UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Disseny i simulació del comportament d'una colònia de robots

Memòria presentada per:

José Luís Lucas Palací

Grau d'Enginyeria Informàtica

Resum

El projecte consistirà en la implementació software d'un model de robot simple, amb el comportament individual semblant al d'un insecte. A partir del robot individual, es proposa la generació virtual d'una colònia d'individus, que hauran de respondre al codi implementat per tal de resoldre col·lectivament els problemes propostes per l'usuari (posicionament, desplaçament, construcció d'elements, etc.). Per a resoldre aquest tipus de problemes i comportaments, el software estar implementat en el llenguatge de programació Matlab, amb les funcionalitats de la creació de objectes del propi llenguatge.

Direcció
Pau Micó, Carlos Sastre

Convocatòria de defensa Juliol de 2017

1 ÍNDEX DE CONTINGUTS

2 Taula d'il·lustracions 5
3 Introducció 6
3.1 Antecedents 6
3.2 Objectius 6
3.3 Requeriments 7
4 Avantprojecte 8
4.1 Estat de l'art 8
4.1.2 Kilobots (Software) 14
4.2 Estudi de propostes 17
4.2.1 Proposta 1 17
4.2.2 Proposta 2 18
4.2.3 Proposta 3 19
4.3 Justificació 20
4.3.1 Proposta 1 20
4.3.2 Proposta 2 22
4.3.3 Proposta Final. 23
5 Implementació 24
5.1 Entorn de desenvolupament. 24
5.1.1 Matlab 24
5.2 Implementació pràctica 27
5.2.1 Classe Kapp. 27
5.2.2 pintaEscenari 28
5.2.3 Kbot. 29
5.3 Proves 38
5.4 Exemples de simulació: 40
6 Manual d'usuari 42
6.1 Interfície Gràfica: 42
7 Conclusió 45
7.1 Conclusions personals 45
7.2 Línies de desenvolupament posterior 47
8 Bibliografia 48
9 Annexes 49
9.1 Codi de Classe Principal i Classe pintaEscenari. 49

9.2 Codi de Classe Kbot. 49
9.3 Codi de GUI 49
9.4 Codi GUI Pop-up. 49
9.5 Scripts de Simulació 49

2 TAULA D'IL·LUSTRACIONS

Fig. 1 Kilobot 8
Fig. 2 Il·lustració de comunicació del Kilobot. 9
Fig. 3 Demostració de funcionament. 10
Fig. 4 Controladora OHC 11
Fig. 5 Distancia entre el OHC i els robots 11
Fig. 6 [A] Controlador. [B] Estació de control. [C] 25 KiloBots. [D] Estació de càrrega. 12
Fig. 7 Planta Kilobot 12
Fig. 8 Parts físiques de un kilobot 12
Fig. 9 KiloGUI executant-se en OSX 13
Fig. 10 Exemple programació Kilobot. Canvia de color cada 100ms 14
Fig. 11 Framerwork del simulador V-REP 15
Fig. 12 Paralel Computing Toolbox 17
Fig. 13 La computació en paral·lel en Matlab 18
Fig. 14 Pantalla principal de Matlab 25
Fig. 15 Esquema de la creació de les classes amb els seus objectes. 27
Fig. 16 Funcionament de la classe PintaEscenari 28
Fig. 17 Jerarquia de la classe Kbot 29
Fig. 18 Inicialització objecte 30
Fig. 19 Comportament de les funcions de moviment.. 31
Fig. 20 Esquema de funcionament de la funció Set_color. 32
Fig. 21 Funció Azimut 33
Fig. 22 Funció Set_Motor 34
Fig. 23 Esquema de Estimate_distance 35
Fig. 24 Funció Get_Voltage 36
Fig. 25 Funció Turn_Right 37
Fig. 26 Funció Turn_Left 37
Fig. 27 Exemple de generació automàtica de Kilobots en posicions aleatòries. 38
Fig. 28 Exemple de la funció "program", executant una orbita de robots 39
Fig. 29 Codi Generació Kilobots (MasterSlave) 40
Fig. 30 Generació robots (MasterSlave) 40
Fig. 31 Codi Assignació Master (MasterSlave) 40
Fig. 32 Assignació Master (MasterSlave) 40
Fig. 33 Resultat final (MasterSlave) 40
Fig. 34 Codi generació kilobots (testEstimatedicstance) 41
Fig. 35 Generació robots (testEstimatedistance) 41
Fig. 36 Simulació Orbit (testEstimatedistance) 41
Fig. 37 Finalització simulació (testEstimatedistance) 41
Fig. 38 Interfície principal 42
Fig. 39 Panell Control 43
Fig. 40 Panell Info 44
Fig. 41 Panell Informe 44
Fig. 42 Panell Script 44

3 INTRODUCCIÓ

3.1 ANTECEDENTS

En la naturalesa, grans grups d'elements individuals poden cooperar i un comportament global altament complexa. En el camp de la robòtica, els investigadors utilitzen la intel·ligència col·lectiva que hi ha en la naturalesa com a font d'inspiració per crear una sèrie de robots diminuts, capaços de comportar-se com una colònia de formigues, abelles i cèl·lules, treballant juntes per a un be comú. No obstant això, encara existeix una bretxa substancial entre els dissenys conceptuals i els sistemes realitzats. La creació de sistemes d'enginyeria amb la capacitat de simular, planteja un repte en el disseny de la algorítmica i sistemes fixes que puguen funcionar a eixa escala. El comportament col·lectiu dels animals es descriu com a un comportament de grans grups d'animals, amb comportaments similars, que són capaços de realitzar una tasca en grup, beneficiar-se de la transferència d'informació en el grup, el procés de decisió en grup, la locomoció i sincronització pròpia.

La robòtica d'eixam és un camp de la robòtica múltiple en la qual es coordina un gran número de robots de manera distribuïda i descentralitzada. Es basa en el us de les regles locals, i els robots simples en comparació amb la complexitat de les feines, inspirat pels insectes socials. Gran número de robots simples poden realitzar una feina complexa d'una forma més eficient que un sol robot, donant robustesa i flexibilitat al grup.

3.2 OBJECTIUS

L'objectiu principal d'aquest projecte és l'estudi dels robots d'eixam Kilobots [1], desenvolupats per la unitat de SSRG[2] de la universitat de Harvard. Es tracta d'un eixam de mil vint i quatre robots dissenyats per a que cada un siga programat de tal manera que s'estudie el comportament col·lectiu d'eixams autònoms a gran escala. Cada un d'ells té les capacitats bàsiques d'un robot autònom (Control programable, locomoció de base i comunicació local).

El principal objectiu es la creació d'un simulador mitjançant el llenguatge de programació Matlab, utilitzant les ferramentes de computació paral·lela i distribuïda. El plantejament és simple; la creació d'un gran numero de robots sense un alt cost de processament.

Estudiarem els diferents simuladors que hi han actualment al mercat per veure les limitacions i s'intentarà millorar en lo màxim possible.

[1] 1 robot: 14$ 1024 robots: 14336$
[2] Self-organizing Systems Research Group

3.3 REQUERIMENTS

Per a fer els estudis d'aquests robots, es realitzaran les diferents proves als simuladors que hi ha, perquè no hem pogut tindre-los físicament ja que el preu per unitat és mínim però el cost de l'eixam és alt (cada robot val aproximadament 14$).

Amb el llenguatge de programació Matlab, es dissenyarà una interfície gràfica, on es crearà un simulador. En la creació d'aquest simulador, estudiarem els diferents paradigmes de programació, tan com la programació orientada a objectes, com la programació distribuïda i paral·lela d'aquest llenguatge de programació.

4 AVANTPROJECTE

4.1 ESTAT DE L'ART

Fig. 1 Kilobot

4.1.1.1 Kilobots (hardware)

El Kilobot és un robot de "low cost", de fàcil us per a sistemes avançats de desenvolupament de robòtica d'eixam (swarm robots[3]), que es poden programar per a realitzar funcions útils mitjançant la coordinació de les interaccions entre molts individus. Aquests eixams estan inspirats en insectes socials com colònies de formigues que poden realitzar de forma eficient moltes tasques de cerca, transport i coordinació en entorns molt complexos. [1] [2]

SDASH[4] és el algoritme desenvolupat per a que funcionen aquests robots de forma col·lectiva, és un mètode que s'executa en cada robot, per a crear la formació desitjada en el moment de la programació. Cada robot té una programació individual, però a la volta igual que el seu veí. Es comuniquen entre ells mitjançant missatges, però en cap moment saben on estan els seus veïns. Una de les principals característiques del seu baix cost és aquesta, que no tenen sensors on localitzar la seva posició dins de l'espai, simplement detecten on estan els veïns utilitzant els infrarojos i estimant la distancia, poden executar el seu propòsit. Aquest comportament és bastant complex i requereix que els robots tinguen la capacitat de:

1. Moure cap avant.
2. Girar.
3. Comunicar-se en els veïns.
4. Mesurar la distancia entre veïns propers.
5. Tindre la suficient memòria per a executar el SDASH.

Per altra banda per a millorar la capacitat dels Kilobots i la seva capacitat de operar en grans col·lectius, així com la plataforma robòtica siga més versàtil, s'afegiren altres requisits addicionals:

6. Mesurar nivells de llum ambient.
7. Mostrar algun estat per a l'ajuda de la seva depuració.
8. Permetre operacions escalables.

[3] https://wyss.harvard.edu/technology/programmable-robot-swarms/
[4] Scalable Distributed Selfassembly and Selfhealing, Mètode distribuït per a controlar cada robot en un col·lectiu robòtic, per a controlar la formació desitjada.

4.1.1.2 Moviment

El Kilobot utilitza dos motor de vibració en forma de monedes segellades. Quan s'activa un d'aquests motors, la força centrípeta generada pel motor es converteix en forces cap a endavant[5]. La vibració d'un sol motor provoca que el Kilobot rote sobre el seu eix, per això hi duen dos motors, un oposat a l'altre. Mitjançant el control de la magnitud de la vibració dels dos motors (independents un de l'altre), el robot pot moure en un ranc continuo de rotació en sentit horari, recte o en sentit contra-horari. Aquests motors permeten que el robot es moguí aproximadament $1\ cm/s$ i rotar $45º deg/s$.

4.1.1.3 Comunicació

Per a comunicar-se amb els robots veïns, cada Kilobot té un transmissor LED d'infrarojos i un receptor de fotodíode d'infrarojos, que s'encontra en el centro de la PCB i apunta directament cap avall, la superfície té que ser quelcom brillant i totalment blanca.

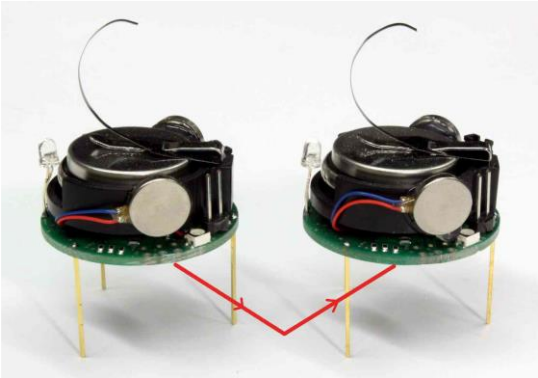


Fig. 2 Il·lustració de comunicació del Kilobot.

Tant el transmissor com el receptor tenen una emissió isotròpica o patró de recepció, que permet al robot rebre i enviar missatges en totes les direccions, a demés els sensors son de gran angular, amb un angle de 60º poden comunicar-se a una velocitat de 30Kb/s fins a 10 cm de distancia. Com tots els robots utilitzen el mateix canal de comunicació infraroja quan hi ha una gran nombre de robots, s'utilitza l'estàndard CSMA/CA[6], per tant hi ha una reducció de ample de banda degut a les col·lisions.

El sensor IR també és l'encarregat de mesurar la distancia entre els seus veïns, gracies a la detecció de la intensitat de llum infraroja entrant. Aquesta intensitat de llum també es veu afectada per el soroll i les diferencies de fabricació, cosa que produeix una precisió de ± 2mm.

4.1.1.4 Controlador

El controlador utilitzat és el microprocessador Atmega326 que corre a 8MHz i té una memòria de 32K, la suficient potència per a controlar l'algoritme SDASH. Alguna de les característiques d'aquest controlador es que té la capacitat de la modulació de polsos (PWM) per a controlar la vibració dels motors i convertidors de 10 bits analògic-digitals per a poder mesurar la intensitat de la llum infraroja. És auto programable i té un mode de baixa potencia. Aquest controlador utilitza el llenguatge de programació C, que permet un desenvolupament rapit i fàcil de comportaments del robot.

[5] Principi Slip-Stik.
[6] Accés múltiple per detecció de portador i prevencions de col·lisions.

4.1.1.5 Alimentació

Per a alimentar el robot, té una bateria de 160mAh de ions de liti a 3,4V. Aquesta bateria pot alimentar el robot de 3 a 24 hores, depenent del seu nivell d'activitat.

4.1.1.6 Demostració de funcionament

Per demostrar les capacitats del Kilobot, es demostra allò evident, la capacitat de moure's dins d'un entorn, comunicar-se amb els veïns i mesurar la distancia entre els veïns.

1a Demostració:

Òrbita (https://www.youtube.com/watch?v=EOEh9xnLB_0)

Com mostra la figura 3[A]. Un Kilobot orbita al voltant de un Kilobot estacionat, on el Kilobot que orbita està tot el temps enviant la informació d'on està el seu veí, este missatge també li serveix per a saber la distancia d'un robot a l'altre i així poder calcular l'orbita que ha de fer gràcies al càlcul PD[7] que ajusta en tot moment la intensitat dels motors.

2a demostració:

"U"

Com mostra la figura 3[B] El Kilobot negre es basa en els Kilobots estacionats de color verd, per a fer el camí desitjat, mitjançant la informació que li comuniquen els altres robots. Una volta el robot es mou coneix la seua posició gràcies a la mesura dels sensors i el càlcul de les coordenades.

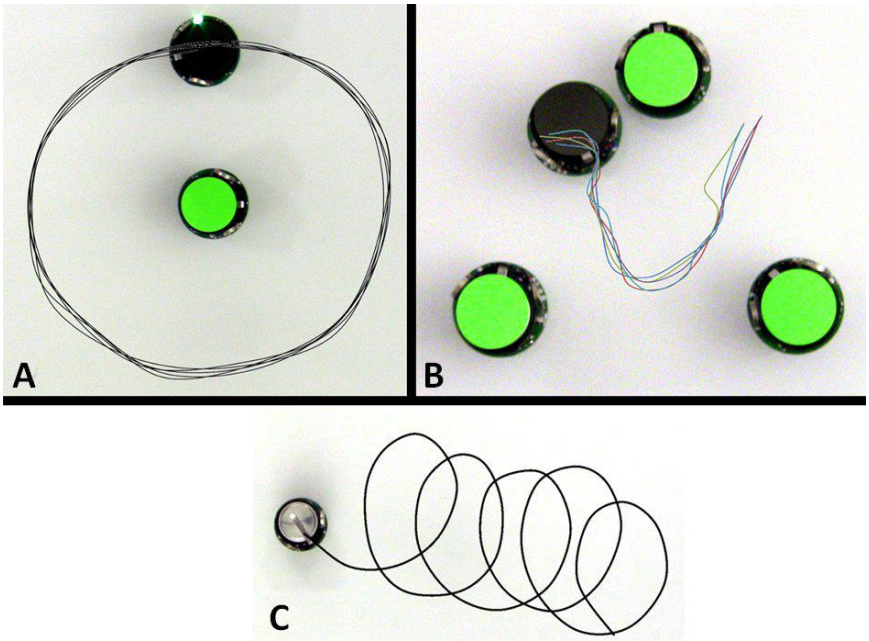


Fig. 3 Demostració de funcionament.

[7] Controlador Proporcional Derivatiu. $u(t) = K(e(t) + Td\frac{de(t)}{dt})$ Encarregat de Controlar i ajustar el error de posicionament.

4.1.1.7 KiloBot Controller

Amb aquest gran col·lectiu de robots, pot ser tediós o inclús impossible treballar amb ells si es requereix un operador humà per a interactuar amb cada un d'ells (engegar o apagar el robot, programar-los mitjançant cables o carregar-los). Per això els Kilobots no requereixen de ninguna atenció. És a dir, els robots són escalables. Per a poder fer això s'utilitza un controlador. Aquest controlador pot enviar missatges mitjançant infrarojos, i programar-los tots a la volta, a més de controlar-los.

Gràcies a aquest controlador (OverHeadController), també podem controlar la tensió de les bateries, reiniciar i pausar.

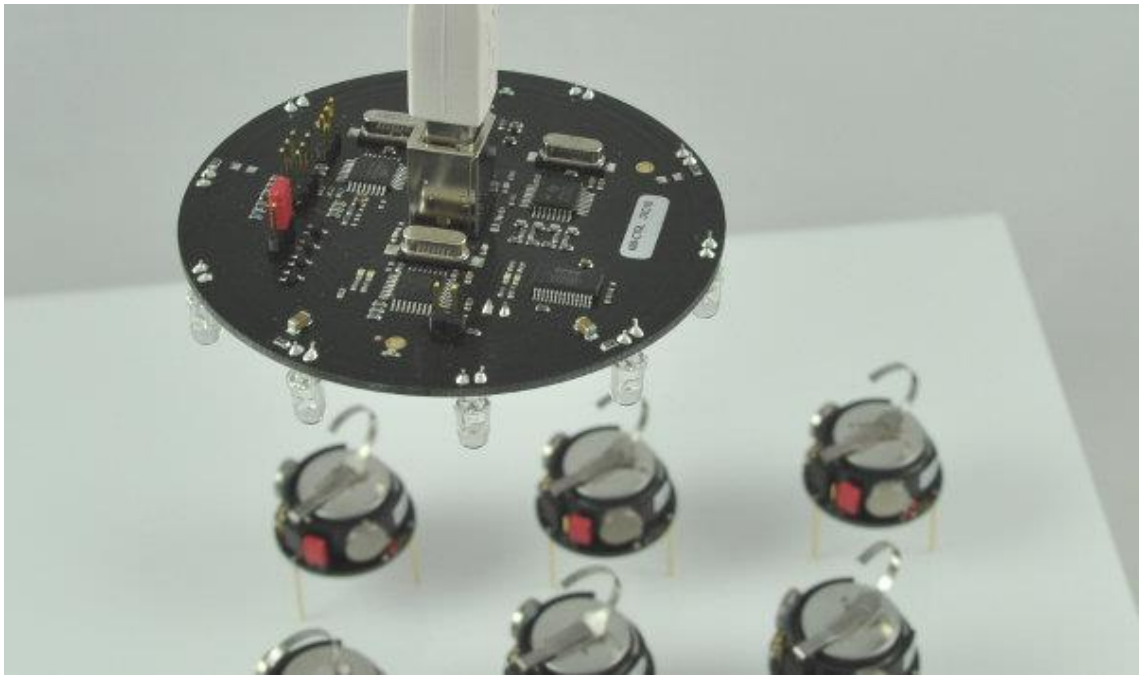


Fig. 4 Controladora OHC

Aquest controlador es deu col·locar per damunt dels Kilobots, a una distancia aproximada de 1 metre. Els robots per baix del OHC en aproximadament un metre de diàmetre podran rebre la informació del controlador mitjançant infrarojos.

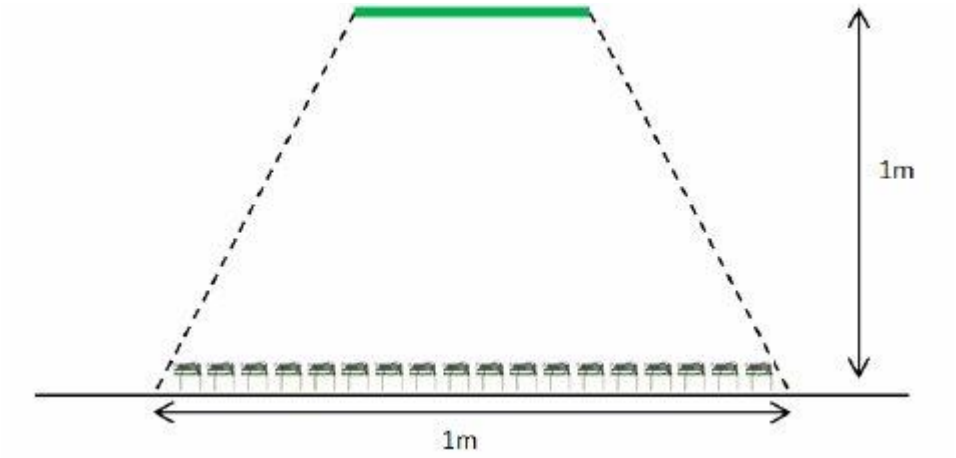


Fig. 5 Distancia entre el OHC i els robots

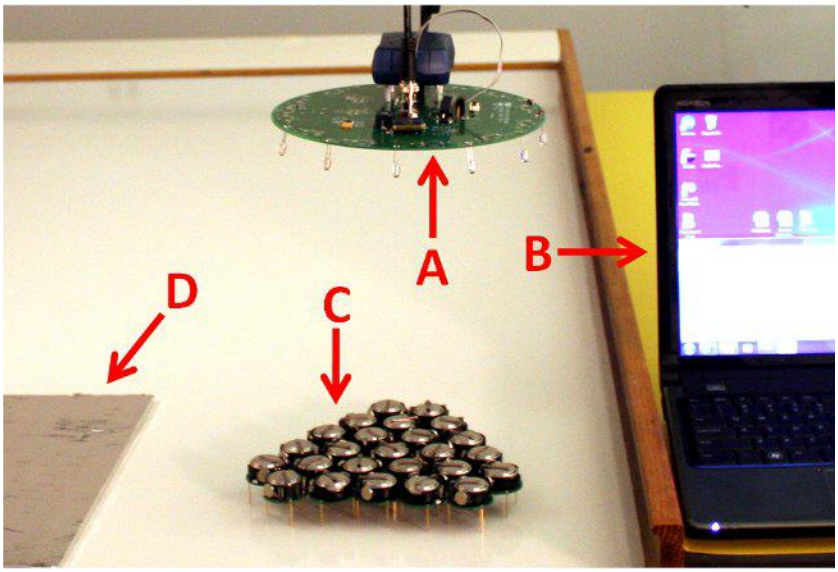


Fig. 6 [A] Controlador. [B] Estació de control. [C] 25 KiloBots. [D] Estació de càrrega.

4.1.1.8 Característiques principals:

- Low Cost
- Petit, 33 mm de diàmetre.
- Control nivell de motors (255 nivells)
- Capacitat de comunicació amb els seus veïns (7cm)
- Càlcul de distancia entre ells
- Sensors de llum ambiental
- RGB LED de senyalització
- Fàcil de manipular. Hi ha un controlador principal que té el control dels robots en tot moment.

Les especificacions del Kilobot a nivell de hardware són les següents:

- Processador: ATmega 328p (8bits @ 8MHz)
- Memòria:
  - 32 KB Flash
  - 1KB EEPROM
- Bateria / Autonomia: Recarregable Li-Ion 3.7V / 3-10 hores continues, 3 mesos en mode "Sleep".
- Comunicacions: IR(fins a 7cm, a 32kb/s i 1 KByte amb 25 robots), serial (256000 baud).
- Sensors:
  - 1 IR.
  - 1 sensor de llum.

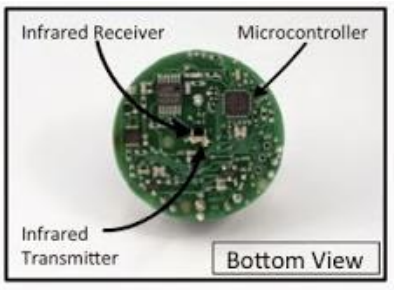


Fig. 7 Planta Kilobot

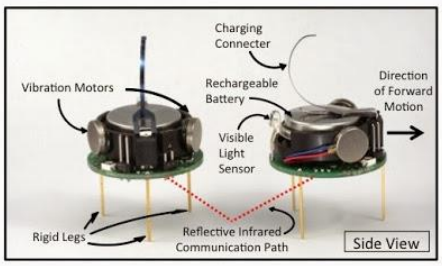


Fig. 8 Parts físiques de un kilobot

4.1.1.9 Entorn de programació

L'entorn de programació ve donat per la universitat de Harvard i el SSR lab. Anomenat Kilobotics. Aquest entorn té l'objectiu de fer la programació dels robots lo mes fàcil i assequible possible per a qualsevol plataforma(Linux, Mac, Windows)[3]. Conte diverses peces:

1. KiloGUI: Esta GUI es per a utilitzar el controlador per a iniciar els Kilobots, carregar-los de nou programes, calibrar els robots, etc.

2. Biblioteca Kilobot: Els Kilobots utilitzen C com a llenguatge de programació (AVR C) i necessiten una biblioteca especifica per a proporcionar les funcions necessàries per a controlar els motors, sensors i comunicació.

3. Editor i Compilador: L'editor esta basat en web i permet que qualsevol puga escriure i compilar programes per als Kilobots, utilitzant servidors d'Amazon per a la compilació i Dropbox pera emmagatzemar arxius. També es pot instal·lar un editor en local.

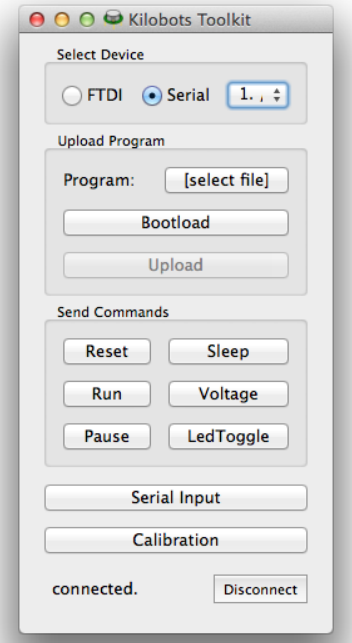


Fig. 9 KiloGUI executant-se en OSX

4.1.1.10 Llibreria de programació

La principal llibreria que s'utilitza per a programar aquests robots és l'anomenada KiloLib.h [8]. Aquesta llibreria és el codi font dels Kilobots, i proporciona les funcions de inicialització del hardware i la seva programació .

Per a la seva inicialització s'utilitza la funció kilo_init(), al executar-la, inicialitza totes les capacitats del robot, incloent la calibració del hardware, els temporitzadors, la configuració dels ports, la inicialització dels convertidors DAC, el registre de missatges i la inicialització de les interrupcions del sistema.

Per a la programació s'utilitza la funció kilo_start() que utilitza el paradigma de programació dirigida per esdeveniments[9]. Aquesta API esta prevista de diferents funcions:

- estimate_distance(): Estima la distancia en mm sobre la base de mesura de intensitat de senyal del sensor.
- get_amientlight(): Llig la quantitat de llum ambiental.
- get_voltage(): Llig la tensió en Voltatge de la bateria.
- get_temperature(): Llig la temperatura de la llum del Kilobot
- message_crc(): Comprova la validesa del missatge.

[8] Kilobot Library (https://www.kilobotics.com/docs/kilolib_8h.html)
[9] La programació es similar a la de Arduino. Hi ha una configuració inicial i un bucle d'execució.

- rand_hard(): Genera números aleatoris per hardware.
- rand_seed(): Genera números aleatoris en base a una seqüència.
- rand_soft(): Generador números aleatoris per software.
- set_motors(): Mitjançant el PWM[10] del controlador, aquesta funció posa dos valors entre 0 i 255 separats per comes per a controlar la velocitat dels motors.
- set_color(): Assigna un color als LED.

En la figura 10 es un exemple de programació de un Kilobot, on en el `void setup( )` es col·loca el codi de configuració, en el `void loop( )` el codi d'execució i en el `int main( )` es crida a `kilo_init ( )` per a inicialitzar i en `kilo_start(setup, loop)`, per a executar-los.

```
#include "kilolib.h"

void setup() {
    // put your setup code here, will be run once at the beginning
}

void loop() {
    // put your main code here, will be run repeatedly
    set_color(RGB(1,0,0));
    delay(100);
    set_color(RGB(0,1,0));
    delay(100);
}

int main() {
    kilo_init();
    kilo_start(setup, loop);

    return 0;
}
```

Fig. 10 Exemple programació Kilobot. Canvia de color cada 100ms.

4.1.2 Kilobots (Software)

En aquest punt anem a estudiar els diferents simuladors dels Kilobots que hi han al mercat.

4.1.2.1 V-REP

Simulador desenvolupat per Coppelia Robotics. El simulador de robots V-REP[11], amb entorn de desenvolupament integrat, es basa en una arquitectura distribuïda: Cada objecte / model pot ser controlat individualment mitjançant un script, un plug-in, un node de ROS, un client API remot o una solució personalitzada. Açò fa que V-REP siga molt versàtil i ideal per a aplicacions multi-robots. Els controladors poden ser escrits en C / C++, Python, Java, Lua, Matlab o Octave. [4]

Aquest simulador és el simulador oficial per a la programació de Kilobots.

https://www.youtube.com/watch?v=Bk7aDcuFzS4

El model de programació de Kilobots en aquest simulador s'implementa mitjançant script en LUA, i utilitzant mòduls de execució de C/C++.

Limitacions:

[10] Pols de Modulació de Amplada.
[11] Virtual Robot Exerimentation Plataform

- Els script corren sobre una màquina virtual que retarden la simulació.
- Depenen del hardware que disposem en la màquina de simulació.
- Moltes de les funcions dels Kilobots no estan implementades.
- En entorns complexes amb molts robots la simulació pot anar lenta.

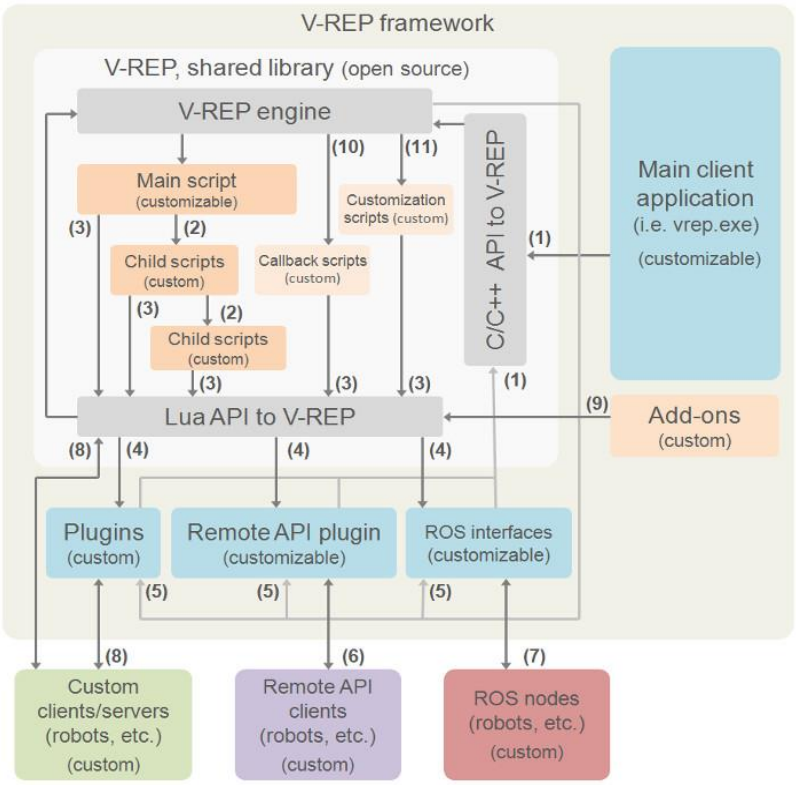


Fig. 11 Framerwork del simulador V-REP

4.1.2.2 Kilobot App[12]

Simulador de Kilobots i robòtica "swarm" dissenyat per Antti Halme / ICS/ Alto University.

Té dos ferramentes principals: KBSimulator, com el seu propi nom diu és el simulador dels Kilobots, i el KBDesigner, que és un editor simple de disseny algorítmic.

- KBSimulation és un entorn de simulació senzilla, que s'utilitza principalment en disseny de prototips de control i algoritmes per a col·lectius de eixam Kilobots.
- KBDesigner és una ferramenta en fase alfa per al disseny de patrons col·lectius de Kilobots. Aquesta ferramenta està molt limitada ja que la generació dels robots és arbitraria i limita el control necessari dels patrons del eixam.

L'aplicació permet la manipulació de les variables més rellevants que van des del esquema de colors fins al comportament del robot.

Limitacions:

- Els patrons de simulació estan pre-definits.
- Està discontinuat.

[12] https://github.com/ajhalme/kbsim

4.1.2.3 Kilombo

És un simulador de Kilobots per a permetre la investigació eficaç en robòtica d'eixam. Desenvolupat per Fredrik Jansson, Matthew Hartley, Martin Hinsch, Tjelvar Olsson, Ivica Slavkov, Noemí Carranza [5].

Està programat en C, i permet a les persones que treballen en els Kilobots a accelerar la programació i depuració dels Kilobots. El simulador compila el mateix codi que s'executa en els Kilobots físics, eliminant la espera de la execució del codi.

Limitacions:

- Esta programat seqüencialment.
- No té simulació paral·lela.

4.2 ESTUDI DE PROPOSTES

4.2.1 Proposta 1

Es proposa a Matlab com a llenguatge de programació per a desenvolupar el simulador, ja que es un entorn de programació enfocat a resoldre problemes en enginyeria mitjançant el càlcul computacional de matrius. Matlab té una ferramenta per a el càlcul computacional en paral·lel. Ja que la creació de cada robot és independent entre ells es proposa estudiar si es pot realitzar el software mitjançant esta ferramenta. [6] [7] [8]

La computació paral·lela en Matlab es basa en el càlcul de matrius gràcies a les capacitats funcionals de les maquines actuals.

Característiques principals de la computació paral·lela en MATLAB:

- Bucles paral·lels mitjançat funcions "parfor" per a executar bucles en paral·lel en múltiples processadors.
- Suport GPU NVIDA CUDA
- Ple ús de processadors multi-nucli de escriptori a traves de Workgroups que s'executen localment.
- Clústers i suport de xarxa.
- Execució interactiva i blocs d'aplicacions paral·leles.
- Vectors distribuïts i de programació únic de dades múltiples (simd[13]) són els encarregats del maneig de dades i dels algoritmes en paral·lel.

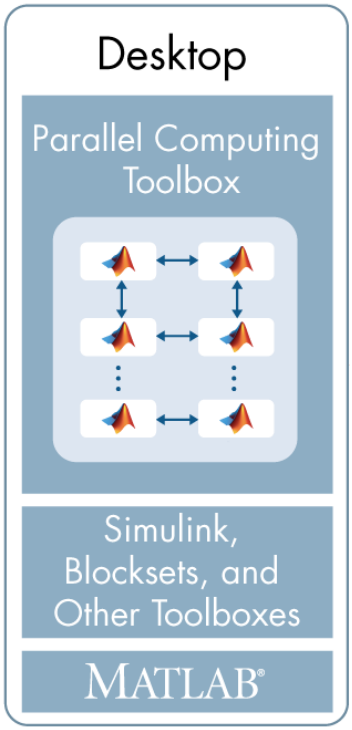


Fig. 12 Paralel Computing Toolbox

Diferents paradigmes de la programació paral·lela en Matlab:

1. Pmode: Sessions interactives paral·leles, permet a l'usuari executar un treball en paral·lel utilitzant 4 "workers" que assisteixen al client principal. Si el hardware té múltiples processadors, s'activaran tants "workers" com processadors hi ha. Utilitza la memòria compartida.
2. Spmd: Igual que el pmode però per línia de comandaments.
3. Drange: És el controlador de bucle de les dos formes anteriors. Es considera una operació de tasques paral·leles.
4. Parfor: És un bucle for paral·lel que són distribuïdes segons el índex del bucle. No depèn del entorn spmd. Permet la reducció de operacions. Es considera una operació de dades paral·lels.

[13] L'ús del SIMD en Matlab es mol complexa.

4.2.2 Proposta 2

L'altre paral·lelisme en Matlab és utilitzar el Toolbox de computació distribuïda. Este Toolbox es va dissenyar per a aprofitar les xarxes de computadores on hi ha un planificador, i les altres son treballadores. Els treballadors reben un programa del planificador, els executa i torna el resultat al mateix. El Toolbox de computació distribuïda ha evolucionat fins a l'actualitat, de manera que es permet simular treballadors locals dins de una computadora amb un processador de varius nuclis. D'esta forma cada nucli treballa sobre un programa (o cicle for) i torna un resultat al acabar.

El model de càlcul s'assembla a MIMD.

- Múltiples Instruccions, múltiples Dades.
- Cada processador executa codi de forma asíncrona e independent.

Aquest paral·lelisme permet la computació paral·lela en clústers, podem distribuir les tasques en distints "workers" i és un paradigma similar a el MPI.

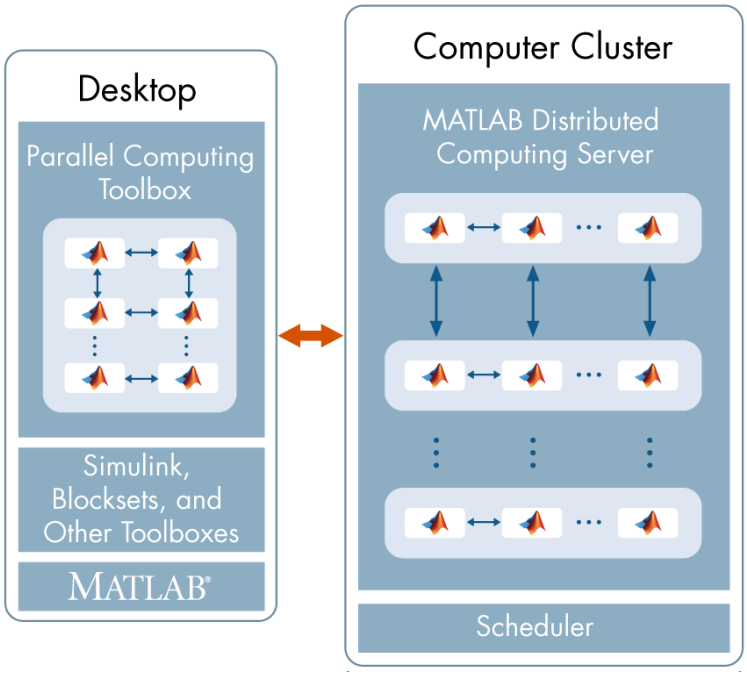


Fig. 13 La computació en paral·lel en Matlab. Al utilitzar el Paralel Computing Toolbox. Es pot escalar de un equip d' escriptori amb múltiples nuclis que aprofiten els GPUS i escalar-los fins a un Cluster (amb MATLAB Distributed Computing Server)

4.2.3 Proposta 3

Programar el simulador de manera seqüencial utilitzant el paradigma de programació orientada a objectes.

Matlab permet utilitzar la programació orientada a objectes en el seu llenguatge. Aquesta tècnica pot simplificar la programació de tasques que impliquen estructures especialitzades de dades o un gran nombre de funcions que interactuen amb un tipus especial de dades. Recolzant-se en les diferents tècniques d'aquest paradigma, es poden programar diferents programes.

4.3 JUSTIFICACIÓ

4.3.1 Proposta 1

Depenent del paradigma de la programació paral·lela ens podem trobar algunes limitacions.

Limitacions de PMODE:

- No poden utilitzar ferramentes gràfiques.
- No es poden fer simulacions interactives.
- No es pot integrar lliurement treballs en sèrie i en paral·lel.

Limitacions spmd:

- Funcions animades.
- Funcions anònimes.
- No suporta ferramentes gràfiques.

Limitacions Drange:

- Indexació explicita de matrius.
- Soles funciona per al control de bucles.

Aquesta proposta ens limita molt a nivell de simulació en temps real, ja que en molts paradigmes de la programació paral·lela no ens deixa la utilització de les interfícies gràfiques, sent molt complicat mostrar el resultat.

Com bé es el propòsit de Matlab, aquestes ferramentes estan proposades per al gran tractament de dades sobre matrius. Cada dada que entra en aquest llenguatge de programació s'interpreta com a un vector i el seu tractament es com a tal.

La pròpia documentació de Matlab aconsella aquestes tècniques per a tasques molt grans, utilitzar aquestes ferramentes en tasques xicotetes poden resultar un problema de temps, de consum de recursos i de control de tasques. [9]

Una de les altres limitacions és la compra d'una llicència del programa Matlab amb la seua toolbox enfocada a la computació paral·lela:

Taula 1 Preus actuals en MathWorks®

Licence Options	Product	Price
Standard	MATLAB + Parallel Computing Toolbox	**EUR 3.000,00**
Education	MATLAB + Parallel Computing Toolbox	**EUR 700,00**
Student	MATLAB and Simulink Student Suite + Parallel Computing Toolbox	**EUR 76,00**
Home	MATLAB + Parallel Computing Toolbox	**EUR 154,00**

Vistes totes aquestes limitacions, aquesta proposta queda descartada per la complexitat i pels grans problemes que ens dona l'ús de programació paral·lela en aquest llenguatge de programació.

4.3.2 Proposta 2

Al igual que la proposta 1 tenim les mateixes limitacions incloent el tema del hardware. Un sistema distribuït requereix d'una sèrie de clústers i una infraestructura de la que no disposem actualment. [9]

Com no disposem de la infraestructura per a fer una computació distribuïda, s'ha estudiat la possibilitat d'utilitzar els serveis proporcionats per MathWorks i Amazon Web Services.

Taula 2 Comparativa dels diferents sistemes distribuïts disponibles per a Matlab

	Parallel Computing Toolbox	Matlab Parallel Cloud		Matlab Distributed Computing Sever for Amazon EC2	Matlab Distributed Computing Sever- Private Cloud.	
Maximum Workers	No limitation (un worker per processador)	16		256	No limitation** (Bajo demanda)	
Hardware Resources	Desktop computer	MathWorks Cloud		Amazon EC2 instances (billed by Amazon Web Services)	Software installation followed by scheduler configuration.	
		Standar	Educational		Standar	Educational
Cost (per hour)		USD $6.08	USD $4.32	USD $4.90 (8cores)	USD $0.18 per worker	USD $0.07 per worker

Com bé està explicat en la taula 2, no podem optar a la part de computació paral·lela mitjançant el "Cloud", ja que els recursos són prou limitats i el preu és molt elevat. Per a poder fer una simulació pràctica i millorar els simuladors que hi ha al mercat, necessitaríem 1 "worker" per robot.

4.3.3 Proposta Final.

La proposta final serà desenvolupar el programa mitjançant el paradigma de programació orientada a objectes.

Tan sols necessitem una llicencia estàndard de Matlab 2015b i un equip capaç de executar el IDE[14]. [9]

Taula 3 Requisits mínims Matlab 2015b

64-bits Matlab				
Sistema Operatiu	Processador	Espai en disc	Memòria Ram	Gràfics
Windows 10, 8.1, 8, 7(sp1), Vista(sp2), XP(Sp3)	Tots els processadors basats en Intel o AMD x86-64	2GB per a Matlab, 4-6GB per instal·lació típica.	2 GB mínim. 4GB recomanada.	Targeta gràfica compatible en OpenGL 3.3 en 1GB GPU de memòria.
Windows server 2012, 2008,2003				
Ubuntu 14.04 LTS, 16.04 LTS, and 16.10 Red Hat Enterprise Linux 6 and 7		2,2 GB per a Matlab, 4-6GB per instal·lació típica.		
macOS Sierra (10.12) macOS El Capitan (10.11) macOS Yosemite (10.10)	Intelx86_64 processador	2,5 GB per a Matlab, 4-6GB per instal·lació típica.		

Vista totes les propostes i els recursos necessaris, la proposta final serà la implementació del software mitjançant Matlab, en el paradigma de programació orientada a objectes. Ja que el desenvolupament serà mes ràpid i natural.

[14] "Integrated Devolopement Envorionment", Entorn integrat de desenvolupament

5 IMPLEMENTACIÓ

Implementació pràctica en Matlab

5.1 ENTORN DE DESENVOLUPAMENT.

Per a la implementació del simulador utilitzarem Matlab en el paradigma de programació orientada a objectes. En els següents punts explicarem que és Matlab i com funciona en aquest paradigma.

5.1.1 Matlab

És una plataforma per a resoldre problemes d'enginyeria i científics, el llenguatge Matlab, basat en matrius i representació de dades.

Llenguatge d'alt nivell, interpretat, i pot executar-se en un entorn interactiu a traves de scripts(arxius ".m"). Aquest llenguatge permet operacions de vectors i matrius, funcions, càlcul lambda, i programació orientada a objectes, és multi-plataforma i mult-paradigma.

5.1.1.1 Instal·lació de Matlab.

Cal instal·lar-se la versió de R2015b, en la qual llançarem el programa desenvolupat.

1. Crear un conter en MathWorks.com amb el e-mail de la universitat.
2. Associar la llicència per a estudiants de la universitat
3. Per a utilitzar Matlab en Windows/Linux/osX...
    a. Descarregar el instal·lador
    b. Utilitzar el correu i la llicència.
    c. Seleccionar els productes a instal·lar.
    d. Activar el Software.

Després de fer els anteriors passos ja podrem començar a treballar sobre la IDE de Matlab.

5.1.1.2 IDE

En la figura 14 es mostra la pantalla principal, on es poden observar els quatre components més significatius. [11] [12]

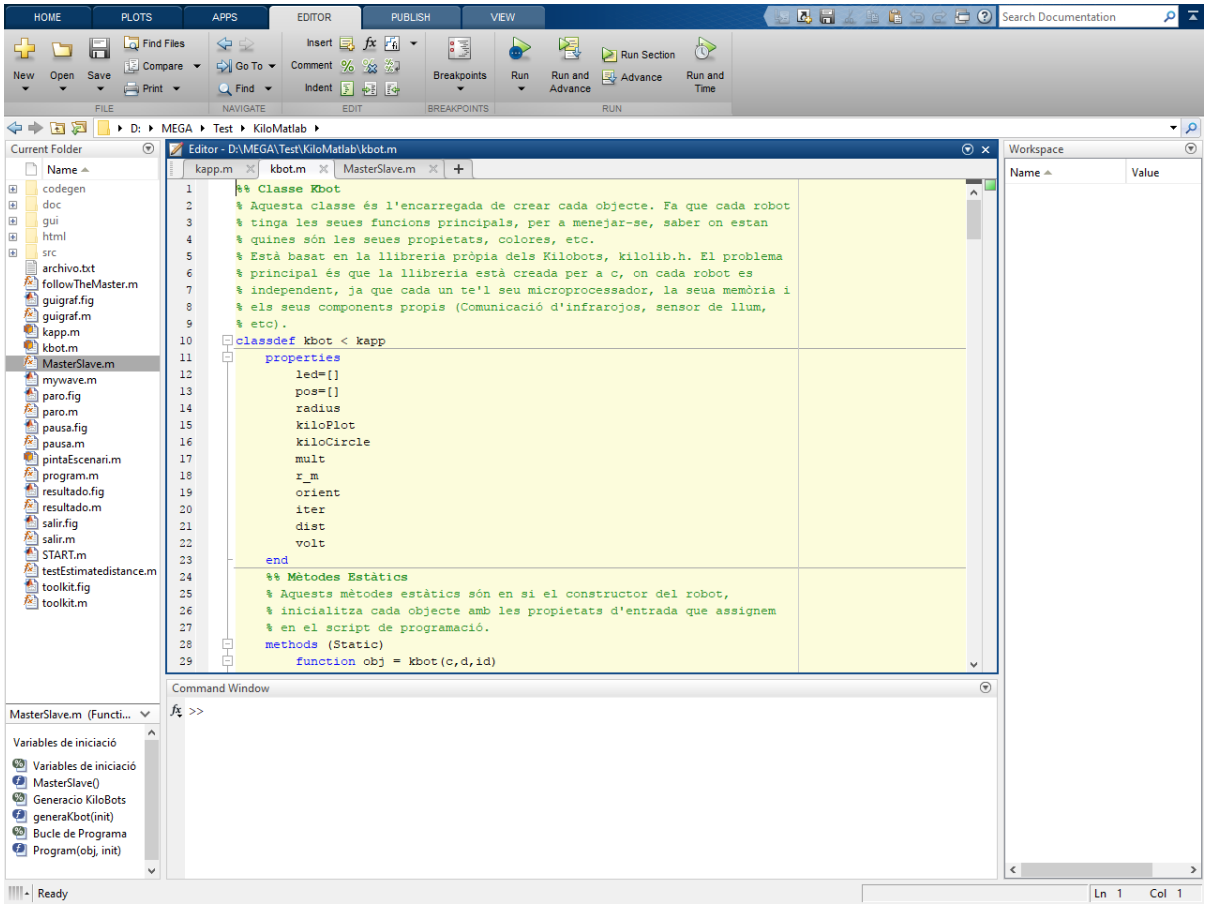


Fig. 14 Pantalla principal de Matlab

- Current Folder : Carpeta de treball actualment, es pot navegar sobre el sistema de fitxers.
- Workspace: On es mostren les variables creades per el usuari.
- Command Windows: Finestra de comandos interactiva en la qual deuen introduir-se les instruccions de Matlab on el prompt >> indica que esta llest per a rebre instruccions.
- Editor: Potent editor de fitxers M[15], en ell podem crear tots els scripts de programació, executar-los i depurar-los de forma interactiva.

[15] Fitxers de text ASCII, que contenen conjunts de comandaments o definicions de funcions.

5.1.1.3 Poo[16] en Matlab

Aquesta capacitat de programació orientada a objectes del llenguatge Matlab permet fer càlculs complexes molt més ràpid que altres llenguatges de programació. Matlab té les tècniques de programació per classes i la sobrecarrega de operadors, accés controlat a les propietats i mètodes, polimorfisme, herència, etc. [10]

5.1.1.4 Classes

Matlab organitza les definicions de classe en blocs modulars, delimitades per "keywords". Tots els mòduls tenen la "keyword" *end* al final de cada estament:

- `classdef...end` - Definició de nom de classe.
- `properties...end` - Declaració de noms de les propietats, propietats dels atributs i valors per defecte.
- `methods...end` - Declaració dels mètodes de la classe, els atributs i el codi de les funcions.
- `event...end` - Declaració de esdeveniments amb noms i atributs.
- `enumeration...end` - Declaració de classes enumerades.

Les propietats, mètodes, esdeveniments i enumeracions tenen que estar definides dins del bloc de `classdef`.

5.1.1.5 GUIDE

Ferramenta de Matlab per a la creació interactiva de interfície d'usuari (GUI). Per a cridar aquesta ferramenta es crida mitjançant el comando GUIDE o sobre la icona del entorn de Matlab. Aquest editor permet construir interfície arrastrant i soltant components en l'àrea de disseny de la GUI. Cada element crea una funció inicial anomenada callback que s'ha de cridar cada vegada que s'interactua amb ella. Després de la creació de la interfície gràfica, es creen els arxius de control que utilitza Matlab per a cridar a cada callback o funció. Aquesta ferramenta utilitza fitxers ".fig" per al disseny de la GUI i els ".m" per al codi que controla la GUI.

[16] Programació orienta a objectes

5.2 IMPLEMENTACIÓ PRÀCTICA.

En aquesta implementació mitjançant Matlab el que s'intenta es simular el comportament dels robots. En la figura 15 es mostra un esquema de creació d'objectes, on es veu que tenim el "handle" de Matlab i sobre ell crearem una classe pare anomenada Kapp, on controlarem els objectes kbots, i tindrem la capacitat de cridar a les diferents funcions implementades en els objectes al igual que accedir a les dades dels mateixos per així poder tractar-los i mostrar-los en pantalla de una forma gràfica i fàcil.

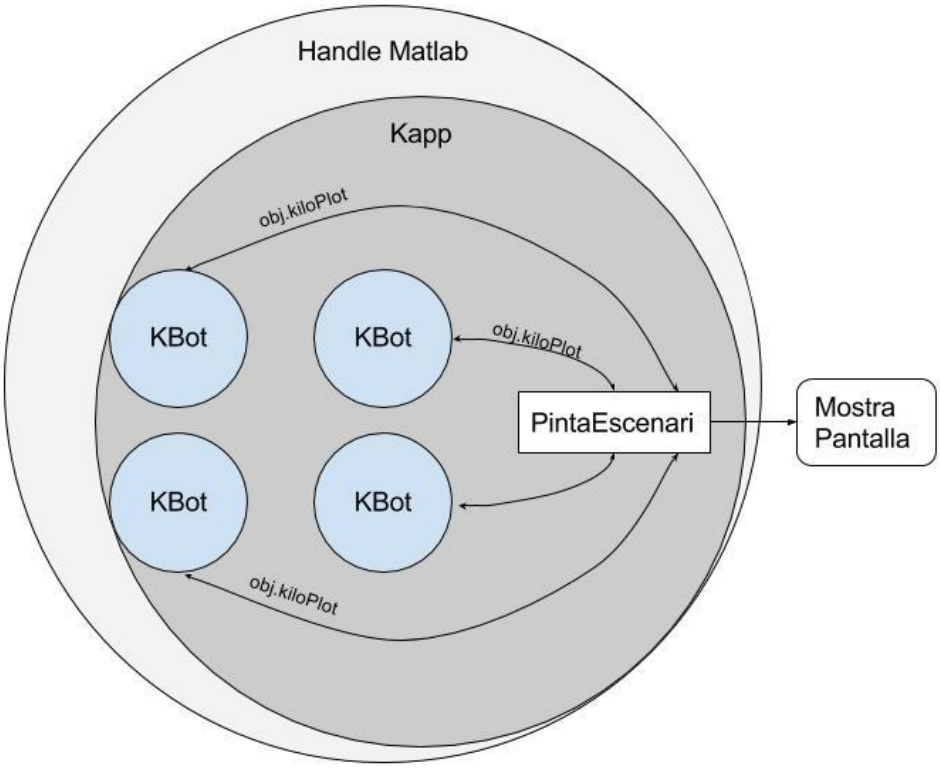


Fig. 15 Esquema de la creació de les classes amb els seus objectes.

5.2.1 Classe Kapp.

Es crea una classe pare anomenada kapp, on es construeixen les funcions utilitzades per el script. Sense aquesta funció l'objecte no pot crear-se, ni tindre interaccions en ells. Aquesta classe crear les propietats per a pintar l'Escenari, assigna la "id" als Kilobots i comprova l'estat d'aquests i posa està informació en comú. També té els mètodes utilitzats pels Kilobots, per poder accedir a ells des de qualsevol dels scripts. En el nostre cas des de la interfície gràfica i el script que dissenyem des de la mateixa. Es podria dir que aquesta classe és el model del qual cada Kilobot es capaç d'agafar la informació comú per a la seva creació.

5.2.2 pintaEscenari

Classe filla que pertany a kapp, és l'encarregada de pintar l'escenari on es produeix la simulació i es mostra el moviment de cada robot. Els paràmetres d'entrada és la base i la altura, i dibuixa el rectangle on es realitzarà la simulació.

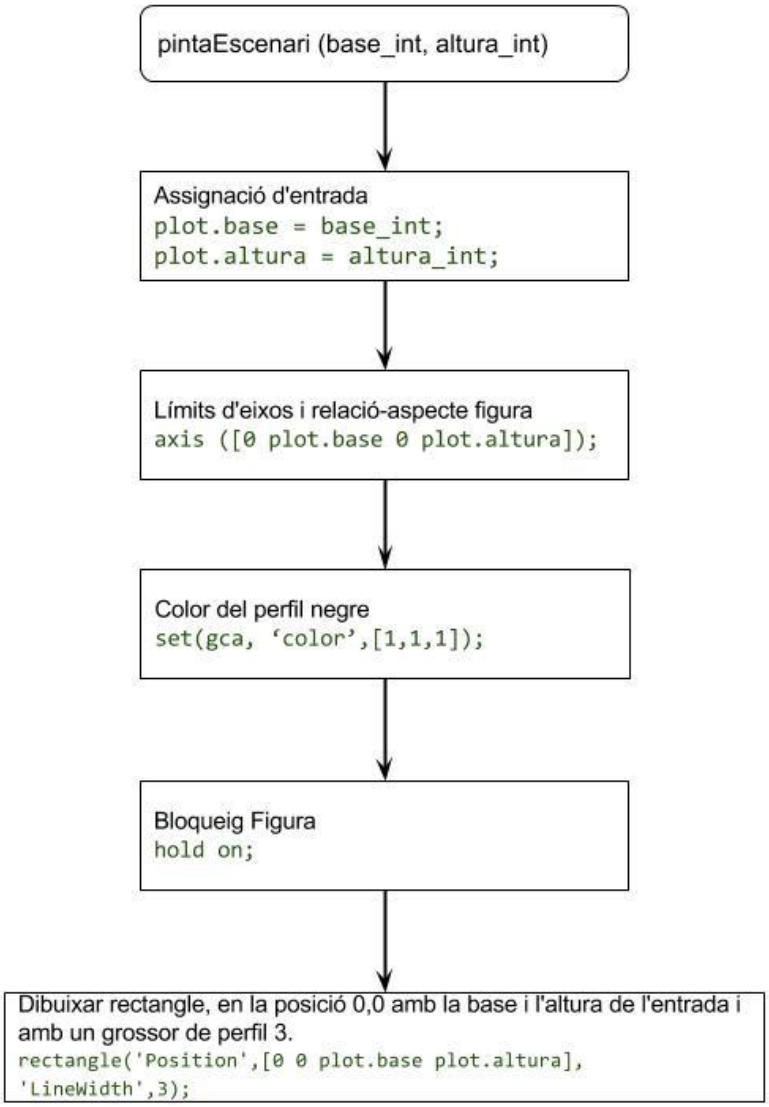


Fig. 16 Funcionament de la classe PintaEscenari

5.2.3 Kbot.

Aquesta classe és l'encarregada de crear cada objecte. Fa que cada robot tinga les seues funcions principals, per a menejar-se, saber on estan quines són les seues propietats, colors, etc. Està basat en la llibreria pròpia dels Kilobots, kilolib.h. El problema principal és que la llibreria està creada per a c, on cada robot es independent, ja que cada un té el seu microprocessador, la seua memòria i els seus components propis (Comunicació d'infrarojos, sensor de llum, etc). En la classe kbot, intentarem fer una aproximació lo màxim possible a la llibreria utilitzada en els Kilobots.

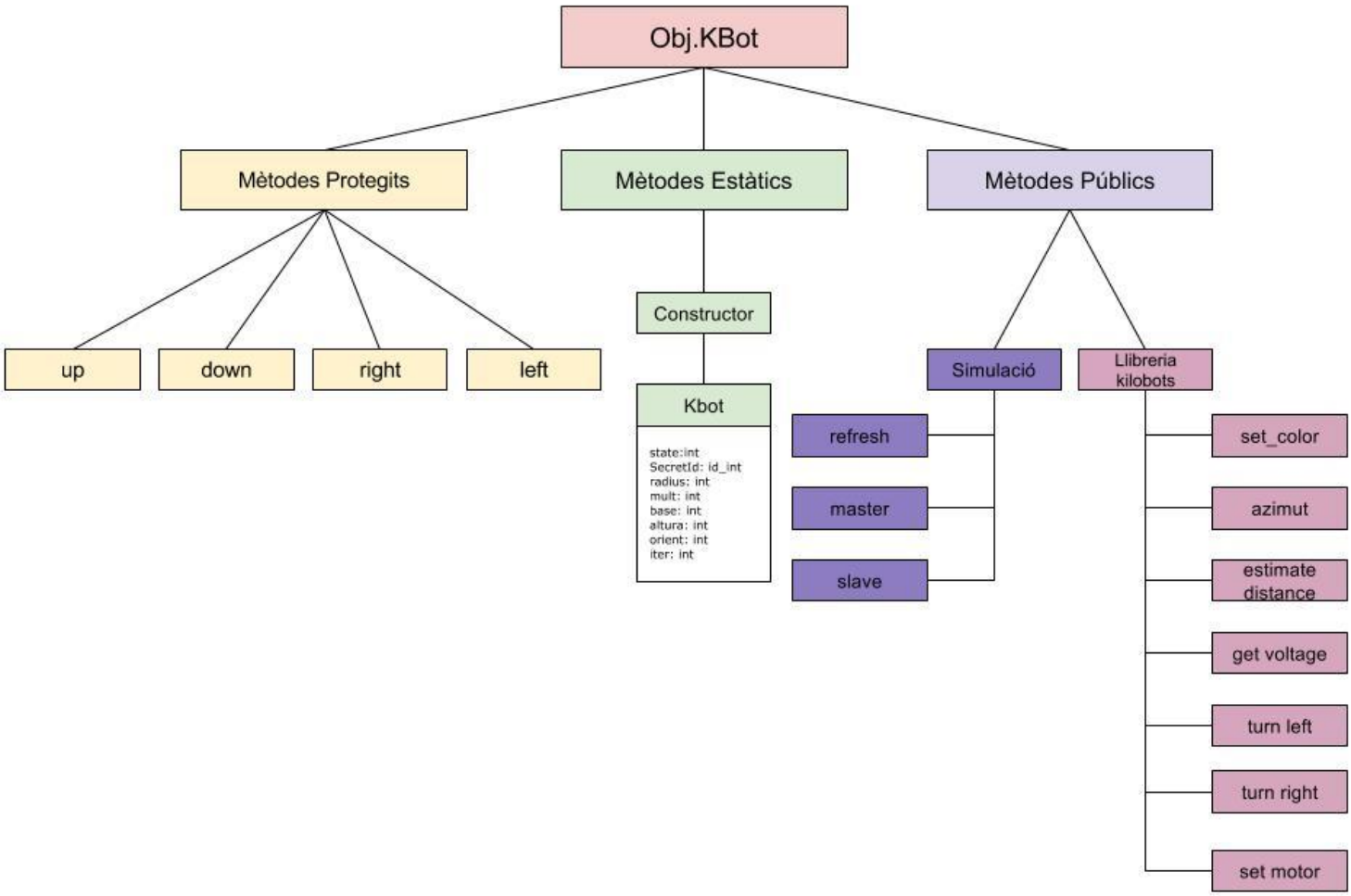


Fig. 17 Jerarquia de la classe Kbot. On obj.Kbot es la creació del objecte d'on pengen tots els mètodes i funcions.

5.2.3.1 Mètodes Estàtics

Com en tot paradigma de programació orientada a objectes es necessita un mètode estàtic com un constructor.

Kbot: Objecte principal, és l'encarregat d'inicialitzar totes les variables, i de crear l'objecte del qual aniran lligades totes les estructures de dades.

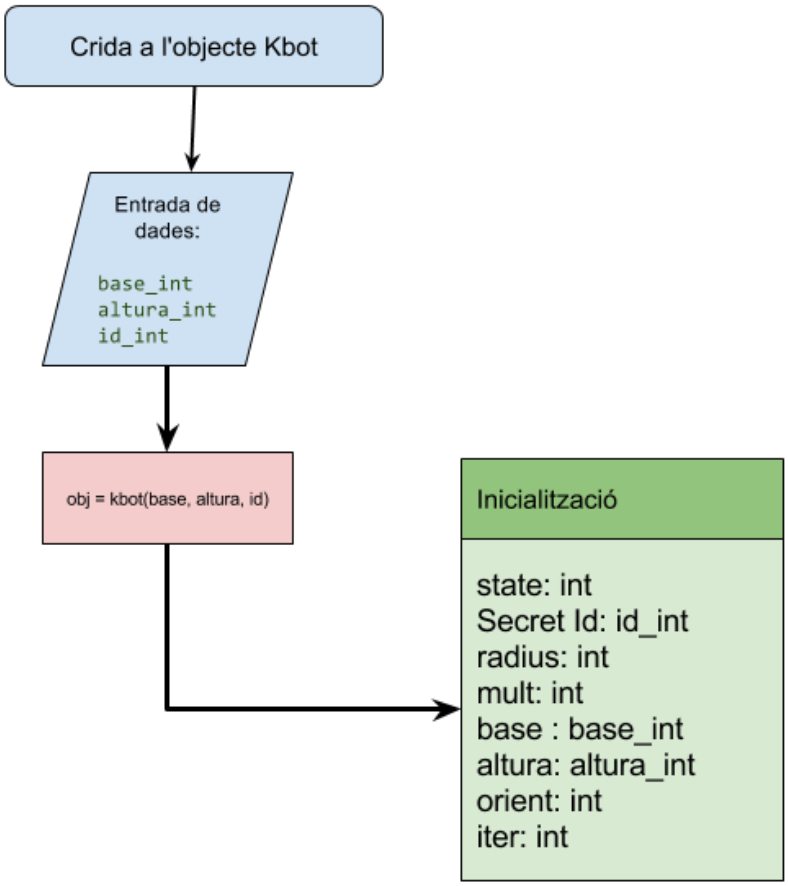


Fig. 18 Inicialització objecte

5.2.3.2 Mètodes Protegits

Aquests mètodes estan protegits ja que des del objecte principal no pugem accedir a ells, al ser propis de cada objecte i per tindre un control sobre ells.

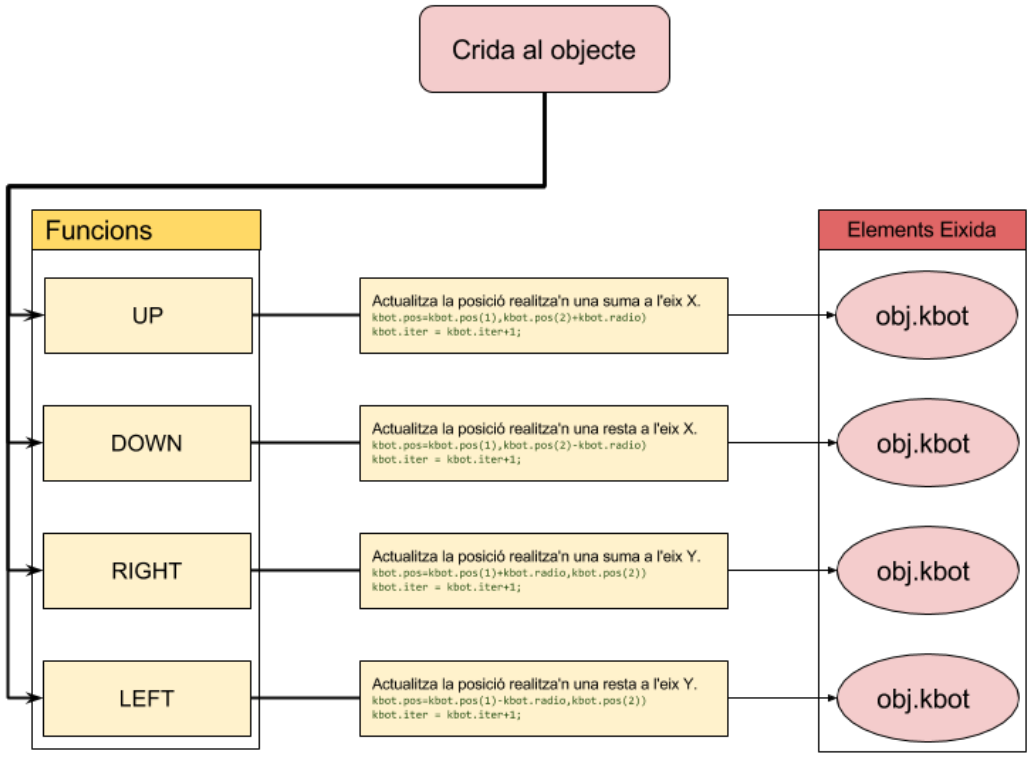


Fig. 19 Comportament de les funcions de moviment. Els elements de eixida es el propi objecte en la seva posició modificada.

Com bé posa en el nom de cada funció, cada vegada que es crida a aquest tipus de funció el robot (objecte) és mourà en l'escenari a la direcció indicada. També hi ha una variable `iter`, que és l'encarregada de contar-nos quantes iteracions fa cada robot, en cada moment, per a fer que el simulador ens conte molts moviments fa, i pugem calcular temps, treball, etc.

- up : Desplaçament cap amunt. Realitzant l'operació de sumar el radi del Kilobot a la posició "X" del propi objecte.
- down: Desplaçament cap avall. Realitzant l'operació de restar el radi del Kilobot a la posició "X" del propi objecte.
- `left`: Desplaçament cap a la esquerra. Realitzant l'operació de restar el radi del Kilobot a la posició "Y" del propi objecte
- `right`: Desplaçament cap dreta. Realitzant l'operació de sumar el radi del Kilobot a la posició "Y" del propi objecte.

5.2.3.3 Funcions de simulació

Aquestes funcions són les que es va a cridar des del script principal d'execució, les quals tenim les següents:

- `refresh`: Refresca la posició nova. Borra la posició anterior i posa la nova.

Funciona de prova: En aquestes fusions es proven els diferents comportament dels Kilobots.

- `slave` : Aquesta funció és la que assigna al abjecte que es un objecte esclau, del qual mira on està l'objecte mestre i va cap a ell (Aquesta funció me la he inventat per a fer una petita simulació, per a comprovar que els objectes van on les funcions els indica).
- màster : Posa el Kilobot en estat 0, va en conjunt amb la funció slave.

5.2.3.4 Funcions implementades en Kilobot.h

Aquestes funcions són les més paregudes a les funcions reals dels Kilobots. Es criden al script principal per a poder realitzar la simulació desitjada.

set_color :

Pinta a l'escenari un punt de forma redona amb la funció pròpia de Matlab plot[17], del color assignat en variable LED i d'una grandària de 5. És comú per a tots, i l'únic valor que podem modificar son els LED, que és un vector de 3 elements, per al qual indiquem el color.

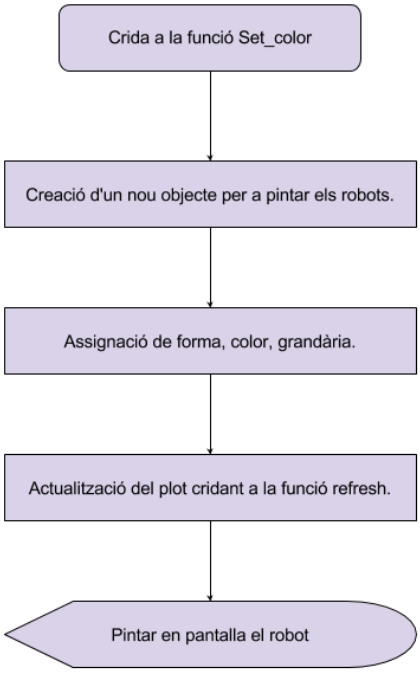


Fig. 20 Esquema de funcionament de la funció Set_color.

[17] Plot(x,y) Crea un diagrama de puts 2-D amb els valors X i Y. Es un objecte propi de Matlab

Azimut:

És una funció, és l'encarregada de calcular l'orientació del robot, si va al nord, sud, est o oest. Com bé mostra en la figura 21, el primer que fa aquesta funció és llegir l'orientació en la que es genera el Kilobot (la podem posar manualment cridant al objecte i assignar-li un valor o es pot crear automàticament en un aleatori) i a partir de l'orientació comprova si l'orientació està dins dels rangs assignats, depenent de l'orientació el robot actua anant cap amunt, avall, dreta o esquerra.

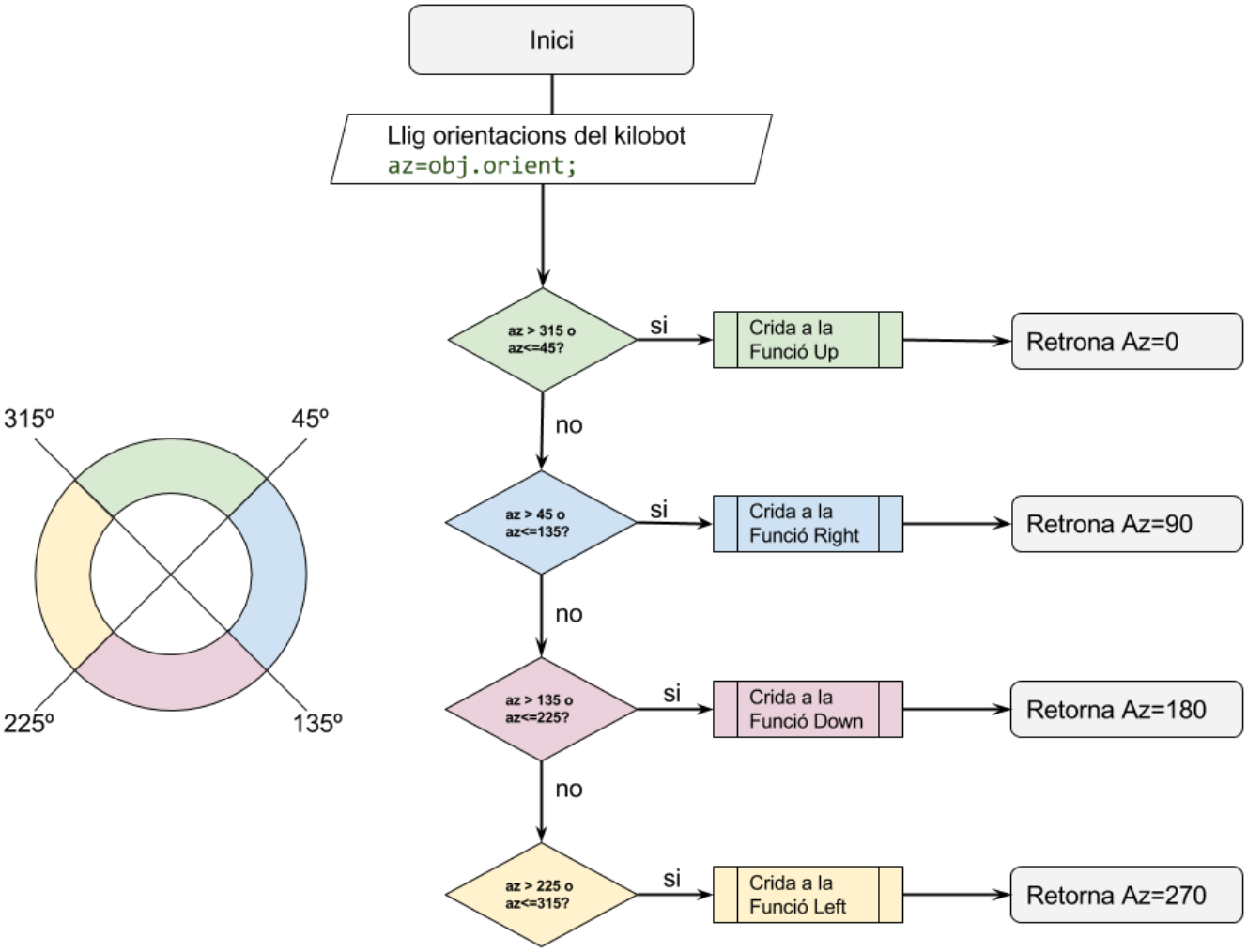


Fig. 21 Funció Azimut

set_motor :

Aquesta funció és l'encarregada de simular els motors, ja que en la llibreria dels Kilobots, hi ha una funció set motor que és l'encarregada d'activar els motors, en un valor de 0 a 255. La funció té tres entrades, l'objecte a menejar, el valor del motor dret i el valor del motor esquerre. Després dins de la funció, intenta simular el moviment circular de cada robot cridant a una funció que es diu turn_left (en el cas de l'esquerra) i turn_right (en cas de la dreta. Simplement van a la dreta i a l'esquerra, faltaria fer que siguen capaços de simular tota la circumferència real, inclòs el soroll que genera el Kilobot, ja que per ells mateixa no fan ni una circumferència perfecta ni poden anar en linneà recta (en el cas que els dos motors tinguen el mateix valor sí).

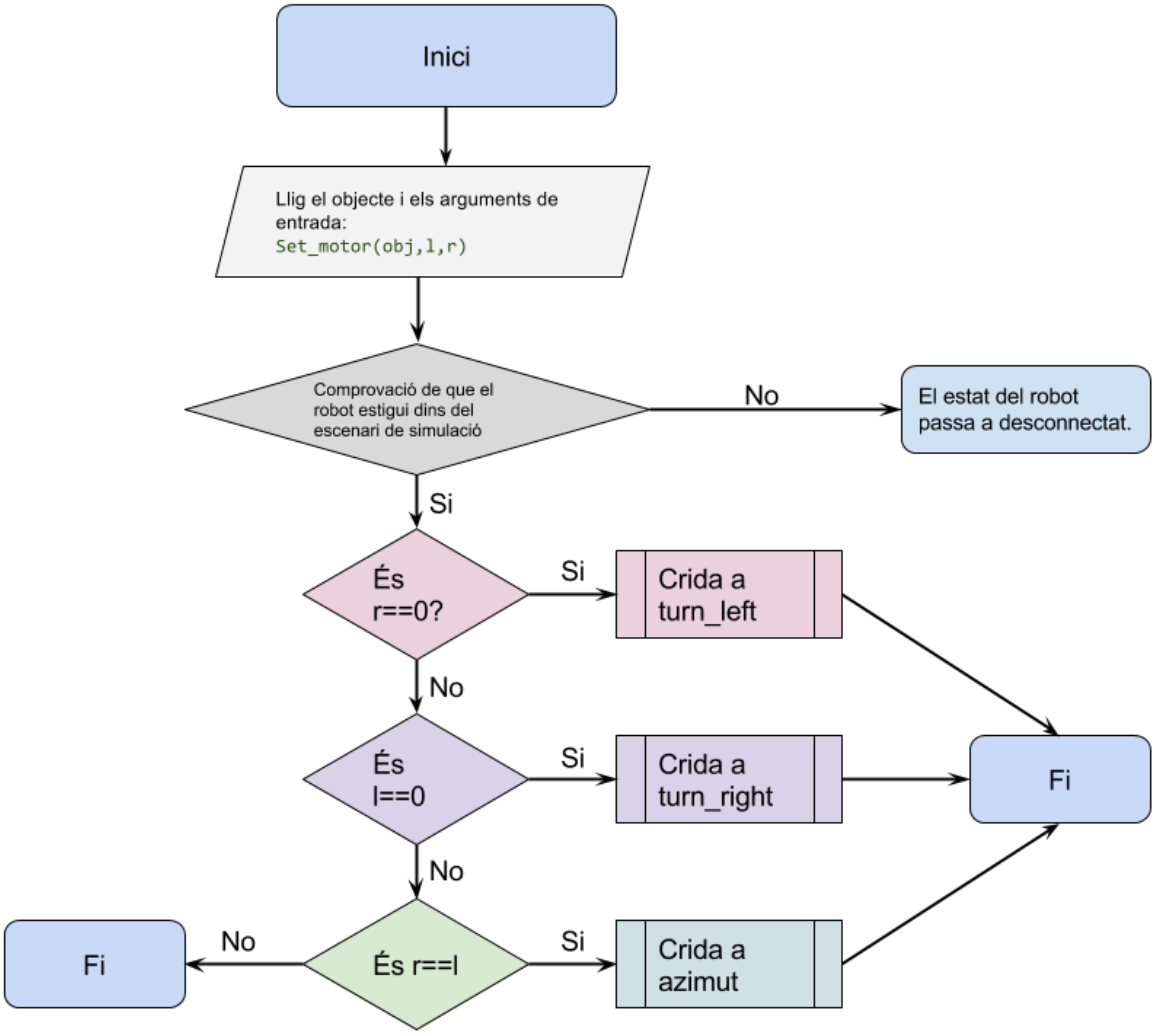


Fig. 22 Funció Set_Motor

Estimate_distance:

Aquesta funció estima la distancia entre els objectes, d'entrada té l'objecte propi i al que es vol estimar, i es fa una resta per a saber la distancia que hi ha entre els dos robots. Simula el sensor IR que du el Kilobot, utilitza la comunicació entre els objectes, passant les variables de posició i realitzant una resta.

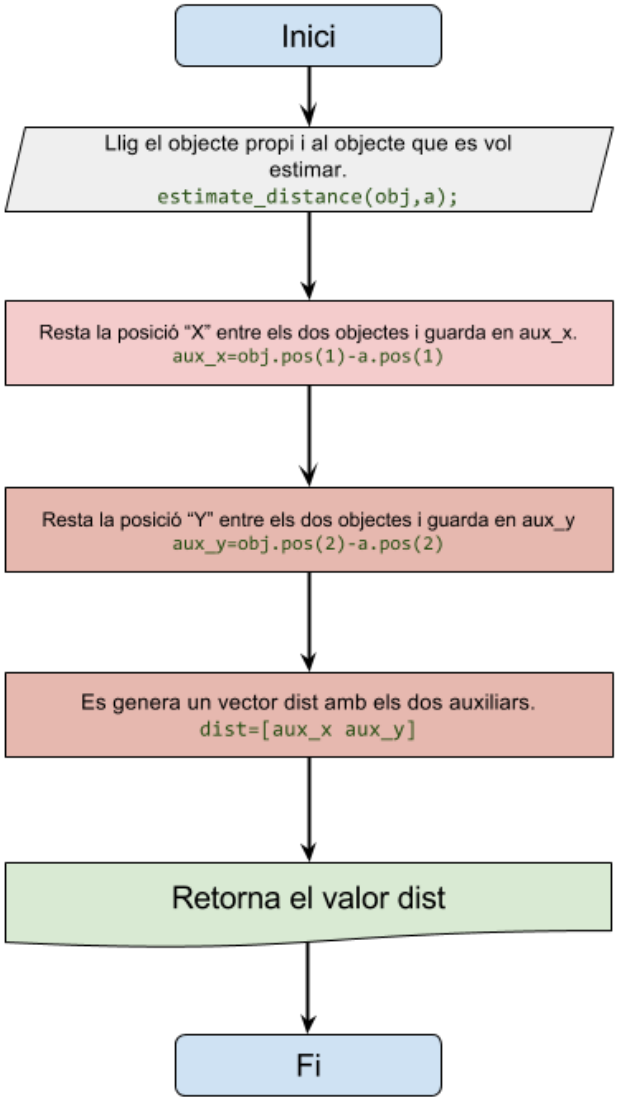


Fig. 23 Esquema de Estimate_distance

get_voltage:

Simula el voltatge de cada Kilobot real. Entra el robot que vols saber quin voltatge té i retorna el valor. Si el voltatge és 0 l'estat del robot passa a ser desconnectat.

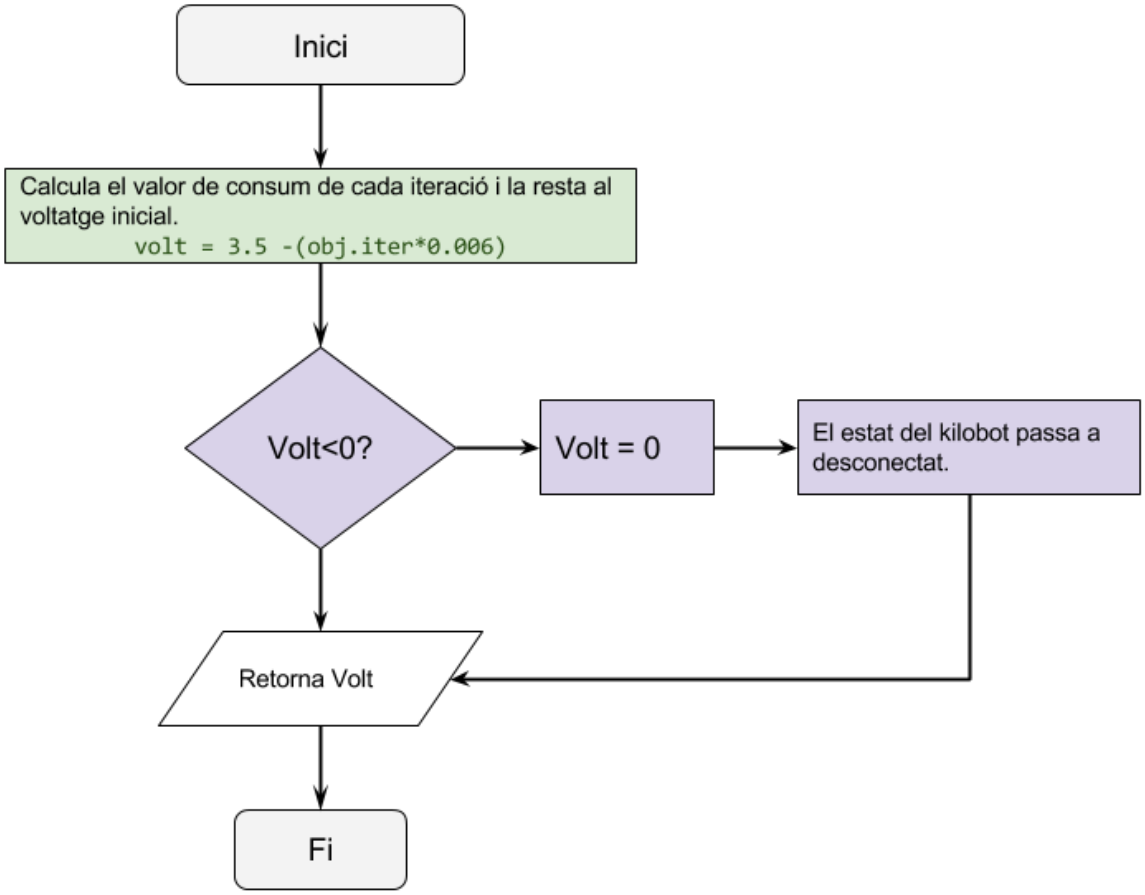


Fig. 24 Funció Get_Voltage

En la Fig. 24 inicia la cridada de la funció i calcula el valor del voltatge mitjançant una resta. Suposem que cada iteració té un consum de 0.006V, per a calcular el valor del voltatge cada vegada que es crida a aquesta funció, es multiplica la quantitat de iteracions per 0.006 i es resta al total del voltatge. Comprovem que el voltatge siga sempre major que 0, en el cas de que siga menor, el robot passarà a estar desactivat i a state 0.

TURN_LEFT / TURN_RIGHT:

Aquesta funció està creada per a simular el moviment del robot, sempre que s'activa un motor del robot, fa que el robot tinga un moviment circular depenent si és el motor dret o l'esquerra. En un escenari real el robot té un moviment circular controlat per un PD. En la simulació es pretén tindre el mateix moviment però es més quadrat que circular, ja que la simulació de creació d'un cercle es computacionalment alta.

Com bé està explicat en la fig 25 i fig 26, la funció inicia i llig l'entrada dels arguments, que són el propi objecte i la entrada del valor del motor. Es comprova que mentre el valor d'entrada siga menor que la variable auxiliar "i", crida a les funcions Up, Right, Down, Left, en el cas de fer el moviment en direcció de les agulles del rellotge i Up, Left, Down, Right en el cas de contra direcció. Això ens proporciona un moviment del robot similar al real.

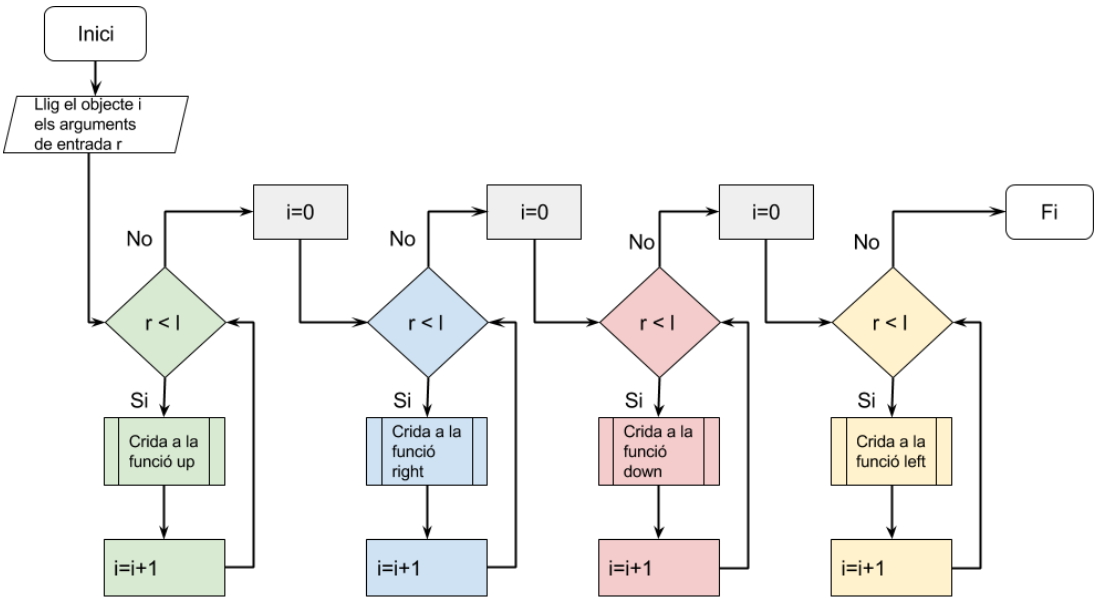


Fig. 25 Funció Turn_Right

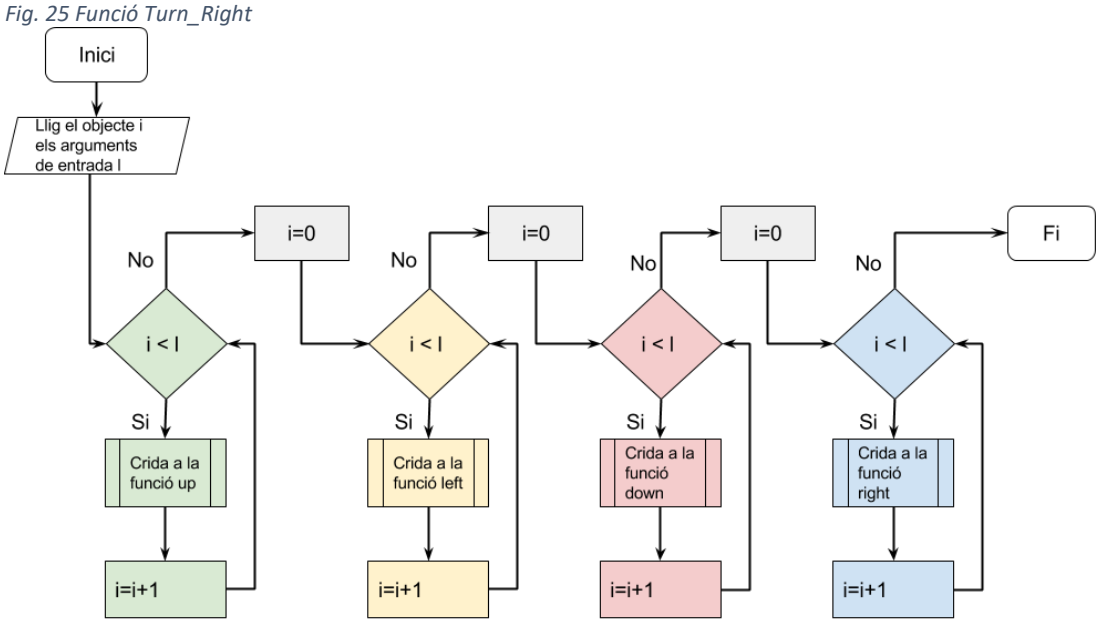


Fig. 26 Funció Turn_Left

5.3 PROVES

Per al funcionament d'aquest simulador s'ha de crear un script de programació dels Kilobots, utilitzant una plantilla.

En aquest script tenim 3 funcions. `Function[result]=Nom_del_programa,`

`function [g]=generaKbot(init)` i `function [p]=Program(obj,init)`.

`function [result]=Nom_del_programa:` En el primer lloc s'ha d'inicialitzar les funcions, i les variables d'inicialització, on el nom de la funció és el mateix nom que el script de programació. En aquesta part s'inicialitzen les constants base i altura del entorn de simulació, el numero de robots que volem a l'entorn i es crida a les funcions de execució.

`function [g]=generaKbot(init)`: A continuació es generen els Kilobots, la generació pot ser manual o automàtica. La funció del script principal generaKbot, retorna els Kilobots creats. En aquesta funció també podem assignar el color dels LED dels Kilobots cridant després de la creació a la funció `set_color`.

Manual: Es crida al objecte kbot mitjançant la generació d'objectes de Matlab `nom_robot=kbot(posició_x, posició_y, id_robot)`.

Automàtica: mitjançant bucles es poden generar automàticament els robots.

```
for x=1:num
    id=x;
    pos_x = randi(base-10);
    pos_y = randi(altura-10);
    if x==1
        g.kbots(x)= kbot(pos_x,pos_y,id);
    else
        g.kbots(x)= kbot(pos_x,pos_y,id);
        for j=1:x
            if x==j
            else
            if g.kbots(x).pos == g.kbots(j).pos
                x=x-1;
            else
                %g.kbots(x).set_color(led)
            end
            end
        end
    end
    g.kbots(x).set_color(led)
end
```

Fig. 27 Exemple de generació automàtica de Kilobots en posicions aleatòries.

`function [p]=Program(obj,init)`: En aquesta funció és on es programa el algoritme que volem que realitze cada robot dins d'un bucle while (al igual que el loop que utilitzen els Kilobots). D'entrada tenen el Kilobots com a objectes de la funció anterior i el número de Kilobots en la simulació i retorna el resultat de la simulació amb totes les dades dels objectes (robots) de la simulació per a ser després tractada com es desitge.

```
function[p]=Program(obj,numk)
tic;
old_orient=obj.kbots(2).orient;
obj.kbots(2).orient=270;
a=1;
i=0;
num=numk;
obj.kbots(1).state=0;
while ~ false
    estados=[obj.kbots.state];
    dist= obj.kbots(2).estimate_distance(obj.kbots(1));

    if max(dist)<70
        if dist(2)==0
            obj.kbots(2).orient=180;
            obj.kbots(2).set_motor(max(dist),max(dist));
                obj.kbots(2).refresh;
        else
    obj.kbots(2).set_motor((max(dist)*2),0);
        end
    else
        obj.kbots(2).orient=270;
        obj.kbots(2).set_motor(70,70);
    end

    if (max(estados)==0)
        break;
    end
    volt=obj.kbots(2).get_voltage;
end
p=obj.kbots;
end
```

Fig. 28 Exemple de la funció "program", executant una orbita de robots

5.4 EXEMPLES DE SIMULACIÓ:

5.4.1.1 Master Slave:

Aquest exemple es simple, es genera un robot que es el mestre, on té un "id=1" i està en estat descontentat. Després es generen els altres robots aleatoris. I el que fa el simulador És que tots els robots que tinguen un "id" superior a 1 es dirigeixen al màster, sent esclaus.

Taula 4 Simulació Master Slave

Funció	Eixida en pantalla
Generació de Kilobots Es automàtica i aleatòria. *Fig. 29 Codi Generació Kilobots (MasterSlave)*	*Fig. 30 Generació robots (MasterSlave)*
Assignació màster *Fig. 31 Codi Assignació Master (MasterSlave)* Ejecció del bucle de programa Els robots marcats en negre van en direcció al roig. Conforme van aplegant al Master, van desactivant-se.	*Fig. 32 Assignació Master (MasterSlave)*
Resultat final Al finalitzar la simulació, els robots estan al voltant del robot Master. Desactivats La simulació retorna totes les propietats dels objectes.	*Fig. 33 Resultat final (MasterSlave)*

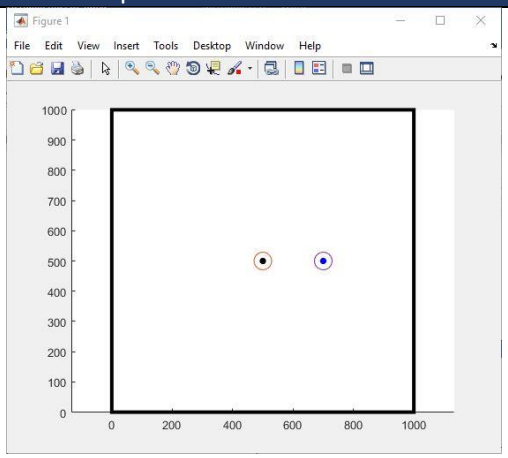
```
for x=1:num
    id=x;
    pos_x = randi(base-10);
    pos_y = randi(altura-10);
    if x==1
        g.kbots(x)= kbot(base/2,altura/2,id);
    else
        g.kbots(x)= kbot(pos_x,pos_y,id);
        for j=1:x
            if x~=j
                if g.kbots(x).pos == g.kbots(j).pos
                    x=x-1;
                end
            end
        end
    end
    g.kbots(x).set_color(led)
end
```

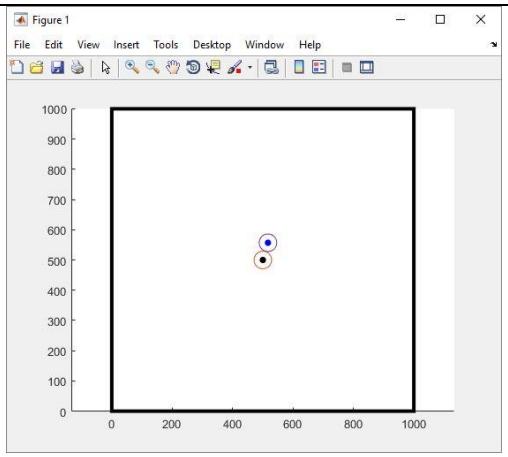
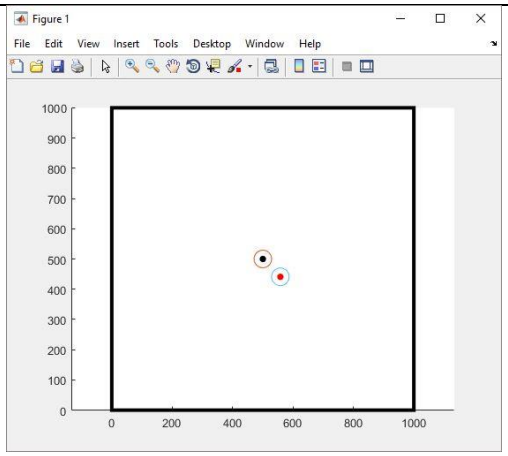
```
obj.kbots(1).master;
obj.kbots(1).state=0;
obj.kbots(1).set_color([1 0 0]);
```

5.4.1.2 testEstimatedistance:

En aquesta simulació, es proven les funcions de `estimate_distance`, `set_motor` i `get_voltatge`. Principal ment el que simula és l'orbita de dos robots, un esta immòbil enviant-li a l'altre en tot moment la informació de la distància que hi ha entre els dos. I l'altre rota al voltant d'ell fins que es quede sense energia.

Funció	Eixida en pantalla

Generació de Kilobots:
Es generen dos robots de forma manual.
`g.kbots(1)= kbot(500,500,1);`
`g.kbots(2)= kbot(700,500,2);`
`g.kbots(1).set_color(led);`
`g.kbots(2).set_color([0 0 1]);`
Fig. 34 Codi generació kilobots (testEstimatedicstance)
	
	Fig. 35 Generació robots (testEstimatedistance)

Algoritme de simulació: El robot que està orbitant comprova en tot moment la distància que té amb el robot quiet. I orbita depenent de la distancia calculada.	 *Fig. 36 Simulació Orbit (testEstimatedistance)*
Resultat final: Quan el robot orbitador es queda sense energia para i es posa en roig. La simulació retorna els dos objectes.	 *Fig. 37 Finalització simulació (testEstimatedistance)*

6 MANUAL D'USUARI

Per a llançar l'aplicació dins de l'entorn de Matlab, tan sols s'ha d'executar el script START.m.

6.1 INTERFÍCIE GRÀFICA:

Després de desenvolupar el codi referent a la creació dels objectes i les classes pertinents es crea una interfície gràfica mitjançant la ferramenta GUIDE que porta Matlab.

Com bé es pot vorer en la fig. 38, la interfície té diferents panells, on el principal i més gran és l'entorn de simulació, on es mostrarà la funció `pintaEscenari` i el moviment dels diferents Kilobots. També hi ha diferents panells on controlarem les diferents funcions de la interfície del simulador.

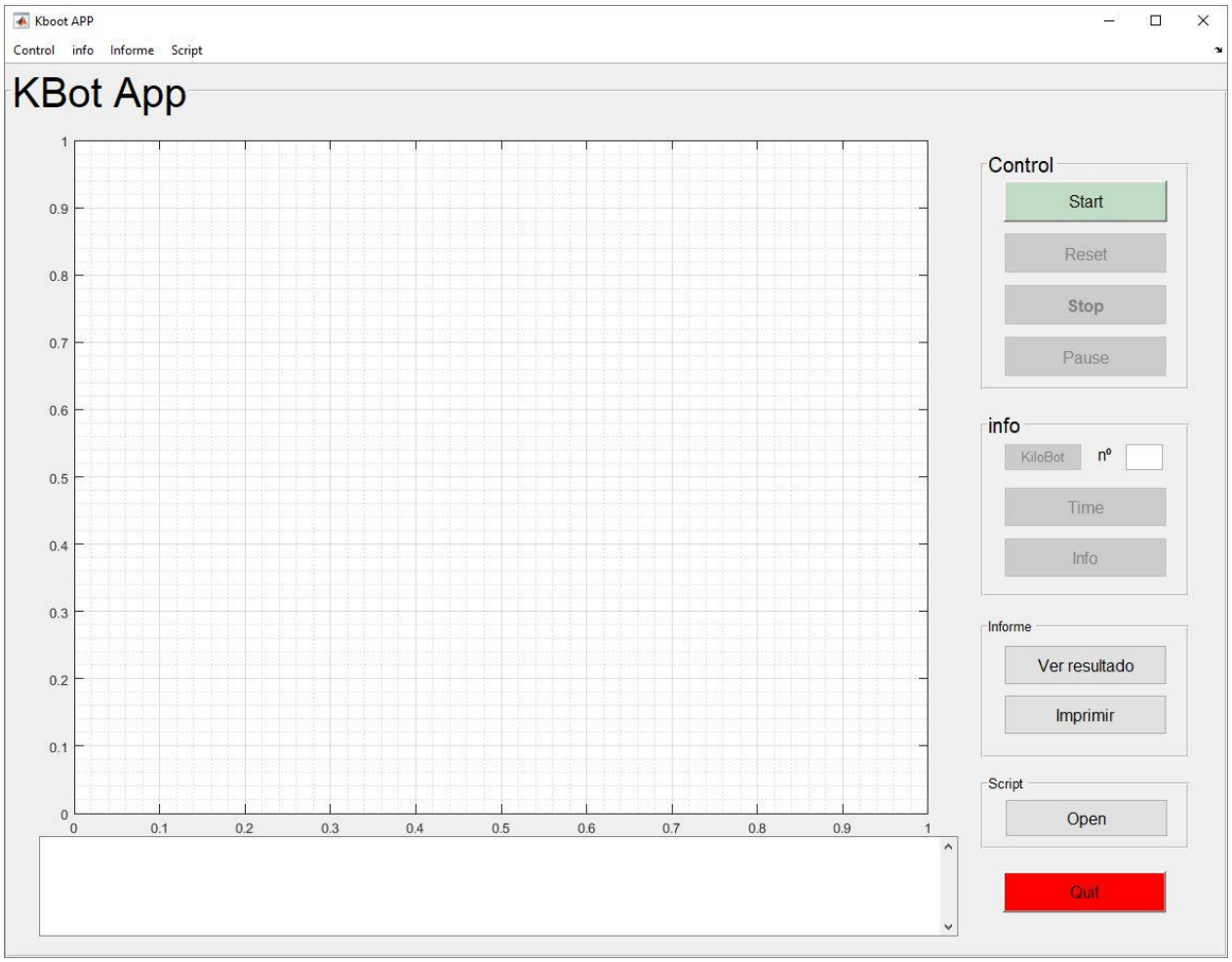


Fig. 38 Interfície principal

6.1.1.1 Panell de Control:

Fig. 39 Panell Control

- Start: L'execució quan polses Start, activa tots els botons de control. Comprova que no està executant-se res i posa la variable start a true. Mostra per consola que la simulació ha començat e inicia el comptador de temps, executa el script de simulació, i en acabar, para el rellotge. Activa els botons d'informació, ja que la informació no està disponible fins que no s'acabe d'executar el script, torna la variable start a false, mostra en la consola que la simulació ha sigut finalitzada, recarrega les dades, i desactiva els botons de control.

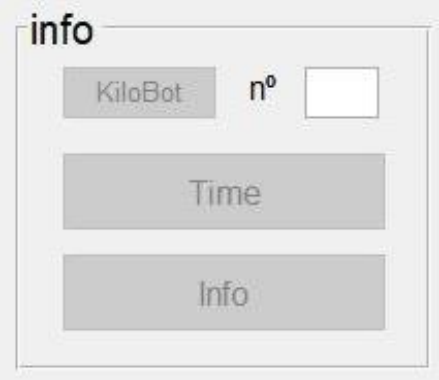
- Reset: L'execució quan es polsa el botó reset. Primer borra tota la informació que hi ha a axes1 (o siga a l'escenari de simulació). Després actualitza l'estructura de dades. I seguidament crida al Callback Start.

- Stop: L'execució quan polses Stop comprova que hi ha alguna cosa executant en el simulador, després crida a la funció paro, que pregunta si realment vol parar l'execució de la simulació, en el cas de ser true, neteja el escenari de simulació, actualitza la variable de simulació activa a false, actualitza l'estructura de dades i desactiva tots els botons de control. Mitjançant la consola de comandos, avisa que la simulació ha sigut avortada.

- Pause: L'execució quan polses Pause, pausa l'execució mostrant un pop-up de què la simulació està pausada.

- Quit: L'execució quan polses el botó Quit. Crida a la funció salir que li retorna un true o false, depenent si prems sí o no a la pregunta de vol tancar el programa, en el cas que sí, quit és true, aleshores és neteja tot el "handles" (la classe principal) i es tanca tot.

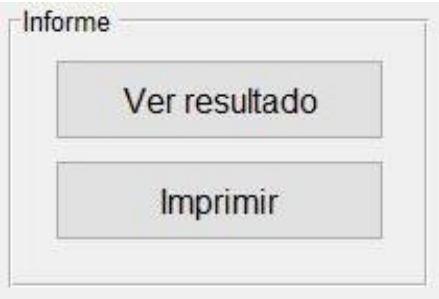
6.1.1.2 *Panell Info*



- Kilobot: L'execució quan polses Kilobot, mostra per la consola la informació del Kilobot seleccionat, mitjançant l'entrada d'un valor kbot_numero.
- Time: Retorna en la línia d'ordres el valor del temps real de la simulació.
- Info: Retorna el temps de màquina per la linneà d'ordres del simulador.

Fig. 40 *Panell Info*

6.1.1.3 *Panell Informe*



- Resultado: L'execució quan pulses resultado, crida a la rutina resultado. Per a mostrar un pop-up amb els resultats de la simulació.
- Imprime: L'execució quan polses imprime, el que fa és crear un fitxer Archivo.txt, i va escrivint línia a línia les variables predefinides anteriorment.

Fig. 41 Panell Informe

6.1.1.4 *Panell Script*

- Open: L'execució quan polses open crida a la rutina open, que es la que s'encarrega de obrir un pop up on et mostra els arxius per a poder seleccionar el script de simulació que es te que executar.
- Quit: Per a tancar l'aplicació

Fig. 42 Panell Script

7 CONCLUSIÓ

7.1 CONCLUSIONS PERSONALS

Per a realitzar aquest software he tingut en compte:

- Estudiar que són els Kilobots, com funcionen, quin es el seu hardware, etc.
- Estudiar el llenguatge de programació Matlab, estudiar el paradigma orientat a objectes.
- Realitzar un estudi sobre que és la computació paral·lela, com funciona, i el seu estat actual.
- Vorer les possibilitats de programació paral·lela orientada a objectes en Matlab.
- Estudiar com mostrar en pantalla la simulació, mitjançant exemples executats en Matlab.
- Estudiar els simuladors que hi han actualment de Kilobots.
- Fer proves de funcionament de la llibreria Kilobots en el V-REP
- Instal·lar els diferents simuladors per estudiar com funcionaven i veure les seues principals limitacions.
- Esquematitzar les funcions dels robots en diagrames de flux i escriure-les en Matlab.

Després de realitzar els punts anteriors, aquestes són les meues conclusions.

L'estudi dels robots Kilobot ha sigut una de les parts que més m'ha agradat, ja que soc un aficionat a la robòtica i als microcontroladors, mai pensava que arribaríem a produir aquesta tecnologia. Robots minúsculs que són capaços de realitzar una feina en comú a partir del sensors propis. Sense necessitat de càmeres externes, ni d'un controlador de posició. Una tecnologia amb molt de futur a l'àmbit del salvament, construcció i un llarg etc. Però per un altre costat he vist que encara es una tecnologia embrionària, queda molt que estudiar i practicar sobre aquest tema.

La programació en Matlab, no ha sigut fàcil, ja que és un llenguatge que hem vist poc a la universitat, en el meu cas l'he utilitzat en matemàtiques i fent alguna que altra pràctica de Simulink en la assignatura de Control per Computador. Però no a nivell de desenvolupament de software.

Matlab és molt fàcil de programar i de realitzar moltes funcions computacionals, però no és un llenguatge enfocat per al desenvolupament de software, i menys si s'utilitza per a mostrar gràfics en pantalla en temps real. La generació d'una figura en Matlab resulta molt tediosa i d'un alt cost computacional. A l'executar software sobre Matlab, te n'adones que aquest llenguatge no està enfocat per a fer software comercial, més bé per a ser una ferramenta de càlcul científic i de enginyeria.

El fet de programar orientat a objectes m'ha servit per a consolidar el que he après durant la carrera, no soc molt aficionat a la programació, però he vist en aquest projecte el perquè es el paradigma de programació més utilitzat actualment.

En quant a la computació paral·lela, no ha sigut molt gratificant, el fet de no poder provar moltes coses, ja que no disposàvem de la llicencia del toolbox fins a última hora. Actualment en Matlab s'utilitza per paral·lelitzar grans volums de dades, executar funcions

repetitives o càlculs de matrius generalment grans. Cosa que el nostre simulador no realitza. Vaig fer diferents proves, sense obtindre un resultat favorable, per això vaig decidir descartar-ho.

Els actuals simuladors que hi ha dels Kilobots es troben en la mateixa situació que nosaltres, no són capaços d'executar un gran número de robots a l'hora, i tots es queden limitats per el hardware d'avui en dia.

El desenvolupament en Matlab a sigut relativament difícil, ja que el llenguatge de Matlab limita molt a les funcions que ja té creades, al ser de tant alt nivell, deixa moltes coses que les trie l'intèrpret, donant problemes d'execució.

7.2 LÍNIES DE DESENVOLUPAMENT POSTERIOR

A continuació es mosta una llista de les possibles línies de desenvolupament:

- Estudiar el pas de missatges entre objectes en Matlab
- Realitzar la simulació en un entorn de simulació com puga ser Simulink.
- Desenvolupar una bona implementació que puga utilitzar els sistemes distribuïts i paral·lels, i aprofitar els actuals ordenadors de baix preu com la RaspBerry, BeagleBoard, etc.
- Utilitzar llibreries en "C", per a una programació mes real dels scripts de simulació.

8 BIBLIOGRAFIA

[1] M. Rubenstein and W. Shen. Automatic scalable size selection for the shape of a distributed robotic collective. In IROS, 2010. PDF

[2] M.Rubenstein, Chirian Ahler and Ridhika Nagpal. Kilobot: A Low Cost Scalable Robot System for Collective Behaviors.

[3] K-team. Kilobot Mauals & downloads, online

[4] V-REP User Manual http://www.coppeliarobotics.com/helpFiles/index.html

[5] F. Jansson, M. Hartley, M. Hinsch, T. Olsson, I. Slavkov, N. Carranza. Kilombo: a Kilobot simulador to enable effective research in swarm robòtics. PDF

[6] Carlos D. Acosta. Programacion científica en paralelo con interface matlab. Revista colombiana de matematiacas.

[7] Gaurav Sharma, Jos Martin. MATLAB®: A Language for Parallel Computing.

[8] Fernando J. Lage, Javier. J. Palmerio, Anibal Damian Coppo, Zulma Cataldi. PROCESAMIENTO PARALELO EN MATLAB. SU APLICACIÓN A MODELOS TEÓRICOS. WICC 2010 - XII Workshop de Investigadores en Ciencias de la Computación

[9] Matlab documentation: https://es.mathworks.com/help/

[10] MathWorks, "Programación orientada a objetos en Matlab", online

[11] Lucia Agud Albesa, Mª Leonor Pla Ferrando. Matalb para matemáticas en ingenierías. Ed. Universitat politècnica de València.

[12] Julio Benítez López, José Luis Hueso Pagoaga. Introducción a MATLAB. Dpt. Matemática aplicada Universitat Politècnica de València.

9 ANNEXES

En aquest apartat s'annexa el codi desenvolupat en Matlab per a aquest TFG. Per qüestions d'espai, el codi s'inclou en format digital amb tota la documentació entregada a la UPV. Si voleu el codi original també podeu sol·licitar-lo directament a l'autor.

9.1 CODI DE CLASSE PRINCIPAL I CLASSE PINTAESCENARI.

9.2 CODI DE CLASSE KBOT.

9.3 CODI DE GUI.

9.4 CODI GUI POP-UP.

9.5 SCRIPTS DE SIMULACIÓ