Staatliche Fachschule für
Mechatronik- und Elektrotechnik

Entwicklung und Realisierung eines erweiterbaren Systems zur Erfassung und Verarbeitung von Sensordaten

von

Jonas Schönberger

3. April 2020

Projektarbeit
Staatliche Fachschule für Mechatronik- und Elektrotechnik
Elektro Leipold GmbH & Co. KG

Betreuer:
Margit Liegl (OStR),
Karl Leipold

Inhaltsverzeichnis

1. Einleitung **5**
1.1 Vorwort . . . 5
1.2 Firmenvorstellung Elektro-Leipold GmbH . . . 5
1.3 Aufgabenstellung . . . 6

2. Projektplanung **7**
2.1 Vorhandene Anlage . . . 7
2.2 Messpunkte und Messgrößen . . . 7
2.3 Konzept . . . 8

3. Komponentenauswahl **9**
3.1 ESP32 Mikrocontroller . . . 9
3.2 Sensoren . . . 10
3.2.1 Temperatursensoren . . . 10
3.2.2 Drucksensoren . . . 12
3.2.3 Stromsensoren . . . 13
3.3 Verwendeter Server . . . 13
3.4 Serversoftware . . . 14
3.4.1 MQTT Protokoll . . . 14
3.4.2 Node-Red . . . 15
3.4.3 MySQL-Datenbank . . . 15

4. Durchführung **16**
4.1 Elektrische Komponenten . . . 16
4.1.1 Anforderungen . . . 16
4.1.2 Schaltplan . . . 17
4.1.3 Mehrzweckplatine . . . 20
4.1.4 Spannungswandler Platine . . . 21
4.1.5 Verdrahtung . . . 22
4.2 Mechanische Komponenten . . . 23
4.2.1 Gehäuse . . . 23
4.2.2 Sensorhalterungen . . . 24
4.3 Programmierung Platine . . . 27
4.3.1 Toolchain . . . 27
4.3.2 Dateistruktur . . . 27
4.3.3 Programmaufbau . . . 27
4.3.4 Konfiguration und Sensordeklaration . . . 28
4.3.5 Temperatursensoren . . . 29
4.3.6 Digitaleingänge . . . 30
4.3.7 Analogsensoren . . . 31
4.3.8 WLAN . . . 32

4.3.9 MQTT . . . 33
4.4 Programmierung Node-Red . . . 34
4.4.1 Daten empfangen und speichern . . . 34
4.4.2 Live-Anzeige . . . 35
4.4.3 Sensorverläufe . . . 35
4.4.4 E-Mail Benachrichtigung . . . 37

5. Fazit **39**
5.1 Erkenntnisse Heizung . . . 39
5.2 Kosten . . . 40
5.2.1 Materialkosten . . . 40
5.2.2 Laufende Kosten . . . 40
5.2.3 Speicherbedarf . . . 41
5.3 Zukünftige Erweiterungen . . . 42

6. Quellen **45**

Anhang **47**

A Schaltplan Platine **48**

B Layout Platine **50**

C Anschlussplan **51**

D MQTT Struktur **52**

E Weboberfläche - Charts **53**

F Stücklisten **54**
F.1 Stückliste Projekt . . . 54
F.2 Stückliste Platine . . . 55

G Technische Zeichnungen **56**
G.1 Drucksensoradapter gerade . . . 56
G.2 Drucksensoradapter abgewinkelt . . . 57
G.3 Temperatursensorhalter . . . 58

H Tätigkeitsnachweis **59**

I Selbstständigkeitserklärung **63**

1. Einleitung

1.1 Vorwort

Im Rahmen der Weiterbildung zum staatlich geprüften Techniker am beruflichen Schulzentrum in Amberg führt jeder Schüler eine Projektarbeit durch. Ziel dieser Projektarbeit ist es, das erlernte Wissen durch eine praxisnahe Aufgabenstellung in den Bereichen Mechanik, Elektrotechnik und Informatik anzuwenden und selbstständig Lösungswege für komplexe Probleme zu finden. In einem Zeitraum von mindestens drei Wochen soll diese Arbeit durchgeführt, dokumentiert und anschließend präsentiert werden.
Diese Projektarbeit wurde in Zusammenarbeit mit der Firma Elektro-Leipold GmbH & Co. KG in Mitterteich durchgeführt. Durch eine fächerübergreifende Aufgabenstellung wurde ein System geplant, Komponenten ausgewählt, Bauteile gefertigt sowie Software programmiert.

1.2 Firmenvorstellung Elektro-Leipold GmbH

Die Firma Leipold wurde im Jahr 1991 als Einzelunternehmen gegründet. Im Jahr 2007 ist die Firma in das aktuelle Betriebsgebäude in Mitterteich umgezogen.
Derzeit werden dort ca. 20 Mitarbeiter als Elektriker beschäftigt und zusätzlich Jugendliche als Elektriker oder IT-Systemkaufmann ausgebildet. [1]
Die Elektro-Leipold GmbH & Co. KG ist überwiegend in der Industrie tätig und führt dort Elektroinstallationsarbeiten in verschiedensten Bereichen, wie z. B. Mess-, Steuerungs- und Regelungstechnik durch.
Auch auf Prozessoptimierung, Prüfen und Messen von Sicherheitseinrichtungen sowie Explosionsschutz hat sich das Unternehmen spezialisiert.

Abbildung 1.1: Firmeneingang Leipold. [1]

1.3 Aufgabenstellung

Für ein komplex erscheinendes Zentralheizungssystem soll eine Möglichkeit gefunden werden, dieses zu überwachen und Vorgänge zu analysieren. Es soll ein Netzwerk aus Sensoren aufgebaut werden, mit dem verschiedene physikalische Größen, wie Temperatur, elektrische Leistung oder Druck an verschiedenen Stellen der Heizungsanlage gemessen werden können.

Abbildung 1.2: Vorhandene Zentralheizungsanlage

Die Sensordaten sollen sowohl live angezeigt als auch gespeichert werden, damit Sensordatenverläufe rückwirkend in einem wählbaren Zeitraum angezeigt werden können. Dadurch soll es möglich werden die ablaufenden Prozesse zu verstehen und eventuelle Fehler zu erkennen.
Außerdem ist es wichtig, dass das System möglichst erweiterbar gestaltet ist, sodass es ohne viel Aufwand möglich ist, weitere Sensoren einzubinden.

2. Projektplanung

2.1 Vorhandene Anlage

Vor Beginn der Projektarbeit wurde das vorhandene Heizungssystem inspiziert. Es handelt sich um eine Zentralheizungsanlage für ein 3-Familienhaus mit einer 30 kW Scheitholzheizung und einer 24 m^2 Solaranlage. Mithilfe dieser beiden Systeme wird Wasser in einem 3700 L Pufferspeicher erhitzt und für drei Heizkreise verwendet:

- HK1: Heizkörper im ersten und zweiten Obergeschoss
- HK2: Brauchwassertank für Warmwasser (Wärmetauscher)
- HK3: Fußbodenheizung im Erdgeschoss

Um einen besseren Überblick über das vorhandene System zu erhalten, wurde anhand vor Ort angefertigter Skizzen ein vereinfachtes Systemschema erstellt:

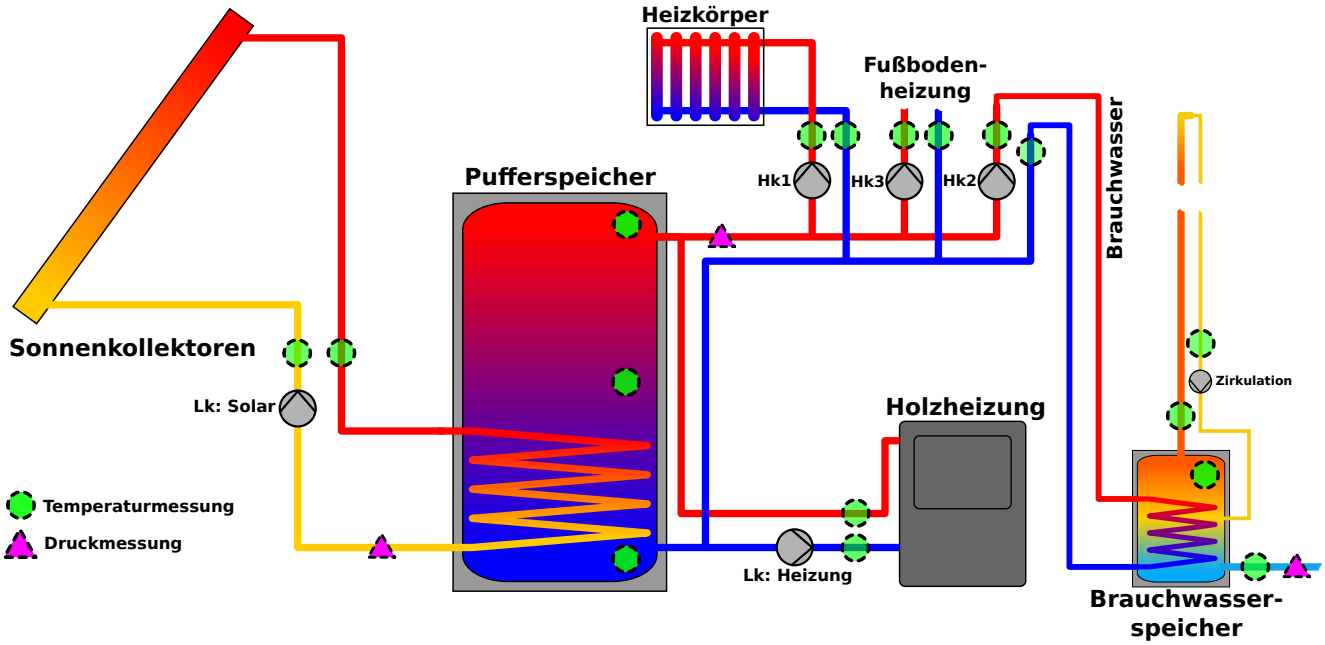


Abbildung 2.1: Systemschema der Zentralheizungsanlage

2.2 Messpunkte und Messgrößen

Anhand der in 2.1 erhaltenen Übersicht konnte eine sinnvolle Anordnung von Sensoren festgelegt werden. Diese Messstellen wurden anschließend im Systemschema mit Kreisen bzw. Dreiecken markiert (siehe Abbildung 2.1).
Es soll von jedem Heizkreis, sowie den beiden Ladekreisen (Solar und Heizung) jeweils

Vorlauf- und Rücklauftemperatur gemessen werden. Zusätzlich soll die Pufferspeicher-, Warmwasser-, Kaltwasser-, und Abgastemperatur der Holzheizung aufgenommen werden.
Des Weiteren sind die Wasserdrücke im Solar-, Heiz-, und Kaltwassersystem sowie die Leistung bzw. Zustände sämtlicher Pumpen auszulesen.

2.3 Konzept

Als zweiter Schritt in der Projektplanung wurde ein grobes Konzept ausgearbeitet, welches beschreibt, wie das geplante System aufgebaut ist. In Abbildung 2.2 sieht man die Anordnung und den Datenfluss zwischen den wichtigsten Komponenten.
Anstatt fertige Industriekomponenten, wie z. B. einer SPS und Industriesensoren zu nutzen, soll ein eigenes, kostengünstiges System mit weit verbreiteter Verbraucherelektronik entwickelt werden. Deshalb war geplant eine einheitliche Platine zu entwickeln, mit der es möglich ist, verschiedenste Sensoren auszuwerten oder auch kleine Steuerungsaufgaben auszuführen. Solche Platinen, welche in beliebiger Anzahl vorhanden sein können, sollen mittels WLAN in das bereits vorhandene lokale Netzwerk eingebunden werden. Über dieses Netzwerk werden die aktuellen Sensordaten laufend an einen zentralen Server übermittelt.
Der Server soll die Sensorwerte in einer Datenbank speichern und diese Daten in verarbeiteter Form für verschiedene Endgeräte zur Verfügung stellen. Somit soll es zum Beispiel möglich sein, die aktuellen oder frühere Sensorwerte über eine Website zu betrachten.

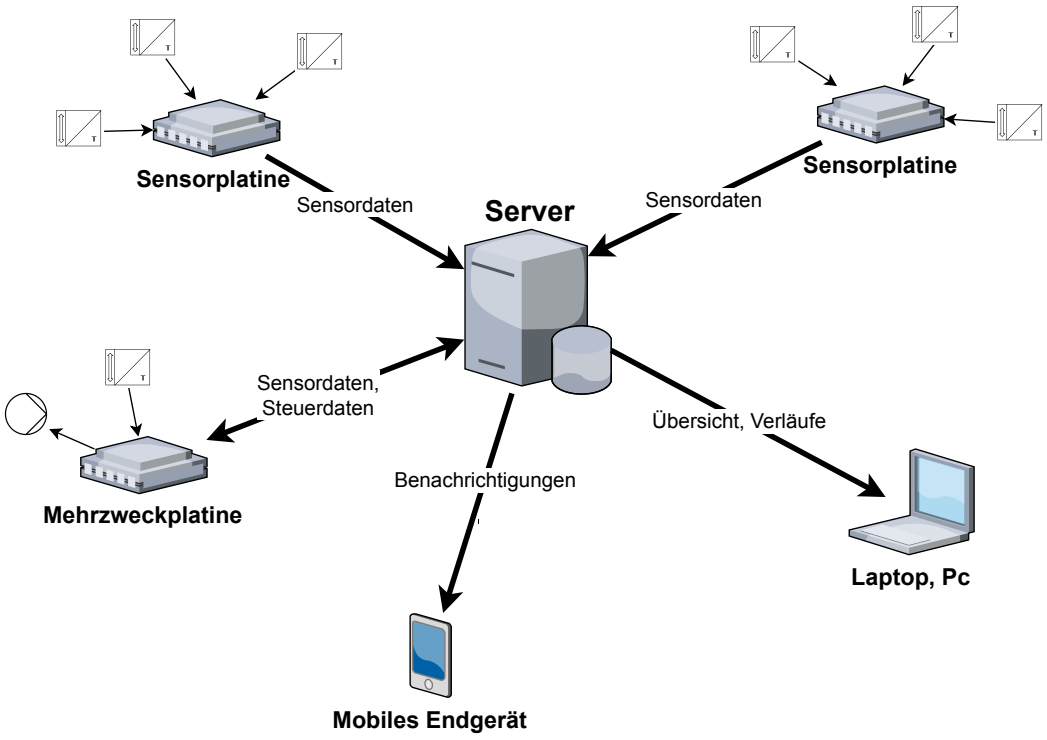


Abbildung 2.2: Aufbau des geplanten Systems

3. Komponentenauswahl

Für die Projektarbeit wurden einige spezielle Komponenten benötigt und ausgewählt. In diesem Kapitel werden die wichtigsten davon genauer erläutert.

3.1 ESP32 Mikrocontroller

Definition Mikrocontroller:

> „Als Mikrocontroller (auch µController, µC, MCU) werden Halbleiterchips bezeichnet, die einen Prozessor und zugleich auch Peripheriefunktionen enthalten. In vielen Fällen befindet sich auch der Arbeits- und Programmspeicher teilweise oder komplett auf demselben Chip. Ein Mikrocontroller ist ein Ein-Chip-Computersystem. Für manche Mikrocontroller wird auch der Begriff System-on-a-Chip oder SoC verwendet“ [2].

Für die geplante Platine wird ein Controller benötigt, der das Auslesen verschiedenster Sensoren, Durchführen von Steuerungsaufgaben, sowie die Integration in das WLAN ermöglicht. Der ESP32 ist im Vergleich zu fertigen Steuerungen ein sehr kostengünstiger und bekannter Mikrocontroller mit vielen Funktionen.
Wie man in Abbildung 3.1 erkennen kann, ist der Prozessor selbst auf einem kleinen Modul mit Flash Speicher und WLAN Antenne untergebracht. Dieses Modul trägt die Bezeichnung ESP-WROOM-32Bit und ist ein Bauteil der „surface mount technology“ (SMT). Somit kann es später direkt auf die Oberfläche der geplanten Platine gelötet werden.
Aus dem Datenblatt[1] können die wichtigsten technischen Daten dieses Moduls entnommen werden (siehe Tabelle 3.1). So sind zum Beispiel insgesamt 36 GPIO-Pins vorhanden. GPIO steht für general purpose input output, das heißt, diese Pins können wahlweise für verschiedenste Anwendungen genutzt werden. Die meisten dieser Anschlüsse können also, neben anderen speziellen Funktionen, als Digitaleingänge sowie als Digitalausgänge benutzt werden. Diese sind für digitale Sensoren sowie für Steueraufgaben gut zu gebrauchen.

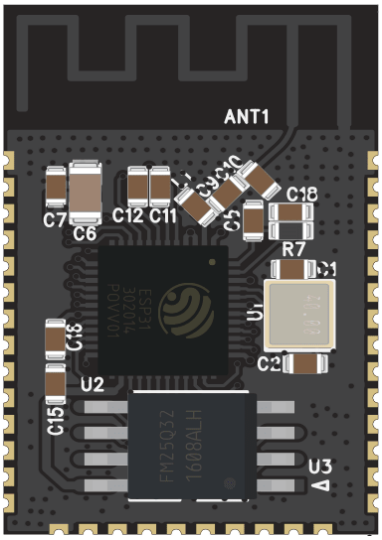


Abbildung 3.1: ESP32 WROOM Aufbau [3]

Bis zu 18 dieser GPIO-Pins, mit integriertem Analog-Digital-Converter (ADC), können außerdem als Analogeingang genutzt werden. Mit einer Auflösung von 12 Bit sind diese sehr nützlich um häufig vorkommende Analogsensoren auswerten zu können.
Aber vor allem die bereits integrierte WLAN Funktion macht diesen Mikrocontroller ideal für die geplante Anwendung.

[1] https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Tabelle 3.1: Technische Daten ESP32 [4]

Spannung	3,3 V
CPU	Xtensa Dual-Core 32-Bit
Taktfrequenz	160 bis 240 MHz
RAM	512 kB
Flash Speicher	16 MB
GPIO-Pins	36
Betriebssystem	FreeRTOS
Analogeingänge (ADC)	18 (12 Bit)
Analogausgänge (DAC)	2
WLAN	IEEE 802.11 2,4 GHz
Schnittstellen	UART, SPI, CAN, I2C

3.2 Sensoren

3.2.1 Temperatursensoren

Da hier eine Heizungsanlage überwacht werden soll, muss sehr häufig die Temperatur gemessen werden. Deshalb sind Temperatursensoren ein sehr wichtiger Bestandteil dieser Projektarbeit.

DS18B20 Temperatursensor

Für fast alle Temperaturmessungen bei diesem Projekt wird der digitale Temperatursensor DS18B20 verwendet.
Der physikalische Effekt, der bei diesem Sensor genuzt wird, ist die so genannte „band gap“ (Bandlücke). Darunter versteht man eine spezielle Eigenschaft von Halbleitern, die eine Größe ist, welche proportional zur absoluten Temperatur T ist. Mit einer integrierten „Brokaw-Zelle“ wird diese Größe in eine Spannung umgewandelt, welche von einem analog zu digital Konverter (ADC) aufgenommen wird. [5]
Der DS18B20 Temperatursensor wird über einen „1-Wire Bus“, welcher vom Hersteller Dallas entwickelt wurde, angesteuert. Über diesen kann der Sensor parametriert und vor allem der Temperaturwert ausgelesen werden. Somit können problemlos über 255 Sensoren an nur einem GPIO-Pin des Mikrocontrollers angeschlossen werden. [6]
Die direkte Anbindung der Sensoren an ein Bussystem mach diesen Sensortyp ideal für das Vorhaben, weil nur eine dreiadrige Busleitung (GND, 3.3V und Daten) bei allen Messpunkten durchgeschleift werden muss und nicht etwa für jeden Sensor ein eigenes Kabel verlegt werden muss. [7]

Spannung	3,0-5,5 V
Standby Strom	1 µA
Temperaturbereich	−55-125 °C
Auflösung	9-12 Bit
Kalibrierung	±0,2 °C
Schnittstelle	1-Wire Bus
Adressierung	64Bit ROM Adresse

Tabelle 3.2: Technische Daten DS18B20 [7]

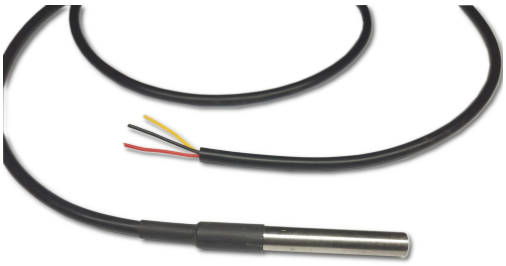


Abbildung 3.2: DS18B29 Temperatursensor im Gehäuse

PT100 Temperatursensor

Neben Wassertemperaturen soll auch die Abgastemperatur der Holzheizung gemessen werden. Da der digitale DS18B20 Sensor nur bis 125 °C eingesetzt werden kann, musste nach einer Alternative gesucht werden.
Für die Abgastemperaturmessung sind Platin Temperatursensoren gut geeignet, da diese meist auch für deutlich höhere Temperaturbereiche ausgelegt sind und auch in geeigneten Gehäusen für Abgasmessungen zu erwerben sind.
Bei einem PT100 Temperatursensor handelt es sich um einen Messwiderstand aus Platin, dessen Widerstandswert abhängig von der Temperatur ist. Bei steigender Temperatur steigt auch der Widerstand des Kaltleiters. Der PT100 besitzt bei einer Temperatur von 0 °C einen elektrischen Widerstand von 100 Ω. Der Temperaturkoeffizient α von Platin ist bekannt und somit kann bei jedem gemessenen Widerstandswert die aktuelle Temperatur berechnet werden.
Um diese Art von Sensor auszuwerten, muss der Widerstandswert in eine vom Mikrocontroller messbare Größe umgewandelt werden. Dies lässt sich am besten realisieren, indem man den Messwiderstand an eine, durch einen Operationsverstärker erzeugte, Konstantstromquelle anschließt und die abfallende Spannung misst.
Da das Entwickeln dieser Schaltung aber viel Zeit in Anspruch nehmen würde und die anschließende Umrechnung auf die exakte Temperatur relativ aufwändig wäre, wurde für diesen Zweck ein kostengünstiger, fertiger Messumwandler ausgewählt. Bei dem gewählten Modul handelt es sich um eine kleine Platine, mit einem MAX31865[2] IC, an die der PT100 Messwiderstand direkt angeschlossen werden kann. Dieser integrierte Schaltkreis (IC) wandelt den Wiederstandswert, exakt nach Kennline, in einen Temperaturwert um und stellt diesen digital über eine SPI Schnittstelle zur Verfügung.

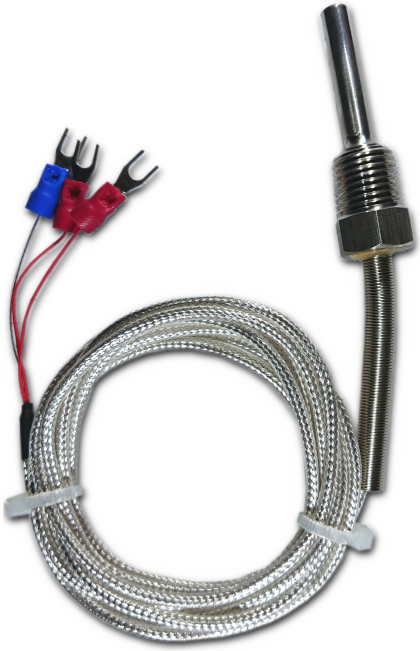


Abbildung 3.3: PT100 Temperatursensor im Schutzrohr

[2]https://datasheets.maximintegrated.com/en/ds/MAX31865.pdf

3.2.2 Drucksensoren

Es ist vorgesehen, den Wasserdruck des Heiz- und Solarkreises, sowie des Brauch- bzw. Kaltwassersystems zu überwachen. Dafür wird ein Bauteil benötigt, welches den aktuellen Wasserdruck in eine elektrische Größe umwandelt.
Um nach geeigneten Drucksensoren suchen zu können, wurden zuerst die Anforderungen an diese bestimmt. Anhand in den Systemen bereits vorhandener Manometer und Thermometer, konnten grob die vorhandenen Druckbereiche sowie die Temperaturbereiche an den gewünschten Messstellen ermittelt werden (siehe Tabelle 3.3).

Messort	**Druck**	**Temperaturbereich**	**gewählter Sensor**
Kaltwassersystem	4,5-5,5 bar	5-30 °C	0-150PSI 0,5-4,5V
Heizkreis	1-3,5 bar	25-95 °C	0-80PSI 0,5-4,5V
Solarkreis	3,5-4,5 bar	−10-120 °C	0-80PSI 0,5-4,5V

Tabelle 3.3: Anforderung an Drucksensoren

Die Firma Akozon bietet kostengünstige Drucksensoren gleicher Bauform für verschiedene Druckbereiche an. Der im Datenblatt angegebene Temperaturbereich dieser Sensoren beträgt −40-130 °C. Somit sind diese Sensoren auch für die vorhandenen Temperaturen von −10-120 °C geeignet.
Um eine hohe Messgenauigkeit zu erreichen sollte der Messbereich des Drucksensors nicht sehr weit über dem zu messenden Druckbereich liegen, damit möglichst viel Auflösung des Sensors und des Analogeingangs genutzt wird.
Zwei Druckaufnehmer mit 80PSI Nenndruck, welche also bis ca. 5,5 bar messen können, wurden für Heiz- und Solarkreis bestellt. Für das Kaltwassersystem wurde allerdings einer mit einem Maximaldruck von 150PSI (10,3 bar) gewählt, da in diesem Wassersystem auch Druckstöße, durch z. B. abrupt geschaltete Wasserverbraucher, auftreten können und der nächst kleinere Sensor dafür zu wenig Sicherheitstoleranz hätte.

Abbildung 3.4: Drucksensor

Diese Sensoren, mit gleicher Bauform besitzen ein ⅛-Zoll NPT Rohrgewinde, werden mit 5V versorgt und geben ein lineares Analogsignal von 0,5 V (0PSI) bis 4,5 V (80PSI) aus.
Es handelt sich dabei um Keramik-Dickschichtsensoren, welche als Messprinzip den piezoresistiven Effekt nutzen. Auf einem massiven Ring liegt eine Membran aus Aluminiumoxid-Keramik, auf der sich ein Dehnungsmessstreifen befindet.
Bei einer Druckänderung verformt sich diese elastische Keramikmembran und der daran angebrachte Dehnungsmessstreifen ändert seinen elektrischen Widerstand. Dieser Widerstandswert wird in Spannung umgewandelt, normiert, verstärkt und als Analogsignal ausgegeben. [8]

3.2.3 Stromsensoren

Für die Überwachung der Pumpen in den verschiedenen Heizkreisen soll deren Leistungsaufnahme gemessen werden. Diese Pumpen arbeiten mit 230 V Wechselspannung und haben eine Nennleistung von 5 bis 45 Watt. Es musste nach einer Methode gesucht werden, die Stromstärke in den Zuleitungen der Pumpen kontinuierlich und sicher zu messen.
Anstatt die Stromstärke, ähnlich wie in einem Multimeter, direkt mittels der abfallenden Spannung über einen niederohmigen Messwiderstand (Shunt) zu messen, wurde ein Stromwandler ausgewählt.
Ein Stromwandler basiert auf dem gleichen Funktionsprinzip wie ein Transformator. Der zu messende Leiter wird durch einen mit Sekundärwindungen umwickelten Eisenkern, geführt. Das durch den Stromfluss entstehende Magnetfeld um den Leiter induziert in dieser, über einen Messwiderstand kurzgeschlossenen Sekundärwicklung einen Strom. Die über diesen Widerstand abfallende Spannung wird verstärkt und kann später in die Eingangsstromstärke umgerechnet werden. [9]
Dieses Verfahren bietet den Vorteil, dass bei der Messung nicht in den 230V Stromkreis eingegriffen werden muss und die Messung somit sehr sicher (galvanisch getrennt) stattfindet. Außerdem müssen keine besonderen Umbaumaßnahmen am bestehenden System durchgeführt werden.

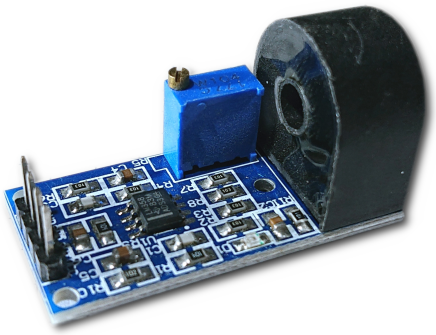


Abbildung 3.5: Stromsensor-Modul

In Abbildung 3.5 sieht man ein Modul, welches das beschriebene Funktionsprinzip durchführt. Dieses Bauteil wird mit 5 V versorgt, kann Stromstärken bis 5 A messen und gibt diese als Analogspannung aus.

3.3 Verwendeter Server

Um die in den nächsten Kapiteln beschriebenen Dienste nutzen zu können, müssen diese Anwendungen oder Dienste auf einem ständig zur Verfügung stehenden Computer (Server) installiert werden.
Es steht ein Server mit Gigabit Ethernet und einem Quad-Core Prozessor zur Verfügung. Dieser ist bereits in das Lokale Netzwerk eingebunden, mit dem auch die Sensorplatinen mittels WLAN verbunden sind.
Das Betriebssystem dieses Computers ist Debian Linux, auf das mittels SSH (Secure Shell) zugegriffen werden kann. Mit dieser Methode können, ebenfalls über das Netzwerk, Befehle auf dem Server ausgeführt werden und unter anderem Software installiert und konfiguriert werden.

CPU	Ryzen 3 1300X
Anzahl der Kerne	4
Arbeitsspeicher	4GB
Taktfrequenz	3,4Ghz
Netzwerk	Gigabit Ethernet
Betriebssystem	Linux, Debian 4.8
Speicher	500GB SATA SSD
IP-Adresse	10.0.0.102

Tabelle 3.4: Wichtigste Daten des Server

Wenn dieser Server nicht ohnehin schon für aufwändigere Aufgaben in Betrieb wäre, wür-

de für diese Projektarbeit ein deutlich leistungsschwächeres System, wie zum Beispiel ein Raspberry PI völlig ausreichen.

3.4 Serversoftware

3.4.1 MQTT Protokoll

Um die Sensordaten von den einzelnen Platinen zum zentralen Server zu übermitteln, wird ein Protokoll benötigt, welches die Kommunikation zwischen Platine und Server über das bestehende WLAN und LAN Netzwerk (TCP/IP) ermöglicht.

MQTT Logo [10]

Nach Einlesen und Abwägen verschiedener Protokolle, insbesondere OPC-UA und MQTT, stellte sich heraus, dass OPC-UA vergleichsweise komplex ist und sehr viele Funktionen bietet, von denen hier aber nur wenige benötigt werden. [11]
Das MQTT Protokoll, welches deutlich weniger komplex ist, bietet dafür genau die benötigten Funktionen und reicht in diesem Fall für das einfache Übermitteln von Sensor- und Steuerdaten aus. [10]

MQTT steht für Message Queuing Telemetry Transport, ist ein offenes Nachrichtenprotokoll und kann über die vorhandene TCP/IP Infrastruktur betrieben werden.
Es gibt im Netzwerk einen Server (Broker) und beliebig viele Clients. Die Clients sind mit dem MQTT Broker verbunden und können diesem Daten unter frei wählbaren Topics senden. Dabei stuft das Topic die Nachricht hierarchisch ein. Man kann es sich wie einen Dateipfad, der zu einem Wert führt, vorstellen. Der Broker empfängt die Daten und leitet diese an alle verbundenen Clients, welche dieses Topic aboniert haben, weiter.
Somit ist die Kommunikation zwischen Clients über den Server möglich, wobei die Software auf den Clients sehr einfach ist und diese nicht viel leisten müssen, da nur der Broker die gesamten Daten aller Clients verwaltet und nicht etwa jeder Client mit jedem eine Verbindung eingehen muss. [12][10]

Damit die Sensordatenübermittlung mittels MQTT Protokoll möglich ist, muss im Netzwerk also ein MQTT-Broker zur Verfügung stehen. Bei der Suche nach einer passenden Software, stellte sich heraus, dass der weit verbreitete open source „Eclipse Mosquitto MQTT Broker“ gut geeignet ist. Dieser unterstützt alle benötigten Funktionen und kann sehr einfach über eine „Debian-Repository“ (Sammlung von vorbereiteten Paketen für Linux) installiert werden.
Nach Installation auf dem vorhandenen Linux Server steht dieser Dienst unter der IP-Adresse des Servers über den TCP-Port 1883 im Netzwerk zur Verfügung. Mit Angabe dieser Adresse und Port (10.0.0.102:1883) können sich später alle Clients wie Mikrocontroller der Platinen, sowie Nod-Red mit dem MQTT Broker verbinden und Daten übermitteln.

3.4.2 Node-Red

„Node-RED ist ein von IBM entwickeltes grafisches Entwicklungswerkzeug. Die Software ermöglicht es, Anwendungsfälle im Bereich des Internets der Dinge mit einem einfachen Baukastenprinzip umzusetzen. Die einzelnen Funktionsbausteine werden durch Ziehen von Verbindungen verbunden. Eine große Auswahl an mitgelieferten Bausteinen deckt die meisten der gängigsten Dienste und Technologien ab“ [13].

Node-Red Logo [13]

Node-Red wird ebenfalls auf dem zentralen Server installiert und ist die gewählte Programmierumgebung, mit der fast alle Funktionen, die auf dem Server ablaufen, programmiert werden. Es dient unter anderem als Schnittstelle zwischen dem MQTT-Broker und der MYSQL-Datenbank. Mit Funktionsbausteinen, welche mit JavaScript frei programmiert werden, kann sowohl sehr effizient mit den Sensordaten gearbeitet, als auch beliebige komplexere Aufgaben durchgeführt werden.
Außerdem ist es sehr leicht möglich mithilfe einer großen Anzahl vorgefertigter Bausteine eine Web-Benutzeroberfläche (Dashboard) zum Darstellen der Sensordaten zu erstellen. Auch die vielen weiteren Möglichkeiten, wie z. B. das Versenden von E-Mail Benachrichtigungen oder das Nutzen einer Vielzahl an Erweiterungen, sprechen für die Verwendung von Node-Red in dieser Projektarbeit.
In Kapitel 4.4 wird genauer auf die konkrete Programmierung mit dieser Software eingegangen.

3.4.3 MySQL-Datenbank

In einer Datenbank werden große Mengen elektronischer Daten oder Informationen organisiert gespeichert. Die Daten werden bei den meisten Datenbanken typischerweise in Tabellen mit Zeilen und Spalten dargestellt. Dadurch ist eine effiziente Datenspeicherung und Abfrage möglich. Die meisten Datenbankverwaltungssysteme werden mit der structured query language (SQL) gesteuert. So können mit sog. SQL-Befehlen zum Beispiel Tabellen verwaltet, Werte eingetragen, sortiert oder ausgelesen werden. [15][16]

MySQL-Logo [14]

Die kontinuierlich erfassten Daten, der Sensoren im Heizraum, müssen gespeichert werden. Dadurch soll es möglich werden, wie gefordert, Sensordaten zu früheren Zeitpunkten anzuzeigen. Für diese Aufgabe wurde die Open-Source-Software MYSQL ausgewählt. MYSQL ist ein weit verbreiteter Dienst, der auf dem lokalen Server läuft und das Ausführen von SQL-Befehlen zum Abspeichern oder Verwalten der Sensordaten ermöglicht.
Vor allem weil MYSQL genau dafür gedacht ist, mit großen Datenmengen zu arbeiten, und bereits viele effiziente Sortieralgorithmen und Methoden zum Verarbeiten der Daten bietet, ist dieses Datenbanksystem gut für diese Anwendung geeignet.

4. Durchführung

4.1 Elektrische Komponenten

Ein sehr großer Bestandteil der Projektarbeit war die Entwicklung einer Mehrzweckplatine für Steueraufgaben und vor allem zur Sensordatenerfassung.

4.1.1 Anforderungen

Als Erstes wurden die gewünschten Funktionen und Anforderungen an die Platine durchdacht und gesammelt. Weil diese Platine universell einsetzbar, also später auch für andere Anwendungen verwendbar sein soll, wurde hier viel Zeit investiert und auch für das Projekt der Heizungsüberwachung unwichtig erscheinende Funktionen implementiert.
Nach Auswählen geeigneter Komponenten und Abwägen der Realisierbarkeit ist die Übersicht der Anforderungen und Funktionen in Tabelle 4.1 entstanden.

Tabelle 4.1: Anforderungen an die Platine

Allgemeine Eigenschaften:

Hohe Effizienz	Die Platine soll möglichst stromsparend sein (für z. B. Batteriebetrieb)
Modularer Aufbau	Je nach Anwendung müssen nur die Bauteile für die gerade benötigten Funktionen bestückt werden.
Spannungsstabilisierung	Die Platine soll auch mit einer unregelmäßigen oder sehr störhaften Spannungsversorgung problemlos arbeiten

Eingänge:

Analogeingänge	Analogeingänge mit frei wählbaren Spannungsteilern und Schutzbeschaltung
Digitaleingänge	Digitaleingänge mit Schutzbeschaltung
Messung der Eingangs-spannung	z. B. im Batteriebetrieb ist es somit möglich den Akkustand abzufragen

Ausgänge:

GPIO-Pins	Mehrzweckpins direkt vom Mikrocontroller stehen zur Verfügung für z. B. Analogausgänge oder Digitalausgänge
Relaisausgang	Onboard Relais z. B. zum direkten Schalten von 230V
Mosfetausgänge	Digitale Ausgänge mit hoher Schaltleistung
Treiber für Schritt- und Servomotoren	Möglichkeit Schritt-, oder Servomotoren direkt mit der Platine anzusteuern
LED, Piepser	Onboard LED und Piepser

Schnittstellen:

OWB (1-Wire Bus)	Bussystem für DS18B20 Temperatursensoren
I2C	Bussystem für diverse Peripheriegeräte
RS485	Industriebus-Schnittstelle für verschiedenste Protokolle (z. B. Modbus)
UART	Serielle Schnitsttelle, vor allem zum Programmieren des Mikrocontrollers
WLAN	Kabellose LAN Verbindung zum Austausch von Daten

4.1.2 Schaltplan

Nachdem die Anforderungen an die Platine festgelegt waren, konnte mit dem Erstellen des Schaltplans mit dem kostenlosen ECAD-Programmpaket KiCad[1] begonnen werden.
In diesem Kapitel werden die wichtigsten Bestandteile der entwickelten Schaltungen genauer erklärt. Der gesamte Schaltplan ist in Anhang A zu finden.

Spannungsstabilisierung

Um die Funktion auch bei einer sehr störhaften oder unregelmäßigen Versorgungsspannung zu gewährleisten wird die Eingangsspannung zuerst einmal stabilisiert. Dies erfolgt mit der Schaltung in Abbildung 4.1.
Über einen 33 Ω Widerstand und die Diode D1 wird ein 2300 µF Kondensator geladen. Falls die Versorgungsspannung einbricht oder kurzzeitig ausfällt, entlädt sich der Elko über die Diode D2 und überbrückt somit diesen Versorgungsengpass. Die Diode D1 dient dabei als Verpolungsschutz und gleichzeitig wird verhindert, dass Ladung vom Kondensator zurück zur Spannungsquelle fließen kann.

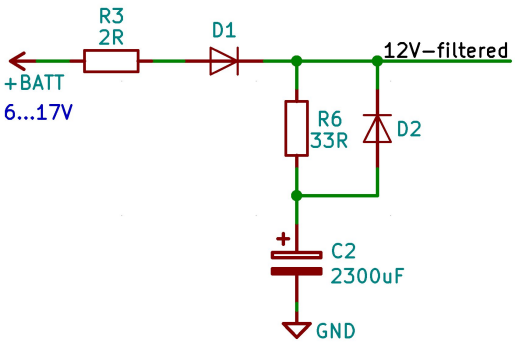


Abbildung 4.1: Spannungsstabilisierung

Spannungsregelung

Auf der Platine werden die Spannungen 3,3 V für den Mikrocontroller und 5 V hauptsächlich für Sensoren benötigt. Diese müssen aus der Eingangsspannung, welche im Bereich von 6 bis 17 V liegen darf, erzeugt werden.
Als effizienteste Methode bieten sich Sannungsregler wie der TPS562200 an. Diese arbeiten nach dem „step-down“ Prinzip und haben dadurch im Vergleich zu einem Linearregler, welcher die überschüssige Spannung in Wärme umwandelt, einen sehr guten Wirkungsgrad von bis zu 93%. [17]
Zwei dieser Spannungsregler wurden jeweils wie im Datenblatt beschrieben für 3,3 V bzw. 5 V beschalten und dem Schaltplan (Anhang A) hinzugefügt.

[1] `https://kicad-pcb.org/`

GPIO - Digital/Analog IO

Da die GPIO-Pins des ESP32 Mikrocontrollers jeweils für beliebige Anwendungen konfiguriert und genutzt werden können, wurde nach einem Weg gesucht, die Platine so zu gestalten, dass diese je nach Bedarf wahlweise als Ausgang oder Eingang genutzt werden können. Realisiert wurde dies mit der Schaltung in Abbildung 4.2. Diese ist für jeden Pin, der an eine Schraubklemme gelegt wird, vorgesehen.
Die geplante Schaltung wird je nach Anwendung verschieden bestückt:

- **Eingang:** Es wird die 3,6V Zenerdiode D_1 als Überspannungsschutz sowie R_1 und R_2 als beliebig wählbarer Spannungsteiler gelötet.
- **Ausgang:** Es muss nur R_1 überbrückt werden, der Rest bleibt ungelötet.

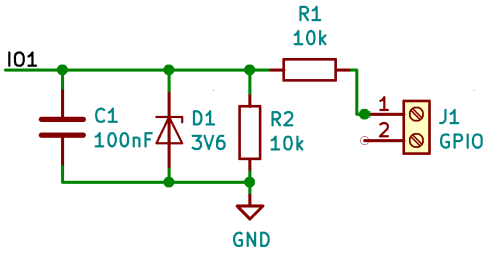


Abbildung 4.2: GPIO-Pins

Mosfet - Digitalausgänge

Die als Ausgang konfigurierten Pins des Mikrocontrollers dürfen maximal mit 12 mA belastet werden. Um also Geräte, wie zum Beispiel Schrittmotoren, Servomotoren oder LEDs schalten zu können, werden Digitalausgänge mit höherer Schaltleistung benötigt.
Die Digitalsignale vom Controller können mithilfe von Mosfets verstärkt werden, dafür wurde der SI232 N-CHANNEL Mosfet ausgewählt. Dieses kostengünstige SMD Bauteil kann eine Spannung von maximal 20 V und eine Stromstärke von bis zu 2,5 A schalten.
Bei der Beschaltung des Mosfets war zu beachten, dass ein Widerstand in Reihe mit dem Controller geschaltet werden muss (R1), damit der Gate-Ladestrom des SI2302 nicht den Maximalstrom des GPIO-Pins überschreitet. Um zufällige Schaltzustände, durch ein undefiniertes Signal (floating) beim Starten des Mikrocontrollers, zu verhindern, wurde noch ein Pulldown-Widerstand (R2) eingeplant. Außerdem wurde am Ausgang eine Freilaufdiode (D1) zum Schutz gegen zu hohe Spannung vorgesehen, falls induktive Lasten wie z. B. Schrittmotoren oder Relais geschaltet werden.

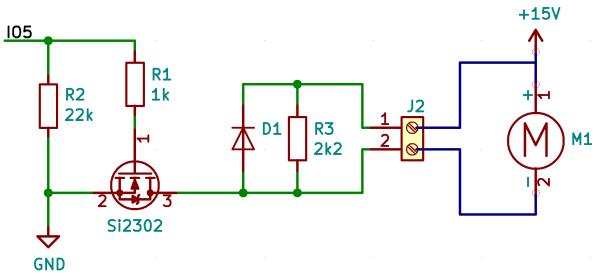


Abbildung 4.3: Mosfet Ausgang

Dadurch, dass diese N-Channel Mosfets in einer „Open-Drain-Schaltung“ aufgebaut wurden und somit den Ausgang bei Aktivierung auf GND schalten, können diese „starken“ Digitalausgänge für Geräte mit einer beliebigen Spannung im Bereich 0 bis 20V benutzt werden. Für solche Anwendungen wurden auf der Platine vier dieser Mosfetschaltungen vorgesehen.

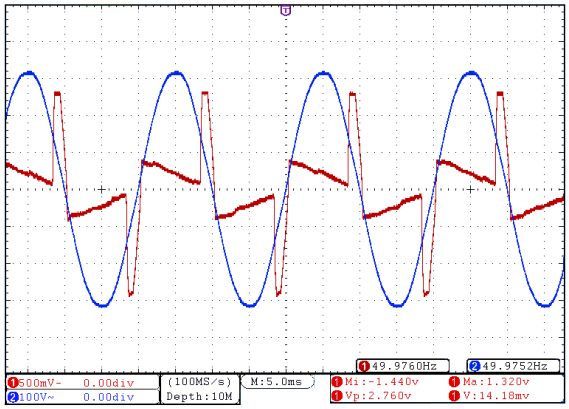
1-Wire-Bus

Für die DS18b20 Temperatursensoren und deren 1-Wire-Bus wird separat eine dreifach Schraubklemme vorgesehen. Mit dieser werden 3,3 V, GND und ein GPIO-Pin vom Controller für die Datenleitung des Busses verbunden. Da der Mikrocontroller den gleichen Pin

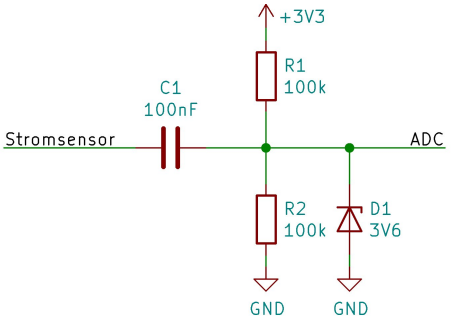
immer abwechselnd als Eingang und Ausgang nutzen kann, ist es problemlos möglich mit nur einem normalen GPIO-Pin sowohl Daten zu senden als auch Daten zu empfangen. Es wird keine zusätzliche externe Hardware benötigt.
Da diese Temperatursensoren die Datenleitung auch mit der „Open-Drain-Methode“ beschalten, muss noch ein Pullup Widerstand eingeplant werden, welcher die Datenleitung im Ruhezustand auf das Spannungslevel von 3,3 V bringt.

Spannungswandler

Nach Erhalt der, in Kapitel 3.2.3 beschriebenen, Stromsensoren, wurde einer testweise bei der Heizungsanlage in Betrieb genommen. Um das ausgegebene Signal bei der Strommessung einer Heizungspumpe zu charakterisieren, wurde ein digitales Oszilloskop angeschlossen.
In Abbildung 4.4a sieht man einen kurzen Verlauf der gemessenen Werte. Mit dem ersten Kanal (rot) wurde das Signal am Ausgang des Stromsensors aufgenommen, durch den die Versorgungsleitung einer gerade aktiven Pumpe geführt wurde. Die blaue, sinusförmige Kurve, stellt die 230V Netzspannung dar, welche zur Referenz mit dem zweiten Kanal gemessen wurde.
Man kann erkennen, dass es sich bei dem Sensorsignal ebenfalls um eine Wechselspannung handelt. Es wurden bei voller Pumpenleistung Werte von $U_{min} = -1{,}4\,\text{V}$ bis $U_{max} = 1{,}3\,\text{V}$ gemessen. Daraus ergibt sich eine Spitze-Spitze-Spannung $U_{SS} = 2{,}7\,\text{V}$.
Da an den Analog-Digital-Wandlern des ausgewählten Mikrocontrollers nur Spannung im Bereich von 0-3,3 V angelegt werden darf, muss eine Schaltung entworfen werden, die das Signal dementsprechend anpasst.



(a) Signal Stromsensor



(b) Schaltung Spannungsverschiebung

Abbildung 4.4: Signalverarbeitung Stromsensoren

Die in Abbildung 4.4b zu sehende Schaltung, soll das Signal um die halbe Eingangsspannung nach oben verschieben. Bei dieser so genannten Bias-Schaltung wird das Eingangssignal, welches durch den Kondensator C1 entkoppelt wird, mit einer durch den Spannungsteiler aus R1 und R2 bereitgestellten Gleichspannung überlagert. Die 3,3 V Zenerdiode D1 soll den Mikrocontroller zusätzlich vor zu hoher Eingangsspannung schützen.
Auf die Umsetzung dieser Schaltung wird in Kapitel 4.1.4 genauer eingegangen.

4.1.3 Mehrzweckplatine

Layout

Nachdem der Schaltplan (siehe Anhang A) fertig erstellt und geprüft war, konnte weiterhin im Programm KiCad mit dem Routen der Platine begonnen werden. Routen bedeutet hier, dass mit Vorlage des erstellen Schaltplans die Leiterbahnen der Platine gezeichnet werden.
Den beim Erstellen des Schaltplans ausgewählten Bauteilen wurden bereits sog. Footprints (Abmessungen der Gehäuse und Pins) zugewiesen. Nun wird versucht die Bauteile möglichst effizient mit möglichst wenig kreuzenden, Verbindungen zu platzieren (siehe Abbildung 4.5).
Da eine doppelseitige Platine geplant war, konnten mittels Vias, mit welchen Leiterbahnen auf den verschiedenen Seiten durchkontaktiert werden können, auch auf der Rückseite der Platine Leiterbahnen verlegt oder Bauteile platziert werden.

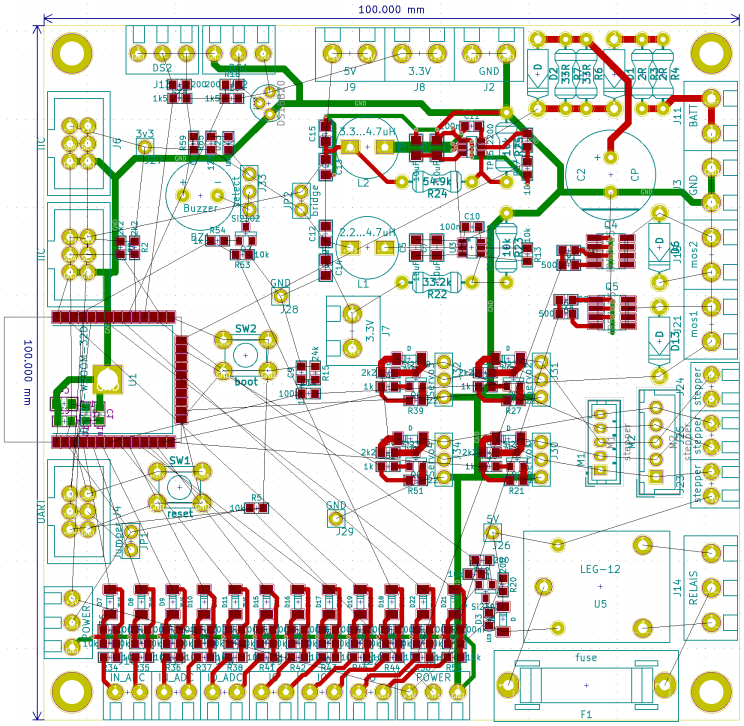


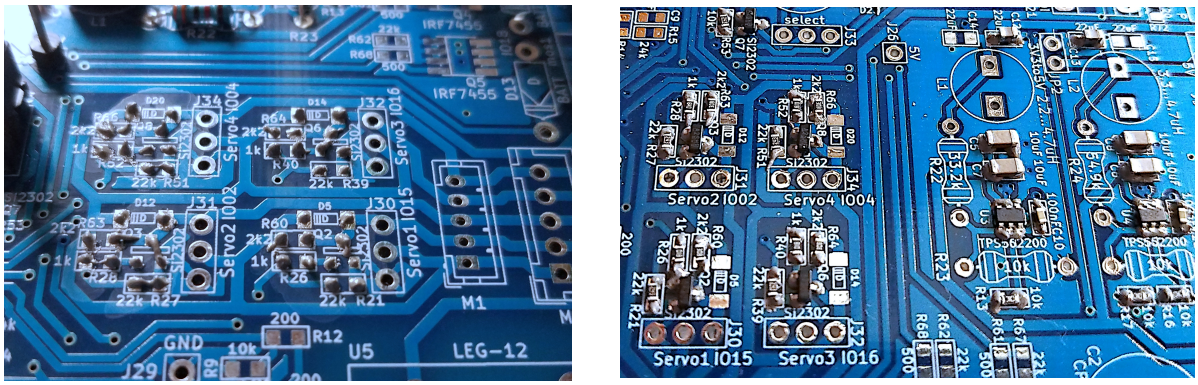
Abbildung 4.5: Routen: Festlegen des Layouts

Das fertiggestellte Platinenlayout mit Beschriftung ist im Anhang B zu finden.
Im nächsten Schritt wurde die fertig erstellte Platine im Programm KiCAD als standardisiertes GERBER-Format exportiert, welches sämtliche Informationen, die zum Herstellen einer Platine benötigt werden, enthält. Diese Datei wurde anschließend mit dem Auftrag zur Platinenfertigung zur Firma Elecrow[2] gesendet.

Bestückung

Nach Erhalt der Leiterplatten, wurden diese kurz auf eventuelle Fehler untersucht. Als keine offensichtlichen Mängel festgestellt und alle benötigten Bauteile vorbereitet waren, konnte mit dem Bestücken der Platine begonnen werden.
Nachdem einige größere Bauteile wie Schraubklemmen, Stiftleisten und andere Bauteile der „Through Hole Technology“ mit dem Lötkolben gelötet waren, konnte mithilfe einer Spritze Lötpaste auf die Pads aller benötigten SMD Bauteile aufgebracht werden (siehe Abbildung 4.6a). Im nächsten Schritt wurden die millimetergroßen SMD Bauteile mit einer Pinzette platziert. In Abbildung 4.6b sieht man links die in der Paste platzierten Mosfets mit Beschaltung sowie rechts die Spannungsregler.

[2] `https://www.elecrow.com/pcb-manufacturing.html`



(a) Lötpaste aufgebracht (b) Bauteile platziert

Abbildung 4.6: Bestückung der Platine

Die verwendete Lötpaste besteht aus Zinn und einem großen Anteil Bismuth (Sn42/Bi58). Durch den Bismuth Anteil ist der Schmelzpunkt sehr niedrig (ca. 130 °C) und kann problemlos mit einem Heissluftföhn erreicht werden. Nachdem alle Bauteile mit den jeweiligen Pads verbunden waren und bei einer Sichtprüfung keine fehlerhaften Verbindungen festgestellt wurden, konnte der erste Test durchgeführt werden, welcher erfolgreich war.

4.1.4 Spannungswandler Platine

Neben der Mehrzweck-Platine war auch die in Kapitel 4.1.2 entwickelte Schaltung zum Anpassen der Stromsensorsignale umzusetzen. Da diese sehr überschaubar ist und aus wenigen Bauteilen besteht, wurde diese Schaltung in zweifacher Ausführung auf einer Streifenrasterplatine aufgebaut und gelötet.
In Abbildung 4.7 wird die angefertigte Platine mithilfe einer 3,3 V Spannungsversorgung, eines Funktionsgenerators und des Oszilloskops getestet. Man sieht, dass die am Eingang angelegte Sinusspannung (rot) wie gewünscht um 1,65 V nach oben verschoben wird (gelb). Die erkennbare Phasenverschiebung zwischen Eingangs und Ausgangssignal entsteht durch den in Reihe geschalteten Kondensator (siehe Abbildung 4.4b).

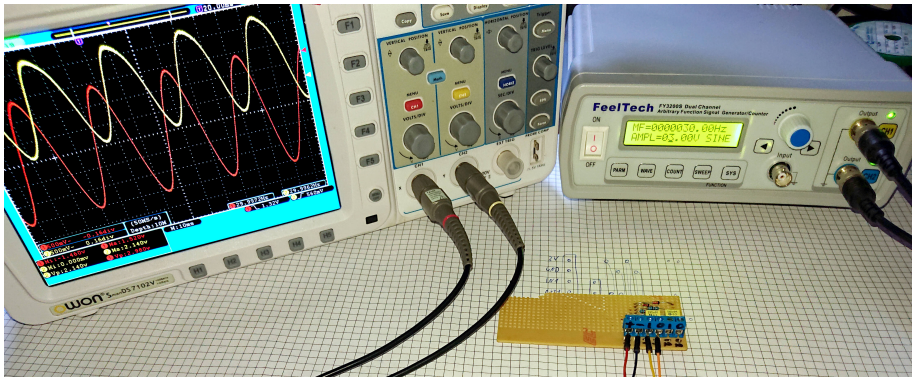


Abbildung 4.7: Test des Spannungswandlers

Somit kann diese Streifenrasterplatine im nächsten Kapitel zwischen Stromsensoren und der

Mehrzweckplatine geschaltet und die Analogspannung in Kapitel 4.3.7 ausgelesen werden.

4.1.5 Verdrahtung

Die vier Stromsensoren für die Pumpen sowie ein 12V Netzteil wurden auf einem Kunststoffstreifen befestigt (Abbildung 4.8), damit diese ordentlich im Verdrahtungskasten des Ofens untergebracht werden können.

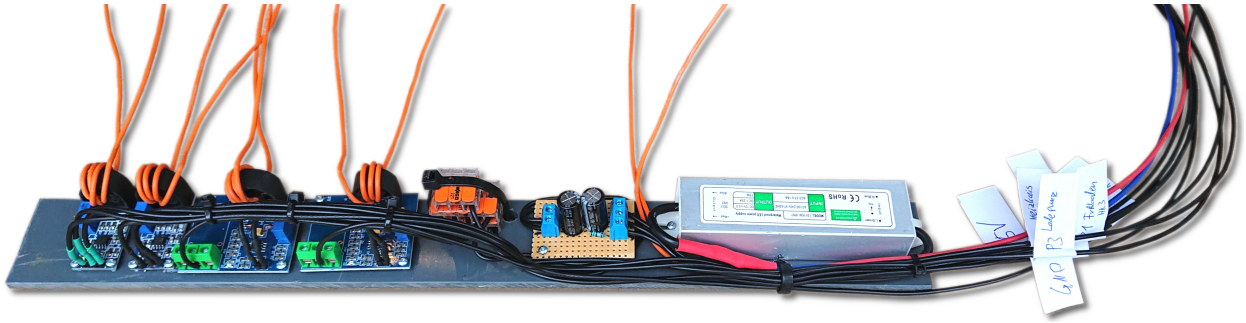


Abbildung 4.8: Stromsensoren und Netzteil

Jede der orangen Verdrahtungsleitungen, welche je mit einer 230V Pumpe in Reihe geschaltet sind, ist 4 mal um den Stromwandler des jeweiligen Stromsensors gewickelt, um die Messgenauigkeit zu erhöhen.
Sämtliche anderen Leitungen (rechts im Bild) werden auf eine 14-adrige, flexible Steuerleitung gelegt, welche zum Gehäuse der gelöteten Platine verlegt ist.
Auch die Busleitung der Temperatursensoren sowie ein Kabel der Drucksensoren werden in den bereits vorhandenen Kabelschächten der Heizungsanlage zum Montageort der Platine geführt.
Die beim Platinengehäuse, welches im nächsten Kapitel 4.2.1 näher beschrieben wird, ankommenden Leitungen wurden zuerst abgelängt und mit Kabelbindern an der vorgesehenen Zugentlastung fixiert. Anhand eines bereits vor der Bestückung erstellten Anschlussplans (siehe Anhang C) konnten die einzelnen Adern der Leitungen an die jeweiligen Schraubklemmen der Platine angeschlossen werden.
Auch die in Kapitel 4.1.4 entwickelte Streifenrasterplatine zum Anpassen der Analogsignale wurde mit 3,3V versorgt und zwischen Stromsensoren und Analogeingänge der Platine geschaltet.

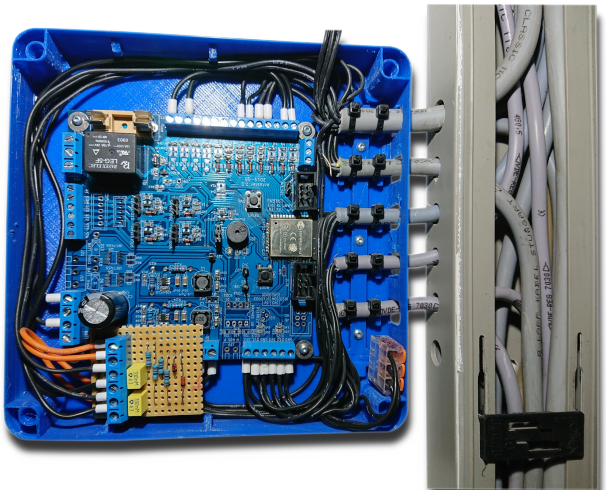


Abbildung 4.9: Platinen im Gehäuse verdrahtet

4.2 Mechanische Komponenten

4.2.1 Gehäuse

Es wird eine Baugruppe benötigt um die Platine zu schützen, Kabel zu sammeln und um eventuelle Leuchtdioden oder Taster ordentlich unterzubringen.

Planung: Als Anhaltspunkt dienten die Abmessungen der bereits vorhandenen Platine. Daraus konnte das Grundgerüst des zweiteiligen Gehäuses schon einmal konstruiert werden. Deckel und Boden sollen durch vier M5 x 30 Schrauben mit Innensechskant zusammengehalten werden. Dafür mussten Ausschnitte für Sechskantmuttern im Boden und Zylindersenkungen für die Schraubenköpfe im Deckel hinzugefügt werden. Außerdem wurden noch Bohrungen für Leuchtdioden und Ausschnitte für die Leitungen ergänzt.
In Abbildung 4.10 sieht man, dass neben kleinen Befestigungspunkten für die Platine und Laschen an den Wänden zum Fixieren einzelner Adern noch ein separates Teil konstruiert wurde, an das die ankommenden Leitungen mit Kabelbindern befestigt werden können (rot im Bild).

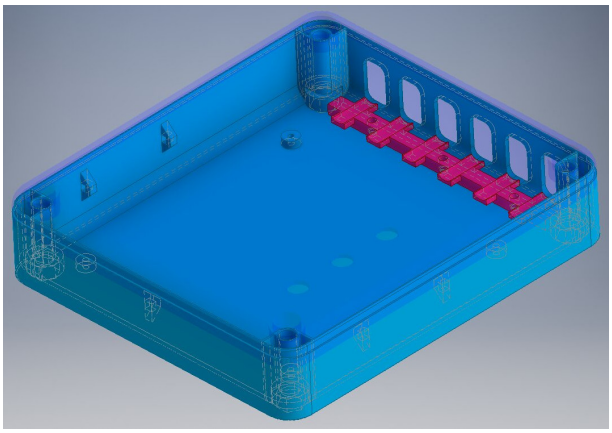


Abbildung 4.10: Konstruierte Gehäusebaugruppe

Fertigung: Da beim Erstellen der Baugruppe bereits auf 3D-Druck gerechtes Konstruieren geachtet wurde, war geplant dieses Gehäuse mit einem 3D-Drucker anzufertigen. Die in der CAD Software exportierten STL-Dateien der einzelnen Komponenten konnten in eine sog. Slicer-Software geladen werden. Dort legt man alle wichtigen Paramater zum Drucken, wie zum Beispiel Schichtdicke, Temperatur oder Geschwindigkeit, fest und lässt die Fahrbefehle für den Drucker (G-Code) generieren.
Weil das Gehäuse keinen großen mechanischen Beanspruchungen und vor allem keinen höheren Temperaturen ausgesetzt ist, wurde als Material das gängigste Druckerfilament PLA ausgewählt, welches sehr gut zum Drucken geeignet ist, aber bereits ab ca. 60 °C weich wird.

Montage: Nachdem der Druck aller Einzelteile nach insgesamt ca. 8 Stunden abgeschlossen war, konnte das Gehäuse direkt am gewünschten Ort befestigt werden.
Da das Gehäuse mit Platine sehr leicht ist und keiner größeren Belastung ausgesetzt ist, wurde es nicht mit Dübeln in der Betonwand befestigt. Stattdessen wurde es mit doppelseitigem Montageklebeband, welches auf der gesamten Bodenfläche aufgebracht wurde, an die Wand geklebt. Anschließend konnte die bereits bestückte Platine und die Zugentlastung mit 2,2 x 6,5 mm Torx Blechschrauben montiert werden (siehe Kapitel 4.1.5 Abbildung 4.9).

4.2.2 Sensorhalterungen

Temperatursensorhalter

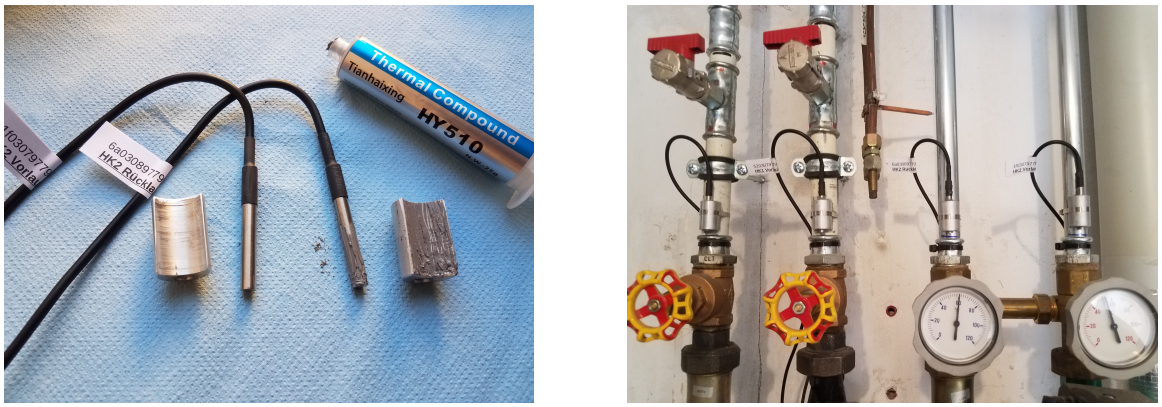
Planung: Die DS18B29 Temperatursensoren werden bereits in einem zylindrischen, wasserdichten Gehäuse geliefert. Doch um diese an den jeweiligen Messstellen zu befestigen, werden individuelle Halter benötigt.
Das Übertragen der Temperatur der Rohrleitung bzw. des darin fließenden Wassers zum Sensor soll mit möglichst wenig Verzögerung und ohne Verfälschung geschehen. Deshalb wurde entschieden verschiedene, an die jeweiligen Rohrdurchmesser angepasste, Sensorhalter zu konstruieren. Dadurch liegt der Halter großflächig am Rohr an und die Temperatur wird optimal zum Sensor übertragen.
Um eine gute Wärmeübertragung vom Rohr zum Sensor zu gewährleisten, wurde als Rohmaterial die Aluminiumlegierung AlCuMg mit einer spezifischen Wärmeleitfähigkeit von $\lambda = 130 - 170\,\frac{\mathrm{W}}{\mathrm{m\,K}}$ ausgewählt. Stahl besitzt zum Vergleich nur eine Wärmeleitfähigkeit von $\lambda = 15 - 50\,\frac{\mathrm{W}}{\mathrm{m\,K}}$. [18]
Nicht nur die gute Wärmeleitfähigkeit, sondern auch der, im Vergleich zu ähnlichen Materialien, geringe Preis und die leichte Zerspanbarkeit sprechen dafür, den Werkstoff Aluminium für die Sensorhalter zu verwenden.
Nach Ausmessen der Rohrdurchmesser an den relevanten Stellen wurde die in Anhang G.3 angehängte Technische Zeichnung mit allen benötigten Haltern erstellt.

Fertigung: Zuerst wurden von einem Aluminiumrundstab mit 20mm Durchmesser die benötigten Rohmaterialstücke heruntergesägt.
Aufgrund der geplanten individuellen Anpassung an die verschiedenen Rohrdurchmesser mussten die Teile mit verschiedenen Radien versehen werden. Dazu wurde erst eine eigene Spannvorrichtung hergestellt, mit der die Aluminiumteile mithilfe der erstellten 6,1 mm Bohrung für den Sensor sicher auf die Fräsmaschine gespannt werden können.
Nachdem dies vorbereitet war, konnten die Teile anhand der vorher erstellen Übersicht mit den geeigneten Fräsern bis jeweils zur Teilmitte angefräst werden. Einige der gefertigten Sensorhalter für verschiedene Rohrduchmesser werden in Abbildung 4.11 gezeigt.

Abbildung 4.11: Fertige Sensorhalter

Montage: Vor der Montage wurden noch mittels einer der gefertigten Platinen die 48 Bit langen, eindeutigen Adressen der Sensoren ausgelesen. Jedem Temperatursensor wurde ein zum Messort passender Sensorname (vgl. Kapitel 2.2) zugeordnet und dieser zusammen mit der Adresse als Beschriftung am Kabel des Sensors angebracht.
Nun konnten die DS18B20 Temperatursensoren in den dafür vorgesehenen 6,1 mm Bohrungen der Aluminiumhalter mithilfe von Wärmeleitkleber befestigt werden. Um eine gute Wärmeleitfähigkeit zwischen Heizungsrohren und Temperatursensoren zu gewährleisten, wurde auf die Auflagefläche der Aluminiumhalter etwas Wärmeleitpaste aufgetragen (sie-

he Abbildung 4.12a). Anschließend wurden diese mit Metallkabelbindern an den relevanten Messstellen angebracht. In Abbildung 4.12b sieht man vier montierte Temperatursensoren an jeweils Vor- und Rücklauf von Heizkreis-1 (links) und Heizkreis-2 (rechts).



(a) Sensoren und Halter mit Wärmeleitpaste

(b) Temperatursensoren an Heizungsrohren befestigt

Abbildung 4.12: Temperatursensorhalter

Drucksensorhalter

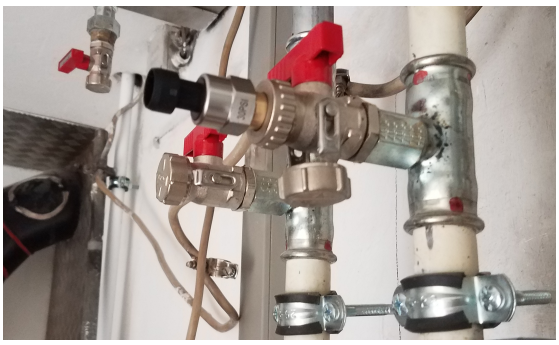
Planung: Um den Druck des Brauchwassers sowie der Solar- und Heizkreise zu messen, musste eine geeignete Methode gesucht werden, die in Kapitel 3.2.2 ausgewählten Drucksensoren in die jeweiligen Wassersysteme zu integrieren. Dafür haben sich die an den relevanten Stellen bereits vorhandenen Wartungskugelhähne angeboten (siehe Abbildung 4.14b). Somit war es möglich die Sensoren anzubringen, ohne das vorhandene Rohrsystem umzubauen.
Es mussten geeignete Adapter angefertigt werden, um von den konisch bzw. flach dichtenden Kugelhähnen auf das ⅛-Zoll NPT Rohrgewinde der Sensoren zu adaptieren. Nach Abmessen und Skizzieren der für das Abdichten relevanten Maße konnte mit dem Konstruieren der Adapter begonnen werden.
Ziel war es, ein Drehteil zu konstruieren, das dichtend mithilfe einer Überwurfmutter an den Wartungshähnen befestigt wird und in welches der Drucksensor eingeschraubt werden kann. Beim Ladekreis der Solaranlage war die Position des Wartungshahns etwas ungünstig. Deshalb wurde entschieden dafür einen speziellen, abgewinkelten Adapter zu entwerfen.
Als Werkstoff wurde die Kupfer-Zink Legierung Messing gewählt, wegen der Korosionsbeständigkeit und weil die bereits vorhandenen Armaturen ebenfalls aus Messing waren.

Fertigung: Anhand der erstellten technischen Zeichnungen (siehe Anhang G.1 und G.2) konnten die Drucksensorhalter aus Messing größtenteils an einer Drehbank gefertigt werden. Der abgewinkelte Adapter musste jedoch zusätzlich, wie in Abbildung 4.13 zu sehen, an einer Fräsmaschine etwas angefräst werden und es musste eine senkrecht zur Drehachse stehende Gewindebohrung gesetzt werden.

Abbildung 4.13: Abgewinkelter Adapter

Montage: Nach dem Reinigen der Teile konnten Dichtscheibe, bzw. Dichtungsringe in den dafür vorgesehenen Einstichen angebracht werden. Die Gewinde der Drucksensoren wurden zusätzlich zur selbstdichtenden Eigenschaft des NPT Gewindes noch mit Teflon-Band versehen und mit den Messingteilen verschraubt.
Die vormontierten Adapter werden in Abbildung 4.14a gezeigt: Links ein Gerader für Kaltwasser sowie für die Heizkreise und rechts der abgewinkelte für den Solarkreis. Mithilfe der integrierten Überwurfmuttern konnten die Sensorbaugruppen nun an den Messstellen befestigt und die Hähne geöffnet werden (siehe Abbildung 4.14b).

(a) Vormontierte Drucksensoren



(b) Sensor an Wartungshahn angeschlossen

Abbildung 4.14: Drucksensor Adapter

4.3 Programmierung Platine

4.3.1 Toolchain

Bevor mit der Programmierung des Mikrocontrollers begonnen werden konnte, musste eine so genannte Toolchain (Werkzeugkette) aufgesetzt werden. Darunter versteht man eine Sammlung an Programmen, die nacheinander systematisch benutzt werden um in der Softwareentwicklung ein Projekt zu implementieren. Solche Komponenten sind unter anderem Texteditor, Compiler, Linker, Bibliotheken, Debugger oder auch eine Methode zum Übertragen des Programms zum Controller. [20]
Bibliotheken (ESP-IDF), ein großer Teil der Toolchain, der Compiler (Build) sowie eine Methode zum Übertragen (Upload) und Verbinden mit dem Mikrocontroller wurde nach einer Ausführlichen Anleitung[3] von Espressif, dem Hersteller des ESP32, aufgesetzt.
Sämtliche Applikationen und Funktionen, die auf dem Mikrocontroller laufen wurden mit der Programmiersprache C, im ausgewählten Texteditor Vim umgesetzt.

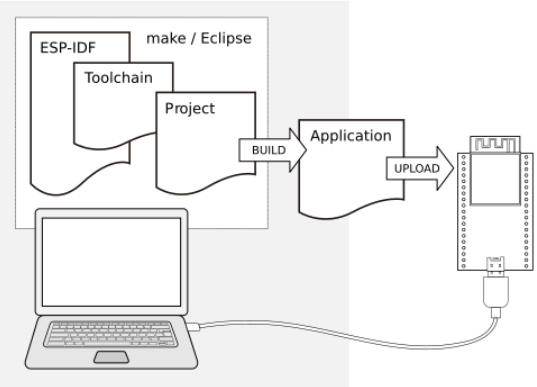


Abbildung 4.15: Benötigte Komponenten [19]

4.3.2 Dateistruktur

Um die zu entwickelnde Software für die Platinen möglichst übersichtlich zu halten, wurde das Projekt in mehrere Dateien aufgeteilt. Durch Abbildung 4.16 wird deutlich, dass in mehreren Dateien Teilfunktionen programmiert werden und diese in der *main.c* Datei mit den jeweiligen Parametern aufgerufen werden.
So sind beispielsweise in der Datei *temperatursensoren.c* Funktionen für das Initialisieren und Auslesen der angeschlossenen Temperatursensoren implementiert, welche dann im Programmcode der *main.c* Datei regelmäßig aufgerufen bzw. ausgeführt werden.

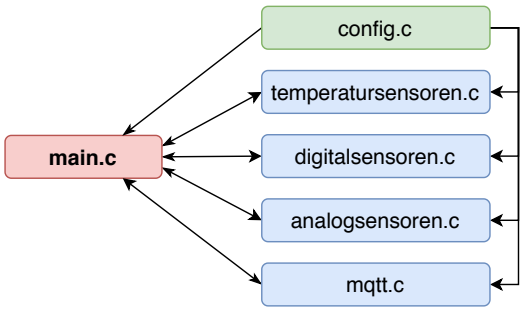


Abbildung 4.16: Dateistruktur

4.3.3 Programmaufbau

Das echtzeitfähige Open Source Betriebssystem FreeRTOS des Mikrocontrollers ermöglicht es, mehrere so genannte Tasks zu starten. Bei diesen Tasks handelt es sich um voneinander unabhängige und parallel ablaufende Prozesse in welchen Programmcode ausgeführt wird. [21]

[3] `https://docs.espressif.com/projects/esp-idf/en/latest/get-started/index.html`

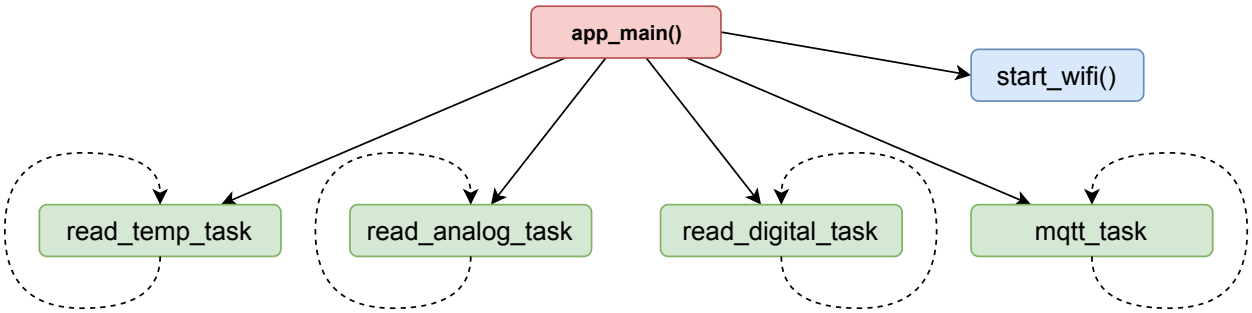


Abbildung 4.17: Übersicht der erstellten Tasks

Wie in Abbildung 4.17 veranschaulicht, werden nach Starten des Controllers durch die `app_main` Funktion vier dieser Tasks für verschiedene Aufgaben gestartet. In jedem dieser Tasks befindet sich eine Endlosschleife, in der die jeweilgen Funktionen zyklisch abgearbeitet werden.
Das Betriebssystem steuert dabei den Zugriff der einzelnen Prozesse auf die CPU. Der so genannte RTOS Scheduler führt also immer abwechselnd jeden Task einzeln für eine gewisse Zeit aus und pausiert diesen dann wieder. Wenn zum Beispiel in einem Task zum Auslesen der Sensoren eine Zeit abgewartet wird, hat die CPU Zeit für andere Aufgaben oder Tasks. Dadurch werden diese Tasks parallel, quasi gleichzeitig, abgearbeitet. [21]

In den folgenden Kapiteln werden die Abläufe der wichtigsten Funktionen der programmierten Software erklärt. Die gesamte, konkrete, Umsetzung in der Programmiersprache C kann in den angehängten *.c*- und *.h*-Dateien im Ordner *Firmware-Platine* gefunden werden.

4.3.4 Konfiguration und Sensordeklaration

Das System soll möglichst erweiterbar gestaltet werden, deswegen wurde nach einer Methode gesucht, alle angeschlossenen Sensoren möglichst einfach zu konfigurieren.
Dafür wurde die separate Datei *config.c* vorgesehen, in der alle angeschlossenen Sensoren aufgelistet und mit den benötigten Informationen versehen werden.
Neben Sensordeklarationen werden in dieser Datei auch beliebige Funktionen für gewünschte Steueraufgaben, sowie platinenspezifische Informationen (wie WLAN und MQTT-Broker Name und Passwort) konfiguriert.
Um neue Sensoren hinzuzufügen oder eine komplett andere Platine zu programmieren muss also nur diese Datei verändert werden, der Rest des Programms bleibt dabei auf allen Geräten einheitlich und unverändert.

Zuerst wurde in der Datei *config.h* allgemein, für jede Art von Konfigurationseintrag (Temperatur-, Analog-, Digitalsensor und Steueraufgaben), eine Typdeklaration einers strukturierten Datentyps angelegt. In der Datei *config.c* werden anschließend Arrays mit Instanzen dieser Datentypen erstellt, welche dann mit den konkreten Informationen der jeweiligen Sensoren oder Aufgaben versehen werden.

Ein Konfigurationseintrag für eine Steueraufgabe zum Schalten eines Relais sieht beispielsweise wie folgt aus:

Auflistung 4.1: Konfiguration MQTT Steueraufgabe

```
1  void pumpeEinschalten(bool data){   //Definition einer Funktion
2      if (data == true){
3          gpio_set_level(12, 1);  //Relais einschalten
4      }else{
5          gpio_set_level(12, 0);  //Relais ausschalten
6      }
7  }
8
9  int MQTT_SubscribedTopics_anzahl = 1;
10 struct MQTT_SubscribedTopic MQTT_SubscribedTopics[] = {
11     {
12     .name="Zirkulationspumpe",                  //Name (nur beim Monitoring zu Sehen)
13     .mqtt_topic="Steuerdaten/Heizraum/Brauchwasser/Pumpe",  //MQTT Topic, welches aboniert wird
14     .timestamp=0,                               //Timestamp der letzten Messung bzw. Aktion
15     .value=0,                                   //Aktueller Wert bzw. Zustand
16     .handleFunction=pumpeEinschalten            //Pointer zur Funktion, die ausgefuehrt wird
17     },
18 };
```

Im Vergleich zu einer Sensordeklaration wird hier für eine Steueraufgabe zusätzlich eine Funktion deklariert, die beim Empfangen von Daten über das angegebene MQTT-Topic ausgeführt wird.

Sämtliche anderen Einträge werden nach dem gleichen Schema in dieser Datei eingetragen. In nachfolgender Tabelle wird gezeigt, welche Angaben bei welchem Konfigurationstyp gemacht werden müssen:

Temperatursensoren	**Digitaleingänge**	**Analogsensoren**	**Steueraufgaben**	**MQTT Broker**	**WLAN**
Name	Name	Name	Name	Adresse	SSID
MQTT Topic	MQTT Topic	MQTT Topic	MQTT Topic	Port	Passwort
Bus Adresse	GPIO Pin	GPIO Pin	auszuführende Funktion		
	Anzahl Messwerte	Spannungsbereich			
	Invertierung	Auflösung (9-12Bit)			
	Umrechnungsfaktor	Modus (AC/DC)			
		Umrechnungs-parameter			

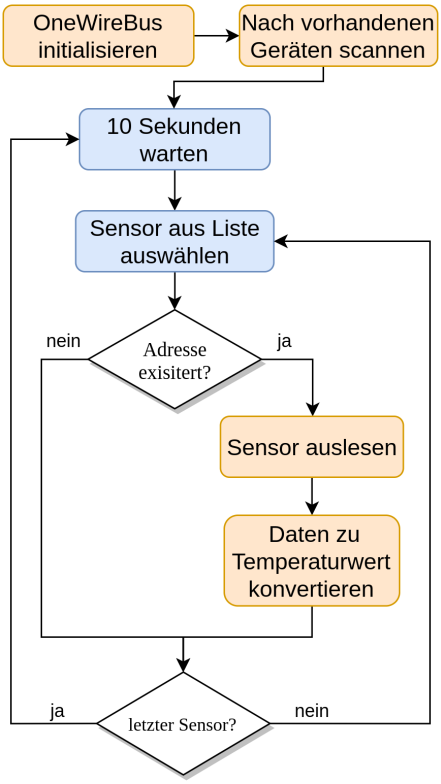
4.3.5 Temperatursensoren

Die digitalen DS18b20 Temperatursensoren müssen regelmäßig ausgelesen werden. Dies erfolgt nach dem in Abbildung 4.18a gezeigten Ablauf.
So sieht man, dass zu Begin der Bus initialisiert wird und alle daran angeschlossenen Geräte einmal abgefragt werden, um eine Liste der verfügbaren Adressen zu erhalten. Anschließend werden alle in der Konfigurationsdatei deklarierten und tatsächlich vorhandenen Sensoren gezielt mit ihrer Adresse abgefragt, die ankommenden Bits in einen Temperaturwert umgerechnet und in der Struktur des jeweiligen Sensors im Wert *.value* gespeichert.

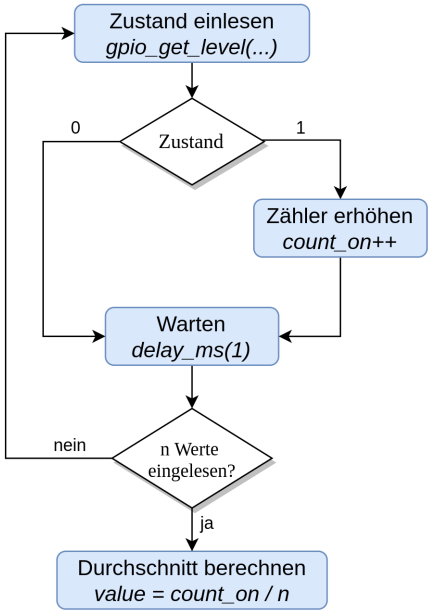
Die orange markierten Funktionen zum Kommunizieren auf dem One Wire Bus und Ansprechen der Temperatursensoren werden aus einer importierten Bibliothek von David Antliff[4] genutzt.

4.3.6 Digitaleingänge

Es sind mehrere Digitaleingänge vorhanden, an welche Komponenten wie z. B. Taster oder das PWM-Signal der Solarpumpe angeschlossen sind. Um diese Signale auszuwerten wurde eine Funktion programmiert, welche für jeden in der *config.c* Datei konfigurierten Digitaleingang regelmäßig durchlaufen wird. Der Ablauf dieser Funktion ist in Abbildung 4.18b vereinfacht dargestellt.
Je nach individueller Konfiguration des aktuellen Eingangs werden über einen Zeitraum `n` Zustände des Digitaleingangs eingelesen und daraus der Mittelwert berechnet.
Dieser erhaltene Wert wird nun, je nach Konfiguration (vgl. Kapitel 4.3.4), skaliert oder invertiert. Der so ermittelte Wert und die aktuelle Zeit, in Form eines Timestamps, werden anschließend in der Datenstruktur (`.value` und `.timestamp`) des Sensors gespeichert.



(a) Auslesen der Temperatursensoren



(b) Auswertung der Digitaleingänge

Abbildung 4.18: Abläufe von Funktionen

[4] `https://github.com/DavidAntliff/esp32-ds18b20`

4.3.7 Analogsensoren

Bei der Funktion zum Auslesen der Analogsensoren läuft für alle deklarierten Sensoren der in folgender Abbildung beschriebene Vorgang ab.

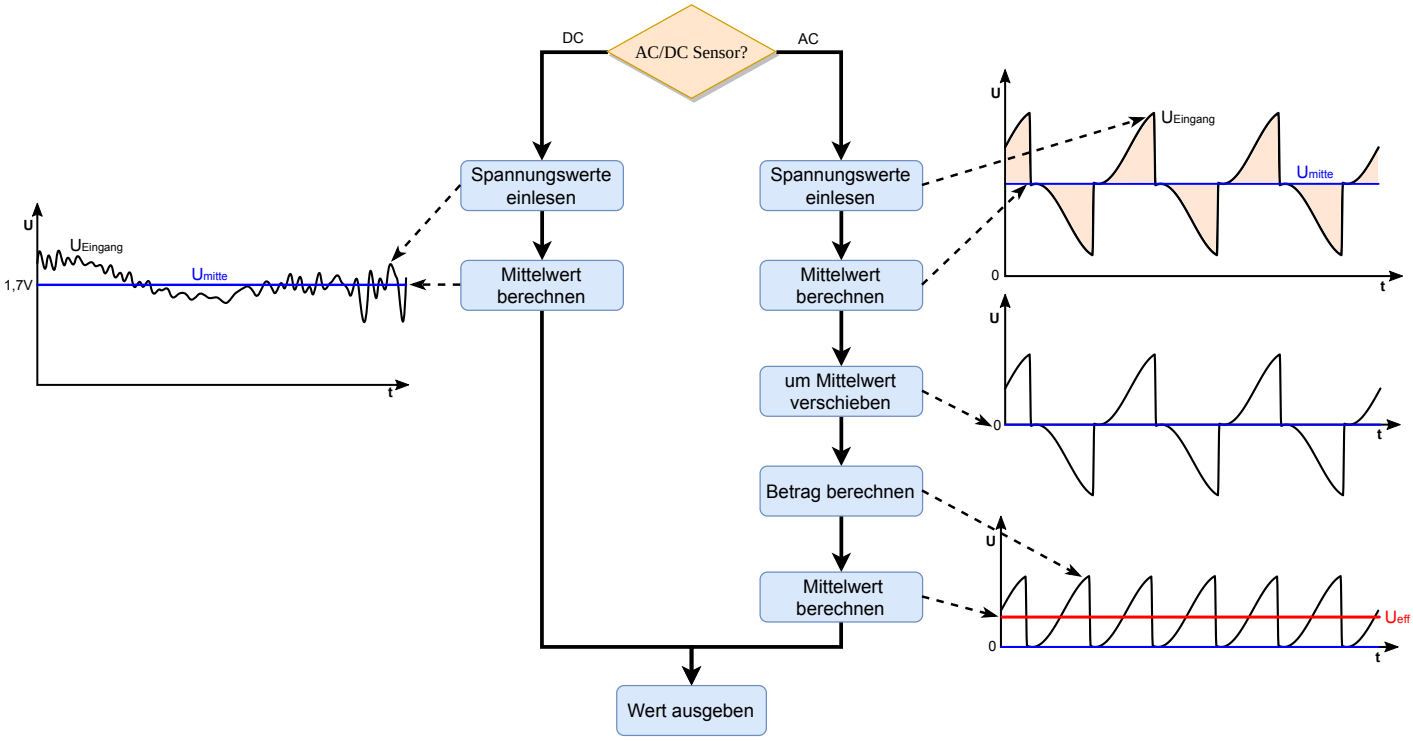


Abbildung 4.19: Auswertung der Analogeingänge

Zuerst wird entschieden, ob es sich um ein Gleichspannungs- oder Wechselspannungssignal handelt. Dies kann für jeden Sensor anhand der bei der Definition gesetzten Variable `MODE` (vgl. Kapitel 4.3.4) entschieden werden.

Gleichspannugsmessung (DC): Um Unregelmäßigkeiten beim Auslesen der Spannung oder kurzzeitige Störungen des Analogsignals zu kompensieren werden über zwei Sekunden 10.000 Spannungswerte gemessen und aus diesen der Mittelwert berechnet (siehe Abbildung 4.19 Diagramm links).

Wechselspannungsmessung (AC):
Bei einer Wechselspannung ist der Effektivwert bzw. die im Diagramm rechts oben angedeutete Fläche unter der Spannungskurve interessant. Um diese zu bestimmen wird im Programm wie folgt vorgegangen:
Wie beim Verfahren für Gleichspannung wird zuerst der Mittelwert gebildet. Diese mittlere Spannung ergibt bei einem Wechselspannungssignal den Spannungswert, um welchen das

Signal schwingt. Nun wird das Eingangssignal durch Abziehen des errechneten Mittelwerts von den Spannungswerten verschoben, sodass dieses um 0 schwingt. Nach Berechnen des Betrags kann erneut ein Mittelwert berechnet werden, welcher dem gesuchten Effektivwert entspricht.

4.3.8 WLAN

Um die integrierte WLAN-Fähigkeit des Mikrocontrollers zu aktivieren, konnten dafür zuständige vorgefertigte Funktionen aus der ESP-IDF Bibliothek verwendet werden.
So wird, direkt nachdem der Mikrocontroller gestartet ist, zuerst das WLAN-Interface mit den Funktionen `wifi_config` und `esp.wifi_init` unter Angabe von in der *config.c* Datei angegebenen WLAN-Kenndaten wie Name (SSID) und Passwort initialisiert.
Anschließend kann das WLAN mit „esp_wifi_start()“ gestartet werden. Dabei weist der im Netzwerk vorhandene DHCP (Dynamic Host Configuration Protocoll) Server dem Mikrocontroller automatisch eine IP-Adresse zu und andere Funktionen wie MQTT können ab sofort diese Schnittstelle nutzen.

4.3.9 MQTT

Der Task, welcher für das Senden und Empfangen von Daten über das Netzwerk mit dem MQTT Protokoll zuständig ist, besitzt zwei Aufgaben:
Über die konfigurierten Topics müssen Daten empfangen und die jeweiligen Funktionen ausgeführt werden. Außerdem werden sämtliche mit den vorher erläuterten Funktionen ermittelten Sensordaten zum MQTT Broker gesendet.

Daten empfangen (Subscribe)

Nach Starten des MQTT Clients werden alle in der Datei *config.c* definierten Topics der Steueraufgaben mit einer Schleife abgearbeitet und jeweils mithilfe der Funktion `esp_mqtt_client_subscribe` aus der ESP-IDF Bibliothek abonniert. Abonnieren bedeutet hier, dass der MQTT Broker (Server) benachrichtigt wird, dass sobald neue Daten am abonnierten Topic ankommen, diese an den Client weitergegeben werden sollen.
Im MQTT Event Handler, welcher aus einer Case-Struktur besteht, wurde im Event Case `MQTT_DATA` eine eigene Funktion programmiert. Diese prüft, ob und mit welchem Topic ankommende Daten mit denen aus der Konfiguration übereinstimmen, und führt dementsprechend deren konfigurierte Funktion aus.
So wird zum Beispiel die in der Datei *config.c* definierte Funktion `pumpeEinschalten` aufgerufen, wenn Daten über das Topic „Pumpe“ empfangen werden.

Daten senden (Publish)

Das Senden der Sensordaten erfolgt nach dem im unteren Teil der Abbildung 4.20 dargestellten Ablauf.
Alle 30 Sekunden werden in einer For-Schleife sämtliche Sensoreinträge aus der Konfigurationsdatei durchlaufen. Zuerst wird anhand des für jeden Senor hinterlegten Timestamps geprüft ob der Wert älter als 60 Sekunden ist. Ist dies nicht der Fall, wird der Gleitkommawert in den Datentyp String umgewandelt, welchen die nächste Funktion `esp_mqtt_client_publish` akzeptiert. Als nächstes sendet diese Funktion den String unter Angabe des zum Wert gehörenden Topics zum MQTT Broker.

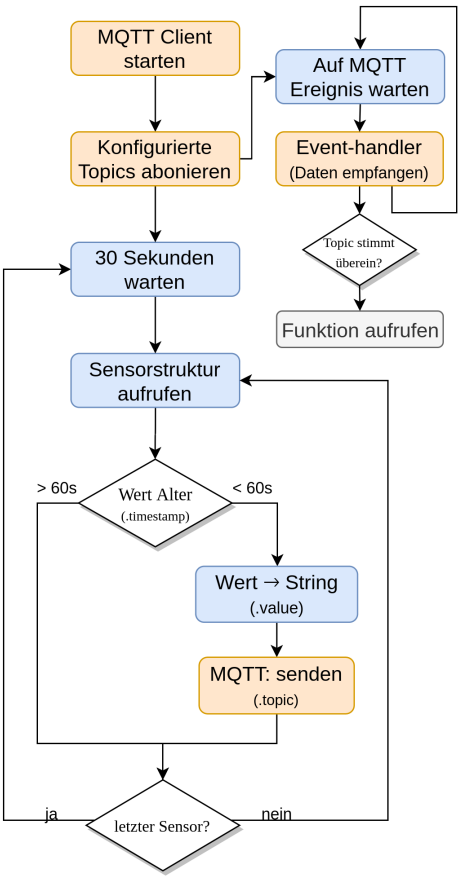


Abbildung 4.20: MQTT Task

4.4 Programmierung Node-Red

Der installierte Node-Red Dienst bietet eine Weboberfläche an, auf welcher alles mit einem beliebig wählbaren Webbrowser programmiert werden kann. Dieser Webserver läuft über den TCP-Port 1880, somit kann diese Seite im Browser mit der URL http://10.0.0.102:1880/ aufgerufen werden.

4.4.1 Daten empfangen und speichern

Eine sehr wichtige Aufgabe, die mit Node-Red umgesetzt wurde, ist das Empfangen und Speichern der Sensordaten. Es musste nach einer Lösung gesucht werden, alle gesendeten Sensorwerte der Platinen über den MQTT-Broker zu erhalten und in der MYSQL-Datenbank zu speichern.

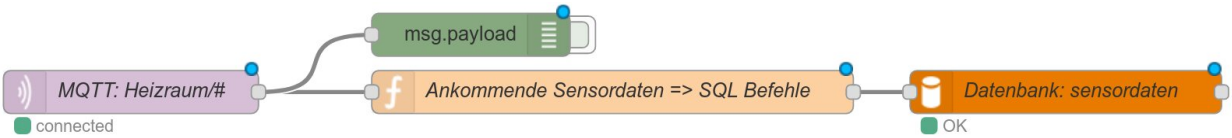


Abbildung 4.21: Node-Red - Sensordaten in Datenbank abspeichern

In Abbildung 4.21 erkennt man bereits das Datenflussprinzip von Node-Red. Jeder Knoten (Node) ist mit anderen verbunden. Wenn über die Verbindung zu einem Knoten Daten anliegen, wird dieser Knoten aufgerufen.
Der linke Node ist mit dem MQTT-Broker verbunden und empfängt die jeweiligen Daten, wenn sich ab dem Topic `Heizraum/` ein Wert verändert. Wie die MQTT-Topics aufgebaut sind, also welche Sensoren sich unter welchem Pfad befinden, wird in der Übersicht in Anhang D deutlich.

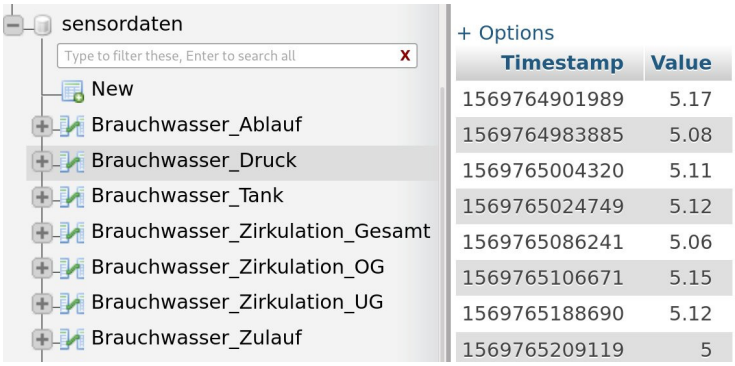


Abbildung 4.22: MYSQL-Datenbank

In Abbildung 4.22 sieht man einen Ausschnitt der MYSQL-Datenbank „sensordaten“. Es wurde für jeden Sensor eine Tabelle mit den Spalten „Timestamp“ und „Value“ angelegt. Die vom MQTT-Broker empfangenen Daten werden an einen Funktionsbaustein übergeben,

in dem mit JavaScript eine Funktion programmiert wurde. Diese wandelt die erhaltenen Strings mit Sensorname und Sensorwert in SQL Befehle um. So ein generierter Befehl sieht beispielsweise so aus:

```
INSERT INTO ‘Brauchwasser_Druck‘ (‘Timestamp‘,‘Value‘) VALUES ("2147483647","5.21")
```

Dieser SQL-Befehl, welcher anschließend durch den orangen Knoten auf dem MYSQL-Server ausgeführt wird, fügt zum Beispiel in der Tabelle „Brauchwasser_Druck“ eine neue Zeile mit Sensorwert und Zeitstempel hinzu.
Auf diese Art und Weise wird jeder ankommende Sensorwert gespeichert und die gesammelten Daten können später mit ähnlichen SQL Befehlen wieder abgerufen werden.

4.4.2 Live-Anzeige

Weiterhin sollen die aktuellen Sensorwerte live auf einer Webseite dargestellt werden.
Mithilfe vorgefertigter Nodes unter der Kategorie Dashboard kann mit Node-Red sehr leicht ein so genanntes User-Interface erstellt werden.

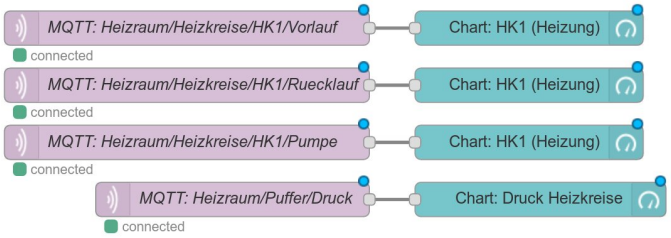


Abbildung 4.23: Node-Red - Sensordaten live anzeigen

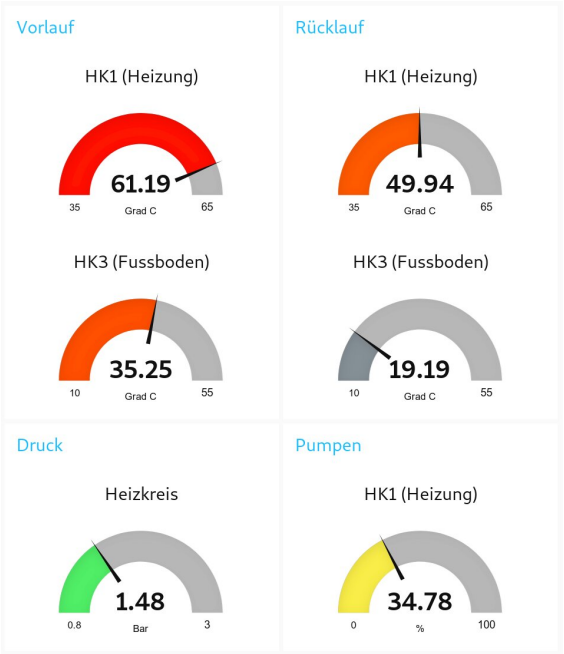


Abbildung 4.24: Ausschnitt der Anzeige von aktuellen Sensorwerten

Um eine Webseite wie in Abbildung 4.24 dargestellt zu realisieren, werden MQTT-Knoten für das jeweilige MQTT-Topic jedes Sensors erstellt, welche neu ankommende Werte sofort an den zugehörigen Knoten weitergeben, der für eine Zeigeranzeige auf der Webseite zuständig ist.

4.4.3 Sensorverläufe

Neben Anzeigen der Livewerte soll auch eine Seite erstellt werden, auf der die Verläufe der Sensordaten über feste bzw. wählbare Zeiträume angezeigt werden. Um das zu realisieren wurde in Node-Red ein sog. SubFlow erstellt, welcher dann im Programm beliebig oft wiederverwendet werden kann.

Wie dieser Subflow aufgebaut ist, sieht man in Abbildung 4.25. Er setzt sich aus mehreren Knoten zusammen: Es werden SQL Befehle zum Anfordern der relevanten Sensorwerte aus der Datenbank generiert, Mittelwerte berechnet und die Struktur der Daten für den in Abbildung 4.26 zu sehenden, nachfolgenden Knoten angepasst.

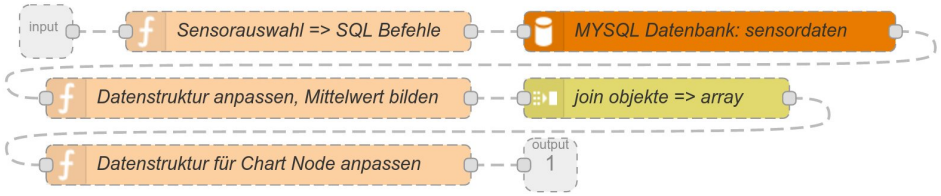


Abbildung 4.25: Node-Red - SubFlow Chart aus MYSQL

In Abbildung 4.26 sieht man, dass dieser SubFlow durch den linken blauen Knoten in regelmäßigen Intervallen (5-60 min, je nach Zeitraum) mit Übergabe gewisser Parameter wie Zeitraum und Sensornamen aufgerufen wird. Das Ergebnis wird dann an einen Node weitergeleitet, der den jeweiligen Chart auf der Website generiert.

Abbildung 4.26: Node-Red - Charts generieren

Ein Ausschnitt dieser entstandenen Seite mit mehreren dieser Sensorverläufe ist in Anhang E zu finden.

Wenn ein Chart für einen längeren Zeitraum (z. B. mehrere Tage oder Monate) generiert werden soll, ist es nicht sinnvoll alle in diesem Zeitraum vorhandenen Werte im Chart darzustellen. Gründe dafür sind, dass dem Webbrowser eine große Datenmenge übermittelt werden muss und es für diesen sehr aufwändig ist, aus den unzähligen Punkten einen Graphen darzustellen.
Die Herausforderung bei der Entwicklung dieses SubFlows war es also, einen Algorithmus zu finden, mit dem tausende Werte auf wenige Hundert reduziert werden können.
In Auflistung 4.2 sieht man einen Ausschnitt des mit JavaScript programmierten Knotens „*Datenstruktur anpassen, Mittelwert bilden*“.
Anhand der, mit `//` beginnenden Kommentare kann die Vorgehensweise nachvollzogen werden, wie eine beliebige Anzahl an Werten aus der Datenbank mithilfe von Mittelwertberechnungen in eine definierte Anzahl umgewandelt wird.

Auflistung 4.2: Mittelwerte der Daten für Charts bilden

```
1  //Variablendeklaration
2  var data_sensor = [];
3  var i, j, k, sum;
4  var object;
5
6  //====Parameter zum Berechnen der Mittelwerte bestimmen====
7  //Anzahl der aus der Datenbank erhaltenen Eintraege bestimmen
8  var value_count = Object.keys(msg.payload).length;
9
10 //Anzahl der geforderten Werte aus lokaler Variable definieren
11 var accuracy = flow.get('anzahl_werte');
12
13 //Anzahl der Werte pruefen
14 if (value_count > accuracy){    //Mehr Werte vorhanden, als gefordert wird
15     //Anzahl der Werte, die jeweils zusammengefasst werden muessen bestimmen (Gruppe)
16     var average = Math.floor(value_count/accuracy);
17 }else{
18     //Jeder Wert wird uebernommen
19     var average = 1;
20 }
21
22
23 //=====Mittelwert bilden, Struktur anpassen======
24 //Jede Gruppe an Werten aus der Datenbank durchlaufen
25 for (i=0; i <= msg.payload.length - average; i=i+average){
26
27             //Aus Gruppe der Werte aus Datenbank Mittelwert bilden
28             sum = 0;
29             for (j=i; j<i+average; j++){
30                 sum = sum + msg.payload[j].Value;
31             }
32             ValueAverage = sum/average;
33
34             //Aus Gruppe der Timestamps aus Datenbank Mittelwert bilden
35             sum = 0;
36             for (j=i; j<i+average; j++){
37                 sum = sum + msg.payload[j].Timestamp;
38             }
39             TimestampAverage = sum/average;
40
41             //Objekt aus Mittelwerten dieser Gruppe erstellen
42              object =
43                 x: TimestampAverage,
44                 y: ValueAverage
45             };
46
47             //Objekt dem Array fuer Sensorchart hinzufuegen
48             data_sensor[k] = object;
49             k++;
50     }
51
52 //====Sensorinformation und Daten ausgeben====
53 msg.payload = {sensor: sensorname, data: data_sensor};
54 return msg;
```

4.4.4 E-Mail Benachrichtigung

Mit Node-Red können, durch Verwendung eines vorgefertigten Nodes sehr leicht E-Mails versendet werden. Der E-Mail Knoten muss mit allen Informationen eines vorhandenen Mail-Accounts konfiguriert werden, wie SMTP Serveradresse, Benutzername und Passwort.

Sobald der Node Daten in Form eines Strings erhält, wird dieser Text an die bei der Konfiguration angegebenen E-Mail Adresse gesendet.

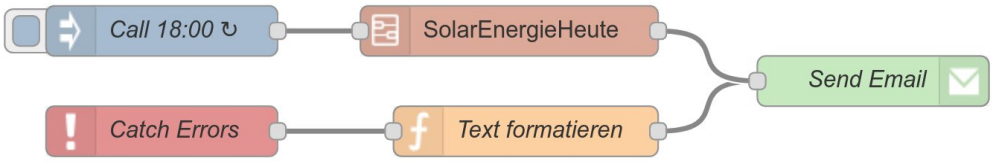


Abbildung 4.27: Node-Red - E-mails versenden

Im oben gezeigten Programmausschnitt wird täglich um 18 Uhr die erzeugte Solarenergie mithilfe einer ausgelagerten Funktion berechnet und anschließend als Mail versendet. Der untere Knoten fängt eventuelle Fehlermeldungen von anderen Knoten (z. B. aus Funktionen oder der Datenbank) auf. Diese werden als nächstes, zu einer übersichtlichen Mail formatiert und ebenfalls mit demselben Node versendet.

Komplexere, prozessabhängige Benachrichtigungen wie zum Beispiel Temperaturwarnungen wurden mit so genannten Zustandsautomaten realisiert.
In nebenstehender Auflistung 4.3 wird ein Ausschnitt eines JavaScript Funktionsknotens gezeigt. Dieser Knoten wird in regelmäßigen Intervallen mit Übergabe relevanter Sensorwerte aus der MYSQL-Datenbank sowie Daten aus dem letzten Durchlauf aufgerufen.
In der gezeigten Switch-Case-Struktur wird anhand dieser Daten entschieden, in welchem Zustand sich die Anlage befindet.

Der Vorteil des Zustandsautomaten ist, dass nur eine Mail, exakt beim Wechseln in einen bestimmten Zustand, versendet wird und nicht etwa bei jedem Aufruf, wenn die Temperatur über einem bestimmten Wert liegt.
So wird hier zum Beispiel in Zeile 26 der Auftrag zum Senden einer Mail erteilt, wenn der Zustand des Pufferspeichers bei 95 °C von `HEISS` auf `KRIT` wechselt.
Das `msg.mail` Objekt wird den nächsten beiden Knoten übergeben, welche, wie in Abbildung 4.27, die jeweilige Nachricht generieren und als E-Mail versenden.

Auflistung 4.3: Zustandsautomat Pufferspeicher

```
1  switch (STATE_PUFFER){
2
3    default:
4      STATE_PUFFER = "KALT";
5    break;
6
7    case "KALT":
8      if ( Puffer_temp > 50 ){
9        STATE_PUFFER = "WARM";
10     }
11   break;
12
13   case "WARM":
14     if ( Puffer_temp > 70 ){
15        STATE_PUFFER = "HEISS";
16     }
17     else if ( Puffer_temp < 40 ){
18        STATE_PUFFER = "KALT";
19        msg.mail.STATE_PUFFER = "KALT";
20     }
21   break;
22
23   case "HEISS":
24     if ( Puffer_temp >95 ){
25        STATE_PUFFER = "KRIT";
26        msg.mail.STATE_PUFFER = "KRIT";
27     }
28     else if ( Puffer_temp < 65 ){
29        STATE_PUFFER = "WARM";
30     }
31   break;
32
33   case "KRIT":
34     if ( Puffer_temp < 90 ){
35        STATE_PUFFER = "HEISS";
36     }
37   break;
38 }
```

5. Fazit

5.1 Erkenntnisse Heizung

Durch die gesammelten Sensordaten kann die in Kapitel 2.1 beschriebene komplexe Heizungsanlage sehr gut veranschaulicht und die Vorgänge verstanden werden.
Auf der mit Node-Red erstellten Website ist es möglich die Werte von Sensoren aus einem wählbaren Zeitraum in einem Chart darzustellen. In Abbildung 5.1 sind als Beispiel zwei dieser Charts zu sehen. Im oberen sind die Puffertemperaturen, im unteren die Systemdrücke über einen Zeitraum von vier Tagen dargestellt.

Abbildung 5.1: Puffertemperaturen und Systemdrücke über vier Tage

Man erkennt, dass das Wasser im Pufferspeicher jeden Vormittag durch die Holzheizung erwärmt wurde und die Wärme im Laufe das Tages verbraucht wird. Außerdem fällt auf, dass der Druck im Heizsystem (blau) sich wie die Puffertemperatur verhält und im Bereich von 1,0-2,5 bar schwankt. Dieses Verhalten lässt sich durch die Wärmeausdehnung der über

3700 L Wasser bei der ständigen Temperaturänderung erklären.
Wenn man die Puffertemperaturen im rechten Bereich genauer betrachtet, stellt man einen kurzen Anstieg und unmittelbar danach gleich wieder ein Abfallen der Temperatur fest. Die Ursache dafür war ein fehlerhaftes, zu frühes Abschalten des Holzkessels durch die Heizungssteuerung. Dieses Fehlverhalten wurde durch einen wie in Kapitel 4.4.4 beschriebenen Zustandsautomaten erkannt und per E-Mail eine Benachrichtung versendet.
Die Unregelmäßigkeiten im Brauchwasserdruck (orange) entstehen vermutlich durch Abfallen des Drucks bei Wasserverbrauch, Druckstöße beim abrupten Abschalten, sowie durch Wärmeausdehnung des Wassers im Warmwassertank.

5.2 Kosten

5.2.1 Materialkosten

Die Materialkosten der Projektarbeit setzen sich neben diversen Kleinteilen hauptsächlich aus den Kaufpreisen von Sensoren, Verdrahtungsmaterial, Rohmaterial für die gefertigten Sensorhalter sowie den Kosten der Platinen zusammen.
Aus der erstellten Stückliste der entwickelten Platine in Anhang F.2, in der alle Bauteile einer Platine aufgelistet sind, geht hervor, dass eine voll bestückte Platine 11,08 € kostet.
Es wurden insgesamt zwei dieser Platinen angefertigt: Eine für sämtliche Sensoren in der Nähe von Heizung und Brauchwassertank (Abbildung 4.9) und die zweite auf der anderen Seite des Raums für Sensoren nahe der Solaranlage.
Insgesamt belaufen sich die Anschaffungskosten, für das Realisieren des Systems zur Überwachung der Heizungsanlage, auf 132,00€. Wie sich diese Summe genau zusammensetzt, kann in der Stückliste des gesamten Projekts in Anhang F.1 nachvollzogen werden. Die Kosten der Entwicklung, Produktion und Installation wurden dabei nicht berücksichtigt.

5.2.2 Laufende Kosten

Laufende Kosten entstehen durch das aufgebaute System ausschließlich durch den Stromverbrauch des Servers, der Platinen und der daran angeschlossenen Sensoren.
Die Gleichstromaufnahme der selbst gefertigten Komponenten wurde bereits nach der Montage einmal gemessen um eventuelle Fehler festzustellen und um die Netzteile richtig auslegen zu können. Diese Messwerte eignen sich aber nicht zur Kostenberechnung, da der Wirkungsgrad der vorgeschalteten Netzteile noch nicht berücksichtigt ist.
Um die tatsächliche Energie, die dem 230V Stromnetz entnommen wird, zu messen, wurde ein Energiemessgerät für mehrere Tage vor die 230V Versorgung der jeweiligen Netzteile geschaltet. Aus diesen Energiewerten und der entsprechenden Messdauer konnte anschliessend mit folgender Formel die durchschnittliche Leistungsaufnahme der Komponenten berechnet werden:

$$P = \frac{W}{t} \qquad P_{Platine} = \frac{86{,}4\,\mathrm{Wh}}{72\,\mathrm{h}} = \underline{1{,}2\,\mathrm{W}}$$

Mittels der bestimmten Leistung können nun die Jahresstromkosten berechnet werden:

$$K = 365 \cdot 24 \frac{\text{h}}{\text{Jahr}} \cdot 1{,}5 \cdot 10^{-3}\,\text{kW} \cdot 0{,}30 \frac{€}{\text{kWh}} = \underline{\underline{3{,}2 \frac{€}{\text{Jahr}}}}$$

Komponente	DC Spannung und Stromstärke	Leistungsaufnahme bei 230V	Kosten pro Jahr
Server	-	30W	79€
Platine Heizung	12V 93mA	1,2W	3,2€
Platine Solaranlage	12V 81mA	1,1W	2,9€

Tabelle 5.1: Energieverbrauch

Vergleicht man die Leistung aus den gemessenen Gleichstromwerten und die errechnete Aufnahmeleisung bei 230V, fällt ein Unterschied von ca. 10% auf. Dieser lässt sich durch die Verlustleistung und somit den Wirkungsgrad des jeweils vorgeschalteten Netzteils erklären.
Bei einem Strompreis von 0,30 $\frac{€}{\text{kWh}}$ ergeben sich also für beide Platinen inklusive der angeschlossenen Sensoren jährliche Kosten von ca. 6,10€. Die Stromkosten des Servers können nicht komplett auf dieses Projekt angerechnet werden, da dieser hauptsächlich für andere Anwendungen verwendet wird.

5.2.3 Speicherbedarf

Da sämtliche Sensoren in einem festen Intervall ausgelesen werden und diese Sensordaten auch in einer Datenbank gespeichert werden, ist ein ständig wachsender Speicherbedarf vorhanden.
Um diesen auf dem Server benötigten Speicherplatz zu berechnen wurde zuerst analysiert, wie viel Speicher ein Eintrag eines Sensorwertes in der MYSQL-Datenbank benötigt.
Für jede gespeicherte Messung wird in der Datenbank der Sensorwert und ein Zeitstempel gespeichert (siehe Abbildung 4.22). Die Variable Timestamp wird als Datentyp BIGINT (64 Bit), die Variable Value als FLOAT (32 Bit) abgespeichert.
Aus dem Speicherbedarf dieser Variablen alleine lässt sich jedoch nicht die genaue Anzahl der benötigten Bits bestimmen, da die MYSQL-Datenbank diese etwas anders im tatsächlichen Festplattenspeicher ablegt, sowie weitere Daten pro Tabelle und Zeile hinzugefügt werden. So werden die Werte zum Beispiel immer in 4-Byte-Blöcken abgelegt sowie ein bis zu 35 Byte großer Schlüssel pro Zeile erstellt. [14]

Da hier nur die grob benötigte Speicherkapazität bestimmt werden soll, reicht es wenn ein guter Näherungswert erhalten wird.
Mithilfe des MYSQL-Befehls
`'SHOW TABLE STATUS FROM sensordaten'`, welcher auf dem MSQL-Server ausgeführt wurde, konnte eine Übersicht der bereits gespeicherten Sensortabellen angezeigt werden (Abbildung 5.2).

Name	Data_length	Rows	Avg_row_length
HK1_Ruecklauf	4734976	123818	38
HK1_Vorlauf	4734976	121836	38
HK2_Pumpe	2637824	57555	40
HK2_Ruecklauf	4734976	123976	38
HK2_Vorlauf	4734976	123888	38
HK3_Pumpe	2637824	56529	40
HK3_Ruecklauf	4734976	124343	38
HK3_Vorlauf	4734976	124753	37

Abbildung 5.2: Mysql: Ausschnitt Tabelleninfo

In der rechten Spalte wird der durchschnittliche Speicherbedarf einer Zeile in Bytes, also eines Eintrags (Timestamp und Value), angezeigt. Da diese annähernd gleich sind, kann der Speicherbedarf näherungsweise wie folgt berechnet werden:

$$N = \text{Speicherbedarf-pro-Zeile} \cdot \text{Messrate} \cdot \text{Sensoranzahl} \cdot \text{Zeitraum}$$

$$N = 39\,\text{Byte} \cdot 120\,\frac{1}{\text{h}} \cdot 31 \cdot 8760\,\frac{\text{h}}{\text{Jahr}} = \underline{\underline{1{,}18\,\frac{\text{GiB}}{\text{Jahr}}}}$$

Bei einer Messrate von zwei Werten pro Minute entsteht mit den vorhandenen 31 Sensoren also eine Datenmenge von bis zu 1,18 GiB pro Jahr, welche auf der 465 GiB SSD des Servers gespeichert wird.

5.3 Zukünftige Erweiterungen

Die Aufgabe der Projektarbeit war das Entwickeln eines erweiterbaren Systems zur Erfassung von Daten. Da dieses nun erfolgreich entwickelt und bereits einmal angewendet wurde, folgen in Zukunft noch einige Erweiterungen sowie andere Projekte, für welche die universell einsetzbare Platine verwendet werden kann.

Weitere Benachrichtigungen

Um weiter vom Projekt der Heizungsüberwachung zu profitieren, ist geplant weitere E-Mail Benachrichtigungen zu programmieren wie z. B.:

- Tägliche Infomail mit wichtigsten Daten
- Benachrichtigung über Heiztermin
- Benachrichtigung über Nachlegezeitpunkt
- Warnung bei Über- bzw. Unterschreiten der Druckbereiche
- Warnung vor Überhitzung der Solaranlage

Da alle Sensordaten bereits zentral gesammelt werden, können mit Node-Red sehr leicht weitere Benachrichtigungen realisiert werden.
In Zukunft kann noch nach einer Möglichkeit gesucht werden, alternativ zu E-Mails unkomplizierte Push-Benachrichtigungen an z. B. Handys zu senden. Da diese auf Dauer weniger Spam, der regelmäßig gelöscht werden muss, verursachen.

Selbstlernende Pumpensteuerung

In der Heizungsanlage ist eine Zirkulationspumpe im Brauchwasserkreis vorhanden, welche das warme Brauchwasser zwischen Brauchwassertank und den Wasserverbrauchern zirkulieren lässt (siehe Abbildung 2.1).

Dies soll bewirken, dass man beim Öffnen eines Wasserhahns, der weit vom Heizraum entfernt ist, nicht lange auf warmes Wasser warten muss. Der Nachteil dabei ist, dass durch ständiges Zirkulieren des warmen Wassers dauerhaft Energie verloren geht, auch wenn kein Warmwasser verbraucht wird. Dieser Energieverlust bei laufender Zirkulationspumpe konnte an den bereits gesammelten Sensordaten deutlich erkannt werden.
Deshalb ist geplant, über Node-Red anhand früherer Daten vorherzusagen an welchen Uhrzeiten Warmwasser gebraucht werden wird. Mit diesem Ergebnis soll die Zirkulationspumpe zu diesen Zeitpunkten und nur so lange wie nötig eingeschaltet werden.

Intelligente Warmwasserverwaltung

Ein weiterer praktischer Nutzen des aufgebauten Sensornetzwerks wäre eine intelligente Warmwasserverwaltung. So soll anhand der gesammelten Daten festgestellt werden, ob überschüssige Wärmeenergie von der Solaranlage vorhanden ist.
Es ist geplant bei Geräten wie z. B. Geschirrspühler oder Waschmaschinen Platinen anzubringen, welche mithilfe eines Schrittmotors ein Brauchwassermischventil steuern können. Aufgrund der Sensordaten aus dem Heizraum kann über Node-Red gesteuert werden, ob diese Wasserverbraucher mit heißem Wasser aus dem Brauchwassertank oder mit kaltem Wasser betrieben werden.
Somit kann überschüssige Solarenergie genutzt und elektrische Energie, die ansonsten für das Aufheizen des Wassers genutzt wird, gespart werden.

Energiemessung

Neben dem Überwachen der Heizungsanlage ist auch geplant die aufgenommene elektrische Leistung des Hauses sowie die eingespeiste Leistung der Photovoltaikanlage dauerhaft zu messen.
Das kann durch Anbringen einer weiteren Platine und mehrerer Stromwandler in der Zählerverteilung umgesetzt werden. Aus diesen Messwerten können dann Übersichten erstellt und Energiewerte berechnet werden.

Solarbetriebene Außenstation

In Zukunft kann mithilfe der entwickelten Platine auch eine Art Wetterstation realisiert werden. Dafür ist geplant ein kleines Solarpanel, Laderegler und Platine in einem wasserdichten Gehäuse unterzubringen. An diesem können Sensoren zur Messung von Wind, Temperatur und Feuchtigkeit angeschlossen werden.
Durch die Versorgung mit Solarstrom kann diese Einheit kabellos beliebig innerhalb der WLAN Reichweite platziert werden.

Speicheroptimierung

Obwohl pro Sensorwert nur zwei Variablen mit insgesamt 94 Bit Speicher abgelegt werden müssen, nimmt die MYSQL-Datenbank dafür bis zu 312 Bit in Anspruch. Der Speicherbedarf könnte also theoretisch um ein Vielfaches reduziert werden.

Ansätze dafür wären die Verwendung einer anderen MYSQL-Datenbankstruktur, oder einer ganz anderen Methode zum Abspeichern der Daten wie z. B. einer Datei im json oder binär Format.

Sicherheitsgedanken

Bevor neue Projekte gestartet werden, muss eine Möglichkeit gesucht werden, Unbefugten den Zugriff auf die Daten zu verwehren.
Dies kann zum Beispiel durch eine Passwortauthentifizierung auf der Website, über die alle Sensorwerte angezeigt werden, geschehen. Auch die Übertragung der Sensordaten zwischen Platine und Node-Red ist derzeit unverschlüsselt. Diese Übertragung mit dem MQTT-Protokoll kann mit einer SSL-Verschlüsselung erweitert werden.

6. Quellen

[1] Elektro Leipold. Firmenbeschreibung, Photo Firmeneingang. URL: http://www.elektro-leipold.de/ (besucht am 25. 10. 2019).

[2] Wikipedia die freie Enzyklopädie. *Mikrocontroller*. Definition Mikrocontroller. URL: https://de.wikipedia.org/wiki/Mikrocontroller (besucht am 25. 10. 2019).

[3] randomnerdtutorials.com. Abbildung 3.1. 2019. URL: https://randomnerdtutorials.com/esp32-pinout-reference-gpios/ (besucht am 25. 10. 2019).

[4] *ESP32-WROOM-32 Modul*. Technisches Datenblatt. Espressif Systems. 2019.

[5] Jan-Philipp Schäfers. Erklärung Bandabstandsreferenz. URL: https://de.linkfang.org/wiki/Bandabstandsreferenz (besucht am 15. 02. 2020).

[6] Maxim Integrated Products, Inc. *1-Wire Communication*. Definition 1-Wire Bus. URL: https://www.maximintegrated.com/en/design/technical-documents/app-notes/1/126.html (besucht am 15. 01. 2020).

[7] *DS18B20 1-Wire Digital Thermometer*. Technisches Datenblatt. Maxim Integrated, Inc. 2019.

[8] Wikipedia die freie Enzyklopädie. *Drucksensor*. Funktionsweise Keramik-Dickschichtsensor. URL: https://de.wikipedia.org/wiki/Drucksensor (besucht am 15. 01. 2020).

[9] Mikrocontroller.net. *Stromwandler*. Funktionsweise Stromwandler. URL: https://www.mikrocontroller.net/articles/Stromwandler (besucht am 15. 01. 2020).

[10] mqtt.org. Definition MQTT, Logo. URL: https://mqtt.org/ (besucht am 10. 02. 2020).

[11] OPC Foundation. Funktionen der OPC-UA. URL: https://opcfoundation.org/about/opc-technologies/opc-ua/ (besucht am 10. 02. 2020).

[12] Wikipedia die freie Enzyklopädie. *MQTT*. Definition MQTT. URL: https://de.wikipedia.org/wiki/MQTT.

[13] Wikipedia die freie Enzyklopädie. *Node-RED*. Definition Node-Red. URL: https://de.wikipedia.org/wiki/Node-RED (besucht am 25. 10. 2019).

[14] Oracle Corporation. Spezifikation MYSQL. URL: https://dev.mysql.com/doc/ (besucht am 15. 01. 2020).

[15] Oracle Corporation. Definition Datenbank, MYSQL. URL: https://www.oracle.com/de/database/what-is-database.html (besucht am 15. 01. 2020).

[16] Wikipedia die freie Enzyklopädie. *Datenbank*. Definition Datenbank. URL: https://de.wikipedia.org/wiki/Datenbank (besucht am 15. 01. 2020).

[17] *TPS56 Step-Down Voltage Regulator*. Technisches Datenblatt. Texas Instruments Incorporated. 2019. URL: http://www.ti.com/lit/ds/symlink/tps562200.pdf.

[18] Anton Schweizer. Wärmeleitfähigkeiten Metalle. URL: https://www.schweizer-fn.de/ (besucht am 15. 01. 2020).

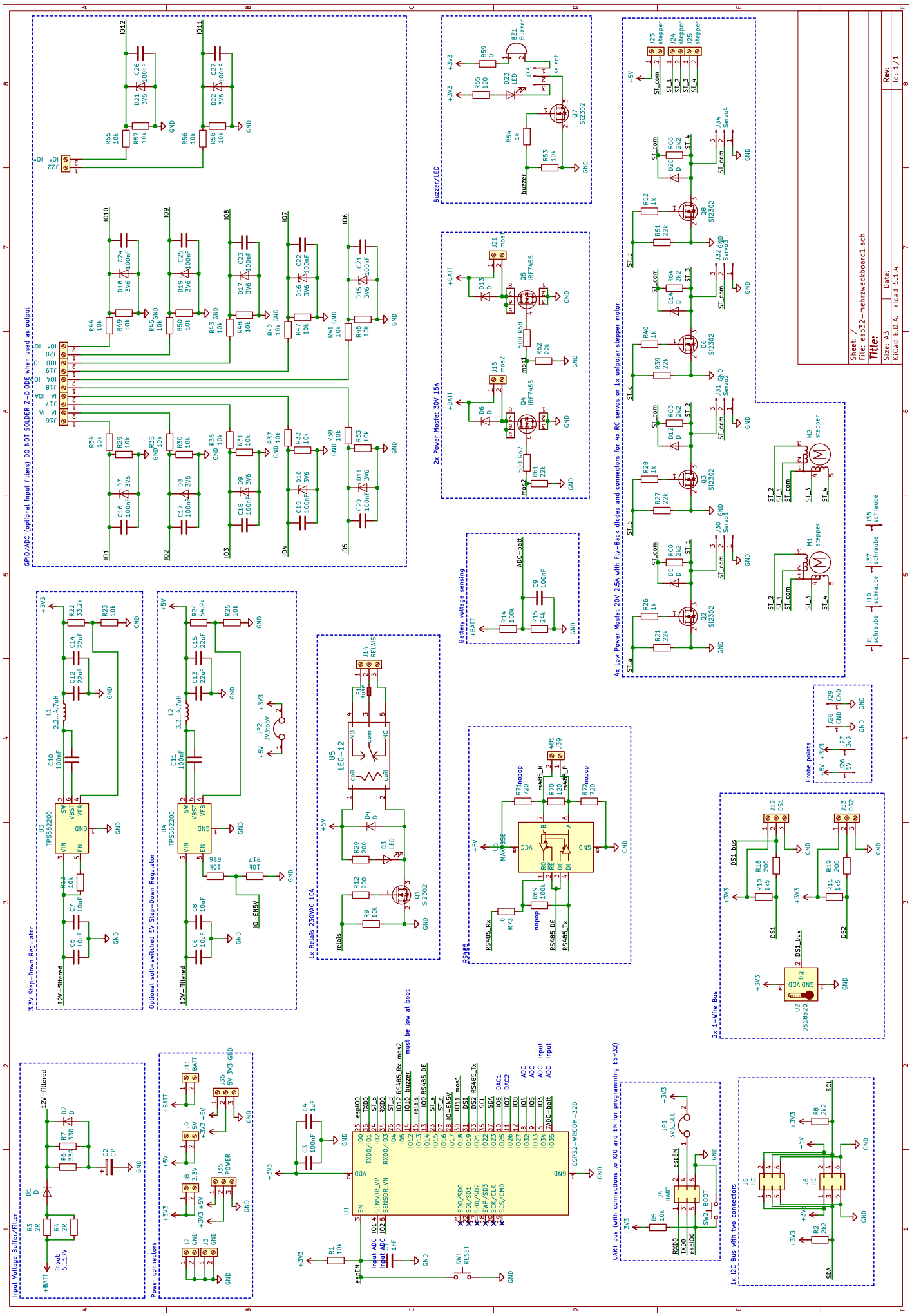
[19] Espressif Systems. Bauteilbeschreibung ESP32. URL: https://www.espressif.com/ (besucht am 25. 10. 2019).

[20] Wikipedia die freie Enzyklopädie. *Toolchain*. Definition Toolchain. URL: https://en.wikipedia.org/wiki/Toolchain (besucht am 15. 01. 2020).

[21] FreeRTOS. Betriebssystem, Tasks und Scheduler. URL: https://www.freertos.org/ (besucht am 15. 02. 2020).

Anhang

A Schaltplan Platine



B Layout Platine

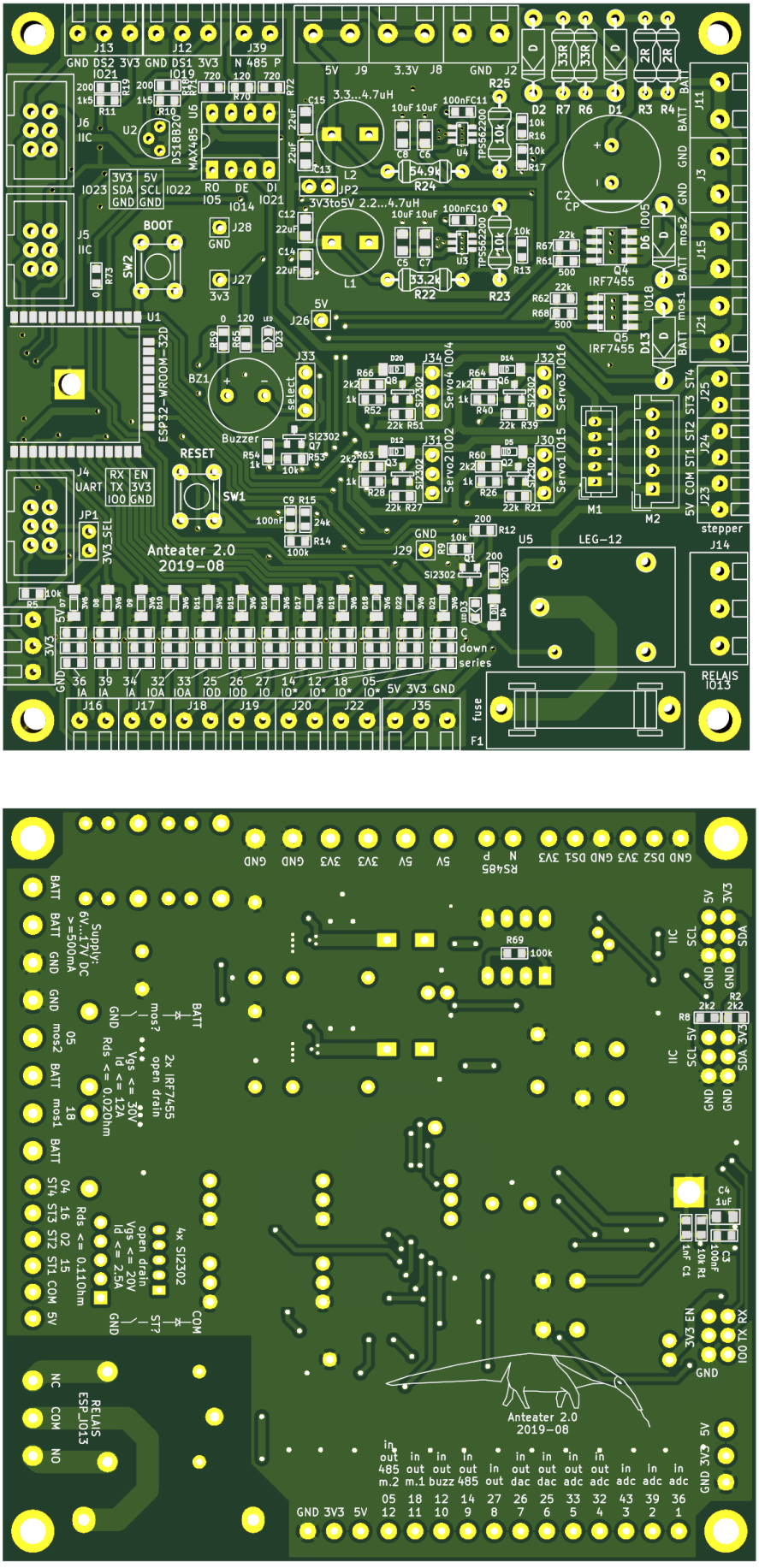
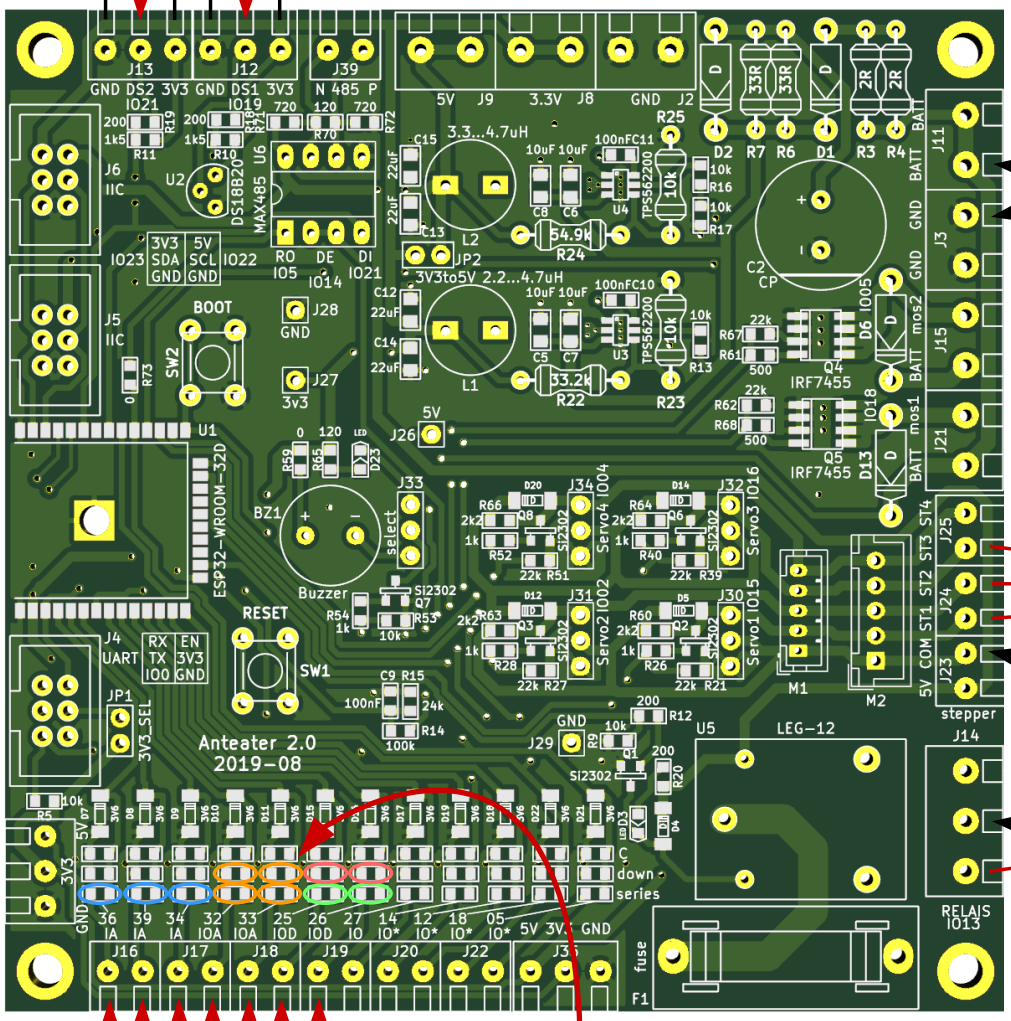


Abbildung 2.1: Modell der fertig gerouteten Platine, Vorder- und Hinterseite

C Anschlussplan

One Wire Bus (OWB) der Temperatursensoren
Versorgung und Datenpins sämtlicher Leitungen
können beliebig auf DS1 und DS2 aufgeteilt werden.



Versorgung:
12V Schaltnetzteil
GND Schaltnetzteil

OUT Digital:
LED Gehäuse rot
LED Gehäuse grün
LED Gehäuse blau

3,3V

OUT Relais:
230V
Zirkulationspumpe

IN Analog: Stromsensor Pumpe Hk1 (0-3V)
IN Analog: Stromsensor Pumpe Hk3 (0-3V)
IN Analog: Abgastemperatur PT100 (~15-50mV)
IN Analog: Drucksensor 80psi Hk/Puffer (0-5V)
IN Analog: Drucksensor 150psi Kaltwasser (0-5V)
IN Digital: Stromsensor Pumpe HK2
IN Digital: Stromsensor Ladepumpe

Spannungsteiler
Widerstandswerte:
1,5k Ohm
0 Ohm
20k Ohm
10k Ohm

Kabelbelegung Drucksensoren
1: GND (schwarz)
2: 5V (rot)
3: Analogsignal Heizkreis (grün)
4: Analogsignal Kaltwasser

Kabelbelegung Temperatursensoren
1: GND (blau)
2: 3,3V (schwarz)
5: OWB Signal (braun)

Kabelbelegung Brauchwassertank
1: GND
2: 3,3V
3: 5V
4: OWB (Temperatursensoren)
5: Zirkulationspumpe
6: Reserve

Kabelbelegung Ofensteuerung
1: GND
2: 12V Netzteil
3: 3,3V
4: 5V
5: Analog Stromsensor Pumpe HK1
6: Analog Stromsensor Pumpke HK3
7: Abgastemperatursensor PT100
8: Digital Stromsensor Pumpe HK2
9: Digital Stromsensor Pumpe Ladekreis
11: Mosfetausgang

D MQTT Struktur

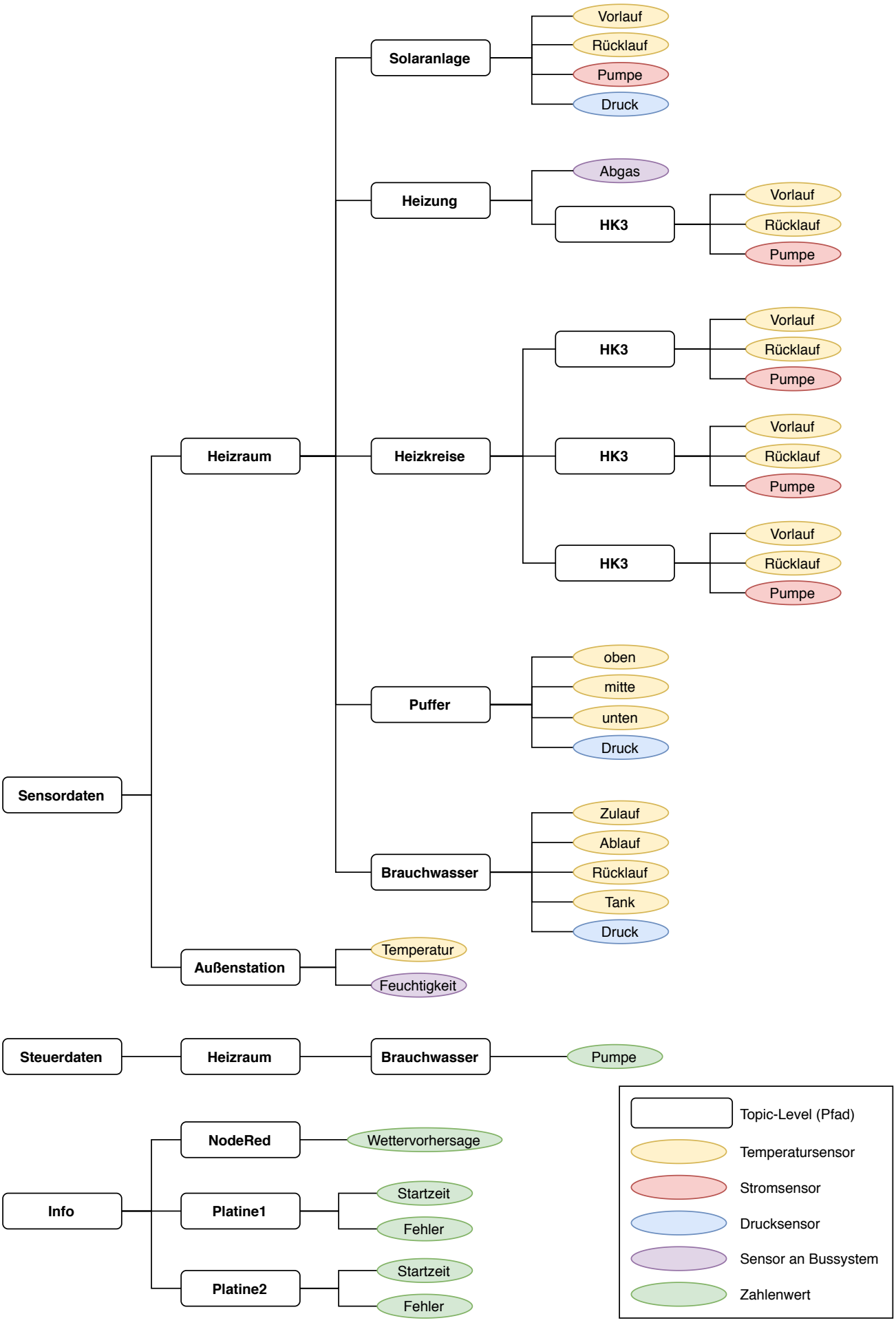
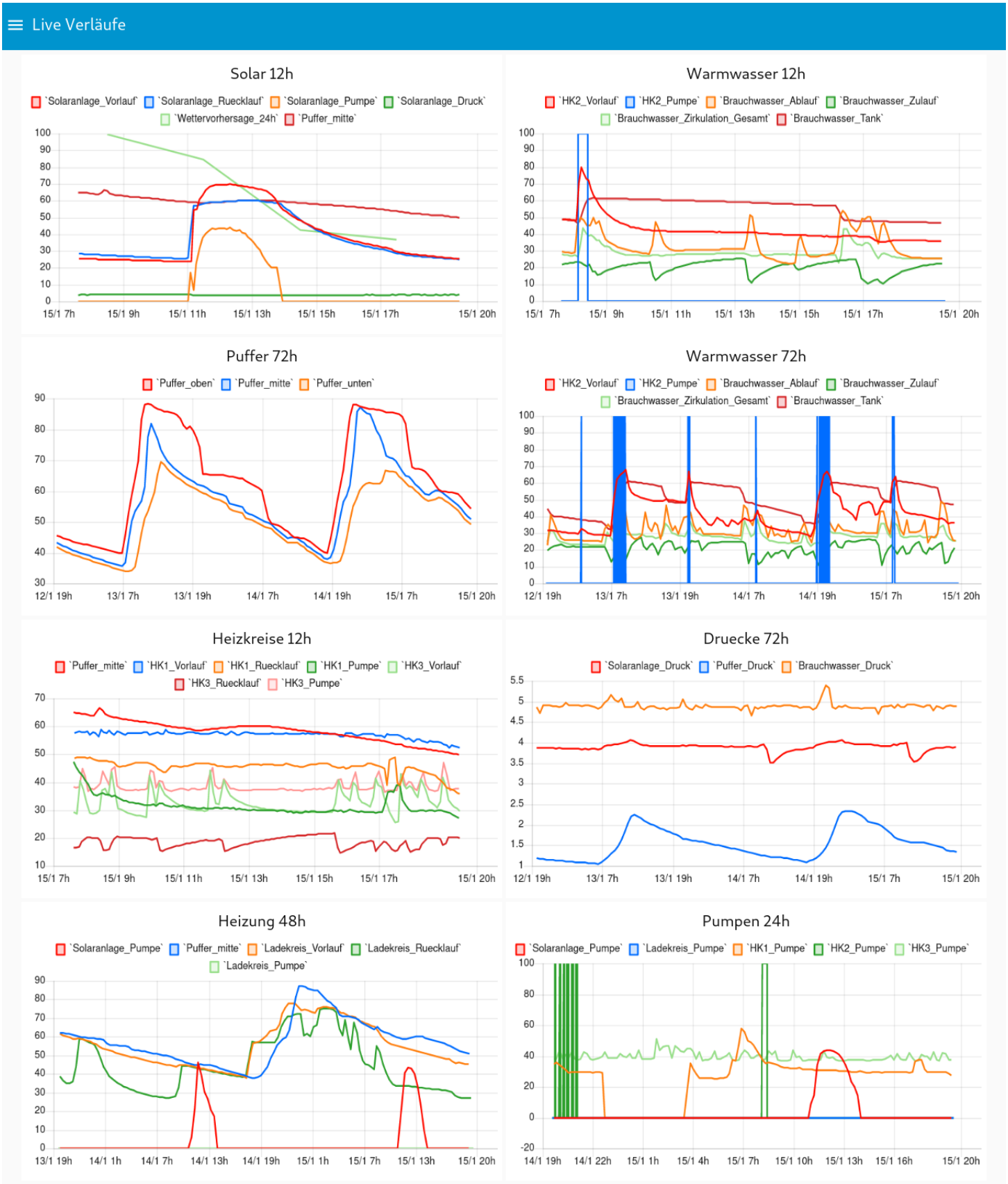


Abbildung 4.1: Übersicht aller Sensoren und Werte im MQTT Netzwerk

E Weboberfläche - Charts



F Stücklisten

F.1 Stückliste Projekt

Pos	Stück	Bezeichnung / Typ	Beschreibung	Preis	Gesamt
1	2	Anteater 2.0	Platine	11.08 €	22.16 €
2	20	DS18B20	Temperatursensor	1.30 €	26.00 €
3	1	PT100D 1/8‘‘	Abgastemperaturfühler	9.03 €	9.03 €
4	3	1/8‘‘NPT 80/150PSI 5V	Drucksensor Druckaufnehmer	8.41 €	25.23 €
5	2	Digital 5V	Stromwandlermodul (Stromerkennung)	2.12 €	4.24 €
6	2	Analog 5V bis 5A	Stromwandlermodul (Stromsensor)	1.19 €	2.38 €
7	2	12V 300mA	Miniatur Schaltnetzteil	1.73 €	3.46 €
8	10	4x0,5mm²	Ölflex Steuerleitung	0.44 €	4.40 €
9	5	14x0,5mm²	Ölflex Steuerleitung	1.57 €	7.85 €
10	10	H05V-K 1x0,5mm²	Verdrahtungsleitung	0.07 €	0.70 €
11	50	0.5N Weiss verzinnt	Aderendhülsen	0.02 €	0.75 €
12	2	PLA Filament 110g	Gehäuse	1.32 €	2.64 €
13	1	CuZn39 20x150mm	Messing Rundmaterial für Drucksensorhalter	5.50 €	5.50 €
14	1	AlCuMg 20x1000mm	Aluminium Rundmaterial für Temperatursensorhalter	15.00 €	15.00 €
15	20		Metallkabelbinder	0.10 €	2.00 €
16	4	2,2x6,5 TX6	Flachkopfschraube für Platine	0.01 €	0.04 €
17	4	M5	Mutter für Gehäusedeckel	0.05 €	0.20 €
18	4	M5x30	Schraube mit Innensechskant für Gehäusedeckel	0.10 €	0.40 €

Summe: 131.98 €

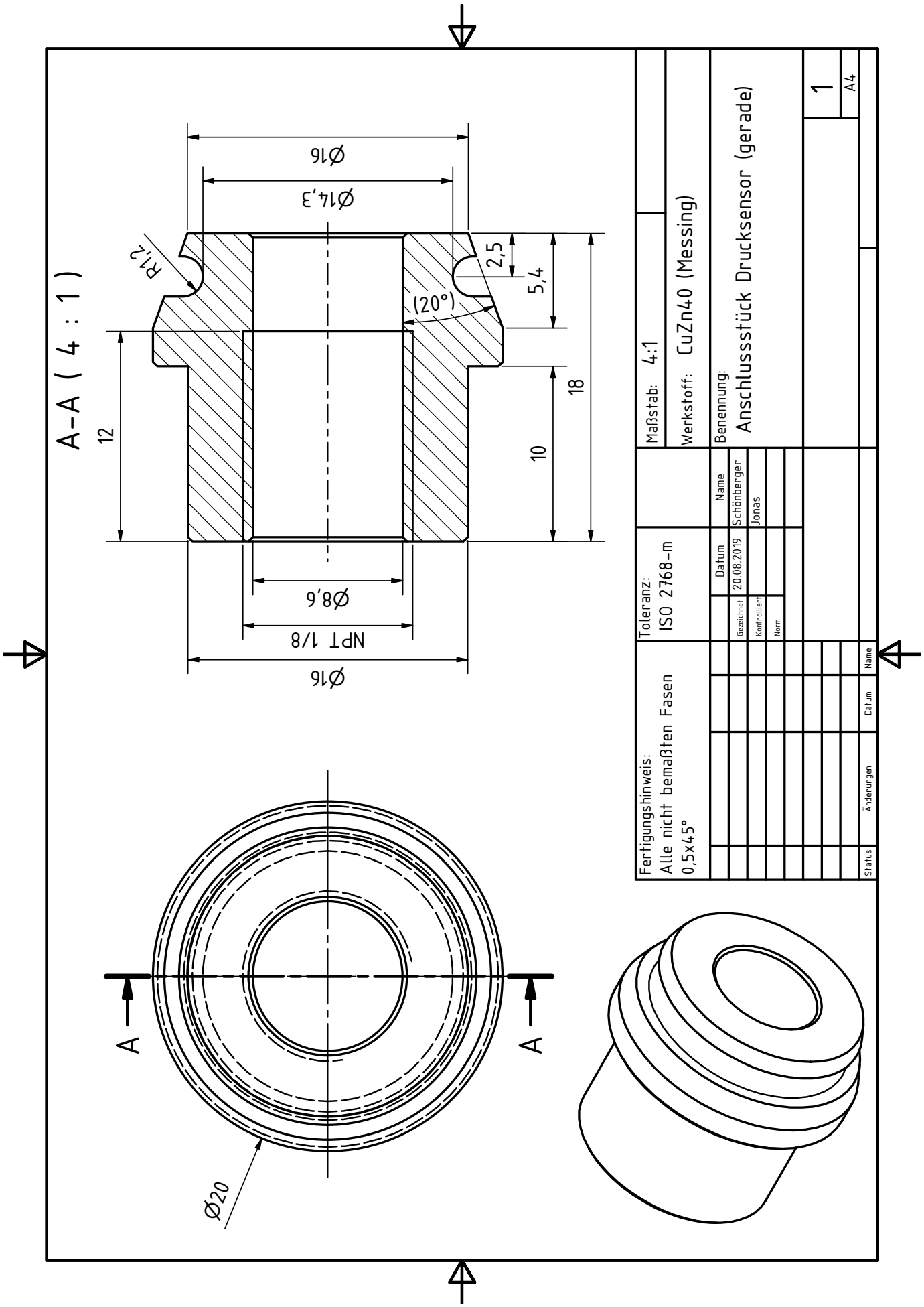
F.2 Stückliste Platine

Pos	Stück	Bezeichnung / Wert	Beschreibung	Preis	Gesamt
1	1	PCB 100x100mm 2 layers	Herstellung Platine	0.49 €	0.49 €
2	1	ESP32-WROOM-32D	Espressif Mikrocontroller	3.00 €	3.00 €
3	2	TPS562200	Spannungsregler	0.15 €	0.30 €
4	1	MAX485E	RS485 Interface	0.08 €	0.08 €
5	6	Si2302	N-Channel Mosfet (SMT)	0.02 €	0.12 €
6	2	IRF7455	N-Channel Mosfet (SMT)	0.30 €	0.60 €
7	1	DS18B20	Temperatursensor	0.40 €	0.40 €
8	12	3V6	Zenerdiode (SMT)	0.03 €	0.36 €
9	9	BAS32L 0,45 A - 75 V	Diode (SMT)	0.015 €	0.135 €
10	4	1N4148 2 A – 100V	Diode (THT)	0.009 €	0.036 €
11	1	1nF	Folienkondensator (SMT)	0.009 €	0.009 €
12	16	100nF	Folienkondensator (SMT)	0.009 €	0.144 €
13	1	1µF	Elektrolytkondensator (SMT)	0.013 €	0.013 €
14	4	10µF	Elektrolytkondensator (SMT)	0.013 €	0.052 €
15	4	22µF	Elektrolytkondensator (SMT)	0.015 €	0.06 €
16	1	2200µF 16V	Elektrolytkondensator (THT)	0.10 €	0.10 €
17	1	CD74R 4.7µH	Induktivität	0.10 €	0.10 €
18	2	2Ω	Widerstand (THT)	0.005 €	0.01 €
19	2	10kΩ	Widerstand (THT)	0.005 €	0.01 €
20	2	33Ω	Widerstand (THT)	0.005 €	0.01 €
21	1	33,2kΩ	Widerstand (THT)	0.005 €	0.005 €
22	1	54,9kΩ	Widerstand (THT)	0.005 €	0.005 €
23	2	120Ω	Widerstand (SMT)	0.006 €	0.012 €
24	4	200Ω	Widerstand (SMT)	0.006 €	0.024 €
25	2	500Ω	Widerstand (SMT)	0.006 €	0.012 €
26	2	720Ω	Widerstand (SMT)	0.006 €	0.012 €
27	10	1kΩ	Widerstand (SMT)	0.006 €	0.06 €
28	2	1,5kΩ	Widerstand (SMT)	0.006 €	0.012 €
29	6	2,2kΩ	Widerstand (SMT)	0.006 €	0.036 €
30	31	10kΩ	Widerstand (SMT)	0.006 €	0.186 €
31	6	22kΩ	Widerstand (SMT)	0.006 €	0.036 €
32	1	24kΩ	Widerstand (SMT)	0.006 €	0.006 €
33	3	100kΩ	Widerstand (SMT)	0.006 €	0.018 €
34	2	6x6x4,3mm	Mikrotaster	0.01 €	0.02 €
35	2	1x2	Jumper	0.03 €	0.06 €
36	4	1x1	Stifleiste	0.04 €	0.16 €
37	5	1x3	Stifleiste	0.03 €	0.15 €
38	2	Orange 620nm 2,7V	LED (SMT)	0.01 €	0.02 €
39	1	10mm 5V	Buzzer	0.09 €	0.09 €
40	1	LEG-12	Relais	0.30 €	0.30 €
41	1	5x20mm	Sicherung	0.03 €	0.03 €
42	3	2x3 6-polig	Wannenstecker	0.12 €	0.36 €
43	1	JST_PH_B5B-PH-K	JST Stecker 5x2,00mm	0.02 €	0.02 €
44	1	JST_XH_B5B-XH-A	JST Stecker 5x2,50mm	0.02 €	0.02 €
45	6	2x5mm XY301V	Schraubklemme	0.15 €	0.90 €
46	1	3x5mm XY301V	Schraubklemme	0.20 €	0.20 €
47	10	2x3,5mm XY350	Schraubklemme	0.15 €	1.50 €
48	4	3x3,5mm XY350	Schraubklemme	0.20 €	0.80 €

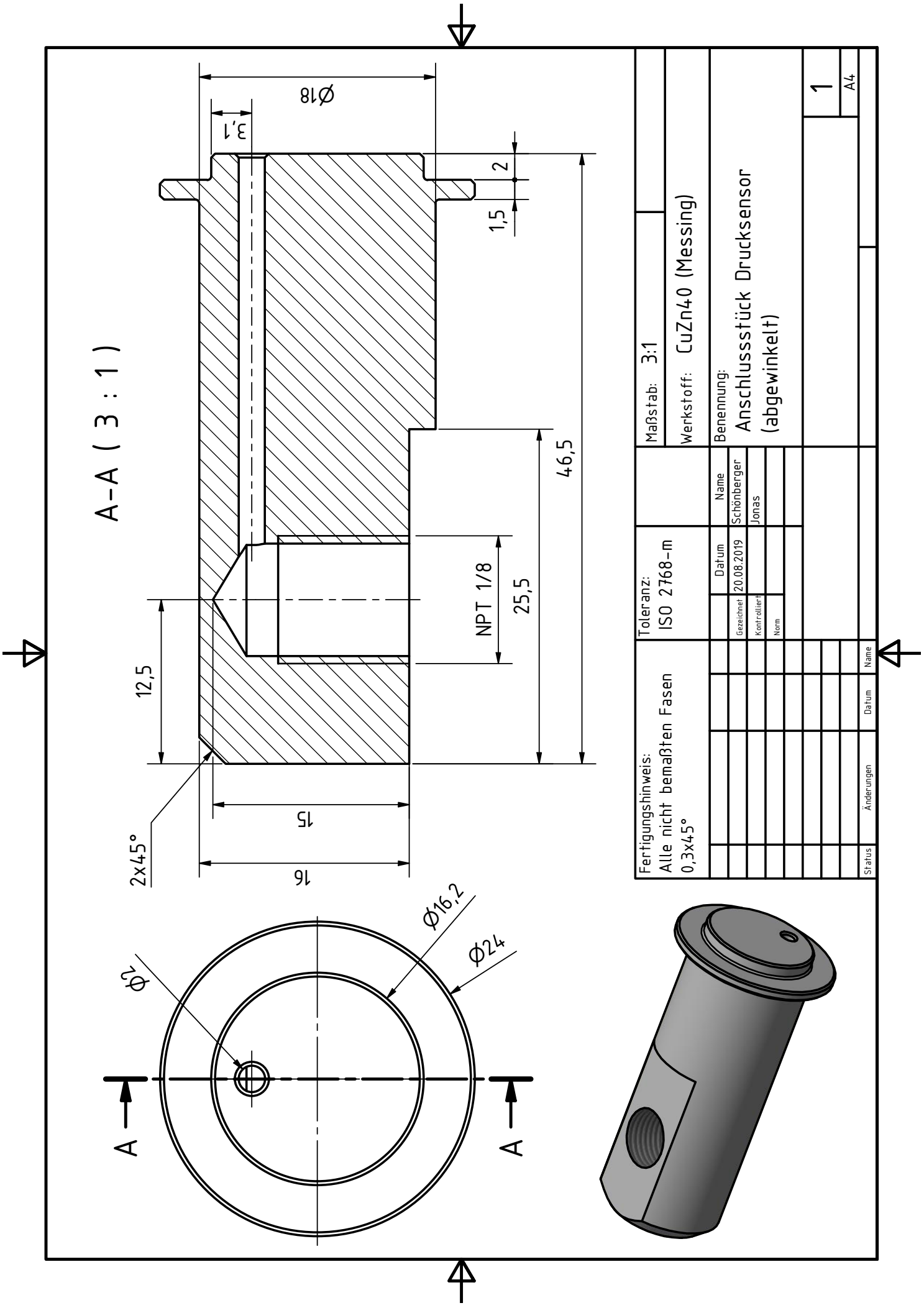
Summe: 11.08 €

G Technische Zeichnungen

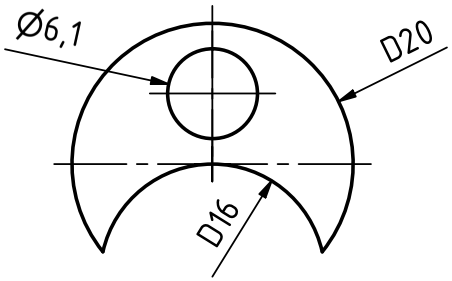
G.1 Drucksensoradapter gerade



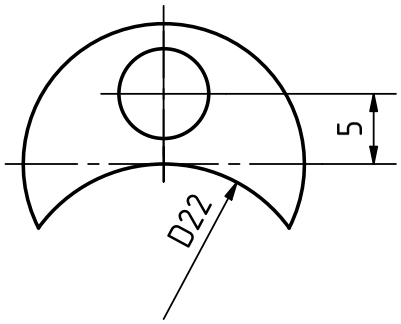
G.2 Drucksensoradapter abgewinkelt



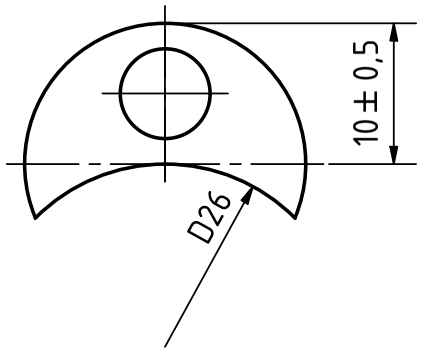
Brauchwasser Umlauf



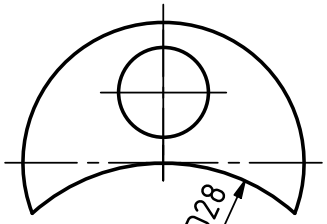
Heizkreis2 Fußboden

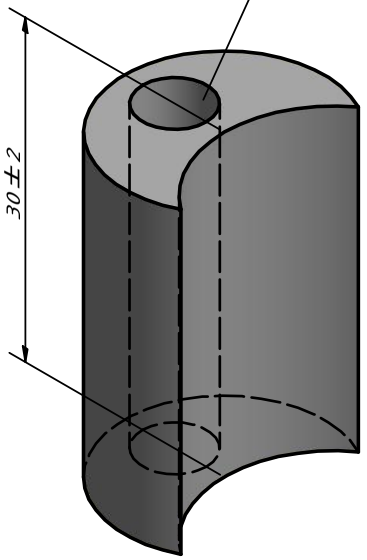


Brauchwasser Ablauf, Zulauf

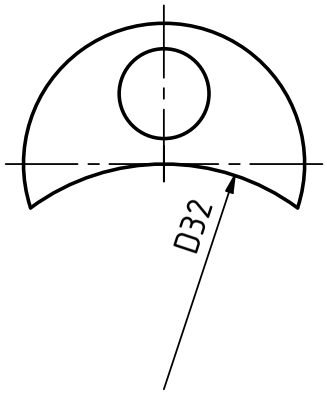


Solar, Heizkreis1

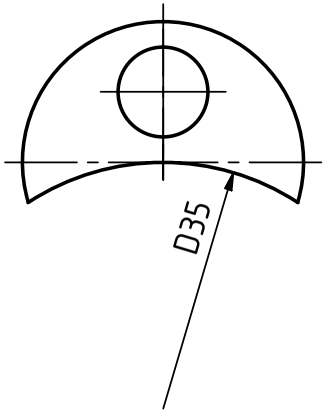




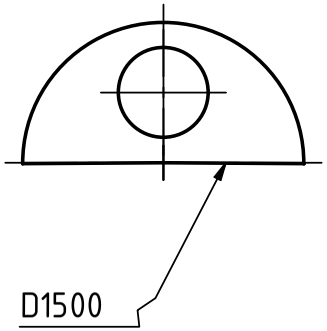
Brauchwasser Zulauf



Ladekreis



Pufferspeicher



Stück	Typ
3	Ø16
4	Ø22
2	Ø26
4	Ø28
1	Ø32
2	Ø35
3	Ø1500

Fertigungshinweis:
Bohrung leicht ansenken
(0,2x45°)

Toleranz:
ISO 2768-m

	Datum	Name
Gezeichnet	20.08.2019	Schönberger
Kontrolliert		Jonas
Norm		

Maßstab: 1:1

Werkstoff: Al (Aluminium)

Benennung:
Übersicht Temperatursensorhalter

Status	Änderungen	Datum	Name

1

A4

H Tätigkeitsnachweis

Tag	Ausgeführte Arbeiten, Tätigkeiten, Unterweisungen, Projektbesprechungen	Stunden
1	- Projektarbeit mit Herrn Leipold besprochen	1
	- Heizungsanlage inspiziert, Skizzen erstellt	2
	- Systemschema erstellt	4
2	- Konzept des Systemaufbaus erarbeitet	2,5
	- Nach geeigneten Komponenten gesucht (z. B. Mikrocontroller oder MQTT) und in diese eingelesen	4
	- Hardware-Komponenten bestellt	2
3	- In das Datenblatt des ESP32 Mikrocontrollers eingelesen um einen Überblick der Funktionen zu erhalten - Eine Toolchain zum Programmieren des ESP32 aufgesetzt	4
	- Versuchsaufbau mit ESP32 Entwicklerboard realisiert	1
	- Erste Funtionen testweise programmiert (WLAN, MQTT) und diese live am Entwicklerboard getestet	3
4	- Alle verschiedenen Rohrdurchmesser der Heizungsanlage gemessen	0,5
	- Temperatursensorhalter konstruiert	2
	- Prototypen von verschiedenen Versionen mittels 3D-Drucker angefertigt und an den Rohren getestet	2
	- Ausgewählte Temperatursensorhalter in benötigter Anzahl gefertigt	3
5	- Sensoren Namen und Adresse zugeordnet - Sensoren mit den Sensorhaltern an der Anlage montiert	3
	- Sensoren an das ESP32 Entwickerboard angeschlossen	0,5
	- Programmierung Platine: Funktionen zum Auslesen der One-Wire Temperatursensoren	2,5
	- Programmierung Platine: Funktion zum Übermitteln der Daten über MQTT	2,5
	Wochenstunden:	39,5

Tag	Ausgeführte Arbeiten, Tätigkeiten, Unterweisungen, Projektbesprechungen	Stunden
6	- Server Software installiert und konfiguriert: MQTT-Broker und Node-Red	1
	- Einlesen und Einarbeiten in die Programmierumgebung Node-Red	3
	- Programmierung Node-Red: Empfangen der Sensordaten und Anzeige auf einer Website	4
7	- Anforderungen an die Drucksensoren und relevante Maße vor Ort festgelegt	0,5
	- Drucksensorhalter bzw. Adapter geplant und konstruiert	3
	- Drucksensorhalter gefertigt und montiert	4
8	- Programmierung Platine: Auswertung Analogeingang	5
	- Serversoftware installert und konfiguriert: MYSQL-Datenbank	1
	- Programmierung Node-Red: Speichern der ankommenden Daten in der Datenbank	3
9	- Signal der Solarpumpe und Stromsensoren mit Oszilloskop analysiert	1
	- Versuchsaufbau einer Schaltung zum Linearisieren des PWM-Signals	1,5
	- Programmierung Platine: Auswertung Digitaleingang	4
10	- Anforderungen an die zu entwickelnde Mehrzweckplatine festgelegt	6
	- Die wichtigsten Bauteile ausgewählt und in deren Datenblätter eingelesen	3
	Wochenstunden:	40

Tag	Ausgeführte Arbeiten, Tätigkeiten, Unterweisungen, Projektbesprechungen	Stunden
11	- Schaltplan der Platine erstellt	7
	- Einzelne Schaltungen aufgebaut und getestet	1
12	- Schaltplan der Platine erstellt	7,5
13	- Schaltplan der Platine erstellt	2
	- Layout der Platine erstellt (Routen)	6

14	- Layout der Platine erstellt (Routen)	8
15	- Platine validiert und bestellt	2,5
	- Gehäuse für Platine konstruiert	2
	- Gehäuse mit 3D-Drucker angefertigt, optimiert und montiert	3
	Wochenstunden:	39

Tag	Ausgeführte Arbeiten, Tätigkeiten, Unterweisungen, Projektbesprechungen	Stunden
16	- Platinen bestückt	3
	- Benötigte Sensorleitungen im Heizraum verlegt	2
	- Platine montiert und verdrahtet	3
17	- Stromsensoren montiert und verdrahtet	2
	- Programmierung Platine: Optimierungen, Konfigurationsdatei mit Sensor-definitionen implementiert	6
18	- Programmierung Node-Red: Funktionen zum Erzeugen von Charts	7
	- Programmierung Node-Red: Benutzeroberfläche (Website) überarbeitet	1
19	- Programmierung Node-Red: E-Mail Benachrichtigungen realisiert	5
	- Programmierung Platine: Möglichkeit zum Durchführen von Steueraufgaben implementiert	3
20	- Sämtliche geforderten Funktionen getestet, kleine Verbesserungen durchgeführt	3
	- Herrn Leipold das fertige Projekt vorgestellt	2
	Wochenstunden:	37

Die Anzahl der insgesamt abgeleisteten Stunden beträgt somit 120 Stunden.

Mitterteich, 12.9.19
Ort, Datum

Schönberger Jonas
Jonas Schönberger

ELEKTRO LEIPOLD
GmbH & Co. KG
Hüblteichstr. 2
95666 Mitterteich
Tel. 0 96 33 / 9 12 30
Fax 0 96 33 / 9 12 31
Betreuer Betrieb, Firmenstempel

I Selbstständigkeitserklärung

Hiermit erkläre ich, Jonas Schönberger, dass ich die vorliegende Dokumentation zur Projektarbeit selbstständig und ohne fremde Hilfe verfasst, sowie alle verwendeten Quellen und Zitate durch Angabe der Herkunft kenntlich gemacht habe.

_____ _____

Ort, Datum Jonas Schönberger