HTTP GET ÜRÜNDEN RÖLE VE SICAKLIK DURUMU OKUMA

MİNİ SERVER ARAYÜZENDEN SICAKLIK VE RÖLE DURUMLARI SAYFALARINA **TEMP VE RELAY STATUS** BUTONLARINDAN ULAŞABİLİRSİNİZ, ÜRÜNE ATILAN İSTEKLER ARASINDA 250MS SÜRE OLMALIDIR.

DEFAULT IP İÇİN DURUM **REQUEST** LİNKLERİ:

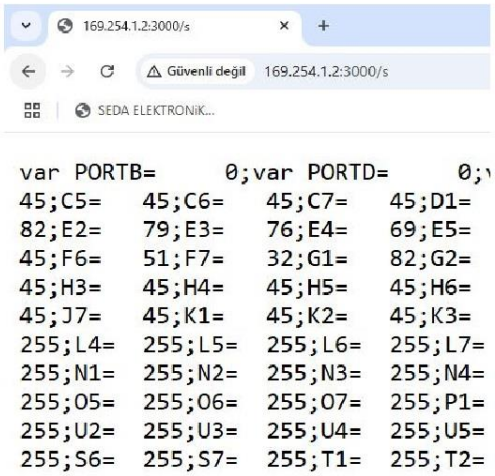
TEMP: http://169.254.1.2:3000/I

RELAY STATUS: http://169.254.1.2:3000/s

IP NUMARASINI DEĞİŞTİRDİĞİNİZDE DE DURUM SAYFALARI **IPNO:PORTNO/I** VEYA **IPNO:PORTNO/s** olacaktır. IP DEĞİŞTİRME PROGRAMINI BURADAN veya **BURADAN** İNDİREBİLİRSİNİZ.

RÖLE İSİMLERİ DEĞŞTİRİLEBİLİR.


```
var TEMP=    32;var FRACT=   81;Pos43Neg45=   43;NETWORK
IP:=  169 .  254 .    1 .    2 PORTNO: 3000  GATEWAY:=
169 .  254 .    1 .    1  SUBNET MASK:=  255 .  255 .
255 .    0    UYARI: IP GATEWAY SUBNET MASK DEĞİŞİKLİK
İŞLEMLERİNİZ BİTİNCE, CİHAZA RST KOMUTU GÖNDEREREK YENİ
CONFİG DEĞERLERİNİ DEVREYE ALABİLİRSİNİZ. ÖRNEK:
169.254.1.2:3000/RST
```



TEMP DEĞİŞKENİ ANA SICAKLIĞI VERİR, İHTİYAÇ OLURSA **FRACT** DEĞİŞKENİ İLE KÜSÜRATI DA OKUYABİLİRSİNİZ. **Pos43Neg45** DEĞİŞKENİ 43 İSE ISI POZİTİF, 45 İSE ISI NEGATİFTİR.

PORTB DEĞİŞKENİDE RÖLE DURUMLARINI DESİMAL OLARAK VERİR. DOKÜMAN SONUNDAKİ C# ÖRNEKLERİNİ YAZILIMINIZA IMPORT EDEBİLİRSİNİZ. **DESİMAL DEĞER, BİNARY DEĞERE** ÇEVRİLDİĞİNDE RÖLE DORUMLARI 1-0 OLARAK GÖZLEMLENMEKTEDİR.

4 IO İÇİN RÖLE ON GET REQUEST PARAMETRELERİ:

RÖLE-1 ON **IPNO:PORTNO/2** VEYA **IPNO:PORTNO/y1 (TOGGLE)**

RÖLE-2 ON **IPNO:PORTNO/3** VEYA **IPNO:PORTNO/y2 (TOGGLE)**

RÖLE-3 ON **IPNO:PORTNO/4** VEYA **IPNO:PORTNO/y3 (TOGGLE)**

RÖLE-4 ON **IPNO:PORTNO/5** VEYA **IPNO:PORTNO/y4 (TOGGLE)**

4 IO İÇİN RÖLE OFF PARAMETRELERİ:

RÖLE-1 ON **IPNO:PORTNO/g** VEYA **IPNO:PORTNO/y1 (TOGGLE)**

RÖLE-2 ON **IPNO:PORTNO/h** VEYA **IPNO:PORTNO/y2 (TOGGLE)**

RÖLE-3 ON **IPNO:PORTNO/j** VEYA **IPNO:PORTNO/y3 (TOGGLE)**

RÖLE-4 ON **IPNO:PORTNO/k** VEYA **IPNO:PORTNO/y4 (TOGGLE)**

C# ÖRNEKLERİ: (PORTB D A E DEĞİŞKENİ OKUMA)

Daha fazla örnek kod: https://sedaelektronik.com.tr/ethernet.html

1- **C# visual studio 2017-2019-2022 için 8 io röle kontrol-giriş kodları aşağıdaki linkten indirilebilir.**

(Net Frame Work 4.5.2.ve 4.6.2 sürümlerini derleme ve exe için gerektir.)

Exe yolu: c#-webbrowser-24-input\WindowsFormsApplication3\bin\Debug\WindowsFormsApplication3.exe

ver.1.2 : zip rar

https://drive.google.com/file/d/1BAjDm3hiCBzEsNawnRm1SXHg3Pf-RReh/view?usp=sharing

https://drive.google.com/file/d/1LZmk5ktGJ5maJeVMSwzIOcXok9nFtKwb/view?usp=sharing

ver.1.1:

https://drive.google.com/file/d/1fzLbRx3M_f1VYD6e2HzuZilFK2EM-cJn/view?usp=sharing

ver.1.0:

https://drive.google.com/file/d/1GQsHScZ4PgN-fOf1_1PQTUnLOAFgYHq5/view?usp=sharing

2- **C# ÖRNEK KOD İLE PORTB OKUMA:**

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
```

```
using System.Linq;
using System.Net.Http;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using VizoKantar.Models;
using VizoKantar.Services;
using System.IO;
using Microsoft.Data.Sqlite;
using LibVLCSharp.Shared;
using LibVLCSharp.WinForms;
using VizoKantar.Helpers;
using System.Diagnostics;
using Newtonsoft.Json.Linq;
using VizoKantar;
using System.Windows.Interop;

namespace VizoKantar
{
    public partial class OtomatikKantarControl : UserControl
    {

        private static readonly HttpClient _http = new HttpClient();
        private LibVLC _libvlc;
        private readonly List<MediaPlayer> _players = new();

        private async void OtomatikKantarControl_Load(object sender, EventArgs e)
        {
            // IP'yi trafik_config.json'dan oku
```

```
    if (_libvlc == null) StartCameras();   // yalnızca ilk gelişte
    _ = RefreshLightsAsync();              // ışık okuma

}

private async Task RefreshLightsAsync()
{
    int portb = await ReadPortBAsync(GetTrafikIp());
    SetLightByPortB(portb);
}

private void StartCameras()
{
    // 1) LibVLC (yalnızca bir kez)
    if (_libvlc == null)
    {
        Core.Initialize();
        _libvlc = new LibVLC(
        "--no-drop-late-frames",
        "--no-skip-frames",
        "--rtsp-tcp",
        "--network-caching=1000",           // 1000ms önerilir
        "--live-caching=1000",
        "--clock-jitter=0",                 // zaman sapması azaltılır
        "--clock-synchro=0",                // senkronizasyon zorlanmaz
        "--file-caching=300",
        "--http-reconnect",                 // yeniden bağlantı kur
        "--avcodec-fast",                   // düşük gecikme
        "--verbose=2",
        "--demux=h264",
        "--file-logging",
```

```
    "--logfile=vlclog.txt"


    );
}

// 2) Veritabanındaki URL'leri al → boşları at
var cfg = CameraConfigHelper.Load();
var urls = new[] { cfg.PlakaKamera, cfg.CevreKamera2, cfg.CevreKamera3, cfg.CevreKamera4 }
        .Where(u => !string.IsNullOrWhiteSpace(u))
        .ToList();
if (urls.Count == 0) return;   // hiç kamera yoksa çık

// 3) Daha önce eklediysek temizle
panelVideoArea.Controls.Clear();
_players.ForEach(p => p.Dispose());
_players.Clear();

// 4) Satır-sütun sayısını otomatik seç (kök-taban)
int n = urls.Count;
int rows = (int)Math.Ceiling(Math.Sqrt(n));
int cols = (int)Math.Ceiling(n / (double)rows);

var grid = new TableLayoutPanel
{
    Dock = DockStyle.Fill,
    RowCount = rows,
    ColumnCount = cols,
    BackColor = Color.Black,
    CellBorderStyle = TableLayoutPanelCellBorderStyle.Single
};
// Satır & sütunları eşit böl
```

```
        for (int r = 0; r < rows; r++) grid.RowStyles.Add(new RowStyle(SizeType.Percent, 100f / rows));
        for (int c = 0; c < cols; c++) grid.ColumnStyles.Add(new ColumnStyle(SizeType.Percent, 100f / cols));
        panelVideoArea.Controls.Add(grid);

        // 5) Her URL için VideoView oluştur ve başlat
        foreach (var url in urls)
        {
            var view = new VideoView { Dock = DockStyle.Fill };
            grid.Controls.Add(view);
            var mp = new MediaPlayer(_libvlc);
            view.MediaPlayer = mp;
            mp.Play(new Media(_libvlc, url, FromType.FromLocation));
            _players.Add(mp);
        }
    }


    private void SetLightByPortB(int portb)
    {
        // bit 1 açık mı (Kırmızı ışık)?
        bool isRedOn = (portb & (1 << 1)) != 0;

        if (isRedOn)
        {
            panelRedLight.BackColor = Color.DarkRed;
            panelGreenLight.BackColor = Color.Lime;
        }
        else
        {
```

```
        panelRedLight.BackColor = Color.Red;
        panelGreenLight.BackColor = Color.DarkGreen;

    }
}


private async Task<int> ReadPortBAsync(string ip)
{
    try
    {

        string txt = await _http.GetStringAsync($"http://{ip}:3000/s");
        var match = Regex.Match(txt, @"PORTB=\s*(\d+)");
        return match.Success ? int.Parse(match.Groups[1].Value) : -1;
    }
    catch
    {
        return -1;
    }
}

private string GetTrafikIp()
{
    string dbPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "vizokantar.db");
    string fallbackIp = "192.168.1.166";

    if (!File.Exists(dbPath))
        return fallbackIp;

    try
```

```
    {
        using (var conn = new SqliteConnection($"Data Source={dbPath}"))
        {
            conn.Open();
            string sql = "SELECT ip FROM trafik_isik WHERE id = 1";

            using (var cmd = new SqliteCommand(sql, conn))
            using (var reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    string ip = reader["ip"]?.ToString();
                    return !string.IsNullOrWhiteSpace(ip) ? ip : fallbackIp;
                }
            }
        }
    }
    catch { }

    return fallbackIp;
}

private readonly System.Windows.Forms.Timer _durumTimer =
new System.Windows.Forms.Timer { Interval = 3000 };
public OtomatikKantarControl()
{
    InitializeComponent();
    this.Load += OtomatikKantarControl_Load;
    panelRedLight.Click += panelRedLight_Click;
    panelGreenLight.Click += panelGreenLight_Click;
```

```
    // 1) Tartım olayına abone ol
    KantarPortService.WeightChanged += KgGeldi;

    // 3) Kontrol kapatılırsa aboneliği bırak
    this.Disposed += (_, __) => KantarPortService.WeightChanged -= KgGeldi;

}

// 2) Veri geldiğinde TextBox'a yaz
private List<(DateTime time, double kg)> _kgBuffer = new();
private bool _plakaTanimaBasladi = false;

private const int SABITLIK_SANIYE = 5;
private const double KG_TOLERANS = 150.0;

private void KgGeldi(string veri)
{
    if (InvokeRequired)
    {
        BeginInvoke((Action)(() => KgGeldi(veri)));
        return;
    }

    textBoxOtoTartim.Text = veri;

    if (!double.TryParse(veri, out double kg))
    {
        LogUyari("⛔ Geçersiz veri alındı. Tartım verisi çözümlenemedi.");
        return;
    }
```

```
DateTime now = DateTime.Now;

// 1. Yeni veriyi buffer'a ekle
_kgBuffer.Add((now, kg));

// 2. 5 saniyeden eski verileri temizle
_kgBuffer = _kgBuffer.Where(k => (now - k.time).TotalSeconds <= SABITLIK_SANIYE).ToList();

// 3. 5 saniyelik buffer dolduysa ve plaka tanıma başlamadıysa
if (!_plakaTanimaBasladi && _kgBuffer.Count >= 5)
{
    double minKg = _kgBuffer.Min(k => k.kg);
    double maxKg = _kgBuffer.Max(k => k.kg);
    double fark = maxKg - minKg;

    if (fark <= KG_TOLERANS)
    {
        double peakKg = maxKg;
        _plakaTanimaBasladi = true;
        LogUyari($"📸 Plaka algılanıyor (KG: {maxKg:N0})");

        LedYollaGuvenli("Plaka Algılanıyor");

        Task.Run(() => PlakaDenemesiYapVeGoster());



        //KaydetTartim(peakKg);
        //BaslatPlakaTanima();
    }
```

```
    }

    // 4. Sıfırlama – araç inerse
    if (kg < 150)
    {

        Task.Run(() => KirmiziIsikYakAsync());
        _plakaTanimaBasladi = false;
        _kgBuffer.Clear();



    }

    _durumTimer.Stop();
    _durumTimer.Start();
}



private async Task KirmiziIsikYakAsync()
{
    try
    {
        string ip = GetTrafikIp();
        await _http.GetAsync($"http://{ip}:3000/g");
        int portb = await ReadPortBAsync(ip);
        InvokeIfRequired(() => SetLightByPortB(portb));
        InvokeIfRequired(() => LogUyari("🚦 Araç çıktı, Yeni Araç Bekleniyor"));
    }
    catch (Exception ex)
    {
        InvokeIfRequired(() => LogUyari($"⚠ Kırmızı ışık hatası: {ex.Message}"));
```

```
    }
}
private void InvokeIfRequired(Action a)
{
    if (InvokeRequired)
        BeginInvoke(a);
    else
        a();
}

private async Task YesilIsikYakAsync()
{
    try
    {
        string ip = GetTrafikIp();
        await _http.GetAsync($"http://{ip}:3000/2");  // yeşil ışık
        int portb = await ReadPortBAsync(ip);
        InvokeIfRequired(() => SetLightByPortB(portb));
    }
    catch (Exception ex)
    {
        InvokeIfRequired(() => LogUyari("⚠ Yeşil ışık gönderimi başarısız: " + ex.Message));
    }
}

// === Asıl döngü ===
private void PlakaDenemesiYapVeGoster()
{
    double peakKg = _kgBuffer.Any() ? _kgBuffer.Max(k => k.kg) : 0;
    int denemeSayisi = 0;
```

```
while (denemeSayisi < 5)
{
    denemeSayisi++;

    try
    {
        string json = PlakaTanimaServisi.GetPlakaJson();
        string plaka = PlakaTanimaServisi.ExtractPlaka(json);

        if (string.IsNullOrWhiteSpace(plaka) || plaka == "0")
        {
            InvokeIfRequired(() => LogUyari("❗ Plaka tespit edilemedi, tekrar deneniyor..."));
            LedYollaGuvenli("Plaka algılanmadı. Bekleyin");
            Thread.Sleep(1500);
            continue;
        }

        LedYollaGuvenli($"Plaka algılandı: {plaka}");
        InvokeIfRequired(() => LogUyari($"📸 Plaka algılandı: {plaka}"));

        Thread.Sleep(3000);

        int sonuc = TartimServisi.PlakaVerisiIsle(plaka, peakKg);

        switch (sonuc)
        {
            case 0:
                LogUyari("❌ Sevkiyat bulunamadı.");
                LedYollaGuvenli($"Sevk Bulunamadı: {plaka}");
                Thread.Sleep(3000);
                continue;
```

```
            case 1:
                LogUyari("✅ 1. tartım kaydedildi.");
                LedYollaGuvenli("İşlem Başarılı");
                break;

            case 2:
                LogUyari("✅ 2. tartım güncellendi.");
                LedYollaGuvenli("İşlem Başarılı");
                break;

            case 4:
                LogUyari("❌ İlk tartım bulunamadı. 2. tartım yapılamaz.");
                LedYollaGuvenli("Giriş Tartımı Yok");
                Thread.Sleep(3000);
                continue;
        }

        Task.Run(() => YesilIsikYakAsync());
        return; // başarıyla bitti

    }
    catch (Exception ex)
    {
        LedYollaGuvenli("Sunucu veya Kamera Hatası");
        InvokeIfRequired(() => LogUyari($"⛔ Hata: {ex.Message}"));
        Thread.Sleep(1500);
    }
}
```

```
        InvokeIfRequired(() => LogUyari("⛔ Plaka tanıma denemeleri başarısız oldu. İşlem sonlandırıldı."));
        _plakaTanimaBasladi = false; // tekrar başlayabilsin
    }


    private void LedYollaGuvenli(string mesaj)
    {
        try
        {
            string ip = LedUyariGonderici.GetLedIp();

            if (!PingIleUlasilabilir(ip))
            {
                LogUyari($"⚠ LED tabela (IP: {ip}) yanıt vermiyor, mesaj ({mesaj}) atlanıyor.");
                return;
            }

            LedUyariGonderici.Gonder(mesaj);
        }
        catch (Exception ex)
        {
            LogUyari($"⚠ LED gönderim hatası: {ex.Message}");
        }
    }



    private bool PingIleUlasilabilir(string ip)
    {
```

```
    try
    {
        using (var ping = new System.Net.NetworkInformation.Ping())
        {
            var yanit = ping.Send(ip, 500); // 500 ms timeout
            return yanit.Status == System.Net.NetworkInformation.IPStatus.Success;
        }
    }
    catch
    {
        return false;
    }
}

private void UyariGoster(string mesaj)
{
    if (InvokeRequired)
    {
        Invoke(new Action(() => UyariGoster(mesaj)));
        return;
    }

    LogUyari(mesaj);
}
private void LogUyari(string mesaj)
{
    if (InvokeRequired)
    {
        Invoke(new Action(() => LogUyari(mesaj)));
        return;
    }
```

```
    // Tarih + saat eklemek istersen:
    string zamanliMesaj = $"[{DateTime.Now:HH:mm:ss}] {mesaj}";

    // Alt alta ekle
    textBoxUyariLog.AppendText(zamanliMesaj + Environment.NewLine);

}




private void BtnOtoCikis_Click(object sender, EventArgs e)
{
    var form = this.FindForm() as Form1;
    if (form != null)
    {
        form.panelMain.Controls.Clear();

        // İsteğe bağlı: Ana ekranı buraya geri ekle
        var secimEkrani = new KantarSecimControl(); // varsa
        secimEkrani.Dock = DockStyle.Fill;
        form.panelMain.Controls.Add(secimEkrani);
    }
}

private async void panelGreenLight_Click(object sender, EventArgs e)
{
    try
    {
        string ip = GetTrafikIp();
```

```
        await _http.GetAsync($"http://{ip}:3000/2");
        int portb = await ReadPortBAsync(ip);
        SetLightByPortB(portb);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Yeşil ışık komutu başarısız: " + ex.Message);
    }
}

private async void panelRedLight_Click(object sender, EventArgs e)
{
    try
    {
        string ip = GetTrafikIp();
        await _http.GetAsync($"http://{ip}:3000/g");
        int portb = await ReadPortBAsync(ip);
        SetLightByPortB(portb);
    }
    catch (Exception ex)
    {
        MessageBox.Show("Kırmızı ışık komutu başarısız: " + ex.Message);
    }
}



// Sınıf gövdenizin en altına veya ctor içinde:
protected override void OnResize(EventArgs e)
{
    base.OnResize(e);
    MakeCircle(panelRedLight);
```

```
            MakeCircle(panelGreenLight);
        }

        private void MakeCircle(Panel pnl)
        {
            var path = new GraphicsPath();
            path.AddEllipse(0, 0, pnl.Width - 1, pnl.Height - 1);
            pnl.Region = new Region(path);
        }
    }


}
```

```
//GetTrafikIp(); ile Trafik Lambasının ipsini alıyor

//PORTB okuma filtreleme:

private async Task<int> ReadPortBAsync(string ip)
{
    try
    {
        string txt = await _http.GetStringAsync($"http://{ip}:3000/s");
        var match = Regex.Match(txt, @"PORTB=\s*(\d+)");
        if (match.Success && int.TryParse(match.Groups[1].Value, out int PORTB))
        {
            if (PORTB >= 0 && PORTB <= 255)
                return portb;
```

```
        }
        return -1;
    }
    catch
    {
        return -1;
    }
}
```