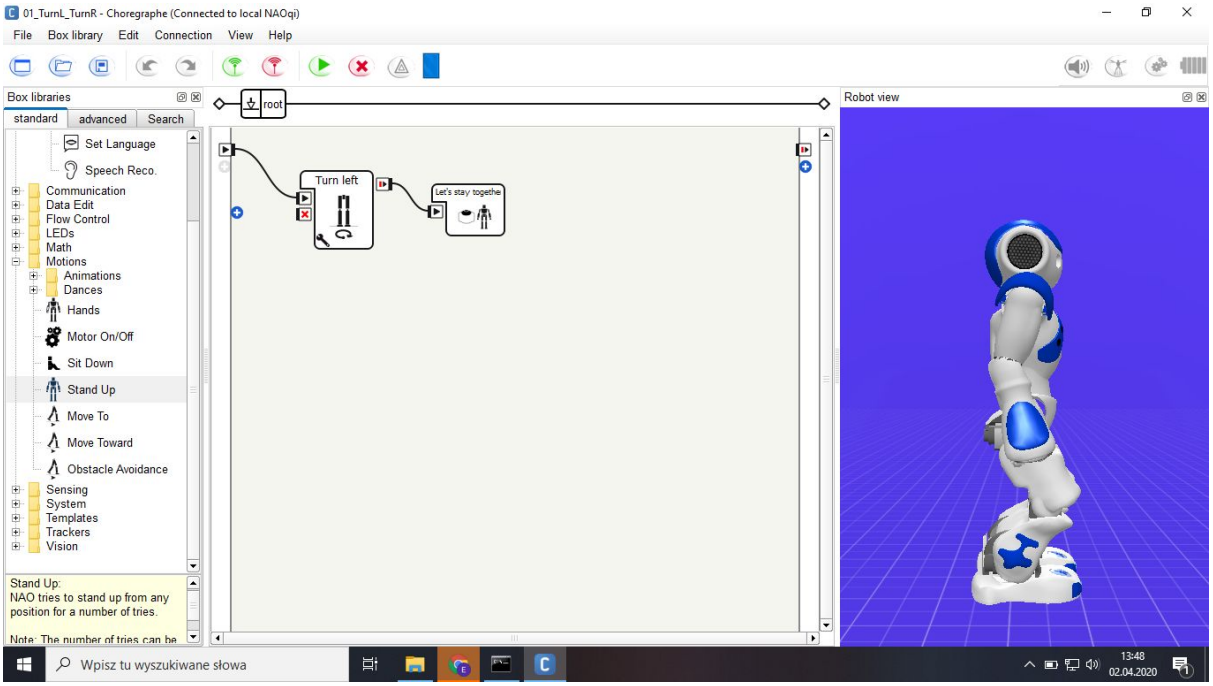
Metody Sztucznej Inteligencji

Oscar Teeninga

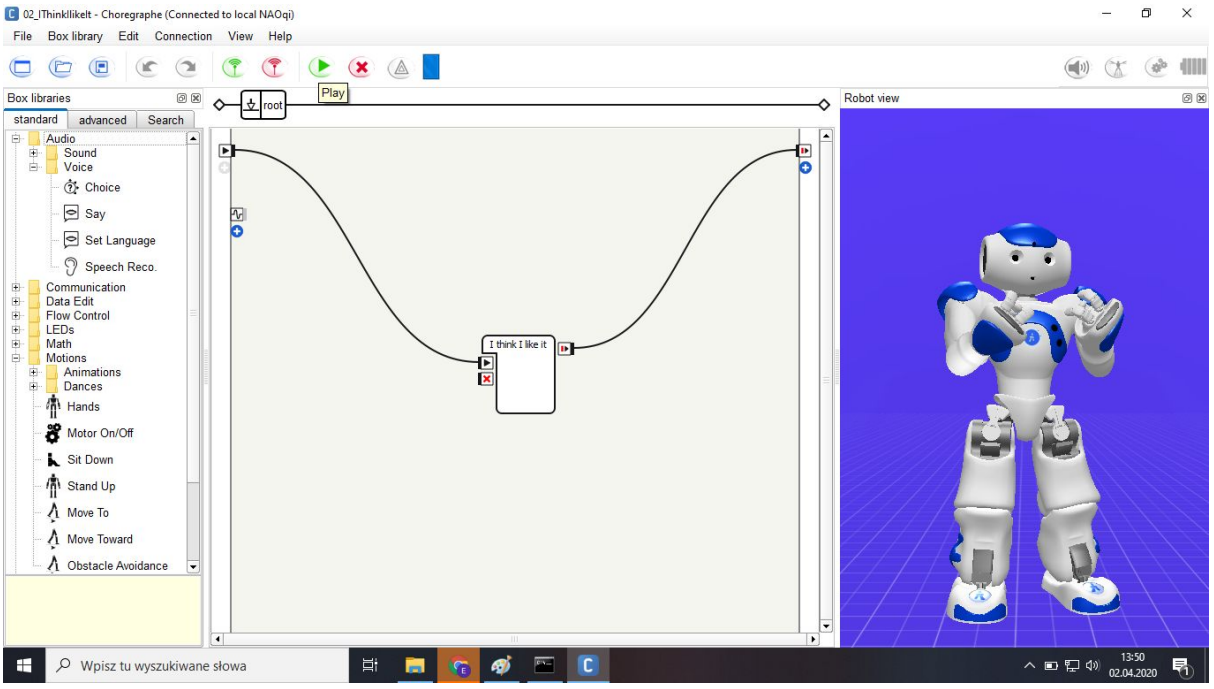
Robot Nao 03.04.2020

Labolatoria

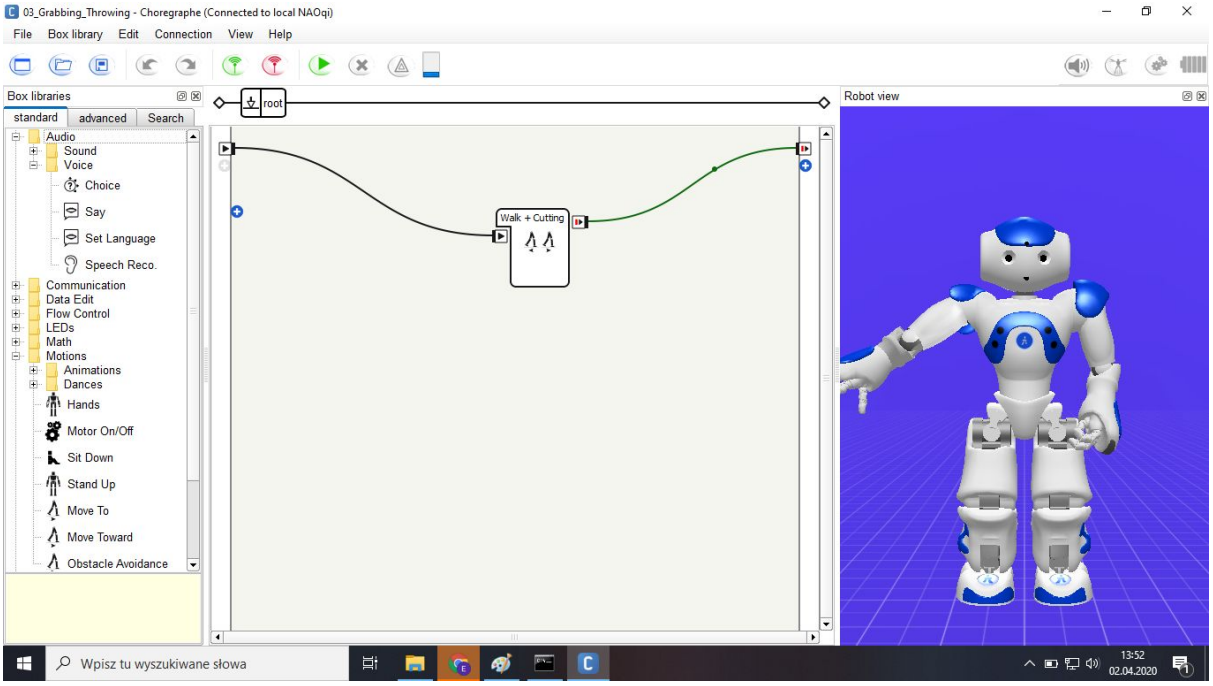
1. TurnL_TurnR



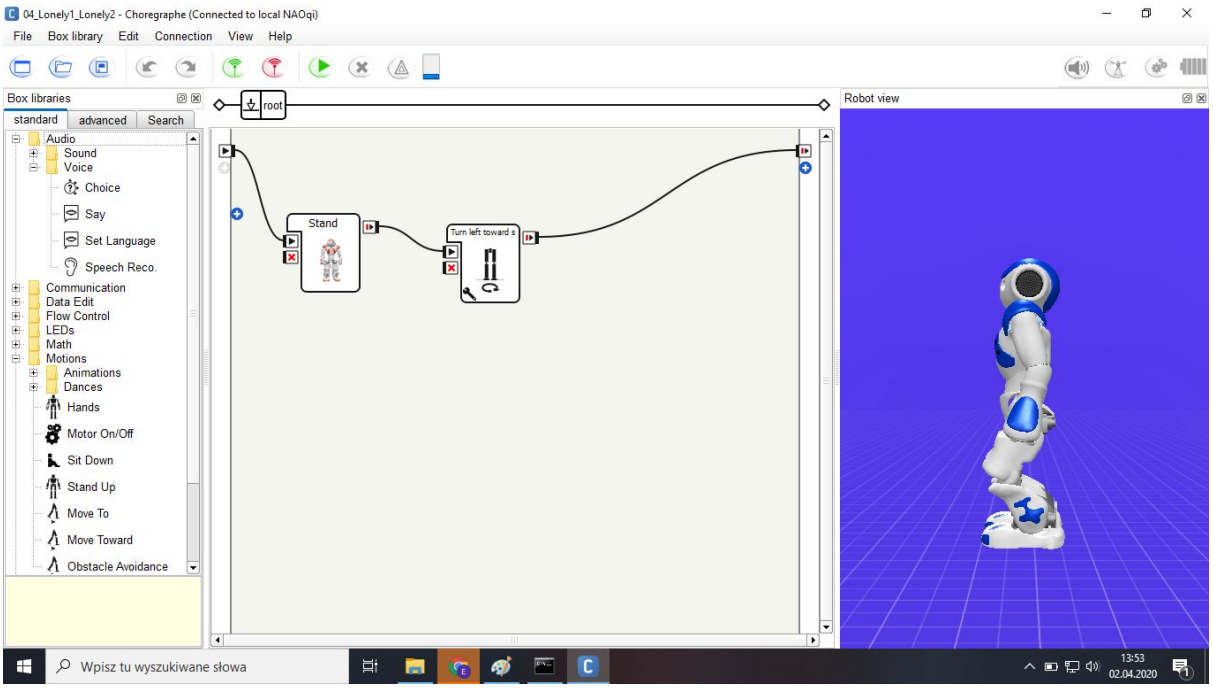
2. IThinklikeIt



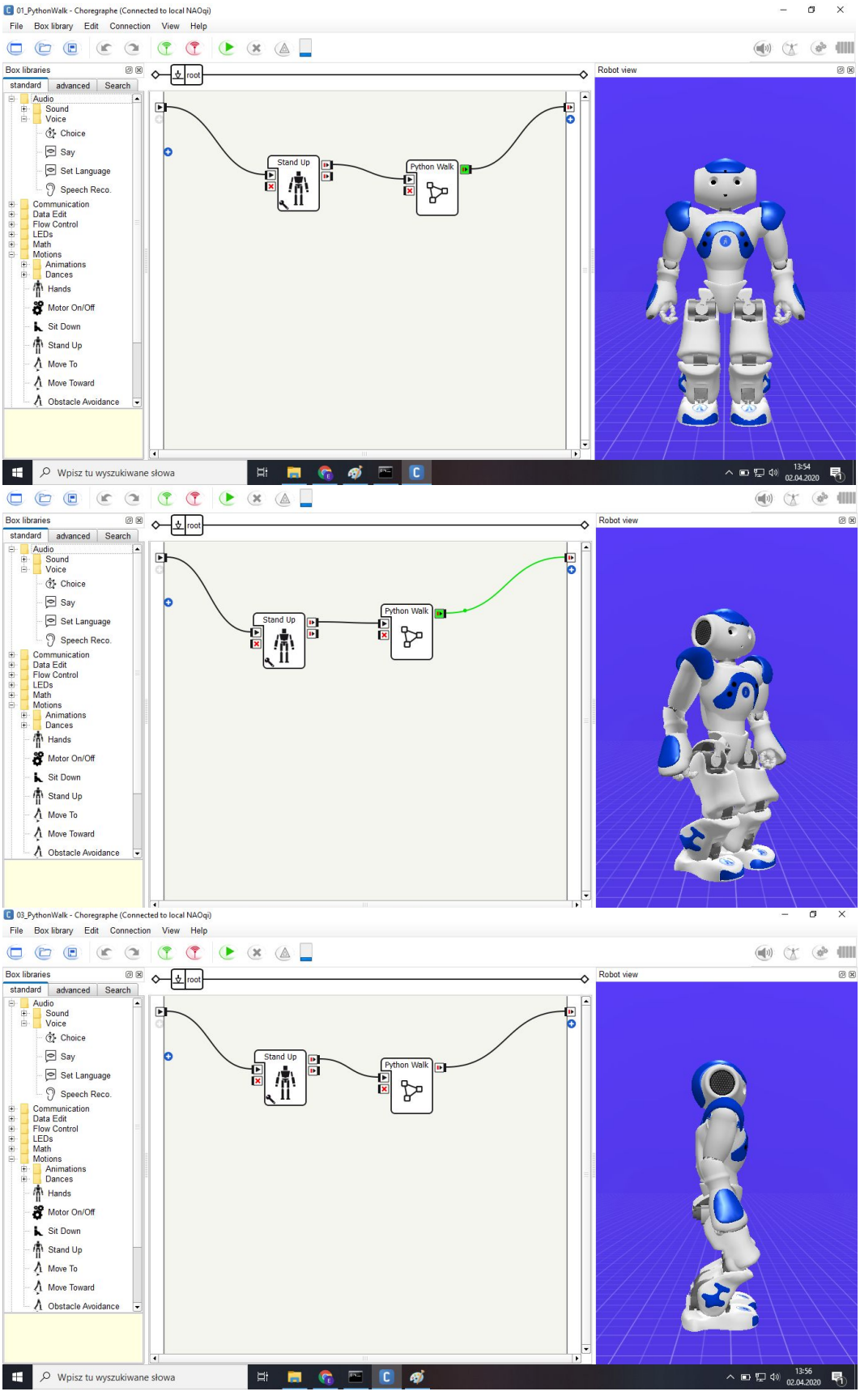
3. Grabbing_Throwing



4. Lonely1_Lonely2

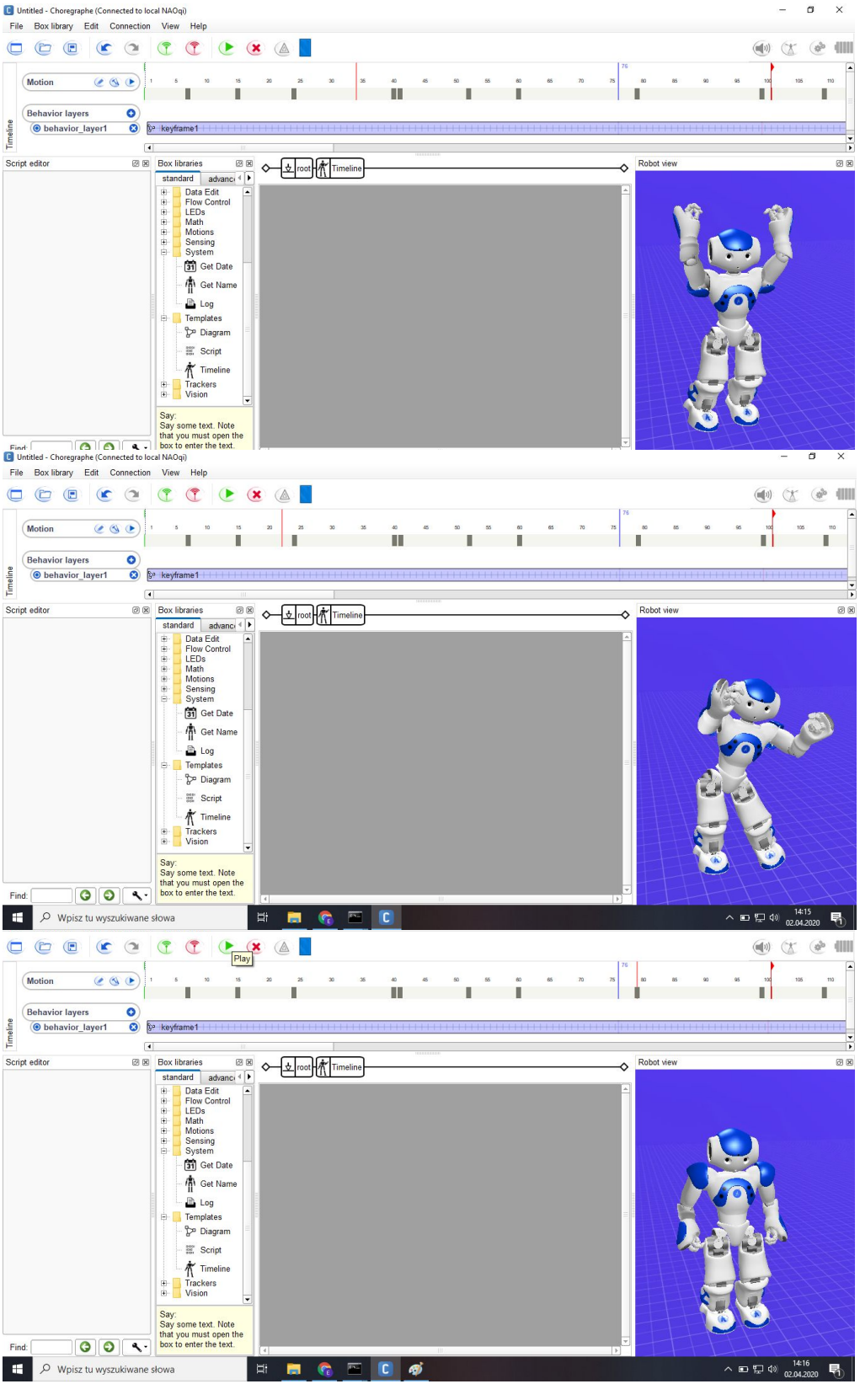


5. PythonWalk



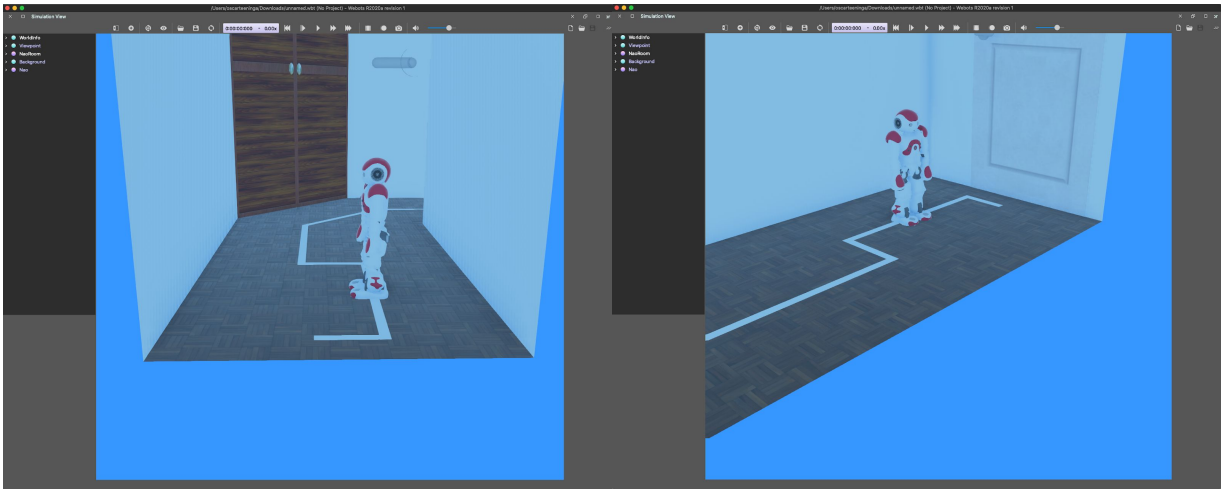
6. Timeline

Stworzyłem tańczącego robota, który po krótkim pokazie wygibasów ładnie się kłania.



Zadanie domowe

Skorzystałem z oprogramowania webots.
Zaimplementowałem w C++ robota Nao, bazując na dostarczonym demo.
Stworzyłem scenę, która umożliwi nieskończone chodzenie, więc dwie równoległe ściany i robot chodzący prostopadle do nich.

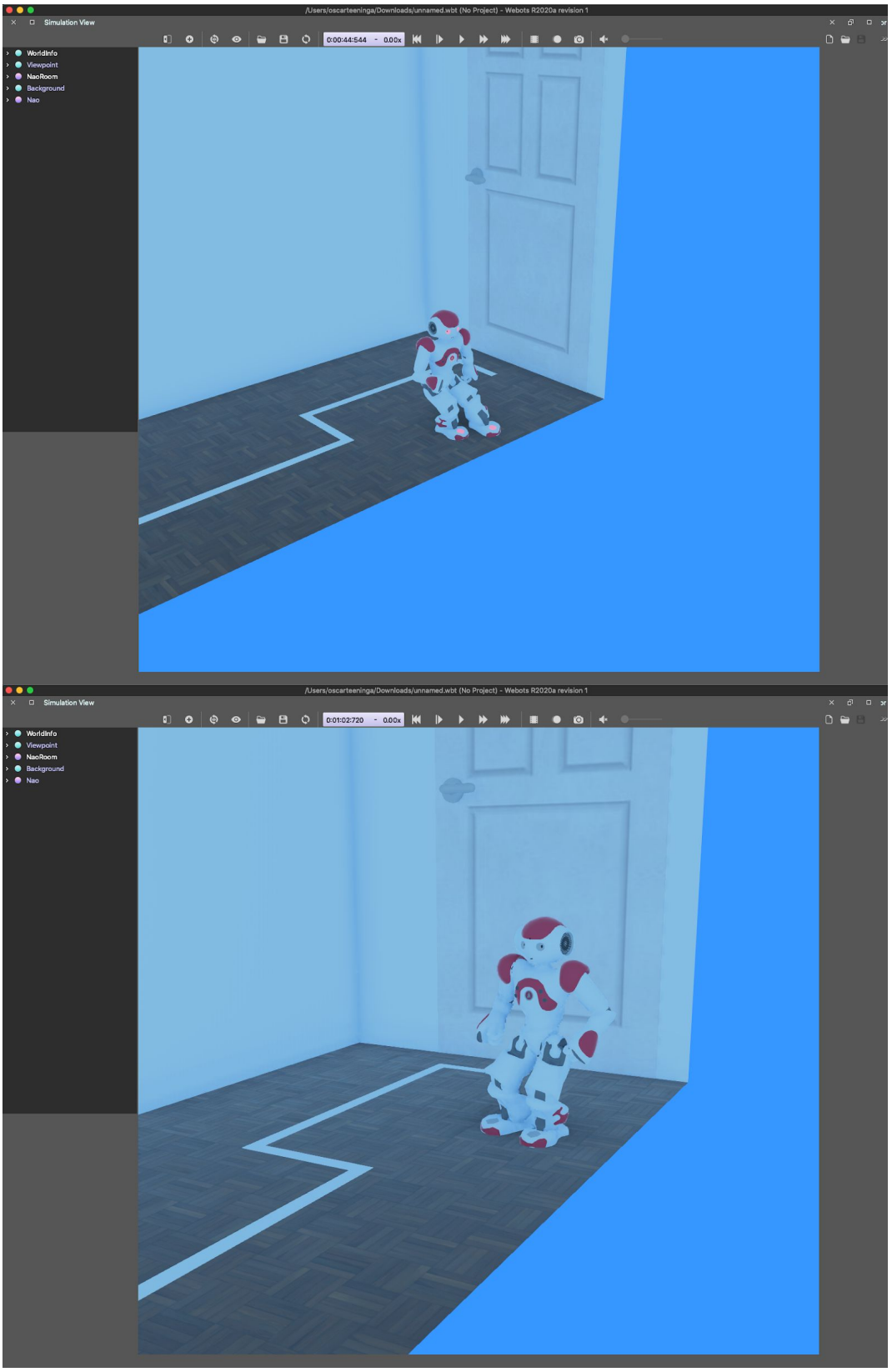


Sama detekcja odległości od ściany jest pobierana w postaci dwóch wartości (left i right), przy czym pod uwagę bierzemy wartość mniejszą.

```
static float get_ultrasound_sensors() {
  double dist[2];
  int i;
  for (i = 0; i < 2; i++)
    dist[i] = wb_distance_sensor_get_value(us[i]);

  printf("-----sensors: %f, %f-----\n", dist[0], dist[1]);
  return dist[0] > dist[1] ? dist[1] : dist[0];
}
```

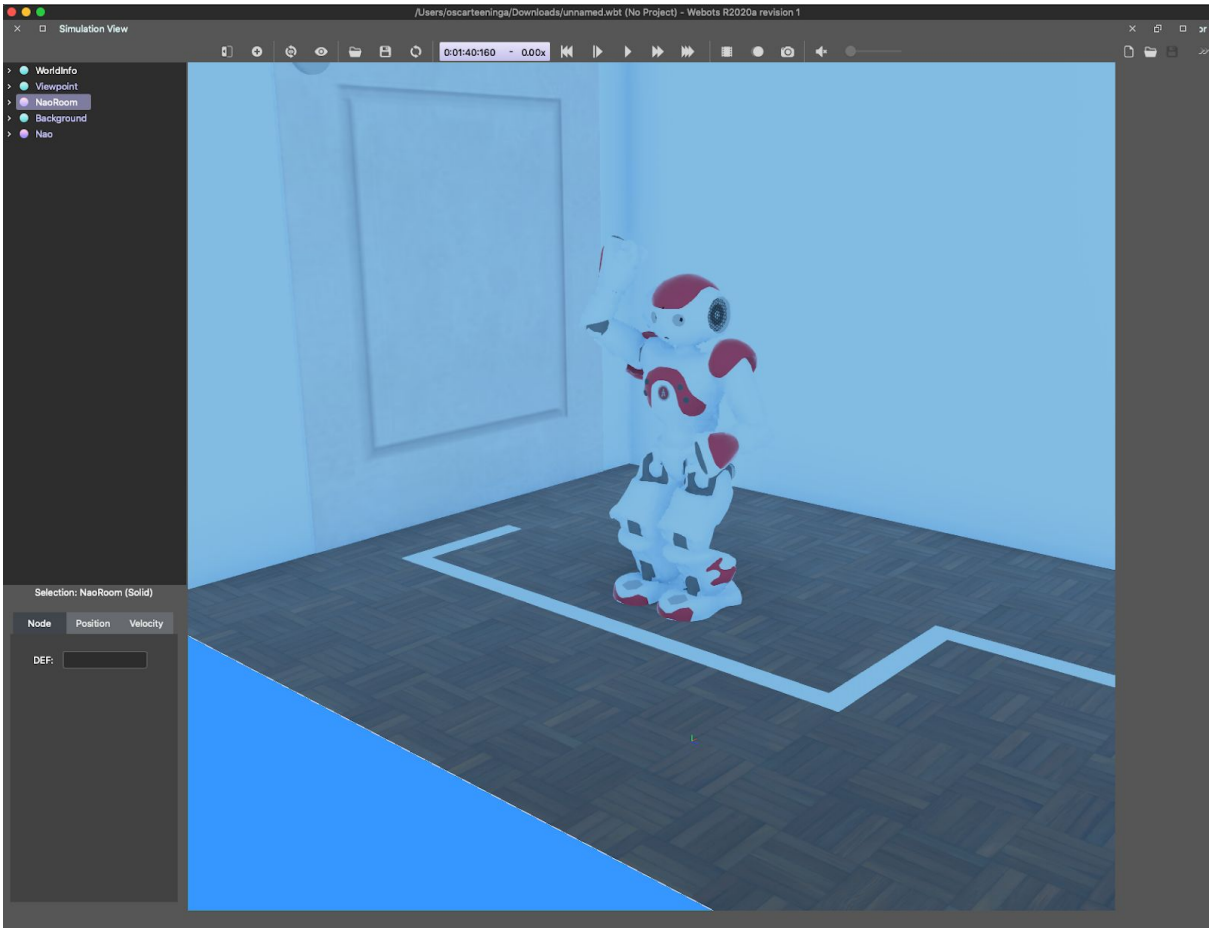
W momencie dotarcia do ściany i wykrycia bliskości ściany, zapalane są wszystkie diody na kolor czerwony, uruchamia się animacja kopnięcia,a następnie ledy są wyłączane, i zostaje uruchomiona animacja obrotu.



Obrót jest realizowany w postaci trzech animacji obrotu o 60 stopni, gdyż dołączony plik TurnLeft180.motion realizujący pełny półobrót tak naprawdę jest niedokładny i obrót jest większy niż 180 stopni.

```
static void turn_around() {
  start_motion(turn_right_60);
  start_motion(turn_right_60);
  start_motion(turn_right_60);
  printf("Turned around!\n");
}
```

W losowych momentach (z prawdopodobieństwem 10%) robot staje i zaczyna machać ręką.




```
if (rand() % 100 > 90) {
  hand_wave_animation();
}
```

Każda animacja jest blokująca i program wpada w pętlę wywołującą funkcję simulation_step(). Pewnym problemem był fakt, że istnieje pewna niedokładność funkcji wbu_motion_get_duration(), gdyż po podzieleniu tej wartości przez time_step nie otrzymujemy realnej liczby kroków (lub są pewne opóźnienia), więc dla pewności ilość kroków symulacji zwiększyłem dwukrotnie względem oczekiwanej wartości (co funkcjonuje rewelacyjnie, tylko czasami robot jest bezczynny przez ułamek sekundy).

```
static void wait_steps(WbMotionRef motion) {
  for (int i = 0; i < 2*wbu_motion_get_duration(motion)/time_step; i++) {
    simulation_step();
  }
}

static void start_motion(WbMotionRef motion) {
  // interrupt current motion
  if (currently_playing)
    wbu_motion_stop(currently_playing);

  // start new motion
  wbu_motion_play(motion);
  wait_steps(motion);
  currently_playing = motion;
}
```

Próbowałem stworzyć własny plik typu *.motion, jednak jest to niebywale trudne, aby funkcjonowało to poprawnie.

Dodatkowo każda animacja kończy się informacją zwrotną wyświetlaną w terminalu.

```
[oscarteeninga@MacBook-Pro-Oscar nao_demo % ./nao_demo
------timestep: 32----------------nao_oscar----------
This is my Nao project
-----sensors: nan, nan-----
-----sensors: nan, nan-----
Made a step!
-----sensors: 0.648967, 0.673786-----
Made a step!
-----sensors: 0.558100, 0.579838-----
Made a step!
-----sensors: 0.467235, 0.486009-----
Made a step!
Hi, how you doin???
-----sensors: 0.376342, 0.392290-----
Made a step!
-----sensors: 0.285542, 0.298733-----
Made a step!
-----sensors: 0.250000, 0.250000-----
Shooooooot!!!!
Turned around!
-----sensors: 0.955747, 0.991512-----
Made a step!
-----sensors: 0.861852, 0.894125-----
Made a step!
-----sensors: 0.771147, 0.800134-----
Made a step!
-----sensors: 0.680538, 0.706442-----
Made a step!
```

Cały algorytm uruchamiany jest w funkcji main w postaci pętli nieskończonej.

```
while (1) {
  if (get_ultrasound_sensors() < 0.27) {
    set_all_leds_color(0xff0000);
    shoot_animation();
    set_all_leds_color(0x000000);
    turn_around();
  } else {
    forward_step();
  }
  if (rand() % 100 > 90) {
    hand_wave_animation();
  }
}
```

Ciekawe jest również to, że za pierwszym wywołaniem metody zwracającej odczyt czujników odległości dostajemy Nan, dlatego przed całym programem trzeba uruchomić tę metodę (lub jest to czasowe i wystarczyło by wywołać krok symulacji).

```
// sensors get initialization (bugs without it)
get_ultrasound_sensors();
```