

BLM4021 Gömülü Sistemler

Laboratuvar Projesi Final Raporu

Grup No: 5

Proje Kategorisi: UYG1

Kişilerin Çalışma Yüzdesi:

|  |  |  |  |
| --- | --- | --- | --- |
| **Grup**  **Sorumlusu:** | Muhammet Kayra Bulut | **25** | **Raspberry Pi Gereksinim Tespiti ve Kurulumu** |
|  | Muahanad Tuameh | 25 | **Raspberry Pi kurulum ve Led ekrana yazı yazdırma** |
|  | Barış Can Yılmaz | 25 | Ekip Yönetimi ve Raspberry Pi Kamera Modülü Bağlama |
|  | Gıyat Mosa | 25 | OpenCV ve Dlib ile Görüntü İşleme ve Yazılım Tasarımı |

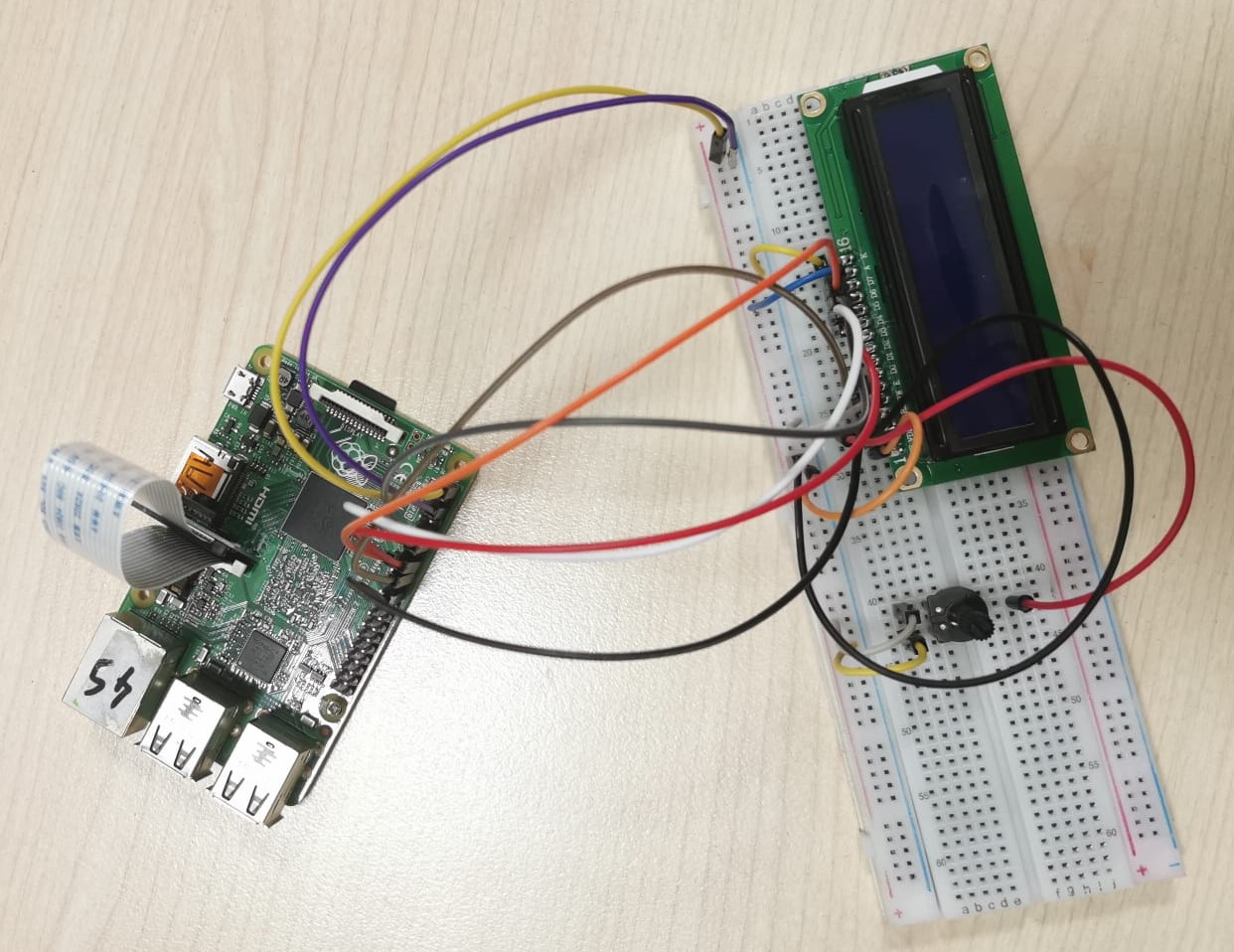
İçerik

1. Giriş ve Proje Tanıtımı…………………………….… Sayfa: 3
2. *Fritsing* ile Ön Tasarım………………………………. Sayfa: 5
3. Kurulan Devre Detayları……………………….…….. Sayfa: 7
4. Yazılım Tasarımı…………………………………..…. Sayfa: 11
5. Sonuçlar, Demo Detayları ve Sunum Linki……..…… Sayfa: 15
6. Referanslar………………………………………..….. Sayfa: 18
7. Giriş ve Proje Tanıtımı

Görüntü işleme, görüntüleri çeşitli yöntemlerle inceleyerek ve değiştirerek bilgi elde etmeyi amaçlayan bir alandır. Görüntü işleme teknikleri, görüntülerdeki bilgiyi daha iyi anlamak ve anlamlandırmak için kullanılır. Bu alanda kullanılan bir yöntem de el tespiti yöntemidir. Bu yöntem, görüntülerde insan elini tanıyarak, el hareketlerini ve pozisyonlarını takip edebilir. Bu sayede, el hareketlerini kontrol eden bir sistem yapılabilir ve bu sistem insanların yapabileceği işlemleri kolaylaştırabilir.

Raspberry Pi, küçük boyutlu ve düşük maliyetli bir mini bilgisayardır. Bu cihaz, birçok farklı işlem için kullanılabilir ve açık kaynaklı bir platformdur. Raspberry Pi, OpenCV ve Python kullanılarak elle kontrol edilebilen bir sistem yapmak için aşağıdaki adımlar izlenebilir:

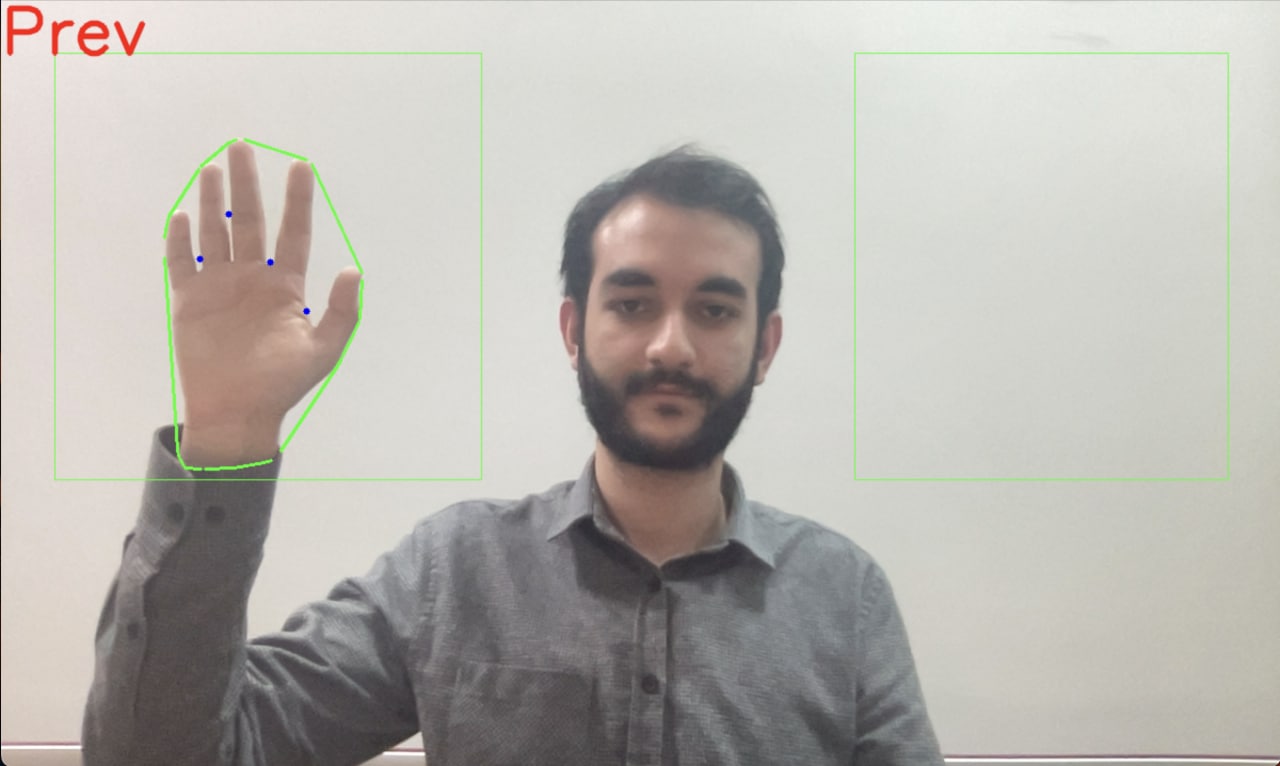
* Raspberry Pi'ye Şekil 1.1’de görüldüğü gibi bir kamera bağlanarak görüntü alma işlemini gerçekleştirilir [1].
* Raspberry Pi'ye OpenCV ve Python kütüphanelerini kurarak bu cihazda görüntü işleme işlemlerini gerçekleştirebiliriz.
* Görüntü işleme algoritmalarını kullanarak eli tespit edebilir ve el hareketlerini takip edebiliriz.
* Eldeki hareketleri sistem tarafından algılayarak, elle kontrol edilebilen bir sistem yapabiliriz. Örneğin, el hareketlerini kullanarak bir robotun hareketini kontrol edebilir veya bir oyunun kontrollerini el hareketleri ile yapabiliriz.
* Bu adımları takip ederek, Raspberry Pi, OpenCV ve Python kullanarak elle kontrol edilebilen bir sistem yapabiliriz.



Şekil 1.1: Raspberry Pi ile Tasarlanmış Gömülü Sistem

OpenCV, görüntü işleme algoritmalarının bulunduğu bir kütüphanedir [5]. Bu kütüphanenin içinde el tespiti için de çeşitli yöntemler bulunur. Bu yöntemlerden biri de el tespiti yöntemidir. Bu yöntem, görüntüde insan elini tespit ederek, el hareketlerini ve pozisyonlarını takip edebilir. Bunun için aşağıdaki adımlar izlenebilir:

* OpenCV kütüphanesini kurarak, görüntü işleme işlemlerini gerçekleştirebileceğimiz bir ortam oluşturabiliriz.
* Görüntüyü aldıktan sonra, el tespiti yöntemini kullanarak insan elini tespit edebiliriz. Bu işlem için OpenCV kütüphanesinde bulunan Contour Approximation yöntemini kullanabiliriz [4].
* El tespit edildikten sonra, el hareketlerini ve pozisyonlarını takip edebiliriz. Bu sayede, el hareketlerini kontrol eden bir sistem yapabilir ve bu sistem insanların yapabileceği işlemleri kolaylaştırabilir.
* Bu adımları takip ederek, şekil 2.2’de görüldüğü gibi OpenCV ile el tespiti yapabiliriz.



Şekil 2.2 OpenCV ile El Tespiti

Contour Approximation, OpenCV'de bulunan bir görüntü işleme yöntemidir. Bu yöntem, bir görüntüdeki nesneleri belirli bir doğrultuda düzleştirerek, bu nesneleri daha kolay tanıyabilmek için kullanılır. Bu vesileyle, bu nesnelerin daha kolay tanınmasını sağlar. Bu yöntem, özellikle görüntülerdeki nesneleri tanımak için kullanılır. Contour Approximation yöntemi, OpenCV kütüphanesinde bulunan cv2.approxPolyDP() fonksiyonu ile gerçekleştirilebilir [3].

1. *Fritsing* ile Ön Tasarım

İlk olarak kamera bağladık. Kamera bağlantısı için herhangi bir arayüz bağlantısına gerek duymadık. Kamerayı Şekil 2.1’de görüldüğü gibi doğrudan Raspberry Pi’a bağladık ve yine Raspberry Pi üzerinden aktifleştirdik.

LCD 1602’nin 1. pinini (ground), Raspberry Pi’ın 5. Pinine (ground) breadboarddaki pozitif hat vasıtasıyla bağladık. Bununla beraber Raspberry Pi’ın 1. pinini (5v) breadboarddaki negatif hatla birleştirdik. LCD 1602’nin 2. pinini (vdd) breadboarddaki negatif hatta bağladık. LCD 1602’nin 3. pinini (V0), breadboarddaki negatif ve pozitif hatta bağladığımız potansiyometrenin orta bacağına bağladık. LCD 1602’nin 4. pinini (RS) Raspberry Pi’ın 22. pinine (GPIO25) ucuna bağladık. LCD 1602’nin 5. pinini (RW), breadboardun pozitif hattına bağladık. LCD 1602 modülünün 6. pinini ise Raspberrry Pi’ın 18. pinine (GPIO24) bağladık. LCD 1602 Modülünün 4 pinini işimize yaramayacağı için boş bırakıyoruz. LCD 1602 Modülünün 11. pinini (DB4) doğrudan Rasbperry Pi’ın 16. pinine (GPIO23) bağladık. LCD 1602 Modülünün 12. pinini (DB5) doğrudan Raspberry Pi’ın 11. pinine (GPIO17) bağladık. LCD 1602 Modülünün 13. pinini (DB6) doğrudan Raspberry Pi’ın 12. pinine (GPIO18) bağladık. LCD 1602 Modülünün 14. pinini (DB7) doğrudan Raspberry Pi’ın 15. pinine (GPIO22) bağladık. LCD 1602 Modülünün 15. pinini (A) ve 16. pinini (K )breadbordun pozitif hattına bağladık. Diagram

Description automatically generated

Şekil 2.1: Fristing Tasarımı

1. Kurulan Devre Detayları

Aldığımız çıktıları Raspberry Pi yardımıyla doğrudan çıktı olarak gösterebilmek için Şekil 3.1’de görülen LCD 1602 modülünü ekledik [2]. Bu modülün bizim için tek işlevi görüntü işleme yardımıyla elde edilen sonucu ekranında anlaşılabilir şekilde yansıtmak.

A close-up of a computer

Description automatically generated with low confidence

Şekil 3.1: LCD 1602 Modülü

Raspberry Pi’a işlemek istediğimiz görüntü verisini aktarabilmek için, yine Raspberry Pi ile uyumlu olan ve Şekil 3.2’de görünen Raspberry Pi Kamera Modülünü kullandık. Bu modülün en büyük avantajı fiyatı ve aynı zamanda Raspberry Pi ile doğrudan haberleşebilmesi. Bu vesileyle arayüz maliyetinden ve arayüzün işlem gücünü yavaşlatmasından kurtulmuş olduk. Raspberry Pi’ın işlem gücünün çok kısıtlı olduğunu dikkate alırsak bu bizim için büyük bir artı.

A close-up of a microchip

Description automatically generated with medium confidence

Şekil 3.2: Raspberry Pi Kamera Modülü

Raspberry Pi’dan aldığımız çıktıyı göstermek için kullandığımız LCD Modülünün ışık seviyesini istediğimiz gibi, fiziksel olarak ayarlamak için Şekil 3.3’de görülen 10K Potansiyometreyi kullandık. Potansiyometre vesilesiyle, LCD 1602 Modülünün ekran parlaklığını ortamın ışı seviyesine göre ayarlama fırsatı yakalıyoruz. Direnç seviyesi arttıkça, parlaklık seviyesi azalıyor. Aynı zamanda direnç seviyesi 10K olduğunda parlaklık en az seviyeyeye geldiği için 10K Potansiyometre kullandık. Bu sayede fiziksel olarak daha rahat kontrol edilebilir bir aralığa sahip olduk. Daha yüksek bir potansiyometre seçmiş olsaydık, hassasiyet elle kontrol edemeyecek kadar yüksek olacaktı.



Şekil 3.3: 10K Potansiyometre

Potansiyometrenin iki bacağından birini breadbordun pozitif hattına, diğeriniyse breadbordun negatif hattına bağladık. Ortanca bacağıysa breadboard yardımıyla LCD 1602 modülünün 3. pinine (V0) bağladık. Buradaki potansiyometrenin işlevi, LCD 1602 Modülünün ekran parlaklığını kontrol etmek. 3. pine bağlanmasının sebebiyse, LCD 1602 Modülünün o pini vasıtasıyla parlaklık kontrolü yapılıyor. Pine gelen voltaj miktarı, parlaklığı belirleyici etken oluyor.

Yaptığımız proje görüntü işlemeye dayalı bir proje olduğu için, kullanmamız gerek kamera modülü önem arz ediyordu. Veriyi kolay alabilmek için uyumlu modülü kullandık. Modül uyumlu olduğu için doğrudan CSI (Camera Serial Interface) aracılığıyla Raspberry Pi bağlantısını gerçekleştirdik.

LCD 1602’nin 1. pinini (ground), Raspberry Pi’ın 5. Pinine (ground) breadboarddaki pozitif hat vasıtasıyla bağladık. Bunu LCD 1602 Modülünün 1. pininin grounda bağlanması gerektiği yaptık.

LCD 1602’nin 2. pinini (vdd) breadboarddaki negatif hatta bağladık. Aynı şekilde Raspberry Pi’ın 1. pinini de breadboarddaki negatif hatta bağladık. Bunun sebebi LCD 1602 Modülünün beslemesini 2. pininden almasıydı.

LCD 1602’nin 3. pinini (V0), breadboarddaki potansiyometrenin orta bacağına bağladık. Bunun sebebiyse, LCD 1602 Modülünün ekran parlaklığının, 3. pininden aldığı voltaj miktarına bağlı şekilde değişmesiydi. Potansiyometre yardımıyla, voltaj miktarını ayarlama imkanı bulduk. Bu sayede, parlaklık miktarını isteğimize göre ayarlayabildik.

LCD 1602’nin 4. pinini (RS) Raspberry Pi’ın 22. pinine (GPIO25) ucuna bağladık. 1602 LCD Modülünde, kayıt seçimi (RS) sinyali, modül içindeki iki farklı kayıt arasında seçim yapmak için kullanılır: komut kaydı ve veri kaydı. Komut kaydı, mikrodenetleyiciden gelen talimatları almak için kullanılır, örneğin ekranı temizleme veya imleci ayarla komutları. Veri kaydı, ekran üzerine gösterilecek verileri göndermek için kullanılır, örneğin gösterilecek karakterler. RS sinyali kullanarak, mikrodenetleyici 1602 LCD Modülüne gerektiğinde komutlar veya veriler gönderebilir. Bu emirleri Raspberry Pi’ın 22. piniyle yapabilecek duruma geliyoruz.

LCD 1602’nin 5. pinini (RW), breadboardun pozitif hattına bağladık. 1602 LCD Modülünde, R/W (okuma/yazma) pini, iki farklı işletim modu arasında seçim yapmak için kullanılır: okuma modu ve yazma modu. Okuma modunda, mikrodenetleyici LCD Modülünden verileri okuyabilir, örneğin ekranın durumu veya imlecin konumu. Yazma modunda ise, mikrodenetleyici LCD Modülüne veriler yazabilir, örneğin ekranı temizleme veya gösterilecek karakterler komutları. R/W pini kullanarak, mikrodenetleyici gerektiğinde okuma ve yazma modları arasında geçiş yaparak 1602 lcd modülünü kontrol edebilir.

LCD 1602 Modülünün 6. pinini ise Raspberrry Pi’ın 18. pinine (GPIO24) bağladık. 1602 LCD Modülünde, E (etkinleştirme) pini, mikrodenetleyici ve LCD Modülü arasındaki veri akışını kontrol etmek için kullanılır. E pinin yüksek ayarlanmış olduğu durumda, LCD Modülü etkinleştirilir ve veri gönderilebilir veya alınabilir. E pinin düşük ayarlanmış olduğu durumda ise, LCD Modülü devre dışı bırakılır ve veri aktarılamaz. E pinini kullanarak, mikroişlemci verinin LCD Modülü tarafından ne zaman gönderileceğini veya alınacağını kontrol edebilir ve bu sayede istenen bilgilerin düzgün bir şekilde gösterilmesini sağlayabilir.

LCD 1602 Modülünün 11. pinini (DB4) doğrudan Rasbperry Pi’ın 16. pinine (GPIO23) bağladık. LCD 1602 Modülünün 12. pinini (DB5) doğrudan Raspberry Pi’ın 11. pinine (GPIO17) bağladık.LCD 1602 Modülünün 13. pinini (DB6) doğrudan Raspberry Pi’ın 12. pinine (GPIO18) bağladık. LCD 1602 Modülünün 14. pinini (DB7) doğrudan Raspberry Pi’ın 15. pinine (GPIO22) bağladık. DB0, DB1, DB2 ve DB3 pinleriniyse boş bıraktık. Çünkü, bizim verilerimiz 4 bitle rahatlıkla ifade edilebilir durumdaydı. 1602 LCD modülünde, veri yolu hat pini (D0-D7) mikrodenetleyici ve LCD Modülü arasında veri aktarımı için kullanılır. Bu pinler, 8 bitlik bir veri yolu oluşturur ve bu da bir defada 8 bit veri aktarabilecekleri anlamına gelir. Bu sayede mikrodenetleyici bir defada bir bayt veri gönderebilir veya alabilir, bu da karakterlerin gösterilmesi veya LCD Modülü üzerinde diğer işlemlerin yapılması için gereklidir. Veri yolu içindeki her bir pinin özel bir işlevi, 1602 LCD Modülünün özel yapılandırmasına göre değişebilir.

LCD 1602 Modülünün 15. pinini (A) breadbordun pozitif hattına bağladık. 1602 LCD Modülünde A pini, ekranın arka ışığını kontrol etmek için kullanılır. Arka ışık, lcd ekranın arkasındaki bir ışık kaynağıdır ve ekranı aydınlatır, bu sayede düşük ışık koşullarında da görülebilir hale gelir. A pin bir transistöre bağlıdır ve bu transistör arka ışığa giden gücü kontrol eder. A pini yüksek veya düşük ayarlayarak, mikrodenetleyici aracılığıyla arka ışık gerektiğinde açılabilir veya kapatılabilir.

LCD 1602 Modülünün 16. pinini (K) breadbordun pozitif hattına bağladık. 1602 LCD Modülünde K pini, arka ışığın parlaklığını kontrol etmek için kullanılır. K pin bir değişken dirençe bağlıdır ve bu değişken direnç arka ışık üzerinden akımı ayarlar. K pinindeki direnci değiştirerek, mikrodenetleyici arka ışığın parlaklığını gerektiğinde ayarlayabilir. Bu sayede kullanıcı ekranın parlaklığını tercihine göre ayarlayabilir ya da arka plan ışığını gerek duymadığında kısarak güçten tasarruf edebilir.

1602 LCD Modülündeki A ve K pinleri benzer ancak ayrı işlevleri vardır. A pini, arka ışığı açma veya kapama işlemi için kullanılırken, K pini ise arka ışığın parlaklığını ayarlamak için kullanılır. Başka bir deyişle, A pini arka ışığa giden gücü kontrol ederken, K pini arka ışık tarafından üretilen ışığın yoğunluğunu kontrol eder. A ve K pinlerini birlikte kullanarak, mikrodenetleyici LCD Modülünün arka ışığını optimum görülebilirlik ve güç verimliliği sağlamak için ayarlayabilir.

1. Yazılım Tasarımı

Diagram

Description automatically generated

Şekil 4.1 Yazılım Akış Diyagramı

import cv2 as cv

import numpy as np

import math

import Adafruit\_CharLCD as LCD

# Raspberry Pi pin configuration:

# Note this might need to be changed to 21 for older revision Pi's.

lcd\_rs        = 25

lcd\_en        = 24

lcd\_d4        = 23

lcd\_d5        = 17

lcd\_d6        = 18

lcd\_d7        = 22

lcd\_backlight = 4

lcd\_columns = 16

lcd\_rows = 2

# Define LCD column and row size for 16x2 LCD.

lcd\_columns = 16

lcd\_rows    = 2

lcd = LCD.Adafruit\_CharLCD(lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7,

                           lcd\_columns, lcd\_rows, lcd\_backlight)

camera = cv.VideoCapture(0)

frame\_count=0

frame\_rate=5

lower\_color = np.array([0, 50, 120], dtype=np.uint8)

upper\_color = np.array([180, 150, 250], dtype=np.uint8)

def get\_contours(binary\_img):

    contours,\_ = cv.findContours(binary\_img,cv.RETR\_TREE,cv.CHAIN\_APPROX\_SIMPLE)

    return contours

def get\_contour\_center(contour):

    M = cv.moments(contour)

    cx = -1

    cy = -1

    if M['m00'] != 0:

        cx = int(M['m10'] / M['m00'])

        cy = int(M['m01'] / M['m00'])

    return cx,cy

def draw\_hand(binary\_image,rgb\_image,contours,img\_no):

    black\_image = np.zeros([binary\_image.shape[0],binary\_image.shape[1],3],np.uint8)

    for c in contours:

        area = cv.contourArea(c)

        perimeter = cv.arcLength(c, True)

        ((x, y), radius) = cv.minEnclosingCircle(c)

        if (area > 130):

            cv.drawContours(black\_image, [c], -1, (150, 250, 150), 1)

            cx, cy = get\_contour\_center(c)

            cv.circle(black\_image, (cx, cy), (int)(radius), (0, 0, 255), 1)

            cv.circle(black\_image, (cx, cy), 5, (150, 150, 255), -1)

    cv.imshow('RGB Image Contours in image ', rgb\_image)

    cv.imshow('Black Image Contours ', black\_image)

def analyze\_defects(cnt,roi):

    epsilon = 0.0005 \* cv.arcLength(cnt, True)

    approx = cv.approxPolyDP(cnt, epsilon, True)

    hull = cv.convexHull(approx, returnPoints=False)

    defects = cv.convexityDefects(approx, hull)

    l = 0

    if defects is not None:

        for i in range(defects.shape[0]):

            s, e, f, d = defects[i, 0]

            start = tuple(approx[s][0])

            end = tuple(approx[e][0])

            far = tuple(approx[f][0])

            pt = (100, 180)

            a = math.sqrt((end[0] - start[0])\*\*2 + (end[1] - start[1])\*\*2)

            b = math.sqrt((far[0] - start[0])\*\*2 + (far[1] - start[1])\*\*2)

            c = math.sqrt((end[0] - far[0])\*\*2 + (end[1] - far[1])\*\*2)

            s = (a + b + c) / 2

            ar = math.sqrt(s \* (s - a) \* (s - b) \* (s - c))

            d = (2 \* ar) / a

            angle = math.acos((b\*\*2 + c\*\*2 - a\*\*2) / (2 \* b \* c)) \* 57

            if angle <= 90 and d > 30:

                l += 1

                cv.circle(roi, far, 3, [255, 0, 0], -1)

            cv.line(roi, start, end, [0, 255, 0], 2)

    return l

def analyze\_contours(frame,cnt,l,img\_no):

    hull = cv.convexHull(cnt)

    areahull = cv.contourArea(hull)

    areacnt = cv.contourArea(cnt)

    arearatio = ((areahull - areacnt) / areacnt) \* 100

    font = cv.FONT\_HERSHEY\_SIMPLEX

    if img\_no == 1:

        if l == 1:

            if areacnt < 2000:

                cv.putText(frame, 'Put hand in the box', (0, 50), font, 2,(0, 0, 255), 3, cv.LINE\_AA)

            else:

                if arearatio < 12:

                    cv.putText(frame, 'Stop', (0, 50), font, 2, (0, 0, 255), 3,cv.LINE\_AA)

                    lcd.clear()

                    lcd.message('Stop')

                else:

                    cv.putText(frame, 'Stop', (0, 50), font, 2, (0, 0, 255), 3,cv.LINE\_AA)

                    lcd.clear()

                    lcd.message('Stop')

        elif l == 4:

            cv.putText(frame, 'Prev', (0, 50), font, 2, (0, 0, 255), 3, cv.LINE\_AA)

            lcd.clear()

            lcd.message('Prev')

        elif l == 5:

            cv.putText(frame, 'Prev', (0, 50), font, 2, (0, 0, 255), 3, cv.LINE\_AA)

            lcd.clear()

            lcd.message('Prev')

        else:

            cv.putText(frame, 'reposition', (10, 50), font, 2, (0, 0, 255), 3,cv.LINE\_AA)

    elif img\_no == 2:

        if l == 1:

            if areacnt < 2000:

                cv.putText(frame, 'Put hand in the box', (800, 50), font, 2,(0, 0, 255), 3, cv.LINE\_AA)

            else:

                if arearatio < 12:

                    cv.putText(frame, 'Start', (800, 50), font, 2, (0, 0, 255), 3,cv.LINE\_AA)

                    lcd.clear()

                    lcd.message('Start')

                else:

                    cv.putText(frame, 'Start', (800, 50), font, 2, (0, 0, 255), 3,cv.LINE\_AA)

                    lcd.clear()

                    lcd.message('Start')

        elif l == 4:

            cv.putText(frame, 'Next', (800, 50), font, 2, (0, 0, 255), 3, cv.LINE\_AA)

            lcd.clear()

            lcd.message('Next')

        elif l == 5:

            cv.putText(frame, 'Next', (800, 50), font, 2, (0, 0, 255), 3, cv.LINE\_AA)

            lcd.clear()

            lcd.message('Next')

        else:

            cv.putText(frame, 'reposition', (800, 50), font, 2, (0, 0, 255), 3,cv.LINE\_AA)

    return frame

while True:

    \_,frame = camera.read()

    frame = cv.flip(frame,0)

    frame = cv.resize(frame,(1200,720)

    cv.rectangle(frame,(50, 50), (450, 450),(0,255,0),1)

    cv.rectangle(frame,(800, 50), (1150, 450),(0,255,0),1)

    crop\_img\_left = frame[50:450,50:450]

    crop\_img\_right = frame[50:450,800:1150]

    hsv1 = cv.cvtColor(crop\_img\_left,cv.COLOR\_RGB2HSV)

    mask1 = cv.inRange(hsv1,lower\_color,upper\_color)

    mask1 = cv.dilate(mask1,np.ones((3,3),np.uint8),iterations=3)

    mask1 = cv.erode(mask1,np.ones((3,3),np.uint8),iterations=3)

    mask1 = cv.GaussianBlur(mask1,(5,5),90)

    hsv2 = cv.cvtColor(crop\_img\_right,cv.COLOR\_RGB2HSV)

    mask2 = cv.inRange(hsv2,lower\_color,upper\_color)

    mask2 = cv.dilate(mask2,np.ones((3,3),np.uint8),iterations=3)

    mask2 = cv.erode(mask2,np.ones((3,3),np.uint8),iterations=3)

    mask2 = cv.GaussianBlur(mask2,(5,5),90)

    contours1 = get\_contours(mask1)

    contours2 = get\_contours(mask2)

    try:

        cnt1 = max(contours1,key=lambda x: cv.contourArea(x))

        l1 = analyze\_defects(cnt1,crop\_img\_left)

        frame = analyze\_contours(frame,cnt1,l1+1,1)

    except:

        pass

    try:

        cnt2 = max(contours2,key=lambda y: cv.contourArea(y))

        l2 = analyze\_defects(cnt2,crop\_img\_right)

        frame = analyze\_contours(frame,cnt2,l2+1,2)

    except:

        pass

    frame\_count+=1

    cv.imshow('Frame',frame)

    key = cv.waitKey(300)

    if key == ord('q'):

        break

cv.destroyAllWindows()

camera.release()

1. Sonuçlar, Demo Detayları ve Sunum Linki

Kamera üzerinden el hareketlerini tanıyan gömülü sistem görsellerde görüldüğü gibi çalışmaktadır. Kullanıcıdan sağ veya sol eli kapalı, açık olduğu durumlara göre yorumlanmış ve 1602 LCD üzerinde gösterilmiştir.

Projenin tanıtımının yapıldığı videoya <https://youtu.be/wfL-sVkdhFM> adresinden ulaşılabilir.

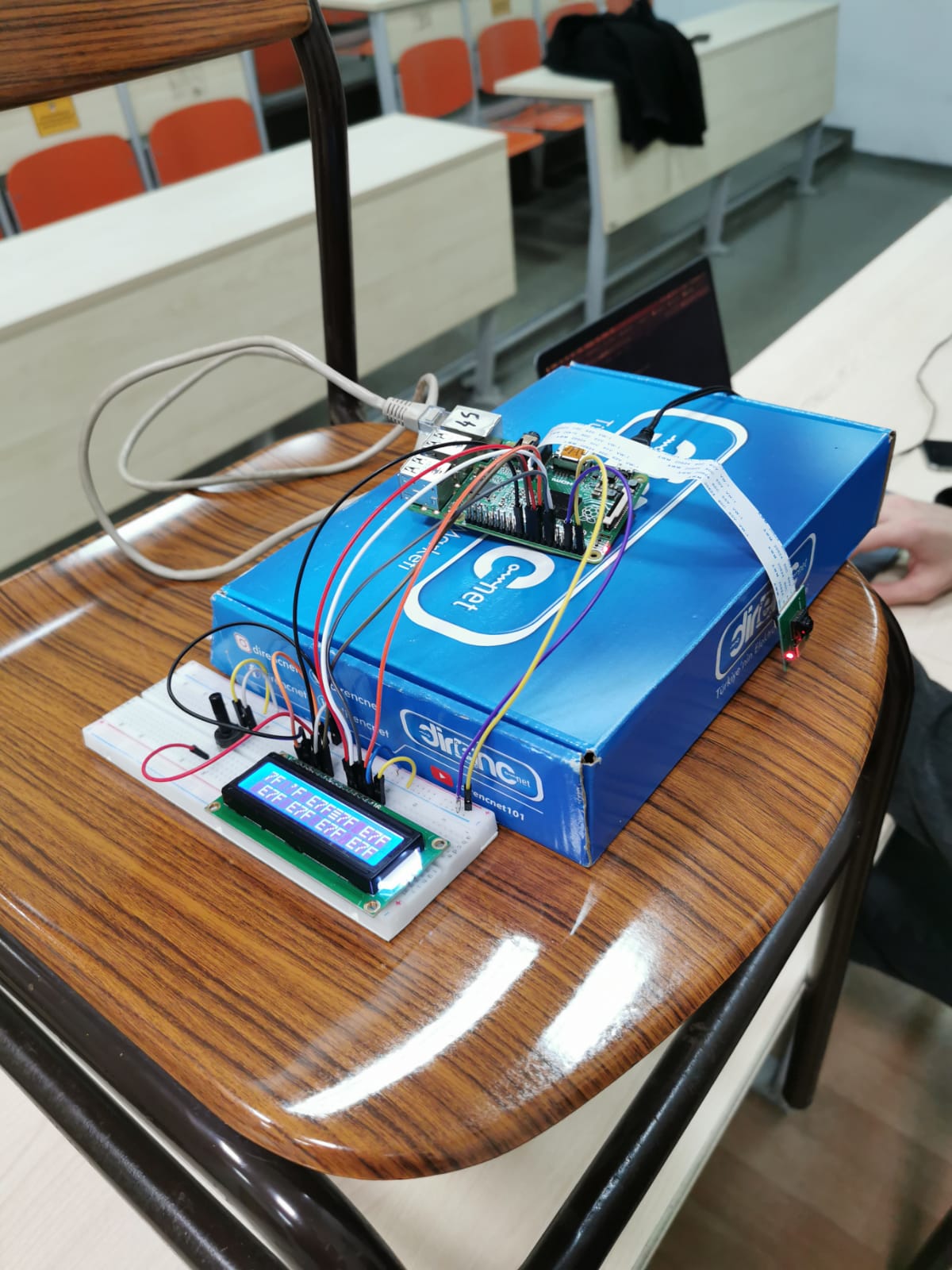


Görsel 1.1: Raspberry Pi üzerinde El hareketinin “Start” olarak tanınması.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

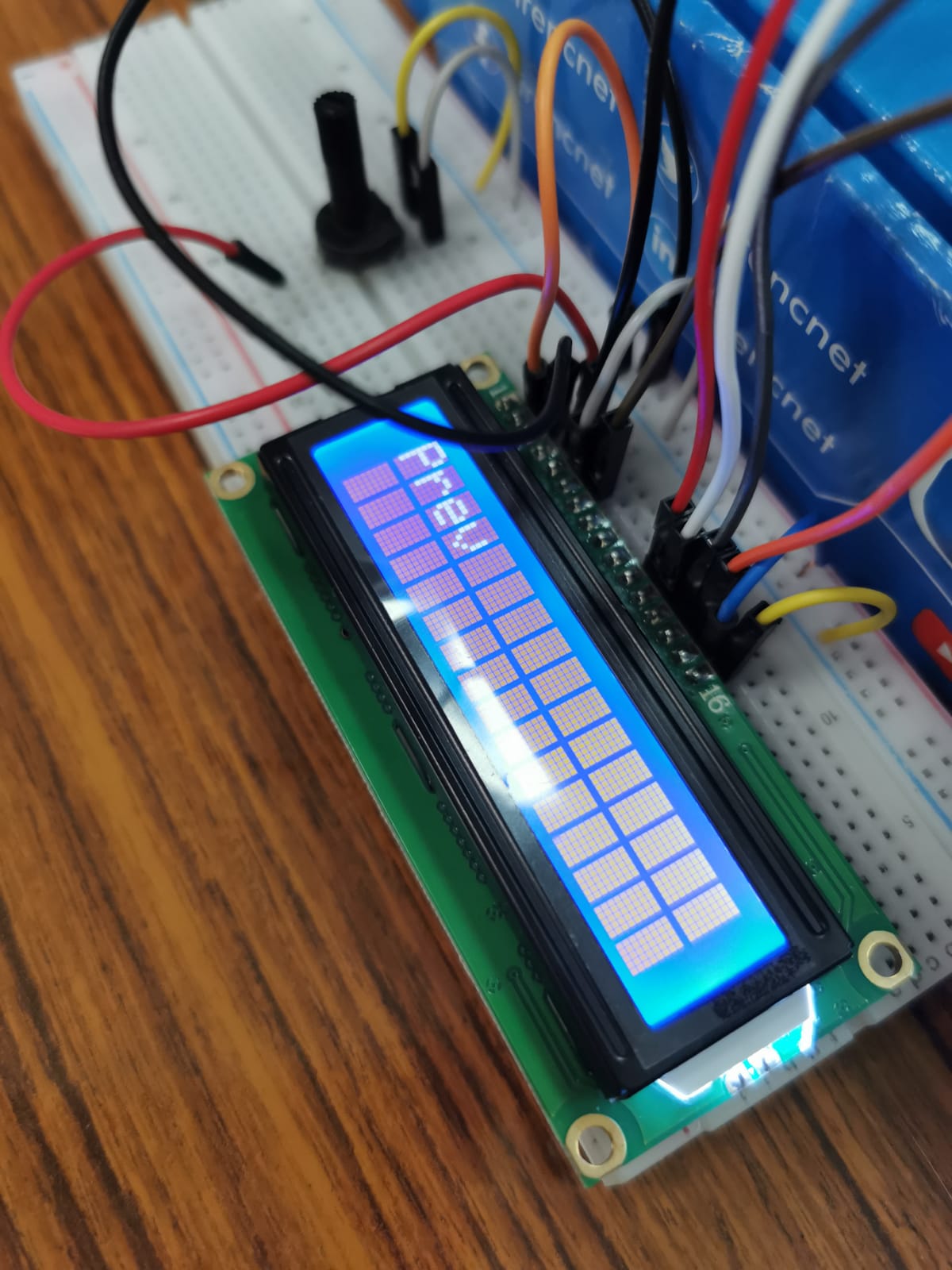
Görsel 1.2: Raspberry Pi üzerinde tanınan el hareketinin 1602 LCD üzerinde “Start” olarak yazılması.



Görsel 1.3: El komutlarını tanıyan sistem, kamera modülü ve 1602 LCD’yle birlikte.



Görsel 1.4: Kameradan alına görüntünün “Prev” komutu olarak tanınması.



Görsel 1.5: Raspberry Pi üzerinde “Prev” olarak tanınan komutun 1602 LCD üzerinde yazılması.

1. Referanslar

[1]“Getting Started with the Camera Module” projects.raspberrypi.org. https://projects.raspberrypi.org/en/projects/getting-started-with-picamera (Erişim Tarihi 27.11.2022)

[2]A.K. Yıldız “Raspberry Pi ile 2x16 Lcd Kullanımı” ahmetkemalyildiz.com.tr https://www.ahmetkemalyildiz.com.tr/raspberry-pi-ile-2x16-lcd-ekran-kullanimi/ (Erişim Tarihi 23.11.2022)

[3]“Contour Detection Using OpenCV (Python/C++)” learnopnecv.com https://learnopencv.com/contour-detection-using-opencv-python-c/ (Erişim Tarihi 26.11.2022)

[4]“OpenCV: Contours: Getting Started” docs.opencv.org https://docs.opencv.org/3.4/d4/d73/tutorial\_py\_contours\_begin.html (Erişim Tarihi 26.11.2022)

[5] “OpenCV Intro” docs.opencv.org https://docs.opencv.org/3.4/d1/dfb/intro.html (Erişim Tarihi 22.11.2022)

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-