



Abschlussprüfung Teil 2 – Sommer 2025

Elektroniker/-in für Informations- und Systemtechnik

Technische Dokumentation

„Environmental Measure Unit“ Steuerung für einen Serverraum

Salzgitter, 28.05.2025

Inhalt

Abbildungsverzeichnis

Tabellenverzeichnis

[1 Auftragsanalyse 6](#_Toc200542338)

[1.1 Ist-Zustand 7](#_Toc200542339)

[1.2 Soll-Zustand 8](#_Toc200542340)

[1.3 Voraussetzungen 10](#_Toc200542341)

[2 Arbeitsplan 11](#_Toc200542342)

[3 Technologieschema 12](#_Toc200542343)

[3.1 Übersicht Schnittstellen 13](#_Toc200542344)

[4 Programmlösung 13](#_Toc200542345)

[4.1 Programmbeschreibung 13](#_Toc200542346)

[4.2 Programmfunktionen 15](#_Toc200542347)

[4.3 Programmablaufpläne 16](#_Toc200542348)

[4.3.1 Hauptprogramm 17](#_Toc200542349)

[4.3.2 Setup 17](#_Toc200542350)

[4.3.3 Initialisierung der externen Interrupts 18](#_Toc200542351)

[4.3.4 Initialisierung des Timer Interrupts 19](#_Toc200542352)

[4.3.5 Messwerte aktualisieren 20](#_Toc200542353)

[4.3.6 Anlagenstatus aktualisieren 21](#_Toc200542354)

[4.3.7 Setzen der LEDs 22](#_Toc200542355)

[4.3.8 Umschalten des Betriebsmodus 23](#_Toc200542356)

[4.3.9 Umschalten des Stickstoffventils 24](#_Toc200542357)

[4.3.10 Laufzeitsteuerung 25](#_Toc200542358)

[4.3.11 Timer0 ISR 26](#_Toc200542359)

[4.3.12 Setzen der Lüftergeschwindigkeit 27](#_Toc200542360)

[4.3.13 Ausgabe über UART 29](#_Toc200542361)

[4.3.14 Auslesen der Temperatur 30](#_Toc200542362)

[4.3.15 Auslesen des Sauerstoffgehalts 31](#_Toc200542363)

[4.3.16 Serverraumtür bedienen 32](#_Toc200542364)

[4.3.17 Auslesen der Lüftergeschwindigkeit 33](#_Toc200542365)

[4.4 Programmcode 34](#_Toc200542366)

[4.4.1 main.c 34](#_Toc200542367)

[4.4.2 main.h 42](#_Toc200542368)

[5 Kurzbedienungsanleitung 44](#_Toc200542369)

[5.1 Allgemeine Informationen 45](#_Toc200542370)

[5.2 Enthaltene Komponenten 45](#_Toc200542371)

[5.3 Herstellung der Betriebsbereitschaft 45](#_Toc200542372)

[5.4 Anlagenbetrieb 47](#_Toc200542373)

[5.4.1 Startsequenz 48](#_Toc200542374)

[5.4.2 Regulärer Betrieb der EMU\_Serverraum\_Steuerung 48](#_Toc200542375)

[5.4.3 Zusätzliche Funktionen 48](#_Toc200542376)

[6 Inbetriebnahmeprotokoll 49](#_Toc200542377)

[6.1 Allgemeine Daten 49](#_Toc200542378)

[6.2 Durchzuführende Prüfungen 50](#_Toc200542379)

[6.3 Vorbereitung 51](#_Toc200542380)

[6.4 Konfiguration der Baugruppen „EMU“ und „ATmega32-Board“ 52](#_Toc200542381)

[6.4.1 Jumperbelegungen 52](#_Toc200542382)

[6.5 Prüfung des Baugruppenrahmen + IK-88/1 nach DIN VDE 0702 und DGUV Vorschrift 3 53](#_Toc200542383)

[6.5.1 Sichtprüfung des Baugruppenrahmen + IK-88/1 53](#_Toc200542384)

[6.5.2 Messungen des Baugruppenrahmen + IK-88/1 54](#_Toc200542385)

[6.6 Prüfung der Spannungsversorgung 54](#_Toc200542386)

[6.7 Sichtprüfung der Baugruppen 55](#_Toc200542387)

[6.7.1 Baugruppe „EMU“ 55](#_Toc200542388)

[6.7.2 Baugruppe „ATmega32-Board“ 55](#_Toc200542389)

[6.7.3 Zusatzplatine zur Umschaltung des Betriebs- und Ventilstatus 56](#_Toc200542390)

[6.8 Prüfung des geschlossenen Systems „EMU-Serverraumsteuerung“ nach DIN VDE 0701 und DGUV Vorschrift 3 56](#_Toc200542391)

[6.8.1 Sichtprüfung des geschlossenen Systems 57](#_Toc200542392)

[6.8.2 Messungen des geschlossenen Systems 58](#_Toc200542393)

[6.9 Funktionsprüfung 58](#_Toc200542394)

[6.9.1 Funktionsprüfung – Hardware: Baugruppe „EMU“ 58](#_Toc200542395)

[6.9.2 Funktionsprüfung – Hardware: Baugruppe „ATmega32-Board“ 59](#_Toc200542396)

[6.9.3 Funktionsprüfung – Software: Baugruppe „EMU“, „ATmega32-Board“ und Zusatzplatine 60](#_Toc200542397)

[6.9.4 Auswertung 61](#_Toc200542398)

[6.9.5 Bestätigung der Prüfung 62](#_Toc200542399)

[7 Übergabe- /Einweisungsprotokoll 63](#_Toc200542400)

[7.1 Allgemeine Daten 63](#_Toc200542401)

[7.2 Gegenstand der Übergabe 64](#_Toc200542402)

[7.2.1 Hardware (EMU-Serverraumsteuerung) 64](#_Toc200542403)

[7.2.2 Software 64](#_Toc200542404)

[7.2.3 Dokumente 64](#_Toc200542405)

[7.3 Übergabe/Einweisung 64](#_Toc200542406)

[7.3.1 Überprüfung der Vollständigkeit 64](#_Toc200542407)

[7.3.2 Funktionsprüfung mit dem Auftraggeber 65](#_Toc200542408)

[7.4 Mängelliste 66](#_Toc200542409)

[7.5 Bestätigung über Erhalt 67](#_Toc200542410)

Abbildungsverzeichnis

Abbildung 1: Beispielhafte Anzeige zur Ausgabe der Messwerte und Zustände……………..9

Abbildung 2: Technologieschema………………………………………………………………………………………13

Abbildung 3: Übersicht Schnittstellen.……………………………………………………………………………..13

Abbildung 4: Programmablaufplan für die main-Funktion………………………………………………17

Abbildung 5: Programmablaufplan für setup( )………………………………………………………………..18

Abbildung 6: Programmablaufplan für initExternalInterrupts( )……………………………………..19

Abbildung 7: Programmablaufplan für initTimerInterrupts( )………………………………………….20

Abbildung 8: Programmablaufplan für updateMeasurements( )…………………………………….21

Abbildung 9: Programmablaufplan für updateState( )…………………………………………………….22

Abbildung 10: Programmablaufplan für setLEDs( )………………………………………………………….23

Abbildung 11: Programmablaufplan für setBetrieb( )………………………………………………………24

Abbildung 12: Programmablaufplan für updateVentil( )………………………………………………….25

Abbildung 13: Programmablaufplan für millis( )………………………………………………………………26

Abbildung 14: Programmablaufplan für ISR(TIMER0\_OVF\_vect)……………………………………27

Abbildung 15: Programmablaufplan für setFanSpeed( )………………………………………………….28

Abbildung 16: Programmablaufplan für uartOutput( )…………………………………………………….29

Abbildung 17: Programmablaufplan für getTemperature(uint8\_t sensor)…………………….30

Abbildung 18: Programmablaufplan für getO2(uint8\_t sensor)………………………………………31

Abbildung 19: ISR(INT1\_vect)…………………………………………………………………………………………….32

Abbildung 20: Programmablaufplan für getFanSpeed(uint8\_t sensor)………………………….33

Abbildung 21: Ausgabe der Startsequenz auf dem LC-Display……………………………….………48

Tabellenverzeichnis

Tabelle 1: Sauerstoffgehalts- und Strombereich sowie Zustände des Stickstoffventils.10

Tabelle 2: Temperaturbereich sowie Lüftergeschwindigkeit…………………………………..………10

Tabelle 3: Voraussetzung für die Realisierung des Soll-Zustandes………………………………...11

Tabelle 4: Arbeitsplan zur Realisierung des Soll-Zustandes…………………………………………….12

Tabelle 5: Liste der Progammfunktionen………………………………………………………………………….16

Tabelle 5: Übersicht benötigter Komponenten…………………………………………………………………45

Tabelle 6: Übersicht der durchzuführenden Prüfungen…………………………………………………..50

Tabelle 7: Benötigte Geräte und Software für die Inbetriebnahme………………………………..51

Tabelle 8: Konfiguration der Baugruppen „EMU“ und „ATmega32-Board“……………………52

Tabelle 9: Jumperbelegungen……………………………………………………..…………………………………….52

Tabelle 10: Sichtprüfung des Baugruppenrahmen + IK-88/1………………………………………….53

Tabelle 11: Messungen des Baugruppenrahmen + IK-88/1.……………………………………………54

Tabelle 12: Überprüfung der Spannungsversorgung…………….…………………………………………54

Tabelle 13: Sichtprüfung der Baugruppe „EMU“………………………………………………………………55

Tabelle 14: Sichtprüfung der Baugruppe „ATmega32-Board“….…………………………………….55

Tabelle 15: Sichtprüfung der Zusatzplatine………………………………………………………………………56

Tabelle 16: Sichtprüfung des geschlossenen Systems…………………………………………………….57

Tabelle 17: Messungen des geschlossenen Systems……..………………………………………………..58

Tabelle 18: Funktionsprüfung Hardware der Baugruppe „EMU“…………………………………….58

Tabelle 19: Funktionsprüfung Hardware der Baugruppe „ATmega32-Board“……..……….59

Tabelle 20: Funktionsprüfung Software der Baugruppen „EMU“, „ATmega32-Board“ und Zusatzplatine…………………………………………………………..………………………………………60

Tabelle 21: Auswertung…………………….……………………………………………………………………………….61

Tabelle 22: Überprüfung auf Vollständigkeit bei der Übergabe………………………………………64

Tabelle 23: Funktionsprüfung bei Übergabe…………………………………………………………………….65

Tabelle 24: Mängelliste bei Übergabe……………………………………………………………………………….66

# Auftragsanalyse

Der Serverraum eines Unternehmens wurde als kritische Infrastruktur (KRITIS) eingestuft. Dies erfordert weitere Sicherheitsmaßnahmen, insbesondere im Bereich des Brandschutzes. Zur Reduktion des Brandrisikos soll ein System entwickelt werden, mit der der Sauerstoffgehalt reguliert werden soll. Hierbei steht die Baugruppe „EMU“ (Environmental Measurement Unit) zur Verfügung, mit der die Temperatur und der Sauerstoffgehalt im Raum überwacht wird. Zusätzlich wird ein Lüfter für die Abluft und ein Ventil zur Steuerung der Stickstoffzufuhr eingesetzt.

Das Stickstoffventil soll durch den Betriebsmodus der Steuerung einstellbar sein. Ist der Automatikmodus aktiviert, so wird das Stickstoffventil ab einem gewissen Sauerstoffgehalt geöffnet. Ist die Steuerung im manuellen Modus, kann das Stickstoffventil über einen Taster geöffnet oder geschlossen werden. In beiden Fällen wird über eine LED der Zustand des Ventils angegeben (geöffnet = LED an, geschlossen = LED aus).

Zum Zugang in den Serverraum muss man die Serveraumtür öffnen. Man soll die Serverraumtür nur öffnen können, wenn der Sauerstoffgehalt im Serverraum sich über 17% befindet und das Stickstoffventil zu ist.

Der Status der Serverraumtür wird über den monitor des Administrators ausgegeben. Der Zugangsstatus (erlaubt, untersagt) wird mit 2 LEDs angezeigt Grün, und Rot: erlaubt = Grün und untersagt = Rot.

Eine Alarmmeldung wird auf dem Monitor ausgegeben sobald die Serverraumtür unerlaubt geöffnet wird.

Zur Simulation des Sauerstoffgehalts soll eine geeignete Schaltung entwickelt und

angeschlossen werden.

Der Sauerstoffgehaltsbereich, sowie der Temperaturbereich soll zusätzlich auf einem Monitor (z.B. über ein Terminalprogramm „PuTTY“) angezeigt werden.

Ebenso soll der Betriebsmodus, die Drehzahl des Lüfters, Status der Serverraumtür, Türzugang, Alarmmeldung und der Zustand des Stickstoffventils auf dem Monitor ausgegeben werden.

Die Lüftergeschwindigkeit wird anhand des Temperaturbereichs angepasst.

## Ist-Zustand

Die Baugruppe „EMU“ ist ein Sensorerfassungsmodul. An dieser kann ein Pt100-Temperatursensor, ein Feuchtigkeitssensor und ein C02-Sensor mit integriertem Pulsweitenmodulator (PWM) zur Ansteuerung von Lüftern angeschlossen werden:

* Der Pt100-Temperatursensor liefert in Kombination mit der nachfolgenden Instrumentenverstärkerschaltung ein analoges Signal zwischen 0-5V. Der Widerstand des Sensors verhält sich linear zur Temperaturänderung.
* Zusätzlich kann an die EMU-Baugruppe ein weiterer Sensor (z.B. Feuchtigkeitssensor) angeschlossen werden, der ein analoges Signal zwischen 0-12V liefert.
* Der C02-Sensor kann ebenfalls an die Baugruppe angeschlossen werden und wird mittels I2C-Schnittstelle mit dem Mikrocontroller verbunden.
* Der Mikrocontroller steuert über I²C die beiden DAC’s (Digital-Analog-Converter) an. Diese wandeln mit 12 Bit ein digitales Signal in ein verstärktes Analogsignal um. Dieses wird auf den PWM-Baustein gelegt und als Ausgangssignal für den Lüfter bereitgestellt.
* Der Lüfter erzeugt an der Tacholeitung ein 12V-Rechtecksignal mit einer Frequenz von ca. 7,7 Hz (210 RPM) für Vmin und 48,8 Hz (1350 RPM) für Vmax. Dieses Signal wird dem Frequenz-Spannungs-Wandler zugeführt, der das PWM-Signal in ein analoges Gleichspannungssignal umwandelt. Dieses kann über einen Anschluss abgegriffen werden. Die Gleichspannung 0-5V entspricht proportional 0-1350 RPM des Lüfters.

Die EMU-Baugruppe verfügt ebenso über einen „Inbetriebnahme-Modus“, welcher die folgenden Funktionen testet: Ausgabe LCD, Ausgabe LED, Taster, Ausgabe UART, DACs (PWM), ADC (Temperatur), DAC (C02-Simulation).

Für diesen Modus muss beim erneuten Einschalten der Taster –S1 so lange gedrückt werden, bis auf dem LC-Display (LCD) „Inbetriebnahme“ erscheint.

## Soll-Zustand

Mittels EMU-Baugruppe und einer externen Mikrocontroller-Baugruppe (z.B. ATmega32-Board) sollen für den Serverraum der Sauerstoff- und Temperaturgehalt überwacht und ausgegeben werden. Ebenso soll der Betriebsmodus, der Zustand des Stickstoffventils, der Status der Serverraumtür, der Zugangsstatus der Serverraumtür, die Alarmmeldung und die Geschwindigkeit des Lüfters angezeigt werden:

* Einlesen der Sensorsignale und Kalibrieren in die entsprechenden Messbereiche:
  + Sauerstoffgehalt: 0 % bis 100 %
  + Temperatur: 0°C bis 100°C
  + Lüftergeschwindigkeit: 0 bis 1350 RPM
* Erstellen einer geeigneten Schaltung zur Simulation des Sauerstoffgehalts
* Umschalten der booleschen Zustände
  + Betriebsmodus: AUTOMATIK/MANUELL
  + Stickstoffventil: geöffnet/geschlossen
    - LED an/ LED aus
  + Türzugang: erlaubt/untersagt
    - LED\_Grün an/ LED\_Rot aus
  + Türstatus: AUF/ZU
* Ausgabe aller Messwerte und Zustände auf einem Monitor:

Abbildung : Beispielhafte Anzeige zur Ausgabe der Messwerte und Zustände

-------------------------------------------

Sauerstoffgehalt: < 15 %

Temperatur: < 15 Grad

Luefter: 700 U/min

Modus: AUTOMATIK

Stickstoffventil: ZU

Tuerzugang: erlaubt

Tuerstatus: AUF

Alarm: Tuer unerlaubt geoeffnet!

-------------------------------------------

|  |  |  |  |
| --- | --- | --- | --- |
| **Sauerstoffgehaltsbereich** | < 15 % | 15 % bis 17 % | > 17 % |
| **Stromsignal des Sauerstoffsensors** | < 6,4 mA | 6,4 mA bis 6,72 mA | > 6,72 mA |
| **Stickstoffventil** | Geschlossen | Geschlossen | Geöffnet |
| **Türzugang** | untersagt | untersagt | erlaubt |

* Zustände des Stickstoffventils des Türzugangs basierend auf dem Sauerstoffgehalt:

Tabelle : Sauerstoffgehalts- und Strombereich sowie Zustände des Stickstoffventils

|  |  |  |  |
| --- | --- | --- | --- |
| **Temperaturbereich** | < 15 Grad | 15 bis 28 Grad | > 28 Grad |
| **Lüftergeschwindigkeit** | 300 U/min | 700 U/min | 1000 U/min |

* Zustände des Lüfters basierend auf dem Temperaturbereich:

Tabelle 2: Temperaturbereich sowie Lüftergeschwindigkeit

* Alarm soll aktiviert werden, wenn die Serverraumtür aufgemacht wird obwohl Türzugang untersagt ist. Beim Alarmfall soll das Stickstoffventil automatisch geschlossen werden.

## Voraussetzungen

|  |
| --- |
| Voraussetzungen |
| 19“ Baugruppenträger mit Netzteileinschub IK-88/1 |
| Baugruppe „EMU“ inkl. Temperatursensor und Lüfter |
| Baugruppe „ATmega32-Board“ mit der Firmware „EMU\_Serverraum\_Steuerung\_V2“ |
| Zusatzplatine mit 4x Tastern und 4x LEDs zur Zustandswechselung des Betriebsmodus und des Stickstoffventils, sowie Bereitstellung weiterer Funktionen |
| Geeignete Schaltung an –X6 zur Sauerstoffsimulation |
| Entwicklungsumgebung: Microchip Studio 7.0 |
| Dokumentation: Microsoft Word 2016 |
| Visualisierungsprogramm für das Technologieschema: Microsoft PowerPoint 2016 |
| Terminalprogramm: PuTTY + 9-Polig Sub-D auf USB Kabel (RS232) |
| Programmentwicklungstool: PapDesigner |
| Programmierschnittstelle: ISP-Programmieradapter (stk500v2) |

Tabelle 3: Voraussetzungen für die Realisierung des Soll-Zustandes

# Arbeitsplan

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nr. | Aufgabe | SOLL Zeit [MIN] | IST Zeit [MIN] | Material / Hilfsmittel |
| 1 | Auftragsanalyse,  Analyse Soll/Ist-Zustand,  Voraussetzungen ermitteln | 50 | 40 | Arbeitsauftrag |
| 2 | Erstellung des Technologieschemas | 50 | 40 | Schaltplan  PowerPoint |
| 3 | Planung der Software und Erstellung der Programmbeschreibung | 50 | 30 | Arbeitsauftrag  Schaltplan |
| 4 | Programmablaufpläne erstellen | 100 | 100 | Programmbeschreibung  PapDesigner |
| 5 | Erstellung der Software | 280 | 290 | Arbeitsauftrag  Programmablaufplan  Microchip Studio |
| 6 | Testen der Baugruppenfunktion sowie Funktion der Software | 50 | 50 | 19“ Baugruppenträger mit Netzteileinschub IK-88/1  Baugruppe „EMU“  Baugruppe „ATmega32-Board“  Lüfter „ARCTIC F12 PWM“  Temperatursensor  Sauerstoffsimulationsschaltung  Zusatzplatine  Microchip Studio  „PuTTY“  Programmbeschreibung  Programmablaufplan |
| 7 | Erstellung einer Kurzbedienungsanleitung | 15 | 25 | Arbeitsauftrag  Programmbeschreibung  Microsoft Word |
| 8 | Erstellung eines Inbetriebnahmeprotokolls | 15 | 25 | Arbeitsauftrag  Schaltplan  EMU-Serverraumsteuerung |
| 9 | Erstellung des Übergabe- und des Einweisungsprotokolls | 50 | 50 | Erstelle Dokumente  Microsoft Word |
| 10 | Erstellung der Dokumentation | 160 | 170 | Dokumentation  Microsoft Word |
| 11 | Übergabe | 20 | 20 | EMU-Serverraumsteuerung |
|  | **Summe** | **840** | **840** |  |

Tabelle 4: Arbeitsplan zur Realisierung des Soll-Zustandes

# Technologieschema

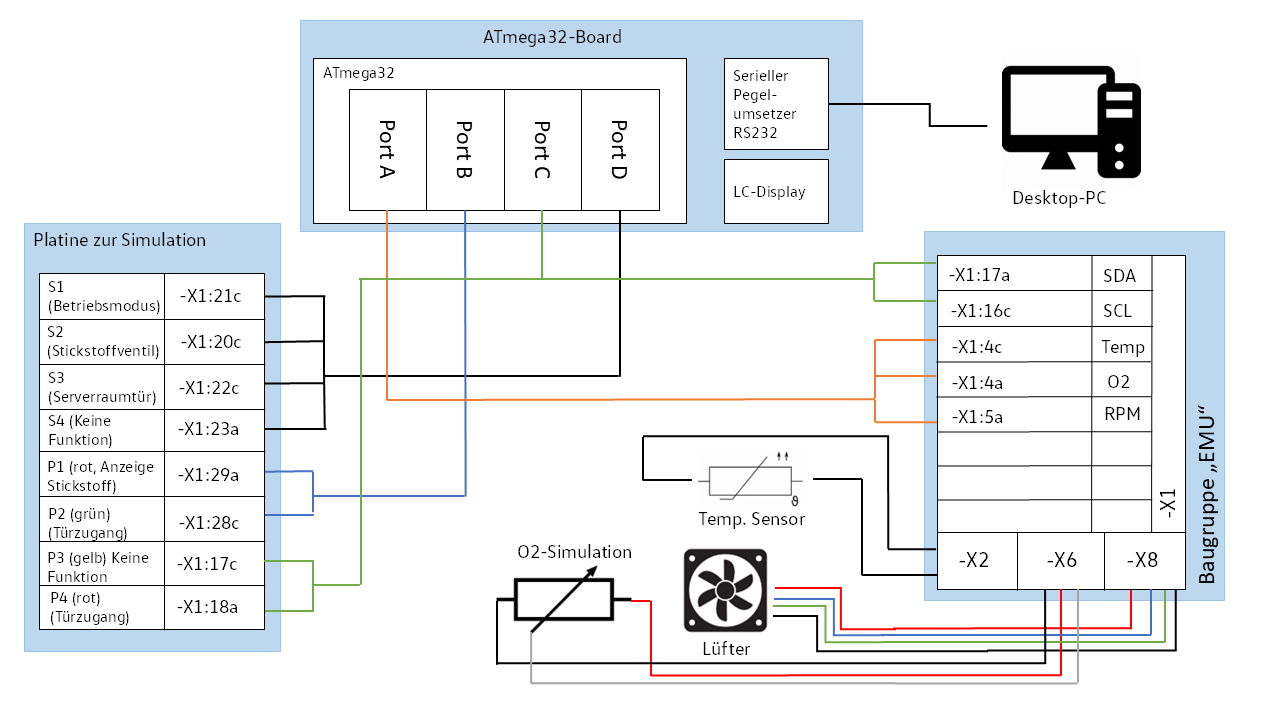
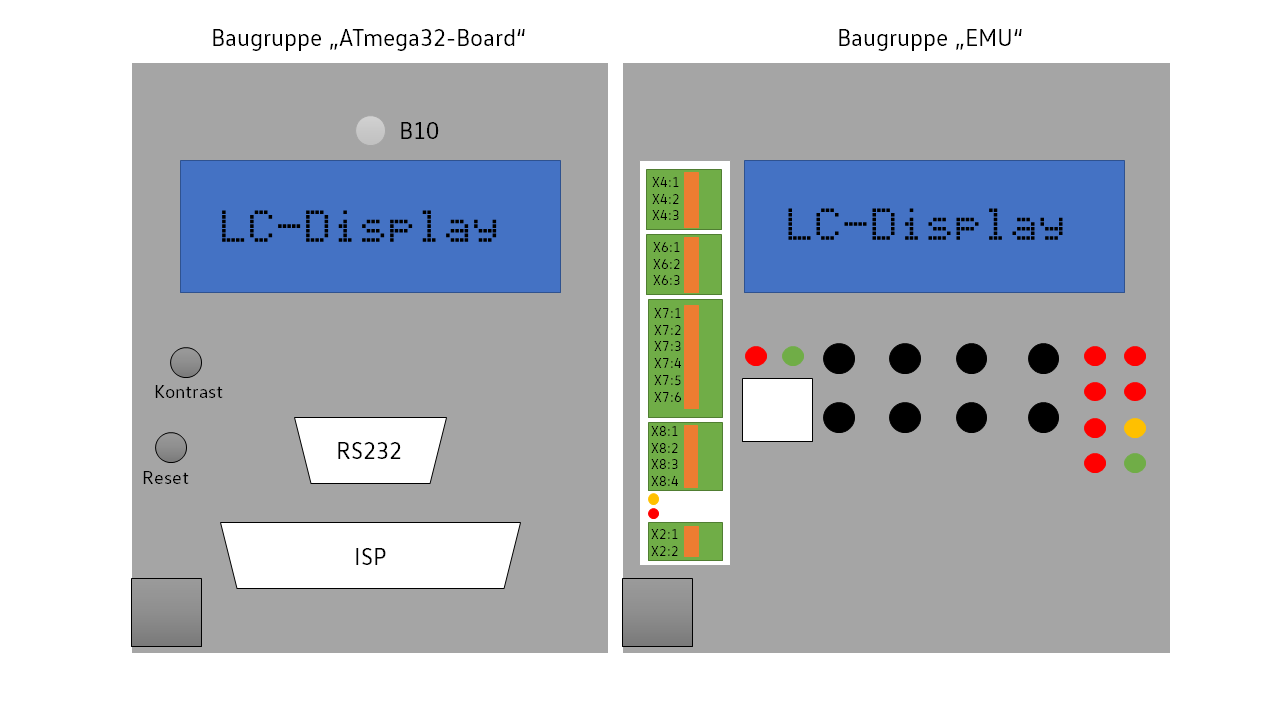


Abbildung : Technologieschema

## Übersicht Schnittstellen

Abbildung : Übersicht Schnittstellen



# Programmlösung

## Programmbeschreibung

Das Programm startet nach der Einbindung der „*main.h*“-Headerdatei mit der Deklaration der globalen Variablen, sowie der Initialisierung der Variable „*modus*“ mit dem Datentyp „*Betriebsmodus*“. Dieser kann zwei Zustände (AUTOMATIK oder MANUELL) annehmen. Daraufhin beginnt die Funktion *setup()*. In dieser Funktion werden die I²C-, LCD- und UART-Schnittstellen, sowie der ADC und der interne Timer initialisiert. Zudem werden die Ein- und Ausgänge der Ports festgelegt, die Sensoren zum ersten Mal ausgelesen. Anschließend wird eine Startmeldung auf dem LC-Display für 5 Sekunden ausgegeben.

Nach der Startsequenz befindet sich das Programm in einer Endlosschleife, in der kontinuierlich die aktuellen Sensorwerte (Temperatur, Sauerstoffgehalt und Lüftergeschwindigkeit) erfasst werden. Anhand des Sauerstoffgehalts wird über die Funktion *updateState()* der aktuelle Anlagenzustand in drei Bereiche (OPTIMAL, TOLERIERT, BEDROHLICH) eingeteilt. Anhand der Temperatur wird über die Funktion *updateState()* der aktuelle Temperaturzustand in drei Bereiche (OPTIMAL, TOLERIERT, BEDROHLICH) eingeteilt, wodurch der entsprechende Lüfter-Sollwert gesetzt wird. Zusätzlich wird das Stickstoffventil automatisch geöffnet oder geschlossen, sofern das System im Automatikmodus ist. Die LED P1 signalisiert dabei den Ventilstatus.

Über *setBetrieb()* lässt sich der Betriebsmodus über den Taster S1 umschalten. Im manuellen Betriebsmodus lässt sich das Stickstoffventil unabhängig vom Anlagenzustand über den Taster S2 öffnen oder schließen.

Mit der Funktion *setFanSpeed()* kann die Geschwindigkeit des Lüfters angepasst werden. Hierbei wird ein Wert mittels I²C-Modul dem DAC der EMU-Baugruppe zugeführt, welcher dann den Lüfter auf die Soll-Drehzahl stellt.

Die Funktionen *getTemperature(),* *getO2()* und *getFanSpeed()* lesen jeweils den Messwert des zugehörigen Sensors ein. Dabei wird ein analoges Spannungssignal in einen digitalen Wert umgewandelt und daraus der entsprechende physikalische Messwert berechnet.

Zur Ausgabe über den Monitor wird mit der Funktion *uartOutputTimed()* alle zwei Sekunden die aktuellen Messwerte und Zustände in Klartext über das Terminalprogramm „PuTTY“ ausgegeben.

Zur Laufzeitsteuerung wird ein interner 8-Bit-Timer verwendet, welcher regelmäßig über den Overflow-Vektor *TIMER0\_OVF\_VECT* einen Milisekundenzähler erhöht.

In der Funktion *updateState()* wird auch der Türzugang auf (erlaubt/untersagt) gesetzt und der Alarm wird de- und aktiviert.

In der Funktion setLEDs() wird bei der Aktivierung des Alarms wird das Stickstoffventil automatisch geschlossen und kann für die Dauer des Alarms nicht wieder geöffnet werden. Außerdem werden in dieser Funktion auch die LEDs P2 und P4 angesteuert und zwar sind diese abhängig von dem Türzugang (erlaubt LED\_Grün/untersagt LED\_Rot).

## Programmfunktionen

Tabelle 4: Liste der Programmfunktionen

|  |  |
| --- | --- |
| Unterprogramm | Aufgabe |
| main() | main-Funktion zum Aufruf der Startsequenz. Dauerschleife, um Messwerte auszulesen, Status aktualisieren, LEDs setzen, Betriebsmodus umschalten, Lüfter einzustellen, Türstatus aktualisieren, Türzugang aktualisieren und Monitorausgabe |
| setup() | Aufruf bei Programmstart und Initialisierung von I²C, UART, LCD, ADC und Timer. Ein- und Ausgänge definieren, erste Messwerte auslesen und Startbildschirm anzeigen |
| i2c\_init() | Initialisierung der I²C-Schnittstelle |
| lcd\_init() | Initialisierung des LC-Displays |
| ADC\_init() | Initialisierung des AD-Wandlers |
| UART\_init() | Initialisierung der UART-Schnittstelle |
| initExternalInterrupts() | Initialisierung der externen Interrupts |
| initTimerInterrupts() | Initialisierung des Timers |
| getTemperature(uint8\_t sensor) | Auslesen der Spannung, Umwandlung in einen digitalen Wert und umrechnen in Temperaturwert |
| getO2(uint8\_t sensor) | Auslesen der Spannung, Umwandlung in einen digitalen Wert und umrechnen in Sauerstoffgehaltswert |
| getFanSpeed(uint8\_t sensor) | Auslesen der Spannung, Umwandlung in einen digitalen Wert und umrechnen in Lüftergeschwindigkeit |
| millis() | Laufzeitsteuerung |
| mcp4725\_sendFast(adressDAC1, dacValue) | Übertragen des DAC-Wertes zur Einstellung der Lüfter-Soll-Drehzahl |
| setBetrieb() | Wechselt den Betriebsmodus bei Betätigung des Tasters S1 |
| setLEDs() | Setzt die LEDs auf vordefinierte Fälle |
| setFanSpeed() | Übergibt einen DAC-Wert in Abhängigkeit der Lüfter-Soll-Geschwindigkeit |
| updateMeasurements() | Liest alle Sensoren aus und setzt die globalen Variablen auf die aktuellen Werte |
| updateState() | Setzt den Anlagenstatus, Temperaturstatus, Türzugangsstatus und den Alarmstatus |
| updateVentil() | Wechselt den Zustand des Stickstoffventils |
| uartOutputTimed() | Sorgt für einen Anzeigenwechsel alle 2 Sekunden |
| uartOutput() | Aktualisiert die Ausgabe über UART |
| ISR(TIMER0\_OVF\_vect) | Laufzeitsteuerung |
| ISR(INT0\_vect) | Externes Interrupt für Taster S4 |
| ISR(INT1\_vect) | Der Taster S3 öffnet und schließt die Tür |

## Programmablaufpläne

### Hauptprogramm

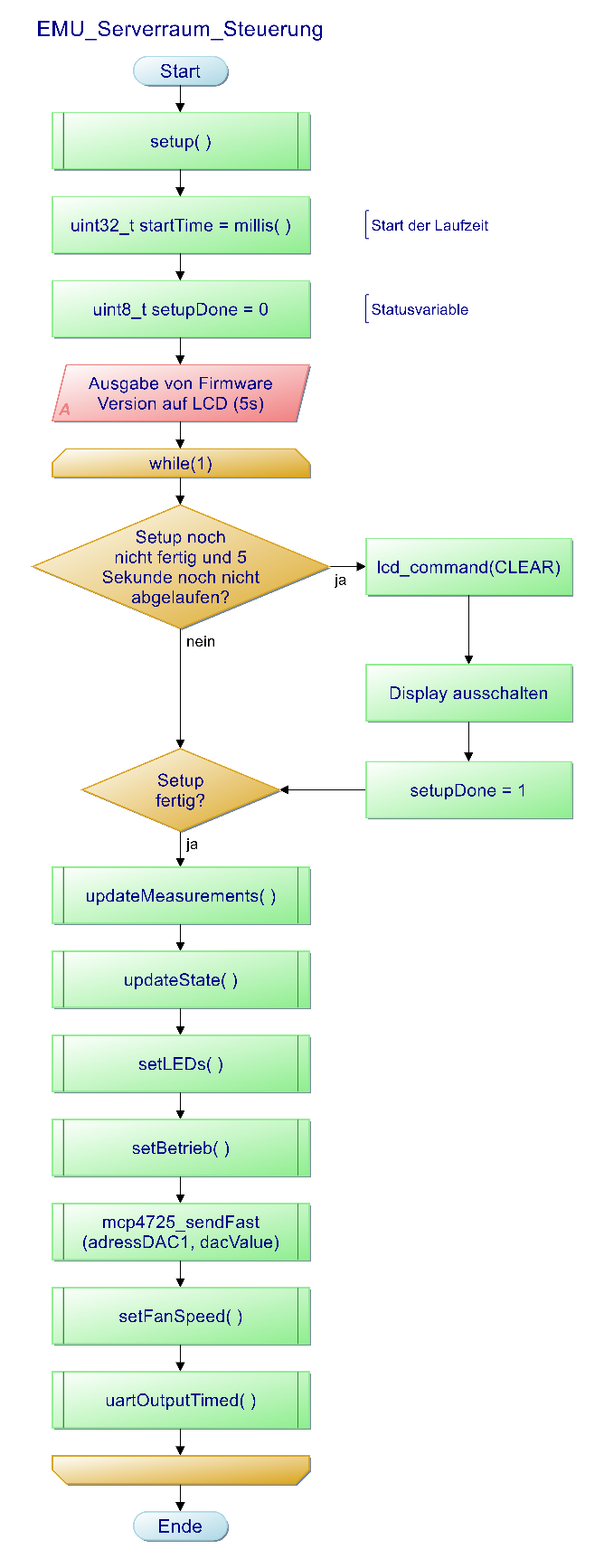


Abbildung : Programmablaufplan für die main-Funktion

### Setup



Abbildung : Programmablaufplan für setup( )

### Initialisierung der externen Interrupts

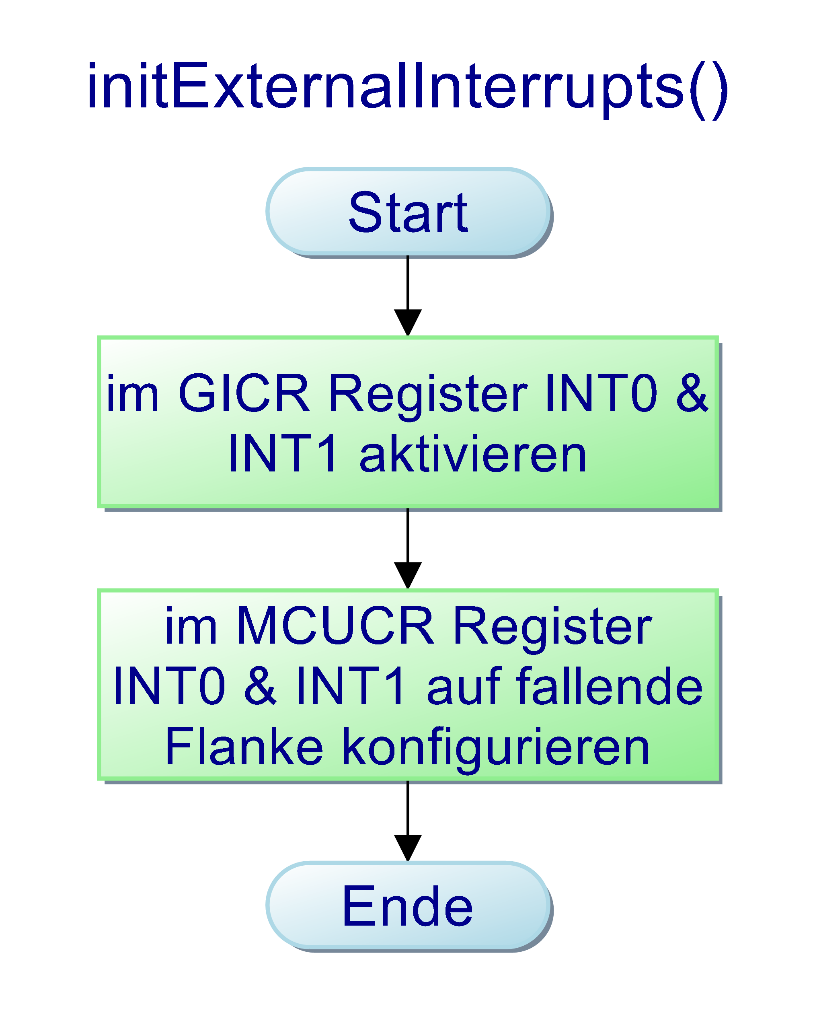


Abbildung : Programmablaufplan für initExternalInterrupts( )

### Initialisierung des Timer Interrupts

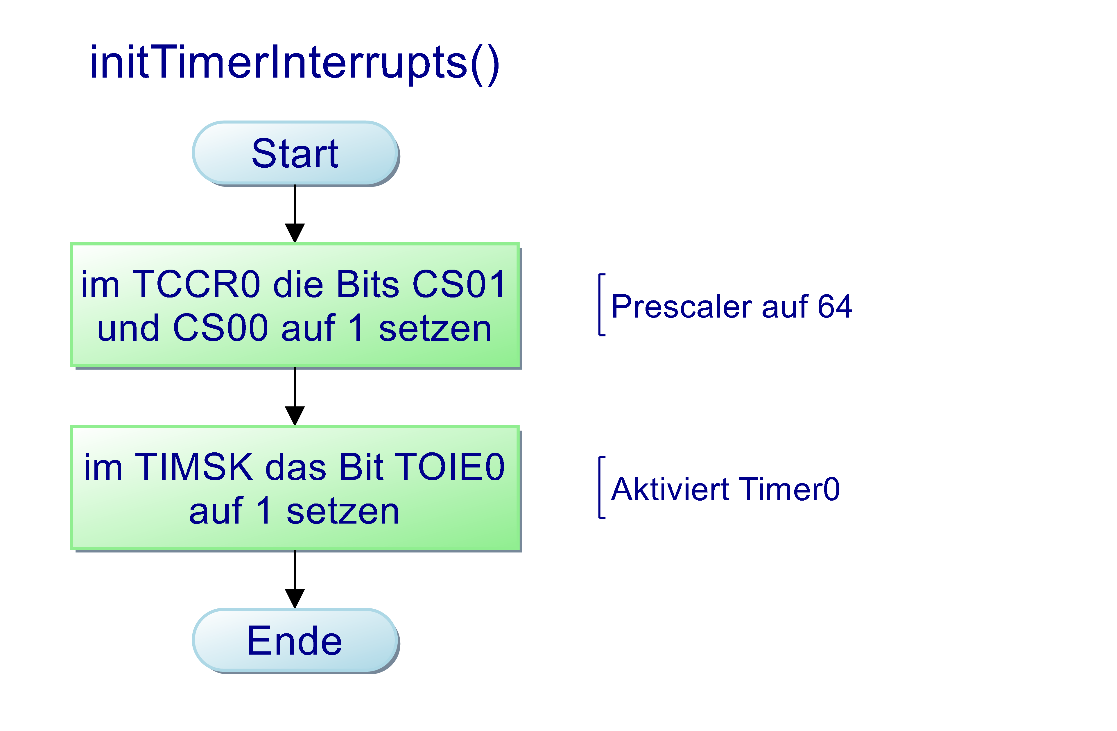


Abbildung : Programmablaufplan für initTimerInterrupts( )

### Messwerte aktualisieren



Abbildung : Programmablaufplan für updateMeasurements( )

### Anlagenstatus aktualisieren

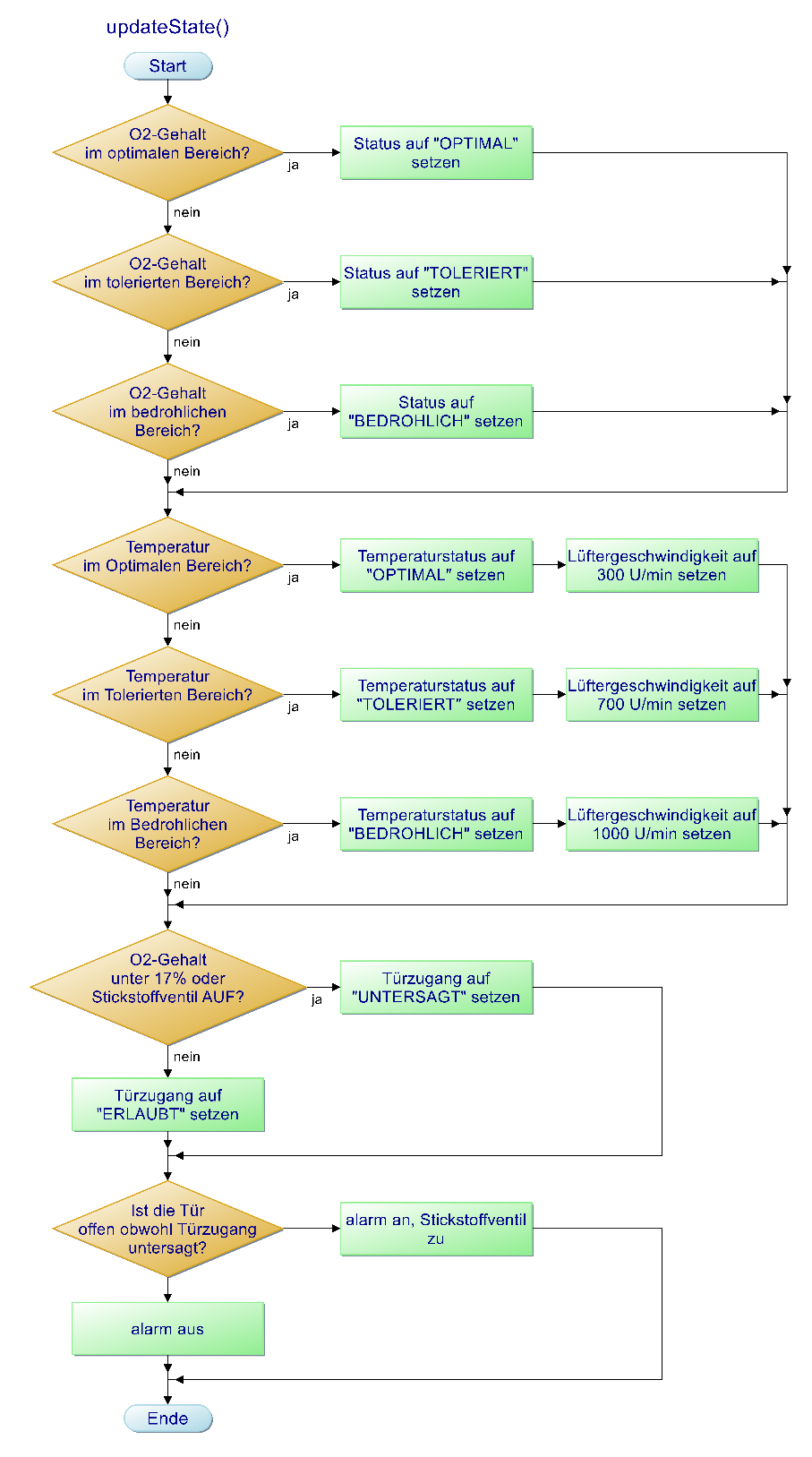


Abbildung : Programmablaufplan für updateState( )

### Setzen der LEDs

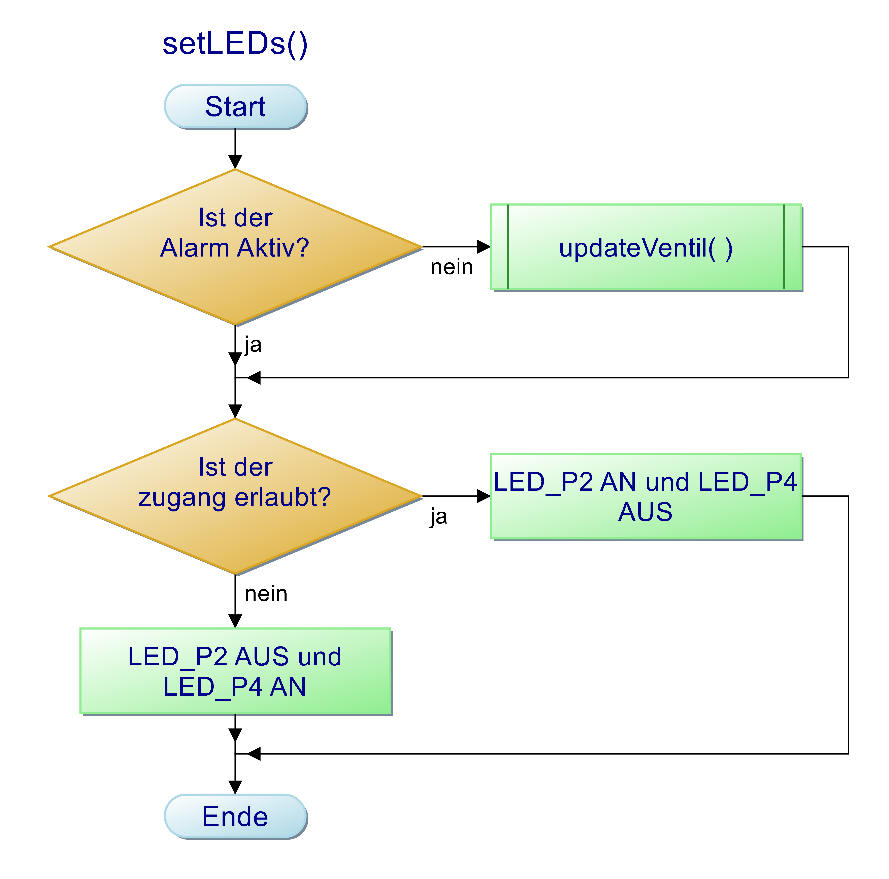


Abbildung : Programmablaufplan für setLEDs( )

### Umschalten des Betriebsmodus

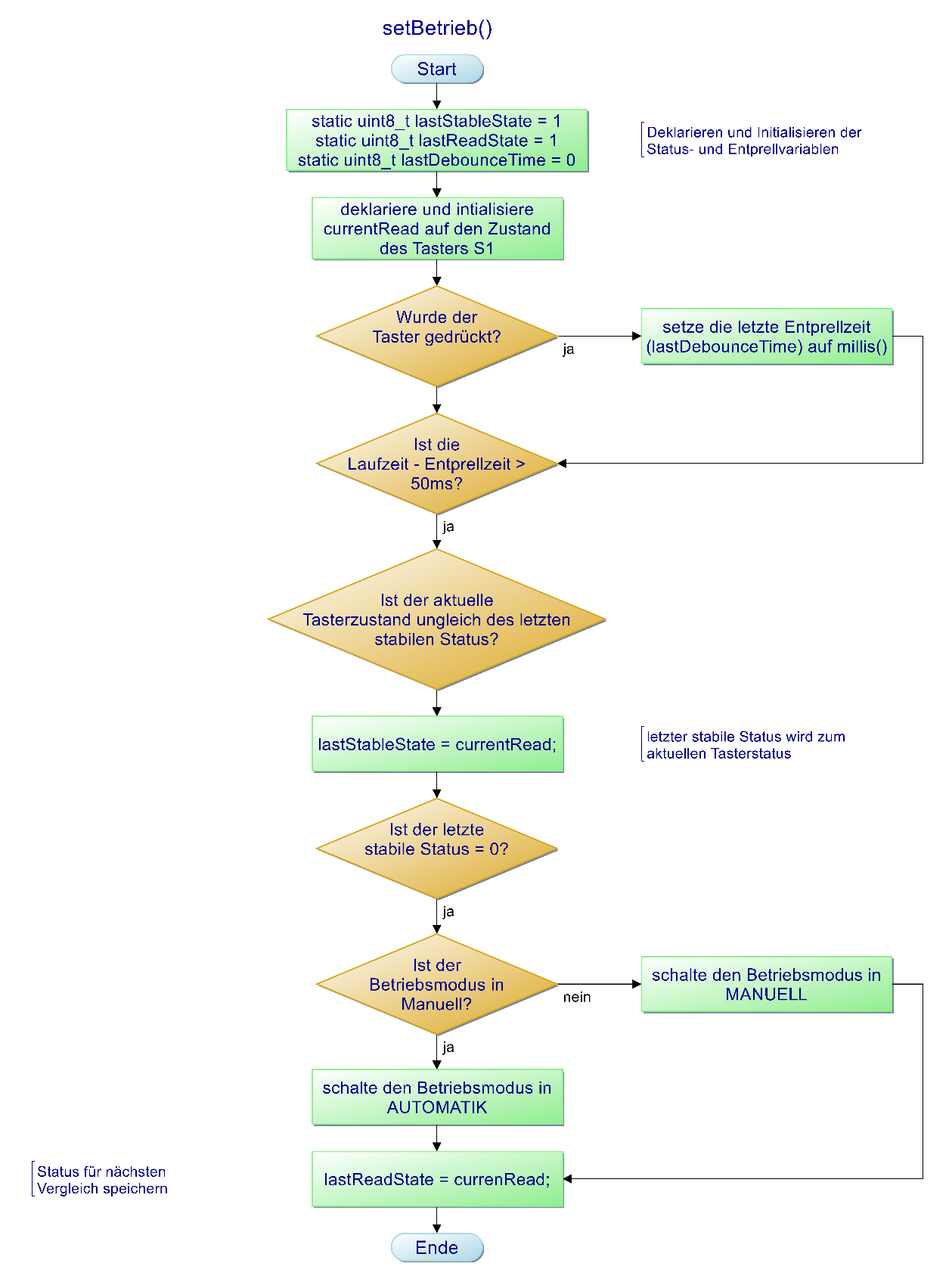
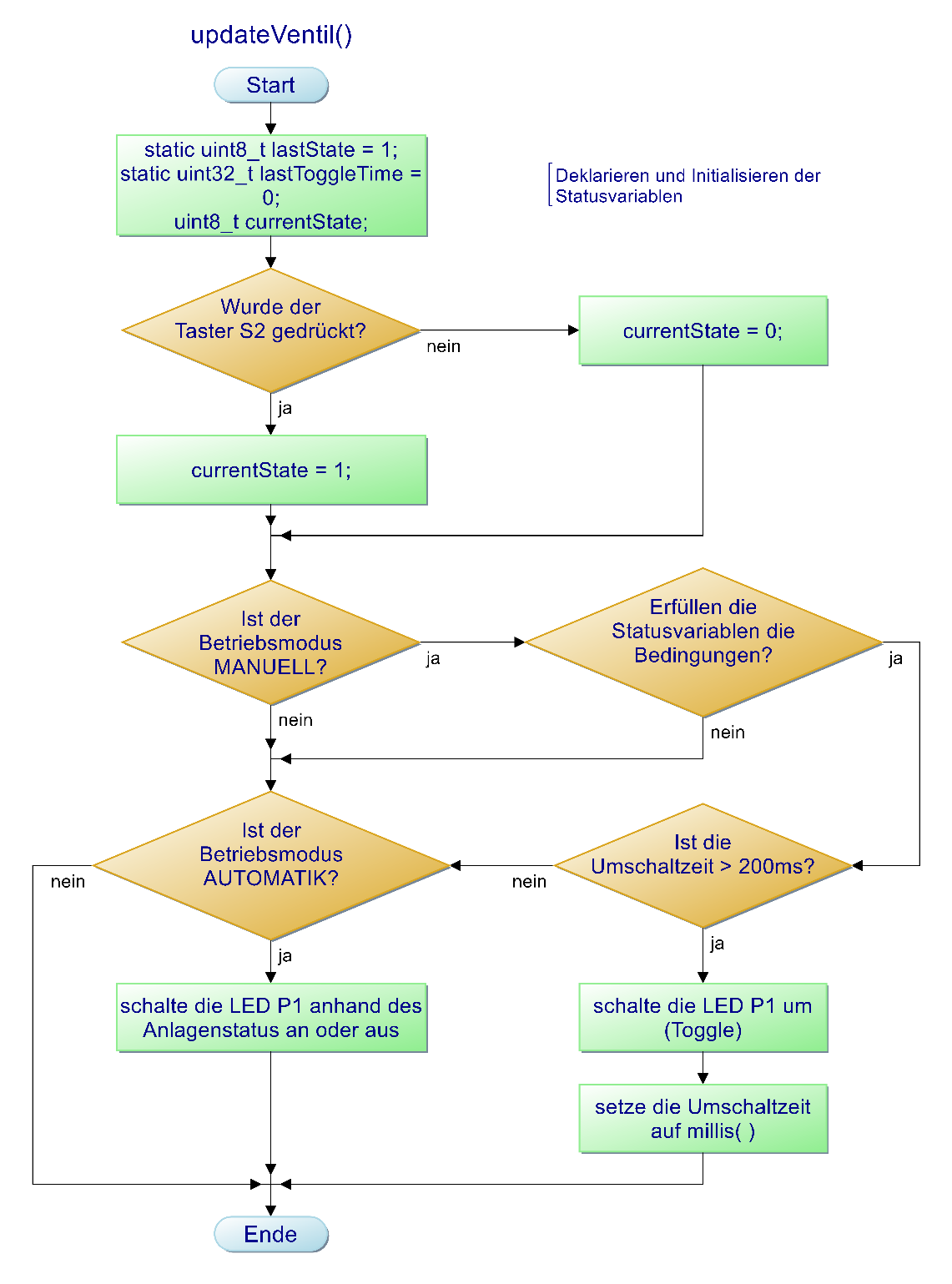


Abbildung : Programmablaufplan für setBetrieb( )

### Umschalten des Stickstoffventils

Abbildung : Programmablaufplan für updateVentil( )



### Laufzeitsteuerung

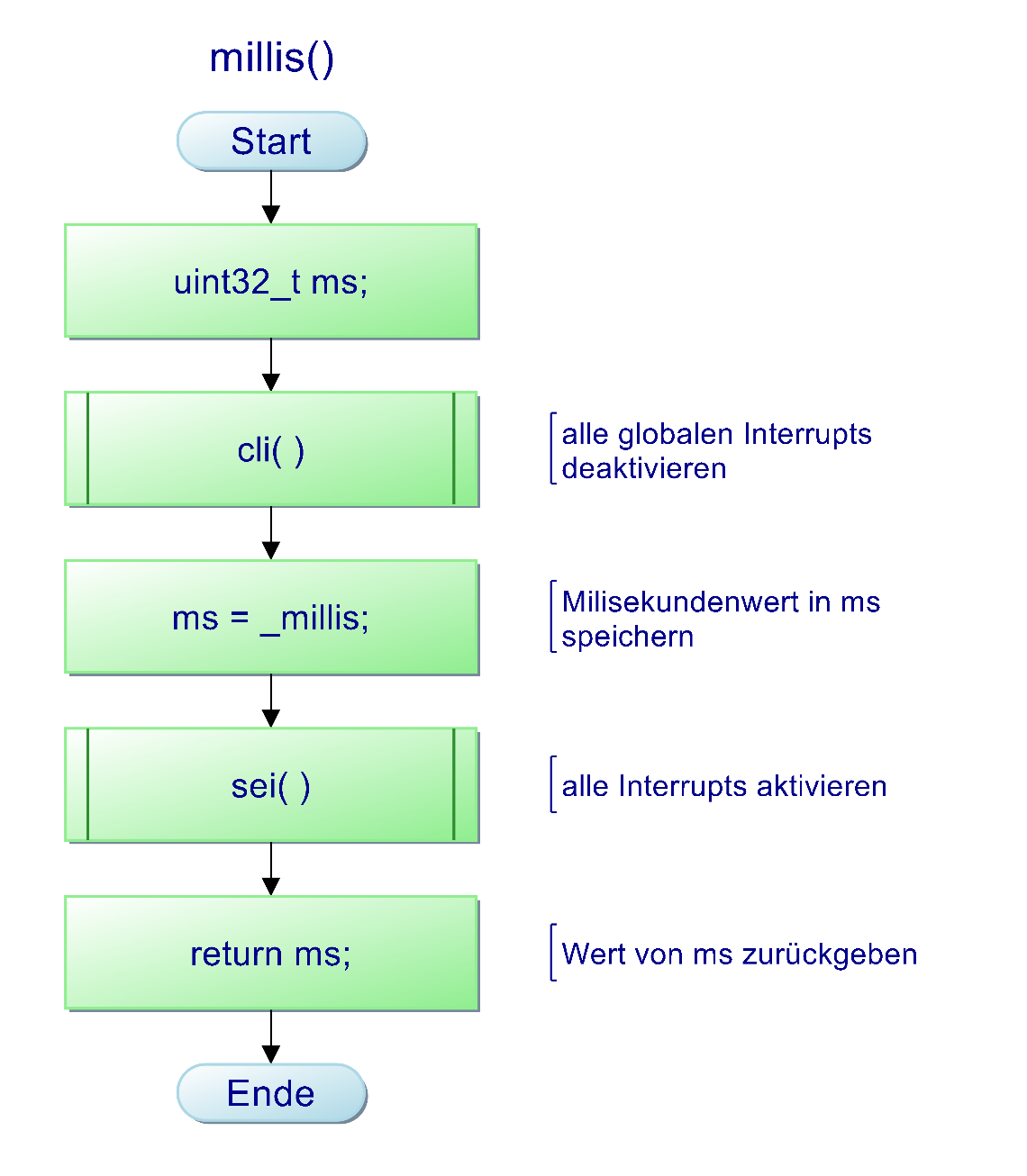


Abbildung : Programmablaufplan für millis( )

### Timer0 ISR



Abbildung : Programmablaufplan für ISR(TIMER0\_OVF\_vect)

### Setzen der Lüftergeschwindigkeit

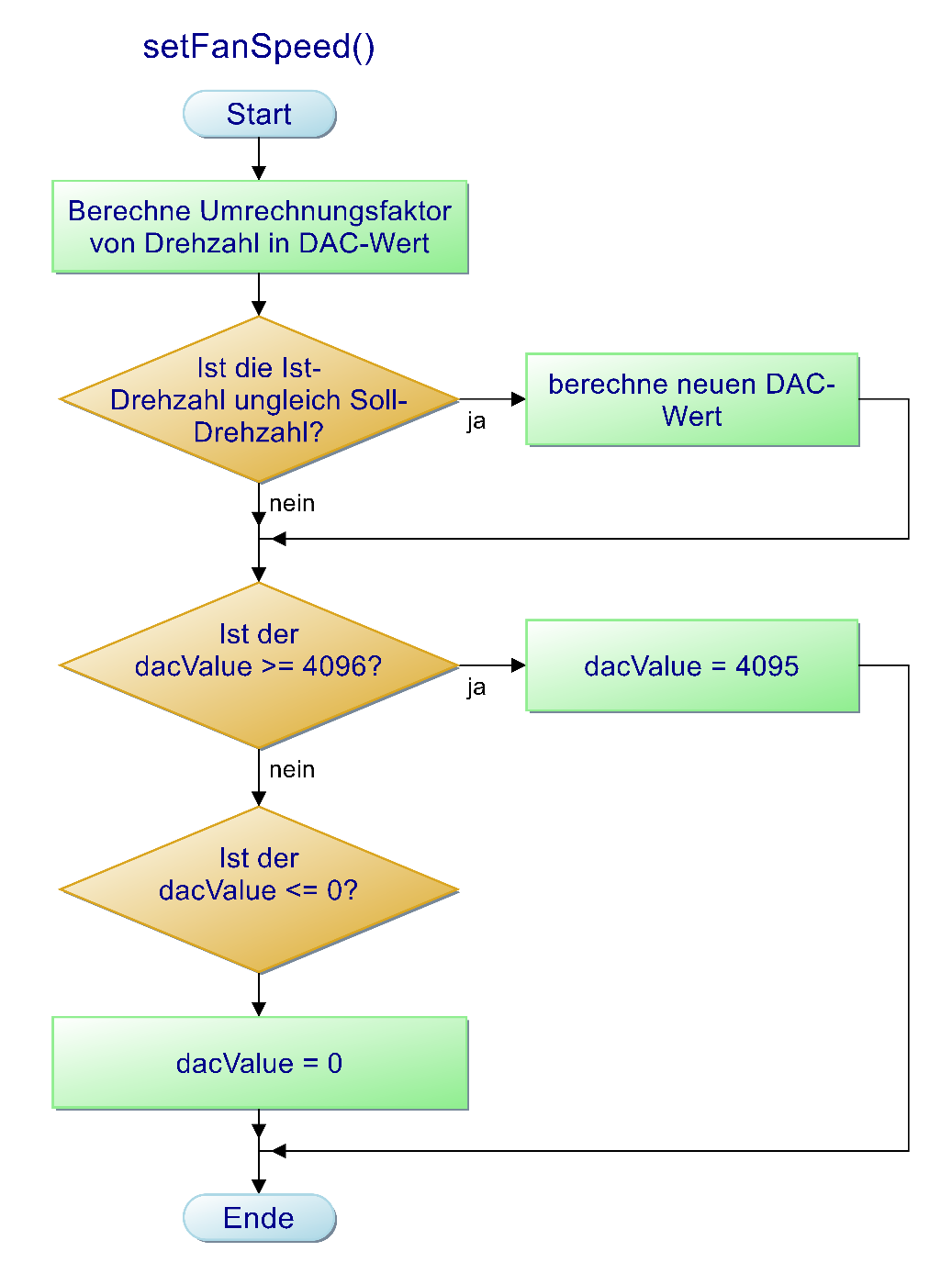


Abbildung : Programmablaufplan für setFanSpeed( )

### Ausgabe über UART

Abbildung : Programmablaufplan für uartOutput( )

### Auslesen der Temperatur

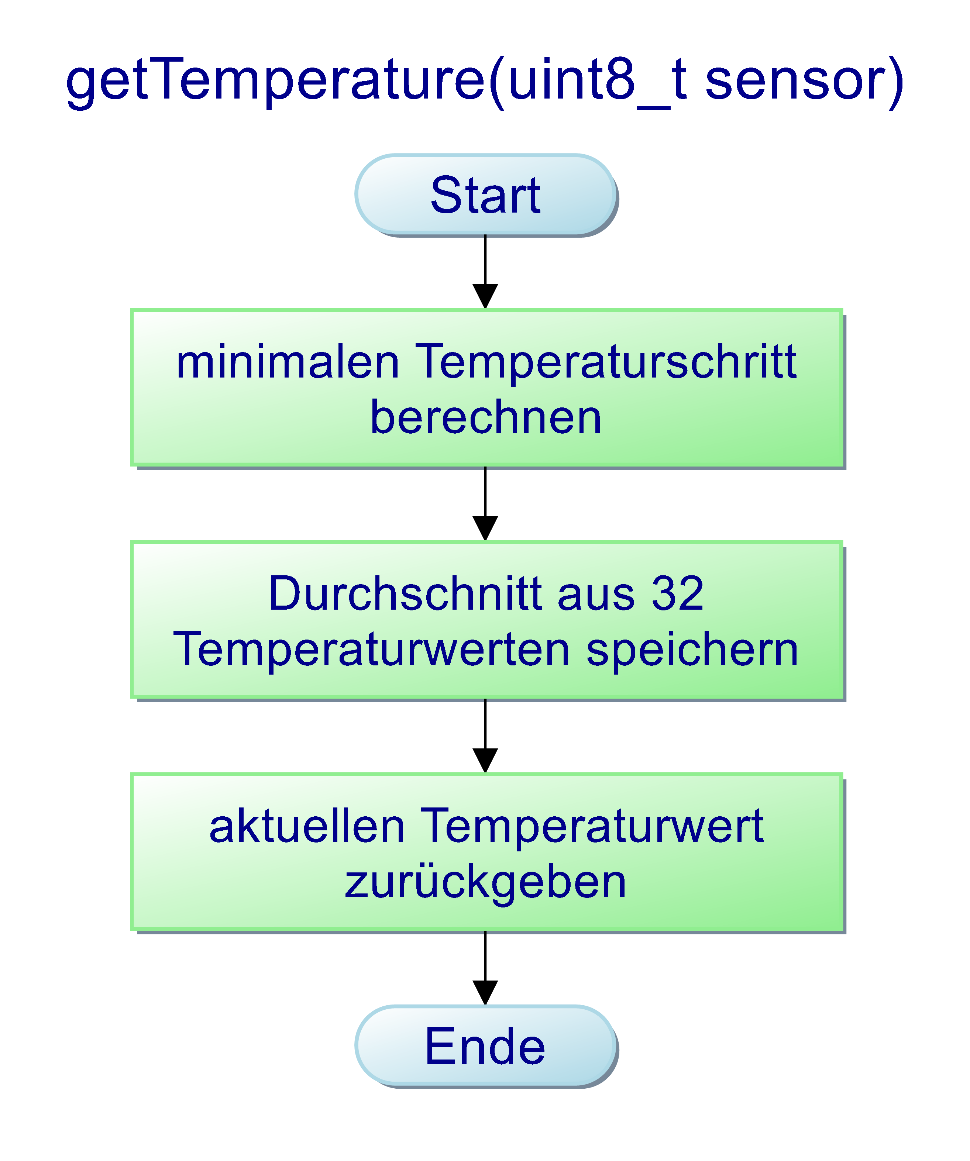


Abbildung : Programmablaufplan für getTemperature(uint8\_t sensor)

### Auslesen des Sauerstoffgehalts

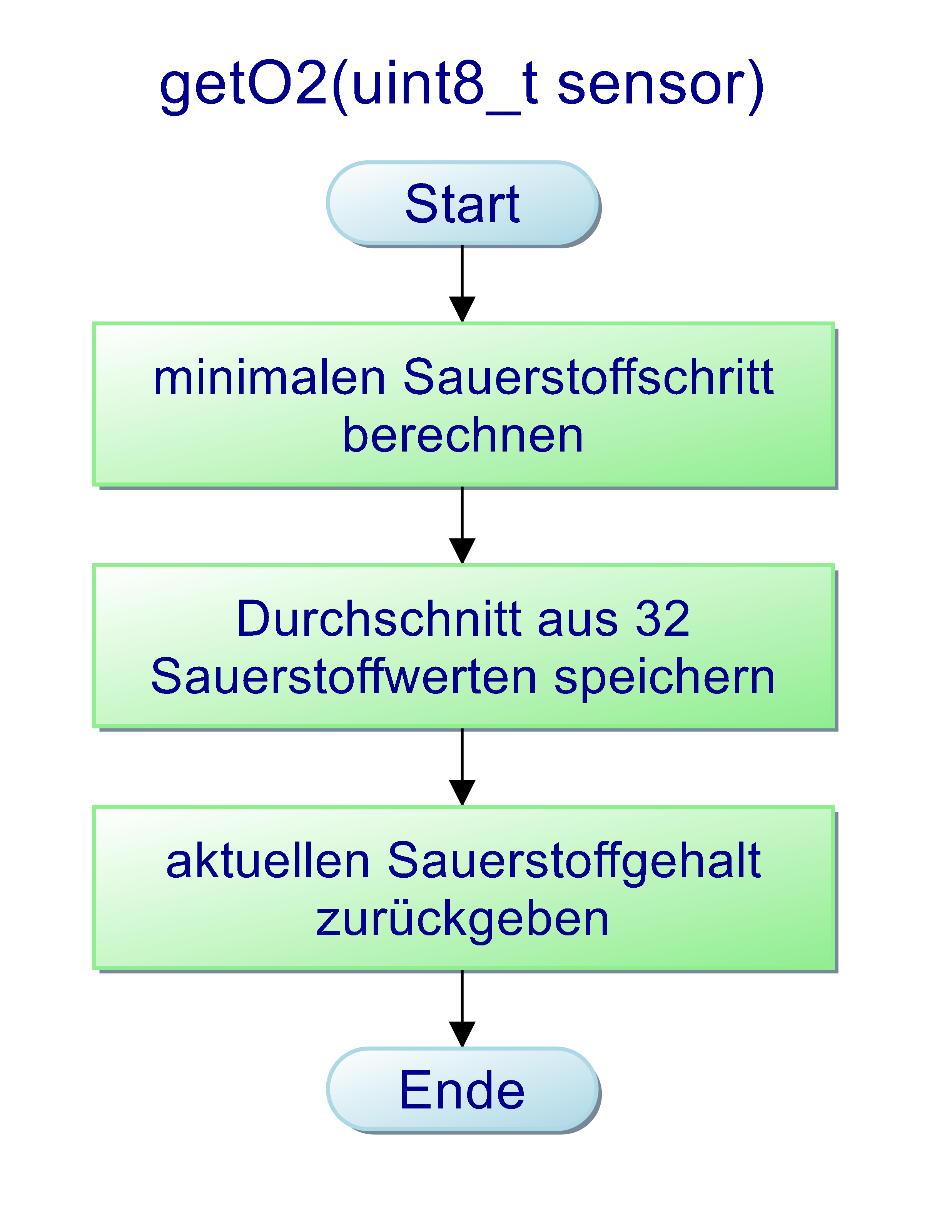


Abbildung : Programmablaufplan für getO2(uint8\_t sensor)

### Serverraumtür bedienen

Abbildung ISR(INT1\_vect)

### Auslesen der Lüftergeschwindigkeit

Abbildung : Programmablaufplan für getFanSpeed(uint8\_t sensor)

## Programmcode

### main.c

/\*

\* EMU\_Serverraum\_Steuerung.c

\*

\* Created: 14.04.2025 07:38:58

\* Author : Kaufrechner19

\*/

#include "main.h" //Einbinden der main.h

//Deklarieren globaler Variablen

volatile *uint32\_t* \_millis = 0;

float temperatur;

float o2;

*uint8\_t* state = 0;

*uint16\_t* fanSpeed; //Lüfter Ist-Drehzahl

*uint16\_t* fanSpeedSetpoint = 0; //Lüfter Soll-Drehzahl

*uint16\_t* dacValue = 1; //Steuert die Lüftergeschwindigkeit

*uint8\_t* tuerKontakt = 0;

*uint8\_t* tempState = 0;

*uint8\_t* tuerZugang = 0;

*uint8\_t* alarm = 0;

volatile Betriebsmodus modus = AUTOMATIK; //Variable modus mit Datentyp Betriebsmodus / Standardwert: AUTOMATIK

///////////////////////////////MAIN//////////////////////////////////////

int main(void)

{

setup(); //Initialisierung aller Module, Timer, Interrupts, Ein- und Ausgänge

*uint32\_t* startTime = millis(); //Start der Laufzeit

*uint8\_t* setupDone = 0; //Statusvariable

lcd\_string("EMU Serverraum"); // Erste Zeile

lcd\_command(FORCE\_SND\_LINE); // Cursor auf zweite Zeile setzen

lcd\_string("Steuerung\_V2"); // Zweite Zeile

while (1)

{

if (!setupDone && millis() - startTime >= 5000) //Überprüfen, ob das Setup schon durchgeführt wurde

{

lcd\_command(CLEAR);

lcd\_command(0x08); // Display ausschalten

setupDone = 1;

}

if (setupDone){ //Nach Startvorgang, Dauerschleife

updateMeasurements();

updateState();

setLEDs();

setBetrieb();

mcp4725\_sendFast(adressDAC1, dacValue); //Ansteuerung des Lüfters über I2C

setFanSpeed();

uartOutputTimed();

}

}

}

//////////////////////////////////////////////////////////////////////////

void setup(){

//Initialisieren der Module

i2c\_init();

lcd\_init();

ADC\_Init();

UART\_init();

//Interrupts initialisieren und aktivieren

initExternalInterrupts();

initTimerInterrupts();

DDRC &= ~((1 << PC0) | (1 << PC1));

PORTC |= (1 << PC0) | (1 << PC1); //PullUp Widerstand auf I2C-Leitung

DDRB |= (1 << LED\_P1) | (1 << LED\_P2); //LED Pins auf Output setzen

DDRC |= (1 << LED\_P3) | (1 << LED\_P4);

//Variablen mit Messwerte füllen

temperatur = getTemperature(TEMP\_SENSOR);

o2 = getO2(O2\_SENSOR);

fanSpeed = getFanSpeed(LUEFTER\_SENSOR);

// Alle Taster als Eingang mit PullUp aktivieren

DDRD &= ~((1 << TASTER\_S1) | (1 << TASTER\_S2) | (1 << TASTER\_S3) | (1 << TASTER\_S4));

PORTD |= (1 << TASTER\_S1) | (1 << TASTER\_S2) | (1 << TASTER\_S3) | (1 << TASTER\_S4);

}

//nicht blockierende Laufzeitsteuerung

*uint32\_t* millis(){

*uint32\_t* ms;

cli(); //Interrupts aus, um sicheren Zugriff zu ermöglichen

ms = \_millis; //Milisekundenwert speichern

sei(); //Set Enable Interrupts

return ms; //Zeit in ms zurückgeben

}

void initExternalInterrupts(){

GICR |= (1 << INT0) | (1 << INT1); // Externe Interrupts INT0 & INT1 aktivieren

MCUCR |= (1 << ISC11); // INT1 bei fallender Flanke

MCUCR |= (1 << ISC01); // INT0 bei fallender Flanke

}

void initTimerInterrupts(){

TCCR0 = (1 << CS01) | (1 << CS00); // Prescaler 64

TIMSK |= (1 << TOIE0); // Overflow-Interrupt an

}

void setBetrieb() {

//Status- und Entprellvariablen deklarieren und initialisieren

static *uint8\_t* lastStableState = 1;

static *uint8\_t* lastReadState = 1;

static *uint32\_t* lastDebounceTime = 0;

*uint8\_t* currentRead = PIND & (1 << TASTER\_S1); //currentRead auf aktuellen Tasterstatus setzen

if (currentRead != lastReadState) {

lastDebounceTime = millis(); // potenzieller Wechsel erkannt

}

// Entprellzeit prüfen

if ((millis() - lastDebounceTime) > 50) {

if (currentRead != lastStableState) {

lastStableState = currentRead;

if (lastStableState == 0) { // Taster gedrückt (fallende Flanke)

// Modus umschalten

if (modus == MANUELL) {

modus = AUTOMATIK;

} else {

modus = MANUELL;

}

}

}

}

lastReadState = currentRead; // Status für nächsten Vergleich speichern

}

void setLEDs(){

//setze alle vorhandenen LEDs auf definierte Fälle

if (alarm == 0)

{

updateVentil(); //überprüfe und ggf. wechsele Zustand des Stickstoffventils

}

if (tuerZugang == 1)

{

PORTB |= (1<<LED\_P2);

PORTC &= ~(1<<LED\_P4);

}

else if (tuerZugang == 0)

{

PORTB &= ~(1<<LED\_P2);

PORTC |= (1<<LED\_P4);

}

}

void setFanSpeed(){

float fanstep = 4095.0 / 1350.0; // Umrechnungsfaktor: RPM ? DAC-Wert

// Nur aktualisieren, wenn Ist-Wert ungleich Soll-Wert

if (fanSpeed < fanSpeedSetpoint || fanSpeed > fanSpeedSetpoint) {

dacValue = (*uint16\_t*)(fanSpeedSetpoint \* fanstep + 0.5f); // Berechnen und aufrunden

}

// DAC-Wert begrenzen

if (dacValue >= 4096) {

dacValue = 4095;

} else if (dacValue <= 0) {

dacValue = 0;

}

}

void updateMeasurements(){

temperatur = getTemperature(TEMP\_SENSOR); //lies den Temperatursensor aus und speichere ihn in der globalen Variable

o2 = getO2(O2\_SENSOR); //lies den Sauerstoffsensor aus und speichere ihn in der globalen Variable

fanSpeed = getFanSpeed(LUEFTER\_SENSOR); //lies die Lüftergeschwindigkeit aus und speichere ihn in der globalen Variable

}

void updateState() {

if (o2 >= 0 && o2 <= O2\_OPTIMAL\_MAX) { //Ist der Sauerstoffgehalt zwischen 0 und 15 %?

state = OPTIMAL; //Stelle Anlagenstatus auf OPTIMAL

}

else if (o2 >= O2\_TOLERIERT\_MIN && o2 <= O2\_TOLERIERT\_MAX) { //Ist der Sauerstoffgehalt zwischen 15 und 17 %?

state = TOLERIERT; //Stelle Anlagenstatus auf TOLERIERT

}

else if (o2 >= O2\_BEDROHLICH\_MIN && o2 <= 100) { //Ist der Sauerstoffgehalt zwischen 17 und 100 %?

state = BEDROHLICH; //Stelle Anlagenstatus auf BEDROHLICH

}

//Temperatur und Lüfter

if (temperatur < TEMP\_OPTIMAL\_MAX)

{

tempState = OPTIMAL;

fanSpeedSetpoint = 300;

}

else if (temperatur >= TEMP\_TOLERIERT\_MIN && temperatur <= TEMP\_TOLERIERT\_MAX)

{

tempState = TOLERIERT;

fanSpeedSetpoint = 700;

}

else if (temperatur >= TEMP\_BEDROHLICH\_MIN)

{

tempState = BEDROHLICH;

fanSpeedSetpoint = 1000;

}

//Türzugang

if (o2 <= O2\_TOLERIERT\_MAX || (PORTB &= (1 << LED\_P1)))

{

tuerZugang = UNTERSAGT;

}

else

{

tuerZugang = ERLAUBT;

}

//Alarm

if (tuerKontakt == 1 && tuerZugang == 0)

{

alarm = 1;

PORTB &= ~(1 << LED\_P1); // Ventil zu

}

else

{

alarm = 0;

}

}

void updateVentil() {

//Status- und Umschaltvariablen deklarieren und initialisieren

static *uint8\_t* lastState = 1;

static *uint32\_t* lastToggleTime = 0;

*uint8\_t* currentState;

//Überprüfe ob der Taster S2 gedrückt wurde

if (PIND & (1 << TASTER\_S2)) {

currentState = 1; // Taster ist NICHT gedrückt (HIGH)

} else {

currentState = 0; // Taster ist gedrückt (LOW)

}

//Überprüfe ob das Stickstoffventil im Manuell oder Automatik geändert wird

if (modus == MANUELL) {

if (lastState == 1 && currentState == 0) {

if (millis() - lastToggleTime >= 200)

{

// Fallende Flanke ? Taster gedrückt

PORTB ^= (1 << LED\_P1); // Toggle Ventil-LED

lastToggleTime = millis();

}

}

} else if (modus == AUTOMATIK) { // Stickstoffventil anhand Anlagenstatus im Automatikmodus umstellen

switch (state) {

case OPTIMAL:

PORTB &= ~(1 << LED\_P1); // Ventil zu

break;

case TOLERIERT:

PORTB &= ~(1 << LED\_P1); // Ventil zu

break;

case BEDROHLICH:

PORTB |= (1 << LED\_P1); // Ventil auf

break;

}

}

}

//alle 2s soll die UART-Ausgabe aktualisiert werden

void uartOutputTimed(){

static *uint32\_t* lastUartTime = 0;

*uint32\_t* now = millis();

if (now - lastUartTime >= 2000)

{

uartOutput();

lastUartTime = now;

}

}

void uartOutput(void) {

char temp[15]; //Zeichen-Array für Umwandelung von Zahlen in String

// ANSI Escape: Cursor 10 Zeilen hoch und Zeile löschen

// (10, weil wir 10 Zeilen Inhalt haben)

UART\_send\_string("\033[10A"); // Cursor 10 Zeilen hoch

UART\_send\_string("\033[J"); // Inhalt ab hier löschen

// Trennlinie oben für eine klare Trennung

UART\_send\_string("----------------------\n\r");

//Sauerstoffgehaltsbereich

UART\_send\_string("Sauerstoffgehalt: ");

if (state == OPTIMAL) {

UART\_send\_string("< 15 %\n\r");

} else if (state == TOLERIERT) {

UART\_send\_string("15 % bis 17 %\n\r");

} else if (state == BEDROHLICH) {

UART\_send\_string("> 17 %\n\r");

}

// Temperaturbereich

UART\_send\_string("Temperatur: ");

if (temperatur < TEMP\_OPTIMAL\_MAX) {

UART\_send\_string("< 15 Grad\n\r");

} else if (temperatur >= TEMP\_TOLERIERT\_MIN && temperatur <= TEMP\_TOLERIERT\_MAX) {

UART\_send\_string("15 Grad bis 28 Grad\n\r");

} else if (temperatur >= TEMP\_BEDROHLICH\_MIN) {

UART\_send\_string("> 28 Grad\n\r");

}

// Lüftergeschwindigkeit

UART\_send\_string("Luefter: ");

*dtostrf*(fanSpeed, 3, 0, temp);

UART\_send\_string(temp);

UART\_send\_string(" U/min\n\r");

// Modus

UART\_send\_string("Modus: ");

if (modus == MANUELL) {

UART\_send\_string("MANUELL\n\r");

} else {

UART\_send\_string("AUTOMATIK\n\r");

}

// Ventilstatus (LED\_P1 gibt den Zustand an)

UART\_send\_string("Stickstoffventil: ");

if (PORTB & (1 << LED\_P1)) {

UART\_send\_string("AUF\n\r");

} else {

UART\_send\_string("ZU\n\r");

}

// Türzugang

UART\_send\_string("Tuerzugang: ");

if (tuerZugang == 1) {

UART\_send\_string("erlaubt\n\r");

} else {

UART\_send\_string("untersagt\n\r");

}

// Türstatus

UART\_send\_string("Tuerstatus: ");

if (tuerKontakt == 1) {

UART\_send\_string("AUF\n\r");

} else {

UART\_send\_string("ZU\n\r");

}

// Alarmstatus

if (alarm == 1)

{

// Platzhalter

UART\_send\_string("\n");

UART\_send\_string("Alarm: Tuer unerlaubt geoeffnet!\n\r");

}

// Abschluss-Trennlinie

UART\_send\_string("----------------------\r");

}

float getO2(*uint8\_t* sensor){

float o2\_step = 100.0 / 1023.0; //Maximaler Sauerstoffwert aufgeteilt auf 10 Bit ADC

float sauerstoff = ADC\_ReadAvgFloat(sensor); //Durchschnittswert aus 32 Sensorwerten auslesen und in o2 speichern

return (sauerstoff \* o2\_step); //Sensorwert \* Sauerstoffschritt zurückgeben = aktueller Sauerstoffgehalt

}

float getTemperature(*uint8\_t* sensor){

float temp\_step = 100.0 / 1023.0; //Maximaler Temperaturwert aufgeteilt auf 10 Bit ADC

float temp = ADC\_ReadAvgFloat(sensor); //Durchschnittswert aus 32 Sensorwerten auslesen und in temp speichern

return (temp \* temp\_step); //Sensorwert \* Temperaturwert zurückgeben = aktueller Temperaturwert

}

*uint16\_t* getFanSpeed(*uint8\_t* sensor) {

float fan\_step = 1350.0f / 1023.0f; // Umrechnungsfaktor: ADC-Wert (0–1023) in RPM (0–1350)

*uint16\_t* adc = ADC\_Read(sensor); // ADC-Wert vom Sensor einlesen

*uint16\_t* result = adc \* fan\_step; // Umrechnen in Drehzahl (RPM)

//Rückgabe der berechneten Drehzahl

return result;

}

//Millis-Funktion hochzählen

ISR(TIMER0\_OVF\_vect){

\_millis++;

}

ISR(INT1\_vect)

{

tuerKontakt = (tuerKontakt == 0) ? 1:0;

}

ISR(INT0\_vect)

{

}

### main.h

/\*

\* main.h

\*

\* Created: 14.04.2025 10:50:41

\* Author: Kaufrechner19

\* Beschreibung: Diese .h-Datei enthält alle einzubindenden Dateien und Definitionen für den Preprozessor

\*/

#pragma once //Inhalt der main.h Datei wird nur einmal kompiliert

// Definition der CPU-Frequenz

#ifndef F\_CPU

#define F\_CPU 16000000UL

#endif

// Einbinden der main.h-Dateien

#ifndef MAIN\_H\_

#define MAIN\_H\_

#include <avr/io.h>

#include <avr/interrupt.h>

#include <util/delay.h>

#include <util/twi.h>

#include <stdlib.h>

#include <string.h>

#include <inttypes.h>

#include <stdbool.h>

#include "adc.h"

#include "lcd.h"

#include "i2c.h"

#include "UART.h"

#endif /\* MAIN\_H\_ \*/

// Definitionen der Portpins

#define O2\_SENSOR PA0 //Anschluss an X1:4a

#define TEMP\_SENSOR PA1 //Anschluss an X1:4c

#define LUEFTER\_SENSOR PA2 //Anschluss an X1:5a

#define SCL PC0 //Anschluss an X1:16c

#define SDA PC1 //Anschluss an X1:17a

#define TASTER\_S1 PD5 //Wechsel Manuell- / Automatikbetrieb

#define TASTER\_S2 PD7 //Wechsel Stickstoffventil (geöffnet oder geschlossen)

#define TASTER\_S3 PD3 //INT1

#define TASTER\_S4 PD2 //INT0

#define LED\_P1 PB0 //LED rot (Anzeige Stickstoff (geöffnet oder geschlossen)

#define LED\_P2 PB3 //LED grün

#define LED\_P3 PC2 //LED gelb

#define LED\_P4 PC3 //LED rot

//Definition der Bereiche anhand Sauerstoffgehalt und Temperatur im Serverraum

#define OPTIMAL 1 //z.B. <15% Sauerstoff

#define TOLERIERT 2 //z.B. 15-17% Sauerstoff

#define BEDROHLICH 3 //z.B. >17% Sauerstoff

//Definition der Zustände der Tür

#define ERLAUBT 1

#define UNTERSAGT 0

//Definition der O2 Schwellwerte

#define O2\_OPTIMAL\_MAX 15.0

#define O2\_TOLERIERT\_MIN 15.1

#define O2\_TOLERIERT\_MAX 17.0

#define O2\_BEDROHLICH\_MIN 17.1

//Definition der Temperatur Schwellwerte

#define TEMP\_OPTIMAL\_MAX 15.0

#define TEMP\_TOLERIERT\_MIN 15.1

#define TEMP\_TOLERIERT\_MAX 28.0

#define TEMP\_BEDROHLICH\_MIN 28.1

//Definition des Betriebsmodus

typedef enum { //Verteilung von konstanten Ganzzahlen an lesbare Namen

MANUELL,

AUTOMATIK

} Betriebsmodus; //Name des Datentyps

extern volatile Betriebsmodus modus; //Variable "modus" kann außerhalb der main.h verwendet werden

void setup(); //Aufruf bei Programmstart und Initialisierung verschiedener Funktion

*uint32\_t* millis(); //Laufzeitsteuerung

void initExternalInterrupts(); //Initialisiert die Interrupts an den PD2 und PD3

void initTimerInterrupts(); //Initialisiert den 8-Bit Timer und setzt seinen Takt (Prescaler)

void setBetrieb(); //wechselt den aktuellen Betriebsmodus bei Betätigung des Tasters S1

void setLEDs(); //setzt die vorhandenen LEDs auf vordefinierte Fälle

void setFanSpeed(); //setzt die Lüftergeschwindigkeit auf den vorgegebenen Setpoint (Lüftereinstellung)

void updateMeasurements(); //liest alle Sensoren aus und setzt die globalen Variablen auf die aktuellen Werte

void updateState(); //setzt die Bereiche basierend auf den Sauerstoffwerten

void updateVentil(); //wechselt den Zustand des Stickstoffventils (Automatik mit Case, Manuell mit Taster S2)

void uartOutput(); //Aktualisiert die Ausgabe über UART (z.B. Putty)

void uartOutputTimed(); //sorgt für einen Anzeigen wechsel

static *uint8\_t* adressDAC1 = 103; //Ansteuerung des Lüfters

float getO2(*uint8\_t*); //liest analogen Sensorwert aus und wandelt das ADC-Ergebnis in Sauerstoffgehalt in % um

float getTemperature(*uint8\_t*); //liest analogen Sensorwert aus und wandelt das ADC-Ergebnis in Temperatur in Grad C um

*uint16\_t* getFanSpeed(*uint8\_t*); //liest analogen Geschwindigkeitswert aus und wandelt das ADC-Ergebnis in U/min um

ISR(TIMER0\_OVF\_vect); //Laufzeitsteuerung

ISR(INT0\_vect); //wird ausgelöst wenn Taster S3 gedrückt wird

ISR(INT1\_vect); //wird ausgelöst wenn Taster S4 gedrückt wird

# Kurzbedienungsanleitung

## Allgemeine Informationen

Die Inbetriebnahme der EMU-Serverraumsteuerung, darunter der Baugruppenrahmen mit dem Netzteileinschub IK-88/1, die Baugruppe „EMU“, die Baugruppe „ATmega32-Board“ und die Zusatzplatine, darf ausschließlich von qualifiziertem und eingewiesenem Fachpersonal durchgeführt werden.

In der vorliegenden Kurzbedienungsanleitung sind alle nötigen Schritte erklärt, die zum ordnungsgemäßen Betrieb des Gesamtsystems eingehalten werden müssen.

## Enthaltene Komponenten

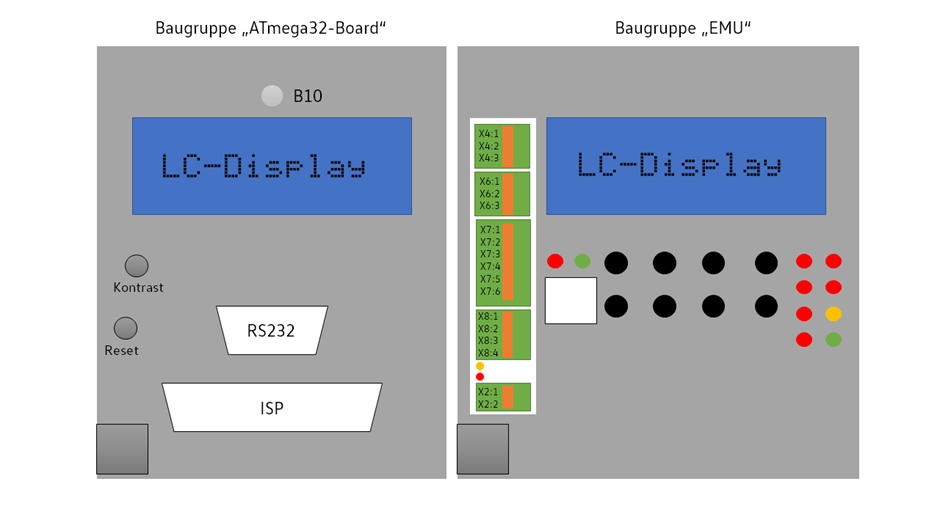
Die folgende Tabelle beinhaltet alle erforderlichen Komponenten für einen ordnungs- und sachgemäßen Gebrauch.

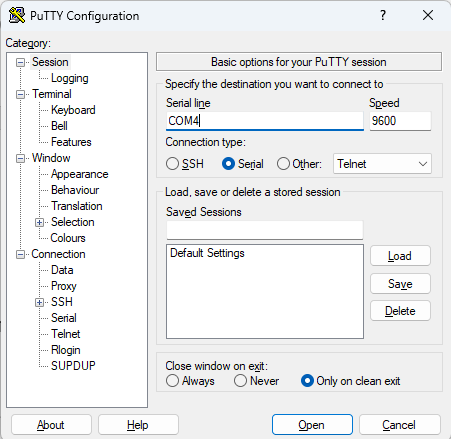
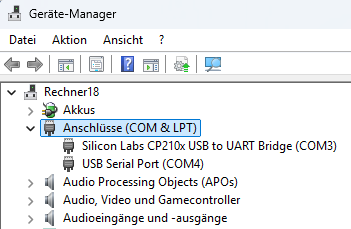
|  |  |
| --- | --- |
| Gerät | Typ |
| 19“ Baugruppenrahmen mit Netzteileinschub | IK-88/1 |
| Baugruppe „EMU“ inklusive Temperatursensor | ---- |
| Baugruppe ”ATmega32-Board” | MPL MEGA 32 |
| Zusatzplatine | ---- |
| Lüfter | ARCTIC F12 |
| Zusatzschaltung zur Simulation des Sauerstoffgehalts | ---- |
| Firmware ”EMU\_Serverraum\_Steuerung\_V2” | ---- |
| Serielle Leitung | 9-Polig Sub-D auf USB Kabel (RS232) |
| Netzanschluss | Kaltgeräteleitung |
| Desktop-PC mit Terminalprogramm | ”PuTTY” |

Tabelle : Übersicht benötigter Komponenten

## Herstellung der Betriebsbereitschaft

* **Platzieren des Baugruppenrahmen mit enthaltenen Baugruppen an gewünschter Ablage**
  + Baugruppe „EMU“
  + Baugruppe „ATmega32-Board“ mit Firmware „EMU\_Serverraum\_Steuerung\_V2“
  + Zusatzplatine
* **Verbinden der Sensoren & Aktoren (siehe 3.1, Übersicht Schnittstellen)**
  + Anschließen des Temperatursensors an die Klemme –X2 der Baugruppe „EMU“ (Polung ist nicht relevant)
  + Anschließen des Lüfters an die Klemme –X8 der Baugruppe „EMU“
    - X8:1 = PWM
    - X8:2 = Tacholeitung
    - X8:3 = Vcc (+12V)
    - X8:4 = GND
  + Anschließen der Zusatzschaltung zur Sauerstoffgehaltssimulation an die Klemme –X6 der Baugruppe „EMU“
    - X6:1 = Vcc (+12V)
    - X6:2 = Spannungssignal
    - X6:3 = GND
* **Verbindung zum Desktop-PC**
  + Anschließen der Baugruppe „ATmega32-Board“ mit dem Desktop-PC über X5 die serielle RS232-Schnittstelle (9-Polig Sub-D zu USB)



* **Überprüfen der Steckverbindung der Baugruppen im 19“-Baugruppenrahmen**
* **Starten des Terminalprogramms „PuTTY“**
  + Den passenden COM-Port der seriellen Leitung über „Geräte-Manager“ ausfindig machen
  + Die Schnittstelle ist wie folgt zu konfigurieren:
    - seriell, Telnet, 9600 Baud
* **Einschalten des IK-88/1-Netzteils im 19“ Baugruppenrahmen**
  + Kippschalter am Netzteil auf „EIN“ schalten

## Anlagenbetrieb

### Startsequenz

Die LED –B10 der Baugruppe „ATmega32-Board“ signalisiert durch dauerhaftes Leuchten den laufenden Betrieb. Nach dem Start des Systems zeigt das LC-Display des ATmega32-Board für eine Dauer von fünf Sekunden die aktuelle Firmware Version an.

EMU\_Serverraum

Steuerung\_V2

Abbildung : Ausgabe der Startsequenz auf dem LC-Display

### Regulärer Betrieb der EMU\_Serverraum\_Steuerung

Vom Mikrocontroller werden Werte für die Temperatur, den Sauerstoffgehalt und die Lüftergeschwindigkeit eingelesen. Währenddessen ist das LC-Display nicht aktiv. Die Ausgabe der aktuellen Messwerte und Zustände werden nur mittels UART an den Monitor des Administrators gesendet.

Zusatzplatine:

Über die Taster S1 und S2 können sowohl Betriebsmodus als auch Ventilstatus geändert werden. Die LED P1 wird über das ATmega32-Board in Abhängigkeit des Ventilstatus angesteuert. Über den Taster S3 kann die Serverraumtür geöffnet und geschlossen werden. Die LEDs P2 und P4 zeigen den aktuellen Türzugangsstatus an also erlaubt/untersagt.

### Zusätzliche Funktionen

Mit dem „Reset“-Taster auf der Frontplatte der Baugruppe „ATmega32-Board“ kann die Baugruppe zurückgesetzt werden.

Zur besseren Lesbarkeit des LC-Displays kann am Kontrastregler auf der Frontplatte der Baugruppe „ATmega32-Board“ der Kontrast des Displays eingestellt werden.

**Alle Taster, LEDs, das Display und die USB-Buchse X3 auf der EMU-Baugruppe sind permanent außer Betrieb gesetzt!**

# Inbetriebnahmeprotokoll

## Allgemeine Daten

*5589*

*EMU-Serverraumsteuerung*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Projektbezeichnung/Gerät Projektnummer

Auftraggeber: **IHK Braunschweig**

Brabandtstraße 11

38100 Braunschweig

Prüfer: **Dominic Taube**

Volkswagen AG

SE-6/6EI2

Industriestraße Nord

38239 Salzgitter

*28.05.2025*

Datum: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Durchzuführende Prüfungen

|  |  |  |
| --- | --- | --- |
| Vorgang |  |  |
| Prüfung des IK-88/1 Baugruppenrahmens nach DIN VDE 0702 und DGUV Vorschrift 3 |  |  |
| Prüfung der Spannungsversorgung des IK-88/1 Baugruppenrahmens |  |  |
| Sichtprüfung der Baugruppen |  |  |
| Prüfung des geschlossenen Systems „EMU-Serverraumsteuerung“ nach DIN VDE 0701 und DGUV Vorschrift 3 |  |  |
| Funktionsprüfung – Hardware: Baugruppe „EMU“ |  |  |
| Funktionsprüfung – Hardware: Baugruppe „ATmega32-Board“ |  |  |
| Funktionsprüfung – Software: Firmware „EMU\_Serverraum\_Steuerung“ |  |  |

Tabelle : Übersicht der durchzuführenden Prüfungen

## Vorbereitung

|  |  |
| --- | --- |
| Gerät | Typ |
| 19“ Baugruppenrahmen | K-IF/1 |
| Netzteileinschub | IK-88/1 |
| Baugruppe „EMU” | --- |
| Lüfter zur Simulation | ARCTIC F12 |
| Temperatursensor und Abgleichwiderstände für 0 °C und 100 °C | --- |
| Baugruppe ”ATmega32-Board” | MPL MEGA 32 |
| Firmware ”EMU\_Serverraum\_Steuerung” | --- |
| Zusatzschaltung zur Simulation des Sauerstoffgehalts | --- |
| Zusatzplatine zur Umschaltung des Betriebs- und Ventilstatus | --- |
| Busverlängerung | --- |
| Serielle Leitung | 9-Polig Sub-D auf USB Kabel (RS232) |
| Netzanschluss | Kaltgeräteleitung |
| Desktop-PC mit Terminalprogramm | ”PuTTY” |
| Multimeter zur Temperaturmessung | ELV 41640 |
| Drehzahlmessgerät | Voltcraft DT-30LK |
| tragbarer Gerätetester | Fluke 6500-2 |

Folgende Geräte/Software wird benötigt, um eine korrekte Prüfung des Systems und die Inbetriebnahme zu gewährleisten:

Tabelle : Benötigte Geräte und Software für die Inbetriebnahme

## Konfiguration der Baugruppen „EMU“ und „ATmega32-Board“

|  |  |  |
| --- | --- | --- |
| Vorgang |  |  |
| Alle Jumper den Anforderungen entsprechend richtig gesteckt (siehe 6.4.1 Jumperbelegungen) |  |  |
| RS232 Schnittstelle der Baugruppe „ATmega32-Board“ ist mittels serieller Leitung am PC verbunden |  |  |
| Das Terminalprogramm „PuTTY“ ist mit folgenden Einstellungen konfiguriert: seriell, COM-Port der seriellen Leitung, 9600 Baudrate |  |  |

Tabelle : Konfiguration der Baugruppen "EMU" und "ATmega32-Board"

### Jumperbelegungen

|  |  |  |
| --- | --- | --- |
| Baugruppe | Jumper | Belegung |
| ATmega32-Board | -J1 | 2-3 |
| ATmega32-Board | -J2 | 1-2 |
| ATmega32-Board | -J3 bis J34 | 1-2 |
| ATmega32-Board | -J35 | 1-2 |
| ATmega32-Board | -J36 | 1-2 |
| EMU | -XJ1 | 1-2 |
| EMU | -XJ5 | 2-3 |
| EMU | -XJ7 | 2-3 |
| EMU | -XJ8 | 2-3 |
| EMU | -XJ9 | 3-2 |
| EMU | -XJ10 | 2-3 |
| EMU | -XJ11 | 2-3 |
| EMU | -XJ12 | 2-3 |

Tabelle : Jumperbelegungen

## Prüfung des Baugruppenrahmen + IK-88/1 nach DIN VDE 0702 und DGUV Vorschrift 3

### Sichtprüfung des Baugruppenrahmen + IK-88/1

|  |  |  |
| --- | --- | --- |
| Vorgang |  |  |
| Typenschild/Kennzeichnungen |  |  |
| Gehäuse/Schutzabdeckung |  |  |
| Anschlussleitung/-stecker |  |  |
| Biegeschutz/Zugentlastung der Anschlussleitung |  |  |
| Befestigungen, Sicherungshalter |  |  |
| Kühlluftöffnungen |  |  |
| Schalter, Steuer-, Einstellvorrichtungen |  |  |
| Bemessung der zugänglichen Gerätesicherung |  |  |
| Baugruppen |  |  |
| Anzeichen von Überlastung/unsachgemäßem Gebrauch |  |  |
| Sicherheitsbeeinträchtigende Verschmutzung/Korrosion/Alterung |  |  |
| Mechanische Gefährdung |  |  |
| Unzulässige Eingriffe und Änderung |  |  |

Tabelle : Sichtprüfung des Baugruppenrahmen + IK-88/1

### Messungen des Baugruppenrahmen + IK-88/1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vorgang | Grenzwert | Messwert |  |  |
| Schutzleiterwiderstand | < 0,3 Ω | 0,07 Ω |  |  |
| Isolationswiderstand | > 1MΩ | >299 MΩ |  |  |
| Schutzleiterstrom | < 3,5 mA | 0,03 mA |  |  |
| Berührungsstrom | < 0,5 mA | 0,01 mA |  |  |

Tabelle : Messungen des Baugruppenrahmen + IK-88/1

## Prüfung der Spannungsversorgung

|  |  |  |  |
| --- | --- | --- | --- |
| Vorgang | Messwert |  |  |
| +5 V Versorgungsspannung ±5 % (-X1:1a/c) | 5,005 V |  |  |
| +12 V Versorgungsspannung ±5 % (-X1:31a) | 12,03 V |  |  |
| -12 V Versorgungsspannung ±5 % (-X1:31c) | -12,04 V |  |  |

Tabelle : Überprüfung der Spannungsversorgung

## Sichtprüfung der Baugruppen

### Baugruppe „EMU“

|  |  |  |
| --- | --- | --- |
| Vorgang |  |  |
| Platine und Bauteile unbeschädigt |  |  |
| Lötstellen in Ordnung |  |  |
| Bauteile richtig eingebaut (nicht verpolt) |  |  |
| Frontplatte korrekt montiert |  |  |
| Frontplatte beschriftet |  |  |
| Jumper den Anforderungen entsprechend gesteckt |  |  |

Tabelle : Sichtprüfung der Baugruppe "EMU"

### Baugruppe „ATmega32-Board“

|  |  |  |
| --- | --- | --- |
| Vorgang |  |  |
| Platine und Bauteile unbeschädigt |  |  |
| Lötstellen in Ordnung |  |  |
| Bauteile richtig eingebaut (nicht verpolt) |  |  |
| Frontplatte korrekt montiert |  |  |
| Frontplatte beschriftet |  |  |
| Jumper den Anforderungen entsprechend gesteckt |  |  |

Tabelle : Sichtprüfung der Baugruppe "ATmega32-Board"

### Zusatzplatine zur Umschaltung des Betriebs- und Ventilstatus

|  |  |  |
| --- | --- | --- |
| Vorgang |  |  |
| Platine und Bauteile unbeschädigt |  |  |
| Lötstellen in Ordnung |  |  |
| Bauteile richtig eingebaut (nicht verpolt) |  |  |
| Frontplatte korrekt montiert |  |  |
| Frontplatte beschriftet |  |  |
| Platine beschriftet |  |  |

Tabelle : Sichtprüfung der Zusatzplatine

## Prüfung des geschlossenen Systems „EMU-Serverraumsteuerung“ nach DIN VDE 0701 und DGUV Vorschrift 3

**Hinweis: Ab diesem Punkt werden die Baugruppen „EMU“, „ATmega32-Board“ und die Zusatzplatine in den Baugruppenrahmen eingeschraubt und als geschlossenes System verschlossen. Freie Steckplätze werden mit Blindabdeckungen abgedeckt!**

### Sichtprüfung des geschlossenen Systems

|  |  |  |
| --- | --- | --- |
| Vorgang |  |  |
| Typenschild/Kennzeichnungen |  |  |
| Gehäuse/Schutzabdeckung |  |  |
| Anschlussleitung/-stecker |  |  |
| Biegeschutz/Zugentlastung der Anschlussleitung |  |  |
| Befestigungen, Sicherungshalter |  |  |
| Kühlluftöffnungen |  |  |
| Schalter, Steuer-, Einstellvorrichtungen |  |  |
| Bemessung der zugänglichen Gerätesicherung |  |  |
| Baugruppen |  |  |
| Anzeichen von Überlastung/unsachgemäßem Gebrauch |  |  |
| Sicherheitsbeeinträchtigende Verschmutzung/Korrosion/Alterung |  |  |
| Mechanische Gefährdung |  |  |
| Unzulässige Eingriffe und Änderung |  |  |

Tabelle : Sichtprüfung des geschlossenen Systems

### Messungen des geschlossenen Systems

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vorgang | Grenzwert | Messwert |  |  |
| Schutzleiterwiderstand | < 0,3 Ω | 0,07 Ω |  |  |
| Isolationswiderstand | > 1MΩ | >299 MΩ |  |  |
| Schutzleiterstrom | < 3,5 mA | 0,04 mA |  |  |
| Berührungsstrom | < 0,5 mA | 0,01 mA |  |  |

Tabelle : Messungen des geschlossenen Systems

## Funktionsprüfung

### Funktionsprüfung – Hardware: Baugruppe „EMU“

|  |  |  |
| --- | --- | --- |
| Vorgang |  |  |
| Baugruppe „EMU“ funktioniert erwartungsgemäß |  |  |

Tabelle : Funktionsprüfung Hardware der Baugruppe "EMU"

**Hinweis: Die Baugruppe „EMU“ wurde vor der Nutzung in der EMU-Serverraumsteuerung bereits auf Funktion überprüft. Für die Serverraumsteuerung wurde die Flachbandleitung der Baugruppe „EMU“ entfernt, da die Frontplatine nicht benötigt wird.**

### Funktionsprüfung – Hardware: Baugruppe „ATmega32-Board“

|  |  |  |
| --- | --- | --- |
| Vorgang |  |  |
| Bicolor LED –B10 auf Frontplatte leuchtet grün |  |  |
| Beim Hochladen eines beliebigen Programmes über die ISP Schnittstelle leuchtet die LED –B10 rot |  |  |
| LC-Display ist erkennbar eingeschaltet |  |  |
| Kontrast des LC-Display lässt sich über den Kontrastregler regeln |  |  |
| Über den „Reset“-Taster lässt sich die ganze Baugruppe zurücksetzen |  |  |

Tabelle : Funktionsprüfung Hardware der Baugruppe "ATmega32-Board"

### Funktionsprüfung – Software: Baugruppe „EMU“, „ATmega32-Board“ und Zusatzplatine

|  |  |  |  |
| --- | --- | --- | --- |
| Baugruppe/PC | Vorgang |  |  |
| ATmega32-Board | Bei Startvorgang wird die aktuelle Version der Firmware „EMU\_Serverraum\_Steuerung“ auf dem LC-Display angezeigt |  |  |
| Desktop-PC | Nach dem Startvorgang beginnt die zyklische Übertragung mittels Terminalprogramm PuTTY |  |  |
| Betriebsmodus | Der Betriebsmodus lässt sich mit Taster S1 zwischen MANUELL und AUTOMATIK umschalten |  |  |
| Stickstoffventil | Das Stickstoffventil (LED P1, rot) lässt sich im Betriebsmodus MANUELL mit Taster S2 öffnen und schließen |  |  |
| Sauerstoffgehalt | Bei O2-Gehalt > 17 % geht das Stickstoffventil im Modus AUTOMATIK auf (LED P1 leuchtet) |  |  |
| Lüftergeschwindigkeit | Der Lüfter dreht mit einer Geschwindigkeit von 700 U/min |  |  |
| Temperaturüberwachung | Der Temperatursensor liefert die aktuelle Raumtemperatur |  |  |
| Desktop-PC | Über PuTTY werden die richtigen Messwerte und Zustände angezeigt |  |  |

Tabelle : Funktionsprüfung Software der Baugruppen "EMU", "ATmega32-Board" und Zusatzplatine

### Auswertung

|  |  |
| --- | --- |
| Lfd. Nr. | Mängel |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tabelle : Auswertung

### Bestätigung der Prüfung

Anmerkung des Prüfers:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



*Salzgitter, 28.05.2025*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, Datum Unterschrift Prüfer

# Übergabe- /Einweisungsprotokoll

## Allgemeine Daten

Auftraggeber: **IHK Braunschweig**

Brabandtstraße 11

38100 Braunschweig

Auftragnehmer: **Dominic Taube**

Volkswagen AG

SE-6/6EI2

Industriestraße Nord

38239 Salzgitter

Datum: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Unterwiesene \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Person: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Weitere \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Teilnehmer: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Gegenstand der Übergabe

### Hardware (EMU-Serverraumsteuerung)

* 19“ Baugruppenrahmen mit Netzteileinschub IK-88/1
* Baugruppe „EMU“
* Baugruppe „ATmega32-Board“
* Zusatzplatine

### Software

* Firmware „EMU\_Serverraum\_Steuerung\_V2“

### Dokumente

* Kurzbedienungsanleitung
* Inbetriebnahmeprotokoll
* Übergabe-/Einweisungsprotokoll

**Hinweis**: **Die Simulationsschaltung dient nur zur Demonstration und sollte nach der Übergabe durch einen passenden Sensor ausgetauscht werden!**

## Übergabe/Einweisung

In diesem Abschnitt wird die Funktion des zu übergebenden Systems mit dem Kunden schrittweise überprüft und bestätigt.

### Überprüfung der Vollständigkeit

|  |  |  |
| --- | --- | --- |
| Gegenstand | | Vollständigkeit |
| Hardware | 19“ Baugruppenrahmen mit Busplatine & Netzteil IK-88/1  Baugruppe „EMU“  Baugruppe „ATmega32-Board“  Zusatzplatine |  |
| Firmware | „EMU\_Serverraum\_Steuerung\_V2“ |  |
| Dokumente | Kurzbedienungsanleitung  Übergabe-/Einweisungsprotokoll  Inbetriebnahmeprotokoll |  |

Tabelle 22: Überprüfung auf Vollständigkeit bei der Übergabe

### Funktionsprüfung mit dem Auftraggeber

|  |  |  |  |
| --- | --- | --- | --- |
| Baugruppe/PC | Vorgang |  |  |
| ATmega32-Board | Bei Startvorgang wird die aktuelle Version der Firmware „EMU\_Serverraum\_Steuerung\_V2“ für 5 Sekunden auf dem LC-Display angezeigt |  |  |
| Desktop-PC | Nach dem Startvorgang beginnt die zyklische Übertragung alle 2 Sekunden mittels Terminalprogramm PuTTY |  |  |
| Betriebsmodus | Der Betriebsmodus lässt sich mit Taster S1 zwischen MANUELL und AUTOMATIK umschalten |  |  |
| Stickstoffventil | Das Stickstoffventil (LED P1, rot) lässt sich im Betriebsmodus MANUELL mit Taster S2 öffnen und schließen |  |  |
| Serverraumtür | Die Serverraumtür lässt sich mit dem Taster S3 öffnen und schließen |  |  |
| Türzugang | Der Türzugang wird in Abhängigkeit von dem Sauerstoffgehalt und Stickstoff- ventil, über die LEDs P2 und P4 richtig visualisiert |  |  |
| Alarm | Der Alarm geht an sobald Die Tür offen und der Türzugang untersagt ist. Dabei wird das Stickstoffventil geschlossen. |  |  |
| Sauerstoffgehalt | Bei O2-Gehalt > 17 % geht das Stickstoffventil im Modus AUTOMATIK auf (LED P1 leuchtet) |  |  |
| Lüftergeschwindigkeit | Der Lüfter ändert seine Geschwindigkeit bei sich änderndem Temperaturbereich |  |  |
| Temperaturüberwachung | Der Temperatursensor liefert die aktuelle Raumtemperatur in Bereichen |  |  |
| Desktop-PC | Über PuTTY werden die richtigen Messwerte und Zustände angezeigt |  |  |

Tabelle 23: Funktionsprüfung bei Übergabe

## Mängelliste

|  |  |
| --- | --- |
| Lfd. Nr. | Mängel |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Tabelle 24: Mängelliste bei Übergabe

**Nachbesserungstermin:**

## Bestätigung über Erhalt

Abnahme

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, Datum Unterschrift Auftraggeber

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, Datum Unterschrift Auftragnehmer

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, Datum Unterschrift einweisende Person

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ort, Datum Unterschrift eingewiesene Person