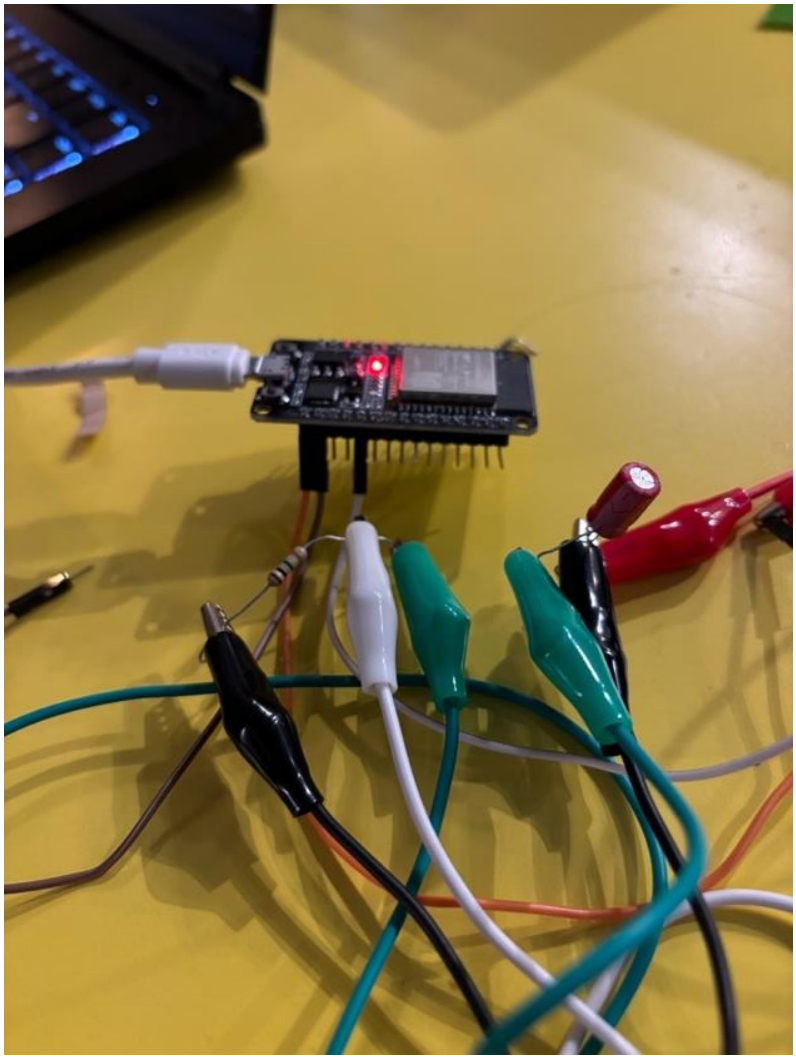
Oblig matematikk 1

Andreas Lindeman, Nicolai Lindeman, Laurits Hegbom

Måling av RC krets

Vi manglet noe av utstyret som var nødvendig. (Voltmeter, breadboard) Det stoppet ikke oss, og vi fant heller noe enda bedre til å måle mer nøyaktig. Vi brukte ESP-32 som spenningsforsyning og måler. Da kunne vi få målingene rett inn på PCen og enda flere målinger. Legger ved bildet av kretsen vår nederst.

Etter vi hadde kjørt målingene våres og fått plottet resultatet inn i python startet vi på utledningen av formelen for den ideele grafen. Utleda formelen på samme måte som vi har gjort 1000 ganger før. (Utledningen ligger ved lenger ned). Så la vi de to grafene inn i python og sammelignet de to grafene. Det stemmet overraskende likt ut.



$R = 1000000\,\Omega \qquad v(0) = 0$

$C = 100 \cdot 10^{-6}\,F$

$RC\dot{v}(t) + v(t) = 3.3 \quad \Big| \cdot \frac{1}{RC}$

$\dot{v}(t) + \frac{1}{RC}v(t) = \frac{3.3}{RC} \quad \Big| \cdot e^{\frac{t}{RC}}$

$\dot{v}(t) \cdot e^{\frac{t}{RC}} + \frac{e^{\frac{t}{RC}}}{RC}v(t) = \frac{3.3}{RC} \cdot e^{\frac{t}{RC}}$

$(v(t) \cdot e^{\frac{t}{RC}})' = \frac{3,3e^{\frac{t}{RC}}}{RC}$

$\int (v(t) \cdot e^{\frac{t}{RC}})' = \int \frac{3,3e^{\frac{t}{RC}}}{RC}$

$v(t) \cdot e^{\frac{t}{RC}} = \frac{3,3}{RC}\int e^{\frac{t}{RC}}$

$v(t) \cdot e^{\frac{t}{RC}} = \frac{3.3}{RC}(RCe^{\frac{t}{RC}} + C_1)$

$v(t) \cdot e^{\frac{t}{RC}} = 3,3e^{\frac{t}{RC}} + C_1 \quad \Big| \cdot e^{-\frac{t}{RC}}$

$v(t) = 3,3 + C_1 \cdot e^{\frac{-t}{RC}}$

$v(t) = 3,3 + C_1 \cdot e^{\frac{-t}{RC}} \qquad v(0) = 0$

$v(0) = 3,3 + C_1 \cdot e^{\frac{-0}{RC}}$

$0 = 3,3 + C_1$

$C_1 = -3,3$

$v(t) = 3,3 + C_1 \cdot e^{-\frac{t}{RC}}$

$v(t) = 3,3 - 3,3e^{-\frac{t}{RC}}$

Måling av data

Vi brukte en esp32 for å måle spenningen over kondensatoren.

Koden for dette så slik ut:

```
const int pin = 4;
int voltage = 0;


void setup() {
  Serial.begin(115200);
  pinMode(pin, INPUT);
}


void loop() {
  voltage = analogRead(pin);
  Serial.println(voltage);


}
```

Den sender konstant spenningen over spenningen til pcen over seriell porten. Vi kan ha et simpelt python script på pcen som mottar denne dataen og lagrer den.

```
import serial
import threading
import time
import pickle

# Configuration
SERIAL_PORT = "COM5"
BAUD_RATE = 115200
```

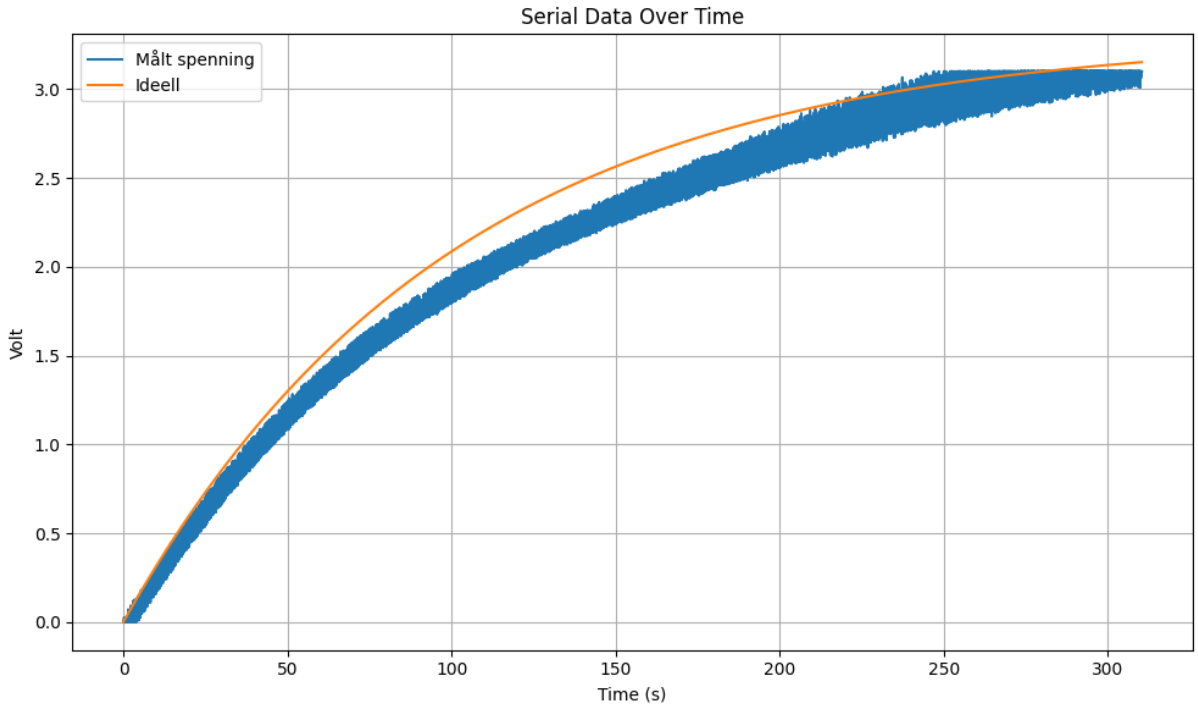
```python
PICKLE_FILE = "lades_opp_data.pkl"
SAVE_INTERVAL = 5  # seconds

# Shared data
logged_messages = []

# Function to read messages from the serial port
def log_serial_data():
    global logged_messages
    try:
        with serial.Serial(SERIAL_PORT, BAUD_RATE, timeout=1) as ser:
            while True:
                if ser.in_waiting > 0:
                    voltage = ser.readline().decode('utf-8').strip()
                    timestamp = time.time()
                    logged_messages.append((timestamp, voltage))
                    print(f"Logged: {timestamp} - {voltage}")
    except serial.SerialException as e:
        print(f"Serial error: {e}")

# Function to save the logged messages to a pickle file
def save_data_periodically():
    global logged_messages
    while True:
        time.sleep(SAVE_INTERVAL)
        with open(PICKLE_FILE, "wb") as f:
            pickle.dump(logged_messages, f)
        print(f"Data saved to {PICKLE_FILE}")

# Main function to start threads
def main():
    logging_thread = threading.Thread(target=log_serial_data, daemon=True)
    saving_thread = threading.Thread(target=save_data_periodically,
daemon=True)

    logging_thread.start()
    saving_thread.start()

    # Keep the main thread alive
    logging_thread.join()

if __name__ == "__main__":
    main()
```

Denne koden kjører også lagring av data parallelt med innsamling av data slik at innsamlingen ikke skal pauses under lagringen og skape støy i systemet. Vi kan så vise denne dataen ved å burke python.

```
import pickle
import numpy as np
import matplotlib.pyplot as plt

PICKLE_FILE = "lades_opp_data.pkl"

R = 1000000
C = 100e-6
V = 3.3

RC = R*C

def scale_data(volt, old_min=0, old_max=4095, new_min=0, new_max=3.1):
    return new_min + (volt - old_min) * (new_max - new_min) / (old_max -
old_min)


def plot_data():
    try:
        # Load data from pickle file
        with open(PICKLE_FILE, "rb") as f:
            logged_messages = pickle.load(f)

        if len(logged_messages) <= 1:
            print("Insufficient data to plot.")
            return

        # Ignore the first input (assumed to be noise) and extract remaining
data
        times, volts = zip(*logged_messages[1:])  # Skip the first entry
        times = np.array([float(t) for t in times])   # Convert to NumPy array
        volt = np.array([int(m) for m in volts])  # Convert to NumPy array

        volt =scale_data(volt)

        times = times-times[0]
        print(len(times)/times[-1])
        v = V - V * np.exp(-times / RC)

        # Plot the data
        plt.figure(figsize=(10, 6))
```

```
        plt.plot(times, volt, linestyle='-', label="Målt spenning")
        plt.plot(times, v, label="Ideell")
        plt.xlabel("Time (s)")
        plt.ylabel("Volt")
        plt.title("Serial Data Over Time")
        plt.grid(True)
        plt.legend()
        plt.tight_layout()
        plt.show()

    except FileNotFoundError:
        print(f"No pickle file found at {PICKLE_FILE}. Please ensure data has
been logged.")
    except ValueError as e:
        print(f"Data conversion error: {e}")

if __name__ == "__main__":
    plot_data()
```

Denne python filen genererer følgene graf:



Her har vi den målte spenningen i blå og den ideelle spenningen i orange.