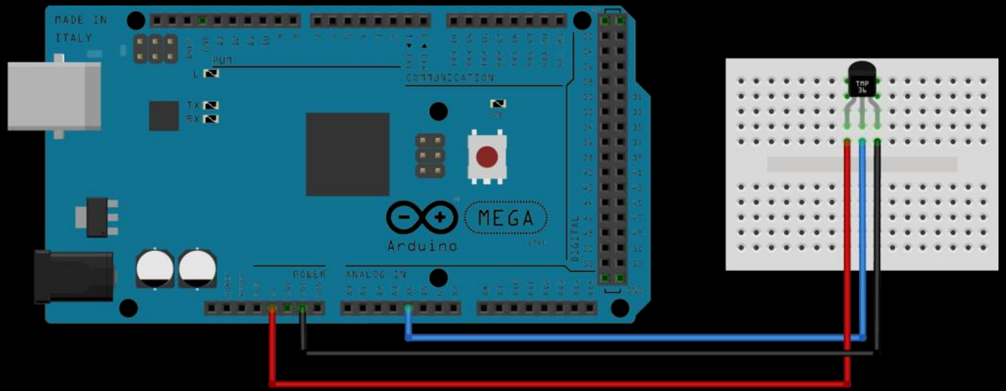
Contents

Temperature Sensor TMP36 and Low Pass Filter



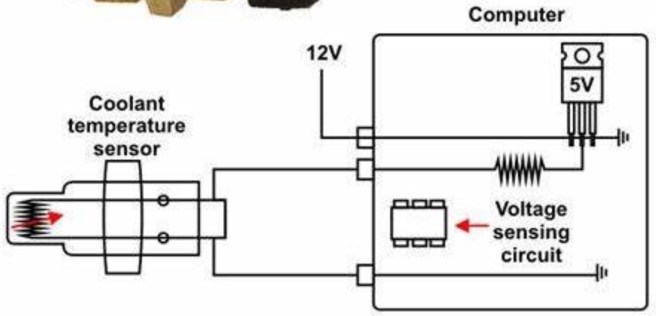
Key Learning Points

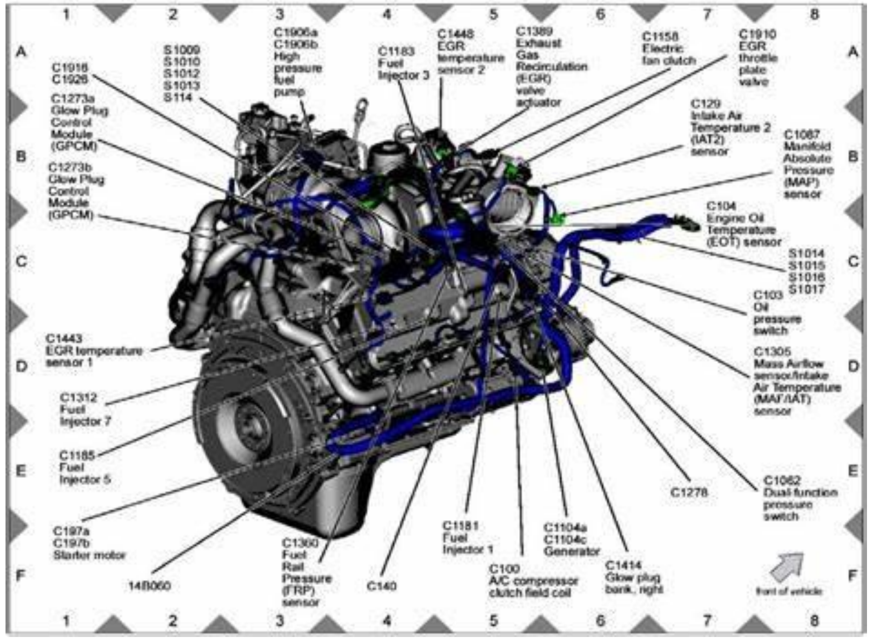
After this Lecture, you will be able to:

- Obtain measured temperature data from the TMP36 sensor, and undertake the process of filtering noisy measured data using a low pass filter

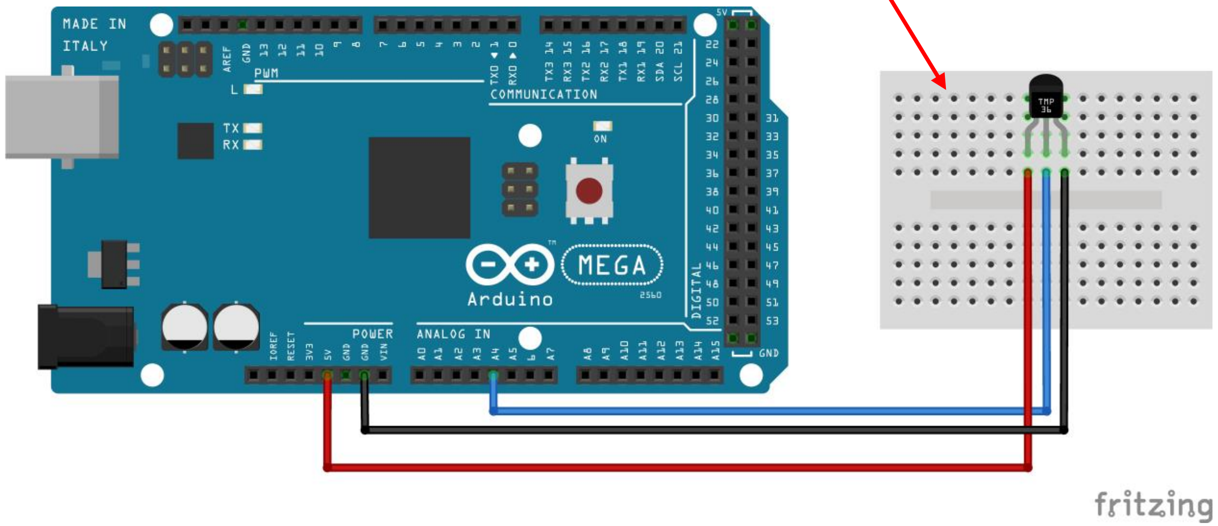
3.1 Hardware

- Required hardware for the exercise:
  i. Supported Arduino Uno board
  ii. USB cable
  iii. Low voltage temperature sensor (TMP36)
  iv. 3 x male-male wires
- Voltage input: $2.7V$ to $5V$
- Operating temperature: -40$°C$ to +125°C

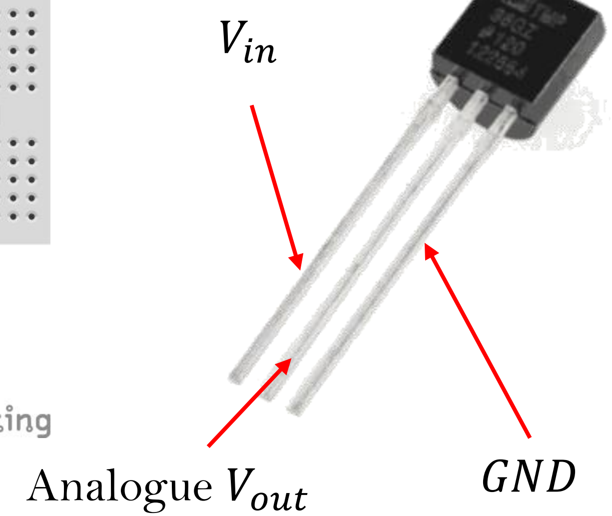




Use the breadboard on the Control-Lab-in-a-Box (CLB) rig



Low voltage temperature sensor (TMP36)



3.2 Algorithm Design

- Output from Pin 4 is a 10-bit analogue value ranging from 0 to 1023
- E.g., 150 output from Pin 4, i.e.

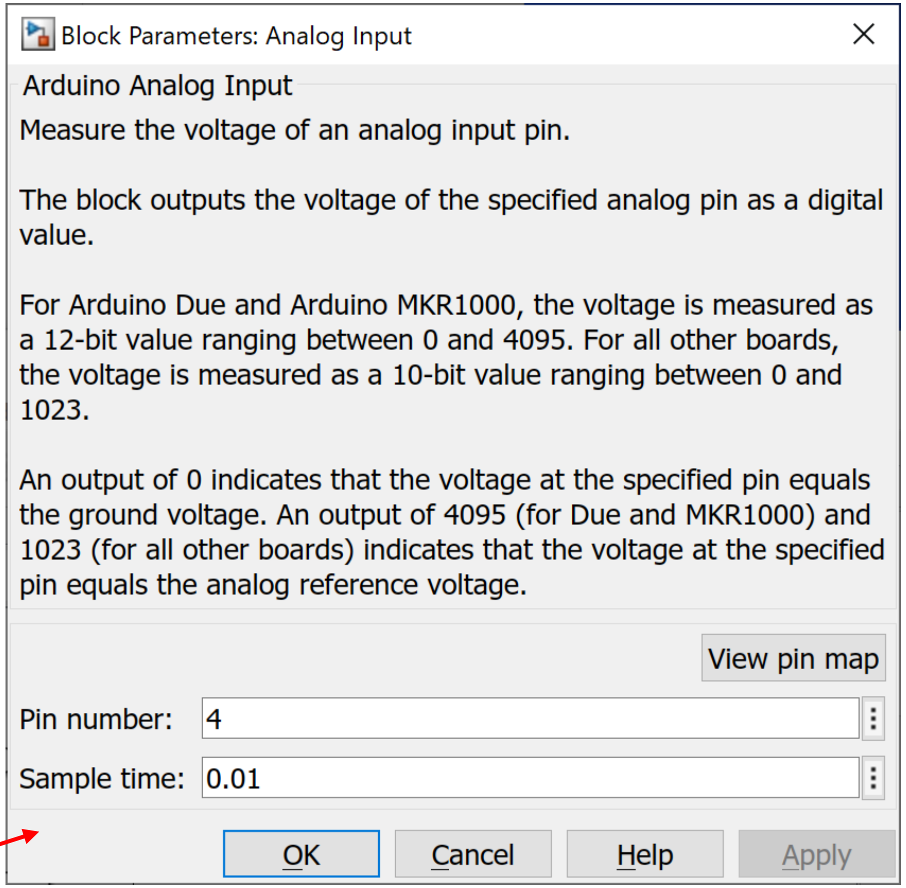
$$\frac{150}{1023} = 0.1466$$

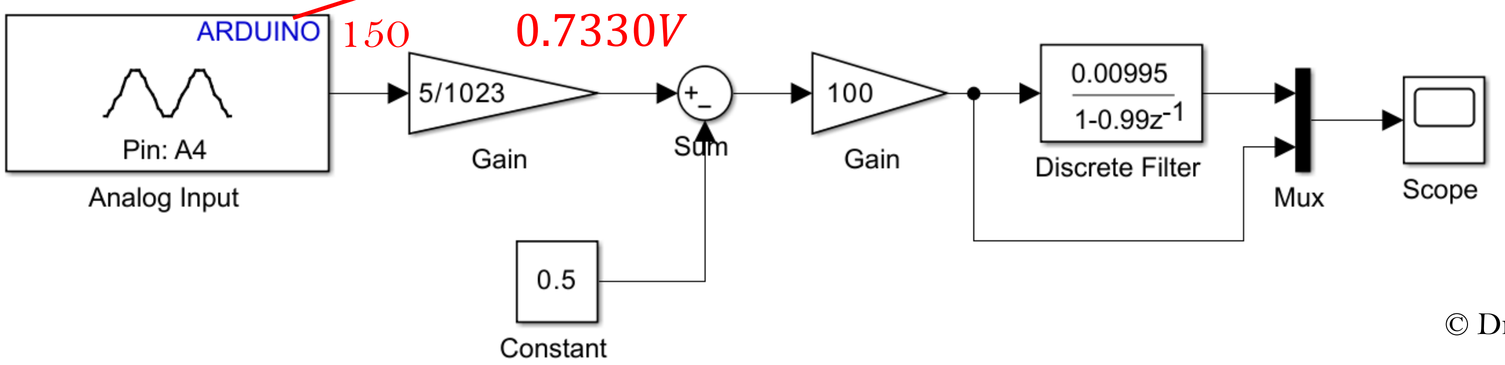
  which is basically a ratio between 0 and 1

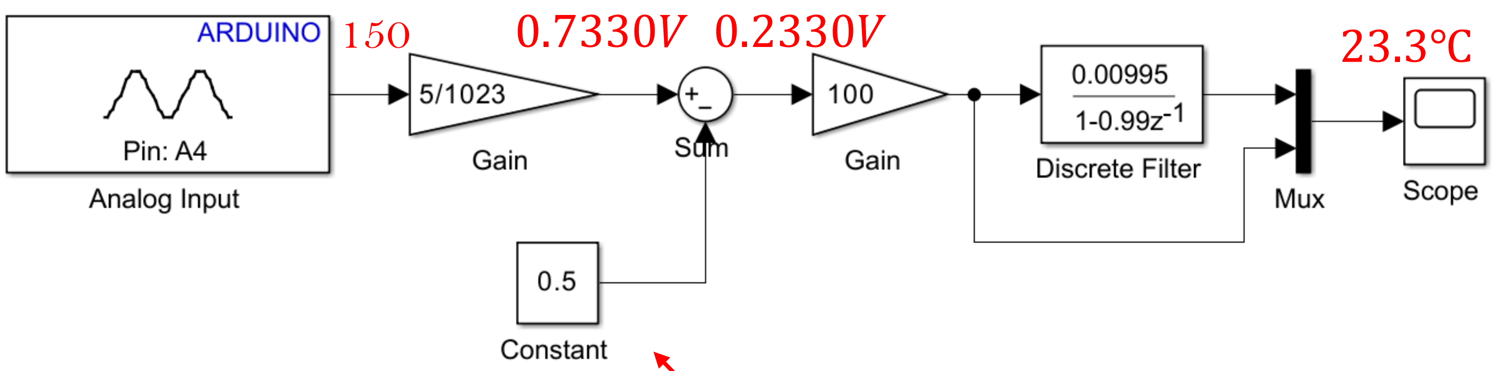
- Multiplied by 5V (Arduino supply) gives:

$$0.1466\ x\ 5 = 0.7330V$$

Select sampling interval T_s of 0.01 seconds







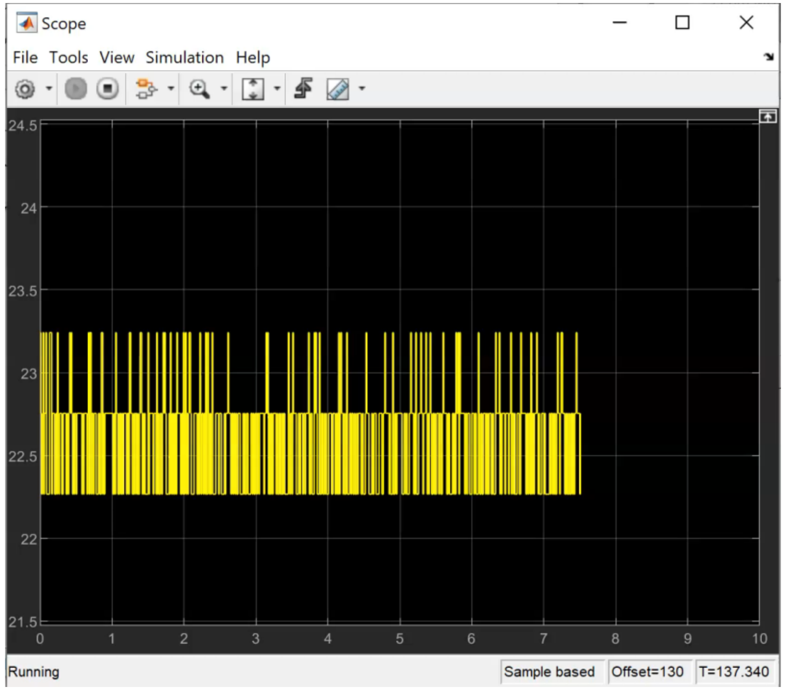
3.2 Algorithm Design

- TMP36 has a voltage offset of $0.5V$ (so that negative values can be accounted for), e.g.,

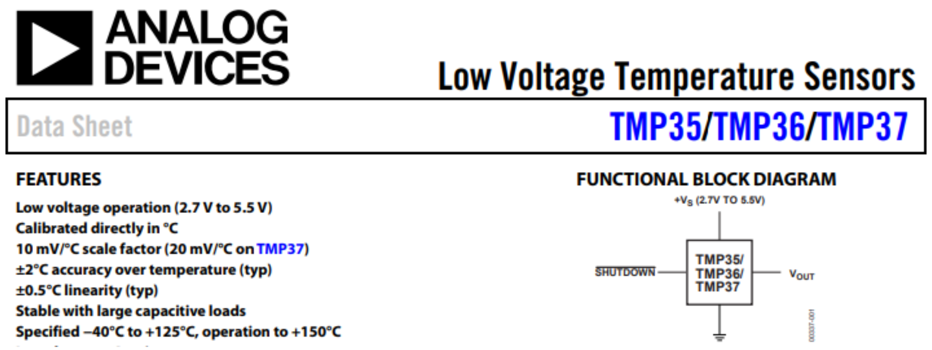
$$0.7330 - 0.5 = 0.2330V$$

- The above value is then multiplied by 100, as the TMP36 sensor has an output scale factor of $10mV/\,°\mathrm{C}$, e.g.,

$$0.2330\left(\frac{1°\mathrm{C}}{0.01V}\right) = 23.3°C$$



Sensor	Offset Voltage (V)	Output Voltage Scaling $\left(\frac{mV}{°C}\right)$	Output Voltage @$25°C\,(mV)$
TMP35	0	10	250
TMP36	**0.5**	**10**	**750**
TMP37	0	20	500



ANALOG DEVICES

Low Voltage Temperature Sensors

Data Sheet TMP35/TMP36/TMP37

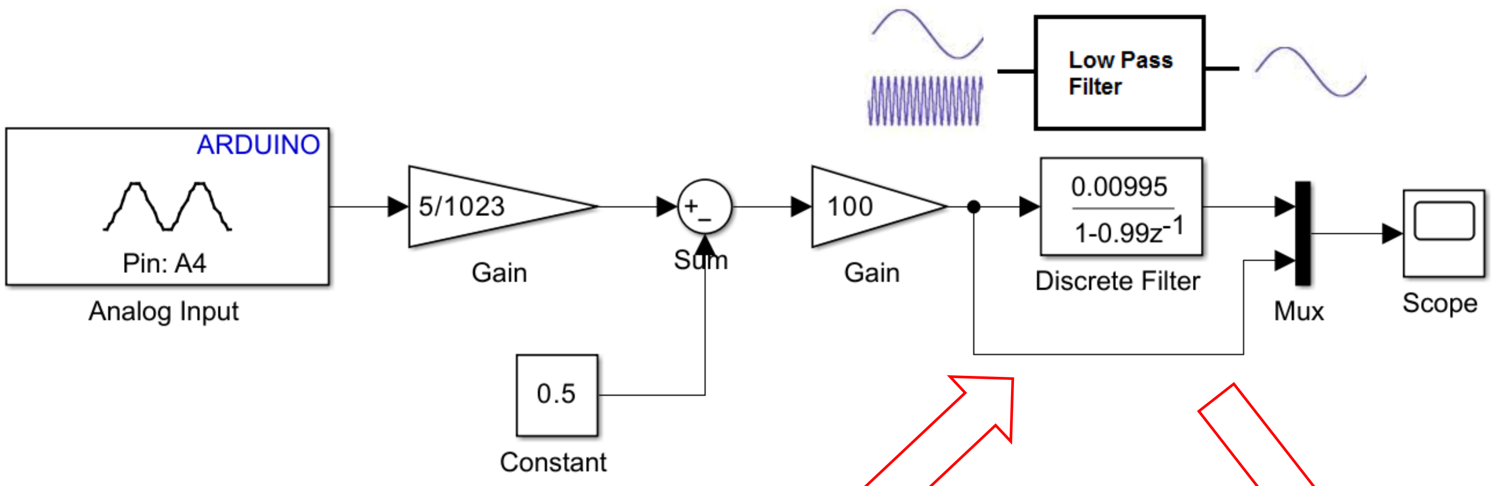
FEATURES

Low voltage operation (2.7 V to 5.5 V)
Calibrated directly in °C
10 mV/°C scale factor (20 mV/°C on TMP37)
±2°C accuracy over temperature (typ)
±0.5°C linearity (typ)
Stable with large capacitive loads
Specified −40°C to +125°C, operation to +150°C

FUNCTIONAL BLOCK DIAGRAM

3.2.1 Low Pass Filter (Digital)

- Low pass filter applied (stops high frequencies passing through)
- A continuous-time transfer function of an RC (time constant τ) circuit has been discretised using MATLAB
- Ensure the sampling interval T_s used throughout is consistent

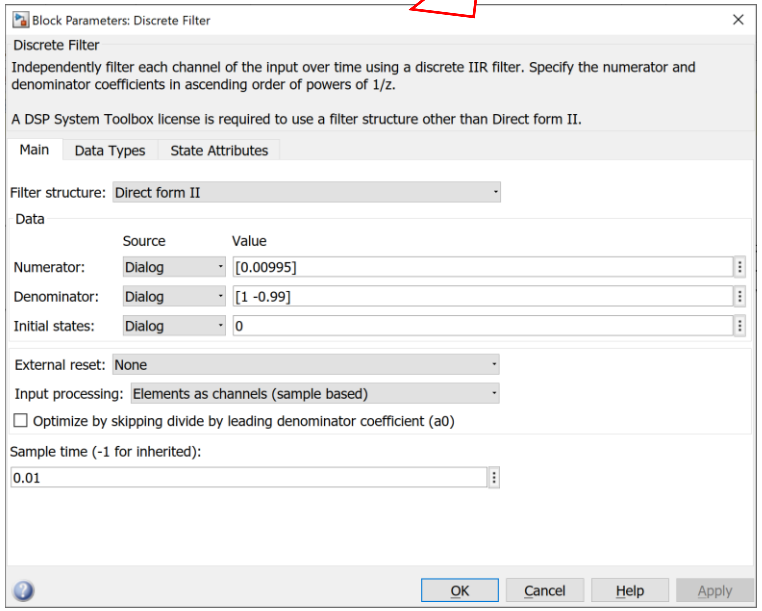


- The discrete-time transfer function will appear in the command window (insert the values into the 'Discrete Filter'
- Recall: $G(s) = \dfrac{1}{RCs+1} = \dfrac{1}{\tau s+1}$

```
clear; clc; close all;

%% Continuous and Discrete Transfer Function
R=1000; % resistor [Ohms]
C=1000*10^(-6); % capacitor [F]
Tau= R*C; % time constant
G = tf([1],[Tau 1]);

T_s = 0.01; % sample interval
Gd = c2d(G,T_s) %ZOH method by default
```



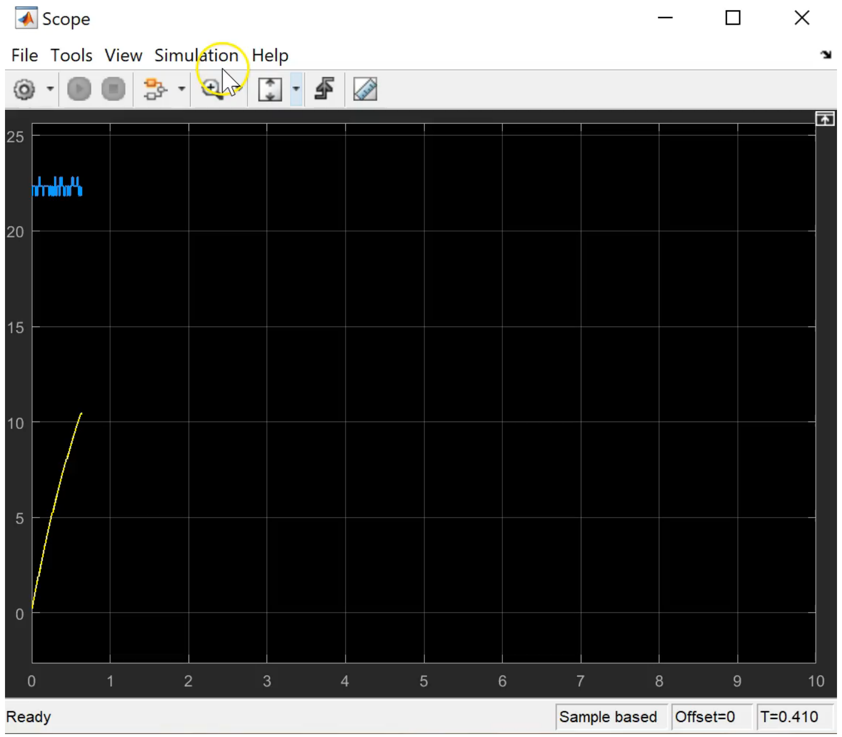
3.2.1 Low Pass Filter (Digital)

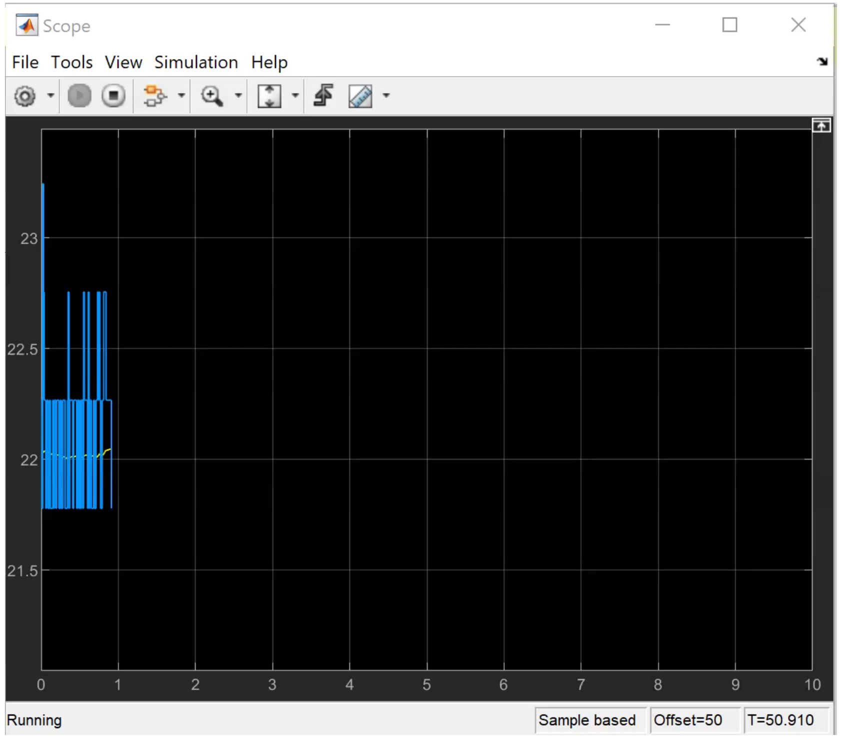
- Unfiltered signal is blue, and the filtered signal is white
- By adding the filter, the temperature estimate is much less noisy
- A drawback of the filtering is that it adds a delay/lag (this is visible when inspecting the unfiltered and filtered signal), hence no information is provided

- Recall:

$$G(s) = \frac{1}{RCs + 1} = \frac{1}{\tau s + 1}$$

- Reducing the time constant τ (i.e., RC) will reduce the lag, however the trade-off is that the noise won't be filtered as well





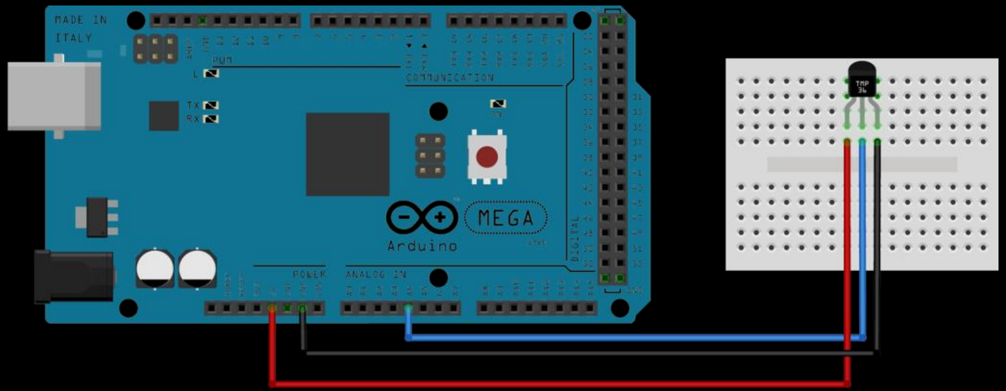
3.2.2 Exercise

- **Further task**: investigate different sampling intervals T_s and time constants τ
- Fill in the table

Sample interval, T_s [seconds]	Time constant, τ (i.e., RC value)	Filter lag [seconds]	Filtering performance [worse/better]
0.01	1	5	N/A as baseline

Contents

Temperature Sensor TMP36 and Low Pass Filter



3.3 Summary

- The low voltage TMP36 temperature sensor has been used with an Arduino Uno to capture data (hardware)
- Simulink has then been used to develop a signal processing algorithm to convert the data outputted from the temperature sensor into a physical value (i.e., degree Celsius)