|  |  |  |
| --- | --- | --- |
|  | B | |
|  |  | **BELEGARBEIT**  Digitalisierung von Maschinen |
|  |  |  |
| Mohammad Ghaith Albaba |  | Angewandte Informatik - Digitale Medien und Spieleentwicklung, 2018, 4068638 |
| Vorname Nachname #1 |  | Studiengang, Matrikel, Matrikelnummer | |

|  |  |  |
| --- | --- | --- |
| Mohammad Al Jazzar |  | Angewandte Informatik - Digitale Medien und Spieleentwicklung, 2018, 4068905 |
| Vorname Nachname #2 |  | Studiengang, Matrikel, Matrikelnummer | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Thema: |  |  |
| Sensoraustausch und Darstellung | | |

|  |  |  |
| --- | --- | --- |
|  |  | Prof. Dr. S. Twieg |
|  |  | Dozent |

|  |  |  |
| --- | --- | --- |
|  |  | 16.09.2020 |
|  |  | Abgabe am |

Selbstständigkeitserklärung

Hiermit erkläre/n ich/wir, dass die Arbeit selbständig verfasst, in gleicher oder ähnlicher Fassung noch nicht in einem anderen Studiengang als Prüfungsleistung vorgelegt wurde und keine anderen als die angegebenen Hilfsmittel und Quellen, einschließlich der angegebenen oder beschriebenen Software, verwendet wurden.

|  |  |  |
| --- | --- | --- |
| Köthen, 18.09.2020 |  |  |
| Ort, Datum |  | Unterschrift/en der/des Studierenden |

Inhaltsverzeichnis

[Selbstständigkeitserklärung II](#_Toc51420245)

[1 Motivation und Zielsetzung 1](#_Toc51420246)

[1.1 Einleitung in die Thematik 1](#_Toc51420247)

[1.2 Zielsetzung der Arbeit 1](#_Toc51420248)

[2 Themenspezifisches Kapitel 2](#_Toc51420249)

[2.1 Sinn und Zweck 2](#_Toc51420250)

[2.2 Funktionale Anforderungen 2](#_Toc51420251)

[2.3 Nichtfunktionale Anforderungen 3](#_Toc51420252)

[3 Umsetzung 5](#_Toc51420253)

[3.1 Kommunikation zwischen Komponenten 5](#_Toc51420254)

[3.2 Verwendete Hardware und ihre Spezifikation 6](#_Toc51420255)

[3.3 Programmierumgebung 8](#_Toc51420256)

[3.4 Struktur der Arbeitsschritte 9](#_Toc51420257)

[3.5 Prototyp und Ausführen der Mobileanwendung 10](#_Toc51420258)

[3.6 Entworfene Schaltung und Programmablaufplan des Projekts 13](#_Toc51420259)

[4 Zusammenfassung und Ausblick 15](#_Toc51420260)

[Abbildungsverzeichnis i](#_Toc51420261)

[Tabellenverzeichnis i](#_Toc51420262)

[Literaturverzeichnis ii](#_Toc51420263)

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |

# Motivation und Zielsetzung

## Einleitung in die Thematik

In unserem Projekt geht es darum, dass die Verbindung zwischen zwei Wemos-Module in verschiedenen Orten ermöglicht wird. Das Projekt beschäftigt sich mit einfachen Sensoren, dabei versuchen wir das Konzept der Datenübertragung leichter mit den aktuellen Programmiersprachen und Frameworks. Somit kann man die Daten von einem X-Sensor, der mit X-Wemos verbunden ist, ohne große Mühe übertragen und andere Daten von Y-Wemos empfangen.

## Zielsetzung der Arbeit

Die getauschten Daten werden durch eine Handyapplikation visualisiert. Außerdem werden alle empfangenen Sensordaten auf einem LCD, das mit einem X-Wemos verbunden ist, dargestellt. Das Darstellen der gesendeten Sensordaten im Projekt wird auf jeweils seriellem Port eingeschlossen.

Die App soll ohne größere Anpassungen auf folgenden Zielplattformen lauffähig sein Android, iOS, damit unser App den großen Anteil des Zielsegments erreichen könnte. Der Fokus liegt bei unserem Projekt auf kurzen Entwicklungszeiten, schneller Ausführungsgeschwindigkeit und auch auf nativer User Experience. Das Design der App soll nicht nur Texte enthalten, sondern auch schöne und moderne (Symbolen, Tasten), die effektiv und attraktiv sind

# Themenspezifisches Kapitel

## Sinn und Zweck

Dieses Projekt wird entwickelt, damit die Studenten lernen, mit Entwicklungsboards und einfache Sensoren umzugehen und diese zu programmieren.

In diesem Projekt werden die neuesten Technologien und Programmiersprachen für die meisten Plattformen wie Android und IOS behandelt.

Dieses Projekt ist ein einfacher Einstieg, um Erfahrungen zu sammeln und die Studenten auf größere Projekte zur Entwicklung elektronischer Bedienfelder und fortschrittlicher Software vorzubereiten.

## Funktionale Anforderungen

* Das System muss dem Nutzer die Möglichkeit geben, mit dem MQTT-Broker verbinden zu können.
* Das System muss dem Nutzer die Möglichkeit geben, das zu abonnierenden Topic einzutragen.
* Das System muss dem Nutzer die Möglichkeit geben, das zu veröffentlichenden Topic einzutragen.
* Das System muss dem Nutzer die Möglichkeit geben, die Nachrichten vom eingetragenen Topic zu empfangen.
* Das System muss dem Nutzer die Möglichkeit geben, ein LED durch einen Knopf an/auszumachen.
* Das System muss dem Nutzer die Möglichkeit geben, die von Sensoren gemessenen Daten darzustellen.
* Das System muss dem Nutzer die Möglichkeit geben, mit dem MQTT-Broker verbinden zu können.
* Das System muss dem Nutzer die Möglichkeit geben, die Sensordaten an dem Topic durch MQTT-Protokoll zu senden.
* Das System muss dem Nutzer die Möglichkeit geben, eine Nachricht (an-/ausmachen) von der App durch den Knopf zu empfangen.
* Das System muss dem Nutzer die Möglichkeit geben, die Sensordaten auf einem LCD zu repräsentieren.
* Das System muss dem Nutzer die Möglichkeit geben, die Daten auf dem Bildschirm durch seriellen Port zu zeigen.

## Nichtfunktionale Anforderungen

Tabelle 1: Qualitätsanforderung

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Produktqualität** | **Sehr gut** | **Gut** | **Normal** | **Nicht relevant** |
| **Funktionalität** | **X** |  |  |  |
| **Angemessenheit** |  | **X** |  |  |
| **Genauigkeit** | **X** |  |  |  |
| **Interoperabilität** |  |  | **X** |  |
| **Sicherheit** |  |  |  | **X** |
| **Zuverlässigkeit** |  | **X** |  |  |
| **Fehlertoleranz** |  | **X** |  |  |
| **Wiederherstellbarkeit** |  |  |  | **X** |
| **Benutzbarkeit** |  | **X** |  |  |
| **Verständlichkeit** |  |  | **X** |  |
| **Erlernbarkeit** |  | **X** |  |  |
| **Bedienbarkeit** | **X** |  |  |  |
| **Attraktivität** |  |  | **X** |  |
| **Effizienz** |  | **X** |  |  |
| **Zeitverhalten** | **X** |  |  |  |
| **Wartbarkeit** |  |  | **X** |  |
| **Stabilität** |  | **X** |  |  |
| **Austauschbarkeit** | **X** |  |  |  |
| **Erweiterbarkeit** | **X** |  |  |  |

Dieses Projekt könnte auf verschiedenen Arten durchgeführt werden. Beispielsweise könnte die berühmte Python-Sprache verwendet werden, aber unser Team hat einen anderen Weg gewählt. Das Team hat die Entscheidung getroffen, sich zu verbessern, indem es neue Umgebung kennenlernt, Diese Umgebung hat zwar einige Schwierigkeiten aber auch viele Vorteile damit. Jedes SDK und jede Programmiersprache hat Vor- und Nachteile. Zusammenfassend lässt sich sagen, dass die Vorteile von Flutter gegenüber ähnlichen Systemen deutlich überwiegen.

Tabelle 2: Vorteile & Schwierigkeiten der Lösungsmöglichkeit

|  |  |
| --- | --- |
| **Vorteile** | **Schwierigkeiten** |
| Gute Performance wie die Native Applikationen | Programmcode wird durch die Einbindung der Widgets leicht unübersichtlich |
| Eine Codebasis für alle wichtigen Zielplattformen | Noch junge, wenig verbreitete Sprache; bis jetzt kleine Community |
| Open-Source |  |
| Einfach zu erweitern |  |
| Performante Ausführung der nativen Apps auf Smartphones |  |
| Umfangreiche Bibliotheken mit vorgefertigten UI-Elementen |  |

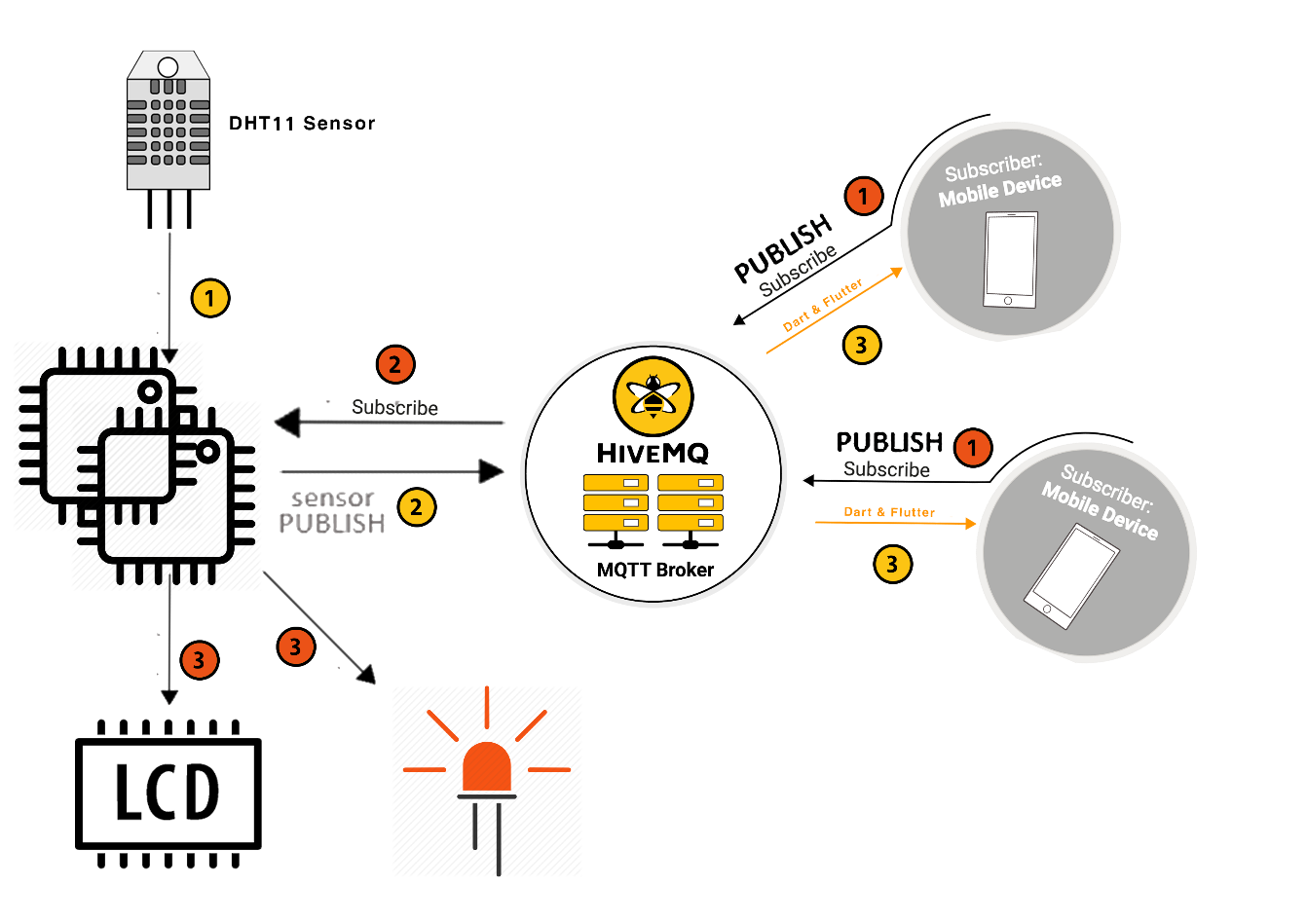


Abbildung 1: Technologieschema

# Umsetzung

## Kommunikation zwischen Komponenten

Da es in diesem Projekt um das Vertauschen der Sensordaten von einer Maschine zu anderer geht, wird das bekannte MQTT-Protokoll verwendet.

MQTT, Das M2M und IoT Protokoll steht für Message Queuing Telemetry Transport und ist offenes Netzwerkprotokoll für Machine-to-Machine-Kommunikation (M2M). Dies Protokoll wurde im Projekt benutzt, damit es möglich wäre, Daten von einem Wemos-Modul zu dem anderen zu übertragen. Es ist also die Schnittstelle zwischen den beiden Wemos-Modulen. Darüber hinaus kann man Mit MQTT Daten nahezu in Echtzeit mit vordefinierten Zustellgarantien über das Internet liefern. [WM20]

Im World Wide Web gibt es viele Provider, die dieses Protokoll den Nutzer beistellen. Ein davon ist HiveMQ MQTT Broker. HiveMQ MQTT Broker ist führend bei MQTT-Bereitstellungen in Unternehmen. HiveMQ ist eine MQTT-basierte Messaging-Plattform, die von Grund auf für eine schnelle und effiziente Datenübertragung zu und von verbundenen IoT-Geräten entwickelt wurde. [HM20]

Das MQTT-Protokoll folgt den Regeln einer Publish-Subscribe-Kommunikation. Es gibt zwei verschiedene Teilnehmer: Einen Broker, und “n” Clients, wobei die Clients als Publisher und Subscriber nicht direkt miteinander kommunizieren, sondern Nachrichten “publishen” (veröffentlichen) und “subscriben” (abonnieren). Die Aufgabe des Brokers ist hier die Nachrichtenverwaltung und -verteilung.

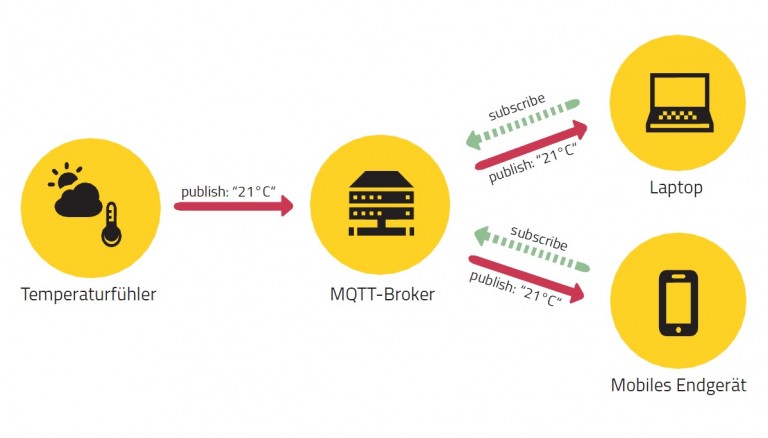


Abbildung 2: MQTT Kommunikationsparadigma [MD20]

Der Client ist, wie man sich ganz typisch vorstellt, der “Endnutzer” der Kommunikation und derjenige, der Nachrichten aktiv sendet. Ein Client kann zu einem Zeitpunkt Nachrichten eines Topics empfangen (Subscriber) als auch Nachrichten in dem gleichen Topic veröffentlichen (Publisher). Jeder Client identifiziert sich durch eine Client ID, die auch seine Session komplett identifiziert, denn MQTT ist für die Client-Seite komplett stateless.

Die Kommunikation von MQTT stützt sich auf ein sogenanntes “Topic”-Prinzip: Jede Nachricht wird einem Topic zugeordnet. Das heißt, jede valide MQTT-Nachricht enthält eine Payload mit einem zugehörigen Topic. Topics müssen von den Clients abonniert werden, um Nachrichten zu empfangen. Schließt sich nun ein neuer Client dem Netz an und schickt dem Broker eine Subscription beispielweise zum Topic “x/y/z ”, wird der Broker alle Nachrichten mit diesem Topic an den Subscriber weiterleiten.

Das “Backend” für MQTT, genannt Broker, verwaltet und administriert jeglichen Datenverkehr. Zu seinen Aufgaben zählen die Speicherung, Verwaltung und Verteilung aller Informationen zu Topics. [MD]

## Verwendete Hardware und ihre Spezifikation

Der Hauptbestandteil in dem Projekt ist das Wemos ESP8266. Das Board Wemos D1 verfügt über einen DC Spannungseingang für eine 9 bis 24 Volt Stromversorgung. Des Weiteren über eine Micro USB-Anschluss. Des Weiteren hat der D1 einen Reset Taster zum neustarten des Modules. Es werden nur die Daten aus dem flüchtigen Speicher gelöscht. Das Wemos D1 Entwicklungsboard ermöglicht es dank dem integrierten ESP-8266 WLAN /WiFi Chip Projekte, welche WLAN benötigen einfach, zu realisieren.

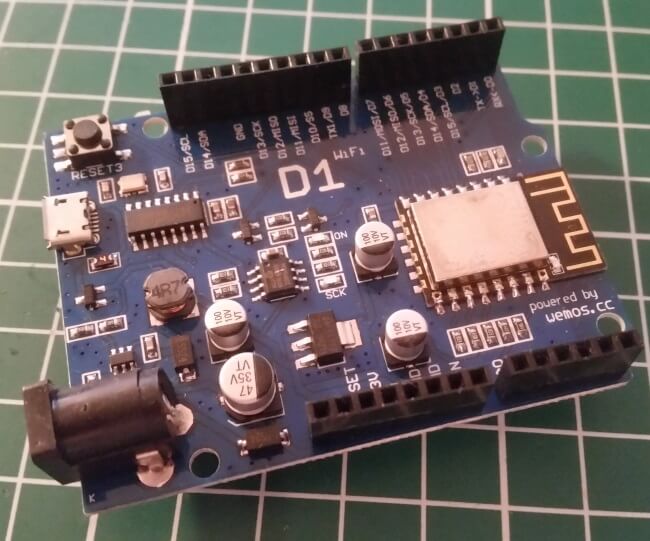
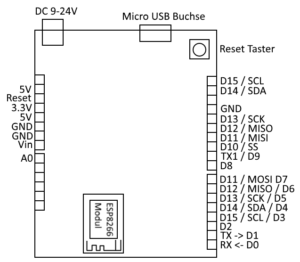


Abbildung 3: Fotos von Wemos ESP8266 [WR20]

[](https://draeger-it.blog/wemos-d1-arduino-uno-kompatibles-board-mit-esp8266-chip/)

Abbildung 4: Spezifikation des Wemos [WR20]

Neben dem Wemos wird den Sensor DHT11 im Projekt verwendet, der die Luftfeuchtigkeit und die Temperatur messt. Der DHT11 ist ein lowcost Sensor mit digitaler Schnittstelle, mit einem 1-Wire Anschluss. Der Sensor ist bereits Werkseitig Kalibriert.

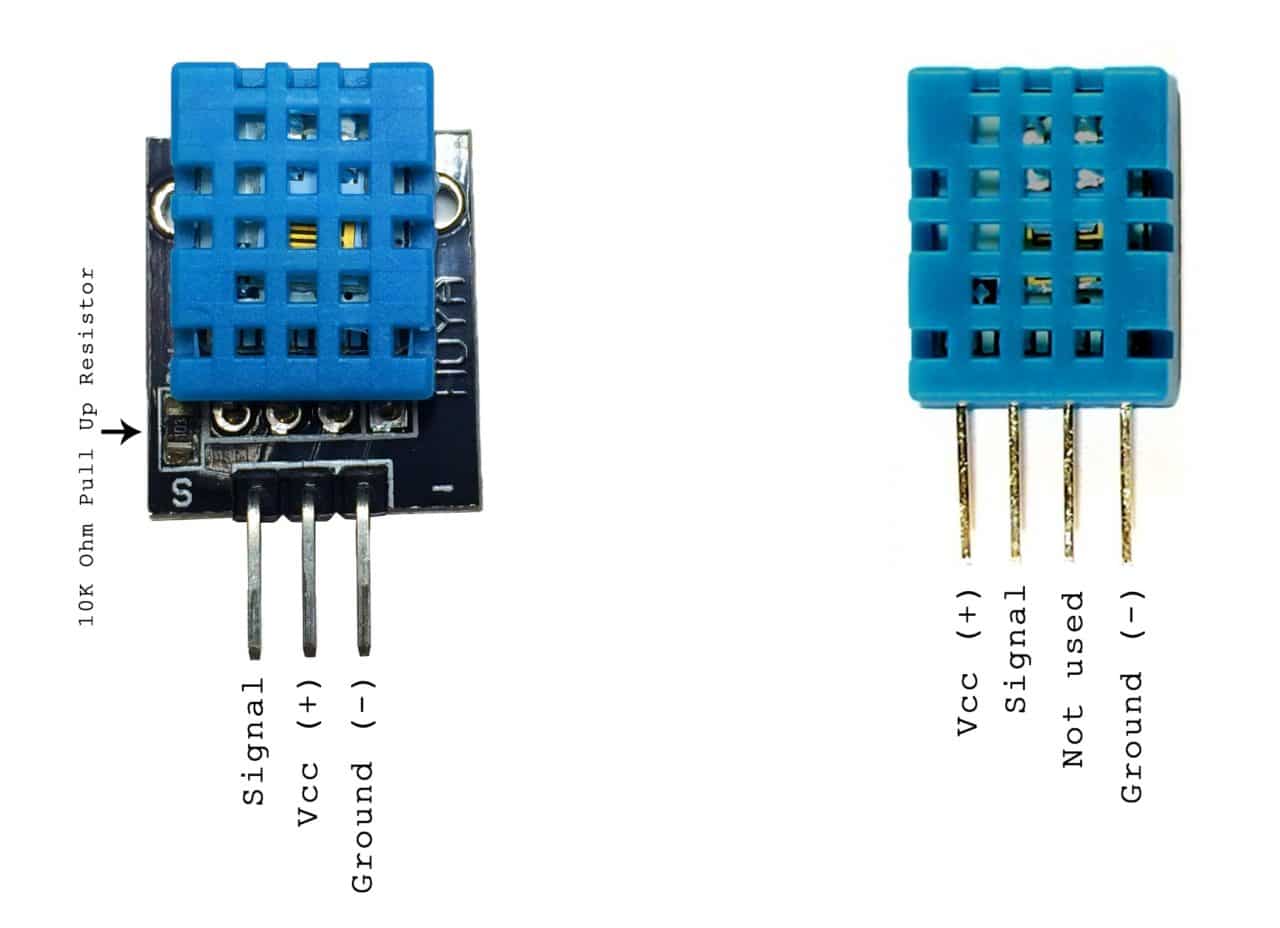


Abbildung 5: Reelles Foto vom DHT11 Sensor [DH20]

Um die vom DHT11-Sensor gemessene Daten anzuzeigen, wird LCD verwendet.  
Dot-Matrix LCD-Displays eignen sich aufgrund ihres erträglichen Preises und des geringen Ansteueraufwandes gut, um Daten anzuzeigen, die max. aus 16 Zeichen \* 2 Zeilen bestehen könnten. Allerdings treten immer wieder Probleme auf, wenn man ein solches Display zum ersten Mal ansteuern will. Dazu muss man 11 Pins vom an dem Microcontroller anschließen, um den LCD zu bedienen. Deshalb wurde I2C-Schnittstelle im Projekt verwendet. So kann man die Anzahl der am Mikrocontroller zu verbindenden Pins verringern.

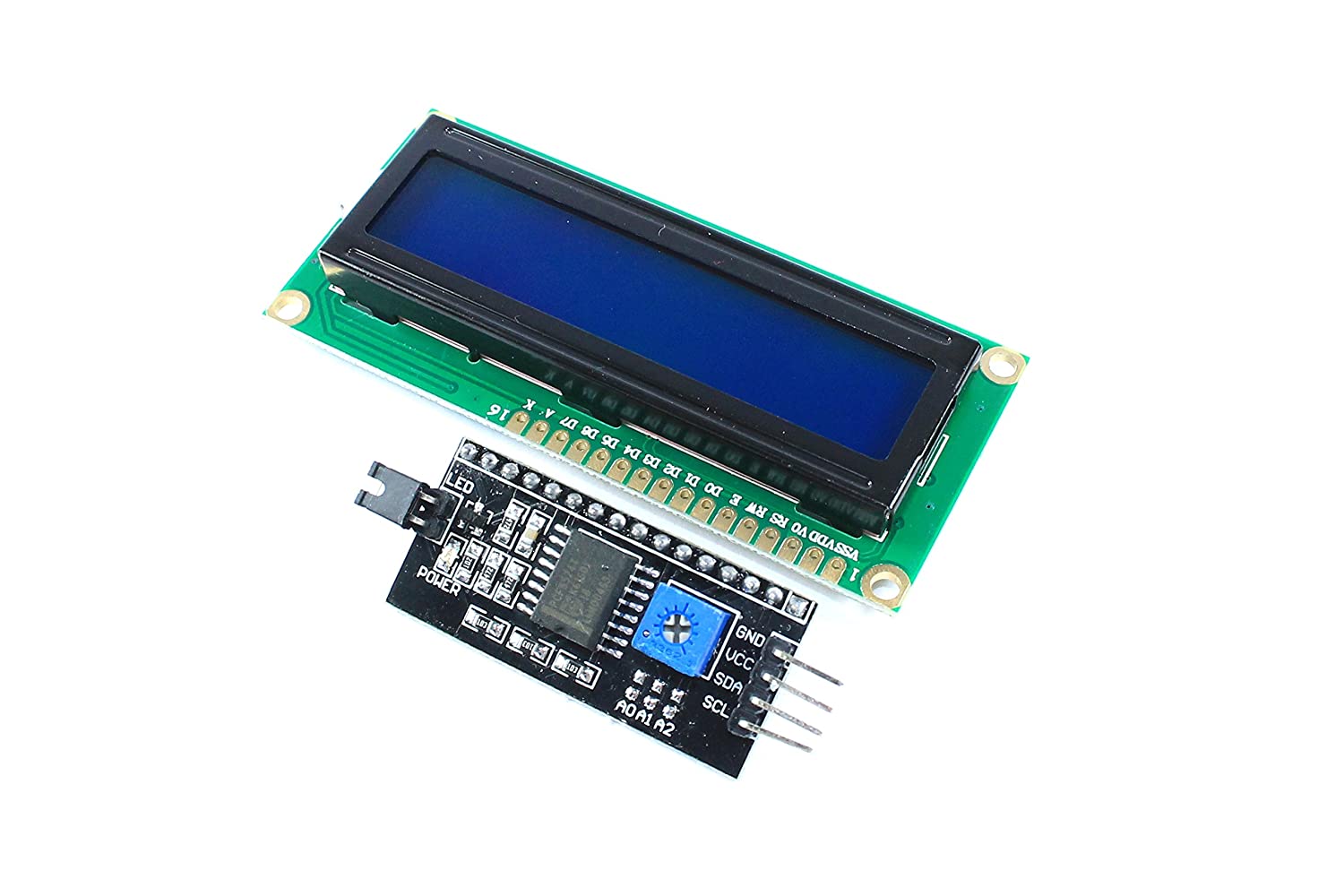


Abbildung 6: LCD mit I2C [LL20]

Um die LED mit dem Microcontroller anzuschließen soll der Widerstand dazwischenstehen. Der Widerstand sorgt um die passende Spannung für die LED. Man soll also die Anode der Microcontroller an einer von den beiden Seiten anschließen. Und dann an die Anode der LED, danach von der Katode der LED an der Katode des Microcontrollers. Damit ist die Schaltung geschlossen ist.

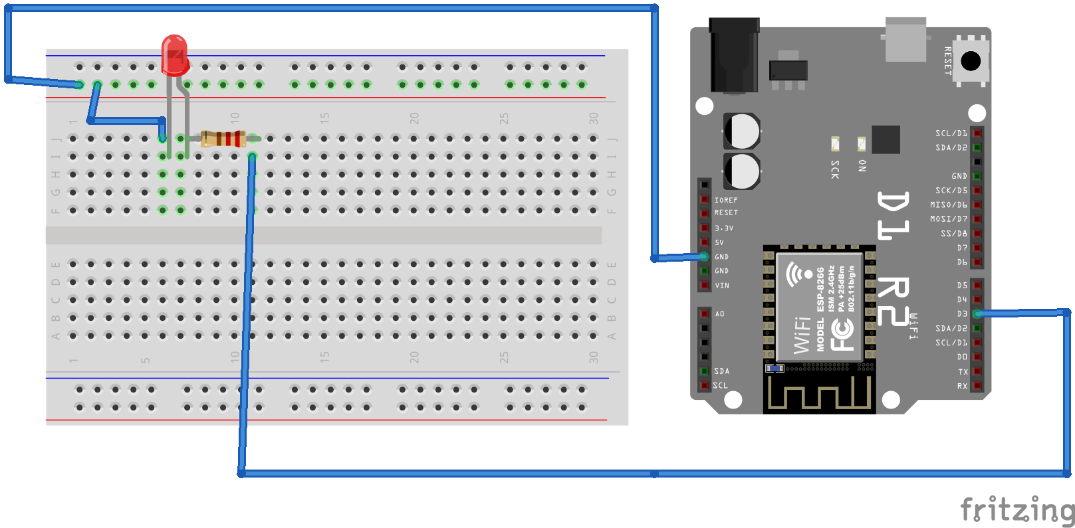


Abbildung 7: Steckplatine von LED und Widerstand

## Programmierumgebung

Es gab viele Variationen um die Applikation umzusetzen, aber die Dart-Programmiersprache wurde aus mehreren Gründen aus allen ausgewählt.

Mit Dart wurde von Google eine neue Programmiersprache vorgestellt – und positioniert sie auf den ersten Blick auf Augenhöhe mit JavaScript. Dabei weiß Dart mit einem neuen Typsystem zu Punkten, das die Vorteile statischer und dynamischer Typisierung miteinander verknüpft. Interessant ist die Sprache besonders für den mobilen Bereich. Dart ist von der Syntax her in die Tradition der C-ähnlichen Sprachen einzureihen. Besonders interessant ist das Typsystem. [DP]

Das Framework Flutter ist ein Open-Source UI-Entwicklungs-Kit von Google und basiert auf der Programmiersprache Dart. Laut Hersteller liegt der Fokus von Flutter auf kurzen Entwicklungszeiten, schneller Ausführungsgeschwindigkeit und "nativer User Experience". Der wichtige Vorteil liegt daran, dass man durch dieses Framework Applikationen für die beiden wichtigen Betriebssysteme Android und IOS mit demselben Code entwickeln kann. [F20]

Mithilfe der quelloffenen Arduino Software IDE schreibt man Anwendungen für Arduino-Boards und überspielen diese auf den Mikrocontroller.

Die Arduino Software bietet eine übersichtliche Oberfläche zum Programmieren, die sich jedoch auf die wichtigsten Funktionen beschränkt. Etwa können Kommentare und Bibliotheken eingefügt werden. Eine automatische Formatierung Vornahme ist auch möglich. Im Hilfe-Menü gibt es eine umfangreiche Anleitung und Einführung in die Software. Außerdem enthält die Installation einige Beispiele, die den Entwicklern die Entwicklung für den Mikrocontroller etwas näherbringen sollen. Neben Windows-Version ist die Arduino Software beim Hersteller auch für Linux und MacOS erhältlich. [AI20]

## Struktur der Arbeitsschritte

Die Arbeitsschritte des Projekts teilen sich in die folgenden vier Kategorien:

1. Vorbereitung und Verbindung der Hardware
2. Programmieren des Wemos-Moduls
3. Erstellung des Anwendungsprototyps hinsichtlich der funktionalen Anforderungen
4. Konvertieren des Prototyps in eine echte App
5. Integration der funktionalen Anforderung
6. Testen der App

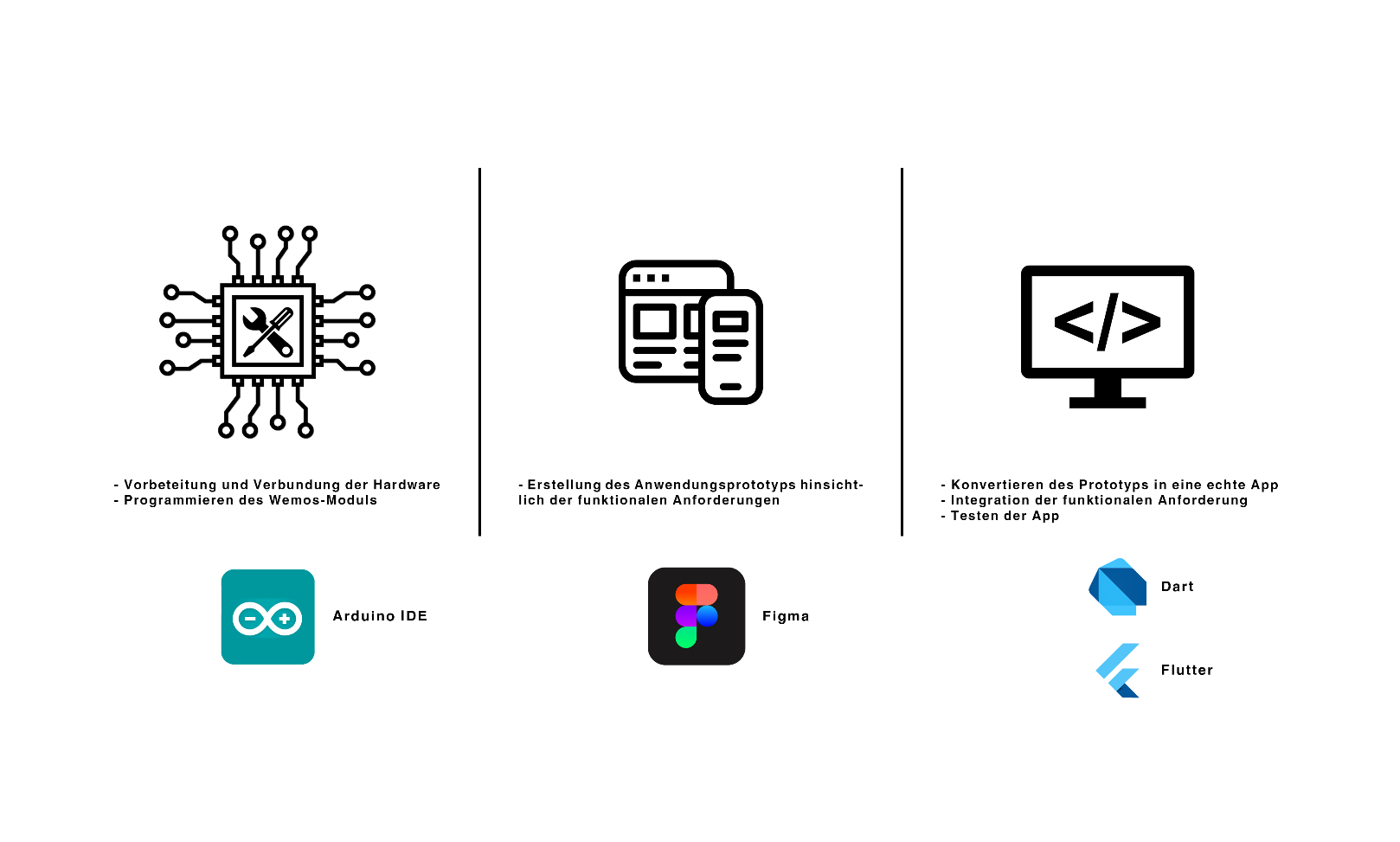


Abbildung 8: Darstellung der Arbeitsschritte

## Prototyp und Ausführen der Mobileanwendung

Um das Design der Mobilapplikation zu entworfen und einen Prototyp zu erstellen wird im Projekt der Vektorgrafikeditor bzw. Prototyping-Tool Figma benutzt. Dieses Tool hat den Vorteil, dass es nicht nur auf Windows und MacOS geeignet, sondern auch ist es webbasiertes Tool. D.h. man kann Figma sowohl online als auch offline bedienen. [FP20]

Die Idee des Designs ist es, dass jedem Wemos-Modul folgenden Designelemente zugeordnet werden:

1. Zwei Textfelder, um Topics einerseits zu abonnieren und anderseits zu veröffentlichen.
2. Eine Schaltfläche, die zum Abonnieren mit dem Topic dient.
3. Eine Schaltfläche, die zum Beenden des Abonnierens mit dem Topic dient.
4. Eine Schaltfläche, um die LED des jeweiligen Wemos-Moduls zu steuern.

Das Design dieser Elemente sieht folgendermaßen aus:



Abbildung 9: Screenshot von Prototyp

Das gesamte Programmdesign sieht so aus:

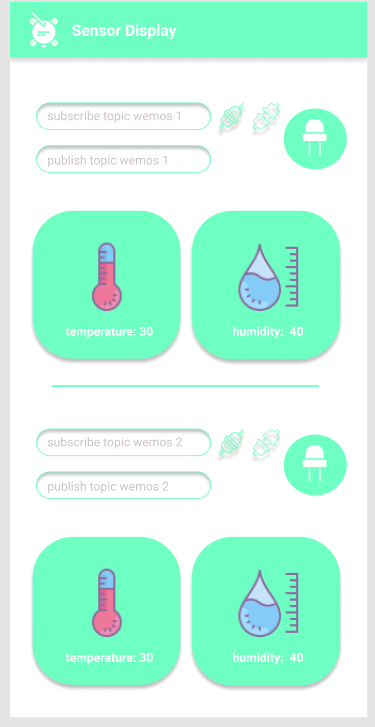


Abbildung 10: Screenshot für gesamtes Programmdesign

Die Struktur des Projekts im Figma sieht so aus:

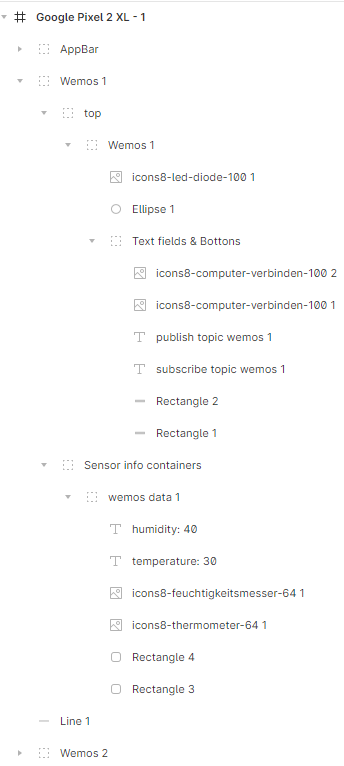


Abbildung 11: Screenshot von Struktur des Figma-Projekts

Damit man das Flutter Projekt bearbeiten oder ausführen kann, muss man die folgenden Schritte folgen:

1. Installieren Sie git, Flutter, und eine von Flutter empfohlen Editor
2. Klonen Sie das [github Repository](https://github.com/ma7amad4088/dht-sensor-data-on-flutter.git), indem Sie den Befehl im Terminal bzw. Command line eingeben:

git clone https://github.com/ma7amad4088/dht-sensor-data-on-flutter.git

1. Navigieren Sie zum „mobile\_app“ Ordner, indem Sie den Befehl im Terminal bzw. Command line eingeben:

cd dht-sensor-data-on-flutter/mobile\_app/

1. Führen Sie ein Deviceemulator aus.
2. Zum Ausführen der App schreiben Sie im Terminal bzw. Command line:

flutter run

1. Möchten Sie das Flutter-Projekt bearbeiten öffnen den „mobile\_app“ Ordner mit dem Editor.

## Entworfene Schaltung und Programmablaufplan des Projekts

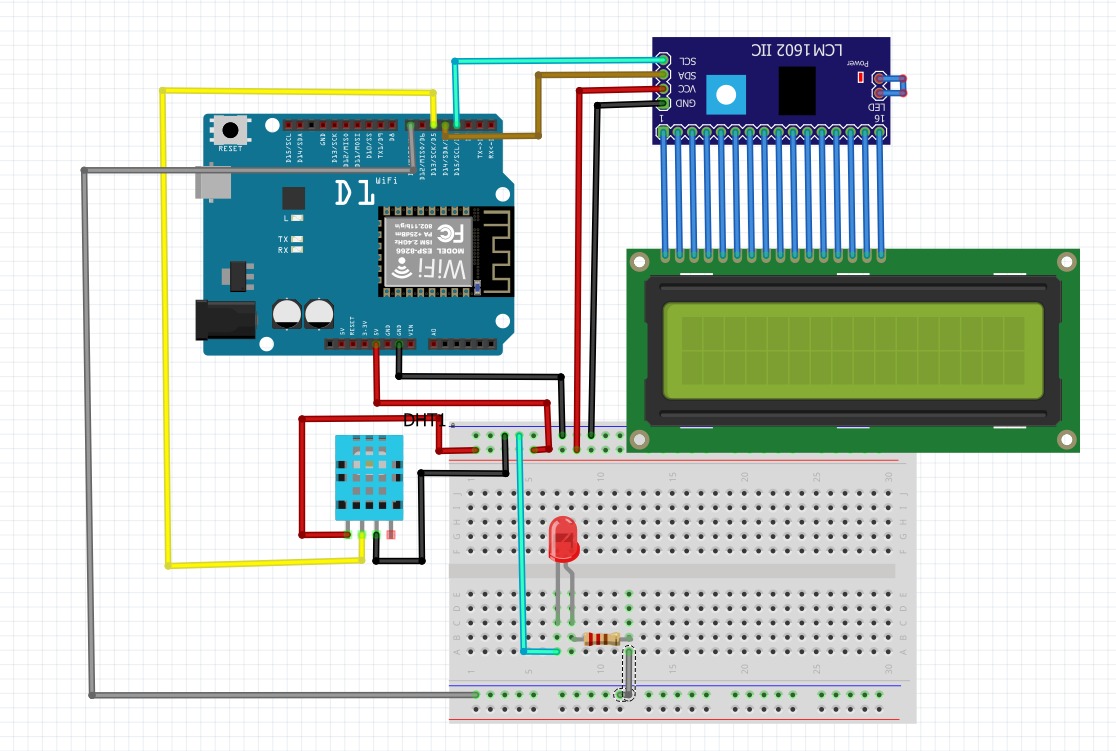


Abbildung 12: Steckplatine der gesammten Bestandteile

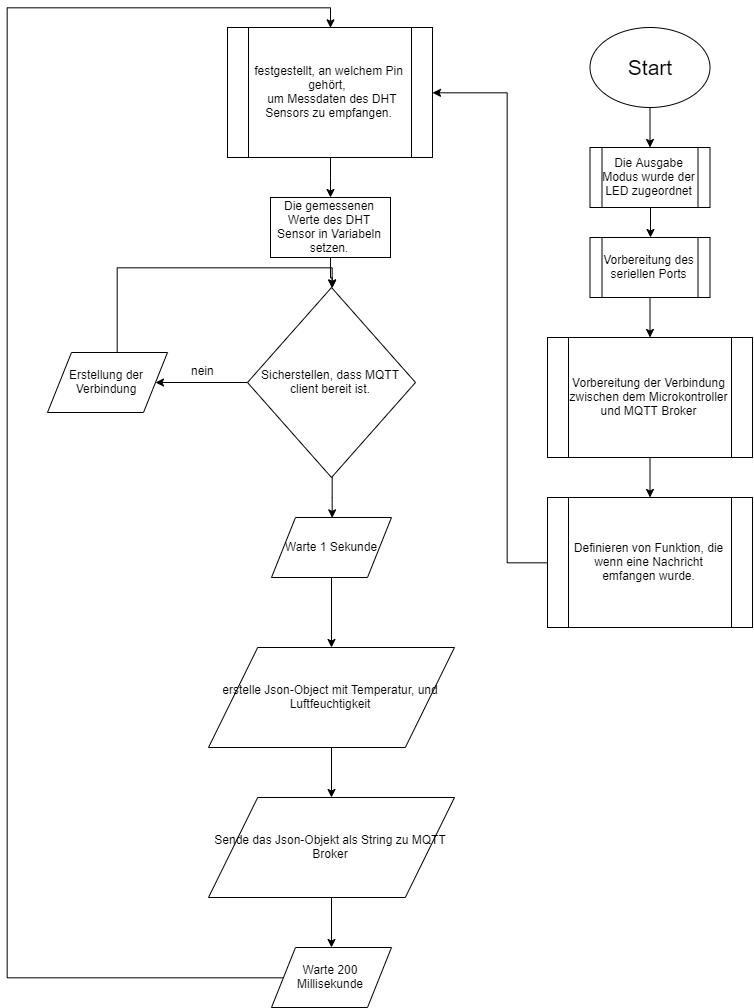


Abbildung 13: Programmablaufplan des Microcontrollers

|  |
| --- |
| „Themenspezifisches Kapitel“ |

# Zusammenfassung und Ausblick

Die Entwicklungsplatine wurde vom Arduino IDE programmiert, wobei der Temperatursensor die Werte misst und die gemessenen Werte dann an die Anwendung sendet, die sie wiederum über das MQTT Protokoll an das Empfangsgerät überträgt.

Die Anwendung wurde über die Flutter-Plattform programmiert und besteht aus einer Reihe von Bottons und Texteingabefelder.

Der Benutzer muss einige Daten eingeben, um mit dem anderen Partner zu kommunizieren, Daten an ihn zu übertragen und Daten auch von dem anderen Partner zu empfangen. Es kann auch die LED-Beleuchtung im anderen Gerät steuern.

Abbildungsverzeichnis

[Abbildung 1: Technologieschema 4](#_Toc51420264)

[Abbildung 2: MQTT Kommunikationsparadigma [MD20] 5](#_Toc51420265)

[Abbildung 3: Fotos von Wemos ESP8266 [WR20] 6](#_Toc51420266)

[Abbildung 4: Spezifikation des Wemos [WR20] 7](#_Toc51420267)

[Abbildung 5: Reelles Foto vom DHT11 Sensor [DH20] 7](#_Toc51420268)

[Abbildung 6: LCD mit I2C [LL20] 8](#_Toc51420269)

[Abbildung 7: Steckplatine von LED und Widerstand 8](#_Toc51420270)

[Abbildung 8: Darstellung der Arbeitsschritte 9](#_Toc51420271)

[Abbildung 9: Screenshot von Prototyp 10](#_Toc51420272)

[Abbildung 10: Screenshot für gesamtes Programmdesign 11](#_Toc51420273)

[Abbildung 11: Screenshot von Struktur des Figma-Projekts 12](#_Toc51420274)

[Abbildung 12: Steckplatine der gesammten Bestandteile 13](#_Toc51420275)

[Abbildung 13: Programmablaufplan des Microcontrollers 14](#_Toc51420276)

Tabellenverzeichnis

[Tabelle 1: Qualitätsanforderung 3](#_Toc51420277)

[Tabelle 2: Vorteile & Schwierigkeiten der Lösungsmöglichkeit 4](#_Toc51420278)

Literaturverzeichnis

|  |  |
| --- | --- |
| [WM20] | Wikipedia. MQTT. Online Quelle. <https://de.wikipedia.org/wiki/MQTT>, 2020.09.13. |
| [HM20] | HiveMQ helps companies connect devices to the Internet. MQTT The Messaging and Data Exchange Protocol of the IoT. Online Quelle. <https://www.hivemq.com/mqtt/>, 2020.09.13. |
| [MD20] | doubleSlash. MQTT für Dummies. Online Quelle. <https://blog.doubleslash.de/mqtt-fuer-dummies/>, 2020.09.13. |
| [WR20] | Technik Blog. WEMOS D1 – Arduino UNO kompatibles Board mit ESP8266 Chip. Online Quelle. <https://draeger-it.blog/wemos-d1-arduino-uno-kompatibles-board-mit-esp8266-chip/> 2020.09.16 |
| [DH20] | circuits4you. Interfacing DHT11 with NodeMCU Example. Online Quelle. <https://circuits4you.com/2019/01/25/interfacing-dht11-with-nodemcu-example/> 2020.09.16 |
| [LL20] | mercado libre. Display Pantalla Lcd 16x2 Arduino Con Interfaz Adaptador I2c. Online Quelle <https://articulo.mercadolibre.com.mx/MLM-607822603-display-pantalla-lcd-16x2-arduino-con-interfaz-adaptador-i2c-\_JM#position=7&type=item&tracking\_id=08dc0795-4b80-4136-ab47-0e3994f54a87> 2020.09.16 |
| [DP20] | Dart. Homepage. Online Quelle. <https://dart.dev/>, 2020.09.13. |
| [F20] | Flutter. Homepage. Online Quelle. <https://flutter.dev/ >, 2020.09.13. |
| [AI20] | Arduino. What is Arduino?. Online Quelle. <https://www.arduino.cc/en/Guide/Introduction>, 2020.09.13. |
| [FP20] | Wikipedia. Figma (Software). Online Quelle. <https://en.wikipedia.org/wiki/Figma\_(software)>, 2020.09.16. |