EPS
Escola Politècnica
Superior

Projecte/Treball Fi de Carrera

Estudi: Eng. Tècn. Informàtica de Sistemes. Pla 2001

Títol: Odometria i planificació de trajectòries amb el robot e-puck

Document: Memòria

Alumne: Arnau Carrera Viñas

Director/Tutor: Marc Carreras Pèrez
Departament: Arquitectura i Tecnologia de Computadors
Àrea: ATC

Convocatòria (mes/any): 04/09

INDEX

1.INTRODUCCIÓ

1.1. Antecedents

Al laboratori docent de robòtica s'utilitzen robots mòbils autònoms per treballar aspectes relacionats amb el posicionament, el control de trajectòries, la construcció de mapes... Es disposa de cinc robots comercials anomenats “e-puck” que es caracteritzen per les seves dimensions reduïdes, i un conjunt complet de sensors i actuadors. Aquests robots es poden programar amb diferents llenguatges utilitzant el simulador Webots, que disposa d'un conjunt de llibreries per usar el robot. Es disposa també d'un entorn de proves on els robots es poden moure i evitar obstacles. Per últim, al laboratori hi ha també càmeres i targes d'adquisició de dades.

1.2. Motivació

Els robots e-pucks s'empren majoritàriament a l'assignatura de màster Autonomous Robots. En aquesta assignatura no es disposa de suficient temps durant les pràctiques per aprofundir en el funcionament del robot aplicat al càlcul de posicionament utilitzant l'odometria i al control de trajectòries. Per aquest motiu s'ha decidit estudiar aquests temes amb més profunditat i així poder oferir als estudiants unes bases prèvies consistents en un software que ja contingui funcions per calcular el posicionament del robot i diversos algoritmes de control de trajectòries.

1.3 Objectius

Els objectius del projecte són els següents:

1. Estudi de la documentació i software proporcionats pels fabricants del robot i de l'entorn Webots.
2. Programació del software de l'odometria i realització de proves per comprovar-ne la precisió.
3. Disseny, programació i verificació del software dels algoritmes de planificació de trajectòries. Realització d'experiments per a comprovar-ne el funcionament.
4. Disseny, programació i verificació d'un sistema de visió artificial que permeti conèixer la posició absoluta del robot en l'entorn.

1.4. Planificació

La planificació del projecte ha consistit en dividir tota la feina en diferents parts i ordenar-les seguint l'ordre amb què són necessàries per poder realitzar-les.

A continuació, s'expliquen per ordre cronològic les diferents tasques que s'han creat i en què consisteix cadascuna d'elles.

1. Estudi de les característiques del robot e-puck i del simulador webots.
Per començar el projecte s'ha llegit la documentació referent al robot autònom e-puck proporcionada pel fabricant i s'ha comprovat la certesa d'aquests. També s'han llegit els

manuals referents al simulador webots.

2. Experimentació de la programació del robot e-puck a través del simulador webots.
Un cop estudiat l'entorn de programació s'han implementat senzills programes per tal de veure amb precisió el funcionament de l'e-puck conjuntament amb el webots.

3. Estudi de l'odometria i implementació en el robot e-puck.
Estudiar en què es basa l'odometria i com es pot aplicar al robot e-puck. Després, programar una funció que calculi l'odometria en el simulador webots. Per últim, comprovar que funcioni correctament a través de diversos experiments.

4. Estudi dels algorismes de control de trajectòries (algoritmes bug) i implementació en el robot e-puck
Estudiar com funcionen els algorismes de control de trajectòries, comprovar quins es poden aplicar utilitzant el sensor del robot. Dissenyar i implementar aquests algoritmes i comprovar el seu correcte funcionament amb diferents experiments.

5. Estudi del software per controlar el sistema de visió artificial i implementació d'un sistema que calculi el posicionament absolut.
Comprovar com es pot utilitzar el material de què es disposa al laboratori de robòtica amb l'objectiu de poder usar el sistema de visió artificial i implementar un programa utilitzant el matlab, que permeti calcular el posicionament absolut.

6. Comunicació entre el sistema de posicionament absolut i els algorismes de control de trajectòria.
Buscar una forma que permeti enviar informació d'una manera senzilla i ràpida entre els dos programes i implementar-la en tots els programes de control de trajectòria i posicionament absolut.

1.5. Estructura de la Memòria

El projecte està estructurat de la següent manera. Al capítol 2, s'explica tot l'entorn de treball, les eines utilitzades i les seves característiques. Al capítol 3, s'explica tota la part teòrica sobre l'odometria i els algorismes de control de trajectòries. Als capítols 4, 5, 6 es descriu el disseny i la implementació dels coneixements teòrics estudiats en els capítols prèvis. El capítol 4 tracta el posicionament mitjançant odometria. El capítol 5 presenta el posicionament absolut realitzat amb matlab. El capítol 6 tracta els algorismes de control de trajectòries. Finalment, al capítol 7 s'expliquen les conclusions a què s'ha arribat i els treballs futurs.

2. EINES DE DESENVOLUPAMENT

2.1.Webots

2.1.1.Introducció

El Webots robotics studio és un software per programar i simular robots mòbils. El projecte Webots va començar el 1996, al Swiss Federal Institute of Technology (EPFL) a Suïssa. Actualment, el desenvolupa Cyberbotics.

El Webots és un software usat tant per professionals com en desenvolupament universitari. Aquest software permet la creació d'entorns de treball simulats, també inclou una llarga llista de models de robots mòbils. Es disposa de llibreries en diferents llenguatges de programació (C/C++, Java i Python, i també Matlab, afegit en la última versió publicada aquest any) que fan possible programar els diferents models. Aquests programes es poden executar a través del Webots tant de manera simulada com de manera real. Per realitzar l'entorn de simulació es fa servir ODE (Open Dynamics Engine), ja que permet detectar els cosos sòlids i les possibles col·lisions.

2.1.2. Programació del robot

El robot e-puck és un robot mòbil de la llista de models ja dissenyats. S'ha escollit programar-lo amb el llenguatge de programació C/C++ degut a una familiarització més gran amb aquest llenguatge.

Qualsevol programa ha d'incloure les llibreries necessàries per poder utilitzar tots els sensors i actuadors de l'e-puck. Aquestes llibreries es lliuren al programador en el conjunt del Webots.

Els programes han de tenir una estructura per poder ser utilitzades. Aquesta estructura serà la següent:

- Una funció d'inicialització o reset en què s'hauran d'incloure totes les inicialitzacions dels sensors, actuadors i variables de control. També s'hi haurà d'incloure la freqüència amb què s'actualitzen els valors dels sensors i actuadors.
- Una funció de vida, que serà la que s'anirà executant d'una manera periòdica segons la freqüència anterior. Aquesta funció és la que inclourà tot el codi d'execució de la tasca que volem que realitzi el robot.

2.1.3. Entorn de Programació

L'entorn de programació està dividit en diferents seccions i cadascuna d'elles té una funcionalitat diferent. Aquestes seccions són les següents:

- **Log:** El log és la sortida estàndard de l'entorn de treball Webots. En aquesta secció, s'hi troba la sortida del nostre programa i els missatges d'error durant l'execució del codi (veure figura 1).
- **Sceen Tree:** En aquesta secció, hi trobem definit tot l'entorn de simulació: des dels

punts de llum de què disposa l'entorn fins als obstacles i els robots. També hi trobem definides qüestions com el fregament, la gravetat, les posicions de tots els objectes... Tots aquests valors poden ser modificats per tal que es pugui ajustar el màxim possible a les condicions de què es disposa en la realitat de l'entorn de treball (veure figura 2).

- **Món:** En aquesta secció és on veiem tot l'entorn de treball (món). La funcionalitat d'aquesta secció és veure com es comporta el robot simulat en l'entorn. En aquesta finestra es donen les ordres de simulació (engegar, parar, reiniciar, avanç ràpid...). A través d'aquesta secció podem veure l'execució del codi compilat pel projecte que estem simulant. Utilitzant el ratolí podem desplaçar-nos per tot l'entorn, fer zoom, etc. (veure figura 3).
- **Interfície del robot:** Aquesta secció ens deixa veure tots els sensors i actuadors que té el nostre robot i quin valor tenen si estem executant el codi del programa. Una altra opció d'aquesta secció és la de seleccionar quin robot s'utilitza i si es tracta de simulació o control remot (veure figura 4).
- **Editor de text:** En aquesta secció és on s'obre el codi amb què es programa la simulació. Aquest editor disposa de diferents funcionalitats que ajuden a programaa el nostre codi, com ara l'auto-completació de les funcions especials del nostre programa. Mitjançant aquest editor també podem compilar el nostre codi i veiem el resultat de la compilació a la subsecció que hi ha sota (veure figura 5).

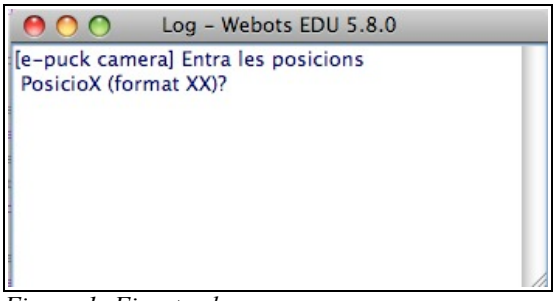


Figura 1: Finestra log.

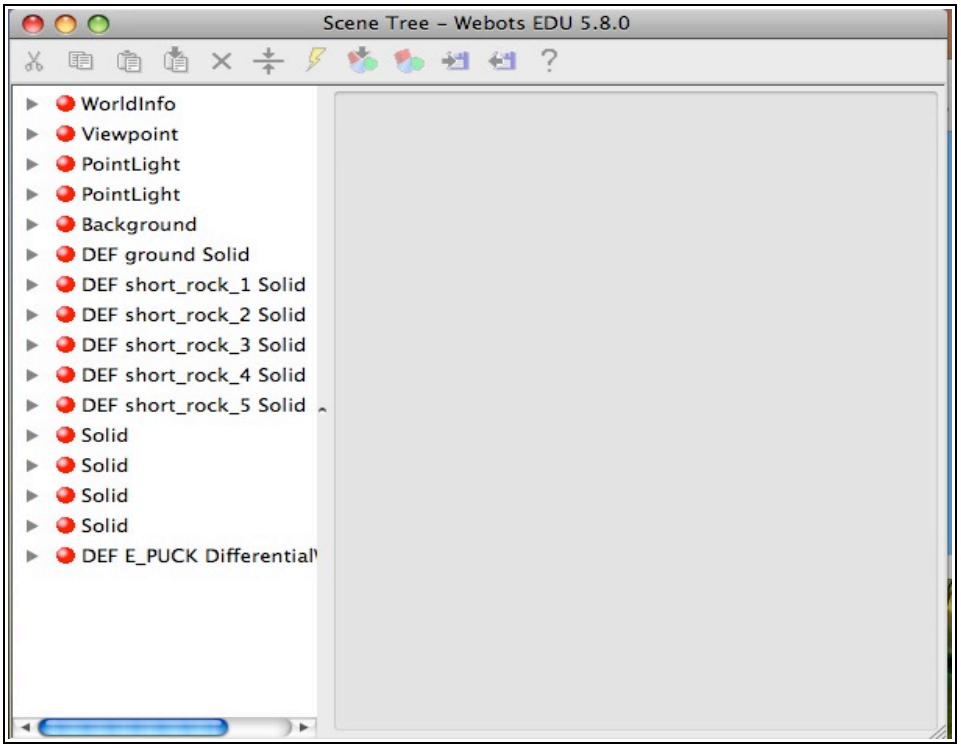


Figura 2: Finestra Scene Tree.

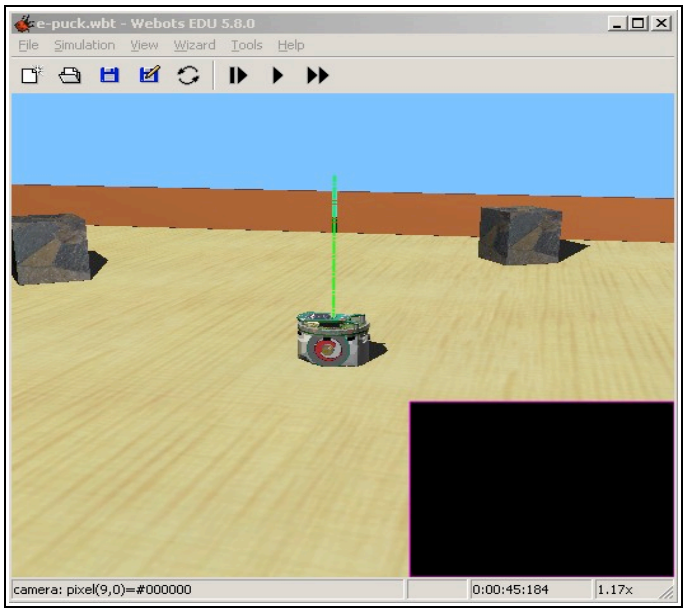


Figura 3: Món.

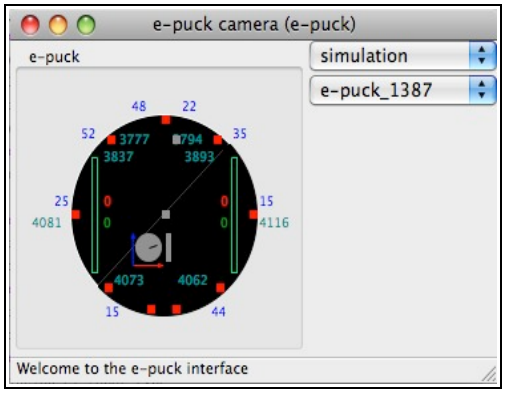


Figura 4: Interfície robot.

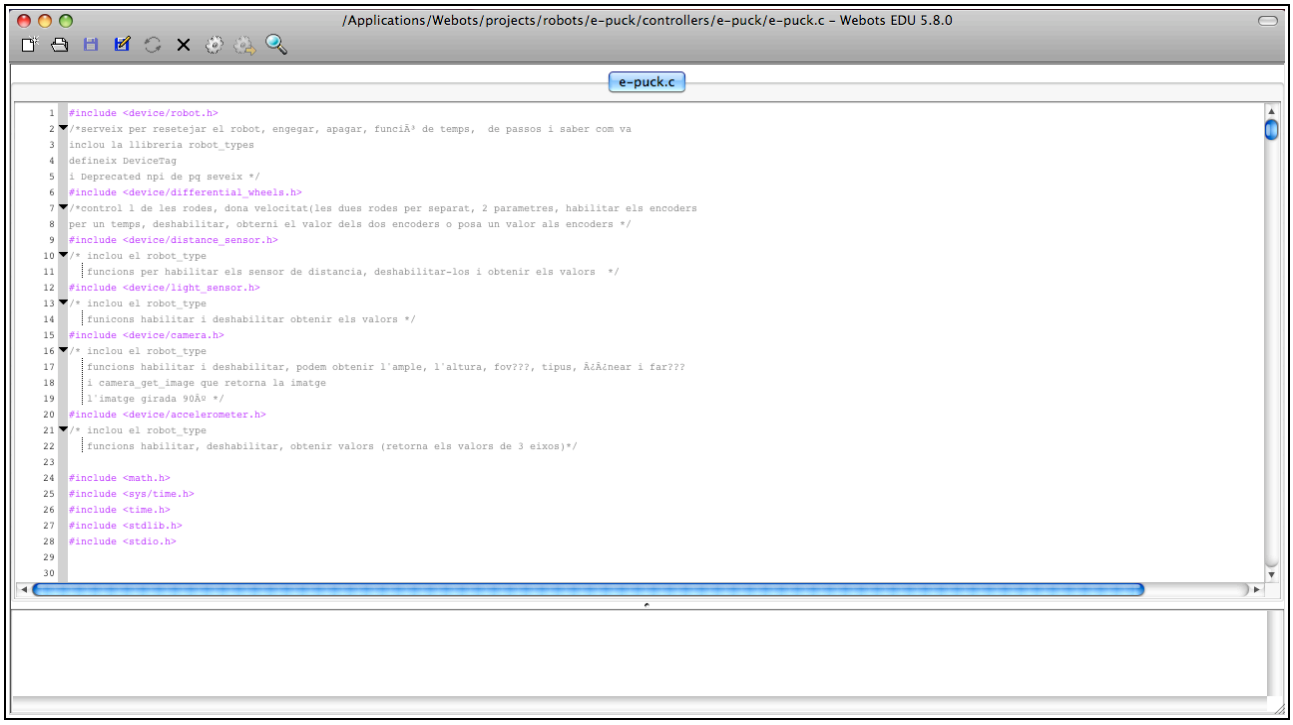


Figura 5: Editor de text.

2.1.4. Diferències entre el simulador i la realitat

El robot e-puck simulat i el real tenen una diferència important, sobretot en aquest projecte. La diferència més important és el fet que el valor obtingut per la funció que consulta els encoders obté un resultat diferent en el simulador i en el robot real. Això fa que en la programació s'han de canviar parts importants per interpretar correctament els valors obtinguts i conèixer la posició en que està la roda. Per aquest motiu s'ha pres la decisió d'utilitzar el robot e-puck real i no el simulat.

2.1.5. Connexió amb el matlab

L'ús del sistema de posicionament absolut requereix la utilització del programa Matlab i per aquest motiu el Matlab i el Webots necessiten enviar-se dades referents a posicions. Per tal de fer això, els dos programes es connecten mitjançant un socket TCP/IP. El Webots serà el programa que farà de client demanant a cada iteració la seva posició i el servidor serà el programa en Matlab que anirà responent les preguntes que li faci el cient.

2.2. Robot e-puck

2.2.1. Descripció general

El robot e-puck (veure figura 6) va ésser concebut com a un robot de baix cost orientat a docència i investigació en robòtica mòbil. Les dimensions del robot són de 7 cm de diàmetre i 4.2 cm d'alçada. L'estructura del robot és d'una sola peça on es poden encabir totes les parts que el formen.

Figura 6: robot e-puck.

A la part superior de l'estructura s'hi col·loca el circuit imprès; a l'interior, dos motors; i, per últim, a sota dels motors s'hi troba la bateria que es pot extreure amb facilitat. El conjunt dels sensor de proximitat i llum van col·locats al voltant del circuit imprès i la càmera se situa en un petit suport que hi ha sota la placa a la part frontal. Podem veure aquesta estructura a la figura 7.

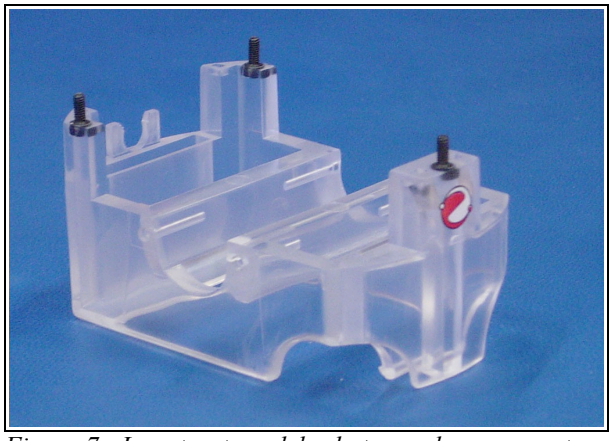


Figura 7 : La estructura del robot e-puck sense muntar.

El sistema d'eixos que utilitzem és el següent: l'eix X és el que segueix el moviment del robot e-puck, l'eix Y perpendicular amb l'eix X i segueix la direcció de l'eix de les rodes i, per últim, l'eix Z va ortogonal als eixos X i Y anant en el sentit del terra cap al sostre. Veiem la figura 8.

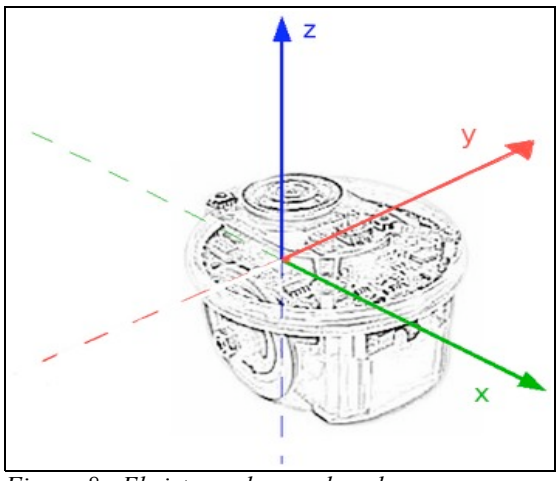


Figura 8 : El sistema de coordenades que utilitzarem en el robot e-puck.

L'encarregat de controlar tots els dispositius és un dsPIC, concretament el model 30F6014A. Les característiques principals són: la mida de la paraula és de 16 bits, utilitza una memòria de tipus Flash. La memòria del programa és de 144KB i disposa de 8 MB de RAM.

Per tal d'obtenir una visió general de tots els elements de què disposa l'e-puck, podem consultar-ne l'esquema elèctric (veure figura 9), on es pot apreciar tot el conjunt de sensors i actuadors que es descriuen a les seccions següents.

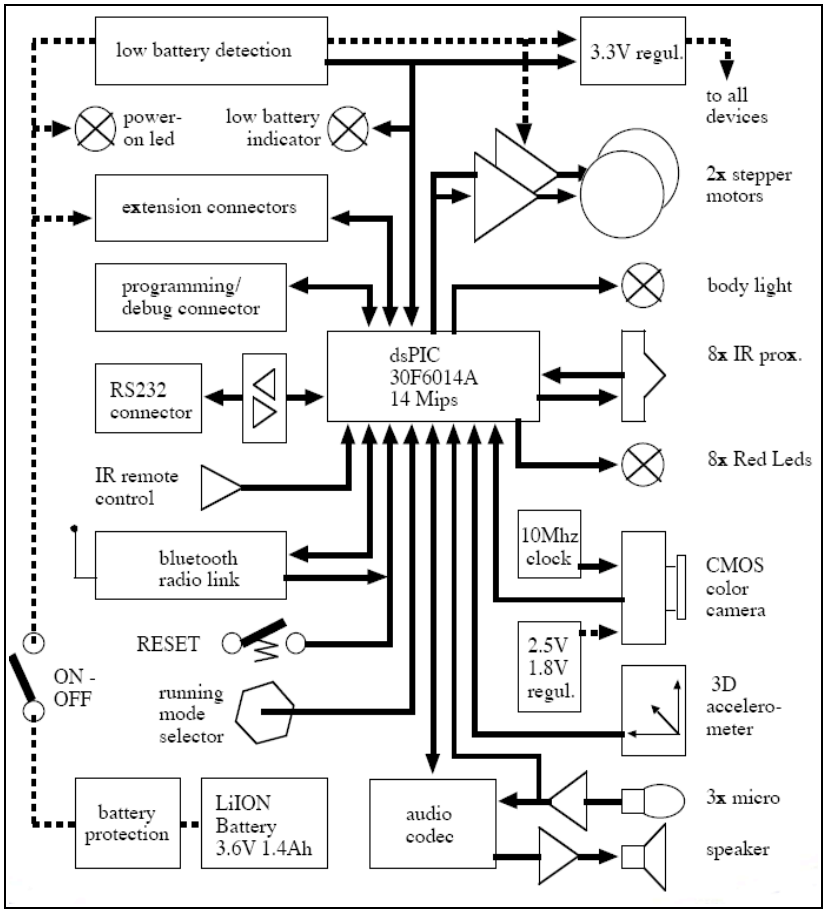


Figura 9: Esquema elèctric del robot e-puck on es poden observar tots els sensors i actuadors del robot.

2.2.2. Sensors

Els sensors són els dispositius que, a partir de l'energia del medi o d'una energia pròpia, donen un senyal (normalment elèctric) de sortida que sol ser una determinada funció de la variable que volem mesurar. Utilitzant els sensors podem mesurar fenòmens físics com poden ser la velocitat, l'acceleració, la distància... El robot e-puck disposa d'un ventall molt ampli de sensors per tal de poder detectar qualsevol canvi en el seu entorn més immediat i en ell mateix.

- **Sensors InfraRojos**

Aquests sensors detecten la proximitat dels objectes que envolten l'e-puck. El funcionament d'aquests sensors consisteix a enviar primer un feix de llum infraroja i a mesurar després la quantitat de llum que retorna. Si algun objecte es troba en el camí d'aquest feix, part de la llum s'hi reflectirà i tornarà al sensor. Segons la proximitat de l'objecte i la seva reflectivitat arribarà més o menys llum.

Amb la intenció de tenir el màxim control de tota la zona al voltant del robot els sensors estan col·locats d'una manera estratègica. A la taula podem veure els angles que corresponen a cada sensor respecte l'eix X del sistema de coordenades del robot e-puck que hem definit anteriorment , i en la seva posició en la Figura 10:

Sensors	Orientació respecta la direcció d'avanç (º)
IR0	-17,2344
IR1	-45,8823
IR2	-90
IR3	-151,489
IR4	156,9448
IR5	90
IR6	45,79
IR7	17,1431

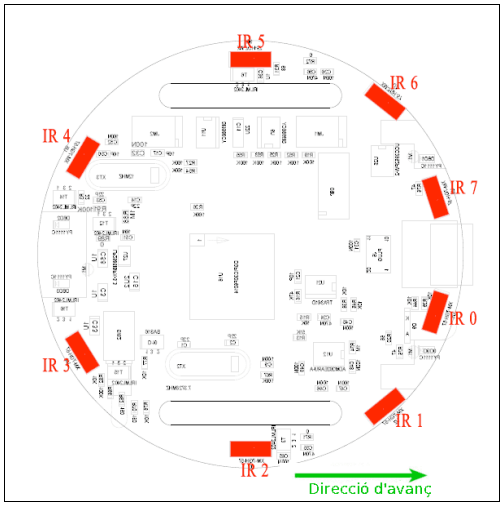
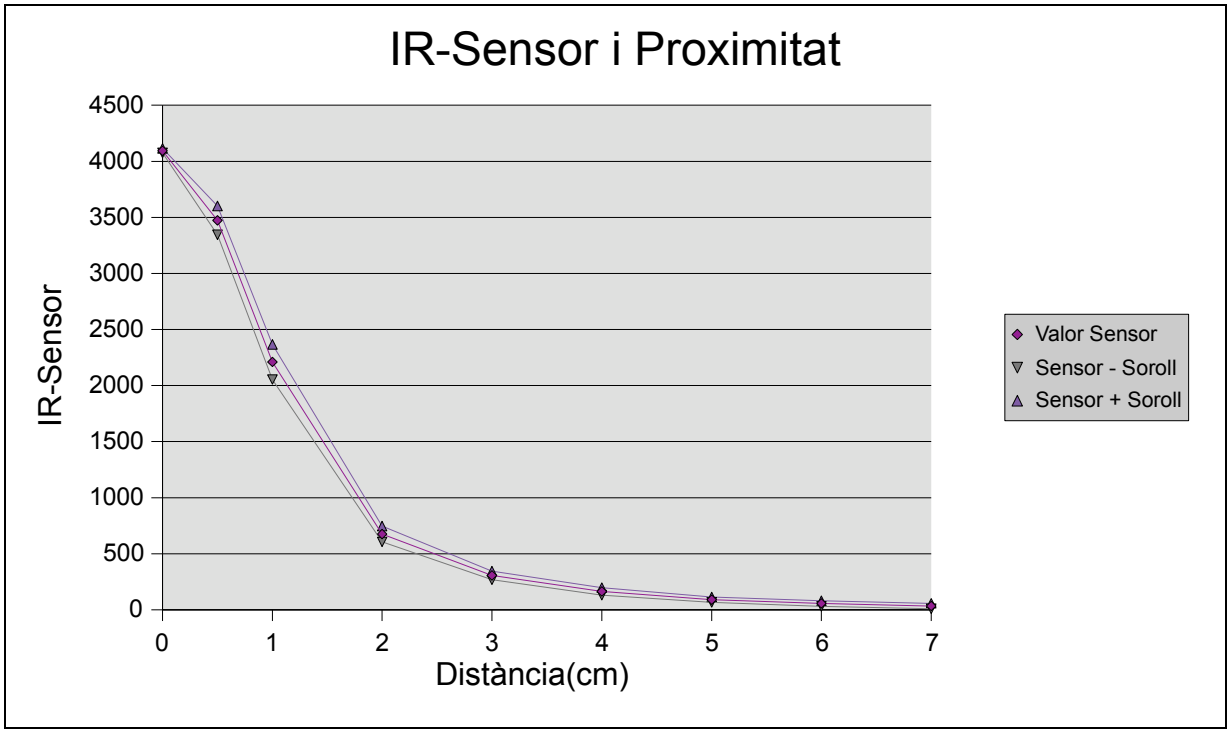


Figura 10: Sensors proximitat a l'e-puck.

Els valors que s'obtenen d'aquest sensor, per conèixer la distància de l'objecte que detecten, es calculen a través d'una transformació. El sensor no té un comportament lineal i per aquesta raó podem dividir la funció en diferents rectes. La taula següent mostra els valors de cada tram.

Distància(cm)	Valor Sensor	Sensor – Soroll	Sensor + Soroll	Soroll(%)
0	4095	4074,53	4115,48	0,5
0,5	3474	3345,46	3602,54	3,7
1	2211	2054,02	2367,98	7,1
2	676	605,02	746,98	10,5
3	306	267,75	344,25	12,5
4	164	130,22	197,78	20,6
5	90	65,79	114,21	26,9
6	56	31,47	80,53	43,8
7	34	10,06	57,94	70,4

La gràfica que obtenim és la següent:



- **Encoders**

El robot disposa de dos encoders que permeten conèixer la posició de cada roda del robot. En realitat no es disposa d'aquests sensors sinó que s'emulen en el firmware del propi robot. Això és possible perquè el robot utilitza dos motors pas a pas connectats a un reductor cadascun; aquests motors transformen un impuls elèctric en un determinat angle i això fa possible que es pugui emular els encoders.

- **Acceleròmetre**

El robot e-puck disposa d'un acceleròmetre que mesura les acceleracions lineals als tres eixos. A la figura 11 es pot observar a quina posició es troba l'acceleròmetre.

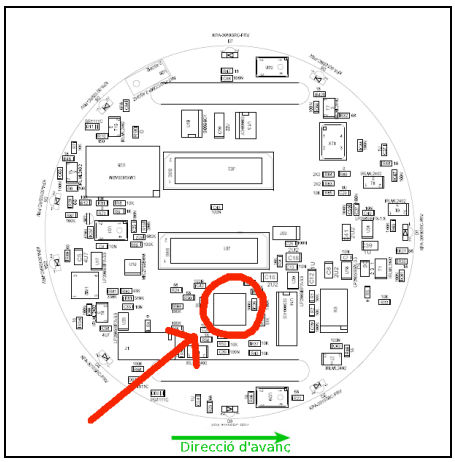


Figura 11: Lloc on hi ha situat l'acceleròmetre en el circuit imprès.

- **Càmera**

Es disposa d'una càmera CMOS en color, amb una resolució d'imatge de 640x480. La càmera no es pot fer servir en el seva màxima potència, ja que el microprocessador no és prou potent ni tampoc tenim prou capacitat de memòria per guardar la imatge completa. Per aquesta raó, es recomana utilitzar una resolució de 40x40 amb 4 frames per segon. La càmera està situada a la part frontal del robot tal i com podem observa a la figura 12.

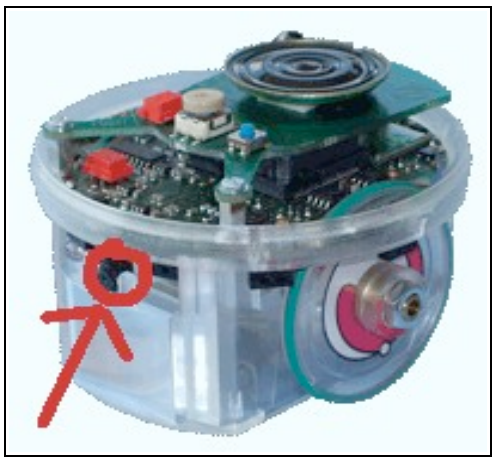


Figura 12 Imatge on es veu en el robot la posició de la càmera.

- **Micròfons**

El robot té tres micròfons situats d'una manera estratègica. Aquesta disposició permet triangular els sons i conèixer aproximadament on es troba l'origen del so respecte al robot. La màxima freqüència d'adquisició per cada micròfon és de 33KHz. A continuació, podem veure a la figura quina disposició tenen els tres micròfons.

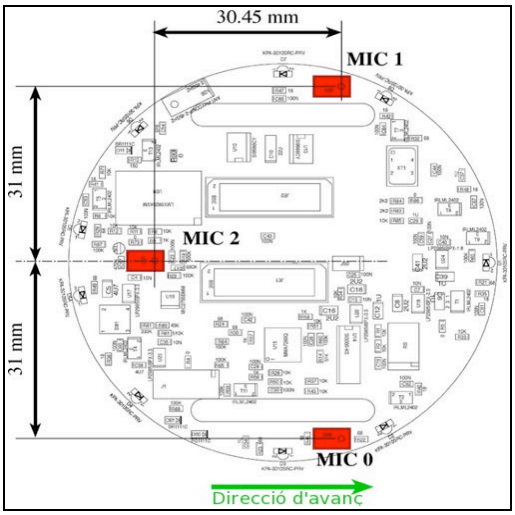


Figura 12: Situació dels 3 micròfons en el circuit imprès.

2.2.3. Actuadors

Els actuadors són dispositius que, a partir d'una energia normalment elèctrica, són capaços de modificar una determinada magnitud física. Utilitzant diferents tipus d'actuadors el robot e-puck pot interaccionar amb l'entorn. Disposa de tres tipus d'actuadors: lumínics, sonors i de moviment.

- **Leds**

El robot e-puck disposa de vuit leds col·locats sobre els sensors de proximitat (a l'anell superior que envolta el circuit imprès). Aquest leds són tots de color vermell. També disposa d'un led intern de color groc que il·lumina el cos del robot i, finalment, té també un led més potent al costat de la càmera. A la figura 13 podem observar el robot amb tots els leds actius i a la figura 14 tenim marcat l'anell de leds amb uns quants leds marcats i el led frontal.

Figura 13: Robot e-puck amb tots els leds oberts.

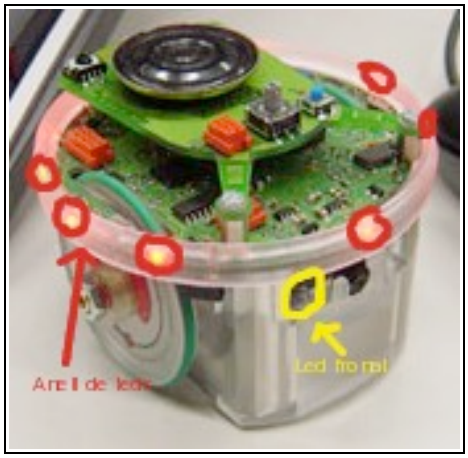


Figura 14: Robot e-puck amb el led frontal i uns quants leds del anell marcats.

- **Altaveu**

Es disposa d'un únic altaveu col·locat a la part superior del robot, des del qual es poden emetre sons. Es tracta de sons pensats per anar de l'altaveu als micròfons d'un altre robot per tal que aquest els pugui interpretar. Vegeu la figura 15 per comprovar on està situat l'altaveu.

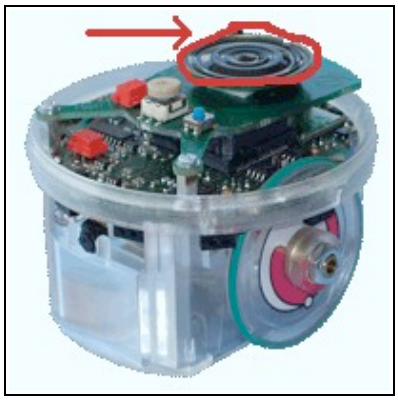


Figura 15: En la figura es veu marcat l'altaveu.

- **Motor**

El robot disposa de dos motors pas a pas de 20 passos per revolució connectats cadascun d'ells amb un reductor. La roda està connectada a l'eix que surt del reductor. El reductor té una relació de 50:1, on sigui, per cada 50 revolucions del motor pas a pas la roda realitza una volta. Per tant, cada 1000 passos del motor la roda haurà fet una volta.

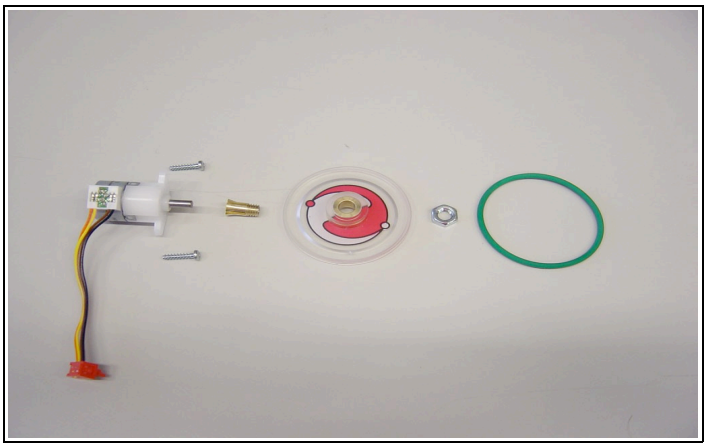


Figura 16: Imatge del motor amb el reductor i la roda al costat desmuntada.

2.2.4. Connectivitat

Per connectar-se amb el robot per tal d'enviar els programes necessaris tenim dues opcions possibles: una opció és utilitzant la connexió bluetooth mitjançant el receptor bluetooth i l'altra opció és utilitzant un connector port sèrie RS232 .

El bluetooth utilitza el xip LMX9820A. Aquest xip permet connectar-nos utilitzant l'uart amb un canal rfcomm de bluetooth. A través d'aquest canal ens comunicarem amb el robot com si es tractés d'un port sèrie amb l'excepció que els missatges de connexió i desconnexió són diferents. Per connectar-nos al robot, cal el codi PIN i s'ha d'introduir a cada connexió.

2.3. Processament d'imatges amb Matlab

2.3.1. Introducció

En el nostre entorn de treball es disposa d'una càmera de vídeo i també d'una targeta d'adquisició d'imatges, que utilitzarem a través del Matlab per calcular la posició absoluta del robot.

- **Càmera**

La càmera de vídeo que s'utilitzarà és una càmera de Sony SSC-DC198P (figura 17) que disposa d'un CCD de transferència interlineal tipus 1/3. Té un sistema d'exploració de 625 línies. Té una resolució de 768 x 576 píxels i pot adquirir imatges a 25 fps. El sistema de senyal que empra és el PAL estàndard i transmet les imatges per cable de vídeo compost.

Figura 17: Càmera.

- **Targeta**

Per tal de poder adquirir les imatges de la càmera, disposem d'una targeta d'adquisició Matrox Meteor II PCI (figura 18), que ens permet processar les imatges enviades per la càmera.

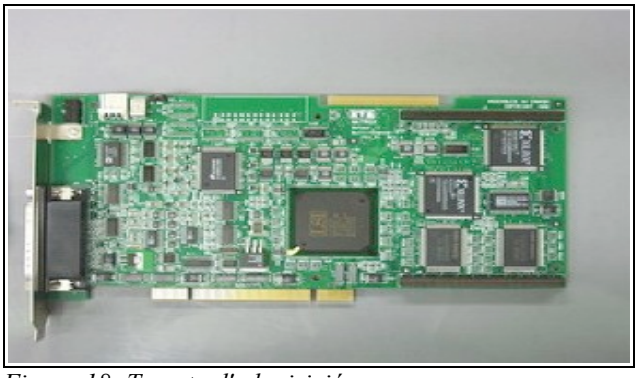


Figura 18: Targeta d'adquisició.

És capaç d'adquirir tant imatges en color com en blanc i negre. Pot transferir les imatges obtingudes a la memòria de l'ordinador per tal de processar-les o a la targeta gràfica per tal de mostrar-les directament. Disposa d'una interfície sèrie RS-232 que permet controlar

remotament una càmera. Els formats de vídeo que suporta són els estàndards: PAL i NTSC. Disposa d'una memòria RAM de 4 MB.

2.3.2. Càlcul de la LUT

Per fer el tractament de la imatge utilitzarem una funció que calcula una LUT (Look Up Table) de 3 colors creada per Ricard Batllori Niubó al seu projecte de final de carrera amb títol *Entorn de programació pel robot Staubli mitjançant Matlab, aplicació a un sistema de recol·lecció de peces basat en visió artificial.*

Abans d'explicar el funcionament d'aquest script en matlab, es farà un petit recordatori sobre què són els espais de color, ja que és necessari tenir uns coneixements mínims sobre aquests temes per entendre l'script.

- **Model de color**

Un model de color és un model matemàtic de forma abstracta que permet representar els colors utilitzant un conjunt de valors o components cromàtics. Existeixen diferents models, nosaltres n'utilitzarem dos de concrets: l'RGB i l'HSV.

RGB: correspon a Red Green Blue (Vermell, Verd, Blau). Aquest model de color té tres valors. Aquest model està basat en la síntesi additiva, que significa que és possible representar un color amb la barreja per addició dels tres colors de llum primaris. Els valors que poden tenir van de 0 a 255 en la majoria dels casos. El conjunt de tots els colors es pot representar com un cub (veure figura 19) i els colors són punts tant de la superfície com de l'interior d'aquest cub.

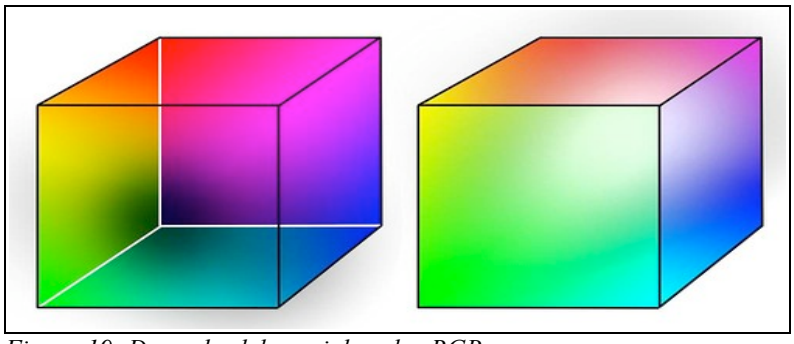


Figura 19: Dos cubs del espai de color RGB.

HSV: significa Hue, Saturation, Value (Tonalitat, Saturació, Valor). Aquest model de color té tres valors. També s'anomena HSB (Hue, Saturation, Brightness – Tonalitat, Saturació, Brillantor). Defineix un model de color basat en termes dels seus components que es constitueixen en coordenades cilíndriques (veure figura 20).

La tonalitat és el component que representa quin tipus de color es tracta. Els valors es representen del 0 a 360 (normalment està normalitzat en tant per cent).

La saturació és la distància que hi ha a l'eix de brillantor blanc-negre. Els valors es representen en tant per cent. Com menor sigui la saturació d'un color, major tonalitat grisenca hi haurà i més descolorit estarà.

El valor de color o la brillantor representa l'altura a l'eix blanc i negre. Els valors també són representats en tant per cent. El 0 és negre i el 100, blanc o un color més o menys saturat.

El model HSV consisteix en una transformació no lineal del model de color RGB i es pot utilitzar en progressions de color.

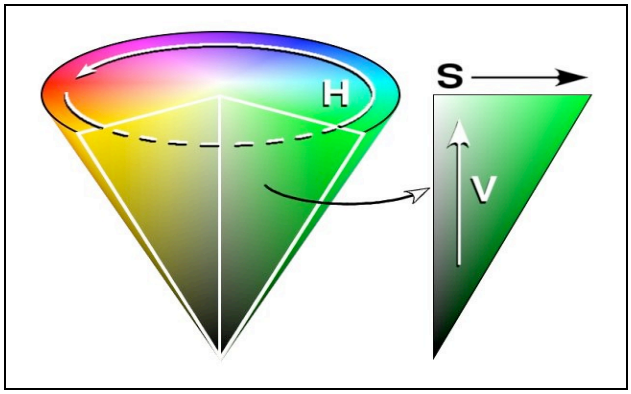


Figura 20: Con de color que representa el model de color HSV.

Ara ja coneixem tot el que ens fa falta per poder entendre l'script que calcula la LUT de 3 colors.

- **Script**

A partir de l'script inicial creat per Ricard Batllori, l'hem adaptat a les nostres necessitats. En aquest punt només explicarem d'una manera poc profunda què feia el seu script i, més endavant, ja comentarem la adaptació que hem fet del seu codi per al nostre propòsit.

L'objectiu de l'script és buscar tres objectes de diferents colors. Per aconseguir aquest objectiu se segueixen quatre passos:

Pas 1: Seleccionar els color que tindran els objectes
Pas 2: Transformar l'espai de color de RGB a HSV
Pas 3: Crear la LUT amb els valor que s'han trobat anteriorment
Pas 4: Determinar a partir de la LUT on es troben aquest objectes

En el primer pas, es veu una imatge del entorn i, amb el ratolí, s'han de seleccionar a la imatge els colors que volem classificar.

En el segon pas, es canvia el model de color, ja que HSV ens permet fer millor la segmentació de la imatge. Això es deu al fet que dos colors semblants poden tenir valors molt diferents en el model de color RGB i, en canvi, en HSV tindran valors semblants. En l'espai HSV només utilitzarem els dos primers components, la tonalitat i la saturació. Per a cada color s'agafaran el valor màxim i mínim que es troben dins la finestra seleccionada.

En el tercer pas, s'utilitza una funció per calcular la LUT. Aquesta funció crea una taula de 256*256*256 on es trobaran totes les combinacions possibles de RGB. A cada combinació se li assignarà un valor. Hi haurà quatre possibilitats, cada color seleccionat tindrà un valor

diferent i, si no és un color seleccionat, tindrà un altre valor.

En el darrer punt, s'aniran agafant imatges de la càmera a 12 fps (s'ha considerat que és una velocitat suficient per poder treballar sense que es col·lapsi l'ordinador) i a cada imatge s'aniran buscant el colors dels tres objectes per determinar-ne el centre.

A continuació, a la figura 21, podem observar el resultat d'aplicar aquesta funció en una imatge obtinguda per Ridard Batllori al seu projecte.

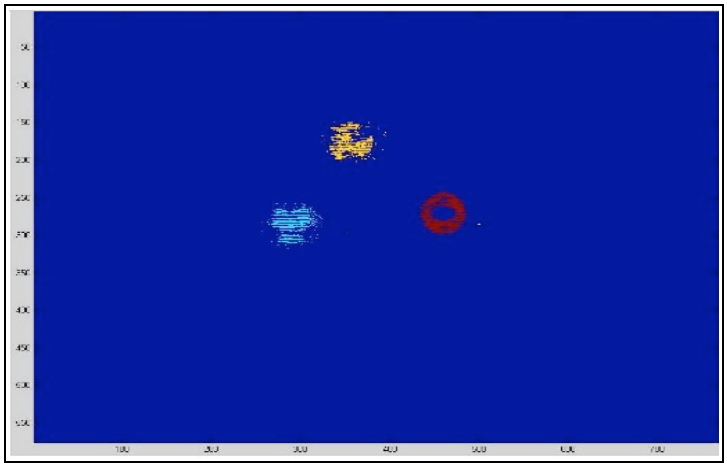


Figura 21: Imatge resultant d'aplicar el script del projecte d'en Ricard Batllori.

3. POSICIONAMENT I CONTROL DE TRAJECTÒRIES D'UN ROBOT MÒBIL

3.1. Introducció

Per a un robot autònom mòbil és molt important conèixer a quina posició es troba en cada moment, ja que per poder assolir la posició objectiu ha de saber la seva orientació i posició actual, la posició objectiu i com pot arribar-hi. Per tal que el robot pugui calcular a quina posició es troba a partir de conèixer la posició inicial, s'utilitzarà l'odometria.

Per una altra banda, a part de la posició també s'ha de definir quin serà el comportament del robot per tal d'evitar possibles obstacles que es trobi pel camí.

Per esquivar objectes hi ha diferents algorismes. En el nostre cas només utilitzarem uns dels més simples que són els Bug 0, 1 i 2. No són algorismes perfectes però, tot i això, són molt útils i senzills d'aplicar.

3.2. Odometria

3.2.1. Definició

L'odometria és una paraula d'origen grec formada per dos mots “Hodos” (viatjar, trajectòria) i “Metron” (mesura).

Definim odometria com l'estimació de la posició d'un vehicle a partir del moviment de les seves rodes durant la navegació. Aquesta posició és relativa a la posició d'origen del robot i és una estimació i no pas una posició absoluta, ja que és molt fàcil que s'acumulin errors.

L'odometria és útil en desplaçaments curts, perquè com més llarg sigui el desplaçament cada cop s'acumula un error més gran. S'utilitza en desplaçaments curts, ja que és molt simple d'implementar, permet un mostreig molt ràpid i l'error serà molt petit. Per usar l'odometria el robot només necessita que les rodes disposin d'algun tipus de sistema que sigui capaç de detectar el moviment de les rodes com ara els encoders.

3.2.2. Dificultats i errors en l'odometria

L'odometria està basada en el fet que la revolució de la roda es transforma en una distància, però a vegades això no és cert, ja que la roda pot patinar i no es produeix desplaçament però sí canvis en els encoders. Aquest error és un dels més importants, tot i que hi poden haver altres causes. A continuació, disposem d'un conjunt d'errors classificats segons si són sistemàtics o ocasionals.

Errors sistemàtics:

- Les dues rodes tenen petites diferències en els seus diàmetres.
- Les rodes estan alineades de forma incorrecta.
- La resolució dels encoders és discreta i no continua. També pot ser que la freqüència de mostreig dels encoders sigui massa gran.

Errors ocasionals:

- Desplaçament en terres desnivellats.
- Patinatge de la roda degut a: superfícies lliscants, sobre acceleració, derrapatge, forces externes (col·lisió amb altres cosos), falta de punt de contacte amb el terra.

Els errors que són més difícils d'eliminar i que causen pitjors erros són els sistemàtics, ja que s'acumulen constantment. Per aquest motiu sempre s'han de tenir en compte i procurar reduir-los al mínim. Els ocasionals poden passar en qualsevol moment.

3.2.3. Càlcul de la posició

Abans de calcular l'odometria és necessari conèixer algunes característiques del robot. Aquestes característiques són:

- La longitud de la roda. Aquesta característica l'obtenim a partir del radi de la roda i aplicant la fórmula matemàtica de la longitud (2*π*Radi).
- El nombre d'increments que obtenim per una revolució de la roda. Aquest valor l'obtindrem a través de les característiques del robot.
- La distància que hi ha entre les dues rodes. La distància correcta és aquella que uneix les dues rodes i és perpendicular a les dues rodes, és la mateixa distància que hauria de tenir l'eix que les uniria.

Les fórmules matemàtiques que ens permeten conèixer la distància aproximada recorreguda per les rodes i el canvi en l'orientació durant aquest desplaçament són:

La distància incrementada serà la mitjana de la distància recorreguda per les dues rodes (veure figura 18)
Distància: D = (DistanciaR + DistanciaL / 2)

L'orientació que agafa el robot l'obtenim suposant que el càlcul de l'odometria és molt freqüent, per tant els increments són molt petits. Per aquest motiu, podem suposar que la diferència entre les distàncies recorregudes per cada roda és igual a l'arc d'una circumferència. Així doncs, sabent això podem utilitzar la fórmula $Arc = Radi \cdot angle$ on com ja hem dit l'arc és la diferència recorreguda per les dues rodes i el radí és la distància que separa les dues rodes, és a dir, l'ample d'eixos (veure figures 22 i 23).

Orientació : $O_{t+1} = O_t$ + (DistanciaR – DistanciaL) / Ample d'eixos(W).

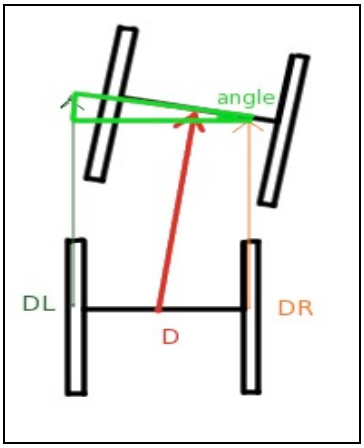


Figura 22: Representació Distància mitjana, de la roda dreta , esquerra i l'angle.

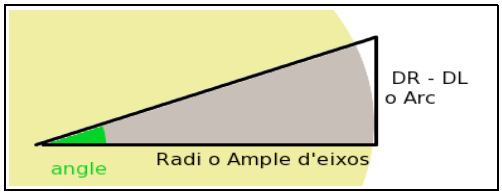


Figura 23: Comparació de l'arc amb l'angle.

Podem aplicar aquestes dues fórmules per tal de conèixer la posició en els eixos X i Y, ja que es tracta d'un triangle rectangle (veure figura 24):

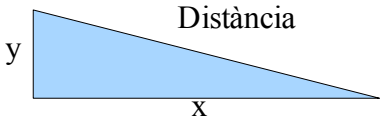


Figura 24: Triangle rectangle.

Posició X: $X_{t+1} = X_t + D * \cos(O_{t+1})$
Posició Y: $Y_{t+1} = Y_t + D * \sin(O_{t+1})$

3.3. Algorismes “Bug”

Els algorismes Bug són algorismes creats per poder evitar obstacles utilitzant estratègies reactives simples. Aquests algorismes estan inspirats en l'observació dels moviments realitzats pels insectes en la natura, d'aquí el seu nom.

Els algorismes Bug estan pensats per poder aconseguir objectius globals només amb coneixement local de l'entorn (és a dir, coneixem una posició objectiu global però no tenim coneixement de com és l'entorn, només sabem el que els nostres sensors detecten a mesura que ens acostem a l'objectiu).

Existeixen 4 bugs: Bug 0, Bug 1, Bug 2, i Bug tangent. Els tres primers bugs només necessiten sensor de contacte o sensor de distància, però no han de tenir un abast molt gran, com és el cas de l'e-puck. Pel quart bug fa falta tenir un sensor de rang que pugui detectar els obstacles que hi ha dins una àrea de 360º al voltant del robot.

Per tal de poder implementar els bug del 0 al 2 només és necessari que el robot sigui capaç de seguir parets, dirigir-se cap a l'objectiu, conèixer la direcció on es troba l'objectiu i disposar de sensor tàctils.

3.3.1. Bug 0

El bug 0 és un bug molt senzill que consisteix a seguir uns passos molt senzills que explicarem a continuació.

En primer lloc, el robot s'ha d'orientar cap a l'objectiu fins que pugui anar cap ell directament. Un cop orientat el robot comença a avançar.

Mentre no arribi a la posició objectiu o es trobi amb un obstacle al davant continua avançant. Si arribem a la posició objectiu, hem acabat.

Si trobem un objecte al davant hem de decidir per quina banda anem, això pot ser una solució aleatòria, o podem haver pres la decisió prèviament i que sempre s'apliqui la mateixa. Un cop estem seguint l'obstacle el continuarem seguint fins que puguem tornar anar en direcció a l'objectiu.

En el diagrama de la figura 25 podem veure tota aquesta explicació però mitjançant un diagrama.

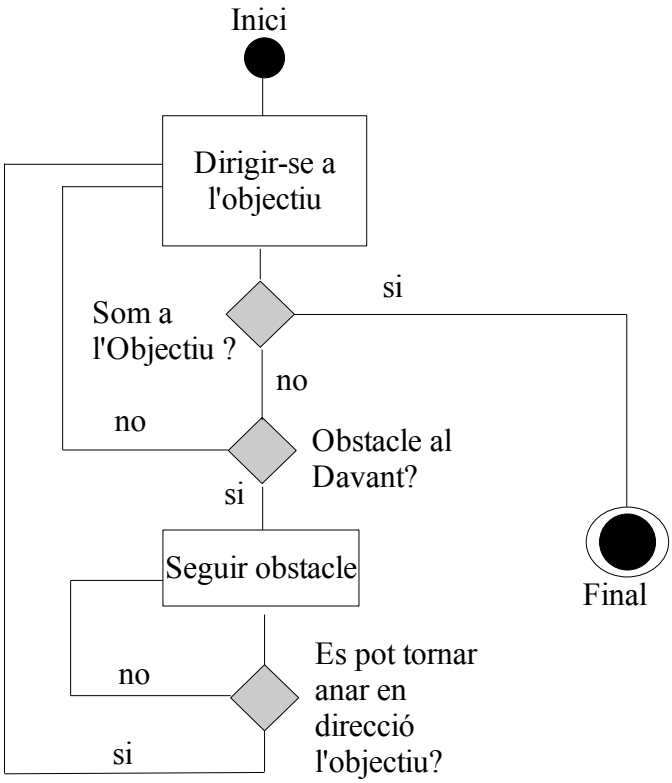


Figura 25: Diagrama explicatiu del Bug 0.

Tot seguit, a la figura 26, podem observar un exemple de com funciona aquest bug en un cas molt senzill.

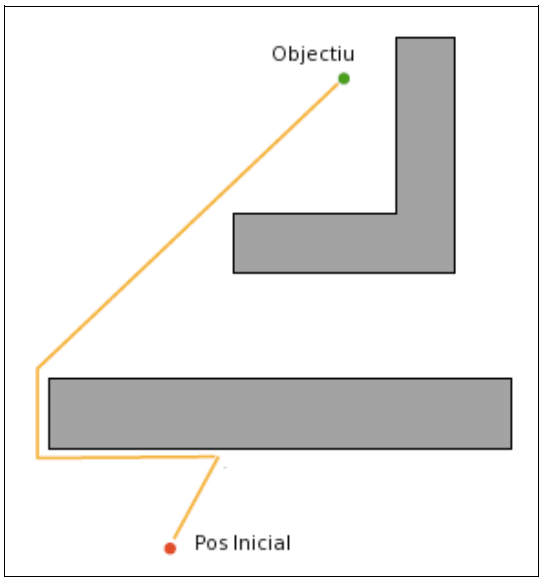


Figura 26: Exemple del comportament del Bug 0.

Com es pot observar, aquest bug té un problema, ja que segons quina forma té l'obstacle el robot no trobaria mai el seu objectiu (podem observar-ho a la figura 27). Com que és un bug poc robust no l'implementarem i només s'implementaran el Bug 1 i el Bug 2.

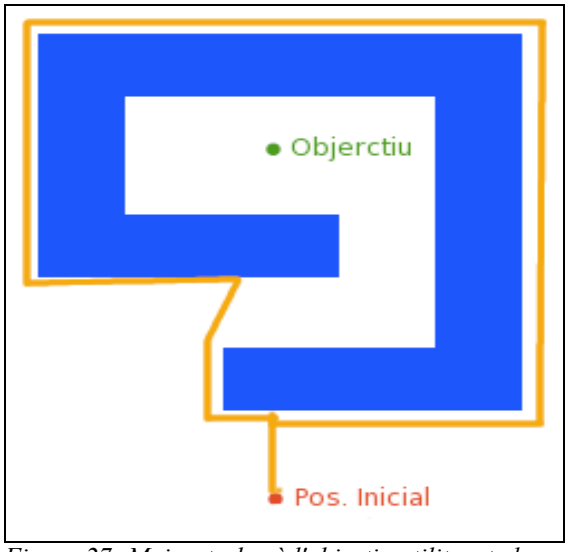


Figura 27: Mai es trobarà l'objectiu utilitzant el bug 0 i girant a l'esquerra.

3.3.2. Bug 1

Les instruccions que s'han de seguir per utilitzar el bug 1 són les següents:

En primer lloc i com en el bug 0, hem d'orientar el robot i començar a avançar fins que arribem a la posició objectiu o ens trobem amb l'obstacle.

Quan ens trobem amb un obstacle l'hem de rodejar completament tot ell i, a mesura que es rodeja l'obstacle, hem de calcular quina és la distància que hi ha d'aquella posició fins a la posició objectiu. Només recordarem la posició on hi ha la distància més curta fins a la posició objectiu.

Un cop siguem a la posició on hem trobat l'obstacle, hem d'anar cap a la posició que hem trobat que era més aprop de la posició objectiu, buscant el camí més curt. Quan arribem allà hem de tornar a començar les instruccions com si ens trobéssim a la posició d'origen.

A les figures 28 i 29 es pot observar un exemple de Bug 1 amb dos obstacles i un diagrama on s'explica com executar el bug 1.

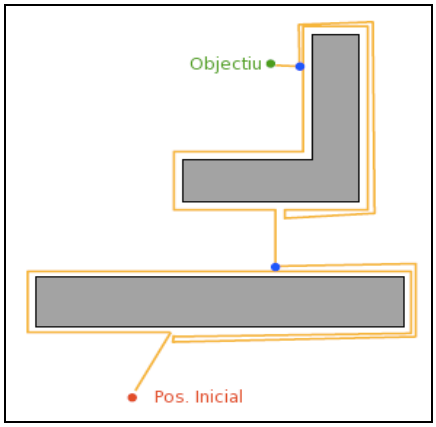


Figura 28: Exemple del Bug 1.

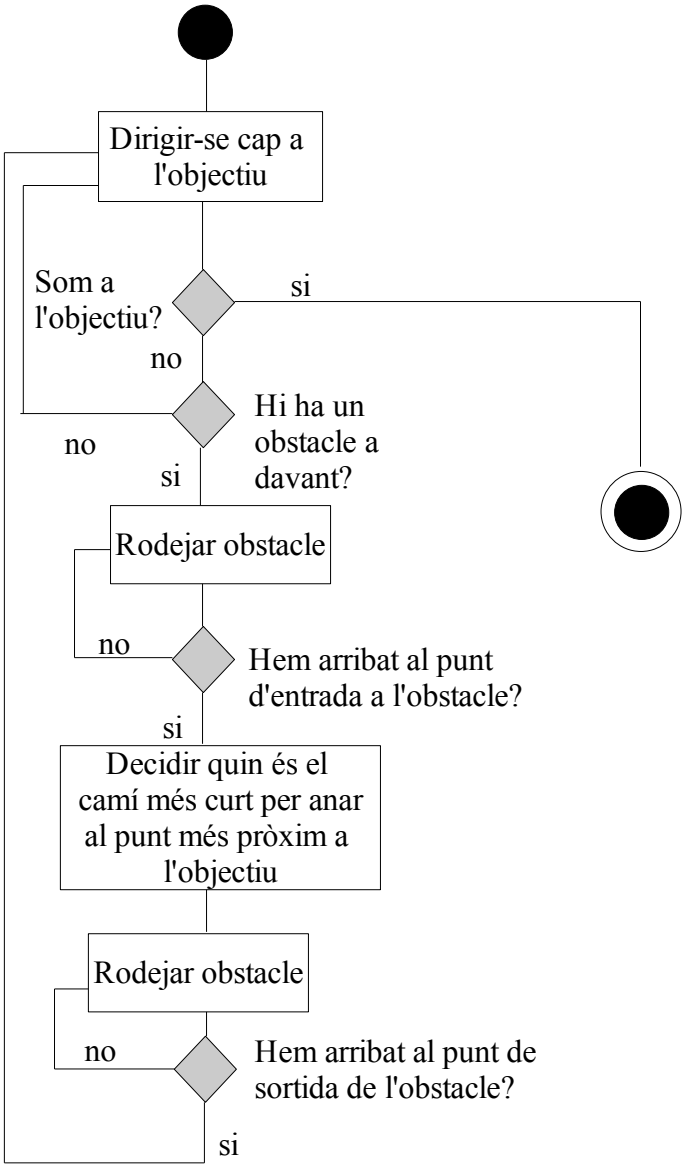


Figura 29: Diagrama explicatiu del Bug 1.

La distància mínima que recorrerà el robot que utilitza el bug 1 serà la distància que hi ha des de la posició d'origen a la posició objectiu. Aquesta distància l'anomenarem D.

La distància màxima que recorrerà el robot que utilitza el bug 1 serà la distància que hi ha des de la posició d'origen a la posició objectiu més un cop i mig el perímetre de cada obstacle que trobem, ja que s'ha de rodejar completament (una volta) i després arribar fins a la posició més pròxima a l'objectiu, que en el pitjor dels casos serà mitja volta.

Dist. mínima : D Dist. màxima : $D + 1.5x \sum_i P_i$

3.3.3. Bug 2

Les instruccions que s'han de seguir per utilitzar el bug 2 són les següents:

En primer lloc, hem de crear la m-line. La m-line és una línia que va des del punt d'origen fins a la posició objectiu pel camí més curt, és a dir, en línia recta. Un cop es coneix la m-line s'ha de començar a seguir.

Si seguint la m-line el robot es troba amb un obstacle, hem de començar a seguir aquest obstacle. Tal com passa amb el bug 1, es pot seguir l'obstacle tant per la dreta com per l'esquerra. Aquesta pot ser una decisió aleatòria o pot estar sempre definida per seguir els objectes per la mateixa banda.

Deixàrem de rodejar l'obstacle quan ens retrobem amb m-line i aquesta posició de la m-line estigui més aprop de l'objectiu que la que hem trobat l'objectiu, també hem de tenir en compte que no haguem passat ja abans per aquesta posició, si fos així continuaríem fins tornar a troba la m-line en un altre punt.

A les figures 30 i 31 es pot observar un exemple de Bug 2 amb dos obstacles i un diagrama on s'explica com executar el bug 2.

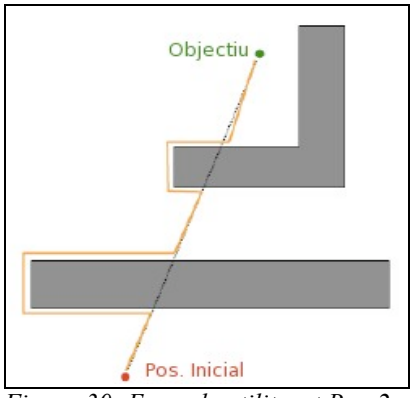


Figura 30: Exemple utilitzant Bug 2.

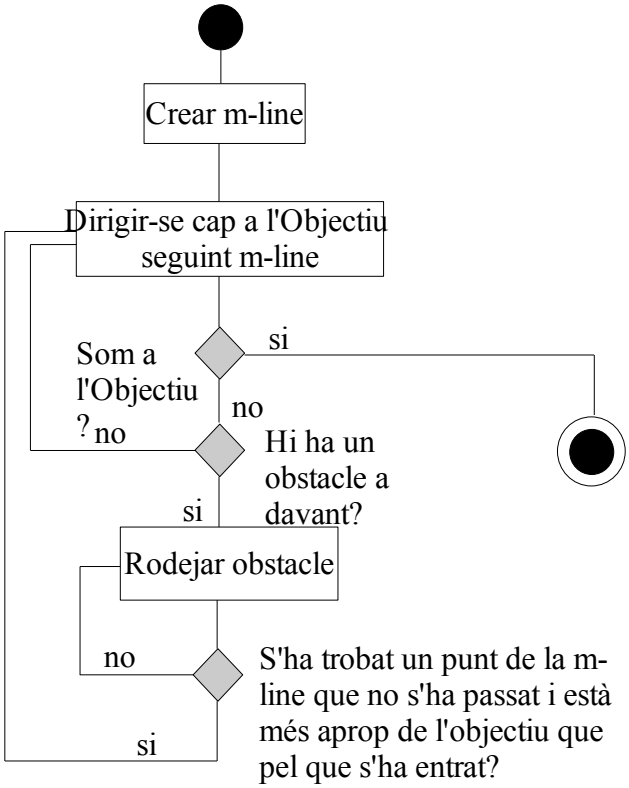


Figura 31 Diagrama explicatiu del Bug 2.

La m-line és la línia que va des de la posició inicial fins a la posició objectiu, sense tenir en consideració cap dels obstacles que es pugui trobar pel camí.

La distància mínima que recorrerà el robot que utilitza el bug 2 serà la distància que hi ha des de la posició d'origen a la posició objectiu. Aquesta distància l'anomenarem D.

La distància màxima que recorrerà el robot que utilitza el bug 2 serà la distància que hi ha des de la posició d'origen a la posició objectiu més n cops, on n és el nombre de cops que interseccciona la m-line amb l'obstacle, el perímetre de cadascun dels obstacles. Està dividida per dos perquè cada intersecció amb la m-line tindrà una intersecció d'entrada i una de sortida.

Dist. mínima : D Dist. màxima : $D+\sum_i (\frac{n_i \cdot P_i}{2})$

3.3.4. Bug 1 i Bug 2

Tant al bug 1 com al bug 2, la funció rodeja obstacle es pot fer per qualsevol de les dues bandes. Aquesta pot ser una decisió aleatòria del programa o decidida prèviament i fer-se sempre així.

Entre aquest dos algorismes no n'hi ha un que sigui millor que l'altre, sinó que depèn del tipus d'obstacle que es trobi entre la posició inicial i la posició objectiu (un bug recorrerà menys distància que l'altre).

Els dos bugs tenen dos comportaments molt diferents. El bug 1 és un bug que pren en consideració totes les opcions i agafa la millor opció. Aquest tipus d'algorisme s'anomena de cerca exhaustiva. El bug 2 és un bug que escull la primera opció que sembla bona. Aquest tipus d'algorismes reben el nom d'algorisme “greedy” (golafre).

4. ODOMETRIA AL ROBOT E-PUCK

4.1. Introducció

Abans de poder calcular les expressions matemàtiques que calculen l'odometria és necessari conèixer d'una manera tant exacte com sigui possible les característiques de l'e-puck. Per això buscarem en les característiques tècniques els valors d'algunes dades i les confirmarem amb el nostre propi robot. D'altres, en canvi, les calcularem a partir de dades obtingudes.

4.2. Càlcul de les constants per utilitzar l'odometria en l'e-puck

En aquesta secció, calcularem totes les constants necessàries per tal de poder aplicar l'odometria al robot e-puck.

És necessari conèixer la distància que recorre una roda en donar una volta. Aquest valor el trobarem calculant la longitud que té la roda del robot e-puck:

$$Longitud = 2 * \pi * Radi = 2 * \pi * 2.05\,cm = 12.88\text{cm}$$

Relacionat amb la longitud de la roda, s'han de conèixer el nombre d'increments que fan falta per tal de recorre aquesta distància. Obtenim aquest valor amb amb un senzill factor de conversió que obtenim mitjançant les característiques dels motors i reductors que té el robot:

$$num.\,increments\ per\ una\ volta = \frac{20\text{ increments}}{1\text{ voltes motor}} \cdot \left(\frac{50\text{ voltes motor}}{1\text{ volta roda}}\right) = 1000\,increments$$

Per calcular l'orientació del robot és necessari conèixer la distància que separa les dues rodes. Aquest valor és l'amplada d'eix del robot.

$$Ample\ del\ eix = 5.2\,cm$$

Per saber la distància que recorre cada roda utilitzarem el valor que s'obté a través dels encorders simulats. Aquest valor l'obtindrem creant una constant mitjançant el nombre d'increments que ha de fer la roda per tal de donar una volta i la longitud que té. A continuació, veiem la fórmula resultant.

$$Distància = \frac{\text{num. increments}}{\text{num increments per volta}} \cdot \text{Longitud} = \frac{\text{num increments}}{1000} \cdot 12.8805 = \frac{\text{num. increments}}{77.63}$$

Podem veure que la constant que hem obtingut és 77.63. L'anomenarem cte.

Amb totes aquestes dades ja és possible calcular l'odometria amb les fórmules de l'odometria. Tot i això per intentar evitar errors sistemàtics per culpa d'un càlcul poc precís (que una constant no sigui del tot correcta), s'ha decidit realitzar una calibració prèvia per ajustar al màxim la constant que utilitzem per calcular la distància.

4.3. Calibració

Per poder calibrar el robot, hem dissenyat tres tipus d'experiments mitjançant els quals podrem comprovar si la constant teòrica és suficientment bona o si la podem millorar. Aquests tres experiments són l'estimació en X, l'estimació del Yaw i l'estimació en moviment combinat.

Els experiments d'estimació en X i del Yaw els utilitzarem per adaptar les nostres constants. En canvi, el moviment combinat només ens servirà per adonar-nos de si els canvis han proporcionat alguna millora.

4.3.1 Resultats

- **Estimació en X**

Per calcular l'estimació en X es programarà el robot e-puck per tal que avanci 10 cm i s'aturi en aquell punt.

En l'entorn de treball, es col·loca a terra paper mil·limetrat per tal de poder mesurar la distància recorreguda pel robot i també es realitzen unes marques per col·locar sempre el robot a la mateixa posició i saber exactament a quina posició s'hauria d'aturar. Podem observar-ho a les figures 32 i 33 on es veuen les dues marques i el robot sobre una de les marques.

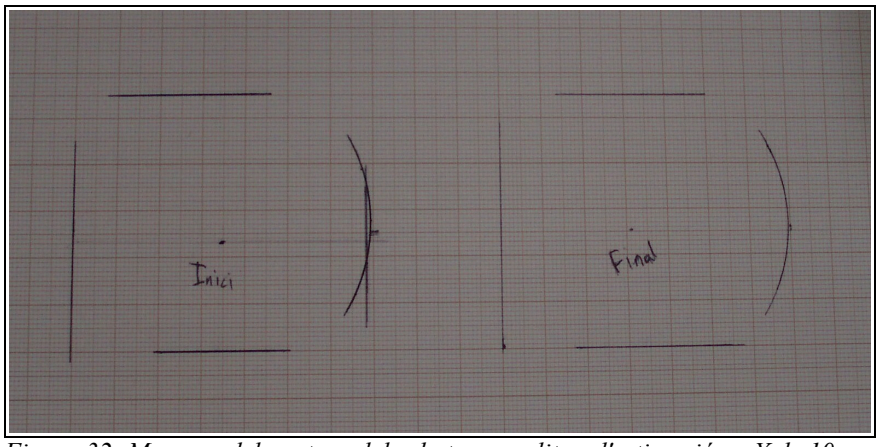


Figura 32: Marques del contorn del robot per realitzar l'estimació en X de 10 cm.

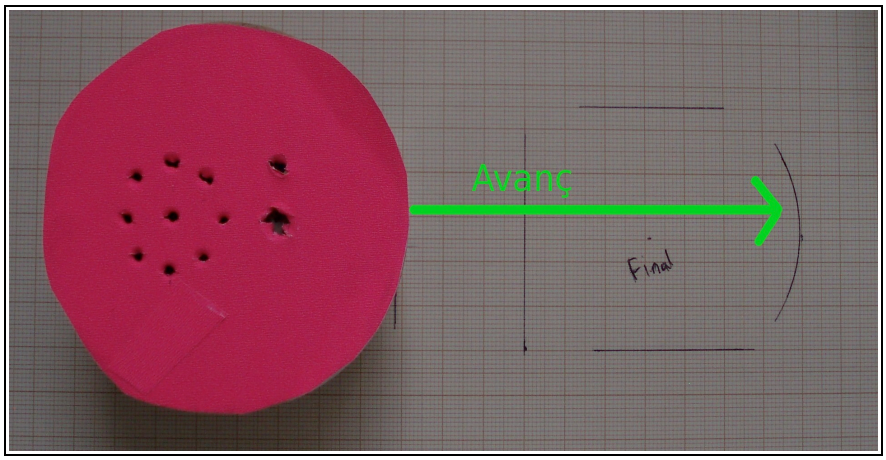
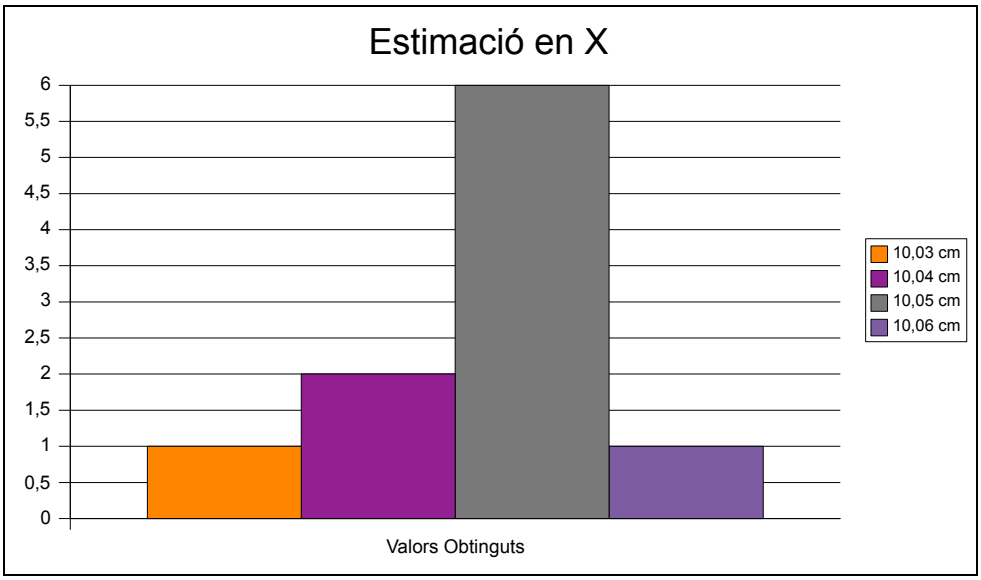


Figura 33: El robot sobre la marca en Inicial per realitza l'estimació en X de 10cm.

A continuació, es poden veure al gràfic 1, els resultat obtinguts al realitzar la prova 10 vegades.



Gràfic 1: Dades en l'estiamció de la constant per l'eix X.

El valor de la cte era 77,63. Ara, a partir dels resultats obtinguts en les 10 proves fetes en recorre 10 cm ,calcularem la nova constant més ajustada.

$$\frac{10}{10{,}044816} = \frac{1/cte}{(1/77{,}63)} \qquad cte = \frac{10{,}044816 \cdot 77{,}63}{10} \quad \text{cte} = 77{,}9779$$

- **Estimació del Yaw**

Per calcular l'estimació del Yaw es controlarà el robot utilitzant un programa que permet teleoperar-lo per tal que giri 90 graus o π/2 radiants sense que hi hagi cap desplaçament en X ni en Y.

En l'entorn de treball es col·loca en el terra un paper en què hi ha un transportador d'angles imprès per tal de poder mesurar l'angle recorregut per el robot, i també es realitzen unes marques per col·locar sempre el robot en la mateixa posició inicial i saber exactament a quina posició s'hauria d'aturar. Podem veure aquest muntatge en les figures 34 i 35.

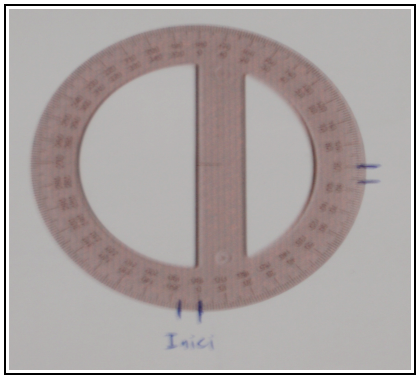


Figura 34: Marques per realitzar l'estimació del Yaw.

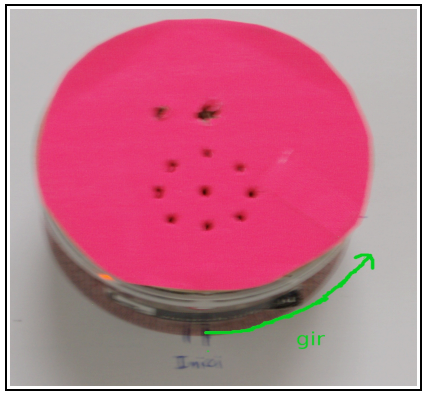
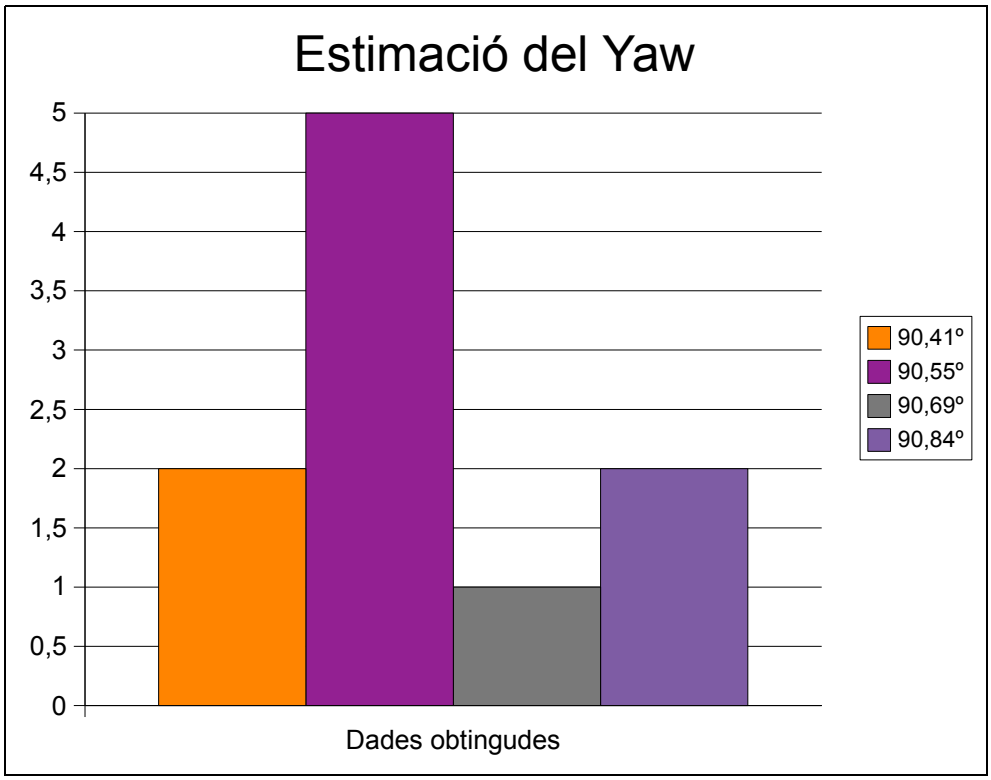


Figura 35: E-puck en la posició d'inici per l'estimació del Yaw.

En el gràfic 2 es pot observar els resultats obtinguts en realitza l'estimació del yaw 10 vegades.



Gràfic 2: Dades de l'estimació de la constant en Yaw.

La constant per l'angle és la distància entre eixos (w) i la podem ajustà a través de les dades obtingudes:

$$\frac{90,3}{90,594} = \frac{1/w}{(1/5,2)} \qquad w = \frac{90,594 \cdot 5,2}{90,3} \qquad \text{w} = 5,2169$$

- **Estimació de la posició en un moviment combinat**

Per a aquesta estimació el que farem serà que el robot es desplaci de manera autònoma sobre el paper mil·limetrat dibuixant un quadrat de 10 cm de costat. Veurem com s'incrementa l'àrea d'incertesa en cada iteració que realitzi el robot, ja que cada cop hi haurà un error acumulat sistemàtic més gran.

Farem que el robot recorri tres cops el quadrant, i compararem la diferència en cada iteració respecte al punt d'origen. Realitzarem aquesta prova amb els valors de les constants sense calibrar i després amb els valors calibrats per comprovar si hi ha hagut millora.

A les figures 36 i 37 tenim dues imatges fetes durant la prova, quan el robot estava entre la 3a cantonada i la 4a cantonada.

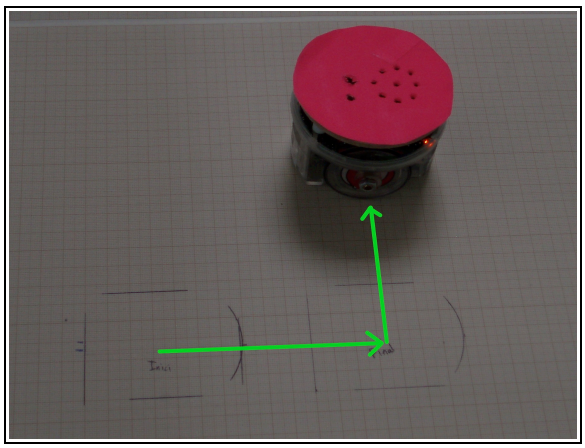


Figura 36: El robot està a la 3a cantonada de l'estimació del moviment combinat.

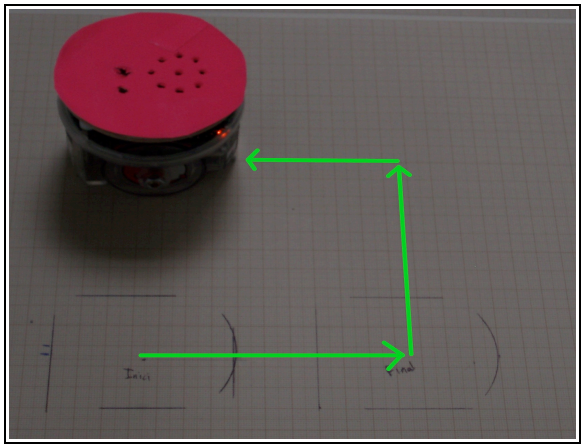
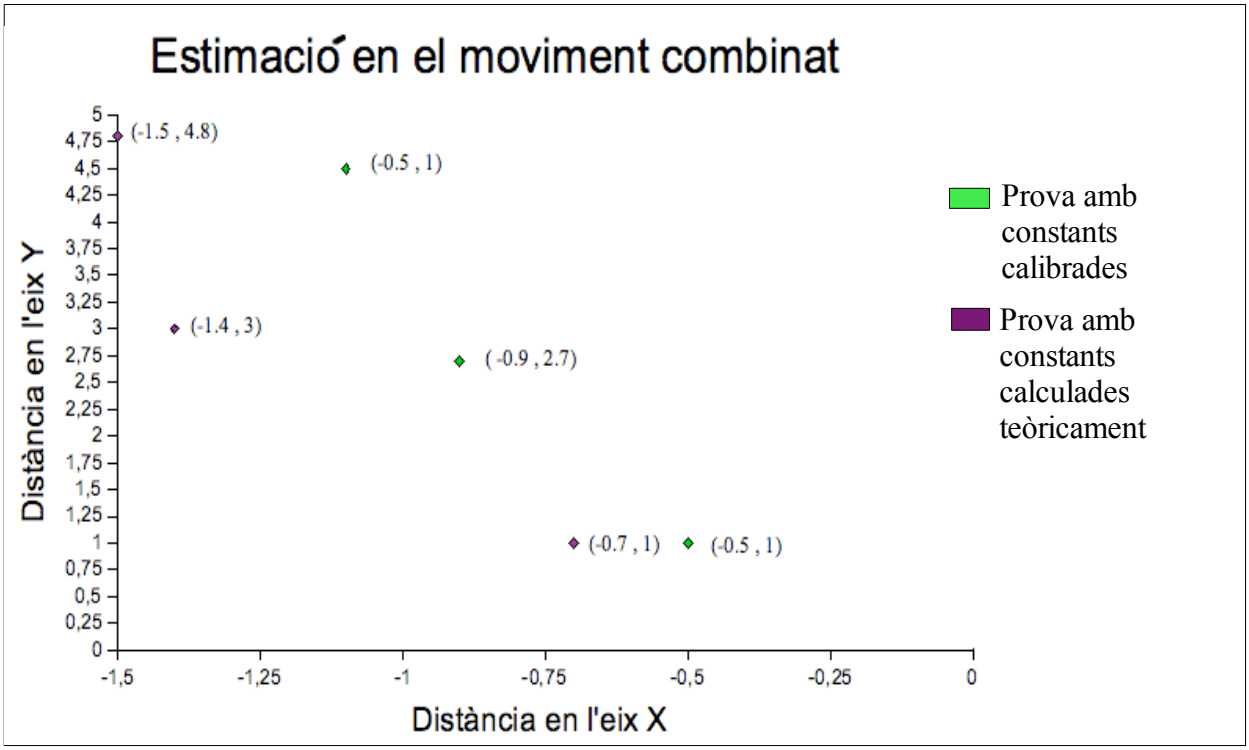


Figura 37: El robot està a la 4a cantonada de l'estimació del moviment combinat.

A continuació, al gràfic 3, es pot observar com a cada iteració la posició ha anat empitjorant. Sobre tot perquè el control del Yaw és el que té una pitjor aproximació a la realitat i fa que no giri exactament 90 graus i cada cop estigui més desplaçat.



Gràfic 3: Resultats en l'estimació en el moviment combinats realitzats amb els dos valors per les constants.

5. SISTEMA DE POSICIONAMENT ABSOLUT

5.1. Introducció

El sistema de posicionament absolut utilitza una càmera situada a sobre l'entorn de proves, una tarja d'adquisició d'imatges, l'entorn matlab i un programa de processament d'imatges mitjançant una LUT.

5.2. Adaptació de l'entorn

És necessari adaptar tot l'entorn de treball perquè la càmera sigui capaç de detectar totes les parts importants que el formen. Per això s'han escollit uns colors molt fàcils de distingir per la càmera, ja que són molt diferents tant entre ells com de l'entorn (veure figura 38). Les parts que necessiten ser adaptades són les cantonades de l'espai de treball, els obstacles i el robot e-puck.

Figura 38: Els tres colors utilitzats per distingir les diferents parts de l'entorn de treball.

5.2.1 Adaptació de l'espai de treball

Per adaptar l'espai de treball hem decidit col·locar unes tires de color verd a cada una de les cantonades. Només s'han col·locat a les cantonades perquè proporcionen prou informació per conèixer els límits de l'espai i també calcular la posició en què es troben els objectes que estan dins aquest límits.

És possible calcular la posició perquè es coneixen les dimensions de l'entorn de treball. D'aquesta manera es poden definir les distàncies entre les diferents cantonades i transformar el valor dels píxels en cm fàcilment.

Així doncs, l'entorn de treball té unes dimensions de 80 x 60 cm i s'ha decidit col·locar unes marques que ocupen 7 cm per eix, tal i com podem observar en la figura 39.

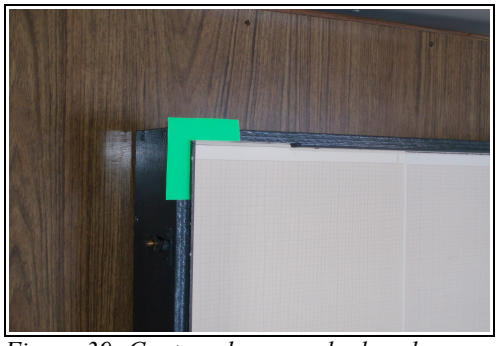


Figura 39: Cantonada marcada de color verd les dues tires.

Una altra modificació que s'ha portat a terme en l'espai de treball consisteix a col·locar paper mil·limetrat sobre tot el terra per tal de facilitar la lectura de posicions i també per evitar que es produïssin reflexes de llum cap a la càmera, ja que la superfície és metàl·lica (veure figura 40).

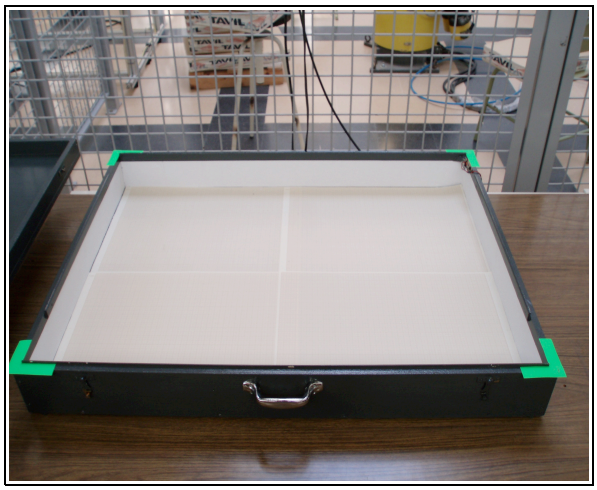


Figura 40: Espai de treball amb una vista complerta.

5.2.2. Obstacles

En l'entorn de treball es disposa d'uns obstacles, que es poden situar en diferents configuracions per tal de poder fer diferents tipus de proves amb el robot e-puck. Aquests obstacles tenen les dimensions de 8,5 x 10 x 7 cm.

Per tal de poder identificar els obstacles en el tractament de la imatge s'ha decidit marcar la part superior dels obstacles amb color groc (veure figura 41).

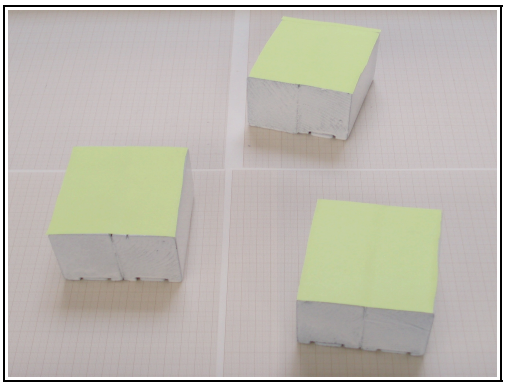


Figura 41: Tres obstacles marcats amb el color groc.

5.2.3. El robot e-puck

Per identificar el robot e-puck, en un primer moment es va intentar utilitzar el mateix color verd de la part superior, ja que era complicat marcar el robot amb un color. En realitzar les proves es va observar que era un color massa fosc i que es confonia amb l'entorn.

La solució a què es va arribar va ser la construcció d'un segon sostre de fusta que es colla al mateix lloc que el circuit imprès. D'aquesta manera, no s'afecta al robot, i es pot intercanviar el sostre entre els diferents robots de què es disposa al laboratori. Des d'aquest segon sostre són accessibles els controls principals del robot.

A la figura 42 podem observar aquest segon sostre ja marcat amb el color corresponent.

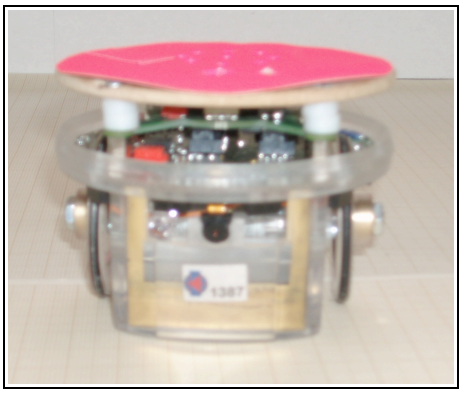


Figura 42: Robot e-puck amb el segon sostre instal·lat i marcat de color.

5.3. Adaptació del software disponible

Per fer el processament de les imatges, es parteix del software desenvolupat per Ricard Batllori en el seu projecte final de carrera. Un cop estudiat s'ha decidit crear un nou script amb una estructura similar a la seva però aplicat al nostre cas.

Les particularitats més importants que cal destacar són:

- Es poden trobar molts més objectes en la imatge.
- Determinats colors en diferent situació en la imatge no seran interpretats de la mateixa manera.
- No és necessari conèixer el centre de gravetat de totes les peces.
- S'han de calcular uns eixos de coordenades i calcular distàncies dins aquest eixos.
- És necessari establir comunicació amb un programa que s'està executant en el webots.

5.4. Connexió amb el webots.

L'script creat portarà a terme el paper de servidor amb un únic client mitjançant un socket. La funció principal del servidor serà la de calcular a quina posició es troba el robot segons el sistema de posicionament absolut i enviar aquest valor al servidor. Si per algun motiu l'script no fos capaç de calcular la posició, enviarà un codi d'error que indica la causa d'aquest error. El programa en el webots valorarà si aquest s'ha d'esperar a solucionar el

problema o pot executar-se sense conèixer la posició en aquell instant.

5.4.1. Estructura de missatges

- Missatges rebuts: Els missatges que es rebran del client seran molt senzills i consistiran en un únic caràcter. Tenim dues opcions: el caràcter 'p' el qual ens està indicant que el programa del webots vol conèixer la posició absoluta i el caràcter 'f' que ens indica que s'ha acabat l'execució del programa del webots i ja es pot deixar d'executar l'sript en el matlab.
- Missatges enviats: L'estructura del missatge que enviarem serà en forma de cadena de caràcters. Aquesta cadena contindrà, en primer lloc, el codi de missatge, que pot anar de -5 a 0. Si l'estat és diferent a 0 només s'enviarà l'estat, ja que significa que no s'ha pogut calcular la posició. Si és 0 l'estructura serà “Estat PosicióX PosicióY” entre cada valor hi haurà un espai en blanc.

La codificació de l'estat en els missatges és la següent:

 0 -> Posicions calculades correctament
-1 -> No es detecta el robot dins l'entorn de treball
-2 -> No es detecta la cantonada inferior dreta
-3 -> No es detecta la cantonada inferior esquerra
-4 -> No es detecta la cantonada superior dreta
-5 -> No es detecta la cantonada superior esquerra

5.5. Disseny de l'algorisme

L'script haurà de seguir uns passos determinats i un ordre per tal de poder complir l'objectiu que, en aquest cas, és enviar la posició absoluta al webots a través del socket.

En iniciar l'script en el matlab, obtindrem una imatge de l'espai de treball en la qual s'ha de seleccionar una mostra de cadascun dels tres colors. L'ordre ha de ser el següent: en primer lloc, el color d'una cantonada, en segon lloc el color del robot, i per últim el color de l'obstacle. Després de conèixer els colors els transformarem a l'espai de color HSV i calcularem la LUT de tres colors utilitzant la funció de Ricard Batllori.

Ara esperarem fins que algú es connecti al socket del servidor. Quan algú es connecti al socket anirem escoltant el port fins rebre algun caràcter, dels que tenen funcionalitat. El caràcter 'f' tancarà el port del socket i finalitzarà l'scprit. El caràcter 'p' retornarà la posició on es troba el robot dins l'espai de treball.

Per calcular la posició inicial, en primer lloc, haurem de buscar el robot i les quatre cantonades. El robot es pot trobar en qualsevol lloc de la imatge; en canvi, les cantonades han d'estar cadascuna en un quadrant diferent de la imatge. Si no és així, es produirà un error. Cada cantonada i el robot han de tenir una àrea mínima, si no la tenen, serà com si no s'haguessin detectat.

Al mateix temps que es recorre tota la imatge buscant les cantonades i el robot, també es marcarà en una altra imatge de les mateixes dimensions si hi ha un color dels que busquem o no. D'aquesta manera en acabar de recórrer tota la imatge original tindrem una altra imatge però segmentada amb els diferents colors trobats.

Al finalitzar es comprovarà si hi havia les cantonades i el robot, es calcularà el seu centre de gravetat i es mostrarà la imatge segmentada.

Si no hem tingut cap error, es buscarà la matriu per tal de poder transformar els píxels del sistema de coordenades inicial al sistema de coordenades de l'entorn (veure figura 43). Podem observar que hi haurà un canvi de translació i un de rotació en els nous eixos.

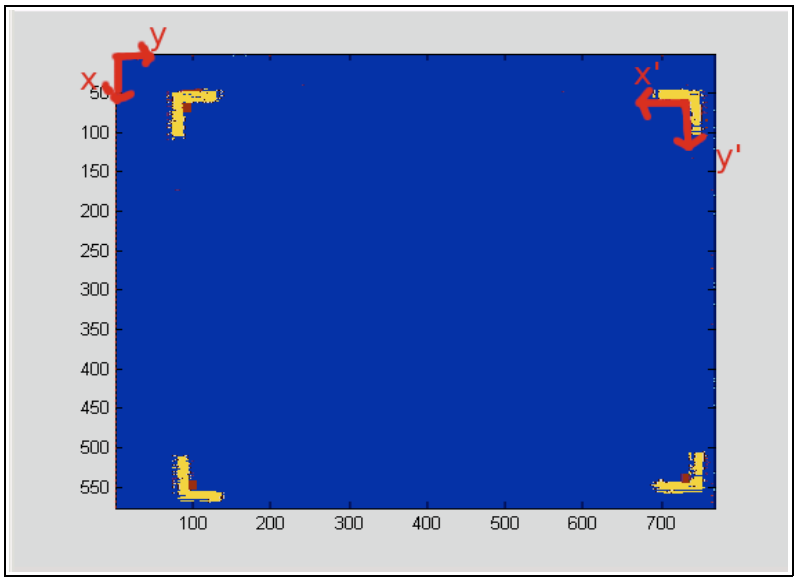


Figura 43: Es poden observar els dos sistemes de coordenades, el del matlab (x y) i el que utilitzarà el webots (x' y').

La matriu de la transformació és la inversa de la translació en X i en Y, per la rotació respecte el Yaw.

angle = atan2((x_Cantonada1-x_Cantonada2) ,(y_Cantonada1-yCantonada2))

On la cantonada1 és la cantonada superior esquerra de la imatge i la cantonada2 és la cantonada superior dreta de la imatge.

$$Matriu\,Translació = \begin{pmatrix} 1 & 0 & 0 & \text{origen de coordendes x'} \\ 0 & 1 & 0 & \text{origen coordenades y'} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$Matriu\ Rotació = \begin{pmatrix} \cos(angle) & -\sin(angle) & 0 & 0 \\ \sin(angle) & \cos(angle) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Matriu Transformació = (Matriu Translació * Matriu Rotació)$^{-1}$

$$Posició = Matriu\,Transformació * \begin{pmatrix} x \\ y \\ 0 \\ 1 \end{pmatrix}$$

Per acabar, s'aplicarà el canvi del sistema de coordenades al centre de gravetat del robot i es transformarà la distància de píxels a centímetres. Aquest càlcul és una senzilla regla de tres, ja que coneixem la distància que hi ha entre totes les cantonades tant en píxels com en cm. Les regles de tres que s'utilitzaran seran les següents:

$$\frac{\text{Posició de la X en cm}}{\text{Posició de la X en píxel}} = \frac{\text{Distància entre les cantonades horitzontals en cm}}{\text{Distància entre les cantonades verticals en píxels}}$$

$$\frac{\text{Posició de la Y en cm}}{\text{Posició de la Y en píxel}} = \frac{\text{Distància entre les cantonades verticals en cm}}{\text{Distància entre les cantonades verticals en píxels}}$$

Un cop enviat es tornarà a escoltar el port.

El resultat d'aquest script és el que podem observar a la figura 44. El resultats enviats pel socket es guarden en un fitxer des del programa del webots i, després, es poden representar les trajectòries tal com es mostra a la figura 45.

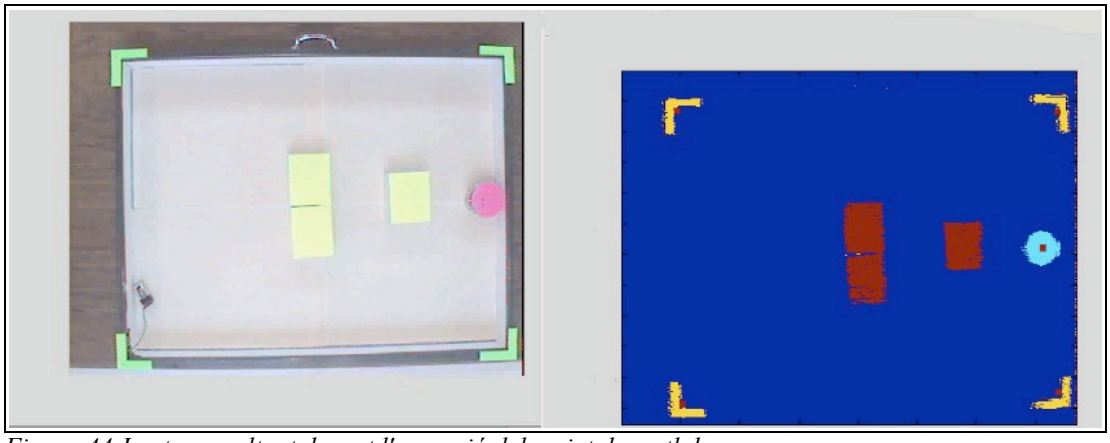


Figura 44:Imatge resultant durant l'execució del script de matlab

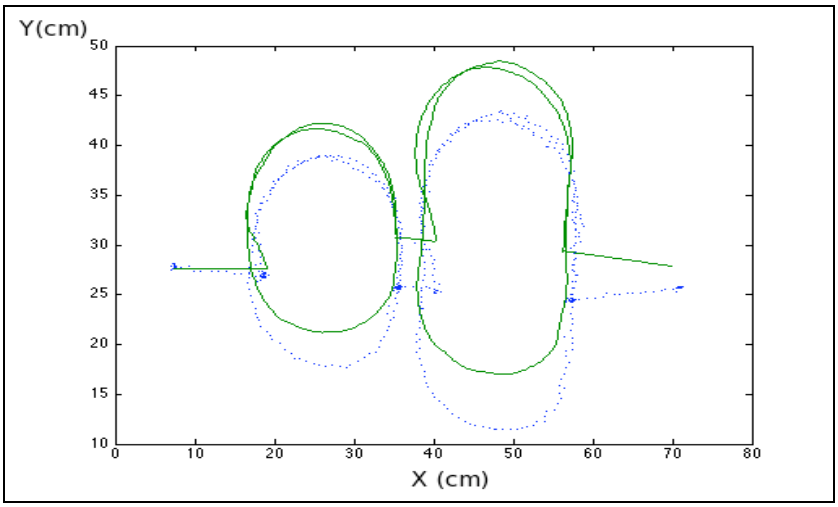


Figura 45: Exemple on es veuen les dues trajectòries: amb verd la descrita per l'odometria i la línia blava discontinua la posició absoluta calculada per el matlab.

6. CONTROL DE LA TRAJECTÒRIA DEL ROBOT E-PUCK

6.1. Introducció

El robot e-puck no disposa d'un planificador de trajectòries sinó que utilitza una estratègia d'accions reactives, és a dir, realitza la seva trajectòria a mesura que va troban els obstacles. D'aquesta manera no és necessari tenir un coneixement global, només amb un coneixement local podrem anar allà on vulguem.

El control de trajectòries està basat en diferents mètodes d'actuació, com són orientar-se cap a una posició, anar en línia recta mantenint un rumb i seguir una paret. Després la combinació d'aquests diferents mètodes d'actuació fa que es puguin aplicar tant l'algorisme Bug 1 com el Bug 2.

Aquest algorismes tenen una freqüència de control de 10 Hz. Així doncs cada 100 ms s'obtenen les dades dels sensors i es calculen els valors de l'odometria.

6.1.1. Connexió amb el matlab

En iniciar el programa pel robot e-puck, s'iniciarà també una connexió amb el matlab per poder conèixer la posició absoluta calculada per l'script. Si no es pot establir la connexió amb el matlab, el programa no s'iniciarà, ja que és necessari conèixer la posició inicial del robot.

Durant l'execució del programa s'anirà preguntant en quina posició es troba el robot igual que l'odometria, tot i que aquest cop si falla la comunicació o el matlab retorna algun error, el programa continuarà executant-se, tot i que no guardarà la posició correcta al fitxer i guardarà el codi del missatge d'error rebut per poder-lo interpretar posteriorment.

Un cop assolida la posició objectiu s'enviarà un senyal perquè l'script del matlab deixi d'executar-se.

Els missatges que s'enviaran des del webots només seran un caràcter: la 'p' que significa que es vol conèixer la posició actual i la 'f' per donar la ordre d'apagar l'script.

6.2. Orientació envers l'objectiu i avanç cap a ell

Per aconseguir orientar el robot en vers l'objectiu, és necessari conèixer diferents valors previs. Aquest valors són la posició i orientació inicial (Yaw) i la posició objectiu.

La posició inicial i la posició objectiu són valors que obtenim a través del posicionament absolut (a través del matlab) i de l'usuari. En canvi, l'orientació del robot no la podem obtenir de manera externa. Per tant, s'ha decidit que l'angle inicial sempre serà 0, és a dir, el robot en la posició inicial sempre ha d'estar orientat de manera que el seu avanç sigui paral·lel a l'eix X del sistema de coordenades.

6.2.1. Disseny

Abans de fer el disseny s'ha de diferenciar entre dues parts: la part en què s'orienta el robot envers l'objectiu i la part en què s'avança cap a l'objectiu intentant mantenir la direcció.

Per dur a terme la primera part s'ha de calcular l'angle en què ens hem d'orientar per tal d'avançar en línia recta cap a la posició objectiu. Per obtenir aquest angle utilitzarem l'atan2, ja que ens donarà l'angle que busquem i el quadrant en què es troba. Un cop calculat, girarem pel costat que sigui més ràpid per assolir la posició objectiu. Després d'aquest pas, el robot ja podrà continuar.

En la segona part, hem de fer que les dues rodes avancin a la mateixa velocitat, per anar cap a la posició objectiu. Tot i això, al mateix temps, també hem de controlar que no arribem a la posició objectiu i comprovar que l'orientació sigui la correcta. Si no és així, hem de fer que una de les rodes avanci més ràpid que l'altra per tal de compensar la desviació sense haver d'aturar el robot per tornar-lo a orientar.

A la figura 46, tenim un diagrama en el qual es pot observar de manera gràfica l'explicació de com funciona el mètode.

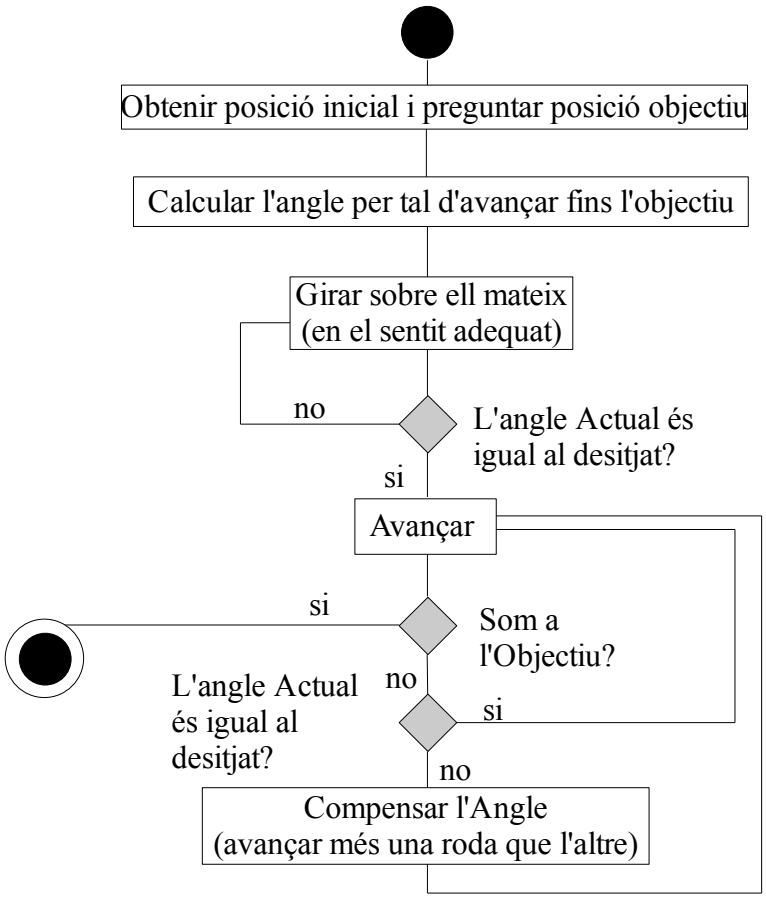


Figura 46: Diagrama de com orientar-se i avançar envers un objectiu.

6.2.2. Resultats

Per comprovar el resultats hem fet l'experiment de fer anar el robot des d'una posició pròxima a (0,0) fins a una posició central (40,30). En acabar l'experiment, podem observar que la posició final del robot segons el posicionament absolut (40,95, 25,95). Veiem com en l'eix X no s'ha desviat massa però l'eix Y té un valor de 5 cm de diferència. A la figura 47, es pot observar la descripció de les dues trajectòries i podem observar que l'error principal apareix en l'orientació del robot. La trajectòria de punts és el posicionament absolut i la continua, el posicionament odmètric.

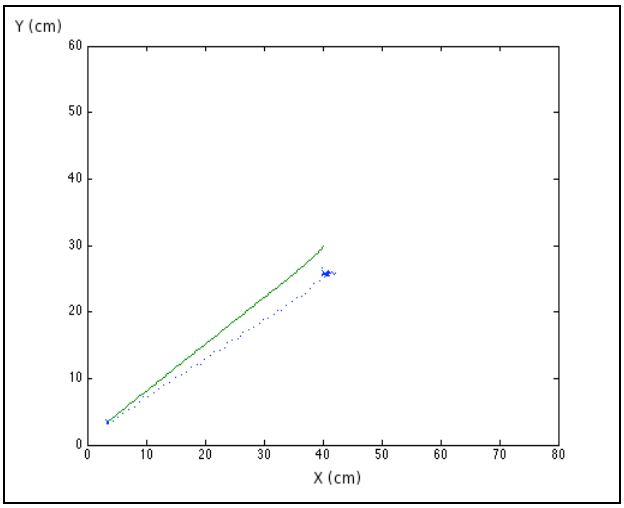


Figura 47: Comparació de les trajectòries per anar fins al centre.

6.3. Seguiment d'obstacle

Per tal de poder seguir un obstacle hem decidit utilitzar només dos sensors, segons per quina banda decidim seguir l'obstacle, es canviaran aquests sensors per els d'una banda o l'altra. Els sensors seran IR1 i IR2 o IR5 i IR6. Podem observar on estan situats a la figura 48.

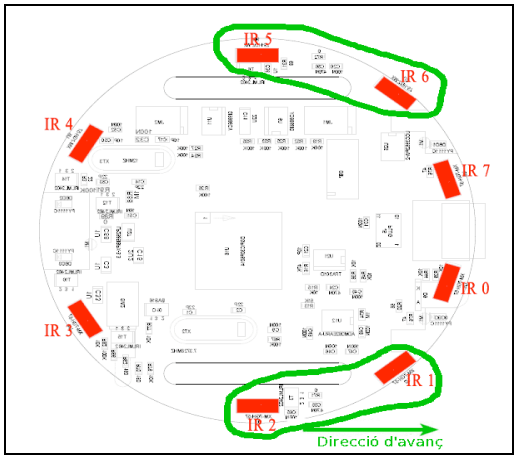


Figura 48: Estan seleccionats les dues parelles de sensors que es poden utilitzar per seguir l'obstacle.

També s'havia considerat l'opció d'utilitzar els sensor IR3 i IR4, però formen un angle massa obert i molts cops no arriben a detectar l'obstacle. Per aquesta raó s'han descartat, ja que causen més problemes que solucions.

6.3.1. Càlculs sobre els sensors de proximitat

Per tal de poder interpretar d'una forma més senzilla els valors que s'obtenen utilitzant els sensor d'infrarojos usarem unes equacions que ens transformaran el valor llegit pel sensor a distància en cm.

Segons el valor que obtinguem del sensor, s'aplicarà una recta que contingui el valor llegit. El conjunt de rectes s'han calculat a partir de les característiques dels sensors i emprant els valors sense soroll en tots els casos, ja que són els valors mitjos. Per cada tram de la funció, utilitzarem aquest tipus de funció:

Recta n (y0 – y1) $m=\frac{(y1-y0)}{(x1-x0)}$ $y1=m*x1+b$

Valor llegit	m	b	equació de la recta
(4095 - 3474)	-1242	4095	$y_0=-1242*x_0+4095$
(3474 - 2211)	-2526	4737	$y_1=-2526*x_1+4737$
(2211 - 676)	-1535	4605	$y_2=-1535*x_2+4605$
(676 - 306)	-370	1416	$y_3=-370*x_3+1416$
(306 - 164)	-142	732	$y_4=-142*x_4+732$
(164 - 90)	-74	460	$y_5=-74*x_5+460$
(90 - 56)	-34	260	$y_6=-34*x_6+260$
(56 - 34)	-22	188	$y_7=-22*x_7+188$

6.3.2. Disseny

Per poder seguir un obstacle que pugui tenir diferents formes es tenen en consideració diferents valors que llegeixen els sensors i s'interpreten de maneres diferents.

Es tindran en compte dues dades importants que són la distància a què tenim l'obstacle segons el sensor IR2 o IR5 i l'angle que formen els dos punts on hem detectat l'obstacle. L'ordre amb què s'interpreten aquests valors és, en primer lloc, les distàncies que tenim fins a l'obstacle i, després, segons l'angle que formen els valors detectats pels dos sensors.

Un altre factor que també hem de vigilar és el fet que, a vegades, podem tenir un obstacle just al davant i, si només es mira el lateral, el robot hi xocarà i no podrà evitar l'obstacle. Per aquesta raó també hem de consultar els dos sensors frontals (IR0 i IR7).

A continuació, tenim un conjunt d'exemples (veure figures 49 a 54) de possibles situacions

que pot trobar el robot i com ho detecten els sensors.

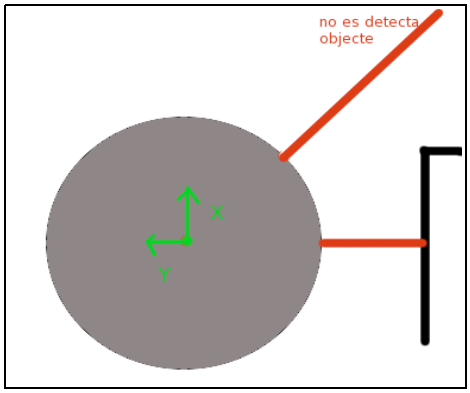


Figura 49: El robot es troba en una cantonada. Això dona un valor normal de distància lateral però un angle molt gran positiu.

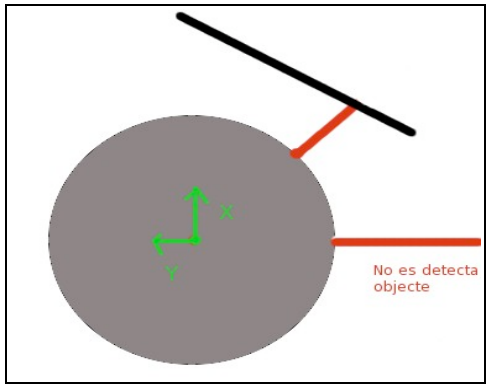


Figura 50: El robot troba una paret al davant però no te cap paret a prop pel costat per tant detecta un angle molt gran negatiu.

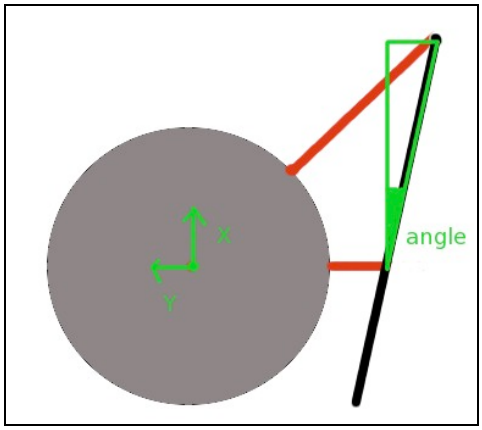


Figura 51: La distància lateral és acceptable i l'angle és positiu.

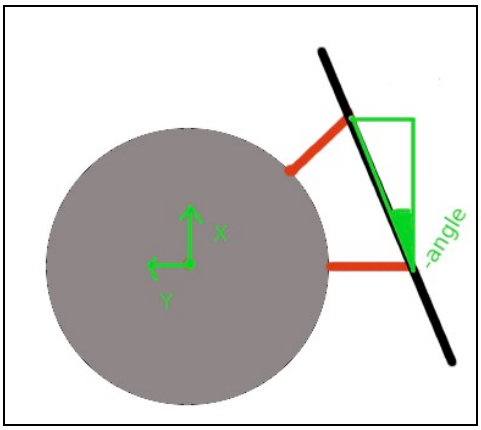


Figura 52: La distància lateral és acceptable i l'angle és negatiu.

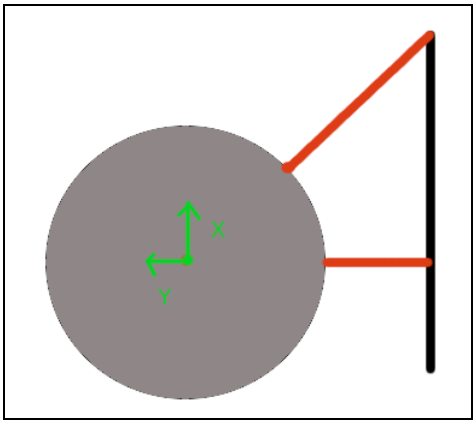


Figura 53: La distància lateral és acceptable i l'angle és pròxim a 0.

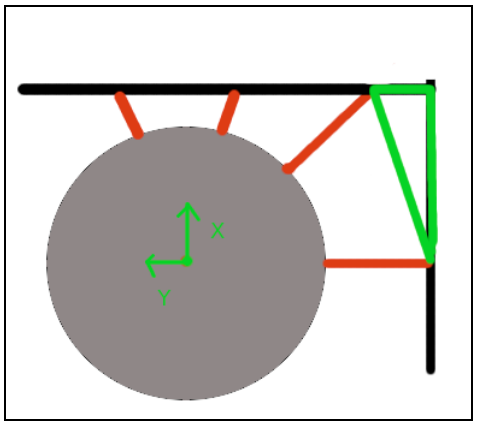


Figura 54: L'angle i la distància lateral són molt semblants a la figura 49 però estem apunt de xocar frontalment.

Després de plantejar tot un conjunt de possibles casos amb què es podria trobar el robot a l'hora d'intentar rodejar diferents tipus d'obstacles, ja és possible definir un model de comportament per tal que el robot sigui capaç de reaccionar davant tots ells i seguir l'obstacle sense tocar-lo en cap moment. El comportament serà el que s'explica a continuació (per explicar com funciona el codi rodejar ho farem utilitzant els sensors de la banda dreta, ja que s'ha decidit utilitzar aquesta banda).

En primer lloc, en detectar que tenim un obstacle al davant hem de col·locar el robot de forma paral·lela a la paret. Per dur a terme aquesta funció, el girarem sobre ell mateix fins que el sensors 1 i 2 detectin que estan a una distància en l'eix Y igual o aproximadament igual.

Un cop estiguem situats en paral·lel, començarem a avançar seguint la paret. Per avançar seguint la paret, el primer que es comprovarà cada cop serà quines són les distàncies.

La distància frontal no pot detectar cap objecte davant seu, si és així ens posarem amb paral·lel a la nova paret de l'obstacle i després tornarem a començar les comprovacions.

Segons quin valor tinguem en la distància lateral avaluarem de manera diferent l'angle que formen els dos punts on es detecta l'obstacle pel lateral. Quan el robot es troba a una distància de la paret molt reduïda girarà sempre cap a fora de la paret sense importar el valor de l'angle. Quan el valor de la distància lateral estigui dins d'un rang acceptable, s'avaluarà l'angle de la següent manera. Si l'angle que formen els dos punts és positiu, girarem en direcció a la paret. Si és pròxim a 0, anirem en línia recta i, si és negatiu, anirem en fora de la paret. El comportament del robot sempre tindrà una tendència més gran a girar cap a la direcció oposada a l'obstacle per evitar la paret. Quan el valor de la distància lateral és massa gran, el robot ha de girar en direcció a la paret.

A la figura 55 podem observar en un diagrama l'explicació del funcionament del mètode de rodejar obstacles.

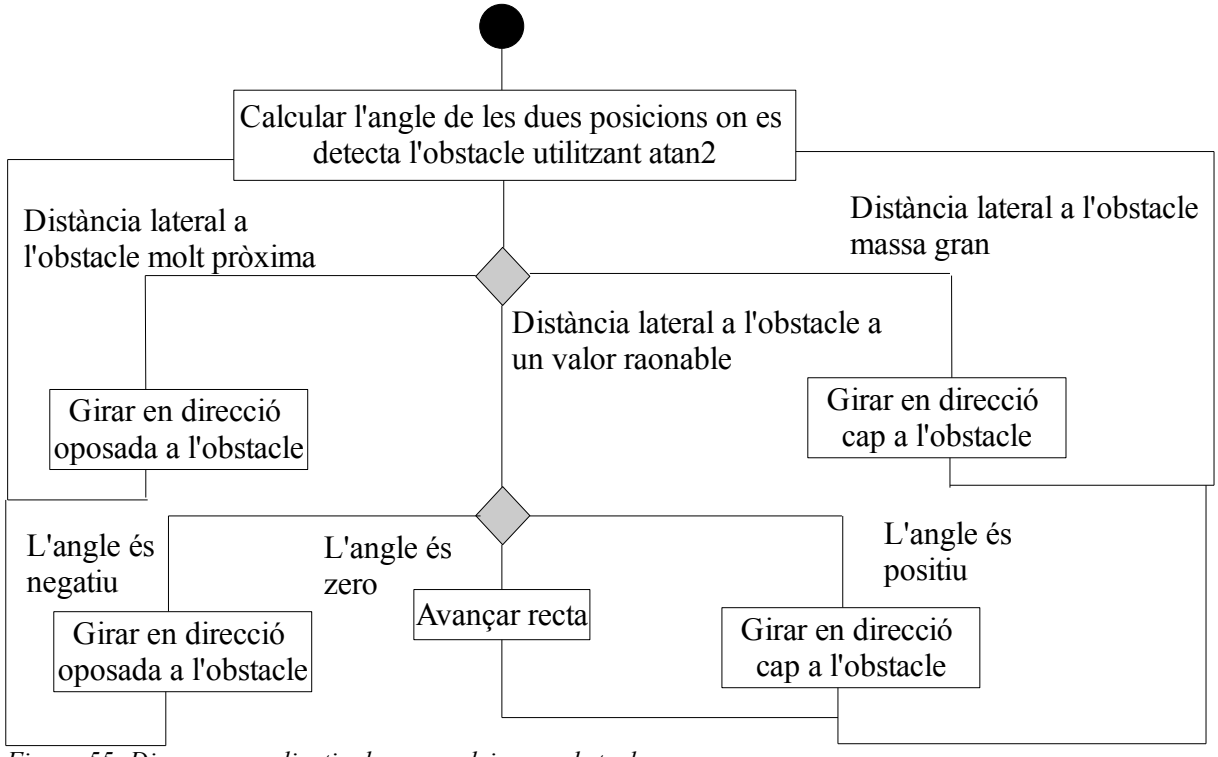


Figura 55: Diagrama explicatiu de com rodejar un obstacle.

Aquesta funció, tal i com s'ha definit, giraria sempre més al voltant de l'obstacle. Per aquest motiu haurem de fer una avaluació de la posició on es troba el robot cada cop abans de començar a rodejar l'obstacle. Si volem donar una volta complerta hem de definir una àrea de proximitat al punt d'entrada. En primer lloc, valorarem si hem sortit de l'àrea al voltant de la posició. Després de sortir d'aquesta àrea, continuarem voltant fins que tornem a entrar dins d'ella.

6.3.3. Observacions

S'ha de tenir en consideració que tots els valors que s'utilitzen per detectar la proximitat del robot amb l'obstacle depenen de molts factors externs, com són el tipus d'il·luminació de què es disposa en l'entorn i del color de l'obstacle (ja que hi ha colors que absorbeixen més la llum que d'altres). Per aquesta raó, s'intenta mantenir l'entorn de treball amb una llum sempre semblant utilitzant un focus que hi ha a sobre de l'espai de treball i evitant que la llum del sol incideixi directament sobre l'entorn.

Per tot això és possible considerar realitzar una calibració exhaustiva prèvia dels sensors o una altra opció seria que el robot, abans de començar a treballar, fes una autocalibració per variar els valors segons les condicions de l'entorn. Aquestes solucions són complexes i s'ha preferit escollir uns valors el màxim robustos possibles. Tot i això, a vegades no funcionen tal com s'esperaria.

6.3.4. Resultats

Per tal de provar el mètode de rodejar obstacles s'ha decidit que el robot només ha de donar una volta entorn de l'obstacle i acabar al punt on s'ha trobat l'obstacle. L'àrea de proximitat que definim al voltant de la posició és de 2 cm, aquesta distància és tan gran perquè al rodejar l'obstacle se sol fer amb una trajectòria força irregular i, de vegades, el robot se separarà considerablement de l'obstacle.

La posició on trobem l'obstacle és (13.46 , 28.56), per tant, l'àrea és de 2 cm al voltant d'aquest punt. El moment que tornem a trobar l'àrea és (11.68 , 27.03) segons l'odometria i segons el sistema de posició absolut és (11.83 , 24,74). A la figura 56 podem observar les dues trajectòries: la línia de l'odometria és continua i verda, i la línia de la trajectòria segons la posició absoluta és blava i discontinua.

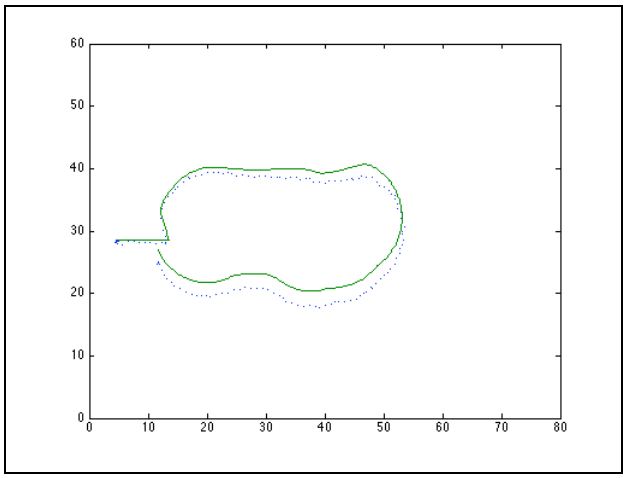


Figura 56: Les dues trajectòries descrites per rodejar un obstacle de 30 x 8 cm.

6.4. Algorisme Bug 1

Per poder aplicar el bug 1 s'adaptaran els mètodes explicats anteriorment (com són l'orientació, l'avanç cap a l'objectiu i el rodejar obstacles) per tal que es pugui realitzar tot l'algorisme del bug 1 i que aquest pugui evitar tots els obstacles que siguin necessaris fins l'objectiu. Per això els mètodes previs es convertiran en part d'aquest algorisme.

L'Algorisme bug 1 explicat a la secció 3.3.2. consisteix a rodejar completament els obstacles i, un cop rodejats, decidir quina és la posició més pròxima a l'objectiu per abandonar l'obstacle.

S'ha escollit rodejar sempre l'obstacle deixant-lo a la dreta, tant quan el robot e-puck realitza la volta complerta a tot l'obstacle com quan torna a la posició més pròxima a la posició objectiu. Això es deu al fet que s'han optimitzat molt els valors d'adquisició utilitzats per la banda dreta del robot.

6.4.1 Disseny

El disseny constarà de diferents parts.

En iniciar l'algorisme haurem d'utilitzar el mètode d'orientació i avanç cap a l'objectiu, ja que podria ser que no ens trobéssim cap tipus d'obstacle per anar a la posició objectiu. Per tal de saber si hi ha algun obstacle que impedeix el pas abans d'avançar sempre s'haurà de comprovar que els sensors frontals (IR 0, IR 1, IR 7, IR 6) no detectin cap obstacle pròxim.

Si es detecta un obstacle per poder aplicar correctament l'algorisme de rodejar obstacles, s'ha de col·locar el robot en una posició paral·lela a l'obstacle tal com s'ha explicat al mètode de rodejar obstacles, ja que és una condició necessària per poder rodejar correctament l'obstacle.

El mètode per rodejar obstacles s'ha modificat per tal que sigui capaç de recordar quina és la posició més pròxima al objectiu i perquè sigui capaç d'aturar-se tant en la posició inicial com en la posició de l'obstacle més pròxima a l'objectiu.

Un cop el robot és a la posició d'entrada de l'obstacle, s'ha de fixar la posició de sortida de l'obstacle i anar fins aquella posició. Per anar fins a la posició, s'utilitza el mateix algorisme que per rodejar-lo.

Per acabar, es torna a iniciar tot el procés d'orientació del robot cap a la posició desitjada i avança fins a ell evitant els possibles obstacles.

A continuació, a la figura 57, es presenta un diagrama on podem observar amb més claredat el disseny de l'algorisme Bug1 aplicat a l'e-puck.

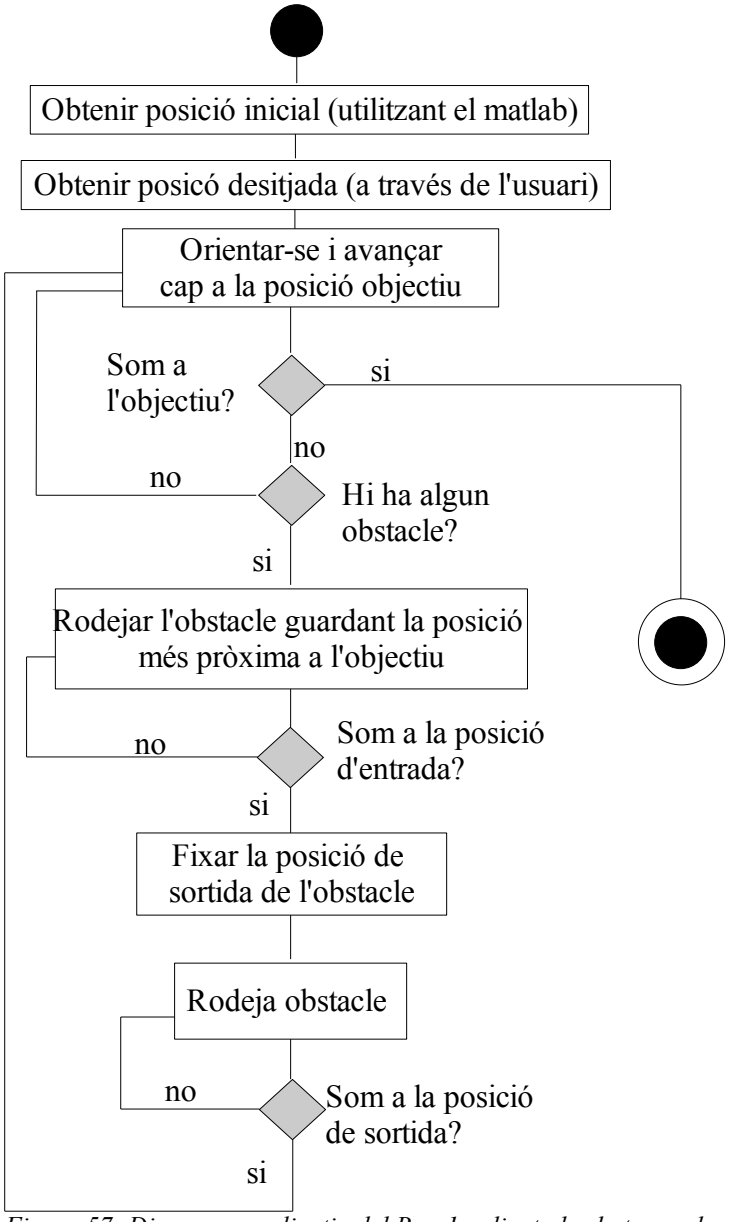


Figura 57: Diagrama explicatiu del Bug 1 aplicat al robot e-puck.

6.4.2. Observacions

La part en què s'ha hagut de vigilar més ha estat el fet de detectar si som en la posició d'entrada o sortida de l'obstacle, ja que com més distància ha recorregut el robot més difícil és acostar-nos a l'àrea de la posició. Com més grans siguin els obstacles, més difícil serà arribar a la posició correcta. Per tant, s'ha de trobar una àrea que no sigui molt petita i no es pugui trobar mai, però si és massa gran s'agafarà una posició cada cop més incorrecta.

6.4.3. Resultats

Per provar aquest algorisme es mostraran dos exemples diferents en què els obstacles estan col·locats en diferents configuracions, també tindrem diferents posicions d'origen pel

robot per tal de comprovar que el posicionament inicial funciona correctament.

En la primera prova l'obstacle té forma de L (format per tres obstacles) i el robot està col·locat inicialment en una posició pròxima a (0,0) (veure figura 58) i l'usuari demana la posició (40,50). Es pot comprovar com el robot ha anat cap a la posició objectiu, ha detectat l'obstacle, l'ha rodejat, ha anat a la posició més propera a l'objectiu i ha assolit aquesta posició. Quant al posicionament, la posició final assolida segons el sistema de posicionament absolut ha sigut (49.27 , 41,72). Aquest error és degut a l'error acumulat de l'odometria en un objecte complex com l'utilitzat. A continuació, a la figura 59 podem veure la representació de les dues trajectòries: l'odomètrica és continua i la posició absoluta és la discontinua.

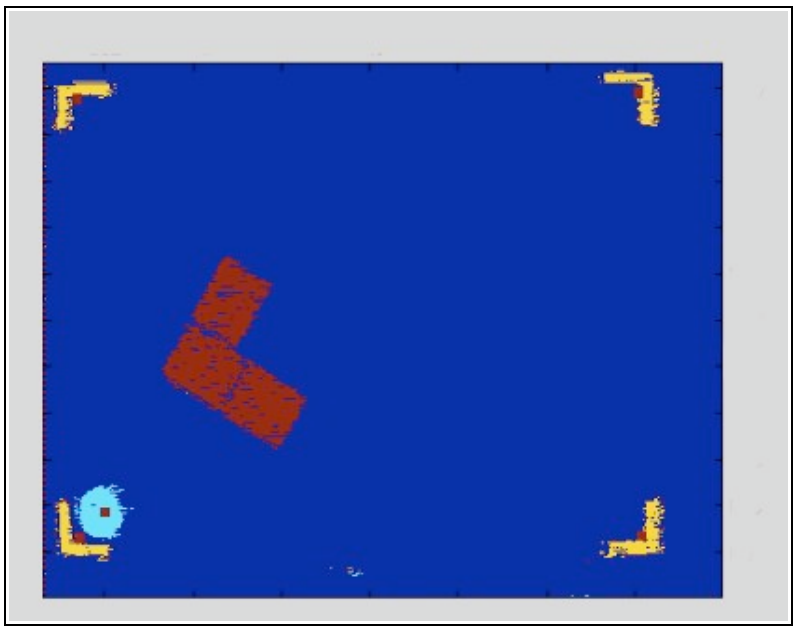


Figura 58: Experiment utilitzant el bug 1 amb l'obstacle en L i amb el robot pròxim a la posició d'origen.

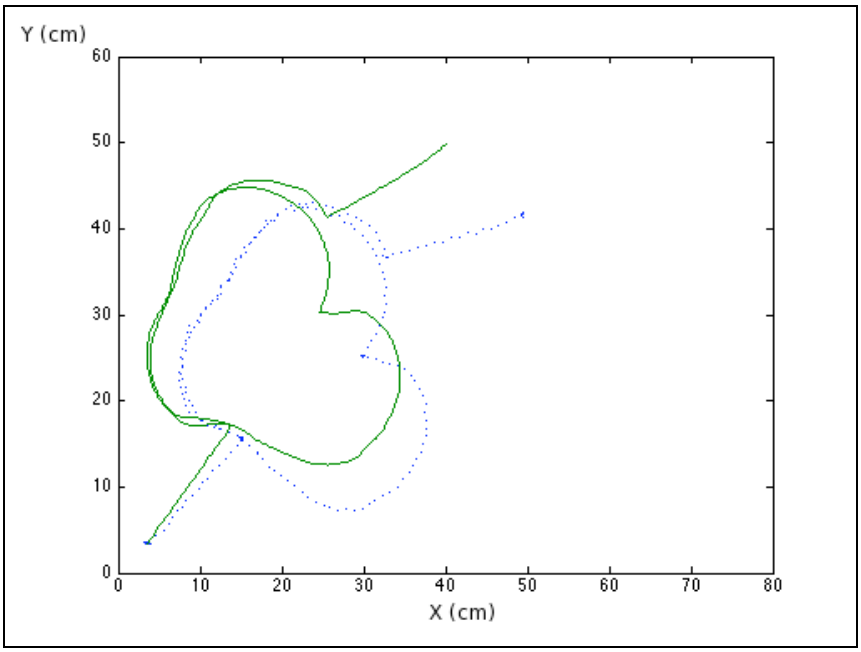


Figura 59: Trajectòries al esquivar obstacle en L utilitzant el bug1.

En la segona prova hi ha dos obstacles diferents: el primer està format per un obstacle i el segon, per dos. Aquest cop el robot es troba aproximadament a la posició (0,30) (veure figura 60). La posició objectiu demanda per l'usuari és (70 , 28). El resultat final és (70.58 , 25.72). Aquest cop el resultat és molt més correcte. Podem veure a la figura 61 les dues trajectòries: la continua és la odomètrica i la discontinua, la absoluta.

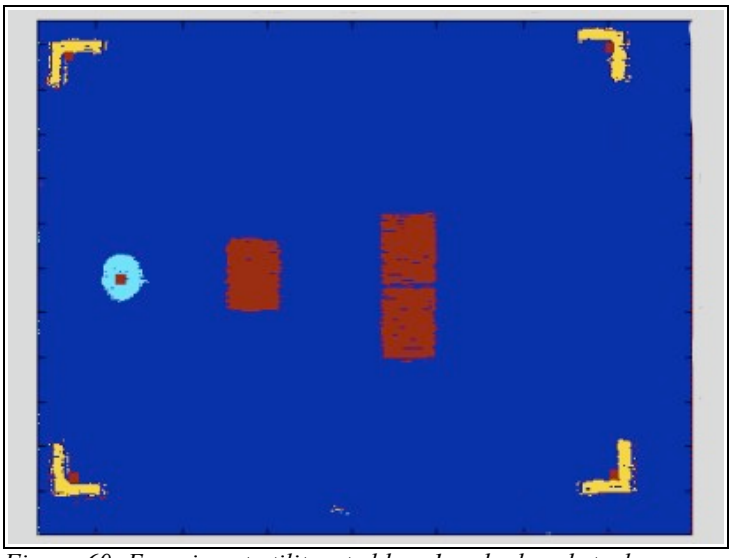


Figura 60: Experiment utilitzant el bug 1 amb dos obstacles.

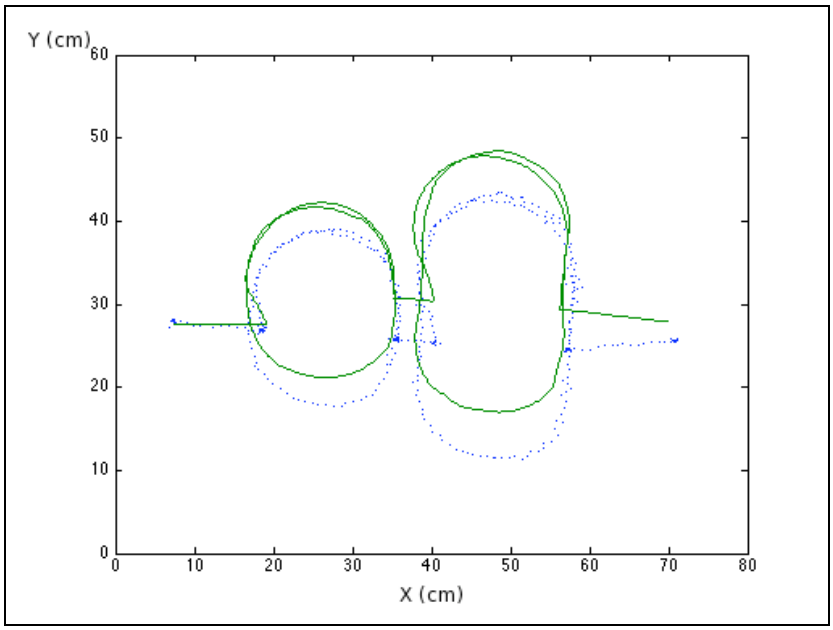


Figura 61: Trajectòries realitzades al esquivar els dos obstacles.

6.5 Algorisme Bug 2

Per poder aplicar el bug 2 s'adaptaran els mètodes anteriors (com són l'orientació, l'avanç cap a l'objectiu i el rodejar obstacles) per tal que es pugui realitzar tot l'algorisme del bug 2 i que aquest pugui evitar tots els obstacles que siguin necessaris fins l'objectiu. Per això, els algorismes previs es convertiran en funcions d'aquest algorisme.

L'algorisme bug 2 explicat a la secció 3.3.3. consisteix, a grans trets, a rodejar l'obstacle fins que es torni a creuar amb la m-line en un punt més pròxim a la posició objectiu.

S'ha escollit rodejar sempre l'obstacle deixant-lo a la dreta.

Per tal de realitzar el bug 2 és necessari utilitzar una recta que va des de la posició d'origen fins a la posició objectiu. Aquest recta rebrà el nom de m-line i no variarà en l'execució de tot l'algorisme.

És necessari recordar uns quants coneixements matemàtics referents a les rectes que són explicats a continuació.

6.5.1. Coneixements matemàtics sobre la recta

Per representa una recta matemàticament s'utilitza la següent equació matemàtica:

$$y = m \cdot x + b$$

On y i x són posicions, b és el desplaçament inicial de la recta i m és el pendent de la recta.

Es calcularà el pendent utilitzant aquesta fórmula que la defineix $m = \frac{(y_f - y_i)}{(x_f - x_i)}$

Es trobarà el valor de la b substituint la m que buscarem prèviament i les posicions inicial o final en lloc de la x o la y. $b = y - m \cdot x$

Un altre informació que hem de ser capaços de conèixer és la distància que hi ha de la recta a un punt, ja que com en el cas del bug1 definirem una zona al voltant de la recta la qual considerarem com si estiguéssim a la recta.

Observant un exemple d'una recta (veure figura 62 i 63) i un punt qualsevol, es pot veure que es forma un triangle rectangle. Utilitzant una formula trigonomètrica podem obtenir la distància que hi ha entre la recta i el punt:

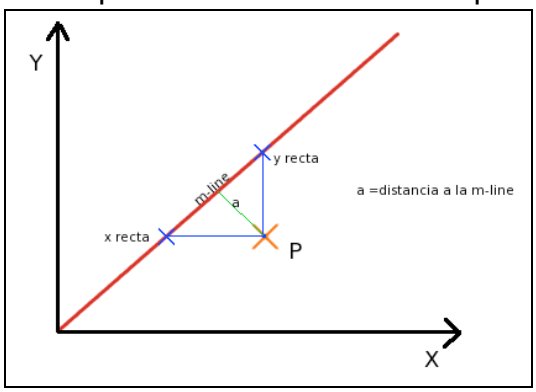


Figura 62: Representació d'una recta amb una b = 0.

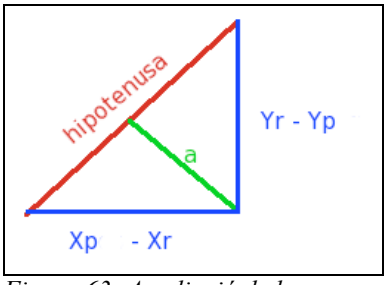


Figura 63: Ampliació de la representació de la figura 62.

$$a = \frac{(y_{recta} - y_{pos}) \cdot (x_{pos} - x_{recta})}{(\sqrt{(y_{recta} - y_{pos})^2 + (x_{pos} - x_{recta})^2})}$$

6.5.2 Disseny

En iniciar el programa utilitzarem una versió modificada del mètode per orientar-se i avançar cap a l'objectiu. Aquesta modificació consistirà a afegir una part que calcula la m-line, tal com hem explicat en l'apartat anterior, i l'angle que forma aquesta recta amb l'eix X. Un cop hem calculat la m-line ja podem començar a avançar en direcció a l'objectiu.

Si ens trobem algun obstacle durant el camí cap a l'objectiu, l'hem de rodejar fins que tornem a trobar la m-line i aquest punt sigui més pròxim que el punt per on hem entrat.

Un cop trobat el punt de la m-line pel qual podem continuar, s'ha de tornar a orientar el robot perquè segueixi de nou la m-line.

A continuació, a la figura 64, podem veure un esquema explicatiu sobre el bug 2 aplicat al robot e-puck.

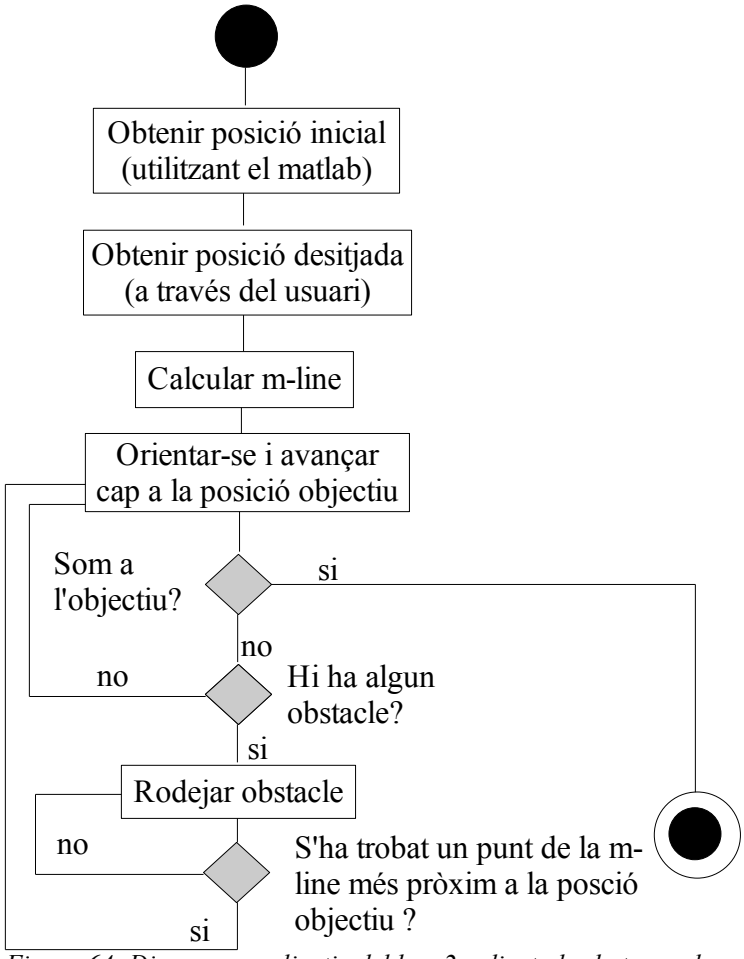


Figura 64: Diagrama explicatiu del bug 2 aplicat al robot e-puck.

6.5.3. Observacions

La part que ha suposat més dificultats en els resultats és el moment de tornar a orientar el robot amb el mateix angle que té la m-line, ja que és molt fàcil que perdi l'orientació amb l'odometria i, després, es perdrà la m-line.

6.5.4 Resultats

gual que en el cas del bug 1 pel qual s'han preparat dos experiments diferents, per al bug 2 usarem els mateixos experiments.

En el primer experiment, l'obstacle té forma de L (format per els tres obstacles) i el robot està col·locat inicialment en una posició pròxima a (0,0) (veure figura 65) i l'usuari demana la posició (40,40). El resultat obtingut pel posicionament absolut ha estat (43.09 , 37.34). Podem considerar que és un resultat correcte. A continuació, en la figura 66, es pot veure la representació de les dues trajectòries: l'odomètrica és la continua i la posició absoluta és la discontinua.

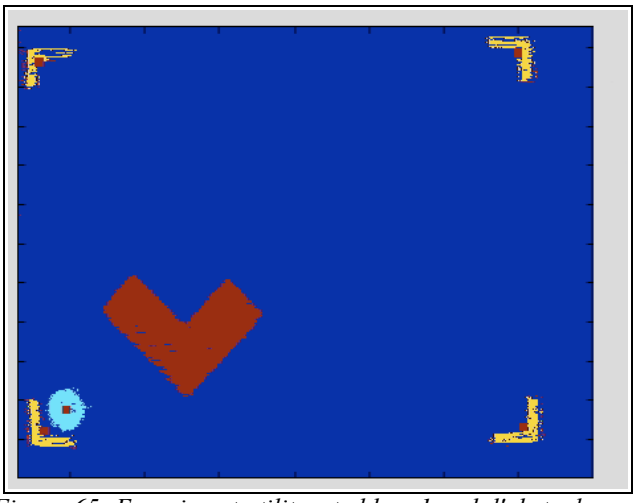


Figura 65: Experiment utilitzant el bug 1 amb l'obstacle en L i amb el robot pròxim a la posició d'origen.

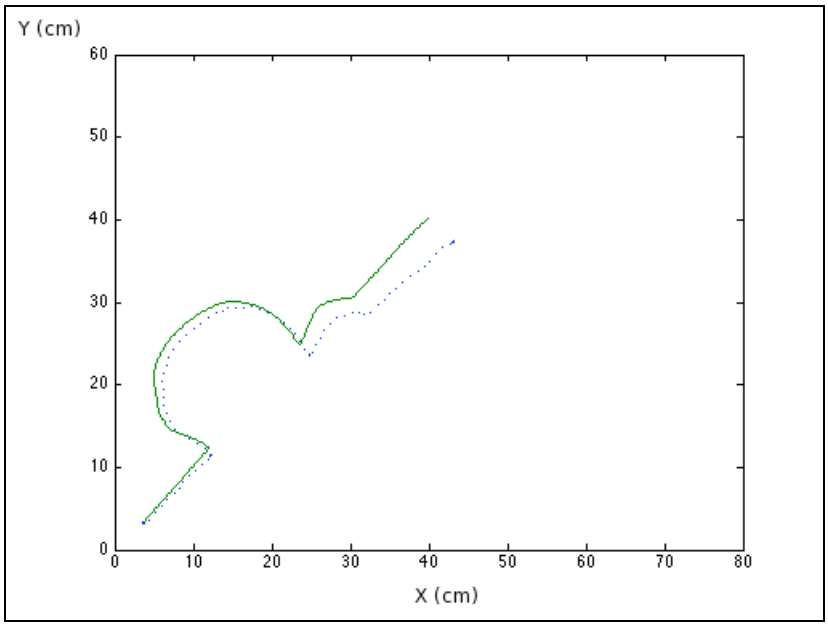


Figura 66: Trajectòries al esquivar l'obstacle en forma L utilitzant el bug2.

En el segon experiment, hi ha dos obstacles diferents: el primer està format per un obstacle i el segon, per dos. Aquest cop el robot es troba aproximadament a la posició (0,30), tal com es es pot veure a la figura 67. La posició objectiu demanada per l'usuari és (50 , 28). El resultat final és (49.60 , 24.80). Així doncs, podem afirmar que aquest cop el resultat és molt correcta. Podem veure, a la figura 68, les dues trajectòries: la continua és l'odomètrica i la discontinua, la absoluta.

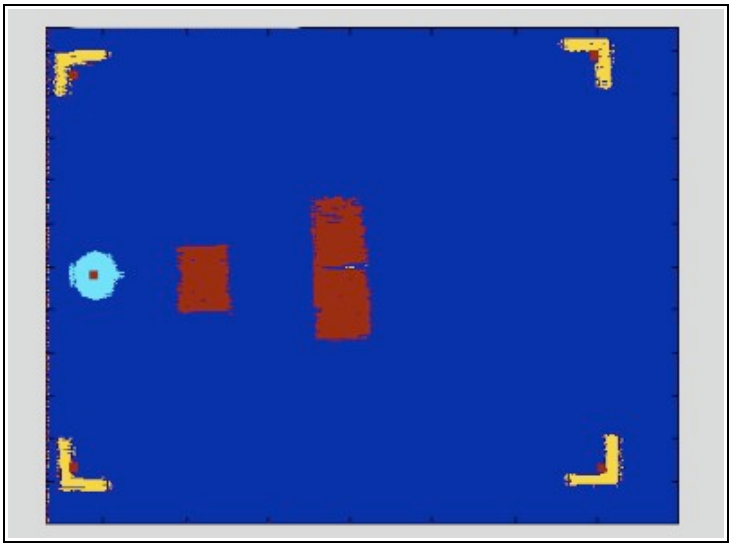


Figura 67: Experiment utilitzant el bug 2 amb dos obstacles.

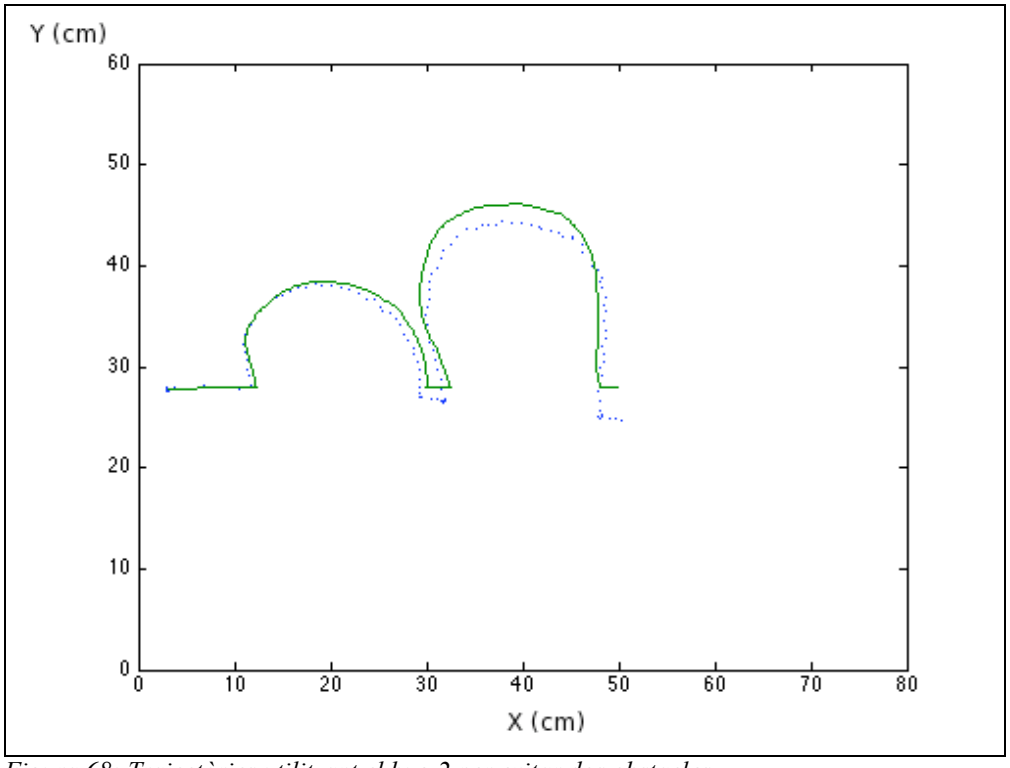


Figura 68: Trajectòries utilitzant el bug 2 per evitar dos obstacles.

7. CONCLUSIONS I TREBALLS FUTURS

7.1. Valoració global

En el transcurs d'aquest projecte s'han adquirit coneixements d'una gran varietat de camps relacionats amb la informàtica.

La part que ha tingut un pes més important ha estat el fet d'aprendre a utilitzar el simulador webots per controlar el robot e-puck. El llenguatge amb què s'ha treballat ha estat el C, però amb diferents particularitats. Una d'elles són les llibreries que proporciona el simulador webots per tal que es pugui controlar tot el conjunt de sensors i actuadors que hi ha als robots. Una altra és el fet d'implementar els algorismes seguint una estructura definida que s'executa d'una manera específica a través del simulador webots. La última particularitat és la de poder canviar paràmetres referents a les característiques del robot que afecten les funcions del simulador.

En relació amb aquesta darrera part, també s'han adquirit coneixements sobre robòtica mòbil autònoma, com són l'odometria i el control de trajectòries. El més important ha estat el fet de conèixer com poder aplicar aquests coneixements a un robot real. Poder observar i intentar solucionar els errors causats per diferents motius: especificacions errònies en els càlculs, variacions de l'entorn, comportaments diferents als esperats... Gràcies a totes aquestes dificultats s'ha pres consciència de la importància d'utilitzar una bona calibració i també de la importància de tenir un entorn controlat. Aquest dos factors ajuden molts cops a la prevenció d'errors.

La part referent a la visió no s'ha treballat tant a fons com la part referent a la robòtica, ja que no ha estat necessari, perquè es disposava del treball previ d'un altre estudiant. Tot i això, s'ha entès el funcionament del codi, com ara el canvi de models de colors i la segmentació mitjançant una LUT.

Aquest fet també ens ha obligat a utilitzar un altre entorn de programació que és el matlab. S'usa el matlab perquè és una plataforma que permet treballar d'una manera senzilla amb tractaments d'imatge, entre altres coses.

L'última part que s'ha treballat ha estat el fet de la intercomunicació entre el matlab i el webots que va suposar uns quants problemes degut a la poca documentació en alguns sentits. Tot i això es va decidir utilitzar la comunicació a través de sockets Ethernet per tal de poder enviar qualsevol tipus de dades entre el matlab i el webots.

En resum, en aquest projecte s'ha treballat en els següents àmbits:

- Robòtica: càlcul de l'odometria i control de trajectòries
- Visió: segmentació d'imatge, canvis de model de color
- Comunicació: comunicació utilitzant sockets Ethernet
- Llenguatges de programació: C, Matlab

7.2. Assoliment d'objectius

Primer objectiu: Estudi de la documentació i software proporcionats pels fabricants del robot i de l'entorn Webots.

S'han consultat els dos manuals del programa webots disponibles tant en format electrònic la web com en paper. També s'han consultat dubtes més específics amb altres usuaris en el mail list. Per estudiar les característiques del robot e-puck, ho hem fet a partir de les característiques descrites en el webots i també des de la pàgina web del fabricant.

Segon objectiu: Programació del software de l'odometria i realització de proves per comprovar-ne la precisió.

S'ha creat una funció capaç de calcular la posició a partir de l'odometria. S'han realitzat proves en línia recta i en angle per tal de calibrar correctament les constants que s'utilitzen en l'odometria. Després, s'ha comprovat la seva eficàcia en moviment combinat.

Tercer objectiu: Disseny, programació i verificació del software dels algoritmes de planificació de trajectòries. Realització d'experiments per a comprovar-ne el funcionament.

S'han dissenyat i implementat dos algorismes de control de trajectòria com són el bug1 i el bug2. S'han realitzat diferents experiments amb diferents configuracions per comprovar la seva eficàcia en múltiples situacions.

Quart objectiu: Disseny, programació i verificació d'un sistema de visió artificial que permeti conèixer la posició absoluta del robot en l'entorn.

S'ha creat un script en matlab que permet segmentar la imatge identificant els límits de l'espai de treball, el robot i els obstacles. Amb aquesta informació es calcula la posició absoluta del robot.

Així doncs, d'aquesta manera podem dir que s'ha assolit la totalitat dels objectius.

7.3. Treballs futurs

A partir de la base que s'ha creat en aquest projecte, es poden fer diferents avanços sobre el mateix tema.

En primer lloc, es pot crear un sistema de calibració automàtic que cada cop que s'inici comprovi que la calibració de l'odometria i la calibració dels sensors de proximitat funcionen correctament. D'aquesta manera, es deixarà de tenir dependència d'un entorn controlat, ja que es podrà adaptar a qualsevol entorn de treball.

En segon lloc, es pot millorar el mètode utilitzat per tal d'evitar obstacles, ja que només s'ha provat amb un tipus d'obstacles concrets i, en el món real, hi ha molts altres tipus d'objectes possibles.

En tercer lloc, es pot millorar el sistema de posicionament absolut, utilitzant un altre sistema per segmentació d'imatges. I també es podria millorar el sistema de visió perquè fos capaç d'identificar l'orientació que té el robot i, d'aquesta manera, el codi del robot pogués substituir la posició odomètrica per la posició absoluta.

En quart lloc i, aprofitant l'increment del nombre de robots disponibles en el laboratori, es podria realitzar algun tipus de control per tal que fossin capaços de treballar en equip o interaccionar entre ells mitjançant els micròfons i l'altaveu o mitjançant el conjunt de leds i la càmera.

Per últim, es podria afegir algun altre tipus de sensor o extensions al robot e-puck que permetessin ampliar les possibilitats d'ús del robot. Per exemple hi ha disponibles diferents tipus de càmeres per fer un control basat en visió, sensors de terra per seguir línies...

8. BIBLIOGRAFIA

Llibres consoltats:

Principles of Robot Motion , Autors Choset, Howie; Lynch, Kevin M.; Hutchinson, Seth; Kantor, George; Burgard, Wolfram; Kavraki, Lydia E.; Thrun, Sebastian Editorial The MIT PRess

Manual Reference Cyberbotics

User Guide Cyberbotics

PFC consoltats:

Entorn de programació pel robot Staubli mitjançant Matlab, aplicació a un sistema de recol·lecció de peces basat en visió artificial, Ricard Batllori, Desenvolupat l'any 2008 en la titulació d'Enginyeria Informàtica

Webs consultades:

http://www.e-puck.org/
http://www.cyberbotics.com/
http://www.itoosoft.com/motorolos/odometria/odometria.html
http://optimus.meleeisland.net/links/01_definicion.html
http://www.jonh.net/~jonh/robots/talk/00.html
http://philohome.com/odin/odin.htm
http://www.simreal.com/content/Odometry
http://www.seattlerobotics.org/encoder/200108/using_a_pid.html