Pictures/Ntnu_logo.png

Designnotat

Tittel: Designprosjekt 5 - Digitalt termometer	
Forfattere: Temmam Nabil Altaie	
Versjon: 2.0	Dato: 15.09.25

Innhold

1 Problembeskrivelse **2**

2 Prinsipiell løsning **2**

2.1 Analog del . . . 3

2.2 Digital del: ADC og estimator . . . 5

2.2.1 Parametere og symboler . . . 5

2.2.2 Kalibrering . . . 6

2.3 Digital del: Display . . . 6

3 Realisering og test **6**

3.1 Analog del . . . 7

3.1.1 Bestemmelse av forsterkning A . . . 7

3.1.2 Lavpassfilter . . . 8

3.1.3 Oppsummerte komponentverdier . . . 8

3.2 Digital del: ADC og estimering av $\hat{v}_{in}$ . . . 8

3.3 Konvertering til temperatur og visning . . . 8

3.4 Oppsummering av digitale variabler . . . 9

3.5 Resultater . . . 9

3.5.1 Referansetemperatur . . . 9

3.5.2 Romtemperatur . . . 10

3.5.3 Lunkent vann . . . 11

3.6 Oppsummering av måleverdier . . . 12

4 Konklusjon **13**

5 Takk **14**

Referanseliste **15**
Vedlegg **16**
A main.c 16

---

1 Problembeskrivelse

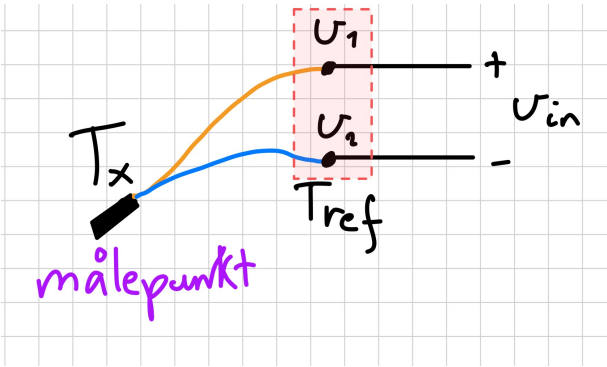
Hensikten med denne rapporten er å presentere utviklingen av en enkel prototype på et digitalt termometer. Løsningen skal vise hvordan et K-type termoelement kan kombineres med en analog forsterker, en mikrokontroller med A/D-omformer og et programbibliotek for temperaturberegning, slik at målt temperatur kan vises på en OLED-skjerm.

Systemet baseres på standard elektroniske komponenter og en rettfram signalbehandlingskjede. Prototypen skal tilfredsstille følgende punkter:

- Temperaturen skal fremkomme i °C med minst én desimal.
- Det skal gjennomføres målinger ved tre eller flere ulike temperaturer.
- Verdiene skal vises på display, og nøyaktighet og mulige avvik skal analyseres.

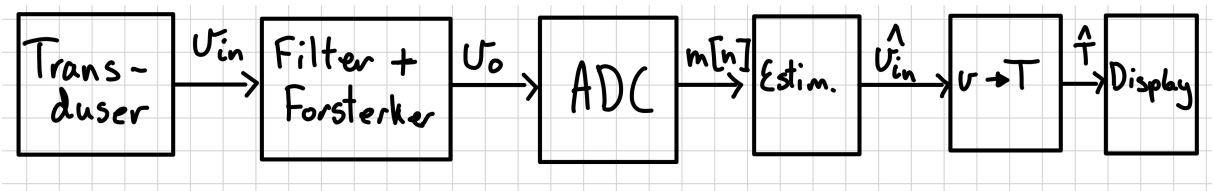
2 Prinsipiell løsning

For å realisere et digitalt termometer benyttes et termoelement som sensor:



Figur 1: Illustrasjon av et K-type termoelement med målepunkt Tx og referansepunkt T_{ref}. Tilpasset fra [1].

I figur 1 vises prinsippet bak termoelementet, som danner en differansespenning $v_{in} = v_1 - v_2$. Denne spenningen er proporsjonal med temperaturforskjellen mellom referansepunktet T_{ref} og målepunktet T_x. Figuren under (figur 2) illustrerer hvordan signalet behandles frem til det til slutt vises på et display:



Figur 2: Blokkskjema som viser hvordan signalet går fra termoelement til visning på OLED-display. Tilpasset etter [2]

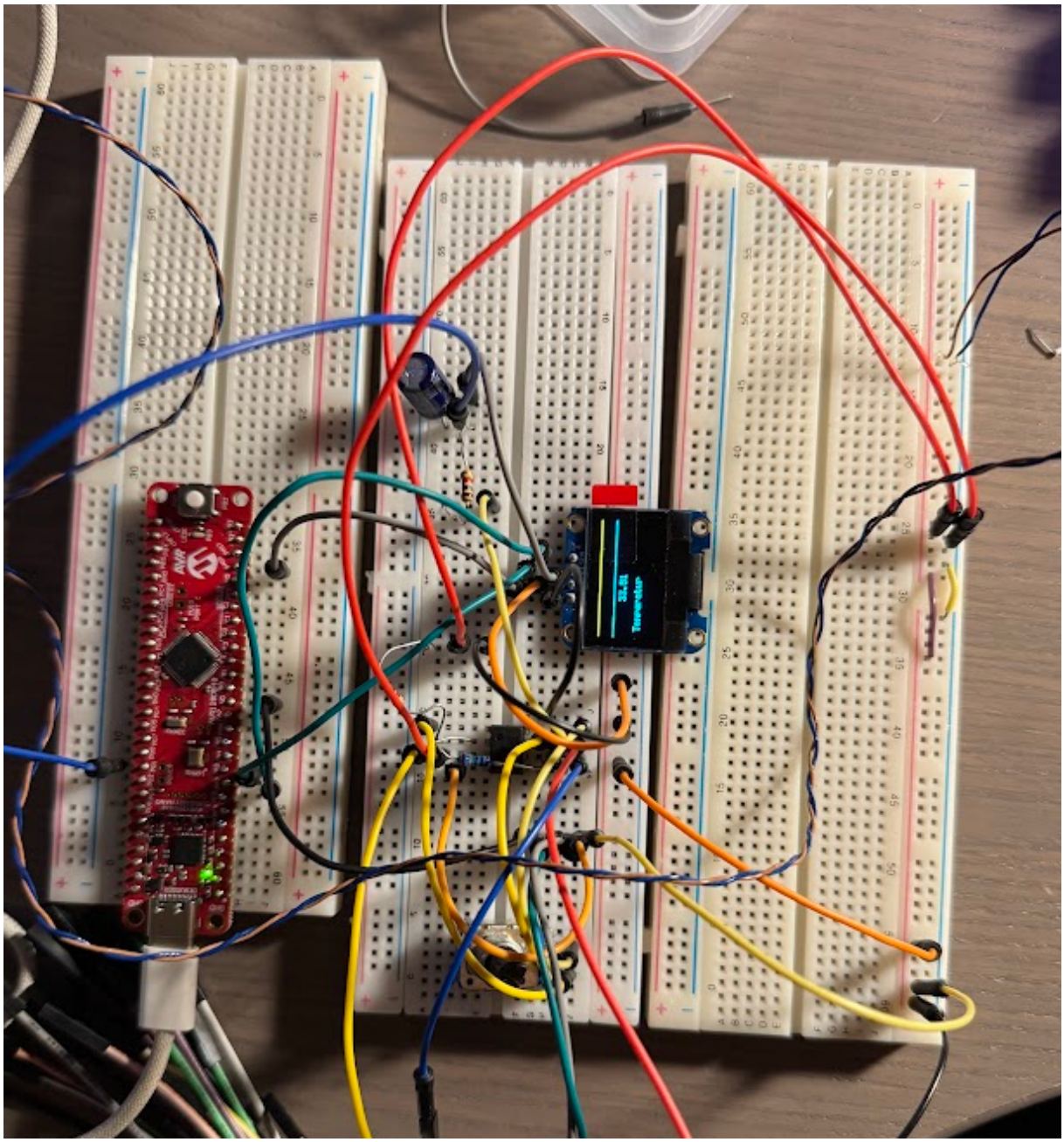
Utgangssignalet fra termoelementet ligger typisk i millivolt-området og er følsomt for støy. For å kunne bruke hele dynamikken til ADC-en, forsterkes og filtreres signalet slik at vi får et nytt signal v_o. ADC-en oversetter deretter v_o til digitale måleverdier m_n, som behandles videre i en estimator. Estimatoren justerer for offset, reduserer tilfeldig støy ved midling, og gir et estimert inngangssignal $\hat{v}_{in}$. Dette kan til slutt omgjøres til en estimert temperatur $\hat{T}$ som vises på skjermen.

Systemet kan deles inn i tre deler: en analog signalbehandlingsdel, en digital del for konvertering og estimering, samt en visningsdel. I seksjonene under beskrives disse nærmere.

—

2.1 Analog del

Den analoge delen sørger for å forsterke og filtrere signalet v_{in} slik at vi får et utgangssignal v_o som kan behandles av ADC-en. Kretsen er vist i figur 3:



Figur 3: Analog krets: ikke-inverterende forsterker med påfølgende RC-lavpassfilter som leverer signal v_o til ADC-en.

Her passerer inngangssignalet v_{in} først gjennom en ikke-inverterende forsterker, og deretter gjennom et lavpassfilter som demper høyfrekvent støy.

Forsterkningsfaktoren til forsterkeren er:

$$A = 1 + \frac{R_2}{R_1} \tag{1}$$

Før man bestemmer A er det lurt å definere temperaturområdet man ønsker å måle, slik at

signalet holder seg innenfor ADC-ens spenningsreferanse. Dette kan uttrykkes slik:

$$v[t_{max}] \cdot A \leq ADC_REF \tag{2}$$

$$A \leq \frac{ADC_REF}{v_{in}[t_{max}]} \tag{3}$$

Her er ADC_REF maksimal inngangsspenning for ADC-en, og $v_{in}[t_{max}]$ signalet ved høyeste temperatur man vil måle.

Lavpassfilteret har en knekkfrekvens gitt av:

$$f_0 = \frac{1}{2\pi R_3 C} \tag{4}$$

Siden temperatur endrer seg langsomt, er det tilstrekkelig at filteret kun slipper gjennom lave frekvenser, typisk under noen få Hz.

Det er også viktig å ta hensyn til instrumentets inngangsimpedans. Sammen med filtermotstanden kan dette danne en spenningsdeler og gi feil måling. For å minimere slike feil kan man bruke lavere motstandsverdier i filteret eller plassere en buffer før ADC-inngangen.

—

2.2 Digital del: ADC og estimator

I denne delen omformes det analoge signalet til digitale verdier ved hjelp av ADC, før signalet estimeres. Fremgangsmåten er inspirert av [2]. Kort fortalt skaleres rådata fra ADC til mikrovolt, det midles over N prøver for å redusere støy, offset trekkes fra, og resultatet korrigeres med kjent forsterkning A. Estimatet uttrykkes slik:

$$\hat{v}_{in}[n] = \frac{1}{A}\left(\frac{1}{N}\sum_{k=0}^{N-1}\hat{v}_o[n-k] - \hat{W}\right), \quad \hat{v}_o[n] = m[n] \cdot \frac{V_{REF}}{2^{ADC_RES}} \tag{5}$$

Her er $m[n]$ rå ADC-verdi, V_{REF} referansespenning i μV, ADC_RES oppløsning i bit, A analog forsterkning, $\hat{W}$ offset, og N antall prøver i midlingen. Mikrokontrolleren gjør utregningen, og resultatet konverteres til temperatur via et bibliotek.

2.2.1 Parametere og symboler

Tabell 1 gir en oversikt over symbolene brukt i formel 5:

Symbol	**Forklaring**	**Enhet**	**Verdi**
V_{REF}	Referansespenning	µV	målt
ADC_RES	ADC-oppløsning	bit	-
A	Analog forsterkning	-	målt/beregnet
$\hat{W}$	Offset i kjeden	μV	kalibrert
N	Antall prøver i midling	-	valgt

Tabell 1: Parametere og symboler brukt i estimeringsmodellen

2.2.2 Kalibrering

For at formel 5 skal gi riktige resultater, må systemet kalibreres. Fremgangsmåten kan være:

- **Mål V_{REF}:** Les av mikrokontrollerens forsyningsspenning og bruk denne verdien i beregningen.
- **Bestem offset $\hat{W}$:** Sett inngangssignalet til 0 µV (eller en kjent verdi) og mål avviket etter midling.
- **Kontroller A:** Gi inn et kjent signal v_{in} og mål v_o på utgangen. Beregn $A = \frac{v_o}{v_{in}}$ og sammenlign med forventet verdi fra (1).

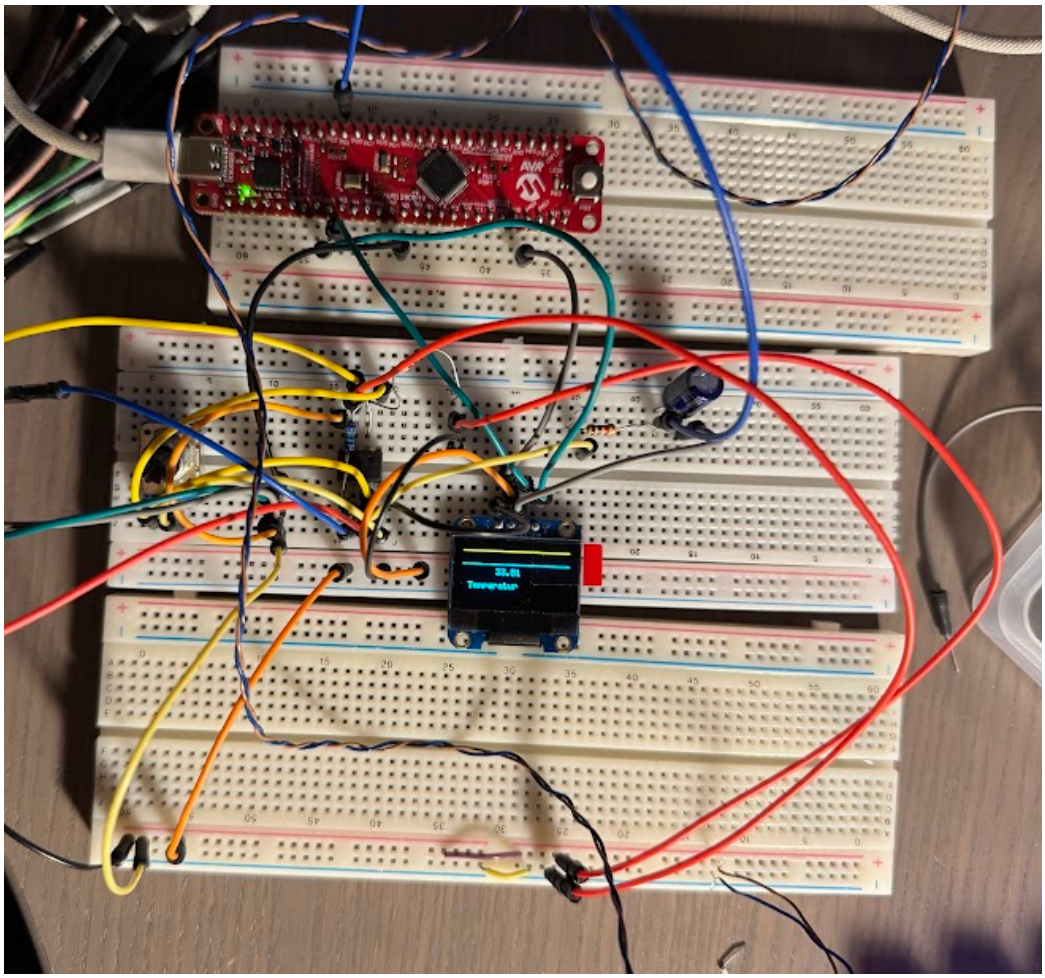
—

2.3 Digital del: Display

Når temperaturverdien $\hat{T}$ er beregnet, skal den vises på en skjerm. I dette prosjektet brukes `I2C`-protokollen for å sende data til OLED-displayet. Et ferdig bibliotek håndterer kommunikasjonen. Biblioteket krever at temperaturverdien sendes som tekststreng, og i C kan dette enkelt gjøres ved å bruke funksjonen `sprintf`.

3 Realisering og test

Prototypen ble bygget opp med operasjonsforsterkeren OP07CP [3], mikrokontrolleren `AVR128DB48`, og et OLED-display av typen SSD1306. I tillegg ble det benyttet nødvendige motstander, kondensatorer og koblingskomponenter. Figur 4 viser det ferdige oppsettet med termoelement, forsterker, mikrokontroller og display:



Figur 4: Bygget prototype bestående av termoelement, OP07-forsterker, AVR128DB48 og SSD1306 OLED-display.

3.1 Analog del

Verdiene for A, f_0, R_3 og C ble hentet ut fra beregningene i seksjon 2.1.

3.1.1 Bestemmelse av forsterkning A

Forsterkningsfaktoren A velges i henhold til formel 3. Referansespenningen for ADC ble målt til 3.29 V. Under testing var det ønsket å begrense måleområdet til maksimalt 80 °C (da vannkoker ikke var tilgjengelig). Ifølge [4] gir 80 °C en inngangsspenning $v_{in}[80] = 3.267$ mV. Dermed ble ønsket forsterkning beregnet til:

$$A = \frac{3.29}{3.267 \cdot 10^{-3}} \approx 1007$$

Basert på formel 1 ble $R_2 = 1$ MΩ og $R_1 = 1$ kΩ valgt. Faktiske måleverdier var $R_2 = 1.002$ MΩ og $R_1 = 985$ Ω, som gir:

$$A = \frac{1.006 \cdot 10^6}{985} \approx 1027$$

3.1.2 Lavpassfilter

Som vist i figur 3 ble utgangen v_{o_1} filtrert av et RC-lavpassfilter for å undertrykke høyfrekvent støy. Filterparametrene ble valgt ut fra formel 4, med krav om at knekkfrekvensen f_0 skulle være under noen få Hz. Etter praktiske tester ble følgende verdier valgt:

$$R_3 = 1\ \text{k}\Omega, \quad C = 100\ \mu\text{F}$$

Målt motstandsverdi for R_3 var 986 Ω, noe som gir:

$$f_0 = \frac{1}{2\pi \cdot 986 \cdot 100 \cdot 10^{-6}}$$
$$f_0 = 1.61\ \text{Hz}$$

Dette sikrer effektiv fjerning av høyfrekvent støy samtidig som temperatursignalet ivaretas.

3.1.3 Oppsummerte komponentverdier

Valgte og målte verdier i den analoge delen er listet i tabell 2:

Symbol	Verdi	Enhet
A	1027	-
R_1	985	Ω
R_2	$1.002 \cdot 10^6$	Ω
R_3	986	Ω
C	$100 \cdot 10^{-6}$	F
f_0	1.61	Hz

Tabell 2: Oppsummering av verdier brukt i den analoge delen.

3.2 Digital del: ADC og estimering av $\hat{v}_{in}$

ADC-en ble satt opp i 10-bit modus på pin `PD1` med VDD som referansespenning. Målt VDD var 3.29 V, tilsvarende $V_{REF} = 3\,290\,000$ µV. For å redusere støy ble det benyttet midling over $N = 100$ prøver, hver med 10 ms intervall. Dermed ble en ny estimert verdi generert ca. hvert andre sekund.

Signalet ble deretter justert for systematiske feil. Offset-verdier ble bestemt til 40 300 µV fra ADC og -45 000 µV fra operasjonsforsterkeren. Etter dette ble signalet skalert tilbake ved å dele på forsterkningsfaktoren $A = 1027$, slik at man fikk et estimat for v_{in}. Da ADC ble brukt i single-ended modus, kan systemet kun måle temperaturer høyere enn referansepunktet.

3.3 Konvertering til temperatur og visning

Det estimerte signalet $\hat{v}_{in}$ ble omgjort til temperatur ved hjelp av funksjonen `thermocoupleMvToC`, som bygger på standardkurven for K-type termoelement [5]. En kompensasjon ble lagt til ved å inkludere $T_{ref} = 8°\text{C}$ (målt i isvann). På denne måten ble en korrigert temperatur $\hat{T}$ beregnet.

Temperaturresultatene ble sendt til OLED-displayet (SSD1306) [6] via I2C-grensesnittet. Databladet til `AVR128DB48` [7] ble brukt for å konfigurere pinneoppsettet: `SDA = PA2` og `SCL = PA3`.

Et bibliotek håndterte kommunikasjonen, mens `sprintf`-funksjonen i C ble brukt for å konvertere temperaturverdien til en streng:

```
    char temperatur_buffer[16];
    sprintf(temperatur_buffer, "%.2f", current_temp_comp);
    SSD1306_DrawString(temperatur_buffer);
```

Kodeutdrag 1: Konvertering av flyttall til streng

3.4 Oppsummering av digitale variabler

Tabell 3 gir oversikt over sentrale variabler og parametere brukt i den digitale implementasjonen:

Variabel/Symbol	Verdi	Enhet
V_{REF}	3290000	µV
N	100	-
ADC_RES	10	bit
T_{ref}	8	°C
ADC_OFFSET	40 300	µV
$OUTPUT_OFFSET$	-45 000	µV
ADC_PIN	PD1	-
SDA	PA2	-
SCL	PA3	-

Tabell 3: Digitale parametere og variabler.

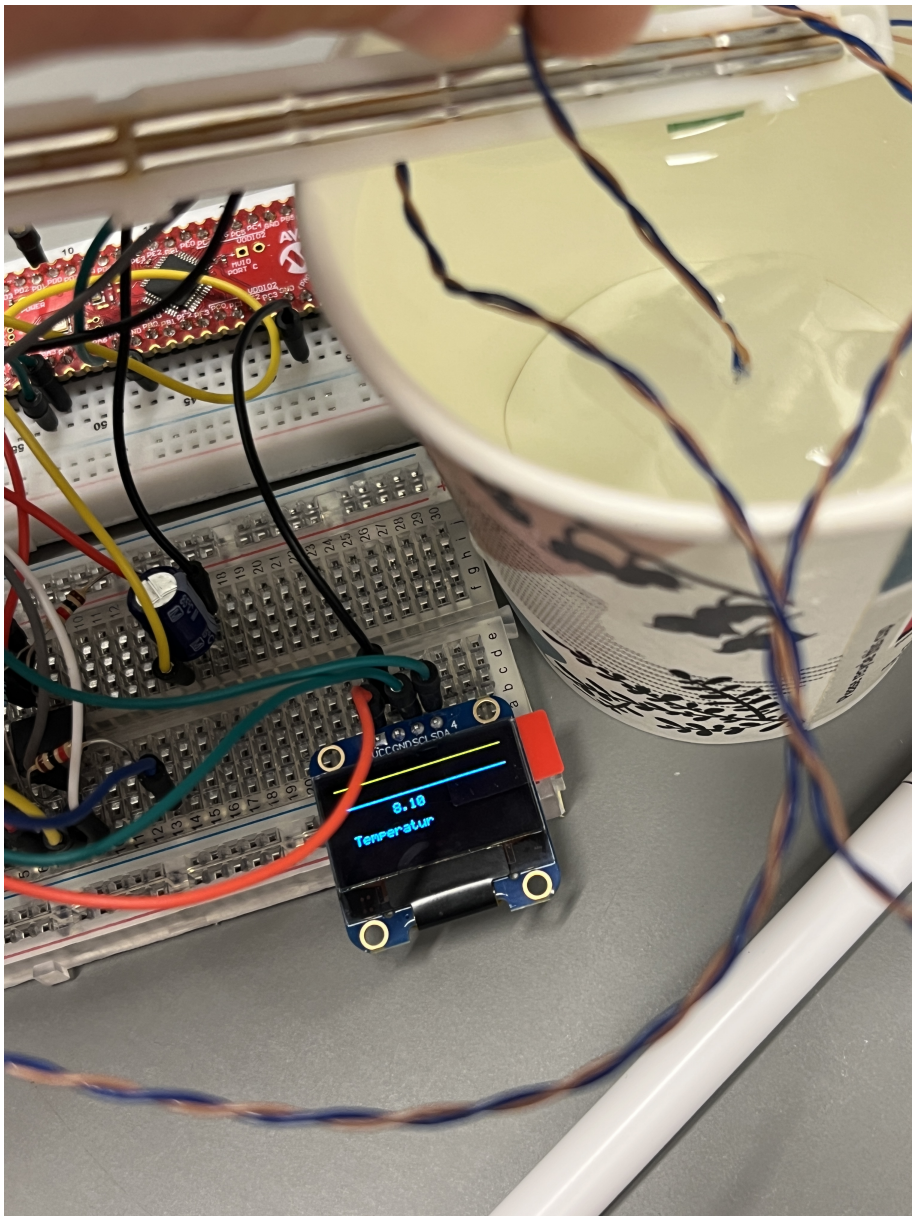
3.5 Resultater

Nøyaktigheten i målingene er begrenset, da en kalibrert referansesensor ikke var tilgjengelig. Derfor ble estimerte temperaturpunkter brukt som sammenligningsgrunnlag. Et annet viktig moment er at systemet ikke kan måle temperaturer lavere enn valgt referanse, noe som setter en begrensning på bruksområdet.

Systemet ble testet ved tre temperaturpunkter: isvann (8.00 °C), romtemperatur (22.00 °C), og lunkent vann (37.50 °C).

3.5.1 Referansetemperatur

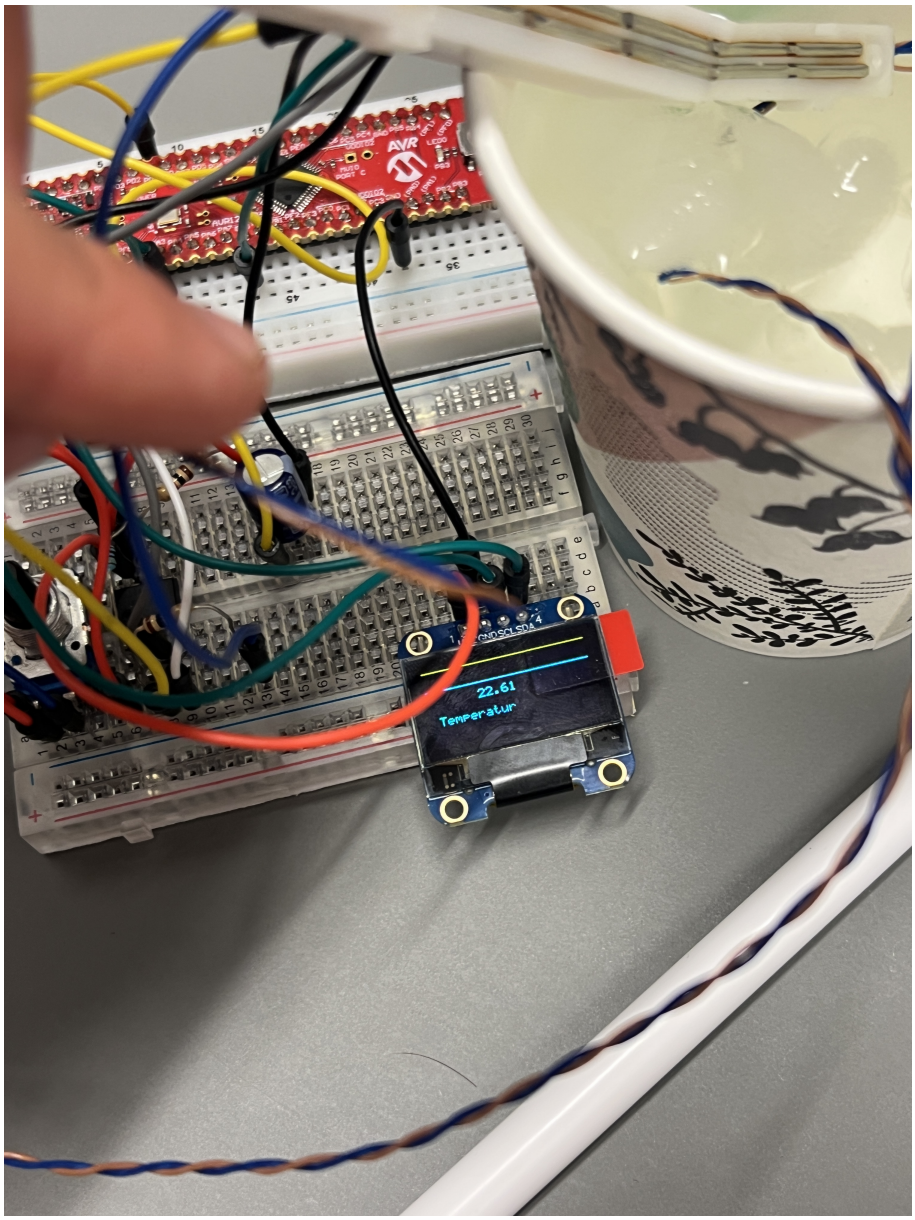
Målepunkt T_x ble lagt i en kopp med isvann sammen med referanseblokken T_{ref}. Forventet temperatur var 8.00 °C, og systemet rapporterte 8.10 °C, et avvik på 0.10 °C.



Figur 5: Måling ved referansetemperatur (isvann).

3.5.2 Romtemperatur

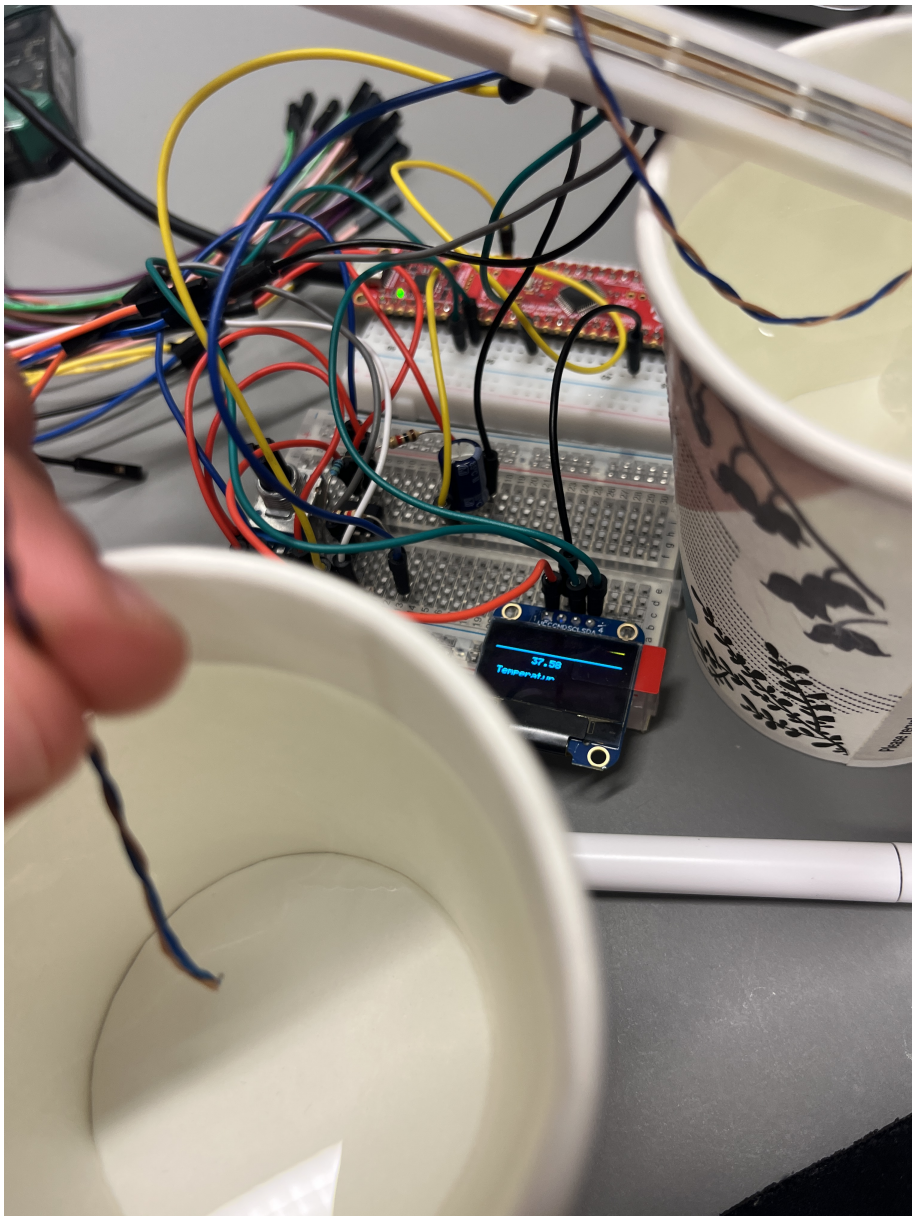
Romluften ble målt til 22.61 °C, mens estimert referanse var 22.00 °C. Dette tilsvarer et avvik på 0.61 °C.



Figur 6: Måling av romtemperatur.

3.5.3 Lunkent vann

Målepunktet ble plassert i en kopp med lunkent vann, forventet til 37.50 °C. Systemet viste 37.59 °C, et avvik på 0.09 °C.



Figur 7: Måling i lunkent vann.

3.6 Oppsummering av måleverdier

Tabell 4 viser resultatene fra testene:

Testpunkt	Forventet	Målt	Avvik	Enhet
Isvann	8.00	8.10	0.10	°C
Romtemperatur	22.00	22.61	0.61	°C
Lunkent vann	37.50	37.59	0.09	°C

Tabell 4: Oppsummering av måleresultater.

Målingene viser at systemet gir stabile og konsistente resultater, med avvik på under 1 °C i området 0–40 °C.

4 Konklusjon

I dette prosjektet ble det utviklet en fungerende prototype på et digitalt termometer basert på et K-type termoelement, en analog forsterkerkrets, mikrokontrolleren AVR128DB48 og et OLED-display. Hele målekjeden ble realisert fra sensoren til visning av temperatur, inkludert forsterkning, filtrering, digitalisering og programvare for estimering.

Resultatene fra testingen viste at systemet er i stand til å måle temperaturer med et avvik under 1°C i området 0–40 °C. Dette ligger innenfor kravene satt til nøyaktighet, og indikerer at løsningen er både stabil og presis i dette temperaturområdet. Begrensningene i systemet er først og fremst knyttet til at ADC-en ble brukt i single-ended modus, noe som gjør at temperaturer lavere enn referansen ikke kan registreres. Videre kan avvikene reduseres ytterligere ved mer nøyaktig kalibrering og bruk av en kjent referansesensor.

Alt i alt demonstrerer prosjektet hvordan enkle, standard elektroniske komponenter kan settes sammen til en helhetlig løsning som leverer på de gitte kravene. Prototypen viser at et digitalt termometer kan realiseres med relativt lav kompleksitet, og gir et godt utgangspunkt for videre arbeid med å utvide temperaturområdet, forbedre kalibreringen og gjøre systemet mer robust for praktisk bruk.

5 Takk

Takk til Mathias Ekse som har bidratt til et godt samarbeid for å løse dette designprosjektet. I tillegg har jeg brukt ChatGPT for forklaringer av begreper, tilbakemeldinger og forbedringspotensialer i teksten min.

Referanseliste

[1] L. Lundheim, *Ein temperaturtransdusar*, versjon 2, internt notat, NTNU, 2025.

[2] L. Lundheim, *Eit digitalt termometer*, internt notat, NTNU, 2025.

[3] Texas Instruments, *OP07: Ultra-Low Offset Voltage Operational Amplifier*, Datasheet, Rev. J, 2015. [Online]. Tilgjengelig: `https://www.ti.com/lit/ds/symlink/op07.pdf` Hentet 09.09.2025.

[4] Type K termoelement referansetabell, *Thermocouple Info*. [Online]. Tilgjengelig: `https://www.thermocoupleinfo.com/type-k-thermocouple.htm` Hentet 09.09.2025.

[5] Miceuz, *k-thermocouple-lib*, GitHub repository, 2016. [Online]. Tilgjengelig: `https://github.com/Miceuz/k-thermocouple-lib/tree/master` Hentet 09.09.2025.

[6] Solomon Systech, *SSD1306: 128 x 64 Dot Matrix OLED/PLED Segment/Common Driver with Controller*, Datasheet, Rev. 1.3, 2008. [Online]. Tilgjengelig: `https://cdn-shop.adafruit.com/datasheets/SSD1306.pdf` Hentet 09.09.2025.

[7] Microchip Technology Inc., *AVR128DB28/32/48/64 Data Sheet*, DS40002247A, 2021. [Online]. Tilgjengelig: `https://ww1.microchip.com/downloads/en/DeviceDoc/AVR128DB28-32-48-64-DataSheet-DS40002247A.pdf` Hentet 09.09.2025.

Vedlegg

A main.c

I kodevedlegg 2 ser vi hvordan systemet ble realisert ved å benytte oss av C-programmering.

```
#include <avr/io.h>
#include <util/delay.h>
#include <string.h>
#include <stdio.h>
#include "usart.h"
#include "thermocouple.h"
#include "ssd1306.h"
#define VOLTAGE_REFERENCE 3290000   // Maalt spenningsverdi for VDD paa AVR128DB48 i mikrovolt
#define MAX_SAMPLES 100             // For averaging
#define ANALOG_AMPLIFICATION 1027   // Utregnet forsterkning fra OP-AMP
#define RESOLUTION 10               // Opploesningen til ADC-en i bit
#define AMBIENT_TEMP 8              // Celsius
#define AVG_ADC_OFFSET_UV 40300     // Justeres basert paa egen maaling av ADC
#define AVG_OUTPUT_OFFSET_UV -45000 // Justeres til aa sette V_REF
/* Definition of the time required by VREF to start up after configuration (see the datasheet).*/
#define VREF_STARTUP_MICROS (50)

// Makro for aa regne ADC-verdi til mikrovolt (gitt at VOLTAGE_REFERENCE er i uV)
#define ADC_VAL_TO_UV ((float)VOLTAGE_REFERENCE)/((float)(1u << RESOLUTION))


/**
 * @brief Initializes the VREF peripheral with the desired reference voltages
 * for the DAC and the ADC.
 */
void VREF_init(void)
{
    /* Voltage reference at VDD */
    VREF.DACOREF |= VREF_REFSEL_VDD_gc;
    VREF.ADCOREF |= VREF_REFSEL_VDD_gc;

    VREF.DACOREF |= VREF_ALWAYSON_bm;
    VREF.ADCOREF |= VREF_ALWAYSON_bm;

    /* Wait VREF start-up time */
    _delay_us(VREF_STARTUP_MICROS);
}

/**
 * @brief Initializes the ADC on PIN D1 with appropriate prescaler.
 */
void ADC0_init(void)
{
    /* Disable digital input buffer */
    PORTD.PIN1CTRL &= ~PORT_ISC_gm;
    PORTD.PIN1CTRL |= PORT_ISC_INPUT_DISABLE_gc;

    /* Disable pull-up resistor */
    PORTD.PIN1CTRL &= ~PORT_PULLUPEN_bm;

    /* Set prescaler - adjust based on F_CPU for proper ADC clock */
    ADC0.CTRLC = ADC_PRESC_DIV4_gc; // CLK_PER divided by 4 for 4MHz
```

```
53
54      ADC0.CTRLA = ADC_ENABLE_bm          /* ADC Enable: enabled */
55      | ADC_RESSEL_10BIT_gc;              /* 10-bit mode */
56
57      /* Select ADC channel (PD1) */
58      ADC0.MUXPOS = ADC_MUXPOS_AIN1_gc; // PD1 on CNano
59  }
60
61  /**
62   * @brief Does a sample on the configured analog pin.
63   */
64  uint16_t ADC0_read(void)
65  {
66      /* Start ADC conversion */
67      ADC0.COMMAND = ADC_STCONV_bm;
68
69      /* Wait until ADC conversion done */
70      while (!(ADC0.INTFLAGS & ADC_RESRDY_bm))
71      {
72          ;
73      }
74
75      /* Clear the interrupt flag by writing 1: */
76      ADC0.INTFLAGS = ADC_RESRDY_bm;
77
78      return ADC0.RES;
79  }
80
81  int main(void)
82  {
83      USART3_init(9600);
84      VREF_init();
85      ADC0_init();
86      uint8_t retVal = SSD1306_Init(SSD1306_ADDR); // 0x3C
87      uint32_t adc_val = 0;
88      float current_uv = 0;
89      uint8_t samples = 0;
90      float accumulated_sum = 0;
91      float avg_voltage_uv = 0;
92      float avg_voltage = 0;
93      float avg_voltage_mv = 0;
94
95      float current_temp = 0;
96      float current_temp_comp = 0;
97
98      printf("--Starter termometer-- \n");
99
100     while (1) {
101         char temperatur_buffer[16];
102
103
104         adc_val = ADC0_read();
105         current_uv = adc_val * ADC_VAL_TO_UV;
106         accumulated_sum += current_uv;
107         samples++;
108
109         if (samples == MAX_SAMPLES) {
110
111             avg_voltage_uv = (float)((accumulated_sum/MAX_SAMPLES)-
    AVG_ADC_OFFSET_UV-AVG_OUTPUT_OFFSET_UV)/ANALOG_AMPLIFICATION;
112
```

```
113             avg_voltage = (float)(avg_voltage_uv / 1000000); // Enklere    lese
    for debugging
114             printf("Spenning i uV: %.0f uV, Spenning i V: %.6f \n",
    avg_voltage_uv, avg_voltage);
115 
116             current_temp = thermocoupleMvToC(avg_voltage_uv)/1000.0f;
117 
118             current_temp_comp = (float)(current_temp + AMBIENT_TEMP);
119             printf("Temperatur: %.2f, Temperatur med hensyn til
    omgivelsestemperatur: %.2f\n", current_temp, current_temp_comp);
120 
121             samples = 0;
122             accumulated_sum = 0;
123 
124             // Lagrer temperaturen i bufferen for display
125             sprintf(temperatur_buffer, "%.2f", current_temp_comp);
126              // DRAWING
127             //
    ------------------------------------------------------------------------------------

128             SSD1306_ClearScreen ();                                          //
    clear screen
129             SSD1306_DrawLine (0, MAX_X, 4, 4);               // draw line
130             SSD1306_SetPosition (7, 5);                              // set
    position
131             SSD1306_DrawString ("Temperatur");                    // draw string
132             SSD1306_DrawLine (0, MAX_X, 18, 18);             // draw line
133             SSD1306_SetPosition (40, 3);                             // set
    position
134             SSD1306_DrawString (temperatur_buffer);
      // draw string
135             SSD1306_UpdateScreen (SSD1306_ADDR);                  // update
136 
137             _delay_ms(1000);
138         }
139 
140         _delay_ms(10);
141 
142     }
143 }
```

Kodeutdrag 2: Main fil for programflyt på AVR128DB48