Ampliació i millora d'un vehicle teledirigit

Memòria del Projecte Final de Carrera
d'Enginyeria Tècnica en Telecomunicacions
Especialitat en Sistemes Electrònics

realitzat per
Xavier Linares Dalmau
i dirigit per
Leonardo Portaña Elettrico

Bellaterra, 22 de Juny de 2010

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, ***Leonardo Portaña Elettrico***

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 ***Xavier Linares Dalmau***.

I per tal que consti firma la present.

Signat: .....

Bellaterra, ...de.....de 2009

Índex

Índex de figures

Índex de taules

1. Introducció

El projecte exposat a continuació es situa dins l'àmbit del ràdio control i parteix de la base d'un prototip de vehicle teledirigit dissenyat en un projecte anterior. La proposta que engega aquest projecte és la de donar una certa continuïtat a aquest projecte anterior, que per les seves característiques disposa d'un ventall molt ampli d'aspectes que es poden ampliar o millorar.

En una primera instància no es defineixen directament els objectius d'aplicació al vehicle, més aviat es vol fer un anàlisi del prototip que ens trobem i fer una pluja d'idees dels aspectes a millorar. Tot això tenint en compte els recursos disponibles per a implementar-ho.

Un dels recursos més significatius emprats en el desenvolupament d'aquest projecte són les plaques Duemilanove d'Arduino, que es fan servir per a controlar els dos dispositius principals del sistema: El comandament i el vehicle. El tret més característic d'aquestes plaques és que es fonamenten en el hardware lliure, cosa que resulta molt interessant d'estudiar i de dur a la pràctica.

També comptem amb un acceleròmetre ADXL 335, que en comptes de mesurar les acceleracions instantànies dels eixos de l'espai en mesura el seu angle d'inclinació. Ens és de gran utilitat per a realitzar una de les ampliacions escollides, manejar el vehicle amb el moviment del comandament.

A continuació es presenta una breu descripció dels capítols que constitueixen aquesta memòria per tal de tenir-ne una visió més global:

En primer lloc, a la *Definició dels objectius* s'exposen totes les idees de millora proposades i la tria dels objectius que es preveuen dur a terme al llarg d'aquest projecte.

Tot seguit entrem a l'*Estat inicial i plantejament de la qüestió* on es començarà per estudiar les funcionalitats del sistema que prenem com a punt de partida i els recursos que s'hi van emprar. Després, dins el mateix apartat, s'analitzen els recursos que tenim per a desenvolupar aquest nou prototip.

A continuació s'explica la *Planificació* del projecte, que inclou un diagrama temporal de les tasques realitzades, i després una explicació de cadascuna d'aquestes tasques. També s'inclou l'explicació els problemes trobats i la seva resolució.

El següent apartat és la *Memòria Tècnica*. Aquí s'entra en profunditat en el desenvolupament de cadascuna de les millores, de quins passos i tècniques s'han seguit per a realitzar-les. És l'explicació del treball dut a terme. Al final d'aquest capítol hi ha un apartat on s'exposen un seguit de proves per tal de demostrar que cadascuna de les ampliacions implementades funciona correctament.

Per a finalitzar s'expliquen les *Conclusions* obtingudes un cop finalitzat el projecte.

2. Definició dels objectius

En la finalització d'aquest projecte es pretén haver desenvolupat un prototip de vehicle teledirigit que millori les funcionalitats d'un altre prototip dissenyat en un projecte anterior. Un cop estudiat aquest punt de partida i tenint en compte el temps i recursos dels quals es disposa, es trien les millores que es duran a terme en el projecte. La selecció es fa entre les propostes del projectista anterior i les nostres. També hi ha alguna millora que apareix durant el desenvolupament del projecte.

A continuació s'exposa el llistat de totes les propostes de millora pel projecte, i subratllades les què es decideix dur a terme:

Millores proposades en el projecte inicial

- Programar un recorregut per tal que el vehicle l'efectuï automàticament

  La idea inicial consisteix en definir un recorregut i transferir-lo al vehicle a través d'alguna interfície per tal que el reprodueixi automàticament. S'opta per dur a terme aquesta millora, però fent que en comptes de programar el recorregut externament, es repeteixi un circuit ja fet pel vehicle.

- Control del prototip remotament, des d'un PC

  Es basa en poder controlar el vehicle des d' un ordinador, fent servir el comandament de pont entre el PC i el vehicle. És una proposta molt similar a l'anterior.

- Detecció de xocs i reacció automàtica a ells

  Consisteix en incorporar un acceleròmetre al vehicle que permeti detectar canvis bruscs de velocitat i reaccionar-hi fent alguna acció sobre els motors i encenent alguns LEDs per informar a l'usuari d'un possible xoc.

- Disseny d'una PCB que integri tota la part electrònica del vehicle

  Es basa en integrar en una sola placa aquells elements electrònics que es fan servir en el vehicle. Quedaria un disseny molt més polit, però més tancat i menys susceptible de modificacions. També s'han de poder separar els dispositius de les plaques que ja tenim, i això suposa una despesa econòmica.

Millores proposades per nosaltres

- Control del vehicle segons la posició del comandament

  Es vol controlar el vehicle amb moviments del comandament, que es poden detectar amb un acceleròmetre. D'aquesta forma el sistema serà més dinàmic i interactiu amb l'usuari.

- Revisió de la marxa endarrere

  En el projecte inicial es planteja un mètode per a què el vehicle es desplaci en marxa enrere, però que finalment no s'acaba implementant. La idea és revisar-ho i aconseguir que el vehicle pugui circular en marxa enrere.

- Telemetria

  Es tracta de poder enregistrar dades sobre el recorregut que del vehicle al llarg d'una sessió, i passar-les a un ordinador per fer un tractament de la informació.

- Velocitat constant

  La idea és implementar l'opció de que el vehicle es desplaci a una velocitat constant de forma automàtica i a voluntat de l'usuari.

- Gestió de la bateria

  Consisteix en proporcionar dades a l'usuari sobre l'estat de la bateria del vehicle, com per exemple el temps de vida que li queda i el consum que s'està fent.

- Millora del codi font inicial

  Es tracta de fer una revisió de la programació de les plaques transmissora i receptora ja que el projecte no es comença des de zero. La idea és simplificar-lo, eliminar redundàncies i examinar el diagrama de flux per tenir un punt de partida bastant ferm.

- Rutines d'actualització dels motors i dels LEDs

  La proposta sorgeix realitzant la millora del codi font inicial, perquè es troba la necessitat de canviar el mètode de com s'actualitzen els motors i els LEDs.

3. Estat inicial i plantejament de la qüestió

3.1 Projecte base

En aquesta secció analitzarem el prototip que ens deixa el projecte que prenem com a punt de partida, per tal de tenir una visió general de la seva estructura, del seu funcionament i dels recursos que es van emprar en el seu disseny. Després, en la memòria tècnica, es profunditzarà més en aquells aspectes que han estat necessaris per a dur a terme el projecte que descriu aquest document.

Descripció Funcional

L'estructura del prototip es basa en dos dispositius, un vehicle i un comandament que permet el seu control a distància per radiofreqüència. Les funcionalitats que garanteix aquest prototip són les següents:

- Desplaçament del vehicle cap endavant i rotació a dreta i esquerra.

Donat que el prototip té forma de tricicle, amb una roda lliure a la part davantera i dues a la part posterior connectades cadascuna a un motor, és possible fer que el vehicle giri provocant una diferència de velocitat entre els dos motors. Per tal que es desplaci en línia recta només cal mantenir la mateixa velocitat a ambdós motors. Cal destacar que la programació d'aquest prototip només permet girar quan el vehicle està aturat, és a dir, rotar. A la següent imatge es pot veure una representació del moviment del vehicle en funció dels motors engegats.

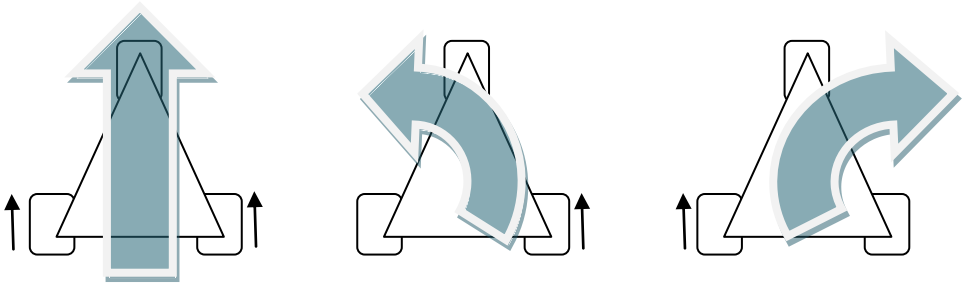


Figura 1: Direcció del vehicle – Prototip Base

El control de la direcció del vehicle es fa mitjançant el comandament, que té 4 polsadors a través del quals l'usuari pot interactuar amb el sistema, i cadascun d'ells correspon a una direcció. Val a dir que el prototip no es pot desplaçar en marxa enrere, en aquest sentit aquest polsador només serveix per a reduir la velocitat, funcionalitat explicada en el següent punt. El quadre de polsadors té un aspecte com el que es mostra a la Figura 2.

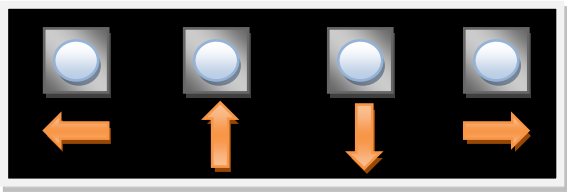


Figura 2: Esquema dels polsadors – Prototip Base

- Desplaçament amb 4 velocitats diferents.

  La potència subministrada als motors s'escala en 4 intervals a través de software, permetent així els canvis entre les 4 diferents velocitats a les quals pot anar el vehicle. S'augmenta o es redueix la velocitat mitjançant els polsadors endavant o endarrere respectivament.

- Velocitat instantània del vehicle monitoritzada.

  Tant el comandament com el vehicle disposen de 4 LEDs que indiquen en tot moment la velocitat a la que s'està desplaçant el prototip. El nombre de LEDs encesos corresponen directament a la velocitat a la que es va, a la següent imatge es mostren les possibles combinacions d'encesa dels LEDs.

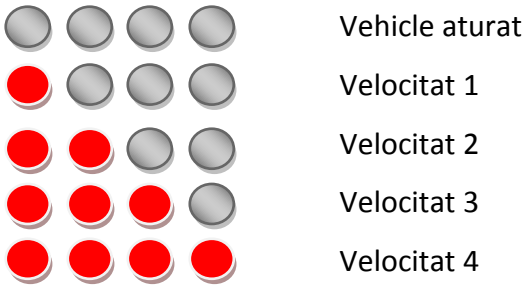
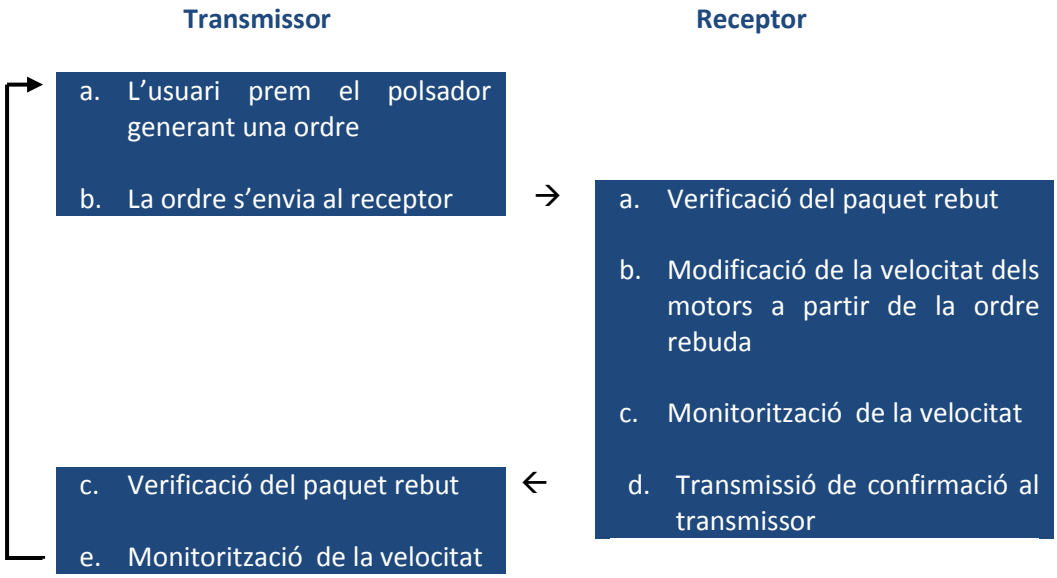


Figura 3: Representació de la velocitat – Prototip Base

A més, els LEDs fan una pampalluga quan es prem un polsador que no altera la velocitat dels motors. Per exemple prémer endavant quan es va a la màxima velocitat.

Programació

Els dispositius transmissor (comandament) i receptor (vehicle) estan programats de forma que el comportament habitual del sistema segueixi la següent lògica:



Recursos utilitzats

Meccano

El cos del vehicle s'ha construït amb Meccano, un joc de construcció basat en peces de metall de diverses mides i formes que s'uneixen entre sí mitjançant cargols, i amb les quals es pot muntar gairebé qualsevol estructura. La figura que es mostra a continuació és una fotografia del vehicle extreta de la memòria del mateix treball. S'hi pot veure el disseny del prototip en forma de tricicle i es poden intuir els motors que hi ha a les rodes del darrere. També es veu part del cablejat que connecta els diferents dispositius electrònics.

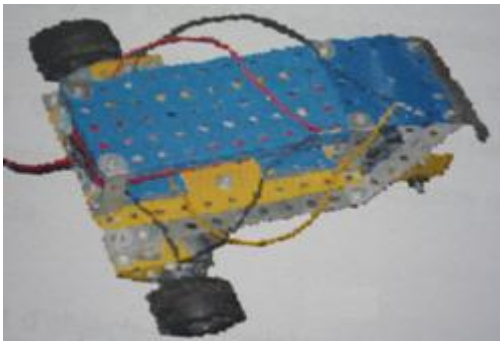


Figura 4: Vehicle – Prototip Base

Motors

En aquest projecte, els motors són els elements encarregats de generar la força necessària per a desplaçar el vehicle, i com hem vist abans en disposem d'un parell connectats a les rodes del darrere. Després d'estudiar la memòria d'aquest projecte, no s'ha trobat cap dada sobre el fabricant o el model d'aquests motors, però les especificacions sí. Aquest fet comportarà haver-ne de buscar uns que compleixin les mateixes característiques de cara a la construcció del nou prototip.

Les especificacions indicades ens diuen que són dos motors de corrent contínua, que cadascun d'ells treballa a una tensió de 7,2V i que sense càrrega arriben a una velocitat de gir màxima de 175rpm.

Per últim i encara que no té un motor associat que la controla, en aquest apartat cal especificar que la roda davantera ha de ser omnidireccional per així garantir que es puguin efectuar les rotacions. Si es posés una roda estàndard i s'aconseguís fer girar cap als costats sense que el fregament que això suposa ho impedís, l'eix que aguanta la roda hauria de suportar una força que a la llarga acabaria desmuntant-lo. Un segon inconvenient seria l'energia malgastada per culpa del fregament.

Plaques d'avaluació 13192-SARD

La definició de hardware ens diu que es tracta del conjunt de components que integren la part material d'una computadora, i en aquest cas són els dispositius que mitjançant la seva programació, duen a terme el control del sistema. Es fan servir dues plaques d'avaluació 13192-SARD de Freescale com la que es pot veure a la següent imatge, una per al control del comandament (transmissora) i l'altra per al control del vehicle (receptora).

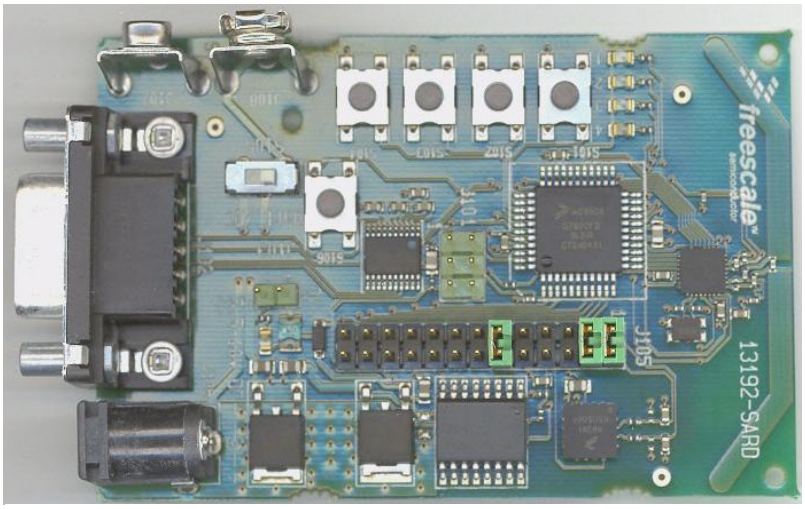


Figura 5: Placa d'avaluació 13192 SARD – Prototip Base

Les plaques d'avaluació normalment s'empren amb finalitats acadèmiques. Es caracteritzen per integrar un conjunt de dispositius que permeten implementar aplicacions de control sobre senyals d'entrada i de sortida. Sempre hi intervé un processat de les senyals dut a terme per un microprocessador. Més concretament, cadascuna de les plaques 13192-SARD conté els següents elements principals:

- Microcontrolador MC9S08GT60 (@20MHz, memòria: 60KB Flash i 4KB RAM).
- Transceptor RF 2.4 GHz amb l'antena impresa al circuit.
- Mòdem ZigBee.
- 2 acceleròmetres: MMA6261Q (eixos X i Y), MMA1260D (eix Z)
- 5 polsadors (4 de propòsit general i 1 de reset) i 4 LEDs.
- 13 pins d'entrada/sortida per propòsits generals.
- Port RS-232 per realitzar connexions sèrie amb un ordinador.
- Port USB de programació amb un mòdul BDM per facilitar la depuració del programa amb suport per a CodeWarrior™.
- Alimentació externa: Connexió Jack i connexió per a bateries de tipus prisma rectangular.

Corresponent-se amb el material cedit al projectista anterior, aquestes plaques es comercialitzen en un *Starter Kit* que conté dues SARD-13192, el software per a programar-les (Freescale CodeWarrior), un mòdul necessari per a transferir el codi de l'ordinador a la placa (USB MULTILINK, de P&E Systems) i un adaptador AC/DC de 9V.

Freescale CodeWarrior

La programació de les plaques es va dur a terme en l'entorn de treball CodeWarrior, de Freescale. Es tracta d'un entorn de desenvolupament de programació format per un editor de codi font, un compilador, eines per carregar el codi al microcontrolador, i un debugger (depurador). El depurador proporciona informació sobre les variables, caselles de memòria, i les línies del codi en execució. També cal dir que els llenguatges de programació que suporta són C/C++ i assemblador, encara que hi ha versions del software que en suporten de més. El fet de poder treballar en llenguatge assemblador fa que en determinats casos on és necessari, es pugui realitzar una programació de baix nivell, molt més laboriosa, però més precisa i òptima.

La figura a continuació mostra l'entorn de treball de CodeWarrior. Es pot veure la seva estructura basada en finestres (finestra de projecte i finestra de codi), que facilita molt la feina al treballar amb múltiples llibreries o fitxers alhora.

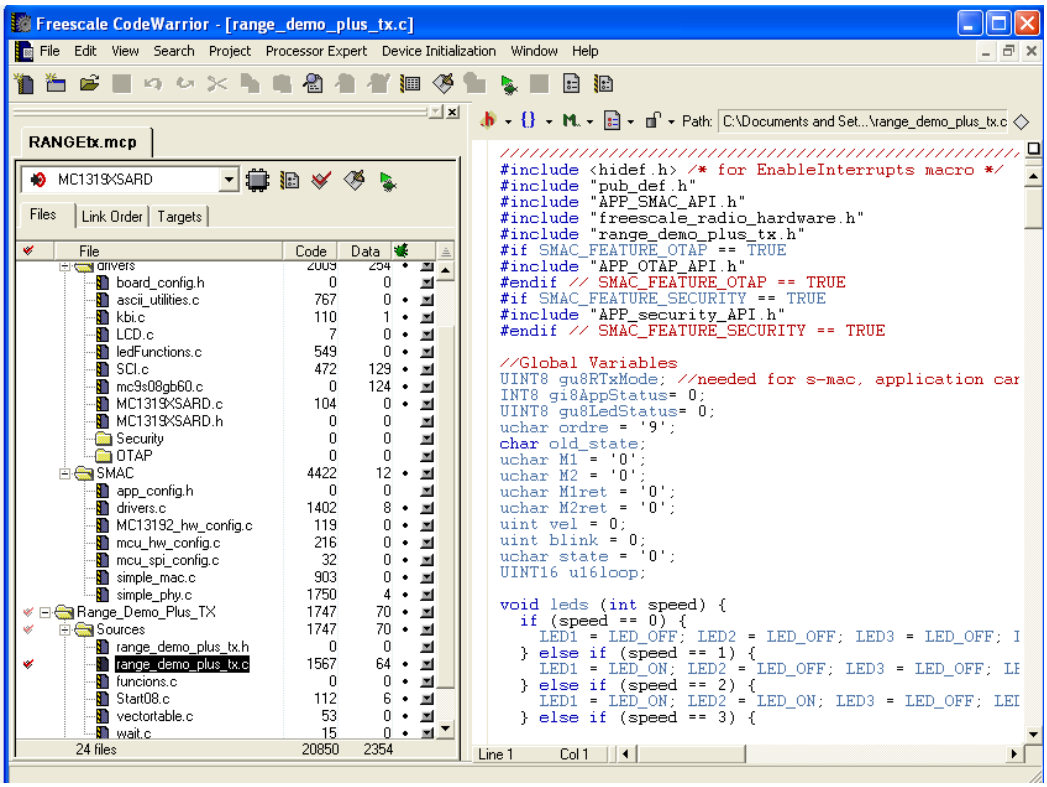


Figura 6: Captura de pantalla de l'entorn Freescale CodeWarrior – Prototip Base

La distribució d'aquest programari és de codi tancat i s'ha d'adquirir una llicència per a poder-lo fer servir. Hi ha algunes versions amb limitacions i amb llicències de prova per a finalitats acadèmiques. També cal tenir en compte que l'última versió de Windows on funciona correctament és XP, i tampoc és compatible per a altres plataformes com Linux o Mac. En aquests casos s'ha de fer ús d'una màquina virtual.

BeeKit

A banda del programa que permet editar i testejar el codi que es carrega a la placa, també es disposa del software BeeKit de Freescale, un programa que genera projectes basats en aplicacions determinades. Els projectes estan predefinits per a diversos models de plaques i microprocessadors, i també es poden modificar alguns paràmetres mitjançant una interfície gràfica. Un cop generat el projecte, per defecte es guarda en format XML, es pot exportar en format MCP (projecte CodeWarrior) per després bolcar-lo directament a la placa, o bé fer-ne una depuració o modificar el codi font.

Com que la gran majoria de les aplicacions generades amb aquesta eina fan ús del ZigBee i hi intervé la comunicació sense fils entre dues plaques, trobem que bona part dels projectes a generar van aparellats. Es crea un projecte diferent per a cadascuna de les plaques que intervenen dins l'aplicació i normalment sempre es diferencia entre placa transmissora i placa receptora. A continuació es pot veure una captura de pantalla del software BeeKit:

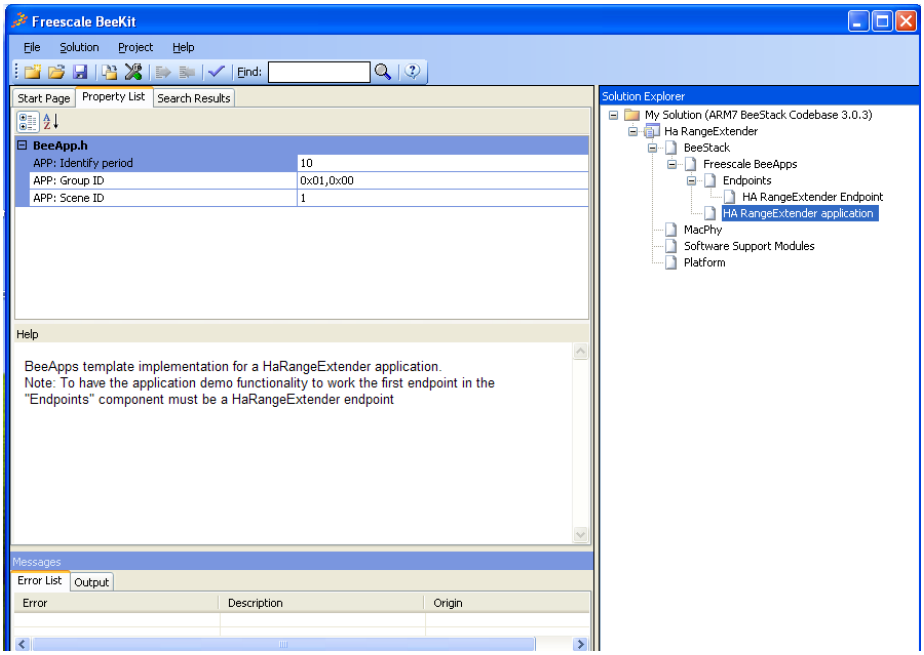


Figura 7: Captura de pantalla de l'aplicació Freescale BeeKit – Prototip Base

El projecte que estem estudiant va ser creat prenent com a base l'aplicació "Range Demostration Plus" generada amb el BeeKit. És una aplicació que permet mesurar l'abast màxim de la comunicació ZigBee entre dues plaques. El projecte de cada placa es va exportar a un projecte de CodeWarrior, i són aquests últims els que es van fer servir com a base per al projecte anterior.

3.2 Recursos per a desenvolupar el nou prototip

Hardware

Meccano

Per tal de construir el prototip disposem del mateix joc de construcció que el projectista anterior, Meccano. És molt utilitzat en la construcció de prototips de ràdio control i robòtica degut a la rapidesa de muntatge i la robustesa als xocs que presenta en comparació amb altres jocs de construcció com ara Lego o K'nex.

Un inconvenient de Meccano enfront d'altres opcions és que els dissenys construïts tenen un pes superior, i per tant els motors emprats han de tenir un parell motor més gran, que comporta un consum de corrent més elevat.

Motors

Disposem de 2 motors elèctrics de corrent contínua Como Drills 238-9822 que s'encarregaran de generar la força necessària per moure el vehicle. El fabricant ens indica les següents especificacions:

- Voltage d'alimentació: Mínim 12 V, Màxim 24 V
- Potència: 1,62 W
- Velocitat de gir (sense càrrega) : 4200 rpm
- Parell motor: 2,46 Ncm

Observant les dades es pot veure que els motors tenen una potència bastant moderada, però creiem que el parell motor és suficient per a desplaçar el nostre prototip. A més, tenim 2 motors, per tant la força es duplica.

Cadascun dels motors ve muntat dins una caixa d'engranatges tal i com es mostra a continuació:

Figura 8: Motor DC Como Drills 238-9822

L'objectiu d'aquest sistema d'engranatges és poder reduir la velocitat de gir del motor a canvi d'obtenir una força de gir més elevada.

Arduino Duemilanove

Les plaques que es faran servir per a dur a terme aquest projecte són dues Arduino Duemilanove. El hardware Arduino parteix d'un projecte que es va iniciar al 2005 per un grup d'estudiants a Itàlia, que tenia com a objectiu crear un dispositiu destinat al disseny de sistemes de control. Pretenia estar orientat a un àmbit acadèmic i que els preus de les plaques fossin molt inferiors a les plaques d'avaluació comercialitzades per altres companyies. Finalment es va aconseguir gràcies a definir-se com a open source hardware, o hardware lliure.

El hardware lliure es refereix a aquell hardware que permet que tota la seva informació sobre el disseny i les especificacions sigui d'accés públic, és a dir, que es trobi a la disposició de tothom. Això permet que s'hi pugui treballar fora de les restriccions que imposen les nacions o els estàndards. Així s'evita que els dissenys siguin tancats, fent que es puguin expandir, adaptar, reutilitzar o millorar d'una forma col·laborativa.

Els components Arduino es van començar a comercialitzar al 2008 i eren fets per l'empresa Smart Projects, però actualment s'han unit a aquest projecte moltes més companyies, com per exemple Adafruit Industries, SparkFun Electronics o Libelium. Gràcies a això, l'evolució d'aquest sistema no s'atura i permet que contínuament vagin sortint nous dissenys de plaques i de components. Alguns d'aquests models poden ser:

- **Arduino Mini**

Amb només 3 centímetres de llarg és la placa Arduino més petita que existeix, i ve a ser una miniaturització de la placa Duemilanove. Dissenyada per a projectes on sigui fonamental l'espai. Conseqüentment, és menys robusta, i el nombre de components que integra és menor.

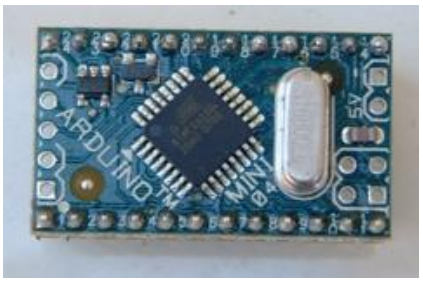


Figura 9: Arduino Mini

- **Arduino Mega**

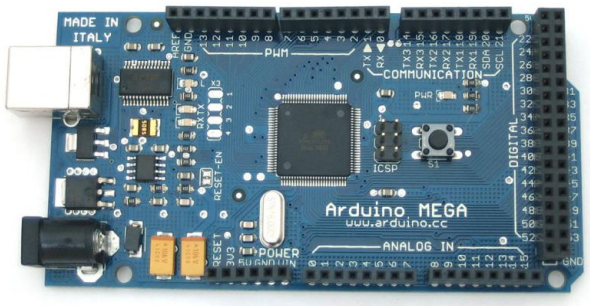


Figura 10: Arduino Mega

Aquesta placa està pensada per a projectes de més envergadura, ja que disposa d'una gran quantitat de pins d'entrada/sortida (54), una memòria amplia (128KB de Flash) i un processador ATmega1280. En quant a espai té unes dimensions considerables.

- **Arduino Duemilanove:**

És la vuitena versió i la més recent de la placa estàndard d'Arduino. El seu nom ha anat variant al llarg de les diverses versions: Arduino USB, Arduino Extreme, Arduino NG, Arduino Diecimilla, i per últim Arduino Duemilanove, que vol dir 2009, i és el seu any de sortida. A la imatge que tenim a continuació es pot visualitzar la placa:

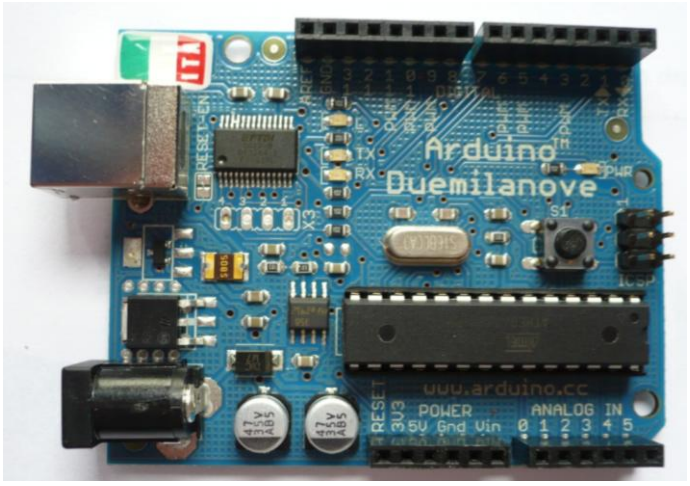


Figura 11: Arduino Duemilanove

Es tracta d'una model pensat per a propòsits generals. Amb les seves prestacions es pot dur a terme un ventall molt ampli d'aplicacions i per això és la més utilitzada en l'àmbit acadèmic. Si es necessiten prestacions més potents o més orientades a un àmbit en concret existeixen els altres models Arduino que poden encaixar segons les necessitats. Anem a veure les característiques principals de la Arduino Duemilanove:

- Microcontrolador ATmega168 (@16MHz, memòria: 16KB Flash, 1KB SRAM i 512B EEPROM).
- Polsador de reset i 4 LEDs (1 d'encès, 1 indica la recepció de dades (TX), 1 indica la transmissió de dades (RX) i 1 lliure).
- 14 pins d'entrada/sortida digital (6 d'ells són PWM).
- 6 pins d'entrada/sortida analògica.
- Port USB per realitzar connexions sèrie amb l'ordinador, i per bolcar el programa al microcontrolador.
- Connexió Jack per alimentació externa.

Si es comparen aquestes característiques amb les de la placa 13192-SARD, veiem que la Duemilanove és una mica inferior, sobretot en quant al microcontrolador. No obstant, el cost de la placa i l'entorn de treball Arduino fan que valgui la pena pagar aquest petit decrement de les prestacions. Podem escollir la opció de triar el microcontrolador ATmega328, que duplica la memòria flash , RAM i EEPROM, però no ens interessa perquè no necessitarem molta capacitat d'emmagatzematge de dades.

A continuació farem un incís en el fet que els pins puguin generar una senyal PWM de sortida, ja que és un recurs que farem servir en el projecte.

Senyal PWM (Pulse-width modulation)

La senyal consisteix en tenir una senyal quadrada a una determinada freqüència, i fer que el seu estat en alt tingui una durada major o menor en funció d'una variable, que en aquest cas es tracta d'un nombre enter entre 0 i 255 (1byte). A continuació es mostra una imatge amb la representació de 4 casos de PWM.

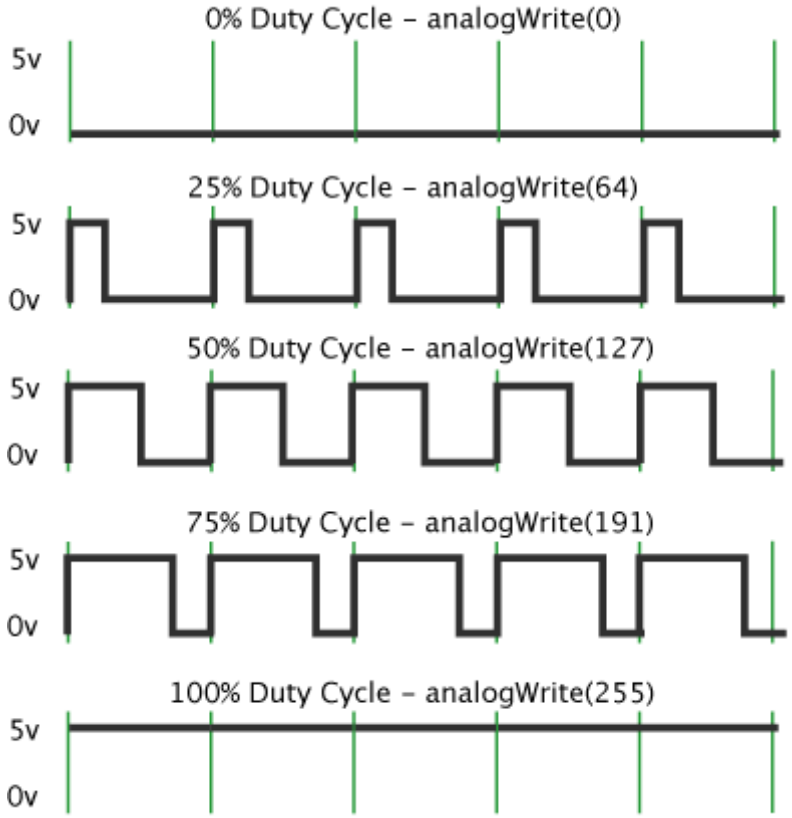


Figura 12: Pulse Width Modulation

Com es pot veure, com més llarg és el temps que la sortida està en alta, més voltatge en corrent continu s'està entregant. La relació entre aquest voltatge entregat i el valor passat a la senyal PWM és totalment lineal, si tenim una sortida de 5V i seleccionem el PWM a 127 (50%) tindrem un voltatge DC de 2,5V. Aquest és el motiu pel qual és molt utilitzada en el control de motors, ja que fàcilment es pot variar el voltatge DC entregat i així la velocitat del motor.

La funció en Arduino per a fer funcionar els pins en mode PWM és *analogWrite(x)* on *x* és el nombre enter d'1 byte.

Xbee Shield

Seguint amb el hardware, Arduino no només es basa en el disseny de les diferents plaques que s'ofereixen, sinó que també hi ha un seguit de complements, anomenats *shields* (escuts) que s'acoblen a les plaques i li ofereixen una funcionalitat afegida.

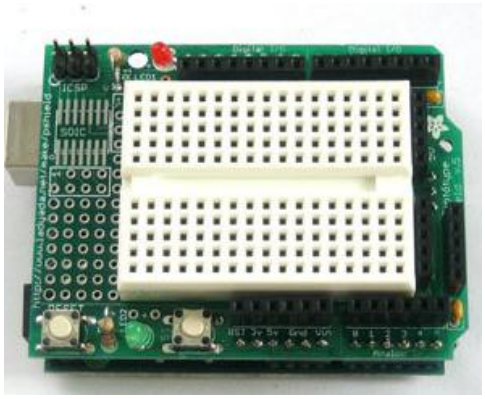
Una de les característiques més rellevant de l'Arduino Duemilanove és que pel fet de ser l'estàndard i el més estès, fa que sigui el model de placa per al qual hi ha més diversitat de shields disponibles. A mode d'exemple, entre tots els què podem trobar, tenim els següents:

- **Ethernet shield**

Ofereix a la placa un port ethernet a través del qual es pot connectar a una internet, proveint una IP i suportant els protocols de la capa de transmissió TCP i UDP. Pot ser molt útil en aplicacions de xarxa.

Figura 13: Ehternet shield

- **Protoboard Shield**



Amb aquest shield col·loquem una petita protoboard que queda a la part superior de la placa. Cal tenir en compte que els pins fets servir per encaixar el shield amb la placa són replicats, de forma que es poden muntar circuits directament sobre la placa sense la necessitat de realitzar llargues connexions. És molt pràctic per a fer prototips i proves amb components discrets.

Figura 14: Protoboard shield

- **Motor Shield**

Incorpora un driver per al control de motors elèctrics. No es pot adquirir ja que encara està en fase de proves, però hagués estat una molt bona opció per a fer servir en aquest projecte.

Figura 15: Motor shield

- **XBee Shield**

Aquest complement es farà servir en el projecte que estem desenvolupant. Consisteix en un adaptador que permet acoblar un mòdul XBee per tal que la placa es pugui comunicar sense fils fent servir l'estàndard ZigBee. El mòdul es comercialitza tant de forma separada, comprant per una banda l'escut i per una altra el mòdem XBee que posteriorment s'haurà de soldar a l'escut, com de forma conjunta, l'escut ja porta incorporat i soldat el mòdem. Generalment aquesta última opció és la més econòmica i pràctica, i és la que farem servir. També cal fer una diferenciació entre XBee, que és el nom del producte comercial, i ZigBee, que és l'estàndard de comunicació IEEE 802.15.4.

El mòdem XBee en el què l'escut es basa és el model de la casa Digi (anteriorment era Maxstream). Nosaltres farem servir el XBee ZNet 2.5, que té les següents característiques:

- Abast en interiors: 40m
- Abast en exteriors: 120m
- Velocitat de transmissió: 250Kbps
- Consum de corrent en transmissió: 40mA
- Consum de corrent en standby: 15mA
- Banda d'operació: ISM 2.4GHz
- Antena integrada a la placa.

A la següent figura es pot veure el shield XBee d'Arduino amb el mòdul Xbee integrat (imatge de l'esquerra), i el conjunt de la placa Duemilanove amb el shield muntat (imatge de la dreta):

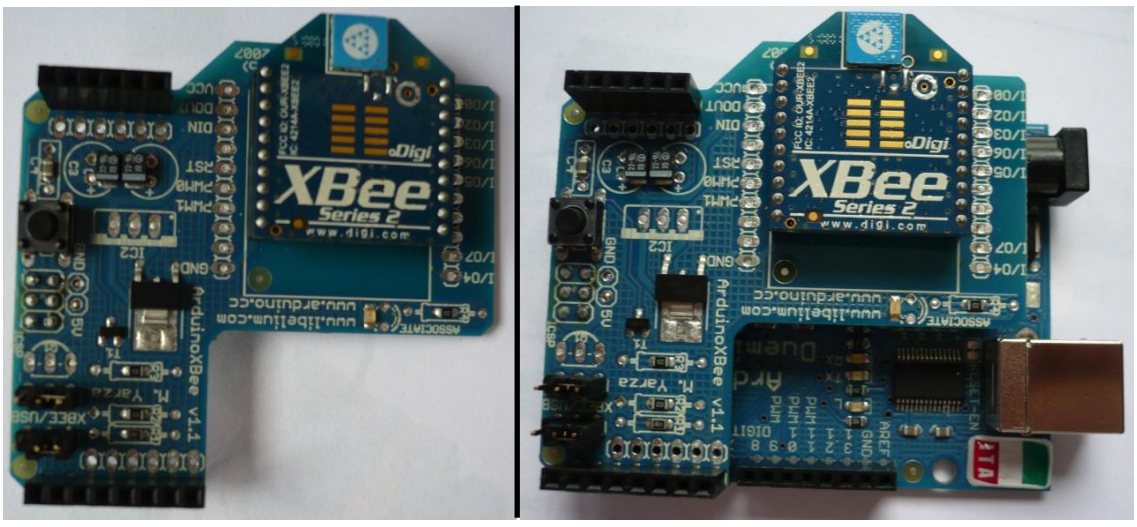


Figura 16: XBee shield acoblat a la Duemilanove

Una de les característiques que cal comentar, és que la unió entre el shield i la placa Duemilanove es fa mitjançant els 6 pins del port analògic i els 8 primers pins del port digital, però això no suposa la pèrdua d'aquestes entrades ja que es repliquen al shield. També queda connectat el port ICSP que és necessari per programar el mòdem XBee.

La única modificació que cal tenir en compte és que la placa no es pot comunicar amb el port USB i el mòdem XBee alhora, per això disposem de dos jumpers que ens permeten seleccionar manualment quin canal es farà servir. Un dels jumpers és per seleccionar la transmissió i l'altre la recepció, de forma que tenim quatre possibles configuracions:

Jumper TX	Jumper RX
XBEE	XBEE
XBEE	USB
USB	XBEE
USB	USB

Taula 1: Posicions dels jumpers del XBee shield

Jugant amb aquestes configuracions podem, per exemple, rebre dades pel mòdem XBee i imprimir-les a la pantalla de l'ordinador a través de la connexió USB, o a l'inrevés, controlar a través de l'ordinador les dades que s'enviaran per l'aire. Donada la seva importància en aquest projecte, descriurem breument l'estàndard ZigBee.

ZigBee

Zigbee és el nom de l'especificació d'un conjunt de protocols basats en l'estàndard IEEE 802.15.4 de xarxes inal·làmbriques d'ús personal (de l'anglès PAN, Personal Area Network). Aquest sistema és pensat per cobrir les necessitats de dispositius que tenen una gran dependència de la bateria, normalment en l'àmbit de la domòtica com ara sensors, joguines, aparells de control remot... Comparat amb aplicacions d'àudio o vídeo, aquests dispositius no transmeten una gran quantitat de dades. Per això els protocols ZigBee s'estableixen per minimitzar el temps d'activitat de transmissió dins de la xarxa. Així s'evita el consum d'energia innecessari, ja que els dispositius ZigBee tenen la capacitat d'adormir-se quan no estan en transmissió, de manera que el consum d'energia baixa notablement.

Hi ha diversos tipus de dispositius ZigBee segons la seva funció dins la xarxa PAN. Nosaltres només en farem servir 2:

- Dispositiu final o ZigBee End Device (ZED): Només es pot comunicar amb un router o un coordinador que faci la funció de node pare, i això permet que la seva actuació dins de la xarxa sigui només en el moment que necessiti rebre o transmetre alguna cosa. Serà el dispositiu allotjat dins el vehicle.

- Coordinador ZigBee (CZ): Només hi ha un per xarxa, i s'encarrega del seu control. En el moment d'encès de la xarxa Zigbee selecciona quin canal RF utilitzar (mirant en quin canal hi ha menys densitat d'energia i per tant menys interferències), interconnecta tots els nodes presents i assigna un identificador a cada dispositiu. Serà el dispositiu situat al comandament.

També hi ha diverses tipologies de xarxa, però donat que només tenim 2 dispositius no ens fa falta estudiar-les, la connexió entre els mòduls ZigBee serà directa.

Acceleròmetre ADXL 335

Donat que la placa Duemilanove no porta cap acceleròmetre incorporat, per a la millora del control del vehicle mitjançant el moviment de la placa transmissora es fa servir un acceleròmetre ADXL 335 com el que es veu a la imatge:

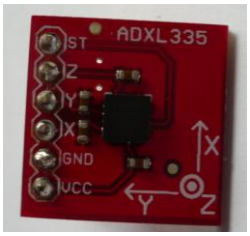


Figura 17: ADXL 335

Amb aquest dispositiu es pot mesurar l'acceleració dels 3 eixos de l'espai (x, y i z) en un rang de +/- 3G. Les mesures que dóna a la sortida estan relacionades amb l'angle que forma el pla de l'eix en qüestió respecte un altre pla normalitzat. Això permet conèixer també la posició (inclinació o rotació) del dispositiu en tot moment.

Té 6 pins, 2 dels quals són per a alimentar el circuit, 1 per a activar-lo i els 3 restants corresponen a les mesures de cada eix. El component s'ha de connectar a l'entrada analògica de la nostra placa, perquè els valors de sortida de l'ADXL 335 no són digitals. La següent imatge mostra com queda l'acceleròmetre connectat a la placa a través dels pins replicats pel XBee shield:

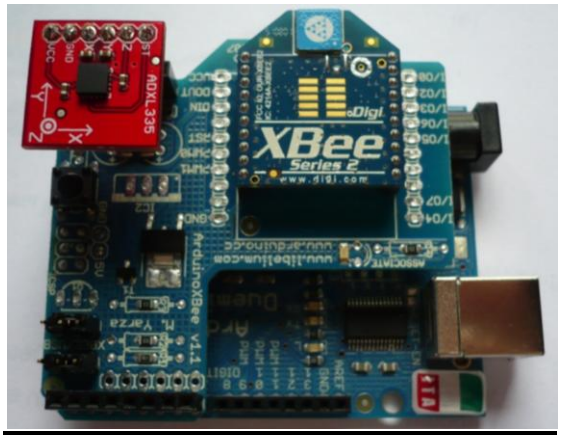


Figura 18: ADXL 335 acoblat a la Duemilanove

Components electrònics discrets

Per acabar amb els recursos, es fan servir diferents components electrònics discrets per al muntatge de circuits. Aquest components es poden adquirir fàcilment en qualsevol tenda d'electrònica i el seu preu és relativament barat. Es van comprant conforme sorgeix la necessitat. Són transistors bipolars 2N2222 i 2N2907, resistències, LEDs, una protoboard i cablejat.

Software

Arduino

A més a més de les plaques i els shields, Arduino també disposa del seu propi entorn de desenvolupament basat al igual que el hardware, en open source, programari lliure. És una aplicació Java i és compatible amb plataformes Linux, Mac OS i Windows, la última versió disponible és la Arduino 0018 i el seu disseny és simple, però totalment pràctic i funcional. Treballa amb llenguatge C/C++, i té definides moltes llibreries amb funcions que simplifiquen en gran mesura el codi redactat.

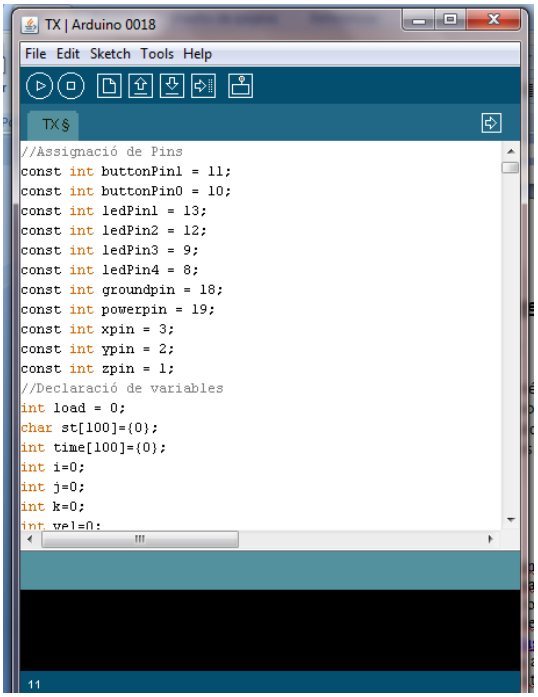


Figura 19: Entorn de desenvolupament Arduino 0018

Podem veure a la imatge anterior que l'aplicació està formada per una única finestra on es mostra el codi que s'està redactant amb la sintaxis colorida, de forma que resulta ràpid interpretar les línies de codi. A la part superior hi ha una botonera amb les accions de compilar, carregar el programa a la placa i obrir el monitor serial, entre d'altres més bàsiques. No disposa de depurador ni de simulador, de forma que per a fer una depuració del codi s'ha de carregar a la placa i anar imprimint a propòsit i a través de la connexió serial indicacions que donin pistes de l'execució del programa.

Les avantatges d'aquesta eina enfront l'entorn de CodeWarrior, són que la simplicitat que té fa que sigui molt més directa la implementació a l'hora de treballar-hi. Les llibreries que té carregades des d'un principi, ofereixen una gran quantitat de funcions que faciliten molt la feina. Un exemple molt clar és la forma de transmetre dades a través del serial, on només cal executar una sola comanda per enviar el paquet.

Els inconvenients venen determinats per les limitacions que té. El fet de no tenir un depurador integrat fa que sigui més laboriós fer un seguiment de l'execució del codi a la placa. També, el fet de no tenir un simulador ens obliga a carregar el programa a la placa cada cop que es vulgui fer un test.

Digi X-CTU

Per tal que es pugui establir una connexió entre els mòduls XBee que tenim a cadascuna de les Duemilanove, cal que els configurem correctament. Per això farem servir l'aplicació X-CTU de Digi, que permet carregar una configuració al mòdem, o bé configurar-lo via terminal. També cal dir que aquesta aplicació només és suportada per Windows. A continuació en veiem una captura de pantalla:

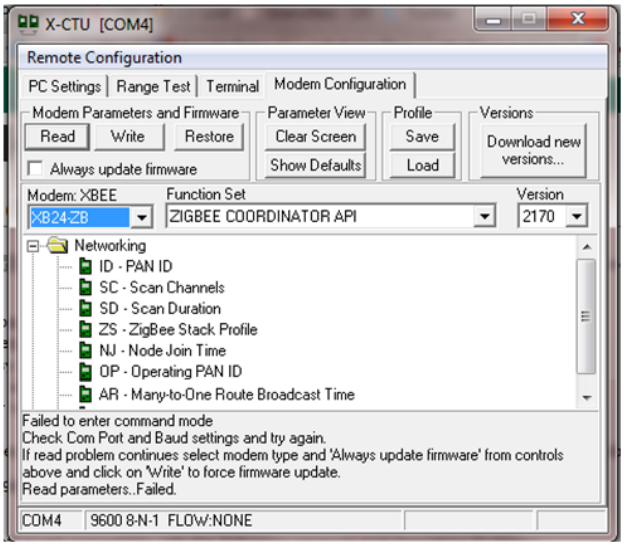


Figura 20: Digi X-CTU

L'aplicació permet configurar diversos paràmetres dels mòdems, nosaltres només definirem als nostres mòduls XBee les funcions. El mòdem de la placa transmissora serà el coordinador de la xarxa (ZNet 2.5 Coordinator AT) i el mòdem de la placa que anirà dins el vehicle serà un dispositiu final (ZNet 2.5 Router/End Device AT). També configurarem el mateix PANID (Identificador de xarxa PAN), per tal que els dos dispositius treballin sempre dins la mateixa xarxa, és un mètode de seguretat per tal que no interfereixin altres dispositius ZigBee aliens. El valor del PANID pot ser qualsevol, posarem 2222.

Cadence PSpice

Per al disseny del Pont H que farem servir per a controlar els motors del nostre vehicle, s'utilitzarà l'eina PSpice, que és un software de simulació de circuits elèctrics, principalment analògics. És un software molt complert amb la qual es poden dur a terme simulacions de diversos tipus (transitòries, estàtiques, escombrats...) i fins i tot l'extracció del circuit per tal de fer-ne el layout amb una altra eina. Nosaltres ens limitarem a fer un anàlisi en estàtic dels ponts H.

Farem servir una versió del programa per a estudiants, que és gratuïta (PSpice Student). Es poden trobar diverses captures de pantalla a la secció *5.2 Marxa enrere*.

4. Planificació

En aquest apartat s'explica el procés que s'ha seguit per tal de realitzar el projecte i complir els seus objectius. En primer lloc es presenta una descripció temporal de la totalitat del projecte, i després l'explicació de totes les etapes que el conformen. No farem cap incís en relació als recursos utilitzats perquè el projecte és fet únicament per un projectista, per tant la seva dedicació és del 100% en totes les tasques.

4.1 Diagrama de Gantt

Per tal de visualitzar el desenvolupament temporal del projecte fem servir el diagrama de Gantt, una eina gràfica que mostra la dedicació en el temps de les diverses etapes que composen un treball, i també les relacions de precedència entre elles. A continuació es pot visualitzar el diagrama de Gantt d'aquest projecte, dut a terme amb Microsoft Project.

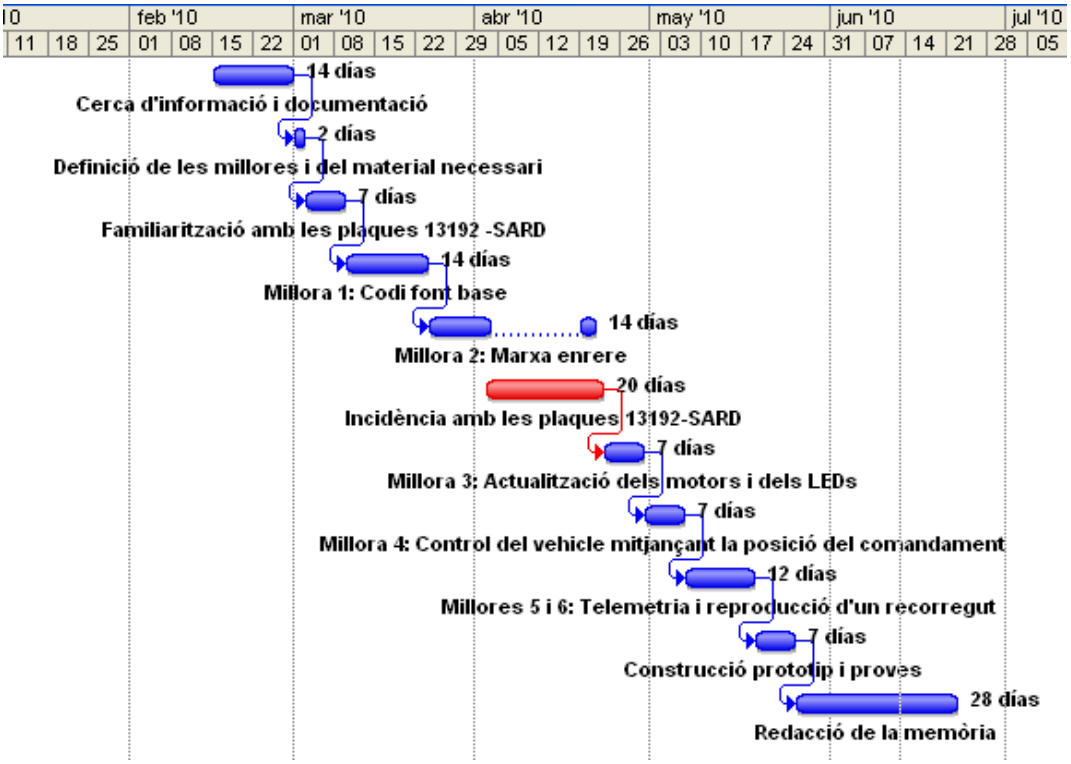


Figura 21: Diagrama de Gantt de la planificació

Com es pot veure al gràfic el projecte avarca una durada de 4 mesos, entre els què es reparteixen les diferents etapes. Al costat de cadascuna es mostra la seva durada en dies, i s'han marcat en color blau aquelles que s'havien planificat des d'un principi. En color vermell tenim una incidència no prevista que ens crea una escletxa en aquest desenvolupament ja previst, però es supera amb èxit i el projecte s'acaba dins el límit d'entrega. La relació entre les diferents etapes també ens indica que és un projecte molt lineal, on no es realitzen tasques en paral·lel, que s'explica pel fet d'estar realitzat només per una persona.

4.2 Etapes

A continuació es descriuen totes les tasques representades en el diagrama de Gantt de la figura 21.

- **Cerca d'informació i documentació**

A partir del primer moment en què comença el projecte, el tutor ens concedeix el document de la memòria del projecte *Disseny d'un vehicle amb control remot sense fils*, que es pren com a punt de partida. Es dedica un temps a la seva lectura i interpretació de tot el seu contingut, fins que s'hi està totalment familiaritzat, ja que és essencial per poder-lo fer servir de punt de partida.

Tot seguit es fa una recerca de la documentació relativa als recursos fets servir en aquest projecte i que segur serà necessària en el desenvolupament del nou projecte: Manuals de les plaques 13192-SARD i de l'eina BeeKit, datasheets dels components utilitzats, i material docent sobre l'entorn CodeWarrior, la programació en C/C++ i en llenguatge assemblador.

- **Definició de les millores i del material necessari**

Després de tenir ben assumit el projecte inicial, es decideixen les diverses millores que s'implementaran , basades en les millores proposades a la memòria que s'ha estudiat i també pensant-ne de noves. Després es fa l'abstracció del material que serà necessari i es sol·licita a la persona encarregada del departament. El material és entregat amb brevetat perquè es troba disponible al mateix departament, i consta d'un *Starter Kit* de Freescale de les plaques 13192-SARD (descrit a l'apartat 3.1 *Què ens trobem: Projecte base*) i del material per a muntar el vehicle.

- **Familiarització amb les plaques 13192 –SARD**

Un cop es tenen les plaques, comencem a treballar am elles i amb l'entorn de treball CodeWarrior per tal d'adquirir un cert rodatge. S'implementen codis de prova, es realitzen simulacions i depuracions d'exemple per refrescar els coneixements, ja que ja s'ha fet servir aquest entorn de treball anteriorment en una assignatura.

- **Millora 1: Codi font base**

Quan es té la suficient agilitat amb les plaques i la seva programació, es comença a treballar en la primera millora. Consisteix en optimitzar el codi font de cadascuna de les plaques que es pren de partida en aquest projecte, per així començar amb una base sòlida i ben estudiada. Progressivament a la millora, es van fent proves de verificació de que el nou codi font manté totes les funcionalitats inicials.

- **Millora 2: Marxa enrere**

En aquest punt es busca solució al problema presentat a la memòria del treball anterior, que fa que la marxa enrere del vehicle no funcioni. En primer lloc es fa una cerca sobre les possibles solucions, i es decideix implementar un pont H que serà el driver per a controlar els motors.

El següent pas és dissenyar els components del pont H tenint en compte les nostres especificacions. Es fan els càlculs teòrics i les corresponents simulacions per verificar-los, i després l'adquisició dels components per soldar-los en forma de prototip sobre una placa perforada.

En el transcurs d'aquesta etapa ens sorgeix un problema amb les plaques que la interromp, just després d'acabar de soldar el primer dels ponts H. Un cop resolt el problema es munta el segon pont H amb normalitat.

- **Incidència amb les plaques 13192-SARD**

Arribats el punt de ja tenir muntats un dels drivers, es va a testejar sobre una plataforma construïda amb Meccano on només hi ha els motors. Una de les plaques es connecta a l'ordinador a través del port USB, i la que controla els motors s'alimenta amb l'adaptador AC/DC de 9V. A l'encendre aquesta segona placa s'observa que deixa de funcionar a l'instant. Pensant que es tracta d'un problema amb el disseny del pont H que s'està provant, se li comenta al tutor, ja que després de refer les simulacions i els càlculs es comprova que hauria de funcionar correctament.

Mentre el tutor analitza el cas es segueix treballant amb l'altra placa, però sense fer proves amb el pont H a l'espera de la seva resposta. Pensant en una de les altres millores es vol testejar la connexió sèrie de la placa amb l'ordinador, per això s'implementa un codi de prova que es volca a la placa. Es vol fer una depuració del codi i com que el port sèrie no alimenta a la placa, la connectem amb l'adaptador de 9V. Lamentablement succeeix el mateix que amb la placa anterior.

Es contacta de nou amb el tutor explicant-li la situació, i s'arriba a la conclusió de què l'adaptador de corrent proporcionat és defectuós. En conseqüència, es retornen els dispositius i s'ha de buscar unes noves plaques que s'ajustin al projecte que estem desenvolupant, i que siguin similars a les 13192-SARD, que ja estan obsoletes al mercat. Trobem dues alternatives:

- **Freescale 1321xNSK (Network Starter Kit)**: És el pack superior al què teníem (Developers Starter Kit). Està pensat per al desenvolupament de xarxes ZigBee i és un pack molt complert. No obstant, el seu preu també és elevat i moltes de les prestacions que ofereix no les necessitem per a aquest projecte. Com que el material es quedarà al departament és una idea atractiva de cara a futurs projectes.

- **Arduino Duemilanove**: És una placa que sorgeix d'un projecte molt recent que es basa en el hardware lliure. D'aquesta manera els seus preus són bastant econòmics i és una placa senzilla, però s'hi poden acoblar shields per ampliar-la. L'inconvenient és que l'entorn de treball fet servir fins ara canviarà bastant.

Ens decantem per Arduino, pel cost, per la motivació de treballar amb aquestes plaques que estan agafant molta força actualment i també fan que el projecte ens resulti més interessant. Sol·licitem el material al departament, consistent en dues Arduino Duemilanove, dos XBee shields i dos acceleròmetres ADXL 335, ja que a diferència de les 13192-SARD, no els tenen incorporats.

Mentre arriba el material, que triga un parell de setmanes, s'acaba de soldar el segon pont H que havia quedat pendent i es comença a fer la recerca d'informació sobre el funcionament de les Duemilanove i l'entorn de programació Arduino. De forma que quan arribin ja tindrem una bona idea del seu funcionament.

Arriben les plaques i es fan proves amb codis d'exemple, també es construeixen alguns dispositius com els polsadors i els LEDs, que sí els tenien les SARD però no les noves plaques. Es prova que tot funciona correctament.

El següent pas és adaptar tot el codi redactat fins ara amb CodeWarrior a l'entorn Arduino. Es tracta del mateix llenguatge, però totes les funcions de les llibreries són diferents. Hi ha noves funcions que ens simplifiquen el codi i per això s'ha de canviar la seva estructura. Un cop fet i comprovat, es pot continuar amb el desenvolupament previst del projecte.

Millora 3: Actualització dels motors i dels LEDs

Aquesta etapa consisteix en el desenvolupament del codi que controla la velocitat dels motors i la seva representació en els LEDs, tenint en compte les modificacions del codi fetes en la primera millora.

Millora 4: Control del vehicle mitjançant la posició del comandament

Es treballa per aconseguir l'objectiu de controlar el vehicle amb un acceleròmetre, realitzant moviments amb la placa transmissora que fa de comandament. Disposem d'un acceleròmetre ADXL 335 que s'ha de soldar als pins que connectaran amb la placa, abans de fer qualsevol prova. Com que es tracta d'un component fràgil, es dediquen unes hores soldant resistències per agafar pràctica i després soldar l'acceleròmetre sense cap problema. Un cop fet ja es pot començar a treballar amb ell.

Al cap d'una setmana la millora queda realitzada, però posteriorment es van realitzant ajustos dels valors de canvi a la màquina d'estats perquè el control del vehicle sigui més natural. Es fa de forma experimental.

Millores 5 i 6: Telemetria i reproducció d'un recorregut

Les ampliacions de reproduir un recorregut i de telemetria es fan quasi simultàniament, ja que el mètode que es fa servir és molt similar. Consisteix en guardar a les dues plaques els canvis d'estats en dos vectors, un de temps i un d'estats. Els vectors serviran a la placa transmissora per a carregar la informació a l'ordinador (telemetria) i la placa receptora per reproduir el recorregut guardat.

Construcció prototip i proves

Es construeix el vehicle en la seva totalitat. Es busca un model pràctic, no estètic, per això la construcció és ràpida. En una primera instància es fa una visita a una tenda d'automodelisme per mirar d'aconseguir una base per al nostre vehicle, de forma que tingui una imatge més cuidada. Els preus que ens proposen en comparació a la finalitat del prototip no ens semblen viables i es descarta.

Amb el vehicle muntat es duen a terme totes les proves de funcionament i s'ajusten els últims retocs, com per exemple els paràmetres dels acceleròmetres que determinen l'estat dels motors en funció de la posició del comandament.

Redacció de la memòria

És el temps invertit en la elaboració d'aquest document. Es van anotant dia a dia totes les coses que es van fent per tal de no tenir problemes de recopilació d'informació o de pèrdua de detalls amb el temps. Un cop acabats els objectius proposats es fa la redacció definitiva.

5. Memòria tècnica

En aquest apartat es descriu detalladament i a nivell tècnic el treball que s'ha dut a terme en aquest projecte desenvolupant cadascuna de les millores.

5.1 Millora del codi font base.

Per tal d'estudiar el codi font de cadascuna de les plaques del projecte inicial, que ve annexat a la memòria del projecte base, el primer que fem és reproduir-lo íntegrament per així poder simular-lo i comprovar que no dóna cap error de compilació. No podrem comprovar el seu funcionament perquè no disposem del vehicle que es va construir, però és necessari replicar-lo ja que el farem servir com a base per al nostre codi font. Anirem aplicant les diferents millores o ampliacions directament sobre aquest codi i de forma progressiva per anar verificant el correcte funcionament dels diferents apartats.

Un cop comencem a replicar el codi, d'entrada veiem que va ser generat amb el software Freescale BeeKit i l'aplicació de la qual es va partir és la Range Demostration Plus. Consisteix en dos codis que seguidament expliquem, un per al receptor i un per al transmissor:

- El transmissor emet contínuament i amb una determinada freqüència un paquet cap al receptor i espera a la recepció d'un ACK (acknowledgement).

- El receptor sempre està a l'espera de rebre un paquet del receptor. Un cop rebut, analitza el seu LQI (Link Quality Indicator) i en funció de la qualitat de recepció (potència de la senyal rebuda), encendrà 1 (mínim), 2, 3 o 4 (màxim) LEDs durant un instant. Després retorna un ACK cap al transmissor i espera a rebre el següent paquet.

La funció d'aquesta aplicació és mesurar l'abast de la senyal ZigBee. Agafant la placa receptora i allunyant-nos de la transmissora es va observant en el nombre de LEDs encesos un deteriorament de la senyal en funció de la distància. Fins que arriba a un punt en què ja no s'encén cap LED, no es reben els paquets.

Malgrat la versió del BeeKit que disposem és més nova que la que es va fer servir en el projecte anterior, encara hi ha l'aplicació Range Demostration Plus disponible per a generar. La exportem a un projecte de CodeWarrior, bolquem cada imatge binària a una placa i verifiquem que funciona correctament. Després, modifiquem aquest codi per tal que sigui exactament igual que el que prenem com a base. En aquest sentit un dels inconvenients que suposa tenir una nova versió del BeeKit és que les variables del codi font que s'ha generat tenen uns literals diferents que la versió anterior i s'han de canviar arreu on apareixen en el codi base.

Un cop reescrit el codi i verificat que no hi ha errors, analitzem detalladament el seu diagrama de flux per tal de conèixer exactament quin ha de ser el seu funcionament. A continuació es mostra el diagrama de flux de tot el sistema, format pel diagrama de flux de la placa transmissora i pel de la receptora. Els diagrames només es creuen quan hi ha una transmissió d'informació pel canal ZigBee:

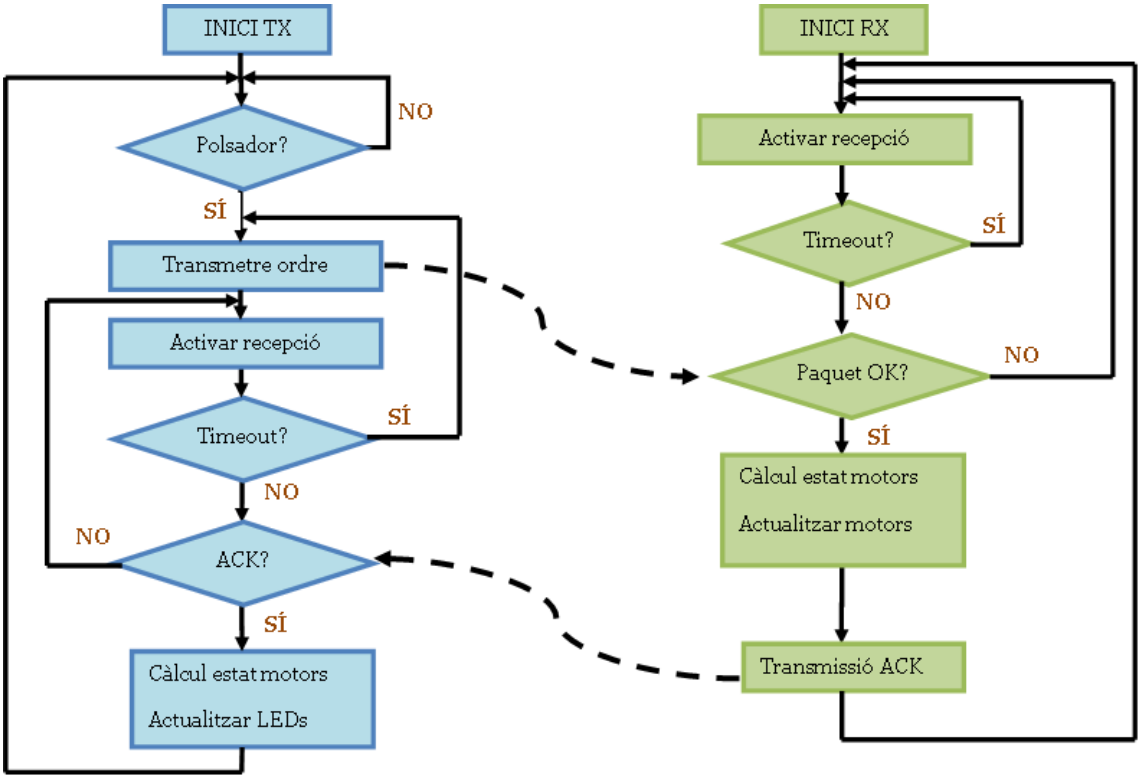


Figura 22: Diagrama de flux del prototip base

Observant el diagrama veiem que el funcionament, de forma descrita, és el següent:

Transmissor: La placa està en espera fins que l'usuari prem un dels polsadors, un cop es dóna la situació, es transmet el número de polsador a la placa receptora i s'espera a rebre un ACK. Com que la recepció està activa durant un temps determinat, si s'arriba a aquest límit de temps (Timeout) i no s'ha rebut cap paquet es transmet de nou la ordre i s'activa la recepció. Un cop rebut s'analitza i si no és un ACK, es descarta i es segueix a l'espera d'un altre paquet. Si es rep l'ACK, es salta a una rutina on es calcula l'estat dels motors (velocitat, sentit de direcció i sentit de gir) en funció de l'ordre rebuda i de l'estat anterior. Per últim, amb la velocitat calculada s'actualitzen els LEDs corresponents i es torna a l'espera de que es premi un polsador.

Receptor: La placa s'inicialitza i s'activa la recepció a l'espera de rebre un paquet. Si s'arriba a Timeout es torna a activar la recepció. En el moment de rebre un paquet es verifica que sigui correcte i en cas de no ser-ho es descarta i es segueix esperant. Si el paquet conté una ordre d'un polsador, és a dir correcte, es calcula l'estat dels motors i s'activen a la velocitat que correspongui per a cadascun. Per últim s'encenen els LEDs que mostren la velocitat del vehicle i es torna a activar la recepció per rebre un nou paquet.

Després de fer la redacció del codi font de les plaques, i l'estudi del diagrama de flux, detectem que es podrien millorar els següents aspectes:

- El càlcul de l'estat dels motors. Vist en el codi, aquesta funció es tracta d'una rutina on hi arriben 3 variables que parametritzen l'estat dels motors, i dues de control. Quatre de les variables són retornades amb les modificacions pertinents, és a dir, són un total de 5 variables on totes depenen entre elles. Això esdevé en un fragment de codi difícil de comprendre i poc flexible a l'hora de fer-ne alguna modificació.
- L'estructura del diagrama de flux es pot millorar i simplificar en diversos sentits. Per exemple s'avaluarà si és necessari un sistema de verificació dels paquets rebuts a nivell d'aplicació, o si es pot evitar fer el càlcul de l'estat dels motors a les dues plaques.

Càlcul de l'estat dels motors

Amb aquesta millora es pretén simplificar al màxim la rutina que realitza la funció de determinar l'estat dels motors del vehicle en funció de la ordre rebuda. En primer lloc donarem un cop d'ull a les variables que prenen joc dins aquesta rutina:

Variable	Descripció	Valors
ordre	Número de polsador premut	0: esquerra 1: endavant 2: endarrere 3: dreta
ma	Sentit de gir	0: recte 1: esquerra 2: dreta
sentit	Sentit de direcció	1: endavant 2: endarrere
vel	Velocitat	0: aturat 1: mínima 2 3 4: màxima
blink	Indica si els LEDs han de fer una pampalluga	0: no 1: sí

Taula 2: Variables que intervenen en el càlcul de l'estat dels motors

Ara cal saber el funcionament de la rutina:

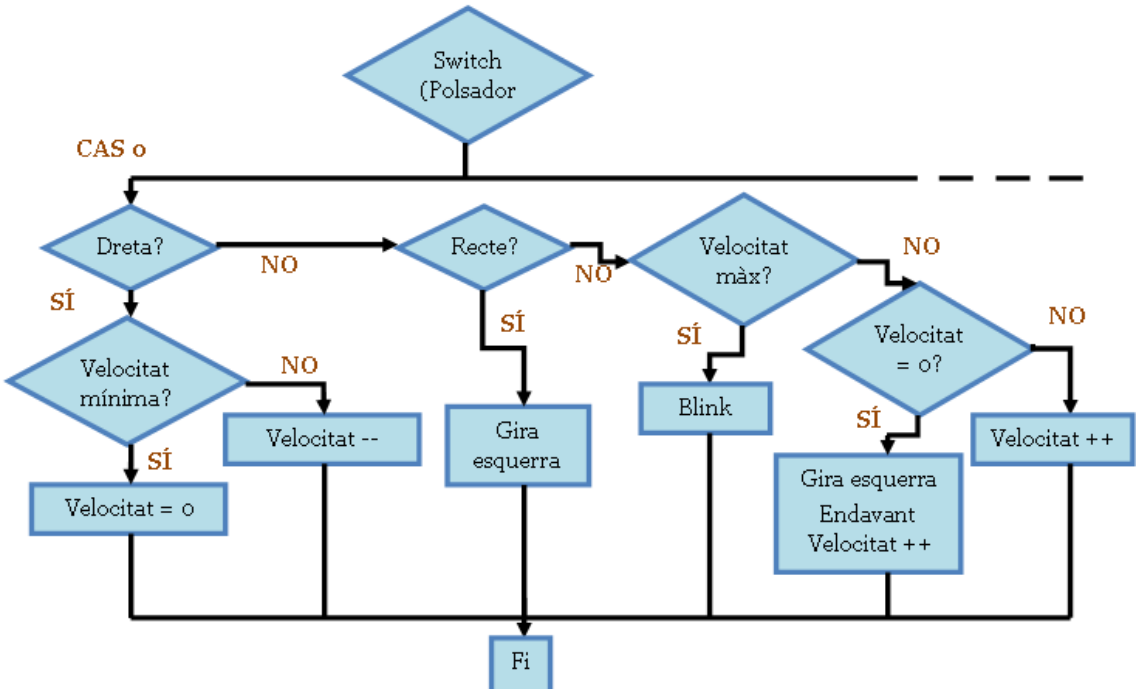


Figura 23: Diagrama del càlcul de l'estat dels motors del prototip base

Com es pot veure al diagrama de flux d'aquesta rutina, es té com a punt de partida un switch en funció de la ordre rebuda. Per tal de no ocupar massa espai i com que ens interessa el funcionament general de la rutina, només hem agafat un dels casos, quan la ordre és la del polsador de l'esquerra. Observem que un cop dins el case corresponent, es van avaluant amb estructures if i else diferents condicions de les variables. I aquestes condicions es poden dividir en condicions secundàries que finalment determinin la modificació a realitzar. Per exemple en el diagrama que il·lustrem, primer s'avalua si el vehicle s'està desplaçant a la dreta. En cas positiu es passa a mirar a quina velocitat va, i en cas negatiu si està avançant en línia recta. La conclusió que es treu és que és una funció difícil de desxifrar i de modificar.

La solució a aquest problema passa per intentar buscar un ordre en la relació entre les variables, de forma que l'estructura quedi molt més simplificada. En primer lloc, agruparem les 3 variables que defineixen l'estat dels motors (*ma, sentit* i *vel)* en una de sola, que li direm *estat*. Al realitzar aquest canvi també forcem un canvi en la funció d'actualitzar els motors i en la d'actualitzar els LEDs, ja que depenen d'aquestes variables. Aquest canvi es tractarà més endavant.

Ara, hem de definir els possible valors que tindrà la variable *estat* a partir dels valors de les variables que representa, per així tenir en compte tots els possibles estats:

Estat	ma	vel	sentit	Descripció
r4	0	4	2	Marxa enrere velocitat màxima
r3	0	3	2	Marxa enrere velocitat 3
r2	0	2	2	Marxa enrere velocitat 2
r1	0	1	2	Marxa enrere velocitat mínima
0	0	0	1	Aturat
1	0	1	1	Endavant, velocitat mínima
2	0	2	1	Endavant, velocitat 2
3	0	3	1	Endavant, velocitat 3
4	0	4	1	Endavant, velocitat màxima
5	2	0	1	Girant a la dreta aturat
6	1	0	1	Girant a l'esquerra aturat
7	2	1	1	Girant a la dreta i velocitat mínima
8	1	1	1	Girant a l'esquerra i velocitat mínima
9	2	2	1	Girant a la dreta i velocitat 2
A	1	2	1	Girant a l'esquerra i velocitat 2
B	2	3	1	Girant a la dreta i velocitat 3
C	1	3	1	Girant a l'esquerra i velocitat 3

Taula 3: Valors de la variable estat

Fem els següents canvis per tal que el funcionament del vehicle sigui més pràctic:

- En comptes de 4 velocitats en tenim 3.
- Es permet fer girs al mateix temps que el cotxe es desplaça cap endavant.
- La marxa enrere només té una velocitat.

estat	ma	vel	sentit	Descripció
r	0	1	2	Marxa enrere
0	0	0	1	Aturat
1	0	1	1	Endavant, velocitat mínima
2	0	2	1	Endavant, velocitat mitja
3	0	3	1	Endavant, velocitat màxima
4	2	0	1	Girant a la dreta amb velocitat de gir 1 i sense desplaçament endavant
5	2	0	1	Girant a la dreta amb velocitat de gir 2 i sense desplaçament endavant
6	2	0	1	Girant a la dreta amb velocitat de gir 3 i sense desplaçament endavant
7	1	0	1	Girant a l'esquerra amb velocitat de gir 1 i sense desplaçament endavant
8	1	0	1	Girant a l'esquerra amb velocitat de gir 2 i sense desplaçament endavant
9	1	0	1	Girant a l'esquerra amb velocitat de gir 3 i sense desplaçament endavant
A	2	1	1	Girant a la dreta amb velocitat de gir 1 i desplaçament endavant 1
B	2	1	1	Girant a la dreta amb velocitat de gir 2 i desplaçament endavant 1
C	1	1	1	Girant a l'esquerra amb velocitat de gir 1 i desplaçament endavant 1
D	1	1	1	Girant a l'esquerra amb velocitat de gir 2 i desplaçament endavant 1
E	2	2	1	Girant a la dreta i desplaçament endavant 2
F	1	2	1	Girant a la dreta i desplaçament endavant 2

Taula 4: Valors de la variable estat modificats

Al tenir concentrats els múltiples estats dels motors en una sola variable, podem implementar una màquina d'estats que només depengui de l'ordre entrada a través dels polsadors. La representació de la màquina d'estats es pot veure a continuació, les circumferències representen els diversos estats dels motors i les fletxes són els possibles valors de la variable ordre.

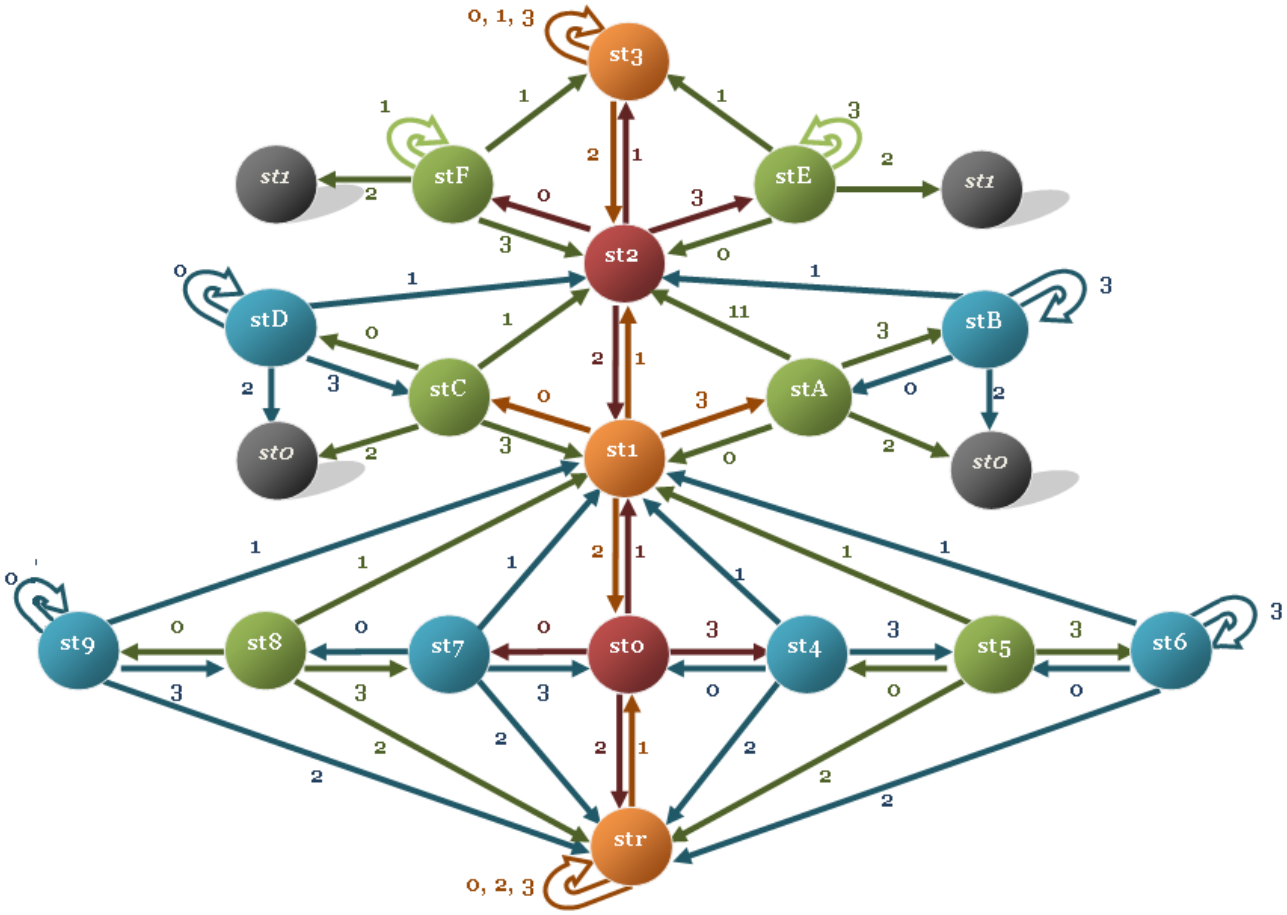


Figura 24: Diagrama de la màquina d'estats

La programació per a implementar la màquina d'estats és molt intuïtiva ja que es pot relacionar directament amb la seva representació:

```
switch(estat){
      case 'x':
      if(ordre==0)
            estat = 'y';
      else if(ordre==1)
            estat = 'w';
      else if(ordre==2)
            estat = 'z';
      else if(ordre==3)
            estat = 'a';
      break;
```

Representem el diagrama de flux de la nova rutina per poder-lo comparar amb el de la rutina anterior. Del switch principal (aquest cop en funció de l'estat) només agafarem un case escollit aleatòriament, donat que la resta són iguals:

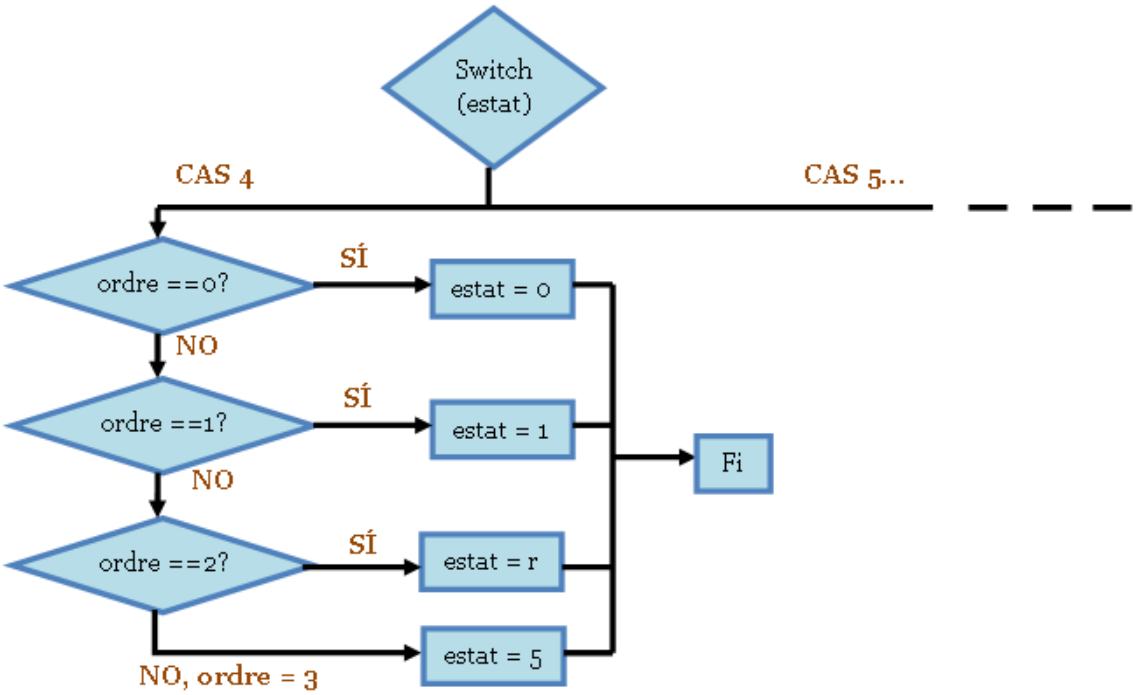


Figura 25: Diagrama de flux de la nova rutina de canvi d'estat dels motors

Podem veure com efectivament el codi està molt més ordenat, i encara que en terme de línies de codi és més llarg que la rutina anterior, és molt més pràctic i intuïtiu. També cal dir que posem aquest fragment de codi dins una funció anomenada *canviEstat()* a la que passarem les variables *ordre* i *estat* i ens retornarà la variable *estat* modificada.

Un cop tenim la nova rutina implementada, hem perdut les variables *blink* i *vel*, i per tant hem de pensar una nova forma de realitzar les funcionalitats que permetien aquestes variables:

- La monitorització de la velocitat del vehicle als LEDs i el control de la velocitat de cada motor. S'explica en l'apartat (5.3).

- La pampalluga dels LEDs al prémer un polsador que no ocasiona un canvi d'estat, que s'explica més endavant en aquest mateix apartat.

Estructura del diagrama de flux

Abans de començar a explicar el desenvolupament d'aquesta part, cal fer referència a què degut a una incidència descrita a la planificació, es va fer un canvi de les plaques d'avaluació i es va començar a treballar amb les plaques Duemilanove. Aquest fet ens provoca una inversió de temps bastant elevada en familiaritzar-nos amb el nou entorn de programació i les noves plaques, però que la llarga ens ho compensa. En aquest apartat, el fet de programar les plaques amb Java i no amb C++ fa que encara es pugui simplificar més tota l'estructura del diagrama de flux, ja que el llenguatge és més pràctic.

Començarem analitzant el diagrama de flux del projecte inicial, que es pot veure il·lustrat i descrit al principi d'aquest apartat (Figura 22):

En un primer lloc, veiem que amb les plaques Duemilanove, el funcionament de la recepció de paquets és diferent al de les 13192-SARD. Amb les anteriors s'havia d'activar la recepció de paquets durant un temps determinat perquè el programa havia d'estar pendent de rebre dades a través del canal de comunicació, i no podia realitzar altres accions mentrestant. En canvi, amb les Arduino només és necessari activar la recepció de paquets un cop i sempre es manté activa mentre el microprocessador pot estar realitzant altres funcions. Quan arriba un paquet al microcontrolador, es guarda en un buffer on pot ser llegit en el moment que es vulgui, i després s'esborra la informació per deixar lloc a un paquet nou. En aquest sentit, ja no hem de tenir en compte el concepte de timeout a l'hora de rebre dades.

En segon lloc, veiem que la variable que es transmet del comandament al vehicle és la ordre que prové dels polsadors. Per tant, s'ha de calcular el nou estat dels motors tant a la placa transmissora per actualitzar els LEDs, com a la placa receptora per actualitzar motors i LEDs. Si en comptes d'enviar la ordre enviem directament l'estat, només haurem de realitzar el càlcul de l'estat dels motors a la placa transmissora, evitant aquesta redundància i permetent que la placa receptora estalviï cicles de rellotge fent aquest càlcul. A la següent imatge queda representat aquest canvi:

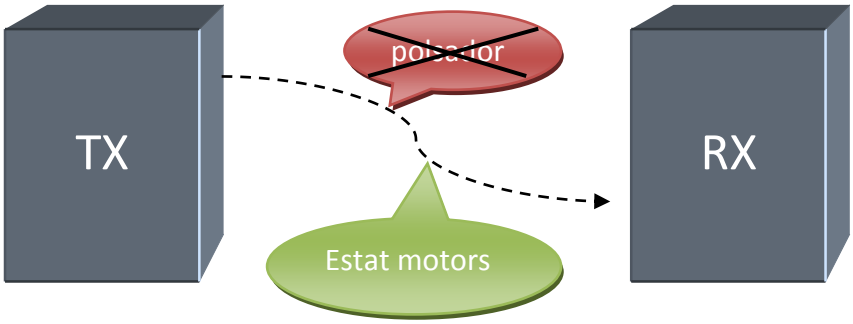


Figura 26: Canvi de la informació transmesa

Com que ja tenim decidit on col·locar la funció que actualitza l'estat dels motors, podem afegir la funcionalitat de blink. El què fem és, al transmissor, abans de cridar a la rutina *canviEstat()*, guardar l'estat actual en una variable *state_ini*. Després d'executar la funció que ens retorna el nou estat, el comparem amb l'inicial (*state* vs *state_ini*) i si és el mateix apaguem els LEDs durant 250ms. Després saltem directament a la funció d'actualització dels LEDs que els tornarà a encendre tal i com estaven perquè l'estat no ha variat. Per aquest motiu tampoc caldrà transmetre l'estat a la placa receptora. En aquest sentit només farà una pampalluga la placa transmissora, però és la que ens interessa perquè és la que rep les ordres de l'usuari.

Per últim valorem la necessitat d'establir un procediment de confirmació de paquets rebuts (ACKs) a la capa d'aplicació, i veiem que no és necessari per aquesta sèrie de motius:

- L'estàndard Zigbee (IEEE 802.15.4) ja disposa d'un mètode de confirmació de recepció a nivell de trames. Intrínsicament al protocol, el receptor sempre envia una trama de retroalimentació cap al transmissor per informar-li de que la trama ha arribat correctament.

- En el nostre sistema només hi ha un transmissor i un receptor, no hi intervenen més dispositius que puguin enviar o rebre paquets. Per tant, el destinatari sempre serà el correcte i no tindrem interferències.

- Els paquet enviats només contenen un caràcter (8 bits), al ser poca informació és poc probable que es corrompi.

- Si degut a algun error aïllat i puntual el paquet arriba defectuós, l'estat no serà identificat i els motors passaran a l'estat per defecte, aturats. De forma que no hi haurà un comportament inesperat del vehicle. Si el paquet es perd l'usuari ho notarà de seguida i tornarà a enviar la ordre, no hi ha risc d'estancament.

La conclusió és que no és necessari un sistema d'assentiment de recepció, degut a la seva baixa probabilitat d'errors i perquè la informació es pot tornar a enviar fàcilment.

Ara apliquem tots aquests canvis proposats al nou diagrama de flux:

- Si l'estat no varia no cal transmetre'l al receptor.

- Calcular l'estat dels motors només al transmissor i abans de transmetre el paquet, que contindrà el nou estat en comptes de la ordre, al receptor. Eliminar la rutina per calcular l'estat dels motors allotjada al receptor.

- Eliminar els casos on es valora la possibilitat que es produeixi un timeout.

- El receptor ja no transmetrà un ACK després d'actualitzar els motors. Per tant el transmissor ja no esperarà rebre la confirmació.

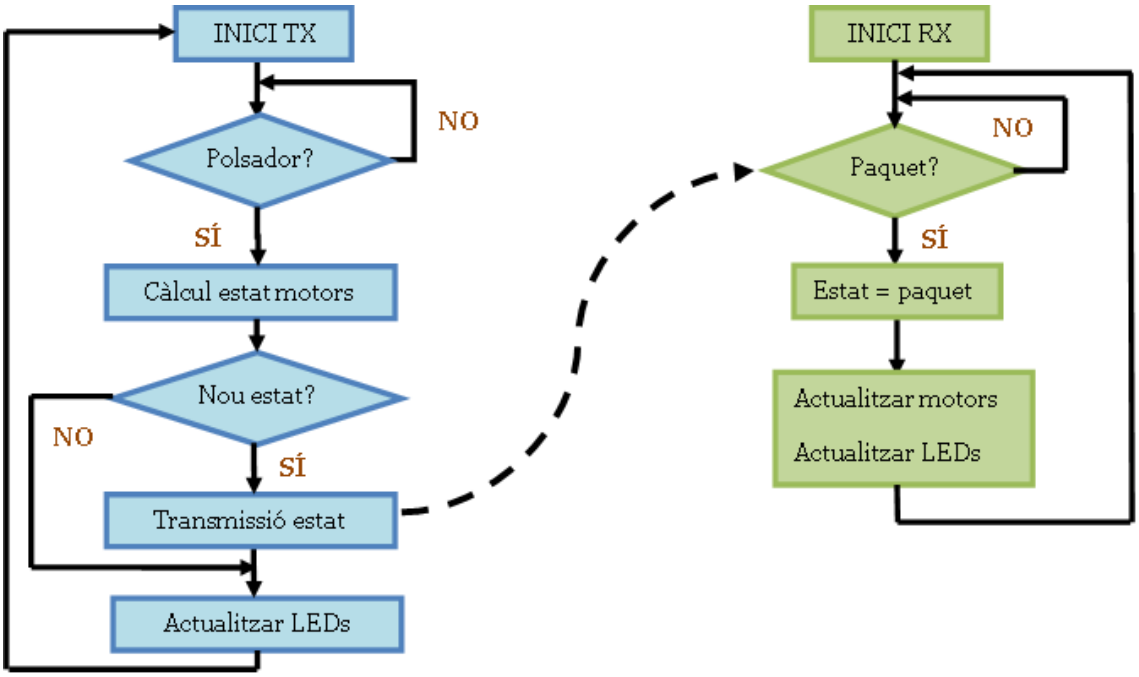


Figura 27: Nou diagrama de flux del prototip

A primer cop d'ull es pot veure com l'estructura és molt més simple, i per tant, més ràpida. El fet de només tenir comunicació en una direcció (Tx -> Rx) suposa no tenir informació de l'estat en què es troba el receptor, però no hi haurà problemes de sincronització perquè el temps entre un paquet enviat i el següent ve determinat per l'usuari que prem un dels polsadors. Aquesta acció és suficientment lenta, des del punt de vista del microprocessador, com perquè tingui temps d'actualitzar totes les seves sortides. L'avantatge que suposa transmetre informació únicament quan hi ha un canvi d'estat, és aprofitar la capacitat "d'adormir-se" que tenen els dispositius ZigBee quan porten una certa estona inactius. Ens interessa que estiguin en aquest estat "d'standby" el màxim de temps possible perquè es redueix bruscament el seu consum de corrent. Enviant només la informació necessària és com aconseguim optimitzar aquesta característica.

5.2 Marxa enrere

En el prototip que hem pres com a punt de partida es va dissenyar un mètode per tal que pogués efectuar la marxa enrere, però finalment no es va implementar. No es va fer perquè era necessari fer servir relés que són molt costosos i els pins del microcontrolador tampoc cedeixen potència suficient activar-los. Per aquest mateix motiu nosaltres també descartem aquesta proposta i en pensem una altra, implementar un pont H.

El pont H és un dispositiu electrònic pensat especialment pel control del sentit de gir en motors DC, com és el nostre cas. Es composa bàsicament per resistències i transistors PNP i NPN. L'estructura i funcionament del circuit es mostra a continuació:

- Si cap de les entrades (S1 o S2) està a alta, no hi ha efecte sobre el circuit. Tots els transistors es tallen i el motor queda aïllat de la font, per tant està aturat.

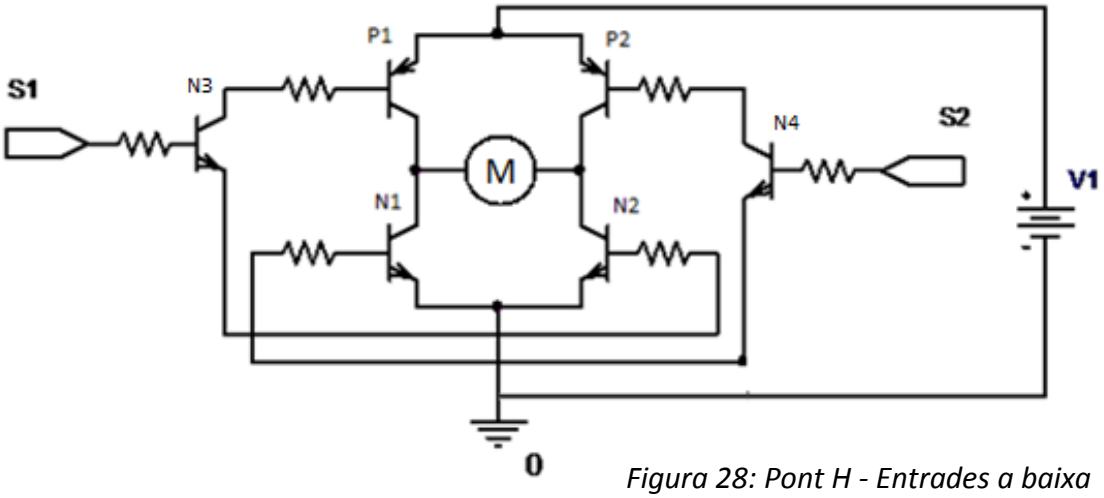


Figura 28: Pont H - Entrades a baixa

- Quan l'entrada S1 està en alta, s'activa el transistor N3, que ataca a les portes dels transistors P1 i N2, activant-los i deixant passar el corrent de la bateria a través del motor. Com que S2 està a baixa, els transistors N4, P2 i N2 estan tallats de forma que obliguen al corrent a passar per P1 i N2.

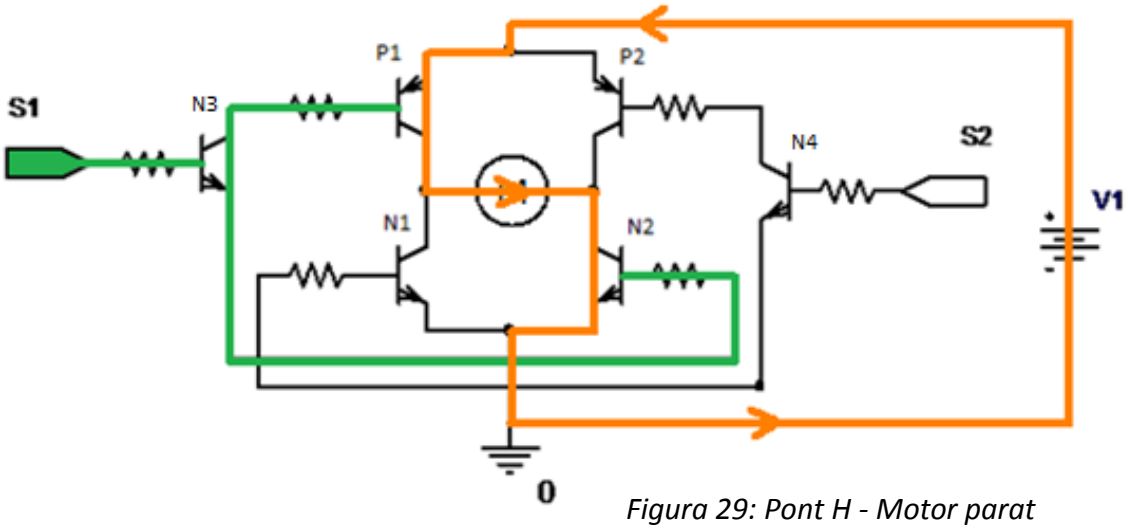


Figura 29: Pont H - Motor parat

- El cas contrari és simètric a l'anterior. L'entrada S2 està activa, de forma que el transistor N4 activarà els transistors P2 i N1, que deixaran pas al corrent subministrat per la bateria a través del motor. La posició dels transistors fa que ara el corrent travessi el motor en el sentit contrari que en el cas anterior, per tant, girarà al revés. Tenim controlada la marxa endavant i la marxa enrere del motor només amb dues entrades (S1 i S2).

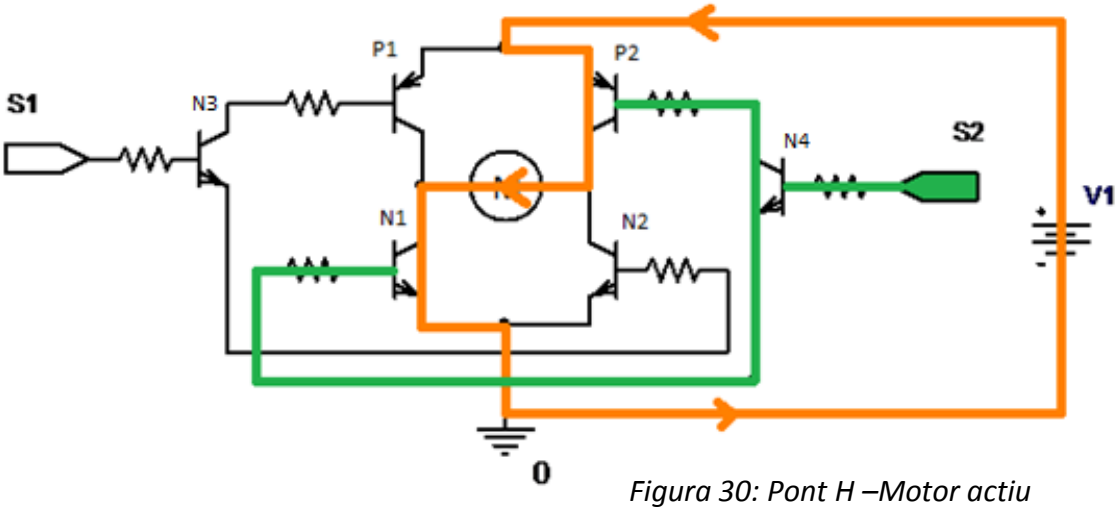


Figura 30: Pont H –Motor actiu

- Si les dues entrades estan actives alhora, tenim que tots els transistors es posaran a conduir. En aquest moment, la intensitat subministrada per la font passarà quasi tota a través dels camins creats pels transistors, ja que la seva resistència en conducció és molt menor que la del motor. Això juntament amb que la font té un voltatge elevat, provocarà una intensitat molt elevada que els transistors no suportaran, cremant-se. Prohibirem aquest estat.

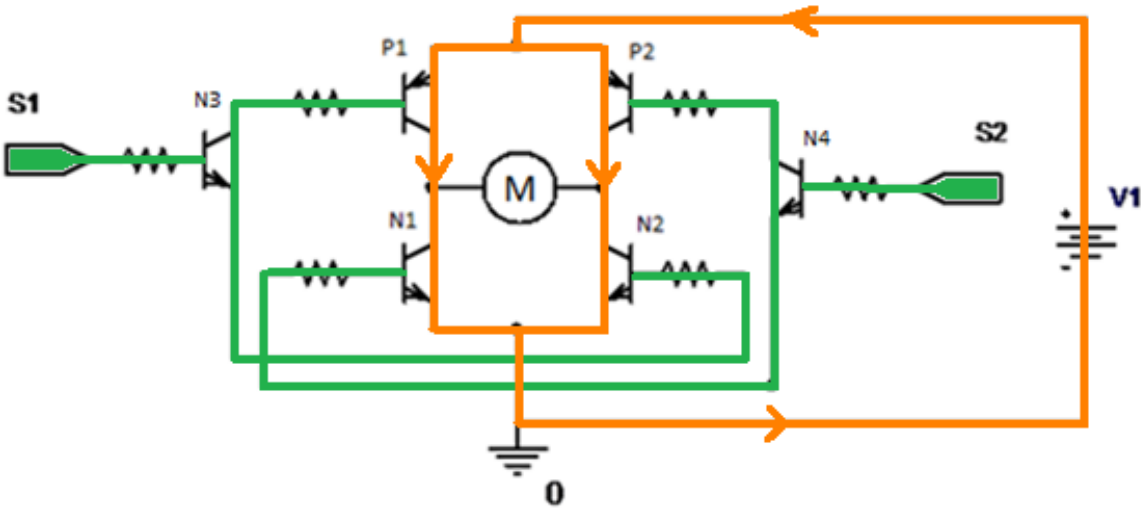


Figura 31: Pont H – Estat prohibit

Amb aquesta informació podem construir una taula per a les entrades i d'un dels ponts H, hem de tenir en compte que tenim un per a cada motor:

S1	S2	Descripció
0	0	Motor aturat
0	1	Marxa endavant
1	0	Marxa enrere
1	1	Prohibit

Taula 5: Possibles entrades del pont H

Tria de components

Per tal d'escollir quins transistors i quines resistències necessitarem, ens fixem en la part central del circuit, els 4 transistors que són travessats directament per la corrent del motor quan està actiu. En primer lloc calcularem el valor màxim que pot tenir aquesta intensitat, que ve donada per la potència del motor i el voltatge al qual s'alimenta:

La potència del motor és de 1.62W i treballa entre 12 i 24V. Alimentant el motor amb el voltatge mínim tindrem la intensitat màxima de col·lector:

$$I_{MAX} = \frac{P}{V_{MIN}} = \frac{1.62W}{12V} = 135mA$$

El voltatge màxim entre col·lector i emissor que pot caure en cadascun dels transistors és produeix quan no hi ha conducció de corrent, i és la tensió total de la bateria, en aquest cas, 24V.

Els transistors que ens poden anar bé són els 2N2222 (NPN) i 2N2907(PNP). Suporten un màxim de 40V en inversa i una intensitat de col·lector en activa de 600mA. Els límits doncs, cobreixen les nostres necessitats.

Ara hem de triar les resistències de base. El següent circuit mostra la configuració d'un dels transistors, hem d'assegurar que sempre treballarà en la zona de saturació. També cal destacar que hem considerat que la caiguda de tensió dels transistors que no es troben en el centre del pont H és nul·la. En primer lloc sabem que per tal de treballar en saturació, un transistor bipolar ha de complir que:

$$\beta \cdot I_B > I_C$$

Sabent que la intensitat de col·lector màxima serà de 135mA, i suposant una β de 100, la intensitat de base mínima ha de ser:

$$I_{B,MIN} = \frac{I_{C,MAX}}{\beta} = 1.35mA$$

Per últim, definim el valor de la resistència de base per tal que com a mínim hi circuli la intensitat mínima que acaben de trobar, quan s'activi el pin adequat, que proporciona un voltatge de 5V:

$$R_{B,MAX} = \frac{V_{PIN}}{I_{B,MIN}} = \frac{5V}{1.35mA} = 3.70k\Omega$$

Donat que el voltatge subministrat per les sortides de la placa pot variar fins a 3.3 en funció de la seva alimentació, posarem resistències d'1kΩ per assegurar que es sobrepassa el corrent de base mínim.

Per finalitzar, els 2 transistors que controlen l'activació dels 4 centrals, estan sotmesos a tensions i intensitats molt més baixes. Per ells hi circula la intensitat de base dels altres transistors, que hem vist que és petita, i a proporció, el voltatge tampoc serà molt gran. Per això podem dir que sempre treballaran en saturació, dos transistors 2N2222 serviran per a aquesta posició. També cal comentar la seva resistència de base, que la posem una 5 vegades més gran que les altres resistències de base (4.7kΩ). Així garantim que a la intensitat que es genera al col·lector dels altres transistors no és massa elevat, però que compleix amb els 1.35mA mínims.

Simulacions

Fem simulacions amb PSpice per assegurar que el circuit tindrà el funcionament que volem. Notar el motor té una resistència equivalent de 300Ω, tal i com indica el fabricant. I que per tal d'assegurar el funcionament en el pitjor cas, hem posat que les sortides de la placa siguin de 3V, és a dir, que la bateria de la placa està molt baixa.

- Només una sortida activada, en aquest cas la corresponent a S1 que hem vist anteriorment. Pràcticament tota la tensió de la bateria cau als extrems del motor, que és travessat per una intensitat de 123.3mV al igual que els transistors Q12 i Q8. Els altres 2 transistors de la part central estan tallats, i així es reflexa en el valor d'uns 20pA que hi circula.

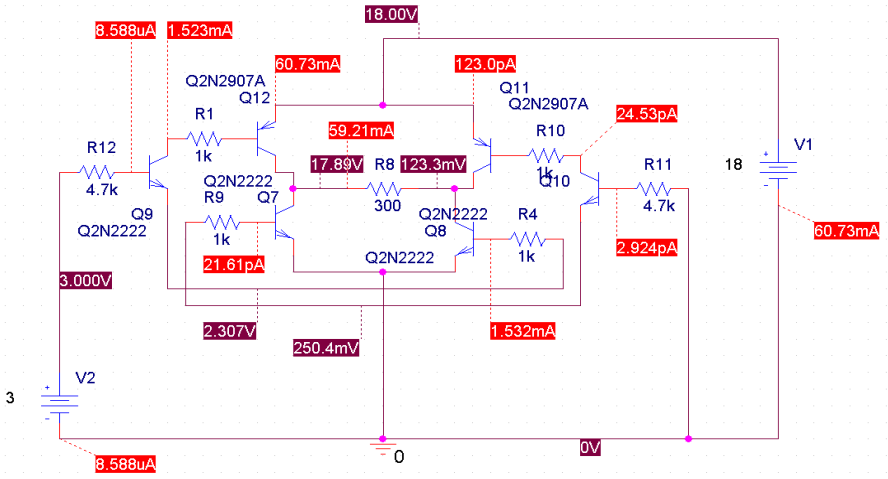


Figura 32: Pont H – Simulació motor actiu

- Las dues entrades a alta. En aquest cas la tensió de la bateria cau als 2 extrems del motor, fent que no hi hagi una diferència de voltatge i que no hi circuli una intensitat que el faci girar. El corrent circula per tots els transistors centrals i té un valor màxim de 237mA. Per aquest motiu es prohibeix posar els dos pins a alta, però ara veiem que els transistors suportarien la intensitat.

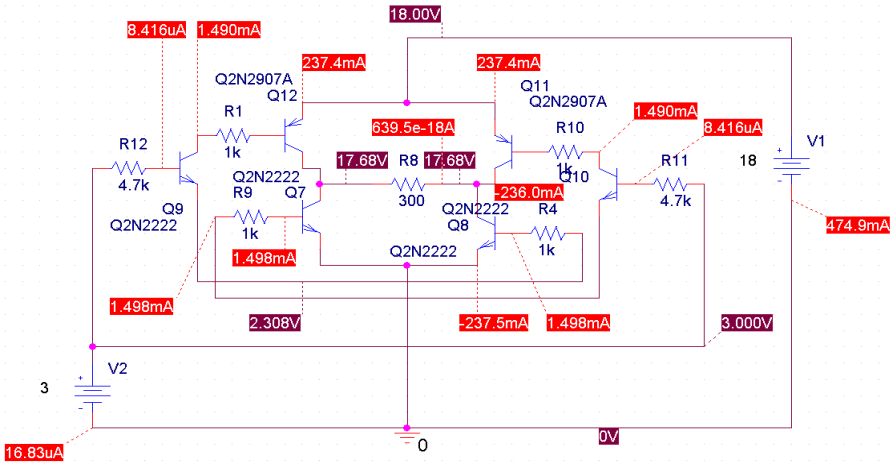


Figura 33: Pont H – Simulació estat prohibit

- Es tracta del cas contrari a l'anterior, les dues entrades a baixa. Ara tenim que cap dels transistors està actiu, i per tant no hi ha circulació de corrent en cap dels sentits. A la imatge es pot veure com tots els corrents implicats són de l'ordre de pA, menyspreables.

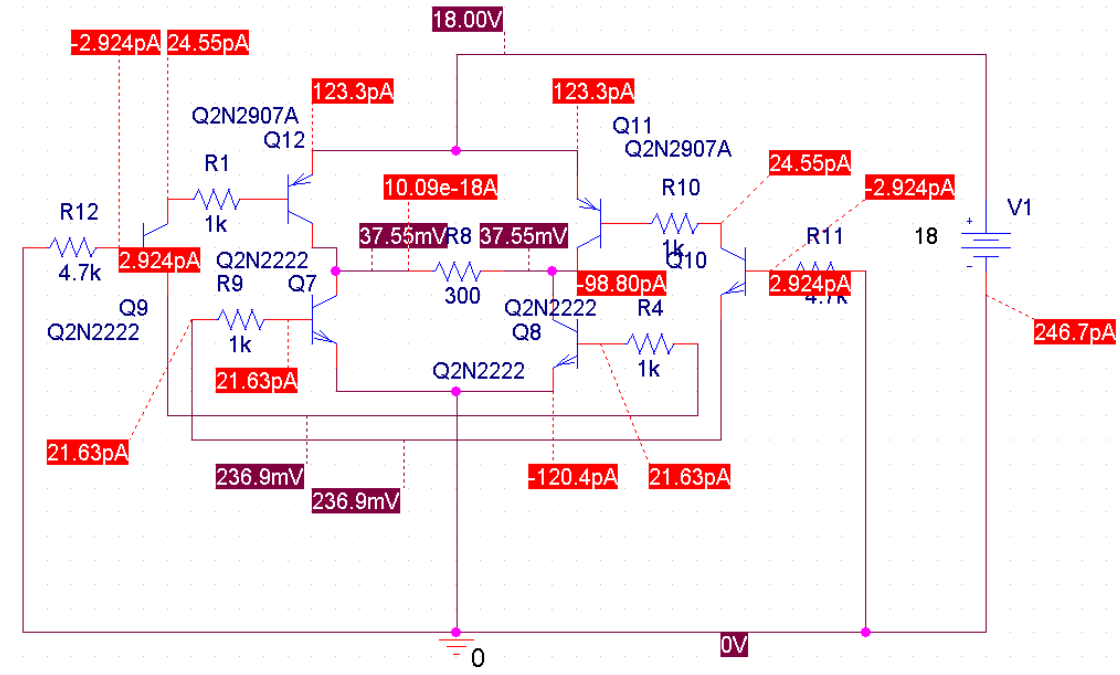


Figura 34: Pont H – Simulació motor aturat

Muntatge

Un cop verificat amb la simulació que els càlculs teòrics són correctes, comprem els dispositius i en fem dos prototips en dues plaques perforades. Tindrem dos circuits, un per a cada motor.

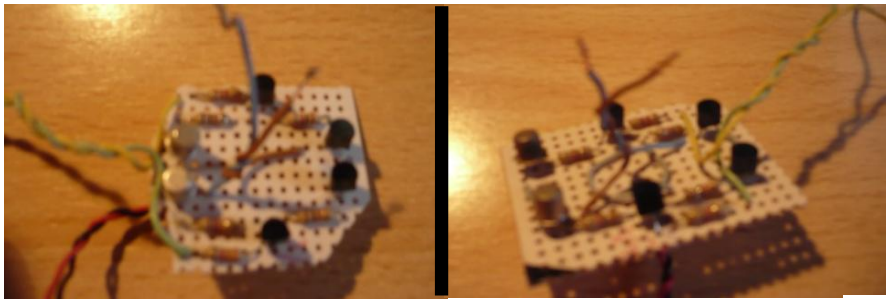


Figura 35: Ponts H construïts sobre plaques perforades

Ja tenim els drivers necessaris per a controlar el sentit dels motors i la seva velocitat.

5.3 Actualització dels motors i dels LEDs

El projecte inicial basava el control de la velocitat de cada motor en generar una senyal de polsos de diferents freqüències, que varien en funció de la variable velocitat, i que ataquen a les bases dels transistors que controlen el pas del corrent de la bateria cap als motors. Trobem una explicació funcional d'aquesta rutina a la memòria del projecte, però no apareix en el codi font. Per això, decidim fer des de zero aquesta funció. També veurem com actualitzar els LEDs per tal de monitoritzar la velocitat ara que no tenim la variable *vel*.

Començarem per l'actualització dels LEDs. A l'apartat anterior hem vist que tenim 3 possibles velocitats, i disposem de 4 LEDs. El què farem serà fer-ne servir 3 per indicar la velocitat i un per a la marxa enrere, quedant representades totes les velocitats de la següent forma, on cal destacar que la marxa enrere només té la velocitat mínima:

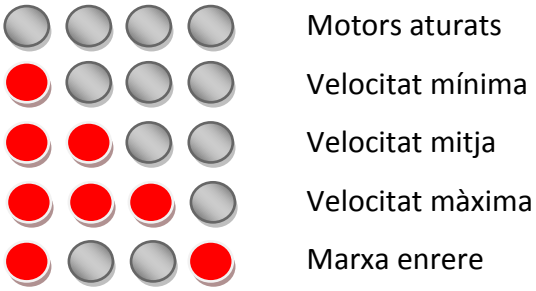


Figura 36: Nova representació de la velocitat

Ara hem de trobar la relació entre la variable *estat*, la velocitat que ha de tenir cadascun dels motors i la velocitat representada als LEDs, que serà la del motor més lent. Ho fem amb la següent taula, on la velocitat pot valdre 0 (motor aturat), 1 (velocitat mínima), 2 (velocitat mitja), 3 (velocitat màxima) i 4 (marxa enrere):

Estat	Motor esquerre	Motor dret	Velocitat representada
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
4	0	1	0
5	0	2	0
6	0	3	0
7	1	0	0
8	2	0	0
9	3	0	0
A	1	2	1
B	1	3	1
C	2	1	1
D	3	1	1
E	2	3	2
F	3	2	2
r	4	4	4

Taula 6: Velocitat dels motors en funció de la variable estat

Per monitoritzar la velocitat farem servir la mateixa rutina que es va implementar en el projecte anterior, on se li passa la variable *vel* i s'encenen els LEDs que corresponen. Només modificarem els LEDs que s'encenen per a cada velocitat per tal que la representació sigui la que hem triat abans. A aquesta funció l'anomenem *LedsOn()*. Ara bé, ja no tenim la variable *vel*, per tant farem una funció que seguint la taula (X) executi la funció *LedsOn()* amb el paràmetre que correspon en funció de l'estat:

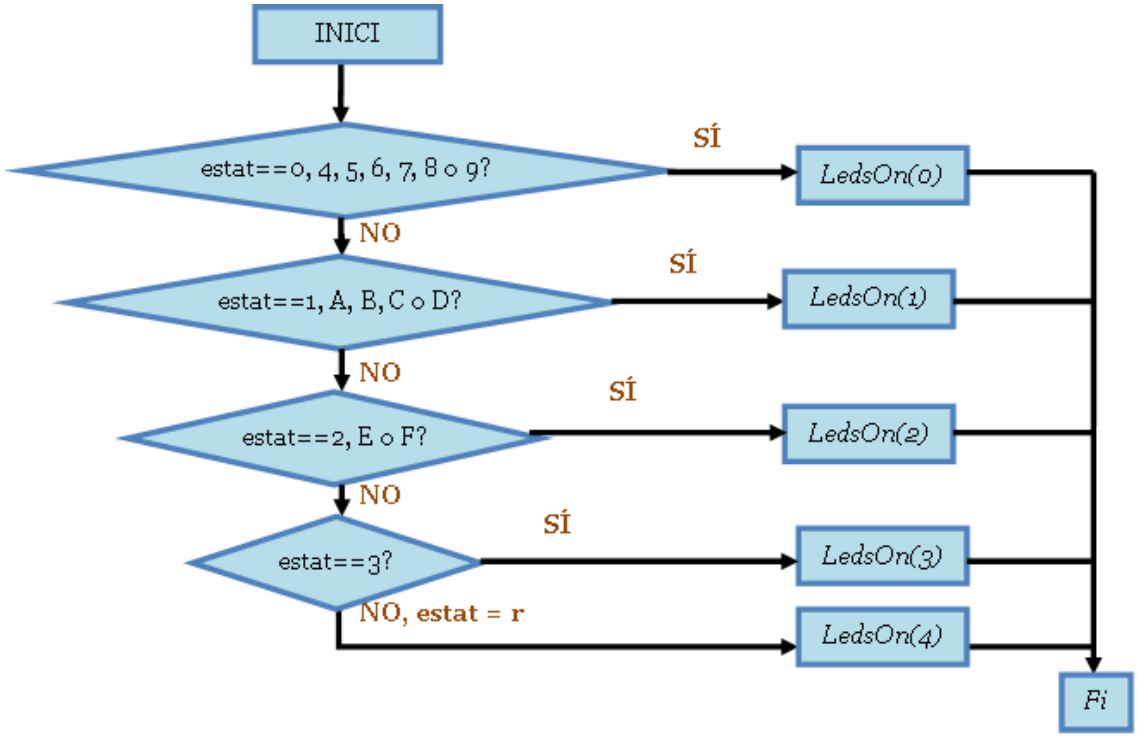


Figura 37: Diagrama de flux de la rutina de monitorització de la velocitat

Només falta veure com actualitzar les sortides que encenen els motors. A l'apartat anterior hem vist que són 4 sortides de la placa d'avaluació que ataquen al circuit de control dels motors. Farem servir els pins PWM de la placa Duemilanove. A continuació podem veure un petit esquema del procés de control dels motors.

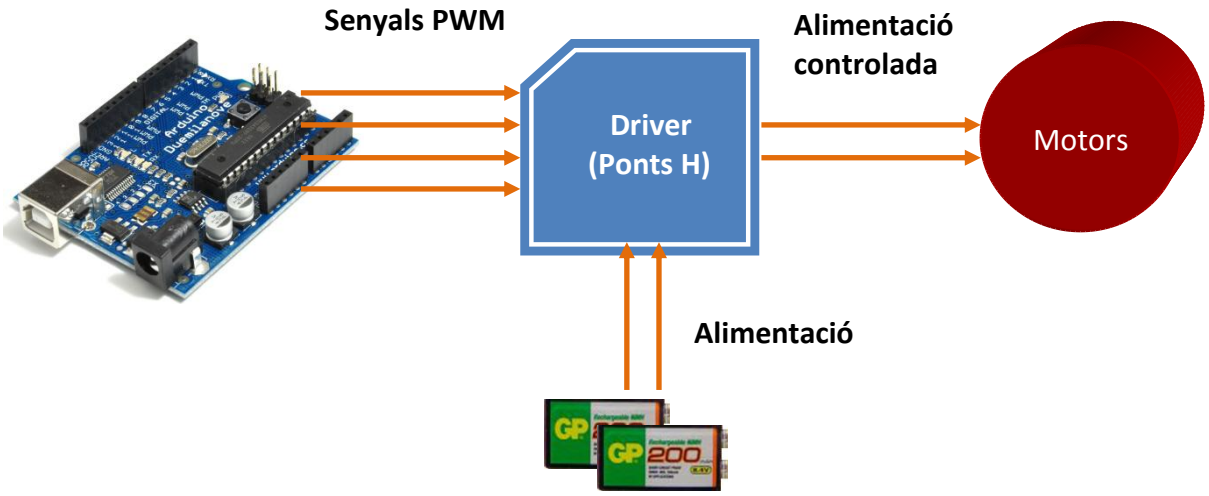


Figura 38: Esquema de control sobre els motors

Els ports PWM, descrits a l'apartat (3.2 Recursos per desenvolupar el nou prototip), s'activen mitjançant la funció *analogWrite()* amb un enter del 0 al 255. L'energia entregada als motors serà proporcional a aquest número (0 → no hi ha alimentació, 64 → 25% de l'alimentació, 255 → tota l'alimentació), però no podem fer aquesta atribució directament, ja que per a voltatges molt baixos els motors no tenen suficient força per desplaçar el vehicle. Hem de trobar la potència mínima a la que els motors comencen a tenir força com per desplaçar el prototip, i ho fem amb un petit programa que va augmentant el valor de les sortides dels ports PWM i l'imprimeix a la consola. Quan veiem que el prototip comença a desplaçar-se, el valor imprès a la consola en aquell moment serà el mínim.

A partir del valor mínim de l'*analogWrite()* trobem per interpolació el valor proporcional a la velocitat mitja, i també en podem extreure el valor aproximat del voltatge que cau als motors. Ho anotem tot a la següent taula:

Velocitat	analogWrite()	Percentatge bateria	Voltatge motors
1	190	74,5%	13,4V
2	230	90,2%	16,2V
3	255	100%	18V

Taula 7: Intervals de velocitat

Finalment, la rutina que actualitzarà els motors consisteix en saber en quin estat ens trobem, i quina velocitat correspon a cada motor segons la taula 6:

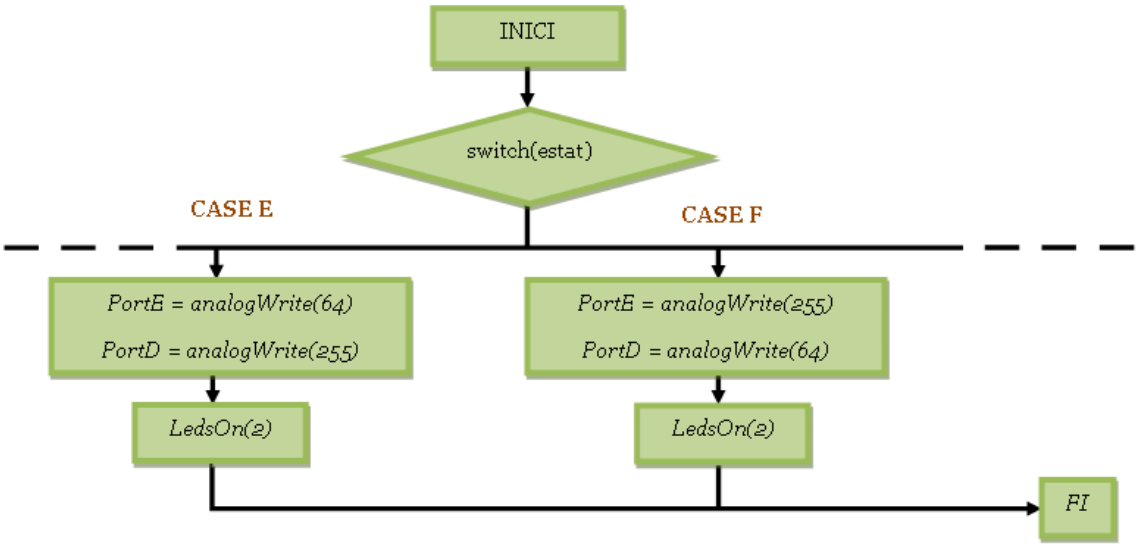


Figura 39: Diagrama de flux de la rutina d'actualització dels motors

Cal tenir en compte que aquesta rutina només es té al receptor, i com es pot veure, s'hi ha integrat la rutina d'actualització dels LEDs.

5.4 Control acceleròmetre

Per tal d'implementar aquesta millora disposem d'un acceleròmetre ADXL335 que connectarem a les entrades analògiques de la placa transmissora tal i com es mostra a la següent imatge, ja que és la placa amb la que controlem el vehicle. Volem que el control del vehicle sigui mitjançant el moviment de la placa.

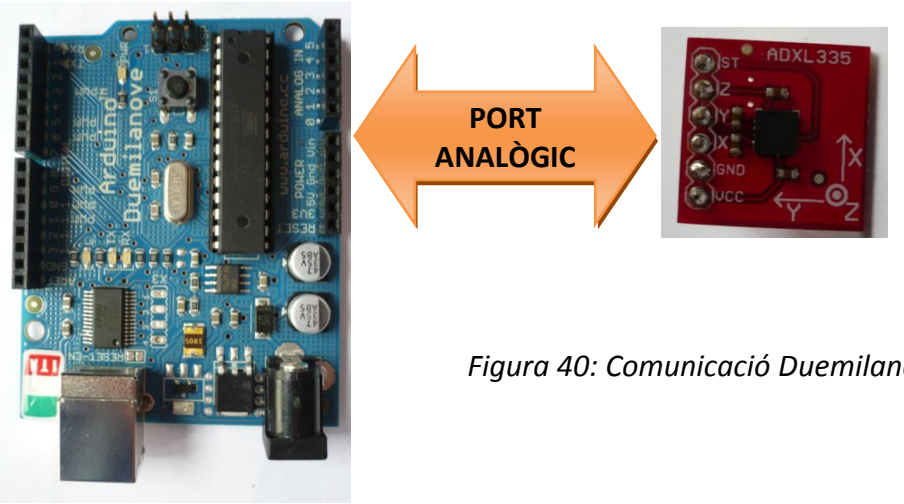


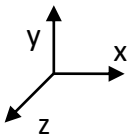
Figura 40: Comunicació Duemilanove - ADXL335

Primer escrivim i carreguem a la placa un petit codi que va llegint les variables associades als ports X Y Z de l'acceleròmetre i les envia a l'ordinador a través de la connexió serial, de forma que amb la consola les podem visualitzar.

Anem posant la placa en diverses posicions i mirem quin valor tenen les sortides:

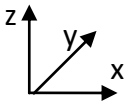
- De cara a l'usuari. Serà la posició inicial, on el cotxe estarà aturat:

X = 505
Y = 610
Z = 510



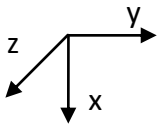
- Col·locada cap amunt

X = 505
Y = 500
Z = 610



- Girada 90º a la dreta respecte la posició inicial

X = 395
Y = 495
Z = 510



Després de diverses proves com les d'abans, acabem veient que el valor que pren la sortida per a qualsevol dels tres eixos depèn de l'angle de l'eix d'aquesta forma:

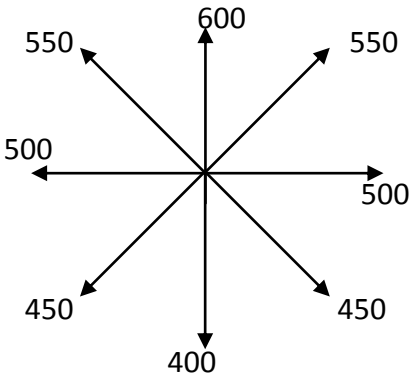
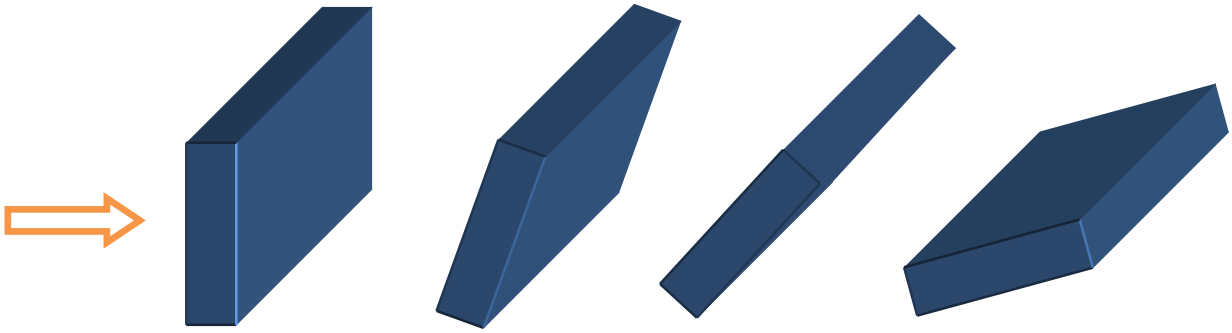


Figura 41: Valors de sortida de l'acceleròmetre

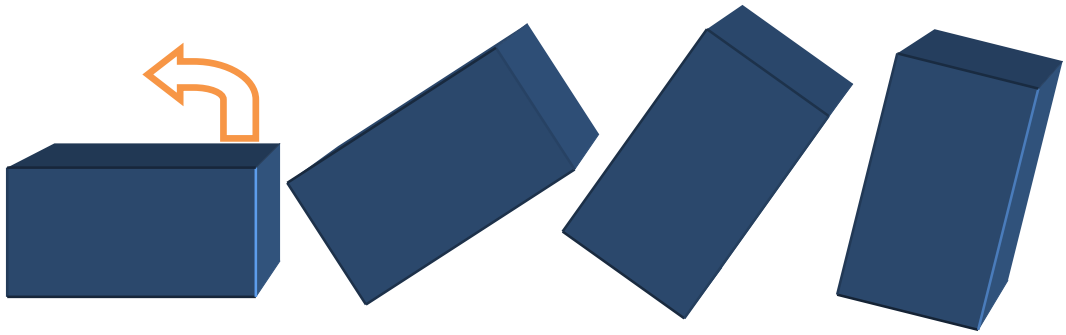
Amb aquestes dades ja podem representar qualsevol orientació de la placa mitjançant els valors dels eixos. També veiem que fent servir dos dels eixos ja en tenim prou, el Z per inclinar la placa endavant i endarrere, i l'X per rotar la placa a dreta i esquerra. Amb això definirem 4 moviments bàsics a partir del repòs:

- Acceleració endavant Ens fixem en la variació d'increment en l'eix z.



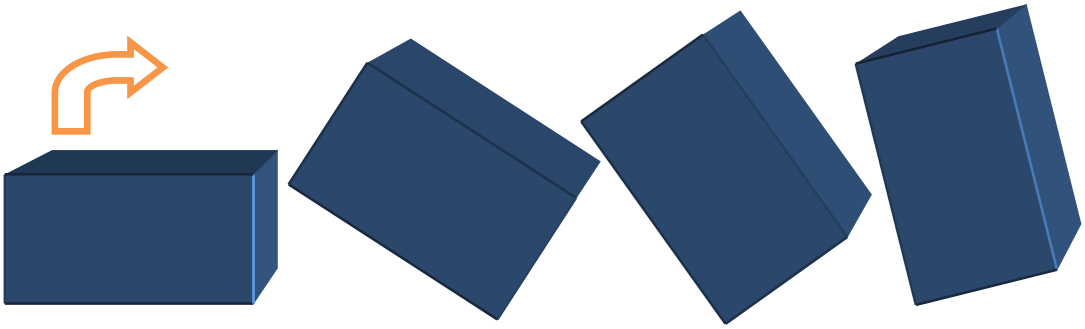
Velocitat	0	1	2	3
Z	510	540	575	600

- Gir cap a l'esquerra. Ens interessa la variació de l'eix x, també incrementa.



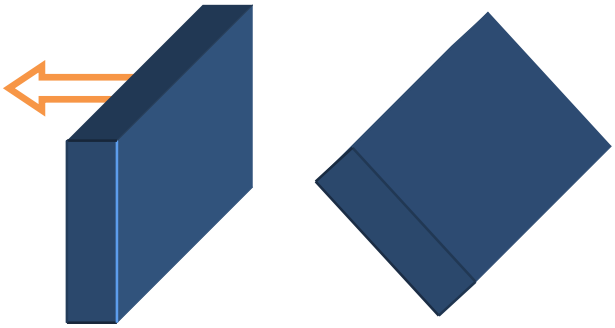
Velocitat de gir	0	1	2	3
X	505	550	570	590

- Gir cap a la dreta. És el cas contrari a l'esquerra, els valors decreixen.



Velocitat de gir	0	1	2	3
X	490	470	450	415

- Marxa enrere. Al contrari que amb la marxa endavant, l'angle de l'eix Z decreix.



Velocitat	0	4
Z	510	440

Tots aquests moviments parteixen de l'estat de repòs, cal tenir en compte que varien lleugerament al combinar-se entre sí. Per exemple si estem avançant cap endavant amb la velocitat mínima, i girem a la dreta, pel fet de tenir la placa lleugerament inclinada endavant els valors de gir cap a la dreta es veuen afectats parcialment. Per aquest motiu anem programant tots els moviments experimentalment amb l'ajuda de la consola, mirant quina posició correspon a cada estat i quins valors han de tenir les variables de l'acceleròmetre.

El següent pas és actualitzar la màquina d'estats ja que ara no variarà l'estat en funció de la ordre, sinó en funció de les mesures obtingudes per l'acceleròmetre.

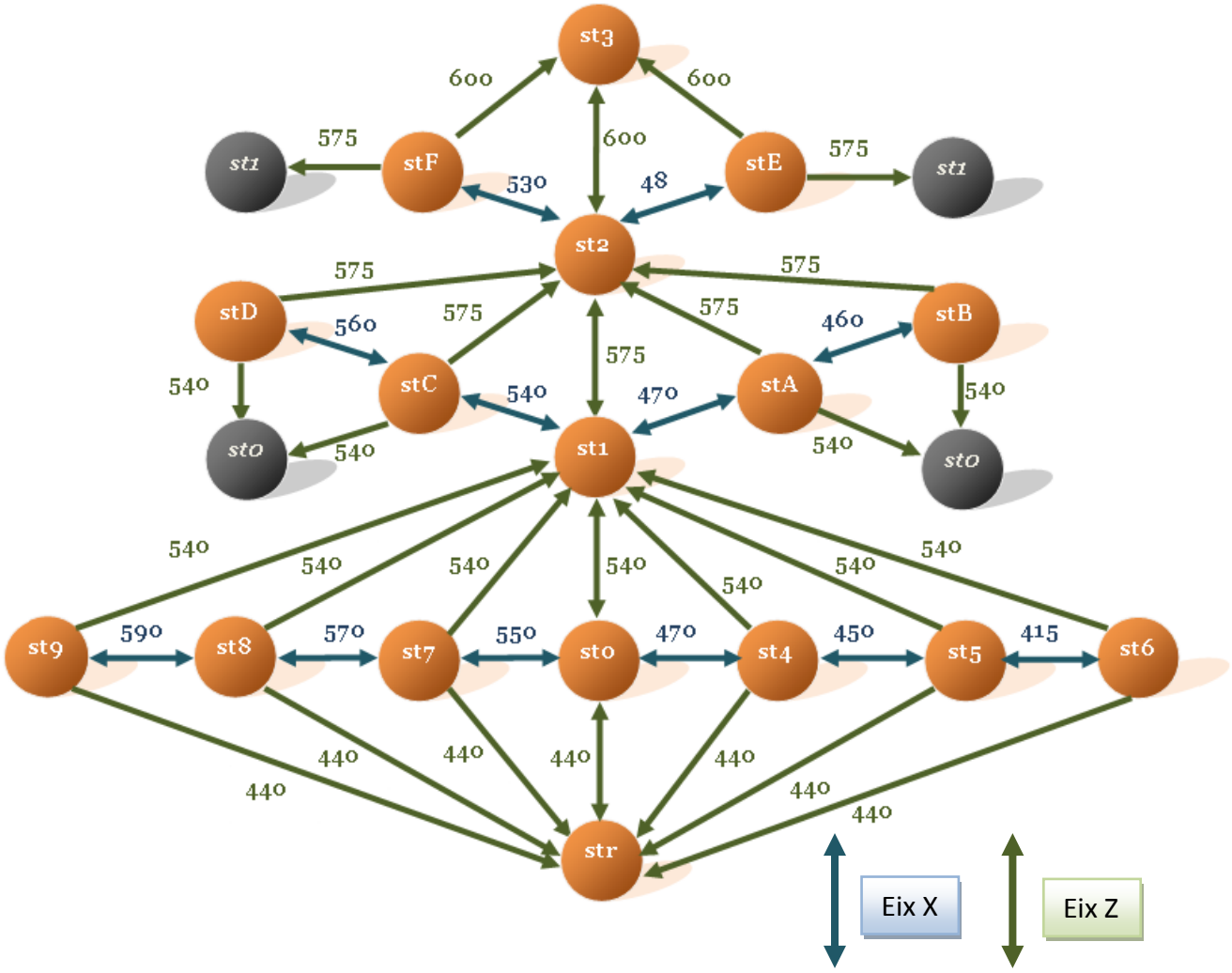


Figura 42: Diagrama de la màquina d'estats segons la lectura de l'acceleròmetre

Com podem veure, els canvis d'estats ja no venen determinats per prémer un polsador concret, sinó per travessar un llindar dels valors que ens va donant l'acceleròmetre. Per aquet motiu hi ha transicions bidireccionals, per exemple en el canvi de st1 a st2, si Z és major de 575 passarem de st1 a st2, i si Z és menor de 575, de st2 a st1. Les comparacions no són doncs d'equivalència sinó de valors més grans o més petits que un valor determinat.

Per últim, cal examinar el diagrama de flux i modificar l'estructura principal, perquè ara ja no és l'usuari humà que prem un polsador quan desitja canviar d'estat, sinó que els acceleròmetres estan contínuament prenent mesures de la posició de la placa. Els canvis que fem són:

- Ja no hem de llegir els polsadors, sinó guardar les mesures dels eixos de l'acceleròmetre per tal que un cop dins de la funció *canviEstat()* no variïn i ens provoquin efectes indesitjats.
- Com que l'usuari no decideix quan executa la ordre, els acceleròmetres es llegeixen contínuament. No tenim un temps d'espera mínim entre dues transmissions que ens garanteixi que al receptor ja s'han realitzat totes les accions corresponents. Per evitar problemes afegim un retard de 200 ms abans de llegir l'acceleròmetre, ho fem amb la funció *delay()* que incorpora Arduino, on es traspassa un valor en mil·lisegons on la placa esperarà sense fer cap acció.

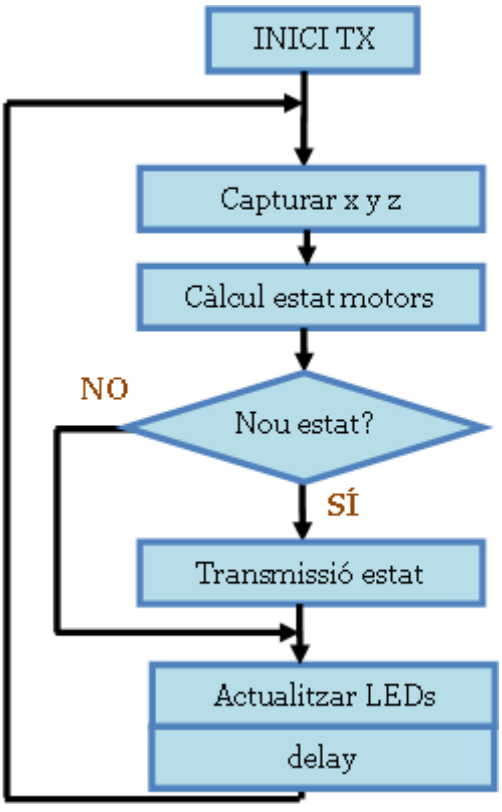


Figura 43: Diagrama de flux de la placa transmissora amb acceleròmetre

5.5 Reproducció d'un recorregut

Amb aquest desenvolupament volem ser capaços d'aconseguir que el vehicle tingui memòria, és a dir, que recordi el recorregut que ha seguit, i el pugui tornar a fer automàticament a petició de l'usuari.

Guardar un recorregut

El primer que hem de fer és definir els límits i les condicions d'aquesta memòria, en el nostre cas només podrà guardar un recorregut, cada cop que es vulgui emmagatzemar un circuit s'esborrarà l'anterior. La forma de guardar aquest recorregut serà automàtica, el vehicle sempre guarda el camí que està fent fins que l'usuari ordena l'acció de reproduir-lo. Després es començarà a repetir cíclicament tot el recorregut fins que l'usuari decideixi aturar la reproducció. En aquest punt, el vehicle esborra el recorregut emmagatzemat i comença a guardar el nou fins que s'ordena la seva reproducció, i així successivament.

Per guardar el recorregut tindrem a la placa receptora un vector d'estats (*st[]*), un vector de temps (*time[]*) i un índex *i*. Cada cop que aquesta placa rebi un paquet amb un nou estat, el guardarem a la posició *i* d'*st[]* i al mateix moment guardarem a la casella *i* de *time[]* el temps què ha passat des de l'últim canvi d'estat. Això ho fem amb la funció *millis()* que es troba per defecte a les llibreries d'Arduino, que retorna els mil·lisegons que han passat des de que s'ha iniciat el microcontrolador.

Per calcular la diferència de temps entre dos estats farem servir una variable *time_ini,* que pren el valor del temps total transcorregut de quan es va rebre l'últim estat. Així, quan arriba un nou estat només s'ha de fer la operació *millis() –time_ini* per tenir la diferència de temps. Nosaltres treballarem amb *millis()/10*0 per tenir dècimes de segon, ja que no ens interessa tanta precisió. Després incrementem l'índex *i.*

Sobre la funció *millis()* també cal dir que retorna un enter sense signe de 32 bits (2^{16} valors), que equival a:

2^{16} valors → 4294967296 mil·lisegons → 4294967 segons → 71583 minuts → 1193 hores → 49,7 dies

Per tant, el valor que retorna la funció no es desbordarà fins al cap de 49 dies i mig sense aturar el microcontrolador, no tindrem limitacions en aquest sentit.

El fet de guardar el recorregut a la placa receptora és perquè així guardem estrictament les dades on hi ha un canvi d'estat, sinó els vectors ocuparien moltes caselles de memòria innecessàriament. També s'estalvia l'ús de la transmissió que s'hauria de fer si es guardessin a la transmissora.

Carregar un recorregut

Transmissor

Abans hem comentat que el vehicle sempre guarda el recorregut, fins que l'usuari dóna l'ordre de començar a reproduir-lo. Aquesta forma serà mitjançant un dels polsadors que es feien servir per al control del vehicle i que s'han canviat per un acceleròmetre en l'apartat (5.4 Control acceleròmetre). Quan la persona prem el polsador, el transmissor enviarà un paquet al receptor amb un estat de càrrega, que li direm *'v'*. El període de temps que estiguem reproduint un recorregut (en estat *'v'*), la placa transmissora no enviarà cap paquet a la receptora, a no ser que sigui el d'aturar la reproducció. Aquest últim es genera tornant a prémer el polsador i aquest cop enviarem al receptor un estat d'aturar la càrrega, li direm *'w'*. A partir d'aquí el transmissor ja tornarà a enviar els estats calculats amb les dades de l'acceleròmetre.

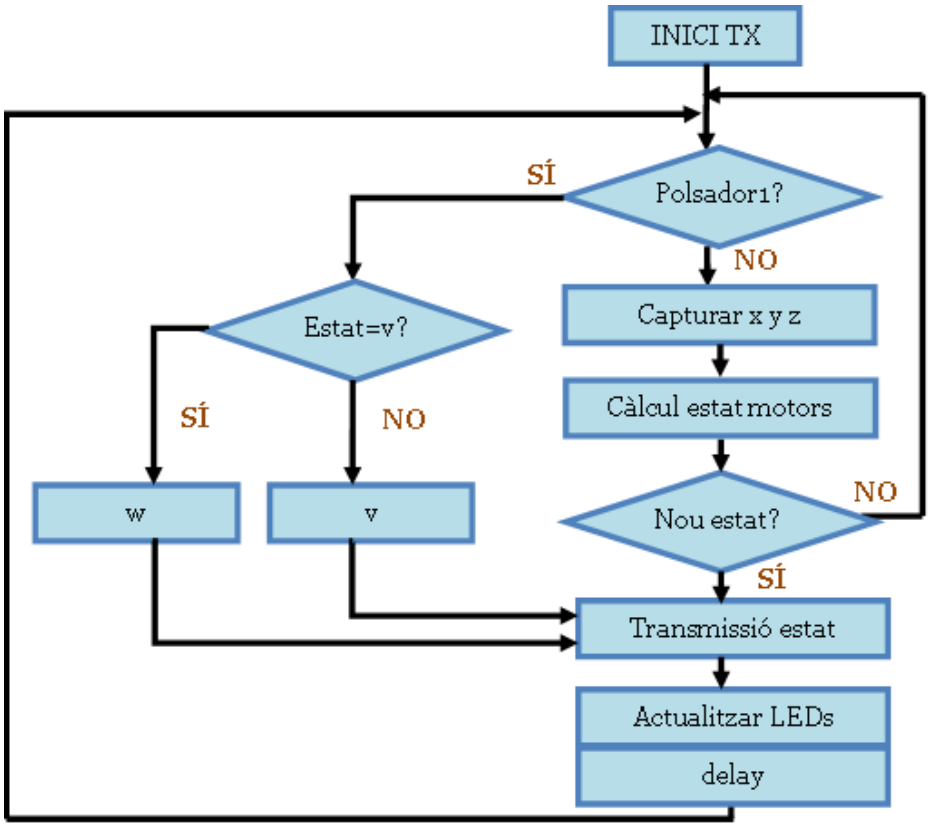


Figura 44: Diagrama de flux de la placa transmissora per carregar un recorregut

Cal tenir en compte que si ens trobem a l'estat de reproducció *'v'*, no enviarem estats dels motors calculats per l'acceleròmetre. Per això, dins la rutina *canviEstat()* afegim el següent cas:

```
case 'v':
     break;
```

Això ens garanteix que l'estat *'v'* no es modificarà mai per la posició de l'acceleròmetre. Sempre serà un estat repetit que no s'enviarà a la placa receptora.

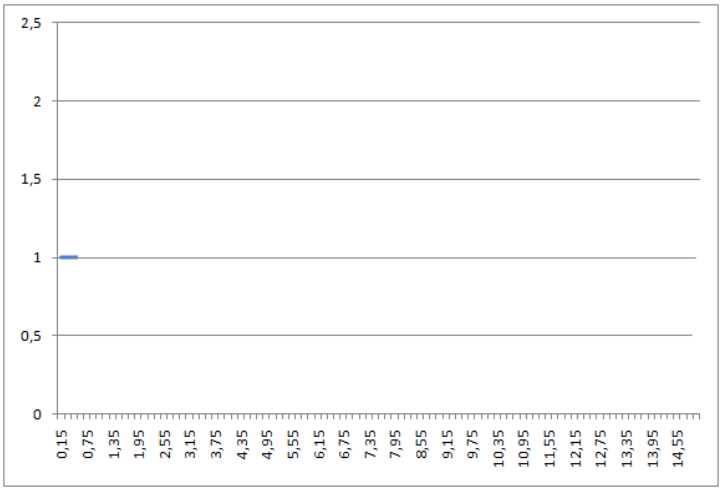
Receptor

Ara ja sabem que quan es rebi a la placa receptora un paquet amb el caràcter *v*, hem de començar a carregar el recorregut guardat. Quan arribi el paquet *w* podrem aturar la reproducció per esborrar el recorregut guardat, seguir el funcionament estàndard i anar guardant el nou circuit.

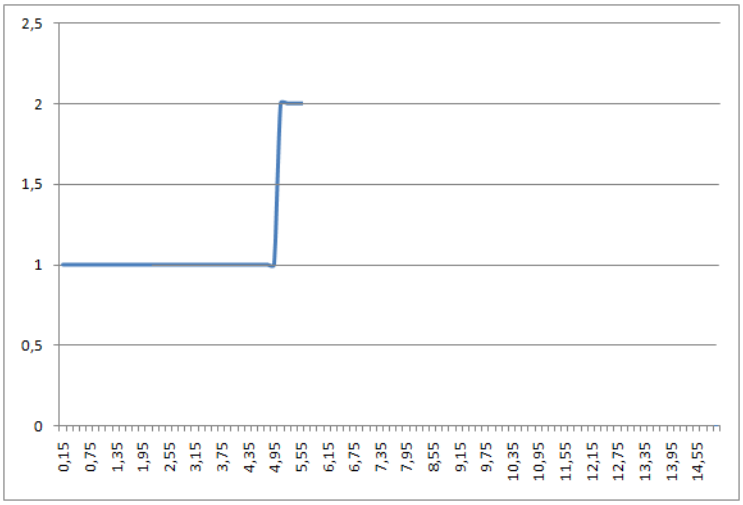
Per a reproduir el recorregut guardat, treballarem amb els vectors *st[]* i *time[]*, l'índex *i* i una nova variable *j* que indicarà el final dels vectors. Ho fem seguint el procés que s'indica a continuació, suposant que tenim guardats els següents vectors:

st	**1**	**2**	**1**
time	10	5	7

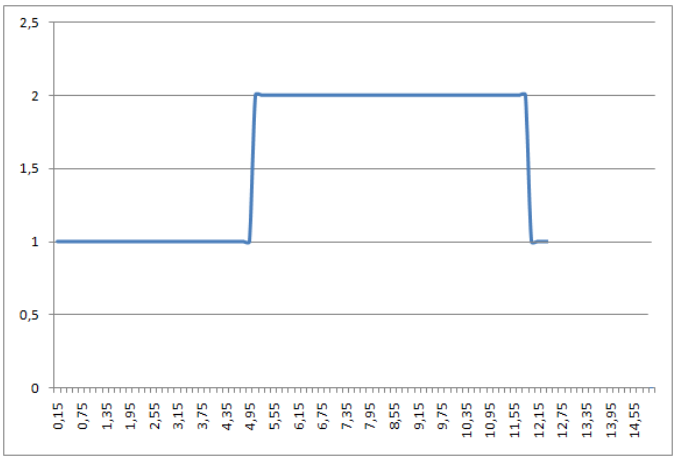
- En el moment de rebre l'estat *v* carreguem l'estat guardat de la posició 0 d'*st[]* directament. No esperem el temps que tenim guardat a *time[0]* perquè l'usuari ha ordenat carregar el vector, no cal que el vehicle estigui 10 segons aturat.



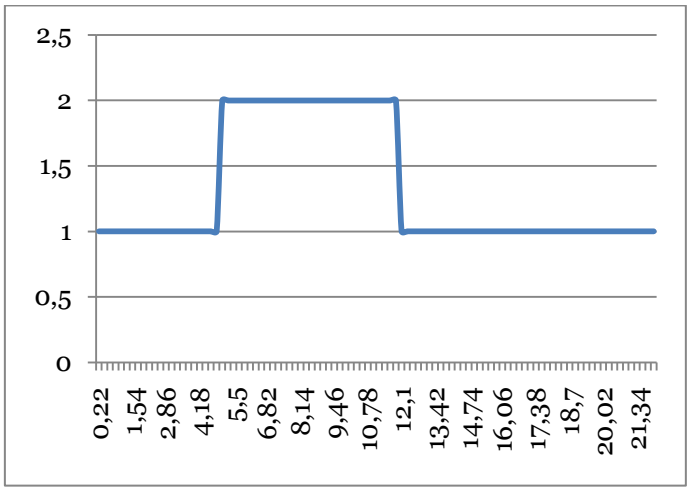
- Esperem el temps transcorregut entre el primer i el segon canvi d'estat *time[1]* i carreguem el segon estat, *st[1]*.



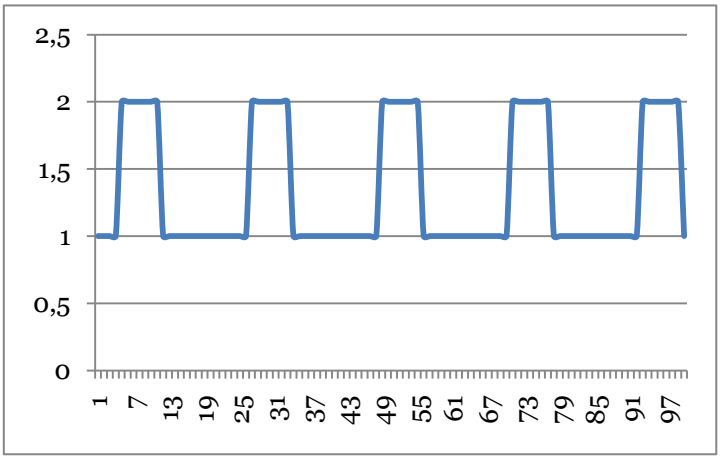
- Esperem el temps transcorregut entre el segon i el tercer canvi d'estat *time[2]* i carreguem el tercer estat, *st[2].* Realitzem aquesta operació fins que arribem a la última posició dels vectors.



- Un cop arribat al final del vector, tornem a començar des del inici. Aquest cop, per a carregar la primer posició, *st[0]*, sí haurem d'esperar el temps corresponent *time[0]*.



- I així successivament, fins que l'usuari decideixi aturar la càrrega.



Sabent el funcionament, ja podem implementar el codi al receptor i el diagrama de flux que el representa:

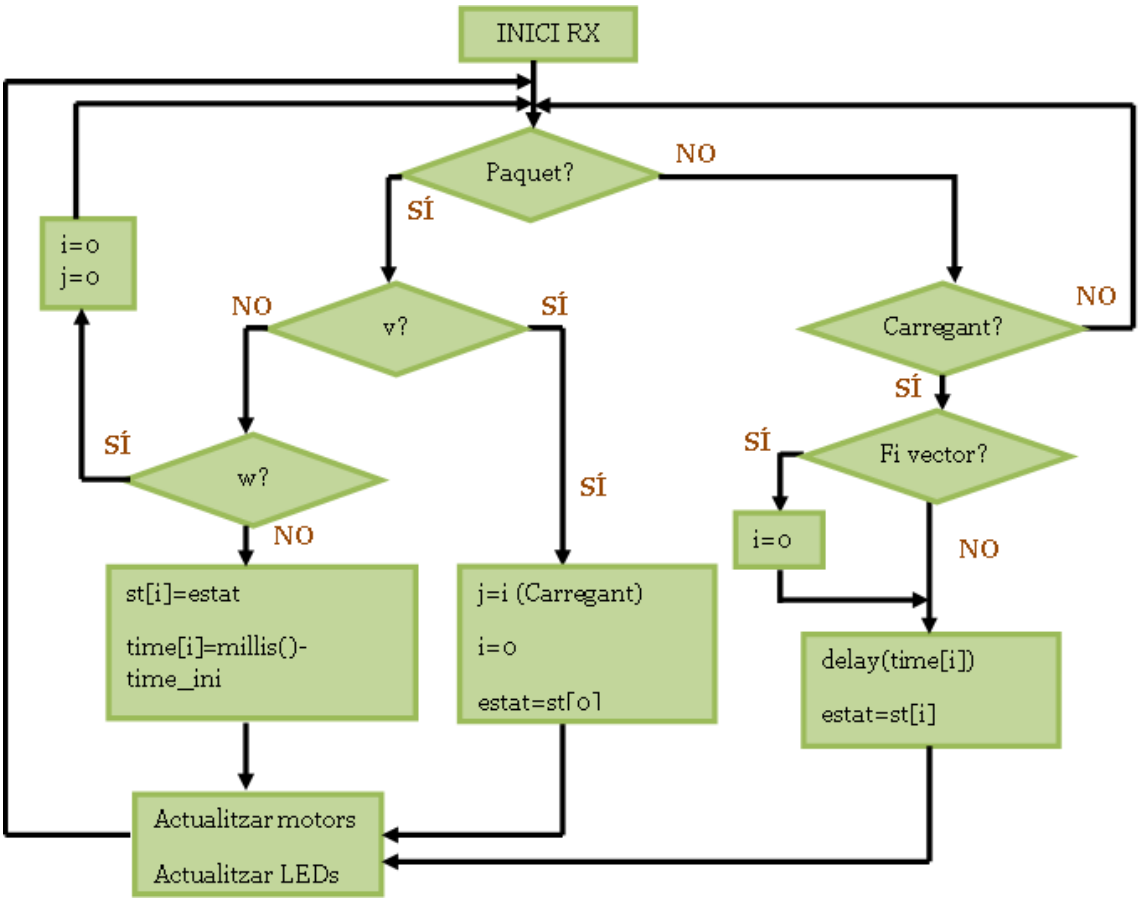


Figura 45: Diagrama de flux de la placa receptora per carregar un recorregut

Veiem que hi haurà un petit decalatge entre els estats del recorregut inicial i les repliques, que vindrà donat pels cicles de rellotge que necessita el microcontrolador des que s'actualitza l'estat fins que es comença el delay del següent. No obstant, aquest decalatge serà mínim, no s'ha de notar a nivell de l'usuari.

5.6 Telemetria

En aquesta darrera ampliació volem poder recollir dades sobre tot el recorregut que ha dut el vehicle durant una sessió, i poder-les exportar a una eina o entorn que ens permeti fer-ne un tractament.

Per començar, sabem que per extreure la informació del recorregut haurem de connectar la placa encarregada de guardar-la a l'ordinador, i per tant el més obvi és que sigui el comandament, la placa transmissora. Així també es podrà aprofitar la connexió USB per a alimentar-la durant tota la sessió, tal i com es mostra en la següent imatge.

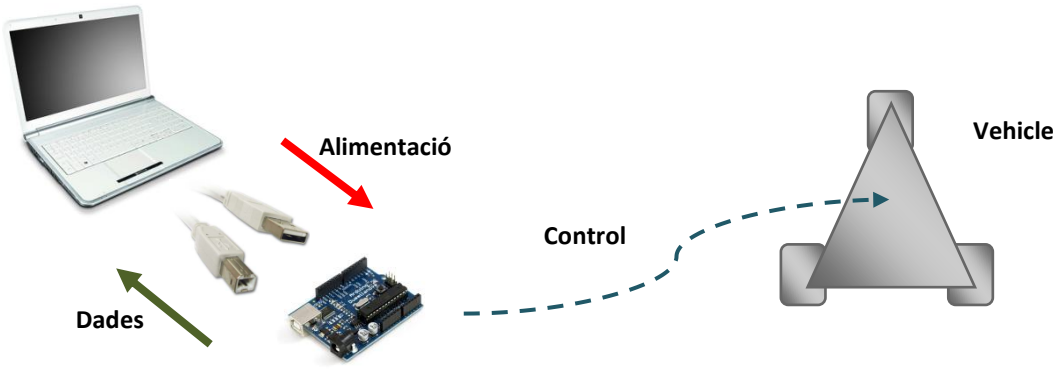


Figura 46: Esquema de connexions de la placa transmissora

Farem servir el mateix mètode que en el punt anterior (5.5 Reproducció d'un recorregut), guardarem dades en el moments on hi hagi un canvi d'estat, i els vectors seran *st[]* per als estats i *time[]* per als instants de temps. La diferència amb la placa receptora és que guardarem absolutament tots els canvis d'estat i que en comptes de diferències de temps, sempre guardem el temps total transcorregut des del inici de la sessió.

- Estats generats per l'usuari controlant el vehicle: Els guardarem directament, després de fer-ne la transmissió a la placa receptora. El temps guardat és el temps que ha passat des del principi del programa. L' increment de la variable k s'explica en el següent punt.

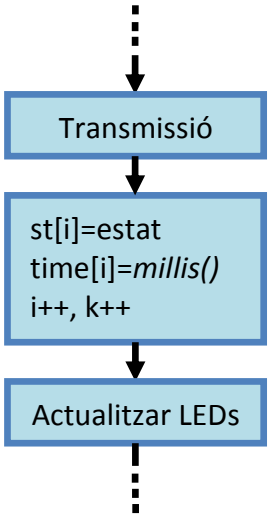


Figura 47: Diagrama de flux per guardar a la placa transmissora els estats enviats

- Estats generats al reproduït un recorregut: Al ser canvis d'estats que es generen a la placa receptora sense cap mena de transmissió, els haurem de simular.

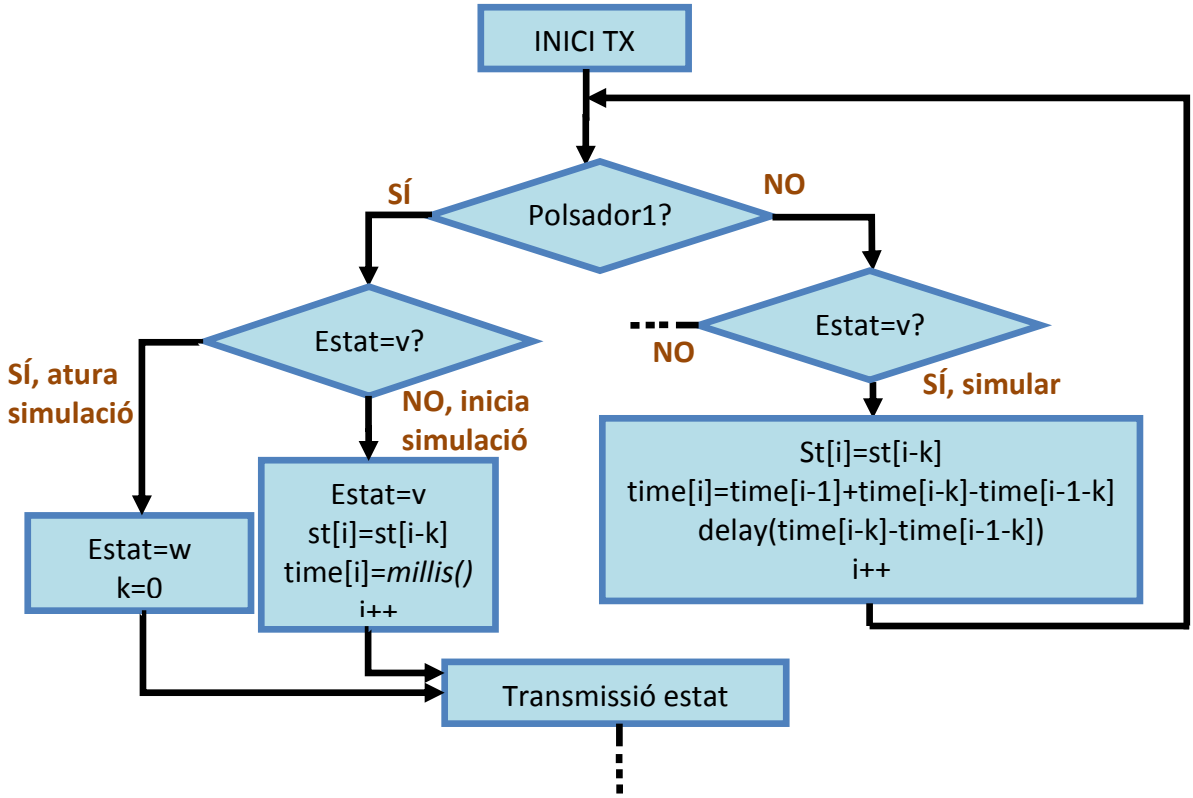


Figura 48: Diagrama de flux per simular i guardar el recorregut reproduït

Ja sabem com guardar el recorregut total de la sessió, ara s'ha de pensar en com exportar-lo a una altra plataforma.

Farem servir la consola que integra el software de desenvolupament d'Arduino. La idea, suposant que la placa ja està connectada a l'ordinador, és que prement un polsador situat a la placa s'imprimeixin al monitor els dos vectors formant una taula. La taula es podrà copiar i enganxar en un programa de full de càlcul on es poden fer una multitud d'operacions sobre les dades.

El polsador per exportar s'haurà de prémer un cop finalitzada la sessió perquè, tal i com s'explica a l'apartat (3.2 Recursos per a desenvolupar el nou prototip), la placa no pot transmetre dades a través del mòdul ZigBee i per la connexió USB alhora.

El funcionament és, que un cop aturat el vehicle, es canvia el jumper pertinent i es prem el botó d'exportar. Per això el codi font que implementa aquesta funció és molt senzill, consisteix en afegir una etapa al principi del codi que ja tenim, que avaluï aquest segon polsador:

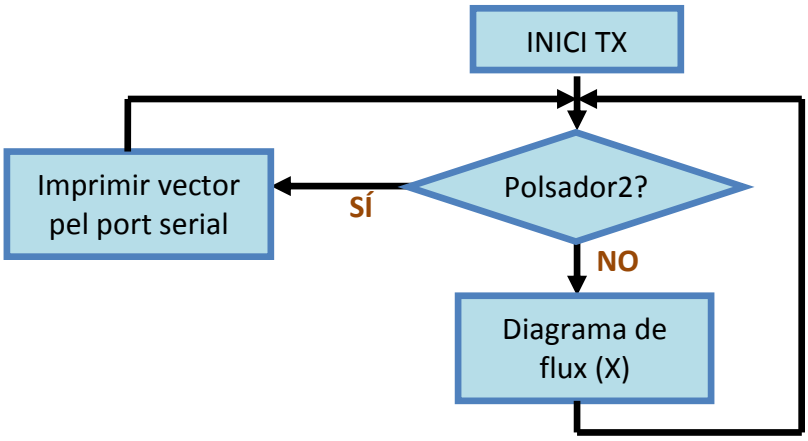


Figura 49: Diagrama de flux de l'exportació

A la consola veurem el següent text que podrem seleccionar, copiar, i enganxar al nostre full de càlcul:

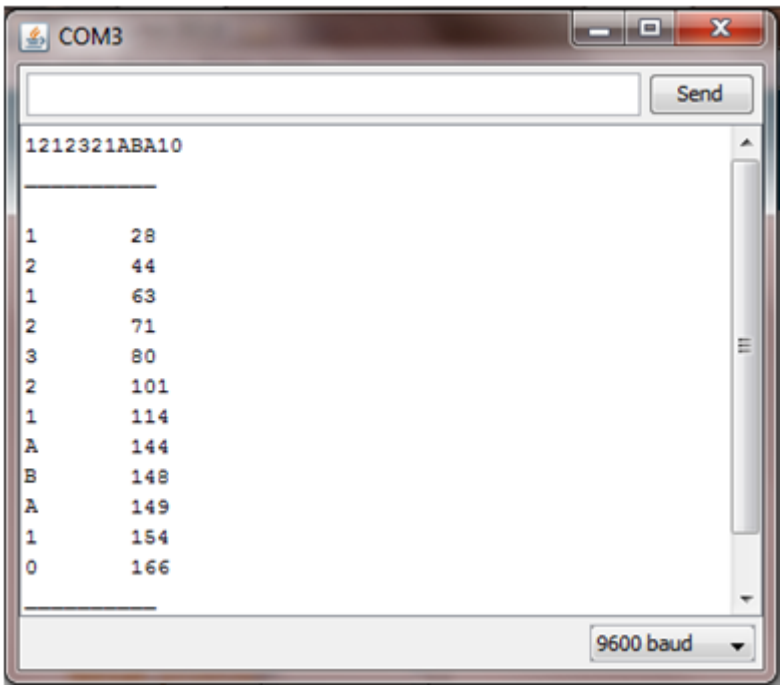


Figura 50: Captura de pantalla de la consola en el moment de l'exportació

Com es pot veure a la imatge, imprimim la taula entre dues línies que defineixen els seus límits. Així visualment és molt més ràpid de seleccionar i copiar.

Entre els diversos programes d'aquest tipus (OpenOffice Calc, Microsoft Excel, iWork Numbers...) ens quedem amb Microsoft Excel. És el més utilitzat i el més estès, encara que l' OpenOffice seria una molt bona alternativa ja que a més és programari lliure i segueix la mateixa filosofia d'Arduino.

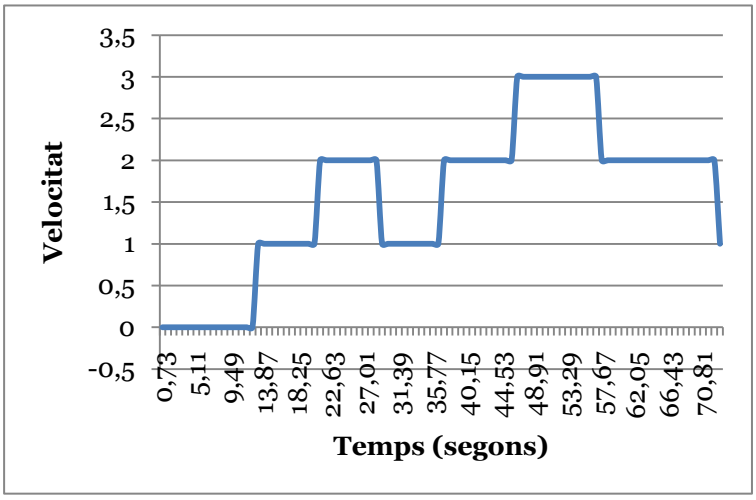
Creem un full de càlcul a mode d'exemple, que consisteix en fer una gràfica de la velocitat i conté els següents camps:

- De la taula que enganxem, en crea una nova taula de 100 mostres de temps corresponents a fer un escalat de la mostra. També converteix el valor de l'estat a velocitat. Per exemple, el tram corresponent a l'estat 2 (taula esquerra) es converteix en el següent (taula dreta):

Estat	Instant
A	21
2	21
1	29

Velocitat	Instant
1	20,44
2	21,17
2	21,9
2	22,63
2	23,36
2	24,09
2	24,82
2	25,55
2	26,28
2	27,01
2	27,74
2	28,47
1	29,2

- Un gràfic on es mostra la velocitat del prototip al llarg del temps, generat a partir de la taula de 100 mostres.



- Una casella on es calcula la velocitat simbòlica mitja que ha dut el vehicle.

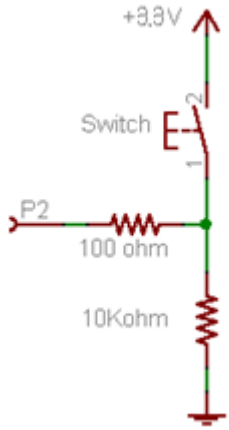
Velocitat mitjana	1,35

5.7 Muntatge del prototip i proves

En aquest apartat anirem descrivint les proves que es van realitzant conforme es va construint el prototip, i que van verificant que cadascuna de les millores s'ha implementat correctament.

Millora del codi font

Donat que les plaques Duemilanove no incorporen polsadors a part del de reset, per a comprovar la millora del codi font necessitarem construir un pad de 4 polsadors. Ho fem en una placa perforada i el funcionament de cada polsador és el següent:



Tal i com es veu a l'esquemàtic de l'esquerra, quan el polsador no està premut el pin 2 de la placa connecta directament a terra, per tant la lectura d'aquesta entrada serà un 0 lògic. Si el polsador s'activa, tindrem un divisor de tensió de 100Ω i 10KΩ, i es pot considerar que tota la tensió cau a la resistència de 10KΩ. Per tant el pin 2 estarà connectat a 3.3V a través d'una resistència de 100Ω, per on fluirà una intensitat de 33mA, que no sobrepassa el màxim (50mA) i farà que la lectura del pin sigui un 1 lògic.

Figura 51: Esquemàtic polsador

Després connectem ambdues plaques a l'ordinador tal i com es mostra a la imatge.

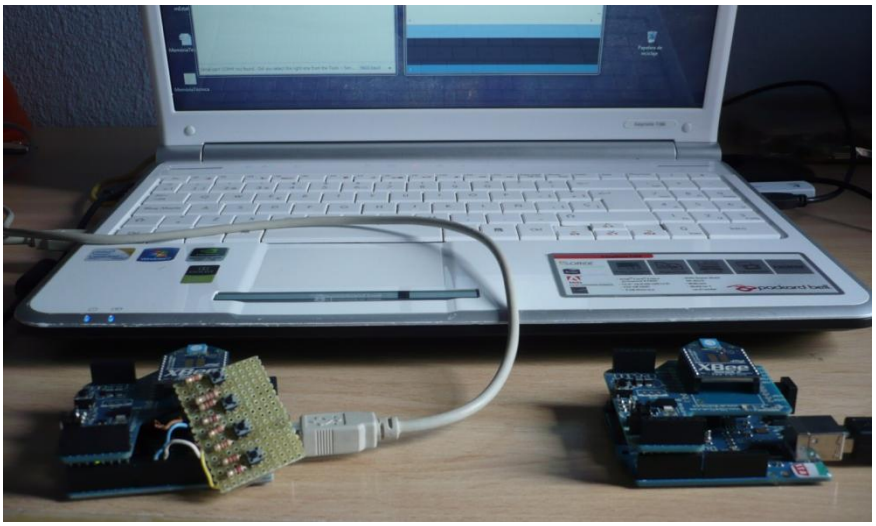
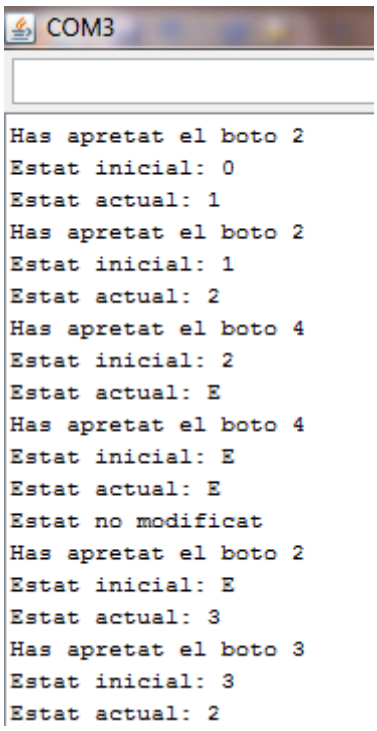
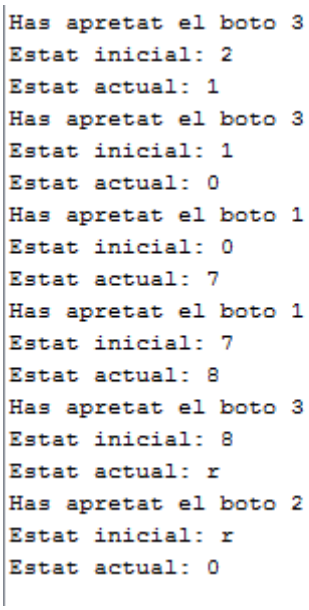


Figura 52: Test de la millora del codi font - Connexió de les plaques

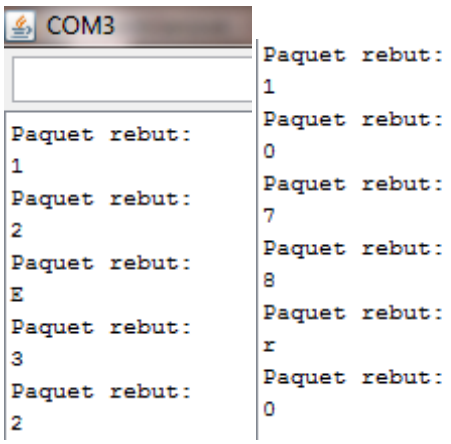
Farem dues proves. La primera consistirà en avaluar que la màquina d'estats funciona correctament i que només es transmeten els nous estats. Per això farem servir la placa transmissora que anirà imprimint al monitor l'execució del programa mitjançant comandes introduïdes per nosaltres del tipus *Serial.print()*.





La captura de pantalla ens mostra les dades que s'han anant enviant al monitor. Visualitzem el polsador premut, l'estat inicial, i l'estat actual. Si anem a mirar el diagrama de la màquina d'estats (Figura 24) podem comprovar que el funcionament és el correcte. També s'imprimeix una cadena de text quan l'estat no s'ha modificat pel fet de prémer un polsador, i per tant, no s'ha enviat a la placa receptora. És la funcionalitat de *blink* i també verifiquem que funciona bé.

La segona prova consisteix en què la placa receptora vagi imprimint pel monitor la informació que rep de la placa transmissora, a través del mòdul XBee. Així sabrem que la comunicació s'estableix correctament i que es reben els paquets corresponents.



Efectivament, tal i com es mostra a la traça del monitor, la placa receptora ha rebut els paquets enviats per la transmissora, i només aquells estats que no són repetits.

Marxa enrere

Per comprovar que els ponts H que hem construït funcionen correctament, muntem una plataforma amb Meccano que inclou els dos motors i els engranatges que els uneixen amb les rodes. A la seva part superior enganxem una protoboard on realitzarem les diferents connexions, tal i com es mostra a la imatge.

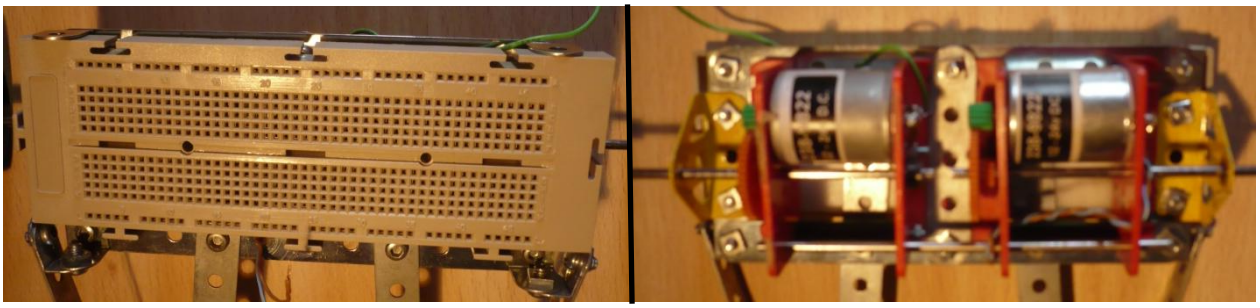


Figura 53: Plataforma motors i protoboard

Tot seguit connectem tots els components relatius a un motor: Un pont H, la bateria i la placa Duemilanove. La roda que farà girar aquest motor queda lliure, sortint de la taula, com a la següent fotografia:

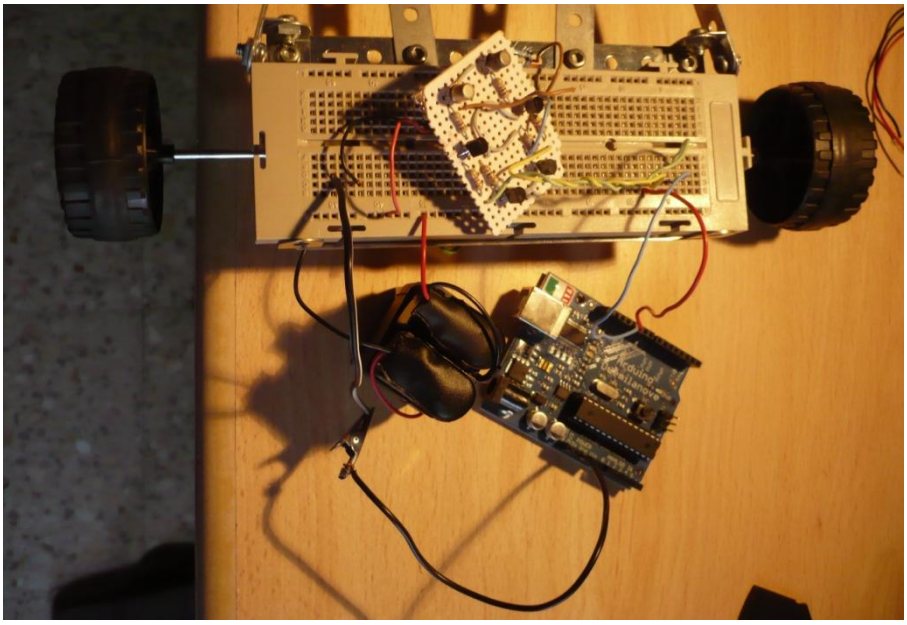


Figura 54: Muntatge per a la prova de la marxa enrere

Carreguem a la placa un codi font que fa activa les sortides que han de fer que el motor giri cap a un sentit i cap a l'altre durant una determinada estona. Observant la roda veiem que funciona correctament. Després fem el mateix experiment amb l'altre pont H i també és efectuat amb èxit.

Actualització motors i LEDs

En aquest cas la prova es realitza amb el mateix muntatge que per a l'anterior. La única diferència és que a la placa Duemilanove se li carrega un programa que activa les sortides per fer que el motor avanci en diferents velocitats durant una estona. També s'afegeixen 4 LEDs per a veure representada la velocitat. A la següent imatge podem veure aquests LEDs que hem incorporat.

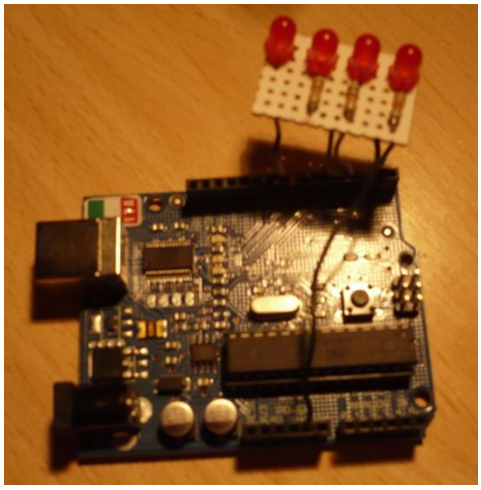
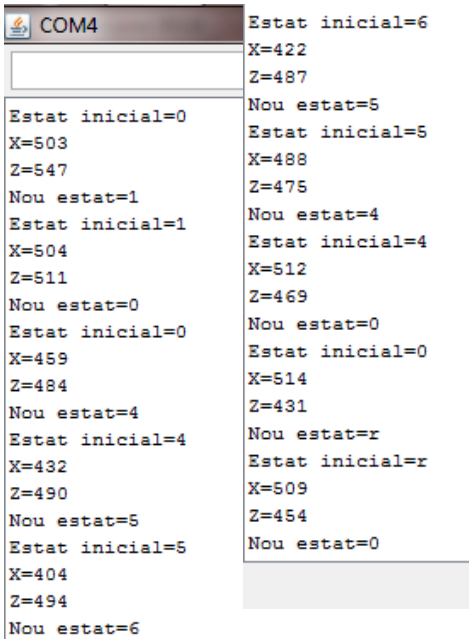


Figura 55: Dispositiu de LEDs connectat a la Duemilanove

Fem la prova i observem que la roda es comporta com havíem previst. Així doncs, donem per vàlida la implementació de la millora.

Control acceleròmetre

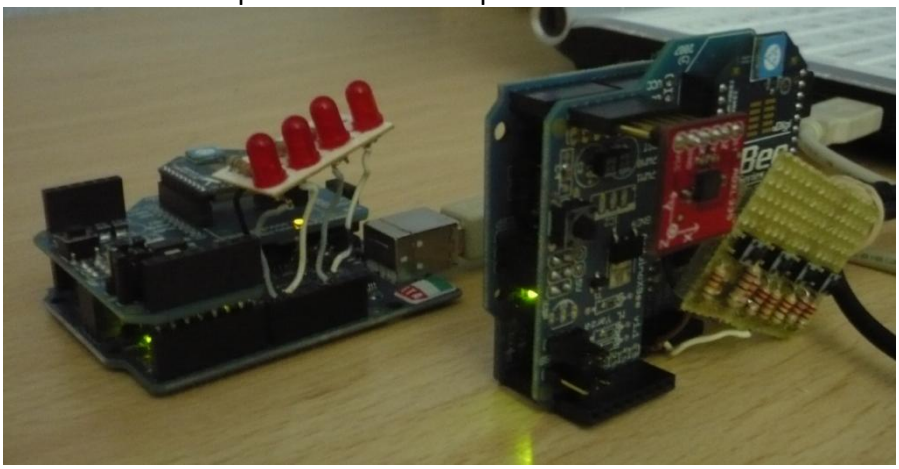
Demostrarem que el vehicle es controla segons la posició del comandament col·locant l'acceleròmetre a la placa transmissora, i fent la mateixa operació que amb els polsadors, anant imprimint pel monitor els canvis d'estat.



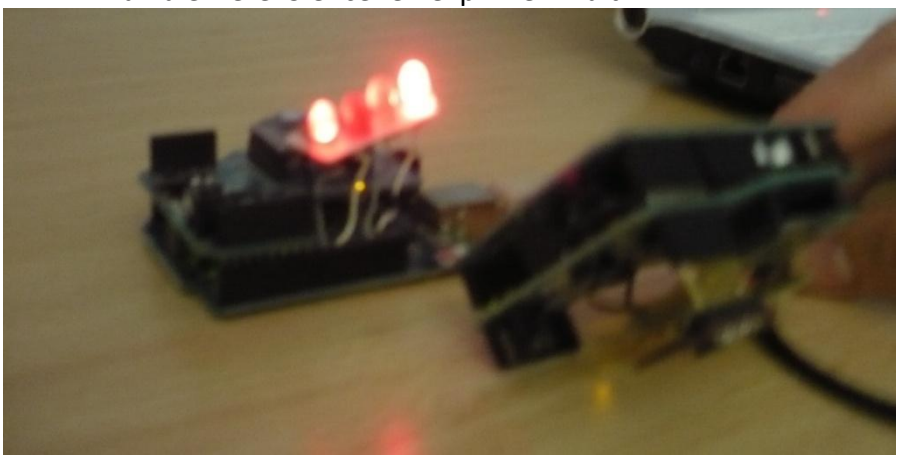
Si comprovem les transicions amb les de la màquina d'estats, es corresponen. Per tant ja sabem que la ordre que es transmet a la placa receptora serà correcta. Hem de tenir en compte en aquest punt, que els valors de l'eix x i z poden variar respecte la màquina d'estats inicial, perquè s'han fet una sèrie d'ajustos.

Ara comprovarem que la ordre generada a partir de l'acceleròmetre arribi al receptor correctament. Ho farem observant els LEDs de la placa receptora, que s'encenen en funció de la velocitat. Comprovant això ja sabem que els motors s'actualitzaran correctament, perquè s'ha testejat a l'apartat anterior.

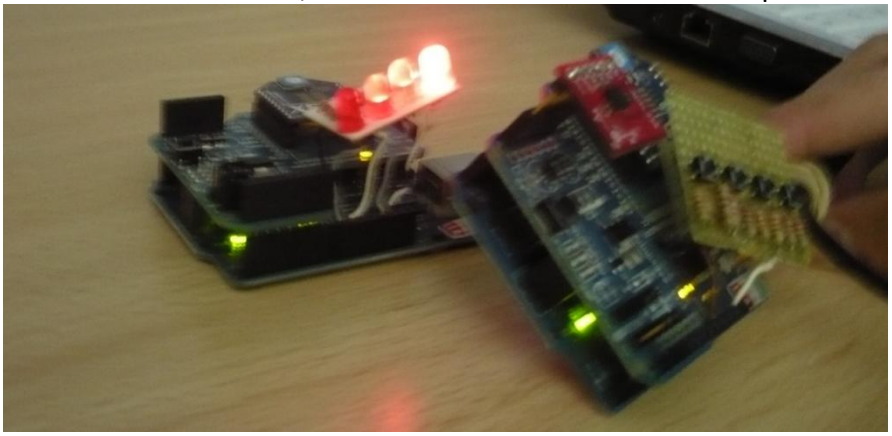
- Placa en repós. No s'encén cap LED:



- Marxa enrere. S'encenen el primer i l'últim LED:



- Marxa endavant, màxima velocitat. S'encenen els 3 primers LEDs:



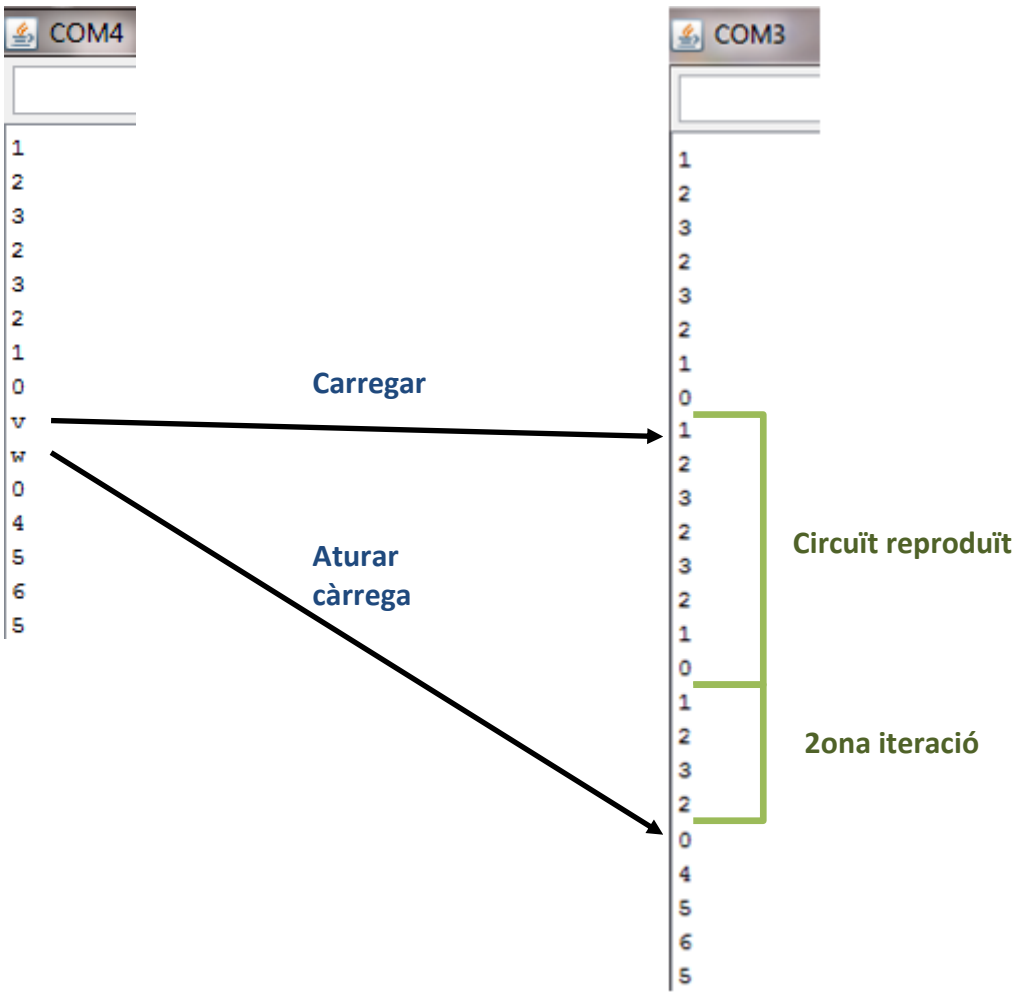
En tots els casos la visualització dels LEDs de la placa receptora ha estat la esperada, per tant, el control amb l'acceleròmetre funciona correctament.

Reproducció d'un recorregut

La verificació d'aquest apartat consisteix en veure com l'últim recorregut dut a terme amb el vehicle es reprodueix al prémer el polsador adequat. Connectem les dues plaques com a la figura 52 amb la diferència que la placa transmissora tindrà l'acceleròmetre incorporat.

En primer lloc imprimim al monitor els estats que s'envien des de la placa transmissora cap a la receptora, després imprimim pel monitor els canvis d'estats que tenen lloc dins de la placa receptora i comparem les dues captures:

Transmissor **Receptor**

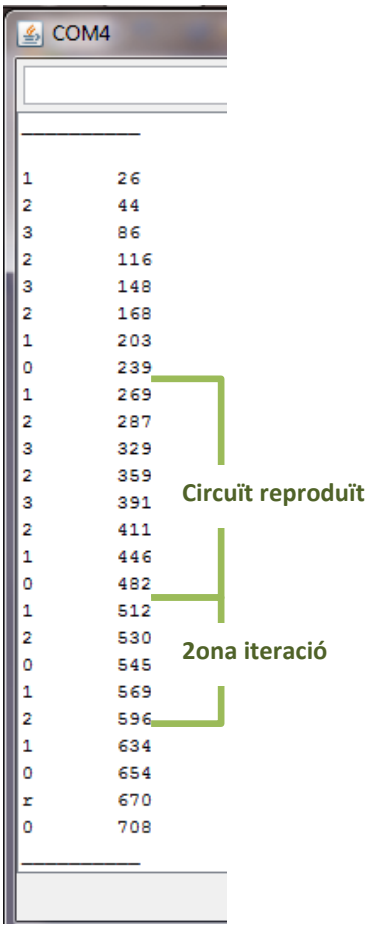


Comprovem que funciona correctament. L'aplicació va guardant els canvis d'estat fins que s'envia l'estat 'v', que comença a reproduir-los. Quan s'envia l'estat 'w' deixa de reproduir i el nou circuit que fet es guardarà sobre l'antic. És a dir, que es sobreescriurà.

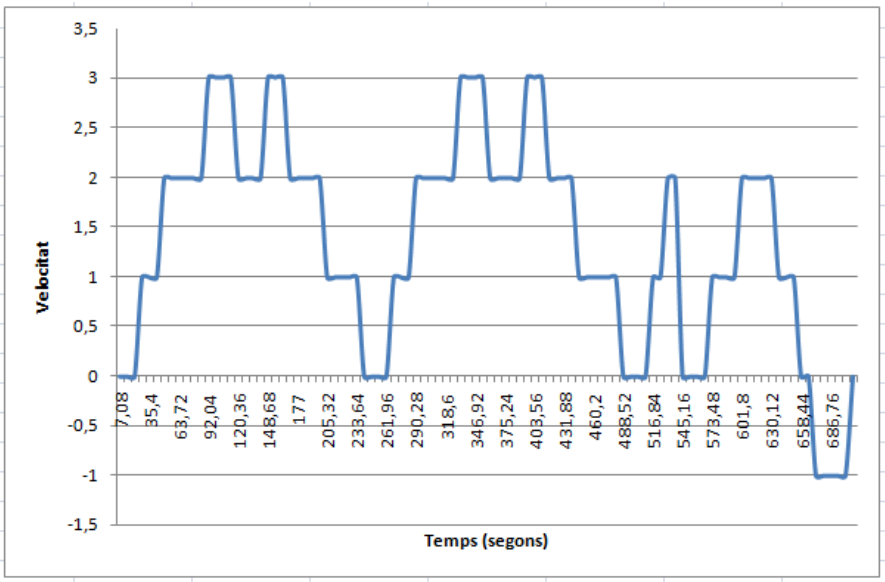
Telemetria

Com a prova per veure si la millora de la telemetria ha estat correctament implementada, farem un recorregut similar al de la comprovació anterior, i seguidament l'exportarem. Enganxem al full de càlcul la taula obtinguda i aquest és el resultat de dita exportació:

Captura del monitor



Dades obtingudes



Velocitat mitja	1,37

Mirem que les dades exportades es corresponen amb el recorregut fet realment (els temps estan en dècimes de segon). Les dades dels estats simulats al fer una reproducció, també són les correctes. Per últim, observem la representació de la velocitat en funció del temps al full de càlcul i no es troba cap error, comprovem també que l'estat *r*, per exemple, s'ha convertit en velocitat '*-1*'. La velocitat mitja també té sentit, 1,37.

6 Conclusions

Un cop finalitzat el projecte i verificat que les millores aplicades funcionen adequadament, es fa una repassada dels objectius marcats inicialment i es valora en quin grau s'han assolit:

- S'ha reproduït i millorat el codi font de les plaques del projecte inicial.
- S'ha revisat i solucionat el problema amb la marxa enrere del prototip inicial.
- S'ha adaptat i millorat el sistema d'actuació sobre els motors i sobre els LEDs. Aquesta adaptació és deguda a les noves plaques utilitzades i al nou disseny del codi font base.
- El vehicle és controlat a partir de moviments (girs i inclinacions) del comandament.
- Es pot reproduir l'últim recorregut fet amb el prototip a voluntat de l'usuari.
- Es pot exportar la informació d'una sessió cap a l'ordinador. Es crea un full de càlcul com exemple per al tractament d'aquesta informació.
- No s'ha dut a terme un sistema de detecció de xocs i reacció automàtica a ells. Ha estat per limitacions de temps ocasionades per una incidència amb les plaques.

Fent el recompte veiem que un 85% dels objectius han estat acomplerts amb èxit. Això, juntament amb la capacitat de superar un problema que ha comportat un canvi d'entorn i de plaques amb el projecte en marxa, fa que l'experiència personal sigui totalment satisfactòria. No obstant, també s'evidencia la importància d'incloure una gestió dels riscos a la planificació del projecte, ja que si s'hagués tractat d'un projecte amb un abast real hagués estat un problema el fet que una incidència puntual suposi la seva aturada. En aquest sentit, hi ha hagut un aprenentatge del pes real que pot tenir la planificació i la gestió del risc en un projecte.

El fet d'haver treballat en dos entorns diferents com són Freescale i Arduino ha permès poder-los comparar i arribar a veure els avantatges que suposa treballar amb plataformes lliures, en aquest cas, Arduino. L'exemple més clar és alhora de familiaritzar-se amb els entorns per tal d'agafar-hi una certa pràctica abans de començar a treballar: Amb les plaques Duemilanove es van trobar directament al site d'Arduino tota mena de recursos per a iniciar-se, com manuals, exemples, tutorials, i fins i tot un sistema de fòrums on hi ha una activitat molt dinàmica. En canvi amb les plaques 13192 de Freescale, sí que es va aconseguir trobar aquesta informació, però havent de flanquejar diferents obstacles, com haver d'adquirir el producte directament, realitzar un registre a la pàgina web, en definitiva identificar-se com a client. També es percep una orientació molt més tècnica en tota la documentació de Freescale. En Arduino s'utilitza un llenguatge més entenedor de cara a usuaris amb perfils no tant tècnics. Tot això està relacionat amb el hardware lliure i el tipus de clients cap a on s'orienta cadascun d'aquests productes.

Propostes de millora

Dins de l'apartat de les conclusions també comentem les diferents propostes d'ampliació i de millora que es poden aplicar al prototip, un cop finalitzat aquest projecte.

En un primer lloc tenim aquelles ampliacions que hem descartat des d'un principi i la que no s'ha pogut dur a terme. A aquestes s'hi sumen aquelles que han anat apareixent al llarg del desenvolupament d'aquest nou prototip. Són les següents:

- **Control del prototip mitjançant un ordinador**
  La idea bàsica seria anar enviant dades des del monitor o la consola del PC, que anessin controlant el vehicle. Una idea més avançada seria dissenyar una aplicació que mitjançant una interfície gràfica permetés dibuixar un recorregut que després es traspassés al vehicle.

- **Disseny d'una PCB que integri tota la part electrònica del vehicle**
  Agrupar en una sola placa tots els dispositius electrònics que es fan servir en el prototip. S'hauria de dissenyar una PCB per al receptor i una altra per al transmissor.

- **Regulador de velocitat**
  Fer que el prototip mantingui una regulació de la seva velocitat a petició de l'usuari. Això implica que el vehicle ha de conèixer la seva velocitat actual, i que hi reaccioni incrementant o reduint la potència entregada als motors.

- **Gestió de la bateria**
  El prototip actual no disposa d'un sistema de control de les bateries que alimenten els diferents dispositius, l'usuari detecta que s'han esgotat quan veu que el prototip ja no funciona.

- **Detecció i reacció a xocs**
  Al produir-se un xoc en el vehicle, que el comandament ens avisi de la seva gravetat i que el receptor realitzi un moviment contrari al xoc o simplement aturi el vehicle, com a mesura de prevenció.

- **Millora de la telemetria**
  Una ampliació de la funcionalitat que permet exportar les dades del recorregut d'una sessió, seria que aquestes dades s'anessin enviant a l'ordinador de forma síncrona. Sense haver d'esperar a acabar la sessió. També es pot mirar la forma d'evitar que l'usuari faci més accions com per exemple copiar i enganxar una taula del monitor al full de càlcul.

- **Millora del control amb l'acceleròmetre**
  Actualment es pot controlar la velocitat i direcció del vehicle amb l'acceleròmetre, però només es fan servir dos eixos dels tres que mesura. Amb aquest eix es podrien dur a terme altres accions sobre el prototip.

7 Bibliografia

Llibres

[1] Fredrick M. Cady (2008): "**Sofware and Hardware Engineering: Assembly and C Programming for the Freescale HCS12 Microcontroller, Second Edition**". Ed Oxford University Press.

Documentació tècnica en format PDF

[1] Freescale Semiconductor: Quick Start Guide – BeeKit Wireless Connectivity Toolkit (BKWCTKQSG)

[2] Motorola: MC9S08GT60 Data Sheet

[3] Freescale Semiconductor: Sensor Applications Reference Design (SARD) User's Guide

[4] Freescale Semiconductor: MC1319x, MC1320x, and MC1321x Demonstration Operation (AN3231)

[5] Philips: 2N2222 Data Sheet

[6] Philips: 2N2907 Data Sheet

[7] Analog Devices: ADXL335 Data Sheet

Apunts d'assignatures

[1] Fonaments de programació

[2] Disseny de Sistemes Electrònics Basats en Microprocessadors.

Pàgines web

[1] **Arduino Tutorial** (2010)
http://www.ladyada.net/learn/arduino/index.html

[3] **Getting Arduino to talk wirelessly: XBee** (2009)
http://blog.fupps.com/2009/06/07/getting-arduino-to-talk-wirelessly-xbee/

[4] **Arduino HomePage** (2010)
http://www.arduino.cc/

[5] **Wikipedia** (2010)
http://es.wikipedia.org/

8 Annex

Juntament amb la memòria del projecte, fem l'entrega d'un DVD que conté els arxius amb el codi font de cadascuna de les plaques que intervenen en el prototip. Aquests arxius estan desats en un format *.pde llegible per al software de desenvolupament d'Arduino. Cal tenir en compte que han estat creats amb la versió 0018 d'aquest programari.

En el cas que l'interessat no disposi del programa necessari per obrir aquests arxius, adjuntem una còpia del codi en versió PDF. No obstant, aquesta versió pot presentar problemes si s'intenta copiar i enganxar a l'entorn Arduino, ja que hi ha alguns caràcters que no es conserven al fer la conversió a PDF.

Per últim, el DVD també inclou una còpia en format digital de la memòria.

Resum

En el projecte *Ampliació i millora d'un vehicle teledirigit* s'ha dut a terme l'ampliació d'un prototip de vehicle ràdio controlat fent servir dues plaques Arduino Duemilanove. Una es situa en el comandament i l'altra en el vehicle i controlen el comportament dels dos dispositius. Es transmet la informació necessària entre elles a través de dos mòduls XBee que posteriorment se'ls hi incorpora. Les plaques fetes servir en el prototip inicial eren unes SARD-13192 de Freescale i el primer que es fa en aquest sentit és una revisió del codi font utilitzat i l'adaptació a les plaques Arduino. Un acceleròmetre ADXL335 que s'incorpora a una de les plaques permet que el prototip es pugui controlar segons la posició del comandament. A més, un cop finalitzat el nou prototip és capaç de desplaçar-se endavant i endarrere, girar aturat i en moviment, i a diferents velocitats que es representen en tot moment en uns LEDs. També guarda l'últim circuit efectuat que es pot reproduir a voluntat de l'usuari, i emmagatzema les dades del recorregut de tota una sessió per exportar a l'ordinador. Per últim s'han dut a terme les proves necessàries per constatar que totes les millores s'han implementat amb èxit.

Resumen

En el proyecto Ampliación y mejora de un vehículo teledirigido se ha llevado a cabo la ampliación de un prototipo de vehículo radio controlado utilizando dos placas Arduino Duemilanove. Una se sitúa en el mando y la otra en el vehículo y controlan el comportamiento de los dos dispositivos. Se transmite la información necesaria entre las placas a través de dos módulos XBee que posteriormente se les incorpora. Las placas utilizadas en el prototipo inicial eran unas SARD-13192 de Freescale y lo primero que se hace es una revisión del código fuente utilizado y la adaptación a las placas Arduino. Un acelerómetro ADXL335 que se incorpora a una de las placas permite que el prototipo se pueda controlar según la posición del mando. Además, una vez finalizado el nuevo prototipo, es capaz de desplazarse hacia adelante y hacia atrás, girar parado y en movimiento, y a diferentes velocidades que se representan en todo momento con unos LEDs. También guarda el último circuito efectuado que se puede reproducir a voluntad del usuario, y almacena los datos del recorrido de toda una sesión para ser exportados al ordenador. Por último se han llevado a cabo las pruebas necesarias para constatar que todas las mejoras se han implementado con éxito.

Abstract

The project Expansion and improvement of a remote-controlled vehicle has made an enlargement of a radio controlled prototype vehicle using two Duemilanove Arduino boards. One is located on the command and the other one in the vehicle and they control the actions of the two devices. All the necessary information is transmitted between them through two XBee modules that were later incorporated. The boards used in the initial prototype were two Freescale SARD-13192 and firstly a revision of the source code used and the adaptation to the Arduino boards are done. An ADXL335 accelerometer attached to one of the boards allows the prototype to be controlled depending on the position of the command. Moreover, when the new prototype is completed, it is able to move forward and backwards, turn when stopped and ongoing at different speeds that are represented all the time in a LEDs. It also records the last circuit made and can be reproduced by the user, and stores the data of the route of a whole session to be exported to the computer. Finally, all the required tests have been made to verify that all the improvements have been successfully implemented.