



MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
BİTİRME PROJESİ

YÜZ TANIMALI KAPI KİLİDİ

ZEYNEP BUSE BAŞOĞLU-202523053

DANIŞMAN

DR.ÖĞR. ÜYESİ HANDAN GÜRSOY DEMİR

OCAK 2024

İSKENDERUN TEKNİK ÜNİVERSİTESİ

ÖZET

Yapay zekâ ve teknoloji alanındaki gelişmeler gün geçtikçe hızlanırken, bu teknolojiler günlük hayatımızda da oldukça geniş bir yer edinmeye devam etmektedir. Özellikle, neredeyse bütün cihazların internet bağlantısı özelliğine kavuştuğu günümüzde, güvenlik amacıyla sunulan teknolojik çözümlerin de bu teknolojilerle daha etkili bir güce kavuştuğunu söyleyebiliriz.

Bu kapsamda, gelişmiş özelliklere sahip güvenlik sistemleri ve akıllı kamera sistemleri, gün geçtikçe daha yaygın bir kullanım alanına sahip olacaktır.

Projede, yüz tanıma özellikli bir güvenlik sistemi tasarımı gerçekleştirilecektir. Bu sistem, kameradan aldığı görüntüyü sürekli işleyecek ve önceden tanımlı kişinin yüzü kameradan tespit edildiğinde kilit açılacaktır. Sistemin elektrik kesintilerinde de çalışmayı sürdürebilmesi için, güç kaynağı olarak şarj edilebilir piller kullanılacaktır.

İÇİNDEKİLER

	sayfa numarası
1-) ŞEKİL LİSTESİ	4
2-) GİRİŞ	6
3-) MATERYAL VE YÖNTEM	9
3.1. Kullanılan Malzemeler ve Teknik Özellikleri	9
3.2. Raspberry Pi Ayarlarının Yapılması	11
3.3. Bilgisayar Ağ Bağlantısının Raspberry Pi ile Paylaştırılması	17
3.4. Virtual Environment ve Python Kütüphanelerinin Kurulumu	19
3.5. Yazılım Aşaması	29
4-) TARTIŞMA VE SONUÇ	34
5-) KAYNAKLAR	37

1. Şekil Listesi

Şekil	Sayfa Numarası
Şekil 2.1. Yüz Tanıma Özellikli Güvenlik Sistemi	8
Şekil 3.1.1. Raspberry Pi 2 Model B [3]	9
Şekil 3.1.2. Raspberry Pi 2 Model B Pinler ve İşlevleri	10
Şekil 3.1.3. USB Kamera	11
Şekil 3.1.4. Solenoid Valf	11
Şekil 3.1.5. L298 Motor Sürücü	12
Şekil 3.2.1. SSH Ayarları	12
Şekil 3.2.2. LXDE Masaüstü Ortamı	13
Şekil 3.2.3. Teminalde raspi-config komutunun girilmesi	13
Şekil 3.2.4. Raspberry Ayar Menüsü	14
Şekil 3.2.5. Raspberry Ayar Menüsünde Interface Ayarlarına Gidilmesi	14
Şekil 3.2.6. Interface Ayarlarında Kamera Arayüzü Ayarlarının Yapılması	15
Şekil 3.2.7. Kamera Arayüzünün Aktif Edilmesi	15
Şekil 3.2.8. Kamera Arayüzünün Aktif Edildiğini Belirten Mesaj	16
Şekil 3.2.9. Ayar Menüsünden Çıkılması	16
Şekil 3.3.1. Ağ ve İnternet Ayarları	17
Şekil 3.3.2. Kablosuz Bağlantı Özellikleri	18
Şekil 3.3.3. Kablosuz Bağlantı Paylaşım Ayarları	18
Şekil 3.4.1. Sanal Ortamın Kurulacağı Dizinin Açılması	19
Şekil 3.4.2. Sanal ortamın oluşturulması	20
Şekil 3.4.3. Komut Çıktısı	20
Şekil 3.4.4. Sanal Ortamın Aktif Edilmesi	21
Şekil 3.4.5. dlib modülü kurulum komutu	22
Şekil 3.4.6. dlib modülünün kurulumu sonrası alınan konsol çıktısı	23
Şekil 3.4.7. pillow modülünün kurulumu	23
Şekil 3.4.8. numpy modülünün kurulumu	23
Şekil 3.4.9. face-recognition modülünün kurulumu	24
Şekil 3.4.10. piwheels.org sitesi	24
Şekil 3.4.11. opencv modülünün kurulması	25

Şekil 3.4.12. Rpi.GPIO modülünün kurulumu	25
Şekil 3.4.13. Thonny IDE	26
Şekil 3.4.14. Thonny IDE Ayar Menüsü	26
Şekil 3.4.15. Interpreter sekmesi	27
Şekil 3.4.16. Sanal ortam dosya yolu ayarları	27
Şekil 3.4.17. Sanal ortam dosya yolu ayarlarının yapılması	28
Şekil 3.4.18. Interpreter ayarlarının son hali	28

2.GİRİŞ

Günümüzde teknolojinin hızla ilerlemesiyle birlikte güvenlik endişeleri de artmaktadır. İşletmeler ve bireyler, verilerini ve değerli varlıklarını korumak için daha güçlü ve etkili güvenlik sistemleri arayışındadır. İşte bu noktada yapay zekanın güvenlik sistemlerinde önemli bir rol oynadığını görmekteyiz.

Yapay zekâ, karmaşık algoritmalar kullanarak verileri analiz edebilen ve öğrenebilen bir teknolojidir. Bu yetenekleri sayesinde, yapay zekâ tabanlı güvenlik sistemleri geleneksel yöntemlere göre daha etkin ve akıllı bir şekilde tehditleri tespit edebilmektedir.

Birinci avantajı, yapay zekanın anormal davranış desenlerini tanıyabilmesidir. Bu sistemler, normal kullanım modellerini öğrenerek anormal aktiviteleri tespit edebilir ve potansiyel tehditleri belirleyebilir. Örneğin, bir ağa sızma girişimi veya yetkisiz erişim denemesi gibi durumlar tespit edildiğinde, yapay zekâ hızla tepki verebilir ve saldırıyı engelleyebilir.

İkinci olarak, yapay zekâ tabanlı güvenlik sistemleri sürekli olarak öğrenme yeteneğine sahiptir. Bu sistemler, verilerden sürekli olarak beslenerek yeni tehditleri tanımayı ve bu tehditlere karşı kendini güncellemeyi öğrenebilir. Böylece, güvenlik açıkları zamanla kapatılabilir ve savunma mekanizmaları sürekli olarak iyileştirilebilir.

Ayrıca, yapay zekâ ile geliştirilen güvenlik sistemleri hızlı tepki süreleri sunar. Geleneksel güvenlik yöntemleri genellikle insan müdahalesine dayanırken, yapay zekâ sistemleri anlık olarak olayları tespit edip analiz edebilir ve hızlı bir şekilde tedbir alabilir. Bu da saldırıların yayılmasını engellemede büyük bir avantaj sağlar.

Yapay zekâ ile geliştirilen güvenlik sistemleri günümüzün karmaşık ve sofistike tehditlerine karşı etkili bir çözüm sunmaktadır. Bu sistemler, analitik yetenekleri, sürekli öğrenme kabiliyetleri ve hızlı tepki süreleri sayesinde güvenliği artırmakta ve potansiyel riskleri minimize etmektedir. İlerleyen dönemlerde yapay zekâ tabanlı güvenlik sistemlerinin daha da gelişeceğini ve yaygınlaşacağını söyleyebiliriz.

Yapay Zekâ Destekli Sistemler

Günümüzde teknoloji hızla ilerledikçe, güvenlik önlemleri de bu hızla evrim geçirmektedir. Geleneksel güvenlik yöntemleri artık yeterli gelmemekte ve daha sofistike bir yaklaşım gerektirmektedir. Bu noktada, yapay zekâ destekli sistemlerin ortaya çıkması güvenlik alanında yeni bir boyutun keşfedilmesini sağlamıştır.

Yapay zekâ, karmaşık algoritmaları ve büyük veri analizini kullanarak, anlık olarak riskleri tespit etme ve önleme yeteneği sunar. Bu sistemler, mevcut güvenlik önlemlerinden daha akıllıca hareket edebilir ve olağandışı durumları önceden tahmin edebilir. Örneğin, bir yapay zekâ destekli kamera sistemi, belirli bir fiziksel saldırı veya hırsızlık girişimini algılayabilir ve derhal uygun bir tepki verebilir.

Yapay zekâ destekli sistemler aynı zamanda veri analizi konusunda büyük bir avantaj sağlar. İnsan gözüyle kaçırılacak ince ayrıntıları algılayabilir ve anlamlı sonuçlar çıkarabilirler. Bu sayede, şüpheli davranışların tespiti ve tehditlerin önceden engellenmesi mümkün hale gelir. Örneğin, bir yapay zekâ destekli siber güvenlik sistemi, ağ trafiğini analiz ederek zararlı yazılımları tespit edebilir ve ağın güvenliğini sağlayabilir.

Yapay zekâ destekli sistemlerin bir diğer avantajı ise sürekli öğrenme yetenekleridir. Bu sistemler, zamanla kendilerini geliştirerek daha etkili hale gelirler. Bir yapay zekâ destekli güvenlik sistemi, saldırıların yeni modellerini tanımak ve buna göre önlem almak için sürekli olarak verileri analiz eder. Bu sayede, güvenlik açıklarının daha hızlı bir şekilde kapatılması ve gelecekteki tehditlere karşı daha iyi hazırlıklı olunması sağlanır.

Akıllı Kameralar ve Güvenlik İyileştirmeleri

Akıllı kameralar ve yapay zekâ teknolojileri, güvenlik sektöründe devrim yaratıyor. Bu yenilikçi kombinasyon, olağanüstü bir gözetim deneyimi sunarken güvenlik önlemlerini de büyük ölçüde artırıyor. Geleneksel güvenlik sistemleriyle karşılaştırıldığında, akıllı kameralar benzersiz bir şekilde hareket ediyor ve insanların günlük yaşamlarını daha güvenli hale getiriyor.

Akıllı kameraların temel özelliği, yapay zekâ algoritmalarının kullanılmasıyla nesneleri tanıma ve analiz etme yetenekleridir. Bu kamera sistemleri, görüntüleri gerçek zamanlı olarak işleyerek tehlikeli durumları algılayabiliyor ve hızlı bir şekilde yanıt verebiliyor. Örneğin, bir akıllı kamera, mağaza içinde şüpheli davranışlar tespit ettiğinde anında alarm verebilir veya güvenlik görevlisine uyarı gönderebilir.

Bu teknolojinin gücü aynı zamanda kayıp eşyaların bulunması ve suçluların tespiti gibi soruşturma süreçlerini de kolaylaştırıyor. Yapay zekâ destekli kameralar, yüz tanıma özelliği sayesinde kişilerin

kimliklerini belirleyebilir ve veri tabanlarıyla eşleştirebilir. Bu, daha önce suç işlemiş veya aranan kişileri tespit etme konusunda önemli bir avantaj sağlar.

Akıllı kameraların güvenlik sistemlerine getirdiği bir diğer yenilik ise analitik yetenekleri. Yapay zekâ algoritmaları, görüntülerdeki hareketlilik, kalabalık yoğunluğu ve olağandışı durumları algılayabilir. Örneğin, bir akıllı kamera, bir çantanın terk edildiğini veya izinsiz girişlerin olduğunu anında fark edebilir. Bu sayede, potansiyel tehlikelerin tespit edilmesi ve hızlı müdahale imkânı sağlanır.

Akıllı kameraların kullanım alanları giderek genişliyor. Havaalanları, alışveriş merkezleri, bankalar, kamu binaları ve konut kompleksleri gibi birçok yerde bu teknoloji yaygın olarak kullanılıyor. İnsanların güvenliğini artırmak ve suç oranlarını azaltmak için akıllı kameraların entegrasyonu büyük bir önem taşıyor.

Yapay zekâ destekli akıllı kameralar, güvenlik sektöründe devrim niteliğinde bir ilerleme sağlamaktadır. Bu teknoloji, olağanüstü gözetim yetenekleriyle güvenlik iyileştirmeleri sunarken aynı zamanda hızlı ve etkili müdahale imkânı sağlamaktadır. Akıllı kameraların günlük hayatımızdaki yaygın kullanımı, daha güvenli bir gelecek için umut verici bir adımdır.



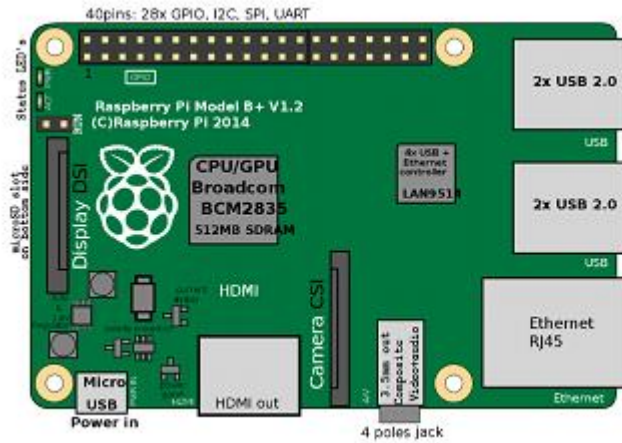
Şekil 2.1. Yüz Tanıma Özellikli Güvenlik Sistemi

3. MATERYAL VE YÖNTEM

3.1.Kullanılan Malzemeler ve Özellikleri

- Raspberry Pi 2 Model B Mini Bilgisayar

Raspberry Pi 2 Model B bilgisayar üzerindeki USB, ethernet, HDMI bağlantı yuvalarının ve GPIO pinlerinin yerleşimleri şekil3.1.1’de görüldüğü gibidir



Şekil 3.1.1. Raspberry Pi 2 Model B [3]

Raspberry Pi 2 Teknik Özellikleri:

- BROADCOM BCM2836 ARMv7 QUAD CORE OC (Dört Çekirdekli)
- 900 MHz İşlemci Hızı
- 1 GB RAM
- 10/100 Ethernet RJ45 JACK
- 4 x USB2.0 port
- microSD kart yuvası
- 40 Pin GPIO
- 27 x GPIO
- UART
- I2C
- SPI - 2 CS ucu
- 3.3V

- 5V
- GROUND
- Güç girişi: 5V 600mA mikro-USB adaptör ve GPIO üzerinden
- Windows 10, RASPIAN DEBIAN, FEDORA, ARCH, RISC OS desteği
- Kart boyutları 8.6cm x 5.6cm x 2.0cm



Şekil 3.1.2. Raspberry Pi 2 Model B Pinler ve İşlevleri

Raspberry Pi 2 Model B’ de bulunan genel amaçlı giriş-çıkış pinlerinin (GPIO) numaralandırılışları ve işlevleri şekil 3.1.2’de görülmektedir. UART ve SPI haberleşme birimleri bulunan mini bilgisayar, aynı zamanda lojik sinyal yazma ve okuma yapıp, kamera ve ekran arayüzleri için bağlantı soketlerine sahiptir.

•USB Web Kamerası

Yüz tanıma sistemi için, USB kamera kullanılacaktır. Kameradan alınan görüntü işlenecektir ve kameradan alınan görüntüdeki yüzün, sisteme önceden tanımlanmış bulunan kişinin yüzü ile eşleşmesi halinde, güvenlik sisteminin kilidi açılacaktır.



Şekil 3.1.3. USB Kamera

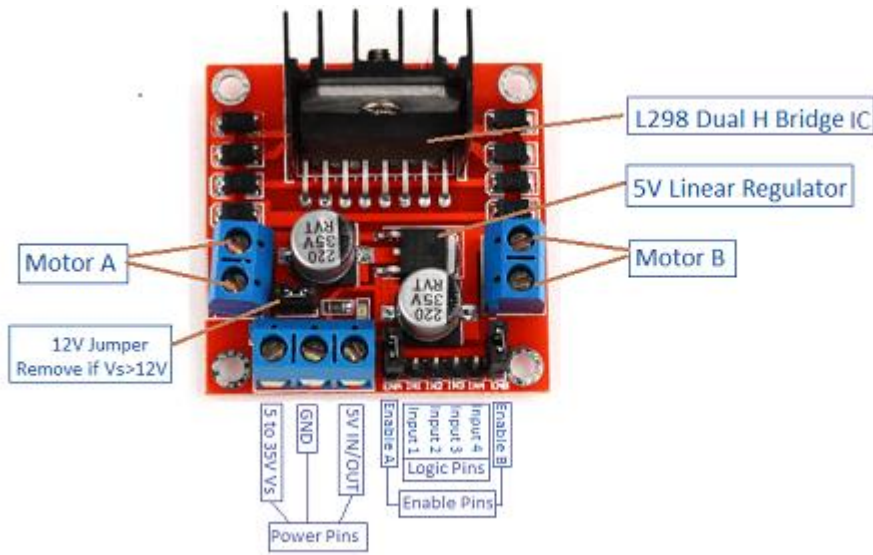
• Kapı Kilidi (Elektromıknatıs Elektromanyetik Solenoid)



Şekil 3.1.4. Solenoid Valf

Kilit mekanizması olarak, şekil 3.1.4'te görülen elektro mıknatıslı solenoid kullanılacaktır.

- L298 Motor Sürücü

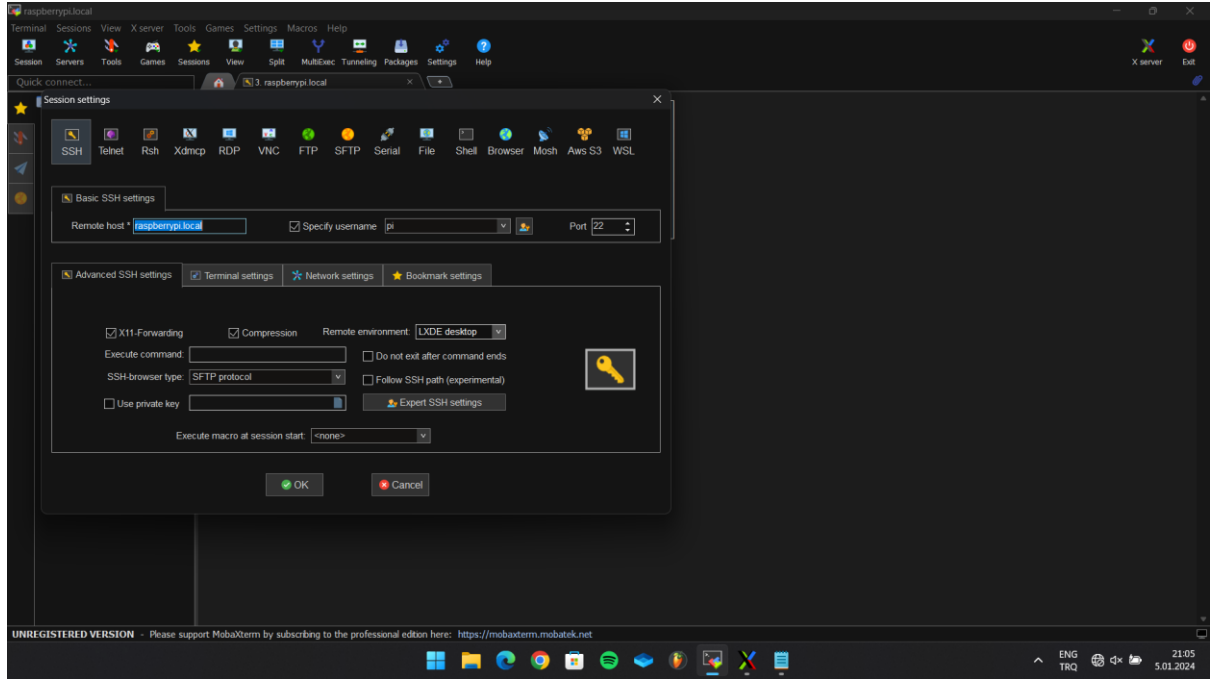


Şekil 3.1.5. L298 Motor Sürücü

3.2. Raspberry Pi Ayarlarının Yapılması

SSH Ayarlarının Yapılması

Raspberry Pi bilgisayar ile laptop bilgisayar arasında SSH bağlantısı kurulacaktır. Böylece, bir monitör, klavye veya mouse 'a ihtiyaç duyulmaksızın Raspberry 'e erişim sağlanabilir ve projenin kodlama aşaması gerçekleştirilebilir.



Şekil 3.2.1. SSH Ayarları

Resimde görüldüğü gibi, Mobaxterm isimli bir program kullanılarak SSH SESSION oluşturulur. Bu program sayesinde monitör kullanmadan kodlama yapabilecek ve ayrıca Raspberry ile laptop arasında dosya aktarımı gerçekleştirebilecektir.

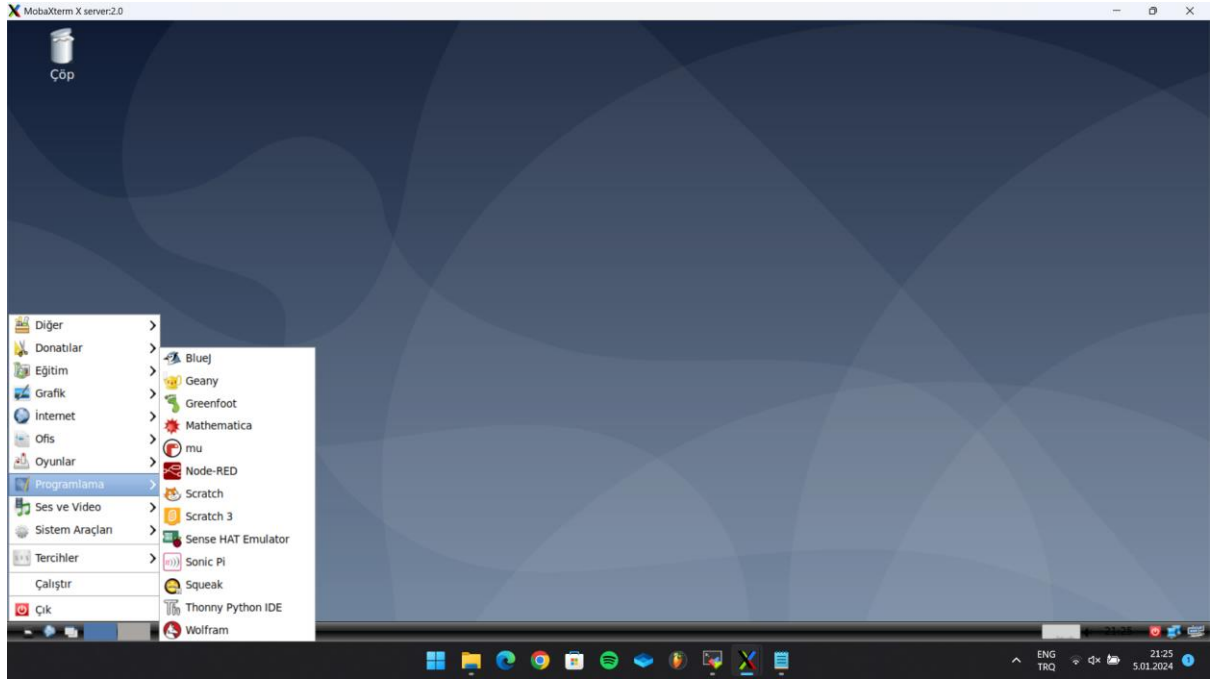
Resimde görüldüğü gibi, host name ve username alanlarını aşağıdaki gibi doldurduk:

Remote Host Name : raspberry.local

Username: pi

Remote environment seçeneğini ise LXDE desktop olarak seçtik.

Mobaxterm programında yeni bir session oluşturduğumuzda, Remote environment seçeneği varsayılan olarak terminal ekranı değerine sahiptir. Bu seçenek seçiliyken, Raspberry'e bağlandığımızda bir terminal ekranı açılır ve komut satırını kullanırız.

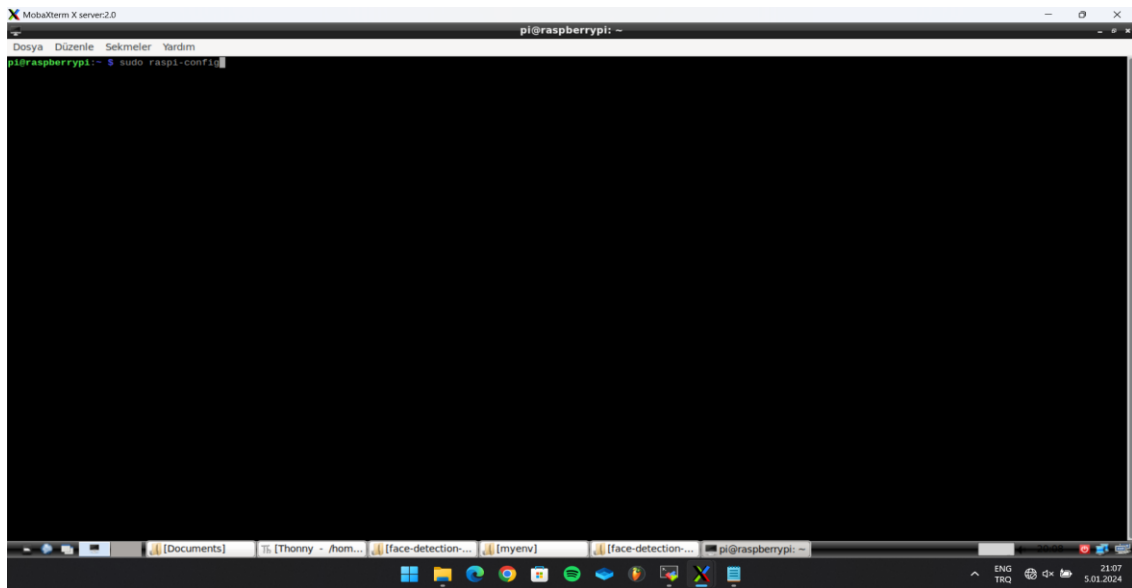


Şekil 3.2.2. LXDE Masaüstü Ortamı

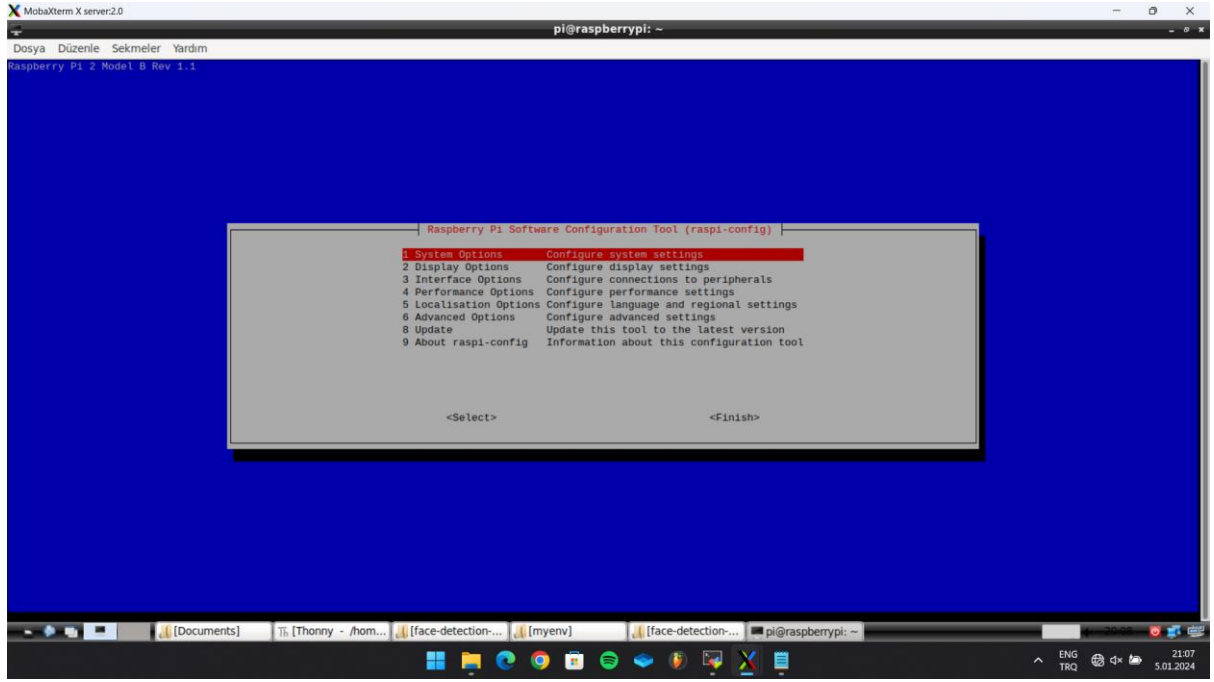
LXDE Desktop seçeneği seçildiğinde, yukarıda 3.1.2.2. numaralı resimde görüldüğü gibi görsel bir masaüstü arayüzü ile Raspberry'i kullanabilmekteyiz. Daha pratik bir kullanım sağlayacağından, LXDE Desktop üzerinden kodlayacağız.

Interface Ayarları

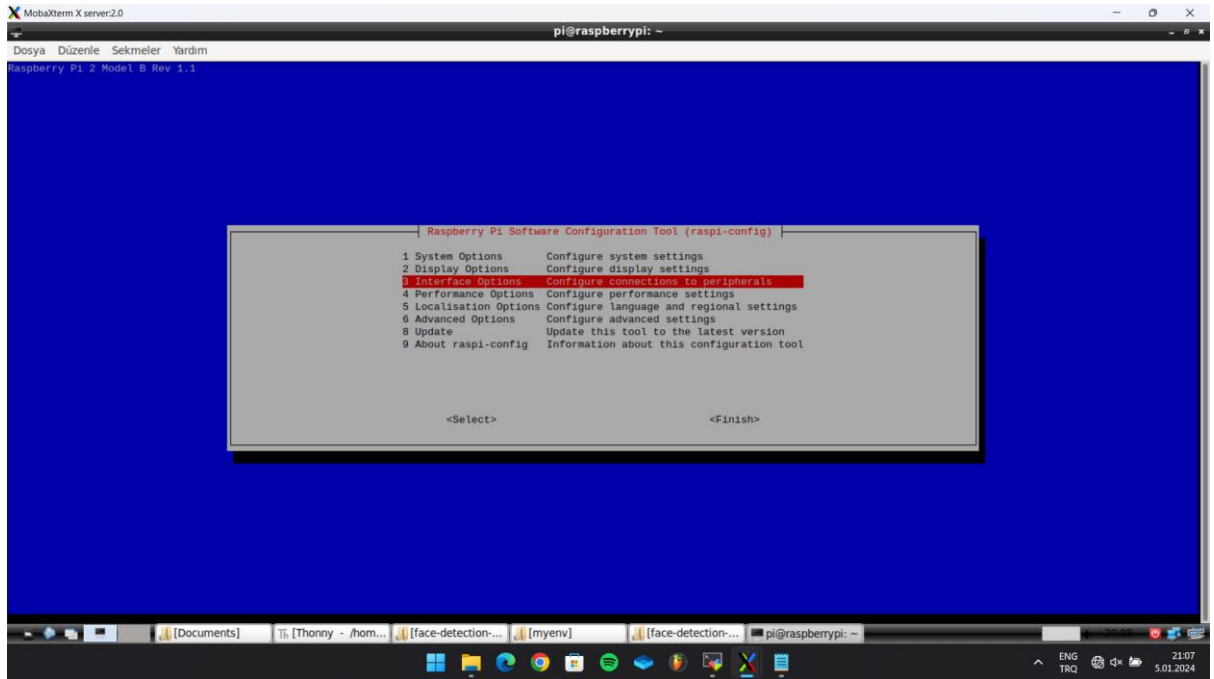
Raspberry pi ile kamera modülü kullanabilmek için, terminal ekranında "sudo raspi-config" komutunu girerek açılan menüde Interface Options bölümünde Camera Interface aktif hale getirilir. Konfigürasyon tamamlandığında, Finish ile menüden çıkılır ve ayarlar tamamlanmış olur.



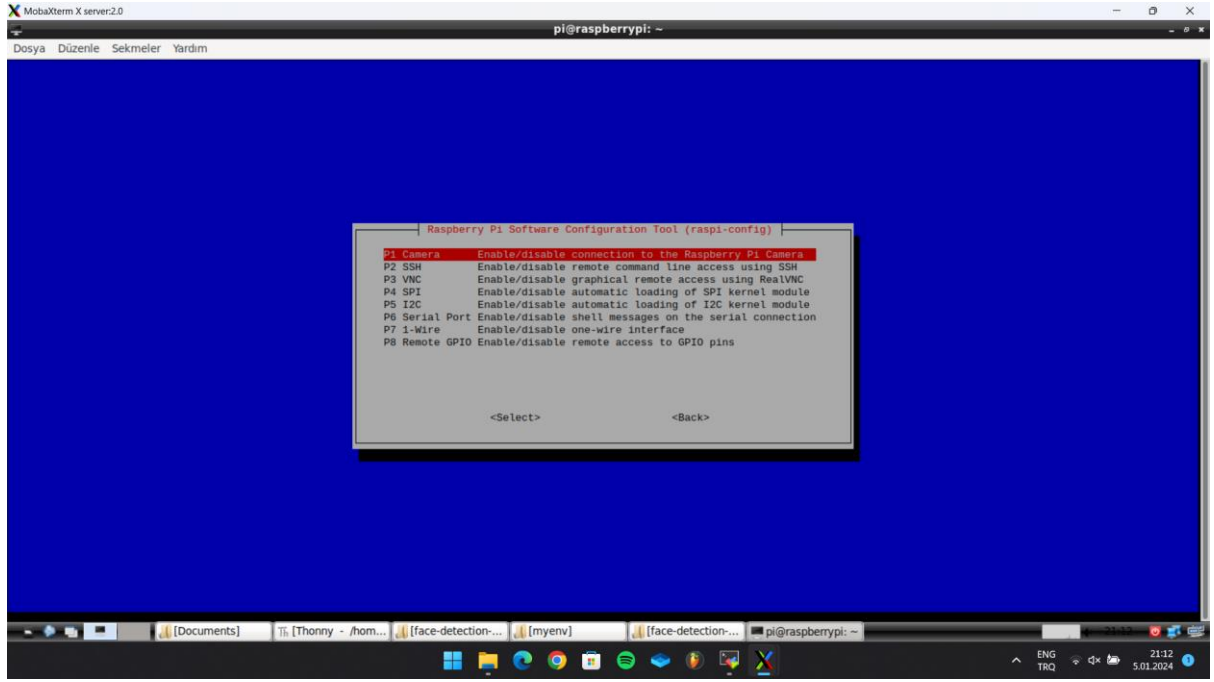
Şekil 3.2.3. Teminalde raspi-config komutunun girilmesi



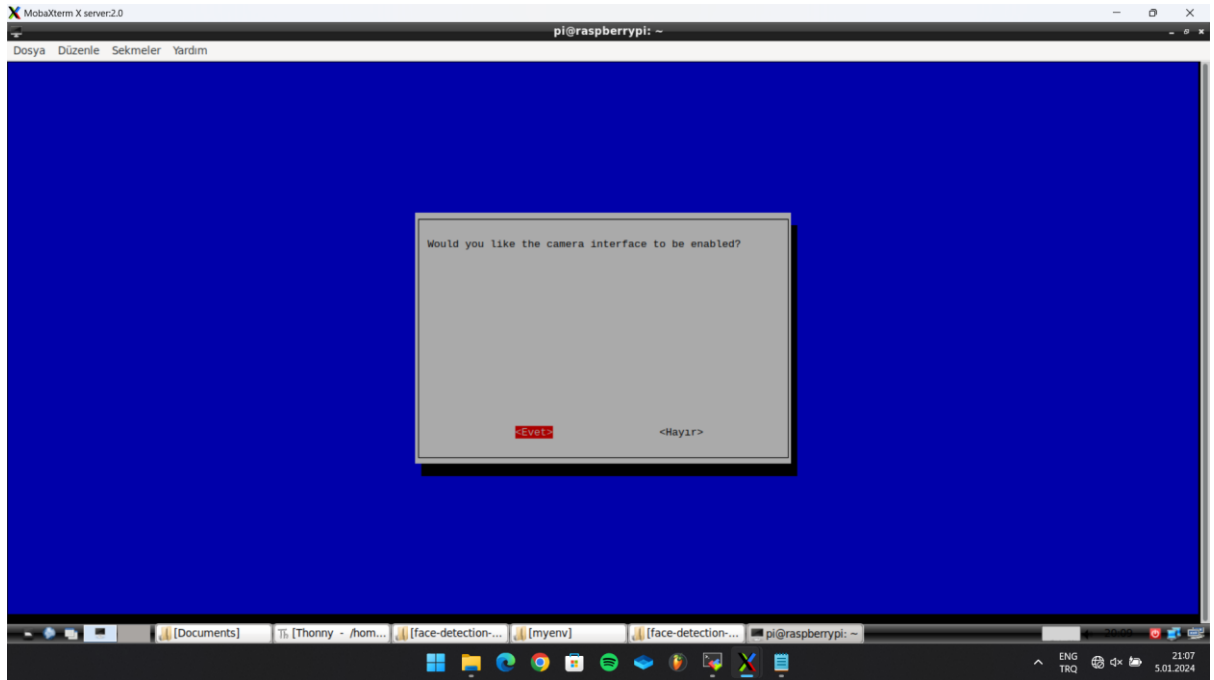
Şekil 3.2.4. Raspberry Ayar Menüsü



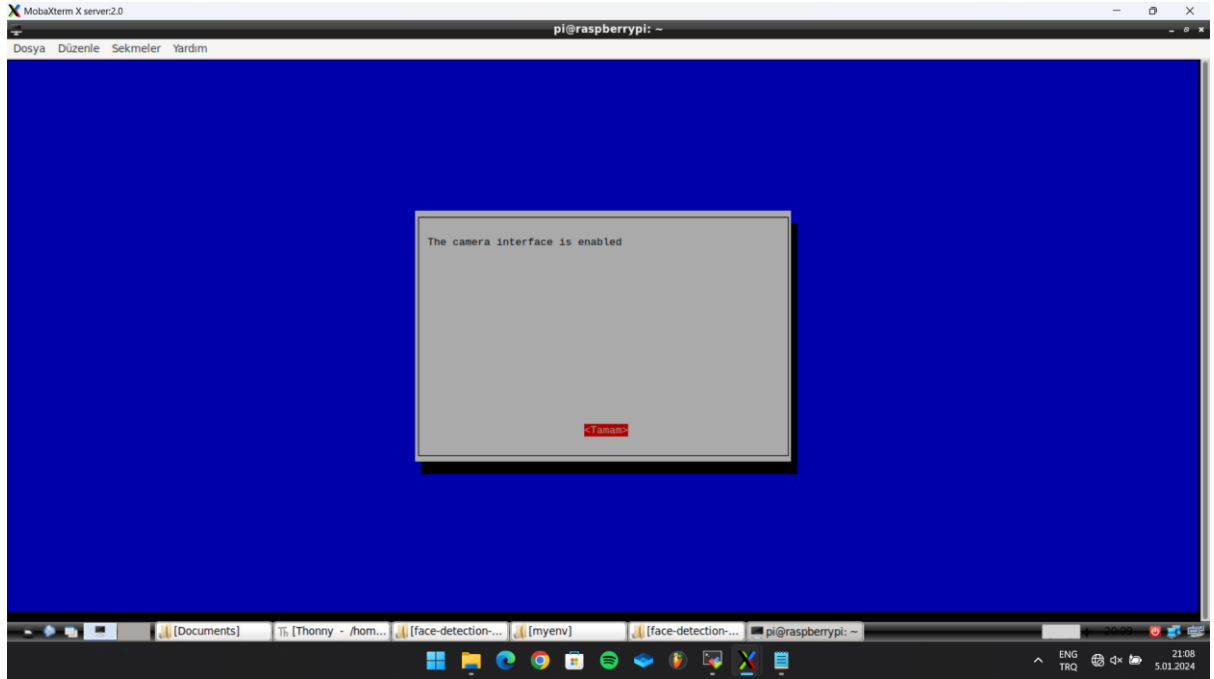
Şekil 3.2.5. Raspberry Ayar Menüsünde Interface Ayarlarına Gidilmesi



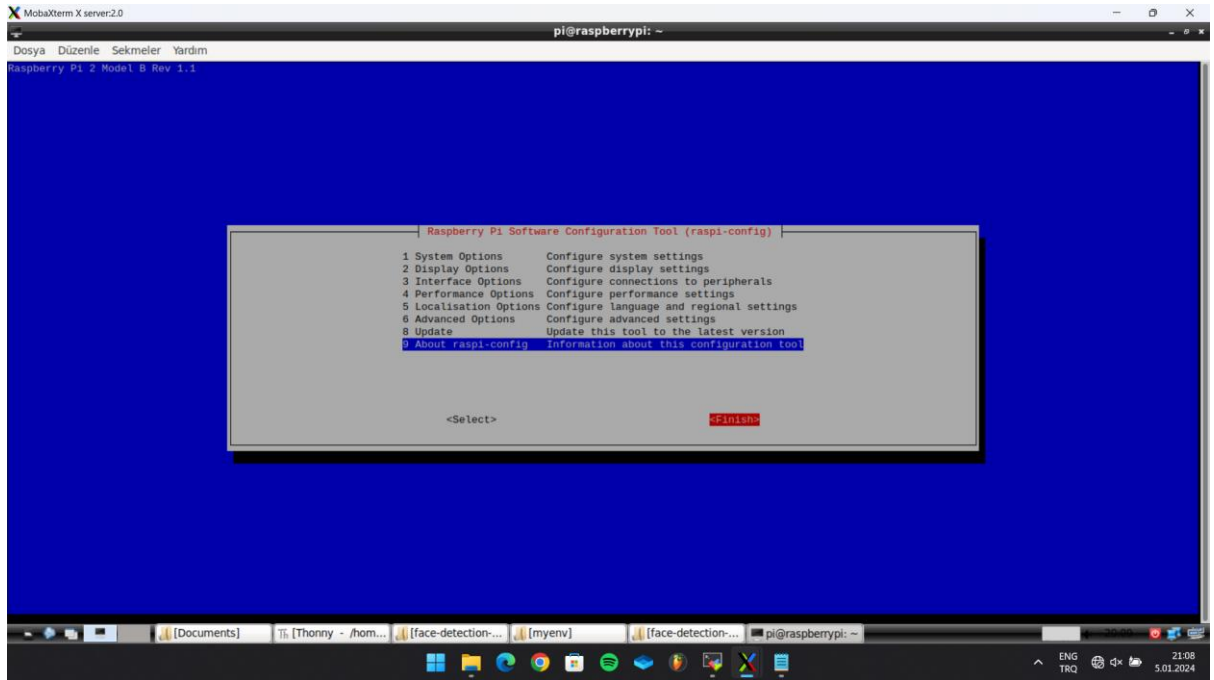
Şekil 3.2.6. Interface Ayarlarında Kamera Arayüzü Ayarlarının Yapılması



Şekil 3.2.7. Kamera Arayüzünün Aktif Edilmesi



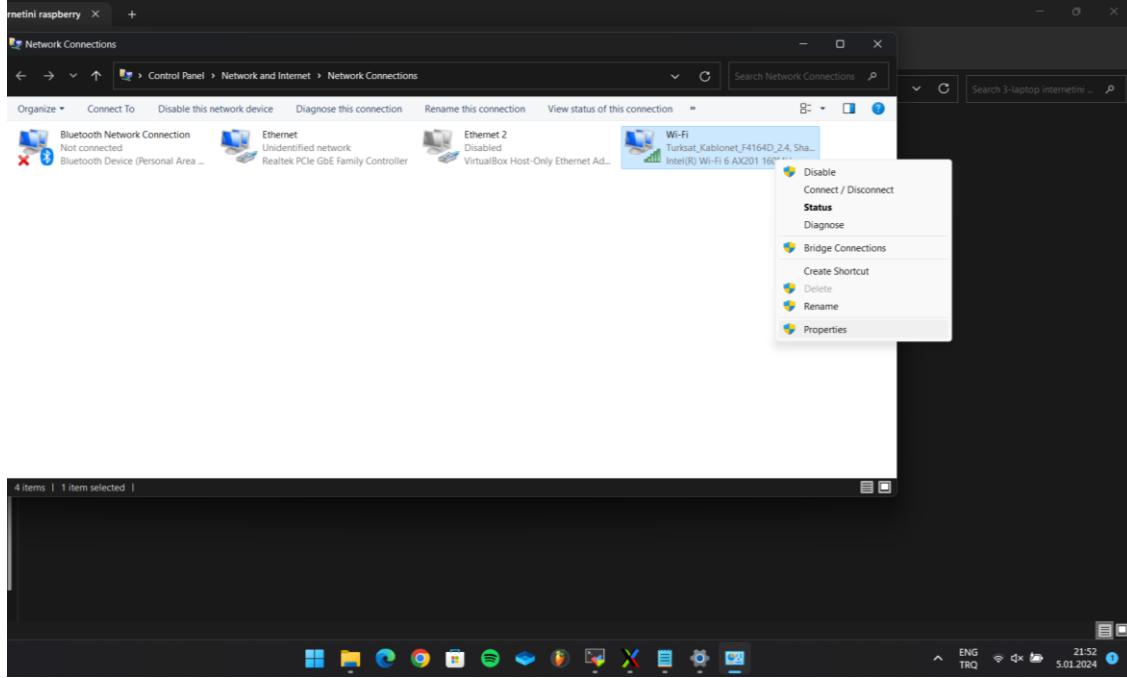
Şekil 3.2.8. Kamera Arayüzünün Aktif Edildiğini Belirten Mesaj



Şekil 3.2.9. Ayar Menüsünden Çıkılması

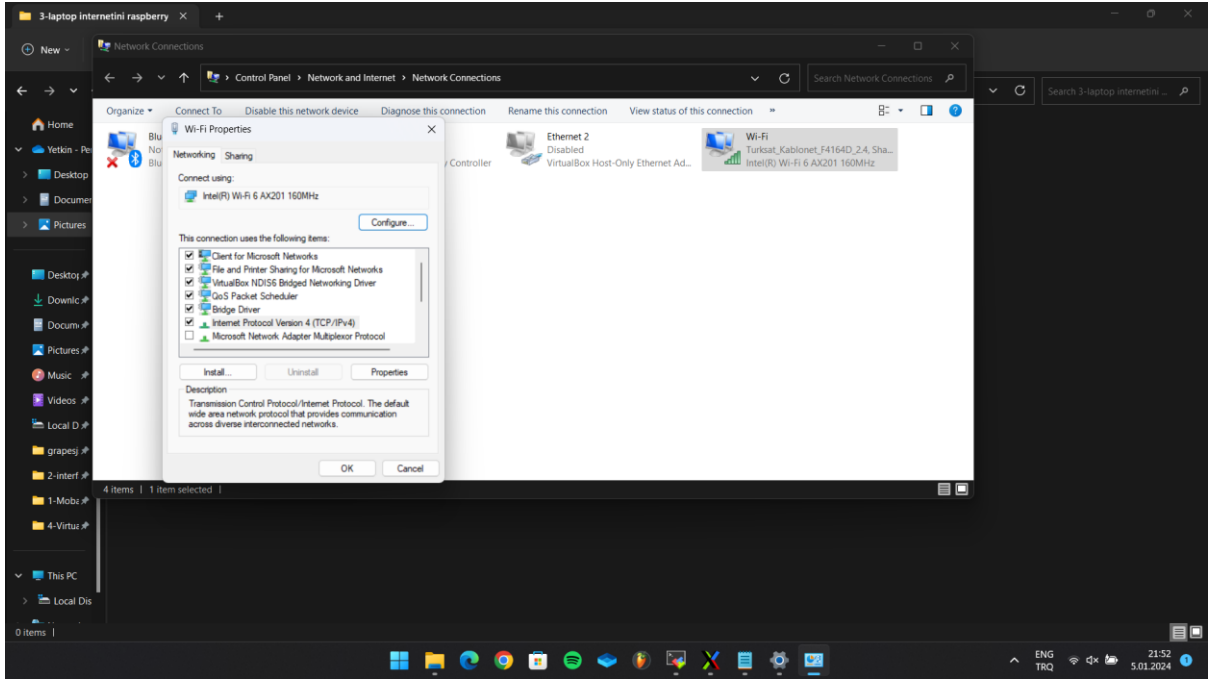
3.3. Bilgisayar Ağ Bağlantısının Raspberry Pi ile Paylaştırılması

Raspberry pi model 2 rev B, herhangi bir kablosuz ağ bağdaştırıcısına sahip olmadığından, bu Raspberry modelini internete bağlamak için aşağıdaki adımları takip edeceğiz.



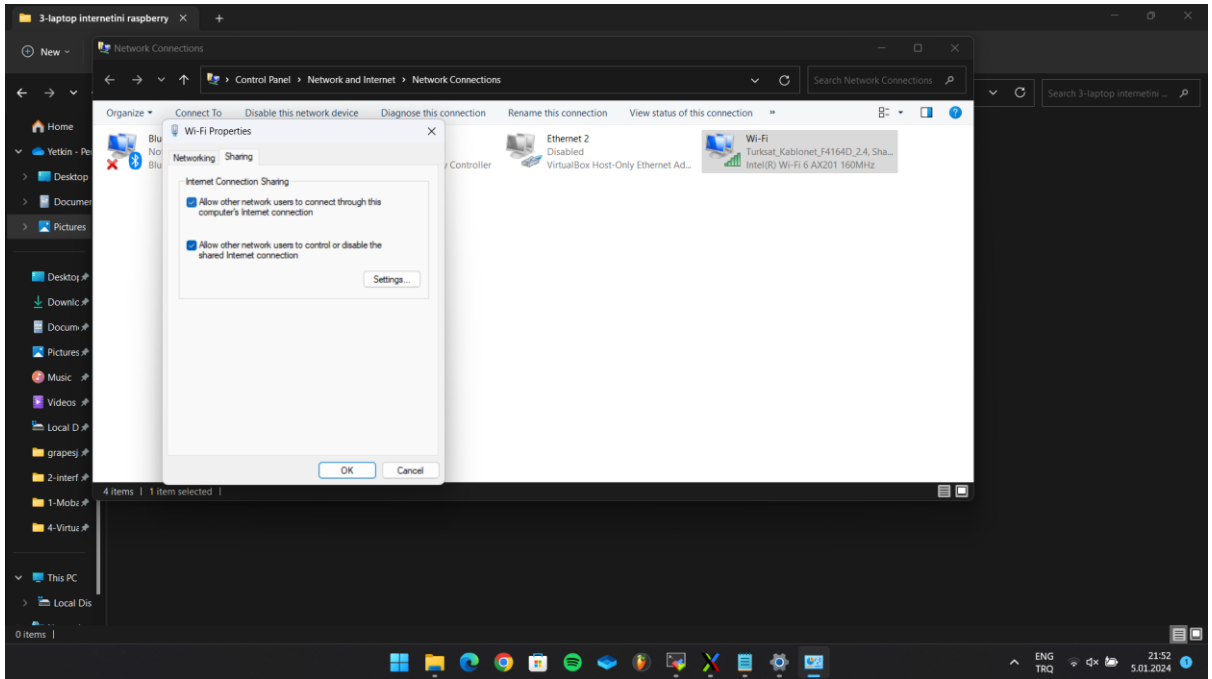
Şekil 3.3.1. Ağ ve İnternet Ayarları

3.3.1 numaralı resimdeki gibi; Denetim Masası>Ağ ve İnternet>Ağ bağlantı ayarlarına gidiyoruz.



Şekil 3.3.2. Kablosuz Bağlantı Özellikleri

3.3.2 numaralı resimdeki gibi, "wifi ağ özellikleri" menüsünü açıp,



Şekil 3.3.3. Kablosuz Bağlantı Paylaşım Ayarları

3.3.3 numaralı resimde görüldüğü gibi; "Sharing(Paylaşım)" sekmesine geliyoruz. Burada "Allow other users to connect through this computer's Internet connection" yazan check-box'a tick atıyoruz.

Böylece, Raspberry Pi internete bağlanmış oldu. Artık proje için gerekli programları ve python kütüphanelerini kurabiliriz.

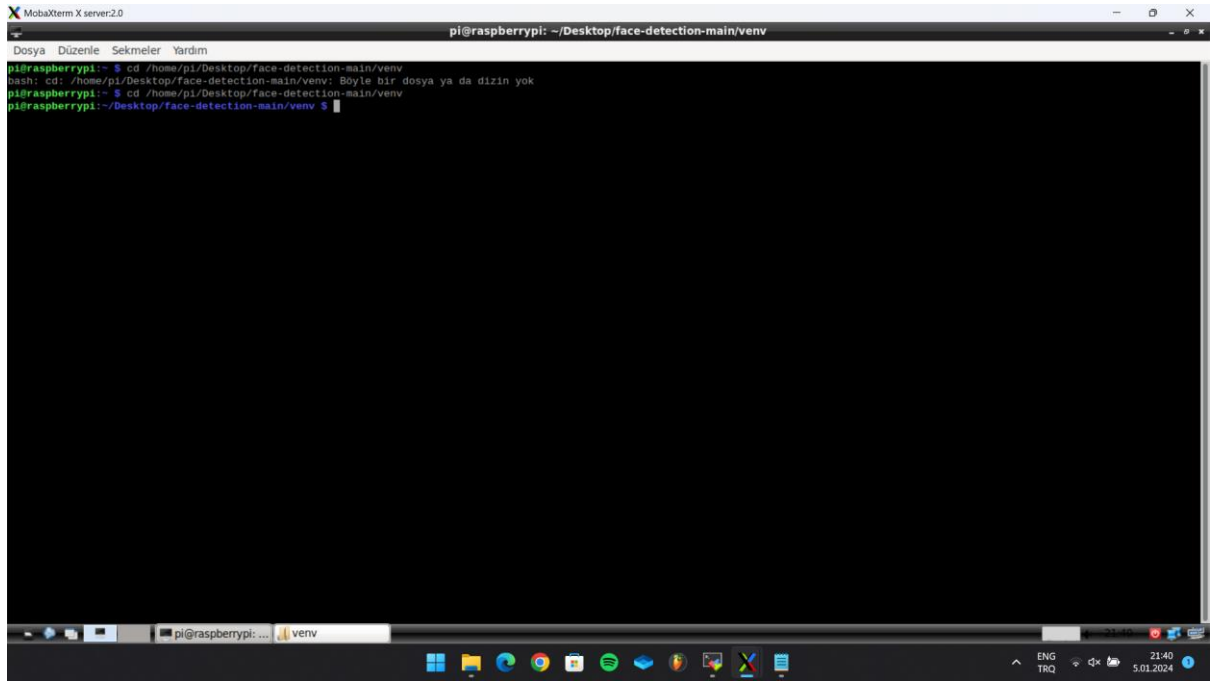
3.4. Virtual Environment ve Python Kütüphanelerinin Kurulumu

Virtual Environment (Sanal Ortam), Python programcılarının projelerini izole ederek bağımlılıkları yönetmelerini sağlayan bir araçtır. Python'un güçlü yanlarından biri, zengin bir kütüphane ekosistemine sahip olmasıdır. Ancak, farklı projelerin farklı paket sürümlerine ihtiyaç duyması veya farklı paketlere bağımlı olması durumunda, bu projeleri tek bir Python ortamında çalıştırmak zor olabilir. İşte bu noktada sanal ortamlar devreye girer. Sanal ortamlar, her projeyi izole ederek her birine kendi bağımlılıklarını ve paketlerini yüklemenizi sağlar. Bu sayede projelerinizin birbirine müdahale etme riskini azaltırken, her bir projenin özel ihtiyaçlarına göre paketleri yönetebilirsiniz.

Projede kullanılacak Python paketlerinin kurulumuna geçmeden önce, sanal ortam kurulumu gerçekleştirilecektir. Projede kullanılacak Python modülleri ise, bu sanal ortama kurulacaktır.

Öncelikle, terminal penceresi açıp aşağıdaki komutu girerek virtualenv kurulumunu gerçekleştiriyoruz.

Sudo pip3 install vitualenv



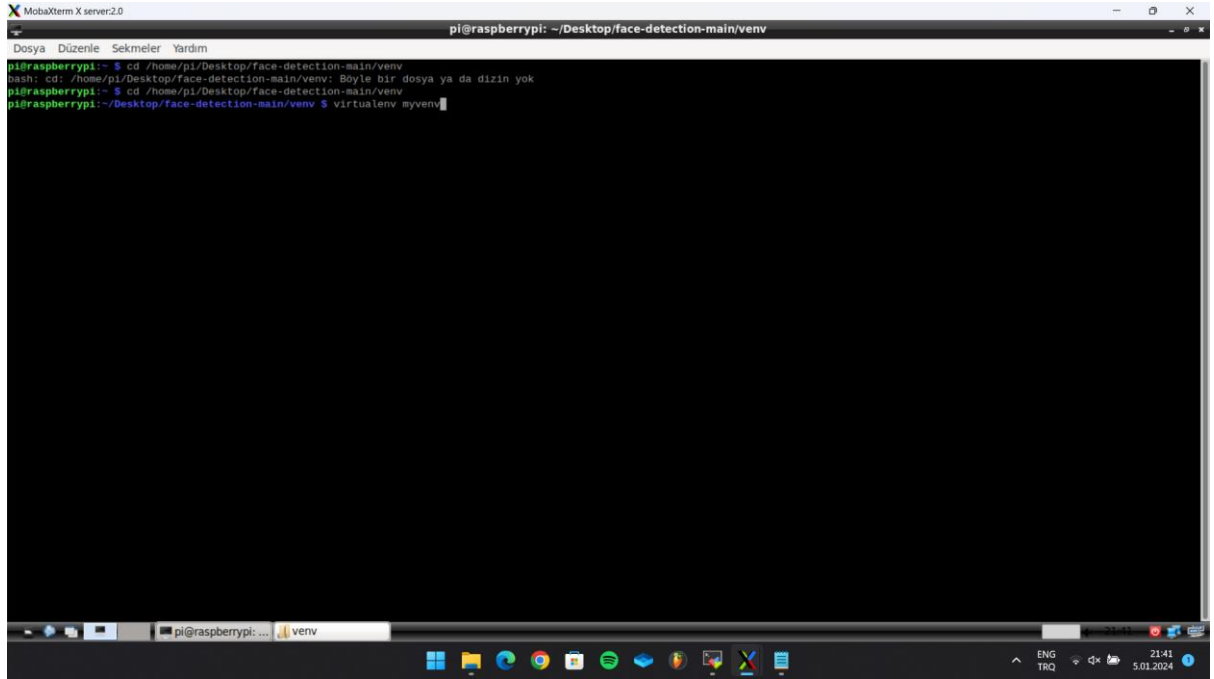
```
pi@raspberrypi: ~/Desktop/face-detection-main/venv
pi@raspberrypi:~$ cd /home/pi/Desktop/face-detection-main/venv
bash: cd: /home/pi/Desktop/face-detection-main/venv: Böyle bir dosya ya da dizin yok
pi@raspberrypi:~$ cd /home/pi/Desktop/face-detection-main/venv
pi@raspberrypi:~/Desktop/face-detection-main/venv $
```

Şekil 3.4.1. Sanal Ortamın Kurulacağı Dizin Açılması

Ardından, Şekil 3.4.1’de görüldüğü gibi

```
cd /home/pi/Desktop/face-detection-main/venv
```

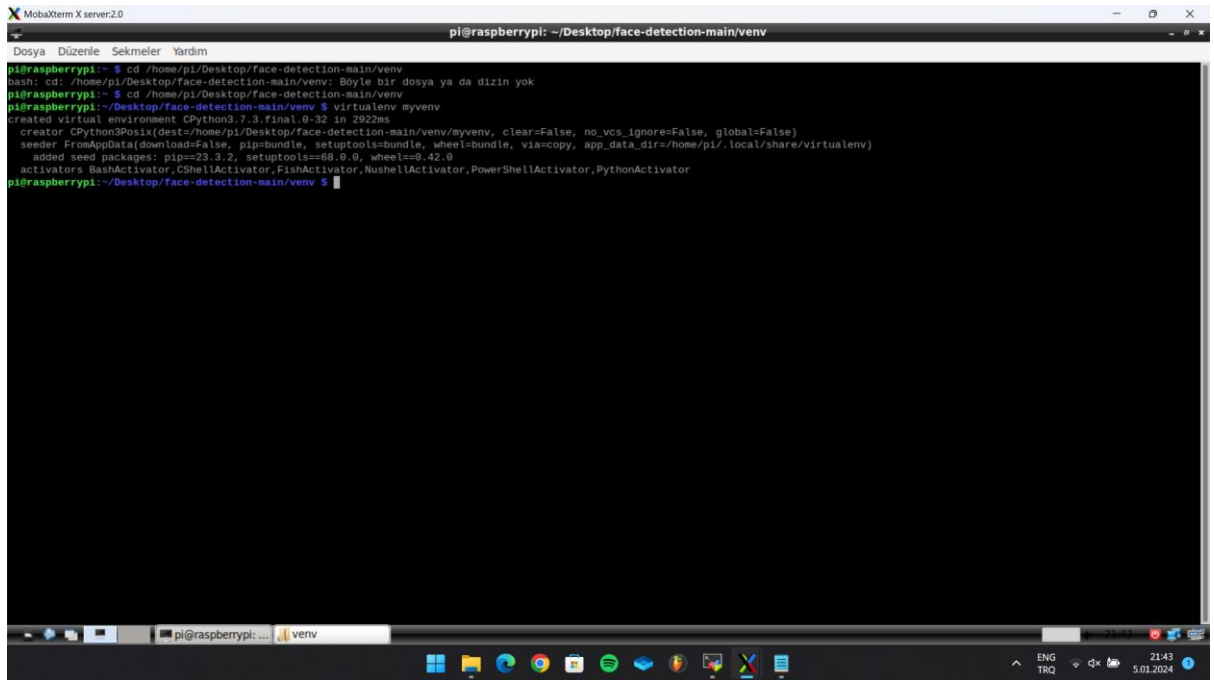
Komutunu girerek, projede kullanacağımız sanal ortam kurulumunun yapılacağı dizine gidiyoruz.



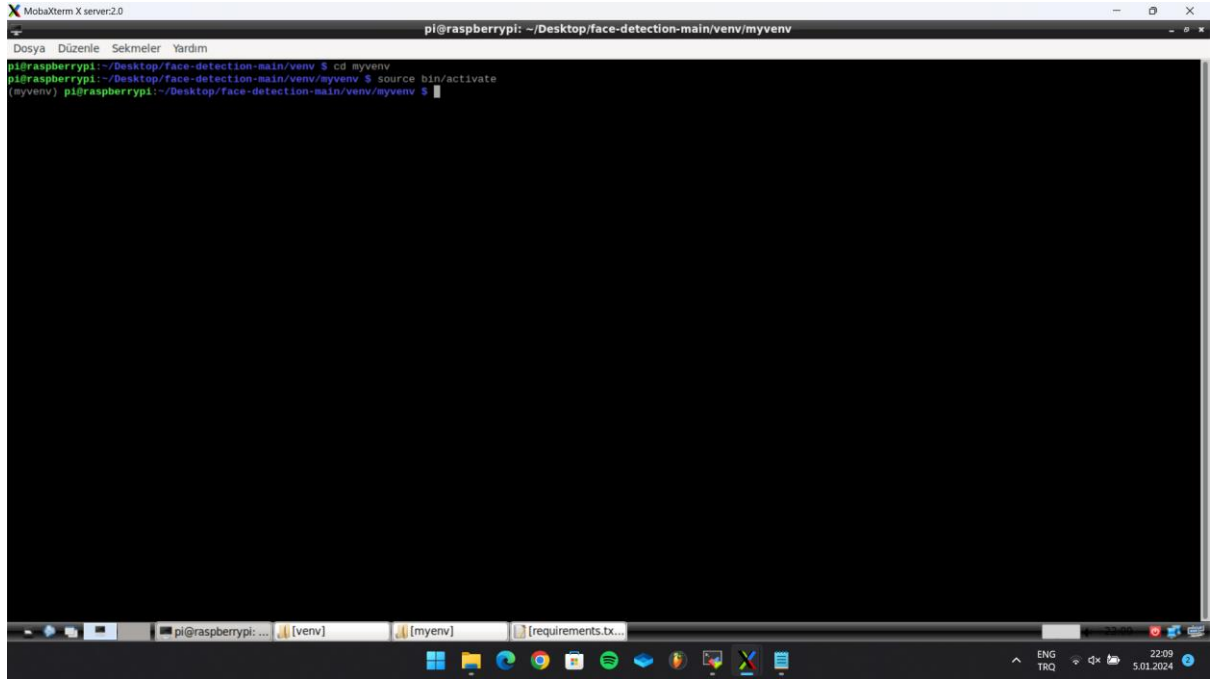
Şekil 3.4.2. Sanal ortamın oluşturulması

Şekil 3.4.2’de görüldüğü gibi, virtualenv myvenv komutu ile home/pi/Desktop/face-detection-main/venv dizini içinde myvenv isimli bir sanal ortam oluşturduk.

Bu komutun çıktısı, Şekil 3.4.3’te görülmektedir



Şekil 3.4.3. Komut Çıktısı



Şekil 3.4.4. Sanal Ortamın Aktif Edilmesi

Şekil 3.4.4'te görüldüğü gibi, cd myvenv komutu ile sanal ortamın kurulduğu dizine gidiyoruz.

Ardından, source bin/activate komutu ile sanal ortamı aktifleştiriyoruz.

Şimdi, projede kullanacağımız python modüllerinin kurulumunu gerçekleştireceğiz.

Projede kullanılan modüller:

- dlib:**

Dlib, C++ ile yazılmış bir kitaplık olan Dlib C++ Toolkit'in Python bağlamını içerir. Genellikle makine öğrenimi, görüntü işleme ve bilgisayar görüşü uygulamalarında kullanılır.

Kullanım Alanları: Yüz algılama, nesne tespiti, yüz tanıma, şekil analizi gibi alanlarda kullanılır.

- Pillow:**

Pillow, Python için bir resim işleme kütüphanesidir. Bu, görüntüleri açma, düzenleme ve kaydetme işlemlerini gerçekleştirmek için kullanılır.

Kullanım Alanları: Resim işleme, resim dosyalarını açma, kaydetme, döndürme, boyutlandırma, filtreleme gibi temel resim manipülasyon işlemlerini içerir.

- Numpy:**

NumPy, çok boyutlu diziler ve matrisler üzerinde çalışmak için güçlü bir kütüphanedir. Bilimsel hesaplamalar ve veri manipülasyonu için yaygın olarak kullanılır.

Kullanım Alanları: Matematiksel işlemler, lineer cebir, rastgele sayı üretimi, istatistiksel analiz gibi veri bilimi ve mühendislik uygulamalarında kullanılır.

- face_recognition:**

Anıma algoritmalarını basitleştirmek ve kullanıcılar için erişilebilir hale getirmek için tasarlanmış bir kütüphanedir.

Kullanım Alanları: Yüz tanıma uygulamalarında, kişi tanıma, yüz özelliklerini çıkarma gibi işlemlerde kullanılır.

•opencv-contrib-python:

OpenCV (Open Source Computer Vision) bilgisayar görüşü ve makine görüşü uygulamaları için geniş bir kütüphanedir. opencv-contrib-python ise OpenCV'nin ek modüllerini içerir.

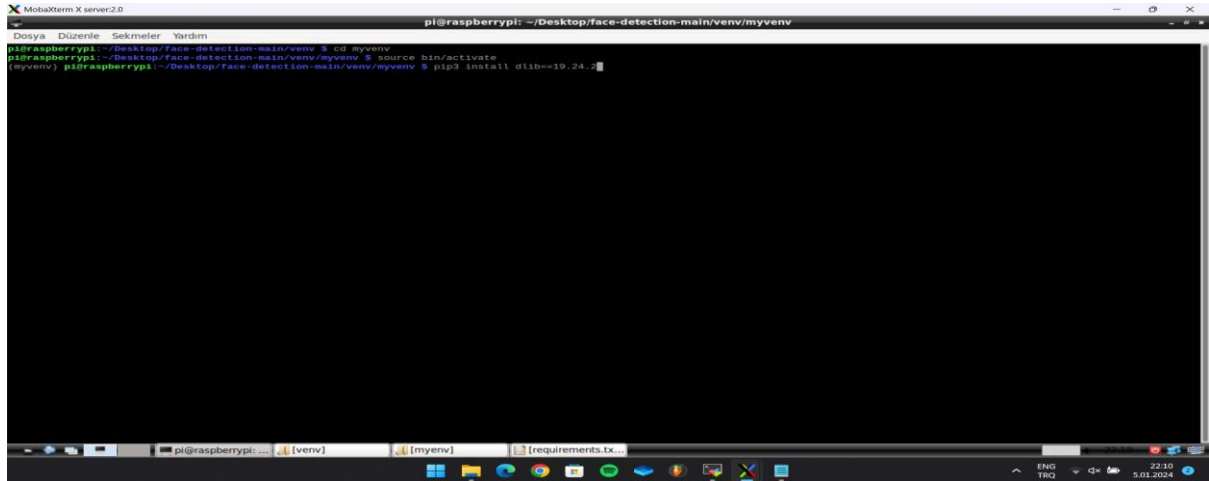
Kullanım Alanları: Görüntü işleme, nesne tespiti, yüz tanıma, stereo görüş, video analizi gibi birçok bilgisayar görüşü görevinde kullanılır.

•Rpi.gpio:

RPI.GPIO, Raspberry Pi üzerinde GPIO (Genel Amaçlı Giriş/Çıkış) pinlerini kontrol etmek için kullanılan bir Python kütüphanesidir.

Kullanım Alanları: Raspberry Pi üzerinde sensörler, motorlar, LED'ler gibi çeşitli elektronik bileşenleri kontrol etmek ve projeler geliştirmek için kullanılır.

pip3 install dlib==19.24.2 komutuyla dlib modülünü kuruyoruz:



```
pi@raspberrypi: ~/Desktop/face-detection-main/venv/myvenv
pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv$ cd myvenv
pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv$ source bin/activate
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv$ pip3 install dlib==19.24.2
```

Şekil 3.4.5. dlib modülü kurulum komutu

Komutun çıktısı aşağıdaki resimde verilmiştir:

```
MobaXterm X server 2.0
pi@raspberrypi: ~/Desktop/face-detection-main/venv/myvenv

Dosya Düzenle Sekmeler Yardım

pi@raspberrypi:~/Desktop/face-detection-main/venv $ cd myvenv
pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv $ source bin/activate
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv $ pip3 install dlib==19.24.2
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting dlib==19.24.2
  Using cached https://www.piwheels.org/simple/dlib/dlib-19.24.2-cp37m-linux_armv7l.whl (2.9 MB)
Installing collected packages: dlib
Successfully installed dlib-19.24.2
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv $
```

Şekil 3.4.6. dlib modülünün kurulumu sonrası alınan konsol çıktısı

pip3 install Pillow==9.5.0 komutu ile pillow modülünü kuruyoruz:

```
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv $ pip3 install Pillow==9.5.0
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting Pillow==9.5.0
  Using cached https://www.piwheels.org/simple/pillow/Pillow-9.5.0-cp37m-linux_armv7l.whl (820 kB)
Installing collected packages: Pillow
Successfully installed Pillow-9.5.0
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv $
```

Şekil 3.4.7. pillow modülünün kurulumu

Pip3 install numpy==1.20.0 komutu ile numpy modülünü kuruyoruz:

```
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv $ pip3 install numpy==1.20.0
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting numpy==1.20.0
  Using cached https://www.piwheels.org/simple/numpy/numpy-1.20.0-cp37m-linux_armv7l.whl (11.5 MB)
Installing collected packages: numpy
Successfully installed numpy-1.20.0
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv $
```

Şekil 3.4.8. numpy modülünün kurulumu

Pip3 install face_recognition --no-cache-dir komutu ile face-recognition modülünü kuruyoruz:

```
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv $ pip3 install face_recognition --no-cache-dir
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting face_recognition
  Downloading https://www.piwheels.org/simple/face_recognition/face_recognition-1.3.0-py2.py3-none-any.whl (15 kB)
Collecting face_recognition_models==0.3.0 (from face_recognition)
  Downloading https://www.piwheels.org/simple/face_recognition_models/face_recognition_models-0.3.0-py2.py3-none-any.whl (100.6 MB)
    100.6/100.6 MB 1.5 MB/s eta 0:00:00
Collecting Click==6.0 (from face_recognition)
  Downloading https://www.piwheels.org/simple/click/click-8.1.7-py3-none-any.whl (97 kB)
Requirement already satisfied: dlib==19.7 in ./lib/python3.7/site-packages (from face_recognition) (19.24.2)
Requirement already satisfied: numpy in ./lib/python3.7/site-packages (from face_recognition) (1.20.0)
Requirement already satisfied: Pillow in ./lib/python3.7/site-packages (from face_recognition) (9.5.0)
Collecting importlib-metadata (from Click==6.0->face_recognition)
  Downloading https://www.piwheels.org/simple/importlib-metadata/importlib_metadata-6.7.0-py3-none-any.whl (22 kB)
Collecting zipp==0.5 (from importlib-metadata->Click==6.0->face_recognition)
  Downloading https://www.piwheels.org/simple/zipp/zipp-3.15.0-py3-none-any.whl (6.8 kB)
Collecting typing-extensions==3.6.4 (from importlib-metadata->Click==6.0->face_recognition)
  Downloading https://www.piwheels.org/simple/typing-extensions/typing_extensions-4.7.1-py3-none-any.whl (33 kB)
Installing collected packages: face_recognition_models, zipp, typing-extensions, importlib-metadata, Click, face_recognition
Successfully installed Click-8.1.7 face_recognition_models-0.3.0 face_recognition-1.3.0 importlib-metadata-6.7.0 typing-extensions-4.7.1 zipp-3.15.0
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv/myvenv $
```

Şekil 3.4.9. face-recognition modülünün kurulumu

Open cv kurulumunu, kütüphanenin ağır olması ve derlenmesinin fazla zaman alması sebebiyle, Piwheels.org sitesinden indirdiğimiz whl uzantılı paket dosyasını (yani pre-compiled binary Python package) kullanarak yapacağız.

Version	Released	Buster Python 3.7	Bullseye Python 3.9	Bookworm Python 3.11	Files
4.9.0.80	2023-12-31	
4.8.1.78	2023-09-28	✗	✗	✗	
4.8.0.76	2023-08-09	✗	✗	✗	
4.8.0.74	2023-06-30	✗	✗	✗	
4.7.0.72	2023-02-22	✗	✗	✗	
4.7.0.68	2022-12-30	✗	✗	✗	
4.6.0.66	2022-06-08	✗	✗	✗	
4.5.5.64	2022-03-09	✗	✗	✗	
4.5.5.62	2021-12-29	✓	✓	✗	How to install this version
opencv_contrib_python-4.5.5.62-cp37-cp37m-linux_armv6l.whl (16 MB)					
opencv_contrib_python-4.5.5.62-cp37-cp37m-linux_armv7l.whl (16 MB)					
opencv_contrib_python-4.5.5.62-cp39-cp39-linux_armv6l.whl (17 MB)					
opencv_contrib_python-4.5.5.62-cp39-cp39-linux_armv7l.whl (17 MB)					
4.5.4.60	2021-11-22	✓	✓	✗	+
4.5.4.58	2021-10-21	✗	✗	✗	
4.5.3.56	2021-07-11	✓	✓	✗	+
4.5.2.54	2021-06-07	✗	✗	✗	
4.5.2.52	2021-05-07	✓	✓	✗	+
4.5.1.48	2021-01-02	✓	✓	✗	+

Şekil 3.4.10. piwheels.org sitesi

Şekil 3.4.10'da görüldüğü üzere;

Sitedeki paketlerden, raspberry pi 2 üzerindeki işlemcinin mimarisi ile uyumlu olan opencv-contrib-python-4.5.5.62-cp37-cp37m-linux_armv7l.whl paketini indiriyoruz ve raspberry pi bilgisayarın masaüstünde proje klasörümüz olan face-detection-main klasörüne kopyalıyoruz.

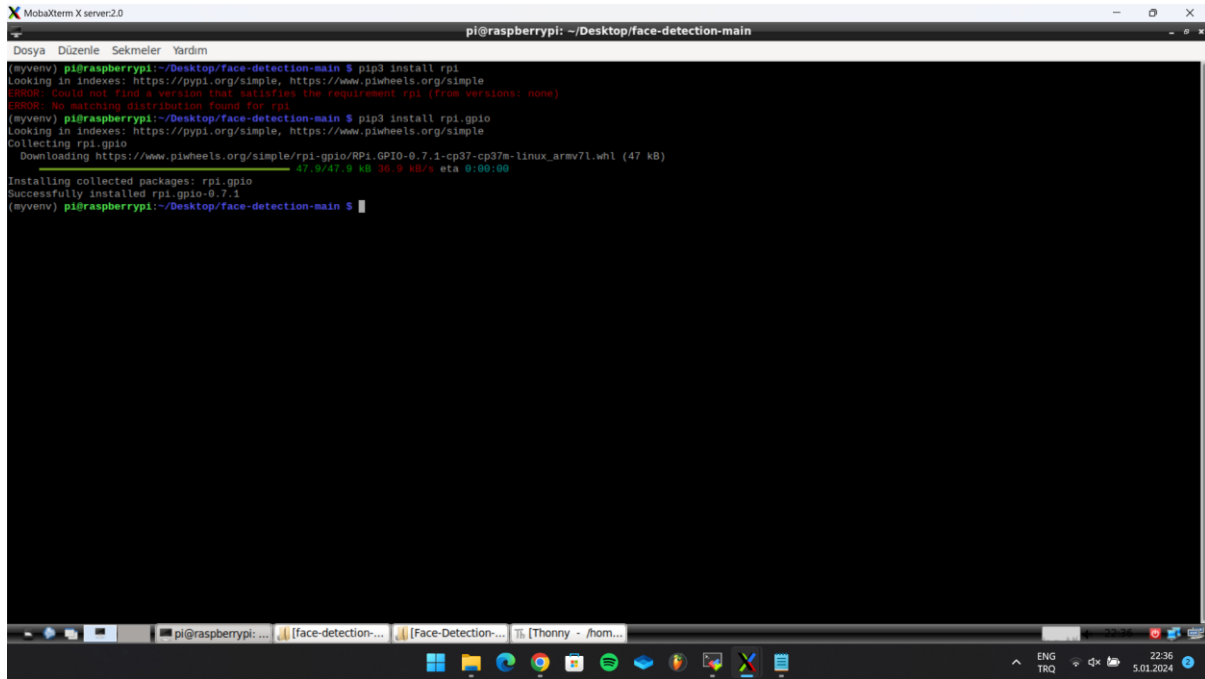
Terminal ekranında, Şekil 3.4.11'de görüldüğü üzere face-detection-main klasörüne giderek, aşağıdaki komutu girerek, open cv modülünün kurulumunu gerçekleştiriyoruz:

Pip3 install opencv-contrib-python-4.5.5.62-cp37-cp37m-linux_armv7l.whl

```
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main/venv $ cd ..
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main $ pip3 install opencv_contrib_python-4.5.5.62-cp37-cp37m-linux_armv7l.whl
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Processing ./opencv_contrib_python-4.5.5.62-cp37-cp37m-linux_armv7l.whl
Requirement already satisfied: numpy>=1.14.5 in ./venv/myvenv/lib/python3.7/site-packages (from opencv-contrib-python==4.5.5.62) (1.20.0)
Installing collected packages: opencv-contrib-python
Successfully installed opencv-contrib-python-4.5.5.62
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main $
```

Şekil 3.4.11. opencv modülünün kurulumu

GPIO pinlerini kullanabilmek için yüklememiz gereken rpi.gpio modülünün kurulumunu, şekil 3.4.12'de görüldüğü gibi gerçekleştiriyoruz:



```
MobaXterm X server2.0
pi@raspberrypi: ~/Desktop/face-detection-main

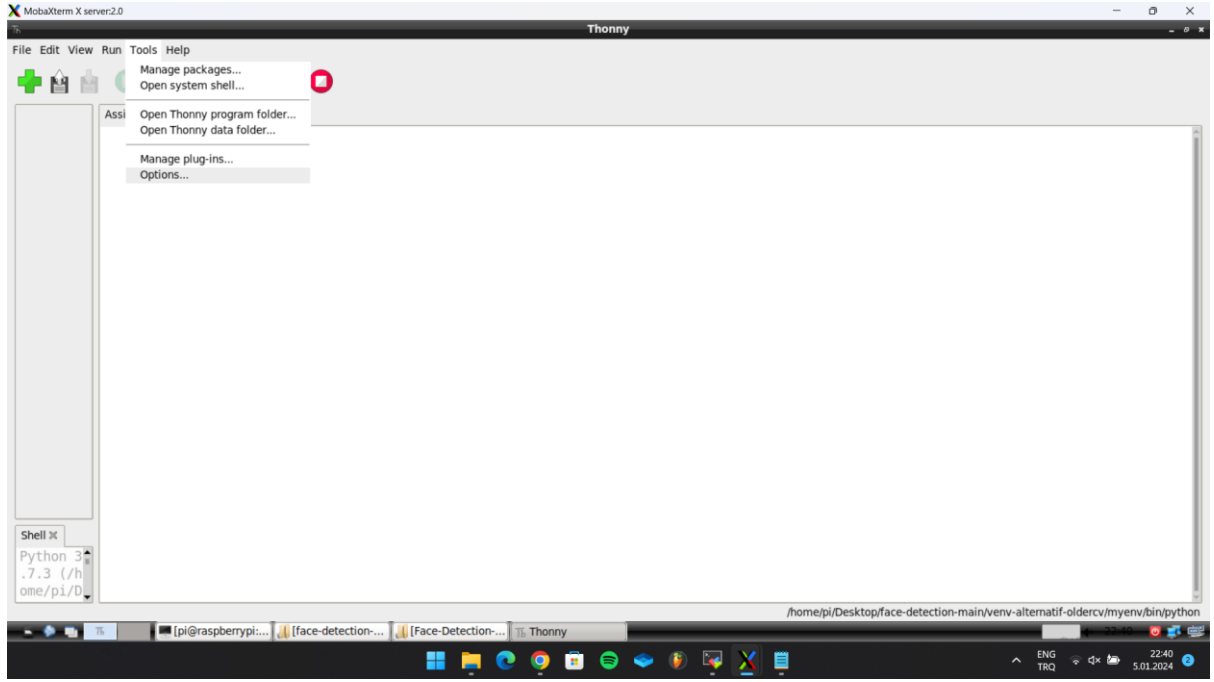
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main $ pip3 install rpi
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
ERROR: Could not find a version that satisfies the requirement rpi (from versions: none)
ERROR: No matching distribution found for rpi

(myvenv) pi@raspberrypi:~/Desktop/face-detection-main $ pip3 install rpi.gpio
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting rpi.gpio
  Downloading https://www.piwheels.org/simple/rpi-gpio/RP1.GPIO-0.7.1-cp37-cp37m-linux_armv7l.whl (47 kB)
    47.9/47.9 kB 30.9 kB/s eta 0:00:00
Installing collected packages: rpi.gpio
Successfully installed rpi.gpio-0.7.1
(myvenv) pi@raspberrypi:~/Desktop/face-detection-main $
```

Şekil 3.4.12. Rpi.GPIO modülünün kurulumu

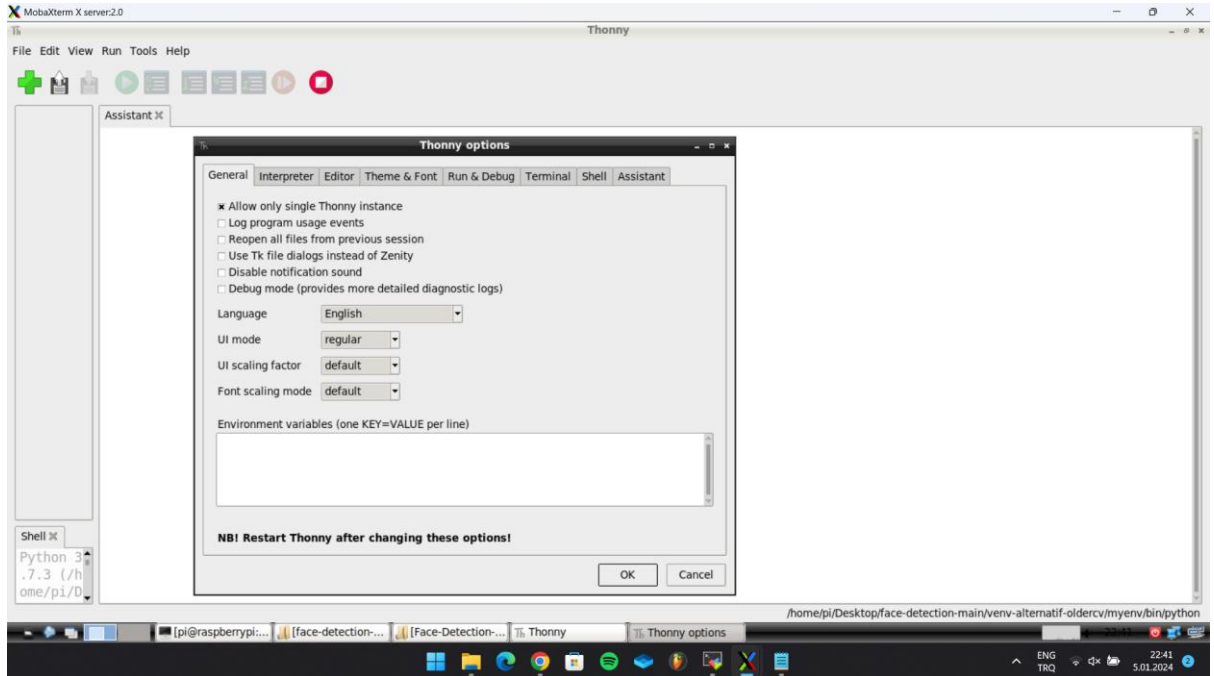
Modüllerimizin kurulumunu tamamlamış bulunuyoruz.

Kodlamayı Thonny IDE' de yapacağız. Kurulumunu yaptığımız virtual environment 'i kullanmak için, Thonny IDE 'de tools>options menüsünü açıyoruz. (Şekil 3.4.13)

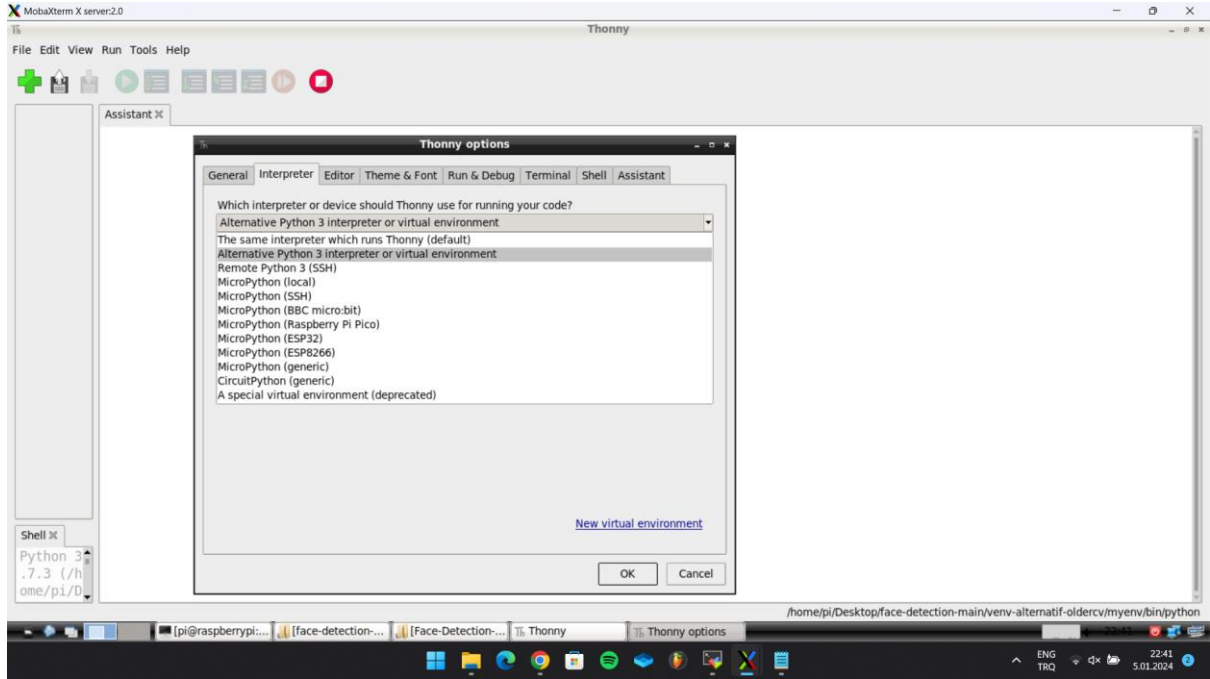


Şekil 3.4.13. Thonny IDE

Şekil 3.4.14'te görülen menüde "Interpreter" sekmesine tıklıyoruz.



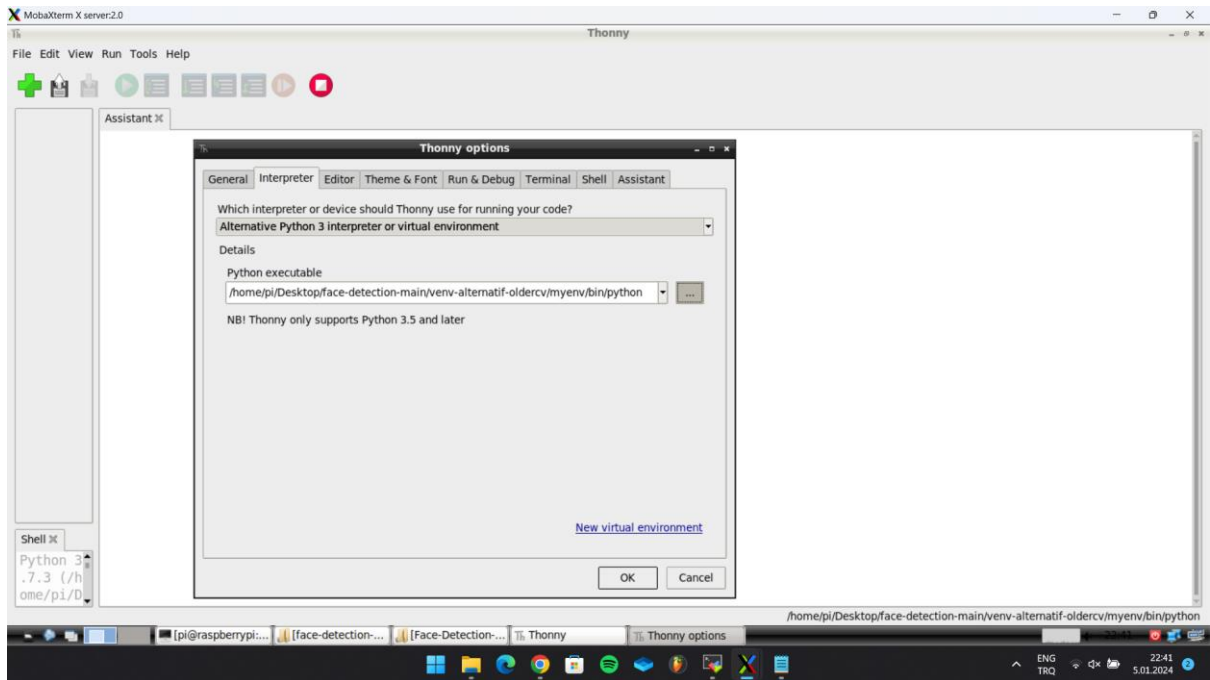
Şekil 3.4.14. Thonny IDE Ayar Menüsü



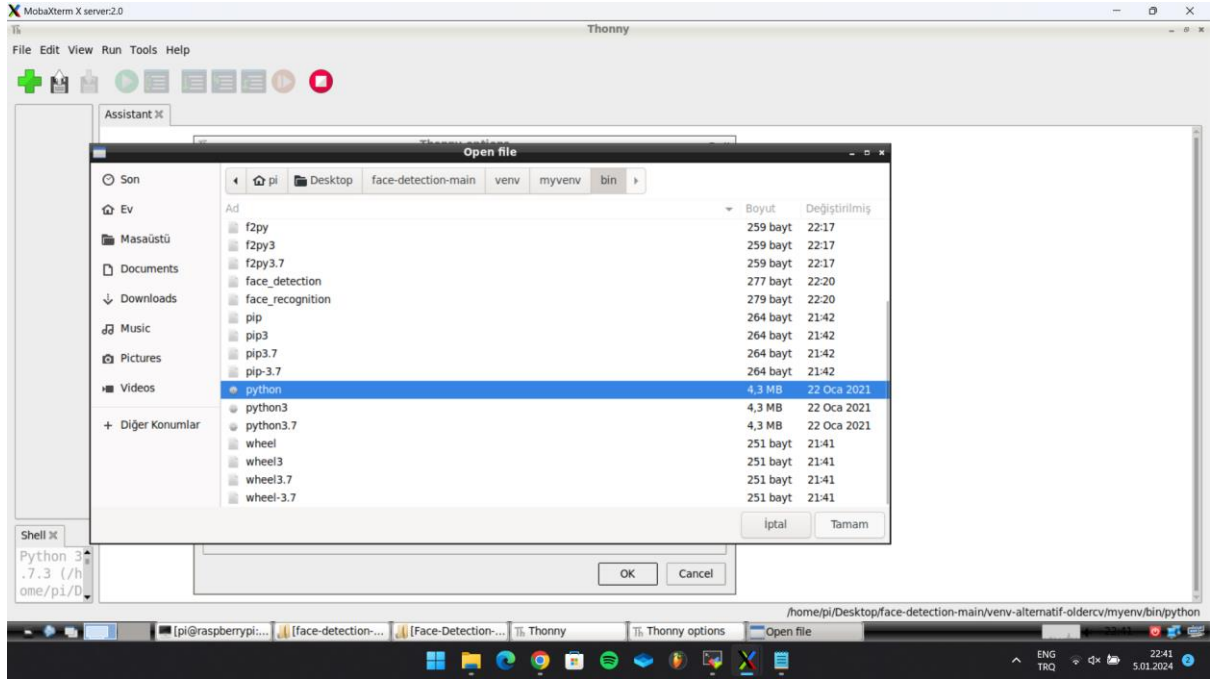
Şekil 3.4.15. Interpreter sekmesi

Şekil 3.4.15'teki gibi, interpreter seçimini "Alternative Python3 Interpreter or virtual environment" olarak değiştiriyoruz.

virtual environment Python executable dosya yolunu tanımlamak üzere, ... butonuna tıklıyoruz (Şekil 3.4.16)

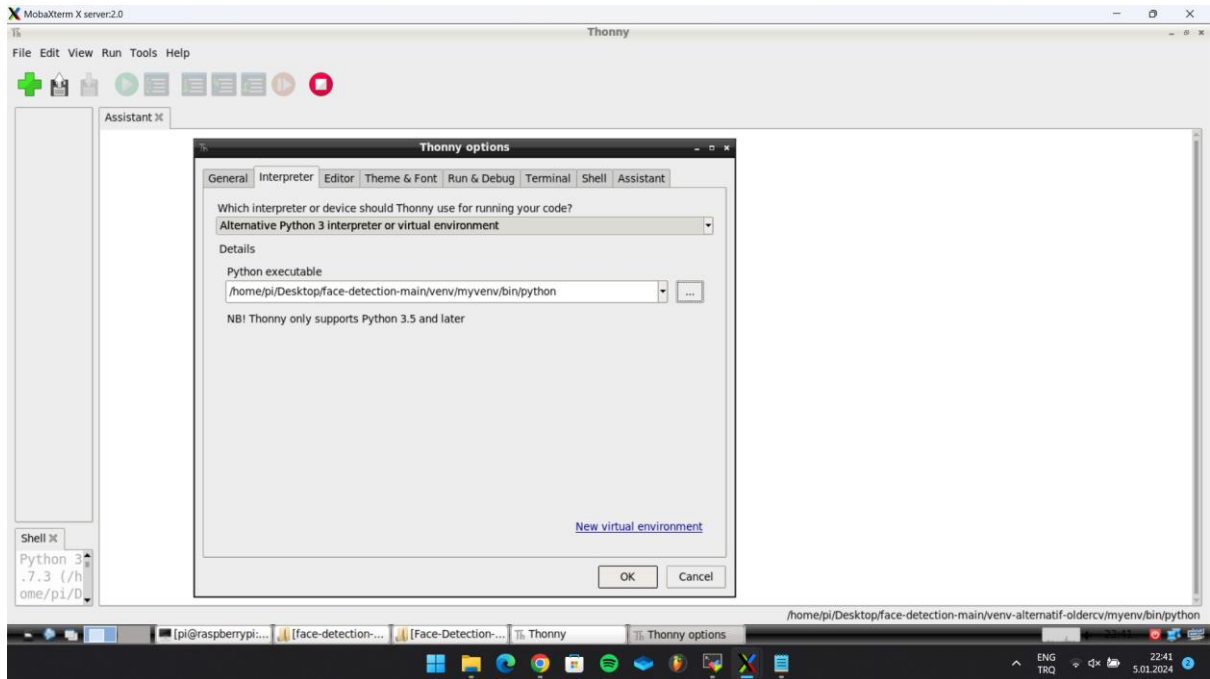


Şekil 3.4.16. Sanal ortam dosya yolu ayarları



Şekil 3.4.17. Sanal ortam dosya yolu ayarlarının yapılması

Dosya yolunu tanımlıyoruz (Şekil 3.4.17) ve "Tamam" butonuna tıklayıp kapatıyoruz.



Şekil 3.4.18. Interpreter ayarlarının son hali

Ayarlarımızın son hali şekil 3.4.18'deki gibidir.

Artık kodlama aşamasına geçebiliriz.

3.5. Yazılım Aşaması

Face_Recog.py Kodlarının Açıklamaları

Import İşlemleri:

```
import cv2
import numpy as np
import os
from PIL import Image
```

Bu satırlar, kodun çalışması için gerekli olan kütüphaneleri içe aktarır.

- **`cv2`**: OpenCV (Open Source Computer Vision) kütüphanesidir ve görüntü işleme işlemleri için kullanılır.
- **`numpy`**: Sayısal hesaplamalar ve matris işlemleri için kullanılır.
- **`os`**: İşletim sistemi ile ilgili işlemleri gerçekleştirmek için kullanılır.
- **`PIL.Image`**: Python Imaging Library'nin (Pillow olarak da bilinir) bir parçasıdır ve resim dosyalarını işlemek için kullanılır.

Etiketler (Labels):

```
labels = ["zeynep", "yetkin"]
```

Bu satır, tanınacak yüzlerin etiketlerini içeren bir liste oluşturur. Her etiket, eğitim sırasında kullanılan etiketle aynı olmalıdır.

Cascade Sınıflandırıcı ve RÖluşturma:

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("face-trainner.yml")
```

- **`face_cascade`**: Yüz tespiti için Haar sınıflandırıcısını yükler.
- **`recognizer`**: LBPH yüz tanıma algoritmasıyla bir yüz tanıyıcı oluşturur ve eğitilmiş modeli ("face-trainner.yml") okur.

Kamera Bağlantısı:

```
cap = cv2.VideoCapture(0)
```

Bu satır, bilgisayarınıza bağlı bir kameradan video akışı almak için bir `VideoCapture` nesnesi oluşturur. `0` genellikle yerleşik kamerayı ifade eder.

Ana Döngü:

```
while(True):
```

Bu satır, sonsuz bir döngü başlatır. Video akışından sürekli olarak kareler alacak ve yüz tanıma işlemlerini gerçekleştirecektir.

Video Çerçevesi Alımı:

```
ret, img = cap.read()
```

`cap.read()` metodu, bir sonraki video çerçevesini alır ve `ret` değişkeni başarılı bir şekilde çerçeve alınıp alınmadığını belirtir.

Gri Tonlamaya Dönüştürme:

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Alınan renkli görüntüyü gri tonlamaya dönüştürür. Gri tonlamalı görüntü üzerinde yüz tanıma daha etkilidir.

Yüzleri Algılama ve Tanıma:

```
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.5,  
for (x, y, w, h) in faces:  
    roi_gray = gray[y:y+h, x:x+w]  
    id_, conf = recognizer.predict(roi_gray)  
    ...
```

Algılanan her yüz için gri tonlamalı yüz bölgesini tanıyıcıya (`recognizer`) gönderir ve tahmin sonuçlarını alır.

Yüz Tanıma Sonuçlarını Kontrol Etme ve İsimleri Ekleme:

```
if conf >= 80:
    font = cv2.FONT_HERSHEY_SIMPLEX
    name = labels[id_]
    cv2.putText(img, name, (x, y), font, 1, (0, 0, 255), 2)
cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

Eğer tahmin güvenilirlik (conf) belirli bir eşiği aşıyorsa, yüzün üzerine etiketi ve dikdörtgeni çizer.

Görüntüyü Gösterme ve Çıkış Kontrolü:

```
cv2.imshow('Preview', img)
if cv2.waitKey(20) & 0xFF == ord('q'):
    break
```

`cv2.imshow` görüntüyü ekranda gösterir. `cv2.waitKey` ise belirli bir tuşa basılana veya belirli bir süre geçene kadar bekler. Bu durumda, "q" tuşuna basılınca veya pencere kapatılınca döngüden çıkılır.

Kaynakları Serbest Bırakma:

```
cap.release()
cv2.destroyAllWindows()
```

Döngüden çıktıktan sonra, kamera kaynağını serbest bırakır ve açık pencereleri kapatır.

Bu kod, kameradan alınan video akışındaki yüzleri tanıyarak ekrana isimlerini ekleyen bir yüz tanıma uygulamasını gerçekleştirir.

Face_Trainer.py Kodlarının Açıklaması

Import İşlemleri:

```
import cv2
import numpy as np
import os
from PIL import Image
```

Bu satırlar, kodun çalışması için gerekli olan kütüphaneleri içe aktarır.

Haar Cascade Yüz Sınıflandırıcısının Yüklenmesi:

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Bu satır, yüzleri tespit etmek için Haar Cascade sınıflandırıcısını yükler.

LBPH (Local Binary Pattern Histograms) Yüz Tanıyıcı Oluşturma:

```
recognizer = cv2.face.LBPHFaceRecognizer_create()
Face_ID = -1
pev_person_name = ""
y_ID = []
x_train = []
```

Bu kısımda, LBPH yüz tanıyıcı oluşturulur ve kullanılacak değişkenler tanımlanır.

`Face_ID` kişilere unique bir kimlik atamak için kullanılır. `pev_person_name` ise bir önceki kişinin adını takip etmek için kullanılır.

Eğitim İçin Klasör Gezme:

```
Face_Images = os.path.join(os.getcwd(), "Face_Images") #Tell the program where we have
print (Face_Images)

for root, dirs, files in os.walk(Face_Images): #go to the face image directory
    for file in files: #check every directory in it
        if file.endswith(".jpeg") or file.endswith(".jpg") or file.endswith(".png"):
            path = os.path.join(root, file)
            person_name = os.path.basename(root)
            print(path, person_name)
```

Yüz Tanımlama ve Eğitim Verisi Oluşturma:

```
if pev_person_name!=person_name: #Check if the name of person has changed
    Face_ID=Face_ID+1 #If yes increment the ID count
    pev_person_name = person_name

Gery_Image = Image.open(path).convert("L") # convert the image to greysclae using Pi
Crop_Image = Gery_Image.resize( (800,800) , Image.ANTIALIAS) #Crop the Grey Image to
Final_Image = np.array(Crop_Image, "uint8")
#print(Numpy_Image)
faces = face_cascade.detectMultiScale(Final_Image, scaleFactor=1.5, minNeighbors=5)
print (Face_ID,faces)
```



```
for (x,y,w,h) in faces:
    roi = Final_Image[y:y+h, x:x+w] #crop the Region of Interest (ROI)
    x_train.append(roi)
    y_ID.append(Face_ID)
```

Bu kısımda, LBPH yüz tanıyıcı oluşturulur ve kullanılacak değişkenler tanımlanır.

`Face_ID` kişilere unique bir kimlik atamak için kullanılır. `pev_person_name` ise bir önceki kişinin adını takip etmek için kullanılır.

Eğitim Verileri ile Yüz Tanıyıcıyı Eğitme ve YML Dosyası Oluşturma:

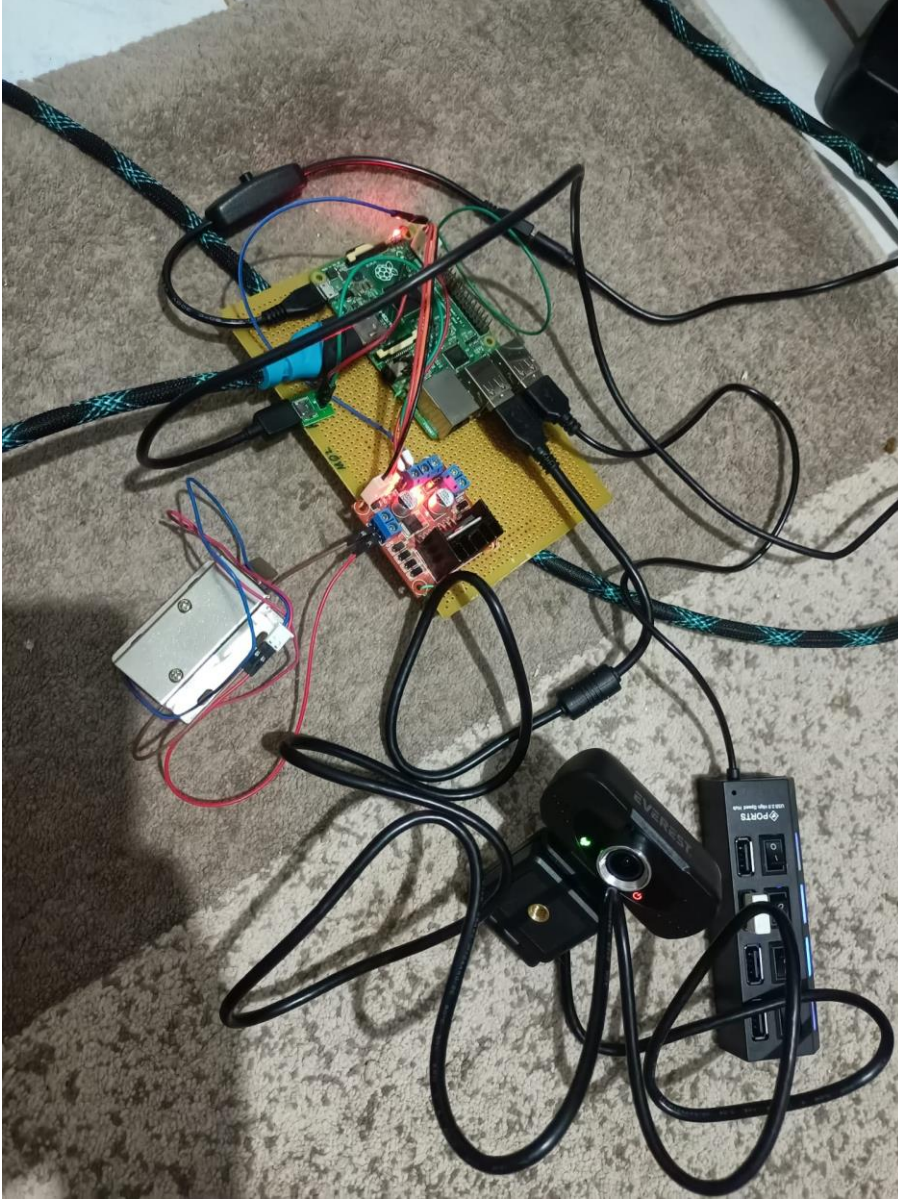
```
recognizer.train(x_train, np.array(y_ID))
recognizer.save("face-trainer.yml")
```

Bu bölümde, `x_train` ve `y_ID` listelerini kullanarak LBPH yüz tanıyıcı eğitilir.

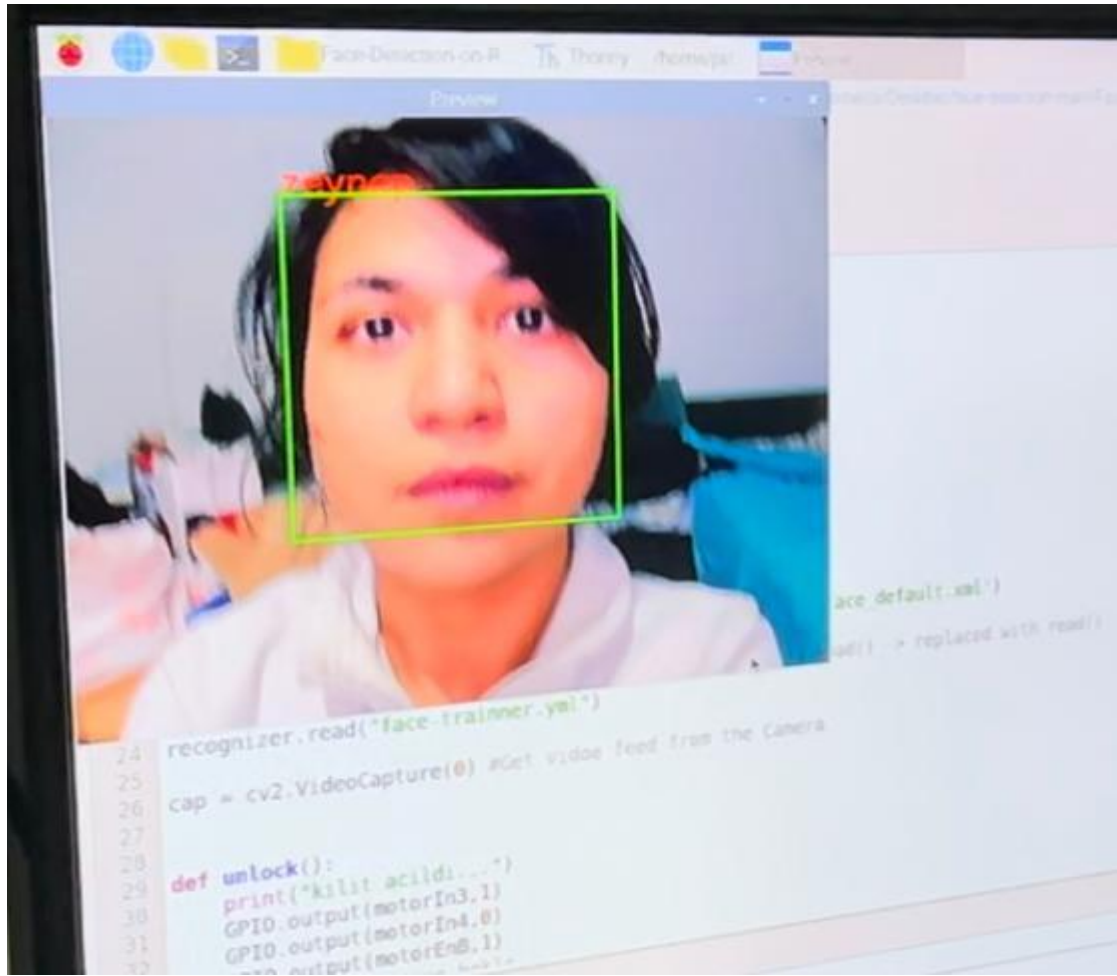
Ardından eğitilmiş model, "face-trainer.yml" adlı bir YAML dosyasına kaydedilir.

Bu kod, yüz tanıma uygulaması için eğitim verisi oluşturur ve ardından bu veriyi kullanarak bir yüz tanıyıcı modeli eğitir.

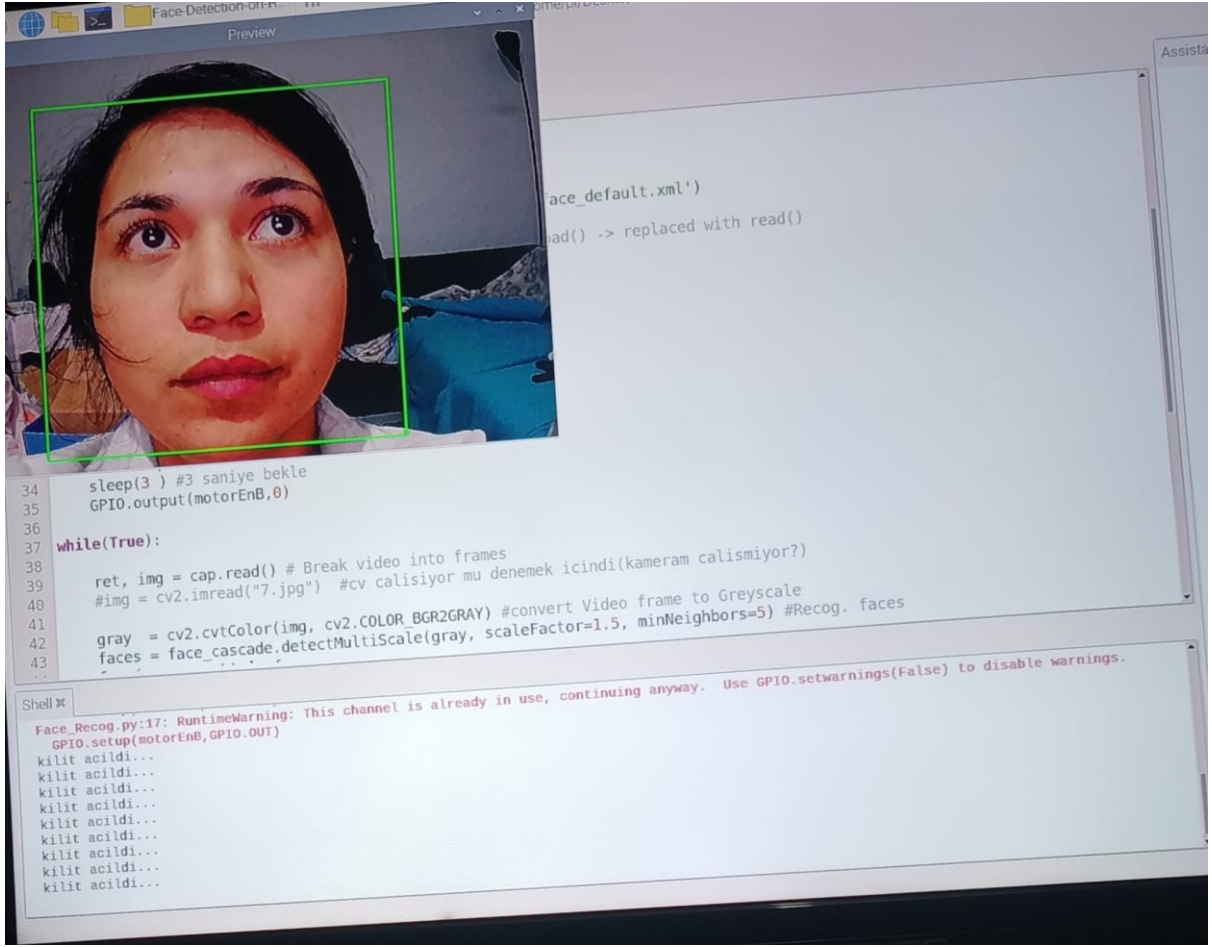
4.TARTIŞMA VE SONUÇ



Sistem, sağlam şekilde muhafaza edilebilmesi amacıyla bir plaka üzerine sabitlenmiş şekilde kurulmuştur. Böylece hem taşınabilir hale getirilmiş , hem de kamera ve motor gibi çevre birimlerin sökölüp takılması gibi durumlarda oluşabilecek kablo bağlantı hataları minimum düzeye indirilmiştir.



Sistem, yüz tanıma fonksiyonunu başarıyla yerine getirmektedir.



Sistem, yüz tanıma fonksiyonunu başarıyla yerine getirdikten sonra motor kısmındaki kod devreye girerek kilidi başarıyla açar.

SONUÇ

İleride gerçekleştirilecek çalışmalarda; sistemde daha yüksek çözünürlüklü ve gece görüşü, derinlik algılama sensörü gibi daha fazla özellik barındıran kameralar kullanılabilir ve böylece sisteme yeni özellikler eklenebilir.

KAYNAKLAR

<https://bizimdergi.com/yapay-zeka-ile-gelistirilen-guvenlik-sistemleri/>

<https://www.netadvi.com/blog/virtual-environment-venv-sanal-ortam-nedir/>

<https://piwheels.org/>