

# Arduino Temperatursensor C\_sharp code

Deze code heb ik grotendeels met behulp van AI gemaakt. De eerdere iteraties waren wel volledig door mijzelf ontwikkeld, maar op een bepaald moment werkte het niet meer. Toen ben ik veel AI gaan gebruiken om fouten te vinden en op te lossen. Hierdoor kan ik dit werk niet als mijn eigen werk verantwoorden, omdat ik door de problemen niet consequent heb gedocumenteerd wat ik zelf heb gedaan.

Uiteindelijk bleek de fout niet in de code zelf te zitten, maar doordat twee applicaties tegelijkertijd de ttyUSB0-poort gebruikten.

```
using System;
using System.IO;
using System.IO.Ports;
using System.Text.Json;
using System.Threading;
using System.Threading.Tasks;

class Program
{
    static void Main()
    {
        string portName = "/dev/ttyUSB0";
        int baudRate = 9600;

        SerialPort serial = new SerialPort(portName, baudRate);
        serial.ReadTimeout = 2000;
        serial.WriteTimeout = 500;

        string csvBestand = "metingen.csv";

        // Header schrijven als bestand nog niet bestaat
        if (!File.Exists(csvBestand))
        {
            File.WriteAllText(csvBestand,
                "Datum;Tijdstip;Temperatuur;Luchtvochtigheid\n");
        }

        bool stoppen = false;

        if (serial != null)
        {
            serial.Open();
            if (serial.IsOpen)
            {
                Console.WriteLine("Verbinding met Arduino gemaakt: " +
```

```

portName);

Thread.Sleep(2000); // wacht even voor Arduino

while (!stoppen)
{
    // --- Handshake ---
    byte handshake = 0b00100100;
    serial.Write(new byte[] { handshake }, 0, 1);

    bool handshakeGelukt = false;
    int wachtTijd = 0;
    while (!handshakeGelukt && wachtTijd < 20)
    {
        if (serial.BytesToRead > 0)
        {
            string reactie = serial.ReadLine().Trim();
            Console.WriteLine("Arduino reageert: " +
reactie);

            if (reactie == "ACK")
                handshakeGelukt = true;
        }
        else
        {
            Thread.Sleep(100);
            wachtTijd++;
        }
    }

    if (!handshakeGelukt)
    {
        Console.WriteLine("Geen reactie op handshake!");
        break;
    }

    // --- Meetcommando ---
    byte command = 0b10100101;
    serial.Write(new byte[] { command }, 0, 1);
    Console.WriteLine("Meetcommando verstuurd...");

    Thread.Sleep(100);

    // --- Data uitlezen ---
    bool lezen = true;
    while (lezen)
    {
        if (serial.BytesToRead > 0)
        {
            string line = serial.ReadLine().Trim();
            if (!string.IsNullOrEmpty(line))
            {

```

```

        Console.WriteLine("Ontvangen: " + line);

        // JSON parsen
        try
        {
            var jsonDoc =
JsonDocument.Parse(line);

            double temperatuur =
jsonDoc.RootElement.GetProperty("temperatuur").GetDouble();
            double luchtvochtigheid =
jsonDoc.RootElement.GetProperty("luchtvochtigheid").GetDouble();

            // Datum en tijd
            string datum =
DateTime.Now.ToString("yyyy-MM-dd");
            string tijdstip =
DateTime.Now.ToString("HH:mm:ss");

            // CSV regel samenstellen
            string csvRegel = $"{datum};
{tijdstip};{temperatuur:F2};{luchtvochtigheid:F2}";

            // Schrijven naar CSV
            File.AppendAllText(csvBestand,
csvRegel + "\n");

            Console.WriteLine("Weergegeven in CSV:
" + csvRegel);
        }
        catch (Exception ex)
        {
            Console.WriteLine("Fout bij parsen
JSON: " + ex.Message);
        }
    }
    else
    {
        lezen = false;
    }
}

// --- Vraag gebruiker of stoppen --- Dit werkt nog
niet

        Console.WriteLine("Wil je het programma afsluiten?
(ja/nee) [10s timeout]");
        string antwoord = ReadLineWithTimeout(10000);
        if (antwoord != null && antwoord.Trim().ToLower() ==
"ja")
        {
            stoppen = true;

```

```

        Console.WriteLine("Programma wordt afgesloten.");
    }
    else
    {
        Console.WriteLine("Nog een meting wordt
gestart...");
    }

    Thread.Sleep(500);
}

serial.Close();
Console.WriteLine("Verbinding gesloten.");
}
}

static string ReadLineWithTimeout(int timeoutMillis)
{
    Task<string> task = Task.Run(() => Console.ReadLine());
    bool completed = task.Wait(timeoutMillis);
    return completed ? task.Result : null;
}
}

```