

**Verteilte Systeme (304271)**

Hochschule Heilbronn – Automotive Systems Engineering

Wintersemester 2022/23

Prof. Dr. -Ing. Ansgar Meroth, M.Sc. Petre Sora

**Projektaufgabe**

**Thermostat**

Abgabetermin: 17 Januar 2023

|  |  |
| --- | --- |
| Hoehnel, Moritz (210258)  Automotive Systems Engineering  mhoehnel@stud.hs-heilbronn.de | Ritter, Mattis (210265)  Automotive Systems Engineering  mritter@stud.hs-heilbronn.de |

Inhaltsverzeichnis

[1. Abstract 3](#_Toc123817847)

[2. Projektmanagement 4](#_Toc123817848)

[3. Lastenheft 5](#_Toc123817849)

[3.1 Einleitung 5](#_Toc123817850)

[a. Zweck 5](#_Toc123817851)

[b. Umfang 5](#_Toc123817852)

[c. Erläuterungen zu Begriffen und / oder Abkürzungen 5](#_Toc123817853)

[d. Verweise auf sonstige Ressourcen oder Quellen 5](#_Toc123817854)

[e. Übersicht 5](#_Toc123817855)

[3.2 Allgemeine Beschreibung 5](#_Toc123817856)

[a. Produktperspektive 5](#_Toc123817857)

[b. Produktfunktionen 5](#_Toc123817858)

[c. Benutzermerkmale 6](#_Toc123817859)

[d. Einschränkungen 6](#_Toc123817860)

[e. Annahmen und Abhängigkeiten 6](#_Toc123817861)

[f. Aufteilung der Anforderungen 6](#_Toc123817862)

[3.3 Spezifische Anforderungen 7](#_Toc123817863)

[a. funktionale Anforderungen 7](#_Toc123817864)

[b. nicht-funktionale Anforderungen 11](#_Toc123817865)

[c. Qualitätsanforderungen 11](#_Toc123817866)

[3.4 Verifikation 11](#_Toc123817867)

[4. Hardware 12](#_Toc123817868)

[5. Softwarearchitektur 14](#_Toc123817869)

[5.2 Main 14](#_Toc123817870)

[5.3 TMP75 14](#_Toc123817871)

[5.4 WS2812 14](#_Toc123817872)

[5.5 Servo-Ansteuerung 15](#_Toc123817873)

[6. Code Dokumentation 16](#_Toc123817874)

[Quellenverzeichnis 17](#_Toc123817875)

[Abbildungsverzeichnis 17](#_Toc123817876)

# Abstract

Das Thema Smart Home wird immer wichtiger. Dieses Projekt setzt einen Teil eines Smart Homes um: Das Thermostat. Das Thermostat besitzt außer der Temperaturegelfunktion die Fähigkeit der Kommunikation. So ist es Smart-Home-fähig. Dieser Bericht beschreibt die Planung und Lösung dieses Thermostates.

Zu Beginn des Projektes wurde ein Zeitplan bestimmt, gefolgt von dem Festsetzen des Lastenheftes. Nach beenden der Planungsphase, wurde die benötigte Hardware entworfen und hergestellt. Danach wurde der Quellcode erstellt. Der Lösungsansatz wird in dem Kapitel Softwarearchitektur beschrieben, die detaillierte Lösung ist in dem Kapitel Code Dokumentation zu finden.

# Projektmanagement

Zum Start des Projektes wurde ein Zeitplan erstellt. Das Team entschied sich einen Phasenplan zu nutzen. Diese Art der Planung ermöglicht eine zeitliche Übersicht des Projekts und berücksichtig Abgabe- und Präsentationsdaten. Für den Phasenplan wurden alle Arbeitsschritte aufgelistet. Das Ende eines Arbeitsschrittes wird durch einen Meilenstein () gekennzeichnet.



Abb. 1.1: Phasenplan

# Lastenheft

## Einleitung

### Zweck

Dieses Dokument beschreibt die Anforderungen an das Projekt Thermostat für den Kurs Verteilte Systeme.

### Umfang

Im Rahmen der Kurses sollen Funktionen eines Smart Homes realisiert werden. Dieses Projekt soll ein Thermostat mit Mehrstufiger-Hysterese-Regelung umsetzen.

Es sollen dabei Hardware als auch Software entwickelt werden. Es soll eine zusätzliche Platine für das Microcontroller-Board, welches im Kurs Microcontroller (304132) bereitgestellt wurde, entworfen werden. Die Platine soll die fehlenden Bauteile des Microcontroller-Board beherbergen, welche für die Realisierung des Thermostats notwendig sind.

Es soll eine Software entwickelt werden, welche auf dem Microcontroller genutzt werden kann und die Funktionalität des Produktes herstellt.

### Erläuterungen zu Begriffen und / oder Abkürzungen

|  |  |
| --- | --- |
| Servo  µC. | Servomotor  Microcontroller |

### Verweise auf sonstige Ressourcen oder Quellen

Bauteile, Schaltplanbibliothek als auch Code-Elemente werden durch die Dozenten des Kurses Verteilte Systeme bereitgestellt.

### Übersicht

Nach der Einleitung folgt eine allgemeine Beschreibung des Systems, gefolgt von spezifischen Anforderungen und der Verifikation.

## Allgemeine Beschreibung

### Produktperspektive

Das Produkt soll als Teil eines Smart Homes eine Heizung steuern. Der Nutzer soll die Solltemperatur an dem Microcontroller-Board und an dem Zentralen Display einstellen können. Durch die LED bekommt der Nutzer Feedback, wie stark geheizt wird.

### Produktfunktionen

Wie oben erwähnt soll eine Platine und Software entwickelt werden.

Platine: Die Kommunikation zwischen den Bauteilen der Platine und dem Microcontroller-Board soll über Serielle-Schnittstellen realisiert werden. Die Art der Schnittstellen sind durch die jeweiligen Bauteile vorgegeben. Auch die Bauteile sind vorgeben. Generell gibt es ein Bauteil zum Messen der Ist-Temperatur, ein Stellglied für die Heizung und eine LED zur Visualisierung. Darüber hinaus soll ein CAN-Controller verwendet werden. Die vergebenen Bauteile sind in d.Einschränkungen zu finden. Alle weiteren Bauteile (Buchsen, Quarze, Kondensatoren und Widerstände) sollen passend zu den vorher genannten Bauteilen gewählt werden.

Software: Diese soll die Daten des Temperatur-Sensors auswerten. Es soll eine Regler Logik (Fünfpunkt Hysterese Regelung) erstellt werden, welche Ausgangssignale zur Steuerung des Servos und der LED erstellt.

### Benutzermerkmale

Da das Produkt Teil eines Smart-Homes muss jeder Bewohner eines Hauses als Benutzer gewertet werden. Somit sind Nutzer jeden Alters, Bildung, Sprache und Sachkenntnis potenzielle Bediener des Produkts.

### Einschränkungen

Es sollen folgende Bauteile verwendet werden:

- Servo vom Typ FXX-3037-TOP

- Temperatursensor vom Typ TMP75B

- Vollfarb-LED vom Typ WS2812

- CAN-Controller vom Typ MCP2515

### Annahmen und Abhängigkeiten

Der Schaltplan und das Board werden mit dem Programm EAGLE erstellt. Die Platine soll vom Auftraggeber gefertigt werden.

Die Software wird mit dem Programm Microchipstudio entwickelt. Die Programmiersprache ist C.

### Aufteilung der Anforderungen

Eine Vernetzung mit weiteren Funktionen des Smart-Homes ist nicht teil des Projekts, kann aber später realisieret werden.

## Spezifische Anforderungen

### funktionale Anforderungen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Name | Detaillierte Beschreibung | Werte-  bereich | Einheit | Priorität | Status | Verifikations-  kriterien |
| A.1 | Stellen | Der Bediener soll dabei die Möglichkeit haben in Ganzzahl-Schritten seine Wunschtemperatur zu stellen. | 5-40 | °C | high | offen | Erfolgreiche Vorgabe durch Versuch |
| A.2 | Stellen | Der Bediener soll die Temperatur am µC und am Zentralen Display stellen;  Die Stellung am µC soll nur als Back-up verwendet werden |  |  | low | offen | Erfolgreiche Änderung der Solltemperatur |
| A.3 | Messen | Der Temperatursensor soll die Ist-Temperatur aufzeichnen | min. 5-40 | °C | high | offen | Prüfen wie in 4. beschrieben |
| A.4 | Messen | Temperatursensor Toleranz | +/-0,1 | °C | low | offen | Hersteller-vorgabe |
| A.5 | Stufen | Soll die Heizung stellen | 0-5 |  | high | offen | Prüfen wie in 4. beschrieben |
| A.5.1 | Stufe 0 | Falls | 0 |  |  |  |
| A.5.2 | Stufe 1 | Falls | 1 |  |  |  |
| A.5.3 | Stufe 2 | Falls 2 | 2 |  |  |  |
| A.5.4 | Stufe 3 | Falls 4 | 3 |  |  |  |
| A.5.5 | Stufe 4 | Falls 6 | 4 |  |  |  |
| A.5.6 | Stufe 5 | Falls | 5 |  |  |  |
| A.6 | Hysterese | An den Schaltpunkten soll eine Hysterese vorhanden sein | +/-0,5 | °C | high | offen | Stellt wie in 4. beschrieben |
| B.1 | Servo, Stufe 0 | Die Heizung bleibt aus | 0 | % | high | offen | Ist-Temperatur soll auf einen Wert gestellt werden, dass die jeweiligen Grenzen der Temperatur-  bereiche erreicht sind. Der Servo soll die geforderte Stellung erreichen. |
| B.2 | Servo, Stufe 1 | Die Heizung wird zu 20% geöffnet | 20 | % | low | offen |
| B.3 | Servo, Stufe 2 | Die Heizung wird zu 40% geöffnet | 40 | % | low | offen |
| B.4 | Servo, Stufe 3 | Die Heizung wird zu 60% geöffnet | 60 | % | low | offen |
| B.5 | Servo, Stufe 4 | Die Heizung wird zu 80% geöffnet | 80 | % | low | offen |
| B.6 | Servo, Stufe 5 | Die Heizung wird zu 100% geöffnet | 100 | % | low | offen |
| C.1 | LED, Stufe 0 | LED leuchtet in der Farbe blau | Blau |  | low | offen | Testdurchführung wie bei Nr. B.X,  jeweilige Farben werden wiedergegeben |
| C.2 | LED, Stufe 1 | LED leuchtet in der Farbe grün | Grün |  | low | offen |
| C.3 | LED, Stufe 2 | LED leuchtet in der Farbe gelb | Gelb |  | low | offen |
| C.4 | LED, Stufe 3 | LED leuchtet in der Farbe orange | Organge |  | low | offen |
| C.5 | LED, Stufe 4 | LED leuchtet in der Farbe magenta | Magenta |  | low | offen |
| C.6 | LED, Stufe 5 | LED leuchtet in der Farbe rot | Rot |  |  |  |  |
| A.7 | Regelung | Software soll Signal an Servo geben wie geöffnet wird | 0-5 |  | high | offen | Stellt wie in 4. beschrieben |
| A.8 | Quellcode | Kommunikation via SPI Schnittstelle mit Servo |  |  | high | offen | Kommunikation möglich |
| A.9 | Quellcode | Kommunikation via I2C mit Temperatursensor |  |  | high | offen | Kommunikation möglich |
| A.10 | Quellcode | Kommunikation via Serieller Schnittstelle mit LED |  |  | low | offen | Kommunikation möglich |
| A.11 | Quellcode | Auswerten Messwerte, mit Regelung Erstellen eines Output Signals für Servo |  |  | high | offen | Servo führt Bewegungen aus |
| A.12 | Quellcode | Melden welche Heizstellung an LED |  |  | low | offen | Farbe passt zur Servostellung |

#### externe Schnittstellen

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Name | Detaillierte Beschreibung | Werte-  bereich | Einheit | Priorität | Status | Verifikations-  kriterien |
| D.1 | CAN-Out | 0x400;  Temperatur Vorkomma;  Signed Byte | 8 | bit | low | offen | Nachricht wird an Display empfangen |
| D.2 | CAN-Out | 0x400;  Temperatur Nachkomma;  Unsigned Byte | 8 | bit | low | offen |
| D.3 | CAN-Out | 0x400;  Status der Heizung entsprechend der in B.X definierten Stufen;  Unsigned Byte | 8 | bit | low | offen |
| D.4 | CAN-In | 0x401;  Solltemperatur;  Signed Byte | 8 | bit | high | offen | Nachricht vom Display wird empfangen |

#### Betriebszustände

Sobald das Gerät eingeschalter ist, soll es dauerhaft die Wunschtemperatur regeln. Die Abtastung der Temperatur wird maximal im 100ms Abstand gefordert.

#### Output

Der Output der Platine ist die Servo-Stellung und die LED.

#### Umgebungsbedingungen

Es wird festgelegt, dass das Thermostat Einsatztemperaturen von 5-40°C hat. Diese Voraussetzung wird durch die eingeschränkte Nutzung in einem Haus festgelegt. Hier werden keine größeren Temperaturschwankungen erwartet. Es soll vernachlässigt werden, dass Mess- und Stellelektronik nah beieinandersitzen und eventuelle Ist-Temperaturunterschiede zwischen Raummitte und Thermometer entstehen können.

### nicht-funktionale Anforderungen

#### Anforderungen an Performance

Die Platinentempertur ist nicht zu managen.

#### Usability

Der Nutzer soll durch Tastendrücke schnell die Temperatur ändern können. Die Stellgeschwindigkeit soll dabei nicht bedeutend langsamer sein als die Heizungsbedienung mit einer herkömmlichen Drehregler Bedienung (Gesamte Dauer zum Ändern der Solltemperatur <= 10s)

#### Wartbarkeit

Die Platine soll einfach auf das Microcontroller-Board gesteckte werden können.

#### Änderbarkeit/Skalierbarkeit

Eingriffe in die Software können auch noch nach Beendigung des Projekts durchgeführt werden können.

### Qualitätsanforderungen

Alle Lötstellen sollen sauber hergestellt werden. Es ist kein Gehäuse gefordert.

## Verifikation

Es soll die Richtigkeit der Temperaturmessung im Labor geprüft werden. Es soll der Messwert des Thermometers mit dem Wert eines herkömmlichen Thermometers zur Ermittlung der Raumtemperatur verglichen werden.

Darüber hinaus soll eine Sichtprüfung durchgeführt werden, welche die richtige Servo-Stellung verifiziert. Messzenario: Per Manipulation soll ein Temperaturdelta gestellt werden, dass alle 5 Stufen durch den Servo gestellt werden.

# Hardware

Im ersten Entwicklungsschritt wurde ein Schaltplan erstellt. Er zeigt alle Verwendeten Bauteile:

* Servo-Motor
* Temperatursensor
* Mehrfarb LED
* Wannenstecker
* CAN-Controller
* CAN-Transiver
* SUB-D Anschluss

Die ersten drei Bauteile sind Teil des eigentlichen Thermostates. Der Wannenstecker ist die Verbindung zum Microcontroller Board. Die letzten drei Bauteile sind für die Kommunikation mit den anderen Elementen des Smart Homes zuständig.

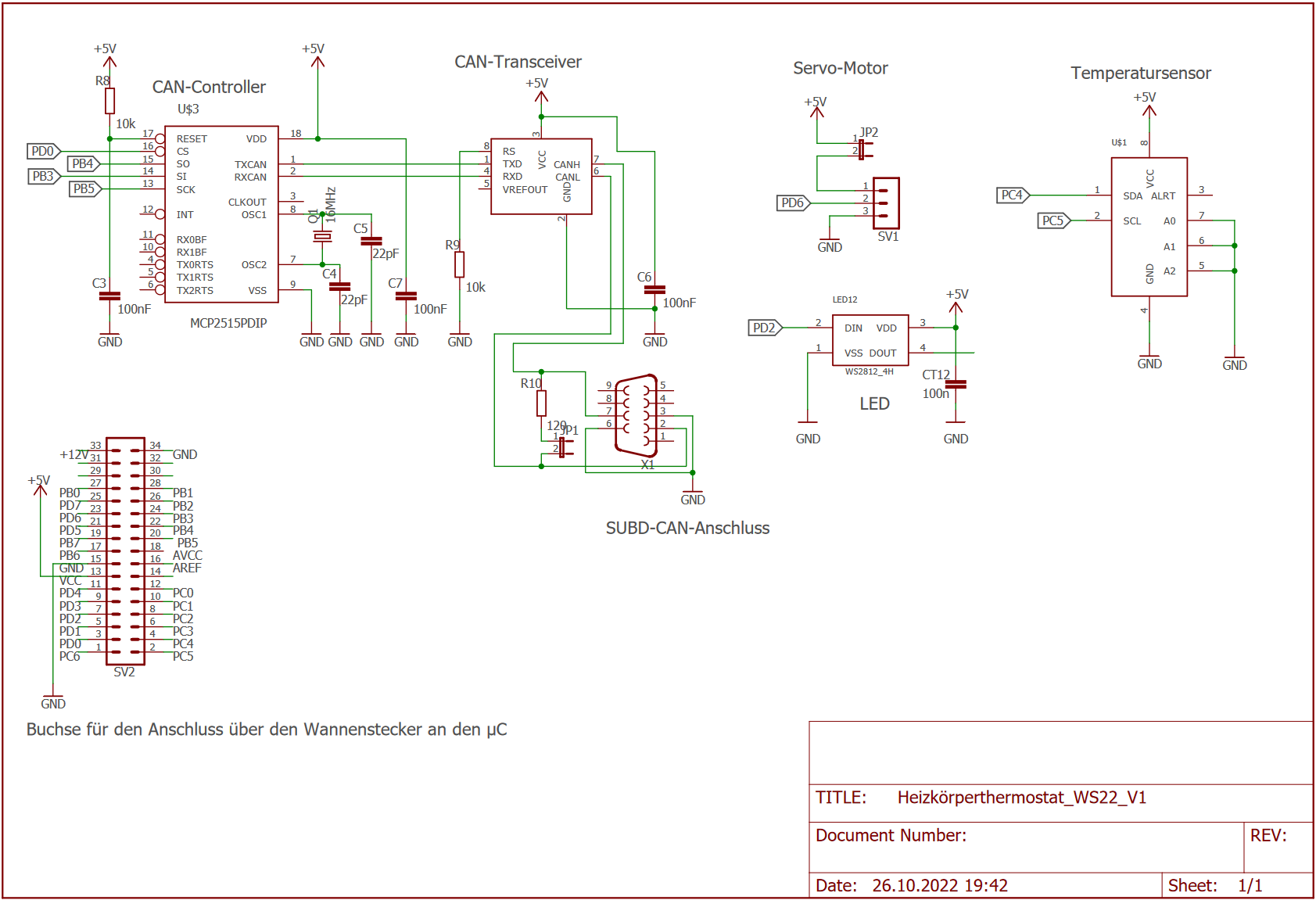


Abb. 4.1: Schaltplan

Beim Erstellen des Boards wurde darauf geachtet, dass Pufferkondensatoren möglichst nah bei den zugehörigen Bauteilen sitzen. Es sollten so wenig wie möglich Durchkontaktierungen gemacht werden.

Ein Bild, das Text, Elektronik enthält.

Automatisch generierte Beschreibung

Abb. 4.2: Board

# Softwarearchitektur

Der Quellcode wird für den ATmega 88PA in C geschrieben. Ziel der Software ist die Kommunikation mit den einzelnen Bauteilen herzustellen, eine Regler Logik zu implementieren und den Benutzer Eingaben tätigen zu lassen.

Die Software wird dazu in einzelnen Modulen entwickelt. Jedes Bauteil bekommt dabei sein eigenes Modul. Auch der Regler wird in einem eigenen Modul programmiert. Aus der main heraus werden so die Module gestartet. Darüber hinaus gibt es noch ein Init-Modul, welches beim Start des Microcontroller alle Module initialisiert.

Im Folgenden werden alle Module beschrieben, die selbst geschrieben worden sind und noch nicht aus dem Kurs Microcontroller bekannt sind.

## Main

Zu Beginn werden der MCP2515, die WS2812 und der TMP75 vorbereitet für den Betrieb. Dies wird durch Definitionen, Defines und Initialisierungen der ersten Messung gemacht.

Nun startet die Hauptschleife. Alle 100 Millisekunden wird darin eine Messung gestartet. Der Wert wird zum einen zur Anzeige an die Displays gesendet und zum anderen in das Regler-Modul, welches eine Regelstufe bestimmt. Diese bestimmt dann die Stellung des Servos und die Farbe der LED.

Für den Fall, dass noch kein Signal via CAN angekommen ist, kann man die Wunschtemperatur mit den Knöpfen auf dem Microcontroller-Board stellen. Diese werden alle 10 Millisekunden abgeprüft. Sobald ein CAN-Signal mit einer Wunschtemperatur angekommen ist, hat dieses Priorität und wird als neue Solltemperatur angenommen.

## TMP75

Der TMP75 wird zum Messen der Umgebungstemperatur verwendet. Der Microkontroller der Platine kommuniziert via TWI mit dem TMP75. Aus diesem Grund ist das TWI\_Atmega Modul von Herr Petre Sora in dem Projekt eingebunden. Ebenfalls von Herr Sora wird die TMP75\_Read\_Temperature Funktion verwendet. Dabei wird der Microkontroller als Master initialisiert. Der TMP75 ist Slave. Seine Adresse wurde durch das Verbinden aller Adresspins mit Ground auf 0b1001000 festgelegt. Der Master möchte, dass das Temperatur-Register des TMP75 gesendet wird und fragt deshalb das Register 0b00000000 an. Der Temperaturwert ist in zwei 8 bit Registern geschrieben. Es werden das High und Low Register empfangen und am Ende der Funktion zusammengesetzt. Der Temperaturwert ist als Modulvariable gespeichert. Diese Funktion muss jedes Mal in der main aufgerufen werden, wenn man eine Messung der Umgebungstemperatur möchte. Damit ein Überhitzen des TMP75 durch Überlastung verhindert wird, startet die main alle 100ms eine Messung.

Durch die zweite Funktion TMP75\_Get\_Temperature kann die Modulvariable mit der Temperatur in der Main abgefragt werden. Es wird dabei eine Nachkommastelle betrachtet. Der übergebene Wert ist mit dem Faktor 10 versehen, sodass einen floatingpoint Datentyp vermieden wird. Der Wert wird vor der Rückgabe, wie im Datenblatt des TMP75 beschrieben, bearbeitet.

## WS2812

Dieses Modul ist für die Ansteuerung der Mehrfarb-LED zuständig. Der Code wurde von Herr Petre Sora bereitgestellt und für die Anwendung angepasst. In der Init Funktion wird der PORT PD2 zum Ausgang der LED definiert. Es wird dabei auf die Definition für den WS2812\_pin (WS2812\_1) aus der main.h zugegriffen.

In der WS2812\_step Funktion wird aus einem 6 mal 3 array eine Zeile aufgerufen und als Input in die WS2812\_Set\_Colour gegeben. Dieses Array hat die Länge drei und gibt die Zusammensetzung der Farbe an. Es werden die Werte für grün, rot und blau übergeben. Die WS2812\_Set\_Colour nimmt noch einen weiteren Input-Wert. Dieser beschreibt die Position der LED und ist in unserem Fall immer 2. Zu Beginn der WS2812\_Set\_Colour Funktion wird der anzusteuernde Pin auf Null gesetzt, erst danach kann die neue Farbe gesetzt werden.

## Servo-Ansteuerung

Es wird ein PWM-Signal an PD6 bzw. OC0A benötigt.

Soll-Periodendauer: , Frequenz:

Eingestellt 🡪



Abb. 5.1: PWM Duty Cycle and Period

|  |  |  |
| --- | --- | --- |
| Stufe | Pulsweite | OCR0A |
| 0 | 0,5ms | 45 (44) |
| 1 | 0,9ms | 38 |
| 2 | 1,3ms | 30 |
| 3 | 1,7ms | 23 |
| 4 | 2,1ms | 16 |
| 5 | 2,5ms | 9 (10) |

Abb. 5. Duty Cyclone V.S. Angle

# Code Dokumentation

# Quellenverzeichnis

Meroth, Ansgar und Sora, Petre: Sensornetzwerke in Theorie und Praxis, Heilbronn und Wiesbaden, 2021 (2. Auflage)

Unbekannt: ATmega48PA/88PA/168PA [DATASHEET], https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-9223-Automotive-Microcontrollers-ATmega48PA-ATmega88PA-ATmega168PA\_Datasheet.pdf (Stand: 05.01.2023)

Unbekannt: Servo-Motor-Kit User Manual, https://eu.mouser.com/datasheet/2/598/ervo\_Motor\_Kit-1020966.pdf (Stand: 05.01.2023)

Unbekannt: TMP75B Datasheet, https://www.ti.com/lit/ds/symlink/tmp75b.pdf?ts=1672919165643&ref\_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FTMP75B (Stand: 05.01.2023)

Tessie: WS2812B Addressable RGB LED: Datasheet, Pinout and Applications, https://www.utmel.com/components/ws2812b-addressable-rgb-led-datasheet-pinout-and-applications?id=534 (Stand: 05.01.2023)

## Abbildungsverzeichnis

|  |  |
| --- | --- |
| Abbildung |  |
| 1.1 | Phasenplan, Erstellt mit MS Word |
| 4.1 | Schaltplan, Erstellt mit Eagle |
| 4.2 | Board, Erstellt mit Eagle |
| 5.1 | PWM Duty Cycle and Period: Servo-Motor-Kit User Manual |
| 5.2 | Duty Cyclone V.S. Angle: Servo-Motor-Kit User Manual |