

İLERİ NESNE TABANLI PROGRAMLAMA DERSİ

PROJE RAPORU: AKILLI SERA OTOMASYONU

Proje Adı: Akıllı Sera (Tarım) Otomasyonu

Geliştirme: C# Windows Forms, SQLite

Tarih: Aralık 2024

1. PROJE TANITIMI

1.1 Proje Konusu

Akıllı Sera Otomasyonu, bitki türlerinin gereksinimlerine göre sensör verilerini okuyarak aktüatörleri otomatik kontrol eden bir sistemdir.

1.2 Modül Yapısı

- Modül 1:** Bitki ve Bölge Yönetimi (BitkiTuru, SeraBolgesi)
- Modül 2:** Sensör Yönetimi (Sensor abstract, SıcaklikSensoru, NemSensoru, IsikSensoru)
- Modül 3:** Aktüatör Yönetimi (Aktuator abstract, SulamaVanasi, Isitici, Fan, Lamba)
- Modül 4:** Kontrol ve Otomasyon (SeraKontroloru, ZamanlanmisGorev)

Bitki Türü Yönetimi

Domates (Su: 2, Işık: 0, Nem: 0)
Salatalık (Su: 2, Işık: 30, Nem: 20)
Döşem (Su: 2, Işık: 0, Nem: 0)

Bitki Adı:

Salatalık

4

Ara

Su:

2,00

Ekle

Güncelle

Işık:

50,00

Sil

Nem:

20,00

Sensör Yönetimi

Sensör: Aktif: True, Sıcaklık: 25,3275878767804°C
Sensör: Aktif: True, Nem: %58,7
Sensör: Aktif: True, Işık: 87,37982596590112 lux

Sensör Adı:

Tip:

Işık

Sensör Ekle

Kontrol ve Otomasyon

Kontrol Et

Toplam Kontrol: 8

Sera Bölgesi Yönetimi

Yeşillik - sıcak alan - bölge a
soğuk kapalı alan - bölge b

Bölge Adı:

Konum:

Ekle

Aktüatör Yönetimi

Aktüatör: ff, Aktif: False, Su Miktarı: 0L

Aktüatör Adı:

Tip:

Fan

Aktüatör Ekle

2. C# ÖZELLİKLERİNİN KULLANIMI

2.1 Interface Inheritance

```
// Interface 1 → Interface 3 miras ilişkisi
public interface Interface1 {
    void Ekle(object item);
    List<object> Listele();
    object Ara(int id);
    void Guncelle(int id, object item);
    void Sil(int id);
}

public interface Interface3 : Interface1 {
    string DetayliBilgiGetir() {
        return "Interface 3 - Interface 1'den miras alıyor";
    }
}
```

Kullanım: BitkiTuruRepository : Interface3 → Interface 1 metotlarını da içerir

2.2 Abstract Class ve Method Overriding

```
public abstract class Sensor {
    public abstract double DegerOku();
    public virtual string BilgiGetir() {
        return $"Sensor: {_ad}, Aktif: {_aktif}";
    }
}

public class SicaklikSensoru : Sensor {
    public override double DegerOku() {
        Random rnd = new Random();
        return rnd.Next(15, 35) + rnd.NextDouble();
    }
    public override string BilgiGetir() {
        return base.BilgiGetir() + $", Sıcaklık: {_sicaklik}°C";
    }
}
```

2.3 Constructor ve Method Overloading

```
public class BitkiTuru {
    public BitkiTuru() { }
    public BitkiTuru(string ad, double su, double isik, double nem) { }
    public BitkiTuru(int id, string ad, double su, double isik, double
nem) { }
}

public class SeraKontroloru {
    public void SensorEkle(Sensor sensor) { }
    public void SensorEkle(string ad, int seraBolgesiId, string tip) { }
}
```

2.4 Custom Exception ve Exception Handling

```
public class SeraException : Exception {
    public string? HataKodu { get; set; }
    public SeraException(string message, string hataKodu) : base(message)
{
    HataKodu = hataKodu;
}
}

try {
    _bitkiTuruRepo.Ekle(bitki);
} catch (SeraException ex) {
    MessageBox.Show($"Hata: {ex.Message} (Kod: {ex.HataKodu})");
}
```

2.5 Generic Collections ve LINQ

```
private List<Sensor> _sensorler;
private Dictionary<int, BitkiTuru> _bitkiTurleri;

var bolgeSensorleri = _sensorler
    .Where(s => s.SeraBolgesiId == seraBolgesiId)
    .ToList();
```

2.6 Static Methods ve Nested Class

```
public class SeraKontroloru {  
    private static int _toplamKontrolSayisi = 0;  
    public static int ToplamKontrolSayisi() {  
        return _toplamKontrolSayisi;  
    }  
  
    public class KontrolSonucu {  
        public bool Basarili { get; set; }  
        public string Mesaj { get; set; } = string.Empty;  
    }  
}
```

2.7 System.IO Kullanımı

```
if (!File.Exists(_dbPath)) { /* Veritabanı kontrolü */ }  
  
using (StreamWriter writer = new StreamWriter(_logDosyasi, append: true))  
{  
    writer.WriteLine(logMesaji);  
}
```

3. UYGULAMA ADIMLARI

3.1 Bitki Türü Ekleme

1. "Bitki Adı" alanına ad girilir (örn: "Domates")
2. Su, Işık, Nem gereksinimleri girilir
3. "Ekle" butonuna tıklanır

Bitki Türü Yönetimi

Domates (Su: 2, Işık: 0, Nem: 0)
Salatalık (Su: 2, Işık: 50, Nem: 20)
Dompğ (Su: 2, Işık: 0, Nem: 0)

Bitki Adı: Salatalık 4 Ara

Su: 2,00 Ekle Güncelle

Işık: 50,00

Nem: 20,00 Sil

Kod:

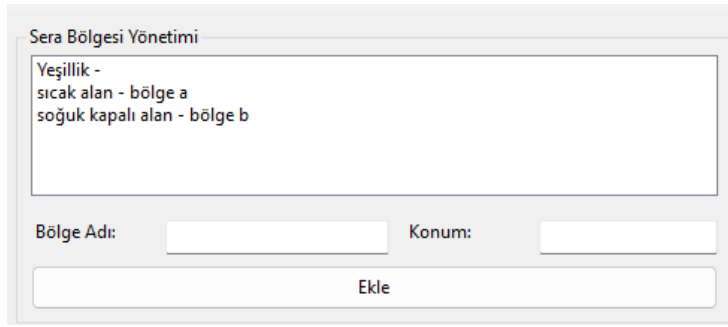
```
private void btnBitkiEkle_Click(object sender, EventArgs e) {  
    BitkiTuru bitki = new BitkiTuru(txtBitkiAd.Text, su, ısı, nem);  
    _bitkiTuruRepo.Ekle(bitki);  
    MessageBox.Show("Bitki türü eklendi!");  
}
```

3.2 Bitki Türü Listeleme ve Arama

Eklenen bitki türleri listede görüntülenir. ID ile arama yapılabilir.

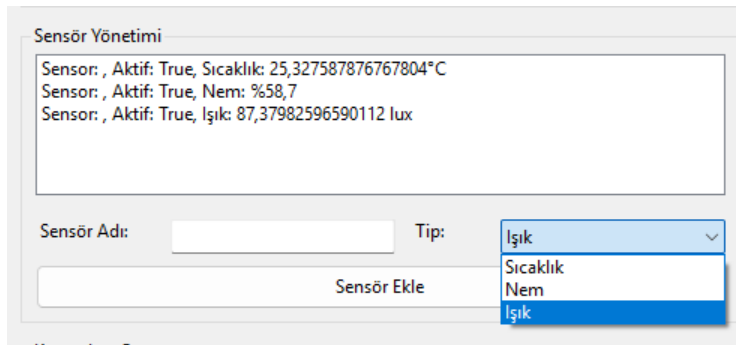
3.3 Sera Bölgesi Ekleme

1. Bitki türü seçilir
2. Bölge adı ve konum girilir
3. "Ekle" butonuna tıklanır



3.4 Sensör Ekleme

1. Sera bölgesi seçilir
2. Sensör adı ve tipi seçilir (Sıcaklık, Nem, Işık)
3. "Sensör Ekle" butonuna tıklanır

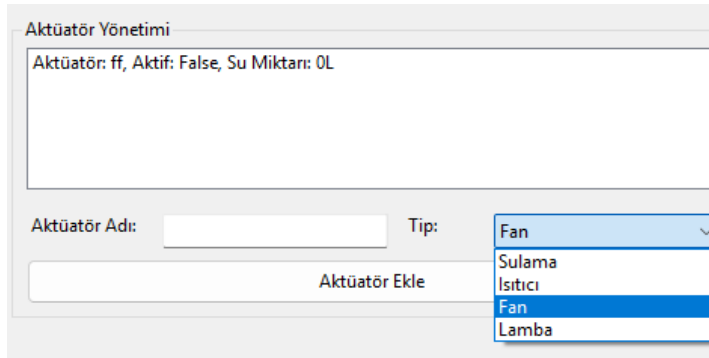


Kod:

```
public void SensorEkle(string ad, int seraBolgesiId, string tip) {
    Sensor? sensor = null;
    switch (tip) {
        case "Sıcaklık": sensor = new SicaklikSensoru(ad, seraBolgesiId);
        break;
        case "Nem": sensor = new NemSensoru(ad, seraBolgesiId); break;
        case "Işık": sensor = new IsikSensoru(ad, seraBolgesiId); break;
    }
    if (sensor != null) SensorEkle(sensor);
}
```

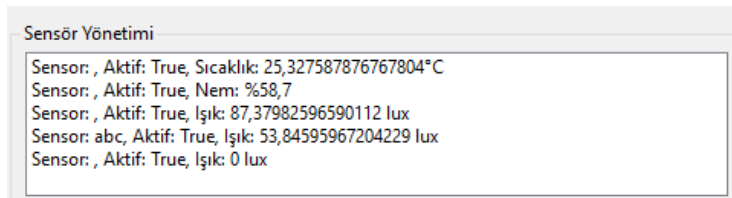
3.5 Aktüatör Ekleme

1. Sera bölgesi seçilir
2. Aktüatör adı ve tipi seçilir (Sulama, Isıtıcı, Fan, Lamba)
3. "Aktüatör Ekle" butonuna tıklanır



3.6 Kontrol ve Otomasyon

1. Sera bölgesi ve bitki türü seçilir
2. "Kontrol Et" butonuna tıklanır
3. Sistem sensör verilerini okur ve aktüatörleri kontrol eder



Kod:

```
public void KontrolEt(int seraBolgesiId, BitkiTuru bitkiTuru) {
    var bolgeSensorleri = _sensorler
        .Where(s => s.SeraBolgesiId == seraBolgesiId).ToList();

    foreach (var sensor in bolgeSensorleri) {
        double deger = sensor.DegerOku();
        if (sensor is NemSensoru && deger < bitkiTuru.NemGereksinimi -
10) {
            var sulama = bolgeAktuatorleri
                .FirstOrDefault(a => a is SulamaVanasi) as SulamaVanasi;
            sulama?.Ac();
        }
    }
}
```

4. CRUD İŞLEMLERİ ÖRNEKLERİ

4.1 Ekleme (Create)

```
public void Ekle(object item) {
    if (item is BitkiTuru bitki) {
        using (var connection = new SQLiteConnection(_connectionString))
        {
            connection.Open();
            string sql = @"INSERT INTO BitkiTurleri
                (Ad, SuGereksinimi, IsikGereksinimi, NemGereksinimi)
                VALUES (@Ad, @Su, @Isik, @Nem)";
            // ... parametreler ve ExecuteNonQuery
        }
    }
}
```

4.2 Listeleme (Read)

```
public List<object> Listele() {
    List<object> liste = new List<object>();
```

```

using (var connection = new SQLiteConnection(_connectionString)) {
    connection.Open();
    string sql = "SELECT * FROM BitkiTurleri";
    // ... reader ile veri okuma ve liste doldurma
}
return liste;
}

```

4.3 Güncelleme (Update)

```

public void Guncelle(int id, object item) {
    string sql = @"UPDATE BitkiTurleri SET Ad = @Ad,
        SuGereksinimi = @Su WHERE Id = @Id";
    // ... parametreler ve ExecuteNonQuery
}

```

4.4 Silme (Delete)

```

public void Sil(int id) {
    string sql = "DELETE FROM BitkiTurleri WHERE Id = @Id";
    // ... ExecuteNonQuery
}

```

Tablolar (3)		
BitkiTurleri		
Id	INTEGER	"Id" INTEGER
Ad	TEXT	"Ad" TEXT NOT NULL
SuGereksinimi	REAL	"SuGereksinimi" REAL
IsikGereksinimi	REAL	"IsikGereksinimi" REAL
NemGereksinimi	REAL	"NemGereksinimi" REAL
SeraBolgeleri		
Id	INTEGER	"Id" INTEGER
Ad	TEXT	"Ad" TEXT NOT NULL
BitkiTuruId	INTEGER	"BitkiTuruId" INTEGER
Konum	TEXT	"Konum" TEXT
sqlite_sequence		
name		"name"
seq		"seq"

	Id	Ad	SuGereksinimi	IsikGereksinimi	NemGereksinimi
	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	Domates	2.0	0.0	0.0
2	3	Salatalık	2.0	50.0	20.0
3	4	Dömpğ	2.0	0.0	0.0

5. SONUÇ

5.1 Proje Özeti

Proje başarıyla tamamlanmış ve tüm modüller uyum içinde çalışmaktadır. Sistem, bitki türlerinin gereksinimlerine göre otomatik kontrol yapabilmektedir.

5.2 Kullanılan C# Özellikleri

- ✓ Interface Inheritance (Interface 1 → Interface 3)
- ✓ Default Interface Methods (C# 8+)
- ✓ Properties (get/set)
- ✓ CRUD İşlemleri
- ✓ Abstract Class ve Abstract Methods
- ✓ Erişim Belirleyicileri (public, private, protected)
- ✓ Custom Exception
- ✓ Exception Handling (try-catch-finally)
- ✓ Nested Class
- ✓ Constructor Overloading
- ✓ Method Overloading
- ✓ Method Overriding
- ✓ Static Methods ve Variables
- ✓ Generic Collections (List, Dictionary<K,V>)
- ✓ LINQ Queries
- ✓ System.IO Kullanımı

5.3 Teknolojiler

- **Programlama Dili:** C# (.NET 8.0)
- **Arayüz:** Windows Forms

- **Veritabanı:** SQLite
- **Mimari:** Object-Oriented Programming (OOP)

.vs	28.12.2025 17:54	Dosya klasörü	
Actuators	28.12.2025 12:31	Dosya klasörü	
bin	28.12.2025 17:50	Dosya klasörü	
Controllers	28.12.2025 12:31	Dosya klasörü	
DataAccess	28.12.2025 12:31	Dosya klasörü	
Exceptions	28.12.2025 12:30	Dosya klasörü	
Interfaces	28.12.2025 12:30	Dosya klasörü	
Models	28.12.2025 12:31	Dosya klasörü	
obj	28.12.2025 17:54	Dosya klasörü	
Sensors	28.12.2025 12:30	Dosya klasörü	
Utils	28.12.2025 17:13	Dosya klasörü	
AkilliSera.csproj	28.12.2025 17:54	C# Project File	1 KB
AkilliSera.csproj.user	28.12.2025 17:56	Per-User Project O...	1 KB
Form1.cs	28.12.2025 23:06	C# Source File	17 KB
Form1.Designer.cs	28.12.2025 12:33	C# Source File	19 KB
Program.cs	28.12.2025 23:04	C# Source File	1 KB
sera.db	28.12.2025 23:08	Data Base File	16 KB
sera_log.txt	28.12.2025 23:09	Metin Belgesi	1 KB

Hazırlayan: Burak Çetinkaya

Öğrenci No: 241103036

Tarih: Aralık 2025