

Autonomiczny Pojazd

Wykonawcy:
Konstanty Odważny,
Jędrzej Szczerbal,
Paweł Chumski

Promotor:
dr hab. inż. Tomasz Pajchrowski

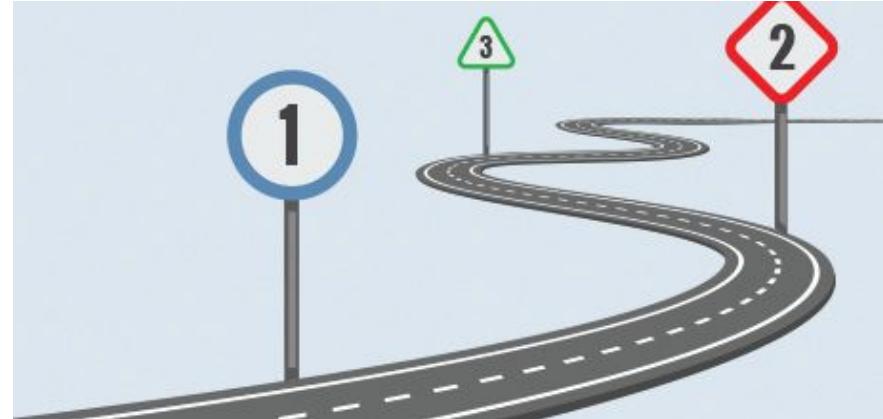
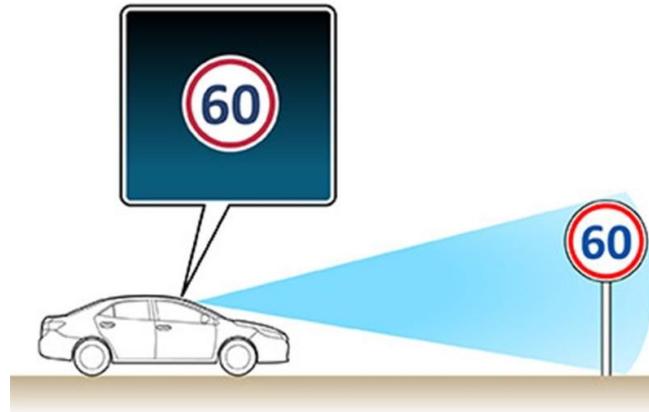


Współpraca z kołem naukowym RAI



Koncepcja projektu

- Stworzenie pojazdu poruszającego się po przygotowanej planszy, który będzie potrafił rozpoznać znaki i odpowiednio się do nich zastosować
- Możliwość sterowania zdalnego za pomocą pada



Zastosowane technologie



Flask



PyTorch



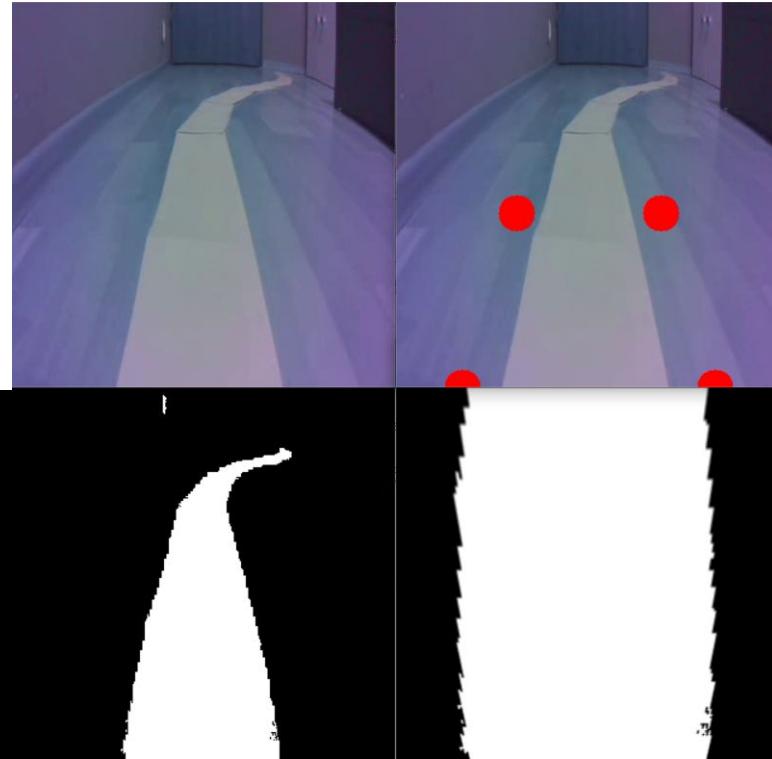
YOLOv5



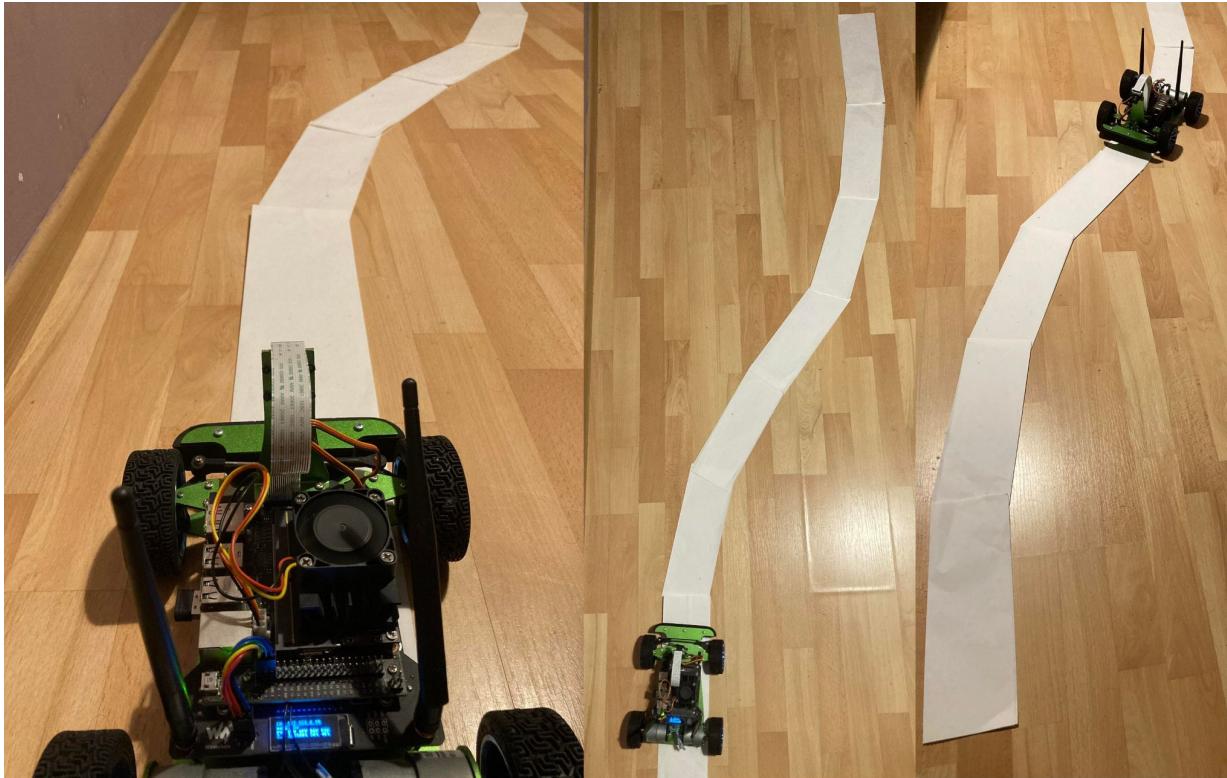
KiCad

Śledzenie linii

- Wyodrębnienie linii
- Poprawa perspektywy
- Wykrywanie zakrętów
- Optymalizacja
- Implementacja sterowania



Testowanie śledzenia linii



YOLO - You Only Look Once

1. Lokalizacja obiektu
2. Klasifikacja obiektu
3. Rozpoznanie obiektu



Klasifikacja

Lokalizacja z klasyfikacją

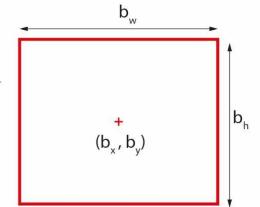


“Stop”

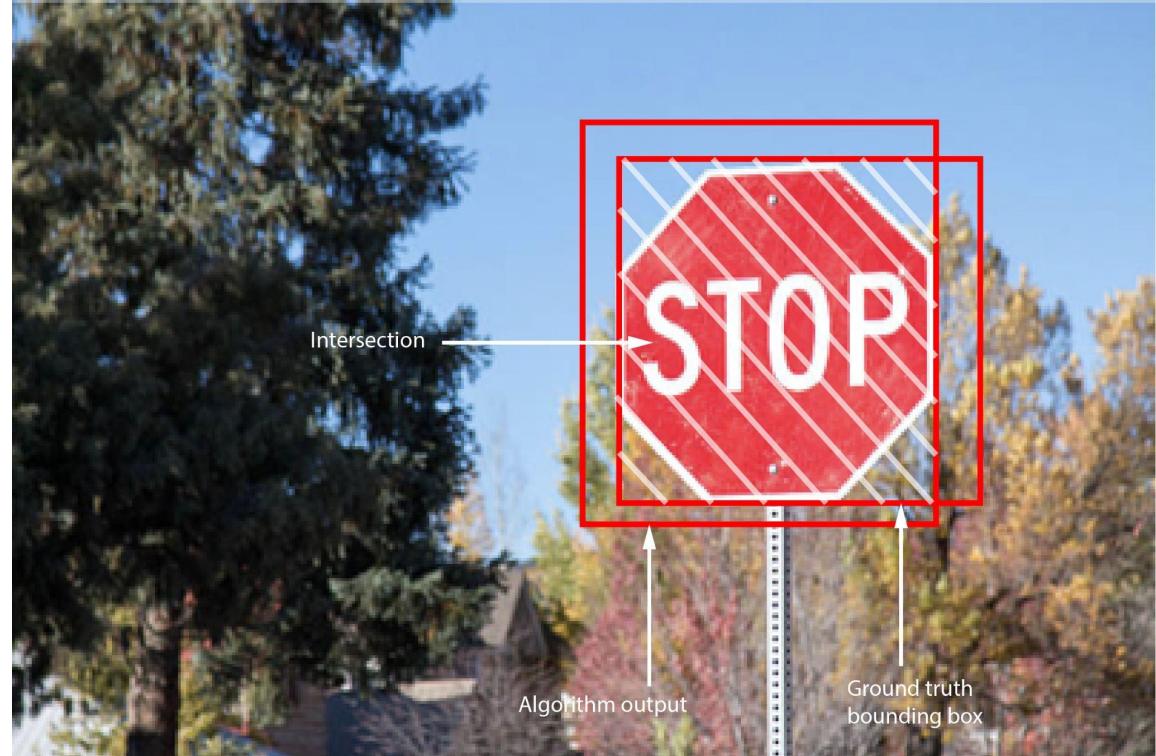
“Stop”



$$y = (p_c, b_x, b_y, b_h, b_w, c)$$



Yolo v5



$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

TP = True positive

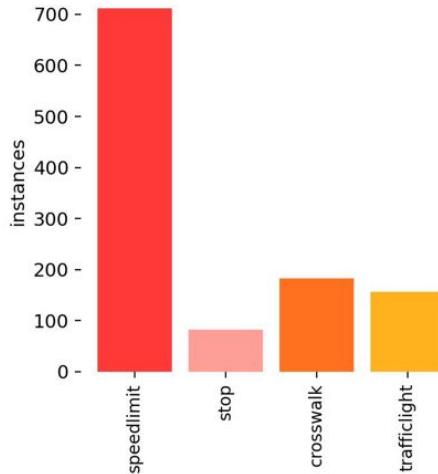
TN = True negative

FP = False positive

FN = False negative

Wybór zbioru danych

- Wybór danych z platformy Kaggle
- Zwrócenie uwagi na ilość klas oraz zbalansowanie danych



About Dataset



Usability ⓘ
8.75

License
CC0: Public Domain

Expected update frequency
Never

About this Dataset

This dataset contains 877 images of **4 distinct classes** for the objective of road sign detection.

Bounding box annotations are provided in the PASCAL VOC format

The classes are:

- Traffic Light;
- Stop;
- Speedlimit;
- Crosswalk.

Przygotowanie zbioru uczącego

- Podział na obrazy oraz pliki opisujące (labels) w formacie .txt, gdzie każdy plik jest zapisany w postaci wzorca:

```
[Class Number] [center in x] [center in y] [Width] [Height] 2 0.7378676470588236 0.5125 0.030147058823529412 0.055
```

- Podział zbioru na zbiór treningowy oraz walidacyjny, w proporcji 90% - treningowy, 10% - walidacyjny
- Utworzenie pliku konfiguracyjnego .yaml
- Wybór wcześniej wytrenowanego modelu Yolov5: yolov5x,yolov5l, yolov5m, yolov5s, yolov5n

Nano	Small	Medium	Large	XLarge
YOLOv5n	YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
4 MB _{FP16} 6.3 ms _{V100} 28.4 mAP _{coco}	14 MB _{FP16} 6.4 ms _{V100} 37.2 mAP _{coco}	41 MB _{FP16} 8.2 ms _{V100} 45.2 mAP _{coco}	89 MB _{FP16} 10.1 ms _{V100} 48.8 mAP _{coco}	166 MB _{FP16} 12.1 ms _{V100} 50.7 mAP _{coco}

```
1 train: ../data/images/training/
2 val: ../data/images/validation/
3
4 # number of classes
5 nc: 4
6
7 # class names
8 names: ['speedlimit', 'stop', 'crosswalk', 'trafficlight']
```

Trenowanie modelu

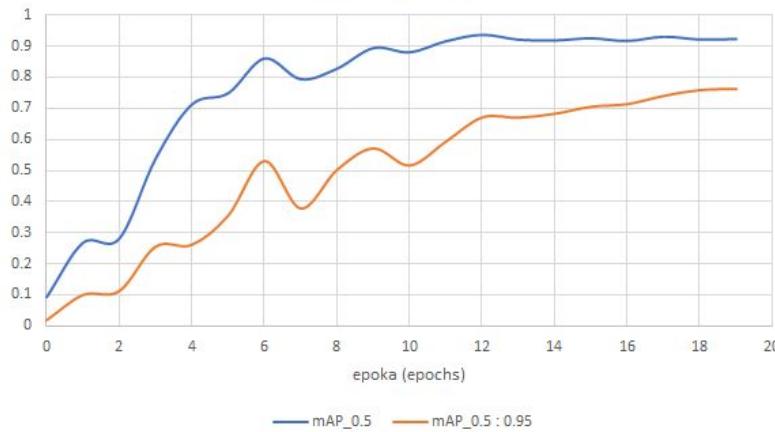
- Wybór wielkości obrazów wejściowych (standardowo 640x640 px)
- Wybór precyzji: FP32, FP16 lub INT8
- Wybór ilości epok szkolenia (ang. epochs) oraz ilości obrazów pobieranych za jednym razem do trenowania (ang. batch size)
- Walidacja - testowanie w trakcie trenowania
- Trenowanie z użyciem GPU
- Ocena jakości modelu, wskaźniki jakości, wykresy

```
Validating runs\train\exp18\weights\best.pt...
Fusing layers...
Model summary: 157 layers, 1764577 parameters, 0 gradients, 4.1 GFLOPs
  Class   Images Instances   P     R   mAP50   mAP50-95: 100% | [ ] | 2/2 00:01
    all      93     119   0.979   0.886   0.923   0.763
  speedlimit  93      72     1   0.969   0.995   0.865
    stop      93      10   0.963     1   0.995   0.886
  crosswalk  93      18   0.974   0.889   0.952   0.784
  trafficlight  93      19   0.98     0.684   0.75   0.517
Results saved to runs\train\exp18
```

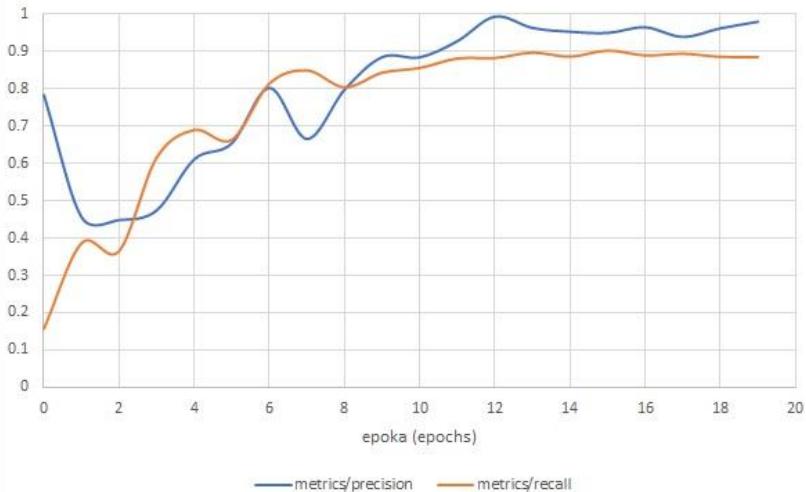
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
0/19	3.77G	0.1052	0.02791	0.04064	51	640: 100% [] 26/26 00:23
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% [] 2/2 00:01
	all	93	119	0.783	0.156	0.0919 0.0183
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
1/19	4.59G	0.07361	0.0238	0.02279	58	640: 100% [] 26/26 00:20
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% [] 2/2 00:01
	all	93	119	0.455	0.386	0.266 0.0996
Epoch	GPU_mem	box_loss	obj_loss	cls_loss	Instances	Size
2/19	4.59G	0.06968	0.02027	0.01931	76	640: 100% [] 26/26 00:20
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% [] 2/2 00:01
	all	93	119	0.447	0.365	0.28 0.112

Wybór modelu

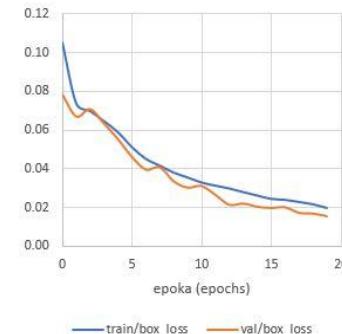
Wykres wyniku średniej wartości precyzyji względem kolejnych epok



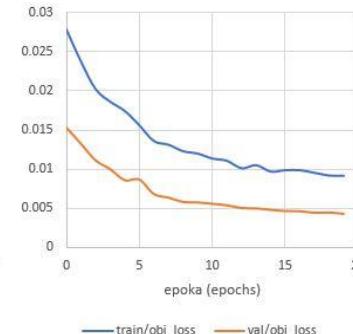
Funkcja metryk precision oraz recall



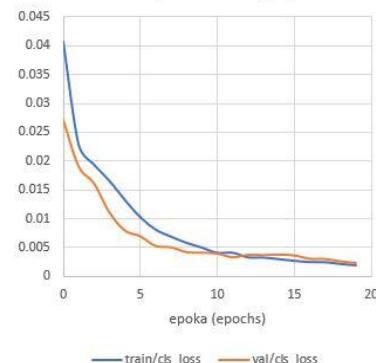
Funkcja strat ramek ograniczających (box)



Funkcja strat rozpoznawanych obiektów (obj)



Funkcja strat klas (cls)



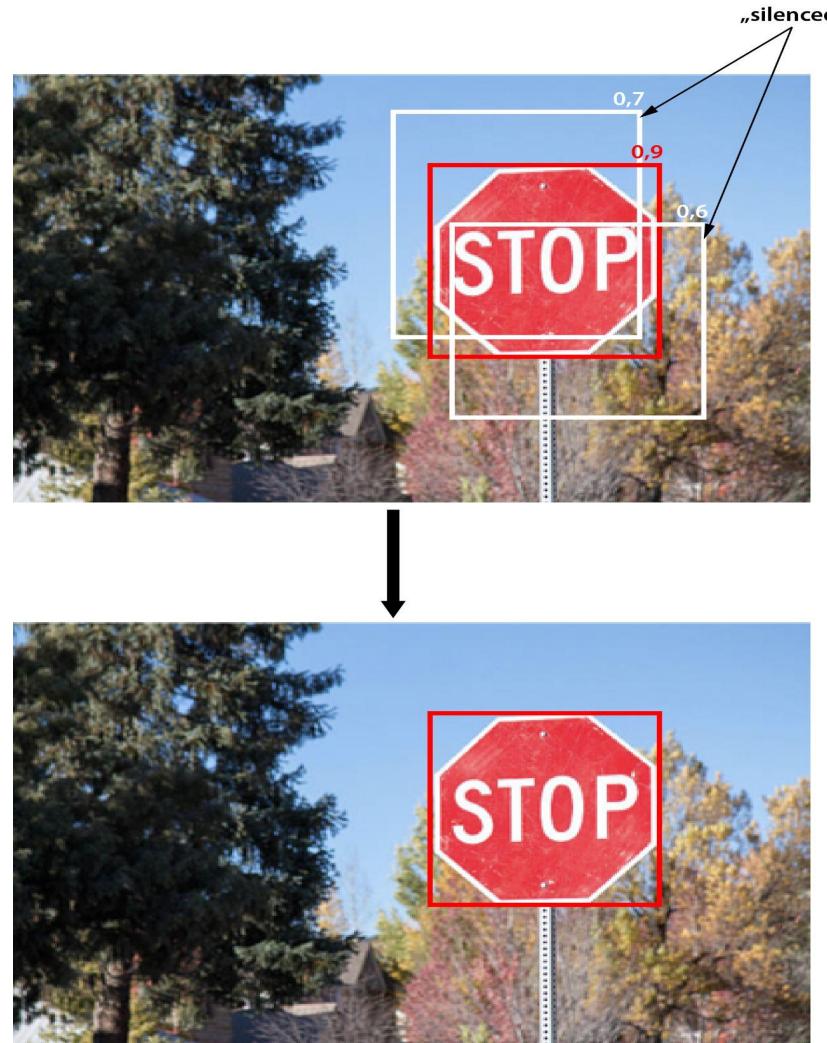
Testowanie na obrazach



Testowanie modelu

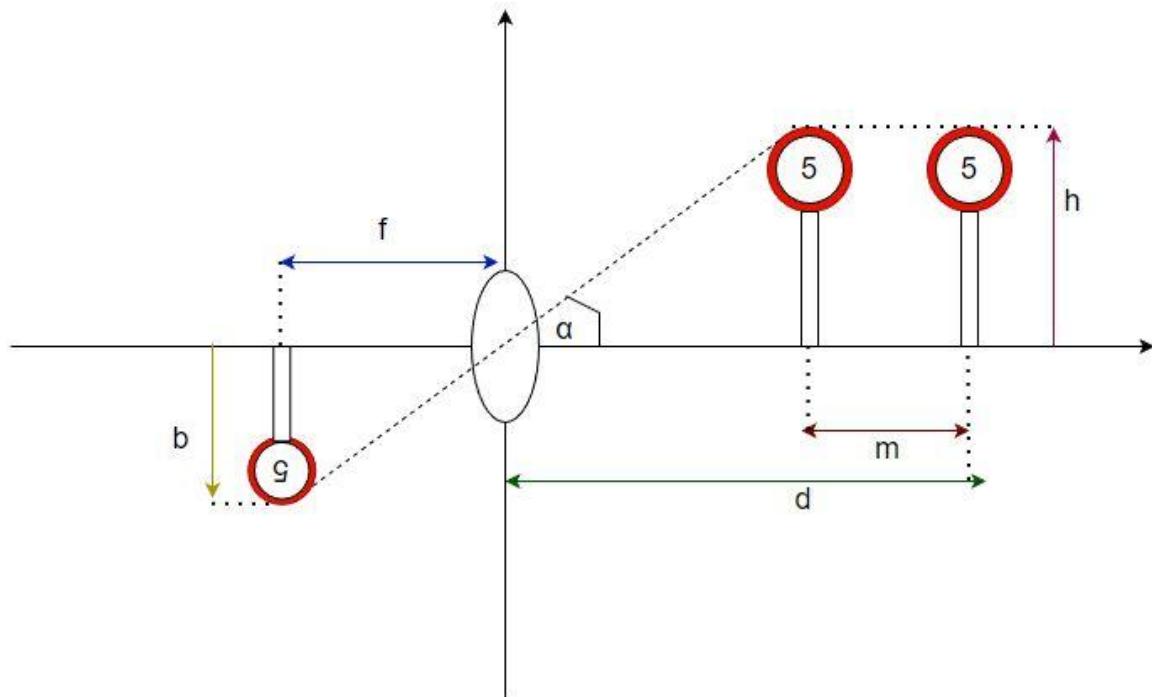
Kroki rozpoznawania obrazu:

- Przygotowanie obrazu (image preprocess) - zmiana wielkości, normalizacja
- Załadowanie modelu sieci neuronowej
- Predykcja - Próg ufności, NMS - non-maximum suppression
- Zaznaczenie rozpoznanego obiektu na obrazie za pomocą ramki ograniczającej (ang. bounding box)



Estymacja odległości od obiektu

1. Odczytanie wartość ogniskowej obiektywu kamery (ang. focal length).
2. Obliczanie odległości na podstawie ramek znalezionych w fazie rozpoznawania obiektów.



Interfejs użytkownika

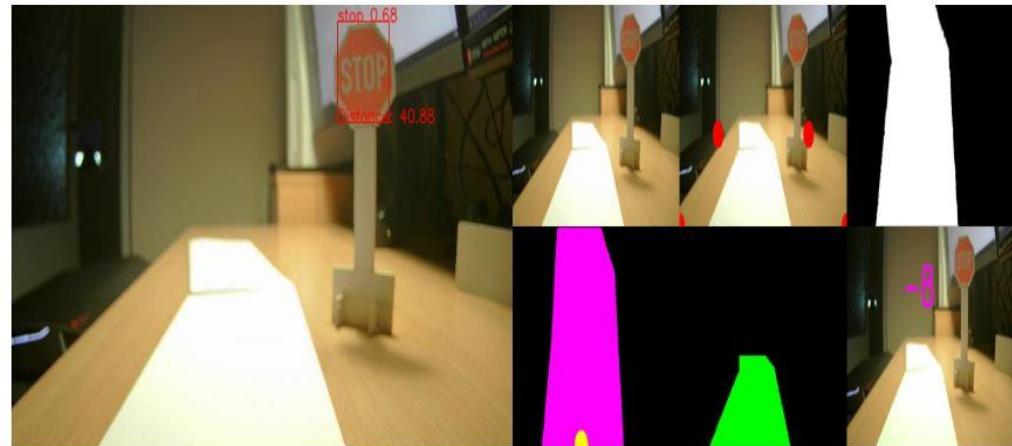
Strona klienta REST, stworzona za pomocą biblioteki Flask

Zalety:

- Łatwość integracji z resztą zadań
- Całość projektu w języku Python
- Możliwość obserwacji obrazu z kamery na żywo



Detection live stream



Własny kontroler

Założenia wstępne:

- komunikacja BT oraz Wifi
- podstawowe sterowanie pojazdem
- przełączanie trybu ręcznego oraz automatycznego
- zasilanie z opcją ładowania
- elastyczność pod przyszłe funkcjonalności



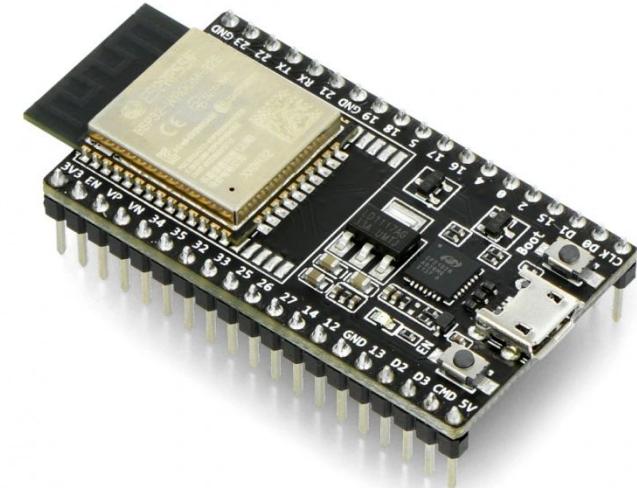
Mikrokontroler

Wybrany model: ESP32-WROOM-32D

- procesor dual-core Xtensa 32-bit LX6 - 240MHz
- 448kB pamięci ROM oraz 520kB pamięci SRAM oraz 4MB pamięci flash
- obsługa magistral cyfrowych, m.in.: I2C, **SPI**, I2S, PWM, UART, IR, CAN
- Standard BT: Bluetooth v4.2 BR/EDR oraz BLE
- Protokoły Wi-Fi: 802.11 b/g/n/d/e/i/k/r
- przetwornik ADC oraz DAC

Wybór:

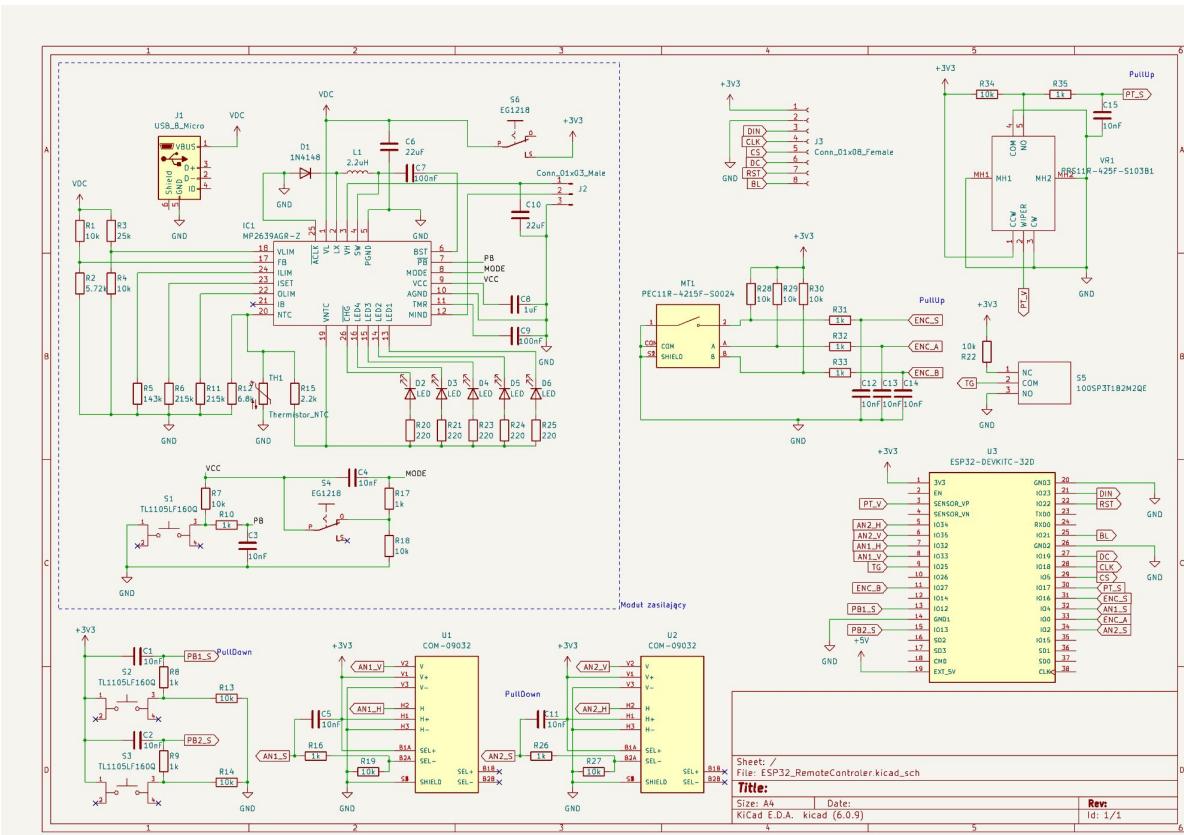
- Wielofunkcyjność
- Duża moc obliczeniowa
- Obsługa komunikacji bezprzewodowej
- Wielowątkowość
- Przystępna cena



Schemat ideowy

Podstawowe elementy:

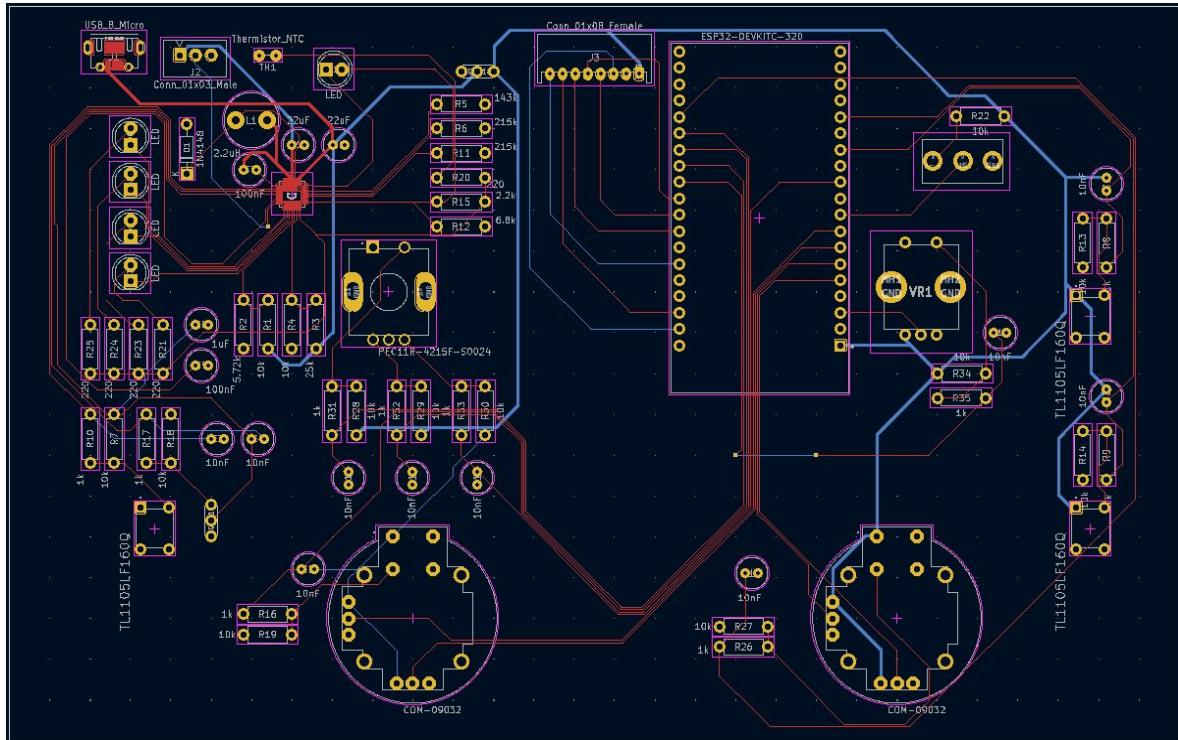
- wyświetlacz LCD TFT(SPI)
- joystick
- enkoder inkrementalny
- potencjometr
- przełącznik
- przyciski
- układ zasilający MP2639AGR -Z



Wstępny projekt PCB

Wersja prototypowa zawiera głównie elementy THT. Ułożenie komponentów pasywnych oraz ich wartości wybrano uwzględniając zalecenia producenta.

Wybrano kompromis między ergonomią urządzenia oraz ograniczeniem miejsca i możliwości lutowniczych.



Układ ładowający

Problemy z układem ładowającym:

- Zaawansowanie układu
- Dobór parametrów komponentów
- Ułożenie komponentów
- Problemy montażowe

Potencjalne rozwiązania:

- Zastosowanie elementów SMD
- Skorzystanie z dostępnych na uczelni technik montażowych
- Wykorzystanie gotowych układów ładowających oraz zabezpieczających



Program

Podział kodu:

- deklaracja programu
- odczytywanie wejść
- przerwanie czasowe
- wysyłanie wiadomości

Struktura danych do wysłania:

```
typedef struct Data_Package{  
    byte AnlX;  
    byte AnlY;  
    byte AnlS;  
    byte PotV;  
    byte PotS;  
    byte EncV;  
    byte EncS;  
    byte Btn;  
};  
  
typedef union PacketToSend{  
    Data_Package myData;  
    byte packet[sizeof(Data_Package)];  
};  
PacketToSend myPacket;
```

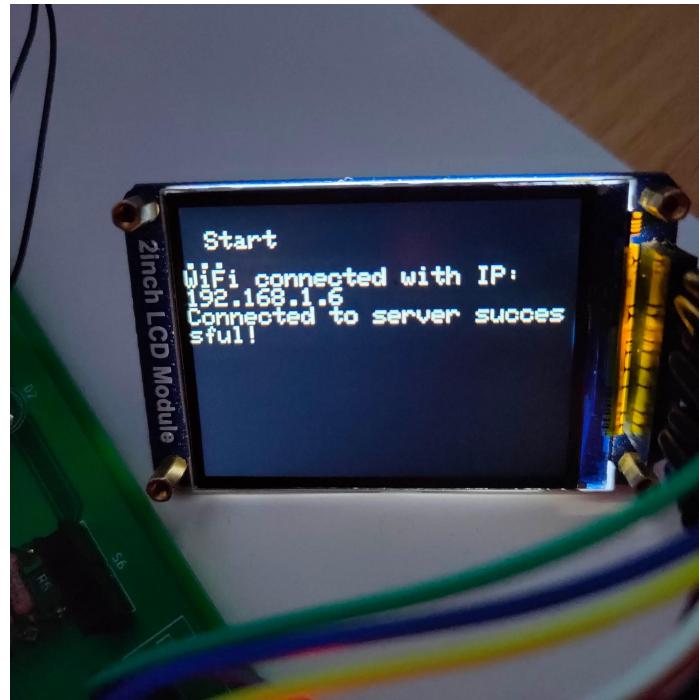
Połączenie z pojazdem

Brak problemów przy sterowaniu wyłącznie z kontrolera.

Przy uruchomieniu wszystkich elementów projektu brak reakcji pomimo połączenia z platformą.

Rozwiązanie:

- Optymalizacja algorytmów
- Zastosowanie wielowątkowości po stronie serwera



Nagranie



Przykładowe zdjęcie

Dziękujemy za uwagę