

**UÇAK VERİLERİNİN YAKALANMASI VE
VERİLER İLE YAPILAN OLASI ÇALIŞMALAR**

Erdem GÜNEŞ
180227062
Onur YILDIZ
180227009

BİTİRME ÇALIŞMASI

KOCAELİ
Mayıs, 2022

**KOCAELİ ÜNİVERSİTESİ  MÜHENDİSLİK FAKÜLTESİ
HARİTA MÜHENDİSLİĞİ BÖLÜMÜ**

**UÇAK VERİLERİNİN YAKALANMASI VE
VERİLER İLE YAPILAN OLASI ÇALIŞMALAR**

**Erdem GÜNEŞ
180227062
Onur YILDIZ
180227009**

BİTİRME ÇALIŞMASI

**Danışman : Doç. Dr. Taner ÜSTÜNTAŞ
Üye : Prof. Dr. Arzu ERENER
Üye : Doç. Dr. Erman ŞENTÜRK**

**KOCAELİ
Mayıs, 2022**

IÇİNDEKİLER

| | Sayfa |
|--|--------------|
| IÇİNDEKİLER..... | ii |
| KISALTMALAR..... | iv |
| ŞEKİL LİSTESİ..... | v |
| ÇİZELGE LİSTESİ..... | vi |
| ÖNSÖZ..... | vii |
| ÖZET..... | viii |
| ABSTRACT..... | ix |
| 1. GİRİŞ..... | 1 |
| 1.1. Python..... | 1 |
| 1.1.1. Python Nedir?..... | 1 |
| 1.1.2. Python'un Tarihi..... | 2 |
| 1.2. Html (Hypertext Markup Language)..... | 4 |
| 1.3. Css (Cascading Style Sheet)..... | 4 |
| 1.4. Javascript..... | 5 |
| 1.5. Qgis (Quantum Gis)..... | 5 |
| 1.6. Rest Api..... | 6 |
| 1.6.1. Application Programming Interface (Api) Nedir?..... | 6 |
| 1.6.2. Rest Nedir?..... | 6 |
| 1.6.3. Opensky Rest Api..... | 7 |
| 1.6.3.1. Tüm Durum Vektörleri..... | 7 |
| 1.6.3.2. Talepte Bulunma..... | 7 |
| 1.6.3.3. Gelen Yanıt..... | 9 |
| 1.6.3.4. Anonim (Kimliği Doğrulanmamış) Kullanıcılar için Kısıtlamalar..... | 10 |
| 1.6.3.5. Opensky Kullanıcıları için Kısıtlamalar..... | 10 |
| 1.7. OpenSky Network için Raspberry Pi ile Ads-b Baz İstasyonu..... | 11 |
| 1.7.1. OpenSky Network Api Nedir?..... | 11 |
| 1.7.2. Önemli Detaylar..... | 12 |
| 1.8. Baz İstasyonu için Gerekli Donanımlar..... | 13 |
| 1.8.1. Raspberry Pi Ekipmanları..... | 13 |
| Raspberry Pi ekipmanları sistemin yazılım kısmından sorumlu bölüm oluyor..... | 13 |
| 1.8.1.1. Raspberry Pi 4 ve Çift Fanlı Metal Kasa Soğutma Sistemi..... | 13 |
| 1.8.1.2. Hafıza Kartı..... | 14 |
| 1.8.1.3. Adaptör..... | 15 |
| 1.8.2. Alıcının Ekipmanları..... | 16 |
| 1.8.2.1. Anten..... | 16 |
| 1.8.2.2. Rtl-Sdr..... | 17 |
| 1.8.2.3. Her İki Ucu Sma Erkek Bağlantı Girişli Koaksiyel Bakır Kablo..... | 18 |
| 1.9. Ayar Aşamaları..... | 19 |
| 1.9.1. Raspberry Pi'nin Birleştirilmesi..... | 19 |
| 1.9.2. Alıcı Ekipmanlarının Birleştirilmesi..... | 20 |
| 1.9.3. Raspberry Pi Cihazının Yazılım Bölümünün Ayarlanması..... | 20 |
| 1.9.3.1. İşletim Sistemi ile Sd Kartın Hazırlanması..... | 20 |
| 1.9.3.2. İşletim Sisteminin Kurulumunun Gerçekleştirilmesi..... | 21 |
| 1.9.3.3. Rtl-Sdr Alıcısı için Sürücüler Oluşturma ve Yükleme..... | 22 |
| 1.9.3.4. Dump1090 Kod Çözücünün Oluşturulması ve Kurulumu..... | 23 |
| 1.9.4. Opensky Network Hesabının Oluşturulması ve Yeni Bir Alıcının Eklenmesi..... | 25 |
| 2. UYGULAMALAR..... | 28 |

| | |
|--|----|
| 2.1. Python Jupyter Notebook ile Yapılan Uygulama..... | 28 |
| 2.1.1. Python Jupyter Notebook ile Veriyi Görselleştirme..... | 28 |
| 2.1.2. Jupyter Notebook Üzerinde Python ile Sonuç..... | 28 |
| 2.2. Javascript ile Web Sitesi Yaparak Online Görselleştirme..... | 29 |
| 2.2.1. Html Kodları..... | 29 |
| 2.2.2. Css Kodları..... | 30 |
| 2.2.3. Javascript Kodları..... | 30 |
| 2.2.4. Html, Css ve Javascript ile Yapılan Web Sitesi Sonuç..... | 32 |
| 2.3. Qgis Üzerinde Canlı Veri ile Yapılan İşlemler..... | 32 |
| 2.3.1. Verilerin Python Yardımı ile Belirli Aralıklarda Sunucudan İstenmesi..... | 32 |
| 2.3.2. Verilerin Python Yardımı ile Belirli Aralıklarda Sunucudan İstenmesi..... | 33 |
| 2.3.3. Verilerin Qgis Üzerinde Görselleştirilmesi..... | 34 |
| 2.3.4. Verilerin Qgis Üzerinde Python Console Yardımı ile Grafiksel Görselleştirilmesi | 37 |
| 2.3.5. Qgis Python Console Sonuç..... | 38 |
| 3. SONUÇ..... | 38 |
| KAYNAKLAR..... | 39 |
| INTERNET KAYNAKLARI..... | 39 |
| EKLER..... | 40 |
| Ek 1 Jupyter Notebook'da Python ile verilerin görselleştirmesi için kullanılan kodlar..... | 41 |
| ÖZGEÇMİŞ..... | 44 |

KISALTMALAR

| | |
|-------|---|
| ADS-B | Automatic Dependent Surveillance-Broadcast |
| API | Application Programming Interface |
| CWI | Centrum Wiskunde & Informatica |
| CNRI | Corporation for National Research Initiatives |
| DNS | Domain Name System |
| HDMI | High-Definition Multimedia Interface |
| HTML | Hypertext Markup Language |
| PHP | Hypertext Preprocessor |
| HTTP | Hypertext Transfer Protocol |
| IOT | Internet of Things |
| IP | Internet Protocol |
| JSON | Javascript Object Notation |
| XLT | Microsoft Excel |
| PSF | Python Software Foundation |
| SOAP | Simple Object Access Protocol |
| USB | Universal Serial Bus |
| WIFI | Wireless Fidelity |
| W3C | World Wide Web Consortium |

ŞEKİL LİSTESİ

| | Sayfa |
|--|-------|
| ŞEKİL 1: Python programlama dili sürümleri ve destek verildiği yıllar gösterildiği görsel (Wikipedia History of Python 2022)..... | 3 |
| ŞEKİL 2: Dump1090 ile tarayıcı üzerinde alınan verinin görsel hali..... | 11 |
| ŞEKİL 3: Başka bir dump1090 ile alınan verinin görseli..... | 12 |
| ŞEKİL 4: Rasspberry Pi 4 2 gb ile çift fanlı soğutma sistemi..... | 13 |
| ŞEKİL 5: 32 gb ufak hafıza kartı..... | 14 |
| ŞEKİL 6: 5 volt 3.4 amperlik adaptör..... | 15 |
| ŞEKİL 7: 1090 MHz anten..... | 16 |
| ŞEKİL 8: RTL-SDR..... | 17 |
| ŞEKİL 9: 10 metrelik koaksiyel bakır kablo..... | 18 |
| ŞEKİL 10: SD Card Formatter programının arayüzü..... | 20 |
| ŞEKİL 11: Raspberry Pi disk yazıcısının arayüzü..... | 21 |
| ŞEKİL 12: Terminal üzerinde RTL-SDR sürücü testinin terminal görüntüsü..... | 23 |
| ŞEKİL 13: Dump1090 yazılımının çalışması sonucu terminalde oluşan görüntü..... | 24 |
| ŞEKİL 14: Dump1090 yazılımının yerel tarayıcı üzerinde oluşturduğu görüntü..... | 24 |
| ŞEKİL 15: OpenSky-Network hesap oluşturma bölümünün ekran görüntüsü..... | 25 |
| ŞEKİL 16: OpenSky-Network'te My OpenSky bölümünün ekran görüntüsü..... | 25 |
| ŞEKİL 17: Cihaz kayıt ekranının ilk yarısı..... | 26 |
| ŞEKİL 18: Cihaz kayıt ekranının sonraki yarısı..... | 26 |
| ŞEKİL 19: Cihazı kayıt ettikten sonraki My OpenSky bölümünün ekran görüntüsü..... | 27 |
| ŞEKİL 20: Kayıtlı cihazın bölümüne girildiğinde çıkan ekran..... | 27 |
| ŞEKİL 21: Jupyter Notebook üzerinde Python kodunu çalıştırdıktan sonra oluşan bölümün anlık ekran görüntüsü..... | 28 |
| ŞEKİL 22: Web sitesinden alınan anlık ekran görüntüsü..... | 32 |
| ŞEKİL 23: QGIS veri yönetimi bölümünün ekran görüntüsü..... | 34 |
| ŞEKİL 24: Verilerin eklenmesinden sonra oluşan ekranın görüntüsü..... | 34 |
| ŞEKİL 25: Katman ayarı bölümünde şekillerin ayarının yapıldığı bölümün ekran görüntüsü | 35 |
| ŞEKİL 26: Katman ayarlarında şekillerin düzenlenmesinden sonra yeni şekillerin ekran görüntüsü..... | 35 |
| ŞEKİL 27: Katman ayarları bölümde bulunan katman yenilenme süresi ayarının yapıldığı kısmın ekran görüntüsü..... | 36 |
| ŞEKİL 28: QGIS içerisinde Python Console bölümünün açılış ekran görüntüsü..... | 37 |
| ŞEKİL 29: Python Console bölümünde çalıştırılan kod sonucu çıkan sonucun ekran görüntüsü | 38 |
| ŞEKİL 30: Verileri toplamak için gerekli olan donanımların birleştirilmiş hali..... | 39 |

ÇİZELGE LİSTESİ

| | Sayfa |
|--|--------------|
| ÇİZELGE 1: Python'un bazı sürüm bilgilerini göstermektedir..... | 2 |
| ÇİZELGE 2: Talep parametreleri..... | 8 |
| ÇİZELGE 3: Koordinat sisteminde alabileceği değerlerin veri tipleri..... | 8 |
| ÇİZELGE 4: Gelen yanıtın veri tipleri..... | 9 |
| ÇİZELGE 5: Gelen veri içerisindeki veriler ve verilerin tipleri..... | 9 |

ÖNSÖZ

Bu çalışmadaki amaç Türkiye hava sahasındaki uçakların anlık olarak konum, hız ve benzeri bilgileri paylaşması ve bu paylaşımıları kurulan anten düzeneği ile yakalayıp yayinallyamaktır. Bunun üzerine çekilen veriler halka açık bir şekilde paylaşılmak için internet ortamında HTML, CSS ve JavaScript programlama dili yardımcı ile web sayfası oluşturulmuştur. Son olarak Python programlama dili ile çekilen verilerin QGIS üzerinde gösterilip bu verilerin değerlendirilmesi ile ilgili bir uygulama yapılmıştır.

Bu çalışmada görüşleri ile bizleri destekleyen Kocaeli Üniversitesi Harita Mühendisliği bölümü öğretim görevlilerinden ve fikir hocamız Doç. Dr. Taner Üstüntaş'a teşekkürlerimizi sunarız.

Erdem Güneş & Onur Yıldız

Mayıs 2022, Kocaeli

ÖZET

Bu çalışmada, araştırma için Açık Hava Trafik Verileri kullanılmıştır. Uçaklar sürekli olarak konum ve başka bilgilerini paylaşarak güvenli seyahat oranını artırmaktadır. Uçaklar bu bilgilerini radyo dalgaları ile her yere gönderirler. Uçakların yaydığı sinyalleri yakalamak için antene ihtiyaç duyulur. Bu anten ile uçak tarafından yayılan sinyaller yakalanır ve diğer donanımlarında yardımcı ile veriler temizlenip işlendikten sonra uçakların yaydığı bilgilere ulaşılmış olur. Bu uygulamada OpenSky-Network baz istasyonu olarak kullanıldı.

Anten çekim alanı sınırlıdır, daha büyük veriye ulaşmak için OpenSky-Network ağından faydalabilir. Bu veriler antenler yardımıyla toplandıktan sonra OpenSky-Network sunucularına gönderilir ve sunucularda toplanır. Daha sonrasında bu verilere ihtiyaç duyanlar verileri OpenSky Rest API sayesinde çeker.

Kodlama yardımıyla veriler çekilir ve veri düzenlenerek sonra içerisinde kullanılmak istenen kısımlar çekilir. Bu işlemlerde yapıldıktan sonra o verilerden ne yapılım istendiği kullanıcıya kalmıştır.

Bu çalışmada, ilk uygulama olarak; Jupyter Notebook üzerinde Python programlama dili ile Türkiye hava sahasına giren uçaklar görselleştirildi. İkinci uygulama olarak; veri çekme ve o verilerden sonuç elde etmeye ihtiyaç duymadan başka kişilerinde uçak konumunu görebilmesi amaçlı internet ortamında JavaScript programlama dili yardımıyla web uygulaması yapıldı. Son olarak; Python programlama dili yardımıyla çekilen verilerin canlı bir şekilde QGIS üzerinde gösterilmesi ve bu verilerin QGIS Python Konsole üzerinde nasıl değerlendirileceğine dair bir uygulama yapıldı.

Anahtar Sözcükler: Baz istasyonu, OpenSky-Network, OpenSky Rest API, Jupyter Notebook, Python programlama dili, JavaScript programlama dili, QGIS.

ABSTRACT

In this study, Open Air Traffic Data was used for research. Airplanes constantly share location and other information, increasing the rate of safe travel. Airplanes send this information to everywhere by radio waves. An antenna is needed to catch the signals emitted by airplanes. With this antenna, the signals broadcast by the aircraft are captured and the information emitted by the aircraft is reached after the data is cleaned and processed with the help of other equipment. In this application, OpenSky-Network was used as a base station.

Antenna coverage is limited, OpenSky-Network network can be used to reach larger data. After this data is collected with the help of antennas, it is sent to OpenSky-Network servers and collected on the servers. Those who need this data later pull the data with the OpenSky Rest API.

With the help of coding, the data is drawn and after the data is organized, the parts that are desired to be used are drawn. After these processes are done, it is up to the user what to do with that data.

In this study, as a first step; The planes entering the Turkish airspace were visualized with the Python programming language on the Jupyter Notebook. In the second stage; A web application was made on the internet with the help of JavaScript programming language in order to enable other people to see the aircraft location without the need to extract data and obtain results from that data. In the last stage; An application was made to show the data captured with the help of Python programming language live on QGIS and how these data can be evaluated on QGIS Python Console.

Keywords: Base station, OpenSky-Network, OpenSky Rest API, Jupyter Notebook, Python programming language, JavaScript programming language, QGIS.

1. GİRİŞ

Bu bölümde Python programlama dili, HTML, CSS, JavaScript programlama dilinin yanında QGIS gibi konulara değinilecektir.

1.1. Python

Konunun daha iyi anlaşılabilmesi için kullanılan programlama dili olan Python programlama dili ne olduğuna ve tarihine değinelim.

1.1.1. Python Nedir?

Python yorumlayıcı, nesne tabanlı, üst düzey bir dinamik semantik programlama dilidir. Yüksek seviye veri yapılarını işleyebilir, dinamik yazma ve dinamik birleştirme yapabilir, bu da hızlı uygulama yazma ve geliştirmeyi kolay hale getirir. Python basit ve kolay öğrenilen bir dil yapısına sahiptir. Python program yapısını ve yeniden kullanım için eklenti ve paketlerini destekler. Python yorumlayıcısı ve kapsamlı standart kütüphaneleri, bütün büyük platformlar için ücretsiz olarak yazılım veya ikilik sisteme mevcuttur ve ücretsiz olarak dağıtılabılır (Özgül, 2016).

Programcılar, sağladığı üretkenlik nedeniyle genellikle Python'a hayran kalırlar. Python'da derleme işlemi olmadığından, düzenleme, test ve hata ayıklama döngüsü hızlıdır. Python programlama dilinde hata ayıklama kolaydır; bir hata veya hatalı giriş bölünme hatasına sebep olmaz. Bunun yerine, yorumlayıcı bir hata bulduğunda bir istisna oluşturur. Program istisnayı yakalayamadığı zaman, yorumlayıcı bir yığın izi yazdırır. Hata ayıklayıcı, yerel ve genel değişkenlerin incelenmesine, rastgele ifadelerin değerlendirilmesine, kesme işaretlerinin ayarlanması, kodda bir seferde bir satır ve adım adım ilerleme gibi seçeneklere izin verir. Hata ayıklayıcısı, Python programlama dili ile yani kendisiyle yazılmıştır. Bir başka özelliği, genellikle bir programda hata ayıklamanın en hızlı yolu, koda birkaç yazdırma (print) fonksiyonu eklemektir. Hızlı düzenleme, test ve hata ayıklama döngüsü bu basit yaklaşımı çok etkili kılar [15].

Python programlama dilinin temelleri C programlama dili ile atılmıştır. Her ne kadar Python programlama dilinin adı piton (python) yılanından geldiği düşünülse de öyle değildir; Guido Van Rossum dilin adını, bir İngiliz komedi grubu olan “Monty Python’s Flying Circus” adlı gösterisinden esinlenerek koymuştur. Buna rağmen Python programlama dilinin pek çok yerde yılan figürü ile temsil edilmesi neredeyse gelenek haline gelmiştir (Özgül, 2016).

1.1.2. Python'un Tarihi

Python programlama dili, Guido van Rossum tarafından 1989'lu yıllarda Hollanda'da CWI (Centrum Wiskunde & Informatica: Matematik ve Bilgisayar Bilimleri Merkezi) araştırma merkezinde yapımına başlandı. Python programlama dili başkaları tarafından destek almasına rağmen Guido van Rossum halen Python'un başyazarı olmaya devam etmektedir.

1995'te, Guido Virginia eyaletinin Reston kasabasında bulunan CNRI'da (Corporation for National Research Initiatives: Ulusal Araştırma Girişimleri Kurumu) Python üzerinden çalışmaya devam etti. CNRI'da çalışırken bir kaç Python sürümü yayınladı.

2000 yılının Mayıs ayında, Guido ve Python yazılım geliştirme ekibi BeOpen PythonLabs ekibini oluşturmak için BeOpen.com'a taşındılar. Aynı yılın Ekim ayında, PythonLabs takımı Digital Creations'a (şimdiki isimleri: Zope Corporation) geçiş yaptılar. 2001 yılında, PSF (Python Software Foundation: Python Yazılım Vakfı) oluşturuldu, Python ile ilgili Intellectual Property'e (Fikri Mülkiyet) sahip olabilmek için özellikle oluşturuldu, herhangi bir kar amacı gütmeyen bir kuruluştur. Zope Corporation, PSF'nin sponsor üyesidir.

Bütün Python sürümleri Open Source (Açık Kaynak) üzerinden yayınladı. Tarihsel olarak, çoğu ama hepsi değil Python sürümlerinin bazıları ayrıca GPL (General Public License) uyumlu olarak yayınladı.

Not: GPL uyumlu olması Python'u GPL altında yayinallyamak demek değildir. Bütün Python lisansları, GPL'den farklı olarak, yaptığınız değişiklikleri açık kaynak olmadan da dağıtmanıza izin verir. GPL uyumlu lisanslar, diğerlerinin aksine; Python'un GPL kapsamında yayınlanan diğer yazılımlarıyla birleştirmeyi mümkün kılar (Samancıoğlu, 2021).

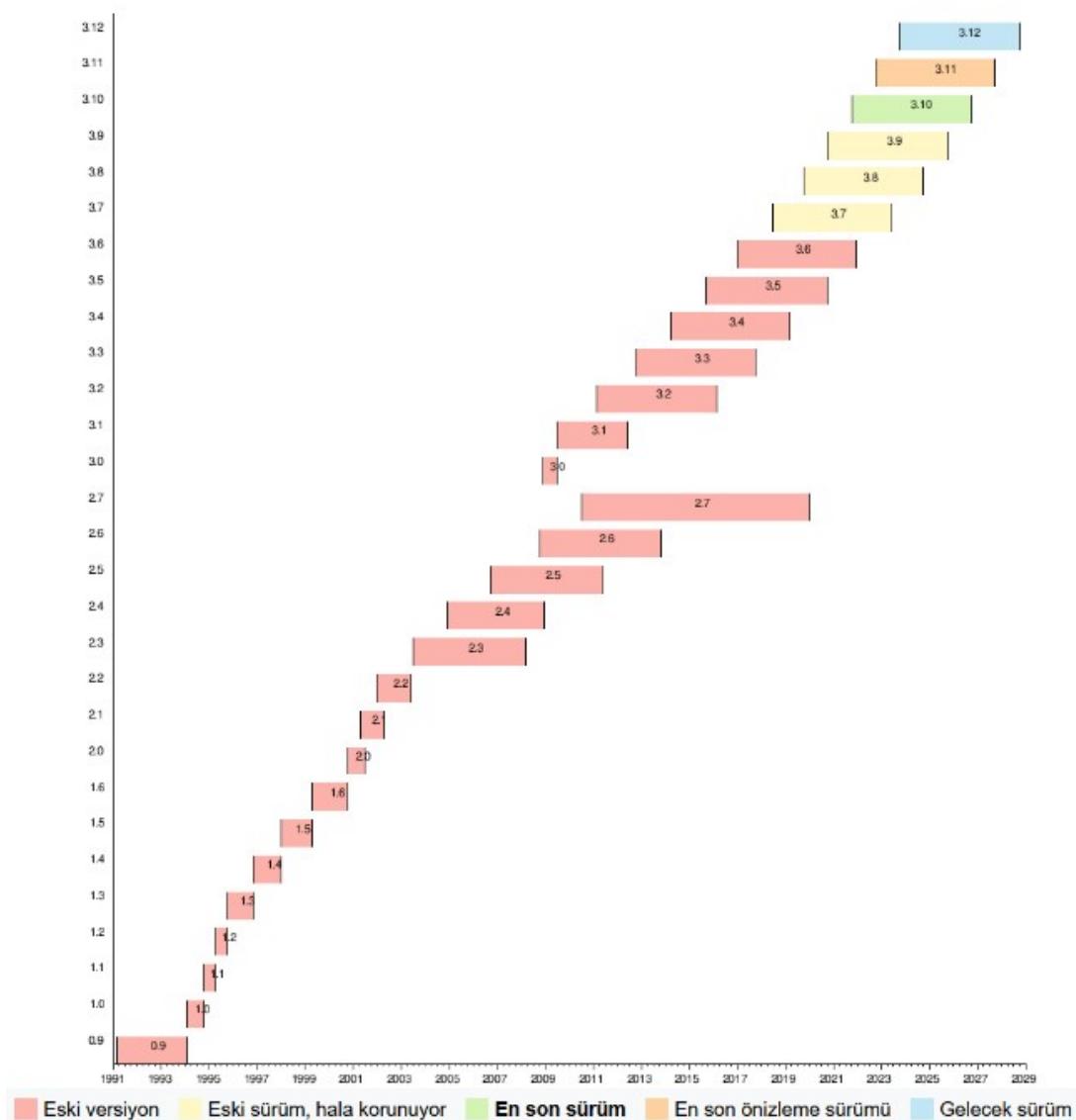
Çizelge 1 içerisinde Python'un hangi sürümünden türetildiği, hangi yılda türetildiği, hak sahibi ve lisans uyumu bilgileri bulunmaktadır [1].

ÇİZELGE 1: Python'un bazı sürüm bilgilerini göstermektedir

| Sürüm | Türetildi | Yıl | Hak Sahibi | GPL Uyumlu? |
|-------------|-----------|-----------|------------|-------------|
| 0.9.0 - 1.2 | n/a | 1991-1995 | CWI | Evet |
| 1.3 - 1.5.2 | 1.2 | 1995-1999 | CNRI | Evet |
| 1.6 | 1.5.2 | 2000 | CNRI | Hayır |
| 2.0 | 1.6 | 2000 | BeOpen.com | Hayır |
| 1.6.1 | 1.6 | 2001 | CNRI | Hayır |

| | | | | |
|----------------|-----------|-----------------|-----|-------|
| 2.1 | 2.0+1.6.1 | 2001 | PSF | Hayır |
| 2.0.1 | 2.0+1.6.1 | 2001 | PSF | Evet |
| 2.1.1 | 2.1+2.0.1 | 2001 | PSF | Evet |
| 2.1.2 | 2.1.1 | 2002 | PSF | Evet |
| 2.1.3 | 2.1.2 | 2002 | PSF | Evet |
| 2.2 ve sonrası | 2.1.1 | 2001 ve sonrası | PSF | Evet |

Çizelge 1 içerisinde Python 2.2 sürümüne kadar gösterilmiştir. Daha sonraki sürümleri ve ileri aşamalarda yapılan sürüm planları ile ilgili oluşturulmuş görsel bilgiye şekil 1 kısmından ulaşabilirsiniz.



ŞEKİL 1: Python programlama dili sürümleri ve destek verildiği yıllar gösterildiği görsel (Wikipedia History of Python 2022)

1.2. Html (Hypertext Markup Language)

Hypertext Markup Language kısaltması ise HTML olan, kullanıcılar için web sayfasında ve uygulamalarda paragraflar, başlıklar ve bağlantılar oluşturmaya yarar.

HTML bir programlama dili değildir, yani dinamik işlevsellik oluşturma gibi bir özelliği bulunmuyor. Sadece metin belgelerini üzerinde düzenlemeler yapmaya yarar [11].

HTML ile basit kod yapıları yardımı ile internet sayfasını şekillendirebilirsiniz. HTML website kurma konusunda tamamen yeni başlayanlar için bile öğrenmesi kolay bir biçimleme dilidir.

HTML, İsviçre’deki CERN araştırma enstitüsünde bir fizikçi olan Tim Berners-Lee tarafından geliştirilmiştir. HTML’in ilk sürümü 1991 yılında yayınlandı.

HTML dosyaları, .html ile biten belgelerdir. Bu dosyaları Firefox ya da başka herhangi bir internet tarayıcısı yardımı ile görüntüleyebilirsiniz. İnternet tarayıcısı .html uzantılı dosyayı okur ve kullanıcının normal metin belgesi şeklinde görebileceği şekilde içeriğe dönüştürür.

HTML nedir sorusuna en basit cevapsa internetin biçim dili demek yeterli olacaktır. Bütün internet tarayıcılarında yerel olarak çalışabilir ve World Wide Web Consortium tarafından denetlenmektedir [11].

HTML kullanarak web sitelerinin ve uygulamaların içerik yapılarını düzenleyebilir, JavaScript kullanarak işlevsellik katabilir ve CSS ile de şekil verebilirsiniz.

1.3. Css (Cascading Style Sheet)

Cascading Style Sheet yani Türkçesi Basamaklı Stil Şablonu kısaltması ise CSS temelde web sitenizdeki HTML elementlerinin renk, boyut, yazı karakterleri gibi her türlü görsel özellikleri üzerinde oynama yapmanızda yardımcı olur. CSS dosyalarını uzantısı .css şeklindedir [10].

CSS, 1996 yılında W3C (World Wide Web Consortium) tarafından geliştirilmiştir. Bunun nedeni HTML’in sayfayı şekillendirmeye yardımcı etiketlere sahip olacak şekilde tasarılmamış olmasıydı.

Web sitesi oluştururken HTML ve CSS arasında güçlü bir ilişki vardır. HTML bir biçimlendirme dili CSS ise stili vurguladığı için bir biri arasında güçlü bir bağ oluşur.

1.4. Javascript

JavaScript, web sayfaları, uygulamalar, sunucular ve oyunlar geliştirirken daha dinamik etkileşimler oluşturmak için yaygın olarak kullanılan bir programlama dilidir. Genellikle HTML ve CSS'nin yanında JavaScript kullanılır.

JavaScript, Netscape çalışanı olan Brandan Eich tarafından 1995 yılının Eylül ayında 10 günlük bir sürede oluşturuldu. İlk ismi Mocha sonrasında Mona ve JavaScript olmadan önce LiveScript adını aldı. JavaScript, farklı tarayıcılarda çalışmanın yanı sıra aynı zamanda mobil ve masaüstü bilgisayarlar gibi farklı makinelerde de çalıştırılabilir [12].

JavaScript, web sitelerine dinamiklik kazandırabilmektedir. Web sitesi oluştururken sadece HTML ve CSS kodu kullanılabilir ama bu ekran sadece statik bir ekrana sahip olur. JavaScript yardımıyla web sitesini ziyaret eden kişi web sitesi ile etkileşime geçebilir.

JavaScript direkt olarak HTML kodlarının içerisinde gömülebilir ya da .js dosyası aracılığıyla dosya çağrılabılır. JavaScript, web sitelerinde genellikle HTML ve CSS ile birlikte kullanılır.

1.5. Qgis (Quantum Gis)

QGIS veri görüntüleme, düzenleme ve çözümleme sağlayan çoklu ortam destekli özgür ve açık kaynak kodlu coğrafi bilgi sistemi (CBS) yazılımıdır [7].

QGIS açık kaynak olması sebebiyle çok hızlı şekilde gelişim sağlayabilmektedir. Aynı zamanda farklı açık kaynak GIS paketleri ile kolay şekilde uyum sağlayabilmektedir. Python veya C++ ile yazılmış eklentilerini de eklenti indirme bölümünden kolayca indirip kullanılabilir.

Gary Sherman 2002 yılı başında Quantum GIS'i geliştirmeye başladı ve 1.0.0 sürümü 2009 yılında yayınlandı [7].

QGIS, C++ ile kodlandığı için hızlı çalışmaktadır. QGIS GNU/Linux, UNIX, Microsoft Windows ve Mac OS gibi farklı ortamda çalışabilmektedir.

QGIS güncellemelerini ve hata düzeltmelerini gönüllü geliştiriciler ile yapmaktadır. QGIS programının 48 dilden fazla tercumesi bulunmaktadır. QGIS'in bu kadar güzel bir ortam sağlama coğrafi bilgi sistemleri için çok kullanılmasını sağlamaktadır.

1.6. Rest Api

Cevrimiçi ortamda veri oluşturma ve paylaşma yöntemlerden biri REST API'dir.

1.6.1. Application Programming Interface (Api) Nedir?

API, uygulama yazılımı oluşturmak ve entegre etmek için bir dizi tanım ve protokoldür. Bazen bir bilgi sağlayıcı ile bilgi kullanıcısı arasındaki, tüketiciden istenen içeriği (çağrı) ve üretici tarafından istenen içeriği (cevap) belirleyen bir sözleşme olarak anılır. Örneğin; bir hava durumu hizmeti için API tasarıımı, kullanıcının bir posta kodu sağlamasını ve üreticinin, ilk olarak yüksek sıcaklık ve ikincisinin düşük olmak üzere iki parçalı bir yanıtla yanıt vermesini belirtebilir.

Başka bir deyişle, bilgi almak veya bir işlevi gerçekleştirmek için bir bilgisayar veya sistem etkileşim kurmak istiyorsanız, API, istediği anlayıp yerine getirebilmesi için o sisteme ne istediğini iletmemize yardımcı olur.

Bir API'yi kullanıcılar veya istemciler ile almak istedikleri kaynaklar veya web hizmetleri arasında bir aracı olarak düşünülebilir. Aynı zamanda bir kuruluşun güvenlik, kontrol ve kimlik doğrulamasını sürdürürken kimin neye erişebileceğini belirleyerek kaynakları ve bilgileri paylaşmasının bir yoludur [14].

API'nin bir başka avantajı da, kaynağınızın nasıl alındığını veya nereden geldiğini bilmenize gerek olmamasıdır.

1.6.2. Rest Nedir?

REST bir protokol veya standart değil, bir dizi mimari kısıtlamadır. API geliştiricileri REST'i çeşitli şekillerde uygulayabilir.

RESTful API aracılığı ile bir istemci isteği yapıldığında kaynağın durumunun bir temsilini istekte bulunana veya son kısma aktarır. Bu bilgi veya beyan, birkaç şekilde sunulabilir. Örneğin; HTTP: JSON (Javascript Object Notation), HTML, XLT, Python, PHP veya metin belgesi formatı tipinde sunabilir. JSON genel olarak en çok kullanılan dosya tipidir çünkü adına rağmen Javascript programlama dilinden bağımsızdır ve insanlar hemde makineler tarafından okunabilir [14].

Bir API'nin RESTful olarak kabul edilebilmesi için şu kriterlere uyması gereklidir.

- İstemciler sunucular ve kaynaklardan oluşan ve istekleri HTTP aracılığı ile yönetilen bir istemci ve sunucu mimarisi olmalıdır.
- İstemci sunucu etkileşimlerini kolaylaştırın önbellege alınabilir veriler olmalıdır.

- Her tür sunucuyu organize eden katmanlı bir sistem, istenen bilgilerin istemci tarafından görülmeyecek şekilde düzenler halinde alınmasını içeriyor olmalıdır.
- İsteğe bağlı kod, istendiğinde sunucudan istemciye yürütülebilir kod gönderme, istemci işlevsellliğini genişletme yeteneğine sahip olmalıdır.

REST API'nin uyması gereken bu özelliklere sahip olmasına rağmen, XML mesajlaşma ve yerleşik güvenlik ve işlem uyumluluğu gibi belirli gereksinimleri olan SOAP (Simple Object Access Protocol) gibi önceden belirlenmiş bir protokole göre kullanımı daha kolay kabul edilir.

Buna karşılık REST, gerektiğinde uygulanabilen artırılmış ölçeklenebilirlik ile REST API'lerini daha hızlı ve hafif hale getiren IOT (Internet of Things: "Nesnelerin İnterneti") ve mobil uygulama geliştirme için mükemmel olan bir dizi yönergedir [14].

1.6.3. Opensky Rest Api

OpenSky Ağrı, 2013'ten beri sürekli olarak hava trafiği gözetleme verilerini toplayan, kâr amacı gütmeyen, topluluk tabanlı bir alıcı ağıdır. Diğer ağların aksine, OpenSky eksiksiz filtrelenmemiş ham verileri tutar ve akademik ve kurumsal araştırmacılar için erişilebilirdir. Kar amacı gütmeyen bir misyon ile, üniversiteler ve diğer kar amacı gütmeyen kuruluşlar tarafından açık küresel hava trafiği araştırmalarını desteklemektedir.

OpenSky REST API bağlantısı: <https://opensky-network.org/api>

Veriler içerisinde durum vektörleri ve uçuş rotalarını almak için kullanılabilecek çeşitli bilgiler vardır.

1.6.3.1. Tüm Durum Vektörleri

Aşağıdaki API çağrısı, OpenSky'ın herhangi bir durum vektörünü almak için kullanılabilir [5].

<https://opensky-network.org/api/states/all>

1.6.3.2. Talepte Bulunma

Çizelge 2 içerisindeki talep parametrelerinden istenilenler kullanılarak belirli uçaklar veya saatler için durum bilgisi talep edilebilir [5].

ÇİZELGE 2: Talep parametreleri

| Nitelik | Veri Tipi | Tanım |
|---------|-------------------|---|
| time | integer (sayısal) | Saniye cinsinden süre durumlarını almak için Unix zaman damgası kullanılır. Yazılmazsa anlık zaman kullanılır. |
| icao24 | string (sözel) | Onaltılk diziyle temsil edilen bir veya daha fazla ICAO24 aktarıcı adresi (örn: abc9f3) kullanır. Birden çok ICAO24 verisini filtrelemek için her adres için bir kez ekleyin. Yazılmazsa tüm uçakların durum vektörleri döndürülür. |

Buna ek olarak WGS84 koordinatlarını ile bir sınırlayıcı bölge oluşturarak sadece o bölgede bulunan uçakların verisini almak mümkündür. Bu amaçla, çizelge 3 içerisindeki parametrelerin hepsini ekleyiniz [5].

ÇİZELGE 3: Koordinat sisteminde alabileceği değerlerin veri tipleri

| Nitelik | Veri Tipi | Tanım |
|---------|-----------|---|
| lamin | float | Enlem için ondalık derece cinsinden alt sınır. |
| lomin | float | Boylam için ondalık derece cinsinden alt sınır. |
| lamax | float | Enlem için ondalık derece cinsinden üst sınır. |
| lomax | float | Boylam için ondalık derece cinsinden üst sınır. |

<https://opensky-network.org/api/states/all>

Zaman ve uçak için örnek soru:

<https://opensky-network.org/api/states/all?time=1458564121&icao24=3c6444>

Türkiye hava sahası için sınırlayıcı bölge örnek soru:

<https://opensky-network.org/api/states/all?lamin=36&lomin=26&lamax=45&lomax=42>

1.6.3.3. Gelen Yanıt

Gelen yanıt, çizelge 4'teki özelliklere sahip bir JSON nesnesidir [5].

ÇİZELGE 4: Gelen yanıtın veri tipleri

| Nitelik | Veri Tipi | Tanım |
|---------|-------------------|---|
| time | integer (sayısal) | Bu yanındaki durum vektörlerinin ilişkili olduğu zamanı temsil eder. Tüm vektörler [time - 1, time] aralığına sahip bir aracın durumunu temsil eder |
| states | array (dizi) | Durum vektörlerini barındırır. |

States (sınıf) özelliği iki boyutlu bir dizidir. Her satır bir durum vektörünü temsil eder ve çizelge 5'teki alanları içerir [5].

ÇİZELGE 5: Gelen veri içerisindeki veriler ve verilerin tipleri

| Dizin Numarası | Nitelik | Veri Tipi | Tanım |
|----------------|----------------|-----------|--|
| 0 | icao24 | string | Özgün ICOA 24 bit adresinde onaltılık gösterimi. |
| 1 | callsign | string | Uçağın çağrı işaretü (8 karakter). Çağrı bilgisi alınmadıysa boş bırakılır. |
| 2 | origin_country | string | ICAO 24 bit adresinden anlaşılan ülke adı. |
| 3 | time_position | int | Son konum güncellemesi için, Unix zaman damgası saniye cinsinden. Son 15 saniye içinde OpenSky tarafından herhangi bir pozisyon bilgisi alınmadıysa boş bırakılır. |
| 4 | last_contact | int | Genel olarak son güncelleme için Unix zaman damgası saniye cinsinden. Bu alan, aktarıcıdan alınan herhangi yeni, geçerli mesaj için güncellenir. |
| 5 | longitude | float | Ondalık derece cinsinden WGS-84 boylam bilgisi. Boş olabilir. |
| 6 | latitude | float | Ondalık derece cinsinden WGS-84 enlem bilgisi. Boş olabilir. |
| 7 | baro_altitude | float | Metre cinsinden barometrik yükseklik. Bol olabilir. |
| 8 | on_ground | boolean | Konumun bir yüzey konumu raporundan alınıp alınmadığını Boolean tipinde gösterir. |
| 9 | velocity | float | Yere göre hız m/s olarak. Boş olabilir. |

| | | | |
|----|-----------------|---------|---|
| 10 | true_track | float | Kuzeyden saat yönünde ($\text{kuzey}=0^\circ$) gerçek yönü. Boş olabilir. |
| 11 | vertical_rate | float | m/s cinsinden dikey hız. Pozitif bir değer uçağın tırmandığını, negatif bir değer ise alçaldığını gösterir. Boş olabilir. |
| 12 | sensors | int[] | Bu durum vektörü katkıda bulunan alıcıların kimlikleri. İstekte sensör için filtreleme kullanılmadıysa boştur. |
| 13 | geo_altitude | float | Metre cinsinden geometrik yükseklik. Boş olabilir. |
| 14 | squawk | string | Aktarıcı kodu. Boş olabilir. |
| 15 | spi | boolean | Uçuş durumunun özel amaçlı göstermesi olup olmadığını. |
| 16 | position_source | int | Konumunun kökeni: 0 = ADS-B, 1 = ASTERIX, 2 = MLAT |

1.6.3.4. Anonim (Kimliği Doğrulanmamış) Kullanıcılar için Kısıtlamalar

Kimlik bilgilerini kullanmadan API'ye erişen kullanıcılar anonimdir. Anonim kullanıcılar için kısıtlamalar şunlardır:

- Anonim kullanıcılar, yalnızca en son durum vektörlerini alabilir, yani zaman parametresi yok sayılacaktır.
- Anonim kullanıcılar, yalnızca 10 saniyelik bir zaman aralığına sahip verileri alabilir. Bu API'nin zaman için durum vektörlerini $[\text{now} - (\text{now mod } 10)]$ döndüreceği anlamına gelir [5].

1.6.3.5. OpenSky Kullanıcıları için Kısıtlamalar

OpenSky kullanıcıı, API'ye erişmek için geçerli bir OpenSky hesabı kullanan kişilere denir.

OpenSky kullanıcılar için kısıtlamalar şunlardır:

- OpenSky kullanıcıları geçmişte 1 saat kadar verileri alabilir. Zaman parametresinin bir değeri varsa $[t < \text{now} (\text{şimdi}) - 3600]$ API, 400 bozuk istek döndürür.
- OpenSky kullanıcıları, 5 saniyelik bir zaman hassasiyeti ile verileri alabilir. Bunun anlamı eğer zaman parametresi olarak ayarlanmışsa 't' API zaman için $[t - (t \text{ mod } 5)]$ durum vektörlerini döndürür [5].

1.7. OpenSky Network için Raspberry Pi ile Ads-b Baz İstasyonu

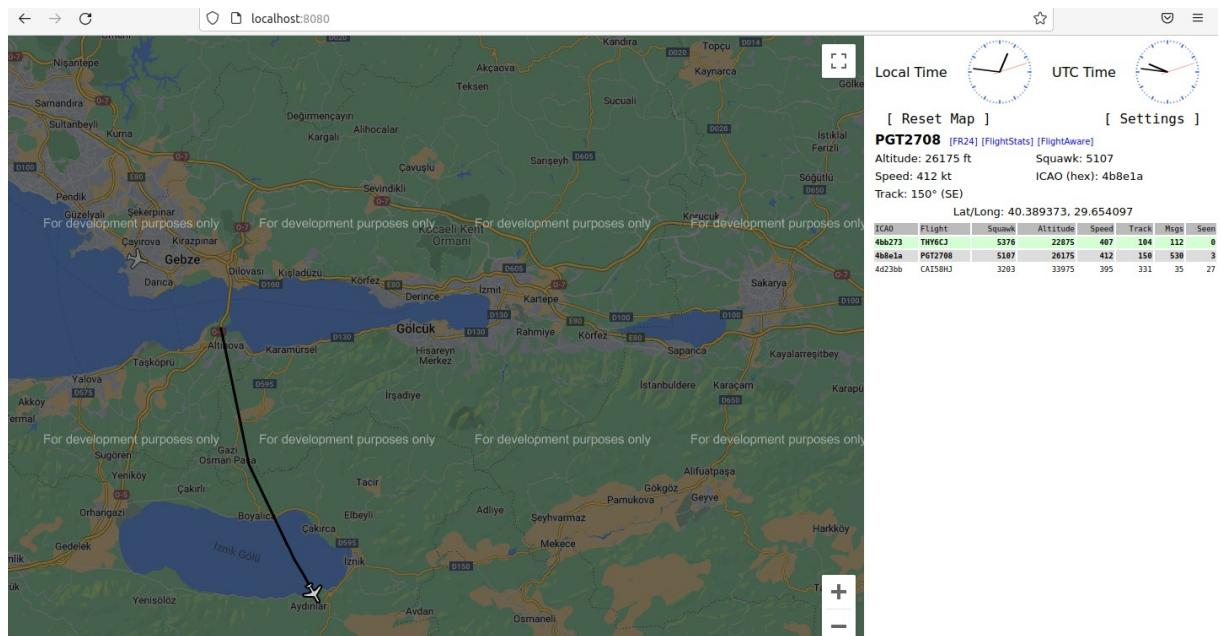
OpenSky gibi siteler uçak verilerini yakalayabilmek için baz istasyonlarına ihtiyaç duyar. ADS-B (Automatic Dependent Surveillance-Broadcast) uçak verilerini yakalayabilmek için kullanılan bir baz istasyonu sistemidir.

1.7.1. OpenSky Network Api Nedir?

Bu yardımcı kılavuz, Raspberry Pi ve hazır parçalar kullanılarak basit bir ADS-B baz istasyonunun nasıl işlevsel hale getirileceğini ve istasyonunuza OpenSky ağına nasıl bağlayacağınızı açıklar.

Yapılacak olan baz istasyonu, 200-300 km'ye kadar yarıçaptaki uçaklardan yayın sinyallerini alabilecek ve kodunu çözebilecek kadar etkili olabilmektedir [4].

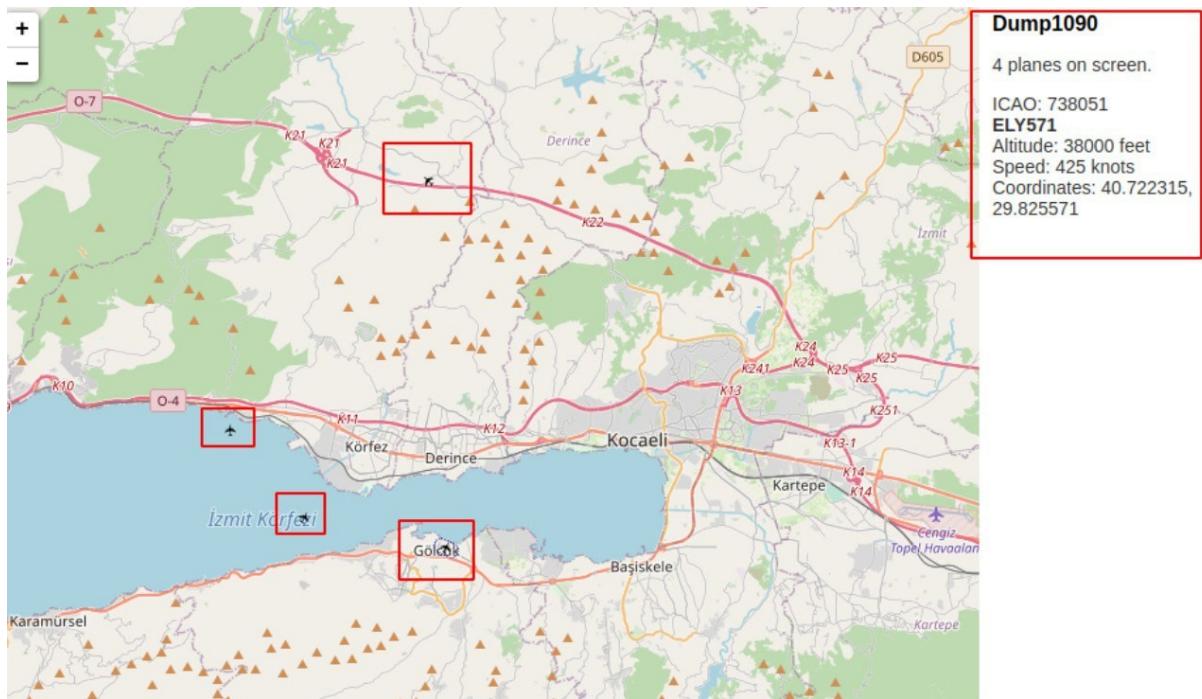
Şekil 2 görselinde bir uçağın uçuş rota hareketi örnek olarak paylaşılmıştır.



ŞEKİL 2: Dump1090 ile tarayıcı üzerinde alınan verinin görsel hali

Aynı zamanda verilerinizi OpenSky Network ve diğer ağlara aktarılabilen ve PlanePlotter gibi yerel uygulamalar üzerinden de kullanılabilecektir.

Şekil 3'te bir başka uçak rotası görseli örneği paylaşılmıştır.



ŞEKİL 3: Başka bir dump1090 ile alınan verinin görseli

Anlatılan yöntem, Raspberry Pi kılavuzu için dump1090 kullanan ADS-B'ye dayanmaktadır.

İnternet üzerinde veri aktarımı sağlamak için internet bağlantısı olan bir ortama ihtiyaç vardır.

1.7.2. Önemli Detaylar

- Antenin yerleştirileceği konum, gökyüzünü net görebilecek ve etrafı açık bir şekilde konumlandırılmış olması gerekmektedir.
- Topladığınız verileri OpenSky Network'üne göndermek istiyorsanız internet bağlantınızı yapıp veri iletmeniz gerekmektedir.
- OpenSky Network, baz istasyonunuza bağlamak için statik (sabit) bir ana bilgisayar adına veya IP adresine ihtiyaç duyar. Statik bir IP (Internet Protocol) adresiniz yoksa (genellikle olmaz), internet sağlayıcınızın DuckDNS.org veya No-IP.com gibi bir sağlayıcı kullanarak dinamik DNS'yi (Domain Name System) desteklemesi gereklidir.
- Raspberry Pi'yi interne kablolu olarak bağlayabilmeniz için modeminizin WIFI'yi (Wireless Fidelity: Kablosuz Ağ Bağlantısı) desteklemesi gereklidir.

1.8. Baz İstasyonu için Gerekli Donanımlar

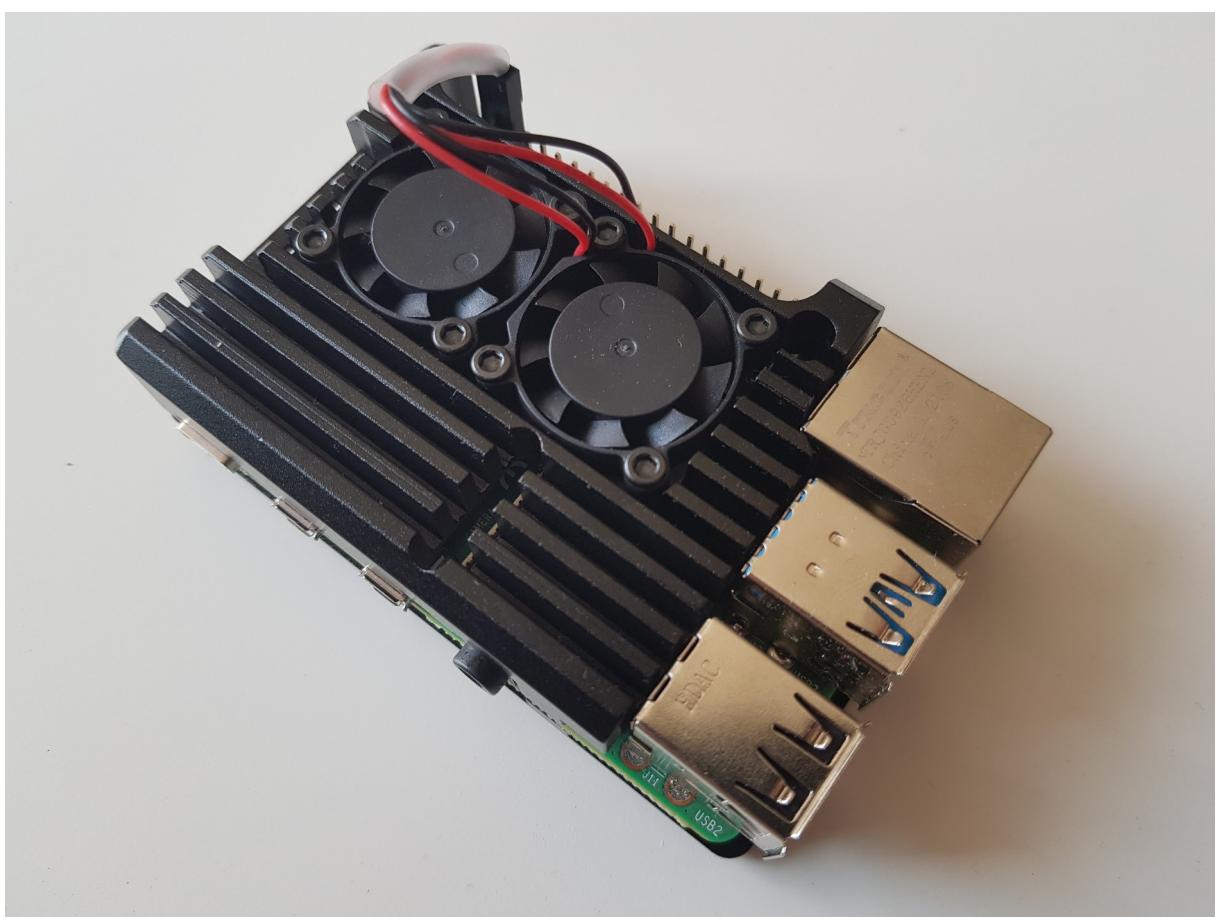
İstasyonun sinyalleri yakalaması ve o sinyalleri düzenlemesi için birtakım ekipmanlara ihtiyaç duymaktadır. Bu ekipmanlar yazılım kısmının çalışacağı Raspberry Pi ve alıcı ekipmanları olarak iki ana gruba ayrılıyor.

1.8.1. Raspberry Pi Ekipmanları

Raspberry Pi ekipmanları sistemin yazılım kısmından sorumlu bölüm oluyor.

1.8.1.1. Raspberry Pi 4 ve Çift Fanlı Metal Kasa Soğutma Sistemi

Şekil 4'te Raspberry Pi ile çift fanlı soğutma sistemi görseline yer verilmiştir.



ŞEKİL 4: Rasspberry Pi 4 2 gb ile çift fanlı soğutma sistemi

Gerekli yazılımları ve donanımlar arasında bağlantıyi sağlamak amaçlı minimal bir bilgisayar sistemi olan Raspberry Pi 4 bilgisayar kullandık. Aynı zaman donanım sürekli çalışacağı için soğutmasını sağlamak amacıyla metal kasa ve çift fanlı bir soğutma tercih ettiğimizdir.

1.8.1.2. Hafıza Kartı

Hafıza kartı için örnek oluşturmazı açısından şekil 5'teki görsel konulmuştur.



ŞEKİL 5: 32 gb ufak hafıza kartı

Bilgisayarın çalışır hale gelebilmesi için mini hafıza kart içerisinde özgür yazılım olan Linux çekirdekli ve GNU/Linux içerisinde barındıran Debian tabanlı dağıtım kullanmayı tercih ettim. Görsel alabilmek amaçlı hafıza kartının içerisinde arayüzü olan bir dağıtımın kurulumu yapıldı ama tercihe bağlı olarak donanımların daha az kaynak harcaması için minimal ve arayüzü olmayan başka Linux çekirdekli dağıtımlar kurulabilir.

1.8.1.3. Adaptör

Güç kaynağı için kullanılan adaptör şekil 6'da yer verilmiştir.



ŞEKİL 6: 5 volt 3.4 amperlik adaptör

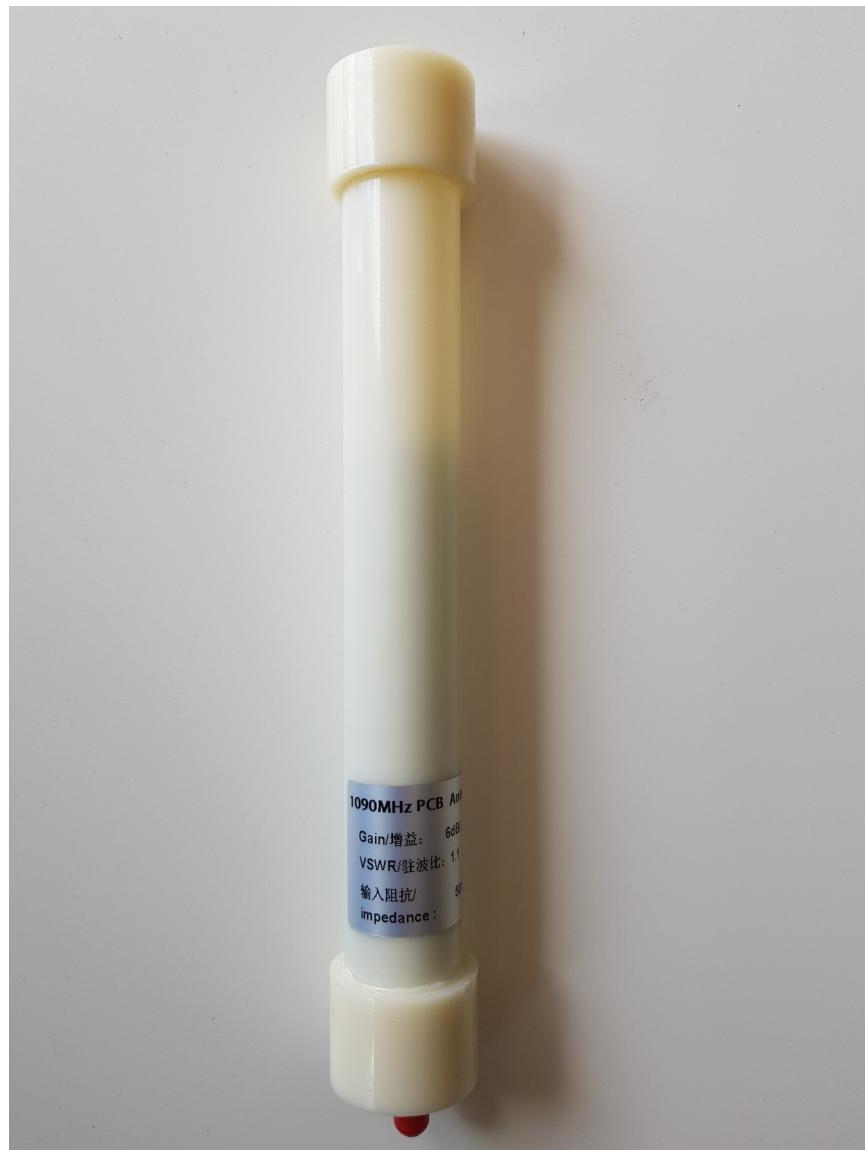
Raspberry Pi 4 cihazının çalışabilmesi önerilen adaptör özellikleri 5 volt ve 3 amper olmalıdır. Kullanılan adaptör ise 5 volt 3.4 amper.

1.8.2. Alıcının Ekipmanları

Uçaklar tarafından yayınlanan sinyalleri yakalamak için 1090 MHz anten, RTL-SDR alıcı ve koaksiyel kabloya ihtiyaç vardır.

1.8.2.1. Anten

Veri toplamak için kullanılan anten şekil 7 bölümünde gösterilmiştir.



ŞEKİL 7: 1090 MHz anten

Uçaklar tarafından yayınlanan radyo dalgalarını yakalayabilmek amaçlı gereklili olan bir donanımıdır. İnternet üzerinden araştırma yapılarak daha başka alternatiflerde kullanılabilir aynı zamanda basit şekilde bakır tel ile de anten yapmak mümkün.

1.8.2.2. RtI-Sdr

Anten tarafından toplanan verilerin szülmezi için kullanılan RTL-SDR donanıma şkil 8'de yer verilmiştir.



ŞEKİL 8: RTL-SDR

Anten tarafından yakalanan sinyallerin daha temiz bir şekilde elde edilmesi amaçlı RTL-SDR donanımına ihtiyaç vardır.

1.8.2.3. Her İki Ucu Sma Erkek Bağlantı Girişli Koaksiyel Bakır Kablo

Anten ile RTL-SDR donanımı arasında bağlantıyı sağlayabilmek amaçlı her iki ucu SMA erkek olan kablo görseline şekil 8'de yer verilmiştir.



ŞEKİL 9: 10 metrelük koaksiyel bakır kablo

Bunlara ek olarak hazır microSD veya SD kart okuyuculu bir bilgisayarınız, HDMI (High-Definition Multimedia Interface) bağlantılı monitörünüz ve kurulum için gereken USB (Universal Serial Bus) tuş takımı ve fareniz olduğunu ve ayrıca sipariş vermeniz gerektiğini varsayıyoruz.

1.9. Ayar Aşamaları

- Raspberry Pi'nin birleştirilmesi
- Ekipmanların birleştirilmesi
- Raspberry'nin ayarlanması
 - ◆ İşletim sistemi ile SD kartın hazırlanması
 - ◆ İşletim sisteminin kurulumunun gerçekleştirilmesi
 - ◆ RTL-SDR alıcısı için sürücüler oluşturma ve yükleme
 - ◆ dump1090 kod çözücüünün oluşturulması ve kurulumu
- Baz istasyonunu OpenSky Network'e bağlama
 - ◆ Raspberry Pi için dinamik (sabit olmayan) DNS kurulumu
 - ◆ Port (bağlantı noktası) 30005 bağlantı noktasını internete erişebilir hale getirilmesi
 - ◆ OpenSky Network hesabının oluşturulması
 - ◆ OpenSky Network'e yeni bir alıcı eklenmesi

1.9.1. Raspberry Pi'nin Birleştirilmesi

Raspberry Pi üzerinde bulunan vida bölümüne çift fanlı soğutma sistemi vidalar yardımı ile bağlanıyor. Duruma göre eğer sisteminizin ısınması fazla olmayacağı tek fanlı soğutma sistemi de kullanılabilir.

Fan üzerinde bulunan kırmızı ve siyah kablolar Raspberry üzerinde bulunan vida çıkışlarına bağlanarak fan ile Raspberry arasındaki elektriksel bağlantı sağlanmış olur.

Raspberry cihazının gerekli yazılımlarının olduğu kısmı olan hafiza kartı, Raspberry Pi üzerinde bulunan hafiza kartı giriş bölümüne bağlanır.

En son olarak Raspberry Pi cihazının elektrik alabilmesi için adaptör bağlanır ve Raspberry cihazının çalışması için gerekli olan kısımlar tamamlanmış olur.

1.9.2. Alıcı Ekipmanlarının Birleştirilmesi

Antenin alt kısmında bulunan dişi giriş bölümü ile kablonun ucunda bulunan erkek giriş bölümü birleştirilir. Anten ile de RTL-SDR donanımı arasında bağlantı sağlayabilmek amaçlı kablonun diğer ucuya RTL-SDR cihazı birbirine bağlanır.

Bu bağlantılar da sağlandıktan sonra Raspberry Pi ile alıcılar arasında bağlantı sağlayabilmek amaçlı RTL-SDR cihazının diğer ucu Raspberry Pi üzerinde bulunan 4 bölümden herhangi birine bağlanır.

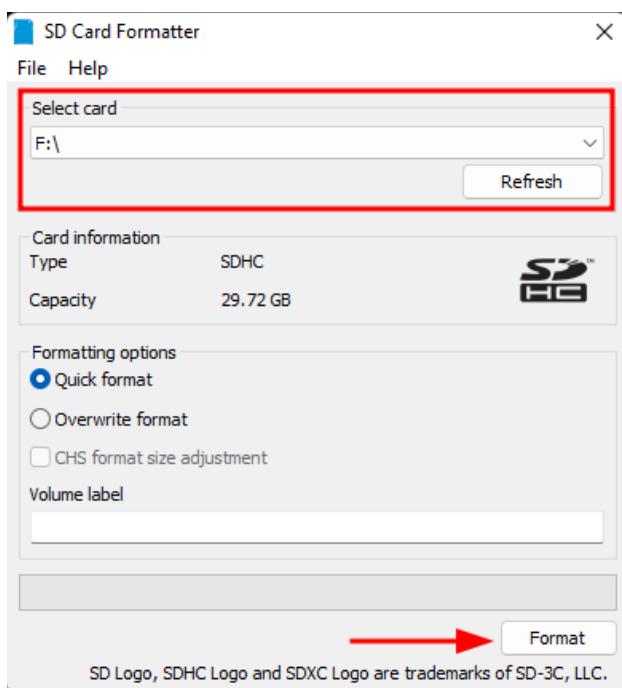
Bu işlemde yapıldıktan sonra donanımların bağlantısı tamamlanmış oluyor. Geriye ise Raspberry Pi içerisinde çalıştırılacak yazılımlar yüklenmelidir.

1.9.3. Raspberry Pi Cihazının Yazılım Bölümünün Ayarlanması

Raspberry cihazının çalışabilmesi için içerisinde herhangi bir işletim sisteme ihtiyaç duyar bu yüzden gerekli ayarlamaların yapılması gerekmektedir.

1.9.3.1. İşletim Sistemi ile Sd Kartın Hazırlanması

Eğer kurulum yapacağınız disk içerisinde daha öncesinde herhangi bir işletim sistemi varsa diski temizlemek için SD Memory Card Formatter sitesinde kendi işletim sisteminize göre programı indirdikten sonra kurulumunu yapmalısınız. Programı çalıştırdıktan sonra karşınıza şékil 10'daki gibi bir ekran gelecektir.



ŞEKİL 10: SD Card Formatter programının arayüzü

Select card bölümünden diskı seçiyoruz ve *Format* bölümünden de diskin formatlanması sağlıyoruz.

Ek olarak *İşletim sisteminin kurulumunun gerçekleştirilmesi* bölümünde anlatılan Raspberry Pi Imager programı yardımı ile de diskinize format atabilirsiniz.

1.9.3.2. İşletim Sisteminin Kurulumunun Gerçekleştirilmesi

Raspberry Pi 4 içerisinde işletim sisteminin kurulumunu yapabilmek amaçlı Raspberry Pi OS sitesinden Raspberry Pi Imager programını bilgisayarımızın işletim sistemine göre indirip kurulumunu yapıyoruz. Kurulumunu yaptıktan sonra şekil 11'de göründüğü gibi bir ekran karşınıza gelecektir.



ŞEKİL 11: Raspberry Pi disk yazıcısının arayüzü

CHOOSE OS bölümünden kurulumunu yapmak istediğimiz işletim sistemini seçiyoruz. *CHOOSE STORAGE* bölümünden kurulumu yapmak istediğimiz diskı seçiyoruz. Daha sonrasında *WRITE* bölümüne basarak işlemi onaylıyoruz. Bundan sonra yazılım işletiminin sistemini kurulumunu kendisi yapmaktadır.

1.9.3.3. Rtl-Sdr Alıcısı için Sürücüler Oluşturma ve Yükleme

Raspberry Pi OS veya başka herhangi bir Debian tabanlı dağıtımın kurulumunu yaptıktan sonra. RTL-SDR alıcısının düzgün çalışması için bazı yazılımların kurulumu gerekmektedir.

Öncelikle sistemin güncellemesini tamamlıyoruz.

```
sudo apt-get update  
sudo apt-get upgrade
```

Güncellemeye işlemi tamamlandıktan sonra, git üzerinden klonlama yapabilmek için git kurulumunu yapıyoruz.

```
sudo apt-get install git-core git
```

Git kurulumunu yaptıktan sonra OpenSky-Network tarafından hazırlanan Bash kodlarını ana dizine indiriyoruz.

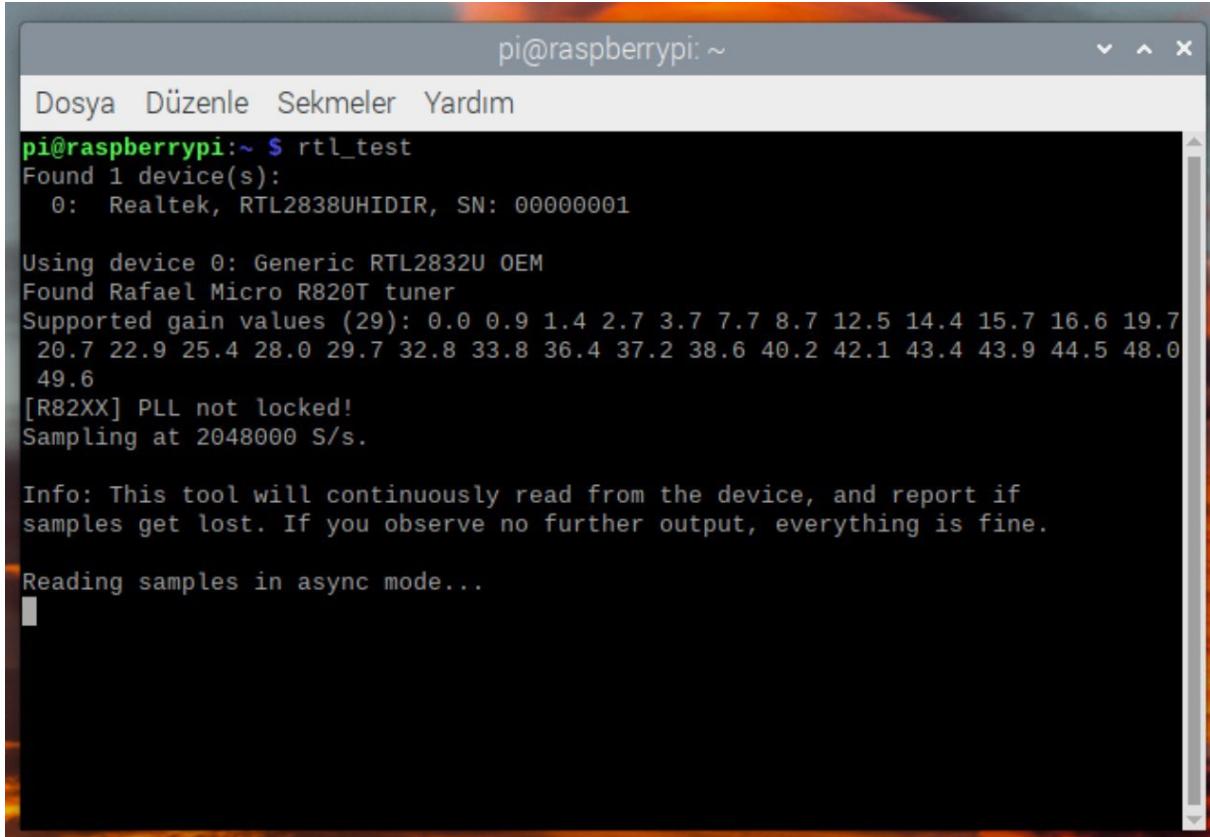
```
cd ~  
git clone https://github.com/openskynetwork/raspberry-pi-adsb.git  
chmod +x ~/raspberry-pi-adsb/*.sh
```

Şimdi sıra RTL-SDR sürücülerini kurmada.

```
cd ~/raspberry-pi-adsb  
./setup-rtl-sdr.sh
```

Bu işlemi de yaptıktan sonra sürüplerin kurulmuş olması gerekiyor. Test yapmak için aşağıdaki komutu terminal üzerinden çalıştırıyoruz.

```
rtl_test
```



```
pi@raspberrypi: ~
Dosya Düzenle Sekmeler Yardım
pi@raspberrypi:~ $ rtl_test
Found 1 device(s):
  0: Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7 7.7 8.7 12.5 14.4 15.7 16.6 19.7
 20.7 22.9 25.4 28.0 29.7 32.8 33.8 36.4 37.2 38.6 40.2 42.1 43.4 43.9 44.5 48.0
 49.6
[R82XX] PLL not locked!
Sampling at 2048000 S/s.

Info: This tool will continuously read from the device, and report if
samples get lost. If you observe no further output, everything is fine.

Reading samples in async mode...
```

ŞEKİL 12: Terminal üzerinde RTL-SDR sürücü testinin terminal görüntüsü

Şekil 12'deki gibi bir ekran gelmesi gerekiyor. Eğer böyle bir sonuç aldıysanız sorunsuz şekilde RTL-SDR sürücülerini çalışıyor demektir.

1.9.3.4. Dump1090 Kod Çözüğünün Oluşturulması ve Kurulumu

Gelen sinyallerin çözülmesi için dump1090 yazılımının kurulumu için OpenSky-Network tarafından hazırlanan " setup-dump1090.sh" dosyasını çalıştırıyoruz.

```
cd ~/raspberry-pi-adsb
./setup-dump1090.sh
```

Yukarıda komut ile dump1090 yazılımının kurulumu sağlamış oluyoruz.

Kurulum sonucunda sonuçlarımızı görmek için terminale alttaki komutu çalıştırıyoruz.

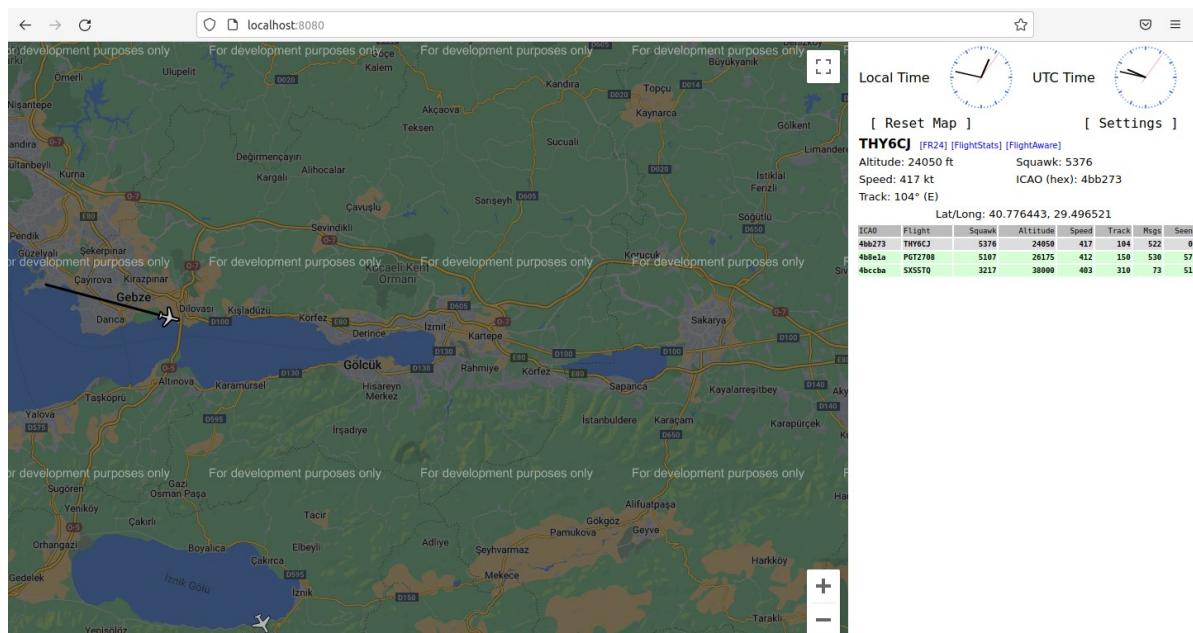
```
./dump1090 -interactive -net
```

Komutu çalıştırdıktan sonra terminalde uçakların verileri şekil 13'teki gibi yansıyacaktır.

| Hex | Flight | Altitude | Speed | Lat | Lon | Track | Messages | Seen | . |
|--------|--------|----------|-------|--------|--------|-------|----------|------|--------|
| 896214 | | 32600 | 0 | 0.000 | 0.000 | 0 | 15 | | 6 sec |
| 4baada | THY6TH | 13650 | 345 | 40.869 | 29.825 | 269 | 62 | | 16 sec |

ŞEKİL 13: Dump1090 yazılımının çalışması sonucu terminalde oluşan görüntü

Terminale ek olarak tarayıcı üzerinden kontrol amaçlı <http://127.0.0.1:8080> ya da <http://localhost:8080> üzerinden yerel bölümü açtığınız zaman şekil 14'teki gibi bir harita ile karşılaşırınsınız.



ŞEKİL 14: Dump1090 yazılımının yerel tarayıcı üzerinde oluşturduğu görüntü

1.9.4. Opensky Network Hesabının Oluşturulması ve Yeni Bir Alıcının Eklenmesi

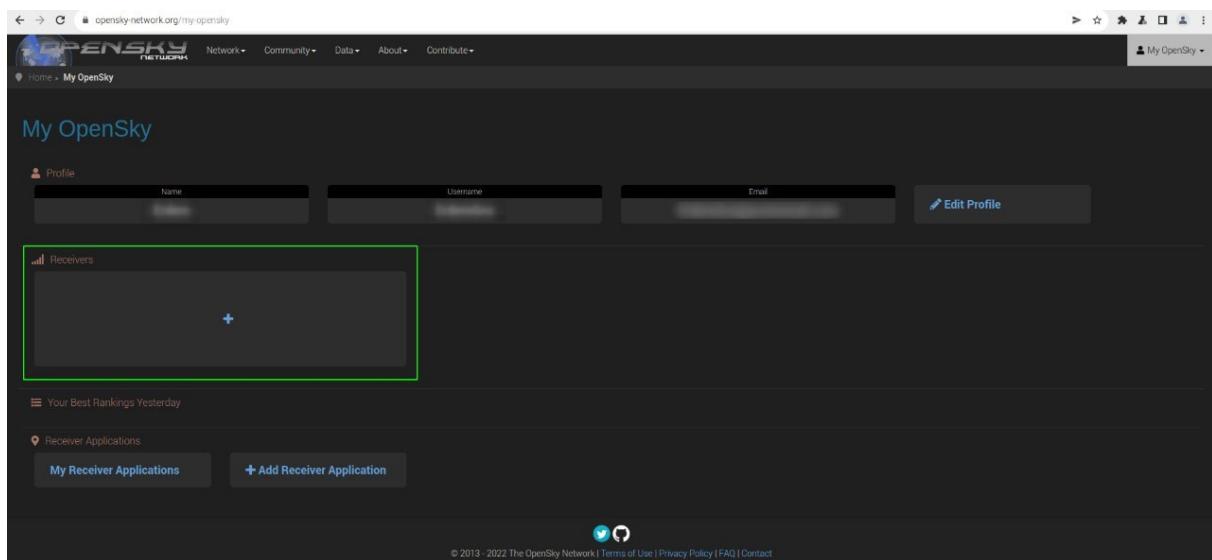
OpenSky-Network internet sayfasında *Register* bölümüne giderek.

The screenshot shows the 'User Registration' form on the OpenSky-Network website. The form is titled 'User Registration' and includes fields for Name*, Username*, Password*, Confirm Password*, and Subscription. Below these are optional fields for Email Address*, Confirm Email Address*, and More Info, which include fields for Institution (optional), Address (optional), Phone Number (optional), Postal Code (optional), City (optional), and Country (with a dropdown menu). At the bottom, there is a checkbox for agreeing to the Terms of Use, with options to Agree or No, followed by 'Register' and 'Cancel' buttons.

ŞEKİL 15: OpenSky-Network hesap oluşturma bölümünün ekran görüntüsü

Şekil 15'teki alanları doldurarak hesabımızı oluşturuyoruz.

Hesabımızı oluşturduktan sonra donanımı eklemeye sıra geliyor. My OpenSky bölümünü giriyoruz ve şekil 16'daki *Receivers* içerisinde giriyoruz.



ŞEKİL 16: OpenSky-Network'te My OpenSky bölümünün ekran görüntüsü

Receivers bölümü içerisinde girdikten sonra şekil 17 ve şekil 18'deki alanları eksiksiz dolduruyoruz.

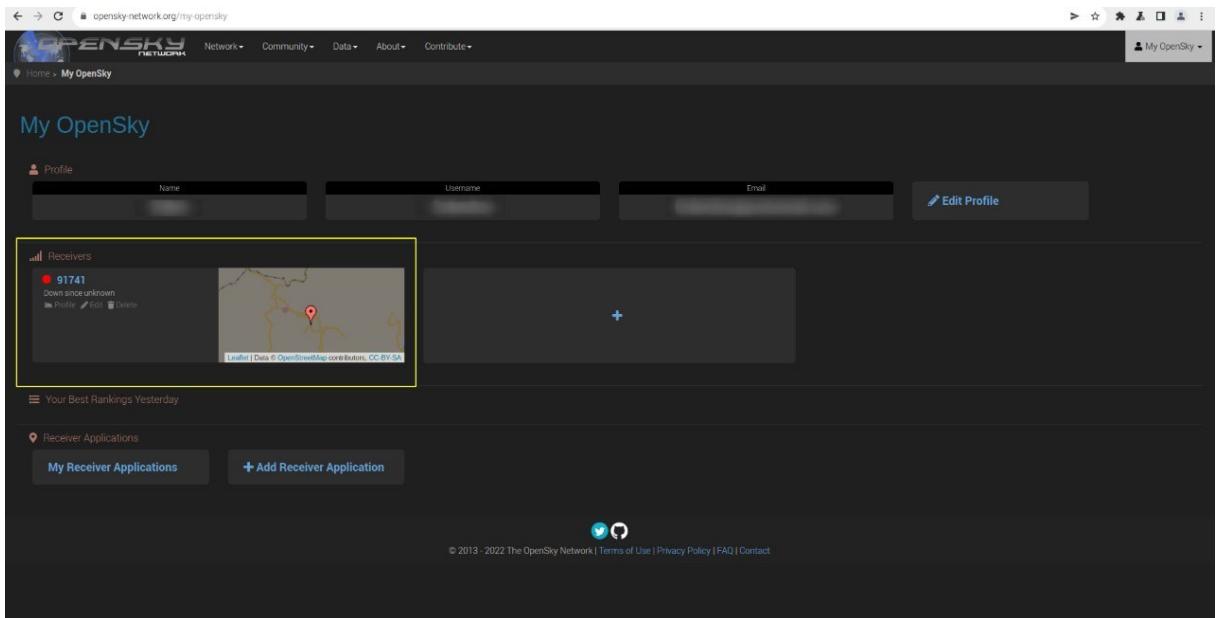
The screenshot shows the 'Add/Edit Receiver Setup' page for the 'Hardware' section. The 'Receiver Type' dropdown is set to 'dump1090'. A warning message states: 'Warning: The preferred mode of operation is our feeder! See [here](#) for instructions on how to install or use it. With our feeder you do **NOT** need to register your receiver. It will be added to your profile automatically. Moreover, you do not need to open a port on your firewall. Register your receiver [here](#) only if you want to stick with your plain dump1090 installation.' Below this, it says: 'We support the following forks of dump1090:'. A list follows: '• https://github.com/opensky-network/dump1090-hotos', '• MalcolmRobb: <https://github.com/MalcolmRobb/dump1090>', and '• mutability: <https://github.com/mutability/dump1090>'. A note below states: 'Most Raspberry Pi images are using the mutability fork.' The 'URL/IP address' field contains '192.168.1.100'. The 'Port' field is set to '30005'. The 'Location' section shows the address 'Izmit/Kocaeli, Türkiye' entered into the 'Address' field, with a 'Set on Map' button next to it.

ŞEKİL 17: Cihaz kayıt ekranının ilk yarısı

The screenshot continues from the previous one, showing the 'Location' and 'Privacy' sections. In the 'Location' section, there is a large map placeholder with the text: 'Please use the following fields to refine the coordinates of the antenna of your receiver. The coordinates should be accurate to the 4th decimal place.' Below the map are three input fields: 'Longitude' (29), 'Latitude' (40), and 'Elevation (in ft)' (1745). In the 'Privacy' section, there is a checkbox labeled 'Anonymize'. A note explains: 'The location of your receiver is stored along with its data on our server. That means, if we hand out raw data to researchers, it may contain the coordinates of your receiver. If you wish to conceal the location of your receiver, please tick this box. In that case, we will remove the location when handing out raw data. In any case, we never give out names or identities of our feeders!' Another note states: 'Also please note that despite the fact that this is prohibited by our terms of use, it is fundamentally not possible to fully prevent reverse engineering of the receiver location based on its data. For example, the coverage of a receiver already provides some indication of its location.' At the bottom, there are 'Submit' and 'Cancel' buttons.

ŞEKİL 18: Cihaz kayıt ekranının sonraki yarısı

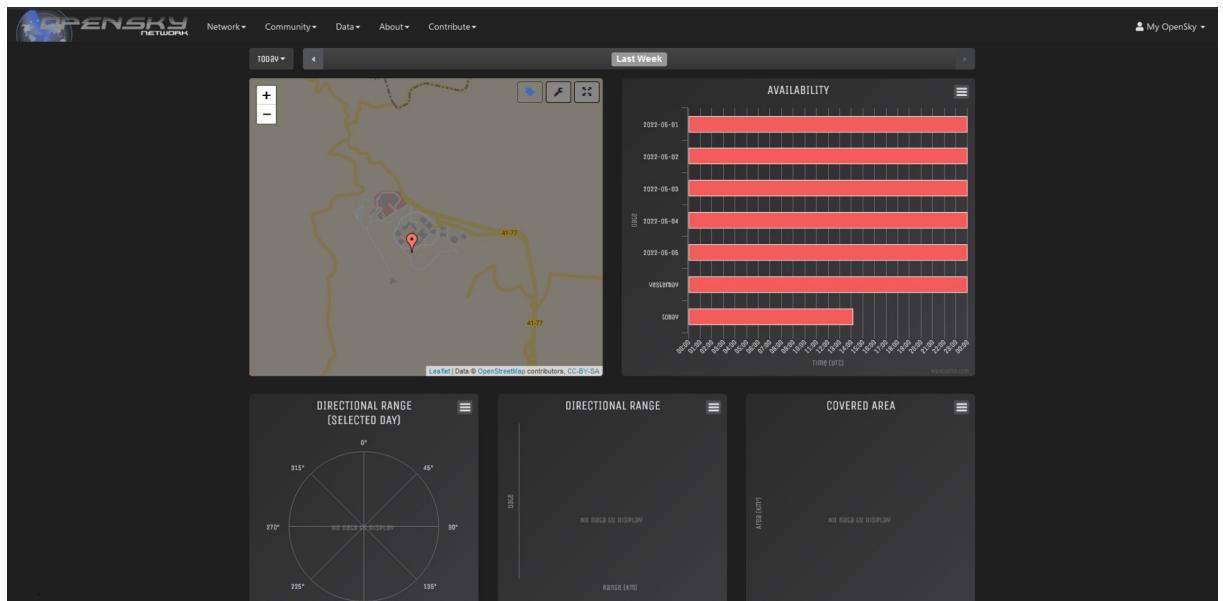
Boş alanlar doğru şekilde doldurulduktan sonra Submit kısmından bilgilerin doğruluğunu onaylıyoruz ve işlem tamamlanmış oluyor.



ŞEKİL 19: Cihazı kayıt ettikten sonraki My OpenSky bölümünün ekran görüntüsü

Submit işleminden sonra *Receivers* bölümü içerisinde şekil 19'da gösterilen donanımızın kayıt bilgileri gelmiş oluyor.

Donanım bölümünüze girdikten sonra donanımızın aldığı bilgileri şekil 20'deki gibi ayrıntılı şekilde görebilirsiniz.



ŞEKİL 20: Kayıtlı cihazın bölümüğe girildiğinde çıkan ekran

Bu işlem de tamamlandıktan sonra artık OpenSky-Network üzerinde veri paylaşımı sağlayan bir donanıma sahip olmuş olacaksınız.

2. UYGULAMALAR

Kurulum tamamlandıktan sonra veriler Açık Hava Trafik Verileri kullanarak, Python, JavaScript ile web ortamı ve QGIS ile Türkiye hava sahasına giren uçaklar için analizler yazılmıştır.

2.1. Python Jupyter Notebook ile Yapılan Uygulama

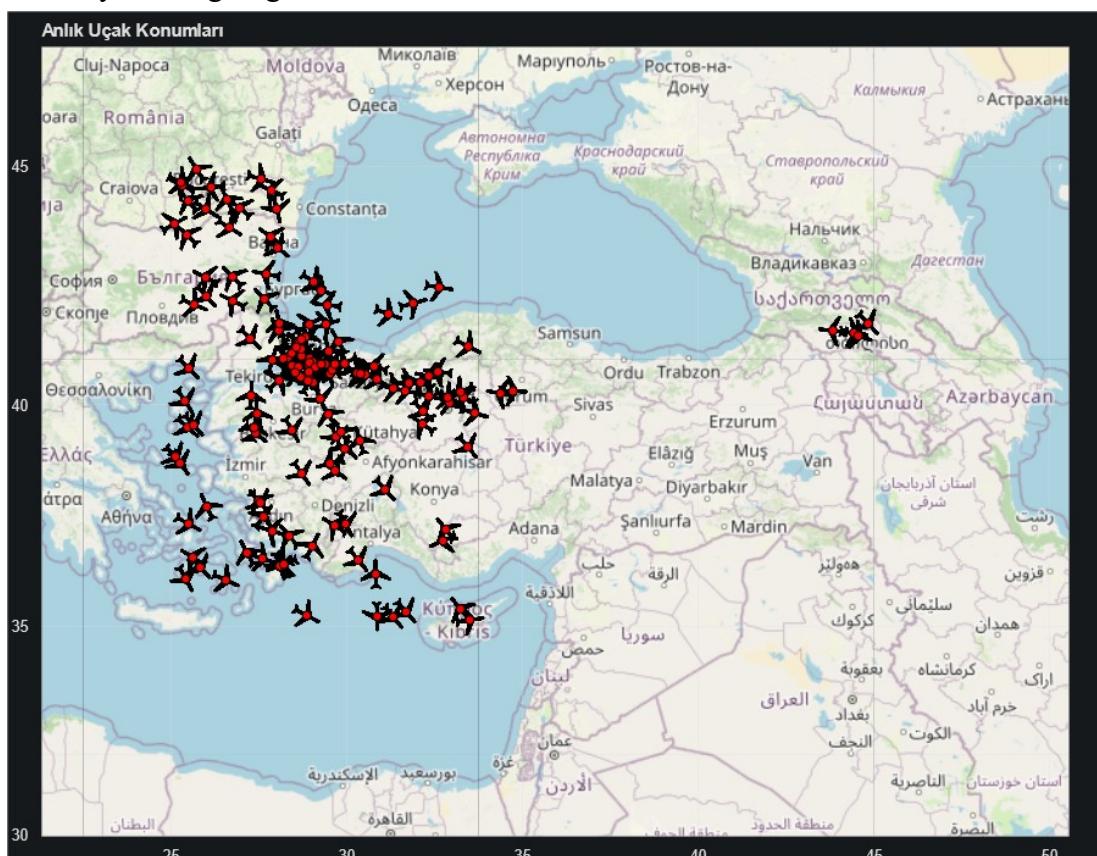
OpenSky üzerinden çekilen verilerin Python Jupyter Notebook ile görselleştirilmesinden nasıl bir sonuç çıkartılabilceği ile ilgili örnek uygulamaya bakalım.

2.1.1. Python Jupyter Notebook ile Veriyi Görselleştirme

Veriyi görselleştirmek için Jupyter Notebook üzerinde EK 1 içerisinde yer alan kodlar çalıştırılmıştır.

2.1.2. Jupyter Notebook Üzerinde Python ile Sonuç

Kodu çalıştırdıktan sonra şekil 21'deki gibi bir bölüm açılır ve uçaklar görülecektir. Kırmızı noktaların üzerine fareyi getirdiğiniz zaman uçakları ile ilgili ülkesini, uçak adını, hızını ve barometrik yüksekliğini görebilirsiniz.



ŞEKLİ 21: Jupyter Notebook üzerinde Python kodunu çalıştırıldıktan sonra oluşan bölümün anlık ekran görüntüsü

2.2. Javascript ile Web Sitesi Yaparak Online Görselleştirme

Web sitesi üzerinde online olarak verinin görselleştirilmesi yapılmıştır. JavaScript programlama dilinin yanında HTML ve CSS biçimlendirme dilleri de kullanılmıştır.

2.2.1. Html Kodları

HTML yardımı ile web ortamını aşağıdaki koddaki gibi hazırlıyoruz.

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
    <meta charset="utf-8" />

    <link
        rel="stylesheet"
        href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
        integrity="sha512-
xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/
keqq/sMZMZ19scR4PsZChSR7A=="
        crossorigin=""
    />
    <script
        src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
        integrity="sha512-
XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaX
aVUearIOBhiXZ5V3ynxwA=="
        crossorigin=""
    ></script>
    <script src="leaflet.rotatedMarker.js"></script>

    <link rel="stylesheet" href="style.css" />

    <title>Airplane</title>
    <link rel="shortcut icon" href="airplane.png" type="image/x-icon" />
</head>

<body>
    <h1>Anlık Uçak Hareket Haritası</h1>

    <hr>

    <div id="map"></div>

    <script src="script.js"></script>
</body>
</html>
```

2.2.2. Css Kodları

Haritanın görselleştirilmesi için CSS biçimlendirme dilinden aşağıdaki gibi yardım alınabilir.

```
#map {  
    width: 70%;  
    height: 600px;  
    border: 5px solid green;  
    margin: 20px;  
}
```

2.2.3. Javascript Kodları

JavaScript programlama dili yardımı ile sürekli olarak yeni veriler oluşturulmasını aşağıdaki kod ile sağlandı.

```
const map = L.map('map').setView([40, 35], 5);  
const attribution = '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributonrs';  
const tileUrl = 'https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png';  
const tiles = L.tileLayer(tileUrl, {  
    attribution  
});  
tiles.addTo(map);  
const api_url = 'https://opensky-network.org/api/states/all?  
lamin=35.8389&lomin=25&lamax=45&lomax=45';  
  
function fetchData() {  
    return fetch(api_url)  
        .then((res) => {  
            return res.json();  
        })  
        .then((res) => {  
            return res.states.filter((state) => {  
                return (state[5]) && (state[6]);  
            });  
        })  
        .catch((err) => {  
            if (err) throw err  
        });  
}  
  
function plotStates(map, markers) {  
    fetchData().then(function (states) {  
        states.forEach((state) => {  
            const  
                icao24 = state[0],  
                callsign = state[1],  
                origin_country = state[2],  
                time_position = state[3],  
                last_contact = state[4],
```

```

    lng = state[5],
    lat = state[6],
    baro_altitude = state[7],
    on_ground = state[8],
    velocity = state[9],
    true_track = state[10],
    vertical_rate = state[11],
    sensors = state[12],
    geo_altitude = state[13],
    squawk = state[14],
    spi = state[15],
    position_source = state[16];

    if (markers[icao24]) {
        markers[icao24].setLatLng([lat, lng]);
    } else {
        const airplaneIcon = L.icon({
            iconUrl: 'airplane.png',
            iconSize: [17],
        });

        //Uçakların gidiş yönü açısı => true_track
        markers[icao24] = L.marker([lat, lng], {
            icon: airplaneIcon,
            rotationAngle: true_track
        }).bindPopup('<p> Ülke: ${origin_country} <br> Enlem: ${lat} <br> Boylam: ${lng} <br> Barometrik Yükseklik: ${baro_altitude} m</p>');
    }

    markers[icao24].addTo(map);
}

});

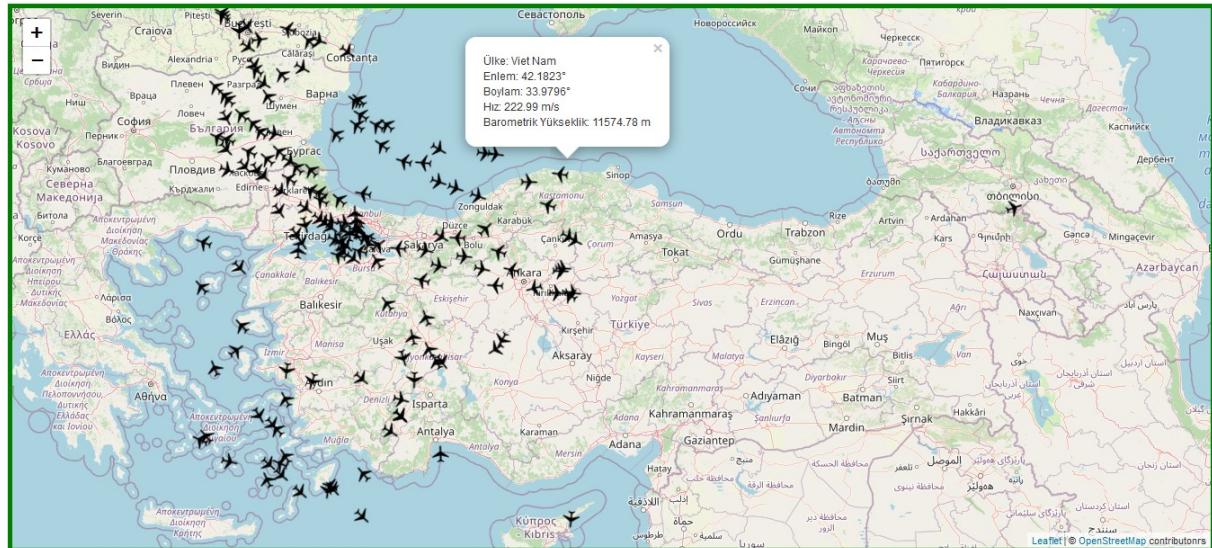
setTimeout(() => plotStates(map, markers), 5000);
};

const markers = {};
plotStates(map, markers);

```

2.2.4. Html, Css ve Javascript ile Yapılan Web Sitesi Sonuç

Tez çalışması kapsamında hazırlamış olduğumuz <https://erdemgns.github.io/flightaware/> internet adresini ziyaret ederseniz Türkiye hava sahasına giren uçakları çevrim içi gerçek zamanlı olarak şekil 22'deki gibi ekran görüntüsü olacaktır. Uçakların üzerine tıklandığı zaman uçakların ülkesi, enlemi, boylamı, hızı ve barometrik yükseklik bilgileri açıklmaktadır.



ŞEKİL 22: Web sitesinden alınan anlık ekran görüntüsü

2.3. Qgis Üzerinde Canlı Veri ile Yapılan İşlemler

QGIS üzerinde canlı şekilde işlem yapabilmek için verilerin belirli aralıklarla çekilmesi gerekmektedir.

2.3.1. Verilerin Python Yardımı ile Belirli Aralıklarda Sunucudan İstenmesi

Aşağıdaki kodlar dizin içerisinde main.py isimli dosya içerisinde kayıt edilmiştir.

```
""" Kütüphaneler """
import requests
import csv
import time
""" Türkiye'yi içine alan coğrafi koordinatlar """
lon_min,lat_min = 25, 35
lon_max,lat_max = 45, 45
# CSV çıkış dosyası
csv_data='data.csv'

# REST API'den veri isteğinde bulun
user_name=""
password=""
url_data='https://'+user_name+':'+password+'@opensky-network.org/api/states/all?'+lamin='+str(lat_min)+'&lomin='+str(lon_min)+'&lamax='+str(lat_max)
+'&lomax='+str(lon_max)
```

```

col_name=['icao24','callsign','origin_country','time_position','last_contact','long','lat','baro_altitude','on_ground','velocity',
'true_track','vertical_rate','sensors','geo_altitude','squawk','spi','position_source']

# Gerekli yetki koşullarını sorgulama
if user_name != " and password != "":
    sleep_time=5
else:
    sleep_time=10

# Çekilen verinin CSV içerisinde depolaması
while col_name != []:
    with open(csv_data,'w') as csv_file:
        csv_writer=csv.writer(csv_file,delimiter=',',quotechar="",quoting=csv.QUOTE_ALL)
        csv_writer.writerow(col_name)
        response=requests.get(url_data).json()

    try:
        n_response=len(response['states'])
    except Exception:
        pass
    else:
        for i in range(n_response):
            info=response['states'][i]
            csv_writer.writerow(info)
    time.sleep(sleep_time)
    print('Get',len(response['states']),'data')

```

2.3.2. Verilerin Python Yardımı ile Belirli Arahlarda Sunucudan İstenmesi

Python kodunu terminal üzerinden kodun bulunduğu dizine giderek aşağıdaki komut ile çalıştırıyoruz.

```
>>python .\main.py
```

Çalıştırdıktan sonra şu aşağıdaki gibi cevap dönecektir.

```

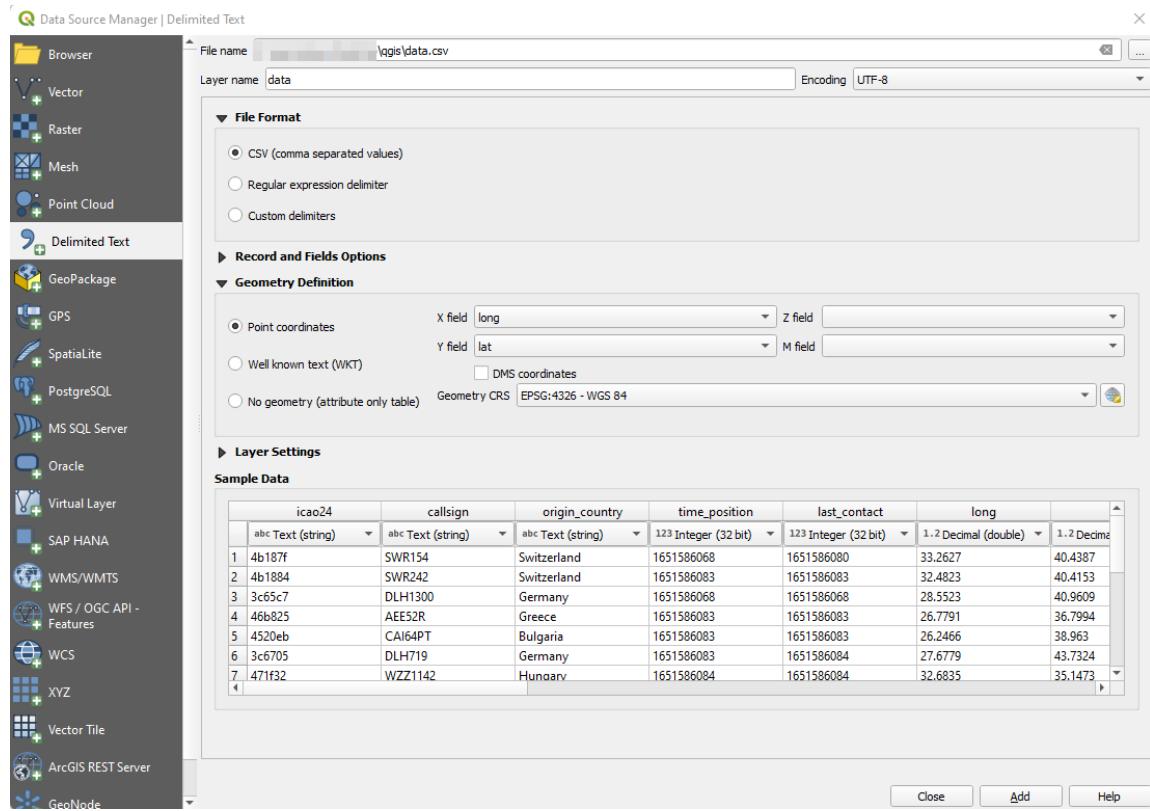
>>python .\main.py
Get 193 data
Get 194 data
Get 196 data
Get 195 data
Get 196 data
Get 196 data
Get 198 data
...

```

Kod çalıştığı sürece eğer OpenSky hesabınız varsa 5 saniyede bir yoksa her 10 saniyede bir siz durdurmadığınız sürece veri çekecektir.

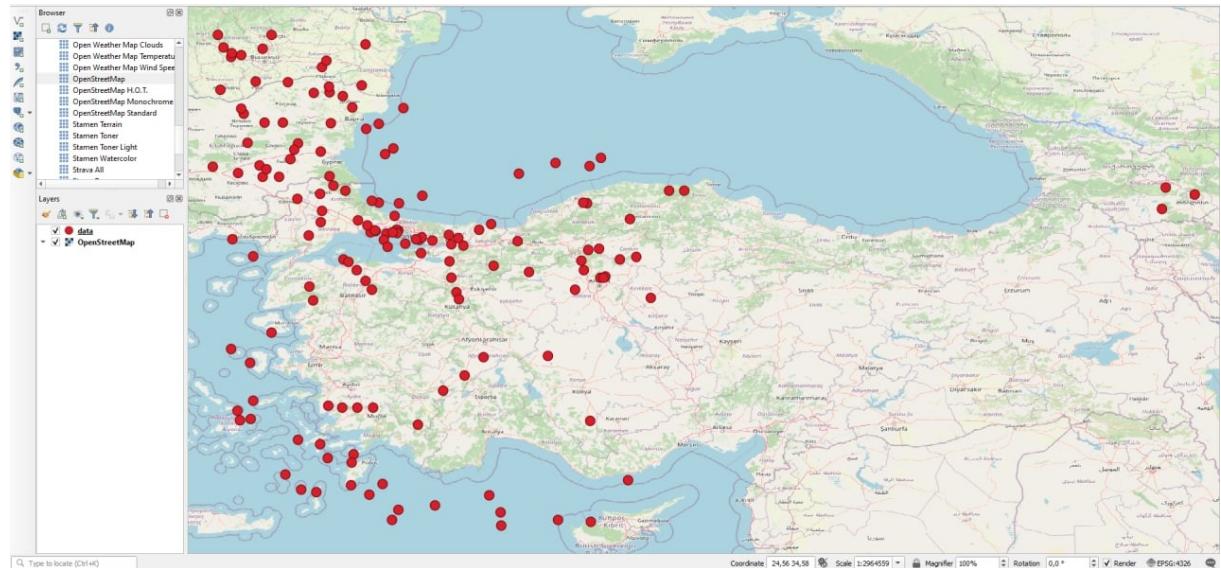
2.3.3. Verilerin Qgis Üzerinde Görselleştirilmesi

Verileri kayıt ettiğimiz data.csv dosyasını QGIS içerisinde bulunan *Delimited Text* yardımı ile şekil 23'teki gibi ayarlıyoruz. *Add* kısmına basarak verileri QGIS içerisinde ekliyoruz.

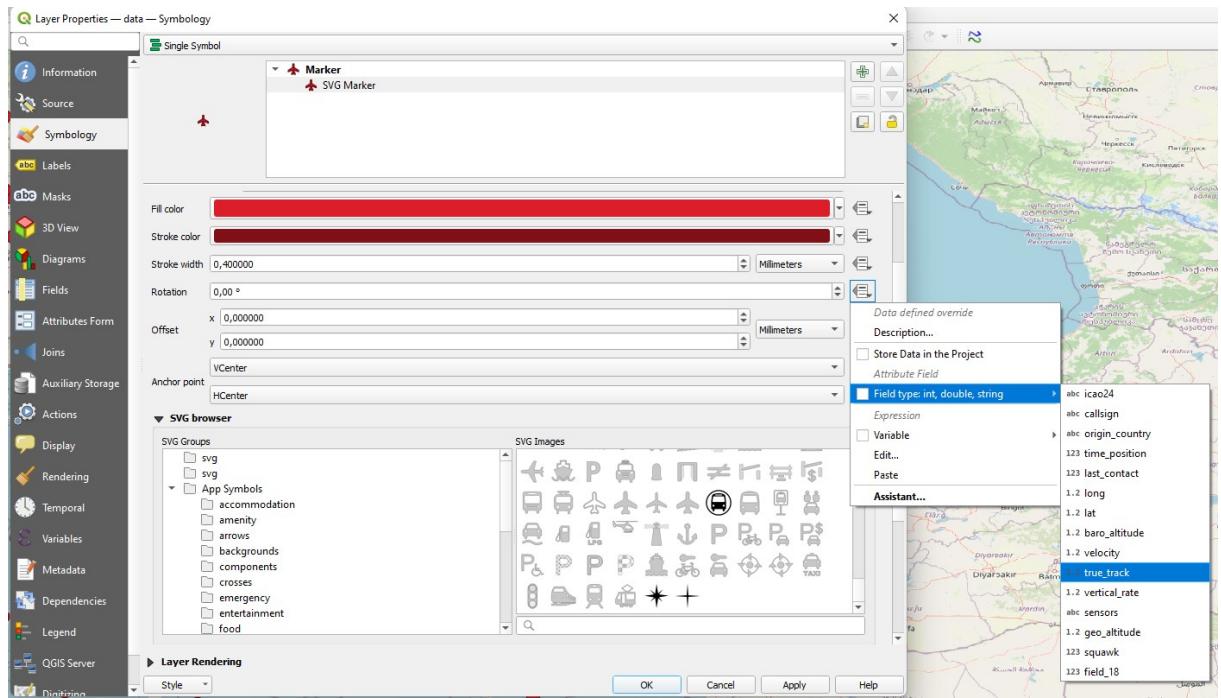


ŞEKİL 23: QGIS veri yönetimi bölümünün ekran görüntüsü

OpenStreetMap harita allığını da ekledikten şekil 24'teki gibi bir ekran oluşacaktır.



ŞEKİL 24: Verilerin eklenmesinden sonra oluşan ekranın görüntüsü



ŞEKİL 25: Katman ayarı bölümünde şekillerin ayarının yapıldığı bölümün ekran görüntüsü

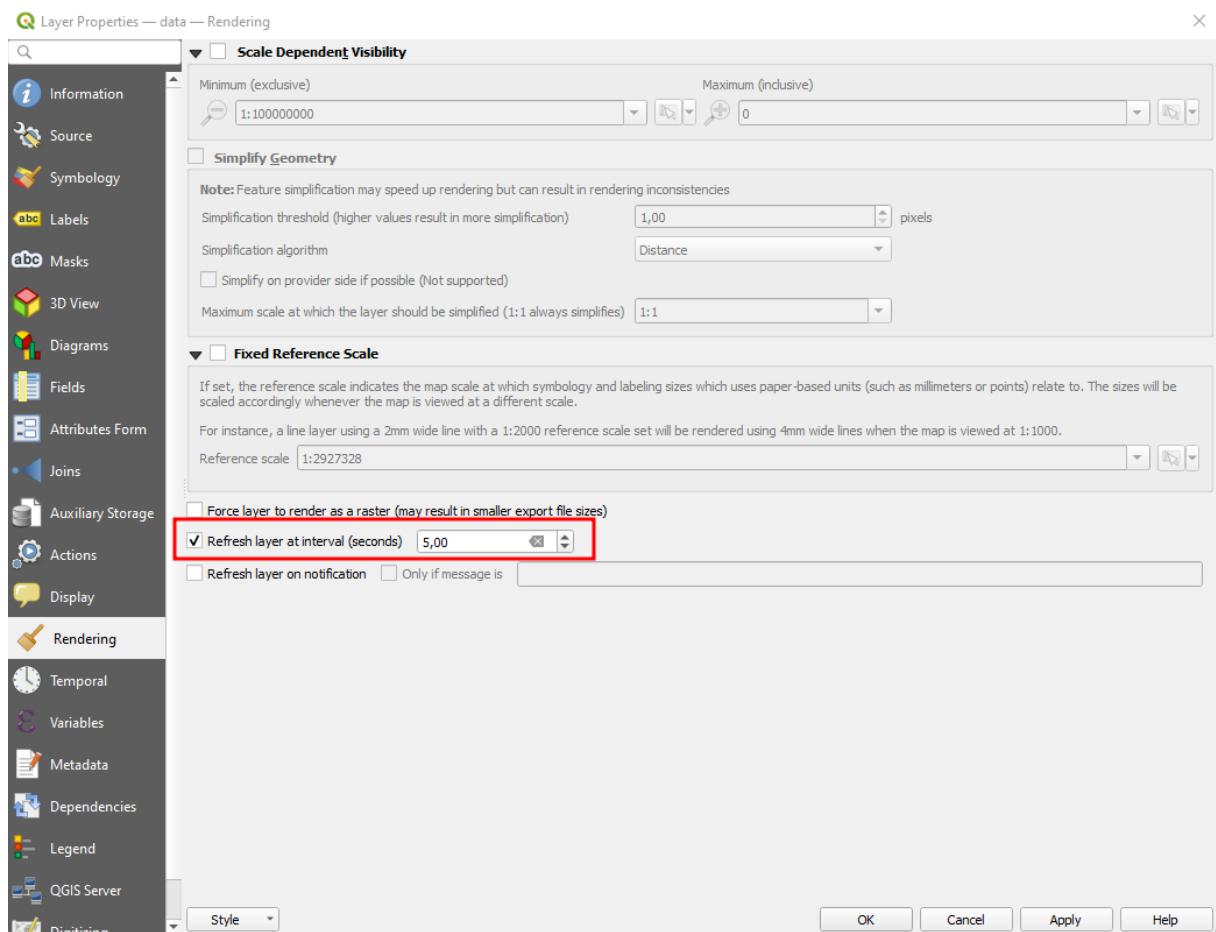
Noktaların uçak şeklinde ve uçakların gidiş yönünün olması için gerekli ayarları şekil 25'teki gibi yapıyoruz.

Bu işlemleri de yaptıktan sonra şekil 26'daki gibi bir görsel oluşacaktır.



ŞEKİL 26: Katman ayarlarında şekillerin düzenlenmesinden sonra yeni şekillerin ekran görüntüsü

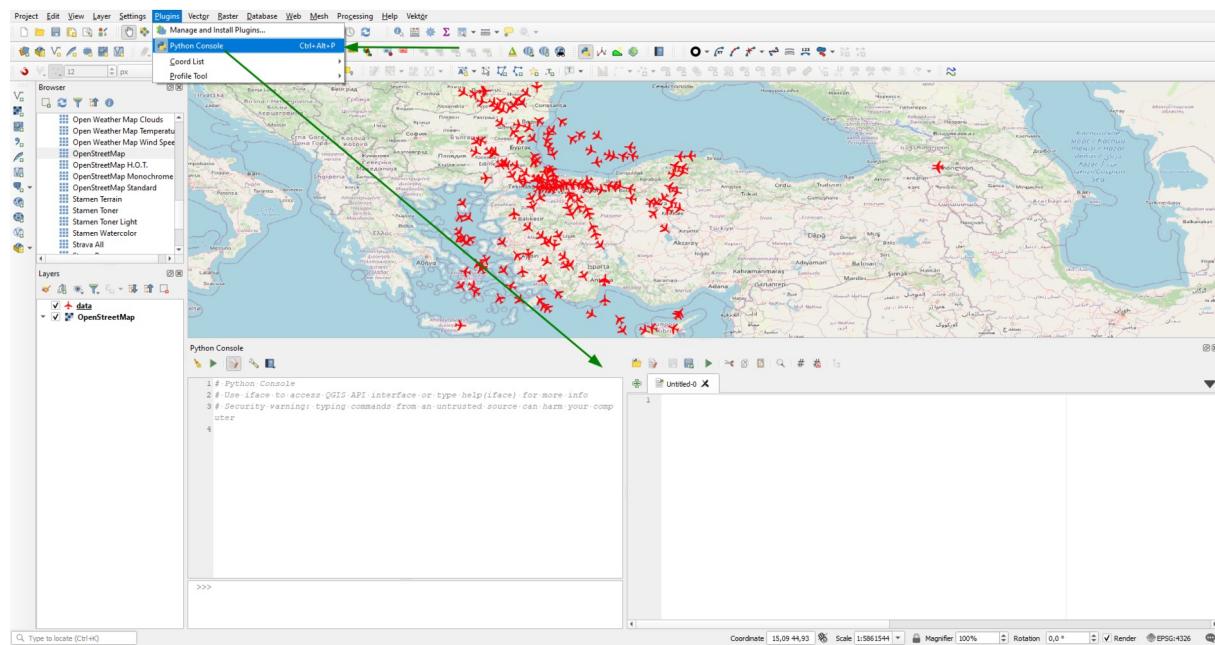
Veri görsellerinin her 5 saniyede bir yenilenmesi için görsel ayarı bölümünden *Rendering* kısmında bulunan *Refresh layer at interval* bölümünü aktif ediyoruz ve kutu içerisinde 5 yazıp şekil 27'deki gibi yapıyoruz ve *OK* kutucuğu ile işlemi onaylıyoruz.



ŞEKİL 27: Katman ayarları bölümde bulunan katman yenilenme süresi ayarının yapıldığı kısmın ekran görüntüsü

2.3.4. Verilerin Qgis Üzerinde Python Console Yardımı ile Grafiksel Görselleştirilmesi

Verilerin hangi ülkelere ait olduğunu dair görsel bir grafik oluşturmak istersek QGIS içerisinde bulunan *Python Console* bölümünden yararlanabiliriz. *Python Console* bölümünü şekilde 28'deki gibi açıyoruz.



ŞEKİL 28: QGIS içerisinde Python Console bölümünün açılış ekran görüntüsü

Python Console bölümüğe aşağıdaki kodu yapıştırıp kaydetten sonra çalıştırıyoruz.

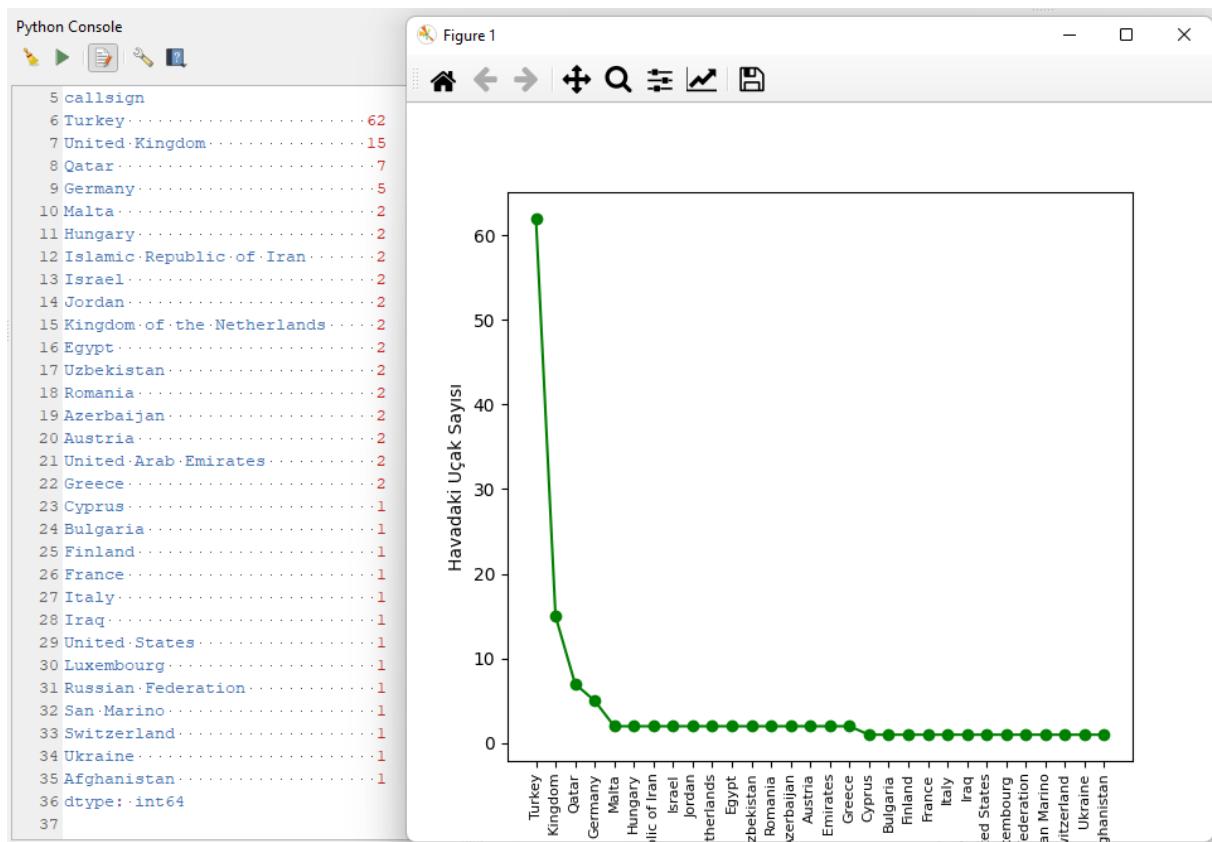
```
""" Kütüphaneler """
import pandas as pd
import matplotlib.pyplot as plt

""" pandas kütüphanesi ile yapılan analizler """
df = pd.read_csv('data.csv')
#df.head()
#print(df)
country_counts = df.value_counts('callsign') #origin_country
print(country_counts)

""" matplotlib kütüphanesi ile yapılabilecek grafikler """
plt.ylabel("Havadaki Uçak Sayısı")
plt.xlabel("Ülkeler")
plt.xticks(rotation='vertical', size=8)
plt.plot(country_counts, 'go-')
plt.show()
```

2.3.5. Qgis Python Console Sonuç

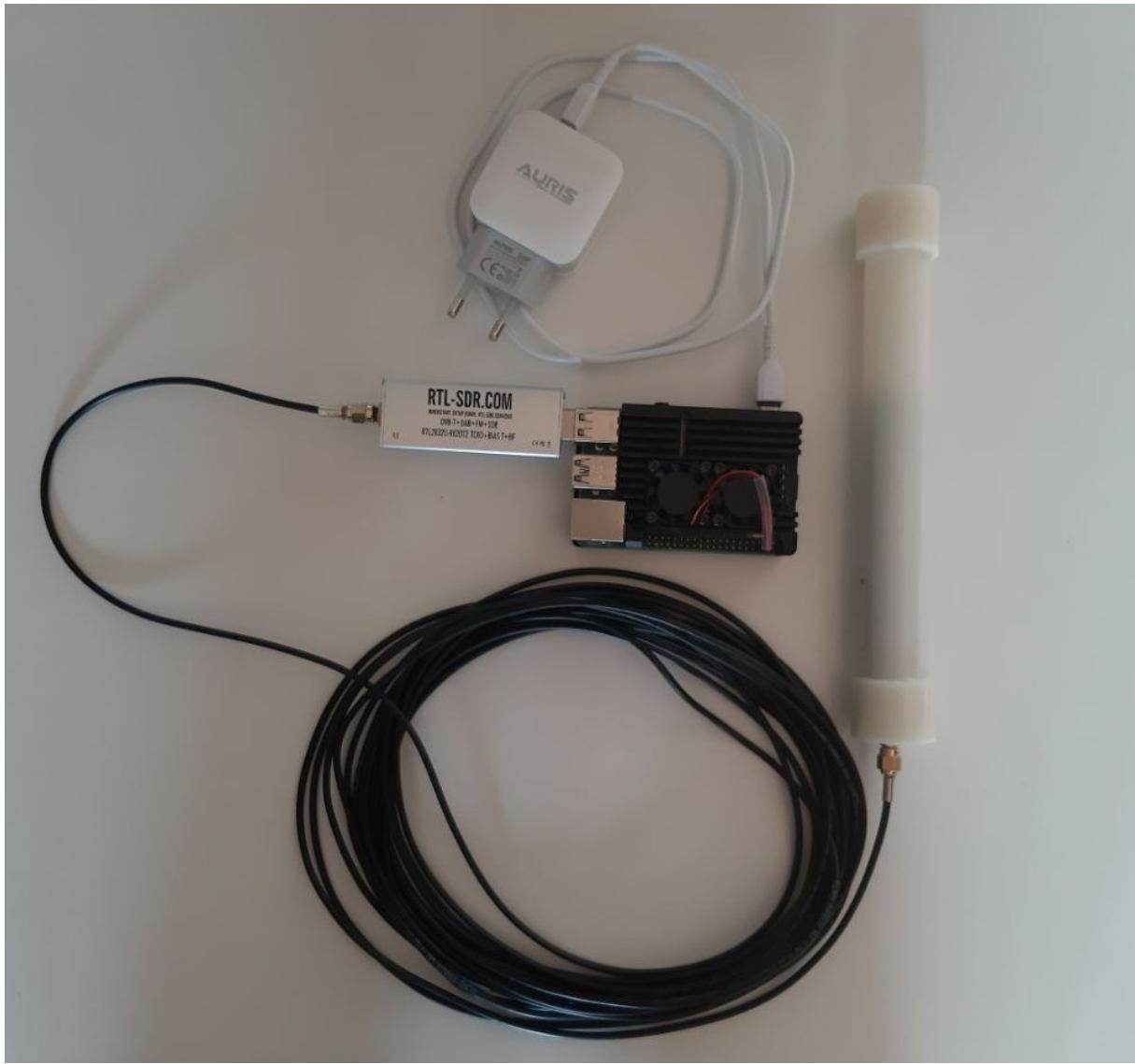
Türkiye hava sahası içerisinde hangi ülkeye ait kaç tane uçak sayısı bulunuyor bunu daha rahat bir şekilde anlayabilmek amaçlı yazdığımız kodu çalıştırdıktan sonra şekil 29'daki gibi bir figür açılır.



ŞEKİL 29: Python Console bölümünde çalıştırılan kod sonucu çıkan sonucun ekran görüntüsü

3. SONUÇ

Dinamik harita verisi oluşturmak için ihtiyaç duyulan verileri sağlayan OpenSky Network sitesinin verileri nasıl sağladığı öğrenilerek gerekli donanımlar temin edildikten sonra şekil 30'daki gibi birleştirildi.



ŞEKİL 30: Verileri toplamak için gerekli olan donanımların birleştirilmiş hali

Donanımlar birleştirildikten sonra veri sağlayıcısı olan OpenSky Network ile gerekli veri gönderme bağlantıları sağlandı. Verilerinde gönderilmesinden sonra Türkiye hava sahasında bulunan uçakların anlık verileri OpenSky Network Api sayesinde çekildi. Bu çekilen veriler ile 3 farklı uygulama yapıldı.

İlk uygulama olarak Jupyter Notebook üzerinde Python programlama dili ile uçakların her 5 saniyedeki bir verileri OpenSky Network üzerinden çekildi ve harita üzerinde sürekli olarak veriler yenilenerek gösterildi. Uygulama üzerinde ülke, uçak adı, hız, barometrik yükseklik bilgileri şekil 21'deki gibi gösterildi.

Bir sonraki uygulama olarak HTML, CSS ve JavaScript programlama dili yardım ile web sitesi yapılarak online bir ortam oluşturuldu. Uygulama içerisinde uçak bilgileri olarak ülkesi, enlemi, boylamı, hızı ve barometrik yüksekliği şekil 22'deki gibi gösterildi.

Son uygulama olarak QGIS üzerinden verilerin görselleştirilmesi için; Python programlama dili yardım ile eğer OpenSky Network hesabı varsa 5 saniyede bir anonim şekilde veri çekiliyorsa 10 saniyede bir veri çekildi. Veriler çekildikten sonra QGIS üzerinde çekilen veriler açıldı ve gerekli düzenlemeler yapıldıktan sonra şekil 26'daki gibi şekiller oluştu. Uçak verilerin her 5 saniyede bir yenilenmesi için QGIS katman ayarlaması yapıldı. Bu işlemde yapıldıktan sonra uçakların konumları her 5 saniyede bir yenilenerek harita üzerinde oluşturuldu. Türkiye hava sahası üzerinden hangi ülkeye ait kaç uçak olduğunu daha rahat bir şekilde görüntüleyebilmek amaçlı Python Console üzerinde Python ile kodlanmış kod çalıştırıldı ve şekil 29'daki gibi görsel oluşturuldu.

Uçakların konum verileri ve diğer bilgileri kullanılarak yapılan bu üç uygulama konum verileri ile bir çok uygulama yapılabileceğini bizlere göstermiş oldu. Bu uygulamaların yanı sıra Blender çizim programında 3 boyutlu haritalar üzerinde 3 boyutlu olarak görünen uçak figürleri yapılabilir yine bir başka uygulama olarak telefonlar için mobil uygulama da yapılabilir. Bu çalışma bizlere konumsal verilerin nasıl değerlendirileceği ve veriler ile neler yapılabileceğini göstermiş oldu.

KAYNAKLAR

- ÖZGÜL, Fırat (2016), PYTHON 3 için Türkçe Kılavuz Sürüm 3, İstihza yayınları, İstanbul.
- SAMANCIOĞLU, Atıl (2021), Python Sıfırdan Uzmanlığa Programlama, Unikod yayınları, 5. Basım, İstanbul.

İNTERNET KAYNAKLARI

- URL 1: <https://docs.python.org/3/license.html>, Python, 18 Mayıs 2022.
- URL 2: https://en.wikipedia.org/wiki/History_of_Python, Pyhton Tarihi, 11 Mayıs 2022.
- URL 3: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language)), Python, 17 Mayıs 2022.
- URL 4: <https://github.com/openskynetwork/raspberry-pi-adsb>, Raspberry Pi-ADSB, 30 Temmuz 2021.
- URL 5: <https://openskynetwork.github.io/opensky-api/rest.html>, OpenSky Api, 2021.
- URL 6: <https://opensky-network.org/community/blog>, 09 Ocak 2021.
- URL 7: <https://tr.wikipedia.org/wiki/QGIS>, QGIS, 06 Nisan 2022
- URL 8: <https://wiki.python.org/moin/>, Python, 16 09 2018.
- URL 9: <https://www.geodose.com/2020/09/realtime%20live%20data%20visualization%20qgis.html>, Qgis'te gerçek zamanlı veri gösterimi, 17 Eylül 2020.
- URL 10: <https://www.hostinger.web.tr/rehberler/css-nedir/>, CSS nedir, 26 Mart 2022.
- URL 11: <https://www.hostinger.web.tr/rehberler/html-nedir/>, Html nedir, 24 Nisan 2022.
- URL 12: <https://www.hostinger.web.tr/rehberler/javascript-nedir/>, JavaScript nedir, 22 Mart 2022.
- URL 13: <https://www.python.org/doc/essays/blurb/>, Python, 2022.
- URL 14: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>, Rest Api nedir, 08 Mayıs 2020.
- URL 15: <https://www.webtekno.com/python-nedir-merak-edilenler-cevaplari-h113594.html>, Python, 15 02 2022.

EKLER

Ek 1 Jupyter Notebook'da Python ile verilerin görselleştirmesi için kullanılan kodlar

Ek 1 Jupyter Notebook'da Python ile verilerin görselleştirmesi için kullanılan kodlar

```
""" Kütüphaneler """
import requests
import pandas as pd
import numpy as np
from bokeh.plotting import figure
from bokeh.tile_providers import get_provider, OSM
from bokeh.io import output_notebook, show, curdoc
from bokeh.models import ColumnDataSource

""" Kodun Notebook üzerinde çıktı vermesi için """
output_notebook()

def modify_doc(doc):

    """ İlk figürlerin oluşmasından önce hata vermemesi için verileri null ata """
    flight_source = ColumnDataSource({
        'icao24':[],'callsign':[],'origin_country':[],
        'time_position':[],'last_contact':[],'long':[],'lat':[],
        'baro_altitude':[],'on_ground':[],'velocity':[],'true_track':[],
        'vertical_rate':[],'sensors':[],'geo_altitude':[],'squawk':[],'spi':[],
        'position_source':[],'MercatorX':[],'MercatorY':[],'rot_angle':[],'url_data':[] })

    def update():
        """ Gelen coğrafi koordinatları web mercator dönüştür """
        def wgs84_to_web_mercator(df, lon="long", lat="lat"):
            k = 6378137
            df["MercatorX"] = df[lon] * (k * np.pi/180.0)
            df["MercatorY"] = np.log(np.tan((90 + df[lat]) * np.pi/360.0)) * k
            return df

    """ Türkiye'yi içine alan coğrafi koordinatlar """
    lon_min,lat_min = 25, 35
    lon_max,lat_max = 45, 45
    """ REST API üzerinden veri çekmek için gerekli argümanlar """
    user_name =
    password =
    url_data = 'https://'+user_name+':'+password+'@opensky-network.org/api/states/all?'+\
    '&lonmin=' + str(lat_min) +'&lonmin=' + str(lon_min) +'&lonmax=' + str(lat_max) +\
    '&lonmax=' + str(lon_max)

    """ REST API üzerinden istekte bulun """
    response = requests.get(url_data).json()
    #print(response)
    #print(type(response))
    """ Gelen verinin sutün ismi """
    col_name =
    ['icao24','callsign','origin_country','time_position','last_contact','long','lat','baro_altitude','on_\
    ground','velocity','true_track','vertical_rate','sensors','geo_altitude','squawk','spi','position_sou
```

```

rce']

        """ Verileri DataFrame tipine çevir """
        flight_df = pd.DataFrame(response['states'])
        #print(flight_df)
        #print(type(flight_df))
        """ Sütün isimlerini numaralandır """
        flight_df = flight_df.loc[:,0:16]
        #print(flight_df)
        #print(type(flight_df))
        """ Sütün numaraları yerine col_name kısmındaki isimlendirmeleri ile değiştir """
        flight_df.columns = col_name
        #print(flight_df.columns)
        #print(type(flight_df.columns))
        """ NAN tipinde gelen boş alanları hata almamak amaçlı No Data yap flight_df"""
        flight_df = flight_df.fillna('No Data') #replace NAN with No Data
        #print(flight_df)
        #print(type(flight_df))

        """ Fonsiyonu çağır """
        wgs84_to_web_mercator(flight_df)
        flight_df['rot_angle'] = flight_df['true_track']*-1
        icon_url = 'https://cdn-icons-png.flaticon.com/512/1679/1679938.png'
        flight_df['url_data'] = icon_url

        """ Verilerin güncelleme işlemini yap """
        n_roll = len(flight_df.index)
        flight_source.stream(flight_df.to_dict(orient="list"),n_roll)

        """ Karanlık mod ekle """
        curdoc().theme = 'dark_minimal'

        doc.add_periodic_callback(update, 5000)

        """ Nerelere şekil çizileceğine dair bilgileri gir """
        p = figure(plot_width=900, plot_height=700, x_range=(3000000, 5000000),
y_range=(3500000, 6000000),
           x_axis_type="mercator", y_axis_type="mercator", tooltips=[("Ülke",
"@origin_country"),
("Uçak Adı", "@callsign"), ("Hız", "@velocity m/s"), ("Barometrik Yükseklik",
"@baro_altitude m")], title = "Anlık Uçak Konumları")

        """ Uçak figürlerini çizdir """
        p.image_url(url='url_data', x='MercatorX', y='MercatorY', source=flight_source,
anchor='center', angle_units='deg', angle='rot_angle', h_units='screen', w_units='screen',
w=25, h=25)

        """ Çizilecek şekilleri ayarla """
        p.circle(x="MercatorX", y="MercatorY", size=7, fill_color="red", line_color="black",
fill_alpha=1, source=flight_source)

```

```
""" Harita allığını çağır """
tile_provider = get_provider(OSM)
""" Harita allığını p şekilleri ile birleştir """
p.add_tile(tile_provider)

doc.add_root(p)

""" Haritayı göster """
show(modify_doc)
```

ÖZGEÇMİŞ

| | | |
|--------------|---------------|---|
| İsim soyisim | Erdem Güneş | |
| Doğum tarihi | 15.07.1997 | |
| Doğum yeri | Kahramanmaraş | |
| Lise | 2012-2016 | Mehmet Gümüşer Anadolu Lisesi |
| Lisans | 2018- | Kocaeli Üniversitesi Mühendislik Fakültesi Harita Mühendisliği |

Akademik ve Mesleki Deneyimler

| | |
|-----------|--------------------|
| 2021-2021 | GeoPerformans staj |
| 2021-2021 | Yer Çizenler staj |

| | | |
|--------------|-------------|--|
| İsim soyisim | Onur Yıldız | |
| Doğum tarihi | 24.12.1999 | |
| Doğum yeri | Samsun | |
| Lise | 2014-2018 | Bahçeşehir Koleji Atakum Kampüsü |
| Lisans | 2018- | Kocaeli Üniversitesi Mühendislik Fakültesi Harita Mühendisliği Bölümü |

Akademik ve Mesleki Deneyimler

Staj Çalışmaları ;

07/2021-08/2021 Köse Harita İnşaat Taahhüt Ve Gayrimenkul Ticaret Limited Şirketi / Samsun

08/2021-09/2021 Karayolları 4. Bölge Müdürlüğü / Ankara