KARADENİZ TEKNİK ÜNİVERSİTESİ

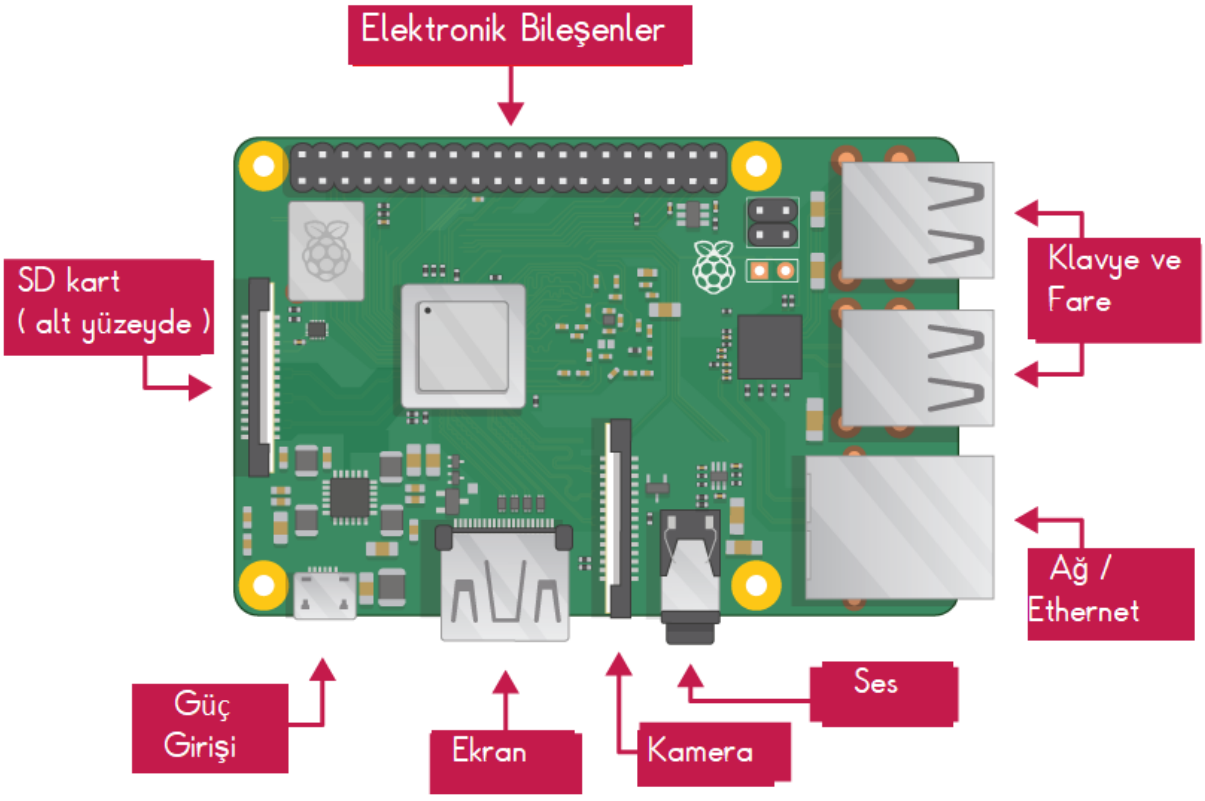
OF TEKNOLOJİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

Raspberry Pi DENEY FÖYÜ

Giriş

Bu deneyde Raspberry Pi ile ilgili bilgiler ve onu kullanabilmek için ihtiyaç bilgiler anlatılacaktır. Deney sonunda servis, zamanlayıcı, mail atma ve python kodu derlenmesi hakkında bilgi sahibi olunacaktır. Ayrıca, gömülü sistemlerin çalışma prensibi ve mini bilgisayarlar hakkında genel bilgiler verilecektir.

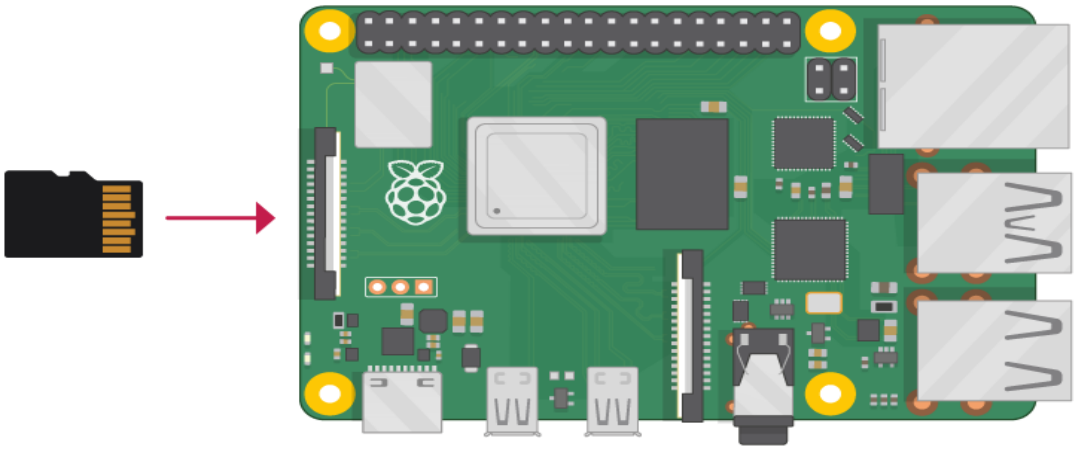


Şekil 1. Temel Bileşenler

Raspberry Pi ‘nin birçok modeli vardır ve çoğu kullanıcı için Raspberry Pi 4 Model B, seçimi en uygun olan modeldir. Raspberry Pi 4 Model B, en yeni, en hızlı ve kullanımı en kolay olanıdır. 1 GB, 2 GB veya 4 GB RAM ile birlikte gelir. Güç kaynağına bağlamak için tüm Raspberry Pi modellerinde bir USB bağlantı noktası bulunur.

Hafıza kartı

Raspberry, tüm dosyalarını ve Raspbian işletim sistemini saklamak için bir SD karta ihtiyaç duyar. En az 8 GB kapasiteye sahip bir microSD kart temel ihtiyaçları karşılamaktadır. Birçok satıcı Raspberry Pi için halihazırda Raspbian kurulmuş olan ve çalışmaya hazır SD kartlar sağlamaktadır.



Şekil 2. microSD Kart Giriş Yuvası

TV veya bilgisayar ekranı

Raspbian masaüstü ortamını görüntülemede ekran ve kabloya ihtiyaç duymaktadır. Ekran bir TV veya bilgisayar monitörü olabilir. Ekranda yerleşik hoparlörler varsa, Pi; sesleri oynatmak için bunları kullanabilir.

Raspberry Kurulumu

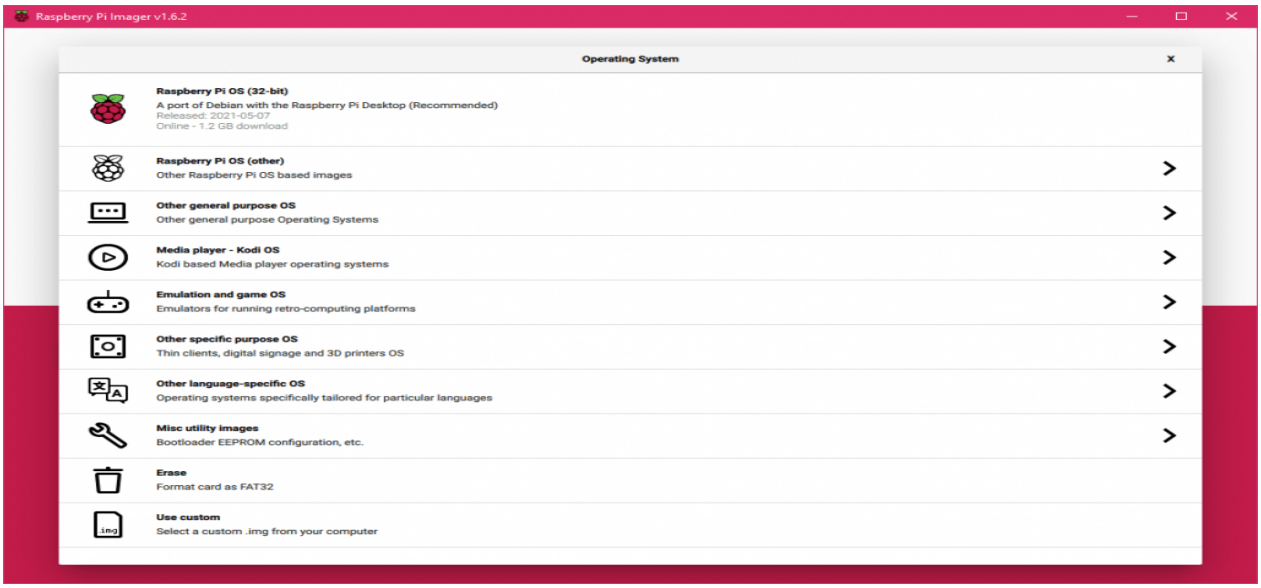
Raspberry Pi OS, Raspberry Pi tarafından geliştirilmiş Debian tabanlı bir işletim sistemidir. Arayüzü ve kullanımı kolay olması sebebiyle Raspberry Pi modellerinde en sık tercih edilen işletim sistemidir.

Bu kurulum yönteminde ilk olarak Raspberry Pi OS imaj dosyasını Raspberry'in web sitesinden indirilir. Web sitesinde 3 farklı işletim sistemi mevcuttur. Bu işletim sistemlerini özellikleri ise şöyle:

- **Raspberry Pi OS with desktop and recommended software:** Bu imajda hem bir arayüz hem de Raspberry Pi tarafından önerilen bazı uygulamalar (LibreOffice gibi) mevcut olarak gelir.
- **Raspberry Pi OS with desktop:** Bu imajda da bir masaüstü mevcut ama önerilen uygulamalar olmadığı için indirme boyutu daha düşüktür.
- **Raspberry Pi OS Lite:** Bu imajda bir masaüstü mevcut değildir, tüm işlemler terminal aracılığıyla yapılır.

Ayrıca yaygın olarak kullanılan Linux dağıtımı olan Ubuntu işletim sistemini de Raspberry Pi'de kullanabiliriz.

Raspberry Pi OS imajı indirildikten sonra Raspberry Pi Imager programı açılır ve "Choose OS" seçilir. İndirilen imaj kullanılacağı için altta bulunan "Use custom" seçeneği tıklanır.

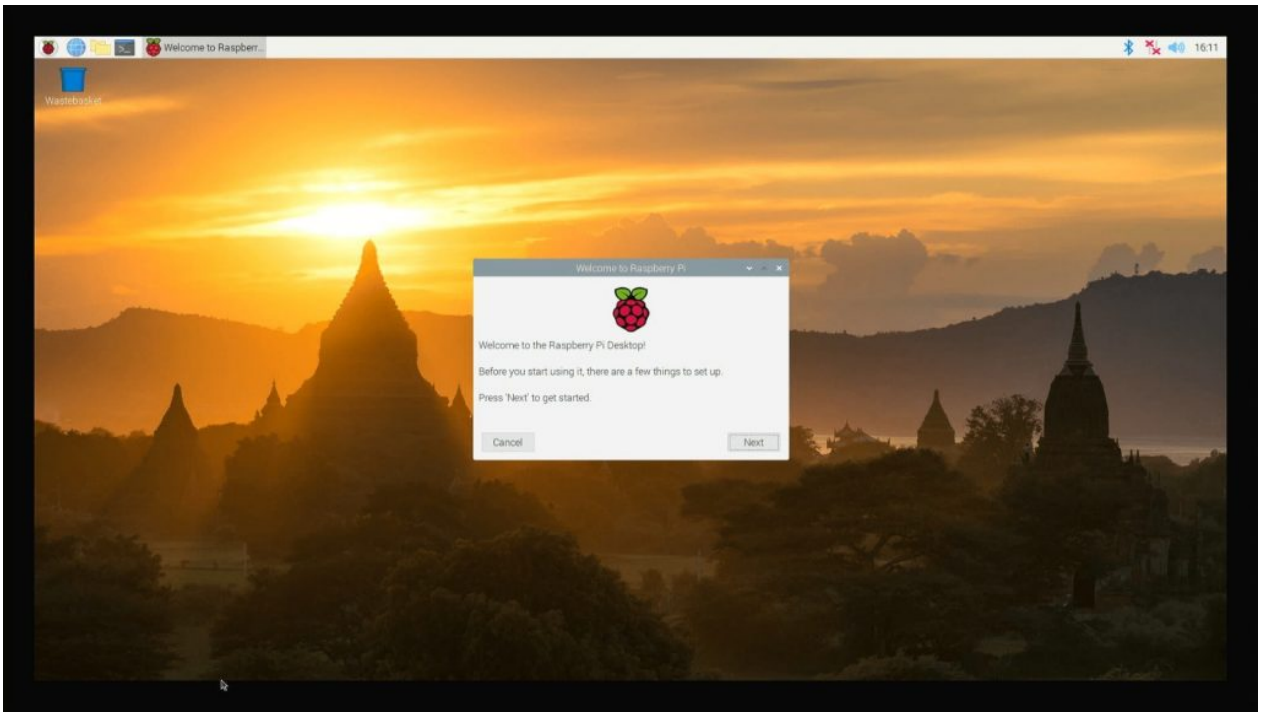


Şekil 3. Raspberry Pi Imager

İndirilen Raspberry Pi OS imaj dosyasının konumu bulunup, *"Open"* seçilir. *"Choose Storage"* butonuna tıklanarak kurulum başlatılır.

Raspberry Pi OS Arayüz Ayarları

Raspberry Pi'nin kullanılabilmesi için klavye, mouse, HDMI ve güç kabloları takılır. Raspberry Pi açılınca bizleri başlangıç ekranı karşılar.



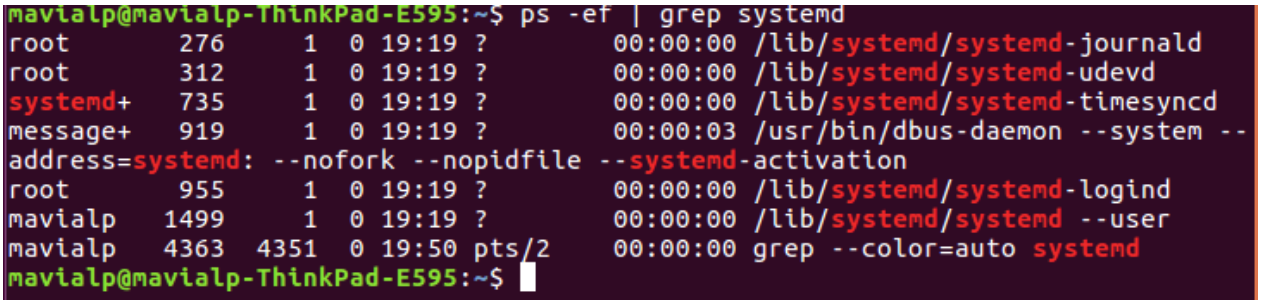
Şekil 4. Raspberry Pi OS Başlangıç Ekranı

Artık Raspberry Pi internete bağlayabilir. Bu ekranda kablosuz ağ seçilir ve gerekli güncellemeler yapılır. Bu deneyde Raspberry sistemleri üzerinde belirli aralıklarla yapılması istenilen işlemlerin tasarımı gerçekleştirilecektir. İçerisinde işletim sistemi yer alan cihazlara ihtiyaç halinde bazı kontrollerin yapılması beklenir. Bu gibi ihtiyaçlar için cron job tarzı yapılar ile sorunlar çözülebilir (Şirketlerden doğum, kutlama vs. mesajlarının gelmesi gibi). Maillerin belirli aralıklarla otomatik gönderilmesi, büyük online satış mağazalarının bizlere gönderdiği doğum/evlilik/ürün mesajları bunlara örnek gösterilebilir.

Ayrıca kullanılan cihazlara tam anlamıyla kontrol etmek biz mühendislerin temel prensibidir. Örneğin, bilgisayarımızın başkaları tarafından açılıp açılmadığını kontrol etmek, bağlı olduğumuz wifi'nin ip değiştiğinde kullanmış olduğumuz cihazın ip adresinin değişmesinin öğrenilmesi vb.

systemd, Linux sistemlerin açılışında hizmetleri kontrol eden yapıdan sorumlu yapıdır. Sistem **boot** edildiğinde belirlenen servisler kullanıcıların yetki seviyelerine göre çalıştırılır. Terminal üzerinden tüm servisler görülmekte ve istenilen servis sonlandırılabilir. Bu yönetimi, systemctl, journalctl, notify, analyze, cgls, cgtop, loginctl ve nspawn olarak adlandırılan araçlar sayesinde gerçekleştirir.

Linux'taki her süreç şeffaf bir şekilde görünür olduğundan, systemd'nin nerede gizlendiğini görerek başlayalım. Terminalde `ps -ef | grep systemd` komutuyla aşağıdaki bilgiler alınır:



Şekil 5. grep komut çıktısı

/etc/systemd/system/ dizinine giderek servis adımızla yeni bir dosyayı **nano** veya **vim** sayesinde oluşturalım. Ardından aşağıdaki örnek şablon üzerinden kullanılabilecek parametrelerin neler oldukları açıklanır.

```
[Unit]
Description=Live
After=network.target
Requires=syslog.target

[Install]
WantedBy=multi-user.target
Alias=hakanaydin.service

[Service]
Type=simple
User=root
Group=wheel
Restart=always
RestartSec=1
ExecStart=/usr/bin/sh /home/hakanaydin/ip.sh
#ExecStart=/usr/bin/python /home/pi/test/ip.py
ExecStop=/bin/kill  ${MAINPID}

# Execute pre and post scripts as root
PermissionsStartOnly=true
```

Description: Servis açıklamasını, **After**: Hangi servisten sonra çalışması gerektiğini söyler. Bu örnekte network servisi başlatıldıktan sonra servisimiz başlar. **Requires**: Servisimizin çalışabilmesi için hangi servisin çalışıyor olması gerektiğini belirleyebiliriz. **WantedBy**: Servisimizin hangi seviyede çalışacağını gösterir. Dolayısıyla, hizmetimizi etkinleştirirsek, multi-user.target.wants klasöründe, bu hizmete sembolik bir bağlantı oluşturulacaktır. Disable edildiğinde de silinir. **Alias**: Servisin ismi, **Type**: simple ile herhangi bir zamanda yalnızca bir

instance çalışacaktır. **User=root**: hangi kullanıcıda çalıştığı; **Group=wheel**: hangi grupta çalışacağı; **Restart**: servis bir şekilde kapanırsa ne zaman restart edileceğini ifade eder. **RestartSec**: Eğer yazılmazsa, sistem default 100ms sonra yeniden başlatmayı dener. **ExecStart**: çalıştırmak istediğimiz binary (sh) ve dosya adını (ip.sh); **ExecStop**: servisi stop ettiğimizde çalışacak komuttur. Burada kill ediyoruz.

Yazılan service ve timer kodları

```
/etc/systemd/system
```

altında yer almaktadır.

Linux dağıtımlarında aktif olan tüm servisleri şu komut ile listeleyebilirsiniz.

```
service --status-all
```

/etc/systemd/system/ altında test.service adında bir servis oluşturulur. Description, örneğin adını; User ise raspberry için pi olmaktadır. Eğer pc üzerinde çalışılacaksa pc ismi olmalıdır.

ExecStart ise hangi kodun ve hangi dosyanın derleneceğini ifade eder. Eğer sadece bu service eklenirse cihaz boot/reboot olduğunda eklenen servis sadece bir kez çalışır. Fakat istenilen her bir dakikada gerekli kontrollerin yapılıp buna göre işlem yapılmasıdır.

test.service

```
[Unit]
Description=Test IPv6 adress

[Service]
User=pi
ExecStart=/usr/bin/python /home/pi/test/ip.py

[Install]
WantedBy=multi-user.target
```

```
import pathlib
import netifaces as ni
from email.mime.multipart import MIMEMultipart
```

```
from email.mime.text import MIMEText
import smtplib
import re

#############################################################################
def test_network():
    interfaces = ni.interfaces()
    ip = ""
    for i in interfaces:  # Will cycle through all available interfaces and che>
        if i == 'teredo':
            ni.ifaddresses(i)
            ip = ni.ifaddresses(i)[ni.AF_INET6][0]['addr']
            print ("IP address: " + ip)
    return ip
#############################################################################
my_old_teredo_ipv6=""
ip_path = "/home/pi/teredo/ip.txt"
#############################################################################
my_teredo_ip_addr6 = test_network()
fileExist = checkFileExistByPathlib(ip_path)
# If file do not exist then create it.
if (not fileExist):
    createNewFile(ip_path)

with open(ip_path, "r") as my_file:
    if check(ip_path) == 0:
        print ("file is empty")
        with open(ip_path, "w") as first_ip_write_txt:
            first_ip_write_txt.seek(0)
            first_ip_write_txt.write(str(my_teredo_ip_addr6))
    else:
        for satir_file in my_file:
            my_old_teredo_ipv6 = satir_file

    my_file.close()

if my_old_teredo_ipv6 != str(my_teredo_ip_addr6):
    print ("my teredo ipv6 addr changed...")
    with open(ip_path, "a") as my_filex:
        if check(ip_path) == 0:
            my_filex.write(str(my_teredo_ip_addr6))
            my_old_teredo_ipv6 = str(my_teredo_ip_addr6)
            with open("/home/pi/x/mailler", 'r') as dosya:
                for satir in dosya:
```

```
            satir = satir.replace("\n", "")
            sentences = re.split(r':', satir)
            print(sentences[0], "----", sentences[1])
            try:
                mail = smtplib.SMTP('smtp.gmail.com', 587)
                msg = MIMEMultipart()
                msg['From'] = "mailadresi"
                msg['To'] = "mailadresi"
                msg['Subject'] = "IP Address"
                body_text = MIMEText(my_old_teredo_ipv6, "plain")
                msg.attach(body_text)

                mail.ehlo()
                mail.starttls()
                mail.login('mail', 'sifre')
                mail.sendmail(msg['from'], sentences[1], msg.as_string())
                print(sentences[1], "gonderildi")
                mail.close()
            except:
                print(sentences[1], "gonderilmedi...")

    else:
        print("dosya full")
        my_filex.seek(0)  # sets  point at the beginning of the file
        my_filex.truncate()  # Clear previous content
        my_filex.write(str(my_teredo_ip_addr6))
        my_old_teredo_ipv6 = str(my_teredo_ip_addr6)
        with open("/home/pi/x/mailler", 'r') as dosya:
            for satir in dosya:
                satir = satir.replace("\n", "")
                sentences = re.split(r':', satir)
                print(sentences[0], "----", sentences[1])
                try:
                    mail = smtplib.SMTP('smtp.gmail.com', 587)
                    msg = MIMEMultipart()
                    msg['From'] = "mail"
                    msg['To'] = "mail"
                    msg['Subject'] = "IP Address"
                    body_text = MIMEText(my_old_teredo_ipv6, "plain")
                    msg.attach(body_text)

                    mail.ehlo()
                    mail.starttls()
                    mail.login('mail', 'sifre')
```

```
                mail.sendmail(msg['from'], sentences[1], msg.as_string())
                print(sentences[1], "gonderildi")
                mail.close()
            except:
                print(sentences[1], "gonderilmedi...")

########################################
```

Yukarıda ip.py dosyası mevcuttur. Bu dosya raspberry nin reboot ya da yeniden ip adresi alması durumunda mailler dosyası içindeki bireylere mail atan bir python dosyasıdır. Bu şekilde 60 saniyede bir kontrol yapılarak teredo ipv6 adresi bireylere atılıp; bireyler bu ipv6 adresi üzerinde ssh bağlantısı sağlamaktadır.

Bu servise ek olarak timer da tanımlanır. Koddan da anlaşılacağı üzere her 60 saniyede bir test.service çalıştırıp ip.py kodu aktif edilmektedir. İstenirse ip kodu sonsuz döngüye alınarak timer kullanılmasına gerek kalmayabilir.

`sudo systemctl enable test.service` ve `sudo systemctl enable test.timer` ile eklenilen bağlantılar aktif edilir. Daha sonrada `sudo systemctl daemon-reload` ile servisler yeniden başlatılır.

Not: ip.py kodlarını tamamı verilmemiştir. Eksik olan kısımlar mail atma ve dosyadan mail adresini çekme kısmıdır. Öğrenciler kodların tamamını incelemeli ve sonuçları araştırmalıdır. İlgili projenin detaylı anlatımı ve kod bloğunun tamamı https://yazilimdnyasi.wordpress.com/2022/02/26/raspberry-uzerinde-service-kontrolu/, sitesi içerisinde mevcuttur.

Kaynak

1. https://maker.robotistan.com/raspberry-pi-kurulum
2. https://yazilimdnyasi.wordpress.com/2022/02/26/raspberry-uzerinde-service-kontrolu/
3. https://www.technopat.net/2018/07/14/raspberry-pi-hakkinda-bilmeniz-gereken-her-sey/
4. https://www.limonhost.net/makaleler/genel/raspberry-pi-nedir-raspberry-pi-ozellikleri/
5. https://kerteriz.net/linux-systemd-detaylari-ve-sunucuda-servis-yazmak/