ELM327\_bluetooth\_program\_version\_3:

using PdfSharp.Drawing;

using PdfSharp.Fonts;

using PdfSharp.Pdf;

using System;

using System.IO;

using System.IO.Ports;

using System.Xml.Linq;

using static System.Net.Mime.MediaTypeNames;

class Program

{

static void Main()

{

GlobalFontSettings.UseWindowsFontsUnderWindows = true;

string[] ports = SerialPort.GetPortNames();

if (ports.Length == 0)

{

Console.WriteLine("Nincs elérhető COM port a gépen.");

Console.ReadKey();

return;

}

bool sikeresKapcsolodas = false;

string responseFile = "Response.txt";

string pdfFile = "Response.pdf";

foreach (string port in ports)

{

using (SerialPort serialPort = new SerialPort(port))

{

serialPort.BaudRate = 115200; //9600

serialPort.DataBits = 8;

serialPort.Parity = Parity.None;

serialPort.StopBits = StopBits.One;

serialPort.Handshake = Handshake.None;

serialPort.ReadTimeout = 5000; // 2 másodperc

serialPort.WriteTimeout = 5000;

serialPort.NewLine = "\r";

try

{

serialPort.Open();

Console.WriteLine($"Sikeresen kapcsolódtunk a {port} porthoz.");

Console.ReadKey();

sikeresKapcsolodas = true;

if (File.Exists("ELM327\_parancsok\_alkalmaz.txt"))

{

string[] atParancsok = File.ReadAllLines("ELM327\_parancsok\_alkalmaz.txt");

using (StreamWriter sw = new StreamWriter(responseFile, false))

{

foreach (string parancs in atParancsok)

{

if (string.IsNullOrWhiteSpace(parancs))

continue;

serialPort.Write(parancs + "\r"); // Explicit CR a parancs végén

Console.WriteLine($">>> Küldve: {parancs}");

//Console.ReadKey();

try

{

string valasz = serialPort.ReadLine();

Console.WriteLine($"<<< Válasz: {valasz}");

//

//Console.ReadKey();

// Válasz írása a fájlba

sw.WriteLine($"{parancs} -> {valasz}");

}

catch (TimeoutException)

{

Console.WriteLine("<<< Válasz timeout.");

Console.ReadKey();

sw.WriteLine($"{parancs} -> (timeout)");

}

System.Threading.Thread.Sleep(200); // Kis várakozás a következő parancs előtt

}

}

// PDF generálás a Response.txt tartalmából

CreatePdfFromTextFile(responseFile, pdfFile);

Console.WriteLine($"PDF riport létrehozva: {pdfFile}");

Console.ReadKey();

}

else

{

Console.WriteLine("Az 'ELM327\_parancsok\_alkalmaz.txt' fájl nem található.");

Console.ReadKey();

}

serialPort.Close();

break;

}

catch (Exception ex)

{

Console.WriteLine($"Nem sikerült kapcsolódni a {port} porthoz. Hiba: {ex.Message}");

Console.ReadKey();

}

}

}

if (!sikeresKapcsolodas)

{

Console.WriteLine("Nem sikerült egyetlen COM porthoz sem kapcsolódni.");

Console.ReadKey();

}

}

static void CreatePdfFromTextFile(string textFilePath, string pdfFilePath)

{

string[] lines = File.ReadAllLines(textFilePath);

PdfDocument document = new PdfDocument();

document.Info.Title = "Riport";

PdfPage page = document.AddPage();

XGraphics gfx = XGraphics.FromPdfPage(page);

XFont font = new XFont("Verdana", 12, XFontStyleEx.Regular);

double margin = 40;

double lineHeight = font.GetHeight();

double y = margin;

foreach (string line in lines)

{

// Ha az y (pozíció) plusz a sor magassága nagyobb, mint az oldal magassága mínusz a margó, új oldalt kezdünk.

if (y + lineHeight > page.Height.Point - margin)

{

// Új oldal hozzáadása, ha megtelik az előző

page = document.AddPage();

gfx = XGraphics.FromPdfPage(page);

y = margin;

}

gfx.DrawString(line, font, XBrushes.Black, new XRect(margin, y, page.Width.Point - 2 \* margin, lineHeight), XStringFormats.TopLeft);

y += lineHeight + 2;

}

document.Save(pdfFilePath);

}

}