

Sistem de auto-organizare folosind roboți colaborativi

Coordonatori științifici: Prof. Dr. Ing. Corneliu Burileanu,
As. Univ. Drd. Ing. Ana Neacșu

Student: Teodora – Cătălina Vaman

Facultatea de Electronică, Telecomunicații
și Tehnologia Informației

Iulie 2021

Introducere

Introducere

Ce se înțelege prin inteligența colectivă?

- frecvent întâlnită în natură
- nu are un sistem centralizat de luat decizii
- respectă unul din principiile:
 - coordonare
 - colaborare
 - deliberare

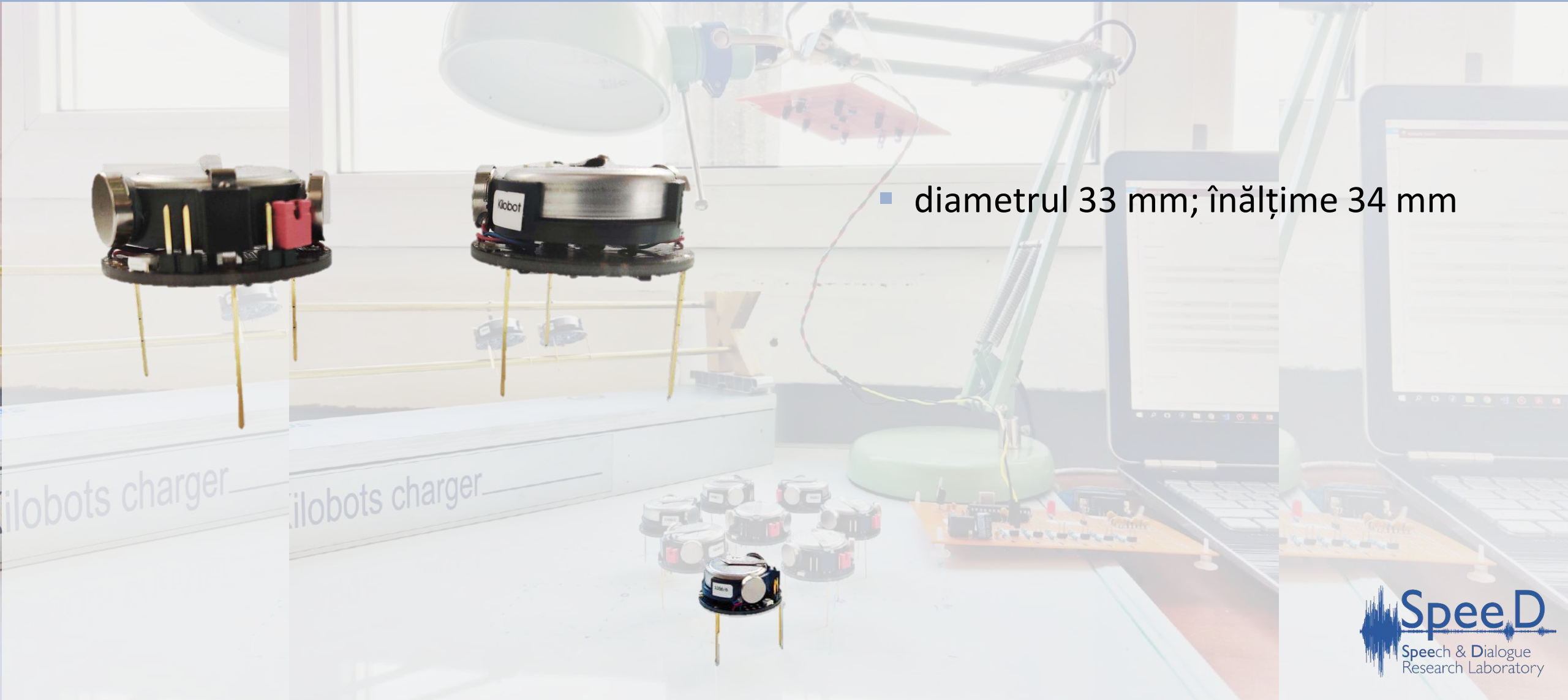


Scopul lucrării

Analiza acestui comportament emergent în natură și implementarea și testarea unor algoritmi prin care un grup de roboți se auto-organizează în diverse forme bidimensionale

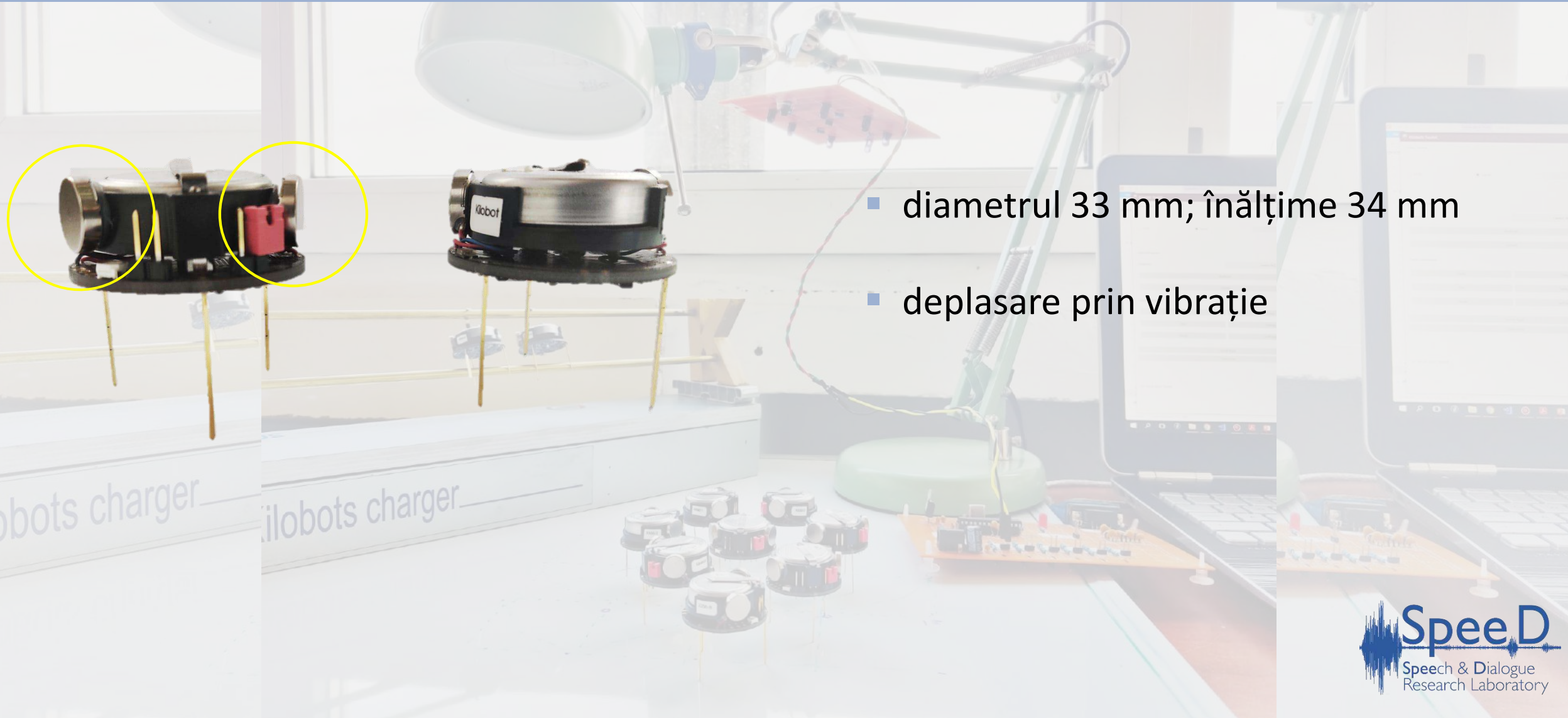
Setup-ul experimental

Kilobots

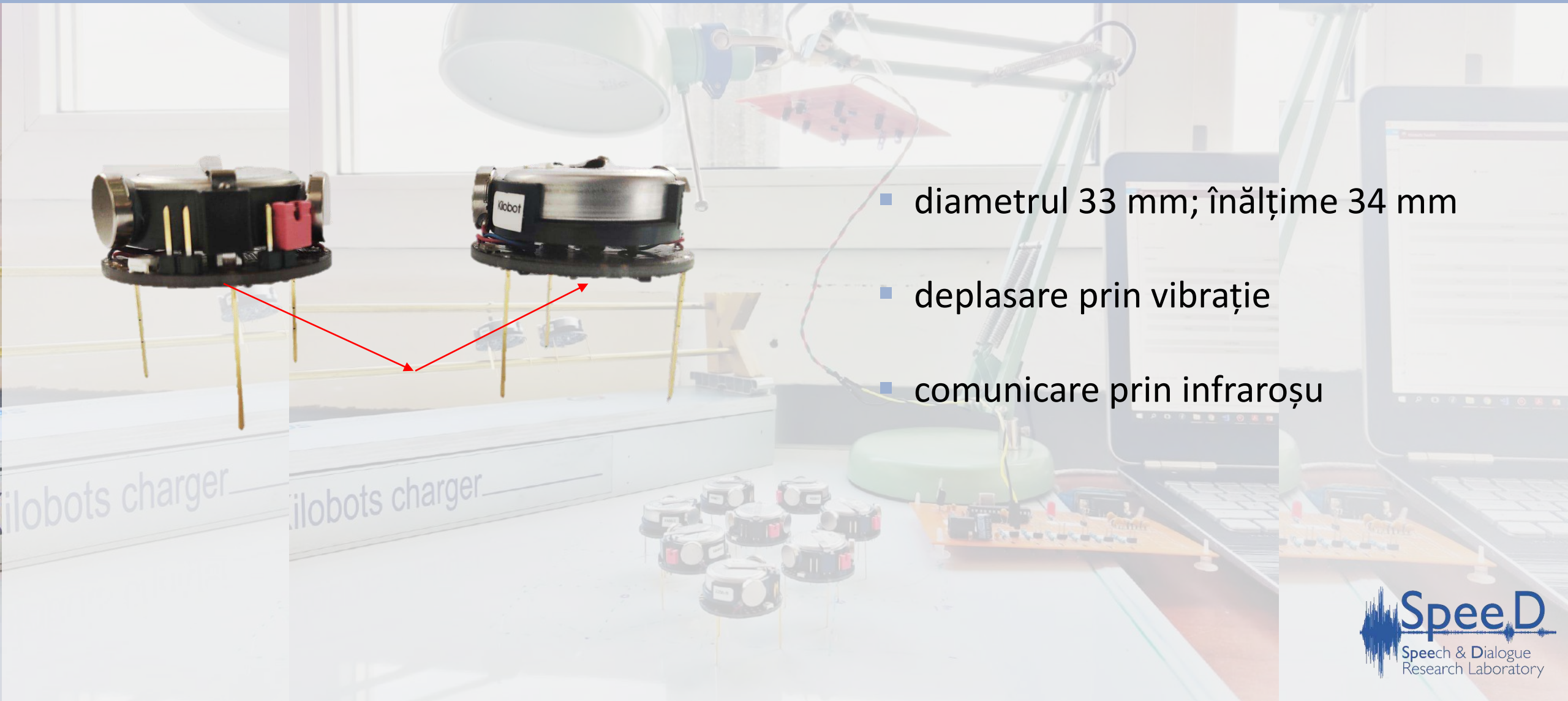


- diametrul 33 mm; înălțime 34 mm

Kilobots

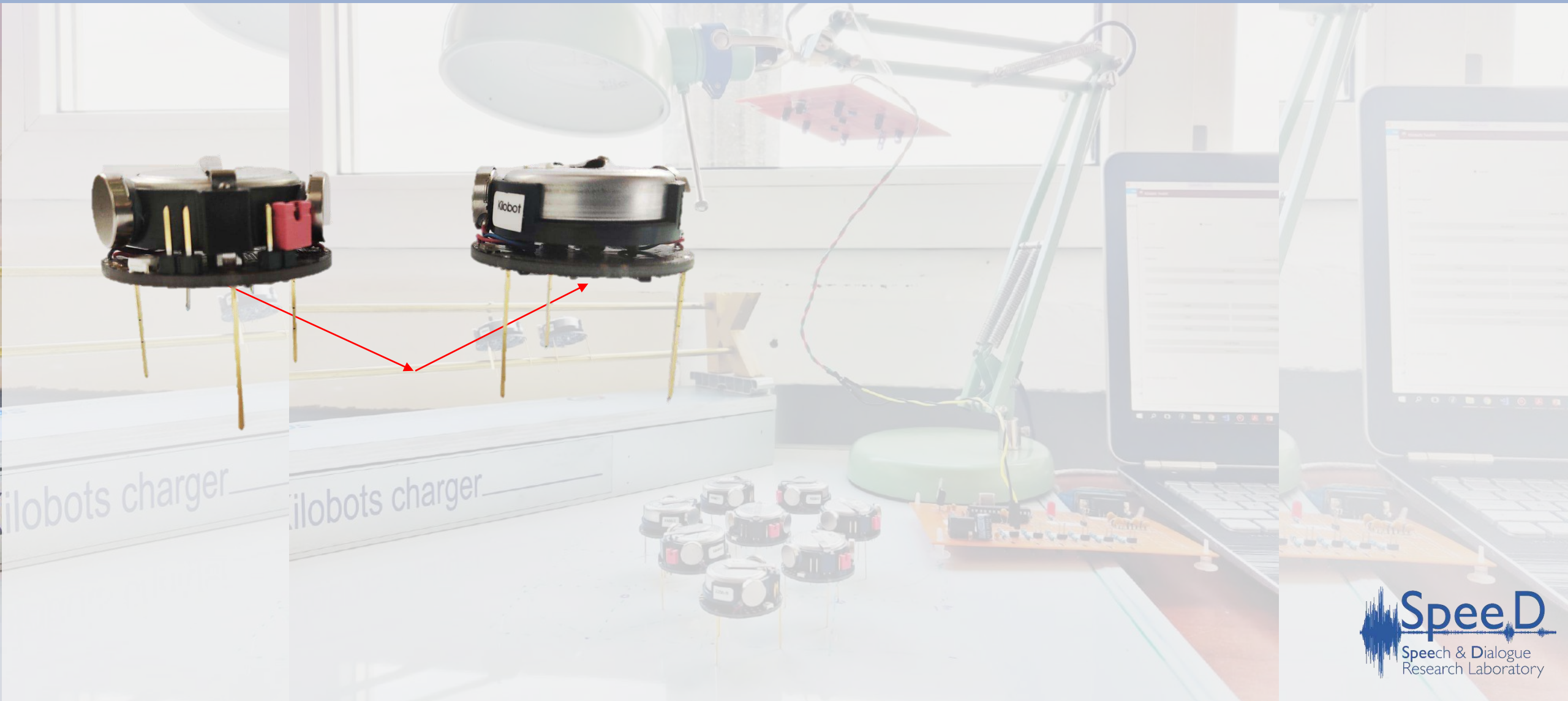


Kilobots

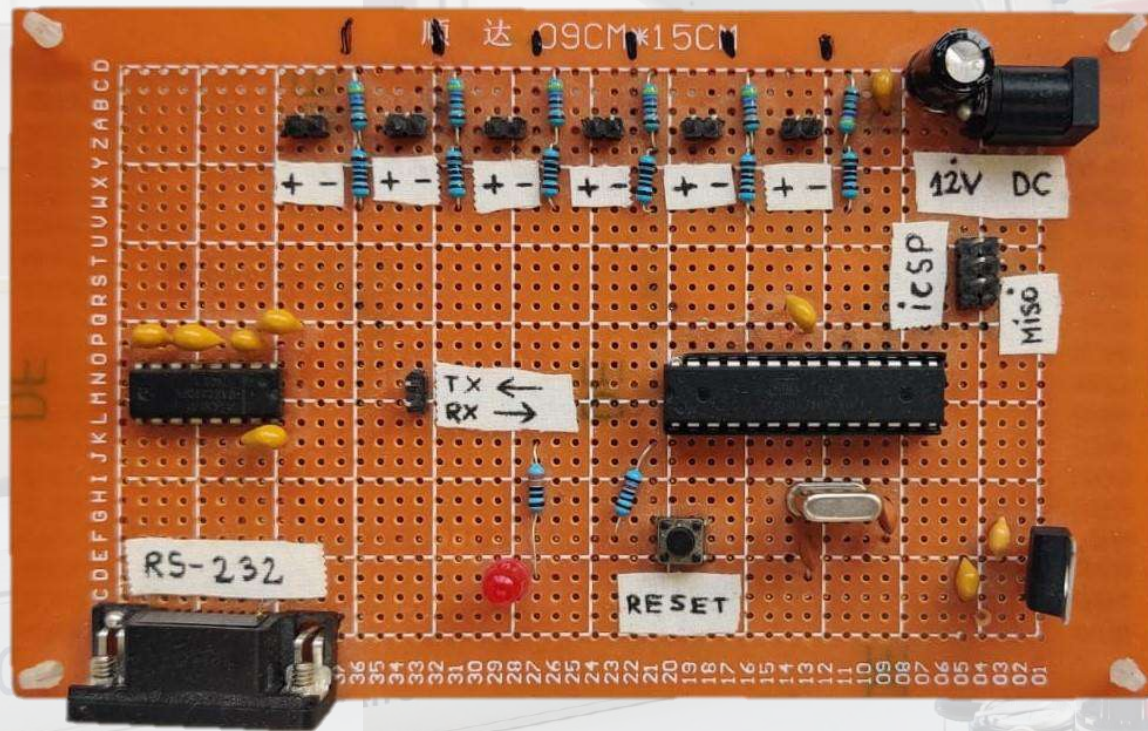


- diametrul 33 mm; înălțime 34 mm
- deplasare prin vibrație
- comunicare prin infraroșu

Kilobots

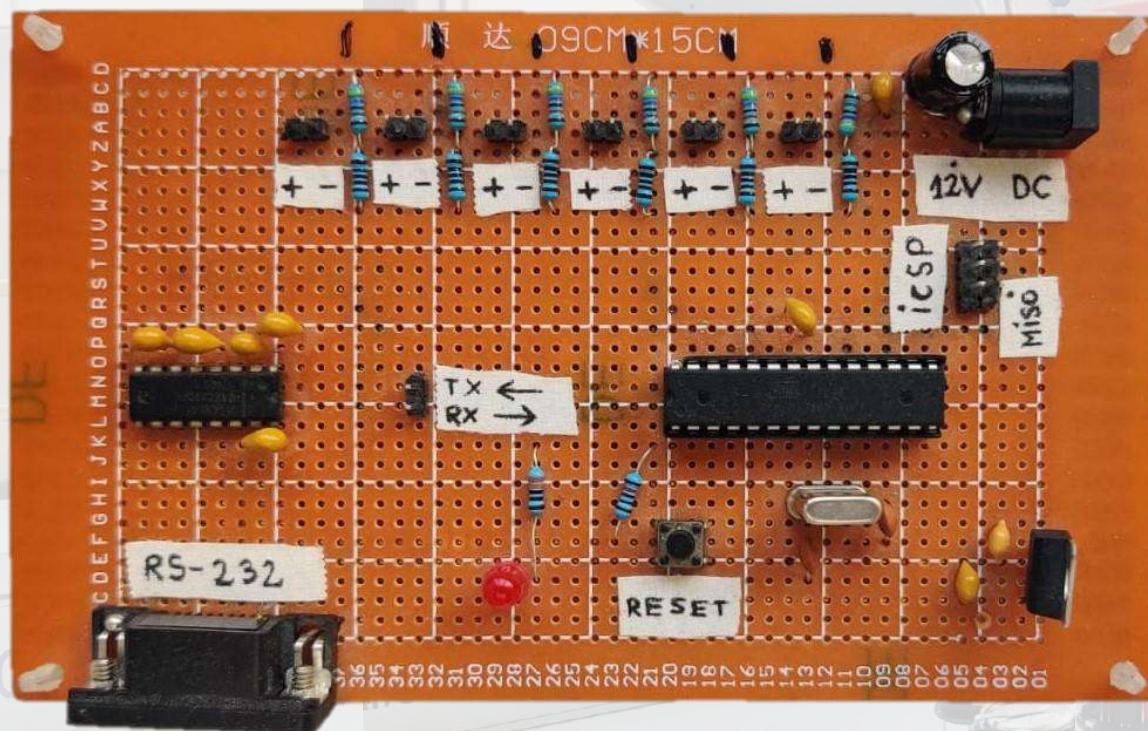


Overhead Controller



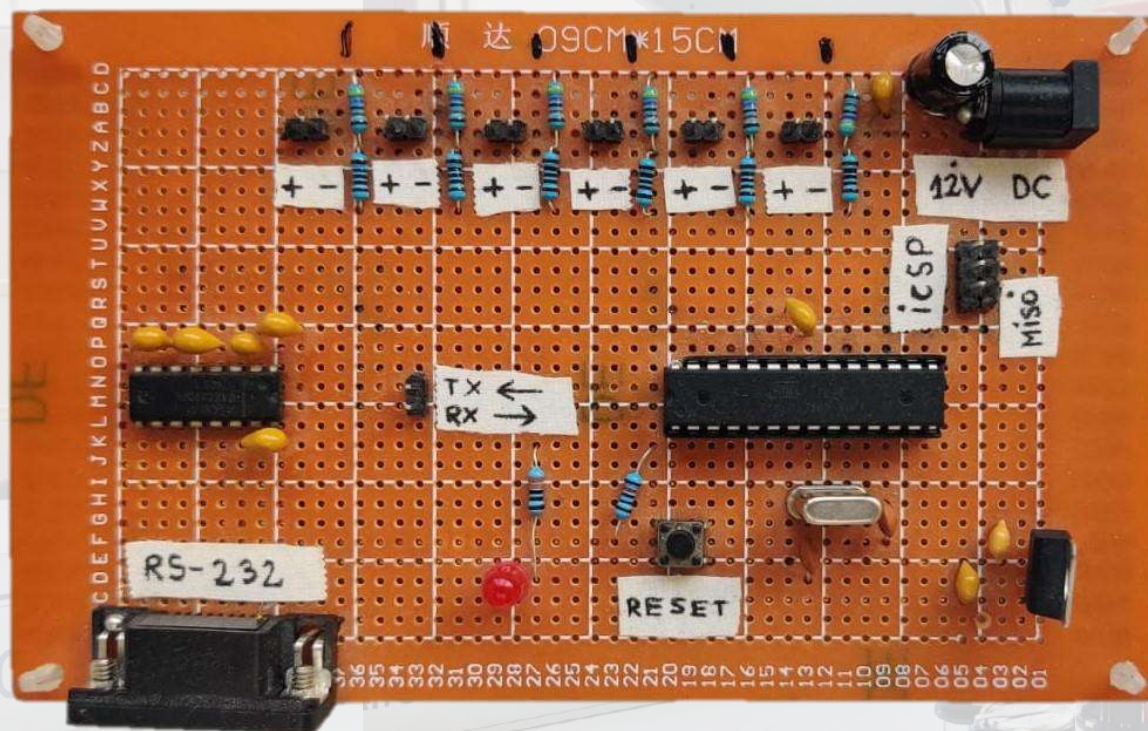
- Interfațează comunicarea cu kiloboții

Overhead Controller



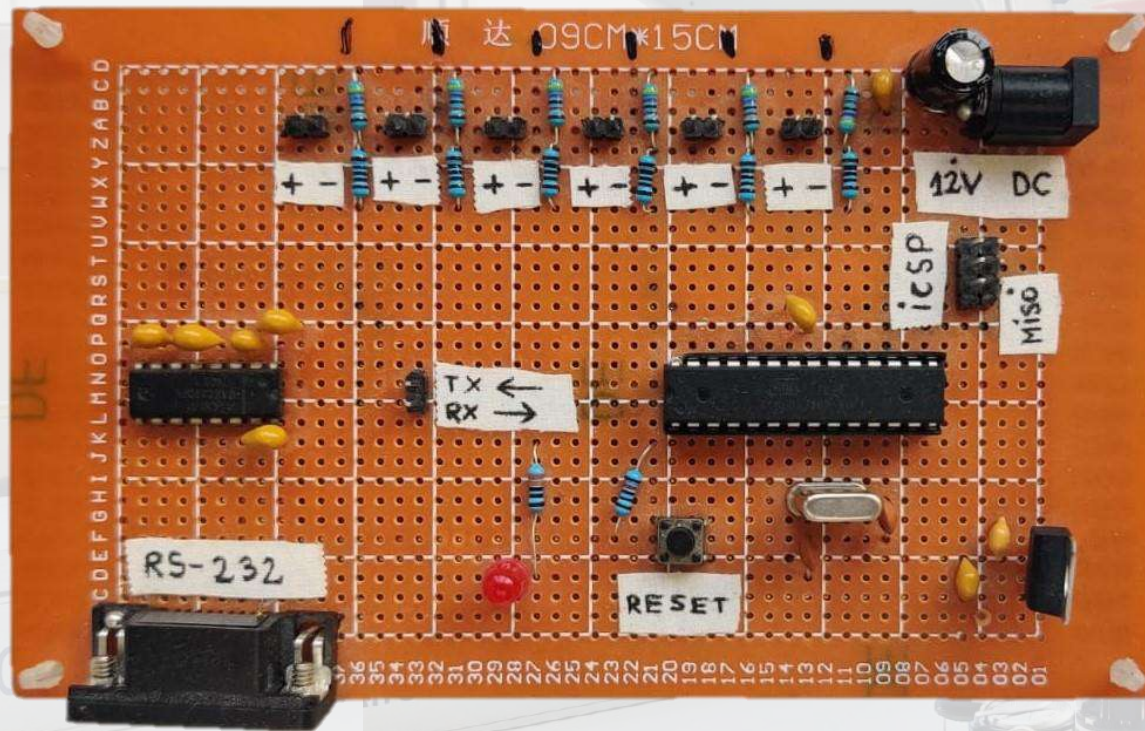
- Interfațează comunicarea cu kiloboții
- Transformă informația în pulsuri de lumină infraroșie prin intermediul plăcii cu LED-uri

Overhead Controller



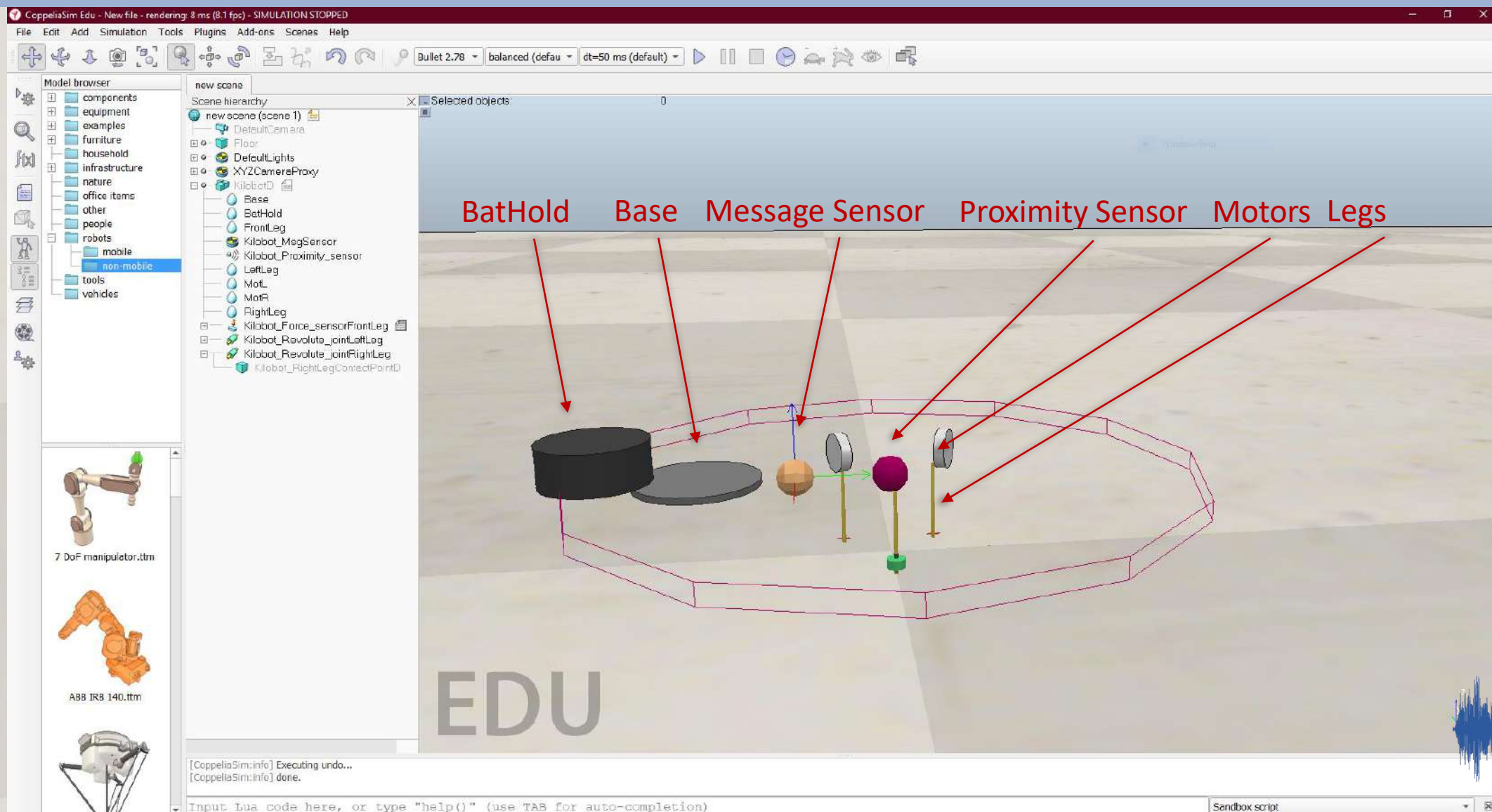
- Interfațează comunicarea cu kiloboții
- Transformă informația în pulsuri de lumină infraroșie prin intermediul plăcii cu LED-uri
- Comunică cu întreg grupul de roboți în același timp

Overhead Controller



- Interfațează comunicarea cu kiloboții
- Transformă informația în pulsuri de lumină infraroșie prin intermediul plăcii cu LED-uri
- Comunică cu întreg grupul de roboți în același timp

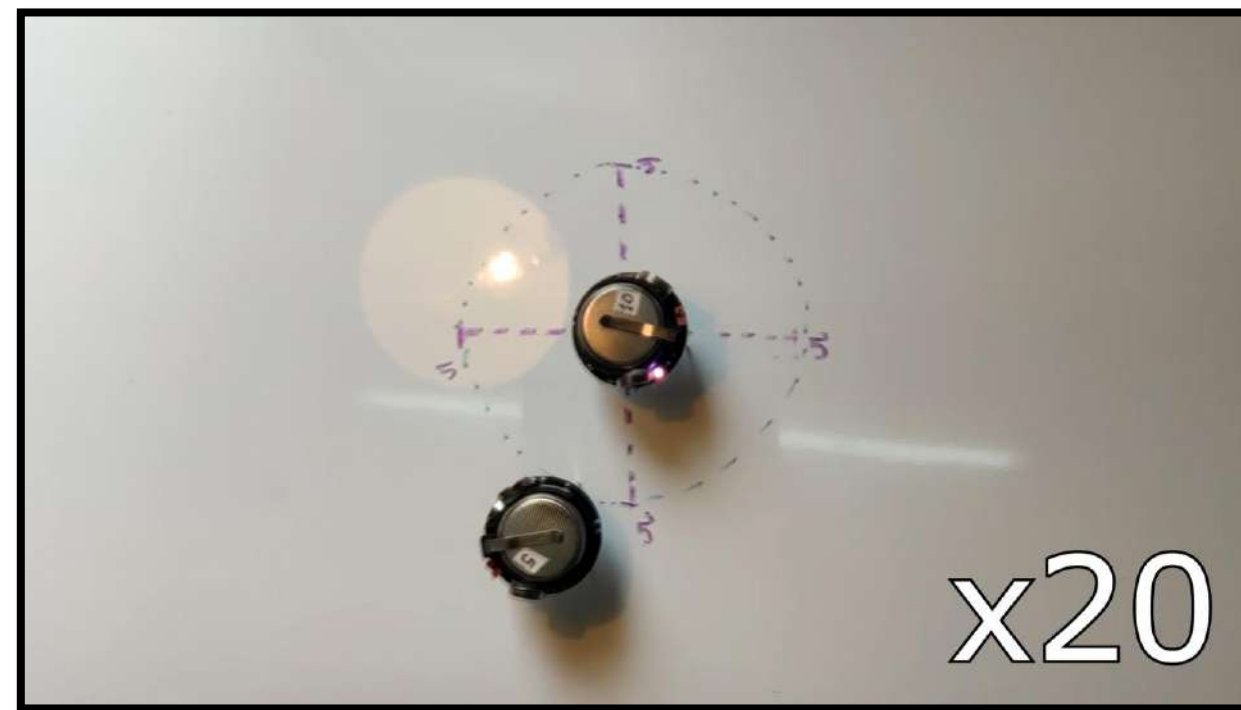
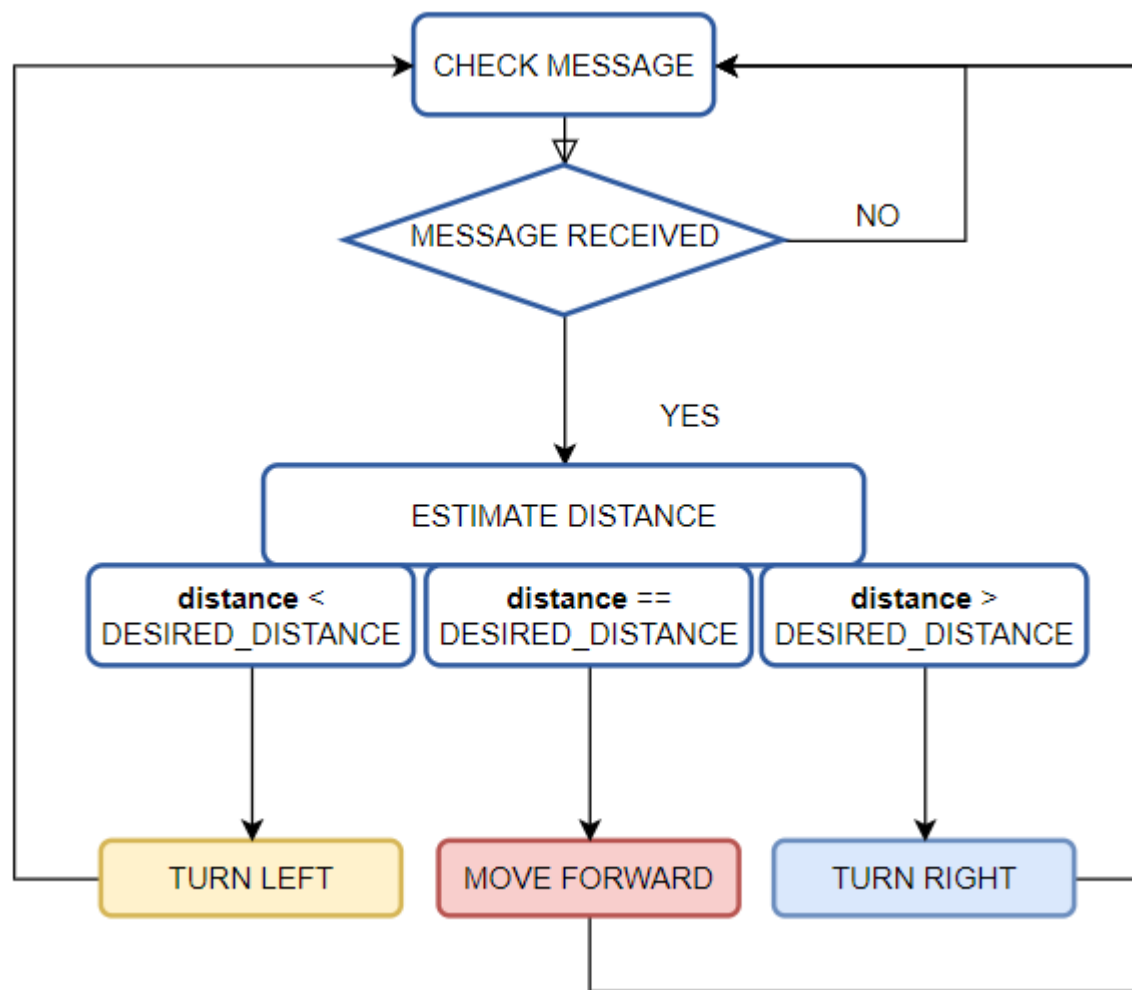
Programul de simulare CoppeliaSim

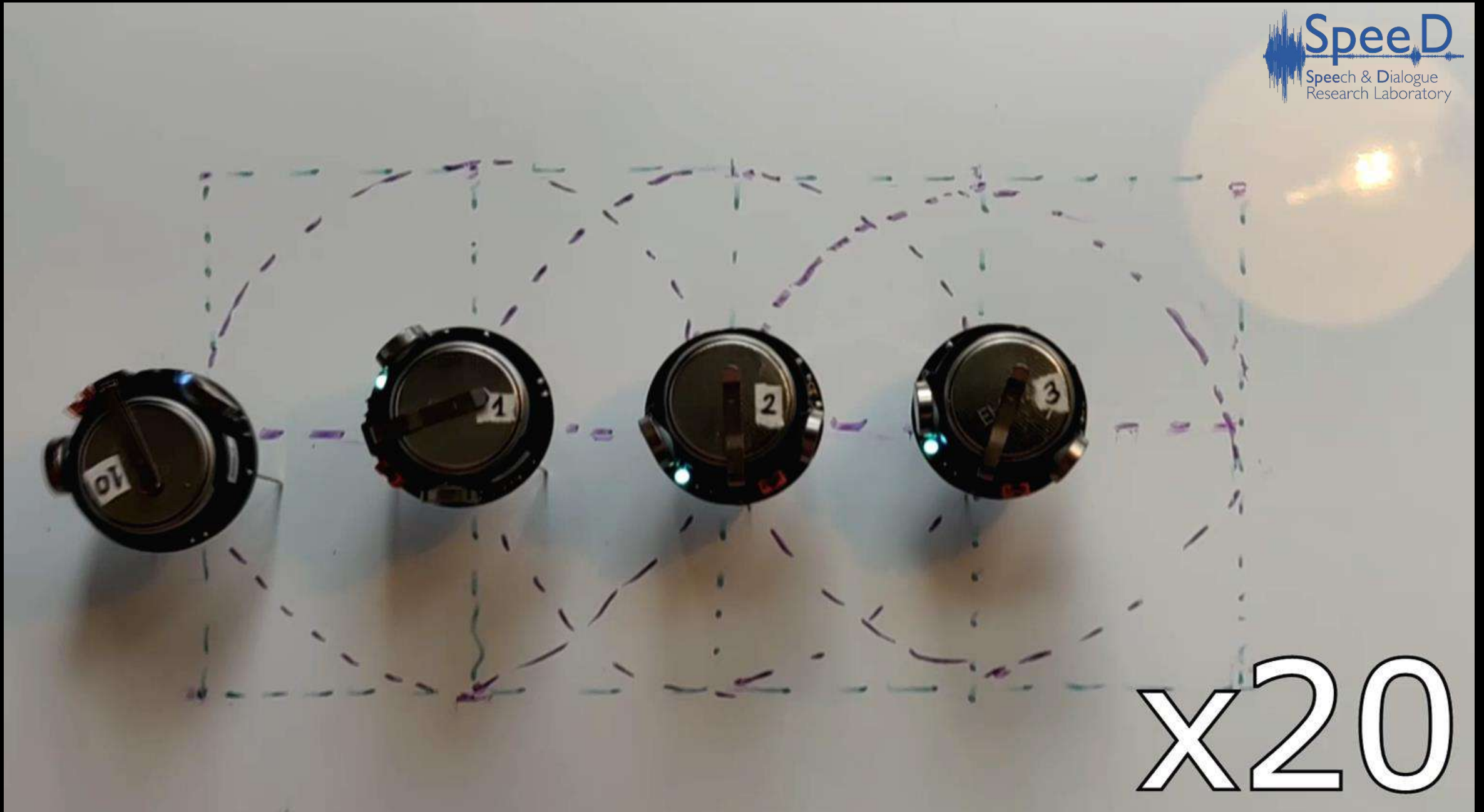


Implementarea software

- Programele încărcate pe kiloboți au fost scrise folosind `embedded C`
- Programele folosite în simulare sunt scrise în limbajul `Lua`
- Toate programele se folosesc de librăria pre-existentă `Kilolib`

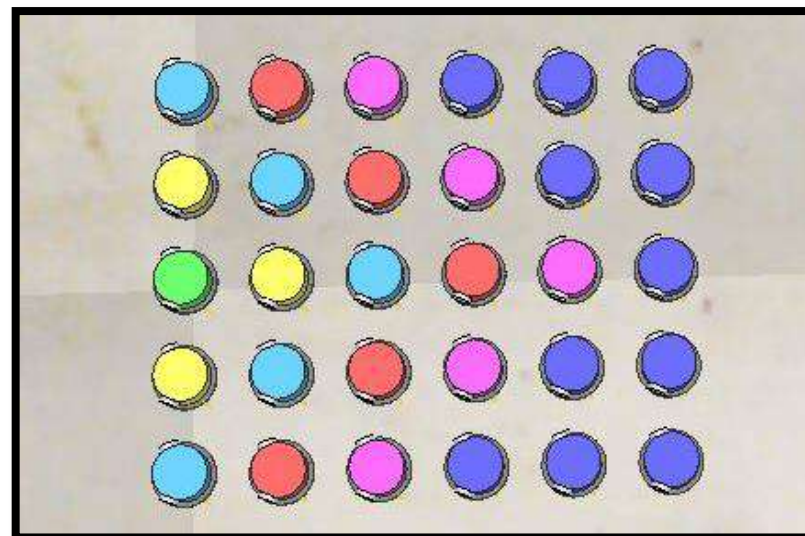
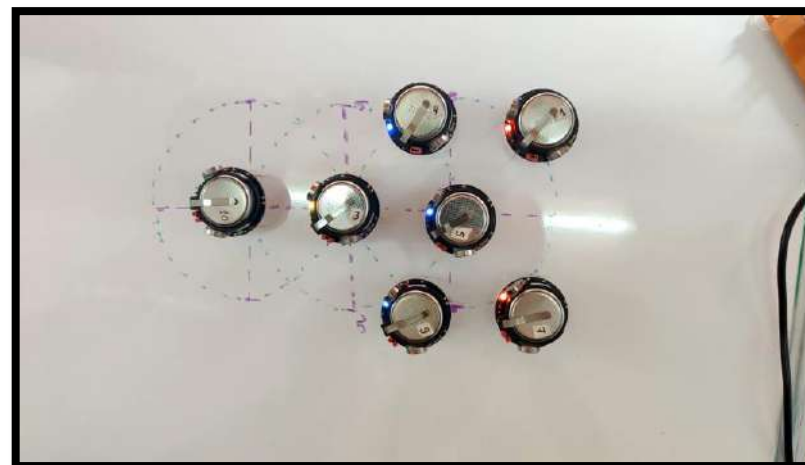
Algoritmul de Orbitare



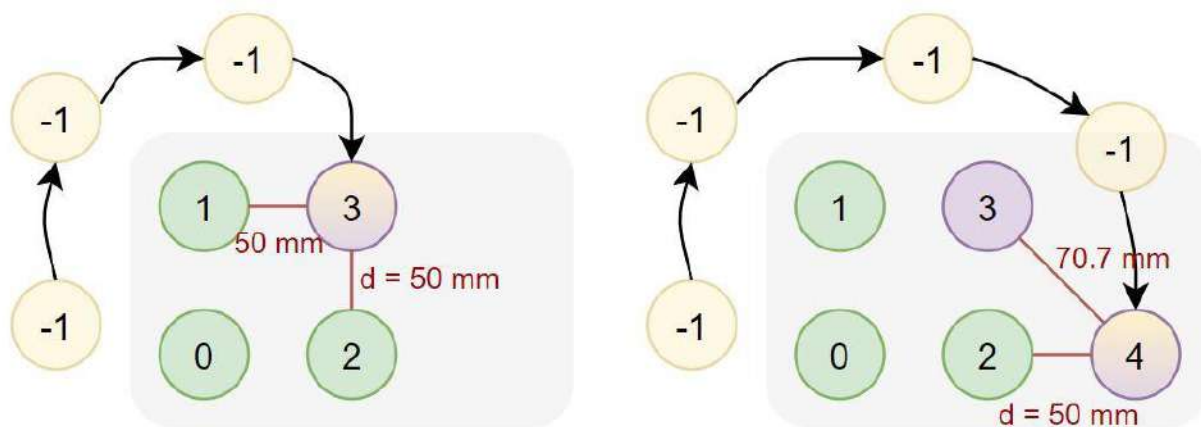


Algoritmul de formare de gradient

- Pentru a oferi robotilor un grad de orientare spatială
- Fiecare robot are o valoare a gradientului între 0 – 255
 - VERDE* -> 0 -> **sursa gradientului**
 - GALBEN* -> 1
 - ALBASTRU* -> 3
 - ROSU* -> 4
 - ROZ* -> 5
 - MOV* -> valoare > 5
- Robotul memorează valoare minimă înregistrată și își setează gradientul ca fiind **minim + 1**



Algoritmul de formare de imagini 1



Starile în care se pot afla kiloboții:

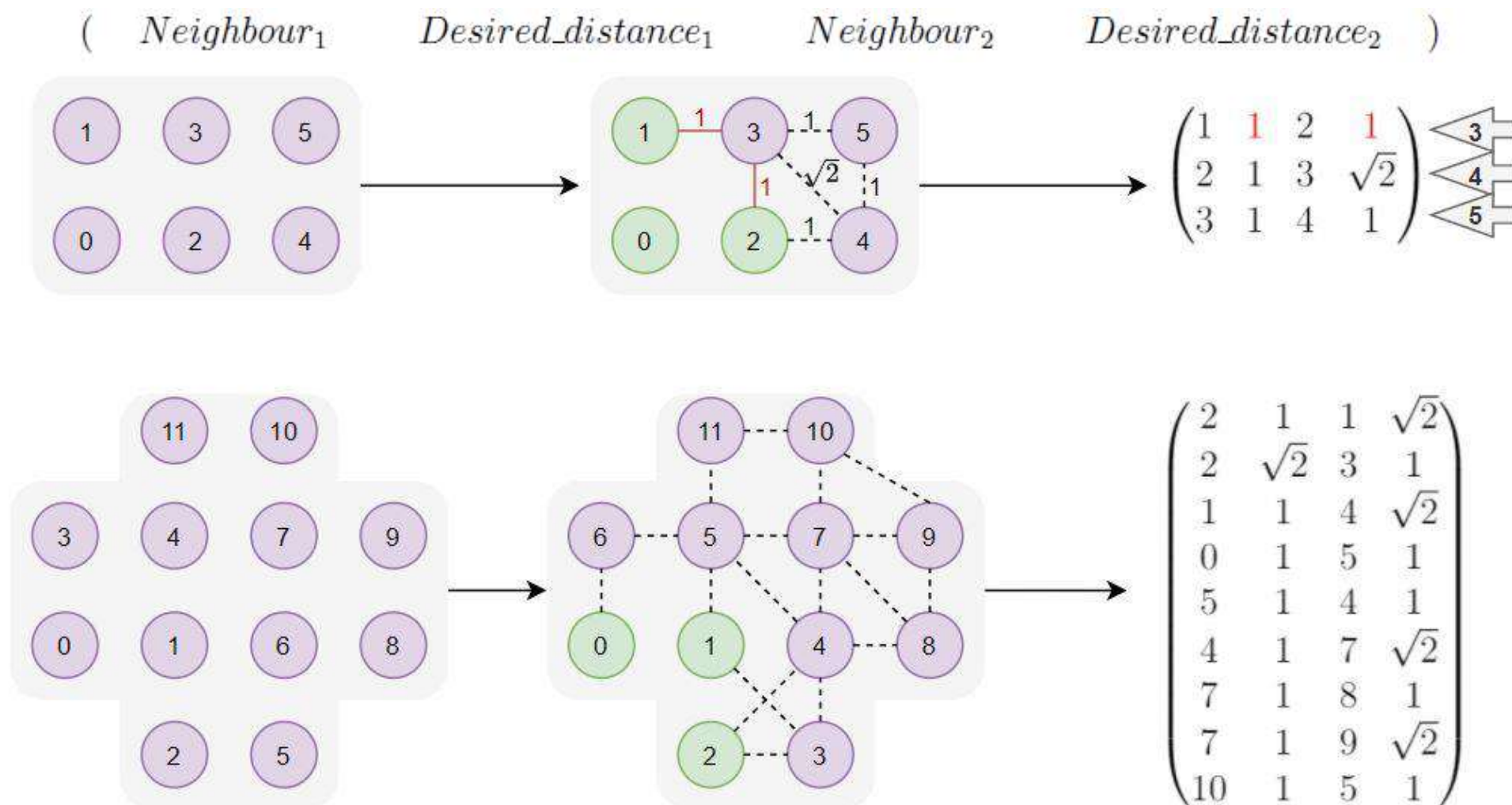
WAIT - edge detection

MOVE - robotul se deplasează și verifică dacă a ocupat un loc valid în formă

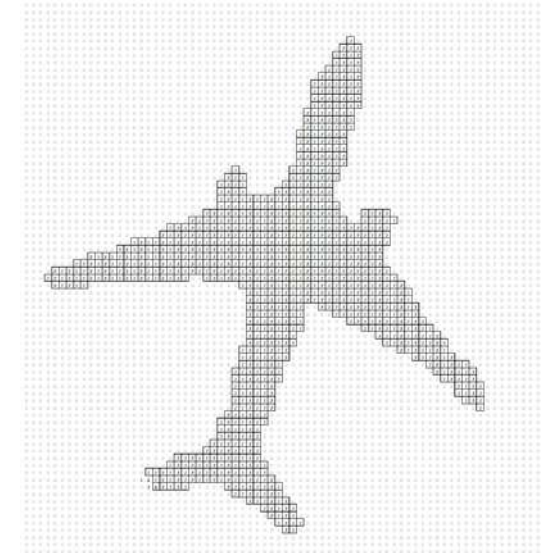
LOCAL - poziție stabilită în formă, modifică propriul index și devine punct de referință pentru alți roboți



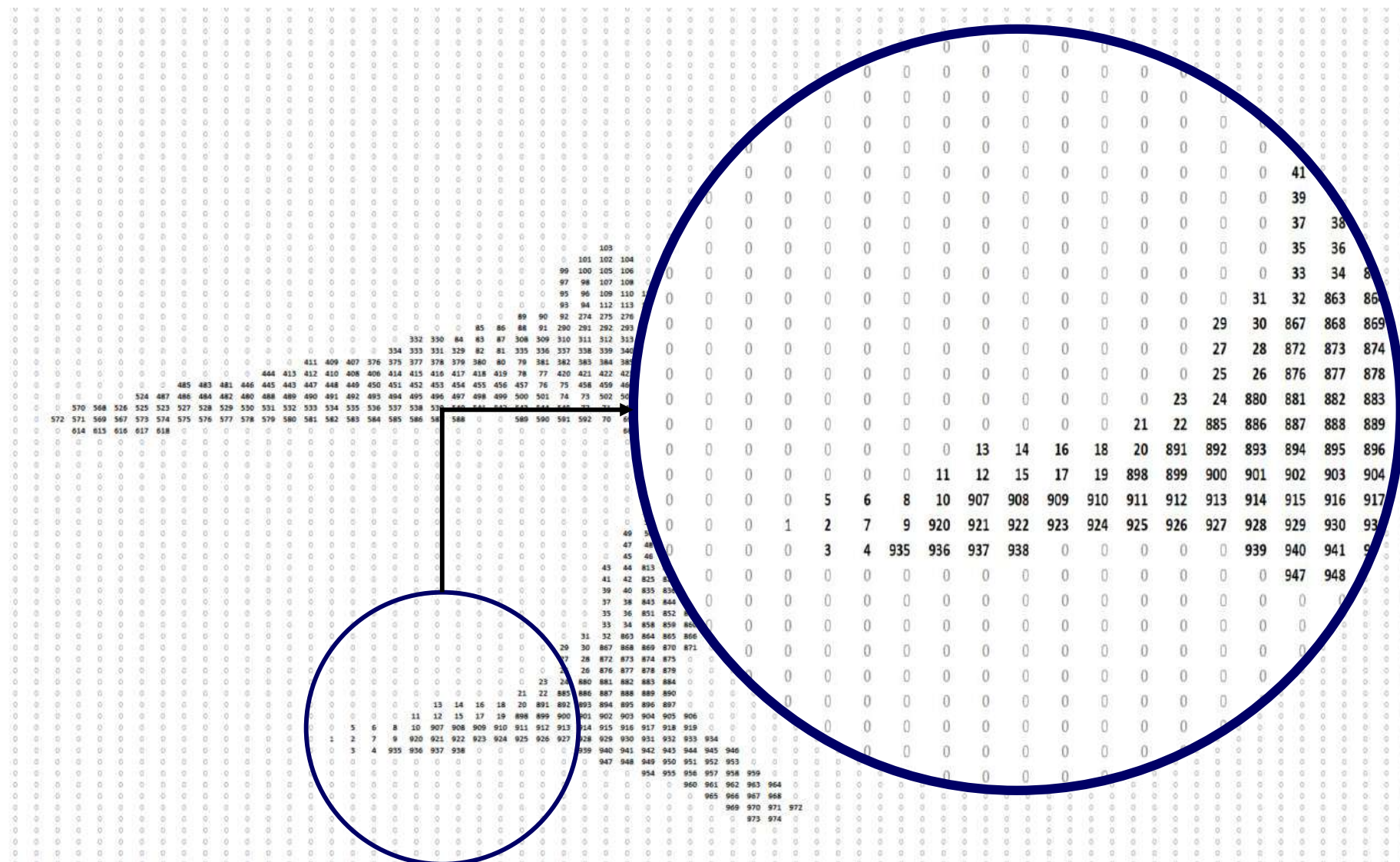
Algoritmul de formare de imagini 1



Generarea automată a matricii



Generarea automată a matricii

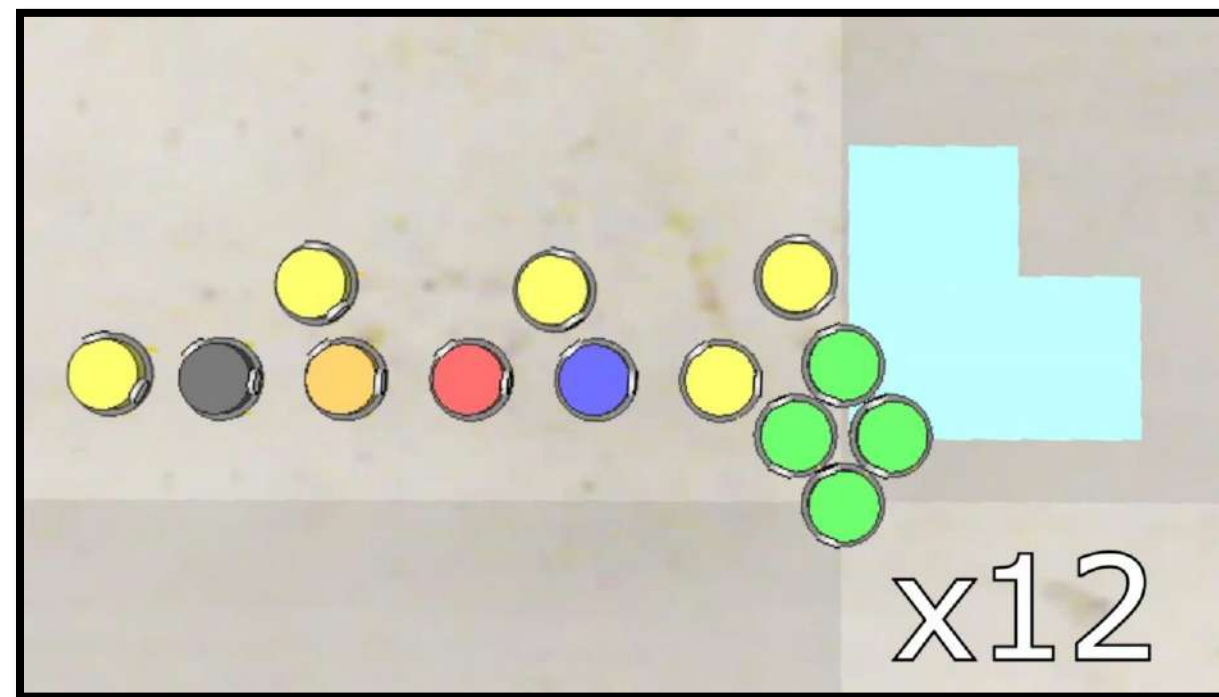


4 : 2 1.41 3 1	17 : 14 1.41 16 1
5 : 1 1.41 2 1	18 : 16 1 17 1.41
6 : 5 1 2 1.41	19 : 16 1.41 18 1
7 : 5 1.41 6 1	20 : 18 1 19 1.41
8 : 6 1 7 1.41	21 : 18 1.41 20 1
9 : 6 1.41 8 1	22 : 21 1 20 1.41
10 : 8 1 9 1.41	23 : 21 1.41 22 1
11 : 8 1.41 10 1	24 : 23 1 22 1.41
12 : 11 1 10 1.41	25 : 23 1.41 24 1
13 : 11 1.41 12 1	26 : 25 1 24 1.41
14 : 13 1 12 1.41	27 : 25 1 26 1.41
15 : 13 1.41 14 1	28 : 27 1 25 1.41
16 : 14 1 15 1.41	29 : 27 1 28 1.41

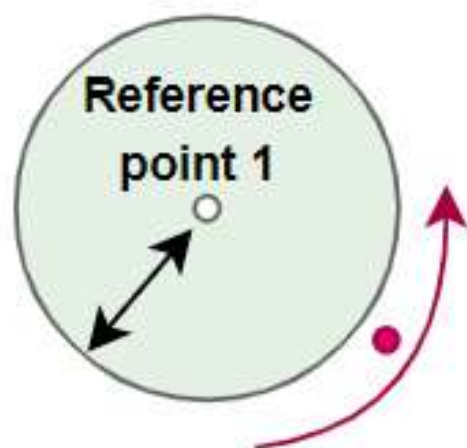
Algoritmul de formare de imagini 2

Stările în care se poate afla robotul :

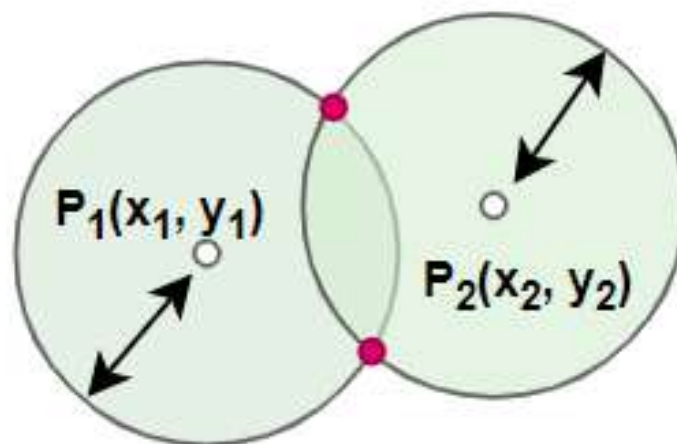
- **WAIT**
- **MOVE** -> calculează poziția
-> verifică dacă se află în interiorul poligonului
- **INSIDE_SHAPE** -> condiții de oprire:
 - Robotul părăsește aria poligonului
 - Robotul ajunge la o distanță mică față de alt robot cu același gradient
- **LOCAL_SHAPE** -> robot localizat în formă



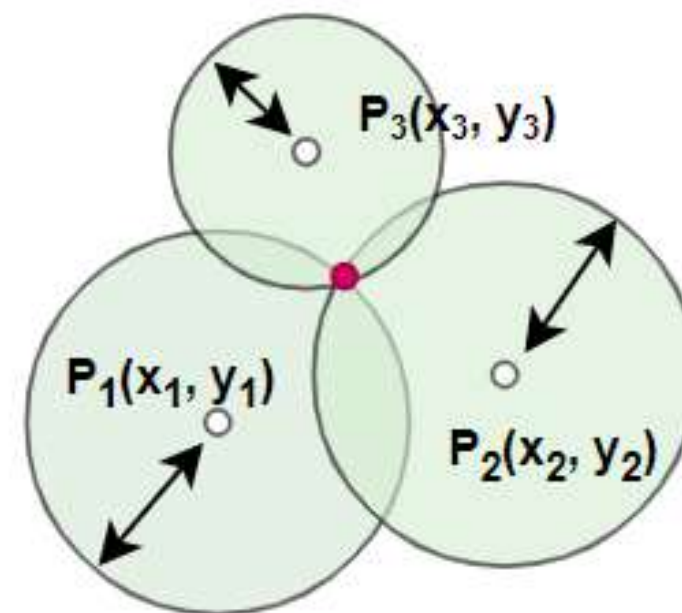
Triangulare



Punctul se poate găsi oriunde pe curcumferința cercului

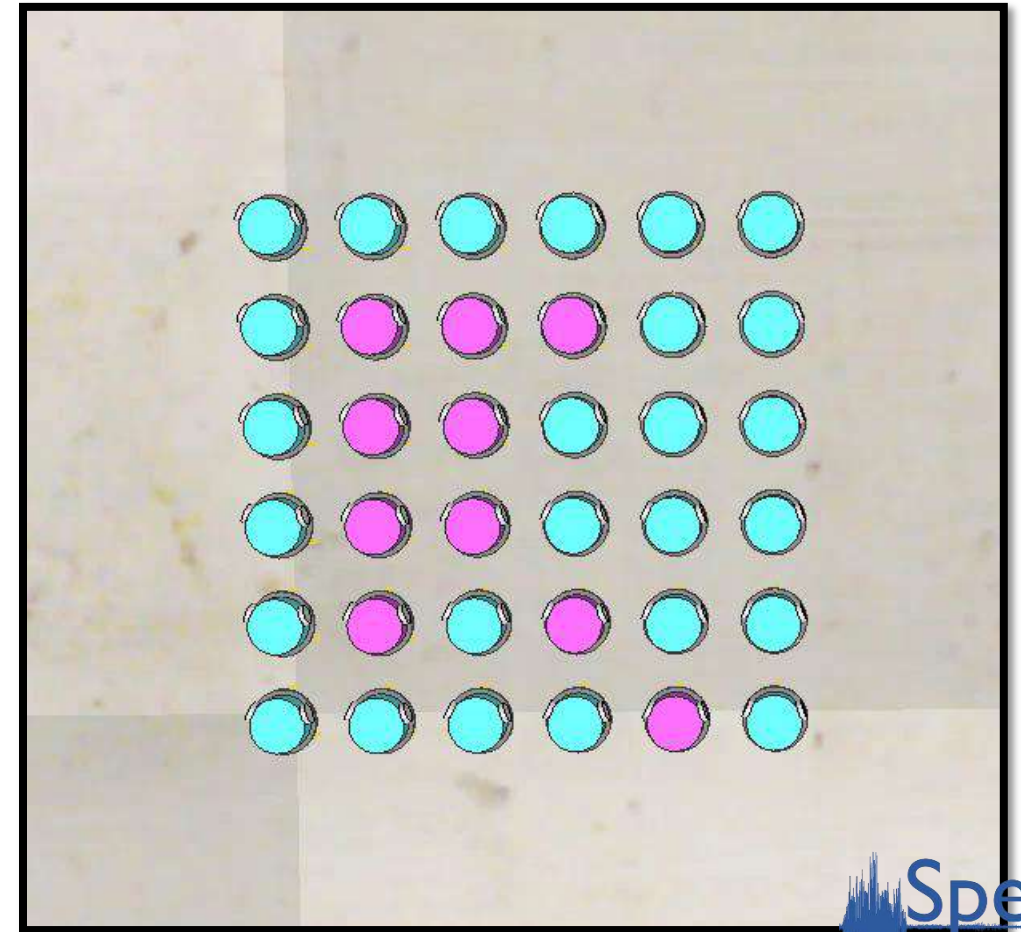
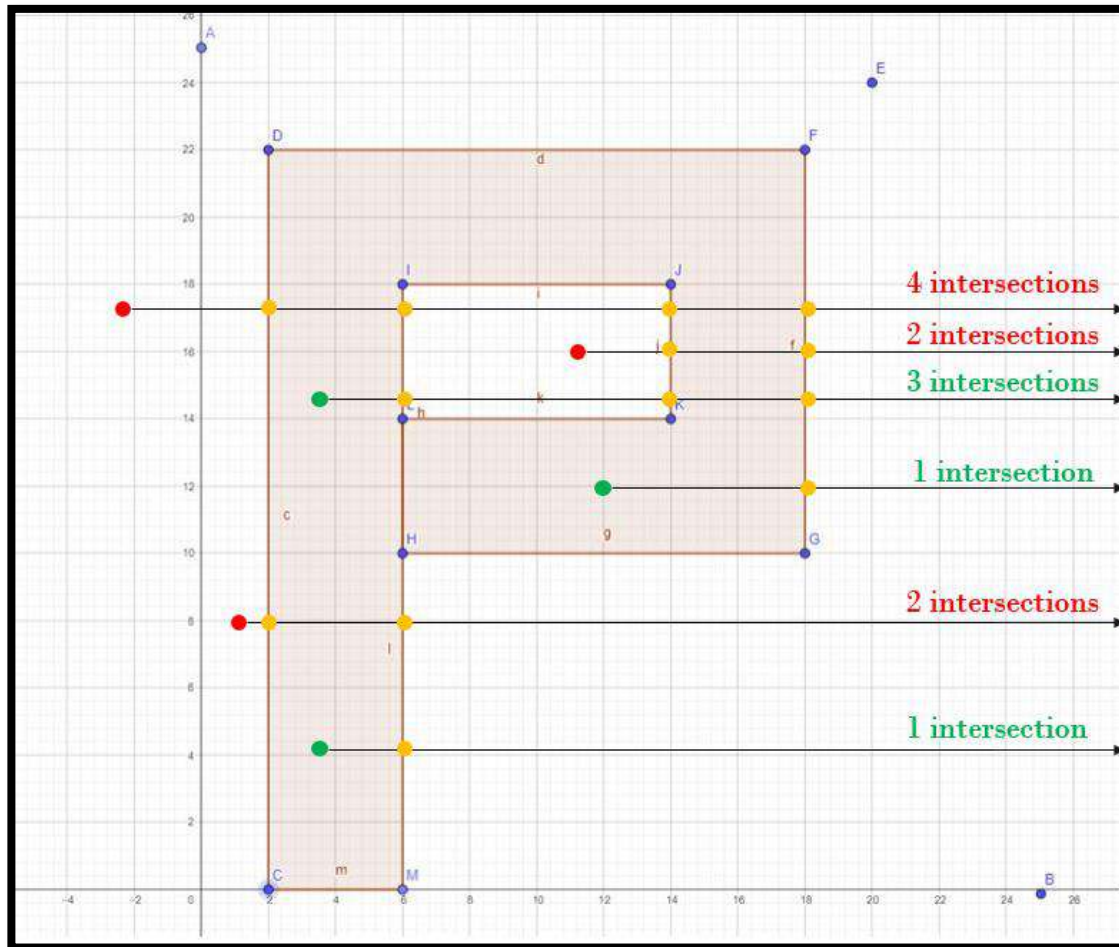


Punctul se poate găsi în una din cele două intersecții



Punctul se află la intersecția dintre cele 3 cercuri

Algoritmul de Ray-Casting



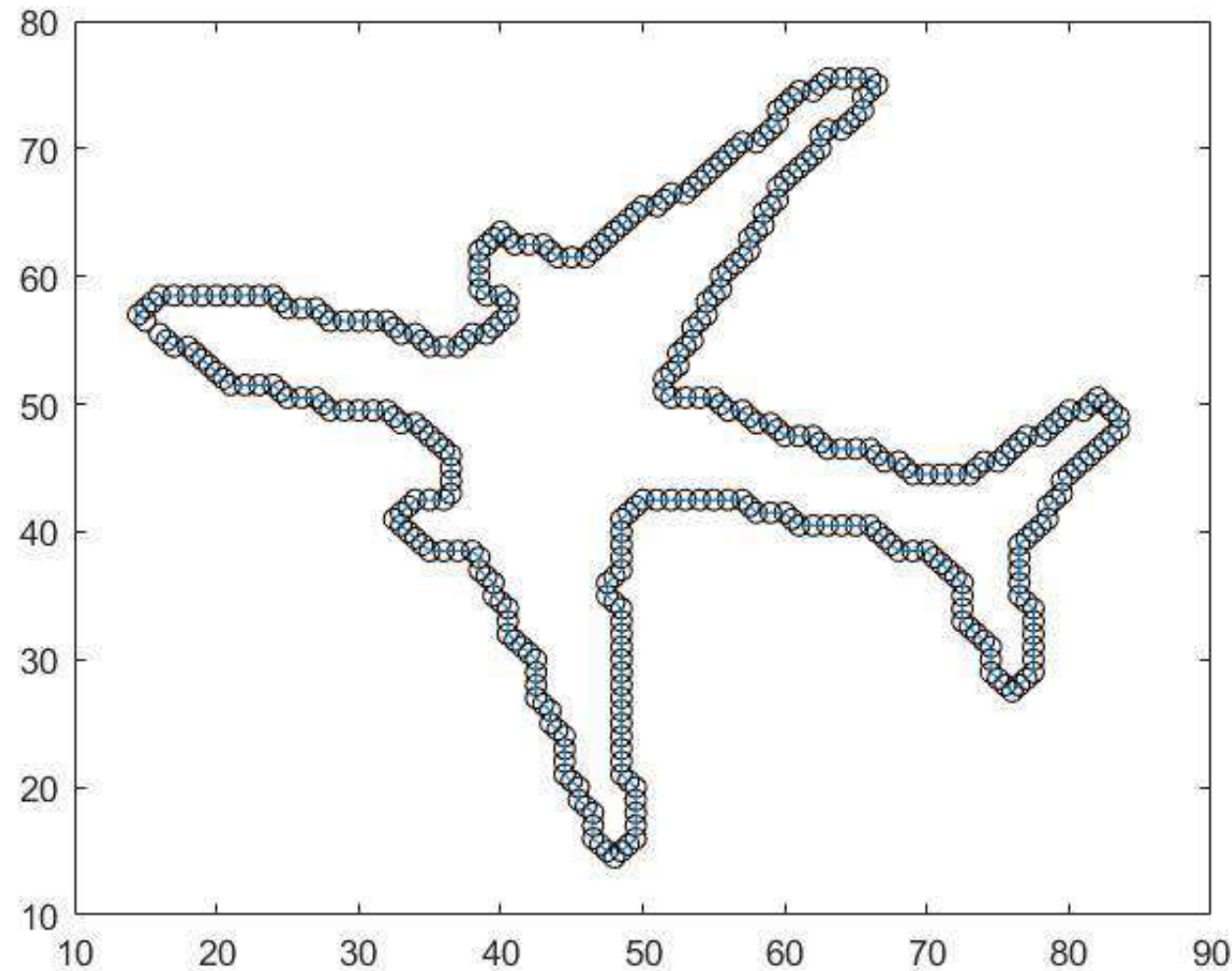
Algoritmul Ramer-Douglas-Peucker

- Algoritm folosit pentru simplificarea setului de date care va fi trimis către kiloboți



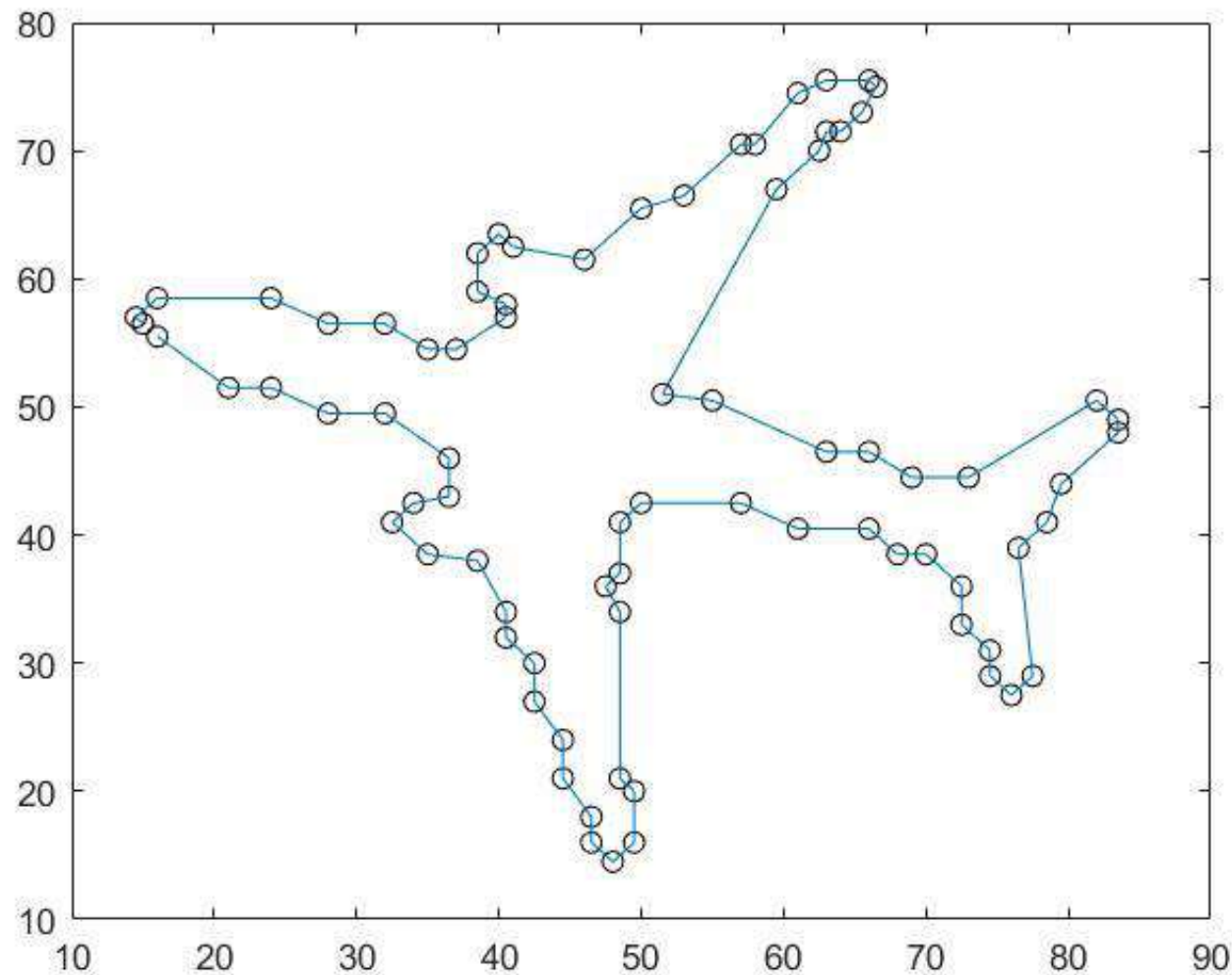
Algoritmul Ramer-Douglas-Peucker

- Setul initial:
361 elemente



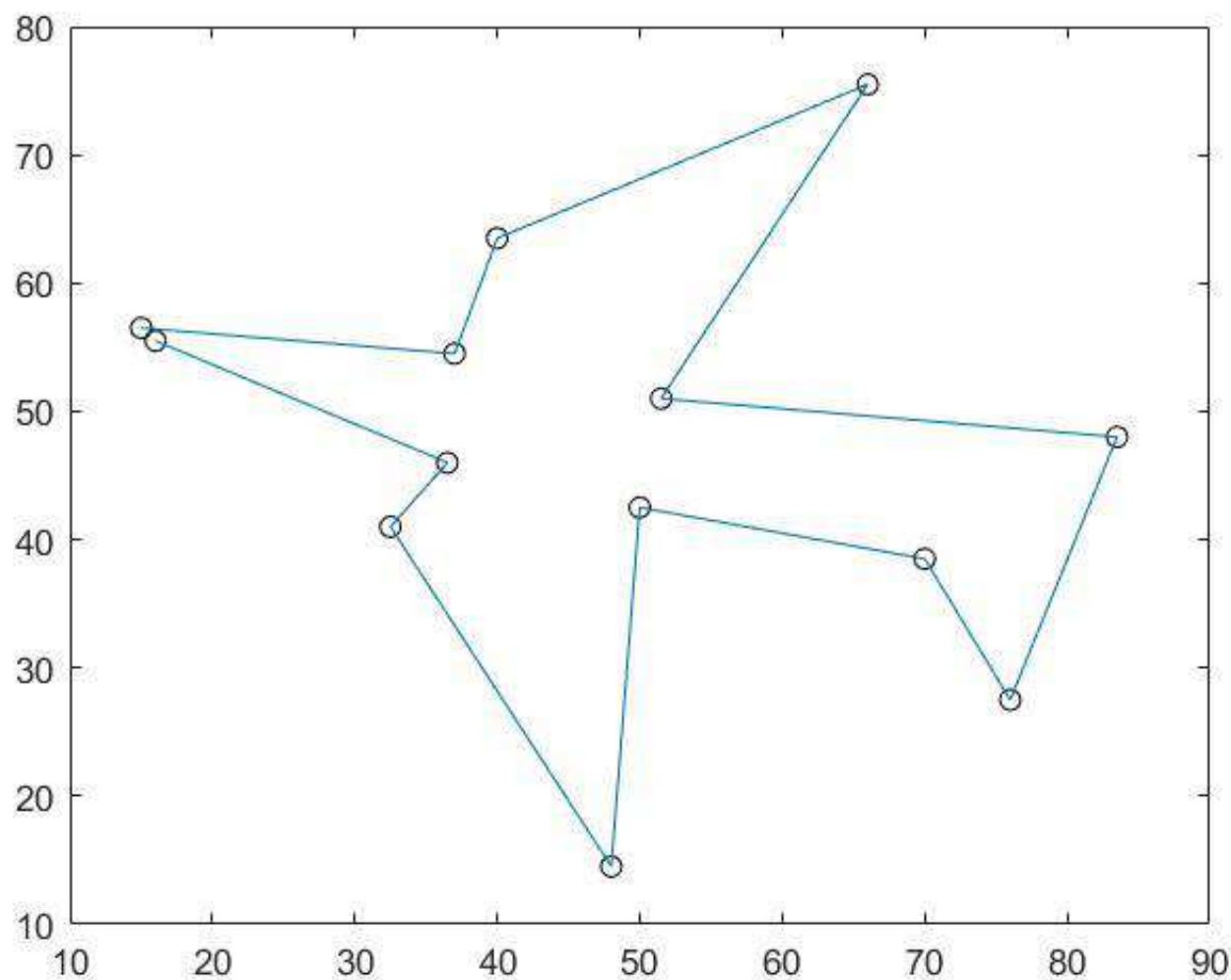
Algoritmul Ramer-Douglas-Peucker

- Epsilon = 0,5:
79 elemente



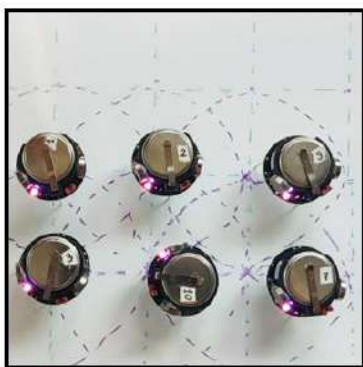
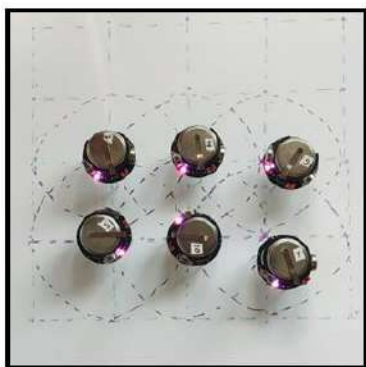
Algoritmul Ramer-Douglas-Peucker

- Epsilon = 5:
13 elemente

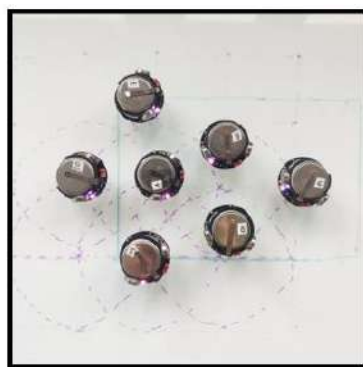
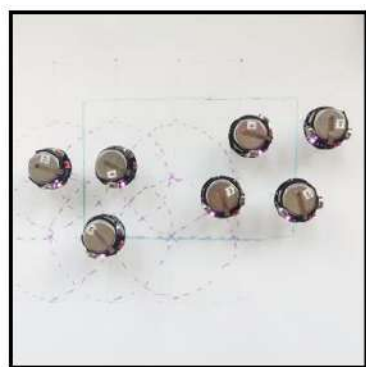
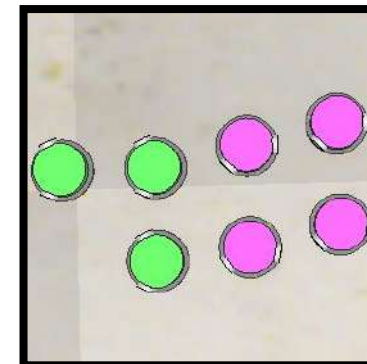
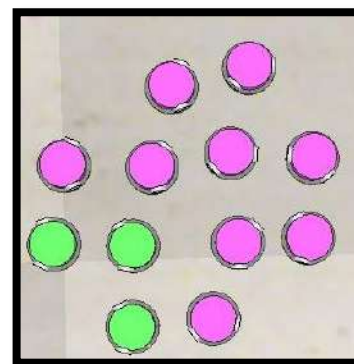


Concluzii

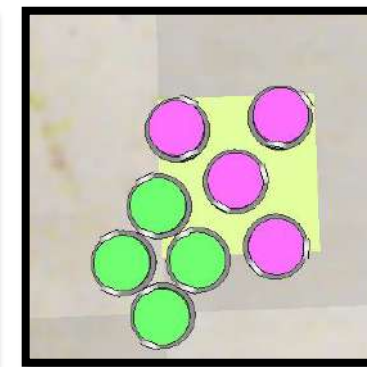
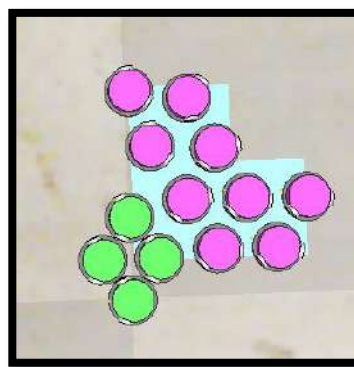
Rezultate experimentale



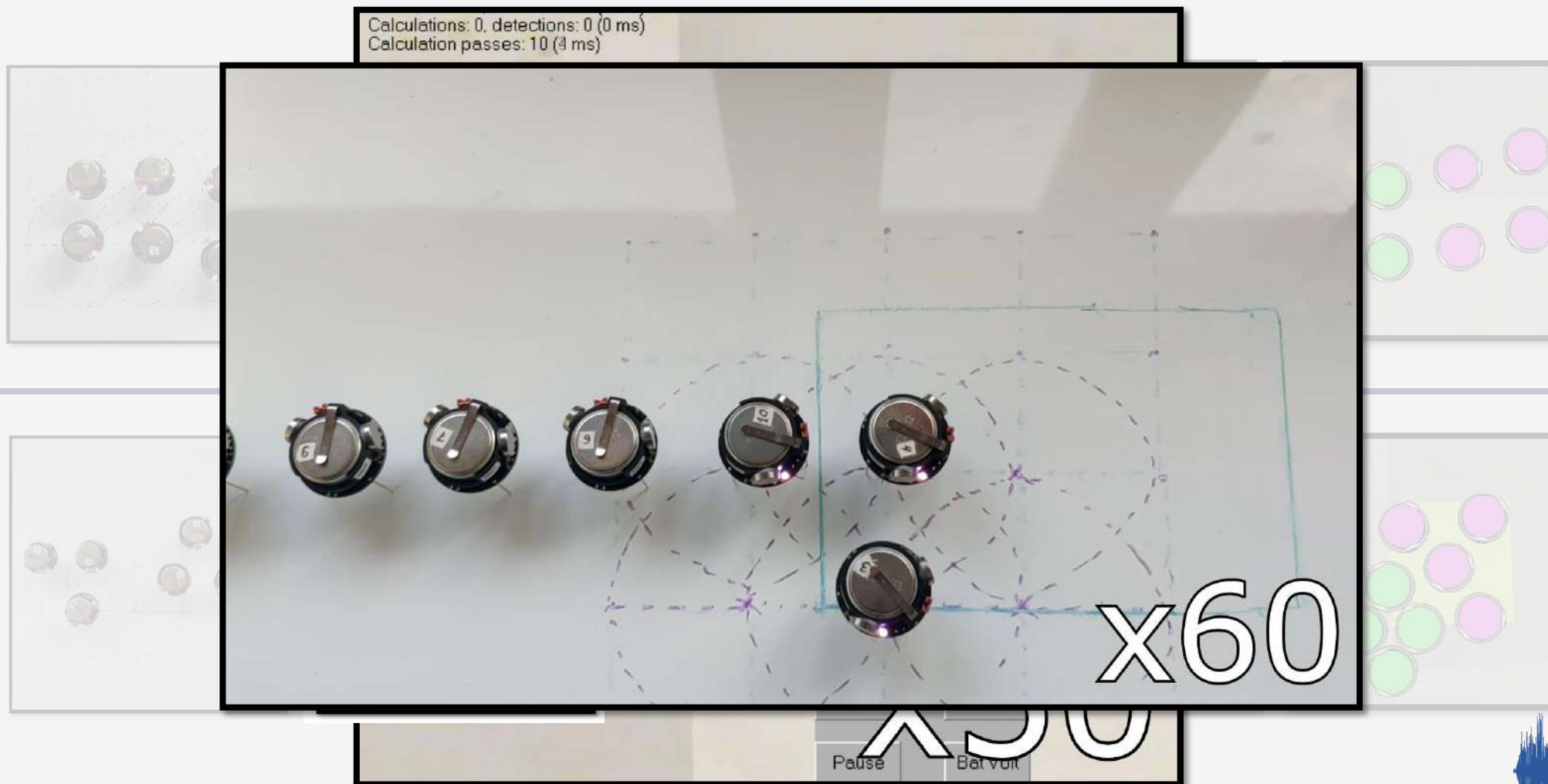
Algoritmul 1



Algoritmul 2



Rezultate experimentale



Contribuții personale

- Organizarea spațiului de lucru
- Implementarea și testarea unor comportamente de bază folosite în domeniul *swarm robotics*
- Implementarea și testarea a doi algoritmi de formare de imagini
- Compararea rezultatelor

Vă multumesc pentru atenție!