



MELDEN

Die klassische KWL wird smart!

Selbst renommierte Hersteller von kontrollierten Wohnraumlüftungen (KWL) bieten immer noch hauptsächlich Systeme an, die nicht nahtlos in ein Smart Home integriert werden können. Gerade die KWL ist jedoch ein System, bei dem eine ver... [MEHR ANZEIGEN](#) ▾

TEAM



RUND UMS HAUS

WETTBEWERBE

Mach dein Smart Home wirklich smart

smart home

ÄLTESTE ZUERST ^



Niedrigenergiehäuser haben in der Regel eine KWL (kontrollierte Wohnraumlüftung). Diese führt Frischluft zu, vermeidet dabei aber Energieverluste. Das gelingt, indem die Außenluft per Wärmetauscher Energie von der Abluft übernimmt. Man tauscht also Luft aus, hält die Wärme aber drinnen (bzw. im Sommer: draußen).

Leider verbraucht die KWL selbst natürlich Energie. Um so mehr, je höher der Luftstrom eingestellt ist. Verbräuche von 100 bis 400 kWh pro Jahr für ein Einfamilienhaus sind hier üblich.

Mit einer ins Smart Home eingebundenen Steuerung der Belüftungsstufe, gelingt es, die Belüftung nur dann "hochzudrehen", wenn dies auch wirklich notwendig ist. Also wenn z.B. der CO₂-Wert in den Wohnräumen steigt oder die Luftfeuchtigkeit in den Bädern zu hoch ist. Neben einem geringeren Stromverbrauch werden hier auch Filterwechsel reduziert und der Verschleiß der Anlage minimiert.

Dieses Projekt bindet eine Viessmann 300-W mit Bedienteil LB1 in Home Assistant als Smart Home-Zentrale ein. Die Programmierung der Steuerung in Home Assistant erfolgt mit Node RED. Die Microcontroller werden ebenfalls direkt aus Home Assistant heraus mit ESPHome programmiert.

Einen Überblick über den Aufbau des gesamten Projekts zeigt die anhängende Grafik.

Randbemerkung

In diesem Projekt erfolgt die Steuerung der KWL-Anlage über das Lüftungsbedienteil, indem ein zusätzlicher Microcontroller eingebaut wird. Dieser simuliert Tastendrücke auf dem Bedienteil.
Eigentlich gäbe es bessere Alternativen zur Ansteuerung:

Die Kommunikation der zentralen Lüftungseinheit mit dem Bedienteil LB1 erfolgt über einen Modbus. Hier könnte man sich einklinken. Doch Viessmann legt die Datenpunkte des Modbus für Endverbraucher nicht offen. Da ist wohl der OpenData-Gedanke noch nicht bei den Entscheidern angekommen.

Alternativ wäre es auch noch möglich, die KWL 300-W per Modbus an die WO1C-Steuerung unserer Wärmepumpe Vitocal 222-S anzuschließen. Von dort aus wäre sie dann über Vitoconnect per App und per API steuerbar. Das ist die Strategie, auf die Viessmann in Sachen Smart Home verweist. Doch erstens ist die Home Assistant Integration dieser API noch nicht in der Lage, KWLs zu steuern. Und zweitens – und viel wichtiger – die KWL kann nur an die Wärmepumpe ODER an das Bedienteil LB1 angeschlossen werden. Ich würde also das Bedienteil im Wohnbereich als Eingabe- und Kontrollpunkt verlieren. Ein notwendiger Filterwechsel würde dann z.B. nur im Keller an der Wärmepumpe angezeigt. Und ich möchte mich auch auf keinen Fall nur auf Home Assistant oder die App verlassen. Meine Strategie in Sachen Haustechnik ist, wo immer möglich ein Fallback auf eine solide Stand-Alone-Technik zu erhalten.

Das klassische Bedienteil, ist für mich deshalb absolut alternativlos.

Also steuern wir die KWL per Tastensimulation am Bedienteil. Auf gehts!



Die Gesamtkosten für dieses Projekt liegen bei etwa 100 bis 150 € wenn die Elektronikteile z.B. bei AliExpress bestellt werden.

Im Detail hängen die Kosten davon ab, wie viele Wohn-/Schlafräume und wie viele Bäder mit Sensoren ausgestattet werden sollen. In meinem Fall waren das 4 Räume und 2 Bäder.

Folgendes benötigt Ihr dafür...

Elektronik:

- 8 x Microcontroller ESP8266, D1 mini (1 x für die KWL, 1 x für das Steuergerät LB1, je 1 x für jeden Wohn-/Schlafräum und jedes Bad)
- 10 x Feuchte-/Temperatursensor [SHT30](#) (4 x für die KWL, je 1 x für jeden Wohn-/Schlafräum und jedes Bad)
- 4 x CO₂/VOC-Sensor [SGP30](#) (je 1 x für jeden Wohn-/Schlafräum)
- 1 x 8-Kanal I₂C-Multiplexer [TCA9548A](#)
- 7 x Spannungsversorgung 5V für die Microcontroller (außer für den im LB1), z.B. USB-Stecker oder [Low-Loss Netzteilmodul](#)
- 14 x [Anschlusskabel 4-polig SH 1.0mm](#), 200mm lang, für alle Feuchte- und CO₂-Sensoren
- ca. 2 Meter [Flachbandkabel](#), min. 4-polig, zum Verlängern der Sensor-Anschlüsse an der KWL
- 8 x [Tantalelektrolytische Kondensatoren 47µF](#) zur Stabilisierung der Versorgungsspannung an den Microcontrollern (zumindest für die ESPs mit CO₂-Sensor, da diese mehr Strom ziehen)
- Isolierband/Gewebeband und [Schrumpfschlauch](#) in verschiedenen Durchmessern
- Ggf. kleine Kunststoffgehäuse für die Microcontroller, wenn diese offen platziert werden sollen (also z.B. nicht in Unterputz- oder Deckendosen verschwinden)

Werkzeug:

- Elektroniklötkolben, Lötzinn, Elektronikseitenschneider

Vorausgesetzt wird:

- [Home Assistant](#), lauffähig installiert z.B. auf einem Raspberry Pi 4b
- Das Wissen, an welchem Ende man einen Lötkolben anfasst

Dateien / Programme

- Alle relevanten Programmteile finden sich in [diesem Github-Repo](#).



In diesem Projektteil wird ein Herzstück des Projektes gebaut. Es ist der Microcontroller im Lüftungsbedienteil, der Tasteneingaben am Bedienteil simuliert und damit die Lüftungsstufen umschaltet.

Bei der Analyse des bei mir vorhandenen Viessmann LB1 stellte sich erfreulicherweise heraus, dass er perfekte Bedingungen für den Anschluss eines ESP bietet. Auf der Platine findet sich eine 5 Volt Versorgungsspannung, mit der der Microcontroller versorgt werden kann. Außerdem arbeiten die Tasten des LB1 mit 3 Volt-Pullups, die beim Druck auf die Taste auf Masse gezogen werden. Daher können die Ausgänge des ESP im Open Drain Mode ebenfalls direkt mit den Tasten verbunden werden. Sie simulieren dann Tastendrücke, indem sie ihrerseits auf Masse ziehen.

Aufgrund der idealen Voraussetzungen kommt dieser Projektschritt ausschließlich mit dem ESP und ein paar Drähten aus. Weitere Elektronik wird nicht benötigt.

Ich vermute, dass auch Bedienteile anderer Hersteller ähnlich aufgebaut sein werden. Ansonsten sind z.B. Pegelwandler für die Ausgänge notwendig. Für diese gibt es im Web zahlreiche Bauvorschläge, in der Regel mit einem MOS-FET pro Kanal.

Aufbau

Die Elektronik besteht ausschließlich aus einem [ESP8266, D1 mini](#).

Ich denke, die Verdrahtung und die Anschlusspunkte auf der LB1-Platine sind anhand der Bilder und der Leitungsfarben gut nachvollziehbar.

Verbindungen ESP zu LB1-Platine:

5V – an 5V des LB1

GND – an Masse des LB1

D1 – an den Zurück-Taster des LB1

- D2 – an den Hoch-Taster des LB1
- D5 – an den Ok-Taster des LB1
- D6 – an den Runter-Taster des LB1
- D7 – an den Menü-Taster des LB1

Erfreulicherweise findet sich im LB1 sogar ein Hohlraum, der den ESP perfekt aufnehmen kann.

Lediglich für die Leitungsführungen waren kleine zusätzliche Ausschnitte in den inneren Kunststoffteilen notwendig.

Programmierung mit ESPHome

Die Programmierung erfolgt über das ESPHome-Addon von Home Assistant.

Im [Github-Repo](#) zu diesem Projekt befindet sich ein Ordner mit allen ESPHome-Dateien. Die Datei ventilation-control.yaml enthält die hier benötigte Programmierung. Für die Sensoren des zweiten Bades wird eine Kopie dieser Datei angelegt, in der die Substitution-Parameter zu Beginn entsprechend angepasst werden.

Nachdem der ESP sich mit Home Assistant verbunden hat, erscheinen in HA neue Switch-Entities mit denen die Buttons der LB1 betätigt werden können. Das ist für erste Tests praktisch.

Für den laufenden Betrieb viel wichtiger sind aber zwei Services, die ebenfalls in HA exponiert werden:

Service send_button_sequence

Dieser Service erwartet einen String-Parameter "button_sequence" mit einer Buchstabenfolge. Jeder Buchstabe steht für einen Tastendruck:

- b = Zurück (back)
- u = Hoch (up)
- d = Runter (down)
- o = Ok
- m = Menü

Diese Tastendrücke werden in schneller Folge abgearbeitet.

Service set_ventilation_level

Dieser Service erwartet einen Integer-Parameter "ventilation_level" mit einer Ganzzahl im Bereich 1 bis 4. Der Service erstellt automatisch die zugehörige Tastendruckfolge, um die entsprechende Lüftungsstufe einzustellen. Dabei wird initial eine Folge von "Zurück"-Tastendrücken ausgeführt, sodass der LB1 aus jedem Zustand heraus stabil in den Grundzustand geführt wird, von wo aus dann die folgenden Tastendrücke die gewünschten Funktionen ausführen.

Die Node RED-Programmierung in HA verwendet natürlich den Service set_ventilation_level, um ihre errechneten Belüftungsstufen zu setzen.



In diesem Projektteil werden die Microcontroller mit den Luftqualitätssensoren (SGP30) gebaut. Diese befinden sich in den Wohn- und Schlafräumen. Sie ermitteln einen Luftqualitätswert genannt VOC (Volatile Organic Compounds = Flüchtige organische Substanzen) und einen CO₂-Wert. Das Messprinzip beruht auf einer Metalloxid-Schicht, die erhitzt werden muss. Daher benötigt der Sensor etwas mehr Strom (ca. 50 mA). Aufgrund dieses Stromverbrauchs sollte der ESP unbedingt mit einem Tantalelektrolytkondensator zwischen 3,3V und GND versehen werden, um Brown-Outs zu vermeiden. Zusätzlich zu den Luftgütesensoren erhalten die ESPs auch noch Luftfeuchte-/Temperatursensoren (SHT30). Diese dienen vor allem dazu, die Messwerte der Luftgütesensoren zu verbessern. Ohne sie wäre die Messgenauigkeit und Trennschärfe zwischen VOC und CO₂ reduziert.

Die Messwerte dienen dazu, eine Lüftungsstufe basierend auf der Luftqualität zu errechnen. Im nächsten Kapitel kommt eine Lüftungsstufe basierend auf der Feuchte in den Bädern hinzu. Der größere der beiden Werte bestimmt später die tatsächliche Lüftungsstufe, die dann über das Lüftungsbedienteil – wiederum per Microcontroller – eingestellt wird.

Die Küche erhält übrigens bewusst keinen Luftgütesensor. Beim Kochen entstehen ständig Dämpfe mit sehr hohem Anteil organischer Komponenten. Diese würden die KWL rasch auf höchste Stufe steigen lassen. Da die Dämpfe fetthaltig sind, wollen wir jedoch die KWL nicht unnötig damit belasten. Um sowas soll sich die Dunstabzugshaube kümmern – die ist dafür gemacht.

Die Luftgütesensoren sollten möglichst hoch in den Räumen platziert werden, da die belastete Luft häufig warm ist und nach oben steigt, während sich die kühлere Frischluft eher am Boden befindet.

Außerdem sollte vor allem der Feuchte/Temperatursensor nicht zu nah am ESP installiert werden. Schon garnicht sollte er auf den ESP aufgesteckt werden. Es kann sonst zur Erwärmung durch die Infrarotstrahlung des ESPs kommen, evtl. sogar zu RF-Heating, also der Erwärmung durch direkte Einkopplung von Wifi-Sendeleistung in Leiterbahnen und Metallteile des Sensors. Der Sensor misst dann ständig zu hohe Temperaturen und die Taupunktberechnung ist dann ebenfalls fehlerhaft.

In den angehängten Bildern wurde die Schaltung in einen Rauchmelder integriert. Die in den Bildern gezeigte Lochrasterplatine ist für das KWL-Projekt irrelevant. Die Sensoren sitzen in den Bildern unter den weißen Plättchen, die aus der Rauchmelderbasis ragen. Die Spannungsversorgung stammt in den Bildern aus einem im Neubau verlegten Bus mit 5-Volt-Leitung. Bei Nachbau werden hier in der Regel eher 3,3 Volt oder 5 Volt [Low-Loss Netzteilmodule](#) zum Einsatz kommen.

Aufbau

Die Elektronik besteht aus einem [ESP8266, D1 mini](#), CO₂/VOC-Sensor [SGP30](#), einem Feuchte-/Temperatursensor [SHT30](#), und einem Tantalelko zum Puffern der Betriebsspannung des ESP.

Verbindungen ESP zu SGP30:

3,3V – 3,3V

GND – GND

D1 – SCL

D2 – SDA

Verbindungen ESP zu SHT30:

3,3V – 3,3V

GND – GND

D1 – SCL

D2 – SDA

Tantalelko am ESP zwischen 3,3V und GND.

Programmierung mit ESPHome

Die Programmierung erfolgt über das ESPHome-Addon von Home Assistant. Im [Github-Repo](#) zu diesem Projekt befindet sich ein Ordner mit allen ESPHome-Dateien. Die Datei humid-voc-wohnzimmer.yaml enthält die hier benötigte Programmierung. Für die Sensoren der weiteren Wohn und Schlafräume werden Kopien dieser Datei angelegt, in denen die Substitution-Parameter zu Beginn entsprechend angepasst werden.

In der YAML-Datei finden sich auch zwei Werte für eco2_baseline und tvoc_baseline. Dies sind Kalibrierungswerte, die der VOC-Sensor selbst errechnet und immer wieder korrigiert. Nachdem der Sensor einige Zeit gelaufen ist, können die aktualisierten Werte für diese beiden Parameter aus dem ESPHome-Log des ESP ausgelesen und in der YAML-Datei korrigiert werden. Bei einem Neustart des ESP verfügt dieser dann sofort über optimale Werte.

Nachdem der ESP sich mit Home Assistant verbunden hat, erscheinen in HA neue Entities mit den Messwerten für VOC, CO₂, Temperatur, Feuchte und Taupunkt (siehe Screenshot). Diese Entities unterstützen bereits die neuen "Long Term Statistics" von HA, sodass auch Datenverläufe über größere Zeiträume angezeigt werden können.



Schritt 3:
Luftfeuchte- und
Temperatursensoren
für die Bäder

In diesem Projektteil werden die Microcontroller mit den Luftfeuchte- und Temperatursensoren für die Bäder gebaut.

Die Messwerte dienen dazu, eine Lüftungsstufe basierend auf dem Taupunkt der Luft zu errechnen. Im vorhergehenden Kapitel wurde ja bereits eine Lüftungsstufe basierend auf der Luftqualität in den Wohn-/Schlafräumen ermittelt. Der größere der beiden Werte bestimmt später die tatsächliche Lüftungsstufe, die dann über das Lüftungsbedienteil – wiederum per Microcontroller – eingestellt wird.

Die Sensoren sollten möglichst hoch in den Räumen platziert werden, da feuchte Luft leichter ist als trockene (ja, das ist tatsächlich so!) und die kühlere Frischluft sich eher am Boden befindet.

Außerdem sollte der Feuchte/Temperatursensor nicht zu nah am ESP installiert werden. Schon gar nicht sollte er auf den ESP aufgesteckt werden. Es kann sonst zur Erwärmung durch die Infrarotstrahlung des ESPs kommen, evtl. sogar zu RF-Heating, also der Erwärmung durch direkte Einkopplung von Wifi-Sendeleistung in Leiterbahnen und Metallteile des Sensors. Der Sensor misst dann ständig zu hohe Temperaturen und die Taupunktberechnung ist dann ebenfalls fehlerhaft.

Die angehängten Bilder zeigen zwei unterschiedliche Installationsweisen. Eine Installation erfolgte in einer Unterputzdose, eine zweite in einer Aufputz-Deckendose. In beiden Fällen stammt die Spannungsversorgung aus einem im Neubau verlegten Bus mit 5-Volt-Leitung. Bei Nachbau werden hier in der Regel eher 3,3 Volt oder 5 Volt [Low-Loss Netzteilmodule](#) zum Einsatz kommen.

Aufbau

Die Elektronik besteht aus einem [ESP8266, D1 mini](#), einem Feuchte-/Temperatursensor [SHT30](#), und einem Tantalelko zum Puffern der Betriebsspannung des ESP.

Verbindungen ESP zu SHT30:

3,3V – 3,3V

GND – GND

D1 – SCL

D2 – SDA

Tantalelko am ESP zwischen 3,3V und GND.

Programmierung mit ESPHome

Die Programmierung erfolgt über das ESPHome-Addon von Home Assistant.

Im [Github-Repo](#) zu diesem Projekt befindet sich ein Ordner mit allen ESPHome-Dateien. Die Datei humid-bad-eltern.yaml enthält die hier benötigte Programmierung. Für die Sensoren des zweiten Bades wird eine Kopie dieser Datei angelegt, in der die Substitution-Parameter zu Beginn entsprechend angepasst werden.

Nachdem der ESP sich mit Home Assistant verbunden hat, erscheinen in HA neue Entities mit den Messwerten für Temperatur, Feuchte und Taupunkt (siehe Screenshot). Diese Entities unterstützen bereits die neuen "Long Term Statistics" von HA, sodass auch Datenverläufe über größere Zeiträume angezeigt werden können.



In diesem Projektteil wird der Microcontroller mit den Luftfeuchte- und Temperatursensoren für die vier Luftströme der KWL-Zentraleinheit gebaut. Mit ihnen werden die eingehenden und ausgehenden Luftströme des Hauses vor und hinter dem Wärmetauscher vermessen. Sie nennen sich in der Reihenfolge der Durchströmung:

- Außenluft,
- Zuluft,
- Abluft,
- Fortluft

Der Microcontroller errechnet aus der Luftfeuchte und der Temperatur außerdem den Taupunkt. Dieser ist für viele Funktionen aussagekräftiger als die relative Luftfeuchte.

Die Messwerte dienen als Bezugsgrößen für zahlreiche Berechnungen der gesamten Steuerung,
z.B. für die Berechnung des Taupunkt-Ziels in den Bädern.

Die Außenluft-Temperatur ist außerdem eine hervorragende Quelle für die Anzeige der Außentemperatur in der Hausautomatisierung. Bei korrekter Platzierung der Luftansaugung (Norden) ist sie weit unabhängiger von der Sonneneinstrahlung als jeder wandmontierte Sensor.

Die KWL-Zentraleinheit besitzt außerdem einen aktivierbaren Bypass, der den Wärmetauscher umgeht. Ohne den Bypass überträgt die abfließende Luft im Wärmetauscher ihre Wärme an die Frischluft. Im Winter ist dieses Energierecycling sehr wünschenswert, um den Abfluss von Heizenergie beim Lüften zu verhindern. Im Sommer umgeht der Bypass jedoch nachts

den Wärmetauscher, um die Abfuhr von Wärme aus dem Haus zu ermöglichen. Das Haus würde sonst nachts nur sehr langsam abkühlen.

Die Logik des Projektes errechnet aus den Temperaturverhältnissen der Luftströme, ob die Bypass-Klappe geöffnet oder geschlossen ist. Hiermit kann kontrolliert werden, ob der Bypass wunschgemäß arbeitet.

Aufbau

Die Elektronik besteht aus einem [ESP8266, D1 mini](#), einem 8-Kanal I₂C-Multiplexer [TCA9548A](#), und vier Feuchte-/Temperatursensor [SHT30](#). Der Multiplexer wird benötigt, da die SHT30-Sensoren nur auf zwei unterschiedliche I₂C-Adressen einstellbar sind. Das genügt nicht, um vier Sensoren am ESP zu betreiben. Der Multiplexer bietet insgesamt 8 unabhängige I₂C-Kanäle, sodass die Sensoren alle auf die gleiche Adresse eingestellt bleiben können.

Die Schaltung lässt sich in den anhängenden Bildern recht gut erkennen. Sie enthält außer den oben genannten Bauteilen keine weitere Elektronik. Ein Tantalelko zwischen 3,3V und GND des ESP ist hier nicht unbedingt erforderlich (schadet aber auch nicht). Ich verbaue diese Elkos prinzipiell sehr gerne, da sie Brown-outs bei raschen Lastwechseln der ESPs verhindern.

Verbindungen ESP zu Multiplexer:

3,3V – Vin

GND – GND

D1 – SCL

D2 – SDA

Verbindungen ESP zu den vier Sensoren:

3,3V – 3,3V (an jedem Sensor)

GND – GND (an jedem Sensor)

Verbindungen Multiplexer zu den vier Sensoren:

SDx – D2 des Sensors x

SCx – D1 des Sensors x

(mit x von 1 bis 4)

Die vier Sensoren werden in die Luftströme der zentralen KWL-Einheit eingehängt. Die Anschlussrohre sind meist nur aufgesteckt. An den Steckverbindungen können die Sensoren eingeschoben werden, sodass sie sich im Luftstrom befinden (siehe Bilder).

Programmierung mit ESPHome

Die Programmierung erfolgt über das ESPHome-Addon von Home Assistant.

Im [Github-Repo](#) zu diesem Projekt befindet sich ein Ordner mit allen ESPHome-Dateien. Die Datei ventilation-metric.yaml enthält die hier benötigte Programmierung.

Nachdem der ESP sich mit Home Assistant verbunden hat, erscheinen in HA insgesamt 12 neue Entities mit jeweils den Werten von Temperatur, rel. Luftfeuchte und Taupunkt für Außenluft, Zuluft, Abluft und Fortluft (siehe Screenshots). Diese Entities unterstützen bereits die neuen "Long Term Statistics" von HA, sodass auch Datenverläufe über größere Zeiträume angezeigt werden können.



Schritt 5: Programmierung und Visualisierung mit Home Assistant

In diesem Projektteil wird die zentrale Steuerung und die Visualisierung in Home Assistant implementiert.

Home Assistant sammelt die Messwerte der Sensoren in den KWL-Luftkanälen, den Wohn/Schlafräumen und den Bädern. Daraus wird eine notwendige Belüftungsstufe errechnet und diese wird dann über den ESP im Lüftungsbedienteil aktiviert. Einen Komplettüberblick über diese Informationsströme liefert der Projektplan "Wohnraumlüftung steuern entlang Luftgüte und Taupunkt" im Kapitel "Die Idee".

Node RED

Zu Verarbeitung der anfallenden Sensordaten setzt dieses Projekt nicht auf die normalen Logikkomponenten ("Automations") von Home Assistant. Statt dessen kommt Node RED zum Einsatz, für das ein voll integriertes Home Assistant Add-on existiert. Node RED ist bei komplexen Verarbeitungsfunktionen ungleich flexibler und übersichtlicher. Außerdem können Funktionen bei Bedarf direkt in JavaScript geschrieben werden.

Grundsätzlich erfolgt die Programmierung in Node RED jedoch mit einem visuellen Editor, mit dem Bausteine in einem Schaltplan ("Flow") platziert und verdrahtet werden.

Der gesamte Node RED Flow für dieses Projekt findet sich als Export im [Github-Repo](#). Er kann direkt in Node RED importiert werden. Er enthält einige Bezüge zu anderen Flows des Autors, die jedoch leicht entfernt oder korrigiert werden können.

Wichtig für die Funktion des Flows ist eine Node RED Ergänzung ("Contrib"), die zuvor installiert werden muss. Hierzu in Home Assistant den Node RED Editor öffnen, rechts oben auf das Hamburger-Icon (drei Striche) klicken, dort auf "Manage Palette", "Install" und dort die Contrib "[node-red-contrib-pid](#)" suchen und installieren. Diese Contrib fügt einen PID-Node (Proportional-Integral-Differential-Regler) hinzu, den der Flow dieses Projektes als Regelglied für die Lüftungssteuerung verwendet.

Eine weitere Voraussetzung für das Funktionieren des Flows ist ein Input-Select, dass in der configuration.yaml von Home Assistant hinzugefügt werden muss. Es ist das Select (= Dropdown), mit dem der Lüftungslevel von Hand eingestellt werden kann. Dieses Select wird auch von Node RED umgestellt, um den aktuellen Lüftungslevel automatisch zu verändern. Der Code hierzu findet sich im [Github-Repo](#) in der Datei /HomeAssistant/add-to-configuration.yaml.

Danach kann der Flow aus dem [Github-Repo](#) in Node RED importiert werden (ebenfalls über das Hamburger Icon). Er liegt in der Datei /NodeRED/flow_Belüftung.json

Der Flow exponiert diverse neue Entities, die Berechnungsergebnisse und Zwischenergebnisse enthalten.

Der Algorithmus

Grundsätzlich wird aus den Messwerten aller Sensoren die gewünschte Lüftungsstufe errechnet.

Hierzu wird zunächst der höchste gemessene CO₂-Wert über alle Räume bestimmt, sowie der höchste gemessene Taupunkt. Aus beiden Werten wird jeweils eine benötigte Lüftungsstufe berechnet. Es gibt also als Zwischenwerte eine Lüftungsstufe basierend auf dem höchsten CO₂-Wert und eine weitere basierend auf dem höchsten Feuchtwert in den Bädern. Die jeweils höhere der beiden Lüftungsstufe gewinnt und wird dann an das Lüftungsbedienteil gesendet.

Die Berechnung der Lüftungsstufen erfolgt über PI-Regler. Der verwendete PID-Node in Node RED wurde mit P- und I-Werten, aber ohne D konfiguriert. Eine leichte Hysterese sorgt dafür, dass die Lüftungsstufe nicht schon bei kleinen Schwankungen hin und her schaltet.

Der Zielpunkt des PI-Reglers für den CO₂-Wert der Räume ist fest auf 900 ppm eingestellt. Dies kann in Node RED geändert werden.

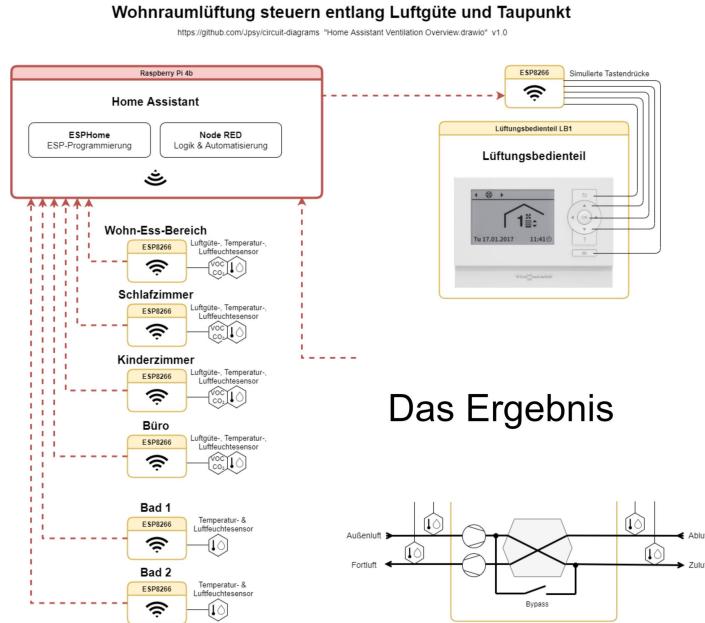
Der Zielpunkt des PI-Reglers für den Taupunktwert der Bäder wird hingegen dynamisch errechnet. Er soll 5,5 °K unter der Temperatur der KWL-Abluft liegen. Hiermit wird garantiert, dass die Temperatur der Bäder weit genug über dem Taupunkt liegt, um ein rasches Abtrocknen zu garantieren. So wird Schimmelbildung (bzw. Bazillus niger) verhindert. Hat allerdings die Zuluft schon einen höheren Taupunktwert, so wird der Zielpunkt entsprechend nach oben korrigiert (Zuluft + 0,5 K), denn die Luft in den Bädern kann ja nicht trockener werden als die frisch zugeführte Luft von außen.

Im der Node RED Flow wird außerdem aus den Temperaturunterschieden der Luftströme am zentralen KWL-Gerät bestimmt, ob die Bypass-Klappe aktuell geöffnet oder geschlossen ist. Bisher wird das Ergebnis dieser Berechnung allerdings im Dashboard nur zu Kontrollzwecken angezeigt, aber noch nicht für Steuerungen verwendet.

Visualisierung

Die Visualisierung der Messwerte erfolgt über ein übliches Home Assistant Lovelace Dashboard.

Der YAML-Code dieses Dashboards liegt ebenfalls im [Github-Repo](#) bereit (Datei /HomeAssistant/dashboard_Beluftung.yaml). In Home Assistant kann dazu ein leeres Dashboard angelegt werden und dann per "Benutzeroberfläche konfigurieren" und "Raw-Konfigurationseditor" durch den Yaml-Code im Repo ersetzt werden. Ein Screenshot des gesamten Dashboards hängt hier als Bild an.



Das Ergebnis

Voller Erfolg

Das Projekt hat unsere KWL wirklich auf einen neuen Level gehoben. Die Belüftung steuert sich jetzt vollständig selbst und spart dabei auch noch Strom, da sie keinem stumpfen Zeitplan folgt, sondern bedarfsgerecht optimal arbeitet.

Bei Abwesenheit schaltet die Lüftung nun nach kurzer Zeit auf Stufe 1 herunter. Dafür geht sie nach dem Duschen aber auch gerne mal vorübergehend auf Stufe 4 um die Bäder rasch abzutrocknen. Das macht sich in den Bädern sehr angenehm bemerkbar – das Gefühl von Schwüle nach dem Duschen verschwindet. Im Mittel läuft die Lüftung trotzdem auf erheblich niedrigeren Stufen und spart deutlich Energie.

Der ursprüngliche Plan, die Lüftung beim Öffnen von Fenstern automatisch auf Stufe 1 herunterzufahren hat sich völlig erübrig. Über die CO₂-Erkennung geschieht das völlig von selbst. Es werden keine Kontakte an sämtlichen Fenstern benötigt.

Das Video im Anhang zeigt die manuelle Steuerung des Lüftungsbedienteils aus Home Assistant heraus. Die rasche Folge von Tastendrücken, die der im LB1 eingebaute Microcontroller erzeugt, ist gut zu erkennen.

Interessante Messwerte

Die Messwerte der eingesetzten Sensoren sind auch über das Projekt hinaus interessant. Der Abschnitt "Pups und Promille" weiter unten beleuchtet einige unerwartete Ableitungen aus den Messergebnissen. Aber auch die

simple Messung der Außenluft im Ansaugkanal der KWL ist erstaunlich hilfreich. Sie liefert die mit Abstand besten Messwerte für die aktuelle Außentemperatur. Weit besser z.B. als der Außenfühler der Wärmepumpe, der im Sommer am Spätnachmittag immer eine ordentliche Delle nach oben zeigt, weil dann die Sonne auf die Nordseite des Hauses scheint. Wir verwenden diese KWL-Temperatur mittlerweile in unserer Visu zur Anzeige der aktuellen Außentemperatur und aller dadurch gesteuerten Prozesse. Auch der Taupunkt der Außenluft ist für diverse Steuerungen ein wichtiger Messwert, den wir hier nebenbei und in hoher Qualität erhalten.

Pups und Promille 😅

Aus der ständig protokollierten Luftqualität lassen sich auch einige ganz unerwartete Details ablesen:

Als die VOC-Kurven immer wieder sehr hohe und kurze Peaks zeigten (siehe Screenshot), glaubte ich zunächst an einen Microcontrollerfehler. Rasch wurde jedoch klar, dass diese Peaks mit den Bewohnern wandern und vorzugsweise nachts in den Schlafräumen auftreten. Die Ursache ist... Biogas!

Die VOC-Sensoren reagieren hochempfindlich auf Wasserstoff. Dieser hat einen Anteil von ca. 20% an unseren Darmwinden. Der Wasserstoff steigt extrem schnell nach oben. Er ist ja das mit Abstand leichteste aller Gase. Bereits 1 bis 2 Minuten nach einem "Flatus" zeigt der an der Decke montierte Sensor einen massiven Peak auf z.T. mehrere 1000 ppb (parts per billion = Teile pro Milliarde). Genau so schnell normalisiert sich der Wert aber auch wieder, da die Gasmenge letztlich gering ist und sich rasch verteilt.

Und noch eine weitere Überraschung fanden wir:

Nach der Rückkehr von einer Party zeigt die nächtliche VOC-Kurve im Schlafgemach der Hausbesitzer den abgeatmeten Alkohol (und/oder dessen flüchtige Abbauprodukte) in einer saubereren e-Kurve bis zum nächsten Morgen (siehe Screenshot). Mit diesem Projekt ist also auch ein ständig aktiver Atemluftalkoholtest für alle Bewohner entstanden. Das wird interessant, wenn uns Nachwuchs in die Pubertät kommt. 😊

Weitere Entwicklung

In einem nächsten Ausbauschritt soll die Steuerung noch um die Belüftungsstufe 0 ergänzt werden, also die komplette Abschaltung der Lüftung. Normalerweise werden KWLs nie abgeschaltet, da Niedrigenergiehäuser weitgehend luftdicht gebaut sind und ohne Belüftung die Gefahr der Schimmelbildung sehr groß ist. Die Stufe 0 ist daher auch in den üblichen Zeitprogrammen der Steuerungen nicht einstellbar. Sie macht jedoch Sinn, während z.B. konventionell gelüftet wird. Die automatische

Steuerung dieses Projekts kann das nun anhand der Luftgüte- und Feuchtigkeitswerte sicher erkennen und die KWL ohne Risiko völlig abschalten. Sinnloser Stromverbrauch wird so vermieden.

Eine weitere Verbesserung plane ich für die Steuerung der Bypass-Klappe. Die im Projekt umgesetzte Erkennung des aktuellen Status dieser Klappe ist schon ein Zwischenerfolg. Die sichere Funktion des Bypass ist in einem Niedrigenergiehaus elementar, da er im Sommer für die Abkühlung sorgt, im Winter aber zuverlässig geschlossen sein muss, da sonst massive Energieverluste auftreten. Die Kontrolle des Status und Meldung von Fehlern in der Visu ist daher eine nächste wichtige Ausbaustufe.

Darüber hinaus möchte ich die Bypassklappe auch aktiv aus Home Assistant heraus steuern. Die in der KWL bereits integrierte Bypasssteuerung hat zwar grundsätzlich den richtigen Ansatz, die KWL misst jedoch nach meinen Tests ständig zu niedrige Temperaturen und die Klappe wird daher in Sommernächten oft nicht oder zu spät geöffnet. Tatsächlich bietet das Lüftungsbedienteil keine direkte Möglichkeit, die Klappe zu öffnen oder zu schließen. Indirekt ist dies jedoch durchaus möglich, indem man die Steuerparameter für die Klappe im LB1 ändert.

Vielleicht fallen Euch ja noch weitere coole Ergänzungen zu diesem Projekt ein. Die Menge an Messwerten, die generiert wird, enthält gewiss noch den ein oder anderen Schatz, den es zu heben gilt.



Du hast das schlaueste Smart Home im ganzen Land? Temperatursensoren steuern die Heizung und Rollläden? Dein Smart Home schaltet automatisch das Licht und die Stereoanlage an, wenn du heim kommst? Dein Türschloss öffnet per Fingerprint? Die Katzenklappe erkennt das Gesicht deiner Katze? Dann bist Du hier richtig! Präsentiere dein Projekt beim Smart Home – The next Level Wettbewerb! Make sucht in Kooperation mit AVM Smart Home Lösungen, die wirklich nützlich sind. Wir suchen Smart Home Projekte die übers WLAN kommunizieren und verschiedene Sensoren und Aktoren nutzen und miteinander verknüpfen. Es ist nicht notwendig, AVM-Produkte einzusetzen; Ihr habt die freie Wahl, welche Hard-und Software Ihr benutzen wollt. Diese Vorlage zeigt dir, wie du das Projekt ausfüllen kannst. Du kannst den Anleitungstext löschen, wenn du deine Inhalte hinzugefügt hast. Sobald du mit dem Projekt begonnen hast, kannst du es unter "Meine Projekte" finden und jederzeit weiterbearbeiten. Dein Projekt wird automatisch zum Einsendeschluss des Wettbewerbs eingereicht. Wenn du dein Projekt neu entwickelst, freuen wir uns, wenn du einzelne Etappen deines Baufortschritts auf MakeProjects festhältst. Die Deadline wird der 10. September 2021, 23:59 Uhr sein. Das Projekt wird erst nach Ende der Abgabefrist bewertet. Eine Jury bestehend aus AVM, dem Make-Magazin und Makern aus der Community prämiert die 10 coolsten Projekte. Bis dahin kannst du dein Projekt zum Beispiel auch zwischenspeichern und später weiterbearbeiten.

Wichtig ist, dass das Projekt ab dem Zeitpunkt der Deadline als „öffentlich“ abgespeichert ist, damit auch andere sich von dir inspirieren lassen können. Mit dem Start eines Projektes unter der Wettbewerbs-Vorlage stimmst du den Teilnahmebedingungen zu und nimmst an dem Wettbewerb teil.

