Roboty przyszłości

Autonomiczna fabryka i inteligentny robot inspekcyjny

European Robotics Week 2026

Projekt Erasmus+ KA220-SCH (2025-1-PL01-KA220-SCH-000352256)

dr inż. Angelika Tefelska
Politechnika Warszawska, Wydział Fizyki

ERGO

Co-funded by
the European Union

Dlaczego warto uczyć się robotyki?

Robotyka łączy wiele dziedzin

Zastosowania

- produkcja i inteligentne magazyny;
- diagnostyka i wsparcie personelu;
- transport, rolnictwo, eksploracja;
- pomoc w codziennych zadaniach.

Umiejętności

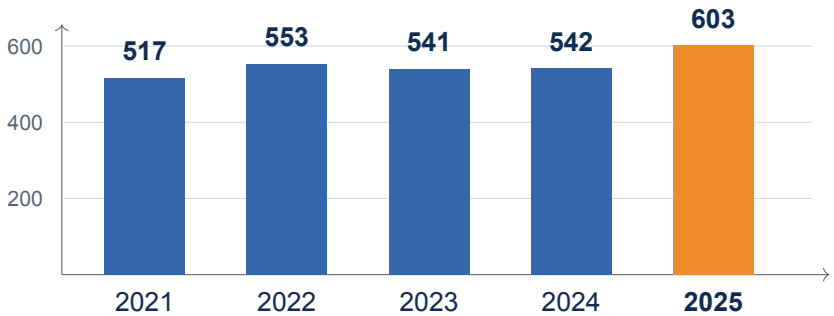
- programowanie i matematyka;
- algorytmy i podejmowanie decyzji;
- czujniki i automatyka;
- testowanie i analiza błędów.

Robot nie „myśli” magicznie: działa według algorytmu, danych i ograniczeń.

Robotyka na świecie: dane zamiast prognoz

Nowe instalacje robotów przemysłowych

Świat, w tysiącach sztuk rocznie



Rok 2025: ok. 603 tys. nowych instalacji (+11% r/r); ponad 5 mln robotów przemysłowych w użyciu.

Źródło: IFR, *World Robotics 2026 – Industrial Robots*, instalacje 2021–2025. Komunikat IFR, 24.09.2026.

Open Roberta Lab: bloczki, ale prawdziwe algorytmy

Dlaczego zaczynamy od programowania wizualnego?

Bezpłatnie i w przeglądarce

Szybki start

- Programujemy w języku wizualnym **NEPO**.
- Testujemy robota w symulatorze, bez sprzętu.
- Skupiamy się na **działaniu algorytmu**, a nie literówkach w kodzie.

To nie tylko zabawa

- pętle i warunki;
- zmienne i odczyt czujników;
- funkcje i podział programu;
- nawigacja i maszyny stanów.



Tę samą logikę można później zapisać w Pythonie, C++ czy innym języku.

Dzisiaj będziecie pracować jako zespoły inżynierskie

Misja 1 · Robotino

Wirtualny robot logistyczny: dostawy do stanowisk, reakcja na przeszkodę i wybór objazdu.

Misja 2 · Edison V3

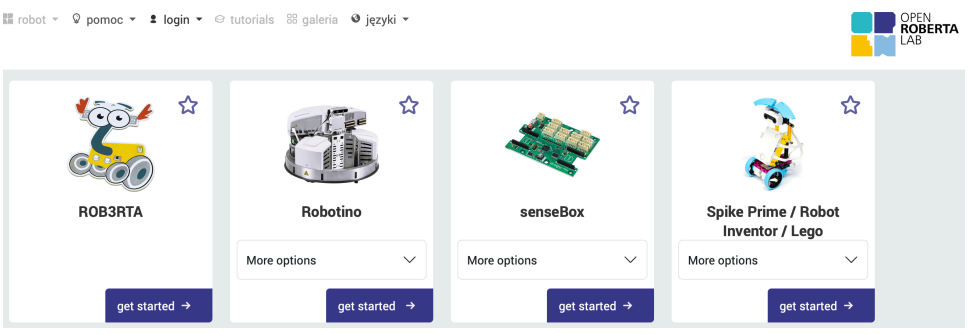
Robot inspekcyjny: sterowanie ruchem, dwa czujniki przeszkód IR, sygnalizacja LED oraz pierwsze eksperymenty z xNN.

Projektujcie, testujcie i poprawiajcie algorytm!

150 minut pracy · dwa roboty · kilka sposobów rozwiązania problemu

Open Roberta Lab: pierwsze kroki

1. Otwórz `lab.open-roberta.org`.
2. Pod najpopularniejszymi robotami znajdziesz przycisk *All systems and filter options*. Wybierz go.
3. Wybierz typ robota: **Robotino**.

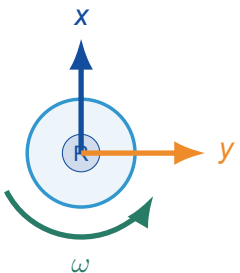


Robotino: mobilny robot przemysłowy

Misja 1: transport w inteligentnej fabryce

Festo Didactic

- Robotino to **platforma mobilna** wykorzystywana w edukacji i automatyce.
- Jego **napęd wielokierunkowy** pozwala zmieniać kierunek jazdy bez konieczności obracania całego robota.
- Bloki `drive`, `turn` i czujniki umożliwiają symulowanie zadań logistycznych.

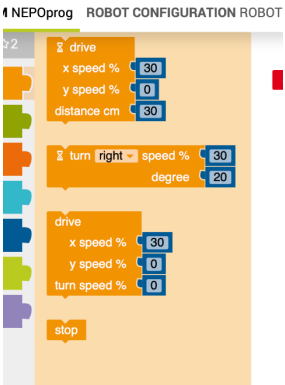


Na rysunku: *x* – jazda w przód/tył; *y* – ruch boczny; *ω* – obrót wokół własnej osi. Kierunki odnoszą się do lokalnego układu robota.

Robotino · jakie bloki mamy?

Ruch robota

Bloki widoczne na zrzucie z Open Roberta



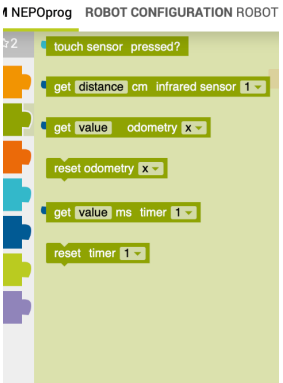
- `drive`: *x* speed %, *y* speed %, distance [cm].
- `turn`: kierunek, speed %, angle [deg].
- `drive` ciągły: *x* speed, *y* speed, turn speed.
- `stop`: zakończenie ruchu.

Wybór bloku zależy od zadania: ustalona odległość czy ciągłe sterowanie.

Robotino · co można odczytać?

Czujniki i pomiary

Bloki dostępne w konfiguracji Robotino



- `touch sensor pressed?` – zwraca **prawdę/fałsz**: czy czujnik dotyku został naciśnięty.
- `infrared sensor` – aktywny czujnik podczerwieni: **mierzy odległość** od obiektu [cm] w wybranym kierunku.
- `odometry x/y/angle` – **szacuje zmianę położenia i orientacji** na podstawie ruchu napędu.
- `timer` – licznik **czasu** [ms]; reset pozwala zacząć pomiar od nowa.

Czujnik IR a odometria – jaka jest różnica?

Dwa różne źródła informacji

Wykrywanie otoczenia i szacowanie własnego ruchu

Podczerwień (IR)

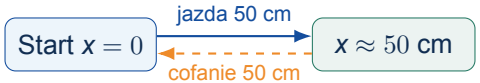
Odpowiada na pytanie: **„Jak daleko jest przeszkoda?”**

Wynik: odległość w cm. Zależy m.in. od kierunku pomiaru i obiektu.

Odometria

Odpowiada na pytanie: **„Jak zmieniło się położenie robota?”**

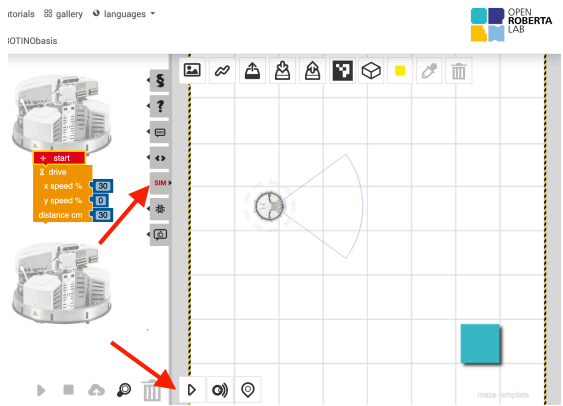
Wynik: przybliżone x, y i kąt θ.



Uwaga: po powrocie do startu odometria x może ponownie wskazać ok. 0 cm, mimo że robot *łącznie* przejechał 100 cm.

R1 · pierwszy przejazd Robotino

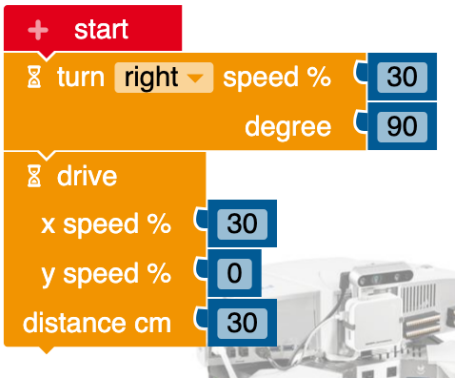
1. Utwórz program przejazdu **30 cm do przodu**.
2. Zacznij od parametrów: `x speed = 30%`, `y speed = 0%`, `distance = 30 cm`.
3. Uruchom symulację i obserwuj rezultat.
4. Zmień prędkość na `50%` `x speed`. Czy zmieniła się odległość, czas przejazdu, czy oba parametry?
5. Nad przyciskiem **SIM** znajduje się przycisk <>. Wybierz go i zobacz jak wygląda kod w Pythonie ruchu robota.



R1 · ruch boczny i obrót

Eksperyment porównawczy

1. Zaplanuj przesunięcie o **30 cm w bok** (oś y) bez obrotu robota. Podpowiedź: w funkcji `drive` ustaw tym razem `y speed` zamiast `x speed`.
2. Zmień wartość `y speed` na ujemną. Co się zmieniło? Zapamiętaj orientacje robota.
3. Teraz stwórz nowy program, w którym robot obróci się najpierw o 90° a następnie przejedzie 30 cm (patrz rysunek obok).
4. Porównaj **położenie końcowe** i **orientację** robota.

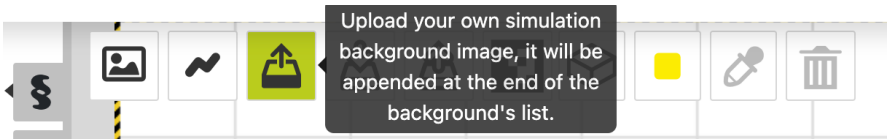


R2 · inteligentny transport w fabryce

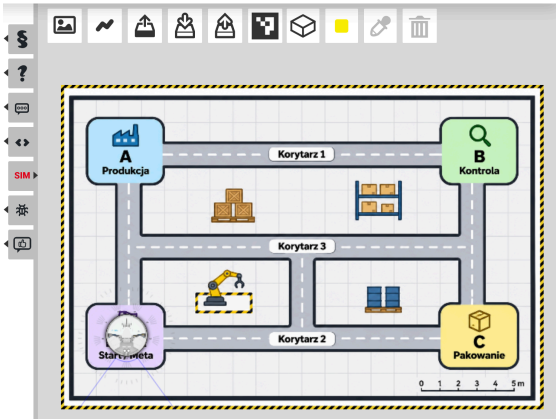
- Celem misji jest wykonanie dostawy w magazynie zgodnie ze schematem:

$$\text{BAZA} \rightarrow \text{A} \rightarrow \text{B} \rightarrow \text{C} \rightarrow \text{BAZA}.$$

- W tym celu pobierz mapę magazynu: **link** i wgraj ją w Open Roberta Lab:



R2 · inteligentny transport w fabryce

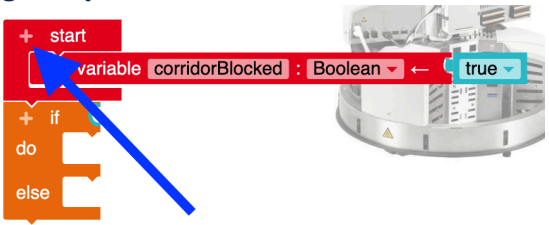


1. Ustal długości odcinków korytarzy.
2. Zaprogramuj trasę robota zgodnie ze schematem: BAZA → A → B → C → BAZA.
3. W każdym punkcie zatrzymaj się na 2 s. Podpowiedź: odnajdź funkcje `wait ms`.
4. Wróć do bazy i zakończ pracę.

Robot podejmuje decyzję

Uwaga: korytarz 1 jest zamknięty!

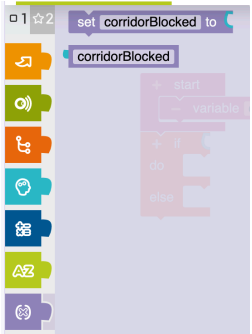
1. **Zaprojektuj trasę podstawową i trasę objazdową.**
2. **Utwórz zmienną logiczną `corridorBlocked`.**

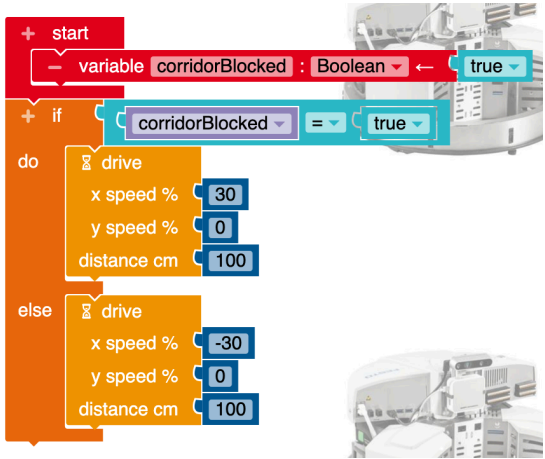


W tej wersji o awarii informuje zmienna. Nie jest to jeszcze automatyczne wykrywanie przeszkody.

R3 · awaria w fabryce

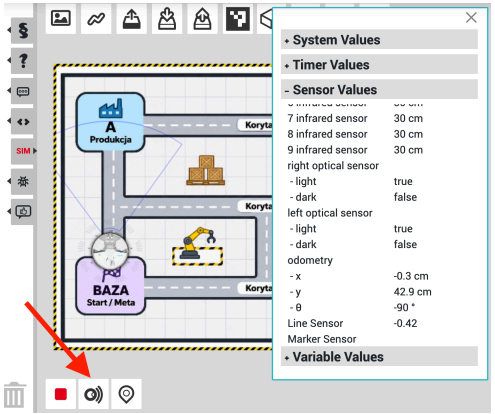
3. Użyj **warunku**, aby wybrać odpowiednią trasę.
4. Przetestuj obie wartości zmiennej: `false` i `true`.





Reagowanie na przeszkodę – przykład

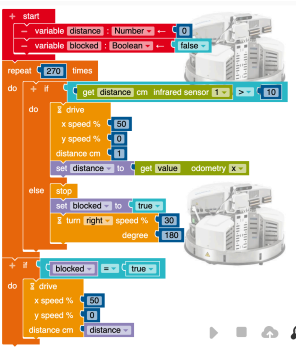
1. Czujnik **podczerwieni (IR)** mierzy odległość od przeszkody.
2. **Odometria** szacuje zmiany *x*, *y* i θ na podstawie ruchu robota. Nie jest to suma długości przebytej trasy.
3. Po prawej: podgląd odczytów z symulatora.
4. Na dole: dodawanie przeszkody do sceny **SIM**.





Reagowanie na przeszkodę - przykład

1. Obok pokazano przykładowy program jak odczytać za pomocą czujnika IR przeszkodę i jeśli jest ona w odległości mniejszej niż 10cm to zatrzymujemy robota i zawracamy.
2. Następnie wracamy się do początku korytarza.
3. Zwróć uwagę, że tym razem ruch robota jest w bloku `repeat` po to by na bieżąco sprawdzać stan czujników.



R4 - Reagowanie na przeszkodę

1. Zmodyfikuj swój program w taki sposób aby tym razem wybór trasy zależał faktycznie od przeszkody w korytarzu nr 1.
2. Odczytaj `distance [cm]` z wybranego czujnika IR.
3. Sprawdź, czy przeszkoda jest w korytarzu.
4. Jeśli tak to zawróć i wybierz drugą trasę.

Dwa roboty, dwa podejścia do sterowania

Robotino

- transport i planowanie tras;
- odometria oraz wykrywanie przeszkód;
- wybór objazdu na podstawie warunków.

Edison V3

- dwa niezależnie sterowane koła;
- reakcja na czujniki i regulacja skrętu;
- `if/else` oraz wprowadzenie do **xNN**.

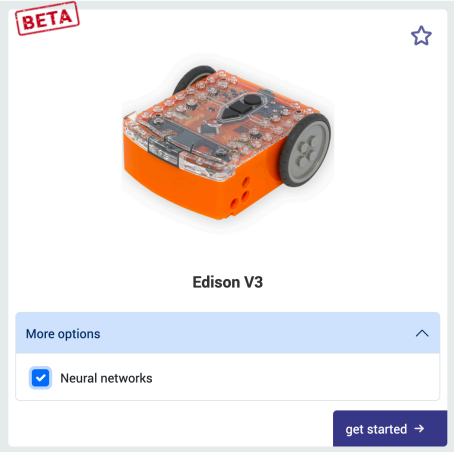
Ten sam problem można rozwiązać za pomocą prostych reguł albo modelu neuronowego.

Poznaj Edison V3

Misja 2: autonomiczny robot inspekcyjny

Microbric / Meet Edison

- **Edison V3** to niewielki robot mobilny z dwoma kołami napędowymi.
- W warsztatach wykorzystamy przede wszystkim **ruch robota**, **czujniki przeszkód IR**, **LED** oraz moduł **xNN**.
- W Open Roberta Lab można go **symulować**, testować algorytmy `if/else` i porównać je z prostą siecią neuronową.

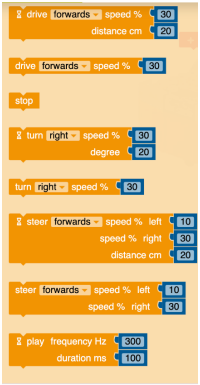


Nasza misja

Zbudować algorytm robota inspekcyjnego, który reaguje na przeszkody i potrafi podejmować decyzje na podstawie dwóch czujników.

Edison V3: sterowanie ruchem

Action

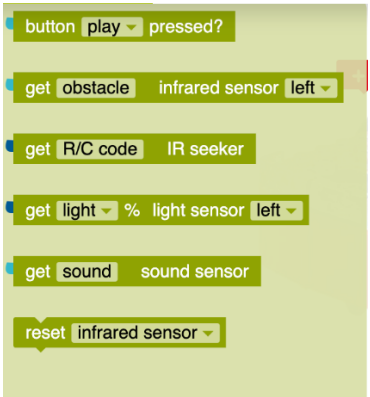


- `drive`: jazda prosto; opcjonalnie o zadaną odległość [cm];
- `turn`: obrót w miejscu o zadany kąt lub ruch obrotowy ciągły;
- `steer`: różne prędkości **lewego i prawego koła** – jazda po łuku;
- `stop`: zatrzymanie;
- `play` i `LED`: sygnalizacja działania programu.

Edison V3: czujniki i sygnały

Sensors

Co robot może odczytać w tej wersji zajęć?



- `get obstacle infrared sensor left/right`: czy czujnik IR wykrywa przeszkodę; **zwraca wartość logiczną** (tak/nie), a nie odległość w cm;
- `button play pressed?`: stan przycisku;
- `get R/C code IR seeker`: kod odebranego sygnału IR;
- `get light` i `get sound`: są widoczne w menu, ale **nie wykorzystujemy ich w zadaniach**, bo w testowanej symulacji nie działają przewidywalnie.

E1 · pierwszy przejazd Edison V3

1. Wybierz **Edison V3** i otwórz **SIM**.
2. Dodaj `drive forwards`, **30%**, **20 cm**.
3. Dodaj `turn right`, **30%**, **90°**.
4. Dodaj kolejny przejazd **20 cm**.
5. Na zakończenie włącz LED lub odtwórz sygnał `play frequency`.

Pseudokod

```
START
drive(20 cm)
turn(right, 90)
drive(20 cm)
LED(ON)
STOP
```

Test zaliczony, gdy: robot wykona dwie proste i obrót, a następnie zasygnalizuje zakończenie misji.

E2 · czym różnią się drive, turn i steer?

Blok	Co robi robot?	Eksperyment
`drive`	Oba koła jadą zgodnie, robot przemieszcza się po prostej.	Przejazd 20 cm.
`turn`	Koła wytwarzają obrót robota wokół jego osi.	Obrót o 90°.
`steer`	Koła jadą z różnymi prędkościami, powstaje **łuk**.	Lewe 10%, prawe 30%, 20 cm.

Wyzwanie

Pobierz i wczytaj mapę **link**. Następnie zaprogramuj tak ruch robota by pokonał zadany tor.

Wskazówka: najpierw robot jedzie na wprost, potem skręca, potem jedzie na wprost, potem jedzie po łuku, znów jedzie na wprost, skręca i jedzie na wprost.

E3 · Edison wykrywa przeszkodę

Czujnik → warunek → reakcja

Misja robota inspekcyjnego

Jedź do przodu. Jeśli czujnik **IR po lewej** wykryje przeszkodę, zatrzymaj się i skręć w prawo, zanim spróbujesz ponownie jechać.

1. Stwórz swoją planszę w SIM. Umieść w niej **rzeczywiste przeszkody**, nie tylko obraz na tle.
2. Odczytaj `get obstacle infrared sensor left` (uwaga: zwraca on wartość `True` bądź `False`).
3. W `repeat` użyj `if/else`; sprawdzaj czujnik podczas pracy.
4. W przypadku przeszkody: `stop`, następnie `turn right`.
5. W przeciwnym razie: `drive forwards`.

Podpowiedź: Do pracy w pętli wykorzystaj `drive` bez parametru odległości. Po manewrze robot może ponownie sprawdzić czujnik.

E3 · pseudokod i test bezpieczeństwa

Sprawdź więcej niż jeden przypadek

Wersja podstawowa i ambitniejsza

Wersja podstawowa

```
REPEAT:
  IF IR_left:
    stop()
    turn(right, 45)
  ELSE:
    drive(forwards)
END REPEAT
```

Test zaliczony, gdy: robot omija przeszkodę zgodnie z rozszerzonym algorytmem..

E4 · dwa czujniki IR – cztery decyzje

Rozszerzamy algorytm o lewy i prawy czujnik

Zasada działania

L	R	Reakcja
0	0	jedź prosto
1	0	skręć w prawo
0	1	skręć w lewo
1	1	stop

Tutaj `L` i `R` oznaczają informację z lewego i prawego czujnika przeszkody: `0` – brak przeszkody, `1` – przeszkoda wykryta.

Pseudokod

```
REPEAT:
  odczytaj L i R
  IF L AND R:
    stop()
  ELSE IF L:
    turn(right,45)
  ELSE IF R:
    turn(left,45)
  ELSE:
    drive(forwards)
END REPEAT
```

Test zaliczony, gdy: robot reaguje różnie na przeszkodę po lewej i po prawej stronie oraz zatrzymuje się przy blokadzie z obu stron.

E5 · jak działa prosty neuron?

Wprowadzenie do xNN

- **Wejścia** przekazują informacje do sieci, np. L i R z dwóch czujników IR.
- Każde wejście ma swoją **wagę**: dodatnia wzmacnia wpływ, ujemna działa przeciwnie.
- **Bias** przesuwa wynik o stałą wartość.
- **Aktywacja** określa, jak z sumy ważonej powstaje wyjście. Dla **Linear** wyjście jest równe obliczonej sumie.

Nasz przykład

$$S = 1 \cdot L + (-1) \cdot R + 0$$

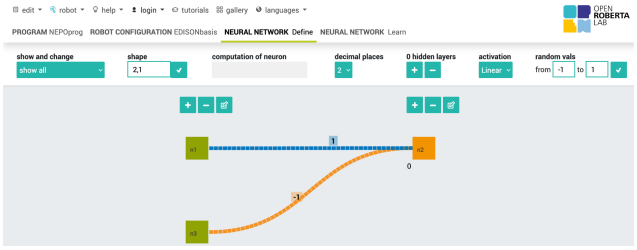
Interpretacja wyniku:

$S > 0$ – skręć w prawo
$S < 0$ – skręć w lewo
$S = 0$ – brak decyzji o skręcie

Sieć pomaga obliczyć wynik, ale nie zastępuje wszystkich zasad bezpieczeństwa. Dla $L = R = 1$ nadal potrzebujemy osobnego warunku `stop`.

E5 · konfiguracja sieci xNN w Open Roberta

Jak ustawić sieć dla naszego zadania?



Nazwy neuronów

W tej konfiguracji **n1** i **n3** są wejściami, a **n2** jest neuronem wyjściowym.

Konfiguracja sprawdzona w symulatorze

- **shape = 2,1**: dwa neurony wejściowe i jeden wyjściowy;
- **0 hidden layers**: brak warstw ukrytych, czyli najprostszy model;
- **activation = Linear**: wyjście równe sumie ważonej;
- **wagi**: $n1 \to n2 = 1$, $n3 \to n2 = -1$;
- **bias neuronu n2 = 0**.

E5 · eksperyment z siecią xNN

1. Utwórz zmienną `L`, do której przekażesz wartość *1* jeśli lewy czujnik IR wykryje przeszkodę. W przeciwnym wypadku zmienna powinna mieć wartość *0*.
2. Utwórz zmienną `R`, do której przekażesz wartość *1* jeśli prawy czujnik IR wykryje przeszkodę. W przeciwnym wypadku zmienna powinna mieć wartość *0*.
3. Użyj bloków `set value input neuron` dla `n1` i `n3`. Do `n1` przekaż wartość `L` a do `n3` wartość `R`.
4. Wykonaj `make one NN step`, a potem odczytaj wynik z `get value output neuron n2` i przekaż go do zmiennej `S`.
5. Na podstawie `S` oraz dodatkowego warunku bezpieczeństwa wybierz reakcję robota.

Oczekiwane wyniki

L	R	S	Reakcja
0	0	0	jazda prosto
1	0	1	skręt w prawo
0	1	-1	skręt w lewo
1	1	0	stop (osobny warunek)

Edison V3: algorytm klasyczny czy xNN?

Podsumowanie misji 2

5 min

Podejście	Zalety	Ograniczenia
`if/else`	Proste i przewidywalne reguły, łatwe testowanie.	Wiele wyjątków trzeba zapisać ręcznie.
xNN	Decyzja zależna od wag, możliwość eksperymentowania z modelem.	Trzeba rozumieć wejścia, wagi i przypadki brzegowe.

„Inteligentny” nie znaczy automatycznie „bezpieczny”.

Bezpieczeństwo ruchu warto sprawdzać także niezależnymi, prostymi warunkami.

EXTRA · trzeci robot: TXT 4.0 Controller

Dla chętnych

1. Wybierz w Open Roberta Lab robot **fischertechnik TXT 4.0** i uruchom **SIM**.
2. Znajdź czujnik **Line camera** / blok `get line infrared sensor`, który zwraca `true` jak wykryje linię.
3. Zaprogramuj jazdę po czarnej linii z korektą prędkości kół.

Robot śledzący czarną linię



Instrukcja z przykładami:

fischertechnik – Line Follower

Materiały i podziękowania

- Open Roberta Lab: `https://lab.open-roberta.org/`
- Open Roberta · opis możliwości i symulacji: `open-roberta.org/features/`
- Robotino: Festo Didactic; Edison V3: Microbric / Meet Edison.
- Edison V3 i xNN: Open Roberta – Edison V3.
- International Federation of Robotics, *World Robotics 2026* – dane z 2025 r.

ERGO

Co-funded by the European Union

Co-funded by the European Union. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or the Foundation for the Development of the Education System (FRSE). Neither the European Union nor the FRSE can be held responsible for them.