startet, sollte die grüne Led ausschalten und die rote und gelbe Led einschalten. Die Benachrichtigung sollte nun beim Nutzer angelangt sein.



*Abbildung 49 Bedienungsanleitung 6*

### 2.5. Update beenden durch keine SIM-Karte im Ablagefach

**!WICHTIG! – Der Standort des ASU sollte gut ausgeleuchtet sein**

Wen der ASU die letzte SIM-Karte aus dem Gehäuse geschoben hat, wird der ASU automatisch gestoppt. Hierbei wird wie auch beim Stopp Knopf die rote wie die gelbe Led angezeigt. Die Benachrichtigung sollte nun beim Nutzer angelangt sein.

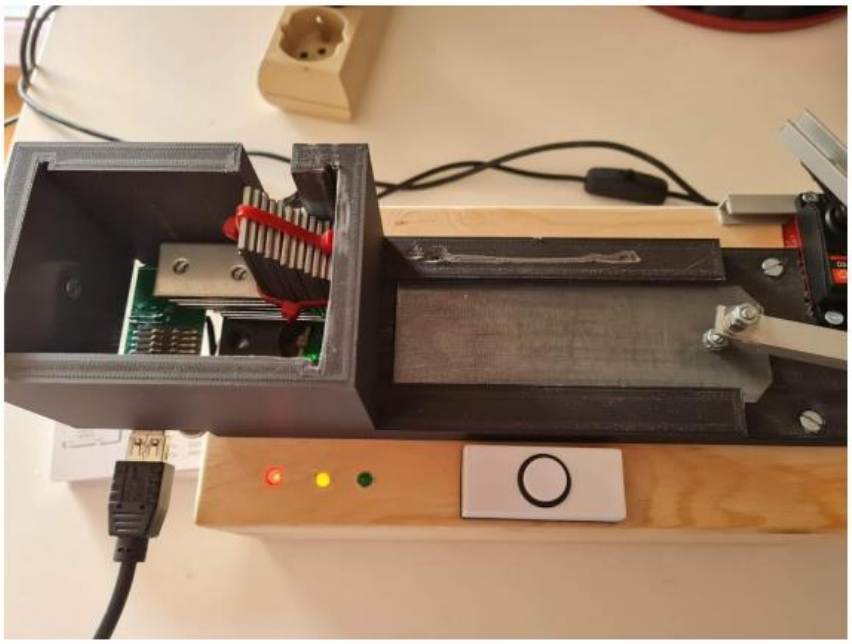




*Abbildung 50 Bedienungsanleitung 7*

## 3. Mögliche Fehler

### 3.1. Motor hängt nach gelaufener Abarbeitung im Gehäuse

| | |
|---|---|
| Beschreibung | Dies kann passieren, wenn der Motor etwas zu viel Kraft gebraucht hat, um eine SIM-Karte aus dem Ablagefach zu schieben. |
| Behebung des Fehlers | Den Start Knopf drücken. Das Programm enthält ein Positionscheck des Motors. Hierbei wird der Motor auf die Nullstellung gestellt. Wenn dann keine SIM-Karte im Ablagefach ist, wird dieser auch gleich wieder ausgeschalten. |

### 3.2. Leds leuchten weiter nach Beendigung des Programmes

| | |
|---|---|
| Beschreibung | Dies kann passieren, wenn ein Command im Terminal hängen bleibt. |
| Behebung des Fehlers | Hierbei hilft es das Gerät auszuschalten, 20 Sekunden zu warten und dann wieder einzuschalten. Dies sollte die überschüssigen Commands aus dem Cache leeren. |

### 3.3. Zyklus läuft weiter obwohl alle SIM-Karten ausgeworfen wurden

| | |
|---|---|
| Beschreibung | Dies kann passieren wen der ASU an einem zu dunklen Ort steht. Somit erkennt er nicht, ob noch eine SIM-Karte vorhanden ist oder nicht. |
| Behebung des Fehlers | Mit einer starken Lichtquelle in das Ablagefach leuchten bis der Stopp Vorgang eingeleitet wird oder den Stopp Taster am ASU drücken.<br>Das Gerät wenn möglich direkt unter einer Lichtquelle wie Stehlampe oder Tischlampe positionieren. |

### 3.4. Karte wird nicht aus dem Ablagefach geschoben

| | |
|---|---|
| Beschreibung | SIM-Karte wird nur halb aus dem Ablagefach gezogen oder SIM-Karte wird gar nicht aus dem Ablagefach gezogen. |
| Behebung des Fehlers | Dies kann passieren, wenn der Kartenbegrenzer nicht sauber eingestellt ist, oder das Zusatzgewicht nicht auf dem Stapel liegt. |



*Abbildung 51 Bedienungsanleitung 8*

## 12.3. Ordnerstruktur

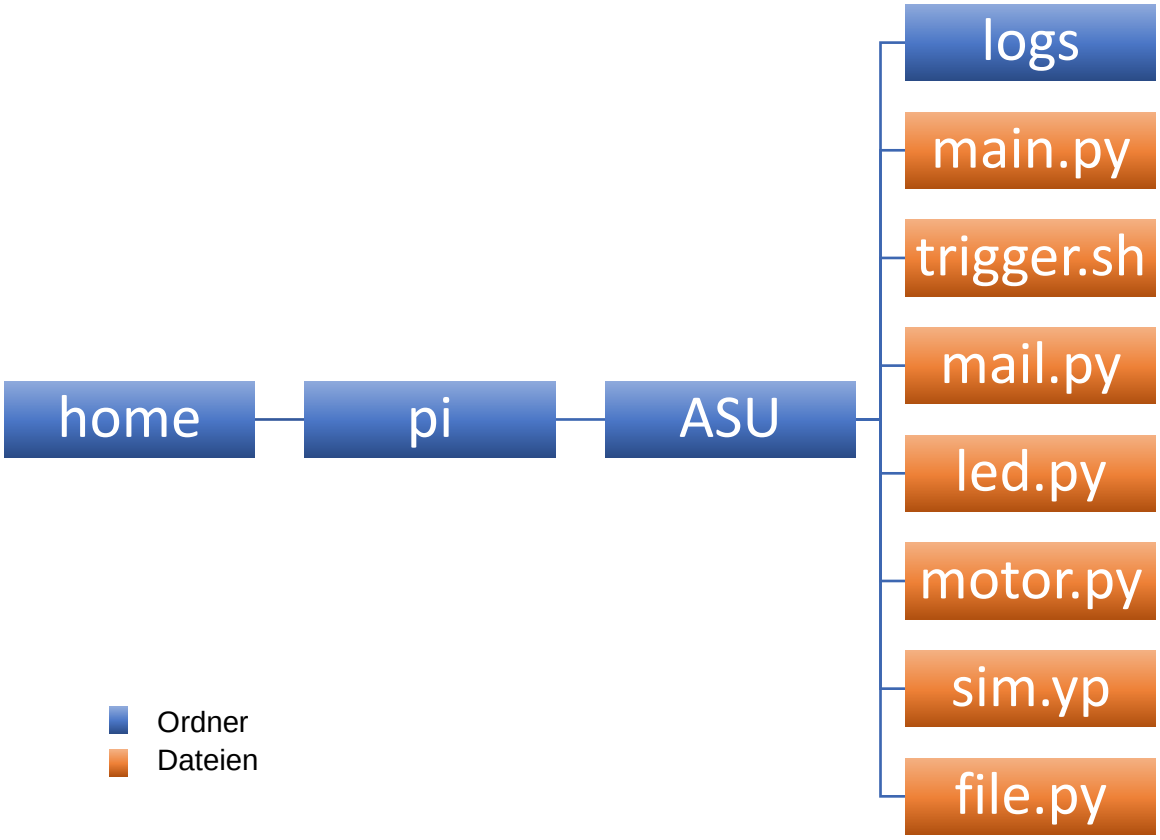



*Abbildung 52 Ordnerstruktur*

## 12.4. Abgeschlossene Arbeit

Der Code kann über GitHub eingesehen werden. Das Repository findet man unter diesem Link. https://github.com/jakal1989/ASU

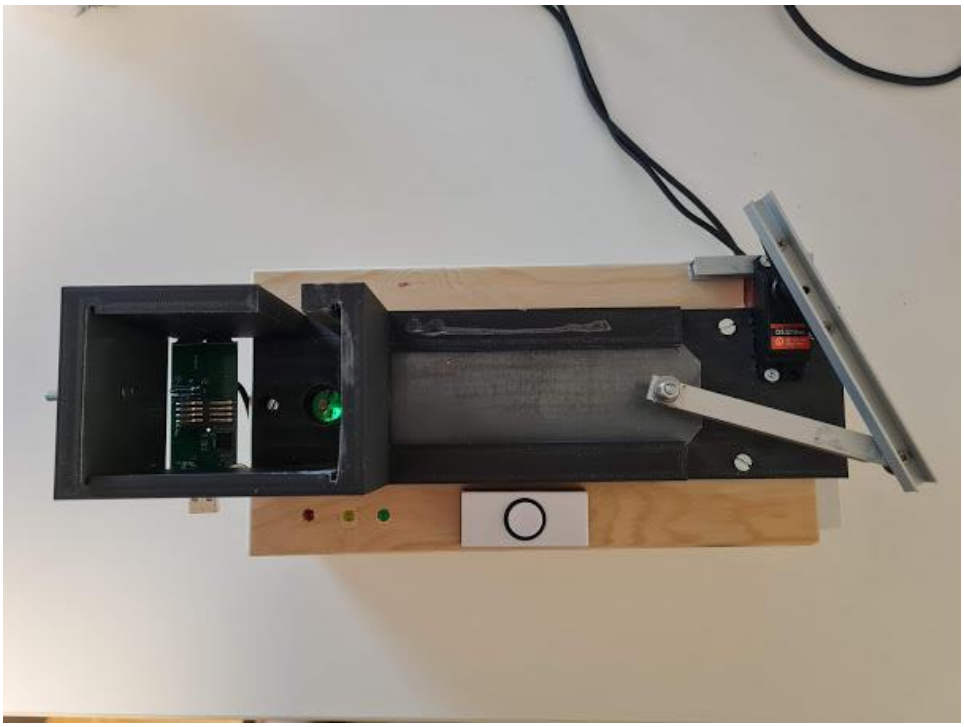


*Abbildung 53 Endprodukt Topview*

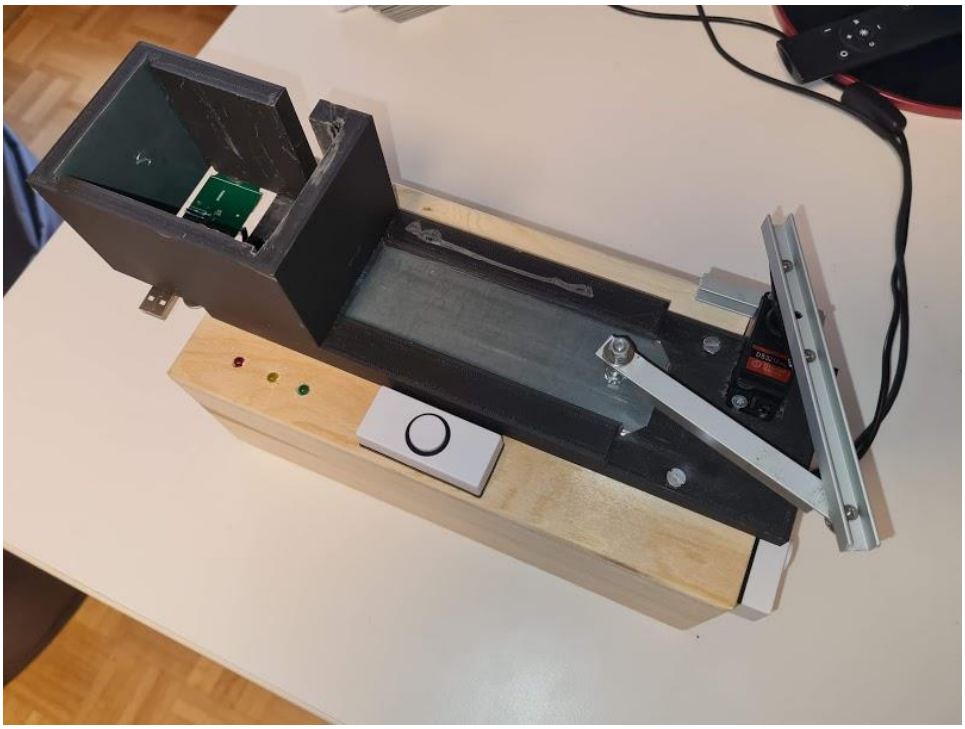


*Abbildung 54 Endprodukt Sideview*

## 12.5. Logbuch

### 12.5.1. Kalenderwoche 36

| Datum | Tätigkeit | Aufwand (soll) [h] | Aufwand (ist) [h] |
|---|---|---|---|
| 01.09.2020 | **Evaluierung/Bestellung Hardware**<br>Es wurde Evaluierung bezüglich folgender Elemente ausgeführt und anschliessend auch direkt bestellt.<br>- 1x Motor<br>- 1x Lichtsensor<br>- 1x Raspberry Pi<br>Folgendes Material wurde direkt bestellt<br>- 1x Jumperkabel<br>- 1x SD-Karte (min 16GB)<br>- Relais<br>- Multiport USB-Ladegerät<br>Zusätzlich wurde eine Liste erstellt für weiter benötigtes Material.<br>- 10xSchrauben M3 mit Unterlagscheiben und Mutter<br>- 1x 0.5 mm Blech ca 20x30 cm<br>- 1x Winkelstange Alu 1x0.5 cm ca. 50 cm lang<br>- 1x Holzkiste ca. 30x15x10 cm<br>- 5x Klemmen für Drähte<br>- 1x Led grün/gelb/rot<br>- 1x 330 Ohm Wiederstand<br>- 2x Taster | 5 | 6 |
| 02.09.2020 | **3D Gehäuse erstellen**<br>Hierbei erstellte ich ein 2D Modell im PowerPoint, welches ich bei Fiverr (Freelancer Plattform) bei einer externen Person in Auftrag gab, um dies in eine 3D Modell Datei umzusetzen.<br>Das Suchen nach der richtigen Person auf Fiverr ging länger als gedacht | 1 | 1.5 |
| Aufwand Total<br>Abweichung | | 6 | 7.5<br>+1.5 |

### 12.5.2. Kalenderwoche 37

| Datum | Tätigkeit | Aufwand (soll) [h] | Aufwand (ist) [h] |
|---|---|---|---|
| 10.09.2020 | **Evaluierung/Bestellung Hardware**<br>Einkauf restlicher Ware im Baumarkt.<br>- 10xSchrauben M3 mit Unterlagscheiben und Mutter<br>- 1x 0.5 mm Blech ca 20x30 cm<br>- 1x Winkelstange Alu 1x0.5 cm ca. 50 cm Lang<br>- 1x Holzkiste ca. 30x15x10 cm<br>- 5x Klemmen für Drähte<br>- 1x Led grün/gelb/rot<br>- 1x 330 Ohm Wiederstand<br>- 2x Taster<br>- Kleinmaterial Schrauben 3mm x 30mm | 4 | 3.5 |
| 11.09.2020 | **3D Gehäuse erstellen** | 1 | 1 |

| | | | |
|---|---|---|---|
| | 3D Datei erhalten. Auftrag zum 3D Druck bei Gaffuri Druckerei ausgelöst | | |
| Aufwand Total<br>Abweichung | | 5 | 4.5<br>-0.5 |

### 12.5.3. Kalenderwoche 38

| Datum | Tätigkeit | Aufwand (soll) [h] | Aufwand (ist) [h] |
|---|---|---|---|
| 18.09.2020 | **Initialisierung**<br>Beginn mit Grundstruktur des Dokumentes | 2 | 3 |
| 18.09 2020 | **Testmappe erstellen**<br>Grundstruktur Testmappe erstellt | 1 | 0.5 |
| 19.09.2020 | **Testmappe erstellen**<br>Test-Cases definiert und beschrieben | 5 | 5 |
| 20.09.2020 | **Projektauftrag erstellen**<br>Projektstrukturplan erstellt<br>Vorgangsliste erstellt<br>Ausgangslage erstellt<br>Vision erstellt<br>Use-Case erstellt<br>Architektur erstellt | 4 | 4 |
| 21.09.2020 | **Projektauftrag erstellen**<br>Risikoanalyse erstellt<br>Projektabgrenzung erstellt | 1 | 1 |
| Aufwand Total<br>Abweichung | | 13 | 12.5<br>-0.5 |

### 12.5.4. Kalenderwoche 39

| Datum | Tätigkeit | Aufwand (soll) [h] | Aufwand (ist) [h] |
|---|---|---|---|
| 21.09.2020 | **Aufbau Hardware**<br>Raspberry Pi mit Rasbian OS aufgesetzt<br>Motor, Taster, LED, Lichtsensor, Relais mit Raspberry Pi verbunden | 4 | 4 |
| 22.09.2020 | **Projektauftrag erstellen**<br>Zielsetzungen geschrieben<br>Projektorganisation erstellt<br>Terminplan/Projekttermine erstellt<br>Wirtschaftlichkeit erstellt<br>Projektbudget erstellt<br>Schnittstellen erstellt<br>Restriktionen erstellt<br>Information und Berichterstattung erstellt<br>Ansprechpartner erstellt<br>Unterschrift erstellt | 4 | 4 |
| 23.09.2020 | **Kundenmeeting**<br>Weiter vorgehen für Kickoffmeeting besprochen<br>Unterzeichnung Projektauftrag (Bestätigung über E-Mail) | 1 | 1 |
| 23.09.2020 | **Kontrolle des Terminplanes**<br>Mit Kunde Terminplan kontrolliert | 0.5 | 0.5 |
| 23.09.2020 | **Aufbau Hardware**<br>Zusammenbau Gehäuse Holzkiste<br>Taster angebracht<br>Raspberry Pi in Holzkiste befestigt | 5 | 5 |

| | | | |
|---|---|---|---|
| | Loch für Servomotor erstellt<br>Loch für Lichtsensor erstellt<br>Einschub und SIM-Karten Auswurf-Begrenzung zugeschnitten | | |
| 24.09.2020 | **Aufbau Hardware**<br>Kolben-Mechanismus zugeschnitten und an Servomotor und Einschub befestigt.<br>Anpassung an 3D Gehäuse gemacht, war zu kurz für SIM-Karte ca. 1 mm abgeschliffen | 4 | 3.5 |
| 25.09.2020 | **Aufbau Hardware**<br>Einbinden SIM-Karte Lesegerät<br>Löten von Modul an Kartenleser | 2 | 2 |
| 26.09.2020 | **Dokumentation bearbeiten**<br>Evaluierung in Dokument eingebunden<br>3D Gehäuse erstellen in Dokument eingebunden | 4 | 4 |
| 27.09.2020 | **Kick-Off Meeting**<br>Präsentation Kick-Off Meeting erstellt | 0.5 | 0.5 |
| Aufwand Total<br>Abweichung | | 25 | 24.5<br>-0.5 |

### 12.5.5. Kalenderwoche 40

| Datum | Tätigkeit | Aufwand (soll) [h] | Aufwand (ist) [h] |
|---|---|---|---|
| 28.09.2020 | **Automatischer Auswurf/Einschub programmieren**<br>Informationen suchen, wie man den Servomotor ansteuern kann<br>Über Python Programm gebaut<br>Klasse Motor mit Methode<br>Run, Stop, Positionscheck | 4 | 4 |
| 28.09.2020 | **Dokumentation bearbeiten**<br>Evaluierung Programmiersprache<br>Informationen zum Einfügen in Diplomarbeit zusammengestellt<br>Automatischer Auswurf/Einschub programmieren | 2 | 2 |
| 29.09.2020 | **Motor Ausschalten, wenn keine SIM-Karte vorhanden**<br>Informationen zu Lichtsensor suchen, um in Python ein Skript zu bauen<br>Skript in Python Programm ergänzt, welches der Auswurf/Einschub programmiert wurde | 5 | 5 |
| 29.09.2020 | **Dokumentation bearbeiten**<br>Informationen zum Einfügen in Diplomarbeit zusammengestellt zu Motor Ausschalten, wenn keine SIM-Karte vorhanden | 2 | 2 |
| 30.09.2020 | **SMS oder E-Mail an Anwender, wenn Stapel abgearbeitet ist**<br>Nach intensiver Recherche wurde eine neue Klasse in Python erstellt (E-Mail), welche folgende Methoden aufweist:<br>withsim():<br>withoutsim():<br>handstop():<br>Bei diesen Methoden werden Emails ausgelöst mit unterschiedlichem Header und Body | 4 | 4 |
| 30.09.2020 | **Kick-Off Meeting** | 1 | 1 |

| | | | |
|---|---|---|---|
| | Das Kickoffmeeting wurde abgehalten | | |
| 30.09.2020 | **Dokumentation bearbeiten**<br>Informationen zum Einfügen in Diplomarbeit zusammengestellt zu SMS oder E-Mail an Anwender, wenn Stapel abgearbeitet ist | 1 | 1 |
| 01.10.2020 | **Per Knopfdruck System starten/stoppen**<br>Starten des Systems per Bashscript ermöglicht<br>Stoppen des Systems per Knopf nach Lösung gesucht. Noch keine Lösung gefunden. Durch dies wurde mehr Zeit verwendet | 2 | 3 |
| 01.10.2020 | **Projektauftrag erstellen**<br>Kleinere Anpassung vorgenommen | 0.5 | 0.5 |
| 02.10.2020 | **Projektauftrag erstellen**<br>Wirtschaftlichkeit, Zeitrechnung und Amortisation angepasst | 2.5 | 2.5 |
| 02.10.2020 | **Dokumentation bearbeiten**<br>Informationen zum Einfügen in Diplomarbeit zusammengestellt (per Knopfdruck System starten/stoppen) | 1 | 1 |
| 02.10.2020 | **Dokumentation bearbeiten**<br>Mehrere Aktivitäts-Diagramme erstellt | 1 | 1 |
| 02.10.2020 | **Testmappe erstellen**<br>Testmappe kontrolliert und kleinere Fehler korrigiert | 0.5 | 0.5 |
| 04.10.2020 | **Per Knopfdruck System starten/stoppen**<br>Lösung für Stopp Knopf gefunden, wurde mit Elif Statement gelöst | 0 | 2 |
| 04.10.2020 | **Dokumentation Bearbeiten**<br>Aktivitätsdiagramm des kompletten Pythonskripts erstellt | 0 | 1 |
| Aufwand Total<br>Abweichung | | 26.5 | 30.5<br>+4 |

### 12.5.6. Kalenderwoche 41

| Datum | Tätigkeit | Aufwand (soll) [h] | Aufwand (ist) [h] |
|---|---|---|---|
| 05.10.2020 | **Automatisch SIM-Karte über PC updaten (Skript auf PC)**<br>Aufgrund fehlender Software seitens SNOW PoS Wurde hier nach einer ergänzenden Test Software gesucht, um das SNOW PoS System zu simulieren | 3 | 3 |
| 06.10.2020 | **Automatisch SIM-Karte über PC updaten (Skript auf PC)**<br>Mock Server in Postman erstellt und dazu gehöriger Post Request | 5 | 5 |
| 07.10.2020 | **Automatisch SIM-Karte über PC updaten (Skript auf PC)**<br>Post Request über Python in Datei Speichern | 4 | 4 |
| 07.10.2020 | **Kundenmeeting**<br>Präsentation von erledigten Arbeiten und Absprache bezüglich Probleme mit Lieferanten | 1 | 1 |
| 07.10.2020 | **Qualitätskontrolle**<br>Aufbau DA kontrolliert. Anpassungen an Dokumentation Systemtechnik gemacht | 1 | 1 |
| 08.10.2020 | **Automatisch SIM-Karte über PC updaten (Skript auf PC)** | 3 | 3 |

| | | | |
|---|---|---|---|
| | Text Datei über Python bei E-Mail als Anhang verschicken | | |
| 09.10.2020 | **Dokumentation bearbeiten**<br>Daten ermittelt, um SNOW PoS Systembeschreibung zu erstellen | 1 | 1 |
| 10.10.2020 | **Dokumentation bearbeiten**<br>Erstellen von SNOW PoS Systembeschreibung | 2 | 2 |
| 11.10.2020 | **Dokumentation bearbeiten**<br>Erstellen von Ausweichziel da SNOW PoS nicht vorhanden ist. | 2 | 2 |
| 11.10.2020 | **Testmappe erstellen**<br>Kontrolle, ob Tests noch mit Projekt übereinstimmen | 0.5 | 0.5 |
| Aufwand Total<br>Abweichung | | 22.5 | 22.5<br>0 |

### 12.5.7. Kalenderwoche 42

| Datum | Tätigkeit | Aufwand (soll) [h] | Aufwand (ist) [h] |
|---|---|---|---|
| 13.10.2020 | **Automatisch SIM-Karte über PC updaten (Skript auf PC)**<br>Text Datei über Python löschen Klasse erstellt<br>Anpassungen in Main Programm gemacht, um SNOW PoS POST Anfragen in Zyklus einzubinden | 3 | 3 |
| 14.10.2020 | **Automatisch SIM-Karte über PC updaten (Skript auf PC)**<br>Vortest bei main Programm gemacht, um bereit für finale Testmappe zu sein | 2 | 0.5 |
| 14.10.2020 | **Kontrolle und Überwachung der Ressourcen**<br>Bisher verbrauchtes Budget kontrolliert stand bei: 404 CHF | 0.5 | 0.5 |
| 15.10.2020 | **Testmappe erstellen**<br>Kontrolle für SNOW PoS Test Cases Anpassungen gemacht | 0.5 | 1 |
| 16.10.2020 | **Tests durchführen**<br>Es wurden alle Tests bis auf die Stress Tests durchgeführt. Kleinere Mängel wurden festgestellt | 3 | 3 |
| 16.10.2020 | **Dokumentation bearbeiten**<br>Test Auswertung dokumentiert | 0.5 | 0.5 |
| 17.10.2020 | **Tests durchführen**<br>Die Stresstests wurden durchgeführt. Auch hier wurden kleinere Mängel festgestellt | 3 | 3 |
| 17.10.2020 | **Tests Auswerten**<br>Nach Beendigung der Tests wurde noch genau untersucht, wieso die entstandenen Fehler aufgetaucht sind. Diese wurden dokumentiert | 2 | 2 |
| 17.10.2020 | **Dokumentation bearbeiten**<br>Test Auswertung dokumentiert | 1 | 2 |
| Aufwand Total<br>Abweichung | | 15.5 | 15.5 |

### 12.5.8. Kalenderwoche 43

| Datum | Tätigkeit | Aufwand (soll) [h] | Aufwand (ist) [h] |
|---|---|---|---|
| 19.10.2020 | **Kundenmeeting**<br>Besprechung von Tests sowie weiteres Vorgehen. | 1 | 1 |
| 19.10.2020 | **Expertenmeeting**<br>Kontrolle Terminplan sowie Fragenklärung | 1 | 1 |
| 20.10.2020 | **Verstopfung des Geräts oder Fehler durch Motor erkennen**<br>Logger in Programm eingebaut | 2 | 2 |
| 20.10.2020 | **Dokumentation bearbeiten**<br>Bedienungsanleitung erstellt | 3 | 3 |
| 21.10.2020 | **Kontrolle des Terminplans**<br>Kontrolle der Stunden sowie des Terminplans | 0.5 | 0.5 |
| 21.10.2020 | **Einbindung beim Kunden**<br>An Netzwerk angebunden<br>Backup erstellt<br>Kunde in ASU eingeführt | 2 | 2 |
| 21.10.2020 | **Verstopfung des Geräts oder Fehler durch Motor erkennen**<br>Fehlererkennung von Motor recherchiert | 2 | 2 |
| 22.10.2020 | **Dokumentation bearbeiten**<br>Anpassungen Evaluierung/Bestellung<br>Anpassungen 3D Gehäuse<br>Aufbau Hardware | 4 | 4 |
| 23.10.2020 | **Dokumentation bearbeiten**<br>Anpassungen Einbindung beim Kunden<br>Anpassung Evaluierung Programmiersprache<br>Anpassung Automatischer Auswurf /Einschub<br>Anpassung Motor Ausschalten, wenn keine SIM-Karte vorhanden<br>Anpassung per Knopfdruck System starten/stoppen | 6 | 8 |
| 24.10.2020 | **Dokumentation bearbeiten**<br>Anpassen automatisch SIM-Karte über PC updaten<br>Dokumentiert Verstopfung des Gerätes oder Fehler durch Motor erkennen<br>Anpassung testen des ASU | 5 | 7 |
| 25.10.2020 | **Dokumentation bearbeiten**<br>Diagramme erstellen<br>Klassen<br>Aktivität<br>Netzwerk | 6 | 7 |
| Aufwand Total<br>Abweichung | | 32.5 | 34.5<br>+5 |

### 12.5.9. Kalenderwoche 44

| Datum | Tätigkeit | Aufwand (soll) [h] | Aufwand (ist) [h] |
|---|---|---|---|
| 26.10.2020 | **Abnahme System**<br>System wurde durch Mathias Hubacher abgenommen | 2 | 2 |
| 26.10.2020 | **Präsentation erstellen**<br>Beginn Aufbau Präsentation | 3 | 3 |
| 27.10.2020 | **Präsentation erstellen**<br>Auswahl Themen | 4 | 4 |
| 27.10.2020 | **Dokumentation bearbeiten**<br>Grammatik Korrektur | 2 | 2 |
| 28.10.2020 | **Dokumentation bearbeiten**<br>Grammatik Korrektur | 2 | 2 |
| 28.10.2020 | **Präsentation erstellen**<br>Anpassung Design und Themen ergänzt | 3 | 3 |
| 28.10.2020 | **Kontrolle und Überwachung der Ressourcen**<br>Benötigte Zeitaufwand und Budget kontrolliert<br>Zeitaufwand in Budget eingetragen | 0.5 | 1 |
| 28.10.2020 | **Kundenmeeting**<br>Überblick auf Projektabschlussmeeting und Dokumentation angeschaut. | 1 | 1 |
| 28.10.2020 | **Qualitätskontrolle**<br>Dokumentation Aufbau kontrolliert | 1 | 1 |
| 29.10.2020 | **Onlinepublikation erstellen**<br>Publikation geschrieben Fotos gemacht | 3 | 4 |
| 29.10.2020 | **Dokumentation bearbeiten**<br>Grammatik Korrektur | 2 | 2 |
| 01.11.2020 | **Upload und Veröffentlichung von Diplomarbeit** | 0 | 0 |
| Aufwand Total<br>Abweichung | | 23.5 | 25<br>+1.5 |