ÜBERWACHUNG EINES HEIZUNGSSYSTEMS VIA CAN - BUS

FH CAMPUS 02 Automatisierungstechnik

Bernhard Forjan / Manuel Knittelfelder

52011623@edu.campus02.at/ 52011629@edu.campus02.at

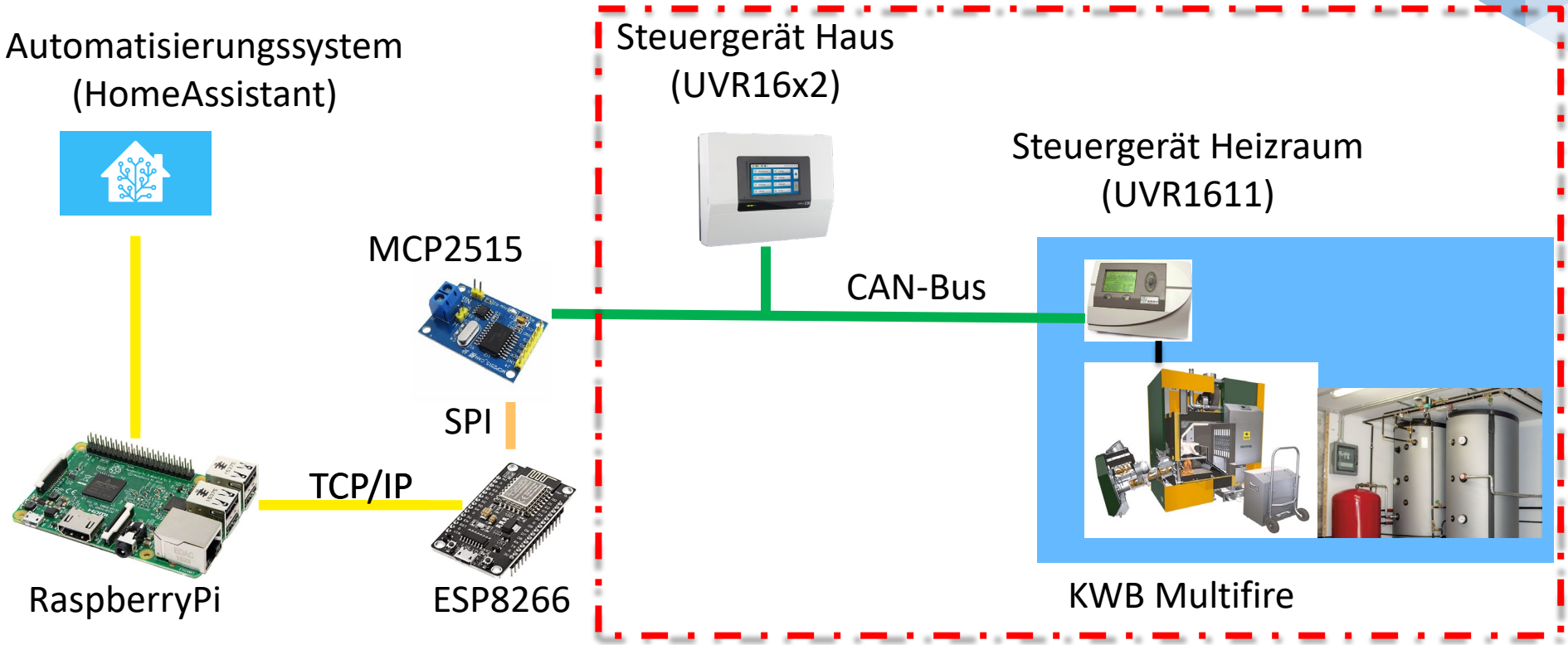
Agenda

- Projektziele
- Schematischer Aufbau
- Erweiterung UVR16x2
- Auslesen CAN Nachrichten mit CANalyzer
- Zuordnung und Programmierung CAN Bus
- Visualisierung und Benachrichtigung

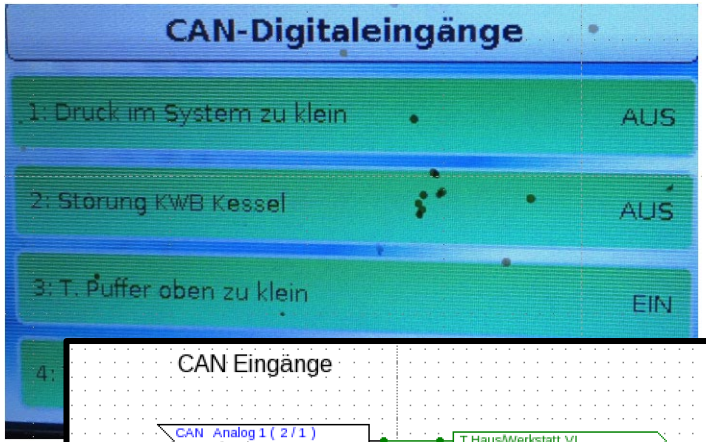
Projektziele

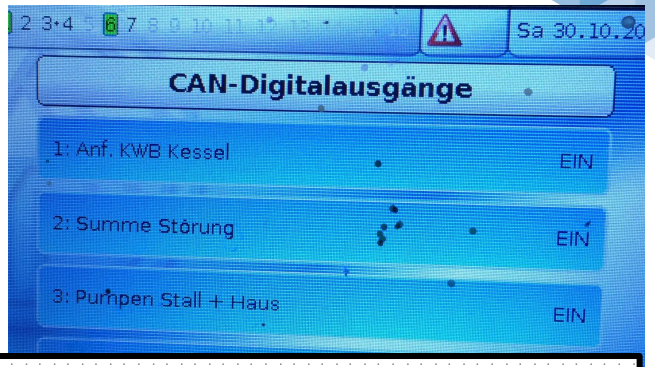
- Überwachung einer Hackschnitzelheizung und des Heizkreislaufes
- Übertragung der Daten in einen Server
- Darstellung der CAN – Werte in einer Visualisierung
- Benachrichtigung bei Grenzwertüberschreitung sowie bei Fehlermeldungen des Heizungssystems

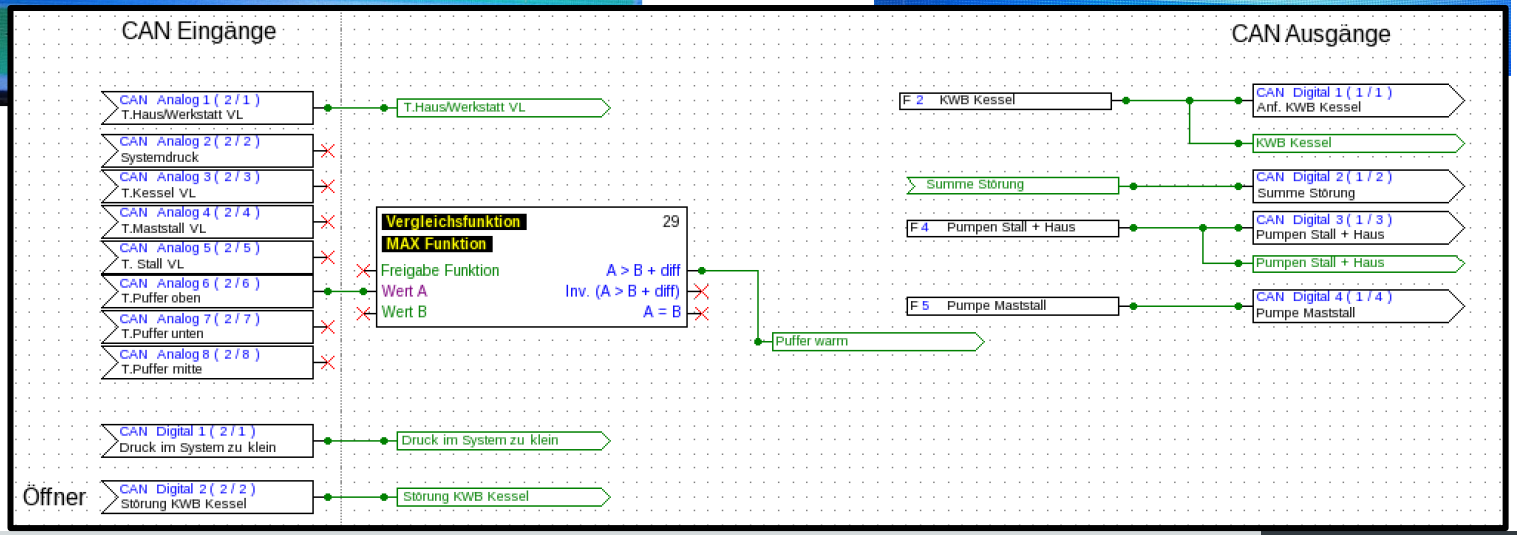
Schematischer Aufbau



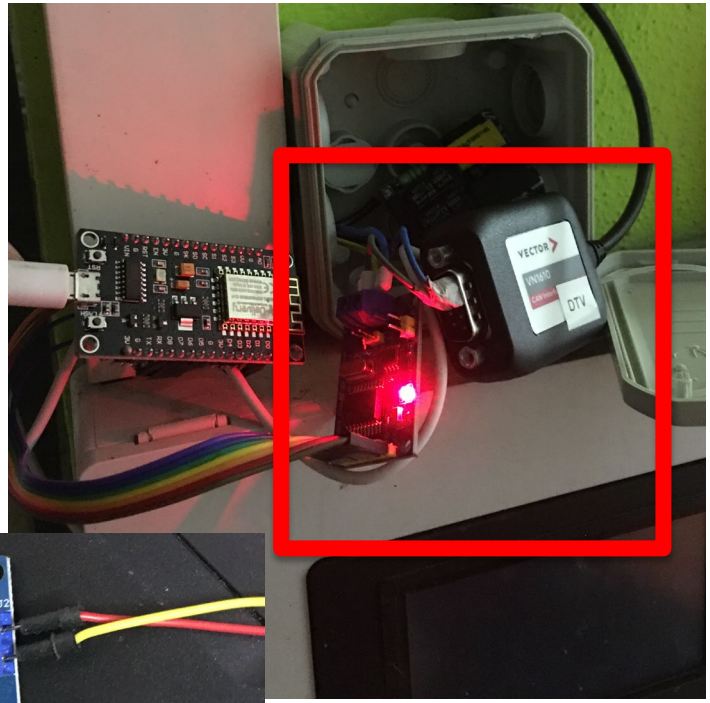
Erweiterung UVR16x2

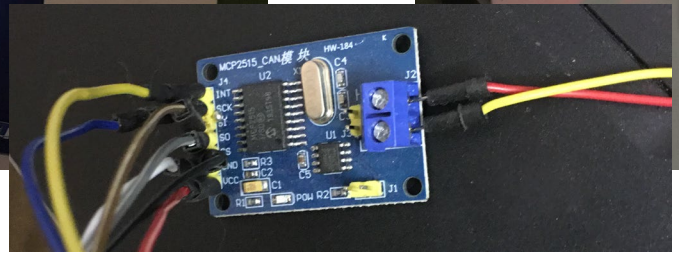




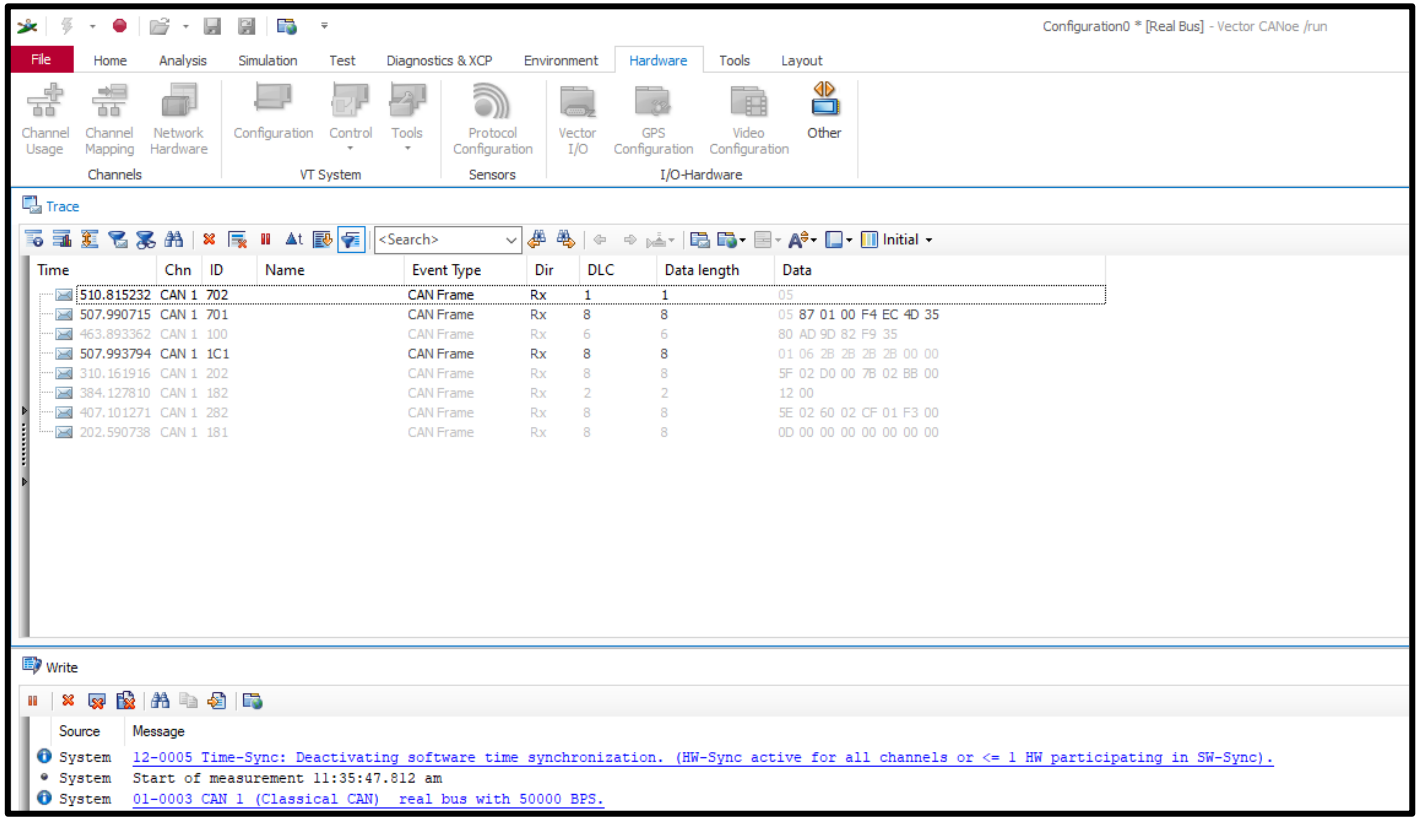


Anbindung CANalyzer





CAN-Bus Nachrichten



Zuordnung CAN Nachrichten

Identifierzuordnung

Identifier	Objekt	Verwendung
0x180 + NID	TPDO	NW Variable DIGITAL 1Bit pro Variable
0x1C0 + NID	TPDO	Einheiten…
0x200 + NID	TPDO	NW Variable ANALOG 1 – 4 je 16bit
0x240+ NID	TPDO	NW Variable ANALOG 17 – 20 je 16bit
0x280 + NID	TPDO	NW Variable ANALOG 5 – 8 je 16bit
0x2C0 + NID	TPDO	NW Variable ANALOG 21 – 24 je 16bit
0x300 + NID	TPDO	NW
0x340 + NID	TPDO	NW
0x380 + NID	TPDO	NW
0x3C0+ NID	TPDO	NW

Nachricht 0x202	NW Variablen Analog 1-4 je 2 Byte
T.Haus/Werkstatt VL	
Systemdruck	
T.Kessel VL	
T. Maststall VL	
Nachricht 0x282	NW Variablen Analog 5-8 je 2 Byte
T. Stall VL	
T. Puffer oben (1)	
T. Puffer unten (1)	
T. Puffer mitte (2)	

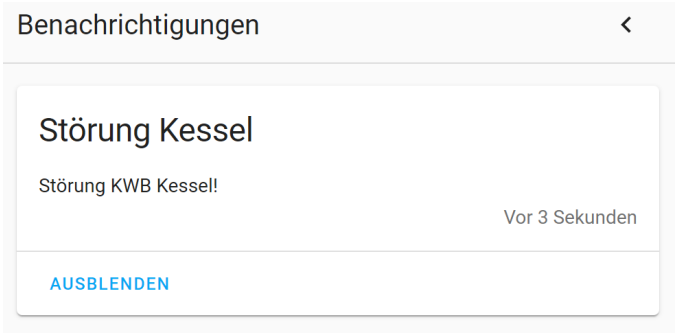
Programmierung ESP8266 + MCP2515

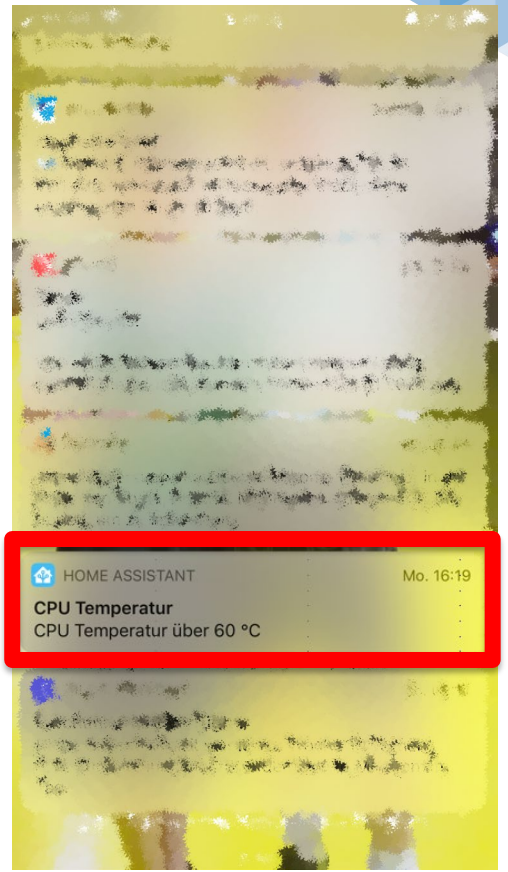
```
################################CAN Messages#####################################


canbus:
  - platform: mcp2515
    id: first_canbus
    #spi_id: McpSpi
    can_id: 4
    cs_pin: GPIO15
    bit_rate: 50kbps    #Baudrate: 5,10,20,31K25,33,40,50,80,83K3,95,100,125,200,250
    clock: 8MHZ
    mode: NORMAL         #NORMAL: Normal operation / LISTENONLY: only receive data
```

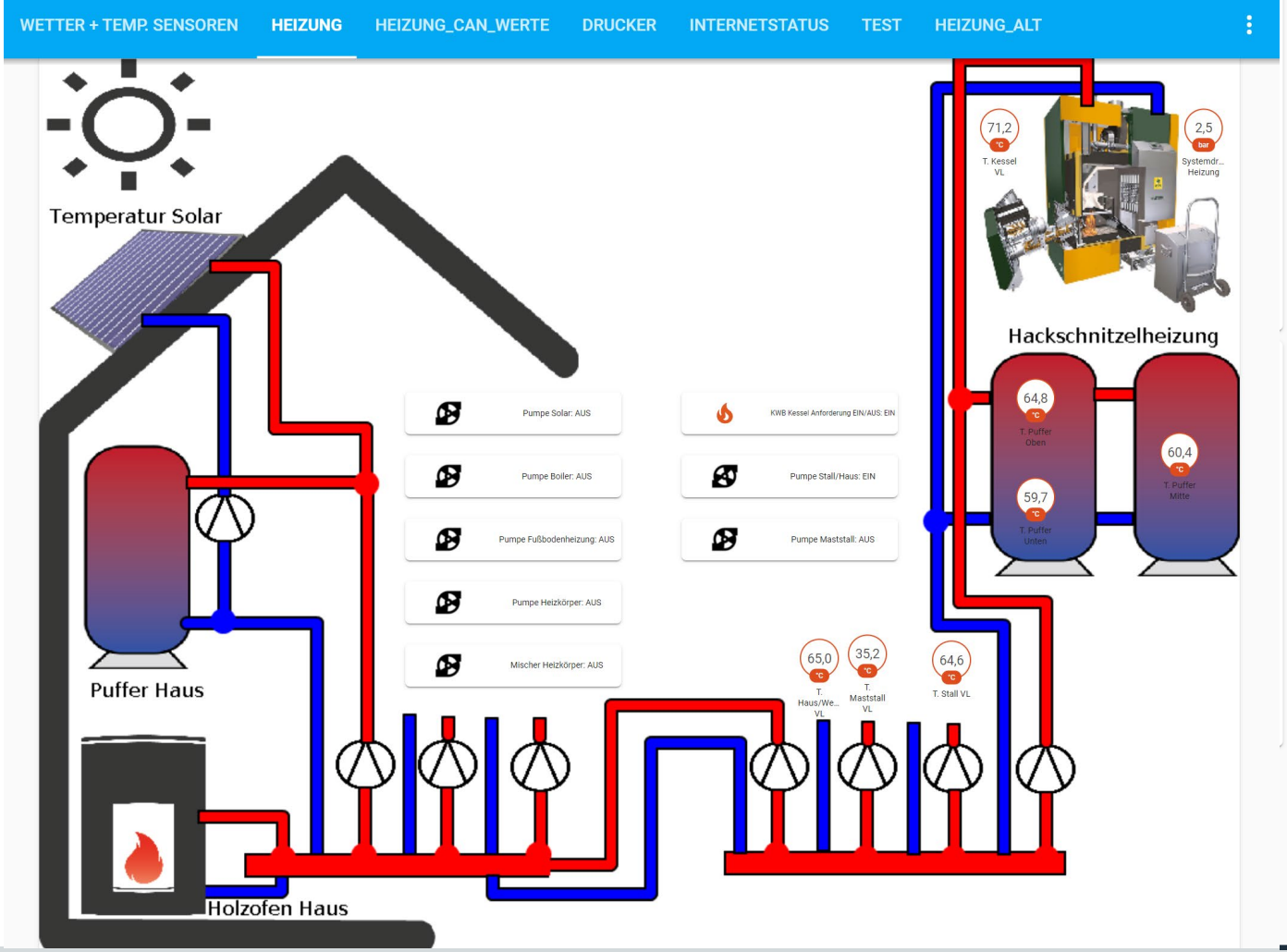
```
################################Begin Message x282########
    - can_id: 0x282
      then:
      - lambda: |-
              double c(x[0]+256*x[1]);
              double d(x[2]+256*x[3]);
              double e(x[4]+256*x[5]);
              double f(x[6]+256*x[7]);

              id(T_Stall_VL).publish_state(c/10);
              id(T_Puffer_Oben).publish_state(d/10);
              id(T_Puffer_Unten).publish_state(e/10);
              id(T_Puffer_Mitte).publish_state(f/10);
```

```
################################Begin Message x181################################

    - can_id: 0x181
      then:
        - lambda: |-
              std::string b(x.begin(), x.end());
              id(x181).publish_state(b[0]);

              std::uint16_t c(x[0]&1);
              std::uint16_t d(x[0]&2);
              std::uint16_t e(x[0]&4);
              std::uint16_t f(x[0]&8);
              std::uint16_t g(x[0]&16);
              std::uint16_t h(x[0]&32);
              std::uint16_t i(x[0]&64);
              std::uint16_t j(x[0]&128);
              std::uint16_t k(x[1]&1);

              id(Anforderung_Kessel).publish_state(c);
              id(Summe_Stoerung).publish_state(d/2);
              id(Pumpen_StHaus_Anforderung).publish_state(e/4);
              id(Pumpen_Maststall_Anforderung).publish_state(f/8);
              id(Pumpe_FBHZ_Anforderung).publish_state(g/16);
              id(Pumpe_Boiler).publish_state(h/32);
              id(Mischer_Heizkoerper).publish_state(i/64);
              id(Pumpe_Heizkoerper).publish_state(j/128);
              id(Pumpe_Solar).publish_state(k);

            # Bitmask verwendet um einzelne Bits zu verschieben bzw. darzustellen.

################################End Message x181##################################
```

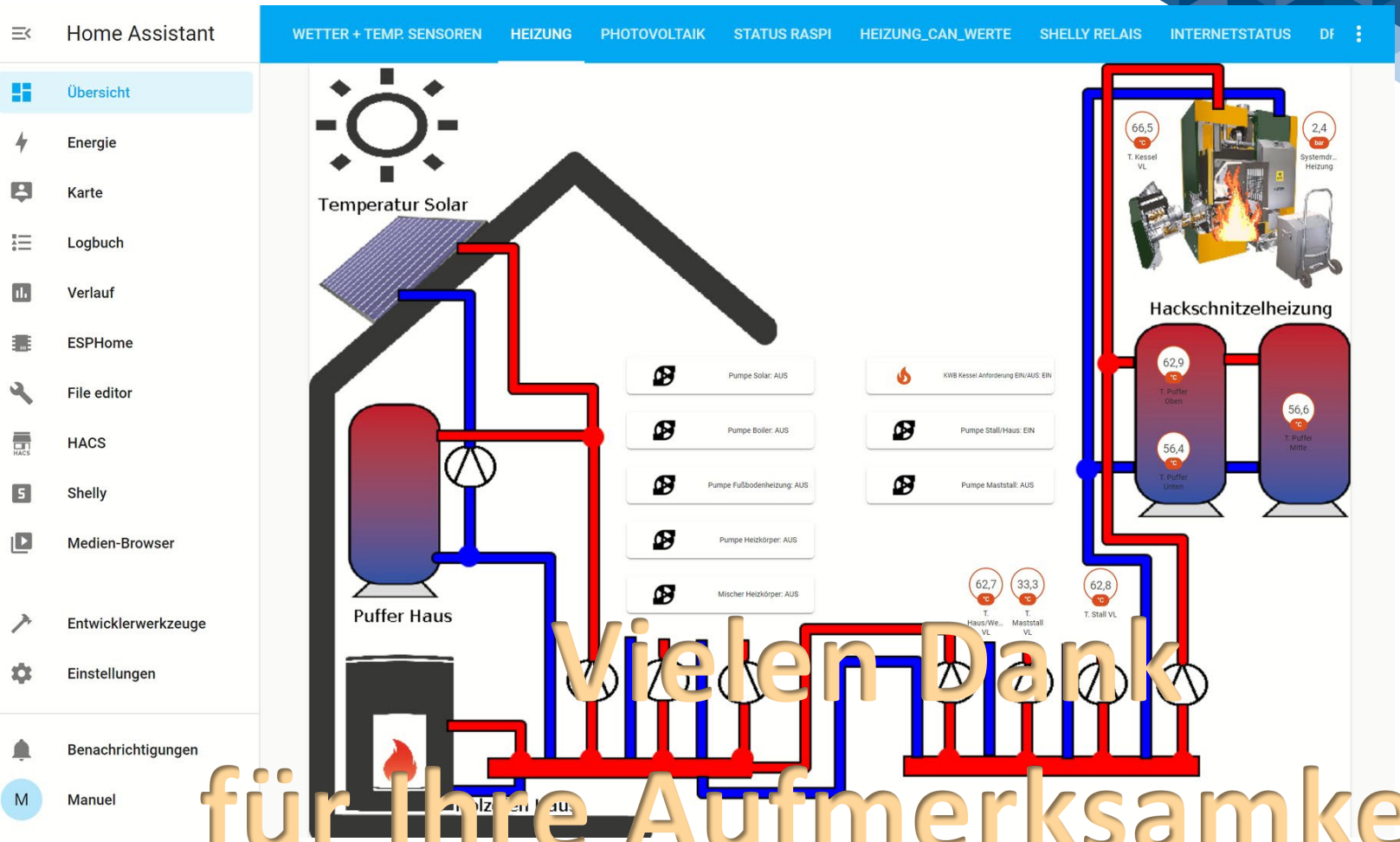
Benachrichtigung via Automatisierungssystem







T. Stall VL	64,6 °C
T. Puffer Oben	64,8 °C
T. Puffer Unten	59,7 °C
T. Puffer Mitte	60,4 °C
T. Haus/Werkstatt VL	65,0 °C
Systemdruck Heizung	2,5 bar
T. Kessel VL	71,2 °C
T. Maststall VL	35,2 °C



Vielen Dank
für Ihre Aufmerksamkeit!