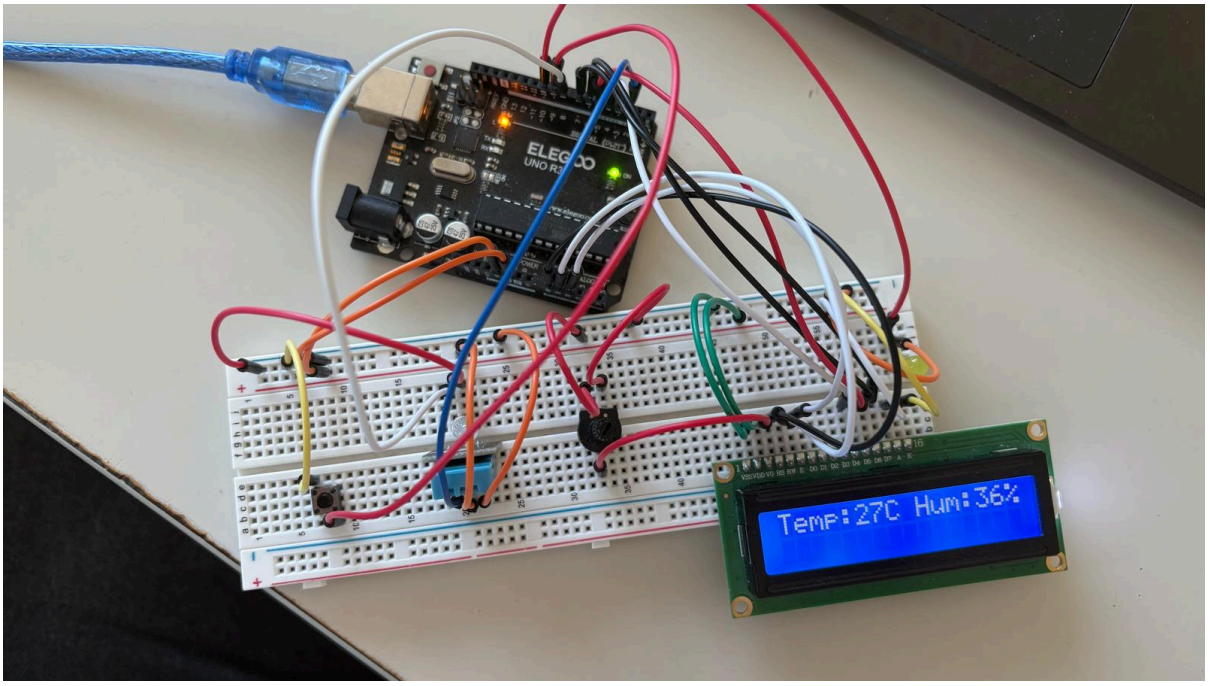
Protokoll - Messomat

Protokoll - Messomat

- **Abgabedatum:** 21.03.2025
- **Klasse:** 5CHIT
- **Gruppe:** 1
- **Leitung:** Mag. Dipl.-Ing.(FH) Brunner Markus
- **Autor:** Greiner Moritz



1. Projektübersicht

In diesem Projekt haben wir einen Messomaten realisiert, der mit einem DHT-11 Temperatursensor Temperatur- und Feuchtigkeitswerte erfasst. Die ermittelten Daten werden über drei Wege dargestellt:

- **LCD-Display:** Lokale Anzeige der Messwerte.
- **UART-Schnittstelle:** Serielle Übertragung der Messwerte.
- **Web-Visualisierung:** Darstellung in einer Blazor-Applikation.

Zusätzlich ermöglicht das System die Steuerung eines Lüfters per Kommando.

---

2. Hardware & Software Aufbau

Hardware:

- Arduino
- DHT-11 Sensor
- LCD-Display
- Lüfter
- Verbindung über UART zur Datenübertragung

Software:

- **Arduino-Programm (main.c):** Erfasst Messwerte, steuert das Display, sendet Daten über UART und regelt den Lüfter.
- **Blazor-Applikation:** Visualisiert die Messwerte und ermöglicht Bedienfunktionen wie Lüftersteuerung, ACK und Neustart.

---

3. Wichtige Codeausschnitte

3.1. Kernlogik im Arduino-Programm (main.c)

```
void check_ack()
{
    char received = uart_getc();
    if(received == ACK) {
        messagesSentWithoutAck = 0;
    }
    else if(received == 'r') {
        sendingAborted = false;
        messagesSentWithoutAck = 0;
        PORTB &= ~(1 << PORTB0);
    }
    else if(received == 'd') {
        sendingAborted = false;
        PORTB &= ~(1 << PORTB0);
    }
    else if(received == 'q') {
        sendingAborted = true;
        lcd_puts("ME gestoppt");
    }
    else if(received == 'e')
    {
        statusFAN = true;
        //Lüfter ein
        PORTB |= (1<<PORTB2);
        lcd_gotoxy(0,2);
        lcd_puts("Lüfter ein");
    }
    else if(received == 'a')
```

```
    {
        statusFAN = false;
        PORTB &= ~(1<<PORTB2);
        lcd_gotoxy(0,2);
        lcd_puts("Lüfter aus");
    }
    else if(received == 's')
    {
        uart_puts("\r\n");
        if(statusFAN == false)
        {
            uart_putc('f2');
        }
        else if(statusFAN == true)
        {
            uart_putc('f1');
        }
    }

}
```

Der Codeausschnitt verarbeitet eingehende UART-Daten: ACKs setzen den Nachrichten-Zähler zurück, 'r' und 'd' setzen den Messvorgang fort, 'q' stoppt ihn, 'e' und 'a' schalten den Lüfter ein bzw. aus, und 's' sendet je nach Lüfterstatus ein Statuszeichen..

3.2. Interrupt-Handler zur Steuerung und Kommunikation

```
ISR(TIMER1_COMPA_vect) {
    check_ack();

    if (!sendingAborted && resumeNormalSend) {
        if (messagesSentWithoutAck < 3)
        {
            send_message();
        }
        else
        {
            PORTB |= (1 << PORTB0);
            sendingAborted = true;
        }
    }
}
```

Hier wird dargestellt, wie durch Interrupts das Sendeintervall und die Kommunikation gesteuert werden.

Protokoll - Visualisierung

Dieses Bild dokumentiert die Webvisualisierung, die über die Blazor-Applikation realisiert wurde. Es zeigt die Darstellung der aktuellen Messwerte sowie die Bedienfelder zur Steuerung (Restart, ACK, Lüfter).

3.3. Blazor-Komponente (home.razor)

```
@page "/"
@using Visualisierung_BlazorApp.Services
@inject SerialPortService SerialPortService
@rendermode InteractiveServer

<div class="container my-5">
    <div class="card shadow-sm">
        <div class="card-body">
            <h3 class="card-title">SerialPort Test</h3>

            <hr />

            @if (!string.IsNullOrEmpty(sensorData))
            {
                <p><strong>Letzte Messung:</strong> @DateTime.Now</p>
                <p>@sensorData</p>
            }
            else
            {
```

```
                <p class="text-muted">Keine Daten empfangen...</p>
            }

            <div class="mt-4">
                <button class="btn btn-primary me-2"
@onclick="SendRestart">Restart</button>
                <button class="btn btn-success me-2" @onclick="SendAck">ACK
senden</button>
                <button class="btn btn-warning" @onclick="ToggleFan">
                    @(isFanOn ? "Lüfter ausschalten" : "Lüfter einschalten")
                </button>
            </div>
        </div>
    </div>
</div>

@code {
    private string? sensorData;
    private string receiveBuffer = "";
    private bool isFanOn = false;

    protected override void OnInitialized()
    {
        SerialPortService.OnDataReceived += SerialPortService_OnDataReceived;
    }

    private void SerialPortService_OnDataReceived(string data)
    {
        // Hänge die neuen Daten an den Puffer an
        receiveBuffer += data;

        // Suche im Puffer nach dem ETX-Zeichen (Ende der Nachricht, \x03)
        int etxIndex = receiveBuffer.IndexOf('\x03');
        if (etxIndex != -1)
        {
            // Extrahiere die komplette Nachricht (inklusive ETX)
            var completeMessage = receiveBuffer.Substring(0, etxIndex + 1);

            // Entferne STX (\x02), ETX (\x03) und überflüssige Steuerzeichen
            var cleanedData = completeMessage.Trim('\x02', '\x03', '\r', '\n');
            sensorData = cleanedData;

            // Entferne den verarbeiteten Teil aus dem Puffer
            receiveBuffer = receiveBuffer.Substring(etxIndex + 1);

            // UI aktualisieren
            InvokeAsync(StateHasChanged);
        }
    }
    private void SendAck()
```

```
    {
        SerialPortService.WriteData("\x06");
    }

    private void ToggleFan()
    {
        if (!isFanOn)
        {
            // Kommando zum Einschalten des Lüfters
            SerialPortService.WriteData("e");
            isFanOn = true;
        }
        else
        {
            // Kommando zum Ausschalten des Lüfters
            SerialPortService.WriteData("a");
            isFanOn = false;
        }
    }
    private void SendRestart()
    {
        // Kommando zum Neustart
        SerialPortService.WriteData("r");
    }
}
```

Dieser Auszug zeigt, wie die Blazor-Komponente serielle Daten empfängt, verarbeitet und Befehle an den Arduino sendet.

---

4. Fazit

Mit dem Messomatenprojekt haben wir erfolgreich eine Mehrwege-Datenübertragung realisiert. Die Kombination aus LCD-Anzeige, serieller Kommunikation und moderner Webvisualisierung (über Blazor) demonstriert, wie Hardware- und Softwarekomponenten effektiv integriert werden können. Die Möglichkeit, den Lüfter über das System zu steuern, unterstreicht den praktischen Einsatz der entwickelten Lösung.