



SZEGEDI TUDOMÁNYEGYETEM
UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
MÉRNÖKI KAR

***VALÓS IDEJŰ KÉPFELDOLGOZÁSSAL TÁMOGATOTT
ÖNVEZETŐ RC AUTÓ MEGVALÓSÍTÁSA***

TDK DOLGOZAT

Jernei Mihály

mechatronikai mérnök MSc

hallgató

Témavezetők:

Dr. Sarcevic Péter

főiskolai docens

Kóczi Dávid

tanársegéd

SZEGED

2024

IRODALOMJEGYZÉK

- Anderson, J. M., Kalra, N., Stanley, K. D., Sorensen, P., Samaras, C., & Oluwatola, O. A. (2016). *Autonomous Vehicle Technology: A Guide for Policymakers*. Santa Monica: RAND Corporation. (22, 30-32, 83-90, 125-128)
- Dong, G., Tang, M., Yan, R., Mu, Z., Cai, L., & Park, B. B. (2023). Deep learning for autonomous vehicles and systems. In I. K. Sethi (Ed.), *Autonomous Vehicles and Systems* (9-47). New York: River Publishers. DOI:10.1201/9781032629537-2. (3-7, 9-10)
- Campbell, S., O'Mahony, N., Krpalcova, L., Riordan, D., Walsh, J., Murphy, A., & Ryan, C. (2018). Sensor technology in autonomous vehicles: A review. *2018 29th Irish Signals and Systems Conference (ISSC)*. DOI: 10.1109/ISSC.2018.8585340. (1-4)
- Szeliski, R. (2022). *Computer Vision: Algorithms and Applications* (2nd ed.). Springer. (396, 408-411, 822-824)
- Fan, R., Guo, S., & Bocus, M. J. (Eds.). (2023). *Autonomous driving perception: Fundamentals and applications*. Advances in Computer Vision and Pattern Recognition. Springer. <https://doi.org/10.1007/978-981-99-4287-9> (72-73, 79-81, 110-115)
- Zhang, X., Li, J., Li, Z., Liu, H., Zhou, M., Wang, L., & Zou, Z. (2023). *Multi-sensor fusion for autonomous driving*. Springer. <https://doi.org/10.1007/978-981-99-3280-1> (20-23)
- Murphey, Y. L., Kolmanovsky, I., & Watta, P. (Eds.). (2023). *AI-enabled technologies for autonomous and connected vehicles*. Lecture Notes in Intelligent Transportation and Infrastructure. Springer. <https://doi.org/10.1007/978-3-031-06780-8> (14-15, 95-100, 165-168)
- Du, M. (2023). *Autonomous vehicle technology: Global exploration and Chinese practice*. Unmanned System Technologies. Springer. <https://doi.org/10.1007/978-981-19-4143-6> (74, 159-160)

- Bimbraw, K. (2015). Autonomous cars: Past, present and future - A review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. In *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO* (191-198). Colmar, Alsace, France. <https://doi.org/10.5220/0005540501910198> (1-9)
- Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., et al. (2006). Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9), 661–692. <https://doi.org/10.1002/rob.20147> (1-32)
- Maanyu, K. N., Raj, D. G., Krishna, R. V., & Choubey, S. B. (2020). A study on Tesla Autopilot. *International Journal of Scientific Progress and Research (IJSPR)*, 71(1), Issue 171, (1-6)
- Ng, T. S. (2021). *Robotic vehicles: Systems and technology*. Springer. <https://doi.org/10.1007/978-981-33-6687-9> (78-81, 104-106)
- Wong, K., & Gu, Y. (2020). Mapping for autonomous driving: Opportunities and challenges. *IEEE Intelligent Transportation Systems Magazine*, Volume: 13, Issue 1. <https://doi.org/10.1109/MITS.2020.3014152> (3, 9)
- Charroud, A., El Moutaouakil, K., Palade, V., Yahyaouy, A., Onyekpe, U., & Eyo, E. U. (2024). Localization and mapping for self-driving vehicles: A survey. *Machines*, 12(2), 118. <https://doi.org/10.3390/machines12020118> (15-19)
- Kamijo, S., Gu, Y., & Hsu, L.-T. (2015). Autonomous vehicle technologies: Localization and mapping. *IEICE Fundamentals Review*, 9(2), 131–141. https://doi.org/10.1587/essfr.9.2_131 (2-4)
- Scaramuzza, D., & Fraundorfer, F. (2011). Visual odometry [Tutorial]. *IEEE Robotics & Automation Magazine*, 18(4), 80–92. <https://doi.org/10.1109/MRA.2011.943233> (1-4)
- Sapkal, A., Arti, Dishant, Pawar, D., & Singh, P. (2023). Lane detection techniques for self-driving vehicles: Comprehensive review. *Multimedia Tools and Applications*, 82, 33983–34004. <https://doi.org/10.1007/s11042-023-14446-6> (1-10)

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2021). Graph neural networks: A review of methods and applications. *AI Open, Volume 1*, 57–81. <https://doi.org/10.1016/j.aiopen.2021.01.001> (1-5)

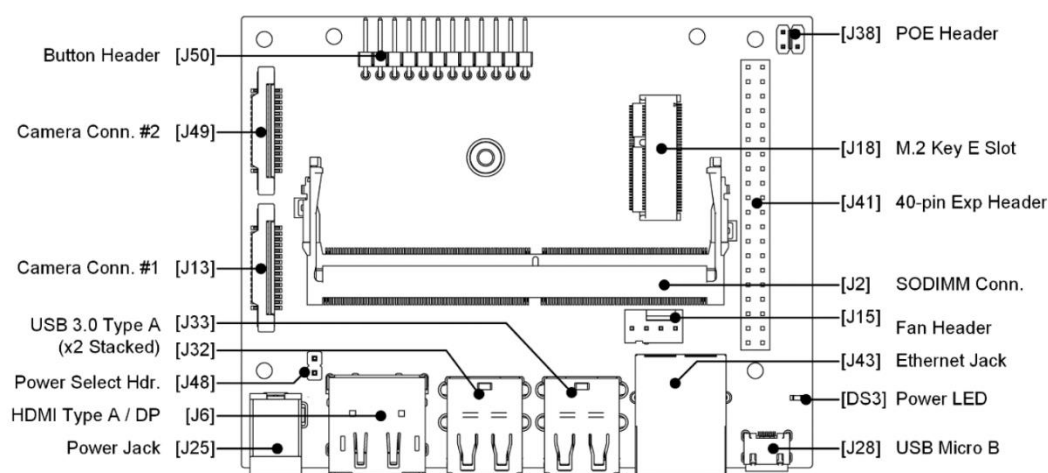
ÁBRAJEGYZÉK

1. **ábra:** Önvezető autó felépítési ábrája
(<https://www.nytimes.com/interactive/2016/12/14/technology/how-self-driving-cars-work.html>, 2024.08.08)
2. **ábra:** Houdina Radio Control rádióvezérelt autója (<https://www.discovermagazine.com/technology/the-driverless-car-era-began-more-than-90-years-ago>, 2024.08.08)
3. **ábra:** Az Amerikát átutazó NavLab 5-ös számú autonóm járműve
(https://www.researchgate.net/figure/Snapshot-of-the-demonstration-of-automated-highway-system-technologies-by-the-National_fig1_238597909, 2024.08.08)
4. **ábra:** Az Amerikát átutazó NavLab 5-ös számú autonóm járműve
(https://www.researchgate.net/figure/Navlab-5-vehicle-by-Carnegie-Mellon_fig4_338412717, 2024.08.08)
5. **ábra:** A 2004-es DARPA győztes "Stanley" autója (https://www.si.edu/object/nmah_1377824, 2024.08.08)
6. **ábra:** Waymo önvezető autó 2017-ben és napjainkban
(<https://europe.autonews.com/article/20180508/ANE/180509829/in-self-driving-car-race-waymo-leads-traditional-automakers>, <https://www.selfdrivingcars360.com/waymo-takes-testing-to-the-next-level-no-human-even-in-the-drivers-seat/>, 2024.08.09)
7. **ábra:** A Tesla teljes önvezető rendszerének béta verziója által modellezett környezet
(<https://www.autoevolution.com/news/tesla-removes-radar-vision-from-autopilot-and-fsd-for-model-3-and-model-y-in-europe-185830.html#>, 2024.08.09)
8. **ábra:** Autonóm autók exteroceptív szenzorrendszere (<https://www.linkedin.com/pulse/autonomous-vehicle-sensor-market-worth-191-billion-2030-shu-cho-50qjf/>, 2024.08.10)
9. **ábra:** Valós városi környezet leképezése nagy felbontású térképpel
(<https://www.autonomousvehicleinternational.com/features/the-road-to-everywhere-are-hd-maps-for-autonomous-driving-sustainable.html>, 2024.08.11)
10. **ábra:** SLAM technikával leképezett háromdimenziós térkép
(<https://medium.com/@krish.bhoopati556/beyond-the-steering-wheel-a-deep-dive-into-autonomous-vehicle-technology-55e4458bf85b>, 2024.08.16)
11. **ábra:** YOLO modell eredménye egy közlekedési kép esetén (<https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31>, 2024.08.17)
12. **ábra:** Szegmentált utcakép pontos objektumhatárokkal (https://www.researchgate.net/figure/Semantic-segmentation-of-a-scene-from-the-Cityscapes-dataset-by-Cordts-et-al-2016_fig24_316270100, 2024.08.17)
13. **ábra:** Sztereó látás epipoláris geometriája (https://www.researchgate.net/figure/Figure-2-The-epipolar-geometry-of-stereoscopic-vision_fig2_331960044, 2024.08.18)
14. Monokuláris mélységbecslés (<https://vladlen.info/publications/towards-robust-monocular-depth-estimation-mixing-datasets-zero-shot-cross-dataset-transfer/>, 2024.08.18)
15. **ábra:** Vizuális odometria lépéseinek szemléltetése (https://www.researchgate.net/figure/The-illustration-of-our-visual-odometry-framework-The-initial-state-vti-of-current_fig2_332103736, 2024.08.18)
16. **ábra:** Objektumkövetés 3D pontfelhőben (<https://www.kitware.com/3d-multi-object-tracking-using-lidar-for-autonomous-driving/>, 2024.08.18)
17. **ábra:** A sávérzékelő rendszerek működésének alapvető lépései
(https://www.researchgate.net/figure/General-architecture-of-lane-detection-system_fig1_369144658, 2024.08.18)
18. **ábra:** Autonóm jármű vezérlési lépései (https://link.springer.com/chapter/10.1007/978-3-031-06780-8_1/figures/1, 2024.08.10)
19. **ábra:** CNN működési ábrája (<https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/>, 2024.08.19)
20. **ábra:** GNN architektúrája (<https://www.datacamp.com/tutorial/comprehensive-introduction-graph-neural-networks-gnns-tutorial>, 2024.08.19)
21. **ábra:** Q-learning és a DQN módszer összehasonlítása (<https://www.baeldung.com/cs/q-learning-vs-deep-q-learning-vs-deep-q-network>, 2024.08.19)
22. **ábra:** RNN alapvető felépítése és a kihajtogatott változata (Deep Learning for Autonomous Vehicles and Systems 5.4 fig)
23. **ábra:** A fejlesztési projekt időterve és célkitűzései
24. **ábra:** "Reely FS53113CEI" típusú modellautó váz (<https://www.inshop.hu/reely-onroad-vaz-1-10-rc-modellauto-elektro-kozuti-modell-4wd-arr>, 2024.08.23)

25. **ábra:** "GPX Extreme SWIFT 550 7,2 V" típusú szénkefés motor (<https://jakomodel.hu/szenkefes-elektromotor-gpx-extreme-swift-550-72-v-detail>, 2024.08.23)
26. **ábra:** "Power HD LW-13MG" típusú kormány szervó (<https://jakomodel.hu/powerhd-lw-13mg-vizallo-digitalis-szervo-13kg-cm?search=PowerHD&description=true>, 2023.08.23)
27. **ábra:** "SURPASS ROCKET ESC 1/10 RC 60A 12V" típusú szabályzó (<https://jakomodel.hu/surpass-rocket-esc-szenkefes-motorhoz-1-10-rc-crawler-60a-12v-detail>, 2024.08.23)
28. **ábra:** "CNHL 2200mAh 7,4V 30C" típusú LiPo akkumulátor (<https://chinahobbyline.com/products/cnhl-black-series-2200mah-7-4v-2s-30c-lipo-battery-with-xt60>, 2024.08.23)
29. **ábra:** RC autó váz összeszerelt vezérlőegységgel
30. **ábra:** NVIDIA Jetson Nano B01 Development Kit (<https://www.cytronetech.vn/p-nvidia-jetson-nano-development-kit-revb01>, 2024.08.23)
31. **ábra:** "PCA9685" típusú 16 csatornás szervó vezérlő (https://www.hestore.hu/prod_10046486.html?gross_price_view=1&source=gads&gad_source=1&gclid=CjwKCAjwiaa2BhAiEiwAQBgYHq3BLirM-hbr1BFsLpCQg8DRwYG3pz1QLB6u4BO41r72iMclicR5LBoC2BQQAuD_BwE, 2024.08.24)
32. **ábra:** Indikátor áramkör
33. **ábra:** Raspberry Pi Camera Module v2 (<https://thepihut.com/products/raspberry-pi-camera-module>, 2024.08.24)
34. **ábra:** Vipfan F09 20000mAh Powerbank (https://kulsoaksi.hu/vipfan-f09-20000mah-powerbank-225w-pd-fekete-33479?utm_source=google_shopping&utm_medium=cpp&utm_campaign=direct_link&gad_source=1&gclid=CjwKCAiAt5euBhB9EiwAdkXWO6zb3s95ifVihq-Fi_hKmbveiUJA4TIFrOQKKV5Xev2KOF56N5U7hhoC3ZUQAuD_BwE, 2024.08.24)
35. **ábra:** A tervezett tartószerkezet és a befoglaló méretek
36. **ábra:** Az önvezető modellautó rendszerét leíró blokkdiagram
37. **ábra:** Az tervek alapján kivitelezett hardver
38. **ábra:** "start_button_v2.py" program folyamatábrája
39. **ábra:** "car_control".py folyamatábrája
40. **ábra:** "training_data_collect.py" programot leíró folyamatábra
41. **ábra:** "capture_and_save_image()" függvény folyamatábrája
42. **ábra:** Példa egy nyers adatpárosra – Pillanatkép és a hozzá tartozó vezérlő értékek
43. **ábra:** "onnx_model_vezerlo.py" működési folyamatábrája
44. **ábra:** "tanito_adatok_elokeszítése.py" program működési folyamatábrája
45. **ábra:** Példa a rendezett és véletlenszerűen összekevert és feldolgozott tanító adatokból
46. **ábra:** Az autó által használt egy kimenetű CNN modell struktúrája
47. **ábra:** A "neuralis_halo_tanitas.py" program folyamatábrája
48. **ábra:** 1-es tesztpálya
49. **ábra:** Ideális és zajos adatok felvételének elvi ábrája az első tesztpályán
50. **ábra:** "modell_2" tanulási görbéje
51. **ábra:** Túllilleszkedés jelensége a 6. kísérletnél
52. **ábra:** A 11. modell tanulási görbéje
53. **ábra:** Az áttörésnél betanított modell tanulási görbéje
54. **ábra:** Tanulási görbe a 23-as kísérletnél
55. **ábra:** A legsikeresebb modell tanulási görbéje
56. **ábra:** A legnagyobb adatbázissal tanított modell tanulási folyamata
57. **ábra:** A két kimenettel rendelkező modell tanulási görbéje
58. **ábra:** Python-OpenCV segítségével feldolgozott videó eredménye
59. **ábra:** Az eredeti pálya és az autó által megtett út ábrázolása közös koordináta-rendszerben – Jobb menetirány – 3.kör
60. **ábra:** A számított távolságok a bal menetirány 3. körében
61. **ábra:** A pályakövetési hiba számított értékei
62. **ábra:** A 2. tesztpálya kialakítása
63. **ábra:** A második tesztpálya első CNN modelljeinek tanulási görbéi
64. **ábra:** A második tesztpálya sikeres modelljeinek tanulási görbéi

MELLÉKLETEK

1. Melléklet: NVIDIA Jetson Nano B01 Development Kit felépítése



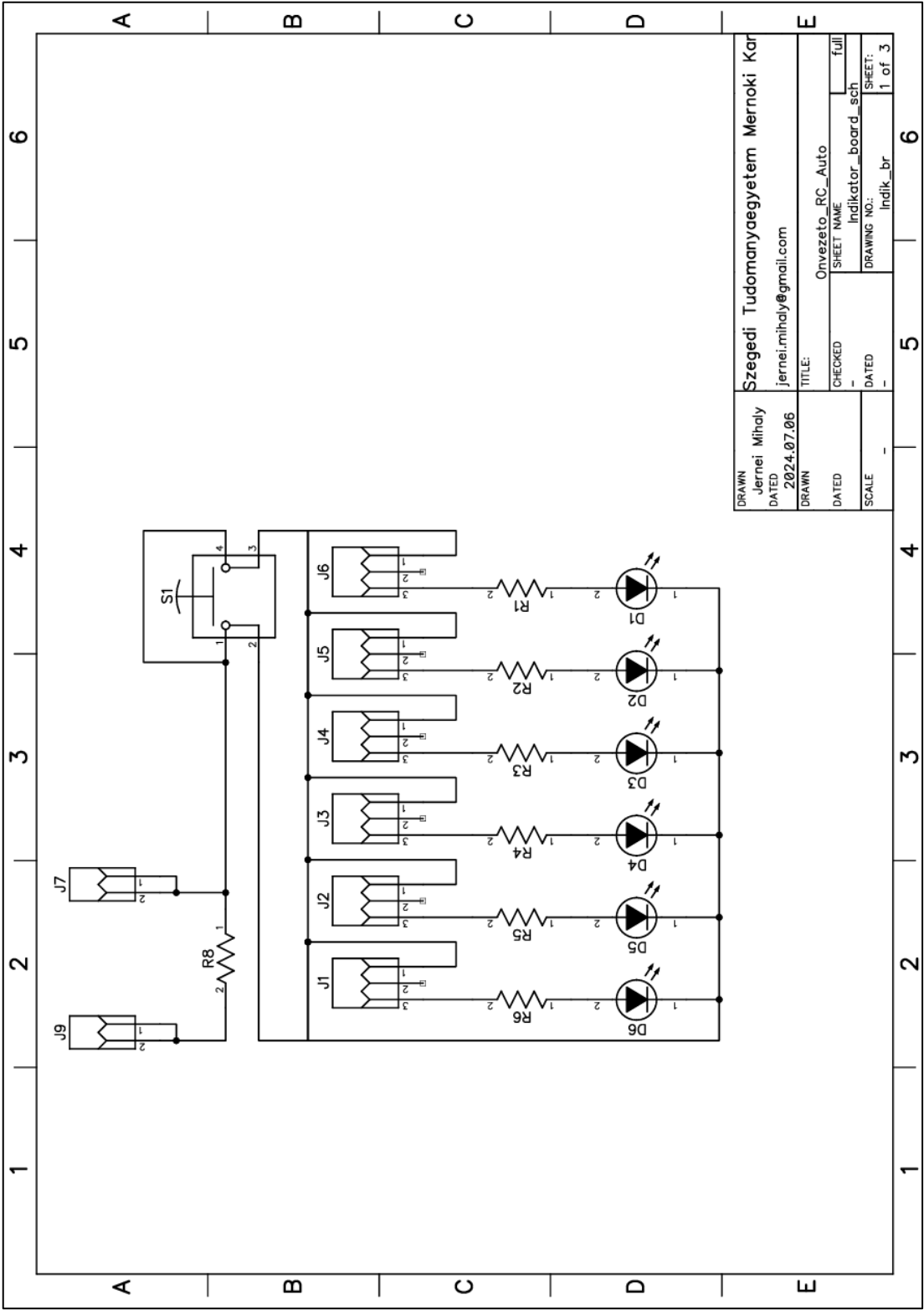
A Jetson Nano nagy teljesítményét a következő elemei teszik lehetővé:

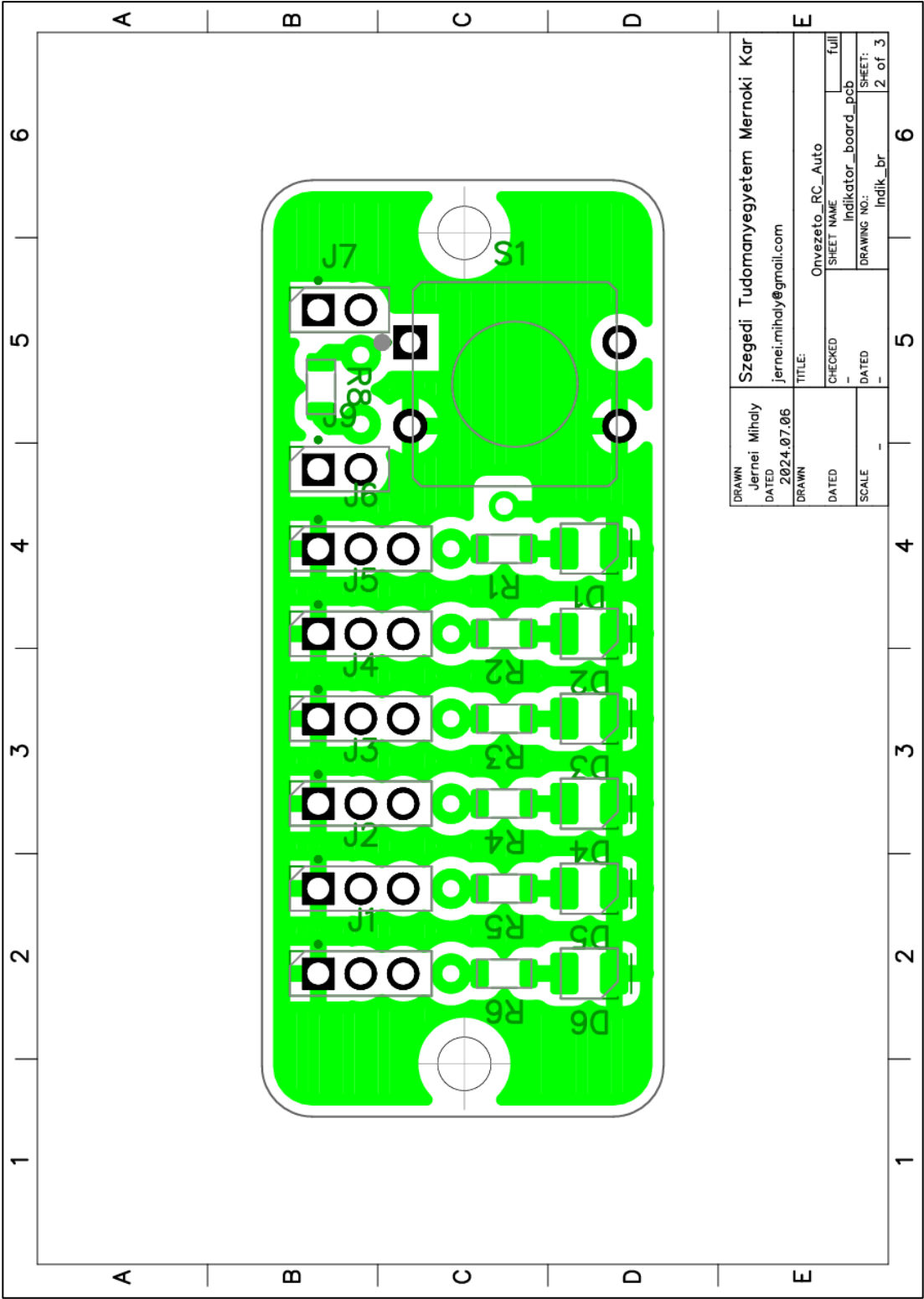
- 1,43 GHz órajelű Quad-core ARM Cortex-A57 processzor,
- 128 CUDA magos GPU,
- 4GB LPDDR4 -es memória.
- 2 db MIPI CSI kamerainterfész
- I/O csatlakozások (elhelyezkedésüket az **1. melléklet** szemlélteti):
 - 4 db USB 3.0 Type-A port,
 - 1 db USB 2.0 Micro-B csatlakozó,
 - 1 db HDMI 2.0 és DisplayPort 1.2 videókimenet,
 - RJ45 -ös Gigabit Ethernet port,
 - 40 db GPIO pin UART, SPI, I²C támogatással (a kiosztás a **2. melléklet**-ben található).

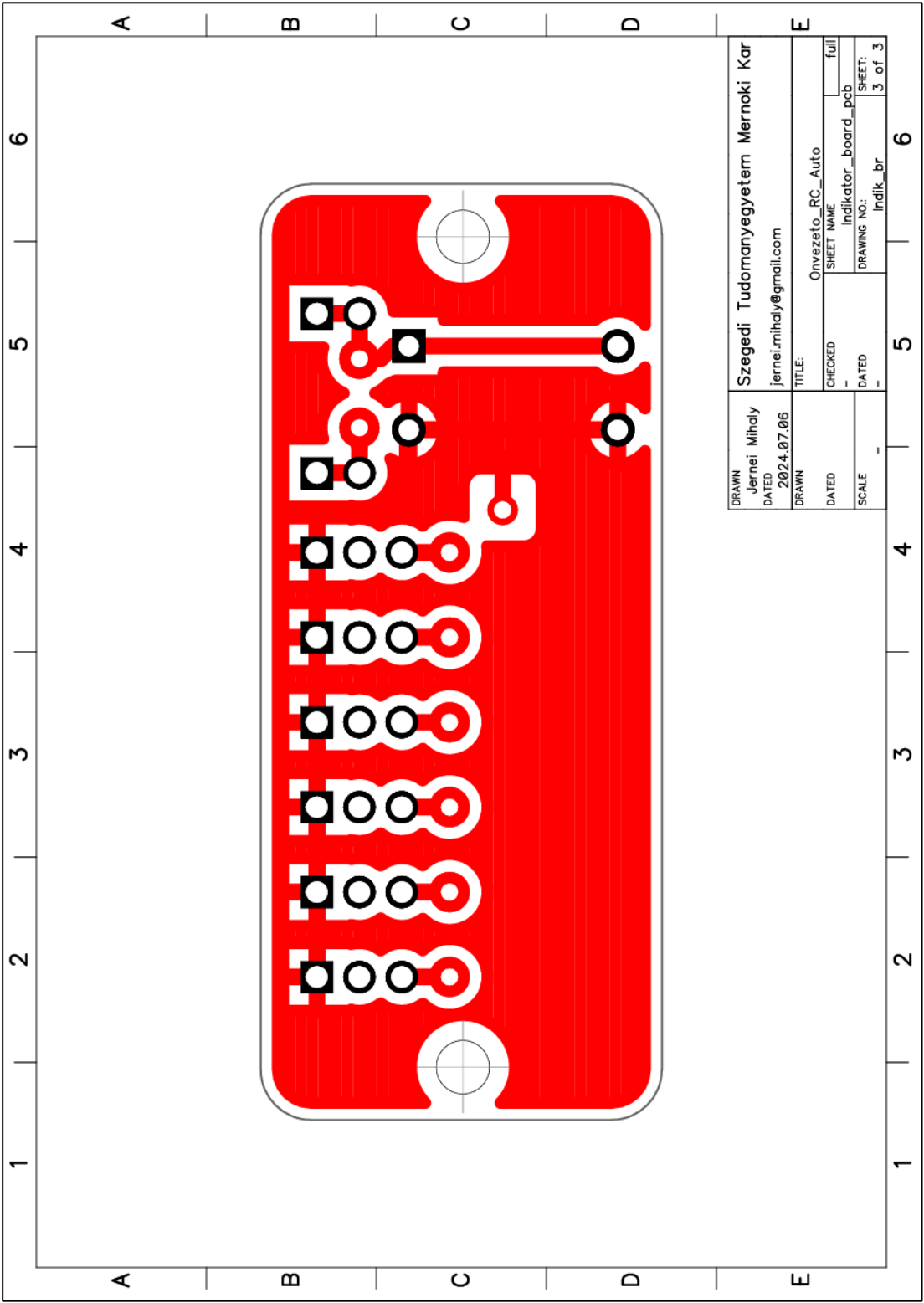
2. Melléklet: NVIDIA Jetson Nano B01 Development Kit 40-pin Exp-Header kiosztás

SoC GPIO	Linux GPIO #	Alternate Function	Default Function			Default Function	Alternate Function	Linux GPIO #	SoC GPIO
			3.3 VDC	①	②	5 VDC			
PJ.03	75	GPIO	I2C1_SDA	③	④	5 VDC			
PJ.02	74	GPIO	I2C1_SCL	⑤	⑥	GND			
PBB.00	216	AUD_CLK	GPIO	⑦	⑧	UART1_TXD	GPIO	48	PG.00
			GND	⑨	⑩	UART1_RXD	GPIO	49	PG.01
PG.02	50	UART1_RTS	GPIO	⑪	⑫	GPIO	I2S0_SCLK	79	PJ.07
PB.06	14	SPI1_SCK	GPIO	⑬	⑭	GND			
PY.02	194		GPIO	⑮	⑯	GPIO	SPI1_CS1	232	PDD.00
			3.3 VDC	⑰	⑱	GPIO	SPI1_CS0	15	PB.07
PC.00	16	SPI0_MOSI	GPIO	⑲	⑳	GND			
PC.01	17	SPI0_MISO	GPIO	㉑	㉒	GPIO	SPI1_MISO	13	PB.05
PC.02	18	SPI0_SCK	GPIO	㉓	㉔	GPIO	SPI0_CS0	19	PC.03
			GND	㉕	㉖	GPIO	SPI0_CS1	20	PC.04
PB.05	13	GPIO	I2C0_SDA	㉗	㉘	I2C0_CLK	GPIO	18	PC.02
PS.05	149	CAM_MCLK	GPIO	㉙	㉚	GND			
PZ.00	200	CAM_MCLK	GPIO	㉛	㉜	GPIO	PWM	168	PV.00
PE.06	38	PWM	GPIO	㉝	㉞	GND			
PJ.04	76	I2S0_FS	GPIO	㉟	㊱	GPIO	UART1_CTS	51	PG.03
PB.04	12	SPI1_MOSI	GPIO	㊲	㊳	GPIO	I2S0_DIN	77	PJ.05
			GND	㊴	㊵	GPIO	I2S0_DOUT	78	PJ.06

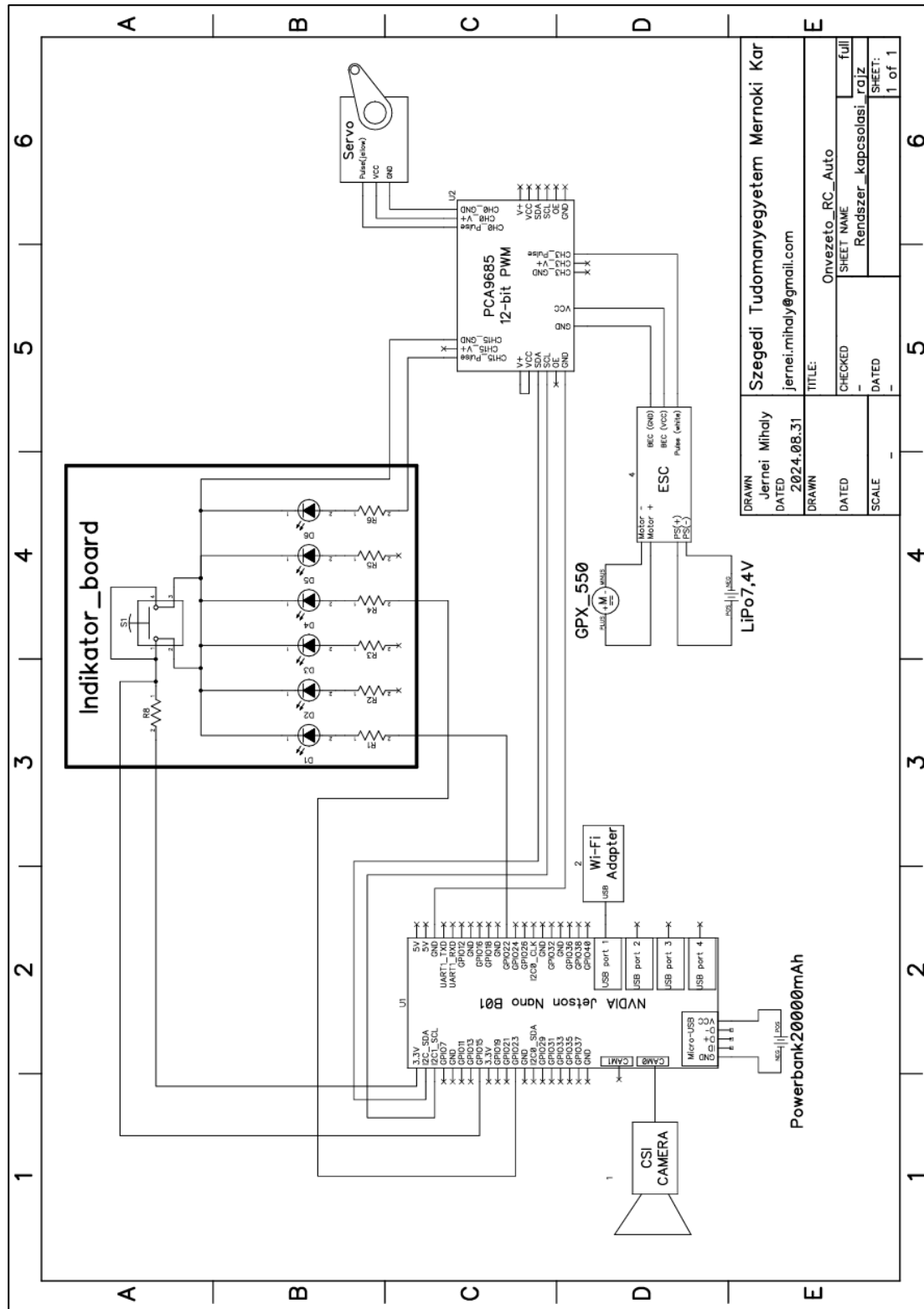
3. Melléklet: Indikátor áramkör kapcsolási és NYÁK-rajz







4. Melléklet: A rendszer teljes kapcsolási rajza



5. Melléklet: Az alapprogram forráskódja - „start_button_v2.py”

```

import Jetson.GPIO as GPIO
import time
import subprocess
import os
import signal
import pickle

# GPIO beállítása
gpio_button_pin = 15
gpio_led_pin = 22
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(gpio_button_pin, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(gpio_led_pin, GPIO.OUT)

running_process1 = None
program_1_closed = False

running_process2 = None
program_2_closed = False

press_count = 0

def send_control_data():
    vezەرles = [90, 95]

    # Megnyitjuk a fájlt és dump()-ot használjuk
    with open('file.pkl', 'wb') as file:
        pickle.dump(vezerles, file)
    #print(vezerles)

def start_program_1():
    global running_process1

    print("Program_1 indítása...")
    try:
        running_process1 = subprocess.Popen(["python3.8", "car_control.py"])

    except Exception as e:
        print(f"Hiba történt a program_1 indítása során: {e}")

def start_program_2():
    global running_process2, inditas_jelzo
    time.sleep(20)
    print("Program_2 indítása...")
    try:
        running_process2 = subprocess.Popen(["python3", "onnx_model_vezerlo.py"])
        #running_process2 = subprocess.Popen(["python3", "training_data_collect.py"])
    except Exception as e:
        print(f"Hiba történt a program_2 indítása során: {e}")

def stop_program_1():
    global running_process1, program_1_closed
    print("Program_1 leállítása...")
    try:
        os.kill(running_process1.pid, signal.SIGTERM)
        running_process1.wait()

    except Exception as e:
        print(f"Hiba történt a program_1 leállítása során: {e}")
    finally:
        running_process1 = None
        program_1_closed = True

def stop_program_2():
    global running_process2, program_2_closed
    print("Program_2 leállítása...")
    try:
        os.kill(running_process2.pid, signal.SIGTERM)
        running_process2.wait()

    except Exception as e:
        print(f"Hiba történt a program_2 leállítása során: {e}")
    finally:
        running_process2 = None
        program_2_closed = True

def button_callback(channel):
    global running_process1, program_1_closed, press_count, running_process2, program_2_closed

    button_state = GPIO.input(gpio_button_pin)
    print("Gombnyomás észlelve!")

```

```

if not GPIO.input(gpio_button_pin):
    time.sleep(1)
    if not GPIO.input(gpio_button_pin):
        press_count += 1
        button_state = GPIO.input(gpio_button_pin)
        GPIO.output(gpio_led_pin, GPIO.LOW)
        time.sleep(0.5)
        GPIO.output(gpio_led_pin, GPIO.HIGH)

if press_count > 2:
    press_count = 1
    running_process1 = None
    program_1_closed = False
    running_process2 = None
    program_2_closed = False

print("Nyomógomb állapota:", button_state)
print("Press count:", press_count)

if press_count == 2 and running_process1 is not None and not program_1_closed:
    send_control_data()
    stop_program_1()
    print("P1 STOP")
elif running_process1 is None and program_1_closed:
    print("A program_1 már be volt zárva.")
elif press_count == 1 and running_process1 is None and not program_1_closed:
    send_control_data()
    start_program_1()

    print("P1 START")

if press_count == 2 and running_process2 is not None and not program_2_closed:
    stop_program_2()

    print("P2 STOP")
elif running_process2 is None and program_2_closed:
    print("A program_2 már be volt zárva.")
elif press_count == 1 and running_process2 is None and not program_2_closed:
    start_program_2()
    print("P2 START")

try:

    # GPIO eseményfigyelő hozzáadása
    GPIO.add_event_detect(gpio_button_pin, GPIO.FALLING, callback=button_callback, bouncetime=2000)

    GPIO.output(gpio_led_pin, GPIO.HIGH)
    send_control_data()
    while True:
        time.sleep(0.1)

except KeyboardInterrupt:
    pass

finally:
    GPIO.output(gpio_led_pin, GPIO.LOW)
    GPIO.remove_event_detect(gpio_button_pin)
    GPIO.cleanup()
    send_control_data()

    if running_process1 is not None:
        print("F6 program leállítása...")
        os.kill(running_process1.pid, signal.SIGTERM)
        running_process1.wait()

    if running_process2 is not None:
        print("F6 program leállítása...")
        os.kill(running_process2.pid, signal.SIGTERM)
        running_process2.wait()

```

6. Melléklet: Az autóvezérlést végző program forráskódja - „car_control.py”

```

from adafruit_servokit import ServoKit
import time
from board import SCL, SDA
import busio
from adafruit_pca9685 import PCA9685

import sys
import pickle

i2c = busio.I2C(SCL, SDA)
pca = PCA9685(i2c)
pca.frequency = 54
led_channel1 = 15
led_pwm1 = pca.channels[led_channel1]

myKit = ServoKit(channels=16)
myKit.servo[0].angle = 95
myKit.servo[3].angle = 90.0
time.sleep(1)

speed = 90
steering = 95

def read_control_data():
    prevValue = None
    vezeres = None
    global speed, steering
    while True:
        # Open the file in binary mode
        with open('file.pkl', 'rb') as file:
            try:
                # Call load method to deserialize
                vezeres = pickle.load(file)
                speed = vezeres[0]
                steering = vezeres[1]

                if prevValue != vezeres:
                    prevValue = vezeres
                    myKit.servo[3].angle = speed
                    myKit.servo[0].angle = steering

                    #time.sleep(0.1)
            except Exception as e:
                pass

def jelzo_led():

    for _ in range(3):
        led_pwm1.duty_cycle = 600
        time.sleep(0.3)
        led_pwm1.duty_cycle = 0
        time.sleep(0.3)
        led_pwm1.duty_cycle = 600

def main():
    jelzo_led()
    read_control_data()

if __name__ == "__main__":
    try:
        main()

    except (KeyboardInterrupt, SystemExit):
        pass
    finally:
        try:
            led_pwm1.duty_cycle = 0x0 # Ki
            myKit.servo[0].angle = 95
            myKit.servo[3].angle = 90
        except Exception as e:
            print(f"Hiba történt a LED kikapcsolása során: {e}")

    try:
        sys.exit() # sys.exit() hívása a program leállításához
    except Exception as e:
        print(f"Hiba történt a második program leállítás során: {e}")

    try:
        del myKit
    except Exception as e:
        print(f"Hiba történt a myKit objektum törlése során: {e}")

```

7. Melléklet: Az tanító adatokat gyűjtő program forráskódja: „training_data_collect.py”

```

import cv2
import time
import pickle
from datetime import datetime
from pynput import keyboard
import threading
import Jetson.GPIO as GPIO

# GPIO beállítás
gpio_led_pin1 = 16
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(gpio_led_pin1, GPIO.OUT)

speed = 90
steering = 95
data_matrix = []

def send_control_data():
    global speed, steering

    # Újra létrehozuk a vezérles listát a megfelelő értékekkel
    vezérles = [speed, steering]

    # Megnyitjuk a fájlt és dump()-ot használjuk
    with open('file.pkl', 'wb') as file:
        pickle.dump(vezérles, file)
        print(vezérles)

def gstreamer_pipeline(
    sensor_id=0,
    capture_width=1280,
    capture_height=720,
    display_width=1280,
    display_height=720,
    framerate=60,
    flip_method=2,
):
    return (
        "nvarguscamerasrc sensor-id=%d ! "
        "video/x-raw(memory:NVMM), width=(int)%d, height=(int)%d, framerate=(fraction)%d/1 ! "
        "nvvidconv flip-method=%d ! "
        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
        "videoconvert ! "
        "video/x-raw, format=(string)BGR ! appsink"
        % (
            sensor_id,
            capture_width,
            capture_height,
            framerate,
            flip_method,
            display_width,
            display_height,
        )
    )

def capture_and_save_images(num_images):
    global speed, steering, data_matrix
    #print(gstreamer_pipeline(flip_method=0))
    video_capture = cv2.VideoCapture(gstreamer_pipeline(flip_method=2), cv2.CAP_GSTREAMER)
    if video_capture.isOpened():
        try:
            for i in range(num_images):
                ret_val, frame = video_capture.read()
                if ret_val:
                    now = datetime.now()
                    formatted_time = now.strftime("%Y_%m_%d_%H_%M_%S") + f"_{now.microsecond // 1000:03d}"
                    cv2.imwrite(f"{formatted_time}.jpg", frame) # Save the image
                    data_matrix.append([now, speed, steering])

                else:
                    print(f"Error: Failed to capture image")
                    time.sleep(0.05)
                    #time.sleep(6)

            finally:
                video_capture.release()

            with open("data.txt", "w") as f:
                for sor in data_matrix:

```

```

        f.write(' '.join(map(str, sor)) + '\n')

        GPIO.output(gpio_led_pin1, GPIO.LOW)
        speed = 90
        steering = 95
        send_control_data()

    else:
        print("Error: Unable to open camera")

def on_press(key):
    global speed, steering
    try:
        if key == keyboard.Key.up:
            speed += 0.2

            if speed > 110:
                speed = 110
                send_control_data()
        if key == keyboard.Key.down:
            speed -= 0.2

            if speed < 75:
                speed = 75
                send_control_data()

        if key == keyboard.Key.left:
            steering -= 10

            if steering < 25:
                steering = 25
                send_control_data()
        if key == keyboard.Key.right:
            steering += 10

            if steering > 165:
                steering = 165
                send_control_data()

    except AttributeError:
        pass

def release_press(key):
    global steering
    try:
        steering = 95
        send_control_data()
        #print("elengedve")

    except AttributeError:
        pass

def keyboard_controlling():
    while not stop_event.is_set():
        with keyboard.Listener(on_press=on_press) as listener:
            listener.join()

stop_event = threading.Event()

def jelzo_led():
    GPIO.output(gpio_led_pin1, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.output(gpio_led_pin1, GPIO.LOW)
    time.sleep(0.5)
    GPIO.output(gpio_led_pin1, GPIO.HIGH)
    print("Running")

def main():
    jelzo_led()

    thread1 = threading.Thread(target=keyboard_controlling)
    thread1.start()

    #capture_and_save_images(1050)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        pass

    finally:

```

```
stop_event.set()  
GPIO.output(gpio_led_pin1, GPIO.LOW)  
GPIO.cleanup()  
  
pass
```

8. Melléklet: A betanított CNN modellt alkalmazó program forráskódja: „onnx_model_vezerlo.py”

```

import cv2
import time
import pickle
import os
from datetime import datetime
from pynput import keyboard
import threading
import Jetson.GPIO as GPIO

import onnxruntime as rt
import numpy as np
from PIL import Image

# ONNX modell betöltése
sess = rt.InferenceSession('model24.onnx')

# GPIO beállítása
gpio_led_pin1 = 16
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(gpio_led_pin1, GPIO.OUT)

speed = 104.0
steering = 95
#data_matrix = []

def send_control_data():
    global speed, steering

    if steering > 165:
        steering = 165

    if steering < 25:
        steering = 25

    # Újra létrehozuk a vezérles listát a megfelelő értékekkel
    vezérles = [speed, steering]

    # Megnyitjuk a fájlt és dump()-ot használjuk
    with open('file.pkl', 'wb') as file:
        pickle.dump(vezérles, file)
        print(vezérles)

def gstreamer_pipeline(
    sensor_id=0,
    capture_width=1280,
    capture_height=720,
    display_width=1280,
    display_height=720,
    framerate=60,
    flip_method=2,
):
    return (
        "nvarguscamerasrc sensor-id=%d ! "
        "video/x-raw(memory:NVMM), width=(int)%d, height=(int)%d, framerate=(fraction)%d/1 ! "
        "nvvidconv flip-method=%d ! "
        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
        "videoconvert ! "
        "video/x-raw, format=(string)BGR ! appsink"
        % (
            sensor_id,
            capture_width,
            capture_height,
            framerate,
            flip_method,
            display_width,
            display_height,
        )
    )

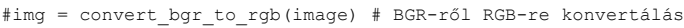
def convert_bgr_to_rgb(image):
    return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

def preprocess_image(image):
    # Kép fényerejének növelése

    alpha = 1.0 # Fényerősség tényező (1.0 jelentése nincs változás, >1.0 növeli a fényerőt)
    beta = 10   # Előtolási érték (a pixelértékekhez hozzáadva, hogy növelje a fényerőt)
    #bright_img = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
    #cv2.imwrite(f"{datetime.now().strftime('%Y%m%d_%H%M%S')}.jpg", image) # Save the image
    #Kép előfeldolgozása a modell számára

```

```


    #img = convert_bgr_to_rgb(image) # BGR-ról RGB-re konvertálás

    img = cv2.resize(image, (64, 64))
    img = convert_bgr_to_rgb(img)
    img = cv2.convertScaleAbs(img, alpha=alpha, beta=beta)
    img = img.astype(np.float32)
    img = np.expand_dims(img, axis=0)

    return img / 255.0

def predict_steering(image):
    #Kormányzási szög előrejelzése a képből
    input_name = sess.get_inputs()[0].name
    img = preprocess_image(image)

    pred_onx = sess.run(None, {input_name: img})[0]
    return pred_onx[0]

def capture_and_prediction():
    global steering
    #print(gstreamer_pipeline(flip_method=0))
    video_capture = cv2.VideoCapture(gstreamer_pipeline(flip_method=2), cv2.CAP_GSTREAMER)
    if video_capture.isOpened():
        try:
            while True:

                ret_val, frame = video_capture.read()
                if ret_val:

                    # Kormányzási szög előrejelzése
                    predicted_angle = predict_steering(frame)
                    steering = int(predicted_angle) # Az előrejelzés alapján frissítjük a kormányzási
                    szöget

                    os.system('clear')

                    send_control_data()

                else:
                    print(f"Error: Failed to capture image")

                keyCode = cv2.waitKey(10) & 0xFF
                if keyCode == 27 or keyCode == ord('q'):
                    break

                #time.sleep(1)

            #time.sleep(0.02)

        finally:
            video_capture.release()

    else:
        print("Error: Unable to open camera")

def on_press(key):
    global speed, steering
    try:
        if key == keyboard.Key.up:
            speed += 0.2

            if speed > 110.0:
                speed = 110.0
                send_control_data()
        if key == keyboard.Key.down:
            speed -= 0.2

            if speed < 75.0:
                speed = 75.0
                send_control_data()

    except AttributeError:
        pass

def keyboard_controlling():
    while not stop_event.is_set():
        with keyboard.Listener(on_press=on_press) as listener:
            listener.join()
    stop_event = threading.Event()

def jelzo_led():
    GPIO.output(gpio_led_pin1, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.output(gpio_led_pin1, GPIO.LOW)

```

```

time.sleep(0.5)
GPIO.output(gpio_led_pin1, GPIO.HIGH)
print("Running")

def main():
    jelzo_led()

    thread1 = threading.Thread(target=keyboard_controlling)
    thread1.start()

    capture_and_prediction()

    #print(steering)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        pass

    finally:

        GPIO.output(gpio_led_pin1, GPIO.LOW)
        GPIO.cleanup()

        speed = 90.0
        steering = 95
        send_control_data()

        pass

```

9. Melléklet: Tanító adatok előkészítését végző program forráskódja: „tanito_adatok_elokeszítése.py”

```
import os
import pandas as pd
import time

#Képek elérési útja
folder_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\osszes_kep_zajjal4'
#Képnevek txt mentési helye
txt_file_path_kepnevek = os.path.join(folder_path,

r'D:\Diplomamunka\Trainin_data\negyedik_adat\osszes_kep_zajjal4\kepnevek.txt')

#összefuzes
# Első txt fájl elérési útja
file1_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\osszes_kep_zajjal4\kepnevek.txt'

# Második txt fájl elérési útja
file2_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\osszes_adat_zajjal4.txt'

# Kimeneti txt fájl elérési útja
output_file_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\kepnevek_adatok_zaj4.txt'

#CSV mentési helye
csv_file_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\driving_data_zajos4.csv'
shuffled_csv_file_path =
r'D:\Diplomamunka\Trainin_data\negyedik_adat\shuffled_driving_data_zajos4.csv'

def image_rename():
    global folder_path
    # Mappa elérési útja, ahol a képek vannak
    #folder_path =
    r'C:\Users\Misi\Desktop\Diplomamunka\Szoftver\Trainin_data\elso_tanito_adatok\osszes_kep'

    # Ellenőrizd a mappa tartalmát
    if not os.path.isdir(folder_path):
        print(f"A megadott elérési út ({folder_path}) nem egy mappa.")
        exit()

    # Fájlok listázása a mappában
    file_list = os.listdir(folder_path)

    # Csak képek kiválasztása (opcionális: támogatott fájltypusok alapján)
    supported_extensions = ['.jpg', '.jpeg', '.png', '.gif', '.bmp', '.tiff']
    image_files = [file for file in file_list if any(file.lower().endswith(ext) for ext in
supported_extensions)]

    # Képek átnevezése
    for index, image_name in enumerate(image_files, start=1):
        old_path = os.path.join(folder_path, image_name)

        # Új fájlnev generálása
        new_image_name = f"image{index}{os.path.splitext(image_name)[1]}" # pl. image1.jpg,
image2.jpg, stb.

        new_path = os.path.join(folder_path, new_image_name)

        # Fájl átnevezése
        os.rename(old_path, new_path)
        print(f"{image_name} átnevezve erre: {new_image_name}")

    print("Átnevezés kész.")

def image_name_to_txt():
    global folder_path
    global txt_file_path_kepnevek
    #folder_path = r'C:\Users\Misi\Desktop\Diplomamunka\Szoftver\Trainin_data\elso_adatok\osszes_kep'
    # Módosítsd a mappa elérési útvonalára

    # Ellenőrizzük, hogy a megadott elérési útvonal egy mappa-e
    if not os.path.isdir(folder_path):
        print(f'Error: {folder_path} is not a valid directory.')
        return

    # Listázzuk ki az összes fájlt a mappában
    files = os.listdir(folder_path)

    # Szűrjük ki csak a képfájlokat (feltételezzük, hogy mindegyik .jpg kiterjesztésű)
    image_files = [file for file in files if file.lower().endswith('.jpg')]

    # Ha nincs képfájl a mappában, akkor hibaüzenet
    if len(image_files) == 0:
        print(f'Error: No image files found in {folder_path}.')
        return
```

```

# Rendezzük a képfájlok neveit numerikus sorrendbe
image_files.sort(key=lambda x: int(x.lstrip('image').rstrip('.jpg')))

# Nyissunk egy txt fájlt írásra

with open(txt_file_path_kepnevek, 'w') as txt_file:
    for image in image_files:
        txt_file.write(image + '\n')

print(f'A képnevek sikeresen kimentve a {txt_file_path_kepnevek} fájlba.')

def txt_osszefuzo():
    global file1_path, file2_path, output_file_path

    # Első fájl sorainak olvasása
    with open(file1_path, 'r') as file1:
        file1_lines = file1.readlines()

    # Második fájl sorainak olvasása
    with open(file2_path, 'r') as file2:
        file2_lines = file2.readlines()

    # Ellenőrzés, hogy a két fájl ugyanannyi sort tartalmaz-e
    if len(file1_lines) != len(file2_lines):
        raise ValueError("A két fájl nem ugyanannyi sort tartalmaz")

    # Sorok összefűzése és írása az új fájlba
    with open(output_file_path, 'w') as output_file:
        for line1, line2 in zip(file1_lines, file2_lines):
            output_file.write(f"{line1.strip()} {line2.strip()}\n")

    print(f"A két fájl sorait sikeresen összefűztük és kiírtuk a(z) {output_file_path} fájlba.")

def csv_generator():
    global output_file_path, csv_file_path
    # TXT fájl beolvasása
    data = []
    with open(output_file_path, 'r') as file:
        for line in file:
            parts = line.strip().split()

            # Kép nevének és dátumának kinyerése

            image_name = parts[0] # Az 1. rész a kép neve (12_41_58.711745.jpg)

            # adat dátuma
            data_date = parts[1] # A 2. rész dátuma (2024 - 05 - 13)

            data_time = parts[2] # A harmadik rész az adat ideje (12: 41:58.733885)
            # Sebesség és kanyarodási szög
            speed = int(parts[3]) # A negyedik rész a sebesség
            angle = int(parts[4]) # Az ötödik rész a kanyarodási szög

            # csak a kanyarodás szabályozása
            # data.append({'image': image_name, 'angle': angle})

            # Ha kellene fog a sebességet is szabályozni)
            data.append({'image': image_name, 'speed': speed, 'angle': angle})

    # DataFrame létrehozása
    df = pd.DataFrame(data)

    # Mentés CSV fájlba
    df.to_csv(csv_file_path, index=False)
    print(f"A csv fájl sikeresen kimentve {csv_file_path} fájlba.")

def csv_data_shuffle():
    # CSV fájl beolvasása
    df = pd.read_csv(csv_file_path)

    # Sorok összekeverése
    df_shuffled = df.sample(frac=1).reset_index(drop=True)

    # Az összekevert adatok mentése egy új CSV fájlba
    df_shuffled.to_csv(shuffled_csv_file_path, index=False)

    print("A sorok sikeresen összekeverésre kerültek és elmentve az 'shuffled_driving_data.csv' fájlba.")

def main():
    image_rename()
    time.sleep(1)
    image_name_to_txt()
    time.sleep(1)
    txt_osszefuzo()
    time.sleep(1)
    csv_generator()
    time.sleep(1)

```

```
    csv_data_shuffle()

if __name__ == '__main__':
    main()
```

10. Melléklet: CNN betanítását végző program forráskódja: „neural_halo_tanitas.py”

```

import pandas as pd
import numpy as np
import os
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.optimizers import Adam

# Callback objektum példányosítása a loss és val_loss nyomon követéséhez
from tensorflow.keras.callbacks import Callback

#modell létrehozás és betanítás
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.regularizers import l2

from tensorflow.keras.callbacks import ReduceLROnPlateau

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.preprocessing.image import ImageDataGenerator

#ONNX formátumba exportálás
import tf2onnx
import onnx

global train_images, test_images, train_angles, test_angles

base_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\osszes_kep_zajjal4'

csv_file_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\shuffled_driving_data_zajos4.csv'

#tf_model hivatkozása
model_path = 'model27.h5'

# ONNX modell mentési útvonala
onnx_model_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\model27.onnx'

loss_history_name = 'loss_history_27'
log_file_path = fr'D:\Diplomamunka\Trainin_data\negyedik_adat\{loss_history_name}.txt'

#-----Adatok betöltése-----

def training_data_loading_and_preprocessing():
    global train_images, test_images, train_angles, test_angles, csv_file_path, base_path

    # CSV fájl betöltése

    data = pd.read_csv(csv_file_path)

    # Képek és címkék betöltése a CSV fájlból
    image_paths = data['image'].tolist()
    angles = data['angle'].tolist()
    speeds = data['speed'].tolist()

    # Képek betöltése és előfeldolgozása
    def load_images(image_paths, base_path):
        images = []
        for img_path in image_paths:
            img = load_img(os.path.join(base_path, img_path), target_size=(64, 64))
            img = img_to_array(img)
            images.append(img)
        return np.array(images)

    #A könyvtár, ahol a tanítási képek találhatóak
    #base_path = r'D:\Diplomamunka\Trainin_data\masodik_tanito_adatok\osszes_kep'
    images = load_images(image_paths, base_path)

    # Képek normalizálása 0 és 1 közötti értékekre
    images = images / 255.0

    # Az adatok felosztása tanító és teszt adatokra
    train_images, test_images, train_angles, test_angles = train_test_split(images, angles,
    test_size=0.15, random_state=42)

    print("Az adatok sikeresen betöltve és felosztva.")

#-----Callback definiálás naplózáshoz-----
class LossHistory(Callback):
    def on_epoch_end(self, epoch, logs=None):
        global log_file_path

        with open(log_file_path, 'a') as f:
            loss = logs.get('loss')

```

```

        val_loss = logs.get('val_loss')
        f.write(f'Epoch {epoch + 1}: loss = {loss}, val_loss = {val_loss}\n')

#-----CNN Modell Létrehozása-----

def CNN_model_create():

    # CNN model létrehozása
    model = Sequential([
        # Konvolúciós réteg 32 szűrővel
        Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
        MaxPooling2D((2, 2)), # Max pooling réteg
        Dropout(0.1), # Dropout réteg

        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Dropout(0.2),

        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Dropout(0.3),

        Flatten(), # Az adat síkba alakítása a dense rétegekhez

        Dense(128, activation='relu', kernel_regularizer=l2(0.001)), # L2 regularizáció
        Dropout(0.3), # Dropout a dense rétegnél

        Dense(64, activation='relu', kernel_regularizer=l2(0.001)), # L2 regularizáció
        Dropout(0.3), # Dropout a dense rétegnél

        Dense(1) # Kimeneti réteg, amely a kanyarodási szöveget adja
    ])
    # Modell fordítása: Adam optimalizáló és MSE veszteségfüggvény
    model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')
    return model

#-----CNN Modell betanítása-----

def CNN_model_trainig():
    global train_images, test_images, train_angles, test_angles, model_path
    # Modell létrehozása
    model = CNN_model_create()

    # Modell betanítása a callback-kel
    history = LossHistory() # Callback objektum példányosítása a loss és val_loss nyomon követéséhez
    # Learning rate dinamikus beállítása
    lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=1e-7)

    # Korai megállítási beállítás
    early_stopping = EarlyStopping(
        monitor='val_loss', # Figyeli a validációs veszteséget
        patience=10, # Ha x epochon keresztül nem javul, leállítja a tanulást
        restore_best_weights=True # A legjobb modell súlyait állítja vissza
    )

    #Adat augmentáció
    datagen = ImageDataGenerator(
        rotation_range=10, # Képek forgatása maximum 10 fokkal
        width_shift_range=0.1, # Vízhintes eltolás 10%-kal
        height_shift_range=0.1, # Függőleges eltolás 10%-kal
        shear_range=0.1, # Nyírás
        zoom_range=0.1, # Zoomolás
        horizontal_flip=False # Tükrözés
    )

    model.fit(train_images, np.array(train_angles),
              epochs=40,
              validation_data=(test_images, np.array(test_angles)),
              callbacks=[history, lr_scheduler, early_stopping],
              shuffle=False
    )
    ...
    model.fit(datagen.flow(train_images, np.array(train_angles), batch_size=32),
              epochs=40,
              validation_data=(test_images, np.array(test_angles)),
              callbacks=[history, lr_scheduler, early_stopping],
              shuffle=False
    )
    ...

    print("A modell sikeresen betanítva.")

    # Modell mentése

    model.save(model_path)
    print(f"A modell elmentve: {model_path}")

#-----Modell Exportálása ONNX Formátumba

```

```

def export_onnx():
    global onnx_model_path
    # Keras modell betöltése
    model = tf.keras.models.load_model(model_path)

    # Input specifikáció
    spec = (tf.TensorSpec((None, 64, 64, 3), tf.float32, name="input"),)

    # Keras modell konvertálása ONNX formátumba
    model_proto, _ = tf2onnx.convert.from_keras(model, input_signature=spec,
output_path=onnx_model_path)

    print(f"A modell sikeresen exportálva ONNX formátumba: {onnx_model_path}")

def main():
    training_data_loading_and_preprocessing()
    CNN_model_trainig()
    export_onnx()

if __name__ == '__main__':
    main()

```

11. Melléklet: A kanyarodási szög és sebesség előrejelzését végző program forráskódja: „onnx_model_vezerlo_v2.py”

```

import cv2
import time
import pickle
import os
from datetime import datetime
from pynput import keyboard
import threading
import Jetson.GPIO as GPIO

import onnxruntime as rt
import numpy as np
from PIL import Image

# ONNX modell betöltése
sess = rt.InferenceSession('model128.onnx')

# GPIO beállítása
gpio_led_pin1 = 16
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(gpio_led_pin1, GPIO.OUT)

speed = 104
steering = 95
#data_matrix = []

def send_control_data():
    global speed, steering

    if steering > 165:
        steering = 165

    if steering < 25:
        steering = 25

    if speed > 106:
        speed = 106

    if speed < 104:
        speed = 104

    # Újra létrehozuk a vezérles listát a megfelelő értékekkel
    vezérles = [speed, steering]

    # Megnyitjuk a fájlt és dump()-ot használjuk
    with open('file.pkl', 'wb') as file:
        pickle.dump(vezérles, file)
        print(vezérles)

def gstreamer_pipeline(
    sensor_id=0,
    capture_width=1280,
    capture_height=720,
    display_width=1280,
    display_height=720,
    framerate=60,
    flip_method=2,
):
    return (
        "nvarguscamerasrc sensor-id=%d ! "
        "video/x-raw(memory:NVMM), width=(int)%d, height=(int)%d, framerate=(fraction)%d/1 ! "
        "nvvidconv flip-method=%d ! "
        "video/x-raw, width=(int)%d, height=(int)%d, format=(string)BGRx ! "
        "videoconvert ! "
        "video/x-raw, format=(string)BGR ! appsink"
        % (
            sensor_id,
            capture_width,
            capture_height,
            framerate,
            flip_method,
            display_width,
            display_height,
        )
    )

def convert_bgr_to_rgb(image):

```

```

    return cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

def preprocess_image(image):
    # Kép fényerejének növelése

    alpha = 1.0 # Fényerősség tényező (1.0 jelentése nincs változás, >1.0 növeli a fényerőt)
    beta = 10    # Előtolási érték (a pixelértékekhez hozzáadva, hogy növelje a fényerőt)
    #bright_img = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)
    #cv2.imwrite(f"{datetime.now()}.jpg", image) # Save the image
    #Kép előfeldolgozása a modell számára
    #img = convert_bgr_to_rgb(image) # BGR-ről RGB-re konvertálás

    img = cv2.resize(image, (64, 64))
    img = convert_bgr_to_rgb(img)
    img = cv2.convertScaleAbs(img, alpha=alpha, beta=beta)
    img = img.astype(np.float32)
    img = np.expand_dims(img, axis=0)

    return img / 255.0

def predict(image):

    # Kormányzási szög és sebesség előrejelzése a képből
    input_name = sess.get_inputs()[0].name
    img = preprocess_image(image)

    # Modell futtatása és predikciók visszaadása
    pred_onx = sess.run(None, {input_name: img})

    # Ellenőrizzük a pred_onx formátumát
    #print(f"Prediction shape: {[p.shape for p in pred_onx]}") # Ellenőrizzük az outputok formátumát
    #print(f"Prediction data: {pred_onx}")

    # Különválasztás az előrejelzett szög és sebesség értékeknél
    steering_angle = pred_onx[0][0, 0] # Az első kimeneti érték (kanyarodási szög)
    speed = pred_onx[1][0, 0] # A második kimeneti érték (sebesség)

    return steering_angle, speed

def capture_and_prediction():
    global steering, speed
    #print(gstreamer_pipeline(flip_method=0))
    video_capture = cv2.VideoCapture(gstreamer_pipeline(flip_method=2), cv2.CAP_GSTREAMER)
    if video_capture.isOpened():
        try:
            while True:

                ret_val, frame = video_capture.read()
                if ret_val:

                    # Kormányzási szög előrejelzése
                    predicted_angle, predicted_speed = predict(frame)
                    steering = int(predicted_angle) # Az előrejelzés alapján frissítjük a kormányzási
                    #speed = int(predicted_speed)
                    os.system('clear')

                    send_control_data()

                else:
                    print(f"Error: Failed to capture image")

                keyCode = cv2.waitKey(10) & 0xFF
                if keyCode == 27 or keyCode == ord('q'):
                    break

                #time.sleep(1)

            #time.sleep(0.02)

        finally:
            video_capture.release()

    else:
        print("Error: Unable to open camera")

def on_press(key):
    global speed, steering
    try:
        if key == keyboard.Key.up:
            speed += 0.2

```

```

        if speed > 110:
            speed = 110
            send_control_data()
        if key == keyboard.Key.down:
            speed -= 0.2

        if speed < 75:
            speed = 75
            send_control_data()

    except AttributeError:
        pass

def keyboard_controlling():
    while not stop_event.is_set():
        with keyboard.Listener(on_press=on_press) as listener:
            listener.join()
    stop_event = threading.Event()

def jelzo_led():
    GPIO.output(gpio_led_pin1, GPIO.HIGH)
    time.sleep(0.5)
    GPIO.output(gpio_led_pin1, GPIO.LOW)
    time.sleep(0.5)
    GPIO.output(gpio_led_pin1, GPIO.HIGH)
    print("Running")

def main():
    jelzo_led()

    thread1 = threading.Thread(target=keyboard_controlling)
    thread1.start()

    capture_and_prediction()

    #print(steering)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        pass

    finally:
        GPIO.output(gpio_led_pin1, GPIO.LOW)
        GPIO.cleanup()

        speed = 90
        steering = 95
        send_control_data()

    pass

```

12. Melléklet: CNN betanítását végző program módosított változata a kanyarodási szög és sebesség feldolgozásával: „neural_halo_tanitas.py_v2.py”

```
import pandas as pd
import numpy as np
import os
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.optimizers import Adam

# Callback objektum példányosítása a loss és val_loss nyomon követéséhez
from tensorflow.keras.callbacks import Callback

#modell létrehozás és betanítás
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.regularizers import l2

from tensorflow.keras.callbacks import ReduceLROnPlateau

from tensorflow.keras.callbacks import EarlyStopping

from tensorflow.keras.preprocessing.image import ImageDataGenerator

#ONNX formátumba exportálás
import tf2onnx
import onnx

global train_images, test_images, train_angles, test_angles

base_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\osszes_kep_zajjal4'

csv_file_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\shuffled_driving_data_zajos4.csv'

#tf_model hivatkozása
model_path = 'model28.h5'

# ONNX modell mentési útvonala
onnx_model_path = r'D:\Diplomamunka\Trainin_data\negyedik_adat\model28.onnx'

loss_history_name = 'loss_history_28'
log_file_path = fr'D:\Diplomamunka\Trainin_data\negyedik_adat\{loss_history_name}.txt'

#-----Adatok betöltése-----

def training_data_loading_and_preprocessing():
    global train_images, test_images, train_angles, test_angles, train_speeds, test_speeds,
    csv_file_path, base_path

    # CSV fájl betöltése

    data = pd.read_csv(csv_file_path)

    # Képek és címkék betöltése a CSV fájlból
    image_paths = data['image'].tolist()
    angles = data['angle'].tolist()
    speeds = data['speed'].tolist()

    # Képek betöltése és előfeldolgozása
    def load_images(image_paths, base_path):
        images = []
        for img_path in image_paths:
            img = load_img(os.path.join(base_path, img_path), target_size=(64, 64))
            img = img_to_array(img)
            images.append(img)
        return np.array(images)

    #A könyvtár, ahol a tanítási képek találhatóak
    #base_path = r'D:\Diplomamunka\Trainin_data\masodik_tanito_adatok\osszes_kep'
    images = load_images(image_paths, base_path)

    # Képek normalizálása 0 és 1 közötti értékekre
    images = images / 255.0

    # Az adatok felosztása tanító és teszt adatokra
    train_images, test_images, train_angles, test_angles, train_speeds, test_speeds =
    train_test_split(images, angles, speeds, test_size=0.15, random_state=42)

    print("Az adatok sikeresen betöltve és felosztva.")

#-----Callback definiálás naplózáshoz-----
class LossHistory(Callback):
    def on_epoch_end(self, epoch, logs=None):
        global log_file_path
```

```

        with open(log_file_path, 'a') as f:
            loss = logs.get('loss')
            val_loss = logs.get('val_loss')
            f.write(f'Epoch {epoch + 1}: loss = {loss}, val_loss = {val_loss}\n')

#-----CNN Modell Létrehozása-----

from tensorflow.keras.layers import Input, Concatenate

def CNN_model_create():
    # Képek inputja
    image_input = Input(shape=(64, 64, 3))

    # CNN rész
    x = Conv2D(32, (3, 3), activation='relu')(image_input)
    x = MaxPooling2D((2, 2))(x)
    x = Dropout(0.1)(x)

    x = Conv2D(64, (3, 3), activation='relu')(x)
    x = MaxPooling2D((2, 2))(x)
    x = Dropout(0.2)(x)

    x = Conv2D(128, (3, 3), activation='relu')(x)
    x = MaxPooling2D((2, 2))(x)
    x = Dropout(0.3)(x)

    x = Flatten()(x)

    # Fully connected rétegek
    x = Dense(128, activation='relu', kernel_regularizer=l2(0.001))(x)
    x = Dropout(0.3)(x)

    x = Dense(64, activation='relu', kernel_regularizer=l2(0.001))(x)
    x = Dropout(0.3)(x)

    # Kimeneti réteg a szög predikcióhoz
    angle_output = Dense(1, name='angle_output')(x)

    # Kimeneti réteg a sebesség predikcióhoz
    speed_output = Dense(1, name='speed_output')(x)

    # Modell létrehozása két kimenettel
    model = tf.keras.models.Model(inputs=image_input, outputs=[angle_output, speed_output])

    # Modell fordítása több veszteségfüggvénnyel (szögre és sebességre külön)
    model.compile(optimizer=Adam(learning_rate=0.001), loss={'angle_output': 'mse', 'speed_output':
'mse'})

    return model

#-----CNN Modell betanítása-----

def CNN_model_trainig():
    global train_images, test_images, train_angles, test_angles, train_speeds, test_speeds, model_path
    # Modell létrehozása
    model = CNN_model_create()

    # Modell betanítása a callback-ekel
    history = LossHistory() # Callback objektum példányosítása a loss és val_loss nyomon követéséhez
    # Learning rate dinamikus beállítása
    lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, min_lr=1e-7)

    # Korai megállítás beállítása
    early_stopping = EarlyStopping(
        monitor='val_loss', # Figyeli a validációs veszteséget
        patience=10, # Ha x epochon keresztül nem javul, leállítja a tanulást
        restore_best_weights=True # A legjobb modell súlyait állítja vissza
    )

    #Adat augmentáció
    datagen = ImageDataGenerator(
        rotation_range=10, # Képek forgatása maximum 10 fokkal
        width_shift_range=0.1, # Vízhintes eltolás 10%-kal
        height_shift_range=0.1, # Függőleges eltolás 10%-kal
        shear_range=0.1, # Nyírás
        zoom_range=0.1, # Zoomolás
        horizontal_flip=False # Tükrözés
    )

    '''
    model.fit(train_images, np.array(train_angles),
              epochs=40,
              validation_data=(test_images, np.array(test_angles)),
              callbacks=[history, lr_scheduler, early_stopping],
              shuffle=False
    )'''

    model.fit(train_images,
              {'angle_output': np.array(train_angles), 'speed_output': np.array(train_speeds)},
              epochs=40,
              validation_data=(test_images,

```

```

        {'angle_output': np.array(test_angles), 'speed_output':
np.array(test_speeds)}),
        callbacks=[history, lr_scheduler, early_stopping],
        shuffle=False)

    print("A modell sikeresen betanítva.")

    # Modell mentése

    model.save(model_path)
    print(f"A modell elmentve: {model_path}")

#-----Modell Exportálása ONNX Formátumba

def export_onnx():
    global onnx_model_path
    # Keras modell betöltése
    model = tf.keras.models.load_model(model_path)

    # Input specifikáció
    spec = (tf.TensorSpec((None, 64, 64, 3), tf.float32, name="input"),)

    # Keras modell konvertálása ONNX formátumba
    model_proto, _ = tf2onnx.convert.from_keras(model, input_signature=spec,
output_path=onnx_model_path)

    print(f"A modell sikeresen exportálva ONNX formátumba: {onnx_model_path}")

def main():
    training_data_loading_and_preprocessing()
    CNN_model_trainig()
    export_onnx()

if __name__ == '__main__':
    main()

```

13. Melléklet: CNN modell betanítási paraméterei és teszt eredményei az első tesztpályán”

CNN modell betanítási paraméterei és tesztelési eredményei																
Pálya	Adatkészlet	Modell neve	Mintakép BAL [db]	Mintakép JOBB [db]	Mintakép BAL_zajos [db]	Mintakép JOBB_zajos [db]	Szegmentált mintakép?	Kanyarodási szög képtéke	Epoch szám	Learning rate	Korai leállítás? [határérték]	Előre kevert adatok?	Véletlenes adatbeosztás tanításkor?	Testst- adatok aránya	Módosítások	Testst eredménye
1.		modell_1	1000	1000	0	0	nem	10	30	0.0001	nem	nem	nem	20%	-	Nem követi a vonalat
		modell_2	1000	1000	0	0	nem	10	60	0.0001	nem	nem	nem	20%	-	Egy fokkal jobb, de nem követi a vonalat
		modell_3	4000	4000	2000	2000	nem	5	30	0.0001	nem	nem	nem	20%	-	Az elődökhöz képest rosszabb
		modell_4	4000	4000	1000	1000	nem	5	60	0.0001	nem	nem	nem	20%	-	Az elődökhöz képest rosszabb
		modell_5	4000	4000	1000	1000	nem	5	30	0.0001	nem	nem	nem	20%	-	Az elődökhöz képest rosszabb
		modell_6	4000	4000	1000	1000	nem	5	60	0.0001	nem	nem	nem	20%	-	Balra minimálisan követi a pályát, de nem jó
		modell_7	2500	2500	500	500	nem	5	40	0.0001	nem	nem	nem	20%	-	Rosszabb az előzőnél
		modell_8	2500	2500	500	500	nem	5	40	0.0001	nem	nem	nem	20%	-	Teljesen balra húz
		modell_9	4000	4000	1000	1000	igen	5	30	0.0001	nem	nem	nem	20%	-	Nem húz balra, viszont össze vissza megy
		modell_10	2000	2000	250	250	igen	5	30	0.0001	nem	nem	nem	20%	-	Nem jó
		modell_11	2000	2000	0	0	nem	5	30	0.0001	nem	nem	nem	10%	learning_rate hozzáadva (0.0001), dropout réteg hozzáadva	Sokkal rosszabb
3.		modell_12	3000	3000	0	0	nem	10	40	0.0001	nem	igen	igen	10%	-	Kanyarodni viszonylag tud, de az egyenes nem megy
		modell_13	3000	3000	0	0	nem	10	40	0.0001	nem	igen	nem	10%	-	Elődökhöz hasonló, de erősen balra húz
		modell_14	3000	3000	0	0	nem	10	40	0.0001	nem	igen	nem	20%	-	Elődökhöz hasonló, de erősen balra húz
		modell_15	3000	3000	0	0	nem	10	40	dinamikus	igen - 10 epoch	igen	nem	10%	Dinamikus learning_rate + early_stop	Nem jó
		modell_16	3000	3000	0	0	nem	10	40	dinamikus	igen - 20 epoch	igen	nem	10%	-	A kanyarodás talán jó, de a többi nem
		modell_17	3000	3000	0	0	nem	10	40	dinamikus	igen - 30 epoch	igen	nem	10%	-	Nagyjából követi a pályát, de nagyon könnyen eltéved
		modell_18	3000	3000	0	0	nem	10	60	dinamikus	nem	igen	nem	10%	-	Nagyjából követi a pályát, de nagyon könnyen eltéved
		modell_19	3000	3000	0	0	nem	10	40	dinamikus	igen -10	igen	nem	15%	Modell struktúrája és dropout mértéke módosítva, plusz dense réteg(128) hozzáadva	Körbement egyszer, ígéretes
		modell_20	3000	3000	250	250	nem	10	40	dinamikus	igen -10	igen	nem	15%	-	Az elődökhöz képest rosszabb
		modell_21	3000	3000	250	250	nem	10	40	dinamikus	igen -20	igen	nem	15%	-	Az 19-hez képest rosszabb
		modell_22	3000	3000	0	0	nem	10	40	dinamikus	igen -15	igen	nem	15%	Adat augmentáció hozzáadva	Az 19-hez képest rosszabb
4.		modell_23	4000	4000	0	0	nem	10	40	dinamikus	igen -10	igen	nem	15%	-	Simán körbement többször, de egy pár kör után el tud tévedni
		modell_24	4000	4000	400	400	nem	10	40	dinamikus	igen -10	igen	nem	15%	Simán körbement és nem veszti el a pályát(több mint 5 percig simán ment)	Ugyan úgy megy, egy kicsit szabadabban veszi az ívkört
		modell_25	4000	4000	600	600	nem	10	40	dinamikus	igen -10	igen	nem	15%	-	Ugyan úgy megy, egy kicsit szabadabban veszi az ívkört
		modell_26	4000	4000	800	800	nem	10	40	dinamikus	igen -10	igen	nem	15%	-	Itt már könnyen eltéved
		modell_27	4000	4000	1000	1000	nem	10	40	dinamikus	igen -10	igen	nem	15%	-	24-hez képest ugyan olyan jól megy
		modell_28	4000	4000	1000	1000	nem	10	40	dinamikus	igen -10	igen	nem	15%	Kanyarodási szög mellett a sebességet is megadja	Megoszlanak az erőforrások, fel olyan jó betanulás, mint a 27-nél, könnyen eltéved

14. Melléklet: CNN modell betanítási paraméterei és teszteredményei a második tesztpályán”

CNN modell betanítási paramtéerei és tesztelési eredményei																
Pálya	Adatkészlet	Modell neve	Mintakép BAL [db]	Mintakép JOBB [db]	Mintakép JOBB_zajos [db]	Mintakép BAL_zajos [db]	Szegmentált mintakép?	Kanyarodási szög léptéke	Epoch szám	Learning rate	Korai leállás? [határérték]	Előre kevert adatok?	Véletlenszerű adatbeolvasás tanításkor?	Teszt-adatok aránya	Módosítások	Teszt eredménye
TESZTPÁLYA - 2	5.	modell_29	5000	5000	400	400	nem	10	40	dinamikus	igen -10	igen	nem	15%	Csak a kanyarodási szöget adja meg	Könnyedén eltéved
		modell_30	5000	5000	400	400	nem	10	40	dinamikus	igen -10	igen	nem	15%	Kanyarodási szög mellett a sebességet is megadja	A sebességet változtatja a kanyarokban, de eltéved
		modell_31	4000	4000	400	400	nem	10	40	dinamikus	igen -10	igen	nem	15%	Csak a kanyarodási szöget adja meg	Eltéved
		modell_32	5000	5000	500	500	nem	10	40	dinamikus	igen -10	igen	nem	15%	Csak a kanyarodási szöget adja meg	Már kezdi követni a vonalat, de könnyen eltéved. Láthatóan a nagyobb mintaszám segített. Igéretes.
		modell_33	5000	5000	500	500	nem	10	40	dinamikus	igen -10	igen	nem	15%	Kanyarodási szög mellett a sebességet is megadja	A sebesség hirtelen megugrása miatt eltéved
	6.	modell_34	8000	8000	800	800	nem	10	40	dinamikus	igen -10	igen	nem	15%	Csak a kanyarodási szöget adja meg	Többször is körbe tud menni
		modell_35	8000	8000	800	800	nem	10	40	dinamikus	igen -10	igen	nem	15%	Kanyarodási szög mellett a sebességet is megadja	Többször is körbe tud menni, a sebesség szabályozása túl durva, így néha eltéved

15. Melléklet: A 24-es kísérlet sikeres működésének videófelvétele

A videófelvétel nem nyilvános, kizárólag a következő elérési linkekkel tekinthető meg.

YouTube link:

<https://youtu.be/XcKOQISyAuk?si=ZdichhaaxSAVPTFh>

QR kód:



16. Melléklet: A 34-es kísérlet sikeres működésének videófelvetele

A videófelvétel nem nyilvános, kizárólag a következő elérési linkekkel tekinthető meg.

YouTube link:

https://youtu.be/eg5Cu_D3jYI?si=pYeyi5iHM_yhOfOl

QR kód:

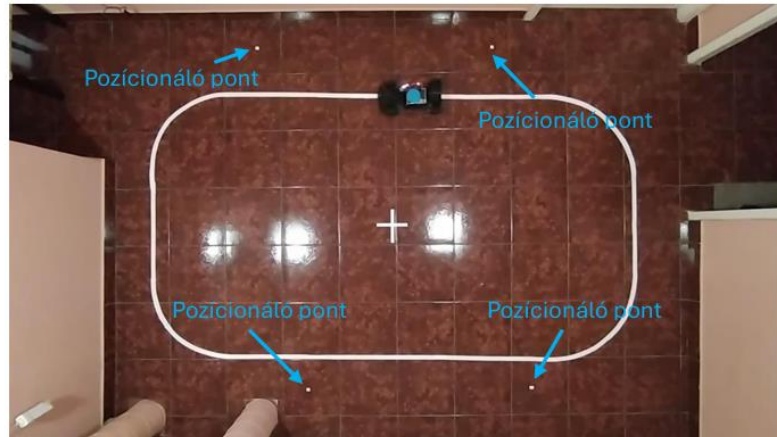


17. Melléklet: A pályakövetési hiba mérésének leírása

AZ AUTÓ ÚTVONALKÖVETÉSI PONTOSSÁGÁNAK VISSZAMÉRÉSE KÉPFELDOLGOZÁSI MÓDSZEREKKEL

1. Lépés: Videófelvétel készítése a pálya elemzéséhez

- Videó készítése a pályáról felülnézetben. A kamera a pálya közepe felett helyezkedik el.

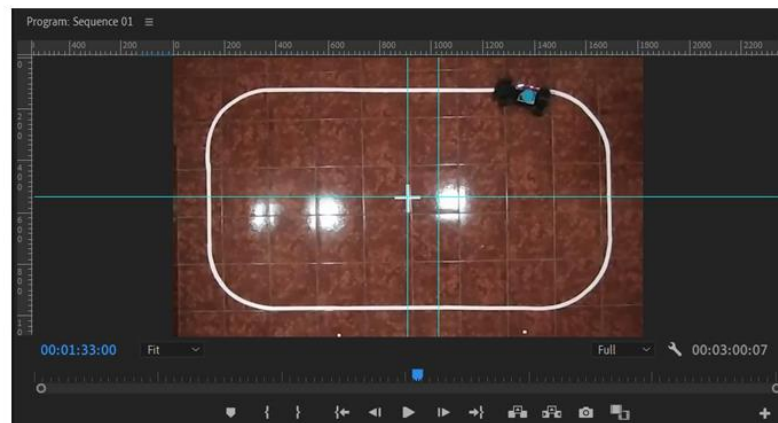


- Az autón egy elkülönülő színes(kék) korong kerül elhelyezésre, amely az elemzésnél segít a mozgási pálya kirajzolásában



2. Lépés: Videó méretezése szerkesztőprogram segítségével

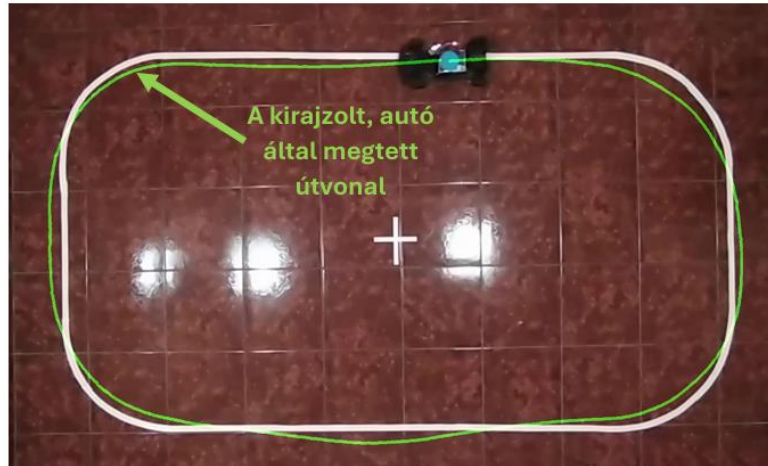
- A pályán elhelyezett pozicionáló pontok segítségével, melyek távolsága ismert, a videoszerkesztő program segítségével elvégezhető megfelelő nagyítás és pozicionálás a videófelvételen



- Ez a lépés elősegíti, hogy a következőkben minél pontosabb pixel/méter átváltást lehessen megvalósítani a videófelvételtől kinyert pálya konverziója során
- A méretezett videó méretei 1820x1080pixel, amely 3,36m x 2,00 m-nek felel meg

3. Lépés: Videó elemzése Python-OpenCV segítségével – pálya kinverése

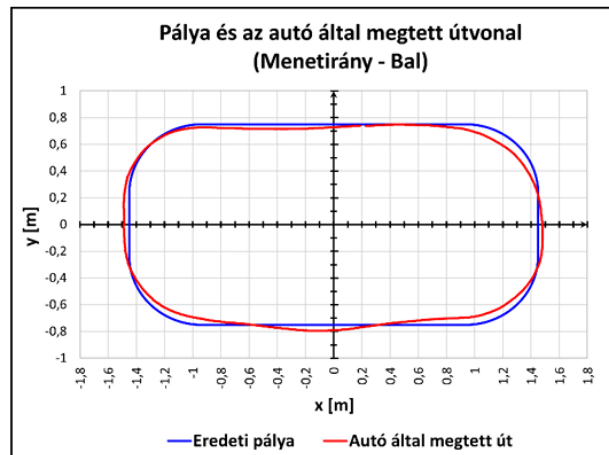
- A program futása során a vizsgált videón, OpenCV modul segítségével, az autón elhelyezett kék korong folyamatos pozíciókövetése történik az egymást követő képkockákon.
- Az elemzés során detektálásra kerül a kék korong kontúrja, majd annak mértani közepe, amely megadja az autó elhelyezkedését az adott képkockán egy pixelben.
- A teljes videó alatt zajlik a követés, így az autó által leírt pálya pixel koordináták formájában rendelkezésre állnak, melyek eltárolhatóak. Emellett az elemzés alatt a videón folyamatosan kirajzolásra kerül a az autó által megtett útvonal.



- Ezt követően az eltárolt pálya koordinátáit pixel mértékegységből méterbe [m] váltjuk át az ábrázolhatóság miatt.
- Az elemzést végző program forráskódja a **18. mellékletben** található

4. Lépés: A kinyert koordináták ábrázolása

- A kinyert útvonal koordinátákat egy grafikonon ábrázoljuk az eredeti pálya koordinátaival együtt, így azok összehasonlíthatóvá válnak
- Az eredeti pálya koordinátái szintén méter mértékegységben állnak rendelkezésre



5. Lépés: Pályakövetési hiba számítása

Az eredeti, illetve az autó által megtett út közötti eltérés számítását a **19. mellékletben** található programkód végzi el, amely kiszámítja a két pálya közötti átlag és a négyzetes átlag hibát, így adva mérőszámot a CNN szabályzással megvalósított pályakövetés pontosságára.

- **Interpoláció:**

- Az eredeti, illetve az autó által leírt pálya pontjainak elemszáma nem egyezik meg, mivel az autó hosszabb utat tesz meg az eltérő ívek miatt, így az eredeti pálya pontjainak számát (x,y koordinátáit) interpoláció segítségével hozzáigazítjuk az autó útvonalát képező adatok elemszámához.
- A programkód ezt **lineáris interpolációval** végzi el, melynek matematikai képlete:

$$f(x) = y_i + \frac{(y_{i+1} - y_i) \cdot (x - x_i)}{x_{i+1} - x_i}$$

ahol:

- $f(x)$ az interpolált y érték az adott x pozícióban
- x_i, y_i és x_{i+1}, y_{i+1} azok a pontok, melyek között az interpolálás történik
- x az a konkrét x érték, amelyhez a köztes y értéket keressük

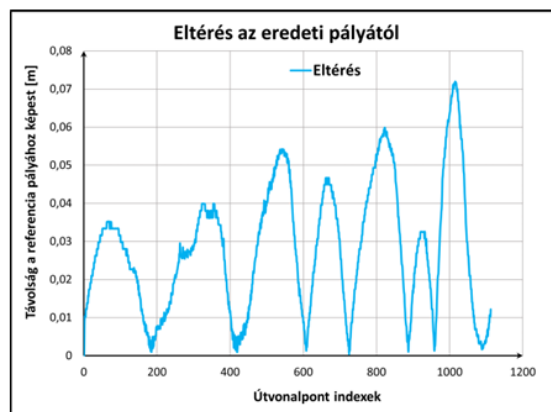
- **Távolság számítása az autó által megtett pálya és a már kibővített pontokkal leírt (interpolált pálya) referencia/eredeti pálya között:**

- A kód ezt követően kiszámítja a legközelebbi távolságot az autó aktuális pozíciója és a referencia pálya pontjai között.
- Ehhez az **euklideszi távolság matematikai képletét** alkalmazza:

$$d = \sqrt{(x_{auto} - x_{palya})^2 + (y_{auto} - y_{palya})^2}$$

ahol:

- x_{auto} és y_{auto} az autó aktuális pozíciójának derékszögű koordinátái
- x_{palya} és y_{palya} az interpolált pálya pontjainak derékszögű koordinátái
- Az így számított távolsági értékek ábrázolhatóak a vizsgált pályapontok függvényében, ahol a 0 [m] jelzi az eredeti pályavonalat:



- **Átlagos hiba számítása (Mean Error - E_{avg}):**

- A meghatározott eltérésekből ezt követően számításra kerül az átlagos hiba értéke, amely a következő képletet használja:

$$E_{avg} = \frac{1}{N} \sum_{i=1}^N e_i$$

ahol:

- N az eltérések száma
- e_i az i -edik pont eltérése
- Eredményként megkapjuk a teljes pályakövetési hiba átlagát

- **Négyzetes átlag hiba (RMS - E_{RMS}):**

- A kiszámított hibák négyzetének átlagából négyzetgyököt vonva meghatározható a pályakövetés RMS hibája
- Ennek matematikai képlete a következő:

$$E_{RMS} = \sqrt{\frac{1}{N} \sum_{i=1}^N e_i^2}$$

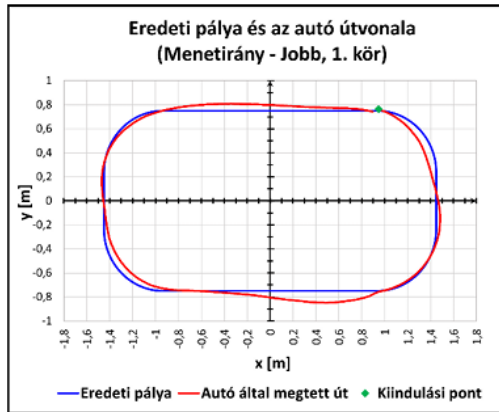
ahol:

- N az eltérések száma
- e_i az i -edik pont eltérése
- Eredményként megkapjuk a teljes pályakövetési hibák átlagának gyökét, amely érzékenyebb a nagyobb eltérésekre, így pontosabb mutatószámot kapunk általa

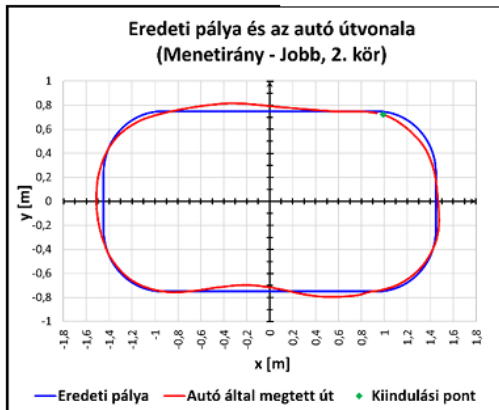
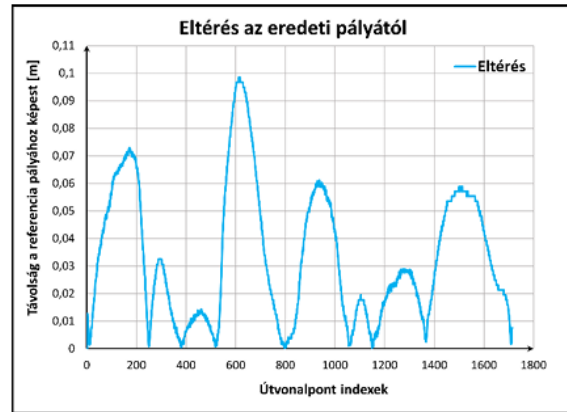
Mérési eredmények:

Pályakövetési hiba mérési eredményei			
Menetirány	Kör	Átlagos hiba [m]	RMS hiba [m]
Jobb	1.	0,0332	0,0413
Jobb	2.	0,0265	0,0324
Jobb	3.	0,0284	0,0380
Bal	1.	0,0380	0,0475
Bal	2.	0,0534	0,0695
Bal	3.	0,0282	0,0329

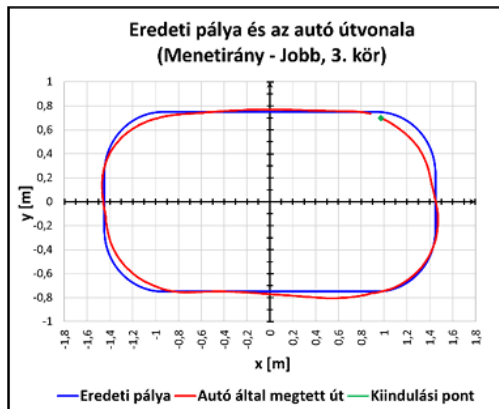
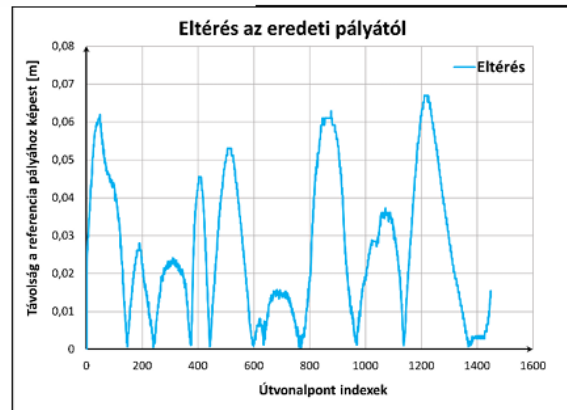
Pályakövetési pontosság mérésének eredményei



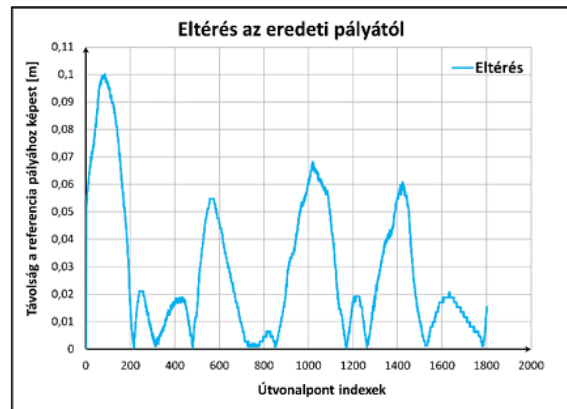
Jobb menetirány 1. kör mérési grafikonjai

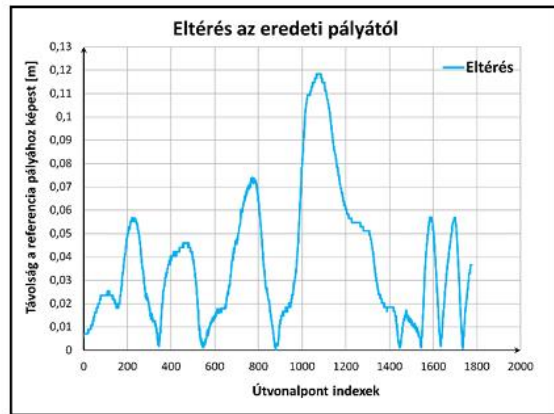
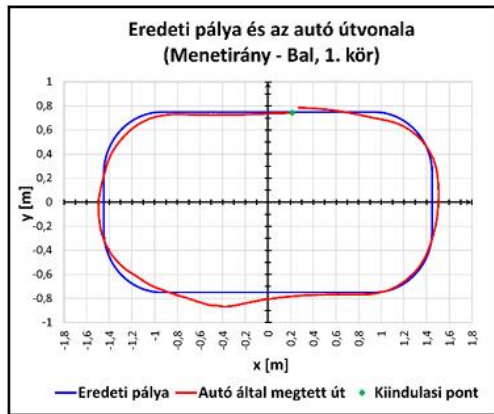


Jobb menetirány 2. kör mérési grafikonjai

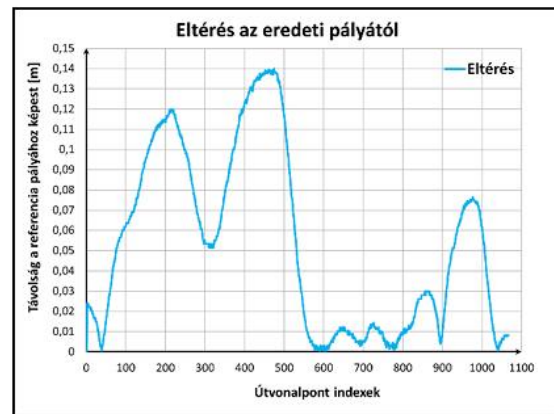
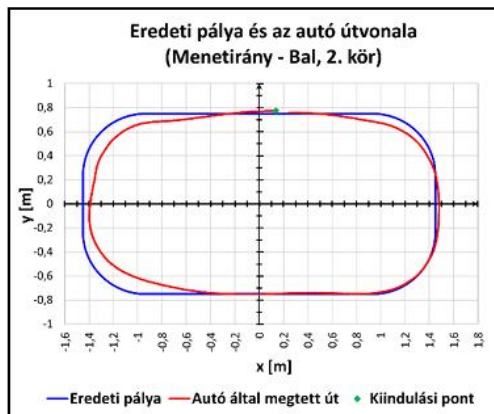


Jobb menetirány 3. kör mérési grafikonjai

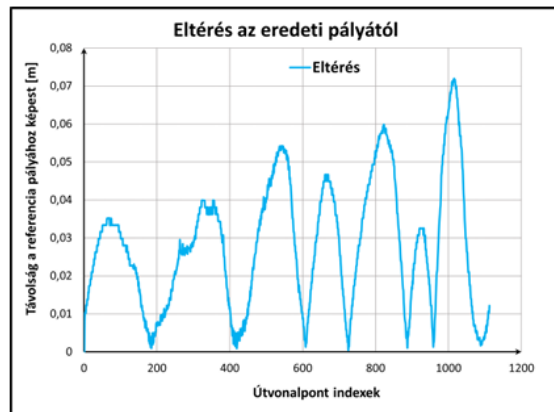
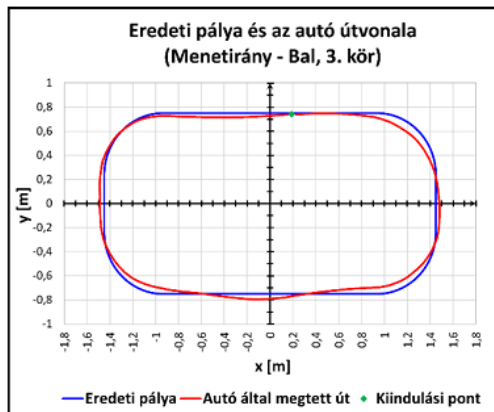




Bal menetirány 1. kör mérési grafikonjai



Bal menetirány 2. kör mérési grafikonjai



Bal menetirány 3. kör mérési grafikonjai

18. Melléklet: Az autó útvonalát meghatározó program forráskódja

```

import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.widgets import Button
import threading

# Videó betöltése
cap =
cv2.VideoCapture(r'E:\Diplomamunka_videok\mukodes_videok\hibameres\masodik_proba\szerkesztett\elso_LEF
T.mp4')

# Az utolsó frame előkészítése a későbbi kirajzoláshoz
last_frame = None

# Üres kép létrehozása a pálya kirajzolásához
ret, first_frame = cap.read()
height, width, _ = first_frame.shape
track_image = np.zeros((height, width, 3), dtype=np.uint8)

# Ablakméret állítása
cv2.namedWindow('Auto palyaja', cv2.WINDOW_NORMAL)
cv2.resizeWindow('Auto palyaja', 1280, 720)

# A keresendő szín színtartománya (HSV) - Autón lévő kék korong keresése
lower_hsv = np.array([90, 120, 130]) # Alsó határ (sötétebb verzió)
upper_hsv = np.array([110, 255, 255]) # Felső határ (világosabb és telítettebb)

# Előző pozíciók
previous_position = None

# Lista az autó által bejárt koordináták tárolására
path_points_x = []
path_points_y = []

# Pálya koordinátáinak beolvasása a CSV fájlból (méterben)
csv_file = r'D:\Diplomamunka\Dimplomadolgozat\szimulacio\palya_koordinatak.csv' # CSV fájl elérési út
track_data = pd.read_csv(csv_file, sep=';') # Pontosvesszővel választott oszlopok

# A pálya méterben megadott X és Y koordinátái
# Az adatok vesszővel vannak elválasztva -> csere pontra -> konverzió
track_data['x'] = track_data['x'].str.replace(',', '.').astype(float)
track_data['y'] = track_data['y'].str.replace(',', '.').astype(float)

# A pálya méterben megadott X és Y koordinátái
track_points_x_m = track_data['x'].values
track_points_y_m = track_data['y'].values

# Videó valós mérete méterben
real_video_width_m = 3.36 # Videó szélessége méterben
real_video_height_m = 2.0 # Videó magassága méterben

# Videó mérete pixelben
video_width, video_height = width, height

# Skálázás kiszámítása pixel/méter
scale_x = video_width / real_video_width_m # Pixel/méter az X tengelyen
scale_y = video_height / real_video_height_m # Pixel/méter az Y tengelyen

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    last_frame = frame.copy() # Utolsó frame mentése

    # Kép átalakítása HSV színtérre
    hsv_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # Maszk létrehozása a kék színű pont alapján
    mask = cv2.inRange(hsv_frame, lower_hsv, upper_hsv)

    # Kontúrok keresése a maszkon belül
    contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    if contours:
        # Legnagyobb kontúr kiválasztása
        largest_contour = max(contours, key=cv2.contourArea)

        # A kontúr középpontjának meghatározása
        M = cv2.moments(largest_contour)
        if M['m00'] > 0:
            cx = int(M['m10'] / M['m00'])
            cy = int(M['m01'] / M['m00'])

```

```

# Vonal rajzolása az autó útvonalára
if previous_position is not None:
    cv2.line(track_image, previous_position, (cx, cy), (0, 255, 0), 6)

# Frissítjük az előző pozíciót
previous_position = (cx, cy)

# Elmentjük a jelenlegi pozíciót a grafikonhoz
path_points_x.append(cx)
path_points_y.append(cy)

# Az eredeti pálya és az útvonal összefésülése
combined_frame = cv2.addWeighted(frame, 1, track_image, 0.6, 0)

# Kép megjelenítése
cv2.imshow('Auto pályája', combined_frame)

if cv2.waitKey(1) & 0xFF == 27:
    break

cap.release()
cv2.destroyAllWindows()

# A videó középpontja (pixel koordinátákban)
video_center_x = video_width / 2
video_center_y = video_height / 2

# Az autó által megtett útvonal átszámítása méteres koordinátákra és középre igazítása
path_points_x_m = [(x - video_center_x) / scale_x for x in path_points_x] # Középre igazítás X tengelyen
path_points_y_m = [(y - video_center_y) / scale_y for y in path_points_y] # Középre igazítás Y tengelyen

# Mentési funkció
def save_path(event):
    # Csak az átszámított koordináták mentése
    new_path_x_m = [x for x in path_points_x_m]
    new_path_y_m = [y for y in path_points_y_m]

    # Adatok mentése CSV fájlba
    df = pd.DataFrame({"x": new_path_x_m, "y": new_path_y_m})
    df.to_csv(r'D:\Diplomamunka\Dimplomadolgozat\szimulacio\car_path.csv', index=False)
    print("Adatok elmentve a car_path.csv fájlba")

# Grafikon kirajzolás
def plot_graph():
    fig, ax = plt.subplots(figsize=(8, 6))
    plt.subplots_adjust(left=0.1, bottom=0.2)

    # Pálya és autó által megtett útvonal rajzolása
    ax.plot(track_points_x_m, track_points_y_m, 'b-', linewidth=1, label='Eredeti pálya')
    ax.plot(path_points_x_m, path_points_y_m, 'r-', linewidth=1, label='Autó által megtett út')
    plt.title("Pálya és autó által megtett útvonal")
    plt.xlabel("X koordináta [m]")
    plt.ylabel("Y koordináta [m]")
    plt.grid()
    plt.legend(loc='center')
    plt.gca().invert_yaxis()

    # Mentés gomb hozzáadása
    save_button_ax = plt.axes([0.8, 0.05, 0.1, 0.075])
    save_button = Button(save_button_ax, 'Mentés')
    save_button.on_clicked(save_path)

    plt.show()

# Külön szálban a grafikon kirajzolása
thread = threading.Thread(target=plot_graph)
thread.start()

# Utolsó képkocka megjelenítése az autó útvonalával
if last_frame is not None:
    combined_last_frame = cv2.addWeighted(last_frame, 1, track_image, 0.6, 0)
    # Kép átméretezése (pl. 800x600 pixel)
    desired_size = (1180, 720) # Állítsd be a kívánt méretet
    resized_frame = cv2.resize(combined_last_frame, desired_size)
    cv2.imshow('Last frame and car path', resized_frame)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

```

19. Melléklet: Az eredeti és az autó által leírt pálya közötti hiba számítását végző program forráskódja:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d

# Eredeti pálya betöltése
original_track_df = pd.read_csv(r'D:\Diplomamunka\Dimplomadolgozat\szimulacio\palya_koordinatak.csv',
delimiteer=';', skiprows=1, names=['x', 'y'])
original_track_df['x'] = pd.to_numeric(original_track_df['x'].str.replace(',', '.'))
original_track_df['y'] = pd.to_numeric(original_track_df['y'].str.replace(',', '.'))
original_track_df = original_track_df.drop_duplicates().reset_index(drop=True)

# Autó útvonalának betöltése
car_path_df = pd.read_csv(r'D:\Diplomamunka\Dimplomadolgozat\szimulacio\car_path_3_R.csv')
car_path_df['x'] = pd.to_numeric(car_path_df['x'])
car_path_df['y'] = pd.to_numeric(car_path_df['y'])

# Ellenőrizzük az NaN értékeket
print("Original Track NaN Values:")
print(original_track_df.isna().sum())
print("\nCar Path NaN Values:")
print(car_path_df.isna().sum())

# NaN értékek eltávolítása
original_track_df = original_track_df.dropna().reset_index(drop=True)
car_path_df = car_path_df.dropna().reset_index(drop=True)

# Eredeti pálya pontjai
x_orig = original_track_df['x'].values
y_orig = original_track_df['y'].values

# Autó pálya pontjai
car_x = car_path_df['x'].values
car_y = car_path_df['y'].values

# Interpoláció az eredeti pálya adatokra
n_points = len(car_path_df) # Az autó pálya számához igazítva
interp_func_x = interp1d(np.arange(len(x_orig)), x_orig, kind='linear', fill_value="extrapolate")
interp_func_y = interp1d(np.arange(len(y_orig)), y_orig, kind='linear', fill_value="extrapolate")

# Új interpolált pálya
new_x_orig = interp_func_x(np.linspace(0, len(x_orig) - 1, n_points))
new_y_orig = interp_func_y(np.linspace(0, len(y_orig) - 1, n_points))

# Legközelebbi távolságok számítása
def nearest_distance(car_point, track_points):
    distances = np.linalg.norm(track_points - car_point, axis=1)
    return np.min(distances)

# Az autó útvonalához tartozó eltérések számítása
errors = []
for i in range(len(car_x)):
    car_point = np.array([car_x[i], car_y[i]])
    track_points = np.column_stack((new_x_orig, new_y_orig))

    # Ellenőrizzük, hogy a car_point érvényes-e
    if np.isnan(car_point).any():
        print(f"Invalid car point at index {i}: {car_point}")
        continue

    distance = nearest_distance(car_point, track_points)
    errors.append(distance)

print(f'Number of errors computed: {len(errors)}')
print(f'Errors: {errors[:10]}') # Az első 10 hiba megjelenítése

# Eltérések és pontindexek mentése CSV fájlba
errors_df = pd.DataFrame({
    'point_index': range(len(errors)), # Útvonalpont indexek
    'error': errors # Eltérések
})
errors_df.to_csv(r'D:\Diplomamunka\Dimplomadolgozat\szimulacio\tracking_errors_3_R.csv', index=False)
print("Eltérések és pontindexek elmentve a tracking_errors_with_index.csv fájlba.")

# Eltérések ábrázolása grafikonon
plt.figure(figsize=(10, 5))
plt.plot(errors, label='Követési hiba', color='b')
```

```

plt.title('Útvonalkövetési hibák')
plt.xlabel('Útvonalpont index')
plt.ylabel('Távolság a referencia pályához képest [m]')
plt.grid()
plt.legend()
plt.show()

# Összesített hiba és RMS hiba számítása - Négyzetes hiba
if len(errors) > 0:
    mean_error = np.mean(errors)
    rms_error = np.sqrt(np.mean(np.square(errors)))
    print(f'Átlagos hiba: {mean_error:.4f} [m]')
    print(f'RMS hiba - Négyzetes hiba: {rms_error:.4f} [m]')
else:
    print("No errors to calculate statistics.")

# Eredeti pálya, autó pályája és interpolált pálya ábrázolása - Segéd ábra
plt.figure(figsize=(10, 6))
plt.plot(x_orig, y_orig, label='Eredeti Pálya', color='b', linestyle='-', linewidth=3)
plt.plot(car_x, car_y, label='Autó Pálya', color='r', linestyle='--', linewidth=2)
plt.plot(new_x_orig, new_y_orig, label='Interpolált Pálya', color='y', linestyle=':', linewidth=2)
plt.title('Pályakövetés')
plt.xlabel('X Koordináta [m]')
plt.ylabel('Y Koordináta [m]')
plt.legend()
plt.grid()
plt.axis('equal') # Az arányok megtartása
plt.show()

```