T.C.

ADNAN MENDERES UNIVERSITESI

FACULTY OF ENGINEERING

COMPUTER ENGINEERING

DIGITAL IMAGE PROCESSING – PROJECT

211805018 - Köksal Kerem TANIL

Advisor: Dr. Öğr. Üyesi Mahmut SİNECEN

MATERIALS

1- **7.4V 2S Lipo Batarya-Pil 2800 mAh 35C**

- Voltaj: 7,4V
- Kapasite: 2800mAh
- 35C (50C anlık, maksimum 10sn)
- Ağırlık: 169gr
- Boyutları: 112x37x18mm
- 2 Hücreli

2- Arduino UNO R3 - SMD Klon (CH340 Çipli)

- Model: Arduino Uno R3 - SMD Klon(CH340 Çipli)
- Harici Besleme Gerilimi: 7-12V
- Toplam Pin Sayısı: 26
- Dijital Pin: 14
- Analog Pin: 6
- PWM Çıkış Sayısı: 6
- Her G/Ç için Akım: 40 mA
- 3.3V Çıkış için Akım: 50 mA
- Flash Hafıza: 32 KB (ATmega328) 0.5 KB kadarı bootloader
- SRAM: 2 KB (ATmega328)
- EEPROM: 1 KB (ATmega328)
- Clock Frekansı: 16 Mhz

3- Elektronik Breadboard - Büyük Boy - 830 Pin

4- 40lı Ayrılabilen Dişi-Dişi Jumper Kablo - 20cm - Arduino Uyumlu

5- 40lı Ayrılabilen Dişi-Erkek Jumper Kablo - 20cm - Arduino Uyumlu

6- 40lı Ayrılabilen Erkek-Erkek Jumper Kablo - 20cm- Arduino Uyumlu

7- Mini Breadboard - 170 Pin – Beyaz

8- HC05 Bluetooth Modülü Kartı - Arduino Uyumlu

- Bluetooth Versiyonu: Bluetooth 2.0+EDR(Enhanced Data Rate)
- Hafıza: Harici 8Mbit Flash
- Çıkış Gücü: -4 ~ +6dBm
- Çalışma Frekansı: 2.4GHz
- Hassasiyet: -80dBm
- Çalışma Gerilimi: 1.8 - 3.6V(Tipik: 3.3V)
- Çalışma Akımı(Max): 40mA
- Haberleşme Arayüzü: UART
- Çalışma Alanı: 10m
- Boyutlar(Modül): 27x13x2mm

- Boyutlar(Modül Kartı): 43x16x7mm

9- 6V 250 Rpm DC Motor Ve Tekerlek Seti

10- 4WD Araba şasesi

11- L298N Çift Motor Sürücü Kartı - Voltaj Regülatörlü

- Model: L298N Çift Motor Sürücü Kartı
- Çip: L298N
- Sürme Voltajı: 5V-35V
- Sürme Akımı: 2A (Max Motor başına)
- Lojik Voltajı: 5V
- Lojik Akımı: 0mA - 36mA
- Max Çıkış Gücü: 25W
- Ağırlık: 30gr
- Boyut: 43x43x27mm

CODES

```
import cv2
import numpy as np
import time
from collections import deque
import serial
```

Importing the Required Libraries

```
arduino = serial.Serial('COM3', 9600)
time.sleep(2)
```

Establishing Serial Communication with Arduino

```
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Kamera açılamadı!")
    exit()
```

Launching the Camera

```
finger_buffer = deque(maxlen=5)
last_command_time = time.time()
command_cooldown = 5
previous_finger_count = 0
```

Stabilization and Timing Variables

```
def detect_fingers(contour):
    fingers = 0
    hull = cv2.convexHull(contour, returnPoints=False)
    if len(hull) > 3:
        defects = cv2.convexityDefects(contour, hull)
        if defects is not None:
            for i in range(defects.shape[0]):
                s, e, f, d = defects[i, 0]
                start = tuple(contour[s][0])
                end = tuple(contour[e][0])
                far = tuple(contour[f][0])

                a = np.linalg.norm(np.array(end) - np.array(start))
                b = np.linalg.norm(np.array(far) - np.array(start))
                c = np.linalg.norm(np.array(far) - np.array(end))

                angle = np.degrees(np.arccos((b**2 + c**2 - a**2) / (2 * b * c)))
                if angle <= 90 and d > 1000:
                    fingers += 1
```

Function of Detecting the Number of Fingers

```
def is_fist(contour):
    area = cv2.contourArea(contour)
    perimeter = cv2.arcLength(contour, True)
    hull = cv2.convexHull(contour)
    hull_area = cv2.contourArea(hull)

    if perimeter == 0 or hull_area == 0:
        return False

    circularity = 4 * np.pi * (area / (perimeter ** 2))
    solidity = float(area) / hull_area

    return circularity > 0.4 and solidity > 0.9 and area < 15000
```

Fist Detection Function

```
def stabilize_finger_count(finger_count):
    finger_buffer.append(finger_count)
    return int(np.mean(finger_buffer))
```

Stabilization Function

```
while True:
    ret, frame = cap.read()
    if not ret:
        print("Görüntü alınamadı!")
        break
```

gets a picture from the camera

```
    frame = cv2.flip(frame, 1)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
```

Converts the image from BGR color space to HSV color space

```
    lower_skin = np.array([0, 30, 60], dtype=np.uint8)
    upper_skin = np.array([20, 150, 255], dtype=np.uint8)
    skin_mask = cv2.inRange(hsv, lower_skin, upper_skin)
```

Setting color limits

```
    kernel = np.ones((5, 5), np.uint8)
    skin_mask = cv2.erode(skin_mask, kernel, iterations=1)
    skin_mask = cv2.dilate(skin_mask, kernel, iterations=2)
    skin_mask = cv2.GaussianBlur(skin_mask, (5, 5), 0)
```

Erosion, dilatation and blurring processes

```
    height, width = frame.shape[:2]
    skin_mask[:int(height / 3), :] = 0
```

Not detecting the face

```
 contours, _ = cv2.findContours(skin_mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
 largest_contour = max(contours, key=cv2.contourArea, default=None)
```

finding the contours

```
finger_count = 0
if largest_contour is not None:
    area = cv2.contourArea(largest_contour)
    if area > 2000:
        if is_fist(largest_contour):
            finger_count = 0
        else:
            finger_count = detect_fingers(largest_contour)
```

Calculating the Area of the Contour and the Number of Fingers

```
hull = cv2.convexHull(largest_contour)
cv2.drawContours(frame, [hull], -1, (0, 255, 0), 2)
```

Drawing the Contour and Contours

```
stabilized_finger_count = stabilize_finger_count(finger_count)
```

Number of Stabilized Fingers Calculation

```
cv2.putText(frame, f'Parmak Sayisi: {stabilized_finger_count}', (10, 50),
            cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)
```

Printing the Number of Fingers on the Image

```
if time.time() - last_command_time > command_cooldown:
    if stabilized_finger_count != previous_finger_count:
        if stabilized_finger_count == 0:
            print("DUR")
            arduino.write(b'0\n')
        elif stabilized_finger_count == 1:
            print("GERİ GİT")
            arduino.write(b'1\n')
        elif stabilized_finger_count == 2:
            print("SAĞA GİT")
            arduino.write(b'2\n')
        elif stabilized_finger_count == 3:
            print("SOLA GİT")
            arduino.write(b'3\n')
        elif stabilized_finger_count in [4, 5]:
            print("İLERİ GİT")
            arduino.write(b'5\n')

        last_command_time = time.time()
        previous_finger_count = stabilized_finger_count
```

Sending Commands to Arduino

```
cv2.imshow("El Algilama", frame)

if cv2.waitKey(1) & 0xFF == ord('q'):
    break
```

Show the Image and Exit the Program

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

Releasing Resources

Challenges and Solutions

1. Orange Light Issue

Challenge: The orange lights in my house made it difficult for the camera to detect my hand accurately. The light color closely resembled skin tones, leading to significant errors in the image processing step.

Solution: I installed white lights in my house to improve the lighting conditions. White light enabled more accurate detection of skin tones, significantly improving the image processing results.

2. Face Detection Problem

Challenge: While trying to detect my hand, the camera sometimes identified my face as a contour. This negatively impacted the accuracy of finger detection.

Solution: I disabled contour detection for the top third of the camera frame, ensuring that only contours at hand level were detected, ignoring the face entirely.

3. Missing Screws for the Chassis

Challenge: The chassis provided for the hardware assembly did not include screws, making it impossible to secure the motors.

Solution: I went to an industrial area to find screws and other missing components compatible with the chassis. This allowed me to assemble the hardware properly.

4. Using Double-Sided Tape

Challenge: While waiting to find suitable screws to attach the motors to the chassis, I needed a temporary solution.

Solution: I used double-sided tape to temporarily secure the motors to the chassis. This enabled me to proceed with testing phases until the proper screws were obtained.

Limitations

1. Ineffectiveness with Short-Sleeved Clothing

The system does not work effectively when short-sleeved clothing is worn, as it may confuse the arm with the hand during contour detection.

2. Instability with Different Skin Tones

The algorithm struggles to deliver stable results across various skin tones, which could lead to inaccuracies in finger detection.

3. Incompatibility with Gloves or Accessories

The system does not produce accurate results when gloves or other hand accessories are worn, as they alter the hand's shape and color.

4. Distance Sensitivity

Due to the limitations of the camera, the accuracy decreases as the hand moves farther away from the camera.

5. Single-Hand Detection

The system is designed to work with only one hand and cannot handle scenarios where both hands are used simultaneously.

6. Background Dependency

The color of the clothing worn and the background plays a significant role. If the background or clothing color is similar to skin tones, detection accuracy drops.

7. Pre-Calibration for Positioning

Before running the system with Arduino, it is recommended to run the code without Arduino to establish the initial position and calibration, ensuring smoother operation afterward.

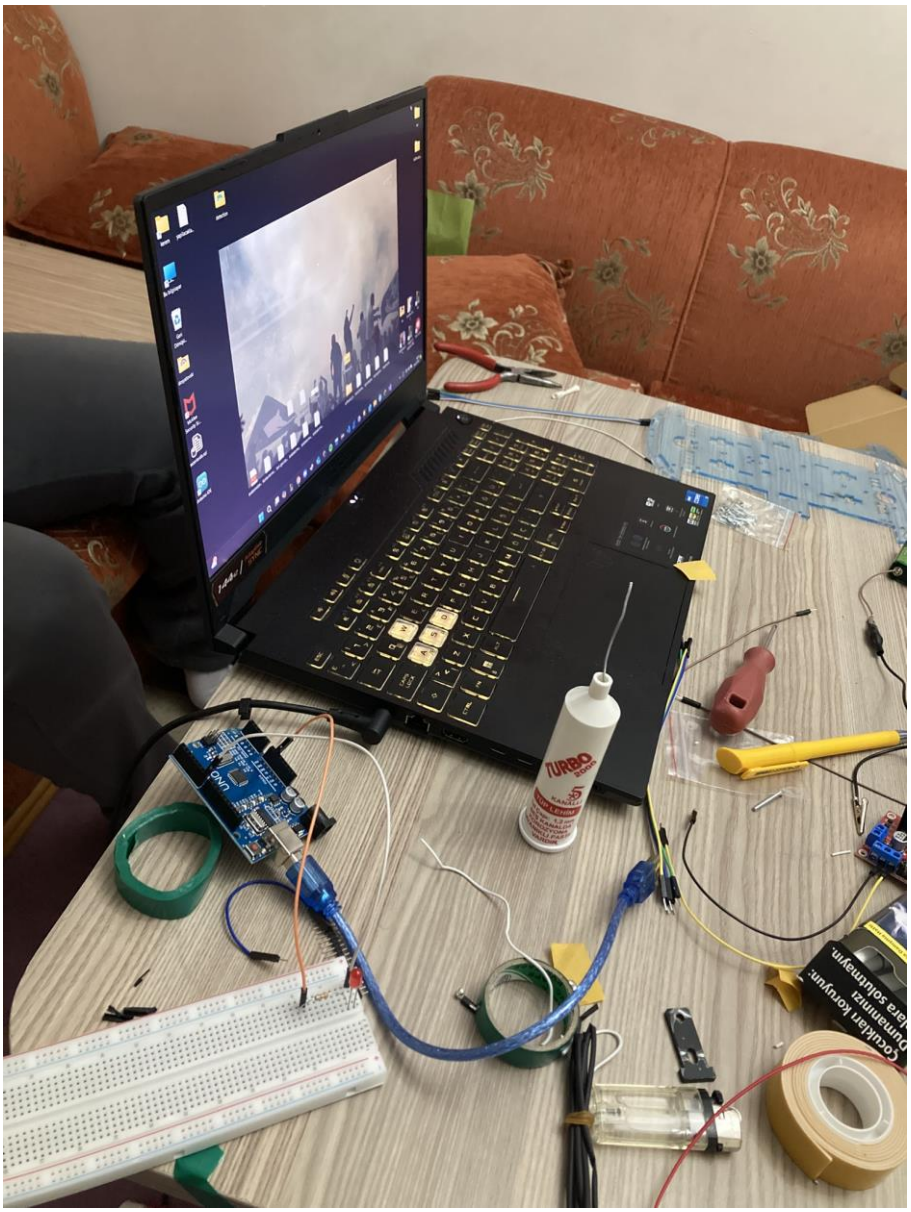
8. Fixed Camera Position Required

The camera must remain stationary for the system to function correctly. Movement or shifting of the camera compromises detection accuracy.

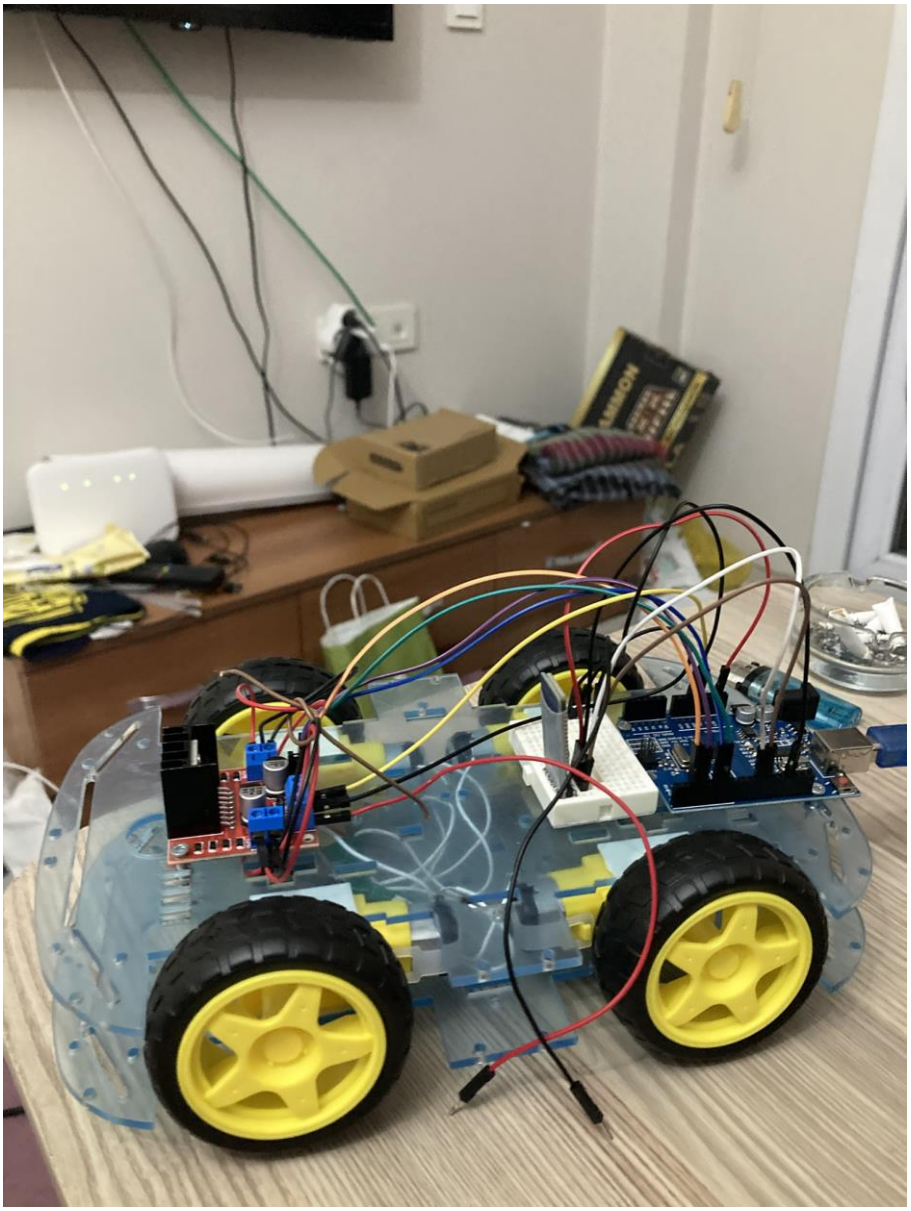
9. Works with Specific Finger Movements

The system is optimized to work with certain specific finger movements and may not respond accurately to other hand gestures or random finger positions.

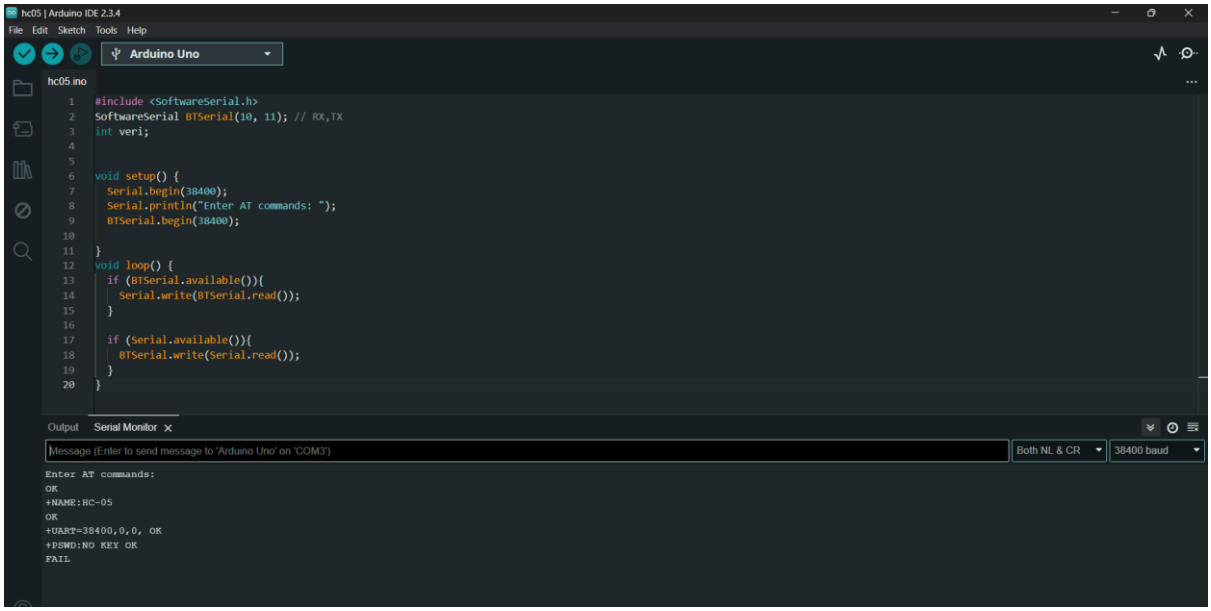
Photos and Videos



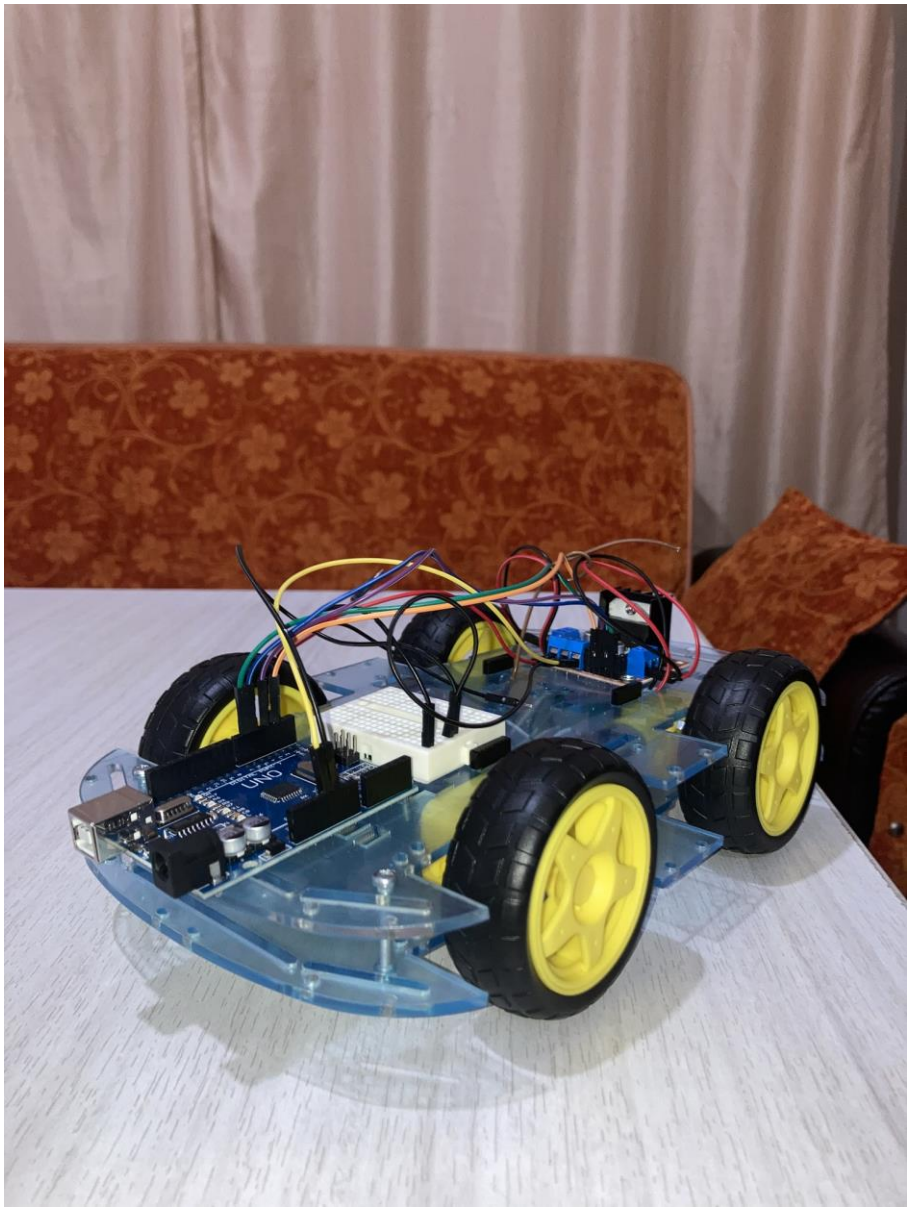

robo90.com



https://youtube.com/shorts/wJF8DTcW8Yw - **Bluetooth Connections**



https://youtube.com/shorts/VBzrtugx6l0 - **Bluetooth Paired**



https://youtube.com/shorts/ZWpGjaC2EpU - **Software testing**

https://youtube.com/shorts/dAbZIFf_iVA - **Car Movements – Test**

```
GERİ GİT
DUR
SAĞA GİT
DUR
SOLA GİT
SAĞA GİT
DUR
İLERİ GİT
DUR
```

https://youtu.be/vu64colVOqY - Testing for the last time in the classroom

REFERENCES

https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

https://stackoverflow.com/

https://www.geeksforgeeks.org/opencv-python-tutorial/