Abbildung 1: Gaszähler Lesekopf V1.4

Gaszähler auslesen
Daten gegen Verlust absichern

Anders als Smart Meter zum Messen der elektrischen Arbeit speichern herkömmliche Gaszähler den gemessenen Verbrauch mechanisch in einem Zählwerk.

Ein direktes Auslesen des Verbrauchs ist etwa mit Hilfe einer Kamera und Bildanalyse möglich, der Aufwand aber recht hoch.

Viele Gaszähler geben einen magnetischen Impuls pro Umdrehung der 2. oder 3. Stelle hinter dem Komma, ab. Meistens ist das 1 Impuls pro 10 Liter (2. Stelle). In der Regel steht das auch auf dem Typenschild.

Das Zählen der Impulse erlaubt also, den Verbrauch zu ermitteln, allerdings nicht den absoluten Wert, sondern nur den Wert seit Beginn der Zählung.

Im Folgenden wird ein Lesekopf mit Speichermöglichkeit und Auswertung über einen ESP8266 Mikrocontroller beschrieben.

Hardware:

Um die Zählimpulse auch bei einem Stromausfall zu speichern, fiel die Wahl auf einen RTC CMOS Baustein, der auch als Zähler eingesetzt werden kann und 240 Bytes an RAM bietet:

PCF8583 von Philips

Der PCF8583 ist zwar recht alt aber immer noch in Stückzahlen im DIL8 Gehäuse zu haben.
Der Baustein hat ein I²C Interface und kann einfach über einen ESP8266 ausgelesen werden, der dann auch die weitere Verarbeitung übernimmt.
Mit einer Stromaufnahme von typisch 10µA bei 5 Volt und SCL inaktiv reicht eine CR2032 3-Volt Knopfzelle für lange Zeit. Das Auslesen über I²C ist bis hinunter zu 2,5 Volt Betriebsspannung möglich, der interne Zähler funktioniert laut Datenblatt sogar bis zu 1 Volt.
Für die Erfassung der Zählimpulse kann entweder ein Reed Schalter oder ein Hall Sensor verwendet werden. Da aber Reed Schalter zum Prellen neigen und aufwendig hardwaremäßig entprellt werden müssten, fiel die Wahl auf einen Hall Sensor. Auch hier sind die Anforderungen niedrige Stromaufnahme und Funktion bis herunter zu 2,5 Volt Betriebsspannung.

AH3661UA 2.4 - 5.4 V/DC Messbereich: -0.007 - +0.007

Mit einer Stromaufnahme von typisch 2µA im Ruhezustand bei 2,4 Volt und einem pull-up Widerstand von 0,5 bis 1 MOhm sind die Anforderungen voll erfüllt. Zusammen mit recht hochohmigen pull-up Widerständen für SDA und SCL war die gemessene Stromaufnahme bei der Musterplatine zwischen 8 und 14µA.

Eine kleine Platine mit Standard Bauteilen kann so in die Aufnahme unterhalb des Zählwerkes gesteckt werden, dass der Sensor genau unter der Stelle mit dem Magneten positioniert ist.

Zum Auslesen der Daten wird ein ESP8266 verwendet, der über das lokale WiFi Netzwerk die Daten zur Auswertung weiterreicht.

Software:

Es liegt nahe, den Speicher des RTC Bausteins für mehr als das Abspeichern der gezählten Impulse zu nutzen, auf diese Weise kann man die Daten der Abrechnungsperiode nachhalten.
Für die Programmierung des ESP8266 habe ich die Arduino IDE 2.1.1 und die Bibliotheken Wire, time, Esp8266WiFi, PicoMQTT und Thingspeak verwendet, alle lassen sich über die Bibliotheksverwaltung herunterladen, soweit nicht schon vorhanden

Hier ist das Speichermodell für das CMOS RAM:

```
ADDRESS_OLD_COUNTER     // hält den letzten Stand der Auslesung mit I²C
ADDRESS_OLD_TIME        // Uhrzeit der letzten Auslesung mit I²C
ADDRESS_OLD_DATE        // Datum der letzten Auslesung mit I²C
ADDRESS_SAVE_COUNT      // Impulse seit der letzten Übermittlung
ADDRESS_START_METER     // Zählerstand beim Start mit Counts = 0
ADDRESS_START_PERIOD    // Zählerstand zum Beginn der Abrechungsperiode
ADDRESS_MONTH_01        // Zählerstand zum Beginn des ersten Monats
                           weitere 11 Adressen bis Monat 12
```

Die Monatswerte sind besonders interessant, weil man damit überprüfen kann, ob der monatliche Abschlag noch ausreichend oder zu hoch ist, im Vergleich mit den Vorjahreswerten kann man auch den Erfolg von Energiesparmaßnahmen beurteilen.

Die Werte für Start_Meter und Start_Period können über MQTT gesetzt werden, ein Senden von Start_Meter setzt den Zähler auf Null, so können Zähler und RTC synchronisiert werden.

Die Daten werden über MQTT zur weiteren Auswertung / Anzeige und Thingspeak zur einfachen Visualisierung weitergereicht.

MQTT:

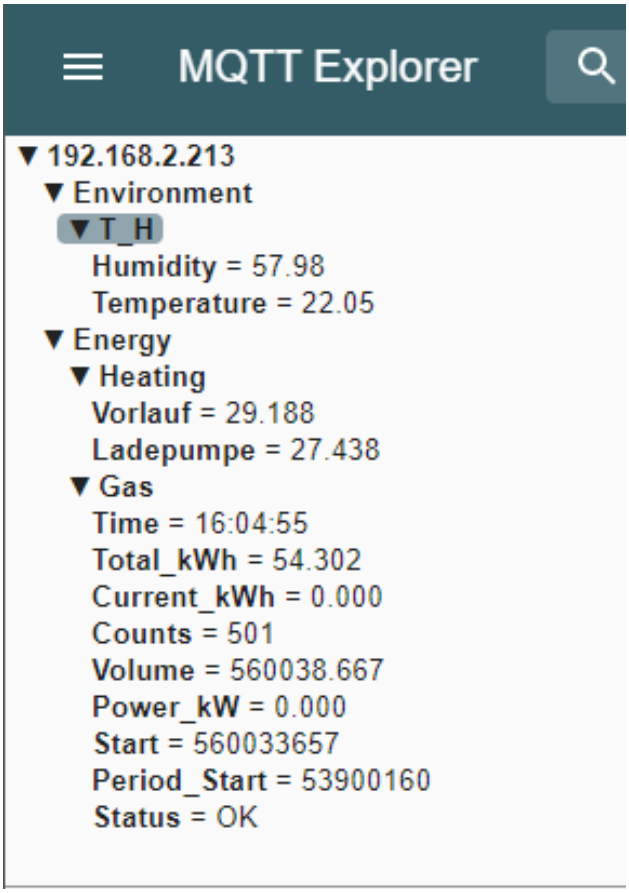


Abbildung 2: MQTT Topics

Thingspeak:

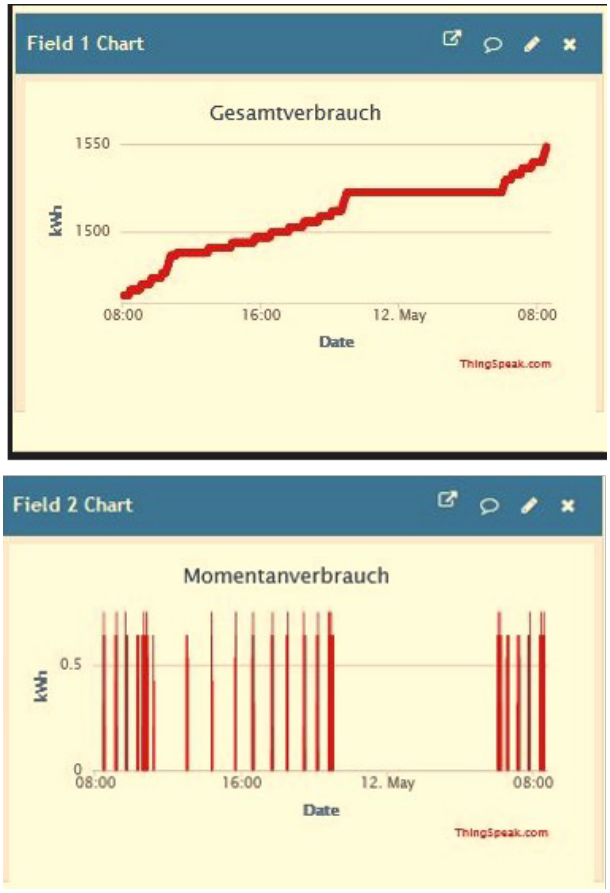
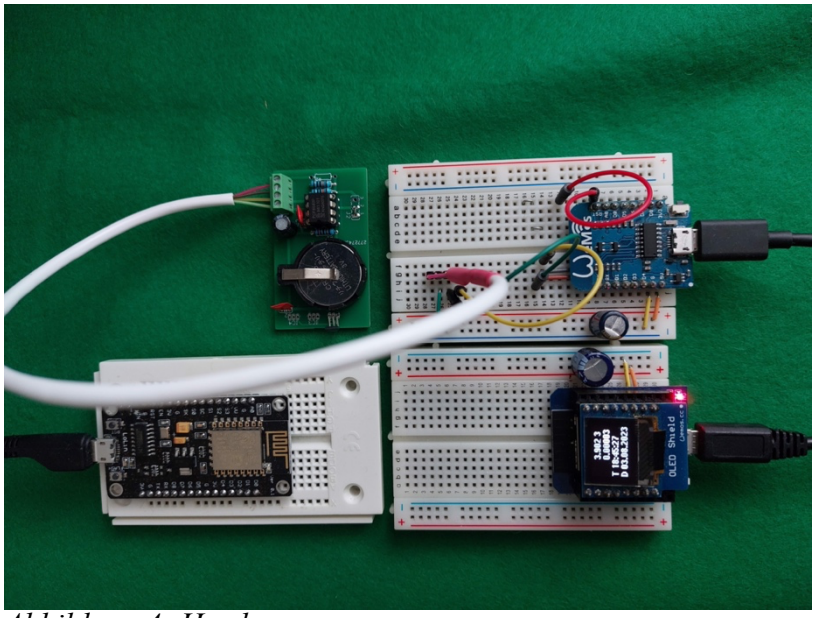


Abbildung 3: Thingspeak Grafiken

Die beteiligten Module:



Oben rechts der ESP8266 mit dem Lesekopf, unten links ein weiterer ESP8266 als MQTT Broker.

Daneben ein Mini „Dashboard“ zur Anzeige zweier Messwerte sowie Datum und Uhrzeit der Ablesung.

Ziel ist es, die Daten aller Sensoren und Leseköpfe in einem großen Dashboard zu vereinen.

Abbildung 4: Hardware

Der „Einbau“:

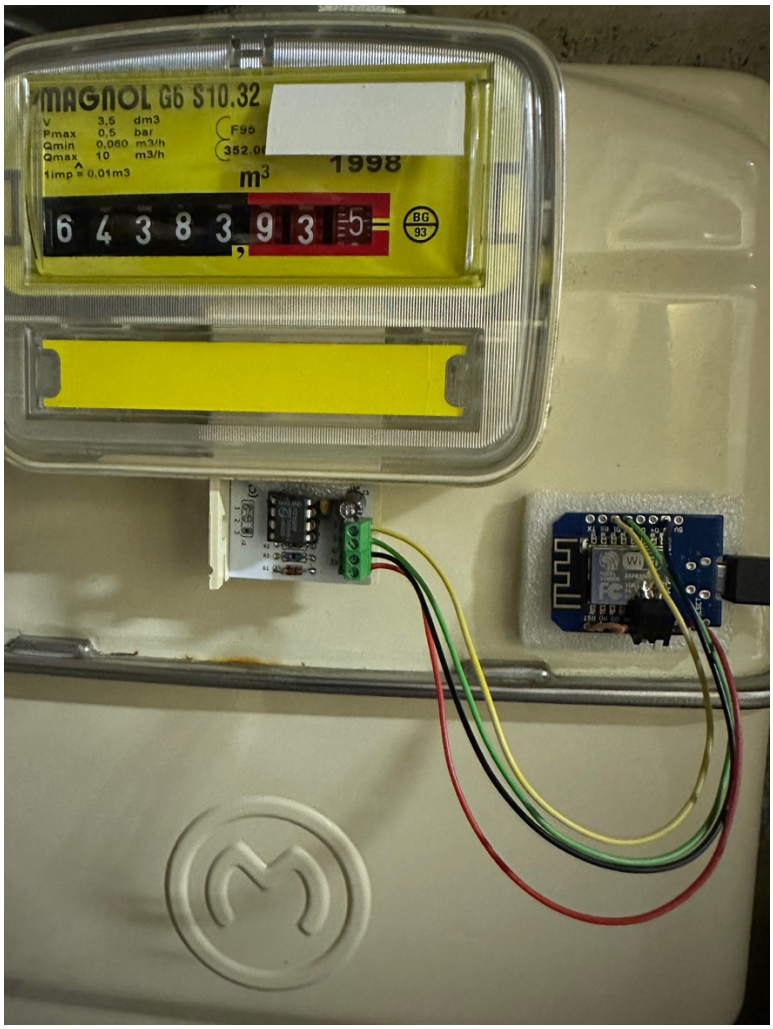


Abbildung 5: Einbau

Der ESP wird aus einem kleinen Netzteil versorgt, zum Stromsparen in den Deep Sleep Modus versetzt und alle 2 Minuten aufgeweckt. Dabei versorgt der ESP auch den Sensor, bei Stromausfall übernimmt die Knopfzelle die Versorgung.

Der Mikrocontroller hat keine weiteren Bedienelemente.
Eine Brücke von Reset auf D0 ist notwendig, um das Aufwachen nach dem Deep Sleep zu ermöglichen, eine Brücke von D7 nach GND ruft beim nächsten Boot bzw. nächsten Loop die Init Routine für den RTC Baustein auf. Dies ist nur bei der ersten Inbetriebnahme oder beim Ausfall der Versorgung über die Knopfzelle notwendig. MQTT meldet in diesem Fall einen Fehler, da dann keine Kommunikation mit dem RTC mehr möglich ist.

Eine weitere Brücke von D6 nach GND verhindert den Deep Sleep, so dass über MQTT die Startwerte für die Abrechnungsperiode und der Startwert für den Verbrauch nach Rücksetzen des Zählers gesendet werden können.
In meinem Setup sind dies die Topics Energy/Gas/InitS für den Startwert Verbrauch und Energy/Gas/InitP für den Startwert der Abrechnungsperiode, jeweils als ganzzahliger Wert in Litern.

Beide Brücken für D6 und D7 sind für den Normalbetrieb nicht notwendig.

Schlussbemerkung:

Der Aufbau mit dem RTC Baustein und dem ESP8266 läuft seit langer Zeit ohne Probleme, auch das Abstecken des Controllers für längere Zeit hat nicht zum Datenverlust geführt, nur die zeitliche Zuordnung der Impulse geht verloren, diese werden der Periode zwischen Abstecken und Wiederanschließen zugeordnet. Wenn diese Zeit nicht über den Monatswechsel hinausgeht, ist die Zurechnung zum „richtigen“ Monat gegeben.

Was gibt es noch zu tun bzw. zu verbessern:

- Um die Daten langfristig lokal abzuspeichern, bietet sich ein EEPROM an.
- Ein Gehäuse für den Lesekopf, welches dann in die Öffnung unterhalb des Zählwerkes passt, würde das Ganze professioneller machen.
- Statt den Verbrauch in kWh pro Tag oder Monat wäre eine Grafik mit Euro pro Stunde auch sinnvoll, das würde die Kosten der Wärme in der Wohnung immer sichtbar machen und die Wirkung von Energiesparmaßnahmen könnte in Verbindung von Außen- und Innentemperatur besser beurteilt werden.