

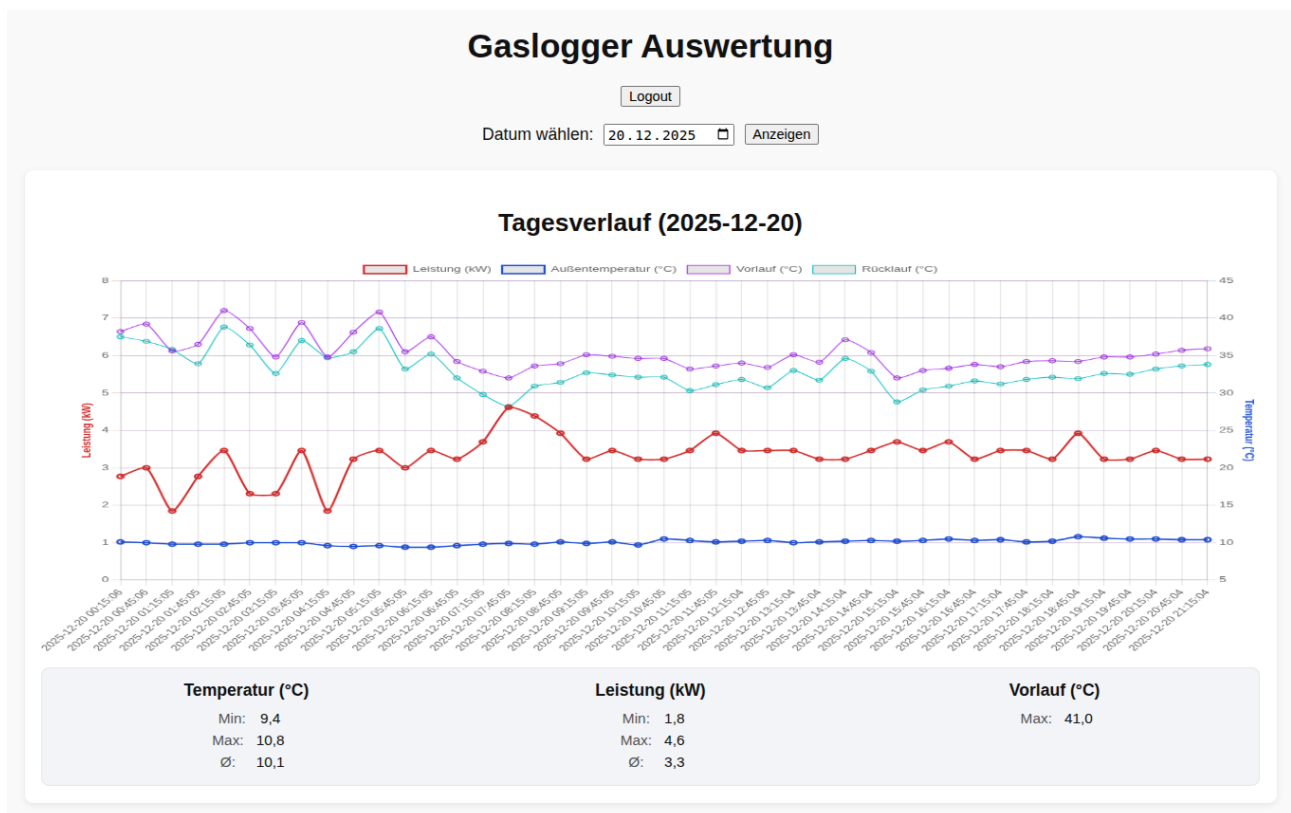
Gasthermen-Logger mit Shelly, ESP und Webserver

Ziel des Projektes

Ich wollte meine Gasheizung und mein Haus besser verstehen, insbesondere in Vorbereitung auf die Umrüstung des Hauses auf eine Wärmepumpe. Ich hatte den Verdacht, dass die Gasheizung, die mein Installateur vor 15 Jahren eingebaut hat, überdimensioniert war. Bei einer Gasheizung ist das nicht so schlimm, aber eine Wärmepumpe sollte man genauer dimensionieren. Ich wollte mich dabei nicht nur auf die Heizlastberechnung nach DIN EN 12831 verlassen, sondern das mit realen Daten meines Hauses ergänzen.

Leider ist die Schnittstelle meiner Gasheizung nicht auslesbar. Ich weiß also nicht, welche Leistung das Haus gerade abfragt. Deshalb musste ich einen kleinen Umweg gehen: über den Gaszähler. Ich habe einen gewöhnlichen BK-G4 Gaszähler. Damit geht das wunderbar, da der einen Magneten im Zählwerk mitlaufen lässt, der eichrechtskonform über ein Reed-Kontakt auslesbar ist. Der BK-G4 ist weit verbreitet, andere Gaszähler haben ähnliche magnetische Schnittstellen.

Das Ergebnis kommt u.a in Form solcher Kurven:



Bestandteile des Projekts

Das Projekt besteht grundsätzlich aus drei Komponenten:

- einem Shelly Plus (mit Addon zum Anschluss von drei Temperaturfühlern DS18B20)
- einem ESP zum Auslesen des Gaszählwerks mit angelötetem Reed-Kontakt
- einem Webserver zum Speichern, Anzeigen und Auswerten der Daten (MySQL, php)

Stückliste

- 1 Stk. Shelly Plus
- 1 Stk. Shelly Addon
- 3 Stk. Temperatursensoren DS18B20
- 1 Stk. AIthinker ESP32Cam
- USB-Board für AIthinker ESP32Cam
- ein Reed-Kontakt
- USB-Netzteil
- Webserver (Shared Hosting oder z.B. Raspberry, nach Belieben). Der Server braucht MySQL (oder kompatibel) und PHP sowie ein Verzeichnis, das im Browser aufgerufen werden kann.

Insgesamt kosten die Teile unter 100€.

Vorbereitung

Vor dem Start des Projektes holt man sich folgende Informationen ein:

- die Impulse pro m³ Gas des Gaszählers. Die stehen vorne auf dem Gaszähler. Bei mir entspricht ein Impuls 0,01m³ Gasverbrauch
- den Brennwert des Gases. Der steht auf der letzten Gasrechnung. Der liegt so um die 10kWh/m³, bei mir z.B. 11,54kWh/m³

Hat man beide Werte, kann es los gehen.

Vorkenntnisse

Ich setze in dieser Anleitung voraus, dass man:

- weiß, wie man einfache Dinge lötet (kein SMD-löten)
- weiß wie man einen ESP mit der ArduinoIDE (o.ä.) programmiert
- weiß, wie man einen Webserver betreibt oder ein Shared Hosting nutzt
- weiß, wie man einen Shelly fachgerecht anschließt (Arbeiten mit 230V!) und einrichtet

Einrichtung

Shelly

ShellyPlus mit ShellyAddon zusammenklicken und drei Temperaturfühler DS18B20 an das Addon anschließen. Die drei Temperaturfühler so installieren:

1. Außerhalb des Hauses im Schatten. Bei mir hängt er einfach durch das Kellerfenster nach außen
2. Heizungsvorlauf (und zwar im Heizkreis der Heizung, nach dem Dreiwegeventil zur Warmwasserbereitung)
3. Heizungsrücklauf

Die Temperaturfühler für Vor- und Rücklauf habe ich einfach in einen Schlitz in der Dämmung der Heizungsrohre gesteckt. Die Temperaturfühler sollten dicht am Rohr anliegen.

Shelly nach Anleitung ans Netz anschließen und ins WLAN bringen.

Dann ruft man in einem Browser diese Zeile auf:

<http://192.168.178.55/rpc/Shelly.GetStatus>

wobei natürlich 192.168.178.55 durch die IP des Shelly ersetzt werden muss.

Dann bekommt man etwas angezeigt, was in etwa so aussieht:

```
{"ble":{}, "cloud":{"connected":true}, "input:0":{"id":0, "state":false}, "mqtt":  
{"connected":false}, "switch:0":{"id":0, "source":"timer", "output":false, "temperature":{"tC":52.0,  
"tF":125.5}}, "sys":  
{"mac":"C4D8D55E74C8", "restart_required":false, "time":"16:46", "unixtime":1766245563, "uptime":6583129  
,"ram_size":246236, "ram_free":141636, "fs_size":458752, "fs_free":151552, "cfg_rev":20, "kvs_rev":0, "sch  
edule_rev":0, "webhook_rev":0, "available_updates":{"stable":{"version":"1.3.3"}}}, "temperature:100":  
{"id": 100, "tC":10.2, "tF":50.5}, "temperature:101":{"id": 101, "tC":31.6,  
"tF":88.9}, "temperature:102":{"id": 102, "tC":33.8, "tF":92.8}, "wifi":  
{"sta_ip":"192.168.178.55", "status":"got ip", "ssid":"MeineWLANSSID", "rssi":-74}, "ws":  
{"connected":false}}
```

Dann weiß man, dass der Shelly läuft und der ESP ihn abfragen kann. Man sucht sich diese Stellen heraus:

```
"temperature:100":{"id": 100, "tC":10.2, "tF":50.5},  
"temperature:101":{"id": 101, "tC":31.6, "tF":88.9},  
"temperature:102":{"id": 102, "tC":33.8, "tF":92.8}
```

Anhand der „tC“ schließen wir auf den Einbauort. Bei 10,2°C Außentemperatur fährt in diesem Beispiel die Heizung gerade eine Vorlauftemperatur von 33,6°C und zurück kommen 31,6°C. Die Vorlauftemperatur ist immer höher als die Rücklauftemperatur.

Wir brauchen die IDs dazu, also weiß ich jetzt:

ID:100 ist die **Außentemperatur**

ID:101 ist die **Rücklauftemperatur**

ID:102 ist die **Vorlauftemperatur**

Diese Infos brauchen wir später im ESP.

ESP

Ich habe ein ESP32Cam-Modul verwendet, einfach weil es alles mitbringt, was man braucht und hier noch rumflieg. Außerdem ist es preiswert.

Hier ist folgendes zu tun:

Hardware

Man lötet einen Reed-Kontakt zwischen GND und IO13, am Besten mit einem kurzen Stück flexiblem Kabel dazwischen.

Software

Ein Arduino-Projekt liegt dem ZIP bei. Ich gehe nicht auf jeden Schritt ein; wie man einen ESP grundsätzlich mit der ArduinoIDE programmiert, setze ich in dieser Anleitung voraus.

Im Code muss man die **fett** gedruckten Texte ändern:

```
const char* WIFI_SSID      = "MeineSSID";  
const char* WIFI_PASSWORD = "MeingeheimenWLANPasswort";  
  
const char* SERVER_UPLOAD_URL   = "https://www.PfadzumeinemServer/gasupload.php";  
const char* SERVER_BRENNWERT_URL = "https://www.PfadzumeinemServer/brennwertabfrage.php";  
const char* AUTH_KEY            = "meinGeheimerAuthToken"; //muss mit config.php identisch sein  
const char* DEVICE_ID           = "ESP32CAM_GAS_1"; //frei wählbar  
  
const char* SHELLY_STATUS_URL = "http://IPvonmeinemShelly/rpc/Shelly.GetStatus";
```

sowie weiter unten:

```
t.temp_out      = doc["temperature:100"]["tC"].as<float>(); //hier ggf. ID ändern  
t.temp_ruecklauf = doc["temperature:101"]["tC"].as<float>(); //hier ggf. ID ändern  
t.temp_vorlauf  = doc["temperature:102"]["tC"].as<float>(); //hier ggf. ID ändern
```

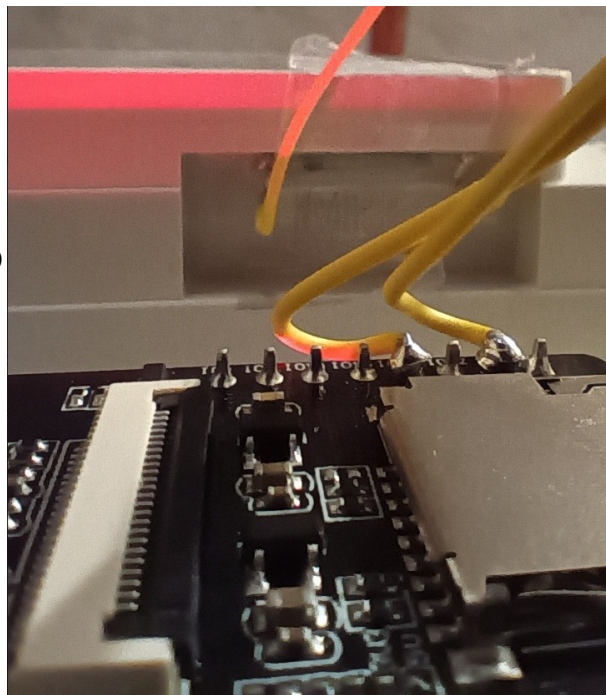
Hier trägt man die IDs der Temperaturfühler ein, die wir oben ermittelt haben.

Fertig geflasht und verlötet wird der Reed-Kontakt mit einem Stück Klebestreifen unter das Zählwerk des Gaszählers in die Aussparung geklebt. Die beste Position muss man ein bisschen experimentell ermitteln.

Der ESP ist so programmiert, dass die Blitz-LED anzeigt, wenn der Reed-Kontakt auslöst, damit kann man die Position halbwegs gut ermitteln.

Bei mir sitzt der Magnet des Zählwerks etwas leicht rechts der dritten Nachkommastelle an der Vorderseite der Aussparung. Er löst aus, wenn die 9 der dritten Nachkommastelle in der Anzeige zur 0 wird und geht aus, wenn die 0 langsam verschwindet.

Rechts sieht man, wie das bei mir aussieht:



Funktionsbeschreibung

Der ESP macht folgendes:

Start

Beim Start zieht der ESP sich den Brennwert des Gases aus der config.php des Webservers. Der Wert wird im NVS des ESP abgelegt, falls der Abruf mal misslingt.

Den Wert braucht der ESP, um von m³ auf kW(h) umzurechnen. Die LED blinkt dreimal, wenn das funktioniert hat. Blinkt die LED viermal, ist der Abruf misslungen, er hat aber einen alten Wert im NVS gefunden, den er weiter verwendet. Hier sollte dann die Verbindung zum Server überprüft werden, da dann ziemlich sicher auch der Datenupload misslingen wird.

Blinkt die LED sechs mal, liegt ein Fehler vor, der ESP kann nicht richtig arbeiten.

Außerdem wird die Zeit von einem Zeitserver abgerufen, da die Datensätze mit einem Zeitstempel versehen werden.

Im Betrieb

Der ESP zählt die Impulse des Gaszählers mit. Daraus berechnet er die aktuelle Leistung der Gastherme. Alle halbe Stunde wird die Leistung der Gastherme, die Gesamtanzahl der Impulse seit erstem Start sowie die drei Temperaturniveaus zum Webserver übertragen. Alle 12 Stunden wird die Gesamtanzahl der Impulse im NVS abgelegt. Falls der ESP mal vom Strom getrennt wird, wird ab da weiter gezählt. Ich habe das nicht häufiger gemacht, um die Lebensdauer des NVS zu schonen. In regelmäßigen Abständen sollte der Blitz angehen, nämlich immer, wenn der Reed-Kontakt eine Umdrehung des Zählwerks detektiert.

Webserver

Man braucht einen fertig eingerichteten Webserver mit PHP und MySQL (oder kompatibel). Wie man einen Webserver einrichtet, setze ich in dieser Anleitung voraus und gehe da nicht weiter drauf ein. Der Webserver kann entweder im Internet (auch als Shared Hosting) oder im eigenen Heimnetz (z.B. ein Raspberry) sein. Beides funktioniert im Prinzip gleich gut, wobei der Raspberry im Heimnetz natürlich nicht von außen erreichbar ist.

Datenbank einrichten

Man richtet auf dem Server eine Datenbank ein. Auf die Details gehe ich hier nicht ein, ich setze die Kenntnisse, wie das geht, in dieser Anleitung voraus. Die Zugangsdaten der Datenbank merkt man sich für die config.php.

Dateien übertragen

Man überträgt die Dateien aus dem Ordner „Server“ des ZIP in ein geeignetes Verzeichnis auf dem Server. Alle Dateien müssen im selben Verzeichnis liegen. Das Verzeichnis muss so liegen, dass es von einem Browser aus aufrufbar ist (z.B. im „public_html“). Das variiert von Server zu Server.

config.php

In der config.php muss man die **fett** gedruckten Texte anpassen:

```
<?php
// Datenbank-Zugangsdaten
define("DB_HOST", "DeinDBServerURL.de"); // Hostname des DB-Servers
define("DB_USER", "dbuser"); // Benutzername der Datenbank
define("DB_PASS", "dbpassword"); // Passwort der Datenbank
define("DB_NAME", "dbname"); // Datenbankname

define('DASHBOARD_PASSWORD', 'deinPasswortHier'); //Im Klartext, wird bei erstem Aufruf der Auswertung abgefragt
define('DASHBOARD_COOKIE_NAME', 'gaslogger_auth');//Cookie zur Speicherung der Zugangsberechtigung, nicht ändern!

// Auth-Key für Upload-Skript
define("API_KEY", "meinGeheimerAuthToken"); // frei wählbarer, muss identisch mit dem Token des ESP-Codes sein

// Start-Zählerstand des Gaszählers in m³ (Ablesewert beim Projektstart)
define("START_M3", 12345.67);

// Faktor pro Tick (m³ pro Impuls) → steht auf dem Gaszähler
define("K_TICK_M3", 0.01);

// Brennwert des Gases → aus der letzten Gas-Abrechnung
define ("BRENNWERT_KWH_PER_M3", 11.54);
?>
```

Datenbank initialisieren

Man ruft im Browser <https://www.MeinGasÜberwachungsServer.de/dbinit.php> auf. Wobei www.MeinGasÜberwachungsServer.de natürlich mit der URL (oder der IP) und dem Pfad des tatsächlichen Servers ersetzt wird. Diese .php braucht nur einmal bei Projektstart aufgerufen werden, die legt die erforderliche Tabelle in der Datenbank an. Der Browser sollte das Anlegen der Tabelle bestätigen. In der Datenbank gibt es jetzt eine Tabelle gas_logger mit den entsprechenden Spalten.

Betrieb

Man ruft im Browser <https://www.MeinGasÜberwachungsServer.de/auswertung.php> auf. Wobei www.MeinGasÜberwachungsServer.de natürlich wieder mit der URL und dem Pfad des tatsächlichen Servers ersetzt wird.

Dann wird man nach einem Passwort gefragt, das ist das Passwort aus der config.php (s.o.). Hat man das Passwort richtig eingegeben, wird ein Cookie gesetzt, dass man sich nicht wieder einloggen muss beim nächsten Besuch. Der Cookie speichert nur den Login, nichts anderes. Er ist 30 Tage gültig und erneuert sich, wenn man innerhalb der 30 Tage die Seite besucht. Ausloggen ist möglich über einen Button auf der Seite ganz oben.

Dort findet man grafisch aufbereitet die Leistungs- und Temperaturkurven für einen Tag, eine Woche, einen Monat und ein Jahr, jeweils mit MIN und MAX-Werten.

Der Zählerstand wird errechnet aus dem Startzählerstand der config.php und den vom ESP gezählten Impulsen. Er sollte immer (ggf. mit geringen Abweichungen) mit dem tatsächlichen Zählerstand übereinstimmen, dann weiß man, dass das System sauber läuft.

Darunter werden zwei Punktwolken abgebildet:

- Heizkurvenanalyse: Vorlauftemperatur über Außentemperatur und
- Leistung über Außentemperatur

Die Prognosen unter den Grafiken sind mit viel Vorsicht zu genießen, hier wird auf -10°C als niedrigste zu erwartende Außentemperatur aus den Punktwolken extrapoliert. **Die Formel ist experimentell und nicht getestet.**

Interpretation der Ergebnisse

Ich habe dieses Projekt gestartet, um in der Planung einer Wärmepumpe der theoretischen Heizlastberechnung reale Messwerte zur Seite zu stellen und zu verstehen, wie meine Heizung aktuell arbeitet.

Folgendes erwarte ich in den Ergebnissen zu sehen:

- Die typische, abgefragte Leistung bei sehr niedrigen Außentemperaturen gibt ein Hinweis auf die erforderliche Nennleistung. Ich gehe davon aus, dass das nicht 1:1 übertragbar ist und es Ausreißer nach oben geben könnte. Aber als grober Anhaltswert (Kernfrage: „Wie hoch läuft die Therme im Moment bei niedrigen Außentemperaturen“) mit Sicherheit nicht schlecht.
- Die maximale Vorlauftemperatur bei niedrigen Außentemperaturen. Diese sollte für den Umstieg auf eine Wärmepumpe nicht über 55°C liegen, damit eine Wärmepumpe effektiv arbeiten kann. Man kann die erforderliche Vorlauftemperatur ggf. durch Heizkörpertausch gegen größere Heizkörper senken.
- Eventuell schon jetzt Optimierungsbedarf abschätzen, z.B. aus dem Spread zwischen Vor- und Rücklauf, Taktungen etc.

Privatsphäre

Es wird lediglich das Login-Cookie gesetzt, keine Drittanbietercookies oder irgendwelche anderen Schweinereien.

Grenzen und bekannte Probleme

- Die Prognosen unter der Auswertung sind experimentell und noch nicht getestet. Bitte nicht darauf verlassen!
- Warmwasserbereitung durch die Gastherme wird nicht berücksichtigt. Ich kann das auch nicht testen, da ich mein Warmwasser nicht über die Gastherme erzeuge. Wem das wichtig ist, kann den Code gerne anpassen. Dies wäre der Weg: Bei Warmwasserbereitung läuft die Gastherme mit sehr hoher Leistung, die Vorlauftemperatur des Heizkreises steigt aber nicht. Deshalb ist der Einbauort des Vorlauffühlers wichtig, er darf nicht im Warmwasser-Heizkreis hängen, sondern muss nach dem Dreiwegeventil im Heizungs-Heizkreis installiert werden. Man filtert in der Auswertung dann alle Datenpunkte mit hoher Leistung aber nicht stark steigender Vorlauftemperatur heraus. Vielleicht mache ich mal ein Update, vielleicht aber auch nicht.
- Der Wirkungsgrad der Therme wird nicht berücksichtigt. Wem das wichtig ist, kann in der config.php die Zeile `define ("BRENNWERT_KWH_PER_M3", 11.54)` anpassen.
- Es wird grundsätzlich mit Mitteleuropäischer Winterzeit gearbeitet.
- **Nur rudimentärer Passwortschutz! Vorsicht: Heizungsdaten sind kritische Nutzerdaten! Man kann aus den Heizkurven ggf. herauslesen, ob und wann die Wohnung genutzt ist.** Der Anwender sollte selber hier ggf. Vorkehrungen treffen, dass die Daten nicht öffentlich erreichbar sind.

- Ich bin sicher, der Code ist nicht überall ganz sauber programmiert. Er war zunächst nur als privates Projekt geplant, ist dann aber über die Zeit stark gewachsen und war nie zur Veröffentlichung vorgesehen. Erst im Nachgang habe ich mir gedacht, er könnte auch für andere Menschen wie mich, die ihre Heizung besser kennenlernen wollen interessant sein. Ich bitte alle Unsauberkeiten im Code zu entschuldigen.

- Für mich macht der Code, was er soll. Ein paar Tests stehen noch aus, aber für mich ist das Projekt ansonsten abgeschlossen. Ich mache keinen Support und verspreche auch keine Updates. Der Code ist Open Source, entwickelt den gerne weiter.

MIT-Lizenz

Der Code und alle Teile des Projektes stehen unter MIT-Lizenz. Das bedeutet:

- **Freie Nutzung Jeder darf die Software verwenden, kopieren und ausführen — privat wie kommerziell.**
- **Freie Veränderung** Der Code darf verändert, erweitert und in eigenen Projekten weiterverwendet werden.
- **Freie Weitergabe** Die Software darf weitergegeben oder veröffentlicht werden, auch in veränderter Form.
- **Kommerzielle Nutzung erlaubt** Die Software darf in kommerziellen Produkten oder Dienstleistungen eingesetzt werden.
- **Namensnennung erforderlich** In allen Kopien oder wesentlichen Teilen muss der Copyright-Hinweis erhalten bleiben.
- **Keine Garantie** Die Software wird „wie sie ist“ bereitgestellt — ohne Zusicherung von Funktion, Qualität oder Eignung.
- **Keine Haftung** Der Autor haftet nicht für Schäden, Ansprüche oder sonstige Konsequenzen, die aus der Nutzung entstehen.