4583-1: Sistema d'AGV (vehicles-robot de guiatge automàtic) per a magatzems i plantes de manufactura

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Albert Martínez Genís
i dirigit per
Lluís Ribas Xirgo
Bellaterra, 15 de Juny de 2012

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, ...

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat: .....

Bellaterra, ...de.....de 20.....

Capítol 0. Index

Capítol 1. Introducció

1.1. Problema

S'entén per cost de producció com la suma de tots els elements que es necessiten per produir un producte determinat. És a dir, el cost de la mà d'obra, cost de l'electricitat i agua, i molts d'altres factors. S'arriba a la conclusió doncs, que quant més temps triguem a produir aquest producte, més ens costarà la seva fabricació.

Actualment el temps de producció i tractament s'ha convertit en un dels principals factors que totes les empreses desitgen reduir per poder reduir costos. No obstant, la velocitat humana és determinada i la maquinària actual és prou eficient. Hi ha diverses maneres de reduir costos i una d'elles és la de reduir el temps de transport. Així doncs, una bona solució és automatitzar la logística interna de la planta de producció per a reduir aquest cost de transport intern del material. Per exemple, es disposa d'una màquina A que realitza una operació a una peça en 5 minuts i quan acaba ha d'esperar 3 minuts a que arribi una altra peça. Si s'aconsegueix millorar aquest temps d'espera a 1 minut, en 24 hores ens estalviem 300 minuts, que equivalen a 50 peces al dia.

1.2. Què és un AGV?

Els humans, en comparació amb les màquines, tenen poca capacitat de càrrega i velocitat, el que fa prou clar que les màquines són la solució correcta.

Les cintes transportadores són una opció econòmica i segura, ja que la càrrega és situada a la cinta i transportada directament al punt de destí, però són molt poc flexibles a canvis. Si es desitja canviar de lloc una màquina s'haurà també de canviar la disposició física de les cintes, el qual pot suposar la pèrdua de molt de temps.

Un Automated Guided Vehicle (en endavant AGV) és un robot mòbil capaç de realitzar tasques que impliquen moviment de manera autònoma com per exemple transportar una càrrega d'un lloc a un altre. Un exemple d'AGV es pot observar aquí:

http://www.youtube.com/watch?v=Kw5qVX_3sig

En el vídeo apareix un AGV seguint el recorregut d'una cinta al terra transportant la càrrega d'un lloc a un altre.

Com són totalment autònoms i mòbils, els AGV han de conèixer el seu entorn, i per això l'han de tenir descrit a memòria. Aquesta descripció és coneguda com a “mapa”. Aquest fet els hi dona la flexibilitat a canvis que les cintes no tenen. Per a realitzar alguna modificació d'alguna localització només caldrà modificar el que es conegut com a “mapa” i elements de guiatge (la cinta que segueix, elements d'identificació, etc).

Aquesta característica els fan la solució idònia en magatzems, plantes de manufactura, laboratoris i altres àmbits industrials on són necessaris continus moviments de càrregues, com s'ha comentat abans.

1.3. Objectius

L'objectiu principal del projecte és tenir un prototip a escala per simular el comportament d'un AGV sent capaç de moure's per un entorn conegut amb una càrrega que haurà de transportar a un lloc o altre segons correspongui. En el nostre cas, s'obtindrà el prototip que podrà realitzar les accions de moviment, però no de transport. Per a aquest transport de càrrega només caldrà afegir un recipient on posar el material per a que no caigui, per tant no es contemplarà aquest fet. Només es tindrà en compte el moviment d'un punt a un altre.

Per aconseguir això, prèviament es necessitarà un model de l'AGV i un altre de l'escenari. Realitzant una sèrie de passos, finalment s'obtindrà el prototip. Aquests passos són descrits al punt següent.

1.4. Tasques

1.4.1. Tasca 1. Reunir informació i coneixements sobre els AGV

Aquesta tasca consisteix en l'adquisició dels coneixements necessaris sobre els AGV per a poder realitzar el projecte d'una manera més eficient ja que no es pot realitzar un bon treball si no es disposa d'uns coneixements adequats sobre els AGV.

Així doncs, la Tasca 1 serà necessària per a complir els dos subobjectius (obtenir el model de

l'escenari i del vehicle) i per a l'objectiu principal (obtenir el prototip).

1.4.2. Tasca 2. Moviments bàsics per al seguiment de línies (motors i sensors)

Com es veurà en capítols posteriors, el seguiment de línies és l'opció que s'implementarà per a guiar l'AGV a través de l'escenari. Per tant, durant la realització d'aquesta tasca es duran a terme la programació i proves necessàries per a disposar del seguidor de línies. Això vol dir que a la finalització d'aquesta tasca el robot haurà de ser capaç de seguir una línia correctament (tot i que la funcionalitat es comprovarà a la tasca 6).

Donat que la detecció de línia depèn totalment dels materials, aquesta tasca quedarà influïda per les especificacions de l'escenari i a la vegada influirà en la construcció física de l'escenari, ja que s'haurà de construir l'escenari amb els materials que haguem trobat convenients per a la detecció.

1.4.3. Tasca 3. Comunicació amb el PC mitjançant Bluetooth

Un cop finalitzada aquesta tasca, el robot haurà de ser capaç de comunicar-se per Bluetooth amb el programa de control que hi haurà executant-se en un PC. Per tant, s'implementaran les parts corresponents tant al robot com a la computadora per a que això sigui possible.

Aquesta tasca correspon al subobjectiu d'obtenció del model del vehicle, ja que aquesta comunicació no depèn de l'escenari (l'escenari influeix en la tria de la tecnologia de comunicació però no en la implementació un cop triada). S'explicarà en capítols posteriors el motiu de la tria de la tecnologia Bluetooth.

1.4.4. Tasca 4. Implementació del programa de control i de l'AGV

Tot i que abans d'iniciar aquesta tasca, el programa de control ja existirà (per a la tasca 3 realitzarem la part corresponent al Bluetooth), en aquesta tasca s'implementarà tota la resta del programa de control (bifurcacions, unions, mapa, etc) i la part adient al robot. Així doncs, un cop finalitzada, tota la implementació corresponent al PC i al robot ja és finalitzada. Això vol dir que si tot és correcte el robot ja pot funcionar correctament.

La implementació del programa de control serà independent de l'escenari, però la implementació del robot, sí que dependrà de l'escenari ja que implementem el moviment a baix nivell. Per tant,

aquesta tasca estarà influïda per les especificacions de l'escenari.

1.4.5. Tasca 5. Preparació de l'escenari

L'AGV ja pot funcionar, però necessita un entorn on es pugui provar el seu funcionament, per tant, en aquesta tasca es construirà una maqueta per a aquest prototip. Com s'ha comentat, influirà directament a les tasques 2 i 4, ja que s'haurà de construir tenint en compte les especificacions de l'escenari (definides prèviament a tota implementació) per a que el vehicle pugui circular en ell.

Un cop finalitzada la tasca, ja es disposa de tot el necessari per a provar el funcionament del prototip.

Aquesta tasca correspon íntegrament al subobjectiu corresponent a la obtenció del model de l'escenari.

1.4.6. Tasca 6. Banc de proves

En aquesta tasca es realitzaran totes les proves necessàries per a comprovar que el funcionament de l'AGV sigui correcte. Així doncs, al finalitzar la tasca, l'AGV funcionarà correctament segons les especificacions donades.

1.4.7. Tasca 7. Redacció de la documentació del projecte

Aquesta tasca dura gran part de la vida del projecte ja que es realitza conjuntament amb la resta. Un cop finalitzada, el resultat serà aquest document, on es recullen les especificacions, detalls del desenvolupament, resultats obtinguts, i altra informació relativa a la realització del projecte.

1.4.8. Tasca 8. Preparació de la presentació

Aquesta tasca correspon a la realització de la presentació, tan al document, com a la preparació oral.

1.4.9. Tasca 9. Reunions de seguiment

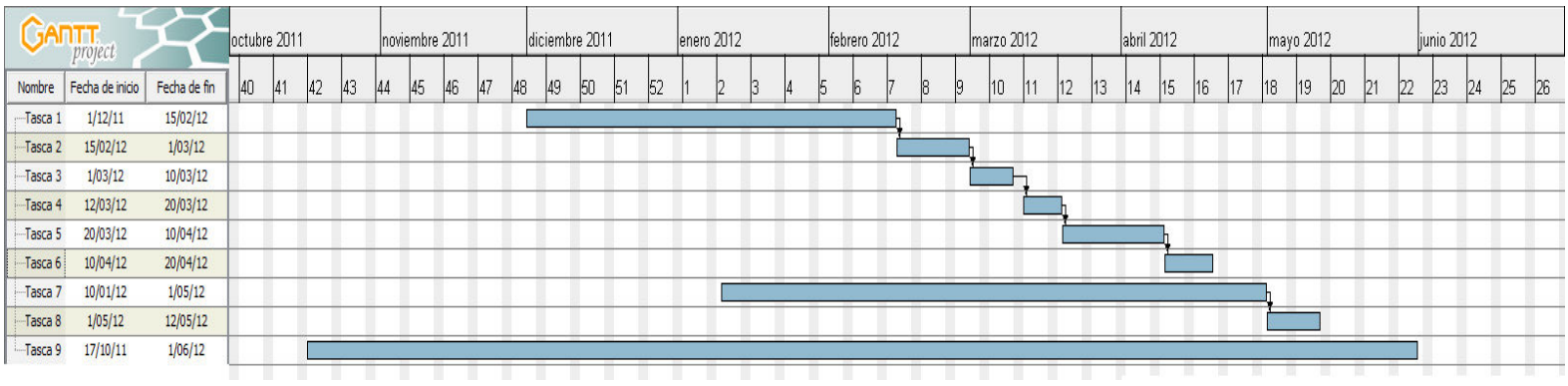
Les reunions de seguiment es realitzen per a que el director del projecte porti un control de la feina realitzada, resolució de dubtes, aclariments, etc. Aquestes reunions es realitzaran cada 15 dies

(excepte festius). Per tant, de la mateixa manera que la tasca 7, serà una tasca que serà present durant tota la vida del projecte.

Tasca	Descripció	Hores planificades
1	Reunir informació i coneixements sobre els AGV	55
2	Moviments bàsics per al seguiment de línies (motors i sensors)	15
3	Comunicació amb el PC mitjançant Bluetooth	20
4	Implementació del programa de control i de l'AGV	15
5	Preparació de l'escenari	15
6	Banc de proves	20
7	Redacció de la documentació del projecte	150
8	Preparació de la presentació	40
9	Reunions de seguiment	32
Total		362 hores

Com es pot observar, el total d'hores del projecte són 362 hores. S'ha optat per utilitzar el 20% restant (de les 450 hores estipulades) per a possibles inconvenients que puguin aparèixer durant la realització del mateix.

1.5. Planificació



Com s'observa, la Tasca 7 (redacció de la documentació) i la Tasca 9 (reunions de seguiment) són actives durant la vida del projecte ja que es fan conjuntament amb el projecte. No obstant, això no vol dir que s'estiguin realitzant sempre i alhora, sinó que s'aniran intercalant amb les altres tasques, és a dir, tot i estar realitzant la Tasca 4, també hem de realitzar les reunions de seguiment (tasca 9), però no al mateix temps.

Cal destacar, que tot i que la planificació és lineal (una tasca darrera una altra), això només serà així suposant que tot sigui correcte directament. A la tasca 6, banc de proves, és on es poden detectar errors, el que provocaria tornar a tasques anteriors per a corregir aquests errors. Per tant, no és exactament lineal, sinó que existeix un enllaç per tornar enrere si és necessari.

D'altra banda, com s'ha comentat en el punt 1.4, s'ha utilitzat el 20% de 450 hores per a possibles imprevistos que puguin aparèixer. D'aquesta manera, com es veu en el gràfic, la finalització del projecte és al Maig.

1.6. Anàlisi de viabilitat

Per determinar la viabilitat del projecte es necessita fer una estimació dels costos, riscos i un anàlisi de viabilitat tècnica.

1.6.1. Estimació de costos

Els costos previstos són els següents:

	Cost per hora	**Hores dedicades**	**Total**
Direcció	30,00 €	64 h	1.920,00 €
Desenvolupament	20,00 €	362 h	7.240,00 €
Robot	-	-	-50,00 €
PC	-	-	-100,00 €
Materials	-	-	-50,00 €
Total	-	-	8.960,00 €

Es pot observar que tant el robot, el PC i els materials són negatius, ja que són considerats com a amortitzacions. Pel que fa als costos de direcció i desenvolupament hem considerat que aquests són adients.

Els costos no afecten a la viabilitat del projecte ja que es tracta d'un projecte de fi de carrera, però en un cas real s'han de tenir en compte.

1.6.2. Estimació de riscos

Durant el transcurs del projecte hi ha diversos riscos que cal vigilar. El primer és la planificació optimista, és a dir, que es planifiquin les coses amb un menor temps del que realment necessiten. El segon risc és l'aparició de tasques imprevistes, per exemple si per realitzar una tasca primer cal adquirir certs coneixements i no ho s'havia previst la seva inclusió.

1.6.3. Viabilitat tècnica

La viabilitat tècnica ve donada per dos elements:

- Com es veurà al capítol 2, hi ha empreses que es dediquen a automatitzar els sistemes de logística interna amb molts tipus d'AGV i sistemes diferents.
- El prototip es pot obtenir combinant diferents exercicis educatius amb el BoeBot.

Aquests dos elements, fan que sigui totalment viable tècnicament. Aquest fet i que els costos no afecten, fan que el projecte sigui totalment viable.

Capítol 2. Estat de l'art

Un AGV, com s'ha comentat resumidament en el capítol anterior, és un vehicle autònom capaç de realitzar determinades tasques que se li encomanen. En aquest capítol es veurà primerament el model bàsic i les seves parts, es llistaran algunes tecnologies que utilitzen i finalment, com a exemple, es veuran algunes empreses i els productes que ofereixen.

2.1. Model bàsic

En la figura 2.1 es poden veure tots els elements que es necessiten per a que un AGV pugui funcionar. Aquest elements es divideixen segons la seva funcionalitat:

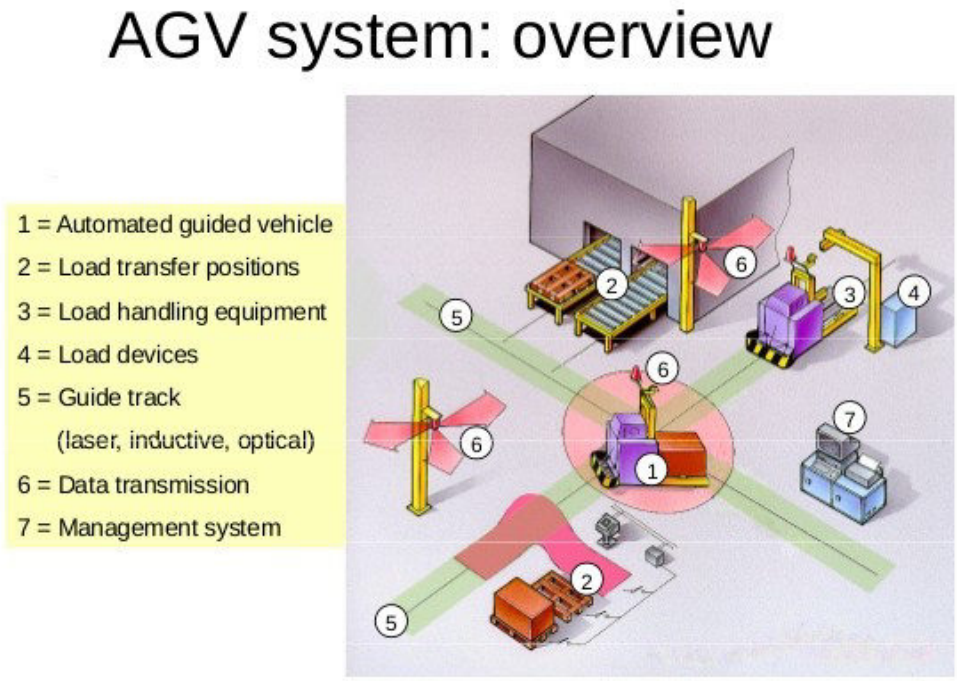


Fig 2.1 Sistema d'AGV [1]

- Seguretat: Aquests elements estan destinats a garantir que tot el sistema sigui segur tant per als treballadors que hi treballen com per als mateixos AGV (recordar que són entorns industrials). En la imatge no apareix cap element marcat, però podria haver-hi tanques delimitadores, barreres, etc.
- Guiatge: (Número 5). Els elements de guiatge són imprescindibles per a que el robot es situï en l'entorn. Segons com es vulgui implementar el sistema de moviment dels AGV aquests elements seran uns o altres (veure punt 2.3).
- Localitzacions (Números 2, 3 i 4): Les localitzacions són aquells elements que actuen com a

punts del mapa independentment de la seva utilitat (alguns d'ells seran de càrrega, descàrrega, parking, etc).

- Sistema Gestor (Números 6 i 7): aquest element és molt important. El sistema gestor s'encarregarà de gestionar el trànsit d'AGV i de donar les ordres. A més, serà el punt d'unió entre cada AGV i els usuaris.
- AGV (Número 1): Agent destinat a realitzar les ordres que donarà el programa gestor.

2.2. Parts d'un AGV

Un AGV per a realitzar les tasques corresponents necessita diversos elements:

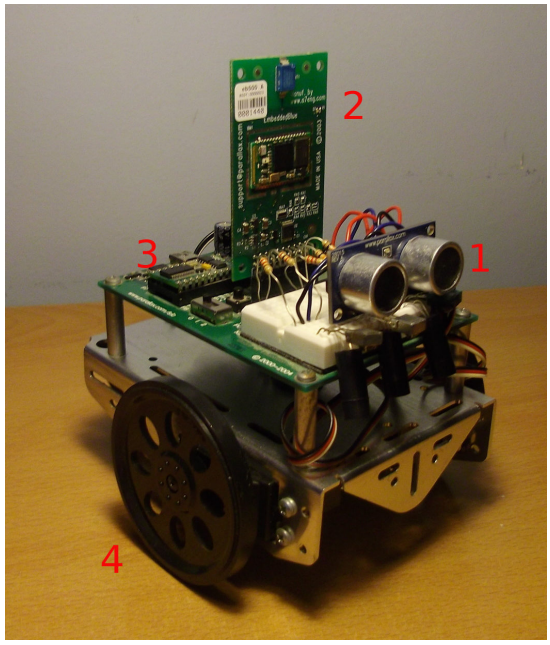


Fig. 2.2 AGV

1. Elements de percepció de l'entorn: Aquests elements es necessiten per a percebre l'entorn ja que l'AGV no es pot moure a cegues. Per tant, segons les necessitats, es poden instal·lar infrarojos, càmeres, ultrasons, o d'altres elements.
2. Elements de comunicació: Segons l'abast de la comunicació que es necessiti es pot optar per diversos tipus de comunicació: Bluetooth, Wifi, radiofreqüència,... Aquestes comunicacions ens serviran per donar les ordres, informar d'events que succeeixin, comunicar-se amb altres AGV, etc.
3. Processador: Cal un element que obtingui les dades dels elements de percepció de l'entorn, les entengui i pugui determinar una acció a realitzar, ja sigui enviar per el canal de comunicació el fet o donar les ordres necessàries als actuadors.

4. Actuadors: Elements finals que actuaran amb l'entorn. Aquest elements poden ser les rodes, braços mecànics, pinces o d'altres elements.

Segons les nostres necessitats, es pot o no incloure un element més: una computadora de control per a l'AGV. Això dependrà de la capacitat de procés i memòria que ofereixi el propi robot. Si aquesta ho permet podem ometre la computadora. En aquest projecte es considerarà com a AGV el conjunt de computadora i robot, tal i com es veu en la figura 2.3. S'ha decidit utilitzar-la perque el BoeBot que s'usarà (veure punt 3.2.1) disposa d'una capacitat de procés limitada.

Amb la inclusió d'aquesta computadora, el robot només es preocuparà de controlar el seu propi moviment i de trobar events. Quan trobi un event informarà a la computadora, aquesta prendrà la decisió corresponent i la comunicarà al robot.

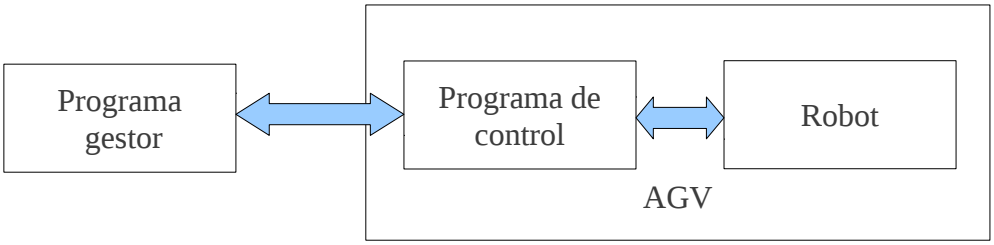


Fig. 2.3. Esquema de relacions

2.3. Sistemes de guiatge

Els sistemes de guiatge són aquells que permeten a l'AGV saber on es troba i li permeten navegar pel seu entorn. Existeixen molts tipus de sistemes de guiatge. Tot seguit es veuen alguns d'ells.

2.3.1. Seguidor de línies

Aquest sistema es basa en que a l'entorn existeix algun tipus de línia que el robot pot seguir mitjançant els elements de percepció de l'entorn. Aquest sistema és un dels més senzills, segurs i fiables, però dels menys flexibles a canvis, ja que la línia és un element físic al terra. Si es volgués canviar els elements de localització de lloc, també s'hauria de canviar la línia (a més del mapa intern de l'AGV).

2.3.2. Visió per càmeres

Aquest sistema consisteix en disposar de certes càmeres al robot i mitjançant el procés de les imatges obtingudes, intentar reconèixer certs objectes per a poder determinar en quin punt del mapa es troba. Aquest sistema és més flexible a canvis. A diferència del seguidor de línies no existeix una línia física al terra i per tant, si es volen modificar les ubicacions caldrà canviar únicament el mapa per a que el robot sàpiga que ha d'identificar i quan. Un inconvenient és la complexitat del processament d'imatges.

2.3.3. Seguidor de cable

Aquest mètode consisteix en enterrar a molt poca profunditat un cable dibuixant el camí que ha de seguir l'AGV. El robot disposa del sensor adient per detectar les senyals que s'enviaran pel cable i així, anar seguint el camí. De la mateixa manera que el seguidor de línia, és un dels sistemes menys flexibles a canvis.

2.3.4. Navegació per làser

Aquest és un dels mètodes més usats. Consisteix en col·locar material reflectant en determinats punts de l'entorn. L'AGV anirà llançant rajos amb el làser i aquests, al reflexar-se, tornen a l'AGV, que els detecta. Així pot saber la posició en que es troba.

És molt usat ja que proporciona una gran precisió. Pot calcular la desviació en graus del làser i fins i tot la distància a la que es troba.

2.4. Sistemes de comunicació

Els sistemes de comunicació ens permeten comunicar-nos amb altres dispositius de l'entorn. Hi ha diferents tipus de comunicació a triar segons les característiques que es necessitin. Tot seguit es llisten algunes.

2.4.1. Wifi

La connexió Wifi proporciona un abast de comunicació gran i una velocitat gran. És el mètode més utilitzat per les comunicacions.

2.4.2. Radiofreqüència

La comunicació per radiofreqüència ofereix un abast encara més gran que el de la Wifi i amb molts menys consum. També és un mètode molt utilitzat en entorns reals.

2.4.3. Bluetooth

La comunicació Bluetooth permet una comunicació a curta distància i amb poc consum d'energia. No obstant, l'establiment de la connexió és lent, ja que busca tots els elements disponibles en el radi d'abast i després es connecta al dispositiu determinat.

En el nostre projecte s'ha triat el Bluetooth perquè és l'únic disponible de manera ràpida per al BoeBot independentment del consum, ja que el que ofereix menor consum és la radiofreqüència.

2.5. Empreses

Hi ha un gran nombre d'empreses que es dediquen a construir AGV i implantar-los en plantes de producció, laboratoris, magatzems, etc. amb aquestes i altres tecnologies. Alguns exemples són: Savant Automation, JBT Corp. o SoftDesign entre moltes altres. Aquest fet reflecteix la importància dels sistemes d'AGV en la indústria actual.

Tot seguit es detallen algunes de les empreses en el sector i els productes que ofereixen als seus clients.

2.5.1. SoftDesign

L'empresa SoftDesign disposa de més de 20 anys d'experiència amb AGV i ofereix diferents models segons els requeriments del client. L'empresa crea els AGV, els programes de control i els sistemes de visió i detecció.

Per al control dels AGV, disposen d'un sistema anomenat MAX, que usa un sistema client-servidor amb Windows. Aquest sistema disposa d'un servidor que es pot connectar a diferents tipus de dispositius i, per suposat, als AGV. Aquest servidor serà l'encarregat de gestionar tots els processos de treball. Aquí disposem d'un gràfic extret de la pàgina web de l'empresa.

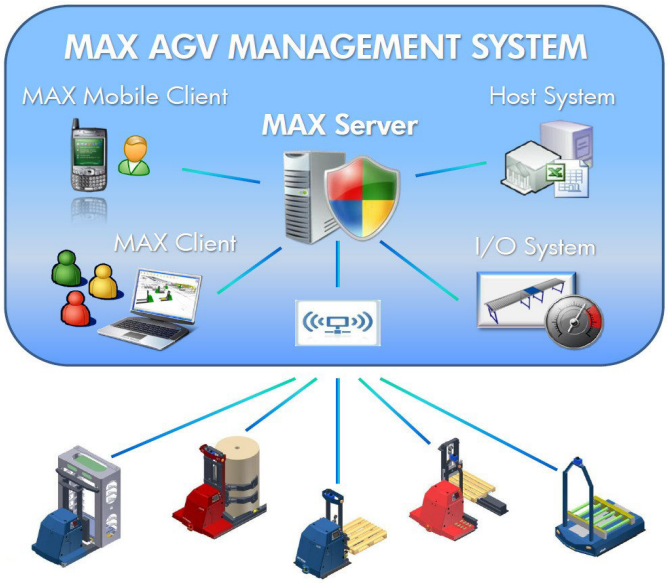


Figura 2.4. MAX management System de SoftDesign [2]

Pel que fa al moviment del robot, l'empresa ofereix moviment per seguiment d'una cinta o navegació per làser entre d'altres.

2.5.2. Savant Automation

L'empresa Savant Automation té seu als EUA i de la mateixa manera que SoftDesign, es dedica a implantar sistemes d'AGV en plantes de producció o altres àmbits industrials.

Savant Automation disposa d'AGV amb un panell de control per on s'introdueixen les ordres manualment. Els AGV calculen les rutes ells mateixos i per tant, els problemes de trànsit els solucionen entre els mateixos vehicles. No obstant, Savant Automation també ofereix un gestor de trànsit, ordres i rutes. Aquest rep les ordres i selecciona l'AGV que cregui convenient per a realitzar la tasca.

L'empresa ofereix diferents tipus d'AGV segons les necessitats del client. Entre els AGV destaquen:

- Tow AGV: se li pot acoblar un remolc amb la càrrega que ha de transportar.

Figura 2.5. Tow AGV [5]

- AGV Carts: petits AGV que se situen sota la càrrega per aixecar-la i transportar-la. També estan disponibles amb un suport per a situar la càrrega sobre directament.

Figura 2.6. AGV Carts [5]

- Fork AGV: vehicles amb pales per aixecar la càrrega. Funcionen de la mateixa manera que una carreta elevadora.

Figura 2.7. Fork AGV [5]

En quant a la navegació dels AGV, Savant Automatio*n* ofereix guiatge per cable o per un petit sensor de radiofreqüència que detecta ones d'emissors situats sota el terra, però destaca un sistema que funciona mitjançant un giroscopi instal·lat al robot. Cada certa distància es col·loca sota el terra un petit dispositiu magnètic que el giroscopi detecta i és capaç de saber on és, la direcció que té, la desviació que hi ha respecte el camí principal, etc.

La comunicació que ofereix principalment l'empresa és la radiofreqüència per a comunicar els AGV entre si, i per evitar col·lisions entre els AGV utilitza ultrasons, infrarojos i *bumpers* (similar a un para-xocs que detecta el contacte).

2.5.3. JBT Corporation

JBT Corporation té casi 30 anys d'experiència en la implantació de sistemes d'AGV amb més de 300 sistemes implantats a tot el món i amb més de 3.000 AGV funcionant de manera continua.

JBT recomana com a sistema de guiatge principalment el làser. Instal·len en l'AGV un làser que serà reflectit en uns blancs col·locats en punts estratègics per a guiar l'AGV. Aquest sistema és molt flexible a canvis, però, si es desitja, també disposen d'un sistema guiat per un cable electromagnètic o fins i tot guiats amb un comandament remot.

La corporació ha desenvolupat una aplicació anomenada SGV Manager que s'encarrega de gestionar l'enrutament, trànsit, emmagatzemar informació i comunicar-se amb altres dispositius mitjançant WiFi. Aquest programa és executable a Windows.

Addicionalment, han desenvolupat una aplicació per a modificar les trajectòries dels AGV o el que s'ha anomenat en aquest projecte com a mapa. És a dir, des d'aquesta aplicació es pot dir que un reflectant determinat sigui l'equivalent a una posició concreta, event, etc.

Pel que fa als AGV, disposa de fins a 27 models diferents segons la utilitat que se'ls vulgui encomanar i l'entorn en el que treballaran. Tot seguit es mostren dos models:

- Carreta: disposa de dues pales per a aixecar càrregues, principalment palets.

Figura 2.8. Carreta de forquilla [4]

- Carreta APL: Aquest AGV està especialitzat en transportar bobines.

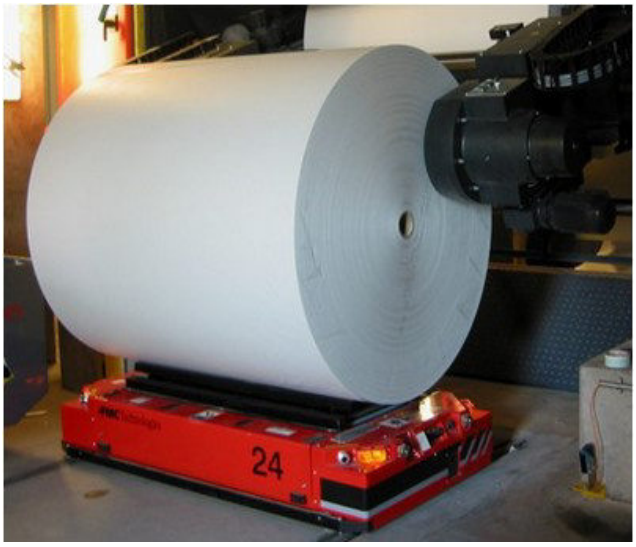


Figura 2.9. Carreta APL [4]

Capítol 3. Especificacions

El nostre objectiu és arribar a tenir un prototip a escala que emuli el comportament d'un AGV en un entorn determinat, ja sigui un magatzem, un laboratori o una planta de producció. Així doncs, per al nostre prototip es necessiten, principalment 2 models.

3.1. Model de l'entorn

El model de l'entorn es necessita per a comprovar que el nostre prototip funciona correctament. Per tant caldrà construir una maqueta per on després l'AGV pugui circular. S'ha optat per implementar un model seguidor de línies, ja que és el més senzill i econòmic.

Com es pot comprovar en la figura 3.1, el model disposa d'una zona de recepció d'ordres (1) on l'AGV rebrà les destinacions a les que ha d'anar, una zona de recirculació (2) en cas de que hagi de tornar a una ubicació a la que no ha pogut accedir (per exemple si estava ocupada per un altre AGV), i finalment les dues ubicacions (3 i 4), on haurà de portar les càrregues. Addicionalment, durant tot el recorregut hi ha marques laterals (5) que indiquen que en aquell punt hi ha algun event.

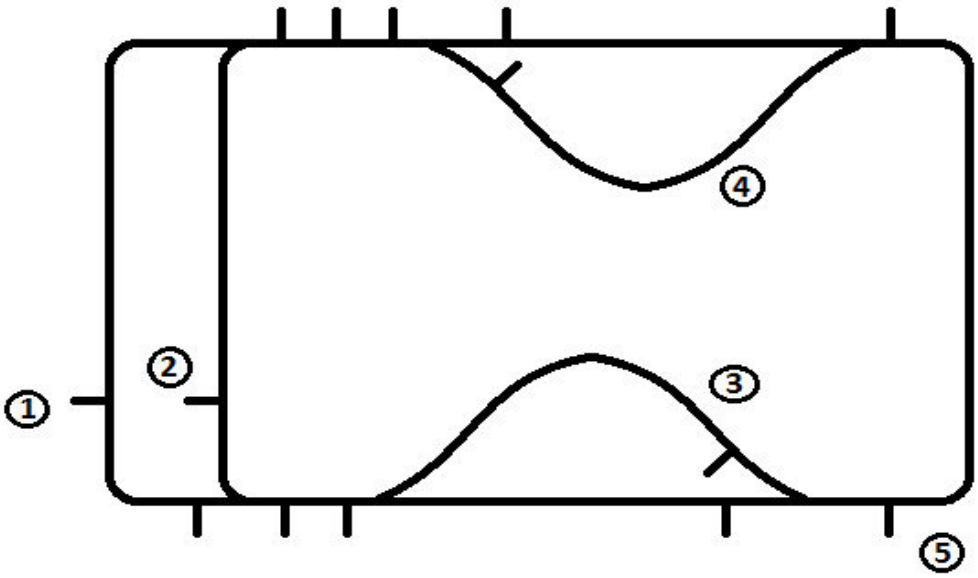


Figura 3.1. Escenari de treball

Donades les especificacions del robot (veure punt 3.2), el robot disposa només de 2 punts de referència per seguir una línia. Per tant, és estrictament necessari que els angles als revolts no siguin superiors a 45º. En angles superiors, amb només dos sensors no es pot garantir que pugui seguir la línia satisfactòriament. En el cas de les bifurcacions i unions també es manté aquest requisit (veure

figures 3.2, 3.3 i 3.4).

Pel que fa a les marques, sempre seran per la dreta (el robot té un sensor dedicat a la dreta per a les marques) i aniran prèviament a un event. Es consideren 5 tipus d'event:

- Inici d'unió entre dos camins.
- Final d'unió entre dos camins.
- Principi de bifurcació en dos camins.
- Rebuda d'ordres.
- Buida d'ordres.

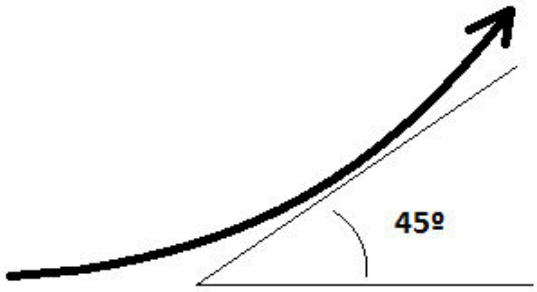


Figura 3.2. Revolt

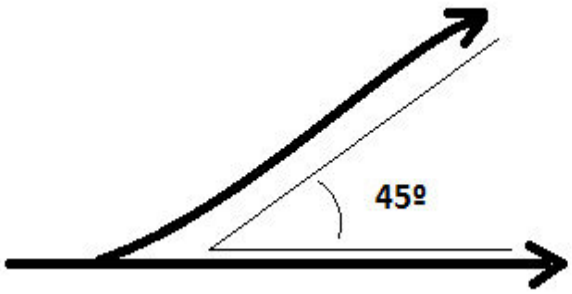


Figura 3.3. Bifurcació

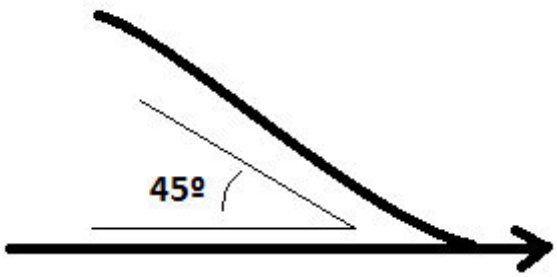


Figura 3.4. Unió

Finalment, cal tenir en compte el número de les posicions associades a cada marca. Com es veurà

en el capítol 4, el programa de control disposarà d'una variable que s'usarà per determinar en quin punt es troba l'AGV. Aquesta variable NO es correspon amb el nombre de marques. Això es deu a que el robot, en unions i bifurcacions, detecta el camí principal (veure punts 4.4.3 i 4.4.4), per tant, algorísmicament s'ignoraran aquestes marques i es considerarà coma posició tota la unió i tota bifurcació. El mapa queda de la següent manera:

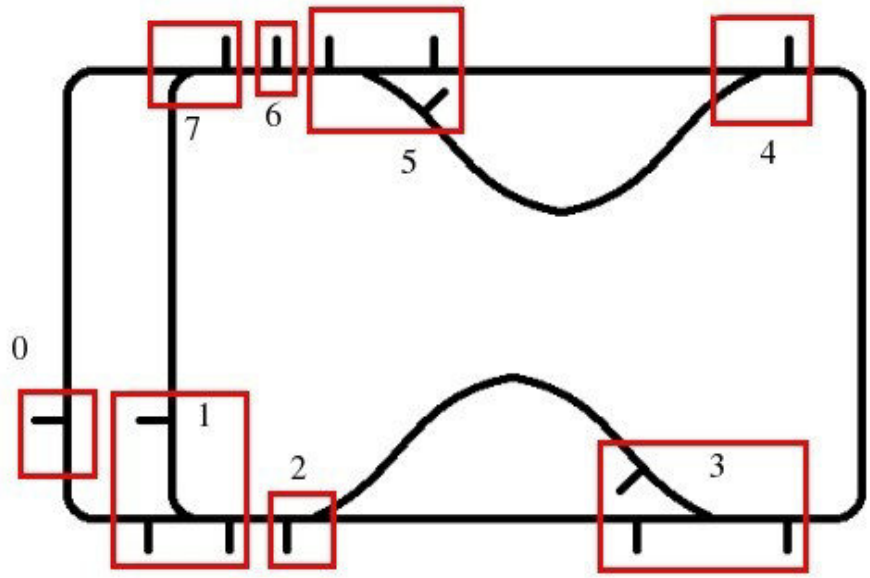


Figura 3.5 Posicions

3.2. Model de l'AGV

Com s'ha comentat en el capítol 2, es considera AGV el conjunt mínimament autònom. Tot i que es pot reduir a que el robot realitzi totes les tasques de control i moviment sense necessitar una computadora associada, en aquest projecte es considerarà el conjunt de PC i BoeBot com a l'AGV. Això es degut a que el BoeBot té una capacitat de procés i memòria molt reduïdes. Així doncs, es dividiran les tasques. La computadora mantindrà la comunicació amb el gestor, el mapa i les accions que ha de realitzar en cada punt del mapa. El robot només s'encarregarà de moure's i informar a la computadora quan trobi un event i esperar l'acció que ha de realitzar.

3.2.1. BoeBot

El robot del que es disposa és un *BoeBot* amb una placa *Board of Education* desenvolupada per *PARALLAX*:

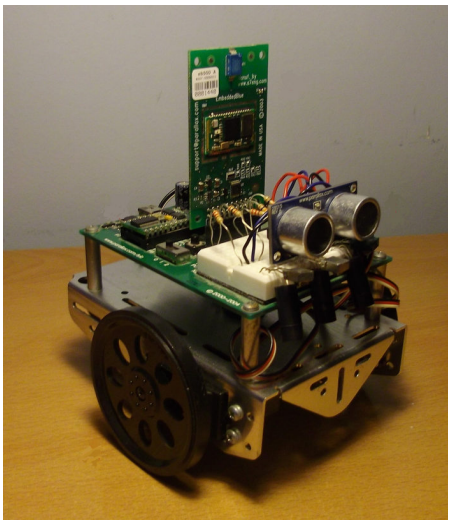


Figura 3.5 BoeBot de Parallax

Per a realitzar les tasques necessita els següents elements:

1. Dos emissors d'infrarojos mirant a terra per al seguiment de línies i els seus respectius receptors.

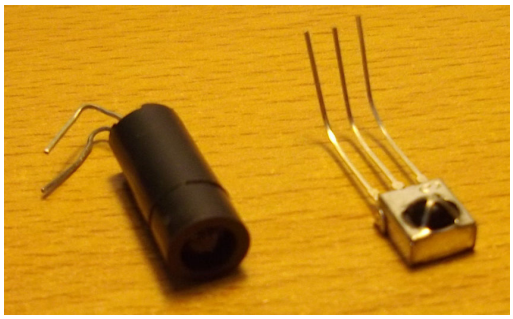


Figura 3.6 Emissor i receptor d'infrarojos

2. Un emissor d'infrarojos en un lateral per la detecció de marques laterals i el seu receptor.
3. Un sensor d'ultrasons *PING)))* de *PARALLAX* per detectar distàncies a objectes en la trajectòria del robot.

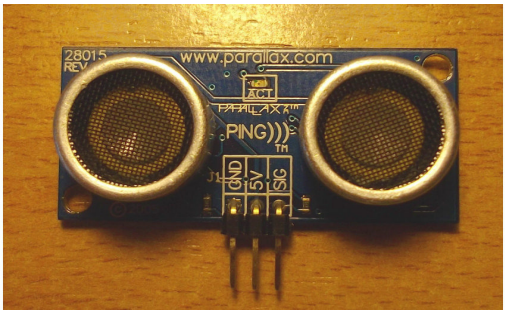


Figura 3.7 Placa d'ultrasons

4. Un transceptor Bluetooth *eb500* de A7 Engineering per a la comunicació amb el programa de control en un ordinador.

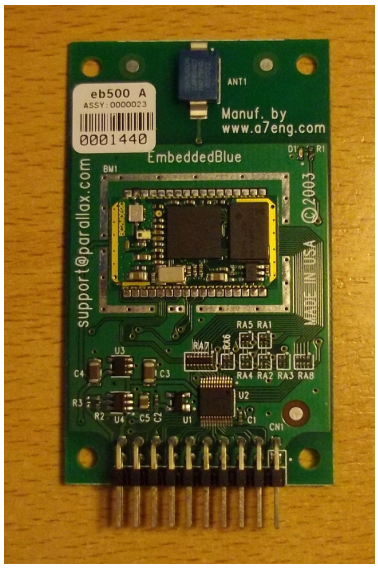


Figura 3.8 Placa Bluetooth

Amb aquests elements, el robot pot seguir línies, detectar marques laterals, calcular distàncies a obstacles i comunicar-se amb l'ordinador de control, que serà tot el necessari per a que pugui realitzar les tasques correctament.

3.2.2. Computadora

Es disposa d'una computadora TOSHIBA NB500 amb les següents característiques:

- 1Gb de memòria RAM.
- Processador Intel ATOM.

Figura. 3.2.5 TOSHIBA NB500 [6]

La computadora no disposa de Bluetooth integrat, per tant, s'ha adquirit un adaptador USB de *Belkin* amb la finalitat d'afegir aquesta utilitat a la computadora.

Figura 3.2.6 Adaptador USB - Bluetooth

El programa del robot serà implementat en *PBasic* amb l'entorn de programació *BasicStamp* de *Parallax* [7] i el programa de control serà implementat en *Java* amb l'entorn *NetBeans* [8]. Ambdós entorns s'executaran sobre el sistema operatiu Windows 7 i es poden obtenir a les seves respectives webs de forma gratuïta.

3.3 Programa gestor

Com s'ha mencionat anteriorment, el programa gestor és l'encarregat de gestionar el trànsit i per tant, d'implementar polítiques de gestió. La realització d'un bon sistema gestor és, en sí mateix, un projecte Així doncs, aquest projecte no inclou aquest i el programa de control serà el que prendrà decisions.

3.4. Problemes previstos

Els AGV coneixen l'escenari per el qual es mouen i, per tant, saben com arribar a les destinacions indicades. Tot i que en aquest projecte només es veurà un únic AGV, aquest ha de poder circular en un entorn amb altres AGV. Així doncs s'ha de definir un codi de circulació que tots els AGV hauran de complir. Els dos punts claus seran les bifurcacions i les unions.

3.4.1. Bifurcacions

En una bifurcació, hauran de decidir si la prenen o no depenent de la destinació que tenen fixada. En el nostre escenari, les bifurcacions porten a una destinació i després tornen a la ruta principal, per tant, l'AGV la prendrà si porta a la destinació objectiu i si no seguirà el seu camí principal. No

obstant, es pot donar el cas de que hagi d'anar per exemple a 3 o a 4 (hi ha dos possibles destinacions). Així doncs, l'AGV haurà de decidir si pren aquesta destinació o prendrà la següent.

Addicionalment, també pot passar que la ubicació sigui ocupada i no pugui prendre-la. Com només es disposa d'un únic AGV les ubicacions mai estaran ocupades, però en una aplicació real amb més d'un AGV s'hauria de tenir en compte aquest cas i fins i tot que es poguessin donar aquest cas i l'anterior.

Es prepararà el programa de control (veure punt 4.4) per a que pugui treballar amb les possibles ubicacions abans comentades. Un cop preparades les estructures de dades per a acceptar les possibles ubicacions, s'hauria d'afegir un semàfor idèntic a l'implementat a les unions (veure punt 3.4.2) per a comprovar si la ubicació és ocupada o no. Finalment, quan l'AGV que l'ocupa arriba a la unió per sortir de la bifurcació alliberaria el semàfor per a deixar l'ubicació lliure.

Aquesta part NO està contemplada en el nostres projecte ja que com s'ha mencionat en el punt 3.3, aquesta tasca correspon al programa gestor, que queda fora de l'abasta del projecte. En aquest projecte, l'AGV disposa de les diferents destinacions, però tria la primera que troba lliure (sempre serà la primera, ja que circula sol).

3.4.2. Unions

En les unions, pot donar-se que, eventualment, dos AGV vulguin passar a la vegada, el qual els portaria a col·lisionar. Així doncs, s'aplicarà un sistema de semàfor (no és un semàfor físicament). És a dir, quan un AGV arribi a la marca d'inici d'una unió preguntarà al programa gestor si hi ha algú passant. Si hi ha un AGV s'esperarà, si no, enviarà al programa gestor que ell passa i en aquell moment ja no hi pot passar cap altre AGV. Quan arribi a la marca final, enviarà de nou al programa gestor que ja ha passat i per tant, ja pot passar un altre AGV.

Aquest semàfor ha d'estar controlat pel sistema gestor i cal afegir un control de prioritats. Com s'ha mencionat diferents cops, en la nostra aplicació només hi ha un únic AGV i queda fora de l'abast del projecte la implementació d'un gestor. Per tant, no s'ha implementat la consulta i notificació al programa gestor, però sí s'han implementat els semàfors. Per tant per a una aplicació real, només ens caldria afegir aquestes dues comandes.

Capítol 4. Desenvolupament

En aquest capítol es veurà detalladament el desenvolupament de l'aplicació, tant pel que fa al muntatge del elements del robot, la programació del robot, el programa de control i el muntatge de l'escenari.

4.1. Muntatge de l'Escenari

Per al muntatge de l'escenari s'ha usat un cartró com a base de mides 100x70 cm. Sobre aquesta base es va posar cinta antielèctrica per fer el circuit, però després de fer proves amb el robot, es va optar per enganxar sobre el cartró una capa de fulls DIN-A4, i sobre aquests fulls posar la cinta. Amb aquesta combinació, els infrarojos diferencien millor les dues superfícies.

L'escenari queda de la següent manera:

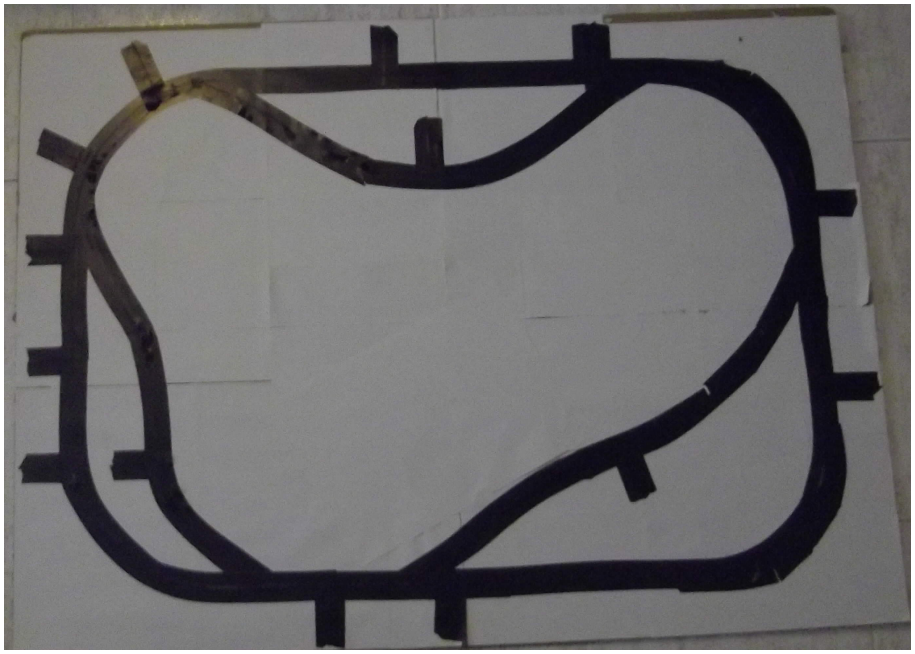


Figura 4.1 Escenari

Es pot observar la diferència amb el model original (veure punt 3.1). Aquestes diferències es deuen al reduït tamany de la base de cartró i la restricció dels angles dels revolts, ja que si fem els revolts més tancats el robot pot perdre la línia.

Cal recordar que es una maqueta i un prototip per provar el funcionament de l'AGV, no es necessita

tenir un entorn totalment realista, només ens cal que es diferenciïn correctament les dues superfícies. En un entorn real caldria buscar altres materials i tenir un entorn de proves més gran.

4.2. Muntatge del robot

El muntatge del robot es pot dividir en 3 parts: el muntatge dels infrarojos, el muntatge de la placa d'ultrasons, i finalment la placa Bluetooth.

4.2.1. Muntatge dels infrarojos

Cal connectar els infrarojos tal i com mostra la figura 4.2. Cal anar amb compte de no confondre les potes positiva i negativa dels LEDs (la pota més llarga és la negativa) i vigilar de no usar els ports reservats al bluetooth (veure punt 4.2.3) ni els pins 12 i 13 (reservats al motors de les rodes) [10].

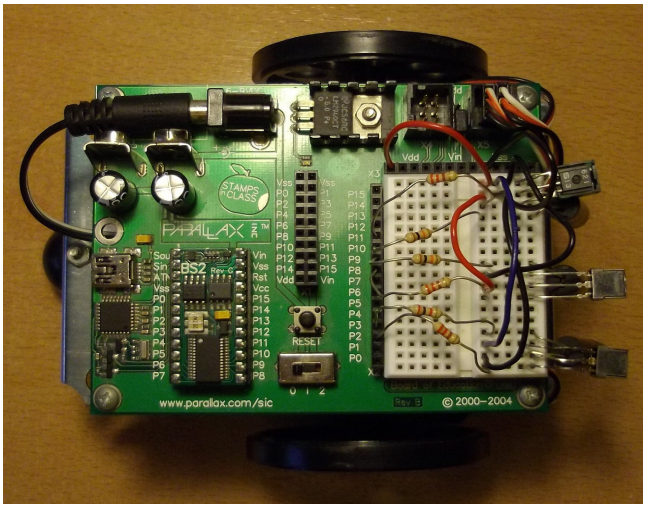


Figura 4.2 Placa amb infrarojos

4.2.2. Muntatge de la placa d'ultrasons

Els ultrasons cal montarlos com mostra la figura 4.3. La figura mostra només la placa d'ultrasons, per tant, s'ha d'anar en compte de no ocupar pins que s'usaran per als infrarojos (punt 4.2.1), els ports reservats al bluetooth (veure punt 4.2.3) ni els pins 12 i 13 (reservats al motors de les rodes) [11].

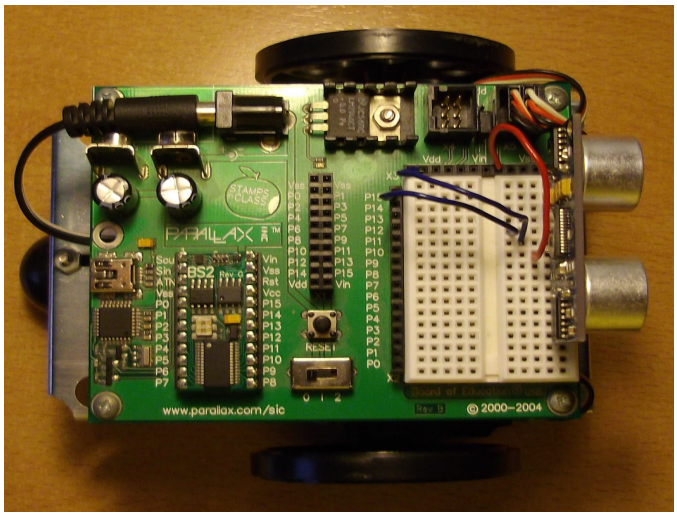


Figura 4.3 Placa amb ultrasons

4.2.3. Placa Bluetooth

La placa Bluetooth té un connector reservat que cal col·locar en la posició adequada (si la posem al revés els pins no es correspondran). Te reservats els pins 0, 1, 5 i 6 [9]. Per tant, cal no connectar cap infraroig ni l'ultrasò en aquests pins.

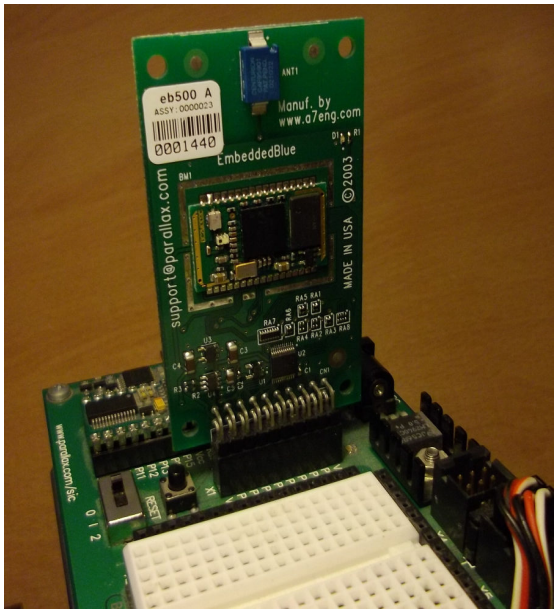


Figura 4.4 Placa amb Bluetooth

4.3. Programa gestor

Com s'ha comentat abans, el programa gestor serà el que transmetrà les ordres al programa de control. Donat que no es realitzarà un sistema gestor pels motius comentats al punt 3.3, aquest programa gestor serà substituït per un fitxer de text d'on el programa de control llegirà la llista

d'ordres. En un cas real, el programa de control informaria, a més de la llista d'ordres, de l'estat dels semàfors quan és necessari o d'altra informació necessària.

Aquest fitxer contindrà en cada línia un enter en el format següent:

9
destinació
destinació
...
9
destinació
...
0

És a dir, l'enter 9 indica l'inici d'una seqüència de destinacions alternatives. Aquesta seqüència finalitza quan es rep un altre 9 (serà l'inici d'una nova seqüència obligatòria) o un 0, que serà l'indicador de final. Per tant, si la sentència és:

9
2
3
9
5
0

Això vol dir que l'AGV ha d'anar a 2 o a 3 i a 5.

Com el nostre escenari només disposa de dues ubicacions, només es podran tenir com a ubicacions 1 i 2 (i no 3 o 5, com a l'exemple anterior).

4.4. Programa de control

El programa de control serà l'encarregat de donar les ordres físiques al robot (para, avança i gira a l'esquerra en la bifurcació). Per tant:

El programa de control rep les ordres en el format que hem indicat al punt 4.3 i comunicarà les

ordres necessàries al robot per a complir aquestes ordres. Addicionalment, haurà d'informar de si pot o no passar en les unions i informar de si ha realitzat satisfactòriament o no la seva tasca (és a dir, si ha de tornar a circular o pot anar a la zona de recepció a esperar noves ordres). En una aplicació real aquestes últimes accions les hauria de determinar el programa gestor. Aquí hi ha una taula amb les variables que s'usen i per a què en el pseudocodi:

Nom	**Descripció**
state	Indica l'estat en que es troba l'AGV. 0 – Inicialització 1 – Tot correcte 2 – S'ha trobat una marca lateral 3 – No es detecta cap línia 4 – Hi ha un obstacle davant el robot.
position	Posició en el mapa on es troba l'AGV.
bif	Indica si el robot ha pres la última bifurcació o no. Per defecte estarà a 1, que vol dir que és al camí principal (per la dreta).

El pseudocodi és el següent:

```
void Main( ){
        state = 0;
        position = 0;
        bif = 1;
        ConnectBluetooth( );
        While (true){
                state = Receive( );
                switch (state){
                        case 2:  print ( “Band found”);
                                switch (position){
                                        case 0: ReceiveOrders( );
                                                position++;
                                                break;
                                        case 1: Union( );
                                                position++;
                                                break;
                                        case 2: Bifurcation(1);
                                                position++;
                                                break;
                                        case 3: Union( );
                                                position++;
                                                break;
                                        case 4: Bifurcation(2);
                                                position++;
                                                break;
                                        case 5: Union( );
```

```
                                        position++;
                                        break;
                                case 6: EmptyOrders( );
                                        position++;
                                        break;
                                case 7: Bifurcation(3)
                                        if (bif == 1)
                                                position = 0;
                                        else
                                                position = 1;
                                        break;
                                }
                                break;
                        case 3: print('There is no line!');
                                break;
                        case 4: print('There is something in my way!');
                                break;
                }
        }
}
```

Com es pot apreciar, el programa està format per un bucle infinit que llegeix l'estat (assumim la funció *Receive* com una primitiva) en que es troba el robot (ha trobat marca lateral, no troba cap línia o té un obstacle; en cas contrari el robot no informarà al PC), i segons l'estat caldrà fer una cosa o una altra.

En cas de trobar marca lateral, el PC necessitarà saber en quina posició es troba el robot, per això disposa de la variable *position* que s'incrementa cada cop que troba una marca (suposem que l'AGV no pot canviar de posició inesperadament). Així doncs, segons la posició, mitjançant el *switch* seleccionarem què hem de fer. Si la marca és la de la última bifurcació, hem de mirar si recirculem o no, ja que si la ho fem saltem directament a la posició 1 i si no, anem a la posició 0 (rebre ordres).

En cas de trobar un obstacle o de no trobar línia, es mostra el missatge i caldrà que un operari autoritzat solucioni el problema.

4.4.1. ConnectBluetooth

Aquesta funció s'encarregarà de gestionar la connexió Bluetooth a la computadora. Les variables seran:

Nom	Descripció
portID	Variable que serveix d'identificador del port. Aquest variable és de tipus *CommPortIdentifier*.
portserie	Variable que representa el port serie que s'està usant. És de tipus *SerialPort*.
entrada	Aquesta variable serà per la qual el programa de control rebrà les dades enviades pel BoeBot. És de tipus *InputStream*.
sortida	Variable per la qual el programa de control enviarà les comandes que haurà d'executar el robot. Tipus *OutputStream*.

```
void ConnectBluetooth( ){
        try {
                portID = CommPortIdentifier.getPortIdentifier("COM14");
        }catch ("Port not found")
        try{
                portserie = (SerialPort) portID.open("BotController", 5000);
        }catch (“Port in use”)
        try {
                portserie.setSerialPortParams(9600, SerialPort.DATABITS_8,
                                                        SerialPort.STOPBITS_1,
                                                        SerialPort.PARITY_NONE);
        }catch (“Cannot open port”)
        try{
                sortida = portserie.getOutputStream();
                entrada = portserie.getInputStream();
        } catch (IOException)
}
```

Primer, s'ha d'obtenir l'identificador del port pel qual el programa de control i el BoeBot es comunicaran. Com es comentarà en el apartat 5.2.3 (problema amb la connexió Bluetooth), el port adequat és el COM14. Si no obtenim cap identificador, llancem l'error de que no s'ha trobat el port.

Seguidament obrim el port i l'associem a la variable que representarà el port serie amb el nom de l'aplicació (“BotController”) i el timeout (5000, ja que es temps suficient per establir la connexió). En cas de donar error aquí voldria dir que el port ja està obert i per tant una altra aplicació l'està usant.

Ara cal configurar el port amb els valors de velocitat de connexió (9600 bps), el nombre de bits d'informació (SerialPort.DATABITS_8), el nombre de bits de stop (SerialPort.STOPBITS_1) i la paritat (SerialPort.PARITY_NONE). Els valors han de ser aquests, ja que la placa Bluetooth

instal·lada treballa amb aquests paràmetres [9].

Finalment només queda associar les variables d'entrada i sortida que s'usaran per la comunicació amb el port serie.

4.4.2. ReceiveOrders

La funció *ReceiveOrders* serà l'encarregada de llegir del fitxer el llistat d'ubicacions on ha d'anar l'AGV. Tindrà les següents variables:

Nom	**Descripció**
OList	Llista d'ordres. Aquesta llista tindrà el format indicat al punt 4.3.
temp	Llista que s'usarà auxiliarment per processar la llista d'ordres.
Ordres	Llista on són les ordres en el format adequat per al seu tractament. Cada element d'aquesta llista serà un conjunt d'ubicacions opcionals.

El pseudocodi és:

```
void ReceiveOrders( ){
        OList = ReadFromFile( );
        temp = [];
        for (i=0;i<OList.size( );i++){
                if (OList[i] == 9) or (Olist == 0){
                        if (temp != []){
                                Ordres.add(temp);
                                temp = [];
                        }
                }else{
                        temp.add(OList[i]);
                }
        }
        SendRobot(1);
}
```

El resultat d'això es una llista de llistes on cada una de les subllistes són les possibles ubicacions. És a dir, cada llista es una ubicació obligatòria, i cada element d'aquesta subllista és cada una de les possibles ubicacions. L'AGV haurà d'anar a un dels elements de cada subllista. Exemple: si *Ordres = [[1 , 2] , [3]]* vol dir que l'AGV ha d'anar a 1 o a 2, i a 3. Aquest format serà més útil per a treballar.

4.4.3. Union

La funció *Union* s'encarregarà de gestionar les unions amb un semàfor. Les variables necessàries són:

Nom	**Descripció**
semàfor	Indica si el semàfor és lliure (0) o ocupat (1). Només es necessitarà una única variable i no un array amb tots els semàfors, ja que l'AGV només necessita saber l'estat del semàfor en el que es troba, i no de tots ells.
finished	Booleà que indica si l'AGV ha finalitzat o no la unió.
bif	Com s'ha comentat abans, indica si l'AGV ha pres el camí esquerra o el de la dreta en la bifurcació anterior.
cont	S'utilitzarà aquesta variable per controlar el primer cop que s'entra al bucle. Això es necessitarà per saber quants cops s'ha realitzat el loop i, per tant, si ha o no acabat.
opt	Variable que s'usarà per llegir l'estat en que es troba el robot dins la unió, ja que el que cal per saber en quin punt està en la unió és trobar marques laterals.

El pseudocodi serà el següent:

```
void Union( ){
    while (semafor == 1){
        SendRobot(0);
    }
    semafor = 1;
    SendRobot(1);
    boolean finished = false;
    if (bif = 0){
        int cont = 0;
        while (finished == false){
            int opt = Read( );
            if (opt == 2){
                if (cont == 0){
                    cont++;
                    SendRobot(1);
            }else{
                finished = true;
                SendRobot(1);
            }
```

```
            }
        }else{
            while (finished == false){
                int opt = Read( );
                if (opt == 2){
                    finished = true;
                    SendRobot(1);
                }
            }
        }
        semafor = 0;
        bif = 1;
    }
```

El primer que es fa es comprovar si la variable semàfor és a 0 o a 1. Si és a 1, vol dir que la unió està ocupada per un altre AGV, per tant el nostre AGV haurà d'esperar. Un cop estigui el semàfor a 0, el nostre AGV el posa a 1, per a indicar que la unió està ocupada, i que per tant, no pot passar cap altre AGV. En aquest punt és on s'ha de realitzar la consulta i notificació al sistema gestor si es disposa d'ell.

Tot seguit cal fer la distinció entre si en la bifurcació anterior s'ha anat pel camí de la dreta o de l'esquerra (es fa mitjançant la variable *bif*). Aquesta diferència es pot veure a les figures 4.5. i 4.6.

A la figura 4.5. es veu la unió. L'AGV ve per l'esquerra. Quan l'AGV arriba a creuar-se amb el camí principal detecta aquest com a marca (els infrarojos estan representats amb línies vermelles). Així doncs, s'haurà de tenir en compte aquest fet i ignorar la detecció de marca. En canvi, a la figura 4.6. es pot veure que si la pren per la dreta només detectarà les marques inicial i final de unió.

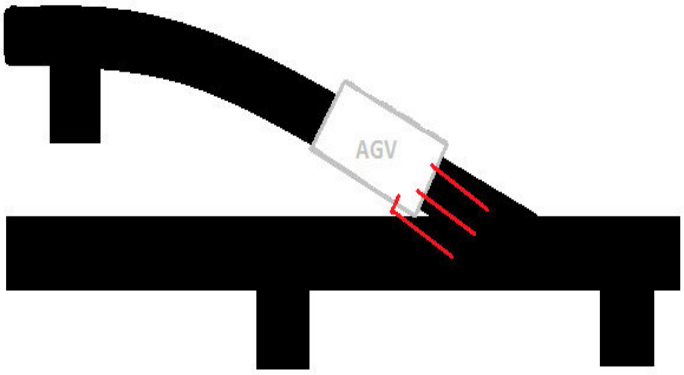


Figura 4.5 Unió per l'esquerra

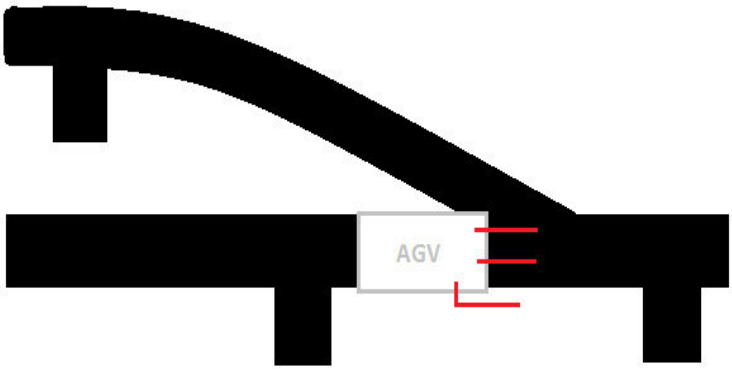


Figura 4.6 Unió per la dreta

4.4.4. Bifurcation

La funció *Bifurcation* s'encarregarà de dir-li al robot si ha de prendre o no una bifurcació. Les variables són:

Nom	**Descripció**
pos	Número de la bifurcació. Coincideix amb el número d'ubicació. S'usarà per saber si el robot ha de sortir o no en la bifurcació. Hi ha 3 posicions (1, 2 i la de recirculació, 3).
trobat	Booleà que servirà per saber si s'ha trobat la posició en la que es troba dins la llista d'ordres. Per tant, si es *true* voldrà dir que el robot ha de sortir.
tmp	Variable auxiliar que s'utilitzarà per a buscar la posició dins les ordres.
OrdresComplertes	Llista d'ordres completades. Si el robot surt per la bifurcació s'haurà d'afegir la posició (pos) a la llista d'ordres completades (excepte si és la posició 3).
bif	Indica si el robot va per la dreta o per l'esquerra en la bifurcació. Si surt haurà de posar-la a 0. Si no, deixar-la a 1.

El pseudocodi és:

```
void bifurcation(int pos){
        trobat = false;
        if (pos == 3){
                if (Ordres == []){
                        trobat = false;
                }else{
```

```
                    trobat = true;
                }
            }else{
                for (int i=0;i<Ordres.size( );i++){
                    ArrayList tmp = Ordres[i];
                    if (tmp.contains(pos)){
                        OrdresComplertes.add(tmp);
                        trobat = true;
                        break;
                    }
                }
            }
            if (trobat){
                SendRobot(2);
                bif = 0;
            }else{
                SendRobot(1);
                bif = 1;
            }
        }
```

Primer cal comprovar si el robot està a la última bifurcació, ja que aquesta té un comportament diferent. Si encara queden ordres per complir el robot ha d'anar a la posició 1 sense passar per la 0 (recollida d'ordres), per tant posarem la variable *trobat* a *true* o *false* segons calgui. En canvi, si està en una altra bifurcació, ha de mirar en quina posició està i comprovar si ha de prendre o no la bifurcació.

Finalment, si *trobat = true* posem *bif* a 0 (prenem la bifurcació) i enviem al robot un 2 (girar a l'esquerra en la bifurcació).

4.4.5. EmptyOrders

Finalment, la funció *EmptyOrders* s'encarregarà d'esborrar de la llista d'ordres les ordres que estiguin completades.

Nom	**Descripció**
Ordres	Llista de llistes d'ordres.
OrdresComplertes	Llista d'ordres completades.
k	Variable de bucle

El pseudocodi és:

```
void EmptyOrders( ){
        for (int k=0;k<OrdresComplertes.size( );k++){
                Ordres.remove(OrdresComplertes[k]);
        }
        OrdresComplertes.removeAll( );
}
```

El que es fa es recórrer la llista d'ordres completades i esborrant-les de la llista d'ordres original. Un cop fet això, es tindrà la llista d'ordres que encara no s'han visitat. A més, s'esborra per complet la llista d'ordres completades, ja que aquesta ha de contenir les ordres de la llista Ordres que estiguin ja completades. És a dir, tot element de la llista d'ordres completades ha d'estar, necessàriament a la llista d'ordres principal.

4.5. Programa del robot

Aquest programa rebrà mitjançant el Bluetooth les ordres del programa de control i les transmetrà directament als actuadors. Les variables del bucle principal seran:

Nom	**Descripció**
state	Indica l'estat en que es troba el robot.
dist	És la distància a l'objecte més proper. Cal destacar que els ultrasons calculen el temps que triga l'ultrasó en rebotar i tornar. Per això, cal fer un petit càlcul per trobar la distància en centímetres.
Band	Per detectar amb IR només es disposa d'un bit, i la detecció de marca lateral és molt important per a que el robot sàpiga situar-se en el mapa. Per això, es decideix no només realitzar una lectura i aplicar un filtre a aquestes lectures.
BandArray	Array de bits que s'usarà per al filtre de marca lateral.

El pseudocodi serà el següent:

```
void Main( ){
        state = 0;
        Band = 0;
        BandArray = '0000'
        ConnectarBluetooth ( );
        while (true){
```

```
                DetectarUltraSons( );
                if (dist > 15){
                        DetectarIR( );
                        BandFilter( );
                        FollowLine( );
                }else{
                        state = 4;
                        SendPC(state);
                }
        }
}
```

La variable *state* indica l'estat en que es troba el robot:

Valor	Significat
0	Inicialització
1	Tot correcte
2	Marca lateral trobada
3	No detecta cap línia a seguir
4	Hi ha un obstacle en el camí

4.5.1. ConectarBluetooth

La funció *ConnectarBluetooth* serà:

```
void ConnectarBluetooth( ){
        SendPC(MAC_Add);
        While (IN5 = 0){
                wait( );
        }
}
```

Nom	Descripció
MAC_Addr	La adreça MAC de la placa del robot. Segons el protocol Bluetooth, la placa que es vol conectar ha d'enviar la petició amb la seva MAC.
IN5	És el pin de la placa que es posa a 1 quan la connexió s'ha realitzat correctament.

Primerament s'envia la MAC Address de la placa del robot, per a realitzar una petició de connexió amb el PC, després, cal esperar a que el IN5 (el pin 5 de la placa) s'activi, el que indicarà que la placa s'ha conectat amb el PC satisfactoriament [9].

4.5.2. DetectarUltrasons

La funció *DetectarUltrasons* es basa en enviar un pols a un pin, recollir-lo i calcular la distància, ja que l'ultrasó donarà el temps de viatje de la senyal, no la distància que ha recorregut [11].

Nom	**Descripció**
pin	El pin d'input de la placa d'ultrasons. És per on s'haurà d'enviar el pols.
dist	Variable on primerament s'emmagatzema el temps i després d'un càlcul s'obté la distància a l'objecte més proper.

```
void DetectarUltrasons( ){
        EnviarPols(pin);
        dist = RecollirResposta( );
        dist = dist**2251
}
```

4.5.3. DetectarIR

DetectarIR funciona d'una manera similar, s'envia un senyal d'una freqüència a un pin, i es recull la dada per un altre pin [10]. En aquest cas donarà un bit, si detecta o no detecta i ho s'ha de fer per cada un dels 3 IR. Les variables usades són:

Nom	**Descripció**
freq	Freqüència a la que s'emetrà el senyal.
pinLeftSortida	Pin per el qual s'envia el senyal a l'IR situat a l'esquerra.
pinLeftEntrada	Pin per on es recull el resultat de l'IR esquerra: 0 – No detectat 1 – Detectat
pinRightSortida	Pin per el qual s'envia el senyal a l'IR de la dreta.
pinRightEntrada	Pin per on es recull el resultat de l'IR dret: 0 – No detectat 1 – Detectat
pinBandSortida	Pin per el qual s'envia el senyal a l'IR lateral.
pinBandEntrada	Pin per el qual es recull el resultat de l'IR lateral: 0 – No detectat 1 – Detectat

I el pseudocodi:

```
void DetectarIR( ){
        EnviarFreq(freq, pinLeftSortida);
        IRLeft = pinLeftEntrada;
        EnviarFreq(freq, pinRightSortida);
        IRRight = pinRightEntrada;
        EnviarFreq(freq, pinBandSortida);
        IRBand = pinBandEntrada;
}
```

4.5.4. BandFilter

Aquesta funció aplicarà un filtre a les lectures de IR. Això s'aplica degut a la importància de detectar perfectament les marques laterals. La il·luminació és molt variable i qualsevol canvi afecta a les deteccions dels IR i amb només un bit d'informació no es pot saber si és una lectura errònia o si realment és una marca. Per tant, eventualment, els errors causarien que l'AGV no sapigues on és i provocaria un mal funcionament del conjunt.

Segons les especificacions, les marques fan 2 cm d'ample, i empíricament s'ha comprovat que en un cicle del programa principal es fan 7 lectures als IR. Per tant, disposem d'un array de bits que actuaran com a finestra de la manera següent:

Figura 4.7 Finestra de detecció

Com s'observa, a mesura que es realitzen lectures, l'array es va omplint amb les lectures y quan els 4 bits siguin tots 1, el robot podrà afirmar que és una marca. S'han triat 4 bits per donar un marge d'error. De la mateixa manera que si no hi ha marca es pot detectar eventualment que n'hi ha, també es pot donar el cas que tot i havent-hi marca no es detectada. Per tant, s'ha considerat empíricament que 4 bits és una mesura addient per a poder detectar marca tot i podent-hi haver encara algún error.

No obstant, com s'ha dit prèviament, el problema de la il·luminació és molt difícil de resoldre. Per tant, en un cas real tenim dues opcions:

- Controlar totalment la llum. Això vol dir crear un entorn amb llum artificial, per a que sempre sigui la mateixa il·luminació. L'avantatge d'aquest métode és que no cal preocupar-se de la il·luminació i, per tant, serà més ràpid el procés. Per contra, s'ha de controlar la il·luminació totalment.
- Realitzar una rutina d'adaptació a l'entorn. Aquesta rutina s'hauria d'executar cada cert nombre de cicles per assegurar-nos que la detecció és correcta. Amb aquest métode no cal controlar totalment la llum, i una rutina d'adaptació permet facilitar la configuració del robot. No obstant, aquesta adaptació consumeix temps cada cop que s'executa.

En aquest projecte s'ha optat per controlar la llum artificialment, ja que un cop tot configurat i preparat, és més ràpid.

Les variables usades en la funció són les següents:

Nom	**Descripció**
BandArray	Array de bits que emmagatzemen les lectures del IR dedicat a la detecció de marques.
Band	Variable que es posarà a 1 quan el robot consideri que ha trobat una marca.
counter	Variable per als bucles *for*.
IRBand	Valor que s'ha detectat amb la funció *DetectarIR*, concretament a l'IR del lateral.

El pseudocodi de la funció és el següent:

```
void BandFilter( ){
        Shift(BandArray);
        BandArray(0) = IRBand;
        Band = 1;
        for counter = 0 to 3 {
                if (BandArray(counter) = 0){
                        Band = 0
                }
        }
        if (Band = 1){
                for counter = 0 to 3
                        BandArray(counter) = 0
```

```
                    }
                }
            }
```

Com s'observa, primer cal fer el *shift* del conjunt de bits *BandArray*. Això vol dir que els bits es desplaçaran una posició. Es considera per al pseudocodi que el *shift* afegeix a la primera posició un 0, per tant cal posar a la posició 0 el valor que s'hagi detectat amb la funció *DetectarIR*.

Seguidament, es posa la variable *Band* a 1 i es comprova si realment tots els bits de l'array són 1. Si és així, la variable *Band* ja és activada. En canvi, si es troba algun que no ho és, torna a 0.

Finalment, si s'ha detectat una marca, cal tornar tot l'array a 0. Això es realitza perque s'ha d'informar al PC de que s'ha trobat una marca, però si no es buida l'array, i segueix llegint marca, tornarà a enviar que ha detectat marca quan realment és la mateixa marca. Això es pot veure millor a la figura 4.8, on tenim el pols de detecció de marca. Quan llegeix 4 cops (línies blaves) es considera detecció de marca i s'envia a la computadora (línia vermella). Sense netejar l'array, cada cop que torna a detectar, torna a enviar, mentre que amb neteja, no torna a enviar.

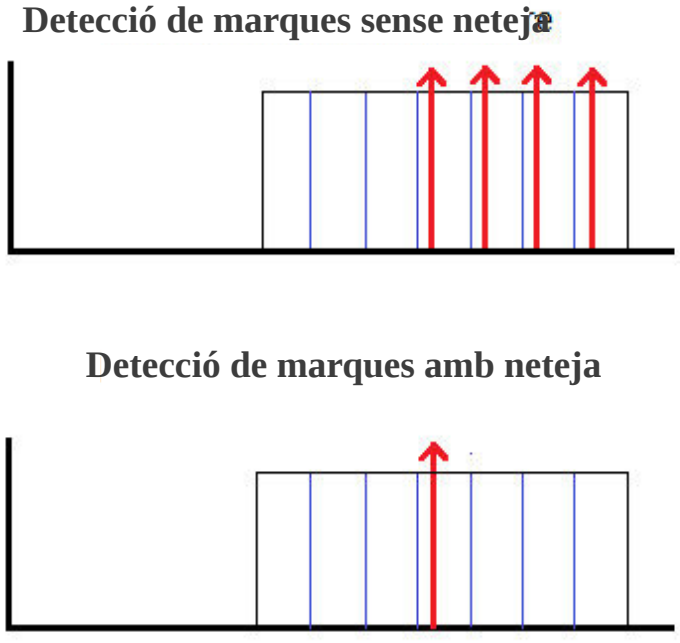


Figura 4.8 Gràfic de detecció de marca

4.5.5. FollowLine

La funció *FollowLine* serà l'encarregada de enviar els polsos corresponents a les rodes per al correcte seguiment de la línia. A més, també enviarà per Bluetooth un missatge al programa de

control si cal. Les variables són:

Nom	**Descripció**
Band	Valor que indica si s'ha detectat marca o no. Aquest és el resultat del filtre de lectures.
IRRight	Valor detectat pel receptor dret. Indica si ha detectat línia (1) o no (0).
IRLeft	Valor detectat pel receptor esquerra. Indica si ha detectat línia (1) o no (0).
pinRodaDreta	Pin pel qual s'envia el pols per a fer que funcioni la roda dreta.
pinRodaEsquerra	Pin pel qual s'envia el pols per a fer que funcioni la roda esquerra.
state	Indica l'estat en que es troba l'AGV.
answer	És la resposta que envia el programa de control i es rep per Bluetooth. 0 – Espera 1 – Avança 2 – Prendre la bifurcació cap a l'esquerra.

I el pseudocodi:

```
void FollowLine( ){
    if (Band = 0){
        if (IRRight = 1) and (IRLeft = 1){
            Pols(pinRodaDreta);
            Pols(pinRodaEsquerra);
        }elseif (IRRight = 1) and (IRLeft = 0){
            Pols(pinRodaEsquerra);
        }elseif (IRRight = 0) and (IRLeft = 1){
            Pols(pinRodaDreta);
        }else{
            state = 3;
            SendPC(state);
        }
    }else{
        state = 2;
        SendPC(state);
        answer = 0;
        while (answer = 0){
            answer = ReceivePC( );
        }
        if (answer = 2){
            GoBifLeft( );
        }
```

```
        }
}
```

Primerament es comprova si hem llegit una marca o no. Si no l'ha llegit, cal mirar els valors dels IR i segons si s'ha detectat un o altre costat, o els dos, s'activarà una, altra o les dues rodes. Si no es detecta cap dels dos, s'envia un missatge al programa de control informant-ne.

Si s'ha detectat marca, el robot informa al programa de control i espera una resposta. Les respostes possibles són 3. Si es rep 0, el robot es para esperant que li enviin una altra resposta diferent de 0. Si la resposta es 1 (en el programa no està especificat), saltarà a la següent iteració, i per tant, seguirà avançant. Finalment, si la resposta és 2, vol dir que l'event trobat és una bifurcació, i que el robot l'ha de prendre girant a l'esquerra. Per tant, s'anirà a la rutina *GoBifLeft*.

4.5.6. GoBifLeft

Finalment, la funció *GoBifLeft* servirà per prendre una bifurcació cap a l'esquerra (cap a la dreta significarà seguir recte).

Nom	**Descripció**
IRLeft	Variable que indica si es detecta línia amb el receptor esquerra.
IRBand	Variable que indica si es detecta marca lateral. En aquest cas no ens cal el filtre.

El pseudocodi és el següent:

```
void GoBifLeft( ){
        While (IRLeft == 1){
                PolsRodaDreta( );
                IRDetection( );
        }
        While (IRBand == 0){
                PolsRodaDreta( );
                PolsRodaEsquerra( );
                IRDetection( );
        }
        While (IRBand == 1){
                PolsRodaDreta( );
                PolsRodaEsquerra( );
                IRDetection( );
        }
}
```

Un cop el robot arriba a la marca que indica que l'event és una bifurcació, primer ha de girar fins que no detecti línia a l'esquerra. En aquest punt, el robot ha d'avançar mentre que no detecti marca (entre la marca i el camí principal) i després avançar mentre detecti el camí principal. Un cop ja no detecti el camí principal, ja pot seguir la línia amb normalitat. En la figura 4.9 es pot veure gràficament.

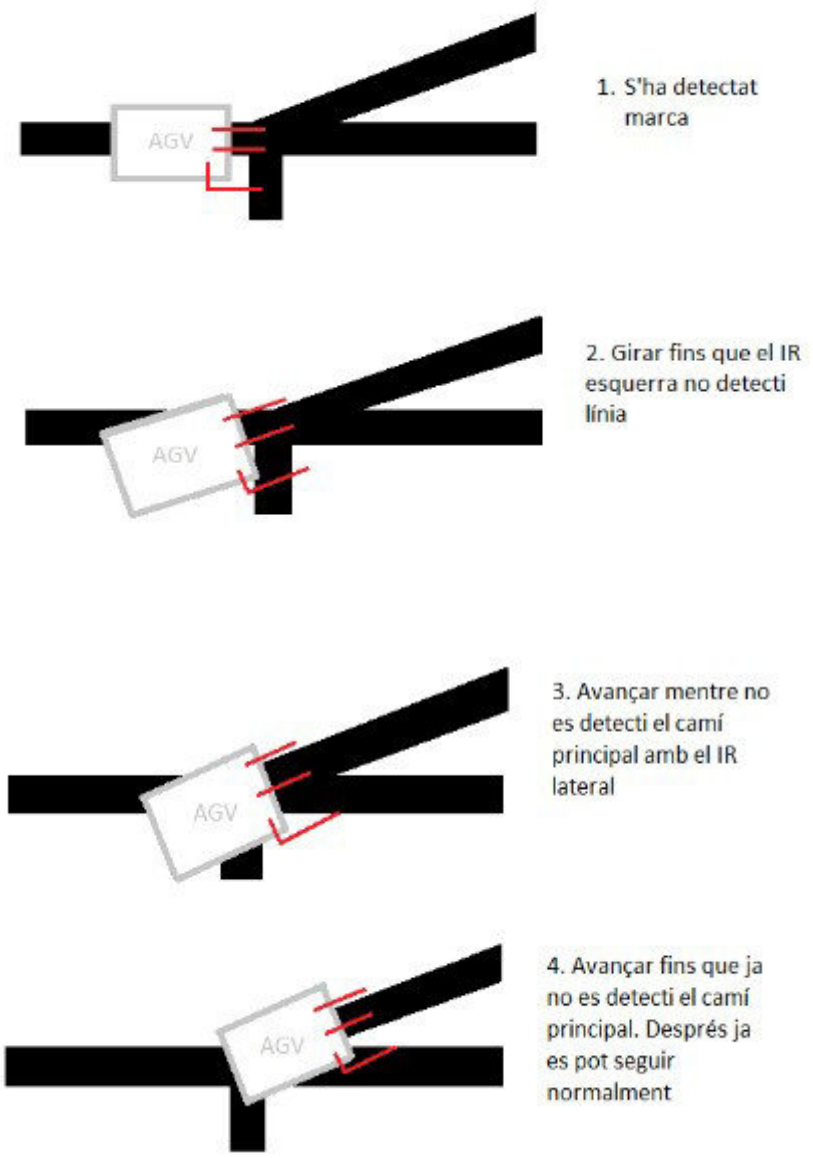


Figura 4.9 Procés d'una bifurcació

4.6. Banc de proves

Per tal de provar el correcte funcionament de l'AGV s'han dissenyat algunes proves. Aquesta tasca és molt important per a detectar possibles errors de disseny dels algorismes, refinar la configuració

del robot, etc. Tot seguit es detallen aquests tests.

4.6.1. Test Case 1. Funcionalitats bàsiques

En aquesta cas es comprova el correcte funcionament del BoeBot i les seves funcionalitats bàsiques. Per tant, no disposa de connexió Bluetooth ni de cap llista d'ordres. S'espera doncs, que el BoeBot segueixi el circuit exterior sense perdre's.

4.6.2. Test Case 2. Detecció d'obstacles

Ara que ja s'ha comprobat el funcionament bàsic del seguidor de línies, cal comprobar la detecció d'objectes amb els ultrasons. Per a això es realitzarà la mateixa proba anterior, pero amb un objecte al recorregut. Per tant, el BoeBot haurà de seguir la línia i quan tingui un objecte davant, parar-se. Quan l'objecte desaparegui, haurà de continuar el seu recorregut.

4.6.3. Test Case 3. Connexió Bluetooth

Un cop s'ha realitzat correctament el seguiment de línies i la detecció d'obstacles, es comprova la connexió Bluetooth, l'enviament i recepció de dades. Com l'objectiu d'aquest *test case* és comprobar la connexió Bluetooth, la llista d'ordres és buida. El resultat que s'espera es que es connectin ambdós dispositius i el robot realitzi el recorregut exterior del circuit informant dels events que troba. Cal destacar que no totes les marques són printades a pantalla, ja que es considera tota una unió com a una posició sencera. Per tant, la marca de sortida no realitzara el print per pantalla.

4.6.4. Test Case 4. Bifurcacions i unions

Un cop ja es tenen realitzats correctament els casos anteriors, ja es pot comprobar el funcionament de les bifurcacions i unions. Per a això, aquest cop, la llista d'ordres serà 9-1-0, és a dir, el robot ha de sortir a la primera bifurcació i tornar al camí principal.

4.6.5. Test Case 5. Recirculacions

En aquest cas es comprovarà que l'AGV prengui la última bifurcació per a recircular. La llista d'ordres és 9-1-9-1-0. Això vol dir que l'AGV ha de prendre la primera bifurcació, tornar al camí principal i quan arribi la última, adonar-se que falten ordres per cumplir i recircular i per tant, saltar-

se la posició 0 (rebuda d'ordres).

4.6.6. Test Case 6. Ubicacions optatives

Ara cal comprovar que les ubicacions optatives funcionen correctament. Com s'ha explicat anteriorment, la política que s'ha triat és la d'anar a la primera destinació buida que troba. En aquest projecte serà sempre la primera de la llista, ja que circula sol. Així doncs, la llista d'ordres que es donarà serà 9-1-2-0 i el resultat que s'espera es que surti per la primera bifurcació.

4.6.7. Test Case 7. Lógica de control

Ara només queda comprobar la lógica del programa de control. Si la realitza correctament es pot assumir que l'aplicació és correcte, ja que s'hauran probat totes les accions que pot realitzar l'AGV. Això no vol dir que aquestes siguin totes les possibles combinacions, ja que la llista d'ordres pot tenir elements repetits (i per tant hauria de recircular un i altre cop) però si ha passat el Test Case 5 es pot suposar que l'AGV ho farà correctament.

Així doncs, en aquest *test case* es realitzaran dos proves:

1. Es provarà que l'AGV prengui totes dues bifurcacions. La llista d'ordres serà 9-1-9-2-0.
2. Es provarà que en cas de tenir ordres repetides dins d'un grup de ubicacions optatives l'AGV se n'adoni i les ignori. És a dir, si tenim [[1,2,1]], l'AGV haurà d'anar a 1 un copi no recircular.

Capítol 5. Resultats

5.1. Explicació de resultats

Com s'ha vist en el capítol anterior, s'han dissenyat unes proves per a comprobar el funcionament. Totes elles estan incloses en un video que s'ha publicat a *YouTube*. L'enllaç és el següent:

http://www.youtube.com/watch?v=v-pXOvKMdLw&feature=plcp

Com es pot observar al video, a partir del test case 3, per a poder apreciar la connexió entre el BoeBot i la computadora, s'ha inclós, a més de la vista del BoeBot, una vista de la pantalla de la computadora on apareixen els prints del programa de control, l'estat de l'AGV, missatges d'error, etc.

Així doncs, tot i que a vegades apareix el missatge *There is no line to follow* (No hi ha línia a seguir), són errors de lectura dels IR esporàdics, que no cal tenir en compte, ja que se solucionen ràpidament i el resultat final és correcte com es pot veure al video dels *test case*.

Tot seguit s'adjunten algunes imatges de l'AGV realitzant algunes de les accions dels diferents tests.

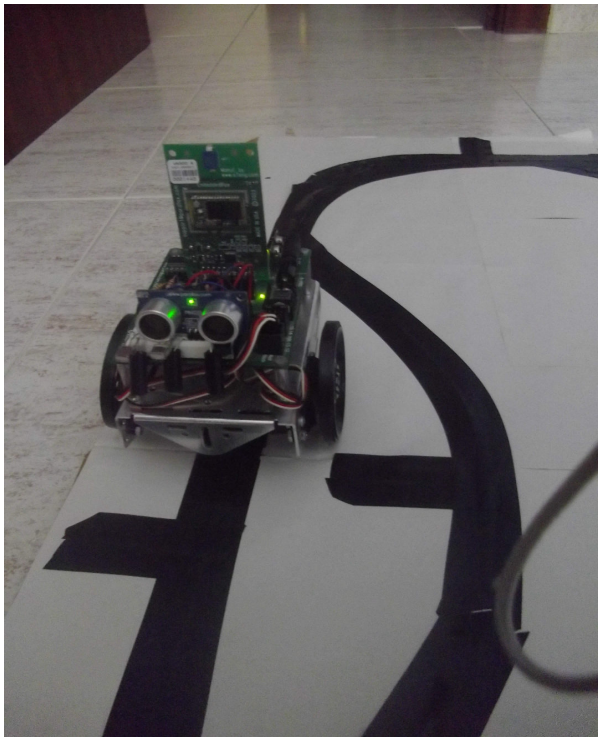


Figura 5.1. AGV circulant pel circuit exterior (Test Case 1)

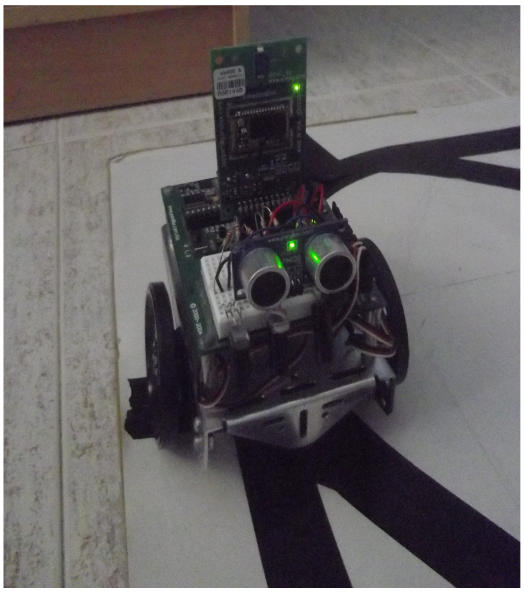


Figura 5.2. BoeBot amb Bluetooth circulant (Test Case 3)

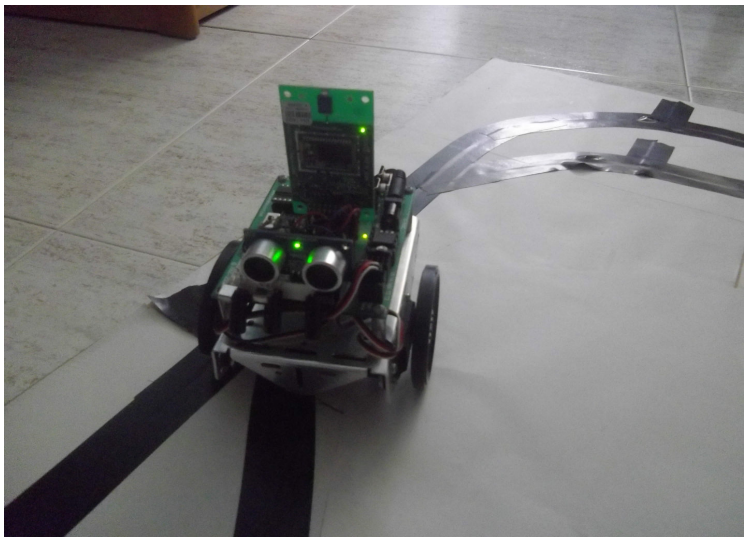


Figura 5.3. AGV Realitzant una bifurcació (Test Case 4)

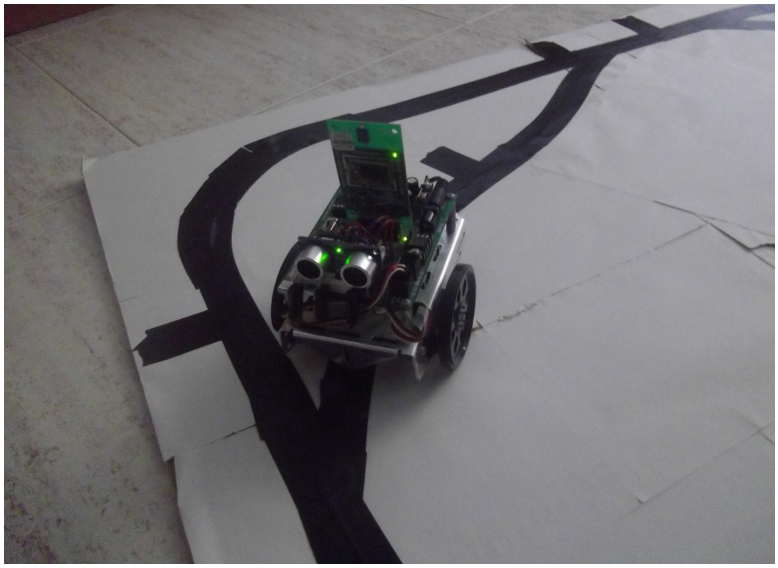


Figura 5.4. AGV realitzant una unió (Test Case 4)

5.2. Problemes trobats

5.2.1. Detecció de línia amb infrarojos

Com s'ha comentat en capítols anteriors, la il·luminació és un problema molt complexe ja que la llum natural varia en funció de la hora del dia o la climatologia. Tot i estar en un entorn tancat, la llum que entra per una finestra influeix molt en el comportament dels infrarojos. Així doncs, per solucionar això, s'ha hagut de tenir en compte diversos aspectes. El primer és intentar que la llum estigui sempre en uns llindars similars per a que no hi hagi gaires canvis.

Amb il·luminació controlada, ara ens cal que detecti correctament quan hi ha línia (detectarà un 1) i quan no (un 0). Per això, s'ha hagut de buscar empíricament, una combinació entre el material base, el material de la cinta que detectarà, la freqüència a la que s'emet la llum i els valors de les resistències que hi ha la placa. El valor d'aquesta resistència influeix directament en el pols que s'envia a l'emissor IR, i per tant, en la detecció.

Així doncs, les característiques finals que han solucionat el problema han estat:

- Material base: Paper DIN-A4.
- Material cinta: cinta adhesiva antielèctrica.
- Freqüència de IR esquerra: 42.700 Hz
- Freqüència de IR dret: 38.000 Hz
- Freqüència de IR lateral: 35.500 Hz
- Resistència 1 de IR esquerra: 220 Ohms (tolerància 5%)
- Resistència 2 de IR esquerra: 10 KOhms (tolerància 5%)
- Resistència 1 de IR dret: 220 Ohms (tolerància 5%)
- Resistència 2 de IR dret: 10 KOhms (tolerància 5%)
- Resistència 1 de IR lateral: 220 Ohms (tolerància 5%)
- Resistència 2 de IR lateral: 2 Kohms (tolerància 5%)

5.2.2. Detecció de marques

Inicialment, la detecció de marca es feia de la mateixa manera que el seguiment de línies, pero de seguida s'observa que aquesta opció és inviable, ja que una iteració del programa del robot és més ràpida que el que triga en avançar i superar la marca. És a dir, en el temps que triga en creuar la

marca, fa més d'una lectura dels IR. Tot i que només realitzés una lectura, tampoc sería una bona solució, ja que com s'ha comentat en el apartat 5.2.1, la il·luminació comporta molts problemes i pot sovint haver-hi lectures errònies.

La solució que s'ha optat per implementar és la vista a l'apartat 4.5.4, que consisteix en realitzar varies lectures i només considerar que s'ha detectar marca si es llegeixen varis valors de detecció (1) seguits. Això ens permet evitar un gran percentatge de lectures errònies.

5.2.3. Connexió Bluetooth

Quan s'intentava connectar l'aplicació Java amb la placa del BoeBot va sorgir el problema de la connexió. Concretament, el problema era que alguns cops connectava, i d'altres cops no.

Després d'investigar el problema, es va aconseguir determinar que era degut a la assignació de ports COM que realitzava la computadora. És a dir, a l'hora d'identificar el primer cop la placa se li assignava un COM i si després es desconnectava i tornava a connectar el port COM era reassignat amb un altre nombre. També passava si es connectaven altres dispositius. Per a aquest problema es van idear dues solucions possibles:

- Ja que es coneix la adreça MAC de la placa es pot buscar per tots els ports disponibles quin d'aquests dispositius té la MAC en qüestió.
- Realitzar un protocol a seguir cada cop que es treballarà amb la placa per a que sempre sigui el mateix COM, ja que els assigna sempre de la mateixa forma.

S'ha triat realitzar la segona opció, ja que la primera, tot i ser la ideal, arriba a trigar molta estona en buscar el port corresponent. Per tant, és una càrrega de temps important a l'hora de fer proves amb el BoeBot. El protocol que s'ha de seguir és:

1. Engegar la computadora.
2. Connectar l'adaptador USB Bluetooth.
3. Llançar l'aplicació Java.
4. Engegar el BoeBot. Aquest pas s'ha de realitzar ràpid, abans de que s'esgoti el time-out de connexió de l'aplicació Java.

Amb aquests passos, el COM assignat és sempre el número 14 i, per tant, podem estalviar-nos una rutina de *polling* a tots els dispositius.

Capítol 6. Conclusions

6.1. Procediment

A la indústria actual, el temps és un factor vital per a reduir costos i la logística interna d'una planta de producció (entre d'altres) no és una excepció. Així doncs, és una bona pràctica automatitzar aquest transport intern de material. Amb això, es pot aconseguir per exemple que una màquina estigui menys temps en espera de material, el que dona una major productivitat.

Una bona solució és la d'implantar un sistema d'AGV a la planta, un grup de vehicles que transportaran el material d'un punt a un altre segons calgui. Aquestes ordres seran introduïdes en un fitxer de text amb un enter per línia. El 9 indica que els enters posteriors seran un conjunt d'ubicacions optatives de les quals l'AGV haurà de triar una. El 0 indica fi d'ordres.

Atès que el BoeBot té una capacitat de procés i memòria petites una bona opció és la de dividir l'AGV en dues parts: un robot i una computadora. El primer s'encarregarà exclusivament del moviment per l'entorn i el segon s'ocuparà del control, on ha d'anar, si ha de prendre les bifurcacions, etc. Això implica que es necessita necessàriament comunicar ambdós dispositius. Per a això s'ha triat el Bluetooth. La computadora rebrà l'estat del robot i enviarà què ha de fer aquest.

Pel que fa a l'entorn físic, s'ha triat el seguidor de línies per la seva senzillesa i fiabilitat. Aquest model ens obliga a utilitzar un mecanisme per a detectar la línia. Aquest mecanisme consisteix en 2 infrarojos per la línia principal i un tercer infraroig per a detectar unes marques laterals col·locades per a informar al robot de que en aquell punt hi ha algun *event*, ja sigui una bifurcació, una unió, rebuda d'ordres o neteja d'ordres completades.

Els AGV, tot i que en aquest projecte només s'ha treballat amb un, estan pensats per a circular entre ells, per tant, es totalment necessari evitar possibles col·lisions. S'ha decidit per instal·lar un emissor d'ultrasons, i així aturar el BoeBot quan hi hagi algun objecte davant seu a poca distància.

Finalment, s'ha dissenyat un conjunt de proves per a testar diferents casos en els quals es pot trobar l'AGV.

Amb tot això, aquest projecte no pot funcionar realment en una planta perquè el BoeBot és un robot amb objectius educatius, però pot servir perfectament de base per a implantacions a més gran escala modificant el mapa (en aquest projecte s'ha utilitzat un model simple per a provar el funcionament) i implementant un programa gestor (i les seves respectives funcions de comunicació al programa de control) per a gestionar no només un AGV, sinó un conjunt més gran de vehicles, destinacions, unions i bifurcacions.

6.2. Temps de treball

La figura 6.1 és la planificació inicial (veure punt 1.6) i la figura 6.2 és el diagrama real de treball. Com es pot observar s'han allargat les Tasques 2, 3 i 4 que són les corresponents a la implementació del programa de control i la connexió Bluetooth (concretament, la detecció de línies, les marques i el Bluetooth. Problemes comentats a l'apartat 5.2). Totes les altres tasques han estat correctament planificades amb un marge d'error petit. Aquest allargament ha provocat que el projecte s'allargués, motiu pel cual ha estat una bona decisió reservar el 20% del temps del projecte a possibles imprevistos.

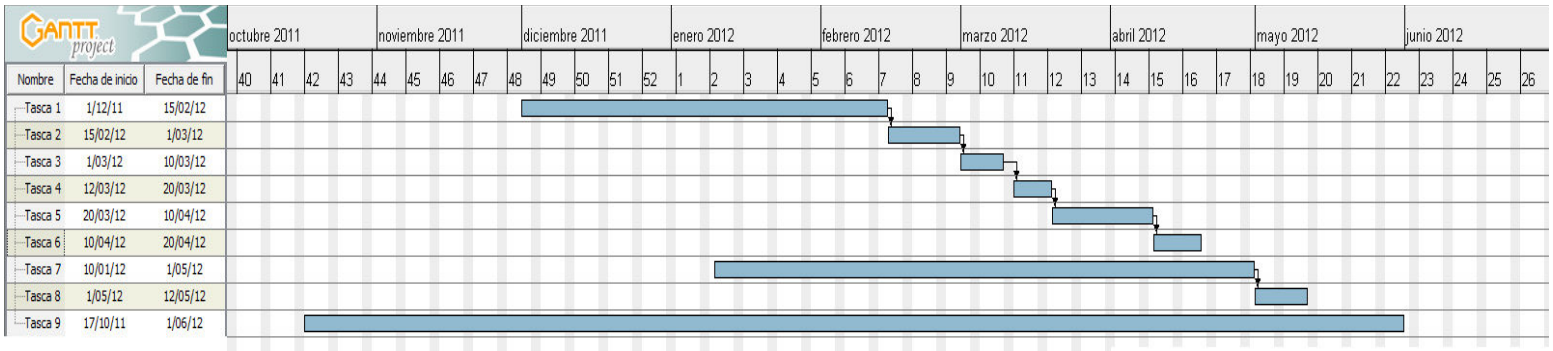


Figura 6.1. Planificació original

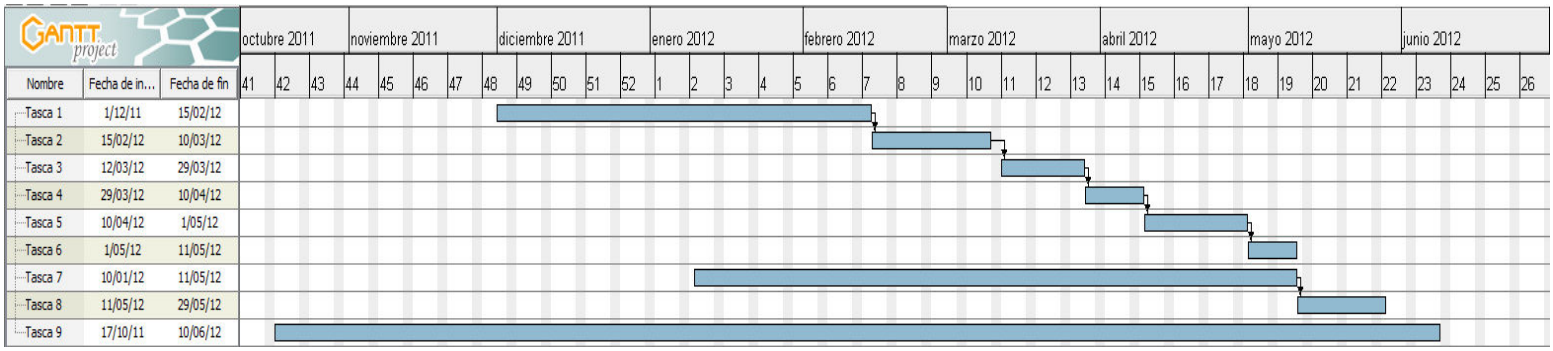


Figura 6.2. Treball real

Annex

Juntament amb aquest document s'adjunta en el CD una carpeta amb:

1. Carpeta *Gantt Diagram*: Conté els diagrames de Gantt.
    - diagrama.png: Diagrama de la planificació original.
    - DiagramaReal.png: Diagrama real de la planificació.
2. Carpeta *Boe Bot*: Conté tot el codi del programa.
    - JavaBotController: Conté el codi Java.
    - PBasic: conté el codi Pbasic.
    - OrdresAGV.txt: fitxer d'ordres de l'AGV.
3. Carpeta *PFC Memoria*: Conté aquest document en format electrònic.
4. Carpeta *PFC Video*: Conté el video final dels resultats de les proves. És el mateix que hi ha penjat a YouTube.

Referències

[1] Ll. Ribas-Xirgo and A. Miró-Vicente, “Multi-Agent Model of a Sample Transport System for Modular IVD Laboratories”, unpublished, from 2010/2011 course on Multi-Physical Agent Systems, Univ. Autònoma de Barcelona (UAB), May 2011.

[2] SoftDesign, http://www.softdesign.se/agv/, consultada el 10 de Desembre de 2011.

[3] “Principles of Computer-Integrated Manufacturing”, S. K.Vajpayee and Prentice Hall, published in 1995.

[4] JBT Corporation, http://www.jbtc-agv.fr/index_es.htm, consultada el 9 de Desembre del 2011.

[5] Savant Automation, http://www.savantautomation.com, consultada el 9 de Desembre de 2011.

[6] Toshiba,”Serie Toshiba NB500”, http://es.computers.toshiba-europe.com/innovation/series/Serie-Toshiba-NB500/1098375/, consultada el 25 de Novembre del 2012

[7] Parallax, “BASIC Stamp Software Downloads”, http://www.parallax.com/ProductInfo/Microcontrollers/BASICStampSoftware/tabid/441/Default.aspx, consultada el 15 de Desembre del 2012.

[8] Netbeans, “Descargar NetBeans IDE 7.1.1”, http://netbeans.org/downloads/7.1.1/, consultada el 15 de Desembre del 2012.

[9] A7 Engineering, “EmbeddedBlue eb500-SER”, published in 2007.

[10] Parallax Inc., “Robotics with the Boe-Bot version 3.0”, Andy Lindsay for Parallax Inc, published in 2004.

[11] Parallax Inc, “PING))) Ultrasonic Distance Sensor”, consultat el 8 de Gener del 2012.

[12] Parallax Inc, “BASIC Stamp Syntax and Reference Manual version 2.2”, Jeff Main, Jon Williams, Ke Gracey, Aristides Alvarez and Stephanie Lindsay, consultat el 15 de Gener del 2012.

[13] Alberto Arenas Toledo, Lluís Ribas Xirgo (director). *Sistema multi-agent físic per a la inspecció de superfícies*, Projecte de Fi de Carrera de la Titulació d'Enginyeria Informàtica de la UAB, Escola d'Enginyeria de la UAB. Setembre de 2010.

Resum

Una de les solucions per a minimitzar els costos de producció o d'emmagatzematge és automatitzar-ne la logística interna. Per dissenyar les aplicacions corresponents és convenient poder validar-les amb un prototipatge. Això ha motivat la realització d'aquest projecte, on s'ha obtingut un prototip d'AGV (*automated guided vehicle*) que rep les ordres per Bluetooth i, mitjançant uns infrarojos per poder seguir unes línies al terra, és capaç d'anar des del punt inicial fins on se li ha encarregat i tornar al punt inicial. Aquest vehicle pot servir de base per a la implementació d'AGV orientats a aplicacions reals i, per tant, per a la construcció de sistemes més grans i que poden ser utilitzats en plantes de producció, laboratoris, magatzems, ports i d'altres aplicacions diverses.

Resumen

Una de las soluciones para minimizar los costes de producción o de almacenaje consiste en automatizar su logística interna. Para diseñar las aplicaciones correspondientes es conveniente poder validarlas con prototipos. Esto ha motivado la realización de este proyecto, donde se ha obtenido un prototipo de AGV (*automated guided vehicle*) que recibe las órdenes por Bluetooth y, mediante unos infrarojos para poder seguir lineas en el suelo, es capaz de ir desde el punto inicial hasta donde se le haya encargado y volver al punto inicial. Este vehículo puede servir de base para la implementación de AGV orientados a aplicaciones reales y, por lo tanto, para la construcción de sistemas más grandes y que pueden ser usados en plantas de producción, laboratorios, almacenes, puertos y un largo etcétera.

Abstract

A possible solution to minimize production or warehousing costs relies on automating internal logistics. To design the corresponding applications it is required to have them validated through prototyping. This fact has motivated the realization of this project, in which a prototype of an AGV (automated guided vehicle) has been obtained. This AGV receives the orders through Bluetooth and, with infrared LEDs to follow lines on the floor, it is able to go from some initial point to some target destination and then come back to the initial point. This vehicle can be used to implement full application-oriented AGVs and, thus, to construct larger systems that can be used in manufacturing plants, automated laboratories, warehouses, ports and so on.