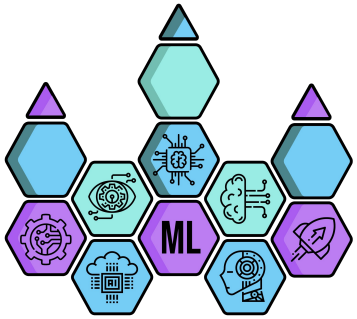
DATA
WORKSHOP



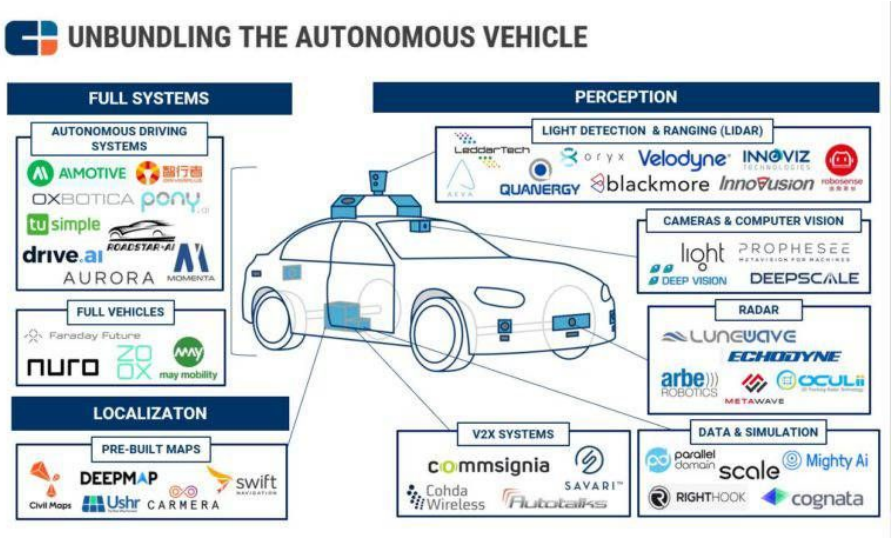
DW Poznań - #12 Projekt autonomicznego pojazdu

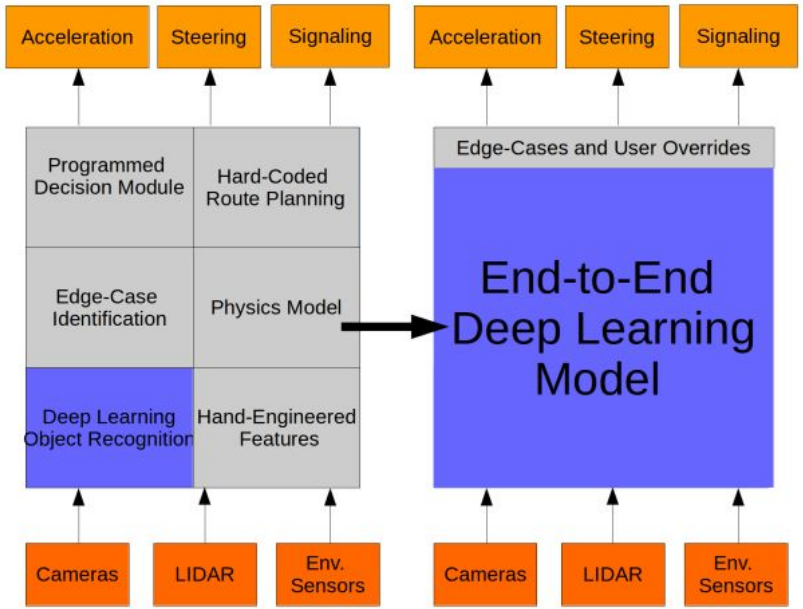
2020-07-09

Agenda

01. **Jak działają autonomiczne pojazdy?**
02. **Elegoo Robot Car**
03. **Części i jak się programuje Elegoo Robot Car?**
04. Jak zaimplementować Machine Learning?
05. **Jak zaimplementować model Tensorflow?**
06. **Wyzwania i jak będzie wyglądał projekt?**

Jak działają autonomiczne pojazdy?





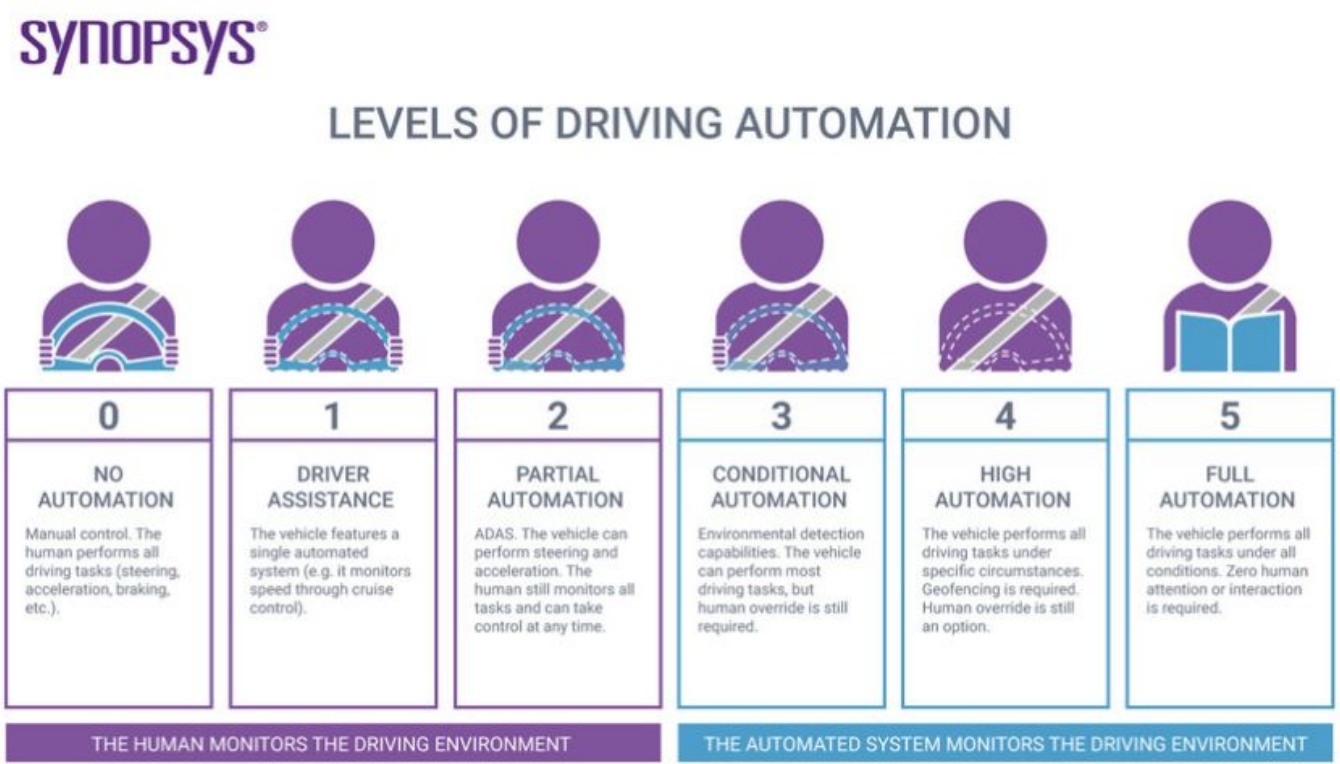
https://en.wikipedia.org/wiki/Vehicle-to-everything

https://www.cbinsights.com/research/startups-drive-auto-industry-disruption/

https://towardsdatascience.com/reinforcement-learning-from-grid-world-to-self-driving-cars-52bd3e647bc4

Jak działają autonomiczne pojazdy?

https://www.synopsys.com/automotive/what-is-autonomous-car.html



Elegoo Robot Car

- Kosztuje 60 GBP + przesyłka
- Chiński tańszy odpowiednik ARDUINO
- Płytka ELEGOO UNO R3 kompatybilna z ARDUINO UNO
- Bluetooth
- Wykrywacz linii (lewy, środkowy i prawy)
- Wykrywacz przeszkód (odległość)
- Bardzo łatwy w budowaniu

Czego nie znajdziemy?

- Bluetooth jest ograniczony tylko do aplikacji na iPhone (Bluetooth Low Energy)
- Brakuje kamery
- ELEGOO UNO R3, zamknięta technologia i ciężko doszukać się informacji czy są jeszcze jakieś ograniczenia

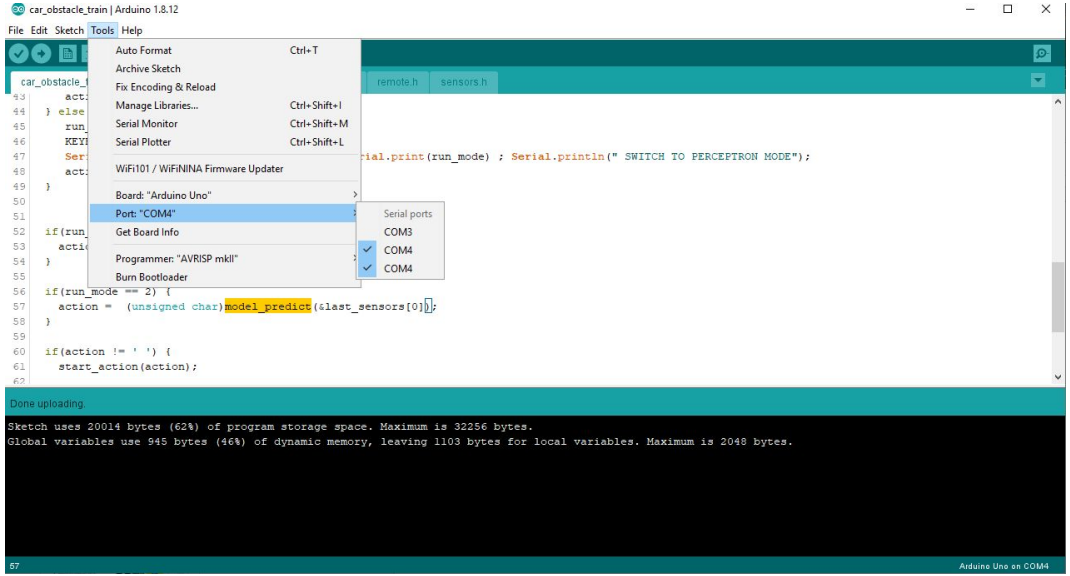
Z czego się składa i jak go programować

- Główna płytka i procesor ELEGOO UNO R3
- Chip: ATmega16u2

	Arduino Uno	Raspberry Pi Model B
Price	$30	$35
Size	7.6 x 1.9 x 6.4 cm	8.6cm x 5.4cm x 1.7cm
Memory	0.002MB	512MB
Clock Speed	16 MHz	700 MHz
On Board Network	None	10/100 wired Ethernet RJ45
Multitasking	No	Yes
Input voltage	7 to 12 V	5 V
Flash	32KB	SD Card (2 to 16G)
USB	One, input only	Two, peripherals OK
Operating System	None	Linux distributions
Integrated Development Environment	Arduino	Scratch, IDLE, anything with Linux support

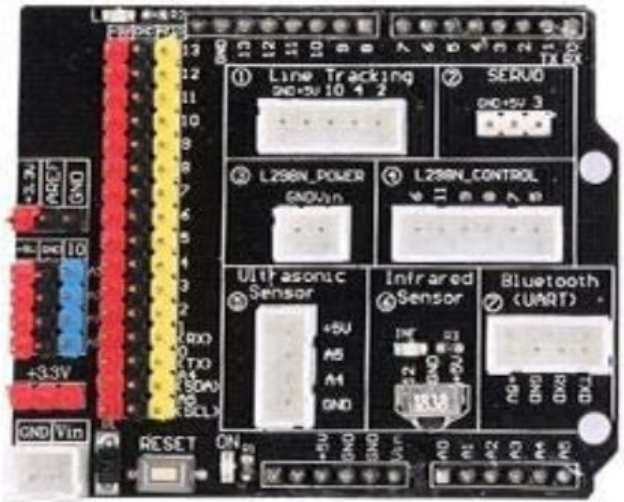
Programuje się go z ARDUINO IDE

- Podłączamy nasze urządzenie przez port USB (wykrywane na którymś COM)
- Piszemy w języku C++
- Są dwie metody **setup() -** uruchamiany przy starcie, oraz **loop() -** metoda działająca w ciągłej pętli
- Kompilujemy, wysyłamy na płytkę i już zaczyna działać
- Komunikujemy się przez porty 1 - 13, oraz analogowe A4, A5



Komunikacja przez porty:

- Dołączana jest dodatkowa płytka która służy tylko do podłączania innych elementów do sterowania.
- Są one podzielone w sekcje tak że łatwo je podłączyć




```
#define LED 13

setup() {
    pinMode(LED, OUTPUT);
}

looop() {
  digitalWrite(LED, 1);
  delay(1000);
  digitalWrite(LED, 0);

}
```

Komunikacja

- IRDA - podczerwień
- Wymagana biblioteka

- Bluetooth

- **Serial.read()**

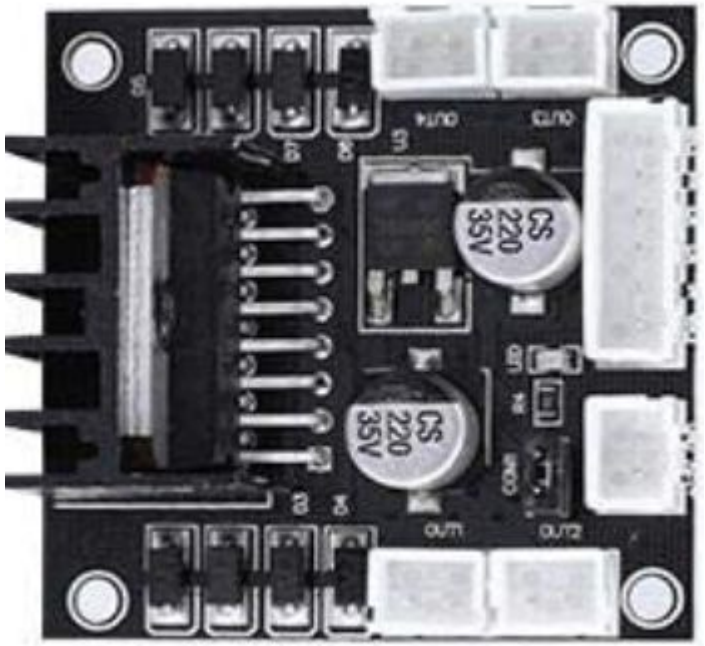
COM4

```
MODE: 2 SWITCH TO PERCEPTRON MODE

INPUT: 1.00, 1.00, 1.00, 59.00, 0.00,
MODEL: -0.44, -38.22, 0.01, -48.77, 0.79, MAX: 0.79 DIV: 1.75 = 4       U
1.00    0.00    1.00    46.00   0.00    0.00
INPUT: 1.00, 0.00, 1.00, 46.00, 0.00,
MODEL: -0.47, -29.75, 0.10, -37.99, 0.60, MAX: 0.60 DIV: 1.95 = 4       U
1.00    0.00    1.00    31.00   0.00    0.00
INPUT: 0.00, 0.00, 1.00, 31.00, 0.00,
MODEL: -1.02, -20.04, 0.58, -25.81, 0.26, MAX: 0.58 DIV: 1.92 = 2       L
1.00    1.00    1.00    26.00   0.00    0.00
INPUT: 1.00, 1.00, 1.00, 26.00, 0.00,
MODEL: 0.30, -17.28, -0.36, -21.81, 0.51, MAX: 0.51 DIV: 2.24 = 4       U
1.00    1.00    0.00    25.00   0.00    0.00
INPUT: 0.00, 1.00, 0.00, 25.00, 0.00,
MODEL: -0.36, -16.10, 0.13, -20.54, 0.32, MAX: 0.32 DIV: 2.33 = 4       U
1.00    1.00    1.00    25.00   0.00    0.00
INPUT: 1.00, 1.00, 1.00, 25.00, 0.00,
MODEL: 0.33, -16.65, -0.37, -20.99, 0.50, MAX: 0.50 DIV: 2.26 = 4       U
1.00    1.00    1.00    25.00   0.00    0.00
MODE: 0 SWITCH TO RUN MODE

Left    Middle  Right   Distance        DistanceLeft    DistanceRight   Action
1.00    1.00    1.00    38.00   0.00    0.00
```

Autoscroll Show timestamp

Silnik

```
void forward() {

  analogWrite(ENA, carSpeed);

  analogWrite(ENB, carSpeed);

  digitalWrite(IN1, HIGH);

  digitalWrite(IN2, LOW);

  digitalWrite(IN3, LOW);

  digitalWrite(IN4, HIGH);

}
```



SERVO

- Służy do obracania naszego czujnika dźwięku
- Wymagana biblioteka

DŹWIĘKOWY CZUJNIK ODLEGŁOŚCI

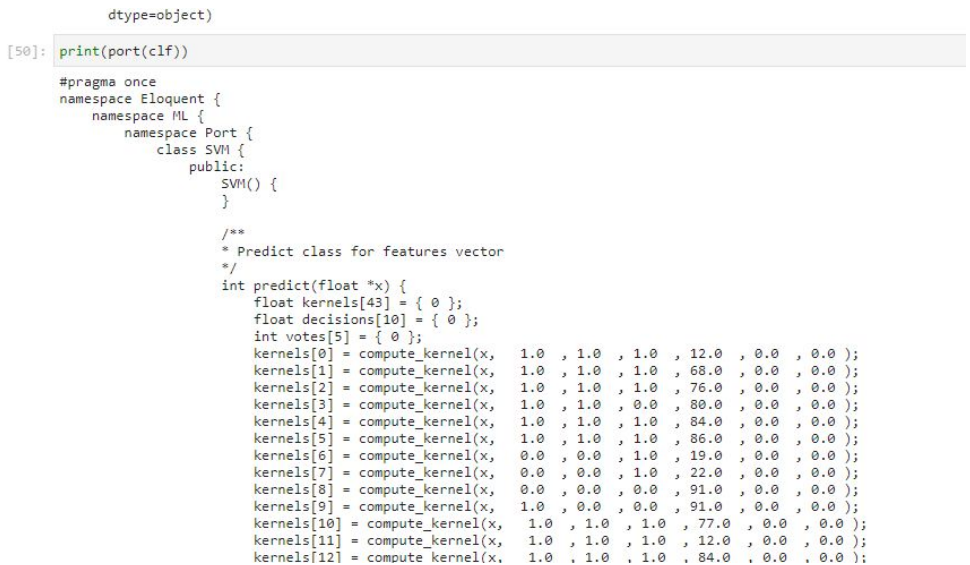
- Trzeba przeliczyć na cm

CZUJNIK TAŚMY

- 1 jeżeli biały
- 0 jeżeli czarny
- R, M, L

SVM

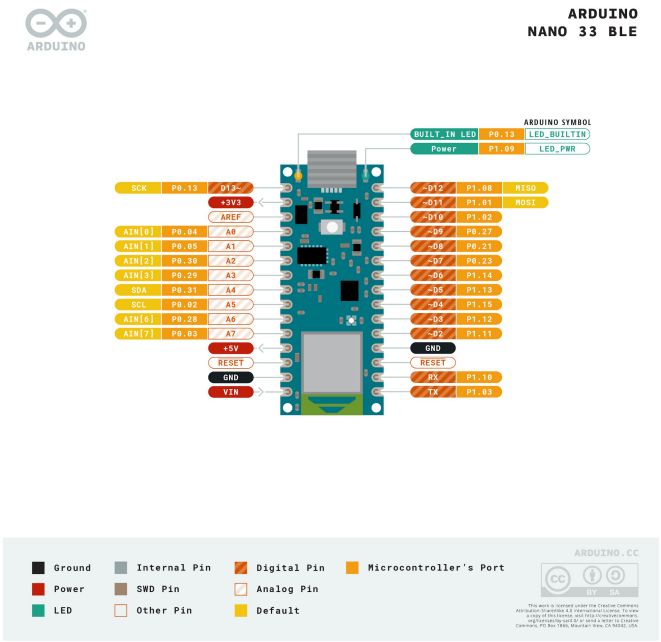
1. Generuje automatycznie kod w C++ do naszego modelu.
2. Etapy
    3. Zbieramy dane z naszego zachowania
    4. Zapisujemy do pliku .csv
    5. Uczymy model, i drukujemy kod w C++ przy pomocy **micromlgen** i JINJA



```
dtype=object)

[50]: print(port(clf))

#pragma once
namespace Eloquent {
    namespace ML {
        namespace Port {
            class SVM {
                public:
                    SVM() {
                    }

                    /**
                    * Predict class for features vector
                    */
                    int predict(float *x) {
                        float kernels[43] = { 0 };
                        float decisions[10] = { 0 };
                        int votes[5] = { 0 };
                        kernels[0] = compute_kernel(x,   1.0  , 1.0  , 1.0  , 12.0  , 0.0  , 0.0 );
                        kernels[1] = compute_kernel(x,   1.0  , 1.0  , 1.0  , 68.0  , 0.0  , 0.0 );
                        kernels[2] = compute_kernel(x,   1.0  , 1.0  , 1.0  , 76.0  , 0.0  , 0.0 );
                        kernels[3] = compute_kernel(x,   1.0  , 1.0  , 0.0  , 80.0  , 0.0  , 0.0 );
                        kernels[4] = compute_kernel(x,   1.0  , 1.0  , 1.0  , 84.0  , 0.0  , 0.0 );
                        kernels[5] = compute_kernel(x,   1.0  , 1.0  , 1.0  , 86.0  , 0.0  , 0.0 );
                        kernels[6] = compute_kernel(x,   0.0  , 0.0  , 1.0  , 19.0  , 0.0  , 0.0 );
                        kernels[7] = compute_kernel(x,   0.0  , 0.0  , 1.0  , 22.0  , 0.0  , 0.0 );
                        kernels[8] = compute_kernel(x,   0.0  , 0.0  , 0.0  , 91.0  , 0.0  , 0.0 );
                        kernels[9] = compute_kernel(x,   1.0  , 0.0  , 0.0  , 91.0  , 0.0  , 0.0 );
                        kernels[10] = compute_kernel(x,   1.0  , 1.0  , 1.0  , 77.0  , 0.0  , 0.0 );
                        kernels[11] = compute_kernel(x,   1.0  , 1.0  , 1.0  , 12.0  , 0.0  , 0.0 );
                        kernels[12] = compute_kernel(x,   1.0  , 1.0  , 1.0  , 84.0  , 0.0  , 0.0 );
```

https://github.com/eloquentarduino/micromlgen

Tensorflow Lite

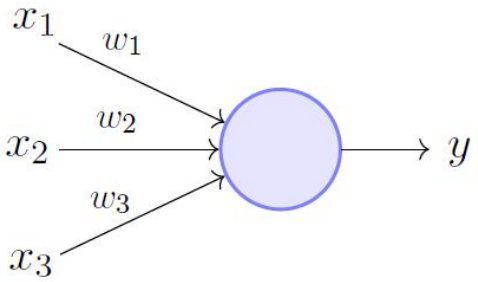


1. Tensorflow na słabsze urządzenia (mikrokontrolery)
2. Tylko ograniczone urządzenia i nie ma tam **ARDUINO UNO**
3. Wersja **BETA**
4. Obslugiwany np: ARDUINO NANO 33 BLE

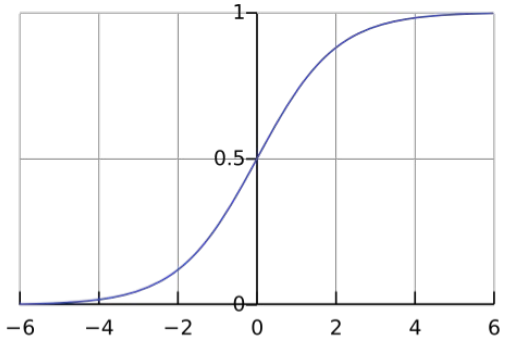
https://blog.arduino.cc/category/arduino/nano-33-ble-sense/

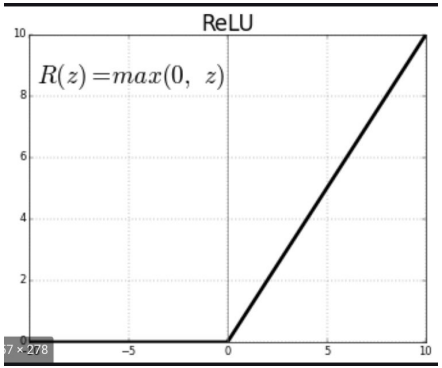
https://www.tensorflow.org/lite

Napisać własny model i jego funkcje, a wagi eksportujemy z modelu



Perceptron Model (Minsky-Papert in 1969)





$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

https://hackaday.com/2019/06/30/blisteringly-fast-machine-learning-on-an-arduino-uno/

Jakie będzie wyglądał projekt?

- **Zmieniamy na Rasberry PI i kamerę**
- **Tworzym Symulator albo korzystamy z gotowego**
- **Zbieramy dane testowe**
- **Programujemy na symulatorze a następnie jak będzie prawidłowo przenosimy na C++ i nasz samochód by móc przetestować.**
- **Tworzymy aplikacje na ReactNative która przy pomocy bluetooth komunikuje się i włącza parkowanie**

https://github.com/microsoft/AirSim

https://github.com/dataworkshop/dw-poznan-project

Kolejne kroki

- Spotkanie za 2 tygodnie
- Szukamy materiałów odnośnie technologii

https://github.com/dataworkshop/dw-poznan-project

Dziękuję