CCA – COMPETENCE CENTRE

HTL Anichstraße

# Laborbericht 2

Zweipunktregelung und Wendetangentenverfahren

**Autoren:** Steiner & Ladinig
**Gruppe**: 1
**Lehrer**: Wischounig Philipp
**Durchführungsdatum**: 22.10.2025

# Inhaltsverzeichnis

# 1. Zweipunktregelung

In der Übung verwenden wir einen Arduino Nano, um eine Zweipunktregelung mithilfe einer Glühbirne zu messen. Wir verwenden einen DS18B20-Sensor zur Temperaturmessung und einen L298N Motor Controller, um die Glühbirne anzusteuern.

## 1.1. Programmcode

### 1.1.1. Erklärung

```
#include "HardwareSerial.h"
#include <Arduino.h>
#include <DallasTemperature.h>
#include <OneWire.h>

#define ONE_WIRE_BUS 2
#define L298N_PIN 5

OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

unsigned long previousMillis = 0;
const long interval = 1000;
const double targetTemp = 3;
const double hysterese = 0.2;
int power = 0;
float startTemp = 0;
bool heating = true;
```

Code 1: Imports und globale Variablen

In Code 1 werden die notwendigen Bibliotheken importiert und der Sensor aufgesetzt. Außerdem werden einige globale Variablen definiert die in der Loop Funktion verwendet werden (siehe Code 3).

Bedeutung der Variablen:

- previousMillis ... speichert die vorherigen Millis ab.
- interval ... gibt das Interval an in dem die Sensordaten gemessem werden.
- targetTemp ... die Zieltemperaturdifferenz die erreicht werden soll.
- hysterese ... der Toleranzbereich der für targetTemp erlaubt ist (in diesem fall also [2.8;3.2] °C).
- startTemp ... die Temperatur zu Messbegin.
- heating ... sagt aus ob gerade geheizt wird oder nicht.

```
void setup() {
  Serial.begin(9600);
  sensors.begin();
  pinMode(L298N_PIN, OUTPUT);

  sensors.requestTemperatures();
  startTemp = sensors.getTempCByIndex(0);
  Serial.println(startTemp);

  delay(3000);
}
```

Code 2: Setup Funktion

In Code 2 wird zuerst der Serial Monitor des Arduino Nano aufgesetzt danach wird der Pin des L298N Motor Controllers konfiguriert, anschließend wird die Starttemperatur gemessen und geloggt. Zuletzt gibt es noch ein Delay von drei Sekunden um genügend Zeit zu haben, das Netzteil einzuschalten (Falls noch ein Code auf dem Arduino Nano läuft welcher die Glühbirne verwendet, bevor man den neuen Code hochlädt).

```
void loop() {
  unsigned long currentMillis = millis();

  if (currentMillis - previousMillis >= interval) {
      sensors.requestTemperatures();
      float temperature = sensors.getTempCByIndex(0);
      float tempDiff = abs(startTemp - temperature);

      if (tempDiff < targetTemp - hysterese) {
          heating = true;
      }

      if (tempDiff < targetTemp + hysterese && heating) {
          power = 50;
      } else {
          power = 0;
          heating = false;
      }

      analogWrite(L298N_PIN, power);
      previousMillis = currentMillis;

      Serial.print(float(currentMillis / (1000.0 * 60.0)));
      Serial.print('\t');
      Serial.print(power);
      Serial.print('\t');
      Serial.println(tempDiff);
  }
}
```

Code 3: Loop Funktion

In Code 3 wird mithilfe von Millis jede Sekunde zunächst die aktuelle Temperatur gemessen und dann anhand der Starttemperatur die absolute Temperaturdifferenz berechnet. Dann wird überprüft, ob diese kleiner ist als die untere Grenze des Zieltemperaturintervalls (in unserem Fall also < 2.8 °C), wenn dies der Fall ist wird heating True gesetzt. Als Nächstes setzen wir die Power (PWM-Wert, der an den L298N Motor Controller gesendet wird) auf 50, falls die absolute Temperaturdifferenz kleiner als die obere Grenze des Zieltemperaturintervalls (3.2 °C) und heating True ist. Falls dies nicht der Fall ist, wird die Power auf Null gesetzt und heating auf False.

### 1.1.2. Überprüfen der Funktionalität

Zur Überprüfung der Funktionalität des Programmcodes haben wir überprüft, ob die Starttemperatur ein realistischer Wert ist. Dann haben wir die targetTemp verkleinert (um den Überprüfungsprozess zu beschleunigen), und mit Logs überprüft ob der Programmcode wenn die Zieltemperaturintervallgrenzen erreicht werden die Glühbirne ein- bzw. ausschaltet.

## 1.2. voraussichtlicher Temperaturverlauf

Wir haben uns gemeinsam überlegt wie der Temperaturverlauf wahrscheinlich aussehen wird. Dies ist in Abbildung 1 dargestellt.

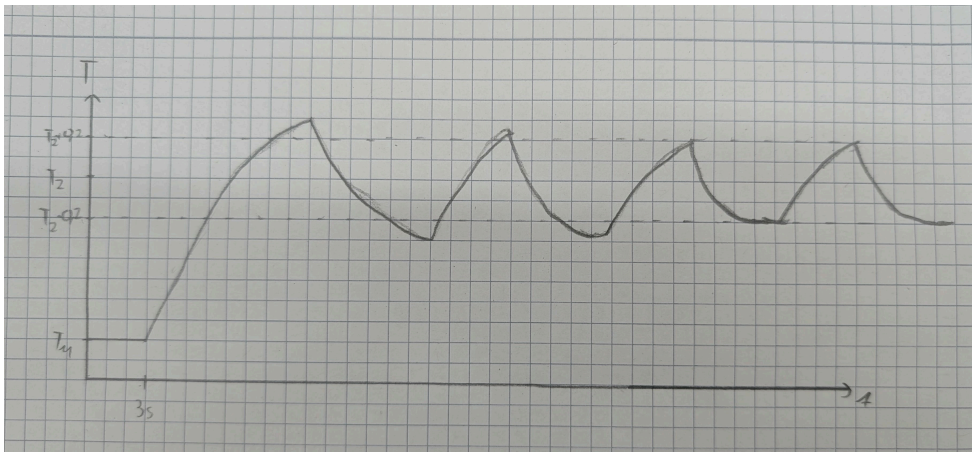


Abbildung 1: Skizze des voraussichtlichen Temperaturverlaufs

Zunächst verläuft die Temperatur genauso wie bei der vorherigen Übung mit der Sprungantwort, bis die obere Zieltemperaturintervallgrenze erreicht wird. Dann wird die Glühbirne ausgeschaltet und die Temperatur sinkt. Anfangs sinkt sie stärker und dann immer schwächer, bis die untere Zieltemperaturintervallgrenze erreicht wird. Das Abkühlen dauert deutlich länger als das Erhitzen. Dann wird die Glühbirne wieder eingeschaltet und die Temperatur steigt wieder genauso wie bei einer Sprungantwort. Dadurch entsteht ein haifischflossenartiger Graph, der sich mit der Zeit zwischen den Zieltemperaturintervallgrenzen einpendelt.

## 1.3. Messergebnisse

In Abbildung 2 ist zu sehen wie sich der Temperaturverlauf tatsächlich verhalten hat.

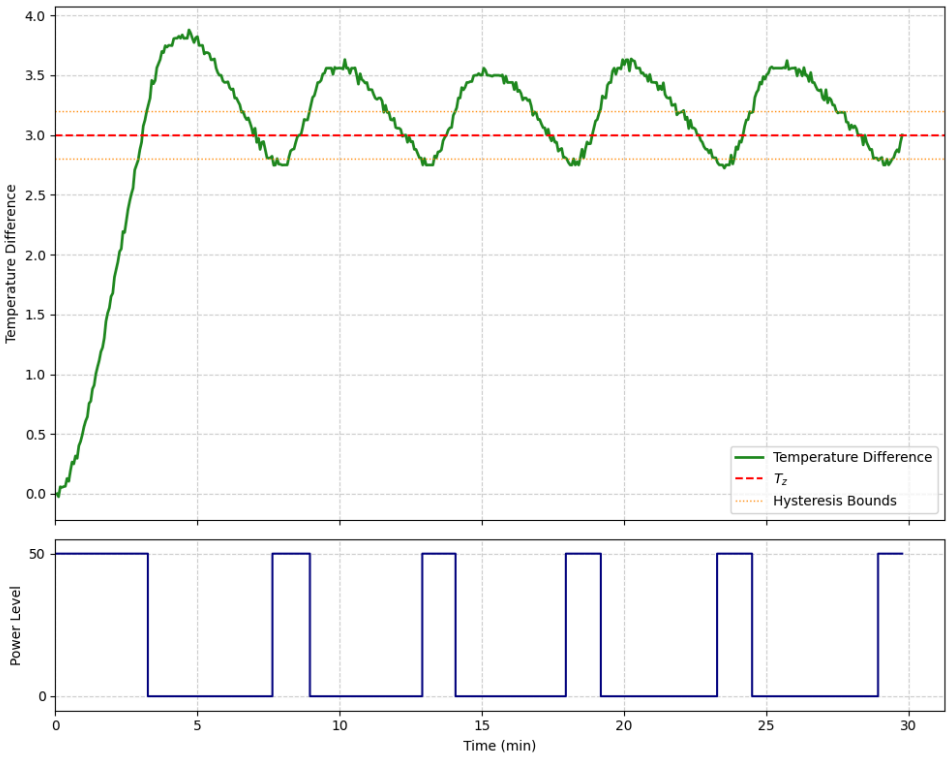


Abbildung 2: Zweipunktregelung Messung (Start Temp. 25.06°C)

### 1.3.1. Erklärung

Wenn die Glühbirne eingeschaltet ist, verhält sich der Graph ähnlich wie bei einer Sprungantwort (Erklärung im letzten Laborbericht). Das Abkühlen dauert länger als das Erhitzen, weil wir den Sensor ja nicht aktiv kühlen, sondern bloß die Glühbirne ausschalten. Die obere Hysteresegrenze wird stärker überschritten, weil, wenn wir die Glühbirne ausschalten, diese nicht automatisch auf Raumtemperatur ist und der Sensor zuerst noch von ihr erhitzt wird und erst dann abkühlen kann, wenn die Glühbirne wieder anfängt abzukühlen.

### 1.3.2. Vergleich zu Voraussage

Die Messung ist relativ ähnlich zu unserer Vorhersage. Allerdings scheinen die Haifischflossen gleich zu bleiben und pendeln sich nicht ein. Zusätzlich lässt sich sagen, dass beim Erhitzen die Intervallgrenze deutlich weiter überschritten wird als beim Abkühlen.

# 2. Wendetangentenverfahren

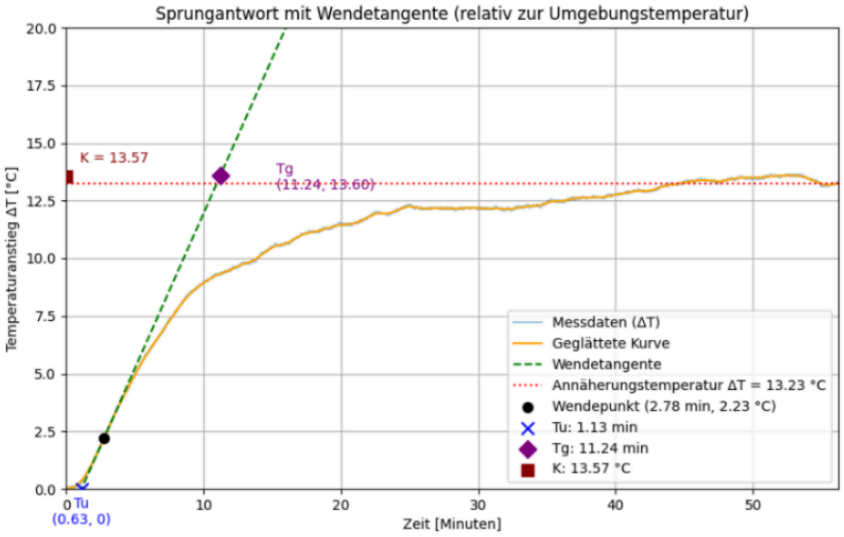


Abbildung 3: Sprungantwort mit Wendetangente

Im Diagramm der Sprungantwort wurde die Wendetangente an der Stelle der größten Steigung eingezeichnet. Die beiden Schnittpunkte der Wendetangente mit der x-Achse und mit der Asymptote K liefern die Werte für $T_u$ und $T_g$. Da zu Beginn des Versuchs 30 Sekunden gewartet wurde, bis die Glühbirne angeschaltet wurde, entspricht der grafisch ermittelte Wert $T_u$ = 1,13 min nicht dem tatsächlichen Wert – korrekt wäre, abzüglich der Wartezeit, $T_u$ = 0,63 min. Für die Berechnung von $T_g$ ergibt sich somit: $T_g$ = 11,24 – 1,13 = 10,11 min.

Die Asymptote K beschreibt den Wert, dem die Temperaturkurve langfristig zustrebt; hier wurde K mit 13,57 °C angesetzt, da dieser Wert nach rund 50 Minuten Messzeit stabil erreicht wurde. Auffällig ist, dass K zwischenzeitlich leicht überschritten wird – das liegt vermutlich an Messfehlern oder an Schwankungen der Raumtemperatur.

## 2.1. Unterschiede zur alten Prognose

Vergleicht man die Messkurve mit der Prognose aus Aufgabe 6, fallen direkt kleinere Unterschiede auf, denn einerseits steigt die reale Kurve anfangs um einiges schneller als wir es vermutet haben. Außerdem erreicht die Kurve den Endwert nicht ganz so glatt wie in der Vorhersage – das liegt wie bereits erwähnt an kleinen Messungenauigkeiten oder äußeren Einflüssen (z.B. Temperaturschwankungen im Raum). Insgesamt stimmt unsere Prognose aber ziemlich gut mit dem Ergebnis überein, besonders der Verlauf um den Wendepunkt passt gut.

Auffällig ist, dass die Kurve insgesamt einen sigmoidalen Verlauf zeigt: Nach einer kurzen Verzögerung geht es schnell los, wird am Wendepunkt steiler und flacht zum Endwert hin wieder ab – abgesehen von kleinen Messungenauigkeiten. Die Anfangsträgheit ist nur sehr kurz zu sehen, aber der typische S-förmige Verlauf einer Sigmoid-Funktion bleibt trotzdem erhalten.

# 3. Quellen

- Perplexity – wurde zur Rechtschreibüberprüfung verwendet
- Gemini – wurde zur Rechtschreibüberprüfung verwendet