T. C.
YOZGAT BOZOK ÜNİVERSİTESİ
MÜHENDİSLİK MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

GELİŞMİŞ HAREKET ALGILAMA: MEDİAPİPE VE ARDUİNO İLE LED YÖNETİMİ

Alper KARATAŞ
Ayşe Gül SAYGI
Hatice Dilara BÜKER
Hüseyin ÇORAKLI
Simge Nur DEMİRAL

BİLGİSAYAR MİMARİSİ
TEKNİK RAPORU
Dr. Öğr. MEHMET KARABULUT

YOZGAT 2023

İçindekiler

1. Giriş 3
2. Metod Ve Metaryal 4
2.1. Arduino İle LED Kontrol Devresi Tasarımı Ve Kodlama 5
2.1.1. Arduino Kodları 6
2.2. Python ile Görüntü İşleme: Mediapipe ile El İzleme Teknolojisi 8
2.2.1. Mediapipe Kütüphanesi 8
2.2.2. OpenCv Kütüphanesi 9
2.2.3. Python Kodları 11
2.3. Arduino ve Python Arasındaki Etkileşim 13
3. Sonuç 13
4. Kaynakça 14

1. Giriş

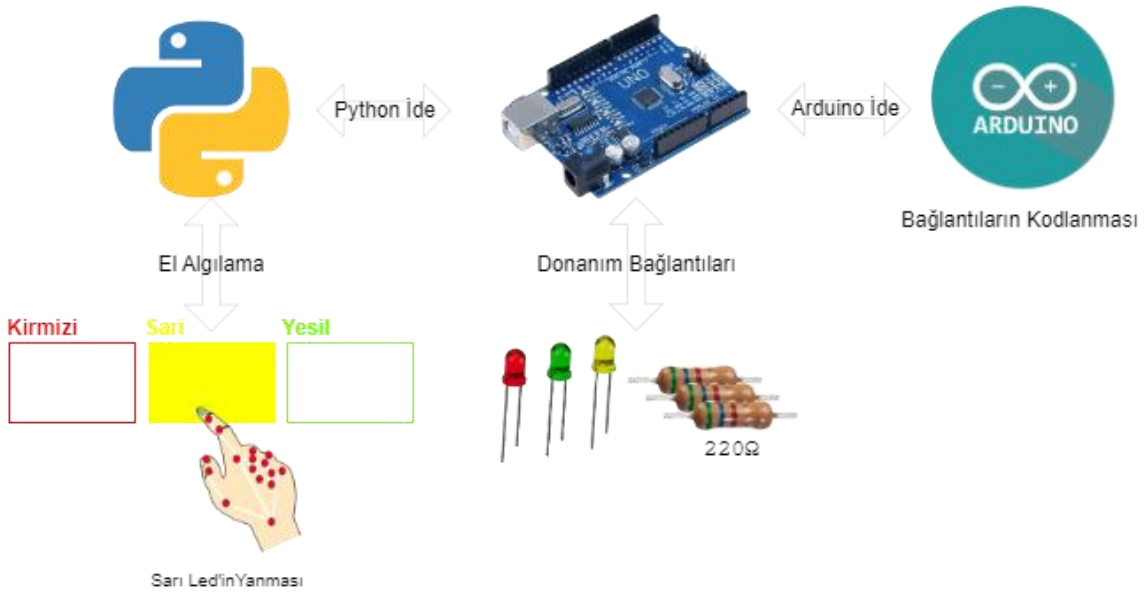
Günümüzde, etkileşimli teknolojilerin hızla gelişmesiyle birlikte, kullanıcı arayüzleri ve kontrol sistemleri de yenilikçi ve etkileyici bir seviyeye ulaşmıştır. Bu bağlamda, el hareketleriyle kontrol edilen sistemler, kullanıcı deneyimini daha da zenginleştiren ve teknoloji ile insan etkileşimini optimize eden önemli bir araştırma alanı haline gelmiştir. Bu projede tez çalışması, el izleme teknolojisi ve görsel bilgi işleme yöntemlerini kullanarak, kullanıcının el hareketleri aracılığıyla bir LED sistemi kontrol etme konseptini ele almaktadır.

Bu çalışmanın temel amacı, Mediapipe ve OpenCV gibi güçlü görüntü işleme kütüphaneleri kullanılarak el izleme teknolojisinin entegrasyonunu içeren bir sistem tasarlamak ve geliştirmektir. Projemiz, el hareketlerini algılayarak Arduino tabanlı bir LED kontrol sistemini etkin bir şekilde yönetmeyi hedeflemektedir. Elde edilen sonuçlar, kullanıcıların el hareketleriyle gerçekleştirdiği kontrol mekanizmasının güvenilirliği ve kullanım kolaylığı açısından değerlendirilecektir.

Bu çalışmanın önemi, El izleme teknolojisinin ve görsel bilgi işleme algoritmalarının bu bağlamda nasıl kullanılabileceğini anlamak, interaktif sistemlerin geliştirilmesinde yeni perspektifler sunabilir. Ayrıca, proje özellikle engellilerin günlük yaşamlarını kolaylaştırabilecek bir teknolojik çözüm sunma potansiyeli de bulunmaktadır.

Bu çalışmada, el izleme teknolojisinin ve görsel bilgi işleme yöntemlerinin bir araya getirildiği bir sistem tasarımını içermektedir. Arduino platformu üzerinde çalışan bir LED kontrol sistemini, Mediapipe ve OpenCV araçları kullanarak geliştirmek, bu çalışmanın ana hedefini oluşturmaktadır.

2. Metot ve Materyal



Şekil 1 Projenin Akış Şeması

Bu proje, Mediapipe ve OpenCV kullanılarak el izleme teknolojisinin entegrasyonunu içeren bir LED kontrol sistemini ele almaktadır. Kullanıcının el hareketleri, belirlenmiş bölgelerdeki LED'leri kontrol etmek için kullanılmaktadır. Örneğin, belirli bir bölgedeki el hareketiyle sarı LED açılırken, farklı bir bölgedeki el hareketiyle yeşil veya kırmızı LED'ler kontrol edilebilmektedir. Sistem, pratik bir örneği olarak interaktif kontrol sistemleri ve engelliler için erişilebilirlik konusunda önemli bir katkı sunma potansiyeline sahiptir. Bu çalışma, el izleme teknolojisinin ve görsel bilgi işleme algoritmalarının etkileşimli kontrol sistemlerinin geliştirilmesindeki uygulamalı bir örneğini sunmayı hedeflemektedir.

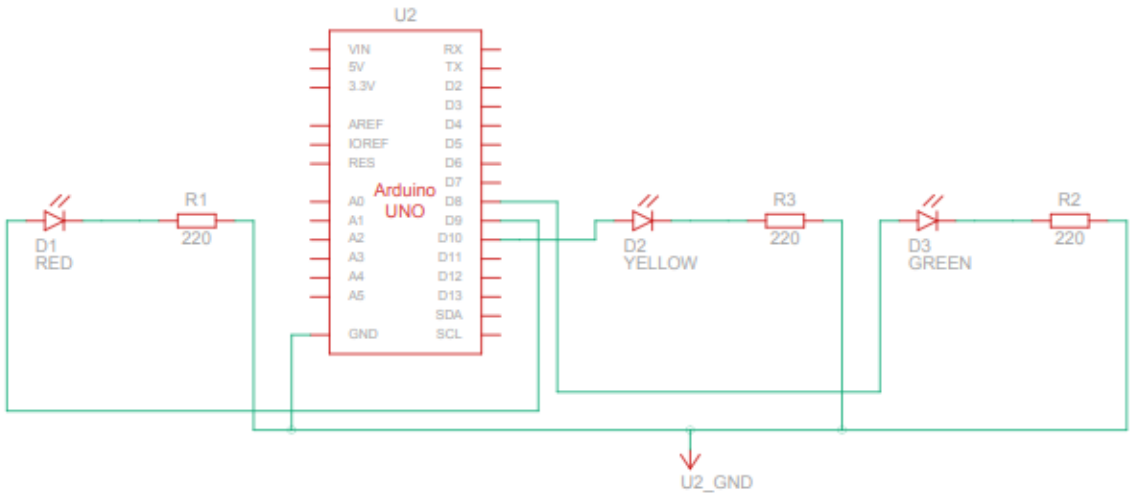
Proje başlangıcında, LED kontrol devresi tasarlandı ve Arduino IDE kullanılarak bu devreye özgü kodlamalar gerçekleştirildi. Ardından, görüntü işleme konseptini uygulamak için Python IDE ve Mediapipe kütüphanesi kullanıldı. Bu aşamada, el izleme teknolojisinin entegrasyonuyla elde edilen bilgiler, Arduino tabanlı LED kontrol sistemine etkileşimli bir şekilde aktarıldı. Bu iki aşama, projenin donanım ve yazılım bileşenlerini başarıyla birleştiren bir bütün oluşturdu.

2.1. Arduino ile LED Kontrol Devresi Tasarımı ve Kodlama

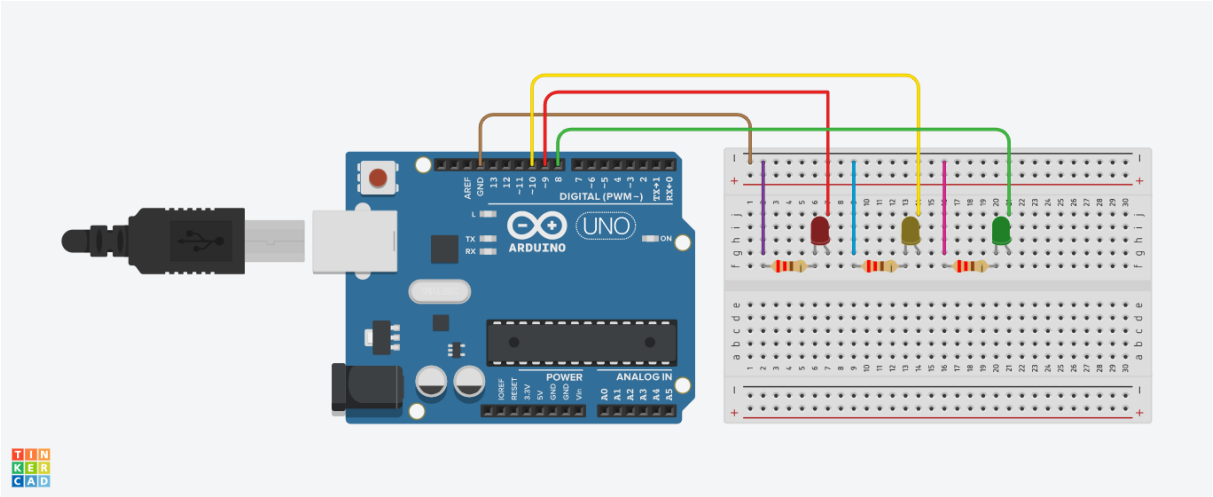
Arduino kısmında, projenin temel donanımı olan LED kontrol devresi tasarlanmıştır. Bu devre, kullanıcı tarafından gerçekleştirilen el hareketlerine bağlı olarak LED'leri kontrol etme yeteneği sağlamak amacıyla oluşturuldu. İlk adım olarak, devrede üç farklı renkte LED (Kırmızı, Yeşil, Sarı) kullanıldı ve her bir LED, Arduino platformuna bağlı olan belirli bir pin ile ilişkilendirildi. Ayrıca Arduino kontrol devresinde, her bir LED'in anotlarına bağlı olan dirençler kullanılarak akım sınırlaması sağlanmıştır. Her LED için birer adet 220 ohm direnç tercih edilmiştir. Bu dirençler, LED'lere bağlı pinlere doğrudan bağlanarak, LED'lerin belirli bir akım değerini aşmalarını önler.

Ardından, Arduino IDE kullanılarak, belirli bir LED'yi açma ve kapama işlemlerini gerçekleştirecek kodlar yazıldı. Her renk için ayrı bir pin tanımlandı ve kullanıcı tarafından gönderilen komutlara göre ilgili LED'nin durumu kontrol edildi. Örneğin, '1' komutu alındığında, kırmızı LED açıldı; '0' komutu alındığında ise kırmızı LED kapatıldı. Aynı prensip, yeşil ve sarı LED'ler için de geçerlidir. Arduino kodu aynı zamanda, belirli bir LED'nin durumunu belirlemek için kullanılan pinlerin başlangıç durumlarını belirleme, pin modlarını tanımlama ve seri iletişim başlatma gibi başlangıç işlemlerini içerir. Bu kodlama adımları, Arduino'nun, Python ile iletişim kurarak elde edilen komutları doğru bir şekilde yorumlamasını ve LED'leri kontrol etmesini sağlar.

Sonuç olarak, Arduino kısmı, projenin fiziksel yönünü oluşturan LED kontrol devresinin tasarımı ve kodlanması aşamasını içerir. Arduino, el izleme teknolojisi tarafından sağlanan bilgileri temel alarak LED'leri etkileşimli bir şekilde kontrol eden bir arayüz sağlar.



Şekil 2 Devrenin Şematik Görünümü



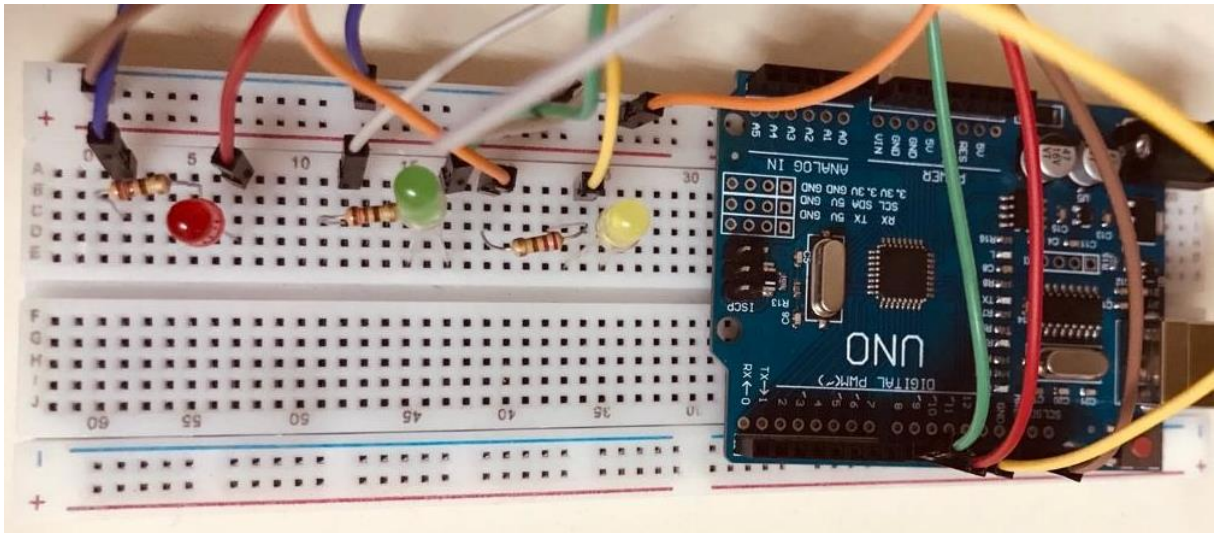
Şekil 3 Devrenin Tinkercad Çizimi

2.1.1. Arduino Kodları

```
// Sabitler: LED pin numaraları
const int kirmizi = 9;
const int yesil = 8;
const int sari = 10;

void setup() {
  // Pin modları ayarlanıyor: OUTPUT (çıkış) olarak tanımlanıyorlar
  pinMode(kirmizi, OUTPUT);
  pinMode(yesil, OUTPUT);
  pinMode(sari, OUTPUT);
  // Seri iletişim başlatılıyor: 9600 bps hızında
  Serial.begin(9600);
}

void loop() {
  // Seri bağlantıda veri var mı kontrolü yapılıyor
  if (Serial.available() > 0) {
    // Bir karakter okunuyor
    char command = Serial.read();

    // Gelen komutlara göre LED'leri kontrol etme
```

```
        if (command == '1') {
            digitalWrite(kirmizi, HIGH);  // Kırmızı LED'i aç
        } else if (command == '0') {
            digitalWrite(kirmizi, LOW);   // Kırmızı LED'i kapat
        }

        if (command == '2') {
            digitalWrite(yesil, HIGH);  // Yeşil LED'i aç
        } else if (command == '3') {
            digitalWrite(yesil, LOW);   // Yeşil LED'i kapat
        }

        if (command == '4') {
            digitalWrite(sari, HIGH);  // Sarı LED'i aç
        } else if (command == '5') {
            digitalWrite(sari, LOW);   // Sarı LED'i kapat
        }
    }
}
```

Tablo 1 Projenin Arduino Kodlaması



Tablo 2 Devre Tasarımı

2.2. Python ile Görüntü İşleme: Mediapipe ile El İzleme Teknolojisi

Bu projede, el izleme teknolojisinin kullanıldığı bir sistem geliştirildi. Python dilinde Mediapipe kütüphanesiyle görüntü işleme yapılarak, kameradan el izleme gerçekleştirildi. El izleme sonuçlarına bağlı olarak belirlenen bölgelerle ilişkilendirilmiş LED'leri kontrol etmek üzere Arduino kullanıldı. El hareketleriyle kırmızı, yeşil ve sarı LED'leri etkileşimli bir şekilde kontrol eden bir sistem tasarlandı.

Python kısmında, görüntü işleme yetenekleri sağlayan Mediapipe kütüphanesi kullanılarak el izleme teknolojisinin entegrasyonu gerçekleştirilmiştir. İlk olarak, OpenCV kütüphanesi aracılığıyla kameradan elde edilen görüntü, Mediapipe Hands modülüne verilmek üzere işlenmiştir. Bu sayede, eldeki belirli noktaların konumları tespit edilmiş ve el izleme sağlanmıştır.

Proje kapsamında, özellikle elin ucu (parmak ucu) noktası üzerinden elin konumunu belirleyen bir algoritma geliştirilmiştir. Mediapipe kütüphanesinin sağladığı el izleme bilgileri kullanılarak, elin bulunduğu konumun ekran üzerindeki belirli bölgelerle ilişkilendirilmesi yapılmıştır. İşaret Parmağının ucu ile gösterilen bölgelerin algılandığı bir sistem oluşturulmuştur.

Belirlenen bölgeler arasında kırmızı, yeşil ve sarı LED'leri kontrol etme yeteneği bulunmaktadır. Örneğin, elin belirli bir bölgesine giriş yapılması durumunda Python tarafından üretilen komutlar aracılığıyla Arduino'ya "1" komutu gönderilir ve bu, kırmızı LED'in açılmasını tetikler.

Python kodları, el izleme sonuçlarına bağlı olarak üretilen bu komutları Arduino'ya seri iletişim yoluyla gönderir. Bu sayede, görüntü işleme tarafından elde edilen bilgiler, fiziksel dünyada LED'leri kontrol etmek üzere Arduino tarafından işlenir.

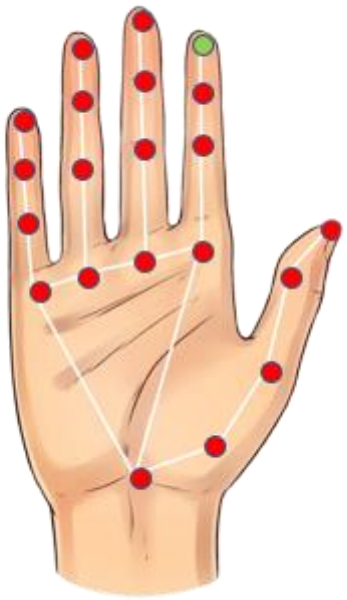
Bu entegrasyon, projenin temel özelliklerinden biri olan el izleme teknolojisinin, Arduino üzerinde fiziksel bir çıktıya dönüştürülmesini sağlar. Python kısmı, bu bilgileri işleyerek etkileşimli bir kontrol sistemi oluşturur ve Arduino ile güçlü bir şekilde entegre edilmiş bir projeyi tamamlar.

2.2.1. Mediapipe Kütüphanesi

MediaPipe, Google tarafından geliştirilen bir açık kaynaklı bir kütüphanedir ve görüntü ve video analizi için kullanılır. MediaPipe, çeşitli görsel bilgi işleme görevlerini kolaylaştırmak için önceden eğitilmiş modeller ve araçlar sağlar. Bu kütüphane, yüz algılama, yüz ifadesi

analizi, el izleme, vücut tutumu tahmini, nesne algılama ve diğer benzeri görevleri gerçekleştirebilen bir dizi modül içerir. MediaPipe'i kullanabilmek için, Python programlama dilini ve OpenCV gibi görüntü işleme kütüphanelerini kullanarak bilgisayar görüşü projeleri geliştirmeniz gerekebilir [1].

Bu projede, Mediapipe kütüphanesi, görüntü işleme yetenekleriyle el izleme teknolojisini başarıyla uygulamıştır. Kameradan alınan görüntü üzerindeki elin konumunu ve belirli noktalarını tespit ederek, parmak ucu, baş parmak, işaret parmağı, orta parmak gibi önemli el noktalarını belirlemiştir. Bu noktaların ekran koordinatlarına dönüştürülmesi sayesinde elin konumu ve parmak hareketleri belirlenmiş ve bu bilgiler, belirli bölgelere göre komut üretimini sağlamak üzere kullanılmıştır. El izleme sonuçları, OpenCV kütüphanesi yardımıyla görüntü üzerinde görselleştirilmiş, böylece el hareketlerinin doğru bir şekilde algılandığını hem kullanıcıya göstermiş hem de geliştiricinin projenin ilerlemesini izlemesine olanak tanımıştır. Sonuç olarak, Mediapipe kütüphanesi, projenin temelini oluşturan el izleme yetenekleriyle etkileşimli bir kontrol sistemi geliştirmek adına kritik bir rol oynamıştır.



Şekil 4 El Hareketleri Algılandığında Alınan Görüntü

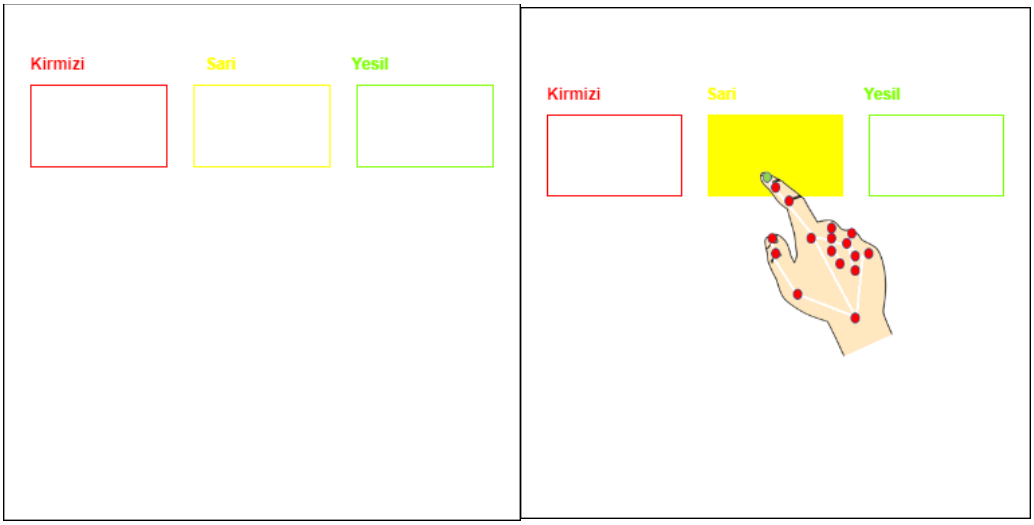
2.2.2. OpenCv Kütüphanesi

OpenCV, bilgisayar görüşü ve makine görüşü uygulamaları geliştirmek amacıyla kullanılan açık kaynaklı bir kütüphanedir. C++, Python ve Java gibi dillerle uyumlu olan bu kütüphane, görüntü işleme, nesne tanıma, yüz tanıma, hareket analizi, kamera kalibrasyonu gibi birçok

alanda kullanılır. Geliştiricilere geniş bir yelpazede görüntü işleme ve analiz araçları sunan OpenCV, güvenlik sistemleri, otomatik odaklama, video izleme gibi çeşitli uygulamalarda kullanılabilir. Ayrıca, ücretsiz olarak kullanılabilir olması ve büyük bir kullanıcı topluluğu tarafından desteklenmesi, sürekli güncellenen ve gelişen bir kütüphane olmasını sağlar [2].

Projede, OpenCV kütüphanesi, el izleme sonuçlarının görselleştirilmesi ve ekran üzerindeki bilgilerin işlenmesi için kritik bir rol oynamıştır. Kamera tarafından sağlanan görüntü, OpenCV aracılığıyla alınmış ve Mediapipe kütüphanesi kullanılarak gerçekleştirilen el izleme işlemleri bu görüntü üzerinde uygulanmıştır. OpenCV, el izleme sonuçlarına bağlı olarak elde edilen noktaları çizim işlevleriyle ekranda belirginleştirmiştir. El izleme sonuçları, özellikle parmak ucu, baş parmak ve diğer el noktaları, kullanıcının el hareketlerini takip etmek ve projenin ilerlemesini görsel olarak izlemek adına ekran üzerinde işaretlenmiştir.

Belirlenen bölgeler, kırmızı, yeşil ve sarı LED'leri kontrol etmek üzere tanımlanmıştır. OpenCV, bu bölgeleri çerçeve içine alarak ve bu bölgeleri belirginleştirerek kullanıcının el pozisyonunu anlamasına yardımcı olmuştur. Ayrıca, ekran üzerinde belirli bölgelere ilişkin metin ve görsel işaretlemeler yapılarak, kullanıcının hangi renk LED'in kontrol edildiğini açıkça görmesi sağlanmıştır. Bu sayede, OpenCV'nin görüntü işleme yetenekleri, projenin etkileşimli kontrol sistemini görsel olarak desteklemiş ve kullanıcıya anlık geri bildirim sunmuştur.



Şekil 5 (a) Webcam Açıldığında Alınan Görüntü (b) Sarı Ledin Algılandığı Durum

2.2.3. Python Kodları

```
import cv2
import mediapipe as mp
import time
import serial

ser = serial.Serial('COM3', 9600)
time.sleep(1)
webcam = cv2.VideoCapture(0)

el = mp.solutions.hands
el_cizim = mp.solutions.drawing_utils

with el.Hands(static_image_mode=False, max_num_hands=2,
min_detection_confidence=0.5) as eller:
    while True:
        _, frame = webcam.read()
        frame = cv2.flip(frame, 1)
        rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        result = eller.process(rgb)
        yuk, gen, _ = frame.shape
        if result.multi_hand_landmarks:
            for cizim in result.multi_hand_landmarks:
                el_cizim.draw_landmarks(frame, cizim, el.HAND_CONNECTIONS)
                koordinat1 = cizim.landmark[8]
                x = int(koordinat1.x * gen)
                y = int(koordinat1.y * yuk)
                cv2.circle(frame, (x, y), 5, (0, 255, 0), -1)
                if 30 <= x <= 160 and 30 <= y <= 120:
                    cv2.rectangle(frame, (30, 30), (160, 120), (0, 0, 255), -1)
                    ser.write(b'1')
                else:
                    ser.write(b'0')
```

```
            if 430 <= x <= 560 and 30 <= y <= 120:
                cv2.rectangle(frame, (430, 30), (560, 120), (0, 255, 0), -1)
                ser.write(b'2')
            else:
                ser.write(b'3')

            if 230 <= x <= 360 and 30 <= y <= 120:
                cv2.rectangle(frame, (230, 30), (360, 120), (0, 255, 255), -1)  # Sarı renkli
kutu
                ser.write(b'4')
            else:
                ser.write(b'5')

        cv2.rectangle(frame, (30, 30), (160, 120), (0, 0, 255), 3)
        cv2.rectangle(frame, (430, 30), (560, 120), (0, 255, 0), 3)
        cv2.rectangle(frame, (230, 30), (360, 120), (0, 255, 255), 3)  # Sarı renkli kutu
        cv2.putText(frame, "Kirmizi ", (30, 20), cv2.FONT_HERSHEY_PLAIN, 1, (0, 0,
255), 2)
        cv2.putText(frame, "Yesil ", (430, 20), cv2.FONT_HERSHEY_PLAIN, 1, (0, 255,
0), 2)
        cv2.putText(frame, "Sari ", (230, 20), cv2.FONT_HERSHEY_PLAIN, 1, (0, 255,
255), 2)  # Sarı renkli kutu

        cv2.imshow("pen", frame)
        if cv2.waitKey(5) & 0xFF == ord("q"):
            break

webcam.release()
cv2.destroyAllWindows()
```

Tablo 3 Projenin Python Kodlaması

2.3. Arduino ve Python Arasındaki Etkileşim

Bu projede, Arduino ve Python arasındaki etkileşim, seri iletişim protokolü aracılığıyla sağlanmaktadır. Arduino tarafında geliştirilen kod, belirli durum ve komutlara bağlı olarak kırmızı, yeşil ve sarı LED'leri kontrol eden işlemleri içermektedir. Öte yandan, Python tarafında kullanılan Mediapipe kütüphanesi, kameradan alınan görüntü üzerinde el izleme işlemi gerçekleştirerek belirli el pozisyonlarını tespit eder. El izleme sonuçlarına bağlı olarak belirlenen bölgelere göre Python kodu, Arduino'ya yönlendirmek üzere seri iletişim portu aracılığıyla belirli komutları gönderir. Örneğin, el belirli bir bölgeye getirildiğinde, Python tarafından üretilen '1' komutu, Arduino'ya iletilerek kırmızı LED'in açılması sağlanır. Bu seri iletişim protokolü, projenin temel fonksiyonları arasında etkili bir iletişim köprüsü oluşturarak, Python tarafındaki görüntü işleme sonuçlarını Arduino tarafında fiziksel tepkilere dönüştürme sürecini yönlendirir. Bu, projenin temel başarı faktörlerinden biridir, çünkü kullanıcı el hareketleriyle belirli komutları ileterek LED'leri etkileşimli bir şekilde kontrol edebilir.

3. Sonuç

Bu projenin gerçekleştirilmesi, el izleme teknolojisinin ve görüntü işleme algoritmalarının başarılı bir şekilde entegrasyonunu sağlamıştır. Python ile Mediapipe kütüphanesi kullanılarak el izleme sonuçları başarılı bir şekilde elde edilmiş ve bu bilgiler, belirlenen bölgelere göre komutlara dönüştürülerek Arduino'ya iletilmiştir. Arduino tarafında yazılan kod, bu komutları alarak kırmızı, yeşil ve sarı LED'leri kontrol etmiş ve fiziksel tepkileri gerçekleştirmiştir. Seri iletişim protokolü sayesinde Python ve Arduino arasında sağlanan etkileşim, projenin temel amacını yerine getirerek kullanıcının el hareketleriyle LED'leri kontrol etmesini mümkün kılmıştır. Elde edilen sonuçlar, el izleme teknolojisinin interaktif uygulamalarda başarıyla kullanılabilirliğini ve Python ile Arduino'nun entegrasyonunun projelerde güçlü bir araç olduğunu göstermektedir.

4. Kaynakça

[1] B. B. D. B. Ö. Pınar KIRCI, «El Hareketlerinden İşaret Dilini Algılayıp Yazıya Dönüştürme,» *Avrupa Bilim Ve Teknoloji Dergisi*, no. 36, pp. 32-35, 2022.

[2] S. Brahmbhatt, Practical OpenCV, Apress, 2013, p. 244.