Proje Raporu: Üretim Cihazları Verileri ile Anlık İzleme ve Analiz

Seymen Alper

August 21, 2024

Contents

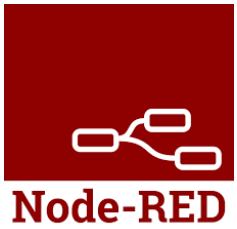
1 Giriş

Bu projede amacımız üretim cihazlarındaki sensörlerden gelen anlık verileri ayıklayıp, onları görselleştirerek anlamlı çıktılar elde etmek. Bunu yaparken algoritmayı oluşturabilmek adına Node-red, veritabanından gelen verileri alabilmek için MQTT explorer, bu verileri görselleştirebilmek için de Grafana aracını kullandık.

2 Kullanılan Teknolojiler

2.1 Node-RED

Node-RED, olay tabanlı bir geliştirme aracıdır. Akış tabanlı programlama modelini kullanarak IoT cihazlarının, API'lerin ve çevre birimlerin birbirine bağlanmasını sağlar.



2.2 MQTT

MQTT (Message Queuing Telemetry Transport), düşük bant genişliği ve düşük gecikme süresi gereksinimleri için optimize edilmiş bir mesajlaşma protokolüdür.

2.3 Grafana

Grafana, metrik verileri analiz etmek ve görselleştirmek için kullanılan bir platformdur. Verileri anlık olarak izlemek ve analiz etmek için güçlü bir araçtır.

3 Proje Adımları

3.1 Veri Alma ve İşleme

Üretim cihazlarından 'Mqtt in' node'u ile aldığımız anlık sensör verilerini (Noise, temperature, humidity) function node'larında 1JavaScript kodu yazarak ayıkladık ve değişkenlere atadık.

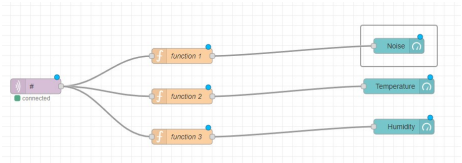


Figure 1: VeriAyiklanmasi

3.2 Verilerin Node-RED Dashboard'da Gösterilmesi

Node-RED Dashboard'da verileri görselleştirmek için 'gauge node' kullandık. Dashboard'da sensör verilerini aşağıdaki gibi2 gösterdik.

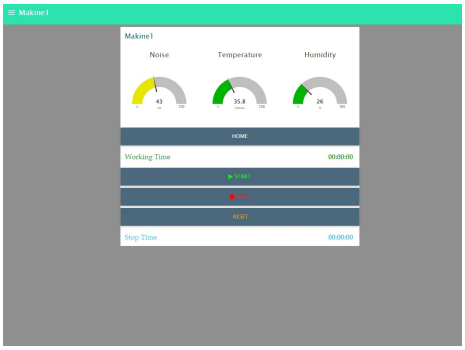


Figure 2: NodeRedDashboardEkrani

3.3 Çalışma ve Durma Sürelerini Ölçme

Cihazların çalışma ve durma sürelerini ölçmek için bir sistem tasarladık.3 Dashboard'a start, stop ve reset butonları ekleyip bu butonları iki farklı stopWatch node'una bağladık. Start'a bastığımızda Working Time, Stop'a bastığımızda Stop Time saymaya başladı. Reset butonuna basıldığında her iki sayaç da sıfırlanıyor.

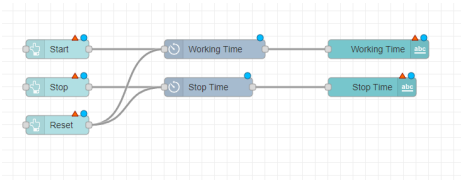


Figure 3: StartStopReset

3.4 Çalışma ve Durma Sürelerini Sıfırlama

Bir sonraki aşama olarak Start'a bastığımızda Stop Time'ın sıfırlanması, Stop'a bastığımızda Working Time'ın sıfırlanmasını, Reset'e bastığımızda da her ikisinin de sıfırlanmasını istedik. Bu nedenle Start ve Stop butonlarını ayrı bir Change düğümüne bağlayarak gelen mesajı revize ederek Timerlarımıza gönderdik.4

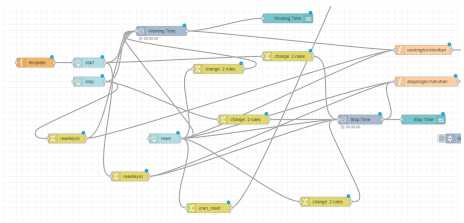


Figure 4: ChangeNodes

3.5 Oran hesaplanması

Çalışma verimliliğini hesaplayabilmek adına 'Working Time/(Working Time + Stop Time)' oranını bulmak istedik. Bunun için kullandığımız Verileri alan ve Mqtt'ye gönderen fonksiyonlarımıza yeni değişkenler atadık. Bu sayede Reset butonuna basana kadar Workin Time ve Stop Time verilerini TotalWorkingTime ve TotalStoppingTime değişkenlerinde tutarak reset öncesi sıfırlanmalardan etkilenmemesini sağladık. İlgili fonksiyonları aşağıda görebilirsiniz.

3.6 İlgili Fonksiyonlar

3.6.1 Working Time İçin Veri Alan Fonksiyon

Bu fonksiyon5, üretim cihazının çalıştığı süreyi kaydetmek için kullanılır. Cihazın çalışma süresi her saniye güncellenir ve fonksiyon bu süreyi bir değişkende saklayarak analiz edilebilir hale getirir.

```
// Mevcut süreyi depola
let lastDuration = context.get('lastDuration') || "00:00:00";
// Default olarak "00:00:00" ayarlandı

// Eğer gelen mesaj reset mesajı ise, süreyi sıfırla
if (msg.topic === 'reset' && msg.payload === true) {
    lastDuration = "00:00:00";
    context.set('lastDuration', lastDuration);
    flow.set('storedDurationWorking', lastDuration);
    return null;
}

// Süreyi güncelle ve sakla
if (msg.payload && msg.payload !== "00:00:00") {
    lastDuration = msg.payload;
    context.set('lastDuration', lastDuration);
    flow.set('storedDurationWorking', lastDuration);
} else {
    // Sayaç durduysa, 0 değeri olarak sakla
    flow.set('storedDurationWorking', "00:00:00");
}

// Veri döndürmeden durdur
return null;

```

Figure 5: WorkingVeriAlan

3.6.2 Stop Time İçin Veri Alan Fonksiyon

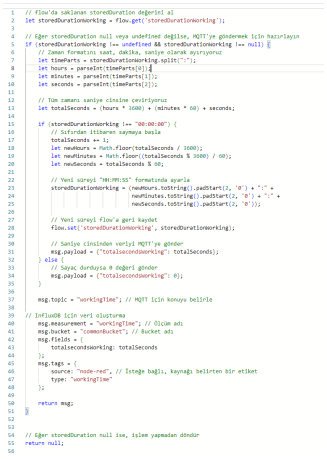
Bu fonksiyon6, cihazın durduğu süreyi kaydeder. Durma süresi her saniye güncellenir ve fonksiyon bu süreyi bir değişkende tutar. Bu sayede durma süreleri analiz edilebilir.

```
// Mevcut süreyi depola
let lastDuration = context.get('lastDuration') || "00:00:00";
// Default olarak "00:00:00" ayarlandı

// Eğer gelen mesaj reset mesajı ise, süreyi sıfırla
if (msg.topic === 'reset' && msg.payload === true) {
    lastDuration = "00:00:00";
    context.set('lastDuration', lastDuration);
    flow.set('storedDurationStopping', lastDuration);
    return null;
}

// Süreyi güncelle ve sakla
if (msg.payload && msg.payload !== "00:00:00") {
    lastDuration = msg.payload;
    context.set('lastDuration', lastDuration);
    flow.set('storedDurationStopping', lastDuration);
} else {
    // Sayaç durduysa, 0 değeri olarak sakla
    flow.set('storedDurationStopping', "00:00:00");
}

// Veri döndürmeden durdur
return null;

```

Figure 6: StoppingVeriAlan

3.6.3 Working Time değerini Mqtt'ye Gönderen Fonksiyon

Bu fonksiyon7, çalışma süresini MQTT aracılığıyla bir sunucuya gönderir. Fonksiyon çalışma süresini saniye cinsinden hesaplar ve bu veriyi Grafana gibi araçlarda görselleştirilmek üzere MQTT üzerinden iletir.

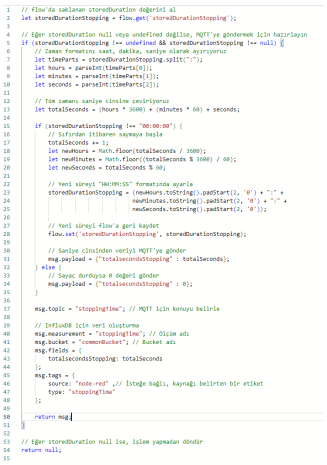


```
// Flow'da saklanan storedDuration değerini al
let storedDurationWorking = flow.get('storedDurationWorking');

// Eğer storedDuration null veya undefined değilse, MQTT'ye göndermek için hazırlayın
if (storedDurationWorking !== undefined && storedDurationWorking !== null) {
    // Zaman formatını saat, dakika, saniye olarak ayrıştırıyoruz
    let timeParts = storedDurationWorking.split(":");
    let hours = parseInt(timeParts[0]);
    let minutes = parseInt(timeParts[1]);
    let seconds = parseInt(timeParts[2]);

    // Tüm zamanı saniye cinsine çeviriyoruz
    let totalSeconds = (hours * 3600) + (minutes * 60) + seconds;

    if (storedDurationWorking !== "00:00:00") {
        // Sıfırdan itibaren saymaya başla
        totalSeconds += 1;
        let newHours = Math.floor(totalSeconds / 3600);
        let newMinutes = Math.floor((totalSeconds % 3600) / 60);
        let newSeconds = totalSeconds % 60;

        // Yeni süreyi "HH:MM:SS" formatında ayarla
        storedDurationWorking = (newHours.toString().padStart(2, '0') + ":" +
                                 newMinutes.toString().padStart(2, '0') + ":" +
                                 newSeconds.toString().padStart(2, '0'));

        // Yeni süreyi flow'a geri kaydet
        flow.set('storedDurationWorking', storedDurationWorking);

        // Saniye cinsinden veriyi MQTT'ye gönder
        msg.payload = {"totalsecondsWorking": totalSeconds};
    } else {
        // Sayaç durduysa 0 değerı gönder
        msg.payload = {"totalsecondsWorking": 0};
    }

    msg.topic = "workingTime"; // MQTT için konuyu belirle

// InfluxDB için veri oluşturma
    msg.measurement = "workingTime"; // Ölçüm adı
    msg.bucket = "commonBucket"; // Bucket adı
    msg.fields = {
        totalsecondsWorking: totalSeconds
    };
    msg.tags = {
        source: "node-red", // İsteğe bağlı, kaynağı belirten bir etiket
        type: "workingTime"
    };

    return msg;
}


// Eğer storedDuration null ise, işlem yapmadan döndür
return null;
```

Figure 7: WorkingToMqtt

3.6.4 Stopping Time Değerini Mqtt'ye Gönderen Fonksiyon

Bu fonksiyon8, durma süresini MQTT aracılığıyla sunucuya gönderir. Durma süresi, fonksiyon tarafından saniye cinsinden hesaplanır ve analiz edilmek üzere iletilir.



```
// Flow'da saklanan storedDuration değerini al
let storedDurationStopping = flow.get('storedDurationStopping');

// Eğer storedDuration null veya undefined değilse, MQTT'ye göndermek için hazırlayın
if (storedDurationStopping !== undefined && storedDurationStopping !== null) {
    // Zaman formatını saat, dakika, saniye olarak ayrıştırıyoruz
    let timeParts = storedDurationStopping.split(":");
    let hours = parseInt(timeParts[0]);
    let minutes = parseInt(timeParts[1]);
    let seconds = parseInt(timeParts[2]);

    // Tüm zamanı saniye cinsine çeviriyoruz
    let totalSeconds = (hours * 3600) + (minutes * 60) + seconds;

    if (storedDurationStopping !== "00:00:00") {
        // Sıfırdan itibaren saymaya başla
        totalSeconds += 1;
        let newHours = Math.floor(totalSeconds / 3600);
        let newMinutes = Math.floor((totalSeconds % 3600) / 60);
        let newSeconds = totalSeconds % 60;

        // Yeni süreyi "HH:MM:SS" formatında ayarla
        storedDurationStopping = (newHours.toString().padStart(2, '0') + ":" +
                                  newMinutes.toString().padStart(2, '0') + ":" +
                                  newSeconds.toString().padStart(2, '0'));

        // Yeni süreyi flow'a geri kaydet
        flow.set('storedDurationStopping', storedDurationStopping);

        // Saniye cinsinden veriyi MQTT'ye gönder
        msg.payload = {"totalsecondsStopping" : totalSeconds};
    } else {
        // Sayaç durduysa 0 değerı gönder
        msg.payload = {"totalsecondsStopping" : 0};
    }

    msg.topic = "stoppingTime"; // MQTT için konuyu belirle

    // InfluxDB için veri oluşturma
    msg.measurement = "stoppingTime"; // Ölçüm adı
    msg.bucket = "commonBucket"; // Bucket adı
    msg.fields = {
        totalsecondsStopping: totalSeconds
    };
    msg.tags = {
        source: "node-red" ,// İsteğe bağlı, kaynağı belirten bir etiket
        type: "stoppingTime"
    };

    return msg;
}

// Eğer storedDuration null ise, işlem yapmadan döndür
return null;
```

Figure 8: StoppingToMqtt

3.6.5 Oran Değerini Mqtt'ye Gönderen Fonksiyon

Bu fonksiyon9, çalışma ve durma sürelerinden hesaplanan verimlilik oranını sunucuya gönderir. Oran, cihazın verimliliğini ölçmek için kullanılır ve bu veri, analiz edilmek üzere sunucuya iletilir.

```
// Oran resetleme butonundan gelen mesajı kontrol edin
if (msg.payload === true && msg.topic === 'oran_reset') {
    // Oran resetleme işlemi yapılıyor
    flow.set('totalWorkingTime', 0);
    flow.set('totalStoppingTime', 0);
    msg.payload = {"workingRatio" : 0};
    msg.topic = "workingRatio"; // MQTT için konuyu belirle
    return msg;
}

// Birikmiş totalWorkingTime ve totalStoppingTime değerlerini al (saniye cinsinden)
let totalWorkingTime = flow.get('totalWorkingTime') || 0;
let totalStoppingTime = flow.get('totalStoppingTime') || 0;

// Şu anki workingTime ve stoppingTime değerlerini al
let storedDurationWorking = flow.get('storedDurationWorking') || "00:00:00";
let storedDurationStopping = flow.get('storedDurationStopping') || "00:00:00";

// workingTime'ı saniyeye çevir
let workingTimeParts = storedDurationWorking.split(":");
let workingTimeSeconds = (parseInt(workingTimeParts[0]) * 3600) +
                         (parseInt(workingTimeParts[1]) * 60) +
                         parseInt(workingTimeParts[2]);

// stoppingTime'ı saniyeye çevir
let stoppingTimeParts = storedDurationStopping.split(":");
let stoppingTimeSeconds = (parseInt(stoppingTimeParts[0]) * 3600) +
                          (parseInt(stoppingTimeParts[1]) * 60) +
                          parseInt(stoppingTimeParts[2]);

// Gelen değerleri totalWorkingTime ve totalStoppingTime'a ekle
totalWorkingTime += workingTimeSeconds;
totalStoppingTime += stoppingTimeSeconds;

// totalWorkingTime ve totalStoppingTime değerlerini flow'a kaydet
flow.set('totalWorkingTime', totalWorkingTime);
flow.set('totalStoppingTime', totalStoppingTime);

// Toplam süreyi hesapla
let totalTime = totalWorkingTime + totalStoppingTime;

// Oranı hesapla: totalWorkingTime / totalTime
let workingRatio = totalTime > 0 ? totalWorkingTime / totalTime : 0;


// Oranı MQTT'ye göndermek için hazırla
msg.payload = {"workingRatio" : workingRatio};
msg.topic = "workingRatio"; // MQTT için konuyu belirle

// InfluxDB için veri oluşturma
msg.measurement = "workingRatio"; // Ölçüm adı
msg.bucket = "commonBucket"; // Bucket adı
msg.fields = {
    value: workingRatio
};
msg.tags = {
    source: "node-red" ,// İsteğe bağlı, kaynağı belirten bir etiket
    type: "workingRatio"
};

return msg;

```

Figure 9: OranToMqtt

3.7 Node-red'deki Genel Yapı

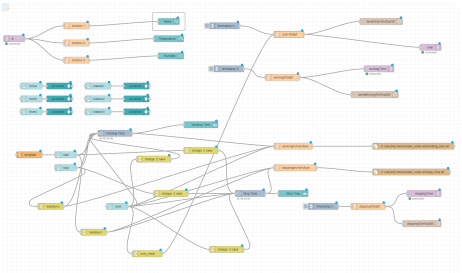


Figure 10: GenelYapi

3.8 Verilerin Grafana'ya Gönderilmesi ve Görselleştirilmesi

Node-RED'den gelen verileri MQTT ile Grafana'ya aktardık ve bu verileri Grafana'da görselleştirdik.11 Aşağıda Grafana'da oluşturduğumuz dashboard örneklerini bulabilirsiniz.

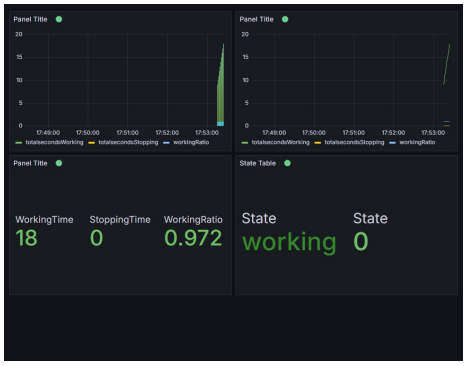


Figure 11: Grafana Dashboard Örneği 1

3.8.1 Dashboard Özelleştirilmesi

Dashboar'ın daha kullanışlı olmasına adına Makine1 ve Makine2 butonlarını ve butonların altında o makinenin verilerinin önizlemesini ekledik.12

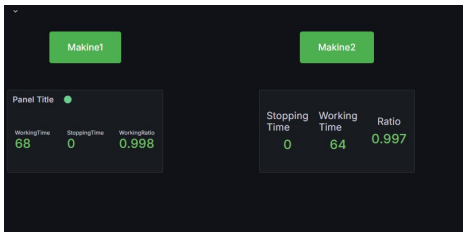


Figure 12: Grafana Dashboard Örneği 2

4 Sonuç

Bu proje ile üretim makinelerinin durumunu anlık olarak takip edebilir, çalışma ve durma sürelerini analiz ederek verimliliği ölçebiliriz. Grafana'da oluşturduğumuz dashboard ile bu verileri kolayca görselleştirdik ve izlenebilir hale getirdik.