**Projektplanung Smart-Bohner**

Inhalt

[1. Projektbeschreibung 2](#_Toc94788657)

[2. Rollen 2](#_Toc94788658)

[3. Funktionen 2](#_Toc94788659)

[3.1. Grundlegende Funktionen 2](#_Toc94788660)

[3.1.1. Kaffee machen 2](#_Toc94788661)

[3.1.2. Web UI 2](#_Toc94788662)

[3.2. Warnungen 3](#_Toc94788663)

[3.3. Statistiken 3](#_Toc94788664)

[3.4. NFC-Tag 3](#_Toc94788665)

[4. Meilensteine 3](#_Toc94788666)

[5. Technischer Rahmen 3](#_Toc94788667)

[5.1. Versionsverwaltung 3](#_Toc94788668)

[5.2. Projekt-Management 3](#_Toc94788669)

[6. Dokumentation der Teammitglieder 4](#_Toc94788670)

[6.1. Felix Eckl 4](#_Toc94788671)

[6.2. Leon Burtscher 5](#_Toc94788672)

# Projektbeschreibung

Bei diesem Projekt handelt es sich um eine smarte Kaffee-Maschine, welche mit einem Raspberry-Pi angesteuert wird. Der Raspberry-Pi wird direkt an die Kaffee-Maschine angeschlossen und mit einer Web-API angesteuert.

# Rollen

Die Rollen im Team sind folgendermaßen verteilt: Leon Burtscher übernimmt den Scrum-Master, allerdings ist er auch in der Entwicklung eingebunden. Felix Eckl, Michael Widerin und Josua Egger übernehmen die Rolle als Entwickler.

# Funktionen

In diesem Teil wird auf alle Funktionen eingegangen, die in diesem Projekt realisiert werden sollen, spätere Änderungen können nicht ausgeschlossen werden, wenn im Verlauf des Projekts Erkenntnisse gesammelt werden, aus welchen Funktionen ausgeschlossen werden müssen.

## Grundlegende Funktionen

Dieser Teil fokussiert sich auf die Elementaren Funktionen, die in diesem Projekt umgesetzt werden müssen. Ohne diese kann das Projekt nicht fortgeführt werden, da dieser Teil die Elementaren Funktionen dieses Projekts darstellt

### Kaffee machen

Als Benutzer möchte ich per API einen Kaffee machen können, ohne dabei eine Aktion an der Kaffee-Maschine ausführen zu müssen. Dieser wird über die Web-API gemacht und kann auch ohne UI möglich sein. Beim Kaffee machen spielt es anfangs keine Rolle, wie der Befehl an die API gesendet wird, da es in diesem Schritt um die Integration des Raspberry-PIs in die Kaffee-Maschine geht. Ein anfänglicher Schritt wäre es beispielsweiße das Absenden eines POST-Befehls über Postman an die API: Bei Erhalt des Befehls sendet der Webserver den Befehl an die Pins sendet und den Knopfdruck an der Kaffee-Maschine „simuliert“.

### Web UI

Als Benutzer möchte ich eine UI zur Verfügung haben, um die jeweiligen Befehle an die Web-API senden zu können. Die UI wird über Blazor realisiert, um eine einfache Anbindung an die API zu gewährleisten.

## Warnungen

Bei der Kaffee-Maschine können diverse Warnungen und Fehler auftreten, welche am besten von der WEB-API ausgegeben werden sollten.

Als Benutzer möchte ich über bestimmte Events in der Web-Page informiert werden, um immer auf dem neuesten Stand sein. Wenn diese Meldung auftritt, kann der Benutzer keinen „Kaffee kochen“-Befehl mehr an die API senden. Wenn die Meldung wieder erfüllt wurde, dann verschwindet die Meldung und der Benutzer kann wieder Kaffee kochen. Folgende Meldungen sind dabei relevant:

* Wassertank füllen
* Kaffeesatzbehälter leeren

Neben diesen Meldungen, ohne welche der Benutzer keinen Kaffee kochen können, gibt es noch Meldungen, bei welchen es sich eher um „Warnungen“ handelt, da der Benutzer weiter Kaffee kochen kann, allerdings evtl. eingeschränkt.

* Geräte entkalken
* Gerät reinigen

## Statistiken

Als Benutzer möchte ich über meine Statistiken Bescheid wissen. Ich möchte Statistiken über meinen täglichen Kaffee-Konsum erhalten. Diese Statistiken sind in der WEB-UI einsehbar

## NFC-Tag

Als Benutzer möchte ich mich mit einem NFC-Tag authentifizieren können, um mich nicht immer anmelden zu müssen

# Meilensteine

Alle Meilensteine des Projekts sind auf Github dokumentiert

# Technischer Rahmen

## Versionsverwaltung

Bei dem Projekt wird Git als Versionsverwaltung gewählt und auf der Plattform Github verwaltet. Dabei wird jeder Commit/Pull-Request auf den „develop“ Branch mit einem Code-Review überprüft, um die Code-Sicherheit sicherstellt.

## Projekt-Management

Das Projekt wird größtenteils über Github verwaltet. Auf Github gibt es die Möglichkeit, Tickets (Issues) anzulegen, darüber wird in diesem Projekt alle Fehler und Aufgaben verwaltet.

# Dokumentation der Teammitglieder

## Felix Eckl

Aufgabe: Control-Unit und Web-API

Meine Aufgabe war primär die Umsetzung der Control-Unit. Dieser Teil des Programmes ist für die Kommunikation mit den GPIOs auf dem Raspberry Pis zuständig. Dafür habe ich als erstes eine passende Library suchen müssen, mit der die Pins angesteuert werden können. Dabei habe ich mich für Microsofts „System.Device.Gpio“ NuGet-Packet entschieden. Als erstes wurde – nach dem Kauf des Raspberry PI Zero 2W für den wir uns entschieden haben – das Framework getestet und geprüft, ob dieses unseren Anforderungen entspricht, was es letztendlich auch tat. Anschließend wurden aus den Erkenntnissen zu den benötigten Funktionen die Interfaces erstellt und Implementierungen erstellt. Es wurden dann Messungen durchgeführt, um die ungefähren Button-Clicks zu kennen, die dann vom Board simuliert werden müssen. Dies wurde anschließend implementiert und mit dem Volt-Meter getestet. Da bis zur Verbindung von Board und Kaffeemaschine gewartet werden musste wurde in der Zwischenzeit die Web-Page mit Docker ausgeliefert und die Web-API verfeinert. Außerdem habe ich aufgrund der Abwesenheit von Michael die UI an die API angebunden.

## Leon Burtscher

Aufgabe: Hardware Kaffeemaschine

* + 1. Erweiterte Hardware

Wir haben am ersten Tag gleich eine Kaffee-Maschine gekauft. Es ist das Model „Philips HD8830“

Nachdem wir sie zuerst auseinander gebaut haben, um zu sehen was alles zu machen ist, und wo wir was verstauen können, diskutierten wir darüber was wir alles benötigen und welche Teile, Extras, Sinn machen würden.

Wir sind zu dem Schluss gekommen, dass wir ein Display, ein Raspberry Zero 2 W, 2 zwischen Platinen, einen Transformator und die dazugehörigen Kabel benötigen.

* + 1. Platinen vorbereiten

Im ersten Schritt mussten wir die bereits vorhandenen Platinen vorbereiten. Zuerst nahmen wir uns die Platine mit den Steuerungsknöpfen und den Fehler-LEDs vor. Wir löteten die Knöpfe ab und machten uns mit der Platine im Allgemeinen bekannt. Leider hatten wir beim Start ein paar Probleme und so rissen wir aus Versehen ein paar Anschlüsse aus. Dies war später jedoch kein Problem, da wir den Leitungen folgten und die Kabel an einer früheren/späteren Stelle befestigten.

Bei der hinteren (Haupt-) Platine mussten wir die Stromverbindung auftrennen und mit WAGO Klemmen uns Strom für den Raspberry abzweigen.

* + 1. Ätzen der zwischen Platinen

Wir mussten für unsere Kaffee-Maschine zwei zwischen Platinen ätzen. Die erste für den Raspberry und die zweite für den Transformator. Die Platinen wurden von unserem Lehrer gezeichnet (wobei die erste Komplizierter war als die zweite). Danach wurden zwei jungfräuliche Platinen ausgesucht und mit der CNC-Maschine die benötigten Steckplätze gebohrt. Mit UV-Lampen wurde die fotobeschichtete Platine belichtet, ca. 2-4 Minuten. Folglich wird die Platine in ein Bad aus 0,5 Liter lauwarmen Wassers und 10g Natriumhydroxid gelegt und etwas geschwenkt. Die Platine wurde dann in einer kleinen Maschine ca. 2 Minuten mit 40 Grad heißen Eisen 3 Chlorid gewaschen, darauf noch einmal belichtet und auch noch einmal im Bad gewaschen. Die nun fertige Platine wurde darauf getrocknet und mit Löt-Lack besprüht.

* + 1. Anlöten der Kabel

Die Platine mit den Steuerknöpfen musste nun mit der ersten Zwischenplatine verbunden werden (diese kann dann einfach mit dem Raspberry zusammengesteckt werden). Somit löteten wir Kabel an die Steuerknöpfe und verbanden sie mit dem richtigen, zuerst geplanten, Steckknopf der zwischen Platine. Nicht zu vergessen war hier das Kabel für den Ground.

Ebenso wurden die Kabel für den NFC-Reader mit gelötet und der Reader auch gleich an der Kaffee-Maschine angemacht.

* + 1. Stromversorgung

Wie schon weiter oben beschrieben müssten wir bei der Haupt-Platine die Stromversorgung auftrennen und WAGO Klemmen angebracht. Somit konnten wir uns den Strom für den Raspberry abzweigen. Der Transformator wurde auf die zweite Zwischen-Platine gelötet und dann in Folge mit Kabeln an den WAGO Klemmen befestigt. Hinter dem Transformator hatten wir nun 5V und ca. 2 Ampere, mit welchen wir den Raspberry versorgen konnten. Diese Stromkabel wurden nun nur noch zum Raspberry gelegt und dort auf die 5V Input Pins gelötet.

* + 1. UI Panel drucken

Mit dem Programm Autodesk Fusion 360 haben wir das UI-Panel nachgebaut und dann das Display mit hinein gebaut. Die Schwierigkeit hierbei war das Display in der Richtigen Größe zu konstruieren und das Display in einem guten/brauchbaren Winkel hineinzubekommen. Dies druckten wir dann mit dem 3D Drucker der Schule aus und schliffen es ab.

## Josua Egger

Aufgabe: UI

Ich trat dem Smart Bohner Projekt in der zweiten Schulwoche bei und übernahm zusammen mit Michael Widerin die Gestaltung der Benutzeroberfläche. Wir erarbeiteten MockUp-Designs für die Weboberfläche und den Touchbildschirm der Kaffeemaschine. Michael übernahm den Touchbildschirm und ich startete mit dem Designen des WebUI.

Durch meine fehlende Erfahrung im Bereich „html“ beschäftigte ich mich mit dem Thema im Allgemeinen. Auch im weiteren Entwicklungsprozess konnte ich immer wieder auf die Erfahrung von Michael zurückgreifen.

Ich teilte den Bildschirm drei Abschnitte in dem jeweils bestimmte Abschnitte des UserInterfaces angezeigt werden sollten. Für die Buttons von „Kaffee“, „Espresso“ und „Espresso Lungo“ benutzten wir ein einheitliches Buttondesign. Da die Anforderungen für die WebOberfläche und die des Touchdisplays unterschiedlich sind, überlegten wir uns passende Lösungen für die jeweilige Oberfläche. Nach einigen Designanpassungen wurde dann die Buttons funktionstüchtig gemacht.

Das Aussehen der Buttons wurde in eigenen css-Ressourcen codiert. Die im Rootverzeichnis gespeichert wurden.

Nach einem anfänglichen guten Start, wurden Style Änderungen aus der Ressourcendatei nicht mehr übernommen und nach Rücksprache mit Kollege Widerin startete ich meine Webpage nicht mehr im Debug-Modus und die Änderungen wurden erfolgreich übernommen.

Durch das Ändern des Programmstartpunktes von Mobile zu Server konnte das Programm meine erstellten Ressourcendateien nicht mehr verwenden. Nach eigenen Versuchen die Buttons wieder über die Ressourcen im Design zu ändern, entschied ich mich neue Ressourcen für das Buttondesign anzulegen.

Bei der Implementierung des Banners für die Notifikationen der Kaffeemaschine war es mir zeitlich dann leider nicht mehr möglich die Funktionstüchtigkeit herzustellen. Der Code und das Design stellen bereit und müssen nur so eingebunden werden dass, das Banner bei dem erhalten einer Meldung die Meldung anzeigt.

## Michael Widerin

Aufgaben: UI und Web-API

### Aufgabenbeschreibung

Meine Aufgaben lagen im Bereich der Webentwicklung. Sie umfassten den grundlegenden Aufbau und Dokumentation der Web API, sowie das Entwerfen und Entwickeln einer UI für das Display der Kaffeemaschine.

### blauf und Schwierigkeiten

Die ersten Schritte meiner Tätigkeiten waren das Einbinden und Konfigurieren der Swagger-Dokumentation für unsere Web API. Nachdem diese Aufgaben erledigt waren, mussten, gemeinsam mit Felix, alle benötigten Schnittstellen der Steuereinheit definiert werden. Anhand der definierten Schnittstellen wurden die API-Controller für Wartungsarbeiten und den alltäglichen Gebrauch der Kaffeemaschine erstellt.

Die grundlegenden Aufgaben der Web API wurden relativ schnell erledigt. Dabei entstanden eigentlich keine richtigen Schwierigkeiten.

Der Entwurf für das UI des Displays wurde danach, gemeinsam mit Josua, ausgearbeitet. Diese Aufgabe hat recht viel Zeit in Anspruch genommen, da wir beide keine richtigen Erfahrungen damit hatten. Nachdem der Entwurf fertig war, entschieden wir uns dazu, die Applikationen zu trennen, in Display (Michael) und Desktop, Mobile, etc. (Josua).

Das Entwickeln der Applikation hat ebenfalls viel Zeit in Anspruch genommen, trotz Erfahrung in der Webentwicklung. Die Schwierigkeiten waren dabei weniger das Entwickeln der einzelnen Komponenten, sondern viel mehr der Umgang mit der Entwicklungsumgebung und Blazor, da oft unerwartete Fehler entstanden, welche viel Zeit in Anspruch nahmen.

Eine weitere Aufgabe war dann das Einbinden von SignalR, welches für den Echtzeitaustausch von Warnungen und Statusmeldung zwischen Clients und Server benötigt wurde. Die Einbindung hat einwandfrei funktioniert.