TOBB EKONOMİ VE TEKNOLOJİ ÜNİVERSİTESİ

ELE495-BİTİRME PROJESİ FİNAL RAPORU

GÖZ KIRPARAK İLETİŞİM	
Yılmaz Çınar	**161201037**
Mesut Demircioğlu	**161201011**
Raporun Teslim Tarihi	*17.06.2020*

YAZ 2020

İçindekiler

1.Teori

Projenin son raporu olan bu dokümanda bahsedilecek sorunların hepsi çözüme kavuşmuştur. Bu raporun daha çok üzerinde duracağı konular şunlardır; kullanıcı deneyimlerinin geliştirilmesi, rassal testlerde karşılaşılan sorunlara getirilen çözümler ve genel olarak tasarım ve yazılım iyileştirmeleri olacaktır.

Projede yapılan işlere kısa bir özet geçersek. Hastanın yüzünü sürekli olarak izleyen kamera, istem dışı gerçekleşmeyen göz kırpmaları algılar. Kelime seçiminin hızlandırılması için göz kırpmanın süresine göre uzun ve kısa olarak iki mod kullanılmıştır. İkili kodlama bir kelime için gereken göz kırpma sayısını gözle görülür şekilde azaltmıştır.

Kullanıcının doğru kelimeyi seçebilmesi için uzun ve kısa göz kırpma sayısı LCD ekrana yazdırılır. Böylelikle kullanıcının yanlış kelimeyi seçme ihtimali en aza düşürüldü. Seçilen kelime 10 saniyenin sonunda LCD tarafından yazdırılır. Ayrıca her bir kelime için kaydetmiş olduğumuz ses dosyalarından doğru kelime için olanı hoparlörden verilir. Bu sayede kullanıcının iletişime geçtiği kişinin LCD ekrana bakmasına gerek kalmamaktadır.

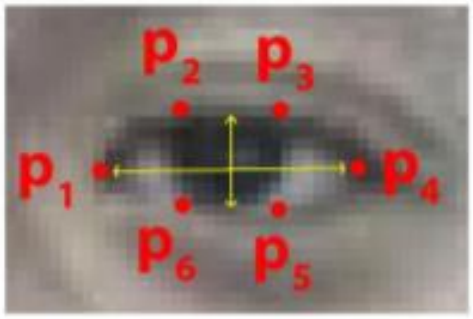
Şekil 1:Sistem düzeneği

2. PROBLEMİN TANITIMI VE LİTERATÜR ARAŞTIRMALARI

ALS hastalığı ilerleyici kas güçsüzlüğü olarak tanımlanır. Zamanla uzuvlar fonksiyonlarını yitirir. Bir sonraki aşamada ise yutma ve konuşma gibi mikro kaslar da işlevsizleşir. *Ancak bu ALS hastalarının bilişsel yetilerini kaybettiği anlamına gelmiyor. Nedir bu bilişsel yeti? Matematik problemi çözebilmek, konuşmak gibi yetiler bilişsel yetiye örnektir. Hastaların bellek, algı ve düşünme gibi becerileri çok büyük oranda sağlam kalıyor. Bu durumda ise tek engelleri bedensel olarak iletişime geçecek kaslarını çalıştıramamak oluyor*[(1)]. *ALS hastaların yaklaşık olarak %75'i iletişimde yardıma ihtiyaç duyar. Yapılan araştırmalara göre %28'i tamamen konuşamazken %47'si konuşma sırasında zorluk çeker. Sadece %25'i anlaşılabilir şekilde konuşabilir*[(2)]. Günümüzde ALS hastalarının kullandığı iletişim yöntemleri ikiye ayrılır. Bir kısım teknolojiden faydalanırken, diğer yöntemler faydalanmaz. Aslında teknolojisiz yöntem, teknoloji kullanarak geliştirilen yöntemlerin de temelini oluşturur. Kabul görmüş birkaç uygulamaya gelirsek; tek göz kırpma evet demekken, iki kere göz kırpma hayır anlamına gelmektedir. Bizim ise burada çözmek istediğimiz olay şudur: Böyle bir beden dilini bilmeyen insanların ALS hastalarının ne dediğini anlaması için bir çevirmen gereklidir.

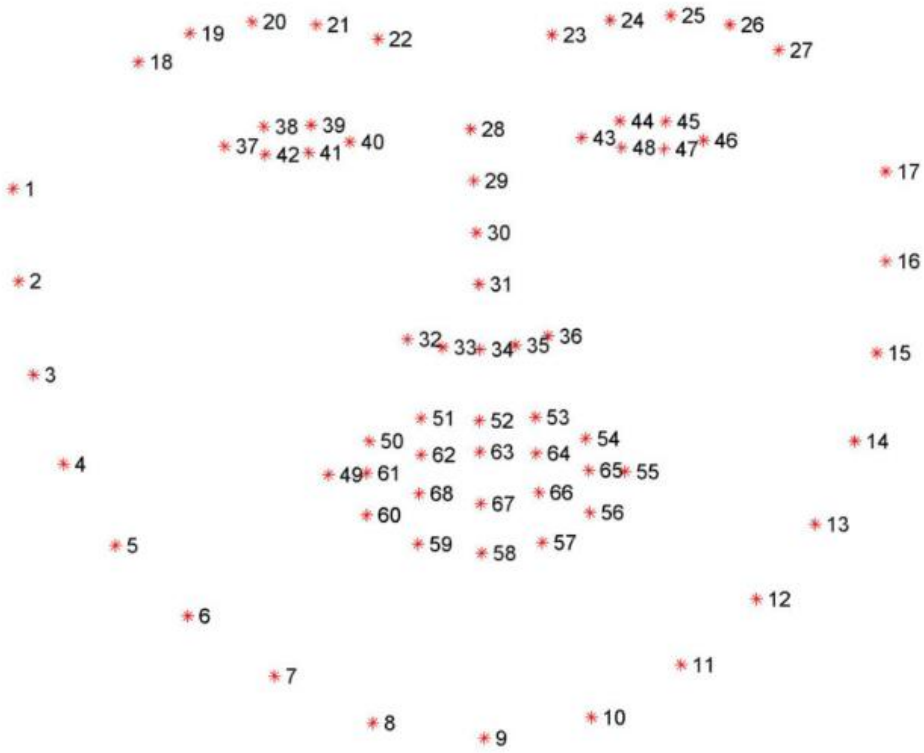
Günümüz teknolojisinde ise bu çevirmenin dijital dünyadan olmayacak olmasını düşünmek mantıksızdır. Göz kırpma ile haberleşme yapılması için ilk olarak gözün kamera tarafından tespit edilmesi gerekir. Bu tespit işleminde Python programlama dili ve OpenCV kütüphanesi kullanılır. Göz kırpma algoritmasının temeli yüzü parselleyerek önce gözün ait olduğu parçayı bulmak ve bu parça üzerinde geometrik olarak işlemler yaparak gözün açıklığını bulmaktır. *Geometrik yöntemler yardımıyla EAR adı verilen göz en boy oranı elde edilir. Bu aşamadan sonra elde edilen EAR verisinin, belirlenen eşiğin altına düşmesi göz kırpma olayı olarak bilgisayar tarafından tespit edilir. Bizim kullanacağımız algoritma da bunla paraleldir*[(3)].

$$\mathrm{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$



Şekil 2:Kırpma algoritması

Bu algoritmanın ve fotoğrafın kaynağına referanslar kısmından ulaşabilirsiniz[(4)].



Şekil 3:yüz algılama

Kullandığımız dlib kütüphanesi yüzü 68 koordinata böler. Şekil 3'de bu noktaları görebilirsiniz. Projemizde ihtiyaç duyulan göz tanıma olduğu için sadece göz ile ilişkili 12 nokta kullanılmıştır. Şekil 2'de görülen 6 nokta buradaki 6 noktayla eştir. Bulunan değerler EAR denklemiyle işleme tabii tutulursa göz kırpma oranı bulunmuş olur.

3.YÖNTEM VE PROJE SÜRECİ (AKTİVİTELER)

Teknik açıdan bu projede kullanılacak işlemci ve çevre birimleri şunlardır:

- Raspberry Pi Model 3B+
- PiCamera v2
- 16x4 LCD Ekran A1604
- WF Tripod WT3110A
- Casper A-113

Bir yazılım projesinin yönteminden bahsedilecekse, başlanılacak ilk nokta kullanılan yazılım kütüphaneleri olmalıdır. Aşağıda sırayla hangi kütüphaneleri kullandığımızı ve kullanış alanlarımızdan birkaçını sizlerle paylaşacağız.

1) Time kütüphanesi : Bu kütüphaneyi kullanma nedenlerimizden ilki sözcük alma sınırı olan on saniyeyi hesaplamaktır. Aynı zamanda test aşamalarında görüntünün FPS(Frame per second) değerini görmek için kullandık.

2)RPLCD Kütüphanesi : Burada sadece LCD ekranımızı kontrol etmek için halihazırda bulunan fonksiyonları kullandık.

3)Math Kütüphanesi : Basit aritmetik işlemler hariç kullandığımız her türlü işlemi buradaki fonksiyonlardan elde ettik.

4)Dlib Kütüphanesi : Bu kütüphane hazır olarak kullanacağımız yüz ve göz tanıma algoritmasını oluşturan dataları alıp işlememize yardımcı oluyor.

5)GPIO Kütüphanesi : Bu kütüphane Raspberry Pi üzerinde bulunan General Purpose Input Output pinlerini kullanmamıza yarıyor.

6)CV2-OpenCV Kütüphanesi : Görüntü üzerinde yaptığımız her türlü manipülasyon işlemlerini bu kütüphanenin fonksiyonları ile yapıyoruz.

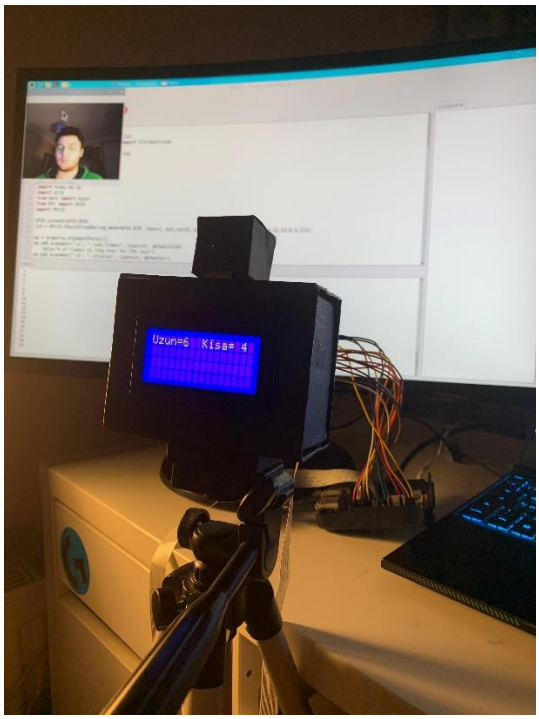
7)Pygame Kütüphanesi : Ses dosyalarının hoparlör tarafından sese çevrilmesini bu kütüphane yapar.

Diğer raporlarımızda belirttiğimiz ve ondan sonra oluşan tasarım sorunlarına ve çözüm yöntemlerine değinmek istiyoruz.

Bu sorunlardan ilki sistemin bir bütün olarak durmamasıydı. Kullanıcının hem göz kırparak kelimeyi seçmesi hem de kelimeyi doğru seçtiğini anlayabilmesi için kameranın ve LCD ekranın yakın olması gerekmekteydi. Bu sorunu LCD ve kamerayı dikeyde ve yatayda aynı konuma yerleştirerek aştık. Böylece kullanıcı LCD'den geri bildirim alırken herhangi bir kamera yanılsaması yaşanmamaktadır. Raspberry pi ve hoparlörü kutunun arkasına yerleştirildi. Böylelikle sistemde bütünlük sağlandı.

Proje sırasında karşılaştığımız en büyük problem olan FPS sorununu dört ayrı iyileştirme yaparak aştık. İlk olarak yapılan araştırmalar sonucu daha hızlı olduğu tespit edilen SD karta geçildi. Yeni SD kartın verileri okurken hız kazandırdığını düşünüyoruz. İkinci yöntemimiz işlemcinin ısınma sorununu çözmek oldu. Yapılan testlerde farkına vardığımız ısınma sorununu ısı transferine uygun bir kasa alarak ve işlemcinin üzerine mekanik soğutucu ekleyerek çözdük. Üçüncü çözüm yöntemimiz ise yazılımda yapılan iyileştirmelerdi. Orijinal raspberry kamerası kullandığımız için kamera verilerini alabileceğimiz daha optimize bir kütüphane bulunmaktaydı. Biz de daha optimize olan bu kütüphaneye geçiş yaptık. Son olarak

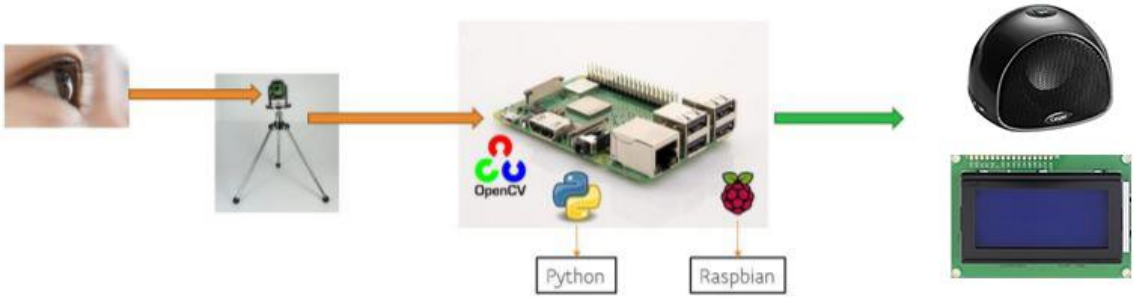
ise görüntü kalitesinin gereksiz olduğunu fark edip piksel sayısını düşürmemiz oldu. Tüm bu işlemlerin katkısıyla FPS değerini istediğimiz seviyeye çekmeyi başardık.



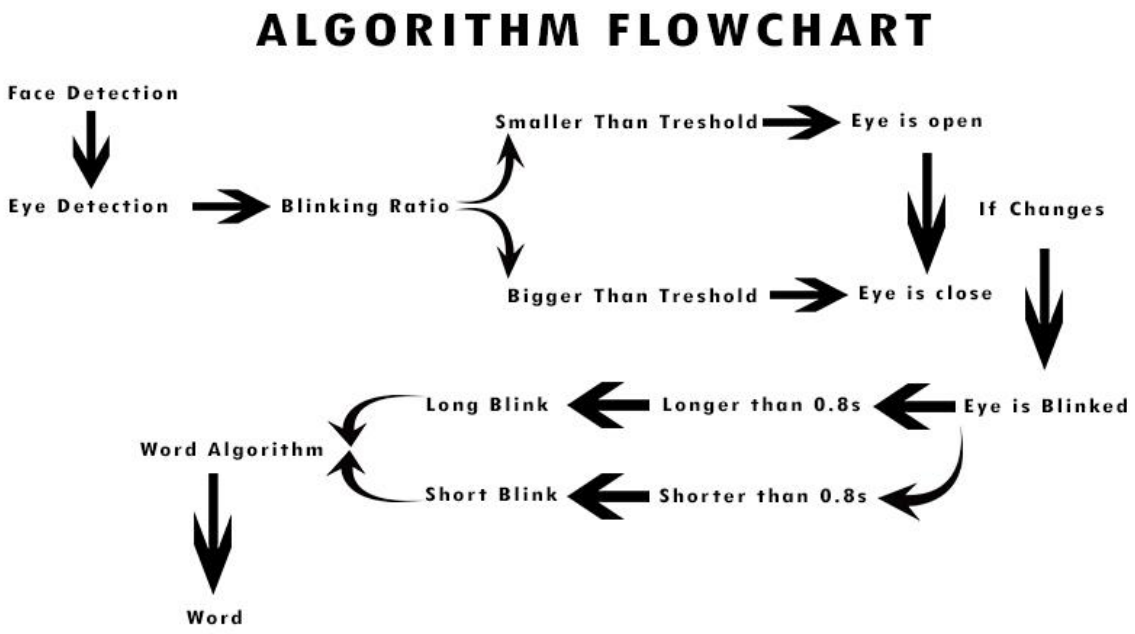
Şekil 5:Projenin çalışma anı

Şekil 4:Raspberry bağlantıları

4. TEKNİK ÖZELLİKLER



Şekil 6:Projenin çalışma şeması



Şekil 7:Projenin algoritma akış şeması

Projenin temel olarak çalışma prensibi yukarıdaki şekillerde verilmiştir. Kısaca özetlemek gerekirse kullanıcının yüzü kamerayla kaydedilir. Görüntü gerçek zamanlı olarak raspberryde işlenir. Eğer kullanıcı kullanım kılavuzunda verilen herhangi bir kelimeyi doğru olarak seçerse, bu kelime hem LCD ekrana yazdırılır hem de hoparlörden verilir.

Projede ikili kodlama kullanarak büyük bir farklılık sağlanmıştır. İkili kodlamadan kasıt uzun ve kısa olarak ikiye ayrılan göz kırpma süresidir. Bu sayede daha az göz kırpma ile daha fazla kelimenin anlatılması başarılmıştır. Döngü içinde gözün kapalı olduğunun tespit edildiği süre 3 döngüye eşit veya büyük ise göz kırpmanın uzun olduğu algılanır. 3 döngüden kısaysa göz kırpmanın kısa olduğu algılanır. Yapılan testlere göre 3 döngünün 0.8 saniye aralığında olduğu tespit edilmiştir. Uzun ve kısa göz kırpma algoritması bir diğer algoritma olan sağ ve sol gözü ayrı kırparak kelime seçme algoritmasına tercih edilmiştir. Seçimdeki en önemli etken kullanıcının tek gözünü kırpmasında zorlanabilecek olmasıdır.

Bir önceki paragrafta bahsedilen uzun ve kısa göz kırpma sayısı LCD ekranda gösterilir. Bunun sebebi kelime seçiminde oluşabilecek hatayı en aza indirgemektir. Kullanıcı LCD ekranda da gösterilen 10 saniye içinde seçimini yapmak zorundadır. Eğer kullanıcı yanlış bir seçim yaparsa LCD ekranda kelime yok yazısı gösterilir.

Kullanıcıya sunulan bir diğer özellik ise kelimenin sesli dışarı verilmesidir. LCD ekran hastanın yüzüne dönük olduğu için hastanın iletişime geçeceği kişi hastanın yakınından ekrana bakmalıdır. Bundan kaçınmak için kelimenin seslendirilmesinin kullanıcıya rahatlık vereceği düşünüldü ve bu özellik eklendi.

Hastanın günlük hayatta en fazla ihtiyaç duyacağı kelimeler seçildi. Tablo 1'de bu kelimelere ve hangi kombinasyonlar sonucu kelimelerin ortaya çıkacağına yer verilmiştir. Kelime algoritmasında daha çok kısa göz kırpma kullanılmasının sebebi süre farkı ve kullanıcının uzun göz kırpma sırasında yorulabilmesidir. Yorulma bulgusuna yapılan testler sonucunda ulaşılmıştır.

KELİME	KISA GÖZ KIRPMA	UZUN GÖZ KIRPMA
Evet	1	0
Hayır	0	1
Acıktım	1	1
Susadım	2	0
Ağrım var	0	2
Üşüdüm	2	1
İyiyim	1	2
Yoruldum	3	0
Tuvalet	3	1
Bay bay	4	0

Tablo 1: Kelime Algoritması

5)BÜTÇE

• Raspberry Pi Model 3B+

https://www.robolinkmarket.com/raspberry-pi-3-model-b

Fiyat = 288,37 TL + KDV

• Raspberry Pi Kamera V2

https://www.robolinkmarket.com/raspberry-pi-kamera-v2

Fiyat = 260,86 TL + KDV

• LCD Ekran (Çin menşeili)

https://www.robitshop.com/urun/16x4-lcd-ekran-1604a-arduino-uyumlu

Fiyat = 24,6 TL + KDV

• WF Tripod WT3110A

Zaten uzun süredir sahip olduğumuz bir üründü. Herhangi bir ücret ödenmedi ancak piyasa değeri 10 Dolar civarı.

• Aliminyum işlemci soğutucu seti

Zaten uzun süredir sahip olduğumuz bir üründü. Herhangi bir ücret ödenmedi ancak piyasa değeri 3 TL civarı.

• Pi Camera Flex Kablo 1M

https://www.robitshop.com/urun/pi-kamera-flex-kablo-raspberry-1m

Fiyat = 14 TL + KDV

• Raspberry Pi Kasa

https://www.robitshop.com/urun/raspberry-pi-fan-uyumlu-kutu-b-2-3-siyah

Fiyat = 22,35 TL + KDV

• Sandisk MicroSD 32 GB

https://www.hepsiburada.com/sandisk-extreme-pro-microsdhc-32gb-sd-adapter-rescue-pro-deluxe-100mb-s-a1-c10-sdsqxcg-032g-gn6ma-p-HBV000006A1VZ?magaza=Hepsiburada

Fiyat = 75,85 TL + KDV

• Raspberry Pi Lisanslı Güç Adaptörü

https://www.robolinkmarket.com/raspberry-pi-lisansli-guc-adaptoru-51v-25a-beyaz

Fiyat = 55,75 TL + KDV

• Casper A-113 Hoparlör

Zaten uzun süredir sahip olduğumuz bir üründü. Herhangi bir ücret ödenmedi ancak piyasa değeri 30 TL civarı.

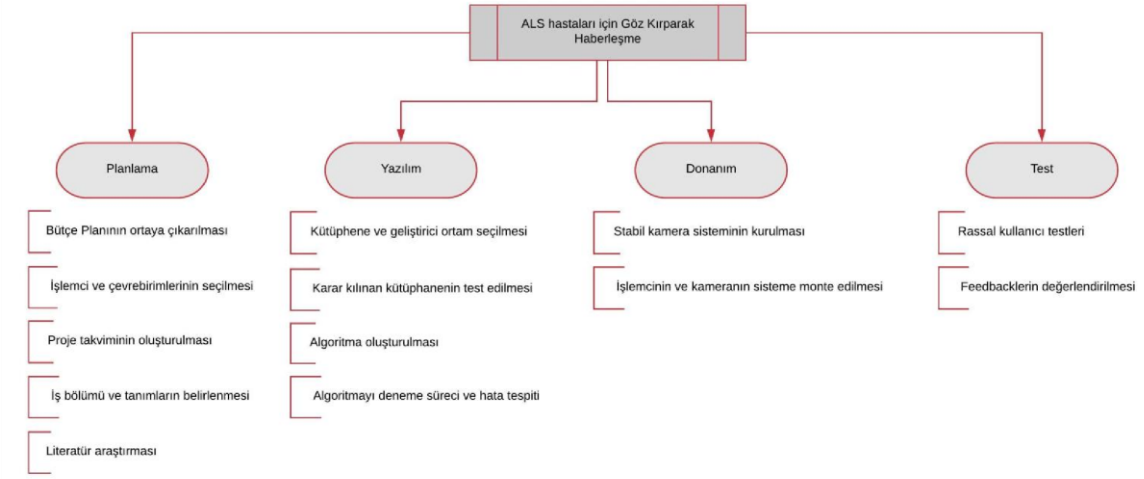
✓ Şu ana kadar toplam harcanan para 741,82 TL + KDV'dir.

✓ Ayrıca daha önceden elimizde bulunan malzemelerin de şu anki piyasa fiyatını eklersek net giderimiz 842,85 TL + KDV 'dir.

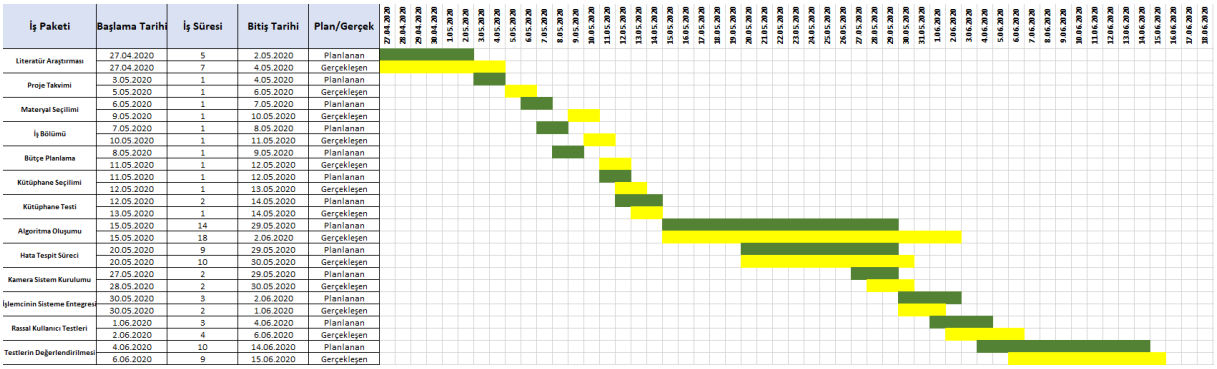
✓ Bazı ürünler elimizde olduğu için ilk raporun yazıldığı tarihteki fiyatlara göre bu rapor yazılmıştır. Güncel fiyatlarla raporda belirtilen fiyatlar arasında farklılık olabilir.

6.PLANLAMA VE ZAMAN TABLOSU

Projenin başında oluşturulan WBS'ye sadık kalındı. WBS'de iş paketleri detaylıca verilmeye çalışıldı. Ancak proje ilerledikçe algoritma oluşturma adı altında toplanan kısmın çok geniş bir paket olduğu fark edildi. Gantt şeması WBS'ye göre yapıldı. Gantt şemasının planlanan ve gerçekleşen olarak ikiye ayrılması projenin olması gerekene göre hangi aşamada olduğunu gösterdi. Gantt şemasında da görüldüğü gibi proje planlamayla eş aşamada yürütülmüştür.



Şekil 8: WBS



İş Paketi	Başlama Tarihi	İş Süresi	Bitiş Tarihi	Plan/Gerçek
Literatür Araştırması	27.04.2020	5	2.05.2020	Planlanan
	27.04.2020	7	4.05.2020	Gerçekleşen
Proje Takvimi	3.05.2020	1	4.05.2020	Planlanan
	5.05.2020	1	6.05.2020	Gerçekleşen
Materyal Seçilimi	6.05.2020	1	7.05.2020	Planlanan
	9.05.2020	1	10.05.2020	Gerçekleşen
İş Bölümü	7.05.2020	1	8.05.2020	Planlanan
	10.05.2020	1	11.05.2020	Gerçekleşen
Bütçe Planlama	8.05.2020	1	9.05.2020	Planlanan
	11.05.2020	1	12.05.2020	Gerçekleşen
Kütüphane Seçilimi	11.05.2020	1	12.05.2020	Planlanan
	12.05.2020	1	13.05.2020	Gerçekleşen
Kütüphane Testi	12.05.2020	2	14.05.2020	Planlanan
	13.05.2020	1	14.05.2020	Gerçekleşen
Algoritma Oluşumu	15.05.2020	14	29.05.2020	Planlanan
	15.05.2020	18	2.06.2020	Gerçekleşen
Hata Tespit Süreci	20.05.2020	9	29.05.2020	Planlanan
	20.05.2020	10	30.05.2020	Gerçekleşen
Kamera Sistem Kurulumu	27.05.2020	2	29.05.2020	Planlanan
	28.05.2020	2	30.05.2020	Gerçekleşen
İşlemcinin Sisteme Entegresi	30.05.2020	3	2.06.2020	Planlanan
	30.05.2020	2	1.06.2020	Gerçekleşen
Rassal Kullanıcı Testleri	1.06.2020	3	4.06.2020	Planlanan
	2.06.2020	4	6.06.2020	Gerçekleşen
Testlerin Değerlendirilmesi	4.06.2020	10	14.06.2020	Planlanan
	6.06.2020	9	15.06.2020	Gerçekleşen

Şekil 9:Gantt şeması

Projede yapılan işler eş zamanlı olarak beraber yürütülmüştür. Grup üyelerinin aynı ortamda bulunamamasından dolayı görüntülü konuşmayla proje ilerletilmiştir. Literatür araştırmaları sonucunda uygun bulunan eklemeler proje üyeleri tarafından ortaklaşa seçilmiştir. Yine kullanılacak ekipmanlara beraber karar verilmiştir. Sistemin tek olmasından dolayı donanım kısmı tamamen Yılmaz Çınar tarafından yapılmıştır. Yazılım kısmında iki birey de katkı vermiştir.

7. REFERANSLAR

(1)https://www.als.org.tr/yardimci-teknoloji-1

(2)Brownlee A, Palovcak M, The Role of Augmentative Communication Devices in the Medical Management of ALS. Journal of NeuroRehabilitation, Volume 22, Number 2, 2007, 445-447.

(3)Tereza Soukupova and Jan Cech , Real-Time Eye Blink Detection using Facial Landmarks, 21st Computer Vision Winter Workshop, Rimske Toplice, Slovenia, February 3–5, 2016

(4) https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

8.KODLAR

```
detector = dlib.get_frontal_face_detector()
print("Predictor verileri alınmaya başlandı.")
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
print("Predictor verileri alındı.")
```

Kod 1

1 numaralı kod parçasında dlib kütüphanesinin yapmış olduğu, yüzü 68 noktaya bölen fonksiyon kullanılır. Şekil 3'de verilen görsel noktaların yüzde nereyi işaret ettiğini gösterir.

```
def get_blinking_ratio(eye_points, facial_landmarks):
    left_point = (facial_landmarks.part(eye_points[0]).x, facial_landmarks.part(eye_points[0]).y)
    right_point = (facial_landmarks.part(eye_points[3]).x, facial_landmarks.part(eye_points[3]).y)
    center_top = midpoint(facial_landmarks.part(eye_points[1]), facial_landmarks.part(eye_points[2]))
    center_bottom = midpoint(facial_landmarks.part(eye_points[5]), facial_landmarks.part(eye_points[4]))

    hor_line = cv2.line(frame, left_point, right_point, (0, 255, 0), 2)
    ver_line = cv2.line(frame, center_top, center_bottom, (0, 255, 0), 2)

    hor_line_lenght = hypot((left_point[0] - right_point[0]), (left_point[1] - right_point[1]))
    ver_line_lenght = hypot((center_top[0] - center_bottom[0]), (center_top[1] - center_bottom[1]))

    ratio = hor_line_lenght / ver_line_lenght
    return ratio
```

Kod 2

get_blinking_ratio adı verilen 2 numaralı kod parçasındaki fonksiyon gözün EAR değerini döndürür. shape_predictor fonksiyonundan alınan noktalardan koordinatlar oluşturulur ve yatay bölü dikey yapılarak oran bulunur.

```
frame = vs.read()
frame = imutils.resize(frame, width=400)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

Kod 3

Görüntünün çözünürlüğünü azaltıp, FPS artışı sağlar.

```
landmarks = predictor(gray, face)

left_eye_ratio = get_blinking_ratio([36, 37, 38, 39, 40, 41], landmarks)
right_eye_ratio = get_blinking_ratio([42, 43, 44, 45, 46, 47], landmarks)
blinking_ratio = (left_eye_ratio + right_eye_ratio) / 2

if blinking_ratio > 4.7:
    counter = counter + 1
    status = 0 # göz kapandı
    #print(counter)
    #cv2.putText(frame, "BLINKING", (50, 150), font, 2, (255, 0, 0))
else:
    status = 1 # göz acildi
    if current_counter != counter:
        if (counter - current_counter) == 3 or (counter - current_counter) > 3:
            uzun_counter = uzun_counter + 1
            #print("Uzun Göz Kırpma")
        if (counter - current_counter) < 3 :
            #print("Kısa Göz Kırpma")
            kisa_counter = kisa_counter + 1
    current_counter = counter
```

Kod 4

get_blinking_ratio fonksiyonundan oranları alır ve oranların ortalamasını alır. Eşik değerinin üzerinde ise göz kapalı demektir. 3 veya daha fazla döngü içerisinde göz kapanmamışsa göz kırpmanın uzun olduğu kabul edilir. Aksi halde kısa göz kırpma yapılmıştır.

```
if kisa_counter == 1 and uzun_counter==0:
    lcd.write_string("Kelime Algilandi")
    time.sleep(2.0)
    lcd.clear()
    lcd.write_string("      Evet      ")
    sound = mixer.Sound('evet.wav')
    sound.play()
    time.sleep(3.0)
```

Kod 5

10 tane olan kelime seçme algoritmasından biri 5 numaralı kod parçasında verilmiştir. Kısa sayaç ve uzun sayaç istenen değerde ise kelime seçilir ve LCD ekranda yazdırılır.