PLAN S - Yer Sistemleri Teknikeri Vaka Çalışması Raporu

1- Giriş Ve Problem Analizi :

İstenilen vaka çalışmasında, uydu haberleşmesinin temeli olan uydu konumunun güncellenmesinin otomatize edilmesi amaçlanıyor. Görev, belirtilen URL'den uydu bilgisini alınması, her uydu için ayrı **.tle** uzantılı dosyalar oluşturmak ve bu dosyaları network de başka bir cihazın içindeki belirli konuma gönderen bir script yazılması isteniyor.

2 - Geliştirme süreci

Öncelikle TLE verisinin formatını anlamaya yönelik bir araştırma gerçekleştirdim. Başta Wikipedia olmak üzere birkaç video ile hangi veri üzerinde çalıştığımı inceledim. Sonrasında istenilen script için basit bir plan oluşturdum ve nasıl yazılabileceği ile hangi kütüphaneleri kullanmanın mantıklı olabileceği konusunu araştırdım. Araştırmalarımın ufak bir kısmında yapay zekadan yardım aldım, ama olabildiğince hazır bilgi almamaya çalıştım.

3 - Scriptin Çalışma Mantığı

Öncelikle çalışma mantığı ve kullanılabilirlik bakımından 4 fonksiyon olacak şekilde yazdım. 3'ü işlemlerin gerçekleştirildiği fonksiyonla, 1'i ise **main** fonksiyonu.

1 - get_tle_data fonksiyonunda, verilen URL'den TLE verisini çekiyorum. Hata olma ihtimaline karşı basit bir kontrol amaçlı durum kodlarını yazdırıyorum. Eğer bağlantı koparsa veya bir sorun olursa except kısmı devreye giriyor.

```python
def get_tle_data(url):
    """
        Verilen urlden tle verilerini çekiyor. Hata durumunda hatayi yazdiriyor. Try bloğunda veri sorunsuz
        çekildi ise durum kodu 200 ekrana yazdiriliyor. Eğer veri düzgün çekilemediyse hangi durum kodu olduğu
        yazdiriliyor. Except bloğu ise bağlanti kesintisi ile ilgili bir sorun varsa devreye giriyor.
    """
    response = requests.get(url)

    try:
        if response.status_code == 200:
            print("ok")
            return response
        else:
            print(response.status_code)
            return None

    except requests.exceptions.RequestException as e:
        print(e)
        return None
```

2 - save_tle fonksiyonunda, döngüye girmeden önce gelen verinin sadece text kısmını alıyorum ve splitlines ile satırlara bölüyorum. Döngü içinde TLE verileri her uydu için 3'lü gruplar halinde olduğu için, önce uydu adını sonra da iki satırda gelen bilgilerini alıyor. Sonrasında uydu adının sağındaki veya solundaki boşlukları temizleyip, aradaki boşluklara "_" koyarak devam ediyorum. Uydu adını dosya adı olarak kullanıp bir dosya oluşturuyorum. with open ile güvenli şekilde dosyayı açıyorum, içine yazılması gereken bilgileri yazdırıyorum ve işi bitince kendisi otomatik kapatıyor. Döngü başlamadan önce oluşturduğum boş listeye her dosya yolunu ekliyorum. Fonksiyon sonunda da bu listeyi doldurulmuş şekilde return ediyorum

```
def save_tle(tle_data,output_dir):
    """
        Çektiğimiz veriyi satir satir düzenleyip dosya ismi olarak uydunun adini yaziyoruz. İçine ise uydu adi ve bilgilerini yazdiriyoruz.
    """
    created_files_list = []
    tle_data_text = tle_data.text
    clean_tle_data = tle_data_text.splitlines(True)#Verimizi satirlara bölüyoruz.
    for i in range (0, len(clean_tle_data), 3):
        tle_name = clean_tle_data[i]
        tle_line1 = clean_tle_data[i+1]
        tle_line2 = clean_tle_data[i+2]

        """Başindaki ve sonundaki gereksiz boşluklari siliyor sonra aralardaki boşluklar yerine _ koyuyor"""
        clean_tle_name = tle_name.strip()
        file_name = f"{clean_tle_name.replace(' ', '_')}.tle"

        file_path = os.path.join(output_dir,file_name)

        """Güvenli şekilde dosyayi açiyor yazdirma işlemleri bitince otomatik kendisi kapatiyor"""
        with open(file_path, 'w') as file:
            file.write(tle_name + '\n')
            file.write(tle_line1 + '\n')
            file.write(tle_line2)
        created_files_list.append(file_path)
    return created_files_list
```

3 - transfer_file fonksiyonunda, döngüye girmeden önce dosyanın gönderileceği hedef oluşturuluyor. Döngü içinde ise dosya listesindeki her dosya hedef noktaya gönderiliyor. Eğer bir hata olursa except bloğu devreye giriyor.

```
def transfer_file(file_list, user, ip, path):
    """Hedef bilgisayardaki klasöre dosyayi gönderme işlemi burada yapiyor"""
    destination = f"{user}@{ip}:{path}"
    for file_path in file_list:
        try:
            #shutil.copy(file_path,path) Kendi masaüstümde denemek için eklemiştim.
            command = ["scp", file_path, destination]
            subprocess.run(command, check=True, capture_output=True, text=True)
        except subprocess.CalledProcessError as e:
            print(e.stderr.strip())
```

4 - main fonksiyonunda ise son olarak artık tüm fonksiyonları çalıştırmaya başladığımız noktaya geliyoruz. Benim de pek kullanmadığım ve bir nevi ilk defa bu kadar yakın olduğum **argparse** konusuna geliyoruz. Gönderilen case'deki "Not: Yazılan scriptte TLE URL'si, networkteki bilgisayar IP'si ve kopyalanacak dosya konumu jenerik olmalı ve kullanıcı tarafından verilmelidir." yazısındaki sorunu çözmek için argparse kullanıyorum. Kullanım mantığı olarak bir cihazı çalıştırmadan önce kontrol panelinde girdiğimiz son ayarlar benzetmesi yapabilirim **argparse** için.

Sonrasında fonksiyonlarımızı çalıştırıyoruz. **with tempfile.TemporaryDirectory()** kısmı ise geçiçi bir klasör oluşturuyor ve script bu klasörü kullanarak burada dosyaları yaratıyor işi bitince ise klasör otomatik siliniyor.

```python
def main():
    """Kontrol paneli mantiği gören scriptin isteklerini verdiğimiz nokta"""
    parser = argparse.ArgumentParser()
    parser.add_argument("--url", type=str, required=True)
    parser.add_argument("--ip", type=str, required=True)
    parser.add_argument("--user", type=str, required=True)
    parser.add_argument("--path", type=str, required=True)

    args = parser.parse_args()

    tle_data = get_tle_data(args.url)

    """Tle verisi alinamazsa script sonlandirilir"""
    if not tle_data:
        return

    with tempfile.TemporaryDirectory() as temp_dir:
        created_files = save_tle(tle_data, temp_dir)
        transfer_file(created_files, args.user, args.ip, args.path)
```

4 - Periyodik Çalıştırma ve Otomasyon Önerisi

Araştırmalarım sonucu istenilen periyodik görevlendirme için uygun çözüm **cron** zamanlayıcısı olduğuna karar verdim. Komut veya scriptleri belirli zaman aralıklarında **cron** ile otomatik çalıştırabiliriz.

Terminalde **crontab -e** yazıyoruz. Açılan kısımda mesela örnek olarak her 3 saatte bir çalışmasını istiyorsak şu şekilde yazabiliriz.

0 */3 * * * python3 /home/btlba/Desktop/tle_script.py --url "https://celestrak.org/NORAD/elements/gp.php?NAME=CONNECTA&FORMAT=tle" --ip "192.168.1.180" --user "btlba" --path "TLE_DATA"

Buradaki **0 */3 * * *** kısmı cronun her 3 saatte bir çalışmasını sağlıyor. Geri kalanı ise scripti çalıştırmak için tam dosya yolu ve argparse ile ayarladığımız zorunlu olan parametreler giriliyor.

5 - Sonuç, Değerlendirme Ve Hatalarım

Case çalışmasında istenilen scripti elimden geldiğince okunabilir ve anlaşılabilir şekilde yazmaya çalıştım. Scripti yazarken en zorlandığım kısım **argparse** ve **tempfile** kısımları oldu. Daha önce neredeyse hiç kullanmadığım argparse için dökümantasyon, örnekler ve biraz yapay zekadan yardım alarak ilerledim. Kodu yazarken bir çok hata ile karşılaştım. Ama yinede kodu çalışabilir şekilde toparladım ve testlerimi gerçekleştirdim. Hatta bu testler sırasında verileri çektiğimiz sistemden 2 saatlik ban bile yedim.

En büyük hatalarımdan biri kodun planlanmasında ve kod aşamasında, kullanıcıdan alınacak bilgileri **input()** ile almaya çalışmak oldu. Çünkü script otomatize edilmek istendiğinde her çalıştırmada tekrar kullanıcıdan bilgi istemesi sorun yarattı. Bu sorunu **argparse** ile çözdüm.

Genel olarak başlangıçta zorlandığım ve ortalarda kullacıdan bilgi alma kısmında yaşadığım sorunda tıkandığım anlar olsada adım adım ilerleyerek çözdüm. Belki ikinci bir gözle daha iyi yazılabilecek yerler vardır ama şimdilik elimden gelen bu oldu diyebilirim.

6 - Ek Bilgi Veriyi Anlamak

```
CONNECTA IOT-11
1 64557U 25135AF  25278.05344044  .00007888  00000+0  38031-3 0  9999
2 64557  97.4541  31.0786 0002175 135.8518 224.2890 15.18978826 16100
```

SATIR 1

64557 : Uydu katalog numarası
U : Sınıflandırma(U sınıflandırılmamış)

25 : Lansman yılının son iki hanesi
135 : O yıl içinde kaçıncı başarılı fırlatma olduğu
AF : Fırlatmadaki parçalardan biri

25 : Yıl
278.05344044 : Yılın günü (5 ekim)

.00007888 : Ortalama hareketin birinci türevi uydunun yörünge hızının zamanla nasıl yavaşladığı veya hızlandığını gösterir. Genellikle atmosferik sürüklenme den kaynaklanır.

00000+0 : Hızdaki değişimin değişim oranıdır. Genellikle sıfır veya çok küçük bir değer olur.

38031-3: BSTAR sürüklenme terimi atmosferik sürüklenmenin uydu üzerindeki etkisini modelleyen değerdir.

0 : Kullanılan yörünge modelinin türüdür.

999 : TLE verisinin kaçıncı kez üretildiğini gösteren sayaç
9 : Satırdaki tüm rakamların toplamının 10 bölümünden kalandır basit hata kontrolü sağlar.

SATIR 2

97.4541 : Eğim uydunun yörünge düzleminin, Dünyanın ekvator düzlemine göre yaptığı açıdır.

31.0786 : Uydunun yörüngesinin güneyden kuzeye doğru ekvatoru kestiği noktanın gökyüzündeki konumudur(derece cinsinden)

0002175: Yörüngenin ne kadar basık veya eliptik olduğunu belirtir.

135.8518 : Yörüngenin Dünyaya en yakın olduğu noktanın, yörünge düzlemi içindeki konumunu belirtir(derece cinsinden)

224.2890 : Epoch zamanında uydunun yörüngesi üzerindeki konumunu belirtir(derece cinsinden).

15.18978826 : Ortalama hareket uydunun bir gün içinde Dünya etrafında kaç tur attığını gösterir.

1610 : Uydunun, bu verinin oluşturulduğu ana kadar tamamladığı yörünge turu sayısıdır.
0 : Satır 2 için hata kontrolü sağlayan sağlama değeridir.