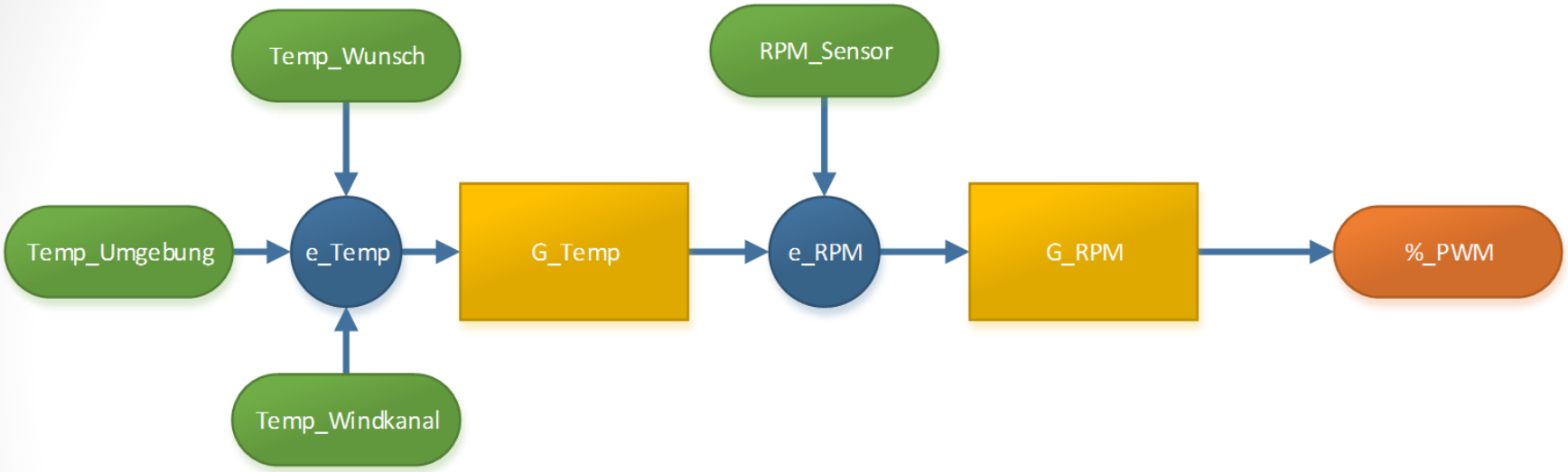
Agenda

1. **Zielstellung und Motivation**
2. **Aufbau des Windkanals**
3. **Regelung mit MATLAB/Simulink**
   1. Temperaturregelung
   2. Drehzahlregelung
4. **GUI & native Programmierung**
   1. GUI & Paketaufbau
   2. Vergleich mit nativem C++ Code
5. **Vorstellung & Ausblick**
6. **Probleme & Fazit**

1. Zielstellung und Motivation

1. **Motivation**
   Wie kann ich mein Modell auf Hardware bringen?
2. **Gesamtziel**
   Regelung einer unbekannten physischen Strecke mittels bekannter Regler in Simulink und Visualisierung des Regelvorgangs mittels GUI.
3. **Aufgaben**
   1. Aufbau des Windkanals
   2. Ansteuerung des Arduino und der Peripherie in Simulink
   3. Anbindung Temperatursensorik und Temperaturregelung
   4. Entwicklung Drehratensensorik und Drehratenregelung
   5. Anbindung des Modells an eine MATLAB GUI
   6. Vergleich mit nativer Programmierung in C++

2. Aufbau des Windkanals



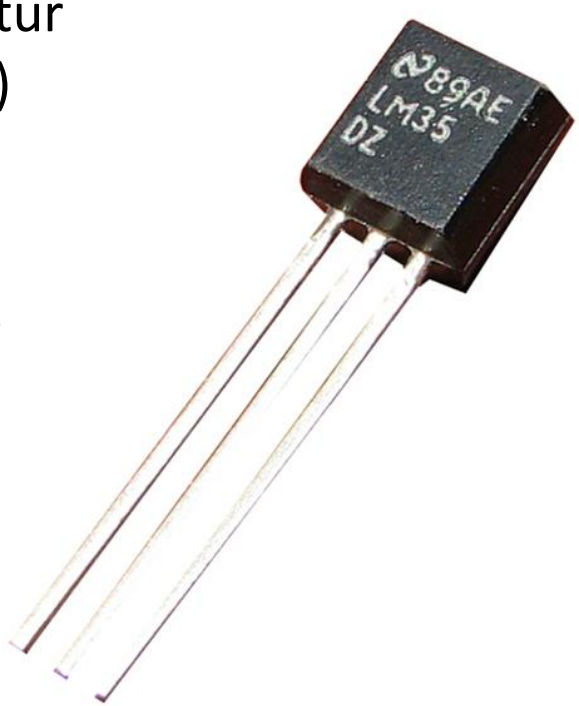
Blockschaltbild Windkanalregelung, erstellt mit MS Visio 2013

Heizung:

- $7{,}92\,\mathcal{V} * 0{,}25\,\mathcal{A} = 1{,}98\,\mathcal{W}$
- $T_{max} \sim 77\,°\mathrm{C}$

3.1. Temperatursensor LM35

- Einfache Auswertung durch linearen Zusammenhang zwischen Temperatur und Ausgangsspannung (10 mV/ °C)
- Kleine Baugröße
- Temperaturbereich -55 °C - +150 °C



3.1. Temperatursensorauswertung

- Analoger Eingang des Arduino:

  0 V → 0x000

  U_max → 0x3FF

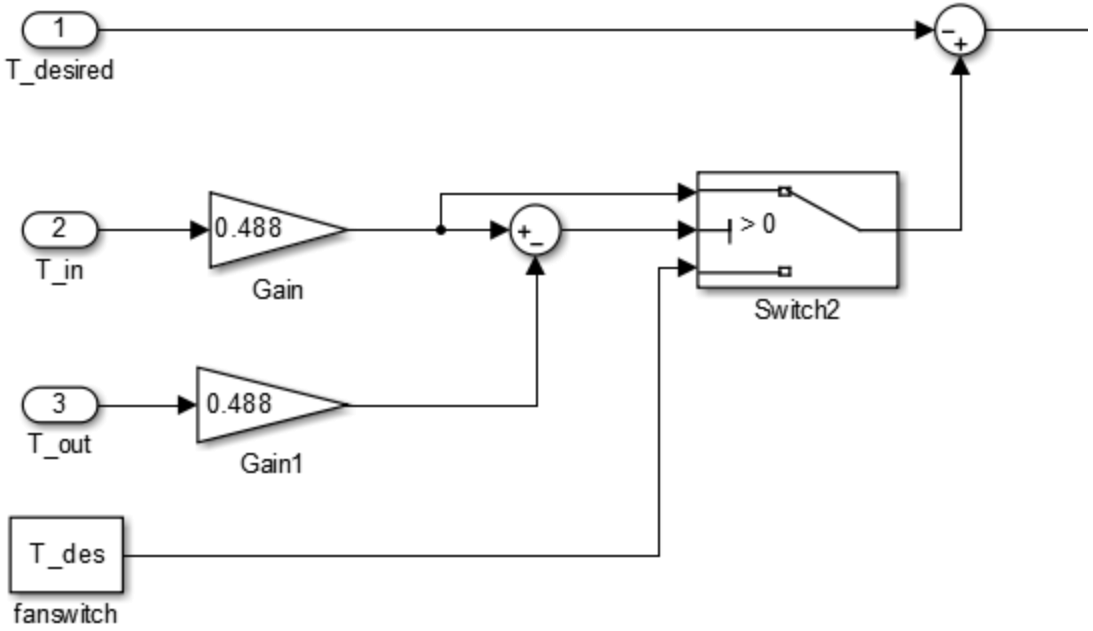
→ 4,88 mV pro Bit bei U_max = 5 V

- Linear ohne Offset 1 mV/°C

→ 1 Bit = 0,488 °C

3.1. Umwandlung von T-Stellgröße in n-Stellgröße

- (T_out > T_in) → minimale Drehzahl
- (T_in – T_soll) > 5 °C → maximale Drehzahl
- (T_in – T_soll) < 0 °C → minimale Drehzahl
- 0 °C < (T_in – T_soll) < 5 °C → linearer Zusammenhang



3.1. Umwandlung von T-Stellgröße in n-Stellgröße

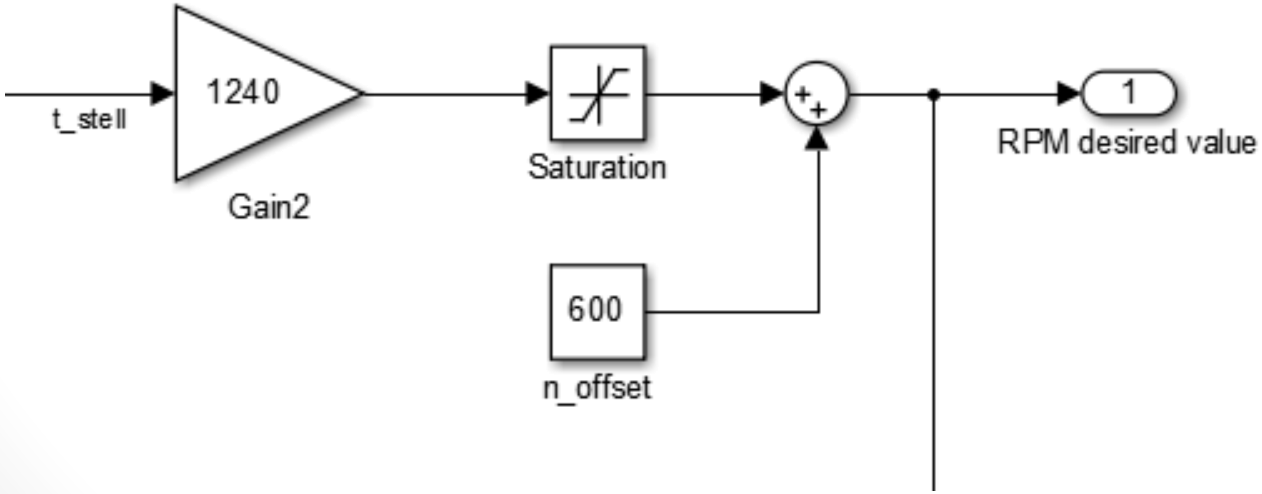
n_max = 6800

n_min = 600

Gain = 6200/5 = 1240

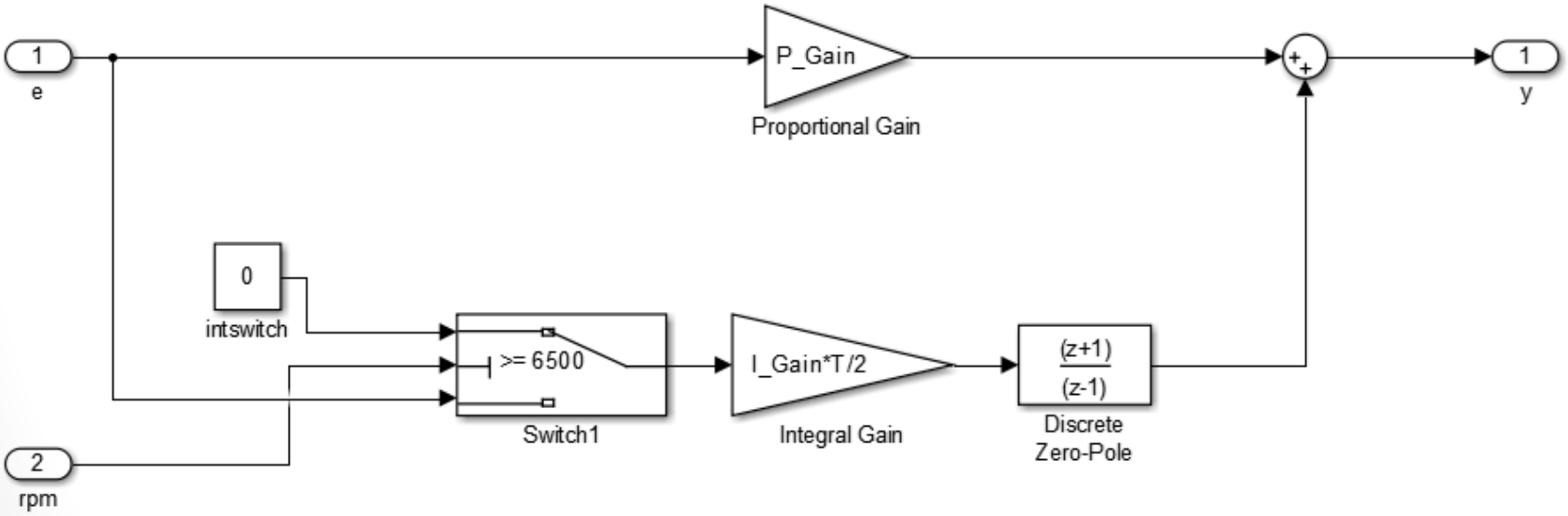
Saturation: 0 - 6200

Offset = 600

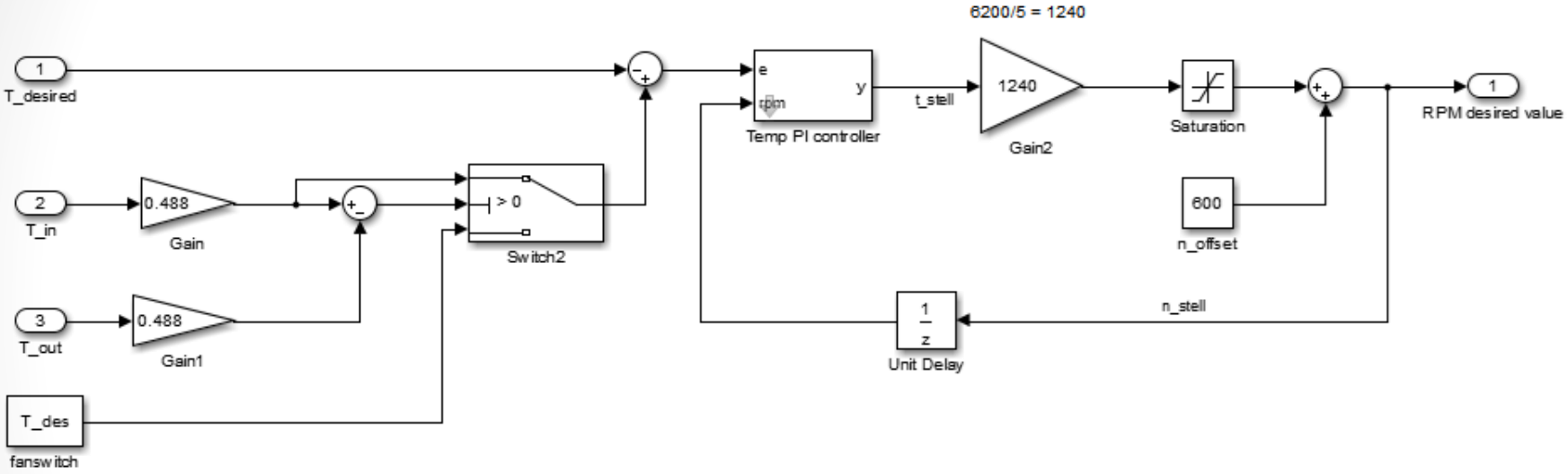


3.1. Temperaturregelung mit PI-Regler

- Bilineare Transformation
- Drehzahl-Stellgröße übersteigt 6500 → I-Glied „abschalten“

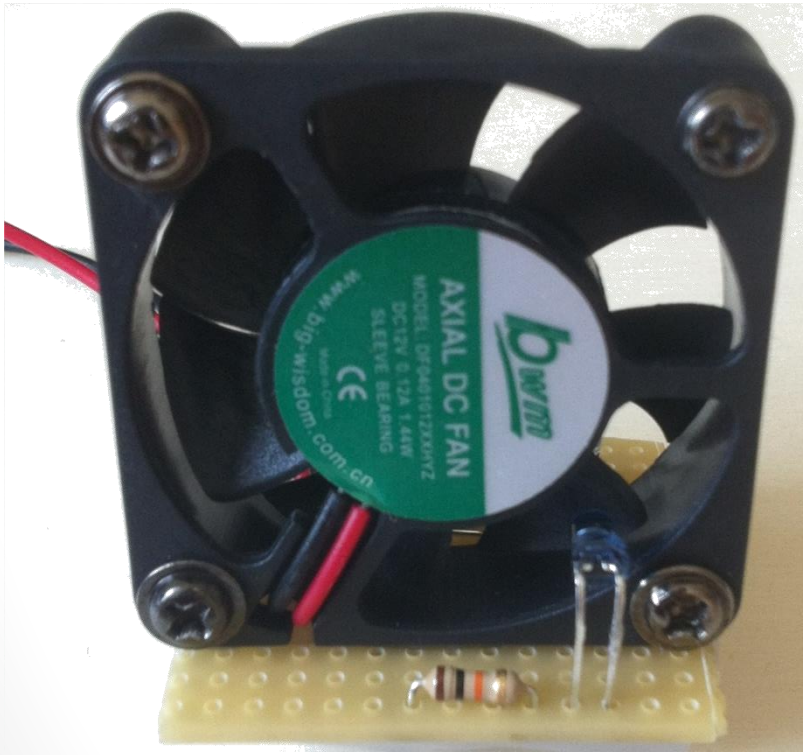


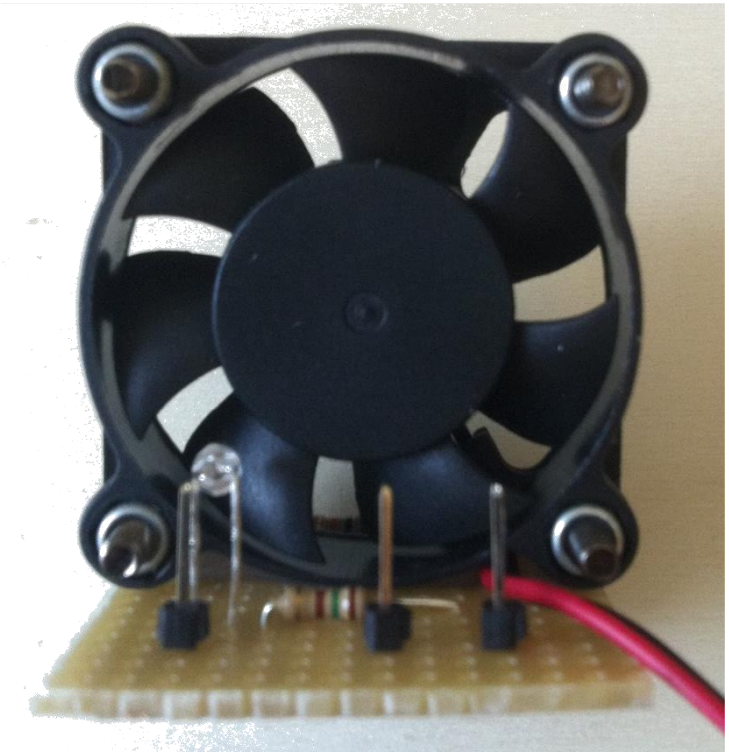
3.1. Gesamte Temperaturregelung



3.2. Drehzahlsensor

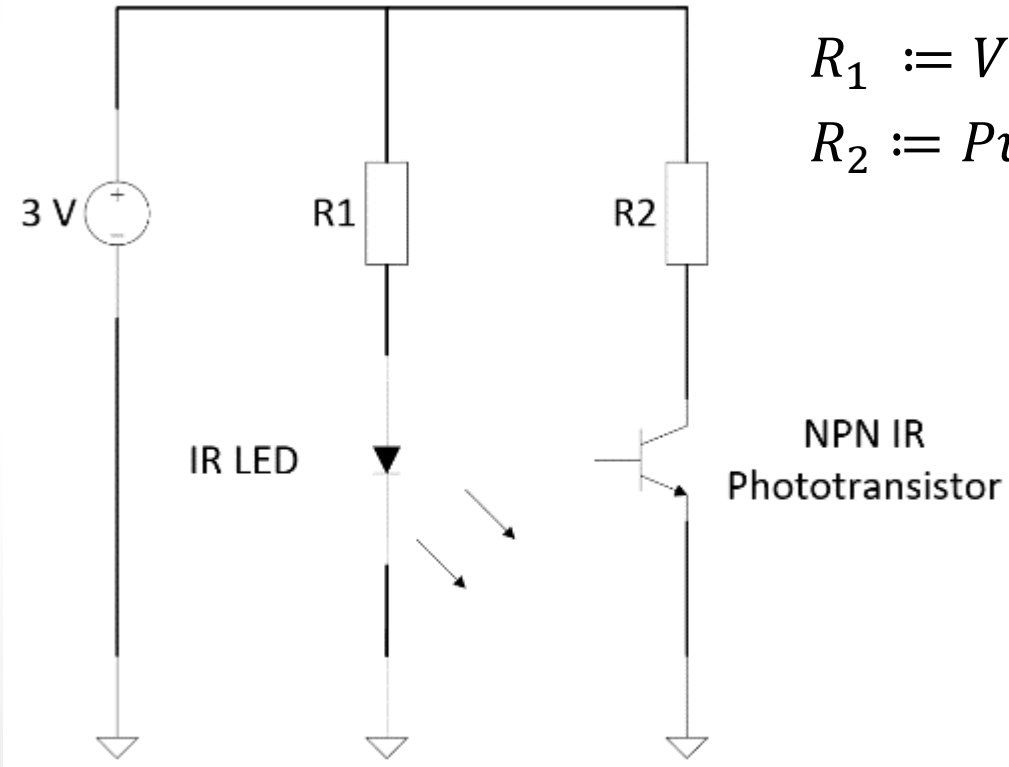
Infrarot LED / Infrarot Phototransistor





3.2. Drehzahlsensor

Infrarot LED / Infrarot Phototransistor

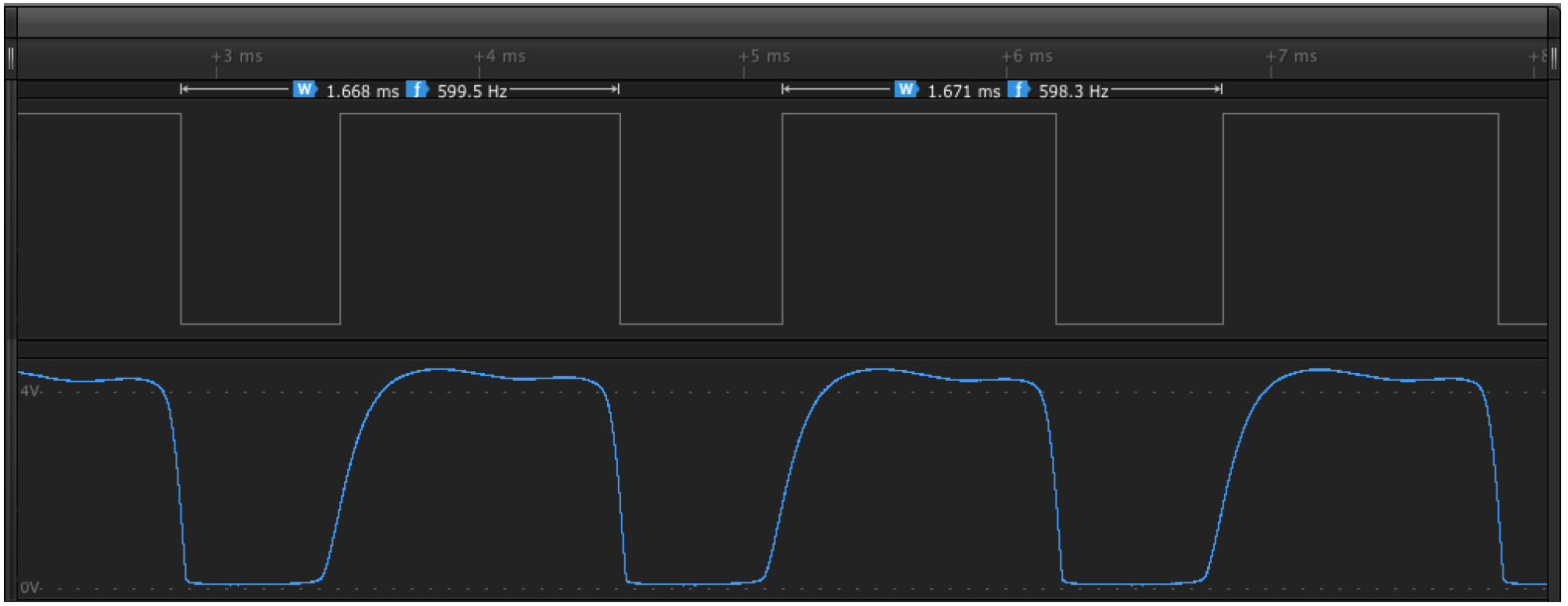


$R_1 \coloneqq Vorwiderstand$

$R_2 \coloneqq Pullup - Widerstand$

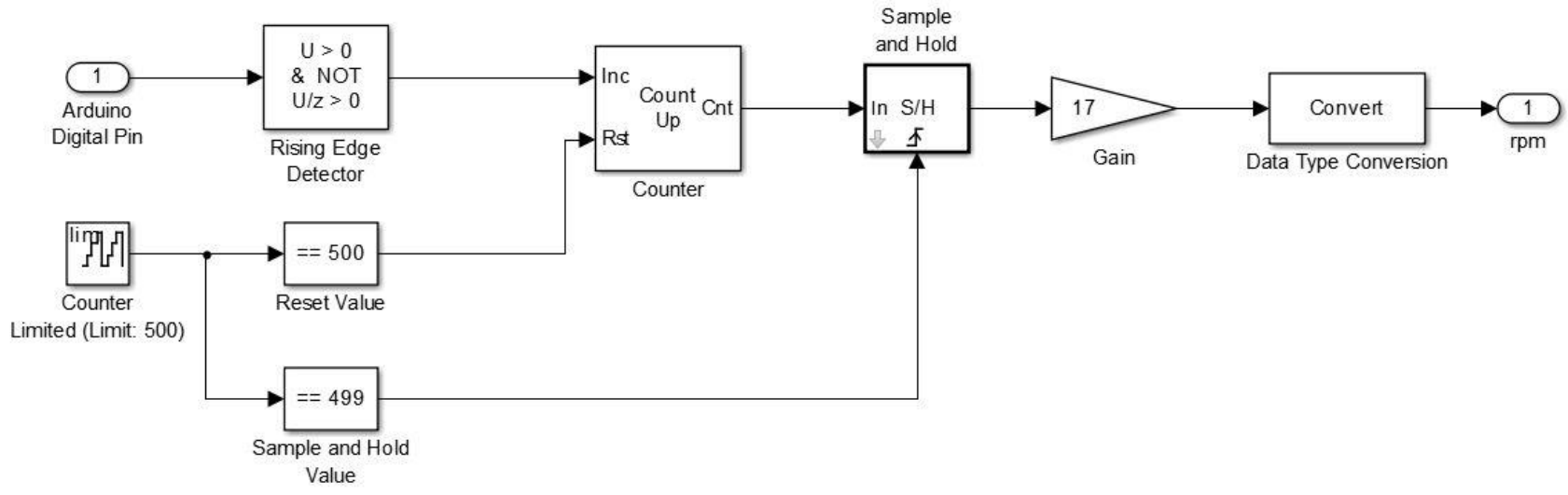
3.2. Drehzahlsensor

Lüfterdrehzahl: 0 bis 6800 rpm

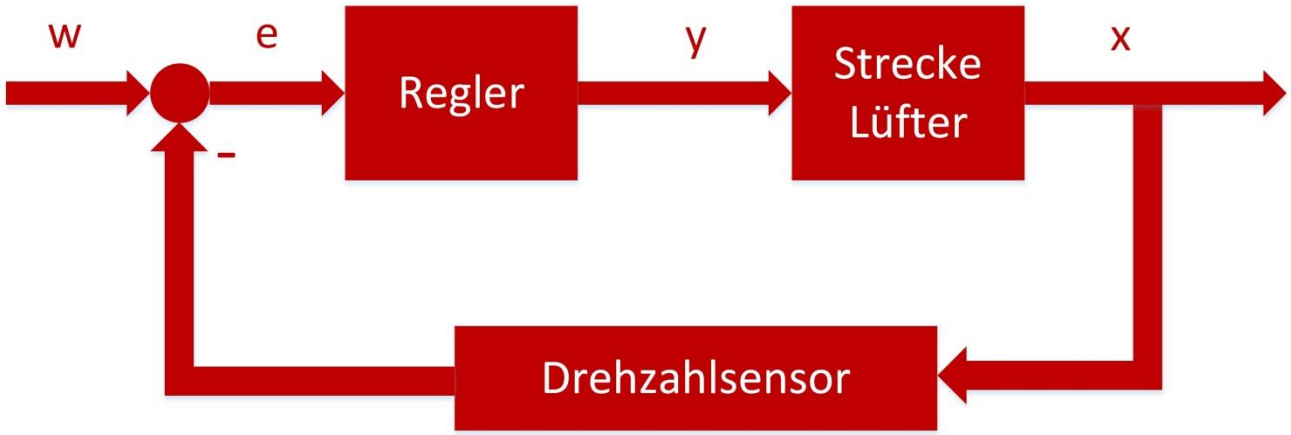


Aufzeichnung bei 5150 rpm, erstellt mit Logic Pro 16 von Saleae

3.2. Drehzahlsensor



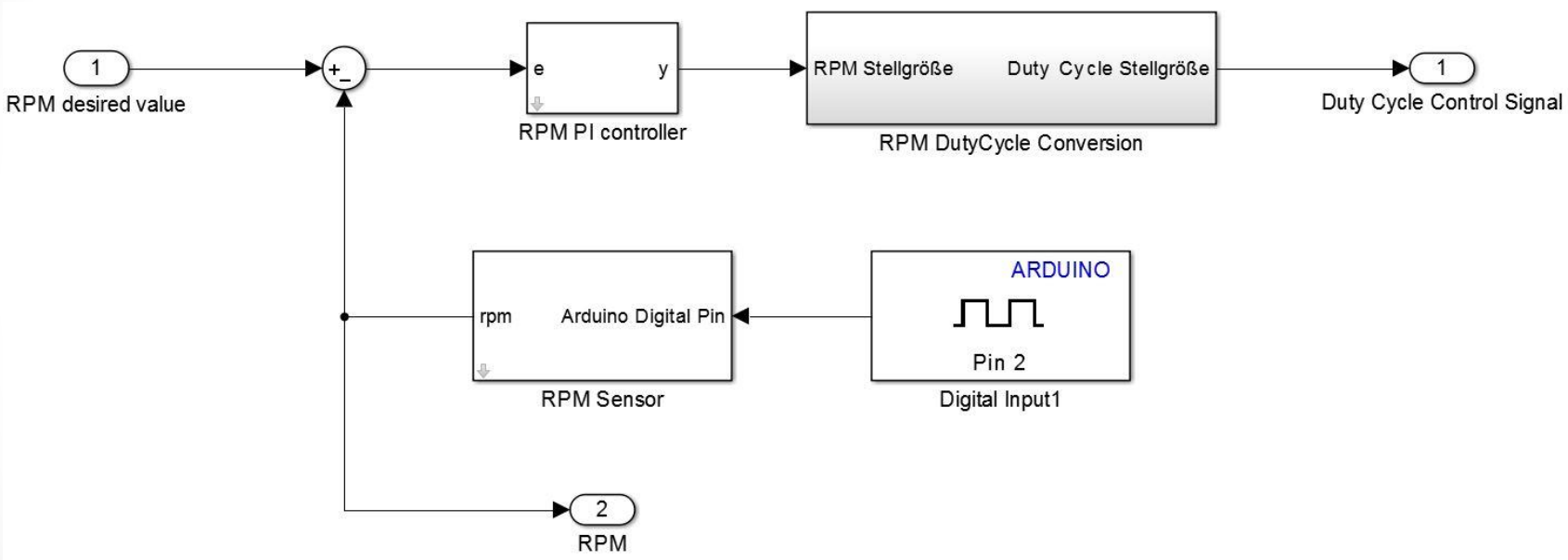
3.2. Drehzahlregelung



- Unbekannte Strecke
- Simulation verschiedener Reglerparameter

  → Ziel: stabile und genaue Regelung der Drehzahl

3.2. Drehzahlregelung



Drehzahlregelung

- Drehzahlregler:

$$PI - Regler: G(s) = k_{pr} + k_i * \frac{1}{s}$$

$$G(z) = k_{pr} + k_i * \frac{T}{2} * \frac{z+1}{z-1}$$

- Laplace-Bereich deutlich rechenintensiver als z-Bereich
- Probleme mit Sample-Zeiten

4.1. GUI & Paketaufbau

GUI:

- Erstellt mit MATLAB
- Liest Daten von Arduino in Form von Paketen ein
- Speichert Werte im CSV-Format

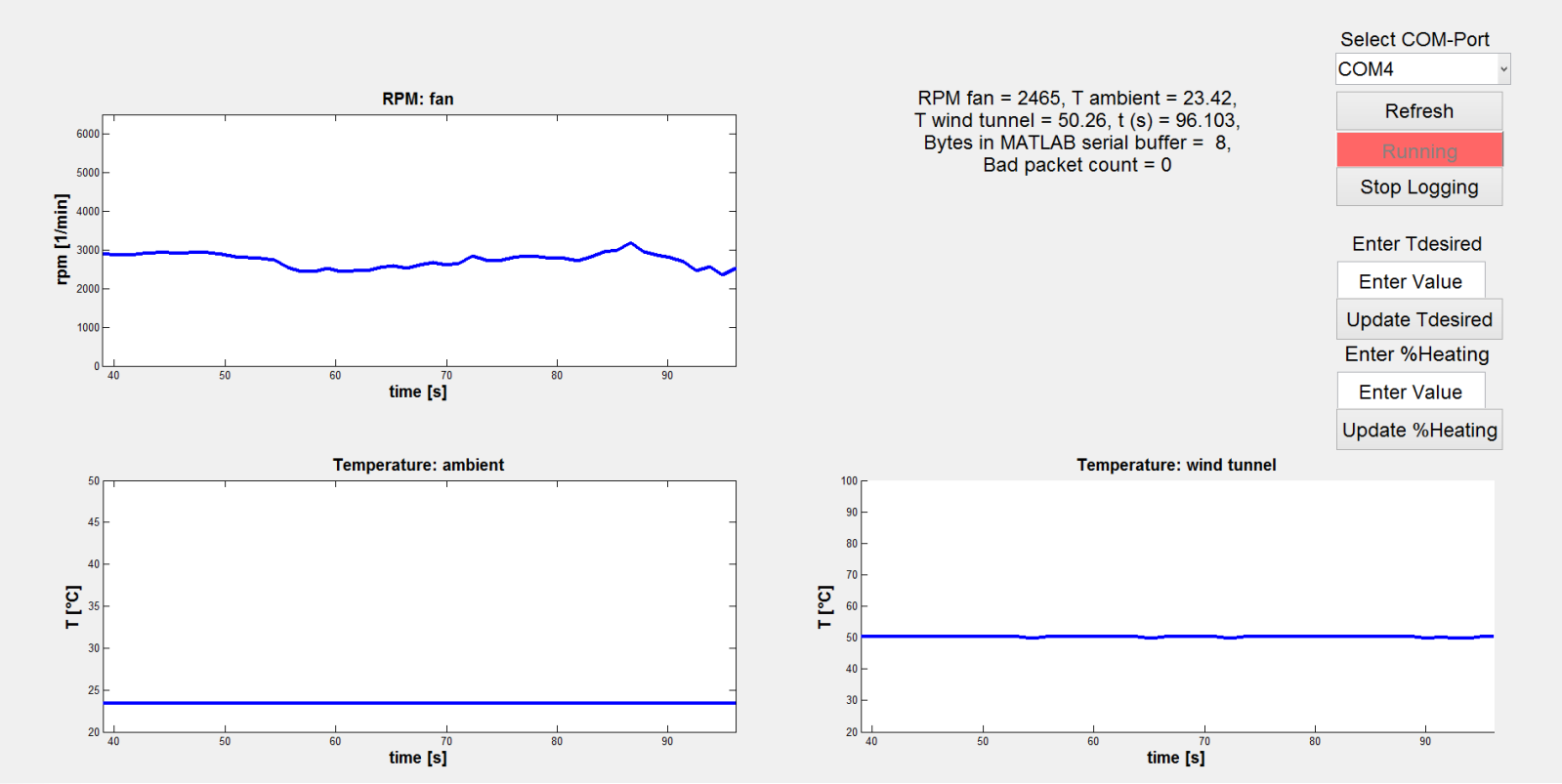
Paketaufbau:

- 16-Bit Start of Frame Delimiter
- 8-Bit $T_{Ambient}$
- 8-Bit $T_{Windkanal}$
- 16-Bit RPM
- 16-Bit End of Frame Delimiter

4.2. Vergleich mit nativem C++ Code

Simulink	C++
Maximal 2 Hz Aktualisierungsrate	Maximal 40 Hz Aktualisierungsrate
Simulink Modell bestimmt Aktualisierungsrate	GUI fordert Daten mit festlegbarer Rate an
PWM Block fest bei 495 Hz	PWM kann beliebige Frequenz haben
PWM Wertebereich 0 bis 255	PWM Wertebereich 0 bis 65535
RPM Sensor mit Polling	RPM Sensor mit Interrupts
Kommunikationsprotokoll schwer zu implementieren	Einfache Implementierung eines Protokolls zur Konfiguration (Wunschtemperatur, Heizleistung, etc.)

5. Vorstellung & Ausblick



GUI-Screenshot bei T_{Wunsch} = 50 °C

5. Vorstellung & Ausblick

Ideen:

- Aktive Kühlung mit Peltier Element und CPU-Lüfter
- Genauere Charakterisierung der Strecke und etwaiger Störgrößen durch mehr Temperatursensoren und Luftmassenmesser

Verbesserungen:

- Erhöhung der Heizleistung und der maximalen RPM
- Optimierung des RPM Sensors hinsichtlich der Flankensteilheit
- Erhöhung der Genauigkeit der RPM Berechnung
- Entwicklung von Messwertverarbeitung zur Filterung etc.