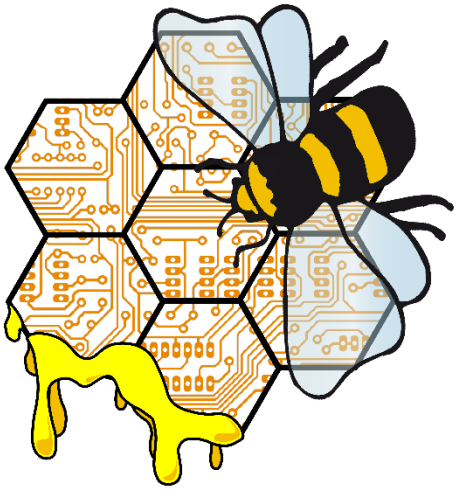
Dokumentation der Projektarbeit im Rahmen der Weiterbildung zum Staatlich geprüften Techniker



Planung und Realisierung einer Mikrocontroller gesteuerten Stockwaage.

Verfasser:	Frank Maximilian Döring
	David Saric
	Frederic von Rüden
Klasse:	FEV18
Betreuende Lehrer/innen:	Frau Riedel
	Herr Gummlich
Projektort:	Berufskolleg der Stadt Bochum
	Technische Berufliche Schule 1
	Fachschule für Technik
	Fachrichtung Elektrotechnik

Bochum, 11.11.2019

Inhaltsverzeichnis

1. Abbildungsverzeichnis 6
2. Tabellenverzeichnis 8
3. Abkürzungsverzeichnis 9
4. Fremdwortverzeichnis 10
5. Projektteam 12
6. Abstract 13
7. Einleitung 14
7.1 Ist-Zustand 14
7.2 Soll-Zustand 15
7.3 Umgesetzter Zustand 15
7.4 Blockschaltbild 16
7.5 Zeitplanung 17
8 Entwicklung der Platinen 18
8.1 BME280-Platine 20
8.2 Display-Platine 21
8.3 Hauptplatine 22
9. Fertigstellung der Platinen 23
10. Komponenten 26
10.1 Mikrocontroller ESP32-DevKitC 26
10.2 Wägezelle H44A 28
10.3 AD-Wandler und Messverstärker HX711 30
10.4 Temperatursensoren 31
10.4.1 GY-21 HTU21 31
10.4.2 GY-BME280 32
10.5 GSM Modul SIM800L 33
10.6 Touchscreen 35
11. Programmieren 36
11.1 Grundlagen 36
11.1.1 Github 36
11.1.2 Quelltext 38

10.1.2.1 Include 38
11.1.2.3 Setup 39
11.1.2.4 Loop 39
11.1.2.5 Funktionen 39
11.1.2.6 Events 40
11.2 Quellcode-Bienomat 1.0 41
11.2.2 Reset-Funktion 41
11.2.3 Deep-Sleep-Modus 41
11.2.4 Aufweck-Funktionen 42
11.2.5 RTC-Speicher 43
11.2.6 Flash-Speicher 44
11.2.7 Tarierfunktion 44
11.2.8 Wiege-Funktion 45
11.2.9 SMS-Funktion 45
11.2.10 RTC-Uhrzeit 46
11.2.11 NTP-Zeitfunktion 47
11.2.12 Graphical User Interface 47
11.2.13 Anmeldung im WLAN 49
11.2.14 Datenbankfunktion 50
11.3 Datenbank 50
11.3.1 Grundlagen 50
11.3.1.1 OpenVPN 51
11.3.1.2 SSH Client 51
11.3.1.3 Debian 51
11.3.1.4 Nginx 51
11.3.1.5 MariaDB 51
11.3.1.6 phpMyAdmin 52
11.3.2 MySQL 52
11.3.3 Einrichtung der Datenbank 52
11.3.4 Erstellung der Datenbanktabelle 52
11.4 PHP 53

11.5 HTML ... 55
11.5.1 Startseite ... 56
11.5.2 Tabellarische Darstellung ... 59
11.5.3 Grafische Darstellungen ... 62
12. Messaufbauten ... 66
12.1 Labormessung Stromaufnahme ... 66
12.2 Testaufbau Realbedingung ... 67
12.3 Messaufbau Solarpanel ... 68
13. Spannungsversorgung ... 69
14. Konstruktionen ... 70
14.1 Wiegegestell ... 70
14.2 Gehäuse ... 73
14.2.1 GY-BME280 ... 74
14.2.2 Mikrocontroller mit Display ... 75
15. Messungen und Protokolle ... 78
16. Herausforderungen ... 79
16.1 Programmierung ... 79
16.1.1 Kalibrierung der Wägezelle/ HX711 ... 79
16.1.2 SIM800L ... 80
16.1.3 GUI ... 80
16.1.4 Datenbank ... 81
16.1.5 PHP/HTML ... 82
16.2 Schaltungsaufbau ... 82
16.3 Konstruktionen/ 3D-Druck ... 83
16.3.2 3D-Druck-Einstellungen ... 84
16.4 Platinen-Layouts ... 84
17. Kosten ... 85
18. Erweiterungsmöglichkeiten ... 89
19. Fazit ... 90
20. Danksagungen ... 91
21. Quellverzeichnis ... 92

22. Anhangsverzeichnis 99
23. Eigenständigkeitserklärung 100

1. Abbildungsverzeichnis

Abbildung 1 Bienenbeuten der TBS1 zu Projektbeginn 14
Abbildung 2 Fertiggestellte Stockwaage "Bienomat 1.0" 16
Abbildung 3 Blockschaltbild 16
Abbildung 4 Ausschnitt aus finalem Zeitplan 17
Abbildung 5 Schaltplan Display-Platine 18
Abbildung 6 Strombelastbarkeit der Leiterbahnen **[vgl. 4]**..... 19
Abbildung 7 Platinen-Layout BME280 20
Abbildung 8 Platinen-Layout Display-Platine 21
Abbildung 9 Platinen-Layout Hauptplatine 22
Abbildung 10 Hauptplatine 24
Abbildung 11 Display-Platine mit aufgestecktem Bildschirm..... 25
Abbildung 12 Gelötete BME-Platine 25
Abbildung 13 Anschluss der Komponenten am ESP32-DevKitC 26
Abbildung 14 ESP32-auf einem DevKitC ohne Abdeckung **[vgl. 5]** 27
Abbildung 15 Montage Beispiel der Wägezelle **[vgl. 6]** 29
Abbildung 16 Messbrücke der Wägezelle **[vgl. 6]** 30
Abbildung 17 Foto des Breakout-Boards von Sparkfun **[vgl. 7]**..... 30
Abbildung 18 GY-21-HTU21 auf dem Breadboard 32
Abbildung 19 GY-BME280 Modul auf dem Breadboard 33
Abbildung 20 SIM800L mit Löt- und PCB-Antenne **[vgl. 23]** 34
Abbildung 21 ILI9341 Touchscreen 35
Abbildung 22 Branch-Verzweigung in einem Git Repository **[vgl. 10]** 37
Abbildung 23 Einbinden von Bibliotheken 38
Abbildung 24 Beispiel für das Definieren von Einstellungen 38
Abbildung 25 Beispiel für die void setup()-Funktion 39
Abbildung 26 Beispiel für die void loop()-Funktion 39
Abbildung 27 Beispiel für das Aufrufen der Funktionen 40
Abbildung 28 Beispiel für ein Callback-Event 40
Abbildung 29 Veranschaulichung der Komponenten im Deep Sleep **[vgl. 16]** .. 42
Abbildung 30 Test der kapazitiven Aufweckfunktion 43

Abbildung 31 Beispiele für die Erzeugung von RTC-Variablen 44
Abbildung 32 Speichern des Tara-Wertes im Flash-Speicher 44
Abbildung 33 Vom ESP32 versendete Test-SMS 46
Abbildung 34 RTC-DS3231-Modul 46
Abbildung 35 Übersichtsseite mit Messwerten 48
Abbildung 36 Visualisierung des Gewichtes inkl. Laufschrift 48
Abbildung 37 Einstellungs-Seite der GUI 49
Abbildung 38 Beispiel für die Anmeldung in einem bestehendem WLAN 50
Abbildung 39 Erstellung der Datenbanktabelle 53
Abbildung 40 Zugangsdaten in Variablen sichern 54
Abbildung 41 Implementierung Zugangsdaten/Überprüfung der Variablen 54
Abbildung 42 try-Anweisung (Verbindungsaufbau/Datenübertragung) 55
Abbildung 43 catch-Anweisung (Fehlermeldung bei Verbindungsproblemen) ... 55
Abbildung 44 Erstellen der Darstellungsform 56
Abbildung 45 Erstellen des Navigationsmenüs 57
Abbildung 46 Einbinden von Text- und Bilddateien 58
Abbildung 47 Schließen der Elemente und Tags 58
Abbildung 48 Ansicht der Startseite (Webseite) 59
Abbildung 49 Ausgabe von Datum/Uhrzeit und einer Textdatei 59
Abbildung 50 Datenbankverbindung und -abfrage 60
Abbildung 51 Erstellen einer Tabelle 60
Abbildung 52 Datenübertragung in die Tabelle 61
Abbildung 53 Schließen der foreach-Schleife, Elemente und Tags 61
Abbildung 54 Ansicht der tabellarischen Darstellung (Webseite) 62
Abbildung 55 Erstellen eines Arrays, einer Datenbankverbindung und -abfrage 63
Abbildung 56 Speichern der Durchschnittwerte in einem Array 63
Abbildung 57 Beschriftung des Liniendiagramms (Script-Bereich) 64
Abbildung 58 Aufrufen des JavaScripts (Hyperlink) 64
Abbildung 59 Ansicht der grafischen Darstellung (Webseite) 65
Abbildung 60 Messaufbau der Schaltung im Elektro-Labor der TBS1 66
Abbildung 61 Messungen unter realen Bedingungen 67

Abbildung 62 Messaufbau des Solarpanels 68
Abbildung 63 Unterkonstruktion Stockwaage 01 71
Abbildung 64 CNC-gestützte Erstellung der Montagelöcher für die Wägezelle . 72
Abbildung 65 Fertig montiertes Wiegegestell inkl. Wägezelle..... 72
Abbildung 66 Bild des Gehäuses in Fuison360 73
Abbildung 67 Gehäusekomponenten des BME280-Gehäuses..... 75
Abbildung 68 Gedrucktes Gehäuseunterteil inkl. Halterung..... 76
Abbildung 69 Gestalteter Deckel des Gehäuses 77
Abbildung 70 Prüfprotokoll zur ortsveränderlichen Prüfung 78

2. Tabellenverzeichnis

Tabelle 1 Stückliste der zu lötenden Bauteile der Hauptplatine 23
Tabelle 2 Stückliste der zu lötenden Bauteile der Display-Platine 24
Tabelle 3 Stückliste der zu lötenden Bauteile der BME-Platine 25
Tabelle 4 Linienstärken in technischen Zeichnungen 70
Tabelle 5 Linienarten in technischen Zeichnungen 70
Tabelle 6 Stückliste der verwendeten Bauteile und Komponenten 85

3. Abkürzungsverzeichnis

Abkürzung	Bedeutung
APB	Advance Peripheral Bus
A/D-Wandler	Analog-Digital Wandler
AT-Befehle	attention Befehle
CPU	Central Processing Unit
DB	Datenbank
DBMS	Datenbankmanagementsystem
GPRS	General Packet Radio Service
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
MPPT	Maximum Power Point Tracking
NTP	Network Time Protocol
PHP	PHP: Hypertext Preprocessor
RTC	Real-Time Clock
SMS	Short Message Service
SRAM	Static random-access memory
SSID	Service Set Indentifier
TBS1	Technische Berufliche Schule 1
TFT_eSPI	Bibliothek für das Display ILI9341

Abkürzung	Bedeutung
URL	Uniform Ressource Locator
VPN	Virtual Private Network
WLAN	Wireless Local Area Network

4. Fremdwortverzeichnis

Fremdwort	Bedeutung
Abschwärmen	Wenn der Bienenschwarm vor hat sich zu teilen und ein Teil des Volkes den Stock verlässt, um ein neues Nest zu bauen
Access-Point	Zugangspunkt für WLAN
AutoCAD	Programm zum Erstellen von 2D- und 3D-Zeichnungen
Beute	Eine Behausung für ein Bienenvolk
Branches	Verzweigungen im Github
Breakout-Board	Leiterplatte zur Befestigung und elektrischen Verbindung von elektronischen Komponenten
Captive-Portal	Anmeldeseite im lokalen WLAN
Compiler	Übersetzt Quelltext in Prozessor Anweisungen
Fusion360	Programm zum Erstellen von 3D-Zeichungen
GET-Methode	Methode, um eine Abfrage per HTTP zu starten

Gitea	Privater Github Server
Github	Onlinedienst zur Versionsverwaltung von Software-Entwicklungsprojekte, über bereitgestellte Server
GPIO	General Purpose In- and Output
High	Zustand der GPIOS
Hohenheimer Einfachbeute	Genormte Art eines Bienenstocks
I^2C	Schnittstelle des ESP32
I^2S	Schnittstelle des ESP32
Low	Zustand der GPIOS
MAC-Adresse	Einzigartige Adresse der Netzwerkhardware
routen	Erstellen von Leiterbahnen in EAGLE
SPI	Serielle Schnittstelle des ESP32
String	Text Datentyp mit der länge
Syntax	Semantik eines Befehls/Programmes

5. Projektteam

Name:	Frank Maximilian Döring
Geb. am:	08.02.1987
Berufliche Erstausbildung:	Elektroniker für Energie- und Gebäudetechnik

Name:	David Saric
Geb. am:	24.04.1992
Berufliche Erstausbildung:	Elektroniker für Energie- und Gebäudetechnik

Name:	Frederic von Rüden
Geb. am:	15.03.1993
Berufliche Erstausbildung:	Technischer Systemplaner für elektrotechnische Systeme

6. Abstract

Background of this project, hosted by the Technical Vocational College TBS1 in Bochum, is to provide evidence on a practical way about the skills and the knowledge gathered during the vocational training to become a state certified electrical engineer. This paper reviews the project to provide a fully functional digital beehive scale to the Imker AG. With this digital beehive scale, it is possible to measure different kinds of values and in order to use them to teach young students the relations between environmental factors and animals.

A second goal of this project is to provide a possibility for the teachers of subjects like “Datenbanken“ to use real world applied informatics in their lessons.

In order to build this digital beehive scale it was necessary to become familiar with new programming languages and new applications, but also to learn about hardware programming and about databases.

Some of the objectives while building the digital beehive scale were:

- Controlling the beehive scale with an ESP32 microcontroller
- Reading the weight of the beehive
- Reading temperature, humidity and air pressure
- Sending alert SMS
- Transferring the values to a MySQL database
- Visualizing the read oft values on a website
- Implementation of a display and a graphical user interface
- Power supply by solarpower
- Custom made 3D-printed cases

7. Einleitung

Im Rahmen der Weiterbildung zum Staatlich geprüften Techniker wurde für die Imker AG der Technischen Beruflichen Schule 1 (TBS1) eine Mikrocontroller gesteuerte Stockwaage geplant und realisiert.

Der Einsatzbereich der Stockwaage besteht in der Unterstützung der Imker AG in den Bereichen aktiver Naturschutz und Umwelterziehung. Durch den effizienten und ökologisch sinnvollen Einsatz von moderner Mikroelektronik in Verbindung mit einer Energieversorgung, welche unabhängig vom Energienetz der Schule ist, wurde dieses Vorhaben realisiert.

Des Weiteren werden die durch die Stockwaage übermittelten Daten bzw. die Datenbank selbst auch im Fachbereich Informationstechnik genutzt, um als Lernträger im Unterricht zu fungieren.

7.1 Ist-Zustand

Eine Hohenheimer Einfachbeute steht aktuell auf dem Flachdach des Gebäudes der TBS1 an der Moritz-Fiege-Straße in Bochum, welche durch die Imker AG betreut wird. Aktuell wird weder das Gewicht der Beute gemessen, noch ist es Teilnehmern der Imker AG möglich, zeitnah auf Situationen mit Handlungsbedarf, wie zum Beispiel das Umfallen der Bienenbeuten oder das Abschwärmen eines Teiles des Volkes, zu reagieren. Durch die Position der Bienenbeuten auf dem Dach ist eine Versorgung mit 230 V nicht möglich.

Abbildung 1 Bienenbeuten der TBS1 zu Projektbeginn

7.2 Soll-Zustand

Am Ende der Projektphase sollen folgende Ziele realisiert sein:

- Im Zuge des Projektes soll eine Stockwaage entstehen, welche die Daten der Beute, inklusive eines Zeitstempels, in einem sinnvollen Intervall erfasst.
- Die erfassten Daten sollen in einer Datenbank zur weiteren Verwendung abgelegt werden.
- Diese Datenbank soll auf einem schuleigenen Server betrieben werden oder lokal an der Stockwaage selbst.
- Die Steuerung der Stockwaage soll mittels eines Mikrocontrollers realisiert werden.
- Die Stockwaage soll über eine Solarenergieversorgung betrieben werden.
- Es ist darauf zu achten, dass sämtliche Unterkonstruktionen und zusätzlichen Anbauten nicht den Boden der Beute bedecken, um eine ausreichende Luftzirkulation zu gewährleisten.
- Die Größe der Unterkonstruktion soll sich an der Größe der Hohenheimer Einfachbeute orientieren.
- Die Wägezelle darf sich nicht plastisch verformen bzw. muss gegebenenfalls kalibrierbar sein.
- Ein Frontpanel für die aktuelle Statusanzeige und die Ausführung von Befehlen mittels Bedienelementen am Gerät soll vorhanden sein.
- Es muss eine Alarmierungsfunktion implementiert werden, welche bei einer Situation, die ein Handeln erfordert, aktiviert wird.
- Die Möglichkeit für zukünftige Erweiterungen soll gegeben sein.
- Das gesamte Projekt soll als open-source auf dem schuleigenen Git-Repository veröffentlicht werden.

7.3 Umgesetzter Zustand

Die Mikrocontroller gesteuerte Stockwaage erwacht in einem 12-Minuten-Intervall aus dem Deep Sleep Modus und erfasst das Gewicht, die Gehäusetemperatur und die äußeren Klimawerte. Die erfassten Daten werden bei Bedarf auf einem in das Gehäuse integrierten Frontpanel angezeigt und über das Internet an ein PHP-Skript und schließlich an eine eigens für diesen Zweck eingerichtete Datenbank übermittelt. Diese wird wiederum durch ein PHP-Skript ausgelesen und auf einer Webseite (www.bees.tbs1.de/Index.php) tabellarisch und grafisch visualisiert. Das Gewicht wird kontinuierlich mit einem vorgegebenen Wert verglichen und aktiviert die Alarmfunktion, sobald dieser Wert abrupt unterschritten wird. Per Short Message Service (SMS) wird der Imker über die Gewichtsveränderung informiert, um zeitnah handeln zu können. Über das Frontpanel und die zusätzlichen Bedienelemente kann die Stockwaage bzw. der Mikrocontroller vor Ort aktiviert, bedient und tariert werden. Die Energieversorgung erfolgt über eine speziell gefertigte Blei-Säure-Batterie, die mit

einem Laderegler und einem Solarpanel verbunden ist. Somit wird die Batterie während des größten Teils des Jahres permanent aufgeladen und ist unabhängig vom herkömmlichen Stromversorgungsnetz.

Abbildung 2 Fertiggestellte Stockwaage "Bienomat 1.0"

7.4 Blockschaltbild

Das folgende Blockschalbild zeigt den grundlegenden Hardwareaufbau des Projektes und der Legende an der rechten Seite ist zu entnehmen, auf welche Art die einzelnen Komponenten miteinander kommunizieren bzw. versorgt werden.

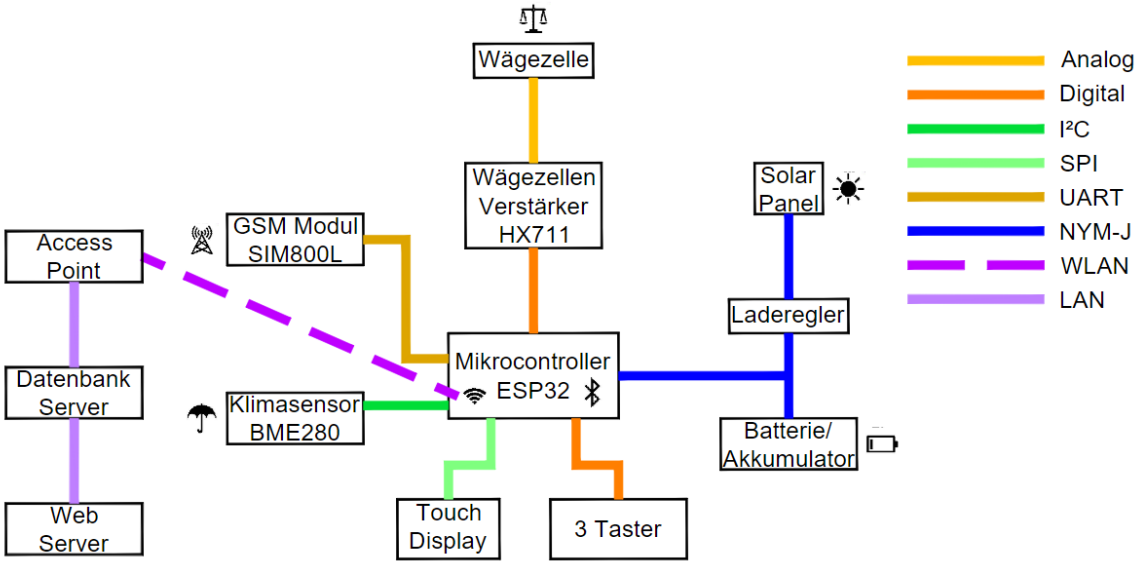


Abbildung 3 Blockschaltbild

7.5 Zeitplanung

Teil der ersten Phase des Projektes war, neben der Erstellung eines Lasten- und Pflichtenheftes, auch die Erstellung des Zeitplans. Dieser wurde mit dem Programm Microsoft Project 2017 erstellt und während des Projektverlaufs kontinuierlich angepasst und erweitert. Durch die in Kapitel 14 erwähnten Herausforderungen haben manche Projektabschnitte länger gedauert als zunächst geplant. Doch durch großzügig eingeplante Pufferzeiten im Anfangsplan war es trotzdem möglich, eine fristgerechte Übergabe des Projektes zu erreichen.

Da der gedruckte Zeitplan in der Größe DIN A0 vorliegt, befindet er sich im Anhang.

Nr.		Vorg	Vorgangsname	Dauer	Anfang	Ende
1			**Projektrealisierung**	**60 Tage**	**Mit 28.08.19**	**Die 19.11.19**
2			**Planung**	**14 Tage**	**Mit 28.08.19**	**Mon 16.09.19**
3			Anfangsplanung (Organisatorisches)	3 Tage	Mit 28.08.19	Fre 30.08.19
4			Recherche Komponenten	9 Tage	Mon 02.09.19	Don 12.09.19
5			Erstellung Lastenheft	9 Tage	Mon 02.09.19	Don 12.09.19
6			Erstellung Pflichtenheft	9 Tage	Mon 02.09.19	Don 12.09.19
7			Erstellung technischer Zeichnung/Recherche (Unterkonstruktion)	11 Tage	Mon 02.09.19	Mon 16.09.19
8			Erstellen einer Topologie (Mikrocontroller)	2 Tage	Die 10.09.19	Mit 11.09.19
9	✓		1. Zwischenbericht	0 Tage	Don 12.09.19	Don 12.09.19
10			Überarbeitung Lasten- & Pflichtenheft	3 Tage	Don 12.09.19	Mon 16.09.19
11			**Montage/Einrichtung/Inbetriebnahme**	**42 Tage**	**Fre 13.09.19**	**Son 10.11.19**
12			**Unterkonstruktion**	**8 Tage**	**Mon 16.09.19**	**Mit 25.09.19**
13			Lieferung & Anfertigung Träger/Rahmen	5 Tage	Mon 16.09.19	Fre 20.09.19
14			Anbringung & Inbetriebnahme der Wägezelle	0 Tage	Mon 23.09.19	Mon 23.09.19
15			**ESP32 Programmierung**	**34 Tage**	**Mon 16.09.19**	**Don 31.10.19**
16			Einarbeitung Github	1 Tag	Mon 16.09.19	Mon 16.09.19
17			Verbindung Wägezelle/AD-Wandler/ESP32	3 Tage	Mon 16.09.19	Mit 18.09.19
18			Kalibrierung + Softwarefunktion	4 Tage	Don 19.09.19	Die 24.09.19

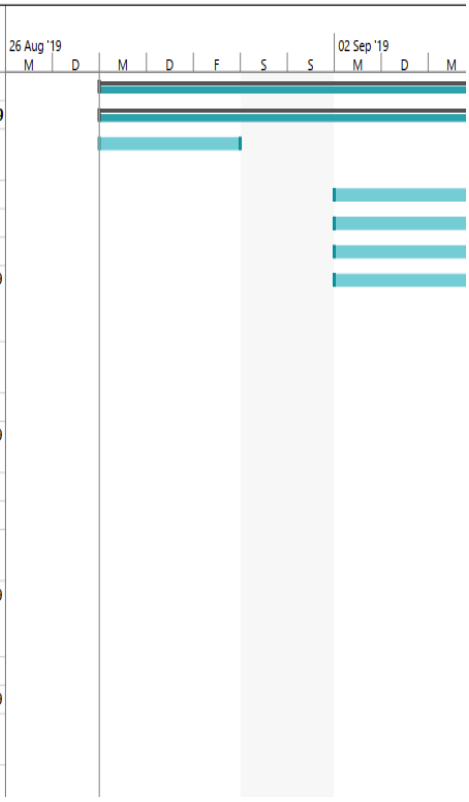


Abbildung 4 Ausschnitt aus finalem Zeitplan

8 Entwicklung der Platinen

Für das Stockwaagen-Projekt war es notwendig, mehrere Platinen anzufertigen. Die Planung erfolgte mithilfe des Programmes *EAGLE* von Autodesk.

Zuerst wurde die *Schematic* Datei angelegt, welche die komplette Schaltung mit allen Bauteilen enthält. Daraus wurde anschließend das Layout erzeugt. Um etwaige Fehler im Schaltplan zu finden, wurde das Werkzeug *Schaltplan prüfen* verwendet. Dieses zeigt in einem gesonderten Fenster die Fehler sowie mögliche Lösungen an.

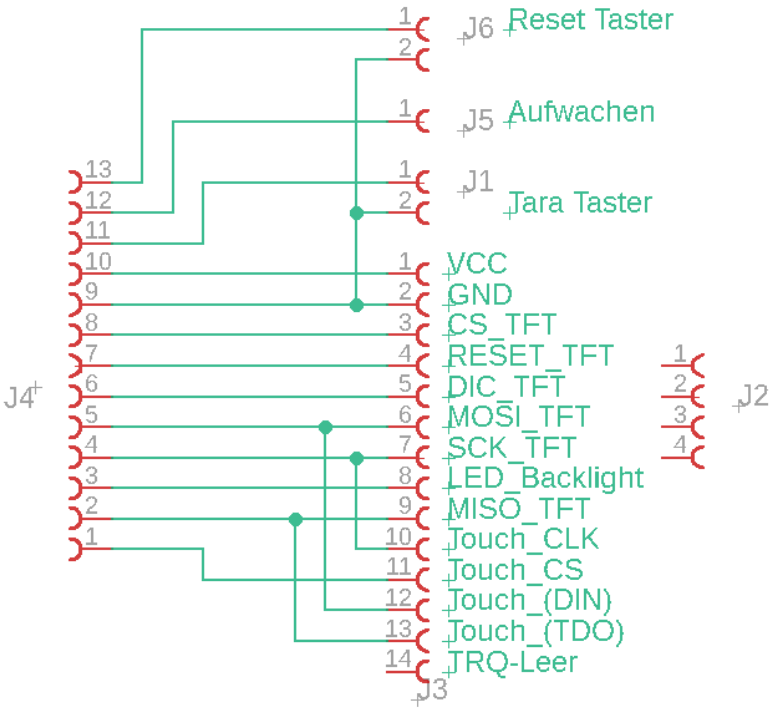


Abbildung 5 Schaltplan Display-Platine

Es wurde eine kleine Platine für den Temperatursensor BME280, eine Platine für das Display mit Abgängen für Taster und eine Hauptplatine für die restlichen Komponenten, wie z.B. den ESP32, benötigt.

Um ein möglichst leichtes Austauschen der einzelnen Komponenten zu ermöglichen, wurden diese alle auf Buchsen-Halter gesteckt. Somit ist ein einfaches Austauschen der Bauteile möglich, falls nötig.

Nach der Überprüfung und Beseitigung von Fehlern konnte das Board erzeugt werden. Die verwendeten Bauteile mussten noch in eine sinnvolle Position gebracht werden, da *EAGLE* diese nur am Rand platziert. Anschließend werden die *Airwaires* geroutet.

Bei der Platzierung der Bauteile wurde darauf geachtet, dass die Leiterbahnen möglichst nicht in einem 90-Grad-Winkel aufeinandertreffen, sondern vorteilhafterweise in einem Winkel von 45 Grad. Durch den 45-Grad-Winkel ist es wahrscheinlicher, dass keine Probleme beim Ätzen auftreten.

Die Leiterbahnbreite der Leiterbahnen wurde ebenfalls an die Strombelastbarkeit angepasst, sodass der Stromkreis des SIM800L-Moduls eine Leiterbahnbreite von 1,016 mm benötigt und die anderen Komponenten sonst 0,6096 mm.

Das SIM800L-Modul benötigt zwischenzeitlich bis zu 2 A wodurch eine breitere Leiterbahn benötigt wird.

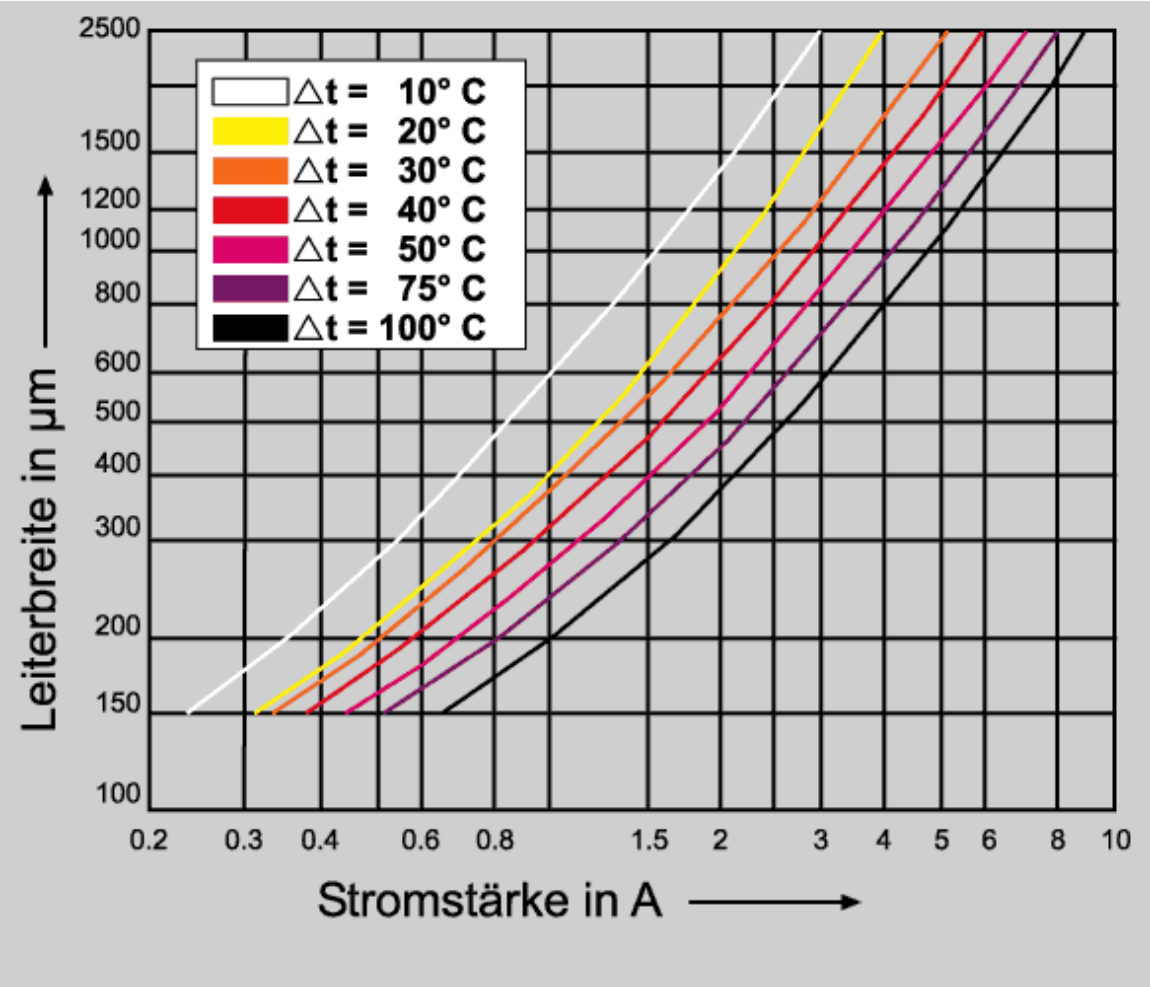


Abbildung 6 Strombelastbarkeit der Leiterbahnen ***[vgl. 4]***

Die Platinen wurden extern zum Ätzen vergeben, da die Platinen bei dem gewählten Hersteller auch professionell verlackt werden und somit einen weiteren Schutz gegen Umwelteinflüsse erhalten.

8.1 BME280-Platine

Die Platine für den BME280 musste ausreichend Platz bieten für eine Schraubklemme und einen Buchsen-Halter, in diesem Buchsen-Halter fand der BME280 Platz. Die Schraubklemme dient der Verbindung der BME280-Platine und der Hauplatine, um so ein Übertragen der Messwerte zu ermöglichen. Des Weiteren wurden vier Schraublöcher zum Befestigen der Platine am Gehäuse geplant. Der Stromlaufplan befindet sich im Anhang.

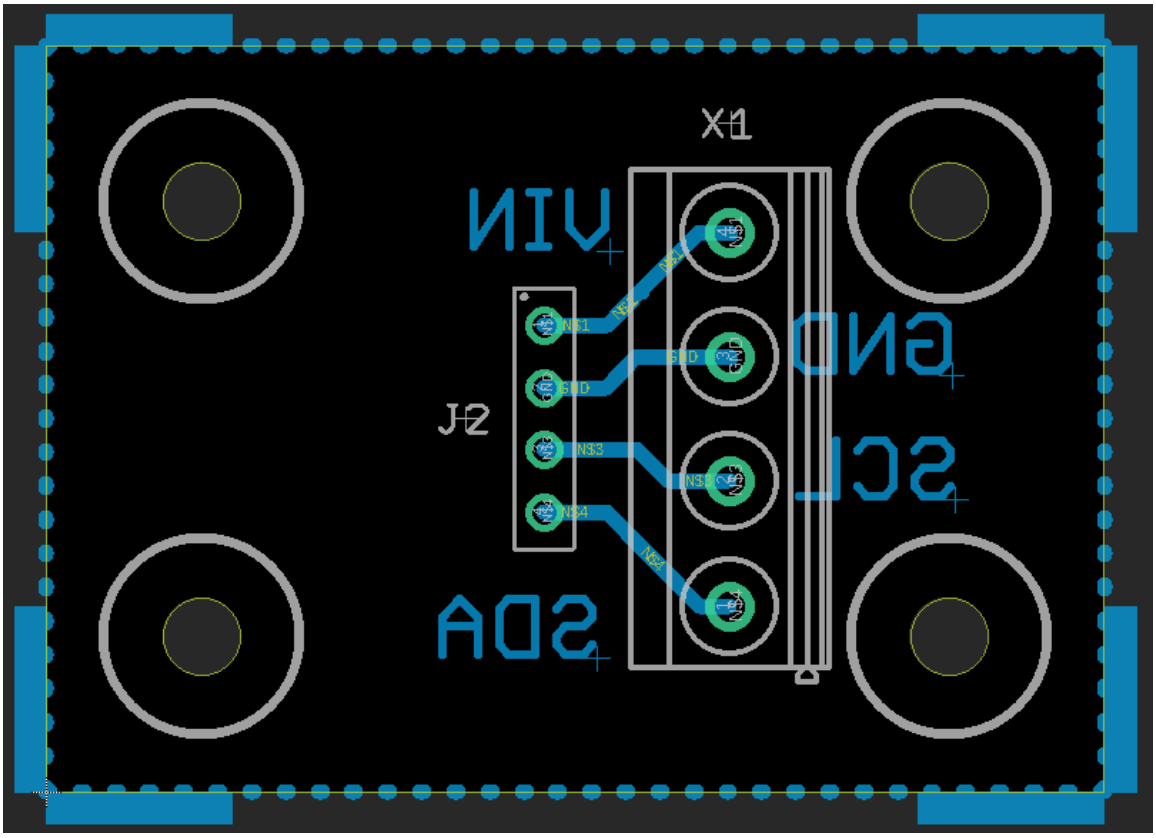


Abbildung 7 Platinen-Layout BME280

8.2 Display-Platine

Auf der Display-Platine wurde eine Stiftleiste für eine Zuleitung vorgesehen, da diese Platine zwar von der Mikrocontroller-Platine versorgt, aber am Gehäuse-Deckel befestigt wurde. Die Stiftleiste benötigt 14 Pins, drei Pins sind für die beiden Taster und den Aufwach-Button vorgesehen und wurden auf einzelne Stiftleisten verteilt. Das Display benötigt einen 14-Pin-Buchsen-Halter sowie einen 4-Pin-Buchsen-Halter. Der 4-Pin-Buchsen-Halter dient hierbei nur zur Befestigung und Stabilisierung des Displays.

Auf der Display-Platine wurden mehrere Stiftleisten für das Display und die Taster vorgesehen sowie Buchsen-Halter für das Display. Eine Stiftleiste wurde für die Kommunikations- und Spannungsversorgung der Display-Platine vorgesehen. Ein 14-Pin-Buchsen-Halter sowie ein 4-Pin-Buchsen-Halter wurden als Halterung für das Display auf der Platine geplant, für die Taster und den Aufwach-Button wurden entsprechende Stiftleisten (2x2 Pin und 1x1 Pin) vorgesehen.

Die Display-Platine erhielt vier Schraublöcher zum Befestigen am Gehäusedeckel. Der Stromlaufplan befindet sich im Anhang.

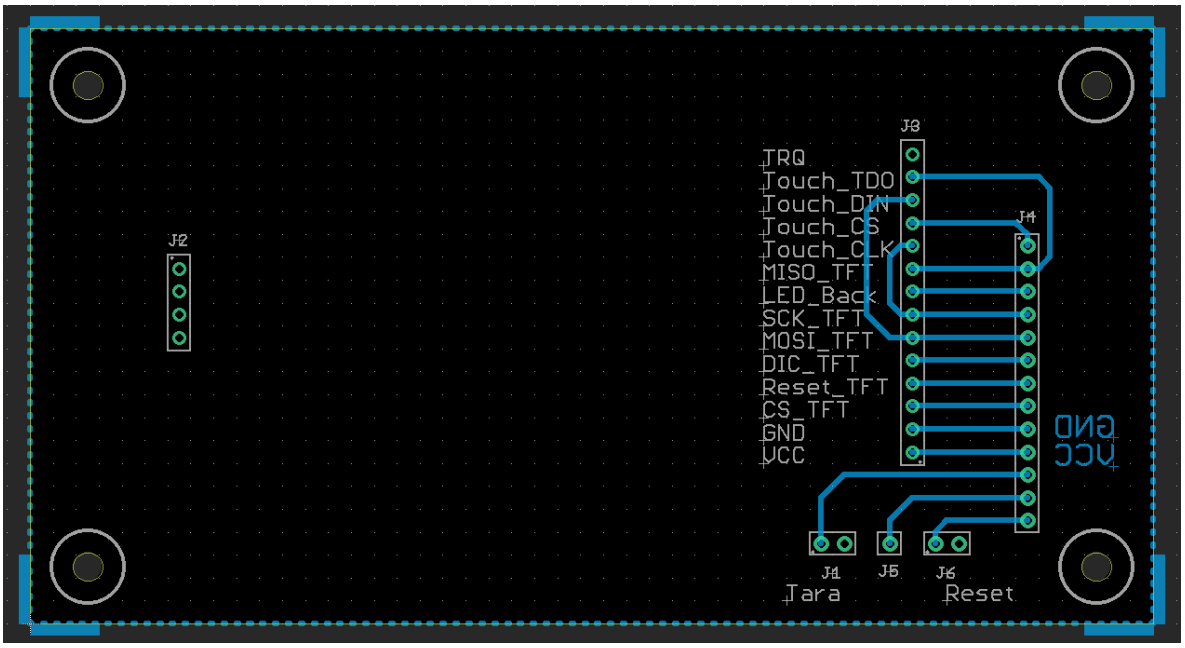


Abbildung 8 Platinen-Layout Display-Platine

8.3 Hauptplatine

Die Hauptplatine ist die größte der drei Platinen, darauf fanden alle benötigten Komponenten in Buchsen-Haltern ihren Platz, außerdem Festspannungsregler, Widerstände, Relais, Freilaufdioden, eine Stiftleiste für die Versorgung der Display-Platine, Schraubklemmen für die Spannungsversorgung, den BME280 und die Leitung der Wägezelle, ein Kondensator und drei Transistoren.

Aufgrund der Pin-Belegung des ESP32 und der Komponenten war es nicht möglich, nur den *BOTTOM-Layer* zu verwenden. An entsprechenden Stellen wurden Durchkontaktierungen gesetzt, um über den *TOP-Layer* zu routen.

Insgesamt wurden für die Hauptplatine fünf Schraublöcher vorgesehen, um die Platine mit dem Gehäuse zu verbinden. Somit ist auch eine gewisse Druckfestigkeit beim Einsetzen der Komponenten auf die dafür vorgesehenen Sockel gegeben.

Zudem war es der Projektgruppe wichtig, auf die Hauptplatine das eigens für das Projekt erstellte Logo sowie Projektteilnehmer und Projektjahr drucken zu lassen. Der Stromlaufplan befindet sich im Anhang.

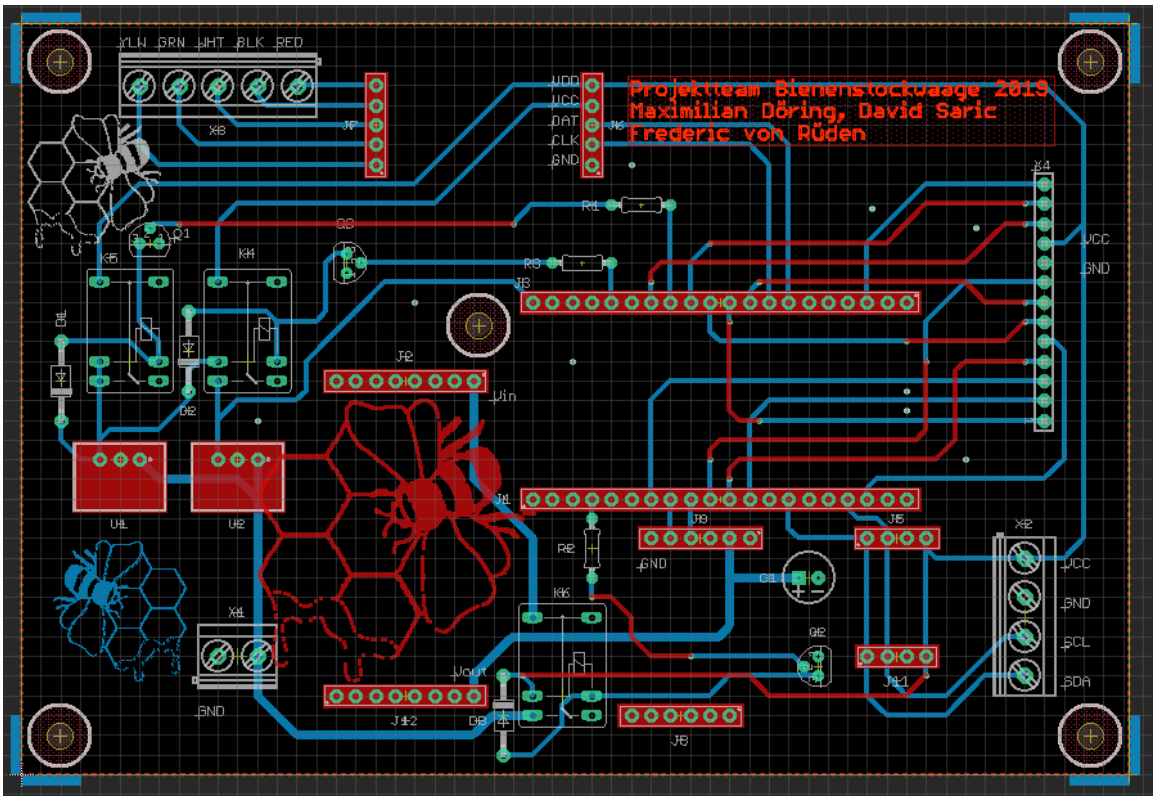


Abbildung 9 Platinen-Layout Hauptplatine

9. Fertigstellung der Platinen

Nachdem alle Platinen und Bauteile vorhanden waren, wurde mit der Fertigstellung der Platinen begonnen. Für das Fertigstellen der Platine sind zwei Schritte erforderlich:

- Bestücken der Platine
- Einlöten der Bauteile

Um einen reibungslosen Lötprozess zu gewährleisten, wurde vor dem Bestücken die Platine mit einer Alkohol-Lösung abgerieben. Jedes Bauteil wurde einzeln gesteckt und anschließend gelötet. Dabei wurde darauf geachtet, dass alle Bauteile wie im Schaltplan bzw. im *EAGLE*-Layout platziert wurden.

Tabelle 1 Stückliste der zu lötenden Bauteile der Hauptplatine

Bauteile der Hauptplatine	**Anzahl**
Festspannungsregler 3,3 V	1
Festspannungsregler 5 V	1
Transistor CBC547B	3
Schottky-Diode	3
Relais	3
Kondensator 100 µF	1
Widerstände 1,5 kΩ	3
Schraubklemme 2-fach	1
Schraubklemme 4-fach	1
Schraubklemme 5-fach	1
Buchsen-Halter 4 Pins	2
Buchsen-Halter 5 Pins	2
Buchsen-Halter 8 Pins	2
Buchsen-Halter 20 Pins	2
Stiftleiste 14 Pins	1
DC-DC Step-Down-Converter	1

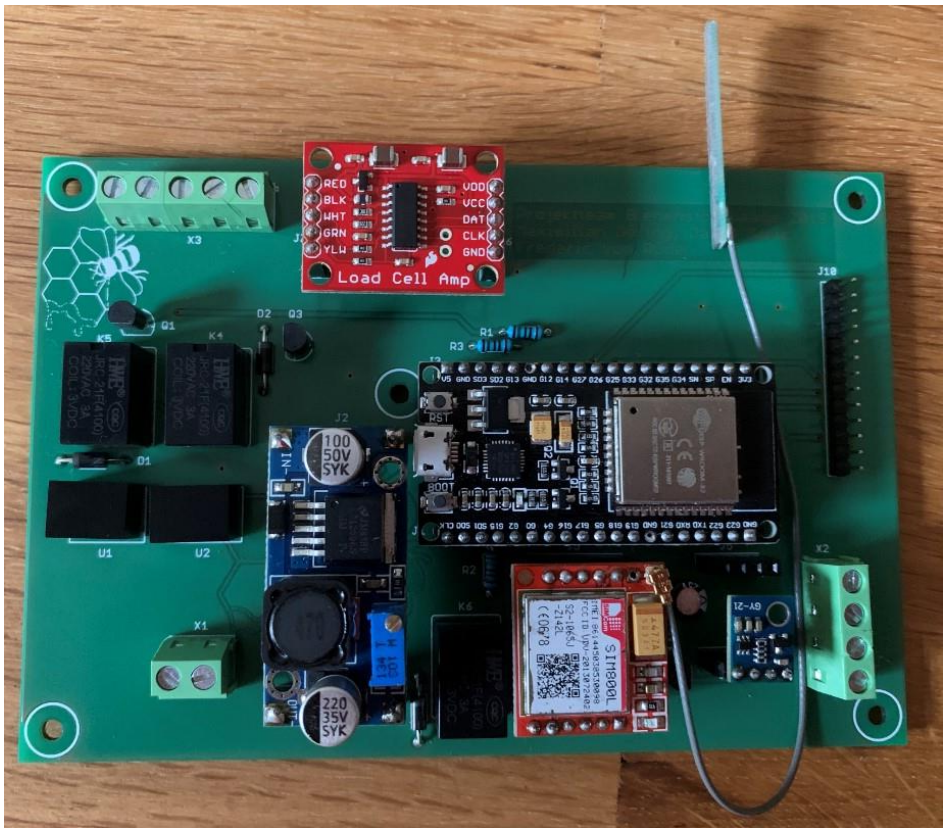


Abbildung 10 Hauptplatine

Tabelle 2 Stückliste der zu lötenden Bauteile der Display-Platine

Bauteile der Display-Platine	**Anzahl**
Buchsen-Halter 4 Pins	1
Buchsen-Halter 14 Pins	1
Stiftleiste 1 Pins	1
Stiftleiste 2 Pins	2
Stiftleiste 14 Pins	1

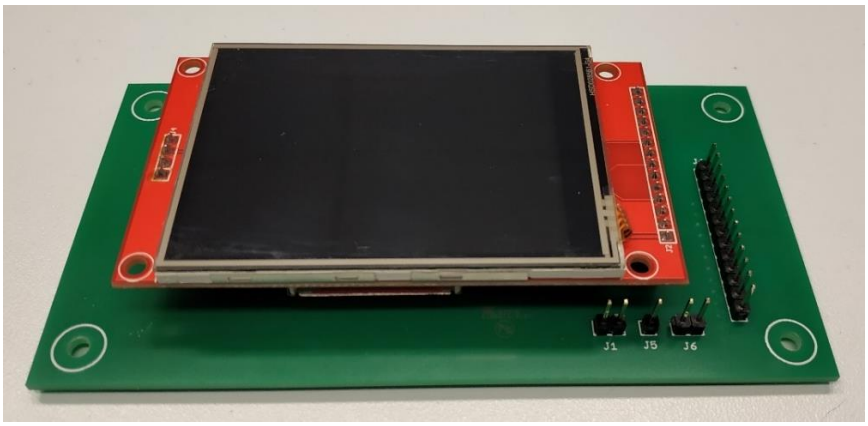


Abbildung 11 Display-Platine mit aufgestecktem Bildschirm

Tabelle 3 Stückliste der zu lötenden Bauteile der BME-Platine

Bauteile der BME-Platine	**Anzahl**
Schraubklemme 4-fach	1
Buchsen-Halter 4 Pins	1

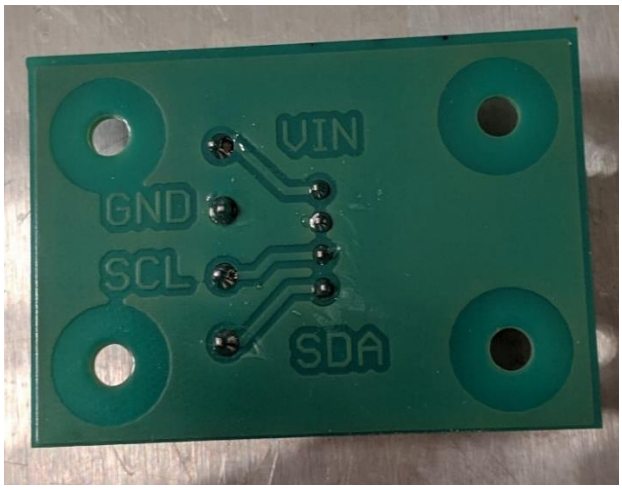


Abbildung 12 Gelötete BME-Platine

10. Komponenten

Für das Projekt wurden verschiedene Komponenten aufgrund ihrer, für das Projekt wichtigen Spezifikationen, welche in diesem Kapitel näher erläutert werden, ausgewählt. Nachfolgend wird die Verbindung der einzelnen Komponenten mit dem ESP32-DevKitC veranschaulicht:

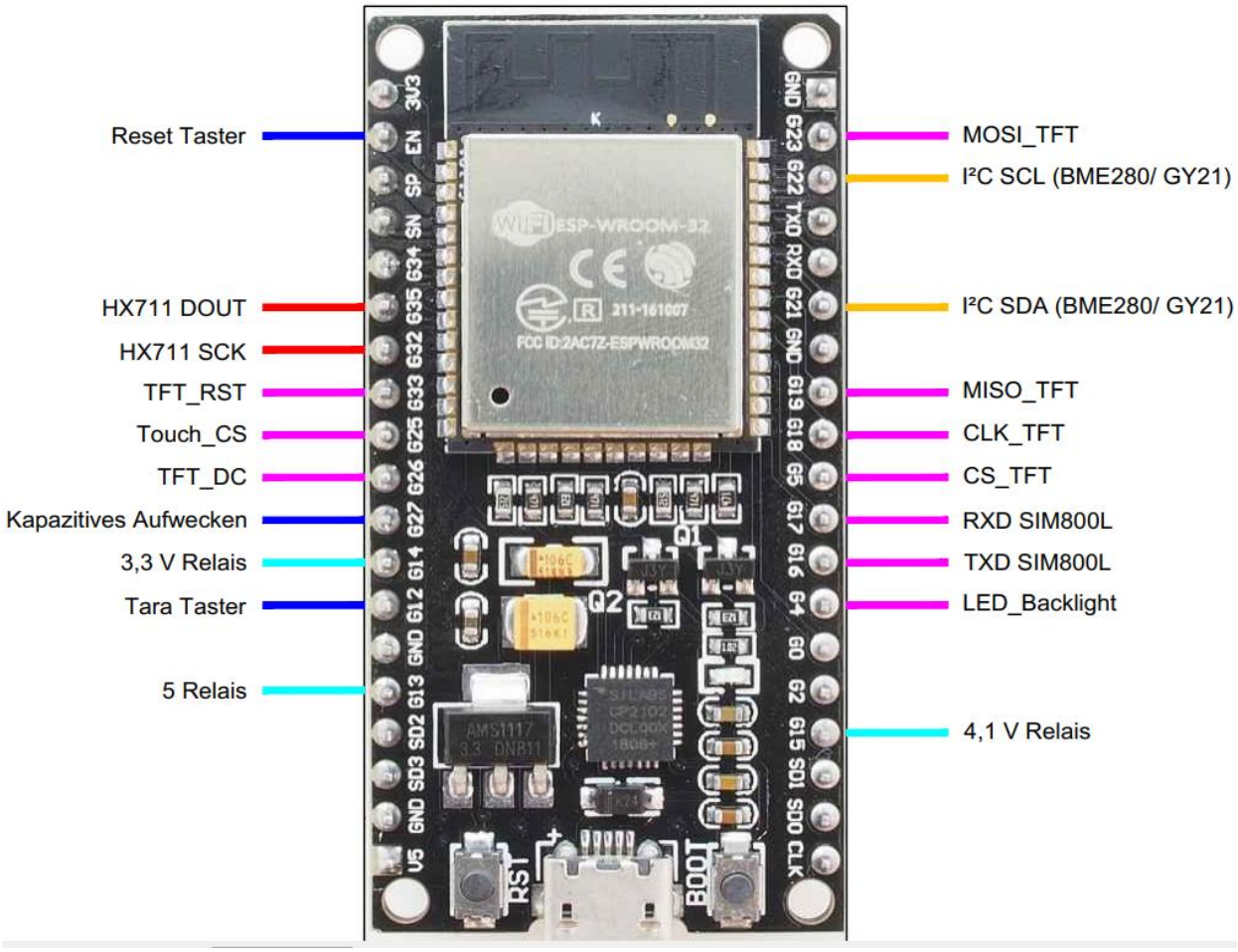


Abbildung 13 Anschluss der Komponenten am ESP32-DevKitC

10.1 Mikrocontroller ESP32-DevKitC

ESP32 ist die Bezeichnung des Mikrocontrollers, welcher die Aufgaben zur Steuerung der Stockwaage übernimmt. Er wird seit 2016 von der Firma Espressif aus Shanghai hergestellt. **[vgl. 1]**

Der ESP32 ist das Herzstück der im Projekt verwendeten Entwicklerplatine ESP32-DevKitC. Es existieren Entwicklerplatinen in verschiedenen Ausführungen, häufig unterscheiden sie sich in der Anzahl der nutzbaren GPIO-Pins und der Ausführung des Prozessors, welcher zum Beispiel als Single-Core- und als Dual-Core-Ausführung erhältlich ist. Auch die jeweilige Stromaufnahme der

einzelnen Entwicklerplatinen unterscheidet sich stark, teilweise sogar bei gleicher Art der Entwicklerplatine und unterschiedlichem Hersteller. Daher wurde nach den später beschriebenen Labormessungen der ESP32-DevKitC vom Auftragshersteller und Verkäufer AZ-Delivery zur Steuerung der Stockwaage ausgewählt.

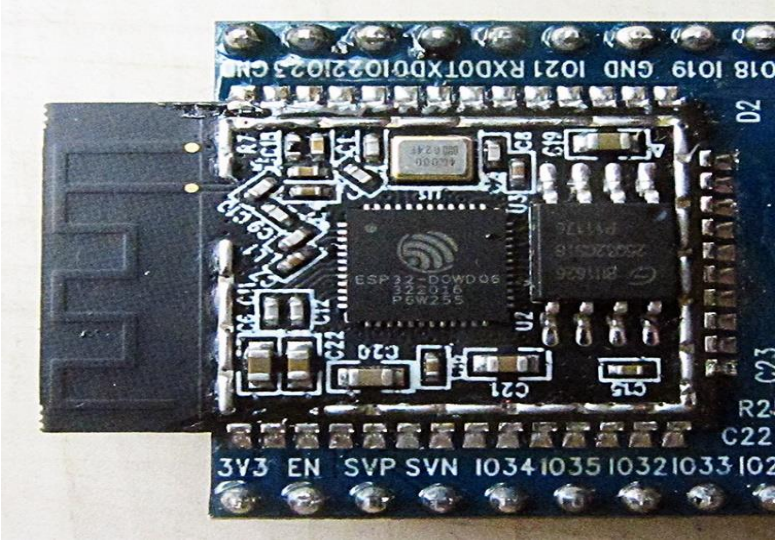


Abbildung 14 ESP32-auf einem DevKitC ohne Abdeckung ***[vgl. 5]***

Hauptmerkmale der im Projekt verwendeten Entwicklerplatine ESP32-DevKitC sind: **[vgl. 101]**

- Tensilica Xtensa LX6 Dual-Core Mikroprozessor (Taktgeschwindigkeit 240 MHz)
- 16 MB Flash-Speicher
- 520 KB SRAM
- 2,4 GHz WLAN mit 802.11 b/g/n Standard
- Bluetooth 4.2, inklusive Bluetooth Low Energy
- CP2102 USB/ Seriell-Interface
- 3,3 Volt Logiklevel
- sehr geringer Stromverbrauch
- 18 Analog/Digital-Konverter (ADC) Kanäle
- 3 SPI-Schnittstellen
- 3 UART-Schnittstellen
- 2 I²C-Schnittstellen
- 16 PWM-Ausgänge
- 2 Digital/Analog-Konverter (DAC) Kanäle
- 2 I²S-Schnittstellen
- 10 GPIOs mit kapazitiver Abtastung
- Maximale Stromstärke 15mA pro GPIO **[vgl. 3]**

Im Vergleich zu den meisten anderen Mikrocontrollern zeichnet sich der ESP32 durch eine sehr hohe CPU Geschwindigkeit, einen großzügig bemessenen Flash-Speicher und die verwendete 3,3 Volt Schalt-Logik aus. Auch die Möglichkeit den ESP32 in verschiedene Stromsparmodi zu versetzen trug maßgeblich zur Auswahl hinzu. Außerdem sind die WLAN- und Bluetooth-Verbindungsmöglichkeiten des ESP32 von großem Nutzen. Bedingt durch die Tatsache, dass diese bereits integriert sind, werden keine extra GSM- oder Bluetoothshields für diese Funktionen benötigt. Programmiert wird der ESP32 mittels der Arduino IDE, welche eine Form der Programmiersprache C, inklusive C++ Syntax-Erweiterungen, nutzt. Außerdem kann der ESP32 mittels der hauseigenen Toolchain der Firma Espressif, der Espressif IDF, programmiert werden. Neben der in der Arduino IDE verwendeten Programmiersprache sind hier auch die Sprachen „MicroPython“ und „Lua“ zur Programmierung verwendbar. **[vgl. 1]**

10.2 Wägezelle H44A

Die ausgesuchte Wägezelle H44A stammt vom Hersteller Bosche und ist bis zu 150 kg geeignet. Die Wägezelle ist für mittlere und größere Plattformen bis zu einer Größe von 600 x 800 mm geeignet und passt daher zu der Hohenheimer Einfachbeute. **[vgl. 6, vgl.108]**

Die Wägezelle besteht aus Aluminium und ist mit vier Dehnungsmessstreifen ausgestattet, welche vollständig vergossen sind, somit erfüllt die Wägezelle IP65. Die Zelle muss mit 5 - 12 V versorgt werden und arbeitet in einem Nenntemperaturbereich von -10 - +40°C zuverlässig. Die Wägezelle ist für den Langzeiteinsatz geeignet und somit optimal für das Projekt. Durch die 7000 Teilungseinheiten lässt sich eine Messabstand errechnen.

$$x = \frac{Nennlast\ der\ gewählten\ Wägezelle}{Teilungseinheiten}$$

$$x = \frac{150kg}{7000} = 0{,}021kg$$

Für die Berechnung der Nennlast der Wägezelle stellt die Firma Bosche eine Formel zur Verfügung:

$$E_{max} = \left(\frac{Max + DL}{N} \times Fa\right)$$

$$E_{max} = \left(\frac{70kg + 10\ kg}{1} \times 1{,}3\right)$$

$$E_{max} = 104kg$$

- E_{max} = Nennlast der Wägezelle
- Max = Nennlast der Waage
- DL = Totlast/ Vorlast
- Fa = Der Faktor für Eckenlastzuschlag für eine mögliche Ungleichmäßigkeit der Lastverteilung (1,3 als Standard) **[vgl. 102]**

Montiert wurde die Zelle an dem Wägegestell, welches eigens vorgesehene Montagelöcher aufweist.

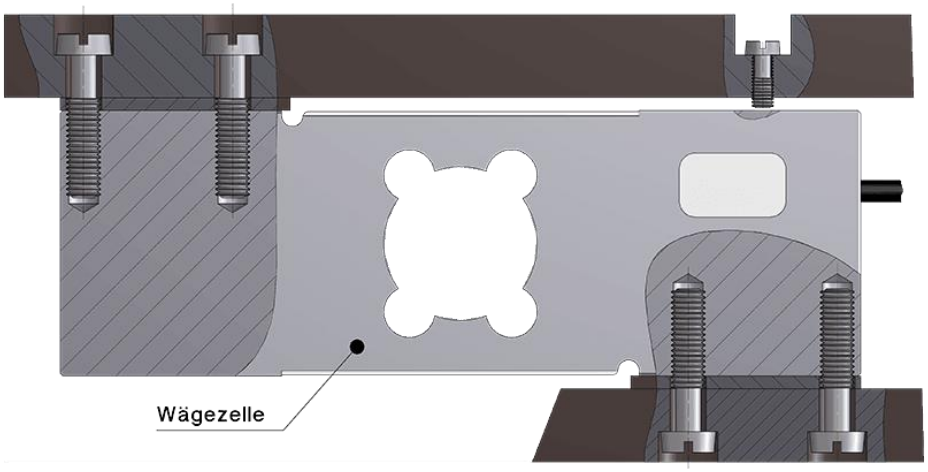


Abbildung 15 Montage Beispiel der Wägezelle ***[vgl. 6]***

Das Kabel der Wägezelle besitzt fünf Adern, die farblich gekennzeichnet sind, Versorgung rot und schwarz, Signal grün und weiß sowie eine Schirmleitung lila. Bis auf die Schirmleitung wurde alles an den AD-Wandler angeschlossen, diese soll für genauere Messwerte an Ground angeschlossen werden. In dem Testaufbau hat dies zu starken Schwankungen der Messwerte geführt, weswegen darauf verzichtet wird, die Schirmleitung an Ground anzuschließen.

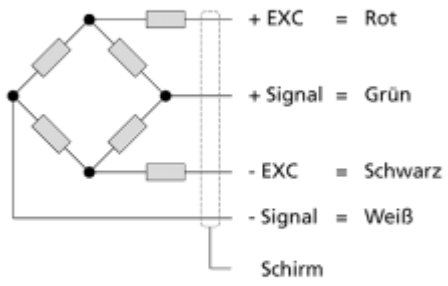


Abbildung 16 Messbrücke der Wägezelle ***[vgl. 6]***

10.3 AD-Wandler und Messverstärker HX711

Um die Widerstandsmessbrücke der Wägezelle auswerten zu können, wird ein HX711 benötigt, dieser enthält analoge Brückenverstärker sowie einen hochauflösenden 24-Bit-A/D-Wandler.

Die prinzipielle Beschaltung wird von diversen Anbietern als Breakout-Board angeboten, für das Projekt wird das Breakout-Board von Sparkfun verwendet. Nach ausführlicher Recherche weist der HX711 von Sparkfun nur geringe Toleranzen der Messwerte und Messfehler auf und ist außerdem qualitativ sehr hochwertig bezüglich der Verarbeitung. Des Weiteren war auch die ausführliche Dokumentation des Moduls ein ausschlaggebender Faktor für die Auswahl.

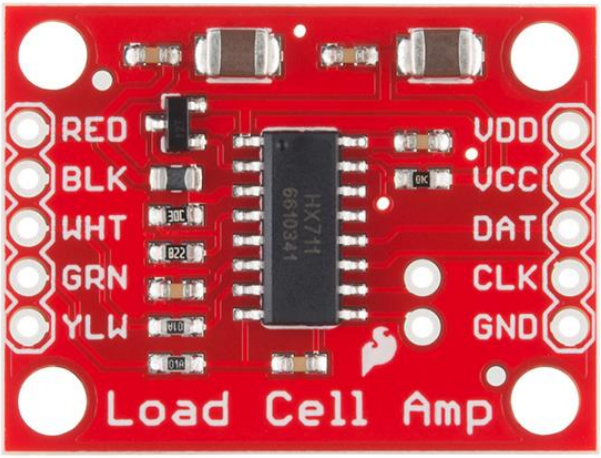


Abbildung 17 Foto des Breakout-Boards von Sparkfun ***[vgl. 7]***

Die Anschlüsse für die Wägezelle sind auf dem HX711 bereits farblich angegeben, RED für Rot, BLK für Schwarz, WHT für Weiß, GRN für Grün und YLW für Gelb. Diese entsprechen der herkömmlichen Farbcodierung der Wägezellen. Bei der verwendeten Wägezelle von Bosche ist die Schirmleitung nicht gelb, sondern lila.

Auf der anderen Seite des HX711 werden die Stromversorgung des Digitalteils (VDD = 3,3 V), die Stromversorgung des Analogteils (VCC = 5 V), Daten (DAT), Takt (CLK) sowie Masse (GND) angeschlossen. Sofern der Microcontroller mit 3,3 V versorgt wird, muss der HX711 mit 5 V und 3,3 V verbunden sein. Ansonsten ist es meist ausreichend, den HX711 mit 3,3 V zu versorgen. Dabei ist darauf zu achten, dass VDD und VCC miteinander verbunden sind. **[vgl. 8]**

Um genaue Messwerte zu erhalten, ist es notwendig, im Quellcode des Mikrocontrollers einen entsprechenden Skalierungsfaktor anzugeben. Dieser Vorgang wird im Verlauf der Dokumentation noch näher erläutert.

10.4 Temperatursensoren

Temperatursensoren sind meist elektrische oder elektronische Komponenten, die es in verschiedene Ausführungen gibt und Anwendung in zahllosen Bereichen finden. Um damit die Temperatur zu ermitteln, wird ein elektrisches Signal übermittelt, das als Maß für die Temperatur gilt. Für das Projekt werden integrierte Umgebungssensoren genutzt, da diese im Bereich der Sensoren-Technologie eine Vielzahl neuer und leistungsfähiger Anwendungen unterstützen sowie platzsparender und auch leistungsfähiger sind. Die je nach Sensor analog oder digital erfassten Werte werden über den seriellen Bus, in diesem Fall I²C, übertragen.

10.4.1 GY-21 HTU21

Der GY-21 HTU21 ist ein zuverlässiger und langlebiger Sensor, der die Temperatur und Luftfeuchte ermittelt. Dieser befindet sich auf der Hauptplatine und dient lediglich zur Überwachung der Gehäusetemperatur.

Durch seine Eigenschaften, wie z.B. eine niedrige Versorgungsspannung und Stromaufnahme, die Ermittlung von zuverlässigen Messergebnissen, eine einfache Ansteuerung über die I²C-Schnittstelle und Bibliotheken, die ESP32 kompatibel sind, war der Sensor die ideale Wahl für dieses Projekt.

Neben der nicht benötigten Erfassung des Luftdrucks war ein weiterer Grund für die Auswahl des GY-21-HTU21, dass die Verwendung von zwei baugleichen Sensoren zu Problemen führen kann. Da jeder Sensor eine vom Hersteller festgelegte I²C Adresse hat, würde es beim Anschließen von zwei baugleichen Sensoren mit der gleichen Adresse zu Adress-Konflikten auf dem I²C-Bus führen, welche nur durch einen I²C-Multiplexer behoben werden können.

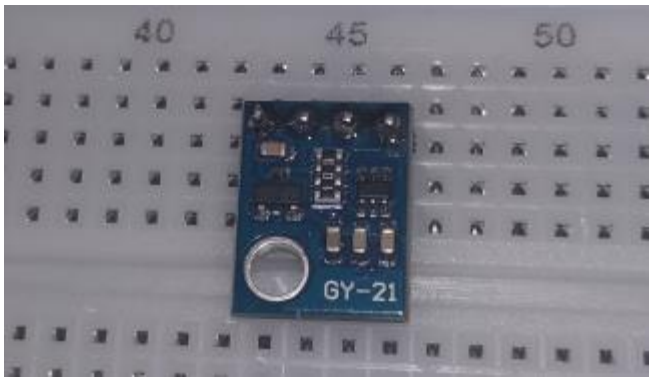


Abbildung 18 GY-21-HTU21 auf dem Breadboard

Technische Eigenschaften **[vgl. 2]**:

- Betriebsspannung: 1,5 - 3,6 V
- Stromaufnahme: 300 - 500 µA
- Messgenauigkeit Temperatur: +/-1°C Abweichung bei 30°C - 90°C
- Messgenauigkeit Feuchtigkeit: +/-2 % RL bei 5 % - 95 % RL
- Art der Ausgabe: digital
- Datenschnittstellen: I²C

10.4.2 GY-BME280

Der GY-BME280 ist im Vergleich zum GY-21 HTU21 der leistungsfähigere Temperatursensor und fungiert als Messwertgeber im Außenbereich der Stockwaage. Der Sensor befindet sich allein auf einer separaten Platine und in einem eigenen Gehäuse, um die äußeren Klimawerte zu ermitteln.

Dieser entspricht ideal den Anforderungen des Projekts und zeichnet sich aus durch seine besonderen Eigenschaften:

- eine hohe Auflösung und präzise Messfunktion
- die einfache Ansteuerung über die I²C-Schnittstelle
- eine niedrige Betriebsspannung und Stromaufnahme
- Kompatibilität mit dem ESP32

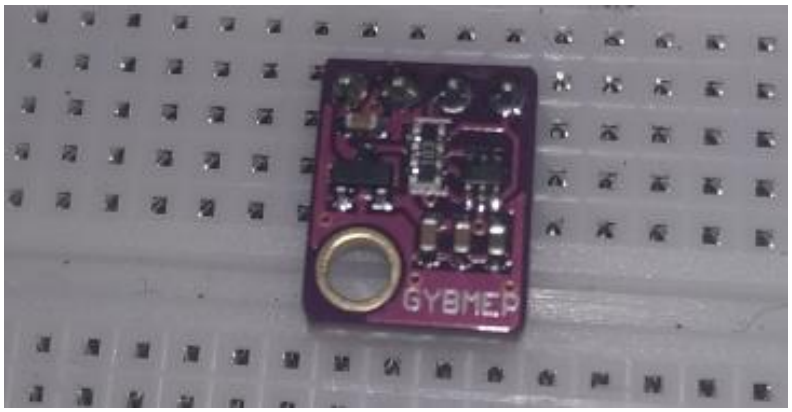


Abbildung 19 GY-BME280 Modul auf dem Breadboard

Technische Eigenschaften:

- Betriebsspannung: 1,7 – 3,6V
- Stromaufnahme: 340 - 714 µA
- Betriebstemperatur: -40 °C bis +85 °C
- Absolute Temperaturgenauigkeit: ± 0,5 °C bei 25 °C
- Genauigkeit der Luftfeuchtigkeitsmessung: ± 3 % der relativen Luftfeuchte
- Absolute Genauigkeit bei Druckmessungen: typ. ± 1hPa nach dem Einlöten
- Art der Ausgabe: digital
- Datenschnittstellen: I²C

10.5 GSM Modul SIM800L

Das SIM800L ist ein GSM-Modul mit einer mitgelieferten Antenne zum Anlöten und einer optionalen Antenne mit IPX-kompatiblem Stecker, welches sich in GSM- und GPRS-Netze einwählen kann.

Das Modul hat in diesem Projekt die Funktion, eine SMS zu Alarmierungszwecken an eine hinterlegte Handynummer zu versenden. Die SMS dient dazu, dem Imker unverzüglich mitzuteilen, ob z.B. ein Umfallen bzw. Abschwärmen des Bienenvolkes vorliegt. So wird sichergestellt, dass der Imker zeitnah auf Situationen mit Handlungsbedarf reagieren kann.

*Abbildung 20 SIM800L mit Löt- und PCB-Antenne **[vgl. 23]***

Eine direkte Spannungsversorgung ist über den ESP32 5V-Ausgang ohne Spannungsregler nicht möglich, da die vom ESP32 ausgegebenen 5 V die Spezifikationen der Versorgungsspannung des SIM800L übersteigen. Außerdem kann das SIM800L-Modul kurzzeitige Spitzenströme von bis zu 2 A benötigen **[vgl. 7,103]**. Aus diesem Grund wurde es in der umgesetzten Schaltung durch einen LM2596S DC-DC Tiefsetzsteller eingespeist, welcher im Bedarfsfall per Relais geschaltet wird. Außerdem wird die Eingangsspannung am SIM800L durch einen hinzugefügten 100 µF-Kondensator stabilisiert, um einen Verbindungsverlust durch Spannungseinbrüche entgegenzuwirken.

Über eine der drei seriellen UART-Schnittstellen des ESP32 kann mit dem Modul über AT-Befehle kommuniziert werden. So sind z.B. die Abfrage der Verbindungsqualität, des Netzbetreibers oder auch des generellen Zustands möglich.

Technische Eigenschaften **[vgl. 103]**:

- Versorgungsspannung: 3,4 - 4,4 V
- Frequenz: 850/900/1800/1900MHz
- Betriebstemperatur: -40 – 85 °C
- Unterstützt 2G GSM- und GPRS-Daten
- Steuerung mit AT-Befehlen

Das SIM800L Modul bietet durch seinen kleinen Formfaktor einen entscheidenden Vorteil im Gegensatz zu einem GSM-Shield. Auch die ausführlichen, vom Hersteller zur Verfügung gestellten Datenblätter des SIM800L sowie die umfangreiche, für den ESP32 angepasste Bibliothek TinyGsm sind Gründe, die mit zur Auswahl beitrugen.

10.6 Touchscreen

Um eine zusätzliche Unterstützung des Umwelterziehungsauftrags der Imker AG zu ermöglichen, besitzt die Stockwaage einen Touchscreen, auf dem das Graphical-User-Interface (GUI) dargestellt wird. Der Bildschirm zeigt die zuletzt aufgenommenen Messwerte an, zeigt eine Visualisierung des aktuellen Gewichtes und erlaubt eine Bedienung durch Berührung des Bildschirms über seine resistive Berührungsfunktion.

Verwendet wurde das SPI Display ILI9341, inkl. Touch-Funktion.

Technische Eigenschaften: **[vgl. 106]**

- Bildschirmgröße: 2,8 Zoll
- Displaytyp: TFT
- Auflösung: 320*240 Pixel
- Displayfarbraum: RGB 65000 Farben
- Schnittstelle: SPI-Bus oder parallel (je nach Funktion 3-10 Adern)
- Betriebstemperatur -20 °C – 70 °C
- Stromverbrauch: 40 mA – 90 mA im aktiven Zustand

Abbildung 21 ILI9341 Touchscreen

Der Bildschirm, sowie das Touchpad darauf, wurde per SPI an den ESP32 angeschlossen. Im Gegensatz zur Anbindung des Bildschirms über eine parallele Schnittstelle ist die Bildwiederholungsrate, bedingt durch die in den Konfigurationsdateien der Bibliothek hinterlegte SPI-Bus-Geschwindigkeit von 26.67 MHz zwar langsamer, aber man benötigt weniger GPIOs zum Anschluss. Da keine Video-Inhalte auf dem Bildschirm dargestellt werden sollen, reichen die bei der Anbindung über SPI erreichbaren 10 - 20 Bilder pro Sekunde.

Da der permanente Betrieb des Displays sehr zur Laufzeitverkürzung der Energieversorgung der Stockwaage beitragen würde, wird es nur durch gezielte Interaktion des Benutzers mit dem kapazitiven Aufweckknopf der Stockwaage aktiviert.

Angesteuert wird das Display über die für den ESP32 angepasste TFT_eSPI-Bibliothek, welche auf GitHub unter der GPL-Lizenz zur Verfügung gestellt wird **[vgl. 24].** Da das Display über einen separaten Anschluss für die Hintergrundbeleuchtung verfügt, wird auch diese nur im Bedarfsfall vom ESP32 geschaltet, um so eine gute Lesbarkeit des Displays zu gewährleisten.

Die ILI9341 Bildschirmplatine besitzt einen zusätzlichen SD-Karten-Leser, welcher seine Spannung vom Bildschirm bezieht. Die verwendete Bibliothek als auch die Spannungsversorgung ließen eine Verwendung in diesem Projekt aber nicht zu.

Das Display wird im separat konstruierten und erstellten Gehäuse für die Stockwaage montiert und mittels einer Dichtung versiegelt. So wird sichergestellt, dass über das Display keine Feuchtigkeit in das Gehäuse gelangen kann.

11. Programmieren

Kernaufgabe des Projektes war die Programmierung des Mikrocontrollers, um die verschiedenen an die Stockwaage gestellten Anforderungen zu erfüllen.

11.1 Grundlagen

Um das Verständnis für die verschiedenen Kernbereiche der Mikrocontroller-programmierung zu ermöglichen, werden im nachfolgenden einige Grundlagen erläutert.

11.1.1 Github

Github ist eine Online-Versionsverwaltung für Software-Projekte. Für das Projekt wurde der schuleigene Gitea-Server verwendet. Auf diesem wurde ein eigenes Repository angelegt, welches den gesamten Quellcode in Branches speichert. Die Branches sind so angelegt, dass jede Komponente ihren eigenen Ast besitzt. In dem Master-Branch wurden nachher alle Quellcodes der Branches zusammengefügt. **[vgl. 9]**

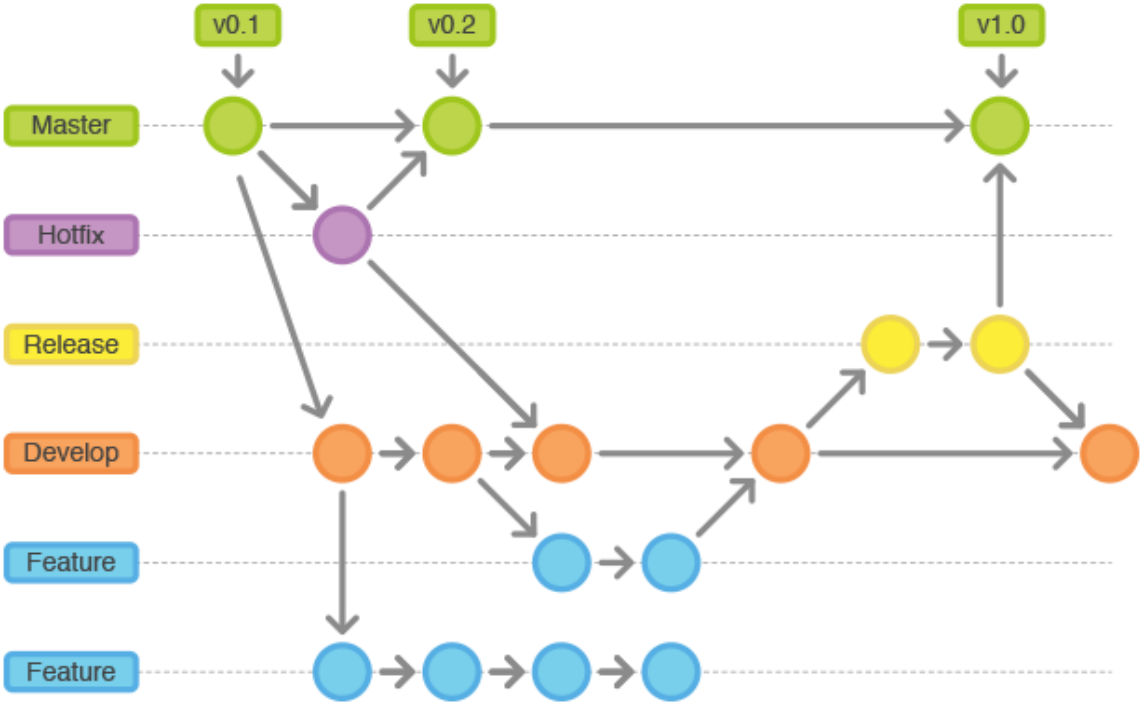


Abbildung 22 Branch-Verzweigung in einem Git Repository ***[vgl. 10]***

11.1.2 Quelltext

Der Quelltext beinhaltet den kompletten Programmcode, welcher von der Arduino IDE mittels eines Compilers kompiliert und anschließend an den ESP32 übertragen wird. Wichtig zu wissen ist es, dass der Quellcode durch den Mikroprozessor des ESP32 sequenziell verarbeitet wird.

Der dem Projekt zu Grunde liegende Quelltext lässt sich grob in folgende verschiedene Sequenzen unterteilen:

- Include
- Define
- Setup
- Loop
- Funktionen
- Events

10.1.2.1 Include

Zu Beginn des Quelltextes müssen die zur Kompilierung notwendigen Bibliotheken angegeben werden. Ohne sie lässt sich der Quelltext nicht kompilieren, da im Quelltext verwendete Hardwarefunktionen ohne ihre entsprechenden Bibliotheken nicht funktionieren können.

Dies geschieht durch folgende Syntax, welche in der Abbildung beispielhaft die Einbindung verschiedener Bibliotheken zeigt.

```
#include <TinyGsmClient.h>      // Bibliothek fuer des SIM800L
#include <WiFi.h>               // Bibliothek fuer das Benutzen der WLAN Funktion des ESP32
#include <TFT_eSPI.h>           // Bibliothek zur Ansteuerung des Touchdisplays via SPI
```

Abbildung 23 Einbinden von Bibliotheken

11.1.2.2 Define

Define ist eine Syntax, welche aus der Programmiersprache C kommt und auch in der Arduino IDE seine Verwendung findet. Meist werden die für die korrekte Abarbeitung eines Quellcodes benötigten Konstanten, wie z.B. Pin-Belegungen oder festgelegte Werte, in diesen Bereich des Quelltextes geschrieben.

Beispiel für das Definieren des SIM800L als GSM-Modul in der TinyGsm-Bibliothek:

```
#define TINY_GSM_MODEM_SIM800   // Auswählen des SIM800L Moduls in der TinyGsm Bibliothek
#define RX2 16    // Pin Define für GSM Modul SIM800L
#define TX2 17    // Pin Define für GSM Modul SIM800L
```

Abbildung 24 Beispiel für das Definieren von Einstellungen

11.1.2.3 Setup

Die *setup()-Funktion* erscheint vor der *void loop*() Funktion und wird zu Beginn eines Programmstarts als Erstes ausgeführt. Sie wird genutzt, um Befehle einmalig auszuführen, wie z.B. die Pin-Zuordnung oder das Einbinden von Bibliotheken.

Beispiel für die *void setup*()-Funktion ist die Verwendung des Pins als Eingang oder Ausgang festzulegen.

```
void setup()                        // Die Setup-Funktion wird nur einmal ausgeführt !
{
  pinMode(LED_BUILTIN, OUTPUT);    // Der digitale Pin LED_BUILTIN wird als Ausgang festgelegt.
}
```

Abbildung 25 Beispiel für die void setup()-Funktion

11.1.2.4 Loop

Die *void loop*()-Funktion erscheint nach der *void setup*()-Funktion und wird als Endlosschleife dargestellt. Sie wird für das Hauptprogramm genutzt, damit die Befehle permanent wiederholt werden.

Beispiel für die *void loop*()-Funktion ist die Zustands-Zuordnung der GPIOs aus der *void setup*()- Funktion. Im Beispiel wurden sie auf *HIGH* und *LOW* gesetzt

```
void loop()                         // Die Loop-Schleife wird immer wieder Ausgeführt !
{
  digitalWrite(LED_BUILTIN, HIGH);  // Die LED erhält den Zustand HIGH.
  delay(1000);                      // Ein zeitlicher Verzug von 1 Sekunde.
  digitalWrite(LED_BUILTIN, LOW);   // Die LED erhält den Zustand LOW.
  delay(1000);                      // Ein zeitlicher Verzug von 1 Sekunde.
}
```

Abbildung 26 Beispiel für die void loop()-Funktion

11.1.2.5 Funktionen

Funktionen werden in verschiedenen Programmiersprachen verwendet, um den Quellcode zu strukturieren. Funktionen können immer wieder aufgerufen werden, ohne den Inhalt der Funktion neu schreiben zu müssen. Somit bleibt der Quellcode übersichtlich. Funktionen, die mit *void* beginnen, haben keinen konkreten Rückgabewert.

```
void setup() {

  Serial.println("Starte Funktion");

  delay(500);
  klima();       // Aufruf der Funktion zur Messung von Temperatur,Luftfeuchtigkeit und Luftdruck
  button();      // Aufruf der Tara-Button-Funktion
  wiegen();      // Aufruf der Wiege-Funktion
  datenbank();   // Aufruf der WLAN und Datenbank Funktion

}
```

Abbildung 27 Beispiel für das Aufrufen der Funktionen

11.1.2.6 Events

Durch sogenannte *events* ist es möglich, über die Berührungsfunktion des Displays mit dem ESP32 zu interagieren. Die durch die GUI verwendeten Objekte lassen sich so nutzen, um zuvor im Quelltext definierte Aktion auszuführen.

Die Events werden bei der Stockwaage genutzt, um den *Deep Sleep-Modus* zu aktivieren oder um die für die SMS-Alarmierung hinterlegte Handynummer zu ändern.

```
// Mittels dieser Event-Funktion wird generell erfasst ob ein Buttonobjekt beruehrt wurde.
// Sobald dies geschieht wird ueber den seriellen Monitor die entsprechende Meldung ausgegeben.
static void event_handler(lv_obj_t * obj, lv_event_t event)
{
    if(event == LV_EVENT_CLICKED) {
        printf("Clicked\n");
    }
```

Abbildung 28 Beispiel für ein Callback-Event

11.2 Quellcode-Bienomat 1.0

Ein großer Teil der Arbeiten an der projektintern *Bienomat* getauften Stockwaage war die Programmierung der Hardware und ihrer Funktionen. Die nachfolgenden Unterkapitel geben einen Überblick über die für den *Bienomaten* realisierten Programm-Funktionen.

11.2.1 Watchdog-Funktion

Die *Watchdog-Funktion* erfüllt den Zweck der Überwachung der verschiedenen Aufgaben, die auf den beiden CPU-Kernen ausgeführt werden. Blockiert eine Aufgabe die CPU über einen festgelegten Zeitwert hinaus, so wird der ESP32 durch die *Watchdog-Funktion* automatisch in den Deep Sleep-Modus versetzt. Auf diesem Weg wurde sichergestellt, dass Fehlfunktionen im Programmablauf, z.B. durch elektromagnetische Störungen oder Störungen des WLAN-Empfangs, nicht zu einem unkontrollierten Absturz und ggf. unnötigem Verbrauchen von Energie durch dauerhaften Betrieb führen. Der Schwellenwert für die Zeit, nach der der Watchdog aktiviert wird, kann im Quelltext frei eingestellt werden.

11.2.2 Reset-Funktion

Für den Fall, dass trotz der implementierten *Watchdog-Funktion* ein Absturz vorkommt, welcher nicht in einem Neustart oder dem Eintritt in den Deep-Sleep-Modus endet, wurde im Gehäuse der Stockwaage ein physikalischer *Reset-Taster* geplant. Dieser Taster ist mit dem Ground (GND) der Platine verbunden und zieht so den PIN mit der Bezeichnung *EN* auf den Zustand *low.* Sobald dies geschieht, wird die Spannungsversorgung des ESP32 unterbrochen und der Mikrocontroller somit neugestartet.

11.2.3 Deep-Sleep-Modus

Der ESP32 verfügt über verschiedene Energiemodi, welche es dem Anwender ermöglichen, den ESP32 bei Bedarf in einen Schlafmodus zu versetzen. Es gibt fünf verschiedene Betriebsmodi, die jeweils eine unterschiedliche Stromaufnahme besitzen: **[vgl. 27]**

- Normalbetrieb (ca. 60-200 mA)
- Modem-Sleep-Modus (ca. 3-20 mA)
- Light Sleep-Modus (ca. 0.8 mA)
- Deep Sleep-Modus (ca. 10 µA)
- Hibernation Modus (ca. 2,5 µA)

Abbildung 29 Veranschaulichung der Komponenten im Deep Sleep ***[vgl. 16]***

Die oben genannten Stromaufnahmewerte beziehen sich auf den Referenz-ESP32 vom Hersteller Espressif selbst. Die verschiedenen im Handel erhältlichen Entwicklerplatinen haben bauart bedingt zum Teil sehr abweichende Werte.

Je nach Schlafmodus werden verschiedene interne Funktionen abgeschaltet, um eine Senkung des Energieverbrauches zu erzielen. Für die Stockwaage wird der *Deep Sleep-Modus* verwendet. Der *Hibernation Modus* würde eine noch größere Energieersparnis als der *Deep Sleep-Modus* bieten, jedoch wird in diesem Modus auch der Real Time Clock (RTC)-Speicher des ESP32 abgeschaltet und die dort gespeicherten Inhalte gehen dementsprechend verloren.

Im für die Stockwaage verwendeten *Deep Sleep-Modus* werden die CPU, der RAM (bis auf den RTC-Bereich) und jegliche Peripherie, welche ihren Takt über den Advance Peripheral Bus (ABP) erhält, abgeschaltet. **[vgl. 17]**

11.2.4 Aufweck-Funktionen

Es bestehen mehrere Möglichkeiten, den ESP32 aus dem Deep Sleep aufzuwecken. Zum einen gibt es die Timer-Funktion, welche u.a. für den, die Stockwaage steuernden, ESP32 verwendet wird, um einen Mess- und Übertragungszyklus zu realisieren. Hierbei wird dem ESP32, vor dem Eintritt in den Deep Sleep, ein im Quelltext festgelegter Zeitwert übergeben. Wird dieser Zeitwert überschritten, erwacht der ESP32 aus dem Deep Sleep, um seine Programmroutine abzuarbeiten.

Aus Gründen, welche im Unterkapitel „14.1.3 GUI“ näher behandelt werden, war es außerdem nötig, eine weitere Art des Aufweckens zu programmieren. Neben der Möglichkeit, den ESP32 über eine Timer-Funktion aufzuwecken, besteht die Option, ihn über ein externes Signal, welches auf einen GPIO gesendet wird, aufzuwecken. Außerdem ist es möglich, über bestimmte GPIOs eine permanente

kapazitive Messung an einem zuvor definierten GPIO durchzuführen. Registriert der ESP32 während des Deep Sleep eine Veränderung des kapazitiven Wertes an diesem Pin, so wird der ESP32 aufgeweckt. Diese Option wurde für die Stockwaage als zweite Aufweck-Möglichkeit genutzt. Hierfür wurde eine leitende Verbindung vom Äußeren des Gehäuses, über die Platinen, bis zum GPIO mit der Nummer 27 geführt.

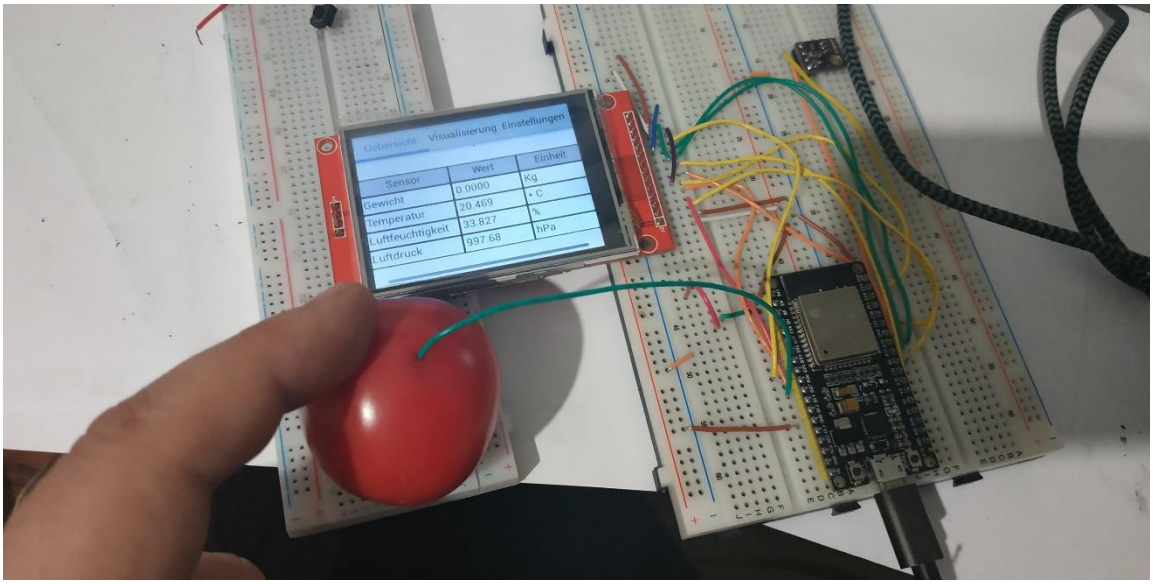


Abbildung 30 Test der kapazitiven Aufweckfunktion

11.2.5 RTC-Speicher

Der ESP32 besitzt die Möglichkeit, Daten und Variablen auch über den Deep Sleep hinaus zu speichern. Hierzu werden insgesamt 16 kB des SRAMs verwendet. 8 kB des Real Time Clock (RTC)-Speichers werden zum Speichern von Daten genutzt, die über den Deep Sleep hinaus erhalten bleiben sollen. Dieser Bereich nennt sich RTC-Fast-Speicher. Die anderen 8 kB des RTC-Speichers werden als RAM genutzt für Prozesse, die während des Deep Sleep stattfinden. Dieser Bereich wird RTC-Slow-Speicher genannt. **[vgl. 26]**

Da die in normalen Variablen gespeicherten Werte nach dem Deep Sleep gelöscht sind, ist es für die korrekte Funktion der Stockwaage nötig, mehrere Variablenwerte in sogenannten RTC-Variablen zu speichern. Die in den RTC-Variablen abgespeicherten Werte bleiben nach dem Deep Sleep erhalten **[vgl.101]** und werden genutzt, um während des Displaybetriebs die Ergebnisse der letzten Sensormessungen anzuzeigen.

```
RTC_DATA_ATTR int BootCounter = 0;   // Variable fuer einen Startzaehler im RTC Speicher
RTC_DATA_ATTR float gewicht;        // Variable fuer das Ergebnis der Gewichtsmessung im RTC Speicher
```

Abbildung 31 Beispiele für die Erzeugung von RTC-Variablen

Ein Nachteil der RTC-Variablen ist allerdings, dass diese nach einem Neustart des Mikrocontrollers z.B. durch Unterbrechen der Energieversorgung oder Betätigen der *Reset*-Taste, nicht erhalten bleiben. Da dieser Umstand zwar für den Displaybetrieb vernachlässigbar, aber für das Speichern des Tarierungswertes der Stockwaage problematisch ist, musste für diesen Wert eine andere Lösung gefunden werden.

11.2.6 Flash-Speicher

Da der ESP32 keinen EEPROM besitzt, aber einen Flash-Speicher, wurde eine Möglichkeit gesucht und gefunden, die über den Spannungsverlust des Mikrocontrollers hinaus zu speichernder Werter abzuspeichern. **[vgl. 28]**

Hierzu wurde durch Einbinden der Espressif eigenen Bibliothek *Preferences.h* eine Möglichkeit geschaffen, Daten im Flashspeicher des ESP32 permanent zu speichern. Die Daten bleiben dort erhalten, bis sie bewusst gelöscht oder überschrieben werden. Um Daten im Flash-Speicher des ESP32 zu speichern, muss ein Name vergeben und die zu speichernde Variable ausgewählt werden. Wurde der Wert der Variable übergeben, wird der Schreibvorgang beendet und die geschriebenen Daten bleiben selbst nach einer Unterbrechung der Stromversorgung erhalten.

```
void loop()
{
        Tare = (10);                                  // manuell gesetzter Tara-Wert
        preferences.begin("Tara", false);            // Oeffnen des Speicherorts fuer den Tara-Wertes
        preferences.putFloat("TaraGewicht", Tare);   // Uebergeben und Abspeichern des Tara-Wertes
        preferences.end();                          // Beenden des Abspeicherns


}
```

Abbildung 32 Speichern des Tara-Wertes im Flash-Speicher

11.2.7 Tarierfunktion

Die Tarierfunktion dient dazu, einen Tara Wert festzulegen, der immer mit dem aktuellen Messwert verrechnet wird. Die Funktion kann bei jedem Neustart oder

Erwachen des ESP ausgeführt werden, indem der Tara-Taster betätigt bleibt, bis der ESP wieder in den Deep Sleep wechselt. So kann die leere Beute tariert werden und die Waage gibt eine „0“ als aktuelles Gewicht aus.

Damit der Wert auch nach einem Neustart gespeichert bleibt, wird er im Flash-Speichers des ESP gespeichert.

11.2.8 Wiege-Funktion

Die Wiegefunktion wird nach dem Starten des ESP32 aufgerufen. In der Funktion wird der HX711 gestartet und nach dem Messvorgang wieder beendet, somit wird der Gesamtstrom während des Betriebes gesenkt. Die Funktion errechnet aus zehn Messwerten einen Durchschnittswert und subtrahiert den gespeicherten Tara-Wert. Dies ist notwendig, da sonst das sich aktuell auf der Waage befindende Gewicht ausgegeben wird und nicht das Gewicht ohne Beute.

Nach jeweils vier Messungen wird die fünfte Messung in die Datenbank hochgeladen. Die vorherigen vier Messungen dienen zur Kontrolle, ob ein Handeln erforderlich ist. Im Moment der Messung der zehn Werte ist es möglich, die Nachkommastellen, die angezeigt werden sollen, festzulegen. Da die Wägezelle eine Genauigkeit von ca. 21 g besitzt, wurde die Anzeige auf zwei Nachkommastellen festgelegt.

11.2.9 SMS-Funktion

Da die Möglichkeit besteht, dass der Bienenstock, z.B. durch einen Sturm oder Fremdeinwirkung von der Wiegekonstruktion gestoßen wird, wurde eine Funktion zum Versenden von Alarmierungs-SMS an den Imker implementiert.

Diese Funktion wird ausgeführt, falls der ESP32 durch Messung des Gewichts feststellt, dass einer von zwei Schwellenwerten unterschritten wird.

Wenn das tarierte Gewicht unter „0“ fällt, so wird die SMS-Funktion aktiviert, da, wenn das Gewicht auf „0“ fällt, von einem Windwurf oder Diebstahl auszugehen ist. Wenn das tarierte Gewicht abrupt um 2 kg abnimmt, so wird die SMS-Funktion aktiviert dieses Mal, weil ein Teil des Bienenvolkes abgeschwärmt ist, um sich einen neuen Stock zu suchen.

Abbildung 33 Vom ESP32 versendete Test-SMS

11.2.10 RTC-Uhrzeit

Bestandteil der Anfangsplanung war ein RTC-Modul, welches an den ESP32 angeschlossen werden sollte. Dieses RTC-Modul wird einmalig mit der aktuellen Zeit synchronisiert und zählt diese eigenständig weiter. Der ausfallsichere Betrieb des RTC-Moduls wird durch eine eigene interne Batterie sichergestellt. Das RTC-Modul sollte genutzt werden, um den an die Datenbank übertragenen Messwerten einen Zeitstempel hinzuzufügen. Dies sollte ein chronologisches Sortieren der Messwertübertragungen ermöglichen.

Schnell wurde aber klar, dass die aktuelle Uhrzeit inkl. Datum über eine WLAN-Verbindung zu einem Network-Time-Protocol (NTP)-Server bezogen werden kann. Darum wurde auf die Verwendung eines RTC-Moduls verzichtet. Dies hat den positiven Nutzen, dass durch den Verzicht auf ein solches Modul keine zusätzlichen GPIOs belegt werden und dass das Modul auch nicht weiter zum Energieverbrauch der Schaltung beiträgt.

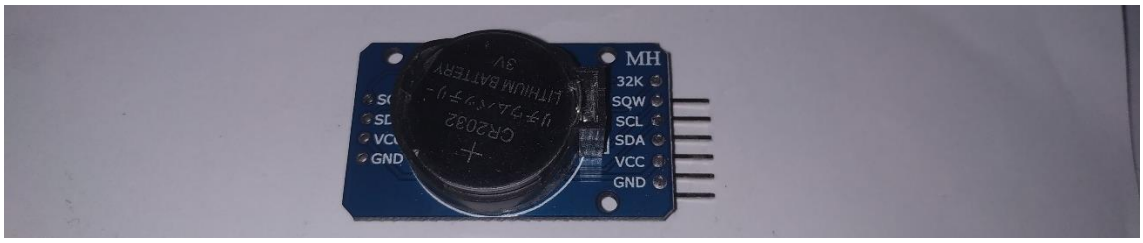


Abbildung 34 RTC-DS3231-Modul

11.2.11 NTP-Zeitfunktion

NTP ist der Name für ein global verwendetes Protokoll zur Zeitsynchronisierung. Ein Klient sendet hierbei eine Anfrage an einen NTP-Server und erhält als Antwort die aktuelle Uhrzeit inkl. Datumsangabe. **[vgl. 18]**

Diese Funktion wurde zuerst in den Quellcode der Stockwaage implementiert, jedoch nach Einrichtung und Inbetriebnahme der Datenbank wieder entfernt. Hintergrund dieser Entscheidung ist die Feststellung, dass ein separates Erfassen bzw. Übertragen der Uhrzeit an die Datenbank nicht nötig ist, da die Datenbank dem übertragenen Datensatz automatisch einen Zeitstempel zuordnen kann.

11.2.12 Graphical User Interface

Um der Imker AG eine Möglichkeit zu bieten, während der auf dem Schuldach stattfinden Arbeiten und Besuche am Bienenvolk aktuelle Informationen direkt an der Stockwaage abzulesen oder um die zum SMS-Versand hinterlegte Handynummer über eine Eingabe am Bildschirm zu ändern, wurde ein eigenes Graphical-User-Interface (GUI) programmiert.

Grundlage dieser GUI ist die äußerst umfangreiche LittlevGL Bibliothek bzw. die für den ESP32 und die TFT-eSPI-Bibliothek angepasste Version der LittlelvGL- Bibliothek. **[vgl. 25]** Mithilfe dieser Bibliothek war es möglich, eine von Grund auf selbsterstellte GUI für die Stockwaage zu programmieren.

Hauptbestandteile dieser GUI sind mehrere Objekte **[vgl. 110]**, welche mit der LittlevGL-Bibliothek erzeugt wurden.

Mithilfe des *tabview* Objektes werden drei verschiedene Seiten auf der GUI angezeigt. Durch Berühren der Reiter in der Kopfzeile oder durch Wischbewegungen lässt sich zwischen den Seiten navigieren.

Auf der ersten Seite wird eine Übersicht über die zuletzt erfassten Messdaten angezeigt. Hierzu wurde mittels des *table*-Objektes eine Tabelle gezeichnet, welche die zuvor in *strings* umgewandelten Messdaten inkl. Zuordnung zu einem Sensor und ihrer jeweiligen Einheit darstellt.

Abbildung 35 Übersichtsseite mit Messwerten

Die zweite Seite der GUI beinhaltet eine Messuhr mit einem Zeiger und einer Skala. Der Zeiger dieser Messuhr gibt das zuletzt gemessene Gewicht auf der Skala an. Der Maximalwert dieser Skala wurde so gewählt, dass er dem ungefähren maximalen Belastungsgewicht der Wägezelle (inkl. dem schon abgezogenen Tara-Wert) entspricht, um so vor einer Überlast zu schützen.

Abbildung 36 Visualisierung des Gewichtes inkl. Laufschrift

Auf der dritten Seite der GUI mit dem Titel „Einstellungen“ werden zwei *button*- Objekte angezeigt, die durch Berührung des Bildschirms betätigt werden können.

Bei Betätigung des „Deep Sleep“ -Buttons wird ein *event* ausgelöst, welches zuerst die angezogenen Relais durch Abschalten der jeweiligen GPIOs deaktiviert. Anschließend wird der ESP32 in den Deep Sleep-Modus versetzt.

Durch Betätigung des „Handy-Nr.“ Buttons wird es dem Benutzer ermöglicht, die zur SMS-Alarmierung hinterlegte Mobilfunknummer durch Eingeben am Bildschirm zu ändern. Die Eingabe der Handynummer geschieht über das Objekt *keyboard,* welches durch Betätigung des *buttons* aufgerufen wird.

Abbildung 37 Einstellungs-Seite der GUI

11.2.13 Anmeldung im WLAN

Bevor eine Übertragung der durch die Sensoren erfassten Messwerte möglich ist, muss eine Verbindung mit dem schuleigenen WLAN hergestellt werden. Dieser Weg der Datenübertragung wurde gewählt, um das Anfallen unnötiger Kosten durch einen GPRS-Datenbetrieb zu vermeiden. Da ein Anmelden des ESP32 über das Captive-Portal der TBS1 nicht gelang und ein Setzen der ESP32 MAC-Adresse auf die Whitelist zur Anmeldung im Netzwerk auch keinen Erfolg brachte, musste eine andere Lösung gefunden werden, um dem ESP32 eine Verbindung zum Übermitteln der Daten zu ermöglichen. Diese Lösung bestand darin, einen separaten Access-Point im Schulgebäude zu installieren und in die bestehende Netzwerktopologie einzubinden. Somit wäre es dem ESP32 möglich, sich mittels der SSID und dem Passwort in das neu erstellte WLAN einzuwählen. Da die Einbindung und die Beschaffung des neuen Access-Points schulinterne Vorgänge sind, welche zum Zeitpunkt der Erstellung dieser Dokumentation noch nicht abgeschlossen waren, konnte ein finaler Test zur Einwahl in das Schulnetzwerk noch nicht durchgeführt werden.

```
#include <WiFi.h>    //Einbinden der WiFi-Bibliothek

void setup() {


}

void loop()
{
WiFi.mode(WIFI_STA);                    //Aktivieren des WiFI-Modus
WiFi.begin("david", "12345678");       //Aufbau der WLAN-Verbindung inkl. SSID und Passwort
Serial.print("IP Addresse: ");        //Ausgabe der lokalen IP
Serial.println(WiFi.localIP());      //Ausgabe der lokalen IP
WiFi.disconnect(true);              // Trennen der WLAN-Verbindung
WiFi.mode(WIFI_OFF);               //Schaltet WLAN-Modul ab

}
```

Abbildung 38 Beispiel für die Anmeldung in einem bestehendem WLAN

11.2.14 Datenbankfunktion

Voraussetzungen für eine erfolgreiche Übertragung der Messwerte in eine Datenbank ist eine bestehende WLAN-Verbindung. Wurde diese aufgebaut, nutzt der ESP32 die der *HTTPClient.h*-Bibliothek entstammende HTTP-Funktion, um die zuvor in *strings* umgewandelten Messdaten zu einer URL zusammenzusetzen. Die Struktur dieser URL besteht aus der genauen Adresse des PHP-Skriptes, welche die in der URL als *strings* angehängten Messergebnisse entgegennimmt und dann in die Datenbank schreibt. Der Aufruf dieses PHP-Skriptes geschieht mittels der GET-Methode. Nachdem die Messwerte auf diese Weise vom ESP32 in die Datenbank übertragen wurden, wird die WLAN-Funktion wieder deaktiviert, um so den Energiebedarf des ESP32 im Deep Sleep-Modus zu senken.

11.3 Datenbank

Bei einer Datenbank handelt es sich um ein elektronisches Verwaltungssystem, in dem eine Vielzahl von Daten gesichert werden. Diese setzt sich aus zwei Komponenten, der Datenbank (DB) und dem Datenbankmanagement-system (DBMS), zusammen. Die DB besteht aus den gesicherten Daten, die nach bestimmten Regeln sortiert und strukturiert werden. Das DBMS ist ein Anwendungsprogramm und verwaltet die gesicherten Daten. Zu den Funktionalitäten des Anwendungsprogramms gehören das Suchen, Sortieren, Filtern und formatierte Ausgeben dieser Daten.

11.3.1 Grundlagen

Da sich die Stockwaage auf dem Flachdach des TBS1-Gedbäudes an der Moritz-Fiege-Straße in Bochum befindet, wurde eine Verbindung per Funk zum Schulnetzwerk benötigt.

Nachfolgend werden die Grundlagen zu den (frei verfügbaren) Programmen bzw. Betriebssystemen, welche für die Umsetzung der Projektziele im Bereich der Datenbank Erstellung, Verwaltung und Visualisierung genutzt wurden, näher erläutert.

11.3.1.1 OpenVPN

OpenVPN ist eine frei verfügbare Open-Source-Software, mit der sich verschlüsselte virtuelle private Netzwerke (VPNs) erstellen lassen, um ortsunabhängige einzelne Clients oder auch ganze Netzwerke sicher miteinander zu verbinden. Dadurch wird über ein unsicheres Netzwerk, wie z.B. das Internet, eine nicht lesbare Kommunikation von Dritten gewährleistet. Damit diese Sicherheit gegeben ist, muss die Software auf beiden Seiten vorhanden und dementsprechend konfiguriert sein.

11.3.1.2 SSH Client

Die frei verfügbare Open-Source-Software SSH (Secure Shell) ist ein Unix-basiertes Kommandozeileninterface und Protokoll, womit eine verschlüsselte Netzwerkverbindung hergestellt wird und der sichere Zugriff auf ein entferntes Gerät erfolgen kann. Die Einrichtung einer Komponente kann über SSH oder die serielle Konsole mittels Kommandozeileninterface ausgeführt werden, um Web- oder andere Server aus der Ferne zu warten.

11.3.1.3 Debian

Das frei verfügbare Betriebssystem, das den Linux-Betriebssystemkern (Kernel) nutzt und aus mehreren grundlegenden Programmen besteht, wird von einem Rechner zum Arbeiten benötigt, um mit dem Benutzer zu kommunizieren und Anweisungen von ihm zu erhalten. Die Aufgabe von Debian ist, all dies zu managen und sorgfältig anzupassen, um eine gute Zusammenarbeit zu gewährleisten.

11.3.1.4 Nginx

Die frei verfügbare Software fungiert als Open-Source-Webserver, Reverse-Proxy („umgekehrter Vermittler“), HTTP-Cache (Puffer-Speicher) und Load-Balancer (Lastverteilung). Diese kann eine Vielzahl an Clients gleichzeitig bedienen und mehrere Anfragen, wie z.B. zur Nachrichten- und Datenübertragung, bearbeiten.

11.3.1.5 MariaDB

Diese Software ist aus einem Fork (Abspaltung) von MySQL entstanden und ist beinahe vollständig kompatibel zu MySQL. Es fungiert als freies, relationales Open-Source-Datenbankmanagementsystem und enthält genau die gleichen

Basisfunktionen wie die Datenbank, z.B. das Erstellen, Lesen, Ändern und Löschen von Daten.

11.3.1.6 phpMyAdmin

PhpMyAdmin ist ein, in PHP geschriebenes, Open-Source Softwarewerkzeug und ermöglicht eine ortsunabhängige Erstellung und Verwaltung von MySQL-Datenbanken oder MariaDB-Datenbankservern über das Internet. Anhand der Kompatibilität zu MySQL können die Basisfunktionen, wie z.B. eine Datenbank/Tabelle anlegen und löschen, Spalten hinzufügen, Datensätze auflisten oder Benutzer verwalten, sowohl manuell als auch textbasiert über die Programmiersprache SQL erfolgen.

11.3.2 MySQL

MySQL ist eine Open-Source-Software, wobei SQL für Structured Query Language (strukturierte Abfragesprache) steht und zur Verwaltung von relationalen Datenbanken dient. Diese wird auch oftmals als Datenbankmanagementsystem oder Datenbanksystem bezeichnet und bildet die Grundlage für viele dynamische Webseiten. Die Funktionsweise einer MySQL-Datenbank wird auch als Client-Server-Prinzip bezeichnet, bei dem die Datenbank als Server fungiert und über die Datenbanksprache SQL angesteuert werden kann. Sobald der MySQL-Server einen Befehl von einem Client erhält, übersetzt ihn die Datenbank, führt ihn aus und sendet das Ergebnis zurück an den Client.

11.3.3 Einrichtung der Datenbank

Für die Projektgruppe existierte kein Benutzer mit der Berechtigung, um auf dem Datenbank-/Webserver zu agieren. Deshalb wurde zuerst von Herrn Schreiber, dem IT-Koordinator der TBS1, eine Debian10-basierende MariaDB/Nginix-virtuelle Maschine erstellt und die Projektgruppe als User eingetragen. Damit konnten über den Debian-User-Account Daten in das Wurzelverzeichnis des Webservers kopiert werden. Zusätzlich wurde ein Datenbankuser angelegt, der berechtigt ist, in der dann bestehenden leeren Datenbank (bs_waage) Tabellen anzulegen und deren Inhalte zu pflegen.

11.3.4 Erstellung der Datenbanktabelle

Über phpMyAdmin wurde die Datenbanktabelle bzw. das relationale Datenmodell *sensordaten* manuell erstellt und mit einer bestimmten Struktur aufgebaut, sodass die erste Normalform gegeben ist. Das heißt, die ermittelten Sensordaten werden an die Datenbank übermittelt und in die Spalten *gewicht*, *temperatur*, *luftfeuchtigkeit*, *luftdruck* und *gehäusetemperatur* gesichert. Alle Spalten in dem Datensatz erhalten nur einen einzigen Wert, zwischen denen wiederum eine Beziehung besteht. Sie besitzen alle den gleichen *zeitstempel*, der anhand des

Datentyps „TIMESTAMP“ von der Datenbank erstellt wird. Der Primärschlüssel *sensordaten_id* dient zur Nummerierung der Datensätze und ist mit dem Attribut *UNSIGNED* sowie dem Extra *A_I (auto increment)* konfiguriert. Diese ermöglichen das automatische Hochzählen, das Sichern von ausschließlich positiven Werten und Null sowie die eindeutige Identifizierung. Dadurch wird die Abfrage der Datenbank erleichtert und die Verschwendung von Speicherplatz vermieden. Durch den Server-Zeichensatz *utf8mb4_unicode_ci* werden eine problemlose Datenübertragung und genaue Darstellung der Daten gewährleistet. Dieser ist die am weitesten verbreitete Zeichencodierung im Internet, enthält eine Vielzahl von Schriftzeichen/Symbolen und beachtet nicht die Groß- und Kleinschreibung.

Server: localhost » Datenbank: bs_waage » Tabelle: sensordaten

Anzeigen | Struktur | SQL | Suche | Einfügen | Exportieren | Importieren | Rechte | Operationen

Tabellenstruktur | Beziehungsansicht

#	Name	Typ	Kollation	Attribute	Null	Standard	Kommentare	Extra	Aktion
1	sensordaten_id	int(10)		UNSIGNED	Nein	*kein(e)*		AUTO_INCREMENT	Bearbeiten Löschen
2	zeitstempel	timestamp			Ja	current_timestamp()			Bearbeiten Löschen
3	gewicht	float			Ja	*NULL*			Bearbeiten Löschen
4	temperatur	float			Ja	*NULL*			Bearbeiten Löschen
5	luftfeuchtigkeit	float			Ja	*NULL*			Bearbeiten Löschen
6	luftdruck	float			Ja	*NULL*			Bearbeiten Löschen
7	gehäusetemperatur	float			Ja	*NULL*			Bearbeiten Löschen

Abbildung 39 Erstellung der Datenbanktabelle

11.4 PHP

PHP ist eine weit verbreitete Open Source-Skriptsprache, und steht für *PHP: Hypertext Preprocessor*. Mit PHP, dessen Syntax den Programmiersprachen C und Pearl ähnelt, kann beispielsweise eine Verbindung zu der Datenbank erstellt werden und die Datenübertragung erfolgen. Sie wird größtenteils für die Webprogrammierung bzw. Erstellung dynamischer Webseiten oder Webanwendungen verwendet und kann in *HTML* implementiert werden.

Damit die Datenübertragung erfolgen kann, musste zuerst die Verbindung vom ESP32 zu der Datenbank hergestellt werden. Dafür wurden die PHP-Skripte *config.php* und *writemysql.php* erstellt, die zum Verbindungsaufbau sowie übertagen der Sensordaten dienen und bestimmte Abschnitte davon, als Grundgerüst für alle weiteren Skripte verwendet wurden.

In der *config.php*-Datei wurden lediglich die Zugangsdaten zur Datenbank in Variablen gesichert, wodurch in Zukunft eine einfache Änderung der Zugangsdaten erfolgen kann, indem diese nur einmal in der *config.php*-Datei geändert werden. Das PHP-Skript wird mit dem Tag *<?php* geöffnet, die Zugangsdaten in Variablen gesichert und mit *?>* geschlossen.

```php
<?php

    // Variablendeklaration der Anmeldedaten zur Datenbank

    $servername = "Servername";
    $dbname = "Datenbankname";
    $username = "Benutzer";
    $passwort = "Passwort";

?>
```

Abbildung 40 Zugangsdaten in Variablen sichern

In der *writemysql.php*-Datei wird zu Beginn mit *include 'config.php';* die *config.php*-Datei implementiert, damit die in den Variablen gesicherten Zugangsdaten zur Datenbank vorhanden sind und der Verbindungsaufbau erfolgen kann. Nachdem der ESP32 die ermittelten Sensordaten in den Variablen *temp*, *luftd*, *luftf*, *gewi* und *gtemp* abgesichert und über die URL übertragen hat, überprüft das PHP-Skript zuerst, ob sie existieren, um sie danach weiterzuleiten. Dies erfolgt über die *isset- und GET-Funktion*. Das heißt, wenn die Variable *temp* existiert wird ihr Inhalt in die neue Variable *temp* überschrieben. Ist dies nicht der Fall wird ein *FALSE* ausgegeben.

```php
<?php

    // Einbinden der config.php-Datei.
    include 'config.php';

    // Es wird überprüft, ob die angegebenen Variablen existieren.
    // Wenn das zutrifft werden diese in die neuen Variablen überschrieben.

    $temp = isset($_GET['temp']) ? $_GET['temp'] : FALSE;
    $luftd = isset($_GET['luftd']) ? $_GET['luftd'] : FALSE;
    $luftf = isset($_GET['luftf']) ? $_GET['luftf'] : FALSE;
    $gewi = isset($_GET['gewi']) ? $_GET['gewi'] : FALSE;
    $gtemp = isset($_GET['gtemp']) ? $_GET['gtemp'] : FALSE;
```

Abbildung 41 Implementierung Zugangsdaten/Überprüfung der Variablen

Über die try-Anweisung wird versucht mittels der Klasse PDO (PHP Data Objects) eine Verbindung zu der Datenbank herzustellen. Diese ist Objektorientiert aufgebaut, fungiert als Datenbankschnittstelle und kann auf unterschiedliche SQL-basierte Datenbanken zugreifen.

Nachdem die Verbindung anhand PDO und der in den Variablen gesicherten Zugangsdaten aufgebaut ist, wird ein Prepared Statement erstellt. Dadurch muss die Abfrage der Datenbank mittels der *SQL INSERT INTO-Funktion*, nur einmal vorbereitet werden und ermöglicht eine ressourcenschonende sowie kürzere Bearbeitungszeit. Letztendlich wird Anhand *execute* das *Prepared Statement* ausgeführt und die Variablen bzw. Sensordaten an die Datenbank übermittelt.

```
try

    {
    // Verbindung zur Datenbank wird aufgebaut

    $pdo = new PDO("mysql:host=$servername;dbname=$dbname", $username, $passwort);

    // Es wird eine Anfrage an die Datenbank vorbereitet, um die Variablen an die Datenbank zu übermitteln.

    $statement = $pdo->prepare("INSERT INTO sensordaten (temperatur, luftdruck, luftfeuchtigkeit, gewicht, gehäusetemperatur)
    VALUES (?, ?, ?, ?, ?)");

    // Die Anfrage an die Datenbank wird ausgeführt und die Daten werden übermittelt.

    $statement->execute(array($temp, $luftd, $luftf, $gewi, $gtemp));
    }
```

Abbildung 42 try-Anweisung (Verbindungsaufbau/Datenübertragung)

Sobald die Datenübertragung nicht durchgeführt wird und ein fehlerhaftes SQL-Statement besteht, gibt die catch-Anweisung mittels *PDOException* eine Fehlermeldung aus.

```
// Begin der catch Anweisung "Überprüfung"

catch (PDOException $e)

    {
    // Gibt diese Fehlermeldung aus, wenn bei dem Verbindungaufbau ein
    // Fehler aufgetretten ist oder die Daten nicht übermittelt wurden.
    echo 'Vorgang Fehlgeschlagen !!!: '.$e->getMessage();
    }

?>
```

Abbildung 43 catch-Anweisung (Fehlermeldung bei Verbindungsproblemen)

11.5 HTML

HTML (Hypertext Markup Language) ist eine textbasierte Auszeichnungssprache und die Grundlage des World Wide Web. Es dient als elektronisches Dokument zum Erstellen bzw. Schreiben einer Webseite und findet oftmals Verwendung in PHP-Skripten. Das heißt, über das Hinzufügen von textbasierten Kommandos kann die Gliederung/Formatierung von Texten und anderen Daten auf einer Webseite erfolgen.

Nachdem der Verbindungsaufbau zu der Datenbank sowie die Datenübertragung über das *writemysql.php* PHP-Skript erfolgen konnte, mussten die Sensordaten kontinuierlich aus der Datenbank an eine Webseite übermittelt werden, um sie dort tabellarisch und grafisch darzustellen. Dafür mussten weitere Dateien erstellt werden, die nicht nur zu Umsetzung der Ziele dienen, sondern auch für eine angemessene Übersicht und sinngemäße Struktur sorgen.

Folgende Dateien wurden erstellt:

- index.php
- readmysql.php
- graf_darst_tage.php
- graf_darst_wochen.php
- graf_darst_monate.php

Diese bestehen aus HTML, einem CSS (Cascading Style Sheets) und den darin implementierten PHP-Codes. Das Stylesheet ist eine Formatvorlage, die am häufigsten mit HTML kombiniert wird und zur Gestaltung einer Webseite dient. Es wird als Grundgerüst für die neu erstellten Dateien verwendet, um ein einheitliches Design zu erhalten. Für die darin implementierten PHP-Codes wurden bestimmte Abschnitte aus dem *writemysql.php* PHP-Skript als Grundgerüst benötigt, damit der Verbindungsaufbau zu der Datenbank und das Ausführen des *Prepared Statments* bzw. Abfragen der Datenbank von jeder dieser Webseiten erfolgen kann. Dementsprechend wurden sie erweitert, um die Sensordaten tabellarisch und grafisch darzustellen.

11.5.1 Startseite

Damit die Startseite erstellt und eine angemessene Übersicht des Dokuments gewährleistet wird, musste in der *index.php*-Datei eine sinngemäße Struktur angelegt werden.

Das Tag *<html></html>*, worin sich auch der gesamte Inhalt der Startseite befindet, dient lediglich zum Öffnen und Schließen des Dokuments. Die nachfolgenden Elemente *<head></head>* und *<body></body>* werden für die Struktur und Darstellung der Webseite genutzt.

Das *head-Element* bezeichnet den Kopfteil und enthält die Zeichenkodierung, den *Hyperlink* zum verwendeten Stylesheet und den Titel der Webseite. Diese Inhalte sind relevant und werden benötigt, um eine angemessene Darstellung der Daten auf der Webseite zu gewährleisten.

```
<!-- Beginn des html-Bereichs. -->
<html>
<!-- Beginn des head-Bereichs. -->
<head>
 <!-- Angaben zu den Erforderlichen Inhalten, die zur Darstellung der Webseite dienen. -->
 <!-- Verwendung der Zeichencodierung, die zur Darstellung der Daten dient. -->
 <meta charset="utf-8">
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <!-- Verwenden des Stylesheets "Bootstrap CSS" für das Webdesign. -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0
    <!-- Titel der HTML-Seite. -->
    <title>Bienenstockwaage</title>
 <!-- Ende des head-Bereichs. -->
 </head>
```

Abbildung 44 Erstellen der Darstellungsform

Das *body-Element* bezeichnet den Körper und beinhaltet das Navigationsmenü sowie Text und Bilder, die zur Gestaltung und für ein ansprechendes Design der Webseite dienen. Dafür wurden zwei *Container-Elemente* erstellt, die zur Gliederung und Strukturierung der Webseite genutzt werden. Der erste Container beinhaltet weitere *div-Elemente*, die zur Darstellung des Navigationsmenüs dienen und einen HTML-Code enthalten, der vom Anbieter des Stylesheets zur Verfügung gestellt wird. Über die *div-Elemente* werden die Menüpunkte in Reihe aufgeführt und das Navigationsmenü, je nach Ansichtsgröße automatisch angepasst. Durch das *li-Element* wird jeder Menüpunkte als Listeneintrag und aktiver Link definiert. Das heißt, sobald der Menüpunkt *Home* ausgeführt wird, ruft das Dokument die *index.php*-Datei auf. Dementsprechend wurden die weiteren Menüpunkt nach diesem Schema aufgebaut.

```
<!-- Beginn des body-Bereichs. -->
<body>
  <!-- Erstellen des Containers, um einen gemeinsamen Bereich zu definieren (Navigationsmenü). -->
  <div class="container">
    <!-- korrekte Ausrichtung und Abstände der Zeilen. -->
    <div class="row">
      <!-- Rastersystem zur Darstellung des Navigatonsmenüs, das Größentechnisch automatisch angepasst wird.  -->
      <div class="col-md-auto">
        <!-- Beginn der vertikalen Ansicht (Navigationsleiste). -->
        <ul class="nav flex-column">
          <!-- Beginn der Listenitems. -->
          <li class="nav-item">
            <!-- Erstellen des Hyperlinks "Home", der bei seiner Ausführung die Datei "index.php" abruft.  -->
            <a class="nav-link active" href="index.php"><b>Home</b></a>
          <!-- Ende der Listenitems. -->
          </li>
          <!-- Beginn der Listenitems. -->
          <li class="nav-item">
            <!-- Erstellen des Hyperlinks "Sensordaten", der bei seiner Ausführung die Datei "readmysql.php" abruft.  -->
            <a class="nav-link" href="readmysql.php"><b>Sensordaten</b></a>
          <!-- Ende der Listenitems. -->
          </li>
          <!-- Ende der vertikalen Ansicht (Navigationsleiste). -->
          </ul>
```

Abbildung 45 Erstellen des Navigationsmenüs

Der Zweite Container beinhaltet Text- und Bilddateien und dient zur Gestaltung des Hauptteils der Webseite. Die Textstrukturierung erfolgt über das *h1*- und *p-Element*, wobei *h1* für die Schriftgröße und *p* für die Zentrierung sowie Absatzgliederung verwendet wird. Danach werden über das *tbody-Element*, welches zur Erstellung eines Tabellenkörper dient, in Verbindung mit dem *img-Element* Bilder eingefügt sowie deren Größe bestimmt. Nach diesem Schema wurden weitere Text- und Bilddateien eingebunden, wodurch sich ein strukturierter Aufbau des Hauptteils ergibt.

```
<!-- Erstellen eines weiteren Containers. -->
<div class="col">
 <!-- Der Text wird in der Größe <h1> (größte Überschrfitengröße) ausgegeben. -->
 <h1>Herzlich Willkommen auf der Übersichtsseite der Projektgruppe Stockwaage !</h1>
  <!-- Dient zur Absatzgliederung und gibt den ausgegeben Text linksbündig sowie mit einem definierten Seitenabstand aus. -->
  <p align="left" style="margin-bottom: 25px">
   <!-- Einen Absatz einfügen. -->
   </br>
    <!-- Beginn des tbody-Bereichs. -->
    <tbody>
      <!-- Einbinden einer Bilddatei in einer bestimmten größe und einem Alternativtext. -->
      <img src="logoTBS1.png" alt="Logo der TBS1" width="700" heigt="150">
      <!-- Einbinden einer Bilddatei in einer bestimmten größe und einem Alternativtext. -->
      <img src="logo_projektgruppe.png" alt="Logo der Projektgruppe" width="500" heigt="370">
      <!-- Einen Absatz einfügen. -->
      </br>
      <!-- Der Text wird in der Größe <h6> (kleinste Überschrfitengröße) ausgegeben. -->
      <h6>
       Im schuleigenen Kiosk kann der Honig für 3,50 € in 250g-Pfandgläsern erworben werden.
       In den 3,50 € sind schon 25ct Pfand enthalten. Der Honig hat einen sehr eigenen Geschmack nach Linde und Kastanie.
      </h6>
      <!-- Ende der Absatzgliederung. -->
      </p>
      <!-- Einbinden einer Bilddatei in einer bestimmten größe und einem Alternativtext. -->
      <img src="honig.jpg" alt="Honig" width="500" heigt="370">
    <!-- Ende des tbody-Bereichs. -->
    </tbody>
   <!-- Ende der Absatzgliederung. -->
   </p>
```

Abbildung 46 Einbinden von Text- und Bilddateien

Letztendlich mussten die verwendeten Elemente und Tags noch geschlossen werden, damit das Dokument im Webbrowser erkannt und aufgerufen wird.

```
      <!-- Schließen des Containers (Hauptteil bestehend aus Text- und Bilddateien). -->
      </div>
    <!-- Ende des Containers. -->
    </div>
  <!-- Ende des Containers. -->
  </div>
<!-- Ende des body-Bereichs. -->
</body>
<!-- Ende des HTML Bereichs. -->
</html>
```

Abbildung 47 Schließen der Elemente und Tags

Die Startseite sieht folgendermaßen aus:

Abbildung 48 Ansicht der Startseite (Webseite)

11.5.2 Tabellarische Darstellung

Für die tabellarische Darstellung der Sensordaten wurde die *config.php*-Datei in die *readmysql.php*-Datei implementiert und Abschnitte aus der *index.php*-Datei eingebunden. Dazu gehören, der gesamte Inhalt des *html-Tags*, *head-Elements* und das im *Container-Element* erstellte Navigationsmenü aus dem *body-Element*. Danach wurde erneut ein zweiter *Container* erstellt, der über den PHP-Code die Datenbank nach dem Datum und der Uhrzeit abfragt, um diese in dem Hauptteil der Webseite auszugeben. Das Einfügen der darauffolgenden Textdatei sowie das Bestimmen der Schriftgröße, als auch die Zentrierung erfolgt über das *h1*- und *p-Element*.

```
<?php
// Ausgabe von Datum und Uhrzeit über die Datenbank
// Variablendeklaration
$datum = date("d.m.Y");
$uhrzeit = date("H:i");

// Ausgabe von Datum und Uhrzeit.
echo $datum," - ",$uhrzeit," Uhr";
?>
    <!-- Der Text wird in der Größe <h1> (größte Überschrfitengröße) ausgegeben. -->
    <h1>Sensordaten der Bienenstockwaage</h1>
    <!-- Dient zur Absatzgliederun und gibt den ausgegeben Text linksbündig sowie mit einem definierten Seitenabstand aus. -->
    <p align="left" style="margin-bottom: 25px">
```

Abbildung 49 Ausgabe von Datum/Uhrzeit und einer Textdatei

Danach musste über einen zweiten PHP-Code die Verbindung zu der Datenbank erstellt werden. Sobald diese hergestellt wurde, kann eine Abfrage zur Datenübertragung vorbereitet werden. Über die *SQL SELECT und COUNT Funktionen* wird die Anzahl der vorhandenen Datensätze ermittelt, durch die *fetch-Funktion* aufgefangen, als Reihe definiert und in die Variable *$row* geschrieben. Anschließend wird die Anzahl der vorhandenen Daten in einer Textdatei auf der Webseite ausgegeben. Dafür mussten die *p- und b-Elemente* eingebunden werden, damit eine angemessene Darstellung erfolgen kann. Die Zweite Abfrage erfolgt über die *SQL SELECT Funktion*, die alle Daten nach dem *zeitstempel* absteigend sortiert und in die Variable *$sql* sichert.

```php
<?php
// Verbindung zur Datenbank wird aufgebaut.
$pdo = new PDO("mysql:host=$servername;dbname=$dbname", $username, $passwort);

// Es wird eine Abfrage der Datenbank vorbereitet.
$statement = $pdo->prepare("SELECT COUNT(*) AS anzahl FROM sensordaten");

// Die Abfrage wird ausgeführt und die Anzahl der vorhandenen Daten ermittelt.
$statement->execute();

// Die Daten werden als Reihe definiert.
$row = $statement->fetch();

// Textausgabe über die Anzahl der vorhandenen Daten.
echo '<p class="h3" style="margin-bottom: 30px">Es wurden  <b>'.$row['anzahl'].'</b> Einträge gefunden.</p>';

// Absteigend nach dem Zeitstempel werden alle Daten übertragen.
$sql = "SELECT * FROM sensordaten ORDER BY zeitstempel DESC";
?>
```

Abbildung 50 Datenbankverbindung und -abfrage

Durch das vorhandene Stylesheet konnte die Tabelle erstellt und auf die Sensordaten bzw. Datenbank angepasst werden. Über das *table-Element* wurde die Tabelle erstellt und anhand des *thead-Elements* der Tabellenkopf festgelegt. Das darauffolgende *tr-Element* definiert den Tabellenkopf als Reihe und erhält durch das *th-Element* die Überschrift *zeitstempel*. Nach diesem Schema wurden die weiteren Zellen erstellt und mit Überschriften versehen.

```html
<!-- Erstellen einer Tabelle. -->
<table class="table">
  <!-- Beginn des Tabellenkopfes. -->
  <thead>
    <!-- Definieren einer Reihe. -->
    <tr>
      <!-- Beginn der vertikalen Überschriftenzelle Zeitstempel. -->
      <th scope="col" style="vertical-align:middle">Zeitstempel</th>
      <!-- Ende der vertikalen Überschriftenzelle Zeitstempel. -->

      <!-- Erstellen der vertikalen Überschriftenzelle Gewicht, einem Absatz und dem Text (kg). -->
      <th scope="col" style="vertical-align:middle">Gewicht <br>(kg)</br></th>
      <!-- Ende der vertikalen Überschriftenzelle Gewicht. -->
```

Abbildung 51 Erstellen einer Tabelle

Bei jeder Abfrage der Datenbank werden die neu erfassten Daten durch die *foreach-Schleife* als definierte Reihe ausgegeben. Danach wird über das *tbody-Element* ein Tabellenkörper erstellt und mit dem *tr-Element* eine Reihe definiert. Das *th-Element* erstellt in Verbindung mit dem PHP-Code die Überschriftenzelle *zeitstempel,* die zum Definieren eines Datensatzes in der tabellarischen Darstellung dient. Dieser Vorgang wird nur für eine Überschriftenzelle durchgeführt, wobei die restlichen als Tabellenzeile definiert werden.

```
            <!-- Ende der Reihe. -->
            </tr>
        <!-- Ende des Tabellenkopfes. -->
        </thead>
              <!-- Beginn der foreach-Schleife. -->
              <?php foreach ($pdo->query($sql) as $row) : ?>
        <!-- Beginn des tbody-Bereichs. -->
        <tbody>
          <!-- Definieren einer Reihe. -->
          <tr>
            <!-- Überschriftenzelle als Reihe definieren. -->
            <th scope="row">
              <!-- Den Wert zeitstempel ausgeben. -->
              <?php echo $row['zeitstempel'] ?>
            <!-- Ende der Überschriftenzelle als Reihe. -->
            </th>
            <!-- Einleiten einer neuen Tabellenzeile. -->
            <td>
              <!-- Den Wert gewicht ausgeben. -->
              <?php echo $row['gewicht'] ?>
            <!-- Ende der Tabellenzeile. -->
            </td>
```

Abbildung 52 Datenübertragung in die Tabelle

Sowohl die *Elemente*, sowie die *foreach-Schleife*, als auch die *Tags* mussten zum Schluss wieder geschlossen werden.

```
            <!-- Ende der Reihe. -->
            </tr>
        <!-- Ende des tbody-Bereichs. -->
        </tbody>
<?php
// Ende der foreach-Schleife.
endforeach;
?>
        <!-- Ende der Tabelle. -->
        </table>
      <!-- Ende der Absatzgliederung. -->
      </p>
    <!-- Ende des Containers. -->
    </div>
  <!-- Ende des Containers. -->
  </div>
 <!-- Ende des Containers. -->
 </div>
<!-- Ende des body-Bereichs. -->
</body>
<!-- Ende des HTML Bereichs. -->
</html>
```

Abbildung 53 Schließen der foreach-Schleife, Elemente und Tags

Die tabellarische Darstellung der Sensordaten sieht folgendermaßen aus:

Zeitstempel	Gewicht (kg)	Temperatur (° C)	Luftfeuchtigkeit (%)	Luftdruck (hPa)	Gehäusetemperatur (° C)
2019-12-30 14:00:00	46.78	5.7	49	993	6.4
2019-12-30 14:00:00	46.78	5.7	49	993	6.4
2019-12-26 14:00:00	49.32	6	52	868	7.2
2019-12-26 14:00:00	49.32	6	52	868	7.2
2019-12-17 16:00:00	55.42	8.3	52	993	9
2019-12-17 16:00:00	55.42	8.3	52	993	9
2019-12-12 17:00:00	57.31	8.5	52	868	9.3
2019-12-12 17:00:00	57.31	8.5	52	868	9.3
2019-12-02 15:00:00	59.37	6	56	986	8
2019-12-02 15:00:00	59.37	6	56	986	8

Abbildung 54 Ansicht der tabellarischen Darstellung (Webseite)

11.5.3 Grafische Darstellungen

Dafür wurden die Dateien *graf_darst_tage.php*, *graf_darst_wochen.php* und *graf_darst_monate.php* benötigt, um das Gewicht über drei Liniendiagramme in der Tages-, Wochen- und Monatsansicht darzustellen.

Für die Erstellung wurde ebenfalls die *config.php-Datei* implementiert und Abschnitte aus der *index.php-Datei* eingebunden. Dazu gehören ebenfalls wie in der *readmysql.php*-Datei, der gesamte Inhalt des *html-Tags*, *head-Elements* und das im *Container-Element* erstellte Navigationsmenü aus dem *body-Element*. Es wurde erneut ein zweiter *Container* erstellt, der über den PHP-Code zuerst ein *Array* erstellt. Dort können die Sensordaten abgelegt und von dem Liniendiagramm jederzeit aufgerufen werden. Danach wird erneut die Verbindung zu der Datenbank erstellt und eine Abfrage vorbereitet. Sobald die Verbindung hergestellt wurde, wird über die *SQL SELECT und COUNT Funktionen* die Anzahl der Datensätze ermittelt und diese als Reihe definiert. Die Zweite Abfrage erfolgt über die *SQL SELECT und AVG Funktionen*, die den Durchschnittswert des Gewichts pro Tag berechnen und durch *DESC LIMIT 7* sowie *ASC* höchsten nur 7 Werte in aufsteigender Sortierung ausgeben.

```php
<?php
// Erstellen eines Arrays.
$dataPoints = array();

// Verbindung zur Datenbank wird aufgebaut.
$pdo = new PDO("mysql:host=$servername;dbname=$dbname", $username, $passwort);

// Es wird eine Abfrage der Datenbank vorbereitet.
$statement = $pdo->prepare("SELECT COUNT(*) AS anzahl FROM sensordaten");

// Die Abfrage der Datenbank wird ausgeführt.
$statement->execute();

// Die Daten werden als Reihe definiert.
$row = $statement->fetch();

// Abfrage der Datenbank zur Ermittlung des Durchschnittswertes (pro Tag)
$sql = "SELECT * FROM (SELECT AVG(gewicht) AS dgewicht, DATE(zeitstempel) as Tag FROM sensordaten
        GROUP BY DATE(zeitstempel) DESC LIMIT 7) AS summary GROUP BY Tag ASC";
?>
```

Abbildung 55 Erstellen eines Arrays, einer Datenbankverbindung und -abfrage

Über die *foreach-Schleife* wird die Datenbank abgefragt und der Durchschnittswert des Gewichts pro Tag als definierte Reihe ausgegeben. Danach wird der im Array gesicherten Durchschnittswert sowie das Datum der x- und y-Achse zugewiesen. Anschließend wird die foreach-Schleife wieder beendet.

```php
<?php
// Beginn der foreach-Schleife.
foreach ($pdo->query($sql) as $row) :
?>

<?php
      // Abrufen der im Array gesicherten Daten.
      // Derfinieren der x- und y-Achse mit den Werten dgewicht und Tag
      $dataPoints[] = array("y" => $row['dgewicht'], "label" => $row['Tag']);
?>

     <!-- Ende der foreach-Schleife. -->
     <?php endforeach;?>
     <!-- Ende der Absatzgliederung. -->
     </p>
     <!-- Einfügen von Absätzen. -->
     </br>
     </br>
     </br>
```

Abbildung 56 Speichern der Durchschnittwerte in einem Array

Das *script-Element* dient zum Definieren des Script-Bereichs und wurde vom Anbieter des JavaScripts zur Verfügung gestellt. Durch die *window.onload = function* wird die grafische Darstellung nach dem Laden der HTML-Codes mit CSS-, Bild- und Text-Dateien aufgerufen, da der Ladevorgang etwas länger dauert. Über *title*, *axisY* und *data* kann den Script-Bereich bzw. das Liniendiagramm eine Überschrift, Beschriftung der y-Achse und der Linie implementiert werden.

```
<!-- Beginn des script-Bereichs. -->
<script>
window.onload = function () {

<!-- Überschrift des Liniengraphen. -->
var chart = new CanvasJS.Chart("chartContainer", {
  title: {
    text: "Tag"
  },
  <!-- Beschriftung der y-Achse. -->
  axisY: {
    title: "Gewicht (kg)"
  },
  <!-- Die Linie des Graphen mit den Daten aus dem Array erstellen. -->
  data: [{
    type: "line",
    dataPoints: <?php echo json_encode($dataPoints, JSON_NUMERIC_CHECK); ?>
  }]
  });
  chart.render();

  }
<!-- Ende des script-Bereichs. -->
</script>
```

Abbildung 57 Beschriftung des Liniendiagramms (Script-Bereich)

Danach wird ein zweiter Container in einer bestimmten Größe erstellt und ein zweiter Script-Bereich definiert. Anhand des *src-Attribut* wird das zur Verfügung gestellte *JavaScript*, über den *Hyperlink* aufgerufen und die Durchschnittswerte grafisch dargestellt. Zum Schluss werden erneut die *Elemente* und *Tags* geschlossen. Nach diesem Schema wurden ebenfalls die Wochen- und Monatsansicht erstellt, nur das die SQL-Abfrage umprogrammiert wurde, um den Wochen- und Monatsdurschnitt des Gewichts zu ermitteln.

```
<!-- Erstellen eines weiteren Containers mit bestimmter größe. -->
<div id="chartContainer" style="height: 370px; width: 100%;"></div>
<!-- Beginn des script-Bereichs (Javascript in HTML einbinden). -->
<script src="https://canvasjs.com/assets/script/canvasjs.min.js"></script>
        <!-- Ende der Absatzgliederung. -->
        </p>
      <!-- Ende des Containers. -->
      </div>
    <!-- Ende des Containers. -->
    </div>
  <!-- Ende des Containers. -->
  </div>
<!-- Ende des tbody-Bereichs. -->
</tbody>
<!-- Ende des html-Bereichs. -->
</html>
```

Abbildung 58 Aufrufen des JavaScripts (Hyperlink)

Die grafische Darstellung der Sensordaten sieht folgendermaßen aus:

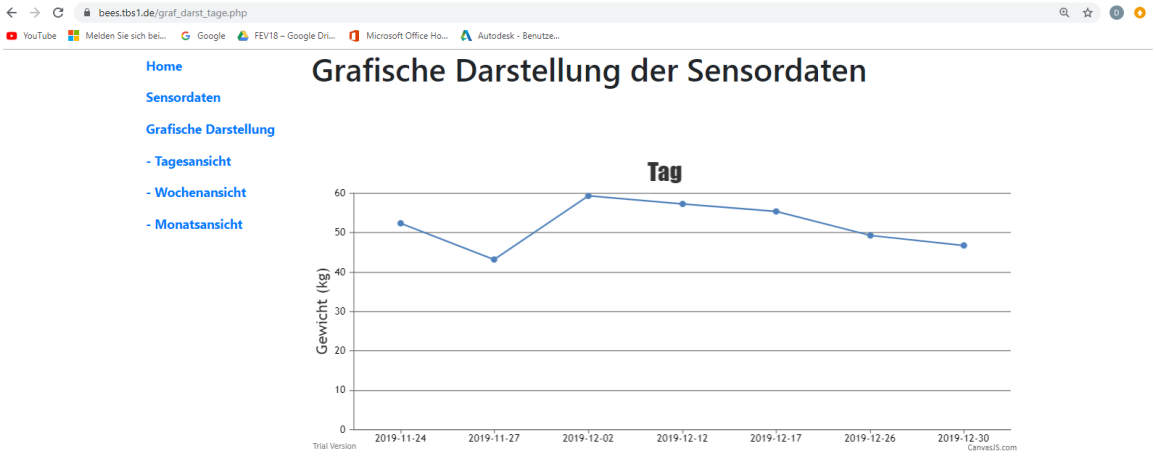


Abbildung 59 Ansicht der grafischen Darstellung (Webseite)

12. Messaufbauten

Für das Projekt wurden mehrere Messaufbauten durchgeführt, es wurden der aufgenommene Strom, die ordnungsgemäße Funktion des Prototyps unter Labor- sowie Realbedingungen und das Solarpanel auf Funktion gemessen.

12.1 Labormessung Stromaufnahme

Um festzustellen wie viel Strom die Schaltung benötigt, war es möglich, das Labor von Frau Riedel zu nutzen. Gemessen wurde der Strom im aktiven Modus sowie im Deep Sleep. Aus den Werten und der Nutzungsdauer des jeweiligen Zustandes ist es möglich, den am Tag benötigten Strom zu errechnen.

Um die Stromaufnahme so gering wie möglich zu halten, wurden Relais zum Schalten der Komponenten eingesetzt. Diese werden, wenn nötig mithilfe von Transistoren, durch den ESP32 geschaltet.

Um die geringste Stromaufnahme feststellen zu können wurden Entwicklerplatinen von verschiedenen Herstellern des ESP32-DevikitC im Einsatz mit der geplanten Schaltung gemessen:

- Firebeetle ESP32
- WeMos LoLin32
- ESP32-DevKitC (AZ-Delivery)

Da der Deep Sleep Modus einen Großteil der Betriebszeit der Stockwaage ausmacht, wurde bei den Labormessungen ein besonderes Augenmerk auf die Stromaufnahme der verschiedenen Entwicklerplatinen in diesem Betriebszustand gelegt.

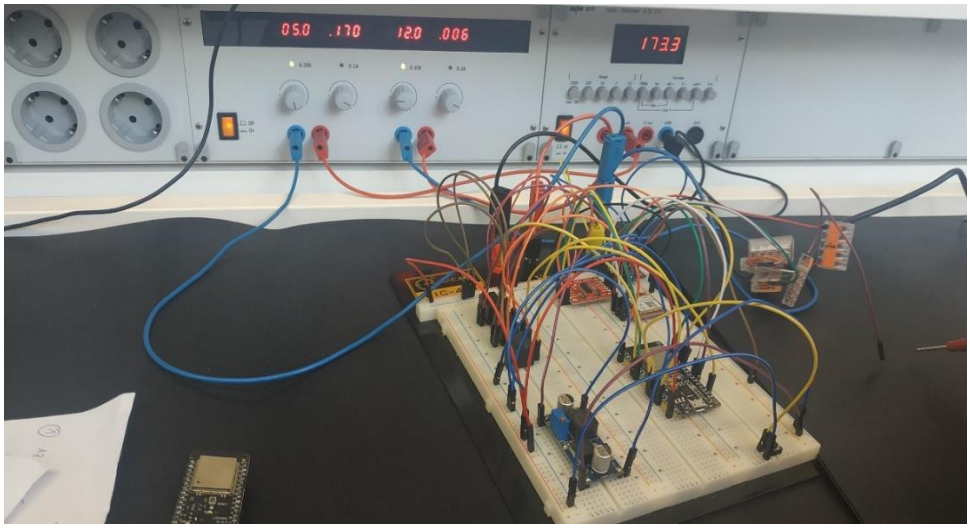


Abbildung 60 Messaufbau der Schaltung im Elektro-Labor der TBS1

Als sparsamste Ausführung, mit 6 mA im Deep Sleep und ca. 120 mA im aktiven Modus, wurde die DevKitC-Entwicklerplatine des Hersteller AZ-Delivery bestimmt. Der Firebeetle und auch der WeMos LoLin32 hatten, wenn sie über den jeweiligen Pin zur Einspeisung versorgt wurden, eine bis um den Faktor 10 erhöhte Stromaufnahme verglichen mit dem DevKitC vom Hersteller AZ-Delivery.

12.2 Testaufbau Realbedingung

Um die kompletten Funktionen, besonders die *SMS-Funktion*, des ESP32 unter realen Bedingungen zu testen, wurde der komplette Prototypenaufbau inkl. Netzteil im Freien aufgebaut und in Betrieb genommen. Hierbei zeigte sich, dass Umgebungsfaktoren, wie z.B. eine freie Sicht zum Himmel oder hohe Gebäudewände, welche das GSM-Signal abschirmen, einen starken Einfluss auf das erfolgreiche Versenden der Alarmierungs-SMS haben. Je nach Konstellation dieser Umgebungsfaktoren war es mit den, im Laborbetrieb voreingestellten, Verzögerungszeiten im Quellcode nicht möglich, eine stabile GSM-Verbindung aufzubauen. Hintergrund dieser Funktionsstörung war, dass die voreingestellte Verzögerungszeit zu kurz war und je nach Standort der Stockwaage individuell angepasst werden muss.

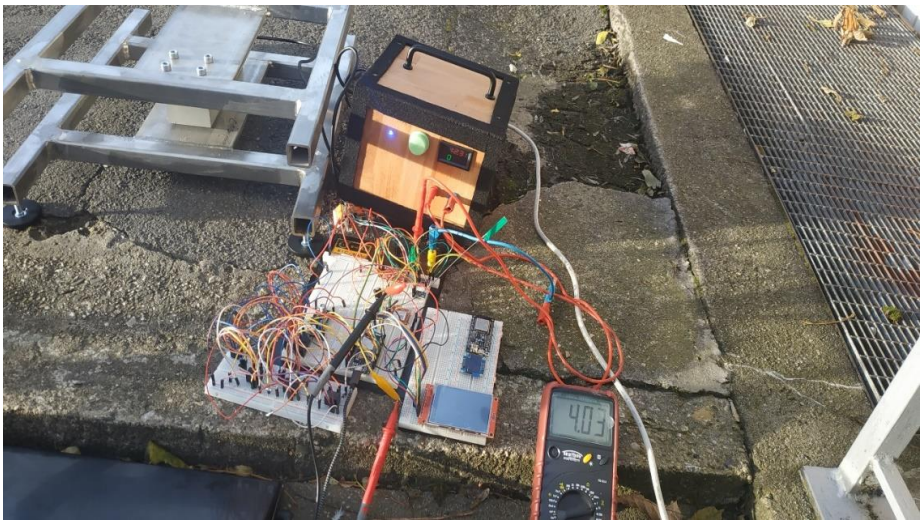


Abbildung 61 Messungen unter realen Bedingungen

12.3 Messaufbau Solarpanel

Das verwendete Solarpanel stammt aus den Beständen der TBS1. Um dieses auf Funktion, zu testen wurde es an einem sonnigen Tag an einen 10 kΩ-Lastwiderstand und zwei Messgeräte angeschlossen. Dies diente dazu die vom Panel erzeugte Spannung und Stromstärke feststellen zu können. Anschließend wurde der Messaufbau ins Feie getragen und auf Funktion überprüft. Das Solarpanel lieferte je nach Neigung, Intensität der Sonnenstrahlung und Einstellung des Lastwiderstandes unterschiedliche Werte im Bereich von 0-25 Volt.

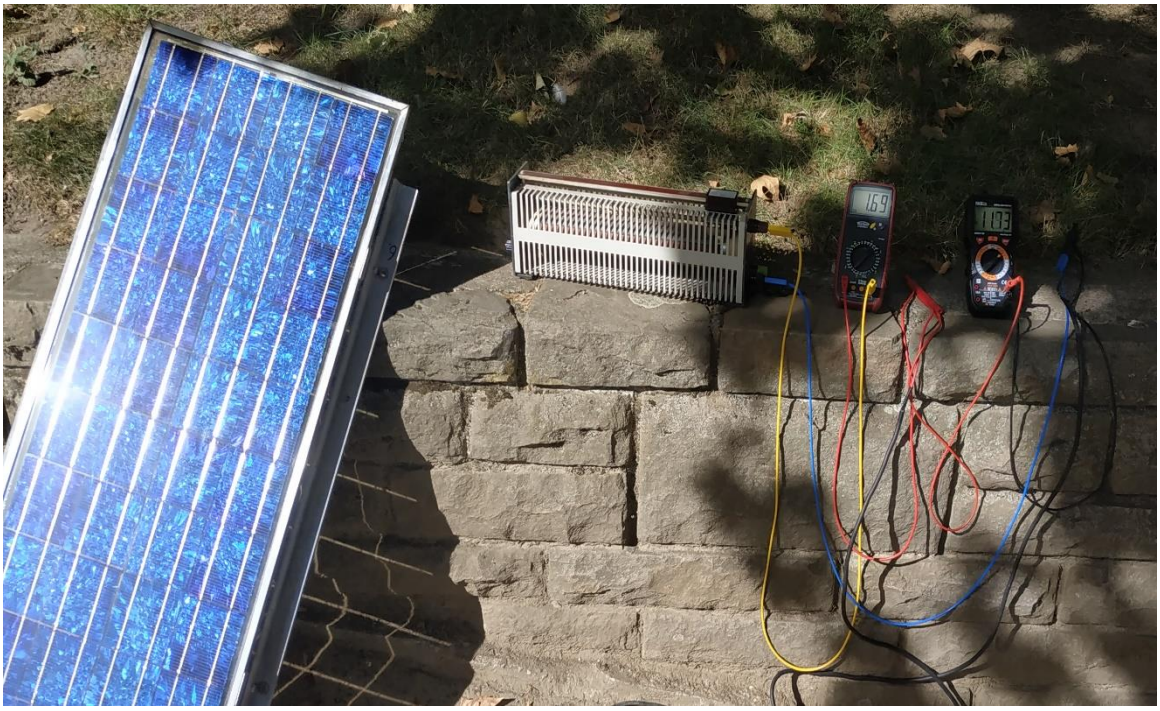


Abbildung 62 Messaufbau des Solarpanels

Aus den Mesergebnissen und der Tatsache, dass das Panel optisch für sein Alter noch in einem guten Zustand ist, ließ sich schließen, dass das Panel die Anforderungen des Projektes erfüllt und somit verwendet werden kann.

Am Tag der Messungen war das Wetter Sonnig mit teilweiser Wölkchen Bildung.

Anstellwinkel in Grad	**Spannung in V**
Ca. 20	11,73
Ca. 45	18,65
Ca. 60	25,46

13. Spannungsversorgung

Die Stockwaage steht auf dem Dach der TBS1 und hat somit keine Möglichkeit über die Stromversorgung des Gebäudes versorgt zu werden. Daher wurde eine Solar-Insellösung als Realisierung geplant. Dies bedeutet, dass die Stockwaage komplett autark, mithilfe eines Solarpanels, eines Ladereglers und einer Batterie, betrieben wird. Das Solarpanel wird aus den Restbeständen der TBS1 genutzt.

Für den Laderegler wurde ein Maximum Power Point Tracking (MPPT) Laderegler **[vgl. 20]** gewählt. Der MPPT-Laderegler beeinflusst den Lastwiderstand des Solarpanels so, dass das Solarpanel immer eine maximale Leistungsabgabe hat. Dieser Punkt ist am Solarpanel von der Sonneneinstrahlung und der aktuellen Temperatur des Moduls abhängig. Der optimale Arbeitspunkt des Solarpanels ist erreicht, wenn der Innenwiderstand des Panels gleich dem der Last ist, dies erfolgt durch eine elektronische Schaltung im Laderegler.

Um die Batteriekapazität auslegen zu können, wurde der gemessene Strom aus dem Messaufbau, mithilfe einer Excel-Tabelle auf einen Tagesbedarf umgerechnet und anschließend auf ca. vier Monate hochgerechnet. Diese Zeit sollte die Batterie überbrücken können, ohne geladen werden zu müssen, da es in den Winter Monaten möglich ist das keine zum Laden ausreichende Sonneneinstrahlung vorhanden ist. Die Berechnung befindet sich im Anhang.

Als Batterie wurde aus preislichen Gründen eine Blei-Säure-Batterie gewählt, bei dieser ist darauf zu achten, dass diese maximal zu 40-50 % entladen werden darf, da sie sonst an Lebensdauer verliert. Zusammen mit dem Auftraggeber Herrn Adams wurde die „Optima BT DC 5,5 Blue Top 75 Ah" gewählt. Durch die besondere Fertigung der Batterie ist der Innenwiderstand sehr gering und somit sind niedrige Temperaturen ein kleineres Problem als bei den üblichen Blei-Säure-Batterien. Dadurch hat die Batterie eine höhere Lebenserwartung.

Das Solarpanel soll auf dem Dach über der Turnhalle installiert werden, da die Batterie im Schatten des Solarpanels aufgestellt werden soll und die Position ca. 25 m von der Stockwaage entfernt ist, musste der zu erwartende Spannungsfall beachtet werden.

$$\Delta u = \frac{100 \times 2I \times L \times \cos\varphi}{\kappa \times A \times U} \quad \Delta u = \frac{200 \times 0{,}15A \times 25m \times 1}{56\frac{m}{\Omega mm^2} \times 2{,}5mm^2 \times 13{,}7V} = 0{,}39\%$$

Es wurde sich für einen Querschnitt von 2,5 mm bestimmt, da somit ein sehr geringer Spannungsfall gewährleistet wird.

14. Konstruktionen

Für das Projekt wurden mehrere Konstruktionen erstellet. Es wurden Zeichnungen für das Wiegegestell mit *AutoCAD* von Autodesk sowie für den 3D-Druck der Gehäuse mit Hilfe von *Fusion360* erstellt.

14.1 Wiegegestell

Bei der technischen Zeichnung wurde auf entsprechende Linienstärken und Arten geachtet. Die Zeichnung wurde als Normalprojektion mit drei Ansichten gezeichnet, der Draufsicht sowie zwei Seitenansichten.

Tabelle 4 Linienstärken in technischen Zeichnungen

Linienbreite/ Strichstärke	**Verwendung**
0,5 mm	Sichtbare Kanten und Umrisse
0,35 mm	Schrift, verdeckte Umrisse und verdeckte Kanten
0,25 mm	Schraffuren, Bemaßungen und Mittellinien

Tabelle 5 Linienarten in technischen Zeichnungen

Linienarten	**Verwendung**
Volllinie, schmal	Maßlinie, Maßhilfslinie und Lichtkanten
Volllinie, breit	Sichtbare Umrisse, sichtbare Kanten und Formteilungslinien
Strichlinie, schmal	Verdeckte Kanten und verdeckte Umrisse
Strichpunktlinie, schmal	Mittellinie und Symmetrielinien

Diese Grundlagen für technische Zeichnungen und die Abmessungen des zur Verfügung gestellten Beutebodens bilden die Grundlage für die zu erstellenden technische Zeichnungen des Wiegegestells.

Das Gestell besteht aus zwei Teilen, die durch die Wägezelle miteinander verbunden wurden. Jedes Teil des Gestelles besteht aus vier 4-Kant-V2A-Rohren und einem Stück V2A-Flachstahl. In den Flachstahl wurden vier Bohrungen eingezeichnet, in welche die Befestigungsschrauben der Wägezelle montiert wurden. Die Abmessungen für die Bohrungen sind dem Datenblatt der Wägezelle entnommen. Für eine klarere Ansicht sind die Materialen mit einer Schraffur kenntlich gemacht.

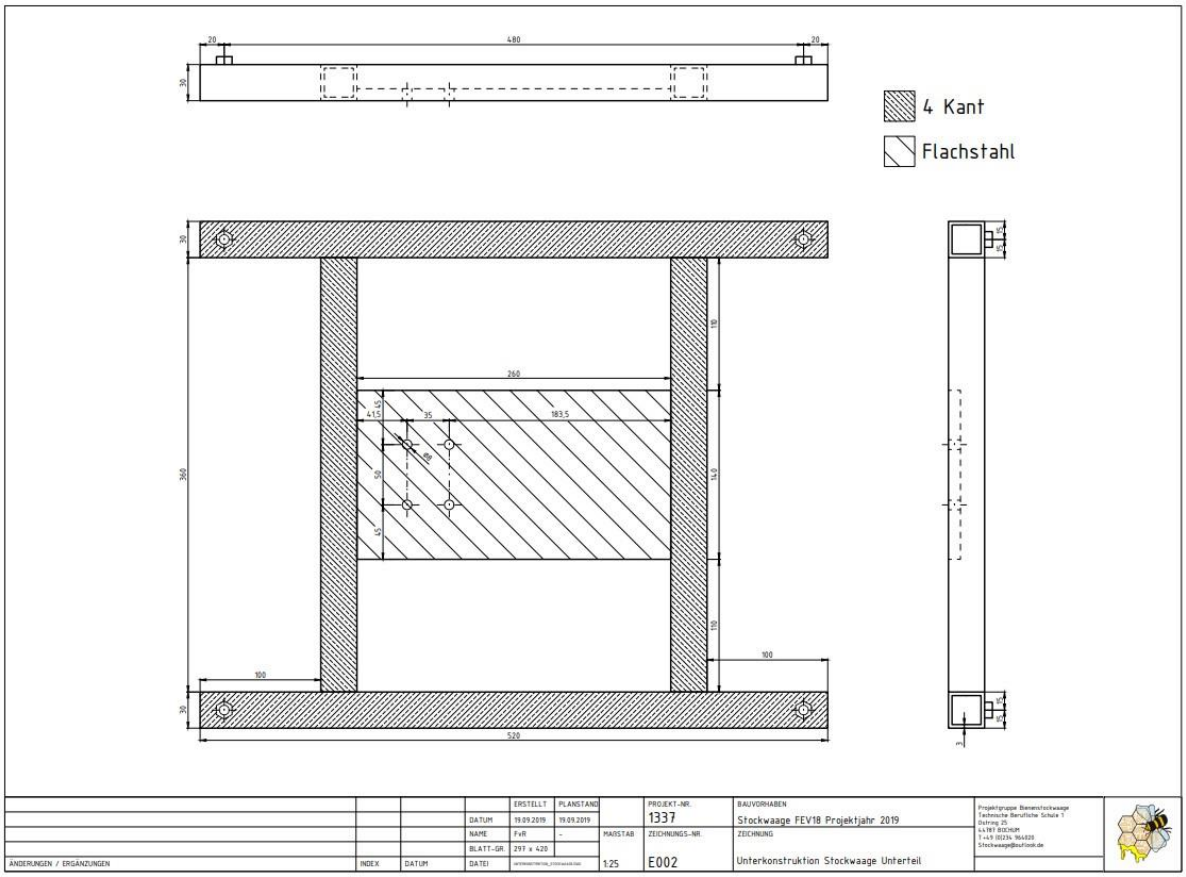


Abbildung 63 Unterkonstruktion Stockwaage 01

Das Material wurde zugeschnitten bestellt und anschließend der Zeichnung entsprechend geschweißt. Um eine spätere Montage von Standfüßen zu ermöglichen, wurden vier M8-Edelstahlmuttern an die Unterseite der Konstruktion geschweißt. Die Bohrungen in den Flachstahl-Stücken wurden mit Hilfe einer CNC-gesteuerten Fräsmaschine erstellt, da eine Maßhaltigkeit dieser Bohrungen Voraussetzung für die Montage der Wägezelle ist. Nach dem Schweißen wurden die Schweißnähte abgeschliffen und zum Schutz vor Korrosion mittels Flusssäure gebeizt.

Abbildung 64 CNC-gestützte Erstellung der Montagelöcher für die Wägezelle

Nach Fertigstellung der Bearbeitung wurden die beiden entstandenen Teile des Wiegegestells mit der Wägezelle und den Distanzplatten zusammengeschraubt. Als letzter Fertigungsschritt wurden die Standfüße montiert.

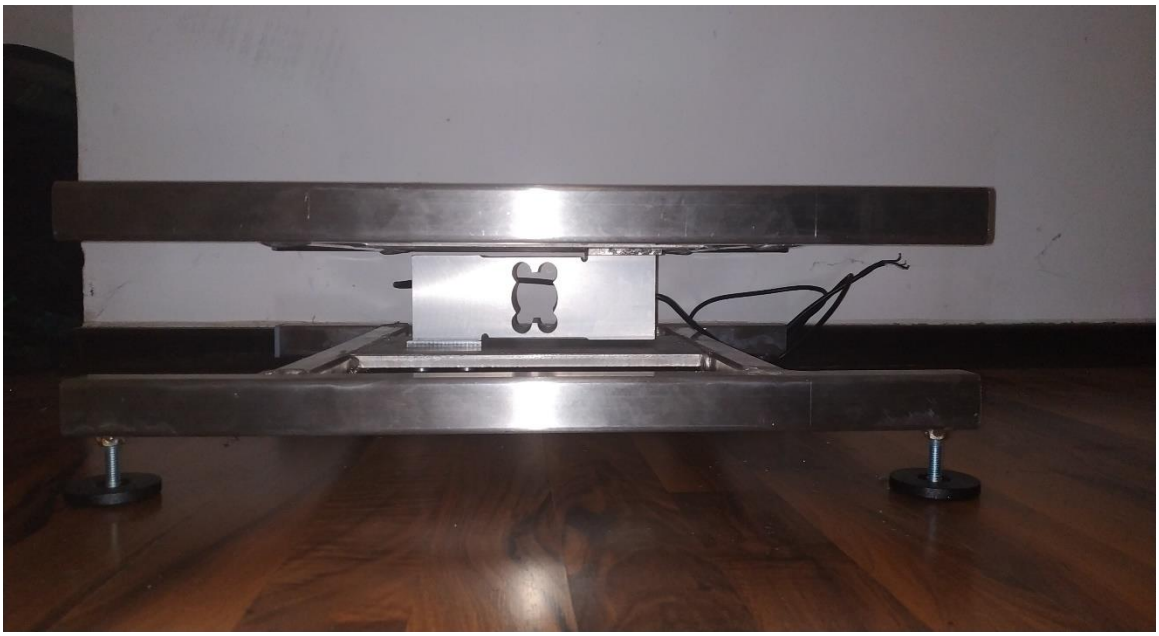


Abbildung 65 Fertig montiertes Wiegegestell inkl. Wägezelle

14.2 Gehäuse

Die Gehäuse für die Platinen wurden mittels 3D-Druck hergestellt, dafür wurden zwei Zeichnungen mit dem Programm *Fusion360* von Autodesk erstellt. Die Fusion360-Dateien wurden als slt.-Datei exportiert und in einen *Slicer* geladen. Als *Slicer* wurde Cura, eine Open-Source Software, verwendet. Cura zerschneidet die 3D-Zeichnung dann in Schichten, welche der 3D-Drucker Schicht für Schicht druckt. In Cura wurden auch alle Druck relevanten Einstellungen vorgenommen. Die durch den Anwender zu treffenden Voreinstellungen unterscheiden sich je nach Drucker und Filament sehr.

Für die Drucke stand der 3D-Drucker des Labor e.V. zu Verfügung. Da viele Einstellungen Erfahrungswerte sind, war es hilfreich, Mitglieder des Labors für die Einstellungen anzutreffen.

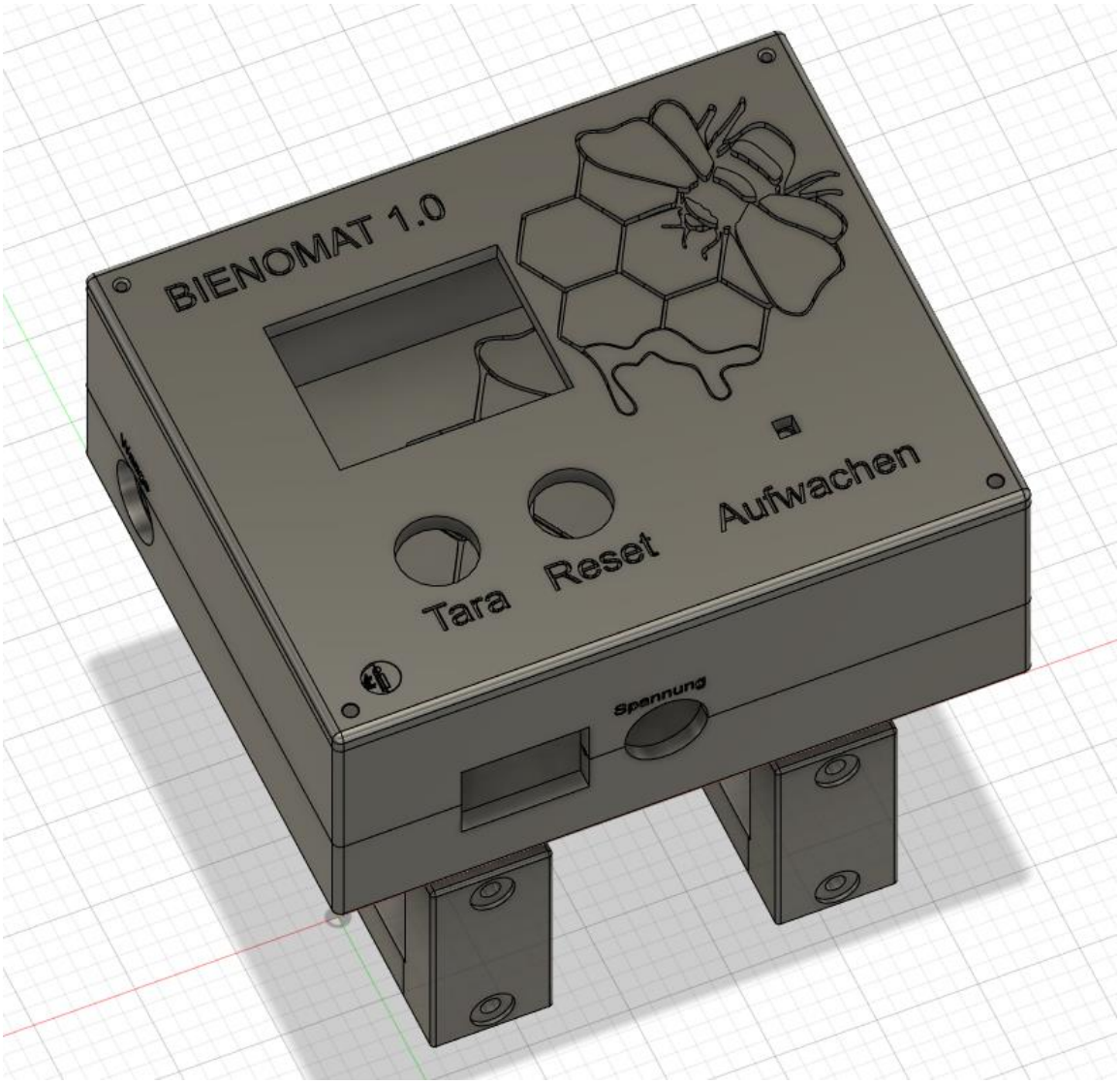


Abbildung 66 Bild des Gehäuses in Fuison360

Für die Gehäuse wurde PETG als Material verwendet, dieses Zeichnet sich durch folgende für das Projekt günstige Eigenschaften aus **[vgl. 19]:**

- Hohe Witterungsbeständigkeit
- Hohe UV-Beständigkeit
- Formstabil bis 70° C
- Hohe Dichte (1.38 g/cm³)

14.2.1 GY-BME280

Da der GY-BME280 extern agiert, wird dieser neben dem Hauptgehäuse mit einer Halterung an dem unteren 4-Kant-Rohr des Wiegegestells befestigt. Dementsprechend sind folgenden Arbeitsschritte vorgesehen:

- 4 Runde Extrusionen zum Auflegen der Platine
- 4 Bohrungen zum Befestigen der Platine
- 4 rechteckige Extrusionen zum Auflegen der Gehäuseabdeckung
- 4 Bohrungen zum Befestigen der Gehäuseabdeckung
- 2 Bohrungen zum Befestigen der Halterung
- 1 Bohrung für die Kabelverschraubung

Die Gehäuseabdeckung benötigt Lufteinlässe, damit der GY-BME280 die äußeren Klimawerte ermitteln kann. Dementsprechend sind folgenden Arbeitsschritte vorgesehen:

- 4 Bohrungen zum Befestigen der Gehäuseabdeckung
- 1 Extrusion zum Ausschneiden der Vorderseite
- 10 Extrusionen von Linien und Bögen zum Erstellen der Lamellen

Für die Befestigung der Gehäuseabdeckung wurden im Inneren des Gehäuses vier Rechtecke in die Eckpunkte gezeichnet und nach der Extrusion mit Bohrungen versehen.

Für die Größe des Gehäuses wurde eine Wandstärke von 3 mm verwendet, die durch die rechteckigen, 10 x 10 mm dicken Befestigungssäulen an den Eckpunkten verstärkt wird und somit eine Grundstabilität aufweist.

Die Vorderseite der Gehäuseabdeckung wurde ausgeschnitten, an der Innenseite ein Rechteck in einem 35-Grad-Winkel gezeichnet und 10-fach projiziert und dementsprechende Teile ausgeschnitten/projiziert. Danach wurden diese extrudiert, um als Lamellen den Lufteinlass zu gewährleisten und trotzdem die Platine vor Regen zu schützen.

Die Gehäusekomponenten wurden einzeln gedruckt, um entstehende Probleme beim Druckvorgang zu vermeiden:

- Gehäuse
- Gehäuseabdeckung
- Befestigung

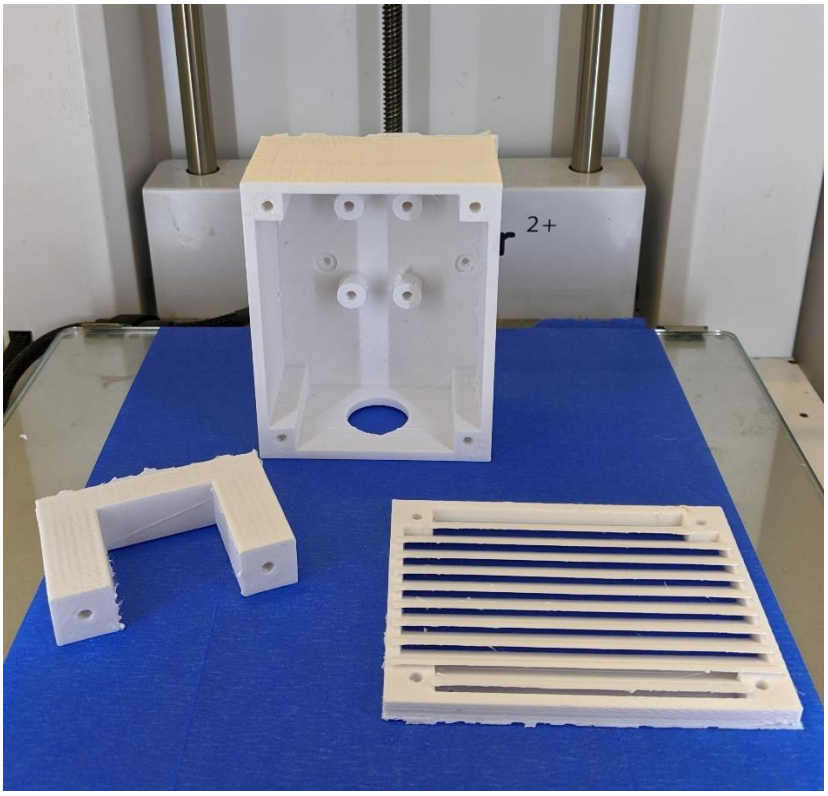


Abbildung 67 Gehäusekomponenten des BME280-Gehäuses

Danach wurden die Kanten der Gehäusekomponenten sauber geschmirgelt sowie von innen und außen mit Klarlack lackiert, um einen besseren Witterungsschutz zu gewährleisten.

In die Gehäusebohrungen wurden 4-mm-Gewindeeinsätze gepresst und die einzelnen Komponenten mittels 3,5-mm-Schrauben miteinander verbunden.

Zum Schluss wurde das Gehäuse an dem Wiegegestell befestigt.

14.2.2 Mikrocontroller mit Display

Das Hauptgehäuse wurde ebenfalls an dem unteren 4-Kant-Rohr des Wiegegestells befestigt, dafür wurden entsprechende Halter vorgesehen. Des Weiteren wurden für das Gehäuse folgende Bohrungen, Vertiefungen, Text und Extrusionen vorgesehen:

- 5 Runde Extrusionen mit Bohrungen zum Befestigen der Platine
- 2 Bohrungen zum Befestigen der Halter
- 3 Bohrungen für Kabelverschraubungen
- 1 Aussparung für den Hauptschalter
- Beschriftung für die Kabelverschraubungen
- Projektlogo als Vertiefung auf dem Gehäuse Boden

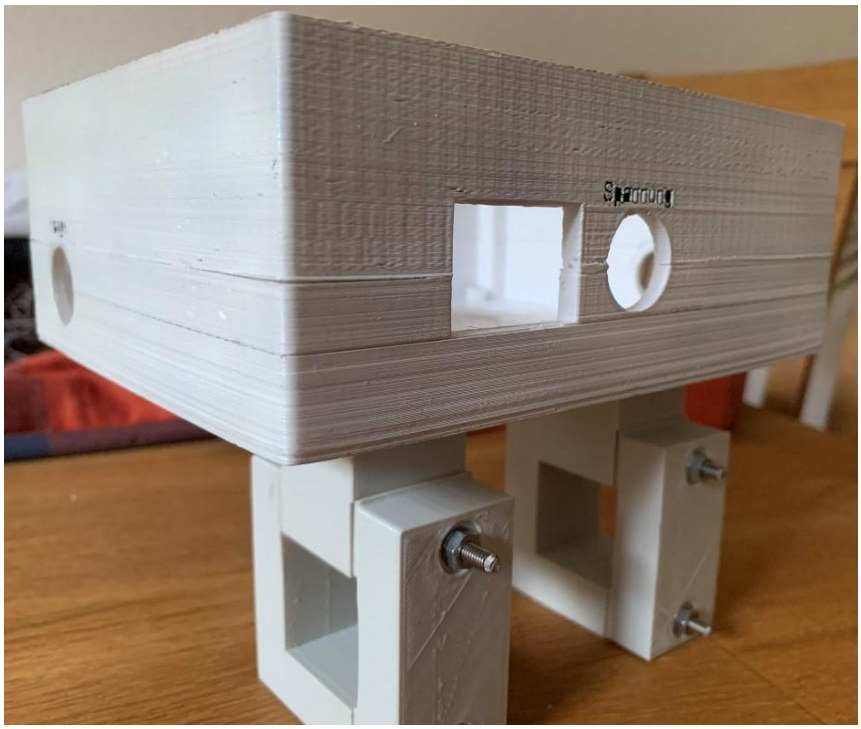


Abbildung 68 Gedrucktes Gehäuseunterteil inkl. Halterung

Für den Deckel wurden folgende Bohrungen, Vertiefungen für Text und Extrusionen vorgesehen:

- 1 Aussparung für das Display
- 2 Bohrungen für die Taster
- 1 Bohrung für den Aufwach-Button
- Projektlogo als Vertiefung
- Labor e.V.-Logo
- Beschriftung der Taster und des Buttons
- Name der Stockwaage „Bienomat 1.0“
- 4 runde Extrusionen zum Befestigen der Platine auf der Innenseite
- 4 Bohrungen an den Ecken zum Zuschrauben des Deckels am Gehäuse

Im Gehäuse wurden in den Ecken Stützen gezeichnet, in welche Bohrungen für den Deckel konstruiert sind. Um vergleichsweise IP65 zu erreichen wurde eine Nut in die Außenwand gedruckt, um dort eine Gummidichtung einzulassen.

Die Wandstärke wurde mit 5 mm festgelegt, um eine Grundstabilität zu gewährleisten. Bei der Wandstärke ist auf die maximale Breite der Verschraubung zu achten.

Um Stabilitäts-Problemen vorzubeugen, wurde das Gehäuse in mehreren einzelnen Druckvorgängen gedruckt:

- Halter zum Befestigen am Gestell
- Deckel
- Unterteil des Gehäuses
- Oberteil des Gehäuses

Das Unter- und Oberteil des Gehäuses wurden anschließend miteinander verklebt. Die Halter wurden am Gehäuse angeschraubt und das fertige Gehäuse anschließend an dem Wiegegestell befestigt.

Da der Deckel aus schwarzem PETG-Filament gedruckt wurde, wurde dieser anschließend extern farbig lackiert, somit ist auch die Lesbarkeit der einzelnen Beschriftungen gegeben.

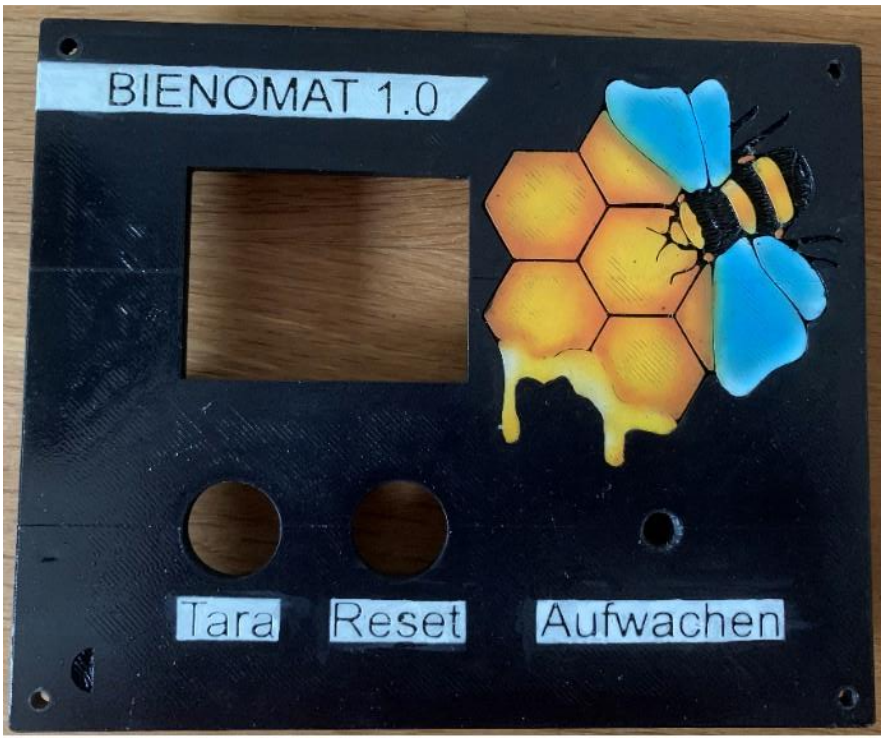


Abbildung 69 Gestalteter Deckel des Gehäuses

15. Messungen und Protokolle

Nach der erfolgreichen Inbetriebnahme der Stockwaage wurde ein Abnahmeprotokoll über die korrekte Funktion der Stockwaage angefertigt. Außerdem wurde die Prüfung von ortsveränderlichen Betriebsmitteln nach DGUV V3 durchgeführt und ein Protokoll darüber angefertigt. Sobald die Stadt Bochum die Freigabe zur Montage des Solarpanels und des dazugehörigen Rohrweges erteilt, wird noch eine Prüfung für ortsfeste Geräte nach DGUV V3 durchgeführt werden.

Die Protokolle für die oben erwähnten Messungen und Abnahme befinden sich im Anhang der Dokumentation.

Prüfprotokoll

Dokumentation der Prüfung ortsveränderlicher elektrischer Arbeitsmittel gemäß TRBS 1201 und § 5 der Unfallverhütungsvorschriften „Elektrische Anlagen und Betriebsmittel" (DGUV Vorschriften 3 und 4).

Betreiber:	Verantwortlich:
Arbeitsmittel:	Prüfling Name:
Fabrikat:	Schutzklasse:
Modell	Standort:
	Besonderheiten:

Datum	Bemerkung	Isolationswiderstand	Funktionsprobe	Bestanden:	Prüfperson

Abbildung 70 Prüfprotokoll zur ortsveränderlichen Prüfung

16. Herausforderungen

Im Verlaufe des Projektes traf die Projektgruppe auf mehrere Herausforderungen, welche alle erfolgreich bestanden wurden.

16.1 Programmierung

Da keiner der Projektteilnehmer nennenswerte Kenntnisse in den verwendeten Programmiersprachen und Bibliotheken hatte, boten die hardwarenahe Programmierung der verschiedenen Komponenten sowie die Entwicklung des PHP- und HTML-basierten Internetauftrittes sowie die Verwaltung und Einrichtung der Datenbank einige Herausforderungen, welche nachstehend näher erläutern werden.

16.1.1 Kalibrierung der Wägezelle/ HX711

Die Kalibrierung der Wägezelle erwies sich als zeitaufwändiger als geplant, da ein Skalierungsfaktor ermittelt und berechnet werden musste. Viele Anwendungsbeispiele, die zum Bestimmen des Skalierungsfaktors ausprobiert wurden, konnten nicht umgesetzt werden, da diese entweder für andere Mikrocontroller bestimmt waren oder ungenaue und falsche Messwerte lieferten. Nach ausführlicher Recherche wurde in einem Forum die Lösung gefunden. Dort hat ein User eine Formel für den Kalibrierungsfaktor eingestellt, welcher sich auf die technischen Eigenschaften der Wägezelle beziehen und nicht auf ein Referenzgewicht. **[vgl. 21, 108]**

$$Kalibrierungsfaktor = \frac{Output \times gain \times ADC_{Range} \times S\text{pannung der Wägezelle}}{Wägezellen\ Belastung_{Range} \times V_{cc}}$$

$$Kalibrierungsfaktor = \frac{2\frac{mV}{V} \times 128 \times 16777215 \times 3{,}3V}{150 \times 3{,}3V} = 28633{,}1136$$

- Output: Ausgangsleistung der Wägezelle
- gain: Verstärkungseinstellung des Verstärkers, standardmäßig bei einer angeschlossenen Wägezelle 128
- ADC_{Rang}: Wertebereich des ADC 2^{24}
- V_{cc}: Systemspannung für den Verstärker

Da der HX711 eine gewisse Temperaturabhängigkeit aufweist, die dafür sorgt das das gemessene Gewicht nicht mehr stimmt, musste noch eine Möglichkeit gefunden werden, diese zu kompensieren, nach längerer Recherche wurde eine Formel zur Temperaturkompensation gefunden. Dafür wird die Temperatur zum Zeitpunkt der Kalibrierung und die aktuelle Umgebungstemperatur benötigt. Da die Kalibrierung über die technischen Daten der Wägezelle stattfand, ist die Temperatur zum Zeitpunkt der Kalibrierung 20 °C. Die aktuelle Umgebungstemperatur liefert der GY21. **[vgl. 22, 108]**

$$m_{komp} = m_{roh} \times [1 \times c \times (T_{kal} - T_{Um})]$$

- m_{komp}: Gewicht nach Kompensation
- m_{roh}: Aktuell gewogenes Gewicht
- c: Kennwerttoleranz
- T_{kal}: Temperatur bei der Kalibrierung (da die Kalibrierung mit den technischen Daten stattfand, wurde hier 20° C gewählt)
- T_{Um}: aktuelle Umgebungstemperatur, gemessen vom GY21

16.1.2 SIM800L

Da die meisten Anwendungsbeispiele im Internet oder in der Bibliothek TinyGsm nicht auf den ESP32 bezogen waren, war es erst nach intensiver Recherche möglich, das SIM800L-Modul mit dem ESP32 anzusprechen. Da zwei von den drei gebotenen UART-Schnittstellen des ESP32 durch den internen Flash-Speicher und die Seriell/USB-Verbindung des ESP32 belegt sind, musste der SIM800L an der dritten UART-Schnittstelle angeschlossen werden. Die Befehle zur Verwendung dieser Schnittstelle waren allerdings nicht korrekt dokumentiert und so funktionierte diese Anbindung erst nach intensiver Recherche, einem Probeaufbau des SIM800L in Verbindung mit einem Arduino Uno (inkl. Programmierung) und dem ausgiebigen Lesen der internen Hardwarebibliotheken des ESP32.

16.1.3 GUI

Bedingt durch die Tatsache, dass die Möglichkeiten der GUI-Erstellung mit der LittlevGL-Bibliothek äußerst vielfältig sind, hat es mehr Zeit als ursprünglich geplant benötigt, sich angemessen in dieses Thema einzuarbeiten. Nur so war es möglich, die gewünschte Umsetzung der GUI inkl. Funktionen zu realisieren.

Eine weitere Hürde bei der Erstellung der GUI war, dass aus ungeklärten Gründen nach Hinzufügen der GUI-Funktionen und Bibliothek zum Bestands-Quelltext ein stabiler Betrieb des ESP32 nicht mehr möglich war. Aus diesem Grund wurden zwei Betriebszustände für den ESP32 programmiert. Zum einen gibt es den Normalbetrieb, in dem der ESP32 zyklisch erwacht, Messwerte ermittelt, diese zur Datenbank überträgt und ggf. eine Alarmierungs-SMS versendet. Nach

erfolgreicher Ausführung begibt der ESP32 sich in den *Deep Sleep-Modus*. Das Display wird in diesem Zustand weder initialisiert noch wird die Hintergrundbeleuchtung geschaltet.

Zum anderen gibt es den Betriebszustand des Bildschirmbetriebes, welcher durch ein Erwecken des EPS32 aus dem *Deep Sleep-Modus* über die *Aufwachen*-Berührungsfläche der Stockwaage aktiviert wird. In diesem Zustand wird nur der Touchscreen aktiviert. Die Sensoren und die Datenbankübertragung bleiben in diesem Betriebszustand deaktiviert.

Auch das Ändern der zur Alarmierung hinterlegten Mobilfunknummer über den Touchscreen dauerte länger als geplant. Da die Handynummer innerhalb eines *strings* mit weiteren Zeichen vom ESP32 an das GSM-Modul übergeben wird, musste eine Möglichkeit gefunden werden, diesen *string* in einzelne *strings* zu zerteilen, um einen der so entstandenen Teilstrings mit der vom Anwender über den Bildschirm eingegebenen Mobilfunknummer zu überschreiben. **[vgl. 108]** Anschließend müssen die drei Teilstrings wieder zu einem *string* zusammengefügt werden, um letztendlich im Flash-Speicher des ESP32 gespeichert zu werden. Von dort aus kann der Inhalt des dort gespeicherten *strings* abgerufen und von der *SMS-Funktion* genutzt werden.

16.1.4 Datenbank

Da Datenbanken ein umfangreiches Themengebiet behandeln, musste zuerst eine intensive Recherche betrieben werden, um den Aufbau, die Struktur, die SQL-Programmiersprache, als auch die Datenübertragung auf Webseiten zu verstehen.

Das erste Hindernis war die Datenübertragung von dem ESP32 in die Datenbank. Nach etlichen Versuchen und weiterer Recherche wurde klar, dass die Datenübertragung mit dem „WiFiClient“ aus der Wifi-Bibliothek des ESP32 über HTTP nicht möglich war, sondern eine zusätzliche Bibliothek Namens „HTTPClient“ verwendet werden musste. Nach der Einbindung dieser Bibliothek und der Anpassung der aufgerufenen URL funktionierte die Datenübertragung reibungslos.

Ein weiteres Hindernis war die Durchschnittswertermittlung der Datenbanktabelle „sensordaten“ über die SQL Query. Mittels der SQL AVG-Funktion konnte der Durchschnittswert aller Daten ermittelt werden, doch sollte dieser für die grafische Darstellung auf der Webseite jeweils pro Tag/Woche/Monat ausgegeben werden können. Durch die Hilfe von Herrn Adams konnte das Problem durch das Hinzufügen einer weiteren SQL-Funktion, die zur Tages-/Wochen/Monats-Abfrage dient, gelöst werden.

16.1.5 PHP/HTML

PHP und HTML sind ebenfalls umfangreiche Themen, die oftmals in Verbindung mit Datenbanken vorkommen. Auch in dieses Themengebiet erforderte zuerst eine intensive Einarbeitung, um den Aufbau und die Struktur der zu verwendenden Programmiersprachen und die Visualisierung von in Datenbanken gespeicherten Datensätzen auf Webseiten verstehen zu können.

Das erste Problem bestand darin, die Programmiersprache PHP korrekt in HTML zu implementieren, damit eine sinngemäße Datenübertragung und Darstellung der Sensordaten auf der Webseite gewährleistet wird. Anhand eines Stylesheets mussten zuerst die einzelnen PHP-Codes mit dem HTML-Code des Stylesheets zusammengesetzt werden. Dies erfolgte durch weitere Recherche über die Programmiersprachen und deren Funktionen, die wiederum viel Zeit in Anspruch nahm.

Ein weiteres Problem war die grafische Darstellung der Sensordaten auf einer Webseite. Durch weitere Recherche wurde klar, dass die übertragenen Daten in ein separat erstelltes Array abgespeichert werden müssen, damit sie dauerhaft aufrufbar sind und grafisch dargestellt werden können.

16.2 Schaltungsaufbau

Während der Testphase auf den Steckbrettern, traten nach einiger Zeit, nicht erklärbare Fehler auf. Der Ablauf ist vom Quellcode her immer gleich und wird sequenziell abgearbeitet. Dennoch wurden zum Teil Funktionen doppelt aufgerufen und ausgeführt, so das kurz hintereinander zwei Messsätze in die Datenbank geschrieben wurden.

Dies als bekannten Fehler im Internet, und somit evtl. eine Lösung zu finden, erwies sich als sehr langwierig.

Nach Messen der Eingangsspannung des ESP32 mittels eines Oszilloskops stellte sich heraus, dass diese im Fehlerfall zusammenbricht. Weiter ist es so, dass die Stromabgabe der GPIO-Pins wohl doch, anders als dem Datenblatt zu entnehmen war, teilweise nicht ausreichte um die Relais, die direkt an die GPIO-Pins angeschlossen waren, anzusteuern. Dies führte zu einem Spannungseinbruch am internen Spannungswandler des ESP32 wodurch unerwartetes Verhalten und Fehler im Programmablauf entstanden.

Daraufhin wurden Transistoren zum Schalten der Relais genutzt, um das Stromproblem zu lösen. Die Transistoren wurden separat mit 3,3V versorgt, zudem wurde an der Basis ein Vorwiderstand vorgesehen. Dieser wurde mittels des Datenblattes **[vgl. 107]** ermittelt. $V_{CE(sat)}$ ist erreicht, wenn pro 10 mA Collector-strom ein Basis-Strom von 0,5 mA verwendet. Dies bedeutet, dass, wenn die

Relaisspule einen Haltestrom von 30 mA aufweist, 1,5 mA Basisstrom benötigt werden. Daraus lässt sich bei 3,3 V Versorgungsspannung nun der Widerstandswert berechnen.

$$R = \frac{U - U_{BE}}{I} = \frac{3{,}3V - 0{,}7V}{1{,}5mA} = 1{,}7k\Omega$$

Nach der E12-Reihe wurde ein Widerstand mit 1,5 kΩ ausgewählt.

$$U_{BE} = \frac{R \times I}{U} = \frac{1{,}5k\Omega \times 1{,}5mA}{3{,}3V} = 0{,}68V$$

Anschließend wurde noch ein Spannungsabfall an dem SIM800l Modul gemessen, diesem wurde mit einem Kondensator entgegengewirkt. Nach ausführlicher Recherche im Datenblatt des SIM800L **[vgl. 103**] stellte sich heraus, dass dies mit einem 100-µF-Kondensator gelöst werden kann.

Bei der anschließenden Testphase traten keine weiteren Fehler auf der auf dem Steckbrett gesteckten Schaltung auf.

16.3 Konstruktionen/ 3D-Druck

Auch in dem Bereich der Gehäuseerstellung, mit Hilfe von verschiedenen 3D-Druckern, war eine intensive Einarbeitung nötig. Auch nach dieser Einarbeitung mussten mehrere Druckversuche durchgeführt werden, da jeder 3D-Drucker individuell auf das genutzte Filament abgestimmt und konfiguriert werden muss, was wiederum einen längeren Zeitraum in Anspruch nimmt, bis letztendlich ein akzeptables Ergebnis erreicht wird.

16.3.1 3D-Druck Filament

Da sich die Stockwaage im Außenbereich befinden wird, musste zuerst eine Recherche betrieben werden, damit ein geeignetes 3D-Druck Filament verwendet wird, das auch den äußeren Einflüssen standhalten kann. Zuerst wurde das ASA (Acrylester-Styrol-Acrylnitril) Filament ausgewählt, da es dem ABS (Acrylnitril-Butadien-Styrol) Filament sehr ähnelt und einige weitere ausschlaggebende Vorteile, wie z.B. eine höhere Materialdichte sowie Resistenz gegen UV-Strahlung und eine ausgezeichneten Witterungsbeständigkeit aufweist. Doch nach mehreren gescheiterten Druckvorgängen, die das Hauptgehäuse mit Rissen in der Gehäusewand versehen hatten, wurde das Material durch das PETG (Glykol modifiziertes Polyethylenterephthalat) Filament ersetzt. Dieses besitzt Vorteile, wie z.B. eine niedrige Viskosität, eine gute Witterungsbeständigkeit und es absorbiert kein Wasser. Durch das neue PETG Filament und die geänderten Druckeinstellungen konnte der Druckvorgang der einzelnen Gehäusekomponenten erfolgreich durchgeführt werden.

16.3.2 3D-Druck-Einstellungen

Da bei einem 3D-Druck-Verfahren eine Vielzahl von Konfigurationsmöglichkeiten erfolgen können, mussten zuerst einige Testdrucke vollzogen werden, um mit dem Filament einen technisch sowie optisch angemessenen und qualitativ hochwertigen Druck des Gehäuses zu erstellen. Nach mehreren misslungenen Druckvorgängen wurde das Material gewechselt und an folgenden Konfigurationen eine Änderung vorgenommen:

- Dicke einer gedruckten Schicht (Layerhöhe)
- Dicke der gedruckten Wandstärke (Shellstärke)
- Füllungsdichte (Infill)
- Einfahren des Filaments (Material)
- Druckgeschwindigkeit (Speed)
- Kühlung (Cooling)
- Hilfskonstruktion zur besseren Haftung (Build plate adhesion)
- Drucktemperatur

Nach diesen Änderungen konnte das Gehäuse problemlos gedruckt werden und im Vergleich zum ASA Filament eine robustere Struktur aufweisen. Dies entsteht durch die niedrigere Viskosität des PETG während des Druckvorganges zustande, wodurch das PETG Filament beim 3D-Druck eine flüssigere Struktur besitzt, sich dementsprechend besser mit den einzelnen Schichten verbinden kann und somit eine erhöhte Wasserdichtigkeit erreicht.

16.4 Platinen-Layouts

Das Erstellen der Platinen-Layouts war ein langwierigerer Prozess als zuerst angenommen. Bedingt durch Änderungen im Aufbau des Prototyps waren mehrfache Anpassungen des Layouts in Eagle nötig. Des Weiteren wies die auf Basis des Layouts gefertigte Platine Mängel auf.

Durch gute Teamarbeit und strukturelles Angehen der Probleme konnten alle Mängel behoben werden. Aus dem angepassten Prototyp entstand so letztlich eine funktionierende Platine.

17. Kosten

Für das Projekt gab es einen Kostenrahmen von ca. 300 bis 400 €, dieser wurde während des Projektes aufgrund umgesetzter optionaler Ziele aufgehoben. Die Komponenten wurden nach Qualität und Widerbeschaffbarkeit ausgewählt, um ein wartungsarmes-, langlebiges und reproduzierbares Produkt zu erhalten.

Tabelle 6 Stückliste der verwendeten Bauteile und Komponenten

Artikelbezeichnung	**Artikelnummer/ Zulieferer**	**Anzahl**	**Preis €**
Wägezellen-Verstärker HX711	ESP-R05-945 EXP Tech	1	10,53
Wägezelle H44A	BW1007.1 Bosche	1	104,13
Distanzplatten	BW1OO324 Bosche	1	11,70
TFT-Display (inkl. Touchfunktion)	B07L4RVP5L Amazon	1	16,29
GSM/GPRS-Modul SIM800L	A 11-4 AZ-Delivery	1	5,99
ESP32-DevKitC	A 2-7/ A 2-8 AZ-Delivery	1	7,49
DC/DC Step-Down-Converter	B 13-7 AZ-Delivery	1	2,99
Temperatursensor GY-21 HTU21	B 13-8 AZ-Delivery	1	4,99
MagI³C-FDSM Fixed Step Down Regulator Module 5 V	173010578 Würth	1	kostenfrei

Artikelbezeichnung	Artikelnummer/ Zulieferer	Anzahl	Preis €
MagI³C-FDSM Fixed Step Down Regulator Module 3,3 V	17301037 Würth	1	kostenfrei
WR-PHD 2.54 mm-Socket-Header 4 Pins	61300411821 Würth	3	kostenfrei
WR-PHD 2.54 mm-Socket- Header 5 Pins	61300511821 Würth	2	kostenfrei
WR-PHD 2.54 mm Socket-Header 6 Pins	61300611821 Würth	2	kostenfrei
WR-PHD 2.54 mm Socket-Header 10 Pins	61301011822 Würth	2	kostenfrei
WR-PHD 2.54 mm Socket-Header 14 Pins	61301411822 Würth	1	kostenfrei
WR-PHD 2.54 mm Socket-Header 20 Pins	61302011822 Würth	2	kostenfrei
WR-PHD 2.54 mm-THT-Pin-Header 1 Pin	61300111121 Würth	2	kostenfrei
WR-PHD 2.54 mm-THT-Pin-Header 2 Pins	61300211121 Würth	2	kostenfrei
WR-PHD 2.54 mm-THT-Pin-Header 13 Pins	61301311121 Würth	1	kostenfrei
Schottkeydiode 20 V/1 A DO-41	1N 5817 Reichelt	3	0,15
Vandalensicherer-Drucktaster, Ø 28,2/ 19 mm, 1 Schliesser (NO)	RND 210-00388 Reichelt	2	9,60

Artikelbezeichnung	Artikelnummer/ Zulieferer	Anzahl	Preis €
Lötbare Schraubklemme – 2-pol, RM 5,08 mm, 90°	RND 205-00232 Reichelt	1	0,25
Lötbare Schraubklemme – 4-pol, RM 5,08 mm, 90°	RND 205-00234 Reichelt	2	0,90
Lötbare Schraubklemme – 5-pol, RM 5,08 mm, 90°	RND 205-00235 Reichelt	1	0,55
40 pol. Stiftleiste, gerade, RM 2,54	SL 1X40G 2,54 Reichelt	1	0,21
Wippschalter, 2 x Aus, transparent weiß, I-O	WIPPE 1935.3113 Reichelt	1	5,50
RND Buchsenleiste, 8-pol, RM 2,54 mm	RND 205-00648 Reichelt	2	0,20
3 Volt 3 A Miniatur Printrelais JRC-21F 4100	421895 Makershop.de	3	2,91
MPPT Laderegler	MT1050 Prevent GmbH	1	42,00
Präzisions Buchsenleiste 1x20 Pol.	733755 Conrad	1	1,79
ESKV-Set	530820 Conrad	3	3,59
PETG Filament Weiß	1531386 Conrad	1	29,99

Artikelbezeichnung	Artikelnummer/ Zulieferer	Anzahl	Preis €
Transistor CBC547B	Labor e.V.	3	kostenfrei
Widerstand 7,4 kΩ	Labor e.V.	3	kostenfrei
Optima BT DC 5,5 Blue Top 75 Ah	TN1320 Autobatterienbilliger.de	1	198,85
Platinen	CCS-Challenge Card System GmbH	3	kostenfrei
4-Kant-Edelstahl 30 x 30 mm 3,2 m	CCS-Challenge Card System GmbH	1	kostenfrei
Solarpanel	TBS1	1	kostenfrei
	Summe		460,60

Die Kosten konnten stark reduziert werden dank unterstützendem Sponsoring. Für die Umsetzung des Projektes wurden weitere Produkte für Versuchsaufbauten benötigt, diese sind in der Stückliste im Anhang aufgelistet.

18. Erweiterungsmöglichkeiten

Im Laufe des Projektes entstanden folgende Ideen bezüglich Erweiterungsmöglichkeiten, welche dazu beitragen könnten, den Nutzen der Stockwaage für die TBS1 erhöhen:

- Erweiterung der Gewichtsmessung auf vier Beuten
- Erweiterung durch einen Infrarotsensor zum Zwecke der Bienenzählung
- Das Abspeichern der erfassten Messwerte lokal auf dem ESP32 oder einer per SPI-Schnittstelle zusätzlich eingebunden SD-Karte
- Die Möglichkeit schaffen den ESP32 per WLAN oder per Bluetooth mit neuen Programmen zu beschreiben
- Das Erfassen und Verarbeiten der Versorgungsspannung direkt an Batterie und Solarpanel inkl. Alarmierung bei kritischen Werten
- Erweiterung durch weitere Umweltsensoren, wie z.B. UV-, Lux-, oder Co2 Sensoren
- Das Installieren eines weiteren Mikrocontrollers, welcher mittels eines Kameramoduls Bilder des Beutebodens zum Zwecke der Varroa-Milben-Identifikation aufnimmt. Diese Bilder könnten z.B. durch eine einfache KI erfasst und ausgewertet werden
- Erweiterungen der GUI durch Verwendung weiterer Objekte aus der LittlevGL-Bibliothek, wie z.B. ein Liniengraph mit mehreren Messwerten des Gewichts- oder Klimasensors

19. Fazit

Über die gesamte Projektarbeit musste sich die Projektgruppe mit verschiedenen Problemen auseinandersetzen und Lösungen finden, damit die Projektanforderungen erfüllt sowie die Optionalen Ziele erreicht werden. Die im Zuge der Weiterbildung vermittelten Lerninhalte konnten dabei angewendet werden, um einen Lösungsweg zu finden oder einen Teil dazu beitragen.

Des Weiteren konnten tiefer Einblicke in den Fachbereich Informationstechnik, die Imkerei und in den Bereiche Naturschutz und Umwelterziehung gewonnen werden.

Außerdem mussten Lösungen für bisher nicht behandelte Themen und nicht vorhersehbare Probleme gefunden werden, die teils mit den Projektbetreuern besprochen wurden, um diese zu lösen.

Während des Projektes musste eigenständig, pflichtbewusst und termin-orientiert gearbeitet werden, wodurch auch verschiedene andere Kenntnisse erlernt wurden, wie zum Beispiel:

- Das Erstellen einer Dokumentation
- Kenntnisse der Mikroelektronik und Solartechnik
- Projektmanagementbezogene Aufgaben
- Kommunikation mit den Lieferanten
- Umgang und Lösung spontan auftretender Problemen

20. Danksagungen

Zuerst gebührt unser Dank Frau Heike Riedel und Herrn Reinhard Gummlich, welche uns und unsere Projektarbeit von Beginn bis Ende jederzeit unterstützt haben.

Auch möchten wir uns bei Herrn Hartmut Schreiber für das Einrichten einer eigenen Virtuellen Maschine inkl. Anwendungssoftware auf dem TBS1-Server bedanken.

Wir bedanken uns auch bei dem Förderverein der TBS1, der durch seine finanzielle Unterstützung dieses Projekt erst ermöglicht hat.

Der Firma Würth Electronics gilt unser Dank für die kostenlose Bereitstellung der benötigten Passiv-Bauteile im Bereich der Platinen-Bestückung.

Außerdem möchten wir uns bei der Firma CSS- Challenge Card Systems GmbH für die kostenlos zur Verfügung gestellten Edelstahlrohre für die Wiegekonstruktion und die Räumlichkeiten bedanken. Auch wäre eine Fertigung der Platinen in Industriequalität ohne die finanzielle Unterstützung der Firma CSS nicht möglich gewesen.

Auch der Firma Klein CNC Fertigung in Sprockhövel gebührt unser Dank für das kostenlose Schweißen und CNC-Bohren der Wiegekonstruktion.

Dem Verein „das Labor e.V“. in Bochum gilt unser besonderer Dank für die Bereitstellung der Möglichkeit des 3-D Drucks und der Beratung im Zusammenhang mit diesem Thema.

Wir möchten uns des Weiteren bei Kevin May für seine Unterstützung bei gestalterischen Fragen und der Erstellung des Logos für dieses Projekt bedanken.

Wir bedanken uns außerdem bei Jan Haferkamp für die Unterstützung bei der künstlerischen Verzierung des Stockwaagen-Gehäuses.

21. Quellverzeichnis

Internet

[1] https://de.wikipedia.org/wiki/ESP32

Verfasser: Wdwd, Stand Website 04.11.2019

[2] https://cdn.shopify.com/s/files/1/1509/1638/files/GY-21_Temperatur_und_Luftfeuchte_Sensor_Modul_Datenblatt_AZ-Delivery_Vertriebs_GmbH.pdf?9398

Verfasser: AZ-Delivery, Stand Website 04.11.2019

[3] https://randomnerdtutorials.com/esp32-pinout-reference-gpios/

Verfasser: randomnerdtutorials, stand Website 04.11.2019

[4] http://www.netzmafia.de/skripten/hardware/Strombelastbarkeit.html

Verfasser: Prof. Jürgen Klaus, stand Website 03.01.2015

[5] https://de.wikipedia.org/wiki/ESP32#/media/Datei:Espressif_ESP-WROOM-32_Wi-Fi_&_Bluetooth_Module.jpg

Ersteller: Brian Krent, Erstellt: 5. April 2017

[6] https://www.bosche.eu/waagenkomponenten/waegezellen/plattform-waegezellen/plattform-waegezellen-h44a

Verfasser: Bosche GmbH & Co. KG, Stand Website: 04.11.2019

[7] https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide/all

Verfasser: Sparkfun, Stand Website 04.11.2019

[8] http://www.netzmafia.de/skripten/hardware/Arduino/Waage/index.html

Verfasser: Prof. Jürgen Plate, Stand Website 20.08.2017

[9] https://rogerdudler.github.io/git-guide/index.de.html

Verfasser: Roger Dudler, Stand Website: 04.11.2019

[10] https://leanpub.com/git-flow/read

Verfasser: Leanpub, Stand Website: 04.11.2019

[11] https://www.electroschematics.com/introducingsim800l/

Verfasser: T.K. Hareendran, Stand Website 04.11.2019

[12] https://www.az-delivery.de/products/gy-bme280

Verfasser: AZ-Delivery, Stand Website 04.11.2019

[13] https://community.hiveeyes.org/t/zwischenpufferung-von-daten-und-log-informationen-im-rtc-slow-memory-des-esp32/2386/7

Verfasser: Verschiedene, Stand Website 14.10.2019

[14] https://www.az-delivery.de/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/esp32-daten-und-einstellungen-dauerhaft-speichern

Verfasser: AZ-Delivery, Stand Website 21.09.2019

[15] https://docs.espressif.com/projects/esp-idf/en/latest/api-guides/deep-sleep-stub.html#loading-data-into-rtc-memory

Verfasser: Espressif, Stand Website 14.10.2019

[16] https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/

Verfasser: LastMinuteEngineers.com, Stand Website 01.11.2019

[17] https://randomnerdtutorials.com/esp32-deep-sleep-arduino-ide-wake-up-sources/

Verfasser: RandomNerdTutorials, Stand Website 01.11.2019

[18] https://de.wikipedia.org/wiki/Network_Time_Protocol

Verfasser: Mehrere, Stand Website 06.11.2019

[19] https://www.filamentworld.de/3d-druck-wissen/was-ist-petg/

Verfasser: Filamentworld, Stand Website 06.11.2019

[20] https://de.wikipedia.org/wiki/Maximum_Power_Point_Tracking

Verfasser: Mehrere, Stand Website 06.11.2019

[21] https://github.com/bogde/HX711/issues/20

Verfasser: Bartimäus, Stand 14.12.2015

[22] https://community.hiveeyes.org/t/temperature-compensation-for-hx711-and-or-load-cell/1601/3

Verfasser: Thias, Stand 05.03.2019

[23] https://www.amazon.de/

Ersteller: Amazon, Stand 05.11.2019

[24] https://github.com/Bodmer/TFT_eSPI

Verfasser: Bodmer, Stand 05.11.2019

[25] https://github.com/littlevgl/lv_port_esp32_ili9341

Verfasser: embeddedt, Stand 05.11.2019

[26] https://community.hiveeyes.org/t/zwischenpufferung-von-daten-und-log-informationen-im-rtc-slow-memory-des-esp32/2386/26

Verfasser: Verschiedene, Stand 02.11.2019

[27] https://lastminuteengineers.com/esp32-sleep-modes-power-consumption/#esp32-active-mode

Verfasser: lastminuteengineers, Stand 02.11.2019

[28] https://www.az-delivery.de/blogs/azdelivery-blog-fur-arduino-und-raspberry-pi/esp32-daten-und-einstellungen-dauerhaft-speichern?ls=de

Verfasser: Albert Vu, Stand 10.10.2019

[29] https://www.arduino.cc/en/Reference/GSM

Verfasser: lastminuteengineers, Stand 04.10.2019

[30] https://github.com/vshymanskyy/TinyGSM

Verfasser: vshymanskyy, Stand 02.10.2019

[31] https://arduino-projekte.info/zisterne-fuellstandsanzeige-2-0-mit-wemos-lolin32-mysql-datenbank/

Verfasser: arduino-pro, Stand 12.10.2019

[32] https://arduino-projekte.info/zisterne-fuellstandsanzeige/

Verfasser: arduino-pro, Stand 12.10.2019

[33] https://randomnerdtutorials.com/esp32-esp8266-mysql-database-php/

Verfasser: randomnerdtutorials, Stand 08.10.2019

[34] https://www.w3schools.com/php/php_mysql_insert.asp

Verfasser: w3schools, Stand 11.10.2019

[35] https://wiki.selfhtml.org/wiki/HTML

Verfasser: selfhtml, Stand 08.10.2019

[36] https://www.filamentworld.de/3d-druck-wissen/was-ist-petg/

Verfasser: filamentworld, Stand 28.10.2019

[37] https://www.php.net/manual/de/pdo.prepared-statements.php

Verfasser: phpnet, Stand 14.10.2019

[38] https://www.php-einfach.de/mysql-tutorial/verbindung-aufbauen/

Verfasser: php-einfach, Stand 09.10.2019

[39] https://www.php-einfach.de/mysql-tutorial/phpmyadmin-datenbank-verwalten/phpmyadmin-tabelle-erstellen/

Verfasser: php-einfach, Stand 15.10.2019

[40] https://www.giga.de/downloads/microsoft-access-2010/tipps/was-ist-eine-datenbank-verstaendlich-erklaert/

Verfasser: giga, Stand 18.10.2019

[41] https://www.pixelfriese.de/unterschied-zwischen-utf8_general_ci-und-utf8_unicode_ci/

Verfasser: Thomas Lison, Stand 01.11.2019

[42] https://www.php-einfach.de/php-tutorial/php-array/#Mehrdimensionale_Arrays_definieren

Verfasser: php-einfach, Stand 31.10.2019

[43] https://www.php.net/manual/de/pdostatement.execute.php

Verfasser: phpnet, Stand 13.10.2019

[44] https://www.php-einfach.de/mysql-tutorial/daten-einfuegen/

Verfasser: php-einfach, Stand 12.10.2019

[45] https://www.youtube.com/watch?v=YqSU0yt19TI

Verfasser: Autodesk Eagle, Stand 29.10.2019

[46] https://getbootstrap.com/docs/4.0/components/navs/

Verfasser: getbootstrap, Stand 22.10.2019

[47] https://getbootstrap.com/docs/4.3/content/typography/

Verfasser: getbootstrap, Stand 23.10.2019

[48] https://canvasjs.com/php-charts/line-chart/

Verfasser: canvasjs, Stand 01.11.2019

[49] https://debiananwenderhandbuch.de/wasistdebiangnu.html

Verfasser: debian.org, Stand 23.09.2019

[50] https://de.wikipedia.org/wiki/Secure_Shell

Verfasser: Wikipedia, Stand 21.09.2019

[51] https://www.php.net/manual/de/function.array-push.php

Verfasser: phpnet, Stand 02.11.2019

[52] http://www.selfphp.de/praxisbuch/praxisbuchseite.php?site=157&group=28

Verfasser: selfphp, Stand 31.10.2019

[53] https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Positionierung#position

Verfasser: selfhtml, Stand 03.11.2019

[54] https://www.w3schools.com/css/

Verfasser: w3schools, Stand 03.11.2019

[55] https://www.mediaevent.de/css/width-height.html

Verfasser: mediaevent, Stand 03.11.2019

[56] https://wiki.selfhtml.org/wiki/CSS/Eigenschaften/Positionierung#position

Verfasser: selfhtml, Stand 03.11.2019

[57] https://wiki.selfhtml.org/wiki/Referenz:HTML/li

Verfasser: selfhtml, Stand 03.11.2019

[58] https://de.wikipedia.org/wiki/Datenbank

Verfasser: Wikipedia, Stand 19.10.2019

[59] https://de.wikipedia.org/wiki/MariaDB

Verfasser: Wikipedia, Stand 18.10.2019

[60] https://de.wikipedia.org/wiki/OpenVPN

Verfasser: Wikipedia, Stand 18.10.2019

[61] https://de.wikipedia.org/wiki/Debian#Nutzung_durch_%C3%B6ffentliche_Einrichtungen

Verfasser: Wikipedia, Stand 18.10.2019

[62] https://de.wikipedia.org/wiki/Nginx

Verfasser: Wikipedia, Stand 18.10.2019

[63] https://www.webhosterwissen.de/know-how/lexikon/mariadb-vs-mysql/

Verfasser: Wikipedia, Stand 18.10.2019

[64] https://www.dmsolutions.de/glossar/phpmyadmin.html

Verfasser: DMSolutions, Stand 18.10.2019

[65] https://de.wikipedia.org/wiki/PhpMyAdmin

Verfasser: DMSolutions, Stand 18.10.2019

Dokumente

[101] ESP_-_32_NodeMCU_Developmentboard_Datenblatt.pdf

Verfasser: AZ-Delivery, Download am 01.09.2019

[102] Hilfe-bei-der-Auswahl-von-Waegezellen.pdf

Verfasser: Bosche GmbH & Co. KG, Download vom 04.11.2019

[103] Datasheet_SIM800L.pdf

Verfasser: SimCOM, 20.08.2013, Download vom 01.09.2019

[104] GY-21_Temperatur_und_Luftfeuchte_Sensor_Modul_Datenblatt_AZ
Verfasser: Delivery_Vertriebs_GmbH

Verfasser: AZ-Delivery, Download am 01.09.2019

[105] GY-BME280_Datenblatt

Verfasser: AZ-Delivery, Download am 01.09.2019

[106] 2.8inch_SPI_Module_MSP2807_User_Manual_EN.pdf

Verfasser LCD-Wiki, Download am 10.09.2019

[107] BC547B Datenblatt

Mouser Electronics, Dowlnload 08.11.2019

[108] Datenblatt Wägezelle

Bosche Wägetechnik, Download am 08.11.2019

[109] Arduino-Referenz.pdf

Arduinoforum.de, Download am 01.11.2019

[110] LittlevGL Documentation.pdf

LittlevGL, Download am 10.11.2019

22. Anhangsverzeichnis

Zeichnungen

- Blockschaltbild
- Anschlussplan der Komponenten
- Wägegestell 01
- Wägegestell 02

Schaltpläne

- BME280
- Display
- Hauptplatine

Lastenheft

Pflichtenheft

Stückliste gesamt

Berechnung Batterieauslegung

Inbetriebnahmeprotokoll

Prüfprotokoll ortsveränderlicher elektrischer Arbeitsmittel

Wochenbericht

Quelltexte

Zeitplanung

Datenträger mit sämtlichen Datenblättern, Quelltexten, Bibliotheken, und Konstruktionsdateien, welche bei diesem Projekt verwendet bzw. erstellt wurden.

23. Eigenständigkeitserklärung

Hiermit versichern wir, dass wir die vorstehende Projektarbeit in allen Teilen (Projektgegenstand, Dokumentation, Präsentation usw.) selbstständig angefertigt, keine anderen Quellen und Hilfsmittel als die angegebenen benutzt und die Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht haben.

Bochum, _____

Unterschriften der Projektteilnehmer:

_____

Maximilian Döring

_____

David Saric

_____

Frederic von Rüden