Wetterstation

Projekthandbuch und -dokumentation

Ing. Hannes Aurednik

Eva Gergely

Salko Nuhanovic

Stephan Nöhrer

im Zuge der Lehrveranstaltung Embedded Systems Engineering

im Sommersemester 2018

betreut durch: Patrick Schmitt MSc

Einleitung

In folgendem Dokument wird das Projekt „Wetterstation“, welches im Zuge der Lehrveranstaltung Embedded Systems Engineering entwickelt wurde detailliert beschrieben und dokumentiert.

Statt einer persönlichen Lerndokumentation wurde in diesem Dokument versucht, eine Projektschilderung lege artis anzufertigen, die einerseits die Anforderungen der Lehrveranstaltung, andererseits aber auch den Interessen eventueller, zukünftiger Stakeholder genügt.

Zusätzlich wurde von den Verfassern Augenmerk auf die Verhältnismäßigkeit der Dokumentation zum eigentlichen Projekt gelegt.

# Inhaltsverzeichnis

[Inhaltsverzeichnis 3](#_Toc518260530)

[Projekthandbuch 5](#_Toc518260531)

[1. Projektauftrag 5](#_Toc518260532)

[2. Projektzieleplan 6](#_Toc518260533)

[3. Changerequests 7](#_Toc518260534)

[Change 1 7](#_Toc518260535)

[4. Projektstrukturplan 8](#_Toc518260536)

[5. Arbeitspaketspezifikationen 9](#_Toc518260537)

[6. Projektmeilensteinplan 14](#_Toc518260538)

[7. Projektfunktionendiagramm 15](#_Toc518260539)

[8. Projektterminliste 16](#_Toc518260540)

[9. Projektregeln und Projektwerte 17](#_Toc518260541)

[10. Projektabschlussbericht 18](#_Toc518260542)

[Projektdokumentation 19](#_Toc518260543)

[1. Einleitung 19](#_Toc518260544)

[2. AP: Projektkommunikation 19](#_Toc518260545)

[Besonderheiten Abschlusspräsentation 20](#_Toc518260546)

[3. AP: Frontend Design 22](#_Toc518260547)

[4. AP: Architektur 23](#_Toc518260548)

[5. AP: Netzwerk 23](#_Toc518260549)

[Verwendete Tools 24](#_Toc518260550)

[6. AP: Sensorik 25](#_Toc518260551)

[Verwendete Tools 26](#_Toc518260552)

[Vorgehensweise 27](#_Toc518260553)

[7. AP: Backend 33](#_Toc518260554)

[8. AP: Alexa Skill 34](#_Toc518260555)

[9. AP: Frontend Implementierung 34](#_Toc518260556)

[10. AP: Embedded Integration 35](#_Toc518260557)

[Analyse und Bewertung Integrationsszenarien 35](#_Toc518260558)

[Integration mittels Mailboxen 36](#_Toc518260559)

[Aufgetretene Herausforderungen 38](#_Toc518260560)

[Refactoring-Maßnahmen 38](#_Toc518260561)

[11. Installationsanleitung 40](#_Toc518260562)

[12. Lessons Learned 40](#_Toc518260563)

[Anhang 42](#_Toc518260564)

[13. Verwendete Tools 42](#_Toc518260565)

[14. Abbildungsverzeichnis 43](#_Toc518260566)

[15. Recherchelisten 44](#_Toc518260567)

[16. Quellenverzeichnis 45](#_Toc518260568)

[Designvorlage 46](#_Toc518260569)

[Überschrift 1 46](#_Toc518260570)

[1. Überschrift 2 46](#_Toc518260571)

[Überschrift 3 46](#_Toc518260572)

# Projekthandbuch

|  |  |
| --- | --- |
| Projektauftrag | |
| **Starttermin:** | **Projektendtermin:** |
| 19.02.2018 | 02.07.2018 |
| **Projektziele** | **Nicht-Projektziele** |
| Entwicklung eines Smarthome-Assistent-Systems zur Temperatur und Luftfeuchtigkeitsinformation inkl. Userinteraktion über Amazon Echo. |  |
| **Projektauftraggeberteam** | **Projektmanager** |
| Patrick Schmitt MSc | Ing. Hannes Aurednik |
| **Projektteammitglieder** | |
| Ing. Hannes Aurednik  Eva Gergely  Salko Nuhanovic  Stephan Nöhrer | |

|  |  |
| --- | --- |
| Projektzieleplan | |
| **Hauptziele** | **Adaptiert per …** |
| Entwicklung eines Amazon Echo Alexa Skills zur Userinteraktion | 02.07.2018 |
| Implementieren eines Webservers zur Datenanzeige, -speicherung und Bereitstellung von Schnittstellen | 02.07.2018 |
| Implementieren einer per Netzwerk erreichbaren Messstation (kabelgebunden) | 02.07.2018 |
| Unterstützung mehrerer Messstationen | 02.07.2018 |
|  |  |
| **Zusatzziele (optional)** | **Adaptiert per …** |
| Entwickeln einer Dashboard Funktionalität mit Visualisierung des Werteverlaufes der Messdaten | 02.07.2018 |
| Implementieren einer per Netzwerk erreichbaren Messstation (WLAN) | 02.07.2018 |
|  |  |
|  |  |
|  |  |
|  |  |
| **Nicht-Ziele** | **Adaptiert per …** |
| Produktreife „Plug and Play – Lösung“ |  |
|  |  |
|  |  |
|  |  |

Annahmen und Interpretation:

|  |  |
| --- | --- |
| Changerequests | |
| **Änderung** | **Adaptiert per …** |
| Change 1 | 02.07.2018 |
|  |  |
|  |  |

|  |
| --- |
| Change 1 |
| **Ursprüngliches Ziel** |
|  |
| **Adaptiertes Ziel** |
|  |
| **Begründung** |
|  |

|  |
| --- |
| Projektstrukturplan |
| [[ PSP kann an dieser Stelle eingefügt werden ]](" \l "Text166" \o "EINFÜGEN - OBJEKT klicken. Dann die Karteikarte \"Aus Datei erstellen\" auswählen und den Pfad der einzufügenden Datei angeben.) |

Annahmen und Interpretation:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Arbeitspaketspezifikationen | | | | |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Lösungsarchitektur** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
| * Entwicklung einer zweckmäßigen Lösungsarchitektur | | | |  |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
| * Anforderungsgerechte Lösungsarchitektur | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
| * Vollständigkeit Lösungsarchitektur * Konformität Lösungsarchtiektur | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Projektkommunikation** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
| * Regelmäßige Kommunikation Projektergebnisse * Planung Abschlusspräsentation * Kommunikation Projektrisiken | | | |  |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
| * Durchführung regelmäßiger Projektkommunikation zum Auftraggeber * Zeitgerechte Kommunikation abnahmeverhindernder Projektrisiken | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Embedded - Netzwerk** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
| * Implementieren einer architekturkonformen Netzwerkanbindung * Unterstützung TCP basierter Kommunikation * Unterstützung von kabelgebundenen Netzwerkverbindungen | | | |  |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
| * Unterstützung UDP basierter Kommunikation * Unterstützung von drahtlosen Netzwerkverbindungen | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
| * Funktion der implementierten Netzwerkanbindung * Funktion der IPC Anbindung | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Sensorik** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
| * Implementieren der Sensorikfunktionalität * Implementieren einer Ausgabe auf der Konsole zwecks Fehlerüberprüfung * Mockups erstellen | | | | 17.05.2018  18.03.2018 |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
| * Implementieren von Mailboxes * Implementieren von CRC | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
| * Sensorik funktionsfähig * Ausgaben auf der Konsole durch bedingtes Kompilieren abgrenzen | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
|  | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Backend** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
| * Datenpersistenz im definierten Umfang * Implementieren der Serverschnittstelle zum Empfang der Sensordaten * Bereitstellung der für das Frontend notwendigen Funktionalitäten | | | |  |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
| * Funktionierende Datenpersistenz * Funktion der Serverschnittstelle * Funktion der vom Frontend benötigten Funktionalitäten | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Frontend Implementierung** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
|  | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Alexa Skill** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
|  | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Dokumentation** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
|  | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Frontend Design** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
|  | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PSP-Code:** |  | **AP-Bezeichnung** | **Embedded Integration** | |
| **AP-Inhalt** | | | | **Adaptiert per …** |
| * Kommunikation der beiden embedded Teile sicherstellen | | | |  |
| **AP-Nicht-Inhalte** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Ergebnisse** | | | | **Adaptiert per …** |
|  | | | |  |
| **AP-Leistungsfortschrittsmessung** | | | | **Adaptiert per …** |
|  | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Projektmeilensteinplan | | | | |
| **PSP-Code** | **Meilensteinbezeichnung** | **Plantermin** | **Plantermin adaptiert per …** | **Isttermin** |
|  | Projektplanung abgeschlossen |  |  |  |
|  | Recherche abgeschlossen |  |  |  |
|  | NW Kommunikation möglich |  |  |  |
|  | NW Stack fertiggestellt |  |  |  |
|  | HTU Sensorik fertiggestellt |  |  |  |
|  | Backend fertiggestellt |  |  |  |
|  | Frontend fertiggestellt |  |  |  |
|  | Alexa Skill fertiggestellt |  |  |  |
|  | Embedded Integration fertiggestellt |  |  |  |
|  | Prototyp präsentierbar |  |  |  |
|  | Dokumentation abgeschlossen | 01.07.2018 |  |  |
|  | Projekt abgenommen | 02.07.2018 |  | 02.07.2018 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Projektfunktionendiagramm | | | | | | | |
| **PSP-Code** | **AP-Bezeichnung** | **Rollen und Umwelten** | | | | | |
| **Projektauftrag-geberteam** | **Aurednik** | **Gergely** | **Nuhanovic** | **Nöhrer** | **Sonstige** |
|  | Projektkommunikation | **I** | **D** | **Z** | **B** | **Z** |  |
|  | Frontend Design | **I** | **Z** | **D** | **Z** | **B** |  |
|  | Architektur | **I** | **Z** | **Z** | **D** | **B** |  |
|  | Netzwerk | **I** | **D** | **Z** | **B** | **B** |  |
|  | Sensorik | **I** | **B** | **D** | **Z** | **Z** |  |
|  | Backend | **I** | **Z** | **ZI** | **B** | **D** |  |
|  | Dokumentation | **I** | **D** | **D** | **D** | **D** |  |
|  | Alexa | **I** | **Z** | **Z** | **D** | **Z** |  |
|  | Frontend Implementierung | **I** | **Z** | **B** | **Z** | **D** |  |
|  | Embedded Integration |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Nach IBZED-Schema:

* I - Information:
* B - Beratung:
* Z - Zustimmung
* E - Entscheidung
* D - Durchführung:

Annahmen und Interpretation:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Projektterminliste | | | | | | | |
| **Arbeitspaket** | | **Anfangstermin** | | | **Endtermin** | | |
| **PSP-Code** | **Bezeichnung** | **Plan** | **aktueller Plan** | **Ist** | **Plan** | **aktueller Plan** | **Ist** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Annahmen und Interpretation:

|  |  |
| --- | --- |
| Projektregeln und Projektwerte | |
| **Projektregeln** | **Adaptiert per …** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **Projektwerte** | **Adaptiert per …** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Annahmen und Interpretation:

|  |
| --- |
| Projektabschlussbericht |
| **Gesamteindruck: Projekt** |
|  |
| **Reflexion: Projektziele** |
|  |
| **Reflexion: Projektleistungsfortschritt** |
|  |
| **Reflexion: Projekttermine** |
|  |
| **Reflexion: Projektkosten, Projektressourcen** |
|  |
| **Reflexion: Projektumwelten, Beziehungen zu anderen Projekten** |
|  |
| **Reflexion: Projektorganisation und Projektkultur** |
|  |
| **Zusammenfassende Erfahrungen** |
|  |

# Projektdokumentation

|  |  |
| --- | --- |
| Einleitung | |
|  | Im Zuge des Projektes Wetterstation wurde von den beteiligten StudentInnen eine Lösung erarbeitet die in diesem Abschnitt genauer beschrieben wird.  Der Abschnitt Projektdokumentation dient insbesondere dazu, die technische Lösung zu beschreiben, eine Installationsanleitung bereitzustellen und die persönlichen Lernergebnisse der Projektmitglieder zu dokumentieren. |

|  |  |
| --- | --- |
| AP: Projektkommunikation | |
|  | Das Arbeitspaket Projektkommunikation umfasste alle Tätigkeiten die eine effiziente Kommunikation innerhalb des Vorhabens sowie durch das Projekt nach außen gewährleisten.  Insbesondere galt es hier die folgenden Punkte sicherzustellen:   * Regelmäßige Statusupdates an den Projektauftraggeber * Koordination und Planung der internen Jours fixes * Sicherstellen des Informationsflusses im Projektteam * Planung und Vorbereitung der Abschlusspräsentation   Während für die ersten drei Punkte obiger Liste, aufgrund der geringen Anzahl an Projektmitgliedern, keine besonderen und erwähnenswerten Maßnahmen, wie beispielsweise ein expliziter Kommunikationsplan, zu setzen waren, erforderte die Abschlusspräsentation besonderes Augenmerk.  Dies ist primär durch die besondere netzwerktechnische Situation in den Räumlichkeiten des FH-Technikums induziert.  Anders als in einem Heimnetzwerk, für welches das Projekt konzipiert wurde, ist der Netzwerkzugang in der Fachhochschule strenger geregelt und die Mikrocontroller erhalten keine IP-Adressen.  Auch das direkte Verbinden eines handelsüblichen WLAN-Routers mit dem Netzwerk der FH, um auf diesem Wege eine Internetverbindung für die verwendeten Geräte zu erhalten, ist aus selbigem Grund nicht möglich.  Da eine funktionierende Netzwerkverbindung für die Sensorik-Komponenten jedoch von essentieller Bedeutung ist musste eine Alternativlösung gefunden werden. |

|  |  |
| --- | --- |
| Besonderheiten Abschlusspräsentation | |
|  | Um den Besonderheiten der Präsentationsumgebung Rechnung zu tragen wurden für die abschließende Vorführung mehrere Strategien entwickelt. Mangel an Temperaturunterschieden Um trotz der, aus dem Blickwinkel des Raumklimas, homogenen Umgebung der Fachhochschule unterschiedliche Temperatur- und Luftfeuchtigkeitswerte erzeugen und darstellen zu können ohne die zur Verfügung gestellte Hardware zu gefährden, wird im Zuge der Abschlusspräsentation ein Mikroklima geschaffen.  Dieses wird mittels handelsüblicher Kühlkörper in einer Kühltasche für den Medikamententransport erzeugt. Das verwendete Behältnis wurde gewählt da es ein weit geringes Volumen als vergleichbare Produkte aufweist und somit in einer kürzeren Zeitspanne die benötigten Umgebungsparameter erzeugt werden können. Netzwerk Eine größere Herausforderung stellten die eingeschränkten Möglichkeiten der Netzwerkumgebung dar.  Im Zuge mehrerer Projektsitzungen konnte aber, durch eine Erhebung der im Kreis der Projektmitglieder vorhandenen Hardwareressourcen eine Lösung entwickelt werden, die keine zusätzlichen Kosten verursacht und sämtliche Anforderungen für eine Live-Demonstration erfüllt, ohne ein Sicherheitsrisiko für das Netzwerk der Fachhochschule zu erzeugen.  Durch die Kombination eines UTMS- Routers und eines gebräuchlichen WLAN-Access Points mit Switch-Funktionalität auf mehreren RJ45 Ports konnte die folgende Netzwerktopologie erzeugt werden. |



Abbildung 1: Netzwerktopologie Abschlusspräsentation

|  |  |
| --- | --- |
|  | Obige Abbildung zeigt die Netzwerktopologie für die Abschlusspräsentation. Durch den UMTS Router wird dem dahinterliegenden Access Point eine Internetverbindung zur Verfügung gestellt die dieser seinerseits per LAN Schnittstellen an die beiden Sensoren und per WLAN an das Amazon Echo Device weiterreicht.  Der Server ist über das Internet erreichbar auf Microsoft Azure deployed.  Benutzerdevices wie Notebooks oder Tablets können sich über eine aktive Internetverbindung mit dem Server verbinden und die aktuellen Daten anzeigen sowie Sensoren konfigurieren.  Das Amazon Echo Device greift über die per WLAN und UMTS Router zur Verfügung gestellte Internetverbindung auf den Server zu. |

|  |  |
| --- | --- |
| AP: Frontend Design | |
|  | Im Zuge des Projektes Wetterstation wurde von den beteiligten StudentInnen eine Lösung geschaffen die in diesem Abschnitt genauer beschrieben wird.  Der Abschnitt Projektdokumentation dient einerseits dazu, die technische Lösung zu beschreiben, eine Installationsanleitung bereitzustellen und die persönlichen Lernergebnisse der Projektmitglieder zu dokumentieren. |

|  |  |
| --- | --- |
| AP: Architektur | |
|  | Der folgende Abschnitt befasst sich mit der Gesamtlösungsarchitektur der einzelnen Module und Komponenten. |

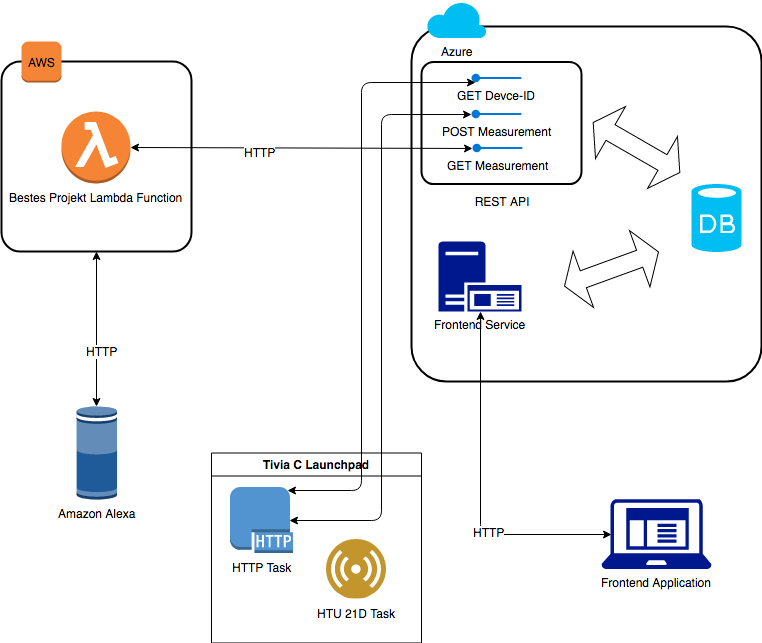


Abbildung 2: Architektur

|  |  |
| --- | --- |
|  | Die Architektur unseres Projekts setzt sich aus den folgenden Komponenten zusammen:   * Alexa Skill * Tivia C Launchpad (inkl. HTU 21D Click Modul) * Backend Application (inkl. Web Dashboard)   **Backend**  Mittelpunkt des Projekts ist die Backendapplikation, welche Alexa Skill, Applikation Server (Web Applikation) und µController verbindet und auf Azure deployed ist.  Bestandteile der Backendapplikation sind:   * Datenbank:   + Enthält aktuelle Messungen und dazugehörigen registrierten Geräte * Applikation Server   + REST API - stellt folgende Endpoints zur Verfügung:     - Vergabe von IDs für Wetterstationen     - Annahme von Messwerten     - Liste der registrierten Wetterstationen     - Auskunft über aktuelle Messwerte   + Dashboard – ermöglicht es dem User Messwerte einzusehen und Wetterstationen zu registrieren   Die REST API ist ein Webservice und ist somit via HTTP aufrufbar.  **Alexa Skill**  Der Alexa Skill besteht aus einem Interaction Model und aus einer Lambda Funktion welche auf AWS deployed sind.  Das Interaction Model ist das Interface für den User welches auf bestimmte Schlagwörter hört und entsprechend Logik durchführt.  Lambda ist ein eine Plattform welche FaaS (Function as a Service) anbietet, somit kann einfach ein Code Snippet nach Lambda deployed werden und dann remote ausgeführt werden.  **µController**  Der Micro Controller hat einen HTU21D Temperatur und Luftfeuchtigkeit Sensor welche periodisch eine Messung durchführt.  **Zusammenspiel der einzelnen Komponenten**  Grundlegend kann man sagen das alle Komponenten der Architektur via HTTP miteinander kommunizieren.  Damit eine Wetterstation (Tivia C Board) Messungen an die Backendapplikation senden kann, muss dieses zuerst registriert werden, dies passiert über das Web Dashboard.  Daraufhin kann der µController Messungen seiner Umwelt durchführen und diese an die Backendapplikation via HTTP POST übergeben. In Azure angelangt werden die Messungen in der Datenbank gespeichert. Die Messungen sind dann über zwei Kanäle aufrufbar:   * Dashboard * Alexa Skill   Der Alexa Skill bekommt seine Daten indem er die Lambda Funktion ausführt, diese ruft dann die Backendapplikation via HTTP GET auf und bekommt die Messungen. Anschließend verarbeitet Alexa das Ergebnis und liest es vor. Further Work Da zum Schluss noch ein bisschen Zeit übriggeblieben ist, wurde überlegt wie die Architektur verbessert werden kann. Nach Recherche hat sich ergeben das man einen MQTT Broker im Backend aufnehmen könnte um die Messungen entgegen zu nehmen, dieser würde dann die HTTP Kommunikation zwischen Wetterstation und Backend ersetzen.  Dies könnte durchgeführt werden indem man einen MQTT Client in der Wetterstation einbaut, welche dann einfach Messungen in den Broker publised. Der Consumer auf der anderen Seite kann dann diese Daten auswerden. Lessons Learned: MQTT hat folgende Vorteile gegenüber HTTP:   * Schneller * Mehr Durchsatz * Weniger Energieverbrauch * Weniger Bandbreite |

|  |  |
| --- | --- |
| AP: Netzwerk | |
|  | Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat   * Effizientes Projektmanagement durch standardisierte Prozesse * Reibungslose Abwicklung von Projekten durch klare Verantwortlichkeiten * Konzernweite einheitliche Projektmanagement Begriffe und Vorgehensweisen   + Standardisiertes Reporting über den Status der Zielerreichung   + Effiziente Lenkung von knappen Ressourcen   Abbildung 1: Organisation Stab QS 12/2016 |

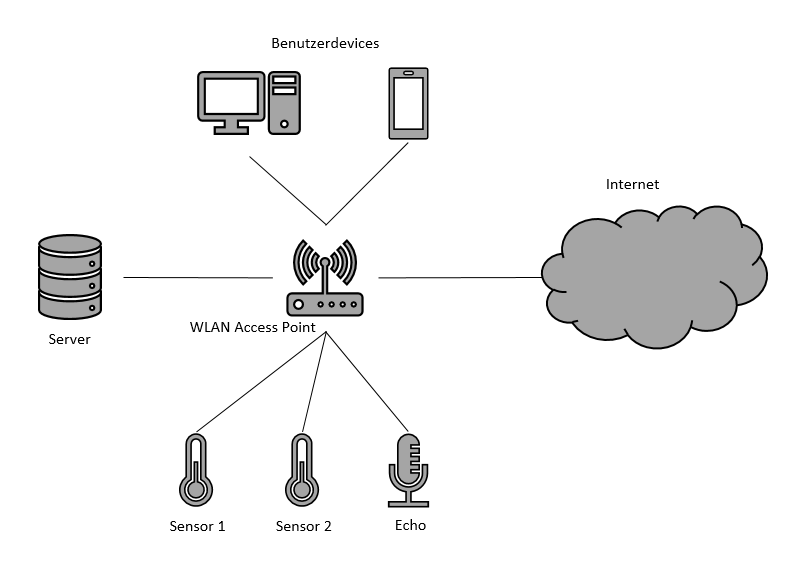


Abbildung 4: Netzwerktopologie Betriebszenario

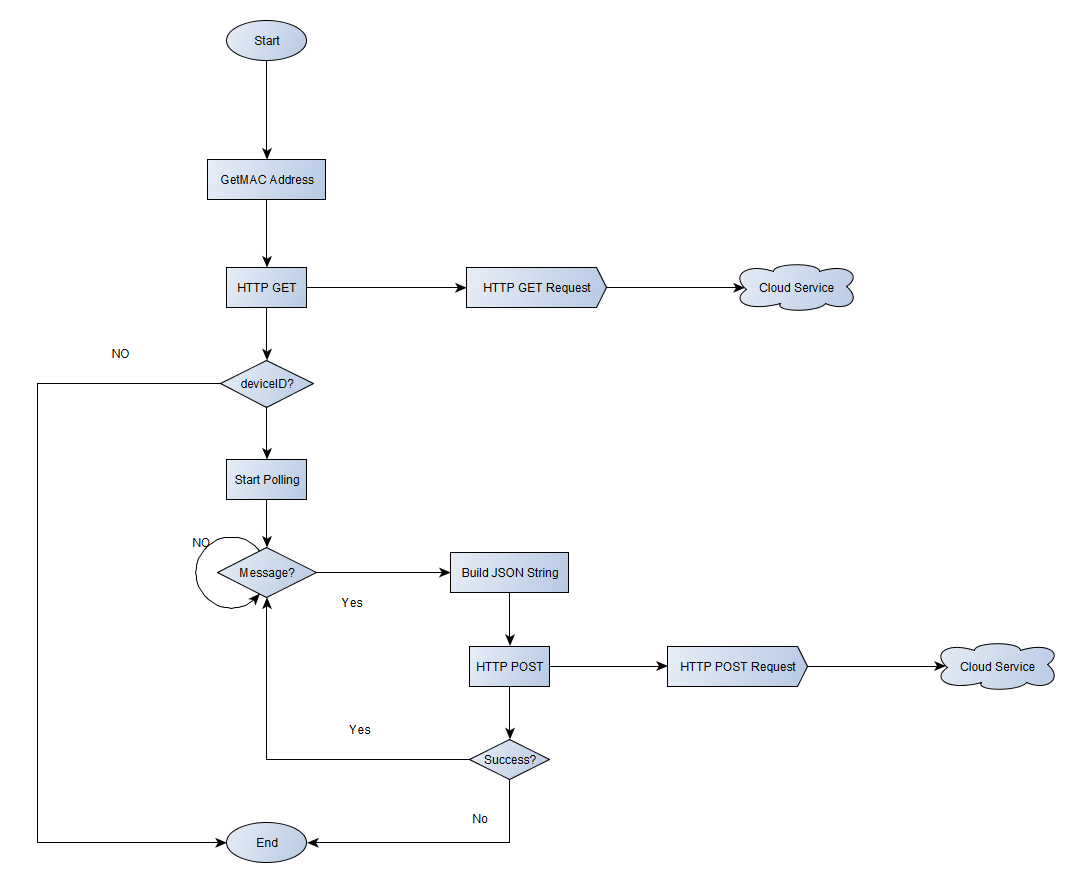


Abbildung 4: Ablaufdiagramm http Task

|  |  |
| --- | --- |
| Verwendete Tools | |
|  | Für das Arbeitspaket wurden folgende Tools verwendet:   * + https://webhook.site (a)   Frei verfügbare Seite zum Testen von Webhooks und HTTP Requests  Erstellt eine eindeutige URL gegen die man seine Implementierung teste kann und welche einem die empfangenen Request inkl. Daten anzeigt.  Im Projekt verwendet um die HTTP POST und GET Funktionalität zu testen ohne, dass das eigentliche Service laufen muss.  Zusätzlich erlaubt die Verwendung des Services eine genauere Inspektion der von der selbst implementierten Methoden versendeten Requests.   * + <https://apitester.com/> (b)   Frei verfügbares Tool zum Erstellen und Versenden verschiedener HTTP Requests. Erlaubt somit das Testen bestehender Endpunkte.  Im Projekt verwendet um die Erreichbarkeit und Funktion des eigenen Servers zu testen.   * + Code Composer Studio (c)   Version: 7.2.0.00013  IDE von Texas Instruments für die Entwicklung von Embedded Projekten  Verwendet zum Entwickeln und Debuggen des Codes für den Mikrocontroller |

|  |  |  |
| --- | --- | --- |
| AP: Sensorik | | |
|  | Bei dem HTU21D Sensor handelt es sich um einen digitalen relativen Feuchtigkeitssensor mit Temperaturausgabe, der dem Plug-and-Play Ansatz folgt. Das Modul bietet kalibrierte, linearisierte Signale, die im digitalen I2C Format ausgelesen werden können. Jeder Sensor wird individuell kalibriert und getestet.  Die Verbindung mit dem Mikrocontroller erfolgt auf einfache Art und Weise durch die modularen, digitalen Feuchtigkeits- und Temperaturausgänge. Der Messbereich umfasst -40 Grad Celsius bis +125 Grad Celsius und 0 bis 100 % relative Feuchtigkeit (relative humidity).  Mögliche Anwendungsbereiche sind unter anderem in der Automobilindustrie, im medizinischen Bereich, bei Luftbefeuchter und in der Wetterbeobachtung.  Der I2C Bus operiert auf zwei bidirektionalen Leitungen, von denen eine mit SDA (serial data) und die andere mit SCL (serial clock) bezeichnet wird. Über SDA findet die tatsächliche Datenübertragung statt. Über SCL werden die Takt-Impulse gesendet. Die I2C-Bausteine lassen sich über ein 7- (oder 10-) Bit breites Adress-Byte selektieren.  Der I2C Bus basiert auf einer Master-Slave-Kommunikation. Der Master sendet eine Start-Condition, wodurch die Slaves angesprochen werden. Sie vergleichen ihre Adresse mit der vom Master spezifizierten Adresse und der angesprochene Slave und der Master können nun eine Kommunikation beginnen.    Die Kommunikation ist folgendermaßen aufgebaut: es wird die Start- oder Repeated Start-Condition gesendet. Darauf folgt die Adresse des angesprochenen Slaves und ein Read/Write Flag. Der Slave bestätigt nun an den Master, dass er bereit ist und der Master kann mit dem Auslesen der Daten fortfahren. Der Mikrocontroller „spricht” mit dem Slave. Möchte der Master weitere Daten lesen, sendet er ein ACK an den Slave. Sonst sendet er kein ACK, sondern abschließend eine Stop-Condition.    Der HTU21D Sensor benötigt eine Betriebsspannung zwischen 1,5V und 3,6V. Nach Inbetriebnahme braucht er höchstens 15ms, um den idle state zu erreichen. Während dieser Zeit muss SCK hoch bleiben. Vor dieser Zeit soll kein Befehl gesendet werden. Um eine einwandfreie Übertragung zu erleichtern, ist es empfohlen, nach Erreichen des idle Zustandes einen Soft Reset durchzuführen. | |
| Verwendete Tools | | |
| Für das Arbeitspaket wurden folgende Tools verwendet:   * + Code Composer Studio (c)   Version: 7.2.0.00013  IDE von Texas Instruments für die Entwicklung von Embedded Projekten  Verwendet zum Entwickeln und Debuggen des Codes für den Mikrocontroller   * + Balsamiq (c)   <https://balsamiq.cloud/>  Eine Webanwendung, mit der man sog. Mockups erstellen kann. | |
| Vorgehensweise | |
| Nach Erhalt des Mikrocontrollers und Sensors begann ich zunächst mit einer ausführlichen Recherche zur Funktionalität des Sensors. Danach musste ich meine Kenntnisse über I2C auffrischen, die bis dahin nur theoretischer Natur waren.    Problem: Davor noch nie mit I2C gearbeitet  Lösung: Beispielimplementierungen aus dem Internet und aus dem letzten Semester heranziehen und versuchen, die Funktionalität nachzuvollziehen.  Nachdem es in der Gruppe diskutiert wurde, wie die Weboberfläche aussehen sollte, erstellte ich Mockups. Dazu musste ich mich auf der Webseite Balsamiq registrieren und eine 30-tägige „Probemitgliedschaft“ abschließen. Weiters musste ich mich mit der Benutzung der Webseite auseinandersetzen.  Anschließend schaute ich mir Videos im Internet zum HTU21D Sensor an und unternahm erste Versuche, die I2C Funktionen, die für das Ein- und Auslesen der Daten benötigt werden, zu programmieren.  Der Beispielcode zum HTU21D Sensor, den ich von der Seite des Herstellers bezogen hatte, half beim Verständnis und bei der Rekonstruktion des grundsätzlichen Ablaufes der Verarbeitung der Daten.  Die Aufgabe bestand in weiterer Folge darin, die im Beispielcode enthaltenen Funktionen so umzusetzen, dass diese mit dem von uns verwendeten Mikrocontroller funktionieren.  Anhand der Codebeispiele auf der Herstellerseite von Texas Instruments, sowie anhand der Beispielimplementierungen aus dem Wintersemester 2018 gelang es mir relativ rasch, die Grundgerüste der ersten Funktionen I2CInit und I2CTransferConfig zu programmieren und die Initialisierung des I2C-Busses und eine Fehlerbehandlung einzubauen.  Der nächste Schritt war das Erstellen eines Grundgerüstes der Funktion setup\_Poll\_Task() - jedoch noch ohne tatsächliche Funktionalität.  Als nächstes wurde die Funktion HTU21DSoftReset entwickelt, welche für ein power off und anschließendes power on des Devices verwendet wird. Es wird empfohlen, die Funktion beim Start des Sensors durchzuführen, um diesen zu rebooten. Es wird 0xFE ins Register geschrieben (11111110), woraufhin das HTU21D Sensor-System neu initialisiert und die Operation gestartet wird.  Problem: Funktion im Beispielcode des Sensors vorhanden, jedoch nicht für den verwendeten Mikrocontroller geeignet.  Lösung: Durch Setzen von Breakpoints und Recherche auf verschiedenen Foren zu Texas Instruments konnte letztendlich ein zufriedenstellendes Grundgerüst gefunden werden.  Im nächsten Schritt wurden die Funktionen HTU21D\_REG\_read und HTU21D\_REG\_write erstellt.  Anschließend wurde die Funktion CalcValues fertig gestellt. In der Funktion werden die Werte aus dem Buffer übergeben und diese ausgelesen. Der so ermittelte Wert wird dann in der jeweiligen Funktion GetTemperature oder GetRelativeHumidity mittels der Berechnung aus der HTU21D Beispielimplementierung in den tatsächlichen Temperatur- / Luftfeuchtigkeitswert umgewandelt.  Für die Funktionen GetTemperature und GetRelativeHumidity wurde einerseits der Beispielcode des HTU21D Sensors herangezogen, andererseits Recherche im Internet durchgeführt. Um die Fehlersuche zu erleichtern, wurden Ausgaben auf der Konsole implementiert. Beim Debugging konnte die korrekte Erfassung der Werte bestätigt werden.  Problem: Beim Debugging war klar ersichtlich, dass die richtigen Werte erfasst werden konnten, jedoch wurde als Ergebnis die Null ausgegeben. Screenshots zum beschriebenen Problem werden weiter unten angehängt.  Lösung: Nach mehreren Debugging Versuchen, Recherche im Internet und Versuchen, Funktionen für die Konvertierung der float Werte zu schreiben, konnte festgestellt werden, dass es sich beim Problem um eine nicht inkludierte Library handelte.  Nachdem diese Library eingebunden wurde, lieferte die Ausgabe der Werte ein zufriedenstellendes Ergebnis.  In weiterer Folge arbeitete ich an den einzelnen Funktionen weiter und recherchierte die logischen Grundlagen hinter der jeweiligen Funktionalität.  Als letztes passte ich die Headerfiles an und versah die jeweiligen Funktionen mit Briefings. Ich führte auch einen Code Refactoring aus und setzte mich mit bedingter Kompilierung auseinander, um die Ausgaben auf der Konsole für Debugging-Zwecke behalten zu können.  Auf die Implementierung eines CRC-Checks wurde aus Zeitgründen verzichtet.  Screenshots:    Abbildung 3: Mockup Hauptseite    Abbildung 4: Mockup Echtzeitdaten    Abbildung 5: Mockup Anzeige Geräte    Abbildung 6: HTU21D Nullwerte bei Ausgabe mit Debugging Hilfen | |

Abbildung 7: HTU21D Variablenwerte



Abbildung 8: Debugging Hilfe



Abbildung 9: Korrekte Variablenwerte

|  |  |
| --- | --- |
| AP: Backend | |
|  | Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat   * Effizientes Projektmanagement durch standardisierte Prozesse * Reibungslose Abwicklung von Projekten durch klare Verantwortlichkeiten * Konzernweite einheitliche Projektmanagement Begriffe und Vorgehensweisen   + Standardisiertes Reporting über den Status der Zielerreichung   + Effiziente Lenkung von knappen Ressourcen   Abbildung 8: Organisation Stab QS 12/2016 |

|  |  |
| --- | --- |
| AP: Alexa Skill | |
|  | Der folgende Abschnitt befasst sich mit der Implementierung des Alexa sowie der Aufsetzung des entsprechenden Environments.  „Alexa frage **bestes Projekt** nach **Temperatur im Abstellraum**.“  Jeder Alexa Skill hat einen sogenannten Invocation Name, welcher sozusagen der Name der App ist. In unserem Fall ist unser Invocation Name „Bestes Projekt“. Jede App kann mehrere Intents haben. Intents sind dann sozusagen die Features der App, im obigen Beispiel haben wir den Intent „Temperatur im Abstellraum“. Somit wird das Feature Temperatur von Bestes Projekt aufgerufen.  Die Spracherkennung wird in der Cloud durchgeführt, wo auch auf Basis der Triggerwords die entsprechenden Funktionen durchgeführt werden.  Grundlegend gibt es verschiedene Wege einen Alexa Skill zu schreiben.  Ein Alexa Skill besteht aus drei grundlegenden Komponenten:   * skill.json * Interaction Model * Business Logik   Im skill.json File wird der Skill anhand von Metainformationen (zb. Name, Kategorie ...) grundlegend beschreiben. Wichtig ist hier die Angabe, wo sich die Business Logik befindet. Prinzipiell gibt es zwei Möglichkeiten die Business Logik zu hosten. Entweder über AWS Lambda oder als Webservice. Wir haben uns aus Komfortgründen für die erste Variante entschieden.  Das Interaction Model (IM) beschreibt die Schnittstelle zum User, pro Sprache des Skills gibt es jeweils ein IM. Das IM setzt sich aus Intents und Slots zusammen.  Slots sind Parameter, die ein User einem Intent mitgeben kann um die Business Logik durchzuführen. In unserem Beispiel haben wir den Ort (zb.Abstellraum) der Wetterstation als Slot.  Je nach gewähltem Intent kann eine andere Business Logik durchgeführt werden.  Das Interaction Model wir anhand eines JSON Files beschrieben. Die Alexa Developer Console ermöglicht es das IM auch grafisch zusammenzustellen.  Hier ein Auszug unseres Interaction Models:  {  "interactionModel": {  "languageModel": {  "invocationName": "bestes projekt",  "intents": [  {  "name": "status",  "slots": [  {  "name": "deviceID",  "type": "AMAZON.Room",  "samples": [  "Abstellraum",  "Kühlraum",  "Kühlschrank",  "Badezimmer",  "Küche",  "Wohnzimmer",  "Schlafzimmer"  ]  }  ],  "samples": [  "{deviceID}",  "Temperatur im {deviceID}",  "nach {deviceID}",  "erzähle mir etwas über {deviceID}",  "Information über {deviceID}",  "Status im {deviceID}"  ]  }  ]  },    }  }  Da entschieden wurde das AWS Lambda genutzt wird, musste entschieden erden in welcher Programmiersprache die Funktion geschrieben werden soll. AWS Lambda unterstützt folgende Sprachen:   * Java * nodeJS * Python * C# * Go   Um einen Einblick in die Welt von JavaScript als Backendsprache zu bekommen, haben wir uns für nodeJS entschieden.  Jede Lambda Funktion benötigt einen Handler welcher ausprogrammiert wird.  Unsere Lambda Funktion ruft die REST API des Backends in Azure auf und retourniert anschließend den Text welcher von Alexa vorgelesen wird.  Auszug aus Lambda Funktion (Aufruf der REST API):  var req = http.request(options, function (res) {  var chunks = [];  res.on("data", function (chunk) {  chunks.push(chunk);  });  //Verarbeite Response  res.on("end", function () {  var body = Buffer.concat(chunks);  var data = JSON.parse(body); //Parse JSON Objekt  var temperatur = data.temperature + '';  temperatur = temperatur.replace('.',',');  var text = 'Vielen Dank, dass du das beste Projekt nutzt. Die Temperatur in deinem ' + event.request.intent.slots.deviceID.value+ ' beträgt ' + temperatur + ' Grad Celsius' + ', und die Luftfeuchtigkeit beträgt ' + data.humidity + ' Prozent';  output(text,context); // Übergebe Antwort  });  });  Mittels event.request.intent.slots.deviceID.value wird es ermöglicht auf den übergebenen Slot (Parameter) zuzugreifen. Der Slot entspricht in unserem Fall dem Raum für welchen die Messungen abgefragt werden. |
|  |  |

|  |  |
| --- | --- |
| AP: Frontend Implementierung | |
|  | Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat   * Effizientes Projektmanagement durch standardisierte Prozesse * Reibungslose Abwicklung von Projekten durch klare Verantwortlichkeiten * Konzernweite einheitliche Projektmanagement Begriffe und Vorgehensweisen   + Standardisiertes Reporting über den Status der Zielerreichung   + Effiziente Lenkung von knappen Ressourcen   Abbildung 1: Organisation Stab QS 12/2016 |

|  |  |
| --- | --- |
| AP: Embedded Integration | |
|  | Der folgende Abschnitt befasst sich mit der Integration der einzelnen Embedded-Module in eine gesamtheitliche Lösung.  Anfangs werden verschiedene Integrationsszenarien verglichen und die Vorzüge der gewählten Variante dargelegt.  Im Zuge des Arbeitspaketes mussten daher einige Änderungen an den bestehenden Modulen durchgeführt werden. Auf diese wird in weiterer Folge näher eingegangen.  Zusätzlich wurde das Arbeitspaket verwendet um Refactoring-Maßnahmen durchzuführen die den Programmcode selbst oder seine Lesbarkeit optimieren. |

|  |  |
| --- | --- |
| Analyse und Bewertung Integrationsszenarien | |
|  | Zunächst Analyse und Bewertung möglicher Integrationsszenarien:   * Mutexes & Semaphore * Queue * Mailboxen  Mutexes & Semaphore Bei Mutexes handelt es sich um binäre Semaphore die benutzt werden um simplen wechselseitigen Ausschluss zu gewährleisten.  Obwohl Mutexes in Kombination mit globalen Variablen für die Kommunikation zwischen dem Sensorik-Task und dem Netzwerk-Task verwendbar wären besitzen sie einige Nachteile weswegen von einer Verwendung abgesehen wurde.  Erstens muss sichergestellt werden, dass ein Task der den Token besitzt diesen auch wieder zurückgibt damit ein anderer auf die Ressource zugreifen kann.  Zweitens besteht zwar die Möglichkeit eine Sperrzeit zu spezifizieren welche in Ticks angegeben wird, jedoch besteht kaum Möglichkeit die Laufzeit eines HTTP Requests im Voraus zu erahnen.  Drittens erfordert die Verwendung eine engere Verzahnung der einzelnen Softwaremodule statt größtmögliche Entkoppelung zu gewährleisten. Queue Queues haben verfügen über ähnliche Eigenschaften wie Mailboxen.  Der Unterschied besteht primär darin, dass Queues auf doppelt verlinkten Listen und Mailboxen auf Kopien basieren.  Queues besitzen aber keine definierte Maximalgröße.  Da durch vorhergehende Lehrveranstaltungen aber bereits Erfahrungen mit der Verwendung von Mailboxen bestehen wurde diese Variante präferiert. Mailboxen Zusätzlich zu den bestehenden Vorerfahrungen des Projektteams mit Mailboxen bieten diese einen weiteren Vorteil. Mit ihnen ist es möglich sicherzustellen, dass der Input an Nachrichten nicht die Anzahl der bearbeitbaren übersteigt.  Die gewählte Lösung mittels Mailboxen wird im nächsten Abschnitt genauer erläutert. |

|  |  |
| --- | --- |
| Integration mittels Mailboxen | |
|  | Um eine möglichst hohe Entkoppelung der Softwaremodule zu erreichen und dennoch eine stabile und performante Lösung zu erhalten wurden für die Kommunikation zwischen den Tasks Mailboxen eingesetzt.  Diese bieten zusätzlich die Möglichkeit auf einfache Art weitere Sensoren und entsprechende Tasks hinzuzufügen ohne hohen Änderungsaufwand bei anderen Modulen zu erzeugen. |

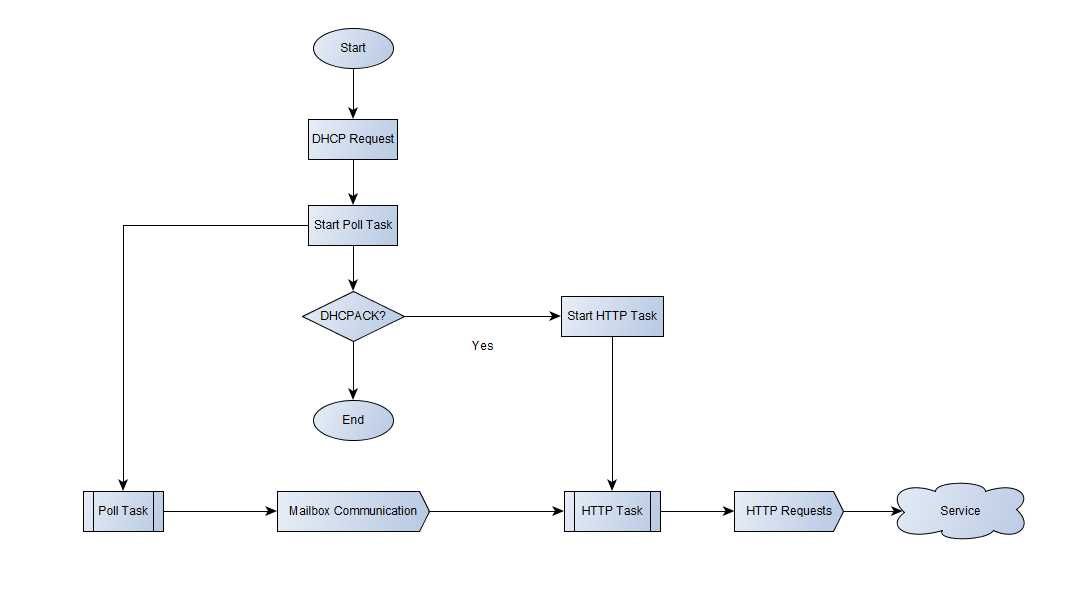


Abbildung 10: Ablaufdiagramm Embedded Lösung

|  |  |
| --- | --- |
|  | Die obige Grafik zeigt den schematischen Ablauf der implementieren Lösung.  Nach dem Start des Mikrocontrollers wird zunächst ein DHCP Request ausgesendet um eine IP-Adresse zu erhalten und die Sensorik Funktionalität gestartet.  Anschließend wartet das Gerät bis es eine IP Adresse erhält. Sollte dies nicht der Fall sein wird die Applikation beendet da es keinen Sinn macht Werte zu ermitteln für die kein Konsument existiert bzw. davon ausgegangen werden muss, dass ein Fehler im Netzwerk vorliegt.  Sobald das Gerät eine gültige Adresse erhält wird der http Task gestartet.  Hat dieser die, in Abschnitt „5 AP Netzwerk“ nötigen Schritte näher beschriebenen Schritte, zum Aufbau einer Serververbindung vollführt werden die Messwerte per Mailbox empfangen und an den Server weitergeleitet.  Die Größe der Mailboxen ist beschränkt und stellt somit sicher, dass es zu keinem Überlauf kommt wenn der http Task nicht mehr in der Lage ist Daten zu versenden, der Sensorik Task aber weiterhin Werte sendet.  Um das Hinzufügen von neuen Funktionen zu erleichtern wurden für Temperatur und Luftfeuchtigkeit unterschiedliche Mailboxen implementiert.  Alternativ hätte auch mittels einem struct ein eigenes Nachrichtenformat definiert werden können. |

|  |  |
| --- | --- |
| Aufgetretene Herausforderungen | |
|  | In diesem Abschnitt werden die Herausforderungen/Probleme erläutert die im Zuge des Arbeitspaketes aufgetreten sind. Zusätzlich wird beschrieben wie diese gelöst wurden und welche Tools oder Recherchequellen dafür dienlich waren. |

|  |  |
| --- | --- |
| Refactoring-Maßnahmen | |
|  | Der Abschnitt erläutert die durchgeführten Refactoring-Maßnahmen sowie warum diese gesetzt wurden und welche Auswirkungen diese haben.  Folgende Refactoring-Maßnahmen wurden im Zuge des Arbeitspaketes durchgeführt:   * Entfernen doppelter Includes * Entfernen von überflüssigen Kommentaren * Entfernen von Hilfswerkzeugen der Entwicklung z.B. Hilfsfunktionen * Conditional Compilation  Entfernen doppelter Includes Dies ist eine Maßnahme um Fehler durch inkonsistente Definitionen zu vermeiden.  Zusätzlich wird der Zeitaufwand des Kompilierens reduziert wenn Schritte nicht mehrfach ausgeführt werden müssen. Entfernen überflüssiger Kommentare Nicht mehr benötigte Kommentare zu entfernen erleichtert einerseits die Lesbarkeit der Sourcefiles andererseits hilft es auch Verwirrungen zu reduzieren.  Oft handelt es sich um Artefakte des Debuggings welche in der fertigen Lösung keinen Zusammenhang besitzen. Entfernen von Hilfswerkzeugen der Entwicklung Das Entfernen von Hilfs-Konstrukten ist nur eine von vielen Maßnahmen um den eigenen Programmcode verständlicher und leichter lesbar zu machen.  Ein weiterer Benefit ist, dass Verwirrungen vermieden werden indem nicht mehr benutzte Codeteile entfernt wird. Ein späterer Leser ist also nicht mehr in Versuchung Verweisen zu folgen die keine Verwendung mehr finden und für die Funktionalität keine Relevanz besitzen. Conditional Compilation - stellt die Möglichkeit dar, dass ein Compiler, abhängig von definierten Parametern, unterschiedliche Programme erzeugt.  Dies hat den Nutzen, dass beispielsweise durch das Hinzufügen einer einzelnen Zeile „#define DEBUG“ das spätere Programm dem Benutzer viel mehr Zwischeninformationen der Verarbeitung anzeigt als üblich.  Dies erzeugt speziell für das Debugging von Software einen hohen Mehrwert.  Die Grafik unterhalb zeigt ein Beispiel wie ein solcher abhängiger Codeabschnitt aussehen kann.    Abbildung 10: Beispiel Conditional Compilation  Diese Ausgabe wird nur in den Kompilierungsprozess miteinbezogen und dadurch im späteren Programm ausgeführt wenn das Schlüsselwort „DEBUG“ definiert wurde. |

|  |  |
| --- | --- |
| Installationsanleitung | |
| Da es sich bei dieser Lösung um kein fertiges Produkt handelt sind für die Inbetriebnahme folgende Tools notwendig:   * Code Composer Studio (c) * Visual Studio Code * nodeJS * ASK CLI   eLorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat |

|  |  |
| --- | --- |
| Lessons Learned | |
| Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat   * Effizientes Projektmanagement durch standardisierte Prozesse * Reibungslose Abwicklung von Projekten durch klare Verantwortlichkeiten * Konzernweite einheitliche Projektmanagement Begriffe und Vorgehensweisen   + Standardisiertes Reporting über den Status der Zielerreichung   + Effiziente Lenkung von knappen Ressourcen |

# Anhang

## Verwendete Tools

(a) Webhook., *Tool zum Testen von diversen HTTP Requests*., <https://webhook.site>

(b) Apitester., *Tool zum Testen von APIs*., <https://apitester.com>

(c) CCS., *IDE zur Entwicklung von embedded Projekten*., Code Composer Studio

(d) yEd., *Tool zum Erstellen von Diagrammen*., <https://www.yworks.com/products/yed>?

## Abbildungsverzeichnis

[Abbildung 1: Netzwerktopologie Abschlusspräsentation 21](#_Toc518260512)

[Abbildung 1: Organisation Stab QS 12/2016 23](#_Toc518260513)

[Abbildung 1: Organisation Stab QS 12/2016 23](#_Toc518260514)

[Abbildung 4: Netzwerktopologie Betriebszenario 24](#_Toc518260515)

[Abbildung 3: Mockup Hauptseite 30](#_Toc518260516)

[Abbildung 8: Organisation Stab QS 12/2016 34](#_Toc518260517)

[Abbildung 1: Organisation Stab QS 12/2016 34](#_Toc518260518)

[Abbildung 1: Organisation Stab QS 12/2016 35](#_Toc518260519)

[Abbildung 10: Ablaufdiagramm Embedded Lösung 37](#_Toc518260520)

[Abbildung 10: Beispiel Conditional Compilation 39](#_Toc518260521)

ANMERKUNG: Dieses Abbildungsverzeichnis generiert sich selbst.

## Recherchelisten

## Quellenverzeichnis

[1] ÖBB-Holding AG, *https://konzern.oebb.at/de/ueber-den-konzern/organisation*., ÖBB-Holding AG, Wien 24.05.2018.

[9] J. Rüegg-Stürm, *Das neue St. Galler Management-Modell. Grundkategorien einer integrierten Managementlehre. Der HSG-Ansatz.*, Haupt, 2003. ISBN 3258066299

https://e2e.ti.com/support/embedded/tirtos/f/355/t/555614?HTTP-POST-sample

# Designvorlage

ANMERKUNG: Bitte die angegebenen Formatierungen verwenden (Format > Formatvorlagen und Formatierungen) – sie entsprechen dem Corporate Design der FH Technikum Wien!

# Überschrift 1

Standard - Fließtext

## Überschrift 2

* Formatvorlage Aufzählungen 1
* Formatvorlage Aufzählungen 1
* Formatvorlage Aufzählungen 1
  + Formatvorlage Aufzählungen 2
  + Formatvorlage Aufzählungen 2
  + Formatvorlage Aufzählungen 3
  + Formatvorlage Aufzählungen 3

### Überschrift 3

Formatierung Hyperlink: [www.technikum-wien.at](http://www.technikum-wien.at/)

#### Überschrift 4