Haftanın Görevi: Tasarım Odaklı Düşünme

Problem: Basit Sensörler Kullanılarak Gıda Deposu İçin Sıcaklık ve Nem Takip Sistemi

Haftanın Sorumlusu: Yusuf Burak Aslan

Elektrik-Elektronik Mühendisliği Perspektifi

1. Kullanıcıyı Anlama :

Gıda deposu personelinin ve yöneticisinin asıl derdi sadece sıcaklık kaç derece diye bakmak değil herhangi bir arıza veya elektrik kesintisi durumunda ürünlerin bozulma riskidir. Kullanıcı sürekli ekran başında durmak yerine sistemin kendisi adına bekçilik yapmasını ve bir sorun olduğunda anında Telegram veya E-posta üzerinden haberdar edilmeyi bekliyor.

2. Sorunu Belirleme:

Temel sorunumuz fiziksel ortamdaki değişimlerin manuel kontrollerle her an takip edilememesi ve donanım tarafında oluşabilecek hatalı verilerin sistemi yanıltma riskidir. EEM tarafı olarak bizim odaklandığımız sorun çevresel verinin en yüksek hassasiyetle toplanıp yazılım ekibinin belirttiği Edge katmanına kesintisiz ve temiz bir şekilde iletilmesini sağlamaktır.

3. Fikir Üretme:

Donanım Altyapısı: Prototip aşamasında Arduino Uno ve DHT22 sensörü kullanarak yüksek hassasiyetli bir veri toplama düğümü oluşturuyoruz.

Güç Yönetimi: Sistemin depo içerisinde her yere taşınabilmesi için pil beslemesi ve enerji tasarrufu sağlayan sleep modları kurguluyoruz.

Entegrasyon: Yazılım tarafının kurguladığı n8n bildirim sistemini ve dashboard'u beslemek için, verileri seri port üzerinden MQTT protokolüne uygun formatta JSON hazırlayıp köprü kuruyoruz.

Endüstri Mühendisliği Perspektifi

1. Kullanıcıyı Anlama

Endüstri mühendisliği açısından kullanıcı analizi, yalnızca sistemi kullanan bireylerin belirlenmesiyle sınırlı değildir. Sistemle etkileşimde bulunan tüm paydaşların rolleri, beklentileri ve süreç performansına etkileri birlikte değerlendirilmelidir. Bu yaklaşım, sistemin yalnızca teknik olarak değil, operasyonel açıdan da verimli çalışmasını sağlar.

Gıda deposu ortamında kullanıcıların temel ihtiyacı yalnızca sıcaklık ve nem değerlerini izlemek değildir. Asıl beklenti, arıza, elektrik kesintisi veya ani çevresel değişim gibi durumlarda ürün bozulma riskinin önceden tespit edilmesi ve zamanında müdahale edilmesidir. Kullanıcı, sürekli manuel kontrol yapmak yerine sistemin kendi adına izleme yapmasını ve kritik durumlarda otomatik olarak bilgilendirme sağlamasını beklemektedir.

Sistem kullanıcıları; depo operasyon personeli, depo yöneticisi, kalite kontrol birimi ve işletme yönetimi olarak sınıflandırılmaktadır. Bu kullanıcıların ortak beklentisi güvenilir veri,

hızlı müdahale ve karar destek mekanizmalarının sağlanmasıdır. Bu nedenle sistemin veri temelli, izlenebilir ve proaktif bir yapıya sahip olması gerekmektedir.

2. Sorunu Belirleme:

Temel problem, mevcut sistemlerde sıcaklık ve nem takibinin manuel veya düzensiz yapılması nedeniyle süreçlerin kontrol altında tutulamamasıdır. Bu durum gecikmiş müdahale, ürün bozulması ve maliyet artışı gibi sonuçlara yol açmaktadır.

Endüstri mühendisliği perspektifinden bakıldığında sorun sadece veri toplanması değil; bu verinin doğru yerde, doğru şekilde ve sürekli olarak elde edilmesidir. Ayrıca depo içindeki sıcaklık dağılımının homojen olmaması, sensörlerin rastgele konumlandırılması durumunda yanıltıcı sonuçlar üretme riskini ortaya çıkarır.

Buna ek olarak sistemin güvenilirliği de kritik bir problemdir. Sensör arızaları, veri kayıpları veya hatalı ölçümler karar mekanizmasını olumsuz etkileyebilir. Bu nedenle süreçlerin standardize edilmesi, risklerin önceden belirlenmesi ve sistemin sürdürülebilir şekilde çalışması gerekmektedir.

3. Fikir Üretme:

Süreç Tasarımı ve Optimizasyon:
Depo içindeki tüm süreçler analiz edilerek sıcaklık ve nem takibinin sistematik hale getirilmesi sağlanır. Sensör yerleşimi, depo içindeki kritik noktalar (kapılar, raflar, havalandırma bölgeleri) dikkate alınarak optimize edilir. Amaç minimum sensör ile maksimum kontrol sağlamaktır.

Veri Tabanlı Karar Mekanizması:
Toplanan veriler istatistiksel yöntemlerle analiz edilerek normal çalışma aralıkları belirlenir. İstatistiksel Süreç Kontrolü (SPC) kullanılarak anormal durumlar tespit edilir ve erken uyarı mekanizması desteklenir.

Risk ve Hata Yönetimi:
FMEA (Hata Türü ve Etkileri Analizi) uygulanarak sensör arızası, veri sapması ve sistem kesintileri gibi riskler önceden belirlenir. Bu sayede sistem daha güvenilir hale getirilir.

Maliyet–Fayda Yaklaşımı:
Kurulacak sistemin yatırım maliyeti ile sağlayacağı faydalar (ürün kaybının azalması, iş gücü tasarrufu, kalite artışı) analiz edilerek işletme açısından sürdürülebilirliği değerlendirilir.

Entegrasyon ve Süreç Akışı:
Sensörlerden gelen verinin yazılım sistemine aktarılması, analiz edilmesi ve kullanıcıya anlamlı çıktı olarak sunulması bir süreç akışı olarak modellenir. Bu sayede sistem bütüncül bir yapıya kavuşturulur.

Yazılım Mühendisliği Perspektifi

1.Kullanıcıyı Anlama:

Depo personelinin ve yöneticilerin teknik karmaşadan uzak, anında eyleme geçebilecekleri sade bir arayüze ihtiyacı var. Kullanıcılar sürekli gelen bildirimler yüzünden "yanlış alarm yorgunluğu" yaşamak istemiyor; bu yüzden sadece gerçekten kritik bir ihlal olduğunda (örneğin eşik üst üste 3 kez aşıldığında) uyarılmayı bekliyorlar. Ayrıca kalite kontrol süreçleri

ve gıda güvenliği denetimleri için geriye dönük verilerin hızlıca bulunabilmesi ve periyodik raporlar (PDF/Excel) sunulması kullanıcılar için kritik bir ihtiyaçtır.

2. Sorunu Belirleme:

Donanım ne kadar hassas olursa olsun, fiziksel ortamdan gelen sensör verileri bazen sistemsel olarak imkansız değerler (örneğin -127°C) içerebilir veya anlık sıçramalar yapabilir. Yazılım ekibi olarak bizim odaklandığımız temel sorun; donanımdan gelen bu ham veriyi (Edge katmanından) alıp veri temizleme sürecinden geçirmek, sistemi yormadan veritabanına işlemek ve son kullanıcıya gecikmesiz, eyleme dönüştürülebilir bir hizmet katmanı olarak sunmaktır.

3. Fikir Üretme:

• Veri İşleme ve Backend: EEM tarafının MQTT üzerinden gönderdiği standart JSON verilerini alıp, Python FastAPI ile asenkron olarak işleyerek PostgreSQL veritabanına zaman damgalı (timestamp) kaydedecek bir mimari kuruyoruz. Hatalı sensör verilerini "kayan ortalama filtresi" ile temizleyerek sistemin yanılmasını önlüyoruz.

• Kullanıcı Arayüzü (Frontend): Teknik olmayan personelin de rahatça kullanabileceği mobil uyumlu React.js + Chart.js tabanlı bir kontrol paneli (dashboard) tasarlıyoruz. Ekranda değerleri anlık ve gecikmesiz göstermek için WebSocket kullanacağız.

• Otomasyon ve Bildirim: Backend sistemi gerçek bir ihlal tespit ettiğinde, n8n otomasyon aracı üzerinden webhook tetikleyerek Telegram ve E-posta bildirimlerini otomatik olarak yöneticilere iletiyoruz.

4. Kullanıcı Senaryosu (User Journey):

Örnek Bir Kriz Anı Akışı Sistemin son kullanıcıya ve işletmeye anlık olarak nasıl dokunduğunu göstermek adına tasarladığımız örnek vaka:

• Olay (Tetikleyici): Et ve süt ürünleri bölümünde soğutucu motor arızalanır ve sıcaklık aniden +4°C kritik eşiğinin üzerine çıkmaya başlar.

• Algılama (EEM Katmanı): DHT22 sensörü bu sıcaklık artışını saniyeler içinde algılar. ESP32 kontrolcüsü, uyku modundan (sleep mode) uyanarak veriyi MQTT protokolü üzerinden anında sisteme iletir.

• Karar ve Bildirim (Yazılım Katmanı): Backend katmanı (FastAPI) gelen veriyi alır. Bunun geçici bir sensör hatası olmadığını doğrulamak için eşiğin 3 kez üst üste aşıldığını teyit eder (yanlış alarm filtresi). Sorun kesinleştiği an n8n otomasyonu üzerinden depo amirinin Telegram hesabına ve E-postasına "Kritik Uyarı: A Blok Soğutucu Arızası! Sıcaklık: +5.2°C" mesajı düşer. Eş zamanlı olarak React Dashboard ekranında ilgili bölge canlı olarak kırmızı renkle vurgulanır.

• Aksiyon ve İyileştirme (Endüstri Katmanı): Depo amiri saniyeler içinde haberdar olduğu için ürünler bozulmadan müdahale eder ve ekonomik kayıp önlenir. Süreç sonunda ise sistemin oluşturduğu geçmişe dönük veriler ve loglar incelenerek arızanın kök nedeni değerlendirilir, depo operasyonları veri temelli olarak iyileştirilir.