

**KOCAELİ ÜNİVERSİTESİ  MÜHENDİSLİK FAKÜLTESİ
HARİTA MÜHENDİSLİĞİ BÖLÜMÜ**

**Uçak Verilerinin Yakalanması ve Veriler İle Olası
Çalışmalar Yapılması**

İsim Soyisim Numara

BİTİRME ÇALIŞMASI

**KOCAELİ
Mayıs, 2022**

**KOCAELİ ÜNİVERSİTESİ  MÜHENDİSLİK FAKÜLTESİ
HARİTA MÜHENDİSLİĞİ BÖLÜMÜ**

**Uçak Verilerinin Yakalanması ve Veriler İle Olası
Çalışmalar Yapılması**

İsim Soyisim Numara

BİTİRME ÇALIŞMASI

Danışman	:	Doç. Dr.
Üye	:	
Üye	:	

**KOCAELİ
Mayıs, 2022**

İÇİNDEKİLER

İçindekiler

İÇİNDEKİLER.....	3
KISALTMALAR.....	5
ŞEKİL LİSTESİ.....	6
ÇİZELGE LİSTESİ.....	7
ÖNSÖZ.....	8
ÖZET.....	9
ABSTRACT.....	10
1. Giriş.....	11
1.1. Python.....	11
1.1.1. Python Nedir.....	11
1.1.2. Python'un Tarihi.....	12
1.2. HTML (Hypertext Markup Language).....	15
1.3. CSS (Cascading Style Sheet).....	15
1.4. JavaScript.....	16
1.5. QGIS (Quantum GIS).....	16
1.6. REST API.....	17
1.6.1. API Nedir.....	17
1.6.2. REST Nedir.....	18
1.6.3. OPENSKY REST API.....	19
1.6.3.1. Tüm Durum Vektörleri.....	19
1.6.3.2. Talepte Bulunma.....	19
1.6.3.3. Gelen Yanıt.....	20
1.6.3.4. Anonim (Kimliği Doğrulanmamış) Kullanıcılar İçin Kısıtlamalar.....	22
1.6.3.5. OpenSky Kullanıcıları İçin Kısıtlamalar.....	22
1.7. OpenSky Network İçin Raspberry Pi İle ADS-B Baz İstasyonu.....	23
1.7.1. API Nedir.....	23
1.7.2. Önemli Detaylar.....	25
1.8. Baz İstasyonu İçin Gerekli Donanımlar.....	26
1.8.1. Raspberry Pi Ekipmanları.....	26
Raspberry Pi ekipmanları sistemin yazılım kısmından sorumlu bölüm oluyor.....	26
1.8.1.1. Raspberry Pi 4 ve Çift Fanlı Metal Kasa Soğutma Sistemi.....	26
1.8.1.2. Hafıza Kartı.....	27
1.8.1.3. Adaptör.....	28
1.8.2. Alıcının Ekipmanları.....	29
1.8.2.1. Anten.....	29
1.8.2.2. RTL-SDR.....	30
1.8.2.3. Her İki Ucu SMA Erkek Bağlantı Girişli Koaksiyel Bakır Kablo.....	31
1.9. Ayar Aşamaları.....	32
1.9.1. Raspberry Pi'nin birleştirilmesi.....	33
1.9.2. Alıcı Ekipmanlarının birleştirilmesi.....	33
1.9.3. Raspberry Pi cihazının yazılım bölümünün ayarlanması.....	34
1.9.3.1. İşletim sistemi ile SD kartın hazırlanması.....	34
1.9.3.2. İşletim sisteminin kurulumunun gerçekleştirilmesi.....	35
1.9.3.3. RTL-SDR alıcısı için sürücüler oluşturulma ve yükleme.....	36
1.9.3.4. dump1090 kod çözümünün oluşturulması ve kurulumu.....	37
1.9.4. OpenSky Network hesabının oluşturulması ve yeni bir alıcının eklenmesi.....	39

2. UYGULAMALAR.....	42
2.1. Python Jupyter Notebook İle Yapılan Uygulama.....	42
2.1.1. Python Jupyter Notebook İle Veriyi Görselleştirme.....	42
2.1.2. Sonuç.....	44
2.2. JavaScript İle Web Sitesi Yaparak Online Görselleştirme.....	45
2.2.1. HTML Kodları.....	45
2.2.2. CSS Kodları.....	46
2.2.3. JavaScript Kodları.....	46
2.2.4. Sonuç.....	48
2.3. QGIS Üzerinde Canlı Veri İle Yapılan İşlemler.....	49
2.3.1. Verilerin Python Yardımı İle Belirli Aralıklarda Sunucudan İstenmesi.....	49
2.3.2. Verilerin Python Yardımı İle Belirli Aralıklarda Sunucudan İstenmesi.....	50
2.3.3. Verilerin QGIS üzerinde görselleştirilmesi.....	51
2.3.4. Verilerin QGIS üzerinde Python Konsole Yardımı İle Grafiksel Görselleştirilmesi.	54
2.3.5. Sonuç.....	55
KAYNAKLAR.....	56
İNTERNET KAYNAKLARI.....	56
EKLER.....	57
ÖZGEÇMİŞ.....	58

KISALTMALAR

ADS-B	: Automatic Dependent Surveillance-Broadcast
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

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

13

ÇİZELGE LİSTESİ

ÇİZELGE 1 Python'un bazı sürüm bilgilerini göstermektedir	12
ÇİZELGE 2 Talep parametreleri	18
ÇİZELGE 3 Koordinat sisteminde alabileceği değerlerin veri tipleri	18
ÇİZELGE 4 Gelen yanıtın veri tipleri	19
ÇİZELGE 5 Gelen veri içerisindeki veriler ve verilerin tipleri	19

ÖNSÖZ

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

Bu çalışmada, Raspberry Pi yardımıyla kurulan düzenek ile Uçak Verilerinin Yakalanması ve Jupyter Notebook kullanarak Python programlama dili ile Verilerin işlenmesi çalışmaları ele alınmaktadır. Böyle bir çalışmanın amacı uçaklara ait verilerin anlık takip edilip görüntülenmesi ve bu takibin kendimizin kurduğu düzenek ile sağlanmasıdır.

Çalışmanın hayata geçirilmesi sürecinde görüşleri ile destekleyen danışman hocamız Doç. Dr. Taner Üstüntaş'a teşekkürlerimizi sunarız.

ÖZET

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 yayımlanan sinyaller yakalanır ve diğer donanımlarında yardımcı ile veriler temizlenip işlendikten sonra uçakların yaydığı bilgilere ulaşılmış olur. Bu uygulamada [baz istasyonu](#) OpenSky-Network için ayarlandı.

Anten çekim alanı sınırlı olduğu ve insanların başka konumlardaki verilere de ulaşabilmesi için [OpenSky-Network](#) gibi internet sitelerinden faydalansılabilir. 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 yapılmak istediği 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 kişilerin 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

Planes constantly share location and other information, increasing the rate of safe travel. Planes send this information to everywhere by radio waves. An antenna is needed to capture the signals emitted by planes. 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 the base station is set for OpenSky-Network.

Websites such as OpenSky-Network can be used to ensure that the antenna area is limited and people can access data in other locations. After this data is collected by antennas, it is sent to OpenSky-Network servers and collected on the servers. After that those who need this data can pull it with the Open Sky 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. Giriş

Bu bölümde Python programlama dili, OpenSky Rest API konularına 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 level 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 ekleni ve paketlerini destekler. Python yorumlayıcısı ve kapsamlı standart kütüphaneleri, bütün büyük platformlar için ücretsiz olarak yazılım veya ikilik sistemde mevcuttur ve ücretsiz olarak dağıtılabılır.

Programcılar, sağladığı üretkenlik nedeniyle genellikle Python'a hayran kalırlar. Python'da derleme işlemi olmadığından, düzenleme, test ve hata ayıklama döngüsü hızlidır. Python programlama dilinde hata ayıklama koladır; bir hata veya hatalı giriş bölünme hatasına sebep olmaz. Bunun yerine, yorumlayıcı bir hata bulduğunda bir istisna oluşturur. Program istisnayı yakalayamadığında, 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.

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.

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.

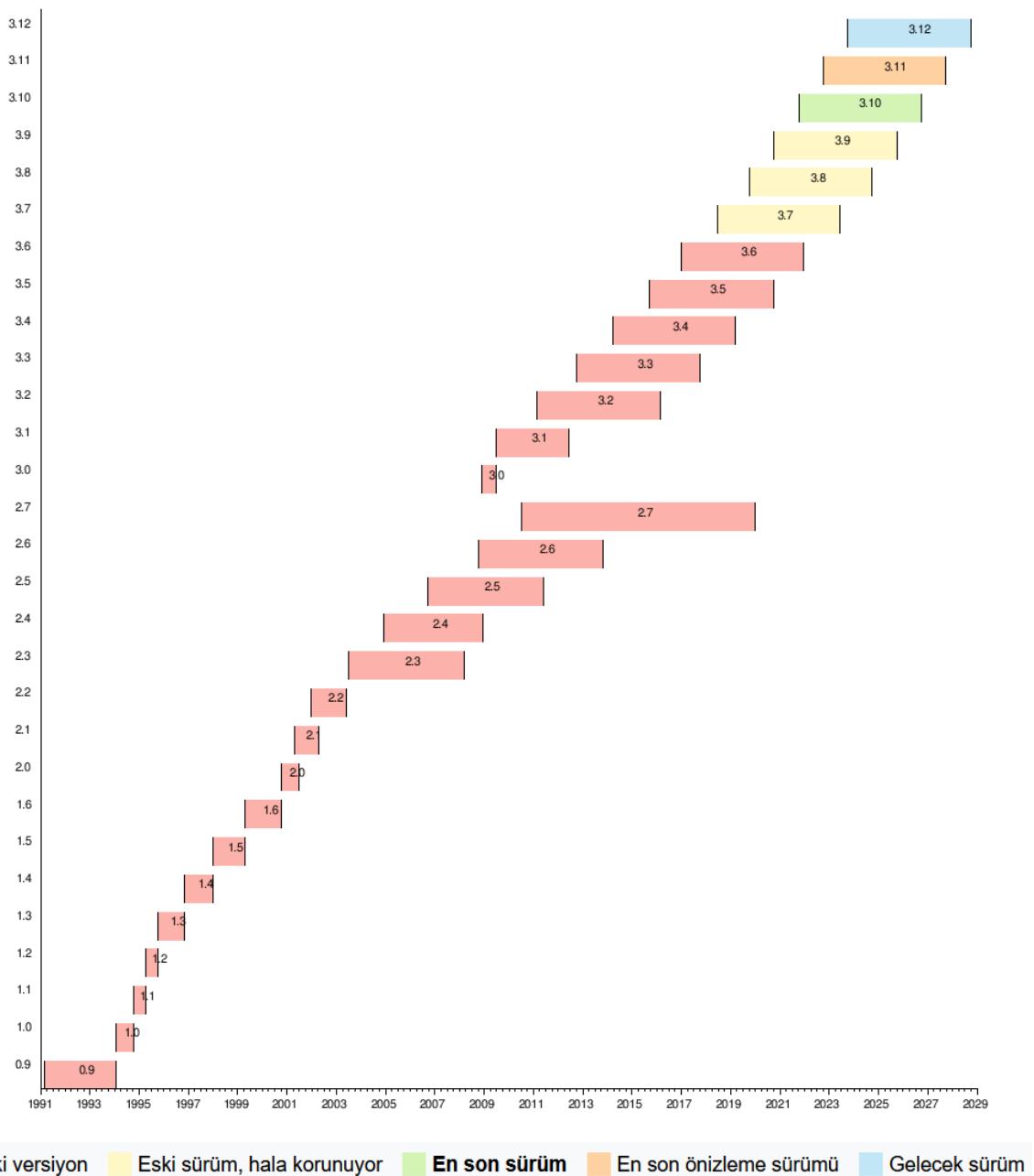
2000 yılının Mayıs ayında, Guido ve Python yazılım geliştirme ekibi BeOpen PythonLabs ekibini oluşturmak için BeOpen.com'a taşındılar. Aynı yılın Ekim ayında, PythonLabs takımı Digital Creations'a (şimdiki isimleri: Zope Corporation) geçiş yaptılar. 2001 yılında, PSF (Python Software Foundation: Python 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 olarakca yayınladı.

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

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



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.

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.

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 şeklinde eder.

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

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.

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 kullanılabılır.

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

QGIS, C++ ile kodlanmasıından ötürü hızlı şekilde çalışabilmektedir. QGIS GNU/Linux, UNIX, Microsoft Windows ve Mac OS gibi bir çok ortamda çalışabilmektedir.

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

1.6. REST API

Online ortamda veri oluşturma ve paylaşmanın hangi yöntemler ile mümkün kılındığını anlatalım.

1.6.1. 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 tasarımlı, kullanıcının bir posta kodu sağlamasını ve üreticinin, ilki yüksek sıcaklık ve ikincisinin düşük olmak üzere iki parçalı bir yanıtla yanıtmasını belirtebilir.

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

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.

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 arasındaki etkileşimleri 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 ölçülebilirlik 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.

1.6.3. OPENSKY REST API

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.

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

1.6.3.2. Talepte Bulunma

Aşağıdaki talep parametrelerinden istenilenler kullanılarak belirli uçaklar veya saatler için durum bilgisi talep edilebilir.

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, aşağıdaki parametrelerin hepsini ekleyiniz.

ÇİZELGE 3 Koordinat sisteminde alabileceğiniz 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 sorgu:

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

İsviçre için sınırlayıcı bölge örnek sorgu:

<https://opensky-network.org/api/states/all?lamin=45.8389&lomin=5.9962&lamax=47.8229&lomax=10.5226>

1.6.3.3. Gelen Yanıt

Gelen yanıt, aşağıdaki özelliklere sahip bir JSON nesnesidir.

ÇİZELGE 4 Gelen yanıtın veri tipleri

Nitelik	Veri Tipi	Tanım
time	integer (sayısal)	Bu yanittaki 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 aşağıdaki alanları içerir.

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 İçin 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 $[now - (now \bmod 10)]$ döndüreceği anlamına gelir.

1.6.3.5. OpenSky Kullanıcıları İçin Kısıtlamalar

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

OpenSky kullanıcılar için kısıtlamalar şunlardır:

- OpenSky kullanıcıları geçmişte 1 saatे kadar verileri alabilir. Zaman parametresinin bir değeri varsa $[t < now (\text{şimdi}) - 3600]$ 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 $[t - (t \bmod 5)]$ durum vektörlerini döndürür.

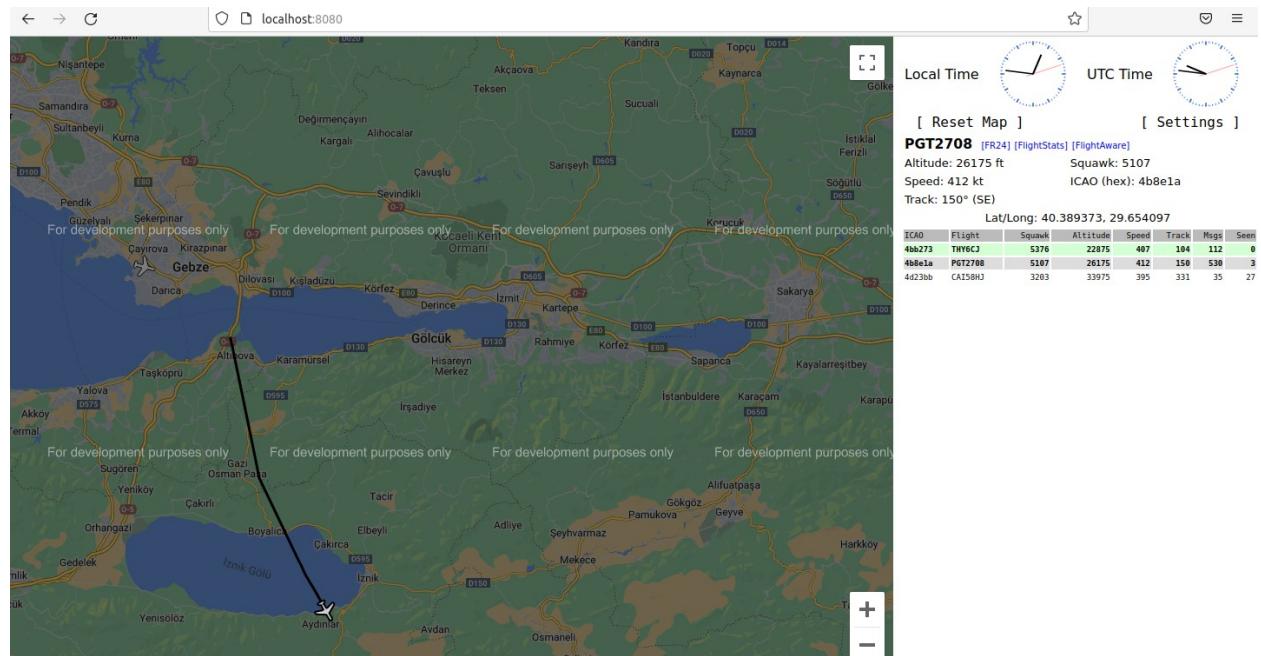
1.7. OpenSky Network İçin Raspberry Pi ile ADS-B Baz İstasyonu

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. API Nedir

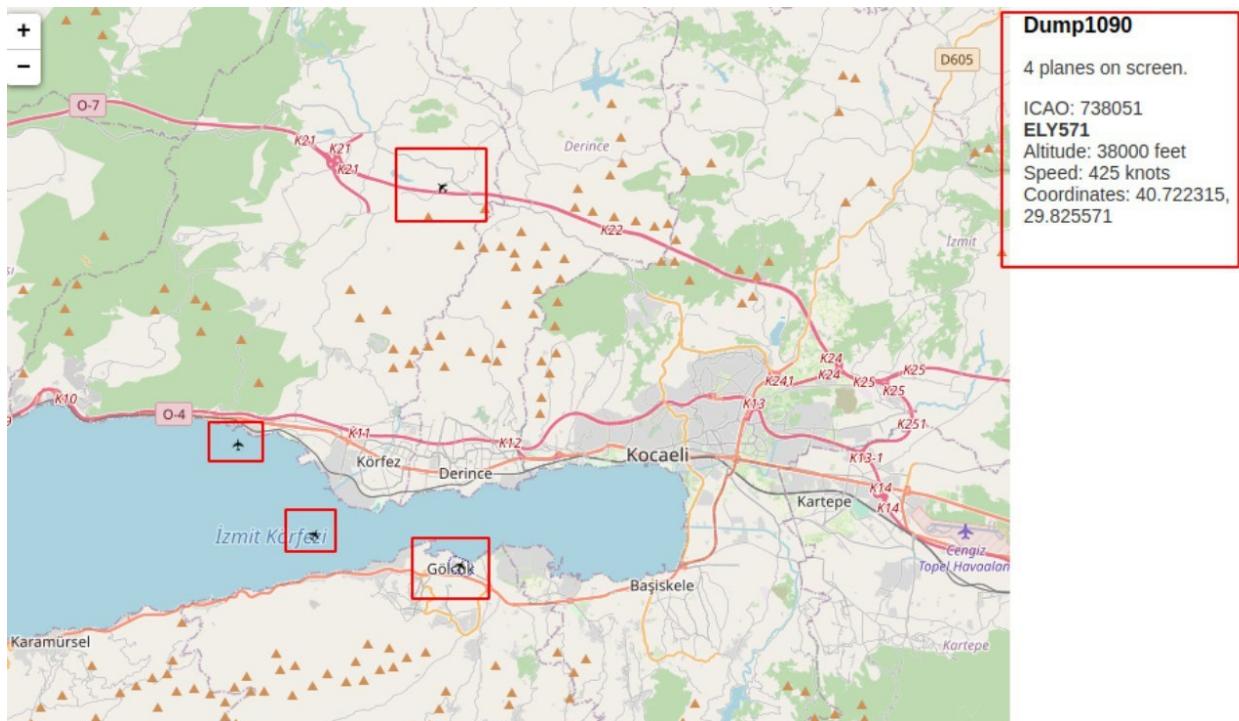
Bu yardımcı kılavuz, Raspberry Pi ve hazır parçalar kullanılarak basit bir ADS-B baz istasyonunun nasıl işlevsel hale getirileceğini ve istasyonunuza OpenSky ağına nasıl bağlayacağınızı açıklar.

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



ŞEKİL

Aynı zamanda verilerinizi [OpenSky Network](#) ve diğer ağlara aktarılabilecek ve [PlanePlotter](#) gibi yerel uygulamalar üzerinden de kullanılabilecektir.



Şekil

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 internete kablosuz olarak bağlayabilmeniz için modeminizin WIFI'yi (Wireless Fidelity: Kablosuz Ağ Bağlantısı) desteklemesi gereklidir.

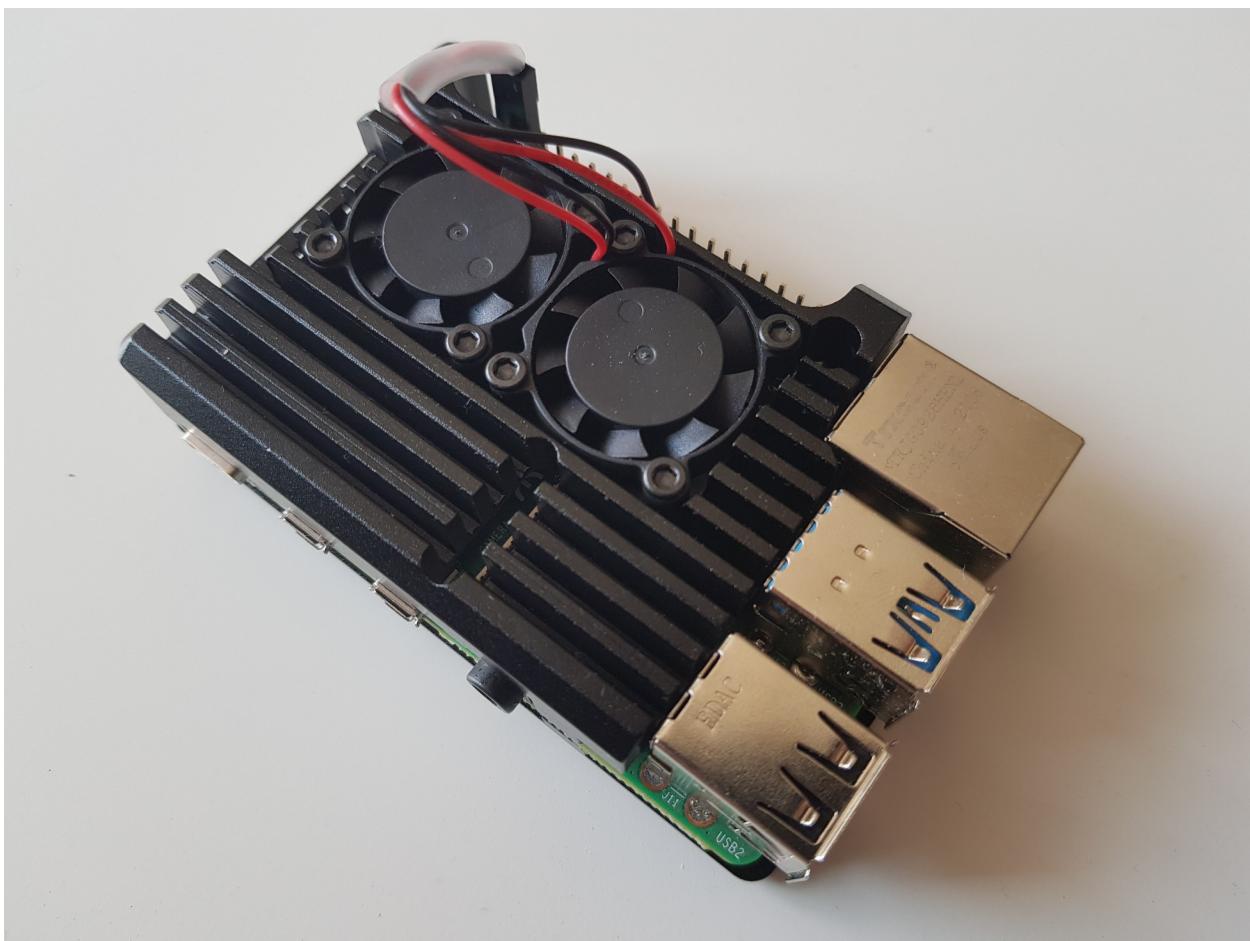
1.8. Baz İstasyonu İçin Gerekli Donanımlar

İstasyonun sinyalleri yakalaması ve o sinyalleri düzenlemesi için bir takım 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

Gerekli yazılımları ve donanımlar arasında bağlantıyı sağlamak amaçlı minimal bir bilgisayar sistemi olan Raspberry Pi 4 2 gb geçici hafızalı bir bilgisayar kullandık. Aynı zaman donanım sürekli çalışacağı için soğutmasını sağlamak amacıyla metal kasa ve çift fanlı bir soğutma tercih ettim.

1.8.1.2. Hafıza Kartı



Şekil

Bilgisayarın çalışır hale gelebilmesi için mini hafıza kart içerisine 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



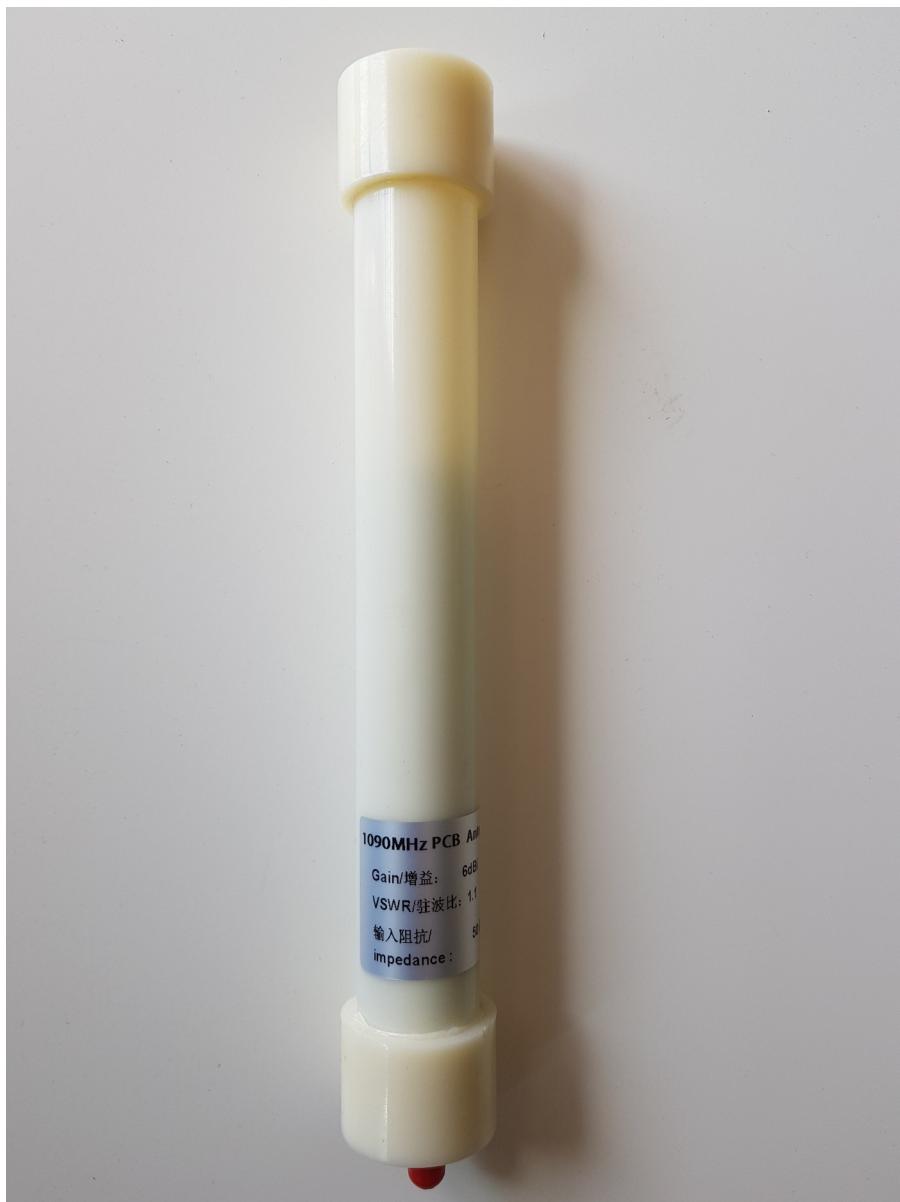
Şekil

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

Uçaktan tarafından yayınlanan sinyalleri yakalamak için bazı donanımlara ihtiyaç vardır. Bu donanımlara bir bakış atalım.

1.8.2.1. Anten



Şekil

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

1.8.2.2. RTL-SDR



Şekil

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



Şekil

Anten ile RTL-SDR donanımı arasında bağlantıyı sağlayabilmek amaçlı her iki ucu SMA erkek olan kablo kullanıldı.

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 gerekmeyeğini 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 çözücünün 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 olmayacaksça tek fanlı soğutma sistemi de kullanılabilir.

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

Raspberry cihazının gerekli yazılımlarının olduğu kısmı olan hafıza kartı, Raspberry Pi üzerinde bulunan hafıza 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 kadın 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 bir birine 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.

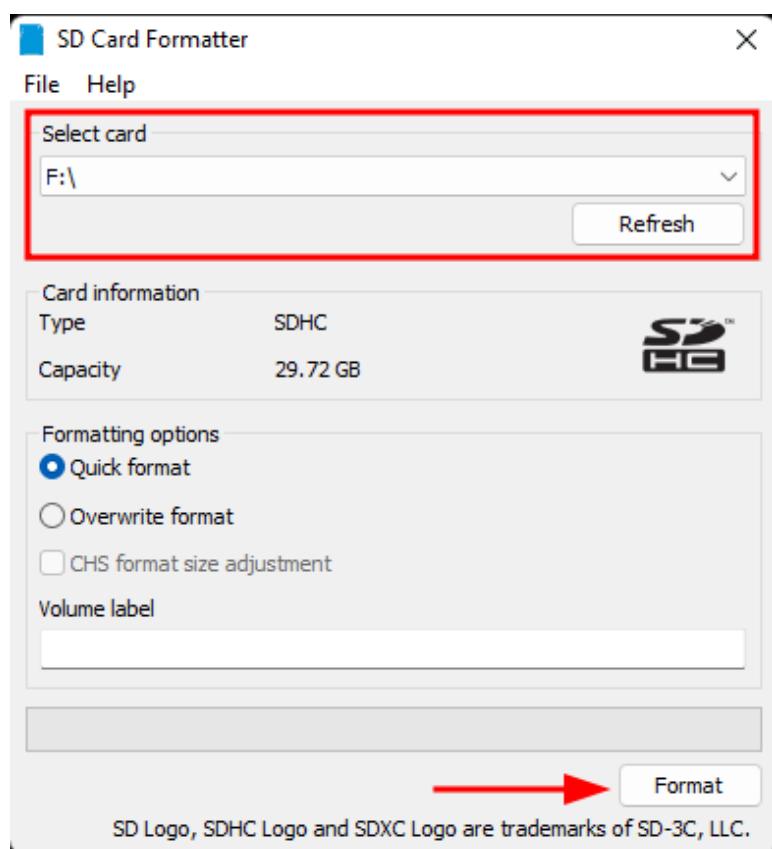
Bu işlemde yapıldıktan sonra donanımların bağlantısı tamamlanmış oluyor. Geriye ise Raspberry Pi içerisinde çalıştırılacak yazılımlar kalıyor.

1.9.3. Raspberry Pi cihazının yazılım bölümünün ayarlanması

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

1.9.3.1. İşletim sistemi ile SD kartın hazırlanması

Eğer kurulum yapacağımız 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 görseldeki gibi bir ekran gelecektir.



Sekil

Select card bölümünden disk seçiyoruz ve Format bölümünden de diskin formatlanmasıını 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 görselde göründüğü gibi bir ekran karşınıza gelecektir.



Şekil

CHOOSE OS bölümünden kurulumunu yapmak istediğimiz işletim sistemini seçiyoruz. *CHOOSE STORAGE* bölümünden kurulumu yapmak istediğimiz diski 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 tamalıyoruz.

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

Güncelleme 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...
```

Şekil

Görseldeki gibi bir ekran gelmesi gerekiyor. Eğer böyle bir sonuç aldıysanız sorunsuz şekilde RTL-SDR sürücülerini çalışıyor demektir.

1.9.3.4. dump1090 kod çözücünün oluşturulması ve kurulumu

Gelen sinyallerin çözülmesi için dump1090 yazılımının kurulumu için OpenSky-Network tarafından hazırlanan yazılımından yardım alıyoruz.

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

Yukarıda komut ile dump1090 yazılımının kurulumu sağlamış oluyoruz.

Kurulum sonucunda sonuçlarımızı görmek için terminale alttaki komutu çalıştırıyoruz.

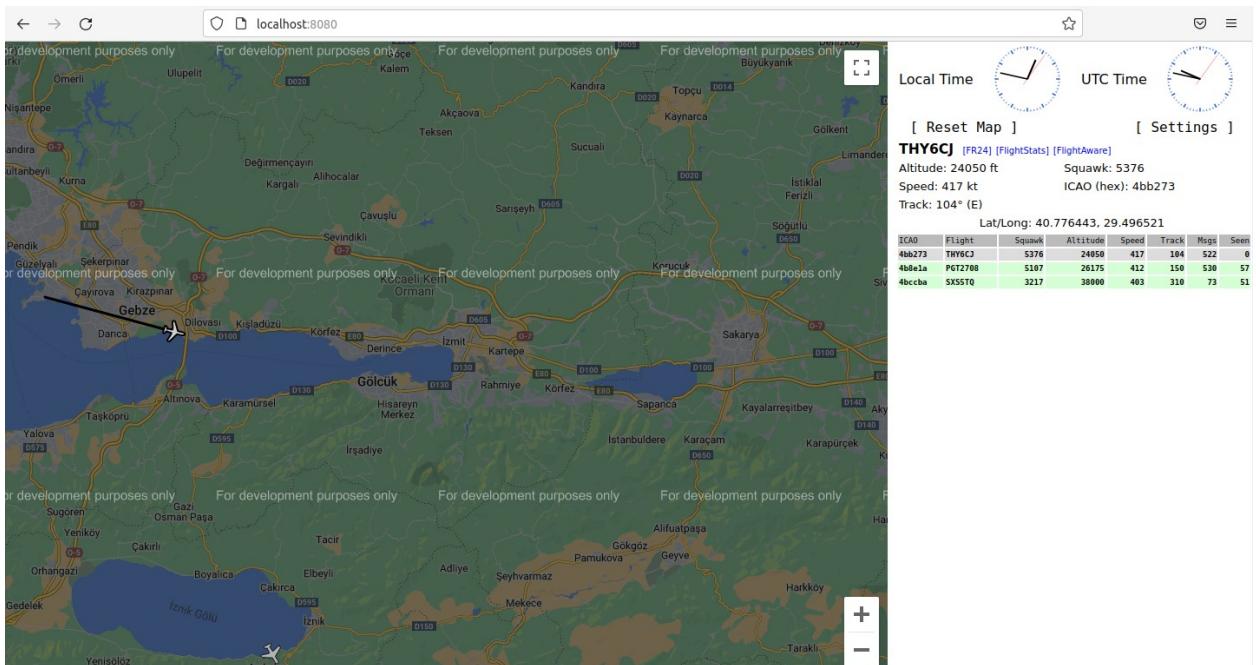
```
./dump1090 --interactive --net
```

Komutu çalıştırdıktan sonra terminalde uçakların verileri görseldeki gibi yansıyacaktır.

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

Şekil

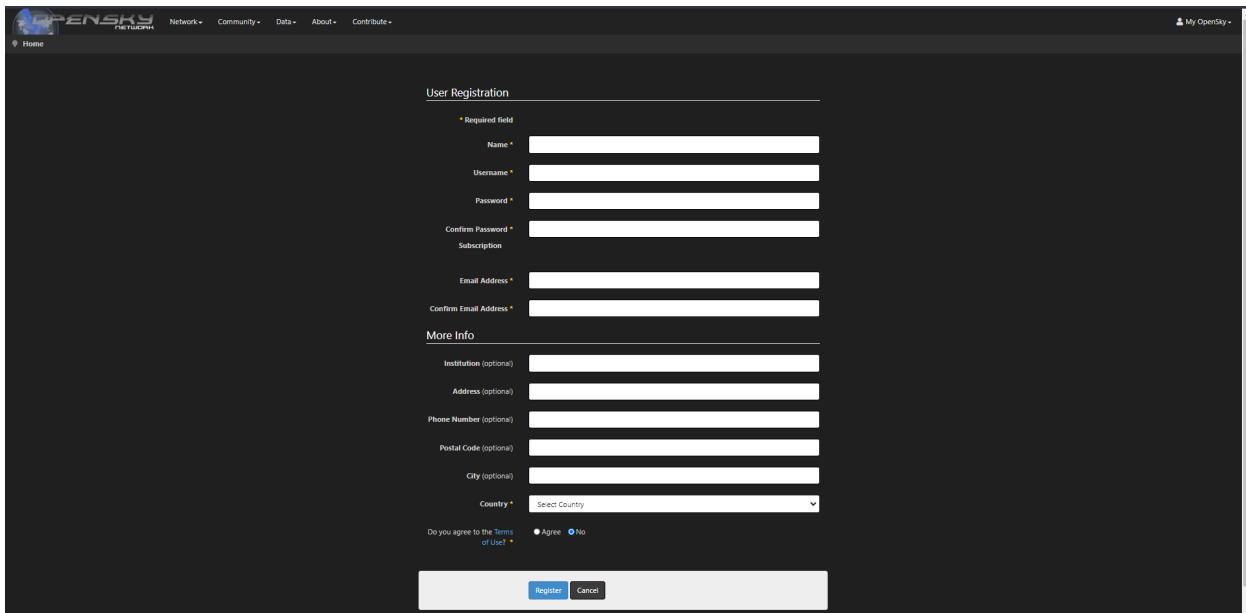
Terminale ek olarak tarayıcı üzerinden kontrol amaçlı <http://127.0.0.1:8080> ya da <http://localhost:8080> üzerinden yerel bölümü açtığınız zaman görseldeki gibi bir harita ile karşılaşırınsız.



Şekil

1.9.4. OpenSky Network hesabının oluşturulması ve yeni bir alıcının eklenmesi

OpenSky-Network internet sayfasında Register bölümüne giderek.



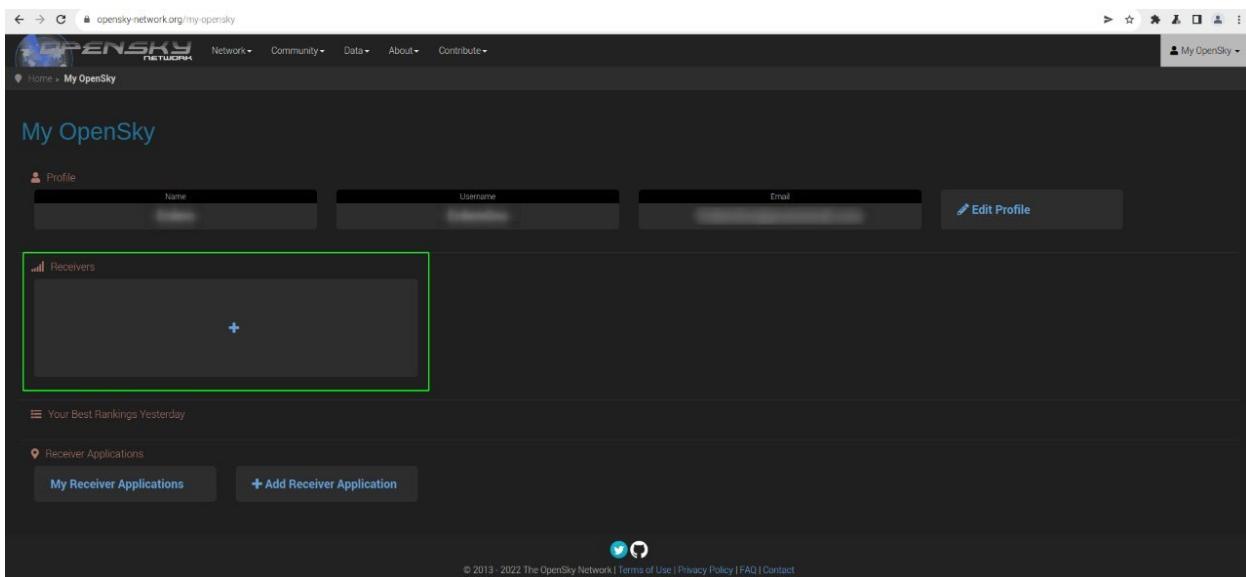
The screenshot shows the 'User Registration' form on the OpenSky Network website. The form is titled 'User Registration' and includes several input fields:

- Name * (Required field)
- Username *
- Password *
- Confirm Password *
- Subscription (dropdown menu)
- Email Address *
- Confirm Email Address *
- More Info (grouped section):
 - Institution (optional) [input field]
 - Address (optional) [input field]
 - Phone Number (optional) [input field]
 - Postal Code (optional) [input field]
 - City (optional) [input field]
 - Country [dropdown menu] (with 'Select Country' placeholder)
- Do you agree to the Terms of Use? (radio buttons: Agree, No)
- Buttons: Register (blue), Cancel (white)

Şekil

Görseldeki alanları doldurarak hesabımızı oluşturuyoruz.

Hesabımızı oluşturduktan sonra donanımı eklemeye sıra geliyor. My OpenSky bölümüne giriyoruz ve görseldeki *Receivers* içerisinde giriyoruz.



Şekil

Receivers bölümü içerisinde girdikten sonra görsellerdeki alanları eksiksiz dolduruyoruz.

Add/Edit Receiver Setup

Hardware

Receiver Type *

dump1090

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.

We support the following forks of dump1090:

- hylo: https://github.com/opensky-network/dump1090_hylo
- MalcolmRobb: <https://github.com/MalcolmRobb/dump1090>
- mutability: <https://github.com/mutability/dump1090>

Most Raspberry Pi images are using the mutability fork.

URL/IP address *

Provide the IP address/URL of your device. Note that our servers must be able to connect to your device, so you have to enable port forwarding (NAT) if you are behind a router. Also, most Internet providers reset the connection once per day and the IP is subject to change. Therefore, we recommend to use a free dynamic DNS service (e.g. [No-IP](#)).

Port *

30005

Enter the port for your device. Defaults: 30005 (dump1090), 10002/10006 (Radarcape), 30003 (SBS-3)

Location

Address *

Izmit/Kocaeli, Türkiye

Set on Map

Enter the address of your receiver's location here. The requested format is street + house number, zip code + city, country, e.g. Example Street 25, 1234 Downton, Australia. Click the button to set the marker in the map. You can fine-tune the marker's position manually. Longitude, latitude and altitude of your location are retrieved automatically.

Şekil

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.

Longitude * 29

Latitude * 40

Elevation (in ft) * 1745

Anonymize

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!

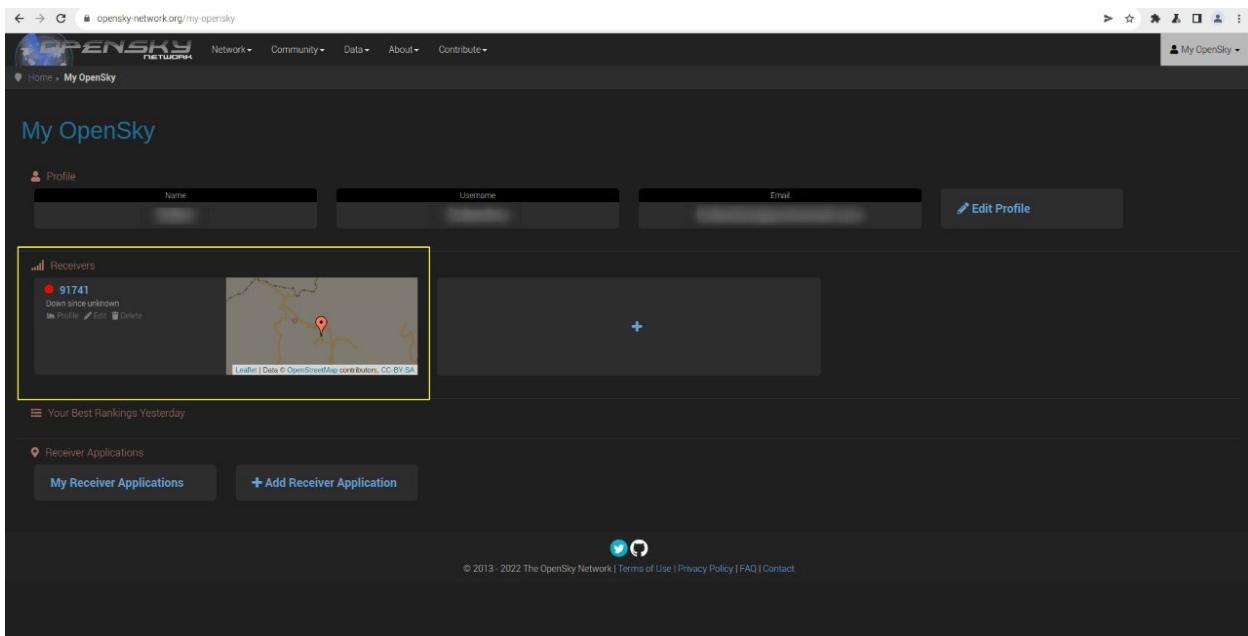
Also please note that despite the fact that this is prohibited by our terms of use, it is fundamentally not possible to fully prevent reverse engineering of the receiver location based on its data. For example, the coverage of a receiver already provides some indication of its location.

For more information, refer to, e.g., this publication: "Eichelberger, Manuel, Kevin Luchsinger, Simon Tanner, and Roger Wattenhofer. "Indoor Localization with Aircraft Signals." In *Sensys*, vol. 17, pp. 6-8. 2017.

Submit Cancel

Şekil

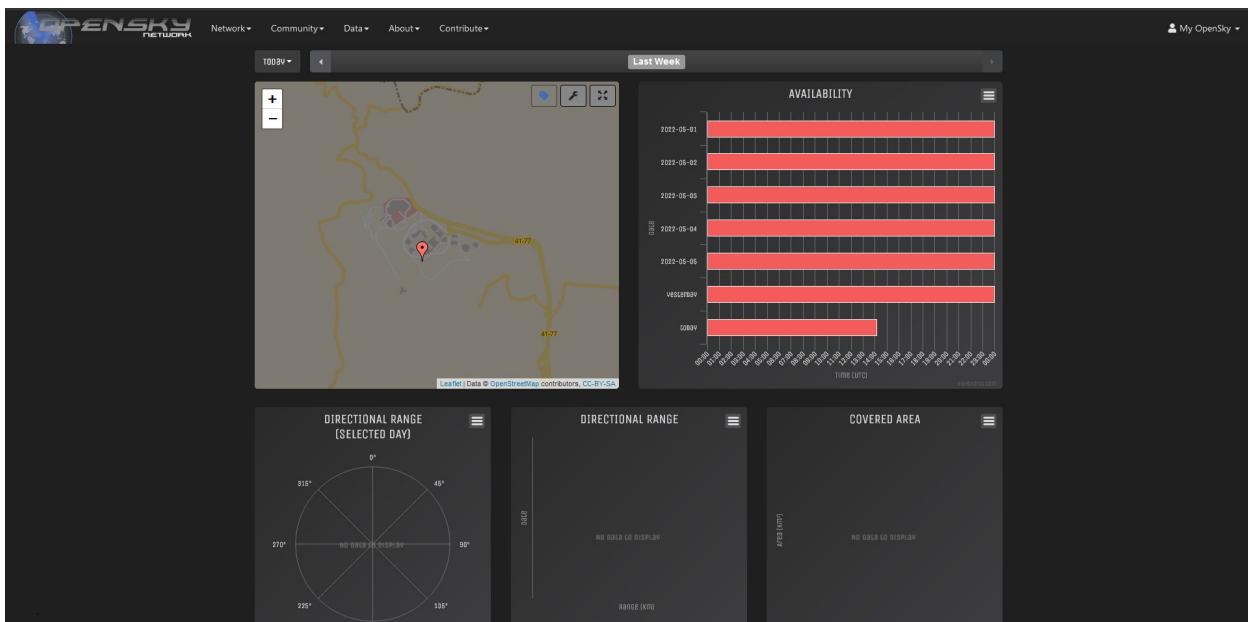
Boş alanlar doğru şekilde doldurulduktan sonra Submit kısmından bilgilerin doğruluğunu onaylıyoruz ve işlem tamamlanmış oluyor.



Şekil

Submit işleminden sonra *Receivers* bölümü içerisinde donanımızın kayıt bilgileri gelmiş oluyor.

Donanım bölümünüze girdikten sonra donanımızın aldığı bilgileri ayrıntılı şekilde görebilirsiniz.



Şekil

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

2. UYGULAMALAR

Bütün gerekli işlemler tamamlandıktan sonra veriler ile ne yapılabilir bunlara bir bakalım.

2.1. Python Jupyter Notebook İle Yapılan Uygulama

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

2.1.1. Python Jupyter Notebook İle Veriyi Görselleştirme

```
""" 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?'+lamin='+str(lat_min)+'&lomin='+str(lon_min)+'&lamax='+str(lat_max)
+'&lomax='+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_source']
""" Verileri DataFrame tipine çevir """
flight_df = pd.DataFrame(response['states'])
#print(flight_df)
#print(type(flight_df))
""" Sutün isimlerini numaralandır """
flight_df = flight_df.loc[:,0:16]
#print(flight_df)
#print(type(flight_df))
""" Sutün 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)

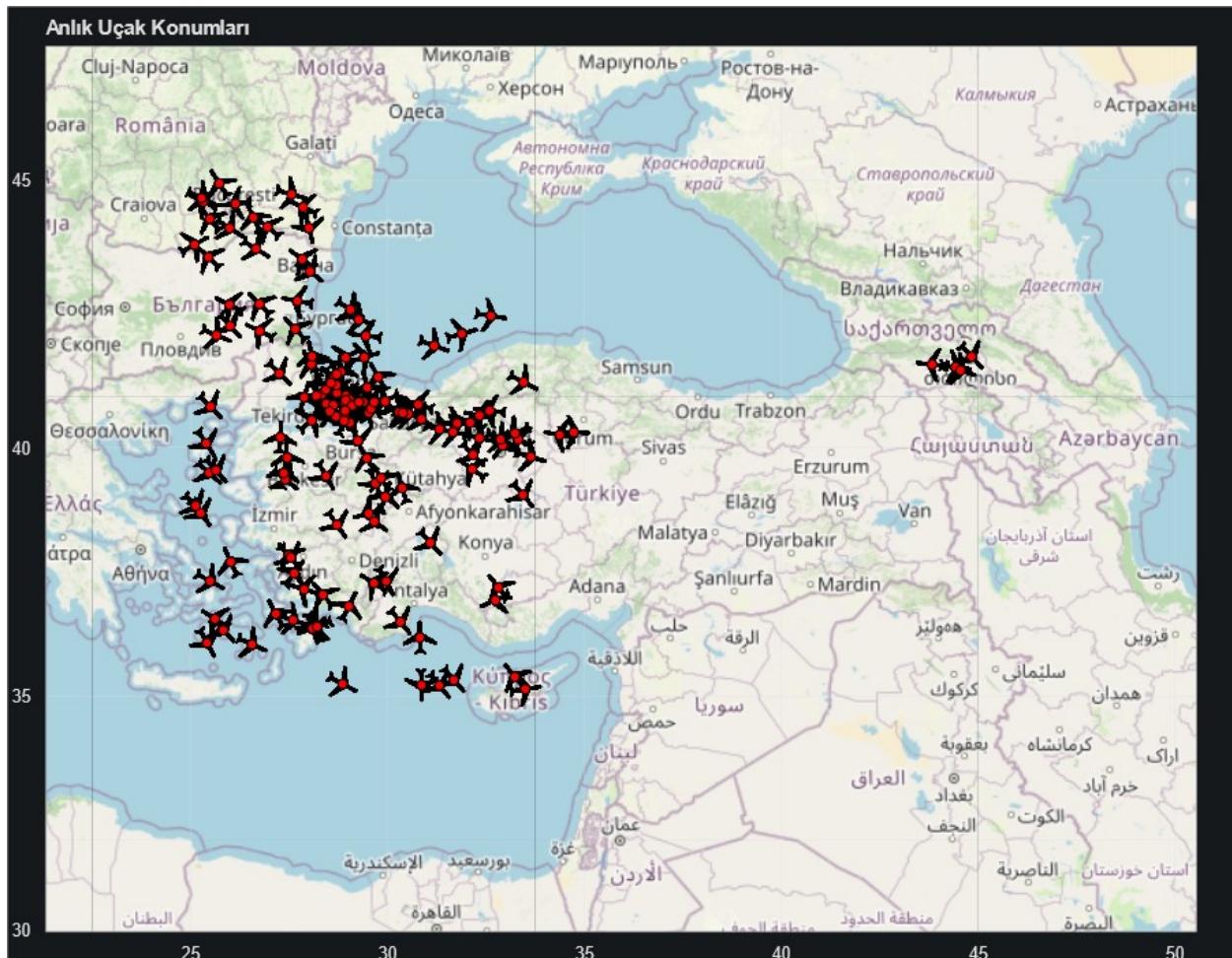
    """ Harita altlığını çağır """
    tile_provider = get_provider(OSM)
    """ Harita altlığını p şekilleri ile birleştir """
    p.add_tile(tile_provider)

doc.add_root(p)

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

```

2.1.2. Sonuç



Şekil

2.2. JavaScript İle 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

```
<!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-
XQoYMQMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaXaV
UearIOBhiXZ5V3ynxwA=="
      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

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

2.2.3. JavaScript Kodları

```
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],
        });

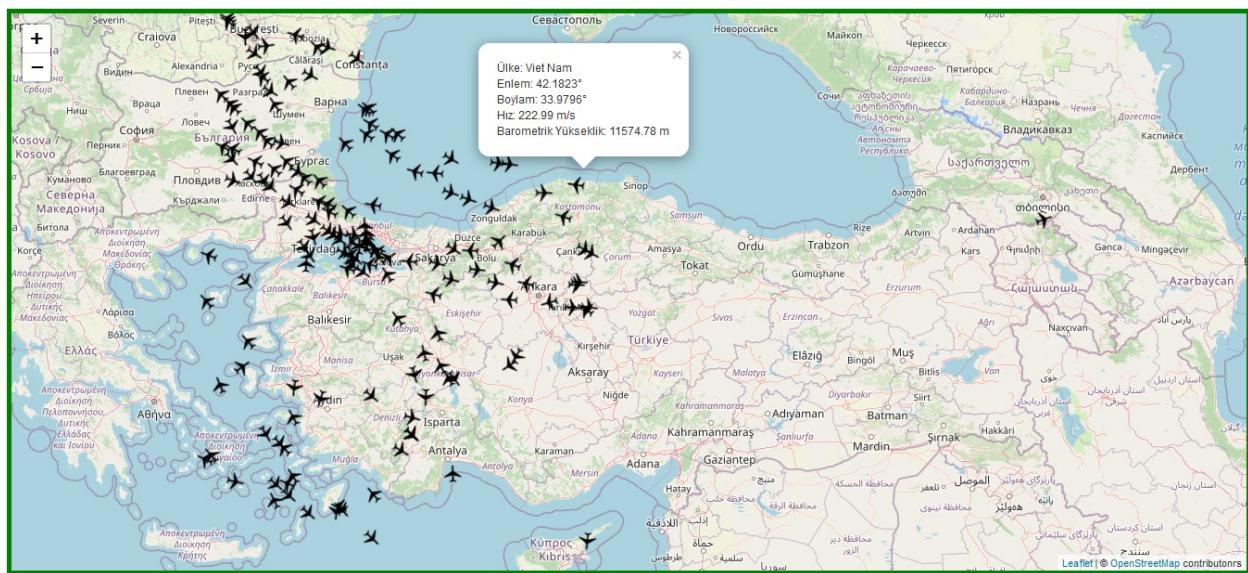
        //Uçakların gidiş yönü açısı => true_track
        markers[icao24] = L.marker([lat, lng], {
            icon: airplaneIcon,
            rotationAngle: true_track
        }).bindPopup(`<p> Ülke: ${origin_country} <br> Enlem: ${lat} <br> Boylam: ${lng} <br> Barometrik Yükseklik: ${baro_altitude} m</p>`);

        markers[icao24].addTo(map);
    });
    setTimeout(() => plotStates(map, markers), 5000);
}
}

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

```

2.2.4. Sonuç



Şekil

2.3. QGIS Üzerinde Canlı Veri İle Yapılan İşlemler

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

2.3.1. Verilerin Python Yardımı İle Belirli Aralıklarda Sunucudan İstenmesi

Aşağıdaki kodlar dizin içerisinde main.py isimli dosya içerisine 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 İle Belirli Aralıklarda Sunucudan İstenmesi

Python kodunu görseldeki gibi çalıştırıyoruz.

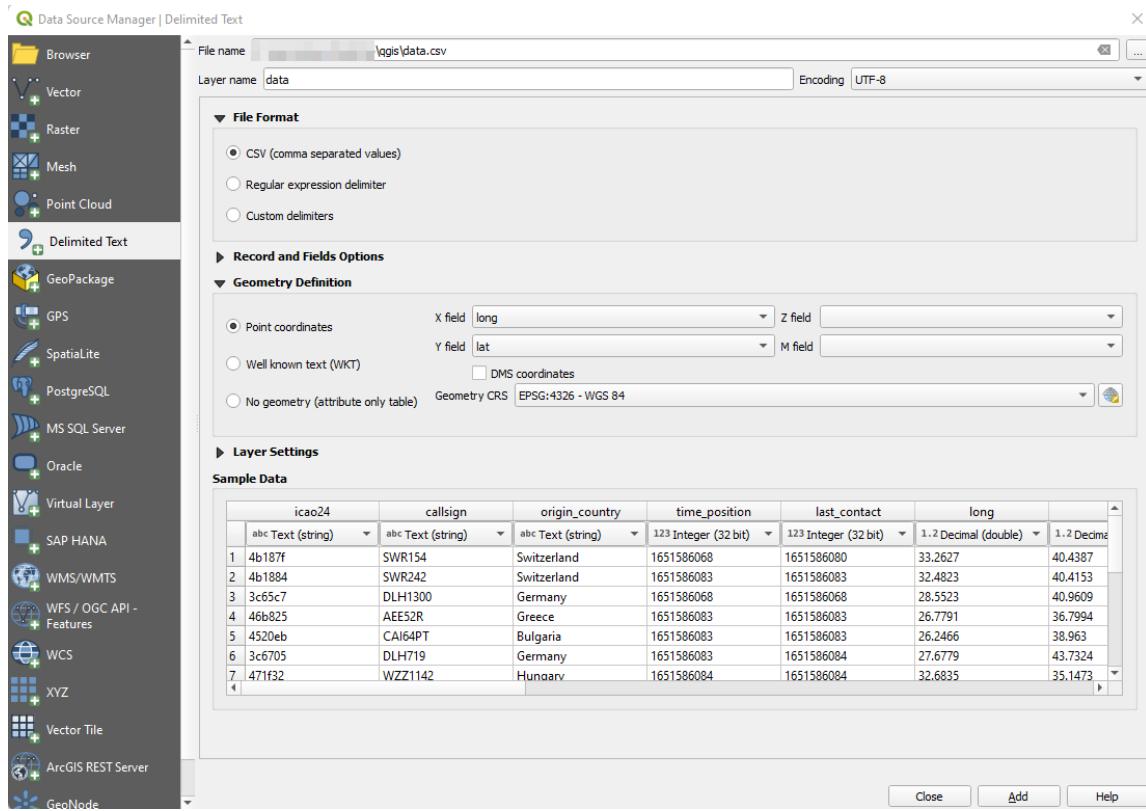
```
PS [REDACTED] qgis> python .\main.py
Get 158 data
Get 159 data
Get 159 data
Get 159 data
```

Şekil

Veri çalıştığı sürece sürekli olarak eğer OpenSky hesabınız varsa 5 saniyede bir yoksa her 10 saniyede bir siz durdurmadığınız sürece veri çekecek.

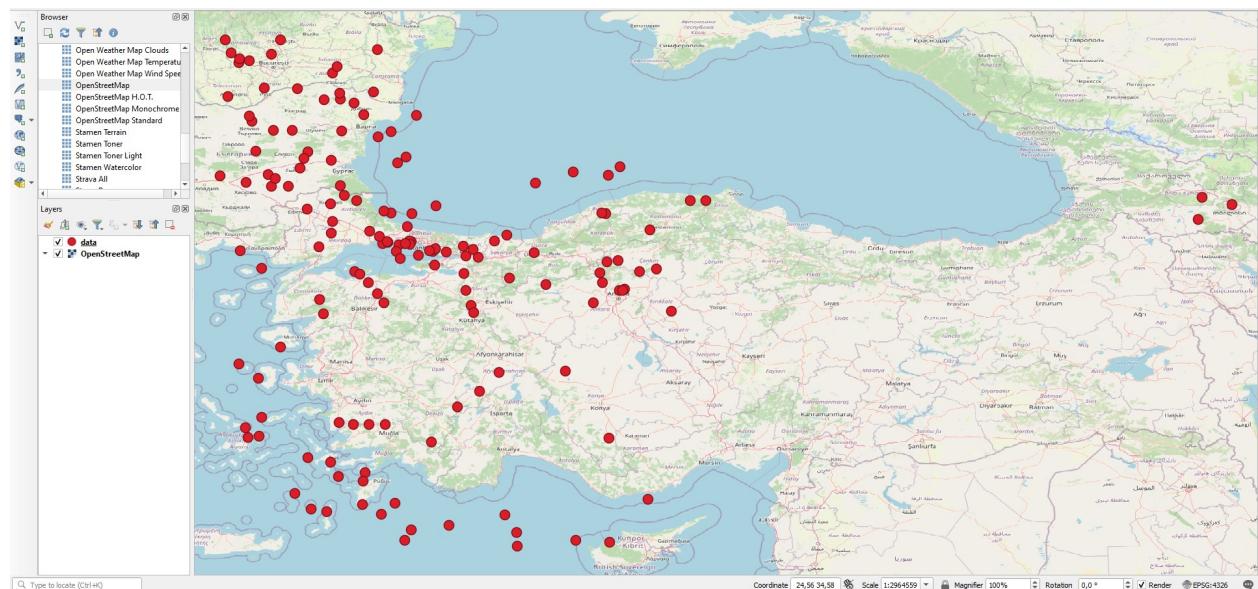
2.3.3. Verilerin QGIS üzerinde görselleştirilmesi

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



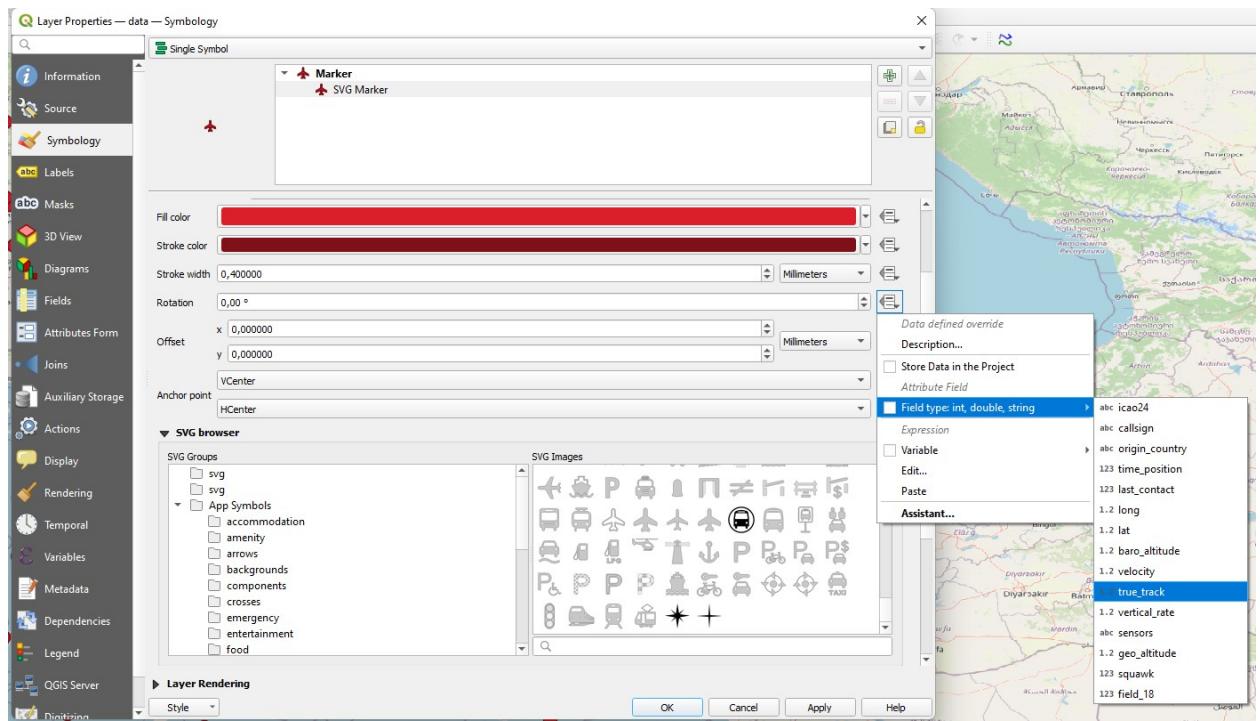
Şekil

OpenStreetMap harita altlığını da ekledikten sonra şu şekilde bir görsel ile karşılaşıyoruz.



Şekil

Noktaların uçak şeklinde ve uçakların gidiş yönünün olması için gerekli ayarları görseldeki gibi yapıyoruz.



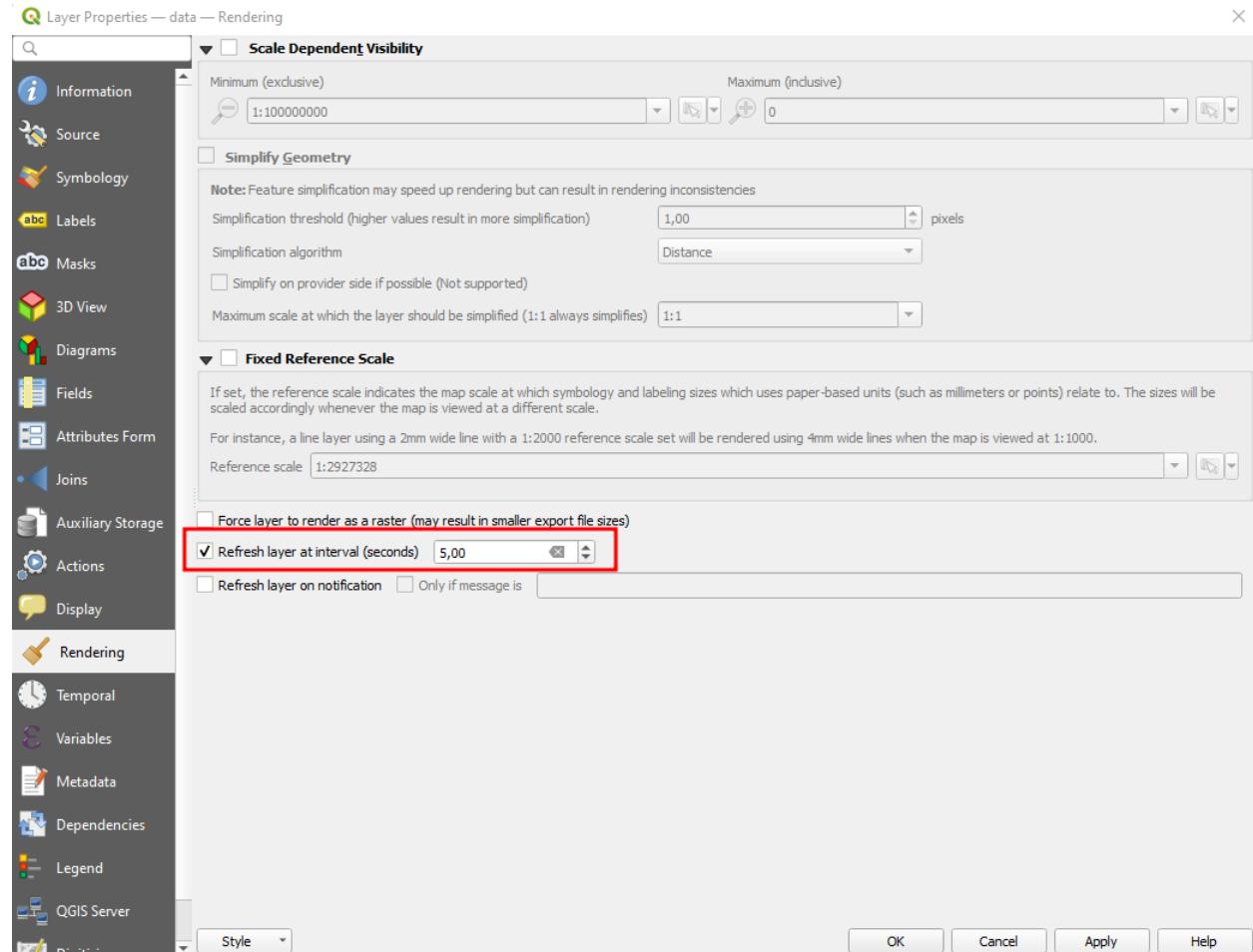
Şekil

Bu işlemleri de yaptıktan sonra şu şekilde bir görsel oluşacaktır.



Şekil

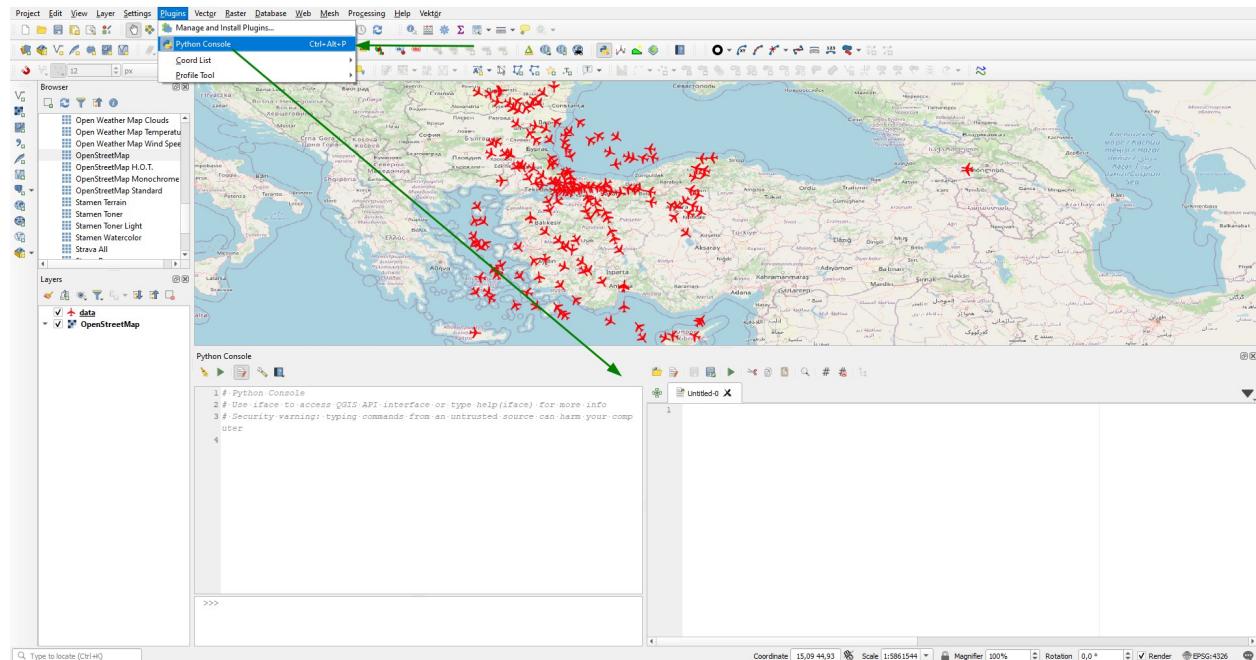
Veri görsellerinin her 5 saniyede bir yenilenmesi için görsel ayarı bölümünden *Rendering* kısmında bulunan *Refresh layer at interval* bölümünü aktif ediyoruz ve kutu içerisinde 5 yazıp *OK* kutucuğuna tıklıyoruz.



Sekil

2.3.4. Verilerin QGIS üzerinde Python Konsole Yardımı İle Grafiksel Görselleştirilmesi

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



Şekil

Python Console bölümüne aşağıdaki gibi bir kodu yazıp kayıt ettikten sonra çalıştırıyoruz.

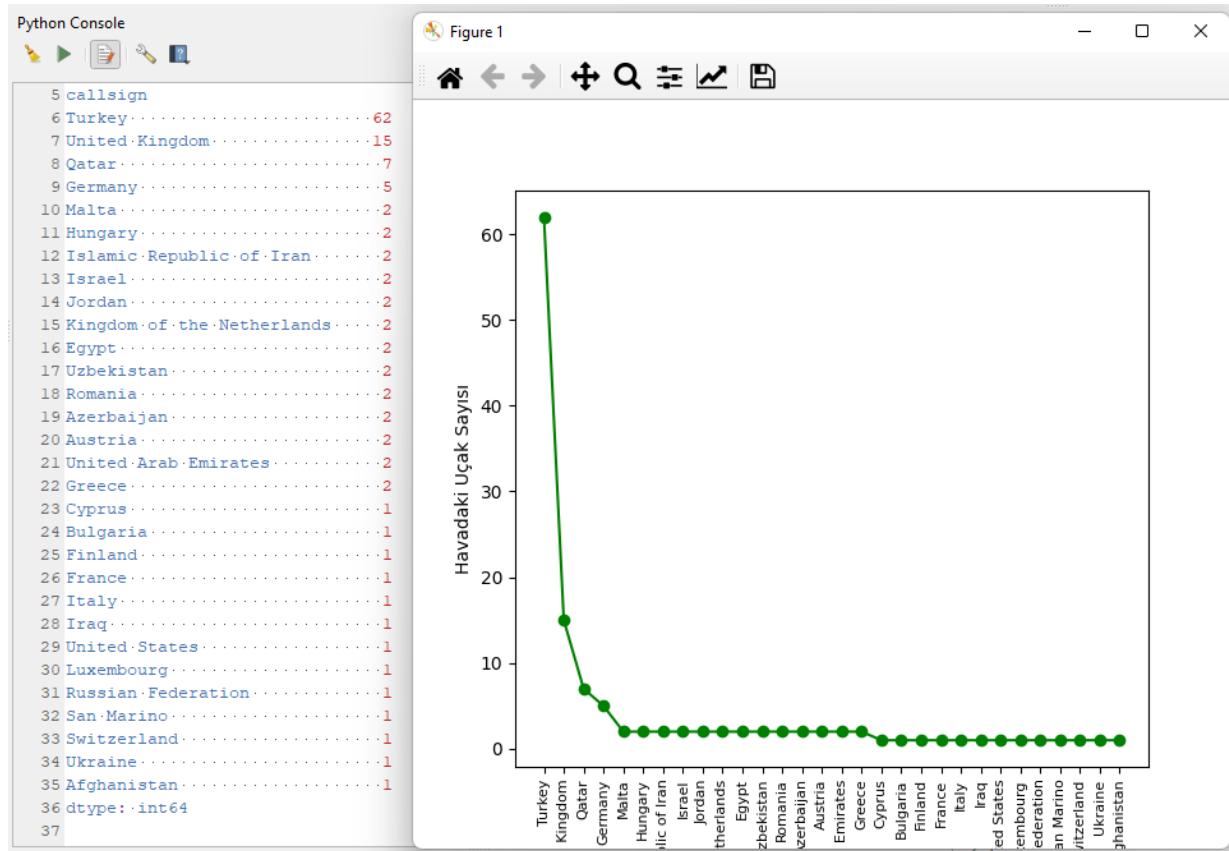
```
""" Kütüphaneler """
import pandas as pd
import matplotlib.pyplot as plt

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

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

2.3.5. Sonuç

Kodu çalıştırıldıkten sonra karşımıza aşağıdaki gibi bir görsel çıkacaktır.



Şekil

KAYNAKLAR

İNTERNET KAYNAKLARI

https://en.wikipedia.org/wiki/History_of_Python

[https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

<https://docs.python.org/3/license.html>

<https://wiki.python.org/moin/>

<https://www.python.org/doc/essays/blurb/>

<https://opensky-network.org/index.php>

<https://openskynetwork.github.io/opensky-api/rest.html>

<https://www.redhat.com/en/topics/api/what-is-a-rest-api>

<https://openskynetwork.github.io/opensky-api/rest.html>

<https://github.com/openskynetwork/raspberry-pi-adsb>

<https://www.hostinger.web.tr/rehberler/html-nedir/>

<https://www.hostinger.web.tr/rehberler/css-nedir/>

<https://www.hostinger.web.tr/rehberler/javascript-nedir/>

<https://tr.wikipedia.org/wiki/QGIS>

<https://www.geodose.com/2020/09/realtime%20live%20data%20visualization%20qgis.html>

EKLER

ÖZGEÇMİŞ