Smarthome-Hacks aus der Praxis

Pi and More 11: 03.11.2018

Smarthome-Projekte mit dem Pi und ESP-Microcontrollern für ein smartes Zuhause

Über mich

- Tobias Blum
- Aus der Nähe von Saarbrücken
- Studium: Angewandte Informatik in Zweibrücken
- Hauptberuflich Webentwickler bei netmedia.de

- Vortrag auf der Pi and More 9:
  „Ein SmartLight im Selbstbau für unter 15 €“
  https://github.com/toblum/McLighting/
- Vortrag auf der Pi and More 10:
  „Software Defined Radio – mit dem Pi per ADS-B den Luftraum in der Großregion beobachten“

Übersicht

- Projekt 1: Erfassung des Wasserverbrauchs per Bilderkennung mit dem Pi und OpenCV
- Projekt 2: Erfassung des Gasverbrauchs per Reed-Sensor
- Projekt 3: Energiesparen durch smarte Umwälzpumpe
- Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi
- Projekt 5: Smarthome-Report mit ePaper-Display und ESP32 Microcontroller

Projekt 1: Wasserverbrauch

Projekt 1: Erfassung des Wasserverbrauchs per Bilderkennung mit dem Pi und OpenCV

- Warum?
  - Ungewöhnlich hoher Verbrauch
  - Wasserzähler zählen evtl. nicht immer zuverlässig
  - Erfassen von „Verbrauchssprüngen“

Hardware

Projekt 1: Erfassung des Wasserverbrauchs per Bilderkennung mit dem Pi und OpenCV

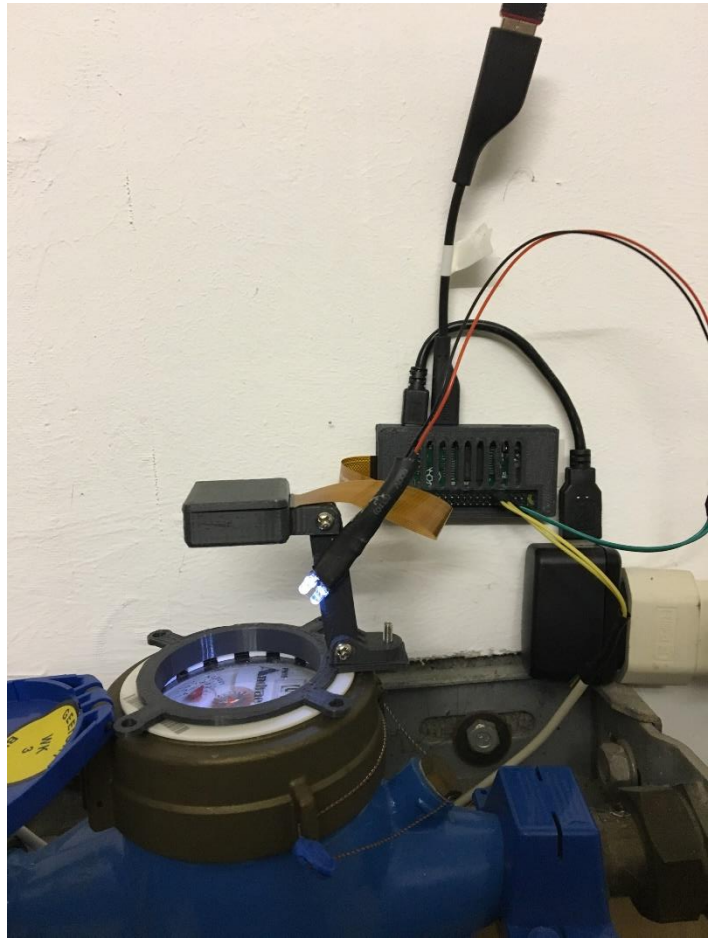
- Wie?
  - Klassische Wasserzähler funktionieren rein mechanisch und haben nur selten eine Möglichkeit, Messwerte „abzugreifen“
  - Kein Signalausgang, keine Rollenmagnete verbaut
  - Letzte Möglichkeit: Optische Erfassung von Rollenanzeige oder Zeigern
  - Problem:
    - Erkennung der Zahlen unzuverlässig und grob (1 m³ == 1000 l)
    - Erfassung der Zeigerbewegung: Bauform ungeeignet für Lichtschranke
  - Lösung: „Abfilmen“ des Zählers und Erkennung der Zeigerposition

Projekt 1: Erfassung des Wasserverbrauchs per Bilderkennung mit dem Pi und OpenCV

- Erfassen des Zählers per PiCamera
  - Sichere Positionierung der Kamera per Halterung
  - Einheitliche Beleuchtung mit 2 weißen LEDs, direkt vom P gespeist

Projekt 1: Erfassung des Wasserverbrauchs per Bilderkennung mit dem Pi und OpenCV

- Gesamtaufbau Hardware
  - Raspberry Pi Zero mit WLAN Dongle oder Pi Zero W
  - PiCamera
  - Rechenleistung von Pi Zero ausreichend, etwa 25 % CPU
  - Gehäuse: 3D-Druck



Projekt 1: Erfassung des Wasserverbrauchs per Bilderkennung mit dem Pi und OpenCV

- Software: Einfaches Python-Skript
- Endlosschleife:
  - Bild erfassen
  - Relevanten Bereich ausschneiden
  - Gauss-Filter, um Bildrauschen zu entfernen
  - In HSV-Farbbereich konvertieren
  - Alle Pixel in einem bestimmten roten Farbbereich weiß einfärben, den Rest schwarz
  - Weiße Pixel zählen, um zu erkennen, ob Zeiger im Ausschnitt

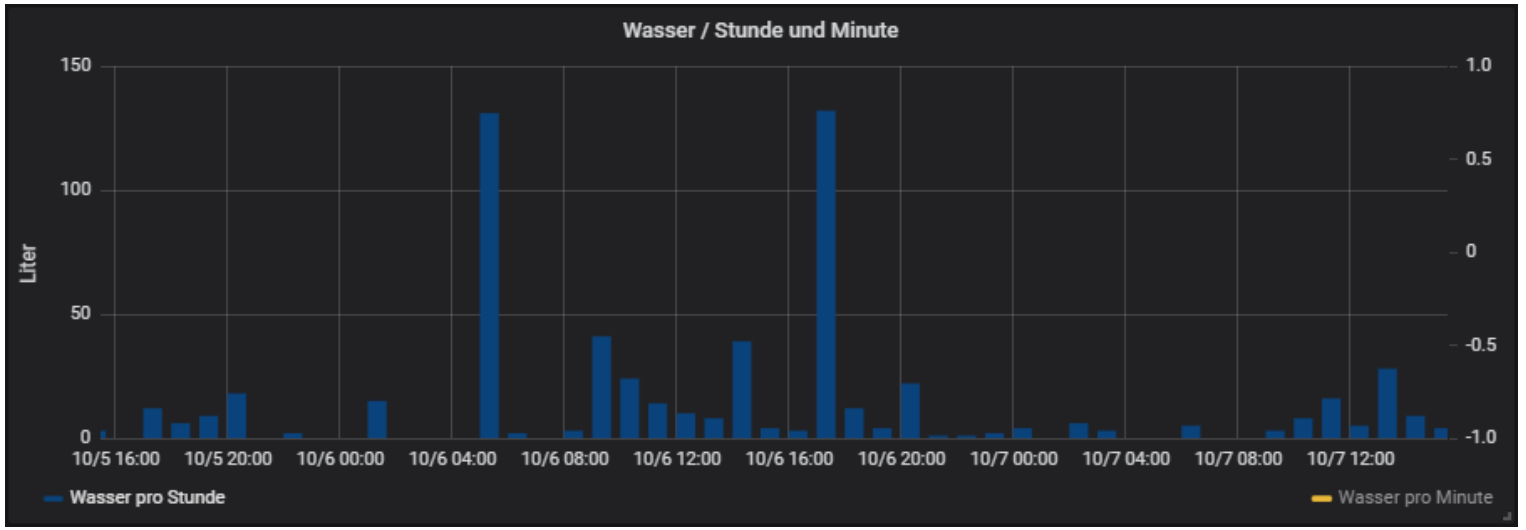
Projekt 1: Erfassung des Wasserverbrauchs per Bilderkennung mit dem Pi und OpenCV

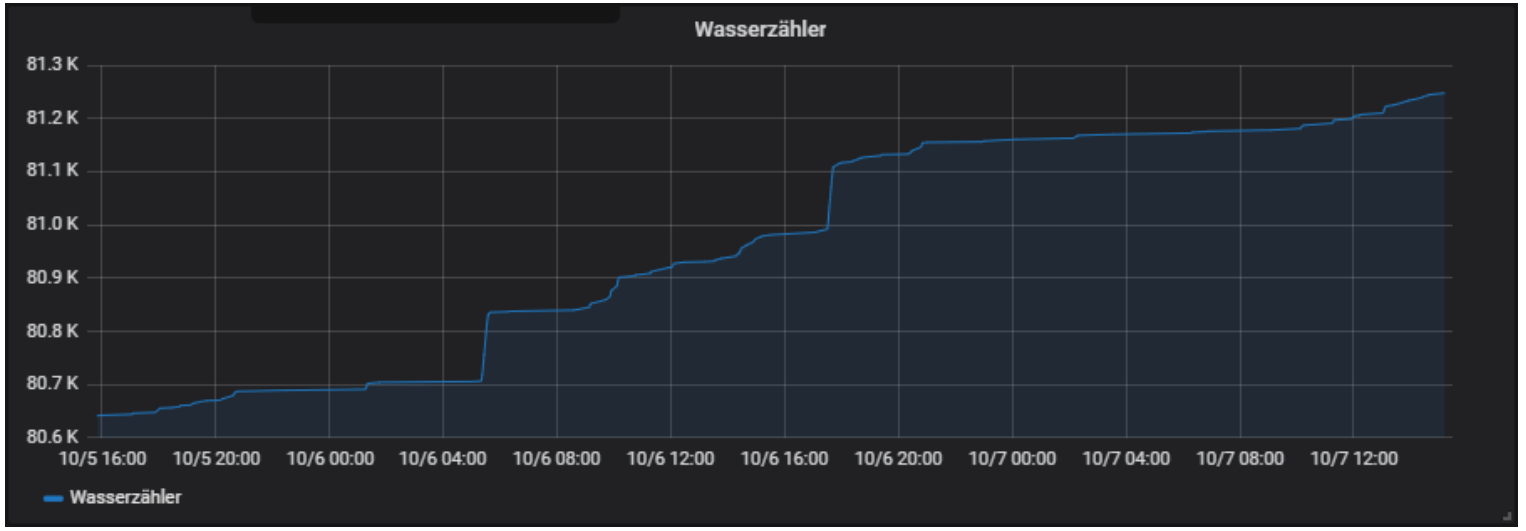
```
crop_img = img[start_y:start_y+crop_size, start_x:start_x+crop_size]  # crop the image to a 50 x 50 rectangle
crop_img=cv2.GaussianBlur(crop_img, (9,9), 9)       # blur the image
reduce=cv2.resize(crop_img, (40,30))                # reduce the size for faster processing

hsv = cv2.cvtColor(reduce, cv2.COLOR_BGR2HSV)       # change color scheme to HSV
red_lower=np.array([0,150,0],np.uint8)              # between this and
red_upper=np.array([190,255,255],np.uint8)          # this red
thresh = cv2.inRange(hsv, red_lower, red_upper)     # check for redness
hist = cv2.calcHist([thresh],[0],None,[1],[0,1])    # transform red to white, anything else to black

if hist[0] > 1120 and lasttrigger == 1:             # if we triggered and it's very black, trigger to 0
    lasttrigger=0
elif hist[0] < 1050 and lasttrigger == 0:           # if it's less black, trigger to 1
    lasttrigger=1
    liter_current = liter_current + 1               # Count 1 liter
    send_data(liter_current)

time.sleep(0.25)                                    # Sleep for 100 ms to reduce load
rval, frame = vc.read()                             # next frame
```

Projekt 1: Erfassung des Wasserverbrauchs per Bilderkennung mit dem Pi und OpenCV

- Datenspeicherung
  - Pi sendet für jeden erfassten Liter eine MQTT-Nachricht an ein definiertes Topic
  - Telegraf-Dienst überwacht das Topic und schreibt die Daten in eine InfluxDB (https://www.influxdata.com/)
  - Grafana zur Darstellung/Analyse (https://grafana.com/)

Projekt 1: Erfassung des Wasserverbrauchs per Bilderkennung mit dem Pi und OpenCV





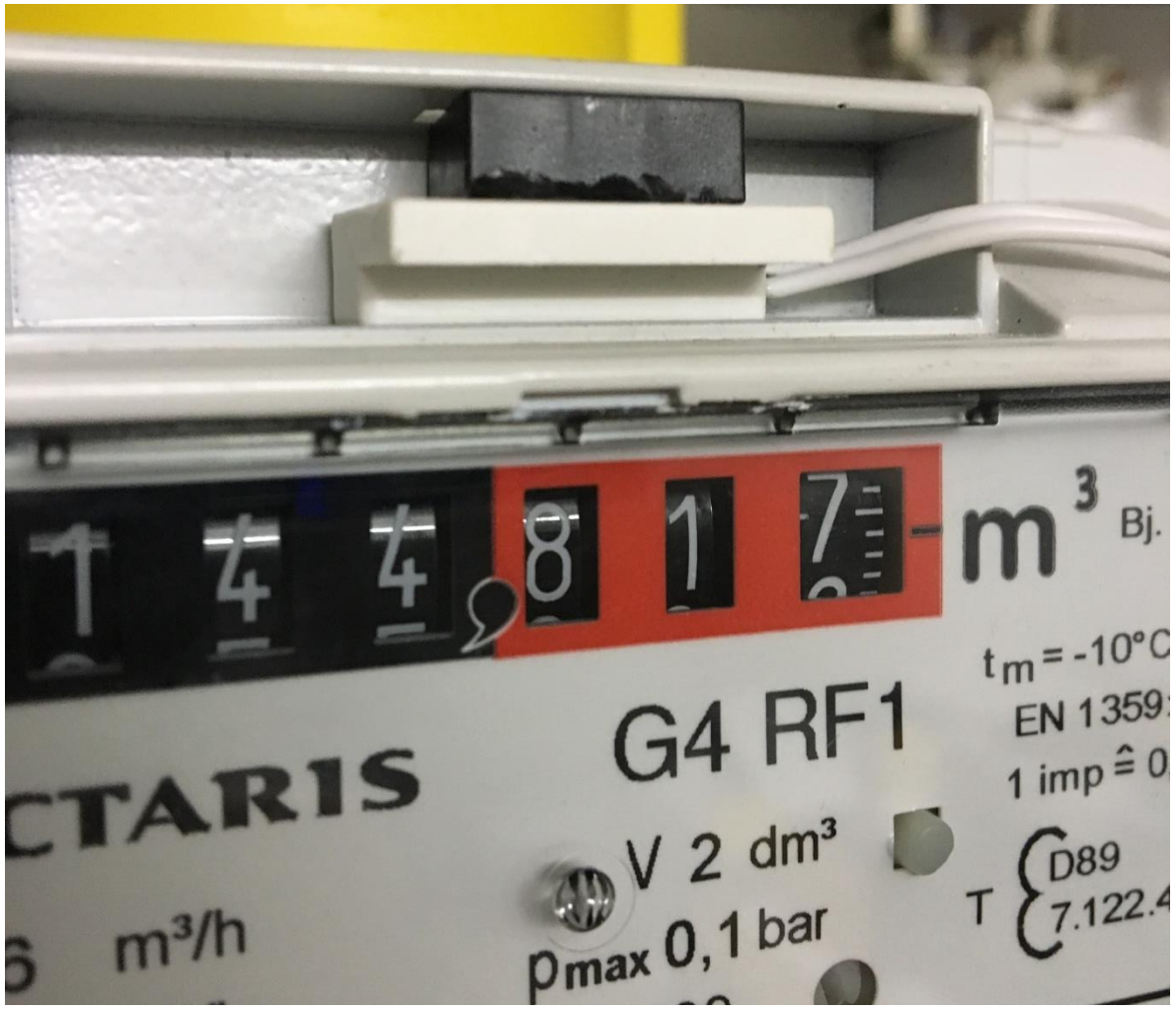
Projekt 2: Gasverbrauch

Projekt 2: Erfassung des Gasverbrauchs per Reed-Sensor

- Ziel:
  - Erfassung des Gasverbrauchs, um einen Überblick über die zu erwartenden Kosten zu erhalten
  - Identifizierung von Optimierungpotenzial
- Wie?
  - Der Gaszähler besitzt einen Steckplatz für einen Sensor
  - Die 0,1-Rolle besitzt einen Magneten, der abgetastet wird
  - „Offizieller“ Sensor schwer zu bekommen und teuer (> 50 €).
  - Leicht per Reed-Sensor auszulesen; Raspberry sowieso „in der Nähe“

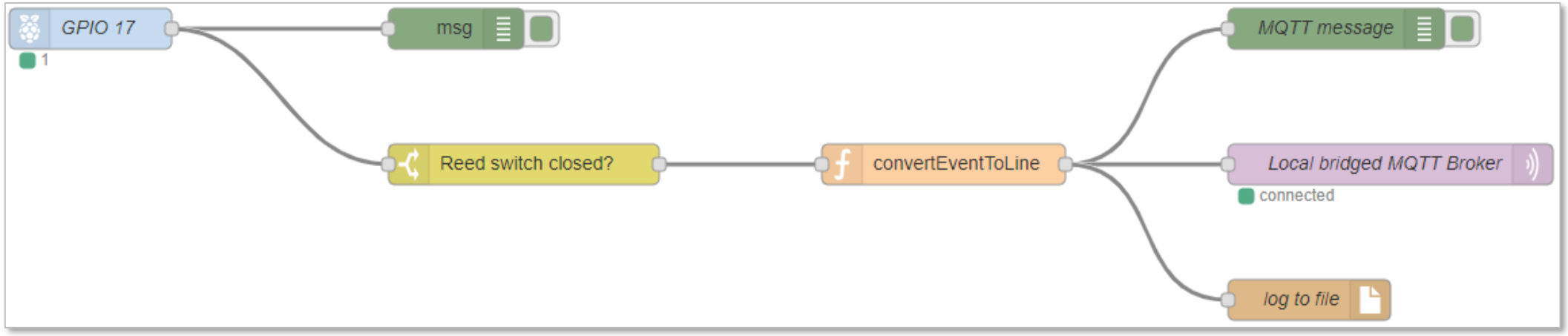
Projekt 2: Erfassung des Gasverbrauchs per Reed-Sensor

› Aufbau?

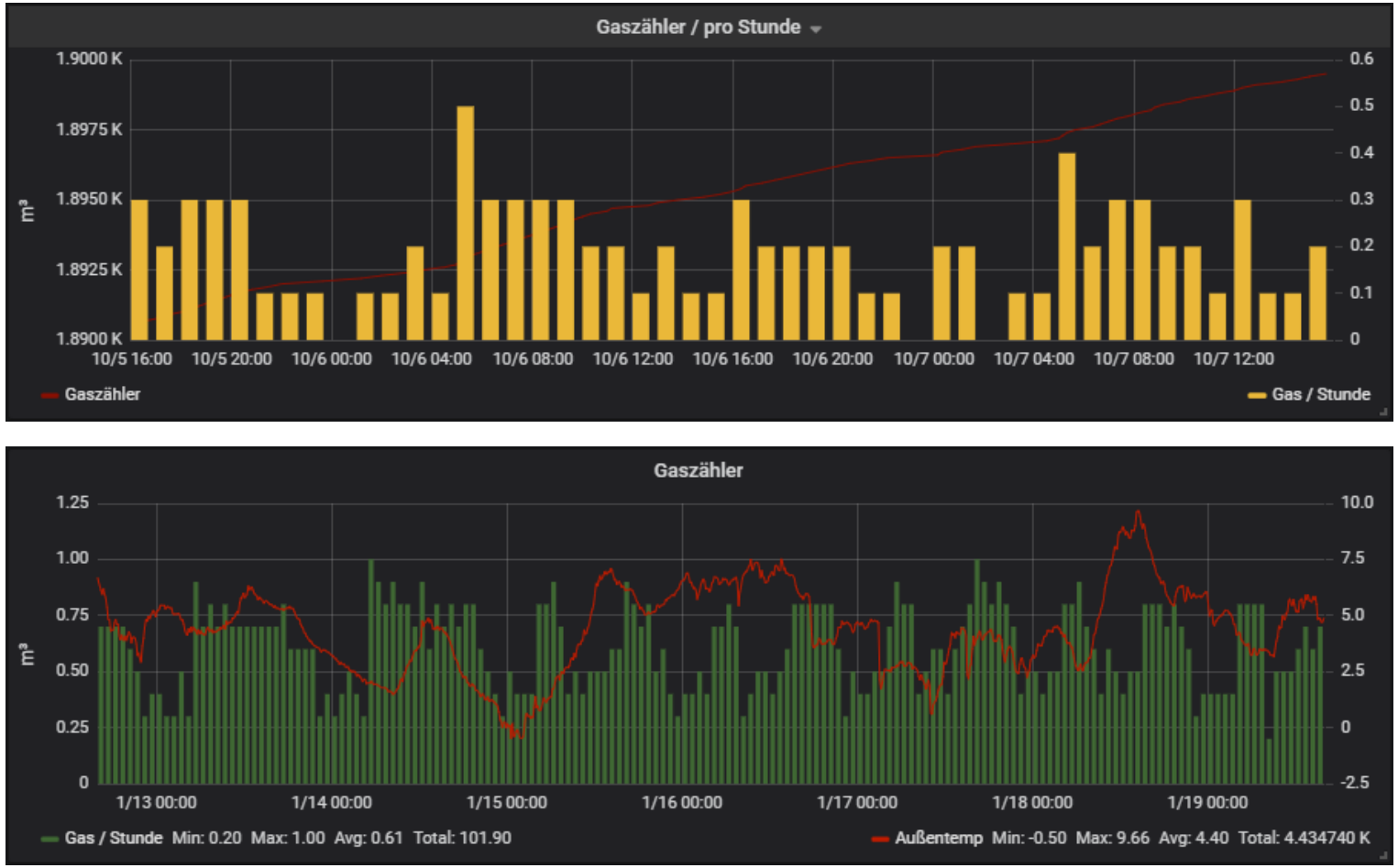


Projekt 2: Erfassung des Gasverbrauchs per Reed-Sensor

- Software
  - NodeRed läuft auf Pi Zero und überwacht GPIO
  - Wenn der Reed-Sensor schließt, wird MQTT-Message gesendet



Projekt 2: Erfassung des Gasverbrauchs per Reed-Sensor

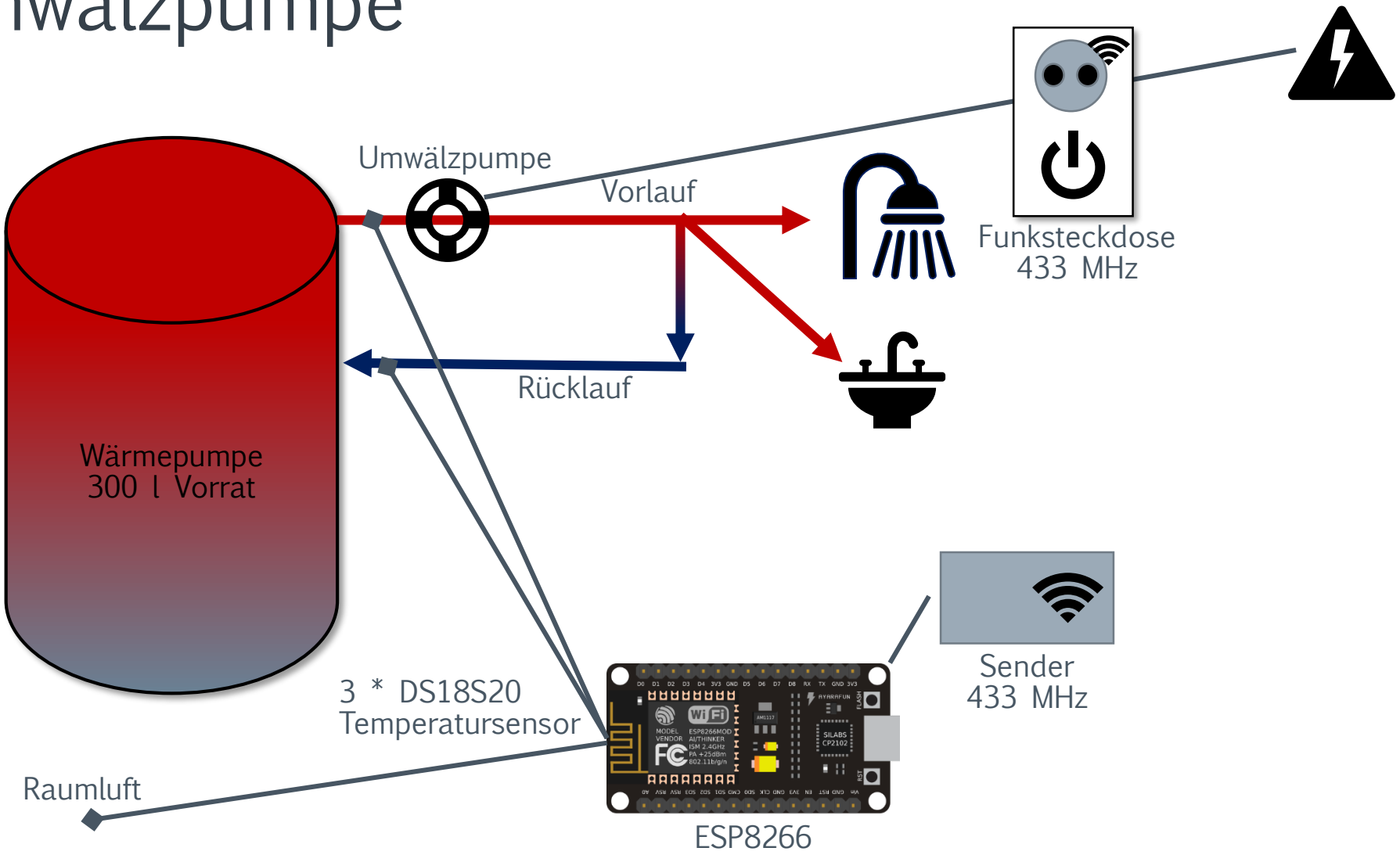


Projekt 3: Smarte Umwälzpumpe

Projekt 3: Energiesparen durch smarte Umwälzpumpe

- Ziel:
  - Die Warmwasser-Umwälzpumpe sorgt dafür, dass sofort warmes Wasser zur Verfügung steht und nicht zuerst abgekühltes Wasser aus den Leitungen kommt.
  - Komfortabel, aber teuer, da Wasser beim Zirkulieren abkühlt, neu erwärmt werden muss und Pumpe selbst Strom verbraucht.
  - Nur dann laufen lassen, wenn nötig.
- Wie?
  - Flexible und komfortabel anpassbare Zeitsteuerung
  - Bei Bedarf auf Knopfdruck zu starten

Projekt 3: Energiesparen durch smarte Umwälzpumpe

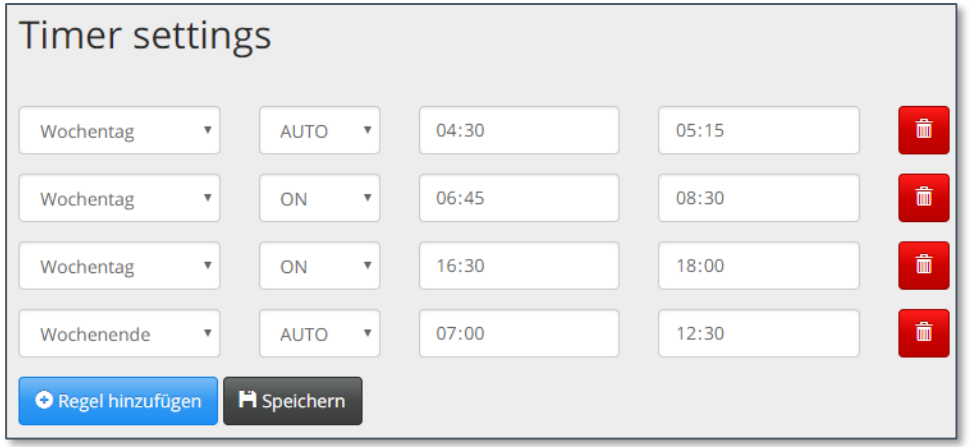


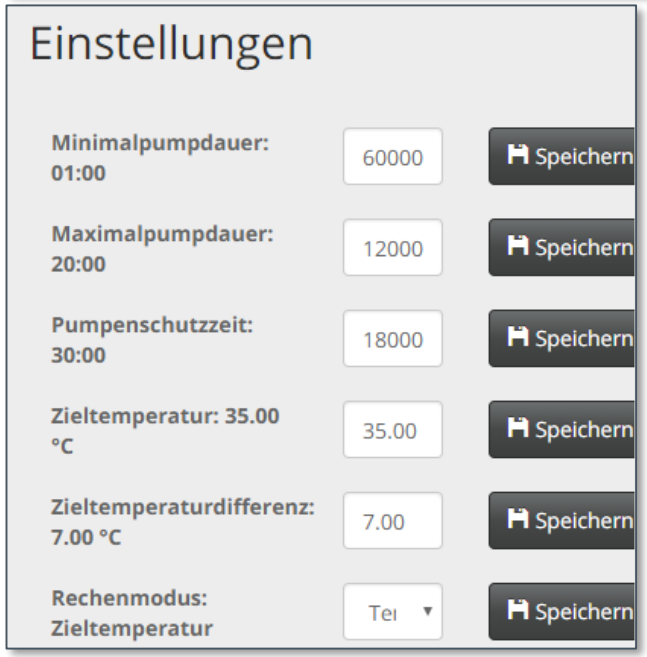
Projekt 3: Energiesparen durch smarte Umwälzpumpe

Projekt 3: Energiesparen durch smarte Umwälzpumpe

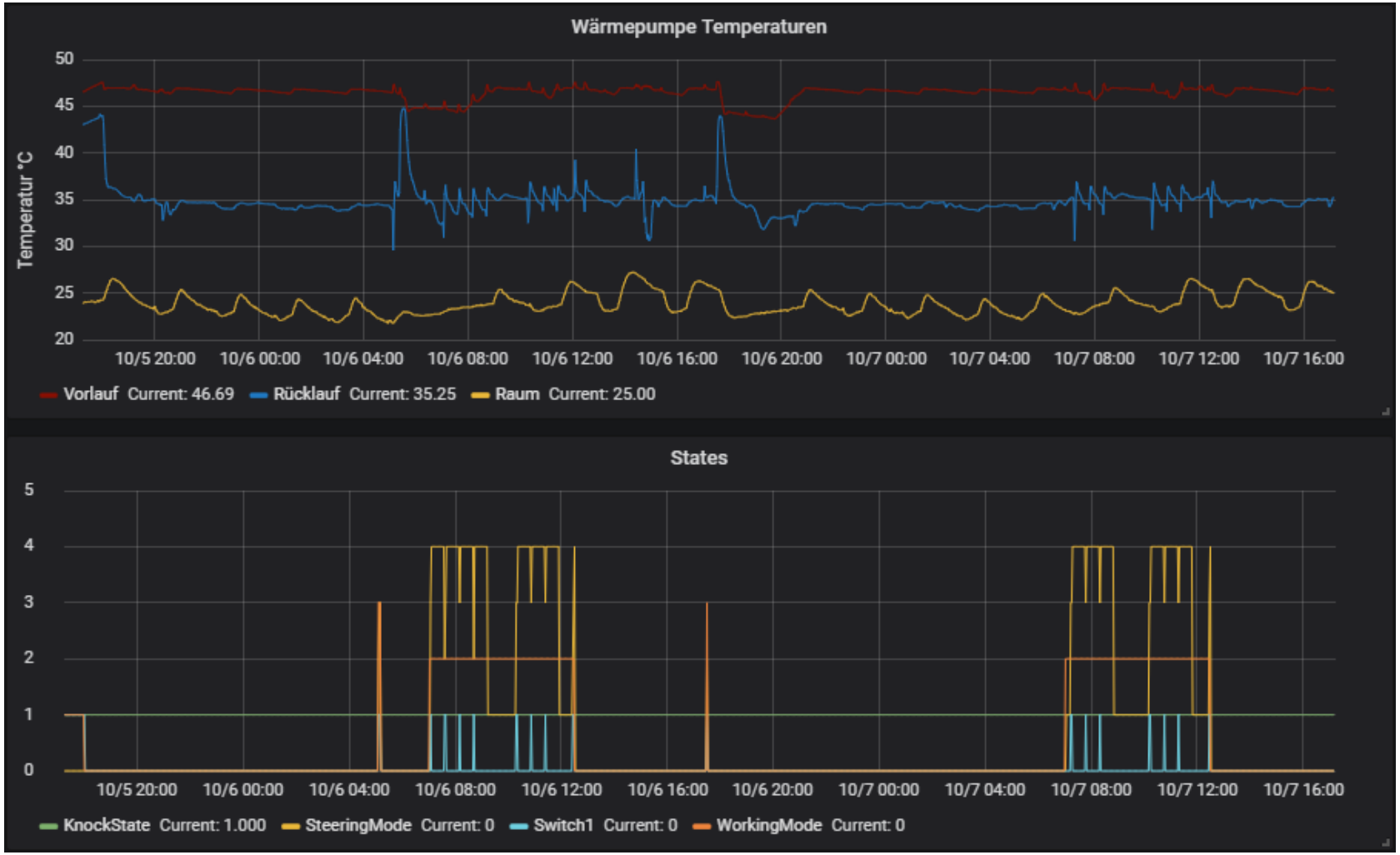
- Software:
  - Vier Betriebsmodi:
    - On/Off: Dauerhaft an/aus
    - Auto: Zieltemperatur Rücklauf in Bereich halten
    - Once: Einmalig Zieltemperatur im Rücklauf erreichen
  - Implementiert als Statemachine
    - 0: Grundmodus (z. B. wenn inaktiv)
    - 1: Warten auf Pumpenaktivierung
    - 2: Pumpe aktiviert, Minimalpumpdauer aktiv
    - 3: Pumpe aktiviert, warten bis Temperatur erreicht
    - 4: Pumpe deaktiviert, Schutzzeit bis nächste Aktivierung
  - Drei Temperatursensoren werden regelmäßig ausgelesen und Werte normalisiert
  - Aktuelle Werte auf OLED-Display ausgeben
  - Sonst keine Logik im Gerät
  - Wird per URL-Aufruf gesteuert. Liefert Status als JSON zurück
  - Webinterface läuft auf Pi

Projekt 3: Energiesparen durch smarte Umwälzpumpe

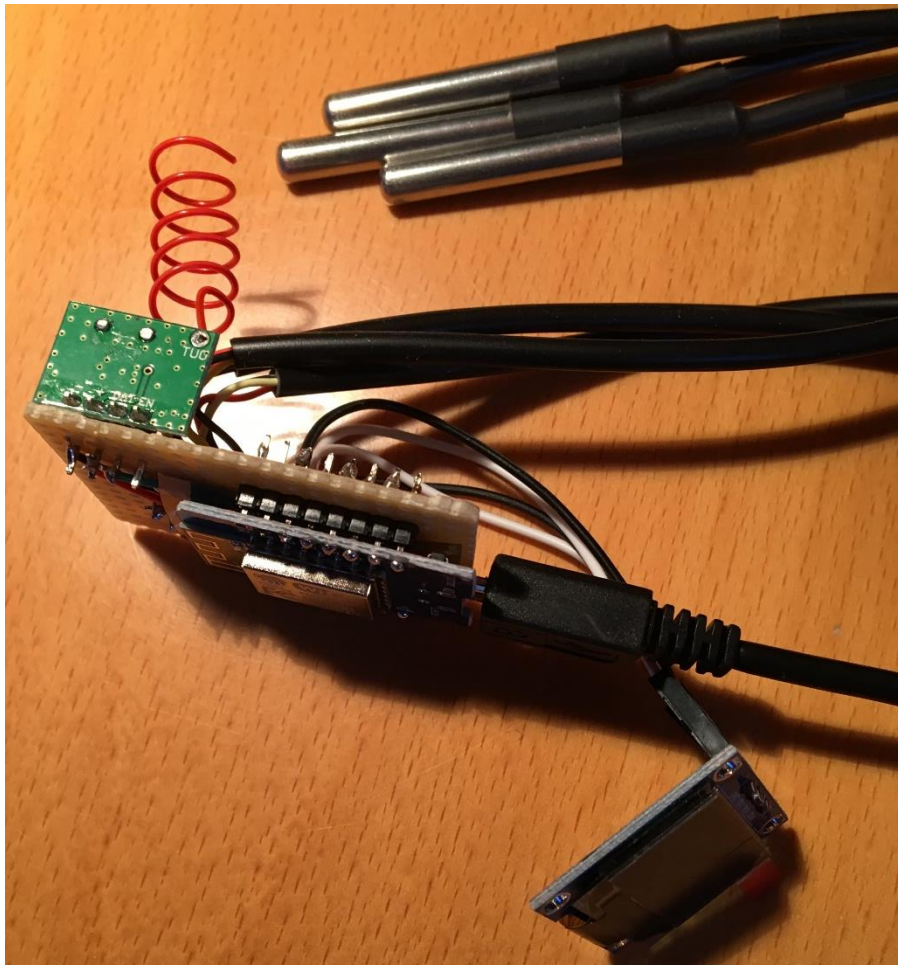


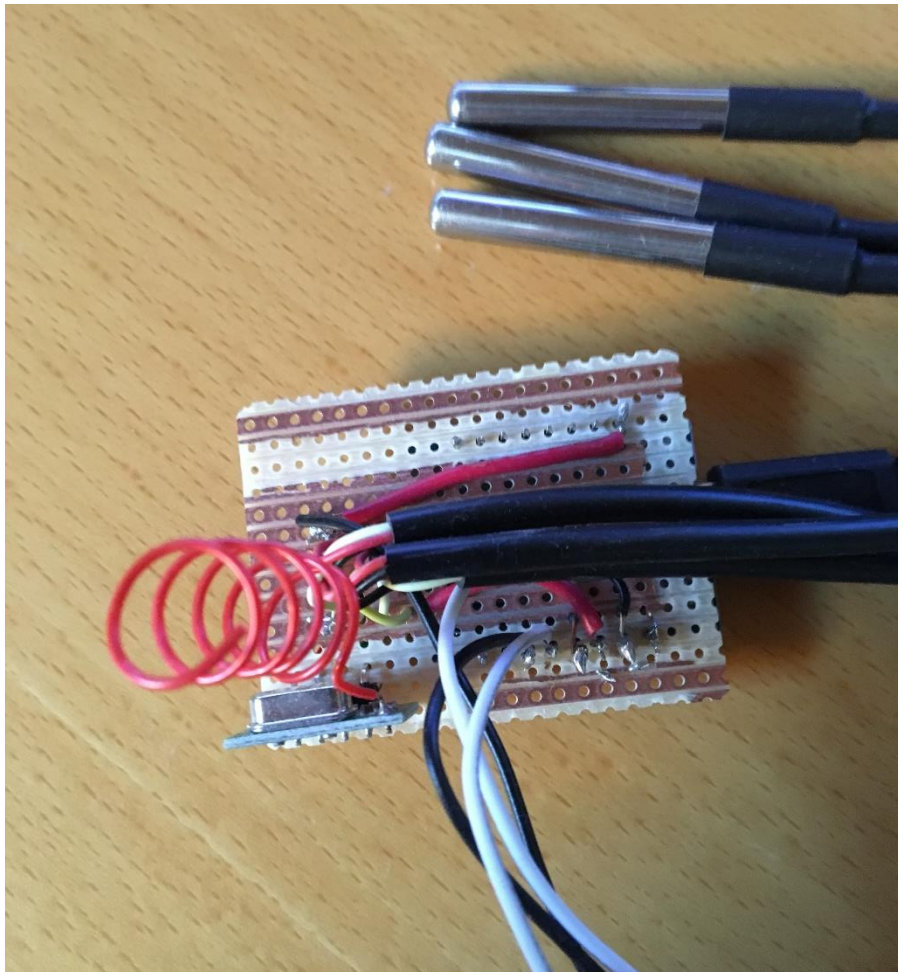


Projekt 3: Energiesparen durch smarte Umwälzpumpe

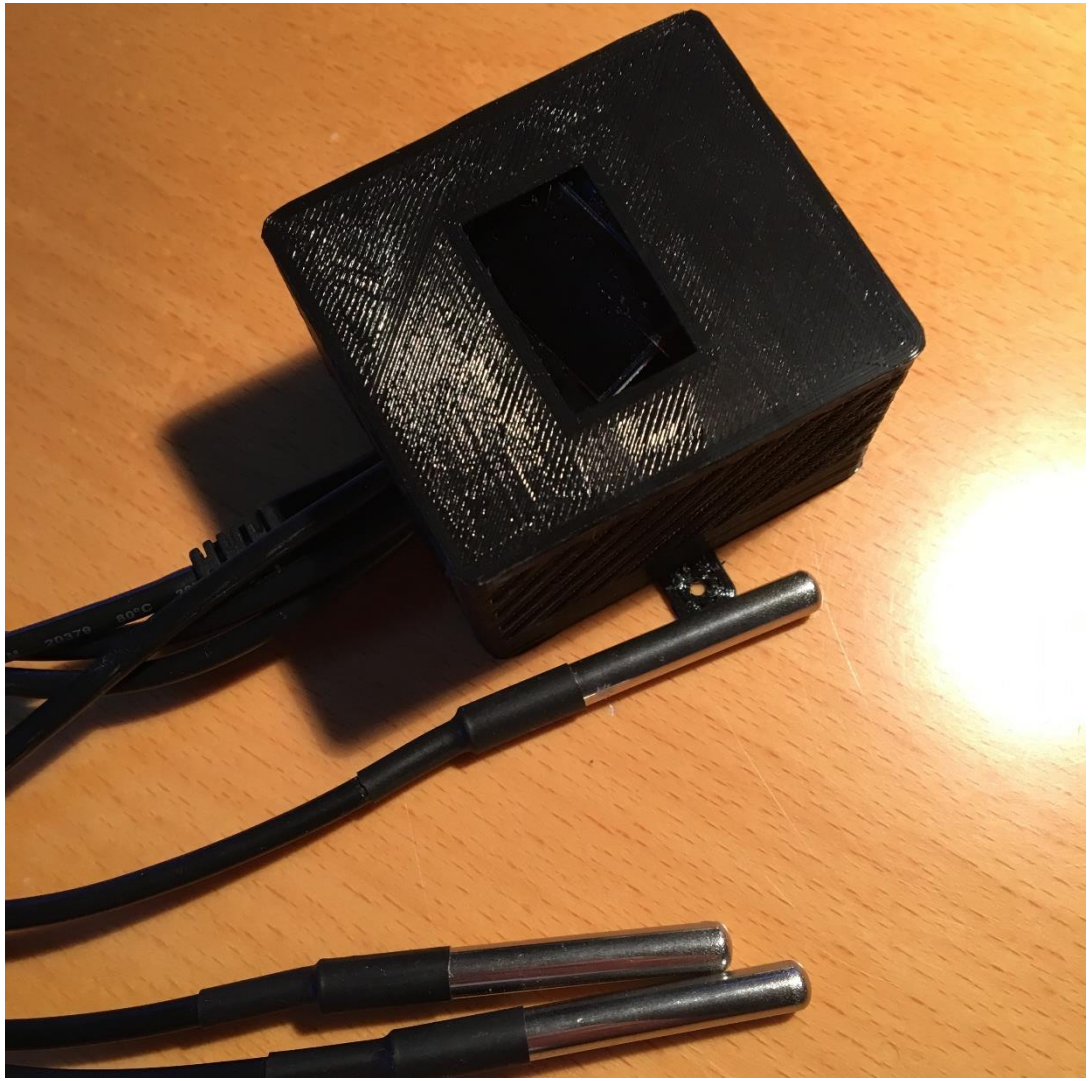


Projekt 3: Energiesparen durch smarte Umwälzpumpe





Projekt 3: Energiesparen durch smarte Umwälzpumpe



Projekt 4: Smartes Garagentor

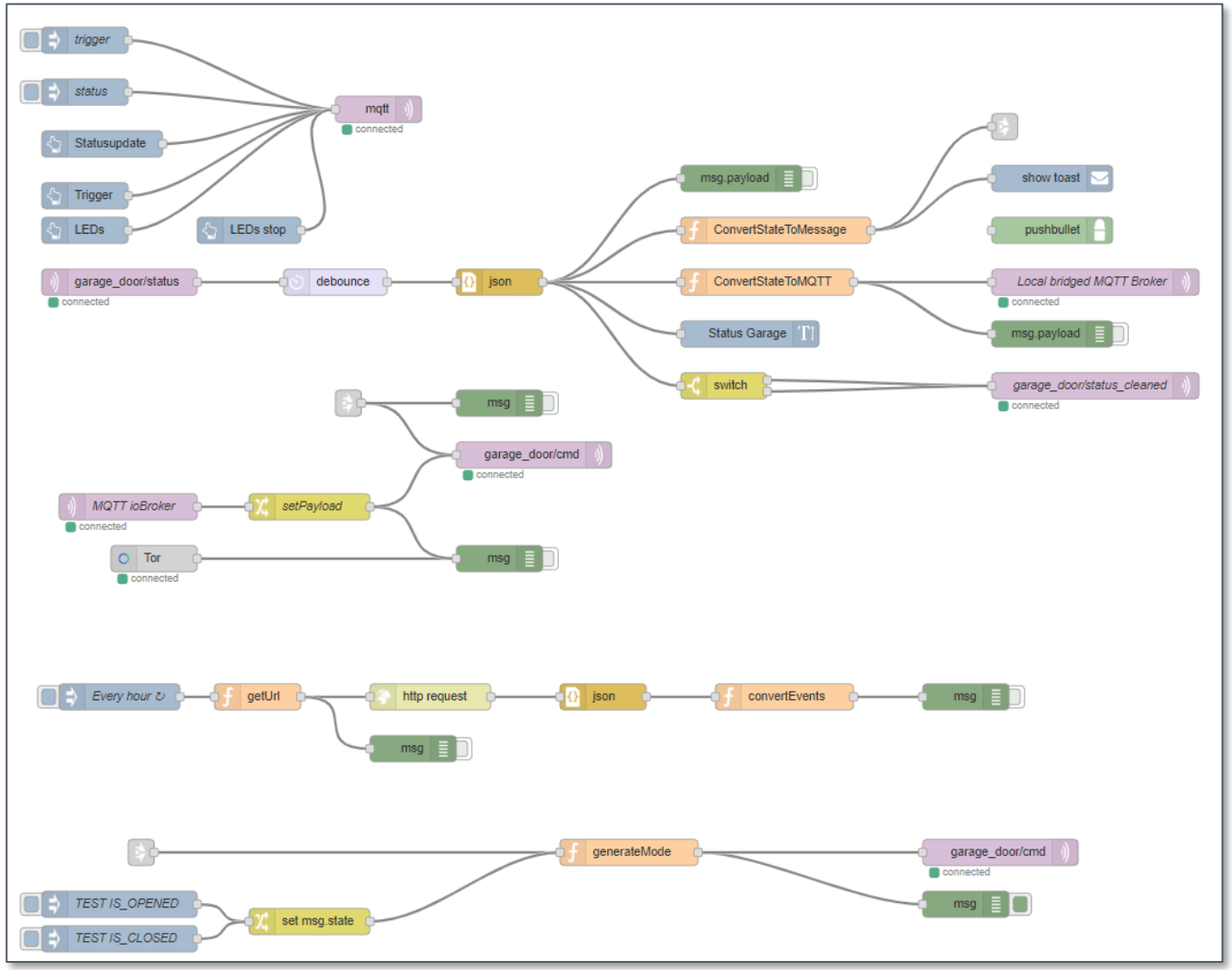
Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi

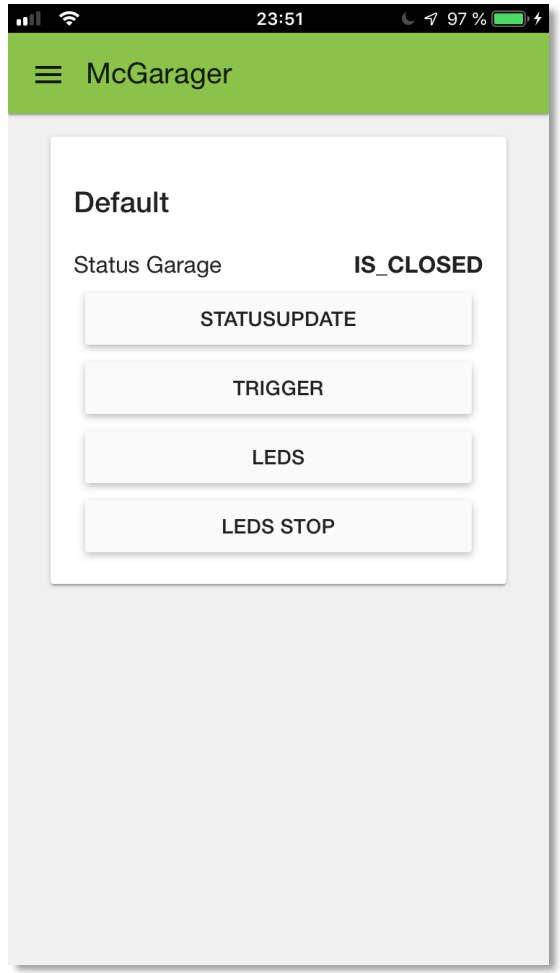
- Warum?
  - Alte Garagentor-Fernbedienung hat nur kurze Reichweite und reagiert unzuverlässig.
  - Überhaupt nicht smart!
- Ziel:
  - Einbindung ins Heimnetzwerk
  - Bessere Fernbedienung
  - Erkennung der Torposition
  - Anzeige von Zusatzinformationen → Müllampel
  - Steuerung per Weboberfläche

Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi

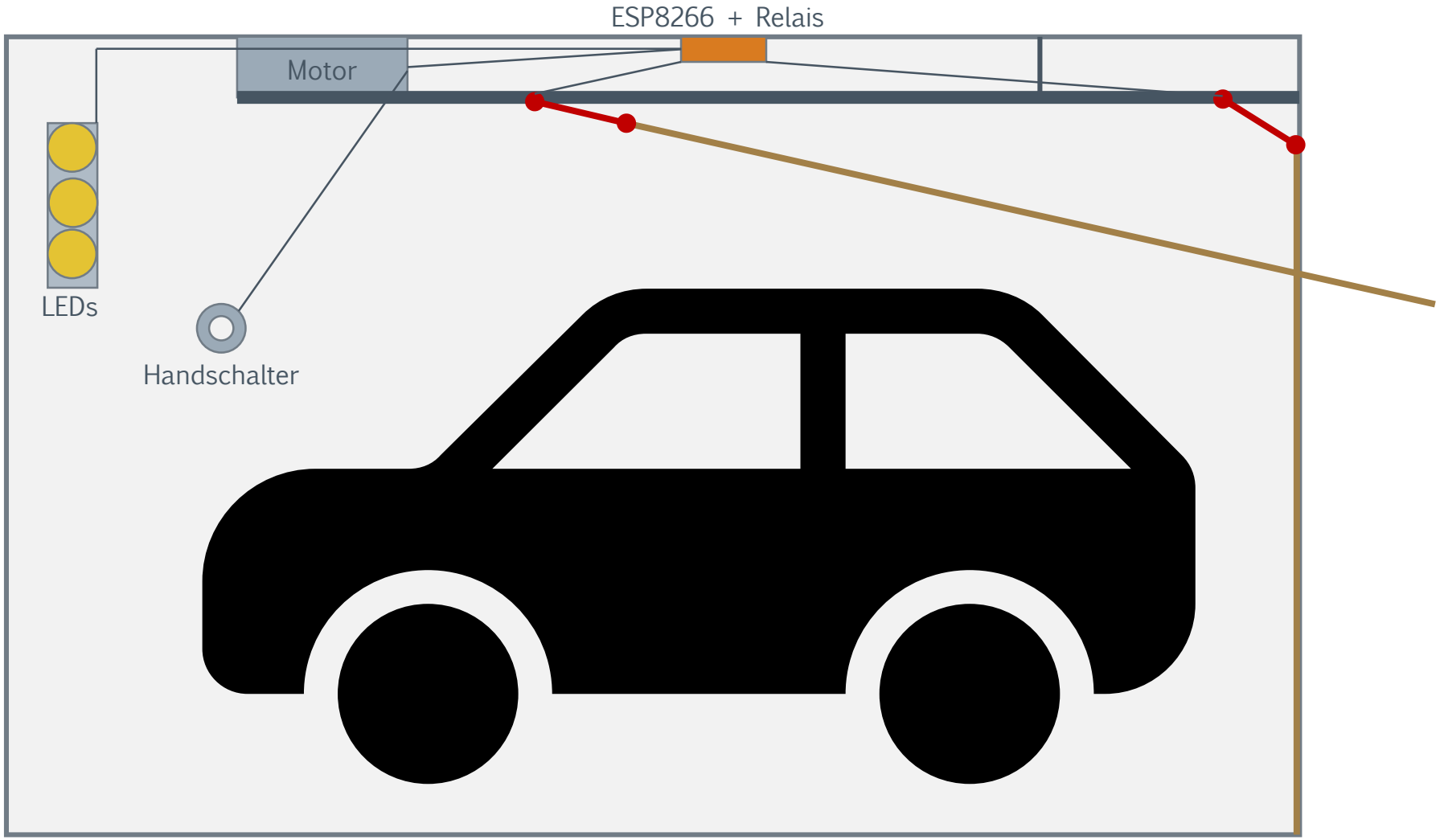
- Hardware:
  - Microcontroller ESP8266
  - 2 * Magnetsensor (an Öffnungs- und Schließposition)
  - Relais, parallel zu Handschalter
  - Neopixel-Strip
  - Fernbedienung: Amazon Dash-Button
- Software:
  - Controller recht „dumm“; Logik in NodeRed auf dem Pi
  - Kommunikation per MQTT
- https://github.com/toblum/McGarager

Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi

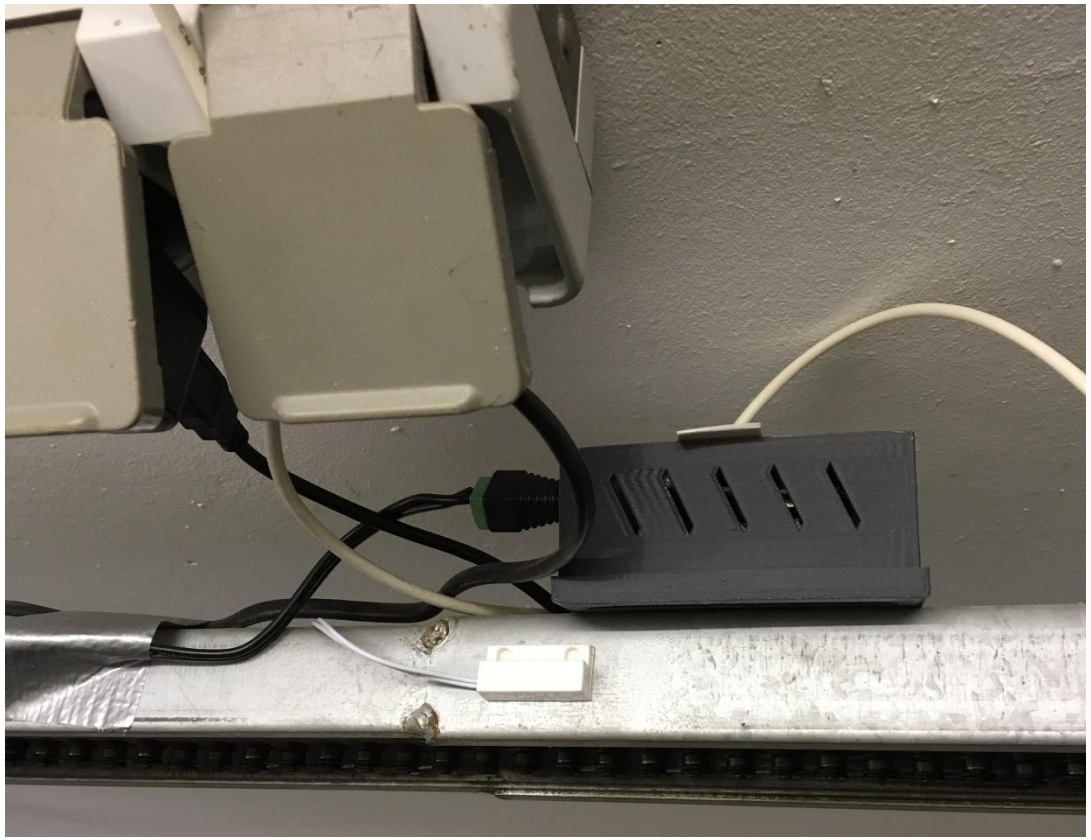


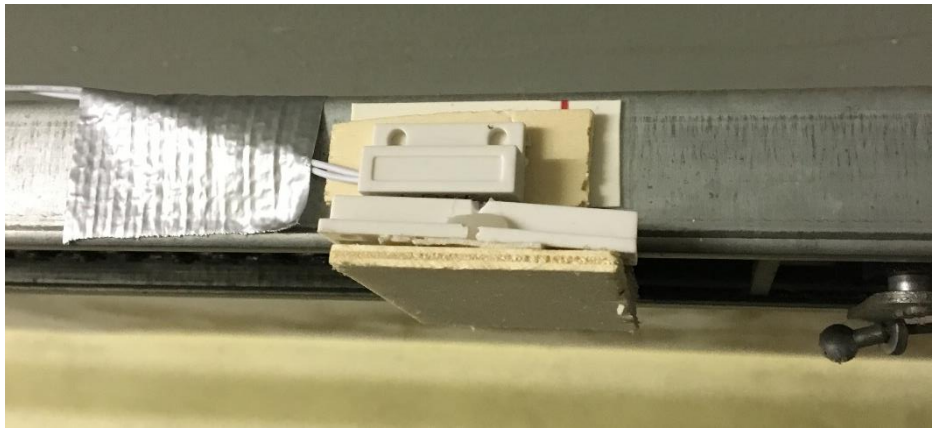


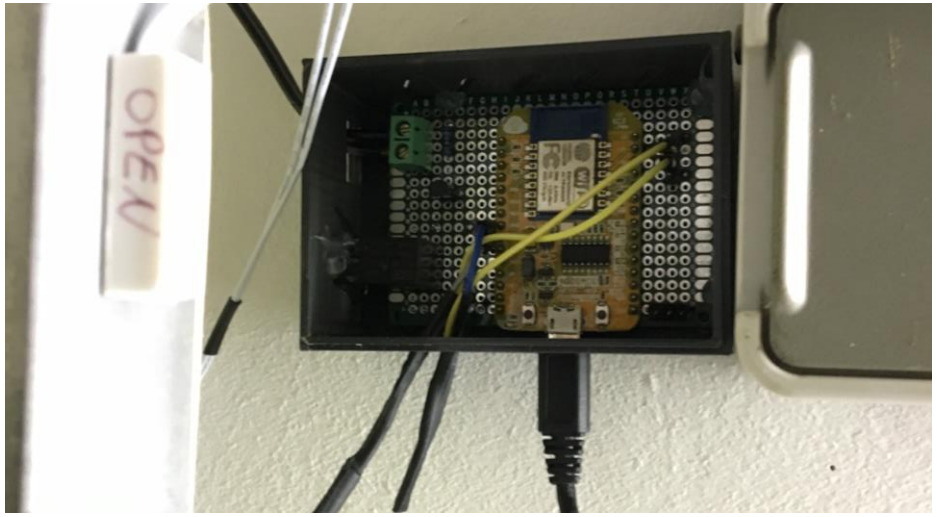
Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi



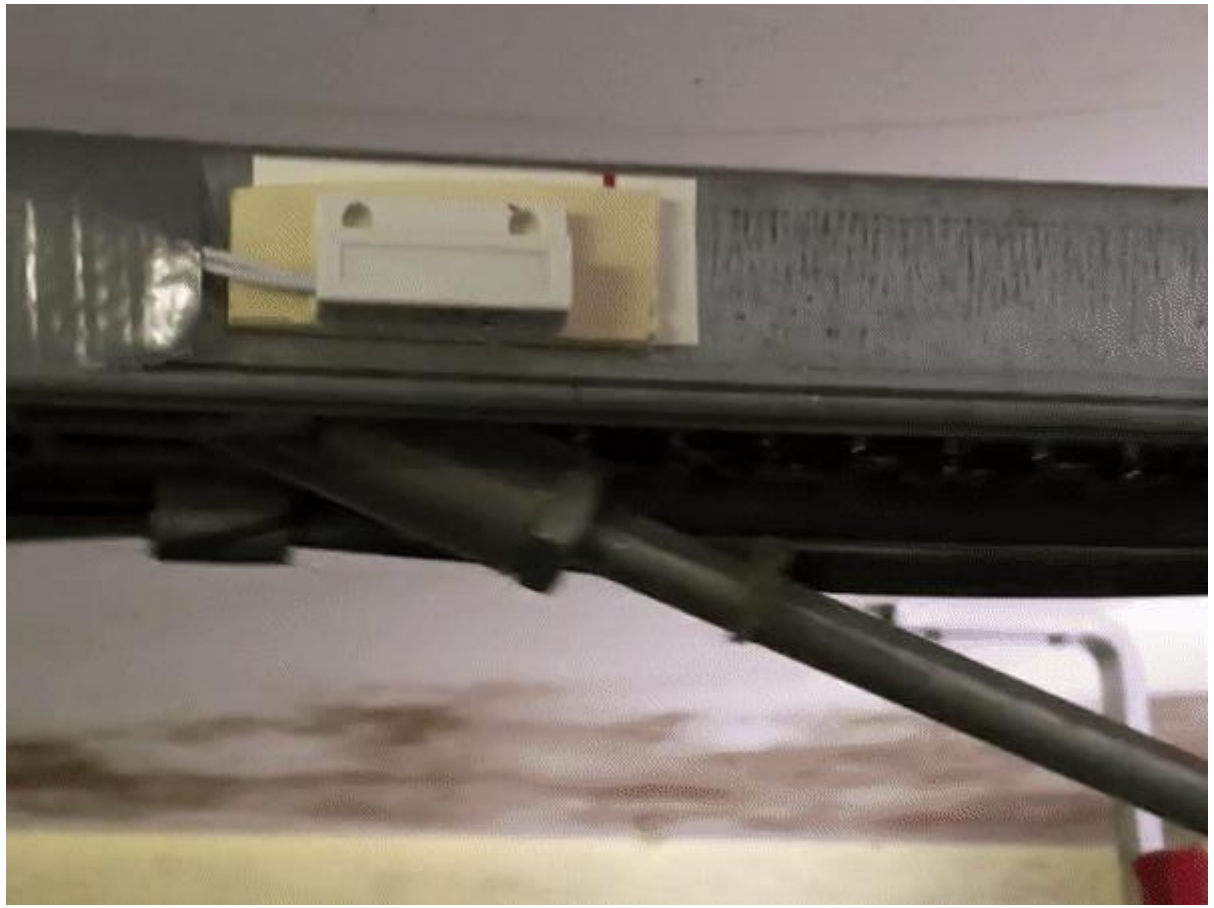
Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi







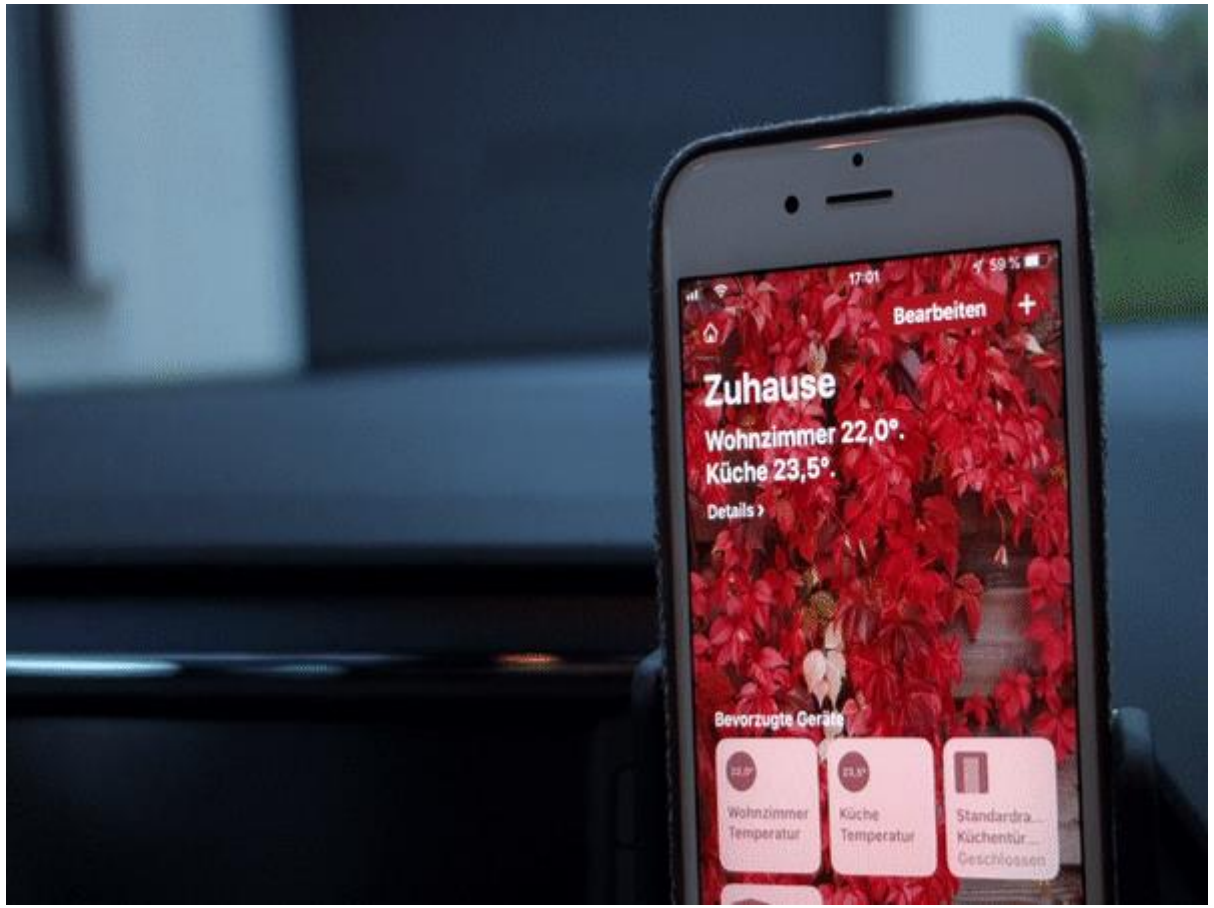
Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi



Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi

Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi

Projekt 4: Smartes Garagentor mit dem ESP8266 Microcontroller und NodeRed auf dem Pi



Projekt 5: ePaper Smarthomedisplay

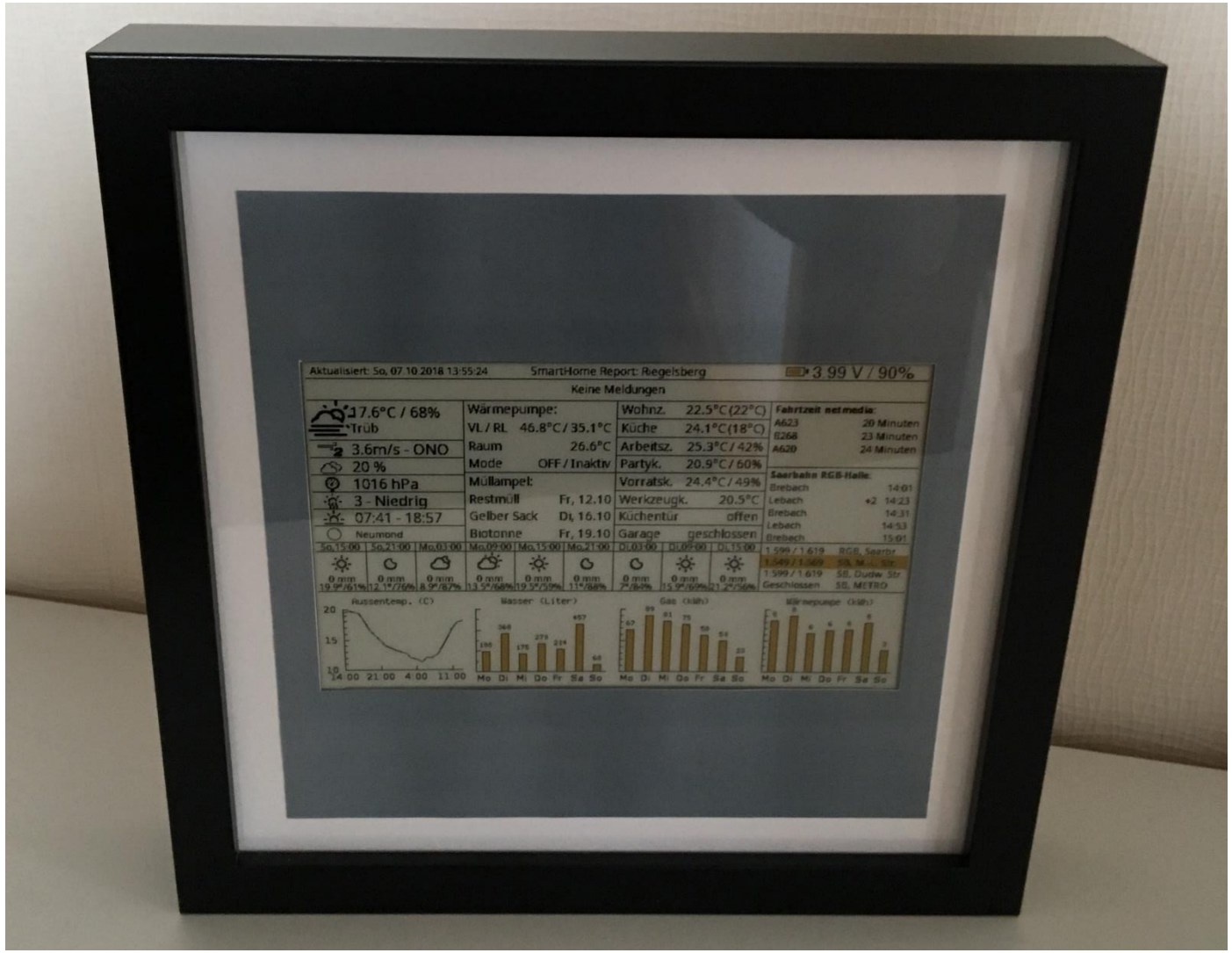
Projekt 5: Smarthome-Report mit ePaper-Display und ESP32 Microcontroller

- Bauidee aus der c‘t: https://www.heise.de/ct/ausgabe/2018-2-Drahtloses-Tuerschild-mit-E-Paper-Display-3929847.html
- Vorteile:
  - ePaper-Display verbraucht keinen Strom, wenn es sich nicht aktualisiert
  - ESP32-Microcontroller kann die meiste Zeit „schlafen“
  - Aktualisierung nur alle 15 Minuten nötig, nachts seltener
  - Durch WLAN und Akku drahtlos zu betreiben
  - Schöne Display-Optik
- Nachteile:
  - Displays nur 2- oder 3-farbig
  - Lange Aktualisierungszeit
  - Teuer

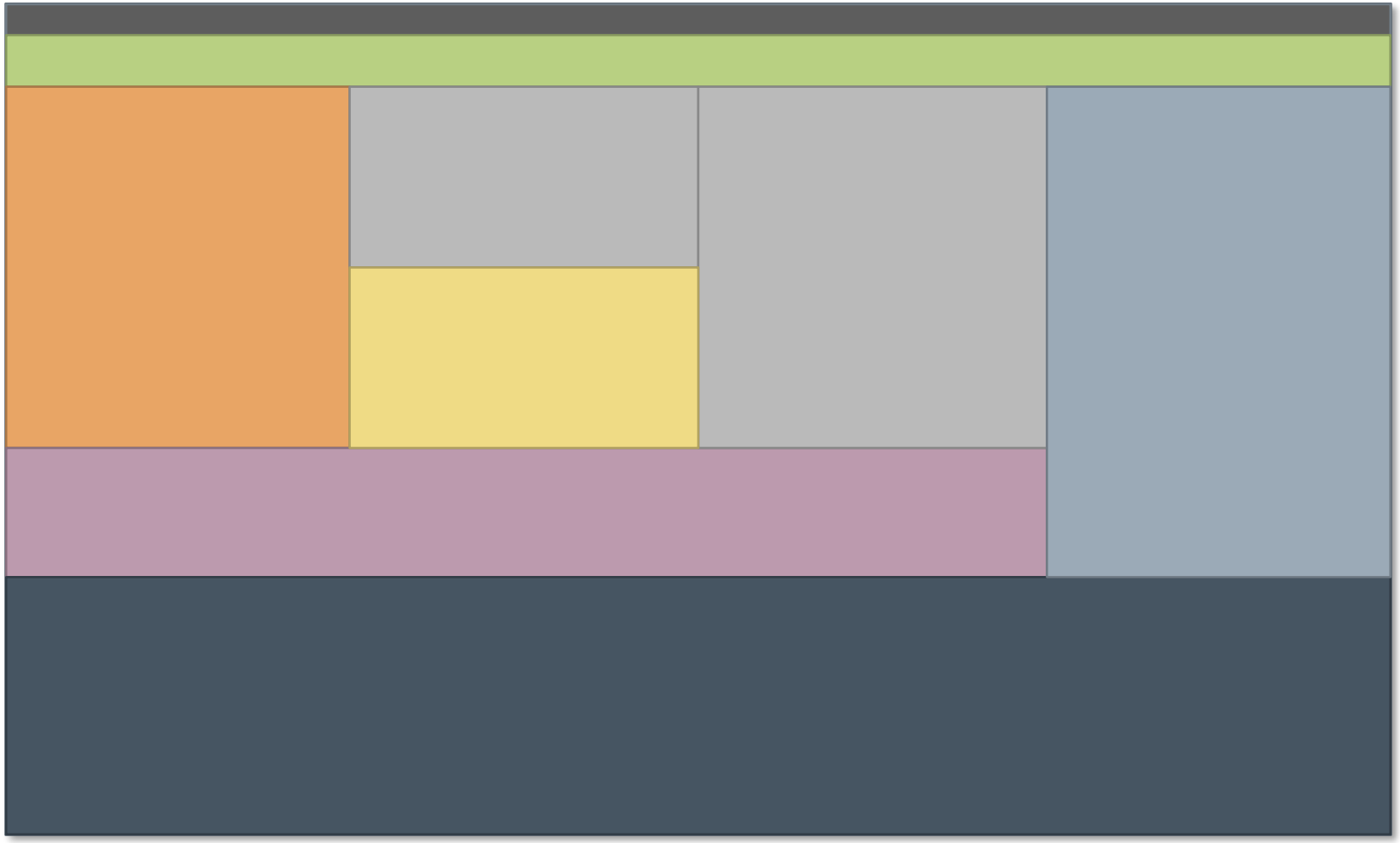
Projekt 5: Smarthome-Report mit ePaper-Display und ESP32 Microcontroller

- Software
  - Display „wacht auf“, verbindet sich mit WLAN
  - Display ruft Bild als Datenstrom von definierter URL ab
    - In der Antwort sind weitere Informationen als HTTP-Header enthalten, z. B. Zeit, die es „schlafen“ soll
  - Display aktualisiert sich
  - Display legt sich für definierte Zeit „schlafen“
- Bilderzeugung
  - PHP-Skript, das externe Datenquellen abfragt und mit GD-Library eine Grafik erzeugt
  - Grafik wird als Datenstrom zurückgeliefert und vom Microcontroller Pixel für Pixel gezeichnet.

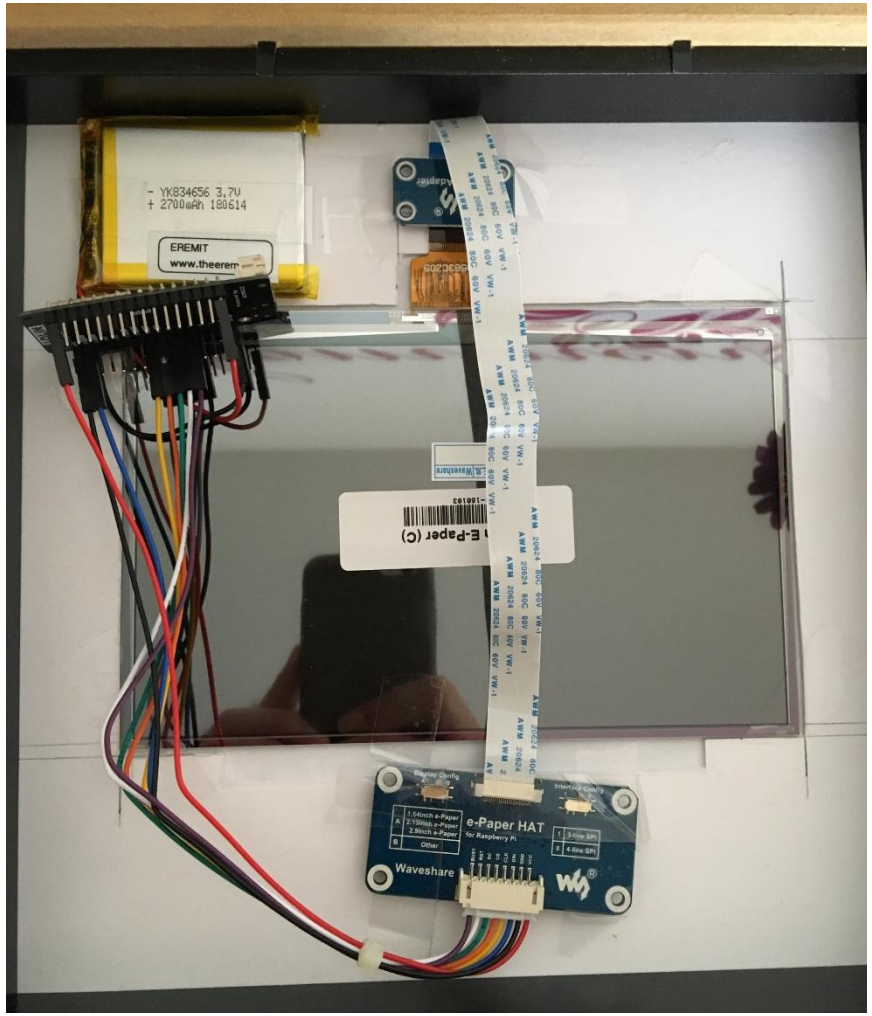
Projekt 5: Smarthome-Report mit ePaper-Display und ESP32 Microcontroller



Projekt 5: Smarthome-Report mit ePaper-Display und ESP32 Microcontroller



Projekt 5: Smarthome-Report mit ePaper-Display und ESP32 Microcontroller



Vielen Dank!

- Feedback und Fragen gerne per E-Mail: pi@tobiasblum.de

- McLighting: https://github.com/toblum/McLighting