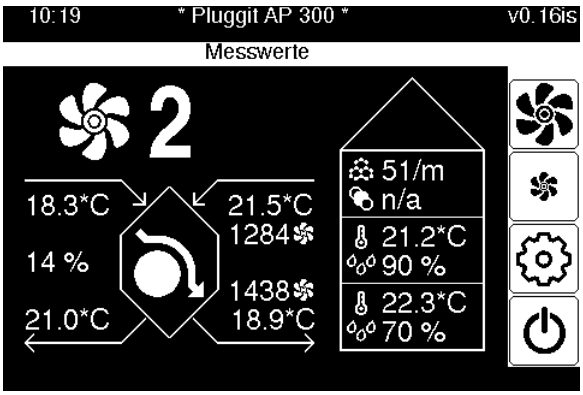
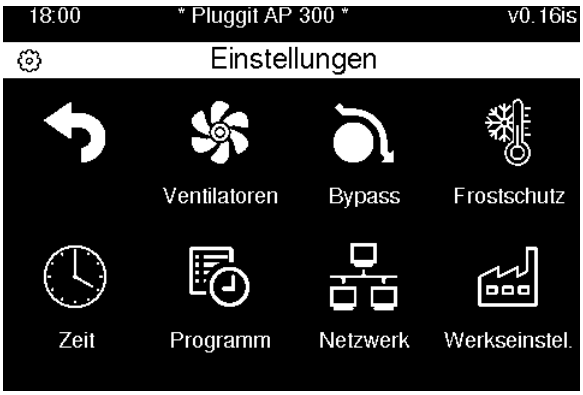
Ersatzsteuerung für eine Lüftungsanlage für Wohnhäuser mit Touch-Display und mqtt Anbindung





Inhaltsverzeichnis

1 Sicherheitshinweis....5
2 Einführung....5
2.1 Ziel....5
2.2 Systemaufbau....6
2.3 Regelungsstrategie der Steuerung....6
2.4 Speicherung des Anlagenparameter....7
3 Hardware....7
3.1 Hardwareauswahl....7
3.1.1 Notwendige Hardware....7
3.1.1.1 Platine für Verbindung der Sensoren und Lüfter mit dem Arduino....8
3.1.1.2 Spannungsversorgung....8
3.1.2 Alternative und optionale Hardware....8
3.1.2.1 Alternativ: DAC 0-10V für Vorheizregister und Lüfter....8
3.1.2.2 Optional: TouchDisplay....8
3.1.2.3 Optional: Platine für abgesetztes TouchDisplay....8
3.1.2.4 Optional: Sommer-Bypass per Relais....9
3.1.2.5 Optional: CO2 Sensor....9
3.1.2.6 Optional: VOC Sensor....9
3.1.2.7 Optional: Luftfeuchtesensoren DHT22....9
3.2 Schaltpläne....10
3.2.1 Interface Platine für Anschluss der Sensoren....10
3.2.2 HiVolt Platine für Stromversorgung und 230V Anschluss der Lüfter....11
3.2.3 Interface Platine für Display....13
3.3 Aufbau der Platinen....15
3.3.1 Bestückung der Interface Platine....15
3.3.2 Bestückung der HiVolt Platine....15
3.3.3 Bestückung der Interface Platine zum Display....15
4 Grundeinstellungen Soucecode....16
4.1 Definition der Anschlüsse....16
4.2 Netzwerkeinstellungen....16

4.3 Werkseinstellung....16
4.4 Ansteuerung der Relais....18
4.5 Debugausgaben....18
4.5.1 Debugausgaben auf der seriellen Schnittstelle....18
4.5.2 Debugausgaben per mqtt....18
5 Build der Software für Arduino....19
5.1 Notwendige Bibliotheken....19
6 Inbetriebnahme....19
6.1 Displaykalibrierung....20
6.2 Feuerstättenmodus (Kaminbetrieb)....20
6.3 Optional, aber empfohlen: MQTT Broker....21
6.4 Temperatursensoren....21
6.5 Lüfter....22
6.5.1 Überprüfung des Tachosignals vom Lüfters....22
6.5.2 Überprüfung der Anzahl der Tachosignale pro Lüfterumdrehung....24
6.5.3 Einstellen der Lüfterstufen....24
6.5.4 Kalibrierung der Lüfteransteuerung....25
6.6 Optional: Sommer-Bypass....25
6.7 Optional: Vorheizregister....26
7 Einstellungen im Betrieb....26
7.1 Setzen der Lüftungsstufe....26
7.2 Steuerung der Bypass-Klappe....26
8 Bedienung mit Touch-Display....27
9 Einbindung dieser Steuerung in fhem....27
10 Bauteillisten....27
10.1 Arduino, LAN-Modul, Temperatursensoren, 4-fach Relais....27
10.2 Interface Board P300....28
10.3 Display Board....30
10.4 Spannungsversorgung und 230V Lüfteranschluss....30
11 Bilder der verwendeten Hardware....31
11.1 Arduino Mega 2560REV3....31
11.2 LAN Shield mit W5100....31

11.3 Relais Shield....32
11.4 DAC von Horter....32
11.5 Temperatursensoren DS18B20....32
11.6 Touch-Display 3,5“, mcufriend-Library, Auflösung 480x320....32

Abbildungsverzeichnis

Abbildung 1: Schaltplan Interface Sensoren – Arduino....10
Abbildung 2: Bestückte Interface Platine Sensoren Arduino mit Molexsteckverbindungen....11
Abbildung 3: HiVolt Schaltplan mit Schaltnetzteil(en) und Pluggit Lüfteranschluss....11
Abbildung 4: Bestückte HiVolt Platine....12
Abbildung 5: Schaltplan Display Platine....14
Abbildung 6: Display Platine zum Aufstecken des Displays....14
Abbildung 7: Kalibrieren des Touchdisplay....20
Abbildung 8: Kalibrierung Touch wird gestartet....20
Abbildung 9: Fehlerhafte Erkennung des Tachosignals (rpm)....23
Abbildung 10: Korrekte Erkennung des Tachosignals (rpm)....24
Abbildung 11: Arduino Mega 2560 REV Bild von Dsimic - Eigenes Werk, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=35850081....31

1 Sicherheitshinweis

ACHTUNG 230V, LEBENSGEFAHR

EIGENES RISIKO BEIM NACHBAU

Die Lüfter, der Bypass und das Vorheizregister werden mit 230V Versorgungsspannung betrieben bzw. mit 230V Spannung (Bypass) gesteuert.

Für den **Betrieb der Lüftungsanlage mit einer Feuerstätte** ohne Fremdluftzufuhr, siehe Abschnitt 6.2

2 Einführung

2.1 Ziel

Ziel dieser Entwicklung ist es, defekte Steuerungen von Lüftungsanlagen für Wohnhäuser durch diese Eigenentwicklung zu ersetzen. Die Entwicklung des Projekt erfolgt in Freizeit als Hobby. Für die Entwicklung werden Standardbauteile verwendet. Dies sind: Arduino Mega, ein LAN-Modul, OneWire Temperatursensoren DS18B20, optional ist die Ansteuerung eines Vorheizregisters, die Ansteuerung für den Sommer-Bypass, Sensoren für CO2, VOC und Luftfeuchtigkeit sowie ein (Touch-) Grafikdisplay für die lokale Anzeige. Die Lüfter können per Optokoppler oder separaten DAC Digital Analog Converter (0-10V) angeschlossen werden. Diese Steuerung implementiert eine MQTT Schnittstelle für die Verbindung zur Haussteuerung.

Die Steuerung wurde ursprünglich für eine Pluggit P300 Anlage entwickelt, funktioniert ebenso für Pluggit P450 und viele weitere Lüftungsanlagen. Die elektrische Anpassung für unterschiedliche Lüftungsanlagen wird im Abschnitt Hardware beschrieben.

Die Steuerung implementiert die wesentlichen Funktionen zum technisch sicheren Betrieb der Lüftungsanlage. Die Frostschutzabschaltung des Zuluftventilators ist immer aktiv, eine Vereisung und damit ggfs Zerstörung des Wärmetauschers sollte damit ausgeschlossen sein. Voraussetzung ist der korrekte Anschluss der Temperatursensoren. Die Autoren dieser Software übernehmen KEINERLEI Garantie. Es werden die folgenden Sensoren/Aktoren geschaltet.

- zwei Lüfter werden angesteuert und deren Tachosignale werden ausgelesen
- vier Temperatursensoren werden abgefragt
- Steuerung des Zuluftventilator zum Frostschutz
- Steuerung des Abluft- und des Zuluftventilators zum Frostschutz bei Kaminbetrieb
- Optional: Vorheizregister als Frostschutz
- Optional: Sommer-Bypass geschaltet
- Optional: Bis zu 2 Temperatur/Luftfeuchtesensoren (DHT22)
- Optional: CO2-Sensor MH-Z14

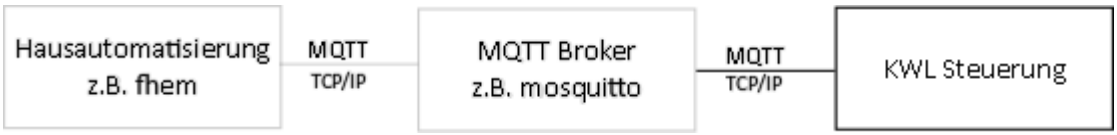
- Optional: VOC-Sensor

Diese Steuerung stellt eine MQTT Schnittstelle zur Verfügung und kann darüber mit Automatisierungssystemen kommunizieren. Alle Topics der MQTT Schnittstelle sind in https://github.com/svenjust/room-ventilation-system/tree/master/Docs/mqtt%20topics dokumentiert.

Komfortfunktionen, wie die Regelung der Lüftungsanlage anhand eines VOC Sensors, sind nicht implementiert. Diese Regelungsaufgaben können in übergeordneten Haussteuerung besser, weil spezifischer für den eigenen Anwendungsfall, implementiert werden. Der Autor verwendet fhem für diese Aufgaben. Die Einbindung dieser Steuerung in fhem wird im Abschnitt 9 beschrieben. Die Messwerte der evtl. verbauten CO2, VOC, oder Feinstaubsensoren in der Lüftungsanlage werden per MQTT gepublishet.

2.2 Systemaufbau

Die Steuerung der Lüftungsanlage kann autark betrieben werden.



Sinnvoll ist ein Betrieb mit einer übergeordneten Haussteuerung. Die Installation von fhem und eines MQTT Brokers werden vorausgesetzt und sind nicht Inhalt dieses Papiers.

2.3 Regelungsstrategie der Steuerung

Der Inbetriebnehmer muss die Normdrehzahl für Zu- und Abluftventilator vorgeben. Differenzdrucksensoren sind nicht verbaut. Hersteller wie Helios verbauen auch keine solchen Sensoren. Pluggit stellt als Normdrehzahl jeweils 1.550 U/min werksseitig ein.

Die Steuerung versucht die vorgegebenen Drehzahlen für Zu- und Abluftventilator zu erreichen. Dem Lüfter wird als Stellgröße ein PWM-Signal oder eine Gleichspannung 0-10V vorgegeben. Über die Tacholeitung teilt der Lüfter über ein PWM Signal seine aktuelle Drehzahl mit. Im Normalbetrieb der Steuerung findet keine Regelung der Drehzahl statt. Die Steuerung kann in den Kalibrierungbetrieb gesetzt werden. Die Lüfter fahren dann die, für die verschiedenen Lüftungsstufen notwendigen Drehzahlen an, und ermitteln per PID-Regler das jeweils notwendige PWM-Signal. Das PWM-Signal für die Drehzahl für jede Lüftungsstufe wird dann getrennt für jeden Lüfter im EEPROM gespeichert. Im Normalbetrieb wird der Lüfter mit dem kalibrierten PWM Signal angesteuert. Die Strategie ist ähnlich dem Betrieb des Originalhersteller, hier findet zu definierten Zeiten (Montagsmorgens) eine „Kalibrierung“ statt. Die Kalibrierung muss per Touch-Display oder per MQTT gestartet werden.

2.4 Speicherung des Anlagenparameter

Die für einen sicheren Betrieb erforderlichen Anlagenparameter werden im EEPROM des Arduino Mega gespeichert. Der Inhalt des EEPROMs kann im Sourcecode und per MQTT-Befehl gelöscht werden. Die Anlage wird damit auf die sogenannte „Werkseinstellungen“ zurückgesetzt. Die Werkseinstellungen definieren für die P300 einen sicheren Betriebszustand.

3 Hardware

Der Hardwareaufbau kann in unterschiedlichen Ausbaustufen erfolgen, ebenso können verschiedene Sensoren ergänzt werden. Zuerst wird die absolut notwendige Hardware besprochen. Weiter unten folgen optionale Teile.

3.1 Hardwareauswahl

3.1.1 Notwendige Hardware

Für die Steuerung ist ein Arduino Mega erforderlich. Für die externe Kommunikation ist ein LAN-Shield erforderlich, aufgebaut und getestet ist das System mit WizNET (W5100) Ethernet Shield, siehe 11.2.

Die Temperaturen werden mit vier Temperatursensoren DS18B20 (siehe 11.5) im Edelstahlgehäuse erfasst und einzeln mit der Steuerung verkabelt. Durch die getrennte Verkabelung können die Sensoren sicher den Luftströmen zugeordnet werden. Für jeden Temperatursensor ist ein 4,7k Widerstand erforderlich.

In den KWL Geräten sind häufig Lüfter mit 230V Anschluss und Steuerung per 10V-PWM-Signal oder per Gleichspannung 0-10V verbaut. Den genauen Lüftertyp muss jeder für seine Anlage bestimmen. In den Pluggit Geräten P300/450 sind Lüfter von Papst verbaut, ähnlich wie http://img.ebmpapst.com/products/manuals/R3G190RC0503-BA-GER.pdf verbaut. In der gerade erwähnten Anleitung, Seite 9 sind die Schaltungsmöglichkeiten des Lüfters aufgeführt. Eine einfache Ansteuerung kann bei dem vorliegenden Lüfter durch Optokoppler erfolgen. Die Optokoppler werden direkt mit dem Lüfter verbunden, siehe Bild 1. Das Tachosignal kann direkt mit dem Arduino verbunden werden.

Hardwareliste (Mindestausstattung):

- 1 x Arduino Mega 2560
- 1 x LAN Module WizNET (W5100) Ethernet Shield
- 4 x OneWire DS18B20 in Edelstahlhülse

Die vollständige Bauteillisten sind im Abschnitt 10 aufgeführt.

3.1.1.1 Platine für Verbindung der Sensoren und Lüfter mit dem Arduino

Für die Verbindung des Arduino Mega mit den Lüftern, Temperatursensoren, Relaisboard und weiteren Sensoren ist in diesem Projekt ein Interface Board entwickelt worden. Der Stecker für die Lüftersteuerung kann direkt von der Pluggitplatine umgesteckt werden

Die technischen Unterlagen zu den Boards und die Gerberfiles für die Erstellung sind bei Github verfügbar https://github.com/svenjust/room-ventilation-system/tree/master/hardware.

3.1.1.2 Spannungsversorgung

Für die Spannungsversorgung gibt es zwei Alternativen. Zum einen ist eine Platine zum Anschluss der 230 V-Lüfter und Netzteil verfügbar. Zum anderen kann ein Standardnetzteil verwendet werden.

Auf der unten vorgestellten Platine für Spannungsversorgung können neben integrierten Step-Down-Netzteilen für 5V und 12V auch die 230V Stecker der Lüfter direkt von der Pluggitplatine umgesteckt werden. Die Platine enthält ist mit einer Littelfuse-Sicherung und einem Varistor als Überspannungsschutz aufgebaut.

3.1.2 Alternative und optionale Hardware

3.1.2.1 Alternativ: DAC 0-10V für Vorheizregister und Lüfter

Alternativ zu den Optokopplern können die Lüfter mit einem 0-10V Gleichstromsignal angesteuert werden. Ein Arduino kann eine solche Gleichspannung nicht direkt zur Verfügung stellen. Ein Digital-Analog-Converter (DAC) wie z.B. https://www.horter-shop.de/de/home/93-bausatz-i2c-analog-input-modul-5-kanal-10-bit-4260404260752.html stellt die erforderliche Spannung bereit und kann einfach vom Arduino angesteuert werden.

Wenn ein Vorheizregister verwendet werden soll, ist die Ansteuerung per DAC evtl. notwendig.

3.1.2.2 Optional: TouchDisplay

Für eine Visualisierung am Gerät bzw. eine Bedienung ohne Automatisierungssystem kann ein Touchdisplay (siehe 11.6) verwendet werden. Die Steuerung setzt auf die mcufriend-Library auf und geht von einer Auflösung von 480x320 Pixel aus. Das Touchdisplay sollte wenigstens 3,5“ groß sein.

3.1.2.3 Optional: Platine für abgesetztes TouchDisplay

Das Display kann entweder direkt auf dem LAN Shield oder per Kabelverbindung entfernt von der Steuerung vor oder auf der Lüftungsanlage betrieben werden. Die direkte Verbindung mit dem LAN Shield kann beim Testen interessant sein. Für die entfernte Anbindung ist eine Platine verfügbar.

3.1.2.4 Optional: Sommer-Bypass per Relais

Wenn das Gerät einen Sommer-Bypass eingebaut hat, kann dieser ansteuert werden. Die Steuerung enthält die dafür notwendigen Funktionen.

GEFAHR: Verschiedene Gerätehersteller bauen bzgl der Ansteuerspannung unterschiedliche Lösungen ein. Hier ist vor Beginn genau zu prüfen, was im eigenen Gerät verbaut.

In Pluggit-Geräten der Serien P300/450 wird der Sommer-Bypass mit 230V~ geschaltet. Die Schaltung entspricht der klassischen Rolladensteuerung, d.h. für die Fahrdauer wird eine Spannung angelegt und nach Ausschalten der Spannung bleibt der Bypass in der aktuellen Stellung stehen.

Helios schaltet den Sommer-Bypass mit 24V=.

Die Schaltung für den Sommer-Bypass entspricht klassischen Rolladensteuerung, es werden zwei Relais verbaut, das erste Relais schaltet die Spannung, das zweite (Wechsel-) Relais bestimmt die Fahrrichtung.

3.1.2.5 Optional: CO2 Sensor

Mit dem folgenden CO2 Sensor hat der Autor gute Erfahrungen gemacht: MH-Z14, https://www.sensor-test.de//ausstellerbereich/upload/mnpdf/en/MHZ14_14.pdf . Das Signal wird über eine serielle Schnittstelle übertragen.

Die Messwerte des CO2 Sensors werden per MQTT an übergeordnete Geräte übertragen. Eine Regelung der Lüftungsanlage anhand des CO2-Sensors, ist nicht implementiert. Diese Regelungsaufgaben kann in einer übergeordneten Haussteuerung besser, weil spezifischer für den eigenen Anwendungsfall, implementiert werden.

Der CO2 Sensor kann im Abluftstrom der Lüftungsanlage verbaut werden.

3.1.2.6 Optional: VOC Sensor

Der Autor verwendet den folgenden VOC Sensor Figaro TGS2600, http://www.figarosensor.com/products/2600pdf.pdf . Das Signal wird durch einen Widerstandswert übertragen. Für das Steuern der Lüftungsanlage ist der VOC Sensor nach Erfahrung des Autors ausreichend, der CO2-Sensor dient eher dem Spiel- und Messtrieb. Beide Werte korrelieren weitgehend.

Die Messwerte des VOC Sensors werden per MQTT an übergeordnete Geräte übertragen. Eine Regelung der Lüftungsanlage anhand des VOC-Sensors, ist nicht implementiert. Diese Regelungsaufgaben kann in einer übergeordneten Haussteuerung besser, weil spezifischer für den eigenen Anwendungsfall, implementiert werden.

Der VOC Sensor kann im Abluftstrom der Lüftungsanlage verbaut werden.

3.1.2.7 Optional: Luftfeuchtesensoren DHT22

Für die Messung der Luftfeuchte im Abluft- und Zuluftstrom können bis zu zwei DHT22 Sensoren eingebaut werden. Die Messwerte der Sensoren werden per MQTT an übergeordnete Geräte

übertragen. Eine Regelung der Lüftungsanlage anhand der DHT22-Sensoren, ist nicht implementiert.

3.2 Schaltpläne

3.2.1 Interface Platine für Anschluss der Sensoren

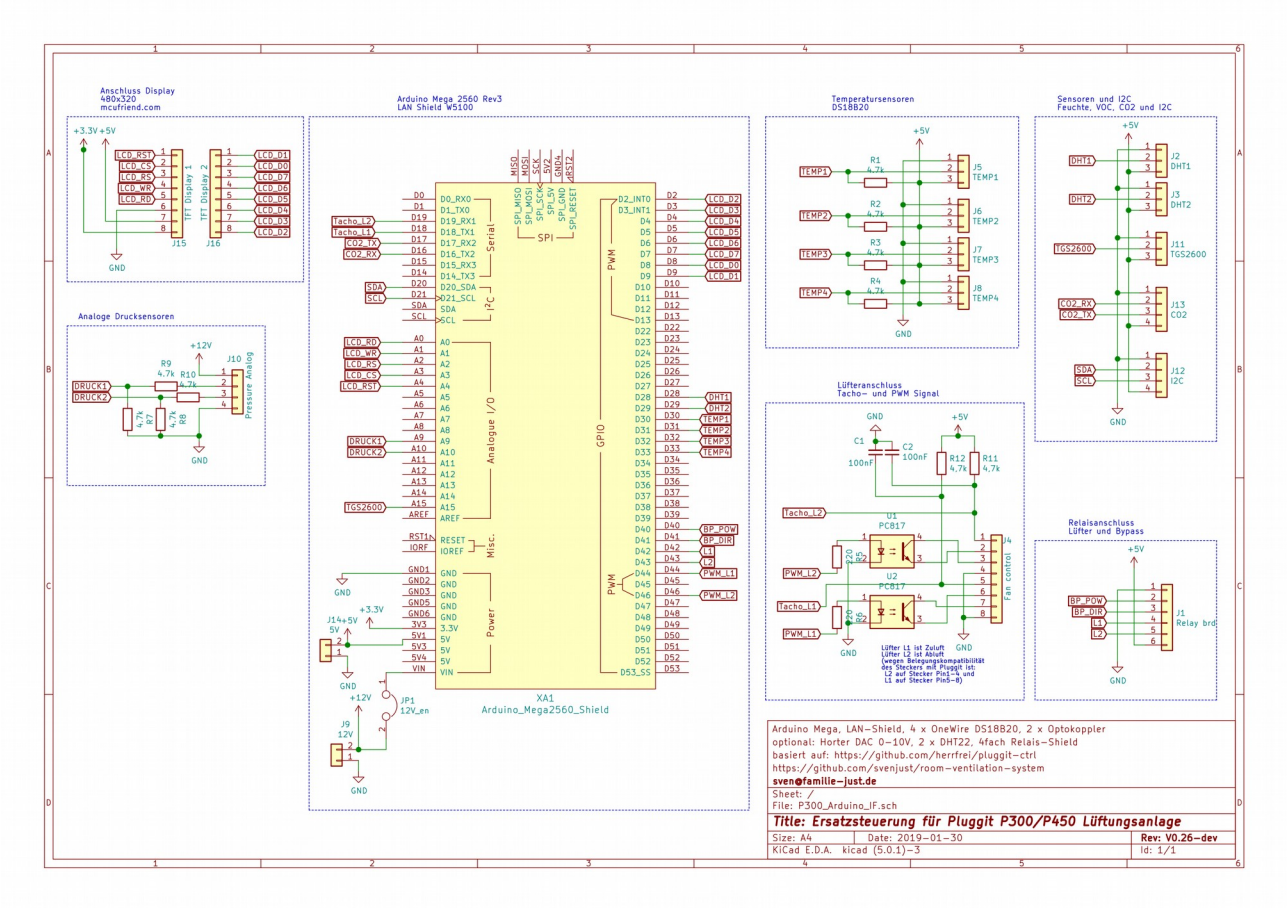


Abbildung 1: Schaltplan Interface Sensoren – Arduino

Für die Verbindung zu den Lüftern wird ein achtpoliger Molex-Stecker mit der Bezeichnung:Molex_KK-254_AE-6410-08A_1x08_P2.54mm_Vertical empfohlen. Damit kann der Stecker der Pluggit-Platine direkt umgesteckt werden, das Pinlayout ist identisch.

Für die anderen 2-, 3-, 4-, 6- und 8-poligen Steckverbindern können Stecker mit 2,54mm Pinabstand verwendet werden. Dies können Steckverbinder aus der Molex-KK-254-Reihe oder auch JST XH-Reihe sein. Die JST XH-Reihe ist aktuell leichter und zu geringeren Kosten verfügbar.

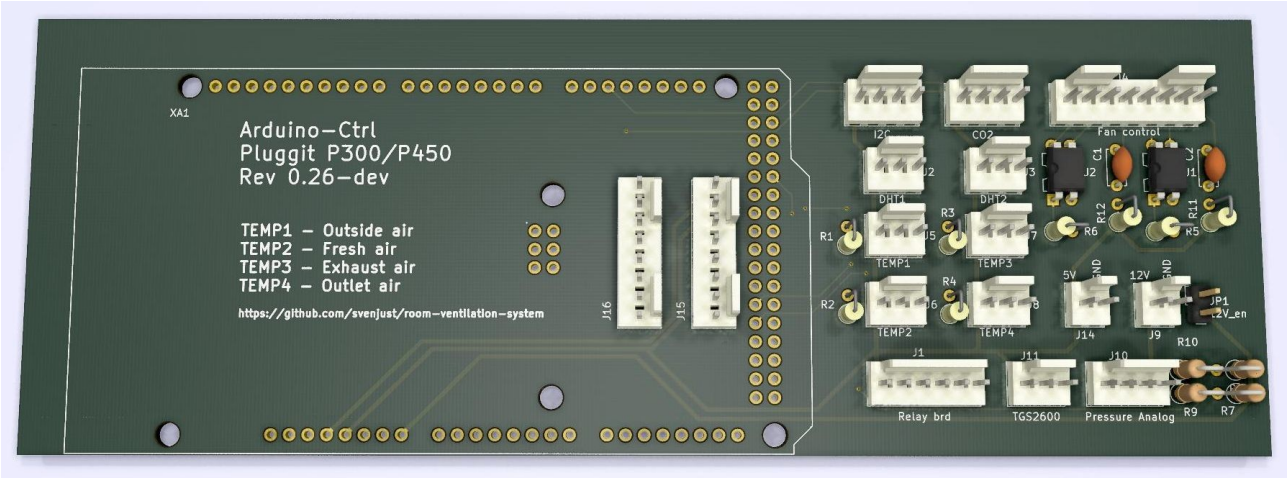


Abbildung 2: Bestückte Interface Platine Sensoren Arduino mit Molexsteckverbindungen

3.2.2 HiVolt Platine für Stromversorgung und 230V Anschluss der Lüfter

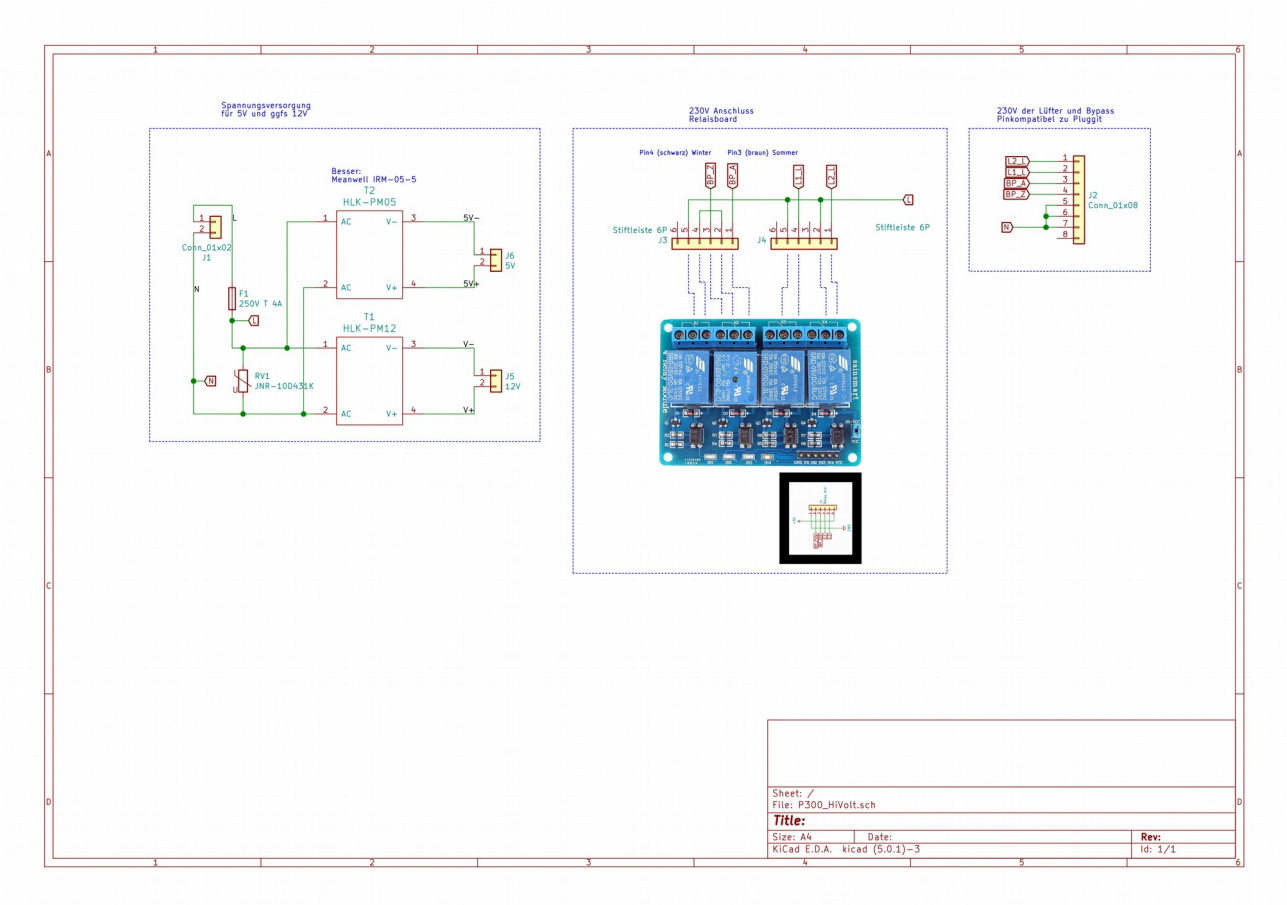


Abbildung 3: HiVolt Schaltplan mit Schaltnetzteil(en) und Pluggit Lüfteranschluss

Für das 5V Schaltnetzteil wird ein Meanwell IRM-05-5 empfohlen, dieses leistet bis zu 1A. Das HLK-PM5 kann beim Betrieb ohne Display verwendet werden. Die 12V Spannungsversorgung ist optional.

Für die 230V Verbindung zu den Lüftern wird der Connector Molex_Mini-Fit_Jr_5566-08A_2x04_P4.20mm_Vertical verwendet, dieser ist pinkompatible zu den Pluggit-Steckern, der Stecker kann umgesteckt werden.

Für den 230V Eingangsstecker wird Molex_Mini-Fit_Jr_5566-02A_2x01_P4.20mm_Vertical empfohlen, wiederum pinkompatible zu Pluggit.

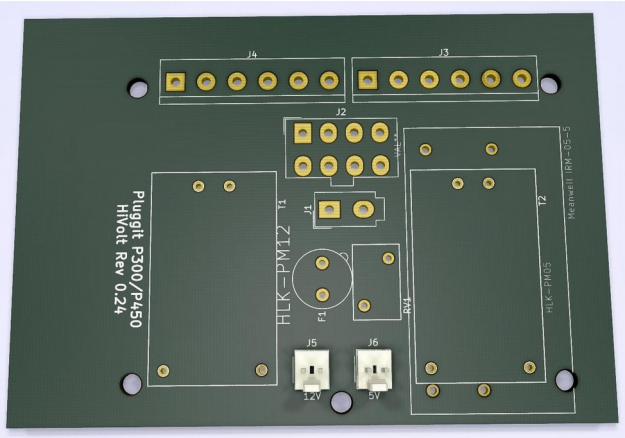


Abbildung 4: Bestückte HiVolt Platine

3.2.3 Interface Platine für Display

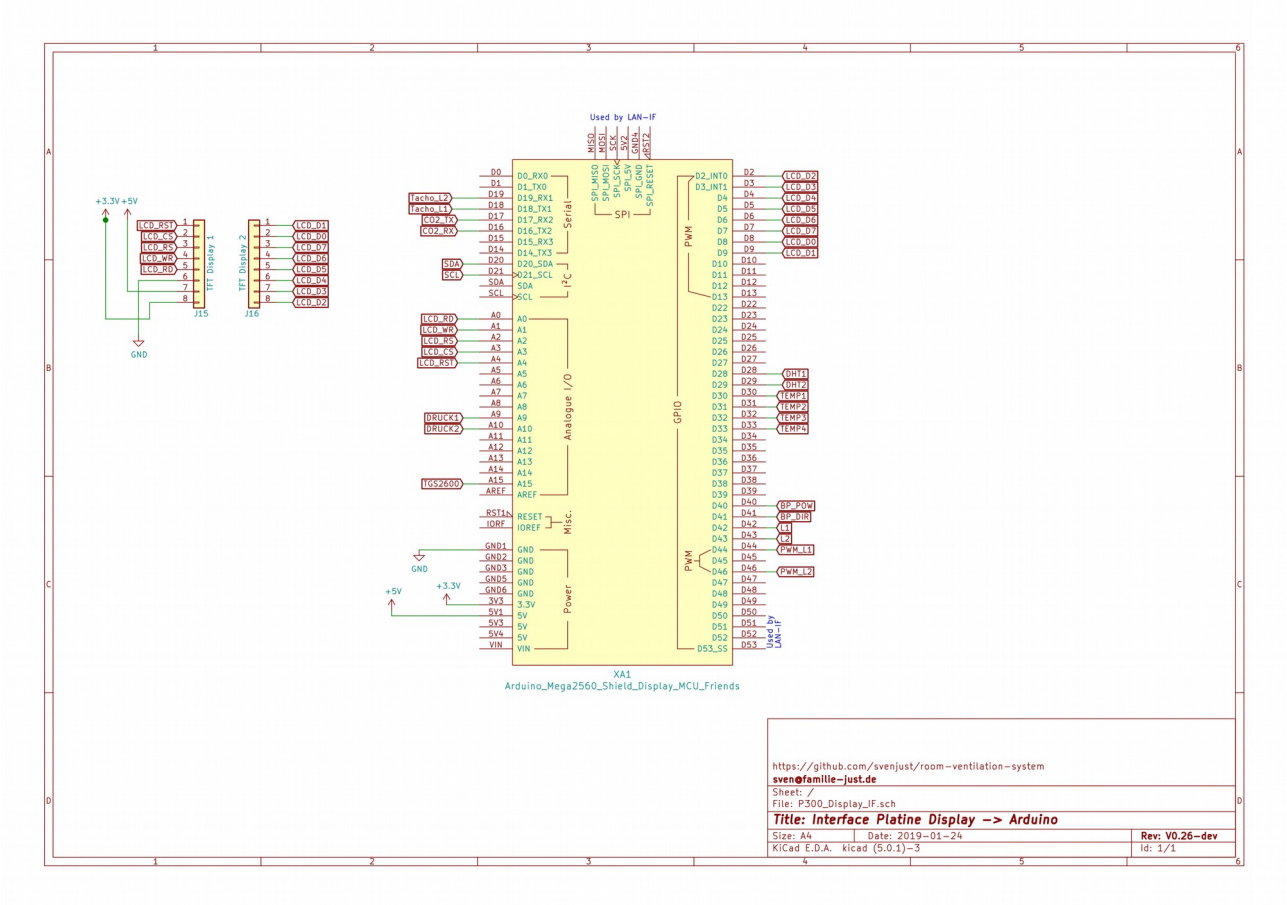


Abbildung 5: Schaltplan Display Platine

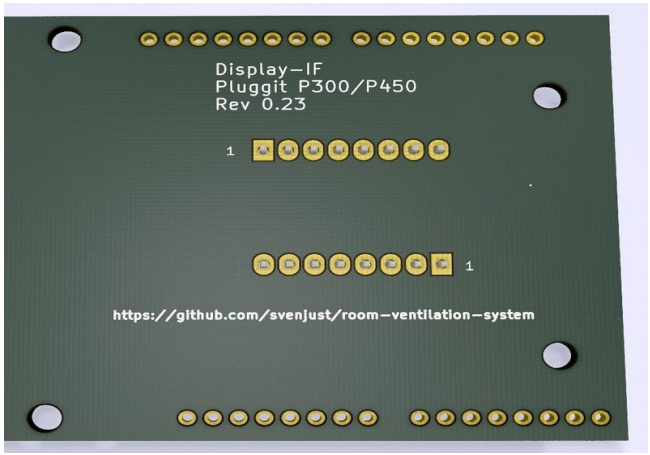


Abbildung 6: Display Platine zum Aufstecken des Displays

Die Interface Platine zum Display ermöglicht lediglich die einfache Befestigung und Verwendung von Steckern zwischen Display und Controller. Die Platine enthält keine weiteren Bauelemente.

3.3 Aufbau der Platinen

Es wird ein fachkundigen Aufbau der Schaltung vorausgesetzt. Dabei ist zwischen der Niederspannungsseite und der 230V-Seite zu trennen.

Für den Aufbau der Schaltung existieren fertige Platinen. Die Verbindungen der Bauteile sind zu löten, bzw. zu schrauben. Nach dem Löten sind die Lötverbindungen zu kontrollieren.

Der Arduino Mega, die Interface Platine und das LAN Shield werden in der Reihenfolge von unten nach oben aufeinander gesteckt. Bei einer Verschraubung der Platinen sind Kunststoffschrauben zu verwenden.

Für die Befestigung der Platinen in der Lüftungsanlage kann als Träger eine zugeschnittene PVC Platte (Bastelplatte) verwendet werden. Um die Festigkeit zu erhöhen können links und rechts schmale Verstärkungen hochkant aufgeklebt werden. Die PVC Platte sorgt gleichzeitig für eine elektrische Trennung, insbesondere der HiVolt-Platine, gegenüber dem Gehäuse der Lüftungsanlage. Bei vollständiger Bestückung der HiVolt-Platine können die 230V führenden Leiterbahnen damit nicht mehr direkt berührt werden.

3.3.1 Bestückung der Interface Platine

Die Bestückung der Interface Platine sollte mit den Optokopplern starten. Anschließend die Stecker der Kabelverbinder und die Verbinder zum Arduino einlöten. Für die korrekte Ausrichtung der Verbinder zum Arduino ist es sinnvoll, die Stiftleisten in den Arduino Mega zu stecken und anschließend erst die Platine auf die Stiftleisten zu stecken. Die Stiftleisten dann auf der Platine verlöten. Für die Stiftleisten zum LAN-Modul den Platine vom Arduino entfernen und die Stiftleisten mit dem LAN-Modul verbinden, die Platine auflegen und von unten verlöten. Nachfolgend die stehenden Widerstände und ggf. Kondensatoren einlöten.

Zu Abschluss der Bestückung auf Fehler (Lötbrücken, kalte Lötstelle) kontrollieren.

3.3.2 Bestückung der HiVolt Platine

Die Bestückung der HiVolt Platine sollte mit der Fassung für die Sicherung und dem Varistor starten. Nachfolgend die Verbinder und Schaltnetzteile einlöten.

Zu Abschluss der Bestückung auf Fehler (Lötbrücken, kalte Lötstelle) kontrollieren.

3.3.3 Bestückung der Interface Platine zum Display

Auf der Interface Platine müssen die Stiftleisten zum Display verlötet werden. Der Anschluss der Leitungen zur Interface Platine zum Arduino kann abhängig von der gewünschten Bauhöhe unterschiedlich ausgeführt werden, hier ist lediglich auf die korrekte Polung zu achten, Pin 1 ist bei beiden Verbindern gekennzeichnet.

Zu Abschluss der Bestückung auf Fehler (Lötbrücken, kalte Lötstelle) kontrollieren.

4 Grundeinstellungen Soucecode

Die benutzerdefinierten Einstellungen sollen in der Datei UserConfig.h erfolgen. Diese Einstellungen bleiben auch bei einem Update des Sourcecodes erhalten. Die Syntax in der UserConfig.h lautet:

```
CONFIGURE(name, value)
CONFIGURE(name, values…)
```

Weitere Hinweise siehe in der Datei KWLConfig.h

4.1 Definition der Anschlüsse

Abhängig vom tatsächlichen Aufbau der Hardware müssen die Anschlüsse (PINs) für Ein- und Ausgänge kontrolliert bzw. umdefiniert werden, insbesondere werden Displays unterschiedlich angeschlossen. Aktuell verwenden wir Displays für die Anschlüsse A0-A9 und 0-13.

Für die Auswertung des Tachosignals der Lüfter sind Interrupt-fähige Eingänge notwendig, aktuell Pin 18, 19. Das PWM Signal für die Lüfter wird an Pin 44 und Pin 46 bereitgestellt.

Der DAC verwendet die Anschlüsse 20 und 21.

Der CO2-Sensor MH-Z14 verwendet die zweite seriellen Schnittstelle Serial2, Pin 16 und 17.

Die Temperatursensoren DS18B20 verwendet die Pins 30 bis 33.

Die Luftfeuchtesensoren DHT1 und DHT2 verwenden die Pins 28 und 29.

Das Relais für den Bypass und die Lüfter verwendet die Anschlüsse 40 bis 43.

Der VOC Sensor verwendet den Analogeingang A15.

Im Sourcecode beginnt und endet der konfigurierbare Bereich mit „A N S C H L U S S E I N S T E L L U N G E N“

4.2 Netzwerkeinstellungen

Es muss die MAC- und die IP Adresse, Subnet, Gateway, DNS dieser Steuerung einstellt werden, ebenso die IP-Adresse des mqtt-Brokers.

```
CONFIGURE(NetworkIPAddress, 192, 168, 20, 201)
CONFIGURE(NetworkGateway,   192, 168, 20, 250)
CONFIGURE(NetworkDNSServer, 192, 168, 20, 250)
CONFIGURE(NetworkNTPServer, 192, 168, 20, 250)
CONFIGURE(NetworkMACAddress, 0x02, 0xC8, 0x07, 0xC2, 0x35, 0x81 )
```

4.3 Werkseinstellung

Die für einen Betrieb erforderlichen Anlagenparameter werden im EEPROM des Arduino Mega gespeichert.

Durch das Setzen von `#define FACTORY_RESET_EEPROM true` wird bei jedem Neustart der Steuerung der EEPROM-Speicher des Arduinos gelöscht und mit den Werkseinstellungen überschrieben. Für den regulären Betrieb muss `#define FACTORY_RESET_EEPROM false` definiert sein. Das EEPROM des Arduinos ist für 100.000 Schreibzyklen ausgelegt.

Alternativ kann das EEPROM mit dem

```
mosquitto_pub -t d15/set/kwl/resetAll_IKNOWWHATIMDOING -m YES
```

gelöscht werden. Anschließend wird die Steuerung neu gestartet.

Es sind die folgenden „Werkseinstellungen“ definiert. Die können in der Datei „UserConfig.h“ überschrieben werden.

Name	*Wert*	*Beschreibung*
StandardModeCnt	4	Anzahl der unterschiedlichen Lüftungsstufen
StandardKwlModeFactor	{0, 0.7, 1, 1.3}	Definition der Drehzahl je Stufe in Relation zu defStandardSpeedSetpointFan
StandardKwlMode	2	Standardlüftungsstufe beim Einschalten.
StandardSpeedSetpointFan1	1550	Drehzahl in Standardlüftungsstufe für den Zuluftventilator
StandardSpeedSetpointFan2	1550	Drehzahl in Standardlüftungsstufe für den Abluftventilator
StandardFan1ImpulsesPerRotation	1.0	Anzahl der Impulse am Tachosignal pro Umdrehung des Lüfters
StandardFan2ImpulsesPerRotation	1.0	Anzahl der Impulse am Tachosignal pro Umdrehung des Lüfters
StandardKwlFanPrecisionPercent	1.5	Max Abweichung der Istdrehzahl zur Solldrehzahl bei Kalibrierung in Prozent
StandardNenndrehzahlFan	3200	Nenndrehzahl des verbauten Ventilators, siehe Typenschild. Die Nenndrehzahl wird benötigt für die Berechnung der Ansteuerung im unkalibrierten Betrieb
StandardBypassTempAbluftMin	24	Mindestablufttemperatur für die Öffnung des Bypasses im Automatik Betrieb
StandardBypassTempAussenluftMin	13	Mindestaussenlufttemperatur für die Öffnung des Bypasses im Automatik Betrieb
StandardBypassHystereseMinutes	60	Hysteresezeit für eine Umstellung des Bypasses im Automatik Betrieb
StandardBypassHysteresisTemp	3	Hysteresetemperatur für eine Umstellung des Bypasses im Automatik Betrieb
StandardBypassManualSetpoint	1 (Closed)	Stellung der Bypassklappen im manuellen Betrieb
StandardBypassMode	0 (Auto)	Automatik oder manueller Betrieb der Bypassklappe. Im Automatikbetrieb steuert diese Steuerung die Bypass-Klappe, im manuellen Betrieb wird die Bypass-Klappe

		durch mqtt-Kommandos gesteuert.
StandardAntifreezeHystereseTemp	3	Hysteretemperatur für Steuerung von Antifreeze
StandardHeatingAppCombUse	false	Einstellung für den Feuerstättenbetrieb. false = keine Feuerstätte, true = Feuerstätte, Kamin, Ofen ohne Fremdluftzufuhr
StandardDST	false	Daylight savings time default settings.
StandardTimezoneMin	60	Standard time zone offset in minutes (CET, GMT+1).

4.4 Ansteuerung der Relais

Für die Lüfter und den Sommer-Bypass können bis zu vier Relais verbaut sein. Ohne Sommer-Bypass kann die Steuerung auch ohne Relais betrieben werden. Da verschiedene Relais unterschiedlich geschaltet werden, kann hier die logische Schaltung definiert werden.

RELAY_ON	LOW	Daylight savings time default settings.
RELAY_OFF	HIGH	Standard time zone offset in minutes (CET, GMT+1).

4.5 Debugausgaben

4.5.1 Debugausgaben auf der seriellen Schnittstelle

Es können Debugausgaben auf der seriellen Schnittstelle für unterschiedliche Bereiche ein oder ausgeschaltet werden. Nach Änderung muss der Sourcecode neu kompiliert werden.

```
  // ************************************************* D E B U G E I N S T E L L U N G
E N ***************************************************
  // Allgemeine Debugausgaben auf der seriellen Schnittstelle aktiviert.
  static constexpr bool serialDebug = false;
  // Debugausgaben für die Lüfter auf der seriellen Schnittstelle aktiviert.
  static constexpr bool serialDebugFan = false;
  // Debugausgaben für die Antifreezeschaltung auf der seriellen Schnittstelle aktiviert.
  static constexpr bool serialDebugAntifreeze = false;
  // Debugausgaben für die Summerbypassschaltung auf der seriellen Schnittstelle
aktiviert.
  static constexpr bool serialDebugSummerbypass = false;
  // Debugausgaben für die Displayanzeige.
  static constexpr bool serialDebugDisplay = false;
  // Debugausgaben für die Sensoren.
  static constexpr bool serialDebugSensor = false;
  // Debugausgaben für das Programm.
  static constexpr bool serialDebugProgram = false;
  // ***************************************E N D E ***  D E B U G E I N S T E L L U
N G E N ***************************************************
```

4.5.2 Debugausgaben per mqtt

Es können Debugausgabe per mqtt ein- und ausgeschaltet, sowie per mqtt übertragen werden. Dazu ist in der UserConfig.h

```
#define DEBUG
```
zu definieren. Anschließend können die Debugausgaben zur Laufzeit per mqtt ein- und ausgeschaltet werden. Die mqtt Topics sind in der Datei https://github.com/svenjust/room-ventilation-system/Docs/mqtt topics/mqtt_topics.ods beschrieben.

5 Build der Software für Arduino

Die Software wird in der Arduino IDE gebaut. Debugausgaben für die serielle Schnittstelle können im Code aktiviert werden, siehe 4.5.1

5.1 Notwendige Bibliotheken

Für das Bauen der Steuerung sind Bibliotheken notwendig.

- SPI,
- EEPROM,
- PubSubClient,
- Adafruit_GFX_Library,
- DHT_sensor_library,
- PID,
- Adafruit_TouchScreen,
- DallasTemperature,
- Adafruit_Unified_Sensor,
- MCUFRIEND_kbv,
- OneWire,
- Wire,
- Ethernet

Das Projekt bringt eigene Libraries mit. Diese müssen ebenfalls installiert werden, am einfachsten als symbolische Links. Der Skript link_libs.sh linkt die Libraries an die (hoffentlich) richtige Stelle.

Aktuelle Infos sind zu finden unter: https://github.com/svenjust/room-ventilation-system

6 Inbetriebnahme

Vor der Inbetriebnahme sind unbedingt die Pinbelegungen für die Motoren und Sensoren, die Netzwerkeinstellungen und die „Werkseinstellungen“ zu kontrollieren und entsprechend den eigenen Bedürfnissen einzustellen. WICHTIG: Wenn Werkseinstellungen im Sourcecode geändert werden, muss das EEPROM gelöscht werden, damit die neuen Werkeinstellungen verwendet werden, siehe 4.3.

Eine schrittweise Inbetriebnahme wird empfohlen. Die Inbetriebnahme kann über das Touchdisplay oder den MQTT Broker erfolgen, diese Anleitung geht vom funktionierenden MQTT Broker aus.

6.1 Displaykalibrierung

Für die Inbetriebnahme und auch für den späteren Betrieb der Anlage ist ein Display hilfreich. Der Touch vom Display muss nach dem Zusammenbau bzw. bei Änderung der Verbindung zwischen Display und Controller neu kalibriert werden. Dazu 10 Sekunden den Touch berühren, die Display-Kalibrierung wird gestartet.

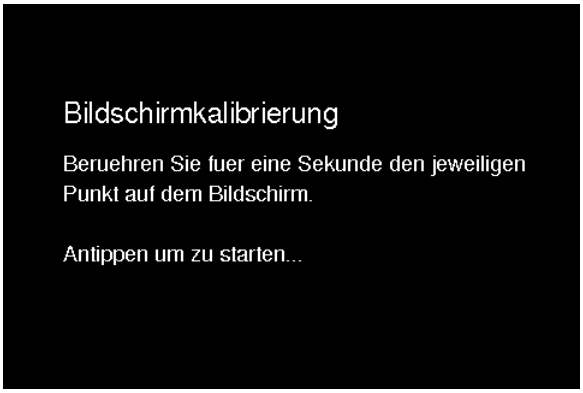


Abbildung 8: Kalibrieren des Touchdisplay

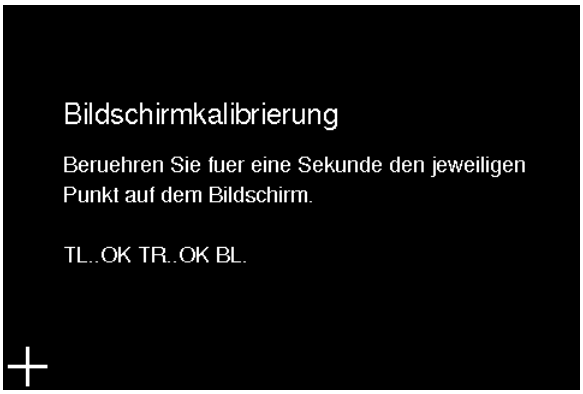


Abbildung 9: Kalibrierung Touch wird gestartet

Anschließend den Anweisung auf dem Display folgen.

6.2 Feuerstättenmodus (Kaminbetrieb)

Wenn eine Feuerstätte (Kamin/Ofen) im Haus ohne Fremdluftzufuhr betrieben wird, besteht die Gefahr, dass bei negativen Außentemperaturen der Zuluftlüfter abgeschaltet wird und giftiges Kohlenmonoxid (CO) aus der Feuerstätte in das Haus gesaugt werden.

- LEBENSGEFAHR -

Mit dem Einstellen des Feuerstättenmodus werden im Antifrostbetrieb beide Lüfter für vier Stunden ausgeschaltet. Das Wiedereinschalten vor vier Stunden ist nur durch einen Neustart der Steuerung möglich.

Der Feuerstättenmodus kann auf zwei Arten eingeschaltet werden, entweder im Sourcecode oder durch Zusenden einer MQTT Nachricht. Bei beiden Arten wird die Einstellung im EEPROM gespeichert und bleibt bei einem Neustart erhalten. Empfohlen wird die **Einstellung im Sourcecode** anzupassen, da diese Einstellung auch bei einer Rücksetzung auf Werkeinstellungen erhalten bleibt.

In der UserConfig muss eingestellt werden:

```
CONFIGURE(StandardHeatingAppCombUse, true)
```

Anschließend ist eine Rückstellung auf Werkseinstellungen notwendig, siehe 4.3.

Alternativ kann folgendes per MQTT an die Steuerung geschickt werden:

```
mosquitto_pub -t d15/set/kwl/heatingapp/combinedUse -m YES
```

Diese Einstellung wird gespeichert, aber bei einer Werkeinstellungen zurückgesetzt!

Beim Starten der Steuerung kann die Einstellung überprüft werden, es wird beim Feuerstättenmodus ausgegeben:

```
...System mit Feuerstaettenbetrieb
```

6.3 Optional, aber empfohlen: MQTT Broker

Für den Betrieb ist ein MQTT Broker sinnvoll. Die Installation eines MQTT Brokers ist an vielen Stellen im Netz beschrieben, u.a. hier: https://wiki.fhem.de/wiki/MQTT_Einf%C3%Bchrung. Wenn der MQTT-Broker installiert ist und die Steuerung gestartet wird, so sendet die Steuerung alle 30 Sekunden ein Heartbeat Signal, diese kann bei einem Mqtt-Client mit

```
mosquitto_sub -v -h localhost -t d15/state/kwl/heartbeat
```

abonniert werden.

6.4 Temperatursensoren

Die vier Temperatursensoren (DS18B20) sind notwendig für einen sicheren Betrieb der Lüftungsanlage. Bei niedrigen Außentemperaturen kann ansonsten der Wärmetauscher durch Frost zerstört werden.

Die Temperatursensoren sind an die Platine anschließen. Im Sourcecode ist die Variable serialDebug = 1 setzten und auf den Arduino flashen. Im seriellen Monitor werden die Temperaturen bei Änderung angezeigt. Für eine Temperaturänderungen können die Sensoren in die Hand genommen werden. Temperaturen von -127.0 zeigen, dass der Sensor nicht gefunden wurde.

Wenn die Temperatursensoren in der Anlage verbaut und verkabelt sind, sollte die korrekte Zuordnung zu den Luftströmen überprüft werden. Bei einer Außentemperatur von 5°C und einer Wohnraumtemperatur von 22 °C hat die Außenluft (T1) die geringste Temperatur, etwa 8 °C, anhängig von den baulichen Gegebenheiten. Die Zuluft (T2) wird bei etwa 18°C liegen. Die Abluft (T3) wird bei etwa 21°C und die Fortluft (T4) bei etwa 10°C liegen (T1 < T4 < T2 < T3). Wird diese Reihenfolge nicht eingehalten, es entweder die Außentemperatur sehr hoch oder die Zuordnung der Temperatursensoren zu den Luftströmen ist nicht korrekt.

6.5 Lüfter

Da die Lüfter üblicherweise mit Netzspannung betrieben werden, ist besondere Vorsicht und Fachkenntnis notwendig.

Die Lüfter sind mit Netzspannung versorgen und an die Platine anschließen. Im Sourcecode die Variable serialDebugFan = 1 setzten und in der UserConfig.h #define DEBUG definieren, kompilieren und den Arduino flashen. Im seriellen Monitor werden die Drehzahlen bei Änderung angezeigt. Hier sollte überprüft werden, dass das Tachosignal der Lüfter korrekt erkannt wird (siehe auch 6.5.1).

Für den erfolgreichen Betrieb der Lüftungsanlage ist die korrekte Drehzahlerkennung die Voraussetzung.

6.5.1 Überprüfung des Tachosignals vom Lüfters

Bei der Inbetriebnahme hat sich die korrekte Erkennung des Tachosignals als nicht immer einfach herausgestellt. Die Firma Pluggit hat während der Bauzeit der Anlage AP300 unterschiedliche Lüfter verbaut. Die Tachosignale der Lüfter haben sich als „unterschiedlich“ gut herausgestellt. Ob die Erkennung korrekt funktioniert, kann mit dem Python-Tool plotfans.py grafisch überprüft werden.

https://github.com/svenjust/room-ventilation-system/Docs/debug_fans/plotfans.py

Hinweise siehe: https://github.com/svenjust/room-ventilation-system/issues/11

Das Skipt plotfans.py stellt die Werte der Soll-, Istdrehzahl und PWM Signal der Lüfter grafisch dar. Das Script benötigt die „mqtt-Logausgabe“ der Lüfter. Um eine Logausgabe für die Lüfter mit Soll-, Istdrehzahl und PWM Signal zu bekommen ist das folgende Vorgehen erforderlich:

Die Software für den Arduino muss mit *#define DEBUG* kompiliert werden. Die Übertragung muss für jeden Lüfter muss einzeln aktiviert werden.

```
   mosquitto_pub -t d15/debugset/kwl/fan1/getvalues -m on
   mosquitto_pub -t d15/debugset/kwl/fan2/getvalues -m on
```

Die übertragenen Werte können mit der folgenden Zeile in einer Datei "/tmp/debug.log" protokolliert werden:

```
mosquitto_sub -v -h localhost -t "d15/debugstate/#" > /tmp/debug.log
```

Die Werte aus der Datei "/tmp/debug.log" können durch das Script plotfans.py dargestellt werden. Im Verzeichnis /tmp werden zwei Dateien mit der grafischen Darstellung der Drehzahlen von Lüfter 1 und 2 angelegt.

```
python <Pfad zu Script>\plotfans.py --infile /tmp/debug.log --out /tmp
```

Der folgende Aufruf zeigt die weiter möglichen Aufrufparameter:

```
python <Pfad zu Script>\plotfans.py -h zeigt die Aufrufparameter
```

Das folgende Bild zeigt im oberen Diagramm einen Lüfter mit **fehlerhafter Tachosignalerkennung**, die Schwankungen der Drehzahl (rpm: Revolution per minute) sind viel zu groß (600 bis etwa 1100 U/min bei gleichem PWM-Signal (ssf).

Im unteren Diagramm wird die Differenz zwischen der Solldrehzahl und der erkannten Istdrehzahl (GAP) dargestellt. Auch hier erkennt man die starken Schwankungen von 500 U/min (+/-250) bei gleichbleibenden PWM-Signal.

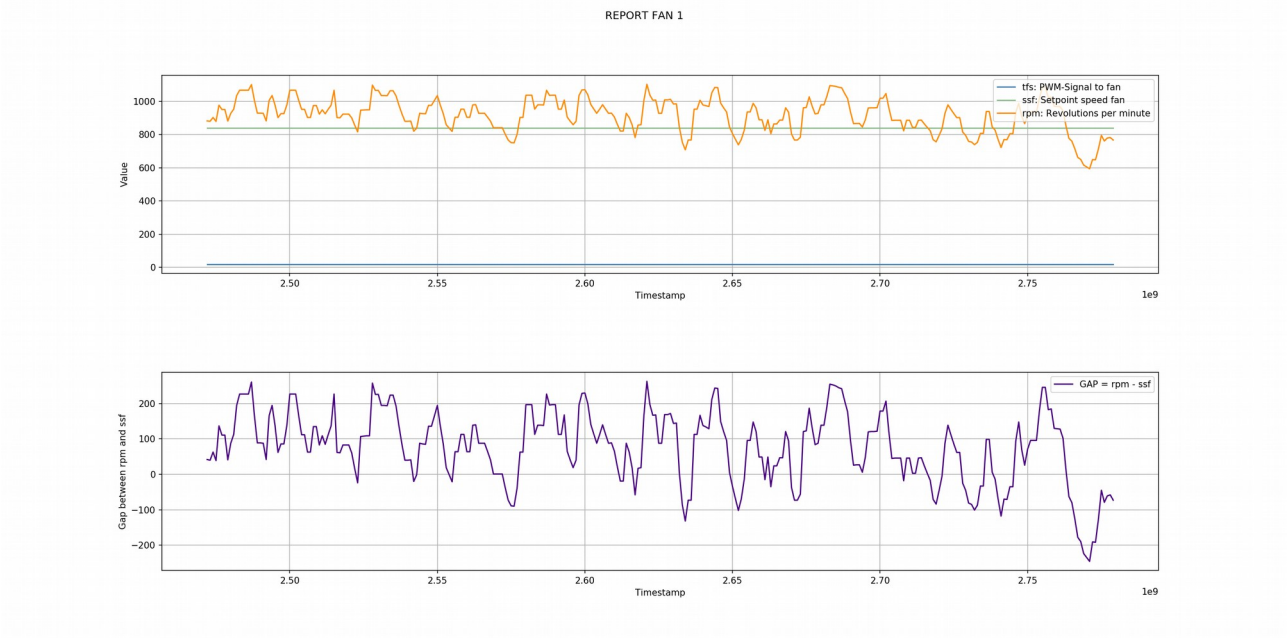


Abbildung 10: Fehlerhafte Erkennung des Tachosignals (rpm)

Eine **korrekte Tachosignalerkennung** zeigt das folgenden Bild. Die Linie *rpm* verläuft stetig mit kleinen Änderungen und im unteren Diagramm (GAP) sind keine „wilden“ Schwankungen zu erkennen, die Sprünge resultieren aus der Änderung der Solldrehzahl, hier während einer Kalibrierungsfahrt. Eine permanente Abweichung zwischen der Soll- und Istdrehzahl kann durch die Lüfterkalibrierung behoben werden.

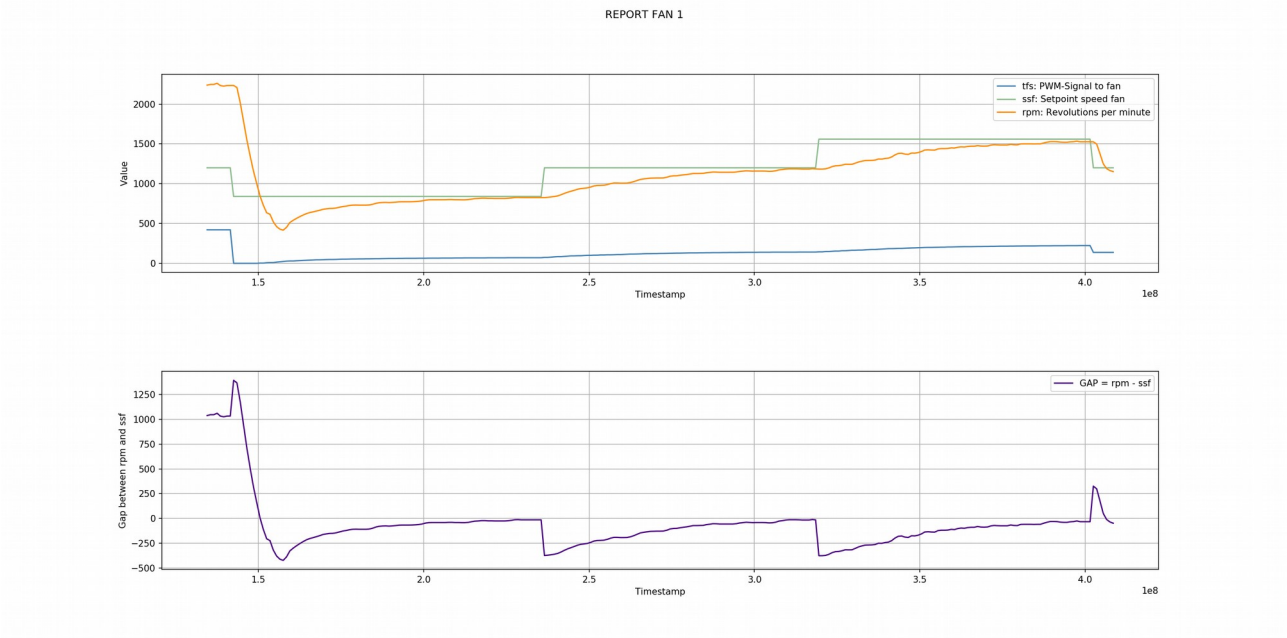


Abbildung 11: Korrekte Erkennung des Tachosignals (rpm)

Gründe für die fehlerhafte Erkennung können u.a. sein:

- Kalte oder fehlerhafte Lötstellen zum Tachosignal des Lüfters → Lötstellen auf Platine und Stecker überprüfen
- Unsauberes elektrisches Tachosignal zum Lüfter, Beurteilung mit Oszilloskop möglich → evtl. Pullup-Widerstand R11, R12 mit 4,7k und/oder Kondensator C1, C2 mit 100nF einbauen oder Änderung der Tachosignalerkennung zwischen FALLING und RISING (KWLConfig.h: TachoSamplingMode), siehe auch https://www.arduino.cc/reference/en/language/functions/external-interrupts/attachinterrupt/

Vor der weiteren Inbetriebnahme ist unbedingt eine korrekte Erkennung der Tachosignals sicher zustellen.

6.5.2 Überprüfung der Anzahl der Tachosignale pro Lüfterumdrehung

Die Tachos der Lüfter können eine unterschiedlich Anzahl von Tachoimpulsen pro Lüfterumdrehung senden. Üblicherweise senden die verbauten Lüfter einen Impuls pro Umdrehung. Ein Anhaltspunkt für den Wert kann dem Datenblatt des Lüfters entnommen werden.

Es sind die folgenden Einstellungen anzupassen: StandardFan1ImpulsesPerRotation und StandardFan2ImpulsesPerRotation.

6.5.3 Einstellen der Lüfterstufen

In der Werkeinstellungen sind vier Lüftungsstufen definiert. Lüftungsstufe 0 ist aus, 1, 2, 3 sind mit jeweils 70,% 100%, bzw 130% Drehzahl der Standardlüftungsstufe definiert. 100% Drehzahl entspricht 1.550 U/min. Die Lüftungsstufen sollten getestet werden:

```
mosquitto_pub -t d15/set/kwl/lueftungsstufe -m 1
mosquitto_pub -t d15/set/kwl/lueftungsstufe -m 2
mosquitto_pub -t d15/set/kwl/lueftungsstufe -m 3
```

6.5.4 Kalibrierung der Lüfteransteuerung

Wenn das Tachosignal korrekt erkannt wird, kann eine Kalibrierung der Lüfteransteuerung vorgenommen werden. Dabei werden die Lüfter in die Lüftungsstufen geschaltet und es werden die PWM-Signale gespeichert, die zur Erreichung der jeweiligen Drehzahlen erforderlich sind. Diese gespeicherten PWM-Signale werden bis zur nächsten Referenzfahrt verwendet.

```
mosquitto_pub -t d15/set/kwl/calibratefans -m YES
```

Die Kalibrierung dauert mehrere Minuten. Maximal wird eine Kalibrierung 10 Minuten (TIMEOUT_CALIBRATION) versucht, dabei darf jeder Schritt einer Lüftungsstufe maximal 5 Minuten (TIMEOUT_PWM_CALIBRATION) dauern. Wenn die Kalibierung nicht erfolgreich ist, bleiben die gespeicherten PWM-Werte unverändert.

6.6 Optional: Sommer-Bypass

Wenn der Sommer-Bypass mit Netzspannung betrieben wird, ist besondere Vorsicht und Fachkenntnis beim elektrischen Aufbau notwendig.

Der Sommer-Bypass ist in der Grundeinstellung im Automatikbetrieb und die Klappe ist geschlossen. Die Betriebsart (Automatik/Manuell) wird im EEPROM gespeichert, ebenso wie die Klappenstellung im manuellen Betrieb.

Um das korrekte Öffnen und Schließen der Klappe in der manuellen Betriebsart zu überprüfen, muss der Sommer-Bypass in den Manuellen Betrieb gesetzt werden und die Klappe per MQTT-Befehl geöffnet und geschlossen werden.

Die folgenden Kommandos setzen die manuelle Betriebsart und öffnen die Klappe.

```
mosquitto_pub -t d15/set/kwl/summerbypass/mode -m manual
mosquitto_pub -t d15/set/kwl/summerbypass/flap -m open
```

Bis zum Schalten der Bypass-Klappe können bis zu 6 Minuten vergehen, nach dem die erforderlichen Bedingungen vorliegen.

Das folgende Kommando schließt die Klappe, die Verzögerung kann ebenfalls wieder bis zu sechs Minuten betragen:

```
mosquitto_pub -t d15/set/kwl/summerbypass/flap -m close
```

Wenn die Klappe in der manuellen Betriebsart korrekt gefahren wurde, kann der Automatikbetrieb überprüft werden. Der Sommer-Bypass öffnet im Automatikbetrieb, wenn die folgenden Bedingungen erfüllt sind:

1. Die Außenlufttemperatur ist höher als TempAussenluftMin (13°C)
2. Die Ablufttemperatur ist höher als TempAbluftMin (24°C)
3. Die Ablufttemperatur ist 2 Grad höher als die Außenlufttemperatur
4. Der Bypass wurde in den letzten HystereseMinutes (60) Minuten nicht geschlossen.

Die Werkseinstellungen für den Sommer-Bypass entsprechen den am Markt anzutreffenden Anlagen.

Für das Testen des Bypasses sind die Temperatursensoren zu entfernen. Es ist ausreichend, die Datenleitungen der Sensoren vom Arduino zu lösen. Die Software für den Arduino muss mit *#define DEBUG* kompiliert werden. Anschließend können die Betriebsart und die Temperaturwerte über den MQTT-Broker in die Steuerung geschrieben werden.

```
mosquitto_pub -t d15/set/kwl/summerbypass/mode -m auto
mosquitto_pub -t d15/debugset/kwl/aussenluft/temperatur -m 14
mosquitto_pub -t d15/debugset/kwl/abluft/temperatur -m 26
```

Der Sommer-Bypass sollte jetzt geöffnet werden.

Die Fahrdauer kann mit der Variablen intervalBypassSummerSetFlaps im Sourcecode angepasst werden.

6.7 Optional: Vorheizregister

Das Vorheizregister wird automatisch bei drohender Vereisung aktiviert. Für das Testen des Bypasses sind die Temperatursensoren zu entfernen. Es ist ausreichend, die Datenleitungen der Sensoren vom Arduino zu lösen. Die Software für den Arduino muss mit *#define DEBUG* kompiliert werden. Anschließend können per mqtt die folgenden Temperaturen vorgegeben werden. Das Vorheizregister sollte jetzt heizen. ACHTUNG: Das Vorheizregister darf nur im Luftstrom betrieben werden.

```
mosquitto_pub -t d15/debugset/kwl/fortluft/temperatur   -m 1
mosquitto_pub -t d15/debugset/kwl/aussenluft/temperatur -m -1.1
```

7 Einstellungen im Betrieb

Alle mqtt-Topics sind in der Datei https://github.com/svenjust/room-ventilation-system/tree/master/Docs/mqtt%20topics/mqtt_topics.ods dokumentiert, ebenso deren Implementierungsstatus. In der Datei sind auch kurze Erläuterung zu den Topics.

Für das Mitlesen aller mqtt-Meldungen der Lüftungsanlage sind zwei Subscribes notwendig:

```
mosquitto_sub -v -h localhost -t d15/state/
mosquitto_sub -v -h localhost -t d15/debugstate/
```

Für das Mitlesen des debugstate muss die Software für den Arduino mit *#define DEBUG* kompiliert werden.

7.1 Setzen der Lüftungsstufe

```
mosquitto_pub -t d15/set/kwl/lueftungsstufe -m <Nr der Lüftungsstufe>
```

7.2 Steuerung der Bypass-Klappe

Um die Bypass-Klappe manuell zu steuern muss sie in den manuellen Betriebsmodus geschaltet werden.

```
mosquitto_pub -t d15/set/kwl/summerbypass/mode -m manual
```

Anschließend kann die Bypass-Klappe geöffnet bzw. geschlossen werden.

```
mosquitto_pub -t d15/set/kwl/summerbypass/flap -m open
mosquitto_pub -t d15/set/kwl/summerbypass/flap -m close
```

8 Bedienung mit Touch-Display

Doku offen...

9 Einbindung dieser Steuerung in fhem

Für die Einbindung ist ein mqtt Broker erforderlich, siehe Abschnitt 6.3. Im fhem muss ein Device für die Verbindungen zum mqtt Broker erstellt werden.

```
define myBroker MQTT 192.168.20.240:1883 ## bitte EIGENE IP-Adresse eintragen
```

Anschließend muss ein Device für die Lüftungsanlage erstellt werden:

```
define kwl MQTT_DEVICE
attr kwl IODev myBroker
attr kwl publishSet 0 1 2 3 d15/set/kwl/lueftungsstufe
attr kwl room DEV_KWL
attr kwl stateFormat transmission-state
attr kwl subscribeReading_Fan1Speed d15/state/kwl/fan1/speed
attr kwl subscribeReading_Fan2Speed d15/state/kwl/fan2/speed
attr kwl subscribeReading_StateLueftungsstufe d15/state/kwl/lueftungsstufe
attr kwl subscribeReading_Temp01Aussenluft d15/state/kwl/aussenluft/temperatur
attr kwl subscribeReading_Temp02Zuluft d15/state/kwl/zuluft/temperatur
attr kwl subscribeReading_Temp03Abluft d15/state/kwl/abluft/temperatur
attr kwl subscribeReading_Temp04Fortluft d15/state/kwl/fortluft/temperatur
```

10 Bauteillisten

10.1 Arduino, LAN-Modul, Temperatursensoren, 4-fach Relais

Basisteile			
Beschreibung	***Bauelement***	***Alternative***	***Anzahl***
Temperatursensoren	DS18B20 im Edelstahlgehäuse		4
Microcontroller	Arduino Mega R3		1
LAN	LAN Shield W5100		1
Relaisboard	4 fach Relais Arduino für Sommer-Bypass und Lüfter		1
optional: 4 fach DAC 0-10V für das Vorheizregister	https://www.horter-shop.de/de/home/93-bausatz-i2c-analog-input-modul-5-kanal-10-bit-4260404260752.html		1
optional: DHT22 Luftfeuchtesensoren,	DHT22		2

für Abluft- und ggfs Zuluftstrom	
optional: VOC Sensor, TGS2600 z.B. Figaro TGS2600	1
optional: CO2-Sensor, MH-Z14 z.B. MH-Z14	1

10.2 Interface Board P300

P300_IF				
Bezeichnung auf Board	***Beschreibung***	***Bauelement***	***Alternative***	***Anzahl***
J1	6 poliger Verbinder zu Relais Board 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 6-polig	Molex_KK-254 AE-6410-06A 1x06_P2.54mm_Vertical	1
J2	3 poliger Verbinder zu DHT1 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 3-polig	Molex_KK-254 AE-6410-03A 1x03_P2.54mm_Vertical	1
J3	3 poliger Verbinder zu DHT2 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 3-polig	Molex_KK-254 AE-6410-03A 1x03_P2.54mm_Vertical	1
J4	8 poliger Verbinder zu Lüftern 2,5mm Pinabstand, MOLEX ist PINKOMPATIBEL	Molex_KK-254 AE-6410-06A 1x08_P2.54mm_Vertical	JST XH2.54 XH 2,54mm, 8-polig	1
J5	3 poliger Verbinder zu DS18B20 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 3-polig	Molex_KK-254 AE-6410-03A 1x03_P2.54mm_Vertical	1
J6	3 poliger Verbinder zu DS18B20 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 3-polig	Molex_KK-254 AE-6410-03A 1x03_P2.54mm_Vertical	1
J7	3 poliger Verbinder zu DS18B20 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 3-polig	Molex_KK-254 AE-6410-03A	1

			1x03_P2.54mm_Vertical	
J8	3 poliger Verbinder zu DS18B20 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 3-polig	Molex_KK-254 AE-6410-03A 1x03_P2.54mm_Vertical	1
J9	2 poliger Verbinder zu 12V Versorgung 2,5mm Pinabstand, nur notwendig bei Horter DAC oder analogem Druckboard	JST XH2.54 XH 2,54mm, 2-polig	Molex_KK-254 AE-6410-03A 1x02_P2.54mm_Vertical	1
J10	4 poliger Verbinder zu analogem Druckboard 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 4-polig	Molex_KK-254 AE-6410-03A 1x04_P2.54mm_Vertical	1
J11	3 poliger Verbinder zu VOC TGS2600 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 3-polig	Molex_KK-254 AE-6410-03A 1x03_P2.54mm_Vertical	1
J12	4 poliger Verbinder I2C 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 4-polig	Molex_KK-254 AE-6410-03A 1x04_P2.54mm_Vertical	1
J13	4 poliger Verbinder zu MH-Z Serial2 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 4-polig	Molex_KK-254 AE-6410-03A 1x04_P2.54mm_Vertical	1
J14	2 poliger Verbinder zu 5V Versorgung 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 2-polig	Molex_KK-254 AE-6410-03A 1x02_P2.54mm_Vertical	1
J15, J16	8 poliger Verbinder zu Display 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 8-polig		2
R1 - R4	Widerstand 4.7k	Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P2.54mm_Vertical		4
R5, R6	Widerstand 220	Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P2.54mm_Vertical		2
R7 - R10	Widerstand 4.7k	Resistor_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P2.54mm_Vertical		4
R11, R12	Widerstand 4,7k	Resistors_THT:R_Axial_DIN0207_L6.3mm_D2.5mm_P2.54mm_Vertical		2
U1, U2	Optokoppler PC817	Package_DIP:DIP-		2

		4_W7.62mm	
C1, C2	Kondensator 100nF	Capacitors_THT:C_Disc_D5.0mm_W2.5mm_P5.00mm	2
ICSP	2x3 ICSP Female Stackable Header	2x3 ICSP Female Stackable Header	1
PinHeader	1x8 Female Stackable Header	1x8 Female Stackable Header	4
PinHeader	1x8 Pinheader	1x8 Pinheader	2
PinHeader	2x18 Pinheader	2x18 Pinheader	1

10.3 Display Board

Display board				
Bezeichnung auf Board	***Beschreibung***	***Bauelement***	***Alternative***	***Anzahl***
J1, J2	8 poliger Verbinder zu Display 2,5mm Pinabstand	JST XH2.54 XH 2,54mm, 8-polig	Kabel direkt ohne Stecker löten	2

10.4 Spannungsversorgung und 230V Lüfteranschluss

HiVolt				
Bezeichnung auf Board	***Beschreibung***	***Bauelement***	***Alternative***	***Anzahl***
F1	250V T 4A	Fuse:Fuseholder_TR5_Littlefuse_No560_No460		1
J1	230V Eingang – Molex kompatibel mit Pluggit	Connector_Molex:Molex_Mini-Fit_Jr_5566-02A_2x01_P4.20mm_Vertical		1
J2	230V Verbindung zu Lüftern – Molex kompatibel zu Pluggit	Connector_Molex:Molex_Mini-Fit_Jr_5566-08A_2x04_P4.20mm_Vertical		1
J3, J4	Stiftleiste 6P	Connectors_Terminal_Blocks:TerminalBlock_bornier-6_P5.08mm		2
J5	12V	Connector_Molex:Molex_KK-254_AE-6410-		1

		02A_1x02_P2.54mm_Vertical		
J6	5V	Connector_Molex:Molex_KK-254_AE-6410-02A_1x02_P2.54mm_Vertical		1
RV1	JNR-10D431K	Varistor:RV_Disc_D12mm_W7.5m~m_P7.5mm		1
T1	HLK-PM12	HLK-PM12		1
T2	5V Schaltnetzteil	Meanwell IRM-05-5 (NOTWENDIG BEI DISPLAY)	HLK-PM05 ohne Display	1

11 Bilder der verwendeten Hardware

11.1 Arduino Mega 2560REV3

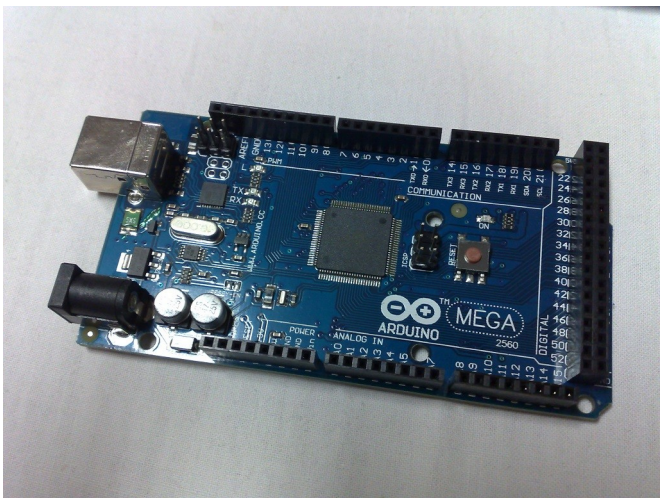
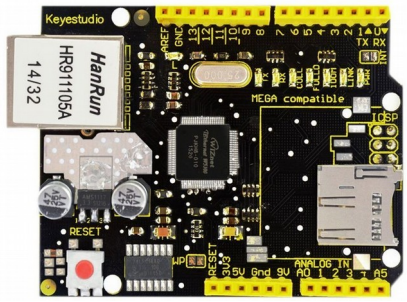
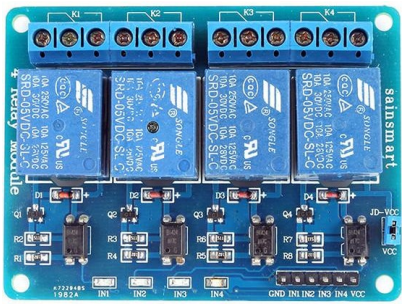


Abbildung 12: Arduino Mega 2560 REV
Bild von Dsimic - Eigenes Werk, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=35850081

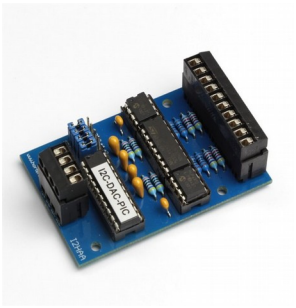
11.2 LAN Shield mit W5100



11.3 Relais Shield

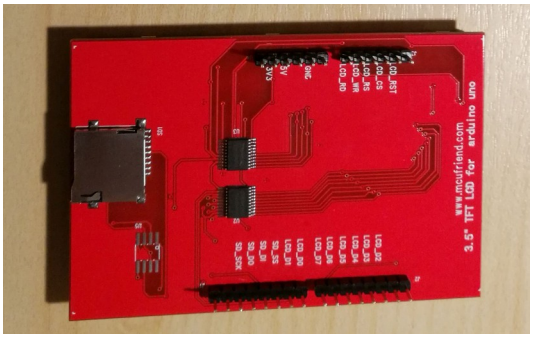


11.4 DAC von Horter



11.5 Temperatursensoren DS18B20

11.6 Touch-Display 3,5“, mcufriend-Library, Auflösung 480x320



Wichtig ist, dass das Display auf das LAN Shield gesteckt werden kann, die Pins also an den beiden Längsseiten und nicht unten an der Querseite hat.