ZESPÓŁ SZKÓŁ NR 4 W JAŚLE

Shadic

Autorzy projektu: Anna Opałka, Hubert Zając

Opiekun projektu: Wincenty Skwarek

Spis treści

Geneza projektu 4

Założenia konstrukcyjne i ogólny opis projektu 5
- Założenia konstrukcyjne 5
- Ogólny opis projektu 6

Część mechaniczna 8
- Podstawa 8
- Zespół napędowy i tylna oś 9
- Przednia oś 10

Część elektryczna 10

Oprogramowanie 11
- Kod projektu - sterowanie 11
- Aplikacja 15
- Program przetwarzający obraz w czasie rzeczywistym 16

Testy 18

Praktyczne zastosowanie projektu 20

Podsumowanie 21

Kosztorys 21

Wykaz rysunków i schematów 22

Źródła 23

1. Geneza projektu

We współczesnym świecie transport odgrywa bardzo ważną rolę. Coraz częstsze dostawy towarów pomiędzy dostawcami, a odbiorcami wymuszają zaangażowanie coraz większej ilości sprzętu. Oczywiście taki sprzęt musi obsługiwać człowiek co generuje dodatkowe koszty produktu końcowego. W dzisiejszych realiach producenci prześcigają się w różnych koncepcjach produkcji dążąc do jak najniższej ceny produktu finalnego dzięki czemu będzie on bardziej atrakcyjny dla klienta. Z drugiej strony producent oraz cały łańcuch dostawców dąży do jak największego zysku dla firmy. Przykładem takiego modelu jest chociażby lean manufacturing. Dlaczego więc nie oszczędzać na transporcie?

Od skonstruowania całkowicie autonomicznego samochodu dzieli nas jeszcze bardzo długa droga. Pojawiające się problemy nie tylko dotyczą zagadnień związanych z konstrukcją czy samym sterowaniem takiego pojazdu, ale również w grę wchodzą inne aspekty jak chociażby etyczne. Dla przykładu - kto ponosi odpowiedzialność za spowodowanie wypadku czy powstanie ofiar w czasie takiego wypadku gdy pojazd jest autonomiczny? - te kwestie nie zostały jeszcze rozwiązane. Łatwiejszą sprawą, przynajmniej pod względem prawnym, wydaje się być “zalegalizowanie” takich pojazdów wewnątrz hal produkcyjnych, gdzie można wydzielić pewne obszary, do których pracownicy nie mają wstępu - nie wchodzi wówczas w grę spowodowanie śmierci lub uszczerbku na zdrowiu pracownika.

Zastanowić się można jeszcze nad inną kwestią będącą kompromisem pomiędzy autonomicznością a manualnością tak jak jest to realizowane w eksploracji innych planet. Za wzorzec można przyjąć tutaj sterowanie łazikiem eksplorującym Marsa. Człowiek wydaje komendy sterowania takim łazikiem jednak ze względu na dużą odległość planet łazik nie może reagować w czasie rzeczywistym musi więc mieć zaimplementowaną możliwość podejmowania decyzji w nagłych nieprzewidzianych sytuacjach. Zastanawiając się nad tą tematyką pojawił się pomysł aby skonstruować samochód (ciężarówka), zdalnie sterowany tzn. operator (kierowca) będzie przebywał w centrali i prowadził go zdalnie do celu, a w przypadku pojawienia się nieprzewidzianych okoliczności zaimplementowany w samochodzie

moduł podejmie decyzję o zachowaniu pojazdu. Korzyści z takiego podejścia są olbrzymie:

- samochód może jechać cały czas, zmieniają się tylko kierowcy co osiem godzin,
- łatwa zmiana kierowcy w przypadku np. choroby kierowcy,
- podczas postoju samochodu na czas rozładunku lub załadunku, kierowca może prowadzić inny pojazd,
- eliminacja problemu trudnego życia w trasie

Takich przykładów można by wymienić jeszcze wiele.

Niekwestionowane zalety takiej koncepcji prowadzenia firmy transportowej stały się genezą do podjęcia niniejszej tematyki pracy.

Reasumując **celem projektu jest budowa modelu samochodu zdalnie sterowanego wiernie odwzorowującego prawdziwy pojazd, wyposażonego w moduł sztucznej inteligencji dzięki której w razie pojawiających się nieoczekiwanych sytuacji pojazd zareaguje na nie autonomicznie.**

2. Założenia konstrukcyjne i ogólny opis projektu

Założenia konstrukcyjne

Z założenia projekt ma jak najwierniej odzwierciedlać budowę samochodu, napęd na tylną oś, zastosowanie dyferencjału i skrętne przednie koła. Konstrukcja ma również pozwolić na zamontowanie komputera Raspberry Pi, kamery, akumulatora oraz pozostałych niezbędnych elementów.

W wyniku prowadzonych dyskusji w grupie wraz z opiekunem uznano, że projekt jest bardzo obszerny i podzielono go na etapy budowy, a także etapy funkcjonalności.

Etap I (realizowany w ramach projektu do konkursu „Od pomysłu do przemysłu”)

- pozyskanie niezbędnych podzespołów do budowy pojazdu: akumulator, silnik, dyferencjał,
- budowa platformy pojazdu z elementów wydrukowanych na drukarce 3d i drewnianej sklejki

- montaż podzespołów do platformy
- montaż elektroniki
- programowanie układu sterowania
- instalacja bibliotek przydatnych w przetwarzaniu obrazu w czasie rzeczywistym
- stworzenie programu pozwalającego na analizę obrazu w czasie rzeczywistym przez komputer
- testy

Etap 2 (Możliwy rozwój projektu po konkursie):

- sterowanie pojazdem za pomocą pada, ewentualnie za pomocą skonstruowanego w zeszłym roku przez inną drużynę symulatora https://www.youtube.com/watch?v=ISzCpdcXRQk&t=85s
- przesyłanie obrazu z kamery do telefonu
- możliwość identyfikacji wszystkich znaków znajdujących się na polskich drogach
- przypisanie znakom odpowiednich reakcji ze strony komputera

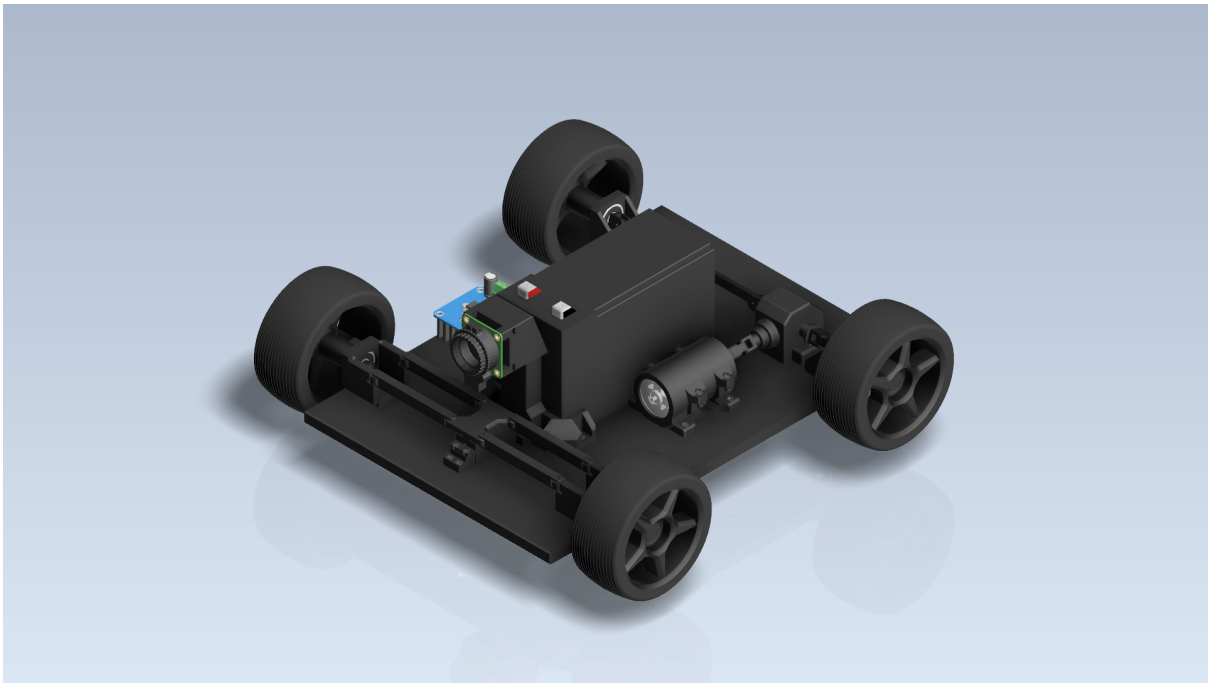
Zakończenie prac nad projektem jest rozumiane w znaczeniu zakończenia I etapu prac.

Ogólny opis projektu

Shadic - to czterokołowy zdalnie sterowany pojazd z napędem na tylną oś. Moment obrotowy generowany jest przez silnik DC o mocy 80W pracujący pod napięciem 12 V. Moment obrotowy przenoszony jest z silnika do dyferencjału modelarskiego za pomocą wału wyposażonego w przeguby Cardana . Zadaniem dyferencjału jest rozdział mocy na dwa koła z dostosowaniem prędkości kątowej dla każdego z kół, a także zmniejszenia prędkości kątowej wału a przez to zwiększenie momentu siły działającej na poszczególne koła tylne samochodu. piasty kół są łożyskowane. W piastach zamontowane są koła wydrukowane na drukarce 3D z ABS. Przednie koła są skrętne sterowane za pomocą serwomechanizmu modelarskiego. Moment siły skrętu przenoszony jest za pomocą drążków kierowniczych. Pojazd jest wyposażony w kamerę szerokokątną przeznaczoną do agregacji obrazu, który będzie

podlegał dalszej obróbce cyfrowej. Centralną jednostką sterującą jest komputer Raspberry Pi 4. Urządzenie jest zasilane akumulatorem żelowym o napięciu 12V i pojemności 7Ah. Całość platformą umożliwiającą wykorzystanie sztucznej inteligencji do rozpoznawania obrazów i wspomagania sterowania.

W pracy zastosowano podejście mechatroniczne. W pierwszym etapie zamodelowano gotowy pojazd przy użyciu oprogramowania Inventor, pozwoliło to na uniknięcie wielu błędów konstrukcyjnych już na etapie projektowania.



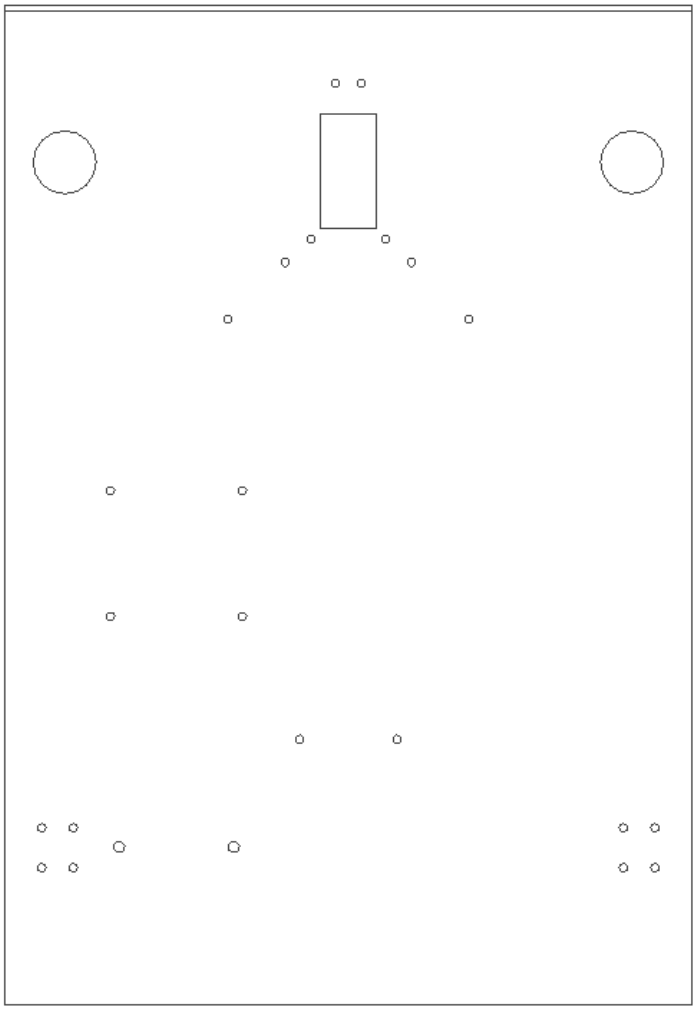
Rys. 1 Złożony model w programie Autodesk Inventor

Na rys. 1 przedstawiony jest model projektowanego urządzenia wykonany za pomocą oprogramowania Autodesk Inventor. Na platformie zostały rozlokowane poszczególne podzespoły samochodu przy zwróceniu uwagi na zachowanie środka ciężkości pojazdu na wzdłużnej osi symetrii. Zwrócono także uwagę by przeguby kardana nie pracowały pod zbyt dużymi kątami ze względu materiał z którego zostały wykonane oraz przy dbaniu o odpowiedni prześwit pod samochodem który umożliwi poruszanie się po lekko nierównej powierzchni.

3. Część mechaniczna

Podstawa

Podstawa jest integralną częścią samochodu do którego zostały zamontowane pozostałe podzespoły urządzenia. Powinna charakteryzować się dużą wytrzymałością na zginanie ze względu na obciążenie wywołane znaczną masą akumulatora zamontowanego w jej środku geometrycznym.

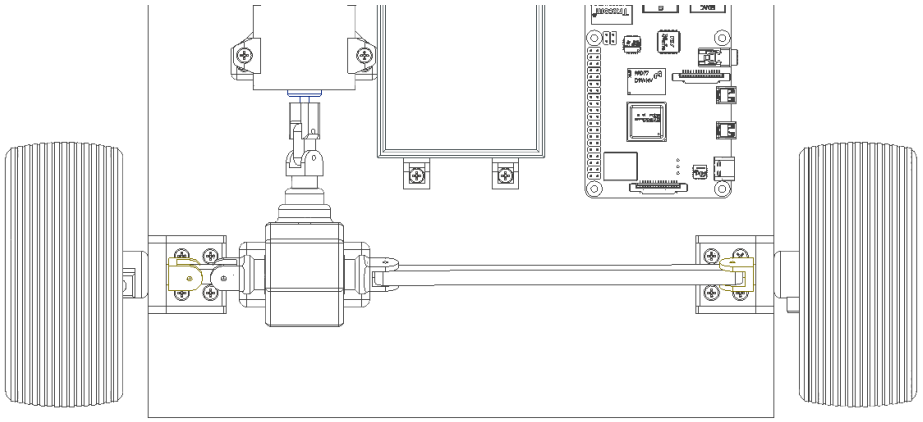


Rys. 2 Podstawa - zdjęcie z programu Inventor Autodesk

Podstawa została wykonana z drewnianej sklejki 10mm. Wstępnie sklejka została przycięta do odpowiednich wymiarów, kolejnie nawiercono otwory, oczyszczono powierzchnię papierem ściernym różnych gradacji i finalnie pomalowano na kolor czarny. Wykonane otwory na śruby mają średnicę 3mm.

Rysunek techniczny wykonawczy podstawy znajduje się w wykazie rysunków i schematów(rys_2.pdf).

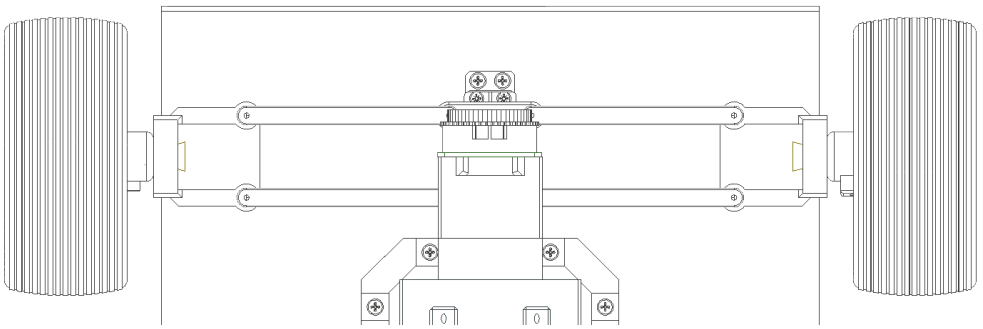
Zespół napędowy i tylna oś



Rys. 3 Zespół napędowy i tylna oś - zdjęcie z programu Inventor Autodesk

Mocy układu dostarcza silnik DC o mocy 80W na napięcie 12V. Moment obrotowy przenoszony jest do dyferencjału modelarskiego HSP 1/10 06063 06064 za pomocą przegubu Cardana wydrukowanego z materiału PLA na drukarce 3D. Zadaniem zastosowanego dyferencjału jest zmiana prędkości kątowych do odpowiednich wartości dla poszczególnych kół napędzających oraz zmniejszenie prędkości kątowej wału a przez to zwiększenie momentów sił działających na poszczególne koła. Tylne koła osadzone są na łożyskowanych piastach piastach. Zastosowane łożyska to 608 2RS. Koła wykonano z materiału PLA natomiast opony z materiału guma - elementy te zostały wydrukowane na drukarce 3D. Tylna oś została wykonanych z wykorzystaniem wałów napędowych także wykonanych na drukarce 3D. Elementy łączące wały z dyferencjałem(rys_17.pdf) oraz silnikiem(rys_18.pdf) i kołami(rys_10.pdf) zostały wymodelowane w programie Inventor Autodesk, a następnie wydrukowane na drukarce 3d. Do połączenia elementów został wykorzystany nieprzetworzony filament PLA.

Przednia oś



Rys. 4 Przednia oś - zdjęcie z programu Inventor Autodesk

Przednia oś jest osią skrętną. Kąt skrętu kół to około 45 stopni. Elementem wykonawczym jest serwomechanizm modelarski typ MG-996R, Na serwomotorze zamocowana jest kolumna kierownicza, której wychylenie przenoszone jest za pomocą drążków kierowniczych na koła. Obudowy piast koła (rys_3.pdf) osadzone są na łożyskach 608 2RS umieszczonych w otworach na podstawie. Prostopadle do podstawy w pieście koła znajduje się otwór na łożysko, do którego montowane jest koło.

4. Część elektryczna

Instalacja elektryczna składa się z akumulatora, włącznika, mostku h, przetwornicy step-down oraz komputera sterującego: Raspberry PI 4. Głównym źródłem zasilania jest akumulator żelowy o napięciu 12V i pojemności 7Ah. Za akumulatorem znajduje się główny wyłącznik prądu. Bezpośrednio z akumulatora zasilany jest mostek H sterownik o dużej wydajności prądowej 43A - BTS7960. Główną jednostką obliczeniową jest komputer Raspberry 4B+ 8GB RAM zasilany z akumulatora poprzez przetwornicę step-down LM2596 3A ustawioną na napięcie 5V. Z napięcia 5V zasilana jest także logika sterownika silnika DC oraz serwomechanizm. Komputer Raspberry Pi wyposażony jest w kamerę Raspberry Pi HQ IMX477R 12,3 MPx z obiektywem szerokokątnym. Szczegółowy zapis połączeń zawarty jest w schemacie instalacji elektrycznej (instalacja-elektryczna.pdf).

5. Oprogramowanie

Kod projektu - sterowanie

Zgodnie z tym co uprzednio zostało opisane Shadic posiada jeden układ sterujący oparty na jednopłytkowym komputerze Raspberry PI czwartej generacji wyposażony w 8 Gigabajtów pamięci operacyjnej i ma ma zainstalowany system operacyjny Raspbian w wersji 32 bitowej. Pozwoliło to na zastosowanie jednego programu, który zajmuje się obsługą sygnałów wejściowych i wyjściowych.

Importowanie bibliotek

```
import RPi.GPIO as GPIO
import bluetooth
import time
```

Rys. 5 Kod programu - importowanie bibliotek

Na samym początku kodu znajdują się instrukcje importujące niezbędne biblioteki.

Inicjacja zmiennych

```
SerwoSrodek = 5
SerwoLewo = 5.5
SerwoPrawo = 4.5

serwoPIN = 32
BlokadaLewa = 19
BlokadaPrawa = 21
PinPwmLewo = 33
PinPwmPrawo = 35

speed = 0
predkoscZ =""
Ruch =""
UstawienieSerwa=""
host = ""
port = 1

server = bluetooth.BluetoothSocket(bluetooth.RFCOMM)
```

Rys. 6 Kod programu - inicjacja zmiennych

W tej części kodu ustanawiane są zmienne odpowiadające za przypisanie nazw odpowiednim wyjściom GPIO, także wartości do danych zmiennych. Pozwala to na szybkie dostosowanie kodu w zależności od potrzeb gdyż zmieniamy daną wartość

tylko raz i w jednym miejscu, zamiast zmieniać ją w każdym miejscu w kodzie. Zapobiega to powstawaniu błędów i ułatwia czytelność kodu.

Przypisywanie roli i nadawanie stanów

```
GPIO.setup(BlokadaLewa, GPIO.OUT)
GPIO.setup(BlokadaPrawa, GPIO.OUT)
GPIO.setup(PinPwmLewo, GPIO.OUT)
GPIO.setup(PinPwmPrawo, GPIO.OUT)
GPIO.setup(serwoPIN, GPIO.OUT)

ser = GPIO.PWM(serwoPIN, 50)
p = GPIO.PWM(PinPwmPrawo, 50)
l = GPIO.PWM(PinPwmLewo, 50)

ser.start(serwo)
p.start(speed)
l.start(speed)
```

Rys. 7 Kod programu - przypisywanie roli i nadawanie stanów

W tym fragmencie ustalane są piny z sygnałem wyjściowym, tworzone są instancje PWM oraz uruchamiany jest sygnał PWM.

Łączenie się za pomocą sygnału Bluetooth z aplikacją na systemie Android

```
try:
        server.bind((host, port))
        print("Połączenie Bluetooth Udane")
except:
        print("Połączenie Bluetooth Nie powiodło się")
server.listen(1)

client, address = server.accept()
print("Połączony z:", address)
print("Klient:", client)
```

Rys. 8 Kod programu - nawiązywanie połączenia Bluetooth

Ten fragment odpowiada za połączenie się z Aplikacją za pomocą sygnału Bluetooth. Po udanym połączeniu wyświetla komunikat informujący o tym, podaje również informacje dotyczące urządzenia, z którym się połączył. W przypadku gdy nie uda się nawiązać połączenia wyświetla o tym informację.

Sterowanie pozycją przednich kół

```
def Lewo():
    if(UstawienieSerwa == "lewo"):
        ser.ChangeDutyCycle(SerwoLewo)

def Prawo():
    if(UstawienieSerwa == "prawo"):
        ser.ChangeDutyCycle(SerwoPrawo)

def Srodek():
    if(UstawienieSerwa == "srodek"):
        ser.ChangeDutyCycle(SerwoSrodek)
```

Rys. 9 Kod programu - zmiana pozycji serwomotoru

W zależności od stanu zmiennej, która jest zależna od danych przesyłanych przez aplikację ustawiona jest pozycja serwomotoru.

Sterowanie kierunkiem jazdy

```
def Przod():
    if(Ruch == "przod"):
        GPIO.output(BlokadaLewa, GPIO.HIGH)
        GPIO.output(BlokadaPrawa, GPIO.HIGH)
        GPIO.output(PinPwmLewo, GPIO.LOW)
        p.ChangeDutyCycle(speed)
        if(speed > 100):
            speed = 10

def Tyl():
    if(data == "tyl"):
        GPIO.output(BlokadaLewa, GPIO.HIGH)
        GPIO.output(BlokadaPrawa, GPIO.HIGH)
        GPIO.output(PinPwmPrawo, GPIO.LOW)
        l.ChangeDutyCycle(speed)
        if(speed > 100):
            speed = 10
```

Rys. 10 Kod programu - zmiana zwrotu w kierunku poruszania się

Funkcje “Przod” i “Tyl” decydują o kierunku obrotów silnika za pomocą sterownika silnika DC co przekłada się na zmianę zwrotu w kierunku poruszania się.

Przypisywanie wartości zmiennym

```
def zmienne():
    if(data == "srodek"):
        UstawienieSerwa = "srodek"
    elif(data == "lewo"):
        UstawienieSerwa = "lewo"
    elif(data == "prawo"):
        UstawienieSerwa = "prawo"
    elif(data == "przod"):
        Ruch = "przod"
    elif(data == "tyl"):
        Ruch = "tyl"
    elif(data == "szybciej"):
        predkoscZ = "szybciej"
    if(data == "wolniej"):
        predkoscZ = "wolniej"
```

Rys. 11 Kod programu - odpowiada za zmianę wartości zmiennych

Funkcja “zmienne” przypisuje zmiennym wartości w zależności od danych otrzymanych z aplikacji.

Pętla wywołująca funkcję

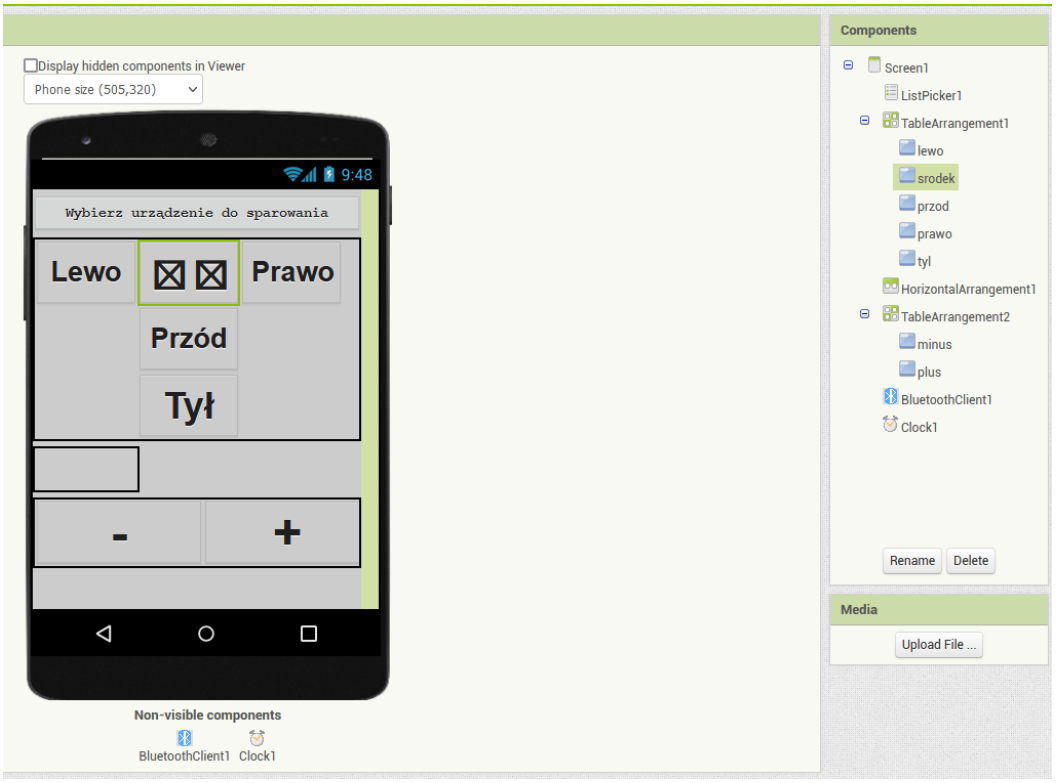
```
try:
        while True:
                data = client.recv(1024)
                print(data)
                zmienne()
                Lewo()
                Prawo()
                Srodek()
                predkosc()
                Przod()
                Tyl()
                client.send(send_data)
except:
        GPIO.cleanup()
        client.close()
        server.close()
```

Rys. 12 Kod programu - wywołuje funkcje

Pętla wywołuje wszystkie funkcje odpowiadające za sterowanie, aktualizuje również informacje przesłane z aplikacji.

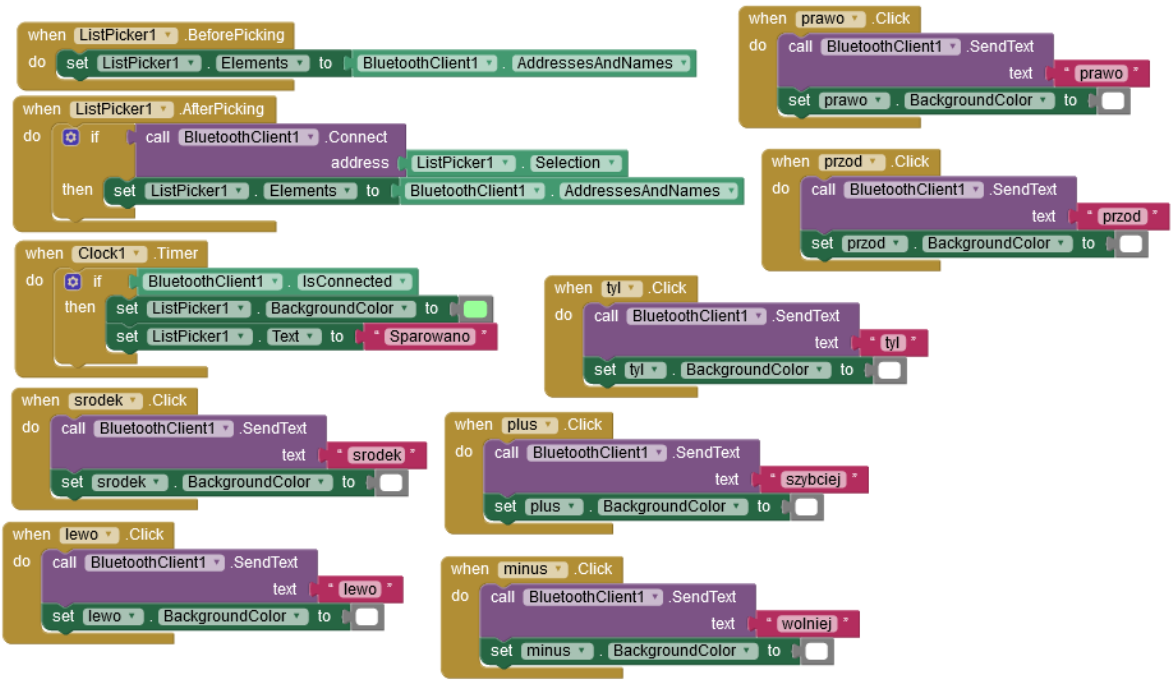
Aplikacja

Aplikacja odpowiadająca za sterowanie pojazdu została wykonana w programie MIT App Inventor. Jest to darmowe środowisko programistyczne aplikacji umożliwiające tworzenie aplikacji n asystem Android. Składa się z dwóch segmentów. Pierwszego w którym zarządzamy wyglądem naszej aplikacji oraz drugiego w którym za pomocą blokowego systemu programowania programujemy ją.



Rys. 13 Widok panelu konfiguracji aplikacji

W panelu konfiguracji dodano 7 przycisków oraz rozwijaną listę. Rozwijana lista po naciśnięciu wyświetla możliwe do sparowania urządzenia, a po sparowaniu zmienia kolor na zielony, a także wyświetlany tekst. Trzy przyciski odpowiadają za sterowanie serwomotorem. Dwa przyciski sterują zwrotem kierunku jazdy, a przyciski plus oraz minus prędkością obrotów silnika.

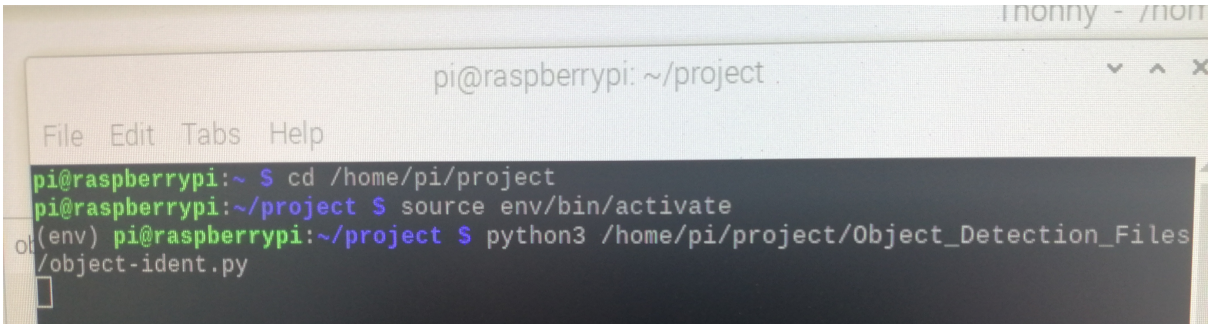


Rys. 14 Kod aplikacji - odpowiada za przesyłanie komend

Zaprogramowano przyciski tak aby po naciśnięciu ich zmieniały swój kolor na biały, a także przesyłały odpowiednią komendę do podłączonego komputera sterującego.

Program przetwarzający obraz w czasie rzeczywistym

Do przetwarzania obrazu w czasie rzeczywistym wykorzystano bibliotekę OpenCv wraz z zbiorem danych COCO(Common Objects in Context). Bibliotekę OpenCV zainstalowano wykorzystując wirtualne środowisko, które dopiero po aktywacji pozwala korzystać z zasobów biblioteki.



Rys. 15 Komendy uruchamiające program rozpoznawania obiektów

Na rysunku 15 w pierwszej komenda powoduje przejście do folderu, w którym umieszczone jest wirtualne środowisko. W drugiej linii widać kod aktywujący

wirtualne środowisko, natomiast w trzeciej linii jest uruchamiany kod odpowiedzialny za przetwarzanie obrazu.

```
import cv2

 = []
classFile = "/home/pi/project/Object_Detection_Files/coco.names"
with open(classFile,"rt") as f:
    classNames = f.read().rstrip("\n").split("\n")

SciezkaKonfiguracyjna = "/home/pi/project/Object_Detection_Files/ssd_mobilenet_v3_large_coco_2020_01_14.pbtxt"
weightsPath = "/home/pi/project/Object_Detection_Files/frozen_inference_graph.pb"

net = cv2.dnn_DetectionModel(weightsPath,SciezkaKonfiguracyjna)
net.setInputSize(320,320)
net.setInputScale(1.0/ 127.5)
net.setInputMean((127.5, 127.5, 127.5))
net.setInputSwapRB(True)
```

Rys. 16 Fragment kodu odpowiadający za importowanie biblioteki i wstępną konfigurację

W tym fragmencie podawane są ścieżki do plików zawierających nazwy możliwych do identyfikacji obiektów(4 linia kodu) oraz do wyszkolonego modelu z zbioru danych COCO. Pozwala on na identyfikację blisko 90 obiektów.

```
def getObjects(img, thres, nms, draw=True, objects=[]):
    classIds, confs, bbox = net.detect(img,confThreshold=thres,nmsThreshold=nms)
    if len(objects) == 0: objects = classNames
    Informacje_Obiektu =[]
    if len(classIds) != 0:
        for classId, confidence,box in zip(classIds.flatten(),confs.flatten(),bbox):
            className = classNames[classId - 1]
            if className in objects:
                Informacje_Obiektu.append([box,className])
                if (draw):
                    cv2.rectangle(img,box,color=(255,0,255),thickness=2)
                    cv2.putText(img,classNames[classId-1].upper(),(box[0]+10,box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX,1,(255,0,255),2)
                    cv2.putText(img,str(round(confidence*100,2)),(box[0]+200,box[1]+30),
                    cv2.FONT_HERSHEY_COMPLEX,1,(255,0,255),2)

    return img,Informacje_Obiektu
```

Rys. 17 Fragment kodu odpowiadający tworzenie obramowań, podpisu i wskaźnika procentowego wykrytego obiektu

Funkcja “getObject” tworzy nakładkę na zidentyfikowany obraz w postaci obramowań zaznaczających ramy wykrycia obiektu, podpisu zaczerpniętego z pliku “coco.names” oraz procentowego wskaźnika podającego prawdopodobieństwo że wykryty obiekt jest tym, którym go zidentyfikowano.

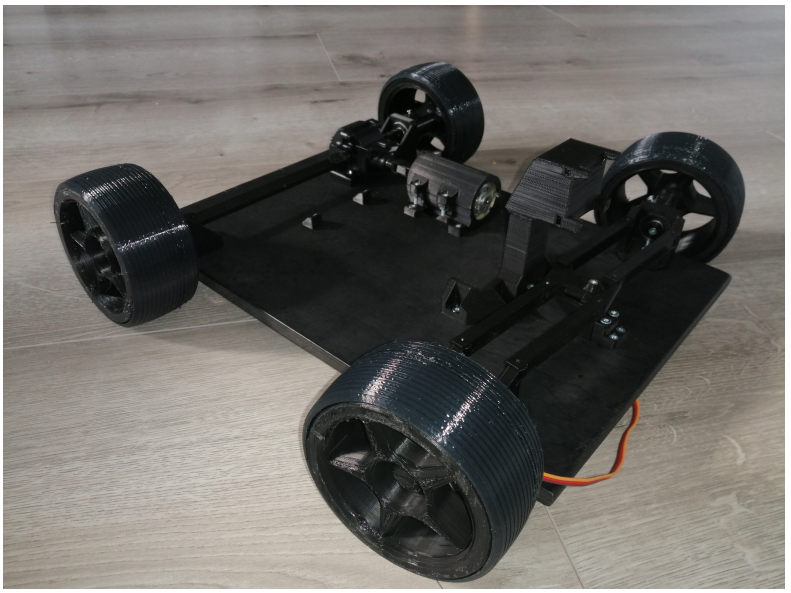
```
if __name__ == "__main__":

    cap = cv2.VideoCapture(0)
    cap.set(3,640)
    cap.set(4,480)


    while True:
        success, img = cap.read()
        result, Informacje_Obiektu = getObjects(img,0.45,0.2)
        cv2.imshow("Wynik",img)
        cv2.waitKey(1)
```

Rys. 18 Fragment kodu, który tworzy okno z wyświetlanym obrazem

Ten fragment kodu pomimo swoich niewielkich gabarytów, jest bardzo kluczowy, gdyż to on odpowiada za stworzenie okna z wyświetlanym obrazem. Parametr “0.45” jest decydującym dla działania całego programu, ponieważ odnosi się on do pewności identyfikacji obiektów. Jak można zauważyć na rysunkach zawartych w rozdziale Testy liczby, które są wskaźnikiem prawdopodobieństwa poprawnej identyfikacji są nie mniejsze niż 45.

6. Testy

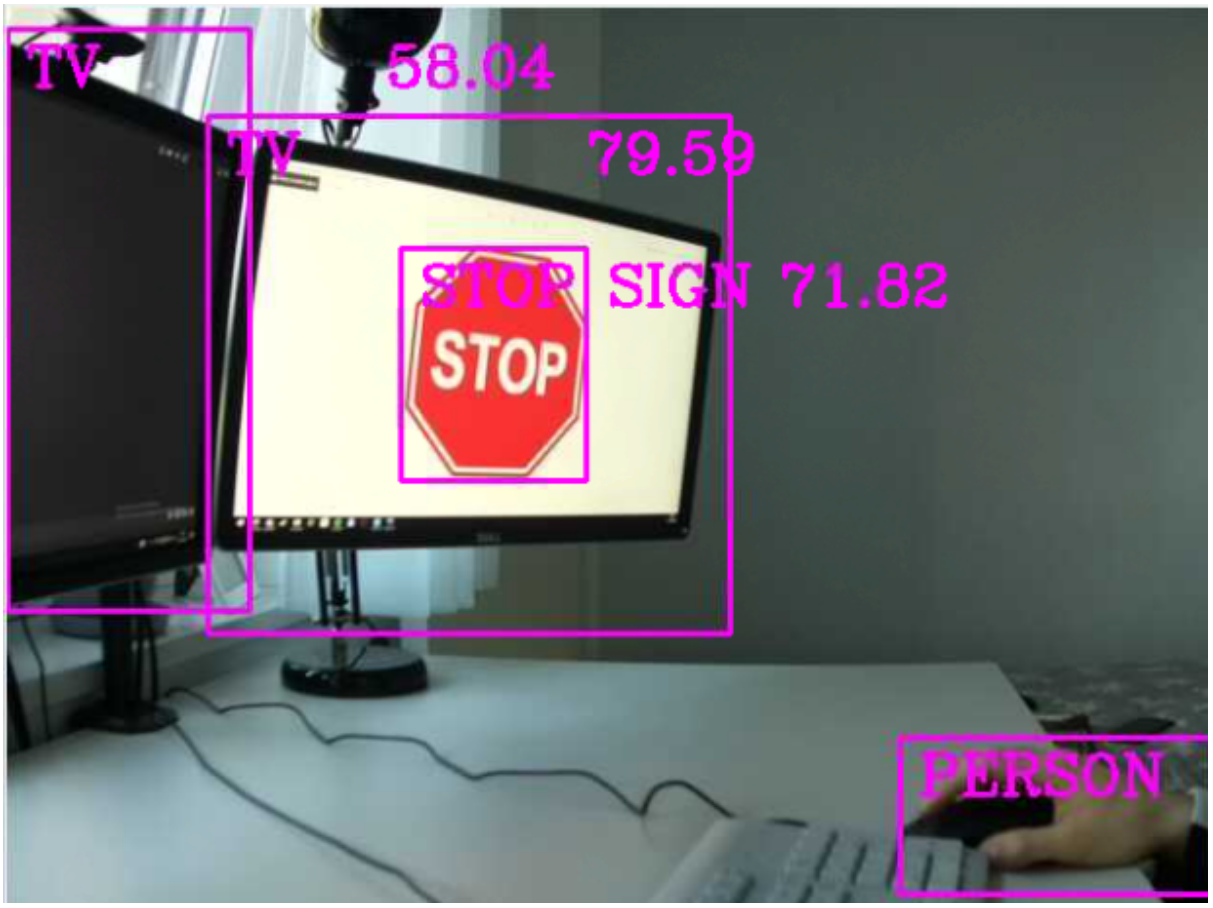
Pierwsze testy dotyczyły mobilności oraz wytrzymałości konstrukcji, wyniki były zadowalające. Pokazały także możliwe ścieżki rozwoju i udoskonalenia konstrukcji od strony mechanicznej.



Rys. 19 Konstrukcja pojazdu podczas pierwszych testów

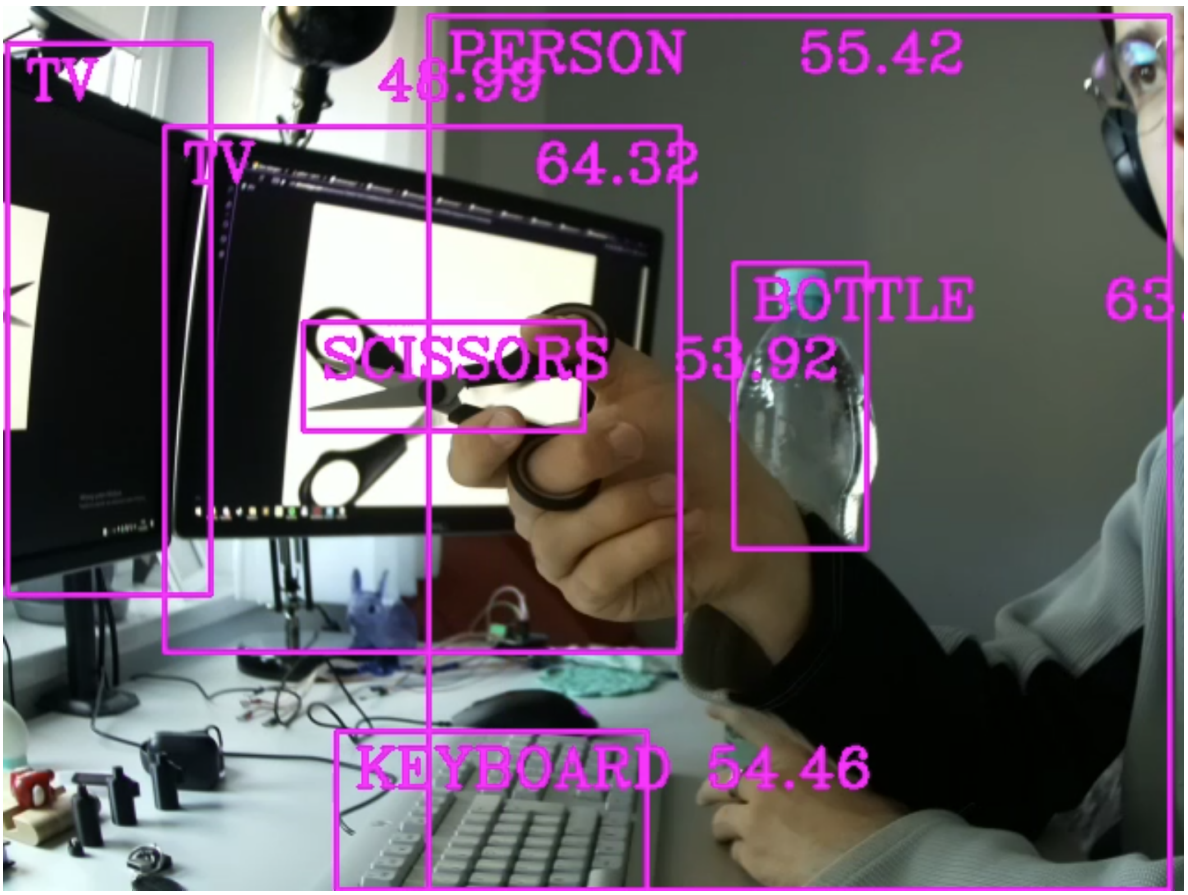
Podczas pierwszych testów mechanicznych sprawdzano zakres skrętu kół oraz wytrzymałość mechaniczną konstrukcji.

Podczas testów programu do przetwarzania obrazów wykorzystywano obrazy wyświetlana na monitorze, a także fizyczne obiekty.



Rys. 20 Testy oprogramowania(na zdjęciu wykryto: osobę, znak stopu oraz telewizor/monitor)

Wykorzystanie monitora podczas testów pozwoliło sprawdzić większą liczbę obiektów niż byłoby to możliwe z wykorzystaniem tylko fizycznych przedmiotów. Jak widać na rysunku powyżej program poprawnie zidentyfikował fizyczne obiekty, a także te wyświetlane na ekranie monitora.



Rys. 21 Testy oprogramowania z wykorzystaniem obiektów fizycznych widocznych w całości i fragmentach(na zdjęciu wykryto: osobę, klawiaturę, telewizor/monitor, nożyczki i butelkę wody)

Program był testowany także pod kątem wykrywania obiektów widocznych w fragmentach(na rysunku 21 jest to butelka i monitor). Fakt że butelka znajdowała się na trzecim planie, w przestrzeni zidentyfikowanego już obiektu i był widoczny tylko jej fragment nie przeszkodził programowi w poprawnej identyfikacji.

Wyniki otrzymane podczas testów programu do analizy obrazu okazały się bardzo dobre. Program pozwala na identyfikację około 90 obiektów. Obiekty, których sieć neuronowa miała więcej dostarczonych podczas szkolenia są wykrywane szybciej i lepiej. Model wykorzystywany do identyfikacji został zaczerpnięty ze zbiorów COCO.

7. Praktyczne zastosowanie projektu

Od samego początku celem było stworzenie platformy umożliwiającej wykorzystywanie sztucznej inteligencji. Budowa nadwozia pozwala na rozbudowę

urządzenia w przyszłości. Obecnie ma wiele zastosowań wśród których do najważniejszych można zaliczyć:

- poznanie podstawowej budowy samochodu
- nauka poprzez zabawę - opracowywanie nowych rozwiązań konstrukcyjnych
- nauka obsługi i rozwoju algorytmów rozpoznawania obrazów, głębokiego uczenia
- ćwiczenie umiejętności jazdy (po integracji z symulatorem Ekonomik Roadster)
- promocja klasy liceum o profilu politechnicznym

8. Podsumowanie

Faza pierwsza projektu została zakończona jednakże w planie jest dalszy rozwój poprzez zamianę sterowania aplikacją na pada, wykorzystanie telefonu jako wyświetlacza obrazu z kamery, stworzenie własnej bazy danych umożliwiającej identyfikację wszystkich znaków znajdujących się na polskich drogach.

9. Kosztorys

Nazwa elementu	Cena
Akumulator	45,99 zł
Dyferencjał	90,00 zł
Mostek H	41,71 zł
Łożyska	9,50 zł
Kontroler	40,00 zł
Filament PLA	50,00 zł

Kabel 2x2,5 mm miedź beztlenowa	16,00 zł
Elementy będące na wyposażeniu kółka:	
Raspberry Pi 4 8GB	650,00 zł
Kamera Raspberry Pi HQ IMX477R 12,3MPx	279,00 zł
Przetwornica Step-Down	10,00 zł
Obiektyw PT361060M3MP12 CS mount - szerokokątny 6mm	139,00 zł
Całkowity koszt:	1 371,20 zł

W kosztorysie nie uwzględniono sklejki drewnianej 10 mm, śrub m3 oraz silnika.

10. Wykaz rysunków i schematów

Część mechaniczna

1. Złożenie - złożenie.pdf
2. Podstawa - rys_2.pdf
3. Obudowa piasty koła - rys_3.pdf
4. Drążek kierowniczy - rys_4.pdf
5. Uchwyt serwa - przód - rys_5.pdf
6. Uchwyt serwa - tył - rys_6.pdf
7. Kolumna kierownicza - rys_7.pdf
8. Uchwyt Akumulatora i kamery - rys_8.pdf
9. Uchwyt Akumulatora - rys_9.pdf
10. Końcówka przegubu Cardana koło element 1 - rys_10.pdf
11. Blokada mocowania koła przedniego - rys_11.pdf
12. Blokada sterowania przodem - rys_12.pdf
13. Blokada podstawy silnika - rys_13.pdf
14. Mocowanie silnika podstawa - rys_14.pdf
15. Opona - rys_15.pdf

16. Koło - rys_16.pdf
17. Końcówka przegubu Cardana koło element 2 - rys_17.pdf
18. Końcówka przegubu Cardana dyferencjał element 1 - rys_18.pdf
19. Wał napędowy silnik-dyferencjał - rys_19.pdf
20. Wał napędowy dyferencjał-lewe koło - rys_20.pdf
21. Wał napędowy dyferencjał-prawe koło - rys_21.pdf
22. Piasta koła - rys_22.pdf
23. Mocowanie zawieszenia tył - rys_23.pdf
24. Instalacja elektryczna - instalacja-elektryczna.pdf
25. Programy do sterowania - program.py
26. Program odpowiadający za przetwarzanie obrazu - object-ident.py

Źródła

- https://singleboardbytes.com/647/install-opencv-raspberry-pi-4.htm
- https://opencv.org/introduction-to-the-coco-dataset/
- https://opencv.org/
- https://cocodataset.org/#home